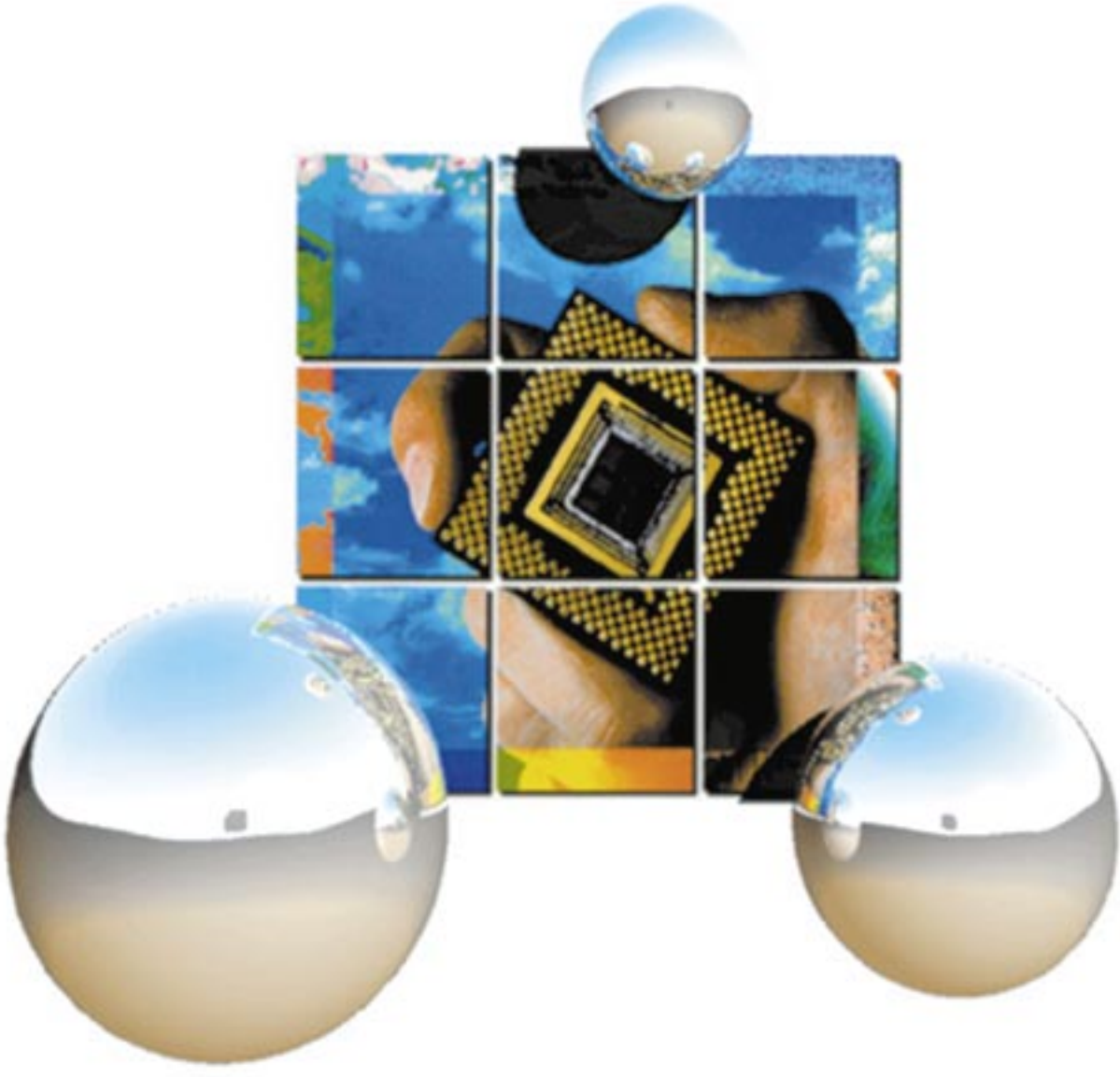
PCI Octavia

User's Guide v4.0



PCI Octavia

User's Guide

v4.0

© Copyright 2020 DSP Control Group, Inc.
4445 W 77th Street
Minneapolis, MN 55435
Phone: (952) 831-9556
FAX: (952) 831-4697

The authors and those involved in the manual's production have made every effort to provide accurate, useful information.

Use of this product in an electro mechanical system could result in a mechanical motion that could cause harm. DSP Control Group, Inc. is not responsible for any accident resulting from misuse of its products.

DSPL, Mx4 Octavia, Acc4, Vx4++, and Vx8++ are trademarks of DSP Control Group, Inc.

Other brand names and product names are trademarks of their respective holders.

DSPCG makes no warranty or condition, either expressed or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding the licensed materials.

Contents

How to Read This Manual

First, we would like to share with you the way this manual is organized, hoping this knowledge will help you find the information you need quickly.

We feel the first step in using an involved computerized product like Mx4 Octavia is to understand its definition and topology (the way it is connected to other subsystems and functions in a system). **Chapter 1** is dedicated to this task. This chapter contains simple block diagrams that will describe Mx4 Octavia's capabilities and functions. Please bear in mind that detailed information on Mx4 Octavia is provided in the following chapters, and in Chapter 1 we only describe this product qualitatively.

Once you have learned about the basic functions of Mx4 Octavia, you may want to test its strength in your system. **Chapter 2** provides you with information on hardware installation. In this chapter,you will find information on wiring Mx4 Octavia to your amplifier, I/O and encoder subsystems, etc.

Once you have installed your hardware and made sure that all communications, switches and jumpers are made and set correctly, you may move onto **Chapter 3.** Chapter 3 briefly describes the main features of the Mx4Pro v5.x development software. *Mx4Pro* v5.x is an easy-to-use Windows based program that has an integrated programming environment, oscilloscopes, function generator, variable viewer features that allows for system programming and tuning (useful for single and multi-axis applications). Mx4Pro v5.x is supported by its independent manual.

Beginning **Chapter 4** and beyond, information on Mx4 Octavia will become more technical and "lower level". You must deal with Mx4 Octavia at this level of detail only when you are ready to write your own customized application program. **Chapter 5** is dedicated to the description of Mx4 Octavia's low-level instruction set. For each instruction we describe its function, code, arguments, and a few applications that can benefit from its strength.

We dedicate **Chapter 6** to explaining Host-Mx4 Octavia communication. To understand the communication between the host computer and Mx4 Octavia, users are required to know about Mx4 Octavia's memory organization and software communication protocols. In describing Mx4 Octavia's memory organization, we have categorized the Mx4 Octavia commands into two major groups of real time commands (the commands that receive the DSP's immediate attention) and contouring commands (those that are executed based on the order they are stored in a special location of the dual port memory called the ring buffer). You will also learn how Mx4 Octavia reports back to the host (special dual port memory location dedicated to these parameters) and what situations cause Mx4 Octavia to interrupt the host.

In **Chapter 7** we describe the sources of errors, how Mx4 Octavia reports them to the host, how the user application program must handle them, and finally, possible ways they may be cured.

Chapter 8 describes Mx4 Octavia’s on-board 32k RAM organization as it pertains to CAM tables, compensation tables, internal cubic spline data, variable tables,and DSPL program storage.

Chapter 9 introduces NURBS-based programming with the Mx4 Octavia motion controller. This option allows the user to transfer true NURBS curve information directly to the controller, avoiding the geometric precision losses due to traditional conversion (such as G code or arc/line) techniques. This chapter includes an introduction to the components of NURBS curves, as well as detailed Mx4 Octavia programming information.

Finally, **Chapter 10** is devoted to Mx4 Octavia's specifications. Detailed electrical and mechanical specifications are listed. On the hardware side, we have included all bus specific information. Numerical values for Mx4 Octavia's parameters and variables have been listed in terms of their binary range. Parameters specifying performance such as sampling period and the maximum encoder speed Mx4 Octavia can handle have been listed under performance specifications.

In conjunction with this manual, the following manuals will assist you to develop and integrate Mx4 into your simple or complex machine. Depending on your application and system integration expertise, you may find none, one, or more of these supplementary manuals necessary.

Mx4Pro: Development Tools v5.x

This manual describes Mx4pro Development Tools, available for both Windows 2k and XP. Mx4pro includes features such as DSPL programming environment, live block diagrams, signal generator, time and xy oscilloscopes, system frequency analysis, and more.

DSPL Programmer's Guide

This manual assists you with high-level programming of Mx4, and Mx4 Octavia. The DSPL is DSPCG's high-level language that is supplied with its own compiler, linker, and downloader.

Mx4 & Windows

If your motion application resides on the Windows 2k or Windows XP operating system, you will be utilizing the Mx4 DLL. The Mx4 & Windows Manual accompanies the DLL, providing information for both Visual Basic and C/C++ programming. The Mx4 DLL includes functionality in all aspects of Mx4 Octavia use, including the complete real-time command set.

1 Introduction to Mx4 Octavia

Mx4 Octavia System Description

Mx4 Octavia is a fully digital high-performance five to eight-axis position controller. This multi-DSP based servo controller uses DSPs in a parallel processing configuration to close tighter, faster and more robust position and velocity loops. It also utilizes DSPCG's ASIC technology which provides exceptional hardware versatility and flexibility.

Mx4 Octavia outputs its control signals (ranging ± 10 volts) via eight 16-bit parallel DACs to any AC/DC industrial servo amplifier. It also incorporates 32 on-board inputs and 32 outputs for PLC applications. See Fig. 1-1.

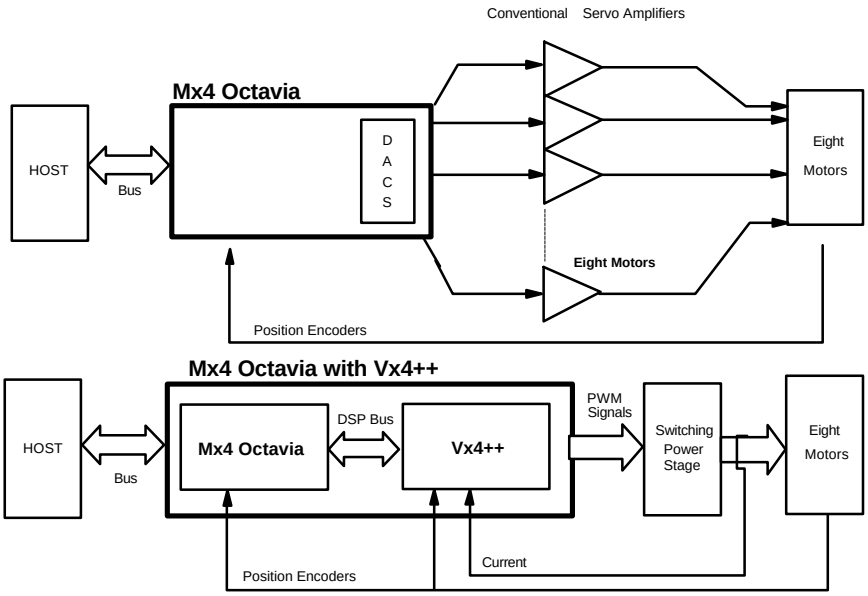


Fig. 1-1: *Top:* Mx4 Octavia with Conventional Servo Amplifiers, *Bottom:* Mx4 Octavia with the Vx4++ Drive Control Option

Mx4 Octavia Programming

Mx4 Octavia incorporates high (DSPL) and low (RTC) level programming. In addition, Mx4 Octavia supports contouring commands for complex control applications.

The DSPL is DSPCG's high-level PLC/motion control programming language that is supported by a compiler, linker, and downloader. The programs written in DSPL run independently of the host and use Mx4 Octavia's internal operating system (for more information see the *DSPL Programmer's Guide*).

The Real Time Commands, or RTCs, are issued by the host and executed by Mx4 Octavia immediately after they are transferred. Contouring commands are issued by the host in the form of transferring a number of widely spaced

position and velocity points to Mx4 Octavia. These commands are stacked up and executed by Mx4 Octavia sequentially.

Mx4 Octavia uses a Dual Port RAM (DPR) for communication with the host processor or computer. The DPR is partitioned into a large ring buffer for downloading host instructions to Mx4 Octavia and a number of register "windows" for bi-directional information transfer. All system states such as position and velocity are reported in real-time to the DPR for the host to read. In addition, Mx4 Octavia supports a debug feature that allows the host to interrogate internal Mx4 Octavia parameters through the DPR.

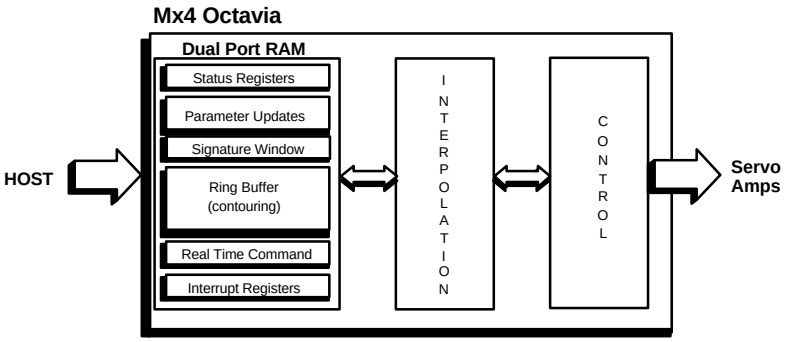


Fig. 1-2: Mx4 Octavia Internal Functional Block

Mx4 Octavia Control Law

The Mx4 Octavia incorporates a state feedback controller with dual feedback loops. A single DSP is dedicated to this task because the control law is important in control quality

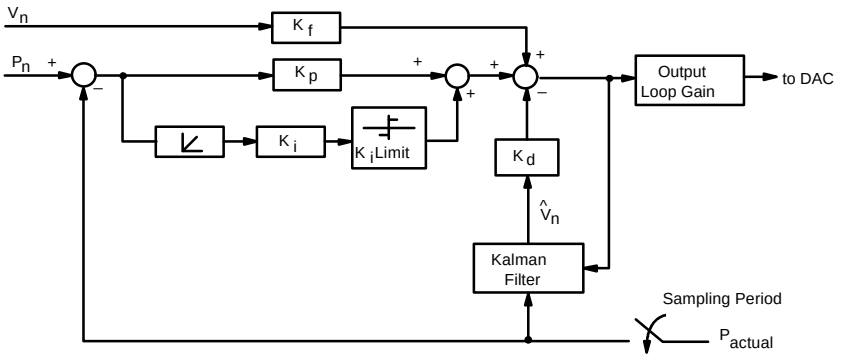


Fig. 1-3: Mx4 Octavia Position, Velocity Control Block Diagram

This control includes position and velocity loops. The actual system speed is estimated by the Kalman filter and fed back to regulate speed. The two states, position and speed, are constantly commanded by an interpolating algorithm and maintained by the control law. The control law provides robust operation for all industrial applications demanding up to a 5,000 Hz position loop bandwidth.

Why State Feedback?

The answer is simple: state feedback is easier to tune and provides a combination of control robustness with high bandwidth. Within this structure, optimum control algorithms such as LQG, dead beat, bang bang, etc. may be implemented.

The Kalman filter provides optimum estimation of speed and acceleration when environmental noise is present. The Kalman filter's output, velocity, provides the best feedback information at speeds with a low encoder pulse rate. The Kalman filter yields the best speed regulation at very low speeds.

In addition to state feedback control, an integration channel with anti-windup capability is provided to enable users to implement a traditional PID algorithm.

Notch Filter

Mx4 Octavia includes an optional notch filter with programmable notch frequency. This feature eliminates the mechanical resonance caused by an imperfect coupling between motor and load or other joint flexibility.

Coordination

Coordination of eight axes requires breaking four dimensional motion vectors down to the individual axis and interpolating the segment positions. Large position segments such as circular and elliptic arcs (for eight or more axes) are broken down to smaller position and velocity pieces. These segments are interpolated by the Mx4. This provides the tight coordination ideal for CNC, robotics, and other applications demanding high-speed precision control.

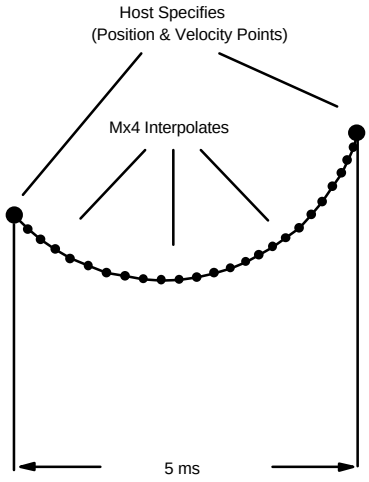


Fig. 1-4: Mx4 Octavia Interpolation

Cubic Spline Contouring

This interpolation provides a path between two user-specified position points that is smooth in velocity and continuous in acceleration. Cubic spline interpolation enhances contouring quality especially when the position points are widely spaced in time.

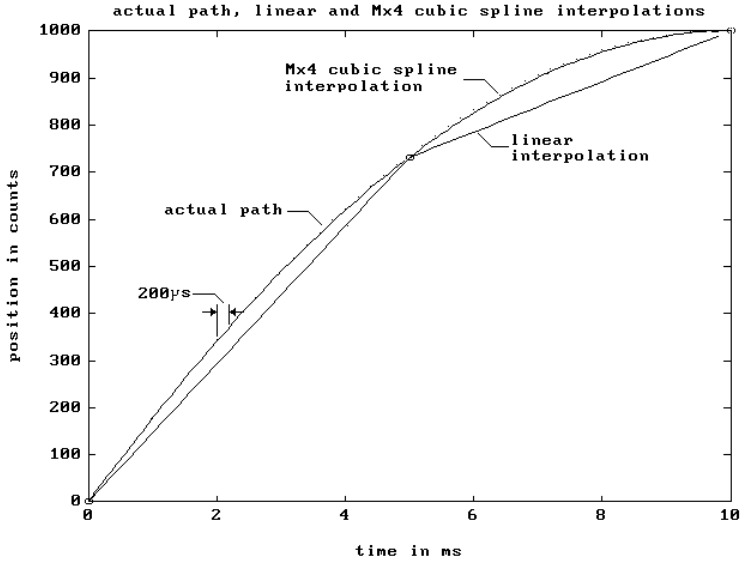


Fig. 1-5: Three User-Specified Pos./Vel. Points, Linear Interpolation and Mx4 Octavia Cubic Spline Interpolation

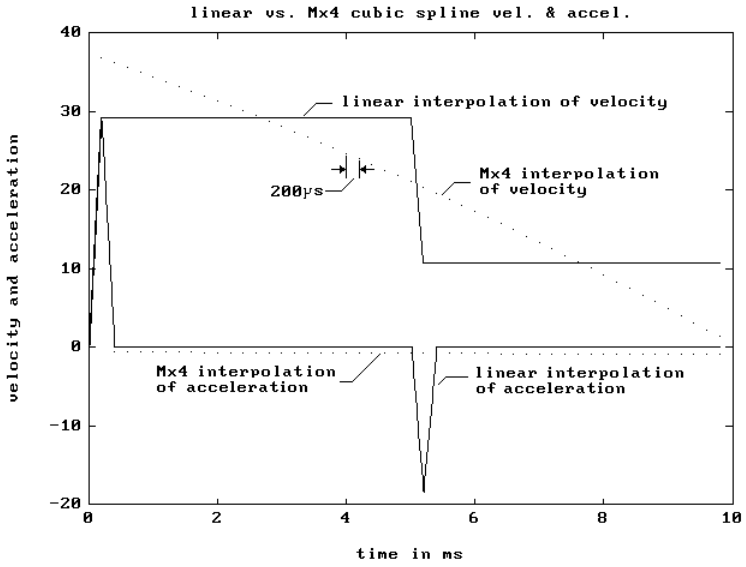


Fig. 1-6: Mx4 Octavia Cubic Spline Interpolations vs. Common Linear Interpolation

Cam and Gearing

Cam and electronic gearing are the two features of Mx4 Octavia that make it useful in master/slave applications. Both are initiated either by an external signal, an axis passing a programmed position, or unconditionally.

For applications requiring many axes (e.g., 20), several Mx4 Octavia cards may be daisy-chained and clock-synchronized. The Mx4 Octavia gearing and camming allow for multi-master operations.

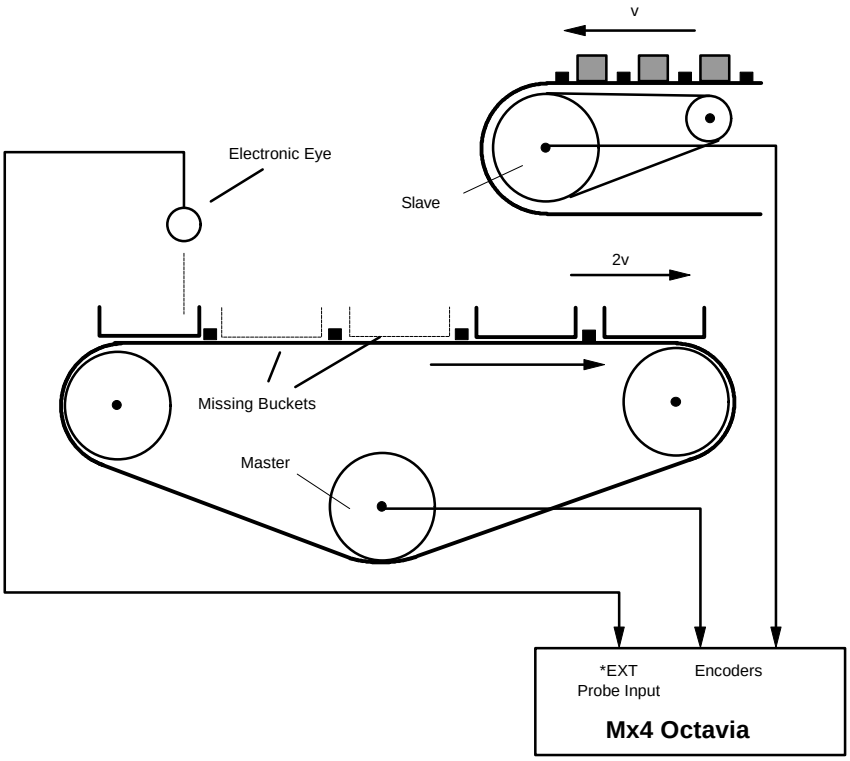


Fig. 1-7: External Signal Trigger Cam

Synchronization

In addition, Mx4 Octavia synchronizes several axes of control using high-speed (100 ns) position and event captures. In applications such as printing, packaging, indexing, paper handling, etc., the initial motion in several axes depends on the position of a master axis or a timing pulse provided by an external event. Proper timing for the execution of motion is crucial for synchronized applications. The Mx4 Octavia's ASICs contain 100 ns position and event captures designed especially for these applications.

Trajectory Generation & Interpolation

The Mx4 Octavia generates linear, circular, and contouring trajectories at high speeds. The tremendous computational power of DSPs arranged in a Hyper-Cube architecture allows real-time calculation of these trajectories. One DSP is dedicated to the task of command generation, while the other DSPs are dedicated to control tasks only.

Trajectory Compensation

The Mx4 Octavia controller provides a unique trajectory compensation that overrides machine non-linearities, backlash, and other inaccuracies to provide an accurate end result. Due to machine inaccuracies, mathematically correct trajectories do not generate mathematically correct motions. Machine inaccuracies influence motion quality, particularly at high speeds. For example, cutting a circle on a machine with inherent backlash will not produce a perfect circle. The Mx4 Octavia provides compensation for both position and velocity trajectories due to machine inaccuracies.

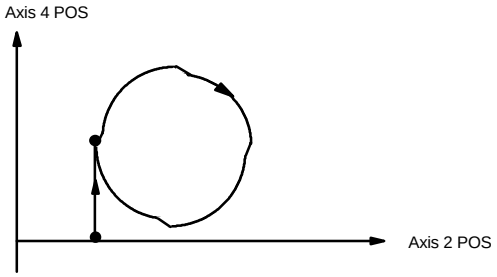


Fig. 1-8: Uncompensated Circle on an x-y Table

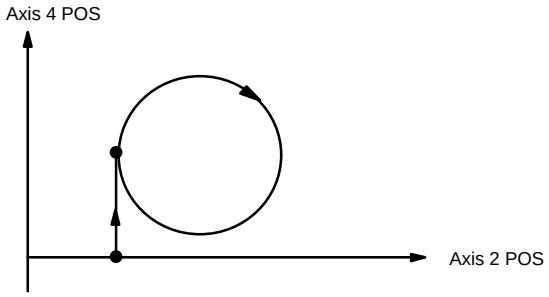


Fig. 1-9: Compensated Circle on an x-y Table

This page intentionally blank.

2 Installing Your Mx4 Octavia Hardware

A typical PCI Mx4 Octavia system consists of:

1. a PCI host computer
2. a Mx4 Octavia card occupying a slot on the host computer
3. one to eight motors with incremental position encoder(s)
4. one to eight servo amplifiers
5. cabling from Mx4 Octavia J6A, J6B connector to servo amplifier(s)
6. encoder feedback cabling to Mx4 Octavia J6A, J6B connector
7. optional cabling of external inputs to Mx4 Octavia (J1, J3 or J6A, J6B) connector
8. optional cabling of user inputs/outputs to Mx4 Octavia J1, J3 connector
9. optional synchronization cable between multiple Mx4 Octavia cards

When installing a Mx4 Octavia card, it is important to follow a procedure so that the card operates correctly in a given system. The installation guidelines detailed here incorporate three important topics: cabling to the Mx4 Octavia card, Mx4 Octavia jumper settings and bus-related Mx4 Octavia settings.

Note: If you are impatient to test the communication between your computer and the Mx4 Octavia card before completing the instructions of this chapter, you may do so by running Mx4pro software (see Chapter 3, *Mx4pro Development Tools*). The monitor will display the Tool Bar screen only if the Mx4 Octavia card and your computer are communicating.

When you are assured of this communication, come back to finish this chapter!

Fig. 2-1 is an illustration of a PCI Mx4 Octavia card that details connector positions and orientations. This figure will be used as a reference in the following pages.

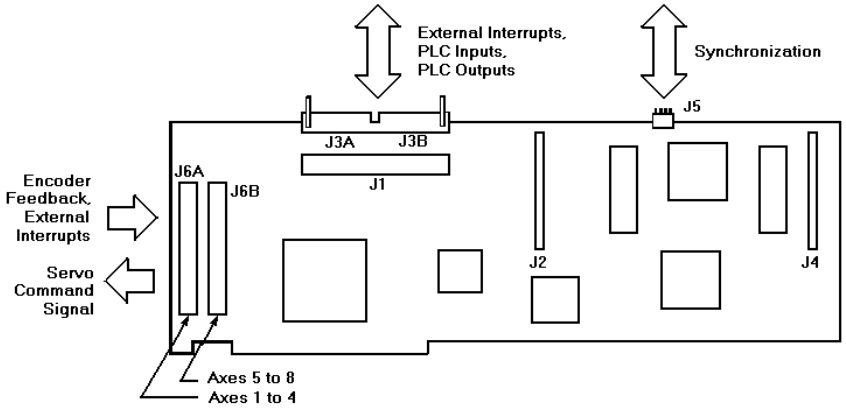


Fig. 2-1: PCI Mx4 Octavia Card Component Side

PCI Mx4 Octavia Cabling

The PCI Mx4 Octavia card contains five connectors as illustrated in Fig. 2-1. These connectors are used for interfacing the Mx4 Octavia card to the motors/system, optional user-defined inputs and outputs, and for the synchronization of multiple PCI Mx4 Octavia cards.

Before using a Mx4 Octavia card in a system application, a cable 'network(s)' must be built. The following sections provide a reference for designing and building PCI Mx4 Octavia cables.

PCI Mx4 Octavia J6A & J6B Connectors ... Motor/System Interfacing

The PCI Mx4 Octavia J6A, and J6B connectors are 50-pin dual row headers. These connectors include the motor and system interfacing signals for four axes each. The signals are divided into four categories: servo command signals, encoder feedback signals, general purpose external interrupt inputs and logic level signals.

Table 2-1 specifies the pinout for the J6A PCI Mx4 Octavia 50-pin header. The table includes signal level (type) and I/O functionality (with respect to the Mx4 Octavia card).

J6A Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
1	+12 volts	-	O	-
2	+5 volts	-	O	-
3	+12 volts	-	O	-
4	-12 volts	-	O	-
5	Digital GND	-	O	-
6	Analog GND	-	O	-
7	nc	-	-	no connection
8	*ESTOP	TTL	I	Mx4 Octavia emergency stop input
9	*EXT1	TTL	I	general purpose (probe) external interrupt
10	*EXT2	TTL	I	general purpose (probe) external interrupt
11	DAC(1)	-10 to +10v	O	DAC/motor output for axis 1
12	Analog GND	-	O	-
13	Digital GND	-	O	-
14	A+(1)	TTL	I	differential encoder signal A+ for axis 1
15	A-(1)	TTL	I	differential encoder signal A- for axis 1
16	B+(1)	TTL	I	differential encoder signal B+ for axis 1
17	B-(1)	TTL	I	differential encoder signal B- for axis 1
18	+5 volts	-	O	-
19	IP+(1)	TTL	I	differential encoder index pulse signal IP+ for axis 1
20	IP-(1)	TTL	I	differential encoder index pulse signal IP- for axis 1
21	DAC(2)	-10 to +10v	O	DAC/motor output for axis 2
22	Analog GND	-	O	-
23	Digital GND	-	O	-

Table 2-1: PCI Mx4 Octavia J6A Connector Pinout (continued on next page)

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
24	A+(2)	TTL	I	differential encoder signal A+ for axis 2
25	A-(2)	TTL	I	differential encoder signal A- for axis 2
26	B+(2)	TTL	I	differential encoder signal B+ for axis 2
27	B-(2)	TTL	I	differential encoder signal B- for axis 2
28	+5 volts	-	O	-
29	IP+(2)	TTL	I	differential encoder index pulse signal IP+ for axis 2
30	IP-(2)	TTL	I	differential encoder index pulse signal IP- for axis 2
31	DAC(3)	-10 to +10v	O	DAC/motor output for axis 3
32	Analog GND	-	O	-
33	Digital GND	-	O	-
34	A+(3)	TTL	I	differential encoder signal A+ for axis 3
35	A-(3)	TTL	I	differential encoder signal A- for axis 3
36	B+(3)	TTL	I	differential encoder signal B+ for axis 3
37	B-(3)	TTL	I	differential encoder signal B- for axis 3
38	+5 volts	-	O	-
39	IP+(3)	TTL	I	differential encoder index pulse signal IP+ for axis 3
40	IP-(3)	TTL	I	differential encoder index pulse signal IP- for axis 3
41	DAC(4)	-10 to +10v	O	DAC/motor output for axis 4
42	Analog GND	-	O	-
43	Digital GND	-	O	-
44	A+(4)	TTL	I	differential encoder signal A+ for axis 4
45	A-(4)	TTL	I	differential encoder signal A- for axis 4
46	B+(4)	TTL	I	differential encoder signal B+ for axis 4
47	B-(4)	TTL	I	differential encoder signal B- for axis 4
48	+5 volts	-	O	-
49	IP+(4)	TTL	I	differential encoder index pulse signal IP+ for axis 4
50	IP-(4)	TTL	I	differential encoder index pulse signal IP- for axis 4

Table 2-1 cont.: PCI Mx4 Octavia J6A Connector Pinout

Table 2-2 specifies the pinout for the J6B PCI Mx4 Octavia 50-pin header. The table includes signal level (type) and I/O functionality (with respect to the Mx4 Octavia card)

J6B Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
1	+12 volts	-	O	-
2	+5 volts	-	O	-
3	+12 volts	-	O	-
4	-12 volts	-	O	-
5	Digital GND	-	O	-
6	Analog GND	-	O	-
7	nc	-	-	no connection
8	*ESTOP	TTL	I	Mx4 Octavia emergency stop input
9	*EXT3	TTL	I	general purpose (probe) external interrupt
10	*EXT4	TTL	I	general purpose (probe) external interrupt
11	DAC(5)	-10 to +10v	O	DAC/motor output for axis 5
12	Analog GND	-	O	-
13	Digital GND	-	O	-
14	A+(5)	TTL	I	differential encoder signal A+ for axis 5
15	A-(5)	TTL	I	differential encoder signal A- for axis 5
16	B+(5)	TTL	I	differential encoder signal B+ for axis 5
17	B-(5)	TTL	I	differential encoder signal B- for axis 5
18	+5 volts	-	O	-
19	IP+(5)	TTL	I	differential encoder index pulse signal IP+ for axis 5
20	IP-(5)	TTL	I	differential encoder index pulse signal IP- for axis 5
21	DAC(6)	-10 to +10v	O	DAC/motor output for axis 6
22	Analog GND	-	O	-
23	Digital GND	-	O	-

Table 2-2: PCI Mx4 Octavia J6B Connector Pinout (continued on next page)

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
24	A+(6)	TTL	I	differential encoder signal A+ for axis 6
25	A-(6)	TTL	I	differential encoder signal A- for axis 6
26	B+(6)	TTL	I	differential encoder signal B+ for axis 6
27	B-(6)	TTL	I	differential encoder signal B- for axis 6
28	+5 volts	-	O	-
29	IP+(6)	TTL	I	differential encoder index pulse signal IP+ for axis 6
30	IP-(6)	TTL	I	differential encoder index pulse signal IP- for axis 6
31	DAC(7)	-10 to +10v	O	DAC/motor output for axis 7
32	Analog GND	-	O	-
33	Digital GND	-	O	-
34	A+(7)	TTL	I	differential encoder signal A+ for axis 7
35	A-(7)	TTL	I	differential encoder signal A- for axis 7
36	B+(7)	TTL	I	differential encoder signal B+ for axis 7
37	B-(7)	TTL	I	differential encoder signal B- for axis 7
38	+5 volts	-	O	-
39	IP+(7)	TTL	I	differential encoder index pulse signal IP+ for axis 7
40	IP-(7)	TTL	I	differential encoder index pulse signal IP- for axis 7
41	DAC(8)	-10 to +10v	O	DAC/motor output for axis 8
42	Analog GND	-	O	-
43	Digital GND	-	O	-
44	A+(8)	TTL	I	differential encoder signal A+ for axis 8
45	A-(8)	TTL	I	differential encoder signal A- for axis 8
46	B+(8)	TTL	I	differential encoder signal B+ for axis 8
47	B-(8)	TTL	I	differential encoder signal B- for axis 8
48	+5 volts	-	O	-
49	IP+(8)	TTL	I	differential encoder index pulse signal IP+ for axis 8
50	IP-(8)	TTL	I	differential encoder index pulse signal IP- for axis 8

Table 2-2 cont.: PCI Mx4 Octavia J6B Connector Pinout

Servo Command Signals

The servo command signals are those signals that 'drive' the axis servo amplifiers or output stage. The PCI Mx4 Octavia card utilizes 16-bit DAC outputs with +10v to -10v voltage swings to drive any voltage level sensitive output stage. The PCI Mx4 Octavia servo command signals are listed in Tables 2-3 and 2-4:

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
DAC(1)	11	-10 to +10v	O	DAC/motor output for axis 1
DAC(2)	21	-10 to +10v	O	DAC/motor output for axis 2
DAC(3)	31	-10 to +10v	O	DAC/motor output for axis 3
DAC(4)	41	-10 to +10v	O	DAC/motor output for axis 4
Analog GND	12	-	O	-
Analog GND	22	-	O	-
Analog GND	32	-	O	-
Analog GND	42	-	O	-

Table 2-4: PCI Mx4 Octavia J6A Servo Command Signals

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
DAC(5)	11	-10 to +10v	O	DAC/motor output for axis 5
DAC(6)	21	-10 to +10v	O	DAC/motor output for axis 6
DAC(7)	31	-10 to +10v	O	DAC/motor output for axis 7
DAC(8)	41	-10 to +10v	O	DAC/motor output for axis 8
Analog GND	12	-	O	-
Analog GND	22	-	O	-
Analog GND	32	-	O	-
Analog GND	42	-	O	-

Table 2-3: PCI Mx4 Octavia J6A Servo Command Signals

The DAC(x) signals must be routed from the J6A, J6B 50-pin header connectors to the respective output stage servo drives. The Mx4 Octavia Analog GND signals are included as a voltage reference for the DAC(x) signals. Analog GND should be utilized accordingly in the cabling between Mx4 Octavia and the output stages.

DAC(x) output offset voltage may be adjusted with the VRx multi-turn potentiometer (VR1 - DAC(1), VR2 - DAC(2), etc.) The Mx4 Octavia is shipped from the factory with minimized offset voltage.

Encoder Feedback

The Mx4 Octavia card requires the use of incremental position encoders for motor-Mx4 Octavia feedback. ***No*** velocity feedback (such as a tachometer) is required as Mx4 Octavia incorporates a Kalman velocity observer algorithm. The PCI Mx4 Octavia encoder feedback signals are listed in Table 2-5.

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
A+(1)	14	TTL	I	differential encoder signal A+ for axis 1
A-(1)	15	TTL	I	differential encoder signal A- for axis 1
B+(1)	16	TTL	I	differential encoder signal B+ for axis 1
B-(1)	17	TTL	I	differential encoder signal B- for axis 1
IP+(1)	19	TTL	I	differential encoder index pulse signal IP+ for axis 1
IP-(1)	20	TTL	I	differential encoder index pulse signal IP- for axis 1
A+(2)	24	TTL	I	differential encoder signal A+ for axis 2
A-(2)	25	TTL	I	differential encoder signal A- for axis 2
B+(2)	26	TTL	I	differential encoder signal B+ for axis 2
B-(2)	27	TTL	I	differential encoder signal B- for axis 2
IP+(2)	29	TTL	I	differential encoder index pulse signal IP+ for axis 2
IP-(2)	30	TTL	I	differential encoder index pulse signal IP- for axis 2
A+(3)	34	TTL	I	differential encoder signal A+ for axis 3
A-(3)	35	TTL	I	differential encoder signal A- for axis 3
B+(3)	36	TTL	I	differential encoder signal B+ for axis 3
B-(3)	37	TTL	I	differential encoder signal B- for axis 3
IP+(3)	39	TTL	I	differential encoder index pulse signal IP+ for axis 3
IP-(3)	40	TTL	I	differential encoder index pulse signal IP+ for axis 3
A+(4)	44	TTL	I	differential encoder signal A+ for axis 4
A-(4)	45	TTL	I	differential encoder signal A- for axis 4
B+(4)	46	TTL	I	differential encoder signal B+ for axis 4
B-(4)	47	TTL	I	differential encoder signal B- for axis 4
IP+(4)	49	TTL	I	differential encoder index pulse signal IP+ for axis 4
IP-(4)	50	TTL	I	differential encoder index pulse signal IP+ for axis 4
Digital GND	13	-	O	-
Digital GND	23	-	O	-
Digital GND	33	-	O	-
Digital GND	43	-	O	-

Table 2-5: PCI Mx4 Octavia J6A Encoder Feedback Signals

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
A+(5)	14	TTL	I	differential encoder signal A+ for axis 5
A-(5)	15	TTL	I	differential encoder signal A- for axis 5
B+(5)	16	TTL	I	differential encoder signal B+ for axis 5
B-(5)	17	TTL	I	differential encoder signal B- for axis 5
IP+(5)	19	TTL	I	differential encoder index pulse signal IP+ for axis 5
IP-(5)	20	TTL	I	differential encoder index pulse signal IP- for axis 5
A+(6)	24	TTL	I	differential encoder signal A+ for axis 6
A-(6)	25	TTL	I	differential encoder signal A- for axis 6
B+(6)	26	TTL	I	differential encoder signal B+ for axis 6
B-(6)	27	TTL	I	differential encoder signal B- for axis 6
IP+(6)	29	TTL	I	differential encoder index pulse signal IP+ for axis 6
IP-(6)	30	TTL	I	differential encoder index pulse signal IP- for axis 6
A+(7)	34	TTL	I	differential encoder signal A+ for axis 7
A-(7)	35	TTL	I	differential encoder signal A- for axis 7
B+(7)	36	TTL	I	differential encoder signal B+ for axis 7
B-(7)	37	TTL	I	differential encoder signal B- for axis 7
IP+(7)	39	TTL	I	differential encoder index pulse signal IP+ for axis 7
IP-(7)	40	TTL	I	differential encoder index pulse signal IP+ for axis 7
A+(8)	44	TTL	I	differential encoder signal A+ for axis 8
A-(8)	45	TTL	I	differential encoder signal A- for axis 8
B+(8)	46	TTL	I	differential encoder signal B+ for axis 8
B-(8)	47	TTL	I	differential encoder signal B- for axis 8
IP+(8)	49	TTL	I	differential encoder index pulse signal IP+ for axis 8
IP-(8)	50	TTL	I	differential encoder index pulse signal IP+ for axis 8
Digital GND	13	-	O	-
Digital GND	23	-	O	-
Digital GND	33	-	O	-
Digital GND	43	-	O	-

Table 2-6: PCI Mx4 Octavia J6A Encoder Feedback Signals

The Mx4 Octavia encoder feedback inputs are TTL-level inputs. The Mx4 Octavia Digital GND signals are included as voltage references for the differential inputs. The Digital GND signal(s) available on the J6 connector must be connected to the appropriate incremental encoder voltage reference points.

When interfacing incremental encoders to Mx4 Octavia, it is important that the following two conventions are followed:

1. Mx4 Octavia detects an active-HIGH index pulse. If the encoder(s) being interfaced to Mx4 Octavia include index pulse signals, it is important to note that the correct polarity is in effect. To reverse the polarity of an index pulse signal, simply 'swap' the IP+ and IP- signals to the Mx4 Octavia card.

2. The incremental encoder signals (A+, A-, B+, B-) should follow the convention of Fig. 2-2. That is, when the motor shaft is manually turned in the clockwise direction, a negative velocity should result.

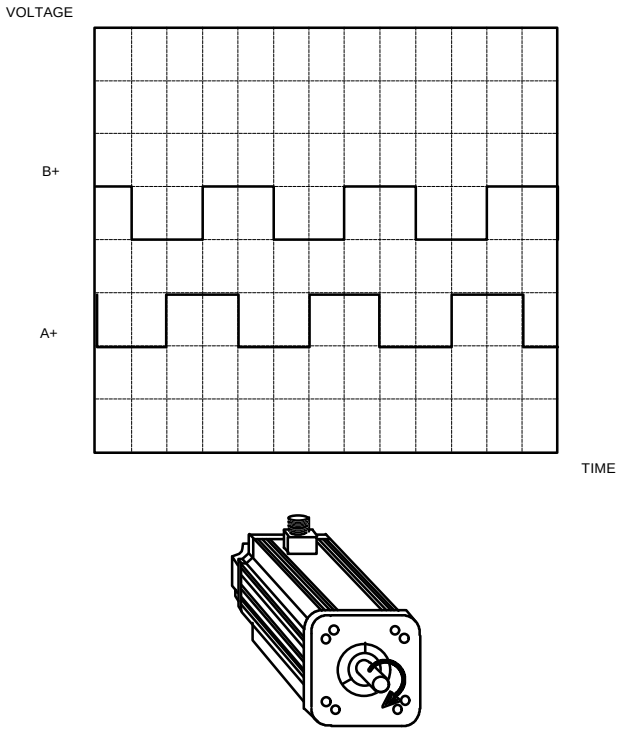


Fig. 2-2: Mx4 Octavia Incremental Encoder Signals Polarity ... Clockwise Shaft Rotation Yields Negative Velocity

If the use of an oscilloscope is not convenient, the encoder signal polarity may be verified later. In a following section, *Verifying Your Mx4 Octavia Hardware Set-Up*, the encoder signal polarity is checked via software. The check is simple and does not require the use of an oscilloscope. If the incremental encoder signal polarity is incorrect, it may be reversed simply by 'swapping' the A and B encoder signals.

General Purpose External Interrupt Inputs

The PCI Mx4 Octavia external interrupt inputs include an 'Emergency Stop' line and two general purpose external interrupts. The J6A, J6B connectors external interrupt inputs are repeated on the J1, J3 connectors. If the user is utilizing these signals, the signals may be accessed from either the J6A, J6B or J1, J3 connectors, but not from both. The external interrupt inputs are listed in Table 2-7 and 2-8:

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
*ESTOP	8	TTL	I	Mx4 Octavia emergency stop
*EXT1	9	TTL	I	general purpose (probe) external interrupt
*EXT2	10	TTL	I	general purpose (probe) external interrupt
Digital GND	5	-	O	-

Table 2-7: PCI Mx4 Octavia J6A External Inputs

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
*ESTOP	8	TTL	I	Mx4 Octavia emergency stop
*EXT3	9	TTL	I	general purpose (probe) external interrupt
*EXT4	10	TTL	I	general purpose (probe) external interrupt
Digital GND	5	-	O	-

Table 2-8: PCI Mx4 Octavia J6A External Inputs

These signals are optional for Mx4 Octavia operation. The definitions of these signals will be presented in later sections of this manual.

The *ESTOP and *EXTx signals are active-LOW signals. That is, Mx4 Octavia detects these active conditions when the voltage level on those lines is LOW. The Mx4 Octavia Digital GND signal is included as a voltage reference for the external input signals.

Logic Level Voltages / GND Signals

The PCI Mx4 Octavia card includes in its connector pin-out the following logic level / GND signals (Table 2-9) [in addition to the previously mentioned logic signals included with different signal groups]:

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
+12 volts	1	-	O	-
+12 volts	3	-	O	-
+5 volts	2	-	O	-
-12 volts	4	-	O	-
Analog GND	6	-	O	-
Digital GND	5	-	O	-

Table 2-9: PCI Mx4 Octavia J6A Logic Level / GND Signals

Partial J6A Connector Pinout

SIGNAL	PIN	LEVEL	I/O	DESCRIPTION
+12 volts	1	-	O	-
+12 volts	3	-	O	-
+5 volts	2	-	O	-
-12 volts	4	-	O	-
Analog GND	6	-	O	-
Digital GND	5	-	O	-

Table 2-10: PCI Mx4 Octavia J6A Logic Level / GND Signals

PCI Mx4 Octavia J1, J3 Connectors ... Inputs/Outputs

The PCI Mx4 Octavia includes J1 (50-pin dual row header) and J3AB (50-pin dual row header) connectors for input/output. These connectors include the Mx4 Octavia input and output signals as well as a repeat of the J6A and J6B general purpose external interrupt inputs.

Tables 2-11 and 2-12 specify the pinout for the J1 and J3AB PCI Mx4 Octavia connectors. The tables include signal level (type) and I/O functionality (with respect to the Mx4 Octavia card).

J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
1	Nc	-	-	no connection
2	IN9	TTL		general purpose input
3	Nc	-	-	no connection
4	IN14	TTL	I	general purpose input
5	Nc	-	-	no connection
6	IN17	TTL	I	general purpose input
7	OUT9	TTL	O	general purpose output
8	IN10	TTL	I	general purpose input
9	OUT7	TTL	O	general purpose output
10	OUT10	TTL	O	general purpose output
11	Nc	-	-	no connection
12	IN16	TTL	I	general purpose input
13	OUT1	TTL	O	general purpose output
14	OUT12	TTL	O	general purpose output
15	IN18	TTL	I	general purpose input
16	IN15	TTL	I	general purpose input
17	OUT3	TTL	O	general purpose output
18	Nc	-	-	no connection
19	OUT11	TTL	O	general purpose output
20	IN12	TTL	I	general purpose input
21	IN11	TTL	I	general purpose input
22	IN13	TTL	I	general purpose input
23	OUT2	TTL	O	general purpose output
24	IN19	TTL	I	general purpose input

Table 2-11: PCI Mx4 Octavia J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
25	IN8	TTL	I	general purpose input
26	OUT0	TTL	O	general purpose output
27	Nc	-	-	no connection
28	OUT8	TTL	O	general purpose output
29	-12 volts	-	O	-
30	Digital GND	-	O	-
31	+12 volts	-	O	-
32	+5 volts	-	O	-
33	Analog GND	-	O	-
34	Digital GND	-	O	-
35	OUT4	TTL	O	general purpose output
36	IN7	TTL	I	general purpose input
37	OUT5	TTL	O	general purpose output
38	IN3	TTL	I	general purpose input
39	OUT6	TTL	O	general purpose output
40	IN6	TTL	I	general purpose input
41	IN21	TTL	I	general purpose input
42	IN2	TTL	I	general purpose input
43	IN20	TTL	I	general purpose input
44	IN5	TTL	I	general purpose input
45	*ESTOP	TTL	I	Mx4 Octavia emergency stop
46	IN1	TTL	I	general purpose input
47	*EXT2	TTL	I	general purpose external (probe) interrupt
48	IN4	TTL	I	general purpose input
49	*EXT1	TTL	I	general purpose external (probe) interrupt
50	IN0	TTL	I	general purpose input

Table 2-11 cont.: PCI Mx4 Octavia J1 Connector Pinout

J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
1	OUT13	TTL	O	general purpose output
2	IN24	TTL	I	general purpose input
3	OUT14	TTL	O	general purpose output
4	IN25	TTL	I	general purpose input
5	OUT15	TTL	O	general purpose output
6	IN26	TTL	I	general purpose input
7	IN22	TTL	I	general purpose input
8	IN27	TTL	I	general purpose input
9	IN23	TTL	I	general purpose input
10	IN28	TTL	I	general purpose input
11	*ESTOP	TTL	I	Mx4 Octavia emergency stop
12	IN29	TTL	I	general purpose input
13	*EXT2	TTL	I	general purpose external (probe) interrupt
14	IN30	TTL	I	general purpose input
15	*EXT1	TTL	I	general purpose external (probe) interrupt
16	IN31	TTL	I	general purpose input
17	OUT19	TTL	O	general purpose output
18	OUT20	TTL	O	general purpose output
19	OUT21	TTL	O	general purpose output
20	OUT22	TTL	O	general purpose output
21	OUT23	TTL	O	general purpose output
22	OUT24	TTL	O	general purpose output
23	OUT25	TTL	O	general purpose output
24	OUT26	TTL	O	general purpose output
25	OUT27	TTL	O	general purpose output
26	OUT28	TTL	O	general purpose output
27	OUT29	TTL	O	general purpose output
28	OUT30	TTL	O	general purpose output
29	OUT31	TTL	O	general purpose output
30	+12v	-	O	-
31	+5v	-	O	-
32	+5v	-	O	-
33	Digital GND	-	O	-
34	Digital GND	-	O	-
35	OUT16	TTL	O	general purpose output

Table 2-12: PCI Mx4 Octavia J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
36	IN11	TTL	I	general purpose input

37	OUT17	TTL	O	general purpose output
38	IN12	TTL	I	general purpose input
39	OUT18	TTL	O	general purpose output
40	IN13	TTL	I	general purpose input
41	IN9	TTL	I	general purpose input
42	IN14	TTL	I	general purpose input
43	IN10	TTL	I	general purpose input
44	IN15	TTL	I	general purpose input
45	*ESTOP	TTL	I	Mx4 Octavia emergency stop
46	IN16	TTL	I	general purpose input
47	*EXT4	TTL	I	general purpose external (probe) interrupt
48	IN17	TTL	I	general purpose input
49	*EXT3	TTL	I	general purpose external (probe) interrupt
50	IN18	TTL	I	general purpose input

Table 2-12 cont.: PCI Mx4 Octavia J3AB Connector Pinout

Inputs

Mx4 Octavia includes 32 user-defined TTL logic inputs. The input signals are listed in Table 2-13 and 2-14.

Partial J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
2	IN9	TTL	I	General purpose input
4	IN14	TTL	I	General purpose input
6	IN17	TTL	I	General purpose input
8	IN10	TTL	I	General purpose input
12	IN16	TTL	I	General purpose input
15	IN18	TTL	I	General purpose input
16	IN15	TTL	I	General purpose input
20	IN12	TTL	I	General purpose input
21	IN11	TTL	I	General purpose input
22	IN13	TTL	I	General purpose input
24	IN19	TTL	I	General purpose input
25	IN8	TTL	I	General purpose input

Table 2-13: PCI Mx4 Octavia J1 Input Signals

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
36	IN7	TTL	I	General purpose input
38	IN3	TTL	I	general purpose input
40	IN6	TTL	I	general purpose input

41	IN21	TTL	I	general purpose input
42	IN2	TTL	I	general purpose input
43	IN20	TTL	I	general purpose input
44	IN5	TTL	I	general purpose input
46	IN1	TTL	I	general purpose input
48	IN4	TTL	I	general purpose input
50	IN0	TTL	I	general purpose input

Table 2-13 (cont.): PCI Mx4 Octavia J1 Input Signals

Partial J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
2	IN24	TTL	I	general purpose input
4	IN25	TTL	I	general purpose input
6	IN26	TTL	I	general purpose input
7	IN22	TTL	I	general purpose input
8	IN27	TTL	I	general purpose input
9	IN23	TTL	I	general purpose input
10	IN28	TTL	I	general purpose input
12	IN29	TTL	I	general purpose input
14	IN30	TTL	I	general purpose input
16	IN31	TTL	I	general purpose input

Table 2-14: PCI Mx4 Octavia J3AB Input Signals

The Mx4 Octavia user-defined input signals are TTL logic level inputs. The inputs are equipped with pull-up resistors which are implemented as current sources (see Fig. 2-3).

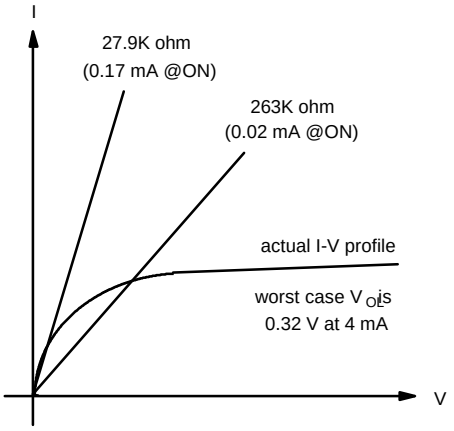


Fig. 2-3: Mx4 Octavia Input (Pull-Up Resistor) Current Source

By default, the inputs are defined as active-LOW. That is, 0v applied to an input results in an active, or ON, input; +5v applied to an input results in an inactive, or OFF input. The inputs may be individually configured as active-high or active-low via the INP_STATE command.

Fig. 2-4 illustrates two possible configurations for interfacing external input circuitry to Mx4 Octavia inputs: optically isolated input and same ground input.

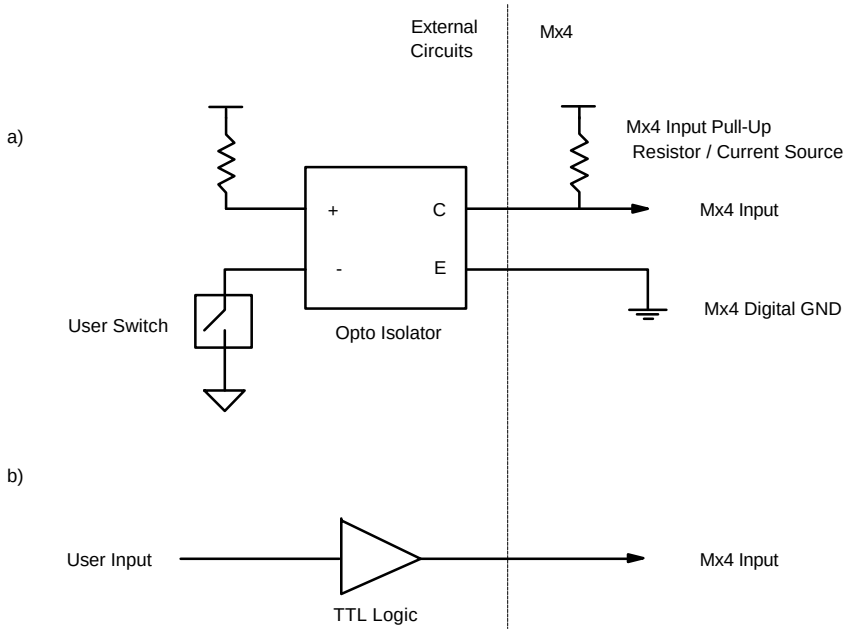


Fig. 2-4: Interfacing Input Signals to Mx4 Octavia
a) Optical Isolated Input
b) Same-Ground Input

Outputs

The PCI Mx4 Octavia controller includes 32 programmable outputs. The output signals are listed in Table 2-14 and Table 2-15.

Partial J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
7	OUT9	TTL	O	general purpose output
9	OUT7	TTL	O	general purpose output
10	OUT10	TTL	O	general purpose output
13	OUT1	TTL	O	general purpose output
14	OUT12	TTL	O	general purpose output
17	OUT3	TTL	O	general purpose output
19	OUT11	TTL	O	general purpose output
23	OUT2	TTL	O	general purpose output
26	OUT0	TTL	O	general purpose output
28	OUT8	TTL	O	general purpose output
35	OUT4	TTL	O	general purpose output
37	OUT5	TTL	O	general purpose output
39	OUT6	TTL	O	general purpose output

Table 2-14: PCI Mx4 Octavia J1 Output Signals

Partial J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
1	OUT13	TTL	O	general purpose output
3	OUT14	TTL	O	general purpose output
5	OUT15	TTL	O	general purpose output
35	OUT16	TTL	O	general purpose output
37	OUT17	TTL	O	general purpose output
39	OUT18	TTL	O	general purpose output
17	OUT19	TTL	O	general purpose output
18	OUT20	TTL	O	general purpose output
19	OUT21	TTL	O	general purpose output
20	OUT22	TTL	O	general purpose output
21	OUT23	TTL	O	general purpose output
22	OUT24	TTL	O	general purpose output

Table 2-15: PCI Mx4 Octavia J3AB Output Signals (continued on next page)

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
23	OUT25	TTL	O	general purpose output

24	OUT26	TTL	O	general purpose output
25	OUT27	TTL	O	general purpose output
26	OUT28	TTL	O	general purpose output
27	OUT29	TTL	O	general purpose output
28	OUT30	TTL	O	general purpose output
29	OUT31	TTL	O	general purpose output

Table 2-15 cont.: PCI Mx4 Octavia J3AB Output Signals

The Mx4 Octavia output signals are TTL logic level outputs with a fan out of one (that is, a Mx4 Octavia output should not be used to drive more than one TTL logic gate). The Mx4 Octavia provides 8mA sink current per output. As an example of interfacing to the Mx4 Octavia output signals, Fig. 2-5 illustrates a relay output circuit.

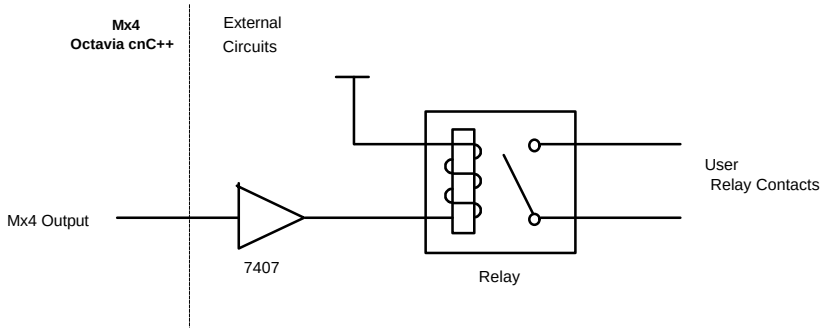


Fig. 2-5: Interfacing a Relay to a Mx4 Octavia Output

The Mx4 Octavia outputs are active-LOW. That is, an ON output is an output at 0v, an OFF output is an output at +5v. The ON/OFF state of the outputs is determined by the OUTP_ON and OUTP_OFF commands.

General Purpose External Interrupt Inputs

The J1 and J3AB connectors include five external interrupt inputs which are repeated on the J6A and J6B connectors. If the user is utilizing signals *EXT1 or *EXT2, these signals may be accessed from any one of the J1, J3AB, or J6A connectors, but not from more than one. Similarly, if the user is utilizing signals *EXT3 or *EXT4, these signals may be accessed from either the J3AB or J6B connector, but not from both. *ESTOP can be accessed through any one of the J1, J3AB, J6A, or J6B connectors, but not from more than one. The PCI Mx4 Octavia external interrupt inputs include an Emergency Stop line and two general purpose external interrupts. The external interrupt inputs are listed in Table 2-9:

Partial J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
45	*ESTOP	TTL	I	Mx4 Octavia emergency stop
49	*EXT1	TTL	I	general purpose (probe) external interrupt
47	*EXT2	TTL	I	general purpose (probe) external interrupt

Table 2-16: PCI Mx4 Octavia J1 External Inputs

Partial J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
11	*ESTOP	TTL	I	Mx4 Octavia emergency stop
45	*ESTOP	TTL	I	Mx4 Octavia emergency stop
15	*EXT1	TTL	I	general purpose (probe) external interrupt
13	*EXT2	TTL	I	general purpose (probe) external interrupt
49	*EXT3	TTL	I	general purpose (probe) external interrupt
47	*EXT4	TTL	I	general purpose (probe) external interrupt

Table 2-17: PCI Mx4 Octavia J3AB External Inputs

These signals are optional for Mx4 Octavia operation. The definitions of these signals will be presented in later sections of this manual.

The *ESTOP and *EXTx signals are active-LOW signals. That is, Mx4 Octavia detects these active conditions when the voltage level on those lines is LOW. The Mx4 Octavia Digital GND signal is included as a voltage reference for the external input signals.

Logic Level Voltages / GND Signals

The PCI Mx4 Octavia card includes in its J1, J3AB connector pinout the following logic level / GND signals (Table 2-18 and 2-19). These logic level voltage signals may be used by external circuits.

Partial J1 Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
31	+12 volts	-	O	-
32	+5 volts	-	O	-
29	-12 volts	-	O	-
33	Analog GND	-	O	-
34	Digital GND	-	O	-
30	Digital GND	-	O	-

Table 2-18: PCI Mx4 Octavia J1 Logic Level / GND Signals

Partial J3AB Connector Pinout

PIN	SIGNAL	LEVEL	I/O	DESCRIPTION
30	+12 volts	-	O	-
31	+5 volts	-	O	-
32	+5 volts	-	O	-
33	Digital GND	-	O	-
34	Digital GND	-	O	-

Table 2-19: PCI Mx4 Octavia J3AB Logic Level / GND Signals

PCI Mx4 Octavia J5 Connector ... Synchronization

Multiple PCI Mx4 Octavia cards may be time-synchronized to the same DSP cycle with the J5 synchronization connector. This Mx4 Octavia feature allows multi-axis systems which require more than eight axes to be synchronized. The Mx4 Octavia synchronization signals are listed in Table 2-20.

J5 Connector Pinout

	SIGNAL			DESCRIPTION
1	NC	-	-	no connection
2	SLAVE	TTL	I	slave Mx4 Octavia synchronization input
3	SLAVE	TTL	I	slave Mx4 Octavia synchronization input
4	MASTER	TTL	O	master Mx4 Octavia synchronization output

Table 2-20: PCI Mx4 Octavia J5 Connector Pinout

Synchronizing multiple Mx4 Octavia cards only requires cabling between the MASTER J5 signal from the "master" Mx4 Octavia to a SLAVE J5 signal (either pin 2 or pin 3) on the "slave" card(s).

The Mx4 Octavia J5 connector includes dual SLAVE signals in order to simplify "daisy chaining" between multiple Mx4 Octavia controllers.

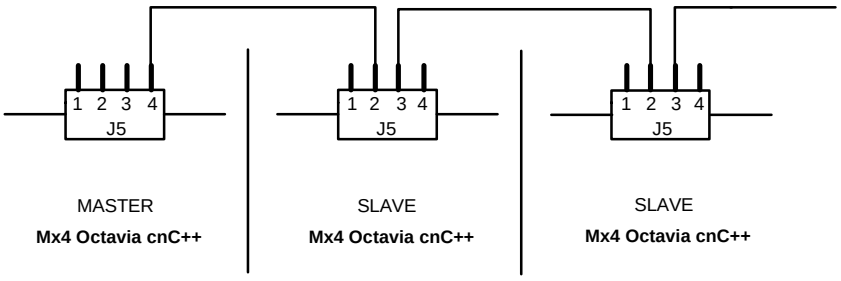


Fig. 2-6: Mx4 Octavia J5 Connector Daisy Chaining Cabling

Verifying the Mx4 Octavia Hardware Set-Up

The Mx4 Octavia hardware set-up may be verified via the Mx4pro Development Tools software (see Chapter 3, *Mx4pro Development Tools v5.x*) or alternatively through the use of any user-developed application software. If an error in the installation of the Mx4 Octavia card becomes evident from testing, it is advised to consult the respective section in this chapter.

3 Mx4pro Development Tools v5.x

This chapter briefly overviews the features of **Mx4pro v5** development tools software. For detailed information on these features, please refer to the *Mx4Pro Development Tools v5* manual.

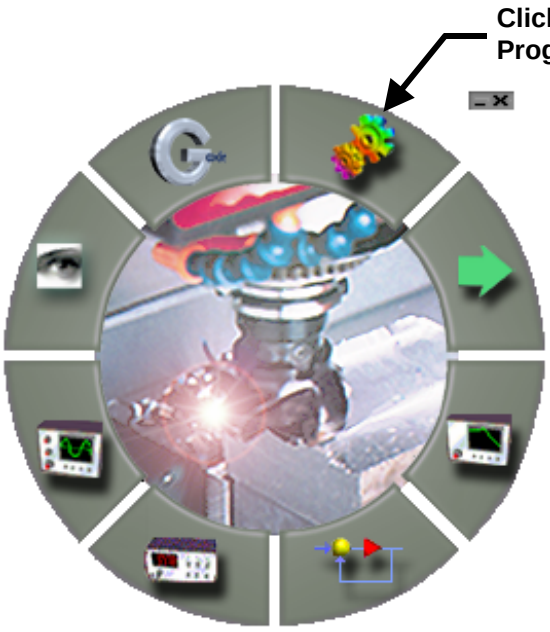
Mx4Pro Development Tools Overview

Fig. 3-1: **Mx4pro v5** Main Menu Screen

Mx4pro Development Tools takes full advantage of the features offered in the 32-bit Windows 2k and XP operating systems. DSPL development, system tuning, system analyzing, table downloading, and several other tasks may be performed in one easy to use environment. Furthermore, all of the tools may be operating simultaneously. The following is a brief overview of tools included with the Mx4pro Development Tools:

- DSPL programs may be edited, compiled, downloaded, and controlled (started and/or stopped) from within the DSPL Development Tool.
- Gains for all axes may be changed in real time and then stored in initialization files that can be linked to your DSPL programs for future use.
- The Signal Generator and Oscilloscope will work in both Bus and Serial mode and allow easy tuning and analyzing of your Mx4 Octavia applications.
- The Frequency Analyzer allows you to analyze your system performance and models your system with up to a 20^{th} order transfer function.
- G Code programs may be edited, compiled, downloaded, and controlled (started and/or stopped) from within the G Code Development Tools.
- The *Mx4 on CD* Tutorials, Mx4 Octavia Hardware Manual, and DSPL Programming Manuals may be started from within the Mx4 Development Tools.
- DSPL variables may be monitored and changed in real time.
- Inputs may be monitored and outputs changed in real time.
- Interrupts may be monitored, cleared, and disabled in real time.
- Position, Velocity, CAM, and Cubic tables may be downloaded from within the Mx4pro Development Tools environment.
- All tools (excluding Frequency Analyzer) work in either Bus or Serial communication mode.

DSPL Program Development



The DSPL programming language is a high-level language resembling C that allows complete motion control programs to be written for the Mx4 Octavia. The DSPL Program Development Tool allows you to create, modify, compile, download, and execute DSPL programs.

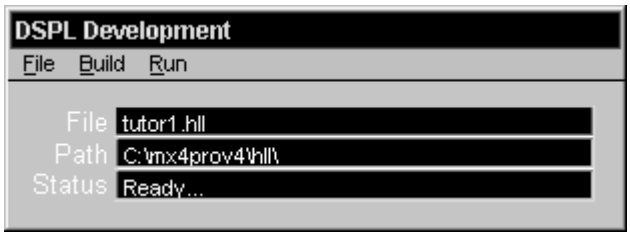


Fig. 3-2: DSPL Development Tool

The DSPL Development tool displays the name of the open DSPL **File**, the **Path** for the open file, and the **Status** of a compile and/or download performed on the file. The following sections describe how to utilize the different features of the DSPL Development Tool.

Gain Tool

The Gain Tool included with the Mx4pro Development Tools allows you to manipulate all of the position loop gains and the notch filter frequency for all axes of the Mx4 Octavia. Also, with the Vx4++ commutation controller option card, the drive control parameters (for all axes) may be modified in real time. There are three windows associated with the Gain Tool; one is dedicated to the Mx4 Octavia controller (Mx4 Gains window), and the other two are dedicated to the optional Vx4++ controller (Vx4++ Gains and Vx4++ Parameter Settings windows).

Mx4 Gains Window

The Mx4 Gains window may be opened by clicking on the **Gain Tool** button on the Mx4pro Development Tools toolbar or by selecting **GainTool...** under the

Tools menu or in the popup menu (right click in the Mx4pro Development Tools main window). The Mx4 Gains window appears as follows:

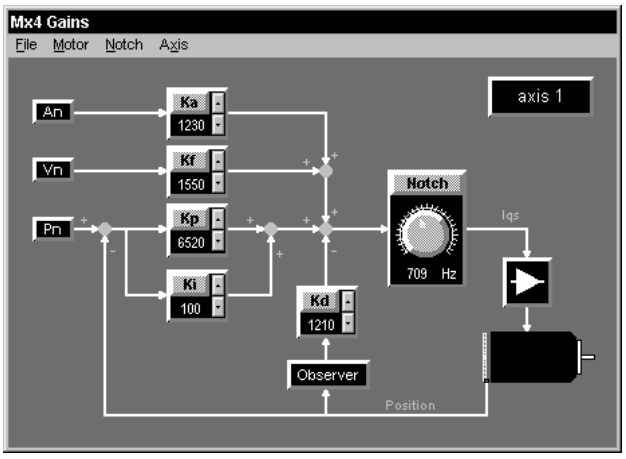


Fig. 3-3: Mx4 Gains window in Gain Tool

The Mx4 Gains window displays the position loop gains, notch filter frequency, and current axis of focus. From within this window, these values can be modified in real-time for all axes. Furthermore, the values for the position loop gains may be saved to an initialization file for future use.

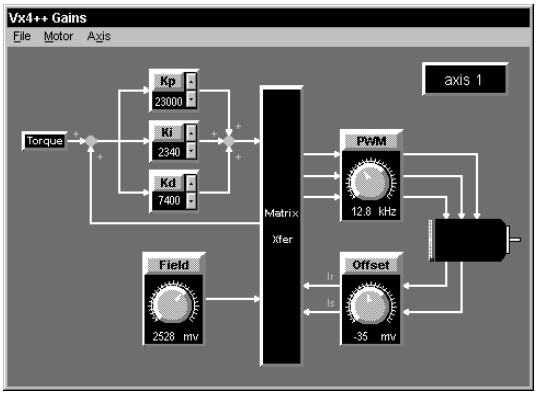


Fig. 3-4: Vx4++ Gains window in Gain Tool

Scopes

The Mx4pro Development Tools supports three separate real-time scopes while in Bus mode and a single real-time scope while in Serial mode. The scopes allow the state variables of the Mx4 Octavia to be viewed graphically as they change with time. The Mx4 Octavia state variables are position, velocity, and error of each axis. Two different types of scopes are available, a Time Scope and an XY Scope.

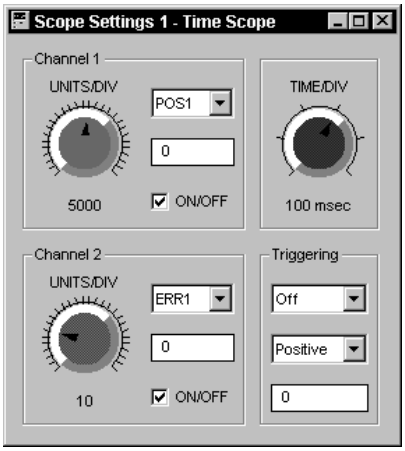
Scope Tool

The Scope Tool allows the selection of Time and XY scopes to be started. To open the Scope Tool, click on the Scope Tool button on the main Mx4pro Development Tools toolbar or select **Oscilloscope** under the **Tools** menu or in the popup menu (right click in the Mx4pro toolbar). The following window will appear:

Fig. 3-5: Main Scope window in Scope Tool

Time Scope

The Time Scope displays up to two separate Mx4 Octavia state variables (vertical axis) against time (horizontal axis). The units per division in both the vertical and the horizontal directions can be adjusted and an offset value may be specified in real-time causing the display to be "shifted" up or down as desired. Triggering options are also available when using a Time Scope. After opening a Time scope, the following two windows will appear:



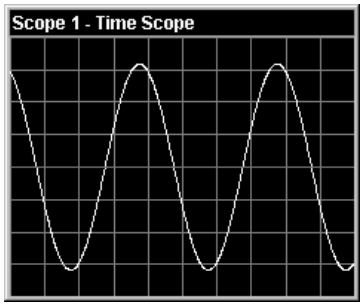


Fig. 3-6: Time Scope windows

XY Scope

The XY Scope plots one value of the Mx4 Octavia state variable against another as the variables change with time. The units per division and offset for the scope channels may also be selected. After opening a Time scope, the following two windows will appear:

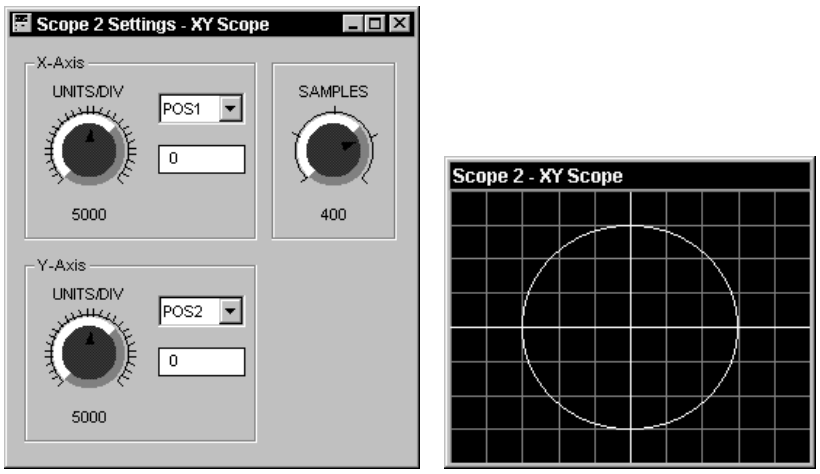
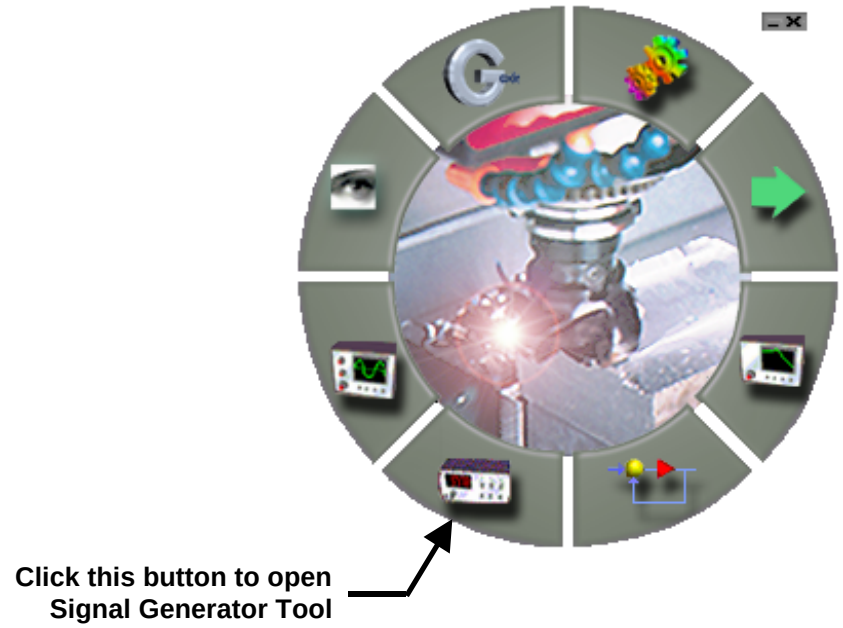


Fig. 3-7: XY Scope windows

Signal Generator



Signal Generator Tool

The Mx4pro Development Tools contains a signal generator useful for optimizing your system. The signal generator is able to generate either trapezoidal or sinusoidal velocity profiles.

The Signal Generator Tool allows trapezoidal and sinusoidal velocity profiles to be configured, started, and stopped for all axes.

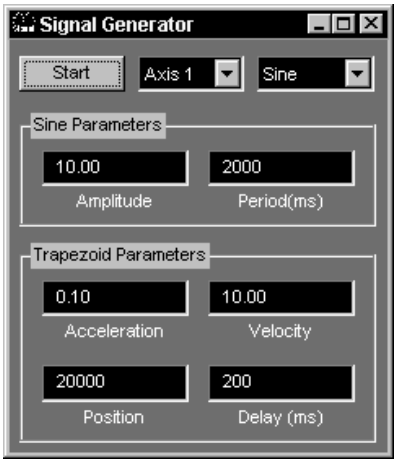
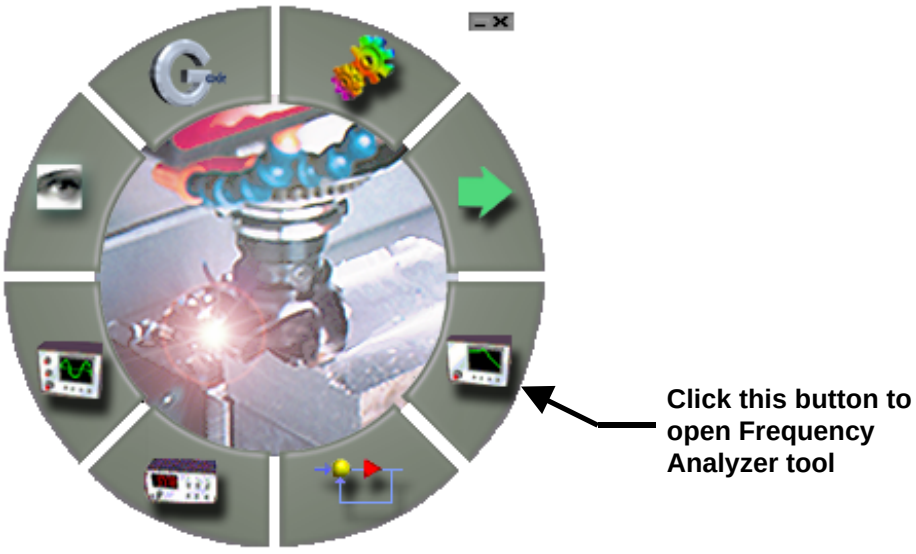


Fig. 3-8: Signal Generator Tool

Frequency Analyzer Tool



Frequency Analyzer Tool

An important tool included with the Mx4pro Development Tools is the Frequency Analyzer or Bode plot. The Frequency Analyzer allows the system performance, bandwidth, stability, and modeling transfer function to be viewed graphically.

The Frequency Analyzer Tool allows the frequency analysis for each axis to be configured and executed.

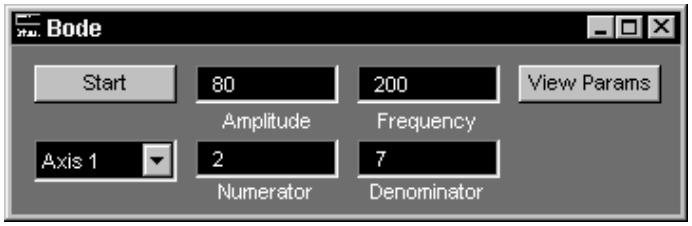


Fig. 3-9: Frequency Analyzer Tool

Two graphs for magnitude and phase display analysis results:

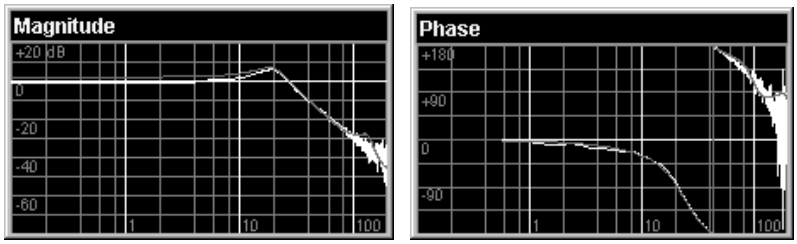


Fig. 3-10: Frequency Analyzer Magnitude and Phase plots

System Modeling

The Frequency Analyzer Tool allows the system to be modeled with a transfer function in the z-domain.

The coefficients for the transfer function may be viewed by clicking on the **View Params** button.

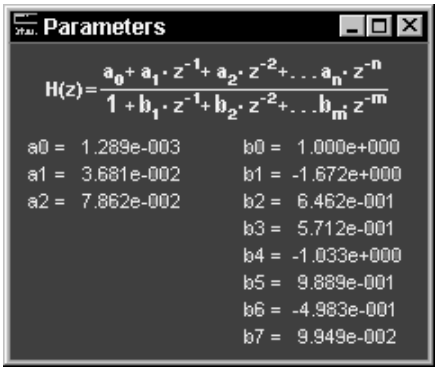
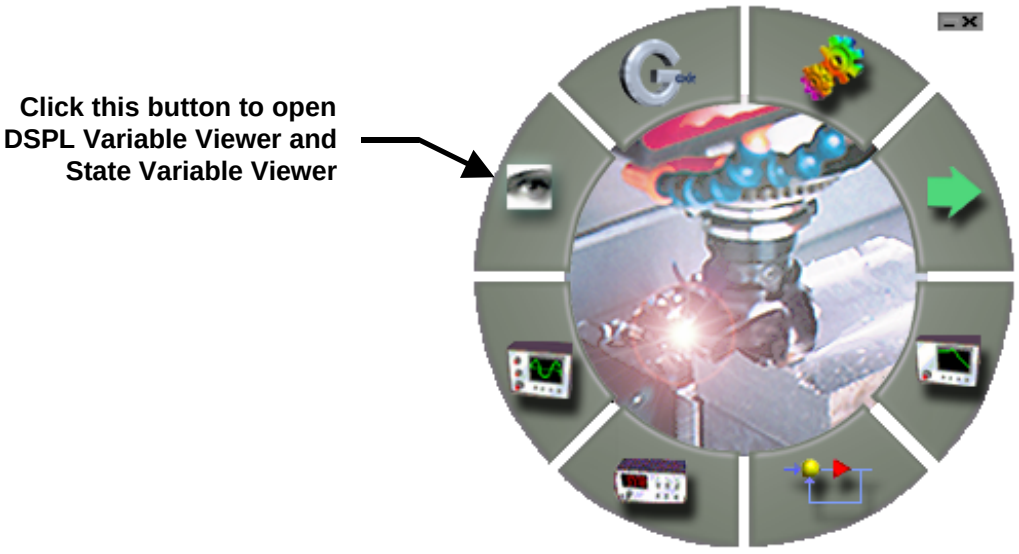


Fig. 3-11: System Model Parameters

Viewers



The Mx4pro Development Tools supports four real-time viewers, a DSPL Variable Viewer, an Inputs and Outputs Viewer, an Interrupts Viewer, and a State Variable Viewer. These viewers monitor the DSPL variables, Mx4 Octavia inputs, Mx4 Octavia interrupts, and the state variables in real-time. They also allow the DSPL variables to be changed, outputs to be set/cleared, and interrupts to be cleared and/or disabled.

DSPL Variable Viewer

The DSPL Variable Viewer allows all 128 DSPL variables to be monitored and changed in real-time.

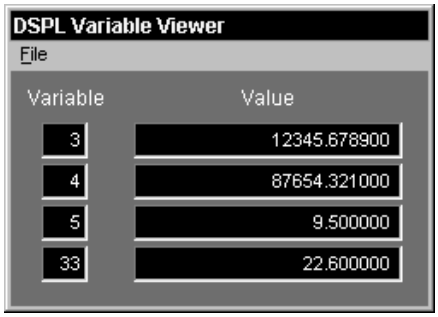


Fig. 3-12: DSPL Variable Viewer

Inputs and Outputs Viewer

The Inputs and Outputs Viewer allows all Mx4 Octavia inputs and outputs to be monitored and changed, respectively, in real-time.

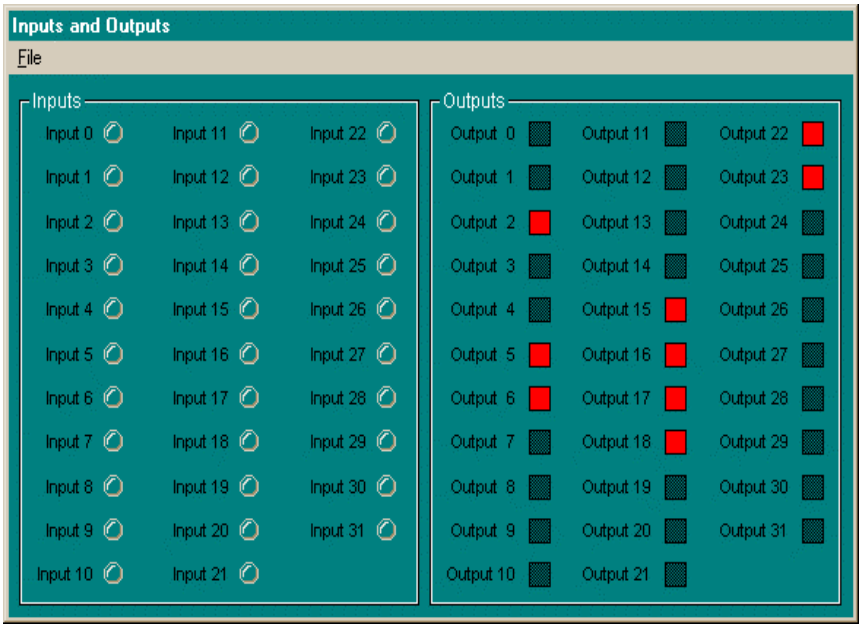


Fig. 3-13: Inputs and Outputs Viewer

Interrupts Viewer

The Interrupts Viewer allows the Buffer Breakpoint, External Int #1, External Int #2, Following Error & Halt, Following Error, Index Pulse, Position Breakpoint, and Motion Complete interrupts to be monitored in real-time.

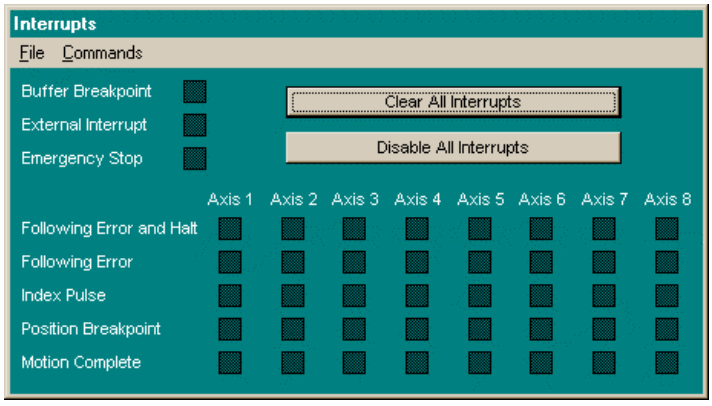


Fig. 3-14: Interrupts Viewer

State Variable Viewer

The State Variable Viewer allows the position, velocity, and following error state variables for all axes to be monitored in real-time.

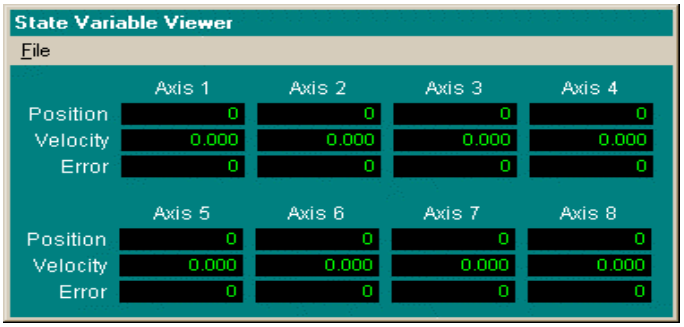
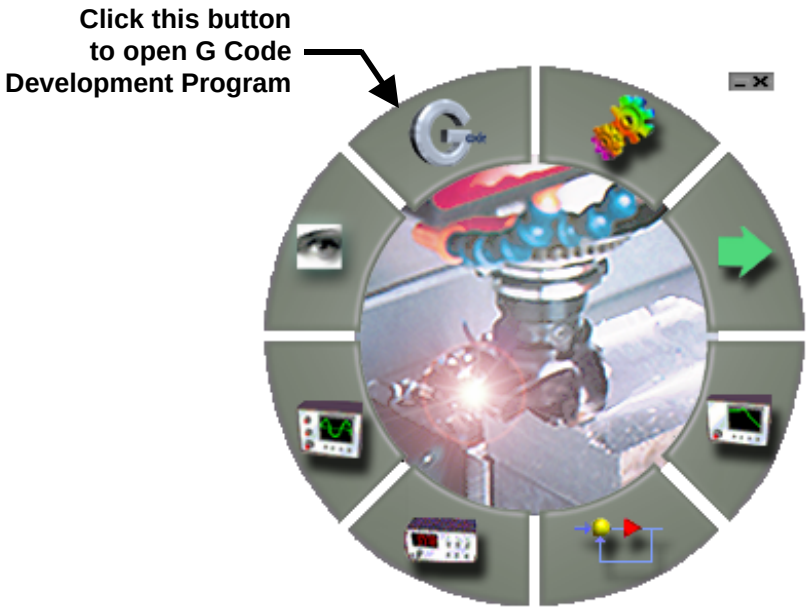


Fig. 3-15: State Variable Viewer

G Code Development



The G Code Development Tool allows you to create, modify, compile, download, and execute G Code programs.

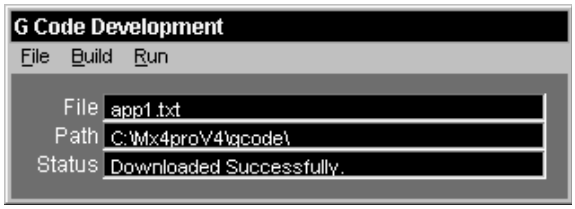


Fig. 3-16: G Code Development Tool

The G Code Development Tool displays the name of the open G Code **File**, the **Path** for the open file, and the resulting **Status** of a compile and/or download performed on the file.

Table

Mx4pro Development Tools allows you to manipulate four major table formats; Cubic Spline Tables, Cam Tables, Position or Velocity End Point Tables, and Position or Velocity Compensation Tables.

Cubic Spline Table

The Cubic Spline Table Tool allows you to create, modify, and download Cubic Spline data tables.

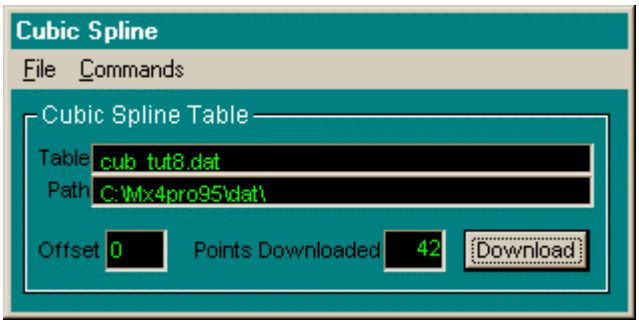


Fig. 3-17: Cubic Spline Table Tool

Cam Tables

The Cam Table Tool allows you to create, modify, and download a table of Cam gear ratios.

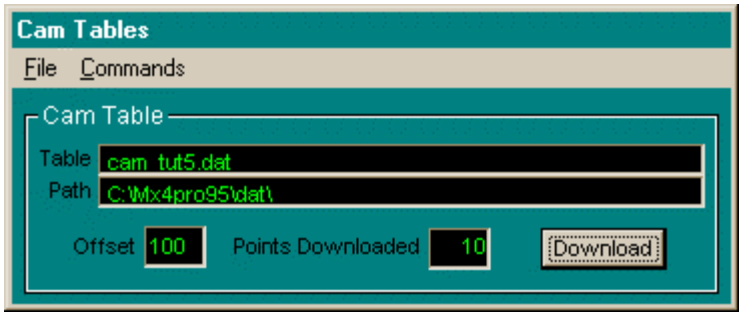


Fig. 3-18: Cam Table Tool

Position or Velocity Points Table

The Position or Velocity Points Table Tool allows you to create, modify, and download tables of either Position or Velocity end points.

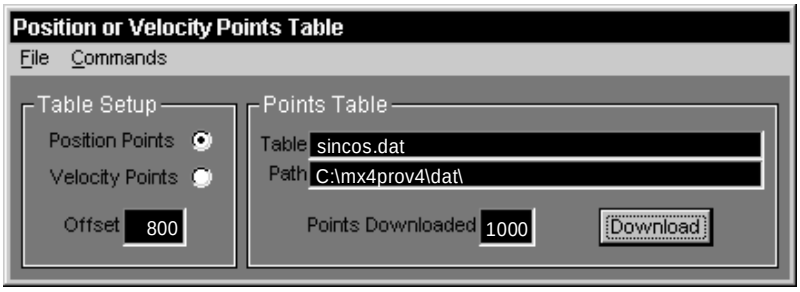


Fig. 3-19: Position or Velocity Points Table Tool

Position and Velocity Compensation Tables

The Position and Velocity Compensation Tables Tool allows you to create, modify, and download tables of Position or Velocity compensation points.

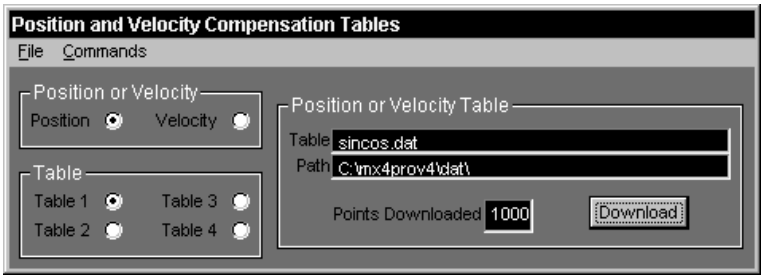


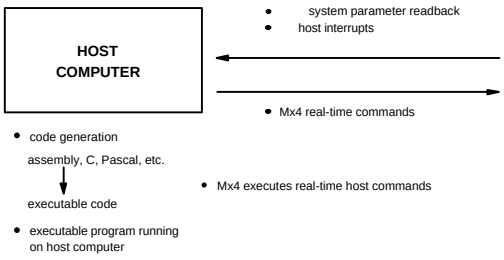
Fig. 3-20: Position and Velocity Compensation Tables Tool

4 Methods of Programming Mx4 Octavia

DSP Control Group has incorporated years of experience in the motion control industry developing Mx4 Octavia's dual programming platform. Mx4 Octavia may be programmed at the C function level through Host-based programming or with the DSPL high-level programming language resident in Mx4.

Host-Based Programming

Low-level Host-based programming entails real-time communication between the host computer and the Mx4 Octavia card via the host computer bus. The host computer may read and write to the Mx4 Octavia card as it would any computer peripheral. The user may choose the programming language for the host computer program. For example, it may be a DOS application written in C, or maybe a Visual Basic Windows NT application. DSPCG provides programming utilities ranging from C functions to Visual Basic/ C DLLs for host-based program development. This host program includes the facilities to transfer commands to the Mx4 Octavia card through the host bus, any conditional program code execution routines, PLC emulating code, an optional interrupt service routine to handle any enabled Mx4 Octavia interrupts, Mx4 Octavia system parameter readback routines and any other software features required for the application. With Host-based programming, an executable host program runs the operation of the Mx4 Octavia card in real-time.



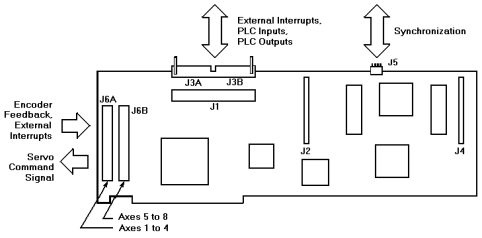


Fig. 4-1: Mx4 Octavia Host-Based Programming

DSPL Programming

The Mx4 Octavia's high-level DSPL programming platform enables complete motion control applications to be written in the DSPL programming language, downloaded once to the Mx4 Octavia card and executed by the Mx4 Octavia card, rather than by the host computer in low-level programming. The DSPL programming language is a powerful, full-featured, yet easy to use language that includes features such as conditional program execution, subroutine calls, separate PLC and motion programming facilities, and the ability to run PLC and multiple Motion programs simultaneously on the Mx4 Octavia card.

A DSPL program consists of a text file, which may be written with any text editor. The DSPL code is then compiled and downloaded to Mx4 Octavia's memory. With the use of the optional non-volatile battery-backup memory available for Mx4 Octavia, stand-alone operation is possible once the DSPL program(s) are downloaded to the card. With the DSPL code loaded into Mx4 Octavia's memory, Mx4 Octavia may begin executing the code. DSPL code execution by Mx4 Octavia is independent of the host computer.

Note: Mx4 Octavia DSPL programming is described in detail in the *DSPL Programmer's Guide*. This document, the *Mx4 Octavia User's Guide*, focuses on Mx4 Octavia Host-based programming./

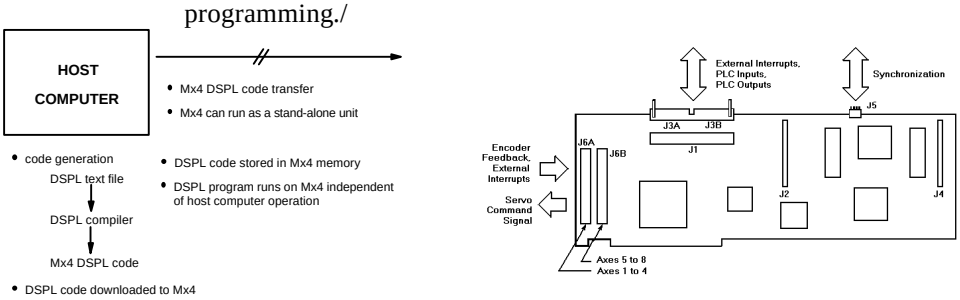


Fig. 4-2: Mx4 Octavia DSPL Programming

Combining DSPL and Host-Based Programming

Although both the Host-based and DSPL Mx4 Octavia programming techniques are full featured and self supporting, the user may choose to combine the two, drawing the advantages of both techniques in solving a particular programming application. While running or executing DSPL PLC and motion programs, Mx4 Octavia is still completely programmable via the host (Host-based programming methods). This feature of Mx4 Octavia allows for a combination of Host-based and DSPL programming. In addition, a synchronizing timing structure may be established between an executing DSPL program and the host computer via Mx4 Octavia's powerful command sets.

Introduction to Mx4 Octavia Host-Based Programming

Note: Mx4 Octavia DSPL programming is described in detail in the *DSPL Programmer's Guide*. This document, the *Mx4 Octavia User's Guide*, focuses on Mx4 Octavia Host-based programming.

The Mx4 Octavia Host-based programming platform includes two types of host-based commands:

Real Time Commands (RTCs)
Contouring Commands

Any combination of the two types of commands is possible for the four axes of control.

Real-Time Commands

Real Time Commands (RTCs) are transferred to Mx4 Octavia through a "window" in the Mx4 Octavia Dual Port RAM (DPR). Mx4 Octavia polls the DPR for RTCs. An RTC is acted upon as soon as Mx4 Octavia reads it. Multi-axis commands are executed simultaneously (not multiplexed), resulting in perfect synchronization for multi-axis control. As soon as a new command is detected, Mx4 Octavia executes it, possibly altering the effects of any previous

commands that were not yet completed. The Mx4 Octavia Host-based programming command set consists entirely of RTCs.

Contouring

Mx4 Octavia supports two types of contouring: 2nd order contouring and cubic spline contouring. Contouring commands consist of segment move commands from which Mx4 Octavia performs 2nd order or cubic spline interpolation. Contouring 'data' is transferred from the host to Mx4 Octavia via a ring buffer in the DPR. (See Fig. 4-3.) Each segment move consists of a 32 bit position value and 32 bit velocity value for each axis included in the contouring motion. Mx4 Octavia interpolates between the [position, velocity] points with programmable intervals. The 'commands' are executed in sequence, with execution commencing only when the previously commanded segment move is complete. A more detailed discussion of contouring commands can be found in Chapter 6 *Mx4 Octavia Host-Based Programming*.

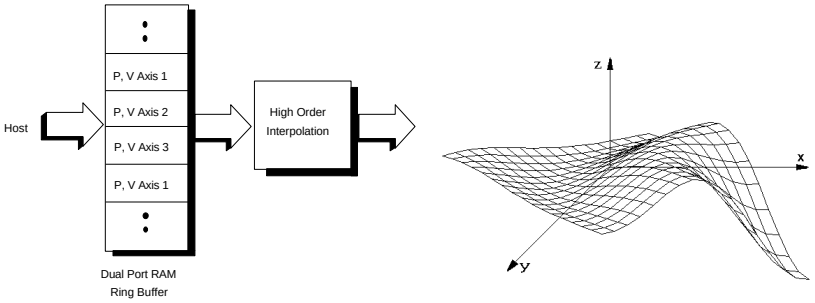


Fig. 4-3: Mx4 Octavia Contouring with Three Axes

This page intentionally blank.

5 Mx4 Octavia Host-Based Instruction Set

Host-Based Programming Command Set

The Mx4 Octavia Host programming platform includes the following Real Time Commands (RTCs). These commands, along with the previously mentioned contouring commands, yield a powerful and very flexible motion control programming platform. The Mx4 Octavia RTCs are categorized as follows:

Control Law & Initialization

Control gains, system parameters, time, position, and velocity units all fall in this category.

COMMAND	DESCRIPTION
CTRL	position, velocity loop control law parameters
CTRL_KA	acceleration feed forward control law parameters
ESTOP_ACC	specify emergency stop maximum acceleration
KILIMIT	integral gain limit
MAXACC	specify maximum acceleration
OFFSET	amplifier offset cancellation
OUTGAIN	position loop output gain
OVERRIDE	feedrate override for LINEAR / CIRCLE motion
POS_PRESET	preset position counters
POS_SHIFT	position counter reference shift
RESET	reset Mx4 Octavia controller card
SYNC	define Mx4 Octavia master/slave status
STEPPER_ON	select servo / stepper axes
TRQ_LIMIT	output torque (voltage) limit

Simple Motion

The instructions within this category control the torque, velocity, and position of one or multiple axes with a trapezoidal profile. The commands in this category may be subcategorized as open and closed loop.

COMMAND	DESCRIPTION
AXMOVE	trapezoidal axis move (closed loop)
AXMOVE_S	s-curve trapezoidal axis move (closed loop)
AXMOVE_T	time-based trapezoidal axis move (closed loop)
DDAC	direct 16-bit DAC command (open loop)
REL_AXMOVE	relative position axis move (closed loop)
REL_AXMOVE_S	relative position s-curve axis move (closed loop)
REL_AXMOVE_T	relative position time-based axis move (closed loop)
STOP	stops the motion (closed loop)
VELMODE	velocity mode (open loop)

Coordinated Motion - Cam

Multi-axis motion control applications may require synchronization of two or more axes in a coordinated task. Mx4 Octavia's cam commands result in master-slaving coordinated motion.

COMMAND	DESCRIPTION
CAM	turns electronic cam on
CAM_OFF	turns only electronic cam off
CAM_OFF_ACC	turns electronic cam off and halts slave(s)
CAM_POINT	place single cam point into cam table
CAM_POS	turns electronic cam on at a specified position
CAM_PROBE	turns electronic cam on after PROBE is set high
LOAD_CAM_TABLE	load cam table data points

Coordinated Motion - Gearing

Multi-axis motion control applications may require synchronization of two or more axes in a coordinated task. Mx4 Octavia's electronic gearing commands result in master/slaving coordinated motion.

COMMAND	DESCRIPTION
GEAR	unconditional 'electronic' gearing
GEAR_OFF	disengage 'electronic' gearing
GEAR_OFF_ACC	turns electronic gearing off and halt slave(s)
GEAR_POS	'electronic' gearing based on position value
GEAR_PROBE	'electronic' gearing based on external interrupt
REL_AXMOVE_SLAVE	superimposes a relative axis move onto a slave engaged in gearing

Input / Output Control

The commands within this category are primarily those performing logical operations on input/output signals.

COMMAND	DESCRIPTION
INP_STATE	configure logic state of inputs
OUTP_OFF	set status of outputs to TTL HIGH
OUTP_ON	set status of outputs to TTL LOW
POSBRK_OUT	position breakpoint output activation

System Diagnostic

In addition to Mx4 Octavia's full diagnostic reporting via the DPR, the host may examine internal Mx4 Octavia parameters and provide debug support with the PARREAD RTC.

COMMAND	DESCRIPTION
PARREAD	Mx4 Octavia system parameter readback

Contouring

Mx4 Octavia Host-based programming includes RTC programming and contouring position, velocity interpolation motion programming. The following RTCs are used in setting up and defining contouring parameters.

COMMAND	DESCRIPTION
BTRATE	block transfer rate for 2nd order contouring
CLEAR_CUBIC_TABLE	clear internal cubic spline data
CUBIC_INT	start the internal cubic spline table
CUBIC_RATE	set cubic spline point transfer rate
CUBIC_SCALE	scales position/velocities, also shifts positions
LOAD_CUBIC_TABLE	load internal cubic spline data
START	start contouring motion
VECCHG	contouring vector change for 2nd order contouring

Interrupt Control

The Mx4 Octavia Host-based programming RTCs include a comprehensive set of instructions to handle interrupts. The complete set of interrupts provided by Mx4 Octavia facilitates data reporting to the host for issues that may have some system level significance.

COMMAND	DESCRIPTION
DISABL_INT	disable the interrupts
EN_BUFBRK	contouring buffer breakpoint interrupt enable
EN_ENCFLT	encoder fault interrupt
EN_ERR	following error interrupt enable
EN_ERRHLT	following error / halt interrupt enable
EN_INDEX	index pulse interrupt enable
EN_MOTCP	motion complete interrupt enable
EN_POSBRK	position breakpoint interrupt enable
EN_PROBE	general purpose external probe interrupt enable
INT5MS	5ms interrupt

DSPL High-Level Language Table Related

These commands are used to set up and control the execution of a DSPL program on Mx4 Octavia.

COMMAND	DESCRIPTION
CLEAR_DSPL	clear DSPL program from Mx4 Octavia memory
CLEAR_POS_TABLE	clear position compensation table
CLEAR_VEL_TABLE	clear velocity compensation table
LOAD_POS_TABLE	load position compensation table
LOAD_VEL_TABLE	load velocity compensation table
TABLE_SEL	select compensation table

DSPL High-Level Language Variable Related

These commands are used to set up and control the execution of a DSPL program on Mx4 Octavia.

COMMAND	DESCRIPTION
CHANGE_VAR	update values for Mx4 Octavia internal variables
MONITOR_VAR	host monitoring of DSPL variables

DSPL High-Level Language Flagging

These commands are used to set up and control the execution of a DSPL program on Mx4 Octavia.

COMMAND	DESCRIPTION
DSPL_AUTOSTART	start DSPL execution at power-up
SIGNAL_DSPL	signal the DSPL program
START_DSPL	begin execution of DSPL program
STOP_DSPL	halt DSPL program execution

Mx4 Octavia State Variables

Before programming the Mx4 Octavia controller, knowledge of Mx4 Octavia's state variables is necessary. The motion state variables are described below.

Acceleration Specified in encoder edge counts/(200 μs)^2. It is presented by a 16-bit unsigned number with 1 bit integer and 15 bits fraction.

i.e., acc = 077Bh = 0.0584 counts/(200 μs)^2

Following Error Specified in encoder edge counts and is presented by a 32-bit two's complement number with all 32 bits as integer.

Position Specified in encoder edge counts and is presented by a 32-bit two's complement number with all 32 bits as integer.

Velocity Specified in encoder edge counts/200 μs and is presented by a 25-bit two's complement number, sign extended to 32 bits. This value is partitioned as 16 bits integer and 16 bits fraction.

i.e., vel = 000A8000h = 10.50 counts/200 μs

Mx4 Octavia Host-Based Programming Command Listing

The Mx4 Octavia Host-based programming RTCs are listed in alphabetical order. Each command listing follows this format:

FUNCTION	indicates the command function
DPR ORDER	order in which the command arguments must be written to the DPR Real Time Command buffer[1]
USAGE	indicates the command usage as follows: Host host programming command DSPL DSPL programming command (PLC) command may be used in PLC programs (Motion) command may be used in Motion programs
ARGUMENTS	command arguments, if any, are defined[2]
DESCRIPTION	explanation of command operation and functionality
SEE ALSO	listing of related commands
APPLICATION	some helpful suggestions describing which applications benefit from the command
EXAMPLE	an example illustrating the command in use

Note 1: See Chapter 6, *Mx4 Octavia Host Programming ... RTCs & Contouring* for a detailed description of how RTCs are transmitted to the Mx4 Octavia controller.

Note 2: Many commands include the argument n ("a single byte, bit coding the axes involved"). The bit coding is as follows:

n	bit 0	axis 1
	bit 1	axis 2
	bit 2	axis 3
	bit 3	axis 4
	bit 4	axis 5
	bit 5	axis 6
	bit 6	axis 7
	bit 7	axis 8

For example, 0x30 bit codes axes 5 and 6; 0x0E bit codes axes 2, 3, 4, etc.

AXMOVE

FUNCTION Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, vel_1, ... , acc_8, pos_8, vel_8

USAGE Host (command code: 60h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit end position for axis x
vel_x	32-bit unsigned slew rate for axis x

Note 1: Position and velocity are always presented in two's complement format, but acceleration is an unsigned value.

Note 2: Velocity must be presented as a 25-bit two's complement value, which is sign extended to 32 bits. For example, the maximum unsigned velocity is 00FFFFFFh.

Note 3: Velocity is partitioned into 16 bits integer and 16 bits fraction. Position is a 32-bit integer value, and acceleration is presented as 1 bit integer, 15 bits fraction.

DESCRIPTION

The AXMOVE RTC allows for trapezoidal command generation with specified endpoint position, slew rate velocity and acceleration for each axis. This command is suitable for linear moves.

SEE ALSO REL_AXMOVE, STOP

AXMOVE_S, AXMOVE_T, REL_AXMOVE_S,

REL_AXMOVE_T

AXMOVE cont.

APPLICATION

This command can be used in almost any imaginable motion control application. Applications may benefit from this command any time there is a need for a linear move from point A to point B in a multi-dimensional space. To name a few applications: pick and place robots (e.g., in component insertion), rapid traverse (e.g., in machining), and master/slaving (e.g., in paper processing and packaging) applications.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gain values
KILIMIT ( )
AXMOVE ( )      ;run system in axis move (linear trapezoidal) mode
:
EN_MOTCP ( )    ;enable motion complete
                ;upon the completion of this (command) trajectory
                ;Mx4 Octavia generates motion complete interrupt
```

EXAMPLE 1

Assuming current positions of zero for axes 1 and 2, we want to move axis 1 to the target position of 234567h and axis 2 to the target position of 112233h. Let's also assume that we want this move to be accomplished with the slew rate velocity of 200000h (200000h/2^{16} counts/200 µs) and acceleration of 150h (150h/2^{15} counts/(200 µs)^2) for both axes.

The values of the RTC arguments are:

n	:	03h
acc_1	:	0150h
pos_1	:	00234567h
vel_1	:	00200000h
acc_2	:	0150h
pos_2	:	00112233h
vel_2	:	00200000h

AXMOVE cont.

EXAMPLE 2

Assuming a current position of zero for axis 6, we want to move axis 6 to the (negative) target position of FFAA0000h with a slew rate of 00E00000h (00E00000h/2^{16} counts/200 µs)(unsigned velocity) and acceleration of 150h (150h/2^{15} counts/(200 µs)^2).

The values of the RTC arguments are:

n	:	20h
acc_6	:	0150h
pos_6	:	FFAA0000h
vel_6	:	00E00000h

EXAMPLE 3

The host can issue a new axis move command before the previous one is completed. For example, assume the AXMOVE RTC of Example 1 is issued by the host. Now, the host changes its mind and decides to stop axis 2 at a new target position of 334455h with a new slew rate of 100000h (100000h/2^{16} counts/200µs) and a new acceleration of 200h (200h/2^{15} counts/(200µs)^2). While the AXMOVE of Example 1 is in progress, the host issues the new command.

The values of the RTC arguments are:

n	:	02h
acc_2	:	0200h
pos_2	:	00334455h
vel_2	:	00100000h

AXMOVE_S

FUNCTION S-Curve Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, vel_1, ... , acc_8, pos_8, vel_8

USAGE Host (command code: 82h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit target position for axis x
vel_x	32-bit unsigned slew rate for axis x

Note 1: Position and velocity are always presented in two's complement format, but acceleration is an unsigned value.

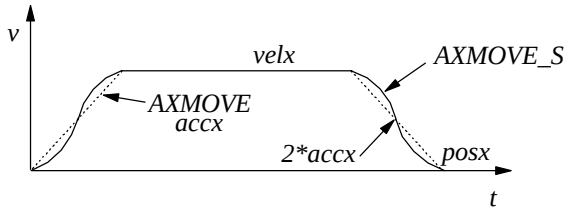
Note 2: Velocity must be presented as a 25-bit two's complement value, which is sign extended to 32 bits. For example, the maximum unsigned velocity is 00FFFFFFh.

Note 3: Velocity is partitioned into 16 bits integer and 16 bits fraction. Position is a 32-bit integer value, and acceleration is presented as 1 bit integer, 15 bits fraction.

DESCRIPTION

The AXMOVE_S RTC allows for s-curve command generation with specified endpoint position, slew rate velocity, and acceleration for each axis. This command is suitable for linear moves where s-curve acceleration is desired.

AXMOVE_S cont.



The figure above illustrates the velocity profile of the AXMOVE_S along with the linear velocity ramp of the AXMOVE command. With AXMOVE_S, the acceleration will reach a value of 2*accx for a maximum (see above figure).

SEE ALSO AXMOVE, AXMOVE_T, REL_AXMOVE, REL_AXMOVE_S, REL_AXMOVE_T, STOP

APPLICATION

Refer to *DSPL Application Programs*.

EXAMPLE 1

Assuming current positions of zero for axes 4 and 5, we want to move axis 4 to the target position of 234567h and axis 5 to the target position of 112233h. Let's also assume that we want this move to be accomplished with the slew rate velocity of 200000h (200000h/2^{16} counts/200 µs) and acceleration reference of 150h (150h/2^{15} counts/(200 µs)^2) for both axes.

The values of the RTC arguments are:

n	:	18h
acc_4	:	0150h
pos_4	:	00234567h
vel_4	:	00200000h
acc_5	:	0150h
pos_5	:	00112233h
vel_5	:	00200000h

AXMOVE_S cont.

EXAMPLE 2

Assuming a current position of zero for axis 7, we want to move axis 7 to the (negative) target position of FFAA0000h with a slew rate of 00E00000h (00E00000h/2^{16} counts/200 µs)(unsigned velocity) and acceleration reference of 150h (150h/2^{15} counts/(200 µs)^2).

The values of the RTC arguments are:

n	:	40h
acc_7	:	0150h
pos_7	:	FFAA0000h
vel_7	:	00E00000h

EXAMPLE 3

The host can issue a new axis move command before the previous one is completed. For example, assume the AXMOVE RTC of Example 1 is issued by the host. Now, the host changes its mind and decides to stop axis 5 at a new target position of 334455h with a new slew rate of 100000h (100000h/2^{16} counts/200µs) and a new acceleration reference of 200h (200h/2^{15} counts/(200µs)^2). While the AXMOVE of Example 1 is in progress, the host issues the new command.

The values of the RTC arguments are:

n	:	10h
acc_5	:	0200h
pos_5	:	00334455h
vel_5	:	00100000h

AXMOVE_T

FUNCTION Time-Based Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, tm_1, ... , acc_8, pos_8, tm_8

USAGE Host (command code: 8Fh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit target position for axis x
tm_x	32-bit unsigned time for axis x

Note 1: Position is always presented in two's complement format, but acceleration and time are unsigned values.

Note 2: The time argument, tm_x, is a 32 bit unsigned value with a unit of 200µsec.

DESCRIPTION

The AXMOVE_T RTC allows for trapezoidal command generation with specified endpoint position, acceleration, and time to complete the move for each axis. This command is suitable for linear moves where endpoint position and motion time are the specifying parameters.

AXMOVE_T cont.

The AXMOVE_T command is similar to AXMOVE, with the exception that the velocity argument is replaced with a time argument. AXMOVE_T will automatically calculate a suitable slew rate velocity to achieve the programmed endpoint position in the programmed amount of time, following a trapezoidal velocity profile (similar to AXMOVE).

SEE ALSO AXMOVE, AXMOVE_S, REL_AXMOVE, REL_AXMOVE_S, REL_AXMOVE_T, STOP

APPLICATION

Refer to *DSPL Application Programs*.

EXAMPLE

Move axis 1 to the target position of 234567h and axis 3 to the target position of 112233h. Let's assume that we want this move to be accomplished with the acceleration reference of 150h (150h/2^{15} counts/(200 µs)^2) and a time of 50msec (250*200µsec) for both axes.

The values of the RTC arguments are:

n	:	05h
acc_1	:	0150h
pos_1	:	00234567h
tm_1	:	000000FAh
acc_3	:	0150h
pos_3	:	00112233h
tm_3	:	000000FAh

BTRATE

FUNCTION Set 2nd Order Contour Block Transfer Rate

DPR ORDER command code, m

USAGE Host (command code: 73h), DSPL (Motion)

ARGUMENTS

m — a byte which selects the block transfer rate for all of the axes

m is an integer ranged from 0 to 3

m=0	block transfer rate is 5 ms per point
m=1	block transfer rate is 10 ms per point
m=2	block transfer rate is 15 ms per point
m=3	block transfer rate is 20 ms per point

DESCRIPTION

This command sets the 2nd order contouring block transfer rate for the system. For example, if the block transfer rate is set at 10 ms, the time interval between each point in the ring buffer is '10 ms' (e.g., the DSP will interpolate each point for 10 ms).

Note: The host should not adjust the block transfer rate when contouring is in process.

Note: The default block transfer rate is set at 5 ms per point.

SEE ALSO CUBIC_RATE

BTRATE cont.

APPLICATION

This command is useful in 2nd order contouring applications. Depending on the capability of the host processor, position/velocity points on multi-dimensional trajectories may be broken down to the points that (timewise) may be near or far from each other. Clearly, slower CPUs are capable of breaking down geometries to position and velocity points that are widely spaced in time. This instruction makes the time interval in between the two adjacent points (in contouring) programmable. Please remember that regardless of the value programmed for this time interval (5, 10, 15, or 20 ms), Mx4 Octavia will internally perform a high-order interpolation of the points breaking them down to 200 μs, all axes included.

Command Sequence Example

See EN_BUFBRK

EXAMPLE

Set a contouring interpolation interval of 10 ms.

The value of the RTC argument is:

m : 01h

CAM

FUNCTION Engage Electronic Cam

DPR ORDER command code, n, m, $tablestart_1$, $tablesize_1$, ..., $tablestart_8$, $tablesize_8$

USAGE Host (command code: A4h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding theONLY master axis
m	a single byte, bit coding the slave axis(es)
$tablestart_x$	16-bit unsigned value, specifies cam table start index for slave axis x
	$0 <= tablestart_x <= 1600$
$tablesize_x$	16-bit unsigned value, specifies cam table size for slave axis x
	$3 <= tablesize_x <= 1600$

DESCRIPTION

The commands that make up the electronic cam feature are CAM, CAM_OFF, CAM_OFF_ACC, CAM_POINT, CAM_POS, and CAM_PROBE.

The Mx4 controller is capable of storing up to 1600 cam points. Each cam point consists of a master relative position and an associated slave relative position. A cam table can be between 3 and 1600 cam points long, and the user may define any number of cam tables in the 1600-point cam table capacity. Cam commands utilize tablestart and tablesize arguments to specify which 'portion' of the 1600-point cam table region to 'run' on.

Cam table points may be downloaded in file format from within Mx4pro, by using the LOAD_CAM_TABLE RTC within an application, or built one point at a time using the CAM_POINT command. The CAM_POINT command may also be used to modify cam points 'on the fly.'

CAM cont.

The DPR identifiers CAMCOUNT1,2,3,etc. indicate at which cam table indices the slave axis(es) are 'at' (CAMCOUNT1 is for axis 1, etc.).

The cam points consist of relative position values for master and slave (when using the LOAD_CAM_TABLE RTC, cam points must also include a gear ratio). The first cam point in a table must be 0, 0. The last point in a cam table is the cycle length for master and slave. For example, if the full cam cycle for a master axis is 5000 counts and the slave would travel -1024 counts in that cycle, the last cam point in that cam table would be 5000, -1024. Note that the master/slave position ratios can not exceed the range [-256 to 255,999]. Also, the minimum ratio is +/- 1/128. For example, for 1000 counts of the master axis, the slave axis(es) can not have more than -256000 counts in the negative direction or 255999 counts in the positive direction.

The slave axes utilize the `MAXACC` acceleration value as the maximum acceleration the slave axis can reach while following the electronic cam trajectory, and therefore must be programmed before cam operation. This command turns on the mechanical cam function for the selected master and slave(s). The slave(s) follow the master according to the master/slave position pairs stored in the cam table. The slave axis(es) utilize `MAXACC` as the maximum acceleration they can achieve in following the master trajectory.

Note: Activation of *ESTOP during cam operation will halt the master axis, and subsequently the slave axis(es). Slave(s) remain "engaged" in cam mode after the input-triggered halt.

SEE ALSO CAM_OFF, CAM_OFF_ACC, CAM_POINT, CAM_POS, CAM_PROBE, LOAD_CAM_TABLE, MAXACC

APPLICATION

General master/slaving, in particular packaging, synchronous cutting, flying shear, and mark registration, require the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table.

CAM cont.

EXAMPLE

Set axis 1 as the master axis, axes 2 and 3 as slaves. The axis 2 slave will use the 10-point cam table beginning at index 0, while the axis 3 slave will use the 25 point cam table beginning at index 100.

The values of the RTC arguments are:

n	=	0x01h
m	=	0x06h
$tablestart_2$	=	0x0000h
$tablesize_2$	=	0x000Ah
$tablestart_3$	=	0x0064h
$tablesize_3$	=	0x0019h

CAM_OFF

FUNCTION Turns Off, Disengages Cam Slave Axis(es)

DPR ORDER command code, n

USAGE Host (command code: A7h), DSPL (Motion)

ARGUMENTS

n — a single byte bit coding the slave axis(es) to be disengaged

DESCRIPTION

This command disengages the system that was under master slave control.

SEE ALSO CAM, CAM_OFF_ACC, CAM_POINT, CAM_POS, CAM_PROBE, LOAD_CAM_TABLE

APPLICATION

General master/slaving, in particular packaging, synchronous cutting, flying shear, and mark registration, require the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table.

EXAMPLE

Immediately disengage slave axes 3 and 4 from the master axis.

The value of the RTC argument is:

n = 0x0Ch

CAM_OFF_ACC

FUNCTION Turns Off, Disengages Cam Slave Axis(es) With Acceleration

DPR ORDER command code, n

USAGE Host (command code: A8h), DSPL (Motion)

ARGUMENTS

n — a single byte bit coding the slave axis(es) to be disengaged

DESCRIPTION

This command disengages the system that was under master slave control. The slave axis(es) will come to a stop at the maximum acceleration rate programmed by MAXACC.

SEE ALSO CAM, CAM_OFF, CAM_POINT, CAM_POS, CAM_PROBE, LOAD_CAM_TABLE

APPLICATION

General master/slaving, in particular packaging, synchronous cutting, flying shear, and mark registration, require the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table.

EXAMPLE

Disengage with acceleration profile slave axes 8 and 4 from the master axis.

The value of the RTC argument is:

n = 0x88h

CAM_POINT

FUNCTION Place Cam Point Into Cam Table

DPR ORDER command code, tablestart, tablesize, index, masterpos, slavepos

USAGE Host (command code: B3h), DSPL (Motion)

ARGUMENTS

tablestart — 16-bit unsigned value specifies cam table start index

0 <= tablestart <= 1600

tablesize — 16-bit unsigned value specifies cam table size

3 <= tablesize <= 1600

index — 16-bit unsigned value specifies index at which to place the cam point

0 <= index <= (tablesize-1)

masterpos — 32-bit two's complement value cam point master axis relative position

slavepos — 32-bit two's complement value cam point slave axis relative position

DESCRIPTION

The `CAM_POINT` allows the user to either build entire cam tables from a host application or, alternatively, edit cam table points (i.e.: change cam points 'on the fly'). Cam table points consist of master, slave position pairs, and cam tables can be anywhere from 3 to 1600 cam points long. The first point of a cam table (index = 0) must be 0,0. The last point of a cam table (index = tablesize-1) is mastercyclelength, slavecyclelength; where the cycle lengths for the master and slave are the relative cam cycle lengths (i.e.: master cycle length is 4096 counts, the slave cycle length is 1024 counts, for a full cycle ratio of 4:1). Cam master/slave position ratios can not exceed the range [-256 to 255,999]. Also, the minimum ratio is +/- 1/128.

CAM_POINT cont.

SEE ALSO CAM, CAM_OFF, CAM_OFF_ACC, CAM_POS, CAM_PROBE, LOAD_CAM_TABLE

APPLICATION

See Application Notes.

EXAMPLE

A 10-point cam table exists at table start index 500. Replace the 3rd point of the table with the master, slave point 1000, 3000.

The values of the RTC arguments are:

tablestart	=	0x01F4h
tablesize	=	0x000Ah
index	=	0x0002h
masterpos	=	0x000003E8h
slavepos	=	0x00000BB8h

CAM_POS

FUNCTION Turns Electronic Cam On at a Specified Position

DPR ORDER command code, n, m, $masterpos_1$, $tablestart_1$, $tablesize_1$, ..., $masterpos_8$, $tablestart_8$, $tablesize_8$

USAGE Host (command code: A5h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding theONLY master axis
m	a single byte, bit coding the slave axis(es)
$masterpos_x$	32-bit two's complement value specifying the master position value for slave axis x that the electronic cam engages
$tablestart_x$	16-bit unsigned value, specifies cam table start index for slave axis x 0 <= $tablestart_x$ <= 1600
$tablesize_x$	16-bit unsigned value, specifies cam table size for slave axis x 3 <= $tablesize_x$ <= 1600

DESCRIPTION

This command engages at the specified master position the mechanical cam function for the selected master and slave(s). The slave(s) follow the master according to the master/slave position pairs stored in the cam table. The slave axis(es) utilizes MAXACC as the maximum acceleration they can achieve in following the master trajectory.

Note: Activation of *ESTOP during cam operation will halt the master axis, and subsequently the slave axis(es). Slave(s) remain “engaged” in cam mode after the input-triggered halt.

CAM_POS cont.

SEE ALSO CAM, CAM_OFF, CAM_OFF_ACC, CAM_POINT, CAM_PROBE, LOAD_CAM_TABLE, MAXACC

APPLICATION

General master/slaving, in particular packaging, synchronous cutting, flying shear, and mark registration, require the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table.

EXAMPLE

Set axis 4 as the master axis, axes 2 and 3 as slaves. The axis 2 slave will use the 10-point cam table beginning at index 0, while the axis 3 slave will use the 25-point cam table beginning at index 100. Axis 2 slave should engage when the master axis is at position 1000, and axis 3 slave should engage when the master axis is at position 4096.

The values of the RTC arguments are:

n	=	0x08h
m	=	0x06h
$masterpos_2$	=	0x000003E8h
$tablestart_2$	=	0x0000h
$tablesize_2$	=	0x000Ah
$masterpos_3$	=	0x00001000h
$tablestart_3$	=	0x0019h
$tablesize_3$	=	0x0064h

CAM_PROBE

FUNCTION Turns Electronic Cam On After Probe Input

DPR ORDER command code, n, m, q, $tablestart_1$, $tablesize_1$, ..., $tablestart_8$, $tablesize_8$

USAGE Host (command code: A6h), DSPL (Motion)

ARGUMENTS

n — a single byte, bit coding theONLY master axis

m — a single byte, bit coding the slave axis(es)

q — a single byte,specifies the *EXTx probe interrupt to be used

[Mx4]

q=1	:	*EXT1
q=2	:	*EXT2

[Mx4 Octavia]

q=1	:	*EXT1
q=2	:	*EXT2
q=4	:	*EXT3
q=8	:	*EXT4

$tablestart_x$ — 16-bit unsigned value, specifies cam table start index for slave axis x

$0 <= tablestart_x <= 1600$

$tablesize_x$ — 16-bit unsigned value, specifies cam table size for slave axis x

$3 <= tablesize_x <= 1600$

CAM_PROBE cont.

DESCRIPTION

This command engages at the occurrence of the specified external interrupt (`*EXT1,2,3,4`) the mechanical cam function for the selected master and slave(s). The slave(s) follow the master according to the master/slave position pairs stored in the cam table. The slave axis(es) utilizes `MAXACC` as the maximum acceleration they can achieve in following the master trajectory.

Note: Execution of the `CAM_PROBE` command will disable any previously enabled `EN_PROBE` interrupt. Probe input (*EXT1, *EXT2, *EXT3, or *EXT4) activation does not generate an interrupt with the `CAM_PROBE` command.

Note: Activation of *ESTOP during cam operation will halt the master axis, and subsequently the slave axis(es). Slave(s) remain "engaged" in cam mode after the input-triggered halt.

SEE ALSO CAM, CAM_OFF, CAM_OFF_ACC, CAM_POINT, CAM_POS, LOAD_CAM_TABLE, MAXACC

APPLICATION

General master/slaving, in particular packaging, synchronous cutting, flying shear, and mark registration, require the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table.

CAM_PROBE cont.

EXAMPLE

Set axis 2 as the master axis, axes 1 and 7 as slaves. The axis 1 slave will use the 100-point cam table beginning at index 0, while the axis 7 slave will use the 250-point cam table beginning at index 220. Engage slave axes in cam at occurrence of *EXT2 interrupt.

The values of the RTC arguments are:

n	=	0x02h
m	=	0x41h
q	=	0x02h
$tablestart_1$	=	0x0000h
$tablesize_1$	=	0x0064h
$tablestart_7$	=	0x00DCh
$tablesize_7$	=	0x00FAh

CHANGE_VAR

FUNCTION Change DSPL Variable (VAR1-128) Values

DPR ORDER command code, m, var_1, $value_1$, ... , var_x, $value_x$

USAGE Host (command code: ADh)

ARGUMENTS

m — a single byte, indicating the number of variables to be updated

$1 \leq m \leq 8$

var_x — a single byte, specifying a variable (number) to be updated

$1 \leq var_x \leq 128$

$value_x$ — 48-bit value (DSPL floating point format) to be loaded to VAR (var_x)

Note: DSPCG provides C-code conversion utilities for converting between C float formats and DSPCG DSPL float format. Contact DSPCG for more information.

DESCRIPTION

DSPL variable values may be changed in real time via the CHANGE_VAR command. In conjunction with the MONITOR_VAR command, CHANGE_VAR provides a host-based real-time inspection and update of DSPL variables and arguments. CHANGE_VAR gives the user complete flexibility when DSPL programs include variables as command arguments.

SEE ALSO MONITOR_VAR

CLEAR_CUBIC_TABLE

FUNCTION Clear Internal Cubic Spline Data

DPR ORDER command code

USAGE HOST (command code: B6h), DSPL (Motion)

ARGUMENTS

none

DESCRIPTION

This command clears the Mx4 Octavia internal cubic spline data storage area.

SEE ALSO LOAD_CUBIC_TABLE

EXAMPLE

Refer to *Cubic Spline Application Notes.*

CLEAR_DSPL

FUNCTION Clear Mx4 Octavia DSPL Program Storage Area

DPR ORDER command code

USAGE HOST (command code: 95h)

ARGUMENTS

none

DESCRIPTION

This command clears the Mx4 Octavia DSPL program storage area.

EXAMPLE

Refer to *DSPL Programmer's Guide*.

CLEAR_POS_TABLE

FUNCTION Clear Position Compensation Table

DPR ORDER command code, n

USAGE HOST (command code: 98h)

ARGUMENTS

n — a single byte, bit coding the position compensation table(s) to be cleared

DESCRIPTION

The specified position compensation table(s) (for circular interpolation) is cleared (to zeroes) upon execution of a CLEAR_POS_TABLE command.

SEE ALSO CIRCLE, CLEAR_VEL_TABLE, TABLE_ON (*DSPL Programmer's Guide*), TABLE_OFF (*DSPL Programmer's Guide*), LOAD_POS_TABLE, TABLE_SEL

APPLICATION

For use with DSPL programming of Mx4 Octavia. (See *DSPL Programmer's Guide*.)

EXAMPLE

Clear the circular interpolation position compensation tables 1, 2, and 3.

The value of the RTC argument is:

n : 07h

CLEAR_VEL_TABLE

FUNCTION Clear Velocity Compensation Table

DPR ORDER command code, n

USAGE HOST (command code: 99h)

ARGUMENTS

n — a single byte, bit coding the velocity compensation table(s) to be cleared

DESCRIPTION

The specified velocity compensation table(s) (for circular interpolation) is cleared (to zeroes) upon execution of a CLEAR_VEL_TABLE command.

SEE ALSO CIRCLE, CLEAR_POS_TABLE, TABLE_ON (*DSPL Programmer's Guide*), TABLE_OFF (*DSPL Programmer's Guide*), LOAD_VEL_TABLE, TABLE_SEL

APPLICATION

For use with DSPL programming of Mx4 Octavia. (See *DSPL Programmer's Guide*.)

EXAMPLE

Clear the circular interpolation velocity compensation table 4.

The value of the RTC argument is:

n : 08h

CTRL

FUNCTION Control Law Parameters

DPR ORDER command code, n, par_{11}, ... , par_{14}, ... , par_{n1}, ... , par_{n4}

USAGE Host (command code: 62h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
par_{x1} (K_i)	16-bit positive value for the first control parameter for axis x
par_{x2} (K_p)	16-bit positive value for the second control parameter for axis x
par_{x3} (K_f)	16-bit positive value for the third control parameter for axis x
par_{x4} (K_d)	16-bit positive value for the fourth control parameter for axis x

DESCRIPTION

This command performs a state feedback control algorithm combined with a modified PID. The state feedback control algorithm includes an observer which estimates the instantaneous values for speed and acceleration. The feedback loops are then individually commanded to provide a robust control, which is smooth and stable over a wide rage of servo operation. In addition this algorithm performs a modified PID with the saturation threshold set for integral action. A common PID includes two zeros and one pole, which may not be suitable for systems with noisy feedback. Also, the integral part of a common PID algorithm may saturate the registers creating overshoots or other forms of instability. A modified PID includes a second pole to solve the latter problem and a programmable integral limit to solve the former one.

In the modified PID algorithm; par_1, par_2, par_3, and par_4 are values representing the integral, proportional, velocity state feedforward, and differential gains, respectively.

CTRL cont.

Scaling Factors

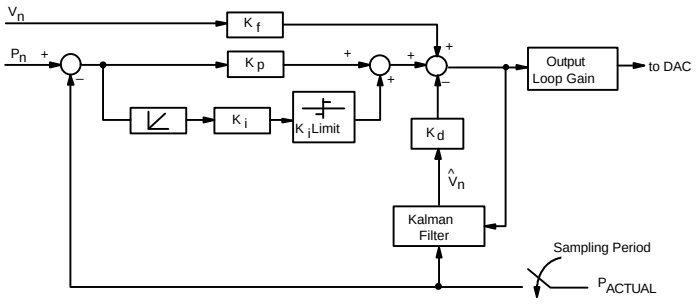
The DSP uses an internal scaling factor for each gain. These factors have been optimally selected for worst case numerical conditions. These factors are:

GAIN	SCALING FACTOR	VALUE
K_f	1.525E-08	v/(c/s)
K_p	0.595E-06	v/c
K_i	3.308E-05	(v/s)/c
K_d	1.9875E-08	v/(c/s)
Output Loop Gain	integer	NA
v = volts, c = encoder edge counts, s = seconds		

For example,

100 counts of position error and K_p of 1000 (other gains are zero) will result in an output voltage of 59.5 millivolts.

i.e. $100 \times 1000 \times 0.595E\text{-}06 = 59.5$



Block Diagram of Control Law

SEE ALSO KILIMIT, OFFSET, OUTGAIN, CTRL_KA

CTRL cont.

APPLICATION

This command is used in all position/velocity control tuning applications. For more information on the effectiveness of each gain on system dynamic response, please refer the *Mx4Pro: Mx4 Tuning Expert* manual to help you understand the significance of gains in tuning. Please read this manual even if you cannot run *Mx4Pro* on your machine because it lacks the DOS operating system.

Command Sequence Example

See AXMOVE and VELMODE

EXAMPLE

Set the following modified PID gain values for axes 4 and 6:

K_i = 100
K_p = 4000
K_f = 3000
K_d = 2500

K_i = 20
K_p = 8000
K_f = 5500
K_d = 7000

The values of the RTC arguments are:

n : 28h
par_{21} : 0064h
par_{22} : 0FA0h
par_{23} : 0BB8h
par_{24} : 09C4h
par_{41} : 0014h
par_{42} : 1F40h
par_{43} : 157Ch
par_{44} : 1658h

CTRL_KA

FUNCTION Acceleration Feedforward Control Law Parameter

DPR ORDER command code, n, ka_1, ... , ka_{08}

USAGE Host (command code: 59h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
ka_x	16-bit positive value for the accel feedforward parameter for axis x

DESCRIPTION

The CTRL_KA command allows the user to program an acceleration feedforward gain for the specified axis(es).

SEE ALSO CTRL, KILIMIT, OFFSET, OUTGAIN

EXAMPLE

Program a Ka of 5000 for both axes 3 and 8.

The values of the RTC arguments are:

n	:	84h
ka_1	:	1388h
ka_3	:	1388h

CUBIC_INT

FUNCTION Start the Internal Cubic Spline Contouring Execution

DPR ORDER command code, m, si, n, ax

USAGE Host (command code: B1h), DSPL (Motion)

ARGUMENTS

m — a 16-bit unsigned word specifying the number of points in the cubic spline table to run. Each point is characterized by the position and velocity for only one motor. The maximum number of points is 2,000.

si — a 16-bit unsigned word specifying the cubic spline table starting index

0 (start at first tablepoint) <= si <= 4095

n — a 16-bit unsigned word specifying the number of times m points of a spline table will be looped over.

ax — a byte, bit coding the axis(es) involved

Note: n = 0 means run the specified number of points an infinite number of times.

DESCRIPTION

This command starts execution of the points stored in the cubic spline table immediately. It takes DSPL (or RTC) approximately 5 ms to interpret this command. After interpretation of this command, DSPL will move on to the next command line. The command sequence for this instruction is as follows:

1) run CUBIC_RATE
2) run CUBIC_SCALE ;if necessary
3) run CUBIC_INT

We assume that the user has already downloaded the table points to the cubic spline table location.

CUBIC_INT cont.

SEE ALSO CLEAR_CUBIC_TABLE, CUBIC_RATE, CUBIC_SCALE, LOAD_CUBIC_TABLE

APPLICATION

See *Cubic Spline Application Notes.*

EXAMPLE

Initiate the execution of a cubic spline contour with axes 1 and 4. The internal cubic spline table points begin at index 100, and the contour should run through the 300 points 8 times.

The values of the RTC arguments are:

m	:	012Ch
si	:	0064h
n	:	0008h
ax	:	09h

CUBIC_RATE

FUNCTION Set Cubic Spline Point Transfer Rate

DPR ORDER command code, m

USAGE Host (command code: A1h), DSPL (Motion)

ARGUMENTS

m — a 16-bit parameter coding the value for cubic spline transfer rate. "m" codes the time interval between the adjacent position points. Its value ranges between 5 and 511, and when divided by 5, it represents the interval in ms. For example, m=5 represents the time interval of 1 ms and m=25 is a 5 ms interval.

DESCRIPTION

This command sets the point transfer rate for the cubic spline. The "transfer rate" sets the interval between two adjacent points in the ring buffer or internal cubic spline table (CUBIC_INT). The two adjacent points can be spaced anywhere between 1.0 to 102.4 ms. *Mx4 Octavia*'s cubic spline interpolates between the two adjacent points at 200 µs increments. This means, for example, *Mx4 Octavia* interpolates 500 points between two adjacent points 100 ms apart.

SEE ALSO CLEAR_CUBIC_TABLE, CUBIC_INT, CUBIC_SCALE, LOAD_CUBIC_TABLE

APPLICATION

Refer to *Cubic Spline Application Notes.*

CUBIC_RATE cont.

EXAMPLE

Using cubic spline interpolation, create 16, 32, 64, and 128-point circles. The following shows the position values for 16 uniformly spaced points on a circle.

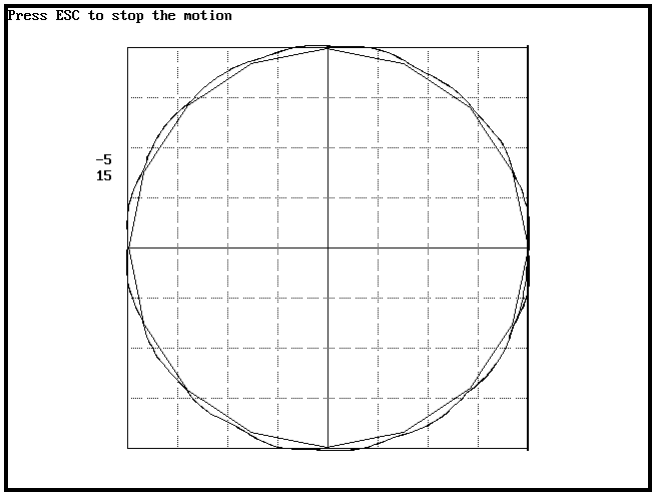
16-point Circle

Point	pos_x
p1	2500 (0x000009C4h)
p2	2310 (0x00000906h)
:	:
p16	2310 (0x00000906h)

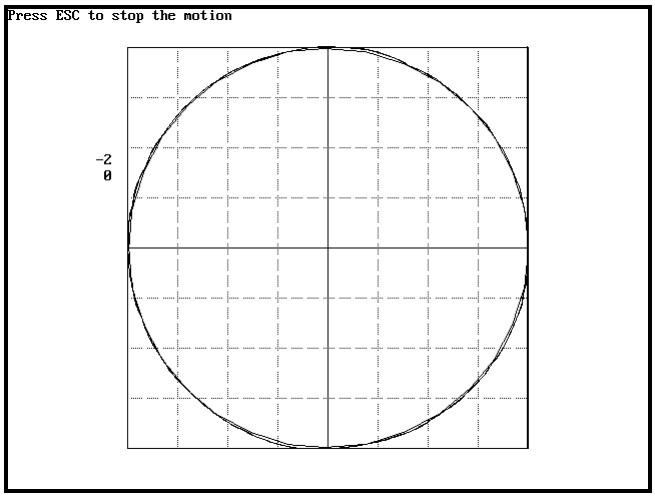
Point	pos_y
p1	0 (0x00000000h)
p2	957 (0x000003BDh)
:	:
p16	-957 (0xFFFFFC43h)

To generate a circle, these points must be loaded to *Mx4 Octavia*'s internal cubic spline table and CUBIC_RATE must be executed. The CUBIC_RATE argument determines the interval between two points of the cubic spline table. For comparison, the following figures illustrate the circles created by 16, 32, 64, and 128 points in a cubic spline interpolation. It takes 1.28 seconds to complete these circles.

CUBIC_RATE cont.

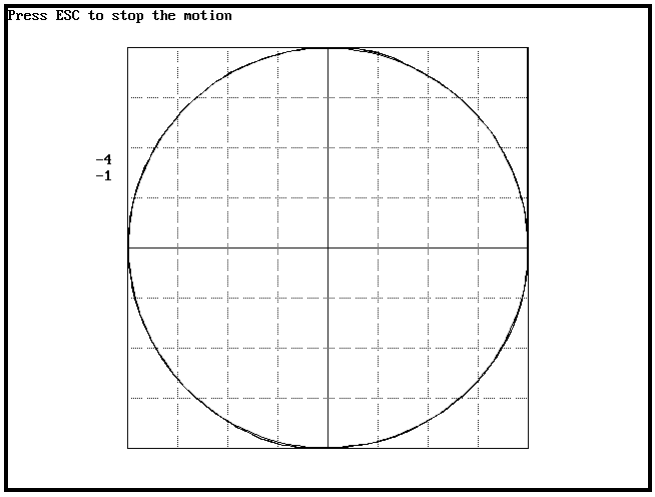


16 points; b.t. rate = 80 ms

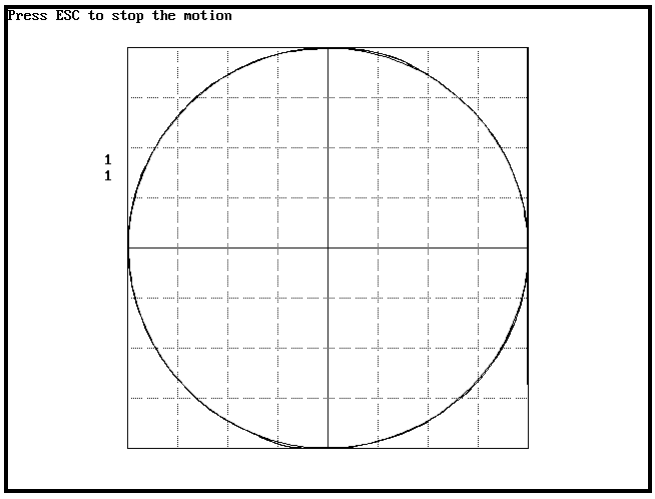


32 points; b.t. rate = 40 ms

CUBIC_RATE cont.



64 points; b.t. rate = 20 ms



128 points; b.t. rate = 10 ms

CUBIC_SCALE

FUNCTION Scale Internal Cubic Spline Data Points

DPR ORDER command code, n, pos_mult_x, pos_shift_x, ...

USAGE Host (command code: B0h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
pos_mult_x	position scaling multiplier for axis x. This is a 16-bit two's complement number with one sign bit, one integer bit, and fourteen bits fraction.
pos_shift_x	position shifter for axis x. This is a 32-bit two's complement integer number that transfers the position to a new origin.

DESCRIPTION

This command scales those table points involved in a cubic spline operation. This command also shifts the positions involved by a user-defined position shift value.

SEE ALSO CLEAR_CUBIC_TABLE, CUBIC_INT, CUBIC_RATE, LOAD_CUBIC_TABLE

APPLICATION See Cubic Spline Application Notes

EXAMPLE

Set a scale of 0.5 for all axes 2 internal cubic spline data points. No position shift is desired.

The values of the RTC arguments are:

n : 0x02h
pos_mult_2 : 0x2000h
pos_shift_2 :0x00000000h

DDAC

FUNCTION Direct DAC Output

DPR ORDER command code, n, val_1, ... , val_8

USAGE Host (command code: 63h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
val_x	16-bit value specifying 16-bit DAC output voltage for axis x

The values range as follows:

FFFFh	:	-10(1/32768)v output
:		
8000h	:	-10v output
7FFFh	:	+10v output
:		
0000h	:	0v output

DESCRIPTION

The DDAC command places the axis(es) in open loop, with DAC (x) output voltage determined by the val_x command argument. DDAC specifies a bipolar analog signal ranging from -10 to +10 volts with a resolution of approximately 0.3 millivolts.

After execution of a DDAC command, in order to return the axis(es) to closed loop operation, a closed-loop command such as AXMOVE or VELMODE must be executed. The following procedure serves as an example:

1. Slow or halt the axis(es) motion:
   -execute DDAC with 0v specified

2. Minimize built-up following error:
   -execute POS_PRESET command

DDAC cont.

3. Return axis(es) to closed loop:
   -execute AXMOVE command with target position specified as that used in the preceding POS_PRESET command

SEE ALSO none

APPLICATION

This command can be used in applications where the voltage command provides adequate control. Voltage commands can be applied to a torque loop (for torque control applications in robotics) or a velocity loop (to a spindle axis in machine tool applications).

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Output +3.7 volts to the axis 5 DAC (DAC4 *Mx4 Octavia* connector signal).

$$\left(\frac{+3.7}{10}\right) \times 7FFFh = 2F5Ch$$

The values of the RTC arguments are:

n : 10h
val_5 : 2F5Ch

DISABL_INT

FUNCTION Disable Interrupts

DPR ORDER command code, n, m_1, ... , m_4

USAGE Host (command code: 64h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
m_x	a single byte, bit mapping the interrupts to disable for axis x (setting a bit to 1 indicates disabling an interrupt)

bit 7	:	not used
bit 6	:	motion complete [EN_MOTCP]
bit 5	:	index [EN_INDEX]
bit 4	:	probe [EN_PROBE]
bit 3	:	position breakpoint [EN_POSBRK]
bit 2	:	following error [EN_ERR]
bit 1	:	following error / halt [EN_ERRHLT]
bit 0	:	buffer breakpoint [EN_BUFBRK]

DESCRIPTION

This command disables the selected enabled interrupts.

SEE ALSO DISABL2_INT, EN_BUFBRK, EN_PROBE, EN_ERRHLT, EN_ERR, EN_INDEX, EN_MOTCP, EN_POSBRK

APPLICATION

This command may be used in conjunction with all applications in which only a few interrupts are needed to be enabled. Also a few enabled interrupts may have to be disabled based on external events.

Command Sequence Example

No preparation is required before running this instruction.

DISABL_INT cont.

EXAMPLE

Disable the previously enabled axis 1 following error [EN_ERR] and axis 3 index pulse [EN_INDEX] interrupts.

The values of the RTC arguments are:

n	:	05h
m_1	:	04h
m_3	:	20h

DISABL2_INT

FUNCTION Disable Interrupts

DPR ORDER command code, n, m_1, ... , m_4

USAGE Host (command code: 5Ah), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
m_x	a single byte, bit mapping the interrupts to disable for axis x (setting a bit to 1 indicates disabling an interrupt)

bit 7	:	not used
bit 6	:	not used
bit 5	:	not used
bit 4	:	not used
bit 3	:	not used
bit 2	:	not used
bit 1	:	not used
bit 0	:	encoder fault [EN_ENCFLT]

DESCRIPTION

This command disables the selected enabled interrupts.

SEE ALSO DISABL_INT, EN_ENCFLT

APPLICATION

This command may be used in conjunction with all applications in which only a few interrupts are needed to be enabled. Also, a few enabled interrupts may have to be disabled based on external events.

Command Sequence Example

No preparation is required before running this instruction.

DISABL2_INT cont.

EXAMPLE

Disable the previously enabled axis 1, axis 4, and axis 7 encoder fault [EN_ENCFLT] interrupts.

The values of the RTC arguments are:

n	:	49h
m_1	:	01h
m_3	:	01h
m_4	:	01h

DSPL_AUTOSTART (option)

FUNCTION Start DSPL Execution at Power-Up

DPR ORDER command code, x

USAGE Host (command code: A3h)

ARGUMENTS

x — a single byte, specifying the enable/disable status of the autostart function.

x = 55h	:	autostart enabled
x <> 55h	:	autostart disabled

DESCRIPTION

The DSPL_AUTOSTART feature requires an *Mx4 Octavia* controller equipped with the optional battery-backup memory. The autostart feature allows *Mx4 Octavia* to begin DSPL program execution immediately after power-up. A DSPL program must have previously been loaded into *Mx4 Octavia*'s battery-backup memory before the DSPL_AUTOSTART command is used. Once a DSPL_AUTOSTART command has been executed by *Mx4 Octavia*, *Mx4 Octavia* will remain in the specified (enable / disable) autostart state until it executes another DSPL_AUTOSTART command; even after power-down.

The *Mx4 Octavia* (with the battery-backup memory option) is shipped from the factory with the autostart feature disabled.

SEE ALSO START_DSPL, STOP_DSPL

EN_BUFBRK

FUNCTION Enable Buffer Breakpoint Interrupt

DPR ORDER command code, buffbrk

USAGE Host (command code: 61h), DSPL (Motion)

ARGUMENTS

buffbrk — 8-bit positive value which represents the delta position for the remaining number of bytes in the contouring ring buffer. Since each point requires 8 bytes, this number must be multiplied by 8 to indicate the real number of bytes left in the contouring ring buffer.

DESCRIPTION

This command will cause an interrupt when the number of contouring data points in the ring buffer falls below a preset breakpoint. The buffer breakpoint interrupt status will appear in bit 0 of the DPR interrupt flag location 1FFEh. This bit gets set if a buffer breakpoint interrupt occurs.

SEE ALSO DISABL_INT

APPLICATION

This command must be used in both 2nd order and cubic spline contouring applications. To maintain continuity in a contouring application, *Mx4 Octavia* must be constantly updated by the host processor with a set of new (position/velocity) points on the contour. Since no application can afford to run out of points, the host must set the buffer breakpoint interrupt to a value such that running the remaining points (what is left in the ring buffer) will give the host enough time to update the buffer. For slower hosts, the argument for this command must be relatively larger.

EN_BUFBRK cont.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
.               ;load the ring buffer with contouring points,
.               ;(position and speed)
BTRATE ( )      ;set the 2nd order contouring block transfer rate to 5,
                 10, 15 or 20 ms
EN_BUFBRK ( ) ;set the breakpoint in buffer
.
.
START (n)       ;start contouring
```

EXAMPLE

Enable a contouring ring buffer breakpoint interrupt for the case that the number of segment move commands in the ring buffer falls below 30.

The value of the RTC argument is:

buffbrk : 1Eh

EN_ENCFLT

FUNCTION Encoder Fault Interrupt

DPR ORDER command code, m, fer_1, ... , fer_8

USAGE Host (command code: 58h), DSPL (Motion)

ARGUMENTS

m	a single byte, bit coding the axes interrupt condition (see Description)
n	a single byte, bit coding the axes involved
fer_x	16-bit unsigned following error for axis x

DESCRIPTION

This command enables the encoder fault interrupt for the specified axes.

With the respective axis bit of argument m equal to 0, the encoder fault interrupt is triggered for the axis in question if,

1. abs[following error] > $ferr_x$ threshold
2. and, hardware encoder status bit is set

With the respective axis bit of argument m equal to 1, the encoder fault interrupt is triggered for the axis in question if,

1. abs[following error] > $ferr_x$ threshold

If an encoder fault interrupt condition is present for an axis, the axis will be put into open loop with DAC output of 0 volts, and an interrupt will be generated. If, however, the axis in question is already in open loop prior to the interrupt condition an interrupt will be generated but no action will be taken (ie: DAC voltage is unaffected).

The encoder fault interrupt is sustiained until the EN_ENCFLT command is reissued to the *Mx4 Octavia*. Reissuing the EN_ENCFLT command also allows the affected axis(es) to be put back into closed loop following the execution of the command.

EN_ENCFLT cont.

The hardware encoder status bits are reported to the lower nibble of DPR location 113h (see *Mx4 Octavia* DPR Organization). A set bit indicates that *Mx4 Octavia* has detected an encoder hardware failure. *Mx4 Octavia* reports an "encoder status" error if for the axis in question,

1. the encoder feedback to *Mx4 Octavia* is losing encoder pulses or one of the encoder signals (A or B) actively toggles while the other one is inactive.

The DPR interrupt status locations 009h (bit 4) and 00Ah record the occurrence and source of this interrupt, respectively. Bit 6 of DPR location 1FFEh is also set.

SEE ALSO DISABL2_INT

APPLICATION

A necessary diagnostic feature for all servo control applications.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Enable the encoder fault interrupt for both axis 3 and axis 4. Set the following error threshold at 500 counts, using the encoder hardware status bits in the interrupt conditions.

The values of the RTC arguments are:

m	:	0Ch
n	:	0Ch
fer_3	:	01F4h
fer_4	:	01F4h

EN_ERR

FUNCTION Enable Following Error Interrupt

DPR ORDER command code, n, fer_1, ... , fer_8

USAGE Host (command code 67h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes for which the interrupt is enabled
fer_x	16-bit unsigned following error for axis x

DESCRIPTION

Upon the execution of this command, if at any time the following error for a specified axis exceeds its programmed value, the servo control card will generate an interrupt. This condition is recorded in DPR interrupt status register location 000h. The DPR status register location 02h will identify the axis(s) responsible. Bit 1 of DPR location 1FFEh is also set.

Note: EN_ERR is not disabled after it occurs. The host is responsible for disabling the interrupt.

SEE ALSO DISABL_INT, EN_ERRHLT

APPLICATION

This command may be used in all applications for two reasons. First, EN_ERR reports a run-away or any other out-of-control condition. Second, it makes sure that position error is within a specified (a programmed argument for EN_ERR) tolerance.

Command Sequence Example

No preparation is required before running this instruction.

EN_ERR cont.

EXAMPLE

Set a EN_ERR interrupt value of 200 encoder counts for axis 6.

The values of the RTC arguments are:

n	:	20h
fer_1	:	00C8h

EN_ERRHLT

FUNCTION Enable Following Error Interrupt and Halt

DPR ORDER command code, n, fer_1, ... , fer_8

USAGE Host (command code: 66h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes for which the interrupt is enabled
fer_x	16-bit unsigned following error for axis x

DESCRIPTION

Upon execution of this command, if at any time the following error for a specified axis exceeds its programmed value, the system will halt and generate an interrupt. The halt brings the motion of the axis in question to a stop using the programmed maximum acceleration rate. This interrupt condition is recorded in DPR interrupt status register location 000h. The DPR status register location 001h reveals the axis(es) responsible. Bit 1 of DPR location 1FFEh is also set.

Note 1: EN_ERRHLT will be ignored if the respective axis abort maximum acceleration is zero.

Note 2: EN_ERRHLT is not disabled after it occurs. The host is responsible for disabling the interrupt.

SEE ALSO DISABL_INT, EN_ERR, ESTOP_ACC

EN_ERRHLT cont.

APPLICATION

Applications of this command are similar to EN_ERR. However, as a result of this command's interrupt, the system will come to a stop. Stop trajectory uses the programmed abort maximum acceleration. Please see ESTOP_ACC. Please note that this command is not appropriate to prevent system run-away in case of encoder loss - since in the absence of encoder, the system cannot be stopped reliably.

Command Sequence Example

```
ESTOP_ACC ( ) ;set the maximum accel. so system can be stopped
CTRL ( )      ;these instructions enable system to stop motion
KILIMIT ( )   ;set gains
.
.
EN_ERRHLT ( )
```

EXAMPLE

Enable a following error/halt interrupt for axis 3 with a threshold of 100 encoder counts.

The values of the RTC arguments are:

n	:	04h
fer_3	:	0064h

EN_INDEX

FUNCTION Enable Index Pulse Interrupt

DPR ORDER command code, n

USAGE Host (command code: 69h), DSPL (Motion)

ARGUMENTS

n — a single byte, bit coding the only axis for which the interrupt is enabled

DESCRIPTION

Upon the execution of this command, the servo control card will search for the first index pulse edge from the specified axis. The pulse edge generates an interrupt and registers the actual position for all axes in DPR locations 103h-112h / 903h-912h. The DPR interrupt status register locations 000h and 003h record the occurrence and source of this interrupt. Bit 1 of DPR location 1FFEh is also set.

Note 1: Only one index pulse can generate an interrupt at any given time. The EN_INDEX command enables the index pulse interrupt for the axis specified and automatically disables the previous one (if any).

Note 2: The EN_INDEX and EN_PROBE commands CAN BE ENABLED simultaneously.

SEE ALSO DISABL_INT, POS_PRESET, POS_SHIFT

APPLICATION

This command is used in homing applications. As a result of this instruction, *Mx4 Octavia* will start searching for the first index pulse edge. Upon the detection of an index pulse edge, position of the axis is immediately recorded. This instruction must be used in conjunction with POS_PRESET to perform homing for linear table (or other index-based) position calibration.

EN_INDEX cont.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Enable the index pulse interrupt for axis 8.

The value of the RTC argument is:

n : 80h

EN_MOTCP

FUNCTION Enable Motion Complete Interrupt

DPR ORDER command code, n

USAGE Host (command code: 65h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes for which the interrupt is enabled

DESCRIPTION

This command enables the motion complete interrupt for the axes specified. The motion complete interrupt is generated when any closed loop motion other than ring buffer 2nd order or ring buffer cubic spline contouring comes to a stop. The DPR interrupt status register locations 000h and 005h record the occurrence and source of this interrupt. Bit 1 of DPR location 1FFEh is also set.

Note: EN_MOTCP is not disabled after it occurs. The host is responsible for disabling the interrupt.

SEE ALSO DISABL_INT

APPLICATION

In any application that a new routine must run based on the end of a motion, this command informs the host of motion completion. An example of such an application is milling in which the spindle and z axes will start moving only when the x-y table has moved to a target position.

Command Sequence Example

See AXMOVE and STOP

EN_MOTCP cont.

EXAMPLE

Enable the motion complete interrupt for all eight axes.

The value of the RTC argument is:

n : FFh

EN_POSBRK

FUNCTION Enable Position Breakpoint Interrupt

DPR ORDER command code, n, pos_1, ... , pos_8

USAGE Host (command code: 6Bh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes for which the interrupt is enabled
pos_x	32-bit two's complement position breakpoint value for axis x

DESCRIPTION

This command enables the position breakpoint interrupt for the axes specified. The position breakpoint interrupt is generated when the actual position, for a specified axis, passes the programmed breakpoint. The DPR interrupt status register locations 000h and 004h record the occurrence and source of this interrupt. Bit 1 of DPR location 1FFEh is also set.

Note 1: The position breakpoint is calculated as the absolute distance from the present position (position at the moment at which the EN_POSBRK RTC is interpreted) to the position breakpoint value entered. The breakpoint interrupt is set when the axis in question travels (in either direction) a distance equal to the calculated absolute distance.

Note 2: EN_POSBRK is automatically disabled after the breakpoint interrupt is generated. To activate this interrupt again, the host must issue a new EN_POSBRK command.

Note 3: POS_PRESET and POS_SHIFT will automatically disable the position breakpoint interrupt. The user is responsible to re-enable the interrupt again.

SEE ALSO DISABL_INT, POS_PRESET, POS_SHIFT

EN_POSBRK cont.

APPLICATION

This instruction may be used in applications such as robotics, indexing machine tools, etc. The CPU must be notified that the system has passed an intermediate position. Based on this interrupt, the CPU will execute a task. For example, in a robotics painting application, the paint mixture may have to change based on the robot's arm location.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
OUTGAIN ( )
```

EXAMPLE

Enable a breakpoint interrupt with a value of 60,000 counts for axis 1 and 500,000 for axis 5.

The values of the RTC arguments are:

n	:	11h
pos_1	:	0000EA60h
pos_2	:	0007A120h

EN_PROBE

FUNCTION Enable General Purpose External Interrupt

DPR ORDER command code, n

USAGE Host (command code: 6Ch), DSPL (Motion)

ARGUMENTS

m — a single byte, bit coding of the only *EXTx input signal enabled

m=1	:	from *EXT1
m=2	:	from *EXT2
m=3	:	from *EXT3
m=4	:	from *EXT4

DESCRIPTION

Upon the execution of this command, the servo control card will search for the first *EXTx pulse edge. The pulse edge generates an interrupt, and registers the actual position for all axes in DPR locations 0A7h-0B6h / 8A7h-8B6h. (The hand shaking bytes are 0C8h and 0D0h for *Mx4 Octavia* and host, respectively.) DPR interrupt status register locations 000h and 006h record the occurrence and source of this interrupt. Bit 1 of DPR location 1FFEh is also set.

Note 1: Only one general-purpose external interrupt can generate an interrupt at any given time. The EN_PROBE command enables the external interrupt specified and automatically disables the previous one (if any).

Note 2: The EN_PROBE and EN_INDEX CAN BE ENABLED simultaneously.

SEE ALSO DISABL_INT

EN_PROBE cont.

APPLICATION

This instruction is useful in probing applications. Since EN_PROBE registers all positions when an interrupt occurs (falling pulse edge is detected), it can be used in accurate recording of surface dimensions by a probe.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Enable the *EXT4 external interrupt.

The values of the RTC arguments are:

n : 08h

ESTOP_ACC

FUNCTION Abort Motion Maximum Acceleration

DPR ORDER command code, n, acc_1, ... , acc_8

USAGE Host (command code: 86h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes for which the interrupt is enabled
acc_x	16-bit unsigned value specifying the maximum halting acceleration (deceleration) for axis x

Note: Acceleration is partitioned into 1 bit integer, 15 bits fraction.

DESCRIPTION

This command specifies the maximum halting acceleration (deceleration) for the axes specified. The maximum acceleration values are used in the following cases: EN_ERRHLT and ESTOP_ACC.

Note: ESTOP_ACC will be ignored if the specified argument is zero.

SEE ALSO EN_ERRHLT, MAXACC, STOP, VELMODE

APPLICATION

This command sets the maximum possible deceleration for a mechanical actuator. This RTC is used to set the deceleration rate for an emergency case. In contrast to MAXACC, ESTOP_ACC provides a sharper deceleration such that the entire system comes to a stop as rapidly as possible. Please remember that the STOP and VELMODE RTCs use MAXACC for their acceleration/deceleration.

ESTOP_ACC cont.

Command Sequence Example

```
ESTOP_ACC ( ) ;set the abort maximum acceleration
CTRL ( )      ;make sure the system is in closed loop
EN_ERRHLT ( ) ;set the maximum tolerance for the following error
              ;if the following error exceeds the ABORTACC
              ;parameter, the system will stop immediately
```

EXAMPLE

Set an abort motion maximum acceleration for axes 2 and 3 of 0.5 encoder counts/200 μsec^2.

(0.5) x 215 = 4000h

The values of the RTC argument are:

n	:	06h
acc_2	:	4000h
acc_3	:	4000h

GEAR

FUNCTION Electronics Gear On

DPR ORDER command code, n, m, r_1, ... , r_8

USAGE Host (command code: 09Ch), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the ONLY axis as LEADER gear
m	a single byte, bit coding the axes as FOLLOWER gears
r_x	a single word; it is a two's complement gear ratio between leader and follower; it has 1 sign bit, 8 integer bit and 7 bits fraction

DESCRIPTION

This command emulates the mechanical gear function with gear ratio range -256 to 255.999. The follower follows the leader with the gear ratio specified by r_x. Upon receiving this command, the electronic gearing is engaged at once.

SEE ALSO GEAR_OFF, GEAR_POS, GEAR_PROBE

APPLICATION

See *DSPL Programmer's Guide, Application Notes*

EXAMPLE

Slave axis 5 to axis 1 with a gear ratio of 2.

The values of the RTC arguments are:

n	:	01h
m	:	10h
r_2	:	2000h

GEAR_OFF

FUNCTION Electronics Gear Off

DPR ORDER command code, n

USAGE Host (command code: 09Fh), DSPL (Motion)

ARGUMENTS

n a single byte, bit coding the axes as FOLLOWER gear to be disengaged

DESCRIPTION

This command disengages the specified follower axes at once.

SEE ALSO GEAR, GEAR_POS, GEAR_PROBE

APPLICATION

See *DSPL Programmer's Guide, Application Notes*

EXAMPLE

Axis 1 is the leader, axis 4 and 5 are the FOLLOWERS. Disengage only axis 5

The value of the RTC argument is:

n : 10h

GEAR_OFF_ACC

FUNCTION Turns Electronic Gearing Off and Halt Slave(s)

DPR ORDER command code, n,

USAGE Host (command code: A0h), DSPL (Motion)

ARGUMENTS

n a single byte, bit coding the axis to be disengaged

DESCRIPTION

This command disengages the system that was under master slave control. The slave axes will come to a complete stop at the maximum acceleration rate specified by MAXACC command.

SEE ALSO GEAR, GEAR_OFF, GEAR_POS, GEAR_PROBE, SYNC

APPLICATION

Axis 1 is the leader, axis 3 and 4 are the followers (slaves). Disengage only axis 4.

The value of the RTC argument is:

n : 08h

GEAR_POS

FUNCTION Electronics Gear On at a Specified Leader Position

DPR ORDER command code, n, m, r_1, tp_1, ... , r_8, tp_8

USAGE Host (command code: 09Dh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the ONLY axis as LEADER gear
m	a single byte, bit coding the axes as FOLLOWER gears
r_x	a single word; it is a two's complement gear ratio between leader and follower; it has 1 sign bit, 8 integer bit and 7 bits fraction
tp_x	a 32-bit two's complement that specifies the leader position where the electronic gear starts engaging

DESCRIPTION

This command emulates a mechanical gear function with gear ratio range -256 to 255.999. The follower follows the leader with the gear ratio specified by r_x. Upon receiving this command, the electronic gearing starts engaging at the specified master position (tp_x).

SEE ALSO GEAR, GEAR_OFF, GEAR_PROBE

APPLICATION

See *DSPL Programmer's Guide, Application Notes*

GEAR_POS cont.

EXAMPLE

Axis 2 is the leader, axis 7 has the gear ratio of 2 to 1. Axis 7 starts following axis 1 at 1000 counts.

The values of the RTC arguments are:

n	:	02h
m	:	40h
r_7	:	2000h
tp_7	:	000003E8h

GEAR_PROBE

FUNCTION Electronics Gear On After Probe Input

DPR ORDER command code, n, m, q, r_1, ... , r_8

USAGE Host (command code: 09Eh), DSPL (Motion)

ARGUMENTS

n — a single byte, bit coding the ONLY axis as LEADER gear

m — a single byte, bit coding the axes as FOLLOWER gears

q — a single byte, specifying the *EXTx probe input to be used

q = 01h	:	*EXT1
q = 02h	:	*EXT2
q = 04h	:	*EXT3
q = 08h	:	*EXT4

r_x — a single word; it is a two's complement gear ratio between leader and follower; it has 1 sign bit, 8 integer bit and 7 bits fraction

DESCRIPTION

This command emulates the mechanical gear function with gear ratio range -256 to 255.999. The follower follows the leader with the gear ratio specified by r_x. The GEAR_PROBE command engages the mechanical gear function for selected master and slave axes after an external signal (*EXT1-4) is activated.

Note 1: Execution of the GEAR_PROBE command will disable any previously enabled EN_PROBE interrupt. Probe input (EXT1-4) activation does ***not*** generate an interrupt with the GEAR_PROBE command.

GEAR_PROBE cont.

Note 2: Activation of *ESTOP during a GEAR operation will halt the master axis, and subsequently the slave axis(es). Slave(s) remain "engaged" in GEAR mode after the input-triggered halt.

SEE ALSO GEAR, GEAR_OFF, GEAR_POS

APPLICATION

See *DSPL Programmer's Guide, Application Notes*

EXAMPLE

Axis 4 is the leader, axis 5 has the gear ratio of 2.5 to 1. Engage gearing upon *EXT4 active.

The values of the RTC arguments are:

n	:	08h
m	:	10h
q	:	20h
r_2	:	2800h

INP_STATE

FUNCTION Configure Logic State of Inputs

DPR ORDER command code, inp_1, inp_2

USAGE Host (command code: B4h), DSPL (Motion)

ARGUMENTS

inp_1 a single word, coding the logic state of inputs

bit = 0	:	active LOW input
bit = 1	:	active HIGH input
bit 15	:	IN15
bit 14	:	IN14
bit 13	:	IN13
bit 12	:	IN12
bit 11	:	IN11
bit 10	:	IN10
bit 9	:	IN9
bit 8	:	IN8
bit 7	:	IN7
bit 6	:	IN6
bit 5	:	IN5
bit 4	:	IN4
bit 3	:	IN3
bit 2	:	IN2
bit 1	:	IN1
bit 0	:	IN0

INP_STATE cont.

inp_2	a single word, coding the logic state of inputs		
	bit = 0	:	active LOW input
	bit = 1	:	active HIGH input
	bit 15	:	IN31
	bit 14	:	IN30
	bit 13	:	IN29
	bit 12	:	IN28
	bit 11	:	IN27
	bit 10	:	IN26
	bit 9	:	IN25
	bit 8	:	IN24
	bit 7	:	IN23
	bit 6	:	IN22
	bit 5	:	IN21
	bit 4	:	IN20
	bit 3	:	IN19
	bit 2	:	IN18
	bit 1	:	IN17
	bit 0	:	IN16

DESCRIPTION

This command allows the user to define the logic state of the Mx4 Octavia inputs. Each input may be configured as active LOW or active HIGH (TTL logic levels) (the Mx4 Octavia inputs are level sensitive).

Note: At power-up and reset, Mx4 Octavia inputs default as active LOW.

SEE ALSO EN_INP

EXAMPLE

Configure the IN0-IN3 inputs as active HIGH inputs. The remaining inputs are to be configured as active LOW.

The value of the RTC arguments is:

inp_1 : 000Fh
inp_2 : 0000h

INT5MS

FUNCTION 5ms Interrupt

DPR ORDER command code, m

USAGE Host (command code: 8Eh)

ARGUMENTS

m — a byte that selects the enable / disable status of the 5ms interrupt

m = 0	:	disable 5ms interrupt
m < > 0	:	enable 5ms interrupt

DESCRIPTION

The INT5MS command specifies the enable/disable status of Mx4 Octavia's 5ms interrupt. The 5ms interrupt, when enabled, generates a host interrupt every 5ms. The interrupt is coded in DPR location 009h. Bit 6 of DPR location 1FFEh is also set.

The INT5MS interrupt is useful in applications where the host requires synchronization with the Mx4 Octavia controller for the purpose of reading data from Mx4 Octavia's DPR.

SEE ALSO none

APPLICATION

EXAMPLE

Enable the 5ms timer interrupt.

The value of the RTC arguments is:

m : 01h

KILIMIT

FUNCTION Integral Gain Limit

DPR ORDER command code, n, val_1, ... , val_8

USAGE Host (command code: 74h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
val_x	a single byte value specifying the limit of the integral action for each axis

Note: $0 \leq val \leq 14$

val = 0 indicates no limit on integration channels
val = 14 indicates maximum limit on integration channels

For example,

Kilimit val = 0 +/- 10v DAC action from K_i control law parameter
Kilimit val = 1 +/- 5v DAC action from K_i control law parameter
Kilimit val = 2 +/- 2.5v DAC action from K_i control law parameter
Kilimit val = 3 +/- 1.25v DAC action from K_i control law parameter
:
:

DESCRIPTION

This command is used to set the limit for integral action related to the choice of par_{x1} in the CTRL RTC. Integral limit is specified for each axis. Default val_x are set to zero (i.e., no limit on integration channels).

SEE ALSO CTRL

KILIMIT cont.

APPLICATION

This command clamps the integral channel by reducing this channel's saturation level. Reducing the saturation level will reduce the channel's depletion time. Using this instruction is essential where large integral gain is required. Clamping the integral channel will let the system zero position error without a lengthy "creeping motion" to its target position.

Command Sequence Example

```
CTRL ( )        ;set the gains
KILIMIT ( )     ;this instruction may be used before or after CTRL
```

EXAMPLE

Set a maximum limit on the integral action of axis 6.

The values of the RTC arguments are:

n	=	20h
val_6	=	0Eh

LOAD_CAM_TABLE

FUNCTION Load Cam Table

DPR ORDER command code, index, numpts

USAGE Host (command code: AAh)

ARGUMENTS

index — 16-bit unsigned value, indicating the cam table index to which data download begins

$0 \leq \text{index} \leq 1600$

numpts — 16-bit unsigned value, indicating the number of cam data points to download

$1 \leq \text{numpts} \leq 67$

Note: If the number of cam data points to download exceeds 67, the data must be split and downloaded in separate LOAD_CAM_TABLE executions.

DESCRIPTION

The LOAD_CAM_TABLE command is used to download cam data points to the cam table. Each cam data point consists of 5 words;

2 words for master axis position
2 words for slave axis position
1 word for gear ratio

Note that a gear ratio is required with the `LOAD_CAM_TABLE` method of downloading cam table data. With the other two methods, downloading a table from within the Mx4pro development tool and downloading individual cam points with the `CAM_POINT` command, there is no mention of a gear ratio value. This is because those routines have the gear ratio calculation built into their functionlity.

LOAD_CAM_TABLE cont.

The gear ratio format is 1 sign bit, 8 integer bits, and 7 bits fraction. This yields a possible range of gear ratios -256 to 255.99, with a minimum resolution of 0.0078 (or 1/2^7 = 1/128).

The first point of a cam table (index = 0) must have master axis position = slave axis position = 0. The last point of a cam table (index = tablesize-1) must have master axis position = master cycle length and slave axis position = slave cycle length. Gear ratios are calculated as follows,

$$\text{gearratio} = \frac{[\text{slaveaxisposition}(x+1) - \text{slaveaxisposition}(x)]}{[\text{masteraxisposition}(x+1) - \text{masteraxisposition}(x)]}$$

For example, consider the following table of master pos/slave pos pairs.

master	slave	gear ratio
0	0	[200-0]/[100-0] = 2.0
100	200	[300-200]/[500-100] = 0.25
500	300	[200-300]/[1000-500] = -0.2
1000	200	0

Note that the gear ratio associated with the last point is always 0. The above example cam table data constitutes a 4-point cam data table. To convert the decimal gear ratios above to Mx4 units, simply multiply by 2^7 or 128. For example, a gear ratio of 0.25 would be represented in Mx4 format as 0.25 * 128 = 32 = 0x0020h.

The LOAD_CAM_TABLE command utilizes the Dual Port RAM Contouring Ring Buffer (120h - 3BFh) in order to download data to the table; thus, LOAD_CAM_TABLE ***cannot*** be used in conjunction with ring buffer-based contouring motion.

LOAD_CAM_TABLE cont.

The proper sequence to follow in order to download a cam table:

1. Load the 5-word cam data points to the Dual Port RAM Contouring Ring Buffer (master pos, slave pos, gear ratio) (low byte of low word first, etc.), beginning at location 120h. The buffer is capable of holding 67 5-word cam data points.

2. Execute a LOAD_CAM_TABLE command, specifying the table index and the number of 5-word cam data points being downloaded.

3. Repeat Steps 1 and 2 as necessary to download all of the desired cam data points.

Note: Cam, internal cubic spline position/velocity compensation, and DSPL variable tables share overlapping data space in Mx4 Octavia. These table features should not be used in conjunction with each other without first reviewing *Chapter 8: Mx4 Octavia RAM Memory Organization.*

SEE ALSO CAM, CAM_OFF, CAM_OFF_ACC, CAM_POINT, CAM_POS, CAM_PROBE

APPLICATION

General master/slaving, in particular in packaging, synchronous cutting, flying shear, and mark registration, requires the coordination of several axes in cam fashion. For these applications, the user is required to load the cam function along with the position spacing that defines the distance between the adjacent gear ratios stored in the cam table. Issuing cam starts the engagement.

LOAD_CUBIC_TABLE

FUNCTION Load Internal Cubic Spline Data Table

DPR ORDER command code, index, num

USAGE Host (command code: AFh)

ARGUMENTS

index — a word, indicating the internal cubic spline table index to which data download starts

$0 \leq \text{index} \leq 4095$

num — a word, indicating the number of 2-word (32-bit position) contouring data points to download

$1 \leq \text{num} \leq 168$

DESCRIPTION

The LOAD_CUBIC_TABLE command is used to download cubic spline contouring data points to the Mx4 Octavia internal cubic spline table.

The LOAD_CUBIC_TABLE command utilizes the Dual Port RAM Contouring Ring Buffer (120h - 3BFh) in order to download data to the table; thus LOAD_CUBIC_TABLE ***cannot*** be used in conjunction with ring buffer-based contouring motion.

The proper sequence to follow in order to download a cubic spline data table:

1. Load 32-bit position cubic spline contouring data points to the Dual Port RAM Contouring Ring Buffer (low byte first), beginning at location 120h. The buffer is capable of holding 168 cubic spline data points.

LOAD_CUBIC_TABLE cont.

2. Execute a LOAD_CUBIC_TABLE command, specifying table index and the number of cubic spline data points being downloaded.

3. Repeat Steps 1 and 2 until the complete 4096-point table is downloaded. (Partial tables may be downloaded as well.)

Note: Cam, internal cubic spline position/velocity compensation, and DSPL variable tables share overlapping data space in Mx4 Octavia. These table features should not be used in conjunction with each other without first reviewing *Chapter 8: Mx4 Octavia RAM Memory Organization.*

SEE ALSO CLEAR_CUBIC_TABLE, CUBIC_INT, CUBIC_RATE, CUBIC_SCALE

LOAD_POS_TABLE

FUNCTION Load Position Compensation Table

DPR ORDER command code, table, index, num

USAGE Host (command code: 9Ah)

ARGUMENTS

table — a single byte, bit coding the position compensation table to download to

index — a word, indicating the table location to which data download starts

$0 \leq \text{index} \leq 1023$

num — a word, indicating the number of 16-bit table entries to download

$1 \leq \text{num} \leq 336$

DESCRIPTION

The LOAD_POS_TABLE command is used to download position compensation data to a specified compensation table. (See *Table Compensation Application Notes* for position compensation data point format information.)

The LOAD_POS_TABLE command utilizes the Dual Port RAM Contouring Ring Buffer (120h - 3BFh) in order to download data to the table; thus, LOAD_POS_TABLE cannot be used in conjunction with contouring motion.

LOAD_POS_TABLE cont.

The proper sequence to follow in order to download a cam table:

1. Load 16-bit compensation data points to the Dual Port RAM contouring Ring Buffer (low byte first), beginning at location 120h. The buffer is capable of holding 336 16-bit data points being downloaded.

2. Execute a LOAD_POS_TABLE command, specifying the table number, table index and the number of 16-bit data points being downloaded.

3. Repeat Steps 1 and 2 until the complete 1024-point table is downloaded. (Partial tables may be downloaded as well.)

Note: Cam, internal cubic spline position/velocity compensation, and DSPL variable tables share overlapping data space in Mx4 Octavia. These table features should not be used in conjunction with each other without first reviewing *Chapter 8: Mx4 Octavia RAM Memory Organization.*

SEE ALSO CIRCLE, CLEAR_POS_TABLE, TABLE_SEL, TABLE_OFF, TABLE_ON (*DSPL Programmer's Guide*)

LOAD_VEL_TABLE

FUNCTION Load Velocity Compensation Table

DPR ORDER command code, table, index, num

USAGE Host (command code: 9Bh)

ARGUMENTS

table — a single byte, bit coding the position compensation table to download to

index — a word, indicating the table location to which data download starts

$0 \leq \text{index} \leq 1023$

num — a word, indicating the number of 16-bit table entries to download

$1 \leq \text{num} \leq 336$

DESCRIPTION

The LOAD_VEL_TABLE command is used to download velocity compensation data to a specified compensation table. (See *Table Compensation Application Notes* for velocity compensation data point format information.)

The LOAD_VEL_TABLE command utilizes the Dual Port RAM Contouring Ring Buffer (120h - 3BFh) in order to download data to the table; thus, LOAD_VEL_TABLE cannot be used in conjunction with contouring motion.

LOAD_VEL_TABLE cont.

The proper sequence to follow in order to download a vel table:

1. Load 16-bit compensation data points to the Dual Port RAM contouring Ring Buffer (low byte first), beginning at location 120h. The buffer is capable of holding 336 16-bit data points being downloaded.

2. Execute a LOAD_VEL_TABLE command, specifying the table number, table index, and the number of 16-bit data points being downloaded.

3. Repeat Steps 1 and 2 until the complete 1024-point table is downloaded. (Partial tables may be downloaded as well.)

Note: Cam, internal cubic spline position/velocity compensation, and DSPL variable tables share overlapping data space in Mx4 Octavia. These table features should not be used in conjunction with each other without first reviewing *Chapter 8: Mx4 Octavia RAM Memory Organization.*

SEE ALSO CIRCLE, CLEAR_VEL_TABLE, TABLE_SEL, TABLE_OFF, TABLE_ON (*DSPL Programmer's Guide*)

LOW_PASS (option)

FUNCTION Implement Low Pass Filter at Controller Output

DPR ORDER command code, n, $freq_x$

USAGE Host (command code: 8Eh*), DSPL (Motion)

Note: This RTC code (8Eh) is the same as the one used with NOTCH, therefore one option (either LOW_PASS or NOTCH) can be used at any time.

ARGUMENTS

n	bit coding of the only specified axis
$freq_x$	unsigned value specifying the low pass filter cut-off frequency for axis x

$0 \leq freq_x \leq 1850$

DESCRIPTION

This command implements a low pass filter at the controller output for the specified axis.

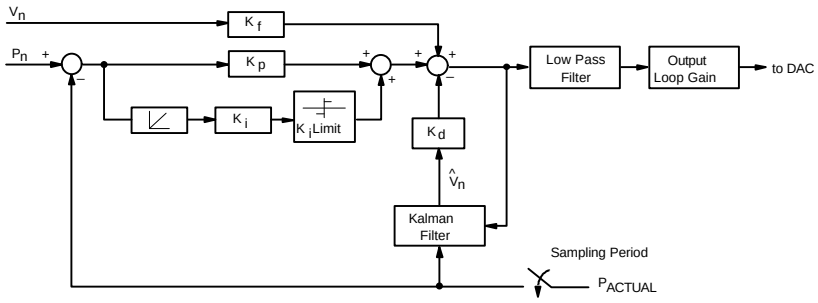


Fig. 4-2: Mx4 Octavia Block Diagram with Low Pass Filter

LOW_PASS cont.

The low pass filter implements the following transfer function:

$$G(s) = \frac{w_n^2}{s^2 + 2zw_n \cdot s + w_n^2}$$

where, $w_n = 2pf_n$, f_n = cut-off frequency, and $z = 0.6$

The frequency and bandwidth of the low pass filter is programmable.

Note: By programming a cut-off frequency of 0, the low pass filter for the specified axis is disabled.

SEE ALSO none

EXAMPLE: DSPL Programming Low Pass

1) Set a low pass filter at 250 Hz for axis 2 (see below).

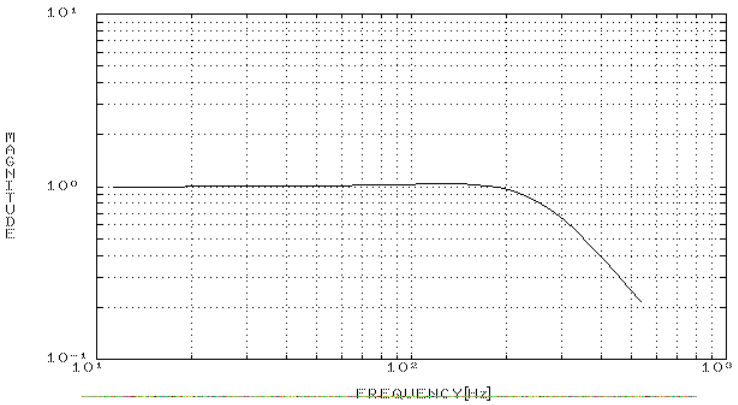
   LOW_PASS (0x2,250)

2) Disable the low pass filter of axis 1.

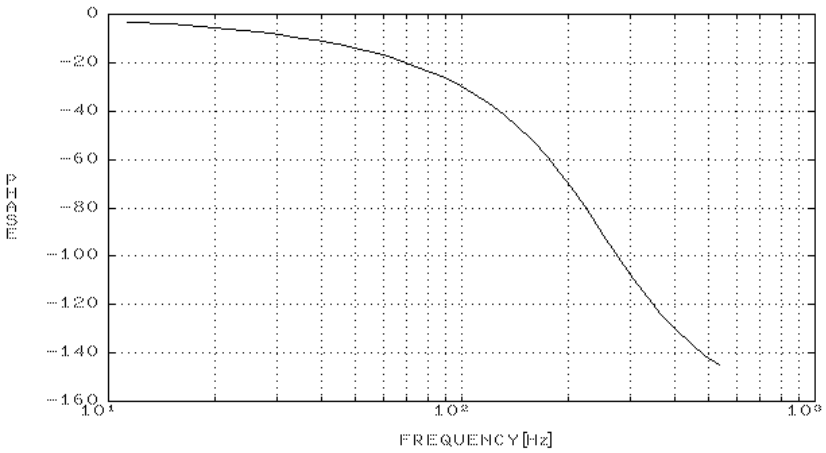
   LOW_PASS (0x1,0)

Note: Mx4 Octavia default setting for low pass filter is no filter (or filter disabled).

LOW_PASS cont.



Magnitude Diagram



Phase Diagram of 250 Hz Low Pass Filter

LOW_PASS cont.

EXAMPLE: RTC Programming Low Pass

The LOW_PASS RTC uses the coded values for low pass frequency. Table 4-1 shows these coded values. Use of the index table is only necessary with RTCs.

Set a low pass filter at 275 Hz for axis 3.

The following shows the DPR's byte stream:

```
3c2   xxh    ;command code
3c3   xxh    ;axis 3
3c4   0Ah    ;index to element 10 of frequency table (275 Hz)
```

FREQ (Hz)	FREQ Index
disable filter	0
50	1
75	2
100	3
125	4
150	5
175	6
200	7
225	8
250	9
275	10
300	11
325	12
350	13
375	14
400	15
425	16
450	17
475	18
500	19
550	20
600	21
650	22
700	23

FREQ (Hz)	FREQ Index
750	24
800	25
850	26
900	27
950	28
1000	29
1050	30
1100	31
1150	32
1200	33
1250	34
1300	35
1350	36
1400	37
1450	38
1500	39
1550	40
1600	41
1650	42
1700	43
1750	44
1800	45
1850	46

Low Pass Filter Frequency Index

MAXACC

FUNCTION Maximum Acceleration

DPR ORDER command code, n, acc_1, ... , acc_8

USAGE Host (command code: 71h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit unsigned value specifying the maximum acceleration / deceleration for axis x

Note: Acceleration is partitioned into 1 bit integer, 15 bits fraction.

DESCRIPTION

This command specifies the maximum acceleration / deceleration for the axes specified. The maximum acceleration values are used in the STOP and VELMODE commands.

Note: MAXACC will be ignored if the specified argument is zero.

SEE ALSO ESTOP_ACC, STOP, VELMODE

APPLICATION

This command sets the maximum acceleration affordable by the servo drive and motor combination. It is useful to program this parameter such that the system will not go to control saturation during VELMODE or STOP.

MAXACC cont.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
.
.
AXMOVE ( )      ;run system in axis move
VELMODE ( )     ;run system in velocity mode
```

EXAMPLE

Set a maximum acceleration for axes 2, 3, 7, and 8 of 0.25 encoder counts / $(200\mu s)^2$.

$$(0.25) \times 2^{15} = 2000h$$

The values of the RTC arguments are:

n	:	C6h
acc_2	:	2000h
acc_3	:	2000h
acc_7	:	2000h
acc_8	:	2000h

MONITOR_VAR

FUNCTION Host Monitoring of DSPL Variables (VAR1-128)

DPR ORDER command code, m, $varnum_1$, ... , $varnum_4$

USAGE Host (command code: B2h), DSPL (Motion)

ARGUMENTS

m — a single byte, bit mapping the DPR variable update location

bit 7
: not used
bit 4
bit 3 variable monitor 4 (DPR 078h-07Dh)
bit 2 variable monitor 3 (DPR 072h-077h)
bit 1 variable monitor 2 (DPR 06Ch-071h)
bit 0 variable monitor 1 (DPR 066h-06Bh)

$varnum_x$ — a single byte, the value of which codes the variable number (1-128) to monitor

DESCRIPTION

This command allows the host to monitor selected DSPL variables in the DSPL updates block of the DPR (066h - 07Dh). The variables are reported to the DPR in 48-bit DSPL floating point format.

Note: DSPCG provides C-code conversion utilities for converting between C float formats and DSPCG DSPL float format. (See Mx4 Octavia Utilities diskette)

SEE ALSO CHANGE_VAR

APPLICATION

See *DSPL Programmer's Guide, DSPL Variables & Tables.*

MONITOR_VAR cont.

EXAMPLE

Select DSPL variables VAR3, VAR12, VAR15, and VAR55 to be reported to the DPR.

The values of the RTC arguments are:

m	:	0Fh
$varnum_1$	:	03h
$varnum_2$	:	0Ch
$varnum_3$	:	0Fh
$varnum_4$	:	37h

NOTCH (option)

FUNCTION Implement Notch Filter at Controller Output

DPR ORDER command code, n, $freq_x$, q_x

USAGE Host (command code: 8Eh*), DSPL (Motion)

Note: This RTC code (8Eh) is the same as the one used with LOW_PASS, therefore one option (either NOTCH or LOW_PASS) can be used at any time.

ARGUMENTS

n — a single byte, bit coding the single specified axis

$freq_x$ — 16 bit unsigned value specifying the notch filter frequency for axis x

$0 \leq freq_x \leq 1544$ Hz

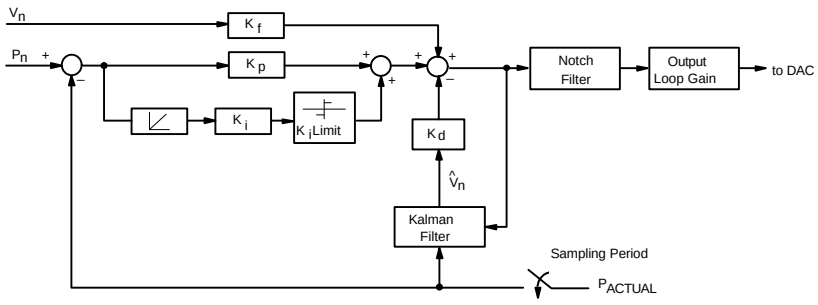
q_x — a single byte, specifying the notch filter quality factor for axis x

q_x = 00h ~25% bandwidth filter
q_x = 01h ~10% bandwidth filter

NOTCH cont.

DESCRIPTION

This command implements a notch filter at the controller output for the specified axis.



Mx4 Octavia Block Diagram with Notch Filter

The notch filter implements the transfer function:

$$G(s) = \frac{s^2 + w_n^2}{s^2 + \frac{w_n}{Q}s + w_n^2}$$

where, $w_n = 2pf_n$ and f_n = notch frequency

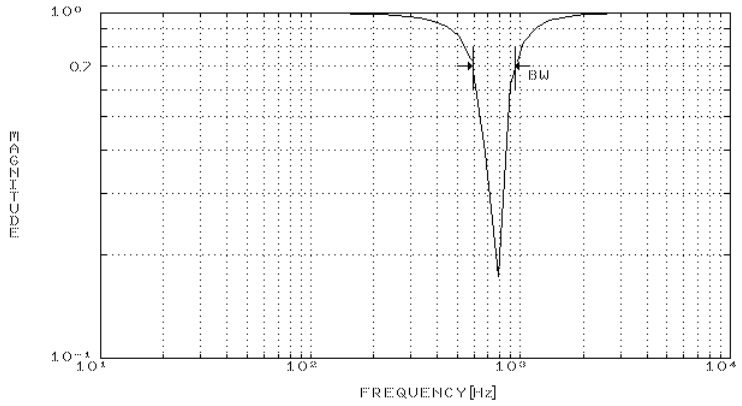
The frequency and bandwidth of the notch is programmable.

Note: By programming a notch frequency of 0, the notch filter for the specified axis is disabled.

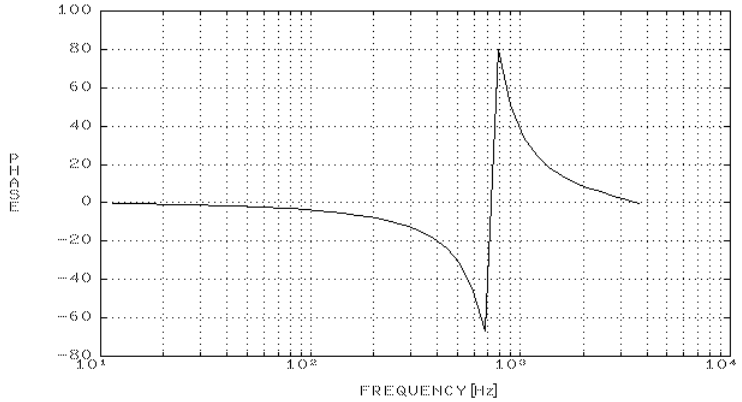
SEE ALSO none

NOTCH cont.

Note: The Mx4 Octavia default setting for notch filter is no notch (or notch disabled).



(a)



(b)

Frequency Response of Discrete 750 Hz, Q=2 Notch Filter

NOTCH cont.

EXAMPLE

Set a notch filter at 300Hz and a q factor of 10% for axis 2.

The values of the RTC arguments are:

n	:	02h
$freq_2$	:	012Ch
q_2	:	01h

OFFSET

FUNCTION Amplifier Offset Cancellation

DPR ORDER command code, n

USAGE Host (command code: 5Fh), DSPL (Motion)

ARGUMENTS

n a single byte, bit coding the ONLY axis involved

DESCRIPTION

This command minimizes the offset generated by the D/A Converter (DAC). Upon completion of offset tuning, an interrupt is generated to the host. The condition is recorded in DPR interrupt status register location 009h. DPR status register location 00Ch will identify the axis responsible. Bit 6 of DPR location 1FFEh is also set.

Note: OFFSET may be run with only one axis at a time. The status of the remaining three axes is not affected by running OFFSET.

To run OFFSET, the following steps should be followed for the corresponding axis:

1. The axis should be in closed loop with optimal gains set.
2. K_i must be non zero for the axis.
3. The axis should be 'stopped', with no motion commands in progress.
4. Start OFFSET with the specified axis.
5. Offset adjust is complete when a host interrupt is generated.

SEE ALSO CTRL

OFFSET cont.

APPLICATION

Most servo amplifiers on the market present an input offset voltage problem that is undesirable for an accurate positioning application. Using OFFSET, you may neutralize amplifier offset. To make this happen, you must:

1. enable OFFSET for the axis whose offset is to be neutralized, and
2. use a non-zero K_i gain that maintains stability and zeros position error. (It is assumed that other control gains are selected such that the system is stable.)

Position error is integrated via the integral channel until position error is forced to zero. In the absence of amplifier offset, the DAC voltage that would have achieved zero position error is zero. Any non-zero DAC value is due to an error caused by amplifier offset voltage. Mx4 Octavia measures the voltage, reports satisfactory completion of the OFFSET command (generates an interrupt) and uses this measured voltage value to neutralize offset throughout the entire control operation (until machine is turned off). Due to the variable nature of amplifier offset, offset calibration may be necessary any time the machine is turned on.

Command Sequence Example

```
MAXACC ( )    ;set the maximum accel. so system can be stopped
CTRL ( )      ;set the gains
KILIMIT ( )   ;put system in a position loop, make sure integral
              ;gain is non-zero
.
.
OFFSET ( )
```

OFFSET cont.

EXAMPLE

After verifying that OFFSET Steps 1-3 (see DESCRIPTION, above) have been followed, do offset tuning for axis 3.

The value of the RTC argument is:

n : 04h

OUTGAIN

FUNCTION Output Loop Gain

DPR ORDER command code, n, m_1, ... , m_8

USAGE Host (command code: 81h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
m_x	a single byte to specify the gains

m=0	gain=1
m=1	gain=2
m=2	gain=4
m=3	gain=8
m=4	gain=16

DESCRIPTION

This command is used to set the gain for the output of the position loops. The default m is set to zero (gain = 1).

Note: Please see block diagram with CTRL command.

SEE ALSO CTRL

APPLICATION

In applications where the number of position encoder counts (per mechanical revolution of the shaft) is low, lack of resolution in the feedback path will manifest itself as a low gain. This may be compensated for by a loop gain adjustment. In practice, this command may use an argument greater than 1 if the encoder line number is less than 1000.

OUTGAIN cont.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
OUTGAIN ( )
```

EXAMPLE

Set output loop gains of eight for axis 3, and two for axis 4, and 7.

The values of the RTC arguments are:

n	=	4Ch
m_3	=	03h
m_4	=	01h
m_7	=	01h

OUTP_OFF

FUNCTION Set Outputs to 'Off' State

DPR ORDER command code, c, $outp_2$

USAGE Host (command code: 55h), DSPL (Motion)

ARGUMENTS

$outp_1$ a single word, bit coding the outputs

if bit=0	no change in output status
if bit=1	output = HIGH TTL voltage
bit 15	OUT15 output
bit 14	OUT14 output
bit 13	OUT13 output
bit 12	OUT12 output
bit 11	OUT11 output
bit 10	OUT10 output
bit 9	OUT9 output
bit 8	OUT8 output
bit 7	OUT7 output
bit 6	OUT6 output
bit 5	OUT5 output
bit 4	OUT4 output
bit 3	OUT3 output
bit 2	OUT2 output
bit 1	OUT1 output
bit 0	OUT0 output

$outp_2$ a single word, bit coding the outputs

if bit=0	no change in output status
if bit=1	output = HIGH TTL voltage
bit 15	OUT16 output
bit 14	OUT17 output
bit 13	OUT18 output
bit 12	OUT19 output
bit 11	OUT20 output

OUTP_OFF cont.

bit 10	OUT21 output
bit 9	OUT22 output
bit 8	OUT23 output
bit 7	OUT24 output
bit 6	OUT25 output
bit 5	OUT26 output
bit 4	OUT27output
bit 3	OUT28 output
bit 2	OUT29 output
bit 1	OUT30 output
bit 0	OUT31 output

DESCRIPTION

This command allows the 'OFF' status of all outputs to be set.

SEE ALSO OUTP_ON, POSBRK_OUT

APPLICATION

This command can be used for a general purpose logical output operation.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Turn 'off' the OUT0, OUT5, OUT6, and OUT12 outputs.

The arguments for this instruction will be:

$outp_1$:	1061h
$outp_2$:	0000h

OUTP_ON

FUNCTION Set Outputs To 'On' State

DPR ORDER command code, $outp_1$, $outp_2$

USAGE Host (command code: 56h), DSPL (Motion)

ARGUMENTS

$outp_1$ a single word, bit coding the outputs

if bit=0	no change in output status
if bit=1	output = LOW TTL voltage
bit 15	OUT15 output
bit 14	OUT14 output
bit 13	OUT13 output
bit 12	OUT12 output
bit 11	OUT11 output
bit 10	OUT10 output
bit 9	OUT9 output
bit 8	OUT8 output
bit 7	OUT7 output
bit 6	OUT6 output
bit 5	OUT5 output
bit 4	OUT4 output
bit 3	OUT3 output
bit 2	OUT2 output
bit 1	OUT1 output
bit 0	OUT0 output

$outp_2$ a single word, bit coding the outputs

if bit=0	no change in output status
if bit=1	output = LOW TTL voltage
bit 15	OUT16 output
bit 14	OUT17 output
bit 13	OUT18 output
bit 12	OUT19 output

OUTP_ON cont.

bit 11	OUT20 output
bit 10	OUT21 output
bit 9	OUT22 output
bit 8	OUT23 output
bit 7	OUT24 output
bit 6	OUT25 output
bit 5	OUT26 output
bit 4	OUT27 output
bit 3	OUT28 output
bit 2	OUT29 output
bit 1	OUT30 output
bit 0	OUT31 output

DESCRIPTION

This command allows the 'ON' status of all outputs to be set.

SEE ALSO OUTP_OFF, POSBRK_OUT

APPLICATION

This command can be used for a general purpose logical output operation.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Enable or turn 'on' the OUT1, OUT11, OUT12, and OUT22 outputs.

The arguments for this instruction will be:

$outp_1$:	1802h
$outp_2$:	0040h

OVERRIDE

FUNCTION Feedrate override for CIRCLE / LINEAR motion

DPR ORDER command code, val

USAGE Host (command code: 8Bh), DSPL (Motion)

ARGUMENTS

val — feedrate override multiplier, 32-bit two's complement value, 16-bits integer, 16-bits fraction

0x00002000h <= val <= 0x000A0000h

DESCRIPTION

This command is used to set the feedrate override for the CIRCLE and LINEAR related commands (see *DSPL Programmer's Guide*).

SEE ALSO CIRCLE, LINEAR_MOVE (*DSPL Programmer's Guide*)

APPLICATION

none

EXAMPLE

Set a feedrate override of 4x.

The value of the RTC argument is:

val : 00040000h

PARREAD

FUNCTION Parameter Readback

DPR ORDER command code, m

USAGE Host (command code: 5Eh)

ARGUMENTS

m a byte which indicates the parameters to echo.

m=10h axis 1, 5 position loop gain values [CTRL]
m=11h axis 2, 6 position loop gain values [CTRL]
m=12h axis 3, 7 position loop gain values [CTRL]
m=13h axis 4, 8 position loop gain values [CTRL]
m=14h Kilimit value [KILIMIT]
m=15h position loop output gain values [OUTGAIN]
m=16h maximum acceleration [MAXACC]
m=17h enabled interrupt
m=18h mode of operation
m=19h following error and halt interrupt setpoint [EN_ERRHLT]
m=1Ah following error interrupt setpoint [EN_ERR]
m=1Bh axis 1, 5 and 2, 6 position breakpoint interrupt setpoint [EN_POSBRK]
m=1Ch axis 3, 7 and 4, 8 position breakpoint interrupt setpoint [EN_POSBRK]
m=1Dh buffer breakpoint interrupt setpoint and contouring block transfer rate [EN_BUFBRK, BTRATE, CUBIC_RATE]
m=1Eh axis 1, 5 and 2, 6 position breakpoint output mask [POSBRK_OUT]
m=1Fh axis 3, 7 and 4, 8 position breakpoint output mask [POSBRK_OUT]
m=20h abort maximum acceleration [ESTOP_ACC]
m=21h master/slave status
m=22h output status [OUTP_ON, OUTP_OFF]
m=23h input state
m=24h encoder fault interrupt setpoint [EN_ENCFLT]
m=25h not used
m=26h acceleration feedforward gain value [CTRL_KA]
m=27h torque limit value [TRQ_LIMIT]

PARREAD cont.

DESCRIPTION

Upon the execution of this command, Mx4 Octavia echoes the desired parameters to DPR locations 0B8h - 0BFh / 8B8h - 8BFh. "m" is echoed to DPR location 0B7h if the parameters are ready in the DPR. Parameters may take more than 5 ms to echo back to the DPR. Host can use the following algorithm:

1. write m to DPR location 3C3h
2. write 0 to DPR location 0B7h
3. write RTC command code to DPR location 3C2h
4. poll DPR location 0B7h until m is echoed
5. read the data from DPR locations 0B8h - 0BFh

DATA FORMAT

For each type of parameter, DPR locations 0B8h - 0BFh / 8B8h - 8BFh are interpreted differently. Axis 1-4 data is echoed to DPR locations 0B8h - 0bfh. Axis 5-8. The following shows the format for each type of parameter:

1. Position loop gains (m=10h - m=13h)

0B8h	K_i low byte
0B9h	K_i high byte
0BAh	K_p low byte
0BBh	K_p high byte
0BCh	K_f low byte
0BDh	K_f high byte
0BEh	K_d low byte
0BFh	K_d high byte

PARREAD cont.

2. Kilimit (m=14h)

0B8h	Kilimit for axis 1
0B9h	Kilimit for axis 2
0BAh	Kilimit for axis 3
0BBh	Kilimit for axis 4
0BCh	
:	not used
0BFh	

Note: $0 \leq \text{Kilimit} \leq 14$

3. Position loop output gain (m=15h)

0B8h	m specified gains for axis 1
0B9h	m specified gains for axis 2
0BAh	m specified gains for axis 3
0BBh	m specified gains for axis 4
0BCh	
:	not used
0BFh	

Note: $0 \leq m \leq 4$

4. Maximum acceleration (m=16h)

0B8h	low byte acceleration for axis 1
0B9h	high byte acceleration for axis 1
0BAh	low byte acceleration for axis 2
0BBh	high byte acceleration for axis 2
0BCh	low byte acceleration for axis 3
0BDh	high byte acceleration for axis 3
0BEh	low byte acceleration for axis 4
0BFh	high byte acceleration for axis 4

PARREAD cont.

5. Enabled interrupt (m=17h)

0B8h	bit 0 codes buffer breakpoint interrupt
0B9h	low nibble bit codes the following error and halt interrupts, high nibble bit codes the following error interrupts
0BAh	low nibble bit codes the index pulse interrupts, high nibble bit codes the position breakpoint interrupts
0BBh	low nibble bit codes the motion complete interrupts, high nibble bit codes the probe interrupts
0BCh : 0BFh	not used

6. Mode of operation (m=18h)

0B8h	low nibble bit codes the axes in axis move operation
0B9h	low nibble bit codes the axes in stop operation
0BAh	low nibble bit codes the axes in velmode operation
0BBh	low nibble bit codes the axes in contouring operation
0BCh : 0BFh	not used

PARREAD cont.

7. Following error and halt interrupt setpoint (m=19h)

0B8h	low byte setpoint for axis 1
0B9h	high byte setpoint for axis 1
0BAh	low byte setpoint for axis 2
0BBh	high byte setpoint for axis 2
0BCh	low byte setpoint for axis 3
0BDh	high byte setpoint for axis 3
0BEh	low byte setpoint for axis 4
0BFh	high byte setpoint for axis 4

8. Following error interrupt setpoint (m=1Ah)

0B8h	low byte setpoint for axis 1
0B9h	high byte setpoint for axis 1
0BAh	low byte setpoint for axis 2
0BBh	high byte setpoint for axis 2
0BCh	low byte setpoint for axis 3
0BDh	high byte setpoint for axis 3
0BEh	low byte setpoint for axis 4
0BFh	high byte setpoint for axis 4

9. Position breakpoint setpoint (m=1B for axes 1 and 2, 1Ch for axes 3 and 4)

0B8h	low word low byte setpoint for axis 1 or 3
0B9h	low word high byte setpoint for axis 1 or 3
0BAh	high word low byte setpoint for axis 1 or 3
0BBh	high word high byte setpoint for axis 1 or 3
0BCh	low word low byte setpoint for axis 2 or 4
0BDh	low word high byte setpoint for axis 2 or 4
0BEh	high word low byte setpoint for axis 2 or 4
0BFh	high word high byte setpoint for axis 2 or 4

PARREAD cont.

10. Buffer breakpoint interrupt setpoint and contouring block transfer rate (m=1Dh)

0B8h	buffer breakpoint interrupt setpoint
0B9h	= 00h : 2nd order contouring
	= FFh : cubic spline contouring
0BAh	low byte, block transfer rate
0BBh	high byte, block transfer rate (for cubic spline only)
0BCh : 0BFh	not used

11. Position breakpoint output masks (m=1E for axes 1 and 2, 1Fh for axes 3 and 4)

0B8h	low byte output mask ON for axis 1 or 3
0B9h	high byte output mask ON for axis 1 or 3
0BAh	low byte output mask OFF for axis 1 or 3
0BBh	high byte output mask OFF for axis 1 or 3
0BCh	low byte output mask ON for axis 2 or 4
0BDh	high byte output mask ON for axis 2 or 4
0BEh	low byte output mask OFF for axis 2 or 4
0BFh	high byte output mask OFF for axis 2 or 4

12. Abort maximum acceleration (m=20h)

0B8h	low byte acceleration for axis 1
0B9h	high byte acceleration for axis 1
0BAh	low byte acceleration for axis 2
0BBh	high byte acceleration for axis 2
0BCh	low byte acceleration for axis 3
0BDh	high byte acceleration for axis 3
0BEh	low byte acceleration for axis 4
0BFh	high byte acceleration for axis 4

PARREAD cont.

13. Master/Slave status (m=21h)

0B8h	=00h, configured as Master =11h, configured as Slave
0B9h :	not used
0BFh	

14. Output status (m=22h)

0B8h	bit 7	: OUT7
	bit 6	: OUT6
	bit 5	: OUT5
	bit 4	: OUT4
	bit 3	: OUT3
	bit 2	: OUT2
	bit 1	: OUT1
	bit 0	: OUT0
0B9h	bit 7	: OUT15
	bit 6	: OUT14
	bit 5	: OUT13
	bit 4	: OUT12
	bit 3	: OUT11
	bit 2	: OUT10
	bit 1	: OUT9
	bit 0	: OUT8
0BAh	bit 7	: OUT23
	bit 6	: OUT22
	bit 5	: OUT21
	bit 4	: OUT20
	bit 3	: OUT19
	bit 2	: OUT18
	bit 1	: OUT17
	bit 0	: OUT16

PARREAD cont.

0BBh	bit 7	: OUT31
	bit 6	: OUT30
	bit 5	: OUT29
	bit 4	: OUT28
	bit 3	: OUT27
	bit 2	: OUT26
	bit 1	: OUT25
	bit 0	: OUT24
0BCh		
	not used	
0BFh		

15. Logic state of inputs (m=23h)

0B8h	echo inp_1 byte of INP_STATE	
0B9h	echo inp_2 byte of INP_STATE	
0BAh	bit 7	echo inp_3 bit 5
	bit 6	echo inp_3 bit 4
	bit 5	0
	bit 4	0
	bit 3	echo inp_3 bit 3
	bit 2	echo inp_3 bit 2
	bit 1	echo inp_3 bit 1
	bit 0	echo inp_3 bit 0
0BBh		
	not used	
0bfh		

PARREAD cont.

16. Encoder fault interrupt setpoint (m=24h)

0B8h	low byte setpoint for axis 1
0B9h	high byte setpoint for axis 1
0BAh	low byte setpoint for axis 2
0BBh	high byte setpoint for axis 2
0BCh	low byte setpoint for axis 3
0BDh	high byte setpoint for axis 3
0BEh	low byte setpoint for axis 4
0BFh	high byte setpoint for axis 4

17. Not Used (m=25h)

18. Acceleration feedforward gain value (m=26h)

0B8h	low byte K_a for axis 1
0B9h	high byte K_a for axis 1
0BAh	low byte K_a for axis 2
0BBh	high byte K_a for axis 2
0BCh	low byte K_a for axis 3
0BDh	high byte K_a for axis 3
0BEh	low byte K_a for axis 4
0BFh	high byte K_a for axis 4

19. Torque limit value (m=27h)

0B8h	low byte trq limit value for axis 1
0B9h	high byte trq limit value for axis 1
0BAh	low byte trq limit value for axis 2
0BBh	high byte trq limit value for axis 2
0BCh	low byte trq limit value for axis 3
0BDh	high byte trq limit value for axis 3
0BEh	low byte trq limit value for axis 4
0BFh	high byte trq limit value for axis 4

PARREAD cont.

SEE ALSO none

APPLICATION

This command can be used as a diagnostic tool to monitor all system parameters.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Verify the gains settings for axis 2 by instructing Mx4 Octavia to echo the values to the DPR with a PARREAD command.

The value of the RTC argument is:

m = 11h

POS_PRESET

FUNCTION Preset Position Counter

DPR ORDER command code, n, $pset_1$, ... , $pset_8$

USAGE Host (command code 68h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
$pset_x$	32-bit two's complement value to preset the axis x position counter

DESCRIPTION

This command will define the present position point for the axes specified.

Note: POS_PRESET will automatically disable the position breakpoint interrupt (if enabled). POS_PRESET should be executed only when the axes specified are not in motion.

SEE ALSO POS_SHIFT, EN_POSBRK

APPLICATION

This command is useful when the position counter must be forced to a new value. POS_PRESET may be used in the establishment of a new reference position. Please also see POS_SHIFT.

Command Sequence Example

No preparation is required before running this instruction.

POS_PRESET cont.

EXAMPLE

Set the present position of axes 4 and 5 to 50,000 counts.

The values of the RTC arguments are:

n	:	18h
$pset_4$	:	0000C350h
$pset_5$	:	0000C350h

POS_SHIFT

FUNCTION Position Reference Shift

DPR ORDER command code, n, $psft_1$, ... , $psft_8$

USAGE Host (command code: 5Dh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
$psft_x$	32-bit two's complement value to add to the axis x position counter

DESCRIPTION

This command will shift the present position point for the axes specified.

Note: POS_SHIFT will automatically disable the position breakpoint interrupt (if enabled) of the specified axes.

SEE ALSO POS_PRESET, EN_POSBRK

APPLICATION

This command may be used in homing a linear system based on index pulse position recording. Adding offset position (in encoder edge counts) to an already recorded position, presets position to a new value without losing position integrity (i.e., no counter information is lost). See also EN_INDEX and POS_PRESET.

Command Sequence Example

No preparation is required before running this instruction.

POS_SHIFT cont.

EXAMPLE

The current axis 1 position is 100h. Shift the axis 1 position to 20100h.
The current axis 3 position is 1010h. Shift the axis 3 position to 1000h.
The current axis 6 position is 36F3h. Shift the axis 6 position to F5D0h.

The values of the RTC arguments are:

n	:	25h
$psft_1$	:	00020000h
$psft_3$	:	FFFFFFF0h
$psft_3$	:	00006EDDh

POSBRK_OUT

FUNCTION Set Outputs After Position Breakpoint Interrupt

DPR ORDER command code, n, $outputon_{11}$, $outputon_{12,}$ $outputoff_{11,}$ $outputoff_{12,}$, ..., $outputon_{81}$, $outputon_{82,}$ $outputoff_{81,}$ $outputoff_{82}$

USAGE Host (command code: 7Dh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
$outpon_{x1}$	a single word, bit coding the outputs to turn 'on' upon occurance of position breakpoint interrupt (EN_POSBRK) for axis x.

if bit=0	no change in output status
if bit=1	output = LOW TTL voltage

bit 15	OUT15 output
bit 14	OUT14 output
bit 13	OUT13 output
bit 12	OUT12 output
bit 11	OUT11 output
bit 10	OUT10 output
bit 9	OUT9 output
bit 8	OUT8 output
bit 7	OUT7 output
bit 6	OUT6 output
bit 5	OUT5 output
bit 4	OUT4 output
bit 3	OUT3 output
bit 2	OUT2 output
bit 1	OUT1 output
bit 0	OUT0 output

POSBRK_OUT cont.

$outpon_{x2}$ a single word, bit coding the outputs to turn 'on' upon occurance of position breakpoint interrupt (EN_POSBRK) for axis x.

if bit=0	no change in output status
if bit=1	output = LOW TTL voltage
bit 31	OUT31 output
bit 30	OUT30output
bit 29	OUT29 output
bit 28	OUT28 output
bit 27	OUT27 output
bit 26	OUT26 output
bit 25	OUT25 output
bit 24	OUT24 output
bit 23	OUT23 output
bit 22	OUT22 output
bit 21	OUT21 output
bit 20	OUT20 output
bit 19	OUT19 output
bit 18	OUT18 output
bit 17	OUT17 output
bit 16	OUT16 output

$outpoff_{x1}$ a single word, bit coding the outputs to turn 'off' upon occurance of position breakpoint interrupt (EN_POSBRK) for axis x.

if bit=0	no change in output status
if bit=1	output = HIGH TTL voltage
bit 15	OUT15 output
bit 14	OUT14 output
bit 13	OUT13 output
bit 12	OUT12 output
bit 11	OUT11 output

POSBRK_OUT cont.

bit 10	OUT10 output
bit 9	OUT9 output
bit 8	OUT8 output
bit 7	OUT7 output
bit 6	OUT6 output
bit 5	OUT5 output
bit 4	OUT4 output
bit 3	OUT3 output
bit 2	OUT2 output
bit 1	OUT1 output
bit 0	OUT0 output

$outpoff_{x2}$ a single word, bit coding the outputs to turn 'off' upon occurance of position breakpoint interrupt (EN_POSBRK) for axis x.

if bit=0	no change in output status
if bit=1	output = HIGH TTL voltage
bit 31	OUT31 output
bit 30	OUT30output
bit 29	OUT29 output
bit 28	OUT28 output
bit 27	OUT27 output
bit 26	OUT26 output
bit 25	OUT25 output
bit 24	OUT24 output
bit 23	OUT23 output
bit 22	OUT22 output
bit 21	OUT21 output
bit 20	OUT20 output
bit 19	OUT19 output
bit 18	OUT18 output
bit 17	OUT17 output
bit 16	OUT16 output

POSBRK_OUT cont.

DESCRIPTION

This command enables the output status of selected outputs to be activated by the occurrence of a position breakpoint interrupt (EN_POSBRK) for a specified axis. The POSBRK_OUT need only be executed once (ie: during initialization) unless the on/off output status desired changes. The specified outputs will change state as programmed through the $outpon_x$ and $outpoff_x$ arguments when an axis (axis x) generates a position breakpoint interrupt. The position breakpoint interrupt (EN_POSBRK) must be enabled for the output status changes to occur.

SEE ALSO EN_POSBRK, OUTP_OFF, OUTP_ON

APPLICATION

This command can be used for an output operation where the output status must be tightly coupled to the position of one or more axes.

Command Sequence Example

```
EN_POSBRK     ;enable the pos breakpoint int for specified axis(es)
POSBRK_OUT    ;set the desired output status changes
```

EXAMPLE

If a position breakpoint interrupt occurs on axis 1, turn on OUT0-OUT3 and turn off OUT4. If a position breakpoint interrupt occurs on axis 2, turn off all outputs.

The arguments for this instruction will be:

$outputon_{11}$	:	000Fh
$outputon_{12}$	:	0000h
$outputoff_{11}$	:	0010h
$outputoff_{12}$	:	0000h
$outputon_{21}$	:	0000h
$outputon_{22}$	:	0000h
$outputoff_{21}$	:	FFFFh
$outputoff_{22}$	:	FFFFh

REL_AXMOVE

FUNCTION Relative Position Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, vel_1, ... , acc_8, pos_8, vel_8

USAGE Host (command code: B7h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit incremental position value for axis x. The end (target) position for the REL_AXMOVE motion is the following where abs[pos_x] ≤ 0x2FFFFFFFh counts. endposition = current command position + pos_x
vel_x	32-bit unsigned slew rate for axis x

Note 1: Position and velocity are always presented in two's complement format, but acceleration is an unsigned value.

Note 2: Velocity must be presented as a 25-bit two's complement value which is sign extended to 32 bits. For example, the maximum unsigned velocity is 00FFFFFFh.

Note 3: Velocity is partitioned into 16 bits integer and 16 bits fraction. Position is a 32-bit integer value, and acceleration is presented as 1 bit integer, 15 bits fraction.

REL_AXMOVE cont.

DESCRIPTION

The REL_AXMOVE command is similar to the AXMOVE command with the exception that relative (or incremental) position is specified, rather than an end position as with AXMOVE.

SEE ALSO AXMOVE, STOP, REL_AXMOVE_S, REL_AXMOVE_T, AXMOVE_S, AXMOVE_T

EXAMPLE

The current position (commanded) of axis 2 is unknown. It is known, however, that we want to move axis 2 8000 counts in the negative direction (that is, -8000 counts from the current position). The move should be accomplished with an acceleration of 1.0 counts/(200µs)^2 and a target velocity of (unsigned) 3.5 counts/200µs.

The values of the RTC arguments are:

n	:	02h
acc_2	:	8000h
pos_2	:	FFFFE0C0h
vel_2	:	00038000h

REL_AXMOVE_S

FUNCTION Relative S-Curve Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, vel_1, ... , acc_8, pos_8, vel_8

USAGE Host (command code: 75h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit relative position for axis x
vel_x	32-bit unsigned slew rate for axis x

Note 1: Position and velocity are always presented in two's complement format, but acceleration is an unsigned value.

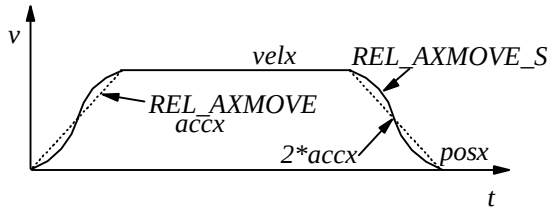
Note 2: Velocity must be presented as a 25-bit two's complement value, which is sign extended to 32 bits. For example, the maximum unsigned velocity is 00FFFFFFh.

Note 3: Velocity is partitioned into 16 bits integer and 16 bits fraction. Position is a 32-bit integer value, and acceleration is presented as 1 bit integer, 15 bits fraction.

DESCRIPTION

The REL_AXMOVE_S RTC allows for s-curve command generation with relative (to current position) endpoint position, slew rate velocity, and acceleration for each axis. This command is suitable for linear moves where s-curve acceleration is desired.

REL_AXMOVE_S cont.



The figure above illustrates the velocity profile of the REL_AXMOVE_S along with the linear velocity ramp of the REL_AXMOVE command. With REL_AXMOVE_S, the acceleration will reach a value of 2*accx for a maximum (see above figure).

SEE ALSO AXMOVE, AXMOVE_S, AXMOVE_T, REL_AXMOVE, REL_AXMOVE_T, STOP

EXAMPLE

The current position (commanded) of axis 2 is unknown. It is known, however, that we want to move axis 2 8000 counts in the negative direction (that is, -8000 counts from the current position). The move should be accomplished with an acceleration of 1.0 counts/(200µs)^2 and a target velocity of (unsigned) 3.5 counts/200µs.

The values of the RTC arguments are:

n	:	02h
acc_2	:	8000h
pos_2	:	FFFFE0C0h
vel_2	:	00038000h

REL_AXMOVE_T

FUNCTION Time-Based Relative Axis Move with Trapezoidal Trajectory

DPR ORDER command code, n, acc_1, pos_1, tm_1, ... , acc_8, pos_8, tm_8

USAGE Host (command code: 78h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc_x	16-bit acceleration for axis x
pos_x	32-bit relative position for axis x
tm_x	32-bit unsigned time for axis x

Note 1: Position is always presented in two's complement format, but acceleration and time are unsigned values.

Note 2: The time argument, tm_x, is a 32 bit unsigned value with a unit of 200usec.

DESCRIPTION

The REL_AXMOVE_T RTC allows for trapezoidal command generation with relative (to current position) endpoint position, acceleration, and time to complete the move for each axis. This command is suitable for linear moves where relative endpoint position and motion time are the specifying parameters.

REL_AXMOVE_T cont.

The REL_AXMOVE_T command is similar to REL_AXMOVE, with the exception that the velocity argument is replaced with a time argument. REL_AXMOVE_T will automatically calculate a suitable slew rate velocity to achieve the programmed relative endpoint position in the programmed amount of time, following a trapezoidal velocity profile (similar to REL_AXMOVE).

SEE ALSO REL_AXMOVE, REL_AXMOVE_S, AXMOVE, AXMOVE_S, AXMOVE_T, STOP

EXAMPLE

The current position (commanded) of axis 4 is unknown. It is known, however, that we want to move axis 4 10000 counts in the negative direction (that is, -10000 counts from the current position). The move should be accomplished with an acceleration of 1.0 counts/(200µs)^2 and be completed in 350msec (1750*200usec).

The values of the RTC arguments are:

n	:	08h
acc_4	:	8000h
pos_4	:	FFFFD8F0h
tm_4	:	000006D6h

REL_AXMOVE_SLAVE

FUNCTION Superimposes a Relative Axis Move onto a Slave Engaged in Gearing

DPR ORDER command code, n, acc, rel_pos, rel_vel

USAGE Host (command code: AEh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
acc	16-bit unsigned value for relative move acceleration
rel_pos	32-bit position value relative to current position
rel_vel	32-bit velocity value relative to current velocity

DESCRIPTION

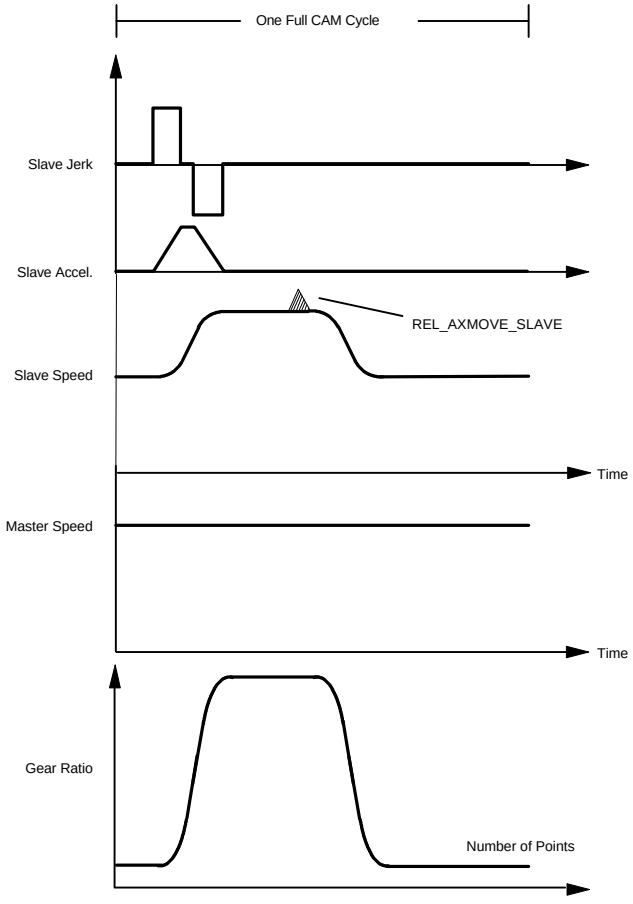
This RTC is similar to AXMOVE with two exceptions. First, it is relative not absolute; and second, it works only on the slave axis(es) involved in electronically geared or cam applications. This command allows the slave to momentarily disengage from the gearing process and compensate for its position shortcomings.

SEE ALSO CAM, CAM_OFF, CAM_OFF_ACC, CAM_POS, CAM_PROBE, CAM_TSPACE, GEAR, GEAR_OFF, GEAR_OFF_ACC, GEAR_POS, GEAR_PROBE, SYNC

APPLICATION

General master/slaving, in particular flying shear applications, can benefit from this instruction. Flying shear with registration marks is handled similarly to that of synchronous cutting. That is, the measured cutting error is used in the next cycle as an added function to compensate for the motion's shortcomings.

REL_AXMOVE_SLAVE cont.



RESET

FUNCTION Reset Mx4 Octavia

DPR ORDER command code, AAh, AAh

USAGE Host (command code: 72h), DSPL (Motion)

ARGUMENTS

AAh reset signature byte

DESCRIPTION

This command brings the servo controller card back to power-up state. Upon Mx4 Octavia's reset completion, a host interrupt is generated via bit 4 of DPR location 1FFEh.

SEE ALSO none

APPLICATION

From time to time all systems may have to be software reset to allow for an initialization.

Command Sequence Example

No preparation is required before running this instruction.

EXAMPLE

Reset the Mx4 Octavia controller card.

The arguments of RESET are AAh, AAh (2 bytes).

SIGNAL_DSPL

FUNCTION Send a Real-Time 'Signal' to DSPL Program

DPR ORDER command code

USAGE HOST (command code: 94h)

ARGUMENTS

none

DESCRIPTION

SIGNAL_DSPL sends a real-time software 'signal' to the running DSPL program. In order for the signal to be received, the DSPL program must be waiting at a WAIT_UNTIL_RTC command while the SIGNAL_DSPL command is executed. The SIGNAL_DSPL - WAIT_UNTIL_RTC pair is used for timing or synchronization purposes between a DSPL program and the host computer.

SEE ALSO WAIT_UNTIL_RTC (*DSPL Programmer's Guide*)

APPLICATION

For use with *DSPL Programmer's Guide*. [see *DSPL Programmer's Guide*, *DSPL Command Set*, WAIT_UNTIL_RTC command description]

EXAMPLE

Send a signal to the active DSPL program.

Only the command code 94h needs to be sent to the Mx4 Octavia controller for the SIGNAL_DSPL command.

START

FUNCTION Start Contouring Motion

DPR ORDER command code, n

USAGE Host (command code: 6Dh), DSPL (Motion)

ARGUMENTS

n a single byte, bit coding the axes involved

DESCRIPTION

This command starts the motion (simultaneously) for the specified axes included in 2nd order and cubic spline contouring. START applies to contouring only.

Note: START will be ignored if contouring is in progress.

SEE ALSO STOP, VECCHG

APPLICATION

This command must be used in all 2nd order and ring buffer cubic spline contouring applications to start contouring with selected axes.

For 2nd Order Contouring Only

This command can be overwritten by VECCHG, which redefines the axes involved in the contouring process. For example, START starts the contouring of axes 1, 3, and 4. If in the course of contouring, a VECCHG is received (with argument) specifying axes 1, 2, and 3, the new contouring points in the ring buffer will be used for the newly defined axes. Please also see VECCHG.

START cont.

Command Sequence Example

```
.                   ;load ring buffer with positions and velocities
.
MAXACC ( )          ;make sure system can stop
CTRL ( )            ;set the gains
KILIMIT ( )
BTRATE ( )          ;set the block transfer rate
EN_BUFBRK ( )       ;set the breakpoint in the ring buffer
.
.
START ( )           ;start contouring
```

EXAMPLE

Start contouring motion in axes 2 and 3.

The values of the RTC argument is:

n : 06h

START_DSPL

FUNCTION Initiate DSPL Code Execution

DPR ORDER command code

USAGE HOST (command code: 91h)

ARGUMENTS

none

DESCRIPTION

START_DSPL is a host command used to initiate execution of a DSPL program that has been previously downloaded to Mx4 Octavia.

SEE ALSO CLEAR_DSPL, STOP_DSPL

APPLICATION

For use with *DSPL Programmer's Guide*. [see *DSPL Programmer's Guide*, *Using the DSPL Compiler & Mx4 Octavia Downloader, Start/Stop Execution of DSPL Program*]

EXAMPLE

Initiate the execution of the DSPL program that has been loaded to Mx4 Octavia's DSPL program storage area.

Only the command code 91h needs to be sent to the Mx4 Octavia controller for the START_DSPL command.

STEPPER_ON

FUNCTION Select Servo / Stepper Axes

DPR ORDER command code, n

USAGE HOST (command code: 8Dh), DSPL (Motion)

ARGUMENTS

n	a byte, bit coding the axis(es) selected as stepper axis(es) (the remaining axes default to servo axes)

DESCRIPTION

This command requires the Stp4 add-on card. STEPPER_ON allows the user to select the axes that are stepper control axes. The axes not selected by the n argument remain servo control axes.

SEE ALSO none

EXAMPLE

Select axes 1 and 8 as stepper control axes. The remaining axes are to be configured as servo axes.

The value of the RTC argument is:

n : 81h

STOP

FUNCTION Stop Motion

DPR ORDER command code, n

USAGE Host (command code: 6Eh), DSPL (Motion)

ARGUMENTS

n a single byte, bit coding the axes involved

DESCRIPTION

This command stops the motion of all specified axes simultaneously. To stop motion, the servo control card uses the programmed values for maximum acceleration / deceleration. Upon receipt of STOP, the servo controller aborts the current command. The host is responsible for clearing the ring buffer of any remaining commands if the axis(es) stopped was involved in contouring motion.

Note 1: An emergency stop signal, ESTOP_ACC, will perform a hardware stop. This is an open collector input signal that is active low and is shared between all of the controller cards.

Note 2: STOP will be ignored if the maximum acceleration / deceleration is equal to zero (e.g., MAXACC not issued).

If an axis is halting to a stop from a previously executed STOP RTC or active ESTOP_ACC input, Mx4 Octavia will ignore any motion commands (AXMOVE, REL_AXMOVE, START or VELMODE) and will report an "RTC Command Ignored" interrupt to the host. The above motion commands should not be sent to Mx4 Octavia for a halting axis until the axis motion has come to a stop.

SEE ALSO AXMOVE, MAXACC, START

STOP cont.

APPLICATION

For all applications involving bringing speed to zero in the quickest possible manner.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
BTRATE ( )      ;set the block transfer rate
EN_BUFBRK ( ) ;set the breakpoint in the ring buffer
.
.
STOP ( )        ;stop the motion
.               ;upon completion of stop (command) trajectory
.               ;Mx4 Octavia generates motion complete interrupt
```

EXAMPLE

Bring the motion of axes 1, 4, and 7 to a halt.

The value of the RTC argument is:

n : 49h

STOP_DSPL

FUNCTION Terminate DSPL Program Code Execution

DPR ORDER command code

USAGE HOST (command code: 97h)

ARGUMENTS

none

DESCRIPTION

STOP_DSPL is a host command used to terminate the execution of the DSPL program.

This command will also halt the motion (if any) of all axes with the programmed MAXACC acceleration.

SEE ALSO CLEAR_DSPL, MAXACC, START_DSPL

APPLICATION

For use with *DSPL Programmer's Guide*. [see *DSPL Programmer's Guide, Using the DSPL Compiler & Mx4 Octavia Downloader, Start/Stop Execution of DSPL Program*]

EXAMPLE

Terminate the execution of the active DSPL program.

Only the command code 97h needs to be sent to the Mx4 Octavia controller for the STOP_DSPL command.

SYNC

FUNCTION Master / Slave Select

DPR ORDER command code, m

USAGE Host (command code: 87h), DSPL (motion)

ARGUMENTS

m — a byte that selects the Master / Slave status of the Mx4 Octavia card

m = 0 : Mx4 Octavia is configured as a Master
m < > 0 : Mx4 Octavia is configured as a Slave

DESCRIPTION

If more than one Mx4 Octavia card is to be used in a system and card-to-card synchronization is required, the SYNC RTC should be used. SYNC allows multiple Mx4 Octavia cards to operate in synchronization within a system by specifying a single Master and the remaining card(s) as Slaves. If only one Mx4 Octavia is used in a host computer system, that Mx4 Octavia must be configured as a Master.

Note: Mx4 Octavia powers-up and resets to a default Master status.

In addition to configuring the Mx4 Octavia cards with SYNC (for multiple card systems), a cable jumper must be included on the J5 connector of each of the boards. The cable must be wired such that the MASTER signal from the Master Mx4 Octavia connects to the SLAVE signal of each of the Slave Mx4 Octavia(s) (see *Installing Your Mx4 Octavia*).

SEE ALSO none

SYNC cont.

APPLICATION

This command is used in applications where tight coordination of more than four axes is required. This command essentially slaves several Mx4 Octavia cards to a single Master Mx4 Octavia. Applications involving many axes contouring may benefit from this command.

Command Sequence Example

This command must be executed immediately after the initialization. Please remember that the default value for m is zero (i.e., the card is initialized as a Master).

EXAMPLE

Configure the Mx4 Octavia controller as a slave in a multi-Mx4 Octavia synchronized system.

The value of the RTC argument is:

m : 0lh

TABLE_SEL

FUNCTION Select Compensation Table

DPR ORDER command code, n, tb_1, ... , tb_8

USAGE Host (command code: A2h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
tb_x	a single byte, specifying the compensation table to be used for axis x

$1 \leq tb_x \leq 4$

DESCRIPTION

The TABLE_SEL command allows the user to arbitrarily select the compensation table for the axis(es) in question. More than one axis may use a compensation table.

SEE ALSO CIRCLE, TABLE_OFF, TABLE_ON (*DSPL Programmer's Guide*)

EXAMPLE

Axes 1 and 2 are to use compensation table 2, while axes 3 and 4 use compensation table 1.

The values of the RTC arguments are:

n	:	0Fh
tb_1	:	02h
tb_2	:	02h
tb_3	:	01h
tb_4	:	01h

TRQ_LIMIT

FUNCTION DAC Output Voltage Limit

DPR ORDER command code, n, val_1, ... , val_8

USAGE Host (command code: 5Bh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
val_x	positive 16-bit value specifying 16-bit DAC output voltage (abs) limit for axis x

The values range as follows:

7FFFh	:	+/-10v output swing
:		
4000h	:	+/-5v output swing
:		
0000h	:	0v output swing

DESCRIPTION

The TRQ_LIMIT command specifies a torque limit (by means of output voltage limiting) value ranging from 0 volts (no output) to +/- 10 volts (full swing) with a resolution of approximately 0.3 millivolts.

The Mx4 Octavia controller powers-up and resets to a default torque limit value allowing full output voltage swing.

SEE ALSO none

APPLICATION

This command can be used in applications where an axis torque needs to be limited, such as packaging or material handling.

Command Sequence Example

No preparation is required before running this instruction.

TRQ_LIMIT cont.

EXAMPLE

Limit the output voltage swing for axis 2 to +/- 7.5 volts.

$$\left(\frac{7.5}{10}\right) \times 7\text{FFFh} = 5\textit{FFF}\text{h}$$

The values of the RTC arguments are:

n	:	02h
val_2	:	5FFFh

VECCHG

FUNCTION 2nd Order Contouring Vector Change

DPR ORDER command code, n, m

USAGE Host (command code: 6Fh), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
m	8-bit positive value which represents the buffer position (in 8 byte offsets from the start of the buffer) where the number of axes involved in contouring must be changed to include only those axes coded by n.

DESCRIPTION

Upon the execution of this command, the 2nd order contouring task assumes a new set of axes at the programmed pointer location.

Note: Three buffer levels are used to implement this instruction.

SEE ALSO START

APPLICATION

See START.

VECCHG cont.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
BTRATE ( )      ;set the block transfer rate
EN_BUFBRK ( ) ;set the buffer breakpoint interrupt
.
.
START ( )       ;start contouring for a selected number of axes
.               ;based on buffer breakpoint interrupt transfer more
.               ;points
VECCHG ( )      ;use points in ring buffer for a new set of axes
```

EXAMPLE

Begin 2nd order contouring in axes 1, 2, and 3 after the 23rd segment move command of the ring buffer.

The values of the RTC arguments are:

n	:	07h
m	:	17h

VELMODE

FUNCTION Velocity Mode

DPR ORDER command code, n, vel_1, ... , vel_8

USAGE Host (command code: 70h), DSPL (Motion)

ARGUMENTS

n	a single byte, bit coding the axes involved
vel_x	32-bit two's complement velocity value for axis x

When used in DSPL, argument vel_x may be selected as variable.

Note: Velocity is presented by a 25-bit two's complement number, which is sign extended to 32 bits. Velocity is partitioned as 16 bits integer, 16 bits fraction.

DESCRIPTION

Upon the execution of this command a velocity loop for the specified axes will be closed. The velocity loop uses the same gains as those specified using the control law command. VELMODE uses the MAXACC maximum acceleration / deceleration value to accelerate or decelerate to the desired velocity.

Note : VELMODE will be ignored if the maximum acceleration / deceleration is equal to zero (e.g., MAXACC not issued).

Note : The commanded and actual position of an axis in VELMODE motion will roll over at abs[2000 0000h] counts..

SEE ALSO MAXACC

VELMODE cont.

APPLICATIONS

This instruction is useful in all general-purpose velocity control applications. Please remember that although VELMODE primarily regulates speed, the outer loop is still position. This means that while regulating speed, Mx4 Octavia continually tries to zero the position error.

Command Sequence Example

```
MAXACC ( )      ;set the maximum accel. so system can be stopped
CTRL ( )        ;set the gains
KILIMIT ( )
.
.
VELMODE ( )
```

EXAMPLE

Engage axis 2 and axis 7 in velocity mode with a velocity of 3.71 counts/200 ms.

The values of the RTC arguments are:

n	:	42h
vel_2	:	0003B5C3h
vel_7	:	0003B5C3h

6 Mx4 Octavia Host-Based Programming

Mx4 Octavia - Host Communication

The host communicates with the PC/AT Mx4 Octavia through the host computer ISA bus. The communication takes place across a Dual Port RAM (DPR) buffer on the Mx4 Octavia card. Through this buffer, the host may read system state variables such as; position and velocity, interrupt internal Mx4 Octavia parameters, write real time instructions to the Mx4 Octavia card, monitor the interrupt status of Mx4 Octavia, and much more.

Host - Mx4 Octavia Interface

The host communicates with Mx4 Octavia via a 2048 byte Dual Port RAM (DPR). This DPR is functionally split into nine blocks, which are described below in Table 6-1.

DPR BLOCK	ADDRESS RANGE	DESCRIPTION
Status Registers	000h - 065h	The status register block includes Mx4 Octavia card status codes as well as interrupt source information.
DSPL Updates	066h - 085h	DSPL programming-related parameters such as variable values and program counter values are available for the host to read in this block.
Hardware Signature	086h - 08Bh	These bytes code, in ASCII, the hardware platform (PC/AT, Multibus or VME), hardware options (with I/O or standard configuration), and the board's revision level.
Parameter Updates	08Ch - 114h 88Ch - 912h	System parameters such as actual position, following error, and actual velocity are available for the host to read in this block.
Signature Window	115h - 11Fh	The Mx4 Octavia card writes a signature using ASCII codes to the signature window at power-up. The host may check for this signature to verify installation of an Mx4 Octavia card.
Contouring Ring Buffer	120h - 3C1h	This block of the DPR is reserved for contouring motion or coordinated move data points. The host downloads data to Mx4 Octavia via this "ring buffer."
RTC Window	3C2h - 3FBh	This window in the DPR serves as a buffer for the host to send RTCs to Mx4 Octavia.
Interrupt Registers	3FCh - 3FFh 1FFEh, 1FFFh	This block is used in the setting and re-setting of hardware interrupts to the host.
Parameter Registers	480h - 509h	System parameter registers such as analog feedback valves are available to the host in this block.

Table 6-1: Mx4 Octavia Dual Port RAM Blocks

Communication Protocols

In order to maintain a healthy communication interface between the host and Mx4 Octavia, some simple communication protocols must be adhered to by the host.

Each location in the DPR is labeled (see *Mx4 Octavia Dual Port RAM Organization*) with two read/write access codes. One code is for the host, one for Mx4 Octavia. The access codes are:

RO	:	read-only access
WO	:	write-only access
RW	:	read and write access

These "restrictions" are enforced only by convention in the host and Mx4 Octavia software. They are included to help the user understand how the various locations in the DPR are used by both the host and Mx4 Octavia.

Much of the data in the DPR that the host and Mx4 Octavia must read or write is multi-byte data (such as 32-bit actual position values) and thus requires multi-address accesses to the DPR. In order to ensure that multi-byte values are not corrupted by unsynchronized accesses, many DPR 'windows' are protected via 'access bytes'. See Fig. 6-3.

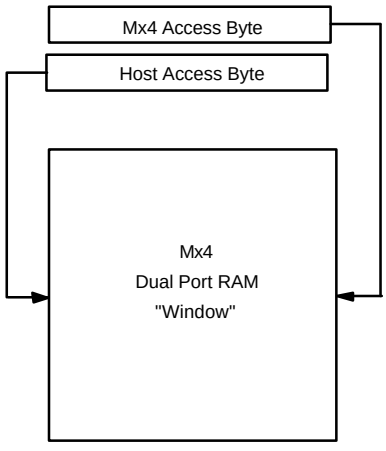


Fig. 6-3: Mx4 Octavia Dual Port RAM Access Bytes

Each 'window' includes both a host access byte and a Mx4 Octavia access byte, which are used to control access to the window. A more detailed explanation of how to use these bytes is offered in the *Mx4 Octavia Dual Port RAM Organization* section and will be discussed further in *Communication Protocols Revisited*.

Mx4 Octavia Dual Port RAM Organization

The 8K DPR is partitioned as follows in Tables 6-2 to 6-10. (The names given to individual bytes and groups of data are for reference only.)

Status Registers (000h - 065h)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
The status register bytes indicate the interrupt status of Mx4 Octavia. Mx4 Octavia updates or writes to these locations in an OR fashion. Thus, Mx4 Octavia does not reset interrupt status bits. The host, after reading or recognizing an interrupt status register, must reset bits at its own discretion.				
DSPSTAT1	000h	RW	RW	Mx4 Octavia status register for interrupts. Polled by the host to determine internal Mx4 Octavia status. bit 0: following error interrupt and halt bit 1: following error interrupt bit 2: index pulse interrupt bit 3: position breakpoint interrupt bit 4: motion complete interrupt bit 5: probe signal interrupt bit 6: conflicting commands (ignore the new motion-related command and send an interrupt) bit 7: RTC command is ignored because STOP is in progress

Table 6-2: Dual Port RAM Status Registers (continued on next page)

Status Registers (000h - 065h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
DSPSTAT2	009h	RW	RW	Mx4 Octavia status register for register to determine internal Mx4 Octavia status. bit Ø: encoder fault interrupt bit 1: unused bit 2: offset cancel finished bit 3: input programmed interrupt bit 4: DSPL host interrupt bit 5: DSPL program running error(s) (see location 00Fh) bit 6: 5ms interrupt bit 7: unused
The INTAXIS bytes code the source(s) of the interrupt(s) by setting a bit(s) (unless otherwise noted): bit 0: axis 1 bit 4: axis 5 bit 1: axis 2 bit 5: axis 6 bit 2: axis 3 bit 6: axis 7 bit 3: axis 4 bit 7: axis 8				
INTAXIS	001h	RW	RW	source of following error and halt interrupt
INTAXIS	002h	RW	RW	source of following error interrupt
INTAXIS	003h	RW	RW	source of index pulse interrupt
INTAXIS	004h	RW	RW	source of position breakpoint interrupt
INTAXIS	005h	RW	RW	source of motion complete interrupt
INTAXIS	006h	RW	RW	source of probe signal interrupt
INTAXIS	007h	RW	RW	source of conflicting commands interrupt
INTAXIS	008h	RW	RW	source of RTC command ignored because STOP is in progress
INTAXIS	00AH	RW	RW	source of encoder fault interrupt
reserved	00Bh	-	-	reserved location

Table 6-2 cont.: Dual Port RAM Status Registers (continued on next page)

Status Registers (000h - 065h) cont.

INTAXIS	00Ch	RW	RW	source of offset cancel finished
INTAXIS	00Dh	RW	RW	source of input programmed interrupt
INTAXIS	00Eh	RW	RW	DSPL Host Interrupt
DSPLERR	00Fh	RW	RW	DSPL running error description bit 0 : too many programs bit 1 : can't find program to stop bits 2-7 : not used
reserved	010h - 065h	-	-	reserved locations

Table 6-2 : Dual Port RAM Status Registers

DSPL Updates Window (066h - 085h)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
DSPLVAR	066h-06Bh	WO	RO	DSP variable monitor 1 (see MONITOR_VAR RTC description). LSB, ... , MSB (48-bit DSPL floating point)
DSPLVAR	06Ch-071h	WO	RO	DSP variable monitor 2 (see MONITOR_VAR RTC description). LSB, ... , MSB (48-bit DSPL floating point)
DSPLVAR	072h-077h	WO	RO	DSP variable monitor 3 (see MONITOR_VAR RTC description). LSB, ... , MSB (48-bit DSPL floating point)
DSPLVAR	078h-07Dh	WO	RO	DSP variable monitor 4 (see MONITOR_VAR RTC description). LSB, ... , MSB (48-bit DSPL floating point)
PRG1STAT	07Eh-07Fh	WO	RO	DSPL Motion Program 1 line number executing LSB, MSB
PRG2STAT	080h-081h	WO	RO	DSPL Motion Program 2 line number executing LSB, MSB
reserved	082h-083h	-	-	reserved locations
PLCSTAT	084h-085h	WO	RO	DSPL PLC Program line number executing LSB, MSB

Table 6-3: Dual Port RAM Hardware Signature Registers

Hardware Signature Window (086h - 08Bh)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
The SIGNATURE bytes contain the ASCII code controller card hardware signature.				
SIGNATURE	086h	WO	RO	Bus designator byte: ASCII "P" : PC/AT ASCII "V" : VME ASCII "M" : Multibus
SIGNATURE	087h	WO	RO	Hardware option byte: ASCII "I" : I/O integer 0 : standard configuration
SIGNATURE	088h	WO	RO	Revision byte: ASCII "A" : revision A ASCII "B" : revision B
SIGNATURE	089h - 08Bh	WO	RO	reserved for future options ... currently unused

Table 6-4: Dual Port RAM Hardware Signature Registers

Parameter Updates (08Ch - 114h)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
reserved	08Ch - 093h	-	-	Reserved locations
INPUT	094h	WO	RO	real-time status of inputs. A set bit indicates active: bit 0: IN0 bit 1: IN1 bit 2: IN2 bit 3: IN3 bit 4: IN4 bit 5: IN5 bit 6: IN6 bit 7: IN7
INPUT	095h	WO	RO	real-time status of inputs. A set bit indicates active: bit 0: IN8 bit 1: IN9 bit 2: IN10 bit 3: IN11 bit 4: IN12 bit 5: IN13 bit 6: IN14 bit 7: IN15
INPUT	096h	WO	RO	real-time status of inputs. A set bit indicates active: bit 0: IN16 bit 1: IN17 bit 2: IN18 bit 3: IN19 bit 4: IN20 bit 5: IN21 bit 6 IN22 bit 7: IN23

Table 6-5: Dual Port RAM Parameter Updates (continued on next page)

Parameter Updates (08Ch - 114h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
INPUT	097h	WO	RO	real-time status of inputs. A set bit indicates active: bit 0: IN24 bit 1: IN25 bit 2: IN26 bit 3: IN27 bit 4: IN28 bit 5: IN29 bit 6: IN30 bit 7: IN31
reserved	098h - 0A6h	-	-	reserved locations
PRB10-3	0A7h - 0AAh	WO	RO	axis 1 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB20-3	0ABh - 0AEh	WO	RO	axis 2 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB30-3	0AFh - 0B2h	WO	RO	axis 3 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB40-3	0B3h - 0B6h	WO	RO	axis 4 probe interrupt position LSB,...,MSB (32-bit two's complement)
PARACC	0B7h	WO	RW	Parameter readback window m echo
PARRDBK	0B8h - 0BFh	WO	RO	Parameter readback window (see PARREAD RTC description)
reserved	0C0h - 0C2h	-	-	reserved locations
The M4ACC bytes are used as access flags. When Mx4 Octavia needs to access a parameter update window, it sets the corresponding M4ACC byte to 01h. The host must test these flags to see if values can be written to or read from the parameter window in question. M4ACC=00h : Mx4 Octavia is not using window. Host may set corresponding HOSTACC=01h and may access window. M4ACC=01h : Mx4 Octavia is using window. Host must wait until this byte is cleared before accessing the window in question.				
M4ACC	0C3h	WO	RO	Mx4 Octavia is using 0D3h - 0E2h / 8D3h - 8E2h window
M4ACC	0C4h	WO	RO	Mx4 Octavia is using 0E3h - 0F2h / 8E3h - 8F2h window

Table 6-5 cont.: Dual Port RAM Parameter Updates (continued on next page)

Parameter Updates (08Ch - 114h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
M4ACC	0C5h	WO	RO	Mx4 Octavia 0F3h - 102h / 8F3h - 902h window
M4ACC	0C6h	WO	RO	Mx4 Octavia 103h - 112h / 903h - 912h window
M4ACC	0C7h	WO	RO	Mx4 Octavia 08Ch - 093h / 88Ch - 893h or 113h - 114h windows
M4ACC	0C8h	WO	RO	Mx4 Octavia 0A7h - 0B6h / 8A7h - 8B6h window
M4ACC	0C9h	WO	RO	Mx4 Octavia 066h - 085h / 866h - 885h window
M4ACC	0CAh	WO	RO	unused location
The HOSTACC bytes are used as access flags. When the host needs to access a parameter update window, it sets the corresponding HOSTACC byte to 01h. The Mx4 Octavia will test these flags to see if values can be written to or read from the parameter window in question. HOSTACC=00h : Host is not using window. Mx4 Octavia may set corresponding M4ACC=01h and may access window. HOSTACC=01h : Host is using window. Mx4 Octavia must wait until this byte is cleared before accessing the window in question.				
HOSTACC	0CBh	RO	RW	host is using 0D3h - 0E2h / 8D3h - 8E2h window
HOSTACC	0CCh	RO	RW	host is using 0E3h - 0F2h / 8E3h - 8F2h window
HOSTACC	0CDh	RO	RW	host is using 0F3h - 102h / 8F3h - 902h window
HOSTACC	0CEh	RO	RW	host is using 103h - 112h / 903h - 912h window
HOSTACC	0CFh	RO	RW	host is using 08Ch - 093h / 88Ch - 893h or 113h - 114h windows
HOSTACC	0D0h	RO	RW	host is using 0A7h - 0B6h / 8A7h - 886h window
HOSTACC	0D1h	RO	RW	host is using 066h - 085h / 866h - 885h window
HOSTACC	0D2h	RO	RW	unused location
See Chapter 5's *State Variables* for details concerning format of position and following error data.				

Table 6-5 cont.: Dual Port RAM Parameter Updates (continued on next page)

Parameter Updates (08Ch - 114h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
ACTPOS10-3	0D3h - 0D6h	WO	RO	axis 1 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS20-3	0D7h - 0DAh	WO	RO	axis 2 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS30-3	0DBh - 0DEh	WO	RO	axis 3 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS40-3	0DFh - 0E2h	WO	RO	axis 4 actual position LSB,...,MSB (32-bit two's complement)
Actual velocity is presented as a 32-bit integer value. In order to obtain units of encoder edge counts/sampling period for velocity, the host must use a division factor:				
Mx4 Octavia Only : 208 Vx4++ installed : 204				
ACTVEL10-3	0E3h - 0E6h	WO	RO	axis 1 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL20-3	0E7h - 0EAh	WO	RO	axis 2 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL30-3	0EBh - 0EEh	WO	RO	axis 3 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL40-3	0EFh - 0F2h	WO	RO	axis 4 actual velocity LSB,...,MSB (32-bit two's complement)
ERROR10-3	0F3h - 0F6h	WO	RO	axis 1 following error LSB,...MSB (32-bit two's complement)
ERROR20-3	0F7h - 0FAh	WO	RO	axis 2 following error LSB,...MSB (32-bit two's complement)
ERROR30-3	0FBh - 0FEh	WO	RO	axis 3 following error LSB,...MSB (32-bit two's complement)
ERROR40-3	0FFh - 102h	WO	RO	axis 4 following error LSB,...MSB (32-bit two's complement)
INDPOS10-3	103h - 106h	WO	RO	axis 1 index position LSB,...MSB (32-bit two's complement)
INDPOS20-3	107h - 10Ah	WO	RO	axis 2 index position LSB,...MSB (32-bit two's complement)
INDPOS30-3	10Bh - 10Eh	WO	RO	axis 3 index position LSB,...MSB (32-bit two's complement)

Table 6-5 cont.: Dual Port RAM Parameter Updates (continued on next page)

Parameter Updates (08Ch - 114h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
INDPOS40-3	10Fh - 112h	WO	RO	axis 4 index position LSB,...MSB (32-bit two's complement)
ENCSTAT	113h	WO	RO	Status of encoders. A set bit indicates a failure of the corresponding hardware item: bit 0 : axis 1 encoder bit 1 : axis 2 encoder bit 2 : axis 3 encoder bit 3 : axis 4 encoder bit 4 : axis 5 encoder bit 5 : axis 6 encoder bit 6 : axis 7encoder bit 7 : axis 8 encoder
SERVOCHK	114h	RW	RO	Real-time marker reporting. Aset bit indicates index pulse.

Table 6-5 cont.: Dual Port RAM Parameter Updates

Signature Window (115h - 11Fh)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
The SIGNATURE bytes contain the ASCII code controller card signature. The signature will be present if the card is operating correctly.				
SIGNATURE	115h	WO	RO	ASCII "M"
SIGNATURE	116h	WO	RO	ASCII "X"
SIGNATURE	117h	WO	RO	ASCII "4"
SIGNATURE	118h	WO	RO	integer part of dsp1 software version number
SIGNATURE	119h	WO	RO	decimal part of dsp1 software version number
SIGNATURE	11Ah	WO	RO	ASCII "+"
SIGNATURE	11Bh	WO	RO	integer part of dsp2 software version number
SIGNATURE	11Ch	WO	RO	decimal part of dsp2 software version number
SIGNATURE	11Dh	WO	RO	* ASCII "+"
SIGNATURE	11Eh	WO	RO	* integer part of Vx4++ version number
SIGNATURE	11Fh	WO	RO	* decimal part of Vx4++ version number
*** NOTE:** If the Vx4++ drive control option is not installed, DPR locations 11Dh - 11Fh will be zero.				

Table 6-6: Dual Port RAM Signature Window

Ring Buffer (120h - 3C1h)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
RINGBUF	120h - 3BFh	RO	WO	672 byte ring buffer that contains host contouring commands (4 bytes position, 4 bytes velocity) to be processed by the DSP. The length of all messages deposited by the host in this buffer must be a multiple of 8 bytes.
INPTR	3C0h	RO	RW	Pointer to the next free location in RINGBUF that the host will write to (expressed as an offset from the start of the buffer in multiples of 8 bytes).
OUTPTR	3C1h	RW	RO	Pointer to the next location in RINGBUF that Mx4 Octavia will read from (expressed as an offset from the start of the buffer in multiples of 8 bytes).

Table 6-7: Dual Port RAM Ring Buffer

Real Time Command (RTC) (3C2h - 3FBh)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
RTC	3C2h	RW	RW	RTC byte. If this byte is non-zero, Mx4 Octavia will interpret it as a command and execute it, using the following locations as its arguments. When execution is complete and Mx4 Octavia is ready for the next command, the byte will be set to zero.
ARGMNTS	3C3h - 3FBh	RO	WO	57 bytes of argument storage area. The usage of this area depends on the RTC to be executed. The host must set up the argument area correctly before writing a command code to this byte to ensure that Mx4 Octavia reads the arguments properly.

Table 6-8: Dual Port RAM Real Time Command (RTC)

Interrupt Registers (3FCh - 3FDh, 1FFEh, 1FFFh)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
MINTACC	3FCh	RW	RO	A Mx4 Octavia access flag byte. When Mx4 Octavia needs to access location 1FFEh or the status registers window (000h - 065h), it sets this byte equal to 01h. The host must test this flag to see if values can be written to or read from the window.
HOSTINT1	1FFEh	RW	RW	bit 0: interrupt source is buffer breakpoint bit 1: interrupt related to DSPSTAT1 register bit 2: ESTOP is detected bit 3: vector change buffer is overflowed bit 4: reset finished bit 5: data runout in ring buffer (abs[vel]>0) bit 6: interrupt related to DSPSTAT2 register bit 7: unused
DSPINT	1FFFh	RW	WO	When the host sets a bit(s) in this location, a Mx4 Octavia interrupt will be generated. The interrupt remains in force until Mx4 Octavia accesses this location. This register currently not used.
reserved	3FEh - 3FFh	-	-	reserved locations
Read accesses to locations 1FFEh do not require use of the MINTACC access byte (due to its single byte status). All write accesses to 1FFEh must use the MINTACC byte, however. (See *Handling Mx4 Octavia Software/Hardware Interrupts*)				

Table 6-9: Dual Port RAM Interrupt Registers (continued on next page)

Parameter Registers (400h - 7FDh)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
reserved	400h - 47Fh	-	-	reserved locations
Mx4ACC	480h	WO	RO	Mx4 Octavia is using 482h-485h window
HOSTACC	481h	RO	RW	host is using 482h-485h window
ICUBCOUNT	482h - 483h	WO	RO	internal cubic spline counter (see CUBIC_TSCALE) LSB, MSB
CAMCOUNT1	484h – 485h	WO	RO	Slave axis 1 table index counter
CAMCOUNT2	486h – 487h	WO	RO	Slave axis 2 table index counter
CAMCOUNT3	488h – 489h	WO	RO	Slave axis 3 table index counter
CAMCOUNT4	48Ah – 48Bh	WO	RO	Slave axis 4 table index counter
CAMCOUNT5	48Ch – 48Dh	WO	RO	Slave axis 5 table index counter
CAMCOUNT6	48Eh – 48Fh	WO	RO	Slave axis 6 table index counter
CAMCOUNT7	490h – 491h	WO	RO	Slave axis 7 table index counter
CAMCOUNT8	492h – 493h	WO	RO	Slave axis 8 table index counter
reserved	494h - 4FFh	-	-	reserved locations
MX4ACC	500h	WO	RO	Mx4 Octavia is using 502h-509h window
HOSTACC	501h	RO	RW	host is using 502h-509h window
The analog feedbacks ADC1 - ADC4 require the ACC4000 daughter board with analog option. ADCx is represented as a 16-bit two's compliment value				
ADC1	502h - 503h	WO	RO	ADC1 value LSB, MSB
ADC2	504h - 505h	WO	RO	ADC2 value LSB, MSB
ADC3	506h - 507h	WO	RO	ADC3 value LSB, MSB
ADC4	508h - 509h	WO	RO	ADC4 value LSB, MSB
reserved	50Ah - 88Bh	-	-	reserved

Table 6-10: Parameter Registers

Parameter Updates (88Ch - 912h)

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	

PRB50-3	8A7h - 8AAh	WO	RO	axis 5 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB60-3	8ABh - 8AEh	WO	RO	axis 6 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB70-3	8AFh - 8B2h	WO	RO	axis 7 probe interrupt position LSB,...,MSB (32-bit two's complement)
PRB80-3	8B3h - 8B6h	WO	RO	axis 8 probe interrupt position LSB,...,MSB (32-bit two's complement)
ACTPOS50-3	8D3h - 8D6h	WO	RO	axis 5 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS60-3	8D7h - 8DAh	WO	RO	axis 6 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS70-3	8DBh - 8DEh	WO	RO	axis 7 actual position LSB,...,MSB (32-bit two's complement)
ACTPOS80-3	8DFh - 8E2h	WO	RO	axis 8 actual position LSB,...,MSB (32-bit two's complement)
ACTVEL50-3	8E3h - 8E6h	WO	RO	axis 5 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL60-3	8E7h - 8EAh	WO	RO	axis 6 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL70-3	8EBh - 8EEh	WO	RO	axis 7 actual velocity LSB,...,MSB (32-bit two's complement)
ACTVEL80-3	8EFh - 8F2h	WO	RO	axis 8 actual velocity LSB,...,MSB (32-bit two's complement)

Table 6-10: Parameter Registers (continued on next page.)

Parameter Updates (88Ch - 912h) cont.

NAME	ADDRESS	ACCESS		DESCRIPTION
		Mx4	HOST	
ERROR50-3	8F3h - 8F6h	WO	RO	axis 5 following error LSB,...MSB (32-bit two's complement)
ERROR60-3	8F7h - 8FAh	WO	RO	axis 6 following error LSB,...MSB (32-bit two's complement)
ERROR70-3	8FBh - 8FEh	WO	RO	axis 7 following error LSB,...MSB (32-bit two's complement)
ERROR80-3	8FFh - 902h	WO	RO	axis 8 following error LSB,...MSB (32-bit two's complement)
INDPOS50-3	903h - 906h	WO	RO	axis 5 index position LSB,...MSB (32-bit two's complement)
INDPOS60-3	907h - 90Ah	WO	RO	axis 6 index position LSB,...MSB (32-bit two's complement)
INDPOS70-3	90Bh - 90Eh	WO	RO	axis 7 index position LSB,...MSB (32-bit two's complement)
INDPOS80-3	90Fh - 912h	WO	RO	axis 8 index position LSB,...MSB (32-bit two's complement)
reserved	913h -1FFDh	-	-	reserved locations

Table 6-10: Parameter Registers

Communication Protocols Revisited

As is evident in *Mx4 Octavia Dual Port RAM Organization*, many "windows" in the DPR are protected with access bytes. Table 6-11 lists each of the protected windows and its corresponding access bytes:

WINDOW	WINDOW DESCRIPTION	Mx4 ACCESS BYTE	HOST ACCESS BYTE
000h - 065h, 1FFEh	Interrupt and Status Registers	3FCh	3FDh
08Ch - 093h	Vx4++ Variable Viewing	0C7h	0CFh
0D3h - 0E2h / 8D3h - 8E2h	Actual Position 1- 8	0C3h	0CBh
0E3h - 0F2h / 8E3h - 8F2h	Actual Velocity 1-8	0C4h	0CCh
0F3h - 102h / 8F3h - 902h	Following Error 1-8	0C5h	0CDh
103h - 112h / 903h - 912h	Index Position 1-8	0C6h	0CEh
113h - 114h	Encoder/Marker	0C7h	0CFh
0A7h - 0B6h / 8A7h - 8B6h	Probe Position 1-8	0C8h	0D0h
066h - 085h	DSPL Updates	0C9h	0D1h
482h - 485h	Cubic Spline / CAM Counters	480h	481h
502h - 509h	Analog Feedback	500h	501h

Table 6-11: Access Bytes for DPR Windows

(It is noteworthy to remember that single byte values in the DPR are always protected by the arbitration hardware built into the DPR.)

A typical protocol that a host would use to access a 'protected' window would be:

1. Host writes 01h to the host access byte
2. Host polls the DSP access byte until it reads 00h
3. Host accesses window
4. When the host is finished with the window, host writes 00h to the host access byte.

Following this convention when accessing the windows listed in the above table ensures data integrity.

The RTC Window of the DPR is not protected with the access byte scheme; however, it does use a different access protocol. As will become evident in later

sections, Mx4 Octavia checks for RTCs by looking for a host-written command code in location 3C2h. If Mx4 Octavia detects a command code, it interrupts the RTC data and when finished, writes a zero to the RTC command code register (3C2h). Therefore, the host knows it can send an RTC command code only when location 3C2h has a zero value and Mx4 Octavia knows there is an RTC to process when location 3C2h contains a non-zero command code. The host should follow a procedure when writing RTCs to the DPR such as:

1. Host polls location 3C2h until it reads 00h
2. Host writes RTC data to locations 3C3h + as needed
3. Host writes RTC command code to location 3C2h

Handling Mx4 Octavia Software / Hardware Interrupts

Mx4 Octavia signals interrupts to the host computer through both hardware and software. The host has the option of responding to "hardware" interrupts across the bus via an interrupt source routine or simply polling for "software" interrupts in the Mx4 Octavia DPR.

Mx4 Octavia signals interrupts to the host by setting a bit(s) in the DPR's 1FFEh location. This in turn generates a hardware interrupt to the host via the bus interrupt signals. The interrupt type and interrupt source information from Mx4 Octavia is written to the Status Registers Block of the DPR via the DSPSTAT1, DSPSTAT2, and INTAXIS registers.

The host may check for interrupts by "software" by polling location 1FFEh for set bits. An important exception to the communication protocols of the previous section is made here: read accesses (or polling) to location 1FFEh do not require the use of the access bytes due to its single byte status. All host-write accesses to 1FFEh must, however, use the MINTACC access byte. If the host detects an interrupt by polling 1FFEh, it may interrogate the proper status registers (remember to use the access bytes) for interrupt type and source information.

If the host incorporates an interrupt source routine responding to bus interrupt signals, the access to 1FFEh serves two purposes. First, an access to the 1FFEh location terminates the hardware interrupt. Second, the 1FFEh byte bit codes some interrupt source data which directs the host as to which status registers

should be interrogated for further interrupt type and source information. For example, if an axis z motion complete interrupt occurs, bit 1 of location 1FFEh(HOSTINT) is set. The interrupt type is coded with a set bit 4 of DSPSTAT1 (000h). The source, axis 2, is evident as bit 1 of location 005h is set.

Note: It is important to remember that Mx4 Octavia does not reset interrupt status register bits or 1FFEh location bits. The host, after reading or recognizing an 'interrupt' location must reset bits of its own discretion.

Mx4 Octavia Host Programming ... RTCs & Contouring

Mx4 Octavia programming includes both contouring and RTC Mx4 Octavia modes of motion. Typical programming applications consist of a combination of contouring and RTCs, and any combination of the two types of commands is possible for the four axes.

Real-Time Commands

Real-Time Commands (RTCs) are sent to Mx4 Octavia via the Real-Time Command Window in the DPR (locations 3C3h to 3FBh). RTCs consist of a single byte command code and an argument list. As was introduced in *Communication Protocols Revisited*, the host should follow this procedure when writing RTCs to the DPR:

1. **Host polls location 3C2h until it reads 00h ***
2. **Host writes RTC arguments to locations 3C3h + (as needed)**
3. **Host writes RTC command code to location 3C2h (RTC command code register)**

Note: Mx4 Octavia polls for RTCs by checking location 3C2h for a valid command code. If a valid code is detected, Mx4 Octavia interprets the RTC and writes 00h to location 3C2h. The host should verify that Mx4 Octavia has executed a previously transmitted RTC before writing another by checking the RTC command code register for value 00h.

When writing an RTC argument list to the RTC Window, the host must follow these rules:

1. RTC argument list always starts at DPR location 3C3h.

2. RTC arguments must be written to the DPR in the order they appear as arguments in the "ARGUMENTS" declaration of RTC listing (see *Mx4 Octavia Host-Based Programming Command Listing* in Chapter 5).

3. When writing multi-byte value RTC arguments, write LSB to MSB order.

4. Argument lists for multi-axis RTCs must be written to the DPR in increasing-axis-number order.

These rules are illustrated in the following examples that depict RTCs written to the Mx4 Octavia DPR Real-Time Command Window.

Note: DPR locations marked "xxh" or "not necessary" in the following examples do not need to be written to (for the examples in question). Mx4 Octavia determines which locations contain valid data via the command code and n argument (if any).

Example 1

Preset axis 3 position to 00112233h.

Use the POS_PRESET RTC,

n : 04h
$pset_3$: 00112233h

The host must write to the following DPR locations as specified:

DPR ADDRESS	BYTE	SYMBOL	DESCRIPTION
03C2h	68h	-	RTC command code
03C3h	04h	n	single byte form
03C4h	33h	$pset_3$	low byte of low word
03C5h	22h	$pset_3$	high byte of low word
03C6h	11h	$pset_3$	low byte of high word
03C7h	00h	$pset_3$	high byte of high word
03C8h	xxh	-	not necessary
:	xxh	-	not necessary
03FBh	xxh	-	not necessary

Example 2

Assuming current positions of zero for axes 2 and 4, we want to move axis 2 to the target position of 234567h and axis 4 to the target position of 112233h. Let's also assume that we want this move to be accomplished with the slew rate velocity of 200000h (200000h/2^{16} counts/200 µsec) and acceleration of 150h (150h/2^{15} counts/(200 µsec)^2) for both axes. The values for the data parameters are:

Use the AXMOVE RTC,

n	:	0Ah
acc_2	:	0150h
pos_2	:	00234567h
vel_2	:	00200000h
acc_4	:	0150h
pos_4	:	00112233h
vel_4	:	00200000h

The host must write to the following DPR locations as specified:

DPR ADDRESS	BYTE	SYMBOL	DESCRIPTION
03C2h	60h	-	RTC command code
03C3h	0Ah	n	single byte for n
03C4h	50h	acc_2	low byte
03C5h	01h	acc_2	high byte
03C6h	67h	pos_2	low byte of low word
03C7h	45h	pos_2	high byte of low word
03C8h	23h	pos_2	low byte of high word
03C9h	00h	pos_2	high byte of high word
03CAh	00h	vel_2	low byte of low word
03CBh	00h	vel_2	high byte of low word
03CCh	20h	vel_2	low byte of high word
03CDh	00h	vel_2	high byte of high word
03CEh	50h	acc_4	low byte
03CFh	01h	acc_4	high byte
03D0h	33h	pos_4	low byte of low word
03D1h	22h	pos_4	high byte of low word
03D2h	11h	pos_4	low byte of high word
03D3h	00h	pos_4	high byte of high word
03D4h	00h	vel_4	low byte of low word

03D5h	00h	vel_4	high byte of low word
03D6h	20h	vel_4	low byte of high word
03D7h	00h	vel_4	high byte of high word
03D8h	xxh	-	not necessary
:	xxh	-	not necessary
03FBh	xxh	-	not necessary

Example 3

Set a following error interrupt at 100, 101, 102, and 103 counts for axes 1 through 4, respectively.

Use the EN_ERR RTC,

n : 0Fh
fer_1 : 0064h
fer_2 : 0065h
fer_3 : 0066h
fer_4 : 0067h

The host must write to the following DPR locations as specified:

DPR ADDRESS	BYTE	SYMBOL	DESCRIPTION
03C2h	67h	-	RTC command code
03C3h	0Fh	n	single byte for n
03C4h	64h	fer_1	low byte
03C5h	00h	fer_1	high byte
03C6h	65h	fer_2	low byte
03C7h	00h	fer_2	high byte
03C8h	66h	fer_3	low byte
03C9h	00h	fer_3	high byte
03CAh	67h	fer_4	low byte
03CBh	00h	fer_4	high byte
03CCh	xxh	-	not necessary
:	xxh	-	not necessary
03FBh	xxh	-	not necessary

Contouring

Contouring commands consist of segment move commands transferred from the host to Mx4 Octavia via the Contouring Ring Buffer in the DPR (locations 120h to 3BFh). Each segment move consists of a 32-bit position value and 32-bit velocity value for each axis included in the contouring motion. The ring buffer size is 672 bytes, and thus will hold 84 'segment move [position, velocity] commands'.

Mx4 Octavia performs either 2nd order or cubic spline interpolation on the 8-byte segment move data points. The interpolation time interval is programmable via the BTRATE (2nd order) or CUBIC_RATE (cubic spline) commands. The segment move 'commands' are executed in sequence, with execution commencing only when the previously commanded segment move is complete.

Before beginning a contour, the ring buffer must first be initialized with contouring points (segment move commands). The host must load the segment move commands into the DPR in round-robin format. The following rules must be followed when initializing the ring buffer with data.

1. Data should begin in the segment command data area indicated by the value of the Mx4 Octavia pointer OUTPTR, loaded at incrementing addresses.

2. Position and velocity are interleaved, position first.

3. Multi-byte position and velocity are written to the ring buffer in LSB to MSB format.

4. For multi-axis contouring, the position/velocity pairs for each axis involved are interleaved, written to the ring buffer in increasing-axis-number order.

5. Host should update the value of the host pointer INPTR to [offset of last segment move command + 1].

Note: The ring buffer is structured such that the next byte after location 3BFh is at location 120h; so the host must implement a roll-over to 120h after location 3BFh when loading data to the ring buffer (and/or a roll-over from 83 to 0 in the value of host pointer INPTR).

These rules are illustrated in the two examples of Fig. 6-4.

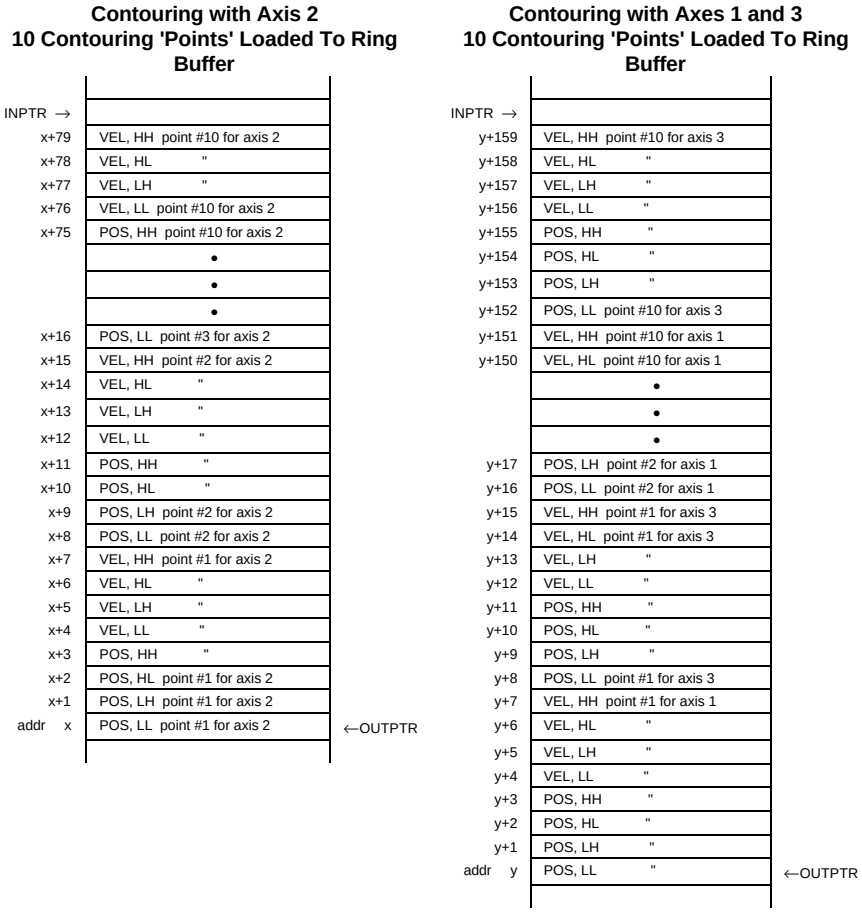


Fig. 6-4: Contouring Ring Buffer Example

As is evident in Rules 1 and 2, the DPR ring buffer area includes two pointers, INPTR and OUTPTR. Both pointers have values expressed as an offset from 120h in multiples of 8 bytes (or a single segment move command). The pointer values range from 0 to 83, pointing to one of the 84-segment move command data areas in the ring buffer.

OUTPTR is an Mx4 Octavia pointer. It indicates which of the 84 data areas of the ring buffer Mx4 Octavia will read the next segment move command from. Mx4 Octavia increments OUTPTR as it reads data from the ring buffer. INPTR is a host pointer. Its value indicates which data in the ring buffer the host should begin to download additional contour data points.

Before initializing the ring buffer with contour data points, INPTR should equal OUTPTR indicating that the ring buffer is empty. The host increments INPTR as the ring buffer is initialized with data. After ring buffer initialization, it is essential to ensure that INPTR always leads OUTPTR so that Mx4 Octavia is never starved of segment move commands to read after a START RTC is issued.

The host may issue a buffer breakpoint interrupt command (EN_BUFBRK RTC) in order that Mx4 Octavia interrupts the host whenever the number of segment move commands in the ring buffer falls below a programmed threshold. This provides a system of informing the host when it should refresh the ring buffer with additional segment move commands. The number of segment move commands indicated by EN_BUFBRK must always be greater than the number of segment move commands read by Mx4 Octavia during a ring buffer refresh to prevent starvation.

When refreshing the ring buffer with additional segment move commands, Rules 2 through 5 should still be followed. Rule 1 is altered as follows:

1. Data should begin in the segment move command data area indicated by value of host pointer, INPTR.

The START RTC starts the contouring motion with the argument of START being a bit coding of the axes involved. Fig. 6-5 depicts a flowchart for a general host contouring algorithm.

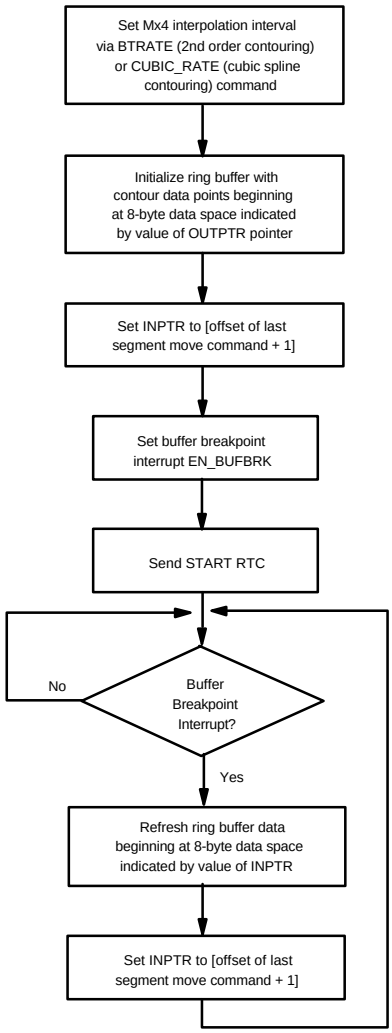


Fig. 6-5: Ring Buffer Contouring Algorithm Flowchart

Mx4 Octavia allows the axes involved in a contouring motion to be changed "on the fly" via the VECCHG (vector change) command (2nd order) or velocity argument bit coding (cubic spline). For 2nd order contouring, upon the execution of VECCHG, the contouring task assumes a new set of axes at the programmed segment move command data space in the ring buffer. The

VECCHG is triple buffered in Mx4 Octavia, so up to three vector changes may be queued at any time. Mx4 Octavia sends a "vector change buffer overflown" interrupt to the host if the host attempts to queue more than three vector changes.

Changing contouring axes with cubic spline requires only a change in the axis-coding bits in the velocity argument upper nibble (see *Cubic Spline Application Notes*).

Contouring motion in a particular axis may be terminated with a STOP command, or if the emergency stop ESTOP_ACC input is active. Attempting to execute a closed loop motion command such as VELMODE or AXMOVE for an axis while that axis is involved in contouring motion (or vice versa) will result in a "conflicting commands detected" host interrupt and the second command will be ignored.

Mx4 Octavia Host Programming Using C, C++, Visual Basic or Visual C++

Programming for the Mx4 Octavia card with the Host programming method may involve many of the following programming items:

- transmitting RTCs to Mx4 Octavia
- sending contouring commands to Mx4 Octavia in contouring applications
- create an interrupt service routine to process any Mx4 Octavia-to-host interrupts
- check Mx4 Octavia status bytes and error codes
- read Mx4 Octavia system state variables such as position, velocity and following error for user-feedback
- utilize the PARREAD RTC for debug support

Mx4 Octavia's powerful instruction set and comprehensive DPR data reporting format enables the user to create application programs from the very simple to the complex. The experienced programmer may want to write low-level code that deals with Mx4 Octavia at the bit level through the 8K DPR interface. Others, however, might want to start out with higher-level Mx4 Octavia programming; utilizing pre-defined functions and routines that take care of the

lower-level Mx4 Octavia programming aspects such as reading and writing bytes and utilizing the correct Mx4 Octavia DPR communication protocols.

For further information about programming the Mx4 Octavia with C, please refer to the *Mx4 & C Programmer's Guide*.

For further information about programming the Mx4 Octavia with C++, Visual Basic, or Visual C++, contact DSP Control Group.

Mx4 Octavia Power-Up / Reset Software Initialization

A typical Mx4 Octavia host programming application program should include a standard initialization routine. Upon power-up, the Mx4 Octavia card resets itself just as it would as if it had received the RESET command from the host. After the Mx4 Octavia reset sequence is completed (signaled by 'reset complete' interrupt to the host), the Mx4 Octavia (and VECTOR, if installed) signature is written to the DPR. If the post-reset signature is not complete, the reset was unsuccessful and the Mx4 Octavia system reset should be repeated. Following this initialization sequence ensures that Mx4 Octavia (and Vx4++, if installed) has been reset and is ready to be programmed before the host programming commences. The suggested initialization sequence is depicted in the flowchart of Fig. 6-6.

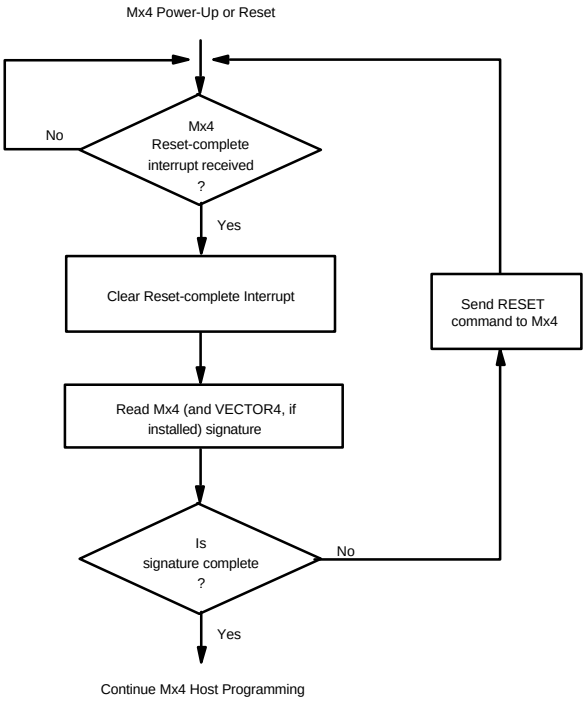


Fig. 6-6: Mx4 Octavia Power-Up / Reset Software Initialization Routine

7 Mx4 Octavia Status & Error Reports

Mx4 Octavia Power-Up / Reset State

Upon power-up, the Mx4 Octavia card resets itself just as it would as if it had received the RESET RTC from the host. Upon completion of the reset, Mx4 Octavia sends the 'reset finished' interrupt to the host. Thus, upon power-up of the Mx4 Octavia card, the host should wait for and then clear the 'reset finished' interrupt generated by Mx4 Octavia. (See Chapter 6, *Mx4 Octavia Power-Up / Reset Software Initialization*.)

An Mx4 Octavia reset results in the clearing of all Mx4 Octavia parameters including the 8K Dual Port RAM (DPR). Any programming of the Mx4 Octavia card prior to reset must be repeated.

A reset, however, does not clear or alter in any way any previously loaded DSPL program, compensation tables, cam tables, or internal cubic spline data table. These tables may be cleared with their respective “CLEAR_” host programming commands.

Mx4 Octavia Interrupts, Status Codes & Error Condition Reports to the Dual Port RAM

Mx4 Octavia data reporting to the host includes a variety of interrupt conditions, status codes, and error conditions. Table 7-1 sums up these reports.

Note: Refer to Chapter 6 *Mx4 Octavia Host-Based Programming* for a detailed description of the Mx4 Octavia DPR interface and protocols for host writing and reading of the DPR.

NAME	HOST INTERRUPT	RTC ENABLED	DESCRIPTION
Following Error and Halt Interrupt	√	√	see EN_ERRHLT command description
Following Error Interrupt	√	√	see EN_ERR command description
Index Pulse Interrupt	√	√	see EN_INDEX command description
Position Breakpoint Interrupt	√	√	see EN_POSBRK command description
Motion Complete Interrupt	√	√	see EN_MOTCP command description
External Interrupt	√	√	see EN_PROBE command description
Conflicting Commands Detected	√		"Conflicting commands detected" is an error interrupt reported to the host via bit 6 of DPR status register DSPSTAT1 (see *Mx4 Octavia DPR Organization*). This interrupt serves to report the host error of attempting to combine both contouring and RTC motion in the same axis simultaneously. Any one of the following cases will yield a "conflicting commands detected" interrupt. For an axis with 1. contouring in progress, and an AXMOVE, REL_AXMOVE or VELMODE RTC involving that axis is sent to Mx4 Octavia 2. AXMOVE, REL_AXMOVE or VELMODE motion in progress, and a contouring START RTC involving that axis is sent to Mx4 Octavia. The "conflicting command(s)" that caused the error is not executed by Mx4 Octavia, it is discarded.

Table 7-1: Mx4 Octavia Interrupts, Status Codes and Error Conditions (continued on next page)

NAME	HOST INTERRUPT	RTC ENABLED	DESCRIPTION
RTC Command Ignored	√		This error interrupt is reported to the host via bit 7 of DPR status register DSPSTAT1 (see *Mx4 Octavia DPR Organization*). The "RTC command ignored" interrupt is generated when a motion command is received by Mx4 Octavia for an axis that is presently halting to a stop via a previously executed STOP RTC or active ESTOP_ACC input. The motion commands are the AXMOVE, REL_AXMOVE, START or VELMODE RTCs.
5 msec Timer Interrupt	√	√	see INT5MS command description
DSPL - host	√	DSPL √	An executing DSPL program may signal an interrupt to the host with the DSPL INT_HOST command.
DSPL program errors	√		An executing DSPL program can generate 'running fault' interrupts to the host if: 1. DSPL attempts to run more than 3 programs simultaneously. 2. DSPL attempts to halt a program which is not running.
Offset Cancel Finished	√	√	see OFFSET command description
Encoder Status			The "encoder status" is reported to DPR location 113h (see *Mx4 Octavia DPR Organization*). A set bit indicates that Mx4 Octavia has detected an encoder hardware failure. Mx4 Octavia reports an "encoder status" error if for the axis in question: 1. the following error count is > 300h, and 2. the encoder feedback to Mx4 Octavia is losing encoder pulses or one of the encoder signals (A or B) actively toggles while the other one is inactive.
Real Time Index Pulse Reporting			In addition to the host-enabled index pulse interrupt, the host may monitor the index pulses of all four axes via the "Real time index pulse reporting" in the DPR location 114h (see *Mx4 Octavia DPR Organization*). A set bit indicates an index pulse and the nibble is updated in real-time (5 msec).

Table 7-1 cont.: Mx4 Octavia Interrupts, Status Codes and Error Conditions (continued on next page)

NAME	HOST INTERRUPT	RTC ENABLED	DESCRIPTION
Buffer Breakpoint Interrupt	√	√	see EN_BUFBRK command description
ESTOP Detected	√		Mx4 Octavia reports the occurrence of an emergency stop (ESTOP_ACC input) to the host with an interrupt. The interrupt is reported via bit 2 of DPR interrupt register HOSTINT (see *Mx4 Octavia DPR Organization*). When Mx4 Octavia detects an active ESTOP_ACC input, the motion (if any) of all four axes is brought to a halt with the programmed ESTOP_ACC acceleration / deceleration.
Vector Change Buffer Overflown	√		The VECCHG RTC utilizes three buffer levels to implement the instruction. If the host attempts to "stack" or buffer more than three VECCHG commands, a host interrupt is generated and reported via bit 3 of DPR interrupt register HOSTINT (see *Mx4 Octavia DPR Organization*).
Reset Finished	√		Upon completion of the power-up sequence or RESET RTC, Mx4 Octavia signals an interrupt to the host. The interrupt is bit-coded as a set bit 4 of DPR interrupt register HOSTINT (see *Mx4 Octavia DPR Organization*).
Data Run-Out In Ring Buffer	√		It is essential for the host to provide Mx4 Octavia segment move commands via the ring buffer while contouring is in progress. If at any time while contouring is active Mx4 Octavia detects INPTR=OUTPTR and any activated axis velocity is not zero, the ring buffer is out of data and Mx4 Octavia sends a "data run-out in ring buffer" interrupt to the host. The interrupt is recorded in bits of DPR interrupt register HOSTINT (see *Mx4 Octavia DPR Organization*).

Table 7-1 cont.: Mx4 Octavia Interrupts, Status Codes and Error Conditions

8 Mx4 Octavia RAM Memory Organization

The Mx4 Octavia controller includes a 32k word static RAM memory (with optional battery back-up). The memory is utilized for the following features:

DSPL program storage
position compensation table storage
velocity compensation table storage
CAM table storage
Internal Cubic Spline data storage
DSPL variable table storage

The memory usage is not self-exclusive to each of the above features. Rather, the memory space is shared and does overlap. That is, some features use the same memory space as others, and may not be used simultaneously. Table 8-1 illustrates Mx4 Octavia's RAM memory organization.

Mx4 Octavia's RAM Memory Organization

DESCRIPTION	# WORDS	RAM ADDRESS RANGE
Position Compensation Table 1	1024	2000h - 23FFh
Position Compensation Table 2	1024	2800h - 2BFFh
Position Compensation Table 3	1024	3000h - 33FFh
Position Compensation Table 4	1024	3800h - 3BFFh
Position Compensation Table 5	1024	400h – 43FFh
Position Compensation Table 6	1024	4800h – 4BFFh
Position Compensation Table 7	1024	5000h – 53FFh
Postion Compensation Table 8	1024	5800h – 5BFFh
Velocity Compensation Table 1	1024	2400h - 27FFh
Velocity Compensation Table 2	1024	2C00h - 2FFFh
Velocity Compensation Table 3	1024	3400h - 37FFh
Velocity Compensation Table 4	1024	3C00h - 3FFFh
Velocity Compensation Table 5	1024	4400h – 47FFh
Velocity Compensation Table 6	1024	4C00h – 4FFFh
Velocity Compensation Table 7	1024	5400h – 57FFh
Velocity Compensation Table 8	1024	5C00h – 5FFFh
Cam Table 1	8192	2000h – 3FFFh
Internal Cubic Spline Table	8192	2000h - 3FFFh
DSPL Table Storage [TABLE_P, TABLE_V]	8192	2000h - 3FFFh

Table 8-1: Mx4 Octavia's RAM Memory Organization

9 NURBS

Introduction

Non-**U**niform **R**ational **B**-**S**plines, referred to as NURBS, have become the *de facto* industry standard for the representation and data exchange of geometric information processed by computers. Many international standards such as IGES, STEP, and PHIGS, recognize NURBS as a powerful tool for geometric designs.

The power of NURBS is largely due to:

- NURBS algorithms are fast and numerically stable
- Designing with NURBS is intuitive; almost every tool and algorithm has easy-to-understand geometric interpretation.
- NURBS provide a unified mathematical basis for representing both analytical shapes (conical sections such as circles, ellipses, parabolas etc.), as well as free-form curves (e.g. complex curvatures on car bodies.)
- NURBS can represent very complex shapes with remarkably little data.
- The NURBS model allows you to define curves with no kinks or sudden changes of direction (such as airplane-wing cross section) or with precise control over where kinks and bends occur (sharp corners of machined object, for instance).

The excellent properties, combined with successful industrial applications, have contributed to the popularity of NURBS to the point that NURBS in the CAD/CAM world has been compared to the English language in science and business.

Many CAD/CAM drawing packages offer low-level geometric primitives for objects such as lines, points and triangles. Because the representation of these objects are mathematically exact -- lines being defined by their two points, and so forth -- their resolution is independent and unaffected by changes in

position, scale, or orientation. The low-level primitives can also be used to define arbitrarily shaped objects, such as a football. However, at the cost of these mathematical properties; for example, a circle that is approximated by a sequence of line segments will change shape when rotated. The main advantage of NURBS is that they offer a way to represent arbitrary shapes while maintaining mathematical exactness and resolution independence.

How to Characterize the NURBS curves

In order to build up a definition for NURBS curve, we must first review parametric functions. Using direct function to represent a curve has its drawback: since we can have only one *y* for a given *x*, our curve can not loop back onto itself. An alternative method is to define the curve with parametric function. In general such function has the form:

$$Q(t) = \{ x(t), y(t) \}$$

This means, for each *t* there is a value for *x*, *y* and Q. For example defining:

$$x(t) = cos(t)$$
$$y(t) = sin(t)$$

produces a circle as can be verified by plugging in values between 0 and 2π for *t*. By evaluating function Q at a number of *t* points we will get a series of {*x*,*y*} particles that we can use to plot our curves as shown in figure 9-1.

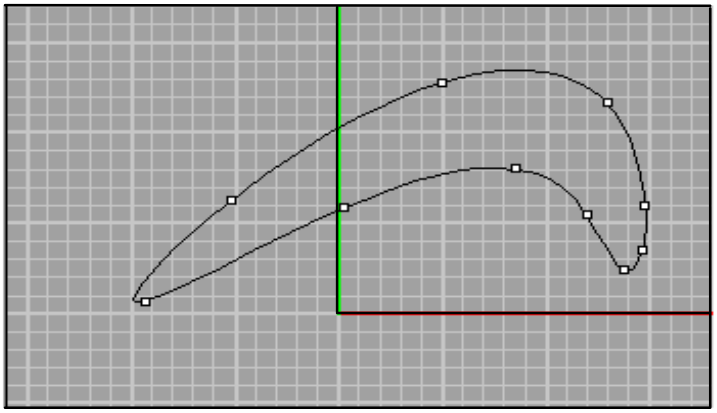


Figure 9-1: a curve obtained by parametric function Q

Control Points

One key character of NURBS curve is that its shape is determined by (among other things) the positions of a set of points called figure *control points.* As in Figure 9-2, the control points are often joined with connecting lines to make them easier to see and to clarify their relationship to curve. These connecting lines form what is known as *control polygon.*

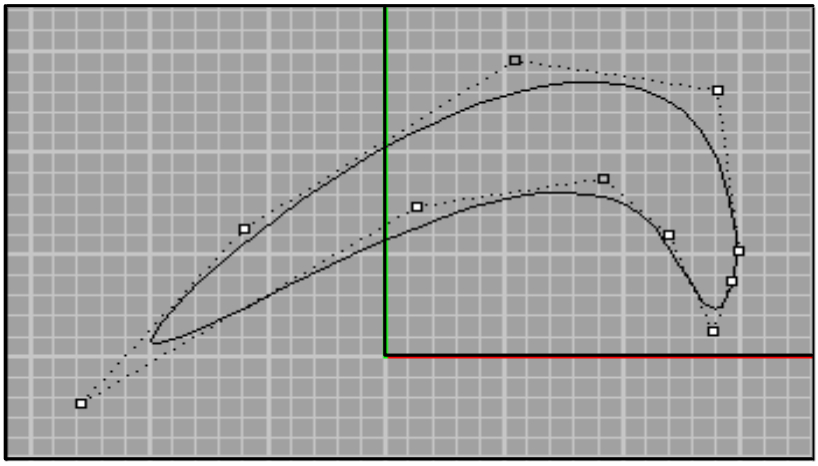


Figure 9-2: The control points and control polygon describing a closed contour

Figure 9-3 describes Figure 9-2 with one of the control points pulled to the right a bit. Notice that the curve's shape isn't changed throughout its entire length, but near the changed control point.

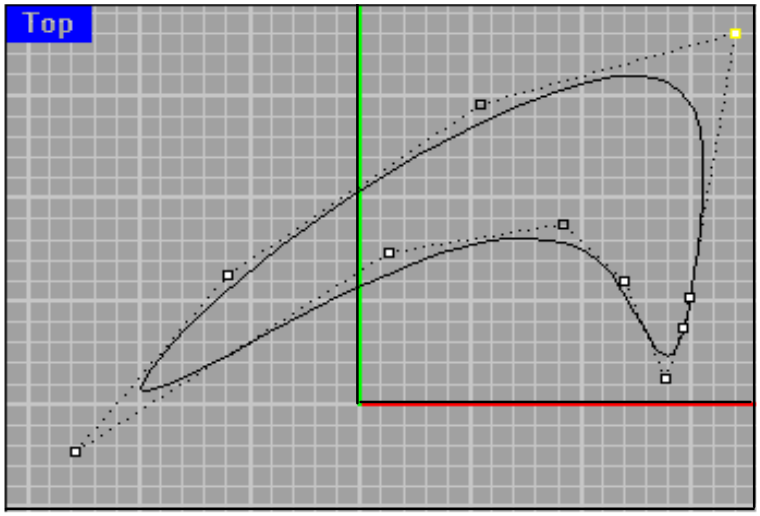


Figure 9-3: The closed contour of figure 2 with highlighted control point pulled to the right

At any point in time, *t*, the evaluated parametric function will be a weighted average of all control points, but with the point closer to the particle carrying more weight than those farther away.

Basis Functions

Each control point has its own basis function, which determines how strongly that control point influences the curve at time *t*. In fact the B in B-splines stands for Basis.

Knot Vector

Knot vectors are used to ensure that basis functions span over variable time intervals. By varying the relative length of interval we can vary the amount of time each control point affects the particle on the curve. The points demarcating the intervals are known as *knots*, and the ordered list of them is a knot vector (Figure 9-4).

Weight Vector

The IGES files include a vector deterring the weight of the control points. Ordinarily, each control point carries a weight of 1.0, meaning that they all have equal influence on the shape of the curve. Increasing the weight of an individual control point gives it more influence and has the effect of “pulling” the curve toward that point.

Curves that are defined in this way, with the weight *w* for each control point, are called rational curves.

Mx4 Octavia and NURBS

The following information is required to program Mx4 Octavia with NURBS curves:

- Control Vector
- Weight Vector
- Knot Vector
- Desired FeedRate
- NURBSRate
- First Knot Point
- Last Knot Point
- Number of Control Points in the control vector

***Note*:** All of these values are represented as C formatted 32-bit floating point numbers.

- The Control Vector contains the 3-D control points for the NURBS curve. 3-D means elements for x,y, and z.
- The Weight Vector contains the scalar weight values associated with each control point
- The Knot Vector contains the Knots for the NURBS curve
- The desired feedrate for the motion along the NURBS curve, in counts/200µsec. This value may also be changed on the fly via the NURBS FeedRate RTC.
- The NURBSRate is the amount of time between the points extracted from the NURBS curve, between these points Octavia will interpolate every 200µsec.
- The First and Last Knot points are simply the end points of the knot vector
- The total number of Control points for the entire NURBS curve. Note the points are three-dimensional, i.e. one control point contains an X, Y, and Z element.

Octavia via three ring buffers, while all other parameters are sent in the NURBS RTC. The three ring buffers are located in the Mx4 Octavia's DPRAM, they are located as follows:

Ring Buffer	DPRAM Address Range
Control Vector	120h to 29Fh
Weight Vector	2A0h to 31Fh
Knot Vector	320h to 3AFh

The Control Vector Ring buffer holds up to 32 3-D floating point Control Points, the Weight Vector will also hold 32 scalar floating points, but the Knot Vector can contain 36 scalar floating point Knot values.

All three ring buffers can be managed with an INPTR and an OUTPTR, these two pointers actually point to the Knot Vector Ring buffer. The INPTR is a 16-bit integer and is located in the DPRAM at address 3B0h, and the OUTPTR is also a 16-bit integer and is located at 3B2h. The INPTR points to the next location in the Knot ring buffer to be filled by the host and the OUTPTR points to the next location in the Knot ring buffer the Mx4 Octavia will read from. The INPTR must always be at least 8 locations away from the OUTPTR, this includes the wrap-around to obey the ring buffer scheme.

The process involved in commanding the Mx4 Octavia to start motion along some NURBS curve can be broken into the following simple steps:

1. Initially fill the ring buffers with the Control, Weight, and Knot Vector points.

2. Issue the NURBS RTC with the FeedRate, NURBSRate, First Knot Point, Last Knot Point, and the total number of Control Points for the NURBS curve as arguments.

3. Keep updating the ring buffers with the Control, Weight, and Knot Vector points as needed, until all of the Vector points have been transmitted to the Mx4 Octavia.

A more detailed explanation of the steps involved with NURBS based contouring is as follows:

Initially all three ring buffers should be empty. To force this condition, set the INPTR and OUTPTR to 0. (see figure 9-4.) Notice that the INPTR and OUTPTR only point to the Knot Points Ring Buffer.

At this point, if the NURBS buffer breakpoint interrupt will be used for the NURBS Contouring, then the RTC must be issued. The argument to the NURBS Buffer Breakpoint Interrupt is the threshold between the INPTR and OUTPTR, the number of Knot points separating the INPTR from the OUTPTR. While the Mx4 Octavia controller is using the Knot Vector Points, it will increment the OUTPTR. When the OUTPTR becomes within the range of the desired threshold from the INPTR, an interrupt will be triggered alerting the host that more Knot, Control, and Weight points are needed in the ring buffers. This is explained more below. Note, instead of using the NURBS Buffer Breakpoint Interrupt, the host may also monitor the INPTR and OUTPTR, then when there is free space in the ring buffers, more Knot, Control, and Weight points may be placed into the ring buffers.

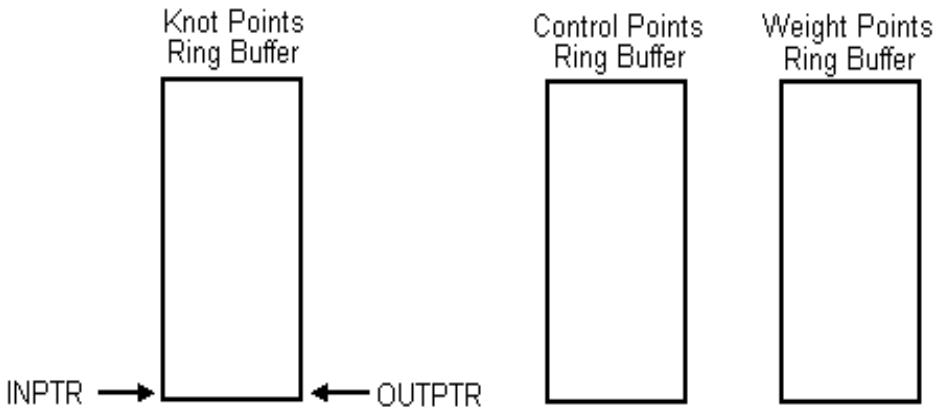


Figure 9-4: Initially, the three ring buffers are empty.

Now the host computer may start filling the three ring buffers with the Knot, Control, and Weight points. (see figure 9-5.) As each Knot point is placed into the Knot buffer the INPTR should be incremented. This informs the Mx4 Octavia how many points in the Knot buffer are valid points. Again, the location in the DPRAM for the beginning of the Knot Points Ring Buffer is 320h, Control Points Ring Buffer is 120h, and the Weight points Ring Buffer is 2A0h. The Control Points Vector contains 3-D information, therefore each Control Point should be placed into the Control Point Ring Buffer with the X element written first, then the Y and Z elements should follow. For example, let the first Control point be represented by (X1, Y1, Z1) and the second Control point be represented by (X2, Y2, Z2), and so on, then the points should be written to the Control Points Ring Buffer in the following order:

X1, Y1, Z1, X2, Y2, Z2, X3, Y3, Z3, …

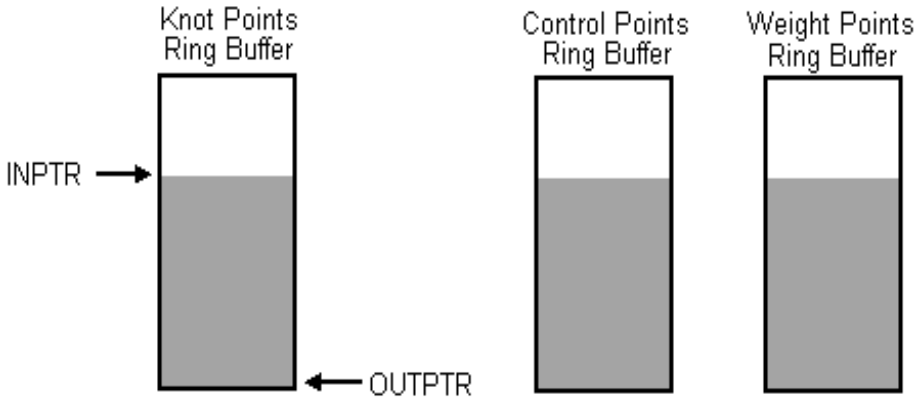


Figure 9-5: Start filling the three ring buffers.

Once the three buffers have been filled, as in figure 6, the RTC to start the NURBS Contouring may be issued to the Mx4 Octavia controller. The arguments to the RTC are FeedRate, NURBSRate, First Knot Vector Point, Last Knot Vector Point, and total number of Control Points for the entire NURBS curve.

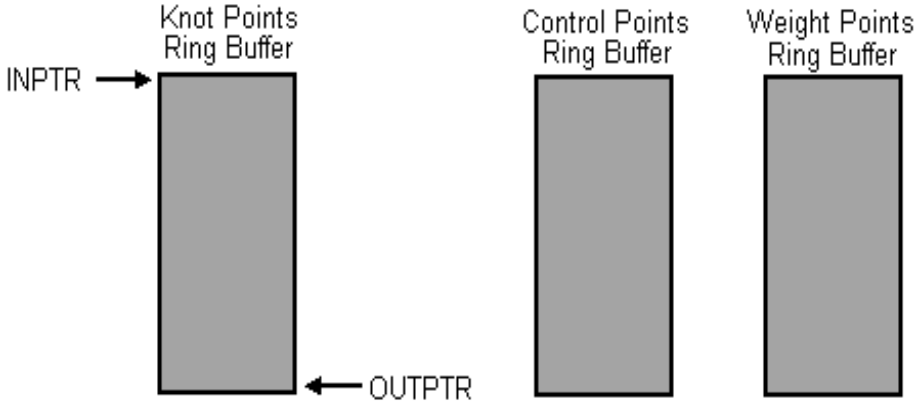


Figure 9-6: The three ring buffers have been filled.

At this point, the Mx4 Octavia has begun the motion following the NURBS curve. While the Knot Points are used for the NURBS calculations within the Mx4 Octavia, the OUTPTR is incremented. Recall that the three buffers are implemented as ring buffers, so when the OUTPTR is pointing at the last element in the buffer and it is incremented, the pointer wraps around and points to the first element in the ring buffer.

If the NURBS Buffer Breakpoint Interrupt RTC was issued, then as soon as the OUTPTR comes within the desired threshold number of Knot points within the INPTR, an interrupt to the host will be issued. Upon acknowledgment of this interrupt, the host can check the INPTR and OUTPTR to determine how many Knot, Control, and Weight points may be sent to the buffers and then send those points. (see figure 9-4.) Remember, while sending the points to the ring buffers, when the last location of any ring buffer is filled, the next location to fill would be the first location in the ring buffer.

If the NURBS Buffer Breakpoint Interrupt is not used, then the host must continually monitor the OUTPTR and send more points to the ring buffers as space opens in the Knot Points Ring Buffer. (see figure 9-7.)

The host must monitor the INPTR and OUTPTR or acknowledge the NURBS Buffer Breakpoint Interrupt until all of the Control, Weight, and Knot points have been sent to the Mx4 Octavia ring buffers in the DPRAM.

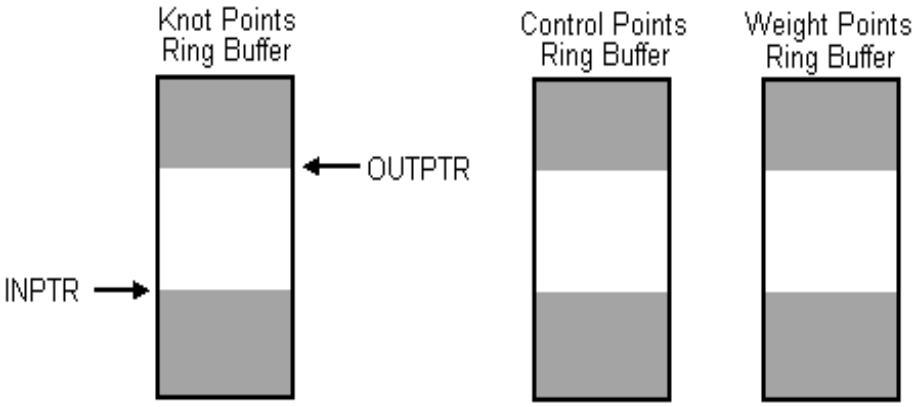


Figure 9-7: NURBS Contouring has started, space between the INPTR andOUTPTR may be filled with Control, Weight, and Knot Points

NURBS Programming Example

Let's say you have a NURBS curve represented by a Control Vector and Weight Vector both containing 41 points and a Knot Vector containing 45 points. Assume the Knot Vector contains the following endpoints,

[0.00 0.00 0.00 0.00 1.12 . . . 100.02 103.45 103.45 103.45 103.45]

To start the NURBS motion, the ring buffers must first be filled with the Knot, Control, and Weight points. (see figure 9-8.) Before filling the buffers the INPTR and OUTPTR must be set to 0. Now the ring buffers can be filled. Notice that 35 Knot points may be placed in the Knot ring buffer, 32 Control points can be in the Control ring buffer, and 32 Weight points may be in the Weight ring buffer. Only 35 Knot values may reside in the Knot vector at any one time since the INPTR should never equal the OUTPTR, this is just a phenomena of implementing a shared memory ring buffer with two pointers.

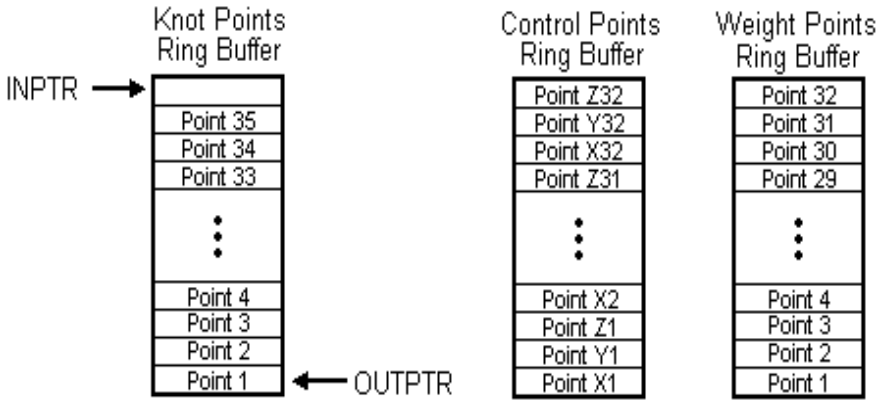


Figure 9-8: The three ring buffers have been filled.

Now, before the NURBS motion is started, the NURBS Buffer Breakpoint Interrupt could be issued. In this example, instead of constantly monitoring the INPTR and OUTPTR, we will use the interrupt with a threshold value of 20 points. That is, when the OUTPTR comes within 20 points of the INPTR, the NURBS Buffer Breakpoint Interrupt will be issued. Upon acknowledgment of the interrupt, we can send more Knot, Control, and Weight points. Remember, at least 8 points must always exist between the INPTR and OUTPTR, therefore the chosen threshold should probably be 16 or more.

Now the NURBS RTC should be issued, the arguments will be as follows,

FeedRate = 1	We would like to traverse the NURBS curve at a feedrate of one count/200µsec.
NURBSRate = 3	Time between each calculated point on the NURBS curve, within the Mx4 Octavia controller, is 3ms. This time scale will result in maximum resolution of the NURBS curve.
First Knot Point = 0.00	First element in the Knot Vector.
Last Knot Point = 103.45	Last element in the Knot Vector.
Total Control Points = 41	Total number of Control Points in the Control Vector.

After the NURBS RTC is issued, the NURBS Contouring will have started. At this point the host computer will not be involved with the NURBS until the NURBS Buffer Breakpoint Interrupt has been issued from the Mx4 Octavia. When the host acknowledges the interrupt, the INPTR and OUTPTR must be read from Mx4 Octavia and the number of possible Knot, Control, and Weight points to send can be calculated from these two pointers. In the case where the NURBS Buffer Breakpoint Interrupt was set up with a threshold of 20 points, the host may be able to send 16 or more Knot points, and 12 or more Control and Weight points, depending on the INPTR and OUTPTR. In our example, the host only has 10 Knot Points, 9 Control Points, and 9 Weight Points remaining to send to the Mx4 Octavia. Let's say that after some latency for the host program to respond to the interrupt, the INPTR and OUTPTR are read from the Mx4 Octavia, the INPTR equals 36 and the OUTPTR equals 17. In this case, 17 Knot, Control, and Weight Points may be placed into their appropriate ring buffers. Therefore, all of the Knot, Control, and Weight Points may be sent to the open locations in the ring buffers. (see figure 9-9.) If the host had more Knot, Control, and Weight points to send to the Mx4 Octavia, then the ring buffers could be filled each time a NURBS Buffer Breakpoint Interrupt was received from the Mx4 Octavia, until all of the points have been transmitted.

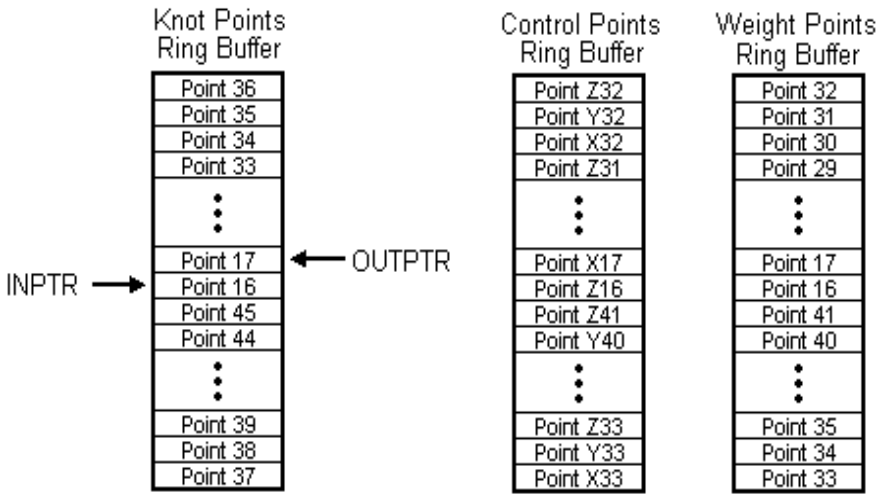


Figure 9-9: NURBS Contouring has started, space between the INPTR and OUTPTR may be filled with Control, Weight, and Knot Points

10 Mx4 Octavia Specifications

Performance

ITEM	DESCRIPTION
Control Algorithms	State Feedback Multi-Input Multi-Output Controller, Kalman Filter, Robust Control, PID, Notch
Position Range	+/- 2, 147, 483, 650 counts, rollover, +/- 1 count
Position Capture	100 ns max. delay from trigger
Velocity Range	0 to 1,280,000 counts/sec, +/- 1 count
Acceleration Range	0 to 50,000,000 counts/sec^2, +/- 1 count
Synchronization	Unlimited number of axes can be synchronized to the same servo cycle

Hardware

ITEM	DESCRIPTION
Processing	Quad DSPs in Hyper-Cube Architecture
Analog Output	16-bit parallel DAC per axis, 300 uV resolution, -10v to +10v output

Input/Output

ITEM	DESCRIPTION
Protective Inputs	*ESTOP emergency stop [1]
External Interrupts	*EXT1-4 [4]
General Purpose Inputs	User-defined inputs [32], TTL logic
General Purpose Outputs	User-defined outputs [32], TTL logic

Position Encoder Feedback

ITEM	DESCRIPTION
Encoder Type	A/B quadrature or single-ended with "I" (index pulse) channel [optional]
Maximum Encoder Count	10 MHz

Programming

ITEM	DESCRIPTION
High Level Language	DSPL high-level motion programming language supporting both on-line motion and background PLC programs.
Battery Backup (option)	non-volatile memory with 10 year battery life permits stand-alone operation.

Electrical

ITEM	DESCRIPTION
+5v Mx4 Octavia Supply Voltage	MIN = 4.75v NOM = 5v MAX = 5.25v
Operating Free Air Temperature Range	0°C to 70°C

Power Consumption

ITEM	DESCRIPTION
+5v	MAX = 3A
+12v	MAX = 250mA
-12v	MAX = 250mA

Mechanical

ITEM	DESCRIPTION
PCI Mx4 Octavia Interface Connectors	protected header, .100 x .100 centers, center and dual polarized
PCI Mx4 Octavia I/O Connectors	protected header, low profile, .100 x .100 centers, center polarized
PCI Mx4 Octavia Synch Connector	AMP 640457-4 friction lock header
Mechanical Dimensions	See Fig. 10-1

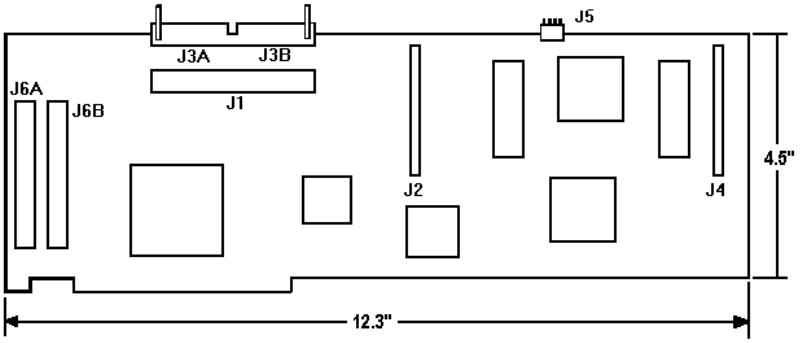


Fig. 10-1: PCI Mx4 Octavia Mechanical Dimensions

This page intentionally blank.