



**ADLINK**  
TECHNOLOGY INC.

**DB-8152**  
**Modulized Function Board - Motion**  
**User's Manual**

**Manual Rev.** 2.00  
**Revision Date:** May 20, 2008  
**Part No:** 50-14010-2000



Recycled Paper

**Advance Technologies; Automate the World.**



Copyright 2008 ADLINK TECHNOLOGY INC.

All Rights Reserved.

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

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

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

## Trademarks

NuDAQ, NuIPC, DAQBench are registered trademarks of ADLINK TECHNOLOGY INC.

Product names mentioned herein are used for identification purposes only and may be trademarks and/or registered trademarks of their respective companies.

# Getting Service from ADLINK

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](mailto: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.

Company Information	
Company/Organization	
Contact Person	
E-mail Address	
Address	
Country	
TEL	FAX:
Web Site	
Product Information	
Product Model	
Environment	OS: M/B: CPU: Chipset: BIOS:

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



# Table of Contents

<b>Table of Contents.....</b>	<b>i</b>
<b>List of Tables.....</b>	<b>iv</b>
<b>List of Figures .....</b>	<b>v</b>
<b>1 Introduction .....</b>	<b>1</b>
1.1 Specifications.....	3
1.2 Supported Software .....	5
<b>2 Installation .....</b>	<b>7</b>
2.1 Package Contents .....	7
2.2 DB-8152 and External I/F Outline Drawing.....	8
2.3 DB-8152 Hardware Installation.....	10
Hardware Configuration .....	10
Installation Procedures .....	10
2.4 Troubleshooting: .....	12
2.5 Software Driver Installation.....	13
<b>3 Signal Connections.....</b>	<b>15</b>
3.1 CN3 and CN4 Connectors Pin Definition .....	15
CN3 Pin Description: .....	16
CN4 Pin Description: .....	17
3.2 Pulse Output Signals SOUT and SDIR.....	18
3.3 Encoder Feedback Signals MEA, MEB, MEZ, SEA and SEB	
20	
3.4 Origin Signal SORG.....	23
3.5 End-Limit Signals SPEL and SMEL.....	24
3.6 In-Position Signal SINP .....	25
3.7 Alarm Signal SALM.....	26
3.8 Deviation Counter Clear Signal SERC .....	27
3.9 Compare Output Signal: SCMP 0-7.....	29
<b>4 Operation Theory .....</b>	<b>31</b>
4.1 Motion Control Modes.....	33
ECAM Control Mode .....	33
4.2 General Motion Control Mode.....	36
Relative Position Move .....	36

	Velocity Move Mode .....	36
	Table Move Mode .....	36
	Home Return Mode .....	37
4.3	Motor Driver Interface .....	38
	Pulse Command Output Interface .....	38
	Pulse Feedback Input Interface .....	40
	In Position Signal .....	42
	Servo Alarm Signal .....	42
	Error Clear Signal .....	43
4.4	Mechanical switch interface .....	44
	Original or Home Signal .....	44
	End-limit Switch Signal .....	44
	Emergency Stop Input .....	44
4.5	The Counters .....	45
	Slave Command Position Counter .....	45
	Feedback Position Counter .....	46
	Remaining Position Counters .....	46
4.6	The Comparators .....	47
	General Comparators .....	47
	Position Latch .....	47
4.7	Interrupt Control .....	48
<b>5</b>	<b>MotionCreatorPro .....</b>	<b>51</b>
5.1	Execute MotionCreatorPro .....	51
5.2	About MotionCreatorPro .....	52
5.3	MotionCreatorPro Introduction .....	53
	Main Menu .....	53
	Select Menu .....	54
	Card Information Menu .....	55
	DB-8152 ECAM Menu .....	56
	Configuration Menu .....	60
	ECAM Table setup Menu .....	62
	Comparator Table setup Menu .....	65
	Help Menu .....	66
<b>6</b>	<b>Function Library .....</b>	<b>67</b>
6.1	List of Functions .....	68
6.2	C/C++ Programming Library .....	71
6.3	System .....	72
6.4	Pulse Input/Output Configuration .....	74

6.5	ECAM Mode Motion.....	76
6.6	Slave Motion .....	79
6.7	Motion Interface I/O .....	81
6.8	Interrupt Control.....	87
6.9	Position Control and Counters.....	92
6.10	Position Compare and Latch .....	96
<b>Appendix.....</b>		<b>99</b>
7.1	Electronic Cam Examples.....	99
	Monitoring an Object's Appearance .....	99
	Tracking Device .....	99
	Coil Spring .....	100
	Remove and Rotate Mechanism .....	100
	Winding Machine .....	101

## List of Tables

Table 1-1: DB-815x Series .....	1
Table 3-1: CN3 Pin Description .....	16
Table 3-2: CN4 Pin Description .....	17
Table 3-3: Pulse Output Signals .....	18
Table 3-4: Encoder Feedback Signals .....	20
Table 3-5: Connection to Open Collector Output .....	21
Table 3-6: Origin Signal .....	23
Table 3-7: End-Limit Signals SPEL and SMEL .....	24
Table 3-8: In-position Signal .....	25
Table 3-9: Alarm Signal .....	26
Table 3-10: Deviation Counter Clear Signal .....	27
Table 3-11: Compare Output Signal .....	29
Table 4-1: ECAM Motion Interrupt Source .....	48
Table 4-2: Error Interrupt Source .....	49
Table 6-1: Data Type Definitions .....	71



## List of Figures

Figure 1-1: Block Diagram of the DB-8152 .....	3
Figure 2-1: DB-8152 PCB Layout .....	8
Figure 2-2: DB-8152 Extension Bracket and Cable Layout .....	9
Figure 2-3: Board Configuration.....	10
Figure 3-1: CN3 and CN4 Connectors.....	15
Figure 4-1: Cam Rotation 1.....	31
Figure 4-2: Cam Rotation 2.....	31
Figure 4-3: Angle of Rotation vs. Motion.....	32
Figure 4-4: ECAM Control Mode.....	33
Figure 4-5: Slave Motion Control Mode .....	35
Figure 4-6: Table Move Mode.....	36



# 1 Introduction

In order to increase the available functions on the PCI-8154/58 4-axis/8-axis pulse train output control channel carrier board, ADLINK offers four daughter boards providing a variety of functions. These DB-815x daughter boards can only be used with the PCI-8154/58 carriers only. Additional functionality provided by the DB-815x series includes high-speed triggering and distributed I/O control. Daughter boards can be added based on application requirements. The four daughter boards in the DB-815x series are:

Model Name	Primary Function	Description
DB-8150	High Speed Trigger Out	<ul style="list-style-type: none"> <li>▶ High speed trigger pulse out up to 1 MHz</li> <li>▶ Simultaneous 8 differential trigger output</li> <li>▶ 2 channels encoder input (from external I/F or carrier. FIFO/Linear function compared trigger output</li> </ul>
DB-8151	HSL Master Controller	<ul style="list-style-type: none"> <li>▶ High Speed Link for distributed I/O</li> </ul>
DB-8152	ECAM Controller	<ul style="list-style-type: none"> <li>▶ 2 channel encoder input (master &amp; slave)</li> <li>▶ Master/Slave controller</li> <li>▶ Easy to implement electronic cam behavior</li> </ul>
DB-8153	Motionnet Master Controller	<ul style="list-style-type: none"> <li>▶ Serial single axis motion control Bus</li> <li>▶ Does not support I/O function</li> </ul>

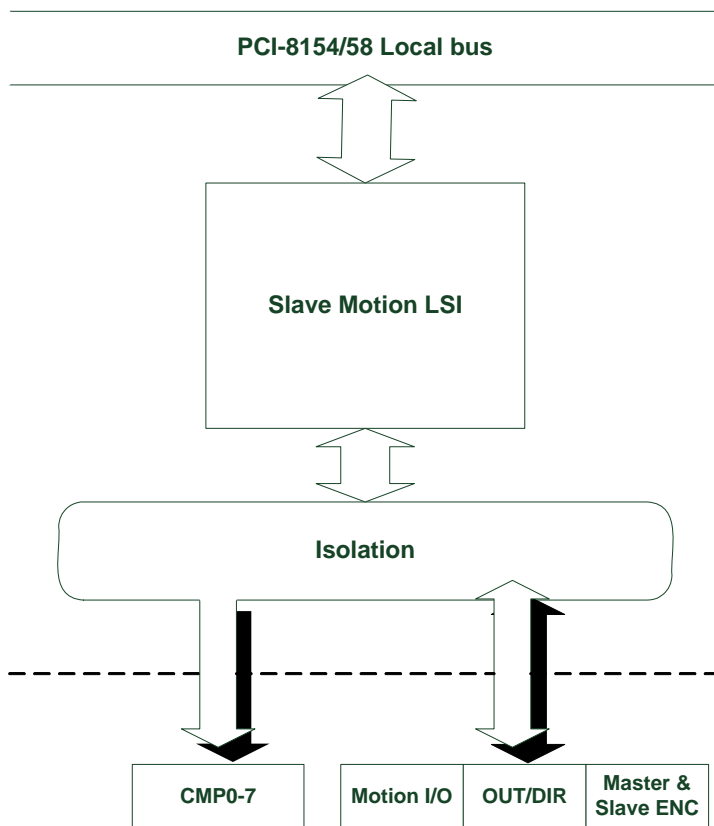
**Table 1-1: DB-815x Series**

The DB-8152 offers synchronization with the movement of the master axis, and this modular board performs the same operations as the motor on the master axis. This board can be used to replace a mechanical cam with a motor. Any axis controlled by the DB-8152 is referred to as a "slave axis." The DB-8152 reads the rotation angle of the master axis from encoder signals and drives the slave axis with the same pattern as is used by the master axis. The DB-8152 is preset with a certain feed amount for the slave axis and, from the angular information provided by the master axis, the slave axis will feed to an appropriate position.

Such slave motion solution is ideal for:

- ▶ Reducing the mechanisms in robotics and other automated manufacturing systems
- ▶ Utilizing mechanical actuators with associated controls for precise positioning or moving of a mechanical body
- ▶ Reducing kinetic and dynamic motion determination, as well as PC loading for more motion calculations
- ▶ Easy to implement ECAM application

The block diagram of the DB-8152 is as follows.



**Figure 1-1: Block Diagram of the DB-8152**

## 1.1 Specifications

### Interface

- ▶ For use with the PCI-8154 and PCI-8158 PCI cards only

### Master Controller

- ▶ Slave motion LSI
- ▶ 20 MHz external clock

**Interface**

- ▶ Slave motion I/O, single axis encoder input
- ▶ Master axis encoder input
- ▶ Compared signal output

**Connector**

- ▶ D-SUB 26P (Standard LPT port)
- ▶ D-SUB 9P (Standard COM port)

**Interrupt**

- ▶ Multi-event
- ▶ Status read back

**Dimensions**

- ▶ 96.42 (L) x 62 (W) mm

**Operating Temperature**

- ▶ 0 to 60°C

**Storage Temperature**

- ▶ -20 to 80°C

**Power Consumption**

- ▶ +3.3V @ 250 mA (typical)
- ▶ +5V @ 100 mA (typical)

## **1.2 Supported Software**

### **Program Library**

ADLINK provides a Windows WDM driver and DLL function library for the DB-8152. These function libraries are shipped with the board and supports Windows 2000/XP.





## 2 Installation

This chapter describes how to install the DB-8152. Please follow these steps below:

- ▶ Check what you have (Section 2.1, page 7)
- ▶ Check the PCB (Section 2.2, page 8)
- ▶ Install the hardware (Section 2.3, page 10)
- ▶ Install the software driver (Section 2.5, page 13)
- ▶ Understanding the I/O signal connections (Chapter 3, page 15) and their operations (Chapter 4, page 31)

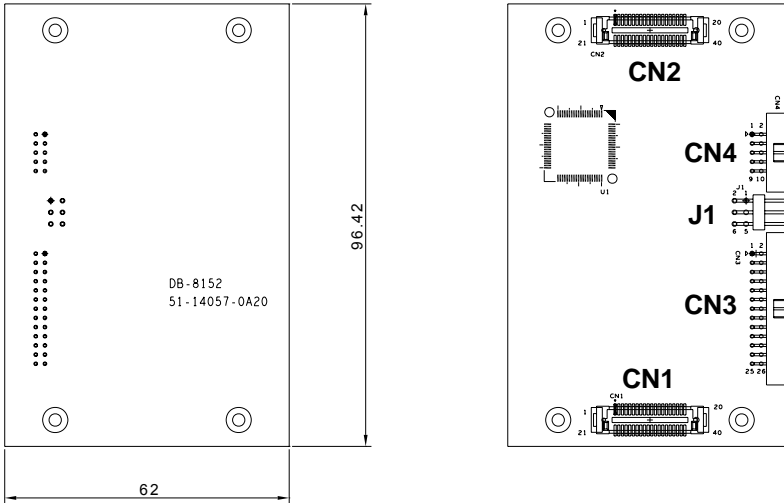
### 2.1 Package Contents

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

- ▶ DB-8152: Slave motion control board x1
- ▶ Copper Pillars x4
- ▶ Screws x8
- ▶ Extension I/F bracket with 25P and 9P D-SUB connectors x1
- ▶ 25-pin & 10-pin flat cable x1

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.

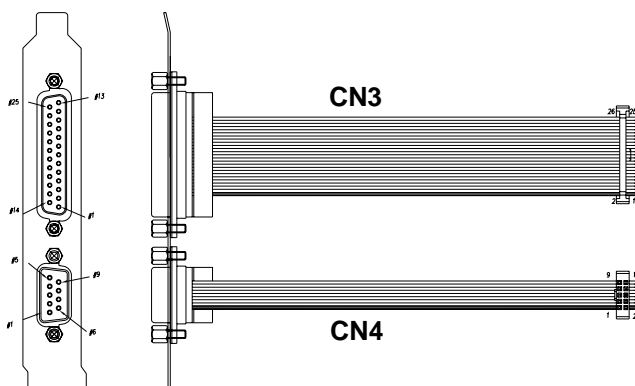
## 2.2 DB-8152 and External I/F Outline Drawing



**Figure 2-1: DB-8152 PCB Layout**

### **DB-8152:**

- ▶ CN1, CN2: Daughter board connectors for the PCI-8154/58
- ▶ CN3: Main slave signal connector
- ▶ CN4: Compare signals
- ▶ J1 Slave pulse output type selection (differential or single-end)



**Figure 2-2: DB-8152 Extension Bracket and Cable Layout**

**DB-8152-RJ45:**

- ▶ CN3: Main slave signal connector
- ▶ CN4: Compare signals

## 2.3 DB-8152 Hardware Installation

### 2.3.1 Hardware Configuration

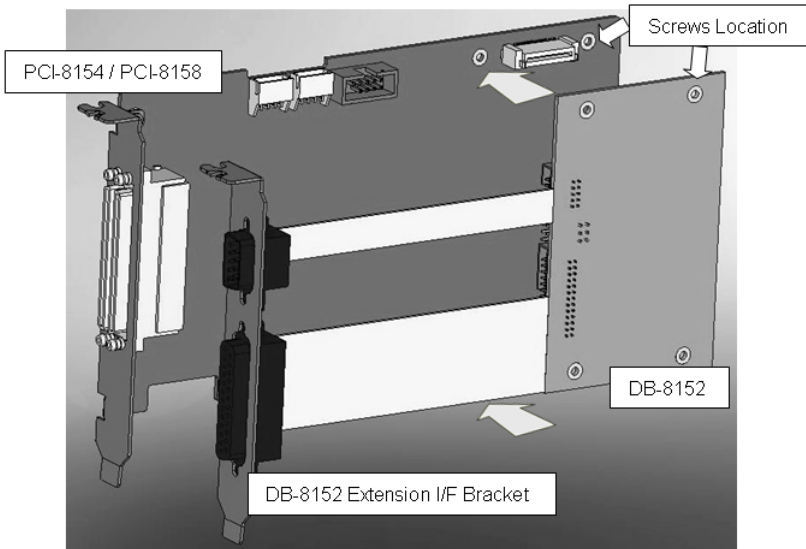
DB-8152 must be installed on to a PCI-8154 or PCI-8158 only. Please ensure correct orientation of the DB-8152 before attaching it to the PCI-8154 or PCI-8158.

### 2.3.2 Installation Procedures

Please follow installation procedure as follows.

#### Daughter Board Installation

1. Attach the DB-8152 on to the PCI-8154 or PCI-8158. Please ensure correct orientation of the DB-8152 daughter board.
2. Screw the eight screws into the corresponding copper pillars.
3. Connect the the extension bracket to the 26P and 10P flat cables.



**Figure 2-3: Board Configuration**

## **Carrier Board Installation**

4. Turn off the computer. Turn off all accessories (printer, modem, monitor, etc.) connected to the computer. Remove the cover from the computer.
5. Select one available 32-bit PCI expansion slot with a neighboring open slot for the DB-8152-RJ45. PCI slots are shorter than ISA or EISA slots and are usually white or ivory.
6. Before handling the PCI-8154 or PCI-8158 with DB-8152 and DB-8152-RJ45, 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.
7. Plug the PCI-8154/58 with DB series vertically down into the PCI slot, and then secure the PCI-8154/58 bracket and DB-8152-RJ45 bracket onto rear panel.

## **2.4 Troubleshooting:**

If the system doesn't boot or if any erratic behavior of the PCI board is experienced, it is most likely caused by an interrupt conflict (possibly an incorrect ISA setup). 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.

## 2.5 Software Driver Installation

Execute the following steps:

1. Auto-run the ADLINK All-In-One CD.
2. Follow the procedures of the installation wizard.
3. After setup installation has completed, reboot the system.





## 3 Signal Connections

Signal connections of all I/O's are described in this chapter. Refer to the contents of this chapter before wiring any cable between the DB-8152 and any motor driver.

### 3.1 CN3 and CN4 Connectors Pin Definition

As a slave motion for DB-8152, master encoder input pulses and salve output pulses are needed, these pins are placed in CN3. Also, some IO pins, like EL, ORG, INP, ALM, ERC are available in CN3. In addition to the functions provided by CN3, DB-8152 also has 8 channel of compare output on CN4. Refer to Section 3.8 for compare output description. CN3 and CN4 outline are illustrated as following figure.

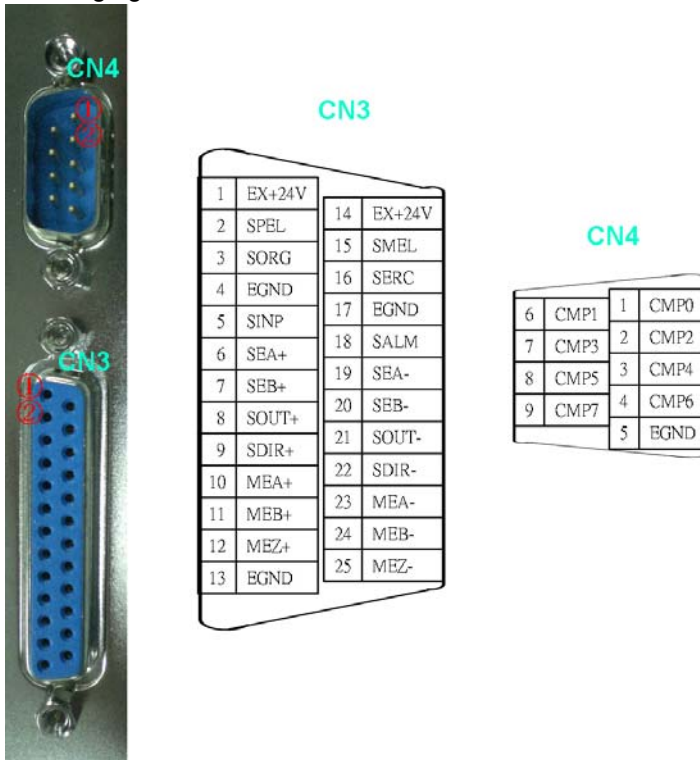


Figure 3-1: CN3 and CN4 Connectors

### 3.1.1 CN3 Pin Description:

CN3 Pin No.	I/O	Signal Name	Description
1	-	EX+24V	External +24V Power
2	-	SPEL	Slave Plus End Limit
3	I	SORG	Slave Origin
4	-	EGND	External Ground
5	I	SINP	Slave In Position
6	I	SEA+	Slave Encoder A(+)
7	I	SEB+	Slave Encoder B(+)
8	O	SOUT+	Slave output pulse OUT(+)
9	O	SDIR+	Slave output pulse DIR(+)
10	I	MEA+	Master Encoder A(+)
11	I	MEB+	Master Encoder B(+)
12	I	MEZ+	Master Encoder Z(+)
13	-	EGND	External Ground
14	-	EX+24V	External +24V Power
15	I	SMEL	Slave Minus End Limit
16	O	SERC	Slave Error Counter Clear
17	-	EGND	External Ground
18	I	SALM	Slave Alarm
19	I	SEA-	Slave Encoder A(-)
20	I	SEB-	Slave Encoder B(-)
21	O	SOUT-	Slave output pulse OUT(-)
22	O	SDIR-	Slave output pulse DIR(-)
23	I	MEA-	Master Encoder A(-)
24	I	MEB-	Master Encoder B(-)
25	I	MEZ-	Master Encoder Z(-)

**Table 3-1: CN3 Pin Description**

### 3.1.2 CN4 Pin Description:

CN4 Pin No.	I/O	Signal Name	Description
1	O	CMP0	Compare Output 0
2	O	CMP2	Compare Output 2
3	O	CMP4	Compare Output 4
4	O	CMP6	Compare Output 6
5	-	EGND	External Ground
6	O	CMP1	Compare Output 1
7	O	CMP3	Compare Output 3
8	O	CMP5	Compare Output 5
9	O	CMP7	Compare Output 7

**Table 3-2: CN4 Pin Description**

## 3.2 Pulse Output Signals SOUT and SDIR

There is 1 axis slave motion pulse output signal on the DB-8152. Two pairs of SOUT and SDIR differential signals are used to transmit the pulse train and indicate the direction. The SOUT and SDIR signals can also be programmed as CW and CCW signal pairs. Refer to Chapter 4 for details of the logical characteristics of the SOUT and SDIR signals. In this section, the electrical characteristics of the SOUT and SDIR signals are detailed. Each signal consists of a pair of differential signals. The following table shows all pulse output signals on CN3.

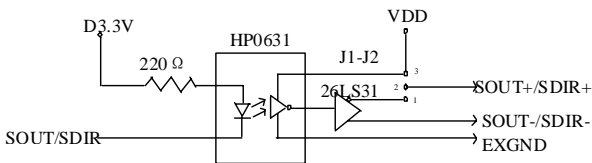
CN3 Pin No.	Signal Name	Description
8	SOUT+	Slave Pulse signals (+)
21	SOUT-	Slave Pulse signals (-)
9	SDIR+	Slave Direction signal (+)
22	SDIR-	Slave Direction signal (-)

**Table 3-3: Pulse Output Signals**

The default setting of SOUT and SDIR is set to differential line driver mode.

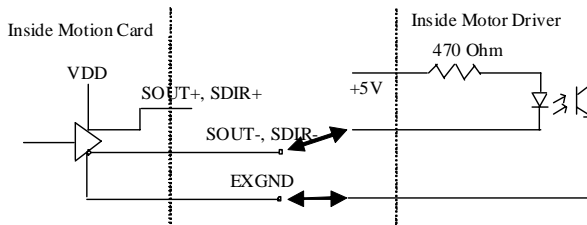
The following wiring diagram is for SOUT and SDIR signals of axis.

DB-8152:



**NOTE:** If the pulse output is set to open collector output mode, SOUT- and SDIR- are used to transmit SOUT and SDIR signals. The sink current must not exceed 20 mA on the SOUT- and SDIR- pins. The default setting is 1-2 shorted on J1 and J2.

Suggested Usage: Jumper 2-3 shorted and connect SOUT-/SDIR- to a 470 ohm pulse input interface's COM of driver. See the following figure. Choose SOUT-/SDIR- to connect to driver's OUT/DIR.



**Warning: The sink current must not exceed 20mA or the 26LS31 will be damaged!**

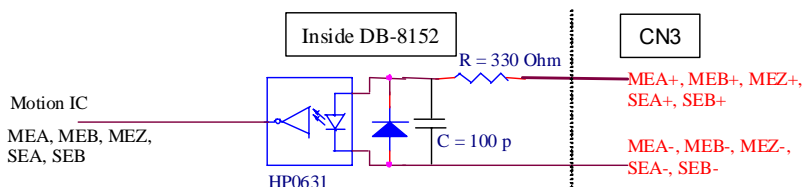
### 3.3 Encoder Feedback Signals MEA, MEB, MEZ, SEA and SEB

The encoder feedback signals include MEA, MEB, MEZ, SEA and SEB. Every signal comes with differential pairs. MEA and MEB are used for master position counting, and MEZ is used for master zero position indexing while SEA and SEB are used for slave position counting. Its relative signal names and pin numbers are shown in the following tables:

CN3 Pin No	Signal Name	CN3 Pin No	Signal Name
10	MEA+	23	MEA-
11	MEB+	24	MEB-
12	MEZ+	25	MEZ-
6	SEA+	19	SEA-
7	SEB+	20	SEB-

**Table 3-4: Encoder Feedback Signals**

The input circuit of the encoder signals is as follows:

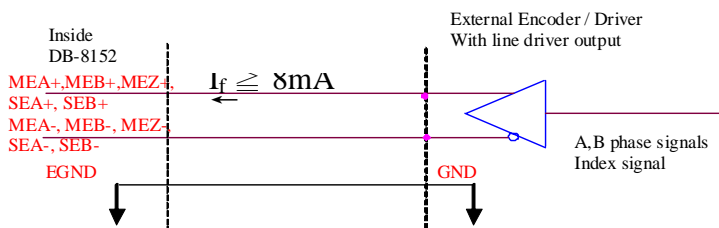


Please note that the voltage across each differential pair of encoder input signals should be at least 3.5 V. Therefore, the output current must be observed when connecting to the encoder feedback or motor driver feedback as not to over drive the source. The differential signal pairs are converted to digital signals MEA, MEB, MEZ, SEA and SEB; then feed to the motion control ASIC.

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

## Connection to Line Driver Output

To drive the DB-8182 encoder input, the driver output must provide at least 3.5V across the differential pairs with at least 8mA driving capacity. The grounds of both sides must be tied together. The maximum frequency of the master encoder is 1Mhz.

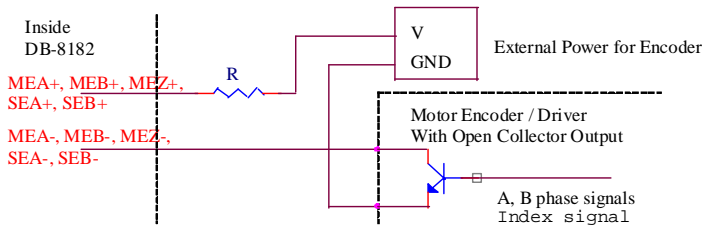


## Connection to Open Collector Output

To connect with an open collector output, an external power supply is necessary. Some motor drivers can provide the power source. The connection between the DB-8182, the encoder, and the power supply is shown in the diagram below. Note that an external current limiting resistor R is necessary to protect the DB-8182 input circuit. The following table lists the suggested resistor values according to the encoder power supply.

Encoder Power (V)	External Resistor R
+5V	0Ω(None)
+12V	1.5kΩ
+24V	3.0kΩ

**Table 3-5: Connection to Open Collector Output**



For more operation information on the encoder feedback signals, refer to Chapter 4.



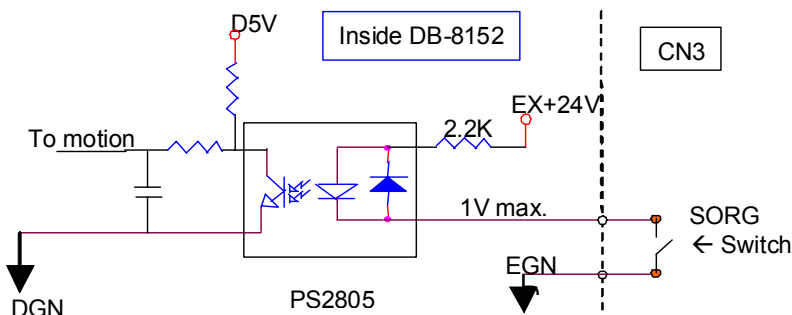
### 3.4 Origin Signal SORG

The slave origin signal (SORG) is used as input signal for the origin of the mechanism. The following table lists signal name, pin number:

CN3 Pin No	Signal Name
3	SORG

**Table 3-6: Origin Signal**

The input circuit of the SORG signals is shown below. Usually, a limit switch is used to indicate the origin on slave motion. The specifications of the limit switch should have contact capacity of +24 V @ 10 mA minimum. An internal filter circuit is used to filter out any high frequency spikes, which may cause errors in the operation.



When the motion controller is operated in the home return mode, the SORG signal is used to inhibit the control output signals (SOUT and SDIR).

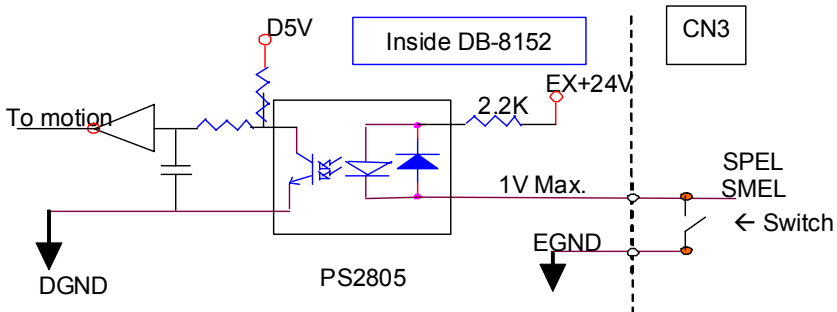
### 3.5 End-Limit Signals SPEL and SMEL

There are two end-limit signals SPEL and SMEL for slave motion. SPEL indicates the end limit signal is in the plus direction and SMEL indicates the end limit signal is in the minus direction. The signal names, pin numbers are shown in the table below:

CN3 Pin No	Signal Name
2	SPEL
15	SMEL

**Table 3-7: End-Limit Signals SPEL and SMEL**

A circuit diagram is shown in the diagram below. The external limit switch should have a contact capacity of +24 V @ 10 mA minimum. Either 'A-type' (normal open) contact or 'B-type' (normal closed) contact switches can be used.



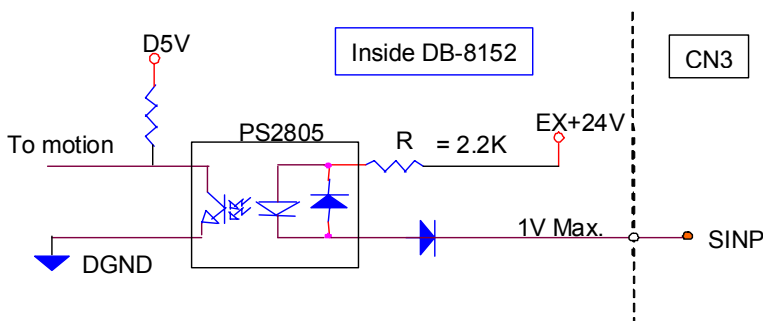
### 3.6 In-Position Signal SINP

The in-position signal SINP from a servo motor driver indicates its deviation error. If there is no deviation error then the servo's position indicates zero. The signal name and pin number is shown in the table below:

CN3 Pin No	Signal Name
5	SINP

**Table 3-8: In-position Signal**

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



The in-position signal is usually generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 8mA current sink capabilities to drive the INP signal.

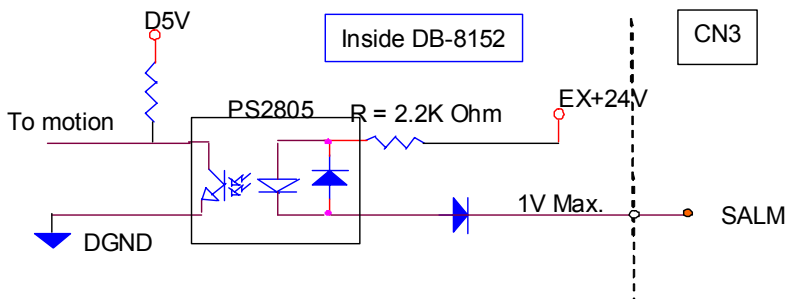
### 3.7 Alarm Signal SALM

The alarm signal SALM is used to indicate the alarm status from the servo driver. The signal name and pin number is shown in the table below:

CN3 Pin No	Signal Name
18	SALM

**Table 3-9: Alarm Signal**

The input alarm circuit is shown below. The SALM signal usually is generated by the servomotor driver and is ordinarily an open collector output signal. An external circuit must provide at least 8mA current sink capabilities to drive the SALM signal.



### 3.8 Deviation Counter Clear Signal SERC

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

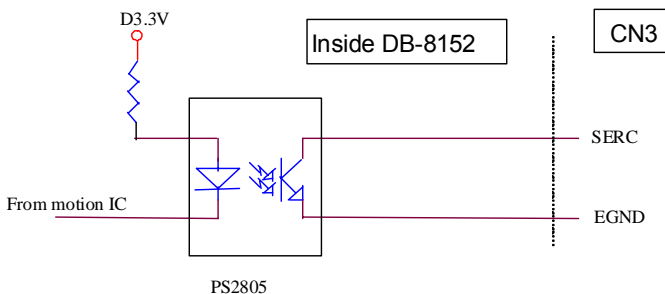
1. Home return is complete
2. End-limit switch is active
3. An alarm signal stops SOUT and SDIR signals
4. An emergency stop command is issued by software (operator)

The signal name and pin number is shown in the table below:

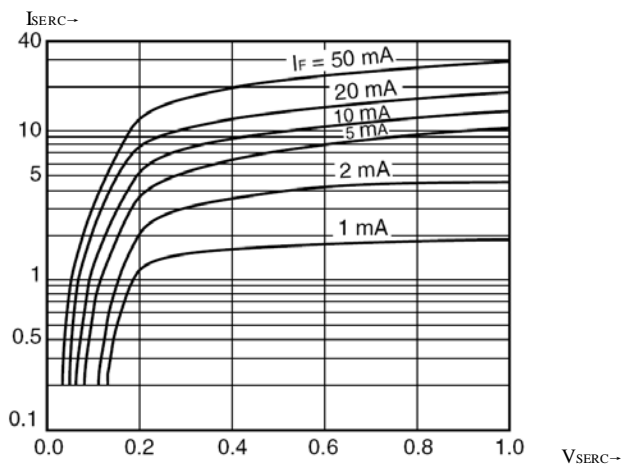
CN3 Pin No	Signal Name
16	SERC

**Table 3-10: Deviation Counter Clear Signal**

The SERC signal is used to clear the deviation counter of the servomotor driver. The SERC output circuit is an open collector with a maximum of 0.2V at 4mA sinking capacity. Below figure shows more detail electrical characteristics on SERC output.



IF current is around **10mA**



### 3.9 Compare Output Signal: SCMP 0-7

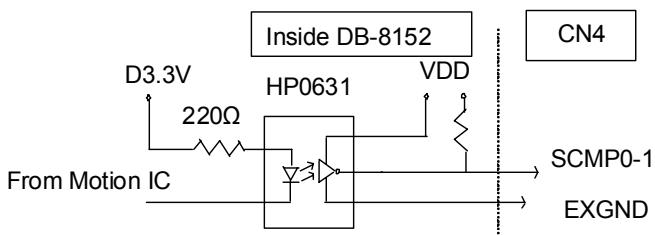
The DB-8152 provides 8 channel compare output: SCMP0 to SCMP7. The first two channels provide high speed compare outputs while the other six channels provide general speed compare output.

The slave compare output is located on CN4. The signal names and pin numbers are shown below:

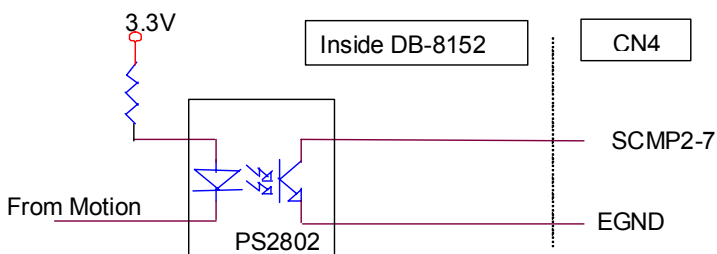
CN4 Pin No	Signal Name
1	SCMP0
2	SCMP1
3	SCMP2
4	SCMP3
6	SCMP4
7	SCMP5
8	SCMP6
9	SCMP7

**Table 3-11: Compare Output Signal**

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



The following wiring diagram is the CMP on the other 6 channels:

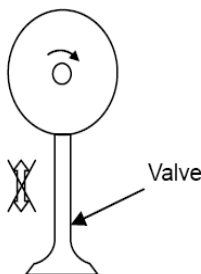




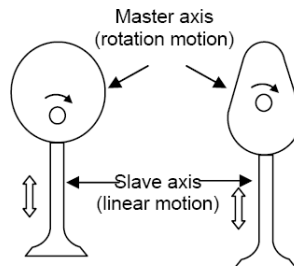
## 4 Operation Theory

Cams are frequently used in several machines. Here is a simple example: picture cams that move in a fashion to open and close the valves in an automobile engine. The rotation of the cams is linked to the up/down motion of a piston in the cylinder. The valves move up/down following the rotation of the cam, so that the intake/outlet valves open and close.

If a cam is simply a round piece of metal with its center of rotation in the center of the cam, as shown in Figure 4-1, a valve in contact with the cam will not move at all. In order to move the valve up and down, the center of rotation have to be off center of the round piece of metal, or the cam must have an eccentric shape, such as shown in Figure 4-2. Therefore, it is easy to understand that the distance moved and the speed of movement depends precisely on the cam shape and center of rotation. That is direct relationship between the angle of rotation and cam's eccentricity (movement on the master axis) to the valve's linear motion and speed (movement on the slave axis).

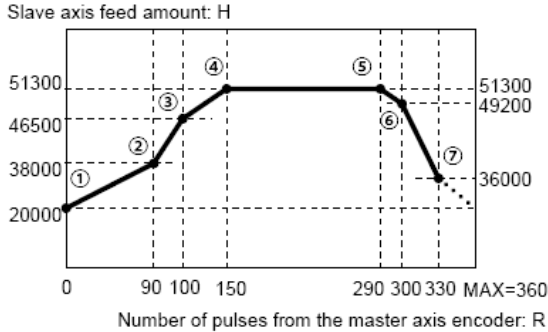


**Figure 4-1: Cam Rotation 1**



**Figure 4-2: Cam Rotation 2**

Figure 4-3 is a graph of the relationship between the cam's angle of rotation and the valve's motion. The shape of the curve on the graph depends on cam's shape. The horizontal axis tracks the master axis rotation and the vertical axis is the slave axis.



**Figure 4-3: Angle of Rotation vs. Motion**

In general, a cam mechanism is used to convert the rotation of motor on the master axis into linear motion on the slave axis. Using two motors (one is master axis and another is slave axis), the cam's operation can be controlled electrically. This is called as "electronic cam control".

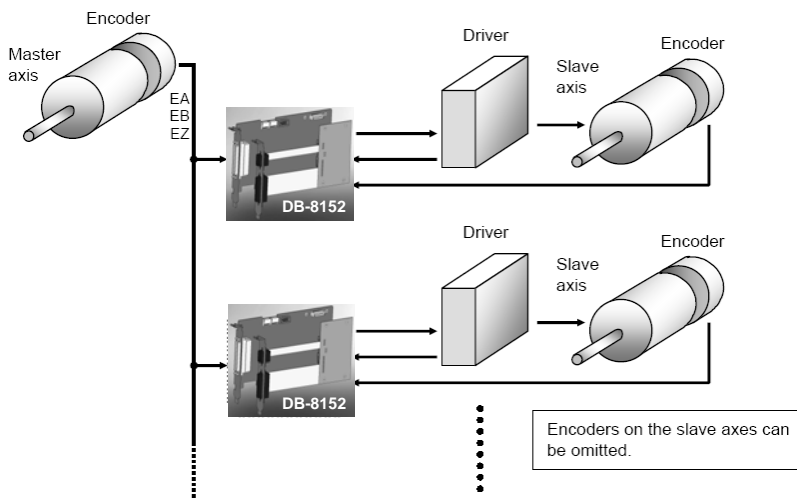
Through ECAM control using, the CPU loading will be greatly reduced while still maintaining precise motion performance.

## 4.1 Motion Control Modes

The DB-8152 provides two modes to control the electronic cam operation (which referred to master axis encoder angle counter to control slave axis simultaneously) or direct control of the slave axis. They are called “ECAM mode” and “Slave motion mode”, respectively.

### 4.1.1 ECAM Control Mode

To synchronize multiple slave axes with one master axis, simply connect them as shown in Figure 4-4. There is no limit on the number of DB-8152 boards that can be connected. The ECAM mode is able to subdivide into Normal mode, SRST mode and EZ mode.



**Figure 4-4: ECAM Control Mode**

### **Normal Mode Under ECAM Mode**

In a Normal mode application, the ECAM function will be performed according to the master axis encoder feedback angle data. The DB-8152 outputs pulses to control a motor driver on the slave axis while it obtains master axis encoder information in angular units. Like a general-purpose cam mechanism, as the master axis rotates through one turn, the slave axis will move and return to its starting position. Of course, as the master axis speed changes, the slave axis will change its speed, accordingly. In addition, if the master axis rotates backward, the slave axis will reverse its motion.

### **SRST Mode Under ECAM Mode**

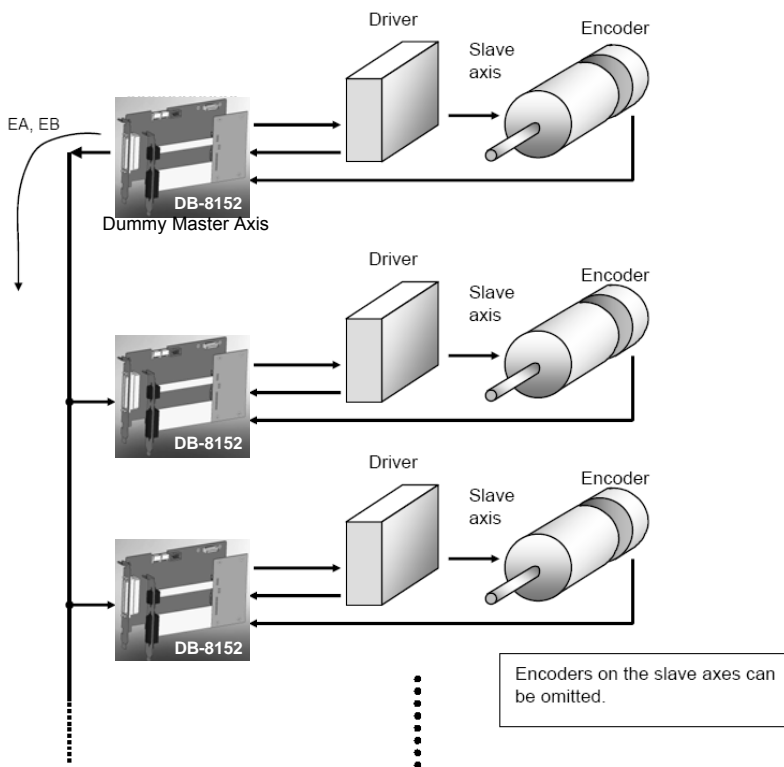
In addition to Normal mode, the DB-8152 also offers SRST mode which is contained within Normal mode. The theory of SRST mode means the slave axis is controlled along master encoder feedback, but the slave axis will be able to move from specified point which user has defined initially. In SRST mode application, the ECAM function will be performed and begin movement at a specified (starting) master encoder position. When this mode is selected, after a start signal is specified the DB-8152 will postpone the slave axis start until the master axis reaches the specified position. Enter the position for the master axis via the `_8154_db52_set_start_position` function. When a stop command is issued, the slave axis will stop when the master axis reaches the specified position.

### **EZ Motion Control Mode**

Besides the two operation modes mentioned above, the DB-8152 also offers an EZ mode also contained within the Normal mode. The theory of EZ mode that means the slave axis was controlled following the EZ signal to begin slave movement. When user uses this mode, after a start signal is specified the DB-8152 will postpone the slave axis start until the master axis EZ signal is turned ON. When a stop command is issued, the slave axis must be stopped with a master axis EZ pulse. Please take notice of EZ logic setting before the EZ mode is performed.

## Slave Motion Control Mode

When using multiple DB-8152 boards, select one DB-8152 to put into the dummy master axis mode. The other DB-8152 can be synchronized with this dummy master axis. The EA/EB terminals on the DB-8152 in dummy master axis mode will be used as output terminals. They will output 90° phase difference signals. They can be synchronized by connecting these signals to the EA/EB terminals of the other DB-8152 which be put into slave axis.



**Figure 4-5: Slave Motion Control Mode**

## 4.2 General Motion Control Mode

### 4.2.1 Relative Position Move

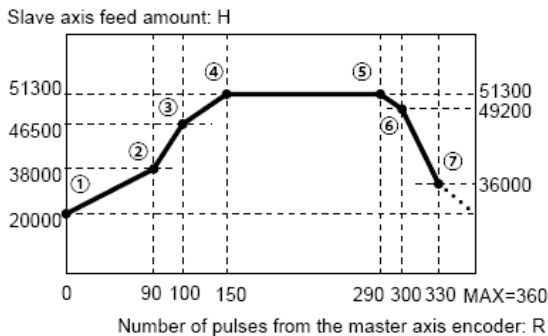
In the coordinate system, one command can be used to locate the target position. Relative command means that the motion controller can be provided a distance, then the motion controller will move motor by the distance from current position. During the movement, the speed profile can be specified, which means how fast and at what speed to reach the position can be defined.

### 4.2.2 Velocity Move 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. This function is to accelerate an axis to the specified constant velocity. The axis will continue to travel at a constant velocity until the axis is commanded to stop. The direction is determined by the sign of the velocity parameter.

### 4.2.3 Table Move Mode

Table move mode means the slave pulse output command, which was specified in DB-8152, according to master axis encoder feedback. The relationship between master axis angle and slave pulse command can thus be defined.



**Figure 4-6: Table Move Mode**

The DB-8152 contains RAM to store this data (the number of pulses in one master axis turn and the feed amount on the slave axis relative to the rotation angle of the master axis). Figure 4-6: shows 7 points (1 to 7) as a simple example. However, the DB-8152 supports up to 128 points of data per rotation of the master axis. A mechanical cam can only impose one pattern on a slave axis since the cam's shape is fixed. Moreover, the DB-8152 can specify any pattern stored in RAM, so the possibilities for using it are much greater.

#### **4.2.4 Home Return Mode**

Home return means searching a zero position point on the coordinate. Sometimes, the ORG pin is used as a zero position on the coordinate. When the machine powers on, the program needs to find a zero point of this machine. The DB-8152 motion controller provides a home return mode to establish the zero point.

This mode will cause the axis to perform a home return move. The direction of movement is determined by the sign of velocity parameter. Please note to handle conditions when the limit switch is touched or other conditions that are can possibly cause the axis to stop.

## 4.3 Motor Driver Interface

Several dedicated I/Os are provided which can be connected to a motor driver directly for custom functions. Motor drivers provide many kinds of I/O pins for external motion controllers to use. We have classified them to two groups:

- ▶ Pulse I/O signals including pulse command and encoder interface used for master axis and slave axis
- ▶ Digital I/O signals including error clear, alarm, INP, alarm reset, end-limit, zero position and emergency stop inputs.

The following sections describe the functions of these I/O pins.

### 4.3.1 Pulse Command Output Interface

The motion controller uses pulse command to control servo/stepper motors via motor drivers. Please set the drivers to position mode which can accept pulse trains as position command. The pulse command consists of two signal pairs. It is defined as OUT and DIR pins on 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 the same as motor driver. The modes vs. signal type of OUT and DIR pins are listed in the table below:

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



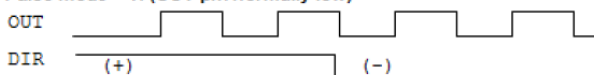
## Single Pulse Output Mode (OUT/DIR Mode)

In this mode, the OUT pin is for outputting 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)**



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



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



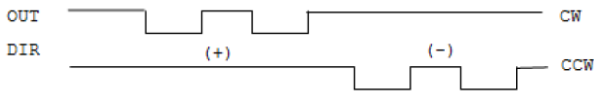
**Pulse mode = 3: (OUT pin normally low)**



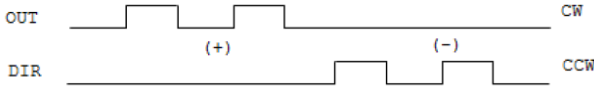
## Dual Pulse Output Mode (CW/CCW Mode)

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

Pulse outmode = 4: (Pulse is normally high)



Pulse outmode = 5: (Pulse is normally low)



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

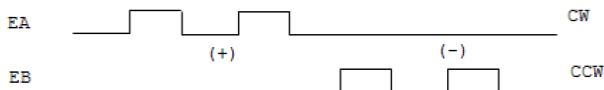
### 4.3.2 Pulse Feedback Input Interface

Our motion controller provides one 28-bit up/down counter of each axis for pulse feedback counting. This counter is called position counter. The position counter counts pulses from the EA and EB signal which have plus and minus pins on 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. Multiply by 4 AB phase 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 encoder for motion controller, the feedback source

for this counter must be set as pulse command output or the counter value will always be zero. If it is set as pulse command output, users can get the position counter value from pulse command output counter because the feedback pulses are internal counted from command output pulses. The following diagrams show these two types of pulse feedback signal.

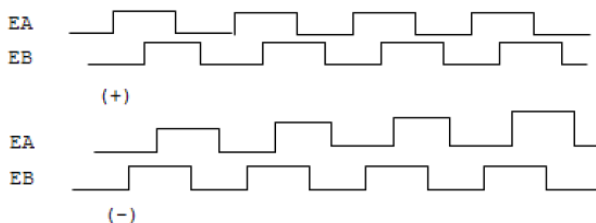
### Plus and Minus Pulses Input Mode (CW/CCW Mode)



The pattern of pulses in this mode is the same as the **Dual Pulse Output Mode** in the Pulse Command Output section except that the input pins are EA and EB. In this mode, pulses from EA pin cause the counter to count up, whereas EB pin caused the counter to count down.

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

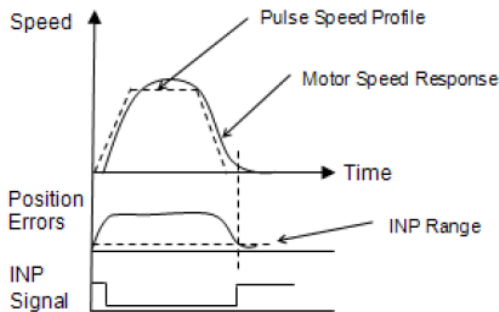
In this mode, the EA signal is a 90° phase leading or lagging in comparison with the EB signal. “Lead” or “lag” of phase difference between two signals is caused by the turning direction of the motor. The up/down counters counts up when the phase of EA signal leads the phase of EB signal. The following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



The index input (EZ) signal is as the zero reference in linear or rotary encoder. The EZ can be used to define the mechanical zero position of the mechanism. The logic of signal must also be set correctly to get correct result.

### 4.3.3 In Position Signal

The in-position signal is an output signal from motor driver. It tells motion controllers a motor has been reached a position within a predefined error. The predefined error value is in-position value. Most motor drivers call it as INP value. After motion controller issues a positioning command, the motion busy status will keep true until the INP signal is ON. Users can disable INP check for motion busy flag. If it is disabled, the motion busy will be FALSE when the pulses command is all sent.



### 4.3.4 Servo Alarm Signal

The alarm signal is an output signal from motor driver. It tells motion controller that there has something error inside servo motor or driver. Once the motion controller receives this signal, the pulses command will stop sending and the status of ALM signal will be ON. The reasons of alarm could be servo motor's over speed, over current, over loaded and so on. Please check the manual of the motor driver for details. The logic of alarm signal must be set correctly. If the alarm logic's setting is not the same as motor driver's setting, the ALM status will be always 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 motor driver to clear the error counter. The error counter is counted from the difference of command pulses and feedback pulses. The feedback position will always have a delay from the command position. It results in pulse differences between these two positions at any moment. The differences are shown in error counter. Motor driver uses the error counter as a basic control index. The larger the error counter value is, the faster the motor speed command will be set. If the error counter is zero, it means that the zero motor speed command will be set. At following four situations, the ERC signal will be outputted automatically from motion controller to motor driver in order to clear 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

## **4.4 Mechanical switch interface**

Dedicated input pins are provided for mechanical switches (such as used for an original switch (ORG), plus and minus end-limit switch ( $\pm$ EL) and emergency stop input (EMG). The response time of these switches is very short; only a few ASIC clock ticks. There is no real-time problem when using these signals. All functions are done by the motion ASIC. The software does not have to perform anything and only needs to wait the results.

### **4.4.1 Original or Home Signal**

Our controller provides one original or home signal for each axis. This signal is used for defining zero position of this axis. The logic of this signal must be set properly before doing home procedure.

### **4.4.2 End-limit Switch Signal**

The end-limit switches are usually installed on both ending sides of one axis. The plus EL signal must be installed at the positive position of the axis and minus EL at the negative position of the axis. These two signals are for safety. If they are installed incorrectly, they will not provide protection. Once a moving part of the motor touches one of the end-limit signals, the motion controller will stop sending pulses and output an ERC signal. This can prevent machine crashes with missed operations.

### **4.4.3 Emergency Stop Input**

Our motion controller provides a global digital input for emergency situation. Once the input is turned on, our motion controller will stop all axes' motion immediately to prevent machine's damage. Usually, an emergency stop button will be connected to this input of the machine. We suggest this input is used as normal closed type for safety.

## 4.5 The Counters

There are four counters for each axis of this motion controller.

They are described in this section.

- ▶ Slave Command position counter: counts the number of output pulses
- ▶ Master / Slave Feedback position counter: counts the number of input pulses
- ▶ Remaining position counters: counts the error between command and feedback pulse numbers.

### 4.5.1 Slave 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 provides the information of the current command position. It is useful for debugging a motion system. In DB-8152, this counter is only used for a slave (dummy master) axis.

Our motion system is an open loop type. The motor driver receives pulses from motion controller and drive the motor to move. When the driver is not moving, we can check this command counter and see if there is an update value on it. If it is, it means that the pulses have been sent and the problem could be on the motor driver. Try to check motor driver's pulse receiving counter when this situation is happened.

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 for slave axis. On the other hand, the feedback position counter is amount to 360 for master axis due to the master axis receives zero to 359 degree data ONLY.

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 source of command position for an option when no external encoder inputs.

Because our motion controller is not a closed-loop type, the feedback position counter is just for reference after motion is moving.

The position closed-loop is done by servo motor driver. If the servo driver is well tuned and the mechanical parts are well assembled, the total position error will remain in acceptable range after motion command is finished.

## 4.5.3 Remaining Position Counters

To execute a positioning operation, the motion ASIC will copy the number of feed pulses from the feed amount, which is issued for a positioning feed command, to this counter and then counts down that number of pulses.

When this counter reaches zero, the DB-8152 will stop operation. If you intentionally stop the slave axis before this counter reaches zero and restart the positioning operation, the slave axis will start operating again based on the value in command counter. (Since this counter is overwritten by the value in command counter, the previous operation pulse count will be deleted.)

Refer to this counter only during a positioning operation. During other operations, readings from here will be meaningless.



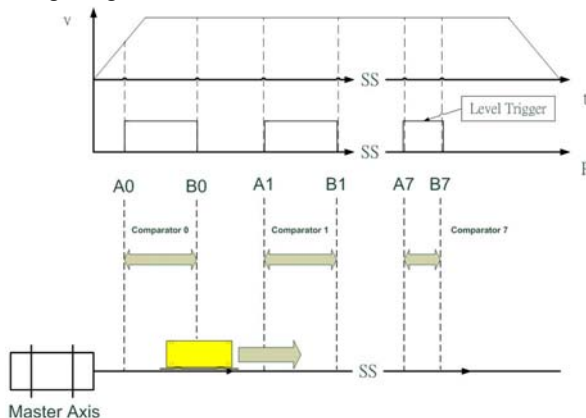
## 4.6 The Comparators

There is one general counter comparator of master/dummy master axis. This comparator has dedicated functions:

- ▶ There are eight axis comparators: 0-7
- ▶ Comparison Interval of each comparator: comparison start point A & comparison destination point B
- ▶ Comparison Mode: comparing method
- ▶ Comparison signal active logic

### 4.6.1 General Comparators

The general comparator is used to select which comparator to compare in a specified axis. The compare methods could be selected from equal, greater than or less than with directional or directionless. The action when the condition is met can also be selected from: generate interrupt, stop motion, or others. The comparator triggers within a dedicated interval which is user-defined (the interval between Ax and Bx). Before a trigger occurs, the comparing start point, comparing destination point and comparing method must be decided. The comparison action is described in the following diagram.



### 4.6.2 Position Latch

After the EZ latch signal arrives, this function is used to read the latched value of counters.

## 4.7 Interrupt Control

The motion controller can generate an interrupt signal to the host PC. It is very useful for event-driven software application. This `_8154_db52_int_control` function can be used to enable if disable the interrupt service.

There are two kinds of interrupt sources on the DB-8152:

- ▶ ECAM motion interrupt source
- ▶ Error interrupt source

The ECAM motion interrupt sources can be masked; however error interrupt sources cannot. The ECAM motion interrupt sources can be masked with `_8154_db52_set_ecam_int_factor`. Its mask bits are shown in Table 4-1:

`ecam_int_status`: can be masked by function call `_8154_db52_set_ecam_int_factor`

Bit	Description
0	When the master axis EZ signal turns ON (edge detection)
1	(Reserved) (Always set to 0)
2	When the conditions for comparators 0A and 0B are met.
3	When the conditions for comparators 1A and 1B are met.
4	When the conditions for comparators 2A and 2B are met.
5	When the conditions for comparators 3A and 3B are met.
6	When the conditions for comparators 4A and 4B are met.
7	When the conditions for comparators 5A and 5B are met.
8	When the conditions for comparators 6A and 6B are met.
9	When the conditions for comparators 7A and 7B are met.
10	When EZ is ON, Master axis position does not indicate the zero position
11	When EZ is ON, Master axis position will not reach the value in SRSR
12	When the slave axis is start
13	When the slave axis is stop(including error stop)
14	When the master axis position reach the value in SRSR

**Table 4-1: ECAM Motion Interrupt Source**

The error interrupt sources are not masked but the error number of the situation can be obtained from the `_8154_db52_wait_error_interrupt` return code if it has not timed out.

`error_int_status`: cannot be masked

Bit	Interrupt Factor
0	ALM happen and stop
1	Emergency on and stop
2	+End Limit on and stop
3	-End Limit on and stop
4	(Reserved)
5	Master axis angle data illegal
6	Master axis angle data overflow
7	Slave axis feed amount overflow and stop
8	Pulser output overflow and stop
9	Slave encoder input signal error
10	Master encoder input signal error
11	A feed command error

**Table 4-2: Error Interrupt Source**



## 5 MotionCreatorPro

After installing the hardware (Chapters 2 and 3), it is necessary to correctly configure all cards and double check the system before running. This chapter gives guidelines for establishing a control system and manually testing the 8154/8158 cards to verify correct operation. The MotionCreatorPro software provides a simple yet powerful means to setup, configure, test, and debug a motion control system that uses 8154/8158 cards.

Note that MotionCreatorPro is only available for Windows 2000/XP with a screen resolution higher than 1024x768. Recommended screen resolution is 1024x768. It cannot be executed under the DOS environment.

### 5.1 Execute MotionCreatorPro

After installing the software drivers for the 8154 in Windows 2000/XP, the MotionCreatorPro program can be located at <chosen path>\PCI-Motion\MotionCreatorPro. To execute the program, double click on the executable file or use **Start>Program Files>PCI-Motion>MotionCreatorPro**.

## 5.2 About MotionCreatorPro

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

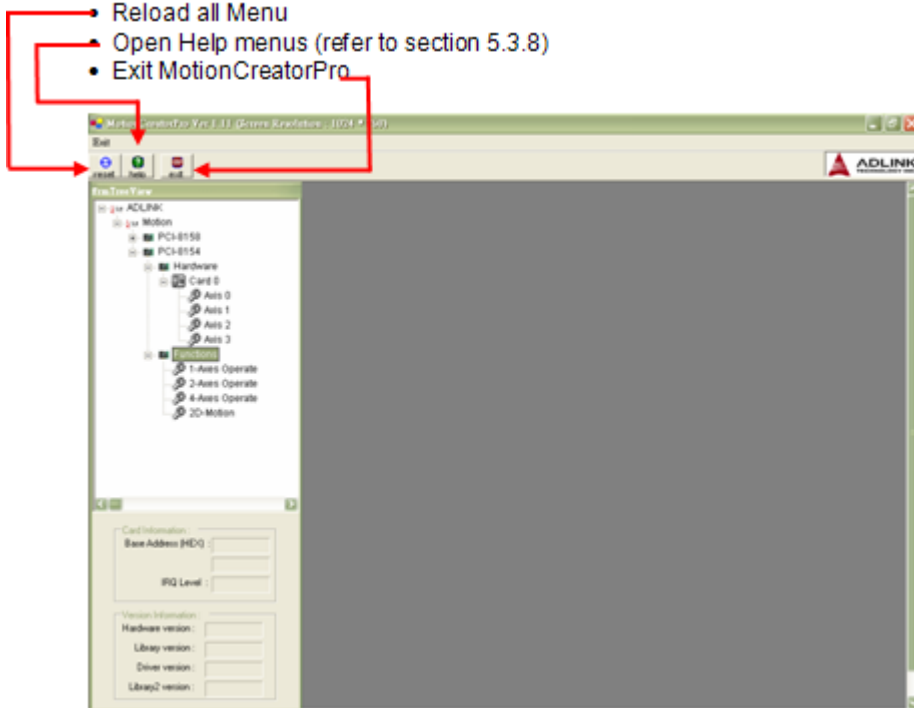
1. MotionCreatorPro is a program written in VB.NET 2003, and is available only for Windows 2000/XP with a screen resolution higher than 1024x768. It cannot be run under DOS.
2. MotionCreatorPro allows users to save settings and configurations for 8154/8158 cards. Saved configurations will be automatically loaded the next time MotionCreatorPro is executed. Two files, **8154.ini** (or **8158.ini**) and **8154MC.ini** (or **8158MC.ini**), in the **windows root directory** are used to save all settings and configurations.
3. To duplicate configurations from one system to another, copy **8154.ini** (or **8158.ini**) and **8154MC.ini** (or **8158MC.ini**) into the windows root directory.
4. If multiple 8154 cards use the same MotionCreatorPro saved configuration files, the DLL function call **\_8154\_config\_from\_file()/\_8158\_config\_from\_file()** can be invoked within a user developed program. This function is available in a DOS environment as well.

## 5.3 MotionCreatorPro Introduction

### 5.3.1 Main Menu

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

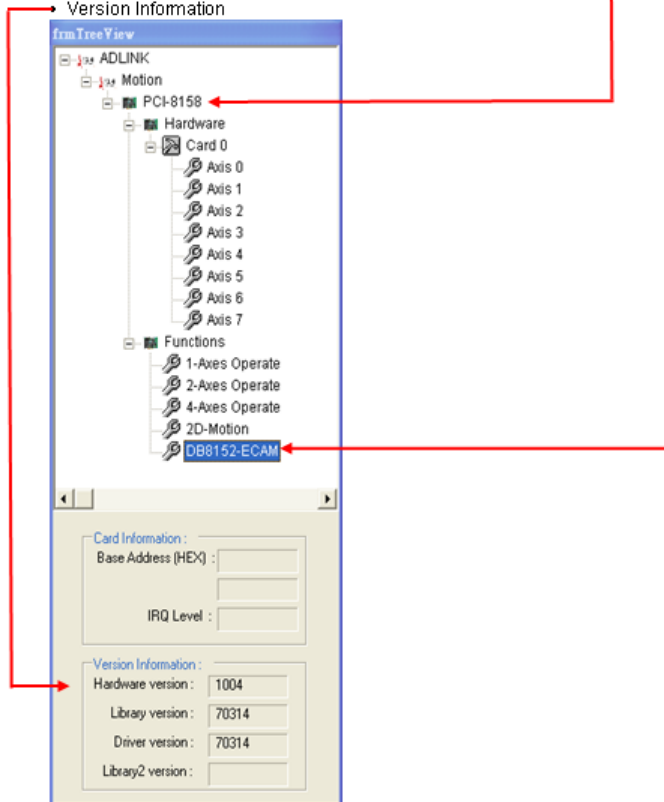
- Reload all Menu
- Open Help menus (refer to section 5.3.8)
- Exit MotionCreatorPro



## 5.3.2 Select Menu

The select menu appears after running MotionCreatorPro. It is used to:

- Select operating card and axis
- Open Card Information Menu (refer to section 5.4.3)
- Open DB8152-ECAM Menu (refer to section 5.4.4)
- Version Information





### 5.3.3 Card Information Menu

In this menu, it shows some Information about this card:



### 5.3.4 DB-8152 ECAM Menu

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

**ECAM Operation**  
**PCI-8154 DB8152**

**Card no : Card 0**

☒ PCI-8154 Pulse Continuous

**Run Mode**  
 ECAM Mode **1** Slave Motion Mode

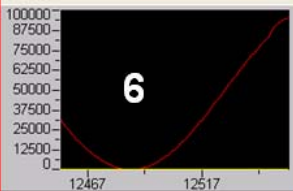
**ECAM Mode**  
 Normal Mode SRST Mode **2** EZ Mode

**Slave Motion Mode**  
 Cont. Move Relative Mode **3** Home. Move

**I/O Status**  
 RDY  
 ALM  
 EMG  
 ERC  
 ORG  
 EZ  
 DIR  
 9 RON  
 +EL  
 -EL  
 INP  
 SSCM  
 VMV  
 VORG  
 VOTP

**Position Status**  
 Master Pos. 67  
 Slave Pos. **4**  
 Slave Comm. 05019

**Set Position**  
 Master Pos. 0  
 Slave Pos. **5**  
 Slave Comm. 50000

**Operate**  
 Position Profile  
 Slave Comm: Slave Pos:  

**6**

**INT Status**  
 Timeout 5000  
☒ INT 1 12  
☒ INT 2 13  
☐ INT 3 2  
☐ INT 4 3  
☒ ERROR

**Play key**  
 Erc On Low **10** Hight  
 Forward Move Backward Move **11**

**Next Card**  
 S **12** onfig  
 Save Config

1. **Run Mode:** Select run mode.
  - ▷ **ECAM Mode:** Normal Mode, SRST Mode or EZ Move.
  - ▷ **Slave Motion Mode:** Cont. Move, Relative Mode or Home Mode.
2. **ECAM Mode:** Select ECAM mode.
  - ▷ **Normal Mode:** Set ECAM start/stop with Normal Mode. The related functions are `_8154_db52_start_ecam (CardId,0)`, `_8154_db52_stop_ecam(CardId,0)`.
  - ▷ **SRST Mode:** Set ECAM start/stop with SRST Mode. “Start Point” will be used as SRST start point. ECAM will start/stop at SRST start point. The related function are `_8154_db52_set_start_position`, `_8154_db52_start_ecam (CardId,1)`, `_8154_db52_stop_ecam(CardId,1)`.
  - ▷ **EZ Move:** Set ECAM start/stop with EZ Mode. ECAM will start/stop after waiting for the EZ signal. The related function are `_8154_db52_start_ecam (CardId,2)`, `_8154_db52_stop_ecam(CardId,2)`.
  - ▷ **Start Point:** To set SRST start point. The related function is `_8154_db52_set_start_position`.
3. **Slave Motion Mode:** Select Slave Motion mode.
  - ▷ **Cont. Move:** Velocity motion mode. The related function is `_8154_db52_slave_v_move()`.
  - ▷ **Relative Mode:** “Distance” will be used as relative displacement for motion. The related function is `_8154_db52_slave_r_move()`.
  - ▷ **Home Mode:** Home return motion. The related function is `_8154_db52_slave_home_move()`.
  - ▷ **Distance:** Set the relative distance for “Relative Mode.”
  - ▷ **Vel:** velocity in units of pulse per second.

#### 4. **Position Status:**

- ▷ **Master Pos:** displays the value of the master feedback position counter. The related function is `_8154_db52_get_master_position()`.
- ▷ **Slave Pos:** displays the value of the slave feedback position counter. The related function is `_8154_db52_get_slave_position()`
- ▷ **Slave Comm:** displays the value of the command counter. The related function is `_8154_db52_get_command()`.

#### 5. **Set Position:** clicking this button will set positioning counter to a specified value. The related functions are:

- ▷ `_8154_db52_set_master_position()`
- ▷ `_8154_db52_set_slave_position()`
- ▷ `_8154_db52_set_command()`

#### 6. **Position\_Profile:** show the Slave Comm and Slave Pos Position Profile.

#### 7. **Buttons:**

- ▷ **Set ECAM Tab:** Open ECAM Table setup Menu (refer to section 5.4.6).
- ▷ **Set Comp data:** Open Comparator Table setup Menu (refer to section 5.4.7).

#### 8. **INT Status:**

- ▷ **Int Start:** Enable the Windows interrupt event. The related function are `B_8154_db52_set_ecam_int_factor()`, `B_8154_db52_int_control()`.
- ▷ **Int Stop:** Disable the Windows interrupt event. The related function are

B\_8154\_db52\_set\_ecam\_int\_factor(),B\_8154\_db52\_int\_control().

- ▷ **Timeout:** Specifies the time-out interval.
- ▷ **Factor bit:** Set int\_factor bit.
- ▷ **INT1-INT4:** display of ECAM INT status. The related function is \_8154\_db52\_wait\_ecam\_interrupt().
- ▷ **Error INT:** display of Error INT status. The related function is \_8154\_db52\_wait\_error\_interrupt().

9. **I/O Status:** The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is \_8154\_db52\_get\_io\_status().

10. **Erc On:** Set the ERC signal output status. The related function is \_8154\_db52\_set\_erc().

#### 11. Play Key:

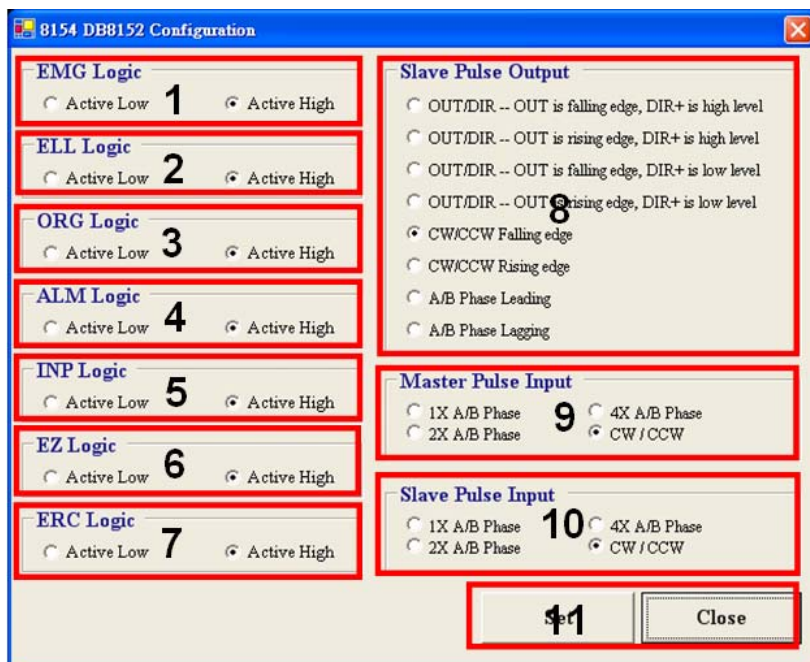
- ▶ **Left play button:** Clicking this button will cause the DB8152 start to outlet pulses according to previous setting.
  - ▷ In “Relative Mode,” it causes the axis to move forward.
  - ▷ In “Cont. Move,” it causes the axis to start to move according to the velocity setting.
- ▶ **Right play button:** Clicking this button will cause the DB8152 start to outlet pulses according to previous setting.
  - ▷ In “Relative Mode,” it causes the axis to move backwards.
  - ▷ In “Cont. Move,” it causes the axis to start to move according to the velocity setting, but in the opposite direction.
- ▶ **Stop Button:** Clicking this button will cause the DB8152 to stop. The related function are \_8154\_db52\_stop\_ecam(), \_8154\_db52\_slave\_stop().

#### 12. Buttons:

- ▷ **Next Card:** Change operating card.
- ▷ **Setup Config:** Open Configuration Menu (See 5.3.5).
- ▷ **Save Config:** Save current configuration to 8154.ini and 8154MC.ini.

### 5.3.5 Configuration Menu

In this menu, users can configure EMG Logic, ELL Logic, ORG Logic, ALM Logic, INP Logic, EZ Logic, ERC Logic, Slave Pulse Output, Master Pulse Input and Slave Pulse Input.



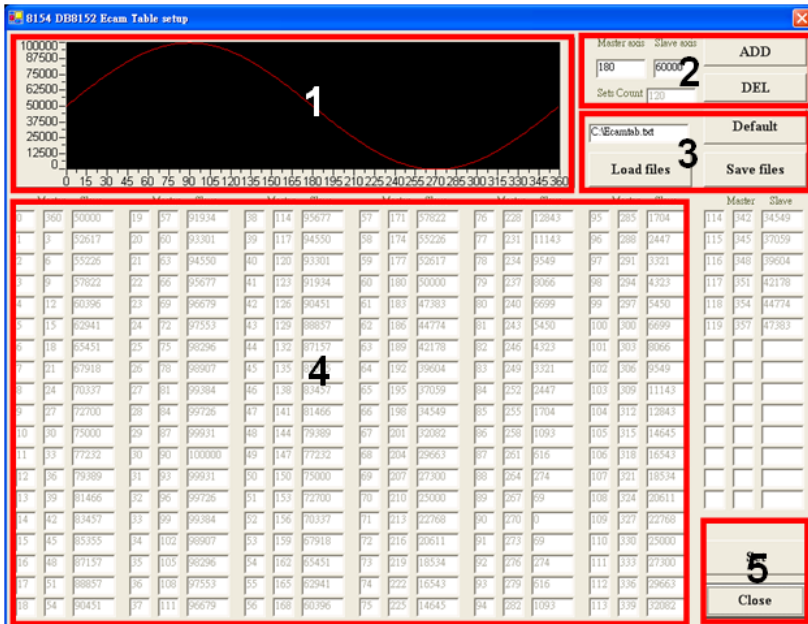
**8154 DB8152 Configuration**

<b>EMG Logic</b> <input type="radio"/> Active Low <b>1</b> <input checked="" type="radio"/> Active High	<b>Slave Pulse Output</b> <input type="radio"/> OUT/DIR -- OUT is falling edge, DIR+ is high level <input type="radio"/> OUT/DIR -- OUT is rising edge, DIR+ is high level <input type="radio"/> OUT/DIR -- OUT is falling edge, DIR+ is low level <input type="radio"/> OUT/DIR -- OUT is rising edge, DIR+ is low level <input checked="" type="radio"/> CW/CCW Falling edge <input type="radio"/> CW/CCW Rising edge <input type="radio"/> A/B Phase Leading <input type="radio"/> A/B Phase Lagging
<b>ELL Logic</b> <input type="radio"/> Active Low <b>2</b> <input checked="" type="radio"/> Active High	
<b>ORG Logic</b> <input type="radio"/> Active Low <b>3</b> <input checked="" type="radio"/> Active High	
<b>ALM Logic</b> <input type="radio"/> Active Low <b>4</b> <input checked="" type="radio"/> Active High	
<b>INP Logic</b> <input type="radio"/> Active Low <b>5</b> <input checked="" type="radio"/> Active High	
<b>EZ Logic</b> <input type="radio"/> Active Low <b>6</b> <input checked="" type="radio"/> Active High	
<b>ERC Logic</b> <input type="radio"/> Active Low <b>7</b> <input checked="" type="radio"/> Active High	<b>Master Pulse Input</b> <input type="radio"/> 1X A/B Phase <b>9</b> <input type="radio"/> 4X A/B Phase <input type="radio"/> 2X A/B Phase <input checked="" type="radio"/> CW / CCW
	<b>Slave Pulse Input</b> <input type="radio"/> 1X A/B Phase <b>10</b> <input type="radio"/> 4X A/B Phase <input type="radio"/> 2X A/B Phase <input checked="" type="radio"/> CW / CCW
<div> <input checked="" type="radio"/> Set           <input type="button" value="Close"/> </div>	

1. **EMG Logic:** Select the logic of the EMG signal. The related function call is `_8154_db52_set_EMG_logic()`.
2. **ELL Logic:** Select the logic of the ELL signal. The related function call is `_8154_db52_set_ELL_logic()`.
3. **ORG Logic:** Select the logic of the ORG signal. The related function call is `_8154_db52_set_ORG_logic()`.
4. **ALM Logic:** Select the logic of the ALM signal. The related function call is `_8154_db52_set_ALM_logic()`.
5. **INP Logic:** Select the logic of the INP signal. The related function call is `_8154_db52_set_INP_logic()`.
6. **EZ Logic:** Select the logic of the EZ signal. The related function call is `_8154_db52_set_EZ_logic()`.
7. **ERC Logic:** Select the logic of the ERC signal. The related function call is `_8154_db52_set_ERC_logic()`.
8. **Slave Pulse Output Mode:** Select the output mode of the slave pulse signal (OUT/ DIR). The related function call is `_8154_db52_set_slave_pls_outmode()`.
9. **Master Pulse Input:** Sets the configurations of the Master Pulse input signal(EA/EB). The related function calls are `_8154_db52_set_master_pls_ipmode()`.
10. **Slave Pulse Input:** Sets the configurations of the Slave Pulse input signal(EA/EB). The related function calls are `_8154_db52_set_slave_pls_ipmode()`.
11. **Buttons:**
  - ▷ **Set:** Set EMG Logic, ELL Logic, ORG Logic, ALM Logic, INP Logic, EZ Logic, ERC Logic, Slave Pulse Output, Master Pulse Input and Slave Pulse Input to memory.  
  
Note: This button must be clicked after selection is made.
  - ▷ **Close:** Click this button close this window..

## 5.3.6 ECAM Table setup Menu

In this menu, users can setup the ECAM Table.



**0154 DBB152 Ecram Table setup**

Master addr: 180 Slave addr: 60000 **2** ADD  
 Sets Count: 120 DEL

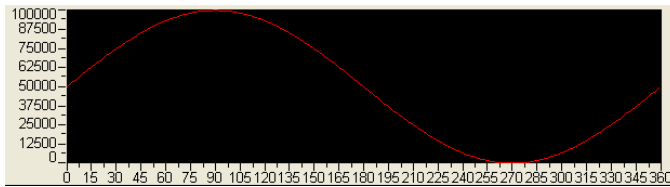
C:\Ecram\b152 **3** Default  
 Load files Save files

	Master	Slave	Master	Slave	Master	Slave	Master	Slave	Master	Slave	Master	Slave	Master	Slave	Master	Slave	Master	Slave		
1	360	50000	19	57	91934	38	114	95677	57	171	57822	76	228	12843	95	285	1704	114	342	34549
2	3	52617	20	60	93301	39	117	94350	58	174	55226	77	231	11143	96	288	2447	115	345	37039
3	6	55226	21	63	94550	40	120	93301	59	177	52617	78	234	9549	97	291	3321	116	348	39604
4	9	57822	22	66	95677	41	123	91934	60	180	50000	79	237	3066	98	294	4323	117	351	42178
5	12	50396	23	69	96679	42	126	90451	61	183	47383	80	240	5699	99	297	5450	118	354	44774
6	15	52941	24	72	97553	43	129	88657	62	186	44774	81	243	5450	100	300	5699	119	357	47383
7	18	55451	25	75	98296	44	132	87157	63	189	42178	82	246	4323	101	303	5066			
8	21	57918	26	78	98907	45	135		64	192	39604	83	249	3321	102	306	3549			
9	24	60337	27	81	99384	46	138	83457	65	195	37039	84	252	2447	103	309	11143			
10	27	62700	28	84	99726	47	141	81466	66	198	34549	85	255	1704	104	312	12843			
11	30	65000	29	87	99931	48	144	79389	67	201	32082	86	258	1093	105	315	14645			
12	33	67232	30	90	100000	49	147	77232	68	204	29663	87	261	616	106	318	16543			
13	36	69389	31	93	99931	50	150	75000	69	207	27300	88	264	274	107	321	18534			
14	39	71466	32	96	99726	51	153	72700	70	210	25000	89	267	69	108	324	20611			
15	42	73457	33	99	99384	52	156	70337	71	213	22768	90	270	0	109	327	22768			
16	45	75355	34	102	98907	53	159	67918	72	216	20611	91	273	69	110	330	25000			
17	48	77157	35	105	98296	54	162	65451	73	219	18534	92	276	274	111	333	27300			
18	51	78857	36	108	97553	55	165	62941	74	222	16543	93	279	616	112	336	29663			
19	54	80451	37	111	96679	56	168	60396	75	225	14645	94	282	1093	113	339	32082			

**5** Close



## 1. ECAM Profile: Show the ECAM Profile.



## 2. Buttons:

- ▷ **Master axis:** Master axis data.  
Range: 1-360
- ▷ **Slave axis:** Slave axis data.
- ▷ **Sets Count:** Specified table sets number.  
Range: 1-128
- ▷ **ADD:** Add Slave axis data to ecam tables when Master axis data empty or edit Slave axis data when Master axis data exist.
- ▷ **DEL:** Delete ecam tables data when Master axis data exist.

Note: Set count must be larger then 0. When the Master axis = 360, it cannot be delete because it is both the starting and end points.

Master axis	Slave axis	ADD
180	60000	
Sets Count	120	DEL

### 3. Buttons:

- ▷ **Default:** Load default ECAM Table data.
- ▷ **Load files:** Load ECAM Table data from file.  
(i.e. "C:\Ecamtab.txt".)
- ▷ **Save files:** Save ECAM Table data to file.  
(i.e. "C:\Ecamtab.txt")



The interface shows a text box containing "C:\Ecamtab.txt", a "Default" button, a "Load files" button, and a "Save files" button.

### 4. ECAM Table data: Display the ECAM Table data.

	Master	Slave		Master	Slave		Master	Slave
0	360	50000	19	57	91934	38	114	95677
1	3	52617	20	60	93301	39	117	94550
2	6	55226	21	63	94550	40	120	93301
3	9	57822	22	66	95677	41	123	91934
4	12	60396	23	69	96679	42	126	90451
5	15	62941	24	72	97553	43	129	88857
6	18	65451	25	75	98296	44	132	87157
7	21	67918	26	78	98907	45	135	85355

### 5. Buttons:

- ▷ **Set:** Set ECAM Table data to memory. The related function call is `_8154_db52_build_ecam_table()`.

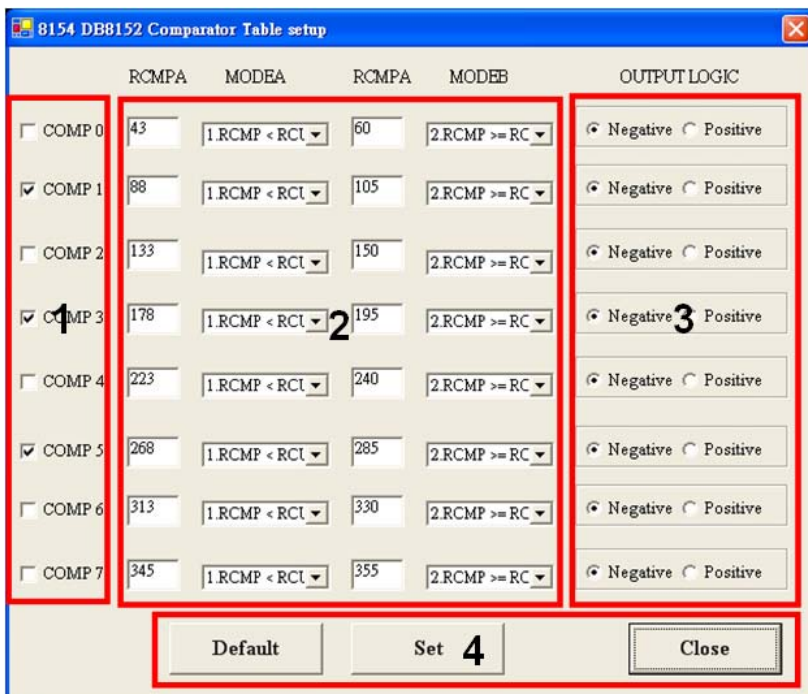
Note: Before using ECAM, this button must be clicked. If there is no problem with the data, the "RDY" bit in the `io_status` will become "1".

- ▷ **Close:** Click this button close this window.

Note: when used at such a high speed, the slave axis feed amount for one section is limited to between 1 and 6 pulses.

### 5.3.7 Comparator Table setup Menu

In this menu, users can setup Comparator Table.



COMP	RCMPA	MODEA	RCMPB	MODEB	OUTPUT LOGIC
<input type="checkbox"/> COMP 0	43	1 RCMP < RCI	60	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input checked="" type="checkbox"/> COMP 1	88	1 RCMP < RCI	105	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input type="checkbox"/> COMP 2	133	1 RCMP < RCI	150	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input checked="" type="checkbox"/> COMP 3	178	1 RCMP < RCI	195	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input type="checkbox"/> COMP 4	223	1 RCMP < RCI	240	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input checked="" type="checkbox"/> COMP 5	268	1 RCMP < RCI	285	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input type="checkbox"/> COMP 6	313	1 RCMP < RCI	330	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive
<input type="checkbox"/> COMP 7	345	1 RCMP < RCI	355	2 RCMP >= RC	<input checked="" type="radio"/> Negative <input type="radio"/> Positive

Default Set 4 Close

1. **COMP0-COMP7:** Enable/Disable comparator.
2. **Comp Data / Mode:** Select the Comp Data and Comp Mode of the comparator. The related function call are `_8154_db52_set_comparator_data()`, `_8154_db52_set_comparator_mode()`.
3. **Comp Logic:** Select the logic of the comparator. The related function call is `_8154_db52_set_comparator_do()`.



## 6 Function Library

This chapter describes the supporting software for the PCI-8154/8158 card. User can use these functions to develop programs in C, C++, or Visual Basic. If Delphi is used as the programming environment, it is necessary to transform the header files, 8154.h/8158.h manually.

## 6.1 List of Functions

### System, Section 6.3

Function Name	Description
_8154_db52_initial	Card initialization
_8154_db52_close	Card close
_8154_db52_emg_stop	Immediate stop

### Pulse Input/Output Configuration, Section 6.4

Function Name	Description
_8154_db52_set_slave_pls_outmode	Set pulse command output mode
_8154_db52_set_master_pls_iptmode	Set encoder input mode
_8154_db52_set_slave_iptmode	Set counter input source

### ECAM Mode Motion, Section 6.5

Function Name	Description
_8154_db52_get_start_position	get start position for SRST mode
_8154_db52_set_start_position	set start position for SRST mode
_8154_db52_start_ecam	Set ecam start mode and start
_8154_db52_stop_ecam	Set ecam stop mode and stop
_8154_db52_build_ecam_table	Set ecam table

### Slave Motion, Section 6.6

Function Name	Description
_8154_db52_slave_v_move	Begin a constant velocity move
_8154_db52_slave_r_move	Begin a relative move
_8154_db52_slave_home_move	Begin a home return action
_8154_db52_slave_stop	Stop a move

## Motion Interface I/O, Section 6.7

Function Name	Description
_8154_db52_set_EMG_logic	Set EMG logic
_8154_db52_get_EMG_logic	Get EMG logic
_8154_db52_set_ELL_logic	Set ELL logic
_8154_db52_get_ELL_logic	Get ELL logic
_8154_db52_set_ORG_logic	Set ORG logic
_8154_db52_get_ORG_logic	Get ORG logic
_8154_db52_set_ALM_logic	Set ALM logic
_8154_db52_get_ALM_logic	Get ALM logic
_8154_db52_set_INP_logic	Set INP logic
_8154_db52_get_INP_logic	Get INP logic
_8154_db52_set_EZ_logic	Set EZ logic
_8154_db52_get_EZ_logic	Get EZ logic
_8154_db52_set_ERC_logic	Set ERC logic
_8154_db52_get_ERC_logic	Get ERC logic
_8154_db52_set_erc	Set ERC operating mode
_8154_db52_get_io_status	Get all the motion I/O status of DB8152

## Interrupt Control, Section 6.8

Function Name	Description
_8154_db52_int_control	Enable/Disable INT service
_8154_db52_wait_error_interrupt	Wait error related interrupts
_8154_db52_wait_ecam_interrupt	Wait ecam related interrupts
_8154_db52_set_ecam_int_factor	Set INT factor

## Position Control and Counters, Section 6.9

Function Name	Description
_8154_db52_get_master_position	Get the value of the Master feedback position counter
_8154_db52_set_master_position	Set the Master feedback position counter
_8154_db52_get_slave_position	Get the value of the Slave feedback position counter
_8154_db52_set_slave_position	Set the feedback Slave position counter
_8154_db52_get_command	Get the value of the command position counter
_8154_db52_set_command	Set the command position counter
_8154_db52_get_res_command	Get remaining pulses until the end of motion
_8154_db52_set_res_command	Set remaining pulses until the end of motion

## Position Compare and Latch, Section 6.10

Function Name	Description
_8154_db52_get_EZ_latch_data	Get EZ latched counter data
_8154_db52_get_comparator_data	Get comparator data
_8154_db52_set_comparator_data	A general function for setting comparator data
_8154_db52_set_comparator_mode	A general function for setting comparator mode
_8154_db52_set_comparator_do	Set comparator logic



## 6.2 C/C++ Programming Library

This section details all the functions. The function prototypes and some common data types are declared in **pci\_8154.h**. We suggest you use these data types in your application programs. The following table shows the data type names and their range.

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

**Table 6-1: Data Type Definitions**

The functions of the 8154 software drivers use full-names to represent the functions real meaning. The naming convention rule is described as follows when using the PCI-8154 as the carrier. If the PCI-8158 is selected as the primary motion controller, all functions will carry the **\_8158\_db51\_** prefix. The **\_8154\_db51\_** prefix is used below for use with the PCI-8154.

In a 'C' programming environment:

**\_**{hardware\_model}\_{action\_name}. For example:  
**\_8154\_db52\_initial()**.

In order to recognize the difference between a C library and a VB library, a capital "B" is placed at the beginning of each function name e.g. **B\_8154\_db52\_initial()**.

## 6.3 System

### @ Name

`_8154_db52_initial` – Card Initialization

`_8154_db52_close` – Card Close

`_8154_db52_emg_stop` – Immediately stop

### @ Description

`_8154_db52_initial` :

This function is used to initialize an DB8152 card without assigning the hardware resources. All DB8152 cards must be initialized by this function before calling other functions

`_8154_db52_close` :

This function is used to close DB8152 card and release its resources, which should be called at the end of an application.

`_8154_db52_emg_stop` :

This function is used to immediately stop an axis.

### @ Syntax

#### C/C++(Windows 2000/XP)

```
I16 _8154_db52_initial(I16 CardId);  
I16 _8154_db52_close(I16 CardId);  
I16 _8154_db52_emg_stop(I16 CardId);
```

#### Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_initial Lib "8154.dll" Alias  
    "_8154_db52_initial" (ByVal CardId As  
        Integer) As Integer  
B_8154_db52_close Lib "8154.dll" Alias  
    "_8154_db52_close" (ByVal CardId As Integer)  
    As Integer  
B_8154_db52_emg_stop Lib "8154.dll" Alias  
    "_8154_db52_emg_stop" (ByVal CardId As  
        Integer) As Integer
```

## **@ Argument**

**CardId:** The 8154 card index number

## **@ Return Code**

ERR\_NoError

ERR\_CardIDMapToCardNo

## 6.4 Pulse Input/Output Configuration

### @ Name

`_8154_db52_set_slave_pls_outmode` – Set the configuration for pulse command output

`_8154_db52_set_master_pls_iptmode` – Set the configuration for Master feedback pulse input

`_8154_db52_set_slave_pls_iptmode` – Set the configuration for Slave feedback pulse input

### @ Description

`_8154_db52_set_slave_pls_outmode:`

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

`_8154_db52_set_master_pls_iptmode:`

Configure the input modes of Master feedback pulses. There are four types for feedback pulse input.

`_8154_db52_set_slave_pls_iptmode:`

Configure the input modes of Slave feedback pulses. There are four types for feedback pulse input.

### @ Syntax

#### C/C++(Windows 2000/XP)

```
I16 _8154_db52_set_master_pls_iptmode(I16 CardId,  
    I16 pls_InputMode);  
I16 _8154_db52_set_slave_pls_iptmode(I16 CardId,  
    I16 pls_InputMode);  
I16 _8154_db52_set_slave_pls_outmode(I16 CardId,  
    I16 pls_outmode);
```

## Visual Basic 6(Windows 2000/XP)

```

B_8154_db52_set_master_pls_iptmode Lib "8154.dll"
    Alias "_8154_db52_set_master_pls_iptmode"
        (ByVal CardId As Integer, ByVal
        pls_InputMode As Integer) As Integer
B_8154_db52_set_slave_pls_iptmode Lib "8154.dll"
    Alias "_8154_db52_set_slave_pls_iptmode"
        (ByVal CardId As Integer, ByVal
        pls_InputMode As Integer) As Integer
B_8154_db52_set_slave_pls_outmode Lib "8154.dll"
    Alias "_8154_db52_set_slave_pls_outmode"
        (ByVal CardId As Integer, ByVal pls_outmode
        As Integer) As Integer

```

### @ Argument

**CardId:** The 8154 card index number

**pls\_outmode:** Setting of command pulse output mode

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

**pls\_iptmode:** setting of encoder feedback pulse input mode

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

### @ Return Code

ERR\_NoError

ERR\_CardIDMapToCardNo

## 6.5 ECAM Mode Motion

### @ Name

`_8154_db52_get_start_position` – get start position for SRST mode

`_8154_db52_set_start_position` – set start position for SRST mode

`_8154_db52_start_ecam` – Set ecam start mode and start

`_8154_db52_stop_ecam` – Set ecam stop mode and stop

`_8154_db52_build_ecam_table` – Set ecam table

### @ Description

`_8154_db52_get_start_position`:

This function is to read the start position value in the master axis setting register.

`_8154_db52_set_start_position`:

This function is to set the start position value in the master axis setting register. It should be setted before `_8154_db52_start_ecam` or `_8154_db52_stop_ecam` in SRST mode.

`_8154_db52_start_ecam`:

This function is used to start ecam with Normal mode, EZ mode or SRST mode. Before calling this function, it is necessary to setup the ecam table by `_8154_db52_build_ecam_table`.

`_8154_db52_stop_ecam`:

This function is used to stop ecam with Normal mode, EZ mode or SRST mode.

`_8154_db52_build_ecam_table`:

This function is used to setup the ecam table. Note: when used at such a high speed, the slave axis feed amount for one section is limited to between 1 and 6 pulses.

## @ Syntax

### C/C++(Windows 2000/XP)

```
I16 _8154_db52_get_start_position(I16 CardId, I32
    *Master_Position);
I16 _8154_db52_set_start_position(I16 CardId, I32
    Master_Position);
I16 _8154_db52_start_ecam(I16 CardId, I16
    ecam_Type);
I16 _8154_db52_stop_ecam(I16 CardId, I16
    ecam_Type);
I16 _8154_db52_build_ecam_table(I16 CardId, I32
    *table_angle, I32 *table_position, I16
    table_size);
```

### Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_get_start_position Lib "8154.dll"
    Alias "_8154_db52_get_start_position"
    (ByVal CardId As Integer, Master_Position As
    Long) As Integer
B_8154_db52_set_start_position Lib "8154.dll"
    Alias "_8154_db52_set_start_position"
    (ByVal CardId As Integer, ByVal
    Master_Position As Long) As Integer
B_8154_db52_start_ecam Lib "8154.dll" Alias
    "_8154_db52_start_ecam" (ByVal CardId As
    Integer, ByVal ecam_Type As Integer) As
    Integer
B_8154_db52_stop_ecam Lib "8154.dll" Alias
    "_8154_db52_stop_ecam" (ByVal CardId As
    Integer, ByVal ecam_Type As Integer) As
    Integer
B_8154_db52_build_ecam_table Lib "8154.dll" Alias
    "_8154_db52_build_ecam_table" (ByVal CardId
    As Integer, table_angle As Long,
    table_position As Long, ByVal table_size As
    Integer) As Integer
```

## @ Argument

**CardId:** The 8154 card index number

**Master\_Position:** SRSR start position value in the master axis setting register.

- ▶ Range: 0-359

**ecam\_Type:** setup ecam mode

- ▶ **ecam\_Type** = 0 , Normal mode,
- ▶ **ecam\_Type** = 1 , EZ mode; start/stop when EZ signal
- ▶ **ecam\_Type** = 2 , SRST mode; start/stop when master axis position matches SRSR

**table\_angle:** Specified angle table

**table\_position:** Specified position table

**table\_size:** Specified table size

## @ Return Code

ERR\_NoError

ERR\_CardIDMapToCardNo

ERR\_PosOutOfRange



## 6.6 Slave Motion

### @ Name

\_8154\_db52\_slave\_v\_move – Begin a constant velocity move

\_8154\_db52\_slave\_r\_move – Begin a relative move

\_8154\_db52\_slave\_home\_move – Begin a home return action

\_8154\_db52\_slave\_stop – Stop a move

### @ Description

\_8154\_db52\_slave\_v\_move:

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

\_8154\_db52\_slave\_r\_move:

This function causes the axis to rotate at constant velocity (Vel), and decelerate to stop at the relative distance.

\_8154\_db52\_slave\_home\_move:

This function will cause the axis to perform a home return move. The direction of movement is determined by the sign of velocity parameter (Vel). Users should also take care to handle conditions when the limit switch is touched or other conditions that are possible causing the axis to stop.

\_8154\_db52\_slave\_stop:

This function is used to stop outputting pulses.

### @ Syntax

#### C/C++(Windows 2000/XP)

```
I16 _8154_db52_slave_v_move(I16 CardId, I32 Vel);  
I16 _8154_db52_slave_r_move(I16 CardId, I32 Dist,  
    I32 Vel);  
I16 _8154_db52_slave_home_move(I16 CardId, I32  
    Vel);  
I16 _8154_db52_slave_stop(I16 CardId);
```

## Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_slave_v_move Lib "8154.dll" Alias
    "_8154_db52_slave_v_move" (ByVal CardId As
        Integer, ByVal Vel As Long) As Integer
B_8154_db52_slave_r_move Lib "8154.dll" Alias
    "_8154_db52_slave_r_move" (ByVal CardId As
        Integer, ByVal Dist As Long, ByVal Vel As
        Long) As Integer
B_8154_db52_slave_home_move Lib "8154.dll" Alias
    "_8154_db52_slave_home_move" (ByVal CardId
        As Integer, ByVal Vel As Long) As Integer
B_8154_db52_slave_stop Lib "8154.dll" Alias
    "_8154_db52_slave_stop" (ByVal CardId As
        Integer) As Integer
```

### @ Argument

**CardId:** The 8154 card index number

**vel:** velocity in units of pulse per second

**Dist:** Specified relative distance to move (unit: pulse)

### @ Return Code

```
ERR_NoError
ERR_CardIDMapToCardNo
ERR_PosOutOfRange
```

## 6.7 Motion Interface I/O

### @ Name

`_8154_db52_set_EMG_logic` – Set EMG logic  
`_8154_db52_get_EMG_logic` – Get EMG logic  
`_8154_db52_set_ELL_logic` – Set ELL logic  
`_8154_db52_get_ELL_logic` – Get ELL logic  
`_8154_db52_set_ORG_logic` – Set ORG logic  
`_8154_db52_get_ORG_logic` – Get ORG logic  
`_8154_db52_set_ALM_logic` – Set ALM logic  
`_8154_db52_get_ALM_logic` – Get ALM logic  
`_8154_db52_set_INP_logic` – Set INP logic  
`_8154_db52_get_INP_logic` – Get INP logic  
`_8154_db52_set_EZ_logic` – Set EZ logic  
`_8154_db52_get_EZ_logic` – Get EZ logic  
`_8154_db52_set_ERC_logic` – Set ERC logic  
`_8154_db52_get_ERC_logic` – Get ERC logic  
`_8154_db52_set_erc` – Set ERC operating mode  
`_8154_db52_get_io_status` – Get all the motion I/O status of DB8152

## @ Description

\_8154\_db52\_set\_EMG\_logic:

Set the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_get\_EMG\_logic:

Get the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_set\_ELL\_logic:

Set the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_get\_ELL\_logic:

Get the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_set\_ORG\_logic:

Set the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_get\_ORG\_logic:

Get the ELL logic, Negative logic or Positive logic.

\_8154\_db52\_set\_ALM\_logic:

Set the active logic of the ALARM signal input from the servo driver.

\_8154\_db52\_get\_ALM\_logic:

Get the active logic of the ALARM signal input from the servo driver.

\_8154\_db52\_set\_INP\_logic:

Set the active logic of the In-Position signal input from the servo driver. Users can select whether they want to enable this function. It is disabled by default.

\_8154\_db52\_get\_INP\_logic:

Get the active logic of the In-Position signal input from the servo driver. Users can select whether they want to enable this function.

\_8154\_db52\_set\_EZ\_logic:

You can set the logic of the EZ with this function. Negative logic or Positive logic.

#### \_8154\_db52\_get\_EZ\_logic:

You can get the logic of the EZ with this function. Negative logic or Positive logic.

#### \_8154\_db52\_set\_ERC\_logic:

You can set the logic of the ERC with this function. Negative logic or Positive logic.

#### \_8154\_db52\_get\_ERC\_logic:

You can get the logic of the ERC with this function. Negative logic or Positive logic.

#### \_8154\_db52\_set\_erc:

You can set the ERC signal output status with this function. High or Low.

#### \_8154\_db52\_get\_io\_status:

Get all the I/O statuses for each card. The definition for each bit is as follows:

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	EMG	EMG status
3	ERC	ERC pin output
4	ORG	Origin Switch
5	EZ	Index signal
6	DIR	DIR output
7	SRON	Master axis position matches SRSR
8	+EL	Positive Limit Switch
9	-EL	Negative Limit Switch
10	INP	In-Position signal input
11	SSCM	During operation
12	VMV	Positioning operation
13	VORG	Zero return operation
14	VOTP	Output pulses continuously

## @ Syntax

### C/C++(Windows 2000/XP)

```
I16 _8154_db52_set_EMG_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_EMG_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_ELL_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_ELL_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_ORG_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_ORG_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_ALM_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_ALM_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_INP_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_INP_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_EZ_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_EZ_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_ERC_logic(I16 CardId, I16
    logic);
I16 _8154_db52_get_ERC_logic(I16 CardId, I16
    *logic);
I16 _8154_db52_set_erc(I16 CardId, I16 logic);
I16 _8154_db52_get_io_status(I16 CardId, I16
    *io_sts);
```

## Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_set_EMG_logic Lib "8154.dll" Alias
    "_8154_db52_set_EMG_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_EMG_logic Lib "8154.dll" Alias
    "_8154_db52_get_EMG_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_ELL_logic Lib "8154.dll" Alias
    "_8154_db52_set_ELL_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_ELL_logic Lib "8154.dll" Alias
    "_8154_db52_get_ELL_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_ORG_logic Lib "8154.dll" Alias
    "_8154_db52_set_ORG_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_ORG_logic Lib "8154.dll" Alias
    "_8154_db52_get_ORG_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_ALM_logic Lib "8154.dll" Alias
    "_8154_db52_set_ALM_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_ALM_logic Lib "8154.dll" Alias
    "_8154_db52_get_ALM_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_INP_logic Lib "8154.dll" Alias
    "_8154_db52_set_INP_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_INP_logic Lib "8154.dll" Alias
    "_8154_db52_get_INP_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_EZ_logic Lib "8154.dll" Alias
    "_8154_db52_set_EZ_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_EZ_logic Lib "8154.dll" Alias
    "_8154_db52_get_EZ_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
B_8154_db52_set_ERC_logic Lib "8154.dll" Alias
    "_8154_db52_set_ERC_logic" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_ERC_logic Lib "8154.dll" Alias
    "_8154_db52_get_ERC_logic" (ByVal CardId As
        Integer, Logic As Integer) As Integer
```

```
B_8154_db52_set_erc Lib "8154.dll" Alias
    "_8154_db52_set_erc" (ByVal CardId As
        Integer, ByVal Logic As Integer) As Integer
B_8154_db52_get_io_status Lib "8154.dll" Alias
    "_8154_db52_get_io_status" (ByVal CardId As
        Integer, io_sts As Integer) As Integer
```

## @ Argument

**CardId:** The 8154 card index number

**logic:** Setting of active logic

- ▶ logic=0, active LOW, Negative logic.
- ▶ logic=1, active HIGH, Positive logic.

**io\_sts:** I/O status. Please refer to 6.7 function description.

## @ Return Code

```
ERR_NoError
ERR_CardIDMapToCardNo
```



## 6.8 Interrupt Control

### @ Name

`_8154_db52_int_control` – Enable/Disable INT service

`_8154_db52_wait_error_interrupt` – Wait error related interrupts

`_8154_db52_wait_ecam_interrupt` – Wait ecam related interrupts

`_8154_db52_set_ecam_int_factor` – Set INT factor

### @ Description

`_8154_db52_int_control`:

This function is used to enable the Windows interrupt event to host PC.

`_8154_db52_wait_error_interrupt`:

When user enabled the Interrupt function by `_8154_db52_int_control()`. He could use this function to wait the error interrupts.

`error_int_status`: can not be masked

Bit	Interrupt Factor
0	ALM happen and stop
1	Emergency on and stop
2	+End Limit on and stop
3	-End Limit on and stop
4	(Reserved)
5	Master axis angle data illegal
6	Master axis angle data overflow
7	Slave axis feed amount overflow and stop
8	Pulser output overflow and stop
9	Slave encoder input signal error
10	Master encoder input signal error
11	A feed command error

### \_8154\_db52\_wait\_ecam\_interrupt:

When user enabled the Interrupt function by \_8154\_db52\_int\_control() and set the interrupt factors by \_8154\_db52\_set\_ecam\_int\_factor(). User could use this function to wait the specific interrupt. When this function was running, the process would never stop until events were triggered or the function was time out.

**ecam\_int\_status:** can be masked by function call \_8154\_db52\_set\_ecam\_int\_factor()

Bit	Description
0	When the master axis EZ signal turns ON (edge detection)
1	(Reserved) (Always set to 0)
2	When the conditions for comparators 0A and 0B are met.
3	When the conditions for comparators 1A and 1B are met.
4	When the conditions for comparators 2A and 2B are met.
5	When the conditions for comparators 3A and 3B are met.
6	When the conditions for comparators 4A and 4B are met.
7	When the conditions for comparators 5A and 5B are met.
8	When the conditions for comparators 6A and 6B are met.
9	When the conditions for comparators 7A and 7B are met.
10	When EZ is ON, Master axis position does not indicate the zero position
11	When EZ is ON, Master axis position will not reach the value in SRSR
12	When the slave axis is start
13	When the slave axis is stop(including error stop)
14	When the master axis position reach the value in SRSR

### \_8154\_db52\_set\_ecam\_int\_factor:

This function allows users to select motion related factors to initiate the event int. The error can never be masked once the interrupt service is turned on by \_8154\_db52\_int\_control(). Once the Interrupt function is enabled, you can use \_8154\_db52\_wait\_ecam\_interrupt() to wait event.

## @ Syntax

### C/C++(Windows 2000/XP)

```
I16 _8154_db52_int_control(I16 CardId, I16
    intFlag);
I16 _8154_db52_wait_error_interrupt(I16 CardId,
    I32 TimeOut_ms );
I16 _8154_db52_wait_ecam_interrupt(I16 CardId,
    I16 IntFactorBitNo, I32 TimeOut_ms );
I16 _8154_db52_set_ecam_int_factor(I16 CardId,
    I16 int_factor );
```

### Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_int_control Lib "8154.dll" Alias
    "_8154_db52_int_control" (ByVal CardId As
    Integer, ByVal intFlag As Integer) As
    Integer
B_8154_db52_wait_error_interrupt Lib "8154.dll"
    Alias "_8154_db52_wait_error_interrupt"
    (ByVal CardId As Integer, ByVal TimeOut_ms
    As Long) As Integer
B_8154_db52_wait_ecam_interrupt Lib "8154.dll"
    Alias "_8154_db52_wait_ecam_interrupt"
    (ByVal CardId As Integer, ByVal
    IntFactorBitNo As Integer, ByVal TimeOut_ms
    As Long) As Integer
B_8154_db52_set_ecam_int_factor Lib "8154.dll"
    Alias "_8154_db52_set_ecam_int_factor"
    (ByVal CardId As Integer, ByVal int_factor
    As Integer) As Integer
```

## @ Argument

**CardId:** The 8154 card index number 0,1,2,3...

**intFlag:** Enable/Disable the Interrupt function

Value	Meaning
0	Disable
1	Enable

**int\_factor:** interrupt factor

ECAM INT factors

Value meaning: (0: Disable, 1:Enable)

Bit	Description
0	When the master axis EZ signal turns ON (edge detection)
1	(Reserved) (Always set to 0)
2	When the conditions for comparators 0A and 0B are met.
3	When the conditions for comparators 1A and 1B are met.
4	When the conditions for comparators 2A and 2B are met.
5	When the conditions for comparators 3A and 3B are met.
6	When the conditions for comparators 4A and 4B are met.
7	When the conditions for comparators 5A and 5B are met.
8	When the conditions for comparators 6A and 6B are met.
9	When the conditions for comparators 7A and 7B are met.
10	When EZ is ON, Master axis position does not indicate the zero position
11	When EZ is ON, Master axis position will not reach the value in SRSR
12	When the slave axis is start
13	When the slave axis is stop(including error stop)
14	When the master axis position reach the value in SRSR

**TimeOut\_ms:** Specifies the time-out interval, in milliseconds. If TimeOut\_ms is zero, the function tests the states of the specified objects and returns immediately. If TimeOut\_ms is -1, the function's time-out interval never elapses (infinite).

**IntFactorBitNo:** Specifies the bit number of the INT factor.  
 eg. IntFactorBitNo = 12, It means waiting the factor of "the slave axis is start" interrupt.

## **@ Return Code**

ERR\_NoError  
ERR\_CardIDMapToCardNo  
ERR\_EventNotEnableYet  
ERR\_FactorNoExceed  
ERR\_AxisIntWaitError

## 6.9 Position Control and Counters

### @ Name

`_8154_db52_get_master_position` – Get the value of the Master feedback position counter

`_8154_db52_set_master_position` – Set the Master feedback position counter

`_8154_db52_get_slave_position` – Get the value of the Slave feedback position counter

`_8154_db52_set_slave_position` – Set the feedback Slave position counter

`_8154_db52_get_command` – Get the value of the command position counter

`_8154_db52_set_command` – Set the command position counter

`_8154_db52_get_res_command` – Get remaining pulses until the end of motion

`_8154_db52_set_res_command` – Set remaining pulses until the end of motion

### @ Description

`_8154_db52_get_master_position:`

This function is used to read the Master feedback position counter value.

`_8154_db52_set_master_position:`

This function is used to change the Master feedback position counter to the specified value

`_8154_db52_get_slave_position:`

This function is used to read the Slave feedback position counter value.

`_8154_db52_set_slave_position:`

This function is used to change the Slave feedback position counter to the specified value

#### \_8154\_db52\_get\_command:

This function is used to read the value of the command position counter. The source of the command position counter is the pulse output of the DB8152.

#### \_8154\_db52\_set\_command:

This function is used to change the value of the command position counter.

#### \_8154\_db52\_get\_res\_command:

This function is used to read remaining pulse counts until the end of the current motion.

#### \_8154\_db52\_set\_res\_command:

This function is used to change remaining pulse counts until the end of the current motion.

### @ Syntax

#### **C/C++(Windows 2000/XP)**

```
I16 _8154_db52_get_master_position(I16 CardId,  
    I32 * MasterPosition);  
I16 _8154_db52_set_master_position(I16 CardId,  
    I32 MasterPosition);  
I16 _8154_db52_get_slave_position(I16 CardId, I32  
    * SlavePosition);  
I16 _8154_db52_set_slave_position(I16 CardId, I32  
    SlavePosition);  
I16 _8154_db52_get_command(I16 CardId, I32  
    *Command);  
I16 _8154_db52_set_command(I16 CardId, I32  
    Command);  
I16 _8154_db52_get_res_command(I16 CardId, I32  
    *Command);  
I16 _8154_db52_set_res_command(I16 CardId, I32  
    Command);
```

## Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_get_master_position Lib "8154.dll"
  Alias "_8154_db52_get_master_position"
  (ByVal CardId As Integer, MasterPosition As
  Long) As Integer
B_8154_db52_set_master_position Lib "8154.dll"
  Alias "_8154_db52_set_master_position"
  (ByVal CardId As Integer, ByVal
  MasterPosition As Long) As Integer
B_8154_db52_get_slave_position Lib "8154.dll"
  Alias "_8154_db52_get_slave_position"
  (ByVal CardId As Integer, SlavePosition As
  Long) As Integer
B_8154_db52_set_slave_position Lib "8154.dll"
  Alias "_8154_db52_set_slave_position"
  (ByVal CardId As Integer, ByVal
  SlavePosition As Long) As Integer
B_8154_db52_get_command Lib "8154.dll" Alias
  "_8154_db52_get_command" (ByVal CardId As
  Integer, Command As Long) As Integer
B_8154_db52_set_command Lib "8154.dll" Alias
  "_8154_db52_set_command" (ByVal CardId As
  Integer, ByVal Command As Long) As Integer
B_8154_db52_get_res_command Lib "8154.dll" Alias
  "_8154_db52_get_res_command" (ByVal CardId
  As Integer, Command As Long) As Integer
B_8154_db52_set_res_command Lib "8154.dll" Alias
  "_8154_db52_set_res_command" (ByVal CardId
  As Integer, ByVal Command As Long) As
  Integer
```



## @ Argument

**CardId**: The 8154 card index number 0,1,2,3...

**MasterPosition**, **\*MasterPosition**: Master Feedback position counter value,

- ▶ Range: 0-359

**SlavePosition**, **\*SlavePosition**: Master Feedback position counter value,

- ▶ Range: 0-536870912

**Command**, **\*Command**: Command position counter value,

- ▶ Range: 0-536870912

## @ Return Code

ERR\_NoError

ERR\_CardIDMapToCardNo

ERR\_PosOutofRange

## 6.10 Position Compare and Latch

### @ Name

`_8154_db52_get_EZ_latch_data` – Get EZ latched counter data

`_8154_db52_get_comparator_data` – Get comparator data

`_8154_db52_set_comparator_data` – A general function for setting comparator data

`_8154_db52_set_comparator_mode` – A general function for setting comparator mode

`_8154_db52_set_comparator_do` – Set comparator logic

### @ Description

`_8154_db52_get_EZ_latch:`

After the EZ latch signal arrived, this function is used to read the latched value of counters.

`_8154_db52_get_comparator:`

This function is used to get current comparing data of the designated comparator.

`_8154_db52_set_comparator:`

This function is used to set current comparing data of the designated comparator.

`_8154_db52_set_comparator_mode:`

This function is used to set the comparator mode

`_8154_db52_set_comparator_do:`

This function is used to set the comparator logic

## @ Syntax

### C/C++(Windows 2000/XP)

```
I16 _8154_db52_get_EZ_latch_data(I16 CardId, I32
    *Latch_Position);
I16 _8154_db52_get_comparator_data(I16 CardId,
    I16 CompNo, I16 Comp_A_B, I32 *Comp_Data);
I16 _8154_db52_set_comparator_data(I16 CardId,
    I16 CompNo, I16 Comp_A_B, I32 Comp_Data);
I16 _8154_db52_set_comparator_mode(I16 CardId,
    I16 CompNo, I16 Comp_A_B, I16 Comp_Mode);
I16 _8154_db52_set_comparator_do(I16 CardId, I16
    CompNo, I16 logic);
```

### Visual Basic 6(Windows 2000/XP)

```
B_8154_db52_get_EZ_latch_data Lib "8154.dll"
    Alias "_8154_db52_get_EZ_latch_data" (ByVal
        CardId As Integer, Latch_Position As Long)
    As Integer
B_8154_db52_get_comparator_data Lib "8154.dll"
    Alias "_8154_db52_get_comparator_data"
    (ByVal CardId As Integer, ByVal CompNo As
    Integer, ByVal Comp_A_B As Integer,
    Comp_Data As Long) As Integer
B_8154_db52_set_comparator_data Lib "8154.dll"
    Alias "_8154_db52_set_comparator_data"
    (ByVal CardId As Integer, ByVal CompNo As
    Integer, ByVal Comp_A_B As Integer, ByVal
    Comp_Data As Long) As Integer
B_8154_db52_set_comparator_mode Lib "8154.dll"
    Alias "_8154_db52_set_comparator_mode"
    (ByVal CardId As Integer, ByVal CompNo As
    Integer, ByVal Comp_A_B As Integer, ByVal
    Comp_Mode As Integer) As Integer
B_8154_db52_set_comparator_do Lib "8154.dll"
    Alias "_8154_db52_set_comparator_do" (ByVal
    CardId As Integer, ByVal CompNo As Integer,
    ByVal Logic As Integer) As Integer
```

## @ Argument

**CardId:** The 8154 card index number 0,1,2,3...

**Latch\_Position:** Latch position counter value

**CompNo:** The comparator number of axis

- ▶ CompNo=0, Comparator1
- ▶ CompNo=1, Comparator2
- ▶ CompNo=2, Comparator3
- ▶ CompNo=3, Comparator4
- ▶ CompNo=4, Comparator5
- ▶ CompNo=5, Comparator6
- ▶ CompNo=6, Comparator7
- ▶ CompNo=7, Comparator8

**Comp\_A\_B:**

- ▶ CmpSrc=0, Comp\_A
- ▶ CmpSrc=1, Comp\_B

**Comp\_Mode:** The comparing method

- ▶ Comp\_Mode=0, No compare
- ▶ Comp\_Mode=1, Comp\_Data <Counter
- ▶ Comp\_Mode=2,
- ▶ Comp\_Mode=3, Comp\_Data >=Counter

**Comp\_Data:** Comparing value,

**Logic:** Setting of active logic

- ▶ logic=0, active LOW, Negative logic.
- ▶ logic=1, active HIGH, Positive logic.

## @ Return Code

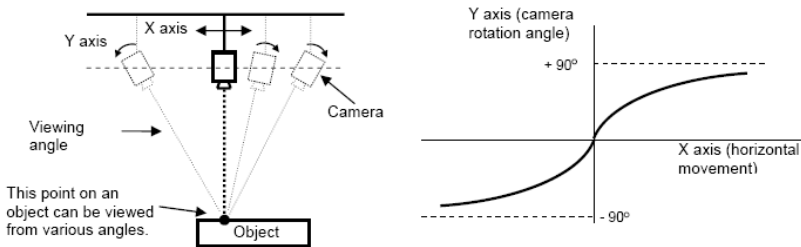
ERR\_NoError  
ERR\_CardIDMapToCardNo  
ERR\_CompareMethodError  
ERR\_PosOutOfRange

# Appendix

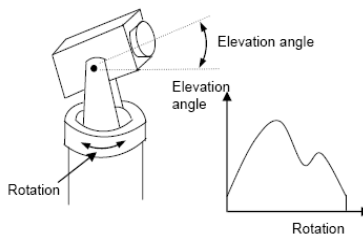
## 7.1 Electronic Cam Examples

### 7.1.1 Monitoring an Object's Appearance

This is an application for viewing a certain part of an object from various angles. When the camera moves horizontally (along the X axis), while always pointed at a certain part of the object, the camera angle (Y axis) must rotate according to the X axis movement. The relationship of these two motions is a tangent, or tan function, as shown in the figure below.

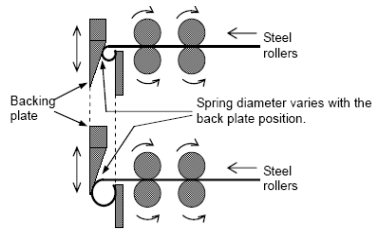


### 7.1.2 Tracking Device



If the angle of elevation of a searchlight is determined by the rotation angle (on the master axis), specify an angle of elevation for each angle of rotation, so that the light will follow a specified trajectory. It can also be set to move in an emergency according to a preset pattern that is different from its normal path.

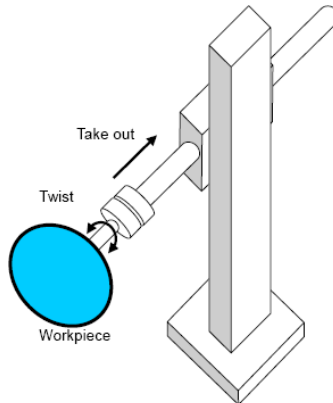
### 7.1.3 Coil Spring



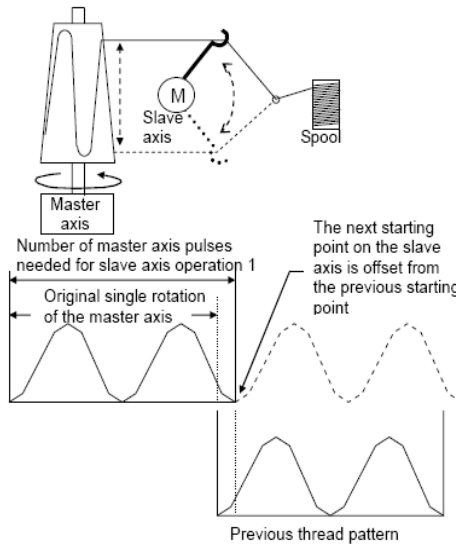
If a backing plate in a fixed position (slave axis) is used to deflect a steel wire (master axis) being extruded at a constant speed, springs will be made with a constant diameter. However, if the backing plate is moved so that the wire hits the plate at different distances, springs of various shapes can be made.

### 7.1.4 Remove and Rotate Mechanism

If you want to rotate a work-piece while taking it out of a mold after it is formed, make the removal axis the master axis and the rotating axis the slave axis. Then, the angle of rotation is always the same relative to any degree of extraction (this is sometimes done to allow an obstacle to be avoided while removing a work-piece).



## 7.1.5 Winding Machine



When the master axis rotates, a thread can be pulled out of a spool and wound on the master axis in any required shape. This system controls the location on the master axis where the thread is wound by controlling the slave axis. In a normal cam operation, the slave axis repeats the same operation for each turn of the master axis.

However, the position of the wound thread is always the same and a good winding shape cannot be obtained. In order to wind the thread well, the winding position of the thread needs to be shifted little by little.

To shift the position, the operation cycle of the slave axis must be set to one rotation of the master axis +  $a$ , not just one turn of the master axis.

When the slave axis ends one operation, if the master axis has rotated a little more than one turn, the next starting position on the slave axis will be offset.

