

**Allen-Bradley**

**IQ Master™  
Version 3.2.4  
for  
IQ-2000 and IQ-5000  
Positioning Drive  
Modules,  
IQ-550 Position  
Control Module,  
and ULTRA Plus™  
Positioning Drive  
Modules**

**Programming Manual**

**Rockwell  
Automation**

---

## Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley® does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley publication SGI-1.1, *Safety Guidelines for the Application, Installation and Maintenance of Solid-State Control* (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

Reproduction of the contents of this copyrighted publication, in whole or part, without written permission of Rockwell Automation®, is prohibited.

Throughout this manual we use notes to make you aware of safety considerations:

---

### ATTENTION



Identifies information about practices or circumstances that can lead to personal injury or death, property damage or economic loss

---

Attention statements help you to:

- identify a hazard
- avoid a hazard
- recognize the consequences

---

### IMPORTANT

Identifies information that is critical for successful application and understanding of the product.

---

Allen-Bradley and IQ-Series are registered trademarks of Rockwell Automation.  
IQ Master and ULTRA Plus are trademarks of Rockwell Automation.  
Microsoft is a registered trademarks of Microsoft Corporation.  
Windows is a trademark of Microsoft Corporation.

# Table of Contents

<b>Preface</b> .....	<b>1</b>
Introduction .....	1
Who Should Use this Manual .....	1
Where to Find Help .....	1
Contents of this Manual .....	2
Related Documentation .....	3
Conventions Used in this Manual .....	3
Using Online Help .....	4
Allen-Bradley Support .....	5
Local Product Support .....	5
Technical Product Assistance .....	5
<b>Introduction to IQ Master</b> .....	<b>3</b>
About the Allen-Bradley IQ Master Manual .....	3
ULTRA Plus or IQ-Series System Overview .....	4
IQ Master .....	6
IQ-550 Position Control Module .....	6
Positioning Drive Modules (PDM) .....	6
PDMs with Integral Power Supply Modules .....	6
PDMs with Separate Power Supply Modules .....	6
Power Supply Module (PSM) .....	6
Motors .....	7
Operator Terminal .....	7
Personality Module (PM) .....	7
Manuals .....	7
Option Cards .....	7

Memory and I/O Expansion Card .....	8
I/O Expansion Card .....	8
Personal Computer (PC) .....	8
Symbols and Conventions .....	8
Minimum Personal Computer Requirements .....	9
IQ Master .....	9
New Features .....	10
New in Version 3.2.4 vs. Version 3.2.3 .....	10
New in Version 3.2 vs. Version 3.0 .....	10
New in Version 3.0 vs. Version 2.1x .....	10
New in Version 2.1 vs. Version 2.0 .....	10
New in Version 2.x vs. Version 1.x .....	10
<b>Quick Start Check List .....</b>	<b>12</b>
Introduction .....	12
Quick Start Check List .....	12
<b>Getting Started with IQ Master .....</b>	<b>15</b>
Hardware and Software Requirements .....	15
Using Windows Without a Mouse - A Quick Review .....	15
Windows Hot Keys .....	15
IQ Master Hot Keys .....	17
Using the IQ Master Setup Program .....	17
The Readme File .....	18
Starting and Quitting IQ Master .....	19
To Start IQ Master .....	19
The IQ Master Screen .....	19
Quitting IQ Master .....	19
On-line Help .....	20
To Start Help: .....	20
Upgrading from Version 1 .....	20
<b>Applying Power for the First Time .....</b>	<b>21</b>
Start-Up Procedure for ULTRA Plus and IQ-2000 Systems .....	21
Start-Up Procedure for ULTRA Plus and IQ-5000 Systems .....	22
Power Supply Module .....	22
Start-Up Procedure for ULTRA Plus or IQ-5000 .....	22
Motor Start-Up Procedure .....	23
Start-Up Procedure for IQ-550 Systems .....	24
Operator Terminal .....	25
Backup Personality Module .....	25
<b>File Menu .....</b>	<b>29</b>
PC Disk Commands .....	30
New .....	30
Open .....	30
Save .....	30

Save As .....	30
<b>IQ Commands .....</b>	<b>30</b>
Open IQ Program .....	30
Save Program to IQ (F7) .....	30
Delete IQ Program .....	31
<b>File Extensions .....</b>	<b>31</b>
<b>Miscellaneous Commands .....</b>	<b>32</b>
Print .....	32
Transfer .....	32
Initialize PM .....	32
Recently Used File List .....	33
Send Mail .....	33
Exit .....	33
<b>Edit Menu .....</b>	<b>34</b>
Undo .....	35
Cutting and Pasting Text—Selecting Text .....	35
Selecting Text .....	35
Clipboard .....	35
Cut (SHIFT+Delete) .....	35
Copy (CTRL+Insert) .....	35
Paste (SHIFT+Insert) .....	35
Finding Text .....	36
Find .....	36
Find Next (F3) .....	36
Find Previous (F4) .....	36
Replace .....	36
The Compiler .....	36
Compiling a Program .....	36
Compile Options .....	37
Compiler Output .....	37
List Files .....	37
Expand Macros .....	37
Program Type .....	37
Debug Information .....	37
<b>Parameter Menu .....</b>	<b>38</b>
Gains/Limits .....	39
System .....	40
Absolute Mode .....	40
Forward Software Travel Limit .....	40
Reverse Software Travel Limit .....	40
Ereturn Position .....	40
Scale .....	41
Scale2 .....	41
Disable on Fault .....	41
Rotation .....	41
Rotation 2 .....	41

Latched Position .....	41
Velocity & Acceleration .....	42
Inputs .....	43
Home Switch Active State .....	44
Pause Switch Active State .....	44
Selectable Inputs .....	44
ADC1 Function .....	44
Debounce Time .....	44
Home to Encoder Index .....	44
Program Select Lines .....	45
Default Run Program .....	45
Outputs .....	46
Error Output when Disabled .....	46
In-Position Mode .....	46
DAC1 Value .....	46
Default Outputs .....	47
Feedback Configuration .....	47
Configuration Menu .....	48
Current Configuration .....	48
Feedback Position Source .....	48
Encoder Input 2 .....	48
Gear Input .....	48
Fkey Set Up .....	48
Mode .....	48
Serial .....	49
Baud Rate .....	49
Parity .....	49
Communications Mode (RS-232C/RS-422) .....	50
Generic Operator Terminal .....	50
Operator Terminal Address .....	50
Variables .....	51
Edit Parameter File .....	51
New .....	51
Open .....	51
<b>Run Menu .....</b>	<b>53</b>
Run Control .....	54
Run Program .....	54
Home .....	54
Pause .....	55
Single Step .....	55
Move .....	55
Stop .....	56
Enable .....	56
Disable .....	56
Reset .....	56
Hardware Reset .....	57
Define Home Position .....	57

Tune .....	57
Auto Tune Velocity and Position Loops .....	58
To Tune the System using Auto Tune .....	58
Manual Tune .....	59
To Tune the Velocity Loop using Manual Tune .....	59
To Tune the Position Loop using Manual Tune .....	60
Extended Debug .....	62
Operational Commands for Debugger .....	62
Go .....	62
Trace .....	63
Auto .....	63
Resume .....	63
Pause .....	63
Abort .....	63
Step Commands for Debugger .....	63
Single Step .....	63
Procedure Step .....	63
Breakpoint Commands and Options for Debugger .....	63
Clear all Breakpoints .....	64
Toggle Breakpoint .....	64
View Active Breakpoints .....	64
Debug Options .....	64
<b>Monitor Menu .....</b>	<b>67</b>
Status .....	68
Inputs/Outputs .....	68
Monitor Variables .....	69
<b>Diagnostics Menu .....</b>	<b>72</b>
Encoder .....	73
DAC1 Output .....	73
Programmable Monitor Output .....	74
Digital Outputs .....	74
Operator Terminal .....	75
Nonvolatile Memory .....	75
Fault History .....	75
Version .....	75
<b>Communications Menu .....</b>	<b>76</b>
Axis Select .....	77
Terminal .....	77
Scrolling Keys .....	77
PC Set Up .....	78
Baud Rate .....	78
Communications Port .....	78
CR (Carriage Return) Translation .....	78
Data Bits .....	78

Stop Bits .....	78
Flow Control .....	78
Parity .....	78
Local Echo .....	79
BRU Emulation .....	79
Send File LF Toggle .....	79
Receive File CR Translation .....	79
<b>Help Menu .....</b>	<b>80</b>
Contents .....	81
How to Use Help .....	81
Context Sensitive Help .....	81
About IQ Master .....	81
<b>Inputs and Outputs .....</b>	<b>85</b>
General Purpose Inputs .....	85
Input Debounce Time .....	86
Assignable Inputs .....	86
Enable (I3) .....	86
Emergency Return (I10) .....	87
Home Command (I6) .....	87
Home Switch (I5) .....	87
Jog Forward (I7) and Jog Reverse (I8) .....	87
When a Jog Input is Turned OFF .....	88
Forward Travel Limit (I1) and Reverse Travel Limit (I2) .....	88
Pause (I9) .....	88
Start (I4) .....	88
Define Home .....	88
Hardware Reset .....	89
X Kill Motion .....	89
Program Select Lines .....	90
Interrupt Inputs .....	90
General Purpose Outputs .....	91
Assignable Outputs .....	92
AtHome (O5) .....	92
Error (O8) .....	92
Home Sequence Complete (O6) .....	92
Program Running (O4) .....	92
In-Position (O7) .....	92
Enabled Output (P3-2) .....	93
Ready Output (P3-1) .....	93
DAC1 Output (P3-6) .....	93
ADC1 Input (P3-5) .....	93
Encoder 2 .....	93
<b>Starting a Program .....</b>	<b>94</b>
Assigned Input .....	95



IQ Master - Run Control .....	95
Function Key Program .....	95
Host Language Commands (Serial) .....	95
<b>Stopping or Suspending a Program .....</b>	<b>96</b>
Assigned Inputs .....	96
Host Language Commands (Serial) .....	97
Program Instructions .....	97
Fault Conditions .....	97
<b>Faults .....</b>	<b>98</b>
When a Fault Occurs .....	98
Clearing Faults .....	99
Enable Input .....	99
Jog Inputs .....	99
Cycle Power .....	99
Run Menu .....	99
Host Language Commands .....	99
<b>Operator Terminal .....</b>	<b>101</b>
Installation .....	101
Displaying & Reading Data .....	101
Function Keys .....	101
Xkeys .....	102
Yes / No Keys .....	102
Status Key .....	102
<b>I/O Expansion .....</b>	<b>104</b>
Using the Additional I/O .....	104
Inputs .....	104
Outputs .....	104
Analog Inputs .....	104
<b>Expansion Memory .....</b>	<b>105</b>
Using Expansion Memory .....	105
Storing Programs in Expansion Memory .....	105
Running Programs Stored in Expansion Memory .....	105
<b>A Tutorial Introduction .....</b>	<b>109</b>
Getting Started .....	109
How to Communicate with the ULTRA Plus or IQ .....	109
Touring a Dialog Box .....	109
Hardware Configuration Considerations .....	110
Scale .....	110
Timebase .....	110

Enable .....	111
Auto Tune .....	111
Motion .....	112
Incremental Motion .....	112
Homing .....	112
Absolute Motion .....	113
Programming Basics .....	114
Inputs and Outputs .....	117
Dedicated Inputs Setup .....	117
Read Inputs .....	117
Set Outputs .....	118
Monitor I/O Status .....	119
Scanned Event to Implement a Programmable Limit Switch .....	120
Local/Run-time Values .....	121
Variables .....	122
IF/ELSE Statement, ASSIGN, Block Structure { }	122
Stick Moves .....	123
S-Curve Acceleration .....	124
Subroutines and Loops .....	125
<b>User Variables and Arithmetic .....</b>	<b>127</b>
User Variables .....	127
Scope .....	127
Nonvolatile vs. Volatile Variables .....	127
Resolution and Accuracy .....	128
Variable Characteristics Summary .....	128
Arithmetic .....	129
Precedence and Order of Evaluation .....	129
Boolean Operators .....	129
Conversions .....	131
Encoder Counter Rollover .....	131
Summary of User Variables and Arithmetic .....	133
<b>System Variables &amp; Flags .....</b>	<b>134</b>
Introduction .....	134
What is a System Flag? .....	134
What is a System Variable? .....	134
The Personality Module .....	134
Variable and Flag Summary .....	135
<b>IQ Programming Structure .....</b>	<b>141</b>
Introduction .....	141
Program Structure .....	141
Program Structure Instructions .....	142
DO/WHILE Structure .....	142
WHILE Structure .....	143
Subroutines .....	143

IF Structure .....	144
IF/ELSE Structure .....	145
WAIT Statement .....	145
ON Structure .....	145
JUMP / Label .....	146
Program Structure Instruction Summary .....	146
Scanned Event (Sn) Instructions .....	147
Sn:IF condition action .....	147
Sn:TMRm condition action .....	148
Sn:TMRm start condition, stop condition action .....	148
Sn ON/OFF/CONT .....	148
Timer .....	148
Scanned Event Instructions Summary .....	148
Xkey (Xn) Functions .....	148
Fkey (Fn) Functions .....	150
Interrupts .....	151
Interrupts Step by Step .....	151
Miscellaneous Structure Instruction .....	153
<b>Motion .....</b>	<b>154</b>
Introduction .....	154
Incremental (MOVD) and Absolute (MOVP) Motion .....	155
Stick Moves .....	156
Gearing .....	158
Step and Direction .....	159
Combination Motion .....	159
Feedrate .....	160
Dwell Statement .....	160
Delay Statement .....	160
Motion Instruction Summary .....	160
<b>Closed Loop Control .....</b>	<b>162</b>
Introduction .....	162
Closed Loop Position Control .....	164
Closed Loop Velocity Control .....	165
<b>Inputs / Outputs .....</b>	<b>166</b>
Introduction .....	166
I/O Instruction Summary .....	166
<b>Operator Terminal .....</b>	<b>167</b>
Introduction .....	167
Operator Terminal Instruction Summary .....	167
<b>Miscellaneous .....</b>	<b>168</b>
Introduction .....	168
CONFIG Statement .....	168

Define Home (DH), Define Position (DP) and Define Position 1 or 2 .....	168
Enable / Disable .....	169
Miscellaneous Instruction Summary .....	169
<b>Reserved Words .....</b>	<b>173</b>
<b>Language Reference .....</b>	<b>175</b>
Format .....	175
Reference .....	177
<b>Introduction .....</b>	<b>289</b>
Overview .....	289
Reserved Host Commands .....	290
Implementation .....	290
Addressing .....	290
<b>Syntax .....</b>	<b>292</b>
Command Syntax .....	292
Response Syntax .....	293
<b>Execution Commands .....</b>	<b>295</b>
Motion Execution Commands .....	295
Non-Motion Execution Commands .....	296
Runtime Status and Version Commands .....	297
<b>Variables and Flags Assignment and Request Commands .....</b>	<b>299</b>
Introduction .....	299
Acceleration and Velocity .....	301
Current .....	301
Gains .....	302
Homing .....	303
Inputs – Analog and Digital .....	304
Interrupts .....	305
Motion .....	306
Outputs – Analog and Digital .....	306
Position .....	308
Serial .....	309
System .....	309
Tune Mode .....	314
User Variables and Flags .....	314
<b>File Commands .....</b>	<b>315</b>
Program Maintenance and Directory Commands .....	315
Transfer Commands .....	316

<b>Operator Terminal Commands</b> .....	<b>317</b>
Commands .....	317
Operator Terminal Key Codes .....	320
<b>DDE Server</b> .....	<b>323</b>
<b>System Programs</b> .....	<b>325</b>
What are System Programs? .....	325
When are system programs run? .....	326
Modifying System Programs .....	326
System Program 0 - Auto .....	327
System Program 1 - Jog Forward .....	328
System Program 2 - Jog Reverse .....	329
System Program 3 - Start .....	330
System Program 4 - Stop .....	331
System Program 5 - Run .....	332
System Program 6 - Execute Home .....	333
System Program 7 - Define Home .....	334
System Program 8 - Set Feedrate .....	335
System Programs 9-12 .....	336
System Program 13 - Monitor VEL1 .....	336
System Program 14 - Monitor POSN .....	337
System Program 15 - Monitor PCMD .....	338
System Program 16 - Monitor FE .....	339
System Program 17 - Monitor ICMD .....	340
System Program 18 - Monitor IAVE .....	341
System Program 19 - Monitor RFDR .....	342
System Program 20 - Monitor ADC1 .....	343
System Programs 21-24 .....	343
System Program 25 - Home .....	344
System Program 26 - Emergency Return .....	353
System Program 27 - Error Routine .....	354
System Programs 28-31 .....	355
<b>Optional Operator Terminal Function Key Programs</b> .....	<b>356</b>
Clear Peaks .....	356
Disable .....	356
Enable .....	356
Hardware Reset .....	357
Monitor FVEL1 .....	357
Monitor PFE .....	357
Monitor PICMD .....	357
Monitor POS1 .....	357
Monitor POS2 .....	357
Monitor PVEL1 .....	357
Monitor VCMD .....	358
Monitor VEL2 .....	358

Pause .....	358
Reset .....	358
<b>Optional Home Programs .....</b>	<b>359</b>
Home to a Limit Switch .....	359
Home without Limit Switches .....	359
Home to Encoder Index only .....	359
<b>Application Examples .....</b>	<b>360</b>
SmartBelt Application .....	360
Feed-To-Length Application .....	363
Continuous Web with Registration .....	370
Auger Dispensing Application .....	377
Thermoformer Application .....	381
In-line Bottle Filler .....	386
Vertical, Form, Fill and Seal .....	392
Lane Diverter .....	398
Automated Test Station .....	403
<b>What's New in Each Version .....</b>	<b>410</b>
New Features .....	410
Changes to Version 3.2.4 from Version 3.2.3 .....	410
Changes to Version 3.2 from Version 3.0 .....	410
Changes to Version 3.0 from Version 2.1 .....	411
Changes to Version 2.1 from Version 2.xx .....	411
Changes to Version 2.0 from Version 1.xx .....	411
<b>Upgrading From Version 1 .....</b>	<b>414</b>
Backup Existing Files and Personality Module .....	414
Install new EPROMS .....	415
Install IQ Master .....	415
Upgrade Personality Module .....	415
<b>Error Messages .....</b>	<b>418</b>

# List of Figures

- ULTRA Plus/IQ-2000/IQ-5000 System Components ..... 5
- IQ-550 System Components ..... 5
- Personality Module Dialog Boxes ..... 33
- ULTRA Plus/IQ-Series Block Diagram ..... 163
- Legend Used in Home Diagrams ..... 345
- Normal Home Sequence Hitting Forward Limit Switch ..... 345
- Normal Home Sequence Starting with the Home Switch Active ..... 346
- Normal Home Sequence, Hitting Home Switch while Moving Forward ..... 346
- Error: (-) Edge of Home Switch not Between Limits ..... 346
- Home Switch Active in the Opposite Sense ..... 347
- Acceptable Deceleration Distance ..... 347
- Deceleration Distance Resulting in an Axis Crash ..... 347





# Preface

---

## Introduction

Read this preface to familiarize yourself with this manual. This preface covers the following topics:

- Introduction
- Who Should Use this Manual
- Where to Find Help
- Contents of this Manual
- Related Documentation
- Conventions Used in this Manual
- Using Online Help
- Allen-Bradley Support

---

## Who Should Use this Manual

Use this manual if you use IQ Master to configure and operate ULTRA Plus™ and IQ-Series® controllers, or to design, test or run programs.

---

## Where to Find Help

You can find help for IQ Master in both this manual, and Online Help.

# Contents of this Manual

This manual contains the following sections:

Title	Contents
Preface	An overview of this manual and sources of information.
Getting Started	Introduces the ULTRA Plus and IQ-Series controllers, and the IQ Master software package.
IQ Master Environment	Examines the menus and menu items in IQ Master.
Software and Hardware Integration	Commissioning the ULTRA Plus and IQ-Series controllers with IQ Master.
Programming	A tutorial on programming motion with IQ Master and the ULTRA Plus and IQ-Series controllers.
Language Reference	Describes in detail each function, system variable and flag.
IQ Host Command Language	Describes the host mode communications capabilities for computers or PLCs to control the operation of an ULTRA Plus and IQ-Series controller.
Appendixes	Provides supplementary information and programming examples for: <ul style="list-style-type: none"> <li>• DDE Server</li> <li>• System Programs</li> <li>• Optional Operator Terminal Function Key Programs</li> <li>• Optional Home Programs</li> <li>• Application Examples</li> <li>• What's New in Each Version</li> <li>• Error Messages</li> </ul>

## Related Documentation

The following documents contain additional information concerning related Rockwell Automation products. To obtain a copy, contact your local Rockwell Automation office or distributor.

<b>For information about:</b>	<b>Read this document:</b>	<b>Publication Number</b>
Installing an ULTRA Plus system.	<i>ULTRA Plus Series Positioning Drive Module Installation Manual</i>	1398-5.1
Installing an IQ-550 system.	<i>IQ-550 Position Control Module Installation Manual</i>	Part Number 0013-1022-004
Installing an IQ-2000 and IQ-5000 system.	<i>IQ-2000 and IQ-5000 Positioning Drive Module Installation Manual</i>	Part Number 0013-1027-004
Programming electronic cams for an ULTRA Plus or IQ system.	<i>IQ-Series IQ-Cam Software Manual</i>	Part Number 0013-1053-001
The ULTRA Plus Positioning Drive	ULTRA Plus brochure	1398-1.1
ULTRA Series Specifications	ULTRA Series Product Data	1398-2.0
Wire sizes and types for grounding electrical equipment	National Electrical Code	Published by the National Fire Protection Association of Boston, MA.
Allen-Bradley documentation, including ordering instructions and alternative media and multi-language availability.	Allen-Bradley Publication Index	SD499



## Conventions Used in this Manual

The following conventions are used throughout this manual:

- Bulleted lists provide information, not procedural steps.
- Numbered lists provide sequential steps.

# Using Online Help

The following types of online help are available:

To use this:	Do this:	Description
<b>Help Menu</b>	Either: <ul style="list-style-type: none"> <li>• Click on Help in the menu bar.</li> <li>• Press ALT to activate the menu bar, then H to pull down the Help menu.</li> <li>• Press the F1 key.</li> </ul>	Displays the pull down Help Menu.
<b>Help Contents</b>	Either: <ul style="list-style-type: none"> <li>• Click on the Contents selection.</li> <li>• Press the underlined letter (C in Contents).</li> <li>• Use the arrow keys to highlight your selection and then press ENTER.</li> </ul>	Contains help on every menu and menu item, and the IQ Basic programming language.
<b>How to Use Help</b>	Either: <ul style="list-style-type: none"> <li>• Click on the How to use Help selection.</li> <li>• Press the underlined letter (H in How).</li> <li>• Use the arrow keys to highlight your selection and then press ENTER.</li> </ul>	Provides information on how to use on-line help.
<b>Context Sensitive Help</b>	Either: <ul style="list-style-type: none"> <li>• Press Shift+F1,</li> <li>• Click  (the context sensitive help toolbar button).</li> </ul> Then click anywhere in the IQ Master window, such as a Toolbar button or menu item.	Provides help on a specific portion of IQ Master.  The mouse pointer will change to an arrow and question mark indicating Context Help mode. The Help topic will be shown for an item you select.
<b>About IQ Master</b>	Either: <ul style="list-style-type: none"> <li>• Select About IQ Master from the Help menu.</li> <li>• Click the  toolbar button.</li> </ul>	Access the About IQ Master section that displays: the version of IQ Master that you are using, the version of Windows™ that you are using, and the amount of memory available.

---

# Allen-Bradley Support

Allen-Bradley offers support services worldwide, with over 75 sales/support offices, 512 authorized distributors and 260 authorized systems integrators located throughout the United States alone, plus Allen-Bradley representatives in every major country in the world.

## Local Product Support

Contact your local Allen-Bradley representative for:

- Sales and order support
- Product technical training
- Warranty support
- Support service agreements

## Technical Product Assistance

If you need to contact Allen-Bradley for technical assistance, please review the information in this manual or in the Online Help file first. Then call your local Allen-Bradley representative. For the quickest possible response, we recommend that you have the catalog numbers of your products available when you call.

**PREFACE**

# Part 1

# Getting Started

This part of the manual introduces the Allen-Bradley ULTRA Plus and IQ-Series controllers and IQ Master software. The requirements for the Personal Computer (PC) to run IQ Master and the installation of IQ Master on your PC are covered.

## INTRODUCTION



# Introduction to IQ Master

---

## About the Allen-Bradley IQ Master Manual

This manual contains the procedures you will need to work with IQ Master software. Throughout this manual, the terms ULTRA Plus, IQ, IQ-Series controller, and controller are used to refer to either an ULTRA Plus, IQ-2000 or IQ-5000 Positioning Drive Module, or an IQ-550 Position Control Module unless otherwise specified. The term PDM is used to refer to either an ULTRA Plus, an IQ-2000 or an IQ-5000 Positioning Drive Module unless otherwise specified.

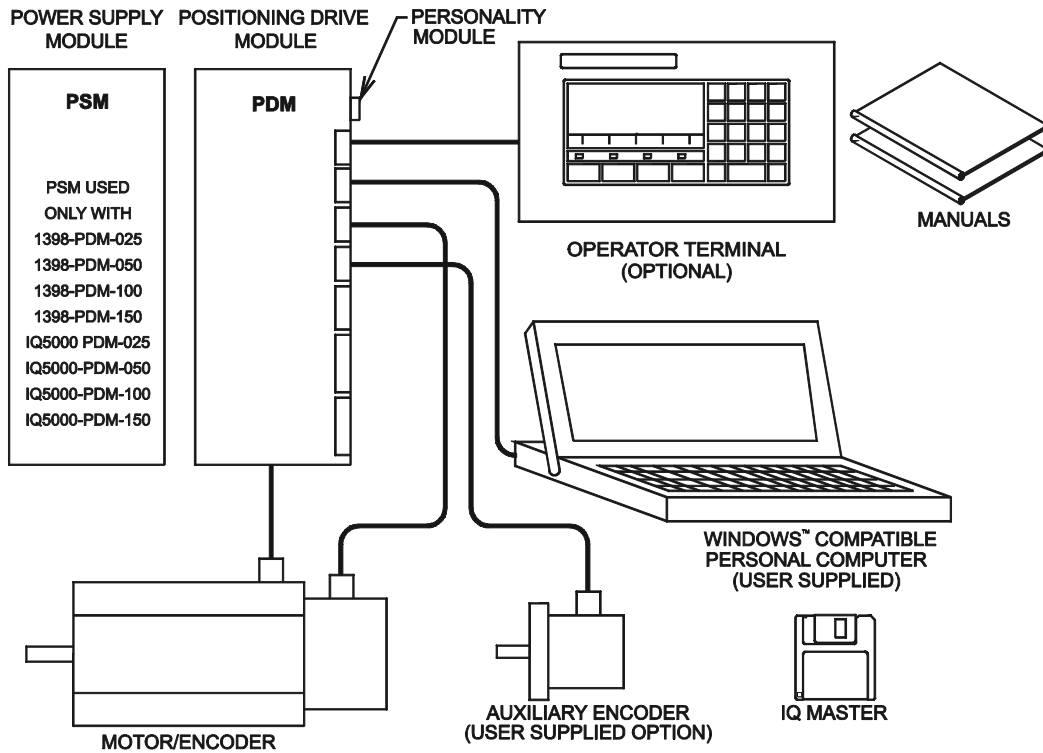
The manual is organized into six parts plus appendixes. If you are using ULTRA Plus or IQ-Series equipment for the first time, you should read Parts 1 through 4 completely. Starting in Part 2, it is helpful if you have an operational ULTRA Plus or IQ-Series controller attached to your PC. As you begin to develop your programs you will begin using Part 5 as your reference guide. You only need to refer to Part 6 if you are using a computer, PLC or some other device as a host.

Part	Description:
Part 1	<p><i>Getting Started</i></p> <p>This part of the manual discusses the requirements for running IQ Master and covers the installation of IQ Master on your Personal Computer (PC).</p>
Part 2	<p><i>IQ Master Environment</i></p> <p>This part of the manual covers, in detail, each of the menu items in IQ Master.</p>
Part 3	<p><i>Software and Hardware Integration</i></p> <p>The interaction of IQ Master and the ULTRA Plus or IQ-Series controller is discussed in this part of the manual. You will learn how to configure your ULTRA Plus or IQ-Series controller for your application using IQ Master.</p>
Part 4	<p><i>Programming</i></p> <p>This part starts out with a tutorial on programming the ULTRA Plus or IQ-Series which is designed to get you up and running quickly. From there it moves into a detailed discussion of programming. The chapters in this part are broken down into the major components of your program, for example: motion, math, etc.</p>
Part 5	<p><i>Language Reference</i></p> <p>The language reference part contains a detailed description of each function, system variable and system flag, arranged in alphabetical order.</p>
Part 6	<p><i>Host Language Commands</i></p> <p>This part contains a detailed description of the host mode communication capability of the ULTRA Plus or IQ-Series controller. The host mode is used if there is a host computer or PLC that will control operation of the ULTRA Plus or IQ-Series.</p>

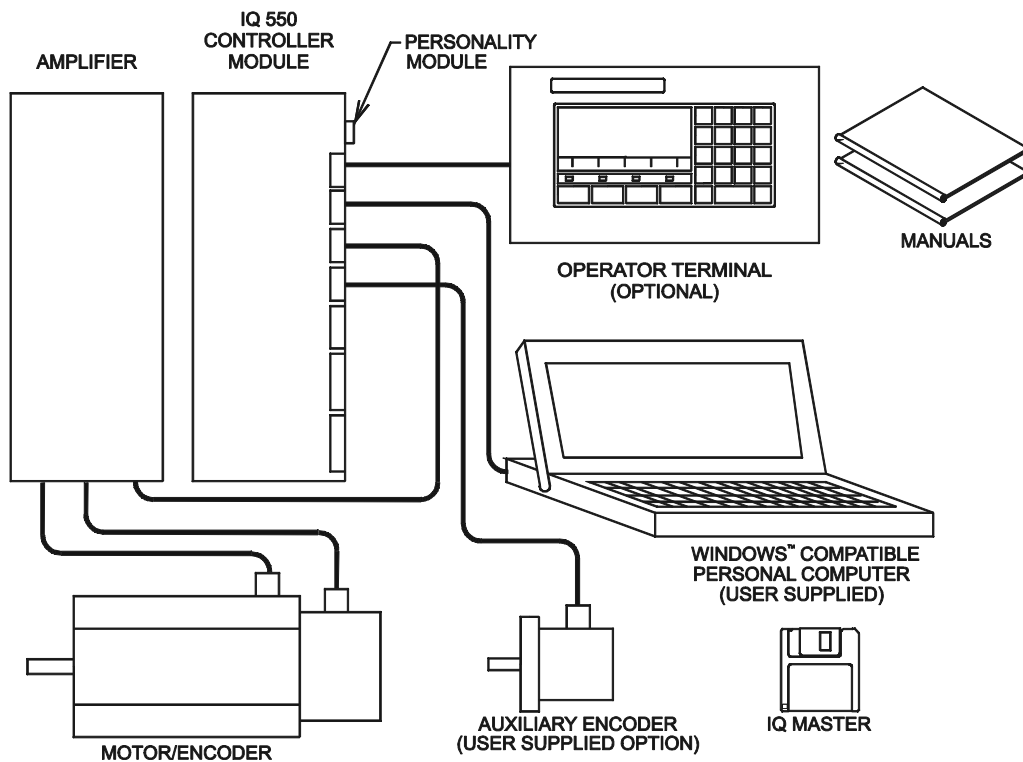
---

## ULTRA Plus or IQ-Series System Overview

An ULTRA Plus or IQ-Series motion control system consists of a number of components connected to accomplish a specific function. This section provides a brief overview of the various components of the ULTRA Plus or IQ-Series motion control system.



**ULTRA Plus/IQ-2000/IQ-5000 System Components**



**IQ-550 System Components**

## IQ Master

IQ Master is a Windows based software package that provides the user interface to the ULTRA Plus or IQ-Series controller. It is used to edit and compile application programs, configure, monitor, and troubleshoot the controller.

## IQ-550 Position Control Module

The IQ-550 Position Control Module is a stand alone single axis programmable position controller. It can interface to most motor controllers through an analog command signal and encoder feedback to form a system with power ranges from small subfractional horsepower DC systems to several hundred horsepower vector drive induction motor systems. Motion programs are stored in on-board nonvolatile memory. Two RS-232/RS-422 serial ports provide communications with the personal computer and the optional operator terminal. Optically isolated digital I/O simplifies machine interfacing and control.

## Positioning Drive Modules (PDM)

The ULTRA Plus or IQ-Series Positioning Drive Module (PDM) is a self-contained single axis programmable motion controller. The PDM provides control and power for the brushless servo motor. Motion programs are stored in on-board nonvolatile memory. Two RS-232/RS-422 serial ports provide communications with the personal computer and the optional operator terminal. Optically isolated digital I/O allows simple machine interfacing and control.

PDMs are manufactured in different packages which cover a wide range of power capability. The ULTRA Plus PDM-10, 20, 30 and 75 and the IQ-2000 PDMs incorporate an integral power supply in each PDM, and supply continuous torque's of 3 to 90 inch-pounds in combination with standard motors. The ULTRA Plus PDM-25, 50, 100 and 150 and the IQ-5000 PDMs use a separate power supply module, which can be shared among multiple PDMs, and provide continuous torque's of 20 to 750 inch pounds with standard motors.

### *PDMs with Integral Power Supply Modules*

The ULTRA Plus (1398-PDM-10, 1398-PDM-20 and 1398-PDM-30) and the IQ2000 (PDM-10, PDM-20, and PDM-30) are rated for 10, 20 and 30 amp peak currents respectively. These modules are packaged with an integral power supply to achieve a small size.

Input power to the PDM-10's, -20's, or -30's is single phase AC. Input power may be optionally isolated through a transformer. These PDMs have a built-in solid-state "soft charge" of the internal DC bus capacitor. They also include a built-in dissipative shunt regulator that provides quick discharge of the DC bus capacitor, and double as an emergency synchronous motor dynamic brake. The PDM-30 allows use of an optional external shunt resistor for applications requiring higher shunt power capability than what is provided by the internal shunt resistor.

### *PDMs with Separate Power Supply Modules*

The ULTRA Plus (1398-PDM-25, 1398-PDM-50, 1398-PDM-100, and 1398-PDM-150) and the IQ5000 (PDM-25, PDM-50, PDM-100, and PDM-150/150X) are rated for 25, 50, 100, and 150 amp peak currents respectively. These higher power PDM modules use a separate power supply module (PSM-50 or PSM-125), which may be shared among multiple PDMs to achieve the most economical system package. Other than the packaging and power ranges, the higher power PDMs are identical to the stand-alone PDMs from a setup and programming perspective.

## Power Supply Module (PSM)

The power supply module is only required for ULTRA Plus (1398-PDM-25, 1398-PDM-50, 1398-PDM-100, and 1398-PDM-150) and IQ-5000 systems. The Power Supply Module (PSM) can supply DC power to as many as six PDM modules. The only inputs to the PSM are non-isolated or isolated three phase power. The output is a two wire DC bus. The PSM requires no adjustments, protects itself, provides troubleshooting diagnostics, and has a built-in solid-state "soft charge" of the DC bus capacitors. It also includes a built-in dissipative shunt regulator that provides quick discharge of the DC bus capacitors and doubles as an emergency synchronous motor dynamic brake.

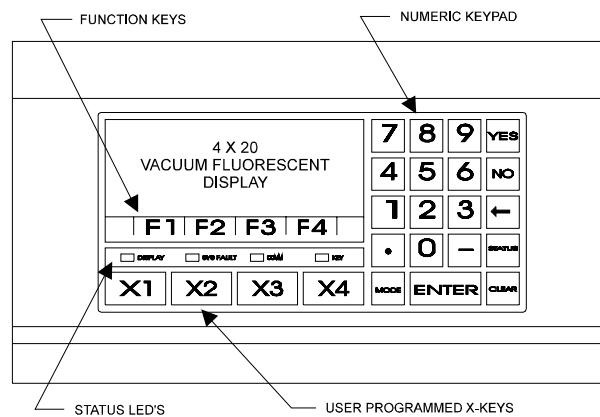
## Motors

A wide range of Allen-Bradley F-Series, H-Series, N-Series and W-Series permanent magnet synchronous motors are available for use with the PDM modules plus the I-6600 induction motor. Each motor includes an integrally mounted encoder. Most motors are available with options including spring set brake and/or shaft oil seal. The synchronous motors have a rear shaft available for mounting optional feedback devices. MS style wiring connectors are standard for all S-Series and F-Series motors.

## Operator Terminal

The optional Operator Terminal is a rugged man/machine interface device. It allows the machine operator convenient access to status information, program variables, and control functions, plus message display capabilities. The operator terminal has a bright 4 line by 20 character vacuum fluorescent display and a sealed membrane keyboard with tactile feedback.

The operator terminal displays multiple status screens for monitoring and diagnostics. Four soft function keys are available to perform up to twenty-four (24) predefined functions, which include selecting and running a program, jogging the system, stopping a program, and more. The user program may display messages and prompts on the screen, and receive input from the operator terminal keypad to enter program variables such as distances, speeds, batch counts, and others. Four additional user programmable keys provide extra flexibility within a user program.



## Personality Module (PM)

The Personality Module is a nonvolatile memory device which stores the information necessary to customize an ULTRA Plus or IQ-Series controller for a specific application. The PM holds parameters to match the motor and the controller, as well as user programs and parameters. A Personality Module may be physically removed and transferred to another ULTRA Plus or IQ-Series controller if the replacement of a controller is necessary to simplify servicing the machine. The Personality Module data can also be saved in a computer file and loaded into the ULTRA Plus or IQ using the File, Transfer, dialog box.

## Manuals

The manuals are conveniently broken into two volumes. The appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) contains all the information required for mounting and wiring the system. This manual contains all the information that a user and programmer require to quickly set up, and develop programs for the controller.

## Option Cards

Each ULTRA Plus or IQ-Series controller can have an option card (*not pictured*) mounted on its main circuit board, inside the cover. The controller has a connector which connects the option card to the power supplies and microprocessor through a ribbon cable. The option cards currently available include a Memory and I/O Expansion Card and an I/O Expansion Card.

**Memory and I/O Expansion Card**

Additional memory and I/O may be added with a Memory and I/O Expansion Card. This board adds the following memory, input and output capabilities to the ULTRA Plus or IQ-Series controller.

- 32 Kbytes of additional nonvolatile memory for the storage of up to 32 additional programs.
- 32 TTL Inputs.
- 16 TTL Outputs.
- 4 twelve bit Analog Inputs.

**I/O Expansion Card**

Additional I/O may be added with an I/O Expansion Card. This board adds the following input and output capabilities.

- 32 TTL Inputs.
- 16 TTL Outputs.

**Personal Computer (PC)**

A user supplied Personal Computer is required to run IQ Master software, and its requirements are specified in “Minimum Personal Computer Requirements” on page 9.

---

## Symbols and Conventions

This manual uses the following typographic conventions:

Example:	Description:
DIF <i>dist,condition</i> ,JUMP <i>label</i>	<p>IQ Basic instructions are shown in all capital letters.</p> <p>Capital letters indicate IQ Basic keywords, which are required parts of IQ Basic statements. Keywords include language commands (e.g. DIF), local parameters (e.g. KP), dedicated flags (e.g. ATHOME) and variables (e.g. G1). In this manual keywords are shown in all capital letters. However, IQ Basic is a case insensitive language and can be typed in lower, upper, or mixed case. A keyword must be followed by a space before any other part of the instruction.</p> <p>Small italic letters indicate user supplied values. You must substitute a value according to rules explained in the text. This can be a variable or a constant.</p>
<i>Gn, Vn, Fn, Bn</i>	<p>User variables. The capital letters (G, V, F, B) can be typed as either lower case or upper case. The small italic letter in each name represents the number of any register. For example G13 might be a substitute for <i>Gn</i>.</p>
MOVD = <i>value</i> [ <i>V = value</i> ]	<p>Square brackets indicate an optional item. If the optional parameters are not included, then the current setting for that parameter is used.</p>
ACCEL = <i>value</i> G1 = VEL	<p>Set a system variable. Read a system variable.</p>
<i>Fn</i> = ATHOME CLRPEAKS = ON	<p>Read a flag variable. Set a flag variable.</p>

MOVD = G1, V = G2	Example programs or program fragments are shown in this font.
win	Text presented in lower case bold is information to type at the DOS prompt. For example: To start Windows from the DOS prompt, type <b>win</b> and then press ENTER.
ALT+F4	Keys that should be pressed simultaneously are shown with a (+) between the key names.
ALT, F, N	Keys that should be pressed in sequence are shown with a (,) between the key names. This example would open the File menu and then open a new file.

---

## Minimum Personal Computer Requirements

### IQ Master

- A DOS system computer with a 286 microprocessor or better (386 or 486 preferred)
- A hard disk, with 1.5MB free
- 3½ inch, 1.44MB floppy disk drive
- 2MB of RAM minimum (4MB recommended)
- An Enhanced Graphics Adapter (EGA) or better resolution monitor
- Microsoft® Windows version 3.1 or later

In addition, a mouse is recommended, but not required.

## New Features

There have been many additions and enhancements to the ULTRA Plus or IQ systems. A detailed description of these new features and command changes is in the appendix “What’s New in Each Version” on page 410. If you have used the ULTRA Plus or IQ system before, reading this appendix is recommended. The following is a brief list of the new features:

### New in Version 3.2.4 vs. Version 3.2.3

- IQ Master software runs on Allen-Bradley ULTRA Plus.

### New in Version 3.2 vs. Version 3.0

- Extended Debug features
- Software programmable Personality Module
- Print and Read statements precision to 4 decimal points.
- Looping and Repeat value range from 0 to 65535.
- Drive current values scaled using drive module (Dm) current.
- Purge Motion function added.
- DP1 and DP2 encoder position statements added.

### New in Version 3.0 vs. Version 2.1x

- Support for IQ CAM
- Tracking function enhancement to electronic gearing
- Compiler options program statements
- Off-line parameter editing
- DDE server capability
- Support for redefining key codes for generic operator terminals
- Movecomplete and Jogactive system flags
- Support for connecting one Operator Terminals to multiple IQs
- Toolbar button for File Transfer

### New in Version 2.1 vs. Version 2.0

- Support for the IQ-550 Position Control Module
- Support for a generic operator terminal
- Default File Extension changes - all IQ files begin with “Q”
- Toolbar buttons
- Context Sensitive Help
- Replace function added to the editor
- Ability to monitor G and V variables
- Ability to directly save a file to the IQ after successful compile

### New in Version 2.x vs. Version 1.x

- Windows based environment
- On line help



- Complete Host Language Command set
- Off line compiler and full screen editor
- IQ Basic language enhancements:
  - Multiple statements can be treated as a group.
  - Two new statements have been added to repeatedly execute a statement or group of statements based on a condition: the While and the Do While statements.
  - Parentheses can be used around a condition for clarification or to establish operator precedence.
  - Two new variables have been added to capture position information based on an index pulse: IX1P2 (POS2 on index 1) and IX2P1 (POS1 on index 2).

# Quick Start Check List

---

## Introduction

New users of ULTRA Plus or IQ-Series controllers are encouraged to read through the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) and this manual before beginning a project. However, for experienced users of the ULTRA Plus or IQ-Series, this check list was developed to guide you through the process of getting your system up and running. If you are a new user you will want to return to this section after you have completed your review of the rest of the material.

---

## Quick Start Check List

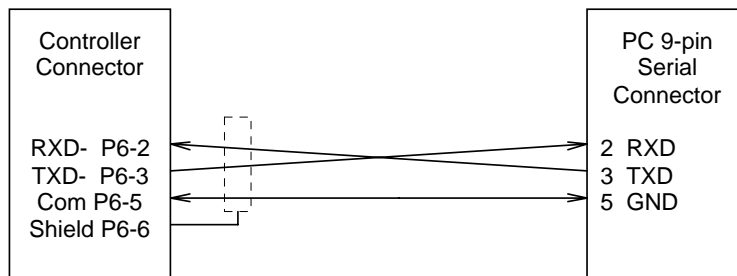
The following check list will step you through the items that are required for the typical application:

1.  Install IQ Master on your PC. See “Using the IQ Master Setup Program” on page 17.
2.  Install the ULTRA Plus or IQ-Series controller and go through the Start-up procedure. See “Applying Power for the First Time” on page 21.
3.  Connect the serial cable between the PC and the ULTRA Plus or IQ-Series controller.
4.  Verify that the ULTRA Plus or IQ-Series Controller and PC can communicate properly over the serial cable connected between plug 6 (P6) on the controller and a serial communications port on the PC.
  - Go to the Diagnostics menu and select Version (page 72). If a dialog box with the version is displayed, communication is working properly and you may skip to step 5. If an error message is displayed, continue with step 4.

- If you get an “IQ not responding” error message, go to the Communications menu, PC Set Up and check your communications setup. You should first verify that the communications port you have selected is the one that your cable is connected to. Then verify that the communications protocol matches the protocol in the ULTRA Plus or IQ-Series controller. The factory settings for a new controller are:
  - RS-232
  - 9600 Baud
  - No Parity
  - 8 Data bits
  - 1 Stop Bit
  - Flow Control XON/XOFF

If this is not a new controller any of these setting may have been changed by the previous user.

- If you have selected an address on the address DIP switches, verify that the same address is selected in the Communications menu, Axis Select dialog box.
- Verify that the serial cable is constructed properly



- Verify that your system has the serial communication port that you have selected. With Windows, you received the Diagnostic program, MSD.EXE. Exit IQ Master and Windows and type **msd** at the DOS prompt. Select the Com Ports button. Verify that there is a port address for the serial port that you have selected. Return to Windows and IQ Master.
  - If you still have not been able to communicate with the controller see “Allen-Bradley Support” on page 5.
5.  Go to the Parameter menu and select System. Make your selections and choose OK.
  6.  Go to the Parameter menu and select Velocity/Accel. Make your selections and choose OK.
  7.  Go to the Parameter menu and select Inputs. Make your selections and choose OK.
  8.  Go to the Parameter menu and select Outputs. Make your selections and choose OK.
  9.  Go to the Parameter menu and select Default Outputs. Make your selections and choose OK.
  10.  Go to the Parameter menu and select Feedback Configuration. Make your selections and choose OK.
  11.  Go to the Parameter menu and select Fkey Set Up. Make your selections and choose OK.

12.  Go to the Run menu, Tune. Follow the Auto Tuning procedure outlined in Part 2 of this manual. If required, perform the manual tuning procedure.
13.  If you have programs that were created for version 1, refer to Appendix C to convert the programs to be compatible with version 2. Then continue at the beginning of Part 4 with a Programming Tutorial Introduction

# Getting Started with IQ Master

---

## Hardware and Software Requirements

At this point you should have Windows installed on your PC if you are going to use IQ Master. If not, refer to the appropriate Microsoft manuals to install Windows on your computer.

To run a program, and/or access many of the IQ Master setup and configuration windows you will need to have the ULTRA Plus or IQ-Series controller installed and ready to apply power. The procedure for installing the hardware is in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004).

---

## Using Windows Without a Mouse - A Quick Review

IQ Master can be used without a mouse if you use the standard key combinations to activate menus, select radio buttons, move between windows, etc. Some of the most commonly used key combinations are summarized in the following two tables.

### Windows Hot Keys

Name	Description
F1	Help
ALT+F4	Exit the application that you are currently in, or exit Windows if you are in the Program Manager.
ALT+HYPHEN	Activate the Control-menu box in a document window, for example, when editing a file.

ALT+F6	Switch between the main application and one child window (for example, to switch between the IQ Master for Windows main menu and the Run Control dialog box); or, to switch between multiple child windows (for example, to switch between the Status and Run Control dialog boxes).
ALT+SPACEBAR <sup>1</sup>	Activate the Control-menu box of the active window (the box with the hyphen in the upper left corner of the window). F10, ENTER can also be used to activate the Control-menu box. Use this menu to restore, move, maximize, minimize, or close the window. (For document windows use ALT, HYPHEN to activate the Control-menu box, for example, when editing a file.)
ALT+TAB	Switch between active tasks.
CTRL+F6	Switch between multiple document windows (for example, to switch between multiple files that you are editing).
SHIFT+ALT+F6	Alternate switching between the main application and multiple child windows (for example, to switch between the IQ Master for Windows main menu and the Run Control dialog box and then switch to the Status dialog box).
SHIFT+TAB	Move to the previous item in a dialog box.
TAB	Move to the next item in a dialog box.

1. Using Restore from the Control menu when a window is maximized has the same result as clicking the Restore button (the up and down arrow in the upper right corner of a window). If the application has been reduced to an icon, Restore will return the window to its previous size.

## IQ Master Hot Keys

Name	Description
F2	Compile the program in the active window.
F7	Save the program in the active window to the ULTRA Plus or IQ.
CTRL+S	Save file in the active window to disk
ALT+Back Space	Restore the most recently deleted text
SHIFT+Delete	Cut the current text selection from your program into the clipboard
CTRL+Insert	Copy the selected text into the clipboard
SHIFT+Insert	Paste the contents of the clipboard into the program at the cursor location
F3	Find Next - repeat the last find, looking in the forward direction
F4	Find Previous - repeat the last find, looking in the reverse direction
F8	Display the Run Control dialog box
CTRL+X	Stop motion
F9	Display the Status dialog box

## Using the IQ Master Setup Program

To install IQ Master to the hard drive:

1. Make a backup copy of your disks before proceeding. If your computer has only one floppy disk drive, type **diskcopy a: b:** from the DOS command line prompt. Otherwise, copy disk from the disk menu in the Windows file manager. You will be prompted when to insert the SOURCE disk and when to insert the TARGET disk. As an alternative, copy the disk to your hard disk and then from the hard disk to another floppy disk.
2. If Windows is not already running, type **win** at the DOS prompt.  
-or-  
If Windows is running, close any open applications.
3. Insert the IQ Master disk in to a 1.44MB floppy disk drive, typically drive a:, and close the drive door.
4. From the File menu, choose Run (ALT, F, R).
5. Type **a:setup** and then press ENTER. A dialog box will appear saying that the setup is initializing (this box may be present for 40 seconds).
6. A dialog box will appear, confirming that you are about to install IQ Master on to your hard drive. Choose Continue, or press ENTER.
7. Next, the Setup Options dialog box will appear. This box lists the different components which may be installed. If you do not wish to install these files, clear the check boxes and choose Continue.
8. Setup will then ask where you would like to install IQ Master.  
To accept the path that Setup proposes, choose Continue.  
-or-  
To choose your own directory, type a new path in the Install To box, and then choose Continue. You will not have the opportunity to confirm your entry so type carefully.
9. A status bar will be displayed to keep you informed of the installation progress. When Setup is complete, choose OK or press ENTER to return to Windows.

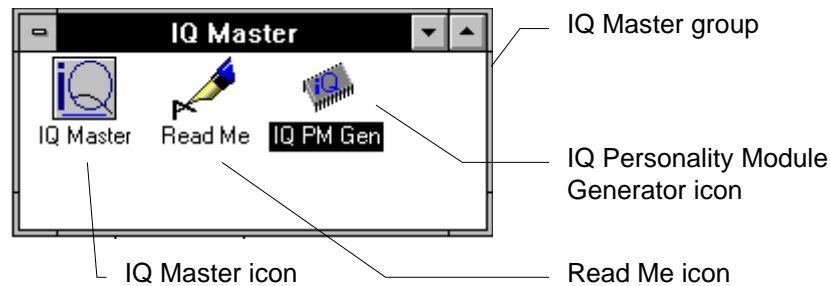
## The Readme File

When Setup installed IQ Master, a file, README.WRI was installed in the IQ Master directory. This file contains information that became available after this manual was printed. A Read Me Icon was added to the IQ Master group. After you install IQ Master you can access this file by double clicking on the Read Me Icon.



## Starting and Quitting IQ Master

Setup automatically creates the IQ Master program group, the IQ Master icon and then returns you to Windows.



### To Start IQ Master

You can start IQ Master by using either the mouse or the keyboard.

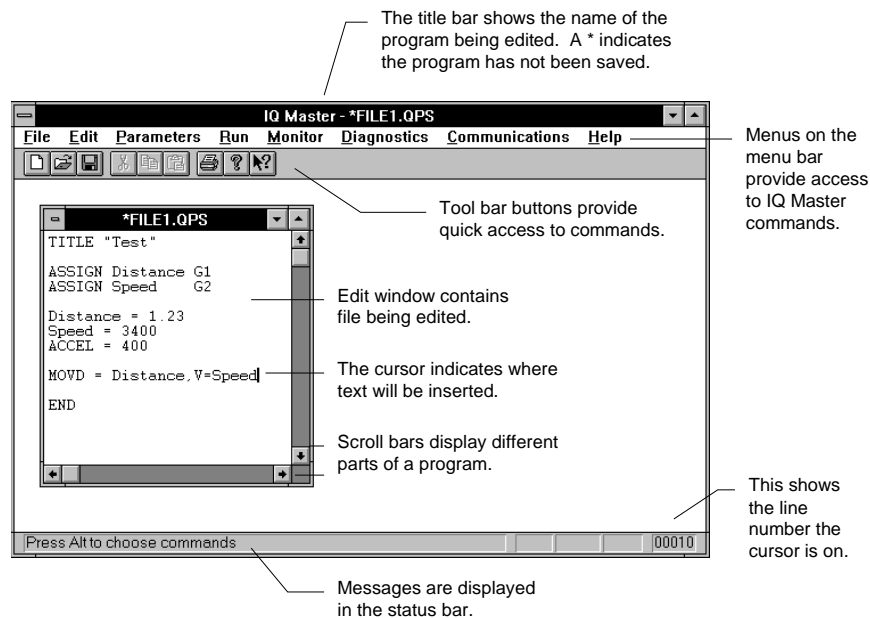
- Double-click the IQ Master icon.
- or-
- Select the IQ Master icon using the arrow keys, and press ENTER.

**Tip:** If the IQ Master group is not active, hold down CTRL and press TAB until the IQ Master title bar and icon are highlighted, or use the Window menu.

IQ Master may also be started as a DDE Server, refer to “DDE Server” on page 323.

### The IQ Master Screen

When you start IQ Master a blank screen with the IQ Master title bar, menus, and toolbar appears. A typical screen with a program being edited is shown below.



### Quitting IQ Master

When you quit IQ Master, if you have made changes to your program, IQ Master will ask if you want to save the changes. If you answer yes and you have not named the program, IQ Master will save it under

the name that it used to open it with, FILE*n*, ⇒with a QPS extension. IQ Master assigns a number *n* to each new file that is open to distinguish it from other unnamed files.

To Quit IQ Master, from the File menu, choose Exit (ALT, F, X) or press ALT+F4.

- If you are using a mouse, point to the File menu and click the left mouse button. Then point to the Exit command and click the left mouse button again.
- If you are using the keyboard, press the ALT key to activate the menu bar, press the F to choose the File menu, and then press X, the underlined letter in the Exit command, or press ALT+F4.

---

## On-line Help

There is extensive help in IQ Master on the commands and procedures you use to accomplish tasks. In addition, there is information on all the language statements, system variables, and system flags.

### To Start Help:


#### Help Menu

1. If you are using a mouse, point to the Help menu and click the left mouse button.  
-or-  
If you are using the keyboard, press the ALT key to activate the menu bar and press H to choose the Help menu.
2. From the Help menu, choose Contents to get help on IQ Master  
-or-  
choose How to Use Help to get help on moving around in Help.

#### F1 Key

Highlight the item you need information on and press the F1 key.

#### Toolbar

Click on the context sensitive Help button, , then click the item you want help on.

Pressing Shift+F1 has the same effect as clicking on the context sensitive Help button.

---

## Upgrading from Version 1

If you are currently using a version 1 IQ, you will need to upgrade the system programs in your personality module and make some minor changes to your programs. A detailed procedure for upgrading is in Appendix D, Upgrading From Version 1.

# Applying Power for the First Time

Outlined below are the steps that should be followed when applying power to the equipment for the first time. This procedure covers ULTRA Plus, IQ-2000 PDMs, IQ-5000 PDMs, IQ-5000 PSM, IQ-550 Controllers, Motors and the Operator Terminal.

This start-up procedure assumes that the equipment has previously been mounted and wired, but has not had power applied to it.

---

## Start-Up Procedure for ULTRA Plus and IQ-2000 Systems

1. Measure voltage between terminals marked L1 and L2 to ensure incoming power is off. Also measure voltage between terminals marked L1 AUX and L2 AUX if used, to ensure power is off. The green READY LED (power supply) and the bi-color status LED should be off.
2. Disconnect input wires connected to terminals L1 and L2 (also terminals L1 AUX and L2 AUX if used). Arrange the wires in a safe position to test the incoming voltage.
3. Turn input power on and measure line voltage to ensure that it is in the proper voltage range (115 or 230 VAC depending on motor speed requirements).
4. Turn power off and reconnect the wires to terminals L1 and L2 (and terminals L1 AUX and L2 AUX if used).
5. With input power still off, disconnect motor leads from terminals R, S, and T. Verify with an ohmmeter that the resistance between terminals R to GND, S to GND, and T to GND is above 500k ohms. If the resistance is OK, reconnect the motor leads to the terminals marked R, S, and T.
6. Make sure that the Personality Module (PM) is properly configured. If necessary, refer to the Applying Power chapter in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for step-by-step instructions on configuring a Personality Module using IQ Master software.
7. Connect all interface cables (P1-P7) to the PDM.

8. Make sure that the motor is not connected to the load when applying power to prevent any mechanical damage in case of a fault. Apply input power with the PDM disabled and check the LEDs. The green READY LED should turn on and the bi-color status LED should turn green. If the status LED is red rather than green, the PDM is in a fault condition. Select Status from the Monitor menu to find out the type of fault. Correct the fault (the status LED will turn green when the fault has been cleared and a hard reset performed). Check that air is coming out of the top of the PDM near the mounting plate to ensure that the fan is operating.
9. Select Encoder from the Diagnostics menu. Verify that Counts increases positively when the motor shaft is rotated clockwise as viewed facing the motor shaft.
10. Remove input power. The LEDs should turn off within one second.

---

## Start-Up Procedure for ULTRA Plus and IQ-5000 Systems

Starting an ULTRA Plus or IQ-5000 involves starting a Power Supply Module (PSM) and at least one Positioning Drive Module (PDM). The PSM is checked first and then the PDMs. Perform the start-up procedure on all the PDM attached to the PSM at the same time.

### ATTENTION



Dangerous voltages may exist after power is removed! Check the DC bus voltage after removing power and before working on an ULTRA Plus or IQ-5000 power supply module or ULTRA Plus or IQ-5000 positioning drive module.

---

### Power Supply Module

1. Measure voltages at L1, L2, and L3 phase to phase to ensure incoming power is off. Make sure that the green PSM READY LED is off. Remove the PSM cover.
2. Disconnect all three phase input wires attached to L1, L2, and L3. Arrange the input wires in a safe position to test incoming voltage levels.
3. Turn input power on and measure phase to phase voltages to ensure that they are balanced and in the proper voltage range (88-265 VAC 3 phase).
4. Turn power off and reconnect the wires to L1, L2, and L3. Phasing is arbitrary.
5. Disconnect all wires from the PSM DC bus terminal posts marked + and -. Install the PSM cover, then turn on input power. Verify that DC bus voltage is in the proper range (125-375 VDC, 325 VDC with 230 VAC input). Check that the green PSM READY LED is on and that all three red LEDs are off and that the PSM fans are operating.
6. Disconnect input power and check that the green LED goes off and the DC bus voltage falls to less than 15 VDC within one second. If so, then the PSM is working properly.
7. Remove PSM cover. Reconnect the DC bus wires to the PSM terminals marked + and - insuring that proper polarity is maintained. Install the PSM cover.

**Note:** Do not substitute the DC bus wires provided with wires of a different length or gauge as a failure may occur.

### Start-Up Procedure for ULTRA Plus or IQ-5000

1. Measure voltage at terminals marked L1, L2, and L3 to ensure incoming power is off.
2. Disconnect all interface cables (P1-P7) from each PDM and remove all covers. Disconnect all motor leads from the terminals marked R, S, and T. Verify with an ohmmeter that the resistance between terminals R to GND, S to GND, and T to GND is above 500k ohms. If the resistance is OK, reconnect the motor leads to the terminals marked R, S, and T. Verify that the DC bus wires are connected to the PDM with the correct polarity.

3. Make sure that each PDM has the Personality Module (PM) properly configured. If necessary, refer to the Applying Power chapter in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for step-by-step instructions on configuring a Personality Module using IQ Master software.
4. Install all covers. Connect all interface cables (P1-P7) to the PDMs.
5. Make sure that the motor is not connected to the load when applying power to prevent any mechanical damage in case of a fault. Apply input power with all PDMs disabled and check the bi-color status LED on the PDM. The LED should turn green. If the status LED is red rather than green, the PDM is in a fault condition. Select Status from the Monitor menu to find out the type of fault. Correct the fault (the status LED will turn green when the fault has been cleared and a hard reset performed). Check that the fans are operating. The green PSM READY LED should turn on and remain on.
6. Select Encoder from the Diagnostics menu. Verify that Counts increases positively when the motor shaft is rotated clockwise as viewed facing the motor shaft.
7. Remove input power. The PDM and PSM LEDs should turn off within one second. Measure DC bus to ensure voltage is below 15 VDC.

---

## Motor Start-Up Procedure

This motor start-up procedure is for ULTRA Plus or IQ-2000 and IQ-5000 systems only.

### ATTENTION

Motors can cause extensive damage and injury if mounted improperly.



1. The motor(s) should be disconnected from the mechanical load(s) when initially checking out the system. If this is not possible, take adequate precautions in case of a fault.
2. Disconnect all interface cables from the PDM and remove the cover. Connect the 4 motor wires (R, S, T, and ground) to the proper PDM. Install all covers and connect all interface cables to the PDM. With each PDM disabled, re-apply input power and check for proper power-up diagnostics as shown by the status LED on each unit.
3. Select Gains/Limits from the Parameter menu. Set Ilimit to a low value (25% of peak current is a typical value). Select Velocity/Acceleration from the Parameter menu and set the Jog Velocity to a relatively low speed, such as 50 RPM. Enable each PDM and make sure there is no shaft motion. Activate the Jog forward input and check for clockwise rotation of each motor shaft as viewed facing the motor shaft. If motor does not turn clockwise, then check that motor power wires and encoder wires are connected properly.
4. Disable all PDMs and turn off input power. Measure the DC bus voltage to be sure that it is below 15 VDC.
5. Now that the system installation is verified, return the Ilimit parameter to a value to suit the application.

## Start-Up Procedure for IQ-550 Systems

1. Measure voltage between IQ-550 terminals marked L1 and L2 to ensure incoming power is off. The bi-color status LED should be off.
2. Disconnect input wires connected to terminals L1 and L2. Arrange the wires in a safe position to test the incoming voltage.
3. Turn input power on and measure line voltage to ensure that it is in the proper voltage range (115/230 VAC).
4. Turn power off and reconnect the wires to terminals L1 and L2.
5. Connect all interface cables (P1-P7) to the IQ-550.
6. Turn power on.
7. Make sure that the IQ-550 has the Personality Module (PM) properly configured. If necessary, refer to the Applying Power chapter in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for step-by-step instructions on configuring a Personality Module using IQ Master software.

Before proceeding from this point, it is important to understand that the ULTRA Plus/IQ2000/IQ5000 needs to receive positive counts from its connected system when delivering a positive current command to that system. It is assumed that the amplifier and motor have already been tested to determine direction of rotation for a positive current command input. Consult your drive/amplifier hardware manual to determine the appropriate procedure for establishing the relationship between the polarity of the current command and the direction of the rotation of the motor.

### ATTENTION



Failure to determine the proper relationship between current command and encoder feedback can result in a runaway condition which may cause damage to equipment and/or personal injury.

8. Ensure the motor is not connected to the load when applying power. This prevents any mechanical damage in case of a fault. Apply input power with the ULTRA Plus/IQ2000/IQ5000 disabled and check the I-color Status LED.
  - Verify the status LED is green.
  - If the status LED is red, the ULTRA Plus/IQ2000/IQ5000 is in a fault condition. Select Status from the Monitor menu to find out the type of fault. Correct the fault (the status LED will turn green when the fault has been cleared and a hard reset performed).
9. Select Encoder from the Diagnostics menu.
  - Verify that the Counts value increases positively when the motor shaft is rotated in the direction (clockwise or counterclockwise) established as the positive current command.
  - If the Counts value decreases, switch A+ and A- on connector P5 (or B+ and B-).

### ATTENTION



Changing the rotation parameter in the Parameter menu, system dialog box of IQ Master will *not* adjust for polarity mismatches in the current command signal and the encoder feedback.

10. Enable the ULTRA Plus/IQ2000/IQ5000 using the Enable input or Enable from the Run menu of IQ Master. Verify the motor is stationary and has holding torque.
11. Command motion using the Jog inputs or the Run Control dialog box of IQ Master. Verify the motor rotates in the proper direction.
12. Remove input power. The LED should turn off within one second.

---

## Operator Terminal

The Operator Terminal requires no special start-up procedure. Simply apply power to the operator terminal as described in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004). The operator terminal should sound a tone on power-up and the display LED should be lit. No message will appear on the screen unless the operator terminal is connected to the ULTRA Plus or IQ serial port and the ULTRA Plus/IQ sends a message to the operator terminal.

---

## Backup Personality Module

After correctly setting up the system, back up the contents of the Personality Module (NVRAM) to a file on your PC, using File Transfer. Refer to Part 2 • IQ Master Environment, File menu, Transfer for details.





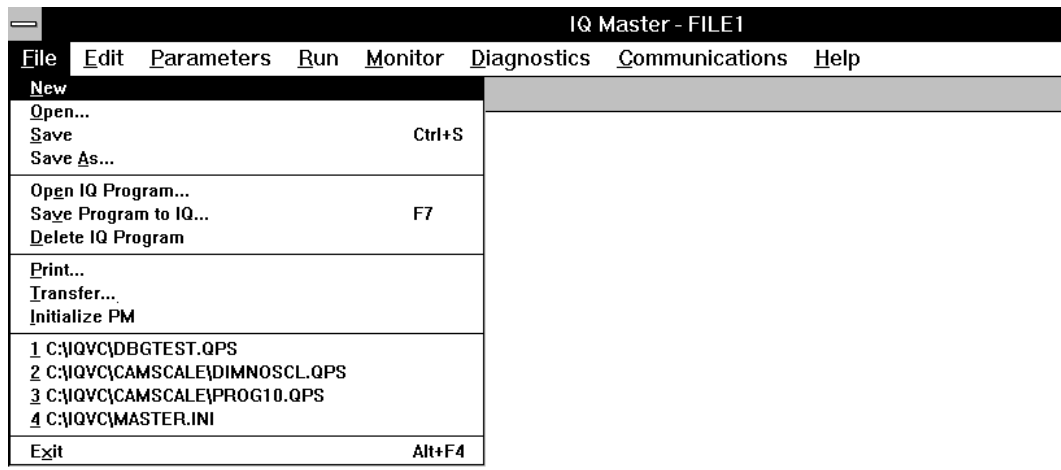
# Part 2

# IQ Master Environment

In this part of the manual, each menu and menu item is examined. The figures with menus and dialog boxes depict images from IQ Master.

**MENU**

# File Menu




The File menu provides all the functions to load (open) and save files to both the PC and the ULTRA Plus or IQ-Series controller. Along with these file commands, the menu contains the Print, Transfer, Initialize (program) Personality Module, and Exit functions.

## To Access File Menu Items

- Select the File menu by clicking on File in the menu bar,  
-or-  
press ALT to activate the menu bar, then F to pull down the File menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the “S” in Save),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

## PC Disk Commands

### New


Select New from the File menu or click the  toolbar button to open (create) a new file on the PC. The default file name will be FILE1.QPS, the next one FILE2.QPS, and so on. After editing the program, use Save As to save the file to disk under a name you choose.

To save the program to the ULTRA Plus or IQ, use Save Program to IQ (compile the program first if you want to save the executable as well as the source).

### Open

Select Open from the File menu or click the  toolbar button to open an existing file on the PC. Select the file to be opened from the Open file dialog box.

### Save

Select Save from the File menu or click the  toolbar button to save the current file as it is named on the PC.

### Save As

Select Save As from the File menu to save the current file under a different name on the PC. This can be used to rename your program or to copy your program.

## IQ Commands

### Open IQ Program


Select Open IQ Program from the File menu to open an existing source file in the ULTRA Plus or IQ-Series controller. Select the file to be opened from the IQ Directory dialog box. The file to be opened must be a source file—executable files cannot be edited.

The IQ Directory dialog box shows current programs that exist in the ULTRA Plus or IQ. The title of the program is shown in the Program List box if the file has been compiled with a Title statement. Select the program from the Program List box or, if you know the program number, enter the number in the Program Number box.

The Directory drop down list box is used to change from Program Directory to System Directory. The Program Directory contains user programs; the System Directory contains System Programs—programs such as Jog, Home, Start, Stop, etc., that are used as predefined functions. The Auto Program is also in the System Directory (program #0). The System Programs can be run by assigning an Fkey on the Operator Terminal to the program (use Fkey Set Up), or by a predefined input in specific cases such as Home or Emergency Return.

After opening an existing source file, if you want to save the file to disk on the PC, use Save As and specify the file name.

### Save Program to IQ (F7)

Select Save Program to IQ from the File menu, click the  toolbar button, or press F7 to save the current file to the ULTRA Plus or IQ. Then select the program number to store the file in the IQ Directory dialog box. Choose the Executable radio button to save only the executable program (if the file has been compiled with no errors). To save both the source and executable programs, choose the Source and Executable radio button. (If the program has not been compiled successfully, only the source will be saved.)

## Delete IQ Program

Select Delete IQ Program from the File menu to delete a file in the ULTRA Plus or IQ. Select the type of program to be deleted—either just the source or both source and executable by selecting the appropriate radio button. Then select the program to be deleted from the IQ Directory dialog box.

---


## File Extensions

IQ Master uses default file extensions for specific types of files, however, any file may be saved with any extension desired. All default file extensions begin with the letter Q to indicate a file used with ULTRA Plus or IQ-Series products and IQ Master. The default file extensions and a description of each is listed below.


QPS	IQ Program Source file. Files with a QPS extension should be IQ programs in ASCII text format.
QPE	IQ Program Executable file. Files with a QPE extension are the files generated by the compiler. These ASCII hex files contain the compiled programs which the ULTRA Plus or IQ executes.
QCS	IQ Cam profile Source file. Files with a QCS extension should be IQ Cam profiles in ASCII text format.
QCE	IQ Cam profile Executable file. Files with a QCE extension are the files generated by the Cam compiler. These ASCII hex files contain the compiled Cam profiles which the ULTRA Plus or IQ executes.
QPA	IQ Parameter file. Files with a QPA extension are text files in HEX format which contain the ULTRA Plus or IQ parameter settings.
QAP	QAPIQ All Program file. Files with a QAP extension are text files in HEX format which contain all programs (including system programs) from memory.
QPM	Personality Module file. Files with a QPM extension are text files in HEX format which contain the entire contents of the Personality Module nonvolatile memory. This includes programs, parameters, G and B variable values, and motor/drive data.

## Miscellaneous Commands

### Print

Select Print from the File menu or click the  toolbar button to print the current file to the local printer. The Windows Print Manager handles the printing, so if you exit IQ Master and close Windows while the file is still being printed, some of the file may not be printed.

### Transfer

An ASCII file may be transferred between the ULTRA Plus or IQ-Series controller and the PC. To bring up the Transfer dialog box, Select Transfer from the File menu or click the  toolbar button.

Note: Use Save Program to IQ to save the current file that is being edited to the IQ – do *not* use Transfer. When Transfer is used to send files to the controller, the files that are saved to the controller are files that currently exist on the PC disk. Save Program to IQ saves files that are currently in memory (the current source file being edited and compiled).

If Transfer is used on a source file that has been edited but not saved, the file that is saved to the IQ-Series controller will be the file that is on disk—that is, the previous version of the file without the latest changes. If the source file being edited is then compiled with Compile Options set to Compile to Disk, and Transfer is used to save the executable file to the controller, the source and executable programs on the controller will be different versions.

#### *To transfer a file*

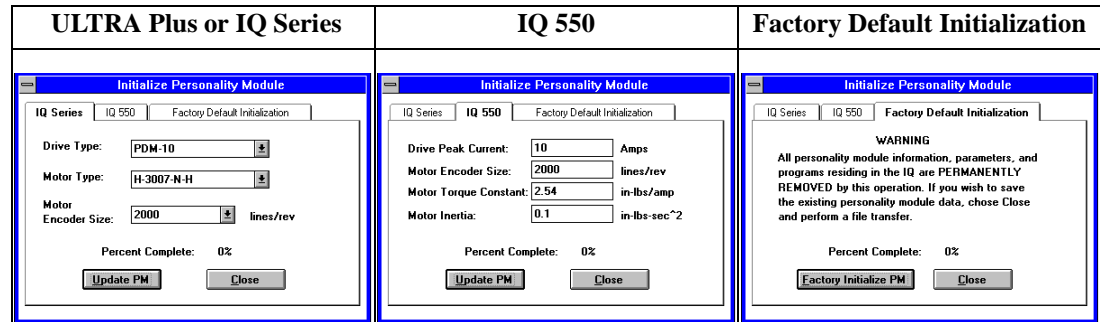
- Select Transfer from the File menu.
- Select the type of file to be transferred from the Transfer Type box: Program, Parameters, NVRAM (all the controller memory), or All Programs.
- Select the Transfer Mode: Receive from IQ or Send to IQ.
- Choose the Start button to select the file source or destination PC file name from the Open file dialog box. If transferring Parameters, NVRAM, or All Programs, the transfer will start after you select the file and choose the OK button.
- If the Transfer Type is Program, after you select the PC file name, select the program type, source or executable, by selecting the appropriate radio button from the IQ Directory dialog box. Then select the program to be transferred.

### Initialize PM

Select Initialize PM from the File menu to initialize the Personality Module. Initialize PM allows the user to configure the drive and motor through software selections, or to revert to factory default settings for the PM. The initial dialog box provides three tabbed choices:

- IQ Series (ULTRA Plus)
- IQ 550
- Factory Default Initialization

The subsequent dialog boxes provide list box choices from which the PM parameters are matched to the controller and motor for the application.



## Personality Module Dialog Boxes

### Recently Used File List

Use the numbers and filenames listed at the bottom of the File menu to open any of the last four most recently opened files. Choose the number that corresponds to the file you want to open.

### Send Mail

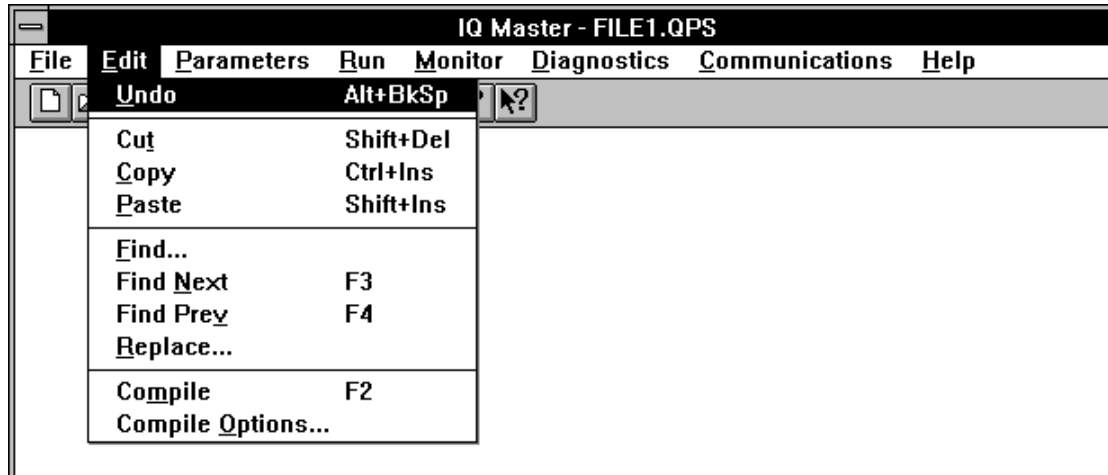
Select Send Mail from the File menu to send a mail message with the active document attached. This command opens a Send Note window in Microsoft Mail. The command is available only if you have installed Microsoft Mail version 3.0 or later (if Mail is not installed on your computer, the Send Mail item will not appear on the File menu).

### Exit

Select Exit from the File menu to quit IQ Master and return control to Windows. You will be warned if you are editing a file and have not saved it since it was changed. You will be able to save the file before exiting, or cancel the exit and return to IQ Master.

# Edit Menu

The Edit menu will not be available unless a file is opened for editing.



The Editor is used to create or edit motion programs for the ULTRA Plus or IQ-Series controller. The Edit menu provides functions to cut, copy, and paste text to and from the Clipboard. You can also search for and replace text. The current program line number (where the cursor is positioned) is shown in the status bar and is updated as you scroll through the program. The program is compiled from within the Editor so any syntax errors can be corrected before storing the file.

## To Access Edit Menu Items

- Select the Edit menu by clicking on Edit in the menu bar,  
-or-  
press ALT to activate the menu bar, then E to pull down the Edit menu.
- Select the menu item by clicking on your selection,  
-or-



pressing the letter that is underlined in your selection (for example the “t” in Cut),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

---

## Undo

Undo cancels the most recent command or action you completed. If you do not like the results of a command or accidentally delete some text, choose Undo as the next action.

---

## Cutting and Pasting Text—*Selecting Text*

### Selecting Text

Select text by holding the left mouse button and dragging the mouse over the text. To select text with the keyboard, position the cursor at the start of the text. Then press SHIFT and move the arrow keys, Home, or End in the desired direction.

### Clipboard

The Clipboard is a utility that stores text or graphics that have been cut or copied from an application. Once text has been cut or copied to the Clipboard, it can be pasted into other applications that support the Clipboard or another location in your program.


### Cut (SHIFT+Delete)

Select the text to be cut, then select Cut from the Edit menu, click the  toolbar button, or press SHIFT+DELETE. The text is cut from your program AND copied to the Clipboard.


To delete text, select the text, then press the Delete key. Deleting text does NOT copy the text to the Clipboard. If you want to delete text AND copy it to the Clipboard, use Cut.

Use the Delete key to delete text only and NOT copy to the Clipboard when you want to preserve what is in the Clipboard, but need to delete text.

### Copy (CTRL+Insert)

Select the text to be copied, then select Copy from the Edit menu, click the  toolbar button, or press CTRL+INSERT. The text is copied to the Clipboard and can be pasted into your document or into any other application that supports the Clipboard.

### Paste (SHIFT+Insert)

Place the cursor where the text is to be inserted, then select Paste from the Edit menu, click the  toolbar button, or press SHIFT+INSERT. The text will be copied from the Clipboard into the text. Text must have been cut or copied to the Clipboard to use Paste.

**Tip:** If text is selected, and you choose paste in order to paste something from the clipboard, the text that is selected will be replaced (deleted).

## Finding Text

### Find

Select Find from the Edit menu to search for text in the current text file. The search can be case sensitive or insensitive. The direction of the search can be forward to the end of the file or backward to the beginning of the file.

### Find Next (F3)

Select Find Next from the Edit menu, or press F3, to find the next occurrence of the search text in the current file.


### Find Previous (F4)

Select Find Prev from the Edit menu, or press F4, to find the previous occurrence of the search text in the current file.

### Replace


Select Replace from the Edit menu to search for text in the current text file, and replace it with different text. The search can be case sensitive or insensitive. The direction of the search is always forward to the end of the file.

## The Compiler

The Compiler is called from within the Editor to compile program statements into opcodes that the controller will use to run the motion program. The Compiler will check language syntax, verify that jump target labels are provided, verify that subroutines exist if called, and, check that variable names are valid. If any errors or warnings are found, they are displayed in the Compile Results dialog box showing the number of errors and warnings generated. A Compiler Results Window is then opened that shows the line number(s) where the error(s) occurred. When editing a program, the line number is shown on the status line, in the lower right hand corner of the screen. When the file is compiled successfully, it can then be transferred to the controller using Save Program to IQ in the Compile Results dialog box, or Save Program to IQ under the File menu, or by clicking the  toolbar button.

**Note:** The Compiler cannot check for logic errors.

### Compiling a Program

- Open a file for editing.
- Select Compile from the Edit menu, click the  toolbar button, or press F2. The Compile Results box will be displayed showing any warnings or errors.

If the program compiled successfully, the compiled program can be directly saved to the ULTRA Plus or IQ by selecting the Save Program to IQ in the Compile Results box. To print the results of the compile session to a file, see Compiler Options.

---

## Compile Options

The Compile Options item under the Edit menu allows you to choose what type of program to compile, generate a list file, include debug information in the compiled file, expand macros, and whether to compile the file to disk or memory. The default compiler settings may be overridden by using compiler options program statements such as PGMTYPE, DEBUG, etc. Refer to Part 5 • Language Reference for more detailed information.

### Compiler Output

If Compile to Memory is selected, the executable program is not stored on the PC hard disk. If Compile to Disk is selected, the executable code is saved to the hard disk with the same name as the source program and a QPE extension. The compiling operation is faster if Compile to Memory is selected. Choose Compile to Disk if you intend to save the executable code and transfer it to a controller later.

### List Files

To create a program list file, select the Compile to Disk button, then select the Generate List File check box. The list file is saved to disk with same filename as the source program and a LST extension. Line numbers will be added to each line of the program, and any compile errors will be shown in the list file.

### Expand Macros

The Expand Macros selection is only used if Generate List File is selected. Select the Expand Macros check box to generate the actual text strings that have been assigned using the ASSIGN statement. If you do not expand macros, the name assigned to the text string will be shown in the list file instead of the text wherever the text string is referenced. If you expand macros you can then look at the list file and read your program as the compiler has interpreted your ASSIGN statements.

### Program Type

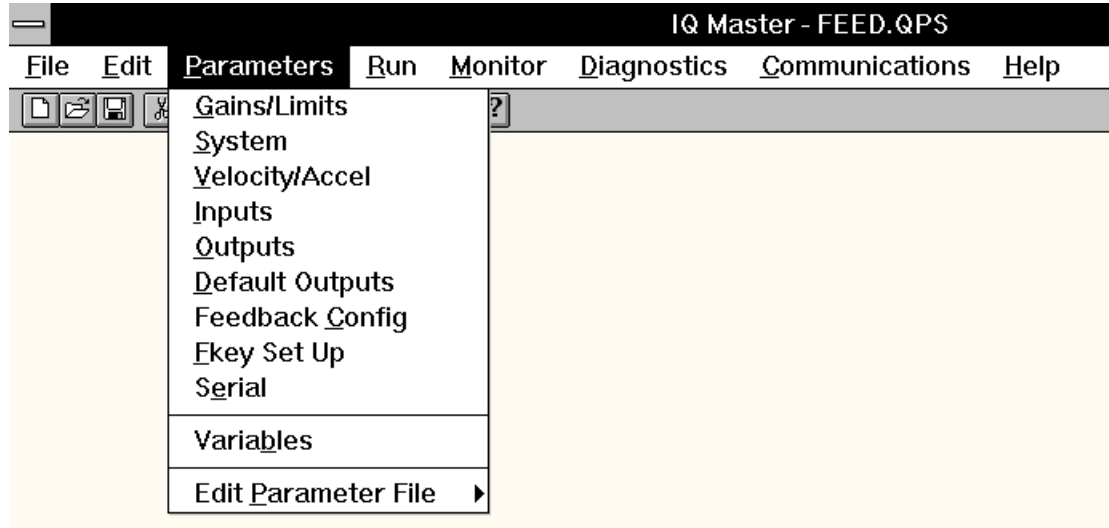
Select the type of program you want to compile from the Program Type box: Main (normal programs), Auto (Auto Program), Fkey Routine (Function key routines for the Operator Terminal), Error (a special program that executes when there is a error detected), or Cam Profile. This selection needs to be made because not all of the commands are valid in all program types. This setting may be overridden by using the PGMTYPE program statement.

### Debug Information

Select the Generate Debug Information check box to generate program status information used in the Status dialog box and to allow Single Stepping from the Run Control dialog box. (Note that this increases executable program size by 5 bytes per line.)

Each line of a program can be monitored while executing programs compiled with debug information. You will also be able to step through your program one line at a time. This is very useful when trying to debug a program that is not functioning properly or that is stopping unexpectedly. This setting may be overridden by using the DEBUG program statement.

# Parameter Menu



The Parameter menu contains the parameter values used to control the ULTRA Plus or IQ system. The values are stored in a parameter table in nonvolatile RAM (NVRAM). All parameters should be reviewed, entered or changed based on the application. The parameters in the table are used unless overridden locally in a program. If a program does override parameters, the values in the parameter table are used once again when the program is stopped.

## To Access Parameter Menu Items

- Select the Parameter menu by clicking on Parameter in the menu bar,  
-or-  
press ALT to activate the menu bar, then P to pull down the Parameter menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the “O” in Outputs),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

## Gains/Limits

Gains are used to adjust both the velocity and position loop performance. The velocity loop gains should be set up first, and then the position loop gains can be set. Change gains with care as excessive settings on any gain can result in motor instability.

After changing any Gain values, choose OK to store the new values and close the dialog box, choose Update to store the new values, but still leave the Gains/Limits dialog box displayed or choose Cancel to leave the dialog box without saving changes. The new values are not active until OK or Update is selected.

Name	Description	Detail Reference
<b>FILTER</b>	Select the FILTER ON check box to enable the low pass filter on the output of the velocity regulator.	Part 5 Language Reference FILTER
<b>PGAIN</b>	PGAIN is the proportional gain of the velocity regulator.	Part 5 Language Reference PGAIN
<b>IGAIN</b>	IGAIN is the velocity regulator integral gain.	Part 5 Language Reference IGAIN
<b>FGAIN</b>	FGAIN is the acceleration feedforward gain.	Part 5 Language Reference FGAIN
<b>KP</b>	KP is the position regulator proportional gain.	Part 5 Language Reference KP
<b>KPZ</b>	KPZ is the position regulator proportional gain used when the system is within the region of the commanded position defined by PZONE.	Part 5 Language Reference KPZ
<b>PZONE</b>	PZONE is the region around the commanded position where the position loop proportional gain is changed to the gain set by the KPZ parameter.	Part 5 Language Reference PZONE
<b>KFF</b>	KFF is the velocity feedforward gain.	Part 5 Language Reference KFF
<b>KI</b>	KI is the position regulator integral gain.	Part 5 Language Reference KI
<b>IZONE</b>	IZONE is the region around the commanded position where the position loop integral gain is active.	Part 5 Language Reference IZONE
<b>In-Position Window</b>	In-Position Window defines the maximum position error that is used to determine if the motor is In-Position.	Part 5 Language Reference WIN

<b>Following Error Limit</b>	Following Error Limit sets the maximum allowable following error and time for fault recognition (following error is defined as the difference between commanded position and actual position). There are two parameters associated with the Following Error Limit: limit size and time.	Part 5 Language Reference FEL and FET
<b>ILIMIT</b>	ILIMIT is the maximum current limit in amperes. The maximum current to the motor will not exceed this value.	Part 5 Language Reference ILIMIT
<b>I AVG</b>	I AVG is the average current trip point for the ULTRA Plus or IQ in amperes (typically set to the motors continuous current rating or less).	Part 5 Language Reference I AVG

## System

System parameters include values that describe the mechanical environment in which the motor is operating. The System parameters will generally not be changed once set. The settings should be reviewed and set based on the application.

To save the values in the dialog box and close the dialog, choose OK. To leave without saving the values, choose Cancel.

### Absolute Mode

If Absolute Mode is ON (checked), the following features will be enabled: Software Travel Limits, the At Home Flag, the At Home Output and the Emergency Return function. These features will not operate until a home position has been defined. Refer to Appendix A, System Programs, Home for more detailed information on defining a home position.

**Note:** In addition to the Absolute mode setting, the At Home output must also be assigned if it is to function. Refer to Part 2 • IQ Master Environment, Parameters, Outputs, for more detailed information on At Home.

### Forward Software Travel Limit

Forward Limit sets the forward software travel limit. The limit is enabled when the Absolute mode parameter is on. If travel exceeds this value in the forward direction, an error occurs and the ULTRA Plus or IQ is disabled. Jog commands may be used to move off a software travel limit. Forward Limit is entered as user units (defined by the Scale parameter). The maximum value is  $\pm 2,147,483,648$  (2<sup>31</sup>) encoder counts.

### Reverse Software Travel Limit

Reverse Limit sets the reverse software travel limit. The limit is enabled when the Absolute mode parameter is on. If travel exceeds this value in the reverse direction, an error occurs and the ULTRA Plus or IQ is disabled. Reverse Limit is entered as user units.

### Ereturn Position

ERETURN Position specifies the position the system moves to when the Emergency Return input is activated. The ERETURN Position is entered as user units. ERETURN is only active while in Absolute mode. For more detailed information on this value refer to Part 5 • Language Reference, and to Appendix A • System Programs, Emergency Return.

## Scale

SCALE is the number of feedback counts for 1 user unit. The feedback counts can be generated from either encoder 1 or encoder 2 as specified in the Feedback Configuration dialog box. For more detailed information on this value, refer to Part 5 • Language Reference.

## Scale2

SCALE2 is the number of Encoder 2 counts for 1 user unit of distance when Encoder 2 is not used for position feedback. When Encoder 2 is used for position feedback SCALE is used. See Scale above for how to determine the proper scaling. For more detailed information on this value refer to Part 5 • Language Reference.

## Disable on Fault

On some machines you may want the ULTRA Plus or IQ to continue running even after some faults occur. This is accomplished by setting Disable on Fault to Partial. In this case, the following faults are disabled: Iavg fault, Motor Overtemperature, Soft Forward Limit, and Soft Reverse Limit. All faults will cause the Error output to turn ON (if enabled). However, if Disable on Fault is set to Partial, and a fault listed above occurs, the ULTRA Plus or IQ will remain enabled and execute motion programs normally. The Error output in this case serves as an alarm, and a programmable logic controller (PLC) or other controller can then gracefully shut down the machine without damaging the tooling or work piece. This setting is made in the Parameter menu, System dialog box of IQ Master, or with the FLTDIS Host Language Command. Refer to Part 6 • Host Language Commands.

### ATTENTION



Use caution when setting Disable on Fault to Partial. This defeats the fault safety protection of the ULTRA Plus or IQ. For example, if Disable on Fault is set to Partial and the ULTRA Plus or IQ exceeds the Iavg value for an extended period of time, the ULTRA Plus or IQ could be damaged because it will not be disabled due to the fault.

## Rotation

Rotation determines the positive direction of the Encoder 1 input (motor). The parameter can be set to clockwise (CW) or counterclockwise (CCW). If set to CW, a clockwise rotation of the motor as viewed from the shaft end is defined as the positive (or forward) direction and counterclockwise rotation is defined as the negative (or reverse) direction. Conversely, if Rotation is set to CCW, counterclockwise rotation is defined as the positive direction and clockwise rotation as the negative direction. This parameter is used to define which rotation direction of the motor is to be considered the positive direction.

## Rotation 2

Rotation2 sets the positive direction of rotation for the Encoder 2 input as either clockwise (CW) or counterclockwise (CCW).

## Latched Position

Latched Position selects either Pos1 (encoder 1—default) or Pos2 (encoder 2) as the input to the hardware position latch. For more detailed information on this value, refer to Part 5 • Language Reference, LPOS.

**Note:** If the Latched Position source is changed, the change will not take effect until the next power-up or Hard Reset.

## Velocity & Acceleration

Velocity, Acceleration, and Timebase settings are set in the Velocity/Acceleration dialog box. These values are the default values which are used if motion commands or programs do not use different settings. The ULTRA Plus or IQ is shipped from the factory with values for each of the settings. Verify that each of the settings is appropriate for the application before operating the system. After changing any settings, choose OK to store the new settings and close the dialog box. Choose Update to store the new settings, but still leave the dialog box displayed. Choose Cancel to leave the dialog box without saving any changes. Refer to Part 5 • Language Reference for more detailed information on each of these values, except Timebase.

Name	Description	Detail Reference
<b>Velocity</b>	Velocity is the default velocity used to calculate motion profiles, MOVD, MOVF.	Part 5 Language Reference VEL
<b>Acceleration</b>	Acceleration is the default acceleration for all motion generated by the ULTRA Plus or IQ-Series controller except jogs and stick moves. Acceleration and deceleration rates for motion profiles and the home program are specified by this parameter. Units for this parameter are user units per second per second.	Part 5 Language Reference ACCEL
<b>Feedrate</b>	Feedrate scales the Timebase for motion. With the Feedrate set at 100%, all velocities and dwells are at the programmed rates. Feedrate can be set less than 100% to slow down a process, or above 100% to speed it up. The maximum value for the feedrate is 200%.  <b>Note:</b> If the analog input ADC1 is selected as a feedrate input, the feedrate will be determined by the analog input instead of the Feedrate parameter.	Part 5 Language Reference FDR
<b>Slew</b>	Slew enables the slew rate limit for the gear output.	Part 5 Language Reference SLEW
<b>Timebase</b>	Timebase sets the units for all velocity values. If set to seconds, velocity units are user units per second. If set to minutes, velocity units are user units per minute.  <b>Note:</b> Timebase only affects velocity statements and Overspeed.	Part 5 Language Reference TBASE
<b>Jog Velocity</b>	Jog Velocity is the velocity used for jog commands.	Part 5 Language Reference JVEL
<b>Jog Acceleration</b>	Jog Acceleration is the acceleration used for jog commands and MOVF.	Part 5 Language Reference JACCEL



<b>Jog Deceleration</b>	Jog Deceleration is the deceleration used for jog commands and MOVV.	Part 5 Language Reference JDECEL
<b>Home Velocity</b>	Home Velocity defines the velocity for the Home Command.	Part 5 Language Reference HVEL, Appendix A
<b>Home Offset</b>	Home Offset specifies the distance from encoder index to home position.	Part 5 Language Reference HOFFS, Appendix A
<b>Overspeed</b>	Overspeed is the overspeed trip point in user units per Timebase for the system. If this speed is exceeded, the ULTRA Plus or IQ will disable and indicate that an error has occurred.	Part 5 Language Reference OVERSPEED, Appendix A

## Inputs

All dedicated inputs can be disabled and used as general purpose inputs. The ULTRA Plus or IQ is shipped from the factory with values for each of the settings. Verify that each of the settings is appropriate for the application before operating the system. The default settings are listed with each description.

After changing any input settings, choose OK to store the new settings and close the dialog box. Choose Update to store the new settings, but still leave the Inputs dialog box displayed. Choose Cancel to leave the dialog box without saving any changes.

Name	Description	Detail Reference
<b>Enable (I3)</b>	The Enable input (I3) is used to enable and disable the ULTRA Plus or IQ. The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>Emergency Return (I10)</b>	The Emergency Return input (I10) is used to run system program number 26. The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>Home Command (I6)</b>	The Home Command input (I6) is used to run system program number 25. The default setting is enabled.	Appendix A Home Program
<b>Home Switch</b>	The Home Switch input (I5) can be used by the home program to identify the home position. The default setting is enabled.	Appendix A Home Program
<b>Jog Forward (I7) and Jog Reverse (I8)</b>	Jogs enable or disable the forward and reverse jog inputs (I7 and I8). The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>Forward Travel Limit (I1) and Reverse Travel Limit (I2)</b>	Limits enable or disable the forward and reverse limit switch inputs (I1 and I2). The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware

<b>Pause</b>	Pause enables or disables the Pause input (I9). The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>Start</b>	Start enables or disables the Start input (I4). The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware

### Home Switch Active State

Home Switch Active State selects the home switch input (I5) as active open or active closed. The default setting for this input is active closed. For details refer to Appendix A, Home Program.

### Pause Switch Active State

Pause Switch Active State selects the Pause input (I9) as active open or active closed. The default setting for this input is active closed.

### Selectable Inputs

There are three functions that allow an available input to be assigned to that function: Define Home, HRESET, and X (Kill motion). To assign an input to a function, enter the input number (range is 1 to 16). To disable a function, set the parameter to OFF by typing the word “OFF” in the selection box. The default setting for all of these inputs is OFF.

Name	Description	Detail Reference
<b>Define Home</b>	The Define Home parameter selects an input that is used to define the home position for the system.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>Hardware Reset</b>	The HRESET (Hardware Reset) parameter assigns an input to the hardware reset function.	Part 3 ULTRA Plus or IQ Software and Hardware
<b>X Kill motion</b>	The X (Kill motion) input enables and selects a stop input. This input halts program execution and holds position when turned on.  <b>Note:</b> The X input does not stop Jog input motion.	Part 3 ULTRA Plus or IQ Software and Hardware

### ADC1 Function

ADC1 selects the use of the analog to digital converter 1 (ADC1) input (P3-5) as either feedrate input or general purpose use (general purpose is the default setting). For general purpose use, the ADC1 input voltage can be -10 to +10 volts. If this input is used as feedrate input, the ADC1 input voltage can range from 0 to 10 volts, corresponding to 0 to 200% feedrate.

### Debounce Time

Debounce Time is the time interval an input signal that has changed must remain stable for the change to be recognized. Debounce Time applies to all inputs. The debounce time is entered as milliseconds (mS), up to a maximum of 255 mS. The default setting is 0 mS.

### Home to Encoder Index

Home to Encoder Index selects the encoder index pulse to indicate home position during a home routine. Disabling this option will cause the HSWEN flag to be OFF. The default setting is enabled. See Appendix A.

## Program Select Lines

Select Program from Inputs selects the program number to be run when a Start input is received, rather than the Default Run Program number. The range of program numbers is limited by the number of Program Select lines. If the Select Program from Inputs check box is selected, the Number of Program Select lines can be entered.

Number of Program Select Lines determines how many external inputs are used as program select lines. Up to 6 inputs can be used to select programs. Input I16 is always the most significant bit of the program select lines, input I15 is next significant and so on, down to input I11 if all 6 inputs are used. The program number is the binary number read on these inputs when the Start input is received. The least significant input specifies a value of 1, the next a value of 2, and so on continuing in multiples of 2 up to a value of 32 for the most significant input (I16) when all 6 inputs are used.

<u>Number of Program Select Lines</u>	<u>Possible Programs Selected</u>	<u>Inputs</u>
1	Up to 2 (0 or 1)	I16
2	Up to 4 (0 through 3)	I16 and I15
3	Up to 8 (0 through 7)	I16 through I14
4	Up to 16 (0 through 15)	I16 through I13
5	Up to 32 (0 through 31)	I16 through I12
6	Up to 64 (0 through 63)	I16 through I11

Example 1: Select Program from Inputs is selected. Program Select Lines is 3 (I16, I15 and I14).

Input states:	I16	I15	I14		
	ON	ON	OFF		
Binary Value:	4	+	2	+	0 = 6

This example would select program 6 (from 8 maximum) to be run when the Start input turns on.

Example 2: Select Program from Inputs is selected. Program Select Lines is 5 (I16 through I12).

Input states:	I16	I15	I14	I13	I12
	OFF	ON	ON	OFF	ON
Binary Value:	0	+	8	+	4 + 0 + 1 = 13

This example would select program 13 (from 32 maximum) to be run when the Start input turns on.

The Program Select lines must be in the proper state at least 2 ms before the Start input is received. Once the Start input turns on, the program is run. The Program Select lines may then change state and can be used for other functions in the program. For example, if using 6 program select lines, inputs I11 and I12 would be the two least significant Program Select lines. These inputs are also used as interrupt inputs INT1 and INT2. Once the program is started, I11 and I12 can be used normally as interrupt inputs.

The maximum number of programs that the ULTRA Plus or IQ supports is 32 without an optional Memory and I/O Expansion Card. If an Expansion card is present, space for an additional 32 programs is available (program numbers 32 through 63).

## Default Run Program

Default Run Program is used to set the program number that runs when the Start input is activated. Allowable values are 0 to 31 inclusive (0 to 63, if an option card with additional program memory is present). If the Select Program from Inputs check box is selected, the external program select inputs determine which program is run when the Start input is activated. The Select Program from Inputs check box must be cleared to enter a Default Run Program number.

## Outputs

All dedicated outputs can be disabled and used as general purpose outputs. The ULTRA Plus or IQ is shipped from the factory with values for each of the settings. Verify that each of the settings is appropriate for the application before operating the system. The default settings are listed with description.

After changing any output settings, choose OK to store the new settings and close the dialog box. Choose Update to store the new settings, but still leave the Outputs dialog box displayed. Choose Cancel to leave the dialog box without saving any changes.

Name	Description	Detail Reference
<b>ATHOME</b>	The AtHome output (O5) can be enabled to provide an output that indicates when the system is at the home position. The default setting is enabled.  <b>Note:</b> The system must have been homed AND must be in the Absolute mode for AtHome to turn on.	Part 3 ULTRA Plus or IQ Software and Hardware Integration
<b>Error</b>	The Error output (O8) can be enabled to indicate if there is a error present. If Error output when Disabled is selected, this output will also turn ON when the ULTRA Plus or IQ is disabled. The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware Integration
<b>Home Complete</b>	The Home Sequence Complete output (O6), if enabled, will turn on when the ULTRA Plus or IQ has been homed. The default setting is enabled.	Appendix A Home Program
<b>Program Running</b>	The Program Running output (O4), if enabled, will be on when a program is running and off when no program is running. The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware Integration
<b>In-Position</b>	The In-Position check box enables the In-Position output (O7). The default setting is enabled.	Part 3 ULTRA Plus or IQ Software and Hardware Integration

### Error Output when Disabled

Error output when Disabled selects whether the Error output turns ON when the ULTRA Plus or IQ is disabled. The ULTRA Plus or IQ can disable due to the Enable input, a disable command from the serial port or program, or an error detected in the system. The default setting is disabled.

### In-Position Mode

In-Position Mode determines operation of the In-Position output (O7). If Relative is selected, the In-Position output will be activated anytime the system is within the In-Position Window, even if it is in motion. If Absolute is selected, the In-Position output will be turned on only when the system is within the In-Position Window and not in motion. The default setting is Absolute.

### DAC1 Value

DAC1 Value is the default value for the 12 bit Digital to Analog Converter (DAC1) output (P3-6) in volts. This is the value the DAC1 output is set to on power-up and when a program is not running. The range is  $\pm 10$  volts with about 5 mV resolution. The default setting is 0 volts.

## Default Outputs

The default outputs settings determine the state of the outputs when a program is stopped. Each output can be set to either turn ON, turn OFF, or remain unchanged when program execution is halted. These settings will affect the output if the program stops for any reason (loss of enable, an error, or the end of program reached). Choose the Next Bank button to set the next 8 outputs. The state for outputs 9-24 can be set, regardless of whether an option board is installed at the time the settings are made.

After changing any settings, choose OK to store the new settings and close the dialog box. Choose Cancel to leave the dialog box without saving any changes.

### ATTENTION



If an output is assigned a default of ON, the system E-stop circuitry should override the controller's outputs if the state of the output presents a safety hazard to personnel.

### ATTENTION



If an output is assigned a default of ON, that output will turn on when power is applied to the ULTRA Plus or IQ, following execution of the Auto Program (system program 0). If there is no Auto Program, the output will remain off.

## Feedback Configuration

The Feedback Configuration determines the source of the feedback signals and the gear input. Also, connector P4 can be set up to output Encoder 1 signals, to input Encoder 2 signals, or as a step and direction input. Up to 4 different configuration menus can be stored in the ULTRA Plus or IQ. The configuration can be changed at any time and can even be changed from within a program.

### IMPORTANT

Changing feedback configuration when the ULTRA Plus or IQ is enabled could result in unexpected machine motion. Disable the ULTRA Plus or IQ before changing the feedback configuration. Failure to observe this precaution could result in bodily injury.

Once a Feedback Configuration menu has been changed, the active configuration can be set by choosing the Set Config button from the Feedback Configuration dialog box or by issuing a "CF = n" command to the ULTRA Plus or IQ (either from a program or from the Terminal). The ULTRA Plus or IQ defaults to Configuration #1 on power up.

To save all changes and close the dialog box, choose the OK button. If you want to save the changes AND set the current menu shown to be the active configuration, choose the Set Config button. Choose Cancel to leave the dialog box without saving any changes. The ULTRA Plus or IQ defaults to Configuration # 1 on power up. The active configuration can also be changed in a program or in the Terminal with the "CF = n" command where n is the new configuration number.

## Configuration Menu

Configuration menu selects the menu number to be set up. Enter the number of the menu to be set up and the values for that menu will be shown.

## Current Configuration

Current Config shows the active configuration menu number.

## Feedback Position Source

Feedback Position selects the source for the position feedback, POSN. The choices are Pos1 (Encoder 1—default), Pos2 (Encoder 2), or Pos3 (Encoder 3—available only with an option card).

When Encoder 2 is used for position feedback, SCALE is used for units, not SCALE2. Any parameters which are scaled position values may need to be changed. This includes: Kp, Overspeed, FEL, and In-Position Window.

## Encoder Input 2

Encoder Input 2 selects signals on connector P4 as an input for Encoder 2 (default) or as an output of Encoder 1 signals.

Select Step and Direction signals to bring step and direction signals into the ULTRA Plus or IQ from an indexer. The ULTRA Plus or IQ may continue to output Encoder 1 signals while receiving step and direction inputs.

## Gear Input

Gear Input selects the source for the gear input. The choices are Pos2, Pos3, the sum of both signals, or none (default). Pos3 will be not be able to be selected unless the absolute encoder interface option card is installed.

---

## Fkey Set Up

The only setup required for the Operator Terminal is for the soft function keys (Fkeys) F1 through F4. The function keys can be programmed to perform different predefined functions or functions defined by you. Text can be assigned to the keys to show their specific function on the Operator Terminal screen. The Operator Terminal status displays are enabled simply by connecting the Operator Terminal to Serial Port 1 (P7).

### Mode

Up to six sets of four functions can be assigned to Fkeys. Each set of four is referred to as a mode. While assigning the functions to the keys, the mode is changed by typing a new number into the text box. The active mode may be changed by pressing the MODE key or by using the PRINT statement to send a code to the operator terminal.

To set the Fkey mode to be displayed on the Operator Terminal from a program, use a PRINT “^Cn” statement where *n* corresponds to the Fkey mode number. The number *n* is zero-based—that is, 0 corresponds to Fkey Mode #1, 1 corresponds to Fkey Mode #2, and so on. Refer to Part 5 • Language Reference, PRINT for details.

The Mode key on the Operator Terminal can also be used to set an Fkey mode. Press the Mode key to step through as many modes as are defined in the Fkey Set Up dialog box. If no Fkey functions are assigned to a mode, the Operator Terminal will pass over that mode and move to the next mode that has Fkeys defined. After the MODE key is pressed and the Fkey labels are displayed, the labels can be cleared from the terminal screen by pressing the CLEAR key.

## To Program the Soft Function Keys on the Operator Terminal

- Select Fkey Set Up from the Parameter menu.
- Enter the Mode number (up to 6 different modes or function key definitions are available).
- To assign a preprogrammed function to one of the function keys, select the function using the drop down list box for each function key you want to define. (If not using a mouse, press TAB to get to the desired drop down box, then press ALT+DOWN ARROW to open the box. Once the box is opened, use the UP and DOWN ARROWS to scroll through the selections. To make a selection, press the ALT+UP ARROW or ALT+DOWN ARROW to select the highlighted item.)

The functions shown in the drop down list box are Fkey programs. The Fkey programs are a subset of the programs (programs 1 through 24) in the System Directory. (User programs are in the Program Directory.) Some of the program numbers are blank when the ULTRA Plus or IQ is shipped. The blank program numbers can still be assigned to a function key. This allows function key assignments to be made and the programs created later.

- Enter the text label for each function key in the Fkey Text box. The text label is displayed above the function keys on the Operator Terminal and can be up to 20 characters long (5 characters per Fkey).
- Choose the Update button to save the new function assignments for that mode, then select a new mode using the Mode # box.  
-or-  
Choose the OK button to store the new function assignments and close the dialog box.

- Note:**
1. The labels can be cleared by pressing the STATUS key, which calls up one of the status displays.
  2. The Fkeys that monitor a variable will display the variable selected on the screen until the same Fkey is pressed a second time.
  3. It is not necessary to assign a function to every Fkey.

---

## Serial

The Serial menu controls the serial communication parameters for the IQ. To set the serial parameters for the PC, use PC Set Up under the Communications menu.

The parameters for Port 1 (Operator Terminal on P7) or Port 2 (Programming Terminal on P6) can be set independently of each other.

After changing any settings, choose OK to store the new settings and close the dialog box. Choose Cancel to leave the dialog box without saving any changes.

**Note:** If any of the IQ serial parameters are changed, the new parameter settings will not take effect until the next power-up or hard reset.

### Baud Rate

Baud Rate can be set to: 1200, 2400, 4800, 9600 (default), or 19200..

### Parity

Parity can be set to none (default), even, or odd.

**Note:** The number of data bits in the IQ is fixed at eight. If parity is used, the parity bit will be the eighth data bit. The number of data bits should be set to seven when using parity.

## Communications Mode (RS-232C/RS-422)

The communications Mode can be set to RS-232C (default) for single axis applications with cable lengths less than 35 feet, or to RS-422 for multi-drop applications or cable lengths longer than 35 feet.

## Generic Operator Terminal

The Generic check box selects the use of any generic operator terminal for the serial port 1 connection on P7. If this setting is cleared, the IQ Operator Terminal is selected. When a generic terminal is used, the IQ will not automatically send any special characters to the serial port. PRINT and READ statements can be used in a program to send any characters to the terminal. The built in status functions, Xkey functions, and Fkey functions cannot be used with a generic terminal.

The Generic check box (available for Port 1 only) is used to specify a generic operator terminal. No Allen-Bradley Operator Terminal control codes will be automatically sent to the terminal when using PRINT, READ, and CLEAR commands if this box is selected. Control codes can still be sent to your terminal using the PRINT "^code" statement, however. The Generic option is available only in firmware versions 2.10 and later. If your firmware version is earlier than 2.10, the check box will be disabled.

If a generic operator terminal is used, the Op Term Key and the Codes list boxes are used to specify the codes the operator terminal generates. Select the specific key from the Op Term Key list box, then select the code for the key in the Codes list box. The code consists of two bytes--the high byte being the left most Code list box. If your codes are in hexadecimal, you must convert to decimal. For example, if the generic terminal sends '5Fh 0h' for F1 ON and '6Fh 0h' for F1 OFF, select '95 0' for F1 ON and '111 0' for F1 OFF. If the operator terminal sends only single byte codes, set the high byte of each key code to 0.

For keys that are not implemented, the key code should be set to 0. The Generic check box must be selected and the code for F1 ON must be non-zero so that the IQ will load the key codes on power up. The Generic check box does not have to be selected to set the key code values, however. If the operator terminal does not support the F1 ON function, a code that will never be sent must be set for the F1 ON code value. This option is available only in firmware versions 2.11 and later. If your firmware version is earlier than 2.11, the list boxes will be disabled.

## Operator Terminal Address

The Operator Terminal Address box is used to specify a unique address (range is 0 to 9) which is used when the Operator Terminal is used in a multi-drop application. (To use the Operator Terminal in multi-drop mode, the Operator Terminal and the IQ must be wired for RS-422 and Port 1 must be set to RS-422 mode.) Each IQ connected to the Operator Terminal in the multi-drop mode must have a different address assigned. Address 0 is automatically selected on power up. To communicate with a different IQ from the Operator Terminal, press the 'NO' key on the Operator Terminal followed by the desired address (0-9). This option is available only in firmware versions 2.12 and later. If your firmware version is earlier than 2.12, the box will be disabled.



## Variables

The Variables menu may be used to set the values of the nonvolatile variables G1 through G64. If values are changed in the program while this dialog box is open the new value will not be displayed. The value is updated whenever you open the dialog box or change from one G variable to another G variable. To continuously monitor the value of a variable continuously, refer to the Monitor menu, Monitor Variables on page 69.

To set a variable, select Variables from the Parameter menu, then select the variable you want to change from the drop down list box. TAB to the Value box and enter the new value in the Value box. If not using a mouse, press ALT+DOWN ARROW to open the drop down box. Once the box is opened, use the UP and DOWN ARROWS to scroll through the selections. As a variable is highlighted, the value of the variable will be shown in the Value box. To make a selection, press the ALT+UP ARROW or ALT+DOWN ARROW to select the highlighted item.

Choose Cancel to leave the dialog box without saving any changes. After changing a variable value, choose OK to store the new value and close the dialog box. Choose Update to store the new value, but still leave the Variables dialog box displayed

### ATTENTION



Modifying variables while a program is running may result in unexpected machine motion. It is also possible that the program will overwrite a variable after the user modifies it. It is the responsibility of the user to determine the potential hazards involved. Failure to observe these precautions could result in bodily injury.

## Edit Parameter File

### New

Select New from the Edit Parameter File menu item under the Parameter menu to create a new Parameter hex file on the PC. The file will contain default values for the parameters independent of motor size.

Once the file has been opened, you may print or edit the file just like any other text file. However, when the file is saved, it will be converted back to a Parameter hex file--not a text file. Only minimum range checking is performed on the parameters (because there is no personality module data to compare against) so use caution when editing the file. Do NOT edit the parameter labels on the left side of the equal sign – only edit the values on the right side of the equal sign. If any parameters are missing (or IQ Master is unable to recognize a parameter label) when the file is saved, the save will be aborted. In other words, the file must contain a full list of parameters before it will be saved to disk.

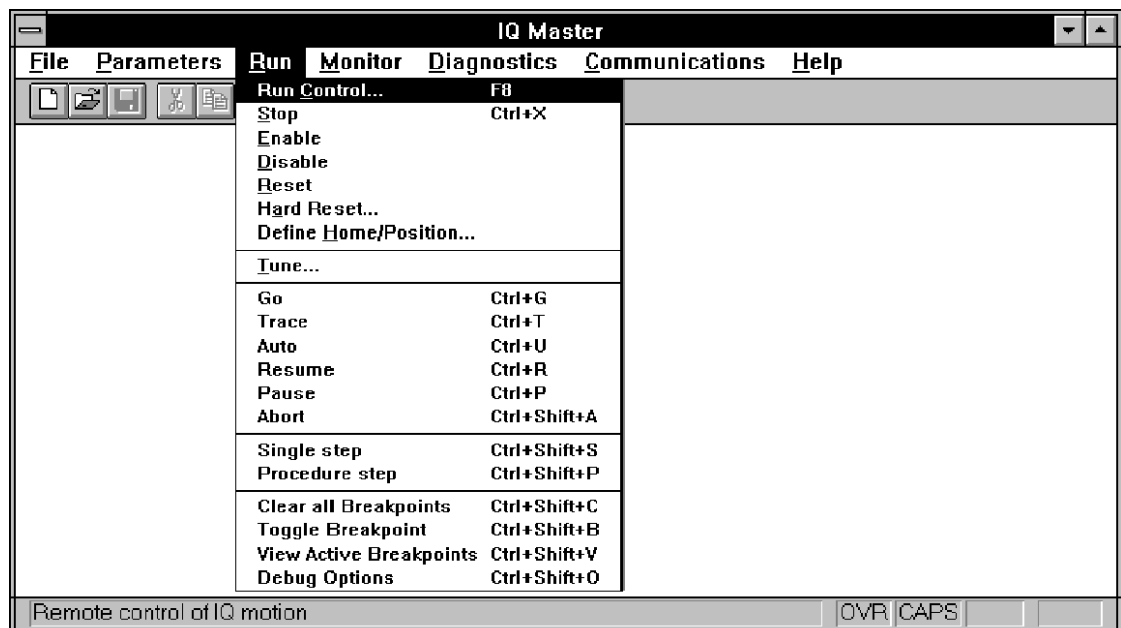
### Open

Select Open from the Edit Parameter File menu item under the Parameter menu to open an existing Parameter hex file on the PC. Select the file to be opened from the Open file dialog box. The file to be opened must be a valid Parameter hex file that was transferred from an ULTRA Plus or IQ or created with the Edit Parameter File, New menu item.

Once the file has been opened, you may print or edit the file just like any other text file. However, when the file is saved, it will be converted back to a Parameter hex file--not a text file. Only minimum range checking is performed on the parameters (because there is no personality module data to compare against) so use caution when editing the file. Do NOT edit the parameter labels on the left side of the equal sign – only edit the values on the right side of the equal sign. If any parameters are missing (or IQ Master is unable to recognize a parameter label) when the file is saved, the save will be aborted. In other words, the file must contain a full list of parameters before it will be saved to disk.



# Run Menu

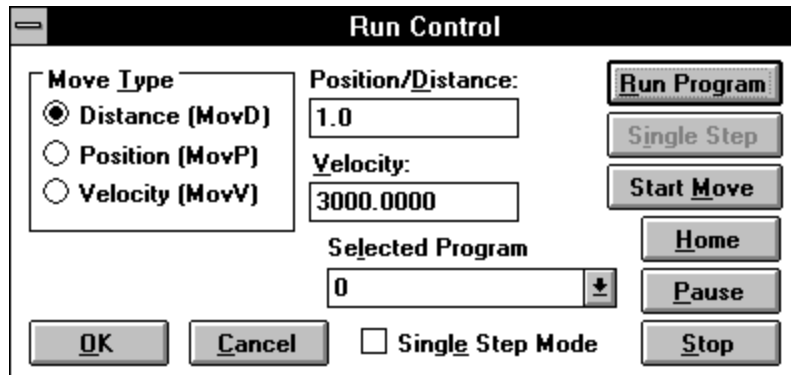


The Run menu provides functions to start, stop, or pause a program, execute a move, start the home sequence, enable/disable the ULTRA Plus or IQ, define the present position as Home or redefine the present position. Along with the motion commands, the Run menu contains the Reset, Hardware Reset, Tune and Extended Debug functions.

### To Access Run Menu Items

- Select the Run menu by clicking on Run in the menu bar,  
-or-  
press ALT to activate the menu bar, then R to pull down the Run menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the “S” in Stop),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

## Run Control



The Run Control dialog box is used to run a program, execute a move, home the machine, or pause or stop a program or move.

### Run Program

- Select Run Control from the Run menu or press F8.
- Select the program number you want to run from the Selected Program drop down list box.
- Choose the Run Program button to start running the program.
- The program can be stopped by choosing the Stop button.
- To pause the program, choose the Pause button. Once a program has been paused, choose the Pause button again to resume the program.
- Choose the OK button to close the Run Control dialog box and to remember all the settings (program number, move type, move velocity and distance) the next time the box is shown.  
-or-  
Choose the Cancel button to close the box and restore the previous settings.

### Home

There are several ways to start the home sequence: use the Home Command input (I6), choose the Home button in the Run Control dialog box, use an Fkey on the Operator Terminal, or issue a Host Mode Language Home command from the Terminal window.

To start the Home sequence from the Run menu

- Select Run Control from the Run menu.
- Choose the Home button.

## Pause

The Pause interrupts motion temporarily. If Pause is activated, the controller decelerates the system to a stop, and maintains position. If Pause is then deactivated, motion is resumed. The time that the system decelerates to zero and accelerates back to full speed is called the system slew time. The system slew time, in seconds is the default velocity, in units/second divided by the default acceleration in units/second<sup>2</sup>. When motion resumes, the position target is the same as before the interruption. The Pause parameter allows the Pause input to be enabled and the active state defined as open or closed. There are two ways to pause motion: choose the Pause button in the Run Control dialog box or use the Pause input (I9).

To Pause motion from the Run menu:

- Select Run Control from the Run menu.
- Choose the Pause button.
- Choose the Pause button again to resume motion.

**Note:** Pause will NOT stop motion if a program is following a master encoder using a gear ratio.

## Single Step

To single step through a program, select the Single Step Mode check box, then select the Run Program button to start the program (if the program is already running, just select the Single Step button). The program can be stepped through one instruction at a time by continuing to choose the Single Step button. The program will stop after executing each instruction. The next instruction to be executed is shown in the Status dialog box.

To run a program normally after single stepping, clear the Single Step Mode check box.

**Note:** The program must have been compiled with Generate Debug Information selected (under Compile Options) to allow single stepping.

## Move

It is not necessary to run a program to make a move. As long as the ULTRA Plus or IQ is enabled and no program is running, a move can be made at any time. The only exception to this is, if the Move button was used to start a MOVV move, another MOVV command can be issued (for example, to change the velocity of the move).

To make a move (not in a program)

- Select Run Control from the Run menu or press F8.
- Check the type of move you want—MOVD (move a distance), MOVP (move to a position), or MOVV (move at a velocity).
- For MOVD and MOVP moves, the acceleration used is the Acceleration parameter. For MOVV moves, velocity changes are made using the Jog Acceleration and Jog Deceleration.
- Enter the distance and/or velocity. You must enter the distance for a MOVD or MOVP move, however, specifying the velocity is optional for either type move. If no velocity is entered, the move will be made at the default velocity (specified under Velocity/Accel in the Parameter menu). If you select a MOVV move, you must enter the velocity.
- Choose the Start Move button to start the move.
- Once the move is started, choose the Stop button to stop the move or choose the Pause button to pause the move. Once a move has been paused, choose the Pause button again to resume the move.
- Choose the OK button to close the Run Control dialog box and to remember all the settings (program number, move type, etc.) the next time the box is shown.  
-or-  
Choose the Cancel button to close the box and restore the previous settings.

---

## Stop

Select Stop from the Run menu to stop a program. A program can also be stopped by pressing CTRL+X (press and hold down the CTRL key and press the X key) when the edit window is the active window or by choosing the Stop button in the Run Control dialog box.

---

**ATTENTION**

It is the responsibility of the user to ensure that the application process stops in a safe manner when the application program stops. Failure to observe this precaution could result in bodily injury.

---

**ATTENTION**

A Stop command will not stop jog input motion if the job inputs remain active.

---

## Enable

Select Enable from the Run menu to enable the ULTRA Plus or IQ. If the ULTRA Plus or IQ is disabled due to the Enable/Disable input, this command will have no effect.

---

## Disable

Select Disable from the Run menu to disable the ULTRA Plus or IQ. If the ULTRA Plus or IQ has been disabled by the Enable/Disable input, this command will have no effect. If this command is used to disable the drive the Enable/Disable input will have no effect until the drive is Enabled from the Run menu.

---

## Reset

Select Reset from the Run menu to reset the ULTRA Plus or IQ if any “soft” errors have occurred. Use Hardware Reset to clear any hard errors. Reset will not clear the Home Sequence Complete output or the HSEQCPL internal flag.

---

## Hardware Reset

Select Hardware Reset from the Run menu to clear any “hard” errors that have occurred. All other errors are “soft” errors that can be cleared with Reset. Refer to “Error Messages” on page 418 for a list of “hard” errors.

Using Hardware Reset is just like removing and restoring power to the ULTRA Plus or IQ. Hardware Reset will set all position variables (Pos1, Pos2, etc.) to 0 and clear the Home Sequence Complete output and the HSEQCPL internal flag.

---

## Define Home Position

Select Define Home/Position from the Run menu to define the present commanded position as Home (position zero) or to redefine the present position. To define the present position as Home, choose the OK button. To redefine the present position, select the Define Position button, enter the new position, then choose the OK button. The ULTRA Plus or IQ must be enabled and not running a program or executing a move to define a position.

---

## Tune

Two types of tune modes are available: Auto Tune and Manual Tune.

Auto Tune mode provides a method for tuning the servo amplifier connected to a machine without any special equipment other than the serial terminal. It allows a reasonable set of tuning parameters to be developed quickly for a particular machine. In Manual Tune mode, the tuning parameters are manually adjusted. For many applications, running Auto Tune will generate tuning parameters that will be adequate for machine performance. However, if very high performance is needed, or special conditions exist (such as changing loads or large inertia mismatches), Manual Tune can be used to “fine tune” the tuning parameters AFTER running Auto Tune.

Auto Tune implements a “self-tuning” algorithm that adjusts the tuning parameters by computing the total inertia consisting of the motor inertia plus the load inertia at the motor shaft. Auto Tune operates by commanding a constant current to the motor, producing a constant motor torque. The acceleration of the motor is measured and used to compute the inertia of the system. Once the actual inertia of the system is measured, the values of the tuning parameters are automatically adjusted to achieve the desired performance.

**Note:** For IQ-550 controllers only, Auto Tune will not operate unless the motor torque constant and the motor inertia have been entered into the Personality Module using the Initialize Personality Module dialog. Refer to the start up procedure in the *IQ-550 Installation Manual* (Part Number 0013-1022-004).

---

### ATTENTION



Auto tune and manual tune will produce rapid back and forth motion of the motor. Ensure the mechanical system will not be damaged by this motion.

---

**Note:** Auto Tune is not recommended if too much compliance or backlash exists in the mechanical load. If this is the case, then manual tune is recommended.

Manual Tune provides a means of tuning the velocity and position control loops independently. The inner velocity loop must be tuned first, as this tuning will affect the position loop response. This procedure assumes that the system, including the machine the motor is connected to, can tolerate small-signal step velocity changes.

## Auto Tune Velocity and Position Loops

### To Tune the System using Auto Tune

1. Disable the ULTRA Plus or IQ.
2. Select Tune from the Run menu.
3. Choose the Auto Tune button (if in the Manual Tune mode).
4. Select the Application Type. The Application Type and Response settings are used to calculate the system tuning gains after the system inertia estimation is complete. The Application Type can be either Point to Point or Contouring. Auto Tune will set the velocity loop IGAIN to zero for Point to Point. For Contouring, IGAIN is set to a nonzero value that results in about a 15% velocity step overshoot. Auto Tune calculates values for KP, PGAIN, and for IGAIN if Application is set to Contouring. The other tuning parameters are set to: KI = 0, IZONE = 0, KFF = 100%, FGAIN = 0%, KPZ = 0, PZONE = 0, and FILTER is turned off.

**Note:** KFF at 100% will induce overshoot in some point to point applications. KFF at 100% is recommended in gearing applications.

5. Set the Direction radio button to Bidirectional, Positive, or Negative. The Direction setting determines the direction of motion during the Auto Tune process. When the direction is Positive or Negative, Max Distance is not the maximum distance moved from the starting point, but rather the maximum distance moved during each acceleration and deceleration cycle of the Auto Tune process. The positive direction is defined as clockwise rotation of the motor shaft when facing the motor shaft end.
6. Choose an appropriate Response radio button (High, Medium, or Low). The Response selection determines the target bandwidth used to calculate the system gains. The velocity and position loop bandwidths are:

Response	High	Medium	Low
Velocity Loop BW	30 Hz	15 Hz	7 Hz
Position Loop BW	10 Hz	5 Hz	2 Hz

Choose Medium if you are not sure which Response will suit your application.

7. Adjust Step Current, Max Distance, and Max Velocity if required. Units are user units for Max Distance and user units per Timebase for Max Velocity. The default values should provide good results for most systems. Reduce Step Current if the torque will exceed machine specifications.
8. Choose the Enable button to enable the ULTRA Plus or IQ, then choose the Start button to begin the Auto Tune process. The motor shaft will oscillate back and forth. The Enable and Start buttons are toggle buttons—when chosen, the meaning of each button changes—the Enable button will change to Disable and the Start button will change to Stop. Choosing the button again will toggle it back to its original meaning.
9. After about 5 seconds, the motion will cease and the ULTRA Plus or IQ will display the calculated load inertia to motor inertia ratio, and the new calculated gains.

Under certain conditions, an incorrect value can be computed for the system inertia that will result in



high gain settings. This could cause instability when returned to the closed loop mode of operation. The conditions are:

Max Distance or Max Velocity set too low. Either of these conditions can cause very rapid reversal of the servo motor and audible vibration. Increase Max Distance or Max Velocity.

Step current set too low. This is shown by the lack of servo motor motion. Increase Step Current.

If the gains are set to the maximum value, repeat the Auto Tune process with higher values for Step Current, Max Distance, and Max Velocity.

After the Auto Tune process is complete, if further fine tuning of the gains is required, choose the Manual Tune button to run Manual Tune. If you are satisfied with the new gain settings, choose the OK button to save the settings and exit. If not, choose the Reset Gains button to restore the gains to the previous settings. The previous settings can be restored only if you have NOT chosen the OK button. The Cancel button is similar to the Reset Gains button except that it will exit Auto Tune.

## Manual Tune

The Manual Tune mode provides a means of tuning the velocity and position control loops independently. The inner velocity loop must be tuned first, as this tuning will affect the position loop response. This procedure assumes that the system, including the machine the motor is connected to, can tolerate small-signal step velocity changes.

The tuning of the velocity regulator is accomplished by changing the parameters PGAIN, IGAIN, and FILTER. PGAIN is the proportional gain of the velocity regulator. Increasing PGAIN reduces the time required to reach the commanded velocity. IGAIN is the integral gain of the velocity regulator. Integration in the velocity regulator forces the motor velocity to precisely follow the commanded velocity with no error under steady state conditions (no changes in velocity command or load). Increasing IGAIN increases the stiffness, or the ability to reject load disturbances. Increasing IGAIN also, however, increases the amount of velocity overshoot when responding to a step change in velocity. Too much IGAIN can cause the system to go unstable. To reduce stress on the mechanical parts of the machine, PGAIN and IGAIN should be set as low as possible while still maintaining the desired performance.

### *To Tune the Velocity Loop using Manual Tune*

1. Follow the Auto Tune procedure to provide a nearly optimal starting point for tuning.
2. Disable the ULTRA Plus or IQ.
3. From the Monitor menu, set the Programmable Monitor output to VEL1, the default setting.
4. Connect an oscilloscope to the Programmable Monitor output (P3-7) and to analog common (P3-4).
5. Select Tune from the Run menu, then choose the Manual Tune button if in Auto Tune.
6. Select Velocity Step as the Command Type. The Command Type can be either Velocity Step to tune the velocity loop or Position Step to tune the position loop. The velocity loop should ALWAYS be tuned first before tuning the position loop as the velocity tuning affects the position loop response.
7. Set the Direction radio button to Bidirectional, Positive, or Negative. The Direction setting determines the direction of motion during the Manual Tune process. The positive direction is defined as clockwise rotation of the motor shaft when facing the motor shaft end.
8. Set the Step Velocity to the desired step velocity of the internal square wave generator. (A good value is between 100 and 500 RPM.) An excessive Step Velocity or a low ILIMIT setting can cause the servo amplifier to enter current limit which should be avoided while tuning the ULTRA Plus or IQ. If the ULTRA Plus or IQ reaches current limit, "In Peak Current" is displayed.
9. Set the Cycle Period value. Do not set the Cycle Period to a high value (one that would allow the system to reach the end of travel) or to a value less than 0.02 seconds. The Cycle Period is the time in seconds to complete one cycle of the command. If the Direction is set to Bidirectional, one cycle is a move forward and back. If the Direction is either Positive or Negative, one cycle is the move in one direction only.
10. Set the ILIMIT value. ILIMIT is the current limit used during Manual Tune. It is set equal to the ILIMIT parameter when Manual Tune mode is first entered. ILIMIT can be changed here without changing the ILIMIT parameter.

11. Turn off the FILTER by clearing the FILTER check box.
12. Choose the Enable button to enable the ULTRA Plus or IQ, then choose the Start button to begin the Manual Tune process. The motor shaft will oscillate back and forth. If the Step Velocity and/or Cycle Period need to be changed, choose the Stop button, adjust the values as necessary, and choose the Start button again.
13. Set IGAIN to a low value (no noticeable overshoot). Set the Programmable Monitor output to Motor Velocity (VEL1). While watching the VEL1 signal on the Monitor output with the oscilloscope, increase PGAIN until the desired rise time is reached.
14. Increase IGAIN until the acceptable limit for the amount of overshoot is reached.
15. Turn on the FILTER by selecting the FILTER check box, then reduce the FILTER value until the overshoot begins to increase.

*Some General Rules for Tuning:*

To increase bandwidth, increase PGAIN. To increase stiffness, increase PGAIN or IGAIN. To reduce overshoot, increase PGAIN or reduce IGAIN. To reduce rise time, increase PGAIN or IGAIN. To reduce resonance, reduce the FILTER value. If the motor rattles, reduce PGAIN, IGAIN, or the FILTER value.

Once the velocity loop has been tuned, the position loop may be tuned.

The parameters used to tune the position loop are KP, KI, IZONE, KFF, PZONE, and KPZ. KP is the position loop proportional gain. Changing KP changes the position loop bandwidth. KI is the position loop integral gain. The use of an integral gain in the position loop reduces the effects of friction and allows zero error when holding position. KI is used in conjunction with the IZONE parameter. IZONE is the area around the commanded position where the KI gain is active. KFF is the velocity feedforward gain used in the position loop to minimize the following error when the system is moving. The PZONE parameter sets a zone around the commanded position where the position loop proportional gain is changed to the gain set by the KPZ parameter. Tuning using the step response allows adjustment of KP, KI, IZONE, PZONE, and KPZ.

***To Tune the Position Loop using Manual Tune***

1. Select Gains/Limits from the Parameter menu and set the Following Error Limit Time and Size parameters to the maximum allowed to avoid excess following error faults while tuning.
2. Select Tune from the Run menu, then choose the Manual Tune button.
3. Select Position Step as the Command Type.
4. Set the Direction radio button to Bidirectional, Positive, or Negative. The Direction setting determines the direction of motion during the Manual Tune process. The positive direction is defined as clockwise rotation of the motor shaft when facing the motor shaft end.
5. Set the Step Position to the desired position change.
6. Set the Cycle Period value. Do not set the Cycle Period to a high value (one that would allow the system to reach the end of travel) or to a value less than 0.02 seconds. The Cycle Period is the time in seconds to complete one cycle of the command. If the Direction is set to Bidirectional, one cycle is a move forward and back. If the Direction is either Positive or Negative, one cycle is the move in one direction only.
7. Set the ILIMIT value. ILIMIT is the current limit used during Manual Tune. It is set equal to the ILIMIT parameter when Manual Tune mode is first entered. ILIMIT can be changed here without changing the ILIMIT parameter.
8. Set the KFF and KI parameters to 0.
9. Choose the Enable button to enable the ULTRA Plus or IQ, then choose the Start button to begin the Manual Tune process. The motor will move back and forth. If the motor response is sluggish, slightly increase the value of KP. If the motor response is too stiff, enter a slightly lower KP value.
10. While watching the VEL1 signal on the Monitor output with the oscilloscope, adjust KP for the quickest response with minimum overshoot. It may be helpful to connect an LED and 2.2k ohm resistor in series to the In-Position output to check the In-Position output during the tuning process.

11. Adjust KI. Enter a value for IZONE (about twice the In-Position Window is a reasonable value). Begin slowly increasing the KI value from 0 while watching the VEL1 signal on the oscilloscope. As KI is increased, the system will begin to overshoot. KI should be adjusted to achieve the fastest possible time to come into position with minimum overshoot.
12. When you are satisfied that the gains have been set to give the correct system response, select the OK button to save the settings and exit. Select Gains/Limits from the Parameter menu and set the Following Error Limit Time and Size parameters back down to reasonable values.
13. To restore the gains to the previous values, choose the Reset Gains button. To restore the gains to the previous values AND exit the Tune mode, choose the Cancel button.

## Extended Debug

Extended debug capabilities allow program execution to be verified. Debugger commands can be used by all program types, except for Fkey programs and the error program. Debugger commands are grouped into three categories:

- Operational
- Step
- Breakpoints and Options

**Note:** Debug information should not be included in the final executable that is transferred to the ULTRA Plus or IQ. Programs containing debug information experience a 5 byte per source line increase in size of the executable file and consequently there is a time penalty involved with executing extended debug images of a program.

Before debugging can occur, perform the following steps:

1. Create a new file or open an existing file with IQ Master.
2. Compile the source file with extended debug information.

Compiling a source program for debugging requires that the following settings be active in the Compiler Options of the Edit menu:

- Compile to Disk
- Generate List File
- Disassemble Source
- Add Debug Information
- Extended Debug Information

Extended debug information also may be generated during the compile by specifying the following preprocessor directives in the source file immediately after the program title statement:

- CompileToMemory = OFF
- ListFile = ON
- Disassemble = ON
- Debug = ON
- ExtendedDebug = ON

Refer to the on-line Help menu for explanation of these preprocessor directives.

3. Save the compiled executable to the IQ by selecting one of the following:
  - Save Program to IQ button in the Compiler Results dialog
  - Save Program to IQ command in the File menu.
4. Enable the IQ and issue the desired extended debug commands from the Run menu.

### Operational Commands for Debugger

#### *Go*

Go initiates a program in a non-traceable mode at full speed. The execution trace bar is *not* displayed in the source code window of IQ Master while the programs executes.

To halt program execution, issue a Pause command from the Run menu.

The commands Auto, Trace or Abort may be issued anytime the program is running. Breakpoints may be added, deleted or disabled while the program is executing.

### ***Trace***

Trace initiates a program in a traceable mode at full speed. The execution trace bar is displayed in the source code window of IQ Master while the programs executes.

To halt program execution, issue a Pause command from the Run menu.

The commands Go, Auto or Abort may be issued anytime the program is running. Breakpoints may be added, deleted or disabled while the program is executing.

### ***Auto***

Auto initiates a program in a traceable mode at a reduced speed. The time delay between execution of source lines is programmable using the Debug Options dialog. The execution trace bar is displayed in the source code window of IQ Master while the programs executes.

To halt program execution, issue a Pause command from the Run menu.

The commands Go, Trace or Abort may be issued anytime the program is running. Breakpoints may be added, deleted or disabled while the program is executing.

### ***Resume***

Resume initiates the last Go, Trace or Auto command issued. It is typically used after a Pause command.

### ***Pause***

Pause temporarily halts execution of a program. The execution trace bar is displayed in the source code window of IQ Master while the program is paused.

Resume is typically used to continue execution of the program, as it reinitiates the Go, Trace or Auto command used to begin program execution.

Breakpoints may be added, deleted or disabled while the program is in the Pause mode.

### ***Abort***

Abort terminates program execution. The execution trace bar is removed if it was displayed while the program was executing.

## **Step Commands for Debugger**

### ***Single Step***

Single step allows the source code to execute one line at a time. If multiple expressions appear on a single source code line, all expressions are executed. Single step will step into subroutine calls.

Breakpoints may be added, deleted or disabled while the program single steps.

### ***Procedure Step***

Procedure Step executes subroutine calls as a complete instruction. If no breakpoint is encountered with the subroutine, the entire subroutine is executed. The execution trace bar is displayed at the next executable source line after the subroutine call. If a breakpoint is set in the subroutine, Procedure Step pauses at the breakpoint.

If a subroutine call is *not* present in the source statement, Procedure Step performs the same as Single Step.

Breakpoints may be added, deleted or disabled while the program procedure steps.

## **Breakpoint Commands and Options for Debugger**

Breakpoints can be set in the main program, scanned events, and Xkey routines. Breakpoints can be manipulated at any time, but typically are modified after program execution has been paused and before restarting program execution. Breakpoints are recognized when the program is executing in the Go, Trace, or Auto modes.

### Clear all Breakpoints

Clear All Breakpoints removes all defined break points in the source file.

The View Active Breakpoint dialog allows defined break points to be enabled or disabled as a group or individually.

### Toggle Breakpoint

Toggle Breakpoint creates or deletes a breakpoint within a line of source code. This allows a breakpoint bar to be toggled (added or deleted).

To add a breakpoint:

1. Position the cursor for a text editor on the source code line where the breakpoint should occur.
2. Issue the Toggle Breakpoint command. The breakpoint bar will appear where the cursor is displayed in the editor window.

To delete a breakpoint:

1. Position the cursor for a text editor on the source code line where the breakpoint resides.
2. Issue the Toggle Breakpoint command. The breakpoint bar will be removed from the editor window where the cursor is displayed.

The View Active Breakpoint dialog allows defined breakpoints to be enabled or disabled, without deleting individual breakpoints.

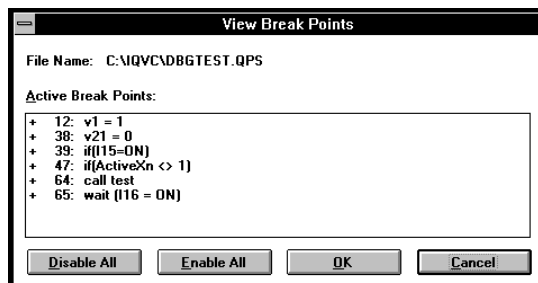
### View Active Breakpoints

View Active Breakpoints, shown below, is a dialog that displays all currently defined breakpoints. The dialog lists the source code line number and text in which the breakpoint is located. A “+” or “-” character is displayed to the left of the source code. Breakpoint bars for enabled and disabled breakpoints are displayed in different colors.

- + indicates a breakpoint is enabled or active.
- - indicates a breakpoint is disabled or inactive.

The View Active Breakpoint dialog allows defined breakpoints to be enabled or disabled individually or altogether:

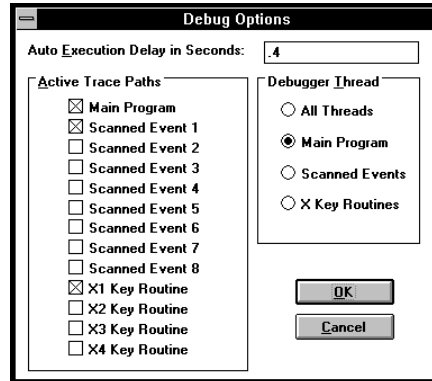
- The Enable All button enables all breakpoints in the program.
- The Disable All button disables all breakpoints in the program.
- Double-clicking the source code line in which a breakpoint occurs causes the breakpoint to be toggled and the color of the breakpoint bar to change. For example, double-clicking a source code line displaying a “-” (disabled) causes the sign to change to “+” (enabled), and the color of the breakpoint bar to change.



### Debug Options

Debug Options provides a series of check boxes and buttons for specifying portions of code on which to

run selected debug tools and methods.



### Auto Execution Delay

Sets the time delay in seconds to be inserted between the execution of source code lines. Typical values for this parameter are 1 to 3 seconds typically. If set to 0 (zero), the auto mode executes at the same speed as the Trace debugger mode. The range of values for this parameter is 0 to 65.5 seconds.

### Active Trace Paths

Active Trace Path check boxes filter specific debug information sent by the IQ to IQ Master when debugging a source program. Execution modes that display an execution trace bar in the source code window (e.g., Trace, Auto, and Pause), require the active trace path to debug a specific code thread. For example, if an application has a main program loop, scanned event #1 and XKey #2, check the Main Program, Scanned Event 1, and X2 Key Routine check boxes. If the application executes code that does not have its associated active trace path defined, IQ Master debugger software will not respond to the executing code and the debug information being sent by the IQ. All application code executes on the IQ even when no active trace path boxes are checked.

Enabling active trace paths for specific thread execution is used primarily when executing in the Auto or Trace debugger modes. In these modes, you have the ability to visually see trace execution for specific code areas of the application.

When in doubt as to the proper settings for the active trace path check boxes, simply check them all.

### Debugger Thread

Debugger Thread radio buttons target specific execution threads for debugging commands. The IQ executes its application programs in a multi threaded programming environment. Under certain situations, it may be beneficial for you to debug the main program only. In another application, you may want to set breakpoints in scanned event code to debug them. The Debugger Thread radio buttons independently specify which execution thread(s) to command.

All Threads should be selected if you are new to IQ multithreaded programming. This setting allows you gain experience with how the main program, scanned events, and XKey routines execute in relation to one another.

Special attention is required when *not* using the “All Threads” command target, pay particular attention to setting break points in code areas that are not being sent debugger commands. For example: If a breakpoint is set in scanned event #1 and the main program is set as the target for debugger commands, the scanned event could hit the defined breakpoint while debugging the main program thread. In this case, *all* scanned event execution stops until commanded to start execution again.

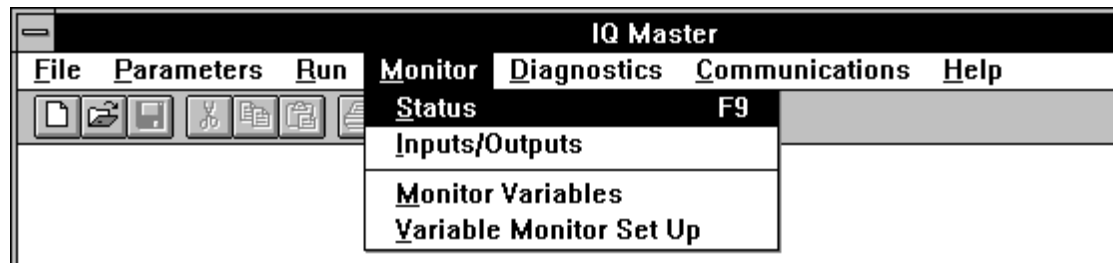
Points to keep in mind when debugging in the IQ's multi threaded environment include:

- Users unexperienced with IQ multi threaded programming should select the “All Threads” radio button.
- When initially starting a program all execution threads receive the first debugger command issued, regardless of the debugger command thread selected.

- When a scanned event hits a breakpoint, all scanned events are halted.
- When an XKey routine hits a breakpoint, all XKey routines are halted.



# Monitor Menu



The Monitor menu is used to show the ULTRA Plus or IQ Status, Input/Output Status, select variables to be monitored, assign a variable to the Programmable Monitor output (P3-7), and call the Variable Monitor display. The Variable Monitor display can monitor up to 8 internal pre-defined variables.

## To Access Monitor Menu Items

- Select the Monitor menu by clicking on Monitor in the menu bar,  
-or-  
press ALT to activate the menu bar, then M to pull down the Monitor menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the “S” in Status),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

---

## Status

To get the status of the ULTRA Plus or IQ, select Status from the Monitor menu or press F9. The Status dialog box shows certain inputs and outputs that are currently active, and machine faults (if any). If a program is running, program information is also shown. If the program was compiled with debug information, the Program Status box shows the program number (and title if there is one), the current line number, and the statement that is currently executing (refer to the Edit menu, Compiler Options for details).

The outputs shown are (refer to Parameters, Outputs for detailed information about each of the outputs):

- Program Running
- At Home
- Home Sequence Complete
- In-Position
- Error
- Drive Ready
- Drive Enabled

The inputs shown are (refer to Parameters, Inputs for detailed information about each of the inputs):

- Forward Limit
- Reverse Limit
- Pause

---

## Inputs/Outputs

The I/O Status Monitor is used to continuously monitor the status of the digital inputs and outputs at any time—even if a program is running.

A check box is used to show the status of each input/output. If the box is selected, the corresponding input/output is active.

If there is an option card present with additional inputs and outputs, the Next Bank button will be enabled. Choose the Next Bank button to monitor the additional inputs and outputs. If an option card with additional inputs and outputs is not present, the Next Bank button will be disabled.

## Monitor Variables

The Monitor Variables menu item displays the Variable Monitor dialog box which continuously displays the values of system variables. Up to eight different variables can be monitored at once. The variables to be monitored are set up in the Variable Monitor Set Up dialog box.

The values assigned to be monitored are saved in the MASTER.INI file so they will remain assigned the next time IQ Master is run.

The Variable Monitor Set Up dialog box is also used to assign a variable to be output on the Programmable Monitor output. The Programmable Monitor output is an 8 bit Digital to Analog Converter (DAC) that can be used to monitor a variable. The range of the output is  $\pm 10$  volts which provides a resolution of about 78 mV. If the variable selected is not present, the output will be -10 volts.

Position variables are output as an analog voltage corresponding to one user unit (defined by the Scale parameter) per 20 volt range of the monitor output. Velocity variables are scaled by the Velocity Scale parameter described below. Current variables are scaled by the drive peak current rating (10 volts equals peak current).

*To assign variables to the Variable Monitor dialog box:*

- Select Variable Monitor Set Up from the Monitor menu.
- To add a variable to the Monitor List, choose the type of variable from the Variable Type list. System variables, G variables, and V variables may be monitored. Double-click on the specific variable from the Monitor Selection box, or select the specific variable from the Monitor Selection box and choose the Add button. The Monitor List can contain up to 8 variables.
- To delete a variable from the Monitor List, double-click on the specific variable from the Monitor List, or select the variable to be deleted from the Monitor List, and choose the Delete button.
- To assign a variable to the Programmable Monitor output, select the variable from the list and choose the Set Monitor Output button. The variable that is currently assigned to the Programmable Monitor output is displayed in the Monitor Output box.
- Choose OK when the selection is complete or Cancel to return to the previous settings. If the Variable Monitor dialog box is not already active, press the Start Monitor button to display the Variable Monitor dialog box.

The variables listed below are available to be monitored. The name in parentheses after the variable is the descriptive name of the variable. The table lists the variables alphabetically, refer to Part 5 • Language Reference, OTMON for a list of variables in the order they appear in the dialog box. In addition to the variables listed below, there are other variables associated with IQ Cam. Refer to the *IQ-Series IQ-Cam Software Manual* (Part Number 0013-1053-001) for complete details.

Name	Description
ADC1	Value of the Analog to Digital Converter (ADC1) (P3-5) input.
ADC2-ADC5	Value(s) of the analog to digital converter(s) on the optional I/O Expansion Card. If an optional card is not present, the voltage is shown as -10 volts.
FE (Following Error)	Following error (difference between Position Command and Feedback Position) shown in user units.
FVEL1 (Filtered Velocity) <sup>1</sup>	Motor velocity after being averaged by the Velocity Monitor filter.
GPOS (Gear Position)	External position command from the Gear function, before the Slew function.

IAVE (Average Current)	Average current command.
ICMD (Current Command)	Instantaneous current command.
IX1P1 (Index 1 Pos1)	Position 1 (Pos1) when the index from Encoder 1 is received.
IX1P2 (Index 1 Pos2)	Position 2 (Pos2) when the index from Encoder 1 is received.
IX2P1 (Index 2 Pos1)	Position 1 (Pos1) when the index from Encoder 2 is received.
IX2P2 (Index 2 Pos2)	Position 2 (Pos2) when the index from Encoder 2 is received.
I1P1 (Int1 Pos1)	POS1 value latched in software when an interrupt 1 (P1-12) is received.
I1P2 (Int1 Pos2)	POS2 value latched in software when an interrupt 1 (P1-12) is received.
I2P1 (Int2 Pos1)	POS1 value latched in software when an interrupt 2 (P1-13) is received.
I2P2 (Int2 Pos2)	POS2 value latched in software when an interrupt 2 (P1-13) is received.
LPOS (Latched Position)	Value of Pos1 or Pos2 (depending on Feedback Configuration) that is latched in hardware when an interrupt 1 (P1-12) is received.
PCAM (Cam Command)	Position command from the Cam function.
PCMD (Position Command)	Position command as defined by the Scale parameter. The Monitor output will roll over from +10 to -10 volts as the command advances beyond full scale on the output. Sum of Jog command, profile command, and External command.
PEXT (External Command)	External position command from the Gear input, but after the Slew calculation is performed on it.
PFE (Peak Following Error)	Maximum following error in user units. When the Monitor display is running, choose the Clear Peak button to clear and reset the peak following error value.
PGEN (Profile Command)	Position command from internal profile generator.
PICMD (Peak Current)	Maximum current command in amperes. When the Monitor display is running, choose the Clear Peak button to clear and reset the peak current.
PJOG (Jog Command)	Jog command from Jog input(s).
POS1 (Position 1)	Current Pos1 value.
POS2 (Position 2)	Current Pos2 value.
POS3 (Position 3)	Current Pos3 value from an optional Encoder Card. If an optional card is not present, the voltage is shown as -10 volts.
POSN (Feedback Position)	Actual feedback position as defined by the Scale parameter. The Monitor output will roll over from +10 to -10 volts as position advances beyond full scale on the output.
PVEL1 (Peak Motor velocity)	Maximum velocity in user units per Timebase. When the Monitor display is running, choose the Clear Peak button to clear and reset the peak motor velocity value.

RFDR (Feedrate)	Current run time feedrate value in percent. When assigned to the Monitor output, +10 volts equals 200% while -10 volts corresponds to 0%.
VCMD (Velocity Command) <sup>2</sup>	Velocity command as defined by the Velocity Scale parameter. (For example, 10 volts is 3,000 RPM if Velocity Scale = 300.)
VEL1 (Motor Velocity) <sup>2</sup>	Actual motor velocity. When assigned to the Monitor output, 10 volts is the maximum motor speed as set by the Velocity Scale parameter. (For example, 10 volts is 3,000 RPM if Velocity Scale = 300.)
VEL2 (Velocity2) <sup>2</sup>	Velocity from Encoder 2 in user units per Timebase.

1. Velocity monitor filter is used to establish the filtering value on FVEL1, the filtered motor velocity.
2. Velocity Scale is used to establish a scale (velocity units/volt) for the Programmable Monitor output.

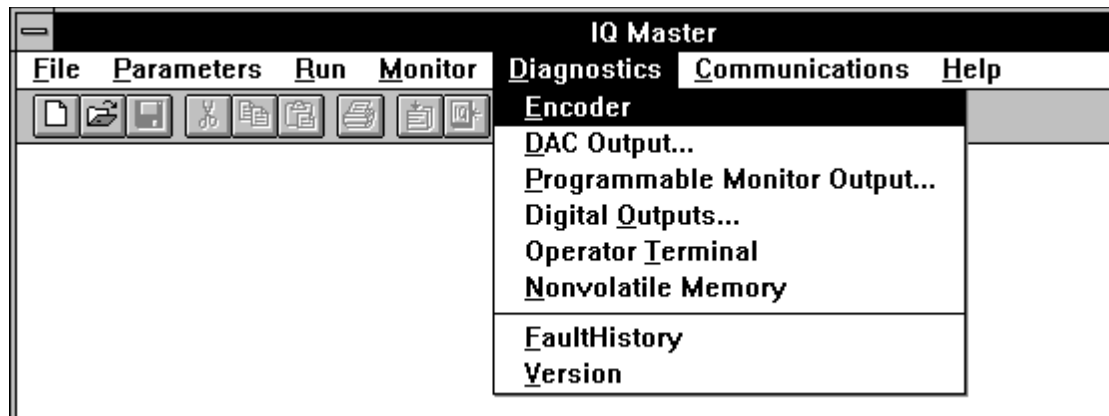
#### *Velocity Scale*

Scale the Programmable Monitor output when set to a velocity variable. The value is used to scale the output in velocity units per volt. Normally, you would enter 1/10 of the maximum speed of your motor. The maximum speed will then result in a monitor voltage of 10 volts. For example, if the Scale parameter is set to motor revolutions, the Timebase is minutes, and the maximum speed of the system is 3000 RPM, enter a value of 300. This will cause the Programmable Monitor output to be 0 volts at 0 RPM, 1 volt at 300 RPM, 10 volts at 3000 RPM and -10 volts at -3000 RPM.

#### *Velocity Monitor Filter*

Velocity Monitor filter establishes the amount of filtering of the FVEL1 variable. This value has a range of 0 to 32,767. A value of 32,767 is no filtering (quickest change of the value of FVEL1) and a value of 0 is maximum filtering (slow change).

# Diagnostics Menu



The Diagnostics menu contains several submenus for various diagnostic tests. The Diagnostics mode is intended for troubleshooting while the ULTRA Plus or IQ is disabled and no program is running. The Encoder test checks both Encoder 1 and Encoder 2. The DAC1 Output test exercises the Digital to Analog Converter (DAC1) output. The Programmable Monitor Output test exercises the Programmable Monitor Output. The Digital Outputs test allows digital outputs to be forced ON or OFF to test the outputs and system wiring. The Operator Terminal test verifies communication between the ULTRA Plus or IQ and the Operator Terminal. The Nonvolatile Memory Test does a nondestructive test of the ULTRA Plus or IQ nonvolatile RAM (NVRAM). Fault History shows the last 46 faults and the running time when the fault occurred. Version shows the current version of the firmware in the ULTRA Plus or IQ.

**Note:** To check the Analog to Digital Converter (ADC), select Monitor Variable from the Monitor menu. To check the Digital Inputs, select Inputs/Outputs from the Monitor menu

### To Access Diagnostics Menu Items

- Disable the ULTRA Plus or IQ. If assigned, turn OFF the Enable input.  
-or-  
Select Disable from the Run menu.
- Select the Diagnostics menu by clicking on Diagnostics in the menu bar,  
-or-  
by pressing ALT to activate the menu bar, then D to pull down the Diagnostics menu.
- Select the menu item by clicking on your selection,  
-or-  
by pressing the letter that is underlined in your selection (for example, the ‘V’ in Version),  
-or-  
by using the arrow keys to highlight your selection and then pressing ENTER.

---

## Encoder

The Encoder test checks both Encoder 1 and Encoder 2.

#### *To Test and/or Check the Size of an Encoder:*

- From the Diagnostics menu, choose Encoder.
- To verify that encoder counts are being received, rotate the encoder. As the counts are received, the count display will increment or decrement depending on the direction the encoder is rotated. The count display can be cleared to zero with the Zero Counts button.
- To check the size of the encoder, rotate the encoder slowly until “Index” flashes once, then rotate it until “Index” flashes again. The size should now be displayed. The size displayed is in encoder counts (number of encoder lines multiplied by 4).
- To check the position of the encoder index relative to the home switch, move to the home switch so “Home Switch Active” is shown, then rotate the encoder until “Index” is shown.

---

## DAC1 Output

The DAC1 Output is a 12 bit Digital to Analog Converter (DAC1) that is used to provide an analog voltage signal. The range of the output is  $\pm 10$  volts which provides a resolution of about 5 mV.

The DAC1 Output test exercises the DAC1 Output. The test outputs either a triangular waveform between  $\pm 10$  volts or a fixed voltage on the DAC1 Output.

### ATTENTION



Use caution when exercising the DAC1 output. Motion may occur when the voltage of the output changes if the output is connected to a device.

#### *To Test the DAC1 Output:*

- Connect a voltmeter or oscilloscope to the DAC1 output (P3-6) and to analog common (P3-4).
- From the Diagnostics menu, choose DAC1 Output.
- To output a triangular waveform with a range of  $\pm 10$  volts, select the Triangular Waveform button and then choose the Start button.

- To output a fixed voltage, select the Set DAC1 Voltage button, enter the desired voltage (between  $\pm 10$  volts), and then choose the Start button.
- Check the voltage signal on the voltmeter or oscilloscope.
- To stop the waveform or the fixed voltage signal, select the Set DAC1 Voltage button and enter a value of 0 volts. To stop the waveform or the fixed voltage signal AND close the dialog box, select the Close button.
- Once the test is running, to change from one type of waveform to the other, select the corresponding radio button, enter the desired voltage, then choose the Start button.

## Programmable Monitor Output

The Programmable Monitor Output Test exercises the Programmable Monitor output. The Programmable Monitor output is an 8 bit Digital to Analog Converter (DAC1) that can be used to monitor a variable. The range of the output is  $\pm 10$  volts. The test outputs either a triangular waveform between  $\pm 10$  volts or a fixed voltage on the Monitor output.

*To Test the Monitor Output:*

- Connect a voltmeter or oscilloscope to the Monitor output (P3-7) and to analog common (P3-4).
- From the Diagnostics menu, choose Programmable Monitor Output.
- To output a triangular waveform with a range of  $\pm 10$  volts, select the Triangular Waveform button and then choose the Start button.
- To output a fixed voltage, select the Set Monitor Voltage button, enter the desired voltage (between  $\pm 10$  volts), and then choose the Start button.
- Check the voltage signal on the voltmeter or oscilloscope.
- To stop the waveform or the fixed voltage signal, select the Set DAC1 Voltage button and enter a value of 0 volts. To stop the waveform or the fixed voltage signal AND close the dialog box, select the Close button.
- Once the test is running, to change from one type of waveform to the other, select the corresponding radio button, enter the desired voltage, then choose the Start button.

## Digital Outputs

The Digital Outputs test allows digital outputs to be forced on or off to test the outputs and system wiring. The relay outputs, Enabled and Ready, can also be turned on and off in this test.

### ATTENTION



Use caution when exercising digital outputs. Motion may occur when the state of the output changes if the output is connected to a device.

*To Test the Digital Outputs:*

- From the Diagnostics menu, choose Digital Outputs.
- To turn on an output, select the check box corresponding to the output number. To turn off individual outputs, clear the corresponding check box.



- To turn on all outputs, choose the All On button. Choose the All Off button to turn all outputs off.
- To turn the Drive Ready or Drive Enabled relay on or off, select or clear the check box next to each item.
- If there is an option card present with additional outputs, the Next Bank button will be enabled. Choose the Next Bank button to test the additional outputs. If an option card with additional outputs is not present, the Next Bank button will be disabled.

The outputs will be restored to their previous state when you select the Close button.

**Note:** The digital inputs and outputs can be monitored continuously by using the I/O Status Monitor.

---

## Operator Terminal

The Operator Terminal test verifies communication between the ULTRA Plus or IQ and the Operator Terminal. This test sends a string of characters to the Operator Terminal and displays the characters sent, so the Operator Terminal display can be visually checked. Any Operator Terminal keys that are pressed while in this test are displayed on the screen.

*To Test the Operator Terminal:*

- From the Diagnostics menu, choose Operator Terminal. The test will start immediately.
- Choose the Close button to stop the test.

Characters sent to the Operator Terminal are also sent to the dialog box for verification. Any key press action at the Operator Terminal (on or off) will also be shown in the dialog box.

---

## Nonvolatile Memory

The Nonvolatile Memory test does a nondestructive test of the ULTRA Plus or IQ nonvolatile RAM (NVRAM). If there is an option card present with additional program memory, the Next Bank button will be enabled. Choose the Next Bank button to test the additional memory. If an option card with additional memory is not present, the Next Bank button will be disabled. The number of passes completed and the current block being tested is shown. If there is a failure, a message box is displayed showing the address where the failure occurred. As long as there are no failures, the test continues until you choose the Close button.

---

## Fault History

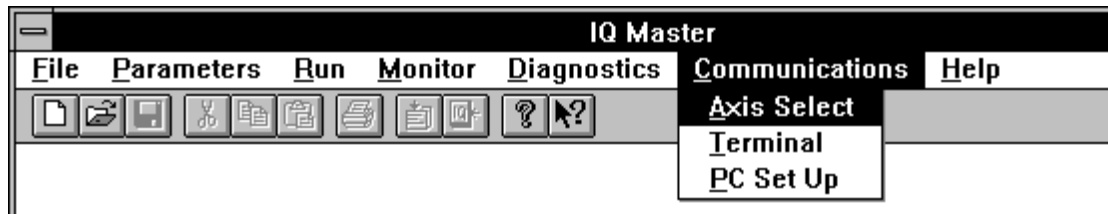
Select Fault History from the Diagnostics menu. Fault History shows the last 46 faults and the running time when the fault occurred. The running time is based on the first time the ULTRA Plus or IQ was powered up. The present running time is displayed at the top of the dialog box. The Fault History is saved in nonvolatile memory so it is not lost when power is removed from the ULTRA Plus or IQ.

---

## Version

Select Version from the Diagnostics menu to view the current version of the firmware in the ULTRA Plus or IQ, the model number and part number of the Personality Module, and the version of the Text Eprom in the ULTRA Plus or IQ. To view the version of IQ Master, select About IQ Master from the Help menu.

# Communications Menu



The Communications menu provides all the functions to select an axis on a multi-drop serial link, communicate using Terminal mode, and set the serial communication parameters for the PC.

## To Access Communications Menu Items

- Select the Communications menu by clicking on Communications in the menu bar,  
-or-  
press ALT to activate the menu bar, then C to pull down the Communications menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the 'P' in PC),  
-or-  
using the arrow keys to highlight your selection and then pressing ENTER.

## Axis Select

Select Axis Select from the Communications menu to select a single ULTRA Plus or IQ from a group of ULTRA Plus or IQ units daisy-chained together on a multi-drop RS-422 serial link. To establish communication, enter the address of the desired ULTRA Plus or IQ in the Axis Address box.

To communicate in multi-drop RS-422 mode, each ULTRA Plus or IQ must have had an address set previously on dip switch SW1 on the ULTRA Plus or IQ board. Addresses are in the range 1 through 63. If the address is left at 0 (default), the ULTRA Plus or IQ will not wait to be selected before transmitting and can cause garbled communication if on a multi-drop link. Set the address as a binary number with position 1 the least significant bit and position 6 the most significant bit of the address. If the address is changed, the change will not take effect until power is cycled or a Hardware Reset is performed.

Address 0 is a global selection address—no response is allowed. If address 0 is sent in the Axis Address box, all ULTRA Plus or IQ units in the system will execute any commands sent until a different address is selected. Since no response is allowed from the ULTRA Plus or IQ, commands will not be echoed back to the host, so Local Echo should be selected from PC Set Up when using Terminal mode in order to view the commands in the Terminal window.

After your initial selection, IQ Master will remember the last address that you selected (unless it was address 0) and display it the next time the Axis Select dialog box is displayed. The last address is valid only for the current session of IQ Master.

## Terminal

The Terminal mode allows direct communication to an ULTRA Plus or IQ or any serial device, including BRU-200 or BRU-500 amplifiers or PRO-Series positioners. All PC communication parameters can be set using PC Set Up from the Communications menu and can be changed even when the Terminal mode is running.

When using the Terminal to communicate with an ULTRA Plus or IQ, any valid Host command may be entered. Refer to Part 6 • Host Command Language for a detailed description of Host commands.

The Terminal window has a command line buffer (similar to Doskey). Up to the last 16 commands are saved. To recall a command, use the following keys (after entering at least one command):

UP ARROW	Recalls the command you used before the one displayed.
DOWN ARROW	Recalls the command you used after the one displayed.

You can edit the current command line. The following list describes the editing keys and their functions (see below for scrolling keys):

LEFT ARROW	Moves the cursor back one character.
RIGHT ARROW	Moves the cursor forward one character.
HOME	Moves the cursor to the beginning of the line.
END	Moves the cursor to the end of the line.
ESC	Clears the command from the display.
INSERT	Inserts text on the command line in the middle of old text without replacing the old text. Once you press ENTER, your keyboard returns to overtyping mode. You must press INSERT again to return to insert mode.

### Scrolling Keys

You can use the following keys to scroll the Terminal window (or use the scroll bars with the mouse):

SHIFT+UP ARROW	Moves the display up one line.
SHIFT+DOWN ARROW	Moves the display down one line.
SHIFT+LEFT ARROW	Moves the display to the left.
SHIFT+RIGHT ARROW	Moves the display to the right.
PAGE UP	Moves the display up one page.
PAGE DOWN	Moves the display down one page.
CTRL+HOME	Moves to the beginning of the display.
CTRL+END	Moves to the end of the display.

**Note:** To transfer a program, use Save Program to IQ or Open IQ Program from the File menu

**Note:** The position and size of the Terminal window is stored every time you close Terminal. IQ Master remembers where you last positioned the window and displays it in the same place the next time you open the Terminal window.

---

## PC Set Up

The serial communications settings for the PC are accessed from PC Set Up under the Communications menu. The settings are stored in the file MASTER.INI which is a standard Windows INI file. After changing any of the settings, choose the OK button to change the settings for the current session only. To make the changes permanent, choose the Save Changes button. The file can also be edited with any editor, but should not be edited while IQ Master is running. To change the serial settings for the ULTRA Plus or IQ, select Serial under the Parameters menu.

### Baud Rate

Baud Rate can be set to 1200, 2400, 4800, 9600 (default), or 19200.

### Communications Port

Com Port selects the communications port. If a Com Port is already open under another Windows application, it must be closed before it can be used in IQ Master.

### CR (Carriage Return) Translation

CR Translation can add a linefeed (LF) to a carriage return (CR) or just pass the CR on without adding a linefeed. This applies to Terminal mode only.

### Data Bits

Data Bits can set the number of data bits to 7 or 8. The default number of data bits for the ULTRA Plus or IQ is 8.

### Stop Bits

Stop Bits can set the number of stop bits to 1 or 2. The ULTRA Plus or IQ is always fixed at 1 stop bit.

### Flow Control

Flow Control determines what type of handshake to use—XON/XOFF for software handshake or RTS/CTS for hardware handshake. The ULTRA Plus or IQ only uses software handshake and does not support hardware handshake.

### Parity

Parity can be set to none (default), even, or odd.

## Local Echo

Local Echo determines whether to echo the character typed at the keyboard to the screen (known as half-duplex or HDX) or only put a character on the screen if it is received through the serial port (known as full-duplex or FDX). The ULTRA Plus or IQ runs in FDX mode. If double characters are seen on the screen when running in Terminal mode, turn off Local Echo.

## BRU Emulation

BRU Emulation allows communication with other Allen-Bradley products such as BRU-Series amplifiers, PRO-100 or -150 position controllers. Enabling (checking) BRU Emulation has no effect on communication with a ULTRA Plus or IQ. Terminal mode can also be used to communicate with a PRO-300, -400, or -450 position controllers regardless of the BRU Emulation setting.

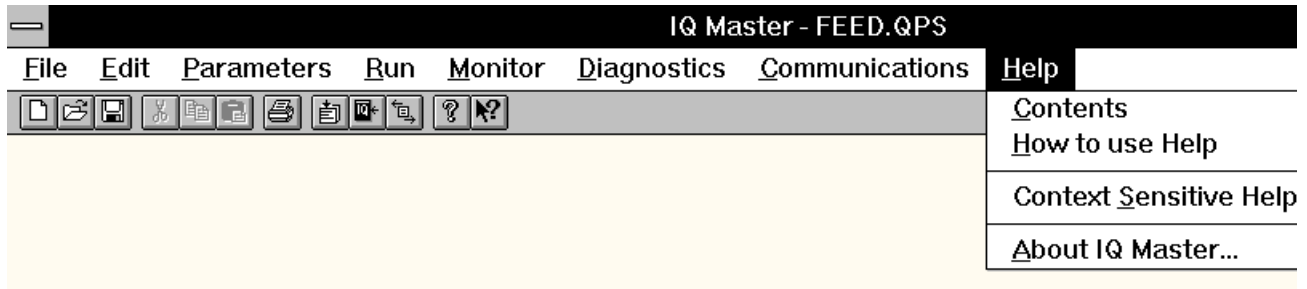
## Send File LF Toggle

Send File LF Toggle determines whether to send linefeeds (LFs) or to strip them when sending a file.

## Receive File CR Translation

Receive File CR Translation is similar to CR Translation, except it applies to receiving files only. If the incoming file includes linefeeds, set the translation to CR only.

# Help Menu



## To Access Help Menu Items

- Select the Help menu by clicking on Help in the menu bar,  
-or-  
press ALT to activate the menu bar, then H to pull down the Help menu.
- Select the menu item by clicking on your selection,  
-or-  
pressing the letter that is underlined in your selection (for example the “C” in Contents),  
-or-  
using the arrow keys to highlight your selection and then press ENTER.

You may access the Help information at any time by pressing the F1 key.

---

## Contents

The Contents section contains help on every menu and menu item, and the IQ Basic programming language.


---

## How to Use Help

This section provides information on how to use the on-line help.


---

## Context Sensitive Help

Select Context Sensitive Help from the Help menu, press Shift+F1, or click the  (context sensitive help) toolbar button to obtain help on some portion of IQ Master. The mouse pointer will change to an arrow and question mark indicating Context Help mode. Click anywhere in the IQ Master window, such as a Toolbar button or menu item, and the Help topic will be shown for the item you clicked.

---

## About IQ Master

Select About IQ Master from the Help menu or click the  toolbar button to display the version of IQ Master that you are using. Also displayed in this dialog box is the version of Windows that you are using and the amount of memory available.

MENUS



# Part 3

# Software and Hardware Integration

The interaction of IQ Master and the ULTRA Plus or IQ-Series controller is discussed in this part of the manual. You will learn how to configure your ULTRA Plus or IQ-Series controller for your application using IQ Master.

I/O

# Inputs and Outputs

## How to Turn an Input ON and OFF

An input is turned on by pulling the input pin to 24 Volt common. When the input is not pulled to 24 Volt common, it is OFF. Consult the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for voltage and current specifications of the inputs.

## What Does Turning ON and OFF an Output Do?

When an output is turned ON by the ULTRA Plus or IQ, the ULTRA Plus or IQ pulls that output pin to 24 Volt common. When an output is turned OFF, the pin is left to float. Consult the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* for voltage and current specifications of the outputs.

---

## General Purpose Inputs

Any digital input 1 through 16 may be used as a general purpose input. Several inputs also may have system functions that may be assigned to them. In addition inputs that are used as program select lines also may be used simultaneously as general purpose inputs. Inputs can be used as the condition for any conditional instruction.

## Associated Input Instruction

Name	Description <sup>1</sup>
IF ELSE	IF I1 = ON ...If input is ON then do something
WHILE	WHILE I1 = ON...While input is ON do something
DO WHILE	DO ... WHILE I1 = ON Do something while input is ON
WAIT	WAIT I1 = ON Wait until input is ON
$F_n = I_n$	$F1 = I1$ Set Flag equal to input
$S_n$ : IF	$S1$ : IF I1 = ON Scanned Event if input is ON

1. Refer to Part 5 • Language Reference detailed information.

Digital inputs are scanned once every velocity loop update (1mS). A flag is set according to the state of the input at the time of the scan. When a program instruction such as,

```
WAIT I1 = ON;Wait for input 1 to turn ON
is executed, the flag is tested not the physical input.
```

### Input Debounce Time

Debounce time affects when an input is recognized by the program. The debounce time is the time an input must remain stable, after it has changed state, before the flag is changed recognizing the change on the input. Debounce time applies to general purpose inputs only (NOT Assigned inputs). The debounce time is entered in the Parameter menu, Inputs dialog box.

## Assignable Inputs

All dedicated inputs can be disabled and used as general purpose inputs. Inputs are assigned in the Parameter menu, Inputs dialog box. Refer to Part 2 • IQ Master Environment, Parameter menu, Inputs, for more detailed information. The function of these inputs when they are assigned is described below

### Enable (I3)

The Enable input is used to enable and disable the ULTRA Plus or IQ. If the Enable input is not selected, the ULTRA Plus or IQ will be enabled on power-up and can be enabled and disabled ONLY by the ENABLE and DISABLE Host Language Commands or by ENABLE and DISABLE instructions in a program.

#### *When Enable Turns ON.*

When the Enable input turns ON, the ULTRA Plus or IQ will do exactly the same procedure as when the ENABLE instruction is executed. Refer to Part 5 • Language Reference, ENABLE, for more detailed information.

#### *When Enable Turns OFF*

When the Enable input turns OFF, the ULTRA Plus or IQ will do exactly the same Procedure as when the DISABLE instruction is executed plus these additional actions.

- If a program is running, the program is aborted.
- If a program was running AND outputs have default assignments to turn OFF (ON) when a program ends, they will turn OFF (ON). See Parameters, Default Outputs.
- If assigned, the Program Running output will turn OFF.

Refer to Part 5 • Language Reference, DISABLE, for more detailed information.

## Emergency Return (I10)

### *When Emergency Return Turns ON*

- If a program is running, it is aborted. Refer to Part 5 • Language Reference, ABORT, for more detailed information on what happens when a program is aborted.
- Once all motion has stopped, System program 26 (Emergency Return) is executed. Refer to Appendix A, System Programs, for more detailed information.

### *When Emergency Return Turns OFF*

This edge is ignored by the ULTRA Plus or IQ.

## Home Command (I6)

The Home Command input is used to start system program number 25. System program 25 is the location for the default home program. Refer to Appendix A for details on the default home program.

### *When the Home Command Turns ON*

System program 25 is started. This input is ignored if another main program is running.

### *When Home Command Turns OFF*

This edge is ignored by the ULTRA Plus or IQ.

## Home Switch (I5)

The Home Switch input can be used by the supplied home program (System Program 25) to identify the home position. The input can be defined as active OPEN, or active CLOSED to 24 Volt common in the Parameter menu, Inputs dialog box. For details on how this input functions with the home program refer to Appendix A, System Program 25, Home Program.

## Jog Forward (I7) and Jog Reverse (I8)

The jog inputs cause a jog command to be generated. Jog commands are controlled by three system variables: JVEL, JACCEL, and JDECEL. JVEL sets the speed of the jog. JACCEL sets the acceleration rate, and JDECEL sets the deceleration rate. The Jog Forward input will cause motion in the forward direction (position increasing), Jog Reverse input will cause motion in the reverse direction (position decreasing).

Jog commands use the same system resources as MOVV commands. When a program starts, JVEL is set to zero. If you want to use Jog inputs while a program is running, you must set JVEL in your program. After a MOVV has finished executing (MOVV = 0), JVEL is set to the Jog velocity value defined in the Parameter menu, Velocity/Accel dialog box.

**Note:** If a MOVV is being executed, jogging in the opposite direction will cause the MOVV motion to ramp to zero velocity. If the Jog inputs are released, the system will continue moving per the MOVV command. Jogging in the same direction will have no effect on the MOVV motion

### *When a Jog Input is Turned ON*

When the Jog input turns ON, a jog position command is generated and added to any GEAR or program MOVD or MOVV that may be in progress. The motion that is created by the Jog input is a ramp to JVEL velocity, using the JACCEL acceleration.

The ULTRA Plus or IQ must be Ready and Enabled for jogging motion to occur.

### ATTENTION



This is a level sensitive input. Even if conditions are not satisfied to permit jogging when this input is turned ON, jogging will begin as soon as they are satisfied. For example, if there is a fault and the jog input is turned ON, when the fault is cleared jogging will begin. If the jog input is turned ON while the ULTRA Plus or IQ is disabled, when the ULTRA Plus or IQ is enabled jogging will begin. If the jog input is on when power is applied to the ULTRA Plus or IQ it will begin jogging as soon as Ready.

#### *When a Jog Input is Turned OFF*

When a Jog input turns OFF, the jog position command is ramped down from its present velocity to zero using the JDECEL deceleration rate.

### Forward Travel Limit (I1) and Reverse Travel Limit (I2)

To help avoid safety hazards or equipment damage, both forward and reverse limit switches may be used. If enabled these inputs must be CLOSED to 24 Volt common for the motor to be enabled and a program run. The Forward and Reverse travel limit inputs are closed to 24 Volt common for normal operation, and open to indicate a fault condition.

#### *When a Travel Limit Turns ON (open)*

- If a program is running, it is aborted. Refer to Part 5 • Language Reference, ABORT, for more detailed information on what happens when a program is aborted.
- Once all motion has stopped, System Program 27, Error Routine is executed. Refer to Appendix A, System Programs, for more detailed information.

#### *When a Travel Limit Turns OFF (closed to 24 Volt common)*

This edge is ignored by the ULTRA Plus or IQ.

### Pause (I9)

The pause input is used to pause motion temporarily. When turned ON this input sets the system Pause flag ON. When turned OFF this input resets the system Pause flag. Refer to Part 5 • Language Reference, PAUSE for detailed information on what actions occur when the Pause Flag is set.

### Start (I4)

The Start input, if enabled, may be used to start a user program. The ULTRA Plus or IQ can be selected to start a specific program or a program selected by inputs. This selection is made in the Parameter menu, Inputs dialog box.

If the Start input is disabled, programs are started only from the Operator Terminal or the Host Language RUN command.

#### *When Start Turns ON*

The ULTRA Plus or IQ must be Ready, Enabled and NOT executing another main program when the Start input switches from OFF to ON for a program to start. When the Start input is turned ON the program selected as the default program or selected by program select lines will begin executing.

#### *When a Start Turns OFF*

This edge is ignored by the ULTRA Plus or IQ.

### Define Home

The Define Home input can be assigned to any general purpose input that is not already assigned to a dedicated function.

*When Define Home Turns ON*

For this input to function the ULTRA Plus or IQ must NOT be running a program (Define Home is ignored while a program is running). When Define Home turns ON the DHCMD flag is turned ON. This internal flag causes the present position to be defined as home (position zero). Refer to Appendix A, System Programs, for more detailed information.

*When Define Home Turns OFF*

This edge is ignored by the ULTRA Plus or IQ.

**Hardware Reset**

The Hardware Reset input can be assigned to any general purpose input that is not already assigned to a dedicated function.

*When Hardware Reset is ON*

- The ULTRA Plus or IQ stops executing any program,
- turns OFF the Ready output,
- disables the ULTRA Plus or IQ immediately (without decelerating the motor to a stop if moving),
- resets any faults,
- performs power-up diagnostic tests,
- turns ON the Ready output,
- enables, if there are no faults, and if the Enable input is on,
- turns ON the Enable Output, if the Enable input is on,
- and then starts the Auto Program (System Program zero).

This cycle is repeated as long as the Hardware Reset is ON

*When Hardware Reset is Turned OFF*

This edge is ignored by the ULTRA Plus or IQ.

**X Kill Motion**

The X Kill Motion input can be assigned to any general purpose input that is not already assigned to a dedicated function. The X Kill Motion is used to stop program motion and stop a program from executing. When turned ON, this input sets the system Stop flag ON. When turned OFF this input resets the system Stop flag. This input has the same effect as the STOP instruction. Refer to Part 5 • Language Reference, STOP for more detailed information.

**Note:** The X input does *not* stop Jog INPUT motion. Jog motion will slew to a stop and then begin again.

**ATTENTION**

It is the responsibility of the user to ensure that the application process stops in a safe manner when the application program stops. Failure to observe this precaution could result in severe bodily injury.

## Program Select Lines

Program select lines may be used to determine which program to execute when the Start input is turned ON. Up to 6 inputs can be used to select programs. Input I16 is always the most significant bit of the program select lines, input I15 is next significant and so on, down to input I11 if all 6 inputs are used. The inputs are decoded as a binary number. The number of select lines used, determines the program numbers that can be selected from the external inputs:

<u>Number of Select Lines</u>	<u>Possible Programs Selected</u>	<u>Inputs</u>
2	Up to 4 (0 to 3)	I16 and I15
3	Up to 8 (0 to 7)	I16 through I14
4	Up to 16 (0 to 15)	I16 through I13
5	Up to 32 (0 to 31)	I16 through I12
6	Up to 64 (0 to 63)	I16 through I11

### Binary Coded Program Select Lines

I16	I15	I14	I13	I12	I11	Program Selected
OFF	OFF	OFF	OFF	OFF	OFF	0
OFF	OFF	OFF	OFF	OFF	ON	1
OFF	OFF	OFF	OFF	ON	OFF	2
OFF	OFF	OFF	OFF	ON	ON	3
OFF	OFF	OFF	ON	OFF	OFF	4
:	:	:	:	:	:	:
ON	ON	ON	ON	OFF	ON	61
ON	ON	ON	ON	ON	OFF	62
ON	ON	ON	ON	ON	ON	63

**Note:** This table assumes that the Number of Program Select Lines is 6.

**Note:** If using 6 program select lines, inputs I11 and I12 would be the two least significant program select lines. These inputs are also used as interrupt inputs INT1 and INT2. If the interrupt lines are also used in a program, care should be taken to set the inputs to the proper state AFTER selecting the program to be run.

The maximum number of programs that an ULTRA Plus or IQ module supports is 32 without an optional Memory and I/O Expansion Card. If an Expansion card is present, space for an additional 32 programs is available (program numbers 32 through 63)

## Interrupt Inputs

There are two inputs that can function as software interrupt inputs, Input 11 and input 12. Input 11 is used to trigger interrupt 1, and input 12 is used to trigger interrupt 2. The software interrupts cause the microprocessor to read the positions of encoder 1 and 2 and save them into dedicated system variables. Input 11 can also be used to trigger a hardware latch to capture either encoder 1 position or encoder 2 position into the LPOS variable.

### Purpose

The purpose of an interrupt is to cause the ULTRA Plus or IQ to instantly record the positions of encoder 1 and encoder 2 in response to an external asynchronous event which triggers an input.



### Propagation Delay

The maximum time delay between the interrupt input turning ON and the position data being captured is 50 microseconds for the software interrupts. The hardware latch captures the position of one encoder in 1.5 microseconds.

### When an Interrupt Turns ON

If the interrupt is enabled, the positions of encoder 1 and encoder 2 are captured and stored in the appropriate system variable. See the table below.

### When an Interrupt Turns OFF

This edge is ignored by the ULTRA Plus or IQ.

### Associated Input Interrupt Instruction, Variables and Flags

Name	Description <sup>1</sup>
FI1	Flag to indicate that interrupt 1 has occurred
FI2	Flag to indicate that interrupt 2 has occurred
I1P1	Position from encoder 1, captured when interrupt 1 occurred
I1P2	Position from encoder 2, captured when interrupt 1 occurred
I2P1	Position from encoder 1, captured when interrupt 2 occurred
I2P2	Position from encoder 2, captured when interrupt 2 occurred
INT1	Enable and disable interrupt 1
INT2	Enable and disable interrupt 2
LPOS	Position from encoder 1 or 2 captured in hardware when interrupt 1 occurred
SINT2	Force an interrupt 2 with a software instruction.

1. Refer to Part 5 • Language Reference for detailed information.

## General Purpose Outputs

Any digital output 1 through 8 that is not assigned a dedicated function may be used as a general purpose output. Digital outputs are set once every velocity loop update (1mS). When a program instruction such as,

```
O1 = ON; Turn ON Output 1
```

is executed, a flag is set, not the physical output. Then, once every velocity loop update (1mS), the outputs are set according to the state of the internal flags.

General purpose outputs can be programmed to turn ON, turn OFF or remain unchanged when a program stops executing for any reason. Refer to Part 2 • IQ Master Environment, Parameter menu, Default Outputs, for more detailed information.

## Assignable Outputs

All dedicated outputs can be disabled and used as general purpose outputs. Outputs are assigned in the Parameter menu, Output dialog box. Refer to Part 2 • IQ Master Environment, Parameter menu, Outputs, for more detailed information. The function of these outputs when they are assigned is described below.

### AtHome (05)

The AtHome output can be enabled to provide an output that indicates when the system is at the home position. You can refer to Part 5 • Language Reference, ATHOME flag for more detailed information, and Appendix A for detailed information on the default home program.

The AtHome output uses the In-Position Window size, but not the In-Position Window time to determine if it should turn ON. When the feedback position POSN has been within the In-Position Window size of the zero position, the AtHome Output will turn ON. It is OFF otherwise. The In-Position Window size is set in Parameter menu, Gains/Limits dialog box. The window size also may be changed at runtime using the WIN instruction.

**Note:** Home position must have been defined AND the system must be in the Absolute mode to use the At-Home output.

### Error (08)

The Error output can be enabled to indicate if there is a system error. Refer to Appendix D, Error Messages for the list of errors that will turn ON the Error output. In addition to errors, this output can be programmed to turn ON when the ULTRA Plus or IQ is disabled. Refer to Part 2 • IQ Master Environment, Outputs for more detailed information.

This output turns OFF when the error is cleared by: disabling and enabling the ULTRA Plus or IQ, issuing a Reset or HRESET, cycling power OFF and ON, or in the case of a travel limit fault, by jogging OFF the limit switch. If the condition remains that caused the fault, the Error output will turn ON again.

### Home Sequence Complete (06)

The Home Sequence Complete output, if enabled, will turn ON when the ULTRA Plus or IQ has been homed. Refer to Part 5 • Language Reference, HSEQCPL for more detailed information, and Appendix A for detailed information on the default home program.

### Program Running (04)

If enabled, the Program Running output will be ON when a program other than an Fkey program is running; otherwise Program Running will be OFF.

### In-Position (07)

The In-Position output is ON when the INPOSN flag is ON and it is OFF otherwise. Refer to Part 5 • Language Reference, INPOSN flag for more detailed information.

#### ATTENTION



External circuitry must be used to ensure the state of digital outputs in an emergency stop situation. The ULTRA Plus or IQ-Series will not automatically turn outputs off.

---

## Enabled Output (P3-2)

The Enabled output is a normally open relay that is closed when the ULTRA Plus or IQ is enabled and opened when the ULTRA Plus or IQ is disabled. Relay Common (P3-3) is the other contact of this relay. The Enable relay will open if any fault occurs in the ULTRA Plus or IQ.

---

## Ready Output (P3-1)

The Ready output is a normally open relay that closes when the ULTRA Plus or IQ is powered up with no faults. Relay Common (P3-3) is the other contact of this relay. The Ready relay will open if any fault occurs in the ULTRA Plus or IQ.

---

## DAC1 Output (P3-6)

A 12-bit digital to analog converter is provided for setting analog voltages. The DAC1 instruction is used to set the voltage. Refer to Part 5 • Language Reference, DAC1, for more detailed information.

### DAC1 Value

The DAC1 Value set in the Parameter menu, Outputs, establishes the default value for the digital to analog converter output, in volts. This is the value the DAC1 output is set to on power-up and when a program is not running. The range is  $\pm 10$  volts with about 5 mV resolution (default setting is 0 volts).

---

## ADC1 Input (P3-5)

The Analog to Digital Converter 1 (ADC) may function as a general purpose input or may be used to set the feedrate. This choice is made in the Parameter menu, Inputs, dialog box.

If set to a general purpose ADC the ADC1 instruction is used to read the voltage. Refer to Part 5 • Language Reference, ADC $n$  for more detailed information.

If the ADC1 input is used as feedrate input, the ADC1 input voltage can range from 0 to 10 volts, corresponding to 0 to 200% feedrate.

---

## Encoder 2

Encoder 2 can perform three different functions: It can be used to input a second encoder, to pass encoder 1 signals out to another device, or to read step and direction inputs from an indexer. These functions are determined by the setting in the Parameter menu, Feedback Configuration dialog box.

# Starting a Program

There are several ways to start a program. They include: Assigned Inputs, the IQ Master Run Control Dialog box, Operator Terminal Fkeys, and Host Language Commands.

In all cases the following conditions must be satisfied before a program will start:

- The ULTRA Plus or IQ must be enabled.
- The ULTRA Plus or IQ must be ready (no faults exist).
- The ULTRA Plus or IQ must not be running a main program already.

---

**ATTENTION**

Understand the application before starting. A program's outputs may change state, resulting in machine movement. Failure to observe these precautions could result in severe bodily injury.

---

---

**ATTENTION**

The user must provide an external, hardwired emergency stop circuit outside the controller circuitry. This circuit must disable the system in case of improper operation. Uncontrolled machine operation may result if this procedure is not followed. Failure to observe this precaution could result in severe bodily injury.

---

---

## Assigned Input

If assigned, the *START* input can be used to start a program. Refer to the *START* input described earlier in Part 3.

---

## IQ Master - Run Control

The Run Control dialog box of IQ Master can be used to select and run a program. Refer to Part 2 • IQ Master Environment, Run menu, Run Control, for more detailed information.

---

## Function Key Program

If you have an Operator Terminal, the default System Program 5 can be used to start your programs. Refer to Appendix A, System Programs, for more detailed information. System programs are not fixed. You can move them and change them to fit the needs of a particular application.

---

## Host Language Commands (Serial)

The Host Language Command, *RUN* can be used to start a program in the user directory. The Host Language Command, *SRUN* can be used to start a program in the system directory. Refer to Part 6 • Host Language Commands, *RUN*, for more detailed information.

# Stopping or Suspending a Program

---

## ATTENTION



It is the responsibility of the user to ensure that the application process stops in a safe manner when the application program stops. Failure to observe this precaution could result in severe bodily injury.

---



---

## Assigned Inputs

The following inputs may be used to stop or suspend a program from executing if assigned. Each input has various additional effects. Refer to Part 3 • above, for more detailed information.

Name	Description
ENABLE	Turn OFF to disable the ULTRA Plus or IQ, stopping motion and the program.
X-Kill Motion	Stop motion and stop the program.
PAUSE	The PAUSE input will cause motion to ramp to a stop. The program will remain running.

## Host Language Commands (Serial)

The following Host Language Command may be used to stop or suspend a program from executing. Each command has various additional effects. Refer to Part 6 • Host Language Commands, for more detailed information.

Name	Description
X	X-Kill motion.
DISABLE	Disable the ULTRA Plus or IQ.
RESET	Soft Reset the ULTRA Plus or IQ.
HRESET	Hardware Reset - Simulate turning the power OFF and ON.
PAUSE	The program will pause at the stop of the next motion instruction in the program. If the PAUSE command is received during a motion instruction the program may pause on that instruction. Refer to Part 5 • Language Reference, PAUSE for more detailed information.

## Program Instructions

The following program instructions may be used to stop or suspend a program from executing. Each command has various additional effects. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
ABORT	Abort the execution of a program, finish the current move.
STOP	Stop motion and program.
PAUSE	Pause Motion. The program continues to execute until it reaches a motion instruction.
DISABLE	The DISABLE instruction will cause the program to “hang” if the ULTRA Plus or IQ is still disabled at the next motion instruction.
PURGEMOTION	Stop motion. Program continues to execute and motion statements are ignored until PURGEMOTION is assigned a value of 0.

## Fault Conditions

Normally any fault will cause the ULTRA Plus or IQ to DISABLE, stopping motion and program execution. Refer to Part 5 • Language Reference, DISABLE for more detailed information.

On some machines you may want the ULTRA Plus or IQ to continue running even after some faults occur. This is accomplished by setting Disable on Fault to Partial. In this case, the following faults are disabled: Iavg fault, Motor Overtemperature, Soft Forward Limit, and Soft Reverse Limit. All faults will cause the Error output to turn ON (if enabled). However, if Disable on Fault is set to Partial, and a fault listed above occurs, the ULTRA Plus or IQ will remain enabled and execute motion programs normally. The Error output in this case serves as an alarm, and a programmable logic controller (PLC) or other controller can then gracefully shut down the machine without damaging the tooling or work piece. This setting is made in the Parameter menu, System dialog box of IQ Master, or with the FLTDIS Host Language Command. Refer to Part 2 • IQ Master Environment, Parameter menu, System, or to Part 6 • Host Language Commands.

# Faults

---

## When a Fault Occurs

When a fault condition has been detected by the ULTRA Plus or IQ the following events occur:

- If a program is executing, it is stopped. Refer to Part 5 • Language Reference, STOP for more detailed information.
- If a program is running and the Program Running output is assigned, the Program Running output will turn OFF.
- If the ULTRA Plus or IQ is enabled, it is disabled. Refer to Part 5 • Language Reference, DISABLE, for more detailed information.
- If the Error output is assigned, it is turned ON. Refer to Part 2 • Software and Hardware Integration, Input and Outputs, for more detailed information.
- If a program was running and outputs are assigned to turn OFF (ON) they will turn OFF (ON).
- The Enabled and Ready relay outputs will both open.
- System Program 27, Error Routine, will run. This program will print the error message to the Operator Terminal screen. This program may be modified to perform additional tasks upon occurrence of a fault.

When the condition that caused a fault is eliminated, the fault does not automatically clear. An action must be taken by something other than the ULTRA Plus or IQ to clear the fault and enable the ULTRA Plus or IQ again. For example if a travel limit was hit, the ULTRA Plus or IQ remains disabled even after the switch is cleared.



## Clearing Faults

Clearing a fault will not fix the cause of the fault. The source of the fault should be investigated and fixed before returning the system to operation. The following procedures describe various methods for clearing faults and enabling the ULTRA Plus or IQ. When a fault is cleared using one of these procedures, the following events will occur:

- The Ready relay will close.
- If assigned, the Error output will turn OFF.
- The commanded position will be set to the feedback position.
- The ULTRA Plus or IQ will be enabled and the Enabled relay will close.
- If assigned the In-Position output will turn ON.

If the condition that caused the fault to occur is still present, the ULTRA Plus or IQ will repeat the steps that occurred when the fault was first detected.

### Enable Input

Turning the Enable input OFF and ON will clear a fault and enable the ULTRA Plus or IQ.

### Jog Inputs

If the fault was a travel limit (hard or soft), the jog inputs may be used to clear the fault. If the forward limit was reached, turn ON the jog reverse input to jog off the limit. If the reverse limit was reached, turn ON the jog forward input to jog off the limit. Once clear of the limit, turn OFF the jog input. The ULTRA Plus or IQ will remain enabled (the fault will clear) and ready.

### Cycle Power

Turn the Power OFF and then back ON. This will have the following additional effect:

- ALL volatile system Flags and variables will be initialized.
- V variables will be set to zero.
- F variables will be set to OFF.
- Position will be set to zero. If the system is in Absolute mode it will need to be homed.
- ALL outputs will be turned OFF. Outputs set to default to ON will turn ON following the execution of the Auto program.

### Run Menu

#### *RESET*

Send a Reset command to the ULTRA Plus or IQ. Refer to Part 2 • IQ Master Environment, Run menu for more detailed information.

#### *HRESET*

Send a Hard Reset command to the ULTRA Plus or IQ. Refer to Part 2 • IQ Master Environment, Run menu for more detailed information.

### Host Language Commands

#### *RESET*

Send a Reset command to serial port 2. Refer to Part 6 • Host Language Commands for more detailed information.

#### *HRESET*

Send a HRESET command to serial port 2. This will have the same effect as cycling power (see above). Refer to Part 6 • Host Language Commands for more detailed information.

*JOGF / JOGR*

Send a JOGR command to serial port 2 to jog off a forward travel limit. Send a JOGF command to serial port 2, to jog off a reverse travel limit switch. These commands will jog the motor in the opposite direction of the limit switch. When the jog is stopped, if the system is off the travel limit switch, the ULTRA Plus or IQ will remain enabled and ready. Refer to Part 6 • Host Language Commands for more detailed information. This procedure also applies to soft travel limits while in Absolute mode.

# Operator Terminal

---

## Installation

Installation instructions for the Operator Terminal are in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004).

---

## Displaying & Reading Data

The following instructions can be used to print text and data and read data from the Operator Terminal. Refer to Part 5 • Language Reference, for more detailed information.

Name	Description
CLEAR	Clear all or part of the operator display
OTMON	Monitor 1 of 34 different system variables
PRINT	Print text and data to the operator terminal
READ	Read data from the terminal

---

## Function Keys

Refer to Part 2 • IQ Master Environment, Parameter menu, Fkey Set Up.

## Xkeys

There are four Xkeys on the operator terminal. These keys are used to activate Xkey routines that are written within an application program. The application program has control over when the Xkeys are active using the *Xn* function. If an Xkey is pressed while it is active, the Xkey routine will begin executing in parallel with the main program.

Xkey routines are frequently used to enter data into the program. By using Xkey routines, the operator can be permitted to enter data without interfering with the normal operation of the machine. This will permit the operator to enter data for the next operation while the current process is finishing.

For a more detailed explanation of Xkey routines and programming refer to Part 4 • Programming. The following table lists the functions that work with the Xkeys and Xkey routines.

Name	Description
XEND	Marks the end of an Xkey routine.
<i>Xn</i> :	Marks the beginning of an Xkey routine.
<i>Xn</i>	Enable and disable an Xkey.
XNACTIVE	Reports the state of the Xkey (ON if pressed, OFF otherwise).
XNPGM	Simulate the pressing of an Xkey from within a program.

## Yes / No Keys

The YES and NO keys are used with the READ instruction to set the value of flag variables. When a READ flag variable instruction is executed only the YES, NO and ENTER key are valid responses. Pressing YES then ENTER will turn the flag ON. Pressing NO then ENTER will turn the flag OFF. If SHOW was used with the READ instruction, pressing ENTER will leave the variable unchanged.

## Status Key

The operator terminal has nine status screens that may be accessed by pressing the STATUS key. Each time the STATUS key is pressed the next status screen is displayed. The BACKSPACE key can be used to scroll backwards through the status screens. The CLEAR Key will remove the status display or clear the peak values. The following variables are available on status screens:

Name	Description <sup>1</sup>
Status Flags	ATHOME, HSEQCPL, INPOSN, ERROR, XFER, READY, ENABLED, PAUSE
Program Status	Program Number. If compiled with the debug option ON, the instruction that is executing will also be displayed.
Position Status	PCMD, POSN, FE
Velocity Status	VCMD, FVEL1, VEL2
Input's Status	Inputs 1 through 16 (up to 48 if expanded I/O is installed)
Output's Status	Outputs 1 through 8 (up to 24 if expanded I/O is installed)
Current Status	ICMD, IAVG, IN PEAK CURRENT

Commanded Position Status	PGEN, PJOG, PEXT
Peak Status	PFE, PVEL1, PICMD

1. Refer to Part 5 • Language Reference for detailed information.

All position values displayed on the operator terminal status screens are in user units, velocity values are in user units per timebase, and torque variables are in amperes.

# I/O Expansion

Additional I/O may be added with either an I/O Expansion Card *or* a Memory and I/O Expansion Card. These boards add the following input and output capabilities to the ULTRA Plus or IQ.

- 32 TTL Inputs
- 16 TTL Outputs
- 4 twelve bit Analog inputs (only with the Memory and I/O Expansion Card)

For detailed specifications and installation instructions refer to the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004).

---

## Using the Additional I/O

### Inputs

The additional inputs can be used just like the inputs on the base ULTRA Plus or IQ. The expansion I/O inputs are numbered I17 through I48.

### Outputs

The additional outputs can be used just like the outputs on the base ULTRA Plus or IQ. The expansion I/O outputs are numbered O9 through O24.

### Analog Inputs

The additional analog inputs can be used exactly like the analog inputs on the base ULTRA Plus or IQ. The expansion I/O analog inputs are referenced as ADC2, ADC3, ADC4 and ADC5. (Feedrate may only be assigned to ADC1)

# Expansion Memory

Additional program memory may be added with the addition of an Memory and I/O Expansion Card. This board adds 32 Kbytes of addition nonvolatile program memory to the base ULTRA Plus or IQ for the storage of up to 32 additional programs.

For detail specifications and installation instructions refer to the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004).

---

## Using Expansion Memory

### Storing Programs in Expansion Memory

Programs located in expansion memory appear in a different directory than programs located in the base ULTRA Plus or IQ. Base ULTRA Plus or IQ programs are in Program Directory 1 and programs in expansion memory are in Program Directory 2.

### Running Programs Stored in Expansion Memory

When using Program Select Lines or Host Language Commands, programs that reside on the Memory and I/O Expansion Card are referenced by program numbers 32 through 63.

I/O



# Part 4

# Programming

This part starts out with a tutorial on programming the ULTRA Plus or IQ which is designed to get you up and running quickly. From there it moves into a detailed discussion of programming. The chapters in this part are broken down into the major components of your program, for example: motion, math, etc.

TUTORIAL

# A Tutorial Introduction

This tutorial demonstrates the operation of the ULTRA Plus or IQ system and introduces the basic system commands. It illustrates how to write a program quickly and spin a motor. The step by step section demonstrates how to create and run programs on the ULTRA Plus or IQ. The tutorial introduces all the main concepts and provides a frame of reference for the remainder of Part 4.

---

## Getting Started

### How to Communicate with the ULTRA Plus or IQ

The IQ Master environment contains all the software tools needed to program and debug an ULTRA Plus or IQ system. It contains a full-screen text editor, program compiler, status and monitor information, diagnostics, on-line help and a host command terminal. If IQ Master is not already installed on your system, install it now, following the instructions on the IQ Master disk and the setup program.

For the purposes of this tutorial, the motor used should be disconnected from any load. This is to avoid damage to equipment and personal injury, and so that program results will be predictable.

Communications from the IQ Master program to the ULTRA Plus or IQ is done via a serial link. If you have not already established communications refer to the “Quick Start Check List” in Part 1 of this manual. Once the serial cable in is place, run IQ Master and apply power to the ULTRA Plus or IQ. Select the Communications, PC Set Up menu item and select the Com Port that your PC is using. To select a menu item using a keyboard, press the ALT key along with the underlined letter of the menu title and then the underlined letter of the menu item.

### Touring a Dialog Box

Dialog boxes can be manipulated with a mouse or keyboard. A mouse is usually easier to use as it just involves moving the cursor with the mouse and pressing the button to activate the item being clicked on. To move from one dialog item to the next with the keyboard, press the TAB key. If a check box (a square box with or without an “x” in it) is highlighted, press the space bar to toggle it. If a radio button (a circle with or without a dot in the middle) is selected, press the arrow keys to select a different radio button. In many dia-

log boxes there are groups of items with an underlined letter in the group name (for example the System Values group in the Parameters System dialog box). The cursor can be moved quickly to this group by pressing ALT+x where x is the underlined letter for that group. From there, the TAB and arrow keys can be used as before.

When a button is highlighted (like the OK button in the following dialog) press the Enter key to activate the button.

## Hardware Configuration Considerations

Display the following dialog box by selecting the System menu item from the Parameters menu and set up your dialog box to look like this one. In this example, the system will not be in the Absolute mode. The Absolute Mode On check box should be left cleared (unchecked). You will not be able to enter values for Forward Limit, Reverse Limit, and Ereturn Position.

The screenshot shows a dialog box titled "System". It contains several sections:

- System Values:**
  - Absolute Mode On:
  - Forward Limit: 1000.0000
  - Reverse Limit: -1000.0000
  - Ereturn Position: 0.0000
  - Scale: 8000.0000
  - Scale2: 4000.0000
- Disable on Fault:**
  - All
  - Partial
- Rotation:**
  - CW
  - CCW
- Rotation2:**
  - CW
  - CCW
- Latched Position:**
  - Pos1
  - Pos2

At the bottom, there are two buttons: "OK" and "Cancel".

### Scale

The Scale parameter tells the ULTRA Plus or IQ how many encoder counts to move when it is told to move one unit. Setting the Scale to 8000 means that when told to move one unit, the motor will move 8000 pulses. For motors with 2000 line encoders (most standard Allen-Bradley motors) this will be one revolution (since the encoders are read in full quadrature, 4 encoder counts per encoder line.) If your motor has a different resolution encoder, enter the number of encoder counts per revolution in the Scale box so that the units will be revolutions. The Rotation parameter tells the ULTRA Plus or IQ which direction is forward: Clockwise or Counter-Clockwise while looking at the face of the motor. Tell the ULTRA Plus or IQ to start using these new parameters by pressing the OK button.

### Timebase

The next parameter that needs to be set is the Timebase. Select the Velocity/Acceleration menu item from the Parameter menu to display the following dialog box.

The image shows a software dialog box titled "Velocity/Acceleration". It is divided into several sections:

- Defaults:** Velocity (1000), Acceleration (200), Feedrate (100), Slew On (checked), Slew Value (300).
- Jog:** Velocity (50), Acceleration (20), Deceleration (200).
- Home:** Velocity (50), Offset (0.452).
- Timebase:** Radio buttons for Minutes (selected) and Seconds.
- Overspeed:** 3000.

At the bottom of the dialog are three buttons: "Update", "OK", and "Cancel".

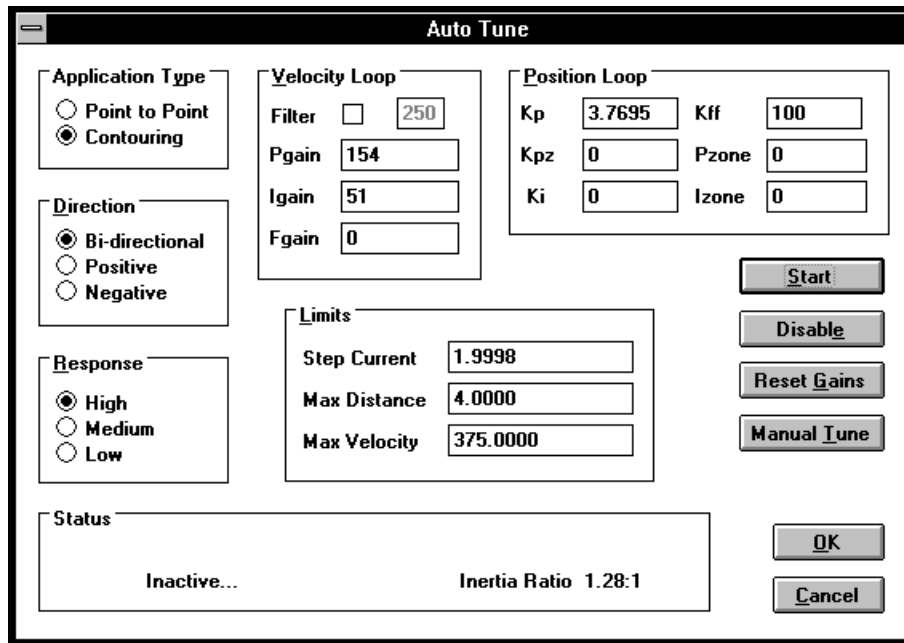
The Timebase parameter specifies the time used for Velocity (minutes or seconds). Select the Minutes radio button. Since the units have already been set up to be revolutions (using the Scale parameter), with a Timebase of minutes, velocities will be in units of revolutions per minute or RPM.

## Enable

If the motor does not have holding torque (it turns freely) the ULTRA Plus or IQ is probably not enabled. If the Enable input is selected in the Inputs dialog box, make sure the Enable input (input 3) is turned on (by connecting P1 pin 4 to P1 pin 1). If the motor still does not have holding torque, select the Enable menu item from the Run menu. The ULTRA Plus or IQ can be disabled by either the Enable input turning off or by issuing a DISABLE command from the Run menu.

## Auto Tune

The last thing that needs to be done before the motor can be moved is to tune the ULTRA Plus or IQ. The parameters that came with the personality module may be good enough to turn a motor without a load connected, but just in case, we will go through the Auto Tune procedure. Display the Auto Tune dialog box by selecting the Tune menu item from the Run menu. This will display the following dialog box.

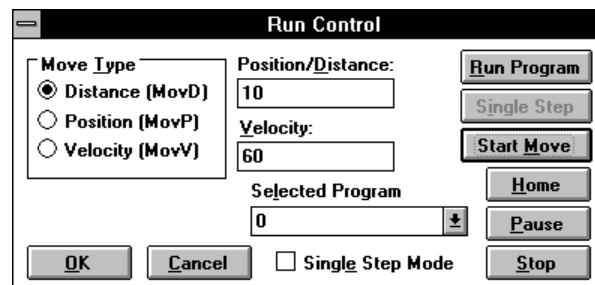


Set the Application Type, Direction, and Response as shown. Select the Start button to start the Auto Tune procedure. This will cause the motor to vibrate for a few seconds. During this time, the ULTRA Plus or IQ is measuring the load inertia and, based on this measurement, is selecting appropriate gain settings. Select the OK button to accept the gain settings and close the dialog box. The motor should now be ready to move.

## Motion

### Incremental Motion

Select Run Control from the Run menu or press F8 to display the following dialog box.



If it is not already selected, set Move Type to Distance (MOVD), set the Position/Distance value to 10, and set the Velocity value to 60. Select the Start Move button (or press ALT+M) to start the move. The motor should take about 10 seconds to do the 10 revolutions. The motor shaft should be in exactly the same position after the move as it was before the move.

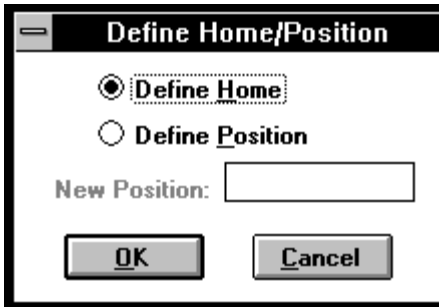
### Homing

If the Move Type is changed to Position (MOVP) and the Start Move button is selected, a move error will occur if the ULTRA Plus or IQ has not been homed since it was powered up. The reason for the error is that the ULTRA Plus or IQ is being told to go to a specific location, but it does not know where zero is—it does not know where the specific location is either. Before any MOVP commands can be

executed, home position must be defined.

Home position can be defined in a number of ways: an input can set the current position to the home position; a Home Program can be run to find a set home position (see Appendix A); or a command can be issued to define the current position.

We will now use the Define Home command to define the home position. Select the Define Home/Position menu item from the Run menu. This will display the following dialog box.



Select the OK button to set the current position to 0 and to set the Home Sequence Complete output. If you knew the motor was at some given position, for example, 0.25 turns from 0, you could use the Define Position option to set the current position to 0.25.

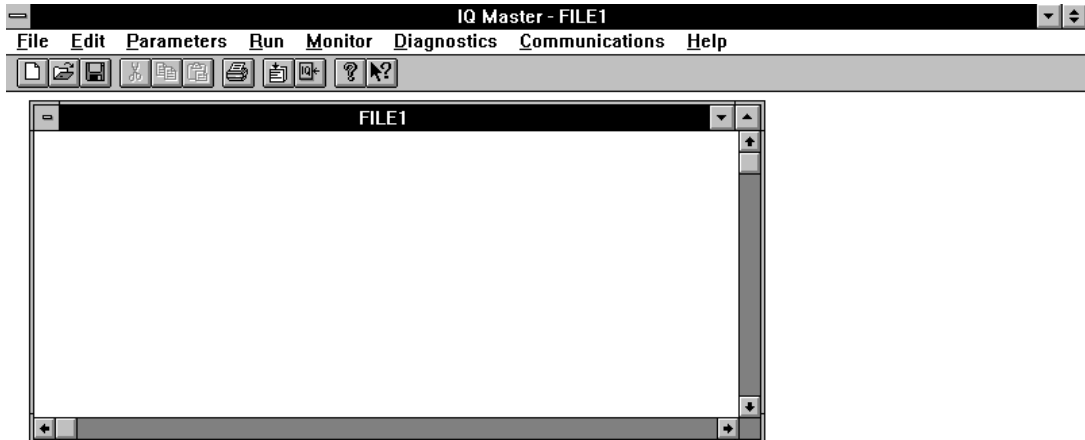
### Absolute Motion

Once the home position has been defined, absolute moves can be done. To try this, display the Run Control dialog again if needed by selecting Run Control from the Run menu or press F8, then execute an absolute move (MOV<sub>P</sub>).

## Programming Basics

Until now all moves have been done in what is called immediate mode: the ULTRA Plus or IQ is told to move and it moves. It is usually more useful to have all the moves a machine will do, along with I/O statements, etc., in a program. We will now explore how programs are written and used.

Select New from the File menu. A window will be displayed and your screen should look something like this:

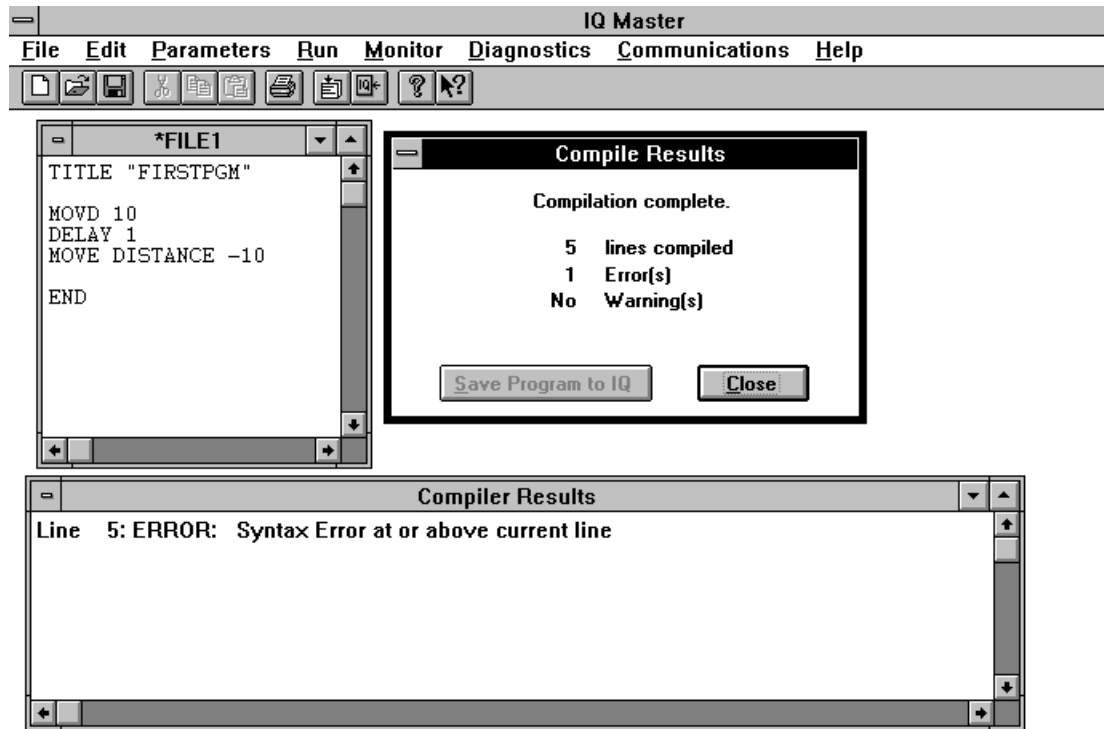


In the FILE1 window, type the following text.

```
TITLE "firstpgm"  
  
MOVD 10  
DELAY 1  
MOVE DISTANCE - 10  
END
```

After the text has been entered, select Compile from the Edit menu or press F2. A dialog box showing that the program is being compiled will be displayed. In a few seconds, your screen should look something like this.





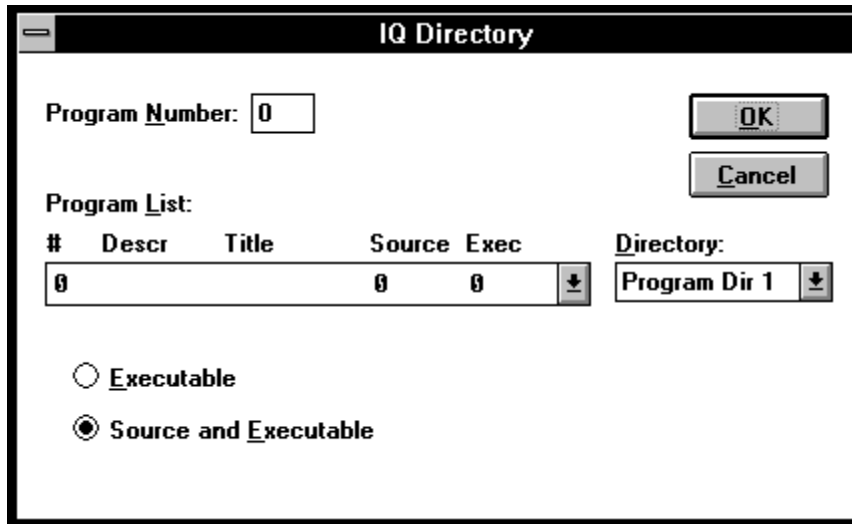
Select the Close button or press Enter on the keyboard to close the Compile Results message box.

At the bottom of your screen there will be a window showing the Compiler errors and warnings that have been generated. In this case, there is a Syntax Error in line 5. If the cursor is in the Edit window (the one that says **\*FILE1** at the top) the bottom right-hand corner shows what line the cursor is on. Move the cursor to line 5 (the line with the Syntax Error) by using the arrow keys or the mouse. The line `MOVE DISTANCE -10`, generated a Syntax Error because “MOVE DISTANCE” is not a valid command. Select the text “MOVE DISTANCE” by either pressing and holding the mouse button when the mouse cursor is just before the M and dragging the cursor to the E in DISTANCE, or positioning the cursor just before the M, holding down the SHIFT key and using the arrow buttons to move the cursor to after the E in DISTANCE. “MOVE DISTANCE” should now be highlighted. Replace MOVE DISTANCE with MOVD by typing MOVD. (Typing while text is selected will replace the selected text with what is typed.) Press F2 to compile. There should no longer be any errors.

Save the program to disk by selecting Save As from the File menu. This will display a dialog box asking for the name of the program (it is possible leave the name “File1”) and to select the directory where you want the program to be stored.<sup>1</sup> Name the program `LEARN.QPS`. Once the program is named and the directory selected, select the OK button to save the file. Notice that the \* in front of the name at the top of the editor window has disappeared. The \* means that changes have been made since the program was last saved.

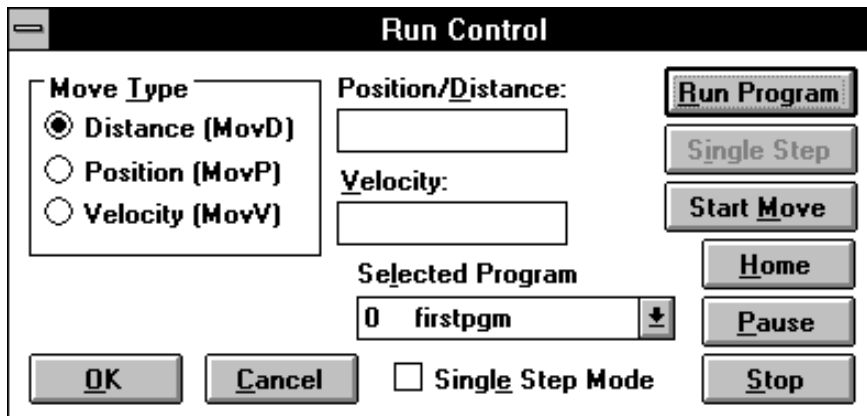
Next save the program to the ULTRA Plus or IQ by selecting Save Program to IQ from the File menu. This will display the following dialog box:

1. The default directory can be set from the Windows Program Manager by clicking once on the IQ Master icon, selecting Properties, and typing the DOS path that you would like to be the default directory into the Working Directory field of the dialog box.



If a program has already been stored in the ULTRA Plus or IQ as program #0, the title (if it exists) and the source and executable size will be displayed. Click the Program Number box or press ALT+N to select the program number you wish to save to. Click on the down-arrow just to the left of the Directory box to scroll through the existing programs. Select Source and Executable to store the program source text as well as the executable files to the ULTRA Plus or IQ. Select Executable if you wish to save only the executable file. Set up this dialog box to save the program source and executable into program number 0. Select the OK button or press Enter to save the program to the ULTRA Plus or IQ.

To run this program, select Run Control from the Run menu or press F8 to display the Run Control dialog box again.



Click the down-arrow by the Selected Program box or press ALT+L and use the up and down arrows to select the program to run. Select program 0, then select the Run Program button or press ALT+R. The motor should move 10 revolutions forward, wait for a second and then move 10 revolutions backward. Select OK to close the Run Control dialog box.

# Inputs and Outputs

## Dedicated Inputs Setup

The ULTRA Plus or IQ has 16 digital inputs and 8 digital outputs. Some of these inputs and outputs have dedicated functions associated with them. For example, if the Forward Limit feature is used, the Forward Limit Switch must be connected to input I1. The input functions are enabled or disabled using the Inputs dialog box from the Parameters menu:

Display this dialog box and set the inputs as shown and select the OK button. With the inputs set as shown, only the Jog input functions are active. If input I8 is activated (by connecting P1 pin 9 to P1 pin 1) the motor will accelerate using the Jog Acceleration parameter to the Jog Velocity. If I7 is activated (by connecting P1 pin 8 to P1 pin 1) the motor will do the same thing, but in the opposite direction. If both inputs are activated the motor will stop because the ULTRA Plus or IQ is being told to move in opposite directions at the same speed which results in no movement.

## Read Inputs

We will now modify the LEARN.QPS program to look for an input to go on before doing the first move, and wait for the input to go off before doing the second move. Modify the program to look like this:

```
TITLE "firstpgm"

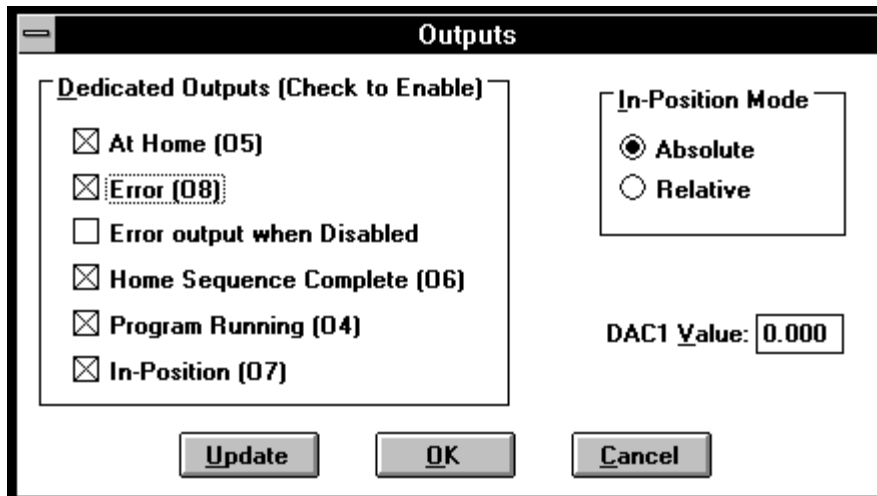
WAIT I16=ON
MOVD 10
WAIT I16=OFF
MOVD - 10
END
```

The program waits for I16 to go on, does a move of ten revolutions, waits for I16 to go off, does a move of minus ten revolutions and then ends. Compile the program, save it to the ULTRA Plus or IQ, and run it. When the program is run, the motor will not move until I16 is activated by connecting P2 pin 3 to P1

pin 1. After I16 turns on, the motor will move ten revolutions. When the move is done, the ULTRA Plus or IQ will wait until I16 is off and then move ten revolutions backward.

## Set Outputs

Some of the eight digital outputs available on the ULTRA Plus or IQ also have selectable dedicated functions. These are selected with the Outputs dialog box from the Parameters menu. Display this dialog box, set the outputs as shown, and select the OK button to accept these assignments.



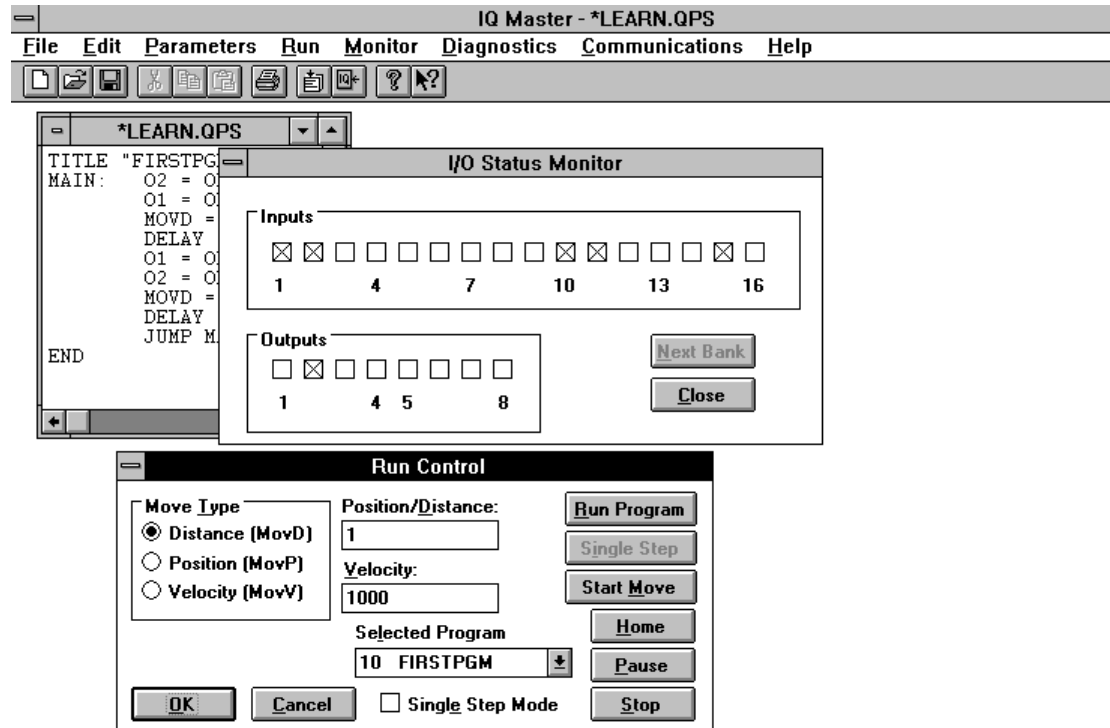
Modify the LEARN.QPS program to look like this:

```
TITLE "firstpgm"
MAIN:O2 = OFF
O1 = ON
MOVD = 10
DELAY 0.5
O1 = OFF
O2 = ON
MOVD - 10
DELAY 0.5
JUMP MAIN
END
```

Compile and save the program in program number ten.

## Monitor I/O Status

Display the I/O Status Monitor by selecting the Inputs/Outputs menu item from the Monitor menu. Display the Run Control dialog box and arrange these two dialogs so you can see both of them on your screen. To move a dialog, click in the title bar, hold down the mouse button and drag the window to where you want it. To use the keyboard, press ALT+F6 to select which window you wish to move, press ALT+Space and select Move. Use the arrow keys to move the window. When the window is where you want it, press Enter. Your screen should look something like this:



Select program ten and select the Run Program button from the Run Control dialog box. The program should turn O2 off and O1 on, move the motor forward ten revolutions, wait for half a second, turn O1 off and O2 on, move backward ten revolutions, wait for half a second and jump back to the beginning and repeat until the program is stopped. While the program is running, the O1 and O2 check boxes in the I/O Status Monitor should change state. Select the Stop button to stop the program.

TUTORIAL

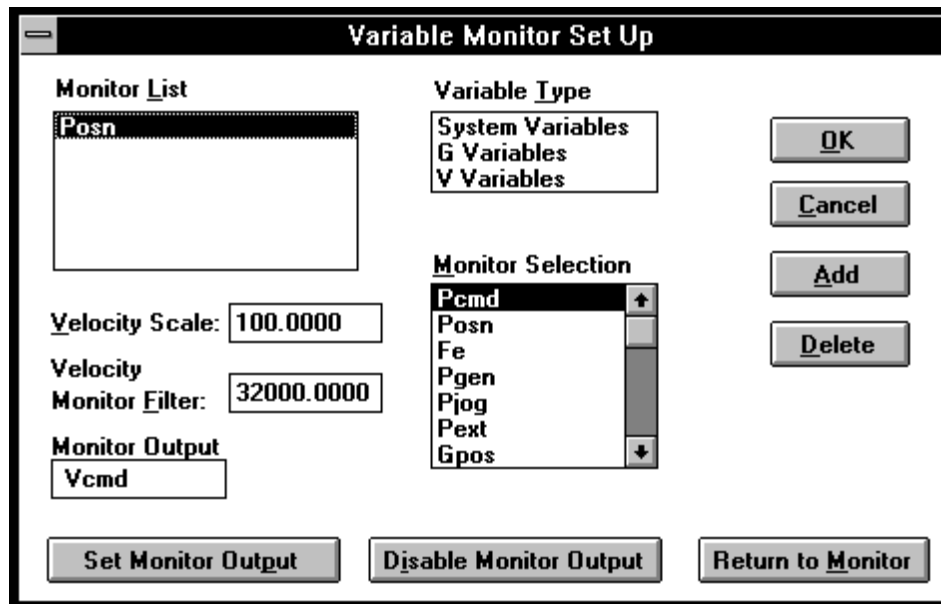
## Scanned Event to Implement a Programmable Limit Switch

Next we will use Scanned Events to implement a programmable limit switch. A programmable limit switch is a mechanism that can be programmed to turn on at a specific position. A Scanned Event is a small program that runs independently of the main program. We will set up a Scanned Event to check if the current position (system variable POSN) is greater than or equal to a given value (3) and less than or equal to another given value (6). If it is, O1 is turned ON. Two other Scanned Events check to see if POSN is less than the minimum value or greater than the maximum value. If it is, O1 is turned OFF. Open a new file and enter the text below. Compile and save the program to the ULTRA Plus or IQ.

```
TITLE "scanpgm";any text following a semicolon is ignored
;this is useful for putting in comments
S1: IF POSN >= 3 IF POSN <= 6 O1 = ON;Scanned Events
S2: IF POSN < 3 O1 = OFF
S3: IF POSN > 6 O1 = OFF

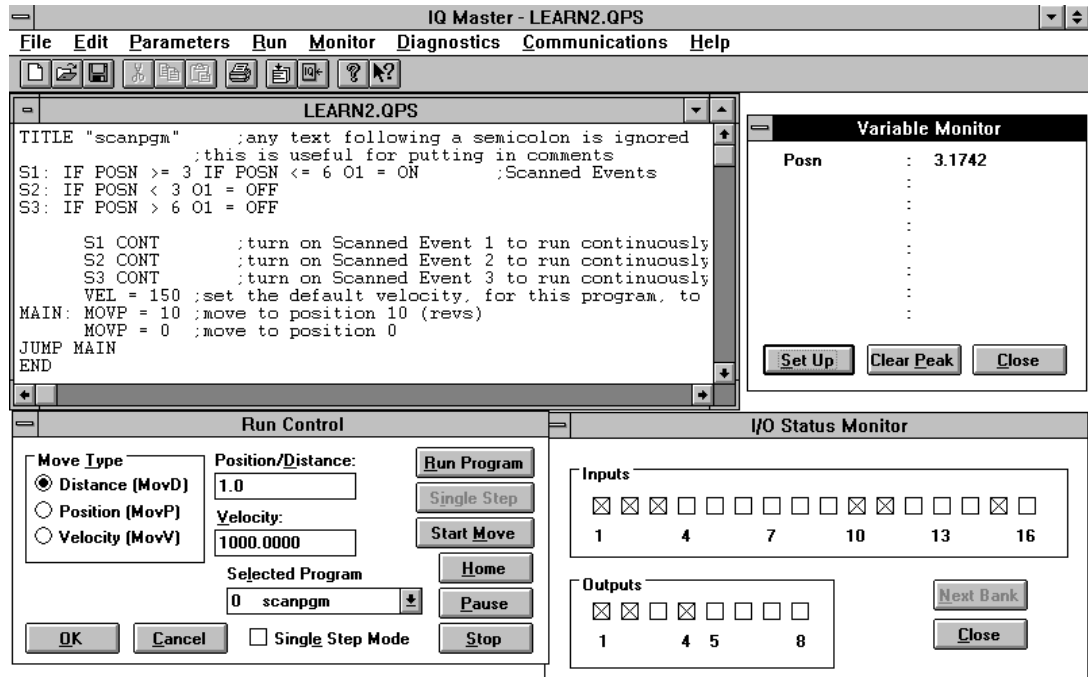
S1 CONT;turn on Scanned Event 1 to run continuously
S2 CONT;turn on Scanned Event 2 to run continuously
S3 CONT;turn on Scanned Event 3 to run continuously
VEL = 150;set the default velocity, for this program, to 150
MAIN: MOVP = 10;move to position 10 (revs)
      MOVP = 0;move to position 0
JUMP MAIN
END
```

To monitor the actual position from IQ Master, select the Variable Monitor Set Up menu item from the Monitor menu. This will display the following dialog box:



Select Posn from the Monitor Selection list and select the Add button. Select the OK button. Now select the Monitor Variables menu item. This will display a box that continuously updates POSN. Display the I/O Status Monitor and the Run Control dialog boxes as before and arrange the windows so that your

screen looks something like this:



Run the program you just entered. Notice that whenever the position (POSN) is between three and six, O1 is ON, otherwise it is OFF.

## Local/Run-time Values

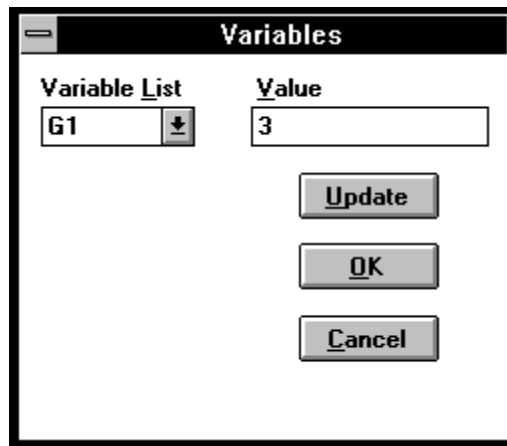
The VEL=150 statement in the previous program overrides the system default for velocity until it is set again in the program or the program ends. While this value is in effect, it will be the default velocity used by move commands that use the default velocity. A complete list of Local/Run-time variables can be found in part 5 of this manual.

## Variables

Programming the limits for the Programmable Limit Switch and compiling probably is not the most convenient way to “program” the limit switch. The ULTRA Plus or IQ has many variables that may be used in programs: G1..G64, V1..V64 number variables and F1..F64 and B1..B8 flag variables. The G and B variables retain their values even after power is removed from the ULTRA Plus or IQ. We will now use two variables to “program” the limits in the Programmable Limit Switch. Modify the Scanned Events in the last program like this:

```
S1: IF POSN >= G1 IF POSN <= G2 O1 = ON;Scanned Events
S2: IF POSN < G1 O1 = OFF
S3: IF POSN > G2 O1 = OFF
```

Notice that the minimum value, 3, has been replaced by G1 and the maximum value, 6, has been replaced by G2. Compile and save this program. Display the Variables dialog box from the Parameters menu (shown below). The value of G1 should be displayed. Set the value of G1 to 3 by typing 3 in the Value box. Select the Update button to accept this value. Select G2 from the Variable List and change it's value to 6. Select the OK button to accept this last value and close the dialog box.



Run the modified program. Does it run the same as the old one? Try changing the values of G1 and G2 by displaying the Variables dialog box again, even while the program is running. The behavior of the Programmable Limit Switch should change.

## IF/ELSE Statement, ASSIGN, Block Structure { }

Three scanned events are a lot of program code to control just one output. By rewriting the program, only one Scanned Event is needed.

It might be difficult to remember that G1 is the minimum value and G2 is the maximum value later on. The ASSIGN statement lets a programmer *assign* a meaningful name to a variable.

The new program looks like this:

```
TITLE "scanpgm2"
ASSIGN MindistG1 ;The min. position the axis can be at for INBAND to be on
ASSIGN MaxdistG2 ;The max. position the axis can be at for INBAND to be on
ASSIGN INBAND01 ;INBAND is output 1

S1:IF F1 = F1 {
  IF POSN < Mindist
    INBAND = OFF
  ELSE
```



```

IF POSN > Maxdist
    INBAND = OFF
ELSE
    INBAND = ON
}

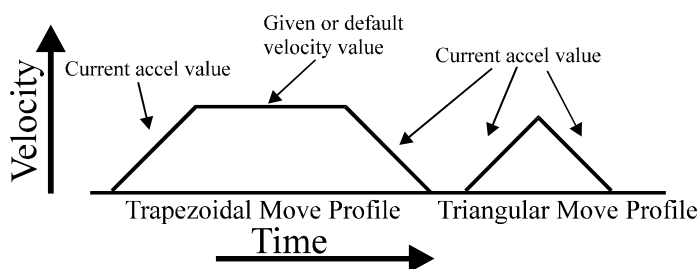
S1 CONT
MOVP = 0,V = 300
VEL = 150
MAIN: MOVD = 10
      MOVD = -9.1
      IF POSN < 7 JUMP MAIN
END

```

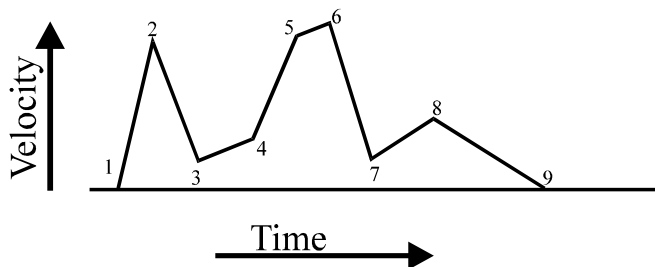
The first line of the Scanned Event is a condition that will always be true: IF F1=F1. This means the Scanned Event action will always be executed. The action, in this case, is actually a block of code. A block of code is signified by placing curly braces, {}, around the code and can be used anywhere a single statement can be used. The code inside the braces, two IF-ELSE statements, could be translated as: if POSN is less than G1, turn O1 OFF; otherwise (ELSE) check to see if POSN is greater than G2. If it is, turn O1 OFF, if it is not, turn O1 ON. Change the program and run it as before.

## Stick Moves

MOVD and MOVP commands generate what is called a trapezoidal motion profile: the motor is accelerated using the current acceleration setting to a given or default top speed and then decelerated at the right time to arrive at the desired position. If the distance to be moved is fairly small, a triangular move profile will be used: the motor will accelerate for half the move and decelerate for the other half of the move, but never reach the given or default velocity.



MOVD and MOVP commands are simple and useful, but if the required move profile is more complex than a simple trapezoid, DV (stick) moves can be used.



The profile shown above can be broken up into 8 DV moves. The first stick would define the distance between point 1 and point 2 and the velocity at point 2. So, if the distance between points 1 and 2 was 3 units and the velocity at point 2 was 56 RPM, the command would be: D=3, V=56. The second stick would give the distance between point 2 and point 3 and the velocity at point 3, and so on. Any profile

TUTORIAL

can be programmed using DV moves.

From the following table, write a program to execute the move profile shown, and run it.

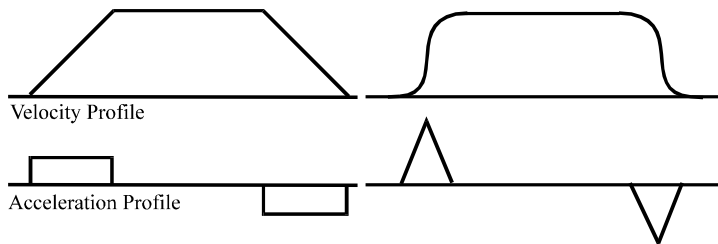
Stick Number	Distance moved during stick	Velocity at end of stick
1	3	56
2	3	12
3	4	16
4	2	57
5	2.5	61
6	3	11
7	5	20
8	8	0

The program for this profile should look something like this:

```
TITLE "stkmoves"
D = 3,V = 56
D = 3,V = 12
D = 4,V = 16
D = 2,V = 57
D = 2.5,V = 61
D = 3,V = 11
D = 5,V = 20
D = 8,V = 0
END
```

## S-Curve Acceleration

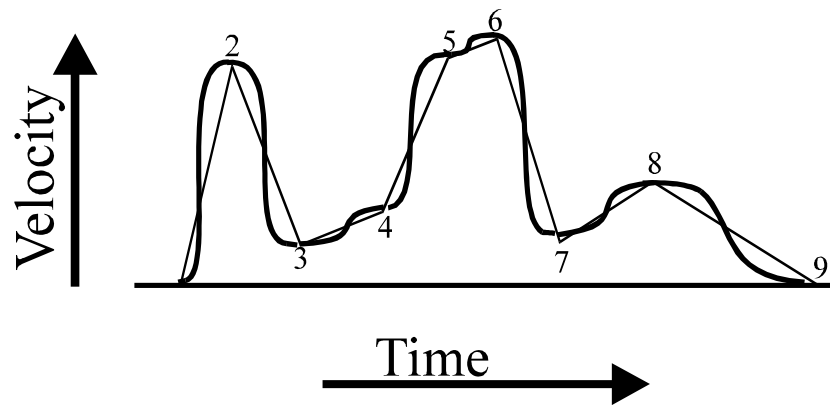
Very often it is important for a move profile to be as smooth as possible. This could be to minimize the wear on a machine, or it could be that a smooth profile is critical to the successful completion of an operation. To perform smooth motion profiles, the ULTRA Plus or IQ supports S-Curve acceleration. The following diagram illustrates the difference between straight and S-Curve acceleration.



With normal, straight line acceleration, the axis is accelerated to the target velocity in a linear fashion. S-Curve acceleration accelerates the axis slowly at first, twice as fast as the straight line acceleration in the middle, and then slowly stops accelerating. With straight line acceleration, the acceleration changes abruptly and then at the end of the acceleration period, changes abruptly again to no acceleration (see the Acceleration Profile). With S-Curve acceleration the acceleration gradually builds to the peak value then gradually decreases to no acceleration. The disadvantage with S-Curve acceleration is that for the same acceleration distance the peak acceleration is twice that of straight line acceleration—which often requires twice the peak torque. Notice that the axis arrives at the end of the stick position at the same

time regardless of which type of acceleration is used.

To use S-Curve acceleration in a D-V stick requires only the addition of an “,S” at the end of the statement. So the first stick in the previous example would be written  $D=3, V=56, S$ . Modify the previous motion profile to use S-Curve acceleration. Using S-Curve acceleration on all of the sticks of the previous example would yield a velocity profile something like this:



## Subroutines and Loops

Often it is necessary to repeat a series of steps in several places in a program. Subroutines can be useful in such situations. The syntax of a subroutine is simple. Subroutines must be placed after the main program, must start with `SUB subname` (where *subname* is the name of the subroutine), and must end with a return (`RET`). Note that there may be more than one `RET` statement in a subroutine. Subroutines are called using the `CALL` statement.

Loops are also supported and can be used for actions that need to be repeated more than once.

The following example shows a subroutine called `MDS` and two types of loops. The `MDS` subroutine implements a `MOVD` command using S-Curve acceleration to accelerate and decelerate. One loop uses the `LOOP` command, the other uses an `IF-JUMP` command and a label.

```
TITLE "MDSTEST"
ASSIGN dist V1
ASSIGN myloopcnt V10

myloopcnt = 0
main:
LOOP 20
dist = loopindex/2
CALL MDS
DWELL 0.1
RPT
dist = -Pcmd
CALL MDS
DELAY 2
myloopcnt = myloopcnt + 1
IF myloopcnt < 10 JUMP main

SUB MDS
;S-Curve MOVD subroutine
;This subroutine will do a MOVD of dist units using S-Curve acceleration and ;deceleration.
```

```

;This subroutine uses dist, V2 and V3
V2 = 500*VEL/TBase;calc. accel distance: (VEL/(TBase/500))^2/2*ACCEL)
V2 = V2*V2/(2*ACCEL)
IF dist < 0 V2 = -V2;If backward move, accel backward
;IF UTOC1(dist) > -10 IF UTOC1(dist)<10 {MOVD dist RETURN}
V3 = 2*V2
IF dist<0 {IF V3>dist JUMP nrmlmd};check for (+) moves
ELSE {IF V3<dist JUMP nrmlmd};check for (-) moves

V2 = dist/2;do a triangle move
V3 = VEL*(dist/V3)
D = V2,V = V3,S
D = V2,V = 0,S
RETURN

nrmlmd:          ;do a normal "trapezoidal" move
V3 = dist-V3;calc constant speed distance
D = v2,V = VEL
D = V3,V = VEL
D = V2,V = 0,S
RETURN
END

```

# User Variables and Arithmetic

---

## User Variables

IQ Basic supports a fixed set of variables that the programmer can use to store data and perform arithmetic. There are four types of user variables: G, V, B, and F. There are 64 nonvolatile G variables, 64 volatile V variables, 64 volatile F flag variables and 8 nonvolatile B flag variables. In addition to the user variables, system variables are also supported.

System variables are dedicated variables that contain particular values. For example, POS1 contains the position of encoder 1. System variables are discussed later in Part 4.

### Scope

IQ Basic variables are available system wide. Each of the variables can be read and set from any user program, system program, or by a Host Language Command at any time. There are no provisions to protect a variable from change. This is referred to as global scope.

Global scope variables permit you to segment your project into multiple programs, each with a particular task. The multiple programs then pass information through global variables. For example, one program may initialize the machine, prompt the operator through a series of data entries, check that the data is correct, and then set a flag indicating the setup program has been run successfully. Another program in the system may run through the steps of making a part. First it ensures that the setup program has been run by checking the flag. It then continues to make parts using the setup variables.

The disadvantage of global scope variables is that you must be careful to avoid inadvertent interaction between your programs. For example, if one program expects a variable to be a certain value and another program modifies the variable, the first program will not run correctly.

### Nonvolatile vs. Volatile Variables

Nonvolatile variables are variables that do not change when power is removed from the ULTRA Plus or IQ. G and B variables are nonvolatile variables. These variables maintain their values when power is removed. Nonvolatile variables are primarily used to store data that does not change frequently. For example, the maximum speed, the total number of cycles that have been run since installation, or part

setup data (all the variables that are used to make a particular part) could be nonvolatile variables.

Volatile variables are variables that are reset to a known state each time power is applied to the ULTRA Plus or IQ or at each Hard Reset. V and F variables are volatile. Volatile variables are useful if you want to know that power has been removed and to force the operator to do some initialization. In addition, volatile variables may be used for temporary variable use in calculations. At power-up or a Hard Reset, V variables are reset to zero and F flag variables are OFF.

## Resolution and Accuracy

B and F variables are flag variables. They have only two states: ON or OFF. Flags are used in logical expressions and as conditions in conditional expressions. Flags are often used to indicate that some event has occurred or that the program has executed to a particular point. For example, if you want some instructions to execute only when power is first applied you might do the following:

```
IF F1 OFF                ;F flags are OFF at Power-up
  {
    ... Statements      ;Whatever code you want
    F1 = ON              ;Indicate code ran once
  }                      ;End of block
```

G and V variables are stored internally as 4 bytes for the mantissa (the part before the decimal point), 2 bytes for a numerator (the top part of a fraction), and 2 bytes for a denominator (the bottom part of a fraction). The range of numbers that can be represented in 4 bytes is  $\pm 2,147,483,648$ . The range of numbers that can be represented in 2 bytes is  $\pm 32,767$ . Therefore, the range of numbers that can be contained in a G or V variable and any intermediate value is:

$$G \text{ or } V = \pm 2,147,483,648 + \frac{\pm 32,767}{\pm 32,767}$$

Numbers that result from division are stored in the numerator and denominator. For example:

```
G1 = 2/3;The numerator of G1 contains 2 and the denominator 3
```

As a result of this type of storage, calculations such as (2/3) are not rounded off (0.6666667) and maintain their full precision. If the result (or intermediate result) of a calculation has a denominator larger than 32,767, the values are converted to the closest fraction with a denominator of 10,000. Subsequent calculations will then appear to be standard arithmetic with 4 decimal places of accuracy.

## Variable Characteristics Summary

Name	Qty	Type	Power-Up	Resolution
G	64	Value	Nonvolatile	4 byte mantissa + 2 byte numerator & 2 byte denominator -or- 0.0001
V	64	Value	Volatile	4 byte mantissa + 2 byte numerator & 2 byte denominator -or- 0.0001
B	8	Flag	Nonvolatile	ON or OFF
F	64	Flag	Volatile	ON or OFF

## Arithmetic

Four function arithmetic, addition (+), subtraction (-), multiplication (\*), and division (/) is supported by IQ Basic. The syntax for arithmetic is:

Syntax	<i>result = operand operator operand [operator operand] ...[operator operand]</i>	
result	The <i>result</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or certain system variables	
operand	The <i>operand</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , a system variable, or a constant	
operator	+, -, *, /	

### Precedence and Order of Evaluation

The following table summarizes the rules for precedence for all operators. Operators on the same line have the same precedence; rows are in order of decreasing precedence. For example, \* and / have the same precedence, and have higher precedence than + and -.

Operators
( ), @, UTOC1, UTOC2, CTOU1, CTOU2
*, /
+, -
NOT, AND, OR, XOR, NAND and NOR
=

Examples:

```
G1 = 3 + 4 * 2;G1 set equal to 11
G1 = (3 + 4) * 2;G1 set equal to 14
G1 = 3 + 4 * 2 + 4;G1 set equal to 15
```

## Boolean Operators

The following Boolean operators are supported: AND, NAND, OR, NOR, XOR and NOT. The operands that are used with these operators include F flags, B flags, system flags and Inputs. The result can be an F flag, B flag, system flag (except read-only flag) or output.

### AND

F3 = F1 AND F2;F3 is ON if F1 and F2 are ON; otherwise, F3 is OFF

F1	F2	F1 AND F2
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

**NAND**

F3 = F1 NAND F2; F3 is OFF if F1 and F2 are ON; otherwise, F3 is ON

F1	F2	F1 NAND F2
OFF	OFF	ON
OFF	ON	ON
ON	OFF	ON
ON	ON	OFF

**OR**

F3 = F1 OR F2; F3 is ON if either F1 or F2 are ON; otherwise F3 is OFF

F1	F2	F1 OR F2
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON

**NOR**

F3 = F1 NOR F2; F3 is OFF if either F1 or F2 are ON; otherwise F3 is ON

F1	F2	F1 NOR F2
OFF	OFF	ON
OFF	ON	OFF
ON	OFF	OFF
ON	ON	OFF

**XOR**

F3 = F1 XOR F2; F3 is ON if F1 and F2 are different  
; F3 is OFF if F1 and F2 are the same

F1	F2	F1 XOR F2
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	OFF

**NOT**

F3 = NOT F1; F3 is ON if F1 is OFF; F3 is OFF if F1 is ON

F1	F1 NOT F2
OFF	ON
ON	OFF

**Examples**

F2 = F4 AND I11

F2 = F4 OR I11

F10 = F1 XOR F12

F7 = ATHOME NAND F3

F7 = I9 NOR F11

O1 = I7 AND I8 AND NOT I9

O1 = NOT F3



## Conversions

Position can be represented in encoder counts or as user units using a scale factor. Encoder counts are the number of encoder counts observed as the motor moves. The IQ counts each edge of an encoder pulse, therefore, the number of encoder lines is multiplied by 4 for one revolution. For a 2,000 line encoder, there are 8,000 encoder counts per motor revolution. If the motor started at zero and moved one revolution, the new position would be 8,000 counts.

To make the IQ easier to use, a user scale factor is used to convert encoder counts to user units. With the SCALE parameter, set in the System dialog box under the Parameter menu, you tell the IQ how many encoder counts there are in the unit of measure that you would like to use. The SCALE argument must be an integer. For example, if you want to work in units of revolutions, set the SCALE parameter to how many encoder counts in one revolution. Once the SCALE is set, all distances or positions will be specified or read in revolutions. Throughout this text, scaled units are referred to as user units: the units specified by the user of the IQ.

Occasionally the need arises to convert between encoder counts and user units. To do this, several functions are available in IQ Basic: @, UTOC1, UTOC2, CTOU1, CTOU2.

Placing an “at” sign, @, in front of a position system variable such as POS1 will cause that value to be read in encoder counts rather than in user units. UTOC1 (UTOC2) will convert from user units to encoder counts using SCALE (SCALE2). CTOU1 (CTOU2) will convert counts to user units using SCALE (SCALE2)

For example:

```
V1 = POS1           ;V1 contains position 1 in user units
V2 = @POS1         ;V2 contains position 1 in encoder counts
V3 = UTOC1 V1     ;V3 contains position 1 in encoder counts
V4 = CTOU1 V2     ;V4 contains position 1 in user units
```

### Encoder Counter Rollover

Many IQ applications require reading encoder positions often as the machine operates. If the application is unidirectional (the net motion is always in one direction), eventually the IQ encoder counter will roll over from a large positive value to a large negative value (or from a large negative value to a large positive value). The IQ encoder counters store encoder positions as a 32 bit signed value. When the counter value reaches  $2^{31}$  (2,147,483,648) counts, the next encoder pulse will cause the value to roll over to  $-2^{31}$  (-2,147,483,647). If math is performed on two different position samples, the calculated result may be wrong if one position sample is read just before rollover occurs, and the second is read just after rollover occurs. To avoid this problem, the conversion functions may be used to perform math on position values in encoder counts rather than user units. Performing math in encoder counts avoids this problem because IQ variables store values in 32 bit signed values just like the encoder counters. When the math is performed, the math itself “rolls over” just like the encoder counter, and the result will be correct.

Example: The SCALE parameter is set to 8000 counts/unit. Variable V1 reads POS1 (with the statement `V1 = POS1`) in user units. The value of POS1 is 268,435 (2,147,480,000 counts). Variable V2 reads POS1 (with the statement `V2 = POS1`) in user units *after* the counter has rolled over when the value of POS1 is -268,432 (-2,147,456,000 counts). If the difference in position between the two different instances is desired, the two values may be subtracted. To do this, V3 is calculated as `V2 - V1`. In this example, since V1 and V2 were read in user units instead of counts, the value calculated in V3 would be  $-268,432 - 268,435 = -536,867$ . This very large negative value is not the correct difference. The solution is to capture the positions into V1 and V2 in counts using the statements `V1 = @POS1` and `V2 = @POS1`. Now, V1 will contain 2,147,480,000 and V2 will contain -2,147,456,000. When these values are subtracted (`V3 = V2 - V1`), the 32 bit math will properly set V3 to a value of 31,295. This is the correct value in counts of the position difference. V3 can now be converted to user units using the statement `V3 = CTOU1 V3` which would convert V3 to a value of 3.9119 units which is the desired position difference.

Another important consideration in correcting for encoder counter rollover is when a program compares

encoder positions to a value, for example, `WAIT POS1 > V4`. The comparison statement `WAIT POS1 > V4` will not properly account for rollover. It should be written so that a direct 32 bit math function is performed on values in encoder counts before the comparison is done.

For example:

```
WAIT POS1 > V4
```

could be rewritten as:

```
V5 = UTOC1 V5  
WAIT (@POS1 - V5) > 0
```

## Summary of User Variables and Arithmetic

The following table contains a summary of the instructions, flags, and variables that relate to user variables and arithmetic. Refer to Part 5 • Language Reference for more detailed information.

Type	Name	Description
Boolean	AND	Logical AND
	NAND	Logical NAND
	NOR	Logical NOR
	NOT	Logical NOT
	OR	Logical OR
	XOR	Logical exclusive OR
Conversion	@	Store position value as encoder counts
	UTOC1	Convert user units to counts using SCALE
	UTOC2	Convert user units to counts using SCALE2
	CTOU1	Convert counts to user units using SCALE
	CTOU2	Convert counts to user units using SCALE2
Miscellaneous	INT	Returns the integer portion of a variable
Variable	$B_n$	Nonvolatile flag variable, $n = 1, 2, \dots, 8$
	$F_n$	Volatile flag variable, $n = 1, 2, \dots, 64$
	$G_n$	Nonvolatile variable, $n = 1, 2, \dots, 64$
	$V_n$	Volatile variable, $n = 1, 2, \dots, 64$

# System Variables & Flags

---

## Introduction

### What is a System Flag?

Flags that have a predefined function are referred to as system flags. These flags represent conditions in the ULTRA Plus or IQ. The flags can be tested and some can be set in your program. For example, the ATHOME flag can be tested to see if the motor is at the home position.

### What is a System Variable?

System variables are variables that have a predefined meaning. In most cases the value of these variables can be read and set in your program. Many of these variables can be set in the Parameter menu. The values that are set in the Parameter menu become the system default values. Though they can be changed in the program, when the program ends, the value is returned to the default value.

For example, velocity can be set in the Parameter menu. If you do not set a velocity in your program, the value set in the Parameter menu will be used for your MOVD and MOVVP moves. If you change the value in your program by executing a VEL command, that value is only in effect while that program is running. After the program ends the velocity will be restored to the value set in the Parameter menu.

### The Personality Module

The personality module contains data (after it is configured) that matches the ULTRA Plus or IQ to a particular motor. System variables and flags that are set in the Parameter menu are also stored in the personality module along with all programs and nonvolatile variables and flags. System variables and flags that are not stored in the personality module are volatile and are calculated or set/cleared during run time.

**Note:** When you change the value of a system variable or flag (that is stored in the personality module) within an application program, the change remains in effect only until the program ends. When the program ends, the value is returned to the value in the personality module. The application program only changes a run-time copy of the system variable or flag.

## Variable and Flag Summary

Below is a table that summarizes the system variables and flags, sorted by type. The Flag/Variable column categorizes each as a flag or a variable. The Read/Write column indicates which can be written (W) and which can be read (R). If the variable or flag is stored in the personality module it is marked with a “Y” in the personality column. In addition to the system variables listed below, there are other variables associated with IQ Cam. Refer to the *IQ-Series IQ-Cam Software Manual* (Part Number 0013-1053-001) for complete details.

Type	Name	Description	Flag/ Variable	Read/ Write	Personality Module
Accel/ Velocity	ACCEL	Acceleration used to calculate MOVP and MOVD moves	V	R/W	Y
	FVEL1	Filtered motor velocity	V	R	N
	JACCEL	Jog acceleration	V	R/W	Y
	JDECEL	Jog deceleration	V	R/W	Y
	JVEL	Jog velocity	V	R/W	Y
	PVEL1	Peak velocity in user units per Time-Base	V	R	N
	SLEWEN	Enables the slew rate limit for the gear output	F	R/W	Y
	VCMD	Velocity command in user units per TimeBase	V	R	N
	VEL	Velocity used for MOVP and MOVD moves	V	R/W	Y
	VEL1	Velocity feedback in user units per TimeBase	V	R	N
VEL2	Encoder 2 velocity in user units per TimeBase	V	R	N	
Current	IAVE	Average current command in Amperes	V	R	N
	IAVG	Average current trip point in Amperes	V	R/W	Y
	ICMD	Current command in Amperes	V	R	N
	ILIMIT	Peak current limit in Amperes	V	R/W	Y
	PICMD	Peak current command in Amperes	V	R	N
Gain	FGAIN	Acceleration feedforward gain for velocity loop	V	R/W	Y
	FILTER	Filter value	V	R/W	Y
	IGAIN	Integral gain for the velocity loop	V	R/W	Y
	IZONE	Region where KI has effect	V	R/W	Y
	KFF	Velocity feedforward gain	V	R/W	Y
	KI	Integral gain for the position loop	V	R/W	Y
	KP	Proportional gain for the position loop	V	R/W	Y
KPZ	Proportional gain used within the PZONE	V	R/W	Y	

Type	Name	Description	Flag/ Variable	Read/ Write	Personality Module
Gain (cont.)	PGAIN	Proportional gain for the velocity loop	V	R/W	Y
	PZONE	Region where KPZ is used instead of KP	V	R/W	Y
Home	ATHOME	At home used in Absolute mode only	F	R	N
	HOFFS	Home offset	V	R/W	Y
	HOMECMD	Runs Home program when set	F	R/W	N
	HSEQCPL	Home sequence complete	F	R	N
	HSWEN	Set if Home Switch input enabled	F	R	Y
	HSWSTAT	Current status of Home Switch input (regardless if active open or closed)	F	R	N
	HVEL	Home velocity	V	R/W	Y
Input and Outputs	ADC1	Analog input 1 in volts	V	R	N
	ADC2	Analog input 2 in volts	V	R	N
	ADC3	Analog input 3 in volts	V	R	N
	ADC4	Analog input 4 in volts	V	R	N
	ADC5	Analog input 5 in volts	V	R	N
	DAC1	Analog output 1 in volts	V	W	N
Interrupt	FI1	Interrupt 1 flag	F	R/W	N
	FI2	Interrupt 2 flag	F	R/W	N
	FIDX1	Encoder 1 index interrupt flag	F	R/W	N
	FIDX2	Encoder 2 index interrupt flag	F	R/W	N
	I1P1	Position 1 latched by interrupt 1 in user units	V	R	N
	I1P2	Position 2 latched by interrupt 1 in user units	V	R	N
	I2P1	Position 1 latched by interrupt 2 in user units	V	R	N
	I2P2	Position 2 latched by interrupt 2 in user units	V	R	N
	IDX1	Enable flag for Encoder 1 index interrupt	F	R/W	N
	IDX2	Enable flag for Encoder 2 index interrupt	F	R/W	N
	IX1P1	Position 1 latched by Encoder 1 index interrupt	V	R	N
	IX1P2	Position 2 latched by Encoder 1 index interrupt	V	R	N
	IX2P1	Position 1 latched by Encoder 2 index interrupt	V	R	N

Type	Name	Description	Flag/ Variable	Read/ Write	Personality Module
Interrupt (cont.)	IX2P2	Position 2 latched by Encoder 2 index interrupt	V	R	N
	LPOS	Hardware latched position in user units	V	R	N
Motion	GEAR	Gearing ratio	V	R/W	N
	GEAREN	Enables gearing	F	R/W	N
	SLEW	Limit for output of gear	V	R/W	Y
Operator Terminal	FNACTIVE	Fkey active flag, indicates that an Fkey is pressed	F	R	N
	XNACTIVE	Xkey active flag, indicates that an Xkey is pressed	F	R	N
Position	ERETPOS	Ereturn position	V	R/W	Y
	FE	Following error in user units	V	R	N
	GPOS	Position at output of GEAR (before SLEW)	V	R	N
	PCAM	Cam position command	V	R	N
	PCMD	Position command in user units	V	R	N
	PEXT	External position command (from gear) in user units	V	R	N
	PFE	Peak following error in user units	V	R	N
	PGEN	Position command from internal profile generator	V	R	N
	PJOG	Position command from jogs	V	R	N
	POS1	Encoder 1 position in user units	V	R	N
	POS2	Encoder 2 position in user units (scaled by Scale2 parameter)	V	R	N
	POS3	Position 3 (from option card)	V	R	N
	POSN	Position feedback in user units (scaled by Scale parameter)	V	R	N
System	ABORT	Stops program execution (but not current motion)	F	R/W	N
	ABSMODE	Flag to indicate if absolute mode is active	F	R	Y
	ACTIVESN	Variable to indicate number of active scanned events	V	R	N
	ACTIVEXN	Variable to indicate number of active Xkey routines	V	R	N
	CLRPEAKS	Clears peak current, motor velocity, and following error	F	W	N
	CONFIG	Selects feedback configuration	V	R/W	Y

Type	Name	Description	Flag/ Variable	Read/ Write	Personality Module
System (cont.)	DHCMD	Defines the present position as the home (zero) position	F	R/W	N
	DISLIM	Temporarily disables hardware limits	F	R/W	N
	EFLAG	Flag to indicate if an error is active	F	R	N
	ENUM	Error number	V	R	N
	FDR	Feedrate value	V	R/W	N
	FEL	Following Error Limit	V	R/W	Y
	FET	Following Error Time	V	R/W	Y
	FNPGM	Start an Fkey program	V	R/W	N
	FNVAR1	Fn routine volatile variable 1	V	R/W	N
	FNVAR2	Fn routine volatile variable 2	V	R/W	N
	FOUNDHOME	Used by the home program to tell system that the home position was found	F	R/W	N
	HLIMITS	Set if limit inputs enabled	F	R	Y
	HRESET	Causes a hardware reset	F	W	N
	INDEXEN	Used to tell the home program to use the encoder index	F	R	Y
	INPOSN	In-position	F	R	N
	JOGACTIVE	Job motion is active	F	R	N
	JOGF	Simulates pressing the forward jog input	F	R/W	N
	JOGR	Simulates pressing the reverse jog input	F	R/W	N
	LOOPINDEX	The LOOP counter for the LOOP instruction	V	R/W	N
	MONOUT	The number of the currently active variable for the monitor analog output	V	R/W	Y
	MOVECOMPLETE	Profile generated motion is active	F	R	N
	MVOEN	Monitor Velocity Output enabled	F	R/W	N
	OTMON	Display monitor variables on the top line	V	R/W	N
	OVERSPEED	Set the overspeed fault trip point	V	R/W	Y
	PAUSE	Simulates pressing the pause input	F	R/W	N
	PCMACTIVE	Flag to indicate PCam position is changing	F	R	N
	PEXTACTIVE	Flag to indicate Pext position is changing	F	R	N
PGMNUM	Specifies which program will execute when STARTP is executed	V	R/W	N	
PGMRUNNING	Indicates a main program is running	F	R	N	



Type	Name	Description	Flag/ Variable	Read/ Write	Personality Module
System (cont.)	PURGEMOTION	Removes motion from queue immediately	V	R/W	N
	PURGEMOTION- ACTIVE	Indicates motion has been purged from queue	F	R	N
	RFDR	Current run time feedrate value in percent	V	R	N
	ROT	Indicates the positive direction of rotation for encoder 1	F	R	Y
	ROT2	Indicates the positive direction of rotation for encoder 2	F	R	Y
	ROTT	Direction of rotation for the Tracking function	F	R/W	N
	SCALE	Encoder counts to user unit conversion	V	R/W	Y
	SCALE2	Encoder 2 counts to user unit conversion	V	R/W	Y
	SEN	Software enable	F	R/W	N
	SRESET	Causes a soft reset	F	R/W	N
	STARTP	Start program flag	F	R/W	N
	STOP	Stops all motion and programs	F	R/W	N
	SYSERROR	System error	V	R/W	N
	SYSWARN	System warning	V	R/W	N
	TBASE	Time base. 500 indicates seconds, 30000 indicates minutes.	V	R	Y
	TIMER1	Timer 1	V	R/W	N
	TIMER2	Timer 2	V	R/W	N
	TRACKINGEXT- POSSELECT	Selects if the Tracking function occurs before or after Gear and Slew.	F	R/W	N
	TRACKINGMODE	Sets the mode for the Tracking function.	V	R/W	N
	TRACKINGYNCED	Indicates the tracking function is not disabling follower motion.	F	R	N
	WFLAG	Flag to indicate if a warning is active	F	R	N
	WINDOW	In-Position window	V	R/W	Y
	WNUM	Currently active warning number	V	R/W	N
	XLOOPINDEX	Counter for the LOOP instruction used in the Xkey program	V	R/W	N
	XNPGM	Simulates pressing an Xkey	V	R/W	N



# IQ Programming Structure

---

## Introduction

One of the most important aspects of programming is developing a structure for the program. Before you begin to write a program you should develop a plan for that program. What tasks must be performed? In what order do they need to be performed? What things can be done to make the program easy to understand and to be maintained by others? Are there any procedures that are repetitive? The answers to these questions and the programming techniques presented in this section will help you to develop a bug free program that can be maintained and enhanced in the future.

Most programs are not a simple linear list of instructions where every instruction is executed in exactly the same order each time the program is run. Programs need to do different things in response to external events and operator input. IQ Basic contains Program Structure instruction, scanned event functions, Xkey functions, Fkey functions, and interrupt functions that may be used to control the flow of execution in an application program.

Program Structure instructions are instructions that cause the program to change the path of execution. Scanned events are instructions that execute at the same time as the main body of the application program. Xkey and Fkey routines are routines that execute in response to an operator pressing keys on the operator terminal. Interrupts are instructions that execute quickly in response to select inputs to the IQ.

## Program Structure

IQ programs are divided into five distinct sections: Title, Header, Main Body, Subroutines, and End. These sections must be in the order outlined below.

### *Title*

TITLE "*example*";Title of the program

### *Header*

PGMTYPE = ...;Compiler Options statements

ASSIGN *name Vn*;Variable name assignments

S1:IF condition action;Scanned events

```

X1::Xkey programs
... statements;Any IQ Basic instructions
XEND

```

*Main Body*

```

... statements;Any IQ Basic instructions
;End of main body of program

```

*Subroutines*

```

SUB SUB1
... statements;Any IQ Basic instructions
RET

```

*End of program*

```

END

```

The Title section of the program consists of a single TITLE instruction, which designates a name up to 8 characters long. The program title is displayed in the directory. The TITLE statement is not required, but a warning will be generated if a program without a title is compiled.

The Header section of the program contains compiler options statements, assignments of names to variables used in the program, plus scanned events and operator terminal Xkey routines. If none of these are used in a program, the header section is not required.

The Main Body of the program contains the main part of the program, which can include all motion and math statements, labels, I/O commands, and subroutine calls.

Subroutines are routines that are called from the body of the program. When a subroutine is called, program execution is transferred to the subroutine until a RETURN statement occurs. Then, program execution returns to the body of the program following the CALL statement.

The END statement signifies the end of the program.

Comments are allowed in any section of the program and are preceded by a semicolon. They may occur on the same line as an instruction, or on a line that contains only the comment. Any text following a semicolon on a line will be ignored by the compiler.

---

## Program Structure Instructions

Program Structure instructions are used to control the course of the program within the main body, header, and subroutine portions of application programs and Fkey programs. The DO/WHILE, WHILE, IF, and ON instructions are used to create conditional execution of instructions. Subroutines are used to simplify and better organize your application program. The WAIT instruction is used to suspend the execution of the application program.

### DO/WHILE Structure

This instruction is used to execute a block of code one time and then continue executing that block until a condition is satisfied. The difference between the DO/WHILE instruction and the WHILE instruction is that the DO/WHILE instruction tests the condition after the block is executed so the conditional statements are always executed at least one time. The syntax for the DO/WHILE instruction is:

```

DO
{
... conditional statements

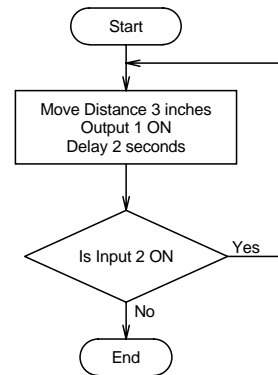
```

} WHILE *condition*

Refer to Part 5 • Language Reference for more detailed information. The following flowchart and code segment illustrate the use of the DO/WHILE instruction.

```

... statements
DO
{
MOVD = 3
O1 = ON
DELAY = 2
} WHILE (I2 = ON)
... statements
    
```



### WHILE Structure

This instruction is used if you want a block of code to execute while a condition is true. The difference between the DO/WHILE instruction and the WHILE instruction is the WHILE instruction tests the condition before any of the conditional statements are executed. If the condition is false when first tested, the conditional statements are never executed. The syntax for the WHILE instruction is:

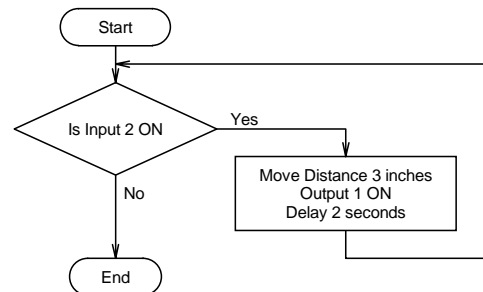
```

WHILE condition
{
... conditional statements
}
    
```

Refer to Part 5 • Language Reference for more detailed information. The following flowchart and code segment illustrate the use of the WHILE instruction.

```

... statements
WHILE (I2 = ON)
{
MOVD = 3
O1 = ON
DELAY 2
}
... statements
    
```



### Subroutines

A subroutine is a group of IQ Basic instructions that is located at the end of the main body of the program. It is marked by the SUB instruction at the top and a RETURN instruction at the end. The subroutine is executed by using a CALL instruction in the main body of the program.

Subroutines are used if there is a procedure that needs to be done in several places in the program. Rather than repeat those instructions at each location, a CALL to a subroutine that contains those instructions may be used.

TUTORIAL

When a CALL statement is executed, execution is transferred to the first line of the subroutine. The subroutine is executed until a RETURN statement is executed. When the RETURN is executed, program execution returns to the program line in the main program following the CALL. Subroutines may contain more than one RETURN statement.

Subroutines may not be nested. Only the main body of the program may contain a CALL to a subroutine. Refer to Part 5 • Language Reference for more detailed information on the SUB, CALL and RETURN statements. The following flowchart and code segment illustrate the use of subroutines.

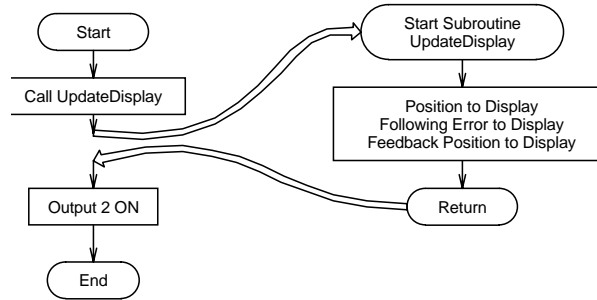
```

... statements

CALL UpdateDisplay
O2 = ON

... statements

SUB UpdateDisplay
V1 = POS1
V2 = FE
V3 = POSN
PRINT 1,1, V1
PRINT 2,1, V2
PRINT 3,1, V3
RETURN
    
```



## IF Structure

The IF statement is used to execute an instruction or a block of instructions one time if a condition is true. The simplified syntax for the IF instruction is:

```

IF condition
{
... conditional statements
}
    
```

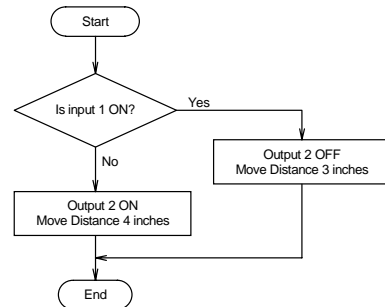
Refer to Part 5 • Language Reference for more detailed information. The following flowchart and code segment illustrate the use of the IF instruction.

```

... statements

If I1 = ON
{
O2 = OFF
MOVD = 3
}

... statements
    
```



TUTORIAL

## IF/ELSE Structure

The IF/ELSE statement is used to execute an instruction or a block of instructions one time if a condition is true and a different instruction or block of instructions if the condition is false. The simplified syntax for the IF/ELSE instruction is:

IF condition

```
{
  ... conditional statements
}
```

ELSE

```
{
  ... conditional statements
}
```

Refer to Part 5 • Language Reference for more detailed information. The following flowchart and code segment illustrate the use of the IF/ELSE instruction.

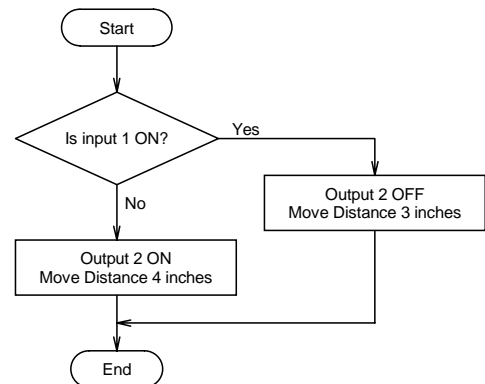
```

... statements

If I1 = ON
{
  O2 = OFF
  MOVD = 3
}
ELSE
{
  O2 = ON
  MOVD = 4
}

... statements

```



## WAIT Statement

The WAIT statement is used to suspend program execution until a condition is true. The simplified syntax for the WAIT statement is:

WAIT *condition*

Refer to Part 5 • Language Reference for more detailed information.

## ON Structure

The ON instruction is used for multiple dimensional branching. The program can branch to as many different labels as required based on the value of a variable. Refer to Part 5 • Language Reference for more detailed information.

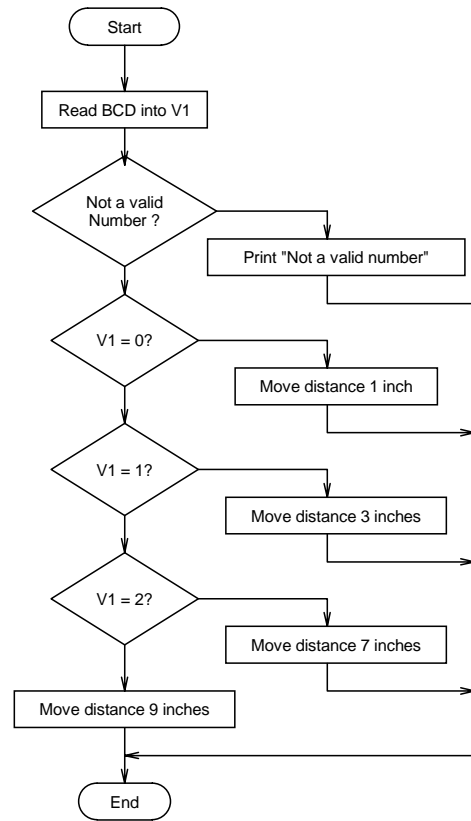
The following flowchart and code segment illustrate the use of the ON instruction.

```

... statements

BCD V1 = I12, 2
ON V1 JUMP MOVE0, MOVE1, MOVE2, MOVE3
BAD_V1:;number < 0 or > 3
    PRINT 1,1 "Not a valid Num."
    JUMP ENDSELECT
MOVE0:
    MOVD = 1
    JUMP ENDSELECT
MOVE1:
    MOVD = 3
    JUMP ENDSELECT
MOVE2:
    MOVD = 7
    JUMP ENDSELECT
MOVE3:
    MOVD = 9
ENDSELECT:

... statements
    
```



### JUMP / Label

The JUMP instruction can be used to transfer program execution to a new point marked by a label. This instruction is often used as the action of an IF statement. The destination label may be above or below the JUMP statement in the application program.

Labels may be any alphanumeric string up to 32 characters in length beginning with a letter and followed by a colon “:”.

```

JUMP TestInputs
...statements

TestInputs:
...statements
    
```

```

IF (I1 = ON) JUMP TestInputs
    
```

Refer to Part 5 • Language Reference for more detailed information.

### Program Structure Instruction Summary

The following table contains a summary of instructions that relate to program branching. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
CALL	Call a subroutine
DO/WHILE	Do once and keep doing while a condition is true
IF & IF/ELSE	Execute if a condition is true

TUTORIAL



Name	Description
ON <i>variable label1, label2,...</i>	Multi-dimensional branch
RETURN	Return from a subroutine
SUB	Define a subroutine
WAIT	Wait till a condition is true
WHILE	Execute while a condition is true

## Scanned Event (Sn) Instructions

Scanned event instructions establish conditions that are scanned on a regular basis. Once established, the scanned event can be enable and disable in the main body of the application program. If the condition is true, and the scanned event is enabled, when scanned, the specified action is executed.

Scanned events are used to record events and perform actions independent of the main body of the program. For example, if you want an output to come ON when the position is greater than 4 inches, or if you need to turn an output ON whenever input 13 and 14 are ON, you may use the following scanned event statements.

```
S1: IF POS1 > 4 O3 = ON
S2: IF (I13 AND I14 = ON) O4 = ON
```

Scanned events may also be used with timers to time an event or perform an action after an event and specified time delay.

The program statements contained in the action portion of a scanned event instruction can be any legal program statement except subroutine calls, DO/WHILE, WHILE, WAIT, CLR, PRINT, READ, LOOP, JUMP or motion statements. Motion statements include MOVD, MOVP, DV, DIF, DELAY, and DWELL.

Scanned event routines must be placed before the main body of a program and after any ASSIGN or TITLE statements.

**Note:** Each program may contain up to 8 scanned events. The scan time of a scanned event is not fixed as it is in a programmable logic controller. The frequency of the scanned event is influenced by other factors such as printing to the operator terminal, the number of scanned events, etc. Scan time is 2 milliseconds minimum.

### Sn:IF condition action

The scanned event IF statement is used in the same way as a normal IF statement, but is preceded by *Sn:*, where *n* is a number from 1 to 8. The condition may be any of the conditions described for the IF statement. If the scanned event is enabled, the condition is checked during program execution, if the condition is found to be true, the specified action is taken. Scanned events are enabled in the program by *Sn ON/OFF/CONT* statements.

If enabled with *Sn CONT* the scanned event will be executed every time the scanned event is tested and the *condition* is found to be TRUE. If the keyword EDGE is used with a scanned event testing inputs, the *condition* is only found to be TRUE when there is a transition on the input.

If enabled with *Sn ON* the scanned event will execute the first time the *condition* is TRUE. If the keyword EDGE is used with a scanned event testing inputs, the *condition* is only found to be TRUE when there is a transition on the input.

```
S1: IF I13 EDGE ON V11 = PCMD
S2: IF POS1 > = G10 O4 = ON
```

### ***Sn:TMRm condition action***

A timer (specified by TMR $m$  where  $m$  is 1 or 2) may be set to start based on a specified condition and run for the time loaded during program execution. Once the programmed time has elapsed, the specified action is executed. The timer scanned event is enabled and disabled by  $S_n$  ON/OFF/CONT commands

```
S2:TMR1 I11 = ON F7 = ON
S4:TMR2 POS2 < G3 V1 = V1+1
```

### ***Sn:TMRm start condition, stop condition action***

A scanned event timer may also be used to determine the elapsed time between two events. In this case two conditions are specified. The timer starts when the first condition is satisfied and stops when the second condition is satisfied. An action such as setting a flag or turning ON an output is then taken. The elapsed time between the two events is then available in the TIMER $m$  variable. The timer scanned event can be enabled or disabled by using the  $S_n$  ON/OFF/CONT statement.

```
S1: TMR1 POS2 > G1, POS2 > G2 F3 = ON
S2: TMR1 POSN > G10, POS2 > G5 {O1 = ON O2 = OFF}
```

### ***Sn ON/OFF/CONT***

This statement is used to enable or disable a scanned event. The  $S_n$  CONT statement will allow a scanned event to be tested continuously, without waiting for the program to re-enable the event.

```
S1 ON
S2 CONT
S1 OFF
```

### **Timer**

There are two timers available for use with scanned events. TIMER1 and TIMER2. These timers can be read to determine the time between two events or they can be set to establish a time delay between a trigger event and an action.

### **Scanned Event Instructions Summary**

The following table contains a summary of instructions that relate to scanned events. Refer to Part 5 • Language Reference for more detailed information.

<b>Name</b>	<b>Description</b>
<i>Sn ON/OFF/CONT</i>	Enable, disable, and continuously enable scanned events
<i>Sn:IF</i>	Scanned event IF statement
<i>Sn:TMR</i>	Scanned event timer
TIMER $n$	Event timers

---

## **Xkey (Xn) Functions**

The keys on the Operator Terminal marked X1 through X4 are special purpose keys that, when pressed, may cause the execution of a routine in parallel with a main program. To enable an Xkey, a routine must be written for the Xkey and the program must contain an X $n$  ON or X $n$  CONT statement, where  $n$  is 1 through 4. Xkey routines are defined in the application program and are unique to that program. If you need the same Xkey program in more than one program, the code must be duplicated in each program. An alternative to this might be to define an Fkey function. Fkey functions can be made available to all

**TUTORIAL**

programs and may be active even when no program is running.

The program statements in an Xkey routine can be any legal program statement except motion statements or subroutine calls. Motion statements include MOVD, MOVF, DV, DIF, DELAY, and DWELL. JUMP commands in an Xkey routine must be to a label contained in that Xkey routine; jumps outside the Xkey routine are not allowed. PRINT and READ commands for the Operator Terminal are allowed.

The  $X_n$  statement is used to mark the beginning of a routine for an Operator Terminal Xkey, where  $n$  is 1 through 4. XEND marks the end of the Xkey routine. Xkey routines must be placed before the main body of a program and after any TITLE, ASSIGN, or  $S_n$ : statements.

If more than one Xkey is enabled, and an Xkey is pressed while the first Xkey routine is still executing, the first Xkey routine will be aborted and execution of the second will begin.  $X_n$  ON,  $X_n$  OFF, and  $X_n$  CONT statements can be used within an Xkey routine to avoid this situation. The following is a code fragment that shows the use of the  $X_n$  ON/OFF instructions to prevent the X1 program from being interrupted.

```
X1:                                ;Beginning of the X1 key routine
      CLEAR                        ;Clear the Operator Terminal Screen
      X4 OFF                        ;Disable X4 while in
X1 routine
      READ "Enter speed " ,SHOW G42
      X4 ON                         ;Re-enable X4 key
XEND                                ;End of X1 key routine
```

### $X_n$ ON/OFF/CONT

The  $X_n$  ON or  $X_n$  CONT statements are used to enable the Xkey program. After  $X_n$  ON is executed, one Xkey press will be recognized and then the Xkey will be disabled. Another  $X_n$  ON statement must be executed to re-enable the Xkey. As an alternative, the  $X_n$  CONT statement allows a key to be recognized continuously, without waiting for the program to re-enable the Xkey.

### XNPGM Statement

The XNPGM statement simulates pressing an Xkey from within a program. If the Xkey program has been enabled previously with an  $X_n$  ON or  $X_n$  CONT statement, the Xkey program will execute. The syntax is XNPGM = *number* where *number* can be any of the Xkey program numbers 1 to 4. Only one Xkey program can be running at a time. If an Xkey program is running when the XNPGM statement is executed, the current Xkey program ends and the new Xkey program is started, if the second Xkey program was enabled.

### Xkey Versus Fkey

You can accomplish the same task using Xkeys or Fkeys. The difference is in how the routines are programmed and executed. Xkey routines are defined right in the application program header. This means that each application program may define the Xkey differently. Fkey programs are stored as system programs. Fkey programs are assigned to the Fkeys through the Parameter menu Fkey Set Up dialog box.

Xkey programs are defined in an application program and therefore can only be executed while a program is executing. Fkey programs are defined separately and can execute without application programs running.

### Xkey Instruction Summary

The following table contains a summary of instructions that relate to Xkeys Refer to Part 5 • Language Reference for more detailed information.

Name	Description
XEND	Marks the end of an Xkey routine
$X_n$	Marks the beginning of an Xkey routine
$X_n$ ON/OFF/CONT	Enables and disables an Xkey routine
XNACTIVE	ON if an Xkey is pressed, OFF otherwise

Name	Description
XNPGM	Simulates the pressing of an Xkey

## Fkey (Fn) Functions

Function key programs are programs that are written and assigned to run whenever an associated Fkey is pressed on the operator terminal. The ULTRA Plus or IQ system directory programs 1 through 24 are designated for the storage of function key programs.

Function key programs may be assigned to the operator terminal keys in the Parameter menu Fkey Set Up dialog box. Six different combinations of function keys can be defined. Each of these sets is referred to as a mode. Once set up, the operator can scroll through the function key menus by pressing the MODE key on the Operator Terminal. Each press of the mode key will advance the Operator Terminal to the next mode. After the last defined mode is reached, the terminal will loop to the first mode.

### Program Control of Fkeys

In addition to the operator having control over the function keys, the application program may select and lock the function key mode. This is done using the PRINT instruction. Refer to Part 5 • Language Reference, PRINT, for more detailed information. The FNPGM statement may also be used to simulate the pressing of an Fkey.

### FNPGM Statement

The FNPGM statement simulates pressing an Fkey from within a program. The syntax is FNPGM = *number* where *number* can be any of the Fkey program numbers 1 to 24. Only one Fkey program can be running at a time. If an Fkey program is running when the FNPGM statement is executed, the current Fkey program ends and the new Fkey program is started.

### Creating and Replacing Fkey Routines

Fkey routines are very similar to regular application programs. They may not, however, contain subroutines, Xkey routines, scanned event routines, the LOOP command or motion statements. Motion statements include MOVD, MOV, DV, DIF, DELAY, or DWELL. To create a new Fkey routine, or replace an existing Fkey program, use the following procedure:

1. Avoiding the commands listed above that are not permitted in Fkey routines, write your program using the IQ Master editor.
2. Set the Program Type to Fkey Routine. Program Type is a compiler options and is set in the Edit menu, Compiler Options dialog box.
3. Choose Compile from the Edit menu or press F2 to compile the Fkey program. If there are any errors, correct them and compile.
4. Choose the “Save Program to IQ” dialog box from the File menu or press F7.
5. In the directory list box, select the system directory (System Dir).
6. In the List Program list box, select the FN program number that you would like to replace with this new routine.
7. Choose the Executable only or Source and Executable radio button.
8. Choose OK to save the program to the ULTRA Plus or IQ.

### Xkey Versus Fkey

You can accomplish the same task using Xkeys or Fkeys. The difference is in how the routines are programmed and executed. Xkey routines are defined in the application program header. This means that each application program may define the Xkey differently. Fkey programs are stored as system programs. Fkey programs are assigned to the Fkeys through the Parameter menu Fkey Set Up dialog box.

Xkey programs are defined in an application program and therefore can only be executed while a pro-

gram is executing. Fkey programs are defined separately and can execute without application programs running.

### Fkey Instruction Summary

The following table contains a summary of instructions that relate to Fkeys. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
FNACTIVE	ON if an Fkey is pressed, OFF otherwise
FNPGM	Starts an Fkey program
PRINT	PRINT can be used to set and change Fkey modes

## Interrupts

Interrupts are used to perform a specific task quickly in response to an event. The ULTRA Plus or IQ has four events that may trigger an interrupt. They are: an index pulse on encoder 1 or encoder 2, or input 11 or input 12 turning ON.

When any of these interrupts occur, the action taken is identical. The position of encoder 1 and encoder 2 are captured and saved in designated system variables, and flags are set to indicate that the interrupt has occurred and the positions are saved.

**Note:** The index interrupt for the 2nd encoder may not detect the index if the duration is less than 2 ms.

Interrupts are most frequently used when position information needs to be gathered while the axis is in motion, or when the relative position of encoder 1 and encoder 2 is required. This would be typical of an application requiring the motor to be synchronized to an encoder attached to some mechanism.

### Interrupts Step by Step

The steps required to use an interrupt are the same, regardless of the type of interrupt. They are:

- clear the flag which indicates that the interrupt has occurred (the flag is not automatically cleared after the interrupt occurs),
- enable the interrupt,
- test the flag to detect when the interrupt occurred,
- read the appropriate system variable.

### Example:

```

FI1 = OFF ;Clear the interrupt flag
INT1 ON ;Enable input interrupt 1
... Statements
;find the difference in counts between encoder 1 and encoder 2 when the
;interrupt occurred.
IF FI1 = ON V1 = @I1P1 - @I1P2

```

### Interrupt Variable, flags and Instructions Summary

The following is a summary of the instructions, variables and flags that relate to interrupts discussed in

this section. Refer to Part 5 • Language Reference detailed information.

Type	Name	Description
Input Interrupt	FI1	Flag to indicate that interrupt 1 has occurred
	FI2	Flag to indicate that interrupt 2 has occurred
	I1P1	Position from encoder 1, captured when interrupt 1 occurred
	I1P2	Position from encoder 2, captured when interrupt 1 occurred
	I2P1	Position from encoder 1, captured when interrupt 2 occurred
	I2P2	Position from encoder 2, captured when interrupt 2 occurred
	INT1	Enable and disable interrupt 1
	INT2	Enable and disable interrupt 2
	LPOS	Position from encoder 1 or 2 captured in hardware when interrupt 1 occurred
	SINT2	Force an interrupt 2 with a software instruction.
Index Interrupt	FIDX1	Flag to indicate that encoder 1 index interrupt has occurred
	FIDX2	Flag to indicate that encoder 2 index interrupt has occurred
	IDX1	Enable and disable encoder 1 index interrupt
	IDX2	Enable and disable encoder 2 index interrupt
	IX1P1	Position from encoder 1, captured when encoder 1 index interrupt occurred
	IX1P2	Position from encoder 2, captured when encoder 1 index interrupt occurred
	IX2P1	Position from encoder 1, captured when encoder 2 index interrupt occurred
	IX2P2	Position from encoder 2, captured when encoder 2 index interrupt occurred

## Miscellaneous Structure Instruction

As you write any program, one of your goals should be to make the program easily understood by others that may be required to read or maintain the program in the future. The following are guidelines that will make your programs easier to develop and maintain:

- Use a lot of comments. There is no such thing as having too many comments in a program. These comments will be invaluable in the future when you or someone else attempts to modify the program to add new features into your application.
- Use the ASSIGN command to assign meaningful names to variables, inputs and outputs. This will make the program easier to write, read, and debug.
- Keep a revision history of program modifications embedded in the comments at the top of your application program.

### Miscellaneous Structure Instructions Summary

The following is a summary of the miscellaneous structure related commands discussed in this section. Refer to Part 5 • Language Reference for more detailed information.

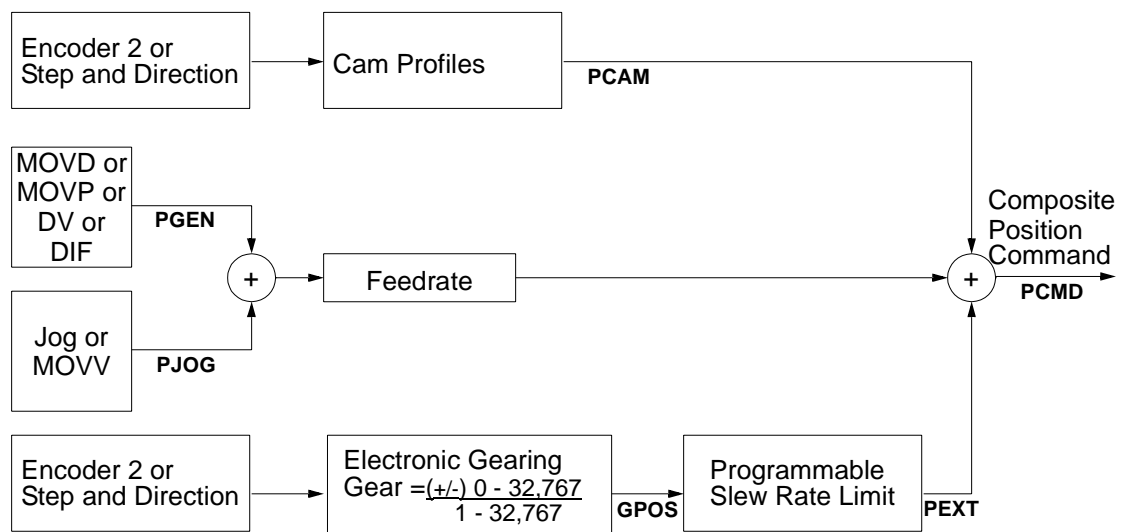
Name	Description
PGMTYPE = <i>type</i>	Compiler option for program type
DEBUG = ON/OFF	Compiler option for debug information
ASSIGN <i>name text</i>	Assign a name to a text string or variables
TITLE " <i>name</i> "	Define a program title
;comment	Comment in the program
END	End of the program

# Motion

## Introduction

Position commands that cause motion can be generated from several sources with the ULTRA Plus or IQ. The figure below illustrates the generation of the position command that is sent to the position regulator. Each of four sources is independent and may run simultaneously. The resulting composite position command is the sum of the position commands from the four different sources.

$$PCMD = PGEN + PJOG + PEXT + PCAM$$



PGEN is the position from the profile generator. The profile generator is used by MOVD, MOV P, DV, and DIF instructions. MOVD commands generate incremental distance moves of a specified length. MOV P commands generate moves to a specified position. DV and DIF instructions are used to con-



struct complex profiles. MOVD and MOVP commands can be in the application program or can be a Host Language command. DV and DIF commands can only be used in the main body of the application program.

PJOG is the position command generated by jogging motion. Jogging motion ramps up to a specified velocity and then continues to run at that velocity until a new velocity is specified. Jogging motion can be initiated by jog inputs, Host Language Commands, or MOVV instructions in the application program.

PEXT, the third component of position command, comes from an external source. Encoder 2 may be connected to an encoder or to a pulse and direction generator. The ULTRA Plus or IQ can be programmed to add the command from this source to the internally generated sources.

PCAM, the fourth component, is the position command generated by executing Cam Profiles. IQ Master with Cam is an optional software program for creating complex electronic cam profiles. Refer to the *IQ-Series IQ-Cam Software Manual* (Part Number 0013-1053-001).

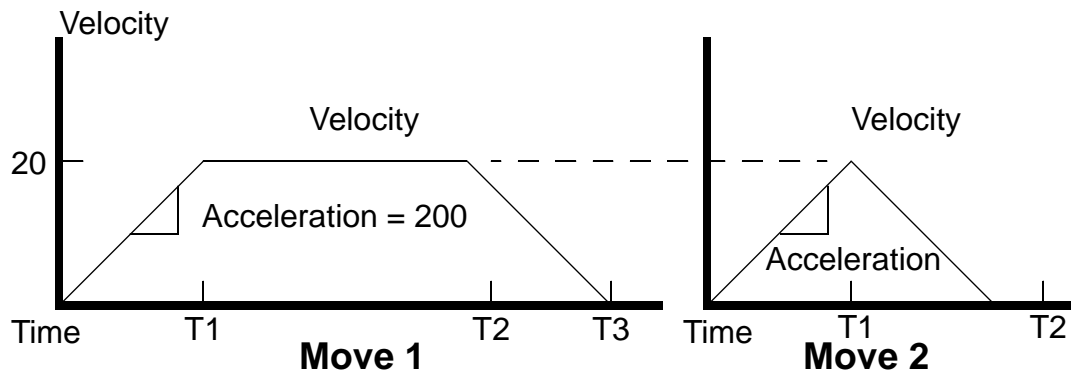
In the remainder of this section, individual sources of motion will be discussed in greater detail followed by a section about using combinations of these sources.

## Incremental (MOVD) and Absolute (MOVP) Motion

MOVD and MOVP instructions are used to create incremental and absolute moves respectively. The motion that results from these commands is a trapezoidal velocity move. The length of the move is determined by a required argument of the instruction. The top velocity of the move is determined by an optional argument in the instruction, by the setting of the velocity system variable, VEL, or by the default velocity in parameters (in order of priority). The acceleration and deceleration rates for the move are determined by the setting of the ACCEL variable or the default value in parameters (in order of priority).

If the values for velocity, acceleration, and move distance are conflicting, the move will be converted from a trapezoidal velocity profile to a triangular profile with the ULTRA Plus or IQ calculating the peak velocity. The resulting peak will be less than the one specified in this case.

```
ACCEL = 200      ;set the acceleration to 200
VEL = 20        ;set the velocity to 20
MOVD 6          ;Move 1 a distance of 6
MOVD = 1.8     ;Move 2 a distance of 1.8
```



In the example above, the commanded acceleration and velocity were equal, but the distances were different. In move 1, the peak velocity reached the specified velocity of 20. In move 2, the profile was changed to a triangle with a peak velocity of less than 20. The acceleration rate of each move was the specified value of 200. The distance was also the specified distance in each case.

### Incremental (MOVD) Motion

Incremental motion is defined as a move of some distance from the current position. Move four revolutions from the current position is an example of an incremental move (MOVD=4).

MOVD is the instruction in IQ Basic to create incremental moves. The simplified syntax for this instruction is:

$$\text{MOVD} = \textit{distance}$$

This instruction causes the motor to move *distance* from the current position. The sign (+/-) of *distance* will determine which direction the motor will go. If the sign is positive the motor will spin in the positive direction. The positive direction is defined in the Parameter menu, System dialog box.

### Absolute (MOVP) Motion

Absolute motion is defined as a move to some fixed position from the current position. The fixed position is defined as a position relative to a fixed zero point. The zero point for a system is established during a homing cycle, typically performed immediately after power-up.

During a homing cycle, the motor will make incremental moves while checking for a physical input, index mark, or both. Once the input(s) is detected, the ULTRA Plus or IQ knows where the motor is relative to the machine and establishes a zero point. See Appendix A for more details.

MOVP is the instruction in IQ Basic to create absolute moves. The simplified syntax for this instruction is:

$$\text{MOVP} = \textit{position}$$

This instruction causes the motor to move to *position* from the current position. The direction the motor moves is determined by the current position relative to the commanded *position*. If the command position is greater than the current position, the motor will go in the positive direction. If it is less, it will go in the negative direction.

---

## Stick Moves

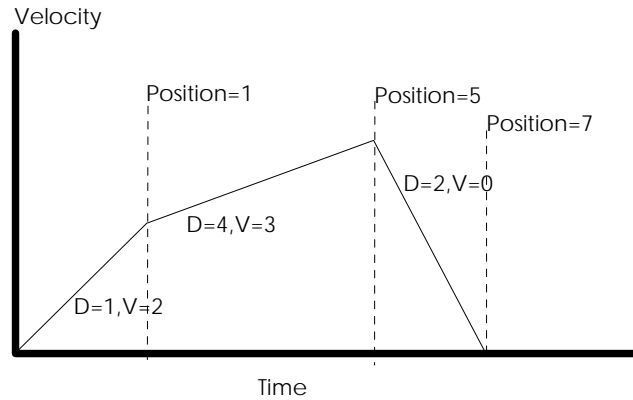
In addition to the simple trapezoidal or triangular velocity profiles that can be generated using the MOVD and MOVP instructions, complex profiles can be generated using stick moves. A stick move represents one portion of a complete move. A complete move is constructed out of two or more sticks, starting and ending at zero velocity.

### D, V Segments (Sticks)

Sticks are created using a sequence of DV and/or DIF instructions. The simplified syntax for the DV instruction is:

$$\text{D} = \textit{distance}, \text{V} = \textit{velocity}$$

Distance is the length of the stick move. *Velocity* is the final velocity for the stick move. The starting velocity is either zero or the final velocity of the previous stick. The final stick in a complete move must have a velocity of zero.



This represents a three segment (stick) move. The first segment has a starting velocity of 0 and final velocity of 2. This velocity is reached by the time the motor travels 1 unit of distance. The second stick starts at a velocity of 2 and accelerates to a velocity of 3 while traveling a distance of 4. The final stick starts at a velocity of 3 and decelerates to a stop while traveling a distance of 2. The instructions to make this move are:

$$D = 1, V = 2$$

$$D = 4, V = 3$$

$$D = 2, V = 0$$

The following equation can be used to calculate the acceleration that results from a stick move.

$$Accel = \frac{(V_f^2 - V_o^2)}{2D}$$

$$V_f \equiv \text{Final Velocity}$$

$$V_o \equiv \text{Starting Velocity}$$

$$D \equiv \text{Distance}$$

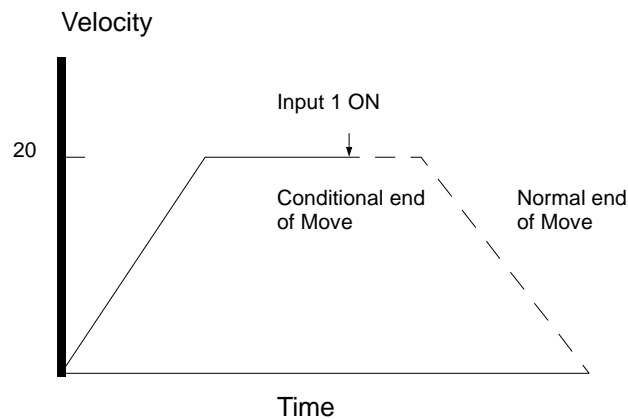
The program execution does not continue until the commanded profile is completed. Execution will then continue at the next instruction.

### DIF Segments (Sticks)

A DIF segment is a special form of a DV instruction and may not be used as the first stick of a move. A DIF instruction will move the distance specified unless a specified condition is satisfied during the move. The velocity for the move is that of the final velocity of the previous DV move. If the condition is met, the action specified in the instruction is taken. If the condition is not met during the move, the next program statement is executed when the move distance is reached. The syntax for this instruction is:

*DIF = distance, condition, JUMP label*

Distance is the incremental distance that will be moved unless the *condition* is satisfied. *Condition* specifies the condition that must be met for the action to take place. *JUMP label* specifies where the program should jump to if the condition is true.



```

D = 1, V = 20                               ;Accel to 20 in a distance of 1
DIF = 4, I1 = ON, JUMP STOPNOW              ;If input 1 comes on stop now else
STOPNOW:                                     ;Stop after a distance of 4
D = 1, V = 0                               ;Decel to a stop in a distance of 1

```

### S-Curve Acceleration

Instead of using linear acceleration, motion created using stick moves (DV commands) can use S-Curve acceleration. The syntax for a DV move with S-Curve acceleration is:

$D = \text{distance}$ ,  $V = \text{velocity}$ ,  $S$

Sticks using S-Curve acceleration will take the same amount of time as linear acceleration sticks. S-Curve acceleration is useful because it is much smoother at the beginning and end of the stick, however, the peak acceleration of the stick will be twice as high as the acceleration used in the linear acceleration stick.

## Gearing

For some applications, it may be desirable to tie the motion of one axis to another for simultaneous movement at some fixed ratio or gear. The GEAR instruction allows this kind of “electronic gearing.” You can use the GEAR instruction to tie the motor to the motion of encoder 2, so that when encoder 2 moves a given distance, the motor moves a proportional distance. The proportion is set by the GEAR instruction.

The GEAR is established in units of encoder counts and is set as the number of encoder counts to command the motor for each encoder count received from encoder 2. A gear of 2/3 would result in 2/3 of a count being commanded for each count on encoder 2.

Negative gear ratios are also allowed. A negative gear will cause the motor to move in the direction opposite to that of encoder 2.

Once the gear ratio is set with the GEAR instruction, electronic gearing must be enabled before the motor will begin following. GEAREN = ON is used to enable the gear and GEAREN = OFF is used to disable the gear.

In some cases, the encoder may be able to accelerate faster than the motor or load. In this case, a maximum acceleration limit for the motor can be set that is acceptable both to the motor and load. The SLEW instruction sets the slew rate and SLEWEN = ON enables this limit.

The slew limit is particularly useful when encoder 2 is in motion before the gear is enabled. If there was

no slew limit, the motor would attempt to accelerate up to the speed of the encoder instantaneously. With a proper slew limit set, the motor will make a smooth acceleration up to the speed of the encoder. Default values for SLEW and SLEWEN may be set in the Parameter menu, Velocity/Acceleration dialog box and changed at runtime. The following steps are required to set the ULTRA Plus or IQ to gear:

- In the Parameter menu, Feedback Configuration, dialog box select Encoder 2 IN. The Step and Direction check box must be cleared to select Encoder 2 IN as they are mutually exclusive.
- If required, set (SLEW) and enable (SLEWEN) a slew rate limit. Default values may be set in the Parameter menu, Velocity/Acceleration dialog box.
- Set (GEAR) and enable (GEAREN) the gear

**Note:** The SLEW limit remains active until it is shut OFF. If the acceleration rate of the master encoder after the GEAR is greater than the SLEW limit, the acceleration of the follower will be limited to the SLEW limit. If this limit is exceeded, the follower will lose position lock with the master.

Gear ratios may be modified while active by executing another GEAR command in the application program or by using a Host Language Command. A motor will continue to gear to the encoder input after the application program ends. To stop the gearing motion, you must execute GEAREN = OFF or disable the ULTRA Plus or IQ.

In addition to standard gearing functions, there is also a function to limit the gearing motion in one direction only. The TRACKINGMODE variable is used to set up the type of uni-directional gearing. The ROTT flag is used to set the direction, and the TRACKINGSYNCED flag is available to indicate whether gearing motion is being presently limited by the tracking function. In addition, the TRACKINGEXTPOSSSELECT flag is used to set whether the tracking function is applied before or after the Gear and Slew functions.

---

## Step and Direction

The ULTRA Plus or IQ can be set to follow step and direction commands entered through encoder 2, similar to gearing. In this case, the GEAR sets the number of step commands per encoder count of motion on the motor. The following steps are required to set the ULTRA Plus or IQ to follow step and direction commands:

- In the Parameter menu, Feedback Configuration, dialog box select Step and Direction.
- If required, set (SLEW) and enable (SLEWEN) a slew rate limit. Default values may be set in the Parameter menu, Velocity/Acceleration dialog box.
- Set (GEAR) and enable (GEAREN) the gear

Gear ratios may be modified while active by executing another GEAR command in the application program or by using a Host Language Command. A motor will continue to follow step and direction signals after the application program ends. To stop this motion, you must execute GEAREN = OFF or disable the ULTRA Plus or IQ.

---

## Combination Motion

The motion that is commanded is the sum of the motion commanded from four sources, the profile generator, jogging motion, external sources and cam functions. The profile generator generates MOVD, MOVP, DV and DIF moves. Jogging motion is created using the assigned jog inputs or MOVV commands. External motion commands are the result of a gear or a step and direction command. Cam motion commands are the result of executing cam profiles. An example of combination motion would be a program that sets a gear so that a motor could follow an encoder. Once the gear is established, the relative position of the motor to the encoder may need to be advanced or retarded. This can be accom-

plished by executing a MOVD. The MOVD command will cause the motor to move by the specified *distance* relative to the encoder while following the encoder.

---

## Feedrate

Feedrate (FDR) scales the timebase for motion. With the feedrate set at 100%, all velocities and DWELLS are at the programmed rates. Feedrate can be set to less than 100% to slow down a process, or above 100% to speed it up. Feedrate can be changed at any time in the program and is entered as a percentage with a range of 0 to 200.

Feedrate will affect MOVD, MOVV, MOVV, DIF, DV, and DWELL instructions. The velocity, acceleration, and time in these functions will be scaled by the FDR percentage.

For example, if the velocity is set to 3,000 RPM and the FDR is set to 50, any motion will have a velocity that is 50% of 3,000 RPM or 1,500 RPM ( $3000 * 0.5 = 1500$ ). A DWELL time is also scaled by FDR. A DWELL of 10 seconds would be scaled to 20 seconds ( $10 / 0.50 = 20$ ). A feedrate can also be set with the analog input. This is accomplished by assigning the analog input to FEEDRATE in the Parameter Inputs dialog box.

Changes made to this variable by a program are only in effect while the program is running. When the program ends, feedrate will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu.

---

## Dwell Statement

The DWELL statement will cause the program to stop for the specified amount of time. The time will be scaled by the feedrate system variable discussed above.

---

## Delay Statement

The DELAY statement will cause the program to stop for the specified amount of time. The time will NOT be scaled by the feedrate system variable.

---

## Motion Instruction Summary

The following is a summary of the commands that can be used to generate motion. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
$D = dist, V = vel [ , S]$	Define one linear segment of a move
$DELAY = time$	Delay for a specified time (not affected by feedrate)
$DIF = dist, cond, JUMP label$	Move until the distance is reached or the condition occurs
$DWELL = time$	Dwell for the specified time (affected by feedrate)
FDR	Adjust the feedrate for motion from 0% to 200%
$GEAR = value$	Set the gear ratio
$GEAREN = ON/OFF$	Enable/Disable gearing

Name	Description
MOVD = <i>dist</i> , <i>vel</i>	Move an incremental distance
MOVP = <i>pos</i> , <i>vel</i>	Move to an absolute position
MOVV = <i>vel</i>	Move at a velocity
ROTT =- ON/OFF	Sets direction of the Tracking function
SLEW = <i>value</i>	Establish the maximum acceleration for the gear
SLEWEN = ON/OFF	Slew enable/disable
TRACKINGEXTPOSSELECT = ON/OFF	Sets whether the Tracking function is applied before or after the Gear and Slew functions
TRACKINGMODE = <i>mode</i>	Sets the mode of the Tracking function
TRACKINGSYNCD	Indicates the Tracking function is presently limiting Gearing motion

# Closed Loop Control

---

## Introduction

Closed loop control systems operate by using feedback to measure the actual output of the system being controlled and comparing the measured output to the desired or commanded output. The comparison of actual to commanded output yields the error of the system, and the closed loop regulator acts on this error to cause the controlled variable to follow the command. The regulator acts on the error based on various gains, which may produce an output proportional to the error (proportional gain, an output proportional to the error accumulated over time (integral gain), or some combination of these. The output of the regulator is the input to the system being controlled, and as the system error changes, the regulator output changes to cause the system to follow the command.

A typical motion control system has several closed control loops operating together to control motion, with a velocity control loop closed inside of a position control loop. The motion controller generates a position command for the system to follow, and the position regulator compares this command to the measured position from the feedback device. The resulting position error is acted on by the loop gains to produce an output which is the input to the inner velocity control loop. This velocity command is then compared to measured velocity, and the velocity loop gains act on the velocity error to produce a torque command. The amplifier applies a current to the motor to produce the desired torque in the motor.

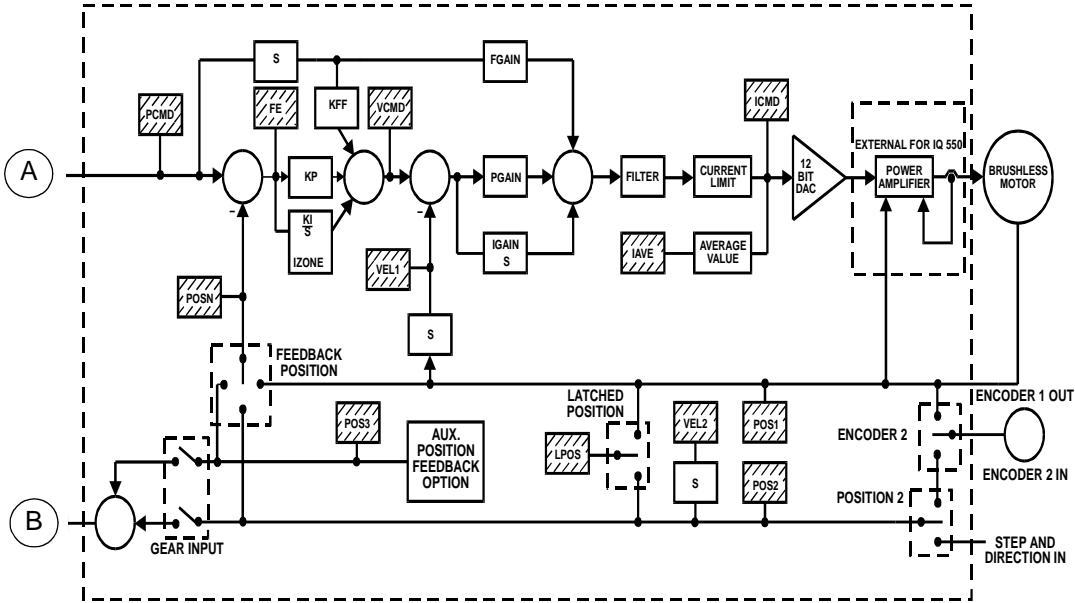
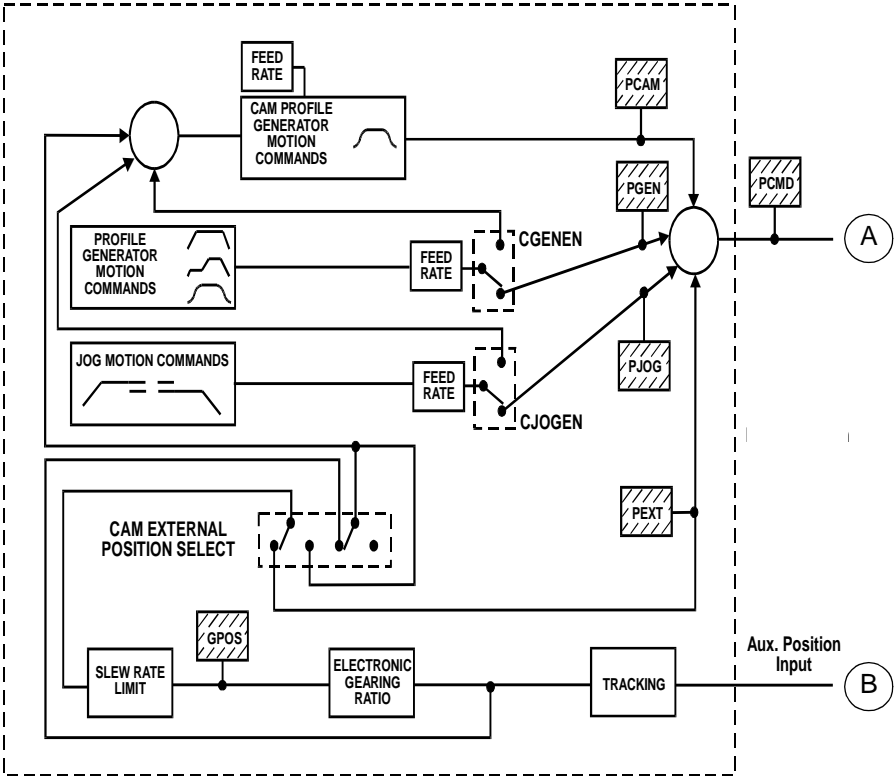
The ULTRA Plus or IQ controller architecture uses a velocity loop inside a position control loop as described above. The feedback device for both loops is an optical incremental encoder, which is used to measure motor position and calculate motor speed. The encoder mounted on the motor is always used for the velocity feedback device, and is usually also used as the position feedback device. Optionally, an external encoder can be used for the position feedback.

Following is a block diagram of the internal workings of the ULTRA Plus or IQ.

- Items within the dashed box represent functions that are performed within the ULTRA Plus or IQ.
- Boxes that are cross hatched are read only system variables that can be read in an application program or read with a Host Language command.
- Clear boxes represent system variables that can be read and written.



- Switches represent choices that are made in the configuration menus.
  - Circles are summing junctions that indicate that the signals flowing in to the circle are summed and the signal flowing out is the result of the summation.
  - The symbol “S” in a block indicates a Laplace transform. S is a derivative and 1/S is an integral.
- For example, the position command (PCMD) flows into a derivative box and the output of that box is the commanded velocity that is fed into KFF and FGAIN.



ULTRA Plus IQ-Series Block Diagram

TUTORIAL

## Closed Loop Position Control

Position control applications generally fall into two broad categories: contouring and point-to-point positioning. Contouring applications require that the actual position follows the commanded position in a very predictable manner and that the system has high stiffness to reject external torque disturbances. Note that the actual position must follow the commanded position in a predictable manner, not necessarily with zero following error at all times. Point-to-point positioning systems are typically not concerned about following a particular path, but move time, settling time, and velocity profile are important.

The position control loop in the ULTRA Plus or IQ-Series implements a Proportional plus Integral (PI) type loop with velocity and acceleration feedforward. The position command input to the position loop is the sum of the motion profile generator output, the jog commands, the external position command generated by the electronic gear, and the cam profile generator output. This total position command is used to generate the velocity and acceleration feedforward terms, and is compared to the position feedback from the encoder. The position feedback is typically from the encoder mounted on the back of the motor, or optionally from an encoder mounted externally on the machine. The selection of the position feedback source is made in the Feedback Config menu, as shown by the switches in the block diagram. The position feedback is subtracted from the position command, resulting in the Following Error (FE). The proportional and integral gains act on the following error.

The proportional gain, KP, is the most important term and generates a velocity command proportional to position error. This means that if only KP gain is used, motion is only possible if a position error exists, and higher velocities require proportionally higher position errors. Increasing the KP gain will reduce the following error. However, KP gains that result in a position loop bandwidth above about a third of the velocity loop bandwidth tend to cause the actual position to overshoot the commanded position, which is usually undesirable.

The velocity feedforward gain, KFF, generates a velocity command signal proportional to the derivative of the position command. Therefore, if there is no change in the position command, the feedforward command is always zero. Ideally, 100% velocity feedforward would provide the exact velocity command without the need for any position error. In practice, actual systems include loads which are not ideal, so a more conservative approach to setting the velocity feedforward gain (100% or less) is often used. Because the feedforward command is generated open-loop, there is no effect on the position loop stability. The function of the velocity feedforward command is to significantly reduce the constant velocity following error when the KP gain is maintained at the proper value for stability.

The acceleration feedforward term, FGAIN, scales the second derivative of the position command. This term allows position following error to be reduced when the velocity is changing. The acceleration feedforward term is useful in combination with the velocity feedforward to maintain a low following error at all times, such as in a tracking application or when a fast settling time is required.

The last term is the integral gain, KI, which can provide a velocity command signal to reduce static position error to zero. Integral gain provides stiffness against torque disturbances and friction torques which are usually handled by the velocity controller. Integral gain in the position controller is normally avoided except for special situations. Integral gain in the position controller tends to cause overshoot, so the ULTRA Plus or IQ-Series controller allows the integrator to be active only when the actual position is close to the commanded position. This zone around the commanded position where KI is active is set by the IZONE parameter.

---

## Closed Loop Velocity Control

The most common velocity controller structure is a Proportional plus Integral (PI) loop. The choice of the proportional (PGAIN) and integral (IGAIN) gains for the desired response is made based on the application requirements. Proportional gain is always used, and higher PGAIN values result in higher velocity loop bandwidths. The integral gain provides stiffness to load torque disturbances and reduces the steady-state velocity error to zero. However, integral gain does add phase shift to the velocity controller and can result in a longer settling time. Therefore, a low or zero IGAIN is sometimes used in point to point positioning applications requiring fast settling times, while a higher IGAIN is used in applications requiring high stiffness and a predictable path. In practice, the velocity controller is usually tuned using trial-and-error techniques with the motor coupled to the real or simulated load. The ULTRA Plus or IQ-Series controller provides automatic tuning of the velocity and position loops which is adequate for many applications. The manual tune mode applies a small signal step command to the loop, and allows real-time adjustment of the gains while observing the response to optimize the tuning.

# Inputs / Outputs

---

## Introduction

Inputs and outputs can be read and set as individuals or in groups. Groups of inputs can be read in a binary (BIN) format or binary coded decimal (BCD) format. Outputs can be set in BIN or BCD format or set all at one time.

---

## I/O Instruction Summary

The following is a summary of the commands that can be used set outputs and read inputs. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
<i>ALL ON/OFF</i>	Turn all outputs ON or OFF
<i>BCD Om,n = variable</i>	Set a group of outputs to the BCD value of the variable
<i>BCD variable = Im,n</i>	Read a group of inputs as a BCD value into the variable
<i>BIN Om,n = variable</i>	Set a group of outputs to the binary value of the variable
<i>BIN variable = Im,n</i>	Read a group of inputs as a binary value into the variable
<i>DACn = value</i>	Set the <i>DACn</i> output to a voltage
<i>Fn = Im</i>	Set a Flag ON if the input is ON and OFF if it is OFF
<i>Om = Fn</i>	Set an output to the state of the F variable
<i>Om = ON/OFF[,T = time]</i>	Turn an output ON or OFF
<i>variable = ADCn</i>	Read the value of an analog to digital converter

# Operator Terminal

---

## Introduction

An optional Operator Terminal can be used to print messages and values for an operator, and input data from an operator. (see Fkeys and Xkeys above).

---

## Operator Terminal Instruction Summary

The following is a summary of the commands that are used with the operator terminal. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
CLR [ <i>row</i> [, <i>column</i> , <i>length</i> ]]	Clear the Operator Terminal screen
OTMON = <i>variable</i>	Display a monitor variable on the Operator Terminal
PRINT ...	Print a message to the terminal
READ ...	Read input from the terminal

# Miscellaneous

## Introduction

Included in this section are the miscellaneous commands that do not fit into any of the other sections.

### CONFIG Statement

Up to four configurations can be defined for a ULTRA Plus or IQ. The configuration determines which encoder is used for the position loop feedback, and whether the encoder 2 connector will bring encoder signals into the controller or source encoder signals out to an external device. Encoder 2 may also be set to receive step and direction commands from an external controller.

For some applications, it may be useful to change the configuration for various operations. For example, it may be desirable to get the position feedback directly from a measuring wheel riding on a piece of material to be cut while indexing that material into the cutoff knife. When the pusher retracts to load in another piece of material, the feedback will need to come from the motor encoder.

To accomplish this, two configurations could be established. the first for when the feedback is from the measuring wheel and the second for when the feedback is from the motor encoder. Once the configurations are established, the application program can select which one is active by executing a `CONFIG = n` instruction, where *n* is 1 through 4.

```
CONFIG = 2;set the current configuration to 2
```

### Define Home (DH), Define Position (DP) and Define Position 1 or 2

To do absolute position moves with an incremental encoder, an absolute reference must be established. The DH, DP, DP1 and DP2 commands are used to define the absolute reference position. The DH command will define the current commanded position as the zero position when executed. The DP command will define the current commanded position as the position specified in the instruction when executed. DP and DH also zero the Encoder 2 position. DP1 redefines the encoder 1 position to the position specified in the instruction when executed. DP1 does not affect encoder 2. DP2 redefines the encoder 2 position to the position specified in the instruction when executed. DP2 does not affect encoder 1.

```

DH                ;define current commanded position as zero
DP = 200          ;define the current commanded position as 200
DP = 0            ;this is equivalent to a DH
DP1 = 100         ;define the encoder 1 position (POS1) as 100

```

These commands change the commanded position. The feedback position is also changed in such a way as to maintain the same following error that was present before the instruction was executed.

### Enable / Disable

The ENABLE and DISABLE commands are used to enable and disable motor torque. These commands can be issued from three different sources: the application program, an assigned input, or a Host Language Command. Any of the three sources can disable the ULTRA Plus or IQ at any time. The only source that can re-enable the ULTRA Plus or IQ (without resetting the ULTRA Plus or IQ) is the source that disabled the ULTRA Plus or IQ.

```

DISABLE           ;Disable the motor
ENABLE           ;Enable the motor

```

---

## Miscellaneous Instruction Summary

The following is a summary of the miscellaneous commands discussed in this section. Refer to Part 5 • Language Reference for more detailed information.

Name	Description
CONFIG = <i>n</i>	Select the ULTRA Plus or IQ configuration
DH	Define the current position as home
DISABLE	Disable the ULTRA Plus or IQ
DP = <i>value</i>	Redefine the current position
DP1 = <i>value</i>	Redefine Encoder1 position
DP2 = <i>value</i>	Redefine Encoder2 position
ENABLE	Enable the ULTRA Plus or IQ

TUTORIAL



# Part 5

# Language Reference

The Language Reference part contains a detailed description of each function, system variable and system flag. The items are arranged in alphabetical order.



# Reserved Words

Variables may be assigned names up to 32 characters in a program using the ASSIGN statement. The names of dedicated variables and dedicated flag variables are reserved, so an attempt to name another variable with a reserved variable name is not allowed. Dedicated variables and dedicated flag variables are pre-defined and available for use in a program. Other names that are reserved are I1 through I16, O1 through O8, and S1 through S8. In addition to the reserved variable names, language statements are reserved keywords that cannot be used as labels or names in a program.

ABORT	ABSMODE	ACCEL	ACTIVESN
ACTIVEXN	ADC1	ADC2	ADC3
ADC4	ADC5	ALL	AND
ASSIGN	ATHOME	AUTOPGM	B1...B8
BCD	BIN	CALL	CAM
CF	CLEAR	CLR (CLEAR)	CLRPEAKS
COMPILEMEMORY	CONFIG	CONT	CTOU1
CTOU2	D	DAC1	DEBUG
DELAY	DH	DHCMD	DIMCAM
DIF	DIS	DISABLE	DISASSEMBLE
DISLIM	DISMTN	DO	DP
DPM	DWELL	DWL	EDGE
EFLAG	EFM	ELSE	ENA (ENABLE)
ENABLE	ENCMD	END	ENUM
EPOS (ERETPOS)	ERET	ERETPOS	ERM
ERRPGM	EXPANDMACROS	F	F1...F64
FDR	FE	FEL	FET
FGAIN	FI1	FI2	FIDX1
FIDX2	FILTER	FKEYPGM	FNACTIVE
FNPGM	FNVAR1	FNVAR2	FOUNDFHOME
FVEL1	G1...G64	GEAR	GEAREN
GPOS	HLIMITS	HOFFS	HOMECMD
HRESET	HSEQCPL	HSWEN	HSWSTAT
HVEL	I1...I48	I1P1	I1P2
I2P1	I2P2	IAVE	IAVG
ICMD	IDX1	IDX2	IF
IGAIN	IL	ILIMIT	INDEXEN
INPOS	INT	INT1	INT2
IX1P1	IX1P2	IX2P1	IX2P2
IZ	IZONE	JACCEL	JDECCEL
JMP	JOGR	JOGR	JUMP
JVEL	KFF	KI	KP
KPZ	LISTFILE	LOOP	LOOPINDEX
LPOS	MAINPGM	MD (MOVD)	MONOUT

MOVD	MOVP	MOVV	MP(MOVP)
MV (MOVV)	MVOEN	NAND	NOR
NOT	O1...O24	OFF	ON
OR	OTMON	OVERSPEED	PAUSE
PCAM	PCAMACTIVE	PCMD	PEXT
PEXTACTIVE	PFE	PGAIN	PGEN
PGMNUM	PGMRUNNING	PGMTYPE	PICMD
PJOG	POS1	POS2	POS3
POSN	PRINT	PRINTERR	PRINTWARN
PURGEMOTION	PURGEMOTIONACTIVE	PVEL1	PZ
PZONE	QUEUECAM	READ	REPEAT
RET	RETURN	RFDR	ROT
ROT2	RPT	S	S1...S8
SCALE	SCALE2	SEN	SHOW
SINT2	SLEW	SLEWEN	SRESET
SSHIFTS	SSHIFTS2	STARTP	STOP
SUB	SYSERROR	SYSWARN	T
TBASE	TIMER1	TIMER2	TITLE
TMR1	TMR2	TRIGGERCAM	UTOC1
UTOC2	V	V1...V64	VCMD
VEL	VEL1	VEL2	WAIT
WFLAG	WHILE	WIN	WINDOW
WNUM	X1...X4	XEND	XLOOPINDEX
XNACTIVE	XNPGM	XOR	

# Language Reference

---

## Format

Each function, system variable, and system flag is documented using the format shown below. If there is no information in one of the fields the label is still shown.

<b>KEYWORD</b>	Long Name	<i>Type</i>
<b>Purpose</b>		
<b>Syntax</b>	KEYWORD = <i>value</i> <i>Variable</i> = KEYWORD <i>arguments</i>	
<b>Remarks</b>		
<b>See Also</b>		
<b>Example</b>		
<b>KEYWORD:</b>	The keyword is the name of the function, system variable, or system flag as it would appear in a program or host command.	
<b>Long Name:</b>	The long name is an interpretation of the keyword. For example: HVEL is the keyword and Home Velocity would be the long name. The long name is provided only as an aid to the reader and may not be used in the program.	
<b>Type:</b>	This field identifies what type of function, system variable, or system flag the keyword is.	
<b>Purpose:</b>	This field identifies the purpose of the keyword.	
<b>Syntax:</b>	This field shows the proper usage of the keyword. If the keyword is a system variable or system flag the syntax will show if the system variable or system	

flag may be read, written, or both. Optional arguments will be contained with square bracket []. Arguments will be written in *italics*.

- Arguments:** The data that is supplied with a function that modifies the behavior of the function. For example, `MOVD = 100`. `MOVD` is the function and `100` is the argument.
- Remarks:** The remark field contains additional information about the usage of the function, system variable, or system flag.
- See Also:** This field contains a list of functions, system variables, and system flags that are related to the purpose of the keyword.
- Example:** The example field contains a code segment that illustrates the usage of the keyword in the program.

---

## Reference

<b>@</b>	<b>Read Position in Counts</b>	<b>Conversion</b>
<b>Purpose</b>	The @ command will cause a position variable to be read in units of counts instead of user units.	
<b>Syntax</b>	<i>variable</i> = @ <i>Posn_Variable</i>	
	Variable	<i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
	<i>Posn_Variable</i>	Any position Variable: POS1, POS2, LPOS, POSN, FE, PCMD, PEXT, PFE, PGEN, PJOG, <i>InPm</i> , <i>IXnPm</i> and GPOS.
<b>Remarks</b>		
<b>See Also</b>	CTOU1, CTOU2, UTOC1, UTOC2, “Conversions” on page 131	
<b>Example</b>	V3 = @POS1	

<b>ABORT</b>	<b>Abort Program</b>	<b>System</b>
<b>Purpose</b>	Stops program execution (but not current motion) when set.	
<b>Syntax</b>	ABORT = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or a system flag.
<b>Remarks</b>	This flag is normally not used in a finished program. It is more frequently used during program development as a method for stopping program execution at a specific point. It may also be used in a system program to stop a main program execution.	
<b>See Also</b>	STOP	
<b>Example</b>	ABORT = ON ABORT = F1	

<b>ABSMODE</b>	<b>Absolute Mode</b>	<b>System</b>
<b>Purpose</b>	If the flag is ON the controller is in Absolute mode.	
<b>Syntax</b>	<i>variable</i> = ABSMODE	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag Bn, a volatile flag Fn, or another system flag.
<b>Remarks</b>	<p>If Absolute mode is ON, the following features will be enabled: Software Travel Limits, the At Home Flag, the At Home Output, and the Emergency Return function. These features will not operate until a home position has been defined. Refer to Appendix A, System Programs, Home for more detailed information on defining a home position.</p> <p><b>Note:</b> In addition to the Absolute mode setting, the At Home output must also be assigned if it is to be used. Refer to Part 2 • IQ Master Environment, Parameters, Outputs, for more detailed information on At Home.</p>	
<b>See Also</b>		
<b>Example</b>	F1 = ABSMODE	

<b>ACCEL</b>	<b>Acceleration</b>	<b>Acceleration</b>
<b>Purpose</b>	Change the acceleration used to compute motion profiles.	
<b>Syntax</b>	ACCEL = <i>value</i>	
	<i>variable</i> = ACCEL	
	<i>value</i>	The units used for acceleration are user units per second squared. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable Gn, a volatile variable Vn or a system variable.
<b>Remarks</b>	<p>The acceleration may be changed at any time in the program. The new value will take effect the next time a MOVD, or MOV P move is executed. ACCEL does not effect the acceleration used in MOVV, DV, or DIF instructions.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.</p>	
<b>See Also</b>	JACCEL, JDECEL	
<b>Example</b>	ACCEL = 200 ACCEL = G2	



<b>ACTIVESN</b>	<b>Scanned Event(s) Active</b>	<b>System</b>
<b>Purpose</b>	Variable to indicate the number of the scanned event that is active.	
<b>Syntax</b>	<i>variable</i> = ACTIVESN	
	<i>variable</i>	The <i>variable</i> will be set to a value from 0 to 8. <i>Variable</i> = 0 indicates no scanned events are executing.
<b>Remarks</b>	This is a read-only variable.	
<b>See Also</b>	<i>Sn</i>	
<b>Example</b>	V1 = ACTIVESN	

<b>ACTIVEXN</b>	<b>Xkey Routine(s) Active</b>	<b>System</b>
<b>Purpose</b>	Variable to indicate the number of the Xkey routine that is active.	
<b>Syntax</b>	<i>variable</i> = ACTIVEXN	
	<i>variable</i>	The <i>variable</i> will be set to a value from 0 to 4. <i>Variable</i> = 0 indicates no Xkey routines are executing.
<b>Remarks</b>	This is a read-only variable.	
<b>See Also</b>	<i>Xn</i>	
<b>Example</b>	V4 = ACTIVEXN	

<b>ADC<math>n</math></b>	<b>Analog Input</b>	<b>Inputs</b>
<b>Purpose</b>	Analog input $n$ in volts.	
<b>Syntax</b>	<i>variable</i> = ADC $n$	
	<i>variable</i>	The <i>variable</i> is set to the voltage that is present on ADC $n$ . The voltage is measured in volts with two places after the decimal point. <i>Variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.
	$n$	The number of the analog input. $n = 1$ if the analog input on the ULTRA Plus or IQ is used. $n = 2, 3, 4,$ or $5$ if a Memory and I/O Expansion Card is installed
<b>Remarks</b>	Analog input 1 has 10 bits resolution to represent the input range of -10 volts to +10 volts which results in approximately 20mV resolution. Analog inputs 2 through 5 (located on an option card) have 12 bits resolution to represent the input range of -10 volts to +10 volts, which results in approximately 5mV resolution.	
<b>See Also</b>	Analog input $n$ in volts.	
<b>Example</b>	V1 = ADC1	

<b>ALL</b>	<b>All Outputs</b>	<b>Outputs</b>
<b>Purpose</b>	Turn all outputs ON or OFF. Assigned outputs such as Program Running, Error, At Home, Home Sequence Complete, or In-Position are not affected unless disabled from the Parameters menu, Outputs dialog box.	
<b>Syntax</b>	<i>ALL ON/OFF</i>	
	<i>ON</i>	Turn all outputs ON. Outputs are current sinking type. ON turns on the transistor and pulls the output to 24V common.
	<i>OFF</i>	Turn all outputs OFF. Outputs are current sinking type. OFF turns off the transistor and leaves the outputs floating. There is no internal pull-up.
<b>Remarks</b>	Refer to the appropriate <i>ULTRA Plus</i> or <i>IQ-Series Installation Manual</i> (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for electrical specifications.	
<b>See Also</b>	<i>On</i> = ON/OFF/ <i>Fn</i>	
<b>Example</b>	ALL ON ALL OFF	

**ASSIGN****Assign***Program Structure*

**Purpose** Assign a name to a text string. The name can be used to represent an actual parameter on a machine such as COUNT, LENGTH, etc., making the application program easier to write and read.

**Syntax** ASSIGN *name text*

*name* The *name* may be up to 32 alphanumeric characters long and must begin with an alphabetical character. *Name* can NOT be a keyword or reserved word.

*text* The *text* may be a variable (V, G, F, B, or dedicated variable), a variable math operation, a constant numerical value, a text string that could be used in a conditional statement, an input designator (I10), an output designator (O6), or a text string in double quotation marks to be used with a Print statement.

**Remarks** Assign statements must be located at the top of the program before any executable statements. When creating the executable program the compiler replaces *name* with *text* and then compiles the program (the source is not changed).

During program development the Expand Macros compiler option can be set to get a program listing that shows how the compiler interpreted your ASSIGN statement. Refer to Part 2 • IQ Master Environment, Edit menu, Compiler Options for more detailed information.

**See Also****Example**

```

ASSIGN COUNT V3      ;COUNT is variable V3
ASSIGN DEPTH G10     ;DEPTH is variable G10
ASSIGN MULT G10 * V2 ;MULT is the result of G10 * V2

ASSIGN TEST F1 = ON  ;TEST can be used in a conditional statement -
                    ;IF TEST MOVD = 3
ASSIGN OK INPOSN     ;OK is the same as the INPOSN flag
ASSIGN GO I7         ;GO is the same as input 7 (I7)

ASSIGN TXT "Length of Move?"
                    ;TXT is a text string to be used with a Print
                    ;statement
PRINT TXT            ;print Length of Move? on the Operator Terminal

```

<b>ATHOME</b>	<b>At Home Flag</b>	<i>Home</i>
<b>Purpose</b>	When the flag is ON it indicates that the motor is within the In-Position window of the zero position. ATHOME is used in Absolute mode only.	
<b>Syntax</b>	<i>variable</i> = ATHOME	
	variable	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	WINDOW, INPOSN	
<b>See Also</b>		
<b>Example</b>	<pre>IF ATHOME = ON O2 = ON F1 = ATHOME</pre>	

<b>Bn</b>	<b>Nonvolatile User Flag</b>	<i>Variable</i>
<b>Purpose</b>	Set a user flag ON or OFF with another flag or constant.	
<b>Syntax</b>	<i>Bn</i> = ON/OFF/ <i>Fm</i> / <i>Bn</i> <i>variable</i> = <i>Bn</i>	
	<i>n</i> , <i>m</i>	The number of the flag. <i>n</i> = 1, 2, 3, ..., 8 or <i>m</i> = 1, 2, 3, ..., 64.
	variable	Variable may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	<p>The changes made to <i>Bn</i> remain in effect even after the power is turned OFF. Changes made to <i>Bn</i> by one program will be seen by other programs.</p> <p><b>TIP:</b> Variables can be assigned names using the ASSIGN command and then referenced by those names to make the program easier to read.</p>	
<b>See Also</b>	<i>Gn</i> , <i>Fn</i> , <i>Vn</i>	
<b>Example</b>	<pre>B1 = ON B2 = OFF IF B1 = ON MOVD = 2 B3=F63</pre>	

<b>BCD <math>Om, n</math></b>	Output, Binary Coded Decimal	<i>Outputs</i>
<b>BIN <math>Om, n</math></b>	Output, Binary	
<b>Purpose</b>	Set a group of outputs, in either binary (BIN) or binary coded decimal (BCD) format, to the value of a variable.	
<b>Syntax</b>	BCD $Om, n = value$ BIN $Om, n = value$	
	<i>m</i>	The output specified by <i>m</i> is set to the least significant bit of <i>variable</i> .
	<i>n</i>	<i>n</i> is the number of outputs included in the group.
	<i>value</i>	The <i>value</i> can be a constant, a nonvolatile variable $Gn$ , a volatile variable $Vn$ or a system variable.
<b>Remarks</b>	If the value is greater than <i>n</i> bits, the least significant <i>n</i> bits are output.	
<b>See Also</b>		
<b>Example</b>	<pre>BCD 03, 4 = V1      ;Set outputs 3, 4, 5, and 6 BCD 03, 2 = 2      ;Set output 3 OFF and output 4 ON BIN 04, 3 = G7     ;Set outputs 4, 5, and 6</pre>	

<b>BCD variable = <math>Im, n</math></b>	Input, Binary Coded Decimal	<i>Inputs</i>
<b>BIN variable = <math>Im, n</math></b>	Input, Binary	
<b>Purpose</b>	Read a group of inputs in either binary (BIN) or binary coded decimal (BCD) format into a variable.	
<b>Syntax</b>	BCD <i>variable</i> = $Im, n$ BIN <i>variable</i> = $Im, n$	
	<i>m</i>	The input specified by <i>m</i> is the least significant bit of the group.
	<i>n</i>	<i>n</i> is the number of inputs included in the group.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ or a system variable.
<b>Remarks</b>	If the BCD instruction is used and the value is out of range, a BCD input out of range error will be generated.	
<b>See Also</b>	<i>Im</i>	
<b>Example</b>	<pre>BCD V3 = I9, 4      ;Read BCD value of inputs 9, 10, 11, and 12                  ;                     ;into V3 BIN V11 = I13, 2    ;Read binary value of inputs 13 and 14                     ;into V11</pre>	

<b>CALL</b>	Call Subroutine	<i>Program Structure</i>
<b>Purpose</b>	Call a subroutine to be executed. After the subroutine is completed, execution returns to the statement following the CALL statement.	
<b>Syntax</b>	CALL <i>subroutine</i>	
	<i>subroutine</i>	The name of the <i>subroutine</i> . Subroutine names can be up to 32 alphanumeric characters long. Names cannot be keywords or reserved words. They must begin with a letter and contain only alphanumeric characters and underscores ( _ ).
<b>Remarks</b>	All variables have global scope. Therefore if they are changed by a subroutine the main program and all other subroutines will see the change in the variable as well.	
<b>See Also</b>	RETURN, SUB, JUMP	
<b>Example</b>	<pre>CALL SUB1 CALL SETUP</pre>	

<b>CLEAR</b>	Clear Operator Terminal	<i>Operator Terminal</i>
<b>Purpose</b>	Clear the Operator Terminal screen. If no <i>row</i> is specified the entire screen is cleared and the cursor is moved to the upper left corner.	
<b>Syntax</b>	CLEAR [ <i>row</i> [, <i>column</i> , <i>length</i> ]]	
	CLR [ <i>row</i> [, <i>column</i> , <i>length</i> ]]	
	<i>row</i>	If only a <i>row</i> is specified, that row on the Operator Terminal screen is cleared, and the cursor is moved to the beginning of that row.
	<i>column</i> , <i>length</i>	The <i>length</i> is the number of characters to be cleared starting at the <i>row</i> , <i>column</i> . In this case, the cursor ends up in the row and column position specified.
<b>Remarks</b>		
<b>See Also</b>	PRINT, READ	
<b>Example</b>	<pre>CLEAR           ;Clear the whole Operator Terminal CLEAR 2        ;Clear row 2 on Operator Terminal CLR 2, 3, 5    ;Clear 5 characters starting at row 2, column 3</pre>	

<b>CLRPEAKS</b>	<b>Clear Peaks</b>	<i>System</i>
<b>Purpose</b>	Clears peak current command (PICMD), motor velocity (PVEL1), and following error (PFE) when set to ON.	
<b>Syntax</b>	CLRPEAKS = ON	
<b>Remarks</b>	This flag is automatically set to OFF when the variables have been cleared.	
<b>See Also</b>	PVEL1, PFE, PICMD	
<b>Example</b>	CLRPEAKS = ON	

<b>COMMENTS</b>	<b>Comments</b>	<i>Program Structure</i>
<b>Purpose</b>	Comments can be placed anywhere in a program. A comment must be preceded by a semicolon (;). Any text on the line following a semicolon is ignored by the compiler. Comments can be on the same line of a program as an instruction, or put on a separate line.	
<b>Syntax</b>	;	
<b>Remarks</b>	<p>There is no such thing as too many comments in a program. For maintainability of your program we recommend having a comment on almost every line of your program. Major functions within a program should also have comments explaining the purpose and methods used. In addition, there should be comments that identify the author, creation date, revision dates, and descriptions of changes.</p> <p>The compiler limits the number of characters per line to 100. Characters over this number are truncated. This number of characters is adequate for programming purposes, but it may prove insufficient for lengthy comments. In cases such as this, one or more additional lines composed solely of commentary may be added.</p>	
<b>See Also</b>	ASSIGN	

**Example**

```

;Any remark can follow a semicolon
;This is an example of a header that might be used.
;=====
;
;Source File Name: xxxxxxxx.xxx
;Created by: JSP
;Creation Date: June 28,1993
;Tested with IQ Master Version: vvvv
;Tested with IQ firmware Version:  vvvv
;Description: (free form description)
;
;Revisions:
;JSP          June 28, 1993(description of change)
;
;----- Inputs -----
;
;----- Outputs -----
;
;----- Optional Accessories Required-----
;Operator Terminal - NO
;Expansion I/O - NO
;Expansion memory - NO
;
;-----Begin Program -----

```



---

**COMPILEMEMORY** Compile to Memory *Compiler Options*



---

<b>Purpose</b>	Selects whether the program executable should be saved in PC memory or to a disk file.
<b>Syntax</b>	<p>COMPILEMEMORY = <i>ON/OFF</i></p> <p>ON                    The program executable information will be stored in PC memory.</p> <p>OFF                    The program executable information will be stored in a disk file.</p>
<b>Remarks</b>	The COMPILEMEMORY statement overrides the default compiler option for Compile to Memory/Compile to Disk set in the Edit menu, Compiler Options dialog box.
<b>See Also</b>	PGMTYPE, DEBUG, LISTFILE, EXPANDMACROS
<b>Example</b>	<pre>COMPILEMEMORY = ON COMPILEMEMORY = OFF</pre>

---

**CONFIG** Configuration *Miscellaneous*


---

<b>Purpose</b>	Change to Feedback Configuration menu <i>n</i> . This allows the feedback configuration to be changed within a program.
<b>Syntax</b>	<p>CONFIG = <i>n</i></p> <p><i>n</i>                    <i>n</i> specifies which of the four configurations setup in the Parameter menu is active. Values for <i>n</i> are 1 through 4. The ULTRA Plus or IQ defaults to Feedback Configuration menu # 1 on power up.</p>
<b>Remarks</b>	<div style="display: flex; align-items: flex-start;"> <div style="margin-right: 20px;"> <p style="background-color: black; color: white; padding: 2px 5px; font-weight: bold; text-align: center;">ATTENTION</p>  </div> <div style="border: 1px solid black; padding: 10px; flex-grow: 1;"> <p>Exercise caution when changing the feedback configuration while the ULTRA Plus or IQ is enabled. Unexpected motion may result from differences between the new configuration and the old configuration.</p> </div> </div> <p>The active configuration menu remains in effect after the program ends. If no change is made to this variable by the program or by a Host Language Command, the current value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Feedback Config dialog box.</p> <p>At power-up the configuration menu is always set to 1.</p>
<b>See Also</b>	
<b>Example</b>	<pre>CONFIG = 2            ;Use configuration number 2 CF = 3                ;Use configuration number 3</pre>

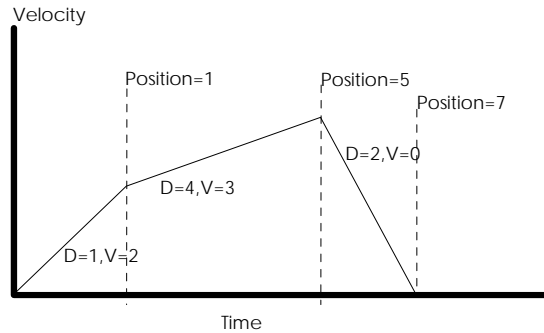
<b>CTOU1</b>	<b>Convert counts to user units</b>	<i>Conversion</i>
<b>Purpose</b>	Convert a variable from encoder counts to user units using SCALE.	
<b>Syntax</b>	$variable\_1 = CTOU1\ variable\_2$	
	<i>variable_1</i>	<i>Variable_1</i> can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable.
	<i>variable_2</i>	<i>Variable_2</i> can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable, or a constant.
<b>Remarks</b>		
<b>See Also</b>	@, UTOC1, UTOC2, CTOU2, “Conversions” on page 131	
<b>Example</b>	$V3 = CTOU1\ V4$ $G2 = CTOU1\ 1.4$	

<b>CTOU2</b>	<b>Convert counts to user units</b>	<i>Conversion</i>
<b>Purpose</b>	Convert a variable from encoder counts to user units using SCALE2.	
<b>Syntax</b>	$variable\_1 = CTOU2\ variable\_2$	
	<i>variable_1</i>	<i>Variable_1</i> can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable.
	<i>variable_2</i>	<i>Variable_2</i> can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable, or a constant.
<b>Remarks</b>		
<b>See Also</b>	@, UTOC1, UTOC2, CTOU1, “Conversions” on page 131	
<b>Example</b>	$V3 = CTOU2\ V4$ $G7 = CTOU2\ 5.6$	

D, V	D, V Segments (Sticks)		Motion
<b>Purpose</b>	The Distance, Velocity (D, V) statement defines one move segment, or stick, of a complex motion profile.		
<b>Syntax</b>	D = <i>distance</i> , V = <i>velocity</i> [, S]		
	<i>distance</i>	The incremental distance in user units to be moved, which may be a constant or a variable. Positive or negative distances may be used to determine the direction of the move. All the distances of a profile composed of sticks must be the same sign. If you need to reverse directions you need to first include a stick that decelerates to zero. User units are defined by the SCALE parameter.	
	<i>velocity</i>	The final velocity for the stick move, in user units per Time-base. The velocity is always positive and may be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> or a system variable. Negative <i>values</i> are treated as if they are positive. Direction is specified by the sign of the <i>distance</i> argument.	
	S	Optionally used to specify that an S-curve acceleration or deceleration will be used for the move. If the S-curve is specified the time to complete the stick will be the same as it would take to complete a linear stick. The peak acceleration of the S-curve stick will be twice as high as the acceleration of the equivalent linear move, but the acceleration of the motor will be much smoother than with a linear acceleration move. This is particularly useful when a flexible coupling is used.	

**Remarks**

The acceleration or deceleration for the move is calculated based on the specified distance, starting velocity, and final velocity. The final velocity of one segment become the starting velocity of the next segment. There is no limit to the number of stick moves in a profile, other than available memory, but the final move in the profile should return the velocity to 0.



The figure above represents a three segment (stick) move. The first segment has a starting velocity of 0 and final velocity of 2. This velocity is reached by the time the motor travels 1 unit of distance. The second stick starts at a velocity of 2 and accelerates to a velocity of 3 while traveling a distance of 4. The final stick starts at a velocity of 3 and decelerates to a stop while traveling a distance of 2. The instructions to make this move are:

```
D = 1, V = 2
D = 4, V = 3
D = 2, V = 0
```

The following equation can be used to calculate the acceleration that results from a stick move.

$$Accel = \frac{\langle V_f^2 - V_o^2 \rangle}{2D}$$

$V_f \equiv FinalVelocity$   
 $V_o \equiv StartingVelocity$   
 $D \equiv Distance$

The program will stop executing at this instruction line until the commanded position has gone the specified *distance*.

The DV statement can only be used in a subroutine or the main body of the program. It may not be used in Fkey, Xkey, or Scanned Event routines.

**See Also**

MOVD, MOVP, MOVV, DIF

**Example**

```
D = .5, V = 150
D = 11, V = 150
D = .5, V = 0, S
D = V1, V = V2
```

WORDS

DACn	Digital to Analog Converter	Outputs
<b>Purpose</b>	Write a value to the analog output. The value may be a variable or a constant between $\pm 10$ , with up to 2 decimal places.	
<b>Syntax</b>	<p>DACn = <i>value</i></p> <p><i>value</i>                      The <i>value</i> in volts for the DACn output. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>n</i>                              The number of the DAC output. Currently n = 1 is the only valid number.</p>	
<b>Remarks</b>	<p>DAC1 is a 12 bit DAC. With a range of <math>\pm 10</math> volts, the resolution of the output is 4.9mV.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Outputs dialog box.</p>	
<b>See Also</b>		
<b>Example</b>	<pre>DAC1 = 2.50 ;Set DAC1 to 2.5 Volts DAC1 = V3 ;Set DAC1 to the value contained in Variable V3</pre>	

DEBUG	Add Debug Information	Compiler Options
<b>Purpose</b>	Add debug information to program executable.	
<b>Syntax</b>	<p>DEBUG = <i>ON/OFF</i></p> <p>ON                              Debug information will be added.</p> <p>OFF                              Debug information will not be added.</p>	
<b>Remarks</b>	<p>The DEBUG statement overrides the default compiler option for Add Debug Information set in the Edit menu, Compiler Options dialog box. Refer to “Debug Information” on page 37 for additional information.</p> <p>This statement requires IQ Master version 3.0 or later.</p>	
<b>See Also</b>	PGMTYPE, COMPILEMEMORY, LISTFILE, EXPANDMACROS	
<b>Example</b>	<pre>DEBUG = ON DEBUG = OFF</pre>	

<b>DELAY</b>	<b>Time Delay</b>	<b>Motion</b>
<b>Purpose</b>	Delay for the specified time. Time is entered in seconds.	
<b>Syntax</b>	DELAY = <i>time</i>	
	<i>time</i>	The <i>time</i> is specified in seconds and has a maximum value of 65,535 seconds. The <i>time</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable. The resolution of <i>time</i> is equal to the position loop update time (2mS).
<b>Remarks</b>	<p>The compiler does not perform range checking for the maximum time for a DELAY statement because the value may not be known until run time (value may be a variable). However, if a program is run with a DELAY greater than 65,535 seconds, the ULTRA Plus or IQ will be disabled and flag the error as “Delay out of Range”.</p> <p>Delay times are NOT affected by the feedrate. To define a delay that IS affected by the feedrate, use the DWELL statement.</p> <p>The DELAY statement can only be used in a subroutine or the main body of the program. It may not be used in Fkey, Xkey, or Scanned Event routines.</p>	
<b>See Also</b>	DWELL, WAIT	
<b>Example</b>	<pre>DELAY = 1.5 DELAY = G1</pre>	

<b>DH</b>	<b>Define Home</b>	<b>Home</b>
<b>Purpose</b>	Define the present commanded position as the zero position. The current commanded position will become 0, the encoder that is not the feedback encoder is set to a position of zero, the Home Sequence Complete output (if assigned) and the HSEQCPL system flag will be turned ON.	
<b>Syntax</b>	DH	
<b>Remarks</b>	<p>Use caution if you use this command while there is motion. The time delay caused by execution speed will result in the zero position being defined at a position that is different than when the command was issued. The difference in position will be proportional to the velocity of the motor when the command was issued</p> <p>This instruction is not permitted in a function key program (see DHCMD).</p> <p>Any following error that is present when this instruction is executed will be maintained (the feedback position will not be set equal to the commanded position).</p> <p>The Home Sequence Complete output is assigned in the Parameter menu, Outputs dialog box.</p> <p>MOVP instructions are not permitted until the HSEQCPL system flag has been turned ON.</p>	
<b>See Also</b>	DHCMD, DP (Define Position), DPn.	
<b>Example</b>	<pre>DH           ;Define the present position as the 0, home position</pre>	

DHCMD	Define Home Command	<i>Home</i>
<b>Purpose</b>	When this write only flag is set to ON, the present commanded position will be set to zero. This flag is used in function key programs to define a home position (DH is not permitted in function key programs).	
<b>Syntax</b>	DHCMD = <i>value</i>	
	value	The <i>value</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , ON, OFF, or another system flag.
<b>Remarks</b>		
<b>See Also</b>	DH, DP	
<b>Example</b>	DHCMD = ON	

DIF	Distance, Conditional	<i>Motion</i>
<b>Purpose</b>	Move the distance specified unless the <i>condition</i> is satisfied during the move. The velocity for the move is that of the previous DV move. If the <i>condition</i> is met, the <i>action</i> specified in the instruction is taken. If the <i>condition</i> is not met during the move, the next program statement is executed when the move distance is reached. The DIF statement may not be used as the final statement in a complex profile.	
<b>Syntax</b>	DIF = <i>distance, condition, action</i>	
	<i>distance</i>	The incremental <i>distance</i> in user units to be moved, which may be a number or a variable. The sign of the distance must be the same as the sign of the preceding DV move. <i>Distance</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , a system variable, or a constant.
	<i>condition</i>	<p>The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a system or user flag being ON or OFF.</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre> I1 ON                ;an input F1 ON                ;a user flag G1 &gt; G2              ;comparison user variable G1 &gt; POS1            ;comparison system variable POS2 &lt;= 7.00        ;comparison constant </pre>

*action*

The action to be taken if the condition is TRUE. The only action that is valid for the DIF instruction is JUMP.

CAUTION: If a DV or DIF instruction is not executed quickly the motor will attempt to come to an instantaneous stop. You should avoid PRINT, READ, and CLEAR statements and only attempt to do one or two instructions before executing the next DIF or DV.

**Remarks**

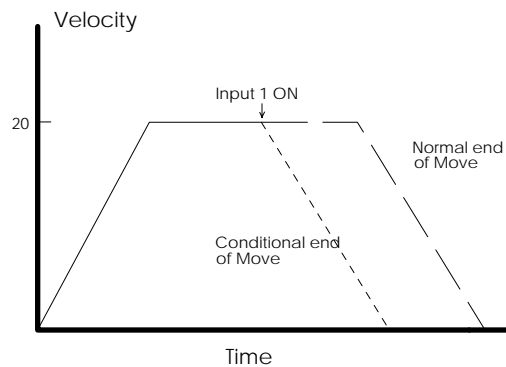
The program will stop executing at this line until the commanded position has gone the specified *distance* or the condition has been satisfied.

The DIF statement can only be used in a subroutine or the main body of the program. It may not be used in Fkey, Xkey, or Scanned Event routines.

**See Also**

DV, MOVD, MOVP, MOVV

**Example**



```

D = 1, V = 20 ;Accel to 20 in a distance of 1
DIF = 4, I1 = ON, JUMP STOPNOW ;If input 1 comes on stop now else
STOPNOW: ;stop after a distance of 4
D = 1, V = 0 ;Decel to a stop in a distance of 1
    
```

WORDS



<b>DISABLE</b>	<b>Disable Amplifier</b>	<i>Miscellaneous</i>
----------------	--------------------------	----------------------

**Purpose** DISABLE the ULTRA Plus or IQ. Once the ULTRA Plus or IQ has been disabled with the DISABLE statement in a program, an ENABLE statement must be used to re-enable the ULTRA Plus or IQ. If a motion statement is executed after the DISABLE statement, the ULTRA Plus or IQ will be “hung” trying to execute the motion profile - the motion statement cannot be executed if the ULTRA Plus or IQ is disabled.

**Syntax** DISABLE

**Remarks** The following action will take place when a DISABLE is executed. The motor may be disabled by an instruction in a program, a Host Language Command, the Enable Input, or the Run Control Dialog box.

When the DISABLE instruction (or Enable input or Disable Host Command) is executed:

- If the motor is being commanded to move, it is decelerated to a stop. The rate of deceleration is determined by the system acceleration variable, system velocity variable, and the present velocity. The deceleration rate will equal:

$$\text{System Feedrate Slew Time} = \frac{\text{System Velocity}}{\text{System Acceleration Rate}}$$

$$\text{Deceleration Rate} = \frac{\text{Current Velocity}}{\text{System Feedrate Slew Time}}$$

System Velocity is the velocity set in the Parameter menu, Velocity/Accel dialog box. System Acceleration is the acceleration set in the Parameter menu, Velocity/Accel dialog box.

- After the Feedrate Slew Time (above), power will be removed from the motor.

**NOTE:** After changing the Velocity or Acceleration parameters, a Hard Reset must be performed to allow the ULTRA Plus or IQ to recalculate the System Feedrate Slew Time.

- The Enabled output will turn OFF.
- If assigned, the In-Position output will turn OFF.
- If the Error output is assigned AND set to turn on when disabled, the Error output will turn ON.

If your program disables the ULTRA Plus or IQ and ends before re-enabling, one of the following actions must be taken to ENABLE the ULTRA Plus or IQ again.

- Cycle the power OFF and then back ON.
- HRESET, either by Host Language Commands or by switching the HRESET input if it has been assigned.
- Use the Host Language Command, ENABLE command.
- Run a system program which uses an ENABLE statement.

Turning the ENABLE input OFF and ON will not re-enable a drive that has been disabled by a program. You will not be able to run another program to re-enable the drive since an enabled drive is a requirement for starting a program.

ENABLE

**Example**

```
DISABLE
DIS
```

<b>DISLIM</b>	<b>Travel Limits Disable</b>	<b>System</b>
<b>Purpose</b>	Disables the hardware travel limits when set to ON. Enables the inputs when OFF.	
<b>Syntax</b>	DISLIM = <i>value</i> <i>variable</i> = DISLIM  <i>value</i> The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.  <i>variable</i> The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.	
<b>Remarks</b>	Regardless of the state of this flag, the limit switches, inputs 1 and 2, may be read. For example: <pre>F1 = I1                    ;read forward limit switch F2 = I2                    ;read reverse limit switch</pre> The travel limit switches must be enabled (HLIMITS = ON) for DISLIM to have any effect.  Changes made to this flag by a program are only in effect while the program is running. When the program ends it will return to the setting stored in the Personality Module.	
<b>See Also</b>	HLIMITS	
<b>Example</b>	<pre>DISLIM = ON                ;Disable the hardware travel limits DISLIM = OFF              ;Enable the hardware travel limits</pre>	

<b>DO/WHILE</b>	<b>Do/While</b>	<b>Program Structure</b>
<b>Purpose</b>	The DO <i>statement</i> WHILE <i>condition</i> executes the statement(s) between DO and WHILE repeatedly as long as the <i>condition</i> specified is TRUE. The statements are executed and then the <i>condition</i> is tested. If the <i>condition</i> is TRUE, execution jumps back to the statement following the DO. If the condition is FALSE, execution continues with the statement after the WHILE.	
<b>Syntax</b>	DO <i>statement</i> WHILE <i>condition</i>  <i>statement(s)</i> Any valid language statement. Multiple statements can be enclosed in curly braces { }.	

*condition* The *condition* to be tested. The *condition* may be a comparison, an input being ON or OFF, or a program flag being ON or OFF

Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (>), less than (<), equal to (=), not equal to (<>), less than or equal to (<=), or greater than or equal to (>=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.

```

I1 = ON           ;an input
F1 = ON           ;a user flag
INPOSN ON        ;system flag
G1 > G2           ;comparison user variable
G1 > POS1         ;comparison system variable
POS2 <= 7.00     ;comparison constant

```

**Remarks** Unlike the WHILE statement, the loop body will always be executed at least once because the DO/WHILE statement tests the *condition* AFTER the loop body is executed.

**See Also** WHILE, IF

**Example**

```

DO
{
  MOVD = G10
  DWELL = 2.5
} WHILE (I11 = ON)

```

---

## DP Home

---

**Purpose** Redefine the present commanded position as a new position. The present commanded position will become the specified *position*, the encoder position that is not the feedback encoder is set to zero, the Home Sequence Complete output (if assigned) and the HSEQCPL system flag will be turned ON.

**Syntax**  $DP = position$

*position* The *position* is specified in user units and can be a constant, a nonvolatile variable  $G_n$ , a volatile variable  $V_n$ , or a system variable. User units are defined by the SCALE parameter.

**Remarks** Use caution if you use this command while there is motion. The time delay caused by execution speed will result in the zero position being defined at a position that is different than when the command was issued. The difference in position will be proportional to the velocity of the motor when the command was issued.

Any following error that is present when this instruction is executed will be maintained (the feedback position will not be set equal to the commanded position).

The Home Sequence Complete output is assigned in the Parameter menu, Outputs dialog box.

**See Also** DH (Define Home), DPn, HOMECMD, HSEQCPL

**Example**

```

DP = -1.34           ;set the current commanded position to -1.34
DP = V3              ;set the current commanded position to the
                    ; value of V3

```

DPn	Define Position Number	<i>Home</i>												
<b>Purpose</b>	Redefine an encoder position as a new position. The present encoder position will become the specified <i>position</i> , the encoder position that is not the feedback encoder is unaffected, the Home Sequence Complete output (if assigned) and the HSEQCPL system flag will be turned ON.													
<b>Syntax</b>	<p><math>DPn = value</math></p> <p><i>n</i>                      The number of the encoder: <math>n = 1</math> or <math>2</math></p> <p><i>value</i>                      Position in user units.</p>													
<b>Remarks</b>	<p>The function performed by <i>DPn</i> is dependent on the status, used or not used, of the position loop feedback (<i>POSn</i>). The following table defines the functions:</p> <p style="text-align: center;"><b>Position Loop Feedback Status</b></p> <table border="0" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 15%;"></th> <th style="width: 40%; text-align: center;"><b>POS1 Used</b></th> <th style="width: 45%; text-align: center;"><b>POS1 Not Used</b></th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;">DP1</td> <td>Pgen is set to the new POS1_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos1 are set to (Pcmd - Fe). Pos2 is not affected.</td> <td>Pos1 is set to the new Pos1_value Pos2 is not affected</td> </tr> <tr> <td></td> <td style="text-align: center;"><b>POS2 Used</b></td> <td style="text-align: center;"><b>POS2 Not Used</b></td> </tr> <tr> <td style="vertical-align: top;">DP2</td> <td>Pgen is set to the new POS2_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos2 are set to (Pcmd - Fe). Pos1 is not affected.</td> <td>Pos2 is set to the new Pos2_value Pos1 is not affected</td> </tr> </tbody> </table> <p>Use caution if you use this command while there is motion. The time delay caused by execution speed will result in the zero position being defined at a position that is different than when the command was issued. The difference in position will be proportional to the velocity of the motor when the command was issued.</p> <p>Any following error that is present when this instruction is executed will be maintained (the feedback position will not be set equal to the commanded position).</p> <p>The Home Sequence Complete output is assigned in the Parameter menu, Outputs dialog box.</p>			<b>POS1 Used</b>	<b>POS1 Not Used</b>	DP1	Pgen is set to the new POS1_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos1 are set to (Pcmd - Fe). Pos2 is not affected.	Pos1 is set to the new Pos1_value Pos2 is not affected		<b>POS2 Used</b>	<b>POS2 Not Used</b>	DP2	Pgen is set to the new POS2_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos2 are set to (Pcmd - Fe). Pos1 is not affected.	Pos2 is set to the new Pos2_value Pos1 is not affected
	<b>POS1 Used</b>	<b>POS1 Not Used</b>												
DP1	Pgen is set to the new POS1_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos1 are set to (Pcmd - Fe). Pos2 is not affected.	Pos1 is set to the new Pos1_value Pos2 is not affected												
	<b>POS2 Used</b>	<b>POS2 Not Used</b>												
DP2	Pgen is set to the new POS2_value, Pjog, Gpos, Pcam, and Pext are set to zero, and Posn and Pos2 are set to (Pcmd - Fe). Pos1 is not affected.	Pos2 is set to the new Pos2_value Pos1 is not affected												
<b>See Also</b>	DH (Define Home), HOMECMD, HSEQCPL, POS1, POS2, POSN													
<b>Example</b>	<pre>DP1 = -1.34                      ;define encoder 1 position (POS1) as -1.34</pre>													

<b>DWELL</b>	<b>Dwell for time</b>	<i>Motion</i>
<b>Purpose</b>	Dwell (wait) for the specified time.	
<b>Syntax</b>	DWELL = <i>time</i>  <i>time</i> The <i>time</i> is in seconds and can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.	
<b>Remarks</b>	Dwell times ARE affected by the Feedrate. For example, if a DWELL = 1 statement is used in a program, the dwell time will be for 1 second as long as the Feedrate is 100%. If the feedrate is changed to 200%, the dwell time will take 0.5 seconds. $\text{Actual Delay} = \text{Dwell} / \text{Feedrate.}$ For more details on feedrate, refer to the feedrate command, FDR. To define a delay that is NOT affected by the feedrate, use the DELAY statement. The DWELL statement can only be used in a subroutine or the main body of the program. It may not be used in Fkey, Xkey, or Scanned Event routines. The maximum time for a DWELL statement is 65 seconds. If this value is exceeded, a Profile Calculation error will occur.	
<b>See Also</b>	DELAY, WAIT	
<b>Example</b>	DWELL = 0.25 DWL = V2	

<b>EFLAG</b>	<b>Error Flag</b>	<i>System</i>
<b>Purpose</b>	Flag to indicate if an error is active. ON indicates there is a fault	
<b>Syntax</b>	<i>variable</i> = EFLAG  <i>variable</i> <i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> or another system flag.	
<b>Remarks</b>		
<b>See Also</b>	ENUM, WFLAG, WNUM, SYSERROR, SYSWARN	
<b>Example</b>	IF EFLAG = ON ...	

<b>ENABLE</b>	<b>Enable Amplifier</b>	<i>Miscellaneous</i>
<b>Purpose</b>	Enable the ULTRA Plus or IQ. The ENABLE statement is used to enable the ULTRA Plus or IQ after a DISABLE statement has been used in a program.	
<b>Syntax</b>	ENABLE ENA	
<b>Remarks</b>	<p>The following action will take place when a ENABLE is executed. The ENABLE may come from an instruction in a program, a Host Language Command, the ENABLE Input or the Run Control dialog box. When the ENABLE instruction is executed:</p> <ul style="list-style-type: none"> <li>• The command position is set equal to the feedback position.</li> <li>• The Enabled output is turned ON.</li> <li>• Power is applied to the motor; it will hold the present position.</li> </ul>	
<b>See Also</b>	DISABLE	
<b>Example</b>	ENABLE ENA	

<b>END</b>	<b>End Program Statement</b>	<i>Program Structure</i>
<b>Purpose</b>	Signals the end of a program. The END statements goes at the very end of the program file after the main body and any subroutines.	
<b>Syntax</b>	END	
<b>Remarks</b>	The END statement is not required.	
<b>See Also</b>		
<b>Example</b>	END	

<b>ENUM</b>	<b>Error Number</b>	<i>System</i>
<b>Purpose</b>	Currently active error number.	
<b>Syntax</b>	<i>variable</i> = ENUM	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
<b>Remarks</b>	ENUM contains a value only while an error is active. Once the error is cleared ENUM is set to zero. Refer to the appendix “Error Messages” on page 418 for a list of errors.	
<b>See Also</b>	EFLAG, WFLAG, WNUM, SYSERROR, SYSWARN	
<b>Example</b>	V1 = ENUM	

<b>ERET</b>	<b>Ereturn Flag</b>	<i>System</i>
<b>Purpose</b>	The ERET flag runs system program 26 (ERETURN).	
<b>Syntax</b>	ERET = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>		
<b>See Also</b>	Appendix A, System Programs	
<b>Example</b>	ERET = ON	

<b>ERETPOS</b>	<b>Emergency Return Position</b>	<i>Position</i>
<b>Purpose</b>	Set the ERETURN position. ERETURN position specifies the position the system moves to when the Emergency Return input is activated (if assigned).	
<b>Syntax</b>	ERETPOS = <i>position</i> <i>variable</i> = ERETPOS  EPOS = <i>position</i> <i>variable</i> = EPOS  <i>position</i> The <i>position</i> is entered in user units. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.  <i>variable</i> The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, System dialog box.	
<b>See Also</b>	ERET	
<b>Example</b>	ERETPOS = 2.5 EPOS = V2	

<b>EXPANDMACROS</b>	<b>Expand Assign Statements in List File</b>	<i>Compiler Options</i>
<b>Purpose</b>	Expand Assign statement information in List files.	
<b>Syntax</b>	EXPANDMACROS = <i>ON/OFF</i>  ON                              Assign statements will be expanded in the list file.  OFF                              Assign statements will not be expanded in the list file.	
<b>Remarks</b>	The EXPANDMACROS statement overrides the default compiler option for Expand Macros set in the Edit menu, Compiler Options dialog box. This setting is ignored if Compile to Memory is selected (no list file will be created).	
<b>See Also</b>	PGMTYPE, DEBUG, COMPILETOMEMORY, LISTFILE	
<b>Example</b>	EXPANDMACROS = ON EXPANDMACROS = OFF	



<b><i>F<sub>n</sub></i></b>	<b>Volatile User Flag</b>	<b><i>Variable</i></b>
<b>Purpose</b>	Set a user flag ON or OFF with another flag, or a constant.	
<b>Syntax</b>	$F_n = ON/OFF/F_n/B_m$ $variable = F_n$	
	<i>n, m</i>	The number of the flag. $n = 1, 2, 3, \dots, 64$ . $m = 1, 2, 3, \dots, 8$
	<i>variable</i>	<i>Variable</i> may be a nonvolatile flag $B_n$ , a volatile flag $F_n$ , or a system flag.
<b>Remarks</b>	<p>The changes made to <math>F_n</math> remain in effect as long as power is maintained. On power-up or HRESET all <math>F_n</math> flags are set to OFF. If a flag needs to be maintained during a power failure use a <math>B_n</math> flag. Changes made to <math>F_n</math> flags by one program will be seen by other programs.</p> <p><b>TIP:</b> Variables can be assigned names using the ASSIGN command and then referenced by those names to make the program easier to read.</p>	
<b>See Also</b>	$G_n, B_n, V_n$	
<b>Example</b>	<pre>F1 = ON F2 = OFF IF F1 ON MOVD = 5</pre>	

FDR	Feedrate	<i>System</i>
<b>Purpose</b>	Set the feedrate value. Feedrate can be changed at any time in the program. Feedrate scales the Timebase for motion. With the Feedrate set at 100%, all velocities and dwells are at the programmed rates. Feedrate can be set to less than 100% to slow down a process, or above 100% to speed it up.	
<b>Syntax</b>	<p>FDR = <i>value</i>  <i>variable</i> = FDR</p> <p><i>value</i>                      Feedrate is entered in percent (the percent sign is not needed) with a range of 0 to 200. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                  The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i> or a system variable.</p>	
<b>Remarks</b>	<p>Feedrate will affect MOVD, MOVF, MOVV, DIF, DV, and DWELL instructions. The velocity, acceleration and time in these functions will be scaled by the FDR percentage.</p> <p>For example, if the velocity is set to 3,000 RPM and the FDR is set to 50, any motion will have a velocity that is 50% of 3,000 RPM or 1,500 RPM (3000 * 0.5 = 1500). A DWELL time is also scaled by FDR. A DWELL of 10 seconds would be scaled to 20 seconds (10 / 0.50 = 20).</p> <p>A feedrate can also be set with the ADC1 analog input. This is accomplished by assigning the analog input to feedrate in the Parameter menu, Inputs dialog box. A common use of this is to connect a potentiometer to this analog input and permit the operator to set the speed of operation. If the ADC1 input is used as feedrate input, the ADC1 input voltage can range from 0 to 10 volts, corresponding to 0 to 200% feedrate.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.</p>	
<b>See Also</b>	RFDR	
<b>Example</b>	<pre>FDR = 100           ;set feed rate to 100% FDR = G4 V2 = FDR</pre>	

<b>FE</b>	<b>Following Error</b>	<i>Position</i>
<b>Purpose</b>	Following error is the difference between the commanded position and the actual position.	
<b>Syntax</b>	$variable = FE$  <i>variable</i> Following error in user units. User units are defined by the SCALE parameter. The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.	
<b>Remarks</b>		
<b>See Also</b>	PFE, FEL, FET	
<b>Example</b>	V3 = FE	

<b>FEL</b>	<b>Following Error Limit</b>	<i>System</i>
<b>Purpose</b>	Changes the Following Error Limit (FEL) used in the program. Following Error is the absolute value of the difference between the commanded position and the actual position. If the Following Error is exceeds the FEL for a time greater than the Following Error Time (FET), a fault occurs and system operation halts. The drive is disabled and the Error system program is run (see Appendix A).	
<b>Syntax</b>	$FEL = value$ $variable = FEL$  <i>value</i> The <i>value</i> is entered in user units. User units are defined by the SCALE parameter. The <i>value</i> can be a constant, a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.  <i>variable</i> The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	FET, FE, PFE	
<b>Example</b>	<pre>FEL = 0.5      ;Set the following error limit to 0.5 user units FEL = V7 V2 = FEL</pre>	

<b>FET</b>	<b>Following Error Time</b>	<i>System</i>
<b>Purpose</b>	Change the Following Error Time limit used in the program. The Following Error Time limit is the length of time that the following error may exceed the Following Error Limit without causing a fault. The following error is checked every position loop update (2mS).	
<b>Syntax</b>	<p>FET = <i>value</i>  <i>variable</i> = FET</p> <p><i>value</i>                      The <i>value</i> is entered in seconds with a resolution of 2mS. The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                    The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	FEL, FE, PFE	
<b>Example</b>	<pre>FET = 1.0 V3 = FET</pre>	

<b>FGAIN</b>	<b>Acceleration Feedforward Gain</b>	<i>Gain</i>
<b>Purpose</b>	Set the acceleration feedforward gain.	
<b>Syntax</b>	<p>FGAIN = <i>value</i>  <i>variable</i> = FGAIN</p> <p><i>value</i>                      The value is entered in percent of command velocity and may be changed anywhere in a program. Value can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                    The variable may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	KFF, IG, PG	
<b>Example</b>	<pre>FGAIN = 10 FG = V9 V4 = FGAIN</pre>	

<b><i>FIn</i></b>	<b>Interrupt Flag</b>	<b><i>Interrupts</i></b>
<b>Purpose</b>	Interrupt <i>n</i> flag. These flags are turned ON automatically by a transition on the interrupt <i>n</i> pin from OFF to ON. Interrupt 1 is on input 11 and interrupt 2 is on input 12.	
<b>Syntax</b>	$FIn = OFF$ $variable = FIn$	
	<i>n</i>	The number of the interrupt $n = 1$ or $2$ .
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	These flags are not automatically cleared when read, so before checking for an interrupt again, use the $FIn = OFF$ statement. These flags are set to OFF when the program ends.	
<b>See Also</b>	$INTn$ , $InPm$	
<b>Example</b>	$F14 = FI1$ $FI2 = OFF$	

<b><i>FIDXn</i></b>	<b>Encoder <i>n</i> index interrupt</b>	<b><i>Interrupts</i></b>
<b>Purpose</b>	Encoder <i>n</i> index interrupt received. These flags are set to ON by an encoder interrupt.	
<b>Syntax</b>	$FIDXn = OFF$ $variable = FIDXn$	
	<i>n</i>	The number of the encoder that is generating the interrupt: $n = 1$ or $2$ .
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	These flags are not automatically cleared when read, so before checking for an interrupt again, use the $FIDX1 OFF$ statement. These flags are set to OFF when the program ends.	
<b>See Also</b>	$IDXn$ , $IDXnPm$	
<b>Example</b>	<pre> ;Use an interrupt to stop quickly following an index FIDX1 = OFF                                ;clear the interrupt D = 0.25, V = 10                            ;start the move DIF = 3, FIDX1 = ON, JUMP StopWhenMark      ;conditional move StopWhenMark: D = .01, V = 0                               ;Stopping move </pre>	

<b>FILTER</b>	<b>Velocity Loop Filter</b>	<b>Gain</b>
<b>Purpose</b>	Set the current command filter. FILTER is the bandwidth in hertz (Hz) of the low pass filter on the output of the velocity regulator (the current command). Reducing the value of FILTER will smooth the current command which reduces noise from high frequency current pulsation.	
<b>Syntax</b>	<pre>FILTER = value variable = FILTER</pre> <p><i>value</i>                      The low pass filter roll off frequency in hertz (Hz). The maximum value is 300 Hz. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                    The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog	
<b>See Also</b>		
<b>Example</b>	<pre>FILTER = 100                    ;set low pass filter roll off to 100Hz</pre>	

<b>FNACTIVE</b>	<b>Fkey active Flag</b>	<b>System</b>
<b>Purpose</b>	Fkey pressed flag.	
<b>Syntax</b>	<pre>variable = FNACTIVE</pre> <p><i>variable</i>                    If TRUE it indicates that an Fkey is pressed on the operator terminal. <i>Variable</i> may be a nonvolatile flag <i>Bn</i>, a volatile flag <i>Fn</i>, or another system flag.</p>	
<b>Remarks</b>	The flag is ON when an Fkey is pressed and OFF otherwise. A common use for this flag is to permit something to happen as long as the Fkey is pressed.	
<b>See Also</b>	XNACTIVE	
<b>Example</b>	<pre>B2 = FNACTIVE IF FNACTIVE = ON JOGF = ON ELSE JOGF = OFF</pre>	

**FNPGM****Function Program, Running****Program Structure**

**Purpose** Starts an Fkey program. Fkey programs are system programs 1 through 24.

**Syntax** FNPGM = *number*

*number* *Number* specifies which function key program will be run. *number* = 1, 2, 3, ..., 24.

No.	Name	Function
1	Jog Rev	;Simulates Jog Reverse input
2	Jog Fwd	;Simulates Jog Forward input
3	Start	;Simulates Start input
4	Stop	;Stops program execution and motion
5	Run	;Asks for program # to run
6	Home Command	;Starts Home program
7	Def Home;	;Defines present position as Home
8	Set FDR	;Set the Feedrate value
9	Blank	
10	Blank	
11	Blank	
12	Blank	
13	Mon VEL1;	;Display VEL1 on Operator Terminal
14	Mon POSN;	;Display POSN on Operator Terminal
15	Mon PCMD	;Display PCMD on Operator Terminal
16	Mon FE	;Display FE on Operator Terminal
17	Mon ICMD	;Display ICMD on Operator Terminal
18	Mon IAVE	;Display IAVE on Operator Terminal
19	Mon FDR	;Display FDR on Operator Terminal
20	Mon ADC1	;Display ADC1 on Operator Terminal
21	Blank	
22	Blank	
23	Blank	
24	Blank	

**Remarks** A complete description of each system program is provided in Appendix A. Fkey programs, like other system programs may be changed by the user. The factory default programs are shown above.

Only one Fkey program can be running at a time. If an Fkey program is running when the FNPGM statement is executed, the current Fkey program ends and the new Fkey program is started.

The program statements in an Fkey routine can be any legal program statement except motion statements and subroutine calls. Motion statements include MOVD, MOVP, DV, DIF, DELAY, and DWELL.

**See Also** Operator Terminal; Appendix A, System Programs

**Example**  
 FNPGM = 1  
 FNPGM = 14

<b>FOUNDHOME</b>	<b>Found Home Flag</b>	<i>System</i>
<b>Purpose</b>	This write only flag is used in a program to signal that the home cycle has been completed.	
<b>Syntax</b>	FOUNDHOME = <i>variable</i>	
	<i>variable</i>	<i>Variable</i> may be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This flag is normally used in the Home Program (System Program 25).	
<b>See Also</b>	Appendix A, System Program 25	
<b>Example</b>	FOUNDHOME = ON FOUNDHOME = OFF FOUNDHOME = F1	

<b>FVEL1</b>	<b>Filtered Motor Velocity</b>	<i>Velocity</i>
<b>Purpose</b>	Filtered velocity feedback from encoder 1 in user units per Timebase.	
<b>Syntax</b>	<i>variable</i> = FVEL1	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	The velocity of encoder 1 is determined every velocity loop update (1mS). The period of the filter is determined by the value set for Velocity Monitor Filter, set in the Monitor menu, Variable Monitor Set Up dialog box.	
<b>See Also</b>	VEL1, VCMD, VEL2, PVEL1	
<b>Example</b>	V1 = FVEL1	



<b>Gn</b>	Variable, Nonvolatile, User	Variable
<b>Purpose</b>	Load a user-defined variable with a value from another variable, result of a math expression, a constant, a timer, or an analog input.	
<b>Syntax</b>	$Gn = value$ $variable = Gn$	
	<i>n</i>	The number of the variable. $n = 1, 2, 3, \dots, 64$ .
	<i>value</i>	<i>Value</i> can be a constant, a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable
<b>Remarks</b>	<p>G and V variables are stored internally as 4 bytes for the mantissa (the part before the decimal point), 2 bytes for a numerator (the top part of a fraction), and 2 bytes for a denominator (the bottom part of a fraction). The range of numbers that can be represented in 4 bytes is <math>\pm 2,147,483,648</math>. The range of numbers that can be represented in 2 bytes is <math>\pm 32,767</math>. Therefore, the range of numbers that can be contained in a G or V variable and any intermediate value is:</p> $G \text{ or } V = \pm 2,147,483,648 + \frac{\pm 32,767}{\pm 32,767}$ <p>The changes made to <math>Gn</math> remain in effect even after power is removed. Changes made to a <math>Gn</math> by one program will be seen by other programs. <math>Vn</math> variables are volatile and will lose their values when power is removed</p> <p><b>TIP:</b> Variables can be assigned names using the ASSIGN command and then referenced by those names to make the program easier to read.</p>	
<b>See Also</b>	$Vn, Fn, Bn$ , Part 4 • Programming, Arithmetic	
<b>Example</b>	$G7 = ADC1$ $G14 = PCMD$ $G7 = V7 + 1$ $G8 = TIMER1$ $G9 = POS1$	

<b>GEAR</b>	Gear	<i>Motion</i>
<b>Purpose</b>	The gear statement is used to set and change the pulse to pulse ratio for electronic gearing and step and direction following. If the gear is enabled, the ratio is changed immediately to the specified ratio. If the gear is not enabled, the value is set but will not affect motion.	
<b>Syntax</b>	<p>GEAR = <i>value</i></p> <p><i>value</i></p> <p>The gear ratio can be set as a ratio of two integers, two variables, a single variable, a combination of a variable and a constant, or as a decimal number. When a ratio of two integers is entered, the numerator can be:</p> $\text{Gear} = \frac{0 \text{ to } \pm 32,767}{1 \text{ to } \pm 32,767}$ <p>The <i>value</i> can be a constant, a nonvolatile variable <math>G_n</math>, a volatile variable <math>V_n</math>, or a system variable.</p>	
<b>Remarks</b>	<p>If the <i>value</i> is not within the range of the numerator or denominator the GEAR value is not changed. It should be noted that the range of <i>value</i> is different than the range of <math>G_n</math> and <math>V_n</math> variables.</p> <p>The current Feedback Configuration must have the Gear Input set to either POS2 or POS3 for the gear statements to have any effect on motion.</p> <p>Other motion can be “added” to gear. For example: If gear is enabled, and a MOVD = 1 statement is executed, the follower will advance 1 user unit relative to the master.</p> <p>Note: The Gear ratio is not affected by the Scale or Scale2 parameters.</p>	
<b>See Also</b>	GEAREN, SLEW, SLEWEN	
<b>Example</b>	<pre>GEAR = 1 GEAR = V3 GEAR = 5/2 GEAR = G1/5 GEAR = G1/G2</pre>	

<b>GEAREN</b>	<b>Gear Enable</b>	<i>Motion</i>
<b>Purpose</b>	Enable electronic gearing. Enable the follower motor to begin following the gear input source at the ratio specified by the last GEAR instruction.	
<b>Syntax</b>	GEAREN = <i>variable</i> GEAREN = <i>ON/OFF</i>	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>ON</i>	Enable electronic gearing.
	<i>OFF</i>	Disable electronic gearing.
<b>Remarks</b>		
<b>See Also</b>	GEAR, SLEW, SLEWEN	
<b>Example</b>	GEAREN = ON GEAREN = OFF GEAREN = F1	

<b>GPOS</b>	<b>Gear Position</b>	<i>System</i>
<b>Purpose</b>	External position command (from gear or step and direction) in user units before any slew is applied to the command.	
<b>Syntax</b>	<i>variable</i> = GPOS	
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>		
<b>See Also</b>	PEXT	
<b>Example</b>	G1 = GPOS V1 = GPOS	

<b>HLIMITS</b>	<b>Travel Limits Enabled Flag</b>	<i>System</i>
<b>Purpose</b>	Set if travel limit inputs are enabled.	
<b>Syntax</b>	<i>variable</i> = HLIMITS	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>		
<b>See Also</b>	DISLIM	
<b>Example</b>	<pre>F4 = HLIMITS IF HLIMITS = ON ...</pre>	
<b>HOFFS</b>	<b>Home Offset</b>	<i>Home</i>
<b>Purpose</b>	Set the Home Offset. Home Offset specifies the distance from the encoder index, or home switch if there is no index, to the home position. The sign specifies the direction of the offset.	
<b>Syntax</b>	HOFFS = <i>value</i> <i>variable</i> = HOFFS	
	<i>value</i>	The Home Offset is entered in user units. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	<p>The default home program finds the home position based on the home switch and/or the encoder index. Once this position is found, an incremental move is made with a distance of HOFFS. Home position (zero) is then defined. A detailed description of the home program is in Appendix A.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.</p>	
<b>See Also</b>	DP, DH, Appendix A, System Programs	
<b>Example</b>	<pre>HOFFS = 1.5 V2 = HOFFS</pre>	

<b>HOME CMD</b>	<b>Home Command</b>	<i>Home</i>
<b>Purpose</b>	Runs the home program when set to ON.	
<b>Syntax</b>	HOME CMD = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag. Setting this flag to ON causes the Home Program (System Program number 25) to run.
<b>Remarks</b>	This flag can only be used in Fkey programs.	
<b>See Also</b>	DH, DP, Appendix A	
<b>Example</b>	HOME CMD = ON	
<b>HRESET</b>	<b>Hardware Reset</b>	<i>System</i>
<b>Purpose</b>	Causes a hardware reset when set to ON. A hardware reset simulates turning power OFF and ON.	
<b>Syntax</b>	HRESET = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag. A value of OFF has no effect.
<b>Remarks</b>	Setting this flag does not cause a hardware reset to occur immediately. A small number of instructions may be executed after this flag is set to ON before the reset occurs. If it is critical in the application that no instructions execute after the HRESET, add a statement to force the program to "stall" until the hard reset occurs. For example, WAIT I = 2.	
<b>See Also</b>	SRESET	
<b>Example</b>	HRESET = ON	
<b>HSEQCPL</b>	<b>Home Sequence Complete</b>	<i>Home</i>
<b>Purpose</b>	Home sequence complete. This flag indicates that a home sequence has been completed.	
<b>Syntax</b>	<i>variable</i> = HSEQCPL	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This flag is turned on when a Define Home (DH) or a Define Position (DP or DPn) instruction is executed, or when the home sequence has been completed.	
<b>See Also</b>	DH, DP, DPn, HOME CMD, FOUNDHOME	
<b>Example</b>	F4 = HSEQCPL IF HSEQCPL °	

<b>HSWEN</b>	Home Switch Enable	<i>Home</i>
<b>Purpose</b>	Set if Home Switch input is enabled.	
<b>Syntax</b>	<i>variable</i> = HSWEN	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This flag is provided so that you can configure the standard home program (system program 25) to your specific needs. This flag informs the home program that a home switch is present. HSWEN is set in the Parameter menu, Inputs dialog box.	
<b>See Also</b>	INDEXEN, Appendix A, System Programs	
<b>Example</b>	<pre>F7 = HSWEN IF HSWEN = ON °</pre>	

<b>HSWSTAT</b>	Home Switch Status	<i>Home</i>
<b>Purpose</b>	Present status of the Home Switch input. This flag reports the home input status as active or inactive. It does NOT report if the input is ON or OFF. The Home Switch Active States parameter in the Parameter menu, Inputs dialog box determines if HSWSTAT is active when the home switch input is ON or OFF.	
<b>Syntax</b>	<i>variable</i> = HSWSTAT	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This flag is provided primarily for use with the standard home routine. If a home program was written for a specific machine, the program could test for I5 ON or OFF rather than test this flag.	
<b>See Also</b>	Appendix A, System Programs	
<b>Example</b>	<pre>IF HSWSTAT = ON . . . F3 = HSWSTAT</pre>	

<b>HVEL</b>	Home Velocity	<i>Home</i>
<b>Purpose</b>	Set the velocity used by the Home program.	
<b>Syntax</b>	HVEL = <i>value</i> variable = HVEL	
	<i>value</i>	The velocity in user units per user Timebase. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.	
<b>See Also</b>	JVVEL, VEL, Appendix A, System Programs	
<b>Example</b>	HVEL = 10 G4 = HVEL	

<b>Im</b>	Input	<i>Inputs</i>
<b>Purpose</b>	Read or test a digital input.	
<b>Syntax</b>	variable = <i>Im</i>	
	<i>m</i>	The number of the input: <i>m</i> = 1, 2, ...,16. If the controller is equipped with an I/O Expansion card or a Memory and I/O Expansion card, <i>m</i> = 1, 2, ...,48.
<b>Remarks</b>		
<b>See Also</b>	<i>Om</i>	
<b>Example</b>	F1 = I1 IF I1 = ON ... WAIT I1 = ON	

<b><i>InPm</i></b>	<b>Position, Latched</b>	<b><i>Interrupts</i></b>
<b>Purpose</b>	Position of encoder <i>m</i> latched by interrupt <i>n</i> in user units.	
<b>Syntax</b>	<i>variable</i> = <i>InPm</i>	
	<i>n</i>	Interrupt number: <i>n</i> = 1 or 2.
	<i>m</i>	Encoder number: <i>m</i> = 1 or 2.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	When interrupt <i>n</i> occurs the position of encoder <i>m</i> is captured and stored in the variable <i>InPm</i> .	
<b>See Also</b>	INT <i>n</i> , FI <i>n</i>	
<b>Example</b>	<pre>V1 = I1P2 G4 = I2P1 - I1P1</pre>	

<b><i>IAVE</i></b>	<b>Current, Average</b>	<b><i>Current</i></b>
<b>Purpose</b>	Average current command in amperes.	
<b>Syntax</b>	<i>variable</i> = IAVE	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	The variable IAVE contains the value equal to the average commanded current to the motor. The variable is scaled by the drive module current ( <i>Dm</i> ).	
<b>See Also</b>	PICMD, ICMD, IAVG, ILIMIT	
<b>Example</b>	V1 = IAVE	



<b>I AVG</b>	<b>Average Current</b>	<i>Current</i>
<b>Purpose</b>	Average current set point. This set point determines the threshold for an average current fault.	
<b>Syntax</b>	I AVG = <i>value</i> <i>variable</i> = I AVG	
	<i>value</i>	<i>Value</i> can be a constant, a global variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
	<i>variable</i>	<i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	I AVE, I LIMIT, PICMD, ICMD	
<b>Example</b>	I AVG = 3.5 V1 = I AVG	

<b>ICMD</b>	<b>Current, Command</b>	<i>Current</i>
<b>Purpose</b>	Instantaneous current command in amperes.	
<b>Syntax</b>	<i>variable</i> = ICMD	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	The variable ICMD contains the value of the instantaneous current command to the motor. The variable is scaled by the drive module current ( <i>Dm</i> ).	
<b>See Also</b>	I AVE, PICMD, I AVG, I LIMIT	
<b>Example</b>	V3 = ICMD IF ICMD > 10.3 °	

<b>IDX<math>n</math> ON/OFF</b>	<b>Index Interrupt Enable/Disable</b>	<b>Interrupts</b>
<b>Purpose</b>	Enable or disable encoder $n$ index interrupt.	
<b>Syntax</b>	IDX $n$ ON/OFF	
	$n$	The number of the encoder: $n = 1$ or $2$ .
	ON	Enable the index interrupt.
	OFF	Disable the index interrupt.
<b>Remarks</b>	<p>After an IDX<math>n</math> ON is executed, one interrupt will be detected and the position of encoder 1 and encoder 2 will be saved into IX<math>n</math>P1 and IX<math>n</math>P2. The FIDX<math>n</math> flag is then set, and the interrupt is disabled.</p> <p>The flag remains set until it is cleared with an FIDX<math>n</math> OFF statement. If the state of an interrupt flag is not known when enabling the interrupt, clear the flag with an FIDX<math>n</math> OFF statement before enabling the interrupt. When a program stops, the interrupt flags are cleared.</p>	
<b>See Also</b>	FIDX $n$ , IX $n$ P $m$	
<b>Example</b>	<pre>IDX1 ON IDX2 OFF</pre>	

<b>IF</b>	<b>If/Then/Else</b>	<b>Program Structure</b>
<b>Purpose</b>	The IF statement tests for a condition and then executes the specified action if the condition is satisfied. If the condition is false, no action is taken and the following instruction is executed. Optionally, a second action may be specified to be executed if the condition is false.	
<b>Syntax</b>	IF <i>condition</i> <i>action1</i> [ELSE <i>action2</i> ]	
	<i>condition</i>	<p>The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a program flag being ON or OFF</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre>I1 ON                ;an input F1 ON                ;a user flag INPOSN ON           ;system flag G1 &gt; G2              ;comparison user variable G1 &gt; POS1            ;comparison system variable POS2 &lt;= 7.00        ;comparison constant</pre>
	<i>action1</i>	The action to be taken if the condition is TRUE. The action may be any programming statement or block of statements. For example, turning an output or program flag ON or OFF, writing a value to the DAC1, jumping to a program label, or an addition or subtraction. A block of statements is any group of statements enclosed in curly braces {}.

*action2* The action to be taken if the condition is FALSE. The action may be any programming statement. Multiple statements may be enclosed in curly braces {}.

**Remarks** Only *action1* or *action2* will happen. It is impossible for both actions to take place.

**See Also** WHILE, DO/WHILE

**Example**

```

IF POSN>5 O2 = ON ELSE O2 = OFF
IF F3 = ON JUMP BEGIN
IF (V1 <> 10) V1 = V1 + 1
IF I12 = ON CLEAR
IF F7 = OFF CALL SUB1
IF G1 < = 0 READ G1
IF I1 = ON                                ;IF input 1 on move 1 and turn on output 1
    {
    MOVD = 1
    O1 = ON
    }
ELSE                                       ;Else move distance 2 and turn on output 2
    {
    MOVD = 2
    O2 = ON
    }

```

---

<b>IGAIN</b>	<b>Integral Gain, Velocity Loop</b>	<i>Gain</i>
--------------	-------------------------------------	-------------

---

**Purpose** Set the integral gain for the velocity regulator. IGAIN can be changed at any time in the program.

**Syntax** IGAIN = *value*  
*variable* = IGAIN

*value* The *value* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

*variable* The *variable* may be a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

**Remarks** Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.

**See Also** PGAIN, FGAIN

**Example** IGAIN = 0  
 IG = V10

<b>ILIMIT</b>	<b>Current Limit</b>	<b>Current</b>
<b>Purpose</b>	Set the current limit used by the ULTRA Plus or IQ. The average and peak current will be held below ILIMIT.	
<b>Syntax</b>	ILIMIT = <i>value</i> <i>variable</i> = ILIMIT  <i>value</i> Specifies the value in Amperes that the present limit is set to. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.  <i>variable</i> Sets the variable equal to the current setting of the current limit. <i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	IAVG, IAVE, PICMD, ICMD	
<b>Example</b>	<pre>ILIMIT = 10 IL = G5</pre>	

<b>INDEXEN</b>	<b>Index Enabled</b>	<b>Home</b>
<b>Purpose</b>	This flag is set to ON to indicate that the standard home program should home to the encoder index mark.	
<b>Syntax</b>	<i>variable</i> = INDEXEN  <i>variable</i> The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.	
<b>Remarks</b>	This flag is provided so that you can configure the standard home program (system program 25) to your specific needs. This flag is set in the Parameter menu, Inputs dialog box.	
<b>See Also</b>	HSWEN, Appendix A, System Programs	
<b>Example</b>		

<b>INPOSN</b>	<b>In-Position</b>	<i>Position</i>
<b>Purpose</b>	In-Position flag. This flag indicates that the feedback position has been within the In-Position window of the commanded position for the time set by the window time parameter. This flag is valid during motion and while stopped, but only if the drive is enabled.	
<b>Syntax</b>	$variable = INPOSN$  <i>variable</i> The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.	
<b>Remarks</b>	<p>If the amplifier is disabled, the INPOSN flag will be OFF.</p> <p>In-Position Mode determines operation of the In-Position flag. If Relative is selected, the In-Position flag will be activated whenever the system is within the In-Position Window, even if it is in motion. If Absolute is selected, the In-Position flag will be turned on only when the system is within the In-Position Window and not in motion (default setting is Absolute). In-Position mode is set in the Parameter menu, Outputs dialog box.</p> <p>If assigned, the INPOSN output (O7) will turn ON when the INPOSN flag is ON and OFF whenever the flag is OFF.</p>	
<b>See Also</b>	WINDOW	
<b>Example</b>	<pre>F3 = INPOSN WAIT INPOSN = ON      ;Wait until in position. O1 = ON                ;Then turn ON output 1</pre>	

<b>INT</b>	<b>Integer</b>	<i>Math</i>
<b>Purpose</b>	Strip off any fractional value of <i>Variable_1</i> and put only the integer part of <i>Variable_2</i> into <i>Variable_1</i> .	
<b>Syntax</b>	$Variable_1 = INT Variable_2$  <i>Variable_1</i> The <i>variable_1</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.  <i>Variable_2</i> Same as <i>Variable_1</i> .	
<b>Remarks</b>	If the value to be truncated is negative, the result will be rounded down to the next lower (more negative) number.	
<b>See Also</b>	<i>Gn</i> , <i>Vn</i>	
<b>Example</b>	<pre>V3 = INT V4 G24 = INT V2 G24 = INT G24</pre>	

<b>INT<math>n</math> ON/OFF/CONT</b>	Interrupt, ON/OFF/CONT	<i>Interrupts</i>
--------------------------------------	------------------------	-------------------

<b>Purpose</b>	Enable, disable, or continuously enable interrupt $n$ .
<b>Syntax</b>	<p><math>INTn</math> <i>ON/OFF/CONT</i></p> <p><math>n</math> Specifies the number of the input interrupt: <math>n = 1</math> or <math>2</math>. Interrupt 1 is on input I1 and interrupt 2 is on input I2.</p> <p><i>ON</i> Enable the interrupt for a single interrupt.</p> <p><i>OFF</i> Disable the interrupt.</p> <p><i>CONT</i> Continuously enable the interrupt.</p>
<b>Remarks</b>	<p>If the interrupt is enabled, an interrupt will cause encoder 1 and encoder 2 position to be saved into system variable <math>InP1</math> and <math>InP2</math> respectively. The system flag <math>FI_n</math> is then set, and the interrupt is disabled.</p> <p>If the interrupt is set to continuous the interrupt will not be disabled after an interrupt is detected. As a result every time an interrupt is detected <math>InP1</math> and <math>InP2</math> will be updated.</p> <p>The flag <math>FI_n</math> remains set until it is cleared with an <math>FI_n</math> OFF statement. If the state of an interrupt flag is not known when enabling the interrupt, clear the flag with an <math>FI_n = OFF</math> statement before enabling the interrupt. When a program stops, the interrupt flags are cleared.</p>
<b>See Also</b>	$FI_n$ , $InPm$ , POS1, POS2, SINT2
<b>Example</b>	<pre> INT1 ON INT2 CONT INT1 OFF ;find an accurate difference in position between encoder 1 and 2. FI1 OFF ;Turn OFF the interrupt flag. INT1 ON ;Enable the input interrupt. WAIT FI1 = ON ;Wait for the input to occur. V1 = I1P2 - I1P1 ;Calculate the difference in position. </pre>

<b>IXnPm</b>	<b>Position, Latched</b>	<b>Interrupts</b>
<b>Purpose</b>	Position of encoder <i>m</i> latched by encoder <i>n</i> index interrupt.	
<b>Syntax</b>	<i>variable</i> = IXnPm	
	<i>n</i>	The number of the encoder source of the index interrupt: <i>n</i> = 1 or 2.
	<i>m</i>	The number of the encoder from which position is read: <i>m</i> = 1 or 2.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	When an encoder 1 index interrupt occurs, the position of encoder 1 is captured and stored in the variable IX1P1.	
<b>See Also</b>	IDX <i>n</i> , FIDX <i>n</i>	
<b>Example</b>	V1 = IX1P1 V2 = IX2P2 G7 = IX1P2 - IX2P1	

<b>IZONE</b>	<b>Integrator Zone, Position Loop</b>	<b>Gain</b>
<b>Purpose</b>	Set the region around the commanded position for which the position regulator integrator is active. If IZONE is set to zero or KI = 0, the integrator is not active.	
<b>Syntax</b>	IZONE = <i>distance</i> <i>variable</i> = IZONE	
	<i>distance</i>	Specifies a <i>distance</i> around the commanded position in user units. User units are defined by the SCALE parameter. The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	KI, KP, KPZ, PZONE, KFF, IGAIN	
<b>Example</b>	IZONE = 0.1 IZONE = V2 V1 = IZONE	

<b>JACCEL</b>	<b>Jog Acceleration</b>	<b>Acceleration</b>
<b>Purpose</b>	Set the Jog acceleration. Jog acceleration is the acceleration used for jog commands and MOVV commands.	
<b>Syntax</b>	<p>JACCEL = <i>value</i>  <i>variable</i> = JACCEL</p> <p><i>value</i>                      Acceleration in user units per second per second. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                  The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.	
<b>See Also</b>	JDECEL, MOVV, JOGF, JOGR	
<b>Example</b>	<pre>JACCEL = 500 V3 = JACCEL</pre>	



<b>JDECEL</b>	<b>Jog Deceleration</b>	<b>Acceleration</b>
<b>Purpose</b>	Set the Jog deceleration. Jog deceleration is the deceleration used for jog commands and MOVV commands.	
<b>Syntax</b>	<p>JDECEL = <i>value</i>  <i>variable</i> = JDECEL</p> <p><i>value</i>                      Deceleration in user units per second per second. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                    The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.	
<b>See Also</b>	MOVV, JACCEL, JOGF, JOGR	
<b>Example</b>	<pre>JDECEL = 500 V3 = JDECEL</pre>	

<b>JOGACTIVE</b>	<b>Jog motion is occurring</b>	<b>System</b>
<b>Purpose</b>	ON indicates jog motion is occurring.	
<b>Syntax</b>	<p><i>variable</i> = JOGACTIVE</p> <p><i>variable</i>                    <i>Variable</i> can be a nonvolatile flag <i>Bn</i>, a volatile flag <i>Fn</i>, or another system flag.</p>	
<b>Remarks</b>	This flag is useful to know when jog commands have finished commanding motion. The system variables PJOG and PCMD are changing while this flag is on.	
<b>See Also</b>	JOGF, JOGR, MOVV, PJOG, MOVECOMPLETE	
<b>Example</b>	<pre>F3 = JOGACTIVE WAIT JOGACTIVE = OFF</pre>	

JOGF	Jog Forward	System
<b>Purpose</b>	Activates forward jog motion when set to ON.	
<b>Syntax</b>	JOGF = <i>value</i> <i>variable</i> = JOGF	
	<i>value</i>	The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	Turning this flag ON will cause the motor to jog in the forward direction at the jog velocity (JVVEL), using JACCEL and JDECEL. <b>TIP:</b> By default, Jog velocity is set to 0 in a program unless the program contains a JVVEL statement. Setting Jog velocity to 0 has the same effect as disabling the Jog inputs. Once the program is stopped, the Jog velocity will return to what it was before the program was run. After executing a MOVV move, JVVEL will be changed to the value of the MOVV.	
<b>See Also</b>	JOGR, MOVV, JVVEL, JACCEL, JDECEL	
<b>Example</b>	<pre>JVVEL = 200           ;set the jog velocity to 200 JOGF = ON             ;start jogging forward</pre>	

JOGR	Jog Reverse	System
<b>Purpose</b>	Activates reverse jog motion when set to ON.	
<b>Syntax</b>	JOGR = <i>value</i> <i>variable</i> = JOGR	
	<i>value</i>	The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	Turning this flag ON will cause the motor to jog in the reverse direction at the jog velocity (JVVEL), using JACCEL and JDECEL. <b>TIP:</b> By default, Jog velocity is set to 0 in a program unless the program contains a JVVEL statement. Setting Jog velocity to 0 has the same effect as disabling the Jog inputs. Once the program is stopped, the Jog velocity will return to what it was before the program was run. After executing a MOVV move, JVVEL will be changed to the value of the MOVV.	
<b>See Also</b>	JOGF, MOVV, JVVEL, JACCEL, JDECEL	
<b>Example</b>	<pre>JVVEL = 200           ;set the jog velocity to 200 JOGR = ON             ;start jogging reverse</pre>	

<b>JUMP</b>	<b>Jump</b>	<i>Program Structure</i>
<b>Purpose</b>	Transfer program execution to the instruction following the label.	
<b>Syntax</b>	<p><code>JUMP = label</code></p> <p><i>label</i>                      The label must be located within the same program as the jump but may be located before or after the jump statement.</p>	
<b>Remarks</b>	The label must be located within the same program as the jump but may be located before or after the jump statement.	
<b>See Also</b>	LABEL, SUB	
<b>Example</b>	<pre> Begin:                                 ... statements JUMP BEGIN JMP LABEL1                                 ... statements  LABEL1: </pre>	
<b>JVEL</b>	<b>Jog Velocity</b>	<i>Velocity</i>
<b>Purpose</b>	Set the Jog velocity.	
<b>Syntax</b>	<p><code>JVEL = value</code> <code>variable = JVEL</code></p> <p><i>value</i>                      The <i>value</i> is specified in user units per user Timebase. User units are defined by the SCALE parameter. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                      The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	<p>By default, Jog velocity is set to 0 in a program unless the program contains a JVEL statement. Setting Jog velocity to 0 has the same effect as disabling the Jog inputs. Once the program is stopped, the Jog velocity will return to what it was before the program was run. After executing a MOVV move, JVEL will be changed to the value of the MOVV statement.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.</p>	
<b>See Also</b>	VEL, HVEL, JOGF, JOGR, MOVV, JACCEL, JDECEL	
<b>Example</b>	<pre> JVEL = 10           ;establish a jog speed of 10 V3 = JVEL </pre>	

<b>KFF</b>	<b>Velocity feedforward Gain</b>	<i>Gain</i>
<b>Purpose</b>	Set the velocity feedforward gain to a new value. Velocity feedforward gain may be changed at any time in the program.	
<b>Syntax</b>	$KFF = value$ $variable = KFF$	
	<i>value</i>	The <i>value</i> is entered in percent. The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable. The range of this value is 0 to 200%.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	KP, KPZ, PZONE, KI, IZONE	
<b>Example</b>	$KFF = 75$ $KFF = V8$ $V7 = KFF$	

<b>KI</b>	<b>Integral Gain, Position Regulator</b>	<i>Gain</i>
<b>Purpose</b>	Set the integral gain to a new value. The position regulator integral gain may be changed at any time in the program. The integral gain is active when the position error is less than IZONE (regardless of commanded velocity). If IZONE is set to zero, integral gain is not active.	
<b>Syntax</b>	$KI = value$ $variable = KI$	
	<i>value</i>	The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	IZONE, KP, KPZ, PZONE, KFF	
<b>Example</b>	$KI = 1.5$ $KI = G3$ $V4 = KI$	

KP	Proportional Gain, Position Regulator	Gain
<b>Purpose</b>	Set the proportional gain to a new value. The position regulator proportional gain may be changed at any time in the program.	
<b>Syntax</b>	$KP = value$ $variable = KP$	
	<i>value</i>	The <i>value</i> is entered in distance units/minute per thousandth of a distance unit. The <i>value</i> can be a constant, a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or a system variable.
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or a system variable.
<b>Remarks</b>	<p>KP = 1 means that if there is 1 thousandth of a distance unit of following error, the resulting velocity command will be 1 distance unit/minute. For example:</p> <ul style="list-style-type: none"> <li>• If distance units are inches, KP = 1 means 1 inch/minute/mil (1/1000 inch).</li> <li>• If distance units are meters, KP = 1 means 1 m/minute/mm (1/1000 meter).</li> </ul> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gain dialog box.</p>	
<b>See Also</b>	KPZ, PZONE, KI, IZONE, KFF	
<b>Example</b>	$KP = 2.5$ $KP = V12$ $V13 = KP$	

<b>KPZ</b>	Proportional Gain Zone, Position Regulator	<i>Gain</i>
<b>Purpose</b>	Set the proportional gain to be used instead of KP in the region defined by PZONE. The units of KPZ are inches/minute/mil or meters/minute/millimeter. Whenever the position error is less than PZONE, the position regulator uses KPZ for the proportional gain, otherwise it uses KP. If PZONE is zero, the KPZ gain is never active.	
<b>Syntax</b>	<p><math>KPZ = value</math>  <math>variable = KPZ</math></p> <p><i>value</i>                      The <i>value</i> is entered in distance units/minute per thousandth of a distance unit. The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                    The <i>variable</i> can be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	<p>KPZ = 1 means that if there is 1 thousandth of a distance unit of following error, the resulting velocity command will be 1 distance unit/minute.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains dialog Box</p>	
<b>See Also</b>	PZONE, KP, KI, IZONE, KFF	
<b>Example</b>	<pre>KPZ = 4 KPZ = G10 G3 = KPZ</pre>	

<b>LABELS</b>	Program Labels	<i>Program Structure</i>
<b>Purpose</b>	Assign the present statement as a target location used by a JUMP instruction. A label can be up to 32 characters long. Labels cannot be keywords or reserved words. They must begin with a letter and contain only alphanumeric characters and underscores ( _ ), and must be followed by a colon.	
<b>Syntax</b>	LABELS:	
<b>Remarks</b>	Labels may be inserted anywhere in the program.	
<b>See Also</b>	JUMP	
<b>Example</b>	<pre>mylabel: Main3: JUMP mylabel</pre>	



<b>LOOPINDEX</b>	<b>Loop Index</b>	<b>System</b>
<b>Purpose</b>	The LOOP counter for the LOOP instruction used in a subroutine or the main body of the program.	
<b>Syntax</b>	$variable = LOOPINDEX$ $LOOPINDEX = value$	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or another system variable.
	<i>value</i>	<i>Value</i> can be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , another system variable, or a constant.
<b>Remarks</b>	This variable may be used to determine how many times the statements in a loop have been executed. This variable initially is set to the value in the loop statement, and counts down to zero. The variable must be in the range of 65535 to 0.	
<b>See Also</b>	LOOP, XLOOPINDEX, REPEAT	
<b>Example</b>	<pre> LOOP 10                 V4 = LOOPINDEX                 PRINT V4 REPEAT </pre>	

<b>LPOS</b>	<b>Position, Latched</b>	<b>Interrupts</b>
<b>Purpose</b>	Hardware latched position in user units.	
<b>Syntax</b>	$variable = LPOS$	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.
<b>Remarks</b>	Value of POS1 or POS2 (depending on Feedback Configuration) that is latched in hardware when an interrupt 1 (INPUT 11, P1-12) is received.	
<b>See Also</b>	<i>InPm</i> , Configuration menu	
<b>Example</b>	<pre> V3 = LPOS G1 = LPOS - POS1 </pre>	



<b>MONOUT</b>	<b>Monitor Output</b>	<b>System</b>
---------------	-----------------------	---------------

**Purpose** The number of the currently active variable for the monitor analog output.

**Syntax** MONOUT = *value*  
*variable* = MONOUT

*variable* Variable can be a nonvolatile flag *Bn*, a volatile flag *Fn*, or another system flag.

*value* The *value* represents the variable number to be displayed. *Value* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable. The numbers corresponding to the variable names are shown below.

Value	Name	Description
0		Clear the top line
1	PCMD	Command Position
2	POSN	Feedback Position
3	FE	Following Error
4	PGEN	Profile Generator Position
5	PJOG	Jog Position
6	PEXT	Gear Position after Slew
7	GPOS	Gear Position before Slew
8	POS1	Encoder 1 Position
9	POS2	Encoder 2 Position
10	POS3	Position from Option Card
11	VCMD	Commanded Velocity
12	VEL1	Un-filtered Feedback Velocity
13	VEL2	Encoder 2 Velocity
14	ICMD	Commanded Current
15	IAVE	Average Commanded Current
16	RFDR	Runtime Feedrate
17	ADC1	Analog to Digital (A to D) Converter 1
18	ADC2	A to D Converter 2 (Option Card)
19	ADC3	A to D Converter 3 (Option Card)
20	ADC4	A to D Converter 4 (Option Card)
21	ADC5	A to D Converter 5 (Option Card)
22	PFE	Peak Following Error
23	PVEL1	Peak Feedback Velocity
24	PICMD	Peak Commanded Current
25	I1P1	Interrupt 1 encoder 1 Position
26	I1P2	Interrupt 1 encoder 2 Position
27	I2P1	Interrupt 2 encoder 1 Position
28	I2P2	Interrupt 2 encoder 2 Position
29	IX1P1	Encoder 1 Index Interrupt encoder 1 Position
30	IX1P2	Encoder 1 Index Interrupt encoder 2 Position
31	IX2P1	Encoder 2 Index Interrupt encoder 1 Position
32	IX2P2	Encoder 2 Index Interrupt encoder 2 Position

<b>Syntax</b> (cont.)	<i>value</i> (cont.)	33	LPOS	Interrupt 1 Latched Position (encoder 1 or 2)
		34	FVEL1	Filtered Feedback Velocity

**Remarks****See Also**

DACn, OTMON

**Example**

```
MONOUT = 3
V3 = MONOUT
MONOUT = G24
```

---

<b>MOVD</b>	<b>Move a Distance</b>	<i>Motion</i>
-------------	------------------------	---------------

---

**Purpose** Move an incremental distance from the present position. The MOVD command generates a trapezoidal or triangular motion profile using the present acceleration and velocity and the specified incremental distance. Optionally, the velocity used for the move may be specified in the instruction.

**Syntax** MOVD = *distance*[, *V* = *velocity*]  
MD = *distance*[, *V* = *velocity*]

*distance* The incremental *distance* to be moved in user units. The *distance* may be positive or negative and can be a number or a variable. *Distances* which result in a fractional number of encoder counts should be programmed as fractions (for example, 22/3 for 7.333333). Distances programmed in this manner will maintain fractional count information so no long term drift in position will occur. *Distance* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

*velocity* The velocity to be used for the move in user units per Timebase. If no velocity is specified, the Velocity variable is used (unless overridden by a Velocity statement in the program). If the distance is too short to reach the velocity, a triangular motion profile will be formed. A velocity specified in the instruction is used for this move only; it does not affect any later moves. *Velocity* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

**Remarks** ACCEL can be used to set the acceleration and deceleration used by the MOVD instruction. If the *velocity* specified cannot be reached using the present acceleration and deceleration without overshooting the *distance*, a triangular velocity profile will result, with a peak less than *velocity*.

The program will stop executing at this line until the commanded position has moved the specified *distance*.

The MOVD statement can only be used in the body or subroutines of a main program. It may not be used in Fkey, Xkey, or Scanned Event routines.

**See Also** MOVP, DV, DIF, MOVV, ACCEL, VEL

**Example**

```
MOVD = 10
MOVD = 5, V = 100
MOVD = V10, V = V21
ACCEL = 150
MD = 5
```

<b>MOVECOMPLETE</b>	Profile generated motion is complete	<i>System</i>
<b>Purpose</b>	OFF indicates profile generated motion is occurring.	
<b>Syntax</b>	<i>variable</i> = MOVECOMPLETE	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This flag is useful to know when a move has finished commanding motion. The system variables PGEN and PCMD are changing while this flag is off.	
<b>See Also</b>	INPOSN, PGEN	
<b>Example</b>	<pre>F3 = MOVECOMPLETE WAIT MOVECOMPLETE = ON</pre>	

<b>MOVP</b>	Move to Position	<i>Motion</i>
<b>Purpose</b>	Move to an absolute position. A trapezoidal or triangular motion profile is formed as described for the MOVD instruction. An optional velocity for the move may be specified in the instruction. An absolute position is defined in the absolute mode as a position relative to the established zero position. A home sequence or a Define Home (DH) must be completed before this instruction can be executed.	
<b>Syntax</b>	MOVP = <i>position</i> [, <i>V</i> = <i>velocity</i> ] MP = <i>position</i> [, <i>V</i> = <i>velocity</i> ]	
	<i>position</i>	The absolute position desired at the completion of the move. Position may be positive or negative to indicate the direction relative to the home position. <i>Position</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>velocity</i>	The <i>velocity</i> to be used for this move in user units per Time-base. If no velocity is specified, the system velocity variable is used (unless overridden by a Velocity statement in the program). If the distance is too short to reach the velocity, a triangular motion profile will be formed. A velocity specified in the instruction is used for this move only; it does not affect any later moves. <i>Velocity</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	ACCEL can be used to change the acceleration and deceleration used for this move. If the <i>velocity</i> specified cannot be reached using the present acceleration and deceleration without overshooting the <i>position</i> , a triangular velocity profile will result, with a peak less than <i>velocity</i> . The program will stop executing at this line until the commanded position has reached the specified <i>position</i> . If a home sequence has not been completed and this instruction is executed, error 17, Home Not Defined, will occur. The MOVP statement can only be used in a subroutine or the main body of the program. It may not be used in Fkey, Xkey, or Scanned Event routines.	

**See Also** MOVD, DV, DIF, MOVV, DH, DP, ACCEL, VEL

**Example**

```

MOVV = 0
MOVV = 0, V = 50
MP = G3, V = G5

```

<b>MOVV</b>	<b>Move at Velocity</b>	<i>Motion</i>
<b>Purpose</b>	Move at a velocity. Once a MOVV command is executed, the motor will continue moving at the commanded velocity until a MOVV = 0 command is executed or the program is stopped. MOVV commands will continue even while other motion commands are executing, such as MOVD, MOVV, or DV commands, so the velocity move is added to the other motion commands. The MOVV command uses the Jog Acceleration (JACCEL) and Jog Deceleration (JDECEL) when changing speed.	
<b>Syntax</b>	<pre> MOVV = <i>velocity</i> MV = <i>velocity</i> </pre> <p><i>velocity</i>                      The <i>velocity</i> is in user units per Timebase. <i>Variable</i> may be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable. The sign of velocity determines direction.</p>	
<b>Remarks</b>	<p>This command uses the same system resources as the jog functions. Jogging in the same direction as the MOVV will have no effect while a MOVV is executing. Jogging in the direction opposite the MOVV will cause motion to ramp to a stop at the Jog deceleration rate (JDECEL).</p> <p><b>Note:</b> Executing this command changes the value of JVEL to the velocity of the MOVV.</p>	
<b>See Also</b>	MOVD, MOVV, DV, DIF, JACCEL, JDECEL, JVEL, JOGR, JOGF	
<b>Example</b>	<pre> MOVV = 100 MV = V16 MOVV = G23 </pre>	

<b>MVOEN</b>	<b>MVO Output Enable</b>	<i>System</i>
<b>Purpose</b>	Test point TP3 may be used to monitor the velocity of the motor if MVOEN is set ON.	
<b>Syntax</b>	MVOEN= <i>value</i> <i>variable</i> = MVOEN	
	<i>value</i>	<i>Value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	Changes made to this flag by a program are only in effect while the program is running. When the program ends it will return to the setting stored in the Personality Module.	
<b>See Also</b>	MONOUT, DACn, OTMON	
<b>Example</b>	MVOEN = ON F3 = MVOEN	

<b><i>Om</i> = ON/OFF/ flag</b>	<b>Output</b>	<i>Outputs</i>
<b>Purpose</b>	Turn Output <i>m</i> ON or OFF or set it to the same state as a flag.	
<b>Syntax</b>	<i>Om</i> = ON/OFF/flag[, T = <i>time</i> ]	
	<i>m</i>	The number of the output: <i>m</i> = 1, 2, ...,8. If the controller is equipped with an I/O Expansion card or a Memory and I/O Expansion card, <i>m</i> = 1, 2, ...,24.
	<i>ON</i>	Turn the output <i>m</i> ON. Outputs are open collector type. ON turns the transistor ON and pulls the output to 24V common.
	<i>OFF</i>	Turn the output OFF. Outputs are open collector type. OFF turns the transistor OFF and leaves the output floating. There is no internal pull-up.
	<i>flag</i>	Set the output to the same state as a flag. The flag may be an F, a B, or a system flag.
	<i>time</i>	An optional <i>time</i> entered in seconds, provides a pulsed output. The output will turn ON or OFF for the period specified. This optional <i>time</i> value may not be used if a flag is used to set the output. The <i>time</i> may be a constant, a <i>Gn</i> variable, or a <i>Vn</i> variable.
<b>Remarks</b>		
<b>See Also</b>	ALL, BCD, BIN, Im	

**Example**

```

01 = ON
02 = OFF
03 = F1
04 = ON, T = 0.1      ;turn on output 4 for 100 mS
05 = OFF, T = 3.2    ;turn off output 5 for 3.2 S

```

---

**ON variable  
JUMP**


---

**Case Statement****Program Structure****Purpose**

Jump to one of a list of labels based on the value of *variable*. The variable acts as a pointer into the list of labels. Labels are zero-based, so if *variable* = 0, the program jumps to the first label. If *variable* = 3, it jumps to the fourth label and so on. If the value of *variable* is greater or equal to the number of labels in the list, execution falls through to the next instruction after the ON *variable* JUMP instruction.

**Syntax**

ON *Variable* JUMP *label1*, *label2*, *label3*, ..., *labeln*

*variable*                    The *variable* may be a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

*labeln*                    *labeln* is any valid label in the program. n = 1, 2, ...

**Remarks**

This instruction is similar to the case instruction in many other programming languages.

**See Also**

JUMP, LABEL

**Example**

```

ON V1 JUMP LABEL1, LABEL2, LABEL3
ON G10 JUMP MODE1, MODE2, MODE3
ON V1 JMP START, LOAD, INDEX, ABORTMOVE, GO

```

**OTMON****Operator Terminal Monitor***Operator Terminal*

**Purpose** The OTMON = *value* statement is used to display monitor variables on the top line of the Operator Terminal screen.

**Syntax** OTMON = *value*  
*variable* = OTMON

**value** The *value* represents the variable number to be displayed. *Value* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable. The numbers corresponding to the variable names are shown below.

Value	Name	Description
0		Clear the top line
1	PCMD	Command Position
2	POSN	Feedback Position
3	FE	Following Error
4	PGEN	Profile Generator Position
5	PJOG	Jog Position
6	PEXT	Gear Position after Slew
7	GPOS	Gear Position before Slew
8	POS1	Encoder 1 Position
9	POS2	Encoder 2 Position
10	POS3	Position from Option Card
11	VCMD	Commanded Velocity
12	VEL1	Un-filtered Feedback Velocity
13	VEL2	Encoder 2 Velocity
14	ICMD	Commanded Current
15	IAVE	Average Commanded Current
16	RFDR	Runtime Feedrate
17	ADC1	Analog to Digital (A to D) Converter 1
18	ADC2	A to D Converter 2 (Option Card)
19	ADC3	A to D Converter 3 (Option Card)
20	ADC4	A to D Converter 4 (Option Card)
21	ADC5	A to D Converter 5 (Option Card)
22	PFE	Peak Following Error
23	PVEL1	Peak Feedback Velocity
24	PICMD	Peak Commanded Current
25	I1P1	Interrupt 1 encoder 1 Position
26	I1P2	Interrupt 1 encoder 2 Position
27	I2P1	Interrupt 2 encoder 1 Position
28	I2P2	Interrupt 2 encoder 2 Position
29	IX1P1	Encoder 1 Index Interrupt encoder 1 Position
30	IX1P2	Encoder 1 Index Interrupt encoder 2 Position
31	IX2P1	Encoder 2 Index Interrupt encoder 1 Position
32	IX2P2	Encoder 2 Index Interrupt encoder 2 Position
33	LPOS	Interrupt 1 Latched Position (encoder 1 or 2)
34	FVEL1	Filtered Feedback Velocity

**variable** The *variable* may be a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

**Remarks** The present value of OTMON can also be read and used in a conditional statement.

**See Also** PRINT

**Example**

```

;Monitor POSN on the Operator Terminal
OTMON = 2
;If POSN is being monitored, stop monitoring POSN and clear the top line
;of the Operator Terminal
IF(OTMON = 2)OTMON = 0
    
```

<b>OVERSPEED</b>	<b>Overspeed Threshold</b>	<i>System</i>
<b>Purpose</b>	Peak velocity trip point in user units per timebase. If the velocity exceeds this trip point, a fault is reported.	
<b>Syntax</b>	<p>OVERSPEED = <i>variable</i>  <i>variable</i> = OVERSPEED</p> <p><i>variable</i>                      <i>Variable</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or another system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box	
<b>See Also</b>	VEL	
<b>Example</b>	<pre> V4 = OVERSPEED OVERSPEED=3500     </pre>	

WORDS



PAUSE	Pause	System
<b>Purpose</b>	This instruction will cause motion to ramp to a stop and pause. Pause can also be set to ON or OFF by the Pause input.	
<b>Syntax</b>	PAUSE = <i>value</i> <i>variable</i> = PAUSE  <i>value</i> The <i>value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.  <i>variable</i> The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.	
<b>Remarks</b>	When set to ON all motion will ramp to a stop. This is a level sensitive condition. As long as the flag is ON, the system will continue to decelerate. Once the flag is OFF the system will accelerate back up to normal operating speed. The system may never come to a stop if the flag is set and cleared quickly. The rate of deceleration and acceleration is determined by the system acceleration variable, system velocity variable, and the present velocity. The deceleration rate will equal: $\text{SystemFeedrateSlewTime} = \frac{\text{SystemVelocity}}{\text{SystemAccelerationRate}}$ $\text{DecelerationRate} = \frac{\text{PresentVelocity}}{\text{SystemFeedrateSlewTime}}$ System Velocity is the velocity set in the parameter menu, Velocity/Accel dialog box. System Acceleration is the acceleration set in the Parameter menu, Velocity/Accel dialog box. If the pause was caused by an instruction and a MOVD, MOVP or DWELL command is executed before PAUSE is set to OFF, the program will be “hung”. Turning the PAUSE input OFF and ON will have no effect. If this occurs do one of the following: <ul style="list-style-type: none"> <li>• Send PAUSE = OFF on serial port 2. The program will continue.</li> <li>• Turn the ENABLE input OFF. The program will be aborted.</li> <li>• Send ABORT on serial port 2.</li> <li>• Turn the power OFF and then ON.</li> </ul> If the pause was caused by the PAUSE input, turning the PAUSE input OFF will cause normal execution to return and executing PAUSE = OFF will have no effect. After the PAUSE has been set to ON, instructions other than MOVD, MOVP, DV, DIF, MOVV, and DWELL will execute normally. The MOVV instruction will establish a new velocity, however no motion will begin until PAUSE is set to OFF	
<b>See Also</b>	STOP, ABORT, DISABLE	
<b>Example</b>	PAUSE = ON PAUSE = OFF PAUSE = F1 F2 = PAUSE	

<b>PCAM</b>	Position Command, Cam	<i>Position</i>
<b>Purpose</b>	Position command from cam profile generator	
<b>Syntax</b>	<i>variable</i> = PCAM	
	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.	
<b>Remarks</b>	The value in this variable represents the position command generated by the cam profile generator.	
<b>See Also</b>	PCMD	
<b>Example</b>	V1 = PCAM	

<b>PCAMACTIVE</b>	Position Command, Cam Active	<i>System</i>
<b>Purpose</b>	Flag to indicate the PCam position value is changing.	
<b>Syntax</b>	<i>variable</i> = PCAMACTIVE	
	variable	The <i>variable</i> will be set to ON or OFF. This is a read-only system flag that signals the PCam position value is changing.
<b>Remarks</b>	TRUE (ON) indicates that the PCam variable is changing.	
<b>See Also</b>	PCAM, PCMD	
<b>Example</b>	PCAMACTIVE = ON	

<b>PCMD</b>	Position Command	<i>Position</i>
<b>Purpose</b>	Position command in user units. This is the sum of the position command from the profile jog, external, and cam sources.	
<b>Syntax</b>	<i>variable</i> = PCMD	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	PCMD = PGEN + PJOG + PEXT + PCAM	
<b>See Also</b>	POSN, POS1, POS2, PEXT, PGEN, PJOG, PCAM	
<b>Example</b>	V2 = PCMD V3 = PCMD - V4	

<b>PEXT</b>	<b>Position Command, External</b>	<i>Position</i>
<b>Purpose</b>	External position command (from gear or step and direction) in user units after the SLEW limit.	
<b>Syntax</b>	<i>variable</i> = PEXT	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	The value in this variable represents the position command generated by an external source. That source can be encoder 2 if a gear is established and enabled, or from a step and direction command.	
<b>See Also</b>	GPOS, PCMD	
<b>Example</b>	V3 = PEXT	

<b>PEXTACTIVE</b>	<b>Position Command, External Active</b>	<i>System</i>
<b>Purpose</b>	Flag to indicate the PExt position value is changing.	
<b>Syntax</b>	<i>variable</i> = PEXTACTIVE	
	<i>variable</i>	The <i>variable</i> will be set to ON or OFF. This is a read-only system flag that signals the PEXT position value is changing.
<b>Remarks</b>	TRUE (ON) indicates that the PEXT variable is changing.	
<b>See Also</b>	GPOS, PEXT, PCMD	
<b>Example</b>	PEXTACTIVE = ON	

<b>PFE</b>	<b>Following Error, Peak</b>	<i>Position</i>
<b>Purpose</b>	Peak following error in user units.	
<b>Syntax</b>	<i>variable</i> = PFE	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
<b>Remarks</b>	The following error is calculated every position loop update (2mS). If the present value is greater than the previous peak the value is saved as the new peak. The magnitude of the peak is calculated based on an absolute value of following error. The value is saved and reported as a signed number, however. CLRPEAKS resets this value to zero.	
<b>See Also</b>	FE, CLRPEAKS	
<b>Example</b>	V1 = PFE	

<b>PGAIN</b>	<b>Proportional Gain, Velocity Loop</b>	<i>Gain</i>
<b>Purpose</b>	Set the proportional gain for the velocity loop. The velocity loop proportional gain may be changed at any time in the program.	
<b>Syntax</b>	PGAIN = <i>value</i> <i>variable</i> = PGAIN	
	<i>value</i>	The <i>value</i> may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
	<i>variable</i>	<i>Variable</i> may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	IGAIN, FGAIN	
<b>Example</b>	PGAIN = 0.5	

<b>PGEN</b>	<b>Position Command, Profile Generator</b>	<i>Position</i>
<b>Purpose</b>	Position command from internal profile generator	
<b>Syntax</b>	<i>variable</i> = PGEN	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	The value in this variable represents the position command generated by the profile generator. The profile generator is responsible for creating MOVD, MOVP, DV and DIF moves.	
<b>See Also</b>	PCMD	
<b>Example</b>	V1 = PGEN	

<b>PGMNUM</b>	<b>Program Number</b>	<i>System</i>
<b>Purpose</b>	PGMNUM is used in conjunction with STARTP. PGMNUM specifies which program will execute when STARTP is executed.	
<b>Syntax</b>	PGMNUM = <i>variable</i>	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable. The variable may be 0 through 31 or 0 through 63 if a Memory and I/O Expansion card is installed.
<b>Remarks</b>		
<b>See Also</b>	STARTP, Appendix A, System Programs, Run	
<b>Example</b>	PGMNUM = 3 STARTP = ON	

<b>PGMRUNNING</b>	<b>Program Running</b>	<i>System</i>
<b>Purpose</b>	If this flag is ON, a main program is running.	
<b>Syntax</b>	<i>variable</i> = PGMRUNNING	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>		
<b>See Also</b>	STARTP	
<b>Example</b>	F4 = PGMRUNNING	

<b>PGMTYPE</b>	<b>Program Type</b>	<b>Compiler Options</b>
<b>Purpose</b>	Set the type of program to be compiled.	
<b>Syntax</b>	PGMTYPE = MAINPGM/AUTOPGM/FKEYPGM/ERRPGM/CAMPROF	
<b>Remarks</b>	<p>The PGMTYPE statement overrides the default compiler option for Program Type set in the Edit menu, Compiler Options dialog box.</p> <p>MAINPGM: Normal motion program  AUTOPGM: Auto program, System program zero  FKEYPGM: Fkey program, System programs 1-24  ERRPGM: Error Routine program, System program 27  CAMPROF: Cam profile, requires IQ Cam compiler</p> <p>This statement requires IQ Master version 3.0 or later.</p>	
<b>See Also</b>	DEBUG, COMPILEMEMORY, LISTFILE, EXPANDMACROS	
<b>Example</b>	PGMTYPE = MAINPGM PGMTYPE = FKEYPGM	
<b>PICMD</b>	<b>Current Command, Peak</b>	<b>Current</b>
<b>Purpose</b>	Peak current command in amperes.	
<b>Syntax</b>	$variable = PICMD$  $variable$ The $variable$ may be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or a system variable.	
<b>Remarks</b>	<p>ICMD is calculated every velocity loop update (1mS). If the current value is greater than the previous peak, the value is saved as the new peak. The magnitude of the peak is calculated based on an absolute value of current command (ICMD). The value is saved and reported as a signed number, however. The variable is scaled by the drive module current (Dm).</p> <p>CLRPEAKS resets this value to zero.</p>	
<b>See Also</b>	ICMD, CLRPEAKS, IAVE, IAVG	
<b>Example</b>	G6 = PICMD	

<b>PJOG</b>	<b>Position Command, Jog</b>	<i>Position</i>
<b>Purpose</b>	The position command that is the result of Jog commands from the Jog input(s), Jog Fkey programs, and MOVV commands.	
<b>Syntax</b>	<i>variable</i> = PJOG	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	<p>The value in this variable represents the position command generated by the jog command generator. The jog command generator is responsible for creating jog commands and MOVV moves. Jog commands can be initiated from the following sources:</p> <ul style="list-style-type: none"> <li>• Jog inputs, if assigned</li> <li>• Fkey programs</li> <li>• Host Language Command</li> <li>• MOVV instructions</li> <li>• JOGF or JOGR</li> </ul>	
<b>See Also</b>	PCMD	
<b>Example</b>	V1 = PJOG	

<b>POS1</b>	<b>Position, Encoder 1</b>	<i>Position</i>
<b>Purpose</b>	Encoder 1 position in user units, scaled by the Scale parameter.	
<b>Syntax</b>	<i>variable</i> = POS1	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>		
<b>See Also</b>	POSN, POS2	
<b>Example</b>	V1 = POS1	

<b>POS2</b>	<b>Position, Encoder 2</b>	<i>Position</i>
<b>Purpose</b>	Encoder 2 position in user units, scaled by the Scale2 parameter.	
<b>Syntax</b>	<i>variable</i> = POS2	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>		
<b>See Also</b>	POS1, POSN	
<b>Example</b>	V1 = POS2	
<b>POS3</b>	<b>Position, Encoder 3 (Optional)</b>	<i>Position</i>
<b>Purpose</b>	Encoder 3 position (from option card).	
<b>Syntax</b>	<i>variable</i> = POS3	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable
<b>Remarks</b>		
<b>See Also</b>	POS1, POS2, POSN	
<b>Example</b>	V1 = POS3	
<b>POSN</b>	<b>Position Feedback</b>	<i>Position</i>
<b>Purpose</b>	Feedback position in user units, scaled by the Scale parameter.	
<b>Syntax</b>	<i>variable</i> = POSN	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	POSN is the position of the encoder that is currently being used as the feedback encoder (typically encoder 1). The CONFIG menu establishes which encoder is currently the feedback encoder. The position of that encoder is read using the corresponding SCALE to determine POSN.	
<b>See Also</b>	POS1, POS2, POS3, PCMD	
<b>Example</b>	G1 = POSN	



PRINT	Print	Operator Terminal
<b>Purpose</b>	Print a message or a variable, or both, to the Operator Terminal or a serial port.	
<b>Syntax</b>	PRINT [# <i>n</i> ] [ <i>row, column</i> ] [" <i>message</i> "] [, <i>variable</i> ] [, <i>field, precision</i> [, <i>F</i> ]]	
	<i>n</i>	Serial port 1 or 2 is specified by <i>n</i> . If a port is not specified, the output defaults to serial port 1, the Operator Terminal port.
	<i>row</i>	The <i>row</i> that printing begins on. The top row is 1.
	<i>column</i>	The <i>column</i> that printing begins on. The left hand column is 1.
	<i>message</i>	The <i>message</i> to be printed can be any string of characters in double quotation marks. The message may also contain codes to control the display.
	<i>variable</i>	If <i>variable</i> is a G or V variable, the value of the variable is displayed. If <i>variable</i> is a B or F flag variable, “Y” is displayed if the variable is ON, “N” is displayed if the variable is OFF
	<i>field</i>	<i>field</i> determines how many characters are allowed for the variable including the decimal point. <i>field</i> does not apply when printing flag variables. If the value to be printed is longer than the specified <i>field</i> , the entire value will be printed, overflowing the field length.
	<i>precision</i>	<i>precision</i> determines the number of places (1 to 4) after the decimal point. The maximum precision that will be permitted is 4. If <i>field</i> and <i>precision</i> are not included, the default precision is 2. <i>precision</i> does not apply when printing flag variables.
	<i>F</i>	If <i>F</i> is used after <i>precision</i> , AND <i>precision</i> is set to 0, the variable will be printed as a fraction if the <i>variable</i> contains a fractional part. If the <i>variable</i> is a whole number, it will be printed as a decimal number even if <i>F</i> is specified. If <i>F</i> is used with <i>precision</i> from 1 to 4, the <i>variable</i> will be printed as a fraction unless the fractional part can be expressed as a decimal number. For example, 5 1/3 would be printed as 5 1/3 rather than 5.3333, but 5 1/4 would be printed as 5.25. This does not apply when printing flag variables.

**Remarks**

Numbers are always displayed justified to the right side of the field. However, the field will be overrun if the number of characters in the variable exceeds the specified field width.

Control codes can be sent to a serial port using the standard PRINT statement. The control codes may be used to position the cursor, print other text, or perform special functions. These control codes are normally used with a generic terminal rather than an Allen-Bradley Operator Terminal. The syntax of the PRINT statement to send the code is:

```
PRINT "^code"
```

where `code` is a two digit hexadecimal representation of the character to be printed.

An example that prints a message to a generic display device is shown below. The example prints a blank line using the control codes '0D'(carriage return) and '0A'(line feed) then prints the message on the next line:

```
PRINT "^0D^0ASpeed=“ ,G23
```

Control codes may be embedded anywhere in a message text string but must always be preceded by the caret character (^).

Other control codes have specific functions when sent to the Allen-Bradley Operator Terminal. The codes available are:

<u>Code</u>	<u>Function</u>
D0	Begin character blink field
D1	End character blink field
D2	Turn Fkey Mode display ON
D3	Turn Fkey Mode display OFF
D4	Lock Fkey Mode ON
D5	Lock Fkey Mode OFF
C0	Display Fkey Mode 1
C1	Display Fkey Mode 2
C2	Display Fkey Mode 3
C3	Display Fkey Mode 4
C4	Display Fkey Mode 5
C5	Display Fkey Mode 6
C8	Write to main program display
C9	Write to Xkey display
CA	Write to Fkey display
CB	Write to Error routine display
0D	Carriage Return
0A	Line Feed
1B	Escape

The codes C8, C9, CA, and CB are used to print to different Operator Terminal screens without affecting what your program is displaying. For example, use the code C9 in a PRINT statement in an Xkey routine to display information while in the Xkey routine. When the Xkey routine ends, the Operator Terminal will display what was shown before the Xkey routine began.

**Note:** If Lock Fkey menu is used in a program, the lock will be turned OFF when the program ends.

Print statements are not allowed in Scanned Event routines.

**See Also**

READ

**Example**

```

PRINT #2 1, 1 "The Length is ", V1, 3, 2
PRINT 1, 1 "Distance is ", V1, 3, 0, F
PRINT 1, 1, V1
PRINT 1, 1, F1
PRINT 2, 6 "^DORUNNING^D1" ;The message RUNNING is displayed on
                               ;line 2 in blinking mode
PRINT "^C1" ;Display Fkey Mode 2
PRINT "^D2" ;Turn Fkey Mode display ON
PRINT "^D4" ;Lock Fkey Mode ON--no other Fkey Mode
                               ;can be displayed from the Operator
                               ;Terminal using the Mode key

PRINT F1

```

<b>PRINTWARN</b>	<b>Print Warning</b>	<b>Operator Terminal</b>
<b>Purpose</b>	Print the proper message to the operator terminal upon occurrence of a warning.	
<b>Syntax</b>	PRINTWARN	
<b>Remarks</b>	This statement would be normally used only in the Error Routine Program (System Program 27).	
<b>See Also</b>	PRINTERR, Appendix A, System Programs	
<b>Example</b>	PRINTWARN	

<b>PURGEMOTION</b>	Purge Motion	<i>Motion</i>
<b>Purpose</b>	This read/write variable will immediately remove the active Pgen motion. Control of deceleration and subsequent Pgen motion are dependent on the variable value.	
<b>Syntax</b>	<i>variable</i> = PURGEMOTION	
	<i>variable</i>	The <i>variable</i> defines which of the purge mode states is valid. The numbers corresponding to the variable modes are shown below.
	Value	Description
	0	Normal Pgen motion in effect.
	1	Purge Pgen motion immediately. No deceleration of Pgen to zero velocity performed.
	2	Purge Pgen motion immediately. Decelerate Pgen to zero velocity. Decel rate is the system Accel value.
	3	Purge Pgen motion immediately. No deceleration of Pgen to zero velocity performed. Pgen motion in effect after current motion is purged.
	4	Purge Pgen motion immediately. Decelerate Pgen to zero velocity. Decel rate is the system Accel value. Pgen motion in effect after current motion is purged.
<b>Remarks</b>	All other values assigned to this variable will result in mode 0 being assigned. When PurgeMotion is non-zero, all subsequent move type statements will be ignored until a value of zero is assigned.  This command is intended for use in an Fkey routine, an Xkey routine, or a scanned event. Use in the main program thread is discouraged.	
<b>See Also</b>	ABORT, DISABLE, PAUSE, PURGEMOTIONACTIVE, STOP	
<b>Example</b>	PURGEMOTION = 3            ;purge Pgen without decel then normalize Pgen	

---

<b>PURGEMOTIONACTIVE</b>	<b>Motion Purged</b>	<i>System</i>
--------------------------	----------------------	---------------

---

<b>Purpose</b>	Flag to indicate a purge motion request is being processed.
<b>Syntax</b>	<p><i>variable</i> = PURGEMOTIONACTIVE</p> <p><i>variable</i>            The <i>variable</i> will be set to ON or OFF. This is a read-only system flag that signals motion queue purging is complete.</p>
<b>Remarks</b>	FALSE (OFF) indicates that zero velocity has been reached. If decelerating to zero velocity, this flag will be ON until Pgen motion is no longer changing and all associated final position quantities have been assigned.
<b>See Also</b>	PURGEMOTION
<b>Example</b>	<pre>F1 = PURGEMOTIONACTIVE if (F1 = ON) {            ;perform other motion program functions while            ;Pgen motion is being purged. }</pre>

---

<b>PVEL1</b>	<b>Peak Motor Velocity</b>	<i>Velocity</i>
--------------	----------------------------	-----------------

---

<b>Purpose</b>	Peak velocity in user units per Timebase.
<b>Syntax</b>	<p><i>variable</i> = PVEL1</p> <p><i>variable</i>            The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>
<b>Remarks</b>	The motor velocity is measured every velocity loop update (1mS). If the present value is greater than the previous peak, the value is saved as the new peak. The magnitude of the peak is calculated based on an absolute value of VEL1. The value is saved and reported as a signed number, however. CLRPEAKS resets this value to zero.
<b>See Also</b>	VEL1, CLRPEAKS
<b>Example</b>	G5 = PVEL1

<b>PZONE</b>	Proportional Gain Zone, Velocity Loop	<i>Gain</i>
<b>Purpose</b>	Set the region in user units around the commanded position where KPZ is used instead of KP for the position regulator proportional gain. This allows the gain to be raised or lowered when the actual position is near the commanded position. If PZONE is zero, the KPZ gain is never active.	
<b>Syntax</b>	<p>PZONE = <i>distance</i>  <i>variable</i> = PZONE</p> <p><i>distance</i>                      Specifies a <i>distance</i> around the commanded position in user units. User units are defined by the SCALE parameter. The <i>distance</i> can be a constant, a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p> <p><i>variable</i>                      The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.	
<b>See Also</b>	KPZ, KP, KI, KIZ, IZONE	
<b>Example</b>	<pre>PZONE = 0.2 PZONE = V5</pre>	

<b>READ</b>	Read	<i>Operator Terminal</i>
<b>Purpose</b>	Read ASCII input from the Operator Terminal or a serial port.	
<b>Syntax</b>	<pre>READ [#n] [row, column] ["message"], [SHOW] variable [, field, precision[, F]] [(min, max)]</pre> <p><i>n</i>                              Serial port 1 or 2 is specified by <i>n</i>. If a port is not specified, the output defaults to serial port 1, the Operator Terminal port.</p> <p><i>row</i>                            The <i>row</i> that printing begins on. The top row is 1.</p> <p><i>column</i>                        The <i>column</i> that printing begins on. The left hand column is 1.</p> <p><i>message</i>                      The <i>message</i> is an optional character string that is printed before the input is accepted. The <i>message</i> must be in double quotation marks.</p> <p><i>SHOW</i>                         If <i>SHOW</i> is specified, the present value of the variable is displayed so you can check the value without entering a new value. To change the value, enter the new value followed by ENTER; to leave the present value unchanged, just press ENTER.</p> <p><i>variable</i>                      <i>variable</i> is where the information that is read is stored. <i>variable</i> may be a G or V variable or a B or F flag variable.</p> <p><i>field</i>                         <i>field</i> is the maximum number of characters to be entered for the variable including the decimal point. <i>field</i> does not apply when printing or reading flag variables.</p>	

<i>precision</i>	<i>precision</i> determines the number of places (1 to 4) after the decimal point. The maximum precision that will be permitted is 4. If <i>field</i> and <i>precision</i> are not included, the default precision is 2. <i>precision</i> does not apply when printing flag variables.
<i>F</i>	If <i>F</i> is used after <i>precision</i> , AND <i>precision</i> is set to 0, the variable will be printed as a fraction if the <i>variable</i> contains a fractional part. If the <i>variable</i> is a whole number, it will be printed as a decimal number even if <i>F</i> is specified. If <i>F</i> is used with <i>precision</i> from 1 to 4, the <i>variable</i> will be printed as a fraction unless the fractional part can be expressed as a decimal number. For example, 5 1/3 would be printed as 5 1/3 rather than 5.3333, but 5 1/4 would be printed as 5.25. Does not apply when printing flag variables.
<i>min,max</i>	min and max can be used to specify the limits of the input that will be accepted by the READ statement. min and max must be integer values. If the value input is not within the limits, a message will appear and a new value must be entered. min and max should not be used with any device other than the Allen-Bradley Operator Terminal.

**Remarks**

Unlike the PRINT statement, there must be a comma between the variable and any other arguments that come before it (the variable is not optional in a READ statement). Control codes may be printed in the *message* portion of a READ statement. See PRINT for more information.

If the SHOW option is specified when READing from a generic serial terminal, the ULTRA Plus or IQ will automatically send backspace characters and space characters to overwrite the old value when a character other than a carriage return (ENTER) is received. (If the SHOW option is used with an Allen-Bradley Operator Terminal, cursor positioning commands are automatically sent to erase the old value.)

**Note:** Use caution when using a READ statement specifying serial port 2. A READ statement can interfere with normal Host communications with an ULTRA Plus or IQ since characters received in port 2 while a READ statement is active will be interpreted as a value for the READ statement rather than a valid Host command. For example, if the IQ Master I/O status window is open, IQ Master is constantly sending and receiving data from port 2 to determine I/O status.

**See Also**

PRINT

**Example**

```
READ #1 1, 1 "The Length is ", V1, 3, 2
READ 1, 1, F1
READ F1
```

<b>REPEAT</b>	Repeat a LOOP	<i>Program Structure</i>
<b>Purpose</b>	Mark the end of a block of instructions to be repeated. The block instructions to be repeated must be in the range of 0 to 65535. A LOOP statement must define the start of the block of instructions.	
<b>Syntax</b>	REPEAT	
<b>Remarks</b>	The LOOP statement defining the block to be repeated must be in the range of 0 to 65535.	
<b>See Also</b>	LOOP	
<b>Example</b>	<pre>REPEAT RPT</pre>	

<b>RETURN</b>	Return from Subroutine	<i>Program Structure</i>
<b>Purpose</b>	Return from a subroutine to the point in the program from where the subroutine was called.	
<b>Syntax</b>	RETURN	
<b>Remarks</b>		
<b>See Also</b>	SUB	
<b>Example</b>	<pre>RETURN RET</pre>	

<b>RFDR</b>	Feedrate, runtime	<i>System</i>
<b>Purpose</b>	Present feedrate value in percent.	
<b>Syntax</b>	<pre><i>variable</i> = RFDR</pre> <p><i>variable</i>                      The <i>variable</i> may be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or a system variable.</p>	
<b>Remarks</b>		
<b>See Also</b>	FDR	
<b>Example</b>	<pre>V1 = RFDR</pre>	



<b>ROT</b>	<b>Positive Rotation Encoder 1</b>	<b>System</b>
<b>Purpose</b>	Flag to indicate the positive direction of rotation for encoder 1.	
<b>Syntax</b>	<i>variable</i> = ROT	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag. FALSE (OFF) indicates that clockwise (viewed facing the motor shaft) is the positive direction.
<b>Remarks</b>		
<b>See Also</b>	ROT2	
<b>Example</b>	F3 = ROT	

<b>ROTT</b>	<b>Tracking Rotation Direction</b>	<b>System</b>
<b>Purpose</b>	Set the direction of the Tracking function.	
<b>Syntax</b>	ROTT = ON/OFF ROTT = <i>variable</i> <i>variable</i> = ROTT	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	ON	Reverse rotation direction of the Tracking function.
	OFF	Normal rotation direction of the Tracking function.
<b>Remarks</b>		
<b>See Also</b>	TRACKINGMODE	
<b>Example</b>	ROTT = ON F3 = ROTT ROTT = F5	

<b>ROT2</b>	<b>Positive Rotation Encoder 2</b>	<b>System</b>
<b>Purpose</b>	Flag to indicate the positive direction of rotation for encoder 2.	
<b>Syntax</b>	<i>variable</i> = ROT2	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag. TRUE (ON) indicates the Rotation2 parameter is set to CCW.
<b>Remarks</b>	Refer to Part 2 • IQ Master Environment, Parameter menu, System.	
<b>See Also</b>	ROT	
<b>Example</b>	F3 = ROT2	

<b>Sn ON/OFF/CONT</b>	<b>Scanned Event</b>	<b>Scanned Event</b>
<b>Purpose</b>	Enable or disable a scanned event	
<b>Syntax</b>	<i>Sn ON/OFF/CONT</i>	
	<i>n</i>	The number of the scanned event: $n = 1, 2, 3, \dots, 8$ .
	<i>ON</i>	Enable the scanned event.
	<i>OFF</i>	Disable the scanned event.
	<i>CONT</i>	<i>CONT</i> will allow a scanned event to be recognized continuously, without waiting for the program to re-enable the event.
<b>Remarks</b>		
<b>See Also</b>	<i>Sn</i> : IF, <i>Sn</i> : TMR	
<b>Example</b>	S1 ON S2 CONT S1 OFF	

<b>Sn: IF</b>	<b>Scanned Event</b>	<i>Scanned Event</i>
<b>Purpose</b>	The scanned event IF statement is used in the same way as a normal IF statement but is preceded by <i>Sn:</i> , where <i>n</i> is a number from 1 to 8. These conditions are checked at every step of the program. If the scanned event is enabled AND the condition is found to be true, the specified action is taken.	
<b>Syntax</b>	<i>Sn: IF condition action</i>	
	<i>n</i>	The number of the scanned event. There are eight scanned events: <i>n</i> = 1, 2, 3, ..., 8.
	<i>condition</i>	<p>The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a flag being ON or OFF.</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre> I1 ON           ;an input F1 ON           ;a user flag INPOSN         ;system flag G1 &gt; G2         ;comparison user variable G1 &gt; POS1       ;comparison system variable POS2 &lt;= 7.00    ;comparison constant </pre>
	<i>action</i>	The program statements in a scanned event routine can be any legal program statement(s) except CALL, RETURN, JUMP, MOVD, MOVP, DV, DIF, DELAY, DWELL, PRINT, READ, ENABLE, and DISABLE statements. Multiple statements can be enclosed in curly braces { }.
<b>Remarks</b>	<p>To detect a transition of an input with a scanned event, the keyword EDGE is used with the IF statement. This will detect the transition of the input from OFF to ON, rather than just the ON state. EDGE can only be used in a scanned event.</p> <p>Scanned events are enabled in the program by <i>Sn</i> ON/OFF statements. Once a scanned event has occurred, it will not be recognized again until the condition has gone false and then true again or until another <i>Sn</i> ON statement is executed. <i>Sn:</i> routines must be placed before the main body of a program after any ASSIGN or TITLE statements.</p>	
<b>See Also</b>	<i>Sn:</i> TMR <i>m</i> , <i>Sn</i> ON/OFF/CONT	
<b>Example</b>	<pre> S1:  IF I13 EDGE ON V11 = PCMD S2:  IF POS1 &gt; = G10 04 = ON  S8:  IF I5 ON       { IF I9 ON 03=ON         ELSE 03=OFF       } </pre>	

<b><i>S<sub>n</sub>: TMR<sub>m</sub></i></b>	<b>Scanned Event</b>	<b><i>Scanned Event</i></b>
<b>Purpose</b>	A timer scanned event may be set up to start the timer based on a specified <i>condition</i> and run for a <i>time</i> . Once the programmed time has elapsed, the specified <i>action</i> is executed.	
<b>Syntax</b>	<i>S<sub>n</sub>: TMR<sub>m</sub> condition action</i>	
	<i>n</i>	The number of the scanned event. There are eight scanned events: <i>n</i> = 1, 2, 3, ..., 8.
	<i>m</i>	The number of the timer used: <i>m</i> = 1 or 2.
	<i>condition</i>	<p>The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a flag being ON or OFF.</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre> I1 ON                ;an input F1 ON                ;a user flag INPOSN              ;system flag G1 &gt; G2              ;comparison user variable G1 &gt; POS1            ;comparison system variable POS2 &lt;= 7.00        ;comparison constant </pre>
	<i>action</i>	The program statements in a scanned event routine can be any legal program statement(s) except CALL, RETURN, JUMP, MOVD, MOVP, DV, DIF, DELAY, DWELL, PRINT, READ, ENABLE, and DISABLE statements. Multiple statements can be enclosed in curly braces {}.
<b>Remarks</b>	The <i>time</i> is loaded during program execution using the TIMER <sub><i>n</i></sub> instruction. The scanned event is enabled and disabled by <i>S<sub>n</sub></i> ON/OFF commands.	
<b>See Also</b>	<i>S<sub>n</sub></i> : IF, <i>S<sub>n</sub></i> ON/OFF/CONT, TIMER <sub><i>n</i></sub>	
<b>Example</b>	<pre> S2: TMR1 I11 = ON F7 = ON S4: TMR2 POS2 &lt; G3 V1 = V1+1 </pre>	

<b><i>S<sub>n</sub>: TMR<sub>m</sub></i></b>	<b>Scanned Event</b>	<b><i>Scanned Event</i></b>
--	----------------------	-----------------------------

<b>Purpose</b>	A timer may also be used to determined the elapsed time between two events. In this case two <i>conditions</i> are specified. The timer starts counting up when the first event occurs and stops when the second event occurs.	
<b>Syntax</b>	<i>S<sub>n</sub>: TMR<sub>m</sub> start_condition, stop_condition action</i>	
<i>n</i>	The number of the scanned event. There are eight scanned events: $n = 1, 2, 3, \dots, 8$ .	
<i>m</i>	The number of the timer used: $m = 1$ or $2$ .	
<i>start_condition</i>	<p>The <i>condition</i> that must be satisfied to start the timer. The condition may be a comparison, an input being ON or OFF, or a flag being ON or OFF.</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre style="margin-left: 20px;"> I1 ON                               ;an input F1 ON                               ;a user flag INPOSN                             ;system flag G1 &gt; G2                             ;comparison user variable G1 &gt; POS1                           ;comparison system variable POS2 &lt;= 7.00                        ;comparison constant </pre>	
<i>stop_condition</i>	The <i>condition</i> that must be satisfied to stop the timer. Same type of conditions as <i>start condition</i> .	
<i>action</i>	The program statements in a scanned event routine can be any legal program statement(s) except CALL, RETURN, JUMP, MOVD, MOVP, DV, DIF, DELAY, DWELL, PRINT, READ, ENABLE, and DISABLE statements. Multiple statements can be enclosed in curly braces {}.	
<b>Remarks</b>	An action such as setting a flag or turning an output ON is then taken. The elapsed time between the two events is then available in the <code>TIMER<sub>n</sub></code> variable. The timer can be enabled or disabled in this mode also using the <i>S<sub>n</sub> ON/OFF/CONT</i> statement.	
<b>See Also</b>	<i>S<sub>n</sub>: IF, S<sub>n</sub> ON/OFF/CONT, TIMER<sub>m</sub></i>	
<b>Example</b>	<pre style="margin-left: 20px;"> S1:   TMR1 POS2 &gt; G1, POS2 &gt; G2       {       F3 = ON       O1 = OFF       } </pre>	

SCALE	Distance Scale	System
-------	----------------	--------

**Purpose**

This instruction defines the scale of the position feedback encoder, in steps per user unit. The scale is defined to produce the desired user units (e.g., inches, centimeters, revolutions) for all measured movements of the motor during the program. For this reason the SCALE instruction should be executed before the execution of any motion instruction. The required scale is calculated as follows: the number of encoder lines multiplied by 4 multiplied by the mechanical ratio of the drive train.

**Syntax**

SCALE = *value*  
*variable* = SCALE

*value*

The number of encoder counts per user unit of measure. *Value* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable. *Value* must be a whole number. Fractional scales are not permitted.

*variable*

The *variable* may be a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

**ATTENTION**

Do not change scale unless home has been defined. The scale statement should not be used while an ULTRA Plus or IQ is executing motion as unexpected motion may result. (The motor may jump if there is a difference between the new and old value.)

**Remarks**

**Example:** Assume that inches are the desired user unit. The motor has a 2000 line encoder and therefore generates 8000 counts per revolution. The drive train consist of a lead screw which makes 5 revolutions per inch of travel. The scale should be set to:

$$\text{Scale} = \frac{8000 \text{ counts}}{\text{motor rev.}} \times \frac{5 \text{ motor rev.}}{\text{inch}}$$

$$\text{Scale} = \frac{40,000 \text{ counts}}{\text{inch}}$$

Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, System dialog box.

**See Also**

SCALE2, CF, “Conversions” on page 131

**Example**

```
;assume the condition of the above example
SCALE = 40000 ;set scale to 40000 count / inch
```

<b>SCALE2</b>	<b>Distance Scale Axis 2</b>	<b>System</b>
<b>Purpose</b>	Set the scale used for encoder 2.	
<b>Syntax</b>	SCALE2 = <i>value</i> <i>variable</i> = SCALE2	
	<i>value</i>	The number of encoder counts per user unit of measure. <i>Value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable. <i>Value</i> must be a whole number. Fractional scales are not permitted
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	See discussion for SCALE. The SCALE2 statement should NOT be used while encoder 2 is in motion. Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, System dialog box.	
<b>See Also</b>	SCALE, CF, “Conversions” on page 131	
<b>Example</b>	SCALE2 = 16000	

<b>SEN</b>	<b>Soft Enable Flag</b>	<b>System</b>
<b>Purpose</b>	Soft enable Flag.	
<b>Syntax</b>	SEN= <i>value</i> <i>variable</i> = SEN	
	<i>value</i>	<i>Value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	If SEN is ON, the controller is ENABLED. If SEN is OFF, the controller is DISABLED.	
<b>See Also</b>	ENABLE, DISABLE	
<b>Example</b>	SEN = ON F7 = SEN	

<b>SETTASKTIMES</b>	System Task Timing	<i>System</i>
<b>Purpose</b>	Allows control of timing of system tasks.	
<b>Syntax</b>	SETTASKTIMES = <i>backgroundtasktime, optermtasktime, i/otasktime</i>	
	<i>backgroundtasktime</i>	The new start time in microseconds of the background task. The background task handles Host command language processing. The default time for this task is 1649 microseconds. The background task start time cannot be less than 1649 microseconds.
	<i>optermtasktime</i>	The new start time in microseconds of the operator terminal task. The default power up time for this task is 1697 microseconds. The operator terminal task handles Xkey routine execution, Fkey routine execution, operator terminal key input, displaying of the active operator terminal screen, read statements from serial port #1, and updating Operator Terminal status LED's.
	<i>i/otasktime</i>	The new start time in microseconds of the I/O task. The default power up time for this task is 1840 microseconds. The I/O task handles updating digital and analog input and output states, scanned event execution, position interrupt processing, and fault handling.
<b>Remarks</b>	<p>The SetTaskTimes statement can be used to modify default internal task scheduling times. It is only available in version 3.xx and higher firmware. The majority of applications will not have to modify background task times. Depending on the requirements of the application (main program execution, I/O processing, scanned events, Xkey routines, operator terminal update rate, host command language processing rate, etc.), the background task execution times can be tailored to better meet the requirements of the application. When specific functions are not required by the application, they can effectively be turned off to allow increased motion program throughput. The primary purpose of the SetTaskTimes statement is to allow for maximum motion program throughput. An examples of this is reducing background processing when performing a large number of computations within the motion program.</p> <p>No task start time greater than 1980 microseconds can be specified.</p> <p>Each task start time must be separated by at least 20 microseconds. In other words, task times must be programmed to allow each task to execute for at least 20 microseconds.</p> <p>Note:If any of the task time rules specified above are violated, the system will use default task start time values.</p> <p>If the motion program requires the default task times to be in effect, execute SETTASKTIMES = 0, 0, 0.</p>	
<b>See Also</b>		
<b>Example</b>	<pre>SETTASKTIMES = 1800, 1820, 1840 SETTASKTIMES = 0, 0, 0</pre>	



SINT2	Software Interrupt 2	<i>Interrupts</i>
-------	----------------------	-------------------

**Purpose****Syntax**

SINT2

**Remarks**

Software interrupt command. The SINT2 instruction will generate an interrupt 2 in the program, without the I12 input changing. INT2 does NOT have to be set ON or CONT in the program to use SINT2. SINT2 will latch the values of POS1 and POS2 into the variables I2P1 and I2P2, respectively. This is useful for applications where the relative position of encoders 1 and 2 is important, since the interrupt will record the two positions with minimum delay between reading POS1 and POS2.

**See Also****Example**

```
SINT2
V2 = I2P1 - I2P2
MOVD = V2
```

SLEW	Gear Slew Rate Limit	<i>Acceleration</i>
------	----------------------	---------------------

**Purpose**

Set the rate of change (acceleration) limit for the output of the gear function.

**Syntax**

SLEW = *value*  
*variable* = SLEW

*value*

The maximum acceleration that will be permitted as a result of a signal received from encoder 2. Slew is entered in user units per second per second. User units are defined by the SCALE parameter. *Value* can be a constant, a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

*variable*

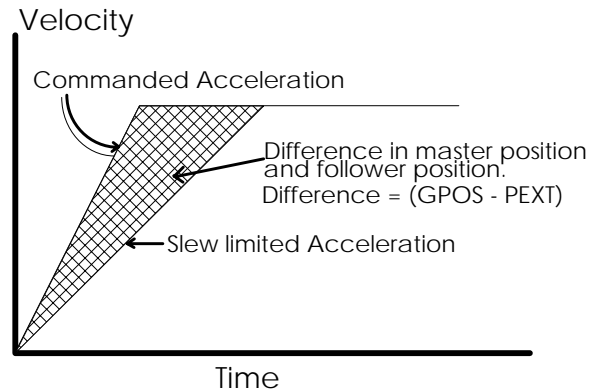
The *variable* may be a nonvolatile variable *Gn*, a volatile variable *Vn*, or a system variable.

**Remarks**

Use the SLEW limit to limit the maximum acceleration of a motor that is geared to the input from encoder 2. If the acceleration of the master motor would result in an acceleration on the follower that is greater than the slew limit, the command would be “clipped” to slew limit.

If the acceleration is clipped, the following axis will not track the encoder input exactly. For example: A gear ratio of 1 and a SLEW limit of 100 rev/s/s is established. If the master turns one revolution the follower will turn one revolution (if the acceleration limit of 100 rev/s/s is not exceeded).

Should the master exceed this limit, the follower will clip its acceleration to 100revs/s/s. At the end of 1 revolution of the master, the follower will stop at a distance less than 1 revolution. The diagram below illustrates this concept. The shaded area represents the difference in ideal position and the position that results because of the limit placed on the acceleration of the follower.



The SLEW limit is frequently used when the follower starts following a master that is already moving. If the master is moving and a gear is enabled with no SLEW limit set, the follower will use maximum acceleration to come up to speed. This can cause excessive wear on most machines and potentially damage equipment. If a SLEW limit has been set, the follower will use this acceleration rate to come up to the speed of the master

Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.

**See Also**

GEAR, SLEWEN

**Example**

SLEW = 500

<b>SLEWEN ON/OFF</b>	<b>Slew Enable</b>	<i>Motion</i>
<b>Purpose</b>	Enable or disable the slew rate limit for the output of the gear calculation.	
<b>Syntax</b>	SLEWEN = <i>value</i>	
	<i>value</i>	The <i>value</i> may be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , ON, OFF or another system flag. If <i>value</i> is equal to ON, enable the slew rate limit. If <i>value</i> is equal to OFF, disable the slew rate limit.
<b>Remarks</b>	<p>When the slew rate limit is enabled, if the rate of change of the gear output exceeds the limit, the command generated by the ULTRA Plus or IQ will be limited to the Slew variable. This will result in the ULTRA Plus or IQ not tracking the master input exactly because the follower system will not respond as quickly as the master when changing velocity.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.</p>	
<b>See Also</b>	SLEW, GEAR	
<b>Example</b>	<pre>SLEWEN = ON SLEWEN = OFF</pre>	

<b>SRESET</b>	<b>Soft Reset</b>	<i>System</i>
<b>Purpose</b>	Causes a soft reset when set ON.	
<b>Syntax</b>	SRESET = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	This command is only useful in an Fkey program.	
<b>See Also</b>	HRESET	
<b>Example</b>	SRESET = ON	

<b>STARTP</b>	<b>Start Program</b>	<b>System</b>
<b>Purpose</b>	Start program flag.	
<b>Syntax</b>	STARTP= <i>value</i>	
	<i>value</i>	<i>Value</i> can be ON, OFF, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag. Setting STARTP OFF has no effect.
<b>Remarks</b>	This instruction may only be used in Fkey programs.	
<b>See Also</b>	PGMNUM	
<b>Example</b>	PGMNUM = 3 STARTP = ON	
<b>STOP</b>	<b>Stop</b>	<b>System</b>
<b>Purpose</b>	Stops all motion and programs when set to ON.	
<b>Syntax</b>	STOP = <i>value</i>	
	<i>value</i>	The <i>value</i> can be ON, a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	<p>The following action will take place when a STOP = ON is executed. The STOP may come from an instruction in a program, a Host Language X command, the X Kill Motion Input or the Run Control Dialog box. When the STOP instruction is executed:</p> <ul style="list-style-type: none"> <li>• If a main program is running, the program is aborted.</li> <li>• If the motor is being commanded to move, it is decelerated to a stop. The rate of deceleration is determined by the system Acceleration variable, system Velocity variable and the present velocity. The system Velocity and Acceleration values are set in the Parameter menu, Velocity/Accel dialog box. The deceleration rate will equal:</li> </ul> $\text{System Feedrate Slew Time} = \frac{\text{System Velocity}}{\text{System Acceleration Rate}}$ $\text{Deceleration Rate} = \frac{\text{Present Velocity}}{\text{System Feedrate Slew Time}}$ <ul style="list-style-type: none"> <li>• After the Feedrate Slew Time (above), if a Jog input is ON the jogging motion will continue.</li> <li>• If a program was running AND outputs have default assignments to turn OFF (ON) when a program ends, they will turn OFF (ON).</li> <li>• If assigned, the Program Running output will turn OFF.</li> </ul> <p><b>Note:</b> The X input does not stop Jog input motion. Jog motion will slew to a stop and then begin again.</p>	
<b>See Also</b>	ABORT, DISABLE, PAUSE	
<b>Example</b>	STOP = ON	

<b>SUB</b>	<b>Subroutine</b>	<b>Program Structure</b>
<b>Purpose</b>	Mark the beginning of a subroutine and assign a name (up to eight characters) to the subroutine. Subroutines must end with a RETURN statement. Subroutines cannot be nested—that is, another subroutine cannot be called from within a subroutine. Program flow must return from the present subroutine before calling another subroutine.	
<b>Syntax</b>	<p>SUB <i>name</i></p> <p><i>name</i>                      Subroutine <i>names</i> can be up to 32 characters long. <i>Names</i> cannot be keywords of reserved words. They must begin with a letter and contain only alphanumeric characters and underscores ( _ ).</p>	
<b>Remarks</b>	Subroutines should be located at the end of program after all of the statements of the main program. An END statement should follow the RETURN of the last subroutine in a program. Subroutines of one program are not accessible to another program.	
<b>See Also</b>	RETURN, CALL	
<b>Example</b>	<pre>SUB MySub     ... Statements RETURN</pre>	

<b>SYSERROR</b>	<b>Cause System Error</b>	<b>System</b>
<b>Purpose</b>	Used to cause a system error. If set to a number, this will cause a system fault.	
<b>Syntax</b>	<p>SYSERROR= <i>value</i></p> <p><i>value</i>                      <i>Value</i> can be a constant, a nonvolatile flag <i>Gn</i>, a volatile <i>Vn</i>, or another system variable.</p>	
<b>Remarks</b>	Refer to the Appendix “Error Messages” on page 418 for a list of errors.	
<b>See Also</b>	FLAG, ENUM, WFLAG, WNUM, SYSWARN	
<b>Example</b>	<pre>SYSERROR = 35                      ;will cause an 'Excessive Speed' fault.</pre>	

<b>SYSWARN</b>	<b>Cause System Warning</b>	<b>System</b>
<b>Purpose</b>	Used to cause a system warning. If set to a value, a system warning occurs.	
<b>Syntax</b>	SYSWARN= <i>value</i>	
	<i>value</i>	<i>Value</i> can be a constant, a nonvolatile <i>Gn</i> , a volatile <i>Vn</i> , or another system variable.
<b>Remarks</b>	This command takes effect only when the Disable on Fault parameter is set to Partial. Refer to Part 2 • IQ Master Environment, Parameter menu, System for more information on the Disable on Fault parameter.	
<b>See Also</b>	EFLAG, ENUM, WFLAG, WNUM, SYSERROR	
<b>Example</b>	SYSWARN = 33 ;will cause an 'IAVG Fault' Warning	

<b>TBASE</b>	<b>Timebase</b>	<b>System</b>
<b>Purpose</b>	Indicates the number of position loop updates (2mS) in one unit of time. The unit of time is the Timebase and is set the Parameter menu, Velocity/ Accel dialog box.	
<b>Syntax</b>	<i>variable</i> = TBASE	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable. A value of 500 indicates the Timebase is in seconds. A value of 30,000 indicates a Timebase of minutes.
<b>Remarks</b>	This variable may be used in a program to determine if velocity units are per minute or per second.	
<b>See Also</b>		
<b>Example</b>	V3 = TBASE	

<b>TIMER<math>n</math></b>	<b>Timer</b>	<b>Scanned Events</b>
<b>Purpose</b>	Load TIMER $n$ with the time specified in this instruction, or read the present value.	
<b>Syntax</b>	TIMER $n$ = <i>time</i> <i>variable</i> = TIMER $n$	
	<i>n</i>	The number of the timer: $n = 1$ or $2$ .
	<i>time</i>	The <i>time</i> is entered in seconds regardless of the setting of the user Timebase. <i>Time</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable
	<i>variable</i>	The <i>variable</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
<b>Remarks</b>	Timers have a resolution of .002 seconds (2 ms).	
<b>See Also</b>	<i>Sn</i> : TMR	

**Example**

```
TIMER1 = 0.250
TIMER2 = V23
G3 = TIMER2
```

<b>TITLE</b>	Program Title	<i>Program Structure</i>
<b>Purpose</b>	Assign a name to a program.	
<b>Syntax</b>	TITLE " <i>name</i> "	
	<i>name</i>	The <i>name</i> can be up to eight characters long. <i>name</i> cannot be keywords or reserved words. <i>name</i> must begin with a letter and contain only alphanumeric characters and underscores ( _ ).
<b>Remarks</b>	The <i>name</i> is shown in the program directory. The TITLE statement must be the first statement in a program and <i>name</i> must be in double quotation marks. The title of function key programs will show up in the Fkey Set Up dialog box. <b>TIP:</b> Program names are not required, however, the compiler will issue a warning message if a program title is not present.	
<b>See Also</b>		
<b>Example</b>	TITLE "PROG1" TITLE "PART3"	

<b>TRACKINGEXTPOSSELECT</b>	Tracking Source Select	<i>System</i>
<b>Purpose</b>	Select the source of the Tracking function.	
<b>Syntax</b>	TRACKINGEXTPOSSELECT = <i>ON/OFF</i> TRACKINGEXTPOSSELECT = <i>variable</i> <i>variable</i> = TRACKINGEXTPOSSELECT	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
	<i>ON</i>	The Tracking function is applied after the Gear and Slew functions.
	<i>OFF</i>	The Tracking function is applied before the Gear and Slew functions.
<b>Remarks</b>		
<b>See Also</b>	TRACKINGMODE	
<b>Example</b>	TRACKINGEXTPOSSELECT = ON F3 = TRACKINGEXTPOSSELECT ROTT = F5	

<b>TRACKINGMODE</b>	<b>Tracking</b>	<i>Motion</i>
<b>Purpose</b>	Set the directional gearing mode of the Tracking function.	
<b>Syntax</b>	TRACKINGMODE = <i>n</i> <i>variable</i> = TRACKINGMODE	
	<i>n</i>	<i>n</i> specifies which of the modes to set. Values for <i>n</i> are 0 through 2.
	<i>variable</i>	<i>Value</i> can be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or another system variable.
<b>Remarks</b>	<p>Mode 0 is normal gearing.</p> <p>Mode 1 is when any forward motion of the master results in forward follower motion, much like a ratchet.</p> <p>Mode 2 is similar to mode 1 except if the master moves in reverse, then forward again, the follower will not move until the master returns to the position where it began moving reverse.</p>	
<b>See Also</b>	ROTT, TRACKINGSYNCED, TRACKINGEXTPOSSELECT, GEAR, GEAREN, SLEW, SLEWEN	
<b>Example</b>	<pre>TRACKINGMODE = 2;Set mode number 2 V1 = TRACKINGMODE;Read present mode</pre>	

<b>TRACKINGSYNCED</b>	<b>Tracking</b>	<i>System</i>
<b>Purpose</b>	ON indicates the Tracking function is not limiting gearing motion.	
<b>Syntax</b>	<i>variable</i> = TRACKINGSYNCED	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag <i>Bn</i> , a volatile flag <i>Fn</i> , or another system flag.
<b>Remarks</b>	If this flag is off, the master has backed up and the tracking function is limiting follower motion.	
<b>See Also</b>	TRACKINGMODE	
<b>Example</b>	F3 = TRACKINGSYNCED	



---

<b>UTOC1</b>	<b>Convert user units to counts</b>	<i>Conversion</i>
--------------	-------------------------------------	-------------------

---

<b>Purpose</b>	Convert a variable from user units to encoder counts using SCALE.	
<b>Syntax</b>	$variable\_1 = UTOC1\ variable\_2$	
	$variable\_1$	$Variable\_1$ can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable
	$variable\_2$	$Variable\_2$ can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , another system variable, or a constant.
<b>Remarks</b>		
<b>See Also</b>	@, UTOC2, CTOU1, CTOU2, SCALE, “Conversions” on page 131	
<b>Example</b>	$V3 = UTOC1\ V4$	

---

<b>UTOC2</b>	<b>Convert user units to counts</b>	<i>Conversion</i>
--------------	-------------------------------------	-------------------

---

<b>Purpose</b>	Convert a variable from user units to encoder counts using SCALE2.	
<b>Syntax</b>	$variable\_1 = UTOC2\ variable\_2$	
	$variable\_1$	$Variable\_1$ can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , or another system variable.
	$variable\_2$	$Variable\_2$ can be a nonvolatile variable $G_n$ , a volatile variable $V_n$ , another system variable, or a constant.
<b>Remarks</b>		
<b>See Also</b>	@, UTOC1, CTOU1, CTOU2, SCALE2, “Conversions” on page 131	
<b>Example</b>	$V3 = UTOC2\ V4$	

<b>Vn</b>	Variable, Volatile, User	Variable
<b>Purpose</b>	Load a user-defined variable with a value from another variable, a result of a math expression, a constant, a timer, or an analog input.	
<b>Syntax</b>	$Vn = value$ $variable = Vn$	
	<i>n</i>	The number of the variable: $n = 1, 2, 3, \dots, 64$ .
	<i>value</i>	<i>Value</i> can be a constant, a global variable $Gn$ , a volatile variable $Vn$ , or a system variable.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable $Gn$ , a volatile variable $Vn$ , or a system variable.
<b>Remarks</b>	<p>G and V variables are stored internally as 4 bytes for the mantissa (the part before the decimal point), 2 bytes for a numerator (the top part of a fraction), and 2 bytes for a denominator (the bottom part of a fraction). The range of numbers that can be represented in 4 bytes is <math>\pm 2,147,483,648</math>. The range of numbers that can be represented in 2 bytes is <math>\pm 32,767</math>. Therefore, the range of numbers that can be contained in a G or V variable and any intermediate value is:</p> $G \text{ or } V = \pm 2,147,483,648 + \frac{\pm 32,767}{\pm 32,767}$ <p>The changes made to <math>Vn</math> remain in effect as long as power is maintained. On power-up <math>Vn</math> is set to zero. If a value needs to be maintained during a power failure use a <math>Gn</math> user variable. Changes made to <math>Vn</math> by one program will be seen by other programs.</p> <p><b>TIP:</b> Variables can be assigned names using the ASSIGN command and then referenced by those names to make the program easier to read.</p>	
<b>See Also</b>	$Gn, Fn, Bn$	
<b>Example</b>	<pre>V7 = ADC1 V14 = PCMD V7 = V7 + 1 V8 = TIMER1 V9 = POS1</pre>	

<b>VCMD</b>	<b>Velocity Command</b>	<i>Velocity</i>
<b>Purpose</b>	Velocity command in user units per Timebase.	
<b>Syntax</b>	<i>variable</i> = VCMD	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	VCMD may not go to zero when there is no commanded position. VCMD represents the value that would be sent to a digital to analog converter as a velocity command if an external amplifier were used. Therefore this command may have an offset that prevents it from returning to zero when the position command is not changing.	
<b>See Also</b>	VEL1	
<b>Example</b>	V1 =VCMD	

<b>VEL</b>	<b>Velocity</b>	<i>Velocity</i>
<b>Purpose</b>	Change the present velocity used for MOVD and MOVP calculations.	
<b>Syntax</b>	VEL = <i>value</i> <i>variable</i> = VEL	
	<i>value</i>	The <i>value</i> is entered in user units per Timebase. User units are defined by the SCALE parameter. The <i>value</i> can be a constant, a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable <i>Gn</i> , a volatile variable <i>Vn</i> , or a system variable.
<b>Remarks</b>	Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Velocity/Accel dialog box.	
<b>See Also</b>	JVEL, HVEL	
<b>Example</b>	VEL = 750 VEL = G4	

<b>VEL1</b>	<b>Motor Velocity</b>	<i>Velocity</i>
<b>Purpose</b>	Velocity feedback from encoder 1 in user units per Timebase.	
<b>Syntax</b>	<i>variable</i> = VEL1	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
<b>Remarks</b>	The velocity of encoder 1 is determined every velocity loop update (1mS). This value is stored in VEL1.	
<b>See Also</b>	FVEL1, VCMD, VEL2	
<b>Example</b>	V1 = VEL1	

<b>VEL2</b>	<b>Velocity Encoder 2</b>	<i>Velocity</i>
<b>Purpose</b>	Encoder 2 velocity in user units per Timebase.	
<b>Syntax</b>	<i>variable</i> = VEL2	
	<i>variable</i>	The <i>variable</i> may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
<b>Remarks</b>		
<b>See Also</b>	VEL1, VCMD	
<b>Example</b>	V1 = VEL2	

<b>WAIT</b>	Wait	<i>Program Structure</i>
-------------	------	--------------------------

<b>Purpose</b>	Wait for a condition to occur before proceeding to the next instruction.
<b>Syntax</b>	<p>WAIT <i>condition</i></p> <p><i>condition</i>      The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a flag being ON or OFF.</p> <p>Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (&gt;), less than (&lt;), equal to (=), not equal to (&lt;&gt;), less than or equal to (&lt;=), or greater than or equal to (&gt;=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.</p> <pre style="margin-left: 20px;"> I1 ON                            ;an input F1 ON                            ;a user flag INPOSN                          ;system flag G1 &gt; G2                         ;comparison user variable G1 &gt; POS1                       ;comparison system variable POS2 &lt;= 7.00                   ;comparison constant </pre>
<b>Remarks</b>	
<b>See Also</b>	DWEELL, DELAY
<b>Example</b>	<pre style="margin-left: 20px;"> WAIT I12 = ON WAIT POS1&gt; = G3 </pre>

<b>WFLAG</b>	Warning Flag	<i>System</i>
--------------	--------------	---------------

<b>Purpose</b>	Flag to indicate if a warning is active.
<b>Syntax</b>	<p>WFLAG= <i>value</i></p> <p><i>value</i>      Value can be ON, OFF, a nonvolatile flag Bn, a volatile flag Fn, or another system flag.</p>
<b>Remarks</b>	
<b>See Also</b>	EFLAG, ENUM, WNUM, SYSERROR, SYSWARN
<b>Example</b>	<pre style="margin-left: 20px;"> IF WFLAG = ON ° </pre>

<b>WHILE</b>	While	<i>Program Structure</i>
--------------	-------	--------------------------

<b>Purpose</b>	The WHILE <i>condition</i> statement executes the <i>statement(s)</i> repeatedly while the condition specified is TRUE.
----------------	---

<b>Syntax</b>	WHILE <i>condition statement(s)</i>
---------------	-------------------------------------

<i>condition</i>	The <i>condition</i> to be tested. The <i>condition</i> may be a comparison, an input being ON or OFF, or a flag being ON or OFF.
------------------	---

Comparisons compare the values of two operands and determine if the condition is TRUE or FALSE. A comparison may be greater than (>), less than (<), equal to (=), not equal to (<>), less than or equal to (<=), or greater than or equal to (>=). The operands of a comparison may be user variables, system variables, analog input values (ADC), or constants.

```
I1 ON           ;an input
F1 ON           ;a user flag
INPOSN         ;system flag
G1 > G2         ;comparison user variable
G1 > POS1       ;comparison system variable
POS2 <= 7.00   ;comparison constant
```

<i>statement(s)</i>	Any valid language statement. Multiple statements can enclosed in curly braces {}.
---------------------	--

<b>Remarks</b>	Unlike the Do While statement, the condition is tested before any statement in the loop is executed. If the condition is FALSE when the loop begins, the loop body is never executed.
----------------	---

<b>See Also</b>	DO/WHILE, IF
-----------------	--------------

<b>Example</b>	<pre>WHILE (POSN&lt;120)           ;execute statements while the     {                       ; position is less than 120         ... Statements     }</pre>
----------------	---

<b>WINDOW</b>	In-Position Window	<i>System</i>
<b>Purpose</b>	Change the present In-Position Window used in the program.	
<b>Syntax</b>	WINDOW = <i>value</i> variable = WINDOW	
	<i>value</i>	Specifies a distance around the commanded position in user units. User units are defined by the SCALE parameter. The distance can be a constant, a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
	<i>variable</i>	The variable may be a nonvolatile variable Gn, a volatile variable Vn, or a system variable.
<b>Remarks</b>	<p>The WINDOW setting in conjunction with the In-Position mode determine the operation of the INPOSN flag and the In-Position output (Output 07 if assigned). If the In-Position mode is set to relative, when the Following Error (FE) is less than WINDOW, the INPOSN flag is set ON and the INPOSN output is turned ON if assigned. If the In-Position mode is set to Absolute, motion will also have to be stopped for the flag and the output to be set. The In-Position mode is set in the Parameter menu, Outputs dialog box.</p> <p>Changes made to this variable by a program are only in effect while the program is running. When the program ends it will return to the value stored in the Personality Module. If no change is made to this variable by the program, the value in the Personality Module is used. The value in the Personality Module is set in the Parameter menu, Gains/Limits dialog box.</p>	
<b>See Also</b>	INPOSN	
<b>Example</b>	WINDOW = 0.005 WINDOW = V7	

<b>WNUM</b>	Warning Number	<i>System</i>
<b>Purpose</b>	Currently active warning number.	
<b>Syntax</b>	variable = WNUM	
	<i>variable</i>	<i>Variable</i> can be a nonvolatile flag Bn, a volatile flag Fn, or another system flag.
<b>Remarks</b>	Refer to the appendix “Error Messages” on page 418 for a list of errors.	
<b>See Also</b>	EFLAG, ENUM, WFLAG, SYSERROR, SYSWARN	
<b>Example</b>	V4 = WNUM	

<b><i>Xn:</i></b>	<b>Xkey</b>	<b><i>Program Structure</i></b>
-------------------	-------------	---------------------------------

**Purpose** Marks the beginning of an Xkey routine.

**Syntax** *Xn:*

**Remarks** The keys marked X1 through X4 on the Operator Terminal are special function keys that, when pressed, execute a routine in parallel with a main program. To enable an Xkey, a routine must be written for the Xkey and the program must contain an *Xn ON* or *Xn CONT* statement.

The program statements in an Xkey routine can be any legal program statement except motion statements or subroutine calls. Motion statements include *MOVD*, *MOVP*, *DV*, *DIF*, *DELAY*, and *DWELL*. Any jump commands in an Xkey routine must be to a label within that same Xkey routine; jumps outside of the Xkey routine are not allowed. *PRINT* and *READ* commands for the Operator Terminal are allowed.

If more than one Xkey is enabled, if a second Xkey is pressed while the first Xkey routine is still executing, the first Xkey routine will be aborted and execution of the second will begin. *Xn ON*, *Xn OFF*, and *Xn CONT* statements can be used within an Xkey routine to avoid this situation.

The *Xn* statement is used to mark the beginning of a routine for an Operator Terminal Xkey, where *n* is 1 through 4. Xkey routines must be placed before the main body of a program after any *TITLE*, *ASSIGN*, or *Sn:* statements.

The *XNPGM = number* statement can be used to simulate pressing an Xkey from within a program.

**See Also** *Xn ON/OFF/CONT*, *XEND*, Operator Terminal

**Example**

```

X1:                                ;Beginning of the X1 key routine
      X1 OFF                        ;Disable Xkey routine while in it
      ; (needed if X1 CONT used)
      X2 OFF                        ;Disable X2 while in X1 routine
      ... statements
      X2 ON                          ;Re-enable X2 key
      X1 ON                          ;Re-enable X1 key
XEND                                ;Mark the end of the X1 routine

```



<b><i>Xn ON/OFF/CONT</i></b>	<b>Xkey Routine, Enable</b>	<b><i>Program Structure</i></b>
<b>Purpose</b>	Enable or disable an Xkey.	
<b>Syntax</b>	<i>Xn ON/OFF/CONT</i>	
	<i>n</i>	Specify the number of the Xkey routine: $n = 1, 2, 3, 4$ .
	<i>ON</i>	After <i>Xn ON</i> is executed, one Xkey press will be recognized and then the Xkey will be disabled. Another <i>Xn ON</i> statement must be executed to re-enable the Xkey.
	<i>OFF</i>	Disable the Xkey routine.
	<i>CONT</i>	<i>CONT</i> allows a Xkey to be recognized continuously, without waiting for the program to re-enable the Xkey.
<b>Remarks</b>	The <i>XNPGM = number</i> statement can be used to simulate pressing an Xkey from within a program.	
<b>See Also</b>	<i>Xn</i> , <i>XNPGM</i>	
<b>Example</b>	<pre>X2 ON X3 CONT X2 OFF</pre>	

<b><i>XEND</i></b>	<b>Xkey Routine End</b>	<b><i>Program Structure</i></b>
<b>Purpose</b>	Mark the end of an Xkey routine within a program. An Xkey routine must end with an <i>XEND</i> statement.	
<b>Syntax</b>	<i>XEND</i>	
<b>Remarks</b>	The <i>XNPGM = number</i> statement can be used to simulate pressing an Xkey from within a program.	
<b>See Also</b>	<i>Xn</i> , <i>XNPGM</i> , Operator Terminal	
<b>Example</b>		

XLOOPINDEX	X-Routine Loop Index	System
------------	----------------------	--------

<b>Purpose</b>	The LOOP counter for the LOOP instruction used in the Xkey routine.
<b>Syntax</b>	<p>XLOOPINDEX= <i>value</i>  <i>variable</i> = XLOOPINDEX</p> <p><i>value</i>                      <i>Value</i> can be a constant, a global variable <i>Gn</i>, a volatile variable <i>Vn</i>, or another system variable.</p> <p><i>variable</i>                    <i>Variable</i> can be a nonvolatile variable <i>Gn</i>, a volatile variable <i>Vn</i>, or another system variable.</p>
<b>Remarks</b>	<p>Mark the end of a block of instructions to be repeated in a loop. The range of the loop counter is 0 to 65535. If <i>count</i> is 0 the statements in the loop will not execute.</p> <p>The LOOP statements cannot be used in Fkey or Scanned Event routines. If a subroutine is called from a LOOP, the subroutine may not contain a LOOP.</p> <p>The XNPGM = <i>number</i> statement can be used to simulate pressing an Xkey from within a program.</p>
<b>See Also</b>	LOOP, LOOPINDEX, REPEAT
<b>Example</b>	<pre>X1 :                 LOOP 3                 V4 = XLOOPINDEX                 PRINT V4                 REPEAT XEND</pre>

XNACTIVE	Xkey Routine Active	System
----------	---------------------	--------

<b>Purpose</b>	Xkey active flag. This flag indicates when an Xkey on the Operator Terminal is pressed.
<b>Syntax</b>	<p><i>variable</i> = XNACTIVE</p> <p><i>variable</i>                    The <i>variable</i> may be a nonvolatile flag <i>Bn</i>, a volatile flag <i>Fn</i>, or another system flag.</p>
<b>Remarks</b>	ON indicates that an Xkey is pressed. OFF indicates there are no Xkeys pressed. The XNPGM = <i>number</i> statement can be used to simulate pressing an Xkey from within a program.
<b>See Also</b>	FNACTIVE, <i>Xn</i> , Operator Terminal
<b>Example</b>	WAIT XNACTIVE = OFF

**XNPGM**

## Run an Xkey Routine

*Program Structure*

<b>Purpose</b>	Simulates pressing an Xkey from within a program if the Xkey routine has been enabled previously with an <i>Xn</i> ON or <i>Xn</i> CONT statement.
<b>Syntax</b>	<i>XNPGM</i> = <i>number</i>  <i>number</i> The number of the Xkey routine to be run: <i>number</i> = 1, 2, 3, 4.
<b>Remarks</b>	Only one Xkey routine can be running at a time. If an Xkey routine is running when the <i>XNPGM</i> statement is executed, the present Xkey routine ends and the new Xkey routine is started.
<b>See Also</b>	<i>Xn</i> , <i>Xn</i> = ON/OFF/CONT, Operator Terminal
<b>Example</b>	<i>XNPGM</i> = 1 <i>XNPGM</i> = 4



# Part 6

# IQ Host Command Language

This part contains a detailed description of the host mode communication capability of the ULTRA Plus or IQ-Series controller. The host mode is used if there is a host computer or PLC that will control operation of the controller.

HOST MODE

HOST MODE

# Introduction

---

## Overview

The Host Command Language specifies the command syntax and protocol used for serial communications with an ULTRA Plus or IQ. It allows a smart host device to operate the ULTRA Plus or IQ. The language is able to perform the following functions:

- control motion and program execution
- monitor status and other runtime information
- execute diagnostics
- set up all nonvolatile memory parameters, programs, etc.
- set up runtime parameters and flags required for control
- control any peripheral devices attached to the ULTRA Plus or IQ.

The Host Command Language serves as the only interface between the ULTRA Plus or IQ and the host. Since this is the only interface available, the command parsing and execution speed has been optimized to provide true host mode operation.

IQ Master uses the Host Command Language to communicate with the ULTRA Plus or IQ.

The IQ Host Command Language only applies to Serial Port 2 (P6).

Many of the commands that are available as the host commands are also available as commands, variables, and flags within an application program. Most variables and flags detailed in the Language Reference are available as host mode commands with the same name. Refer to the Language Reference for a complete description of each variable and flag.

## Reserved Host Commands

The following host commands are part of IQ Cam. Refer to the *IQ-Series IQ-Cam Software Manual* (Part Number 0013-1053-001) for a complete description.

CAM	CAMAUTOSYNC	CAMEN	CAMERROR
CAMEXTPOSSELECT	CAMINPOSNACCEL	CAMQUEUEFULL	CAMSTARTACCEL
CAMSTOPDECEL	CAMSTOPTIMEDECEL	CAMSYNCED	CAMSYNCEDPOSLIMIT
CAMSYNCEDTIME	CAMSYNCMAXVEL	CAMTIMEBASE	CGENEN
CJOGEN	CPOS	CURONESHOTDEF	CURPLSDEF
CURPROFILEDEF	CURTIMELOCKEDDEF	CYCLESPPENDING	CYCLESPPERTRIGGER
DISCONTINUOUSCOUNTS	DISCONTINUOUSPROFILE	FCYCLEPOS	GLOBALPFDISP
GLOBALMCYCLE	INHIBITCORRECTIONS	INHIBITPROFILE	MCYCLE
MCYCLEPOS	MCYCLETIME	NEGSYNCDIR	OFFSETALWAYS
ONESHOTWAITING	PFDISP	PLSEN	PROFILEACTIVE
PROFILECOMPLETE	PROFILECOUNT	PROFILENUM	RELSYNC
ROTF	ROTM	SHORTESTDIST	STARTSYNCMODE
STOPSYNCMODE	SYNCOFFSETPOS	SYNCOFFSETTIME	VELLOCKPOSError
VLAUTOCORRECT	VLAUTOCORRECTVEL	VLCORRECTIONCMPLT	

## Implementation

Incoming commands and outgoing responses execute through interrupt driven circular buffers. A command is parsed and executed when a carriage return is received. Since the Host Command Language is the only interface available to the ULTRA Plus or IQ, the use of an ASCII command set allows it to be used not only by a smart host, but also as a simple control method in a dumb terminal environment.

Some control characters remain available and are used to carry out XON/XOFF handshaking, axis selection (when multiple ULTRA Plus or IQ units are on a single serial line), backspace, and ACK/NACK.

The command parser and executor run as a background task of the ULTRA Plus or IQ. The ULTRA Plus or IQ interfaces to the host only through internally defined flags and variables, as necessary, for control purposes. It is a passive interface in that it only responds to commands and never initiates transmissions. The ULTRA Plus or IQ can initiate transmissions with a program PRINT statement.

When nonvolatile memory data is changed, the ULTRA Plus or IQ updates the checksum information contained in the personality module. The update happens automatically after the data has been changed; no further host action is required.

## Addressing

An ULTRA Plus or IQ is normally configured to address zero for single axis communication. When set to address zero, the ULTRA Plus or IQ will respond to all commands. When multiple ULTRA Plus or IQ units are connected on a single serial link in either RS-232C daisy chain or RS-422 multi-drop configuration, only one ULTRA Plus or IQ should transmit at a time. To accomplish this, each ULTRA Plus or IQ is set to a different address and a special character sequence is transmitted to activate an individual ULTRA Plus or IQ.

Each drive must have a unique address (1-63) set using dip switch SW1. Refer to “Networking IQ-



Series Controllers” in the appropriate *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004) for details on setting an address and wiring the ULTRA Plus or IQ. Refer to Part 2 • IQ Master Environment, Communications menu, Axis Select for setting IQ Master to communicate with an ULTRA Plus or IQ set to an address other than zero.

A two character sequence is used to select an ULTRA Plus or IQ by address. A Control-B character (ASCII STX, Hexadecimal 02) followed by the address number (01 through 3F Hex) will select an individual ULTRA Plus or IQ. All ULTRA Plus or IQ units with addresses other than the one selected will not transmit and will ignore all serial commands other than the address select command.

Example: An ULTRA Plus or IQ is configured to address 3. The two character sequence, 02 Hex and 03 Hex, will select the ULTRA Plus or IQ.

**Note:** A drive configured to address 2 requires the sequence, 02 Hex followed by another 02 Hex character. Even though the same character is used, each has a different purpose. The first 02 Hex character is the address command and the second is the actual address

Even if an ULTRA Plus or IQ is configured with a non zero address, the address zero may be used to transmit identical commands to multiple ULTRA Plus or IQs. Address zero may be selected by transmitting the sequence 02 Hex, 00 Hex. (00 Hex may be sent on most PC keyboards by pressing Control-@.) When address zero is selected, and SW1 is set to an address other than zero, all ULTRA Plus or IQ units will respond to commands, but no ULTRA Plus or IQ will transmit data back to the host or echo the incoming command. This mode is useful to tell all ULTRA Plus or IQ units to simultaneously perform a similar function. For example, sending “RUN 1” to all ULTRA Plus or IQ units, will cause all ULTRA Plus or IQs to run their own program number 1.

On power up, any ULTRA Plus or IQ with SW1 set to an address other than zero will neither transmit nor act on an incoming command other than an address select command. In other words, no ULTRA Plus or IQ is selected. Therefore, to communicate to an ULTRA Plus or IQ with SW1 set to an address other than zero after power-up, the host must select either address zero or the address set on SW1.

If two or more ULTRA Plus or IQ units are set to the same address on a serial link, scrambled communications will result. Exercise caution when setting communication addresses to avoid duplicate addresses.

# Syntax

This section details the syntax of commands expected by the ULTRA Plus or IQ and the responses to those commands.

---

## Command Syntax

The ULTRA Plus or IQ determines the command type and the required action(s) from the syntax of the incoming command. Incoming commands are echoed to the host unless the ULTRA Plus or IQ is in a multi-drop mode of operation and not selected, or in a multi-drop mode and processing a global command.

The Host Command Language is implemented with three general types of commands:

- Execution - requires some action to be taken by the ULTRA Plus or IQ.
- Assignment - sets the value of variables or flags or structures.
- Request - reads the value of variables or flags or structures.

All commands have the basic syntax:

**Syntax**                      *[WS] COMMAND [[WS] [=] data[, data]] [WS] [SUM] END*

*COMMAND*                      The command field is the name of the host command. It is case insensitive alphanumeric with the first character always a letter.

HOST MODE

<i>WS</i>	<i>WS</i> (white space) may be an ASCII space or tab. It is ignored except where required to differentiate between data items associated with the command.
<i>data</i>	The data field is present in all assignment commands and specifies the value of the associated parameter. This field is most often numeric although a few commands use text strings as data. Numeric data is decimal unless specified otherwise for a specific command. Flag values are set as 1 = TRUE and 0 = FALSE. The equal sign “=” is optional in assignment commands. When a command requires multiple parameters, the parameters are separated by white space or a comma.
<i>SUM</i>	<p><i>SUM</i> is an optional field consisting of a caret “^” followed by two hexadecimal characters. If present, the <i>SUM</i> is the eight bit binary sum of all characters in the command line up to but not including the “^”. When the sum is summed with the preceding characters, the result must be zero. Any characters between the sum and the <i>END</i> field are ignored. Any letters in the sum must be in upper case. If data is being requested, it will be returned with a <i>SUM</i> check only if one was present in the requesting command.</p> <p>If the <i>SUM</i> indicates an error in the command line, the command line will not be processed and the appropriate error response will be returned.</p> <p>WARNING: To guarantee data integrity, the checksum must be used in host communications with the ULTRA Plus or IQ.</p>
<i>END</i>	The <i>END</i> field consists of an ASCII carriage return (^M) followed by an optional linefeed character (^J).

**Remarks**

Items in brackets indicate optional parameters.

---

## Response Syntax

A response is returned to all command types unless the ULTRA Plus or IQ is not echoing in a multi-drop mode. In this case, it is up to the controller to take any action required to determine compliance with the command. For instance, a global command could be issued to start motion of all axes, and then each axis could be selected and checked for proper program execution. There are three main types of responses:

- Acknowledge
- Value
- Error.

## Acknowledge

The acknowledge response is used as a simple handshake acceptance of all execution and assignment commands that are accepted without error. It consists of an ASCII ACK (^F) control character.

## Value

The value response returns variable information to request commands. It consists of the ACK character, the value(s) requested, an optional *SUM*, and an END sequence. The value of the checksum, if present, when summed with the characters in value, will be zero. (Note: the ACK character is not included in the *SUM*.) The END field for a value response will always be present.

## Error

The error response is used for all commands that could not be accepted. It consists of an ASCII NAK (^U) character, and an “E”, followed immediately by a decimal error number. If the HOST runtime command flag is FALSE (0) an appropriate error message in the format “: message” will follow the error number. The error response is terminated with the END field. The following decimal error codes are returned following a negative acknowledge error response:

Error Code	Description
0	Unrecognized command
1	Syntax error in data or missing data
2	Invalid data or data out of range
3	Cannot execute command in current operating mode
4	Command or file, data checksum error
5	Illegal access of read-only or write-only variable
6	Illegal program number
7	Out of memory during file load
8	File transfer error or transfer aborted

# Execution Commands

Execution commands are directives to the ULTRA Plus or IQ that require some internal action to be taken. These commands are not queued and must be executed as received. An error acknowledge will result if the command cannot be executed at the time it was received (for example, attempting to execute a program when one is already running). Unnecessary or duplicated commands such as disabling the drive when it is already disabled will be passively accepted without error.

---

## Motion Execution Commands

**HOME** (*System Program #25*)

**ERET** (*System Program #26*)

These commands cause the Auto, Home, or Emergency Return to Position programs to execute and are alternate methods of executing programs from the system directory. To execute other programs from the system directory (including the Error program #27), the SRUN command must be used and the program number specified.

**MOVD or MD** *distance* [[, ][V=] *velocity*]

**MOVP or MP** *position* [[, ][V=] *velocity*]

Executes a motion profile of the specified distance (MOVD, MD) or to the specified position (MOVP, MP). The velocity parameter is optional; if not specified, the default velocity will be used.

**MOVV or MV** *velocity*

Executes a move at the specified velocity using the jog acceleration. Motion will continue until stopped by another MOVV, an X command, or Inputs. If a MOVV is executing, it will accelerate to the new velocity.

**RUN** [*program*]

**SRUN *program***

Executes the specified program number. If no program number is given with the Run command, the command simulates the action of the “Start” input, executing either the default program or the program number specified on the program select inputs. The program number is required for the SRUN command to execute a program from the system program directory. If the value of the single-step flag (SS) is TRUE, the program will execute in the single-step mode. (See PSELEN, PGM, PGMSEL, SS.)

**STEP**

Clears the “Ready for Single Step” flag in status information and single-steps the program running if executing in single-step mode. (See S, DBG, SS.)

**X**

Stops program and move execution.

---

## Non-Motion Execution Commands

**(Null)**

The null command consists of an empty command (only the END field is present). This command is simply acknowledged-no other action is taken.

**CLRPEAKS**

Clears peak current (PICMD), motor velocity (PVEL1), and following error (PFE) when set.

**DH****DP *position*****DPn *position***

These commands are used to set the present commanded position to zero (DH), to the specified position (DP) or to redefine the specified encoder position (DPn). The present commanded position is then defined as the home position simulating the completion of a home cycle. The Home Sequence Complete output will be turned ON if assigned. The ULTRA Plus or IQ must not be executing a move and the drive must be enabled before these commands will be accepted. Note that “DH” has the same effect as “DP 0”.

**ENABLE or EN****DISABLE or DIS**

These commands cause a software enable or disable of the ULTRA Plus or IQ, simulating the action of the Enable input. (See SEN)

**HRESET**

Causes a hardware reset when set. This command causes the controller to reboot the processor.

**RESET or RES**

Software reset.

## Runtime Status and Version Commands

### S

Returns a word (16 bits) bit map of present status information.

Bit	Description (1=True)
0	Program Active
1	In Home Position
2	Home Completed
3	In Position
4	Drive Ready (See RRLY)
5	Drive Enabled (See ERLY)
6	Forward Travel Limit
7	Reverse Travel Limit
8	Paused (See PAUSE)
9	In Peak Current (See PICMD)
10	I/O PLC Logic Disabled (See XPLC)
11	Error Active (See ENUM)
12	Warning Active (See WNUM)
13	New Debug Information Available (See DBG)
14	Ready for Single Step (See STEP, SS)
15	Executing Tune Cycle

### SI

Returns a byte (8 bits) bit map of present interrupt and home switch status.

Bit	Description (1=True)
0	INT1 Interrupt received (See FI1)
1	INT2 Interrupt received (See FI2)
2	Encoder 1 Index Interrupt received (See FIDX1)
3	Encoder 2 Index Interrupt received (See FIDX2)
4	(reserved)
5	(reserved)
6	(reserved)
7	Home Switch Active

HOST MODE

### DBG

Returns information on the next program line to execute in the following format:

program#, line#, "text"

The text field may be empty if no source program is available for the program (although the quotes will

still be present). A program must have been compiled with the debug option selected before debug information will be available.

### **VER [option]**

Returns ULTRA Plus or IQ version information for the specified option in the following formats:

<b>Option</b>	<b>Description</b>	<b>Return Value</b>
0	ULTRA Plus or IQ Software Revision	vv.rr
1	NVRAM Personality	“personality info. string”
2	Text ROM Id	“text id string”
3	Option Card Revision	vv.rr

If the *option* parameter is not present it defaults to zero. If no option card is present, VER 3 will return 0.00.



# Variables and Flags Assignment and Request Commands

---

## Introduction

All variable assignment commands have an associated request command that can be used to query the current value of the variable. The value returned by the request command is in the same format as the data required in the assignment. When nonvolatile memory parameters are changed, the ULTRA Plus or IQ will automatically update internal checksum data if appropriate.

### Personality Module Variables and Flags

Many variables have both a nonvolatile value stored in the Personality Module and an internal runtime value. When the value in the Personality Module is changed, the runtime value is updated automatically. When a program changes a value (for example, KP) it is actually changing only the internal runtime value. Most runtime values are returned to the value stored in the Personality Module upon completion of the program.

If a value is stored in the Personality Module, the command will set or read the value in the Personality Module. To set or read the runtime value, precede the variable name with a “#”.

**Example Commands**

Serial Command	Resulting Action
ACC = 200	Set the Personality Module default acceleration to 200 user units per Timebase.
#ACC = 300	Set the current runtime value of acceleration to 300 user units per Timebase.
ACC	Read the value for acceleration stored in the Personality Module in user units per Timebase.
#ACCEL	Read the current runtime value for acceleration in user units per Timebase.

**Note:** The runtime value of most variables is stored in internal units. When the runtime value is accessed the conversion to and from internal units is transparent. Small rounding errors may occur during conversion of runtime parameters, so that the value returned may not exactly match the value set.

**Table Format**

The following tables divide variables and flags into functional groups.

The Flag/Variable column contains an “F” if the command sets or reads a flag, a “V” if it contains a variable with a standard size, or the size of the variable if it is something other than the standard size. Refer to Part 4 • Programming, User Variables, and Arithmetic for more detailed information on the standard size of variables.

The Read/Write column identifies which variables can be read (R) and which can be written (W) or set.

The Personality Module (PM) column identifies variables stored in the Personality Module, making them nonvolatile. A “#” identifies variables stored in the Personality Module that have a runtime value as well. Values marked with a “#” are returned to the value in the Personality Module at program completion.

## Acceleration and Velocity

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
ACCEL or ACC	Acceleration used to calculate moves in user units per second <sup>2</sup>	V	R/W	Y	#
FVEL1	Filtered VEL1 in user units per Timebase	V	R	N	
JACCEL	Jog acceleration in user units per second <sup>2</sup>	V	R/W	Y	#
JDECEL	Jog deceleration in user units per second <sup>2</sup>	V	R/W	Y	#
JVEL or JV	Jog velocity in user units per Timebase	V	R/W	Y	#
PVEL1	Peak velocity in user units per Timebase	V	R	N	
VCMD	Velocity command in user units per Timebase	V	R	N	
VEL or V	Default system velocity for motion profiles in user units per Timebase	V	R/W	Y	#
VEL1	Velocity feedback in user units per Timebase	V	R	N	
VEL2	Encoder 2 velocity in user units per Timebase	V	R	N	
VFILT	FVEL1 Velocity filter	V	R/W	Y	#
VSCALE	FVEL1 velocity scale velocity monitor output scale	V	R/W	Y	#

## Current

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
IAVE	Average current command in amperes	V	R	N	
IAVG	Average current fault trip point	V	R/W	Y	#
ICMD	Current command in amperes	V	R	N	
ILIMIT	Peak current limit	V	R/W	Y	#
MXIAVE	Maximum IAVE	V	R	Y	
MXILIMIT	Maximum ILIMIT	V	R	Y	
PICMD	Peak current command in amperes	V	R	N	

## Gains

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
				Y	#
FGAIN	Acceleration feedforward gain	V	R/W	Y	#
FILTEN	TRUE to enable the velocity loop filter	F	R/W	Y	#
FILTER or FIL	Velocity loop filter value	V	R/W	Y	#
IGAIN	Integral gain for the velocity loop	V	R/W	Y	#
IZONE	Region where KI has effect	V	R/W	Y	#
KFF	Velocity feedforward gain	V	R/W	Y	#
KI	Integral gain for the position loop	V	R/W	Y	#
KP	Proportional gain for the position loop	V	R/W	Y	#
KPZ	Proportional gain used within the PZONE	V	R/W	Y	#
MXFGAIN	Maximum FGAIN	V	R	Y	
MXFILTER	Maximum FILTER	V	R	Y	
MXIGAIN	Maximum IGAIN	V	R	Y	
MXKFF	Maximum KFF	V	R	Y	
MXKI	Maximum KI	V	R	Y	
MXKP	Maximum KP	V	R	Y	
MXPGAIN	Maximum PGAIN	V	R	Y	
PGAIN	Proportional gain for the velocity loop	V	R/W	Y	#
PZONE	Region where KPZ is used instead of KP	V	R/W	Y	#

## Homing

These variables and flags are used in conjunction with the standard Home Program. Refer to Appendix A, System Programs, for more detailed information about the home program and the uses of these variables.

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
DHINP	If the Define Home input is assigned, the single bit set in the word corresponds to the input assigned to the function. A value of zero indicates that the function is not assigned. Care must be taken to avoid assigning multiple functions to the same input.	word	R/W	Y	
HACTIVE	TRUE for active closed home switch.	F	R/W	Y	
HOFFS	Home offset.	V	R/W	Y	#
HRINP	If the HRESET input is assigned, the single bit set in the word corresponds to the input assigned to the function. A value of zero indicates that the function is not assigned. Care must be taken to avoid assigning multiple functions to the same input.	word	R/W	Y	
HSEQEN	TRUE to enable Home Sequence Complete output.	F	R/W	Y	
HVEL or HV	Home velocity.	V	R/W	Y	#
INDEX	TRUE for encoder index positioning during home.	F	R/W	Y	

## Inputs – Analog and Digital

The commands in this table cover all digital and analog inputs including inputs located on optional I/O Expansion Cards.

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
ADC1	Analog input 1 in volts (option card required).	V	R	N	
ADC2	Analog input 2 in volts (option card required).	V	R	N	
ADC3	Analog input 3 in volts (option card required).	V	R	N	
ADC4	Analog input 4 in volts (option card required).	V	R	N	
ADC5	Analog input 5 in volts (option card required).	V	R	N	
AFEED	TRUE to enable ADC1 for analog feedrate.	F	R/W	Y	
DBNCE	Debounce time (mS).	byte	R/W	Y	
I1...I48	Digital inputs status.	F	R	N	
IB1 IB2 IB3	These read-only variables access the status of the digital inputs as an entire block. Bits 0..15 correspond to inputs 1..16. If an I/O expansion card is present, the IB2 and IB3 commands access inputs 17..32 and 33..48 respectively.	word	R	N	
PACTIVE	TRUE for active closed PAUSE switch.	F	R/W	Y	
PGMSEL	Number of program select lines. Input 16 is always the MSB.	byte	R/W	Y	
PSELECT	TRUE to enable running from program switch.	F	R/W	Y	
SWINP	This word variable represents a bit map indicating whether or not various inputs are assigned. Bits 0..15 correspond to inputs 1..16 and reflect the assignment (1) or non-assignment (0) of fixed inputs. Unused bits that are not associated with fixed inputs should be cleared.  <b>BIT      Description</b> 0      Forward Travel Limit 1      Reverse Travel Limit 2      Enable 3      Start Program 4      Home Switch 5      Start Home 6      Jog Reverse 7      Jog Forward 8      Pause 9      Emergency Return to Position 10..15 (Reserved)	word	R/W	Y	
XINP	If the STOP input is assigned, the single bit set in the word corresponds to the input assigned to the function. A value of zero indicates that the function is not assigned. Care must be taken to avoid assigning multiple functions to the same input.	word	R/W	Y	

## Interrupts

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
FI1	TRUE if INT1 interrupt received. If TRUE, the I1P1, I1P2 positions are available. It is your responsibility to clear these flags. (See SI, INT1, INT2, I1P1, I1P2, I2P1, I2P2.)	F	R/W	N	
FI2	TRUE if INT2 interrupt received. If TRUE, the I2P1, I2P2 positions are available. It is your responsibility to clear these flags. (See SI, INT1, INT2, I1P1, I1P2, I2P1, I2P2.)	F	R/W	N	
FIDX1	TRUE if Encoder 1 Index interrupt received. If TRUE, the IX1P1, IX1P2 positions are available. It is your responsibility to clear these flags. (See SI, IDX1, IDX2, IX1P1, IX1P2, IX2P1, IX2P2.)	F	R/W	N	
FIDX2	TRUE if Encoder 2 Index interrupt received. If TRUE, the IX2P1, IX2P2 positions are available. It is your responsibility to clear these flags. (See SI, IDX1, IDX2, IX1P1, IX1P2, IX2P1, IX2P2.)	F	R/W	N	
I1P1	Position 1 latched by interrupt 1 in user units.	V	R	N	
I1P2	Position 2 latched by interrupt 1 in user units.	V	R	N	
I2P1	Position 1 latched by interrupt 2 in user units.	V	R	N	
I2P2	Position 2 latched by interrupt 2 in user units.	V	R	N	
IDX1	TRUE if Encoder 1 Index interrupt is enabled. Internally set FALSE upon receiving the corresponding index interrupt. (See FIDX1.)	F	R/W	N	
IDX2	TRUE if Encoder 2 Index interrupt is enabled. Internally set FALSE upon receiving the corresponding index interrupt. (See FIDX2.)	F	R/W	N	
INT1 INT2	The value of these variables indicate the enabled status of interrupt inputs one and two. The value will be 0 if the interrupt is disabled, 1 if it is enabled in a one-shot mode, and 2 if enabled in a continuous mode. When enabled in a one-shot mode, the variable will be zeroed (interrupt disabled) upon receiving an interrupt. (See SI, FI1, FI2, I1P1, I1P2, I2P1, I2P2.)	byte	R/W	N	
IX1P1	Position 1 latched by Encoder 1 index interrupt.	V	R	N	
IX1P2	Position 2 latched by Encoder 1 index interrupt.	V	R	N	
IX2P1	Position 1 latched by Encoder 2 index interrupt.	V	R	N	
IX2P2	Position 2 latched by Encoder 2 index interrupt.	V	R	N	
LP2EN	TRUE for latch position (LPOS) from encoder 2. <b>Note:</b> Power must be cycled or a Hard reset performed before latched position selection will take effect.	F	R/W	Y	
LPOS	Hardware latched position in user units.	V	R	N	

## Motion

The commands contained in this table are used to control the effect of motion. They do not cause any motion, they set conditions that affect motion. See also “Motion Execution Commands” on page 295.

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
GEAR	Current value of the gear ratio.	V	R/W	N	
GEAREN	TRUE if the GEAR is enabled.	F	R/W	N	
ROTT	Rotation direction for tracking function.	F	R/W	N	
SLEW	Limit for output of gear.	V	R/W	Y	#
SLEWEN	Enables the slew rate limit for the gear output.	F	R/W	Y	#
TRACKINGEXT- POSSELECT	TRUE if the Tracking function applies after Gear and Slew functions. FALSE if the Tracking function applies before Gear and Slew functions.	F	R/W	N	
TRACKINGMODE	Sets the mode for the Tracking function: 0 = Off 1 = Ratchet mode 2 = Anti-backlash mode	V	R/W	N	
TRACKING- SYNCED	TRUE if the Tracking function is not disabling follower motion.	F	R	N	

## Outputs – Analog and Digital

The commands in this table cover all digital and analog outputs including the monitor output and outputs located on optional I/O Expansion Cards.

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
ATHOME	TRUE to enable At Home output.	F	R/W	Y	
DAC1	Default DAC1 output value. (Note that the runtime value of DAC1 is write only.)	V	R/W	Y	#
DISERR	TRUE for ERROR output ON when disabled.	F	R/W	Y	
ERLY	TRUE if the DRIVE ENABLED relay is closed. The status of these relays are normally handled internally and these commands are used only in diagnostics modes. The PLC logic of the drive should be disabled (XPLC TRUE) before directly accessing either of these relays. <b>Note:</b> The relay status is available in the S (status) command.	F	R	N	
ERREN	TRUE to enable ERROR output.	F	R/W	Y	
IPEN	TRUE to enable In-Position output.	F	R/W	Y	

HOST MODE



IPREL	TRUE for relative In-Position output. FALSE for absolute In-Position output.	F	R/W	Y	
MDAC	Monitor DAC value. This parameter is write-only and sets the MON analog output value. This output is normally updated internally but may be set directly if no MON variable is being converted. (See MONOUT.)	V	W	N	
MONOUT	Select the monitor variable for MON DAC output.	V	R/W	Y	
MVO	TRUE if internal monitor velocity output (MVO) is enabled. This 8-bit analog PWM output, if enabled, always monitors feedback velocity. The output does not go off-board but exists as a diagnostics tool. This flag is FALSE on power-up.	F	R/W	N	
O1...O24	Digital outputs status (1=ON, 0=OFF).	F	R/W	N	
OB1 OB2 OB3	These read-only variables access the status of the digital outputs as an entire block. Bits 0..7 correspond to outputs 1..8. If an I/O expansion card is present, the OB2 and OB3 commands access outputs 8..16 and 17..24 respectively.	byte	R	N	
OCHG1 OCHG2 OCHG3 OSET1 OSET2 OSET3	These parameters work together to define the state of the outputs when a program ends. Bits 0..7 of OCHG1 or OSET1 correspond to outputs 1..8. If additional outputs are available, OCHG2 or OSET2 and OCHG3 or OSET3 correspond to outputs 9..16 and 17..24 respectively. The “OCHG” bits indicate which outputs should be updated (1) or left unchanged (0). If a value is to be updated, the “OSET” bits indicate whether the corresponding output is to be turned ON (1) or OFF (0).	byte	R/W	Y	
RAMPD	TRUE to continually ramp DAC1 between $\pm 10$ volts.	F	R/W	N	
RAMPM	TRUE to continually ramp MON between $\pm 10$ volts. These commands are normally used only by diagnostics and require that the DAC1 or MON first be disabled.	F	R/W	N	
RRLY	TRUE if the READY relay is enabled.	F	R/W	N	
RUNEN	TRUE to enable Program Running output.	F	R/W	Y	

## Position

This table contains commands for reading and setting most variables that relate to distance, position, position error, or commanded position. Position variables associated with interrupts are contained in the table dedicated to interrupts. See “Interrupts” on page 305. s.

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
				Y	#
EPOS	EReturn position.	V	R/W	Y	#
FE	Following error in user units.	V	R	N	
GPOS	External position command (from gear) in user units, before slew limit.	V	R	N	
JOGACTIVE	TRUE if jog motion is occurring (PJOB is changing).	F	R	N	
MOVECOMPLETE	TRUE if profile generated motion (MD, MP, or DV moves) is occurring (PGEN is changing).	F	R	N	
PCAM	Position command from Cam function.	V	R	N	
PCMD or P	Position command in user units.	V	R	N	
PEXT	External position command (from gear) in user units, after slew limit.	V	R	N	
PFE	Peak following error in user units.	V	R	N	
PGEN	Position command from internal profile generator.	V	R	N	
PJOB	Jog command from Jog input(s).	V	R	N	
POS1	Encoder 1 position in user units.	V	R	N	
POS2	Encoder 2 position in user units (scaled by Scale2 parameter).	V	R	N	
POS3	Position 3 (from option card).	V	R	N	
POSN or F	Position feedback in user units (scaled by Scale parameter).	V	R	N	

## Serial

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
ADDR	Returns the host axis identifier specified by the dip switch. The value will be in the range of 0..63. (Zero indicates no individual axis identifier.)	byte	R	N	
HOST	TRUE if the programming terminal is in host mode. In host mode, the ULTRA Plus or IQ will give less verbose responses to some commands and will not supply error message strings.	F	R/W	N	
NOECHO	TRUE if the ULTRA Plus or IQ is not to echo incoming characters in a full duplex mode. The default state on power up is FALSE.	F	R/W	N	
NOLF	TRUE if the ULTRA Plus or IQ is not to add a line-feed (^J) to responses. If TRUE, only a carriage return (^M) will be transmitted.	F	R/W	N	

### COM1 word

### COM2 word

These commands set the serial parameters for the Operator Terminal (Com1) and the Programming Terminal (Com2). The parameters are encoded as a bit map in the following format:

Bit	Description	Values
0, 1	Unused	
2		0 = RS-232C 1 = RS-422
3	Unused	
7, 6, 5, 4	Baud Rate	0000 = 1200 0001 = 2400 0010 = 4800 0011 = 9600 (default) 0100 = 19200
9, 8	Parity	00 = None 01 = Odd 10 = Even
10	Data Bits	0 = 7 Data Bits 1 = 8 Data Bits
11..15	Unused	

HOST MODE

## System

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
ABSMODE	TRUE to enable Absolute mode	F	R/W	Y	

ACTIVESN	Indicates the number of scanned events active. <u>Bit</u> <u>Description</u> 0          No scanned events executing 1 to 8      Number of scanned events active	V	R	N	
ACTIVEXN	Indicates the number of Xkey routines active. <u>Bit</u> <u>Description</u> 0          No Xkey routines executing 1 to 4      Number of Xkey routines active	V	R	N	
AUXID	Returns a byte value indicating the option board type attached to the ULTRA Plus or IQ. <u>BIT</u> <u>Description</u> 0      No option board present 1      I/O and Memory expansion board 2      I/O expansion only 3      Reserved for future expanded capabilities 4      Reserved for future expanded capabilities	byte	R	N	
CF	Access the current run-time feedback configuration. Configuration is automatically set to 1 on power-up. The value must be in the range 1...4. (See CF1, CF2, CF3, CF4.)	byte	R/W	N	
DM	Drive Peak Current. For an ULTRA Plus or IQ-Series Positioning Drive Module, this value is the peak current rating in Amps. For an IQ-550, this value represents the drive peak current in Amps for a 10 V ICMD output. This value is set up using the IQPMGEN program.	V	R	Y	
ENUM WNUM	Read-only variables which reflect the value latched upon receipt of the last status (S) command. If the status command indicated that a error or warning was active, these commands may be used to get the actual error number. The value returned will be in the range 0..74. (See S, FTXT in the appendix "Error Messages" on page 418)	byte			
ESIZE	Number of Encoder Lines (counts per rev divided by 4). For an IQ-550, this value is set up using the IQPMGEN program.	V	R	Y	
FDR	Default Feedrate value.	V	R/W	Y	#
FEL	Following Error Limit.	V	R/W	Y	#
FET	Following Error Time.	V	R/W	Y	#
FLIMIT	Forward travel limit.	V	R/W	Y	#
FLTDIS	TRUE to disable some faults.	F	R/W	Y	
FTXT	This query command will return a quoted text string for the error or warning number specified. The number must be in the range 0..74. (See ENUM, WNUM.)	string	R	N/A	
GTERM	TRUE if generic operator terminal is selected.	F	R/W	Y	

HISTORY or HIS [n]	Returns a quoted string in the format “HHHH- HHH:MM message” giving ULTRA Plus or IQ EEPROM fault information for the last “nth” fault number in reverse chronological order. If no fault # is given it defaults to zero (the latest fault). The time field is in the same format used for HTIME (see below) and the message specifies the fault information. (See HTIME, FTXT.)	string	R	Y	
HTIME	Returns the current ULTRA Plus or IQ power-on time in the format “HHHHHHH:MM”. The “H” field specifies the hours and the “M” field speci- fies the minutes (to the nearest 10 minutes) that the ULTRA Plus or IQ has been powered up.	string	R	Y	
JM	Motor inertia. For an IQ-550, this value is set up using the IQPMGEN program.	V	R	Y	
JOGF	Software Jog Forward command.	F	R/W	N	
JOGR	Software Jog Reverse command.	F	R/W	N	
KT	Motor torque constant in inch-pounds per Amp. For an IQ-550, this value is set up using the IQP- MGEN program.	V	R	Y	
MXOVRSPD	Maximum OVERSPEED.	V	R	Y	
NPGMS	Returns the number of programs available in the user program directory. User programs are num- bered 0..(NPGMS-1).	byte	R	N	
NVPGS	Returns a byte bit map of information on the number of extra NVRAM pages available in the ULTRA Plus or IQ encoded as follows:  <u>Bit</u> <u>Description</u> 0          NVRAM page2 present 1          NVRAM page3 present 2          NVRAM page3 present  An option board must be present before extra NVRAM pages are available.	byte	R	N	
OVERSPEED	OVERSPEED fault trip point.	V	R/W	Y	#
PAUSE	TRUE if software PAUSE.	F	R/W	N	
PCMACTIVE	TRUE if PCAM is changing.	F	R	N	
PEXTACTIVE	TRUE if PEXT is changing.	F	R	N	
PGM	Default program number.	byte	R/W	Y	
PSELEN	TRUE to enable choosing program number to run from program select inputs. FALSE enables running default program number.	F	R/W	Y	

PURGEMOTION	<p>Purge motion queue in the following manner:</p> <table border="0"> <thead> <tr> <th><u>Mode bit</u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Normal Pgen</td> </tr> <tr> <td>1</td> <td>Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed.</td> </tr> <tr> <td>2</td> <td>Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value.</td> </tr> <tr> <td>3</td> <td>Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed. Normal Pgen motion after purge.</td> </tr> <tr> <td>4</td> <td>Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value. Normal Pgen motion after purge.</td> </tr> </tbody> </table> <p>All other values result in Mode 0. Subsequent moves are ignored until a zero value is assigned.</p>	<u>Mode bit</u>	<u>Description</u>	0	Normal Pgen	1	Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed.	2	Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value.	3	Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed. Normal Pgen motion after purge.	4	Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value. Normal Pgen motion after purge.	V	R/W	N	
<u>Mode bit</u>	<u>Description</u>																
0	Normal Pgen																
1	Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed.																
2	Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value.																
3	Purge Pgen motion queue immediately. No deceleration of Pgen to zero velocity performed. Normal Pgen motion after purge.																
4	Purge Pgen motion from stick queue. Decelerate Pgen to zero velocity. Decel rate is the system Accel value. Normal Pgen motion after purge.																
PURGEMOTION-ACTIVE	TRUE if Motion Purging is complete.	F	R	N													
RFDR	Current run time feedrate value in percent.	V	R	N													
RLIMIT	Reverse travel limit.	V	R/W	Y	#												
ROT	TRUE if counter-clockwise rotation is positive direction for encoder 1.	F	R/W	Y													
ROT2	TRUE if counter-clockwise rotation is positive direction for encoder 2.	F	R/W	Y													
SCALE	Encoder counts per 1 user unit of distance.	V	R/W	Y	#												
SCALE2	Encoder 2 counts per 1 user unit of distance.	V	R	Y	#												
SEN	TRUE if software enable. (see ENABLE, DIS-ABLE.)	F	R/W	N													
SS	TRUE if program execution in single step mode. (see STEP.)	F	R/W	N													
TBASE	FALSE (0) for velocity in user units per second, TRUE (1) for velocity in user units per minute.	F	R/W	Y	#												
WIN	In-Position window.	V	R/W	Y	#												
WINT	In-Position window time.	V	R/W	Y	#												
XPLC	TRUE to stop the ULTRA Plus or IQ from executing its internal I/O logic. Input status will still be available but no action will be taken on changing inputs and outputs will be updated only from host commands. This command is normally used only by diagnostics.	F	R/W	N													

**CF1 word****CF2 word****CF3 word****CF4 word**

These commands set the four feedback configurations available to the CF command. The configurations are bit map encoded in the following format:

Bit	Description	Values
0, 1	Unused	
2	Encoder 2	0 = In 1 = Out
3	Unused	
6, 5, 4	Feedback	001 = Position 1 010 = Position 2 100 = Position 3
8, 7	Gear Input	01 = Position 2 10 = Position 3 (Option Card required)
9..11	Unused	
12		0 = Pos2 from Encoder 1 = Pos2 from Step and Dir Inputs
13..15	Unused	

These commands set only the NVRAM parameters. The CF command must be used to select one of the four configurations as the current run-time configuration. (See CF)

## Tune Mode

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
TDIR	Tune direction: 0 = bi-directional 1 = positive 2 = negative	V	R/W	N	
TMXDIST	Maximum distance in Auto Tune.	V	R/W	N	
TMXVEL	Maximum velocity in Auto Tune.	V	R/W	N	
TPER	Time period for Manual Tune.	V	R/W	N	
TRATIO	Inertia ratio calculated by Auto Tune.	V	R	N	
TSCUR	Step current for Auto Tune.	V	R/W	N	
TSPOS	Position Step for Manual Tune.	V	R/W	N	
TSTART	Command to start tuning.	N/A	N/A	N	
TSTOP	Command to stop tuning.	N/A	N/A	N	
TSVEL	Step Velocity for Manual Tune.	V	R/W	N	
TTLIM	Total time period for Auto Tune.	V	R/W	N	

## User Variables and Flags

Command	Description	Flag/ Variable	Read/ Write	Personality Module	
B1...B8	Nonvolatile Flag.	F	R/W	Y	
F1...F64	Volatile flag, false on power-up.	F	R/W	N	
G1 ...G64	Nonvolatile numeric variables.	V	R/W	Y	
V1...V64	Volatile numeric variables, zero on power-up.	V	R/W	N	



# File Commands

---

## Program Maintenance and Directory Commands

The following commands work with program and directory information. Commands with an “S” prefix work on the system program directory.

### **DEL *program***

### **SDEL *program***

Deletes both the program executable and program source code. No error will be issued if the program is empty.

### **DELS *program***

### **SDELS *program***

Deletes only the source code, if present, for the specified program.

### **DIRI *program***

### **SDIRI *program***

Returns directory information for the program specified in the format:

“name”, SSIZE, ESIZE

Name will be an empty string (although the quotes will still be present) if no executable program code exists. ESIZE and SSIZE refer to the size, in bytes, of the executable and source files, respectively.

### **FREE [*page*]**

Returns the number of bytes of free program storage on the specified page of NVRAM. If the optional page is not given it defaults to zero.

**Note:** System and user programs share space on page 0 so no alternate “SFREE” command is needed.

### **DIR [*page*]**

### **SDIR**

Prints an entire directory. This command is normally only useful in a “terminal” mode. A host should use the more specific DIRI command.

## Transfer Commands

The following commands are used to transfer portions of the Personality Module to and from the Host.

### **SAVE or LOAD *MEM* [*page*]**

### **SAVE or LOAD *ALL* [*page*]**

### **SAVE or LOAD *PAR***

The Save command will transfer data from the ULTRA Plus or IQ to the host. The Load command will transfer from the host to the ULTRA Plus or IQ. *MEM* transfers the entire nonvolatile memory page. *PAR* transfers only the user nonvolatile parameters. *ALL* transfers all programs (user and system) on a page.

### **SAVE or LOAD *program***

### **SSAVE or SLOAD *program***

### **SAVEE or LOADE *program***

### **SSAVEE or SLOADE *program***

The Save command will transfer programs from the ULTRA Plus or IQ to the host. The Load command will transfer from the host to the ULTRA Plus or IQ. These commands load and save user or system programs and require a valid program number. SAVE or LOAD (SSAVE or SLOAD) transfer ASCII source files while SAVEE or LOADE (SSAVEE or SLOADE) transfer executable files.

For all Save commands, the ULTRA Plus or IQ will acknowledge the command and then wait for a NULL command (CR). This CR is not echoed regardless of the state of the NOECHO flag. When the NULL command is received, the ULTRA Plus or IQ will transfer the file to the host, followed by an ASCII EOF (end of file).

Load commands are acknowledged and the ULTRA Plus or IQ is then immediately ready to receive the specified file. Preceding blank lines (NULLs) will be ignored for all file types. The file will be echoed depending on the state of the NOECHO flag. The file may be terminated with either an ASCII EOF or ESC character. If an error occurs, the Error response will not be given until the EOF or ESC is received.

# Operator Terminal Commands

---

## Commands

The following commands are available to directly access the Operator Terminal for diagnostics or simulation purposes.

### **FNPGM *fn#***

This command may be used to start execution of an Fkey program without forcing the operator to press a key on the operator terminal.

The “*fn#*” must be in the range 1..24 and will only be meaningful if the specified program exists in the System Directory. A menu does not need to be present on the bottom line of the Operator Terminal to use this command.

### **OTADDR *address***

Sets the address when using the Operator Terminal Multidrop mode. Valid range is 0-9.

### **OTCLS**

Clears the Operator Terminal Screen. This action is taken only if the Operator Terminal is inactive and is normally used only by diagnostics.

### **OTCP**

Returns the current position of the cursor and clears the new cursor position flag of OTS. The cursor is only visible when the Operator Terminal is doing input. The position is returned in the format:

Row, Col (Note: *Row* and *Col* are zero based)

### **OTFILL *byte***

This command may be used to fill the Operator Terminal screen with an ASCII character. It is normally used only in diagnostics mode. The request will only be carried out if the Operator Terminal is inactive and the character byte is in the range 0x20..0x7f.

**OTFLUSH**

May be used to flush the characters to be presented to the OTGET command. Since the characters are available in a circular queue, the next character to be returned will be meaningless unless the queue has been continually monitored. After the OTFLUSH command is executed, no input will be available. This command should be issued before starting the monitoring of keystrokes.

**OTGET**

Returns an ASCII character indicating the last Operator Terminal key action. This command should only be used if the OTS status indicates that key input is available. (See Operator Terminal Key Codes.)

**OTLN1, OTLN2, OTLN3, OTLN4**

These commands will return a string of up to twenty characters for the Operator Terminal line requested and clear the appropriate bit flag in OTS. If less than twenty characters are returned, spaces may be assumed for the unspecified characters.

**OTM $n$** 

Operator Terminal Mode Command

Command	Syntax
OTM1	FN1, FN2, FN3, FN4, "Text"
OTM2	FN1, FN2, FN3, FN4, "Text"
OTM3	FN1, FN2, FN3, FN4, "Text"
OTM4	FN1, FN2, FN3, FN4, "Text"
OTM5	FN1, FN2, FN3, FN4, "Text"
OTM6	FN1, FN2, FN3, FN4, "Text"

These parameters set the mode menus for the Operator Terminal. Six menus may be defined and are cycled through by the Mode Key if any FN programs are assigned to the menu. (Unused menus will be skipped.)

*FN1..4* are the System Program numbers assigned to Fkeys 1..4 for the menu. The program numbers must be in the range 1..24. "Text" is the prompt string associated with the menu that appears on the bottom line of the Operator Terminal and must be no more than twenty characters (unfilled characters will default to an ASCII space). Example:

OTM2 5, 4, 7, 1, "Run Stop DefHm JogR"

Assigns *FN5* to Fkey 1, *FN4* to Fkey 2, *FN7* to Fkey 3, *FN1* to Fkey 4, and the associated string for Operator Terminal menu #2.

**OTMON**

This variable, if non-zero, sets or reads the monitor variable to display on the top line of the Operator Terminal. Refer to Part 5 • Language Reference, OTMON, for a listing of the variables that can be displayed.

**OUTPUT char**

Simulates the pressing of an Operator Terminal key. (See FNPGM, XNPGM, Operator Terminal Key Codes.)

**OTS word**

Returns a word bit map of Operator Terminal status information encoded as follows:

Bit	Description
0	New output on line 1 (See OTLN1..4)
1	New output on line 2

2	New output on line 3
3	New output on line 4
4	New cursor position (See OTCP)
5	Cursor is active
6	Unused
7	Operator Terminal is inactive (See OTX)
8	Key press input available (See OTGET, OTFLUSH)
9	Alarm LED is active
10	Terminal input buffer is full (See OTPUT)

**OTTBL key [,code]**

Sets key codes for generic operator terminal set up. Refer to Part 2 • IQ Master Environment, Parameter menu, Serial for more details.

**OTW Row, Col, “Text”**

Writes “*Text*” to the operator terminal at the specified *row* and *col* position. This action will only be taken if the Operator Terminal is inactive. (Note: *Row* and *Col* are zero based)

**OTX flag**

Inhibits the Operator Terminal from performing its normal screen updates and key monitoring. This command is normally used only in diagnostics mode. The OTS status should be checked after an inhibit request until the Operator Terminal returns inactive. After inhibiting, the Operator Terminal Screen will be cleared. The original screen will be replaced when the inhibit is removed.

**XNPGM Xkey#**

This commands may be used to start execution of an Xkey routine without forcing the operator to press a key on the operator terminal. The “*Xkey#*” must be in the range 1..4 and will only be meaningful if a program is running and the specified Xkey routine has been enabled.

## Operator Terminal Key Codes

The following codes may be used with the OTPUT command to simulate key input to the Operator Terminal. These codes are returned by the OTGET command.

Code	Description
M	Mode
S	Status
C	Clear
Y	Yes
N	No
B	Backspace
E	Enter
a	F1 pushed
b	F2 pushed
c	F3 pushed
d	F4 pushed
e	F1 released
f	F2 released
g	F3 released
h	F4 released
i	X1 pushed
j	X2 pushed
k	X3 pushed
l	X4 pushed
m	X1 released
n	X2 released
o	X3 released
p	X4 released

# Appendixes

This part of the manual includes supplementary information and programming examples.

The following appendixes are included:

- DDE Server
- System Programs
- Optional Operator Terminal Function Key Programs
- Optional Home Programs
- Application Examples
- What's New in Each Version
- Error Messages





# DDE Server

IQ Master supports DDE (Dynamic Data Exchange) using the DDEML (Dynamic Data Exchange Management Library). IQ Master is configured as a server only; another application (the client) can query IQ Master for information about variables that will be gathered over the serial link from the currently connected ULTRA Plus or IQ and sent to the client. In addition, the client can set variables to a specific value and also send executable commands to the ULTRA Plus or IQ.

The only transactions supported by IQ Master are the Request (request data), Poke (write data), and Execute Transactions. IQ Master does *not* support Advise Transactions.

To enable DDE conversations with IQ Master, a command-line argument is used when starting IQ Master. The command-line argument is either '/d' or '/D' (for example, *pathname\IQMASTER.EXE /d* where *pathname* is the drive and directory path of IQ Master).

The Service Name to establish a conversation with IQ Master is 'IQMASTER'. The Topic name is 'System'. Item names supported are shown below. The clipboard format for passing data is CF\_TEXT. All data passed will be in the form of strings using the DDEML DdeClientTransaction() function.

To establish a conversation with IQ Master and a client, IQ Master should be running before the client starts. If IQ Master is not running before starting the client, the client should try to run IQ Master using the WinExec() function with a '/d' or '/D' command-line argument as the optional parameter.

ULTRA Plus or IQ variables (known as Item names) that can be queried using DDE include:

Adc1	Adc2	Adc3	Adc4
Adc5	Addr	Fe	Fvel1
Gpos	l1p1	l1p2	l2p1
l2p2	lave	lcmd	lx1p1
lx1p2	lx2p1	lx2p2	Lpos
Pcmd	Pext	Pfe	Pgen
Picmd	Pjog	Pos1	Pos2
Pos3	Posn	Pvel1	Rfdr
Vcmd	Vel1	Vel2	

ULTRA Plus or IQ flags or variables (additional Item names) that can be queried or set using DDE:

- All G variables(G1-G64)
- All V variables(V1-V64)
- All B flags(B1-B8)
- All F flags(F1-F64)
- Gear
- Gearen

ULTRA Plus or IQ executable commands that can be executed using DDE (no data returned):

- Define HomeMovp
- Define PositionMovv
- DisableReset
- EnableRun # (# is program number to run)
- Hard ResetSelect # (# is address for multi-drop)
- HomeX
- Movd

ULTRA Plus or IQ executable commands that return data:

- EnumOb2
- Ib1Ob3
- Ib2Status
- Ib3Ver
- Ob1

### **DDE Initialization Failure**

A 'DDE Initialization Failure' message appears when IQ Master is started with the '/d' command-line argument but IQ Master could not register with the DDEML (Dynamic Data Exchange Management Library), IQ Master can still be used, but no DDE functions will be supported.

If this message appears, check for the presence of DDEML.DLL in your Windows\System directory. If the file is missing or damaged, run IQ MASTER Setup again.

# System Programs

---

## What are System Programs?

System programs are a group of ULTRA Plus or IQ programs which operate differently than normal motion programs. Some system programs automatically run upon power up. Some system programs are used with the function keys on the Operator Terminal. Other system programs have specific functions such as homing an axis, handling errors and warnings, or performing the Emergency Return function. Some system programs execute at the same time as (in parallel with) motion programs.

System programs are stored in a separate directory in the Personality Module called the system directory. System programs may be edited just like a motion program and saved to program locations in the system directory. Default system programs are included in standard Personality Modules and may be modified as needed by the application.

The system programs are:

- System Program 0-AUTO (Automatically runs on power up)
- System Programs 1-24-Operator Terminal function key Programs
- System Program 25-HOME
- System Program 26-Emergency Return
- System Program 27-Error Routine
- System Programs 28-31-Reserved for future development

---

## When are system programs run?

### On Power Up

The AUTO program (System Program 0) is run each time the ULTRA Plus or IQ is powered up or after an HRESET command is executed. The AUTO program runs even if the ULTRA Plus or IQ is disabled.

### Operator Terminal Function Keys

Operator Terminal function key programs (System Programs 1-24) are run using an Fkey on the Operator Terminal. Functions are assigned to the Fkeys in the IQ Master Parameter menu, Fkey Set Up. See Part 2 • IQ Master Environment, for a complete description.

### Assigned Inputs

The HOME program (System Program 25) is started by an off to on transition of the Home Command input (input 6), if input 6 is defined as the Home Command input in the Parameter menu. See Part 2 • IQ Master Environment, for a complete description.

The ERETURN program (System Program 26) is started by an off to on transition of the ERETURN input (input 10), if input 10 is defined as the ERETURN input in the Parameter menu. See Part 2 • IQ Master Environment, for a complete description.

### Error conditions

The EROUTINE program (System Program 27) is run each time a system error or warning occurs.

### Serial Host Command

Any system program may be run using the SRUN host command. Operator Terminal function key programs (System Programs 1-24) may also be run using the FNPGM host command. See Part 6, IQ Host Command Language, for a complete description.

### From another program

Operator Terminal function key programs (System Programs 1-24) may be run by using the FNPGM statement within another program. The EROUTINE program (System Program 27) may be run by causing a system error or warning with the SYSERROR or SYSWARN statements. The ERETURN program (System Program 26) may be run by setting the system flag ERET. The Execute Home program (System program 6) runs the HOME program by setting the system flag HOMECMD. See Part 5 • Language Reference, for a complete description.

---

## Modifying System Programs

Any system program may be edited, compiled, and saved like any other ULTRA Plus or IQ program.

The source text for system programs is listed below but is *not* included in standard Personality Modules (to save memory). The source text may be installed with IQ Master to hard disk in a subdirectory named Syspgms beneath the directory where IQ Master is installed (The source text of optional system programs described later in this appendix may be installed with IQ Master to hard disk in a subdirectory named Optsys beneath the Syspgms subdirectory). The system programs may then be edited by choosing Open from the File menu, and selecting the desired program.

When system programs are compiled, the Compiler Options in the Edit menu must be set according to the type of system program, or by using the PGMTYPE statement. The Add Debug Information check box should be cleared or the statement DEBUG = OFF should be used when compiling any system program. If compiling an AUTO program (system program 0), choose “Auto” for the Program Type. If compiling an Operator Terminal function key program (system programs 1-24), choose “Fkey Routine”. If compiling an EROUTINE program, choose “Error Program”. For other system programs including HOME (system program 25), and ERETURN (system program 26), choose “Main”.

## System Program 0 - Auto

The AUTO program is a unique system program that runs automatically upon power up or after an HRESET command is executed. All programming statements including motion commands may be used. The AUTO program may be used to print a message to the Operator Terminal screen, initialize variables, or may contain the entire application program. Caution should be used when using motion commands in the AUTO program since motion could occur immediately upon power up. The AUTO program will run even if the ULTRA Plus or IQ is disabled. If a motion command is reached in the AUTO program while the ULTRA Plus or IQ is disabled, the program will be “hung” until the ULTRA Plus or IQ is enabled by the Enable Input or Enable host command or until power is removed or an HRESET command is executed. The Auto program will continue once enabled.

The AUTO program is considered a motion program, so it cannot run in parallel with another motion program. Other system programs such as Operator Terminal function key programs may run in parallel with the AUTO program.

When a program is compiled to save to the ULTRA Plus or IQ as the AUTO program, the Compiler Options must be set for Program type - “Auto” and Add Debug Information - cleared (off). See Part 2 • IQ Master Environment, for a complete description of compiler options.

The example AUTO program listed below prints a message to the Operator Terminal screen.

### Program Listing

```
;Source File Name: AUTO.QPS
;Version: 3.20
;Tested with IQMaster Version: 3.20
;Tested with IQ firmware Version: 3.20
;----- Description -----
;This is an example 'Auto' program.
;
;This program prints "Allen-Bradley" to the operator terminal
;screen(serial port 1).
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;The message will be printed to the operator terminal screen.
;----- Begin Program -----
TITLE "Auto"
PGMTYPE = AUTOPGM
CLR                               ;Clear operator terminal screen
PRINT 1,7 "Allen-Bradley"        ;Print at row 1, column 5
END
```

Operator Terminal function key programs (System Programs 1-24) are run using a Function Key (F1-F4) on the Operator Terminal. Functions are assigned to the Fkeys in the IQ Master Parameter menu. See Part 2 • IQ Master Environment, for a complete description.

These programs may also be run from within a program using the FNPGM command. See Part 5 • Language Reference, for a complete description.

They may also be run using the SRUN and FNPGM host commands. See Part 6 • IQ Host Command Language, for a complete description.

Function key programs are unique because they run in parallel with a motion program. Flags and variables may be shared with the motion program to affect the flow of the motion program.

Motion statements (MOVD, MOVV, MOVV, D V, DIF, and DWL) and DELAY statements are NOT allowed in function key programs.

When a program is compiled to save to the ULTRA Plus or IQ as a function key program, the Compiler Options must be set for Program type - “Fkey Routine” and the Add Debug Information check box cleared (off). See Part 2 • IQ Master Environment, for a complete description of compiler options.

Standard Personality Modules contain default function key programs which are described below. These programs may be modified or replaced. System programs 9-12 and 21-24 are intentionally left blank and are available for any additional functions required by an application.

---

## System Program 1 - Jog Forward

This program activates jog motion in the forward direction. Motion will continue as long as the Fkey is pressed. When the Fkey is released, motion will stop. The Jog Velocity, Jog Acceleration, and Jog Deceleration parameters are used for jog motion. NOTE: Run this program only by pressing an Fkey. This program will not cause any motion if run using the FNPGM or SRUN commands.

### Program Listing

```
;Source File Name: JOGFWD.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;jog in the forward direction.
;This program is saved as System Program #1 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will initiate jog motion in the forward direction using the
;Jog Velocity, Jog Accel,and Jog Decel parameters.
;This program simulates activating the Jog Forward input.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;An operator terminal function key is used to run this program and initiate
;jog motion. Motion will stop when the key is released.
;----- Begin Program -----
```

```

TITLE "Jog Fwd"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
JOGF = ON                      ;Initiate jog motion
WAIT FNACTIVE = OFF           ;Wait for release of key
JOGF = OFF                     ;Stop jog motion
END

```

---

## System Program 2 - Jog Reverse

This program activates jog motion in the reverse direction. Motion will continue as long as the Fkey is pressed. When the Fkey is released, motion will stop. The Jog Velocity, Jog Acceleration, and Jog Deceleration parameters are used for jog motion. NOTE: Run this program only by pressing an Fkey. This program will not cause any motion if run using the FNPGM or SRUN commands.

### Program Listing

```

;Source File Name: JOGREV.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;jog in the reverse direction.
;This program is saved as System Program #2 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will initiate jog motion in the reverse direction using the
;Jog Velocity, Jog Accel,and Jog Decel parameters.
;This program simulates activating the Jog Reverse input.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;An operator terminal function key is used to run this program and initiate
;jog motion. Motion will stop when the key is released.
;----- Begin Program -----
TITLE "Jog Rev"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
JOGR = ON                      ;Initiate jog motion
WAIT FNACTIVE = OFF           ;Wait for release of key
JOGR = OFF                     ;Stop jog motion
END

```

## System Program 3 - Start

This program will run a motion program by simulating the Start input. Parameters are used to determine which program to run. If parameters are set to run a specific program, that program will run. If parameters are set to read a program number from the Program Select Inputs, those inputs will be read to determine which program to run.

### Program Listing

```
;Source File Name: START.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;run a program by simulating pressing the start input.
;This program is saved as System Program #3 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program simulates the 'Start' switch input. If the 'Default Run
;Program' parameter is set to a specific program number, that program is
;run. If parameters are set to read a program number from the program
;select inputs, the select inputs are used to determine which program to
;run.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;An operator terminal function key may be used to run this program.
;----- Begin Program -----
TITLE "Start"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
PGMNUM = 255 ;Setting a program number of 255 causes the IQ to
; check parameters to determine which program to run.
STARTP = ON ;Run the program.
END
```



## System Program 4 - Stop

This program stops motion and program execution by setting the system flag STOP. This duplicates the action of activating the X-Killmotion input or sending the X host command.

### Program Listing

```

;Source File Name: STOP.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to stop motion
;and program execution.
;This program is saved as System Program #4 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program stops motion and program execution. It duplicates the
;action of the X-Killmotion input and the 'X' host command.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;An operator terminal function key may be used to run this program.
;----- Begin Program -----
TITLE "Stop"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
STOP = ON                                ;Set the system flag STOP. This stops motion and
                                         ; program execution. This duplicates the action of
                                         ; the X-Killmotion input or the 'X' host command.

END

```

# System Program 5 - Run

This program runs a motion program by asking an operator to enter a program number to run. The Operator Terminal screen and keypad is used to prompt the operator.

## Program Listing

```
;Source File Name: RUN.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to run a
;specific program number.
;This program is saved as System Program #5 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program prompts an operator to enter a program number from the
;operator terminal keypad. After the operator verifies this action with
;a Yes or No, the selected program is run.
;
;----- Variables used -----
;F63      used for yes/no input from operator
;FNVAR1   special variable reserved for Function Key programs
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;An operator selects the program to run using the operator
;terminal keypad.
;----- Begin Program -----
TITLE "Run"
PGMTYPE = FKEYPGM          ;Set compiler options
DEBUG = OFF
CLR
PRINT 1,4 "Run Program"    ;Ask operator for program number.
FNVAR1 = PGMNUM            ;Use current program as default selection.
IF (PGMNUM = 255)          ;The current program # is changed to 0 if
    PGMNUM = 0             ; it is at 255. A 255 indicates that the
                           ; default run sequence has been used for
                           ; last program execution.
READ 3,1 "Program # ", SHOW FNVAR1 ,2,0 (0,31)
                           ;The program range is limited here to #31.
                           ;If a Memory and I/O option card is present,
                           ; this number should be increased to 63.
PRINT 3,1 "Run Pgm ",FNVAR1 ,2,0
READ " Y/N?",F63           ;Ask for verification before starting program.
                           ; (operator must press 'Y' <Enter>)
IF (F63 = ON)             ;If operator says 'Y',
{
    PGMNUM = FNVAR1       ; set program number,
    STARTP = ON           ; and start program.
}
CLR                        ;Clear F-Key screen on exit.
END
```

## System Program 6 - Execute Home

This program runs the Home program (system program 25). Note: No motion program can be running in order to run the Home program.

### Program Listing

```
;Source File Name: EXHOME.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;execute the home program.
;This program is saved as System Program #6 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;An operator terminal function key may be used to run this program.
;----- Begin Program -----
TITLE "Home"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
HOMECMD = ON ;Set system flag HOMECMD to run the home program.
END
```

## System Program 7 - Define Home

This program defines the present position as home (position zero). No motion programs may be running in order to define home.

### Program Listing

```
;Source File Name: DEFHOME.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;define home position (position zero).
;This program is saved as System Program #7 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will define the present motor position as home position
;(position zero). This program simulates activating the Define Home input,
;using the DH statement within a main program, or using the DH host command.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;An operator terminal function key may be used to run this program.
;----- Begin Program -----
TITLE "Def Home"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
DHCMD = ON ;Set system flag DHCMD to define home position.
END
```

## System Program 8 - Set Feedrate

This prompts an operator to enter a new Feedrate percentage. The Operator Terminal screen and keypad is used to READ the new value. The present value of Feedrate is displayed.

### Program Listing

```
;Source File Name: SETFDR.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to
;set feedrate FDR.
;This program is saved as System Program #8 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program prompts an operator to enter feedrate with the operator
;terminal keypad. The present value is displayed to allow pressing
;Enter to keep the present value.
;
;----- Variables used -----
;FNVAR1 special variable reserved for Function Key programs
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;An operator enters the feedrate using the operator terminal keypad.
;----- Begin Program -----
TITLE "Set FDR"
PGMTYPE = FKEYPGM ;Set compiler options
DEBUG = OFF
CLR ;Clear screen
PRINT "Feedrate: %" ;Print header
FNVAR1 = FDR ;Get present value
READ 1,10 ,SHOW FNVAR1 ,6,2 (0,200) ;Read in new value. Note that the
; present value is displayed. Values
; from 0 to 200 with 2 decimal places
; of accuracy are allowed.
CLR ;Clear screen
FDR = FNVAR1 ;Set new feedrate
END
```

## System Programs 9-12

These program locations may be used for other Operator Terminal function key programs as needed. Future standard Personality Modules may contain default programs in these locations.

### System Program 13 - Monitor VEL1

This program sets the system variable OTMON to the proper value which displays the system variable VEL1 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

#### Program Listing

```
;Source File Name: MONVEL1.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;motor velocity VEL1.
;This program is saved as System Program #13 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause VEL1 to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display VEL1.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;VEL1 will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon VEL1"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 12) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 12 ; otherwise activate monitor.
END
```

# System Program 14 - Monitor POSN

This program sets the system variable OTMON to the proper value which displays the system variable POSN on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

## Program Listing

```
;Source File Name: MONPOSN.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;feedback position POSN.
;This program is saved as System Program #14 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause POSN to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display POSN.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;POSN will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon POSN"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 2) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 2 ; otherwise activate monitor.
END
```

## System Program 15 - Monitor PCMD

This program sets the system variable OTMON to the proper value which displays the system variable PCMD on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONPCMD.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;position command PCMD.
;This program is saved as System Program #15 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause PCMD to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display PCMD.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;PCMD will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon PCMD"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 1) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 1 ; otherwise activate monitor.
END
```



## System Program 16 - Monitor FE

This program sets the system variable OTMON to the proper value which displays the system variable FE on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONFE.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;following error FE.
;This program is saved as System Program #16 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause FE to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display FE.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;FE will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon FE"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 3) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 3 ; otherwise activate monitor.
END
```

## System Program 17 - Monitor ICMD

This program sets the system variable OTMON to the proper value which displays the system variable ICMD on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONICMD.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;current command ICMD.
;This program is saved as System Program #17 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause ICMD to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display ICMD.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;ICMD will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon ICMD"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 14) ;If already on,
    OTMON = 0 ; turn it off
ELSE OTMON = 14 ; otherwise activate monitor.
END
```

## System Program 18 - Monitor IAVE

This program sets the system variable OTMON to the proper value which displays the system variable IAVE on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONIAVE.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;average current command IAVE.
;This program is saved as System Program #18 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause IAVE to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display IAVE.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;IAVE will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon IAVE"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 15) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 15 ; otherwise activate monitor.
END
```

## System Program 19 - Monitor RFDR

This program sets the system variable OTMON to the proper value which displays the system variable RFDR on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONRFDR.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor
;runtime feedrate RFDR.
;This program is saved as System Program #19 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause RFDR to be displayed on the top line of the
;operator terminal screen. This monitor display is continuously updated.
;Running this program will act as a toggle. If the monitor display is on,
;it will be turned off. If it is off or monitoring another variable, it
;will turn on and display RFDR.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;RFDR will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon RFDR"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 16) ;If already on,
    OTMON = 0 ; turn it off,
ELSE OTMON = 16 ; otherwise activate monitor.
END
```

---

## System Program 20 - Monitor ADC1

This program sets the system variable OTMON to the proper value which displays the system variable ADC1 (analog input 1 voltage) on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

### Program Listing

```
;Source File Name: MONADC1.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default operator terminal function key program to monitor the
;analog input ADC1.
;This program is saved as System Program #20 in standard personality modules.
;It may also be saved as any Function Key System Program (range is 1-24).
;The source is NOT saved in standard personality modules.
;
;This program will cause the ADC1 voltage to be displayed on the top line
;of the operator terminal screen. This monitor display is continuously
;updated. Running this program will act as a toggle. If the monitor display
;is on, it will be turned off. If it is off or monitoring another variable,
;it will turn on and display the ADC1 voltage.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;Operator Terminal - REQUIRED
;The ADC1 voltage will be displayed on the operator terminal screen.
;----- Begin Program -----
TITLE "Mon ADC1"
PGMTYPE = FKEYPGM;Set compiler options
DEBUG = OFF
IF (OTMON = 17)           ;If already on,
    OTMON = 0           ; turn it off,
ELSE OTMON = 17         ; otherwise activate monitor.
END
```

---

## System Programs 21-24

These program locations may be used for other Operator Terminal function key programs as needed. Future standard Personality Modules may contain default programs in these locations.

---

## System Program 25 - Home

To move to an absolute position a motion controller must first know where it is. But, when incremental encoders are used in the system, when the motion controller is first powered up, it does not know where the axis is. The motion controller must run a routine that will give it some absolute position reference. This is the function of a home program: to precisely locate a position on a machine. For example, if a motor is turning a ballscrew that is in turn moving the cutting tool of a lathe, the motion controller must know where the cutting tool is before it can make a move.

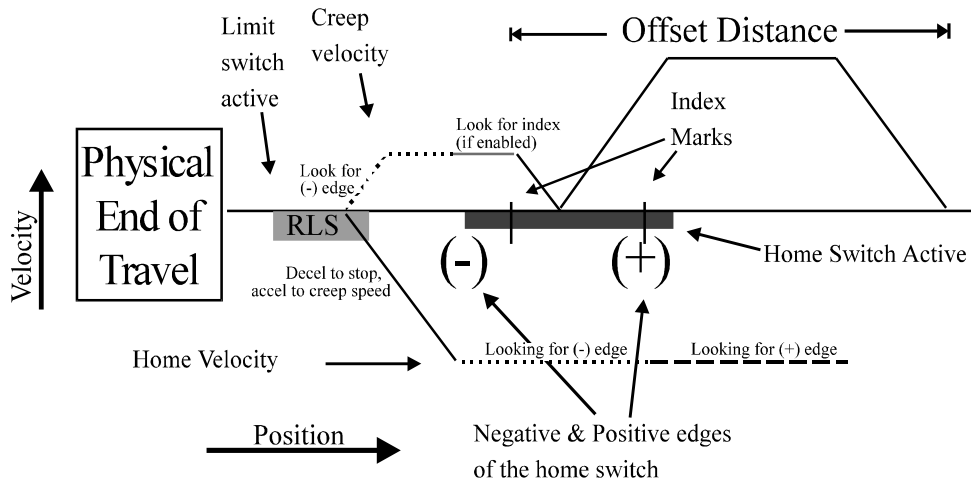
A home program needs to fit the application. In the ballscrew example just given, the normal procedure is to locate the edge of a Home Switch and then find the next Encoder Index Mark. (An Index Mark is a signal that comes from an encoder exactly once per revolution.) Since the switch edge locates one out of many possible index marks, and the index mark locates one pulse out of thousands of pulses/rev, this type of routine will precisely locate the tooling to within one pulse of the encoder. If the motor is not tightly coupled to the load (for example, a rubber wheel running on a slide) it would probably be better to just home the axis to the edge of a switch because the distance between the index mark and the edge of the switch will not always be the same and the final position would vary by one revolution.

After the ULTRA Plus or IQ loses main and auxiliary power, some sort of a home routine will need to be run if absolute moves are to be done. (Incremental moves are allowed before a home sequence is complete.) If the ULTRA Plus or IQ has been supplied with auxiliary power while the main power has been removed, another home sequence is not needed.

The ULTRA Plus or IQ has several built-in functions for running a home program: an input can tell the ULTRA Plus or IQ to run the home program, the home switch active sense (home switch active open or closed) can be selected, outputs can tell if the home program has been run and also if the axis is presently at home. The home program itself is completely user programmable.

Standard ULTRA Plus or IQ Personality Modules come with a “standard” home program that is documented below. There are additional home programs that have been written to support special cases to home an axis. The standard home program is a general purpose program that will handle most homing requirements; the other home programs are considerably smaller and are easier to understand, but do not have all of the features of the standard program. If memory space is a consideration, choosing an alternative home program that will do what you want should work well. If you need a custom home program, it may be better to start with an alternative home program that has the basic features you want and add to it. The finished custom home program will probably be smaller and easier to follow than if you started with the standard home program.

When a program is compiled to save to the ULTRA Plus or IQ as the HOME program, the Compiler Options must be set for Program type - “Main” and Add Debug Information - cleared (off). See Part 2 • IQ Master Environment, for a complete description of compiler options.

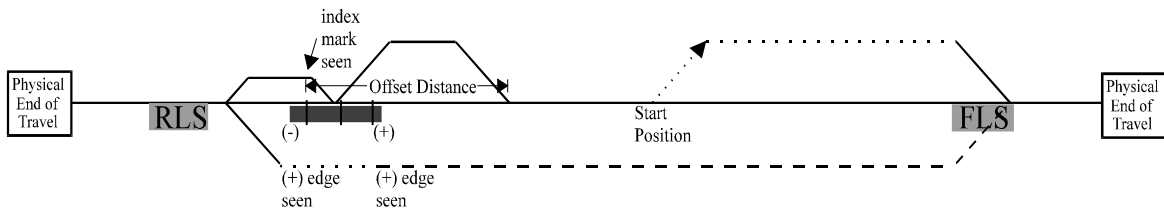


**Legend Used in Home Diagrams**

**Standard Home Program**

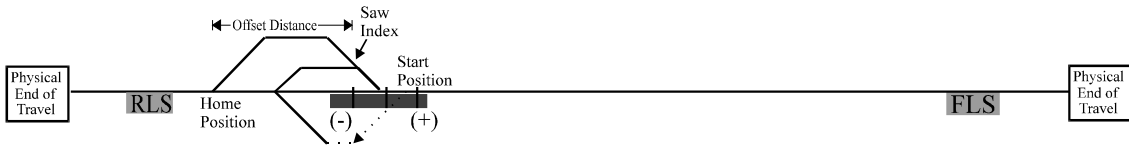
The standard home program has several configurations:

- home switch may be active in either sense  
**Parameter menu: Inputs**  
 Home Switch Active parameter: open or closed
- limit switches may or may not be used  
**Parameter menu: Inputs**  
 Limits parameter: on or off
- home velocity may be positive or negative  
**Parameter menu: Velocity/Accel...**  
 Home Velocity parameter
- home to switch then index  
**Parameter menu: Inputs**  
 Home Switch: enabled  
 Home to Encoder Index: enabled
- home to switch only  
**Parameter menu: Inputs**  
 Home Switch: enabled  
 Home to Encoder Index: disabled
- home to index only.  
**Parameter menu: Inputs**  
 Home Switch: disabled  
 Home to Index: enabled



**Normal Home Sequence Hitting Forward Limit Switch**

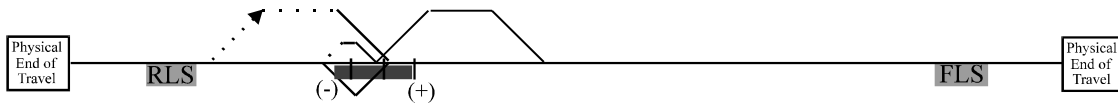
When the standard home program (`home_std.qps`) starts and the home switch is not active, as shown in the figure Normal Home Sequence Hitting Forward Limit Switch, the axis will start moving in the direction and speed set by the Home Velocity parameter. The axis will continue to move at this speed until a Home Switch or a limit switch is seen.



### Normal Home Sequence Starting with the Home Switch Active

If the home switch is active when the home program starts, the axis will move in the reverse direction as shown in the figure Normal Home Sequence Starting with the Home Switch Active. If limit switches are used and the axis is going forward and the forward limit switch is hit or, if going reverse and the reverse limit is hit, the axis will decelerate to a stop (as shown in the figures Normal Home Sequence Hitting Forward Limit Switch and Acceptable Deceleration Distance), change direction and continue.

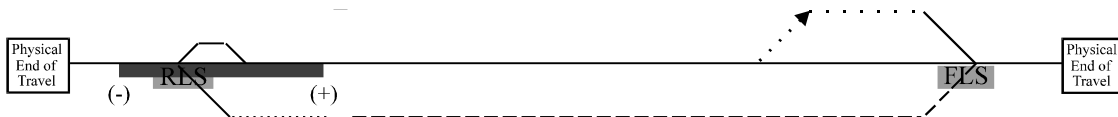
If the axis is moving backward when the (-) edge is seen, the axis will decelerate to a stop, accelerate to the creep speed in the forward direction looking for the (-) edge as shown in the figure Legend Used in Home Diagrams. The creep speed is 500 counts/second if Home to Encoder Index is not selected and 10,000 counts/second if the Home to Encoder Index is selected.



### Normal Home Sequence, Hitting Home Switch while Moving Forward

If the axis is moving forward when the (-) edge is seen, the axis will stop, back up rapidly to where the edge was seen, then back up at creep speed until it is off the switch and then creep forward onto the switch as shown in the figure Normal Home Sequence, Hitting Home Switch while Moving Forward.

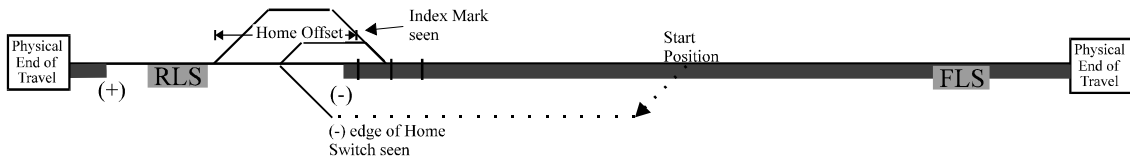
Once the (-) edge of the switch is found at creep speed, the axis will continue to move at the creep speed looking for the first index (if the Home to Encoder Index option was selected in the Inputs dialog box.) Once the index is seen, the index position will be used as the basis for the offset move as shown in the figure Legend Used in Home Diagrams,. The Home Offset is set in the Velocity/Accel dialog box. It is the distance between the index pulse or the (-) edge of the Home Switch (depending on the setup) and the position the axis will finally call home (position 0). The Home Offset can be either positive or negative as shown in the figures Normal Home Sequence Hitting Forward Limit Switch and Normal Home Sequence Starting with the Home Switch Active. If the Home to Encoder Index option is not selected, the edge of the switch will be used as the basis for the offset move.



### Error: (-) Edge of Home Switch not Between Limits

If both limit switches are hit before seeing a home switch (-) edge, an error will occur, as shown in the figure Error: (-) Edge of Home Switch not Between Limits. To avoid this problem, the Home Switch Active State needs to be changed in the Inputs dialog box. The result of changing from active open to active closed or vice-versa is shown in the figure Home Switch Active in the Opposite Sense.

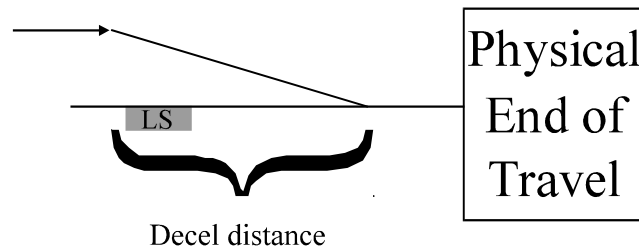




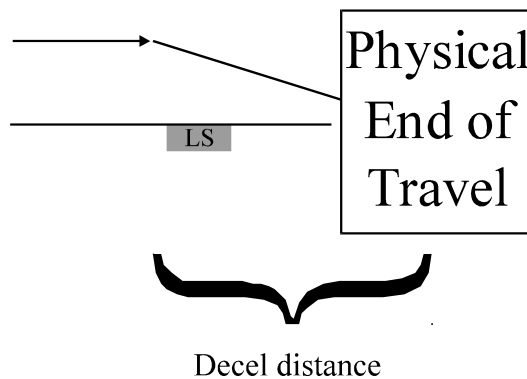
### Home Switch Active in the Opposite Sense

If only the Home to Encoder Index is selected, the axis will move at the creep speed in the direction set by the Home Velocity parameter until it sees an index. The index position will be the basis of the offset move.

The figures Acceptable Deceleration Distance and Deceleration Distance Resulting in an Axis Crash show the interaction of the Acceleration parameter and the physical placement of the forward and reverse limits. The standard home program uses the default system acceleration to accelerate and decelerate. The lower this value, the longer the deceleration distance. If the limit switches are placed too close to the physical end of travel, a crash could occur during the home cycle, as illustrated in Deceleration Distance Resulting in an Axis Crash. The deceleration distance (in user units) is the Home Velocity squared (units<sup>2</sup>/second<sup>2</sup>) divided by twice the acceleration parameter (units/second<sup>2</sup>):  $d = V^2/2a$ . If your Timebase is set to minutes, divide the Home Velocity by 60 before using this formula.



### Acceptable Deceleration Distance



### Deceleration Distance Resulting in an Axis Crash

## Program Listing

```

;Source File Name:  HOME_STD.QPS
;Version:  3.00
;Tested with IQMaster Version:  3.00
;Tested with IQ firmware Version:  3.00
;----- Description -----
;This is the default 'Home' program.
;This program is saved as System Program #25 in standard personality modules.
;The source is NOT saved in standard personality modules.
;
; This home program will home to a home switch, or an index or
; both. If limit switches are enabled and a limit switch is hit before the
; home switch is seen, the axis will change direction and continue looking
; for the home switch. Once the home switch is seen, the axis will slowly
; look for the (-) edge of the home switch (the edge of the switch that
; will be seen first when moving forward and coming on to the switch). If
; the index is not enabled, this edge position will be the reference position
; for the offset move. If the index is enabled the axis will continue to
; move into the switch and look for the next index. Once the index is seen,
; the index position will be reference position for the offset move. Hitting
; a limit switch during the offset move is not allowed and will cause a fault.
;
;v2.10 5/24/94  Changed S3 CONT and S5 CONT to ON commands so that no
;                overlapping of scanned events would happen.
;                Removed some unused labels.
;v3.00 2/13/96  Added compiler options statements
;
;----- V variables used -----
TITLE "Home_std"
PGMTYPE = MAINPGM          ;Set compiler options
DEBUG = OFF
ASSIGN origJacc  V1        ;holds original Jog accel
ASSIGN origJdec  V2        ;holds original Jog decel
ASSIGN creep     V3        ;Slow speed into index
ASSIGN HomeVel   V4        ;HomeVel (changed during program execution)
ASSIGN limcnt    v5        ;how many limit switches has it seen so far?
ASSIGN temp      V6        ;temp for calculations
ASSIGN state     V7        ;State variable for Home program
      ;initial          0        ;Initial state
      ;FL & RL Hit      1        ;both forward and reverse limits hit at same time
      ;FLHit            2        ;Forward limit hit state
      ;RLHit            3        ;Reverse limit hit state
      ;HS edge ON       4        ;Home switch edge on
      ;HS edge OFF      8        ;
      ;IOn              16       ;Index hit
;----- No G variables, F or B flags used -----
;
;----- Inputs -----
;Input 01          ;if limit switches are enabled, this is the reverse limit
;Input 02          ;if limit switches are enabled, this is the forward limit
;Input 05          ;if home switch is enabled, this is it
;Input 06          ;if home input is enabled,
                  ; this input can start the home program
;
;----- Outputs -----
;Output 06         ;if Home Sequence complete output is enabled, this
                  ; output will come on after the axis has successfully
                  ; homed
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED for reporting errors

```

```

;Expansion I/O - Not used
p;Expansion memory - Not used
;
;-----Scanned Events-----
S1: IF I1=OFF state=2          ;looks for reverse limit
S2: IF I2=OFF state=3          ;looks for forward limit
S3: IF HSWSTAT=ON state=4      ;look for Home Switch active
S4: IF HSWSTAT=OFF state=8     ;look for Home Switch not active
S5: IF I1=I2 state=1           ;used during move off limit switch

;-----Begin Program -----
;Variable Initialization

Start:
ALL OFF

limcnt = 0                      ;set number of limit switches seen so far to 0.
HomeVel = HVEL                  ;set HomeVel (which will change) to HVel to start
origJacc = JACCEL               ;save original Jog accel
origJdec = JDECEL               ;save original Jog decel
JACCEL = ACCEL                  ;set home accel to default accel
JDECEL = ACCEL                  ;set home decel to default accel

creep = TBASE / SCALE           ;Compute creep speed = 1 cnt / 2 ms.
IF INDEXEN=ON creep=creep*20   ;if doing index home, speed up creep

                                ;limit all speeds to HomeVel
IF HomeVel>0 IF creep>HomeVel creep=HomeVel
IF HomeVel<0 IF creep>-HomeVel creep= -HomeVel

                                ;If not already on the home switch or a limit switch,
                                ;assume that home switch is in direction of the home
                                ;velocity sign from the present position and move
                                ;until limit switch or home switch is hit.

FOUNDHOME = OFF
IF HSWEN=ON JMP init01          ;check for Home Switch not defined

;if here, index home only.
;Make sure in same direction as HomeVel
IF HomeVel < 0 creep = -creep
IF INDEXEN=ON JMP jonsw0       ;if should look for index, do it
JMP finish                     ;no index, no home switch,
                                ; call where we are "home"

                                ;if on switch, start in (-) direction
init01: IF HSWSTAT=ON IF HomeVel>0 HomeVel = -HomeVel
                                ;if FAC, start out negative dir
IF I1=OFF IF HomeVel>0 HomeVel = -HomeVel
                                ;if RAC, start out positive dir
IF I2=OFF IF HomeVel<0 HomeVel = -HomeVel

state=0
                                ;skip limit checks if not enabled
IF HLimits=OFF JMP init02
IF I1=ON S1 CONT ELSE limcnt = limcnt+1;look for FAC
IF I2=ON S2 CONT ELSE limcnt = limcnt+1;look for RAC
                                ;if both limit switches, error
IF limcnt=2 JMP err

init02: IF HomeVel>0 JMP fwd

```

```

DISLIM = ON          ;if here, start by moving backward
MOVV HomeVel
                    ;will be looking for (-) edge if Home switch already on
IF HSWSTAT=ON JMP C2

                    ;This is the backward move section
bkwr: S3 ON          ;turn on (+) edge detector
      WAIT state>1

c1:   IF state>3 JMP c2
      MOVV 0
      IF state=2 JMP err ;hit forward limit while moving backward
      S3 OFF
      WAIT VEL1 > -0.1 ;wait until decel about done
      MOVV -HomeVel    ;move off limit switch
                    ;wait until limit turns on (if decel took axis past
                    ; switch)

      WAIT i2=off

      state=0        ;wait for both limit switches to be the same
      S5 ON
      WAIT state=1
      state=0
      HomeVel = -HomeVel
      limcnt = limcnt+1
      IF limcnt < 2 JMP fwd
      MOVV 0          ;saw both limit switches, no home switch
      JMP err

c2: state=0
      S4 ON           ;now look for edge OFF of switch
      WAIT state>1

      S4 OFF
      IF HSWSTAT=OFF JMP ae2
      MOVV 0
      WAIT VEL1 > -.10 ;wait for decel to about finish
                    ; we are on the switch. If we see a limit before edge
                    ; of switch, fault

      limcnt=1
      JMP c1

;This is the forward move section
fwd:  S3 ON          ;enable edge ON detector
      DISLIM = ON
      MOVV HomeVel
      WAIT state>1   ;wait for (-) edge or limit switch

      IF state>3 JMP mf3 ;jump if saw (-) edge, not limit switch
      MOVV 0          ;saw limit switch, stop
      IF state=3 JMP err ;error: saw reverse limit while moving forward
      S3 off
      WAIT VEL1 < .10 ;wait for decel to about finish
      MOVV -HomeVel  ;move backward off (+) limit switch
      WAIT i1=off
      state=0
      S5 ON
      WAIT state=1

;S5 off
      ;delay .01      ;make sure S5 is off before clearing state

```

```

state=0
HomeVel = -HomeVel
limcnt = limcnt+1
IF limcnt < 2 JMP bkwrdr
MOVV 0
JMP err ;saw both limit switches, no home switch

mf3: MOVV 0 ;state=4 home switch was on
WAIT VEL1 < .10
DISLIM=OFF ;in case we hit a limit switch here
MOVV -creep ;make sure we didn't go through the switch
WAIT HSWSTAT=ON

atedge: DISLIM=OFF
IF HSWSTAT=OFF JMP ae2 ;if already off switch, creep onto switch on the
; switch, move off then creep on looking for (-) edge

MOVV -creep
WAIT HSWSTAT=OFF
MOVV 0

;don't look for rev limit, needed if using reverse
; limit as home switch

ae2: S2 OFF
state=0 ;we are off the switch, creep onto the switch
S3 ON ;looking for (-) edge of switch
MOVV creep
WAIT state>1

DISLIM=OFF
IF INDEXEN=ON JMP jonsw1 ;if will be looking for index, do it
MOVV 0 ;else, move to offset
MOVD HOFFS,V=HVEL
JMP finish

jonsw0: MOVV creep ;looking for index, move forward w/ creep speed

jonsw1: FIDX1 OFF ;clear index flag
IDX1=on ;turn on index1 interrupt
WAIT FIDX1=ON ;wait for index

ifound: MOVV 0
WAIT VEL1 < .1 ;wait for decel to about finish
DWELL .1 ;dwell to let move end
temp=PCMD-IX1P1-HOFFS ;calculate current position
DP temp ;note: dp does not affect FE
MOVD -temp,V=HVEL ;move to the zero spot
JMP fsdh ;jump to finish, skip Define Home

err: DISLIM=OFF ;restore previous settings
JACCEL = origJacc
JDECEL = origJdec
SYSERROR = 72
DWELL 5 ;wait for syserror to abort program

finish: DWELL 1
DH

fsdh: FOUNDDHOME = ON
DISLIM=OFF ;restore previous settings
JACCEL = origJacc
JDECEL = origJdec

END

```



## System Program 26 - Emergency Return

This system program runs after an off to on transition of the Ereturn input, or by setting the system flag ERET. This program is unique because a system STOP function is performed before running the program (all motion programs and present motion are stopped). The default program below performs a MOVP to the position defined in Parameters as the Emergency Return Position. Note: Home must be defined before executing a MOVP.

This program location is useful for other applications because it allows the Ereturn input to automatically stop present operation and run this program. Any program statements are allowed including motion statements.

The ERETURN program is considered a motion program, so it cannot run in parallel with another motion program. Other system programs such as Operator Terminal function key programs may run in parallel with the ERETURN program.

When a program is compiled to save to the ULTRA Plus or IQ as the ERETURN program, the Compiler Options must be set for Program type - "Main" and Add Debug Information - cleared (off). See Part 2 • IQ Master Environment, for a complete description of compiler options.

### Program Listing

```
;Source File Name: ERETURN.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default 'Emergency Return' program.
;This program is saved as System Program #26 in standard personality modules.
;The source is NOT saved in standard personality modules.
;
;This program executes by either activating the Emergency Return input or
;setting the system flag ERET. Any motion or programs will be aborted before
;this program is executed. The only action of this program is an absolute
;move to the position specified by the Emergency Return Position parameter.
;NOTE: Home position must be defined before using an absolute move.
;
;----- Variables used -----
;none
;----- Inputs -----
;Input 10      Emergency Return
;----- Outputs -----
;none
;----- Optional Accessories Required/Supported -----
;none
;----- Begin Program -----
TITLE "Ereturn"
PGMTYPE = MAINPGM           ;Set compiler options
DEBUG = OFF
MOVP=ERETPOS                ;Move to the absolute position specified by the Emergency
                             ; Return Position parameter at the default velocity.
END
```

## System Program 27 - Error Routine

This program runs each time a system error or warning occurs. If a fault has occurred, motion and programs are stopped, the ULTRA Plus or IQ is disabled, and the Error output is set (if defined) before this program runs. If a warning has occurred, motion and programs are allowed to continue, but the Error output is set (if defined) before this program runs.

When a program is compiled to save to the ULTRA Plus or IQ as the EROUTINE program, the Compiler Options must be set for Program type - "Error Program" and Add Debug Information - cleared (off). See Part 2 • IQ Master Environment, for a complete description of compiler options.

The default EROUTINE program below uses the PRINTERR function to print the error message to the Operator Terminal screen.

This program location is useful for applications which may need to automatically reset faults and restart the application program. Other functions such as using a group of outputs to display the error number, or use PRINT statements to prompt an operator on the cause of the fault, may be implemented in this program.

### Program Listing

```
;Source File Name: EROUTINE.QPS
;Version: 3.00
;Tested with IQMaster Version: 3.00
;Tested with IQ firmware Version: 3.00
;----- Description -----
;This is the default Error Handler Program.
;This program is saved as System Program #27 in standard personality modules.
;The source is NOT saved in standard personality modules.
;
;This program runs upon occurrence of a system error or warning. The only
;action is to print an error message to the operator terminal screen.
;NOTE: No message is printed for warnings.
;
;----- Variables used -----
;none
;----- Inputs -----
;none
;----- Outputs -----
;none *      The Error Output(8), if enabled, will turn on
;            automatically upon occurrence of an error.
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;The error message will be printed to the operator terminal screen.
;----- Begin Program -----
TITLE "Eroutine"
PGMTYPE = ERRPGM           ;Set compiler options
DEBUG = OFF
DO
{
  IF (EFLAG = ON)          ;Print messages for errors
  {
    CLR                    ;Clear Operator Terminal Screen
    PRINT 1,8 "ERROR"      ;Print 'ERROR' on first line
    PRINT 2,1              ;Position cursor on 2nd line
    PRINTERR               ;Prints error message
    WAIT EFLAG = OFF       ;Wait for error to be reset
  }
} WHILE (WFLAG = ON)      ;Wait for warning to be reset
;The program does not end until the error is cleared to keep the error
```



```
; message on the operator terminal screen. This is because the Eroutine  
; program uses different 'screen' memory than a main program, so when the  
; error is reset and the eroutine program ends, the screen is restored to  
; what was previously printed.  
END
```

---

## System Programs 28-31

Allen-Bradley reserves these system programs for future development.

# Optional Operator Terminal Function Key Programs

These programs may be used in place of or in addition to the default Operator Terminal function key programs (System Programs 1-24) included in standard Personality Modules.

**NOTE:** The listings for these programs may be viewed or optionally installed to disk. They are stored on the IQ Master disk in the SYSPGM/OPTSYS directory.

---

## Clear Peaks

This program, CLRPEAKS.QPS, clears the peak values of PFE, PVEL1, and PICMD.

---

## Disable

This program, DISABLE.QPS, performs a software disable of the ULTRA Plus or IQ. A software disable can only be cleared by a software enable - not using the Enable input. Motion and programs will be stopped.

---

## Enable

This program, ENABLE.QPS, performs a software enable of the ULTRA Plus or IQ.

---

## Hardware Reset

This program, HRESET.QPS, performs a Hardware Reset of the ULTRA Plus or IQ.

---

## Monitor FVEL1

This program, MONFVEL1.QPS, sets the system variable OTMON to the proper value which displays the system variable FVEL1 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor PFE

This program, MDNPFQPS, sets the system variable OTMON to the proper value which displays the system variable PFE on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor PICMD

This program, MONPICMD.QPS, sets the system variable OTMON to the proper value which displays the system variable PICMD on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor POS1

This program, MONPOS1.QPS, sets the system variable OTMON to the proper value which displays the system variable POS1 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor POS2

This program, MONPOS2.QPS, sets the system variable OTMON to the proper value which displays the system variable POS2 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor PVEL1

This program, MONPVEL1.QPS, sets the system variable OTMON to the proper value which displays the system variable PVEL1 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON

are run, the display will switch to monitor the proper variable.

---

## Monitor VCMD

This program, MONVCMD.QPS, sets the system variable OTMON to the proper value which displays the system variable VCMD on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Monitor VEL2

This program, MONVEL2.QPS, sets the system variable OTMON to the proper value which displays the system variable VEL2 on the top line of the Operator Terminal screen. This display is continuously updated. If this program is run again, the display will be cleared. If other Fkey programs using OTMON are run, the display will switch to monitor the proper variable.

---

## Pause

This program, PAUSE.QPS, will toggle the system flag PAUSE to either pause motion or remove the pause function to allow motion to continue.

---

## Reset

This program, RESET.QPS, will set the system flag SRESET to reset software faults. NOTE: Hardware latched faults such as Encoder error and Peak Current will not be reset by this program. A Hardware Reset (HRESET) is required to reset these faults.

# Optional Home Programs

**NOTE:** The listings for these programs may be viewed or optionally installed to disk. They are stored on the IQ Master disk in the SYSPGM/OPTSYS directory.

---

## Home to a Limit Switch

It is possible to home to the edge of a limit switch. The standard home program can be used by wiring the reverse limit switch to the home switch, and making the Home Switch active open. The alternative home program, `home_rls.qps`, may be used in situations where the application requires use of hardware limit switches but it is not desirable to install an additional home switch. This program uses the reverse hardware limit switch to establish home position. The hardware limit switches functions normally once the program has completed.

---

## Home without Limit Switches

Another alternative home program, `home_nls.qps`, provides the same functions as the standard home program but does not support changing directions if a limit switch is hit.

---

## Home to Encoder Index only

Another alternative home program, `home_inx.qps`, will home an axis to the motor Encoder Index. A direct drive rotary table is an example where this program could be used. This program will seek an index mark in the direction set by the Home Velocity parameter. If the Home Velocity is less than 10,000 pulses/second, the Home Velocity will be used, otherwise the speed will be 10,000 pulses/second.

# Application Examples

**ATTENTION**

These sample programs are included for illustration purposes only. They do *not* include the fault handling logic or safety considerations necessary for an actual machine operation.

---

## SmartBelt Application

### Description

In packaging applications, products are often transported in random fashion from a filling or wrapping operation to a final cartoning machine on a conveyor belt. Many packaging machines require these products to be positioned to a moving reference point (flights, overhead sweeps, etc.). A “smart servo belt” driven by an Allen-Bradley ULTRA Plus or IQ-Series Positioning Drive Module and brushless servo motor will provide uniform spacing to match the moving reference point. The servo controlled belt follows the main machine drive (master), so motion is relative to the machine. The ULTRA Plus or IQ monitors the position of the flights. When a product is detected, a correction distance is calculated and the correction move is performed. The correction move is relative to the machine motion and is added to the speed required to follow the master. This ensures the product will be positioned correctly relative to the flight.

### *Advantages of a Servo System*

- Motion follows machine speed automatically using electronic gearing
- Flexibility of a programmable positioning system allows different product sizes without mechanical changes

- Microprocessor control provides maximum accuracy and speed

## Design Considerations

### *Sensor Placement & Wiring*

The sensor to detect products should be placed so products are detected only after the majority of product weight is on the correction belt so that products will not slip. The sensor to detect flights should be placed so the flight that is detected is the same flight that the current product will be positioned to. This minimizes effects of mechanical differences in flight spacing. The sensors are wired to the interrupt inputs on the ULTRA Plus or IQ for maximum accuracy and response. The example program which follows uses the flight detect sensor wired to Interrupt 1; the product sensor is wired to Interrupt 2. The flight sensor may be mechanically adjustable to eliminate software offset adjustment. The interrupt inputs are edge triggered. Sensors with sinking outputs (NPN) should be used. The interrupt occurs when the input transitions from off to on. Response times of the interrupt inputs are 35 microseconds  $\pm$ 15 microseconds, which determines maximum position accuracy.

### *Mechanics*

The choice of the servo motor and its connection to the belt must be chosen carefully to minimize reflected inertia to the motor. While gear reduction may be used to reduce reflected inertia, make sure motor top speed allows maximum correction belt speed. The system should be tuned to provide maximum response and stability for all ranges of product weights.

### *Correction Belt Length*

The length of the correction belt must be chosen so that worst case correction moves are completed before the product leaves the correction belt. This depends on the acceleration rate used for correction moves and the maximum speed attainable at top machine speed. Make sure that a second product does not enter the correction belt before the first product leaves (based on nominal product spacing). This would cause an inaccurate correction. These two criteria determine the minimum and maximum correction belt lengths. If a belt length cannot be chosen to satisfy both criteria, an alternate approach is to use two correction belts sequentially. The first belt would be limited to correcting only one half of the worst case correction distance, and the second belt would complete the correction. This approach requires a second smart belt servo system and a product sensor at the entrance of the second belt. Both systems may monitor the same flight sensor.

### *Product Slippage*

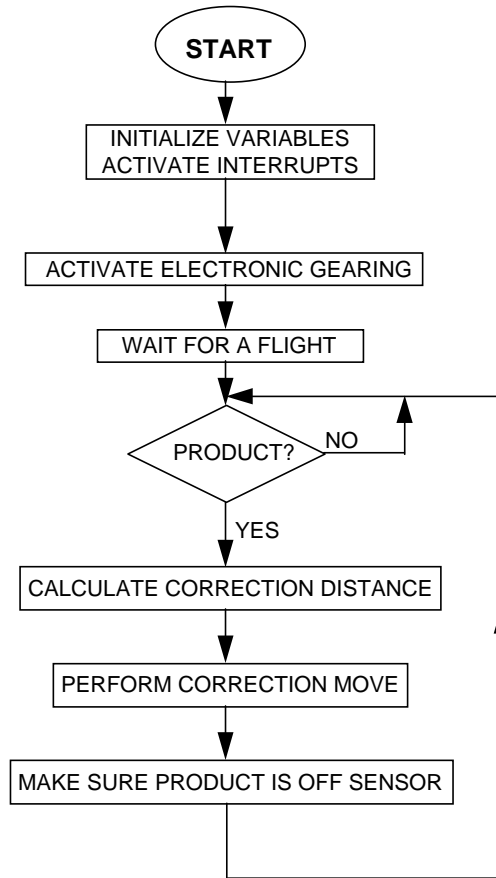
The acceleration rate used for correction moves must be chosen to be small enough so product does not slip on the surface of the belt but large enough so worst case corrections can be completed over the belt length. If friction between product and belt is so low that worst case corrections cannot be completed without slippage, an alternate approach is to “sandwich” the products between two belts. The two belts are mechanically coupled together either above and below the product or side to side. This minimizes product slippage and allows higher acceleration rates.

### *Performance/Accuracy*

Typically, the maximum throughput is limited by product spacing, which limits the correction belt length. If product spacing is small, the corrections need to be made in the positive direction only to prevent product collisions caused by correcting one product negative and the next product positive, targeting the same position. One method of increasing effective correction belt length is to use two correction belts working in conjunction with each other rather than independently. This approach requires both belts to look at product position at the front of the first belt. Both axes calculate a gear ratio which over the length of both belts would properly correct product position. The first belt immediately changes to this gear ratio, while the second belt waits until the product reaches the second belt before changing gear ratios. This allows one product to be on each belt at any time, yet doubles the effective correction belt length.

The positions captured with the flight and product sensors can be motor position or master encoder position. Whichever position has the higher resolution should determine which one to use. A software offset variable can be used to fine tune the target position of products.

## Flow Chart



### ULTRA Plus or IQ Parameter Settings

- CONFIGSet ULTRA Plus or IQ configuration to accept encoder 2 input for electronic gearing
- SCALEnumber of encoder counts per unit distance on driven belt
- SCALE2number of encoder counts per unit distance of belt
- ACCELallowable acceleration rate without slippage of product
- VELmotor top speed during correction at machine top speed

```

TITLE "SBELT"
;Version 3.00 2/29/96
PGMTYPE = MAINPGM
;
ASSIGN OFFSET V1 ;OFFSET is desired distance: flight to product
ASSIGN OFFSET2 V2
;
OFFSET = 1.2345 ;Set OFFSET in user units (inches, cm, etc.)
OFFSET2 = UTOC1 OFFSET ;Convert OFFSET to encoder counts
; based on SCALE parameter setting
INT1 CONT ;Enable interrupts to capture positions
INT2 CONT ;Input 11 is INT1, Input 12 is INT2
GEAR = 1.00 ;Initialize gear ratio
GEAREN ON ;Start tracking the master encoder
WAIT I11 ON ;Wait for first flight
;
MAIN: ;Main loop of program
  
```



```

WAIT I12 ON                ;Wait for a product
V3=@I2P2-@I1P2-OFFSET2    ;Calculate correction move distance:
                            ;product position - flight position - offset. The
                            ;'@'symbol reads position variables in encoder counts
MOV D = -CTOU2 V3         ;Perform correction move (converted back to user
                            ; units)
WAIT I12 OFF              ;Make sure product has cleared the sensor
JUMP MAIN                 ;Continue the process
;
END

```

**NOTE:**The correction distance is calculated in encoder counts so the correction distance is still properly calculated if the internal position counter rolls over between flight and product.

---

## Feed-To-Length Application

### Description

Feed-to-length operations are common in many different industries for a variety of manufacturing processes. Examples of feed-to-length applications include discrete part cutting, forming, or punching, plastic bag manufacturing, press feeds, thermoformers, and sheet metal shearing or forming. A set of pinch or nip rollers is controlled by an Allen-Bradley ULTRA Plus or IQ-Series Positioning Drive Module and brushless servo motor. The rollers feed material the required length into the machine, and then hold the material in position while the operation is in process. On pre-printed materials the ULTRA Plus or IQ-Series controller positions the material relative to a reference or registration mark on the material.

#### *Advantages of a Servo System*

- The operator interface capability of the controller allows quick set-up and change-over between different product lengths.
- Advanced microprocessor control produces maximum accuracy and speed.
- Programmable motion controller provides flexible system for future enhancements.

### Design Considerations

#### *Mechanics*

The servo motor is chosen based on the torque and speed required, and the reflected inertia of the mechanics. Gear reduction through belts and pulleys or a gearbox can help reduce the system inertia reflected to the motor. A low backlash gearbox should be used to maintain accuracy from the motor to the pinch rollers. If timing belts and pulleys are used, HTD tooth profile belts are recommended.

The conversion from rotary to linear motion is accomplished through the pinch rollers. As the diameter of the rollers changes, the inches per turn conversion changes by PI (3.14159) times the change in diameter. Therefore, the rollers should be constructed of a material which will not change diameter significantly with temperature or time. The effect of changing pinch roller diameter is more noticeable for longer moves.

#### *Slippage*

Force is transmitted to the material by the friction of the pinch rollers. As the acceleration increases, the rollers may slip on the material, resulting in short moves. The acceleration must be kept low enough to prevent slippage between the rollers and the material. A second encoder can be added to a measuring wheel on the material to verify the position and allow corrections for slip. The Allen-Bradley ULTRA Plus or IQ-Series controller can generate an S curve motion profile to minimize discontinuities in the velocity which reduces the chance of slippage. The S curve profile requires higher peak torques from the motor and higher peak currents from the drive.

*Accuracy*

The accuracy of the system is determined by the mechanics of the system and the performance of the position controller. To achieve a given accuracy, the resolution of the position feedback should be at least five times better than the desired accuracy. Tuning of the position and velocity control loops also affects the accuracy of the system, as well as the settling time for the system to be in position after a move.

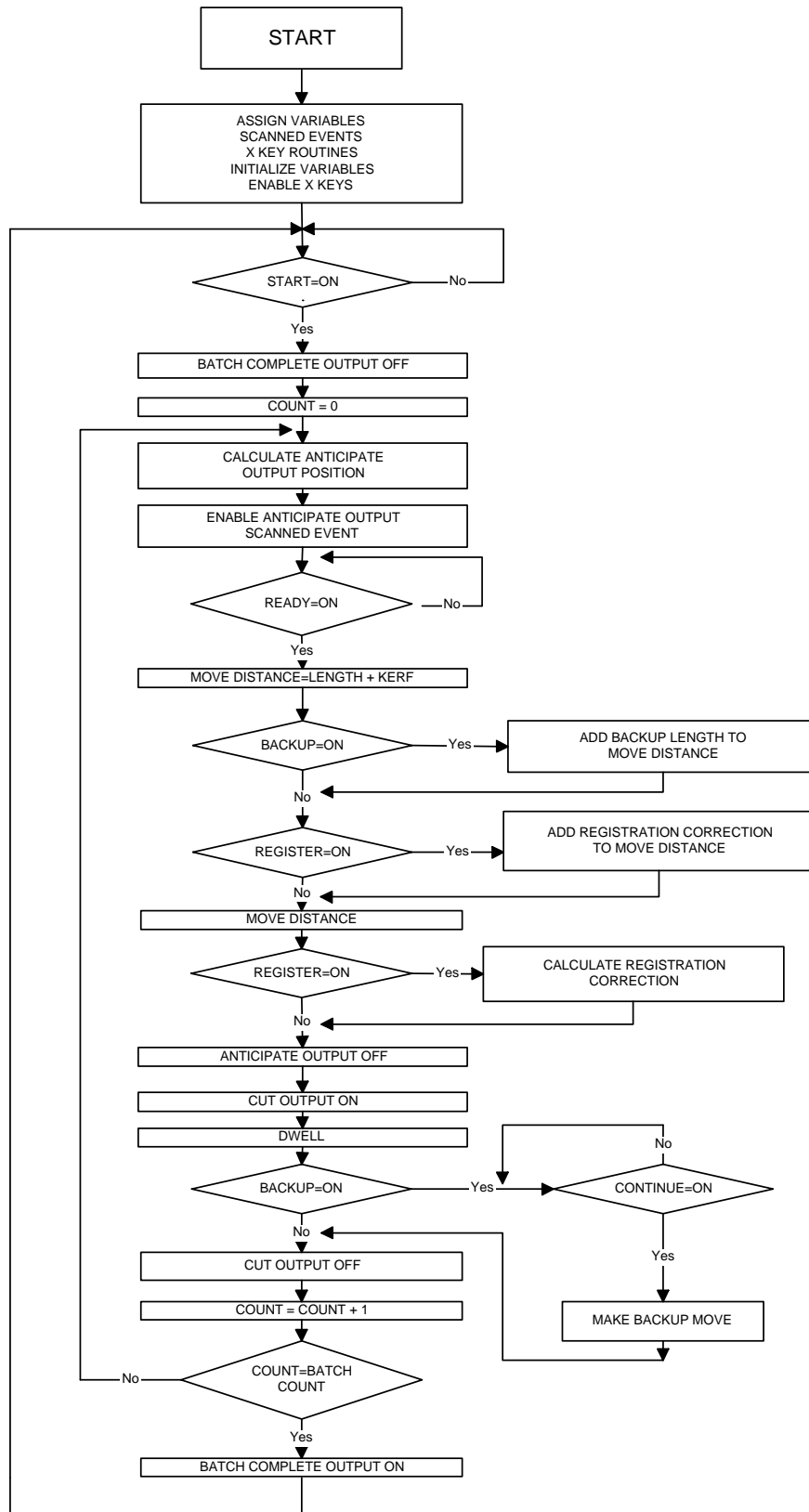
*Interfacing*

The ULTRA Plus or IQ-Series controller provides great flexibility as a programmable motion controller for interfacing to the machine. The program illustrated in this application note provides a “cut” output which turns on to indicate that the material is in position and the operation can begin. A “ready” input indicates that the operation is complete and the controller can begin the next move. Another output is programmed to turn on before the move is complete and can be used to initiate an action prior to the material being in position. Some applications require the material to back away from the knife as the cut is made. In these cases, a “continue” input is used to signal the controller to make the back-up move, before the “ready” input starts the next forward move.

*Registration*

Registration is useful when an operation must be performed relative to a reference position or printed mark on the material. An external registration sensor turns on when the registration mark is detected. The ULTRA Plus or IQ controller records the position where the registration mark is detected, and at the end of the move compares the distance moved past the registration mark to the desired registration distance. If the two distances are different, the correction is added to the next move. This technique works well for correcting for stretch or printing variations which change gradually as the material is unrolled, but it does not work well if the variation from one move to the next is significant. The accuracy of the registration system is affected by the speed of the sensor used, resulting in an error that is a function of the material speed when the registration mark is sensed. The registration sensor output is connected to input 11 on the ULTRA Plus or IQ controller. Input 11 is the trigger for a hardware latch which latches the position within 1.5 microseconds of the off to on transition of the sensor.

Flow Chart



NOTE: Kerf is the offset required to compensate for the thickness of the blade in a saw or cutting operation.

```

TITLE "FEED"
;Version 3.00 2/29/96
;This program implements a Feed-to-Length control with basic features
;such as backup moves, kerf, and registration. It also provides I/O for
;handshaking to the machine such as a cut output, ready and continue
;inputs, and a batch complete output.
;
PGMTYPE = MAINPGM
;Variable name assignments
ASSIGN LENGTH      G1          ;Cut length
ASSIGN COUNT       G2          ;Batch count
ASSIGN SPEED       G3          ;Move speed
ASSIGN ALPHA       G4          ;Acceleration
ASSIGN TIME        G5          ;Dwell time
ASSIGN KERF        G6          ;Saw kerf
ASSIGNBACK_LEN    G7          ;Back up length
ASSIGN OUT_LEN     G8          ;Anticipation output length
ASSIGN REG_LEN     G9          ;Registration length
ASSIGN REL_LEN     V31         ;Output length relative to start of move
ASSIGN DIST        V32         ;Move distance used in cycle
ASSIGNCORRECT     V33         ;Last move distance
ASSIGN SCREEN      V34         ;Current operator terminal screen
ASSIGN STATE       V35         ;State of the control
ASSIGN CREG_LEN    V36         ;Registration length in counts
ASSIGN REG_DIST    V37         ;Recorded registration distance
ASSIGN TEMP_CNT    V38         ;Temporary counter used for batch count
ASSIGN REG_ERR     V39         ;Calculated registration error
ASSIGN READY       I10        ;I10 is the Ready input
ASSIGN CONTINUE    I12        ;I12 is the Backup/continue input
ASSIGN ANTICIPATE  O2         ;O2 is the Anticipate output
ASSIGN CUT         O3         ;O3 is the Cut output
ASSIGN BATCH_CMPLT O4         ;O4 is the Batch complete output
ASSIGN BACKUP      F1         ;Backup move enable flag
ASSIGN REGISTER    F2         ;Registration enable flag
;
;Scanned event S1 turns the anticipate output on when the position
;command exceeds the relative output length
S1:IF @PCMD > REL_LEN ANTICIPATE=ON
;
;This X1 key routine is used to for operator entry of three different
;parameters for the feed to length operation. The actual print and read
;statements used are determined by the variable screen, which indicates
;which menu the operator interface is displaying.
X1:
CLEAR 1
  ON SCREEN JMP X1_0,X1_1,X1_2,X1E
  X1_0:
    READ 1,4 "^D0LENGTH^D1 " ,SHOW LENGTH,7,3(0,999)
    PRINT 1,1 "X1:LENGTH" ,LENGTH,7,3
    JMP X1E
  X1_1:
    READ 1,4 "^DOACCEL^D1 " ,SHOW ALPHA,4,0(0,9999)
    PRINT 1,1 "X1:ACCEL " ,ALPHA,4,0
    ACCEL=ALPHA
    JMP X1E
  X1_2:
    READ 1,4 "^DOOUT ON^D1 " ,SHOW OUT_LEN,7,3(0,999)
    PRINT 1,1 "X1:OUT ON" ,G8,7,3
  X1E:
XEND

```

```

;
;The X2 key routine is similar to X1 for different variables.
X2:
CLEAR 2
ON SCREEN JMP X2_0,X2_1,X2_2,X2E
X2_0:
    READ 2,4 "^DOSPEED^D1    " ,SHOW SPEED,4,0(0,9999)
    PRINT 2,1 "X2:SPEED " ,SPEED,4,0
    JMP X2E

X2_1:
    READ 2,4 "^DODWELL^D1    " ,SHOW TIME,6,3(0,99)
    PRINT 2,1 "X2:DWELL    " ,TIME,6,3
    JMP X2E

X2_2:
    READ 2,4 "^DOBACKUP^D1    " ,SHOW BACK_LEN,7,3(0,999)
    PRINT 2,1 "X2:BACKUP    "
    IF BACK_LEN=0 BACKUP=OFF ELSE BACKUP=ON
    IF BACKUP=OFF PRINT "OFF    "
    IF BACKUP=ON PRINT BACK_LEN,7,3

X2E:
XEND
;
;The X3 key routine is similar to X1 for different variables.
X3:
CLEAR 3
ON SCREEN JMP X3_0,X3_1,X3_2,X3E
X3_0:
    READ 3,4 "^D0COUNT^D1    " ,SHOW COUNT,5,0(0,99999)
    PRINT 3,1 "X3:COUNT    "
    IF COUNT<>0 PRINT COUNT,5,0
    IF COUNT=0 PRINT "CONT"
    JMP X3E

X3_1:
    READ 3,4 "^D0KERF^D1    " ,SHOW KERF,6,3(0,99)
    PRINT 3,1 "X3:KERF    " ,KERF,6,3
    JMP X3E

X3_2:
    READ 3,4 "^D0REG LEN^D1    " ,SHOW REG_LEN,6,3(0,99)
    PRINT 3,1 "X3:REG LEN    "
    IF REG_LEN=0 REGISTER=OFF ELSE REGISTER=ON
    IF REGISTER=OFF PRINT "OFF    "
    IF REGISTER=ON PRINT REG_LEN,6,3

X3E:
XEND
;
;The X4 key routine changes the operator terminal from one menu to the next.
X4:
SCREEN=SCREEN+1
IF SCREEN=4 SCREEN=0
ON SCREEN JMP M0,M1,M2,DI
M0:
    CLEAR
    PRINT 1,1 "X1:LENGTH    " ,LENGTH,7,3
    PRINT 2,1 "X2:SPEED    " ,SPEED,4,0
    PRINT 3,1 "X3:COUNT    "
    IF COUNT<>0 PRINT COUNT,5,0
    IF COUNT=0 PRINT "CONT"
    PRINT 4,1 "X4:NEXT MENU"
    JMP X4E

M1:
    CLEAR
    PRINT 1,1 "X1:ACCEL    " ,ALPHA,4,0

```

```

        PRINT 2,1 "X2:DWELL      " ,TIME,6,3
        PRINT 3,1 "X3:KERF      " ,KERF,6,3
        PRINT 4,1 "X4:NEXT MENU"
        JMP X4E

M2:
    CLEAR
    PRINT 1,1 "X1:OUT ON      " ,OUT_LEN,7,3
    PRINT 2,1 "X2:BACKUP      "
    IF BACKUP=OFF PRINT "OFF "
    IF BACKUP=ON PRINT BACK_LEN,7,3
    PRINT 3,1 "X3:REG LEN      "
    IF REGISTER=OFF PRINT "OFF "
    IF REGISTER=ON PRINT REG_LEN,6,3
    PRINT 4,1 "X4:NEXT MENU"
    JMP X4E

DI:
    CLEAR
    PRINT 4,1 "X4:NEXT MENU"
    IF STATE<>1 PRINT 2,2 "WAITING FOR START"
    IF STATE=1 PRINT 2,3 "RUNNING BATCH"
    IF STATE=2 PRINT 3,3 "BATCH COMPLETE"

X4E:
XEND
;
MAIN:
IF BACK_LEN=0 BACKUP=OFF ELSE BACKUP=ON;BACKUP ON=backup enabled
IF REG_LEN=0 REGISTER=OFF ELSE REGISTER=ON;REGISTER ON=registration enabled
ALL OFF                                ;Initialize all outputs off
CLEAR
PRINT "^D3"
PRINT 2,3 "FEED TO LENGTH"
PRINT 3,5 "CONTROLLER"
DELAY 2
CLEAR
SCREEN=3
STATE=0
PRINT 4,1 "X4:NEXT MENU"
X1 CONT                                ;Enable the X key routines for continuous operation
X2 CONT
X3 CONT
X4 CONT
INT1 CONT
FEEDLOOP:
IF SCREEN<>3 JMP FEED05
CLEAR 2
PRINT 2,2 "WAITING FOR START"
FEED05:
WAIT I4 ON                                ;Wait for start input
BATCH_CMPLT OFF                            ;Turn off batch complete output
STATE=1
TEMP_CNT=0
CORRECT = 0
IF SCREEN<>3 JMP FEED10
CLEAR 2
PRINT 2,2 "RUNNING BATCH"
FEED10:
ACCEL = ALPHA                                ;Set acceleration used in move
CREG_LEN = UTOC1 REG_LEN                    ;Convert registration length to
; encoder counts

IF SCREEN<>3 JMP FEED20
CLEAR 3
PRINT 3,1 "COUNT = " ,TEMP_CNT,5,0

```

```

IF COUNT<>0 PRINT 3,14 "/",COUNT,5,0
FEED20:
REL_LEN = UTOC1 OUT_LEN           ;Calculate new relative output length
                                   ;REL_LEN is position where anticipate output will
                                   ; turn on

REL_LEN = @PCMD + REL_LEN
IF BACKUP ON REL_LEN = REL_LEN + UTOC1 BACK_LEN
S1 ON                               ;Enable scanned event S1
WAIT READY ON                       ;Wait for ready input
DIST = LENGTH + KERF                ;Add Kerf to move length
                                   ;If backup enabled add backup distance to move length

IF BACKUP ON DIST = DIST + BACK_LEN
                                   ;If registration enabled add correction to move
                                   : length

IF REGISTER ON DIST = DIST + CORRECT
MOVD DIST,V=SPEED

                                   ;Measured registration distance is difference between
                                   ; position command and the position latched by the
                                   ; registration mark

REG_DIST = @PCMD - @LPOS
IF REGISTER OFF JMP FEED30

                                   ;Calculate registration error in counts

REG_ERR = CREG_LEN - REG_DIST
CORRECT = CTOU1 REG_ERR             ;Convert to user units
IF CORRECT >= LENGTH CORRECT=0
FEED30:
ANTICIPATE OFF                       ;Anticipate output off
CUT ON                               ;Cut output on
DWELL TIME
IF BACKUP OFF JMP FEED40
WAIT CONTINUE ON                   ;Wait for continue input
MOVD -BACK_LEN,V=SPEED             ;Backup move
FEED40:
CUT OFF                             ;Cut output off
TEMP_CNT = TEMP_CNT + 1
IF TEMP_CNT<0 TEMP_CNT=0
IF COUNT=0 JMP FEED10
IF TEMP_CNT<=COUNT JMP FEED10
BATCH_CMPLT ON                     ;Batch Complete output on
TEMP_CNT=0
STATE = 2
IF SCREEN<>3 JMP FEED50
CLEAR 2
CLEAR 3
PRINT 3,3 "BATCH COMPLETE"
FEED50:
JMP FEEDLOOP
END

```

**NOTE:** The correction distance is calculated in encoder counts so the distance is still properly calculated if the internal position counter rolls over

## Continuous Web with Registration

### Description

Controlling motion relative to a continuous web with registration is common in printing, die-cutting, and labeling processes. The application shown is slicing a printed web into strips. The printed patterns have been staggered to optimize web usage, so the slicing axis needs to cut with a zigzag pattern. This pattern must be registered to the printed pattern.

### Requirements

This axis is required to follow the web speed and correct for any offset differences between the pattern already printed on the web and the pattern the axis is cutting on the web. The controller will need to be able to work in a noisy environment with finite sensor response time.

### Description of the ULTRA Plus or IQ Solution

A registration mark is printed on the web to ensure proper phase alignment. The slave (slicing) axis will follow a printing axis encoder, preferably the axis printing the registration mark. Since the ULTRA Plus or IQ drive knows where the cutter should be each time a registration mark signal is generated, it can measure the difference between where it is and where it should be, and correct for any errors.

### Design Considerations

- Since registration of the slave axis is to the print, not the web, the master encoder (the encoder the slave is following) should be tied to a printing axis, not the web itself. If the print pattern length is truly fixed to a certain amount of web then this distinction is meaningless. In most applications this is not true, however, and since stretch or slippage errors are likely to always be in the same direction, the error introduced by these effects will accumulate. So even though, for example, a print pattern is only 1/64th of an inch longer at the end of a roll than it was at the beginning, this error is cumulative so that each cycle adds another 1/64th of an inch to the error.
- As speeds increase, a significant source of error is the finite response time of registration mark readers, typically a photo eye. If the delay time is known and fixed, its effects can be minimized through software. If high speed precision is crucial, a high-speed reader should be considered because the response of a slow reader will vary more than a high speed reader.
- To avoid having to manually re-align the print each time the machine is started up, the axis must be able to home itself. This need *not* imply that the axis needs to be able to move freely to complete a traditional home cycle. A home routine can be written so that the axis homes itself when the machine is first powered up and the product starts moving. However, if there is more than one encoder marker pulse per revolution of the axis (if there is any reduction from the motor to the axis) then a switch will need to be placed on the axis so that the ULTRA Plus or IQ has some way to know which encoder index it should home to. Also, any motor/axis reduction ratios need to be whole integers, i.e., 6:1, 3:1, 4:1.

### Correction Algorithm

This note deals with web processes that are stable but where small errors (due to web stretch, slippage, etc.) can be introduced and need to be corrected in a manner that will not damage the process. The algorithm is simple: in any given cycle the ULTRA Plus or IQ drive knows the ideal position where it should receive a registration pulse interrupt. This position is called the target. When an interrupt happens, the ULTRA Plus or IQ captures the actual position the ULTRA Plus or IQ drive is in. The difference between the target and the actual position is the Offset Error. Once the pre-determined number of cycles has been measured and the average taken, a correction move is done and the process starts over.

If the machine is homed and the offset known, the initial target position can be computed. If this is not known, the axis must be manually aligned using inputs I15 and I16 to advance and retard the axis. Once aligned, the offset is measured for future reference.

After the initial error is corrected, the subsequent errors should be quite small. If this is the case it is



usually desirable to limit the position around the target that an interrupt will be responded to. This will eliminate most spurious registration marks (due to ink spillage, material blemishes, etc.) and reduce the effect of those that do get through because only those marks close to the correct mark will be seen. This process is called windowing; there is a window around the target position in which registration interrupts will be allowed. The window radius is set by the user.

With a “stable” process there are measures that can be taken to reduce the effects of noise. For example, averaging position error over several cycles can reduce the effect of reader noise. Obviously the more cycles, the less noise there will be; but the system will be slower in responding to a real error. If the patterns per revolution are not evenly spaced, averaging over the number of patterns on the axis will eliminate this “noise.” For these two reasons it is recommended that the number of cycles an offset is averaged over should be a multiple of the number of patterns/revolution.

#### *Program Parameters*

The following program has several parameters that control its behavior. These parameters are the number of patterns to average over, the gear ratio, detector delay, capture window radius, and the pattern increment.

The number of patterns to average over determines how many measurements the program should make before calculating any offset error and performing a corresponding correction. In the following program this is set using the X1 key.

The second parameter is the gear ratio. This is determined by the mechanics of the machine. The gear ratio is how many times the controlled axis should turn for each turn of the master encoder. The ULTRA Plus or IQ can compensate for position errors, even errors due to incorrect gear ratio, but the fewer errors it has to correct the better a system will run. For this reason it may be necessary to “tweak” the gear ratio to compensate for dimensional tolerances, web thickness, etc., to achieve maximum performance from a system.

The next parameter is the detector delay. All position detectors have a finite response time. Some photo detectors can have several milliseconds of delay between the time a photo mark is in a position to be sensed and the time the photo detector output is activated. This delay (in mS) is called PDelay in this program. If the web speed ever changes this can be a real problem since the amount of web passing by in this time will be determined by the speed of the web. Without some sort of compensation, what will happen is that as the web speed changes the ULTRA Plus or IQ will see that the photo-mark position has changed by  $(\text{web speed1} - \text{web speed2}) * \text{PDelay}$ . To compensate for this problem this program uses the position the axis was in PDelay milli-seconds before the detector signal was seen. This works providing that real PDelay is known and repeatable.

The next parameter is the capture window radius (windRad). In a stable web process (this cannot be used with a random position process), once the system is running, it is possible to closely predict where the next photo signal will be and signals that are too far from this spot can be assumed to be noise and ignored. The windowing system in this program does this. Once a measured error is smaller than windRad, only signals that are within windRad from the predicted spot will be used to compute the next position error.

The final parameter is the pattern repeat increment (incr). This is the distance, in motor encoder counts, between patterns and is the distance the target and window positions will be incremented for each pattern. This can be any number of counts, including fractional counts. This is usually easy to calculate. For example, if the slave axis is printing 6 patterns per rev and there is a 4:1 reduction from the motor to the axis, and 8000 counts/rev of the motor, there would be  $4 \times 8000$  counts/rev of the slave axis and  $4 \times 8000 / 6 = 5333 \frac{1}{3}$  counts per pattern. In this case pattern repeat increment should be set to  $1600/3$ . This is hard coded in the program, but if need be the program could be modified to allow the operator to change this number at run-time.

#### *I/O Connections*

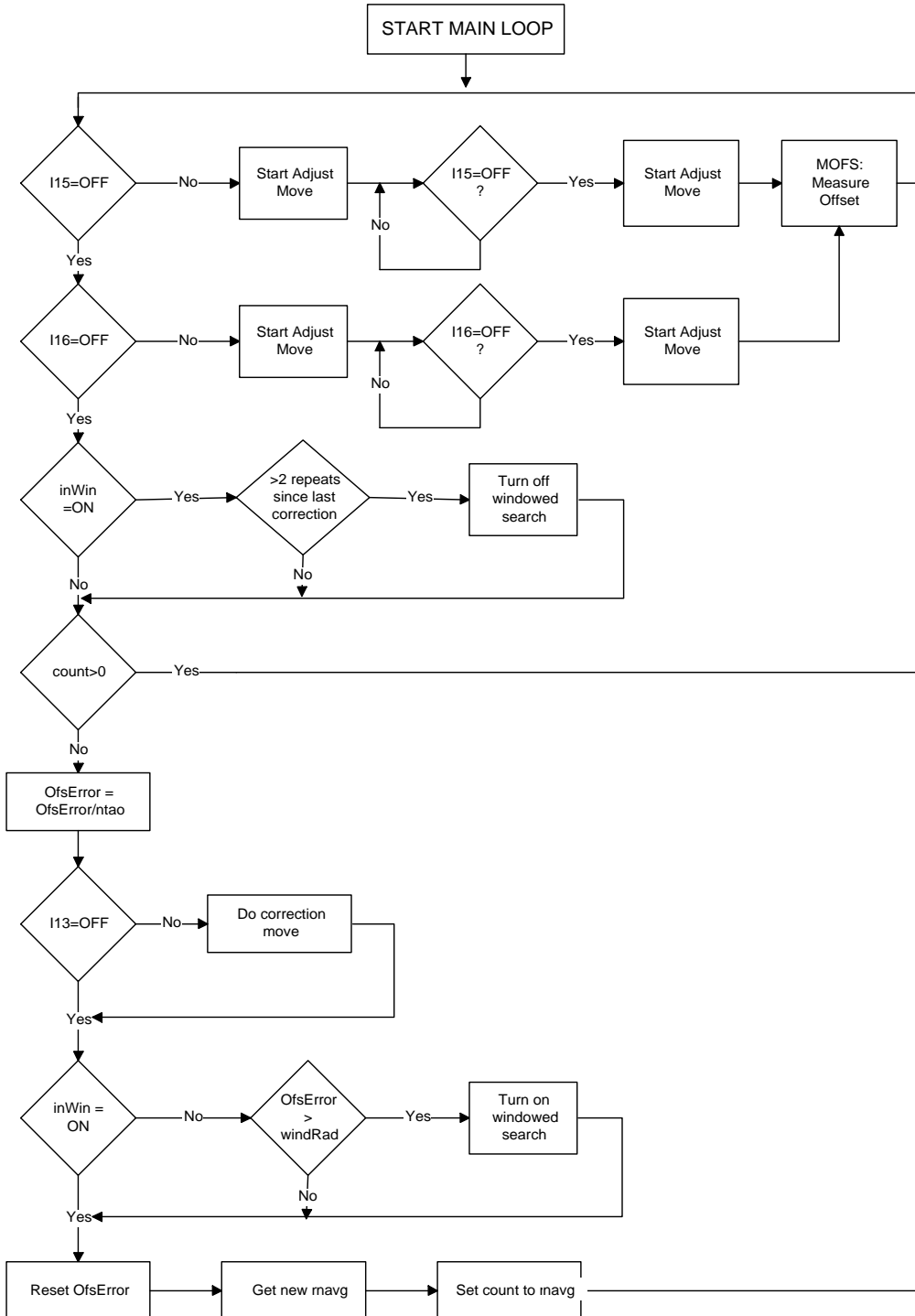
There are four I/O connections in this program. The only critical connection is the sensor connection, which must be connected to input 11. The other connections may be omitted or moved to other inputs by modifying the code. I13, when turned off, will inhibit corrections from being made. This can be useful in determining the action the correction is having on print registration. I15 and I16 are used for man-

ual registration forward and reverse. After either of these switches is let up, the program will look for a signal from the sensor to measure where it is, and, from then on, attempt to hold the axis where it is. I15 is used to advance and I16 is used to retard.

*ULTRA Plus or IQ System Considerations*

There are a few parameters that are set by the programmer in the ULTRA Plus or IQ system. SCALE1 and SCALE2 should be set to 1. The Timebase should be set to seconds. The speed (in counts/second) of correction moves should be put into the default velocity parameter.

**Flow Chart**



APPENDIXES

```

TITLE "WEB"
;Version 3.00 2/29/96
PGMTYPE = MAINPGM
;This program corrects every n patterns and has a capture band to reject
; noise. This program uses a Scale of 1 to compensate for roll-over.
;This was written assuming the web is moving "forward."
; ----- Start of Assignments -----
ASSIGN Target      V1      ;absolute position INT1 should ideally capture
ASSIGN EnPos       V2      ;absolute position to ENable position capture
ASSIGN DisPos      V3      ;absolute position to DISable position capture
ASSIGN OfsError    V4      ;Offset Error--error between ideal and measured
                        ; positions
ASSIGN Incr        V5      ;holds counts between leading edges of patterns
ASSIGN Count       V6      ;number of cycle measurements until error measurement
ASSIGN Navgbuf     V7      ;buffer for changing Navg
ASSIGN PDelaybuf   V8      ;detector delay xkey temp.
ASSIGN CorPos      V9      ;last correction position
ASSIGN Ratio       G1      ;follower/master ratio
ASSIGN Offset      G2      ;ideal offset as a modulus of Incr Valid between 0 and 1
ASSIGN Navg        G4      ;number of prints to use for average
ASSIGN PDelay      G5      ;photo detector delay
ASSIGN WindRad     G6      ;radius of the capture window surrounding the expected
                        ; photo mark
ASSIGN InWin       F1      ;capture window mode flag

; ----- Start of Scanned Events -----
                        ;S1 & S2 are the windowed search
S1: IF (POS1 - EnPos ) > 0;if POS1>EnPos, enable interrupt
INT1 ON                ; (done this way to avoid roll-over problems)

S2: IF (POS1 - DisPos) > 0;if POS1>DisPos
{
    INT1 OFF
    EnPos = EnPos + Incr ;Increment "window" Enable and Disable
                        ; positions
    DisPos = DisPos + Incr
    Target = Target + Incr ;Increment Target
}

S3: IF (POS1 - DisPos) > 0;if POS1>DisPos--this is full-width search
{
    EnPos = EnPos + Incr
    DisPos = DisPos + Incr
    Target = Target + Incr
    INT1 ON                ;enable interrupt for next cycle
}

S4: IF FI1 = ON
{
    IF Count > 0 ;if capture has happened and Count>0
    {
        OfsError = OfsError + Target - (I1P1-PDelay*VEL1)
        OfsError = OfsError + Target - I1P1 + PDelay * VEL1
        Count = Count -1
        FI1 = OFF ;turn off interrupt flag
    }
}

; ----- Start of Xkey Routines -----
X1:                ;X1 key routine
CLEAR
PRINT 3,1 "Patterns to use for"
READ 4,1 "average: ",SHOW Navgbuf (1,99)
CLEAR

```

```

XEND

X2:                                ;X2 key routine
CLEAR
READ 3,1 "Gear Ratio: ",SHOW Ratio
GEAR=Ratio
CLEAR
XEND

X3:                                ;X3 key routine
CLEAR
PDelaybuf=PDelay*1000
IF TBASE=30000 PDelaybuf=PDelaybuf*60
PRINT 3,1 "Detector delay "
READ 4,1 "(mS): ",SHOW PDelaybuf
PDelaybuf = PDelaybuf/1000
IF TBASE=30000 PDelaybuf=PDelaybuf/60
PDelay=PDelaybuf
CLEAR
XEND

X4:                                ;X4 key routine
CLEAR
READ 3,1 "Window Radius: ",SHOW WindRad
IF (WindRad > Incr/2) WindRad=Incr/2
EnPos = Target - WindRad
DisPos = Target + WindRad
CLEAR
XEND

; ----- Start of Main Program -----
main:
X1 CONT
X2 CONT
X3 CONT
X4 CONT
SCALE = 1                                ;scales should already be set to 1
SCALE2 = 1
Incr = 8000                               ;this value should be set to the correct value or
this                                       ; line deleted and the assignment moved to an Xkey

Count = 0
DP -2147400000                            ;start at close to the (-) Count limit
IF Offset >= 1 Offset = .5                 ;if offset is valid, use it
IF Offset < 0 Offset = .5                 ; otherwise use .5
Target =(Offset*Incr) - 2147400000
IF (WindRad > Incr/2) WindRad=Incr/2
EnPos = Target - WindRad
DisPos = Target + WindRad
OfsError = 0
S3 CONT
S4 CONT
InWin = OFF
IF Navg=0 Navg=1
Navgbuf=Navg                               ;make buffer current
CLEAR
GEAR = Ratio
GEAREN=ON
INT1 ON                                    ;for the first time, scanned events
; do this from now on

Ck1: IF I15=OFF JMP Ck2;if I15=ON, offset forward

```

```

D=1,V=1000
DIF Incr,I15=OFF, JMP i1em
i1em: D=1,V=0
      JMP mofs
Ck2:  IF I16=OFF JMP Ck3;if I16=ON, offset backward
      D=-1,V=1000
      DIF -Incr,I16=OFF,i2em
I2em: D=-1,V=0
      JMP mofs
Ck3:
      ;If more than 2 measurement lengths have
      ; gone by with no measurement,
      ; bring window out to a full-width search.
      IF InWin=ON IF (CorPos-pos1)/(Incr*Navg) > 2
      {
          ;use a full-width search
          S1 OFF
          S2 OFF
          S3 CONT
          InWin=OFF
          INT1 CONT
      }
      IF Count>0 JMP Ck1      ;if not ready for a correction yet, loop to Ck1
      CorPos=POS1           ;save correction position for in-window time-out
      OfsError=OfsError/Navg ;find the average Offset Error

      IF I13=OFF JMP skpcor  ;if input 13 is off, no correction is done
                          ; (optional)
      MD OfsError           ;do correction move using system velocity and accel

skpcor: IF InWin=ON JMP scc;if already windowing, skip check to start
        ; windowing
      IF OfsError > WindRad JMP scc;if the Offset Error is less than
        ; window radius, use window
      S3 OFF                ;start windowing
      S1 CONT
      S2 CONT
      InWin = ON
      scc:
      Navg = Navgbuf        ;if new number to average (from Xkey),
                          ; start using it

      OfsError=0
      Count = Navg         ;allow scanned events to collect data again
      JMP Ck1
      mofs:
      CLEAR                ;this measures the current offset and puts the
                          ; phase into Target.
      S1 OFF                ;turn off all scanned events
      S2 OFF
      S3 OFF
      S4 OFF
      FI1 = OFF            ; ignore any previous interrupts
      INT1 ON              ;get next interrupt
      WAIT FI1 ON
      Target = I1P1 - PDelay * VEL1 ;position=position - sensor delay*VEL1
      Offset = Target      ;used later
      FI1 = OFF
      EnPos = Target - WindRad
      DisPos = Target + WindRad
      IF InWin=OFF JMP mofsO
      S1 CONT                ;InWind is ON
      S2 CONT
      JMP mofsO

```

```
mofs0: S3 CONT
mofsc: S4 CONT
Offset = Offset/Incr      ;calculate the new offset
Offset = Offset-(INT Offset)
JMP skpcor
END
```

# Auger Dispensing Application

## Description

In auger dispensing machinery, high performance motion control systems are essential to meet the demands of the application. Since the material being packaged varies in consistency, Allen-Bradley products enable machine builders to provide systems that can adapt readily to changes in packaging sizes and fill rates. Changes are easily made by operator keypad entries to the ULTRA Plus or IQ-Series operator terminal, or by commands sent from a host computer. The conveyor and auger are driven by S-Series or F-Series brushless motors and controlled by ULTRA Plus or IQ-Series Positioning Drive Modules. The weight of the product being packaged is accurately controlled with the ULTRA Plus or ULTRA Plus or IQ-Series Positioning Drive Module by monitoring an analog output signal from a load cell. The signal is proportional to the product weight during the fill cycle. An alternative technique is to move the auger a distance that is proportional to product weight. Coordination of two axis control is accomplished by I/O handshaking between two ULTRA Plus or IQ-Series Positioning Drive Modules.

### *Advantages of Servo System*

- Less waste, higher profits.
- Accurate control of the fill process by an integrated system.
- Operator terminal provides quick setup for changes in packaging process.
- No mechanical changeovers for new production runs.
- Easily interfaces with machine controls such as PLCs or host computers.
- Clutch brake mechanisms and pulley systems are eliminated which means less maintenance.
- Brushless systems provide long life and less down time.
- Precise packaging means better quality control.

### *Mechanics*

Sizing the servo motor and positioning drive module is important for system performance. Ideally a direct drive system (motor to load) with a zero backlash coupling is recommended. If this is not possible due to excessive inertia mismatch between the motor and load, the use of a higher inertia motor (F-Series) and/or a minimal backlash gearbox is recommended. If pulleys are used, timing belts with an HTD tooth profile or belts with similar performance characteristics are suggested.

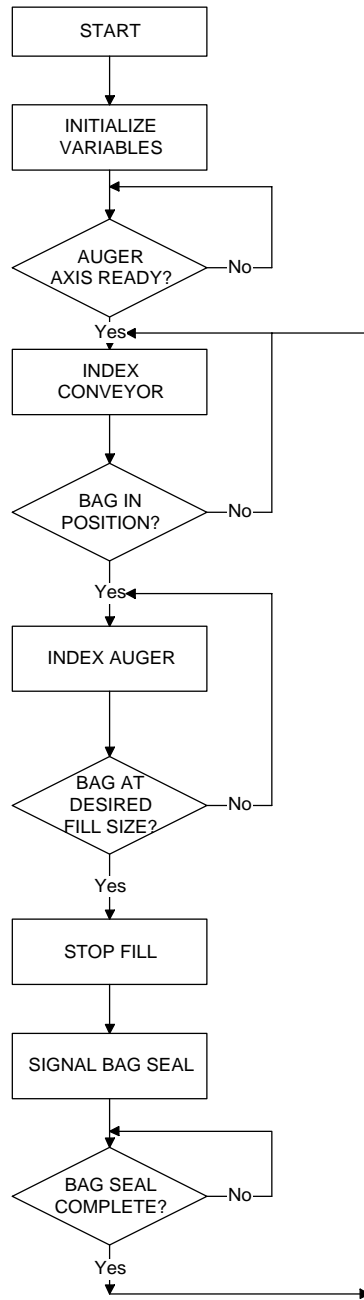
Due to the density of products being dispensed by the auger, speed becomes an important factor. Certain products become compressed at various speeds depending on the pitch of the auger and the size of the drip washer. This causes friction to build up, possibly overloading the drive, and therefore needs consideration when sizing and designing the system. With this in mind, an operator can enter a speed or acceleration variable into the Allen-Bradley operator terminal when packaging a different product.

### *Interfacing*

The ULTRA Plus or IQ controller provides excellent flexibility for interfacing to machine controls whether it be a host computer or PLC sending ASCII commands to the serial port, a PLC controller using 24 VDC opto-isolated inputs and outputs, or sensors interfacing directly to the ULTRA Plus or IQ digital and analog I/O connections.

The program example in this application illustrates the use of the analog input from a load cell to monitor the weight of the package during the filling operation. As the bag nears the specified weight, the fill is finished with a trickle feed. Once the fill cycle is complete, an output turns on to signal the sealing operation to begin. The ULTRA Plus or IQ then waits for a bag fill and seal complete signal before instructing the conveyor axis to index until the next product package is detected.

### Flow Chart



```

TITLE "AUGER"
;Version 3.00 2/29/96
PGMTYPE = MAINPGM
; V Variables Assigned
ASSIGN FILLWGHT V1 ;Weight of bag
ASSIGN AUGERSPD V2 ;Speed of auger
ASSIGN RAPWGHT V3 ;Weight during rapid fill
ASSIGN SLOWWGHT V4 ;Weight during slow fill
ASSIGN OFFSET V9 ;Offset weight for slow fill
;Inputs Assigned
ASSIGN BAGRDY I11 ;Bag in position
ASSIGN SEALDONE I12 ;Sealing operation complete
; Outputs Assigned
  
```

APPENDIXES



```

ASSIGN FSDONE      01          ;Fill & seal complete
ASSIGN AUGERRDY   02          ;Auger axis ready
ASSIGN SEALBAG    03          ;Instruct sealing operation
; Xkey Routines
X1:
CLR
PRINT 1,2 "ENTER BAG WEIGHT"
READ 2,1 "IN LBS -- ", SHOW FILLWGHT
CLR
PRINT 1,2 "X1 = FILL WEIGHT"
PRINT 2,2 "X2 = AUGER SPEED"
F1=ON
XEND
X2:
CLR
READ 1,1 "AUGER SPEED = ", SHOW AUGERSPD
CLR
PRINT 1,2 "X1 = FILL WEIGHT"
PRINT 2,2 "X2 = AUGER SPEED"
F2=ON
XEND
;Start of Main Program
F1=OFF
F2=OFF
X1 CONT
X2 CONT
CLR
PRINT 1,1 "X1 = FILL WEIGHT"
PRINT 2,1 "X2 = AUGER SPEED"
WAIT F1=ON
WAIT F2=ON
NEXT:
CLR
PRINT 1,1 "X1 = FILL WEIGHT"
PRINT 2,1 "X2 = AUGER SPEED"
OFFSET=1          ;Slow feed weight
RAPWGHT=FILLWGHT-OFFSET ;Calculate rapid feed weight variable
SLOWWGHT=RAPWGHT+OFFSET ;Calculate slow feed weight variable
AUGERRDY=ON       ;Output to conveyor, auger ready
WAIT BAGRDY=ON    ;Wait for bag present
AUGERRDY=OFF
CLR
FILL:
D=1,V=AUGERSPD   ;Index auger at variable speed
IF ADC1>=RAPWGHT JMP SLOWFILL
JMP FILL         ;Fill until weight variable is reached
SLOWFILL:
D=.1,V=AUGERSPD
SLOW:
D=.1,V=50
IF ADC1>=SLOWWGHT JMP STOPFILL
JMP SLOW        ;Slowfill until weight variable is reached
STOPFILL:
D=.5,V=0
SEALBAG=ON      ;Seal bag
WAIT SEALDONE=ON ;Bag seal complete
SEALBAG=OFF
FSDONE=ON       ;Output to conveyor, fill & seal complete
DWL .25
FSDONE=OFF
JMP NEXT       ;Continue process
END

```



---

# Thermoformer Application

## Description

Thermoformers are used in many applications where plastic sheet material is formed into shapes to make a product or package. The basic operation of a thermoformer requires the plastic sheet to feed into the machine where it is heated and formed, and then indexed again to a cutter which cuts out the formed part. An Allen-Bradley motion controller drives the sticker chains which pull the plastic material through the machine. An interface terminal allows the machine operator to easily change the move distance, dwell, speed, and other variables.

### *Advantages of Servo System*

- The operator interface capability of the controller allows quick set-up and change-over between different product lengths.
- Advanced microprocessor control produces maximum accuracy and speed.
- Programmable motion controller provides a flexible system for future enhancements.

## Design Considerations

### *Mechanics*

The servo motor should be chosen based on the torque and speed required plus the reflected inertia of the mechanics. Gear reduction through belts and pulleys or a gearbox can help reduce the system inertia reflected to the motor. If a gearbox is used, a low backlash type, such as a planetary technology, should be specified to maintain accuracy. If timing belts and pulleys are used, HTD tooth profile belts are recommended. Except for the main sticker chains, timing belts are recommended because belts provide higher stiffness and less stretch.

### *Thermal Management*

Thermal management within the thermoformer has a significant effect on the performance of the machine. Even cooling of the plastic material is critical to maintain consistent accuracy. Uneven cooling can create stresses that will pull the material out of position as it shrinks, and requires high torques to overcome the additional friction.

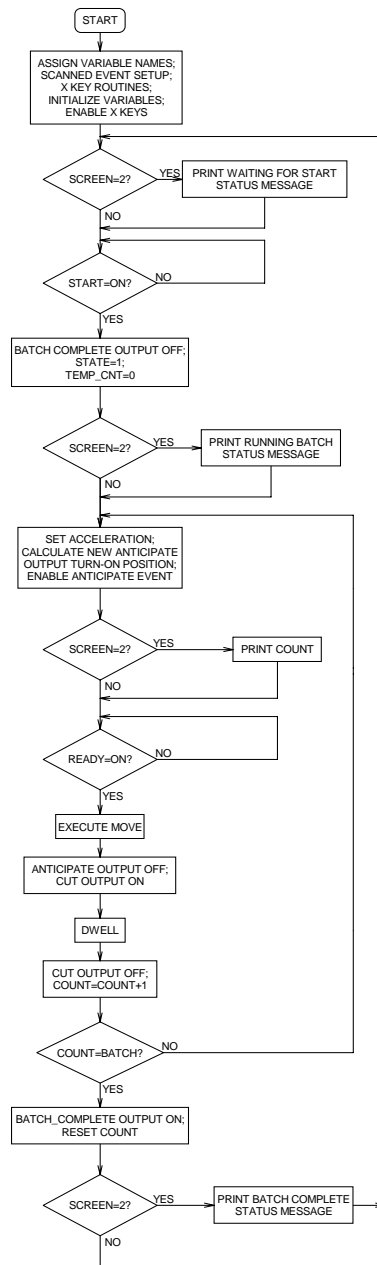
### *Accuracy*

The accuracy of the system is determined by the mechanics of the system and the performance of the position control. To achieve a given accuracy, the resolution of the position feedback should be at least five times better than the desired accuracy. The tuning of the position and velocity control loops also affects the accuracy of the system, as well as the settling time for the system to be in position after a move.

### *Interfacing*

The ULTRA Plus or IQ controller provides great flexibility as a programmable motion controller for interfacing to the machine. The program illustrated in this application note provides an output which turns on to indicate that the material is in position and the forming or cutting operation can begin. A ready input indicates that the operation is complete and the controller can begin the next move. Another output is provided which can be set to turn on before the move is complete and can be used to initiate an action, such as a preheat, prior to the material being in position.

## Flow Chart



TITLE "TFORMER"

;Version 3.00 2/29/96

PGMTYPE = MAINPGM

;This program implements a Thermoformer control with basic features. It also provides I/O for handshaking to the machine such as a cut output, ready input, and a batch complete output.

;Variable name assignments

ASSIGN LENGTH G1

;Move length

ASSIGN COUNT G2

;Batch count

APPENDIXES

```

ASSIGN SPEED      G3          ;Move speed
ASSIGN ALPHA      G4          ;Acceleration
ASSIGN TIME       G5          ;Dwell time
ASSIGN OUT_LEN    G8          ;Anticipation output length
ASSIGN REL_LEN    V31         ;Output length relative to start of move
ASSIGN SCREEN     V34         ;Current operator terminal screen
ASSIGN STATE      V35         ;State of the control
ASSIGN TEMP_CNT   V38         ;Temporary counter used for batch count
ASSIGN READY      I10        ;I10 is the Ready input
ASSIGN ANTICIPATE O2         ;O2 is the Anticipate output
ASSIGN CUT        O3         ;O3 is the Cut output
ASSIGN BATCH_CMPLT O4        ;O4 is the Batch complete output
;
;Scanned event S1 turns the anticipate output on when
; the position
;command exceeds the relative output length
S1:IF @PCMD > REL_LEN ANTICIPATE=ON
;
;This X1 key routine is used for operator entry of
; two different parameters for the thermoformer
; operation. The actual print and read statements
; used are determined by the variable SCREEN, which
; indicates which menu the operator interface is
; displaying.
X1:
CLEAR 1
;Clear the operator terminal screen line 1
;Determine which menu is currently displayed
ON SCREEN JMP X1_0,X1_1,X1E
X1_0:
;If screen = 0 read new length
READ 1,4 "^D0LENGTH^D1 " ,SHOW LENGTH,7,3(0,999)
PRINT 1,1 "X1:LENGTH " ,LENGTH,7,3
JMP X1E
X1_1:
;If screen = 1 read new accel
READ 1,4 "^D0ACCEL^D1 " ,SHOW ALPHA,4,0(0,9999)
PRINT 1,1 "X1:ACCEL " ,ALPHA,4,0
ACCEL=ALPHA
;Update accel used for move
X1E:
XEND
;End of X1 key program
;
;The X2 key routine is similar to X1 for different
; variables.
X2:
CLEAR 2
;Clear operator terminal line 2
;Determine which menu is currently displayed
ON SCREEN JMP X2_0,X2_1,X2E
X2_0:
;If screen = 0 read new speed
READ 2,4 "^D0SPEED^D1 " ,SHOW SPEED,4,0(0,9999)
PRINT 2,1 "X2:SPEED " ,SPEED,4,0
JMP X2E
X2_1:
;If screen = 1 read new dwell
READ 2,4 "^D0DWELL^D1 " ,SHOW TIME,6,3(0,99)
PRINT 2,1 "X2:DWELL " ,TIME,6,3
X2E:
XEND
;End of X2 key program
;
;The X3 key routine is similar to X1 for different
; variables.
X3:
CLEAR 3
;Clear operator terminal line 3
;Determine which menu is currently displayed
ON SCREEN JMP X3_0,X3_1,X3E
X3_0:
;If screen = 0 read new count

```

```

READ 3,4 "^DOCOUNT^D1      " ,SHOW COUNT,5,0(0,99999)
PRINT 3,1 "X3:COUNT      "
IF COUNT <> 0 PRINT COUNT,5,0 ;If count = 0 run continuously
IF COUNT = 0 PRINT "CONT"
JMP X3E

X3_1:                                ;If screen = 1 read new anticipate output length
READ 3,4 "^DOOUT ON^D1      " ,SHOW OUT_LEN,7,3(0,999)
PRINT 3,1 "X1:OUT ON      " ,G8,7,3

X3E:
XEND                                ;End of X3 key program
;
;The X4 key routine changes the operator terminal from one menu to the next.
X4:                                  ;X4 key program
SCREEN = SCREEN + 1                  ;Change screen variable to next screen
IF SCREEN = 3 SCREEN = 0              ;Screen must be 0, 1, or 2
ON SCREEN JMP M0,M1,DI                ;Determine the new screen to be displayed
M0:                                  ;If screen = 0 change display to menu 0
CLEAR                                 ;Clear operator terminal screen
PRINT 1,1 "X1:LENGTH      " ,LENGTH,7,3
PRINT 2,1 "X2:SPEED      " ,SPEED,4,0
PRINT 3,1 "X3:COUNT      "
IF COUNT <> 0 PRINT COUNT,5,0 ;If count not zero print value
IF COUNT = 0 PRINT "CONT"           ; else continuous operation
PRINT 4,1 "X4:NEXT MENU"
JMP X4E

M1:                                  ;If screen = 1 change display to menu 1
CLEAR                                 ;Clear operator terminal screen
PRINT 1,1 "X1:ACCEL      " ,ALPHA,4,0
PRINT 2,1 "X2:DWELL      " ,TIME,6,3
PRINT 3,1 "X3:OUT ON      " ,OUT_LEN,7,3
PRINT 4,1 "X4:NEXT MENU"
JMP X4E

DI:                                  ;If screen = 2 change display to status
CLEAR                                 ;Clear operator terminal screen
PRINT 4,1 "X4:NEXT MENU"
IF STATE <> 1 PRINT 2,2 "WAITING FOR START"
IF STATE = 1 PRINT 2,3 "RUNNING BATCH"
IF STATE = 2 PRINT 3,3 "BATCH COMPLETE"

X4E:
XEND                                ;End of X4 key program
;
ALL OFF                              ;Initialize all outputs off
CLEAR
PRINT "^D3"                          ;Print initial message on operator terminal
PRINT 2,3 "THERMOFORMER"
PRINT 3,7 "CONTROL"
DELAY 2                              ;Wait for 2 seconds
CLEAR                                 ;Clear operator terminal screen
SCREEN = 2                            ;Set screen to status display
STATE = 0                             ;Initial state=0 (waiting for start input)
PRINT 4,1 "X4:NEXT MENU"
X1 CONT                              ;Enable the X key routines for continuous
; operation

X2 CONT
X3 CONT
X4 CONT
TFMRLOOP:                            ;Main program loop
IF SCREEN <> 2 JMP TFMR05
CLEAR 2                               ;If screen = 2 print status message
PRINT 2,2 "WAITING FOR START"
TFMR05:
WAIT I4 ON                            ;Wait for start input

```

```

BATCH_CMPLT OFF ;Turn off batch complete output
STATE = 1 ;Change state to 1 (running batch)
TEMP_CNT = 0 ;Initialize temporary batch counter
IF SCREEN <> 2 JMP TFMR10 ;If screen = 2 print status message
CLEAR 2
PRINT 2,2 "RUNNING BATCH"
TFMR10:
ACCEL = ALPHA;Set acceleration used in move
IF SCREEN <> 2 JMP TFMR20;If screen = 2 print count
CLEAR 3 ;If not continuous show programmed batch
; count
PRINT 3,1 "COUNT = ",TEMP_CNT,5,0
IF COUNT<>0 PRINT 3,14 "/",COUNT,5,0
TFMR20:
REL_LEN = UTOC1 OUT_LEN ;Calculate new relative output length
;REL_LEN is position where anticipate output will turn on
REL_LEN = @PCMD + REL_LEN
S1 ON ;Enable scanned event S1
WAIT READY ON ;Wait for ready input
MOVD LENGTH,V=SPEED ;Execute move
TFMR30:
ANTICIPATE OFF;Anticipate output off
CUT ON ;Cut output on
DWELL TIME ;Dwell for programmed time
TFMR40:
CUT OFF ;Cut output off
TEMP_CNT = TEMP_CNT + 1;Increment count
IF TEMP_CNT < 0 TEMP_CNT = 0 ;Take care of rollover if in
; continuous mode
IF COUNT = 0 JMP TFMR10 ;Check for batch count reached
IF TEMP_CNT <= COUNT JMP TFMR10
BATCH_CMPLT ON ;Batch Complete output on
TEMP_CNT = 0 ;Reset count
STATE = 2 ;Change state to 2 (Batch complete)
IF SCREEN <> 2 JMP TFMR50;If screen = 2 print status message
CLEAR 2
CLEAR 3
PRINT 3,3 "BATCH COMPLETE"
TFMR50:
JMP TFMRLOOP ;Return to start of main program
END

```

---

# In-line Bottle Filler

## Description

The In-line Bottle Filler is typical of a motion control application generally referred to as a flying shear. Flying shear applications require work to be performed on a material while the material is in motion. A flying shear control must accelerate from a stop to match the speed of the moving material at a specific position, and then maintain synchronization of both position and velocity while the work is performed. After the operation is completed, the tool decelerates to a stop and returns to the start or home position. When the correct point on the moving material approaches, the process is repeated. The motion of the flying shear occurs within a fixed distance which cannot be exceeded, so the control must accelerate, track the material while the operation is performed, and decelerate within this distance at the maximum machine throughput. Flying shear applications are common in a variety of industries, including filling, cutting, punching, stamping, forming, labeling, and printing. The Allen-Bradley ULTRA Plus or IQ Positioning Drive Module and brushless servo motor provide the capability necessary for a flying shear control, with the flexibility of a programmable motion controller and the low maintenance of brushless servo technology.

### *Advantages of a Servo System*

- Motion automatically follows machine speed so no reprogramming is necessary if the speed of the system changes.
- The operator interface capability of the controller allows quick set-up and change-over between different sizes and spacing.
- Accurate, repeatable digital control produces high throughput with minimum waste.

## Design Considerations

### *Mechanics*

The servo motor must produce the required torque and speed, and should match the reflected inertia of the mechanics as closely as possible for the best dynamic response. The torque and speed needed will vary depending on the throughput, product, and/or repeat length. A typical machine will run at a higher throughput with a smaller repeat length, and longer repeat lengths at lower throughputs. The worst case torque and speed required is not always at the highest machine throughput and shortest repeat length, so the size of the servo should be based on an analysis of the range of operation of the machine. The operating environment of the motors should also be considered. Food handling applications such as the bottle filler illustrated often require motors which can withstand high pressure wash down with detergents. The bottle filler uses Allen-Bradley W-Series motors, a sealed stainless steel motor which meets IP65 and IP66 ratings.

A belt and pulley system may be more appropriate than a lead screw for a high speed application. The pitch and other mechanics of the lead screw limit the maximum acceleration rate and velocity of the system.

### *Interfacing*

The ULTRA Plus or IQ controller's 24 VDC optically isolated digital I/O provides easy interfacing to sensors and programmable logic controllers. The FILL\_EN output in the program turns on to indicate that the fill head is tracking the bottles with no relative position error and the filling operation can begin. The FILL\_CMPLT input turns on when the filling operation is complete, and the FILL\_EN output is turned off. If the fillerhead reaches the forward travel limit before completing the filling operation, the FILL\_EN output turns off, and the POS\_ERR output will turn on, indicating that a positioning error has occurred.

### *System Parameters*

The ULTRA Plus or IQ Positioning Drive Module system parameters must be set properly for correct operation. Two scale parameters set the number of encoder counts per unit of distance for the motor



encoder and master encoder. The scale parameter should be set for each encoder according to the machine mechanics. For example, if the leadscrew pitch is 5 revolutions per inch, and the encoder on the controlled axis has 8,000 pulses per revolution, then the SCALE parameter for the motor encoder (encoder 1) is  $8,000 \times 5 = 40,000$  counts per inch. The scale factor for the master encoder on the machine is entered in the SCALE2 parameter in counts per user unit.

The ABSOLUTE mode parameter should be turned on. The ULTRA Plus or IQ controller will not allow absolute moves (MOVP) to execute until a home position is established to insure that the lead screw mechanism is operating within the travel limits.

The VELOCITY parameter must be high enough so that when the filling operation is completed the fillerhead can return to its home position before it needs to begin tracking the next batch of bottles. The velocity feedforward gain (KFF parameter) is used to reduce the following error and is entered in percent. The velocity feedforward gain can improve following error while gearing, but can also cause the system to overshoot during incremental moves.

The fillerhead must operate within the travel limits of the lead screw. Hardware limit switches should be used to protect the machine and personnel. The LIMIT inputs parameter enables or disables the forward and reverse limit switch inputs (input 1 and input 2). For additional protection, the FORWARD LIMIT and REVERSE LIMIT parameters set software limits so that system operation will halt and a fault will occur if the fillerhead reaches one of these limits. Position limits are defined within the program to execute a programmed action such as stopping the fill operation if these limits are exceeded. A forward position limit can be defined in a scanned event within the program. The scanned events allow certain conditions to be detected without requiring the use of program statements to poll for their occurrence. When the event is detected, a programmed action is executed.

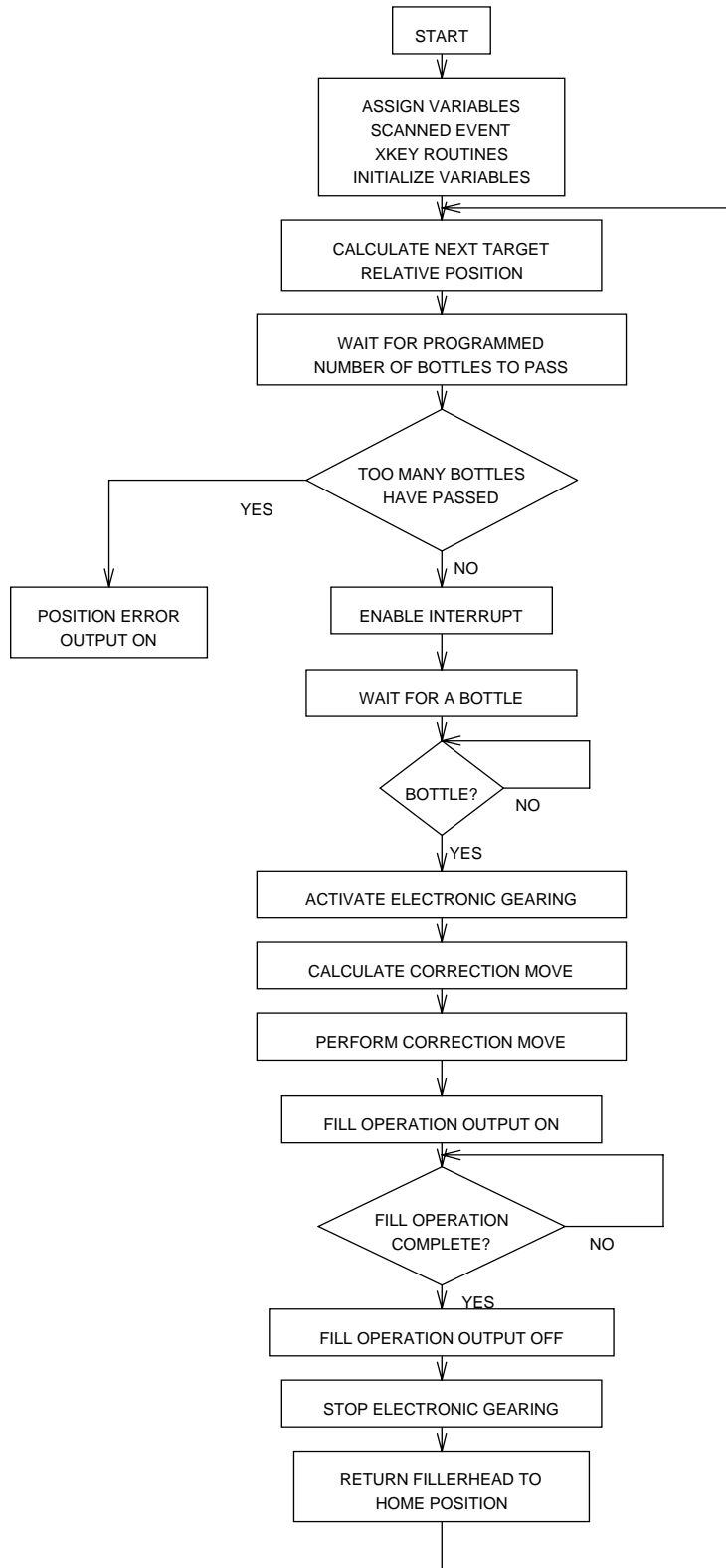
The In-Position WINDOW parameter should be set to relative, so that the In-Position output will be activated anytime the system is within the In-Position window, even while in motion. The In-Position Window defines the maximum position error that is considered in position.

#### *Sensor Placement & Wiring/Registration*

The fillerhead home sensor should be wired to the Home Switch input (Input 5). The Home Switch Active parameter selects the home switch input as active open or active closed.

The bottle sensor should be placed so that when the sensor reads the leading edge of the last bottle to be filled in that sequence, the fillerhead nozzles are approximately over the opening of each bottle. A software offset variable can be used to fine-tune the target position. The program assumes consistent spacing between bottles and waits for the proper distance to pass based on the master encoder before the interrupt is enabled to capture the last bottle to be filled in that sequence. It is also possible to have the sensor detect each bottle and then wait for the correct number of bottles to pass before tracking. The bottle sensor is wired to an interrupt input (Input 11) for maximum accuracy and fastest response. The interrupt inputs are edge triggered. Sensors with sinking outputs (NPN) should be used. The interrupt input latches the position within 50 microseconds of the off to on transition of the sensor. There will be a slight error in position due to this delay. The program calculates this error and executes a move to correct the error.

### Flow Chart



APPENDIXES

```

;Source File Name: FILLER.QPS
;Version 3.00 2/29/96
TITLE "FILLER"
PGMTYPE = MAINPGM
; G Variable Assignments
ASSIGN BOTL_LEN G1 ;Length between each bottle
ASSIGN BOTL_NUM G2 ;Number of bottles being filled in one operation
ASSIGN LAST_BTL G3 ;Position of last bottle where fillerhead began
; tracking conveyor speed
ASSIGN POS_LIM G4 ;Fillerhead forward position limit
;
; V Variable Assignments
ASSIGN FILL_LEN V1 ;Length between bottle 1 and botl_num + 1
ASSIGN NEXT_BTL V2 ;Next position where fillerhead will begin
; tracking conveyor
ASSIGN TEMP1 V3 ;Botl_num - 1 position in counts (Scale2)
ASSIGN FILLER_HOM V4 ;Fillerhead home position
ASSIGN REL_POS V5 ;Relative position between bottles and fillerhead
ASSIGN MAKE_UP V6 ;Correction move distance
ASSIGN CBOTL_LEN V7 ;Length between each bottle in counts (Scale2)
ASSIGN CFILL_LEN V8 ;Length between bottle 1 and botl_num + 1 in
; counts (Scale2)
ASSIGN CFILLER_HOM V9 ;Fillerhead home position in counts (Scale)
; Input Assignments
ASSIGN FILL_CMPLT I10 ;I10 is the fill operation complete input
; Output Assignments
ASSIGN FILL_EN O1 ;O1 is the fill operation enable output
ASSIGN POS_ERR O2 ;O2 is the fillerhead's position error output
; Scanned Events
S1:IF POS1 > POS_LIM ;If fillerhead travels past the forward limit, then:
{
FILL_EN=OFF ;Turn off fill operation output
GEAREN=OFF ;Deactivate electronic gearing
POS_ERR=ON ;Turn on fillerhead position error output
}
; Xkey Routines
;The X1 key routine is used for operator entry of the
; length between each bottle for the bottle
; filling operation.

X1:
CLEAR
PRINT "BOTTLE LENGTH"
READ 2,1 ,SHOW BOTL_LEN,7,3
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
;The X2 key routine allows operator entry of the number
; of bottles being filled in each operation.

X2:
CLEAR
PRINT "NUMBER OF BOTTLES"
READ 2,1 ,SHOW BOTL_NUM,7,0
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
;The X3 key routine allows operator entry of the
; forward travel limit of the fillerhead.

```

```

X3 :
CLEAR
PRINT "POSITIVE LIMIT"
READ 2,1 ,SHOW POS_LIM,7,3
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
;Main Program
X1 CONT ;Enable the Xkey routines for continuous operation
X2 CONT
X3 CONT
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
ALL OFF ;Initialize all outputs off
MOVP 0 ;Move fillerhead to home position (home position
; must first be defined using the home routine)
;Initialize variables
BOTL_LEN = 0
BOTL_NUM = 0
POS_LIM=0
GEAR = 2
LAST_BTL = @POS2 ;Initialize positions
FILLER_HOM = POS1
CFILLER_HOM = UTOC1 FILLER_HOM
WAIT BOTL_LEN <> 0
WAIT BOTL_NUM <> 0
WAIT POS_LIM <> 0
FILL_LEN = BOTL_LEN * BOTL_NUM
CFILL_LEN = UTOC2 FILL_LEN
CBOTL_LEN = UTOC2 BOTL_LEN
S1 CONT ;Scanned event enabled to continuously detect the
event
MAIN:
NEXT_BTL= LAST_BTL + CFILL_LEN ;Calculate next fill position
V20 = 2 * NEXT_BTL ;Scale next fill position according
; to gear ratio
REL_POS = CFILLER_HOM - V20 ;Calculate target relative position between
; fill position and fillerhead
TEMP1 = NEXT_BTL - CBOTL_LEN
WAIT @POS2 >= TEMP1 ;Wait for botl_num - 1 empty bottles to pass
IF @POS2>=NEXT_BTL JMP PSN_ERR ;If botl_num or more empty
; bottles have passed, go to psn_err
INT1 ON ;Enable interrupt to capture next bottle
WAIT I11 ON ;Wait for a bottle
GEAREN ON ;Begin gearing
LAST_BTL = NEXT_BTL ;Store last fill position
DELAY .01
WAIT INPOSN ON ;Wait until fillerhead is tracking fill position
SINT2 ;Generate interrupt to latch pos1 and pos2
V20 = 2*@I2P2
V21 = @I2P1 - V20 ;Calculate relative position
; between fill position and fillerhead
MAKE_UP = V21 - REL_POS ;Calculate difference from target
MAKE_UP = CTOU1 MAKE_UP
MOVD -MAKE_UP ;Execute correction move
WAIT INPOSN ON
FILL_EN ON ;Filling operation output on

```

```
WAIT FILL_CMPLT ON      ;Wait until the fill operation has completed
FILL_EN OFF             ;Turn off filling machine
GEAREN OFF              ;Turn off electronic gearing
MOVP FILLER_HOM        ;Return fillerhead to home position
JMP MAIN                ;Return to start of main program
PSN_ERR:
POS_ERR ON
END
```

**NOTE:** The correction distance is calculated in encoder counts so the distance is still properly calculated if the position counter rolls over.

---

# Vertical, Form, Fill and Seal

## Description

Vertical form, fill, and seal (VFFS) machines are used in a variety of packaging applications. As the name implies, the machine starts with film in a roll, forms the film into a tube, fills the tube with the correct amount of product, and seals the top and bottom of the package. The VFFS machine typically is used in packaging of dry materials such as cereals, snack foods, kitty litter, and fertilizer. The product itself is gravity fed into the tube, often by a precision scale which measures the proper amount of product before the fill operation. The motion control system must feed the film the correct length for each bag. If the film is pre-printed, registration is required to maintain the seal in the correct position relative to the printing. On some machines, a second axis may be used to control the tension on the film. The ULTRA Plus or IQ Positioning Drive Module provides the capabilities required for the film feed with optional registration plus the film tension control. The ULTRA Plus or IQ combines the flexibility of a programmable position controller with the reliability and low maintenance of brushless servo systems.

### *Advantages of a Servo System*

- The operator interface and programmability of the ULTRA Plus or IQ allows quick set-up and change-over between different product lengths.
- Accurate, repeatable digital control produces quality parts with minimum scrap.
- The high speed interrupt input yields precise registration adjustment.

## Design Considerations

### *Mechanics*

The servo motor should be chosen based on the torque and speed required plus the reflected inertia of the mechanics. Gear reduction through belts and pulleys or a gearbox can help reduce the system inertia reflected to the motor. If a gearbox is used, a low backlash type gear (using planetary technology) should be used to maintain accuracy from the motor to the pinch rollers. If timing belts and pulleys are used, HTD tooth profile belts are recommended.

### *Accuracy*

The accuracy of the system is determined by the mechanics of the system and the performance of the position control. In order to achieve a given accuracy, the resolution of the position feedback should be at least five times better than the desired accuracy. The tuning of the position and velocity control loops also affect the accuracy of the system, as well as the time to settle into position after a move.

### *Operator Interface*

The controller must provide the operator with the capability to easily and quickly change the setup of the machine. The variables that can be changed are the product length, the move speed, the registration distance and window (if used), the seal time or dwell between moves, and the gear ratio between the pull belts and the feed rolls which control the tension. The operator should be able to change these variables while the machine is running to allow fine-tuning of the different parameters.

### *I/O Requirements*

The VFFS control requires several I/O connections. The controller must output a signal to begin the seal and cut operation, and must monitor an input indicating that the seal and cut operation is complete. Additional I/O is required for stopping and starting the machine, and possibly for a jog mode to thread the film into the machine.

### *Registration*

Registration is required when the film is pre-printed so the bags are sealed and cut relative to the printing on the film. An external registration sensor turns on when the registration mark is detected. In many cases the registration mark is printed on the material in a position where no other printing or marks will be detected by the registration sensor. Sometimes it is not possible to put the registration mark in a clear lane, and other printing will be detected by the registration sensor. In this case the controller must estab-

lish a window around the nominal registration mark position where it will recognize registration sensor transitions. Any sensor transitions outside this window are ignored. The controller must detect the registration marks and adjust the motion so that the cut and seal is in the proper position relative to the printing on the film.

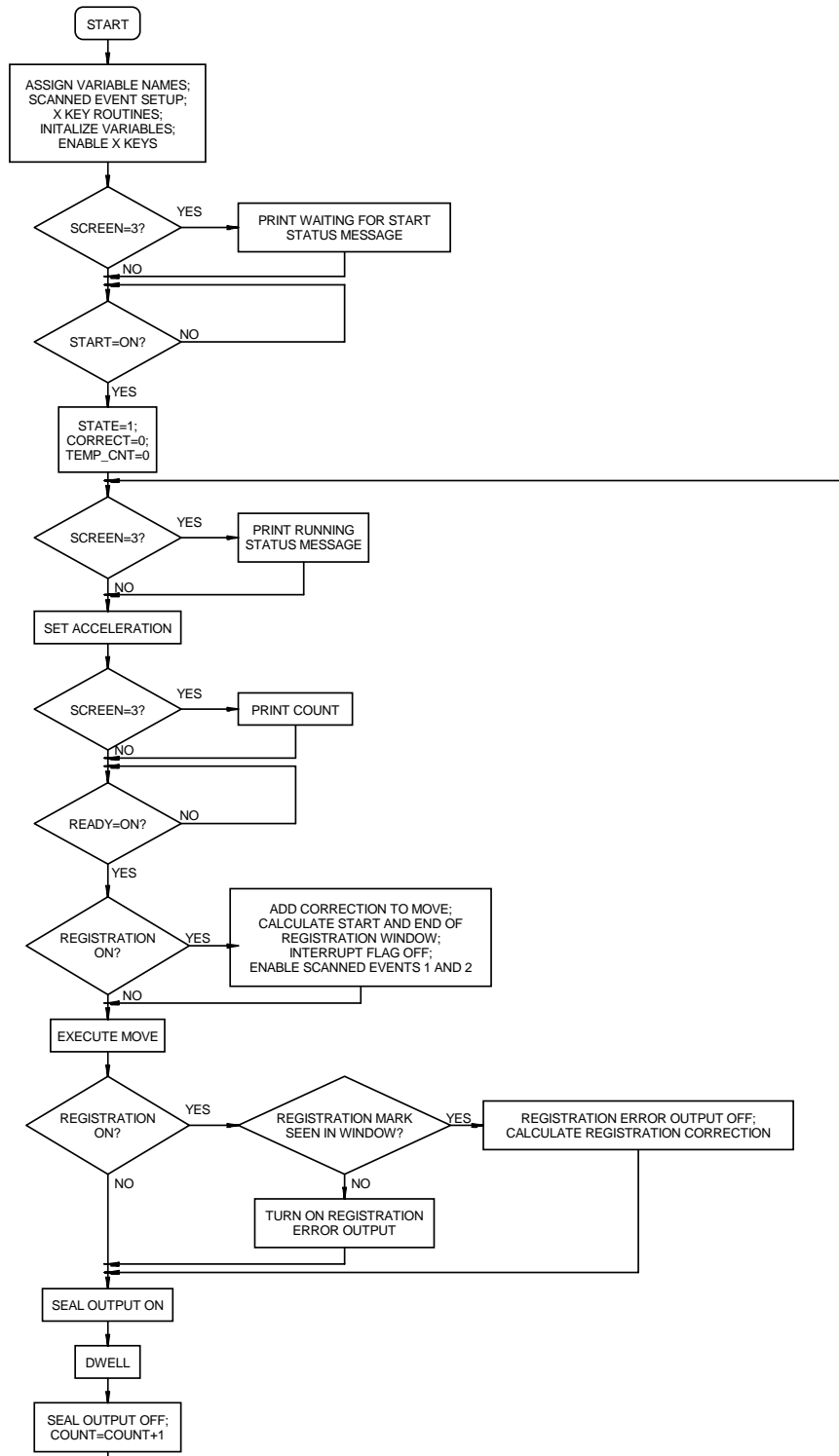
*The ULTRA Plus or IQ Solution*

The ULTRA Plus or IQ controller with the optional operator terminal provides great flexibility as a programmable motion controller for interfacing to the machine. The optically isolated I/O can interface to a wide variety of sensors and output devices. The program shown in this application note provides a seal output which turns on to indicate that the material is in position and the seal operation can begin. A ready input indicates that the operation is complete and the controller can begin the next move.

The operator interface allows the operator to change variables for setup while the machine is running. The programmable function keys on the operator terminal can also replace pushbuttons and switches for stopping, starting, and pausing the machine. The program listed in this application note uses inputs for these functions, and the function keys can easily be set up to either replace or supplement the I/O.

The accuracy of registration is affected by the speed of the sensor used, resulting in an error that is a function of the material speed when the registration mark is sensed. The registration sensor output is connected to input 11 on the ULTRA Plus or IQ controller. Input 11 is the trigger for a hardware latch which latches the position within 1.5 microseconds of the off to on transition of the sensor. The sensor used should have a sinking output (NPN), which turns on when the registration mark is detected. The ULTRA Plus or IQ controller records the position where the registration mark is detected, and at the end of the move compares the distance moved past the registration mark to the desired registration distance. If the two distances are different, the correction is added to the next move. This technique works well for corrections needed due to material stretch or for printing variations which change gradually as the material is unrolled, but it does not work well if the variation from one mark to the next is significant.

### Flow Chart



APPENDIXES

```

;Source File Name: FILLER.QPS
;Version 3.00 2/29/96
TITLE "FILLER"
PGMTYPE = MAINPGM
  
```



```

; G Variable Assignments
ASSIGN BOTL_LEN   G1      ;Length between each bottle
ASSIGN BOTL_NUM   G2      ;Number of bottles being filled in one operation
ASSIGN LAST_BTL   G3      ;Position of last bottle where fillerhead began
                        ; tracking conveyor speed
ASSIGN POS_LIM    G4      ;Fillerhead forward position limit
; V Variable Assignments
ASSIGN FILL_LEN   V1      ;Length between bottle 1 and botl_num + 1
ASSIGN NEXT_BTL   V2      ;Next position where fillerhead will begin
                        ; tracking conveyor
ASSIGN TEMP1      V3      ;Botl_num - 1 position in counts (Scale2)
ASSIGN FILLER_HOM V4      ;Fillerhead home position
ASSIGN REL_POS    V5      ;Relative position between bottles and fillerhead
ASSIGN MAKE_UP    V6      ;Correction move distance
ASSIGN CBOTL_LEN  V7      ;Length between each bottle in counts (Scale2)
ASSIGN CFILL_LEN  V8      ;Length between bottle 1 and botl_num + 1 in
                        ; counts (Scale2)
ASSIGN CFILLER_HOM V9     ;Fillerhead home position in counts (Scale)
; Input Assignments
ASSIGN FILL_CMPLT I10    ;I10 is the fill operation complete input
; Output Assignments
ASSIGN FILL_EN     O1     ;O1 is the fill operation enable output
ASSIGN POS_ERR     O2     ;O2 is the fillerhead's position error output
; Scanned Events
S1:IF POS1 > POS_LIM    ;If fillerhead travels past the forward limit, then:
{
FILL_EN=OFF            ;Turn off fill operation output
GEAREN=OFF            ;Deactivate electronic gearing
POS_ERR=ON            ;Turn on fillerhead position error output
}
; Xkey Routines
                                ;The X1 key routine is used for operator entry of the
                                ; length between each bottle for the bottle filling
                                ; operation.
X1:
CLEAR
PRINT "BOTTLE LENGTH"
READ 2,1 ,SHOW BOTL_LEN,7,3
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
                                ;The X2 key routine allows operator entry of the number of
                                ; bottles being filled in each operation.
X2:
CLEAR
PRINT "NUMBER OF BOTTLES"
READ 2,1 ,SHOW BOTL_NUM,7,0
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
                                ;The X3 key routine allows operator entry of the
                                ; forward travel limit of the fillerhead.
X3:
CLEAR
PRINT "POSITIVE LIMIT"
READ 2,1 ,SHOW POS_LIM,7,3

```

```

CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
XEND
;Main Program
X1 CONT ;Enable the Xkey routines for continuous operation
X2 CONT
X3 CONT
CLEAR
PRINT "LENGTH = " ,BOTL_LEN
PRINT 2,1 "NUMBER = " ,BOTL_NUM
PRINT 3,1 "LIMIT = " ,POS_LIM
PRINT 4,1 "X1:LEN X2:NUM X3:LIM"
ALL OFF ;Initialize all outputs off
MOVP 0 ;Move fillerhead to home position (home position
; must first be defined using the home routine)
;Initialize variables
BOTL_LEN = 0
BOTL_NUM = 0
POS_LIM=0
GEAR = 2
LAST_BTL = @POS2 ;Initialize positions
FILLER_HOM = POS1
CFILLER_HOM = UTOC1 FILLER_HOM
WAIT BOTL_LEN <> 0
WAIT BOTL_NUM <> 0
WAIT POS_LIM <> 0
FILL_LEN = BOTL_LEN * BOTL_NUM
CFILL_LEN = UTOC2 FILL_LEN
CBOTL_LEN = UTOC2 BOTL_LEN
S1 CONT ;Scanned event enabled to continuously detect the
; event
MAIN:
NEXT_BTL= LAST_BTL + CFILL_LEN ;Calculate next fill position
V20 = 2 * NEXT_BTL ;Scale next fill position according
; to gear ratio
REL_POS = CFILLER_HOM - V20 ;Calculate target relative
; position between fill position
;and fillerhead
TEMP1 = NEXT_BTL - CBOTL_LEN
WAIT @POS2 >= TEMP1 ;Wait for botl_num - 1 empty bottles to pass
IF @POS2>=NEXT_BTL JMP PSN_ERR ;If botl_num or more empty
; bottles have passed, go to psn_err
INT1 ON ;Enable interrupt to capture next bottle
WAIT I11 ON ;Wait for a bottle
GEAREN ON ;Begin gearing
LAST_BTL = NEXT_BTL ;Store last fill position
DELAY .01
WAIT INPOSN ON ;Wait until fillerhead is tracking fill position
SINT2 ;Generate interrupt to latch pos1 and pos2
V20 = 2*@I2P2
V21 = @I2P1 - V20 ;Calculate relative position
; between fill position and fillerhead
MAKE_UP = V21 - REL_POS ;Calculate difference from target
MAKE_UP = CTOU1 MAKE_UP
MOVD -MAKE_UP ;Execute correction move
WAIT INPOSN ON
FILL_EN ON ;Filling operation output on
WAIT FILL_CMPLT ON ;Wait until the fill operation has completed
FILL_EN OFF ;Turn off filling machine
GEAREN OFF ;Turn off electronic gearing

```

```
MOV P FILLER_HOM           ;Return fillerhead to home position
JMP MAIN                   ;Return to start of main program
PSN_ERR:
POS_ERR ON
END
```

**NOTE:** The correction distance is calculated in encoder counts so the distance is still properly calculated if the position counter rolls over.

# Lane Diverter

## Description

In material handling, it is often necessary to sort packages based on size. The machine sorts packages by length, using a gate to steer the package to the appropriate output lane.

The ULTRA Plus or IQ Positioning Drive Module (PDM) is required to measure package length by watching a photo-eye and an encoder connected to the conveyor. The axis must wait for any other package to exit the back of the gate before moving the gate to the new destination. Once a package has cleared the back of the gate, it must be moved to the new position before the next package reaches the back of the gate. To minimize mechanical wear and stress, the speed of the moves should be the slowest speed possible that still allows the move to be done in time for the package to be sorted to the correct lane.

The capability of the PDM to monitor an additional encoder and I/O points is used to measure package length. The package is then sorted into three different size groups based on the measured length and a new gate position is chosen. Once any package currently in the gate exits, the speed to move to the new gate position is calculated (based on web speed, package position and distance to be moved) and the move is executed. If the web changes speed during the move, the speed of the move will change to follow it.

## Design Considerations

A homing procedure appropriate to the mechanics of the machine is needed.

This program has been designed to track the progress of up to two packages at a time: one package can be measured and tracked while a second package is directed to its destination position. If more than two packages need to be tracked at a time, the program will need to be modified.

Consideration should be given to the gate length and the longest package length. The gate length needs to be longer than the longest package length and should be long enough so that the difference between the gate length and the longest package, divided by the fastest web speed, gives enough time to do the worst case move. For this reason, the longest packages should be routed to the center lane, since this results in the shortest average move distance for the moves that need to be completed in the shortest time.

### *I/O Connections*

#### Input 1: in sense

This is the input sensor which is used to determine length and position of the incoming packages.

#### Input 12: exit sense1, Input 13: exit sense2, and Input 14: exit sense3

Exit sensors are used to determine if a package is out of the gate.

### *System Parameters*

The following system parameters will affect program behavior. System parameters that affect tuning, I/O, etc. will also need to be set, but will not be covered here. See “Language Reference” on page 175 for more details.

#### Scale

This is the number of motor encoder pulses/unit. The Scale parameter determines the units of numbers entered for G3, G4 & G5 (target positions, described below). For example, this parameter may be set up so that the target positions are in inches from the Home Position.

#### Scale2

This is the number of auxiliary encoder pulses/unit. The Scale2 parameter determines the units of speed for G6 (minimum velocity) as well as the units of distance for G7 (debounce distance).

#### Timebase

This parameter determines if the G6 (minimum velocity) speed is in units/minute or units/second.

#### Accel

This parameter determines the acceleration and deceleration rates used to move to a new position. See Part 5 • Language Reference for more details.

#### *Program Parameters*

There are eight parameters that need to be set by the programmer. These specify details about the particular machine being controlled.

#### G1: range1 and G2: range2

Since there are three destination belts, there are three size ranges. The three size ranges are: i) 0 to range1; ii) greater than range1 to range2, iii) greater range2.

#### G3: gate\_pos1, G4: gate\_pos2 and G5: gate\_pos3

The three size ranges are given three gate destinations, for example, gate\_pos1 is the destination for packages that are sorted into the smallest range, etc.

#### G6: min\_vel

Since for very slow operation, it is desirable to have the gate move at some minimum speed so that the operation of the gate can be observed, this program implements a minimum web velocity. This is the slowest speed the web will be seen as moving, so that even if the web is stopped, the move will continue as if the web were going this speed.

#### G7: deb\_dist

This is the debounce distance, the distance a package must go before looking for a trailing edge. The input sensor will turn on at the leading edge of a package and since this is an edge, the signal may be unstable at this point. For this reason the program requires a certain minimum distance to go by before allowing any OFF transition to be read as a trailing edge of a package.

#### G8: gate\_length

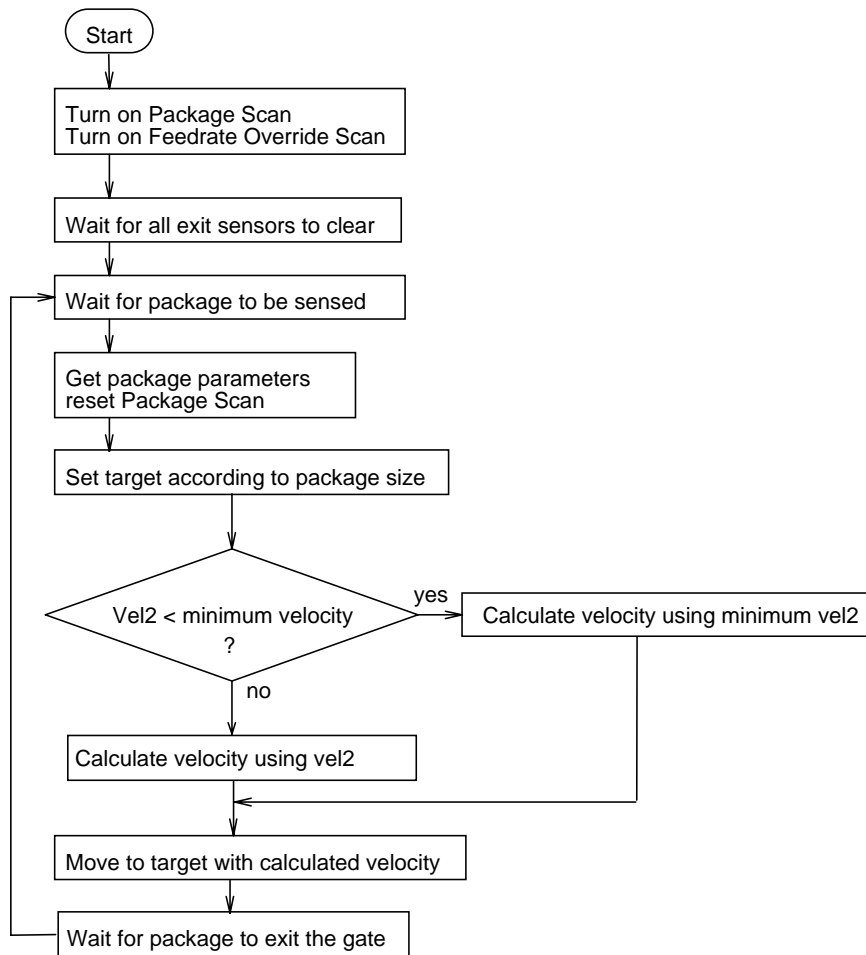
This is the distance from input sensor (in\_sense) to the back of the gate. The difference between this number and the longest package length will be the shortest duration move in terms of web distance.

### **Program Description**

This program consists of two parts: the scanned events and the main program. The scanned events are used to measure packages moving by the input sensor. When a package has been measured, this information is made available to the main program. The scanned events also track the speed of the web and set the feedrate accordingly.

The main program waits for a package to be sensed and measured. Once the package is sensed, the measurement is sorted using G1 and G2 (range1 and range2) and the correct destination is chosen. The velocity to move to this destination is calculated based on the web speed and the distance from the leading edge of the package to the back of the gate. The move is done and the main program waits for the next package to be sensed and measured.

## Flow Chart



```

TITLE "lane_dv2"
;Version 3.00 2/29/96
PGMTYPE = MAINPGM
; G Variable Assignments
;3 sizes of packages: 0-range1, range1-range2, range2 or
; larger
ASSIGN Range1G1
ASSIGN Range2G2
ASSIGN Gate_pos1 G3 ;the first gate position
ASSIGN Gate_pos2 G4 ;the second gate position
ASSIGN Gate_pos3 G5 ;the third gate position
ASSIGN Min_vel G6 ;the minimum web velocity allowed (in case belt stops)
ASSIGN Deb_dist G7 ;debounce distance: distance package must go before
; looking for a trailing edge
ASSIGN Gate_length G8 ;the distance from in_sense to the back of the gate

; V Variable Assignments
ASSIGN Deb_pos V1 ;position that debounce will be done
ASSIGN Pkg_length1 V2 ;the length of the package being sorted
ASSIGN Pkg_length2 V3 ;the length of the package being measured
ASSIGN Eon1 V4 ;leading edge web position for the package being
; sorted
ASSIGN Eon2 V5 ;leading edge web position for the package being
; measured
ASSIGN Target V6 ;the target gate position

```

```

ASSIGN Mov_vel      V7          ;the calculated move speed
ASSIGN V2hold       V8          ;holds vel2 for computation and S4
ASSIGN Calc_tmp1    V9          ;temporary variable for calculation

;      Input Assignments
ASSIGN In_sense     I11         ;the input sensor, used to determine length and
                                ; position
ASSIGN Exit_sense1  I12         ;first exit sensor, used to determine if package
                                ; is out of gate
ASSIGN Exit_sense2  I13         ;second exit sensor
ASSIGN Exit_sense3  I14         ;third exit sensor

;      Scanned Events
S1:IF I11 EDGE ON              ;look for the leading edge of a package
{
    Eon2=pos2
    Deb_pos = Eon2 + Deb_dist
    S2 ON
}

S2:IF pos2 > Deb_pos           ;wait for debounce distance before enabling S3
{
    S3 ON
}

S3: IF I11=OFF                 ;look for trailing edge of package
{
    Eon1=Eon2
    Pkg_length2 = pos2-Eon2
    S1 ON
}

S4:IF V2hold <> 0
{
    IF VEL2 > Min_vel           ;if the master changes speed during the move,
    FDR = 100*VEL2/V2hold      ; this tracks it
    ELSE
    FDR = 100*Min_vel/V2hold
}

;      Main Program Setup
Setup:
Pkg_length2=0                 ;clear any previous measurement
S1 ON                          ;turn on package measurement scans
S4 CONT                        ;turn on feedrate adjust scan
WAIT Exit_sense1=OFF          ;make sure back of gate is clear
WAIT Exit_sense2=OFF
WAIT Exit_sense3=OFF

;      Main Program
Main:
    WAIT Pkg_length2<>0        ;wait for a package to be measured
    Pkg_length1 = Pkg_length2  ;get measurement
                                ;reset so we can tell if another package has been
                                ; measured
    Pkg_length2 = 0
    IF Pkg_length1 > Range2 Target=Gate_pos3
                                ;sort measurement and set
    IF Pkg_length1 <= Range2 Target=Gate_pos2
                                ; gate target position
    IF Pkg_length1 <= Range1 Target=Gate_pos1

```

```

;Multiplying Min_vel by 8 brings the feedrate in
; Scanned Event S4 to 12.5%, and the mov_vel
; calculation to 800% of normal. This brings velocity to
; computed value, but allows speed-up to 16x this speed
; instead of the 2x FDR limit of 200%.
IF VEL2 < Min_vel      ;if web velocity is < Min_vel,
V2hold = 8*Min_vel    ; use Min_vel for calculations
ELSE                  ; else use real web speed for calculation
V2hold = 8*VEL2
Calc_tmp1 = Gate_length-pos2+Eon1
                    ;web distance to do move
                    ; if web distance is zero, do move in .01 web units
IF Calc_tmp1 <= 0 Calc_tmp1 = .01
                    ;mov_vel is (distance to move)/(web distance to do move/
                    ; speed of the web)
Mov_vel = (PCMD-Target)*V2hold/Calc_tmp1
MOVVP Target,V=Mov_vel ;move to target at calculated speed
IF Target=Gate_pos1   ;wait for package to exit gate
{
    WAIT Exit_sense1=ON
    WAIT Exit_sense1=OFF
}
IF Target=Gate_pos2
{
    WAIT Exit_sense2=ON
    WAIT Exit_sense2=OFF
}
IF Target=Gate_pos3
{
    WAIT Exit_sense3=ON
    WAIT Exit_sense3=OFF
}
JMP main
END

```



# Automated Test Station

## Description

An automated test station is used to perform various tests on a manufactured part or assembly. An operator loads the part into a mechanical fixture and starts the test. The Allen-Bradley ULTRA Plus or IQ Positioning Drive Module and brushless motor perform the tests and transmit results to a host computer through a serial interface. A pass or fail indication is given to allow rejected parts to be directed to a rework process and good parts to continue on in the manufacturing process. Some tests that can be performed include: measuring torque at a fixed velocity, stress testing an assembly by applying torque at zero velocity, measuring velocity at a fixed torque, and measuring position deflection at a fixed torque.

### *Advantages of Servo System*

- RS232 or RS422 serial interface to host computer provides data acquisition of test results for analysis to aid in statistical process control and quality management, and flexibility of test setup.
- Digital Positioning Drive Module design allows direct current (torque) measurements as well as velocity and position from within the motion program.
- The operator interface capability allows quick set-up of different test procedures.

## Design Considerations

### *Mechanics*

The mechanical fixture(s) and connections to the motor shaft are dependent on the type of tests being performed, and the physical properties of the parts being tested. If performing torque testing, avoid gear reductions and couplings that may introduce errors in measurements due to frictional losses. If performing velocity and/or position testing, use low backlash, high efficiency gear reductions and couplings to avoid measurement errors.

### *Test Options*

There are four basic test options to consider:

1. Measuring torque at a fixed velocity - This method is used when measuring the running torque of a rotating assembly such as testing efficiency and frictional losses of a gearbox assembly. The motor is connected to the input shaft of the assembly and the controller cycles through different velocities and measures current (ICMD) to the motor at each velocity. The current measurement is then converted to torque by multiplying by the torque constant (KT) of the motor.
2. Stress testing - This method is used when testing the static torque of a fixed part or assembly such as testing the torque of a screw or nut. The motor is connected to the part and the controller applies a fixed torque. If the part does not turn, it passes the test. If the part turns more than a preset tolerance, it fails the test.
3. Measuring velocity at a fixed torque - This method is similar to option 1, but measures velocity while applying a fixed torque. This can also be used to measure frictional losses of a rotating assembly.
4. Measuring position deflection at a fixed torque - This method is used when measuring the amount of deflection that results from applying a fixed torque to an assembly. An example is testing a spring clutch mechanism for the proper range of motion. The motor is connected to the assembly and the controller moves in one direction at a fixed velocity with a preset torque limit. When the motion is stopped by the mechanism, motor position is captured, and motion is reversed. When motion stops again, the position is captured and a total position deflection is calculated. This deflection is compared to a preset tolerance to determine if the part passes or fails. In either direction, if the assembly does not stop the motion within a preset position tolerance, the part fails.

### *Torque Measurement Considerations*

The ULTRA Plus or IQ PDM controls current to the motor, not torque produced. To convert measured

current to torque, the current can be multiplied by the  $KT$  of the motor. There are factors that can affect the accuracy of these results so if high accuracy is needed for torque measurements, some additional calculations may be necessary.

The first factor is the variation of  $KT$  from one motor to another. If multiple stations are performing the same tests, this can be a concern. This variation should be no more than  $\pm 3\%$ , but if necessary, a precise factory measurement of  $KT$  may be obtained.

Another factor is that the  $KT$  of a motor is not completely linear with current. At higher current levels, less torque will be produced per unit current. This can be corrected by determining an equation relating torque to current or by using a table of torque values measured at specific currents.

A third factor is that the  $KT$  of a motor changes slightly with temperature. In this case, a sensor can be placed in the motor to measure temperature, and an analog signal proportional to temperature may be connected to the ADC1 input of the PDM. This temperature information can then be used to correct the torque measurement.

#### *Serial Host Interface*

There are two methods of transmitting test data serially to the host computer. The first method is to use PRINT statements in the program to send data directly to the serial port. This method is not recommended because the host must always be ready to read this data, and also because if a fault occurred while data was being transmitted, the data would be lost.

The preferred method is for the program to store test results in a group of nonvolatile variables. The host computer can then request the test results at any time using host commands. Nonvolatile variables should be used so even if power is removed before the host requests data, the data is preserved so the host can still request the data once power is restored.

The host computer can optionally run tests without running a program on the PDM. Move commands can be sent directly to the PDM, and current and velocity measurements can be obtained with direct serial host commands.

#### *Input/Output Connections*

No I/O connections are needed between the PDM and the host computer. The serial interface is used to pass data and allow the host to control the test process. If an operator terminal is used, no I/O connections are required either - the operator can control the process using the terminal. The following example program assumes the use of a few I/O connections:

#### Input 3 - Enable

If the Enable function for input 3 is assigned in the PDM parameter setup, this input will enable or disable the PDM. A normally closed push-button should be used. The program will halt, the motor will decelerate to a stop, and the PDM will be disabled if this input opens.

#### Input 4 - Start

The Start input is connected to a push-button to allow the operator to start the test after loading the part into the fixture.

#### Input 13 - Stop

If the parameters are set to define input 13 as an “X-Killmotion” input, activating this input with a push-button will Stop motion and program execution but will leave the PDM enabled. This allows a non-emergency method of stopping the test.

#### Output 1 - Pass

Output 1 is connected to a light to indicate the part passed the test. It may also be used to fire a solenoid that moves a gate or conveyor to deflect the part to the proper location to continue in the manufacturing process.

#### Output 2 - Fail

Output 2 is connected to a light to indicate the part failed the test. It may also be used to fire a solenoid that moves a gate or conveyor to deflect the part to the proper location for rework or scrap.

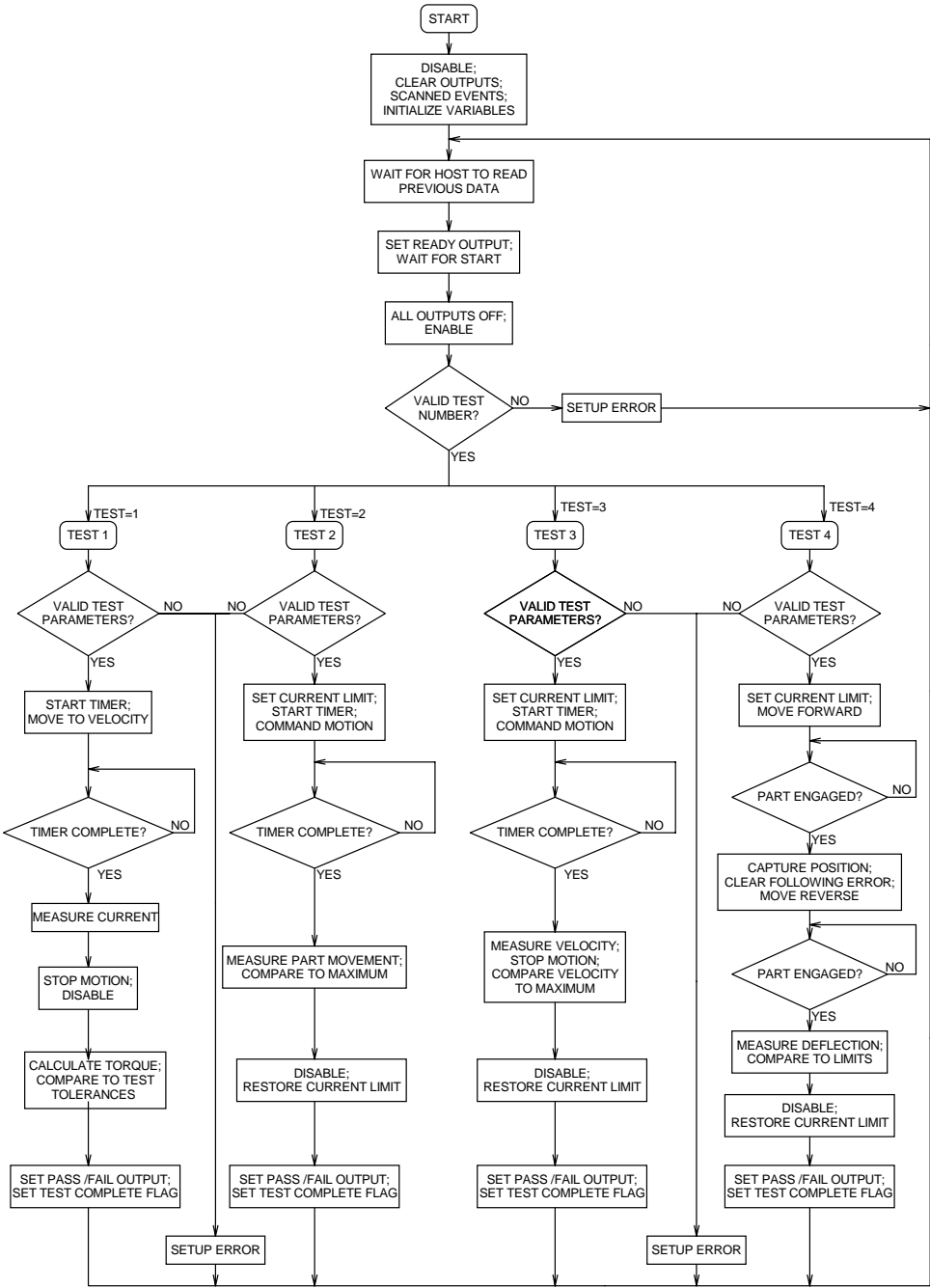
Output 3 - Ready

Output 3 is used to tell the operator that the previous test is complete and to load a new part.

Output 4 - Error

Output 4 can be used to indicate an error in the test setup.

**Flow Chart**



```

TITLE "ATS"
;Source File Name: ATS.QPS
;Version: 3.00 2/29/96
PGMTYPE = MAINPGM
;----- Description -----
;This program performs four different tests on a part:
;Test 1 is used to measure torque of a rotating part or assembly
;Test 2 is used to apply constant torque and measure deflection of a part
;Test 3 is used to apply constant torque and measure velocity
;Test 4 is used to measure bi-directional deflection of an assembly
;Outputs are used to signal if ready for a test and if the part passes or
; fails. The operator activates a start input when the part is loaded.
;A serial interface to a host computer is used to transmit test results and
;accept test parameters.
;
;----- G Variables used -----
ASSIGN Torqueconst      G1      ;Motor torque constant to convert measured motor
                           ; current in amps to torque
ASSIGN Testnumber       G2      ;Determines which test to perform
ASSIGN Testvel          G3      ;Move velocity for torque measurements
ASSIGN Testdelay        G4      ;Delay time before making measurements to allow
                           ; system to stabilize
ASSIGN Currentlimit     G5      ;Current limit used for some tests
ASSIGN Normalcurrent    G6      ;Default current limit
ASSIGN Maxtorque        G7      ;Maximum torque tolerance
ASSIGN Mintorque        G8      ;Minimum torque tolerance
ASSIGN Maxdistance      G9      ;Maximum distance tolerance
ASSIGN Maxvel           G64     ;Maximum velocity tolerance
ASSIGN Minvel           G63     ;Minimum velocity tolerance
                           ;Result storage variables
ASSIGN Test1torque      G10     ;Calculated torque result for test 1
ASSIGN Test2deflect     G20     ;Measured position deflection for test 2
ASSIGN Test3velocity    G30     ;Measured velocity for test 3
ASSIGN Test4deflect     G40     ;Measured position deflection for test 4
;----- V Variables used -----
ASSIGN Meascurrent      V1      ;Used to store measured current
ASSIGN Startpos         V2      ;Used to capture starting position of test
ASSIGN Targetpos        V3      ;Used to calculate target position of move
ASSIGN Test4pos1        V4      ;First engaged position for test 4
;----- B Variables used -----
ASSIGN Testcomplete     B1      ;Used to signal host that data is available
;----- F Variables used -----
ASSIGN Starttimer       F1      ;Used in scanned event to start timer
ASSIGN Timerdone        F2      ;Used in scanned event to indicate timer is done
;----- Inputs -----
ASSIGN Starttest        I4      ;Start the test
ASSIGN Stoptest         I13     ;Stop the test
;----- Outputs -----
ASSIGN Pass             O1      ;Indicates the part passed the test
ASSIGN Fail             O2      ;Indicates the part failed the test
ASSIGN Ready            O3      ;Indicates the test is complete and ready for the
                           ; next part
ASSIGN Setuperror       O4      ;Indicates setup error
;----- Optional Accessories Required/Supported -----
;Operator Terminal - SUPPORTED
;The program can be modified to print messages to the operator terminal
; screen and allow key press control over the tests.
;----- Scanned Events -----
S1: IF Stoptest = ON Stop=ON;If Stoptest input on, stop the program
S2: TMR1 Starttimer = ON Timerdone = ON;Generic Timer routine

```

```

;----- Main Program -----
S1 CONT          ;Activate scan for Stop input
DISABLE         ;Disable IQ to allow operator to move motor
                ; shaft to load the part
ALL OFF        ;Clear all outputs
Ready = ON     ;Signal ready for a new part
Timerdone = OFF ;Clear timer

Begin:         ;Beginning of test loop
WAIT Testcomplete = OFF ;Wait for host to read data and clear flag
WAIT Starttest = ON   ;Wait for start input
ALL OFF        ;Clear all outputs
ENABLE        ;Enable IQ
                ;Check to see which test to perform and jump to the
                ; proper section ON Testnumber JUMP Setuperr, Begin1,
                ; Begin2, Begin3, Begin4

Setuperr:
Setuperror = ON   ;Signal error in test data
DISABLE         ;Disable IQ
JUMP Begin       ;Go back to beginning

                ;Test 1 - Measure torque at a fixed velocity

Begin1:
IF Testvel <= 0 JUMP Setuperr ;Check if valid velocity
IF Maxtorque <= 0 JUMP Setuperr ;Check if valid tolerance
IF Mintorque < 0 JUMP Setuperr ;Check if valid tolerance
TIMER1 = Testdelay ;Set timer value
S2 ON          ;Start timer scanned event
Starttimer = ON ;Start timer
MOVV = Testvel ;Start motion
WAIT Timerdone = ON ;Wait for timer to finish
Meascurrent = ICMD ;Read current
MOVV = 0       ;Stop motion
DISABLE       ;Disable IQ
Test1torque = Meascurrent ;Calculate torque
               * Torqueconst
IF Test1torque > Maxtorque ;Check if torque within tolerance
Fail = ON      ; if not - the part fails
ELSE
{
IF Test1torque > Mintorque
    Pass = ON ; otherwise - the part passes
ELSE
    Fail = ON
}
JUMP Done     ;Finish the test

                ;Test 2 - Stress test

Begin2:
IF Testvel <= 0 JUMP Setuperr ;Check if valid velocity
IF Maxdistance <= 0 ;Check if valid tolerance
JUMP Setuperr
IF Currentlimit <= 0 ;Check if valid Ilimit
JUMP Setuperr
ILIMIT = Currentlimit ;Set current limit
Startpos = PCMD ;Capture starting position
Targetpos = PCMD + Maxdistance ;Calculate minimum move distance
TIMER1 = Testdelay ;Set timer value
S2 ON          ;Start timer scanned event
Starttimer = ON ;Start timer
MOVV = Testvel ;Start moving
WAIT PCMD >= Targetpos ;Wait until commanded motion reaches target

```

```

MOVV = 0 ;Stop commanded motion
WAIT Timerdone = ON ;Wait for timer to finish
Test2deflect = ;Measure how much the part moved
  POSN - Startpos
IF Test2deflect > Maxdistance ;Compare to tolerance
Fail = ON ; to determine pass or fail
ELSE
Pass = ON
DISABLE ;Disable IQ
ILIMIT = Normalcurrent ;Restore current limit
JUMP Done ;Finish the test

;Test 3 - Measure velocity at constant torque

Begin3:
IF Testvel <= 0 JUMP Setuperr ;Check if valid velocity
IF Maxvel <= 0 JUMP Setuperr ;Check if valid tolerance
IF Currentlimit <= 0 ;Check if valid Ilimit
JUMP Setuperr
ILIMIT = Currentlimit ;Set current limit
TIMER1 = Testdelay ;Set timer value
S2 ON ;Start timer scanned event
Starttimer = ON ;Start timer
MOVV = Testvel ;Start moving
WAIT Timerdone = ON ;Wait for timer to finish
Test3velocity = VEL1 ;Measure velocity
DISABLE ;Stop motion
IF Test3velocity > Maxvel ;Check if velocity within tolerance
Fail = ON ; if not - the part fails
ELSE
{
IF Test3velocity > Minvel ; otherwise - the part passes
  Pass = ON
ELSE
  Fail = ON
}
ILIMIT = Normalcurrent ;Restore current limit
JUMP Done ;Finish the test

;Test 4 - Measure position deflection at a fixed
; torque

Begin4:
IF Testvel <= 0 JUMP Setuperr ;Check if valid velocity
IF Maxdistance <= 0 ;Check if valid tolerance
JUMP Setuperr
IF Currentlimit <= 0 ;Check if valid Ilimit
JUMP Setuperr
ILIMIT = Currentlimit ;Set current limit
Targetpos = Maxdistance * 1.1 ;Move ten percent further than tolerance
MOVD = Targetpos ;Start moving
WAIT VEL1 = 0 ;Wait for motion stopped (part engaged)
Test4pos1 = POSN ;Capture engaged position
MOVD = - FE ;Command move to clear following error
MOVD = - Targetpos ;Move in reverse direction
WAIT VEL1 = 0 ;Wait for motion stopped (part engaged)
Test4deflect = ;Calculate total deflection
  Test4pos1 - POSN
IF Test4deflect > Maxdistance ;If greater than tolerance,
Fail = ON ; the part fails
ELSE
Pass = ON ; otherwise the part passes
DISABLE ;Disable IQ
ILIMIT = Normalcurrent ;Restore current limit

```

```
                                ;This section contains instructions done at the end of each
                                ; of the tests
Done:
Ready = ON                      ;Turn on output to indicate test is complete
Testcomplete = ON              ;Set flag to tell host to read data
Timerdone = OFF                ;Clear timer
JUMP Begin                     ;Return to beginning
END
```

# What's New in Each Version

---

## New Features

### Changes to Version 3.2.4 from Version 3.2.3

- IQ Master software runs on Allen-Bradley ULTRA Plus.

### Changes to Version 3.2 from Version 3.0

- Software driven programming of Personality Modules which removes the need for preprogrammed Personality Modules to accommodate each drive/motor configuration.
- Addition of a Purge Motion function.
- Ability to define positions (POS1 and POS2) independently.
- Print and Read statement precision selectable to 4 decimal points.
- Looping and Repeat values are verified for a valid range of 0 to 65535.
- Drive current values assigned to the Monitor Output are now scaled using the drive module (Dm) current. (Previously current values were scaled using ILimit).
- Extended Debug features that allow program execution to be verified, including:
  - a. The ability to execute a program in a traceable mode at full speed or at a reduced speed. The speed reduction is user selectable from 0 to 65.5 seconds.
  - b. The ability to execute subroutine calls as a complete instruction.
  - c. Breakpoints that can be set, cleared, or toggled in the main program, scanned events or Xkey routines.
  - d. User selectable filtering of debug information based on up to thirteen possible program threads.
  - e. A Debugger Thread command that allows the user to specify which source code threads are to be debugged.



## Changes to Version 3.0 from Version 2.1

- Support for IQ Cam
- Tracking function enhancement to electronic gearing
- Compiler options program statements
- Off-line parameter editing
- DDE Server capability
- Support for redefining key codes for generic operator terminals
- Support for connecting one Operator Terminal to multiple IQs
- Movecomplete and Jogactive system flags
- Toolbar button for File Transfer

## Changes to Version 2.1 from Version 2.xx

- Support for the IQ-550 Position Control Module
- Support for a generic operator terminal
- Default File Extension changes - all IQ files begin with "Q"
- Toolbar buttons
- Context Sensitive Help
- Replace function added to the editor
- Ability to monitor G and V variables
- Ability to directly save a file to the IQ after successful compile

## Changes to Version 2.0 from Version 1.xx

- Multiple statements may be on one line:

```
START: V1=2.5 G1=V1*3.1 X3 CONT MD=G13 JUMP MAIN
```

- Statements may be continued on multiple lines:

```
IF F1 = ON
G1 = POS2 ELSE G1 = POS1
```

- Multiple statements may be treated as a group when contained in curly braces:

```
IF F1=ON {O1=ON G1=POS2}
```

- Multiple statements in curly braces can also be used in scanned events:

```
S1: IF F1=ON {O2=ON G1=POS2 F2=OFF}
```

- Two new programming statements have been added:

DO statement WHILE condition example: DO V1=V1+1 WHILE I12=ON

WHILE condition statements example: WHILE I12=ON {V1=V1+1 G3=V1+2}

- Two additional variables have been added for positions captured on occurrence of an index pulse: IX1P2 (POS2 on index 1) and IX2P1 (POS1 on index 2).

```
V1 = IX1P2
```

```
G2 = IX2P1
```

- The ERETPOS variable for the emergency return position parameter may now also be called EPOS.

```
V1 = ERETPOS
```

```
G2 = EPOS
```

- Two variables have been added to allow setting the present jog acceleration and jog deceleration parameters without changing the nonvolatile parameter values:

```
JACCEL = 100
JDECEL = V1
```

- Many other parameters, system variables, and system flags have been added to the programming language. Refer to the Help menu or the Version 2 IQ manual for a complete list of programming language statements.

## Syntax Changes to Version 2.0 from Version 1.xx

### Define Home (DH)

The Define Home statement, DH, cannot specify a position. DH can only be used to define the present position as Home (0). A new statement, DP = *position*, has been added to allow the present position to be redefined. The new position may be a constant or a variable. Both DH & DP reset POS2 to zero.

```
DH ;define present position as home (position 0)
DP = 2.5;define present position as position 2.5
DP = V1;define present position as the position defined
    ;by the contents of V1
```

### PRINT and READ

All PRINT and READ statements must have a comma before a variable name:

```
PRINT 1,2 "distance = " ,G17,5,2;PRINT "The answer is :" ,V1
READ 2,4 "Enter distance " ,V23;READ "Enter distance " ,SHOW G8
```

PRINT and READ statements that print a variable value will now always print the fractional portion as a decimal with the specified precision (or default of 2 decimal places if precision is not specified). A new option has been added to allow printing as a fraction if desired--add "F" after precision. If precision is zero and "F" is specified, the fractional portion will always be displayed as a fraction even if it can be displayed accurately as a decimal (for example 1/100). If precision is 1 to 4 and "F" is specified, the fractional portion will be displayed as a decimal if accuracy allows or else it will be displayed as a fraction.

The complete syntax for the PRINT and READ statements is:

```
PRINT [#n] [row,column] ["text"] [,variable] [,field,precision[,F]]
READ [#n] [row,column] ["text"] , [SHOW] variable [,field,precision[,F]] [(min,max)]
```

### GEAREN

The Gear enable statement, GEAR ON or GEAR OFF, must be changed to GEAREN ON or GEAREN OFF. The statement to set a gear ratio does not change:

```
GEAREN ON
GEAREN OFF
GEAR = 2.5
```

### LOOP

The LOOP statement cannot have a loop name, only a count. The RPT statement cannot be followed by a name as well. Loops may be nested--the RPT statement will refer to the first LOOP statement preceding it.

```
LOOP 100
LOOP V1
RPT
```

The variables for capturing positions on occurrence of an index pulse have changed. The variable IXP1 (POS1 captured on index 1) has been changed to IX1P1. IXP2 (POS2 on index 2) is now IX2P2.

```
G2 = IX1P1
```

The values for the OTMON variable have been incremented by one. (OTMON is used to automatically display system variables on the Operator Terminal screen.) For example, OTMON=1 was used to display POSN. Now, OTMON=2 is used to display POSN. OTMON=255 was used to clear the top line of the Operator Terminal (the monitor line)--now OTMON=0 must be used to clear the monitor line.

```
IF OTMON = 3 OTMON = 0 ELSE OTMON = 3
```

Some Error messages have been changed. Refer to the Help menu or the version 2 IQ Manual.

### **Name Changes for Version 2**

The names of the following system variables and flags have been changed:

<u>Old Name</u>	<u>New Name</u>	<u>Description</u>
Tach	Vel1	Feedback velocity
Ptach	Pvel1	Peak Feedback velocity
Ftach	Fvel1	Filtered feedback velocity
Tcmd	Icmd	Commanded current
Ptcmd	Picmd	Peak commanded current
Dac	Dac1	Analog output

# Upgrading From Version 1

---

## Backup Existing Files and Personality Module

Before you begin the installation process you should make a backup copy of your Personality Module. The Personality Module contains all the parameter and programs for your existing application. If something fails in the installation the backup copy will be required to operate again with version 1.

To upgrade an existing IQ with Version 1.xx firmware, use ECCOMM to make a disk backup of the Personality Module and any application programs.

---

**IMPORTANT**

Backup the Personality Module information before applying power to the IQ. The change makes the Personality Module unusable with Version 1.xx firmware.

---

1. To save the Personality Module contents: From the “PDM> ” prompt, type “SAVE MEM” and press ENTER. The IQ will respond with a message, “Ready to transmit...”. Press PgDn and type a filename (for example, ver1mem.qpm), then press ENTER. When the IQ has finished sending the data (about 1 to 2 minutes), press ESC to complete the transfer.
2. To save application programs: If you already have all your programs saved to disk, you can skip this step. From the “PDM>” prompt, type “SAVE n” and press ENTER, where n is the number of the program you wish to save. The IQ will respond with a message, “Ready to transmit...”. Press PgDn and type a filename (for example, pgm1.qps), then press ENTER. When the IQ has finished sending the data, press ESC to complete the transfer. Repeat this procedure for as many programs that you wish to save to disk.

---

## Install new EPROMS

The following instructions are excerpted from the *ULTRA Plus* or *IQ-Series Installation Manual* (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004). Refer to the appropriate manual for more information.

1. Remove power from the IQ. Remove the cover from the IQ.
2. Remove the firmware EPROMS from the sockets U57, U66, U67, and U87.
3. Install the new EPROMS into the appropriate sockets. The EPROMS are labeled with their corresponding socket locations. Verify that the EPROMS are installed with the correct polarity, as marked by the notch on one end of the EPROM. The notches on all EPROMS should be facing the front of the IQ.

---

### **IMPORTANT**

Electronic components are subject to damage from ElectroStatic Discharge (ESD). Observe ESD procedures when handling electronic circuits.

---



---

## Install IQ Master

Refer to “Getting Started with IQ Master” on page 15.

---

## Upgrade Personality Module

**NOTE:** You will not be able to load, save or run a program until this procedure is completed.

The file, ALLPROG.QAP (in the directory where you installed IQ Master), contains all system programs (formerly called hidden programs), such as the Home and Fkey programs. The file must be uploaded to the IQ after starting IQ Master to upgrade the Personality Module to work with IQ Master and the new firmware.

All parameter settings are maintained from the exiting Personality Module.

### To upload ALLPROG.QAP to the IQ

1. After starting IQ Master, apply power to the IQ.
2. You should see the fault message, “NVRAM Compatibility”, when you select Status from the Monitor menu. This indicates that the Personality Module needs to be upgraded.

**NOTE:** If you are upgrading from a previous release of IQ firmware, you may see a “Compiler Version” error. You will still have to follow these steps to upgrade the Personality Module.

3. Select Transfer from the File menu. Select the All Programs and Send to IQ radio buttons. Choose Start when ready.
4. The Open File Directory dialog box will appear. Select the ALLPROG.QAP file or type “ALLPROG.QAP” and choose OK or press Enter. The file will be sent to the IQ.
5. When you see the message, “File transfer complete”, choose OK. Close the Transfer Control box, then choose Hard Reset from the Run menu. Choose Yes to perform a hard reset. The IQ will be reset. If a fault is present, check Status to determine the fault.

The IQ is now ready to run. You may edit previously saved programs from disk using Open from the File menu. Due to minor syntax changes, programs may have to be edited. Refer to the Appendix C for any required changes to your programs. Compile the program using Compile from the Edit menu. Refer to the Help menu for information on compiling programs and compiler options. After compiling, the

program must be saved to the IQ using Save Program to IQ from the File menu.

**NOTE:** All parameter settings will be preserved from previous values.



# Error Messages

If there is a fault, the ULTRA Plus or IQ can provide specific error messages. The error messages are displayed on the optional Operator Terminal. If a personal computer running IQ Master is connected to the P6 connector, the error message can be displayed in the Monitor menu, Status dialog box.

The error output may be disabled and used as a general purpose output. Therefore, this output may not be available for diagnostic purposes. The remainder of the discussion about the Error Output assumes that output 8 is assigned as an error output. The output can be assigned in the Parameter menu, Outputs dialog box of IQ Master. Refer to Part 2 • IQ Master Environment, Parameter menu, Outputs, for more detailed information.

The Error output becomes active whenever the ULTRA Plus or IQ detects a fault. The ULTRA Plus or IQ will not begin to execute motion programs while a fault is present. After removing the cause of the fault, toggling the ENABLE input OFF and back ON, or issuing a RESET command will clear the fault and allow execution of motion programs. Some faults, however, can only be cleared by an HRESET command. These faults include numbers 42, 45-54, 68, 69, 73, and 74. Refer to the table below for a description of each fault.

On some machines you may want the ULTRA Plus or IQ to continue running even after some faults occur. This is accomplished by setting Disable on Fault to Partial. In this case, the following faults are disabled: Iavg fault, Motor Overtemperature, Soft Forward Limit, and Soft Reverse Limit. All faults will cause the Error output to turn ON (if enabled). However, if Disable on Fault is set to Partial, and a fault listed above occurs, the ULTRA Plus or IQ will remain enabled and execute motion programs normally. The Error output in this case serves as an alarm, and a programmable logic controller (PLC) or other controller can then gracefully shut down the machine without damaging the tooling or work piece. This setting is made in the Parameter menu, System dialog box of IQ Master, or with the FLTDIS Host Language Command. Refer to Part 2 • IQ Master Environment, Parameter menu, System, or to Part 6 • Host Language Commands.



The faults that activate the error output are (the number preceding the fault is the error number):

No.	Error Message	Description
3	2ms Reentrancy Fault	The system did not finish required calculations in the previous 2ms position loop update in time for the next position loop update.
4	Math Overflow	A math overflow occurred (a calculation exceeded the internal system limit). Check the values of the variables used in the program.
5	Divide by Zero	A division by zero calculation was attempted. Check the values of the variables used in the program.
6	LOOP Variable Out of Range	Variable specifying the Loop constant is not within the range of 0 to 65535.
8	PRECISION Out of Range	Decimal point precision is not within the range of 0 to 4.
9	BCD Input Range	A number larger than 9 was read in as a BCD number in a BCD statement in a program. Check the inputs that are being used for the BCD input.
12	LOOP Count Out of Range	Loop constant value is not within the range of 0 to 65535.
13	Profile Calculation	The move cannot be made based on the parameters for the move: distance, velocity, acceleration, or time. Possible causes are variable values used for the move parameters that result in an acceleration or time of 0.
16	Invalid Opcode	The ULTRA Plus or IQ tried to execute a program opcode that was not valid (programs are compiled into opcodes that the ULTRA Plus or IQ executes). Compile the program again to make sure it compiles correctly.
17	Home Not Defined	Home has not been defined when trying to execute a MOV P (absolute) move. Run the Home program or Define Home and execute the MOV P command again.
19	Delay out of Range	Time value for a Delay statement is greater than 65,535 seconds.
20 21 22 23	KP Out of Range KFF Out of Range KI Out of Range KPZ Out of Range	The parameter listed has exceeded the internal system limit. The limit is determined at run time based on the Scale parameter. Check the gain values under the Parameter menu.
24	Gear Out Of Range	The calculated value for the gear ratio denominator is greater than 32,767. Check that the value is correct.
30 31 32	S-Curve Calculation Accel Calculation Velocity Calculation	The calculation listed has exceeded the internal system limit. The limit is determined at run time based on the Scale parameter. Check the velocity and acceleration values under the Parameter menu.

33	Iavg Fault	The average current output of the ULTRA Plus or IQ exceeded safe levels for the motor or PDM module. Check for correct connection of the motor encoder to the ULTRA Plus or IQ or of the motor leads to the PDM. Another possible cause is motor stall or end of travel condition, excessive duty cycle, or no DC bus (drive) voltage. For PDMs only, this fault can also occur if there is power supplied to the L1 and L2 AC AUX inputs, but not to the L1 and L2 AC main inputs (this provides logic voltage, but no DC bus voltage).
34	Excessive FE	The following error has exceeded the Following Error Limit for a time greater than the Following Error Time. Possible causes are loss of encoder feedback, low gain settings, low current limit settings, a commanded velocity or acceleration that exceeds system capabilities, or entries for Following Error Limit and/or Following Error Time that are beyond system capabilities.
35	Excessive Speed	Commanded velocity of a move exceeded the Over-speed value set in the Parameter menu, Velocity/Accel dialog box
36	Cam Profile Overflow	The target follower position in a cam profile could not be calculated.
37	Pos Track Overflow	The target follower position with tracking mode active could not be calculated.
39	Drive Not Ready	IQ-550 only. The Ready to Run signal from the drive went false, indicating a fault in the drive.
40	Motor Overtemperature	The temperature sensor in the motor windings indicated that the motor temperature has exceeded the rating. Check for mechanical binding, undersized motor, or duty cycle too high.
41	Heat Sink Overtemperature	This fault typically indicates that the amplifier is trying to supply currents above its rating to the motor. Check for mechanical binding, excessive duty cycle, or faulty motor wiring. ULTRA Plus or IQ-2000: The ULTRA Plus or IQ-2000 power module has detected excessive internal temperature or excessive current to the motor. On earlier power modules, this fault indicates the heat sink temperature has exceeded the rated temperature. ULTRA Plus or IQ-5000: The heat sink temperature has exceeded the rating.
42	Logic Supply Fault	Power supplies for the amplifier circuitry were below the nominal output voltage. This indicates a problem in the ULTRA Plus or IQ internal power supply.
43	Peak Overcurrent	The PDM current sensors detected a short circuit in the motor leads. Check motor power cables for shorts.

44	Bus Overvoltage	The DC bus voltage exceeded safe levels. This error usually indicates that the power supply shunt is not operating properly. Check the shunt fuse on the Power Supply Module (ULTRA Plus or IQ-5000 only). Refer to the appropriate <i>ULTRA Plus</i> or <i>IQ-Series Installation Manual</i> (1398-5.1, Part Number 0013-1027-004 or Part Number 0013-1022-004).
45 46	Encoder 1 Fault Encoder 2 Fault	Incorrect encoder signals were detected by the ULTRA Plus or IQ circuitry where 'n' is the encoder number. The encoder signals may be out of quadrature or an encoder signal is missing (broken wire).
47	Watchdog Test Fault	The Watchdog circuitry test (performed at power up) failed. The ULTRA Plus or IQ should be replaced.
48	Personality Fault	A checksum error was detected in the personality data for the ULTRA Plus or IQ. Reload the Personality Module from a previously saved file, or replace the Personality Module.
49	Watchdog Reset	The watchdog timer is pulsed by the microprocessor every millisecond to ensure that the microprocessor is running properly. If the microprocessor fails to pulse the watchdog, a Watchdog Reset fault occurs.
50	RAM R/W Fault	The static RAM failed a read/write test. The ULTRA Plus or IQ should be replaced.
51	EPROM Checksum	One of the EPROMs failed the checksum test at power up. All three EPROMs, U66, U67, and U87 should be replaced.
52	TEXT Checksum	The checksum that is stored in the text EPROM did not agree with the checksum calculated at power up. The text EPROM, U57, should be replaced.
53	NVRAM R/W Fault	The NVRAM failed a read/write test and should be replaced.
54	Parameter Checksum	When parameters are entered or changed, a checksum is computed and stored. When powering up, the ULTRA Plus or IQ recalculates the checksum and compares it with the stored value. If the two sums differ, the contents of the NVRAM are assumed to be invalid. The most likely cause of this will be a power failure while entering new data. When a Parameter Checksum error occurs, use the Parameter menu to check for invalid values.
60 61 62 63	Hard Forward Limit Hard Reverse Limit Soft Forward Limit Soft Reverse Limit	A limit input activated (hard limit). If the error is a soft limit, a software travel limit has been exceeded. The Jog inputs can be used to move the ULTRA Plus or IQ off the limit (in the opposite direction).

66	Compiler Version	Indicates that one or more programs need to be re-compiled (the program was compiled with an older version of the compiler). Open the program, compile the program, and save the new compiled program to the ULTRA Plus or IQ. This may have to be done for every program in the ULTRA Plus or IQ (this error may occur when a new version of IQ Master is installed). After compiling and saving all programs to the ULTRA Plus or IQ, do a Hard Reset to clear the error. If you do not re-compile the programs before doing the Hard Reset, any executable programs that have not been re-compiled will be deleted (however, any source programs will still be intact in the ULTRA Plus or IQ).
67	Firmware Version	Indicates that a compiled program requires newer firmware to run (this error may occur when trying to run a program compiled under a newer version of ULTRA Plus or IQ using an older version of firmware). For example, the error can occur if an NVRAM with programs compiled under a newer version of IQ Master is taken from an ULTRA Plus or IQ with newer firmware and installed in an ULTRA Plus or IQ with an older version of firmware. The error occurs because the newer firmware may have new features/program statements not supported in the old firmware. Update the firmware in the ULTRA Plus or IQ.
68	Program Directory	Indicates a checksum error in the program directory. Perform a Hard Reset, then check the program directory. Due to the unpredictable results of losing the directory information, this error may cause the loss of some programs.
69	Program File	When programs are entered or changed, the ULTRA Plus or IQ computes and stores a checksum for the source program and a checksum for the executable program. At power-up and other times during ULTRA Plus or IQ operation, the checksums are recalculated and compared to the stored checksums. If the sums differ, the ULTRA Plus or IQ will determine if the error is in the source or executable program. If the error is in the source program, a warning message is generated. If the error is in the executable program, the executable program is deleted, but the source remains intact. The source should then be re-compiled to create the executable program.
70	Illegal Pgm Number	An invalid program number was used with the Start or Run command. Check to see if the program exists, or, if selecting a program to run from inputs, check that the inputs are set correctly.
71	File Transfer Active	An attempt was made to run a program while a file transfer was in progress.
72	Home Sw Not Found	The Home Switch or the Encoder Index was not seen by the home program between the hardware limits.

73	Uninitialized PM	The ULTRA Plus or IQ cannot operate until the Personality Module (NVRAM) has proper information about the connected drive and motor. Use the Initialize PM selection from the File menu to enter the drive, motor and encoder data.
74	NVRAM Compatibility	The program directory structure (stored in the NVRAM) is incompatible with the current version of the firmware. See the README.WRI file in the IQ Master directory for information on how to transfer the correct directory structure to the ULTRA Plus or IQ.



# Index

## Symbols

@ ..... 177

## Numerics

2ms Reentrancy Fault ..... 419

## A

ABORT ..... 177

Abort Program ..... 177

About IQ Master ..... 4, 81

ABSMODE ..... 178, 309

Absolute (MOV<sub>P</sub>) Motion ..... 156

Absolute Mode ..... 40, 178

ACCEL ..... 42, 178, 301

Accel Calculation ..... 419

Acceleration ..... 42, 178

    Host Commands ..... 301

    JACCEL ..... 226

    JDECEL ..... 227

    Jog Acceleration ..... 226

    Jog Deceleration ..... 227

    MOV<sub>V</sub> Acceleration ..... 226

    MOV<sub>V</sub> Deceleration ..... 227

    SLEW ..... 267

Acceleration Feedforward Gain ..... 206

Accuracy, Math ..... 128

ACTIVESN ..... 310

ACTIVEXN ..... 310

ADC1 Fkey Program ..... 343

ADC<sub>2,3,4,5</sub> ..... 304

ADC<sub>n</sub> ..... 44, 93, 179

ADDR ..... 309

Addressing ..... 290

AFEED ..... 304

ALL ..... 180

All Outputs ..... 180

Allen-Bradley Support ..... 5

Analog Input ..... 93, 179

Analog Output ..... 93

AND ..... 129

Arithmetic ..... 129

ASSIGN ..... 181

Assignable Inputs ..... 86

Assignable Outputs ..... 92

ATHOME ..... 46, 92, 182, 306

AUTO ..... 295

Auto Program ..... 327

Auto Tune ..... 58

AUXID ..... 310

Average Current ..... 219

Axis Select ..... 77

## B

Backup Existing Files and Personality Module .  
414

Baud Rate ..... 49, 78

BCD ..... 183

BCD Input Out of Range ..... 183

BCD Input Range ..... 419

BIN ..... 183

Bn ..... 182, 314

Boolean

AND ..... 129

NAND ..... 130

NOR ..... 130

NOT ..... 130

OR ..... 130

XOR ..... 130

Boolean Operators ..... 129

BRU Emulation ..... 79

Bus Overvoltage ..... 421

## C

CALL ..... 184

Call Subroutine ..... 184

Cam Profile Overflow ..... 420

Case Statement ..... 240

Cause System Error ..... 271

Cause System Warning ..... 272

CF ..... 187, 310, 313

Checksum ..... 293

CLEAR ..... 184

Clear Operator Terminal ..... 184

Clear Peaks Fkey Program ..... 356

Clear Peaks Variables ..... 185

Clipboard ..... 35

Closed Loop Position Control ..... 164

Closed Loop Velocity Control ..... 165

CLR see CLEAR

CLRPEAKS ..... 185, 296

Combination Motion ..... 159

COMMENTS ..... 185

Communications Menu ..... 49

Axis Select ..... 77

Baud Rate ..... 78

BRU Emulation ..... 79

Communications Port ..... 78

CR (Carriage Return) Translation ..... 78

Data Bits ..... 78

Flow Control ..... 78

Local Echo ..... 79

Parity ..... 78

PC Set Up ..... 78

Receive File CR Translation ..... 79

Send File LF Toggle ..... 79

Stop Bits ..... 78

Terminal ..... 77

Communications Port ..... 78

COMn ..... 309

Compile ..... 36

Compiler Options

Compiler Output ..... 37

COMPILEMEMORY ..... 187

DEBUG ..... 191

Debug Information ..... 37

Expand Macros ..... 37

EXPANDMACROS ..... 202

List Files ..... 37

LISTFILE ..... 233

PGMTYPE ..... 248

Program Type ..... 37

Compiler Version ..... 422

COMPILEMEMORY ..... 187

CONFIG ..... 168, 187

Configuration ..... 168, 187

Configuration Menu ..... 48

Control Codes ..... 252

Conventions Used in this Manual ..... 3

Conversion ..... 131, 177, 188, 275

Copy a File ..... 30

Copying Text ..... 35

Counter rollover ..... 131

CR (Carriage Return) Translation ..... 78

Create a New File ..... 30

CTOU1 ..... 188

CTOU2 ..... 188

Current

Average ..... 218

Command ..... 219

Host Commands ..... 301

IAVE ..... 218

IAVG ..... 219

ICMD ..... 219

ILIMIT ..... 222



Limit ..... 222  
 Peak Current Command ..... 248  
 PICMD ..... 248  
 Current Configuration ..... 48  
 Cutting Text ..... 35

**D**

D, V ..... 189  
 D, V Segments (Sticks) ..... 156, 189  
 DAC1 ..... 306  
 DACn ..... 46, 73, 93, 191  
 Data Bits ..... 78  
 DBG ..... 297  
 DBNCE ..... 304  
 DDE Server ..... 323  
 Debounce Time ..... 44, 86  
 DEBUG ..... 191  
 Debug ..... 297  
 Debug Information ..... 37  
 Debug Options  
     Abort ..... 63  
     Clear All Breakpoints ..... 64  
     Go ..... 62  
     Pause ..... 63  
     Procedure Step ..... 63  
     Resume ..... 63  
     Single Step ..... 63  
     Toggle Breakpoint ..... 64  
     Trace ..... 63  
     View Active Breakpoints ..... 64  
 Debug Options Dialog ..... 64  
 Dedicated Flag ..... 134  
 Dedicated Inputs ..... 86  
 Dedicated Variable ..... 134  
 Default Run Program ..... 45  
 Define Home ..... 192  
 Define Home Command ..... 193  
 Define Home Fkey Program ..... 334  
 Define Position ..... 197, 198  
 DEL ..... 315  
 DELAY ..... 160, 192  
 Delay Out of Range ..... 419  
 Delete IQ Program ..... 31, 315  
 DELS ..... 315  
 DH ..... 57, 88, 168, 192, 412  
 DHCMD ..... 193  
 DHINP ..... 303

Diagnostics Menu  
 Digital Outputs ..... 74  
 Encoder ..... 73  
 Fault History ..... 75  
 Nonvolatile Memory ..... 75  
 Operator Terminal ..... 75  
 Programmable Monitor Output ..... 74  
 Version ..... 75

Diagnostics menu

DAC1 Output ..... 73  
 DIF Segments (Sticks) ..... 157  
 DIF segments (Sticks) ..... 193  
 Digital Outputs ..... 74  
 Digital to Analog Converter ..... 191  
 DIR ..... 315  
 Directory ..... 315  
 DIRI ..... 315  
 DISABLE ..... 46, 56, 169, 195, 296  
 Disable Fkey Program ..... 356  
 Disable on Fault ..... 41  
 DISERR ..... 306  
 DISLIM ..... 196  
 Displaying & Reading Data ..... 101  
 Distance Scale ..... 264  
 Distance, Conditional ..... 193  
 Divide by Zero ..... 419  
 DM ..... 310  
 DO/WHILE ..... 142, 196  
 DP ..... 57, 168, 197, 198  
 Drive Not Ready ..... 420  
 DWELL ..... 160, 199

**E**

EDGE ..... 147, 261  
 Edit an Existing File ..... 30  
 Edit Menu  
     Compile ..... 36  
     Compiler Output ..... 37  
     Copy ..... 35  
     Cut ..... 35  
     Debug Information ..... 37  
     Expand Macros ..... 37  
     Find ..... 36  
     Find Next ..... 36  
     Find Previous ..... 36  
     Paste ..... 35  
     Program Type ..... 37

Replace	36
Undo	35
EFLAG	199
ELSE	220
Emergency Return (I10)	43, 87
Emergency Return Position	202
Emergency Return System Program	353
ENA see ENABLE	
ENABLE	43, 56, 86, 169, 200, 296
Enable Amplifier	200
Enable Fkey Program	356
Enabled Output	93
Encoder	73
Encoder 1 Fault	421
Encoder 2	48, 93
Encoder 2 Fault	421
Encoder counter rollover	131
Encoder n index interrupt	207
END	200
End of program	142
ENUM	201, 310
EPOS	202, 308
EPOS see ERETPOS	
EPROM Checksum	421
ERET	201, 295
ERETPOS	40, 202
ERETURN	40, 43, 87
ERETURN Flag	201
ERETURN Position	40
ERLY	306
ERREN	306
ERROR (O8)	46, 92
Error Flag	199
Error Number	201
Error Output when Disabled	46
Error Program	354
Error Routine System Program	354
ESIZE	310
Excessive FE	420
Excessive Speed	420
Execute Home Fkey Program	333
Exit IQ Master	33
Expand Macros	37
EXPANDMACROS	202
Expansion Memory	105

**F**

F	308
Faults	98
Clearing Faults	99
His	311
History	75, 311
When a fault occurs	98
FDR	42, 204, 310
FE	205, 308
FE Fkey Program	339
Feedback Configuration	47, 168
Feedback Position Source	48
Feedrate	42, 93, 160, 204
Feedrate Fkey Program	335
Feedrate, Runtime	258
FEL	40, 205, 310
FET	206, 310
FGAIN	39, 206, 302
FI1	207, 305
FI2	207, 305
FIDX1	207, 305
FIDX2	207, 305
FIL	302
File Extensions	31
File Menu	
Delete IQ Program	31
List Files	37
New	30
Open	30
Open IQ Program	30
Print Program	32
Recently Used File List	33
Save	30
Save IQ Program	30
Send Mail	33
Transfer	32
File menu	
Exit	33
File Transfer Active	422
FILTEN	302
FILTER	39, 208, 302
Find Next	36
Find Previous	36
Finding Text	36
Firmware Version	422
Fkey	
Active Flag	208
FLIMIT	310

FLTDIS ..... 310  
 Fn ..... 203, 314  
 FNACTIVE ..... 208  
 FNPGM ..... 150, 209, 317  
 Following Error ..... 205  
   Limit ..... 40, 205  
   Peak ..... 246  
   Time ..... 40, 206  
 FREE ..... 315  
 FTXT ..... 310  
 Function Keys ..... 101  
 Function Program, Run ..... 209  
 FVEL1 ..... 210, 301  
 FVEL1 Fkey Program ..... 357  
 Fkey  
   Creating Programs ..... 150  
   FNPGM ..... 150  
   Instruction Summary ..... 151  
   Mode ..... 48  
   Print ..... 150  
   Routines ..... 150  
   Set Up ..... 48

**G**

Gains ..... 39  
 FGAIN ..... 206  
 FILTER ..... 208  
 IGAIN ..... 221  
 Integral Gain ..... 230  
 IZONE ..... 225  
 KFF ..... 230  
 KI ..... 230  
 KP ..... 231  
 KPZ ..... 232  
 PGAIN ..... 246  
 Proportional ..... 231  
 PZONE ..... 256  
 PZONE Proportional ..... 232  
 Velocity Feedforward ..... 230  
 GEAR ..... 158, 212, 306  
   Enable ..... 213  
   GEAREN ..... 213  
   Slew Rate Limit ..... 267  
 Gear  
   Input ..... 48  
 Gear Out Of Range ..... 419  
 Gear Position ..... 213

GEAREN ..... 213, 306, 412  
 General Purpose Inputs ..... 85  
 General Purpose Outputs ..... 91  
 Generic Operator Terminal ..... 50  
 Gn ..... 211, 314  
 GPOS ..... 213, 308  
 GTERM ..... 310

**H**

HACTIVE ..... 303  
 Hard Forward Limit ..... 421  
 Hard Reverse Limit ..... 421  
 Hardware Reset ..... 57, 89, 215  
 Hardware Reset Fkey Program ..... 357  
 Header, Program ..... 141  
 Heat Sink Overtemperature ..... 420  
 Help Menu  
   About IQ Master ..... 4, 81  
   Contents ..... 4, 81  
   Context Sensitive ..... 4, 81  
   How to Use Help ..... 81  
   Opening ..... 4  
 History ..... 311  
 HLIMITS ..... 214  
 HOFFS ..... 43, 214, 303  
 Home ..... 295  
   ATHOME ..... 182  
   Command (I6) ..... 43, 87  
   Complete (O6) ..... 46, 92  
   Define Home ..... 192  
   Define Position ..... 197, 198  
   DH ..... 192  
   DHCMD ..... 193  
   DP ..... 197, 198  
   HOFFS ..... 214  
   HOMECMD ..... 43, 87, 215  
   Host Commands ..... 303  
   HSEQCPL ..... 92, 215  
   HSWEN ..... 216  
   HSWSTAT ..... 216  
   HVEL ..... 217  
   INDEXEN ..... 222  
   Offset ..... 43, 214  
   Sequence Complete ..... 215  
   Switch (I5) ..... 43, 87  
   Switch Active State ..... 44  
   Switch Enable ..... 216

Switch Status .....	216	HTIME .....	311
System Program .....	344	HV .....	303
Velocity .....	43, 217	HVEL .....	43, 217, 303
Home Not Defined .....	419		
Home Sw Not Found .....	422	<b>I</b>	
Home to a Limit Switch Program .....	359	I/O Expansion .....	104
Home to Encoder Index Program .....	359	I1°I48 .....	304
Home Without Limit Switch Program .....	359	I1P1 .....	218, 305
HOMECMD .....	43, 87, 215	I1P2 .....	218, 305
Homing .....	57	I2P1 .....	218, 305
Homing to Encoder Index .....	44	I2P2 .....	218, 305
HOST .....	309	IAVE .....	218, 301
Host Command Language		IAVE Fkey Program .....	341
Acceleration .....	301	IAVG .....	40, 219, 301
Command Syntax .....	292	Iavg Fault .....	420
Current .....	301	IB1,2,3 .....	304
Directory Commands .....	315	ICMD .....	219, 301
Disable .....	296	ICMD Fkey Program .....	340
Enable .....	296	IDX1 .....	305
Gains .....	302	IDX2 .....	305
Homing .....	303	IDXn ON/OFF .....	220
Implementation .....	290	IF .....	144, 147, 220
Inputs – Analog and Digital .....	304	IF/ELSE .....	145, 220
Interrupts .....	305	IGAIN .....	39, 221, 302
Motion .....	306	ILIMIT .....	222, 301
Motion Execution Commands .....	295	Illegal Pgm Number .....	422
Non-Motion Execution Commands .....	296	Im .....	217
Operator Terminal .....	317	Incremental (MOVD) Motion .....	156
Outputs – Analog and Digital .....	306	Incremental Vs. Absolute .....	155
Overview .....	289	INDEX .....	303
Position .....	308	Index Enabled .....	222
Program Maintenance .....	315	Index Interrupt Enable/Disable .....	220
Response Syntax .....	293	Index Interrupts .....	151
Runtime Status and Version Commands .....	297	INDEXEN .....	222
Serial .....	309	Initialize PM .....	32
System .....	309	INPOSN .....	39, 223
Transfer Commands .....	316	Input Interrupts .....	151
Tune Mode .....	314	Inputs .....	43, 68, 85, 166
User Variables and Flags .....	314	ADCn .....	93, 179
Velocity .....	301	Assignable Inputs .....	86
Hot Keys .....	15, 17	BCD .....	183
How to Use Help .....	81	BIN .....	183
HRESET .....	215, 296	Binary Coded Decimal .....	183
Hreset Fkey Program .....	357	Debounce Time .....	44, 86
HRINP .....	303	Define Home .....	88
HSEQCPL .....	215	Emergency Return .....	87
HSEQEN .....	303	Emergency Return (I10) .....	43
HSWEN .....	216		
HSWSTAT .....	216		

Enable (I3) .....	43, 86	InPosition .....	92, 223
Expansion .....	104	Mode .....	46
General Purpose Inputs .....	85	Output (O7) .....	46
Hardware Reset .....	89	Window .....	39, 281
Home Command .....	87	IPEN .....	306
Home Command (I6) .....	43	IPREL .....	307
Home Switch .....	87	IQ Master Screen .....	19
Home Switch (I5) .....	43	IX1P1 .....	225, 305
Host Commands .....	304	IX1P2 .....	225, 305
In .....	217	IX2P1 .....	225, 305
Interrupt Inputs .....	90	IX2P2 .....	225, 305
Jog Forward (I7) .....	87	IXnPm .....	225
Jog Reverse (I8) .....	87	IZONE .....	39, 225, 302
Kill Motion .....	89		
Pause .....	44, 88	<b>J</b>	
Program Select Lines .....	45, 90	JACCEL .....	42, 226, 301
Selectable .....	44	JDECEL .....	43, 227, 301
Start (I4) .....	44, 88	JM .....	311
Travel Limit .....	43, 88	Jog	
X Kill Motion .....	89	Acceleration .....	42
Install New EPROMS .....	415	Deceleration .....	43
INT .....	223	Forward (I7) .....	43, 87
INT1, INT2 .....	305	Forward Flag .....	228
Integer .....	223	Inputs .....	43, 87, 99
Integral Gain, Velocity Loop .....	221	JOGF .....	100
Integrator Zone, Position Loop .....	225	JOGR .....	100
Interrupts .....	151	PJOG .....	70
Enable/Disable .....	220, 224	Reverse (I8) .....	43, 87
FI1 .....	207	Reverse Flag .....	228
FI2 .....	207	Velocity .....	42
FIDXn .....	207	Jog Forward Fkey Program .....	328
Hardware Latched Position .....	234	Jog Reverse Fkey Program .....	329
Hardware Position Latch .....	225	JOGACTIVE .....	227, 308
Host Command .....	305	JOGF .....	228, 311
IDXn .....	220	JOGR .....	228, 311
InPm .....	218	JUMP .....	146, 229
Inputs .....	90	JV .....	301
Instruction Summary .....	151	JVEL .....	42, 229, 301
INTn .....	224		
IXnPm .....	225	<b>K</b>	
LPOS .....	234	KFF .....	39, 230, 302
ON/OFF/CONT .....	224	KFF Out of Range .....	419
Position .....	218, 225	KI .....	39, 230, 302
Propagation Delay .....	91	KI Out of Range .....	419
Purpose .....	90	Kill Motion .....	89
Step by Step .....	151		
INTn ON/OFF/CONT .....	224		
Introduction .....	1		
Invalid Opcode .....	419		

KP	39, 231, 302
KP Out of Range	419
KPZ	39, 232, 302
KPZ Out of Range	419
KT	311

## L

Label	146
LABELS	232
Latched Position	41
Limits	
Forward Travel (I1)	43
Hard Forward Error (60)	421
Hard Reverse Error (61)	421
Reverse Travel(I2)	43
Soft Forward Error (62)	421
Soft Reverse Error (63)	421
List Files	37
LISTFILE	233
LOAD	316
LOAD ALL	316
LOAD MEM	316
LOAD PAR	316
LOADE	316
Local Echo	79
Logic Supply Fault	420
LOOP	233, 412
LOOP Count Out of Range	419
LOOP Variable Out of Range	419
LOOPINDEX	234
LP2EN	305
LPOS	41, 234, 305

## M

Main body, Program	142
Manual Tune	59, 60
Math	
@	177
Accuracy	128
AND	129
Conversion	177, 188, 275
CTOU1	188
CTOU2	188
NAND	130
NOR	130

NOT	130
OR	130
Order of Evaluation	129
Precedence	129
Resolution	128
UTOC1	275
UTOC2	275
XOR	130
Math Overflow	419
MD see MOVD	
MDAC	307
Memory, Expansion	105
Miscellaneous	
CF	187
CONFIG	187
DIS	195
DISABLE	195
EN	200
ENABLE	200
Mode key	48
Monitor Menu	
Inputs	68
Monitor Output	69
Outputs	68
Status	68
System Flags	69
System Variables	69
Monitor Output	69, 235
Monitor Variables	69
MONOUT	235, 307
Motion	154
Combination Motion	159
D, V Segments (Sticks)	156, 189
DELAY	160, 192
DIF	193
DWELL	160, 199
Feedrate	160
GEAR	158, 212
Gear Enable	213
GEAREN	213
Host Commands	306
Incremental Vs. Absolute	155
MOVD	236
MOVP	237
MOVV	238
S-Curve Acceleration	158
SLEWEN	269
Step and Direction	159
Stick Move	189

Stick Moves ..... 156  
 TRACKINGMODE ..... 274  
 Motor Overtemperature ..... 420  
 MOVD ..... 236, 295  
 Move a Distance ..... 236  
 Move at Velocity ..... 238  
 Move to Position ..... 237  
 MOVECOMPLETE ..... 237, 308  
 MOVP ..... 237, 295  
 MOVV ..... 238, 295  
 MP see MOVP  
 MV see MOVV  
 MVO ..... 307  
 MVO Output Enable ..... 239  
 MVOEN ..... 239  
 MXFGAIN ..... 302  
 MXFILTER ..... 302  
 MXIAVE ..... 301  
 MXIGAIN ..... 302  
 MXILIMIT ..... 301  
 MXKFF ..... 302  
 MXKI ..... 302  
 MXKP ..... 302  
 MXOVRSPD ..... 311  
 MXPGAIN ..... 302

**N**

Name Changes for Version 2 ..... 413  
 NAND ..... 130  
 New Features ..... 410  
 New File ..... 30  
 No Key ..... 102  
 NOECHO ..... 309  
 NOLF ..... 309  
 Nonvolatile ..... 75, 127  
 NOR ..... 130  
 NOT ..... 130  
 NPGMS ..... 311  
 Null ..... 296  
 NVPGS ..... 311  
 NVRAM Compatibility ..... 423  
 NVRAM R/W Fault ..... 421

**O**

OB1,2,3 ..... 307

OCHG1,2,3 ..... 307  
 ON ..... 145, 240  
 On ..... 307  
 Online Help ..... 4  
 Open an Existing File ..... 30  
 Operator Terminal ..... 25, 75, 167  
     CLEAR ..... 184  
     Displaying & Reading Data ..... 101  
     FNPGM ..... 317  
     Function Keys ..... 101  
     Host Commands ..... 317  
     Installation ..... 101  
     Key Codes ..... 320  
     Monitor ..... 241  
     OTADDR ..... 317  
     OTCLS ..... 317  
     OTCP ..... 317  
     OTFILL ..... 317  
     OTFLUSH ..... 318  
     OTGET ..... 318  
     OTLNn ..... 318  
     OTMn ..... 318  
     OTMON ..... 241, 318  
     OTPUT ..... 318  
     OTS ..... 318  
     OTTBL ..... 319  
     OTW ..... 319  
     OTX ..... 319  
     PRINT ..... 251, 253  
     READ ..... 256  
     StartUp ..... 25  
     Status Key ..... 102  
     XNPGM ..... 319  
     Xkeys ..... 102  
     Yes / No Keys ..... 102  
 Operator Terminal Address ..... 50  
 Operator Terminal Multi-Drop ..... 50  
 Operator Terminal, Generic ..... 50  
 Option Control Codes ..... 252  
 OR ..... 130  
 Order of Evaluation, Math ..... 129  
 OSET1,2,3 ..... 307  
 OTCLS ..... 317  
 OTCP ..... 317  
 OTFILL ..... 317  
 OTFLUSH ..... 318  
 OTGET ..... 318  
 OTLNn ..... 318  
 OTMn ..... 318

OTMON	241, 318
OTPUT	318
OTS	318
OTW	319
OTX	319
Output Defaults	47
Outputs	46, 68, 166
ALL	180
Assignable Outputs	92
ATHOME	46, 92
BCD	183
BIN	183
Binary	183
Binary Coded Decimal	183
DAC1 Value	93
DACn	46, 93, 191
DISABLE	46
Enabled	93
ERROR (O8)	46, 92
Expansion	104
General Purpose Outputs	91
Home Complete (O6)	46, 92
Host Commands	306
InPosition (O7)	46, 92
Om	239
Program Running (O4)	92
Program Running(O4)	46
Ready	93
OVERSPEED	43, 242, 311

## P

PACTIVE	304
Parameter Checksum	421
Parameter Menu	
Absolute Mode	40
Acceleration	42
ADC1 Function	44
ATHOME (O5)	46
Baud Rate	49
Configuration menu	48
Current Configuration	48
DAC1 Value	46
Debounce Time	44
Default Run Program	45
Disable on Fault	41
Edit Parameter File	51
Emergency Return (I10)	43

Enable (I3)	43
Encoder 2	48
ERETURN Position	40
Error (O8)	46
Error output when Disabled	46
F key Mode	48
F key Set Up	48
Feedback Configuration	47
Feedback Position Source	48
Feedrate	42
FGAIN	39
FILTER	39
Following Error Limit	40
Forward Software Travel Limit	40
Forward Travel Limit (I1)	43
Gains	39
Gear Input	48
Generic Operator Terminal	50
Home Command (I6)	43
Home Complete (O6)	46
Home Offset	43
Home Switch (I5)	43
Home Switch Active State	44
Home Velocity	43
Homing to Encoder	44
I AVG	40
IGAIN	39
In Position (O7)	46
In Position Mode	46
In Position Window	39
Inputs	43
IZONE	39
Jog Acceleration	42
Jog Deceleration	43
Jog Forward (I7)	43
Jog Reverse (I8)	43
Jog Velocity	42
KFF	39
KI	39
KP	39
KPZ	39
Latched Position	41
Operator Terminal Address	50
Operator Terminal Multi-Drop	50
Output Defaults	47
Outputs	46
OVERSPEED	43
Parity	49
Pause	44



Pause Switch Active State	44	PICMD Fkey Program	357
PGAIN	39	PJOG	249, 308
Program Running (O4)	46	Pos Track Overflow	420
Program Select Lines	45	POS1	249, 308
PZONE	39	POS1 Fkey Program	357
Reverse Software Travel Limit	40	POS2	250, 308
Reverse Travel Limit (I2)	43	POS2 Fkey Program	357
Rotation	41	POS3	250, 308
Rotation 2	41	Position	
RS 232C/RS 422	50	EPOS	202
SCALE	41	ERETPOS	202
SCALE2	41	FE	205
Selectable Inputs	44	Feedback Position	250
Serial	49	Following Error	205
Slew	42	Host Commands	308
Start	44	INPOSN	223
Step and Direction Input	48	PCAM Active	244
System	40	PCMD	244
TimeBase	42	Peak Following Error	246
Variable Defaults	51	PEXT	245
Velocities	42	PFE	246
Velocity	42	PGEN	247
Parity	49, 78	PJOG	249
Pasting Text	35	POS1	249
PAUSE	243, 311	POS2	250
Pause	44, 88, 243	POS3	250
Pause Fkey Program	358	POSN	250
Pause Switch Active State	44	Position Command	
PC Set Up	78	Cam	244
PCAM	244, 308	Cam Active	244
PCAMACTIVE	244, 311	Composite	154, 244
PCMD	244, 308	External	154, 245
PCMD Fkey Program	338	Jog	154, 249
Peak Overcurrent	420	PEXT Active	245
Personality Fault	421	Profile Generator	154, 247
Personality Module	134	Sn Active	179
Personality Module Initialization	32	Xn Active	179
PEXT	245, 308	Position Encoder 2	250
PEXTACTIVE	245	Position Encoder 3 (Optional)	250
PFE	246, 308	Position Feedback	250
PFE Fkey Program	357	Position Loop Tuning	164
PGAIN	39, 246, 302	Auto Tune	58
PGEN	247, 308	Host Commands	302
PGM	311	IZONE	39
PGMNUM	247	KFF	39
PGMRUNNING	247	KI	39
PGMSEL	304	KP	39
PGMTYPE	248	KPZ	39
PICMD	248, 301	Manual Tune	60

PZONE .....	39	Programmable Monitor Output .....	74
Position, Encoder 1 .....	249	Programming Structure	
Position, Latched .....	234	JUMP .....	146
Positive Rotation Encoder 1 .....	259	Proportional Gain .....	39
Positive Rotation Encoder 2 .....	260	Proportional Gain Zone, Velocity Loop ...	256
POSN .....	250, 308	Proportional Gain, Velocity Loop .....	246
POSN Fkey Program .....	337	PSELEN .....	311
PRECISION Out of Range .....	419	PURGEMOTION .....	312
PRINT .....	251, 412	PURGEMOTIONACTIVE .....	312
Print Program .....	32	PVEL1 .....	255, 301
PRINTWARN .....	253	PVEL1 Fkey Program .....	357
Profile Calculation .....	419	PWXTACTIVE .....	311
Program Directory .....	422	PZONE .....	39, 256, 302
Program File .....	422		
Program Header .....	141	<b>Q</b>	
Program Labels .....	232	QAP File Extension .....	31
Program Running .....	247	QPA File Extension .....	31
Program Running (O4) .....	46, 92	QPE File Extension .....	31
Program Select Lines .....	45, 90	QPM File Extension .....	31
Program Structure .....	141, 142	QPS File Extension .....	31
ASSIGN .....	181	Quitting IQ Master .....	19
CALL .....	184		
COMMENTS .....	185	<b>R</b>	
DO/WHILE .....	142, 196	RAM R/W Fault .....	421
END .....	200	RAMPD .....	307
FNPGM .....	209	RAMPM .....	307
IF .....	144, 220	Ratio .....	158
IF/ELSE .....	145	READ .....	256, 412
Instruction Summary .....	146	Ready Output .....	93
Jump .....	229	Receive File CR Translation .....	79
LABELS .....	232	Recently Used File List .....	33
LOOP .....	233, 258	Related Documentation .....	3
ON .....	145, 240	Rename a File .....	30
Purge Motion .....	254	REPEAT .....	258
RETURN .....	258	Replace .....	36
Run Fkey Program .....	209	Res .....	296
SUB .....	271	Reset .....	56, 296
Subroutine .....	271	Reset Fkey Program .....	358
Subroutines .....	143	Resolution, Math .....	128
TITLE .....	273	RET .....	258
WAIT .....	145, 279	RETURN .....	258
WHILE .....	143, 280	Return from Subroutine .....	258
XEND .....	283	RFDR .....	258, 312
Xn .....	282, 283	RFDR Fkey Program .....	342
Xn Enable/Disable .....	283	RLIMIT .....	312
XNPGM .....	285		
Program Title .....	141, 273		
Program Type .....	37, 248		
Program, Main Body .....	142		

Rollover	131	SAVE PAR	316
ROT	259, 312	SAVEE	316
ROT2	260, 312	SCALE	41, 264, 312
Rotation	41	SCALE2	41, 265, 312
Rotation 2	41	Scanned Event	261
ROTT	259, 306	IF statement	261
RPT	258	Scanned Events	147
RRLY	307	Enable	148, 260
RS232C/RS422	50	IF	147
RUN	296	Instruction Summary	148
Run Control	54	Sn ON/OFF/CONT	260
Run Fkey Program	332	Sn\ TMR	262, 263
Run Menu		TIMER	272
Abort	63	Timer	148
Auto	63	S-Curve Acceleration	158
Auto Tune	58	S-Curve Calculation	419
Clear All Breakpoints	64	SDEL	315
Debug Options	62	SDELS	315
Define Home Position	57	SDIR	315
Disable	56	SDIRI	315
Enable	56	Selectable Inputs	44
Go	62	Selecting Text	35
Hardware Reset	57	SEN	265, 312
Manual Tune	59	Send File LF Toggle	79
Modifying Debug Options	64	Send Mail	33
Pause	63	Serial Communications	49
Procedure Step	63	Baud Rate	49
Reset	56	COMn	309
Resume	63	Generic Operator Terminal	50
Run Control	54	Host Commands	309
Single Step	63	Operator Terminal Address	50
Soft Reset	56	Parity	49
Stop	56	RS232C/RS422	50
Toggle Breakpoint	64	SETTASKTIMES	266
Trace	63	SI, Interrupt Status	297
Tune	57, 67	SINT2	267
View Active Breakpoint	64	SLEW	42, 267, 306
RUNEN	307	Slew Enable	269
Running a Program	54	SLEWEN	306
		SLEWEN ON/OFF	269
<b>S</b>		SLOAD	316
S, Status	297	SLOADE	316
SAVE	316	Sn	
Save a File to Disk	30	IF	261
SAVE ALL	316	Sn ON/OFF/CONT	260
Save IQ Program	30	Sn\ TMRm	262, 263
SAVE MEM	316	Soft Enable Flag	265

Soft Forward Limit	43, 421	FNACTIVE	208
Soft Reset	56, 269	Following Error Limit	205
Soft Reverse Limit	43, 421	Following Error Time	206
Software Interrupt 2	267	Gear Position	213
SRESET	269	GPOS	213
SRUN	296	Hard Travel Limits	214
SS	312	Hardware Reset	215
SSAVE	316	HLIMITS	214
SSAVEE	316	Host Commands	309
Start Fkey Program	330	HRESET	215
Start Program (I4)	44, 88	JOGACTIVE	227
Start Program Flag	270	JOGF	228
Starting a Program	94	JOGR	228
Starting IQ Master	19	LOOPINDEX	234
STARTP	270	MONOUT	235
Start-Up Procedure		MOVECOMPLETE	237
IQ-2000	21	MVOEN	239
IQ-5000	22	OVERSPEED	242
IQ-550	24	PAUSE	243
StartUp Procedure	21	PGMNUM	247
Status Dialog	68	PGMRUNNING	247
Status Key	102	Program Number	247
STEP	296	RFDR	258
Step and Direction	48, 159	ROT	259
Stick Move	189	ROT2	260
Stick Moves	156	ROTT	259
STOP	56, 270	Runtime Feedrate	258
Stop Bits	78	SCALE	264
Stop Fkey Program	331	SCALE2	265
Stopping a Program	56, 96	SEN	265
SUB	271	SETTASKTIMES	266
Subroutines	142, 143, 271	SRESET	269
SUM	293	STARTP	270
SWINP	304	STOP	270
Syntax Changes for Version 2	412	SYSError	271
SYSError	271	SYSWARN	272
System	40	Timebase	272
ABORT	177	TRACKINGEXTPOSSELECT	273, 274
ABSMODE	178	Variable	69, 134
CLRPEAKS	185	WFLAG	279
DISLIM	196	WINDOW	281
EFLAG	199	WNUM	281
ENUM	201	XLOOPINDEX	284
ERET	201	XNACTIVE	284
Error Flag	199	System Program	325
FDR	204	Auto	327
FEL	205	Define Home	334
FET	206	Emergency Return	353
Flags	69, 134	Error Routine	354

Execute Home ..... 333  
 Feedrate ..... 335  
 Home ..... 344  
 Jog Forward ..... 328  
 Jog Reverse ..... 329  
 Monitor ADC1 ..... 343  
 Monitor FE ..... 339  
 Monitor IAVE ..... 341  
 Monitor ICMD ..... 340  
 Monitor PCMD ..... 338  
 Monitor POSN ..... 337  
 Monitor RFDR ..... 342  
 Monitor VEL1 ..... 336  
 Run ..... 332  
 Start ..... 330  
 Stop ..... 331  
 SYSWARN ..... 272

**T**

TBASE ..... 272, 312  
 TDIR ..... 314  
 Terminal ..... 77  
 TEXT Checksum ..... 421  
 Time Delay ..... 192  
 TimeBase ..... 42, 272  
 Timer ..... 148  
 TIMERn ..... 272  
 TIMERn = time ..... 272  
 TITLE ..... 273  
 Title, Program ..... 141  
 TMRn ..... 272  
 TMXDIST ..... 314  
 TMXVEL ..... 314  
 To Upload ALLPROG.QAP to the IQ ..... 415  
 TPER ..... 314  
 TRACKINGEXTPOSSELECT ..... 273, 306  
 TRACKINGMODE ..... 274, 306  
 TRACKINGSYNCED ..... 274, 306  
 Transfer ..... 32  
     Host Commands ..... 316  
 TRATIO ..... 314  
 Travel Limit ..... 43  
     Disable ..... 196  
     Enabled Flag ..... 214  
     Forward (I1) ..... 43, 88  
     Forward Software ..... 40  
     Reverse (I2) ..... 43, 88

Reverse Software ..... 40  
 TSCUR ..... 314  
 TSPOS ..... 314  
 TSTART ..... 314  
 TSTOP ..... 314  
 TSVEL ..... 314  
 TTLIM ..... 314  
 Tune Mode  
     Host Commands ..... 314  
 Tuning ..... 57, 62, 63, 64, 67  
     Auto ..... 58  
     Manual ..... 59  
 Tutorial ..... 109

**U**

Undo ..... 35  
 Uninitialized PM ..... 423  
 Upgrade Personality Module ..... 415  
 User Variables ..... 127  
 UTOC1 ..... 275  
 UTOC2 ..... 275

**V**

Variables ..... 51  
     Bn ..... 182  
     Characteristics ..... 128  
     Defaults ..... 51  
     Fn ..... 203  
     Gn ..... 211  
     Host Commands ..... 314  
     Nonvolatile User Flag ..... 182  
     Nonvolatile User Variable ..... 211  
     User Variables ..... 127  
     Vn ..... 276  
     Volatile User Flag ..... 203  
     Volatile User Variable ..... 276  
 VCMD ..... 277, 301  
 VCMD Fkey Program ..... 358  
 VEL ..... 277, 301  
 VEL1 ..... 278, 301  
 VEL1 Fkey Program ..... 336  
 VEL2 ..... 278, 301  
 VEL2 Fkey Program ..... 358  
 Velocity ..... 42, 277  
     Encoder 2 ..... 278

Filtered Motor Velocity	210
FVEL1	210
Home	217
Host Commands	301
Jog Velocity	229
JVEL	229
Motor	278
MOVV Velocity	229
Peak Velocity	255
PVEL1	255
VCMD	277
VEL	277
VEL1	278
VEL2	278
Velocity Calculation	419
Velocity Loop Filter	208
Velocity Loop Tuning	165
Auto Tune	58
FGAIN	39
FILTER	39
Host Commands	302
IGAIN	39
Manual Tune	59
PGAIN	39
VER	298
Version	75, 298
VFILT	301
Viewing Break Points Dialog	64
Vn	276, 314
Volatile	127
VSCALE	301

## W

WAIT	145, 279
Warning Flag	279
WARNING No Program Title	273
Warning Number	281
Watchdog Reset	421
Watchdog Test Fault	421
WFLAG	279
Where to Find Help	1
WHILE	143, 280
Who Should Use this Manual	1
WIN	312
WINDOW	281
WINT	312
WNUM	281, 310

## X

X Kill Motion	89, 296
XEND	283
XINP	304
Xkey	282
Routine Active	284
Routine End	283
Routine, Enable	283
XLOOPINDEX	284
Xn	282
Xn ON/OFF/CONT	283
XNACTIVE	284
XNPGM	149, 285, 319
XOR	130
XPLC	312
X-Routine Loop Index	284
Xkey	102
Enable	149
Instruction Summary	149
Routines	148
XNPGM	149

## Y

Yes / No Keys	102
Yes Key	102



[www.rockwellautomation.com](http://www.rockwellautomation.com)

---

**Power, Control and Information Solutions Headquarters**

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444

Europe/Middle East/Africa: Rockwell Automation, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640

Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1398-PM601A-EN-P — October 2000

Supersedes 0013-1020-004

0013-1088-001

© 2000 Rockwell International Corporation. Printed in USA.