Getting Service from ADLINK NK

Customer Satisfaction is top priority for ADLINK Technology Inc. Please contact us should you require any service or assistance.

ADLINK TECHNOLOGY INC.

Web Site: http://www.adlinktech.com
Sales & Service: Service@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 this completed service form for prompt and satisfactory service.

<table>
<thead>
<tr>
<th>Company Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Company/Organization</td>
</tr>
<tr>
<td>Contact Person</td>
</tr>
<tr>
<td>E-mail Address</td>
</tr>
<tr>
<td>Address</td>
</tr>
<tr>
<td>Country</td>
</tr>
<tr>
<td>TEL</td>
</tr>
<tr>
<td>FAX:</td>
</tr>
<tr>
<td>Web Site</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Model</td>
</tr>
<tr>
<td>Environment</td>
</tr>
<tr>
<td>OS:</td>
</tr>
<tr>
<td>M/B:</td>
</tr>
<tr>
<td>CPU:</td>
</tr>
<tr>
<td>Chipset:</td>
</tr>
<tr>
<td>BIOS:</td>
</tr>
</tbody>
</table>

Please give a detailed description of the problem(s):
# Table of Contents

Table of Contents................................................................................................. i  

List of Figures ....................................................................................................... v  

## 1 Introduction ................................................................................................. 1  
1.1 Features......................................................................................... 4  
1.2 Specifications............................................................................. 5  
1.3 Supported Software ................................................................. 8  
  Programming Library ............................................................... 8  
  MotionCreator ......................................................................... 8  
1.4 Available Terminal Board......................................................... 8  

## 2 Installation ................................................................................................. 9  
2.1 Package Contents ....................................................................... 9  
2.2 cPCI-8168 Outline Drawing ....................................................... 10  
2.3 cPCI-8168 Hardware Installation ............................................... 11  
  Hardware configuration ....................................................... 11  
  PCI slot selection ................................................................ 11  
  Installation Procedures ....................................................... 11  
  Troubleshooting .................................................................. 11  
2.4 Software Driver Installation ....................................................... 13  
2.5 CN2 Pin Assignments: External Power Input ......................... 14  
2.6 CN3 Pin Assignments: SCSI 68 Pins Connector .................... 15  
2.7 CN4 Pin Assignments: SCSI 68 Pins Connector .................... 16  
  CN5 Pin Assignments: SCSI 68 Pins Connector .................. 17  
2.8 CN6 Pin Assignments: SCSI 68 Pins Connector .................... 18  
2.9 Overview of CN3-CN6 .......................................................... 19  
2.10 CN1: RJ45 Connector for HSL .............................................. 20  
2.11 Switch Setting ...................................................................... 21  

## 3 Signal Connections .................................................................................... 23  
3.1 Pulse Output Signals OUT and DIR ...................................... 24  
3.2 Encoder Feedback Signals EA, EB and EZ ......................... 25  
  Connection to Line Driver Output ...................................... 25  
  Connection to Open Collector Output ............................. 26  
3.3 Origin Signal ORG ................................................................. 27  
3.4 End-Limit Signals PEL and MEL .......................................... 28  
3.5 In-position Signal INP .......................................................... 29
3.6 Alarm Signal ALM .............................................................. 30
3.7 Deviation Counter Clear Signal ERC ................................. 31
3.8 General-purpose Signal SVON .......................................... 32
3.9 General-purpose Signal RDY ............................................ 33
3.10 Isolated Digital Output DOx ............................................... 34
3.11 Isolated Digital Input DIx .................................................... 35
3.12 High Speed High Current Digital Output HSOUT .............. 36
3.13 High Speed Digital Input HSIN ........................................... 37
3.14 Comparison Output CMP1 and CMP2 .............................. 38

4 Operation Theory .............................................................. 39
4.1 Classifications of Motion Controller ................................. 39
  Voltage type motion control Interface ............................... 39
  Pulse type motion control Interface ................................ 40
  Network type motion control Interface ............................. 40
  Software real-time motion control kernel ...................... 41
  DSP based motion control kernel ................................ 41
  ASIC based motion control kernel ................................ 41
  Compare Table of all motion control types ................... 42
  cPCI-8168’s motion controller type ............................... 42
4.2 Motion Control Modes ........................................................ 43
  Coordinate system ........................................................ 43
  Absolute and relative position move ......................... 44
  Trapezoidal speed profile ........................................... 45
  S-curve and Bell-curve speed profile ............................ 45
  Velocity mode ............................................................... 47
  One axis position mode ............................................. 48
  Two axes linear interpolation position mode ................. 49
  Two axes circular interpolation mode ........................... 50
  Continuous motion ........................................................ 51
  Home Return Mode ...................................................... 54
  Home Search Function ................................................ 61
  Manual Pulser Function .............................................. 62
  Simultaneous Start Function ...................................... 62
  Speed Override Function ........................................... 63
  Position Override Function ........................................ 63
4.3 The motor driver interface ................................................ 64
  Pulse Command Output Interface ................................ 64
  Pulse feedback input interface ..................................... 66
  In position signal ......................................................... 68
Servo alarm signal ........................................................ 69
Error clear signal ........................................................... 69
Servo ON/OFF switch ................................................... 69
Servo Ready Signal ...................................................... 70
Servo alarm reset switch .............................................. 70

4.4 Mechanical switch interface ............................................... 70
Original or home signal ................................................. 71
End-Limit switch signal ................................................. 71
Slow down switch ......................................................... 71
Positioning Start switch ................................................. 71
Counter Clear switch .................................................... 72
Counter Latch switch .................................................... 72
Emergency stop input ................................................... 72

4.5 The Counters ..................................................................... 72
Command position counter ........................................... 73
Feedback position counter ............................................ 73
Command and Feedback error counter ........................ 74
General purpose counter .............................................. 74
Target position recorder ................................................ 74

4.6 The Comparaters............................................................... 75
Soft end-limit comparators ............................................ 75
Command and feedback error counter comparators ... 75
General comparator ...................................................... 75
Trigger comparator ....................................................... 76

4.7 Other Motion Functions ..................................................... 77
Backlash compensation and slip corrections .............. 77
Vibration restriction function ......................................... 77
Speed profile calculation function ................................. 78

4.8 Interrupt Control................................................................. 79

4.9 Multiple Card Operation..................................................... 80

5 Motion Creator.................................................................. 81
5.1 Execute Motion Creator ..................................................... 81
5.2 About Motion Creator......................................................... 82
5.3 Motion Creator Form Introducing....................................... 83
    Initial forms ............................................................ 83
    Operation Form ......................................................... 84
    Interface I/O configuration form .................................... 90
    Pulse I/O and INT factor configuration form .......... 92
    HSLink testing form .................................................. 93
6 Function Library ................................................................. 97
  6.1 List of Functions ............................................................. 98
  6.2 C/C++ Programming Library .......................................... 108
  6.3 Initialization ................................................................. 109
  6.4 Pulse Input/Output Configuration ..................................... 112
  6.5 Velocity mode motion..................................................... 115
  6.6 Single Axis Position Mode ............................................. 119
  6.7 Linear Interpolated Motion ............................................. 125
  6.8 Circular Interpolation Motion ........................................... 132
  6.9 Home Return Mode.......................................................... 136
  6.10 Manual Pulser Motion ................................................... 139
  6.11 Motion Status ............................................................... 144
  6.12 Motion Interface I/O ...................................................... 146
  6.13 Motion I/O Monitoring.................................................... 150
  6.14 Interrupt Control ......................................................... 152
  6.15 Position Control and Counters ........................................ 160
  6.16 Position Compare and Latch .......................................... 165
  6.17 Continuous motion ........................................................ 173
  6.18 Multiple Axes Simultaneous Operation ............................ 175
  6.19 HSL Initialization .......................................................... 178
  6.20 HSL Connection Information ......................................... 180
  6.21 HSL DIO Read/Write ..................................................... 182
  6.22 General-purposed DIO .................................................... 185
  6.23 General-purposed AIO .................................................... 187
  6.24 High Resolution Timer Functions .................................... 188
  6.25 DI Change of state ....................................................... 191

7 Appendix ........................................................................... 195
  7.1 General I/O usages on PCL6045B .................................... 195
  7.2 Dual purpose pins on CN3~6 ........................................... 196

Warranty Policy ...................................................................... 197
List of Figures

Figure 1-1: Block Diagram of the cPCI-8168 ............................... 2
Figure 1-2: Flow chart for building an application ....................... 3
Figure 2-1: PCB Layout of the CPCI-8168 .................................. 10
1 Introduction

The cPCI-8168 is an advanced 8-axis motion controller card with a 6U Compact PCI interface. It can generate high frequency pulses (6.55MHz) to drive stepper or servomotors. As a motion controller, it can provide 8-axis linear and circular interpolation and continuous interpolation for continuous velocity. Also, changing position/speed on the fly is available with a single axis operation.

Multiple cPCI-8168 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. cPCI-8168 features the position compare and trigger output function which allows users to coordinate the comparing points with ADLINK library and send the triggering pulse to another device. In addition, a mechanical sensor interface, servo motor interface, and general-purposed I/O signals are provided for easy system integration.

Figure 1-1 shows the functional block diagram of the cPCI-8168 card. The motion control functions include trapezoidal and S-curve acceleration/deceleration, linear and circular interpolation between two axes and continuous motion positioning, and 13 home return modes. All these functions and complex computa-
tions are performed internally by the ASIC, thus it can save CPU loading.

Figure 1-1: Block Diagram of the cPCI-8168

MotionCreator is a Windows-based application development software package included with the cPCI-8168. MotionCreator 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 cPCI-8168.

Windows programming libraries are also provided for C++ compiler and Visual Basic. Sample programs are provided to illustrate the operations of the functions.
Figure 1-2 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 1-2: Flow chart for building an application
1.1 Features

The following lists summarize the main features of the cPCI-8168 motion control system.

- 32-bit cPCI-Bus plug and play, Rev 2.2
- 6U Compact PCI form factor
- 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
- 28-bit up/down counter for incremental encoder feedback.
- Programmable acceleration and deceleration time
- Trapezoidal and S-curve velocity profiles
- Any 2 of 4 axes circular interpolation.
- 2-4 axes linear interpolation.
- Continuous interpolation
- Change position and speed on the Fly. (Position/speed override)
- 2 axes position compare trigger output with 4K FIFO auto-loading.
- 13 home return modes
- Software limit function
- Home switch, index signal, positive and negative limit switches, slow down point interface provided for all axes.
- All digital input and output signals are 2500Vrms isolated
- Programmable interrupt sources
- Simultaneous start/stop motion on multiple axes
- 4 channels analog input; 4 channels analog output
- 8 channels general purpose digital input/output
- 4 channels high speed digital input/output
- 2 channels high speed remote I/O master control (HSL)
- All general digital input pins have change of state interrupt
- One high resolution hardware timer
- Software supports maximum up to 6 cPCI-8168 cards (48
axes) operation.

- MotionCreator, Microsoft Windows based application development software.
- cPCI-8168 Library and Utility for DOS library and Windows NT/2000/XP DLL.

1.2 Specifications

**Applicable Motors:**
- Stepping motors.
- AC or DC servomotors with pulse train input servo drivers.

**Performance:**
- Number of controllable axes: 8 axes.
- Maximum pulse output frequency: 6.55Mpps, linear, trapezoidal or S-Curve velocity profile drive.
- Internal reference clock: 19.66 MHz
- Position pulse setting range: -134,217,728~ +134,217,728 pulses (28-bit).
- Up / down counter counting range: 0~268,435,455 (28-bit.) or –134,217,728 to +134,217,727

**I/O Signals:**
- All I/O signals 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: (1)EL, SD/PCS and ORG.
- Servomotor interface I/O pins: INP, ALM and ERC.
- Position compare output pin: CMP
- General-purpose digital output pin: SVON.
- General-purpose digital input pin: RDY.
General-Purpose Digital Output

- 8 channels isolated level Digital Output: DOUT
  - Output voltage range: 5V - 35V
  - Output type: NPN open collector
  - Darlington transistor array: TD62083
  - Sink current: 90 mA Max each channel

- 8 channels isolated level Digital Input: DIN
  - Input voltage range: 0V - 30V
  - Input resistor 4.7Kohm (1/2W)
  - Isolation optocoupler: PC3H4

- 4 channels high speed digital input: HSIN
  - Input voltage range: 0 - 5V
  - Change of State detection
  - Isolation optocoupler: HP0631

- 4 channels high speed digital output with high current output ability: HSOUT
  - Output voltage range: 5V - 30V
  - Output type: Open Drain Power MOS
  - Sink current: 1Amp. Max each channel
  - Signal switching time: Maximum 15 ns.

Analog Input: AIN

- Converter and Resolution:
  - 12 Bit Linear Technology LTC1402
- Input channels: 4 Single-Ended
- Input range: +/- 10V; Bipolar
- Conversion Time: 5u sec
- Sampling rate: 200K samples/sec maximum
- Output Voltage: Min. 5V; Max. 35V
- Over voltage protection: Continuous +/-35V
- Accuracy: 0.01% of FSR +/-1 LSB
Analog Output: AOUT

- Converter and Resolution:
  - 16 Bit Analog Device AD1866R
- Output channels: 4 Single-Ended
- Output range: +/- 10V; Bipolar
- Settling Time: 2μ sec (-10V ~ +10V)

High Speed Remote I/O

- Connector: RJ45
- Data rate: 6Mpps
- 2 master controller on RJ45
- Each master controller controls 63 slave modules at most
- I/O refreshing rate: 30.1 usec per slave module
- Full/Half Duplex mode selectable on each master
- Multi-drop full duplex RS422 with transformer isolation

General Specifications

- Connectors: 4 x 68-pin mini SCSI-type connector
- Operating Temperature: 0°C - 50°C
- Storage Temperature: -20°C - 80°C
- Humidity: 5 - 85%, non-condensing
### 1.3 Supported Software

#### 1.3.1 Programming Library

Windows NT/2000/XP DLL are provided for the cPCI-8168 users. These function libraries are shipped with the board.

#### 1.3.2 MotionCreator

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.

### 1.4 Available Terminal Board

DIN-68S/2, DIN-68Y-SGII, DIN-68P-A4 and DIN-68M-J3A are available for cPCI-8168 wiring. DIN-68S/2 is general purposed pin-to-pin terminal board. Users can use this to connect servos and steppers. DIN-68M-J3A is designed for Mitsubishi servo J3A amplifier. DIN-68Y-SGII is designed for Yaskawa SigmaII amplifer. DIN-68P-A4 is designed for Panasonic Minax A4 amplifier. Users can use these boards to connect servo driver and steppers.
2 Installation

This chapter describes how to install the cPCI-8168. Please follow these steps below:

- Check what your package contents (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:

- cPCI-8168: advanced 8-axis Servo / Stepper Motion Control Card
- ADLINK All-in-one Compact Disc
- Terminal board is an optional accessory. This would not be included in cPCI-8168 package.

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.
2.2 cPCI-8168 Outline Drawing

Figure 2-1: PCB Layout of the CPCI-8168

- CN1: High Speed Link RJ45 Connector
- CN2: External Power Input Connector (+24V)
- CN3: SCSI 68Pin Connector for Axis0 &1
- CN4: SCSI 68Pin Connector for Axis2 &3
- CN5: SCSI 68Pin Connector for Axis4 &5
- CN6: SCSI 68Pin Connector for Axis6 &7
- S1,S2: Axis0~7 end-limit logic setting
- J1,J2: Full/Half duplex setting for chip of HSL port1 and port2
- JP2-JP5: Full/Half duplex setting for route of HSL port1 and port2
2.3 cPCI-8168 Hardware Installation

2.3.1 Hardware configuration
The cPCI-8168 is fully Plug and Play compliant. Hence memory allocation (I/O port locations) and IRQ channel 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 cPCI-8168 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 cPCI-8168, 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.
2.4 Software Driver Installation

cPCI-8168:

1. Autorun the ADLINK All-In-One CD. Choose Driver Installation -> Motion Control -> cPCI-8168
2. Follow the procedures of the installer.
3. After setup installation is completed, restart windows.

**Suggestion:** Please download the latest software from ADLINK website if necessary.
2.5 CN2 Pin Assignments: External Power Input

<table>
<thead>
<tr>
<th>CN1 Pin No</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EX+24V</td>
<td>External power supply of +24V DC ± 5%</td>
</tr>
<tr>
<td>2</td>
<td>EXGND</td>
<td>Grounds of the external power</td>
</tr>
</tbody>
</table>

Notes:

1. CN2 is a plug-in terminal board with no screw.
2. Be sure to use the external power supply. The +24V DC is used by the external input/output signal circuit. The power circuit is configured as follows:
3. Wires for connection to CN2
   - Solid wire: φ 0.32 mm to φ 0.65 mm (AWG28 to AWG22)
   - Twisted wire: 0.08 mm² to 0.32 mm² (AWG28 to AWG22)
   - Naked wire length: 10 mm standard

The following diagram shows the external power supply system of the cPCI-8168. The external +24V power must be provided, an on-board regulator generates +5V for both internal and external usage.
## 2.6 CN3 Pin Assignments: SCSI 68 Pins Connector

### Axis 1 - 2

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (1)</th>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>25</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>2</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>26</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>3</td>
<td>OUT1+</td>
<td>O</td>
<td>Pulse signal (+),(1)</td>
<td>27</td>
<td>OUT2+</td>
<td>O</td>
<td>Pulse signal (+),(2)</td>
</tr>
<tr>
<td>4</td>
<td>OUT1-</td>
<td>O</td>
<td>Pulse signal (-),(1)</td>
<td>28</td>
<td>OUT2-</td>
<td>O</td>
<td>Pulse signal (-),(2)</td>
</tr>
<tr>
<td>5</td>
<td>DIR1+</td>
<td>O</td>
<td>Dir. signal (+),(1)</td>
<td>29</td>
<td>DIR2+</td>
<td>O</td>
<td>Dir. signal (+),(2)</td>
</tr>
<tr>
<td>6</td>
<td>DIR1-</td>
<td>O</td>
<td>Dir. signal (-),(1)</td>
<td>30</td>
<td>DIR2-</td>
<td>O</td>
<td>Dir. signal (-),(2)</td>
</tr>
<tr>
<td>7</td>
<td>SVON1</td>
<td>O</td>
<td>Multi-purpose signal, (1)</td>
<td>31</td>
<td>SVON2</td>
<td>O</td>
<td>Multi-purpose signal, (2)</td>
</tr>
<tr>
<td>8</td>
<td>ERC1</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (1)</td>
<td>32</td>
<td>ERC2</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (2)</td>
</tr>
<tr>
<td>9</td>
<td>ALM1</td>
<td>I</td>
<td>Alarm signal, (1)</td>
<td>33</td>
<td>ALM2</td>
<td>I</td>
<td>Alarm signal, (2)</td>
</tr>
<tr>
<td>10</td>
<td>INP1</td>
<td>I</td>
<td>In-position signal, (1)</td>
<td>34</td>
<td>INP2</td>
<td>I</td>
<td>In-position signal, (2)</td>
</tr>
<tr>
<td>11</td>
<td>RDY1</td>
<td>I</td>
<td>Multi-purpose signal, (1)</td>
<td>35</td>
<td>RDY2</td>
<td>I</td>
<td>Multi-purpose signal, (2)</td>
</tr>
<tr>
<td>12</td>
<td>EA1+</td>
<td>I</td>
<td>Encoder A-phase (+), (1)</td>
<td>36</td>
<td>EA2+</td>
<td>I</td>
<td>Encoder A-phase (+),(2)</td>
</tr>
<tr>
<td>13</td>
<td>EA1-</td>
<td>I</td>
<td>Encoder A-phase (-), (1)</td>
<td>37</td>
<td>EA2-</td>
<td>I</td>
<td>Encoder A-phase (-),(2)</td>
</tr>
<tr>
<td>14</td>
<td>EB1+</td>
<td>I</td>
<td>Encoder B-phase (+), (1)</td>
<td>38</td>
<td>EB2+</td>
<td>I</td>
<td>Encoder B-phase (+),(2)</td>
</tr>
<tr>
<td>15</td>
<td>EB1-</td>
<td>I</td>
<td>Encoder B-phase (-), (1)</td>
<td>39</td>
<td>EB2-</td>
<td>I</td>
<td>Encoder B-phase (-),(2)</td>
</tr>
<tr>
<td>16</td>
<td>EZ1+</td>
<td>I</td>
<td>Encoder Z-phase (+), (1)</td>
<td>40</td>
<td>EZ2+</td>
<td>I</td>
<td>Encoder Z-phase (+),(2)</td>
</tr>
<tr>
<td>17</td>
<td>EZ1-</td>
<td>I</td>
<td>Encoder Z-phase (-), (1)</td>
<td>41</td>
<td>EZ2-</td>
<td>I</td>
<td>Encoder Z-phase (-),(2)</td>
</tr>
<tr>
<td>18</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>42</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>19</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>43</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>20</td>
<td>PEL1</td>
<td>I</td>
<td>End limit signal (+), (1)</td>
<td>44</td>
<td>PEL2</td>
<td>I</td>
<td>End limit signal (+),(2)</td>
</tr>
<tr>
<td>21</td>
<td>MEL1</td>
<td>I</td>
<td>End limit signal (-), (1)</td>
<td>45</td>
<td>MEL2</td>
<td>I</td>
<td>End limit signal (-),(2)</td>
</tr>
<tr>
<td>22</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>46</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>23</td>
<td>SD1</td>
<td>I</td>
<td>Slow Down signal, (1)</td>
<td>47</td>
<td>SD2</td>
<td>I</td>
<td>Slow Down signal, (2)</td>
</tr>
<tr>
<td>24</td>
<td>ORG1</td>
<td>I</td>
<td>Origin signal, (1)</td>
<td>48</td>
<td>ORG2</td>
<td>I</td>
<td>Origin signal, (2)</td>
</tr>
<tr>
<td>25</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>49</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>26</td>
<td>HSIN1</td>
<td>I</td>
<td>High Speed DI, (1)</td>
<td>50</td>
<td>HSIN2</td>
<td>I</td>
<td>High Speed Digital Input,(2)</td>
</tr>
<tr>
<td>27</td>
<td>HO_COM</td>
<td>I</td>
<td>Common of HSOUT</td>
<td>51</td>
<td>HO_COM</td>
<td>I</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>28</td>
<td>HSOUT1</td>
<td>O</td>
<td>High Speed DO, (1)</td>
<td>52</td>
<td>HSOUT2</td>
<td>O</td>
<td>High Speed DO, (2)</td>
</tr>
<tr>
<td>29</td>
<td>DI_COM</td>
<td>I</td>
<td>Digital Input COM+</td>
<td>53</td>
<td>DO_COM</td>
<td>I</td>
<td>Digital Output COM+</td>
</tr>
<tr>
<td>30</td>
<td>DIN1</td>
<td>I</td>
<td>Digital Input (1)</td>
<td>54</td>
<td>DIN2</td>
<td>I</td>
<td>Digital Input (2)</td>
</tr>
<tr>
<td>31</td>
<td>DOUT1</td>
<td>O</td>
<td>Digital Output (1)</td>
<td>55</td>
<td>DOUT2</td>
<td>O</td>
<td>Digital Output (2)</td>
</tr>
<tr>
<td>32</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>56</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>33</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>57</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>34</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
<td>58</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
</tr>
</tbody>
</table>
### 2.7 CN4 Pin Assignments: SCSI 68 Pins Connector

#### Axis 3 - 4

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (3)</th>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>35</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>2</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>36</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>3</td>
<td>OUT3+</td>
<td>O</td>
<td>Pulse signal (+),(3)</td>
<td>37</td>
<td>OUT4+</td>
<td>O</td>
<td>Pulse signal (+),(4)</td>
</tr>
<tr>
<td>4</td>
<td>OUT3-</td>
<td>O</td>
<td>Pulse signal (-),(3)</td>
<td>38</td>
<td>OUT4-</td>
<td>O</td>
<td>Pulse signal (-),(4)</td>
</tr>
<tr>
<td>5</td>
<td>DIR3+</td>
<td>O</td>
<td>Dir. signal (+),(3)</td>
<td>39</td>
<td>DIR4+</td>
<td>O</td>
<td>Dir. signal (+),(4)</td>
</tr>
<tr>
<td>6</td>
<td>DIR3-</td>
<td>O</td>
<td>Dir. signal (-),(3)</td>
<td>40</td>
<td>DIR4-</td>
<td>O</td>
<td>Dir. signal (-),(4)</td>
</tr>
<tr>
<td>7</td>
<td>SVON3</td>
<td>O</td>
<td>Multi-purpose signal, (3)</td>
<td>41</td>
<td>SVON4</td>
<td>O</td>
<td>Multi-purpose signal, (4)</td>
</tr>
<tr>
<td>8</td>
<td>ERC3</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (3)</td>
<td>42</td>
<td>ERC4</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (4)</td>
</tr>
<tr>
<td>9</td>
<td>ALM3</td>
<td>I</td>
<td>Alarm signal, (3)</td>
<td>43</td>
<td>ALM4</td>
<td>I</td>
<td>Alarm signal, (3)</td>
</tr>
<tr>
<td>10</td>
<td>INP3</td>
<td>I</td>
<td>In-position signal, (3)</td>
<td>44</td>
<td>INP4</td>
<td>I</td>
<td>In-position signal, (4)</td>
</tr>
<tr>
<td>11</td>
<td>RDY3</td>
<td>I</td>
<td>Multi-purpose signal, (3)</td>
<td>45</td>
<td>RDY4</td>
<td>I</td>
<td>Multi-purpose signal, (4)</td>
</tr>
<tr>
<td>12</td>
<td>EA3+</td>
<td>I</td>
<td>Encoder A-phase (+),(3)</td>
<td>46</td>
<td>EA4+</td>
<td>I</td>
<td>Encoder A-phase (+),(4)</td>
</tr>
<tr>
<td>13</td>
<td>EA3-</td>
<td>I</td>
<td>Encoder A-phase (-),(3)</td>
<td>47</td>
<td>EA4-</td>
<td>I</td>
<td>Encoder A-phase (-),(4)</td>
</tr>
<tr>
<td>14</td>
<td>EB3+</td>
<td>I</td>
<td>Encoder B-phase (+),(3)</td>
<td>48</td>
<td>EB4+</td>
<td>I</td>
<td>Encoder B-phase (+),(4)</td>
</tr>
<tr>
<td>15</td>
<td>EB3-</td>
<td>I</td>
<td>Encoder B-phase (-),(3)</td>
<td>49</td>
<td>EB4-</td>
<td>I</td>
<td>Encoder B-phase (-),(4)</td>
</tr>
<tr>
<td>16</td>
<td>EZ3+</td>
<td>I</td>
<td>Encoder Z-phase (+),(3)</td>
<td>50</td>
<td>EZ4+</td>
<td>I</td>
<td>Encoder Z-phase (+),(4)</td>
</tr>
<tr>
<td>17</td>
<td>EZ3-</td>
<td>I</td>
<td>Encoder Z-phase (-),(3)</td>
<td>51</td>
<td>EZ4-</td>
<td>I</td>
<td>Encoder Z-phase (-),(4)</td>
</tr>
<tr>
<td>18</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>52</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>19</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>53</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>20</td>
<td>PEL3</td>
<td>I</td>
<td>End limit signal (+),(3)</td>
<td>54</td>
<td>PEL4</td>
<td>I</td>
<td>End limit signal (+),(4)</td>
</tr>
<tr>
<td>21</td>
<td>MEL3</td>
<td>I</td>
<td>End limit signal (-),(3)</td>
<td>55</td>
<td>MEL4</td>
<td>I</td>
<td>End limit signal (-),(4)</td>
</tr>
<tr>
<td>22</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>56</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>23</td>
<td>SD3</td>
<td>I</td>
<td>Slow Down signal, (3)</td>
<td>57</td>
<td>SD4</td>
<td>I</td>
<td>Slow Down signal, (4)</td>
</tr>
<tr>
<td>24</td>
<td>ORG3</td>
<td>I</td>
<td>Origin signal, (3)</td>
<td>58</td>
<td>ORG4</td>
<td>I</td>
<td>Origin signal, (4)</td>
</tr>
<tr>
<td>25</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>59</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>26</td>
<td>HSIN3</td>
<td>I</td>
<td>High Speed Digital Input (3)</td>
<td>60</td>
<td>HSIN4</td>
<td>I</td>
<td>High Speed Digital Input (4)</td>
</tr>
<tr>
<td>27</td>
<td>HO_COM</td>
<td>I</td>
<td>Common of HSOUT</td>
<td>61</td>
<td>HO_COM</td>
<td>I</td>
<td>Common of HSOUT</td>
</tr>
<tr>
<td>28</td>
<td>HSOUT3</td>
<td>O</td>
<td>High Speed DO (3)</td>
<td>62</td>
<td>HSOUT4</td>
<td>O</td>
<td>High Speed DO (4)</td>
</tr>
<tr>
<td>29</td>
<td>DI_COM</td>
<td>I</td>
<td>Digital Input COM+</td>
<td>63</td>
<td>DO_COM</td>
<td>I</td>
<td>Digital Output COM+</td>
</tr>
<tr>
<td>30</td>
<td>DIN3</td>
<td>I</td>
<td>Digital Input (3)</td>
<td>64</td>
<td>DIN4</td>
<td>I</td>
<td>Digital Input (4)</td>
</tr>
<tr>
<td>31</td>
<td>DOUT3</td>
<td>O</td>
<td>Digital Output (3)</td>
<td>65</td>
<td>DOUT4</td>
<td>O</td>
<td>Digital Output (4)</td>
</tr>
<tr>
<td>32</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>66</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>33</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>67</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>34</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
<td>68</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
</tr>
</tbody>
</table>
### 2.7.1 CN5 Pin Assignments: SCSI 68 Pins Connector

**Axis 5 - 6**

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (5)</th>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>35</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>2</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>36</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>3</td>
<td>OUT5+</td>
<td>O</td>
<td>Pulse signal (+),(5)</td>
<td>37</td>
<td>OUT6+</td>
<td>O</td>
<td>Pulse signal (+),(6)</td>
</tr>
<tr>
<td>4</td>
<td>OUT5-</td>
<td>O</td>
<td>Pulse signal (-),(5)</td>
<td>38</td>
<td>OUT6-</td>
<td>O</td>
<td>Pulse signal (-),(6)</td>
</tr>
<tr>
<td>5</td>
<td>DIR5+</td>
<td>O</td>
<td>Dir. signal (+),(5)</td>
<td>39</td>
<td>DIR6+</td>
<td>O</td>
<td>Dir. signal (+),(6)</td>
</tr>
<tr>
<td>6</td>
<td>DIR5-</td>
<td>O</td>
<td>Dir. signal (-),(5)</td>
<td>40</td>
<td>DIR6-</td>
<td>O</td>
<td>Dir. signal (-),(6)</td>
</tr>
<tr>
<td>7</td>
<td>SVON5</td>
<td>O</td>
<td>Multi-purpose signal, (5)</td>
<td>41</td>
<td>SVON6</td>
<td>O</td>
<td>Multi-purpose signal, (6)</td>
</tr>
<tr>
<td>8</td>
<td>ERC5</td>
<td>O</td>
<td>Dev. ctr. clr. signal, (5)</td>
<td>42</td>
<td>ERC6</td>
<td>O</td>
<td>Dev. ctr. clr. signal, (6)</td>
</tr>
<tr>
<td>9</td>
<td>ALM5</td>
<td>I</td>
<td>Alarm signal, (5)</td>
<td>43</td>
<td>ALM6</td>
<td>I</td>
<td>Alarm signal, (6)</td>
</tr>
<tr>
<td>10</td>
<td>INP5</td>
<td>I</td>
<td>In-position signal, (5)</td>
<td>44</td>
<td>INP6</td>
<td>I</td>
<td>In-position signal, (6)</td>
</tr>
<tr>
<td>11</td>
<td>RDY5</td>
<td>I</td>
<td>Multi-purpose signal, (5)</td>
<td>45</td>
<td>RDY6</td>
<td>I</td>
<td>Multi-purpose signal, (6)</td>
</tr>
<tr>
<td>12</td>
<td>EA5+</td>
<td>I</td>
<td>Encoder A-phase (+),(5)</td>
<td>46</td>
<td>EA6+</td>
<td>I</td>
<td>Encoder A-phase (+),(6)</td>
</tr>
<tr>
<td>13</td>
<td>EA5-</td>
<td>I</td>
<td>Encoder A-phase (-),(5)</td>
<td>47</td>
<td>EA6-</td>
<td>I</td>
<td>Encoder A-phase (-),(6)</td>
</tr>
<tr>
<td>14</td>
<td>EB5+</td>
<td>I</td>
<td>Encoder B-phase (+),(5)</td>
<td>48</td>
<td>EB6+</td>
<td>I</td>
<td>Encoder B-phase (+),(6)</td>
</tr>
<tr>
<td>15</td>
<td>EB5-</td>
<td>I</td>
<td>Encoder B-phase (-),(5)</td>
<td>49</td>
<td>EB6-</td>
<td>I</td>
<td>Encoder B-phase (-),(6)</td>
</tr>
<tr>
<td>16</td>
<td>EZ5+</td>
<td>I</td>
<td>Encoder Z-phase (+),(5)</td>
<td>50</td>
<td>EZ6+</td>
<td>I</td>
<td>Encoder Z-phase (+),(6)</td>
</tr>
<tr>
<td>17</td>
<td>EZ5-</td>
<td>I</td>
<td>Encoder Z-phase (-),(5)</td>
<td>51</td>
<td>EZ6-</td>
<td>I</td>
<td>Encoder Z-phase (-),(6)</td>
</tr>
<tr>
<td>18</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>52</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>19</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>53</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>20</td>
<td>PEL5</td>
<td>I</td>
<td>End limit signal (+),(5)</td>
<td>54</td>
<td>PEL6</td>
<td>I</td>
<td>End limit signal (+),(6)</td>
</tr>
<tr>
<td>21</td>
<td>MEL5</td>
<td>I</td>
<td>End limit signal (-),(5)</td>
<td>55</td>
<td>MEL6</td>
<td>I</td>
<td>End limit signal (-),(6)</td>
</tr>
<tr>
<td>22</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>56</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>23</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>57</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>24</td>
<td>ORG5</td>
<td>I</td>
<td>Origin signal, (5)</td>
<td>58</td>
<td>ORG6</td>
<td>I</td>
<td>Origin signal, (6)</td>
</tr>
<tr>
<td>25</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
<td>59</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>26</td>
<td>AIN1</td>
<td>I</td>
<td>Analog Input (5)</td>
<td>60</td>
<td>AIN2</td>
<td>I</td>
<td>Analog Input (6)</td>
</tr>
<tr>
<td>27</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
<td>61</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>28</td>
<td>AOUT1</td>
<td>O</td>
<td>Analog Output (5)</td>
<td>62</td>
<td>AOUT2</td>
<td>O</td>
<td>Analog Output (6)</td>
</tr>
<tr>
<td>29</td>
<td>DI_COM</td>
<td>I</td>
<td>Digital Input COM+</td>
<td>63</td>
<td>DO_COM</td>
<td>I</td>
<td>Digital Output COM+</td>
</tr>
<tr>
<td>30</td>
<td>DIN5</td>
<td>I</td>
<td>Digital Input (5)</td>
<td>64</td>
<td>DIN6</td>
<td>I</td>
<td>Digital Input (6)</td>
</tr>
<tr>
<td>31</td>
<td>DOUT5</td>
<td>O</td>
<td>Digital Output (5)</td>
<td>65</td>
<td>DOUT6</td>
<td>O</td>
<td>Digital Output (6)</td>
</tr>
<tr>
<td>32</td>
<td>CMP1</td>
<td>O</td>
<td>Compare Trigger Output (5)</td>
<td>66</td>
<td>CMP2</td>
<td>O</td>
<td>Compare Trigger Output (6)</td>
</tr>
<tr>
<td>33</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>67</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>34</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
<td>68</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
</tr>
</tbody>
</table>
## 2.8 CN6 Pin Assignments: SCSI 68 Pins Connector

### Axis 7 - 8

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (7)</th>
<th>No.</th>
<th>Name</th>
<th>I/O</th>
<th>Function Axis (8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>35</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>2</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>36</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>3</td>
<td>OUT7+</td>
<td>O</td>
<td>Pulse signal (+),(7)</td>
<td>37</td>
<td>OUT8+</td>
<td>O</td>
<td>Pulse signal (+),(8)</td>
</tr>
<tr>
<td>4</td>
<td>OUT7-</td>
<td>O</td>
<td>Pulse signal (-),(7)</td>
<td>38</td>
<td>OUT8-</td>
<td>O</td>
<td>Pulse signal (-),(8)</td>
</tr>
<tr>
<td>5</td>
<td>DIR7+</td>
<td>O</td>
<td>Dir. signal (+),(7)</td>
<td>39</td>
<td>DIR8+</td>
<td>O</td>
<td>Dir. signal (+),(8)</td>
</tr>
<tr>
<td>6</td>
<td>DIR7-</td>
<td>O</td>
<td>Dir. signal (-),(7)</td>
<td>40</td>
<td>DIR8-</td>
<td>O</td>
<td>Dir. signal (-),(8)</td>
</tr>
<tr>
<td>7</td>
<td>SVON7</td>
<td>O</td>
<td>Multi-purpose signal, (7)</td>
<td>41</td>
<td>SVON8</td>
<td>O</td>
<td>Multi-purpose signal, (8)</td>
</tr>
<tr>
<td>8</td>
<td>ERC7</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (7)</td>
<td>42</td>
<td>ERC8</td>
<td>O</td>
<td>Dev. ctr, clr. signal, (8)</td>
</tr>
<tr>
<td>9</td>
<td>ALM7</td>
<td>I</td>
<td>Alarm signal, (7)</td>
<td>43</td>
<td>ALM8</td>
<td>I</td>
<td>Alarm signal, (8)</td>
</tr>
<tr>
<td>10</td>
<td>INP7</td>
<td>I</td>
<td>In-position signal, (7)</td>
<td>44</td>
<td>INP8</td>
<td>I</td>
<td>In-position signal, (8)</td>
</tr>
<tr>
<td>11</td>
<td>RDY7</td>
<td>I</td>
<td>Multi-purpose signal, (7)</td>
<td>45</td>
<td>RDY8</td>
<td>I</td>
<td>Multi-purpose signal, (8)</td>
</tr>
<tr>
<td>12</td>
<td>EA7+</td>
<td>I</td>
<td>Encoder A-phase (+),(7)</td>
<td>46</td>
<td>EA8+</td>
<td>I</td>
<td>Encoder A-phase (+),(8)</td>
</tr>
<tr>
<td>13</td>
<td>EA7-</td>
<td>I</td>
<td>Encoder A-phase (-),(7)</td>
<td>47</td>
<td>EA8-</td>
<td>I</td>
<td>Encoder A-phase (-),(8)</td>
</tr>
<tr>
<td>14</td>
<td>EB7+</td>
<td>I</td>
<td>Encoder B-phase (+),(7)</td>
<td>48</td>
<td>EB8+</td>
<td>I</td>
<td>Encoder B-phase (+),(8)</td>
</tr>
<tr>
<td>15</td>
<td>EB7-</td>
<td>I</td>
<td>Encoder B-phase (-),(7)</td>
<td>49</td>
<td>EB8-</td>
<td>I</td>
<td>Encoder B-phase (-),(8)</td>
</tr>
<tr>
<td>16</td>
<td>EZ7+</td>
<td>I</td>
<td>Encoder Z-phase (+),(7)</td>
<td>50</td>
<td>EZ8+</td>
<td>I</td>
<td>Encoder Z-phase (+),(8)</td>
</tr>
<tr>
<td>17</td>
<td>EZ7-</td>
<td>I</td>
<td>Encoder Z-phase (-),(7)</td>
<td>51</td>
<td>EZ8-</td>
<td>I</td>
<td>Encoder Z-phase (-),(8)</td>
</tr>
<tr>
<td>18</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
<td>52</td>
<td>EX+5V</td>
<td>O</td>
<td>+5V power supply output</td>
</tr>
<tr>
<td>19</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>53</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>20</td>
<td>PEL7</td>
<td>I</td>
<td>End limit signal (+),(7)</td>
<td>54</td>
<td>PEL8</td>
<td>I</td>
<td>End limit signal (+),(8)</td>
</tr>
<tr>
<td>21</td>
<td>MEL7</td>
<td>I</td>
<td>End limit signal (-),(7)</td>
<td>55</td>
<td>MEL8</td>
<td>I</td>
<td>End limit signal (-),(8)</td>
</tr>
<tr>
<td>22</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>56</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>23</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>57</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>24</td>
<td>ORG7</td>
<td>I</td>
<td>Origin signal, (7)</td>
<td>58</td>
<td>ORG8</td>
<td>I</td>
<td>Origin signal, (8)</td>
</tr>
<tr>
<td>25</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
<td>59</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>26</td>
<td>AIN3</td>
<td>I</td>
<td>Analog Input (7)</td>
<td>60</td>
<td>AIN4</td>
<td>I</td>
<td>Analog Input (8)</td>
</tr>
<tr>
<td>27</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
<td>61</td>
<td>AGND</td>
<td>O</td>
<td>Analog Ground</td>
</tr>
<tr>
<td>28</td>
<td>AOUT3</td>
<td>O</td>
<td>Analog Output (7)</td>
<td>62</td>
<td>AOUT4</td>
<td>O</td>
<td>Analog Output (8)</td>
</tr>
<tr>
<td>29</td>
<td>DI_COM</td>
<td>I</td>
<td>Digital Input COM+</td>
<td>63</td>
<td>DO_COM</td>
<td>I</td>
<td>Digital Output COM+</td>
</tr>
<tr>
<td>30</td>
<td>DIN7</td>
<td>I</td>
<td>Digital Input (7)</td>
<td>64</td>
<td>DIN8</td>
<td>I</td>
<td>Digital Input (8)</td>
</tr>
<tr>
<td>31</td>
<td>DOUT7</td>
<td>O</td>
<td>Digital Output (7)</td>
<td>65</td>
<td>DOUT8</td>
<td>O</td>
<td>Digital Output (8)</td>
</tr>
<tr>
<td>32</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>66</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>33</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
<td>67</td>
<td>EXGND</td>
<td>O</td>
<td>Ext. power ground</td>
</tr>
<tr>
<td>34</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
<td>68</td>
<td>EX+24V</td>
<td>O</td>
<td>Ext. power supply, +24V</td>
</tr>
</tbody>
</table>
## 2.9 Overview of CN3-CN6

<table>
<thead>
<tr>
<th>CN3</th>
<th>CN4</th>
<th>CN5</th>
<th>CN6</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPP 1</td>
<td>36</td>
<td>VPP 1</td>
<td>36</td>
</tr>
<tr>
<td>26</td>
<td>36</td>
<td>2</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>37</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>46</td>
<td>46</td>
<td>46</td>
<td>46</td>
</tr>
<tr>
<td>55</td>
<td>55</td>
<td>55</td>
<td>55</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>82</td>
<td>82</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>91</td>
<td>91</td>
<td>91</td>
<td>91</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>109</td>
<td>109</td>
<td>109</td>
<td>109</td>
</tr>
<tr>
<td>118</td>
<td>118</td>
<td>118</td>
<td>118</td>
</tr>
<tr>
<td>127</td>
<td>127</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>136</td>
<td>136</td>
<td>136</td>
<td>136</td>
</tr>
<tr>
<td>145</td>
<td>145</td>
<td>145</td>
<td>145</td>
</tr>
<tr>
<td>154</td>
<td>154</td>
<td>154</td>
<td>154</td>
</tr>
<tr>
<td>163</td>
<td>163</td>
<td>163</td>
<td>163</td>
</tr>
<tr>
<td>172</td>
<td>172</td>
<td>172</td>
<td>172</td>
</tr>
<tr>
<td>181</td>
<td>181</td>
<td>181</td>
<td>181</td>
</tr>
<tr>
<td>190</td>
<td>190</td>
<td>190</td>
<td>190</td>
</tr>
<tr>
<td>209</td>
<td>209</td>
<td>209</td>
<td>209</td>
</tr>
<tr>
<td>218</td>
<td>218</td>
<td>218</td>
<td>218</td>
</tr>
<tr>
<td>227</td>
<td>227</td>
<td>227</td>
<td>227</td>
</tr>
<tr>
<td>236</td>
<td>236</td>
<td>236</td>
<td>236</td>
</tr>
<tr>
<td>245</td>
<td>245</td>
<td>245</td>
<td>245</td>
</tr>
<tr>
<td>254</td>
<td>254</td>
<td>254</td>
<td>254</td>
</tr>
<tr>
<td>263</td>
<td>263</td>
<td>263</td>
<td>263</td>
</tr>
<tr>
<td>272</td>
<td>272</td>
<td>272</td>
<td>272</td>
</tr>
<tr>
<td>281</td>
<td>281</td>
<td>281</td>
<td>281</td>
</tr>
<tr>
<td>290</td>
<td>290</td>
<td>290</td>
<td>290</td>
</tr>
</tbody>
</table>

**Notes:**
- VPP: Voltage Pin Power
- CN3-CN6: Connector Numbers 3 to 6

**Installation**
## 2.10 CN1: RJ45 Connector for HSL

### RJ45 Connector (front)

<table>
<thead>
<tr>
<th>Pin</th>
<th>Signal</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HSL2_R+</td>
<td>2nd master receive data (+)</td>
</tr>
<tr>
<td>2</td>
<td>HSL2_T-</td>
<td>2nd master transmit data (-)</td>
</tr>
<tr>
<td>3</td>
<td>HSL1_R+</td>
<td>1st master receive data (+)</td>
</tr>
<tr>
<td>4</td>
<td>HSL1_T-</td>
<td>1st master transmit data (-)</td>
</tr>
<tr>
<td>5</td>
<td>HSL1_T+</td>
<td>1st master transmit data (+)</td>
</tr>
<tr>
<td>6</td>
<td>HSL1_R-</td>
<td>1st master receive data (-)</td>
</tr>
<tr>
<td>7</td>
<td>HSL2_T+</td>
<td>2nd master transmit data (+)</td>
</tr>
<tr>
<td>8</td>
<td>HSL2_R-</td>
<td>2nd master receive data (-)</td>
</tr>
</tbody>
</table>

![RJ45 Connector Diagram]
2.11 Switch Setting

Limit Switch Type (S1, S2)
- ON: Normal Open
- OFF: Normal Close

(For each axis)

HSL Half/Full Duplex Selection
- JP2: HSL #1
- JP3: HSL #1
- JP4: HSL #2
- JP5: HSL #2

HSL Half/Full Duplex Selection
- J1: HSL #1
- J2: HSL #2

Card Number Select (Optional)
The switch bits of S1 and S2 are used to set the EL limit switch's type. The default setting of EL switch type is “normal open” type limit switch (or “A” contact type). The switch on is to use the “normal closed” type limit switch (or “B” contact type). The default setting is set as normal open type.

Placement of S1,S2 switch on board

Axis No. | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
--- | --- | --- | --- | --- | --- | --- | --- | ---
S1 ON | OFF
S2 ON | OFF

'b' contact EL Switch (Normal Closed)
'a' contact EL Switch (Normal Open)
Signal Connections

Signal connections of all I/O’s are described in this chapter. Refer to the contents of this chapter before wiring any cable between the cPCI-8168 and any motor driver.

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 In-position signals INP
Section 3.6 Alarm signal ALM
Section 3.7 Deviation counter clear signal ERC
Section 3.8 General-purposed signals SVON
Section 3.9 General-purposed signal RDY
Section 3.10 Isolated Digital Output DOx
Section 3.11 Isolated Digital Input DIx
Section 3.12 High Speed High Current Digital Output HSOUT
Section 3.13 High Speed Digital Input HSIN
Section 3.14 Comparison Output CMP1 and CMP2
3.1 Pulse Output Signals OUT and DIR

There are 8-axis pulse output signals on cPCI-8168. For every axis, two pairs of OUT and DIR signals are used to send the pulse train and to indicate the direction. The OUT and DIR signals can also be programmed as CW and CCW signals pair. In this section, the electronic characteristics of the OUT and DIR signals are shown. Each signal consists of a pair of differential signals. For example, the OUT2 is consistent of OUT2+ and OUT2- signals. The following wiring diagram is for the OUT and DIR signals of the 8 axes.
3.2 Encoder Feedback Signals EA, EB and EZ

The encoder feedback signals include the EA, EB, and EZ. Every axis has six pins for three differential pairs of phase-A (EA), phase-B (EB) and index (EZ) input. The EA and EB are used for position counting, the EZ is used for zero position index. The input circuits of the EA, EB, EZ signals are shown as follows.

Please note that the voltage across every differential pair of encoder input signals (EA+, EA-), (EB+, EB-) and (EZ+, EZ-) should be at least 3.5V or higher. Therefore, you have to take care of the driving capability when connecting with the encoder feedback or motor driver feedback. The differential signal pairs will be converted to digital signal EA, EB and EZ to connect to PCL6045 ASIC.

Here are two examples of connecting the input signals with the external circuits. The input circuits can connect to the encoder or motor driver, which are equipped with: (1) differential line driver or (2) open collector output.

3.2.1 Connection to Line Driver Output

To drive the cPCI-8168 encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 6 mA driving capability. The ground level of the two sides must be tight together.
3.2.2 Connection to Open Collector Output

To connect with open collector output, an external power supply is necessary. Some motor drivers also provide the power source. The connection between cPCI-8168, encoder, and the power supply is shown in the following diagram. Please note that the external current limit resistor R is necessary to protect the cPCI-8168 input circuit. The following table lists the suggested resistor value according to the encoder power supply.

<table>
<thead>
<tr>
<th>Encoder Power(VDD)</th>
<th>External Resistor R</th>
</tr>
</thead>
<tbody>
<tr>
<td>+5V</td>
<td>0 Ω (None)</td>
</tr>
<tr>
<td>+12V</td>
<td>1.8kΩ</td>
</tr>
<tr>
<td>+24V</td>
<td>4.3kΩ</td>
</tr>
<tr>
<td>If=6mA max.</td>
<td></td>
</tr>
</tbody>
</table>

If the external current limit resistor R is necessary to protect the cPCI-8168 input circuit.
3.3 Origin Signal ORG

The origin signals (ORG1~ORG8) are used as input signals for origin of the mechanism.

The input circuits of the ORG signals are shown as following. Usually, a limit switch is used to indicate the origin of one axis. The specifications of the limit switches should with contact capacity of +24V, 6mA minimum. An internal filter circuit is used to filter out the high frequency spike, which may cause poor operation.

When the motion controller is operated at the home return mode, the ORG signal is used to stop the control output signals (OUT and DIR).
3.4 End-Limit Signals PEL and MEL

There are two end-limit signals PEL and MEL for one axis. PEL indicates end limit signals in the plus direction and MEL indicates end limit signals in the minus direction.

The signals connection and relative circuit diagram is shown in the following diagram. The external limit switches featuring a contact capacity of +24V, 6mA minimum. You can use either A-type (normal open) contact switch or B-type (normal closed) contact switch by setting the DIP switch S1. The cPCI-8168 is delivered with all bits of S1 set to OFF, refer to section 1.7.
3.5 In-position Signal INP

The in-position signals INP from the servo motor driver indicates the deviation error is zero, that is the servo position error is zero. The relative signal name, pin number and axis number are shown in the following table.

The input circuit of the INP signals are shown in the following diagram.

The in-position signals are usually from servomotor drivers, which usually provide open collector output signals. The external circuit must provide at least 5 mA current sink capability to drive the INP signal active.
3.6 Alarm Signal ALM

The alarm signal ALM is used to indicate the alarm status from the servo driver. The relative signal name, pin number and axis number are shown in the following table.

The input circuit of alarm circuit is shown in the following diagram. The ALM signals are usually from servomotor drivers, which usually provide open collector output signals. The external circuit must provide at least 5 mA current sink capability to drive the ALM signal active.
3.7 Deviation Counter Clear Signal ERC

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

1. home return is complete;
2. the 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 relative signal name, pin number and axis number are shown in the following table.

The ERC signal is used to clear the deviation counter of servomotor driver. The ERC output circuit is in the open collector with maximum 35 V external power at 50mA driving capability.

![Diagram of signal connections]
3.8 General-purpose Signal SVON

The SVON signals can be used as servomotor-on control or general-purpose output signals. The output circuit of SVON signal is shown in the following diagram.
3.9 General-purpose Signal RDY

The RDY signals can be used as motor driver ready input or general-purpose input signals. The input circuit of RDY signal is shown in the following diagram.
3.10 Isolated Digital Output DOx

The connection of isolated-digital output is shown in the following diagram. When the isolated digital output goes to high, the sink current will be from external DOUT supplied voltage. Each transistor on TD62083 is OFF when reset.

![Diagram of isolated digital output](image)

Spec. of TD62083

- Output sustaining voltage: 50V
- Output Current: 123 mA/ch (Duty=50%), 500 mA/ch (MAX)
- Clamp Diode Reverse Voltage: 50V
- Clamp Diode Forward Current: 500mA
- Power Dissipation: 1.47W (maximum)
3.11 Isolated Digital Input DIx

The isolated digital input is an open collector transistor structure. The input voltage range is from 5V to 24V and input resistor is 4.7K (1/2W). The connection between outside signal is shown below. Maximum forward current through the diode of photocoupler is +/- 50mA.
3.12 High Speed High Current Digital Output HSOUT

There are 4 channels of high speed high current digital outputs on cPCI-8168. They are available on CN3 and CN4. Connections of these outputs are shown in the following diagram. When the isolated digital output goes to high, the sink current will come from the external HSOUT supplied voltage. The time delay from HSDOx to HSOUT is within 15ns. Maximum current output ability is limited by the wire of cPCI-8168 to be 1 ampere. Each transistor on IRF7303 is OFF when reset.

Spec. of IRF7303

- Drain to Source breakdown voltage: 30V
- Continuous drain current, VGS @ 10V: 4.9A
- Clamp Diode Reverse Voltage: 50V
- Power Dissipation: 2W (maximum)
3.13 High Speed Digital Input HSIN

There are 4 channels high speed digital inputs on cPCI-8168. They are available on CN3 and CN4. Connections of these inputs are shown in the following diagram. The isolated digital input is open collector transistor structure. The Input voltage is 5V. The connection between outside signal is shown bellow. Maximum forward current through the diode of photocoupler is +/- 20mA.
3.14 Comparison Output CMP1 and CMP2

The cPCI-8168 provides two pins for position compare trigger output. The pulse width of this trigger is 100 micro seconds for most industrial CCD camera. The pin assignment and wiring are as follows:
4 Operation Theory

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

Section 4.1: Classifications of Motion Controller
Section 4.2: Motion Control Modes
Section 4.3: Motor Driver Interface
Section 4.4: Mechanical Switch Interface
Section 4.5: The Counters
Section 4.6: The Comparators
Section 4.7: Other Motion Functions
Section 4.8: Interrupt Control
Section 4.9: Multiple Cards Operation

4.1 Classifications of Motion Controller

In the early years of the servo/stepper driver, motion control drew more attention than motor control. They separate motor control into two layers: one is motor control and the other is motion control. Motor control refers to the PWM, power stage, closed loop, hall sensors, vector space, and so on. Motion control refers to the speed profile generating, trajectory following, multi-axes synchronization, and coordinating.

4.1.1 Voltage type motion control Interface

The interfaces of motion and motor control are changing rapidly. In the early years, people used voltage singals to command the motor controller. The amplitude of the signal signifies how fast the motor is rotating and the time duration of the voltage changes signifies how fast the motor accelerates from one speed to another. Motor drivers with voltage signal commands are called “analog” motion controller. It is much easier to integrate into an analog circuit motor controller, but sometimes noise is a big problem for this type of motion control. In order to control the positioning of a motor the analog type motion controller must have a position feedback signal and use a closed loop control algorithm. This increases the complexity of motion control for a beginner user.
### 4.1.2 Pulse type motion control Interface

The second interface of motion and motor control is pulse train type. The counts of pulses show how many steps a motor rotates and the frequency of pulses show how fast a motor runs. The time duration of frequency changes represents the acceleration rate of a motor. This interface is easier to use than an analog type motion controller when controlling position applications of a servo or stepper motor, allowing motion and motor control to be separated easily.

Both of these interfaces require manual gain tuning. For analog type position controllers, the control loops are built inside and users must tune the gain from the controller. For pulse type position controller, the control loops are built outside on the motor drivers and users must tune the gains on the drivers.

For more than one axes’ operation, motion control seems more important than motor control. In industrial applications, reliability is a very important factor. Motor driver vendors make good performance products and motion controller vendors make powerful controllers and offer a variety of motion software. Integrated two products make our machine go into perfect.

### 4.1.3 Network type motion control Interface

The network type motion controller is the newest control interface. With this interface the command between motor driver and motion controller is not an analog or a pulse signal. Instead, the command is a network packet containing position and motor information. This type of controller is more digitized and packetized making it more reliable than the other interfaces. Since a motion controller must be real-time, the network must have real-time capacity around a cycle time below 1 mini-second. This means that a non-commercial network can do the job. It must have a specific network like Mitsubishi SSCNET. The network may have opto-fiber type to increase communication reliability.
4.1.4 Software real-time motion control kernel

There are three ways to accomplish motion control kernel: DSP-based, ASIC based, and software real-time based.

A motion control system needs an absolute real-time control cycle and the calculation on the controller must provide control data at the same cycle. If not, the motor will not run smoothly. Many machine makers will use a PC’s computing power to do this by simply using a feedback counter card and a voltage output or pulse output card. This method is cost effective but requires a great deal of software effort. For realtime performance a real-time software is necessary, increasing the complexity of the system. However, this method is the most flexible way for a professional motion control designers. Most of these methods are on NC machines.

4.1.5 DSP based motion control kernel

A DSP-based motion control kernel solves real-time software problems on a computer. DSP is a micro-processor that can be used for all motion control calculations. DSP does not have the problems real-time software does since DSP has its own OS to arrange all procedures and there aren’t any interruptions from other inputs or context switching problems like Windows based computers. Although real-time provides optimal performance, the calculation speed is not currently as fast as a PC’s CPU. The software interfacing between the vendors of DSP based controllers and its users is difficult to use. To simplify the software interface for users some controller vendors provide an assembly language to learn while other controller vendors provide only a handshake documents. Neither method is user friendly. However, a DSP based controller is a more efficient way for machine makers to build their applications than software kernel.

4.1.6 ASIC based motion control kernel

An ASIC-base motion control kernel does not have any real-time problems since all motion functions are done via ASIC. Users or controller vendors only need to set some required parameters and the motion control will be done easily. This kind of motion control separates all system integration problems into 4 parts: motor
driver’s performance, ASIC outputting profile, vendor’s software parameters to ASIC, and users’ command to vendors’ software. ASIC-based motion control allows the motion controller to cooperate more smoothly between devices.

4.1.7 Compare Table of all motion control types

<table>
<thead>
<tr>
<th></th>
<th>Software</th>
<th>ASIC</th>
<th>DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Price</strong></td>
<td>*Fair</td>
<td>Cheap</td>
<td>Expensive</td>
</tr>
<tr>
<td><strong>Functionality</strong></td>
<td>Highest</td>
<td>Low</td>
<td>Normal</td>
</tr>
<tr>
<td><strong>Maintenance</strong></td>
<td>Hard</td>
<td>Easy</td>
<td>Fair</td>
</tr>
</tbody>
</table>

*real-time OS included

** DSP or software realtime OS is needed

4.1.8 cPCI-8168’s motion controller type

The cPCI-8168 is an ASIC based, pulse type motion controller made into three blocks: motion ASIC, PCI card, software motion library. Users can access motion ASIC via our software motion library under Windows 2000/XP, Linux, and RTX driver. Our software motion library provides one-stop-function for controlling motors. All the speed parameters’ calculations are done via our library.

For example, if users want to perform an one-axis point to point motion with a trapezoidal speed profile, they need only fill the target position, speed, and acceleration time in one function. Then the motor will run according to the profile. It does not occupy any of the CPU’s resources since every control cycle’s pulse generation is done by ASIC. The precision of target position depends on the motor drivers’ closed loop control performance and mechanical parts, not on the motion controller’s command since the motion controller is only responsible for sending correct pulse counts via a
desired speed profile, thus making it is much easier for programmers, and mechanical or electrical engineers to discover the source of a problem.

4.2 Motion Control Modes

Unlike motor control which is only for positive or negative moving, motion control make the motors run according to a specific speed profile, path trajectory and synchronous condition with other axes. The following sections describe the motion control modes of this controller.

4.2.1 Coordinate system

We use Cartesian coordinates and pulses for units of length. The physical length depends on mechanical parts and a motor’s resolution. For example, if users install a motor on a screw ball the pitch of the screw ball is 10mm, and the pulses needed for a round of motor are 10,000 pulses. We can say that one pulse’s physical unit is equal to 10mm/10,000p = 1 micro-meter.

Just set a command with 15,000 pulses for motion control if we want to move 15mm. How about if we want to move 15.0001mm? The motion controller will automatically keep the residue value less than 1 pulse and add it to next command.

The motion controller sends incremental pulses to motor drivers. Meaning that only relative commands can be sent to the motor driver. However, an absolute command can be sent by first calculating the difference between current position and target position and then sending the differences to the motor driver. For example, if current position is 1000 and we want to move a motor to 9000, the user can use an absolute command to set a target position of 9000 by adding on the command of +8000 pulses to the current 1000. The motion controller will send 8000 pulses to the motor driver to move the position to 9000.
Occasionally, users will need to install a linear scale or an external encoder to check the machine's position. If the resolution of an external encoder is 10,000 pulses per 1mm and the motor will move 1mm if the motion controller sends 1,000 pulses, then if we want to move 1 mm, we need to send 1,000 pulses to the motor driver in order to get the encoder feedback value of 10,000 pulses. If we want to use an absolute command to move a motor to 10,000 pulses and the current position read from encoder is 3500 pulses, how many pulses will it send to motor driver? The answer is \( \frac{10000 - 3500}{10,000 / 1,000} = 650 \) pulses. The motion controller will calculate this automatically when “move ratio” is set. The “move ratio” means the (feedback resolution/command resolution).

### 4.2.2 Absolute and relative position move

In the coordinate system, we have two kinds of commands for users to locate the target position. One is absolute and the other is relative. Absolute command means that the user gives the motion controller a position, then the motion controller will move a motor to that position from the current position. Relative command means the user gives the motion controller a distance, then the motion controller will move the motor the distance from current position. During the movement, users can specify the speed profile. It means user can define how fast and at what speed to reach the position.
4.2.3 Trapezoidal speed profile

Trapezoidal speed profile means the acceleration/deceleration area follows a 1st order linear velocity profile (constant acceleration rate). The profile chart is shown as below:

The area of the velocity profile represents the distance of this motion. Sometimes, the profile looks like a triangle because the desired distance from user is smaller than the area of given speed parameters. When this situation happens, the motion controller will lower the maximum velocity but keep the acceleration rate to meet user’s distance requirement. The chart of this situation is shown as below:

This kind of speed profile could be applied on velocity mode and position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.

4.2.4 S-curve and Bell-curve speed profile

S-curve means the speed profile in the accelerate/decelerate area following a 2nd order curve. It can reduce vibration when the motor is starting and stopping. In order to speed up the acceleration/deceleration during motion, we need to insert a linear part into
these areas. Once maximum velocity is met, there is a line connecting the acceleration s-curve and the deceleration s-cure, demonstrating the maintained maximum velocity over time. This graph is called a “Bell” curve. This speed profile improves the speed of acceleration and also reduces the vibration of acceleration.

For a bell curve, we define its shape’s parameter as below:

- $T_{acc}$: Acceleration time in second
- $T_{dec}$: Deceleration time in second
- $Str_{Vel}$: Starting velocity in PPS
- $Max_{Vel}$: Maximum velocity in PPS
- $VS_{acc}$: S-curve part of a bell curve in deceleration in PPS
- $VS_{dec}$: S-curve part of a bell curve in deceleration in PPS
If VSacc or VSdec=0, it means acceleration or deceleration use pure S-curve without linear part. The Acceleration chart of bell curve is shown below:

![Acceleration Chart](chart.png)

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.

This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.

4.2.5 Velocity mode

Velocity mode means the pulse command is continuously outputting until a stop command is issued. The motor will run without a target position or desired distance unless it is stopped by other reasons. The output pulse accelerates from a starting velocity to a specified maximum velocity and then followed by a linear or S-curve acceleration shape. The pulse output rate is kept at maximum velocity until another velocity command is set or a stop command is issued. The velocity could be overridden by a new speed setting. Notice that the new speed could not be a reversed to the original running speed. The speed profile for this kind of motion is shown below:
4.2.6 One axis position mode

Position mode means the motion controller will output a specific amount of pulses which is equal to users’ desired position or distance. The unit of distance or position is pulsed internally on the motion controller. The minimum length of distance is one pulse. Provided with the cPCI-8168 is a floating point function for users to transform a physical length to pulses. Inside our software library, we will keep the distance less than one pulse in register and apply it to the next motion function. Besides positioning via pulse counts, our motion controller provides three types of speed profiles to accomplish positioning. There is 1st order trapezoidal, 2nd order S-curve, and mixed bell curve. Users can call respective functions to perform that. The following chart shows the relationship between distance and speed profile. A trapezoidal speed profile is demonstrated below.

The distance is the area of the V-t diagram of this profile.
4.2.7 Two axes linear interpolation position mode

“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. Assume that we run a motion from (0,0) to (10,4). The linear interpolation results are shown below.

The pulse output from X or Y axis remains 1/2 pulse difference according to a perfect linear line. The precision of linear interpolation is shown below:

If users want to stop an interpolation group, just call a stop function on the first axis of the group.

As in the diagram below, 8-axis linear interpolation means to move the XY position 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 \( ?X : ?Y \), respectively, and the vector speed is:

\[
\frac{\Delta P}{\Delta t} = \sqrt{\left(\frac{\Delta X}{\Delta t}\right)^2 + \left(\frac{\Delta Y}{\Delta t}\right)^2}
\]

When calling 8-axis linear interpolation functions, the vector speed needs to define the start velocity, \( \text{StrVel} \), and maximum velocity, \( \text{MaxVel} \).

### 4.2.8 Two axes circular interpolation mode

Circular interpolation means XY axes simultaneously start from an initial point, \((0,0)\) and stop at an end point, \((1800,600)\). The path between them is an arc, and the MaxVel is the tangential speed. Notice that if the end point of the arc is not at a proper position, it will move circularly without stopping.

The motion controller will move to the desired final point even if this point is not on the path of arc. If the final point is not located in the shadow area of the following graph, it will run circularly without stopping.
The command precision of circular interpolation is shown below. The precision range is at radius ±1/2 pulse.

4.2.9 Continuous motion

Continuous motion means a series of motion commands or positions can be run continuously. Users can set a new command right after a previous one without interrupting it. The motion controller can make it possible because there are three command buffers (pre-registers) inside.

While the first command is still executing, users are able to set a second command into the first buffer, and a third command into the second buffer. Once the first command is finished, the motion controller will push the second command to the executing register, and the third command to first buffer. Now, the second buffer is empty and the 4th command can be set into the 2nd buffer. If
users have enough time to set a new command into the 2nd buffer before the current command is finished, the motion can run endlessly. The following diagram shows this architecture of continuous motion.

In addition to the position command, the speed command should be set correctly to perform a continuous speed profile. In the following diagram, the continuous motion consists of three motion commands. The second command has a higher speed than the two. Notice how the interconnections of speed between these three motion functions are set:

1\textsuperscript{st} command's $T_{\text{dec}} = 0$
2\textsuperscript{nd} command's $\text{StrVel} = 1\textsuperscript{st}$ command's $\text{MaxVel}$
3\textsuperscript{rd} command's $T_{\text{acc}} = 0$
3\textsuperscript{rd} command's $\text{MaxVel} = 2\textsuperscript{nd}$ command's $\text{StrVel}$
The settings for when the 2nd command’s speed value is lower than the other two are displayed in the following diagram:

![Diagram showing speed settings for 2nd command](image)

(1) $T_{acc}=0$
(2) $T_{dec}=0$
(3) $2^{nd}$ command's MaxVel = 1$^{st}$ command's StrVel
(4) $2^{nd}$ command's MaxVel = 2$^{nd}$ command's StrVel

The same concept is used with an 8-axis continuous arc interpolation. Users are able to set the speed matched between two command's speed setting.

If the INP checking is enabled, there will be a delay in the motion in between each command in the buffers. Having the INP check enabled allows the desired position to be reached, but reduces the smoothness between each command. Turn INP off when a delay is unnecessary and smooth movement is necessary.
### 4.2.10 Home Return Mode

Home return means searching a zero position point on the coordinate. Sometimes, users use a ORG, EZ or EL pin as a zero position on the coordinate. When a machine is starting up, the program needs to find a zero point. Our motion controller provides a home return mode to set the zero point.

We have many home modes and each mode contains many control phases. All of these phases are done by ASIC, requiring no software efforts or CPU loading. After home return is complete, the target counter will be reset to zero as the desired position of home mode. Consider a raising edge when in ORG input. Sometimes, the motion controller will still output pulses to make the machine slow down after resetting the counter. When the motor stops, the counter may not be at the zero point, but the home return procedure is finished. The counter value you see is a reference position from the machine’s existing zero point.

The following diagrams show the various home modes: R means counter reset (command and position counter). E means ERC signal output.
Home mode=0: (Turn ON ORG then reset counter)

When SD is not installed

When SD is installed and SD is not latched
Home mode=1: (Turn ON ORG twice then reset counter)

Home mode=2: (Turn ORG ON then Slow down to count EZ numbers and reset counter)
Home mode=3: (Turn ORG ON then count EZ numbers and reset counter)

Home mode=4: (Turn ORG On then reverse to count EZ number and reset counter)
Home mode=5: (Turn ORG On then reverse to count EZ number and reset counter, not using FA Speed)

Home mode=6: (Turn EL On then reverse to leave EL and reset counter)

Home mode=7: (Turn EL On then reverse to count EZ number and reset counter)
Home mode=8: (Turn EL On then reverse to count EZ number and reset counter, not using FA Speed)

Home mode=9: (Turn ORG On then reverse to zero position, an extension from mode 0)
Home mode=10: (Turn ORG On then counter EZ and reverse to zero position, an extension from mode 3)

Home mode=11: (Turn ORG On then reverse to counter EZ and reverse to zero position, an extension from mode 5)
Home mode=12: (Turn EL On then reverse to count EZ number and reverse to zero position, an extension from mode 8)

4.2.11 Home Search Function

This mode is used to add auto searching function on normal home return mode described in the previous section no matter which position the axis is. The following diagram shows the example for home mode 2 via home search function. The ORG offset can’t be zero. The suggested value is twice the length of ORG area.
4.2.12 Manual Pulser Function

A manual pulser is a device to generate pulse trains by hand. The pulses are sent to the motion controller and re-directed to pulse output pins. The input pulses can be multiplied or divided before sending out.

The motion controller receives two kinds of pulse trains from a manual pulser device: CW/CCW and AB phase. If the AB phase input mode is selected and the multiplier has the additional selections of 1, 2, or 4.

The following figure shows a pulser ratio block diagram.

4.2.13 Simultaneous Start Function

Simultaneous motion means more than one axis can be started by a simultaneous signal which could be external or internal signals. For external signals, users must set move parameters for all axes first, then these axes will wait for an external start/stop command to start or stop. For internal signals, the start command could come from a software start function. Once it is issued, all axes which are in waiting synchronous mode will start at the same time.
4.2.14 Speed Override Function

Speed override means that users can change a command’s speed during the operation of motion. The changed parameter is a percentage of the speed originally defined. Users can define a 100% speed value then change the speed by a percentage of the original speed while the machine is still in motion. If users didn’t define the 100% speed value. The default 100% speed is the latest motion command’s maximum speed. This function can be applied to any motion function. If the running motion is an S-curve or bell curve, the speed override will be a pure s-curve. If the running motion is a t-curve, the speed override will be a t-curve.

![Speed Override Diagram]

4.2.15 Position Override Function

Position override is when users want to change the target position of the original positioning command during its operation. If the new target position is behind the current position when the override command is issued, the motor will slow down and then reverse to the new target position. If the new target position is far away from current position but in the same direction, the motion will maintain its speed and move to the new target position. If the override timing requires deceleration from the current motion, and the target position is far away from the current position but the same direction, motion will accelerate to the original speed and run to the new target position. The operation examples are shown below.
Notice that if the new target’s position’s relative pulses are smaller than the original slow down pulses, this function can’t work properly.

4.3 The motor driver interface

We provide several dedicated I/Os which can be connected to the motor driver directly and have their own functions. Motor drivers have many kinds of I/O pins for an external motion controller to use. We classify them in two groups. One is pulse I/O signals, including pulse command and encoder interface. The other is digital I/O signals, including servo ON, alarm, INP, servo ready, alarm reset and emergency stop inputs. The following sections will describe the functions of these I/O pins.

4.3.1 Pulse Command Output Interface

The motion controller uses a pulse command to control servo/stepper motors via motor drivers. To use the position command function the drivers must be set to position mode. The pulse command consists of two signal pairs. It is defined as OUT and DIR pins on a connector. Each signal has two pins as a pair for differential output. There are two signal modes for pulse output command: (1) single pulse output mode (OUT/DIR), and (2) dual pulse output mode (CW/CCW type pulse output). The mode must be
same as motor driver. The modes vs. signal type of OUT and DIR pins are listed in the table below:

<table>
<thead>
<tr>
<th>Mode</th>
<th>Output of OUT pin</th>
<th>Output of DIR pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dual pulse output (CW/CCW)</td>
<td>Pulse signal in plus (or CW) direction</td>
<td>Pulse signal in minus (or CCW) direction</td>
</tr>
<tr>
<td>Single pulse output (OUT/DIR)</td>
<td>Pulse signal</td>
<td>Direction signal (level)</td>
</tr>
</tbody>
</table>

**Single Pulse Output Mode (OUT/DIR Mode)**

In this mode, the OUT pin is for outputing command pulse chain. The numbers of OUT pulse represent distance in pulse. The frequency of the OUT pulse represents speed in pulse per second. The DIR signal represents command direction of positive (+) or negative (-). The diagrams below show the output waveform. It is possible to set the polarity of the pulse chain.

**Pulse mode = 0: (OUT pin normally high)**

![Diagram for Pulse mode = 0](image)

**Pulse mode = 1: (OUT pin normally low)**

![Diagram for Pulse mode = 1](image)

**Pulse mode = 2: (OUT pin normally high)**

![Diagram for Pulse mode = 2](image)

**Pulse mode = 3: (OUT pin normally low)**

![Diagram for Pulse mode = 3](image)
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. The number of pulses represent distance. The frequency of the pulse represents speed per second. Pulse 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.

Pulse outmode = 4: (Pulse is normally high)

\[ \text{OUT} \quad \text{DIR} \]
\[ \begin{array}{c}
\text{CW} \\
(+) \\
CCW \\
(-)
\end{array} \]

Pulse outmode = 5: (Pulse is normally low)

\[ \text{OUT} \quad \text{DIR} \]
\[ \begin{array}{c}
\text{CW} \\
(+) \\
CCW \\
(-)
\end{array} \]

The command pulses are counted by a 28-bit command counter. The command counter can store a value of total pulses put out from the controller.

4.3.2 Pulse feedback input interface

The cPCI-8168 provides one a position counter, a 28-bit up/down counter on each axis for pulse feedback counting. The position counter counts pulses from the EA and EB signals, which have plus and minus pins on the connector for differential signal inputs. It accepts two kinds of pulse types. One is dual pulses input (CW/CCW mode) and the other is AB phase input. The AB phase input can be multiplied by 1, 2 or 4. The AB phase multiplied by 4 mode is the most commonly used in incremental encoder inputs.
For example, if a rotary encoder has 2000 pulses per rotation, then the counter value read from the position counter will be 8000 pulses per rotation when the AB phase is multiplied by four.

If users don’t use an encoder for motion controller, the feedback source for this counter must be set as a pulse command output or the counter value will always be zero. If it is set as a pulse command output, users can get the position counter value from pulse command output counter because the feedback pulses are internally counted from command output pulses.

The following diagrams show these two types of pulse feedback signals.

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

![Diagram of Plus and Minus Pulses Input Mode](image)

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 pins cause the counter to count up, whereas EB pin 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 to 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 the EA signal leads the phase of the EB signal.
The index input (EZ) signal is the zero reference in a linear or rotary encoder. The EZ can be used to define the mechanical zero position of the mechanism. The signal logic must also be set correctly in order to get a correct result.

### 4.3.3 In position signal

The in-position signal is an output signal from the motor driver. It tells the motion controller a motor has reached a position within a predefined error. The predefined error value is an in-position value, commonly referred to as an INP value. After the motion controller issues a positioning command, the motion busy status will keep true until the INP signal is ON. Users can disable the INP check for motion busy flag. If it is disabled, the motion busy will be FALSE when the pulse command is sent.
4.3.4 Servo alarm signal

The alarm signal is an output signal from the motor driver. It tells the motion controller that there is an error inside the servo motor or driver. Once the motion controller receives this signal, the pulse commands will stop sending and the ALM signal will be ON. The reason for the alarm could be that the servo motor is over speed, over current, over loaded and so on. Please check the motor driver’s manual about the details.

The logic of an alarm signal must be set correctly. If the alarm logic’s setting is not the same as the motor driver’s setting, the ALM status will always be ON and the pulse command can never be outputted.

4.3.5 Error clear signal

The ERC signal is an output from the motion controller. It tells the motor driver to clear the error counter. The feedback position will always have a delay from the command position. The error counter counts and displays the difference between command pulses and feedback pulses. Motor driver use the error counter as a basic control index. The larger the error counter value, the faster the motor speed command will be set. If the error counter is zero, then zero motor speed command will be set.

In the following four situations, the ERC signal will be outputted automatically from the motion controller to the motor driver in order to clear the error counter at the same time.

1. Home return is complete
2. The end-limit switch is touched
3. An alarm signal is active
4. An emergency stop command is issued

4.3.6 Servo ON/OFF switch

The servo on/off switch is a general digital output signal on the motion controller. It is defined as an SVON pin on the connector and is used to switch to and from the motor driver’s controlled state. When servo switch is on the motor will be held in place in place as the control loop of driver is active. When the axis is verti-
cally installed and the servo signal is turned off, the axis will be in uncontrolled state and can fall down to the ground. Some situations, like servo alarm and emergency signal ON, can have the same negative result.

4.3.7 Servo Ready Signal

The servo ready signal is a general digital input on the motion controller and has no relative purpose to the motion controller. Users can connect this signal to the motor driver’s RDY signal to check if the motor driver is in a ready state. This signal can let users know if the motor driver’s power has been inputed or not, or it can be connected as a general input for another purpose. This signal has no effect on motion control.

4.3.8 Servo alarm reset switch

The servo driver will raise an alarm signal if there is something wrong inside the servo driver. Some alarm situations are the servo motor is over current, over speed, over loading and so on. Power reset can clear the alarm status but users usually don’t want to power off the servo motor while in operation. There is one servo driver pin used to reset the alarm status. cPCI-8168 provides one general output pin for each axis to be used to reset servo alarm status.

4.4 Mechanical switch interface

We provide some dedicated input pins for mechanical switches like original switch (ORG), plus and minus end-limit switch (?EL), slow down switch (SD), positioning start switch (PCS), counter latch switch (LTC), emergency stop input (EMG) and counter clear switch (CLR). These switches’ response time is very fast, only a few ASIC clock times. There is no real-time problem when using these signals. All functions are done by motion ASIC. The software does not need to do anything other than wait for the results.
4.4.1 Original or home signal

cPCI-8168 provides one original or home signal per axis. This signal is used for defining the axes zero position. The logic of this signal must be set properly before performing a home procedure. Please refer to the home mode section for details.

4.4.2 End-Limit switch signal

The end-limit switches are usually installed on both ends of an axis. Plus EL must be installed at the positive position of the axis and minus EL at the negative position of the axis. If the moving part of a motor touches one of the end-limit switch signals, the motion controller will stop sending pulses and will output an ERC signal. The purpose of these two switch signals is to prevent the machine from crashing during misoperation. If switches are installed reversely, the protection will be invalid.

4.4.3 Slow down switch

The slow down signals can be used to force the command pulse to decelerate to the starting velocity. This signal helps protect a mechanical part moving at high speed from exceeding the mechanism's limit. The SD signal is effective for both plus and minus directions.

4.4.4 Positioning Start switch

The positioning start switch is used to move from a specific position when the machine is turned on. The function is demonstrated in the diagram below.
4.4.5 Counter Clear switch

The counter clear switch is an input signal used to reset the counters of the motion controller. If a counter needs to be reset according to an external command, this pin can be used as a control source.

4.4.6 Counter Latch switch

The counter latch switch is an input signal used to tell the counter value at the active moment of one input.

4.4.7 Emergency stop input

cPCI-8168 has a global digital input that can stop all axes motion immediately to prevent machine damage in emergency situations. Users can connect an emergency stop button to this input on their machine, although it is suggested that this input is kept as a normal closed type for safety.

4.5 The Counters

There are four counters per axis on the cPCI-8168. They are as follows:

- **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**: can be configured as command position, feedback position, manual pulser, or half of ASIC clock.
- **Target position recorder**: A software-maintained target position value of latest motion command.
4.5.1 Command position counter

The command position counter is a 28-bit binary up/down counter. Its input source is the output pulses from the motion controller. It informs the user of the current command position and is useful for debugging the motion system.

The cPCI-8168 is an open loop type motion controller. The motor driver receives pulses from motion controller. When the driver is not moving, we can check this command counter and see if there is an updated value. If it is, it means that the pulses have been sent and the problem could be on the motor driver. In this situation the motor driver’s pulse receiving counter should be checked.

The unit of command counter is in pulse. The counter value could be reset by a counter clear signal or home function completion. Users can also use a software command counter setting function to reset it.

4.5.2 Feedback position counter

The feedback position counter is a 28-bit binary up/down counter. Its input source is the input pulses from the EA/EB pins. It counts the motor position from motor’s encoder output. This counter could be set from a command position when there are no external encoder inputs.

The command output pulses and feedback input pulses will not always be the same ratio in mini-meters. Users must set the ratio if these two pulses are not 1:1.

Since the cPCI-8168 is not a closed-loop type motion controller the feedback position counter is just for reference after motion begins. The position closed-loop is done by the servo motor driver. If the servo driver is well tuned and the mechanical parts are well assembled, the total position error will remain in an acceptable range after the motion command is finished.
4.5.3 Command and Feedback error counter

The command and feedback error counter is used to calculate the error between the command position and the feedback position. The value is calculated by subtracting the feedback position from the command position.

If the ratio between command and feedback is not 1:1, the error counter is meaningless.

This counter is a 16-bit binary up/down counter.

4.5.4 General purpose counter

The source of general purpose counter could be any of the following:

1. Command position output – the same as a command position counter
2. Feedback position input – the same as a feedback position counter
4. Clock Ticks – Counter from a timer about 9.8 MHz

4.5.5 Target position recorder

The target position recorder is used during continuous motion to let the motion controller know the previous motion command's target position and the current motion command's target position in order to calculate relative pulses of current command then send results into pre-register. Please check if the target position is the same with current command position before continuous motion. Especially after the home function and stop function.
4.6 The Comparators

There are 5 counter comparators of each axis:

1. Positive soft end-limit comparator to command counter
2. Negative soft end-limit comparator to command counter
3. Command and feedback error counter comparator
4. General comparator for all counters
5. Trigger comparator for all command and feedback counters

4.6.1 Soft end-limit comparators

There are two comparators for end-limit function on each axis. We call them for the soft end-limit comparators. One is for plus (positive) end-limit and the other is for minus (negative) end-limit. The end-limit prevents a machine crash during travel. Soft limit can be used instead of a real end-limit switch. Notice that these two comparators only compare the command position counter. Once the command position is over the limited set inside the positive or negative comparators, it will stop moving as it touches the end-limit switch.

4.6.2 Command and feedback error counter comparators

This comparator is only for command and feedback counter errors. This comparator can be used to check if the error is too large. When this condition is met an action can be set, such as generating interrupt, immediate stop, and deceleration to stop.

4.6.3 General comparator

With general comparator the source of comparison can be chosen from command, feedback position counter, error counter or general counter. The compare methods could be chosen by equal, greater than or less than with direction or directionless. Once the condition is met the action can also be chosen from generating interrupt, stop motion or others.
4.6.4 Trigger comparator

The trigger comparator is much like the general comparator only with the ability to generate a trigger pulse when the condition is met. Once the condition is met, the CMP pin on the connector will output a pulse for a specific purpose, like triggering a camera to catch a picture. Not all axes have this function only those with a CMP pin. The following diagram shows the application of triggering.

In this application, the table is controlled by the motion command, and the CCD Camera is controlled by the CMP pin. When the comparing position is reached, the pulse will be outputted and the image is captured. This is an on-the-fly image capture. If users want to get more images during the motion path, try to set a new comparing point right after the previous image is captured. This method allows continuous image capturing.
4.7 Other Motion Functions

Described in the following sections are the many other functions included with the cPCI-8168 motion controller.

4.7.1 Backlash compensation and slip corrections

The motion controller has backlash and slip correction functions. These functions output the number of command pulses in FA speed. The backlash compensation is performed each time when the direction changes on operation. The slip correction function is performed before a motion command, regardless of the direction. The correction amount of pulses can be set by the function library.

4.7.2 Vibration restriction function

The vibration restriction function adds one pulse in the reverse direction and then one pulse in the forward direction shortly after completing a motion command. The timing of these two dummy pulses are shown below: (RT is reverse time and FT is forward time)
4.7.3 Speed profile calculation function

Several speed parameters must be set for motion function. Sometimes parameters conflict in a speed profile. For example, if a very fast speed and a very short distance is inputed, there is no profile for these parameters. In this situation, the motion library will maintain the acceleration and deceleration rate attempting to automatically lower the high speed originally inputed to form a feasible speed profile. The following diagram shows this concept.

The motion library has a series of functions that can be used to find the actual speed profile developed from the original commands from the user.
4.8 Interrupt Control

The cPCI-8168 motion controller can generate an interrupt signal to the host PC. The parameter “intFlag” of the software function called _8168_int_control() is to enable/disable the interrupt service. There are two kinds of interrupt groups in the cPCI-8168. One is for motion control and the other is for FPGA interrupt. The major interrupt sources from FPGA are timer interrupt and DI change of state interrupt. As for motion control interrupt, it is divided into two groups. One is for event interrupt which is maskable by _8168_int_control() and the other is for error interrupt which isn’t maskable and it only can be turned off by _8168_int_control().

The cPCI-8168 DLL and driver use Windows event object as a media for carrying hardware interrupt. Once the interrupt is enabled by users, the corresponding events for interrupt signals are created. Once the interrupt is happened, the ISR in device driver will translate the source of interrupts and trigger the corresponding event. The event will be in signal state for user’s program to received. The event type is created as ‘manual reset’ mode. Don’t forget to reset it after it is received.
4.9 Multiple Card Operation

The motion controller allows more than one card in one system. Since the motion controller is plug-and-play compatible, the base address and the IRQ setting of the card are automatically assigned by the PCI BIOS during system startup. Users don’t need and can’t change the resource settings.

When multiple cards are applied to a system, the number of cards must be noted. The card number depends on the card ID switch setting on the board. The axis number is dependant on the card ID. For example, if three motion controller cards are plugged in to PCI slots, and the corresponding card ID is set, then the axis number on each card will be:

Axis No.

<table>
<thead>
<tr>
<th>Card ID No.</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>U</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td>3</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

Notice that if multiple cards have the same card ID the function will not work correctly.
5 Motion Creator

After installing all the hardware properly according to Chapter 2 and 3, it is necessary to correctly configure cards and double check before running. This chapter gives guidelines for establishing a control system and manually exercising the cPCI-8168 cards to verify correct operation. Motion Creator provides a simple yet powerful means to setup, configure, test and debug motion control systems using cPCI-8168 cards.

Note that Motion Creator is available only for Windows 95/98 or Windows NT with the screen resolution higher than 800x600 environment and cannot run on DOS.

5.1 Execute Motion Creator

After installing the software driver of cPCI-8168 on Windows 95/98/NT/2000, the motion creator program can be found in <chosen path>/cPCI-8168/Utility. To execute it, double click it or use desktop “Start” > “Program files” > “cPCI8168” > “Motion Creator”.
5.2 About Motion Creator

Before Running Motion Creator for cPCI-8168, the following issues should be kept in mind.

1. MotionCreator is a software utility written by VB 5.0, and is available only for Windows with a screen resolution higher than 800x600 environment and can not run on DOS.

2. Motion Creator allows users to save settings or configurations for cPCI-8168 cards and those saved configurations will be loaded automatically when motion creator is executed later again. The two files 8168.ini and 8168MC.ini in windows root directory are used to save all settings and configurations.

3. To duplicate configurations from one system to another, just copy 8168.ini and 8168MC.ini into windows root directory.

4. If users want to use the configurations set by Motion Creator, the DLL function call is "_8168_config_from_file" is helpful. After calling this function users can use the configurations set by the Motion Creator for their cPCI-8168 cards.
5.3 Motion Creator Form Introducing

5.3.1 Initial forms

The following Figure shows the initial view after running Motion Creator.
5.3.2 Operation Form

There are two cPCI-8168 operating forms in Motion Creator. They are “Home Move” and “Single Axis Operation”. To activate the Home Move form, click the “Home Move” button in “Main” form. To activate the Single Axis Operation form, click the “Single Axis Operation” button in “Main” form.

A. I/O Status: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is _8168_get_io_status().

B. Position:

- Command: display value of command counter. The related function is _8168_get_command().
- Feedback: display value of feedback position counter. The related function is _8168_get_position()
- Pos Error: display value of position error counter. The related function is _8168_get_error_counter().
- Target Pos: display value of target position recorder. The related function is _8168_get_target_pos().
C. Position Reset Button:
Sets all position counter to specified value. The related functions are:

- _8168_set_position()
- _8168_set_command()
- _8168_reset_error_counter()
- _8168_reset_target_pos()

D. Motion Status: display return value of _8168_motion_done function. The related function is _8168_motion_done().

E. INT Status:

- Event: display of event_int_status in Hex value. The related function is _8168_get_int_status().
- Error: display of error_int_status in Hex value. The related function is _8168_get_int_status().
- Count: total count of interrupt.
- INT Reset Button: this button clears all INT status and INT counter to ‘0’.

F. Velocity: The absolute value of velocity in unit of PPS. The related function is _8168_get_current_speed().

G. Operation Mode: Select operation mode.

- Absolute Mode: “Position1” and “position2” will be used as absolution target position for motion. The related function is _8168_start_ta_move(), _8168_start_sa_move().
- Relative Mode: “Distance will” be used as relative displacement for motion. The related function is _8168_start_tr_move(), _8168_start_sr_move().
- Cont. Move: Velocity motion mode. The related function is _8168_tv_move(), _8168_start_sv_move().
Home Move: Home move configuration form will be invoked. Allows user to set home move configuration. The related function is _8168_set_home_config()

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

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

J. Repeat Mode: When “On” is selected, the motion will go in repeat mode (forward <-> backward or position1 <-> position2). It is only effective when “Relative Mode” or “Absolute Mode” is selected.
K. Vel. Profile: Select the velocity profile. Both Trapezoidal and S-Curve are available for “Absolute Mode”, “Relative Mode” and “Cont. Move”.

L. Motion Parameters: Set the parameters for single axis motion. These parameter are meaningless if “Manual Pulse Move” is selected, since the velocity and moving distance is decided by pulser input.

- Start Velocity: Set the start velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -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 negative direction.
- Maximum Velocity: Set the maximum velocity of motion in unit of PPS. In “Absolute Mode” or “Relative Mode”, only the value is effective. ie, -5000.0 is the same as 5000.0. In “Cont. Move”, both the value and sign are effective. –5000.0 means 5000.0 in the negative direction.
- Accel. Time : Set the acceleration time in units of seconds.
- Decel. Time : Set the deceleration time in units of seconds.
- SVacc: Set the S-curve range during acceleration in units of PPS.
- SVdec: Set the S-curve range during deceleration in units of PPS.
- Move Delay: This setting is effective only when repeat mode is set “On”. It will cause cPCI-8168 to delay specified time before it continues to the next motion.

M. 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.

N. Servo On: Set the SVON signal output status. The related function is _8168_set_servo().
O. Play Key:

**Left play button**: Starts cPCI-8168 outlet pulses according to the following settings.

- In “Absolute Mode”, the axis moves to position1.
- In “Relative Mode”, the axis moves forward.
- In “Cont. Move”, the axis moves according to the velocity setting.
- In “Manual Pulser Move”, the axis goes into pulser move. The speed limit is the value set by “Maximum Velocity”
- In “Home Mode”, the axis returns home.

**Right play button**: Starts cPCI-8168 outlet pulses according to the following settings.

- In “Absolute Mode”, the axis moves to position2.
- In “Relative Mode”, the axis moves backward.
- In “Cont. Move”, the axis moves in the opposite direction according to the velocity setting.
- In “Manual Pulser Move”, the axis goes into pulser move. The speed limit is the value set by “Maximum Velocity”
- In “Home Mode”, it cause axis starting home return.

P. **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 _8168_v_change()

Q. **Change Velocity On The Fly Button**: Changes the velocity of the current motion. The new velocity must be defined in “Maximum Velocity”. The related function is _8168_p_change().

R. **Stop Button**: Decelerates the cPCI-8168 to a stop. The deceleration time is defined in “Decel. Time”. The related function is _8168_sd_stop().
S. **Show Velocity Curve Button**: Opens a window showing the velocity vs. time curve. In this curve, every 100ms a new velocity data will be added in. To close it, click this button again. To clear data, click on the curve.

![Velocity Curve](image)

T. **Next Axis**: Click this button to change operating axis.

U. **Save Config**: Click this button to save current configuration to 8168.ini.

V. Switch to Interface I/O configuration form (5.3.3)

W. Switch to Pulse I/O and INT factor configuration form (5.3.4)

X. **Back**: Click this button to go back “main” form.
5.3.3 Interface I/O configuration form

In this form user can set the configuration of EL, ORG, EZ, ERC, ALM, INP, SD, and LTC.

A. ALM Logic and Response mode: Select logic and response mode of ALM signal. The related function call is _8168_set_alm().

B. INP Logic and Enable/Disable selection: Select logic and Enable/Disable the INP signal. The related function call is _8168_set_inp().

C. ERC Logic and Active timing: Select the Logic and Active timing of ERC signal. The related function call is _8168_set_erc().

D. EL Response mode: Select the response mode of EL signal. The related function call is _8168_set_el().

E. ORG Logic: Select the logic of ORG signal. The related function call is _8168_set_home_config().

F. EZ Logic: Select the logic of EZ signal. The related function call is _8168_set_home_config().

G. LTC Logic: Select the logic of LTC signal. The related function call is _8168_set_ltc_logic().

H. SD Configuration: Configuration of SD signal. The related function call is _8168_set_sd().

I. Next Axis: Click this button to change operating axis.
J. **Save Config** : Click this button to save current configuration to 8168.ini.

K. Switch Operate form (5.3.2)

L. Switch to Pulse I/O and INT factor configuration form (5.3.4)

M. **Back** : Click this button to go back to the “main” form.
5.3.4 Pulse I/O and INT factor configuration form

In this form users can set the configuration of pulse input/output, move ration, and INT factor.

A. Pulse Output Mode: Select the output mode of pulse signal (OUT/DIR). The related function call is _8168_set_pls_outmode().

B. Pulse Input: Set the configurations of Pulse input signal(EA/EB). The related function call is _8168_set_pls_iptmode(), _8168_set_feedback_src().

C. Move Ratio: Set the move ratio (feedback / pulse command) for current target axis. The value should not be ‘0’. The related function call is _8168_set_move_ratio().

D. INT Factor: Select factors to initiate the event int. The related function call is _8168_set_int_factor().

E. Next Axis: Click this button to change operating axis.

F. Save Config: Click this button to save current configuration to 8168.ini.

G. Switch Operate form (5.3.2)

H. Switch to Interface I/O configuration form (5.3.3)

I. Back: Click this button to go back “main” form.
5.3.5 HSLink testing form

In this form users can test the HSLink system in cPCI-8168.

- **“Current Select Card ID”**: The total master cards will display in the “Current Select Card ID” item. User can use it to specify the card ID he wants to operate.

- **“Current Select Set ID”**: 0, for the first HSLink set; 1, for the second.

- **“Connect / Auto Scan”**: Connects the master card and slave I/O modules, then scans all the slave I/O modules in the “Current Select Set ID” of the “Current Select Card ID”. This utility will show all the slave I/O module’s attributes (include the address and slave type) on the screen.

- **“Slaves Disconnect”**: This button disconnects the master card and slave I/O modules.
“Monitor”: When users press “Connect / Auto Scan” button, all slave I/O modules in the “Current Select Set ID” of the “Current Select Card ID” will be displayed on the screen. Now user can press the “Monitor” button to enter the “HSL System Monitor” window.

“Page Up”: Displays previous 9 slave I/O modules.

“Page Down”: Displays the next 9 slave I/O modules.

“Output”: Allows users to key in the “Digital Output Value” to be outputed to the specified slave I/O module.
“Test Slave”: When user press “Connect / Auto Scan” button, all slave I/O modules in the “Current Select Set ID” of the “Current Select Card ID” will display in the screen. Now user can select a slave I/O module in the screen, then press the “Test Slave” button to enter the Slave testing window (The following is a testing utility for HSL DI16DO16 module)
5.3.6 GPIO testing form

In this form user can test the GPIO function in cPCI-8168.

A. Digital output
B. Digital input
C. High speed digital output
D. High speed digital input
E. Analog output, range from –10 to +10 V
F. Analog input, range from –10 to +10 V
G. Next Card: Click this button to change operating card.
H. Back: Click this button to go back to the “main” form.
6 Function Library

This chapter describes the supporting software for cPCI-8168 cards. These functions can be used to develop application program in C or Visual Basic or C++ language. If Delphi is used the programming environment, it is necessary to transform the header file,pci_8168.h, manually.
### 6.1 List of Functions

#### Initialization: Section 6.3

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_initial</td>
<td>Software initialization</td>
</tr>
<tr>
<td>_8168_close</td>
<td>Software Close</td>
</tr>
<tr>
<td>_8168_get_base_addr</td>
<td>Get base address of cPCI-8168</td>
</tr>
<tr>
<td>_8168_get_irq_channel</td>
<td>Get the cPCI-8168 card’s IRQ number</td>
</tr>
<tr>
<td>_8168_config_from_file</td>
<td>Configure cPCI-8168 cards according to configuration file ie. 8168.ini, which is created by Motion Creator.</td>
</tr>
<tr>
<td>_8168_get_fpga_version</td>
<td>Get the FPGA firmware version</td>
</tr>
</tbody>
</table>

#### Pulse Input/Output Configuration: Section 6.4

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_pls_outmode</td>
<td>Set pulse command output mode</td>
</tr>
<tr>
<td>_8168_set_pls_iptmode</td>
<td>Set encoder input mode</td>
</tr>
<tr>
<td>_8168_set_feedback_src</td>
<td>Set counter input source</td>
</tr>
</tbody>
</table>

#### Velocity mode motion: Section 6.5

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_tv_move</td>
<td>Accelerate an axis to a constant velocity with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_sv_move</td>
<td>Accelerate an axis to a constant velocity with S-curve profile</td>
</tr>
<tr>
<td>_8168_v_change</td>
<td>Change speed on the fly</td>
</tr>
<tr>
<td>_8168_sd_stop</td>
<td>Decelerate to stop</td>
</tr>
<tr>
<td>_8168_emg_stop</td>
<td>Immediately stop</td>
</tr>
<tr>
<td>_8168_fix_speed_range</td>
<td>Define the speed range</td>
</tr>
<tr>
<td>_8168_unfix_speed_range</td>
<td>Release the speed range constrain</td>
</tr>
<tr>
<td>_8168_get_current_speed</td>
<td>Get current speed</td>
</tr>
</tbody>
</table>
### Single Axis Position Mode: Section 6.6

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_start_tr_move</td>
<td>Begin a relative trapezoidal profile move</td>
</tr>
<tr>
<td>_8168_start_ta_move</td>
<td>Begin an absolute trapezoidal profile move</td>
</tr>
<tr>
<td>_8168_start_sr_move</td>
<td>Begin a relative S-curve profile move</td>
</tr>
<tr>
<td>_8168_start_sa_move</td>
<td>Begin an absolute S-curve profile move</td>
</tr>
<tr>
<td>_8168_set_move_ratio</td>
<td>Set the ratio of command pulse and feedback pulse</td>
</tr>
<tr>
<td>_8168_p_change</td>
<td>Change position on the fly</td>
</tr>
<tr>
<td>_8168_set_pcs_logic</td>
<td>Set the logic of PCS (Position Change Signal)</td>
</tr>
<tr>
<td>_8168_set_sd_pin</td>
<td>Set SD/PCS pin</td>
</tr>
<tr>
<td>_8168_backlash_comp</td>
<td>Set backlash corrective pulse for compensation</td>
</tr>
<tr>
<td>_8168_suppress_vibration</td>
<td>Set vibration suppressing timing</td>
</tr>
</tbody>
</table>

### Linear Interpolated Motion: Section 6.7

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_start_tr_move_xy</td>
<td>Begin a relative 2-axis linear interpolation for X &amp; Y (Z &amp; U), (A &amp; B), (C &amp; D) with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_tr_move_zu</td>
<td></td>
</tr>
<tr>
<td>_8168_start_tr_move_ab</td>
<td></td>
</tr>
<tr>
<td>_8168_start_tr_move_cd</td>
<td>Begin a relative 2-axis linear interpolation for X &amp; Y (Z &amp; U), (A &amp; B), (C &amp; D) with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_ta_move_xy</td>
<td></td>
</tr>
<tr>
<td>_8168_start_ta_move_zu</td>
<td></td>
</tr>
<tr>
<td>_8168_start_ta_move_ab</td>
<td></td>
</tr>
<tr>
<td>_8168_start_ta_move_cd</td>
<td>Begin an absolute 2-axis linear interpolation for X &amp; Y (Z &amp; U), (A &amp; B), (C &amp; D) with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_sr_move_xy</td>
<td></td>
</tr>
<tr>
<td>_8168_start_sr_move_zu</td>
<td></td>
</tr>
<tr>
<td>_8168_start_sr_move_ab</td>
<td></td>
</tr>
<tr>
<td>_8168_start_sr_move_cd</td>
<td>Begin a relative 2-axis linear interpolation for X &amp; Y (Z &amp; U), (A &amp; B), (C &amp; D), with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_sa_move_xy</td>
<td></td>
</tr>
<tr>
<td>_8168_start_sa_move_zu</td>
<td></td>
</tr>
<tr>
<td>_8168_start_sa_move_ab</td>
<td></td>
</tr>
<tr>
<td>Function Name</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_8168_start_sa_move_cd</td>
<td>Begin a absolute 2-axis linear interpolation for X &amp; Y (Z &amp;U), (A &amp; B), (C &amp; D) with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_tr_line2</td>
<td>Begin a relative 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_tr_line2s</td>
<td>Begin a relative 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_sr_line2</td>
<td>Begin a relative 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_sr_line2s</td>
<td>Begin a relative 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_ta_line2</td>
<td>Begin a absolute 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_ta_line2s</td>
<td>Begin a absolute 2-axis linear interpolation for any 2 axes among axis 0 to 3 (s, for axis 4 to 7), with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_tr_line3</td>
<td>Begin a relative 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_tr_line3s</td>
<td>Begin a relative 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_sr_line3</td>
<td>Begin a relative 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_sr_line3s</td>
<td>Begin a absolute 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_ta_line3</td>
<td>Begin a absolute 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_ta_line3s</td>
<td>Begin a absolute 3-axis among axis 0 to 3 (s, for axis 4 to 7) linear interpolation with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_tr_line4</td>
<td>Begin a relative 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_tr_line4s</td>
<td>Begin a relative 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
</tbody>
</table>
### Circular Interpolation Motion: Section 6.8

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_start_sr_line4s</td>
<td>Begin a relative 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with S-curve profile</td>
</tr>
<tr>
<td>_8168_start_ta_line4</td>
<td>Begin a relative 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
<tr>
<td>_8168_start_sa_line4</td>
<td>Begin a absolute 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with S-curve profile,</td>
</tr>
<tr>
<td>_8168_start_ta_line4s</td>
<td>Begin a absolute 4-axis (axis 0 to 3; s, for axis 4 to 7) linear interpolation with trapezoidal profile</td>
</tr>
</tbody>
</table>

### Home Return Mode: Section 6.9

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_home_config</td>
<td>Set the home/index logic configuration</td>
</tr>
<tr>
<td>_8168_home_move</td>
<td>Begin a home return action</td>
</tr>
</tbody>
</table>
**Manual Pulser Motion: Section 6.10**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_pulser_iptmode</td>
<td>Set pulser input mode</td>
</tr>
<tr>
<td>_8168_pulser_vmove</td>
<td>Start pulser v move</td>
</tr>
<tr>
<td>_8168_pulser_pmove</td>
<td>Start pulser p move</td>
</tr>
<tr>
<td>_8168_pulser_home_move</td>
<td>Start pulser home move</td>
</tr>
<tr>
<td>_8168_pulser_r_line2</td>
<td>Start pulser line move</td>
</tr>
<tr>
<td>_8168_pulser_r_arc2</td>
<td>Start pulser arc move</td>
</tr>
<tr>
<td>_8168_set_pulser_ratio</td>
<td>Set pulser input ratio</td>
</tr>
</tbody>
</table>

**Motion Status: Section 6.11**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_motion_done</td>
<td>Return the motion status</td>
</tr>
</tbody>
</table>

**Motion Interface I/O: Section 6.12**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set alm</td>
<td>Set alarm logic and operating mode</td>
</tr>
<tr>
<td>_8168_set inp</td>
<td>Set INP logic and operating mode</td>
</tr>
<tr>
<td>_8168_set erc</td>
<td>Set ERC logic and timing</td>
</tr>
<tr>
<td>_8168_set servo</td>
<td>Set state of general purpose output pin</td>
</tr>
<tr>
<td>_8168_set sd</td>
<td>Set SD logic and operating mode</td>
</tr>
<tr>
<td>_8168_set el</td>
<td>Set EL logic and operating mode</td>
</tr>
</tbody>
</table>

**Motion I/O Monitoring: Section 6.13**

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_get io status</td>
<td>Get all the motion I/O status of cPCI-8168</td>
</tr>
</tbody>
</table>
## Interrupt Control: Section 6.14

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_int_control</td>
<td>Enable/Disable INT service</td>
</tr>
<tr>
<td>_8168_int_enable</td>
<td>Enable event (Windows only)</td>
</tr>
<tr>
<td>_8168_int_disable</td>
<td>Disable event (Windows only)</td>
</tr>
<tr>
<td>_8168_int_enableA</td>
<td>Enable bit mapped interrupt event (Windows only)</td>
</tr>
<tr>
<td>_8168_int_disableA</td>
<td>Disable bit mapped interrupt event (Windows only)</td>
</tr>
<tr>
<td>_8168_map_axis_event</td>
<td>Mapping bit mapped interrupt event (Windows only)</td>
</tr>
<tr>
<td>_8168_get_int_status</td>
<td>Get INT Status (Windows only)</td>
</tr>
<tr>
<td>_8168_link_interrupt</td>
<td>Set link to interrupt call back function (Windows only)</td>
</tr>
<tr>
<td>_8168_set_int_factor</td>
<td>Set INT factor</td>
</tr>
<tr>
<td>_8168_get_int_type</td>
<td>Get INT type (DOS only)</td>
</tr>
<tr>
<td>_8168_enter_isr</td>
<td>Enter interrupt service routine (DOS only)</td>
</tr>
<tr>
<td>_8168_leave_isr</td>
<td>Leave interrupt service routine (DOS only)</td>
</tr>
<tr>
<td>_8168_get_event_int</td>
<td>Get event status (DOS only)</td>
</tr>
<tr>
<td>_8168_get_error_int</td>
<td>Get error status (DOS only)</td>
</tr>
<tr>
<td>_8168_get_irq_status</td>
<td>Get IRQ status (DOS only)</td>
</tr>
<tr>
<td>_8168_not_my_irq</td>
<td>Not My IRQ (DOS only)</td>
</tr>
<tr>
<td>_8168_isr0~9, a, b</td>
<td>Interrupt service routine (DOS only)</td>
</tr>
</tbody>
</table>
Position Control and Counters: Section 6.15

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_get_position</td>
<td>Get the value of feedback position counter</td>
</tr>
<tr>
<td>_8168_set_position</td>
<td>Set the feedback position counter</td>
</tr>
<tr>
<td>_8168_get_command</td>
<td>Get the value of command position counter</td>
</tr>
<tr>
<td>_8168_set_command</td>
<td>Set the command position counter</td>
</tr>
<tr>
<td>_8168_get_error_counter</td>
<td>Get the value of position error counter</td>
</tr>
<tr>
<td>_8168_reset_error_counter</td>
<td>Reset the position error counter</td>
</tr>
<tr>
<td>_8168_get_general_counter</td>
<td>Get the value of general counter</td>
</tr>
<tr>
<td>_8168_set_general_counter</td>
<td>Set the general counter</td>
</tr>
<tr>
<td>_8168_get_target_pos</td>
<td>Get the value of target position recorder</td>
</tr>
<tr>
<td>_8168_reset_target_pos</td>
<td>Reset target position recorder</td>
</tr>
<tr>
<td>_8168_get_rest_command</td>
<td>Get remain pulse till end of motion</td>
</tr>
<tr>
<td>_8168_check_rdp</td>
<td>Check the ramping down point data</td>
</tr>
</tbody>
</table>
### Position Compare and Latch: Section 6.16

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_ltc_logic</td>
<td>Set the LTC logic</td>
</tr>
<tr>
<td>_8168_get_latch_data</td>
<td>Get latched counter data</td>
</tr>
<tr>
<td>_8168_set_soft_limit</td>
<td>Set soft limit</td>
</tr>
<tr>
<td>_8168_enable_soft_limit</td>
<td>Enable soft limit function</td>
</tr>
<tr>
<td>_8168_disable_soft_limit</td>
<td>Disable soft limit function</td>
</tr>
<tr>
<td>_8168_set_error_counter_check</td>
<td>Step-losing detection</td>
</tr>
<tr>
<td>_8168_set_general_comparator</td>
<td>Set general-purposed comparator</td>
</tr>
<tr>
<td>_8168_set_trigger_comparator</td>
<td>Set Trigger comparator</td>
</tr>
<tr>
<td>_8168_check_compare_data</td>
<td>Check current comparator data</td>
</tr>
<tr>
<td>_8168_check_compare_status</td>
<td>Check current comparator status</td>
</tr>
<tr>
<td>_8168_set_auto_compare</td>
<td>Set comparing data source for auto loading</td>
</tr>
<tr>
<td>_8168_build_compare_function</td>
<td>Build compare data via constant interval</td>
</tr>
<tr>
<td>_8168_build_compare_table</td>
<td>Build compare data via compare table</td>
</tr>
<tr>
<td>_8168_cmp_v_change</td>
<td>Speed change by comparator</td>
</tr>
</tbody>
</table>

### Continuous motion: Section 6.17

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_continuous_move</td>
<td>Enable continuous motion for absolute motion</td>
</tr>
<tr>
<td>_8168_check_continuous_buffer</td>
<td>Check if the buffer is empty</td>
</tr>
</tbody>
</table>
### Multiple Axes Simultaneous Operation: Section 6.18

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_tr_move_all</td>
<td>Multi-axis simultaneous operation setup</td>
</tr>
<tr>
<td>_8168_set_sr_move_all</td>
<td></td>
</tr>
<tr>
<td>_8168_set_ta_move_all</td>
<td></td>
</tr>
<tr>
<td>_8168_set_sa_move_all</td>
<td></td>
</tr>
<tr>
<td>_8168_start_move_all</td>
<td>Start motion simultaneously for all configured axis</td>
</tr>
<tr>
<td>_8168_stop_move_all</td>
<td>Stop motion simultaneously for all configured axis</td>
</tr>
</tbody>
</table>

### HSL Initialization: Section 6.19

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_HSL_Start</td>
<td>Start HSLink communication</td>
</tr>
<tr>
<td>W_HSL_Auto_Start</td>
<td>Auto start HSLink communication</td>
</tr>
<tr>
<td>W_HSL_Stop</td>
<td>Stop HSLink communication</td>
</tr>
</tbody>
</table>

### HSL Connection Information: Section 6.20

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_HSL_Connect_Status</td>
<td>Get slave module's status</td>
</tr>
<tr>
<td>W_HSL_Slave_Live</td>
<td>Check if the slave alive</td>
</tr>
</tbody>
</table>

### DIO Read/Write: Section 6.21

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>W_HSL_DIO_In</td>
<td>Digital input from all bits in one slave module</td>
</tr>
<tr>
<td>W_HSL_DIO_Channel_In</td>
<td>Digital input from one bit</td>
</tr>
<tr>
<td>W_HSL_DIO_Out</td>
<td>Digital output for all bits in one slave module</td>
</tr>
<tr>
<td>W_HSL_DIO_Channel_Out</td>
<td>Digital output for one bit</td>
</tr>
<tr>
<td>W_HSL_DIO_Memory_Out</td>
<td>Set output values of all slave modules</td>
</tr>
<tr>
<td>W_HSL_DIO_Memory_In</td>
<td>Get input values from all slave modules</td>
</tr>
</tbody>
</table>
## General-purposed DIO: Section 6.22

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_d_output</td>
<td>Digital output</td>
</tr>
<tr>
<td>_8168_d_input</td>
<td>Digital input</td>
</tr>
<tr>
<td>_8168_hd_output</td>
<td>High speed digital output</td>
</tr>
<tr>
<td>_8168_hd_input</td>
<td>High speed digital input</td>
</tr>
</tbody>
</table>

## General-purposed AIO: Section 6.23

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_read_ad</td>
<td>Read AD value</td>
</tr>
<tr>
<td>_8168_write_da</td>
<td>Write DA value</td>
</tr>
</tbody>
</table>

## High Resolution Timer: Section 6.24

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_h_timer</td>
<td>Set high resolution timer</td>
</tr>
<tr>
<td>_8168_h_timer</td>
<td>Get high resolution timer counter</td>
</tr>
<tr>
<td>_8168_ssleep</td>
<td>Pause for a specific time in program by timer</td>
</tr>
<tr>
<td>_8168_delay_time_mt</td>
<td>Pause for a specific time in program by motion axis</td>
</tr>
<tr>
<td>_8168_delay_time</td>
<td>Pause for a specific time in program by motion axis</td>
</tr>
</tbody>
</table>

## DI Change of State: Section 6.25

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_set_d_cos</td>
<td>Set DI change of state status</td>
</tr>
<tr>
<td>_8168_wait_d_cos</td>
<td>Wait DI change of status status</td>
</tr>
<tr>
<td>_8168_set_hd_cos</td>
<td>Set high speed DI change of state status</td>
</tr>
<tr>
<td>_8168_wait_hd_cos</td>
<td>Wait high speed DI change of state status</td>
</tr>
<tr>
<td>_8168_get_cos_status</td>
<td>Get all DI change of state status</td>
</tr>
</tbody>
</table>
### 6.2 C/C++ Programming Library

The function prototypes and some common data types are decelerated in Pci_8168.H. These data types are used by the cPCI-8168 library. The following table shows the data type names and their range. It is suggested that these data types are used in your application programs.

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

The functions of cPCI-8168's software drivers use full-names to represent the functions' real meaning. The naming convention rules are:

In ‘C’ programming Environment:

_{(hardware_model)}_{(action_name)}. e.g. _8168_Initial().

In order to recognize the difference between C library and VB library, a capital “B” is put on the head of each function name e.g. B_8168_Initial().
6.3 Initialization

@ Name

_8168_initial – Software Initialization for cPCI-8168
_8168_close – Software release resources of cPCI-8168
_8168_get_base_addr – Get the base address of cPCI-8168
_8168_get_irq_channel – Get the cPCI-8168 card’s IRQ number
_8168_config_from_file – Configure cPCI-8168 card according to configuration file ie. 8168.ini.
_8168_get_fpga_version – Get the FPGA version number

@ Description

_8168_initial:
This function is used to initialize cPCI-8168 cards. All cPCI-8168 cards can be initialized by calling this function once before using other functions.

_8168_close:
This function is used to close cPCI-8168 cards and release the cPCI-8168 related resources, which should be called at the end of an application.

_8168_get_irq_channel:
This function is used to get the cPCI-8168 card’s IRQ number.

_8168_get_base_addr:
This function is used to get the cPCI-8168 card’s base address.

_8168_config_from_file:
This function is used to load the configuration of cPCI-8168 according to the specified file. By using MotionCreator, users can test and configure cPCI-8168 correctly. After pressing the “save config” button, the 8168.ini file in the window directory is used to record the configurations. By specifying it in the parameter, the configuration will automatically load.
When this function is executed, all cPCI-8168 cards in the system will be configured as the following functions are called according to the recorded parameters in 8168.ini.

- _8168_set_pls_outmode
- _8168_set_feedback_src
- _8168_set_pls_iptmode
- _8168_set_home_config
- _8168_set_int_factor
- _8168_set_el
- _8168_set_ltc_logic
- _8168_set_erc
- _8168_set_sd
- _8168_set_alm
- _8168_set_inp
- _8168_set_move_ratio

(!_8168_get_fpga_version_):
This function is for getting the FPGA firmware version number. The version history is:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>2001/10/30</td>
<td>First Release</td>
</tr>
<tr>
<td>0.2</td>
<td>2001/12/15</td>
<td>Second Release</td>
</tr>
<tr>
<td>0.3</td>
<td>2002/02/20</td>
<td>Third release</td>
</tr>
<tr>
<td>0.5</td>
<td>2007/08/10</td>
<td>H/W COS read clear and read Firmware</td>
</tr>
</tbody>
</table>

@ Syntax

_C/C++ (DOS, Windows)_

I16 _8168_initial(I16 *existCards);
I16 _8168_close(void);
I16 _8168_get_irq_channel(I16 cardNo, U16 *irq_no);
I16 _8168_get_base_addr(I16 cardNo, U16 *base_addr);
I16 _8168_config_from_file(char *filename);
Visual Basic (Windows)

B_8168_initial (existCards As Integer) As Integer
B_8168_close () As Integer
B_8168_get_irq_channel (ByVal CardNo As Integer, 
                   irq_no As Integer) As Integer
B_8168_get_base_addr (ByVal CardNo As Integer, 
                   base_addr As Integer) As Integer
B_8168_config_from_file(ByVal filename As 
                       string)as integer

@ Argument
*existCards*: numbers of existing cPCI-8168 cards
*cardNo*: The cPCI-8168 card index number.
*irq_no*: Irq number of specified cPCI-8168 card.
*base_addr*: base address of specified cPCI-8168 card
*Filename*: The specified filename recording the configuration of 
cPCI-8168. This file must be created by **MotionCreator** of 
cPCI-8168.

@ Return Code
ERR_NoError
ERR_NoCardFound
ERR_PCIBiosNotExist
ERR_ConfigFileOpenError
6.4 Pulse Input/Output Configuration

@ Name

_function_name_ – Set the configuration for pulse command output.

_function_name_ – Set the configuration for feedback pulse input.

_function_name_ – Enable/Disable the external feedback pulse input

@ Description

_function_name_: Configure the output modes of command pulse. There are more than 6 modes for command pulse output.

_function_name_: Configure the input modes of external feedback pulse. There are four types for feedback pulse input. Note that this function makes sense only when src parameter in _function_name_ function is set to 0 as external source.

_function_name_: If external encoder feedback is available in the system, set the src parameter in this function to the enabled state. Then internal 28-bit up/down counter will count according to the configuration of _function_name_() function, or the counter will count the command pulse output.

@ Syntax


C/C++ (DOS, Windows)

```c
I16 _8168_set_pls_outmode(I16 AxisNo, I16 pls_outmode);
I16 _8168_set_pls_iptmode(I16 AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _8168_set_feedback_src(I16 AxisNo, I16 Src);
```
Visual Basic (Windows)

B_8168_set_pls_outmode (ByVal AxisNo As Integer, ByVal pls_outmode As Integer) As Integer
B_8168_set_pls_iptmode (ByVal AxisNo As Integer, ByVal pls_iptmode As Integer, ByVal pls_logic As Integer) As Integer
B_8168_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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>OUT/DIR  OUT Falling edge, DIR+ is high level</td>
</tr>
<tr>
<td>1</td>
<td>OUT/DIR  OUT Rising edge, DIR+ is high level</td>
</tr>
<tr>
<td>2</td>
<td>OUT/DIR  OUT Falling edge, DIR+ is low level</td>
</tr>
<tr>
<td>3</td>
<td>OUT/DIR  OUT Rising edge, DIR+ is high level</td>
</tr>
<tr>
<td>4</td>
<td>CW/CCW   Falling edge</td>
</tr>
<tr>
<td>5</td>
<td>CW/CCW   Rising edge</td>
</tr>
<tr>
<td>6</td>
<td>AB phase A leading B (Only for PCL6045B)</td>
</tr>
<tr>
<td>7</td>
<td>AB phase B leading A (Only for PCL6045B)</td>
</tr>
</tbody>
</table>

pls_iptmode: setting of encoder feedback pulse input mode

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1X A/B</td>
</tr>
<tr>
<td>1</td>
<td>2X A/B</td>
</tr>
<tr>
<td>2</td>
<td>4X A/B</td>
</tr>
<tr>
<td>3</td>
<td>CW/CCW</td>
</tr>
</tbody>
</table>

pls_logic: Logic of encoder feedback pulse

pls_logic=0, Non-inverse direction
pls_logic=1, Inverse direction
**Src:** Counter source

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>External Feedback</td>
</tr>
<tr>
<td>1</td>
<td>Command pulse</td>
</tr>
</tbody>
</table>

**@ Return Code**

ERR_NoError
6.5 Velocity mode motion

@ Name

_8168_tv_move – Accelerate an axis to a constant velocity with trapezoidal profile

_8168_sv_move – Accelerate an axis to a constant velocity with S-curve profile

_8168_v_change – Change speed on the fly

_8168_sd_stop – Decelerate to stop

_8168_emg_stop – Immediately stop

_8168_fix_speed_range – Define the speed range

_8168_unfix_speed_range – Release the speed range constraint

_8168_get_current_speed – Get current speed

@ Description

_8168_tv_move:
This function is to accelerate an axis to the specified constant velocity with 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 velocity parameter.

_8168_sv_move:
This function is to accelerate an axis to the specified constant velocity with 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.

_8168_v_change:
This function changes the moving velocity according to the trapezoidal profile or S-curve profile. Before calling this function, it is necessary to define the speed range by _8168_fix_speed_range. The velocity change is only valid inside the range.
_8168_v_change is also applicable on positioning motion. Note: The velocity profile is decided by the original motion profile. When using in S-curve, set the motion to be pure S-curve. There are some limitations for this function under position motion, please refer to section 4.6.1 before using it.

_8168_sd_stop:
This function is used to decelerate an axis to a stop according to a trapezoidal profile or S-curve profile. This function is also useful when velocity move, positioning move (both trapezoidal and S-curve motion), manual pulser or home return move function is performed. Note: The velocity profile is decided by original motion profile.

_8168_emg_stop:
This function is used to immediately stop an axis for all motion moves.

_8168_fix_speed_range
This function is used to define the speed range. It should be called before starting motion that may contain velocity changing or increasing the maximum acceleration rate.

_8168_unfix_speed_range
This function is used to Release the speed range constraint.

_8168_get_current_speed
This function is used to read current pulse output rate of a specified axis. It is applicable at any time and during any operating mode.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_tv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8168_sv_move(I16 AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc, F64 SVacc);
I16 _8168_v_change(I16 AxisNo, F64 NewVel, F64 Tacc);
I16 _8168_sd_stop(I16 AxisNo,F64 Tdec);
I16 _8168_emg_stop(I16 AxisNo);
F64 _8168_fix_speed_range(I16 AxisNo, F64 MaxVel);
I16 _8168_unfix_speed_range(I16 AxisNo);
I16 _8168_get_current_speed(I16 AxisNo, F64 *speed);

Visual Basic (Windows 95/NT)

B_8168_tv_move (ByVal AxisNo As Integer, ByVal StrVel As Double, ByVal MaxVel As Double, ByVal Tacc As Double) As Integer
B_8168_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_8168_v_change (ByVal AxisNo As Integer, ByVal NewVel As Double, ByVal TimeSecond As Double) As Integer
B_8168_sd_stop (ByVal AxisNo As Integer, ByVal Tdec As Double) As Integer
B_8168_emg_stop (ByVal AxisNo As Integer) As Integer
B_8168_fix_speed_range (ByVal AxisNo As Integer, ByVal MaxVel As Double) As Integer
B_8168_unfix_speed_range (ByVal AxisNo As Integer) As Integer
B_8168_get_current_speed (ByVal AxisNo As Integer, Speed As Double) As Integer

@ Argument

AxisNo: axis number designated to move or stop.
StrVel: starting velocity in unit of pulse per second
MaxVel: maximum velocity in unit of pulse per second
Tacc: specified acceleration time in unit of second
SVacc: specified velocity interval in which S-curve acceleration is performed.

Note: SVacc = 0, for pure S-Curve
NewVel: New velocity in unit of pulse per second
**Tdec**: specified deceleration time in unit of second

*Speed*: Variable to save current speed.

(speed range: 0~6553500)

**@ Return Code**

- ERR_NoError
- ERR_SpeedError
- ERR_SpeedChangeError
- ERR_SlowDownPointError
- ERR_AxisAlreadyStop
6.6 Single Axis Position Mode

@ Name

_8168_start_tr_move – Begin a relative trapezoidal profile move
_8168_start_ta_move – Begin an absolute trapezoidal profile move
_8168_start_sr_move – Begin a relative S-curve profile move
_8168_start_sa_move – Begin an absolute S-curve profile move
_8168_set_move_ratio – Set the ratio of command pulse and feedback pulse.
_8168_p_change – Change position on the fly
_8168_set_pcs_logic – Set the logic of PCS (Position Change Signal) pin
_8168_set_sd_pin – Set SD/PCS pin for first 4 axes of cPCI-8168
_8168_backlash_comp – Set backlash compensating pulse for compensation
_8168_suppress_vibration – Set vibration suppressing timing

The moving direction is determined by the sign of 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, VSdec, will also become shorter while the dV/dt (acceleration / deceleration) and d(dV/dt)/dt (jerk) keep unchanged.

@ Description

_8168_start_tr_move:
This function causes the axis to accelerate from a starting velocity, slow at constant velocity, and decelerate to a stop at the relative distance with trapezoidal profile. The acceleration and deceleration time is specified independently. It won’t let the program wait
for motion completion but immediately return control to the program.

**_8168_start_ta_move_**
This function causes the axis to accelerate from a starting velocity, slow at a constant velocity, and decelerate to a stop at the specified absolute position with trapezoidal profile. The acceleration and deceleration time is specified independently. It won’t let the program wait for motion completion but immediately return control to the program.

**_8168_start_sr_move_**
This function causes the axis to accelerate from a starting velocity, slow at a constant velocity, and decelerate to a stop at the relative distance with S-curve profile. The acceleration and deceleration time is specified independently. It won’t let the program wait for motion completion but immediately return control to the program.

**_8168_start_sa_move_**
This function causes the axis to accelerate from a starting velocity, slow at a constant velocity, and decelerate to a stop at the specified absolute position with S-curve profile. The acceleration and deceleration time is specified independently. It won’t let the program wait for motion completion but immediately return control to the program.

**_8168_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 the resolution of command pulse, then ratio = 2.

**_8168_p_change_**
This function is used to change target position on the fly. There are some limitations for this function. Please refer to section 4.6.2 before using it.

**_8168_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 _8168_set_sd_pin, this _8168_set_pcs_logic function becomes effective.

_8168_set_sd_pin:
This function is used to set the operating mode of the SD pin. The SD pin may be used either as Slow-Down signal input or as Position Change Signal (PCS) input. This function is only valid for the first 4 axes of each cPCI-8168 board. Please refer to section 4.3.1

_8168_backlash_comp:
Whenever direction change is occurred, The cPCI-8168 outputs backlash corrective pulses before sending commands. This function is used to set the compensation pulse numbers.

_8168Suppress_vibration
This function is used to suppress the vibration of the mechanical system by outputting a single pulse in the negative direction and then a single pulse in the positive direction right after the completion of a command movement.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_start_tr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_ta_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_sr_move(I16 AxisNo, F64 Dist, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8168_start_sa_move(I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8168_set_move_ratio(I16 AxisNo, F64 move_ratio);
I16 _8168_p_change(I16 AxisNo, F64 NewPos);
I16 _8168_set_pcs_logic(I16 AxisNo, I16 pcs_logic);
I16 _8168_set_sd_pin(I16 AxisNo, I16 Type);
I16 _8168_backlash_comp(I16 AxisNo, I16 BCompPulse);
I16 _8168_suppress_vibration(I16 AxisNo, U16 T1, U16 T2);

Visual Basic (Windows)

B_8168_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_8168_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_8168_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_8168_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_8168_set_move_ratio (ByVal AxisNo As Integer, ByVal move_ratio As Double) As Integer
B_8168_p_change (ByVal AxisNo As Integer, ByVal NewPos As Double) As Integer
B_8168_set_pcs_logic (ByVal AxisNo As Integer, ByVal pcs_logic As Integer) As Integer
B_8168_set_sd_pin (ByVal AxisNo As Integer, ByVal Type As Integer) As Integer
B_8168_backlash_comp (ByVal AxisNo As Integer, ByVal BCompPulse As Integer, ByVal ForwardTime As Integer) As Integer
B_8168_suppress_vibration (ByVal AxisNo As Integer, ByVal ReserveTime As Integer, ByVal ForwardTime As Integer) As Integer

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 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
Move_ratio: ratio of (feedback resolution)/(command resolution), should not be 0
NewPos: specified new absolute position to move
Type: When type=0, the PCS is enabled and type=1, the SD is enabled.
BcompPulse: Specified number of corrective pulses
T1: Specified Reverse Time
T2: Specified Forward Time
@ Return Code

    ERR_NoError
    ERR_SpeedError
    ERR_PChangeSlowDownPointError
    ERR_MoveRatioError
6.7 Linear Interpolated Motion

@ Name and description

_8168_start_tr_move_(MN) – Begin a relative 2-axis linear interpolation for M & N axis, with trapezoidal profile, M&N could be XY,ZU, AB and CD.

_8168_start_ta_move_(MN) – Begin a absolute 2-axis linear interpolation for M & N, with trapezoidal profile, M&N could be XY,ZU, AB and CD.

_8168_start_sr_move_(MN) – Begin a relative 2-axis linear interpolation for M & N, with S-curve profile, M&N could be XY,ZU, AB and CD.

_8168_start_sa_move_(MN) – Begin a absolute 2-axis linear interpolation for M & N, with S-curve profile, M&N could be XY,ZU, AB and CD.

_8168_start_tr_line2(s) – Begin a relative 2-axis linear interpolation for any 2 axes from one motion chip with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_sr_line2(s) – Begin a relative 2-axis linear interpolation for any 2 axes from one motion chip with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_ta_line2(s) – Begin an absolute 2-axis linear interpolation for any 2 axes with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_sa_line2(s) – Begin an absolute 2-axis linear interpolation for any 2 axes from one motion chip with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_tr_line3(s) – Begin a relative 3-axis linear interpolation for any 3 axes from one motion chip with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_sr_line3(s) – Begin a relative 3-axis linear interpolation for any 3 axes from one motion chip with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.
_8168_start_ta_line3(s) – Begin an absolute 3-axis linear interpolation for any 3 axes from one motion chip with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_startSa_line3(s) – Begin an absolute 3-axis linear interpolation for any 3 axes from one motion chip with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_tr_line4(s) – Begin a relative 4-axis linear interpolation for all 4 axes from one motion chip with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_sr_line4(s) – Begin a relative 4-axis linear interpolation for all 4 axes from one motion chip with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_ta_line4(s) – Begin an absolute 4-axis linear interpolation for all 4 axes from one motion chip with trapezoidal profile. The (s) means secondary motion chip in one cPCI-8168.

_8168_start_sa_line4(s) – Begin a absolute 4-axis linear interpolation with S-curve profile. The (s) means secondary motion chip in one cPCI-8168.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_start_tr_move_(MN)(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_ta_move_(MN)(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_sr_move_(MN)(I16 CardNo, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8168_start_sa_move_(MN)(I16 CardNo, F64 PosX, F64 PosY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8168_start_tr_line2(s)(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_ta_line2(s)(I16 CardNo, I16
  *AxisArray, F64 PosX, F64 PosY, F64 StrVel,
  F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_sr_line2(s)(I16 CardNo, I16
  *AxisArray, F64 DistX, F64 DistY, F64
  StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
  SVacc, F64 SVdec);
I16 _8168_start_sa_line2(s)(I16 CardNo, I16
  *AxisArray, F64 PosX, F64 PosY, F64 StrVel,
  F64 MaxVel, F64 Tacc, F64 Tdec, F64 SVacc,
  F64 SVdec);
I16 _8168_start_tr_line3(s)(I16 CardNo, I16
  *AxisArray, F64 DistX, F64 DistY, F64 DistZ,
  F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_ta_line3(s)(I16 CardNo, I16
  *AxisArray, F64 PosX, F64 PosY, F64 PosZ,
  F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_sr_line3(s)(I16 CardNo, I16
  *AxisArray, F64 DistX, F64 DistY, F64 DistZ,
  F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec,
  F64 SVacc, F64 SVdec);
I16 _8168_start_sa_line3(s)(I16 CardNo, I16
  *AxisArray, F64 PosX, F64 PosY, F64 PosZ,
  F64 StrVel, F64 MaxVel, F64 Tacc, F64 Tdec,
  F64 SVacc, F64 SVdec);
I16 _8168_start_tr_line4(s)(I16 CardNo, F64
  DistX, F64 DistY, F64 DistZ, F64 DistU, F64
  StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_ta_line4(s)(I16 CardNo, F64
  PosX, F64 PosY, F64 PosZ, F64 PosU, F64
  StrVel, F64 MaxVel, F64 Tacc, F64 Tdec);
I16 _8168_start_sr_line4(s)(I16 CardNo, F64
  DistX, F64 DistY, F64 DistZ, F64 DistU, F64
  StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
  SVacc, F64 SVdec);
I16 _8168_start_sa_line4(s)(I16 CardNo, F64
  PosX, F64 PosY, F64 PosZ, F64 PosU, F64
  StrVel, F64 MaxVel, F64 Tacc, F64 Tdec, F64
  SVacc, F64 SVdec);

Visual Basic (Windows)

B_8168_start_tr_move_(MN) (ByVal CardNo As
  Integer, ByVal Dist As Double, ByVal Dist As
Function Library

B_8168_start_ta_move_(MN) (ByVal CardNo 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_8168_start_sr_move_(MN) (ByVal CardNo 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_8168_start_sa_move_(MN) (ByVal CardNo 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_8168_start_tr_line2(s) (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_8168_start_ta_line2(s) (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_8168_start_sr_line2(s) (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_8168_start_sa_line2(s) (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
Function Library

B_8168_start_tr_line3(s) (ByVal CardNo 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_8168_start_ta_line3(s) (ByVal CardNo As Integer, ByVal PosX As Double, ByValPosY 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_8168_start_sr_line3(s) (ByVal CardNo 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_8168_start_sa_line3(s) (ByVal CardNo As Integer, ByVal PosX As Double, ByValPosY 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_8168_start_tr_line4(s) (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_8168_start_ta_line4(s) (ByVal CardNo As Integer, ByVal PosX As Double, ByValPosY 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_8168_start_sr_line4(s) (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
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_8168_start_sa_line4(s) (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

@ Argument

**CardNo**: Card number designated to perform linear interpolation

**Dist#**: specified relative distance of axis to move

**Pos#**: specified absolute position of axis to move

**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:

```csharp
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.
Note: If the function is for secondary axes (Axis 4~7), the AxisArray must have the axes indexing from the range of 4~7.

@ Return Code

ERR_NoError
ERR_SpeedError
ERR_AxisArrayError
6.8 Circular Interpolation Motion

@ Name

_8168_start_r_arc_xy – Begin a relative circular interpolation for X & Y on primary motion chip

_8168_start_a_arc_xy – Begin a absolute circular interpolation for X & Y on primary motion chip

_8168_start_r_arc_zu – Begin a relative circular interpolation for Z & U on primary motion chip

_8168_start_a_arc_zu – Begin a absolute circular interpolation for Z & U on primary motion chip

_8168_start_r_arc_ab – Begin a relative circular interpolation for A & B on secondary motion chip

_8168_start_a_arc_ab – Begin a absolute circular interpolation for A & B on secondary motion chip

_8168_start_r_arc_cd – Begin a relative circular interpolation for C & D on secondary motion chip

_8168_start_a_arc_cd – Begin a absolute circular interpolation for C & D on secondary motion chip

_8168_start_r_arc2 – Begin a relative circular interpolation for any 2 axes on primary motion chip

_8168_start_a_arc2 – Begin a absolute circular interpolation for any 2 axes on primary motion chip

_8168_start_r_arc2s – Begin a relative circular interpolation for any 2 axes on secondary motion chip

_8168_start_a_arc2s – Begin a absolute circular interpolation for any 2 axes on secondary motion chip
@ Description

<table>
<thead>
<tr>
<th>Functions</th>
<th>Relative / Absolute</th>
<th>Target Axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>_8168_start_r_arc_xy</td>
<td>R</td>
<td>Axis 0 &amp; 1</td>
</tr>
<tr>
<td>_8168_start_a_arc_xy</td>
<td>A</td>
<td>Axis 0 &amp; 1</td>
</tr>
<tr>
<td>_8168_start_r_arc_zu</td>
<td>R</td>
<td>Axis 2 &amp; 3</td>
</tr>
<tr>
<td>_8168_start_a_arc_zu</td>
<td>A</td>
<td>Axis 2 &amp; 3</td>
</tr>
<tr>
<td>_8168_start_r_arc_ab</td>
<td>R</td>
<td>Axis 4 &amp; 5</td>
</tr>
<tr>
<td>_8168_start_a_arc_ab</td>
<td>A</td>
<td>Axis 4 &amp; 5</td>
</tr>
<tr>
<td>_8168_start_r_arc_cd</td>
<td>R</td>
<td>Axis 6 &amp; 7</td>
</tr>
<tr>
<td>_8168_start_a_arc_cd</td>
<td>A</td>
<td>Axis 6 &amp; 7</td>
</tr>
<tr>
<td>_8168_start_r_arc2</td>
<td>R</td>
<td>Any 2 of 4 from Axis0~3</td>
</tr>
<tr>
<td>_8168_start_a_arc2</td>
<td>A</td>
<td>Any 2 of 4 from Axis0~3</td>
</tr>
<tr>
<td>_8168_start_r_arc2s</td>
<td>R</td>
<td>Any 2 of 4 from Axis4~7</td>
</tr>
<tr>
<td>_8168_start_a_arc2s</td>
<td>A</td>
<td>Any 2 of 4 from Axis4~7</td>
</tr>
</tbody>
</table>

@ Syntax

C/C++ (DOS, Windows)

```c
I16 _8168_start_r_arc_xy(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc_xy(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8168_start_r_arc_zu(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc_zu(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8168_start_r_arc_ab(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc_ab(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8168_start_r_arc_cd(I16 CardNo, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc_cd(I16 CardNo, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
```
I16 _8168_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);
I16 _8168_start_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);
I16 _8168_start_a_arc2(I16 CardNo, I16 *AxisArray, F64 Cx, F64 Cy, F64 Ex, F64 Ey, I16 DIR, F64 MaxVel);

Visual Basic (Windows)
B_8168_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_8168_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_8168_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_8168_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_8168_start_a_arc2s (ByVal CardNo As Integer, ByVal 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_8168_start_r_arc2s (ByVal CardNo As Integer, ByVal 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
Double, ByVal OffsetCy As Double, ByVal OffsetEx As Double, ByVal OffsetEy As Double, ByVal DIR As Integer, ByVal MaxVel 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 of the end of arc
Ey: specified Y-axis absolute position of the end of arc
DIR: Specified direction of arc, CW:0 , CCW:1
MaxVel: Tangential velocity in unit of pulse per second
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]
Note: If the function is for secondary axes (Axis 4~7), the AxisArray must have the axes indexing from the range of 4~7

@ Return Code
ERR_NoError
ERR_SpeedError
ERR_AxisArrayError
6.9 Home Return Mode

@ Name

_8168_set_home_config – Set the configuration for home return.
_8168_home_move – Perform a home return move.
_8164_escape_home – Escape Home Function
_8164_home_search – Auto-Search Home Switch

@ Description

_8168_set_home_config:
Configure the home return mode, origin & index signal(EZ) logic, EZ count and ERC output options for home_move() function. Refer to Section 4.1.8 for the setting of home_mode control.

_8168_home_move:
This function will cause the axis to perform a home return move according to the setting of _8168_set_home_config() function. The direction of movement is determined by the sign of velocity parameter(svel, mvel). Since the stopping condition of this function is determined by home_mode setting, user should be sure to select the initial moving direction and handle the condition when the limit switch is touched or other possible conditions causing the axis to stop. Executing v_stop() function during home_move() can also cause the axis to stop.

_8168_escape_home:
After homing, use this function to leave the home switch

_8168_home_search:
Auto searching home position. The axis will reverse when limit switch is reached and start over the home move without being stopped by the limit switch. It is very useful when users don’t know the actual position of axis.
@ Syntax

C/C++ (DOS, Windows)

I16 _8168_set_home_config(I16 AxisNo, I16 home_mode, I16 org_logic, I16 ez_logic, I16 ez_count, I16 erc_out);
I16 _8168_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)

B_8168_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_8168_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 As Double) As Integer

@ Argument

AxisNo: axis number designated to configure and perform home returning

home_mode: stopping modes for home return, 0~12 (Please refer to section 4.1.8)

org_logic: Action logic configuration for ORG signal

org_logic=0, active low;
org_logic=1, active high

ez_logic: Action logic configuration for EZ signal
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 homing finish

**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

**ORGOffset**: The escape pulse amounts when home search touches the ORG singal. The unit is pulses.

**@ Return Code**

ERR_NoError
6.10 Manual Pulser Motion

@ Name

_8168_set_pulser_iptmode - set the input signal modes of pulser
_8168_pulser_vmove – manual pulser v_move
_8168_pulser_pmove – manual pulser p_move
_8168_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

_8168_set_pulser_iptmode:
This function is used to configure the input mode of manual pulser.

_8168_pulser_vmove:
As this command is written, the axis begins to move the axis according to the manual pulser input. The axis will output one pulse upon receiving a pulse from the pulser, until the sd_stop or emg_stop command is written.

_8168_pulser_pmove:
As this command is written, the axis begins to move the axis according to the manual pulser input. The axis will output one pulse upon receiving one pulse from the pulser, until the sd_stop or emg_stop command is written or the output pulse number reach dist.

_8168_pulser_home_move:
As this command is written, the axis begins to move the axis according to manual pulser input. The axis will output one pulse
upon receiving one pulse from the pulser, until the \texttt{sd\_stop} or \texttt{emg\_stop} command is written or the home move finish.

\texttt{\_8164\_set\_pulser\_ratio:}

Set manual pulse ratio for actual output pulse rate. The formula for manual pulse output rate is:

\[
\text{Output Pulse Count} = \text{Input Pulse Count} \times 4 \times (\text{MultiF}+1) \times (\text{DivF}+1)/2048
\]

The DivF=0~2047 Divide Factor
The MultiF=0~31 Multiplication Factor

\texttt{\_8168\_pulser\_r\_line2:}

Pulser mode for 2-axis linear interpolation (relative mode only).

\texttt{\_8168\_pulser\_r\_arc2:}

Pulser mode for 2-axis arc interpolation (relative mode only)

\texttt{\_8168\_pulser\_r\_line2s:}

Pulser mode for 2-axis linear interpolation of secondary motion chip (relative mode only).

\texttt{\_8168\_pulser\_r\_arc2s:}

Pulser mode for 2-axis arc interpolation of secondary motion chip(relative mode only)

\textbf{@ Syntax}

\textbf{C/C++ (DOS, Windows)}

\begin{verbatim}
I16 _8168_set_pulser_iptmode(I16 AxisNo, I16 InputMode, I16 Inverse);
I16 _8168_pulser_vmove(I16 AxisNo, F64 SpeedLimit);
I16 _8168_pulser_pmove(I16 AxisNo, F64 Dist, F64 SpeedLimit);
I16 _8168_pulser_home_move(I16 AxisNo, I16 HomeType, F64 SpeedLimit);
I16 _8168_set_pulser_ratio(I16 AxisNo, I16 DivF, I16 MultiF);
\end{verbatim}
I16 _8168_pulser_r_line2(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 SpeedLimit);

I16 _8168_pulser_r_arc2(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);

I16 _8168_pulser_r_line2s(I16 CardNo, I16 *AxisArray, F64 DistX, F64 DistY, F64 SpeedLimit);

I16 _8168_pulser_r_arc2s(I16 CardNo, I16 *AxisArray, F64 OffsetCx, F64 OffsetCy, F64 OffsetEx, F64 OffsetEy, I16 DIR, F64 MaxVel);

**Visual Basic (Windows 95/NT)**

B_8168_set_pulser_iptmode (ByVal AxisNo As Integer, ByVal InputMode As Integer, ByVal Inverse As Integer) As Integer

B_8168_pulser_vmove (ByVal AxisNo As Integer, ByVal SpeedLimit As Double) As Integer

B_8168_pulser_pmove (ByVal AxisNo As Integer, ByVal Dist As Double, ByVal SpeedLimit As Double) As Integer

B_8168_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 DivF As Integer, ByVal MultiF As Integer) As Integer

B_8164_pulser_r_line2(ByVal CardNo As Integer, ByVal 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, ByVal 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_pulser_r_line2s(ByVal CardNo As Integer, ByVal AxisArray As Integer, ByVal DistX As Double,
ByVal DistY As Double, ByVal SpeedLimit As Double) As Integer
B_8164_pulser_r_arc2s(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 pulser input mode from 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 pulser move.
  For example, if SpeedLimit is set to be 100 pps, then the axis can move at fastest 100 pps, even the input pulser signal rate is more then 100 pps.
Dist: specified relative distance to move (unit: pulses)
HomeType: specified home move type
  HomeType =0, Command Origin.(that means axis stops when command counter becomes ‘0’)
  HomeType =1, ORG pin.
DivF: Divide factor (0~2047)
MultiF: Multiplication factor (0~31)
DistX: specified relative distance of first axis to move
**DistY**: specified relative distance of second axis to move

**OffsetCx**: X-axis offset from center

**OffsetCy**: Y-axis offset from center

**OffsetEx**: X-axis offset from the end of arc

**OffsetEy**: Y-axis offset from the 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

**AxisArray**: Array of axis number to perform interpolation.

Example:

```c
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]**

Note: If the function is for secondary axes (Axis 4~7), the **AxisArray must have the axes indexing from the range of 4~7**

**@ Return Code**

- ERR_NoError
- ERR_PulserHomeTypeError
6.11 Motion Status

@ Name
_8168_motion_done – Return the motion status

@ Description
_8168_motion_done:
Return the motion status of cPCI-8168.

@ Syntax

C/C++ (DOS, Windows)
I16 _8168_motion_done(I16 AxisNo);

Visual Basic (Windows)
B_8168_motion_done (ByVal AxisNo As Integer) As Integer

@ Argument
AxisNo: axis number designated to start manual move

@ Return Value

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal stop condition</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>Waiting for CSTA input</td>
</tr>
<tr>
<td>3</td>
<td>Waiting for an internal synchronous signal</td>
</tr>
<tr>
<td>4</td>
<td>Waiting for the other axis to stop</td>
</tr>
<tr>
<td>5</td>
<td>Waiting for a completion of ERC timer</td>
</tr>
<tr>
<td>6</td>
<td>Waiting for a completion of direction change timer</td>
</tr>
<tr>
<td>7</td>
<td>Backlash compensating</td>
</tr>
<tr>
<td>8</td>
<td>Wait PA/PB</td>
</tr>
<tr>
<td>9</td>
<td>At FA speed</td>
</tr>
<tr>
<td>10</td>
<td>At FL speed ( start velocity )</td>
</tr>
<tr>
<td>11</td>
<td>In acceleration</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td>---</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>12</td>
<td>At FH speed (maximum velocity)</td>
</tr>
<tr>
<td>13</td>
<td>In deceleration</td>
</tr>
<tr>
<td>14</td>
<td>Waiting for INP signal</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
6.12 Motion Interface I/O

@ Name

_8168_set_alm – Set alarm logic and operating mode
_8168_set_el – Set EL logic and operating mode
_8168_set_inp – Set Inp logic and operating mode
_8168_set_erc – Set ERC logic and timing
_8168_set_servo – Set state of general purpose output pin
_8168_set_sd – Set SD logic and operating mode

@ Description

_8168_set_alm_logic:
Set the active logic of ALARM signal input from servo driver. Two reacting modes are available when ALARM signal is active.

_8168_set_el:
Set the reacting modes of EL signal.

_8168_set_inp_logic:
Set the active logic of In-Position signal input from servo driver. Users can select whether they want to enable this function. Default state is disabled.

_8168_set_erc:
You can set the logic and on time of ERC by this function.

_8168_set_servo:
You can set the ON-OFF state of SVON signal by this function. The default value is 1 (OFF), which means the SVON is open to GND.

_8168_set_sd_logic:
Set the active logic, latch control and operating mode of SD signal input from mechanical system. Users can select whether they want to enable this function. Default state is disabled.
@ Syntax

**C/C++ (DOS, Windows)**

```
I16 _8168_set alm(I16 AxisNo, I16 alm_logic, I16 alm_mode);
I16 _8168_set_el(I16 AxisNo, I16 el_mode);
I16 _8168_set_inp(I16 AxisNo, I16 inp_enable, I16 inp_logic);
I16 _8168_set_erc(I16 AxisNo, I16 erc_logic, I16 erc_on_time);
I16 _8168_set_servo(I16 AxisNo, I16 on_off);
I16 _8168_set_sd(I16 AxisNo, I16 enable, I16 sd_logic, I16 sd_latch, I16 sd_mode);
```

**Visual Basic (Windows)**

```
B_8168_set_alm (ByVal AxisNo As Integer, ByVal alm_logic As Integer, ByVal alm_mode As Integer) As Integer
B_8168_set_el (ByVal AxisNo As Integer, ByVal el_mode As Integer) As Integer
B_8168_set_inp (ByVal AxisNo As Integer, ByVal inp_enable As Integer, ByVal inp_logic As Integer) As Integer
B_8168_set_erc (ByVal AxisNo As Integer, ByVal erc_logic As Integer, ByVal erc_on_time As Integer) As Integer
B_8168_set_servo (ByVal AxisNo As Integer, ByVal On_Off As Integer) As Integer
B_8168_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 signal

- alm_logic=0, active LOW.
- alm_logic=1, active HIGH.

**alm_mode**: reacting modes when receiving ALARM signal.
alm_mode=0, motor immediately stops (Default)
alm_mode=1, motor decelerates then stops.

el_mode: reacting modes when receiving EL signal.
el_mode=0, motor immediately stops (Default)
el_mode=1, motor decelerates then stops.

inp_enable: INP function enable/disable
inp_enable=0, Disabled (Default)
inp_enable=1, Enabled

inp_logic: setting of active logic for INP signal
inp_logic=0, active LOW.
inp_logic=1, active HIGH.

erc_logic: setting of active logic for ERC signal
erc_logic=0, active LOW.
erc_logic=1, active HIGH.

erc_on_time: Setting of 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**: setting of active logic for INP signal
   - sd_logic=0, active LOW.
   - sd_logic=1, active HIGH.

**sd_latch**: setting of latch control for SD signal
   - sd_latch=0, do not latch.
   - sd_latch=1, latch.

**sd_mode**: setting the reacting mode of SD signal
   - sd_mode=0, slow down only
   - sd_mode=1, slow down then stop

@ **Return Code**
   ERR_NoError
6.13 Motion I/O Monitoring

@ Name
_8168_get_io_status – Get all the motion I/O status of cPCI-8168

@ Description
_8168_get_io_status:
Get all the I/O status for each axis. The definition for each bit is as follows:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RDY</td>
<td>RDY pin input</td>
</tr>
<tr>
<td>1</td>
<td>ALM</td>
<td>Alarm Signal</td>
</tr>
<tr>
<td>2</td>
<td>+EL</td>
<td>Positive Limit Switch</td>
</tr>
<tr>
<td>3</td>
<td>-EL</td>
<td>Negative Limit Switch</td>
</tr>
<tr>
<td>4</td>
<td>ORG</td>
<td>Origin Switch</td>
</tr>
<tr>
<td>5</td>
<td>DIR</td>
<td>DIR output</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>PCS</td>
<td>PCS signal input</td>
</tr>
<tr>
<td>8</td>
<td>ERC</td>
<td>ERC pin output</td>
</tr>
<tr>
<td>9</td>
<td>EZ</td>
<td>Index signal</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>11</td>
<td>Latch</td>
<td>Latch signal input</td>
</tr>
<tr>
<td>12</td>
<td>SD</td>
<td>Slow Down signal input</td>
</tr>
<tr>
<td>13</td>
<td>INP</td>
<td>In-Position signal input</td>
</tr>
<tr>
<td>14</td>
<td>SVON</td>
<td>Servo-ON output status</td>
</tr>
</tbody>
</table>

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_get_io_status(I16 AxisNo, U16 *io_sts);
Visual Basic (Windows)

B_8168_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. Where “1” is ON and “0” is OFF. ON/OFF state is read based on the corresponding set logic.

@ Return Code

ERR_NoError
6.14 Interrupt Control

@ Name

_8168_int_control – Enable/Disable INT service
_8168_set_int_factor – Set INT factor

For Windows:

_8168_int_enable – Enable event
_8168_int_disable – Disable event
_8168_get_int_status – Get INT Status
_8168_link_interrupt – Set link to interrupt call back function
_8168_int_enableA – Enable bit axis event
_8168_int_disableA – Disable bit axis event
_8168_map_axis_event – Map interrupt factor to axis event

For DOS:

_8168_get_int_type – Get INT type
_8168_enter_isr – Enter interrupt service routine
_8168_leave_isr – Leave interrupt service routine
_8168_get_event_int – Get event status
_8168_get_error_int – Get error status
_8168_get_irq_status – Get IRQ status
_8168_not_my_irq – Not My IRQ
_8168_isr0~9, a, b – Interrupt service routine

@ Description

_8168_int_control:
This function is used to control interrupt generating from the host PC. Once it is disabled, no interruptions will be generated from the Host PC.
The interrupt status of cPCI-8168 is separated by two independent parts: error type interrupt and event type interrupt. This function allows users to select event type interrupt factors of each axis. But the error type interrupt factors of each axis can’t be selected because it is non-maskable when the interrupt service is turned on by _8168_int_control(). The event interrupt status recodes the motion and comparator event under normal operation. This kind of interrupt status can be masked by _8168_set_int_factor(). The error interrupt status is for abnormal stop of axis. For example: End-limit, ALM …etc. This kind of interrupt cannot be masked. The following table contains the definitions of event type interrupt factors. By setting the relative bit as 1, cPCI-8168 can generate interrupt signal to host PC when the condition of bit is met.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normal Stop</td>
</tr>
<tr>
<td>1</td>
<td>Next command in buffer starts</td>
</tr>
<tr>
<td>2</td>
<td>Continuous pre-register 2 is empty and allow new command to write</td>
</tr>
<tr>
<td>3</td>
<td>Always set to 0</td>
</tr>
<tr>
<td>4</td>
<td>Acceleration Start</td>
</tr>
<tr>
<td>5</td>
<td>Acceleration End</td>
</tr>
<tr>
<td>6</td>
<td>Deceleration Start</td>
</tr>
<tr>
<td>7</td>
<td>Deceleration End</td>
</tr>
<tr>
<td>8</td>
<td>+Soft limit or comparator 1 is ON</td>
</tr>
<tr>
<td>9</td>
<td>-Soft limit or comparator 2 is ON</td>
</tr>
<tr>
<td>10</td>
<td>Error comparator or comparator 3 is ON</td>
</tr>
<tr>
<td>11</td>
<td>General comparator or comparator 4 is ON</td>
</tr>
<tr>
<td>12</td>
<td>Trigger comparator or comparator 5 is ON</td>
</tr>
<tr>
<td>13</td>
<td>(Reserved)</td>
</tr>
<tr>
<td>14</td>
<td>Counter is latched by LTC pin</td>
</tr>
<tr>
<td>15</td>
<td>Counter is latched by ORG pin</td>
</tr>
<tr>
<td>16</td>
<td>SD pin is turned on</td>
</tr>
<tr>
<td>17</td>
<td>Always set to 0</td>
</tr>
</tbody>
</table>
For Windows:

_8168_int_enable:
This function is used to pass a Windows event array for one card. The size of this array is 8, one-dimension. Each index indicates one axis.

_8168_int_disable:
This function is used to close the Windows event array passed to _8168_int_enable().

_8168_get_int_status:
This function allows user to identify the causes of interrupt. After user gets this information, the interrupt status register will be cleared to 0. There are two kinds of interrupt status. The first one is for error interrupt status which can’t be masked by _8168_set_int_factor(). The definitions in bit of error_int_status are as following:

table

<table>
<thead>
<tr>
<th>Bit</th>
<th>Interrupt Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+Soft Limit is ON and axis is stopped</td>
</tr>
<tr>
<td>1</td>
<td>-Soft Limit is ON and axis is stopped</td>
</tr>
<tr>
<td>2</td>
<td>Comparator 3 is ON and axis is stopped</td>
</tr>
<tr>
<td>3</td>
<td>General Comparator or comparator 4 is ON and axis is stopped</td>
</tr>
<tr>
<td>4</td>
<td>Trigger Comparator or comparator 5 is ON and axis is stopped</td>
</tr>
<tr>
<td>5</td>
<td>+End Limit is ON and axis is stopped</td>
</tr>
<tr>
<td>6</td>
<td>-End limit is ON and axis is stopped</td>
</tr>
<tr>
<td>7</td>
<td>ALM is ON and axis is stopped</td>
</tr>
<tr>
<td>8</td>
<td>CSTP is ON or stop_move_all stop the axis</td>
</tr>
<tr>
<td>9</td>
<td>CEMG is ON and axis is stopped</td>
</tr>
<tr>
<td>10</td>
<td>SD input is ON and axis is slow down then stop</td>
</tr>
<tr>
<td>11</td>
<td>Always 0</td>
</tr>
<tr>
<td>12</td>
<td>Interpolation operation error and stop</td>
</tr>
</tbody>
</table>
The second one is for event interrupt status which can be masked by `_8168_set_int_factor()`. The definitions in bit of `event_int_status` are as following:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Normally Stop</td>
</tr>
<tr>
<td>1</td>
<td>Next command in buffer started</td>
</tr>
<tr>
<td>2</td>
<td>Command pre-register 2 is empty and allow users to fill new command</td>
</tr>
<tr>
<td>3</td>
<td>Always 0</td>
</tr>
<tr>
<td>4</td>
<td>Acceleration Start</td>
</tr>
<tr>
<td>5</td>
<td>Acceleration End</td>
</tr>
<tr>
<td>6</td>
<td>Deceleration Start</td>
</tr>
<tr>
<td>7</td>
<td>Deceleration End</td>
</tr>
<tr>
<td>8</td>
<td>+Soft limit or comparator 1 is ON</td>
</tr>
<tr>
<td>9</td>
<td>-Soft limit or comparator 2 is ON</td>
</tr>
<tr>
<td>10</td>
<td>Error comparator or comparator 3 is ON</td>
</tr>
<tr>
<td>11</td>
<td>General comparator or comparator 4 is ON</td>
</tr>
<tr>
<td>12</td>
<td>Trigger comparator or comparator 5 is ON</td>
</tr>
<tr>
<td>13</td>
<td>Counter is reset by CLR input</td>
</tr>
<tr>
<td>14</td>
<td>Counter is reset by Latch input</td>
</tr>
<tr>
<td>15</td>
<td>Counter is reset by ORG input</td>
</tr>
<tr>
<td>16</td>
<td>SD input turns ON</td>
</tr>
<tr>
<td>17</td>
<td>Always 0</td>
</tr>
<tr>
<td>18</td>
<td>Always 0</td>
</tr>
<tr>
<td>19</td>
<td>CSTA input or start_move_all turns ON</td>
</tr>
<tr>
<td>20~31</td>
<td>Always 0</td>
</tr>
</tbody>
</table>
_8168_link_interrupt:
This function is used to link interrupt call back function. Don’t place too many codes inside the call back function. We suggest using Windows event to handle interrupt

_8168_int_enableA:
This function is used to pass a interrupt factor mapping event arrays for each axis. The size of this array is 8x6. Please enter the correct size of 2-dimension array and the type of HANDLE into this function. Notice that this function can’t be used if the other event enable function, _8168_int_enable() is used because it will cause conflicts while operating interrupt, and vice versa.

_8168_int_disableA:
This function is used to close the events passed to _8168_int_enableA().

_8168_map_axis_event:
This function is used for mapping an interrupt factor of axis to the event array created by _8168_int_enableA(). Users must use this function after the event array is created. Once users use this method to handle interrupt, they don’t need to get the event type interrupt status through _8168_get_int_status() because the event is already mapped to the interrupt factor by bit. As for error type interrupt status, the last element in event array of each axis is fixed for error type interrupt. If users get this event, they must use _8168_get_int_status() to find out which bit of error interrupt is ON.

For DOS
_8168_get_int_type:
This function is used to detect which kind of INP occurred.

_8168_enter_isr:
This function is used to inform system that process is now entering interrupt service routine.

_8168_leave_isr:
This function is used to inform the system that the process is now leaving interrupt service routine.
_8168_get_event_int:
This function is used to get event_int_status.

_8168_get_error_int:
This function is used to get error_int_status.

_8168_get_irq_status:
This function allows user to confirm if the designated card generates the INT signal to host PC.

_8168_not_my_irq:
This function must be called after knowing not the designated card generates the INT signal to host PC.

_8168_isr0, _8168_isr1, _8168_isr2, _8168_isr3, ..... _8168_isr9, _8168_isra, _8168_isrb:
Individual Interrupt service routine for card 0~11.

@ Syntax

C/C++ (DOS)

I16 _8168_int_control(U16 cardNo, U16 intFlag );
I16 _8168_set_int_factor(I16 AxisNo, U32 int_factor );
I16 _8168_get_int_type(I16 AxisNo, U16 *int_type);
I16 _8168_enter_isr(I16 AxisNo);
I16 _8168_leave_isr(I16 AxisNo);
I16 _8168_get_event_int(I16 AxisNo, U32 *event_int);
I16 _8168_get_error_int(I16 AxisNo, U32 *error_int);
I16 _8168_get_irq_status(U16 cardNo, U16 *sts);
I16 _8168_not_my_irq(I16 CardNo);
void interrupt _8168_isr0 (void);
void interrupt _8168_isr1 (void);
void interrupt _8168_isr2 (void);
void interrupt _8168_isr3 (void);
void interrupt _8168_isr4 (void);
void interrupt _8168_isr5 (void);
void interrupt _8168_isr6 (void);
void interrupt _8168_isr7 (void);
void interrupt _8168_isr8 (void);
void interrupt _8168_isr9 (void);
void interrupt _8168_isra (void);
void interrupt _8168_isrb (void);

C/C++ (Windows)
I16 _8168_int_control(U16 cardNo, U16 intFlag);
I16 _8168_set_int_factor(I16 AxisNo, U32 int_factor);
I16 _8168_int_enable(I16 CardNo, HANDLE *phEvent);
I16 _8168_int_disable(I16 CardNo);
I16 _8168_get_int_status(I16 AxisNo, U32 *error_int_status, U32 *event_int_status);
I16 _8168_link_interrupt(I16 CardNo, void (__stdcall *callbackAddr)(I16 IntAxisNoInCard));
I16 _8168_int_enableA(I16 CardNo, HANDLE *pheventA);
I16 _8168_int_disableA(I16 CardNo);
I16 _8168_map_axis_event(I16 AxisNo, I16 EventNo, U32 EventIntfactor);

Visual Basic (Windows)
B_8168_int_control (ByVal CardNo As Integer, ByVal intFlag As Integer) As Integer
B_8168_set_int_factor (ByVal AxisNo As Integer, ByVal int_factor As Long) As Integer
B_8168_int_enable (ByVal CardNo As Integer, ByVal phEvent As Long) As Integer
B_8168_int_disable (ByVal CardNo As Integer) As Integer
B_8168_get_int_status (ByVal AxisNo As Integer, error_int_status As Long, event_int_status As Long) As Integer
B_8168_link_interrupt (ByVal CardNo As Integer, ByVal lpCallBackProc As Long) As Integer
B_8168_int_enableA(ByVal CardNo As Integer, ByVal pheventA As Long) As Integer
B_8168_int_disableA(ByVal CardNo As Integer) As Integer
Function Library  159

B_8168_map_axis_event(ByVal AxisNo As Integer, ByVal EventNo As Integer, ByVal EventIntfactor As Long) As Integer

@ 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: DOS only, Interrupt type, (1: error int, 2: event int, 3: both happened )
*event_int: event_int_status, , refer to event_int_status table
*error_int: error_int_status, refer to error_int_status table by bit
*sts: DOS only, (0: not this card's IRQ, 1: this card's IRQ)
*phEvent: Windows only, event array handle pointer for interrupt. Ex. HANDLE phEvent[8];
*error_int_status: refer to error_int_status table by bit
*event_int_status: refer to event_int_status table by bit
*phEventA: Windows only, 2 dimensions event array handle pointer for interrupt. Ex. HANDLE phEventA[8][6];

@ Return Code

ERR_NoError
ERR_EventNotEnableYet
ERR_LinkIntError
ERR_CardNoErrot
6.15 Position Control and Counters

@ Name

_function_name_ – Get the value of feedback position counter
_function_name_ – Set the feedback position counter
_function_name_ – Get the value of command position counter
_function_name_ – Set the command position counter
_function_name_ – Get the value of position error counter
_function_name_ – Reset the position error counter
_function_name_ – Get the value of general counter
_function_name_ – Set the general counter
_function_name_ – Get the value of target position recorder
_function_name_ – Reset target position recorder
_function_name_ – Get remaining pulse till end of motion
_function_name_ – Get the ramping down point data

@ Description

_function_name_():
This function is used to read the value of feedback position counter. Note, this value has already been processed by move ratio. If move ratio is 0.5, than the value read will be twice as much as the counter value. The source of feedback counter is selectable by function _function_name_() to be external EA/EB or pulse output of cPCI-8168.
_function_name_():
This function is used to change the feedback position counter to the specified value. Note, the value to be set will be processed by the move ratio. If move ratio is 0.5, than the set value will be twice the given value.

_8168_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 cPCI-8168.

_8168_set_command():
This function is used to change the value of the command position counter.

_8168_get_error_counter():
This function is used to read the value of the position error counter.

_8168_reset_error_counter():
This function is used to clear the position error counter.

_8168_get_general_counter():
This function is used to read the value of the general counter.

_8168_set_general_counter():
This function is used to set the counting source of and change the value of the general counter. (By default, the source is pulser input.)

_8168_get_target_pos():
This function is used to read the value of target position recorder. The target position recorder is maintained by cPCI-8168 software driver. It records the position to settle down for current running motion.

_8168_reset_target_pos():
This function is used to set new values for the target position recorder. It is necessary to call this function during home return completion or when the new feedback counter value is set by function _8168_set_position().
_8168_get_rest_command():
This function is used to read remaining pulse counts until the current motion is finished.

_8168_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 pulse count, and it causes the axis start to decelerate when remaining pulse counts reach the data.

@ Syntax

**C/C++ (DOS, Windows 95/98/NT)**

```c
I16 _8168_get_position(I16 AxisNo, F64 *pos);
I16 _8168_set_position(I16 AxisNo, F64 pos);
I16 _8168_get_command(I16 AxisNo, I32 *cmd);
I16 _8168_set_command(I16 AxisNo, I32 cmd);
I16 _8168_get_error_counter(I16 AxisNo, I16 *error_counter);
I16 _8168_reset_error_counter(I16 AxisNo);
I16 _8168_get_general_counter(I16 AxisNo, F64 *CntValue);
I16 _8168_set_general_counter(I16 AxisNo, I16 CntSrc, F64 CntValue);
I16 _8168_get_target_pos(I16 AxisNo, F64 *T_pos);
I16 _8168_reset_target_pos(I16 AxisNo, F64 T_pos);
I16 _8168_get_rest_command(I16 AxisNo, I32 *rest_command);
I16 _8168_check_rdp(I16 AxisNo, I32 *rdp_command);
```

**Visual Basic (Windows)**

```vb
B_8168_get_position (ByVal AxisNo As Integer, Pos As Double) As Integer
B_8168_set_position (ByVal AxisNo As Integer, ByVal Pos As Double) As Integer
B_8168_get_command (ByVal AxisNo As Integer, cmd As Long) As Integer
B_8168_set_command (ByVal AxisNo As Integer, ByVal cmd As Long) As Integer
```
B_8168_get_error_counter (ByVal AxisNo As Integer, error_counter As Integer) As Integer
B_8168_reset_error_counter (ByVal AxisNo As Integer) As Integer
B_8168_get_general_counter (ByVal AxisNo As Integer, CntValue As Double) As Integer
B_8168_set_general_counter (ByVal AxisNo As Integer, ByVal CntSrc As Integer, ByVal CntValue As Double) As Integer
B_8168_get_target_pos (ByVal AxisNo As Integer, Pos As Double) As Integer
B_8168_reset_target_pos (ByVal AxisNo As Integer, ByVal Pos As Double) As Integer
B_8168_get_rest_command (ByVal AxisNo As Integer, rest_command As Long) As Integer
B_8168_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
rdp_command, *rdp_command: Ramping down point data
range: 0~16777215

\textbf{CntSrc: Source of general counter}

0 : Command pulses
1: EA/EB
2: PA/PB (Default)
3: CLK/2

@ \textbf{Return Code}

- ERR_NoError
- ERR_PosOutOfRange
6.16 Position Compare and Latch

@ Name

_8168_set_ltc_logic – Set the LTC logic
_8168_get_latch_data – Get latched counter data
_8168_set_soft_limit – Set soft limit
_8168_enable_soft_limit – Enable soft limit function
_8168_disable_soft_limit – Disable soft limit function
_8168_set_error_counter_check – Step-losing detection setup
_8168_set_general_comparator – Set general-purposed comparator
_8168_set_trigger_comparator – Set trigger comparator
_8168_set_trigger_type – Set the trigger output type
_8168_check_compare_data – Check current comparator data
_8168_check_compare_status – Check current comparator status
_8168_set_auto_compare – Set comparing data source for auto loading
_8168_build_compare_function – Build compare data via constant interval
_8168_build_compare_table – Build compare data via compare table
_8168_cmp_v_change – Speed change by comparator

@ Description

_8168_set_ltc_logic():
This function is used to set the logic of latch input. This function is applicable only for the last two axes in every cPCI-8168 card.
_8168_get_latch_data():
After the latch signal arrives, this function is used to read the latched value of counters.

_8168_set_soft_limit():
This function is used to set the value of soft limit.

_8168_enable_soft_limit() and _8168_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 the physical limit.

_8168_set_error_counter_check():
This function is used to enable the step losing checking facility. By giving a tolerance value, the cPCI-8168 will generate an interrupt (event_int_status, bit 10) when position error counter exceeds tolerance.

_8168_set_general_comparator():
This function is used to set the source and comparing value for general comparator. When the source counter value reaches the comparing value, the cPCI-8168 will generate an interrupt (event_int_status, bit 11).

_8168_set_trigger_comparator():
This function is used to set the comparing method and value for trigger comparator. When the feedback position counter value reached the comparing value, the cPCI-8168 will generate a trigger pulse via CMP pin and an interrupt (event_int_status, bit 12) will also be sent to the host PC. If _8168_set_auto_compare() is used, then comparing value set by this function will be ignored automatically.

Because the value will be replaced by the first value of the source assigned by _8168_set_auto_compare() function.

Note: Trigger output is only applicable for axis 4,5 in every cPCI-8168 card.
_8168_check_compare_data():
This function is used to obtain current comparing data of designated comparator.

_8168_check_compare_status():
This function is used to obtain the status of all comparators. When some comparators come into existence, the relative bit of cmp_sts will become ‘1’, otherwise ‘0’. Notice that the status will not be latched after compare condition is true.

_8168_set_auto_compare():
This function is used to set the comparing data source of trigger comparator. The source can be either a function or a table.

_8168_build_compare_function():
This function is used to build a comparing function (equal interval) by defining the start/end point and interval. There is no limitation on the max number counts of comparing data in Windows but has 4096 limitation in DOS.

Note: Please make sure that interruptions of this card do not happen frequently while the comparing trigger is active. Otherwise the comparing trigger will be stopped due to real-time issue.

_8168_build_compare_table():
This function is used to build a comparing table by defining data array. There is no limitation on the max number counts of comparing data but has 4096 limitation in DOS. The compare data table could be non-equal interval for the compare data but it must be an increasing or decreasing data table.

Note: Please make sure that interruption of this card do not happened frequently while the comparing trigger is active. Otherwise the comparing trigger will be stopped due to real-time issue.

_8168_cmp_v_change():
This function is used to setup comparator velocity change function. It is like a v_change() function but acts when general comparator comes into existence. When this function is issued, the parameter “CmpAction” of
_8168_set_general_comparator() must be set to ‘3’. The compare data is also set by _8168_set_general_comparator(). The remain distance, the compare point’s velocity, the new velocity and the acceleration time are set by _8168_cmp_v_change().

@ Syntax

**C/C++ (DOS, Windows)**

```c
I16 _8168_set_ltc_logic(I16 AxisNo_2or3, I16 ltc_logic);
I16 _8168_get_latch_data(I16 AxisNo, I16 Counter, F64 *Pos);
I16 _8168_set_soft_limit(I16 AxisNo, I32 PLimit, I32 NLimit);
I16 _8168_disable_soft_limit(I16 AxisNo);
I16 _8168_enable_soft_limit(I16 AxisNo, I16 Action);
I16 _8168_set_error_counter_check(I16 AxisNo, I16 Tolerance, I16 On_Off);
I16 _8168_set_general_comparator(I16 AxisNo, I16 CmpSrc, I16 CmpMethod, I16 CmpAction, F64 Data);
I16 _8168_set_trigger_comparator(I16 AxisNo, I16 CmpMethod, F64 Data);
I16 _8168_check_compare_data(I16 AxisNo, I16 CmpSrc, F64 *Pos);
I16 _8168_check_compare_status(I16 AxisNo, U16 *cmp_sts);
I16 _8168_set_auto_compare(I16 AxisNo, I16 SelectSrc);
I16 _8168_cmp_v_change(I16 AxisNo, F64 Res_dist, F64 oldvel, F64 newvel, F64 AccTime)
I16 _8168_build_compare_function(I16 AxisNo, F64 Start, F64 End, F64 Interval, I16 Device);
I16 _8168_build_compare_table(I16 AxisNo, F64 *TableArray, I16 Size, I16 Device);
```

**Visual Basic (Windows)**

```vb
B_8168_set_ltc_logic (ByVal AxisNo As Integer, ByVal ltc_logic As Integer) As Integer
```
B_8168_get_latch_data (ByVal AxisNo As Integer, ByVal Counter As Integer, Pos As Double) As Integer
B_8168_set_soft_limit (ByVal AxisNo As Integer, ByVal PLimit As Long, ByVal NLimit As Long) As Integer
B_8168_disable_soft_limit (ByVal AxisNo As Integer) As Integer
B_8168_enable_soft_limit (ByVal AxisNo As Integer, ByVal Action As Integer) As Integer
B_8168_set_error_counter_check (ByVal AxisNo As Integer, ByVal Tolerance As Integer, ByVal On_Off As Integer) As Integer
B_8168_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_8168_set_trigger_comparator (ByVal AxisNo As Integer, ByVal CmpMethod As Integer, ByVal Data As Double) As Integer
B_8168_check_compare_data (ByVal AxisNo As Integer, ByVal CmpSrc As Integer, Pos As Double) As Integer
B_8168_check_compare_status (ByVal AxisNo As Integer, cmp_sts As Integer) As Integer
B_8168_set_auto_compare (ByVal AxisNo As Integer, ByVal SelectSrc As Integer) As Integer
B_8168_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_8168_build_compare_table (ByVal AxisNo As Integer, TableArray As Double, ByVal Size As Integer, ByVal Device As Integer) As Integer
B_8168_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
ltc_logic: 0 means active low, 1 means active high
AxisNo: Axis number

Counter: Specified Counter
  Counter = 0, Command counter
  Counter = 1, Feedback counter
  Counter = 2, Error Counter
  Counter = 3, General Counter

Pos: Latched counter value,

Plimit: Soft limit value in positive direction

NLimit: Soft limit value in 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
  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, Immediately stop
- CmpAction =2, slow down then stop
- CmpAction =3, speed change

**Data:** Comparing value,

**cmp_sts:** status of comparator

<table>
<thead>
<tr>
<th>Bit</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+Softlimit On</td>
</tr>
<tr>
<td>1</td>
<td>-SoftLimit On</td>
</tr>
<tr>
<td>2</td>
<td>Error counter comparator On</td>
</tr>
<tr>
<td>3</td>
<td>General comparator On</td>
</tr>
<tr>
<td>4</td>
<td>Trigger comparator On (for 0 , 1 axis only)</td>
</tr>
</tbody>
</table>

**SelectSrc:** The comparing data source

- SelectSrc =0, disable auto compare
- SelectSrc =1, use FIFO as compare data storage

**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 =0, Useless
- Device =1, use FIFO as compare data storage

**Res_dist:** The remain distance from the compare point. After comparison, the original target position will be ignored, and the axis 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

_8168_set_continuous_move – Enable continuous motion
_8168_check_continuous_buffer – check if the buffer is empty

@ Description

_8168_set_continuous_move():
This function is necessary before and after continuous motion.

_8168_check_continuous_buffer():
This function is used to detect if the command buffer is empty or not. If the command buffer is empty, user may write next motion command into it. If the command buffer is not empty, the users new command will overwrite previous.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_set_continuous_move(I16 AxisNo, I16 conti_flag);
I16 _8168_check_continuous_buffer(I16 AxisNo);

Visual Basic (Windows)

B_8168_set_continuous_move (ByVal AxisNo As Integer, ByVal conti_flag As Integer) As Integer
B_8168_check_continuous_buffer (ByVal AxisNo As Integer) As Integer

@ Argument

AxisNo: axis number designated
conti_flag: Flag for continuous motion
    conti_flag = 0, one-shoot motion, end of continuous motion
    conti_flag = 1, continuous motion, start of continuous motion
@ Return Value

ERR_NoError

Return value of _8168_check_continuous_buffer():

0: Continuous buffer is not full
1: Continuous buffer is full
6.18 Multiple Axes Simultaneous Operation

@ Name

_8168_set_tr_move_all – Multi-axis TR mode simultaneous operation setup.

_8168_set_sr_move_all – Multi-axis SR mode simultaneous operation setup.

_8168_set_ta_move_all – Multi-axis TA mode simultaneous operation setup.

_8168_set_sa_move_all – Multi-axis SA mode simultaneous operation setup.

_8168_start_move_all – Start motion simultaneously for all configured axis

_8168_stop_move_all – Stop motion simultaneously for all configured axis

@ Description

These functions are 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 move axes are specified by parameter “AxisArray” and the number of axes are defined by parameter “TotalAxes” in _8168_set_tr_move_all(), _8168_set_ta_move_all(), _8168_set_sr_move_all() or _8168_set_sa_move_all().

When properly setup with _8168_set_XX_move_all(), the function _8168_start_move_all() will cause all specified axes to begin trapezoidal relative moving, and _8168_stop_move_all() will stop them. Both functions guarantee that motion Start/Stop on all specified axes at the same time. Note that it is necessary to make connections according to Section 3.14 on CN3 if these two functions are needed.

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 (simultaneous motion).
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};

    _8168_set_tr_move_all(2, axes, dist, str_vel, max_vel, Tacc, Tdec);
    _8168_start_move_all(axes[0]);

    return ERR_NoError;
}

@ Syntax

C/C++ (DOS, Windows)
I16 _8168_set_tr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8168_set_sr_move_all(I16 TotalAxes, I16 *AxisArray, F64 *DistA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 _8168_set_ta_move_all(I16 TotalAxes, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA);
I16 _8168_set_sa_move_all(I16 TotalAxes, I16 *AxisArray, F64 *PosA, F64 *StrVelA, F64 *MaxVelA, F64 *TaccA, F64 *TdecA, F64 *SVaccA, F64 *SVdecA);
I16 _8168_start_move_all(I16 FirstAxisNo);
I16 _8168_stop_move_all(I16 FirstAxisNo);

Visual Basic (Windows)
B_8168_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_8168_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_8168_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_8168_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_8168_start_move_all(ByVal FirstAxisNo As Integer);

B_8168_stop_move_all(ByVal FirstAxisNo 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 unit of pulse

* StrVelA: starting velocity array in unit of pulse per second

* MaxVelA: maximum velocity array in unit of pulse per second

* TaccA: acceleration time array in unit of second

* TdecA: deceleration time array in unit of second

* SvaccA: specified velocity interval in which S-curve acceleration is performed. Note: 0, for pure S-Curve

* SvdecA: specified velocity interval in which S-curve acceleration is performed. Note: 0, for pure S-Curve

FirstAxisNo: the first element in AxisArray.

@ Return Code

ERR_NoError
ERR_SpeedError
6.19 HSL Initialization

@ Name

W_HSL_Start – Start HSLink communication with specified maximum slave ID

W_HSL_Auto_Start – Start HSLink communication by automatically detecting maximum slave ID

W_HSL_Stop – Stop HSLink communication

@ Description

W_HSL_Start():
This function is used to set the maximum ID of connected slave I/O module and start to scan these slave I/O modules.

W_HSL_Auto_Start():
This function is used to auto-detect the maximum ID of connected slave I/O module and start HSLink communication.

W_HSL_Stop():
This function stop communication of the HSL master to slaves.

@ Syntax

C/C++ (DOS, Windows)

I16 W_HSL_Auto_Start(I16 card_ID, I16 set_ID);
I16 W_HSL_Start(I16 card_ID, I16 set_ID, I16 sat_num);
I16 W_HSL_Stop(I16 card_ID, I16 set_ID);

Visual Basic (Windows)

W_HSL_Auto_Start (ByVal card_ID As Integer, ByVal set_ID As Integer) As Integer
W_HSL_Start (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal sat_num As Integer) As Integer
W_HSL_Stop Lib (ByVal card_ID As Integer, ByVal set_ID As Integer) As Integer
@ Argument

card_ID: Designated card number

set_ID: 0 for the first HSLink master IC, 1 for the second.

sat_num: The maximum slave ID connect to the HSL master card by the

@ Return Value

ERR_NoError
ERR_ConnectIndex
ERR_InvalidBoardNumber
6.20 HSL Connection Information

@ Name

W_HSL_Connect_Status – Get slave module's status
W_HSL_Slave_Live – Check if the slave alive

@ Description

W_HSL_Connect_Status():
This function is to read the communication status of specified slave I/O module.

W_HSL_Slave_Live():
This function is to get status of specified slave I/O module (live or die).

@ Syntax

C/C++ (DOS, Windows)

I16 W_HSL_Connect_Status(I16 card_ID, I16 set_ID, I16 slave_No, U8* sts_data);
I16 W_HSL_Slave_Live(I16 card_ID, I16 set_ID, I16 slave_No, U8* live_data);

Visual Basic (Windows)

W_HSL_Connect_Status (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, sts_data As Integer) As Integer
W_HSL_Slave_Live (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, live_data As Integer) As Integer

@ Argument

card_ID: Designated card number
set_ID: 0 for the first HSLink master IC, 1 for the second.
slave_No: Specified slave I/O module.
sts_data: The communication status of this slave I/O module.
Bit 0 is Data_Req bit.

Bit 2 is for CHK1. (If Bit2 is 1. It mean that there is 1 time communication error).

Bit 3 is for CHK3. (If Bit3 is 1. It mean that there are 3 times communication error).

Bit 4, BIT 5 and BIT 6 bits are for CHK7. (If Bit4, Bit5 and Bit6 all are 1. It mean that there are 7 times communication error).

@ Return Value

ERR_NoError
ERR_InvalidBoardNumber
ERR_ConnectIndex
ERR_SatelliteNumber
ERR_NotADLinkSlaveType
6.21 HSL DIO Read/Write

@ Name

W_HSL_DIO_In – Digital input from all bits in one slave module
W_HSL_DIO_Channel_In – Digital input from one bit
W_HSL_DIO_Out – Digital output for all bits in one slave module
W_HSL_DIO_Channel_Out – Digital output for one bit
W_HSL_DIO_Memory_Out – Set output values of all slave modules
W_HSL_DIO_Memory_In – Get input values from all slave modules

@ Description

W_HSL_DIO_In():
This function is to read the digital input value of the slave I/O module.

W_HSL_DIO_Channel_In():
This function is to read the digital input value of the specified channel on the slave I/O module.

W_HSL_DIO_Out():
This function is to write the digital output value to the slave I/O module.

W_HSL_DIO_Channel_Out():
This function is to write the digital output value to the specified digital channel of slave I/O module.

W_HSL_DIO_Memory_Out():
This function is to write all digital output values to all slave I/O modules which the set value is set_ID and the card no is card_ID. In this function, user can write all digital output values to all the slave I/O modules at one time.
**W_HSL_DIO_Memory_In():**

This function is used to read the digital input values from all slave I/O modules which the set value is set_ID and card no is card_ID. In this function, user can read all digital input values from all slave I/O modules at one time.

@ Syntax

**C/C++ (DOS, Windows)**

```c
I16 W_HSL_DIO_In(I16 card_ID, I16 set_ID, I16 slave_No, U32* in_data)
I16 W_HSL_DIO_Channel_In(I16 card_ID, I16 set_ID, I16 slave_No, I16 channel, U16* in_data)
I16 W_HSL_DIO_Out(I16 card_ID, I16 set_ID, I16 slave_No, U32 out_data)
I16 W_HSL_DIO_Channel_Out(I16 card_ID, I16 set_ID, I16 slave_No, I16 channel, U16 out_data)
I16 W_HSL_DIO_Memory_Out(I16 card_ID, I16 set_ID, U16 *data_out)
I16 W_HSL_DIO_Memory_In(I16 card_ID, I16 set_ID, U16 *data_in)
```

**Visual Basic (Windows)**

```vb
W_HSL_DIO_In (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, in_data As Long) As Integer
W_HSL_DIO_Channel_In (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, ByVal channel_No As Integer, in_data As Integer) As Integer
W_HSL_DIO_Out (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, ByVal out_data As Long) As Integer
W_HSL_DIO_Channel_Out (ByVal card_ID As Integer, ByVal set_ID As Integer, ByVal slave_No As Integer, ByVal channel_No As Integer, ByVal out_data As Integer) As Integer
W_HSL_DIO_Memory_Out (ByVal card_ID As Integer, ByVal set_ID As Integer, Buffer As Any) As Integer
```
W_HSL_DIO_Memory_In (ByVal card_ID As Integer,
    ByVal set_ID As Integer, Buffer As Any) As Integer

@ Argument

card_ID: Designated card number
set_ID: 0 for the first HSLink master IC, 1 for the second.
slave_No: Specified slave I/O module.
in_data: the digital input data of this slave I/O module. In this value.
    Channel 0 data is assign to bit 0,
    Channel 1 data is assign to bit 1....etc.
channel_No: Specify the digital input channel of the slave I/O module which want to perform this function. The valid value is described as below:
    HSL-R8DI16 : 0 ~ 15.
    HSL-DI16DO16 : 0 ~ 15.
    HSL-DI32 : 0 ~ 31.

@ Return Value
    ERR_NoError,
    ERR_InvalidBoardNumber
    ERR_ConnectIndex
    ERR_SatelliteNumber
    ERR_SatelliteType
    ERR_NotADLinkSlaveType
    ERR_ChannelNumber
    ERR_NotADLinkSlaveType
6.22 General-purposed DIO

@ Name

_8168_d_output – Digital output
_8168_d_input – Digital input
_8168_hd_output – High speed digital output
_8168_hd_input – High speed digital input

@ Description

_8168_d_output():
Set the on_off status for general-purposed digital output pin.

_8168_d_input():
Read status of general-purposed digital input pin.

_8168_hd_output():
Set the on_off status for general-purposed high speed DO pin.

_8168_hd_input():
Read status of general-purposed high speed DI pin.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_d_output(I16 AxisNo, I16 value)
I16 _8168_d_input(I16 AxisNo)
I16 _8168_hd_output(I16 AxisNo, I16 value)
I16 _8168_hd_input(I16 AxisNo)

Visual Basic (Windows)

B_8168_d_output (ByVal AxisNo As Integer, ByVal value As Integer) As Integer
B_8168_d_input (ByVal AxisNo As Integer) As Integer
B_8168_hd_output (ByVal AxisNo As Integer, ByVal value As Integer) As Integer
B_8168_hd_input (ByVal AxisNo As Integer) As Integer
@ Argument

AxisNo: Designated axis number

Value: On-Off Value for output
  Value =0, output OFF
  Value =1, output ON

@ Return Value
 (ERR_NoError)
6.23 General-purposed AIO

@ Name
_8168_read_ad – Read AD value
_8168_write_da – Write DA value

@ Description
_8168_read_ad():
Read value for specified AD channel
_8168_write_da():
Write value into specified DA channel

@ Syntax

C/C++ (DOS, Windows)

F64 _8168_read_ad(I16 CardNo, I16 ch);
I16 _8168_write_da(I16 CardNo, I16 ch, F64 v);

Visual Basic (Windows)

B_8168_read_ad (ByVal card As Integer, ByVal ch As Integer) As Double
B_8168_write_da (ByVal card As Integer, ByVal ch As Integer, ByVal v As Double) As Integer

@ Argument
CardNo: Designated card number
Ch: Designated channel number
v: Voltage value, range: -10.0 ~ +10.0

@ Return Value
ERR_NoError
6.24 High Resolution Timer Functions

@ Name

_8168_set_h_timer – Start high resolution timer counter
_8168_h_timer – Get high resolution timer counter value
_8168_ssleep – Sleep according to a high resolution timer
_8168_delay_time – Delay for a while
_8168_delay_time_mt – Delay for a while within thread

@ Description

_8168_set_h_timer:
This function is used for starting a high resolution timer on cPCI-8168. The timer base clock is 40 nero-second. Once the timer is set, the hardware interrupt will start to generate frequently. Don’t set the timer interval too small. Suggested value is more than 100 micro-second. That means the suggested minimum value is 2500 for this function. The minimum value for this function is 25. Notice that the _8168_int_control() must be enable first.

_8168_h_timer:
This function is used for reading back the 32-bit timer counter value after the timer counter is started.

_8168_ssleep:
This function is used for pause program for a specific time. It will not consume CPU time under Windows. It is waiting for timer interrupt event by WaitForSingleObject() Windows API. Users can create up to 64 sleep functions at the same time. The minimum time out time for WaitForSingleObject() is 200ms. The time-out formula is twice the setting value. The sleep time is (setting value x time base). The time base is come from _8168_set_h_timer().

_8168_delay_time:
This function is used to pause the program for a specific amount of time. It will consume CPU time and occupy the axis as a motion axis. Once it is called, the axis will start a dummy movement with a
duration equal to the function’s delay time parameter. The minimum delay time is 0.1ms.

_8168_delay_time_mt:
This function is used to pause the program for a specific amount of time. It will not consume CPU time under Windows by using `WaitForSingleObject()`. Users must pass a normally stopped event of this axis into this function. It will occupy one axis as a motion axis. Once it is called, the axis will start a dummy movement with a duration equal to the function’s delay time parameter. The minimum delay time is 0.1ms. Notice that time `Timeout_ms` parameter is for `WaitForSingleObject()`. Please don’t use a value smaller than 10ms because of the inaccuracy of Windows timer.

@ Syntax

**C/C++ (Windows)**

```c
I16 _8168_set_h_timer(I16 CardNo, I32 T_base_Mulx, I16 Enable);
I16 _8168_h_timer(I16 CardNo);
I16 _8168_ssleep(I16 CardNo, I16 timer_no, I32 sleep_time_mulx);
I16 _8168_delay_time_mt(I16 AxisNo, F64 miniSecond, I32 Timeout_ms, HANDLE waitEvent);
I16 _8168_delay_time(I16 AxisNo, F64 miniSecond);
```

**Visual Basic (Windows)**

```vb
B_8168_set_h_timer (ByVal CardNo As Integer, ByVal T_base_Mulx As Long, ByVal Enable As Integer) As Integer
B_8168_h_timer (ByVal CardNo As Integer) As Long
B_8168_ssleep (ByVal CardNo As Integer, ByVal timer_no As Integer, ByVal sleep_time_mulx As Long) As Integer
B_8168_delay_time_mt (ByVal AxisNo As Integer, ByVal miniSecond As Double, ByVal Timeout_ms As Long, ByVal waitEvent As Long) As Integer
B_8168_delay_time (ByVal AxisNo As Integer, ByVal miniSecond As Double) As Integer
```
@ Argument

CardNo: Card number designated

T_base_Mulx: The counter value for timer counter and interrupt. The counter base clock is fixed to 40ns. Suggested timer counter is more than 100us so the minimum value for suggestion is 2500

Enable: 1 for enable and 0 for disable

timer_no: 0~63

sleep_time_mulx: according to set_h_timer() base. 1 is one timer base, 2 is twice of timer base and so on

miniSecond: The minimum value is 0.1

Timeout_ms: The minimum value is 10

waitEvent: The event handle of axis normally stop

@ Return Value

ERR_NoError
ERR_CardNoError
ERR_UseDefault
ERR_AxisRangeError
ERR_DelayTimeError
ERR_EventInvalid
ERR_DelayDistError
ERR_WaitAbandoned
ERR_WaitDelayTimeOut
ERR_Unknown
ERR_FIFOModeOn
6.25 DI Change of state

@ Name

_8168_set_d_cos – Enable DI pin change of state function
_8168_wait_d_cos – Wait DI pin change of state event
_8168_set_hd_cos – Enable HDI pin change of state function
_8168_wait_hd_cos – Wait HDI pin change of state event
_8168_get_cos_status – Get change of state status

@ Description

_8168_set_d_cos:
This function is used for enable/disable the DI pins’ change of state function. The DI change of state interrupt will be enabled first then it will reset the corresponding event when calling this function. Notice the _8168_int_control() must be set when using this function. All axes have one DI pin for this function.

_8168_wait_d_cos:
This function is used for waiting the change of state event. Once this function is called, it will start WaitForSingleObject() to wait the event. No reset event is inside this functions. Notice that the time out time must be set more than 10ms to prevent the inaccuracy of Windows timer.

_8168_set_hd_cos:
This function is used for enable/disable the HDI pins’ change of state function. The DI change of state interrupt will be enabled first then it will reset the corresponding event when calling this function. Notice the _8168_int_control() must be set when using this function. Only the first 4 axes of one card has this function.

_8168_wait_hd_cos:
This function is used for waiting the change of state event. Once this function is called, it will start WaitForSingleObject() to wait the event. No reset event inside this functions. Notice that the
time out time must be set more than 10ms to prevent the inaccuracy of Windows timer.

_8168_get_cos_status:
This function is used for getting the change of state status in one card. Calling the status will clear the current status. It is not suggested that this function is used during interrupt for COS functions.

@ Syntax

C/C++ (DOS, Windows)

I16 _8168_set_d_cos(I16 AxisNo, I16 Logic, I16 Enable);
I16 _8168_wait_d_cos(I16 AxisNo, I32 Time_Out);
I16 _8168_set_hd_cos(I16 AxisNo, I16 Logic, I16 Enable);
I16 _8168_wait_hd_cos(I16 AxisNo, I32 Time_Out);
U16 _8168_get_cos_status(I16 CardNo);

Visual Basic (Windows)

_8168_set_d_cos( ByVal AxisNo As Integer, ByVal Logic As Integer, ByVal Enable As Integer) As Integer
_8168_wait_d_cos(ByVal AxisNo As Integer, ByVal Time_Out As Long) As Integer
_8168_set_hd_cos(ByVal AxisNo As Integer, ByVal Logic As Integer, ByVal Enable As Integer) As Integer
_8168_wait_hd_cos(ByVal AxisNo As Integer, ByVal Time_Out As Long) As Integer
_8168_get_cos_status(ByVal CardNo As Integer) As Integer

@ Argument

AxisNo: axis number designated
Logic: 0 for rising edge, 1 for falling edge
Enable: 0 for disable, 1 for enable
Time_Out: The minimum value is 10
**Cos_status**: return by \_8168\_get\_cos\_status()

- Bit0~7 for DI0~7 change of state status, 1 for happened.
- Bit8~11 for HDI0~3 change of status status 1 for happened
- Bit15 is for change of state status when any bit has COS happened.

**@ Return Value**

- ERR\_NoError
- ERR\_AxisRangeError
- ERR\_WaitAbandoned
- ERR\_WaitDCOSTimeOut
- ERR\_WaitHSCOSTimeOut
- ERR\_Unknown
# Appendix

## 7.1 General I/O usages on PCL6045B

<table>
<thead>
<tr>
<th>PCL6045B #1</th>
<th>PCL6045B #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3x</td>
<td>P3x</td>
</tr>
<tr>
<td>DIN1</td>
<td>DIN5</td>
</tr>
<tr>
<td>P3y</td>
<td>P3y</td>
</tr>
<tr>
<td>DIN2</td>
<td>DIN6</td>
</tr>
<tr>
<td>P3z</td>
<td>P3z</td>
</tr>
<tr>
<td>DIN3</td>
<td>DIN7</td>
</tr>
<tr>
<td>P3u</td>
<td>P3u</td>
</tr>
<tr>
<td>DIN4</td>
<td>DIN8</td>
</tr>
<tr>
<td>P4x</td>
<td>P4x</td>
</tr>
<tr>
<td>HSOUT1</td>
<td>H</td>
</tr>
<tr>
<td>P4y</td>
<td>P4y</td>
</tr>
<tr>
<td>HSOUT2</td>
<td>H</td>
</tr>
<tr>
<td>P4z</td>
<td>P4z</td>
</tr>
<tr>
<td>HSOUT3</td>
<td>H</td>
</tr>
<tr>
<td>P4u</td>
<td>P4u</td>
</tr>
<tr>
<td>HSOUT4</td>
<td>H</td>
</tr>
<tr>
<td>P5x</td>
<td>P5x</td>
</tr>
<tr>
<td>HSIN1 &amp; LTCA</td>
<td>H</td>
</tr>
<tr>
<td>P5y</td>
<td>P5y</td>
</tr>
<tr>
<td>HSIN2 &amp; LTCB</td>
<td>H</td>
</tr>
<tr>
<td>P5z</td>
<td>P5z</td>
</tr>
<tr>
<td>HSIN3 &amp; LTCC</td>
<td>H</td>
</tr>
<tr>
<td>P5u</td>
<td>P5u</td>
</tr>
<tr>
<td>HSIN4 &amp; LTCD</td>
<td>H</td>
</tr>
<tr>
<td>P6x</td>
<td>P6x</td>
</tr>
<tr>
<td>STP1_OE</td>
<td>H</td>
</tr>
<tr>
<td>P6y</td>
<td>P6y</td>
</tr>
<tr>
<td>STA1_OE</td>
<td>H</td>
</tr>
<tr>
<td>P6z</td>
<td>P6z</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>P6u</td>
<td>P6u</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>P7x</td>
<td>P7x</td>
</tr>
<tr>
<td>CMPX_ORG</td>
<td>CMP1</td>
</tr>
<tr>
<td>P7y</td>
<td>P7y</td>
</tr>
<tr>
<td>CMPY_ORG</td>
<td>CMP2</td>
</tr>
<tr>
<td>P7z</td>
<td>P7z</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>P7u</td>
<td>P7u</td>
</tr>
<tr>
<td>H</td>
<td>CEMG5~8</td>
</tr>
</tbody>
</table>
### 7.2 Dual purpose pins on CN3~6

<table>
<thead>
<tr>
<th>CN3</th>
<th>CN4</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>SD1,LTC1,PCS1</td>
</tr>
<tr>
<td>57</td>
<td>SD2,LTC2,PCS2</td>
</tr>
<tr>
<td>26</td>
<td>HSIN1,LTC5,SD5</td>
</tr>
<tr>
<td>60</td>
<td>HSIN2,LTC6,SD6</td>
</tr>
<tr>
<td></td>
<td>23 SD3,LTC3,PCS3</td>
</tr>
<tr>
<td>57</td>
<td>SD4,LTC4,PCS4</td>
</tr>
<tr>
<td>26</td>
<td>HSIN3,LTC7,SD7</td>
</tr>
<tr>
<td>60</td>
<td>HSIN4,LTC8,SD8</td>
</tr>
</tbody>
</table>

The numbers 1~8 map to axis0~7
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 which can be downloaded from: http://rma.adlinktech.com/policy/.

2. All ADLINK products come with a limited two-year warranty, one year for products bought in China:
   - The warranty period starts on the day the product is shipped from ADLINK’s factory.
   - Peripherals and third-party products not manufactured by ADLINK will be covered by the original manufacturers’ warranty.
   - For products containing storage devices (hard drives, flash cards, etc.), please back up your data before sending them for repair. ADLINK is not responsible for any loss of data.
   - Please ensure the use of properly licensed software with our systems. ADLINK does not condone the use of pirated software and will not service systems using such software. ADLINK will not be held legally responsible for products shipped with unlicensed software installed by the user.
   - For general repairs, please do not include peripheral accessories. If peripherals need to be included, be certain to specify which items you sent on the RMA Request & Confirmation Form. ADLINK is not responsible for items not listed on the RMA Request & Confirmation Form.
3. Our repair service is not covered by ADLINK's guarantee in the following situations:

- Damage caused by not following instructions in the User's Manual.
- Damage caused by carelessness on the user's part during product transportation.
- Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
- Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
- Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
- Damage from improper repair by unauthorized ADLINK technicians.
- Products with altered and/or damaged serial numbers are not entitled to our service.
- This warranty is not transferable or extendible.
- Other categories not protected under our warranty.

4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.

5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

If you have any further questions, please email our FAE staff: service@adlinktech.com.