PCI-8164 / MPC-8164
Advanced 4-Axis Servo / Stepper
Motion Control Card
User's Guide



Recycle Paper

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

© Copyright 2003 ADLINK Technology Inc

All Rights Reserved.

Manual Rev. 1.30: 4 June, 2003

Part No: 50-11124-103

The information in this document is subject to change without prior notice in order to improve reliability, design, and function and does not represent a commitment on the part of the manufacturer.

In no event will the manufacturer be liable for direct, indirect, special, incidental, or consequential damages arising out of the use or inability to use the product or documentation, even if advised of the possibility of such damages.

This document contains proprietary information protected by copyright laws. All rights are reserved. No part of this manual may be reproduced by any mechanical, electronic, or other means in any form without prior written permission of the manufacturer.

Trademarks

NuDAQ and PCI-8164/MPC-8164 are registered trademarks of ADLINK Technology Inc, MS-DOS & Windows 95/NT/2000/XP are registered trademarks of Microsoft Corporation and Borland C++ is a registered trademark of Borland International, Inc. Other product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Getting service from ADLINK

Customer Satisfaction is the most important priority for ADLINK Tech Inc. If you need any help or service, please contact us.

ADLINK Technology Inc.			
Web Site	http://www.adlinktech.com		
Sales & Service	Service@adlinktech.com		
Technical Support	NuDAQ + USBDAQ	nudaq@adlinktech.com	
	Automation	automation@adlinktech.com	
	NuIPC	nuipc@adlinktech.com	
	NuPRO / EBC	nupro@adlinktech.com	
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan.		

Please email or FAX us your detailed information for prompt, satisfactory, and consistent service.

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			
Questions			
Product Model			
Environment	OS: Computer Brand: M/B: CPU: Chipset: BIOS: Video Card: NIC: Other:		
Detail Description			
Suggestions for ADLINK			

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Table of Contents

Chapter 1 Introduction 1
1.1 Features 5
1.2 Specifications 7
1.3 Supported Software 8
1.3.1 Programming Library 8
1.3.2 Motion Creator 8

Chapter 2 Installation 9
2.1 Package Contents 9
2.2 PCI-8164 Outline Drawing 10
2.2A MPC-8164 Outline Drawing 11
2.3 PCI-8164 Hardware Installation 12
2.3.1 Hardware configuration 12
2.3.2 PCI slot selection 12
2.3.3 Installation Procedures 12
2.3.4 Troubleshooting: 12
2.3A MPC-8164 Hardware Installation 13
2.3A.1 Hardware configuration 13
2.3A.2 Troubleshooting: 14
2.4 Software Driver Installation 14
2.5 CN1 Pin Assignments: External Power Input (PCI-8164 Only) 15
2.6 CN3 Pin Assignments: Manual Pulse Input (PCI-8164 Only) 16
2.6A CN3 Pin Assignments: General Purpose DIO (MPC-8164 Only) 16
2.7 CN2 Pin Assignments: Main connector 17
2.8 CN4 Pin Assignments: Simultaneous Start/Stop (PCI-8164 Only) 18
2.9 CN5 Pin Assignment: TTL Output (PCI-8164 Only) 18
2.10 Jumper Setting for Pulse Output (PCI-8164 Only) 19
2.11 Switch Setting for EL Logic 19
2.12 CN3 Pin Assignment: General Purpose DI/DO ports (MPC-8164 Only) 20

Chapter 3 Signal Connections 21
3.1 Pulse Output Signals OUT and DIR 22
3.2 Encoder Feedback Signals EA, EB and EZ 24
3.3 Origin Signal ORG 26
3.4 End-Limit Signals PEL and MEL 27
3.5 Ramping-down & PCS 28
3.6 In-position Signal INP 29
3.7 Alarm Signal ALM 30
3.8 Deviation Counter Clear Signal ERC 31
3.9 General-purpose Signal SVON 32
3.10 General-purpose Signal RDY 32
3.11 Position compare output pin: CMP 33

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.12 Position latch input pin: LTC 34
3.13 Pulse Input Signals PA and PB (PCI-8164) 35
3.14 Simultaneously Start/Stop Signals STA and STP(PCI-8164 Only) . 35
3.15 General Purpose TTL Output (PCI-8164 only)..... 37
3.16 Termination Board 37
3.17 General Purpose DIO (MPC-8164 only) 38
3.17.1 Isolated Input channels 38
3.17.2 Isolated Output channels 38
3.17.3 Example of input connection 39
3.17.4 Example of output connection..... 40

Chapter 4 Operation Theory 41
4.1 Motion Control Modes..... 41
4.1.1 Pulse Command Output 42
4.1.2 Velocity mode motion 44
4.1.3 Trapezoidal Motion..... 45
4.1.4 S-curve Profile Motion 47
4.1.5 Linear interpolation for 2-4 axes 49
4.1.6 Circular interpolation for 2 axes..... 53
4.1.7 Circular interpolation with Acc/Dec time 54
4.1.8 Relationship between Velocity and Acceleration Time. 55
4.1.9 Continuous motion 57
4.1.10 Home Return Mode 62
4.1.11 Manual Pulse Mode (PCI-8164 Only)..... 70
4.2 The motor driver interface..... 70
4.2.1 INP 71
4.2.2 ALM..... 72
4.2.3 ERC..... 73
4.2.4 SVON and RDY..... 73
4.3 The limit switch interface and I/O status 74
4.3.1 SD/PCS..... 74
4.3.2 EL..... 75
4.3.3 ORG 76
4.4 The Counters 76
4.4.1 Command position counter 76
4.4.2 Feedback position counter 77
4.4.3 Position error counter..... 79
4.4.4 General purpose counter..... 79
4.4.5 Target position recorder 80
4.5 Multiple PCI-8164 Card Operation (PCI-8164 Only) 81
4.6 Change position or speed on the fly 82
4.6.1 Change speed on the fly 82
4.6.2 Change position on the fly..... 86
4.7 Position compare and Latch 88
4.7.1 Comparators of the 8164..... 88
4.7.2 Position compare..... 89
4.7.3 Position Latch..... 92

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.8 Hardware backlash compensator and vibration suppression 92
4.9 Software Limit Function 93
4.10 Interrupt Control 94

Chapter 5 Motion Creator 99
5.1 Execute Motion Creator 100
5.2 About Motion Creator 100
5.3 Motion Creator Form Introducing 101
5.3.1 Main Menu 101
5.3.2 Interface I/O Configuration Menu 101
5.3.3 Pulse IO & Interrupt Configuration Menu 103
5.3.4 Operation menu: 104

Chapter 6 Function Library 110
6.1 List of Functions 110
6.2 C/C++ Programming Library 117
6.3 Initialization 118
6.4 Pulse Input/Output Configuration 121
6.5 Velocity mode motion 122
6.6 Single Axis Position Mode 125
6.7 Linear Interpolated Motion 129
6.8 Circular Interpolation Motion 134
6.9 Home Return Mode 140
6.10 Manual Pulse Motion 141
6.11 Motion Status 144
6.12 Motion Interface I/O 145
6.13 Motion I/O Monitoring 147
6.14 Interrupt Control 148
6.15 Position Control and Counters 153
6.16 Position Compare and Latch 156
6.17 Continuous motion 161
6.18 Multiple Axes Simultaneous Operation 162
6.19 General-purposed TTL output (PCI-8164 Only) 165
6.20 General-purposed DIO (MPC-8164 Only) 166

Chapter 7 Connection Example 167
7.1 General Description of Wiring 167
7.2 Connection Example with Servo Driver 168
7.3 Wiring with DIN-814M 171
7.4 Wiring with DIN-814P 175

Appendix A Color code of CN3 Cable (MPC-8164 Only) 179

Warranty Policy 180

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

How to Use This Guide

This manual is designed to help you use the PCI-8164/MPC-8164 and describes how to modify various settings to meet your requirements. It is divided into seven chapters:

Chapter 1 Introduction

An overview of the product features, applications, and specifications.

Chapter 2 Installation

Describes how to install the PCI-8164/MPC-8164.

Chapter 3 Signal Connections

Details the connector pin assignments and how to connect external signals and devices to the PCI-8164/MPC-8164.

Chapter 4 Operation Theory

Describes the operations of the PCI-8164/MPC-8164.

Chapter 5 Motion Creator

Details how to use the Windows based utility program to configure and run tests with the PCI-8164/MPC-8164.

Chapter 6 C/C++ Function Library

Describes high-level programming in C/C++ to aid in programming the PCI-8164/MPC-8164.

Chapter 7 Connection Example

Illustrates typical connection examples between the PCI-8164/MPC8164 and servo/stepping drivers.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1

Introduction

The PCI-8164 is an advanced 4-axis motion controller card with a PCI interface. It can generate high frequency pulses (6.4MHz) to drive stepper or servomotors. As a motion controller, it can provide 2-axis circular interpolation, 4-axis linear interpolation, or continuous interpolation for continual velocity. Also, changing position/speed on the fly is available with a single axis operation.

Multiple PCI-8164 cards can be used in one system. Incremental encoder interface on all four axes provide the ability to correct positioning errors generated by inaccurate mechanical transmissions. With the aid of on board FIFO, the PCI-8164 can also perform precise and extremely fast position comparison and trigger functions without compromising CPU resources. In addition, a mechanical sensor interface, servo motor interface, and general-purposed I/O signals are provided for easy system integration.

Figure 1 shows the functional block diagram of the PCI-8164 card. The PCI-8164 uses one ASIC (PCL6045) to perform all 4 axes motion controls. The motion control functions include linear and S-curve acceleration/deceleration, circular interpolation between two axes, linear interpolation between 2~4 axes, continuous motion positioning, and 13 home return modes. All these functions and complex computations are performed internally by the ASIC, thus limiting the impact on the PC's CPU usage.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

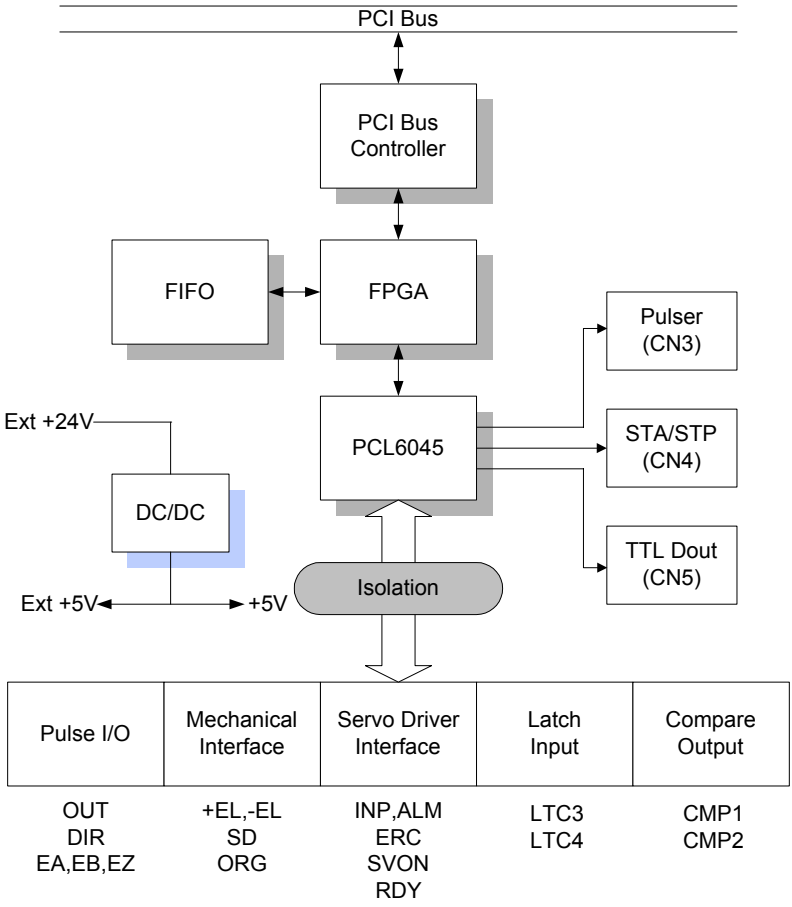


Figure 1: Block Diagram of the PCI-8164

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The MPC-8164 is an advanced 4-axis motion controller card with a PC104 interface. All features and specification are the same as the PCI-8164, except for slight differences in the user I/O interfaces. Refer to the previous introduction for more details. Figure 2 is the block diagram of the MPC-8164 card.

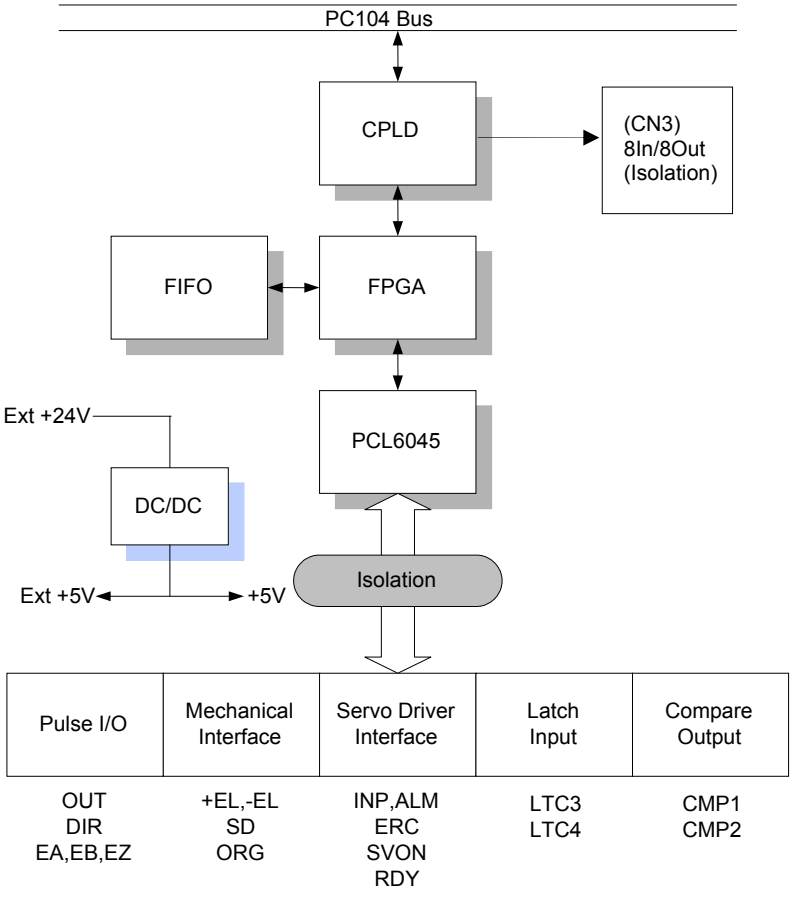


Figure 2: Block Diagram of the MPC-8164

Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Motion Creator is a Windows-based application development software package included with the PCI-8164/MPC-8164. *Motion Creator* is useful for debugging a motion control system during the design phase of a project. An on-screen display lists all installed axes information and I/O signal status of the PCI-8164/MPC-8164.

DOS and Windows programming libraries are also included for C++ and Visual Basic. Sample programs are provided to illustrate the operations of the functions.

Figure 3 illustrates a flow chart of the recommended process in using this manual in developing an application. Refer to the related chapters for details of each step.

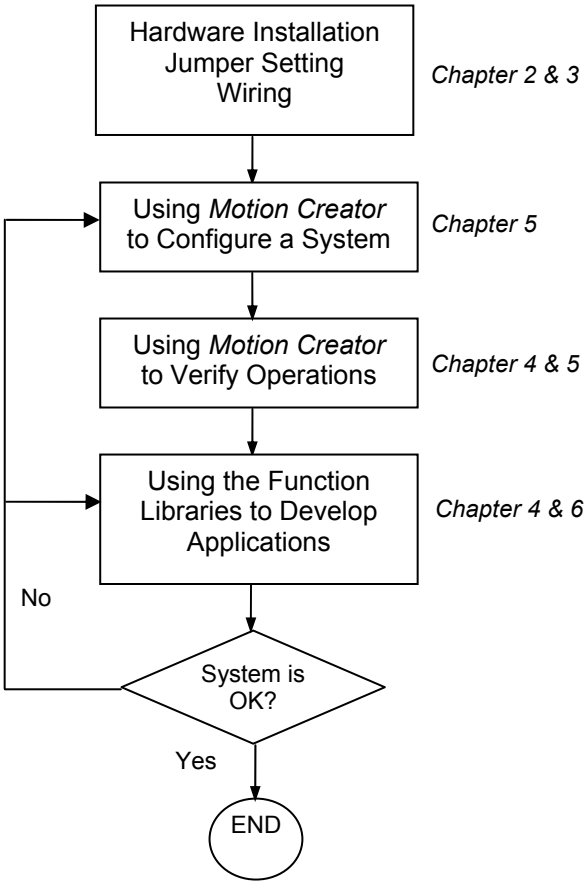


Figure 3: Flow chart for building an application

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1.1 Features

The following list summarizes the main features of the PCI-8164 motion control system.

- 32-bit PCI bus Plug and Play
- 4 axes of step and direction pulse output for controlling stepping or servomotor
- Maximum output frequency of 6.55 MPPS
- Pulse output options: OUT/DIR, CW/CCW
- Programmable acceleration and deceleration time for all modes
- Trapezoidal and S-curve velocity profiles for all modes
- Any 2 of 4 axes circular interpolation
- Any 2-4 of 4 axes linear interpolation
- Continuous interpolation for contour following motion
- Change position and speed on the fly
- Change speed by condition comparing
- 13 home return modes with auto searching
- Hardware backlash compensator and vibration suppression
- 2 software end-limits for each axis
- 28-bit up/down counter for incremental encoder feedback
- Home switch, index signal (EZ), positive, and negative end limit switches interface on all axes
- 2-axis high speed position latch input
- 2-axis position compare trigger output with 4k FIFO auto-loading
- All digital input and output signals are 2500Vrms isolated
- Programmable interrupt sources
- Simultaneous start/stop motion on multiple axes
- Manual pulser input interface(Manual Pulser is a device like a small steering wheel which generate pulses when turning it)
- Software supports a maximum of up to 12 PCI-8164 cards (48 axes) operation in one system
- Compact, half size PCB
- Includes *Motion Creator*, a Microsoft Windows-based application development software
- PCI-8164 libraries and utilities for DOS and Windows 9x/NT/2000/XP. Also supported under Linux

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The following list summarizes the main features of the MPC-8164 motion control system.

- 16-bit PC104 Bus
- 4 axes of step and direction pulse output for controlling stepping or servomotor
- Maximum output frequency of 6.55 MPPS
- Pulse output options: OUT/DIR, CW/CCW
- Programmable acceleration and deceleration time for all modes
- Trapezoidal and S-curve velocity profiles for all modes
- Any 2 of 4 axes circular interpolation
- Any 2-4 of 4 axes linear interpolation
- Continuous interpolation for contour following motion
- Change position and speed on the fly
- Change speed by comparator condition
- 13 home return modes with auto searching
- Hardware backlash compensator and vibration suppression
- 2 Software end-limits for each axis
- 28-bit up/down counter for incremental encoder feedback
- Home switch, index signal(EZ), positive, and negative end limit switches interface on all axes
- 2-axis high speed position latch input
- 2-axis position compare trigger output with 4k FIFO auto-loading
- All digital input and output signals are 2500Vrms isolated
- Programmable interrupt sources
- 8 channels of general purpose photo-isolated digital inputs
- 8 channels of general purpose open collector digital outputs
- Software supports a maximum of up to 4 MPC-8164 cards (16 axes) operation in one system
- Includes *Motion Creator*, Microsoft Windows-based application development software
- MPC-8164 Libraries and Utilities for DOS and Windows 98/NT/2000/XP. Also support Windows XP/NT Embedded
- MPC-8164 Libraries for Linux and Windows CE systems

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1.2 Specifications

- **Applicable Motors:**
  - Stepping motors
  - AC or DC servomotors with pulse train input servo drivers
- **Performance:**
  - Number of controllable axes: 4
  - Maximum pulse output frequency: 6.55MPPS, linear, trapezoidal, or S-Curve velocity profile drive
  - Internal reference clock: 19.66 MHz
  - 28-bit up/down counter range: 0-268,435,455 or –134,217,728 to +134,217,727
  - Position pulse setting range (28-bit): -134,217,728 to +134,217,728
  - Pulse rate setting range (Pulse Ratio = 1: 65535):
    - ✓ 1 PPS to 6553.5 PPS. (Multiplier = 0.1)
    - ✓ 1 PPS to 65535 PPS. (Multiplier = 1)
    - ✓ 100 PPS to 6553500 PPS. (Multiplier = 100)
- **I/O Signales:**
  - Input/Output signals for each axis
  - All I/O signal are optically isolated with 2500Vrms isolation voltage
  - Command pulse output pins: OUT and DIR
  - Incremental encoder signals input pins: EA and EB
  - Encoder index signal input pin: EZ
  - Mechanical limit/switch signal input pins: ±EL, SD/PCS, and ORG
  - Servomotor interface I/O pins: INP, ALM, and ERC
  - Position latch input pin: LTC
  - Position compare output pin: CMP
  - General-purposed digital output pin: SVON
  - General-purposed digital input pin: RDY
  - Pulse signal input pin: PA and PB (PCI-8164 only)
  - Simultaneous Start/Stop signal: STA and STP (PCI-8164 only)

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **General-Purposed Output**
  - 6 TTL level digital outputs (PCI-8164 only)
  - 8 digital inputs / 8 digital outputs (MPC-8164 only)
- **General Specifications**
  - Connectors: 100-pin SCSI-type connector
  - Operating Temperature: 0°C - 50°C
  - Storage Temperature: -20°C - 80°C
  - Humidity: 5 - 85%, non-condensing
- **Power Consumption**
  - Slot power supply (input): +5V DC ±5%, 900mA max
  - External power supply (input): +24V DC ±5%, 500mA max
  - External power supply (output): +5V DC ±5%, 500mA, max
- **PCI-8164 Dimension: 185mm(L) X 98.4mm(W)**
- **MPC-8164 Dimension: 152mm(L) X 104.7mm(W)**

1.3 Supported Software

1.3.1 Programming Library

MS-DOS Borland C/C++ (Version: 3.1) programming libraries and Windows 95/98/NT/2000/XP DLLs are provided for the PCI-8164. These function libraries are shipped with the board. Support for Linux is also included.

MPC-8164 supports DOS/Windows 98/NT/2000/XP, Windows XP/NT Embedded, Linux, and Windows CE.

1.3.2 Motion Creator

This Windows-based utility is used to setup cards, motors, and systems. It can also aid in debugging hardware and software problems. It allows users to set I/O logic parameters to be loaded in their own program. This product is also bundled with the card.

Refer to Chapter 5 for more details.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2

Installation

This chapter describes how to install the PCI-8164/MPC-8164. Please follow these steps below:

- Check what you have (section 2.1)
- Check the PCB (section 2.2)
- Install the hardware (section 2.3)
- Install the software driver (section 2.4)
- Understanding the I/O signal connections (chapter 3) and their operation (chapter 4)
- Understanding the connector pin assignments (the remaining sections) and wiring the connections

2.1 Package Contents

In addition to this *User's Guide*, the package also includes the following items:

- PCI-8164/MPC-8164: advanced 4-Axis Servo / Stepper Motion Control Card
- ADLINK All-in-one Compact Disc
- +24V power input cable (for CN1) accessory (PCI-8164 only)
- An optional terminal board for wiring purposes if a different model is ordered.

If any of these items are missing or damaged, contact the dealer from whom you purchased the product. Save the shipping materials and carton to ship or store the product in the future.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.2 PCI-8164 Outline Drawing

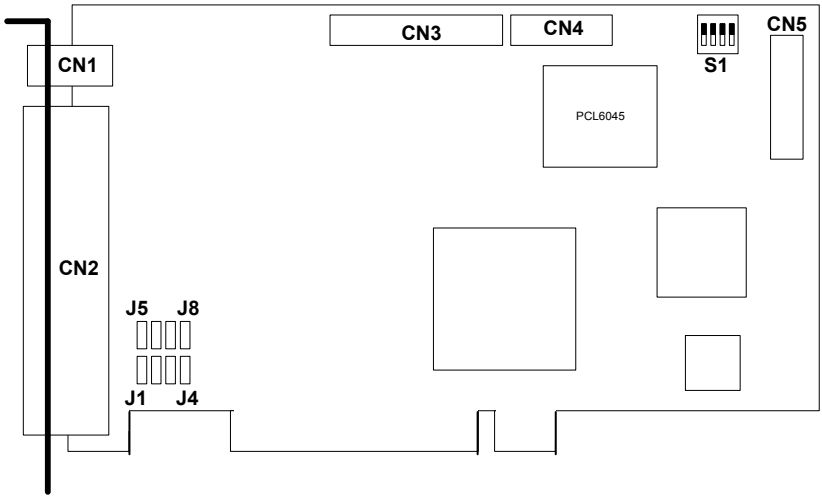


Figure 4: PCB Layout of the PCI-8164

CN1: External Power Input Connector
CN2: Input / Output Signal Connector
CN3: Manual Pulse Signal Connector
CN4: Simultaneous Start / Stop Connector
CN5: General purpose TTL output
S1: End limit logic selection switch
J1~J8: Pulse output selection jumper

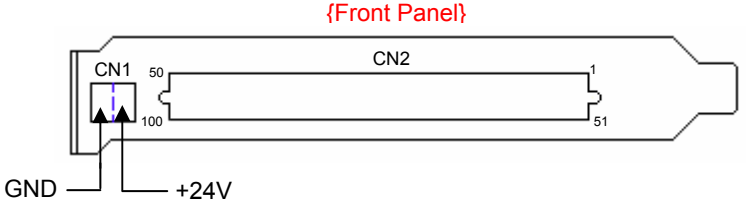


Figure 5: PCI-8164 Face Plate

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.2A MPC-8164 Outline Drawing

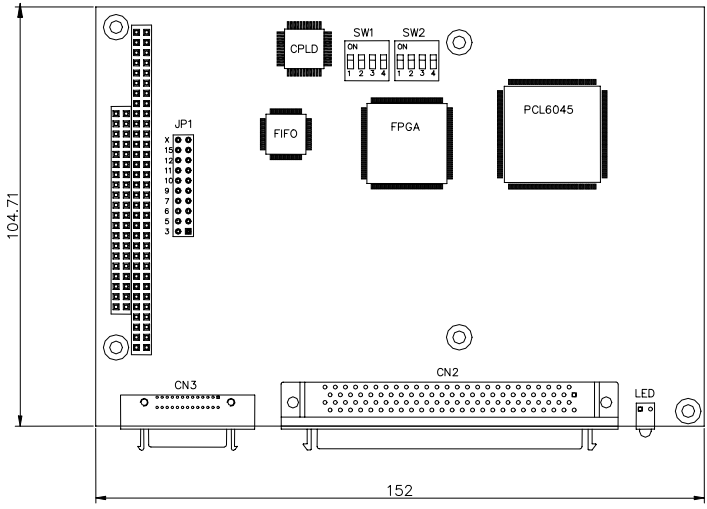


Figure 6: PCB Layout of the MPC-8164

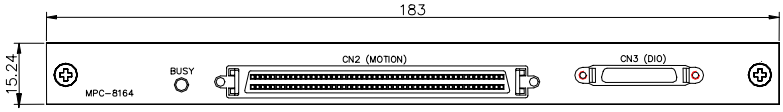


Figure 7: MPC-8164 Face Plate

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.3 PCI-8164 Hardware Installation

2.3.1 Hardware configuration

The PCI-8164 is fully Plug and Play compliant. Hence memory allocation (I/O port locations) of the PCI card are assigned by the system BIOS. The address assignment is done on a board-by-board basis for all PCI cards in the system.

2.3.2 PCI slot selection

Your computer system may have both PCI and ISA slots. Do not force the PCI card into a PC/AT slot. The PCI-8164 can be used in any PCI slot.

2.3.3 Installation Procedures

1. Read through this manual and setup the jumper according to your application
2. Turn off your computer. Turn off all accessories (printer, modem, monitor, etc.) connected to computer. Remove the cover from your computer.
3. Select a 32-bit PCI expansion slot. PCI slots are shorter than ISA or EISA slots and are usually white or ivory.
4. Before handling the PCI-8164, discharge any static buildup on your body by touching the metal case of the computer. Hold the edge of the card and do not touch the components.
5. Position the board into the PCI slot you have selected.
6. Secure the card in place at the rear panel of the system unit using screws removed from the slot.

2.3.4 Troubleshooting:

If your system doesn't boot or if you experience erratic operation with your PCI board in place, it's most likely caused by an interrupt conflict (possibly an incorrect ISA setup). In general, the solution, once determined it is not a simple oversight, is to consult the BIOS documentation that comes with your system.

Check the control panel of the Windows system if the card is listed by the system. If not, check the PCI settings in the BIOS or use another PCI slot.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

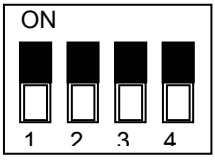
2.3A MPC-8164 Hardware Installation

2.3A.1 Hardware configuration

The MPC-8164 is PC104 compliant. I/O port locations and IRQ channels for the card are assigned by onboard DIP switches and jumpers. Refer to the following settings:

Base address setting:

The base address is set by pin 2 to 4 of SW2. Note that pin1 is reserved. If all dips are set to the "OFF" position, the base address would be 0x200. Default settings are dependant on the order.

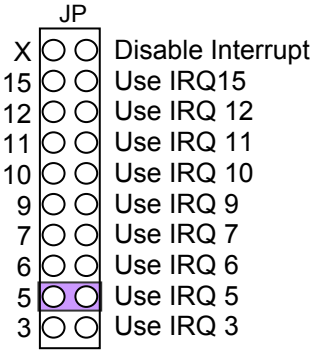


DIP Switch (2 3 4)	Base Address
1 1 1	0x3C0
0 1 1	0x380
1 0 1	0x340
0 0 1	0x300

DIP Switch (2 3 4)	Base Address
1 1 0	0x2C0
0 1 0	0x280
1 0 0	0x240
0 0 0	0x200

IRQ setting:

The IRQ channel is assigned by setting JP1



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.3A.2 Troubleshooting:

MPC-8164: Make sure that the system's I/O address and IRQ channel are available for the card. If not, change the setting to an empty I/O address and IRQ channel.

2.4 Software Driver Installation

PCI-8164:

Step 1: Auto run the ADLINK All-In-One CD. Choose Driver Installation -> Motion Control -> PCI-8164.

Step 2: Follow the procedures of the installer.

Step 3: After setup installation is completed, restart windows.

Note: If using MS-DOS, locate the directory \Motion Control\PCI-8164\DOS_BC in the CD-ROM.

MPC-8164

Step1: Insert the ADLINK All-In-One CD and let Windows auto-run the setup program. Choose Driver Installation -> Motion Control -> MPC-8164

Step2: Run the "MPC-8164 Add/Remove" utililty from the start menu or installed directory to register the new card. The I/O address and IRQ channel must be the same as the settings on the board.

Step3: Restart computer

Note: If using MS-DOS, use the software in the directory `\Motion Control\MPC-8164\DOS_BC` on the CD-ROM.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.5 CN1 Pin Assignments: External Power Input (PCI-8164 Only)

CN1 Pin No	Name	Description
1	EXGND	External power ground
2	EX+24V	+24V DC ± 5% External power supply

Notes:

1. CN1 is a plug-in terminal board with no screws.
2. Be sure to use the external power supply. A +24V DC is used by external input/output signal circuits. The power circuit is configured as shown below.
3. Wires for connection to CN1

   Solid wire: φ 0.32mm to φ 0.65mm (AWG28 to AWG22)

   Twisted wire: 0.08mm^2 to 0.32mm^2 (AWG28 to AWG22)

   Naked wire length: 10mm standard

The following diagram shows the external power supply system of the PCI-8164. An external +24V power must be provided. An on-board regulator generates +5V for both internal and external usage.

Note: DO NOT use the +5V power source to drive too many devices simultaneously, especially stepper motors or external encoders. The output current capacity is limited.

Note: MPC-8164 does **NOT** have a CN1 for power input. Use the E_24V and GND pins of CN2 for power input.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.6 CN3 Pin Assignments: Manual Pulse Input (PCI-8164 Only)

CN3 is for manual pulser input.

No.	Name	Function (Axis)
1	GND	Bus power ground
2	PB4	Pulser B-phase signal input, ④
3	PA4	Pulser A-phase signal input, ④
4	PB3	Pulser B-phase signal input, ③
5	PA3	Pulser A-phase signal input, ③
6	+5V	Bus power, +5V
7	GND	Bus power ground
8	PB2	Pulser B-phase signal input, ②
9	PA2	Pulser A-phase signal input, ②
10	PB1	Pulser B-phase signal input, ①
11	PA1	Pulser A-phase signal input, ①
12	+5V	Bus power, +5V

Note: +5V and GND pins are provided by the PCI-Bus. Therefore, these signals are not isolated.

2.6A CN3 Pin Assignments: General Purpose DIO (MPC-8164 Only)

Pin No	Signal Name	Pin No	Signal Name
1	DOCOM	2	DOCOM
3	DOCOM	4	DOCOM
5	DO0	6	DO1
7	DO2	8	DO3
9	DO4	10	DO5
11	DO6	12	DO7
13	--	14	DICOM
15	DICOM	16	DICOM
17	DICOM	18	DI0
19	DI1	20	DI2
21	DI3	22	DI4
23	DI5	24	DI6
25	DI7	26	--

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.7 CN2 Pin Assignments: Main connector

CN2 is the major connector for the motion control I/O signals.

No.	Name	I/O	Function (axis①/②)	No.	Name	I/O	Function (axis③/④)
1	VPP	O	+5V power supply output	51	VPP	O	+5V power supply output
2	GND		Ext. power ground	52	GND		Ext. power ground
3	OUT1+	O	Pulse signal (+),①	53	OUT3+	O	Pulse signal (+), ③
4	OUT1-	O	Pulse signal (-),①	54	OUT3-	O	Pulse signal (-),③
5	DIR1+	O	Dir. signal (+),①	55	DIR3+	O	Dir. signal (+), ③
6	DIR1-	O	Dir. signal (-),①	56	DIR3-	O	Dir. signal (-), ③
7	SVON1	O	Multi-purpose signal, ①	57	SVON3	O	Multi-purpose signal, ③
8	ERC1	O	Dev. ctr, clr. signal, ①	58	ERC3	O	Dev. ctr, clr. signal, ③
9	ALM1	I	Alarm signal, ①	59	ALM3	I	Alarm signal, ③
10	INP1	I	In-position signal, ①	60	INP3	I	In-position signal, ③
11	RDY1	I	Multi-purpose signal, ①	61	RDY3	I	Multi-purpose signal, ③
12	GND		Ext. power ground	62	EXGND		Ext. power ground
13	EA1+	I	Encoder A-phase (+), ①	63	EA3+	I	Encoder A-phase (+), ③
14	EA1-	I	Encoder A-phase (-), ①	64	EA3-	I	Encoder A-phase (-),③
15	EB1+	I	Encoder B-phase (+), ①	65	EB3+	I	Encoder B-phase (+),③
16	EB1-	I	Encoder B-phase (-), ①	66	EB3-	I	Encoder B-phase (-),③
17	EZ1+	I	Encoder Z-phase (+), ①	67	EZ3+	I	Encoder Z-phase (+),③
18	EZ1-	I	Encoder Z-phase (-), ①	68	EZ3-	I	Encoder Z-phase (-),③
19	VPP	O	+5V power supply output	69	VPP	O	+5V power supply output
20	GND		Ext. power ground	70	GND		Ext. power ground
21	OUT2+	O	Pulse signal (+), ②	71	OUT4+	O	Pulse signal (+),④
22	OUT2-	O	Pulse signal (-), ②	72	OUT4-	O	Pulse signal (-),④
23	DIR2+	O	Dir. signal (+), ②	73	DIR4+	O	Dir. signal (+),④
24	DIR2-	O	Dir. signal (-), ②	74	DIR4-	O	Dir. signal (-),④
25	SVON2	O	Multi-purpose signal, ②	75	SVON4	O	Multi-purpose signal, ④
26	ERC2	O	Dev. ctr, clr. signal, ②	76	ERC4	O	Dev. ctr, clr. signal, ④
27	ALM2	I	Alarm signal, ②	77	ALM4	I	Alarm signal, ④
28	INP2	I	In-position signal, ②	78	INP4	I	In-position signal, ④
29	RDY2	I	Multi-purpose signal, ②	79	RDY4	I	Multi-purpose signal, ④
30	GND		Ext. power ground	80	GND		Ext. power ground
31	EA2+	I	Encoder A-phase (+), ②	81	EA4+	I	Encoder A-phase (+), ④
32	EA2-	I	Encoder A-phase (-), ②	82	EA4-	I	Encoder A-phase (-), ④
33	EB2+	I	Encoder B-phase (+), ②	83	EB4+	I	Encoder B-phase (+), ④
34	EB2-	I	Encoder B-phase (-), ②	84	EB4-	I	Encoder B-phase (-), ④
35	EZ2+	I	Encoder Z-phase (+), ②	85	EZ4+	I	Encoder Z-phase (+), ④
36	EZ2-	I	Encoder Z-phase (-), ②	86	EZ4-	I	Encoder Z-phase (-), ④
37	PEL1	I	End limit signal (+), ①	87	PEL3	I	End limit signal (+), ③
38	MEL1	I	End limit signal (-), ①	88	MEL3	I	End limit signal (-), ③
39	CMP1	O	Position compare output ①	89	LTC3	I	Position latch input ③
40	SD/PCS1	I	Ramp-down signal ①	90	SD/PCS3	I	Ramp-down signal ③
41	ORG1	I	Origin signal, ①	91	ORG3	I	Origin signal, ③
42	GND		Ext. power ground	92	GND		Ext. power ground
43	PEL2	I	End limit signal (+), ②	93	PEL4	I	End limit signal (+), ④
44	MEL2	I	End limit signal (-), ②	94	MEL4	I	End limit signal (-), ④
45	CMP2	O	Position compare output ②	95	LTC4	I	Position latch input, ④
46	SD/PCS2	I	Ramp-down signal ②	96	SD/PCS4	I	Ramp-down signal ④
47	ORG2	I	Origin signal, ②	97	ORG4	I	Origin signal, ④
48	GND		Ext. power ground	98	GND		Ext. power ground
49	GND		Ext. power ground	99	E_24V		Ext. power supply, +24V
50	GND		Ext. power ground	100	E_24V		Ext. power supply, +24V

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.8 CN4 Pin Assignments: Simultaneous Start/Stop (PCI-8164 Only)

CN4 is for simultaneous start/stop signals for multiple axes or multiple cards.

No.	Name	Function (Axis)
1	GND	Bus power ground
2	STP	Simultaneous stop signal input/output
3	STA	Simultaneous start signal input/output
4	STP	Simultaneous stop signal input/output
5	STA	Simultaneous start signal input/output
6	+5V	Bus power output, +5V

Note: +5V and GND pins are provided by the PCI Bus power.

2.9 CN5 Pin Assignment: TTL Output (PCI-8164 Only)

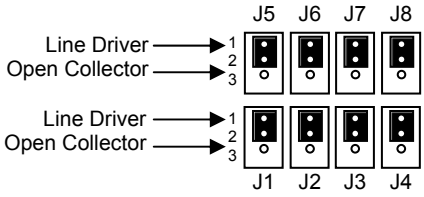
CN5 is for general-purposed TTL output signals.

Pin No.	Name	Function
1	DGND	Digital ground
2	DGND	Digital ground
3	ED0	Digital Output 0
4	ED1	Digital Output 1
5	ED2	Digital Output 2
6	ED3	Digital Output 3
7	ED4	Digital Output 4
8	ED5	Digital Output 5
9	VCC	VCC +5V
10	N.C.	Not used

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.10 Jumper Setting for Pulse Output (PCI-8164 Only)

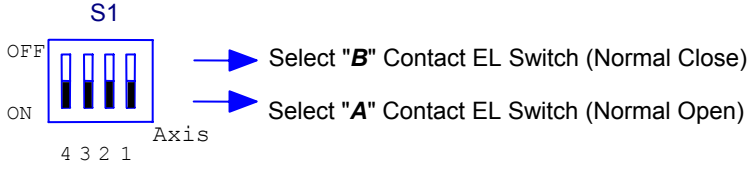
J1-J8 are used to set the type of pulse output signals (DIR and OUT). The output signal type can either be differential line driver or open collector output. Refer to section 3.1 for detail jumper settings. The default setting is differential line driver mode.



2.11 Switch Setting for EL Logic

The S1 switch is used to set the EL limit switching type. The default setting of the EL switch is ON, which is the “normally open” position (or "A" contact type), while OFF is the “normally closed” position (or “B” contact type).

For safety reasons, users must set a type, which will make the end-limit active when it is broken or disconnected.



Note: MPC-8164 uses SW2 for this setting.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2.12 CN3 Pin Assignment: General Purpose DI/DO ports (MPC-8164 Only)

CN3 Pin No	Signal Name	CN3 Pin No	Signal Name
1	DOCOM	2	DOCOM
3	DOCOM	4	DOCOM
5	DO0	6	DO1
7	DO2	8	DO3
9	DO4	10	DO5
11	DO6	12	DO7
13	--	14	DICOM
15	DICOM	16	DICOM
17	DICOM	18	DI0
19	DI1	20	DI2
21	DI3	22	DI4
23	DI5	24	DI6
25	DI7	26	--

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3

Signal Connections

Signal connections of all I/O's are described in this chapter. Refer to the contents of this chapter before wiring any cables between the 8164 and any motor drivers.

This chapter contains the following sections:

Section 3.1 Pulse Output Signals OUT and DIR
Section 3.2 Encoder Feedback Signals EA, EB and EZ
Section 3.3 Origin Signal ORG
Section 3.4 End-Limit Signals PEL and MEL
Section 3.5 Ramping-down & PCS signals
Section 3.6 In-position signals INP
Section 3.7 Alarm signal ALM
Section 3.8 Deviation counter clear signal ERC
Section 3.9 general-purposed signals SVON
Section 3.10 General-purposed signal RDY
Section 3.11 Position compare output pin: CMP
Section 3.12 Position latch input pin: LTC
Section 3.13 Pulse input signals PA and PB
Section 3.14 Simultaneous start/stop signals STA and STP
Section 3.15 General-purposed TTL DIO
Section 3.16 Termination Board
Section 3.17 General-purposed DIO

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.1 Pulse Output Signals OUT and DIR

There are 4 axis pulse output signals on the 8164. For each axis, two pairs of OUT and DIR signals are used to transmit the pulse train and to indicate the direction. The OUT and DIR signals can also be programmed as CW and CCW signal pairs. Refer to section 4.1.1 for details of the logical characteristics of the OUT and DIR signals. In this section, the electrical characteristics of the OUT and DIR signals are detailed. Each signal consists of a pair of differential signals. For example, OUT2 consists of OUT2+ and OUT2- signals. The following table shows all pulse output signals on CN2.

CN2 Pin No.	Signal Name	Description	Axis #
3	**OUT1+**	Pulse signals (+)	①
4	**OUT1-**	Pulse signals (-)	①
5	**DIR1+**	Direction signal (+)	①
6	**DIR1-**	Direction signal (-)	①
21	**OUT2+**	Pulse signals (+)	②
22	**OUT2-**	Pulse signals (-)	②
23	**DIR2+**	Direction signal (+)	②
24	**DIR2-**	Direction signal (-)	②
53	**OUT3+**	Pulse signals (+)	③
54	**OUT3-**	Pulse signals (-)	③
55	**DIR3+**	Direction signal (+)	③
56	**DIR3-**	Direction signal (-)	③
71	**OUT4+**	Pulse signals (+)	④
72	**OUT4-**	Pulse signals (-)	④
73	**DIR4+**	Direction signal (+)	④
74	**DIR4-**	Direction signal (-)	④

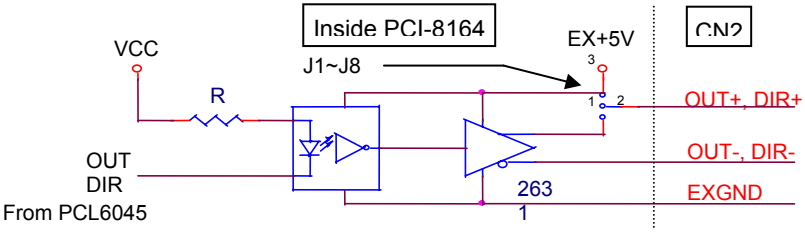
The output of the OUT or DIR signals can be configured by jumpers as either differential line drivers or open collector output. Users can select the output mode either by closing breaks between 1 and 2 or 2 and 3 of jumpers J1-J8 as follows:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Output Signal	For differential line driver output, close breaks between 1 and 2 of:	For open collector output, close breaks between 2 and 3 of:
OUT1-	J1	J1
DIR1-	J2	J2
OUT2-	J3	J3
DIR2-	J4	J4
OUT3-	J5	J5
DIR3-	J6	J6
OUT4-	J7	J7
DIR4-	J8	J8

The **default** setting of OUT and DIR is set to differential line driver mode.

The following wiring diagram is for OUT and DIR signals on the 4 axes.



NOTE: If the pulse output is set to open collector output mode, OUT- and DIR- are used to transmit OUT signals. The sink current must not exceed 20mA on the OUT- and DIR- pins. The current may be provided by the EX+5V power source, however, note that the maximum capacity of the EX+5V source is 500mA.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

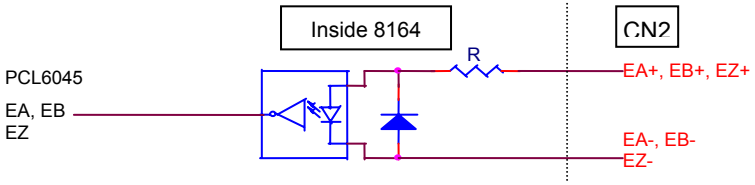
3.2 Encoder Feedback Signals EA, EB and EZ

The encoder feedback signals include EA, EB, and EZ. Every axis has six pins for three differential pairs of phase-A (EA), phase-B (EB), and index (EZ) inputs. EA and EB are used for position counting, and EZ is used for zero position indexing. Its relative signal names, pin numbers, and axis numbers are shown in the following tables:

CN2 Pin No	Signal Name	Axis #	CN2 Pin No	Signal Name	Axis #
13	EA1+	①	63	EA3+	③
14	EA1-	①	64	EA3-	③
15	EB1+	①	65	EB3+	③
16	EB1-	①	66	EB3-	③
31	EA2+	②	81	EA4+	④
32	EA2-	②	82	EA4-	④
33	EB2+	②	83	EB4+	④
34	EB2-	②	84	EB4-	④

CN2 Pin No	Signal Name	Axis #	CN2 Pin No	Signal Name	Axis #
17	EZ1+	①	67	EZ3+	③
18	EZ1-	①	68	EZ3-	③
35	EZ2+	②	85	EZ4+	④
36	EZ2-	②	86	EZ4-	④

The input circuit of the EA, EB, and EZ signals is shown as follows:



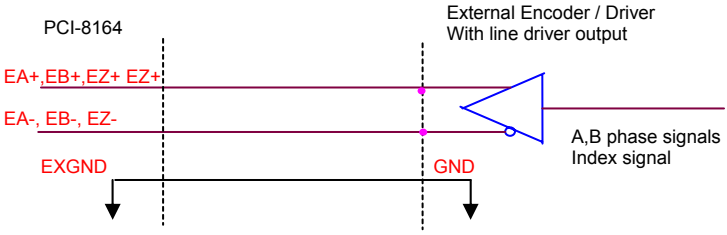
Please note that the voltage across each differential pair of encoder input signals (EA+, EA-), (EB+, EB-), and (EZ+, EZ-) should be at least 3.5V. Therefore, the output current must be observed when connecting to the encoder feedback or motor driver feedback as not to over drive the source. The differential signal pairs are converted to digital signals EA, EB, and EZ; then feed to the PCL6045 ASIC.

Below are examples of connecting the input signals with an external circuit. The input circuit can be connected to an encoder or motor driver if it is equipped with: (1) a differential line driver or (2) an open collector output.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

◆ Connection to Line Driver Output

To drive the 8164 encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 6mA driving capacity. The grounds of both sides must be tied together.

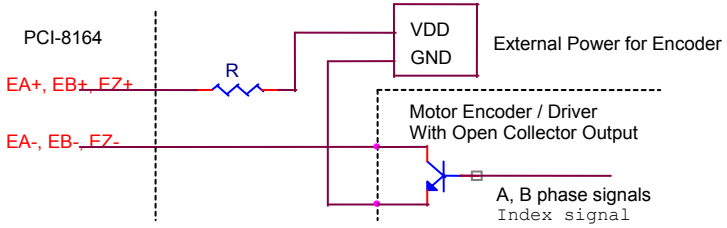


◆ Connection to Open Collector Output

To connect with an open collector output, an external power supply is necessary. Some motor drivers can provide the power source. The connection between the 8164, encoder, and the power supply is shown in the diagram below. Note that an external current limiting resistor R is necessary to protect the 8164 input circuit. The following table lists the suggested resistor values according to the encoder power supply.

Encoder Power (VDD)	External Resistor R
+5V	0Ω (None)
+12V	1.8kΩ
+24V	4.3kΩ

I_f = 6mA max.



For more operation information on the encoder feedback signals, refer to section 4.4.

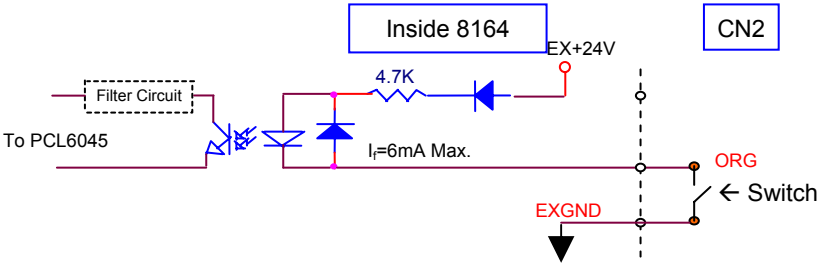
LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.3 Origin Signal ORG

The origin signals (ORG1~ORG4) are used as input signals for the origin of the mechanism. The following table lists signal names, pin numbers, and axis numbers:

CN2 Pin No	Signal Name	Axis #
41	ORG1	①
47	ORG2	②
91	ORG3	③
97	ORG4	④

The input circuit of the ORG signals is shown below. Usually, a limit switch is used to indicate the origin on one axis. The specifications of the limit switch should have contact capacity of +24V @ 6mA minimum. An internal filter circuit is used to filter out any high frequency spikes, which may cause errors in the operation.



When the motion controller is operated in the home return mode, the ORG signal is used to inhibit the control output signals (OUT and DIR). For detailed operations of the ORG signal, refer to section 4.3.3.

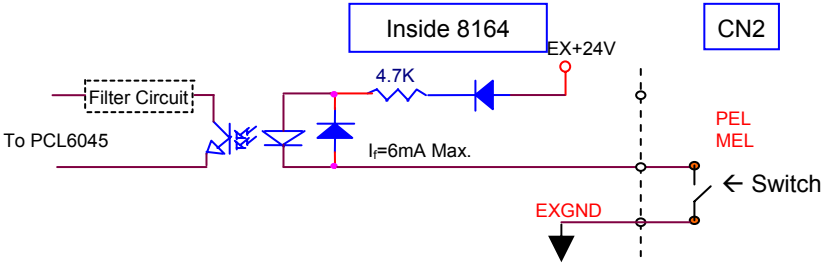
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.4 End-Limit Signals PEL and MEL

There are two end-limit signals PEL and MEL for each axis. PEL indicates the end limit signal is in the plus direction and MEL indicates the end limit signal is in the minus direction. The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #	CN2 Pin No	Signal Name	Axis #
37	PEL1	①	87	PEL3	③
38	MEL1	①	88	MEL3	③
43	PEL2	②	93	PEL4	④
44	MEL2	②	94	MEL4	④

A circuit diagram is shown in the diagram below. The external limit switch should have a contact capacity of +24V @ 6mA minimum. Either 'A-type' (normal open) contact or 'B-type' (normal closed) contact switches can be used. To set the type of switch, configure dipswitch S1/SW2. The 8164 is defaulted with all bits of S1 set to ON (refer to section 2.10). For more details on EL operation, refer to section 4.3.2.



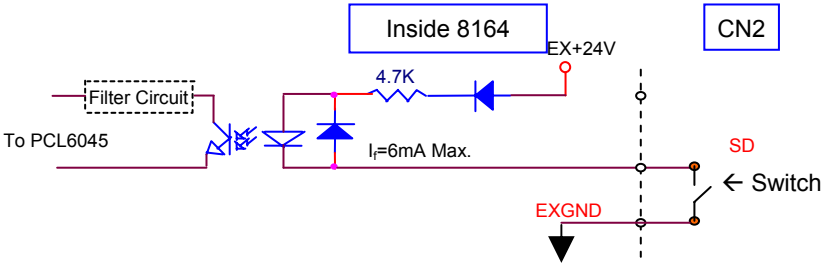
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.5 Ramping-down & PCS

There is a SD/PCS signal for each of the 4 axes. The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #
40	SD1/PCS1	①
46	SD2/PCS2	②
90	SD3/PCS3	③
96	SD4/PCS4	④

A circuit diagram is shown below. Typically, the limit switch is used to generate a slow-down signal to drive motors operating at slower speeds. For more details on SD/PCS operation, refer to section 4.3.1.



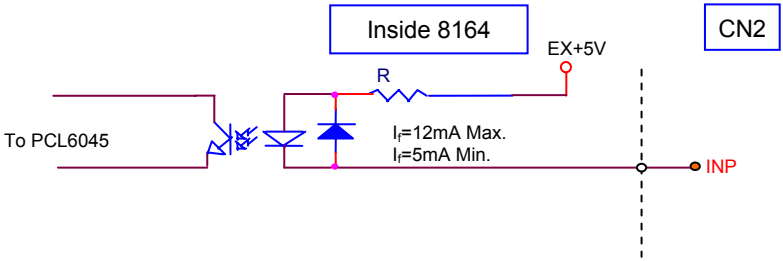
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.6 In-position Signal INP

The in-position signal INP from a servo motor driver indicates its deviation error. If there is no deviation error then the servo's position indicates zero. The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #
10	INP1	①
28	INP2	②
60	INP3	③
78	INP4	④

The input circuit of the INP signals is shown in the diagram below:



The in-position signal is usually generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 5mA current sink capabilities to drive the INP signal. For more details of INP signal operations, refer to section 4.2.1.

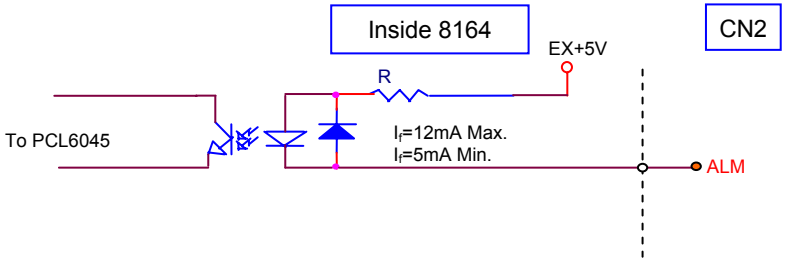
Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.7 Alarm Signal ALM

The alarm signal ALM is used to indicate the alarm status from the servo driver. The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #
9	ALM1	①
27	ALM2	②
59	ALM3	③
77	ALM4	④

The input alarm circuit is shown below. The ALM signal usually is generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 5mA current sink capabilities to drive the ALM signal. For more details of ALM signal operations, refer to section 4.2.2.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.8 Deviation Counter Clear Signal ERC

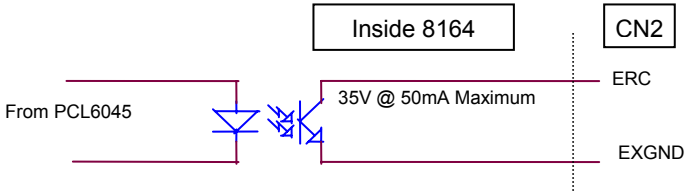
The deviation counter clear signal (ERC) is active in the following 4 situations:

1. Home return is complete
2. End-limit switch is active
3. An alarm signal stops OUT and DIR signals
4. An emergency stop command is issued by software (operator)

The signal names, pin numbers, and axis numbers are shown in the table below:

CN2 Pin No	Signal Name	Axis #
8	ERC1	①
26	ERC2	②
58	ERC3	③
76	ERC4	④

The ERC signal is used to clear the deviation counter of the servomotor driver. The ERC output circuit is an open collector with a maximum of 35V at 50mA driving capacity. For more details of ERC operation, refer to section 4.2.3.



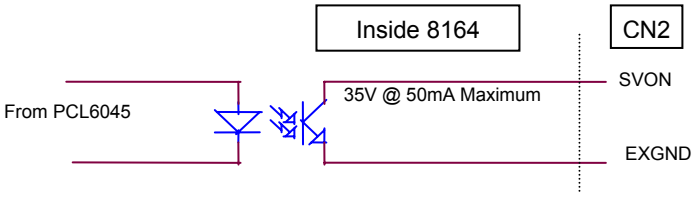
LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.9 General-purpose Signal SVON

The SVON signal can be used as a servomotor-on control or general purpose output signal. The signal names, pin numbers, and its axis numbers are shown in the following table:

CN2 Pin No	Signal Name	Axis #
7	SVON1	①
25	SVON2	②
57	SVON3	③
75	SVON4	④

The output circuit for the SVON signal is shown below:

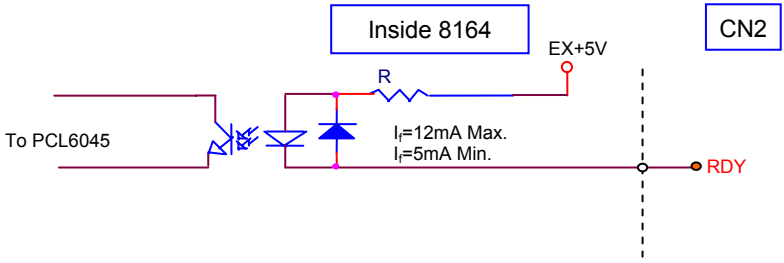


3.10 General-purpose Signal RDY

The RDY signals can be used as motor driver ready input or general purpose input signals. The signal names, pin numbers, and axis numbers are shown in the following table:

CN2 Pin No	Signal Name	Axis #
11	RDY1	①
29	RDY2	②
61	RDY3	③
79	RDY4	④

The input circuit of RDY signal is shown in the following diagram:



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

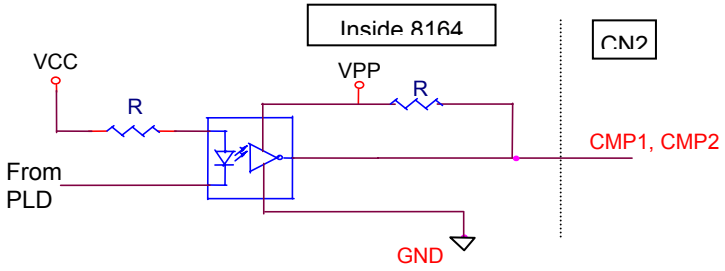
3.11 Position compare output pin: CMP

The 8164 provides 2 comparison output channels, CMP1 and CMP2 used by the first 2 axes, ① & ②. The comparison output channel will generate a pulse signal when the encoder counter reaches a pre-set value set by the user.

The CMP channel is located on CN2. The signal names, pin numbers, and axis numbers are shown below:

CN2 Pin No	Signal Name	Axis #
39	CMP1	①
45	CMP2	②

The following wiring diagram is of the CMP on the first 2 axes:



Note: CMP trigger type can be set as normal low (rising edge) or normal high (falling edge). Default setting is normal high. Refer to function ***_8164_set_trigger_type()*** in section 6.16 for details.

This CMP pin can be regarded as a TTL output.

In the above figure:

VPP: Isolated +5V

VCC: Computer +5V

R1: 470 Ohms

R2: 1K Ohms

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

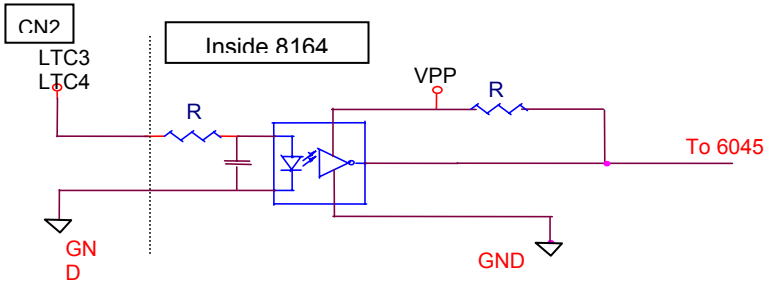
3.12 Position latch input pin: LTC

The 8164 provides 2 position latch input channels, LTC3 and LTC4 used by the last 2 axes, ③ & ④. The LTC signal will trigger the counter-value-capturing functions, which provides a precise position determination.

The LTC channel is on CN2. The signal names, pin numbers, and axis numbers are shown in the following table:

CN2 Pin No	Signal Name	Axis #
89	LTC3	③
95	LTC4	④

The following wiring diagram is for the LTC of the last 2 axes:



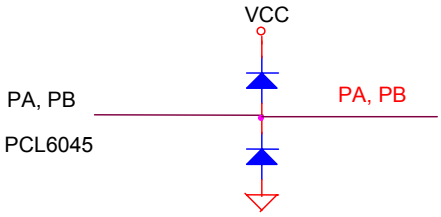
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.13 Pulser Input Signals PA and PB (PCI-8164)

The PCI-8164 can accept input pulser signals through the pins of CN3 listed below. The pulses behaves like an encoder. The signals generate the positioning information, which guides the motor.

CN3 Pin No	Signal Name	Axis #	CN3 Pin No	Signal Name	Axis #
11	PA1	①	5	PA3	③
10	PB1	①	4	PB3	③
9	PA2	②	3	PA4	④
8	PB2	②	2	PB4	④

PA and PB pins of CN3 are directly connected to PA and PB pins of the PCL6045. The interface circuit is shown below.



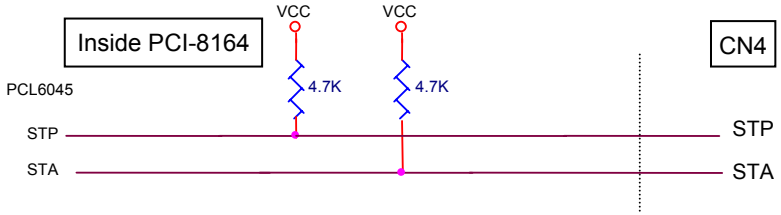
If the signal voltage of the pulser is not +5V or if the pulser is distantly placed, we recommended that a photocoupler or line driver be placed in between. Note that the +5V and GND lines of CN3 are provided from the PCI bus. Note that this source is not isolated.

3.14 Simultaneously Start/Stop Signals STA and STP(PCI-8164 Only)

The PCI-8164 provides STA and STP signals, which enable simultaneous start/stop of motions on multiple axes. The STA and STP signals are on CN4.

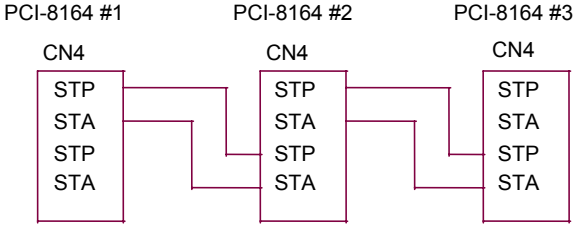
The diagram below shows the onboard circuit. The STA and STP signals of the four axes are tied together respectively.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

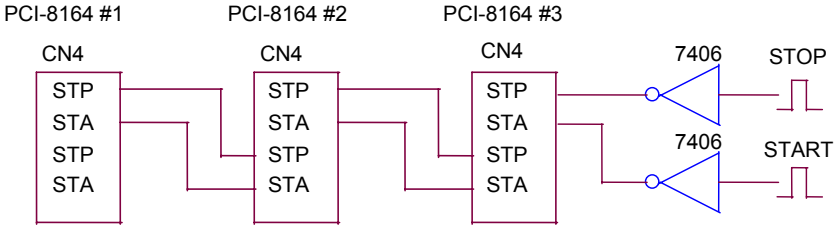


The STP and STA signals are both input and output signals. To operate the start and stop action simultaneously, both software control and external control are needed. With software control, the signals can be generated from any one of the PCL6045. Users can also use an external open collector or switch to drive the STA/STP signals for simultaneous start/stop.

If there are two or more PCI-8164 cards, cascade the CN4 connectors of all cards for simultaneous start/stop control on all concerned axes. In this case, connect CN4 as below:



To allow an external signal to initiate the simultaneous start/stop connect a 7406 (open collector) or an equivalent circuit as shown below:



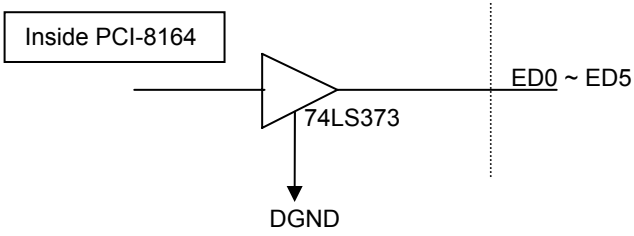
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.15 General Purpose TTL Output (PCI-8164 only)

The PCI-8164 provides 6 general purpose TTL digital outputs. The TTL output is on CN5. The signal names, pin numbers, and axis numbers are shown in the table below:

Pin No.	Name	Function
1	DGND	Digital ground
2	DGND	Digital ground
3	ED0	Digital Output 0
4	ED1	Digital Output 1
5	ED2	Digital Output 2
6	ED3	Digital Output 3
7	ED4	Digital Output 4
8	ED5	Digital Output 5
9	VCC	VCC +5V

The following wiring diagram is for the LTC of the last 2 axes:



3.16 Termination Board

CN2 of the 8164 can be connected with a DIN-100S, including the ACL-102100 cable (a 100-pin SCSI-II cable). The DIN-100S is a general purpose 100-pin SCSI-II DIN-socket. It has easy wiring screw terminals and an easily installed DIN socket that can be mounted onto the DIN rails

We also provide DIN-814M termination boards for Mitsubishi JS2 Servo Motor Drivers.

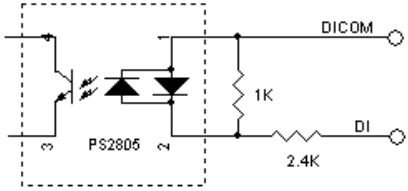
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.17 General Purpose DIO (MPC-8164 only)

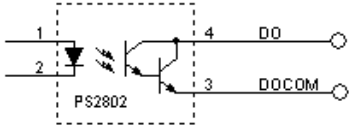
There are 8 opto-isolated digital outputs and 8 open collector digital inputs for general purpose use. Pin assignments are illustrated in the table below:

CN3 Pin No	Signal Name	CN3 Pin No	Signal Name
1	DOCOM	2	DOCOM
3	DOCOM	4	DOCOM
5	DO0	6	DO1
7	DO2	8	DO3
9	DO4	10	DO5
11	DO6	12	DO7
13	--	14	DICOM
15	DICOM	16	DICOM
17	DICOM	18	DI0
19	DI1	20	DI2
21	DI3	22	DI4
23	DI5	24	DI6
25	DI7	26	--

3.17.1 Isolated Input channels

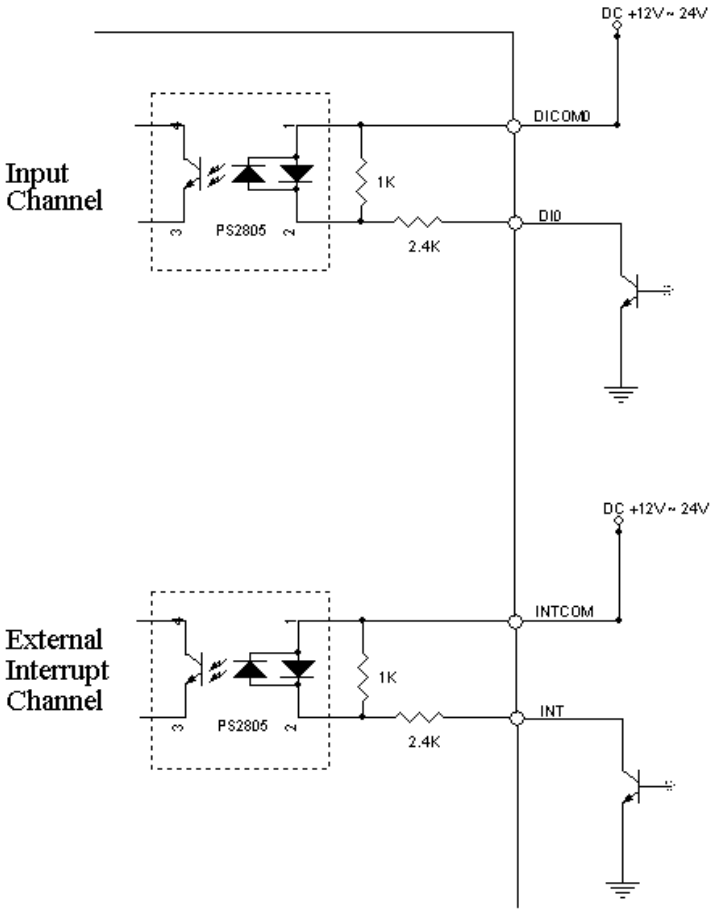


3.17.2 Isolated Output channels



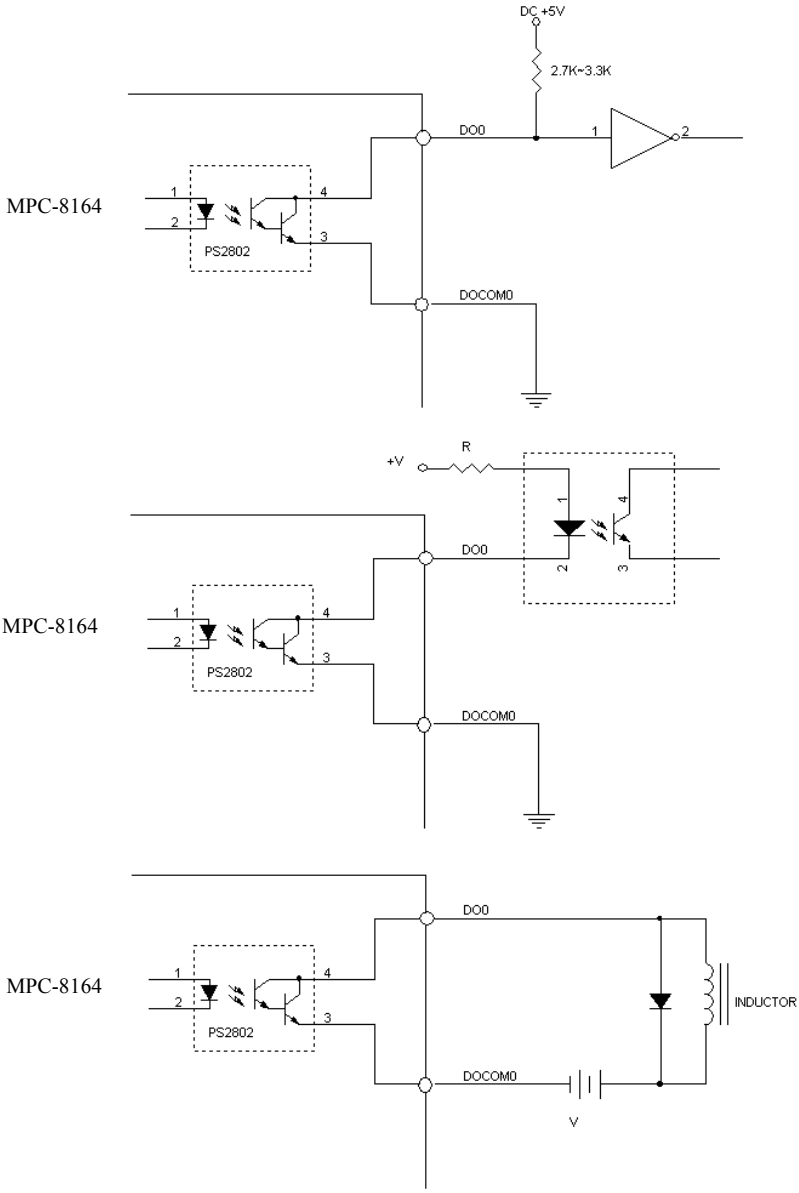
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.17.3 Example of input connection



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3.17.4 Example of output connection



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4

Operation Theory

This chapter describes the detail operation of the 8164 card. Contents of the following sections are as follows:

Section 4.1: The motion control modes
Section 4.2: The motor driver interface (INP, ERC, ALM, SVON, RDY)
Section 4.3: The limit switch interface and I/O status (SD/PCS, EL, ORG)
Section 4.4: The counters (EA, EB, EZ)
Section 4.5: Multiple 8164 cards operation.
Section 4.6: Change position or speed on the fly
Section 4.7: Position compare and latch
Section 4.8: Hardware backlash compensator
Section 4.9: Software limit function
Section 4.10: Interrupt control

4.1 Motion Control Modes

In this section, the pulse output signal configuration and the following motion control modes are described.

- 4.1.1 Pulse command output
- 4.1.2 Velocity mode motion for one axis
- 4.1.3 Trapezoidal motion for one axis
- 4.1.4 S-Curve profile motion for one axis
- 4.1.5 Linear interpolation for 2-4 axes
- 4.1.6 Circular interpolation for 2 axes
- 4.1.7 Continuous motion
- 4.1.8 Home return mode for one axis
- 4.1.9 Manual pulse mode for one axis

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.1 Pulse Command Output

The 8164 uses pulse commands to control servo/stepper motors via the drivers. A pulse command consists of two signals: OUT and DIR. There are two command types: (1) single pulse output mode (OUT/DIR), and (2) dual pulse output mode (CW/CCW type pulse output). The software function, **_8164_set_pls_outmode()**, is used to program the pulse command mode. The modes vs. signal type of OUT and DIR pins are listed in the table below:

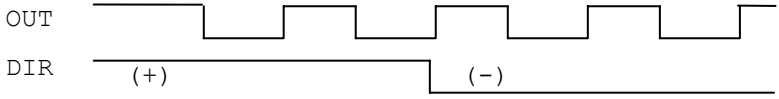
Mode	Output of OUT pin	Output of DIR pin
Dual pulse output (CW/CCW)	Pulse signal in plus (or CW) direction	Pulse signal in minus (or CCW) direction
Single pulse output (OUT/DIR)	Pulse signal	Direction signal (level)

The interface characteristics of these signals can be differential line driver or open collector output. Please refer to section 3.1 for the jumper setting for different signal types.

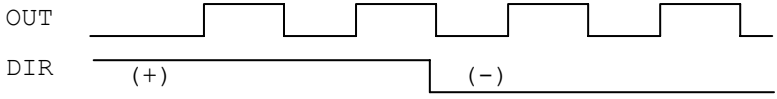
Single Pulse Output Mode (OUT/DIR Mode)

In this mode, the OUT signal is for the command pulse (position or velocity) chain. The numbers of OUT pulse represent the relative “distance” or “position.” The frequency of the OUT pulse represents the command for “speed” or “velocity.” The DIR signal represents direction command of positive (+) or negative (-). This mode is most commonly used. The diagrams below show the output waveform. It is possible to set the polarity of the pulse chain.

pls_outmode = 0:

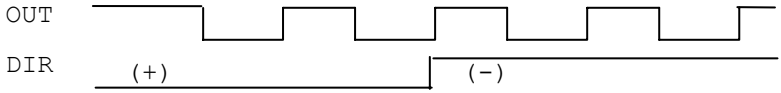


pls_outmode = 1:

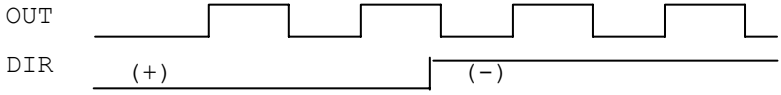


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

pls_outmode = 2:



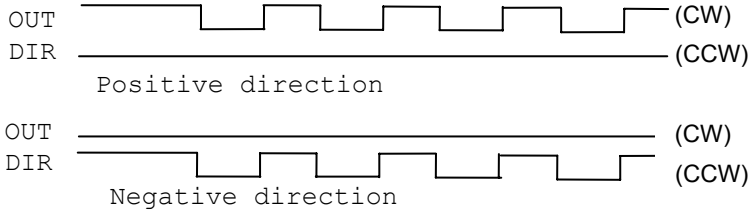
pls_outmode = 3:



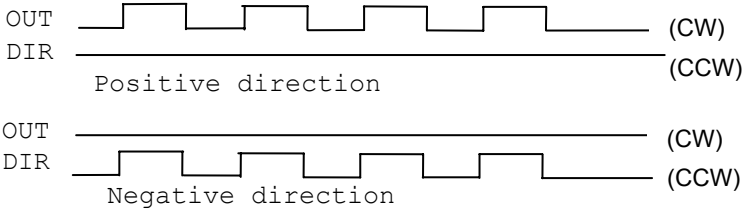
Dual Pulse Output Mode (CW/CCW Mode)

In this mode, the waveform of the OUT and DIR pins represent CW (clockwise) and CCW (counter clockwise) pulse output respectively. Pulses output from the CW pin makes the motor move in positive direction, whereas pulse output from the CCW pin makes the motor move in negative direction. The following diagram shows the output waveform of positive (+) commands and negative (-) commands.

pls_outmode = 4:



pls_outmode = 5:



Relative Function:

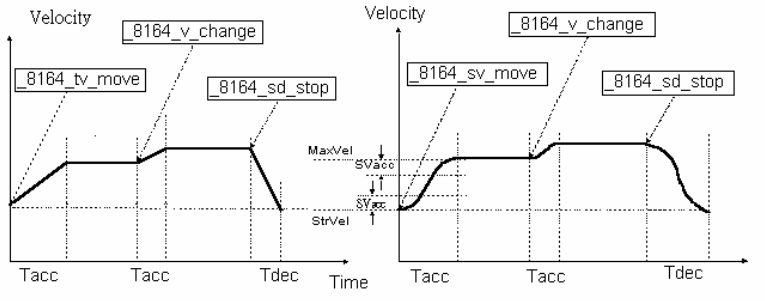
_8164_set_pls_outmode(): Refer to section 6.4

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.2 Velocity mode motion

This mode is used to operate a one-axis motor with Velocity mode motion. The output pulse accelerates from a starting velocity (StrVel) to a specified maximum velocity (MaxVel). The **_8164_tv_move()** function is used for constant linear acceleration while the **_8164_sv_move()** function is use for acceleration according to the S-curve. The pulse output rate is kept at maximum velocity until another velocity command is set or a stop command is issued. The **_8164_v_change()** is used to change the speed during an operation. Before this function is applied, be sure to call **_*8164*_fix_speed_range()**. Please refer to section 4.6 for more detail explanation. The **_8164_sd_stop()** function is used to decelerate the motion until it stops. The **_8164_emg_stop()** function is used to immediately stop the motion. These change or stop functions follow the same velocity profile as its original move functions, tv_move or sv_move. The velocity profile is shown as follows:

Note: The v_change and stop functions can also be applied to **Preset Mode** (both trapezoidal, refer to 4.1.3, and S-curve Motion, refer to 4.1.4) or **Home Mode** (refer to 4.1.8)**.**



Relative Functions:

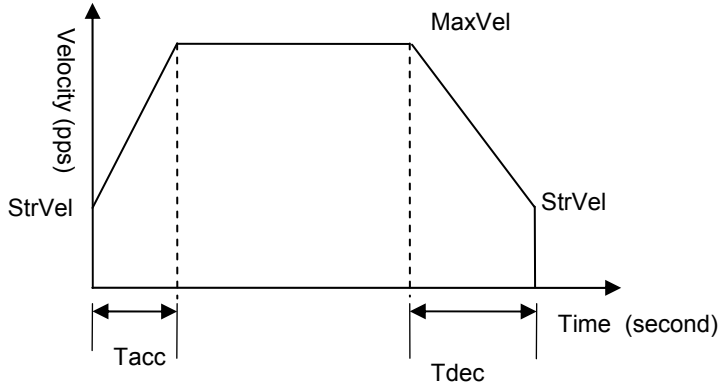
_8164_tv_move(), _8164_sv_move(), _8164_v_change(), _8164_sd_stop(), _8164_emg_stop(), _8164_fix_speed_range(), _8164_unfix_speed_range(): Refer to section 6.5

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.3 Trapezoidal Motion

This mode is used to move a singe axis motor to a specified position (or distance) with a trapezoidal velocity profile. The single axis is controlled from point to point. An absolute or relative motion can be performed. In absolute mode, the target position is assigned. In relative mode, the target displacement is assigned. In both cases, the acceleration and deceleration can be different. The function _8164_motion_done() is used to check whether the movement is complete.

The following diagram shows the trapezoidal profile:



There are 2 trapezoidal point-to-point functions supported by the 8164. In the **_8164_start_ta_move()** function, the absolute target position must be given in units of pulses. The physical length or angle of one movement is dependent on the motor driver and mechanism (including the motor). Since absolute move mode needs the information of current actual position, the "External encoder feedback (EA, EB pins)" should be set in **_8164_set_feedback_src()** function. The ratio between command pulses and external feedback pulse input must be appropriately set by the **_8164_set_move_ratio()** function.

In the **_8164_start_tr_move()** function, the relative displacement must be given in units of pulses. Unsymmetrical trapezoidal velocity profile (Tacc is not equal Tdec) can be specified with both **_8164_start_ta_move()** and **_8164_start_tr_move()** functions.

The StrVel and MaxVel parameters are given in units of pulses per second (PPS). The Tacc and Tdec parameters are in units of second to represent accel./decel. time respectively. Users need to know the physical meaning of "one pulse" to calculate the physical value of the relative velocity or acceleration parameters. The following formula gives the basic relationship between these parameters:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

MaxVel = StrVel + accel*Tacc;
StrVel = MaxVel + decel *Tdec;

Where accel/decel represents the acceleration/deceleration rate in units of pps/sec^2. The area inside the trapezoidal profile represents the moving distance.

Units of velocity setting are pulses per second (PPS). Usually, units of velocity of the manual of motor or driver are in rounds per minute (RPM). A simple conversion is necessary to match between these two units. Here we use an example to illustrate the conversion:

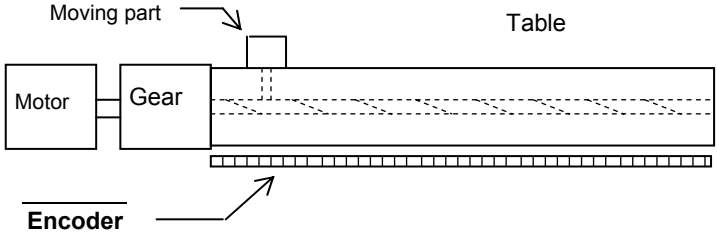
For example:

A servomotor with an AB phase encoder is used in a X-Y table. The resolution of encoder is 2000 counts per phase. The maximum rotating speed of motor is designed to be 3600 RPM. What is the maximum pulse command output frequency that you have to set on 8164?

Answer: MaxVel = 3600/60*2000*4 = 480000 PPS

Multiplying by 4 is necessary because there are four states per AB phase (See Figures in Section 4.4).

Usually, the axes need to set the move ratio if their mechanical resolution is different from the resolution of command pulse. For example, if an incremental encoder is mounted on the working table to measure the actual position of moving part. A servomotor is used to drive the moving part through a gear mechanism. The gear mechanism is used to convert the rotating motion of the motor into linear motion (see the following diagram). If the resolution of the motor is 8000 pulses/round, then the resolution of the gear mechanism is 100 mm/round (i.e., part moves 100 mm if the motor turns one round). Then, the resolution of the command pulse will be 80 pulses/mm. If the resolution of the encoder mounting on the table is 200 pulses/mm, then users have to set the move ratio to 200/80=2.5 using the function _8164_set_move_ratio (axis, 2.5).



If this ratio is not set before issuing the start moving command, it will cause problems when running in “Absolute Mode” because the 8164 won’t recognize the actual absolute position during motion.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Relative Functions:

_8164_start_ta_move(), _8164_start_tr_move(): Refer to section 6.6

_8164_motion_done(): Refer to section 6.11

_8164_set_feedback_src(): Refer to section 6.4

_8164_set_move_ratio(): Refer to section 6.6

4.1.4 S-curve Profile Motion

This mode is used to move a single-axis motor to a specified position (or distance) with a S-curve velocity profile. S-curve acceleration profiles are useful for both stepper and servomotors. The smooth transitions between the start of the acceleration ramp and transition to constant velocity produce less wear and tear than a trapezoidal profile motion. The smoother performance increases the life of the motor and the mechanics of the system.

There are several parameters that need to be set in order to make a S-curve move. They are:

Pos: target position in absolute mode, in units of pulses

Dist: moving distance in relative mode, in units of pulses

StrVel: start velocity, in units of PPS

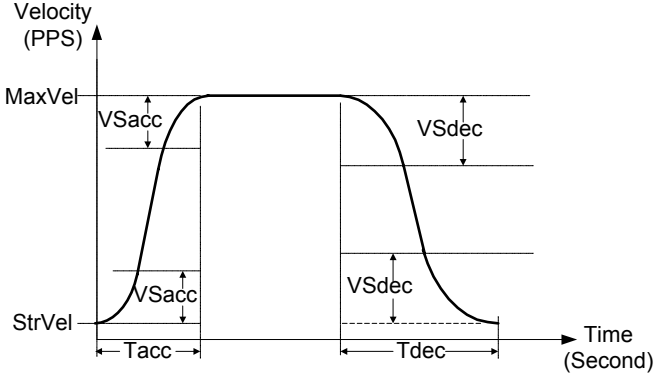
MaxVel: maximum velocity, in units of PPS

Tacc: time for acceleration (StrVel → MaxVel), in units of seconds

Tdec: time for deceleration (MaxVel → StrVel), in units of seconds

VSacc: S-curve region during acceleration, in units of PPS

VSdec: S-curve region during deceleration, in units of PPS



LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Normally, the accel/decel period consists of three regions, two VSacc/VSdec curves and one linear. During VSacc/VSdec, the jerk (second derivative of velocity) is constant, and, during the linear region, the acceleration (first derivative of velocity) is constant. In the first constant jerk region during acceleration, the velocity goes from StrVel to (StrVel + VSacc). In the second constant jerk region during acceleration, the velocity goes from (MaxVel – StrVel) to MaxVel. Between them, the linear region accelerates velocity from (StrVel + VSacc) to (MaxVel - VSacc) constantly. The deceleration period is similar in fashion.

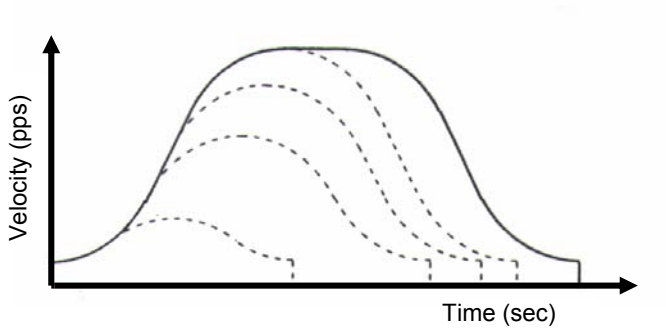
Special case:

If user wants to disable the linear region, the VSacc/VSdec must be assigned "0" rather than "0.5" (MaxVel-StrVel).

Remember that the VSacc/VSdec is in units of PPS and it should always keep in the range of [0 to (MaxVel - Strvel)/2], where "0" means no linear region.

The S-curve profile motion functions are designed to always produce smooth motion. If the time for acceleration parameters combined with the final position don't allow an axis to reach the maximum velocity (i.e. the moving distance is too small to reach MaxVel), then the maximum velocity is automatically lowered (see the following Figure).

The rule is to lower the value of MaxVel and the Tacc, Tdec, VSacc, VSdec automatically, and keep StrVel, acceleration, and jerk unchanged. This is also applicable to Trapezoidal profile motion.



Relative Functions:

_8164_start_sr_move(),_8164_start_sa_move(): Refer to section 6.6

_8164_motion_done(): Refer to section 6.11

_8164_set_feedback_src(): Refer to section 6.4

_8164_set_move_ratio(): Refer to section 6.6

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The Following table shows the differences between all single axis motion functions, including **preset mode** (both trapezoidal and S-curve motion) and **constant velocity mode**.

	Velocity Profile			
	Trapezoidal	S-Curve	Relative	Absolute
_8164_tv_move	✓		----------	----------
_8164_sv_move		✓	----------	----------
_8164_v_change	✓	✓	----------	----------
_8164_sd_stop	✓	✓	----------	----------
_8164_emg_stop()	----------	----------	----------	----------
_8164_start_ta_move	✓			✓
_8164_start_tr_move	✓		✓	
_8164_start_sr_move		✓	✓	
_8164_start_sa_move		✓		✓

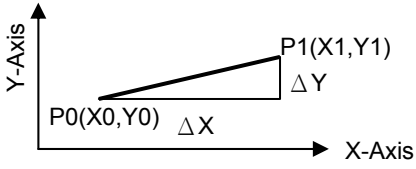
4.1.5 Linear interpolation for 2-4 axes

In this mode, any 2 of the 4, 3 of the 4, or all 4 axes may be chosen to perform linear interpolation. “Interpolation between multi-axes” means these axes start simultaneously, and reach their ending points at the same time. Linear means the ratio of speed of every axis is a constant value.

Note that you cannot use 2 groups of 2 axes for linear interpolation on a single card at the same time. You can however, use one 2-axis linear and one 2-axis circular interpolation at the same time. If you want to stop an interpolation group, the function *_8164_sd_stop() or _8164_emg_stop()* can be used.

2 axes linear interpolation

As in the diagram below, 2-axis linear interpolation means to move the XY position (or any 2 of the 4 axis) from P0 to P1. The 2 axes start and stop simultaneously, and the path is a straight line.



The speed ratio along X-axis and Y-axis is (△X: ΔY), respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2}$$

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

When calling 2-axis linear interpolation functions, the **vector speed** needs to define the start velocity, **StrVel**, and maximum velocity, **MaxVel**. Both trapezoidal and S-curve profiles are available.

Example:

_8164_start_tr_move_xy(0, 30000.0, 40000.0, **1000.0, 5000.0**, 0.1, 0.2) will cause the XY axes (axes 0 & 1) of Card 0 to perform a linear interpolation movement, in which:

$\triangle$X = 30000 pulses; $\triangle$Y = 40000 pulses

Start vector speed=1000pps, X speed=600pps, Y speed = 800pps

Max. vector speed =5000pps, X speed=3000pps, Y speed = 4000pps

Acceleration time = 0.1sec; Deceleration time = 0.2sec

There are two groups of functions that provide 2-axis linear interpolation. The first group divides the 4 axes into XY (axis 0 & axis 1) and ZU (axis 2 & axis 3). By calling these functions, the target axes are already assigned.

_8164_start_tr_move_xy(), _8164_start_tr_move_zu(),

_8164_start_ta_move_xy(), _8164_start_ta_move_zu(),

_8164_start_sr_move_xy(), _8164_start_sr_move_zu(),

_8164_start_sa_move_xy(), _8164_start_sa_move_zu()

(Refer to section 6.7)

The second group allows user to freely assign the 2 target axes.

_8164_start_tr_line2(), _8164_start_sr_line2(),

_8164_start_ta_line2(), _8164_start_sa_line2()

(Refer to section 6.7)

The characters “t”, “s”, “r”, and “a” after **_8164_start** mean:

t – Trapezoidal profile

s – S-Curve profile

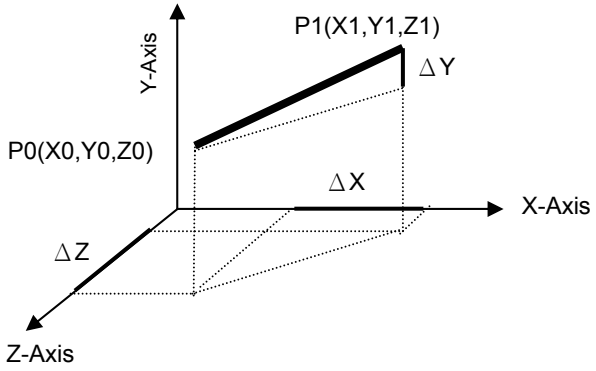
r – Relative motion

a – Absolute motion

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3-axis linear interpolation

Any 3 of the 4 axes of the 8164 may perform 3-axis linear interpolation. As shown the figure below, 3-axis linear interpolation means to move the XYZ (if axes 0, 1, 2 are selected and assigned to be X, Y, Z respectively) position from P0 to P1, starting and stopping simultaneously. The path is a straight line in space.



The speed ratio along X-axis, Y-axis, and Z-axis is (ΔX: ΔY: ΔZ), respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2 + (\frac{\Delta Z}{\Delta t})^2}$$

When calling 3-axis linear interpolation functions, the vector speed is needed to define the start velocity, **StrVel**, and maximum velocity, **MaxVel**. Both trapezoidal and S-curve profiles are available.

For example:

_8164_start_tr_line3(….,1000.0 /*ΔX */ , 2000.0/*ΔY */, 3000.0 /*DistZ*/, 100.0 /*StrVel*/, 5000.0 /* MaxVel*/, 0.1/*sec*/, 0.2 /*sec*/)

ΔX = 1000 pulse; ΔY = 2000 pulse; ΔZ = 3000 pulse

Start vector speed=100pps, X speed = 100/$\sqrt{14}$ = 26.7pps

Y speed = 2*100/$\sqrt{14}$ = 53.3pps

Z speed = 3*100/$\sqrt{14}$ = 80.1pps

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Max. vector speed =5000pps, X speed= 5000/$\sqrt{14}$ = 1336pps

Y speed = 2*5000/$\sqrt{14}$ = 2672pps

Z speed = 3*5000/$\sqrt{14}$ = 4008pps

The following functions are used for 3-axis linear interpolation:
_8164_start_tr_line3(), _8164_start_sr_line3()

_8164_start_ta_line3() , _8164_start_sa_line3()

(Refer to section 6.7)

The characters “t”, “s”, “r”, and “a” after **_8164_start** mean:

t – Trapezoidal profile

s – S-Curve profile

r – Relative motion

a – Absolute motion

4-axis linear interpolation

With 4-axis linear interpolation, the speed ratio along X-axis, Y-axis, Z-axis and U-axis is (ΔX: ΔY: ΔZ: ΔU), respectively, and the vector speed is:

$$\frac{\Delta P}{\Delta t} = \sqrt{(\frac{\Delta X}{\Delta t})^2 + (\frac{\Delta Y}{\Delta t})^2 + (\frac{\Delta Z}{\Delta t})^2 + (\frac{\Delta U}{\Delta t})^2}$$

The following functions are used for 4-axis linear interpolation:

_8164_start_tr_line4(), _8164_start_sr_line4()

_8164_start_ta_line4(),_8164_start_sa_line4()

(Refer to section 6.7)

The characters “t”, “s”, “r”, and “a” after **_8164_start** mean:

t – Trapezoidal profile

s – S-Curve profile

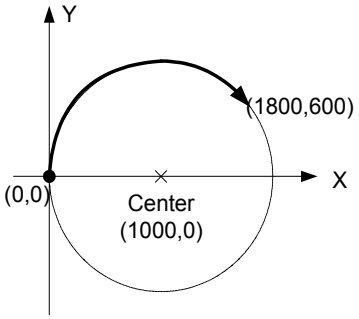
r – Relative motion

a – Absolute motion

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.6 Circular interpolation for 2 axes

Any 2 of the 4 axes of the 8164 can perform circular interpolation. In the example below, circular interpolation means XY (if axes 0, 1 are selected and assigned to be X, Y respectively) axes simultaneously start from initial point, (0,0) and stop at end point,(1800,600). The path between them is an arc, and the MaxVel is the tangential speed.



Example:

_8164_start_a_arc_xy(0 /*card No**/**, **1000,0 /***center X***/, 0 /***center Y***/**, **1800.0** /* End X **/**, **600.0 /***End Y **/** ,1000.0 /* MaxVel */)

To specify a circular interpolation path, the following parameters must be clearly defined:

Center point: The coordinate of the center of arc (In absolute mode) or
The off_set distance to the center of arc (In relative mode)

End point: The coordinate of end point of arc (In absolute mode) or
The off_set distance to center of arc (In relative mode)

Direction: The moving direction, either CW or CCW.

It is not necessary to set radius or angle of arc, since the information above gives enough constrains. The arc motion is stopped when either of the 2 axes reached end point.

There are two groups of functions that provide 2-axis circular interpolation. The first group divides the 4 axes into XY (axis 0 & axis 1) and ZU (axis 2 & axis 3). By calling these functions, the target axes are already assigned.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_start_r_arc_xy(), _8164_start_r_ arc _zu(),

_8164_start_a_ arc _xy(), _8164_start_a_ arc _zu(),

(Refer to section 6.8)

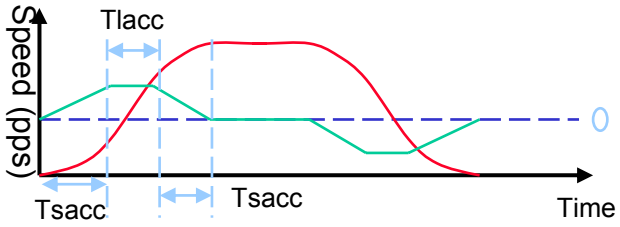
The second group allows user to freely assign any targeted 2 axes.

_8164_start_r_arc2(),_8164_start_a_arc2(): Refer to section 6.8

4.1.7 Circular interpolation with Acc/Dec time

In section 4.1.6, the circular interpolation functions do not support acceleration and deceleration parameters; therefore, they cannot perform a T or S curve speed profile during operation. However, sometimes the need for an Acc/Dec time speed profile will help a machine to make more accurate circular interpolation. The 8164 has another group of circular interpolation functions to perform this type of interpolation, but requires the use of Axis3 as an aided axis, which means that Axis3 cannot be used for other purposes while running these functions. For example, to perform a circular interpolation with a T-curve speed profile, the function _8164_start_tr_arc_xyu() is used. This function will used Axis0 and Axis1, and also Axis3 (Axis0=x, Axis1=y, Axis2=z, Axis3=u). For the full lists of functions, refer to section 6.8.

To check if the board supports these functions use the _8164_version_info() function. If hardware information for the card returns a value with the 4th digit greater then 0, for example '1003', users can use this group of circular interpolation to perform S or T-curve speed profiles. If the hardware version returns a value with the 4th digit being 0, then that board does not support these functions.



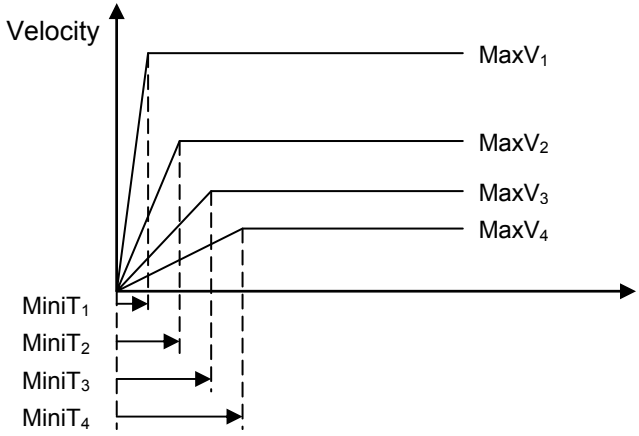
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.8 Relationship between Velocity and Acceleration Time.

The maximum velocity parameter of a motion function will eventually have a minimum acceleration value. This means that there is a range for acceleration time over one velocity value. Under this relationship, to obtain a small acceleration time, a higher maximum velocity value to match the smaller acceleration time is required. Function _8164_fix_speed_range() will provide such operation. This function will raise the maximum velocity value, which in turn results in a smaller acceleration time. Note it does not affect the actual end velocity. For example, to have a 1ms acceleration time from a velocity of 0 to 5000(pps), the function can be inserted before the motion function as shown.

_8164_fix_speed_range(AxisNo,OverVelocity);

_8164_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);



How do users decide an optimum value for "OverVelocity" in the *_8164*_fix_speed_range() function? The _8164_verify_speed() function is provided to calculate such value. The inputs to this function are the start velocity, maximum velocity and over velocity values. The output value will be the minimum and maximum values of the acceleration time.

For example, if the original acceleration range for the command is:

_8164_start_tr_move(AxisNo,5000,0,5000,0.001,0.001),

then use the following function:

_8164_verify_speed(0,5000,&minAccT, &maxAccT,5000);

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The value miniAccT will be 0.0267sec and maxAccT will be 873.587sec. This minimum acceleration time does not meet the requirement of 1mS. To achieve such a low acceleration time the over speed value must be used.

By changing the OverVelocity value to 140000,

_8164_verify_speed(0,5000,&minAccT, &maxAccT,140000);

The value miniAccT will be 0.000948sec and maxAccT will be 31.08sec. This minimum acceleration time meets the requirements. So, the motion command can be changed to:

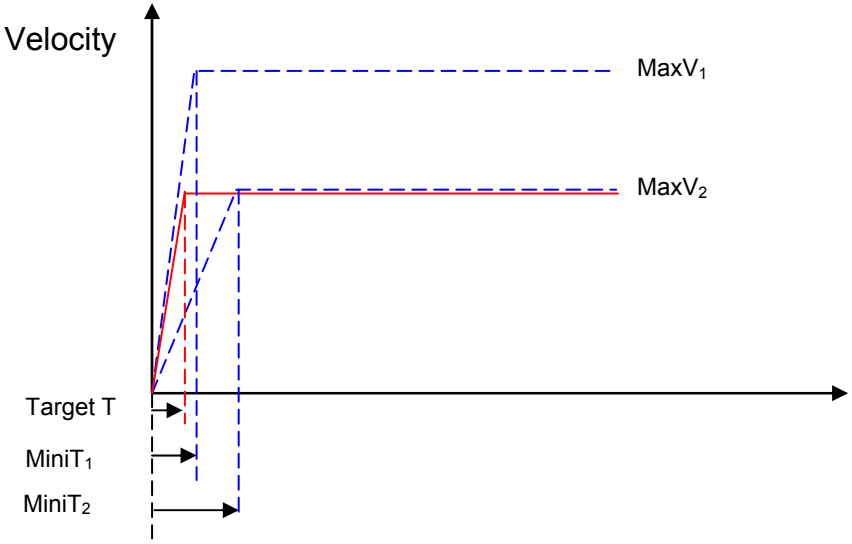
_8164_fix_speed_range(AxisNo,140000);

_8164_start_tr_move(AxisNo,5000,0,5000,0.001,0.001);

Note1: The return value of *_8164*_verify_speed() is the minimum velocity of motion command, it does not always equal to your start velocity setting. In the above example, it will be 3pps more than the 0pps setting.

Note2: To disable the fix speed function _8164_fix_speed_range() use _8164_unfix_speed_range()

[Note3] Minimize the use of the OverVelocity operation. the more it is used, the coarser the speed interval is.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Example:

User's Desired Profile: (MaxV$_2$, Target T) is not possible under MaxV$_2$ according to the (MaxV, MiniT) relationship. So one must change the (MaxV, MiniT) relationship to a higher value, (MaxV$_1$, MiniT$_1$). Finally, the command would be:

_8164_fix_speed_range(AxisNo, MaxV$_1$);

_8164_start_tr_move(AxisNo,Distance, 0 , MaxV$_2$, Target T, Target T);

Relative Functions:

_8164_fix_speed_range(),
_8164_unfix_speed_range(),_8164_verify_speed()

Refer to section 6.5

4.1.9 Continuous motion

The 8164 allows users to perform continuous motion. Both single axis movement (section 4.1.3: Trapezoidal, section 4.1.4: S-Curve) and multi-axis interpolation (4.1.5: linear interpolation, 4.1.6: circular interpolation) can be extended to be continuous motion.

For example, if a user calls the following function to perform a single axis preset motion:

_8164_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)

It will cause the axis "0" to move to position "50000.0." Before the axis arrives, the user can call a second pressed motion:

_8164_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)

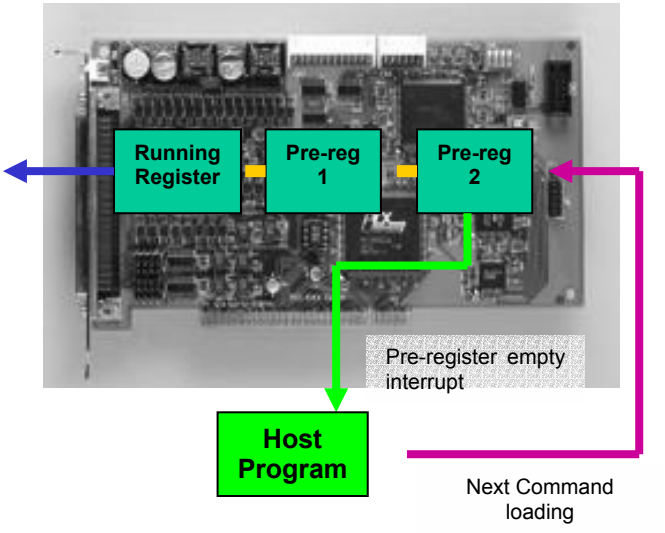
The second function call won't affect the first one. Actually, it will be executed and written into the pre-register of the 8164. After the first move is finished, the 8164 will continue with the second move according to the pre-register value. So, the time interval between these two moves can be seen as a continuous move and pulses will be continuously be generated at the "50000.0" position.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The theory of continuous motion is described below:

Theory of continuous motion

The following diagram shows the register data flow of the 8164.



Step 0: All Registers and Pre-Registers are cleared.

Step 1: The first motion is executed and the CPU writes corresponding values into Pre-Register2.

_8164_start_ta_move(0,50000.0,100.0,30000.0,0.1,0.0)

Step 2: Since Pre-Register1 & Register are empty, the data in pre-register 2 is automatically moved to the Register and executed immediately by the ASIC.

Step 3: The second function is called. The CPU writes the corresponding values into pre-register2.

_8164_start_tr_move(0,20000.0,100.0,30000.0,0.0,0.2)

Step 4: Since Pre-register1 is empty; the data in pre-register 2 is automatically moved to Pre-Rregister1 and waits to be executed.

Step 5: Now the user can execute a 3rd function, and it will be stored to Pre-register2.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Step 6: When the first function is completed, the Register becomes empty, and data in pre-register1 is allowed to move to Register and is executed immediately by the ASIC. Data in Pre-Register2 is then moved to Pre-Register1.

Step 7: The ASIC will inform the CPU generating an interrupt that a motion is completed. Users can then write the 4th motion command into Pre-Register2.

Procedures for continuous motion

The following procedures are to help user making continuous motion.

Step 1: (If Under DOS)

Enable the interrupt service using ***_8164_*int_contol()**

(If Under Windows)

Enable the interrupt service using ***_8164_*int_contol()** and ***_8164_*int_enable()**.

Step 2: Set bit "2" of INT factor to be "True" using ***_8164_*set_int_factor()/**

Step 3: Set the "conti_logic" to be "1" by: ***_8164_*set_continuous_move()** *(note: if all motions are in relative mode, this function can be ignored).*

Step 4: Call the first three motion functions.

Step 5: Wait for INT (under DOS) or EVENT (under Windows) of pre-register empty.

Step 6: Call the 4th motion function.

Step 7: Wait for INT (under DOS) or EVENT (under Windows) generated if any pre-register is empty.

Step 8: Repeat steps 6 and 7 until all functions are called.

Step n: Wait for all motions to complete.

(Note: Another method to determine a motion-completed action is by polling. User may constantly check the buffer status using the **_8164_check_continuous_buffer()** function**)**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Restrictions of continuous motion

The statements below are restrictions and suggestions for continuous motion:

1. When the Pre-Registers are full, users may not execute any more motion functions. Otherwise, the new function one will overwrite the existing function in Pre-Register2.
2. To get a continuity of velocity between 2 motions, the previous end velocity of and starting velocity of the next must be the same. There are several methods to achieve this. The easiest way is to set the deceleration/acceleration time to '0.'

For example:

1st motion: *_8164_start_tr_move_XY(*0,1000,0,0,5000,0.2, 0.0*)*

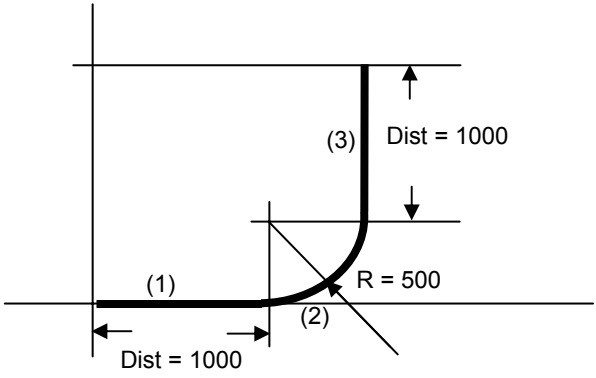
(Start a relative 2-axis linear interpolation, x distance =1000, y distance = 0, start vel = 0, max vel = 5000, Tacc = 0.2, Tdec = 0)

2nd motion: *_8164_start_r_arc_xy*(0,0,500,500,500,1,5000);

(Start a relative 2-axis circular interpolation, center x distance = 0, center y distance = 500, End x distance = 500, end y distance = 500. max vel = 5000. It is a quarter ccw circle, with velocity = 5000)

3rd motion: *_8164_start_tr_move_XY(*0,0,1000,0,5000,0.0, 0.2*)*

(Start a relative 2-axis linear interpolation, x distance=0, y distance = 1000, start vel = 0, max vel = 5000, Tacc = 0.0,Tdec = 0)



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Explanation of example:

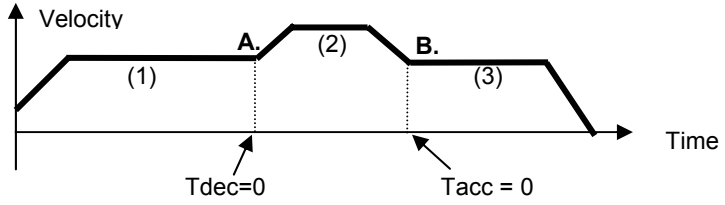
When these three motions were executed sequentially, the 1st occupies the Register and is executed immediately; the 2nd occupies Pre-Register1 and waits for completion of the 1st motion. The 3rd occupies Pre-Register2 and waits for completion of the 2nd motion. Since the 1st motion has a '0' deceleration time and the 2nd motion is an arc of constant velocity, which is the same as the max_vel of the 1st, the 8164 will output a constant frequency at intersections between them.

1. Continuous motion between different axes is meaningless. Different axes have their own register and pre-register system.
2. Continuous motion between different numbers of axes is not allowed. For example: ***_8164_start_tr_move()*** can not be followed by ***_8164_start_ta_move_XY()*** nor vice versa.
3. It is possible to perform a 3-axis or 4-axis continuous linear interpolation, but speed continuity is impossible to achieve.
4. If any absolute mode is used during continuous motion, make sure that ***_8164_reset_target_pos()*** is executed at least once after home move. Refer to 4.1.8: Home return mode for more details

Examples of continuous motion

The following are examples of continuous motion:

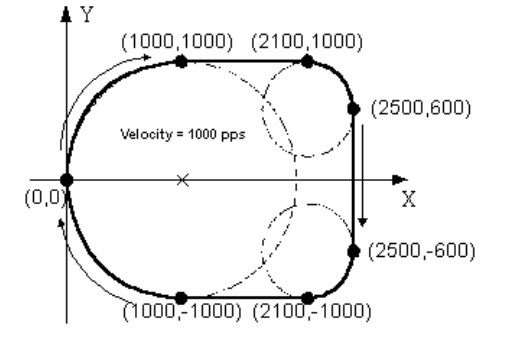
1. Single axes continuous motion: Changing velocity at preset points.



This example demonstrates how to use the continuous motion function to achieve velocity changing at pre-set points. The 1st motion (ta) moves the axis to point A, with Tdec =0, and then the 2nd continues immediately. The start velocity of (2) is the same with max velocity of (1), so that the velocity continuity exists at A. At point B. the Tacc of (3) is set to be 0, so the velocity continuity is also continued.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2. 2-axis continuous interpolation:



This example demonstrates how to use continuous motion function to achieve 2-axis continuous interpolation. In this application, the velocity continuity is the key concern. Refer to the previous example.

The functions related to continuous motion are listed below:

_8164_set_continuous_move(), _8164_check_continuous_buffer()

Refer to section 6.17.

4.1.10 Home Return Mode

In this mode, the 8164 is allowed to continuously output pulses until the condition to complete the home return is satisfied after writing the command **_8164_home_move()**. There are 13 home moving modes provided by the 8164. The "home_mode" of function **_8164_set_home_config()** is used to select whichever mode is preferred.

After completion of home move, it is necessary to keep in mind that all related position information should be reset to be "0." The 8164 has 4 counters and 1 software-maintained position recorder. They are:

Command position counter: counts the number of pulse outputs

Feedback position counter: counts the number of pulse inputs

Position error counter: counts the error between command and feedback pulse numbers.

General-Purposed counter: can be configured as pulse output, feedback pulse, manual pulse, or CLK/2.

Target position recorder: records the target position.

Refer to section 4.4 for a more detailed explanation about position counters.

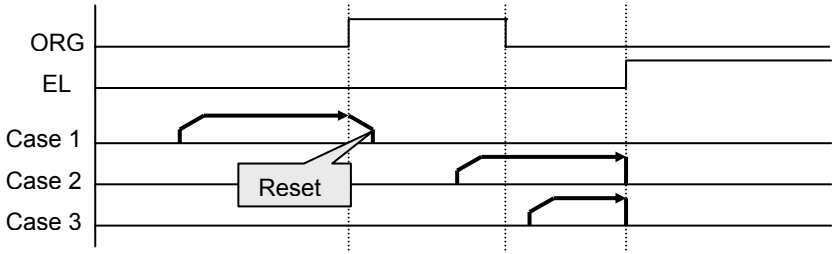
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

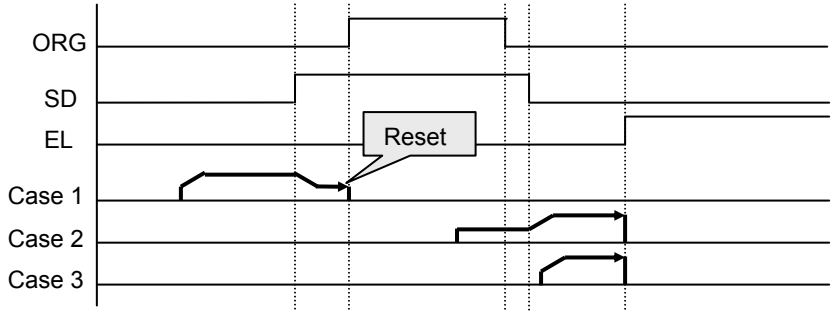
After home move is complete, the first four counters will be cleared to “0” automatically, however, the **target position** recorder will not. Because it is software maintained, it is necessary to manually set the target position to “0” by calling the function **_8164_reset_target_pos()**.

The following figures show the various home modes and the reset points, when the counter is cleared to “0.”

home_mode=0: ORG → Slow down → Stop

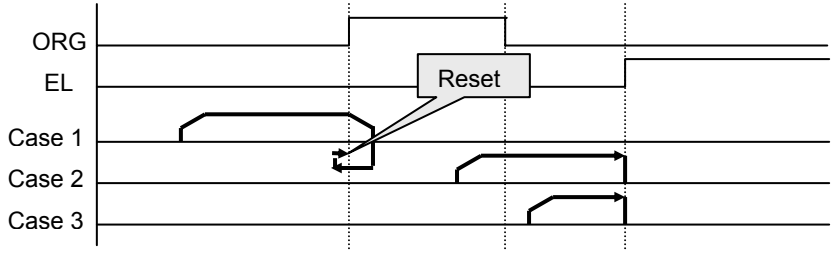
- When SD (Ramp-down signal) is inactive.





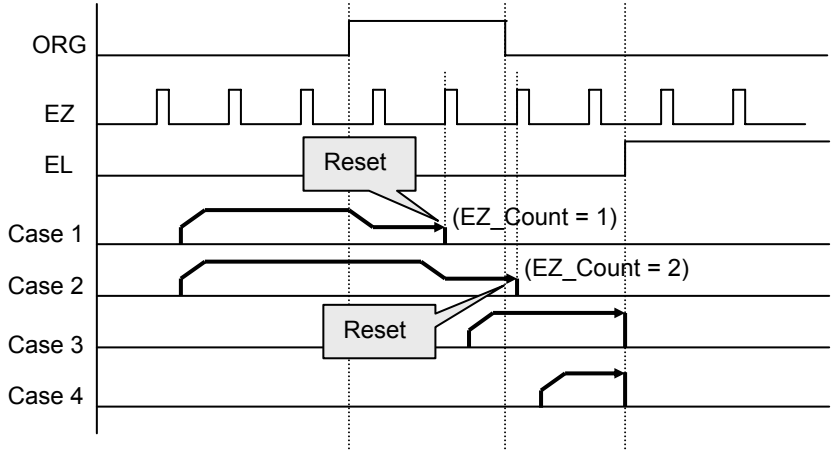
- When SD (Ramp-down signal) is active.

home_mode=1: ORG → Slow down → Stop at end of ORG

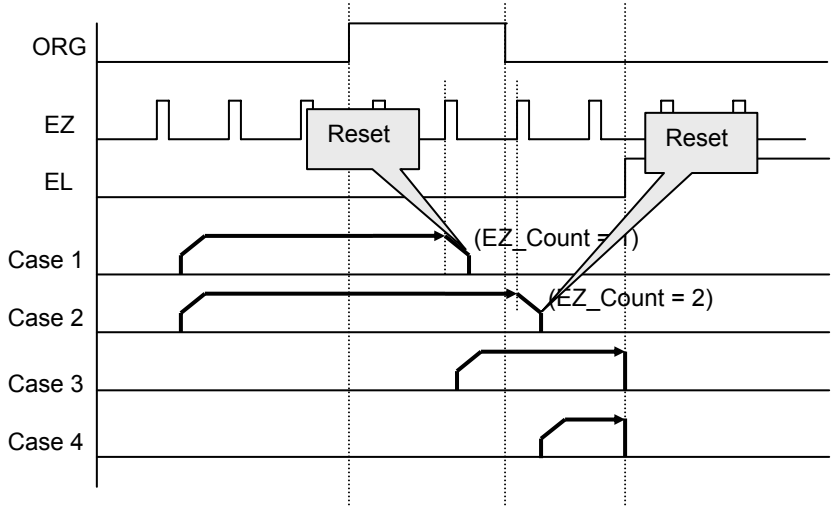


LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

home_mode=2: ORG → Slow down → Stop on EZ signal

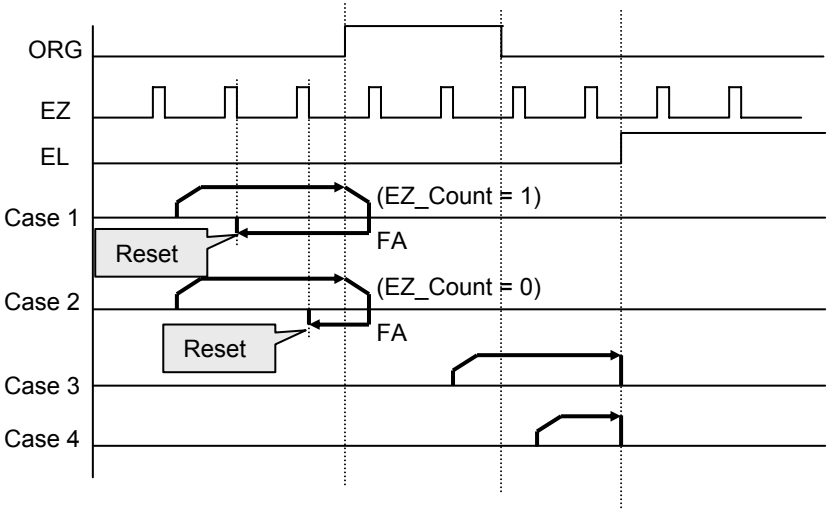


home_mode=3: ORG → EZ -→ Slow down → Stop

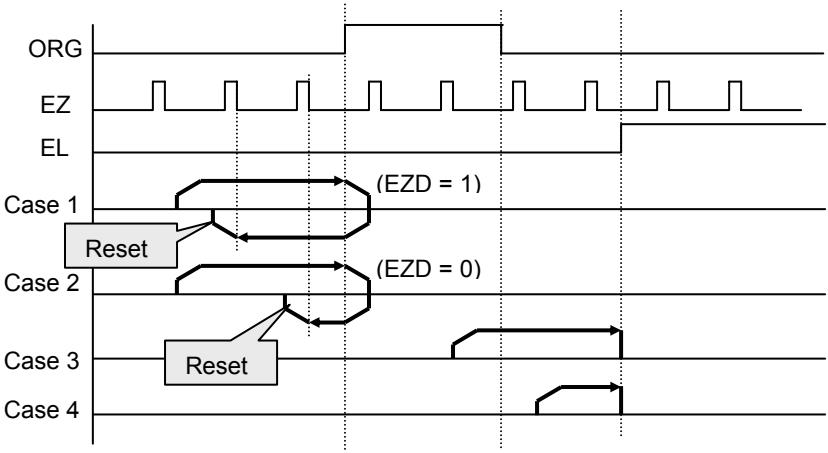


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

home_mode=4: ORG → Slow down → Go back at FA speed→ EZ→ Stop

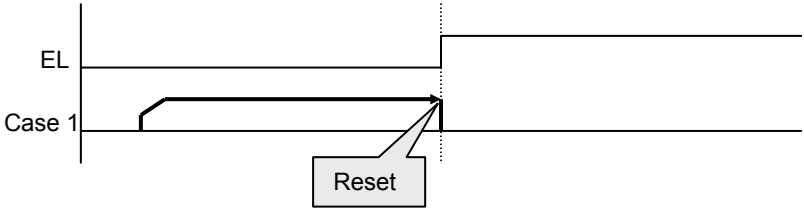


home_mode=5: ORG → Slow down → Go back → Accelerate to MaxVel → EZ → Slow down → Stop

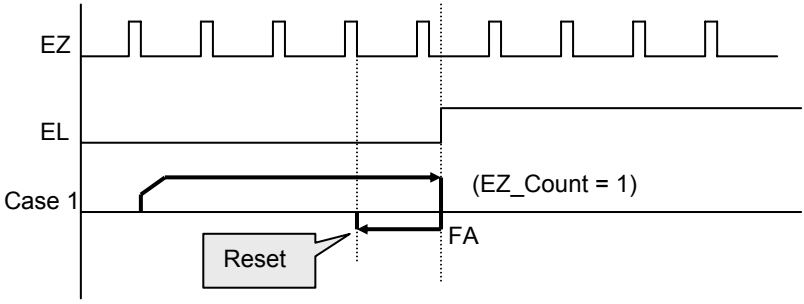


Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

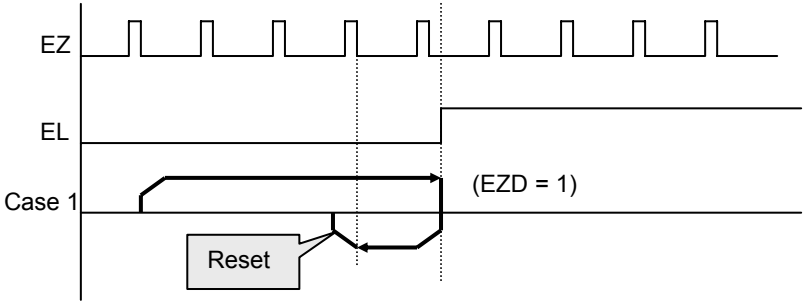
home_mode=6: *EL only*



home_mode=7: EL → Go back → Stop on EZ signal

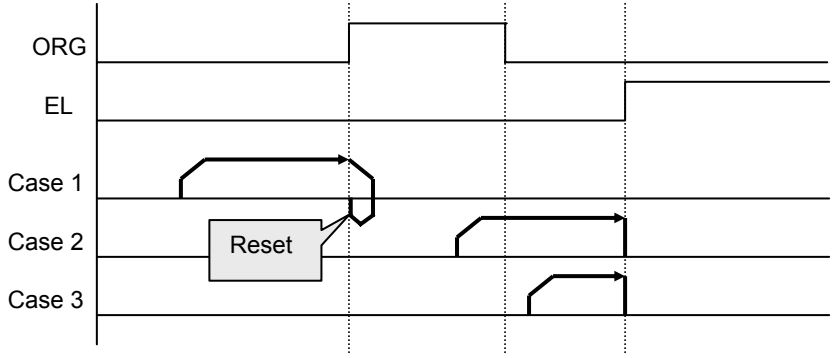


home_mode=8: EL → Go back → Accelerate to MaxVel →EZ → Slow down → Stop

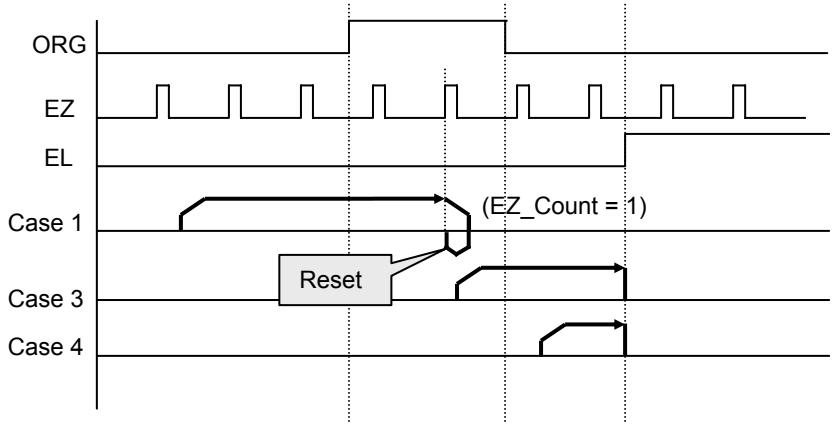


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

home_mode=9: ORG → Slow down → Go back → Stop at beginning edge of ORG

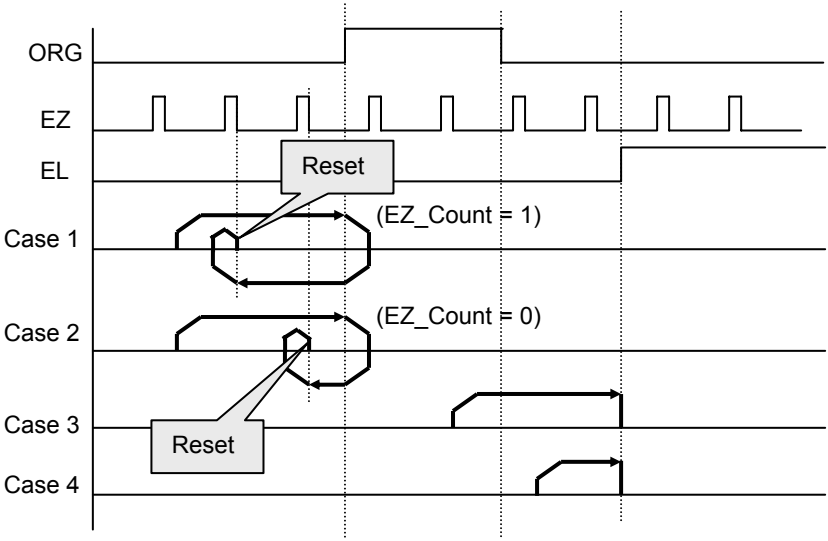


home_mode=10: ORG → EZ →Slow down → Go back → Stop at beginning edge of EZ

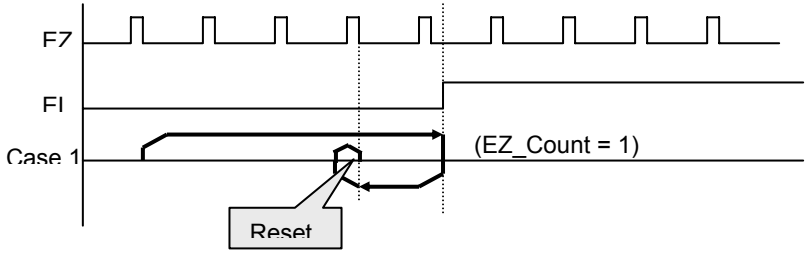


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

home_mode=11: ORG → Slow down → Go back (backward)→ Accelerate to MaxVel → EZ → Slow down → Go back again (forward)→ Stop at beginning edge of EZ

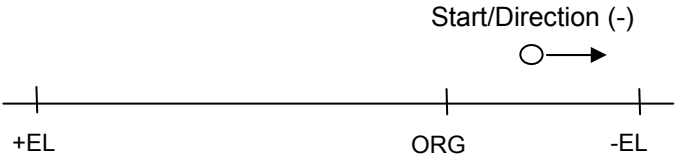


home_mode=12: EL → Stop → Go back (backward)→ Accelerate to MaxVel → EZ → Slow down → Go back again (forward)→ Stop at beginning edge of EZ

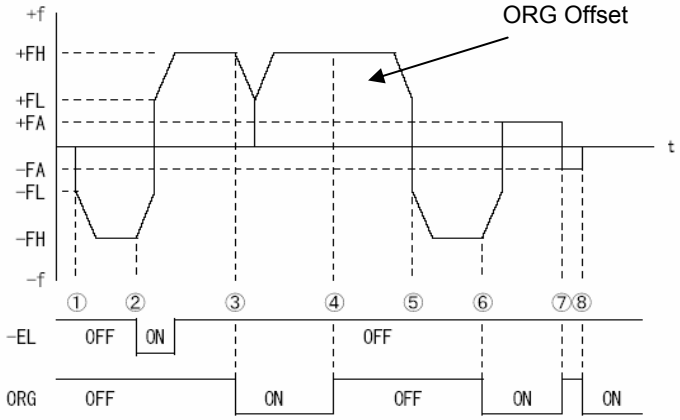


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

- **Home Search Example (Home mode=1)**



FL=Start Velocity
FH=Max Velocity (the sign stands for direction)
FA=Search Speed (half of the FH)



Moving Steps

1. Home searching start (-)
2. –EL touches, slow down and reverse moving (+)
3. ORG touches, slow down
4. Escape from ORG according to ORG offset
5. Start searching again (-)
6. ORG touches, slow down then using searching speed to escape ORG (+)
7. After escape ORG, search ORG with search speed again (-)

Relative Functions:

_8164_set_home_config(), _8164_home_move(), _8164_home_search(), Refer to section 6.9.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.1.11 Manual Pulse Mode (PCI-8164 Only)

For manual operation of a device, you may use a manual pulse such as a rotary encoder. The PCI-8164 can receive input signals from a pulser and output its corresponding pulses from the OUT and DIR pins, thereby allowing a simplified external circuit.

This mode is effective when the **_8164_pulser_vmove(), _8164_pulser_pmove(),** or **_8164_pulser_home_move()** command has been called. To terminate the command use the **_8164_sd_stop()** or**_8164_emg_stop()** command.

The PCI-8164 receives positive and negative pulses (CW/CCW) or 90 degrees phase difference signals (AB phase) from the pulser at the PA and PB pins. To set the input signal modes of the pulser, use the **_8164_set_pulser_iptmode()** function. The 90° phase difference in signals can be set by a multiplication of 1, 2, or 4. If the AB phase input mode is selected, PA and PB signals should have a 90°-phase shift, and the position counter increases when the PA signal is leading the PB signal by 90°.

Relative Functions:

_8164_pulser_vmove(), _8164_pulser_pmove(), _8164_pulser_home_move(), _8164_set_pulser_iptmode()

Refer to section 6.10

4.2 The motor driver interface

The 8164 provides the INP, ALM, ERC, SVON, and RDY signals for a servomotor driver control interface. The INP and ALM are used for feedback of the servo driver status, ERC is used to reset the servo driver's deviation counter under special conditions, VON is a general purpose output signal, and RDY is a general purpose input signal. The meaning of "general purpose" is that the processing of the signal is not a build-in procedure of the hardware. The hardware processes INP, ALM, and ERC signals according to pre-defined rules. For example, when receiving ALM signal, the 8164 stops or decelerate to stop output pulses automatically. However, SVON and RDY are not the case, they actually act like common I/O's.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.1 INP

The processing of the INP signal is a hardware build-in procedure, and it is designed to cooperate with the in-position signal of the servomotor driver.

Usually, servomotor drivers with a pulse train input has a deviation (position error) counter to detect the deviations between the input pulse command and feedback counter. The driver controls the motion of the servomotor to minimize the deviation until it becomes 0. Theoretically, the servomotor operates with some time delay from the command pulses. Likewise, when the pulse generator stops outputting pulses, the servomotor does not stop immediately but keeps running until the deviation counter is zero. Only after stopping does the servo driver send out the in-position signal (INP) to the pulse generator to indicate the motor has stopped running.

Normally the 8164 stops outputting pulses upon completion of outputting designated pulses. However, by setting parameter ***inp_enable*** with the ***_8164_set_inp()*** function, the delay in completion of the motion to the time the INP signal is issued can be adjusted, i.e., the motor arrives at the target position. Status of **_8164_motion_done()** and INT signal are also delayed. That is, when performing under position control mode, the completion of **_8164_start_ta_move()**, **_8164_start_sr_move()**, etc, is delayed until the INP signal is issued.

The in-position function can be enabled or disabled, and the input logic polarity is also programmable by the "**inp_logic"** parameter of ***_8164_set_inp()***. The INP signal status can be monitored by software with the function: **_8164_get_io_status()**.

Relative Functions:

_8164_set_inp(): Refer to section 6.12

_8164_get_io_status(): Refer to section 6.13

_8164_motion_done(): Refer to section 6.11

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.2 ALM

The processing of the ALM signal is a hardware build-in procedure, and it is designed to interact with the alarm signal of the servomotor driver.

The ALM signal is an output signal from servomotor driver. Usually, it is designated to indicate when something is wrong with the driver or motor.

The ALM pin receives the alarm signal output from the servo driver. The signal immediately stops the 8164 from generating any further pulses or stops it after deceleration. If the ALM signal is in the ON status at the start of an operation, the 8164 will generate the INT signal and thus not generate any command pulses. The ALM signal may be a pulse signal with a minimum time width of 5 microseconds.

Setting the parameters “alm_logic” and “alm_mode” of the **_8164_set_alm** function can alter the input logic of the ALM. Whether or not the 8164 is generating pulses, the ALM signal allows the generation of the INT signal. The ALM status can be monitored by using the software function: ***_8164_get_io_status()***. The ALM signal can generate an IRQ, if the interrupt service is enabled. Refer to section 4.7.

Relative Functions:

_8164_set_alm(): Refer to section 6.12

_8164_get_io_status(): Refer to section 6.13

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.2.3 ERC

The ERC signal is an output from the 8164. The processing of the ERC signal is a hardware build-in procedure, and it is designed to interact with the deviation counter clear signal of the servomotor driver.

The deviation counter clear signal is inserted in the following 4 situations:

1. Home return is complete
2. The end-limit switch is active
3. An alarm signal stops the OUT and DIR signals
4. The software operator issues an emergency stop command

Since the servomotor operates with some delay from the pulse generated from the 8164, it continues to move until the deviation counter of the driver is zero even if the 8164 has stopped outputting pulses because of the ±EL signal or the completion of home return. The ERC signal allows immediate stopping of the servomotor by resetting the deviation counter to zero. The ERC signal is outputted as a one-shot signal. The pulse width is of time length defined by the function call ***_8164_set_erc()***. The ERC signal will automatically be generated when the ±EL and ALM signal are turned on and the servomotor is stopped immediately.

Relative Functions:

_8164_set_erc(): Refer to section 6.12

4.2.4 SVON and RDY

All axes of the 8164 are equipped with SVON and RDY signals, which are general purpose output and input channels, respectively. Usually, the SVON is used to interact with the servomotor drivers as a Servo ON command, and RDY to receive the Servo Ready signal. There are no built-in procedures for SVON and RDY.

The SVON signal is controlled by the software function **_8164_Set_Servo()**.

RDY pins are dedicated for digital input usage. The status of this signal can be monitored using the software function **_8164_get_io_status()**.

Relative Functions:

_8164_Set_Servo(): Refer to section 6.12

_8164_get_io_status(): Refer to section 6.13

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.3 The limit switch interface and I/O status

In this section, the following I/O signal operations are described.

- SD/PCS: Ramping Down & Position Change sensor
- ±EL: End-limit sensor
- ORG: Origin position

In any operation mode, if an ±EL signal is active during any moving condition, it will cause the 8164 to stop automatically outputting pulses. If an SD signal is active during moving conditions, it will cause the 8164 to decelerate. If operating in a multi-axis mode, it automatically applies to all related axes.

4.3.1 SD/PCS

SD/PCS signal pins are available for each axis and acts as the input channel. It can be connected to a SD (Slow Down) or Position Change Signal (PCS). To configure the input signal type use the function **_8164_set_sd_pin()**.

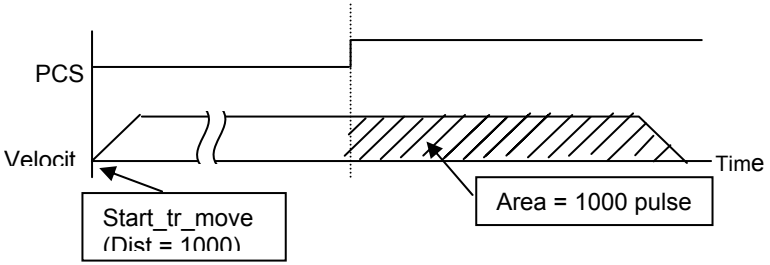
When the SD/PCS pin is directed to a SD (the default setting), the PCS signal is kept at a low level and visa versa. Care must be taken with the logic attributes of the signal not being used.

The slow-down signals are used to force the output pulse (OUT and DIR) to decelerate to and then maintain the StrVel when it is active. The StrVel is usually smaller than MaxVel. This signal is useful in protecting a mechanism moving under high speeds toward the mechanism's limit. SD signal is effective for both plus and minus directions.

The ramping-down function can be enabled or disabled using the software function **_8164_set_sd()**. The input logic polarity, level operation mode, or latched input mode can also be set by this function. The signal status can be monitored using **_8164_get_io_status()**.

The PCS signal is used to define the starting point of current tr and sr motions. Refer to the chart below. The logic of PCS is configurable using **_*8164*_set_pcs_logic()**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Relative Functions:

_8164_set_sd_pin(),_8164_set_pcs_logic(): Refer to section 6.5

_8164_set_sd(): Refer to section 6.12

_8164_get_io_status(): Refer to section 6.13

4.3.2 EL

The end-limit signal is used to stop the control output signals (OUT and DIR) when the end-limit is active. There are two possible stop modes, “stop immediately” and “decelerate to StrVel then stop.” To select either mode use **_8164_set_el().**

The PEL signal indicates the end-limit in the positive (plus) direction. MEL signal indicates the end-limit in negative (minus) direction. When the output pulse signals (OUT and DIR) is towards the positive direction, the pulse train will be immediately stopped when the PEL signal is asserted, where the MEL signal is meaningless, and vise versa. When the PEL is asserted, only a negative (minus) direction output pulse can be generated when moving the motor in a negative (minus) direction.

The EL signal can generate an IRQ if the interrupt service is enabled. Refer to section 4.7.

You can either use ‘A’ or ‘B’ type contact switches by setting the S1 dipswitch. The 8164 is delivered from the factory with all bits of S1 set to ON. The signal status can be monitored using the software function **_8164_get_io_status().**

Relative Functions:

_8164_set_el(): Refer to section 6.12

_8164_get_io_status(): Refer to section 6.13

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.3.3 ORG

The ORG signal is used when the motion controller is operating in the home return mode. There are 13 home return modes (Refer to section 4.1.8), any one of 13 modes cam be selected using "*home_mode*" argument in the function **_8164_set_home_config()**. The logic polarity of the ORG signal level or latched input mode is also selectable using this function as well.

After setting the configuration for the home return mode with **_8164_set_home_config()**, the **_8164_home_move()** command can perform the home return function.

Relative Functions:

_8164_set_home_config(), _8164_home_move(): Refer to section 6.19

4.4 The Counters

There are four counters for each axis of the 8164. They are described in this section:

Command position counter: counts the number of output pulses

Feedback position counter: counts the number of input pulses

Position error counter: counts the error between command and feedback pulse numbers.

General purpose counter: The source can be configured as pulse output, feedback pulse, manual pulse, or CLK/2.

Also, the **target position recorder**, a software-maintained position recorder, is discussed.

4.4.1 Command position counter

The command position counter is a 28-bit binary up/down counter. its input source is the output pulse from the 8164, thus, it provides accurate information of the current position. Note: the command position is different from target position. The **command position** increases or decreases according to the pulse output, while the **target position** changes only when a new motion command has been executed. The target position is recorded by the software, and needs manually resetting after a home move is completed.

The command position counter will clear (reset to "0") automatically after a home move has completed. The function **_*8164*_set_command()** can be executed at any time to set a new command position value. To read current command position use **_*8164*_get_command().**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Relative Functions:

_8164_set_command(), _8164_get_command(): Refer to section 6.15

4.4.2 Feedback position counter

The 8164 has a 28-bit binary up/down counter managing the present position feedback for each axis. The counter counts signal inputs from the EA and EB pins.

It accepts 2 kinds of pulse inputs: (1). Plus and minus pulse inputs (CW/CCW mode). (2). 90° phase shifted signals (AB phase mode). 90° phase shifted signals maybe multiplied by a factor of 1, 2 or 4. 4x AB phase mode is the most commonly used in incremental encoder inputs. For example, if a rotary encoder has 2000 pulses per phase (A or B phase), then the value read from the counter will be 8000 pulses per turn or –8000 pulses per turn depending on its rotating direction. These input modes can be selected using the **_8164_*set_pls_iptmode()*** function.

In cases where the application has not implemented an encoder, it is possible to set the feedback counter source to generate the output pulses, just as with the command counter. Thus, the feedback counter and the command counter will have the same value. To enable the counters to count the number of pulses inputted, set the "**Src**" parameter of the software function ***_8164_*set_feedback_src*()*** to "1."

Plus and Minus Pulses Input Mode (CW/CCW Mode)

The pattern of pulses in this mode is the same as the ***Dual Pulse Output Mode*** in the Pulse Command Output section; except that the input pins are EA and EB.

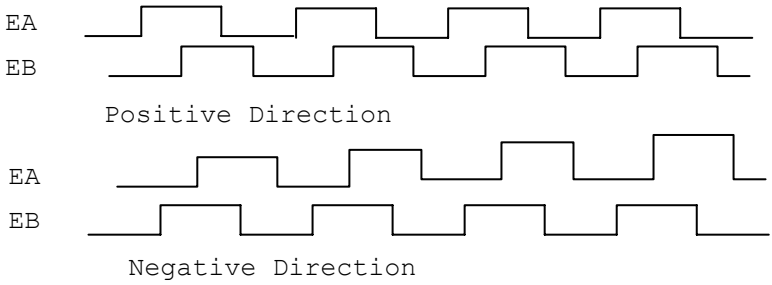
In this mode, pulses from EA cause the counter to count up, whereas EB caused the counter to count down.

90°phase difference signals Input Mode (AB phase Mode)

In this mode, the EA signal is a 90° phase leading or lagging in comparison with the EB signal. "Lead" or "lag" of phase difference between two signals is caused by the turning direction of the motor. The up/down counter counts up when the phase of EA signal leads the phase of EB signal.

The following diagram shows the waveform.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



The index input (EZ) signals of the encoders are used as the “ZERO” reference. This signal is common on most rotational motors. EZ can be used to define the absolute position of the mechanism. The input logic polarity of the EZ signals is programmable using software function **_8164_*set_home_config().*** The EZ signals status of the four axes can be monitored by ***get_io_status()***.

The feedback position counter will be automatically cleared to “0” after a home move is complete. Besides setting a position with the function call, ***_8164_*set_position()**, it can also be executed at any time to set a new position value. To read the current command position use ***_8164_*get_position().**

Relative Function:

_8164_set_pls_iptmode(), _8164_set_feedback_src() :Refer to section 6.4

_8164_set_position(), _8164_get_position(): Refer to section 6.15

_8164_set_home_config(): Refer to section 6.9

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.4.3 Position error counter

The position error counter is used to calculate the error between the command position and the feedback position. It will add one count when the 8164 outputs one pulse and subtracts one count when the 8164 receives one pulse (from EA, EB). It is useful in detecting step-loses (stalls) in situations of a stepping motor when an encoder is applied.

Since the position error counter automatically calculates the difference between pulses outputted and pulses fed back, it is inevitable to get an error if the motion ratio is not equal to "1."

To obtain a position error reading, use the function call **_8164_get_error_counter().** To reset the position error counter, use the function call **_8164_reset_error_counter().** The position error counter will automatically clear to "0" after home move is complete.

Relative Function:

_8164_get_error_counter(), _8164_reset_error_counter(): Refer to section 6.15

4.4.4 General purpose counter

The general purpose counter is very versatile. It can be any of the following:

1. Pulse output – as a command position counter
2. Pulse input – as a feedback position counter
3. Manual Pulse input – Default status.
4. Clock – an accurate timer (9.8 MHz)

The default setting of the general purpose counter is set to manual pulse. (Refer to section 4.1.9 for a detailed explanation of manual pulsing). To change the source type, use the function **_8164_set_general_counter().** To obtain the counter status, use the function **_8164_get_general_counter().**

Relative Function:

_8164_set_general_counter(), _8164_get_general_counter(): Refer to section 6.15

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Table below summarizes all functions used for the different counter types

Counter	Description	Counter Source	Function	Function description
Command	Counts the number of output pulses	Pulse output	*_8164*_set_command	Set a new value for command position
			*_8164*_get_command	Read current command position
Feedback	counts the number of input pulses	EA/EB or Pulse output	_8164_set_pls_iptmode	Select the input modes of EA/EB
			_8164_set_feedback_src	Set the counters input source
			*_8164*_set_position	Set a new value for feedback position
			*_8164*_get_position	Read current feedback position:
Position error	Counts the error between command and feedback pulse	EA/EB and Pulse output	_8164_get_error_counter	Gets the position error
			_8164_reset_error_counter	Resets the position error counter
General Purpose	General Purpose counter	Pulse output EA/EB manual pulse CLK/2.	*_8164*_set_general_counter	Set a new counter value
			*_8164*_get_general_counter	Read current counter value

4.4.5 Target position recorder

The target position recorder is used for providing target position information. For example, if the 8164 is operating in continuous motion with absolute mode, the target position lets the next absolute motion know the target position of previous one.

It is very important to understand how the software handles the target position recorder. Every time a new motion command is executed, the displacement is automatically added to the target position recorder. To ensure the correctness of the target position recorder, users need to manually maintain it in the following two situations using the function **_8164_reset_target_pos()**:

1. After a home move completes
2. After a new feedback position is set

Relative Functions:

_8164_reset_target_pos(): Refer to section 6.15

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.5 Multiple PCI-8164 Card Operation (PCI-8164 Only)

The software function library can support a maximum of 12 PCI-8164 cards. This means up to 48 motors can be connected. Since the PCI-8164 is Plug-and-Play compatible, the base address and IRQ settings for card are are automatically assigned by the BIOS of the system when it is powered on. The base address and IRQ settings assigned by the BIOS can view by using the Motion Creator Tool.

When multiple cards are applied to a system, each card number must be noted. The card number of a PCI-8164 depends on the location on the PCI slot. They are numbered either from left to right or right to left on the PCI slots. These card numbers will affect its corresponding axis number. Note that the axis number is the first argument for most functions called in the library. Hence, it is important to identify the slot number before writing any application programs. For example, if three PCI-8164 cards are plugged in to PCI slots, then the corresponding axis number on each card will be:

Axis No. / Card No.	Axis 1	Axis 2	Axis 3	Axis 4
1	0	1	2	3
2	4	5	6	7
3	8	9	10	11

Example: To accelerate Axis 3 of Card2 from 0 to 10000pps in 0.5sec for Constant Velocity Mode operation, the axis number is 6, and the code for the program will be:

_8164_start_tv_move(6, 0, 10000, 0.5);

To determine the right card number, trial and error may be necessary before an application. The Motion Creator utility can be utilized to minimize the search time.

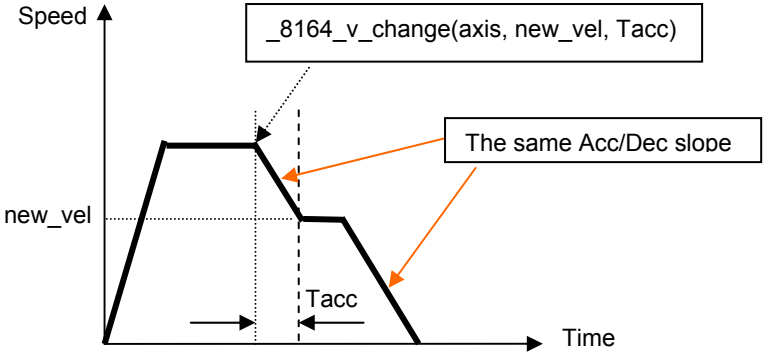
For applications requiring many axes to move simultaneously on multiple PCI_8164 cards, connection diagrams in Section 3.12 should be followed to connect between CN4 connectors. Several functions listed in Section 6.8 may be useful when writing programs for such applications.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.6 Change position or speed on the fly

The 8164 provides the ability to change position or speed while an axis is moving. Changing speed/position on the fly means that the target speed/position can be altered after the motion has started. However, certain limitations do exist. Carefully study all constraints before implementing the on-the-fly function.

4.6.1 Change speed on the fly



The change speed on the fly function is applicable on single axis motion only. Both velocity mode motion and position mode motion are acceptable. The graph above shows the basic operating theory.

The following functions are related to changing speed on the fly.

_8164_v_change() – change the MaxVel on the fly

_8164_cmp_v_change() –change velocity when the general comparator comes into existence

_8164_sd_stop() – slow down to stop

_8164_emg_stop() – immediately stop

_8164_fix_speed_range() – define the speed range

_8164_unfix_speed_range() – release the speed range constrain

The first 4 functions can be used for changing speed during a single axis motion. Functions **_8164_sd_stop()** and **_8164_emg_stop()** are used to decelerate the axis speed to "0." **_8164_fix_speed_range()** is necessary before any **_8164_v_change()** function, and **_8164_unfix_speed_range()** releases the speed range constrained by **_8164_fix_speed_range()**.

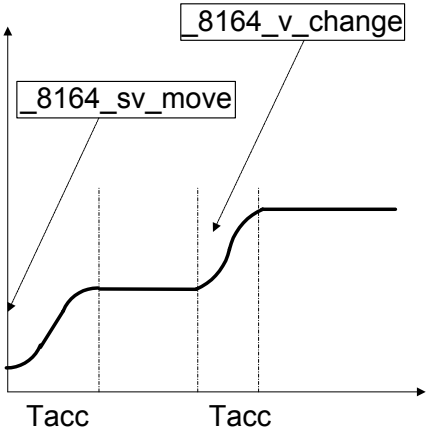
LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The function **_8164_cmp_v_change()** almost has the same function as **_8164_v_change()**, except **_8164_cmp_v_change()** acts only when a general comparator comes into existence. Refer to section 4.4.4 for more details about the general comparator.

The last 4 functions are relatively easy to understand and use. So, the discussion below will be focused on **_8164_v_change()**.

Theory of _8164_v_change():

The _8164_v_change() function is used to change MaxVel on the fly. In a normal motion operation, the axis starts at StrVel speed, accelerates to MaxVel, and then maintains MaxVel until it enters the deceleration region. If MaxVel is change during this time, it will force the axis to accelerate or decelerate to a new MaxVel in the time period defined by the user. Both Trapezoidal and S-curve profiles are applicable. The speed changes at a constant acceleration for a Trapezoidal and constant jerk for a S-curve profile.



Constraints of _8164_v_change():

In a single axis preset mode, there must be enough remaining pulses to reach the new velocity, else the _8164_v_change() will return an error and the velocity remains unchanged.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

For example:

A trapezoidal relative motion is applied:

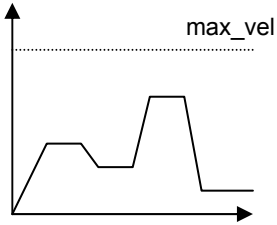
_8164_start_tr_move(0,10000,0,1000,0.1,0.1).

It cause axis 0 to move for 10000 pulses, and the maximum velocity is 1000 PPS.

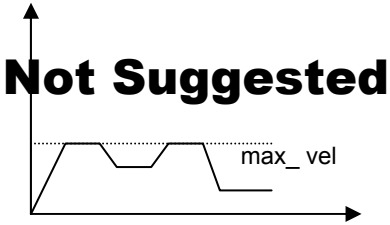
At 5000 pulses, _8164_v_change(0,NewVel,Tacc) is applied.

NewVel (PPS)	Tacc (Sec)	Necessary remaining pulses			OK / Error
		Acceleration	Deceleration	Total	
5000	0.1	300	313	613	OK
5000	1	3000	3125	6125	Error
10000	0.1	550	556	1106	OK
50000	0.1	2550	2551	5101	Error

1. To set the maximum velocity, the function **_8164_fix_speed_range()** must be used in order for the function **_8164_v_change()** to work correctly. If **_8164_fix_speed_range()** is not applied, MaxVel set by **_8164_v_move()** or **_8164_start_ta_move()** automatically becomes the maximum velocity, where **_8164_v_change()** can not be exceeded.

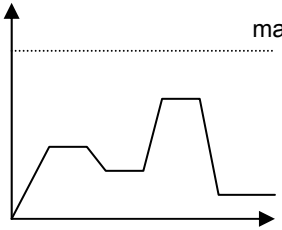


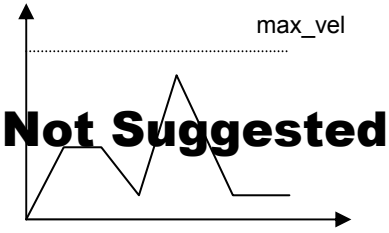
With **fix_speed_range**



Without **fix_speed_range**

2. During the acceleration or deceleration period, using **_8164_v_change()** is not suggested, although it does work in most cases, the acceleration and deceleration time is not guaranteed.

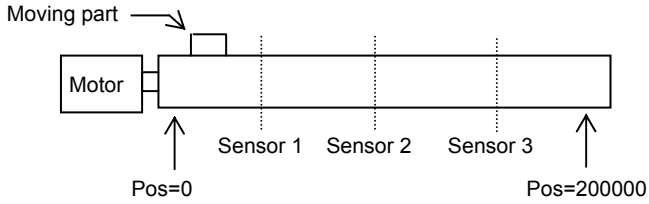




LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Example:

There are 3 speed change sensors during an absolute move for 200000 pulses. Initial maximum speed is 10000pps. Change to 25000pps if Sensor 1 is touched. Change to 50000pps if Sensor 2 is touched. Change to 100000pps if Sensor 3 is touched. Then the code for this application and the resulting velocity profiles are shown below.

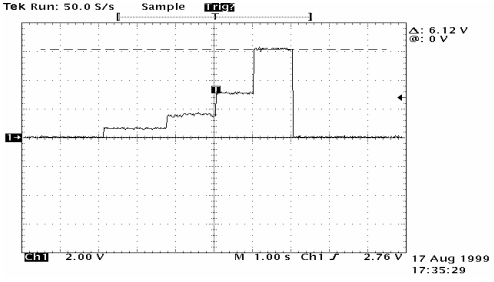



```
#include “pci_8164.h”

_8164_fix_speed_range(axis, 100000.0);
_8164_start_ta_move(axis, 200000.0, 1000, 10000, 0.02,0.01);
while(!_8164_motion_done(axis))
  {
// Get sensor’s information from another I/O card

if((Sensor1==High) && (Sensor2==Low) && (Sensor3 == Low))
_8164_v_change(axis, 25000, 0.02);
else if((Sensor1==Low) && (Sensor2==High) && (Sensor3 == Low))
_8164_v_change(axis, 50000, 0.02);
else if((Sensor1==Low) && (Sensor2==Low) && (Sensor3 == High))
_8164_v_change(axis, 100000, 0.02);
  }
```

The information of the three sensors is acquired from another I/O card, and the resulting velocity profile from experiment is shown below:



LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Relative Function:

_8164_v_change(), _8164_sd_stop(), _8164_emg_stop()

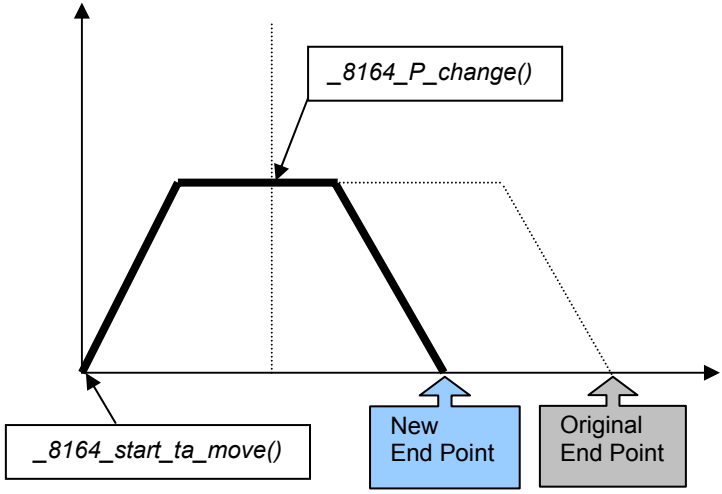
_8164_fix_speed_range(),_8164_unfix_speed_range()

_8164_get_currebt_speed()

Refer to section 6.5

4.6.2 Change position on the fly

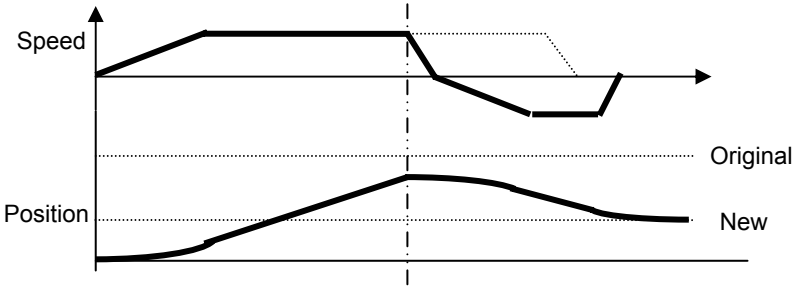
When operating in single-axis absolute pre-set motion, it is possible to change the target position during moving by using the function **_8164_p_change()**.



Theory of _8164_p_change():

The _8164_p_change() is applicable to the _8164_start_ta_move(), and _8164_start_sa_move() functions only. It is used to change the target position, defined originally by these two functions. After changing position, the axis will move to the new target position and totally disregard the original position. If the new position is in the passed path, it will cause the axis to decelerate and eventually stop, then reverse, as shown in the chart. The acceleration and deceleration rate, and StrVel and MaxVel are kept the same as the original setting.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Constraints of _8164_p_change():

1. **_8164_p_change()** is only applicable on single-axis absolute pre-set motion, i.e., **_8164_start_ta_move()**, and **_8164_start_sa_move()** only.
2. Position change during the deceleration period is not allowed.
3. There must be enough distance between the new target position and current position where **_8164_p_change()** is executed because the 8164 needs enough space to finish deceleration.

For example:

A trapezoidal absolute motion is applied:

*_8164_start_ta_move(0,**10000**,0,1000,0.5,1).*

It cause axis 0 to move to pulse 10000 position with a maximum velocity of 1000 PPS. The necessary number of pulses to decelerate is 0.5*1000*1 = 500.

At position "CurrentPos," _8164_p_change(0, NewPos) is applied.

NewPos	CurrentPos	OK / Error	Note
5000	4000	OK	
5000	4501	Error	
5000	5000	Error	
5000	5499	Error	
5000	6000	OK	Go back
5000	9499	OK	Go back
5000	9500	Error	
5000	9999	Error	

Relative Function:

_8164_p_change(): refer to section 6.6

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.7 Position compare and Latch

The 8164 provides position comparison functions on axes 0 and 1, and position latching functions on axes 2 and 3. The comparison function is used to output a trigger pulse when the counter reaches a preset value set by the user. CMP1 (axis 0) and CMP2 (axis 1) are used as a comparison trigger. The latch function is used to capture values on all 4 counters (refer to section 4.4) at the instant the latch signal is activated. LTC3 (axis 2) and LTC4 (axis 3) are used to receive latch pulses.

4.7.1 Comparators of the 8164

There are 5 comparators for each axis of the 8164. Each comparator has its unique functionality. Below is a table for comparison:

	Compare Source	Description	Function Related
Comparator 1	Command position counter	Soft Limit (+) **(Refer to section 4.9)**	*_8164_set_*softlimit *_8164_*enable_softlimit *_8164_*diable_softlimit
Comparator 2	Command position counter	Soft Limit (-) **(Refer to section 4.9)**	
Comparator 3	Position error counter	Step-losing detection	*_8164_*error_counter_*check*
Comparator 4	Any counters	General- purposed	*_8164_*set_general_*comparator*
Comparator 5 (Only Axis 0 & 1)	Feedback position counter	Position compare function (Trigger)	*_8164_*set_trigger_comparator *_8164_*build_compare_function *_8164_*build_compare_table *_8164_*set_auto_compare

Note: Only comparator 5 has the ability to trigger an output pulse via the CMP.

Comparators 1 and 2 are used for soft limits. Refer to section 4.9. Comparator 3 is used to compare with the position error counter. It is useful for detecting if a stepping motor has lost any pulses. To enable/disable the step-losing detection, or set the allowable tolerance use ***_8164_set_error_counter_check()***

The 8164 will generate an interrupt if step-losing is enabled and has occurred.

Comparator 4 is a general purpose comparator, which will generate an interrupt (default reaction) if the comparing condition comes into existence. The comparing source counter can be any counter. The compared value, source counter, comparing method, and reaction are set by the function ***_8164_*set_general_*comparator()***.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4.7.2 Position compare

The 5th comparator, whose comparing source is the feedback position counter, performs the position compare function. Only the first 2 axes (0 and 1) can do a position comparison. The position comparison function triggers a pulse output via the CMP, when the comparing condition comes into existence.

The comparing condition consists of 2 parts, the first is the value to be compared, and the second is the comparing mode. Comparing mode can be “>“, “=“ or “<“. The easiest way to use the position comparison function is to call the function:

_8164_set_trigger_comparator (AxisNo, CmpSrc, Method, Data)

The second parameter, “Method,” indicates the comparing method, while the third parameter, “Data,” is for the value to be compared. In continuous comparison, this data will be ignored automatically since the compare data is built by other functions.

Continuously comparison with trigger output

To compare multiple data continuously, functions for building comparison tables are provided and are shown below:

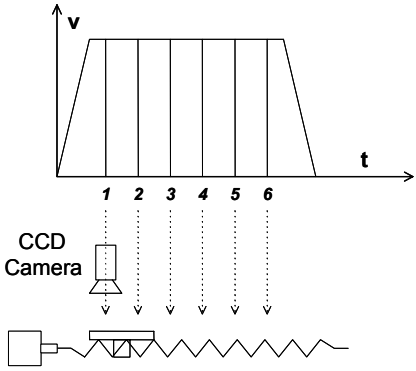
1. *_8164_build_comp_function(AxisNo, Start, End, Interval)*
2. *_8164_build_comp_table(AxisNo, tableArray, Size)*
3. *_8164_set_auto_compare(AxisNo, SelectSource)*

Note 1. Please turn off all interrupt function when these functions are running.

The first function builds a comparison list using start and end points and constant intervals. The second function builds on an arbitrary comparison table (data array). The third function is a source comparing selection function. Set this parameter to “1” to use the FIFO mode. Once it is set, the compare mechanism will start. Users can check current values used for comparison using the function**_8164_check_compare_data():**

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Example: Using the continuous position comparison function.



Trigger Output

In this application, the table is controlled by the motion command, and the CCD Camera is controlled by the position comparison output of the 8164. An image of the moving object is easily obtained.

Working Spec: 34000 triggering points per stroke, trigger speed is 6000 pts/sec)

Program Settings:

- Table starts moving from 0 to 36000
- Compare points are on 1001 35000, total 34000 pts, points to points interval=1pulse
- Moving Speed is 6000 pps
- Compare condition is "="

Program codes:

```
_8164_set_trigger_comparator(0, 1, 1, 1001);

_8164_build_compare_function(0, 1001, 35000, 1, 1);

_8164_set_auto_compare(0, 1);

_8164_start_tr_move(0, 36000, 0, 6000, 0.01, 0.01);
```

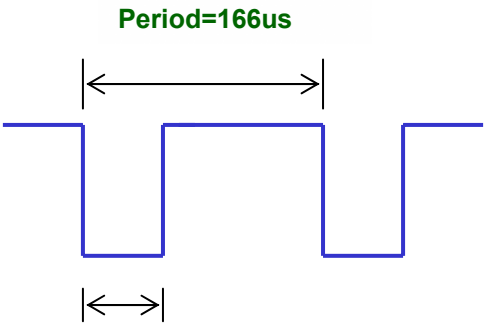
Monitoring or Check the current compare data:

```
_8164_check_compare_data(0, 5, *CurrentData);
```

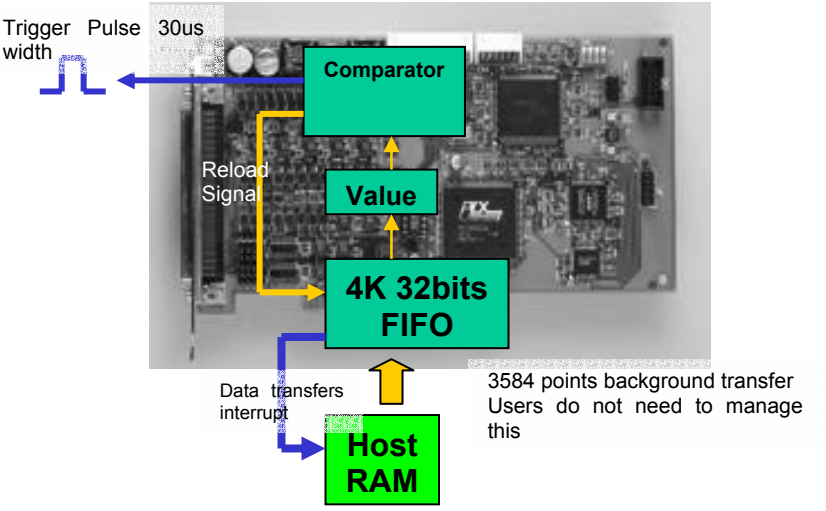
Users can use this function to check if auto-trigger is running.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Results:



The compare mechanism is shown below:



The “Value” block in this figure is the position where the comparison occurs, and where the data can be checked by using _8164_check_compare_data().

Note that at the final compared point will still load an “After-final” point into the “Value” block. Fill a dummy point into the comparison table array at the final position. This value must be far enough from the table’s stroke.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

If using _build_compare_function(), a dummy “after-final” point is automatically loaded. This value is equal to (End point + Interval x Total counts) x moving ratio.

Relative Function:

_8164_set_trigger_comparator(), _8164_build_comp_function()

_8164_build_comp_table(), _8164_set_auto_compare()

_8164_check_compare_data(),_8164_set_trigger_type ()

Refer to section 6.16

4.7.3 Position Latch

The position latch is different than the position compare function in the following way: the position compare function triggers a pulse output via the CMP, when the comparing condition comes into existence, the position latch function receives pulse inputted via the LTC, and then captures all data in all counters at that instant (refer to section 4.4). The latency between the occurring latch signal and the finish position of the captured data is extremely short as the latching procedure is done by hardware. Only axes 2 and 3 can perform a position latch function. LTC3 (axis 2) and LTC4 (axis 3) are used to receive latch pulses.

To set the latch logic use **_8164_set_ltc_logic().**

To obtain the latch values of the counters use **_8164_get_latch_data(AxisNo, CntNo, Pos).** The second parameter “CntNo” is used to indicate the counter of which the latched data will be read.

Relative Function:

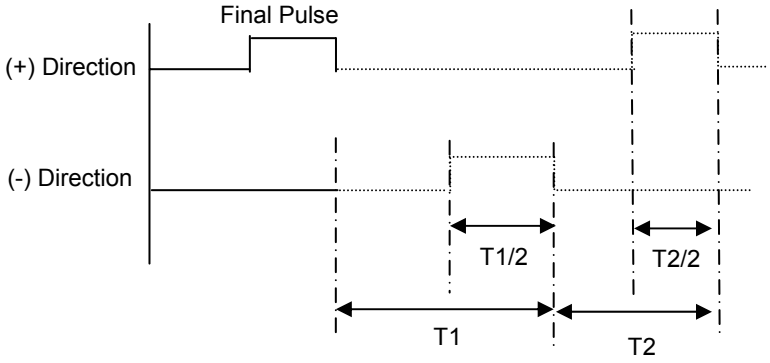
_8164_set_ltc_logic(),_8164_get_latch_data(: refer to section 6.16

4.8 Hardware backlash compensator and vibration suppression

Whenever direction change has occurred, the 8164 outputs a backlash corrective pulse before sending the next command. The function ***_8164_backlash_comp()*** is used to set the pulse number.

In order to minimize vibration when a motor stops, the 8164 can output a single pulse for a negative direction and then single pulse for a positive direction right after completion of a command movement. Refer to the timing chart below, the **_8164_suppress_vibration()** function is used to set T1 & T2.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000



Relative Function:

_8164_backlash_comp(), _8164_suppress_vibration()

Refer to section 6.6

4.9 Software Limit Function

The 8164 provides 2 software limits for each axis. The soft limit is extremely useful in protecting a mechanical system as it works like a physical limit switch when correctly set.

The soft limits are built on comparators 1 and 2 (Refer to section 4.7.1), and the comparing source is the command position counter.

A preset limit value is set in comparators 1 and 2, then, when the command position counter reaches the set limit value, the 8164 reacts by generating the stop immediately or decelerates to stop pulse output.

To set the soft limit: ***_8164_set*_softlimit();**

To enable soft limit: ***_8164*_enable_softlimit();**

To disable soft limit: ***_8164*_diable_softlimit();**

Note: The soft limit is only applied to the **command position** and not the **feedback position** (Refer to 4.4). In cases where the moving ratio is not equal to "1," it is necessary to manually calculate its corresponding command position where the soft limit would be, when using **_8164_set_softlimit()**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Relative Function:

_8164_set_softlimit(), _8164_enable_softlimit(), _8164_diable_softlimit()

Refer to section 6.16

4.10 Interrupt Control

The 8164 motion controller can generate an INT signal to the host PC. The parameter, "**intFlag,**" of the software function **_8164_int_control(),** can enable/disable the interrupt service.

After a interrupt occurs, the function _8164_get_int_status() is used to receive the INT status, which contains information about the INT signal. The INT status of the 8164 comprises of two independent parts: **error_int_status** and **event_int_status**. The **event_int_status** recodes the motion and comparator event under **normal operation.** This INT status can be masked by **_8164_set_int_factor()**. The **error_int_status** is for abnormal stoppage of the 8164 (i.e. EL, ALM, etc.). This INT cannot be masked. The following are the definitions of the two int_status:

event_int_status : can be masked by function call ***_8164_int_*factor()**	
Bit	Description
0	+Soft Limit on and stop
1	-Soft Limit on and stop
2	(Reserved)
3	General Comparator on and stop
4	(Reserved)
5	+End Limit on and stop
6	-End Limit on and stop
7	ALM happen and stop
8	CSTP, Sync. stop on and stop
9	CEMG, Emergency on and stop
10	SD on and slow down to stop
11	(Reserved)
12	Interpolation Error and stop
13	Other axis stop on Interpolation
14	Pulse input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error
17	Pulse input signal error
11~30	(Reserved)
31	Axis Stop Interrupt

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

error_int_status: can not be masked if interrupt service is activated.	
Bit	Description
0	Normal Stop
1	Next command starts
2	Command pre-register 2 is empty
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	(Reserved)
11	General Comparator compared
12	Trigger Comparator compared
13	(Reserved)
14	Counter Latched for axis2,3
15	ORG Input and Latched
16	SD on
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. Start on
20~31	(Reserved)

Use Events to handle interrupts under Windows

To detect an interrupt signal from the 8164 under Windows, a user must create an events array first, then use the functions provided by the 8164 to obtain the interrupt status. A sample program is listed below:

Steps:

1. Define a Global Value to deal with interrupt events. Each event is linked to an axis

   HANDLE hEvent[4];

2. Enable interrupt event service and setup interrupt factors and enable interrupt channel

   _8164_int_enable(0,hEvent);

   _8164_set_int_factor(0,0x01); // Normal Stop interrupt

   _8164_int_control(0,1);

3. Start move command

   _8164_start_tr_move(0,12000,0,10000,0.1,0.1);

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

4. Wait for axis 0 interrupt event

```
STS=WaitForSingleObject(hEvent[0],15000);
ResetEvent(hEvent[0]);

if( STS==WAIT_OBJECT_0 )
{
        _8164_get_int_status(0, &error, &event);
        if( event == 0x01 ) …… ; // Success
}
else if( STS==WAIT_TIME_OUT)
{
// Time out, fail
}
```

8164 Interrupt Service Routine (ISR) with DOS

A DOS function library is included with the 8164 for developing applications under a DOS environment. This library also includes a few functions to work with the ISR. It is highly recommended that programs be written according to the following example for applications working with the ISR. Since the PCI bus has the ability to do IRQ sharing when multiple 8164 are installed, each 8164 should have a corresponding ISR. The library provided have the names of the ISR fixed, for example: *_8164_isr0(void), _8164_isr1(void)…etc.* A sample program is described below. It assumes that two 8164 are present in the system, axes 1 and 5 are requested to work with the ISR:

```
// header file declare
#include“pci_8164.h”

void main(void) {
        I16 TotalCard,i; // Initialize  cards
        _8164_initial(&TotalCard);
        if( TotalCard == 0 ) exit(1);

        _8164_set_int_factor(0,0x1);// Set int factor
        _8164_int_control(0,1);// enable int service

:
        :  // Insert User’s Code in Main
        :  //

_8164_int_control(0,0);// disable int service
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
  _8164_close();// Close PCI-8164
}

void interrupt _8164_isr0(void)
{
   U16 irq_status;// Declaration
   U16 int_type;
   I16 i;
   U32 i_int_status1[4],i_int_status2[4];

   disable();// Stop all int service
   _8164_get_irq_status(0, &irq_status);// Check if this card's int
   if(irq_status)
      {
         for(i=0;i<4;i++) _8164_enter_isr(i);// enter ISR
            for(i=0;i<4;i++)
               {
                  _8164_get_int_type(i, &int_type); // check int type
                    if( int_type & 0x1 )
                    {
                      _8164_get_error_int(i, &int_status1[i]);

                         // Insert User's Code in Error INT
                        //
                        //

                    }
                    if( int_type & 0x2 )
                    {
                          _8164_get_event_int(i, &int_status2[i]);

                         // Insert User's Code in Event INT
                         //
                         //
                    }
               }
                 // end of for every axis on card0

                 for(i=0;i<4;i++) _8164_leave_isr(i);
      }
  else  _8164_not_my_irq(0);

  // Send EOI
       _OUTPORTB(0x20, 0x20);
  _OUTPORTB(0xA0, 0x20);
 enable();// allow int service
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

```
}

void interrupt _8164_isr1(void){}
void interrupt _8164_isr2(void){}
void interrupt _8164_isr3(void){}
void interrupt _8164_isr4(void){}
void interrupt _8164_isr5(void){}
void interrupt _8164_isr6(void){}
void interrupt _8164_isr7(void){}
void interrupt _8164_isr8(void){}
void interrupt _8164_isr9(void){}
void interrupt _8164_isra(void){}
void interrupt _8164_isrb(void){}
```

Relative Function:

_8164_int_control(), _8164_set_int_factor(), _8164_int_enable(), _8164_int_disable(), _8164_get_int_status(), _8164_link_interrupt(),

_8164_get_int_type(), _8164_enter_isr(), _8164_leave_isr()

_8164_get_event_int(), _8164_get_error_int(), _8164_get_irq_status()

_8164_not_my_irq(), _8164_isr0~9, a, b

Refer to section 6.14

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5

Motion Creator

After installing the hardware (Chapters 2 and 3), it is necessary to correctly configure all cards and double check the system before running. This chapter gives guidelines for establishing a control system and manually testing the 8164 cards to verify correct operation. The Motion Creator software provides a simple yet powerful means to setup, configure, test, and debug a motion control system that uses 8164 cards.

Note that Motion Creator is only available for Windows 95/98 or Windows NT/2000/XP with a screen resolution higher than 800x600. It does not run under a DOS environment.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.1 Execute Motion Creator

After installing the software drivers for the 8164 in Windows 95/98/NT/2000/XP, the motion creator program can be located at <chosen path >\PCI-Motion\MotionCreator. To execute the program, double click on the executable file or use **Start→Program Files→PCI-Motion→ MotionCreator**.

5.2 About Motion Creator

Before Running Motion Creator, the following issues should be kept in mind.

1. Motion Creator is a program written in VB 5.0, and is available only for Windows 95/98 or Windows NT/2000/XP with a screen resolution higher than 800x600. It cannot be run under DOS.
2. Motion Creator allows users to save settings and configurations for 8164 cards. Saved configurations will be automatically loaded the next time motion creator is executed. Two files, ***8164.ini*** and ***8164MC.ini,*** in the ***windows root directory*** are used to save all settings and configurations.
3. To duplicate configurations from one system to another, copy 8164.ini and 8164MC.ini into the windows root directory.
4. If multiple 8164 cards use the same Motion Creator saved configuration files, the DLL function call **_8164_config_from_file()** can be invoked within a user developed program. This function is available in a DOS environment as well.

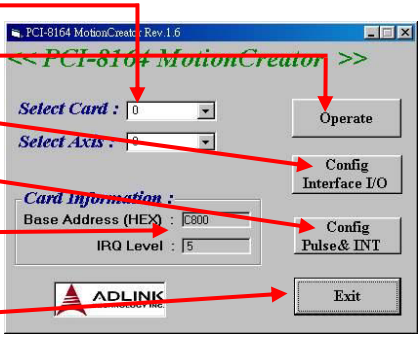
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3 Motion Creator Form Introducing

5.3.1 Main Menu

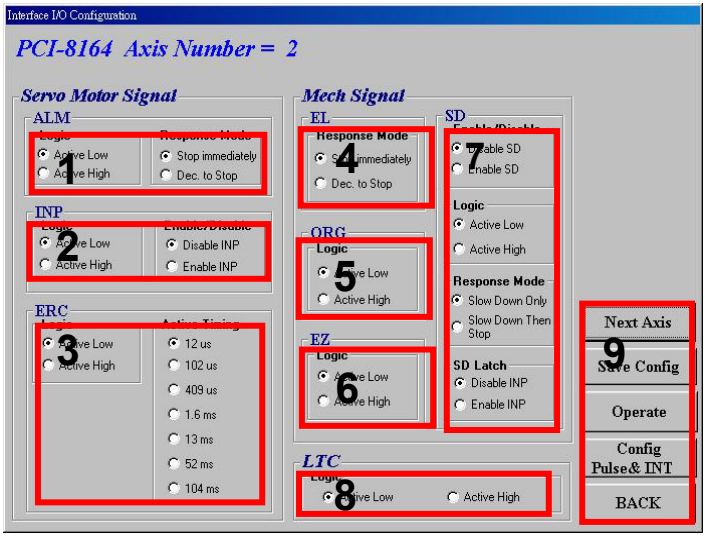
The main menu appears after running Motion Creator. It is used to:

- Select operating card and axis
- Go to **operation** menus (refer to section 5.3.4)
- Go to **Interface I/O** configuration menus (refer to section 5.3.2)
- Go to **Pulse & INT** configuration menus (refer to section 5.3.3)
- Show card information. Related function are : *_8164_get_base_addr(), _8164_get_irq_channel().*
- Exit Motion Creator



5.3.2 Interface I/O Configuration Menu

In this menu, users can configure EL, ORG, EZ, ERC, ALM, INP, SD, and LTC.



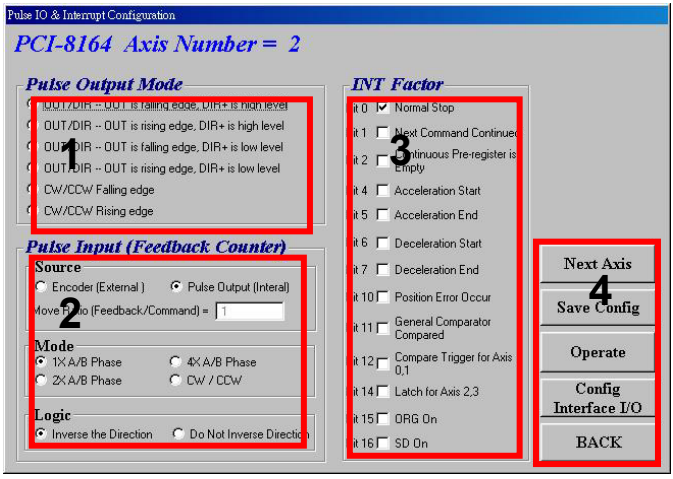
LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

1. **ALM Logic and Response mode:** Select logic and response modes of ALM signal. The related function call is *_8164_set_alm()*.
2. **INP Logic and Enable/Disable selection:** Select logic, and Enable/ Disable the INP signal. The related function call is *_8164_set_inp()*
3. **ERC Logic and Active timing:** Select the Logic and Active timing of the ERC signal. The related function call is *_8164_set_erc()*.
4. **EL Response mode:** Select the response mode of the EL signal. The related function call is *_8164_set_el()*.
5. **ORG Logic:** Select the logic of the ORG signal. The related function call is *_8164_set_home_config()*.
6. **EZ Logic:** Select the logic of the EZ signal. The related function call is *_8164_set_home_config()*.
7. **SD Configuration:** Configure the SD signal. The related function call is *_8164_set_sd()*.
8. **LTC Logic:** Select the logic of the LTC signal. The related function call is *_8164_set_ltc_logic()*.
9. **Buttons:**
    - **Next Axis:** Change operating axis.
    - **Save Config:** Save current configuration to 8164.ini.
    - **Operate:** Go to the operation menu, refer to section 5.3.4
    - **Config Pulse & INT:** Go to the Pulse IO & Interrupt Configuration menu, refer to section 5.3.3
    - **Back:** Return to the main menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3.3 Pulse IO & Interrupt Configuration Menu

In this menu, users can configure pulse input/output and move ratio and INT factor.

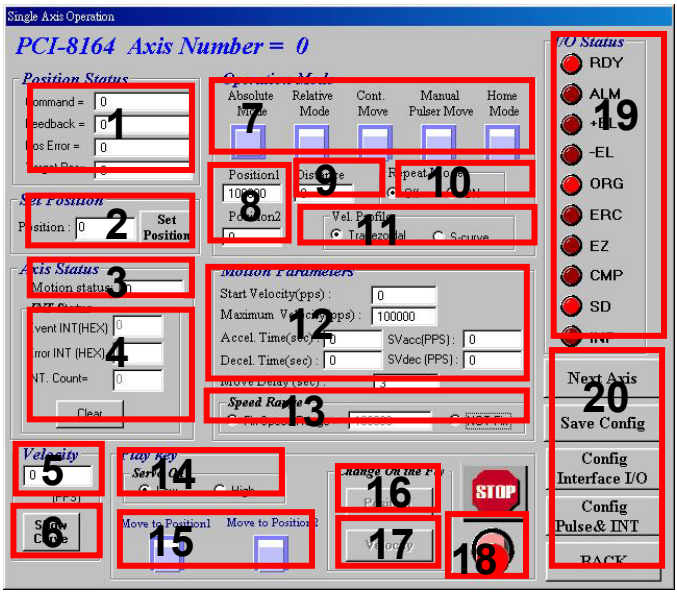


1. **Pulse Output Mode:** Select the output mode of the pulse signal (OUT/ DIR). The related function call is _8164_set_pls_outmode().
2. **Pulse Input:** Sets the configurations of the Pulse input signal(EA/EB). The related function calls are _8164_set_pls_iptmode(), _8164_set_feedback_src().
3. **INT Factor:** Select factors to initiate the event int. The related function call is _8164_set_int_factor().
4. **Buttons:**
   - **Next Axis:** Change operating axis.
   - **Save Config:** Save current configuration to 8164.ini.
   - **Operate:** Go to the operation menu, refer to section 5.3.4
   - **Config Pulse & INT:** Go to the Pulse IO & Interrupt Configuration menu, refer to section 5.3.3
   - **Back:** Return to the main menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5.3.4 Operation menu:

In this menu, users can change the settings a selected axis, including velocity mode motion, preset relative/absolute motion, manual pulse move, and home return.



1. **Position**:

   - Command: displays the value of the command counter. The related function is *_8164_get_command()*.
   - Feedback: displays the value of the feedback position counter. The related function is *_8164_get_position()*
   - Pos Error: displays the value of the position error counter. The related function is *_8164_get_error_counter()*.
   - Target Pos: displays the value of the target position recorder. The related function is *_8164_target_pos()*.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

2. **Position Reset:** clicking this button will set all positioning counters to a specified value. The related functions are:

   _8164_set_position()

   _8164_set_command()

   _8164_reset_error_counter()

   _8164_reset_target_pos()

3. **Motion Status:** Displays the returned value of the _8164_motion_done function. The related function is *_8164_motion_done().*

4. **INT Status:**

   Event: display of event_int_status (in hexadecimal). The related function is *_8164_get_int_status().*

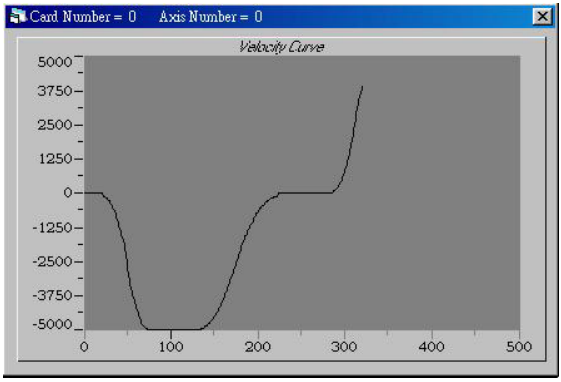
   Error: display of error_int_status (in hexadecimal). The related function is *_8164_get_int_status().*

   Count: total count of interrupt.

   Clear Button: click this button will clear all INT status and counter to '0'.

5. **Velocity:** The absolute value of velocity in units of PPS. The related function is *_8164_get_current_speed().*

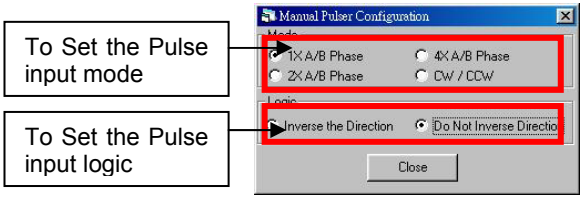
6. **Show Velocity Curve Button:** Clicking this button will open a window showing a velocity vs. time curve. In this curve, every 100ms, a new velocity data point will be added. To close it, click the same button again. To clear data, click on the curve.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7. **Operation Mode:** Select operation mode.

- **Absolute Mode:** “Position1” and “position2” will be used as absolution target positions for motion. The related functions are *_8164_start_ta_move()*, *_8164_start_sa_move()*.
- **Relative Mode:** “Distance” will be used as relative displacement for motion. The related function is *_8164_start_tr_move(), _8164_start_sr_move()*.
- **Cont. Move:** Velocity motion mode. The related function is *_8164_tv_move(), _8164_start_sv_move()*.
- **Manual Pulser Move:** Manual Pulse motion. Clicking this button will invoke the manual pulse configuration window.



- **Home Mode:** Home return motion. Clicking this button will invoke the home move configuration window. The related function is *_8164_set_home_config()*.

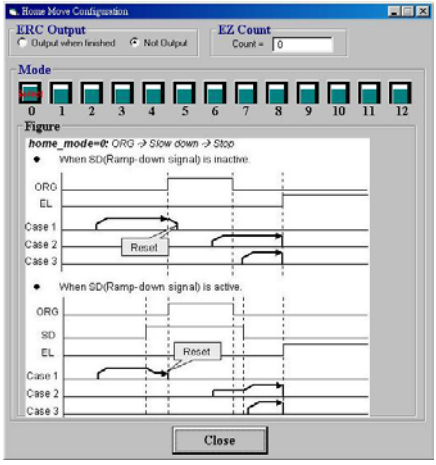
ERC Output: Select if the ERC signal will be sent when home move completes.

EZ Count: Set the EZ count number, which is effective on certain home return modes.

Mode: Select the home return mode. There are 13 modes available.

Home Mode figure: The figure shown explains the actions of the individual home modes.

Close: Click this button close this window.



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

8. **Position:** Set the absolute position for “Absolute Mode.” It is only effective when “Absolute Mode” is selected.

9. **Distance:** Set the relative distance for “Relative Mode.” It is only effective when “Relative Mode” is selected.

10. **Repeat Mode:** When “On” is selected, the motion will become repeat mode (**forward←→backward** or **position1←→position2**). It is only effective when “Relative Mode” or “Absolute Mode” is selected.

11. **Vel. Profile:** Select the velocity profile. Both Trapezoidal and S-Curve are available for “Absolute Mode,” “Relative Mode,” and “Cont. Move.”

12. **Motion Parameters:** Set the parameters for single axis motion. This parameter is meaningless if “Manual Pulser Move” is selected, since the velocity and moving distance is decided by pulse input.

    - **Start Velocity:** Set the start velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -100.0 is the same as 100.0. In “Cont. Move,” both the value and sign are effective. –100.0 means 100.0 in the minus direction.

    - **Maximum Velocity:** Set the maximum velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -5000.0 is the same as 5000.0. In “Cont. Move,” both the value and sing is effective. –5000.0 means 5000.0 in the minus direction.

    - **Accel. Time:** Set the acceleration time in units of second.

    - **Decel. Time**: Set the deceleration time in units of second.

    - **SVacc**: Set the S-curve range during acceleration in units of PPS.

    - **SVdec**: Set the S-curve range during deceleration in unit sof PPS.

    - **Move Delay**: This setting is effective only when repeat mode is set “On.” It will cause the 8164 to delay for a specified time before it continues to the next motion.

13. **Speed Range:** Set the max speed of motion. If “Not Fix” is selected, the “Maximum Speed” will automatically become the maximum speed range, which can not be exceeded by on-the-fly velocity change.

14. **Servo On:** Set the SVON signal output status. The related function is *_8164_set_servo()*.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

15. **Play Key:**

    Left play button: Clicking this button will cause the 8164 start to outlet pulses according to previous setting.

    - In "*Absolute Mode,*" it causes the axis to move to position1.
    - In "*Relative Mode,*" it causes the axis to move forward.
    - In "*Cont. Move,*" it causes the axis to start to move according to the velocity setting.
    - In "*Manual Pulser Move,*" it causes the axis to go into pulse move. The speed limit is the value set by "Maximum Velocity."

    Right play button: Clicking this button will cause the 8164 start to outlet pulses according to previous setting.

    - In "*Absolute Mode,*" it causes the axis to move to position.
    - In "*Relative Mode,*" it causes the axis to move backwards.
    - In "*Cont. Move,*" it causes the axis to start to move according to the velocity setting, but in the opposite direction.
    - In "*Manual Pulser Move,*" it causes the axis to go into pulse move. The speed limit is the value set by "Maximum Velocity."

16. **Change Position On The Fly Button:** When this button is enabled, users can change the target position of the current motion. The new position must be defined in "Position2." The related function is *_8164_p_change().*
17. **Change Velocity On The Fly Button:** When this button is enabled, users can change the velocity of the current motion. The new velocity must be defined in "Maximum Velocity." The related function is *_8164_v_change()*
18. **Stop Button:** Clicking this button will cause the 8164 to decelerate and stop. The deceleration time is defined in "Decel. Time." The related function is *_8164_sd_stop().*
19. **I/O Status:** The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is *_8164_get_io_status().*

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

20. **Buttons**:

    - **Next Axis:** Change operating axis.
    - **Save Config:** Save current configuration to 8164.ini.
    - **Config Pulse & INT:** Go to the Pulse IO & Interrupt Configuration menu, refer to section 5.3.3
    - **Config Interface I/O:** Go to the Interface I/O Configuration menu, refer to section 5.3.2
    - **Back:** Return to the main menu.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6

Function Library

This chapter describes the supporting software for the 8164 card. User can use these functions to develop programs in C, C++, or Visual Basic. If Delphi is used as the programming environment, it is necessary to transform the header files, 8164.h manually.

6.1 List of Functions

Initialization Section 6.3

Function Name	Description
_8164_initial	Card initialization
_8164_initialx	Card initialization with I/O base address and IRQ Channel
_8164_close	Card Close
_8164_get_base_addr	Get base address of 8164
_8164_get_irq_channel	Get the 8164 card's IRQ number
_8164_delay_time	Delay execution of program for specified time in units of ms.
_8164_config_from_file	Configure 8164 cards according to configuration file i.e. 8164.ini, which is created by Motion Creator.
_8164_version_info	Check the hardware and software version

Pulse Input/Output Configuration Section 6.4

Function Name	Description
_8164_set_pls_outmode	Set pulse command output mode
_8164_set_pls_iptmode	Set encoder input mode
_8164_set_feedback_src	Set counter input source

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Velocity mode motion Section 6.5

Function Name	Description
_8164_tv_move	Accelerate an axis to a constant velocity with trapezoidal profile
_8164_sv_move	Accelerate an axis to a constant velocity with S-curve profile
_8164_v_change	Change speed on the fly
_8164_sd_stop	Decelerate to stop
_8164_emg_stop	Immediately stop
_8164_fix_speed_range	Define the speed range
_8164_unfix_speed_range	Release the speed range constrain
_8164_get_current_speed	Get current speed
_8164_verify_speed	Check the min/max acceleration time under max speed

Single Axis Position Mode Section 6.6

Function Name	Description
_8164_start_tr_move	Begin a relative trapezoidal profile move
_8164_start_ta_move	Begin an absolute trapezoidal profile move
_8164_start_sr_move	Begin a relative S-curve profile move
_8164_start_sa_move	Begin an absolute S-curve profile move
_8164_set_move_ratio	Set the ratio of command pulse and feedback pulse.
_8164_p_change	Change position on the fly
_8164_set_pcs_logic	Set the logic of PCS (Position Change Signal)
_8164_set_sd_pin	Set the SD/PCS pin
_8164_backlash_comp	Set backlash corrective pulse for compensation
_8164_suppress_vibration	Set vibration suppressing timing
_8164_set_idle_pulse	Set suppress vibration idle pulse counts
_8164_start_sa_line4	Begin an absolute 4-axis linear interpolation with S-curve profile

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Linear Interpolated Motion Section 6.7

Function Name	Description
_8164_start_tr_move_xy	Begin a relative 2-axis linear interpolation for X & Y, with trapezoidal profile
_8164_start_ta_move_xy	Begin an absolute 2-axis linear interpolation for X & Y, with trapezoidal profile
_8164_start_sr_move_xy	Begin a relative 2-axis linear interpolation for X & Y, with S-curve profile
_8164_start_sa_move_xy	Begin an absolute 2-axis linear interpolation for X & Y, with S-curve profile
_8164_start_tr_move_zu	Begin a relative 2-axis linear interpolation for Z & U, with trapezoidal profile
_8164_start_ta_move_zu	Begin an absolute 2-axis linear interpolation for Z & U, with trapezoidal profile
_8164_start_sr_move_zu	Begin a relative 2-axis linear interpolation for Z & U, with S-curve profile
_8164_start_sa_move_zu	Begin an absolute 2-axis linear interpolation for Z & U, with S-curve profile
_8164_start_tr_line2	Begin a relative 2-axis linear interpolation for any 2 axes, with trapezoidal profile
_8164_start_sr_line2	Begin a relative 2-axis linear interpolation for any 2 axes, with S-curve profile
_8164_start_ta_line2	Begin an absolute 2-axis linear interpolation for any 2 axes, with trapezoidal profile
_8164_start_sa_line2	Begin an absolute 2-axis linear interpolation for any 2 axes, with S-curve profile
_8164_start_tr_line3	Begin a relative 3-axis linear interpolation with trapezoidal profile
_8164_start_sr_line3	Begin a relative 3-axis linear interpolation with S-curve profile
_8164_start_ta_line3	Begin an absolute 3-axis linear interpolation with trapezoidal profile
_8164_start_sa_line3	Begin an absolute 3-axis linear interpolation with S-curve profile,
_8164_start_tr_line4	Begin a relative 4-axis linear interpolation with trapezoidal profile
_8164_start_sr_line4	Begin a relative 4-axis linear interpolation with S-curve profile
_8164_start_ta_line4	Begin an absolute 4-axis linear interpolation with trapezoidal profile

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Circular Interpolation Motion Section 6.8

Function Name	Description
_8164_start_a_arc_xy	Begin an absolute circular interpolation for X & Y
_8164_start_r_arc_xy	Begin a relative circular interpolation for X & Y
_8164_start_a_arc_zu	Begin an absolute circular interpolation for Z & U
_8164_start_r_arc_zu	Begin a relative circular interpolation for Z & U
_8164_start_a_arc2	Begin an absolute circular interpolation for any 2 of the 4 axes
_8164_start_r_arc2	Begin a relative circular interpolation for any 2 of the 4 axes
_8164_start_tr_arc_xyu	Begin a t-curve relative arc with U axis sync.
_8164_start_ta_arc_xyu	Begin a t-curve absolute arc with U axis sync.
_8164_start_sr_arc_xyu	Begin a s-curve relative arc with U axis sync
_8164_start_sa_arc_xyu	Begin a s-curve absolute arc with U axis sync
_8164_start_tr_arc_xy	Begin a t-curve relative circular interpolation for X & Y
_8164_start_ta_arc_xy	Begin a t-curve absolute circular interpolation for X & Y
_8164_start_sr_arc_xy	Begin a s-curve relative circular interpolation for X & Y
_8164_start_sa_arc_xy	Begin a s-curve absolute circular interpolation for X & Y
_8164_start_tr_arc_zu	Begin a t-curve relative circular interpolation for Z & U
_8164_start_ta_arc_zu	Begin a t-curve absolute circular interpolation for Z & U
_8164_start_sr_arc_zu	Begin a s-curve relative circular interpolation for Z & U
_8164_start_sa_arc_zu	Begin a s-curve absolute circular interpolation for Z & U
_8164_start_tr_arc2	Begin a t-curve relative circular interpolation for any 2 of the 4 axes
_8164_start_ta_arc2	Begin a t-curve absolute circular interpolation for any 2 of the 4 axes
_8164_start_sr_arc2	Begin a s-curve relative circular interpolation for any 2 of the 4 axes
_8164_start_sa_arc2	Begin a s-curve absolute circular interpolation for any 2 of the 4 axes

Home Return Mode Section 6.9

Function Name	Description
_8164_set_home_config	Set the home/index logic configuration
_8164_home_move	Begin a home return action
_8164_escape_home	Escape Home Function
_8164_home_search	Auto-Search Home Switch

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Manual Pulser Motion Section 6.10

Function Name	Description
_8164_set_pulser_iptmode	Set pulser input mode
_8164_pulser_vmove	Start pulser v move
_8164_pulser_pmove	Start pulser p move
_8164_pulser_home_move	Start pulser home move
_8164_set_pulser_ratio	Set manual pulser ratio for actual output pulse rate
_8164_pulser_r_line2	pulser mode for 2-axis linear interpolation
_8164_pulser_r_arc2	pulser mode for 2-axis arc interpolation

Motion StatusSection 6.11

Function Name	Description
_8164_motion_done	Return the motion status

Motion Interface I/O Section 6.12

Function Name	Description
_8164_set_alm	Set alarm logic and operating mode
_8164_set_inp	Set INP logic and operating mode
_8164_set_erc	Set ERC logic and timing
_8164_set_servo	Set state of general purpose output pin
_8164_set_sd	Set SD logic and operating mode
_8164_set_el	Set EL logic and operating mode

Motion I/O Monitoring Section 6.13

Function Name	Description
_8164_get_io_status	Get all the motion I/O status of 8164

Interrupt ControlSection 6.14

Function Name	Description
_8164_int_control	Enable/Disable INT service
_8164_int_enable	Enable event (For Windows only)
_8164_int_disable	Disable event (For Windows only)
_8164_get_int_status	Get INT Status (For Windows only)
_8164_link_interrupt	Set link to interrupt call back function (For Windows only)
_8164_set_int_factor	Set INT factor
_8164_get_int_type	Get INT type (For DOS only)
_8164_enter_isr	Enter interrupt service routine (For DOS only)
_8164_leave_isr	Leave interrupt service routine (For DOS only)
_8164_get_event_int	Get event status (For DOS only)
_8164_get_error_int	Get error status (For DOS only)
_8164_get_irq_status	Get IRQ status (For DOS only)
_8164_not_my_irq	Not My IRQ (For DOS only)
_8164_isr0~9, a, b	Interrupt service routine (For DOS only)
_8164_set_axis_stop_int	Enable axis stop int
_8164_mask_axis_stop_int	Mask axis stop int

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Position Control and Counters Section 6.15

Function Name	Description
_8164_get_position	Get the value of the feedback position counter
_8164_set_position	Set the feedback position counter
_8164_get_command	Get the value of the command position counter
_8164_set_command	Set the command position counter
_8164_get_error_counter	Get the value of the position error counter
_8164_reset_error_counter	Reset the position error counter
_8164_get_general_counter	Get the value of the general counter
_8164_set_general_counter	Set the general counter
_8164_get_target_pos	Get the value of the target position recorder
_8164_reset_target_pos	Reset target position recorder
_8164_get_rest_command	Get remaining pulses until the end of motion
_8164_check_rdp	Check the ramping down point data

Position Compare and Latch Section 6.16

Function Name	Description
_8164_set_ltc_logic	Set the LTC logic
_8164_get_latch_data	Get latched counter data
_8164_set_soft_limit	Set soft limit
_8164_enable_soft_limit	Enable soft limit function
_8164_disable_soft_limit	Disable soft limit function
_8164_set_error_counter_check	Step-losing detection
_8164_set_general_comparator	Set general-purposed comparator
_8164_set_trigger_comparator	Set Trigger comparator
_8164_set_trigger_type	Set the trigger output type
_8164_check_compare_data	Check current comparator data
_8164_check_compare_status	Check current comparator status
_8164_set_auto_compare	Set comparing data source for auto loading
_8164_build_compare_function	Build compare data via constant interval
_8164_build_compare_table	Build compare data via compare table
_8164_cmp_v_change	Speed change by comparator

Continuous Motion Section 6.17

Function Name	Description
_8164_set_continuous_move	Enable continuous motion for absolute motion
_8164_check_continuous_buffer	Check if the buffer is empty

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Multiple Axes Simultaneous Operation Section 6.18

Function Name	Description
_8164_set_tr_move_all	Multi-axis simultaneous operation setup
_8164_set_ta_move_all	Multi-axis simultaneous operation setup
_8164_set_sr_move_all	Multi-axis simultaneous operation setup
_8164_set_sa_move_all	Multi-axis simultaneous operation setup
_8164_start_move_all	Begin a multi-axis trapezoidal profile motion
_8164_stop_move_all	Simultaneously stop multi-axis motion
_8164_set_sync_option	Optional sync options
_8164_set_sync_stop_mode	Set the stop mode when CSTOP signal is ON

General-purposed TTL Output Section 6.19 (PCI-8164 Only)

Function Name	Description
_8164_d_output	Digital Output
_8164_get_dio_status	Get DO status

General-purposed DIO Section 6.20 (MPC-8164 Only)

_8164_write_do –	Digital Output
_8164_read_di –	Digital Input

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.2 C/C++ Programming Library

This section details all the functions. The function prototypes and some common data types are declared in **PCI-8164.H or MPC-8164.H**. We suggest you use these data types in your application programs. The following table shows the data type names and their range.

Type Name	Description	Range
U8	8-bit ASCII character	0 to 255
I16	16-bit signed integer	-32768 to 32767
U16	16-bit unsigned integer	0 to 65535
I32	32-bit signed long integer	-2147483648 to 2147483647
U32	32-bit unsigned long integer	0 to 4294967295
F32	32-bit single-precision floating-point	-3.402823E38 to 3.402823E38
F64	64-bit double-precision floating-point	-1.797683134862315E308 to 1.797683134862315E309
Boolean	Boolean logic value	TRUE, FALSE

The functions of the 8164's software drivers use full-names to represent the functions real meaning. The naming convention rules are:

In a 'C' programming environment:

_{hardware_model}_{action_name}. e.g. **`_8164_Initial()`**.

In order to recognize the difference between a C library and a VB library, a capital "B" is placed at the beginning of each function name e.g. **`B_8164_Initial()`**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.3 Initialization

@ Name

_8164_Initial – Card Initialization
_8164_Initialx – Card Initialization with I/O base address and IRQ channel
_8164_Close – Card Close
_8164_get_base_addr – Get the base address of 8164_
_8164_get_irq_channel – Get the 8164 card's IRQ number
_8164_delay_time – delay execution of program for specified time in units of ms.
_8164_config_from_file – Configure 8164 card according to configuration file i.e. 8164.ini.
_8164_version_info – Check hardware and software version information

@ Description

_8164_Initial :

This function is used to initialize an 8164 card without assigning the hardware resources. All 8164 cards must be initialized by this function before calling other functions. 8164 uses this function in all platforms because it is PCI bus Plug and Play compatible. MPC-8164 uses this function in Windows 98/NT/2000/XP.

_8164_initialx:

This function is used to initialize 8164 cards with an I/O base address and IRQ channel. MPC-8164 uses this function under DOS, Windows CE, and Linux.

_8164_Close :

This function is used to close 8164 card and release its resources, which should be called at the end of an application.

_8164_get_irq_channel :

This function is used to get the 8164 card's IRQ number.

_8164_get_base_addr:

This function is used to get the 8164 card's base address.

_8164_delay_time:

This function is used to delay execution of program for specified time in units of ms.

_8164_config_from_file:

This function is used to load the configuration of the 8164 according to specified file. By using **Motion Creator**, users can test and configure the 8164 correctly. After pressing the "save config" button, the configuration is saved as 8164.ini in the Windows directory. By specifying it in the parameter, the configuration will be automatically loaded.

When this function is executed, all 8164 cards in the system will be configured as the following functions were called according to parameters recorded in 8164.ini.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_set_pls_outmode
_8164_set_feedback_src
_8164_set_pls_iptmode
_8164_set_home_config
_8164_set_int_factor
_8164_set_el
_8164_set_ltc_logic
_8164_set_erc
_8164_set_sd
_8164_set_alm
_8164_set_inp
_8164_set_move_ratio

_8164_version_info:

Lets users read back version information

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_initial(I16 *existCards);
I16 _8164_close(void);
I16 _8164_get_irq_channel(I16 cardNo, U16 *irq_no );
I16 _8164_get_base_addr(I16 cardNo, U16 *base_addr );
I16 _8164_delay_time(I16 AxisNo, U32 MiniSec);
I16 _8164_config_from_file(char *filename);
I16 _8164_version_info(I16 CardNo, U16 *HardwareInfo, U16
 *SoftwareInfo, U16 *DriverInfo);
```

Visual Basic (Windows 95/NT)

```
B_8164_initial (existCards As Integer) As Integer
B_8164_close () As Integer
B_8164_get_irq_channel (ByVal CardNo As Integer, irq_no As Integer) As
        Integer
B_8164_get_base_addr (ByVal CardNo As Integer, base_addr As Integer)
        As Integer
B_8164_delay_time (ByVal AxisNo As Integer, ByVal MiniSec As Long) As
        Integer
B_8164_config_from_file(ByVal filename As string)as integer
B_8164_version_info (ByVal CardNo As Integer, HardwareInfo As Integer,
        SoftwareInfo As Integer, DriverInfo As Integer) As Integer
```

@ Argument

***existCards**: Number of existing 8164 cards
cardNo: The 8164 card index number
AxNo: To specify which axis is used to measure the delay time
***irq_no**: IRQ number of a specified 8164 card.
***base_addr**: base address of specified 8164 card
***Filename**: The specified filename recording the configuration of 8164. This file must be created by **Motion Creator** of the 8164.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

***Hardwareinfo**: Hardware version readback in decimal

Digit 3	Digit 2	Digit 1	Digit 0
0: PCL-6045 1: PCL-6045A	Undifined	0:PCI-8164 1: MPC-8164	0: CPLD A1, A2 3: CPLD A3

***SoftwareInfo**: Software library version readback in decimal

	Digit 4	Digit 3	Digit 2	Digit 1	Digit 0
Win32	3: Year 2003	Month:1~12		Day:01~31	
WinCE		Month + 12			
DOS		Month + 24			
DOSExt		Month + 36			
Linux		Month + 48			

***DriverInfo**: Device driver version readback in decimal

	Digit 4	Digit 3	Digit 2	Digit 1	Digit 0
Win32	3: Year 2003	Month: 1~12		Day:01~31	
WinCE		Month + 12			
DOS		Month + 24			
DOSExt		Month + 36			
Linux		Month + 48			

@ Return Code

ERR_NoError
ERR_NoCardFound
ERR_PCIBiosNotExist
ERR_ConigFileOpenError

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.4 Pulse Input/Output Configuration

@ Name

_8164_set_pls_outmode – Set the configuration for pulse command output.
_8164_set_pls_iptmode – Set the configuration for feedback pulse input.
_8164_set_feedback_src – Enable/Disable the external feedback pulse input

@ Description

_8164_set_pls_outmode:

Configure the output modes of command pulses. There are 6 modes for command pulse output.

_8164_set_pls_iptmode:

Configure the input modes of external feedback pulses. There are four types for feedback pulse input. Note that this function makes sense only when the **Src** parameter in **_8164_set_feedback_src()** function is enabled.

_8164_set_feedback_src:

If external encoder feedback is available in the system, set the **Src** parameter in this function to an *Enabled* state. Then, the internal 28-bit up/down counter will count according to the configuration of the **_*8164*_set_pls_iptmode()** function. Else, the counter will count the command pulse output.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_pls_outmode(I16 AxisNo, I16 pls_outmode);
I16 _8164_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _8164_set_feedback_src(I16 AxisNo, I16 Src);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_pls_outmode (ByVal AxisNo As Integer, ByVal pls_outmode
        As Integer) As Integer
B_8164_set_pls_iptmode (ByVal AxisNo As Integer, ByVal pls_iptmode As
        Integer, ByVal pls_logic As Integer) As Integer
B_8164_set_feedback_src (ByVal AxisNo As Integer, ByVal Src As Integer)
        As Integer
```

@ Argument

AxisNo: Axis number designated to configure pulse Input/Output.
pls_outmode: Setting of command pulse output mode

Value	Meaning
0	OUT/DIROUT Falling edge, DIR+ is high level
1	OUT/DIROUT Rising edge, DIR+ is high level
2	OUT/DIROUT Falling edge, DIR+ is low level
3	OUT/DIROUT Rising edge, DIR+ is low level
4	CW/CCW Falling edge
5	CW/CCW Rising edge

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

pls_iptmode: setting of encoder feedback pulse input mode

Value	Meaning
0	1X A/B
1	2X A/B
2	4X A/B
3	CW/CCW

pls_logic: Logic of encoder feedback pulse
pls_logic=0, Not inverse direction.
pls_logic=1, Inverse direction

Src: Counter source

Value	Meaning
0	External Feedback
1	Command pulse

@ Return Code

ERR_NoError

6.5 Velocity mode motion

@ Name

_8164_tv_move – Accelerate an axis to a constant velocity with trapezoidal profile
_8164_sv_move – Accelerate an axis to a constant velocity with S-curve profile
_8164_v_change –Change speed on the fly
_8164_sd_stop –Decelerate to stop
_8164_emg_stop –Immediately stop
_8164_fix_speed_range – Define the speed range
_8164_unfix_speed_range – Release the speed range constrain
_8164_get_current_speed – Get current speed
_8164_verify_speed – get speed profile's minimum and maximum acc/dec time

@ Description

_8164_tv_move:

This function is to accelerate an axis to the specified constant velocity with a trapezoidal profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of the velocity parameter.

_8164_sv_move:

This function is to accelerate an axis to the specified constant velocity with a S-curve profile. The axis will continue to travel at a constant velocity until the velocity is changed or the axis is commanded to stop. The direction is determined by the sign of velocity parameter.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_v_change:

This function changes the moving velocity with a trapezoidal profile or S-curve profile. Before calling this function, it is necessary to define the speed range by _8164_fix_speed_range. _8164_v_change is also applicable on pre-set motion. Note: The velocity profile is decided by an original motion profile. When using in S-curve, please set the motion to be pure S-curve. There are some limitations for this function; please refer to section 4.6.1 before use it.

_8164_sd_stop:

This function is used to decelerate an axis to stop with a trapezoidal or S-curve profile. This function is also useful when a ***preset move*** (both trapezoidal and S-curve motion), ***manual move,*** or ***home return*** function is performed. Note: The velocity profile is decided by original motion profile.

_8164_emg_stop:

This function is used to immediately stop an axis. This function is also useful when a ***preset move*** (both trapezoidal and S-curve motion), ***manual move,*** or ***home return*** function is performed.

_8164_fix_speed_range

This function is used to define the speed range. It should be called before starting motion that may contains velocity changing.

_8164_unfix_speed_range

This function is used to release speed range constrains.

_8164_get_current_speed

This function is used to read the current pulse output rate of a specified axis. It is applicable in any time in any operating mode.

_8164_verify_speed

Find a speed profile's minimum and maximum accelerating time.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8164_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64
        SVacc);
I16 _8164_v_change(I16 AxisNo, F64 NewVel, F64 Tacc);
I16 _8164_sd_stop(I16 AxisNo,F64 Tdec);
I16 _8164_emg_stop(I16 AxisNo);
F64 _8164_fix_speed_range(I16 AxisNo, F64 MaxVel);
I16 _8164_unfix_speed_range(I16 AxisNo);
I16 _8164_get_current_speed(I16 AxisNo, F64 *speed);
F64 _8164_verify_speed(F64 StrVel,F64 MaxVel,F64 *minAccT,F64
        *maxAccT, F64 MaxSpeed);
```


LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visual Basic (Windows 95/NT)

B_8164_tv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer
B_8164_sv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer
B_8164_v_change (ByVal AxisNo As Integer, ByVal NewVel As Double, ByVal TimeSecond As Double) As Integer
B_8164_sd_stop (ByVal AxisNo As Integer, ByVal Tdec As Double) As Integer
B_8164_emg_stop (ByVal AxisNo As Integer) As Integer
B_8164_fix_speed_range (ByVal AxisNo As Integer, ByVal MaxVel As Double) As Integer
B_8164_unfix_speed_range (ByVal AxisNo As Integer) As Integer
B_8164_get_current_speed (ByVal AxisNo As Integer, Speed As Double) As Integer
B_8164_verify_speed Lib "8164.DLL" Alias "_8164_verify_speed" (ByVal StrVel As Double, ByVal MaxVel As Double, minAccT As Double, maxAccT As Double, ByVal MaxSpeed As Double) As Double

@ Argument

AxisNo: Axis number designated to move or stop.
StrVel: Starting velocity in units of pulse per second
MaxVel: Maximum velocity in units of pulse per second
Tacc: Specified acceleration time in units of second
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve
NewVel: New velocity in units of pulse per second
Tdec: specified deceleration time in units of second
***Speed:** Variable to save current speed.
(speed range: 0~6553500)

@ Return Code

ERR_NoError
ERR_SpeedError
ERR_SpeedChangeError
ERR_SlowDownPointError
ERR_AxisAlreadyStop

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.6 Single Axis Position Mode

@ Name

_8164_start_tr_move – Begin a relative trapezoidal profile move
_8164_start_ta_move – Begin an absolute trapezoidal profile move
_8164_start_sr_move – Begin a relative S-curve profile move
_8164_start_sa_move – Begin an absolute S-curve profile move
_8164_set_move_ratio –Set the ratio of command pulse and feedback pulse.
_8164_p_change – Change position on the fly
_8164_set_pcs_logic –Set the logic of PCS (Position Change Signal) pin
_8164_set_sd_pin –Set SD/PCS pin
_8164_backlash_comp – Set backlash compensating pulse for compensation
_8164_suppress_vibration – Set vibration suppressing timing
_8164_set_idle_pulse – Set suppress vibration idle pulse counts

@ Description

General: The moving direction is determined by the sign of the **Pos** or **Dist** parameter. If the moving distance is too short to reach the specified velocity, the controller will automatically lower the MaxVel, and the Tacc, Tdec, VSacc, and VSdec will also become shorter while dV/dt(acceleration / deceleration) and d(dV/dt)/dt (jerk) are keep unchanged.

_8164_start_tr_move:

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerates to stop at the relative distance with trapezoidal profile. The acceleration and deceleration time is specified independently–it does not let the program wait for motion completion but immediately returns control to the program.

_8164_start_ta_move :

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerates to stop at the specified absolute position with trapezoidal profile. The acceleration and deceleration time is specified independentlyThis command does not let the program wait for motion completion, but immediately returns control to the program..

_8164_start_sr_move:

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerates to stop at the relative distance with S-curve profile. The acceleration and deceleration time is specified independently.This command does not let the program wait for motion completion, but immediately returns control to the program..

_8164_start_sa_move :

This function causes the axis to accelerate from a starting velocity, slew at constant velocity, and decelerates to stop at the specified absolute position with S-curve profile. The acceleration and deceleration time is specified independently.This command does not let the program wait for motion completion but immediately returns control to the program..

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_set_move_ratio :

This function configures scale factors for the specified axis. Usually, the axes only need scale factors if their mechanical resolutions are different. For example, if the resolution of feedback sensors is two times resolution of command pulse, then ***ratio = 2***.

_8164_p_change

This function is used to change target position on the fly. There are some limitations on this function. Please refer to section 4.6.2 before use it.

_8164_set_pcs_logic :

This function is used to set the logic of Position Change Signal (pcs). The PCS share the same pin with SD signal. Only when the SD/PCS pin was set to PCS by _8164_set_sd_pin, this _8164_set_pcs_logic function becomes effective.

_8164_set_sd_pin :

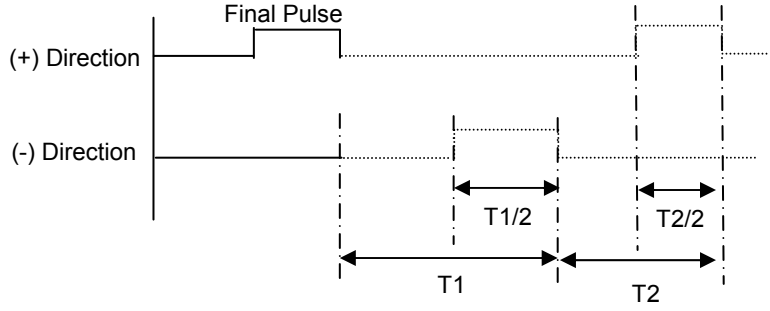
This function is used to set the operating mode of the SD pin. The SD pin may be used either as a Slow-Down signal input or as a Position Change Signal (PCS) input. Please refer to section 4.3.1

_8164_backlash_comp :

Whenever direction change occurs, the 8164 outputs backlash corrective pulses before sending commands. This function is used to set the compensation pulse numbers.

_8164_suppress_vibration

This function is used to suppress vibration of mechanical systems by outputting a single pulse for negative direction and then single pulse for positive direction right after completion of command movement.



_8164_set_idle_pulse :

Set suppress vibration idle pulse counts.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Syntax

C/C++ (DOS, Windows 95/NT)

I16 _8164_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc,F64 Tdec);
I16 _8164_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _8164_p_change(I16 AxisNo, F64 NewPos);
I16 _8164_set_pcs_logic(I16 AxisNo, I16 pcs_logic);
I16 _8164_set_sd_pin(I16 AxisNo, I16 Type);
I16 _8164_backlash_comp(I16 AxisNo, I16 BCompPulse);
I16 _8164_suppress_vibration(I16 AxisNo, U16 T1, U16 T2);
I16 _8164_set_idle_pulse(I16 AxisNo, I16 idl_pulse);

Visual Basic (Windows 95/NT)

B_8164_start_tr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_8164_start_ta_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer
B_8164_start_sr_move (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_8164_start_sa_move (ByVal AxisNo As Integer, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer
B_8164_set_move_ratio (ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer
B_8164_p_change (ByVal AxisNo As Integer, ByVal NewPos As Double) As Integer
B_8164_set_pcs_logic (ByVal AxisNo As Integer, ByVal pcs_logic As Integer) As Integer
B_8164_set_sd_pin (ByVal AxisNo As Integer, ByVal Type As Integer) As Integer
B_8164_backlash_comp (ByVal AxisNo As Integer, ByVal BCompPulse As Integer, ByVal ForwardTime As Integer) As Integer
B_8164_suppress_vibration (ByVal AxisNo As Integer, ByVal ReserveTime As Integer, ByVal ForwardTime As Integer) As Integer
B_8164_set_idle_pulse(ByVal AxisNo As Integer, ByVal idl_pulse As Integer);

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Argument

AxisNo: Axis number designated to move or change position.
Dist: Specified relative distance to move
Pos: Specified absolute position to move
StrVel: Starting velocity of a velocity profile in units of pulse per second
MaxVel: Starting velocity of a velocity profile in units of pulse per second
Tacc: Specified acceleration time in units of seconds
Tdec: Specified deceleration time in units of seconds
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve
Move_ratio: ratio of (feedback resolution)/(command resolution) , should not be 0
NewPos: specified new absolute position to move
pcs_logic: Specify he pcs logic.
Value = 0: low active ,
Value = 1: high active
Type: define the SD pin usage
Value = 0 : SD pin as SD signal
Value = 1: SD pin as PCS signal
BcompPulse: Specified number of corrective pulses, 12 bit
T1: Specified Reverse Time, 0 ~ 65535, unit 1.6 us
T2: Specified Forward Time, 0 ~ 65535, unit 1.6 us
Idl_pulse: Idl_pulse=0~7

@ Return Code

ERR_NoError
ERR_SpeedError
ERR_PChangeSlowDownPointError
ERR_MoveRatioError

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.7 Linear Interpolated Motion

@ Name

_8164_start_tr_move_xy – Begin a relative 2-axis linear interpolation for X & Y, with trapezoidal profile,
_8164_start_ta_move_xy – Begin an absolute 2-axis linear interpolation for X & Y, with trapezoidal profile,
_8164_start_sr_move_xy – Begin a relative 2-axis linear interpolation for X & Y, with S-curve profile,
_8164_start_sa_move_xy – Begin an absolute 2-axis linear interpolation for X & Y, with S-curve profile,
_8164_start_tr_move_zu – Begin a relative 2-axis linear interpolation for Z & U, with trapezoidal profile,
_8164_start_ta_move_zu – Begin an absolute 2-axis linear interpolation for Z & U, with trapezoidal profile,
_8164_start_sr_move_zu – Begin a relative 2-axis linear interpolation for Z & U, with S-curve profile,
_8164_start_sa_move_zu – Begin an absolute 2-axis linear interpolation for Z & U, with S-curve profile,
_8164_start_tr_line2 – Begin a relative 2-axis linear interpolation for any 2 axes, with trapezoidal profile,
_8164_start_sr_line2 – Begin a relative 2-axis linear interpolation for any 2 axes,, with S-curve profile
_8164_start_ta_line2 – Begin an absolute 2-axis linear interpolation for any 2 axes,, with trapezoidal profile
_8164_start_sa_line2 – Begin an absolute 2-axis linear interpolation for any 2 axes,, with S-curve profile,
_8164_start_tr_line3 – Begin a relative 3-axis linear interpolation with trapezoidal profile,
_8164_start_sr_line3 – Begin a relative 3-axis linear interpolation with S-curve profile
_8164_start_ta_line3 – Begin an absolute 3-axis linear interpolation with trapezoidal profile
_8164_start_sa_line3 – Begin an absolute 3-axis linear interpolation with S-curve profile,
_8164_start_tr_line4 – Begin a relative 4-axis linear interpolation with trapezoidal profile,
_8164_start_sr_line4 – Begin a relative 4-axis linear interpolation with S-curve profile
_8164_start_ta_line4 – Begin an absolute 4-axis linear interpolation with trapezoidal profile
_8164_start_sa_line4 – Begin an absolute 4-axis linear interpolation with S-curve profile
_8164_set_axis_option – Choose the interpolation speed mode

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Description

Function	No. of interpolating axes	Velocity Profile	Relative / Absolute	Target Axes
_8164_start_tr_move_xy	2	T	R	Axes 0 & 1
_8164_start_ta_move_xy	2	T	A	Axes 0 & 1
_8164_start_sr_move_xy	2	S	R	Axes 0 & 1
_8164_start_sa_move_xy	2	S	A	Axes 0 & 1
_8164_start_tr_move_zu	2	T	R	Axes 2 & 3
_8164_start_ta_move_zu	2	T	A	Axes 2 & 3
_8164_start_sr_move_zu	2	S	R	Axes 2 & 3
_8164_start_sa_move_zu	2	S	A	Axes 2 & 3
_8164_start_tr_move_line2	2	T	R	Any 2 of 4
_8164_start_ta_move_ line2	2	T	A	Any 2 of 4
_8164_start_sr_move_ line2	2	S	R	Any 2 of 4
_8164_start_sa_move_ line2	2	S	A	Any 2 of 4
_8164_start_tr_move_ line3	3	T	R	Any 3 of 4
_8164_start_ta_move_ line3	3	T	A	Any 3 of 4
_8164_start_sr_move_ line3	3	S	R	Any 3 of 4
_8164_start_sa_move_ line3	3	S	A	Any 3 of 4
_8164_start_tr_move_ line4	4	T	R	Any 4 of 4
_8164_start_ta_move_ line4	4	T	A	Any 4 of 4
_8164_start_sr_move_ line4	4	S	R	Any 4 of 4
_8164_start_sa_move_ line4	4	S	A	Any 4 of 4

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_start_tr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_ta_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_move_xy(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_start_sa_move_xy(I16 CardNo, F64 PosX, F64 PosY, F64
        StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_start_tr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_ta_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_move_zu(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel,
        F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_start_sa_move_zu(I16 CardNo, F64 PosX, F64 PosY, F64
        StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8164_start_tr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
        F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_ta_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
        F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
        F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
        SVdec);
I16 _8164_start_sa_line2(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
        F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64
        SVdec);
I16 _8164_start_tr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
        F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_ta_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
        F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_line3(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY,
        F64 DistZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
        SVacc, F64 SVdec);
I16 _8164_start_sa_line3(I16 CardNo, I16 *AxisArray, F64 PosX, F64 PosY,
        F64 PosZ, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
        SVacc, F64 SVdec);
I16 _8164_start_tr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64
        DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_ta_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ, F64
        PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8164_start_sr_line4(I16 CardNo, F64 DistX, F64 DistY, F64 DistZ, F64
        DistU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc,
        F64 SVdec);
I16 _8164_start_sa_line4(I16 CardNo, F64 PosX, F64 PosY, F64 PosZ,
        F64 PosU, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
        SVacc, F64 SVdec);
I16 FNTYPE _8164_set_axis_option(I16 AxisNo, I16 option);
```


LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visual Basic (Windows 95/NT)

B_8164_start_tr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_move_xy (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_move_xy (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_tr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_move_zu (ByVal CardNo As Integer, ByVal Dist As Double, ByVal Dist As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_move_zu (ByVal CardNo As Integer, ByVal Pos As Double, ByVal Pos As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_tr_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_line2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

B_8164_start_tr_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_line3 (ByVal CardNo As Integer, AxisArray As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_tr_line4 (ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal DistU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_line4 (ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal PosU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_line4 (ByVal CardNo As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal DistZ As Double, ByVal DistU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_line4 (ByVal CardNo As Integer, ByVal PosX As Double, ByVal PosY As Double, ByVal PosZ As Double, ByVal PosU As Double, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_set_axis_option (ByVal AxisNo As Integer, ByVal option1 As Integer) As Integer

@ Argument

CardNo: Card number designated to perform linear interpolation
DistX: specified relative distance of axis 0 to move
DistY: specified relative distance of axis 1 to move
DistZ: specified relative distance of axis 2 to move
DistU: specified relative distance of axis 3 to move
PosX: specified absolute position of axis 0 to move
PosY: specified absolute position of axis 1 to move
PosZ: specified absolute position of axis 2 to move
PosU: specified absolute position of axis 3 to move
StrVel: starting velocity of a velocity profile in units of pulse per second
MaxVel: starting velocity of a velocity profile in units of pulse per second
Tacc: specified acceleration time in units of seconds

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Tdec: specified deceleration time in units of seconds
SVacc: specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve
SVdec: specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve
AxisArray: Array of axis number to perform interpolation.
Example: Int AxisArray[2] = {0,2}; // axis 0 & 2
Int AxisArray[3] = {0,1,3}; // axis 0,1,3
Note: AxisArray[n] must be smaller than AxisArray[m], if n<m.
Option1: 0=default line move mode
1=Composite speed constant mode

@ Return Code

ERR_NoError
ERR_SpeedError
ERR_AxisArrayErrot

6.8 Circular Interpolation Motion

@ Name

_8164_start_r_arc_xy – Begin a relative circular interpolation for X & Y
_8164_start_a_arc_xy – Begin an absolute circular interpolation for X & Y
_8164_start_r_arc_zu – Begin a relative circular interpolation for Z & U
_8164_start_a_arc_zu – Begin an absolute circular interpolation for Z & U
_8164_start_r_arc2 – Begin a relative circular interpolation for any 2 axes
_8164_start_a_arc2 – Begin an absolute circular interpolation for any 2 axes

_8164_start_tr_arc_xyu – Begin a T-curve relative circular interpolation
_8164_start_ta_arc_xyu – Begin a T-curve absolute circular interpolation
_8164_start_sr_arc_xyu – Begin a S-curve relative circular interpolation
_8164_start_sa_arc_xyu – Begin a S-curve absolute circular interpolation
_8164_start_tr_arc_yzu – Begin a T-curve relative circular interpolation
_8164_start_ta_arc_yzu – Begin a T-curve absolute circular interpolation
_8164_start_sr_arc_yzu – Begin a S-curve relative circular interpolation
_8164_start_sa_arc_yzu – Begin a T-curve absolute circular interpolation

_8164_start_tr_arc2 – Begin a T-curve relative circular interpolation
_8164_start_ta_arc2 – Begin a T-curve absolute circular interpolation
_8164_start_sr_arc2– Begin a S-curve relative circular interpolation
_8164_start_sa_arc2– Begin a S-curve absolute circular interpolation
_8164_start_tr_arc_xy – Begin a T-curve relative circular interpolation
_8164_start_ta_arc_xy – Begin a T-curve absolute circular interpolation
_8164_start_tr_arc_zu – Begin a T-curve relative circular interpolation
_8164_start_ta_arc_zu – Begin a T-curve absolute circular interpolation
_8164_start_sr_arc_xy – Begin a S-curve relative circular interpolation
_8164_start_sa_arc_xy – Begin a S-curve absolute circular interpolation
_8164_start_sr_arc_zu – Begin a S-curve relative circular interpolation
_8164_start_sa_arc_zu – Begin a S-curve absolute circular interpolation

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Description

Function	Relative / Absolute	Speed Profile	Target Axes	Hardware version bit 12
_8164_start_r_arc_xy	R	Flat	Axes 0 & 1	0 or 1
_8164_start_a_arc_xy	A	Flat	Axes 0 & 1	0 or 1
_8164_start_r_arc_zu	R	Flat	Axes 2 & 3	0 or 1
_8164_start_a_arc_zu	A	Flat	Axes 2 & 3	0 or 1
_8164_start_r_arc2	R	Flat	Any 2 of 4	0 or 1
_8164_start_a_arc2	A	Flat	Any 2 of 4	0 or 1
_8164_start_tr_arc_xyu	R	T-curve	Axes 0 & 1	0 or 1
_8164_start_ta_arc_xyu	A	T-Curve	Axes 0 & 1	0 or 1
_8164_start_sr_arc_xyu	R	S-Curve	Axes 1 & 2	0 or 1
_8164_start_sa_arc_xyu	A	S-Curve	Axes 1 & 2	0 or 1
_8164_start_tr_arc_xy	R	T-curve	Axes 0 & 1	1
_8164_start_ta_arc_xy	A	T-Curve	Axes 0 & 1	1
_8164_start_sr_arc_xy	R	S-Curve	Axes 0 & 1	1
_8164_start_sa_arc_xy	A	S-Curve	Axes 0 & 1	1
_8164_start_tr_arc_zu	R	T-curve	Axes 2 & 3	1
_8164_start_ta_arc_zu	A	T-Curve	Axes 2 & 3	1
_8164_start_sr_arc_zu	R	S-Curve	Axes 2 & 3	1
_8164_start_sa_arc_zu	A	S-Curve	Axes 2 & 3	1
_8164_start_tr_arc2	R	T-curve	Any 2 of 4	1
_8164_start_ta_arc2	A	T-Curve	Any 2 of 4	1
_8164_start_sr_arc2	R	S-Curve	Any 2 of 4	1
_8164_start_sa_arc2	A	S-Curve	Any 2 of 4	1

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_start_r_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
        OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8164_start_a_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
        I16 DIR, F64 MaxVel);
I16 _8164_start_r_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
        OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8164_start_a_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey,
        I16 DIR, F64 MaxVel);
I16 _8164_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64
        OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8164_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64
        Ex, F64 Ey, I16 DIR, F64 MaxVel);

I16 _8164_start_tr_arc_xyu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64
        OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64
        Tacc);
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

I16 _8164_start_ta_arc_xyu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8164_start_sr_arc_xyu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8164_start_sa_arc_xyu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8164_start_tr_arc_yzu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8164_start_ta_arc_yzu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8164_start_sr_arc_yzu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8164_start_sa_arc_yzu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8164_start_tr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec);
I16 _8164_start_ta_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,F64 Tdec);
I16 _8164_start_sr_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _8164_start_sa_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel, F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

I16 _8164_start_tr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _8164_start_ta_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _8164_start_tr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);
I16 _8164_start_ta_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel,F64 Tacc,F64 Tdec);

I16 _8164_start_sr_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _8164_start_sa_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _8164_start_sr_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);
I16 _8164_start_sa_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 StrVel,F64 MaxVel, F64 Tacc,F64 Tdec,F64 SVacc,F64 SVdec);

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visual Basic (Windows 95/NT)

B_8164_start_a_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_r_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_a_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_r_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_a_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_r_arc2 (ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

B_8164_start_tr_arc_xyu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8164_start_ta_arc_xyu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8164_start_sr_arc_xyu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8164_start_sa_arc_xyu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

B_8164_start_tr_arc_yzu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8164_start_ta_arc_yzu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

B_8164_start_sr_arc_yzu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer

B_8164_start_sa_arc_yzu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal SVacc As Double) As Integer

B_8164_start_tr_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_arc2 (ByVal CardNo As Integer, AxisArray As Double, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_tr_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_tr_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_ta_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double) As Integer

B_8164_start_sr_arc_xy (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double,

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_arc_xy (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sr_arc_zu (ByVal CardNo As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

B_8164_start_sa_arc_zu (ByVal CardNo As Integer, ByVal Cx As Double, ByVal Cy As Double, ByVal Ex As Double, ByVal Ey As Double, ByVal DIR As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double, ByVal Tdec As Double, ByVal SVacc As Double, ByVal SVdec As Double) As Integer

@ Argument

CardNo: Card number designated to perform linear interpolation
OffsetCx: X-axis offset to center
OffsetCy: Y-axis offset to center
OffsetEx: X-axis offset to end of arc
OffsetEy: Y-axis offset to end of arc
Cx: Specified X-axis absolute position of center
Cy: Specified Y-axis absolute position of center
Ex: Specified X-axis absolute position end of arc
Ey: Specified Y-axis absolute position end of arc
DIR: Specified direction of arc, CW:0 , CCW:1
StrVel: Starting velocity of a velocity profile in unit of pulse per second
MaxVel: Starting velocity of a velocity profile in unit of pulse per second
Tacc: Specified acceleration time in unit of second
Tdec: Specified deceleration time in unit of second
SVacc: Specified velocity interval in which S-curve acceleration is performed.
Note: SVacc = 0, for pure S-Curve
SVdec: Specified velocity interval in which S-curve deceleration is performed.
Note: SVdec = 0, for pure S-Curve
AxisArray: Array of axis number to perform interpolation.
Example: Int AxisArray[2] = {0,2}; // axis 0 & 2
Int AxisArray[2] = {1,3}; // axis 1 & 3
Note: AxisArray[0] must be smaller than AxisArray[1]

@ Return Code

ERR_NoError
ERR_SpeedError
ERR_AxisArrayErrot

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.9 Home Return Mode

@ Name

_8164_set_home_config – Set the configuration for home return.
_8164_home_move – Perform a home return move.
_8164_escape_home – Escape Home Function
_8164_home_search –Auto-Search Home Switch

@ Description

_8164_set_home_config:

Configures the home return mode, origin & index signal(EZ) logic, EZ count, and ERC output options for the home_move() function. Refer to Section 4.1.8 for the setting of home_mode control.

_8164_home_move:

This function will cause the axis to perform a home return move according to the **_8164_set_home_config()** function settings. The direction of movement is determined by the sign of velocity parameter (svel, mvel). Since the stopping condition of this function is determined by the *home_mode* setting, users should take care in selecting the initial moving direction. Users should also take care to handle conditions when the limit switch is touched or other conditions that are possible causing the axis to stop. Executing **v_stop()** function during **home_move()** can also cause the axis to stop.

_8164_escape_home:

After homing, use this function to leave the home switch

_8164_home_search:

Auto-Search Home Switch.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic,
        I16 ez_logic, I16 ez_count, I16 erc_out);
I16 _8164_home_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8164_escape_home(I16 AxisNo, F64 SrVel,F64 MaxVel,F64 Tacc);
I16 _8164_home_search(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc,
        F64 ORGOffset);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_home_config (ByVal AxisNo As Integer, ByVal home_mode As
        Integer, ByVal org_logic As Integer, ByVal ez_logic As Integer,
        ByVal ez_count As Integer, ByVal erc_out As Integer) As Integer
B_8164_home_move (ByVal AxisNo As Integer, ByVal StrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double) As Integer
B_8164_escape_home(ByVal AxisNo As Integer, ByVal SrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double);
B_8164_home_search (ByVal AxisNo As Integer, ByVal StrVel As Double,
        ByVal MaxVel As Double, ByVal Tacc As Double, ByVal ORGOffset
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

As Double) As Integer

@ Argument

AxisNo: Axis number designated to configure and perform home return
home_mode: Stopping modes for home return, 0~12
(Please refer to section 4.1.8)
org_logic: Action logic configuration for ORG
org_logic=0, active low;
org_logic=1, active high
EZ_logic: Action logic configuration for EZ
EZ_logic=0, active low;
EZ_logic=1, active high.
ez_count: 0~15 (Please refer to section 4.1.8)
erc_out: Set ERC output options.
erc_out =0, no erc out;
erc_out =1, erc out when home finishing
StrVel: starting velocity of a velocity profile in units of pulse per second
MaxVel: starting velocity of a velocity profile in units of pulse per second
Tacc: specified acceleration time in units of seconds
ORGOffset: The escape pulse amounts when home search touches the ORG singal

@ Return Code

ERR_NoError

6.10 Manual Pulser Motion

@ Name

_8164_set_pulser_iptmode - set the input signal modes of pulser
_8164_pulser_vmove – manual pulser v_move
_8164_pulser_pmove – manual pulser p_moce
_8164_pulser_home_move – manual pulser home move
_8164_set_pulser_ratio –Set manual pulser ratio for actual output pulse rate
_8164_pulser_r_line2 –Pulser mode for 2-axis linear interpolation
_8164_pulser_r_arc2 –Pulser mode for 2-axis arc interpolation

@ Description

_8164_set_pulser_iptmode:

This function is used to configure the input mode of manual pulser.

_8164_pulser_vmove:

With this command, the axis begins to move according to the manual pulse input. The axis will output one pulse when it receives one manual pulse, until the ***sd_stop*** or ***emg_stop*** command is written.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_pulser_pmove:

With this command, the axis begins to move according to the manual pulse input. The axis will output one pulse when it receives one manual pulse, until the ***sd_stop*** or ***emg_stop*** command is written or the output pulse number reaches the distance.

_8164_pulser_home_move:

With this command, the axis begins to move according to manual pulse input. The axis will output one pulse when it receives one manual pulse, until the ***sd_stop*** or ***emg_stop*** command is written or the home move finishes.

_8164_set_pulser_ratio:

Set manual pulse ratio for actual output pulse rate. The formula for manual pulse output rate is:

- Output Pulse Speed=(PA_PB Speed) * 4 * (PMG+1)*PDV/2048
- The PDV=0~10 Divide Factor
- The PMG=0~4 Multi Factor

_8164_set_pulser_ratio:

Pulser mode for 2-axis linear interpolation (relative mode only).

_8164_pulser_r_arc2:

Pulser mode for 2-axis arc interpolation (relative mode only)

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_pulser_iptmode(I16 AxisNo,I16 InputMode, I16 Inverse);
I16 _8164_pulser_vmove(I16 AxisNo, F64 SpeedLimit);
I16 _8164_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);
I16 _8164_pulser_home_move(I16 AxisNo, I16 HomeType, F64
      SpeedLimit);
I16 _8164_set_pulser_ratio(I16 AxisNo,I16 PDV, I16 PMG);
I16 _8164_pulser_r_line2(I16 CardNo,I16 *AxisArray, F64 DistX, F64 DistY,
      F64 SpeedLimit);
I16 _8164_pulser_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64
      OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_pulser_iptmode (ByVal AxisNo As Integer, ByVal InputMode
      As Integer, ByVal Inverse As Integer) As Integer
B_8164_pulser_vmove (ByVal AxisNo As Integer, ByVal SpeedLimit As
      Double) As Integer
B_8164_pulser_pmove (ByVal AxisNo As Integer, ByVal Dist As Double,
      ByVal SpeedLimit As Double) As Integer
B_8164_pulser_home_move (ByVal AxisNo As Integer, ByVal HomeType
      As Integer, ByVal SpeedLimit As Double) As Integer
B_8164_set_pulser_ratio(ByVal AxisNo As Integer, ByVal PDV As Integer,
      ByVal PMG As Integer) As Integer
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

B_8164_pulser_r_line2(ByVal CardNo As Integer, AxisArray As Integer, ByVal DistX As Double, ByVal DistY As Double, ByVal SpeedLimit As Double) As Integer

B_8164_pulser_r_arc2(ByVal CardNo As Integer, AxisArray As Integer, ByVal OffsetCx As Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel As Double) As Integer

@ Argument

AxisNo: Axis number designated to start manual move

InputMode: Setting of manual pulse input mode from the PA and PB pins
ipt_mode=0, 1X AB phase type pulse input.
ipt_mode=1, 2X AB phase type pulse input.
ipt_mode=2, 4X AB phase type pulse input.
ipt_mode=3, CW/CCW type pulse input.

Inverse: Reverse the moving direction from pulse direction
Inverse =0, no inverse
Inverse =1, Reverse moving direction

SpeedLimit: The maximum speed in a manual pulse move.
For example, if SpeedLimit is set to be 100pps, then the axis can move at fastest 100pps , even the input pulser signal rate is more then 100pps.

Dist: specified relative distance to move

HomeType: specified home move type
HomeType =0, Command Origin.(that means axis stops when command counter becomes '0')
HomeType =1, ORG pin.

PDV, PMG: Divide and Multi Factor.
PDV=0~10 Divide Factor
PMG=0~4 Multi Factor
The Output Pulse Speed=(PA_PB Speed) * 4 * (PMG+1)*PDV/2048

DistX: specified relative distance of axis 0 to move

DistY: specified relative distance of axis 1 to move

OffsetCx: X-axis offset from center

OffsetCy: Y-axis offset from center

OffsetEx: X-axis offset from end of arc

OffsetEy: Y-axis offset from end of arc

DIR: Specified direction of arc, CW:0 , CCW:1

SpeedLimit: Maximum tangential velocity in units of pulse per second

MaxVel: Maximum tangential velocity in units of pulse per second

@ Return Code

ERR_NoError
ERR_PulserHomeTypeError

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.11 Motion Status

@ Name

_8164_motion_done – Return the motion status

@ Description

_8164_motion_done:

Return the motion status of the 8164.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_motion_done(I16 AxisNo);
```

Visual Basic (Windows 95/NT)

```
B_8164_motion_done (ByVal AxisNo As Integer) As Integer
```

@ Argument

AxisNo: Axis number designated to start manual move

@ Return Value

0	Stop
1	Reserved
2	Reserved
3	Reserved
4	Wait for other axis
5	Wait ERC finished
6	Wait DIR Change
7	Backlash compensating
8	Wait PA/PB
9	In home special speed motion
10	In start velocity motion
11	In acceleration
12	In Max velocity motion
13	In deceleration
14	Wait INP
15	Other axis is still moving

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.12 Motion Interface I/O

@ Name

_8164_set_alm – Set alarm logic and operating mode
_8164_set_el – Set EL logic and operating mode
_8164_set_inp– Set Inp logic and operating mode
_8164_set_erc– Set ERC logic and timing
_8164_set_servo – Set state of general purpose output pin
_8164_set_sd – Set SD logic and operating mode

@ Description

_8164_set_alm_logic:

Set the active logic of the **ALARM** signal input from the servo driver. Two reacting modes are available when the **ALARM** signal is active.

_8164_set_el:

Set the reacting modes of the **EL** signal.

_8164_set_inp_logic:

Set the active logic of the **In-Position** signal input from the servo driver. Users can select whether they want to enable this function. It is disabled by default.

_8164_set_erc:

You can set the logic and on time of the ERC with this function.

_8164_set_servo:

You can set the ON-OFF state of the SVON signal with this function. The default value is 1(OFF), which means the SVON is open to GND.

_8164_set_sd:

Set the active logic, latch control, and operating mode of the **SD** signal input from a mechanical system. Users can select whether they want to enable this function. It is disabled by default.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);
I16 _8164_set_el(I16 AxisNo, I16 el_mode);
I16 _8164_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);
I16 _8164_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_on_time);
I16 _8164_set_servo(I16 AxisNo, I16 on_off);
I16 _8164_set_sd(I16 AxisNo, I16 enable, I16 sd_logic, I16 sd_latch, I16
    sd_mode);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_alm (ByVal AxisNo As Integer, ByVal alm_logic As Integer,
    ByVal alm_mode As Integer) As Integer
B_8164_set_el (ByVal AxisNo As Integer, ByVal el_mode As Integer) As
    Integer
B_8164_set_inp (ByVal AxisNo As Integer, ByVal inp_enable As Integer,
```


LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

ByVal inp_logic As Integer) As Integer
B_8164_set_erc (ByVal AxisNo As Integer, ByVal erc_logic As Integer, ByVal erc_on_time As Integer) As Integer
B_8164_set_servo (ByVal AxisNo As Integer, ByVal On_Off As Integer) As Integer
B_8164_set_sd (ByVal AxisNo As Integer, ByVal enable As Integer, ByVal sd_logic As Integer, ByVal sd_latch As Integer, ByVal sd_mode As Integer) As Integer

@ Argument

AxisNo: Axis number designated to configure
alm_logic: Setting of active logic for ALARM signals
alm_logic=0, active LOW.
alm_logic=1, active HIGH.
alm_mode: Reacting modes when receiving an ALARM signal.
alm_mode=0, motor immediately stops (default)
alm_mode=1, motor decelerates then stops.
el_mode: Reacting modes when receiving an EL signal.
el_mode=0, motor immediately stops (default)
el_mode=1, motor decelerates then stops.
inp_enable: INP function enabled/disabled
inp_enable=0, Disabled (default)
inp_enable=1, Enabled
inp_logic: Set the active logic for the INP signal
inp_logic=0, active LOW.
inp_logic=1, active HIGH.
erc_logic: Set the active logic for the ERC signal
erc_logic=0, active LOW.
erc_logic=1, active HIGH.
erc_on_time: Set the time length of ERC active
erc_on_time=012us
erc_on_time=1102us
erc_on_time=2409us
erc_on_time=31.6ms
erc_on_time=413ms
erc_on_time=552ms
erc_on_time=6104ms
on_off: ON-OFF state of SVON signal
on_off = 0 , ON
on_off = 1 , OFF
enable: Enable/disable the SD signal.
enable=0, Disabled (Default)
enable=1, Enabled
sd_logic: Set the active logic for the SD signal
sd_logic=0, active LOW.
sd_logic=1, active HIGH.
sd_latch: Set the latch control for the SD signal
sd_latch=0, do not latch.
sd_latch=1, latch.
sd_mode: Set the reacting mode of the SD signal
sd_mode=0, slow down only
sd_mode=1, slow down then stop

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Return Code

ERR_NoError

6.13 Motion I/O Monitoring

@ Name

_8164_get_io_status –Get all the motion I/O statuses of each 8164

@ Description

_8164_get_io_status:

Get all the I/O statuses for each axis. The definition for each bit is as follows:

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	+EL	Positive Limit Switch
3	-EL	Negative Limit Switch
4	ORG	Origin Switch
5	DIR	DIR output
6	Reserved	
7	PCS	PCS signal input
8	ERC	ERC pin output
9	EZ	Index signal
10	Reserved	
11	Latch	Latch signal input
12	SD	Slow Down signal input
13	INP	In-Position signal input
14	SVON	Servo-ON output status

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

I16 _8164_get_io_status(I16 AxisNo, U16 *io_sts);

Visual Basic (Windows 95/NT)

B_8164_get_io_status (ByVal AxisNo As Integer, io_sts As Integer) As Integer

@ Argument

AxisNo: Axis number for I/O control and monitoring

***io_status**: I/O status word. “1” is ON and “0” is OFF. ON/OFF state is read based on the corresponding set logic.

@ Return Code

ERR_NoError

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.14 Interrupt Control

@ Name

_8164_int_control – Enable/Disable INT service
_8164_set_int_factor – Set INT factor
_8164_int_enable – Enable event (For Window only)
_8164_int_disable – Disable event (For Window only)
_8164_get_int_status – Get INT Status (For Window only)
_8164_link_interrupt – Set link to interrupt call back function (For Window only)
_8164_get_int_type – Get INT type (For DOS only)
_8164_enter_isr – Enter interrupt service routine (For DOS only)
_8164_leave_isr – Leave interrupt service routine (For DOS only)
_8164_get_event_int – Get event status (For DOS only)
_8164_get_error_int – Get error status (For DOS only)
_8164_get_irq_status – Get IRQ status (For DOS only)
_8164_not_my_irq – Not My IRQ (For DOS only)
_8164_isr0~9, a, b – Interrupt service routine (For DOS only)
_8164_set_axis_stop_int – enable axis stop int
_8164_mask_axis_stop_int – mask axis stop int

@ Description

_8164_int_control:

This function is used to enable interrupt generating to host PC.

_8164_set_int_factor:

This function allows users to select factors to initiate the event int. The error can never be masked once the interrupt service is turned on by _8164_int_control().

The INT status of 8164 is composed of two independent parts: **error_int_status** and **event_int_status**. The event_*int*_status recordes the motion and comparator event under **normal operation**, and this kind of INT status can be masked by **_8164_set_int_factor()**. The error_int_status is for abnormal stop of the 8164, for example: EL, ALM …etc. This kind of INT cannot be masked. Below is the definition of these two int_status. By setting the relative bit as 1, 8164 can generate INT signal to host PC.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Bit	Description
0	Normal Stop
1	Next command Starts
2	Command pre-register 2 is empty
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	(Reserved)
11	General Comparator compared
12	Trigger Comparator compared
13	(Reserved)
14	Counter Latched for axis2,3
15	ORG Input and Latched
16	SD on
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. Start on
20~31	(Reserved)

_8164_int_enable : (For **Window** only.)

This function is used to enable the Windows INT event.

_8164_int_disable: (For **Window** only.)

This function is used to disable the Windows INT event.

_8164_get_int_status: (For **Window** only.)

This function allows user to identify what caused the interrupt signal. After the value is obtained, the status register will be cleared to 0. The return value is two 32 bits unsigned integers. The first one is for error_int_status, which is not able to mask by _8164_set_int_factor(). The definition for bit of error_int_status is as following:

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

error_int_status : can not be masked	
Bit	Interrupt Factor
0	+Soft Limit on and stop
1	-Soft Limit on and stop
2	(Reserved)
3	General Comparator on and Stop
4	(Reserved)
5	+End Limit on and stop
6	-End Limit on and stop
7	ALM happen and stop
8	CSTP, Sync. Stop on and stop
9	CEMG, Emergency on and stop
10	SD on and slow down to stop
11	(Reserved)
12	Interpolation Error and stop
13	Other Axis stop on Interpolation
14	Pulser input buffer overflow and stop
15	Interpolation counter overflow
16	Encoder input signal error
17	Pulser input signal error
11~31	(Reserved)

The second is for event_int_status, which can be masked by _8164_set_int_factor(). See table below:

event_int_status: can be masked by function call ***_8164_int_*factor()**	
Bit	Description
0	Normal Stop
1	Next command Starts
2	Command pre-register 2 is empty
3	(Reserved)
4	Acceleration Start
5	Acceleration End
6	Deceleration Start
7	Deceleration End
8	(Reserved)
9	(Reserved)
10	(Reserved)
11	General Comparator compared
12	Trigger Comparator compared
13	(Reserved)
14	Counter Latched for axes 2 and 3
15	ORG Input and Latched
16	SD on
17	(Reserved)
18	(Reserved)
19	CSTA, Sync. Start on
20~30	(Reserved)
31	Axis Stop Interrupt

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_link_interrupt: (For Window **only.)**

This function is used to link to the interrupt call back function.

_8164_get_int_type: (This function is for **DOS** only **)**

This function is used to detect which kind of INT occurred.

_8164_enter_isr: (This function is for **DOS** only **)**

This function is used to inform the system that the process is now entering interrupt service routine.

_8164_leave_isr: (**This function is for** DOS **only**)

This function is used to inform the system that the process is now leaving interrupt service routine.

_8164_get_event_int: (This function is for **DOS** only **)**

This function is used to get event_int_status.

_8164_get_error_int: (This function is for **DOS** only **)**

This function is used to get error_int_status.

_8164_get_irq_status: (This function is for **DOS** only **)**

This function allows user to confirm if the designated card generates the INT signal to host PC.

_8164_not_my_irq: (This function is for **DOS** only **)**

This function must be called after the designated card generates the INT signal to host PC.

_8164_isr0, _8164_isr1, _8164_isr2, _8164_isr3, ….. _8164_isr9, _8164_isra, _8164_isrb: (These functions are for **DOS** only **)**

Individual Interrupt service routine for cards 0-11.

_8164_set_axis_stop_int:

This function will enable an axis stop interrupt. Once it is enabled, the interrupt will happen no matter it is a normal stop or error stop. This interrupt condition can be turned on or off accompanied with every motion command by setting _8164_mask_axis_stop_int(). This kind of interrupt condition is different from _8164_set_int_factor(). It can be controlled in each motion command, which is very useful in continuous motion when users only need a final command interrupt. The interrupt status is in “event interrupt status” bit 31.

_8164_mask_axis_stop_int:

This function will affect the axis stop interrupt factor which is set by _8164_set_axis_stop_int().

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Syntax

C/C++ (DOS)

```
I16 _8164_int_control(U16 cardNo, U16 intFlag );
I16 _8164_set_int_factor(I16 AxisNo, U32 int_factor );
I16  _8164_get_int_type(I16 AxisNo, U16 *int_type);
I16  _8164_enter_isr(I16 AxisNo);
I16  _8164_leave_isr(I16 AxisNo);
I16  _8164_get_event_int(I16 AxisNo, U32 *event_int);
I16  _8164_get_error_int(I16 AxisNo, U32 *error_int);
I16  _8164_get_irq_status(U16 cardNo, U16 *sts);
I16  _8164_not_my_irq(I16 CardNo);
void interrupt _8164_isr0 (void);
void interrupt _8164_isr1 (void);
void interrupt _8164_isr2 (void);
void interrupt _8164_isr3 (void);
void interrupt _8164_isr4 (void);
void interrupt _8164_isr5 (void);
void interrupt _8164_isr6 (void);
void interrupt _8164_isr7 (void);
void interrupt _8164_isr8 (void);
void interrupt _8164_isr9 (void);
void interrupt _8164_isra (void);
void interrupt _8164_isrb (void);
```

C/C++ (Windows 95/98/NT)

```
I16 _8164_int_control(U16 cardNo, U16 intFlag );
I16 _8164_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _8164_int_enable(I16 CardNo, HANDLE *phEvent);
I16 _8164_int_disable(I16 CardNo);
I16 _8164_get_int_status(I16 AxisNo, U32 *error_int_status, U32
      *event_int_status );
I16 _8164_link_interrupt(I16 CardNo, void ( __stdcall *callbackAddr)(I16
      IntAxisNoInCard));
I16 _8164_set_axis_stop_int(I16 AxisNo, I16 axis_stop_int);
I16 _8164_mask_axis_stop_int(I16 AxisNo, I16 int_disable);
```

Visual Basic (Windows 95/NT)

```
B_8164_int_control (ByVal CardNo As Integer, ByVal intFlag As Integer) As
      Integer
B_8164_set_int_factor (ByVal AxisNo As Integer, ByVal int_factor As Long)
      As Integer
B_8164_int_enable (ByVal CardNo As Integer, phEvent As Long) As
      Integer
B_8164_int_disable (ByVal CardNo As Integer) As Integer
B_8164_get_int_status (ByVal AxisNo As Integer, error_int_status As Long,
      event_int_status As Long) As Integer
B_8164_link_interrupt (ByVal CardNo As Integer, ByVal lpCallBackProc As
      Long) As Integer
B_8164_mask_axis_stop_int (ByVal AxisNo As Integer, ByVal int_disable
      As Integer) As Integer
B_8164_set_axis_stop_int (ByVal AxisNo As Integer, ByVal axis_stop_int
      As Integer) As Integer
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Argument

cardNo: card number 0,1,2,3…
AxisNo: axis number 0,1,2,3,4…
intFlag: int flag, 0 or 1 (0: Disable, 1:Enable)
int_factor: interrupt factor, refer to previous table
***int_type:** Interrupt type, (1: error int, 2: event int, 3: both happened)
***event_int:** event_int_status, , refer to previous table
***error_int:** error_int_status, refer to previous table
***sts:** (0: not this card's IRQ, 1: this card's IRQ)
***phEvent:** event handler (Windows)
***error_int_status**: refer to previous table
***event_int_status**: refer to previous table
int_disable: (0:make axis stop interrupt active, 1:make axis stop interrupt in-active)
axis_stop_int: (0: disable axis stop interrupt factor, 1: enable axis stop interrupt factor)

@ Return Code

ERR_NoError
ERR_EventNotEnableYet
ERR_LinkIntError
ERR_CardNoErrot

6.15 Position Control and Counters

@ Name

_8164_get_position – Get the value of feedback position counter
_8164_set_position – Set the feedback position counter
_8164_get_command – Get the value of command position counter
_8164_set_command – Set the command position counter
_8164_get_error_counter – Get the value of position error counter
_8164_reset_error_counter – Reset the position error counter
_8164_get_general_counter – Get the value of general counter
_8164_set_general_counter – Set the general counter
_8164_get_target_pos – Get the value of target position recorder
_8164_reset_target_pos – Reset target position recorder
_8164_get_rest_command – Get remaining pulse till end of motion
_8164_check_rdp – Get the ramping down point data

@ Description

_8164_get_position():

This function is used to read the feedback position counter value. Note that this value has already been processed by the move ratio. If the move ratio is 0.5, than the value read will be twice as the counter value. The source of the feedback counter is selectable by the function **_8164_set_feedback_src()** to be external EA/EB or pulse output of 8164.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_set_position():

This function is used to change the feedback position counter to the specified value. Note that the value to be set will be processed by the move ratio. If move ratio is 0.5, then the set value will be twice as given value.

_8164_get_command():

This function is used to read the value of the command position counter. The source of the command position counter is the pulse output of the 8164.

_8164_set_command():

This function is used to change the value of the command position counter.

_8164_get_error_counter():

This function is used to read the value of the position error counter.

_8164_reset_error_counter():

This function is used to clear the position error counter.

_8164_get_general_counter():

This function is used to read the value of the general counter.

_8164_set_general_counter():

This function is used to set the counting source of and change the value of general counter (By default, the source is pulse input).

_8164_get_target_pos():

This function is used to read the value of the target position recorder. The target position recorder is maintained by the 8164 software driver. It records the position to settle down for current running motion.

_8164_reset_target_pos():

This function is used to set new value for the target position recorder. It is necessary to call this function when home return completes, or when a new feedback counter value is set by function _8164_set_position().

_8164_get_rest_command():

This function is used to read remaining pulse counts until the end of the current motion.

_8164_check_rdp():

This function is used to read the ramping down point data. The ramping down point is the position where deceleration starts. The data is stored as a pulse count, and it causes the axis start to decelerate when remaining pulse count reach the data.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

```
I16 _8164_get_position(I16 AxisNo, F64 *pos);
I16 _8164_set_position(I16 AxisNo, F64 pos);
I16 _8164_get_command(I16 AxisNo, I32 *cmd);
I16 _8164_set_command(I16 AxisNo, I32 cmd);
I16 _8164_get_error_counter(I16 AxisNo, I16 *error_counter);
I16 _8164_reset_error_counter(I16 AxisNo);
I16 _8164_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _8164_set_general_counter(I16 AxisNo,I16 CntSrc, F64 CntValue);
I16 _8164_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _8164_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _8164_get_rest_command(I16 AxisNo, I32 *rest_command);
I16 _8164_check_rdp(I16 AxisNo, I32 *rdp_command);
```

Visual Basic (Windows 95/NT)

```
B_8164_get_position (ByVal AxisNo As Integer, Pos As Double) As Integer
B_8164_set_position (ByVal AxisNo As Integer, ByVal Pos As Double) As
    Integer
B_8164_get_command (ByVal AxisNo As Integer, cmd As Long) As Integer
B_8164_set_command (ByVal AxisNo As Integer, ByVal cmd As Long) As
    Integer
B_8164_get_error_counter (ByVal AxisNo As Integer, error_counter As
    Integer) As Integer
B_8164_reset_error_counter (ByVal AxisNo As Integer) As Integer
B_8164_get_general_counter (ByVal AxisNo As Integer, CntValue As
    Double) As Integer
B_8164_set_general_counter (ByVal AxisNo As Integer, ByVal CntSrc As
    Integer, ByVal CntValue As Double) As Integer
B_8164_get_target_pos (ByVal AxisNo As Integer, Pos As Double) As
    Integer
B_8164_reset_target_pos (ByVal AxisNo As Integer, ByVal Pos As Double)
    As Integer
B_8164_get_rest_command (ByVal AxisNo As Integer, rest_command As
    Long) As Integer
B_8164_check_rdp (ByVal AxisNo As Integer, rdp_command As Long) As
    Integer
```

@ Argument

AxisNo: Axis number
Pos, *Pos: Feedback position counter value,
range: -134217728~134217727
cmd, *cmd: Command position counter value,
range: -134217728~134217727
error_counter, *error_counter: Position error counter value,
range: -32768~32767
T_pos, *T_pos: Target position recorder value,
T_ range: -134217728~134217727
CntValue, * CntValue: General counter value,
range: -134217728~134217727
rest_command, *rest_command: Rest pulse count till end,
range: -134217728~134217727

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

rdp_command, *rdp_command: Ramping down point data
range: 0~167777215
CntSrc: Source of general counter
0 : command
1: EA/EB
2: PA/PB (Default)
3: CLK/2

@ Return Code

ERR_NoError
ERR_PosOutofRange

6.16 Position Compare and Latch

@ Name

_8164_set_ltc_logic – Set the LTC logic
_8164_get_latch_data – Get latched counter data
_8164_set_soft_limit – Set soft limit
_8164_enable_soft_limit – Enable soft limit function
_8164_disable_soft_limit – Disable soft limit function
_8164_set_error_counter_check – Step-losing detection setup
_8164_set_general_comparator – Set general-purposed comparator
_8164_set_trigger_comparator – Set trigger comparator
_8164_set_trigger_type – Set the trigger output type
_8164_check_compare_data – Check current comparator data
_8164_check_compare_status – Check current comparator status
_8164_set_auto_compare – Set comparing data source for auto loading
_8164_build_compare_function – Build compare data via constant interval
_8164_build_compare_table – Build compare data via compare table
_8164_cmp_v_change – Speed change by comparator

@ Description

_8164_set_ltc_logic():

This function is used to set the logic of the latch input. This function is applicable only for last two axes in every 8164 card.

_8164_get_latch_data():

After the latch signal arrived, this function is used to read the latched value of counters.

_8164_set_soft_limit():

This function is used to set the soft limit value .

_8164_enable_soft_limit(),_8164_disable_soft_limit():

These two functions are used to enable/disable the soft limit function. Once enabled, the action of soft limit will be exactly the same as physical limit.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_set_error_counter_check():

This function is used to enable the step losing checking facility. By giving a tolerance value, the 8164 will generate an interrupt (event_int_status , bit 10) when position error counter exceed tolerance.

_8164_set_general_comparator():

This function is used to set the source and comparing value for the general comparator. When the source counter value reached the comparing value, the 8164 will generate an interrupt (event_int_status , bit 11).

_8164_set_trigger_comparator():

This function is used to set the comparing method and value for the trigger comparator. When the feedback position counter value reaches the comparing value, the 8164 will generate trigger a pulse output via **CMP** and an interrupt (event_int_status , bit 12) will also be sent to host PC. If _8164_set_auto_compare is used, then the comparing value set by this function will be ignored automatically. *Note: it is applicable only for first two axes in every 8164 card.*

_8164_set_trigger_type():

This function is used to set the trigger output mode

In hardware version A2, it is used for setting the output pulse as a one shot or constant on.

In hardware version A3, it is used for setting the output pulse as normal high or normal low.

_8164_check_compare_data():

This function is used to get current comparing data of the designated comparator.

_8164_check_compare_status():

This function is used to get the status of all comparators. When some comparators come into existence, the relative bits of cmp_sts will become '1,' otherwise '0.'

_8164_set_auto_compare():

This function is used to set the comparing data source of the trigger comparator. The source can be either a function or a table.

_8164_build_compare_function():

This function is used to build a comparing function by defining the start / end point and interval. There is no limitation on the max number of comparing data. It will automatically load a final point after user's end point. That is, (end point + Interval x total points) x move ratio.

Note: Please turn off all interrupt functions when triggering is running

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

_8164_build_compare_table():

This function is used to build a comparing table by defining a data array. The size of array is limited to 1024 when using RAM mode.

Note: Please turn off all interrupt functions when triggering is running

_8164_cmp_v_change():

This function is used to setup the comparator velocity change function. It is a V_change function but acts when a general comparator comes into existence. When this function is issued, the parameter "CmpAction" of **_8164_set_general_comparator()** must be set '3.' The compare data is also set by **_8164_set_general_comparator()**. While the remain distance, the compare point's velocity, the new velocity, and the acceleration time are set by **_8164_cmp_v_change()**.

@ Syntax

C/C++ (DOS, Windows 95/98/NT)

```
I16 _8164_set_ltc_logic(I16 AxisNo_2or3, I16 ltc_logic);
I16 _8164_get_latch_data(I16 AxisNo, I16 LatchNo, F64 *Pos);
I16 _8164_set_soft_limit(I16 AxisNo, I32 PLimit, I32 NLimit);
I16 _8164_disable_soft_limit(I16 AxisNo);
I16 _8164_enable_soft_limit(I16 AxisNo, I16 Action);
I16 _8164_set_error_counter_check(I16 AxisNo, I16 Tolerance, I16
      On_Off);
I16 _8164_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16
      CmpMethod, I16 CmpAction, F64 Data);
I16 _8164_set_trigger_comparator(I16 AxisNo, I16 CmpSrc,  I16
      CmpMethod, F64 Data);
I16 _8164_set_trigger_type(I16 AxisNo, I16 TriggerType);
I16 _8164_check_compare_data(I16 AxisNo, I16 CompType, F64 *Pos);
I16 _8164_check_compare_status(I16 AxisNo, U16 *cmp_sts);
I16 _8164_set_auto_compare(I16 AxisNo ,I16 SelectSrc);
I16 _8164_cmp_v_change(I16 AxisNo, F64 Res_dist, F64 oldvel, F64
      newvel, F64 AccTime)
```

C/C++ (Windows 95/98/NT)

```
I16 _8164_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64
      Interval, I16 Device);
I16 _8164_build_compare_table(I16 AxisNo, F64 *TableArray, I16 Size, I16
      Device);
```

C/C++ (Dos)

```
I16 _8164_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64
      Interval);
I16 _8164_build_compare_table(I16 AxisNo, F64 *TableArray, I16 Size);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_ltc_logic (ByVal AxisNo As Integer, ByVal ltc_logic As Integer)
      As Integer
B_8164_get_latch_data (ByVal AxisNo As Integer, ByVal Counter As
      Integer, Pos As Double) As Integer
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

B_8164_set_soft_limit (ByVal AxisNo As Integer, ByVal PLimit As Long, ByVal NLimit As Long) As Integer
B_8164_disable_soft_limit (ByVal AxisNo As Integer) As Integer
B_8164_enable_soft_limit (ByVal AxisNo As Integer, ByVal Action As Integer) As Integer
B_8164_set_error_counter_check (ByVal AxisNo As Integer, ByVal Tolerance As Integer, ByVal On_Off As Integer) As Integer
B_8164_set_general_comparator (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal CmpAction As Integer, ByVal Data As Double) As Integer
B_8164_set_trigger_comparator (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, ByVal CmpMethod As Integer, ByVal Data As Double) As Integer
B_8164_set_trigger_type (ByVal AxisNo As Integer, ByVal TriggerType As Integer) As Integer
B_8164_check_compare_data (ByVal AxisNo As Integer, ByVal CompType As Integer, Pos As Double) As Integer
B_8164_check_compare_status (ByVal AxisNo As Integer, cmp_sts As Integer) As Integer
B_8164_set_auto_compare (ByVal AxisNo As Integer, ByVal SelectSrc As Integer) As Integer
B_8164_build_compare_function (ByVal AxisNo As Integer, ByVal Start As Double, ByVal End As Double, ByVal Interval As Double, ByVal Device As Integer) As Integer
B_8164_build_compare_table (ByVal AxisNo As Integer, TableArray As Double, ByVal Size As Integer, ByVal Device As Integer) As Integer
B_8164_cmp_v_change(ByVal AxisNo, ByVal Res_dist as Double, ByVal oldvel as Double, ByVal newvel as Double, ByVal AccTime as Double)

@ Argument

AxisNo_2or3: Axis number, for last two axes in one card
Itc_logic: 0 means active low, 1 means active high
AxisNo: Axis number
LatchNo: Specified Counter to latch
LatchNo = 1 , Command counter
LatchNo = 2 , Feedback counter
LatchNo = 3 , Error Counter
LatchNo = 4 , General Counter
Pos: Latched counter value,
PLimit: Soft limit value, positive direction
NLimit: Soft limit value, negative direction
Action: The reacting method of soft limit
Action =0, INT only
Action =1, Immediately stop
Action =2, slow down then stop
Action =3, reserved
Tolerance: The tolerance of step-losing detection
On_Off: Enable / Disable step-losing detection
On_Off =0, Disable
On_Off =1, Enable
CmpSrc: The comparing source counter

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

CmpSrc =0, Command Counter
CmpSrc =1, Feedback Counter
CmpSrc =2, Error Counter
CmpSrc =3, General Counter

CmpMethod: The comparing method
CmpMethod =0, No compare
CmpMethod =1, CmpValue=Counter (Directionless)
CmpMethod =2, CmpValue=Counter (+Dir)
CmpMethod =3, CmpValue=Counter (-Dir)
CmpMethod =4, CmpValue>Counter
CmpMethod =5, CmpValue<Counter

CmpAction: The reacting mode when comparison comes into exist
CmpAction =0, INT only
CmpAction =1, Immediate stop
CmpAction =2, Slow down then stop
CmpAction =3, Speed change

Data: Comparing value,

TriggerType: Selection of type of trigger output mode
Hardware Version A2
TriggerType =0, one shoot (default)
TriggerType =1, constant high
Hardware Version A3
TriggerType =0, normal high (default)
TriggerType =1, normal low

CompType: Selection of type of comparator
CompType =1, + Soft Limit
CompType =2, -Soft Limit
CompType =3, Error Counter Comparator Value
CompType =4, General Comparator Value
CompType =5, Trigger Output Comparator Value

cmp_sts: status of comparator

Bit	Meaning
0	+Softlimit On
1	-SoftLimit On
2	Error counter comparator On
3	General comparator On
4	Trigger comparator On (for 0 , 1 axis only)

SelectSrc: The comparing data source
SelectSrc =0, disable auto compare
SelectSrc =1, use FIFO

Start: Start point of compare function
End: End point of compare function
Interval: Interval of compare function
TableArray: Array of comparing data
Size: Size of table array
Device: Selection of reload device for comparator data
Device =1, FIFO
Res_dist: The remaining distance from the compare point. After comparison, the original target position will be ignored, and the axis

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

will keep moving the Res_dist.

oldvel: The velocity at compare point. User must specify it manually.
newvel: The new velocity.
AccTime: The acceleration time.

@ Return Code

ERR_NoError
ERR_CompareNoError
ERR_CompareMethodError
ERR_CompareAxisError
ERR_CompareTableSizeError
ERR_CompareFunctionError
ERR_CompareTableNotReady
ERR_CompareLineNotReady
ERR_HardwareCompareAxisWrong
ERR_AutocompareSourceWrong
ERR_CompareDeviceTypeError

6.17 Continuous motion

@ Name

_8164_set_continuous_move – toggle continuous motion sequence flags
_8164_check_continuous_buffer – check if the command register buffer is empty

@ Description

_8164_set_continuous_move():

This function is necessary before and after continuous motion command sequences.

_8164_check_continuous_buffer():

This function is used to detect if the command pre-register is empty or not. Once the command pre-register is empty, users may write the next motion command into it. Otherwise, the new command will overwrite the previous command in the 2nd command pre-register.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_continuous_move(I16 AxisNo, I16 conti_flag);
I16 _8164_check_continuous_buffer(I16 AxisNo);
```

Visual Basic (Windows 95/NT)

```
B_8164_set_continuous_move (ByVal AxisNo As Integer, ByVal conti_flag
        As Integer) As Integer
B_8164_check_continuous_buffer (ByVal AxisNo As Integer) As Integer
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

@ Argument

AxisNo: axis number designated
conti_flag: Flag for continuous motion
conti_flag = 0, declare continuous motion sequence is finished
conti_flag = 1, declare continuous motion sequence is started

@ Return Value

ERR_NoError

Return value of _8164_check_continuous_buffer():
Hardware version bit 12=0
0: command register 2 is empty
1: command register 2 is in-use
Return value of _8164_check_continuous_buffer():
Hardware version bit 12=1
0: all command registers are empty
1: command register is in-use
2: command register 1 is in-use
3: command register 2 is in-use

6.18 Multiple Axes Simultaneous Operation

@ Name

_8164_set_tr_move_all – Multi-axis simultaneous operation setup.
_8164_set_ta_move_all – Multi-axis simultaneous operation setup.
_8164_set_sr_move_all – Multi-axis simultaneous operation setup.
_8164_set_sa_move_all – Multi-axis simultaneous operation setup.
_8164_start_move_all – Begin a multi-axis trapezoidal profile motion
_8164_stop_move_all –Simultaneously stop Multi-axis motion
_8164_set_sync_option – Other sync. motion setting
_8164_set_sync_stop_mode – Setting the stop mode of CSTOP signal

@ Description

Theses functions are related to simultaneous operations of multi-axes, even in different cards. The simultaneous multi-axis operation means to start or stop moving specified axes at the same time. The axes moved are specified by the parameter **"AxisArray,"** and the number of axes are defined by parameter "***TotalAxes"*** in **_8164_set_tr_move_all()**.

When properly setup with **_8164_set_xx_move_all()**, the function **_8164_start_move_all()** will cause all specified axes to begin a trapezoidal relative movement, and _8164_stop_move_all() will stop them. Both functions guarantee that motion Start/Stop on all specified axes are at the same time. Note that it is necessary to make connections according to Section 3.14 on CN4 if these two functions are needed.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

The following code demos how to utilize these functions. This code moves axis 0 and axis 4 to distance 8000.0 and 120000.0 respectively. If we choose velocities and accelerations that are proportional to the ratio of distances, then the axes will arrive at their endpoints at the same time.

```
int main()
{
        I16 axes[2] = {0, 4};
        F64 dist[2] = {8000.0, 12000.0},
        str_vel[2]={0.0, 0.0},
        max_vel[2]={4000.0, 6000.0},
        Tacc[2]={0.04, 0.06},
        Tdec[2]= {0.04, 0.06};

        _8164_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
        _8164_start_move_all(axes[0]);

        returnERR_NoError;
}
```

_8164_set_sync_option()

It lets two or more different command groups start at the same time. For example, if you want a 2-axis linear interpolation and a 1-axis single motion to start at the same time, you can turn on this option before the command starts. Besides, this function can also be used when waiting for another command's finish signal before starting. For example, axis1 must start after axis2 is done.

_8164_set_sync_stop_mode()

It provides two options for stop types: One is immediately stop and the other is slow down to stop. When the _8164_stop_move_all() or CSTOP signal is used, the axes will stop according to this setting.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8164_set_sa_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA,
        F64 *SVdecA);
I16 _8164_set_ta_move_all(I16 TotalAx, I16 *AxisArray, F64 *PosA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8164_set_sr_move_all(I16 TotalAx, I16 *AxisArray, F64 *DistA, F64
        *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA,
        F64 *SVdecA);
I16 _8164_start_move_all(I16 FirstAxisNo);
I16 _8164_stop_move_all(I16 FirstAxisNo);
I16 _8164_set_sync_option(I16 AxisNo, I16 sync_stop_on, I16
        cstop_output_on, I16 sync_option1, I16 sync_option2);
I16 _8164_set_sync_stop_mode(I16 AxisNo, I16 stop_mode);
```


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Visual Basic (Windows 95/NT)

B_8164_set_tr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, DistA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double);

B_8164_set_sa_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, PosA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double, SVaccA As double, SVdecA As Double);

B_8164_set_ta_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, PosA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double);

B_8164_set_sr_move_all(ByVal TotalAxes As Integer, AxisArray As Integer, DistA As Double, StrVelA As double, MaxVelA As double, TaccA As double, TdecA As double, SVaccA As double, SVdecA As Double);

B_8164_start_move_all(ByVal FirstAxisNo As Integer);

B_8164_stop_move_all(ByVal FirstAxisNo As Integer);

B_8164_set_sync_option (ByVal AxisNo As Integer, ByVal sync_stop_on As Integer, ByVal cstop_output_on As Integer, ByVal sync_option1 As Integer, ByVal sync_option2 As Integer) As Integer

B_8164_set_sync_stop_mode (ByVal AxisNo As Integer, ByVal stop_mode As Integer) As Integer

@ Argument

TotalAxes: Number of axes for simultaneous motion, 1~48.

* **AxisArray**: Specified axes number array designated to move.
* **DistA**: Specified position array in units of pulse
* **StrVelA:** Starting velocity array in units of pulse per second
* **MaxVelA :** Maximum velocity array in units of pulse per second
* **TaccA:** Acceleration time array in units of seconds
* **TdecA:** Deceleration time array in units of seconds
* **SVaccA**: Specified velocity interval array in which S-curve acceleration is performed.
* **SVdecA**: specified velocity interval array in which S-curve deceleration is performed.

FirstAxisNo: the first element in AxisArray.

Sync_stop_on: Axis will stop if the CSTOP signal is on

Cstop_output_on: CSTOP signal will output with an abnormal stop (ALM,EL..etc)

Sync_option1: Choose command start type:
0: default (immediate start)
1: waiting _8164_start_move_all() or CSTA signal
2: Reserved
3: Check Sync_option2's condition to start

Sync_option2: For example:
0: default (useless)
1: after Axis0 stops
2: after Axis1 stops
4: after Axis2 stops
8: after Axis3 stops
5: after Axis0 and Axis2 stop
15: Axis0~Axis3 stop

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

stop_mode: 0: immediate stop
1: slow down to stop

@ Return Code

ERR_NoError
ERR_SpeedError

6.19 General-purposed TTL output (PCI-8164 Only)

@ Name

_8164_d_output – Digital Output
_8164_get_dio_status – Get DIO status

@ Description

_8164_d_output():

Set the on_off status for general-purposed TTL Digital output pin.

_8164_get_dio_status():

Read status of all digital output pin.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_d_output(I16 CardNo, I16 Ch_No, I16 value);
I16 _8164_get_dio_status(I16 CardNo, U16 *dio_sts);
```

Visual Basic (Windows 95/NT)

```
B_8164_d_output (ByVal CardNo As Integer, ByVal Ch_No As Integer,
        ByVal value As Integer) As Integer
B_8164_get_dio_status (ByVal CardNo As Integer, dio_sts As Integer) As
        Integer
```

@ Argument

CardNo: Designated card number
Ch_No: Designated channel number 0~5
Value: On-Off Value for output
Value =0, output OFF
Value =1, output ON
dio_status: Digital output status
bit0~bit5 for channel 0~5 , respectively

@ Return Value

ERR_NoError
ERR_DioNoError

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

6.20 General-purposed DIO (MPC-8164 Only)

@ Name

_8164_write_do – Digital Output
_8164_read_di – Digital Input

@ Description

_8164_write_do():

Output an 8-bit value once to control 8 output channels.

_8164_read_di():

Read back an 8-bit value once from 8 input channels.

@ Syntax

C/C++ (DOS, Windows 95/NT)

```
I16 _8164_write_do(I16 CardNo, U16 Value);
U16 _8164_read_di(I16 CardNo);
```

Visual Basic (Windows 95/NT)

```
B_8164_write_do (ByVal CardNo As Integer, ByVal value As Integer) As
        Integer
B_8164_read_di (ByVal CardNo As Integer) As Integer
```

@ Argument

CardNo: Designated card number
Value: Value for output
Bit =0, output OFF
Bit =1, output ON

@ Return Value

ERR_NoError
Digital Input Value for 8 channels

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7

Connection Example

This chapter shows some connection examples between the 8164 and servo drivers and stepping drivers.

7.1 General Description of Wiring

CN1: Receives +24V power from the external power supply. (PCI-8164 Only)

CN2: Main connection between the PCI-8164 and the pulse input servo driver or stepping driver.

CN3: Receives pulse commands from manual pulse in PCI-8164.

General Purpose DIO for MPC-8164

CN4: Connector for simultaneously start or stop of multiple PCI-8164 cards.

CN5: TTL digital output for PCI-8164

Figure 8 illustrates how to integrate the PCI-8164 with a physical system.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

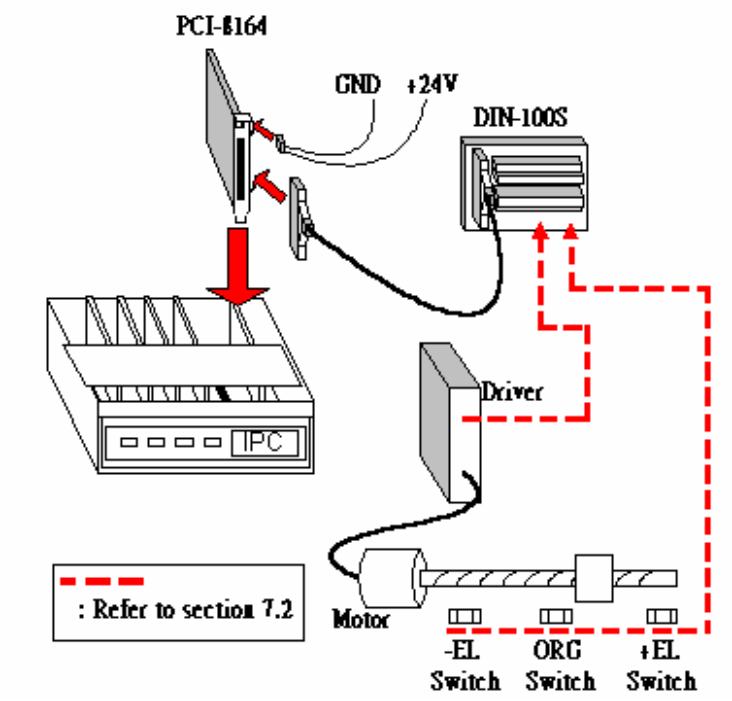


Figure 8: System Integration with PCI-8164

7.2 Connection Example with Servo Driver

This example will illustrate the connection between a ***Panasonic Servo Driver*** and the ***8164***. Figure 9 shows the wiring diagram.

Note that:

1. For convenience, the drawing shows connections for one axis only.
2. Default pulse output mode is **OUT/DIR**. Default input mode is 1X **AB phase**. Other modes can be set using the available software functions.
3. Most general purpose servomotor drivers can operate in ***Torque Mode, Velocity Mode, or Position* Mode.** To connect with the 8164, users should set the operating mode to **Position Mode**.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Wiring of PCI-8164 with Panasonic MSD

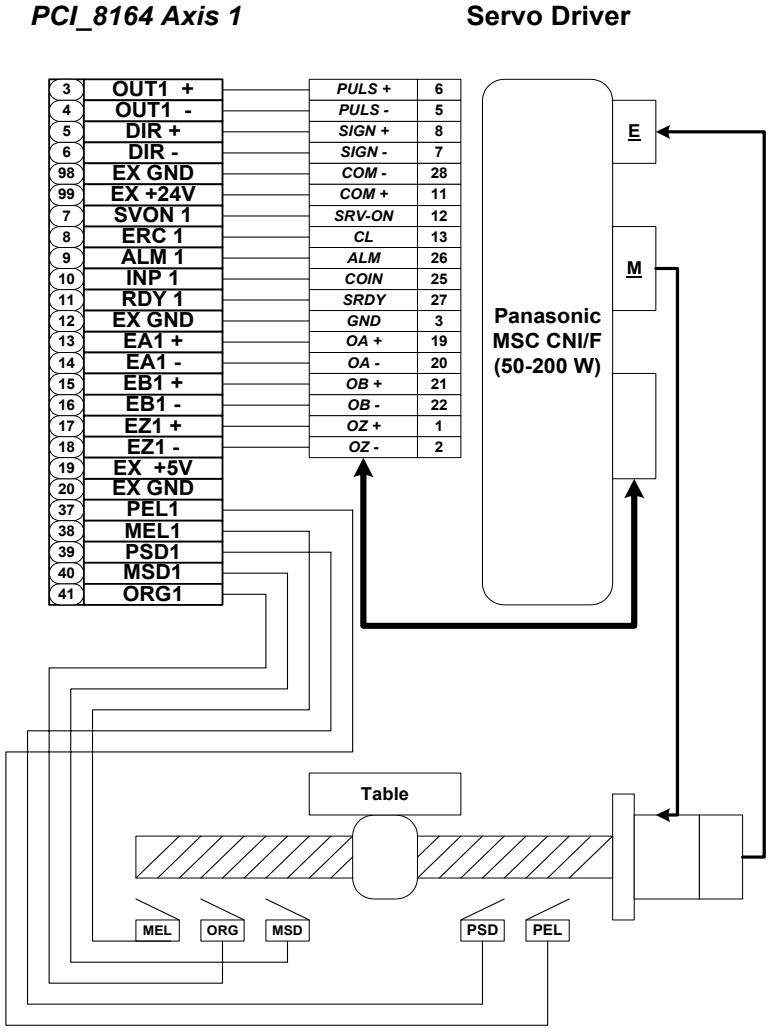


Figure 9: Connection of PCI-8164 with Panasonic Driver

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Wiring of PCI-8164 with SANYO AC Servo PY2

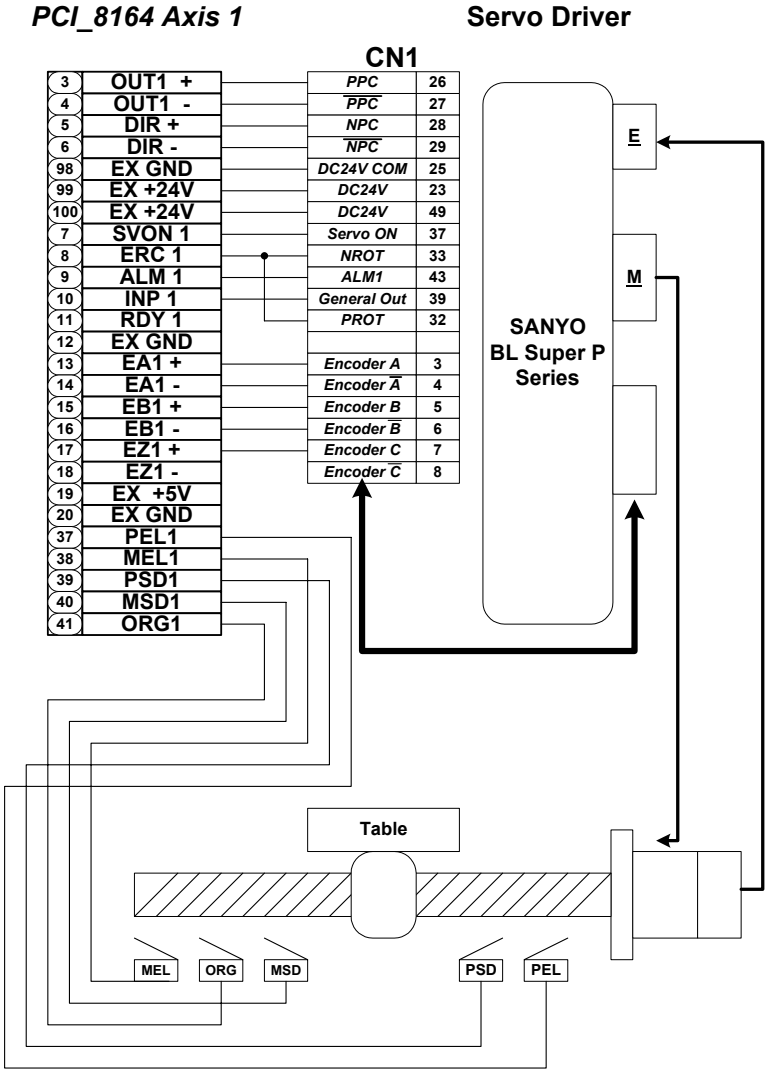
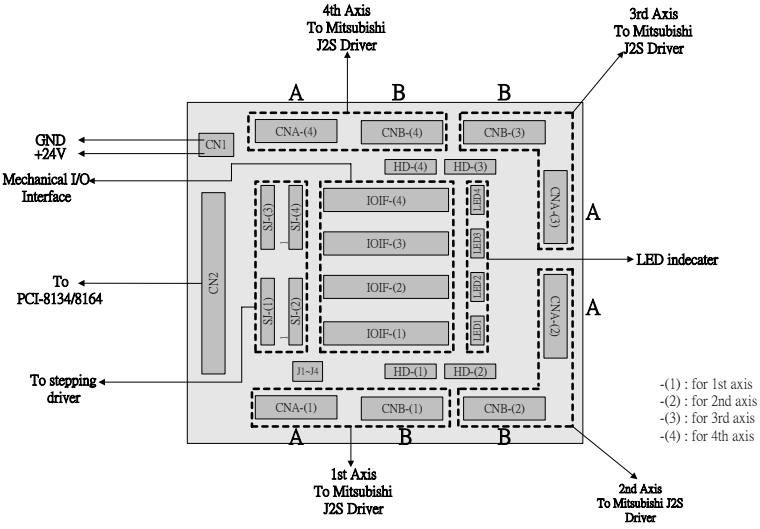


Figure 10: Connection of PCI-8164 with SANYO Driver

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.3 Wiring with DIN-814M

Warning
The DIN-814M is used for wiring between **Mitsubishi J2S** series servo drivers and ADLINK **PCI-8134, PCI-8164, or MPC-8164** motion controller card ONLY. Never try it on any other servo driver and other cards.



Note:

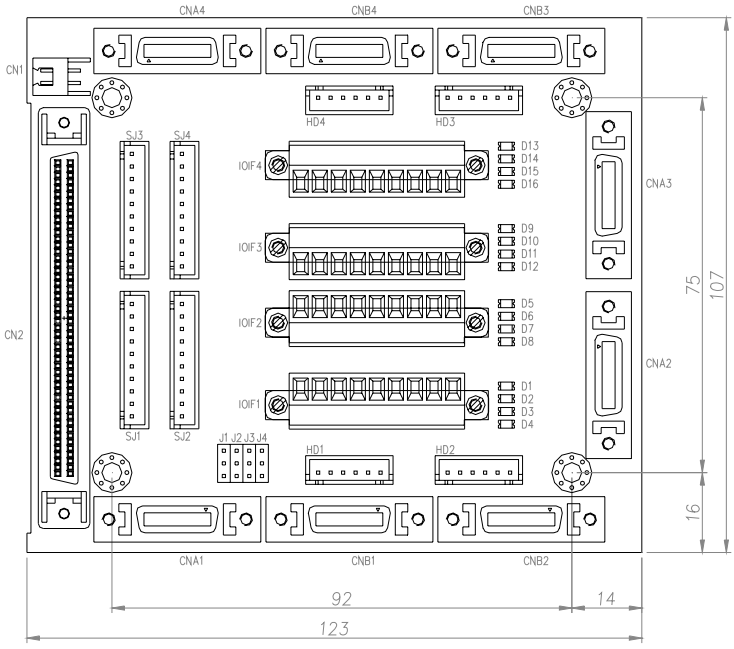
1. The DIN-814M provides 2 connection methods for every axis. The first is through the CNA & CNB connectors. This is for Mitsubishi J2S series servo driver. The second is through SJ connector. This is for stepping driver or other servo drivers (for Panasonic MINAS MSD driver, please use DIN-814P). Keep in mind that the signals in SJ and CNA & CNB of the same axis are directly shorted. DO NOT use both connectors at the same time.
2. Two one-to-one 20-PIN cables are required for connection between the CNA & CNB and the Mitsubishi J2S driver. It is available from ADLINK, or users may contact the local dealer or distributor to get cable information.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3. Depending on which PCI-8134 or PCI-8164/MPC-8164 card used, some signals (PSD and MSD) in the IOIF connector will function differently. When PCI-8134 is used, The PSD and MSD are for positive slow down and negative slow down signal respectively. While PCI-8164 is used, PSD is for CMP and LTC and MSD is for SD. For more detail s, please refer to the PCI-8134 and PCI-8164 user manuals.
4. Ext EMG and EMG: Due to the existence of EMG (Emergence stop signal) in the Mitsubishi J2S driver, users may select either of the following two operations by setting jumpers (J1-J4, J1 for 1st axis, J2 for 2nd axis, etc.).

- **1-2 shorted:** The EMG is shorted to GND, so Ext. EMG of IOIF pin 2 is not used.
- **2-3 shorted:** The Ext. EMG of IOIF pin 2 is connected to EMG at the driver; so, to externally stop the motor set Ext. EMG open to GND.

Mechanical Dimensions:



LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

PIN Assignment:

CNA1~CNA4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	IGND	--	Isolated Ground	2	DIR+	O	Direction Signal (+)
3	OUT+	O	Pulse Signal (+)	4			
5	EZ+	I	Encoder Z-phase (+)	6	EA+	I	Encoder A-phase (+)
7	EB+	I	Encoder B-phase (+)	8	ERC	O	Error counter Clear
9	+24V	O	Voltage output	10	IGND	--	Isolated Ground
11				12	DIR-	O	Direction Signal (-)
13	OUT-	O	Pulse Signal (-)	14			
15	EZ-	I	Encoder Z-phase (-)	16	EA-	I	Encoder A-phase (-)
17	EB-	I	Encoder B-phase (-)	18	INP	I	Servo In Position
19	RDY	I	Servo Ready	20	IGND	--	Isolated Ground

CNB1~CNB4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	IGND	--	Isolated Ground	2			
3				4			
5	Servo ON	O	Servo On	6			
7				8			
9				10	IGND	--	Isolated Ground
11				12			
13	+24V	O	Voltage output	14			
15	EMG	I	Internal EMG Signal	16	IGND	--	Isolated Ground
17	IGND	--	Isolated Ground	18	ALM	I	Servo Alarm
19				20	IGND	--	Isolated Ground

IOIF1~IOIF4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	+24V	O	Voltage output	6	MSD	I	Negative Slow Switch (+)
2	EX_EMG	I	External EMG Signal	7	ORG	I	
3	PEL	I	Positive Limit (+)	8	IGND	--	
4	MEL	I	Negative Limit (-)	9	IGND	--	
5	PSD	I	Positive Slow Switch (+)				

SJ1~SJ4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	OUT+	O	Pulse Signal (+)	6	ALM	I	Servo Alarm
2	OUT-	O	Pulse Signal (-)	7	+5V	O	Voltage output
3	DIR+	O	Direction Signal (+)	8	Servo ON	O	Servo On
4	DIR-	O	Direction Signal (-)	9	+5V	O	Voltage output
5	EZ+	I	Index Signal	10	IGND	--	Isolated Ground

CN1

No.	Name	I/O	Function
1	EX+24V	I	External Power Supply Input (+24V DC ± 5%)
2	EXGND	--	External Power Supply Ground.

HD1~HD4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	+24V	O	Voltage output	4	EX_EMG	I	External EMG Signal
2	Servo ON	O	Servo On	5	ALM	I	Servo Alarm
3	RDY	I	Servo Ready	6	IGND	--	Isolated Ground

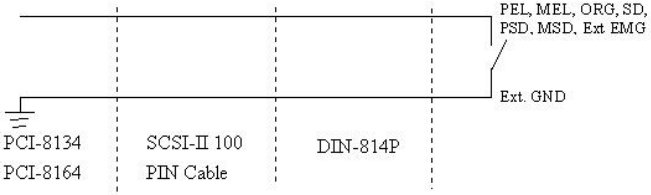
Jumper

J1~J4		1: GND	2: EMG4	3: EX_EMG

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

How to wire

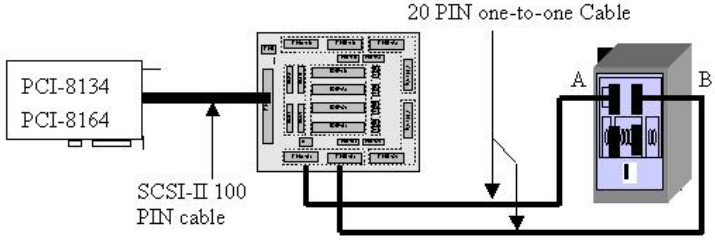
PEL, MEL, ORG, SD, PSD, MSD, Ext.EMG (in IOIF):



CMP, LTC (in IOIF)

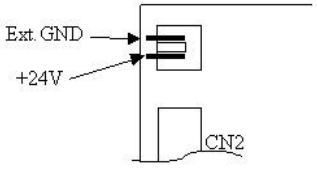
CMP is a TTL 5V or 0V output (vs. Ext GND)
LTC is a TTL 5V or 0V input (vs. Ext. GND)

CNA & CNB, CN2



SJ: Please refer to PCI-8134 / PCI-8164 user manual for wiring.

CN1:

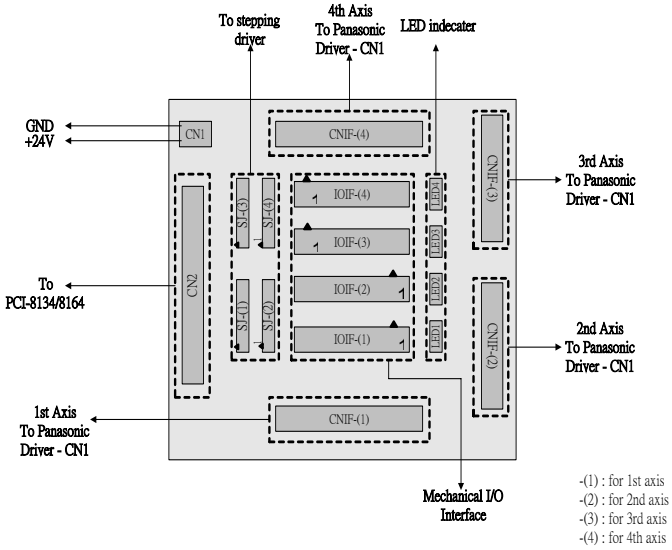


LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

7.4 Wiring with DIN-814P

Warning

The DIN-814M is used for wiring between the **Panasonic MINAS MSD** series servo driver and ADLINK **PCI-8134, PCI-8164** motion controller cards ONLY. Never try it on any other servo drivers or cards.



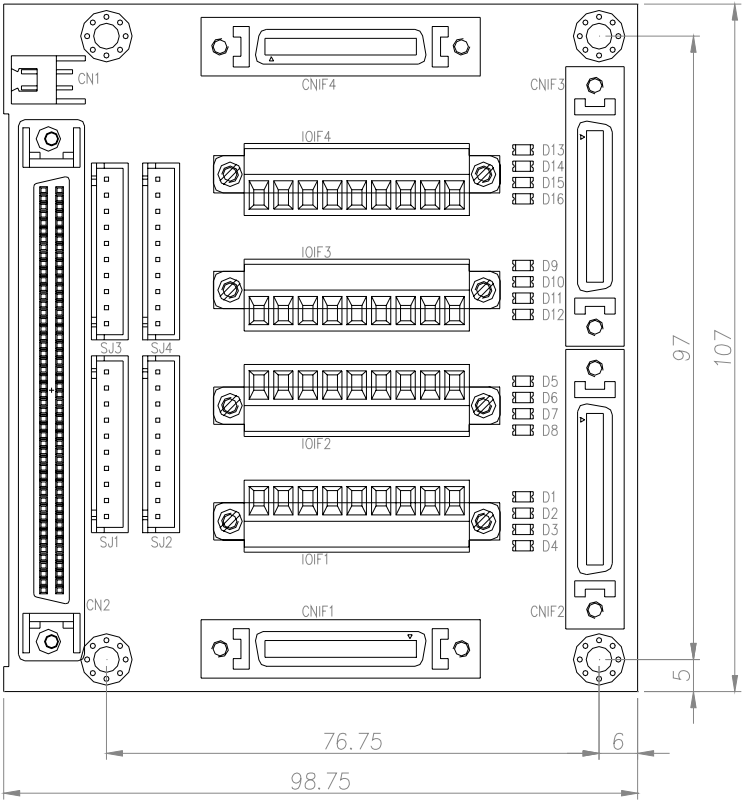
Note:

1. The DIN-814P provides 2 connection methods for every axis. The first is through the CNIF connector for the Panasonic MINAS MSD series servo driver. The second is through SJ connector for stepping drivers or other servo drivers (for the Mitsubishi J2S driver, please use DIN-814M). Keep in mind that the signals in SJ and CNIF of the same axis are directly shorted. DO NOT use both connectors at the same time.
2. A one-to-one 36-PIN cable is required to connect between the CNIF and the Panasonic MINAS MSD driver. It is available from ADLINK, or users may contact a local dealer or distributor to get cable information.

LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

3. Depending on the PCI-8134 or PCI-8164 card used, some signals (PSD & MSD) in the IOIF connector will function differently. When PCI-8134 is used, the PSD and MSD signals are for positive slow down and negative slow down signal respectively. When PCI-8164 is used, PSD is for CMP and LTC, and MSD is for SD. For more details, please refer to the PCI-8134 and PCI-8164 user manuals.

Mechanical Dimensions:



LRi Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

PIN Assignment:

CNIF1~CNIF4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	EZ+	I	Encoder Z-phase (+)	2	EZ-	I	Encoder Z-phase (-)
3	IGND	--	Isolated Ground	4			
5	OUT+	O	Pulse Signal (+)	6	OUT-	O	Pulse Signal (-)
7	DIR+	O	Direction Signal (+)	8	DIR-	O	Direction Signal (-)
9	IGND	--	Isolated Ground	10			
11	+24V	O	Voltage output	12	Servo ON	O	Servo On
13	ERC	O	Error counter Clear	14			
15	IGND	--	Isolated Ground	16			
17				18			
19	EA+	I	Encoder A-phase (+)	20	EA-	I	Encoder A-phase (-)
21	EB+	I	Encoder B-phase (+)	22	EB-	I	Encoder B-phase (-)
23				24			
25	INP	I	Servo In Position	26	ALM	I	Servo Alarm
27	RDY	I	Servo Ready	28	IGND	--	Isolated Ground
29				30			
31				32			
33				34			
35				36			

IOIF1~IOIF4

No.	Name	I/O	Function	No.	Name	I/O	Function
1	+24V	O	Voltage output	6	MSD	I	Negative Slow Switch (+)
2	+24V	O	Voltage output	7	ORG	I	
3	PEL	I	Positive Limit (+)	8	IGND	--	
4	MEL	I	Negative Limit (-)	9	IGND	--	
5	PSD	I	Positive Slow Switch (+)				

SJ1~SJ4

No.	Name	I/O	Function	No	Name	I/O	Function
1	OUT+	O	Pulse Signal (+)	6	ALM	I	Servo Alarm
2	OUT-	O	Pulse Signal (-)	7	+5V	O	Voltage output
3	DIR+	O	Direction Signal (+)	8	Servo ON	O	Servo On
4	DIR-	O	Direction Signal (-)	9	+5V	O	Voltage output
5	EZ+	I	Index Signal	10	IGND	--	Isolated Ground

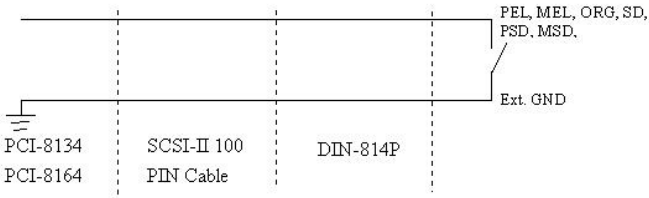
CN1

No.	Name	I/O	Function
1	EX+24V	I	External Power Supply Input (+24V DC ± 5%)
2	EXGND	--	External Power Supply Ground

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

How to wire

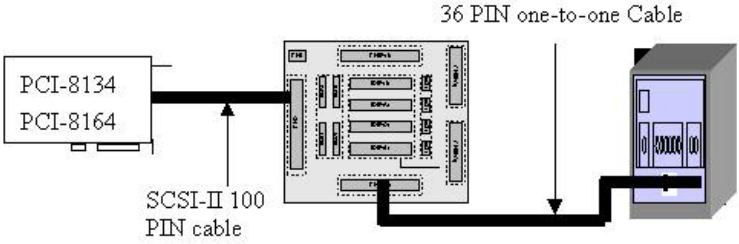
PEL, MEL, ORG, SD, PSD, MSD (in IOIF):



CMP, LTC (in IOIF)

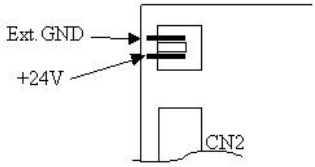
CMP is a TTL 5V or 0V output (vs. Ext GND)
LTC is a TTL 5V or 0V input (vs. Ext. GND)

CNA & CNB, CN2



SJ: Please refer to PCI-8134 / PCI-8164 user manual for wiring.

CN1:



LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Appendix A Color code of CN3 Cable (MPC-8164 Only)

CN3 Pin No	Signal Name	Color	CN3 Pin No	Signal Name	Color
1	DOCOM	Brown	2	DOCOM	Pink-Black
3	DOCOM	Grey	4	DOCOM	Blue-White
5	DO0	Red	6	DO1	Grey-Black
7	DO2	White	8	DO3	Purple-White
9	DO4	Orange	10	DO5	Light Green-Black
11	DO6	Pink	12	DO7	White-Blue
13	--	Yellow	14	DICOM	Light Blue-Black
15	DICOM	Light Blue	16	DICOM	Red-White
17	DICOM	Green	18	DI0	Green-Black
19	DI1	Light Green	20	DI2	Brown-White
21	DI3	Blue	22	DI4	Yellow-Black
23	DI5	Red Black	24	DI6	White-Black
25	DI7	Purple	26	--	Black-Orange

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

1. Before using ADLINK's products please read the user manual and follow the instructions exactly. When sending in damaged products for repair, please attach an RMA application form.

2. All ADLINK products come with a two-year guarantee, repaired free of charge.

    - The warranty period starts from the product's shipment date from ADLINK's factory
    - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers' warranty
    - End users requiring maintenance services should contact their local dealers. Local warranty conditions will depend on the local dealers.

3. Our repair service does not cover the two-year warranty, if the following items cause damage:

    a. Damage caused by not following instructions on user menus.

    b. Damage caused by carelessness on the users' part during product transportation.

    c. Damage caused by fire, earthquakes, floods, lightening, pollution, and/or incorrect usage of voltage transformers.

    d. Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity or volatile chemicals.

    e. Damage caused by leakage of battery fluid when changing batteries.

    f. Damage from improper repair by unauthorized technicians.

    g. Products with altered and/or damaged serial numbers are not entitled to our service.

    h. Other categories not protected under our guarantees.

4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000

5. To ensure the speed and quality of product repair, please download a RMA application form from our company website www.adlinktech.com. Damaged products with RMA forms attached receive priority.

For further questions, please contact our FAE staff.

ADLINK: service@adlinktech.com

Test & Measurement Product Segment: NuDAQ@adlinktech.com

Automation Product Segment: Automation@adlinktech.com

Computer & Communication Product Segment: NuPRO@adlinktech.com; NuIPC@adlinktech.com

LRI Distribuída no Brasil pela LRI | vendas@lri.com.br | 51 2129-3000