



**ADLINK**  
TECHNOLOGY INC.

**DB-8153**  
**Modulized Function Board - Motionnet**  
**User's Manual**

**Manual Rev.** 2.00  
**Revision Date:** May 20, 2008  
**Part No:** 50-14011-1000



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>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 .....	4
1.2 Supported Software .....	5
<b>2 Installation .....</b>	<b>7</b>
2.1 Package Contents .....	7
2.2 DB-8153 and DB-8153-RJ45 Outline Drawing .....	8
2.3 DB-8153 Hardware Installation .....	10
Hardware configuration .....	10
Installation Procedures .....	10
Troubleshooting: .....	11
2.4 Software Driver Installation .....	12
2.5 DB-8153 Pin Assignments .....	13
CN3 Pin Description .....	13
2.6 DB-8153-RJ45 Pin Assignments .....	14
CN1 Pin Description .....	14
RJ1 Pin Description .....	15
<b>3 MNET Slave Motion System .....</b>	<b>17</b>
3.1 MNET System .....	18
Specifications .....	18
Wiring Cable .....	19
3.2 .....	MNET Motion Modules 21
Single Axis Motion Modules .....	21
3.3 Operation Theories .....	26
MNET System Communication .....	26
MNET Communication Types .....	28
Motion Control Modes .....	30
3.4 The motor Driver Interface .....	56
Pulse Command Output Interface .....	56
Pulse feedback input interface .....	59
In position signal .....	62
Servo alarm signal .....	63
Error clear signal .....	64

	Servo ON/OFF switch .....	65
	Servo Ready Signal .....	65
	Servo alarm reset switch .....	65
	Gain Selection switch .....	65
3.5	Mechanical switch interface .....	66
	Original or Home (Zero position) signal .....	66
	End-Limit switch signal .....	66
	Slow down switch signal .....	67
	Emergency stop input .....	67
3.6	The Counters .....	68
	Command position counter .....	68
	Feedback position counter .....	69
	Command and Feedback error counter .....	69
	Target position recorder .....	69
3.7	The Comparators .....	70
	Soft end-limit comparators .....	70
	Command and feedback error counter comparators ....	70
3.8	Other Motion Functions.....	71
	Backlash compensation and slip corrections .....	71
	Vibration restriction function .....	71
	Speed profile calculation function .....	72
<b>4</b>	<b>DB-8153 Motionnet Master Utility.....</b>	<b>73</b>
4.1	Software Installation.....	74
4.2	ADLINK DB8153 Motionnet Master Utility .....	75
	Select the type of carrier card .....	75
	Run Motionnet Master Utility .....	76
	MNET motion Utility .....	77
	Slave Live Scan .....	87
<b>5</b>	<b>Function Library.....</b>	<b>89</b>
5.1	List of Functions.....	90
5.2	C/C++ Programming Library .....	95
5.3	System & Initialization.....	96
5.4	Pulse Input/Output Configuration.....	102
5.5	Velocity Mode Motion.....	105
5.6	Single Axis Position Mode .....	108
5.7	Home Return Mode.....	113
5.8	Motion Status.....	117
5.9	Motion Interface I/O .....	119

5.10	Position Control and Counters .....	127
5.11	Position Compare and Latch .....	132
5.12	Soft Limit.....	136
5.13	Backlash Compensation/Vibration Suppression .....	138
5.14	Speed Profile Calculation .....	141
5.15	Return Code .....	146

## List of Tables

Table 1-1: DB-815x Series .....	1
Table 2-1: DB-8153 CN3 Pin Assignment .....	13
Table 2-2: DB-8153-RJ45 CN1 Pin Assignment .....	14
Table 2-3: DB-8153-RJ45 RJ1 Pin Assignment .....	15
Table 3-1: MNET Motion Module Series .....	21



## List of Figures

Figure 1-1: Block Diagram of the DB-8153 .....	3
Figure 2-1: DB-8153 PCB Layout .....	8
Figure 2-2: DB-8153-RJ45 PCB Layout.....	9
Figure 2-3: Board Configuration.....	10
Figure 4-1: PCI-8154/58 Installation Page.....	74
Figure 4-2: Carrier Selection Dialog.....	75



# 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**

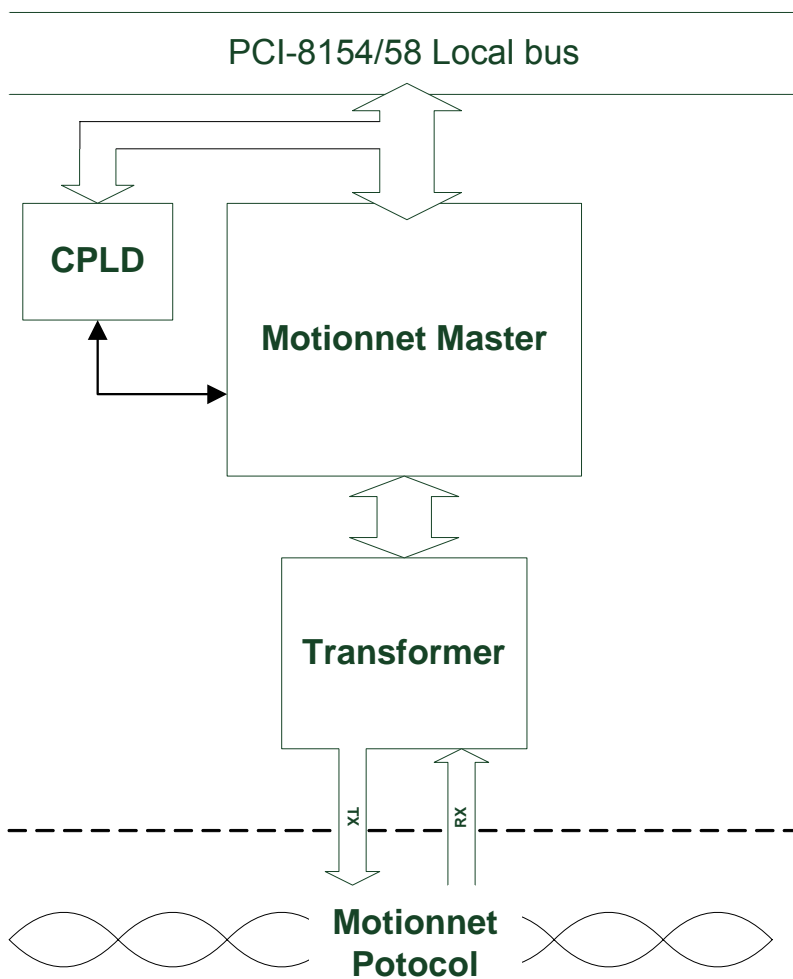
The DB-8153 provides a Motionnet system (referred to as “MNET”) for distributed motion on a PCI-8154/58 used in machine system. MNET is an innovative distributed motion technology which allows 64 of single axis motion control module to be scanned in millisecond-level real time by using master-slave architecture. By using general Ethernet cable with 3-pins connector, users may easily set up the MNET single axis motion control module on servomotor. The remote single axis motion module can control continuous operations of a servomotor with a variety of speed patterns (constant speed, linear acceleration/deceleration, S-curve acceleration/deceleration, as well as a preset positioning operation, and a zero return operation) using serial communications.

Since this board can be connected directly to the mechanical I/O signals from a servo driver, it does not need the special servo driver cable that is required for a conventional motion control board. Therefore, this board will save you many man-hours in designing and wiring your system. It brings many advantages such as simplified wiring, shortened wiring runs, and it eliminates problems caused by faulty wiring. It also offers high noise immunity and takes full advantage of high-speed signal lines to handle command pulses. This remote motion module is so compact that it does not need any extra space left for wiring. This local network features rapid response, real-time scanning. With DB-8153 module, users can also make the integration with multiply motion modules operation.

Motionnet solutions are for those who want:

- ▶ Serial single axis motion control
- ▶ More flexible motion system configurations
- ▶ Easy wiring solution for distributed single axis motion modules
- ▶ Low-profile discrete motion module for fitting into limited space
- ▶ Up to 64 axis motion controller
- ▶ Real-time and fast scanning
- ▶ High speed and response motion control

The block diagram of DB-8153 is as follows.



**Figure 1-1: Block Diagram of the DB-8153**

## **1.1 Specifications**

### **Interface**

- ▶ For use with the PCI-8154 and PCI-8158 PCI cards only
- ▶ The DB-8151-RJ45 must also be used when using the DB-8153

### **Master Controller**

- ▶ Motionnet ASIC master control
- ▶ 80 MHz external clock

### **Interface**

- ▶ RS-485 with transformer isolation
- ▶ Half duplex communication
- ▶ 2.5/5/10/20 Mbps transmission rate can be set by software (20 Mbps default)

### **Connector**

- ▶ RJ45 connector x 2 on DB-8153-RJ45

### **Interrupt**

- ▶ Status read back

### **LED Indicator**

- ▶ Link status

### **Dimension**

- ▶ 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-8153. These function libraries are shipped with the board and supports Windows 2000/XP.





## 2 Installation

This chapter describes how to install the DB-8153. 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 2.4, page 12)
- ▶ Understanding the I/O signal connections (Chapter 3, page 17) and their operations (Chapter 4, page 73)

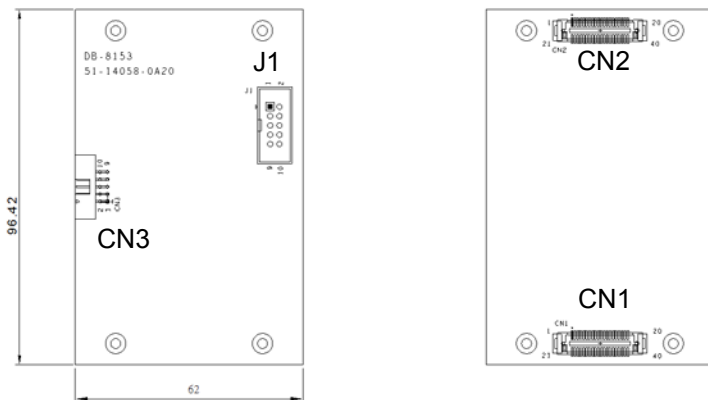
### 2.1 Package Contents

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

- ▶ DB-8153: 1 channel high speed trigger board x1
- ▶ DB-8153-RJ45: Motionnet interface board x1
- ▶ Copper Pillar x4
- ▶ Screws x8
- ▶ 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.

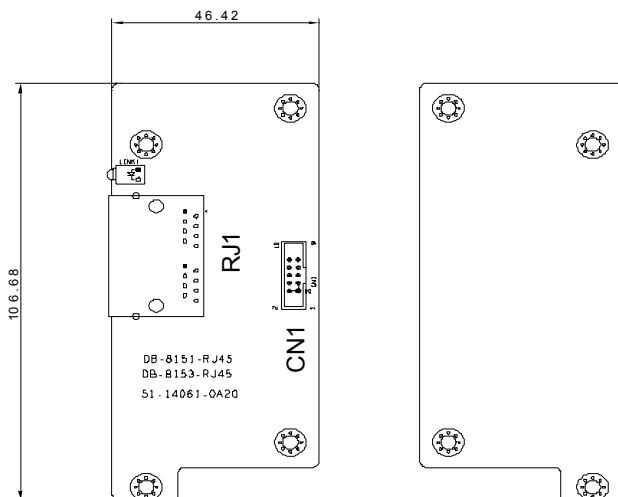
## 2.2 DB-8153 and DB-8153-RJ45 Outline Drawing



**Figure 2-1: DB-8153 PCB Layout**

### DB-8153:

- ▶ CN1, CN2: Daughter board connectors for the PCI-8154/58
- ▶ CN3: Main signal connector



**Figure 2-2: DB-8153-RJ45 PCB Layout**

**DB-8153-RJ45:**

- ▶ RJ1: Main MNET master controller connector
- ▶ CN1: Link to DB-8153

## 2.3 DB-8153 Hardware Installation

### 2.3.1 Hardware configuration

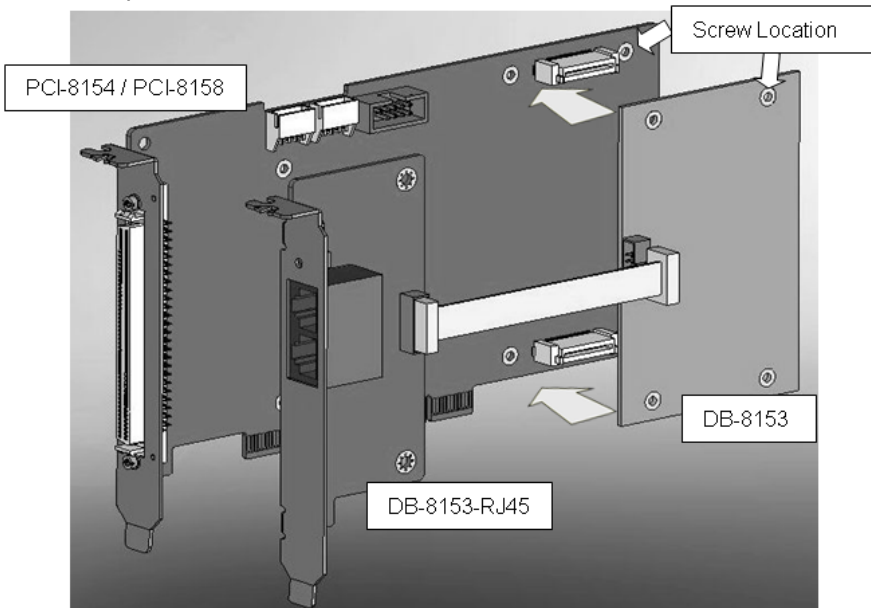
DB-8153 must be installed on to a PCI-8154 or PCI-8158 only. Please ensure correct orientation of the DB-8153 before attaching it to the PCI-8154 or PCI-8158.

### 2.3.2 Installation Procedures

Please follow installation procedure as follows.

#### Daughter Board Installation

1. Attach the DB-8153 on to the PCI-8154 or PCI-8158. Please ensure correct orientation of the DB-8153 daughter board.
2. Screw the eight screws into the corresponding copper pillars.
3. Connect the DB-8153 and DB-8153-RJ45 with the 10-pin flat cable.



**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. 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-8153 and DB-8153-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-8153-RJ45 bracket onto rear panel.

### **2.3.3 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.

## 2.4 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.

## 2.5 DB-8153 Pin Assignments

### 2.5.1 CN3 Pin Description

CN3 is the main DB-8153 connector linking to DB-8153-RJ45 via a 10-pin flat cable.

PIN NO	PIN OUT
PIN 1	+5V
PIN 2	FG
PIN 3	DG
PIN 4	LED Signal
PIN 5	RXD1
PIN 6	TXD
PIN 7	RXD2
PIN 8	TXE
PIN 9	DG
PIN 10	FG

**Table 2-1: DB-8153 CN3 Pin Assignment**

## 2.6 DB-8153-RJ45 Pin Assignments

### 2.6.1 CN1 Pin Description

CN1 is the main DB-8153-RJ45 connector linking to the DB-8153 through a 10-pin flat cable.

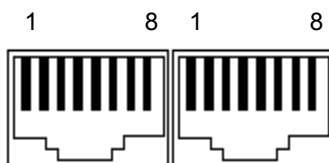
PIN NO	PIN OUT
PIN 1	+5V
PIN 2	FG
PIN 3	DG
PIN 4	LED Signal
PIN 5	RXD1
PIN 6	TXD
PIN 7	RXD2
PIN 8	TXE
PIN 9	DG
PIN 10	FG

**Table 2-2: DB-8153-RJ45 CN1 Pin Assignment**



## 2.6.2 RJ1 Pin Description

RJ1 is the RJ45 connector on the DB-8153-RJ45.



PIN NO.	PIN OUT
PIN 1	NC
PIN 2	NC
PIN 3	NC
PIN 4	Data-
PIN 5	Data+
PIN 6	NC
PIN 7	NC
PIN 8	NC

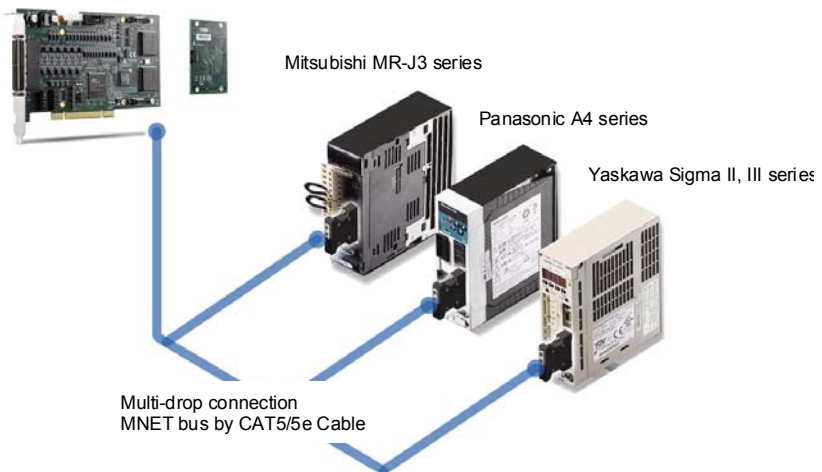
**Table 2-3: DB-8153-RJ45 RJ1 Pin Assignment**



### 3 MNET Slave Motion System

The MNET is a master-slave network system featuring an ultra-speed distributed motion architecture that modularizes communication. ADLINK provides single axis slave motion modules to implement multi-axes system which used simply motion operation by multiply single axis module. This MNET system contains a master controller, single axis motion modules, wiring cables, and several servo drivers (motors).

Central Board: PCI-8154/58 +DB-8153



## 3.1 MNET System

### 3.1.1 Specifications

Functions of MNET system can be classified as serial communications and motion control.

Item	Specifications
Cyclic communication length	<ul style="list-style-type: none"> <li>▶ Maximum of 0.12 msec, when using 8 axes. @#1</li> <li>▶ Maximum of 0.24 msec, when using 16 axes. @#1</li> <li>▶ Maximum of 0.49 msec, when using 32 axes. @#1</li> <li>▶ Maximum of 0.97 msec, when using 64 axes. @#2</li> </ul> <p>(Data transfer speed: 20 Mbps, when using our recommended cable #1: 100 m, #2: 50 m)</p>
Total serial communication line length	<ul style="list-style-type: none"> <li>▶ Maximum of 100 m (At a data transfer speed of 20 Mbps with 32 axes connected)</li> <li>▶ Maximum of 50 m (At a data transfer speed of 20 Mbps with 64 axes connected)</li> <li>▶ Maximum of 100 m (At a data transfer speed of 10 Mbps with 64 axes connected)</li> </ul> <p>(Using our recommended cables)</p>
Serial communication interface	Pulse transformer and RS-485 specification line transceiver
Serial communication protocol	Our proprietary protocol
Serial communication	NRZ signed
Serial communication method	Half-duplex communication
Connection method	Multi-drop connection using a LAN cable (CAT5/CAT5e STP/S-STP)
Serial data transfer speed	20 Mbps/10 Mbps/5 Mbps/2.5 Mbps programmable speed setting

### 3.1.2 Wiring Cable

This system guarantees enhanced quality for high-speed communication, and is designed to be connected with LAN cables suitable for 100BASE and 1000BASE. These cables have well known specifications, are cheap and easily obtained close to you. Therefore, we do not include these items in our product lines and do not supply them. To select cables you need to connect, make sure they meet the following specifications.

Wiring standard:

- ▶ TIA/EIA-568-B
- ▶ Category 5 (CAT5)
- ▶ Enhanced category 5 (CAT5e)
- ▶ Category 6 (CAT6)

UTP (Unshielded Twisted Pair) cables or STP (Shielded Twisted Pair) cables that meet the specifications above. For an environment with lots of electromagnetic noise, use a shielded cable (STP).

Observe the following when connecting your system.

1. Total serial line length

This system employs a multi-drop connection method. The maximum total extension distance of the line varies, depending on the data transfer speed and the number of local boards that are connected.

- ▶ Max. 100 m (Transfer speed; 20 Mbps with 32 modules connected)
- ▶ Max. 50 m (Transfer speed; 20 Mbps with 64 modules connected)
- ▶ Max. 100 m (Transfer speed; 10 Mbps with connecting 64 modules connected)

2. Minimum cable length

The shortest cable must be at least 60 cm long.

3. Do not mix cables of different types and model in the same serial line.
4. Keep the total serial line length as short as possible.
5. If you are using shielded cables, do not connect the shield on both ends to the FG terminals.

Connecting only one end of the shield on each cable will improve noise immunity.

## 3.2 MNET Motion Modules

MNET single-axis slave module is a wire-saving solution. ADLINK offers three types of modules for connecting to Mitsubishi J3 servos, Panasonic A4 servos and Yaskawa Sigma II and III servos. Each module is easy and convenient to plug into the servo drivers. All of the servos can be connected serially through the recommended cable, greatly reducing wiring requirements.

Series	Model	Servo Driver	Axis Num.	Mechanical I/O
MNET Motion Module	MNET- MIA	Panasonic A4	1	PEL, MEL, ORG, SD, EMG
	MNET- J3	Mitsubishi MR-J3	1	PEL, MEL, ORG, SD, EMG
	MNET- S23	Yaskawa Sigma II, III	1	PEL, MEL, ORG, SD, EMG

**Table 3-1: MNET Motion Module Series**

### 3.2.1 Single Axis Motion Modules

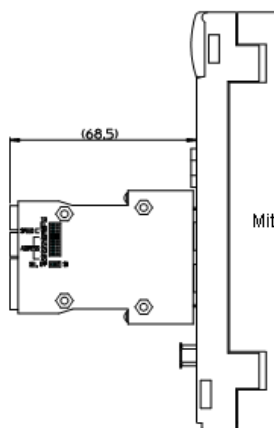
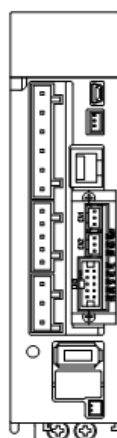
ADLINK offers three different types motion module which was used for three different servos.

1. The MNET-J3 can control a servomotor when I/O signals from a Mitsubishi's MR-J3 series servo driver CN1 are routed directly to this connector, CN4.
2. The MNET-MIA can control a servomotor when I/O signals from a Panasonic's (Matsushita's) servo amplifier MINAS, A, AIII, A4 series (pulse command supporting type) servo amplifier CN1/F or CNX5 are routed directly to this connector, CN4.
3. The MNET-M341-S23 you have purchased is a motion control board that contains NPM G9003 PLC device. When I/O signals from a Yasukawa  $\Sigma$ II,  $\Sigma$ III series servo pack (pulse train supporting types) CN1 connector are routed directly to this connector (CN4), it can control a servomotor.

These modules can control continuous operations of a servomotor with a variety of speed patterns (constant speed, linear acceleration/deceleration, S-curve acceleration/deceleration, as well as a preset positioning operation, and a zero return operation) using serial communications.

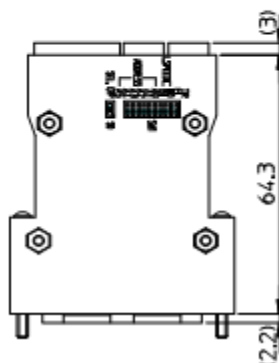
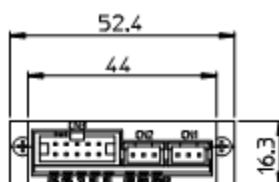
Since these modules can be connected directly to the mechanical I/O signals from a servo driver, they do not need the special servo driver cable that is required for a conventional motion control module. Therefore, these modules will save you many man-hours in designing and wiring your system. The motion module brings many advantages such as simplified wiring, shortened wiring runs, and it eliminates problems caused by faulty wiring. It also offers high noise immunity and takes full advantage of high-speed signal lines to handle command pulses. This module is so compact that it does not need any extra space left for wiring. These modules are controlled by serial communications from DB-8153, a Motionnet central controller.



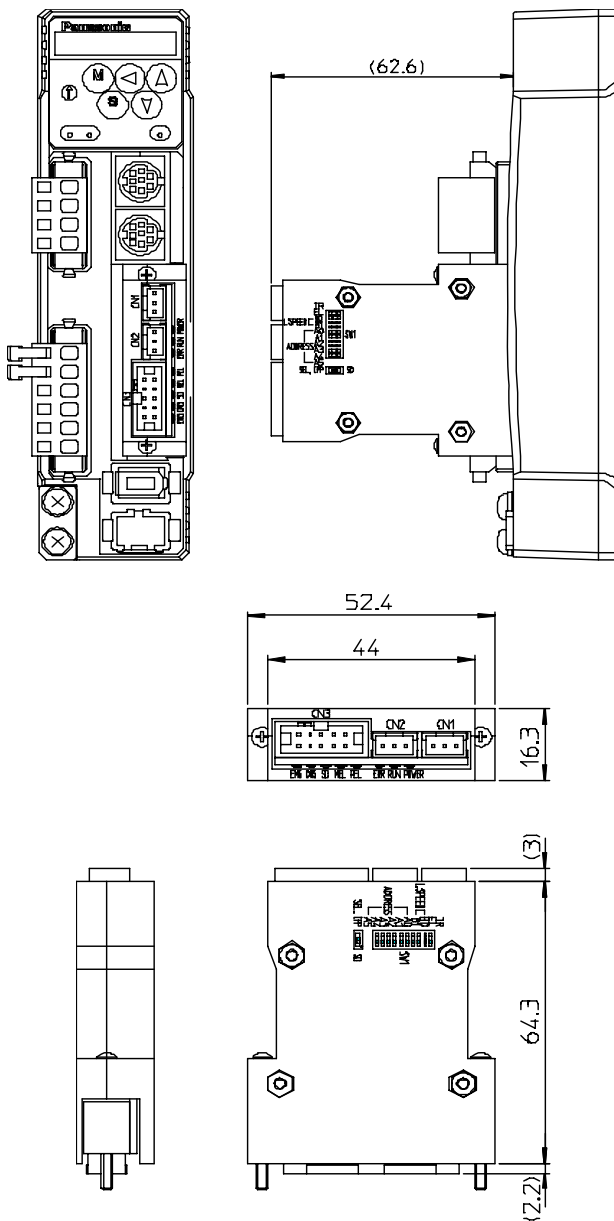


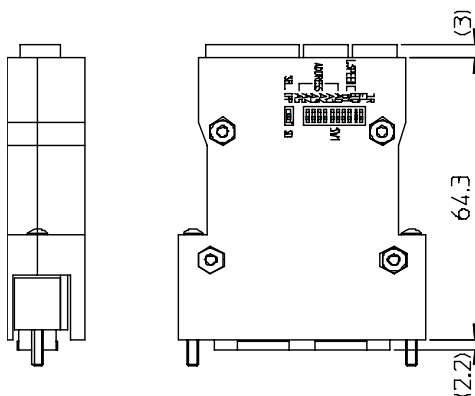
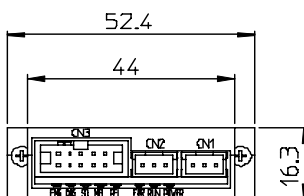
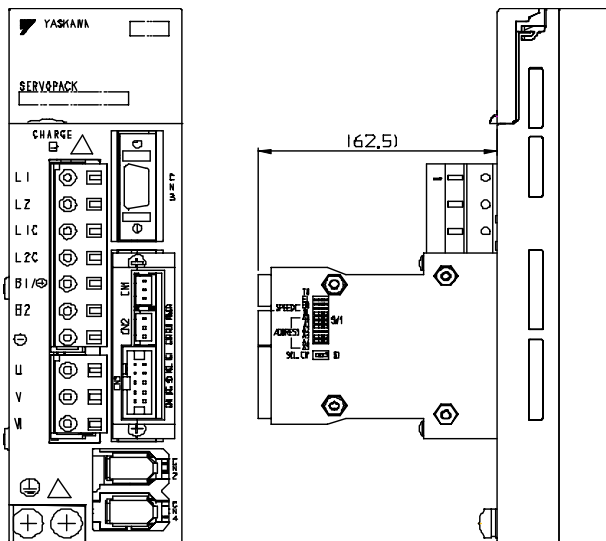
Mitsubishi J3 Servo

Unit: mm



Unit: mm

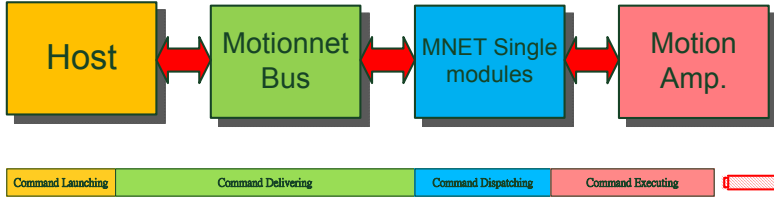




## 3.3 Operation Theories

### 3.3.1 MNET System Communication

There is MNET system communication block diagram as follows.



#### Command Launching:

Within the MNET system, remote modules communicate with each other with MNET network packets. In fact, users do not have to understand what the content of the packet is. Instead, many types of API functions are provided for controlling this module. These functions are easy to understand and use.

APIs can analyze the parameters from user's command and pack them as MNET network packets. The packets are then passed to the remote modules. Then, the remote modules will interpret those packets and execute the commands correctly. Before launching the packet, all the commands issued by users are written into the RAM and transferred on MNET network.

Consequently, the RAM is the bridge between MNET master controller and host PC. The accessing time of RAM for one packet is about 600ns. It is quite fast on host PC. Furthermore, the delivering time of one command on network depends on the number of modules and operating clock rate. Besides the basically RAM usage, user is able to write data into a FIFO in the central device, and issue a send command either above method. This communication will be sent and received automatically by interrupting the cyclic communication. A complete command delivering time depends on the number of MNET packets. One packet command can be delivered in one MNET scan (cycle) time.

## **Command Delivering:**

For a command deliver procedure which is transferred. In cyclic communications, the time allowed for communication by a single module is fixed. However, in data communication, the communication time will vary, depending on how the communication is controlled by the user's program and the time needed to access the DB-8153. We will skip these elements and simply calculate the basic data communication time in this section.

### **[Example:]**

Write a feed amount in 4 bytes of data into single axis motion module.

### **[Calculation formula:]**

Data transfer speed: 20 Mbps

(1) Data transfer time:

- ▷ Number of bytes sent Sending 6 bytes  
(4 bytes of feed amount data + 2 write command bytes are sent)
- ▷ Sending time:  
(Number of bytes x 0.6 + 3.25) = 6 x 0.6 + 3.25  $\mu$ s

(2) Data reception time (Number of bytes received)

- ▷ Receiving the 0 byte  
(No data is received since this is only a writing operation.)
- ▷ Data reception time 5.05  $\mu$ s  
(If there is no data to return, this is a fixed value.)

### **[Data communication time:]**

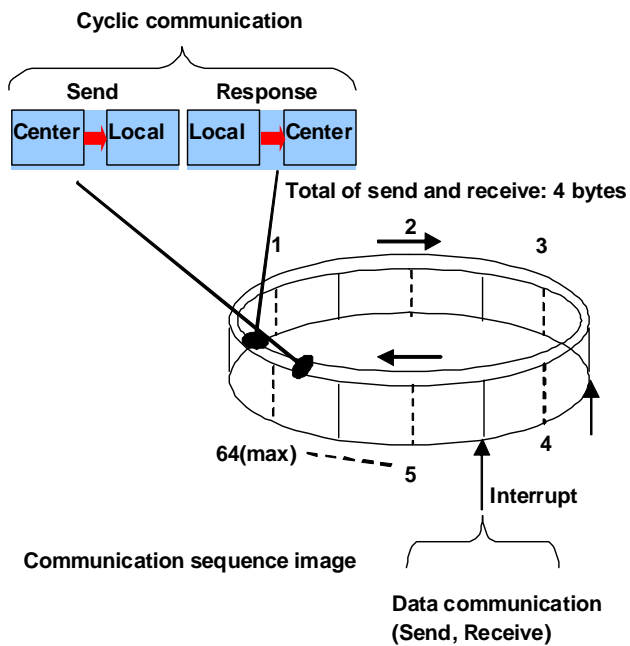
Sending and receiving time:

Data sending time + Data reception time + 7.4 $\mu$ s = 19.3  $\mu$ s  
(maximum)

### 3.3.2 MNET Communication Types

Serial communication in the Motionnet consists of the following three types of communication.

- ▶ **System communication:** By polling the Motionnet line, the number of local devices that are connected, the device numbers, device types, and I/O port allocation status can all be checked.
- ▶ **Cyclic communication:** The system starts communication with the local device that has the lowest device number. When the communication cycle reaches the device with the highest device number, the system starts over again, communicating with the device that has the lowest device number. The process of communicating with all active devices, from the lowest to the highest device, is one cycle. The system repeats this communication cycle endlessly.
- ▶ **Data communication:** This communication type is used to handle data between a motion module and Host. Write data into a FIFO in the central device within Host, and issue a send command. This communication will be sent and received automatically by interrupting the cyclic communication.



### 3.3.3 Motion Control Modes

When motor/stepper control first started, motion control was widely discussed instead of motor control. Motor control was separated into two layers: motor control and motion control. Motor control relates to PWM, power stage, closed loop, hall sensors, vector space, etc. Motion control relates to speed profile generating, trajectory following, and coordinating.

Here the interface of motion and motor control is a pulse train type. As a trend of digital world, pulse trains represent a new concept to motion control. The counts of pulses show how many steps of a motor rotate and the frequency of pulses show how fast a motor runs. The time duration of frequency changes represent the acceleration rate of a motor. Because of this interface, a servo or stepper motor can be easier than an analog type for positioning applications. It means that motion and motor control can be separated more easily by this way.

For pulse type position controllers, the control loops are built outside on the motor drivers and users must tune the gains on drivers.

#### **Pulse Command Output:**

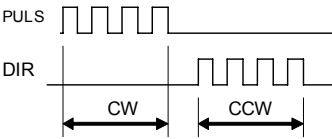
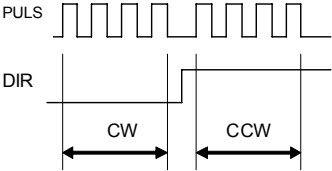
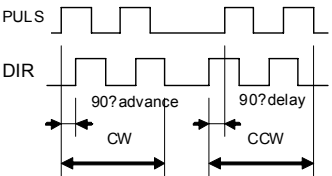
The MNET single axis module uses pulse commands to control servo/stepper motors via the drivers. A pulse command consists of two signals: OUT and DIR.

There are two command types:

1. Single pulse output mode (OUT/DIR type pulse output),
2. Dual pulse output mode (CW/CCW type pulse output).



The modes vs. signal type of OUT and DIR pins are listed in the table below:

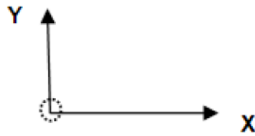
Pulse command method	Description	Examples of command pulse outputs
CW/CCW method (2-pulse mode)	This is a method for outputting pulse signals as motor rotation direction signals CW or CCW. With this system, the PULS terminal outputs CW direction pulse signals, and the DIR terminal outputs CCW direction pulse signals.	
Direction method (common pulse mode)	This is a method for outputting both CW and CCW command pulse signals through the PULS terminal. The output the direction is controlled by a HIGH/LOW level signal on the DIR terminal. As shown in the figure on the right, the motor rotates CW when the DIR signal is LOW and CCW when it is HIGH.	
90° phase difference method (90° phase pulse mode)	This is a method for outputting pulse trains with a 90° phase difference. Signals are output by both the PULS and DIR terminals. In general, when the PULS pulse train is 90° ahead of the DIR pulse train, this is used as a CW rotation command. (This method is only available without a multiplier. It cannot use 2x and 4x multipliers.)	

## Coordinate system

The Cartesian coordinate is used and pulses are in the unit of length. The physical length depends on mechanical parts and motor's resolution. For example, if a motor is on a screw ball, and the pitch of screw ball is 10mm and the pulses needed for a round of motor are 10,000 pulses. One pulse's physical unit is equal to  $10\text{mm}/10,000\text{p} = 1\text{ mm}$ .

Just set a command with 15,000 pulses for motion controller if we want to move 15mm. How about if we want to move 15.0001mm?

**Simple! The motion controller will keep the residue value less than 1 pulse and add it to next command.**

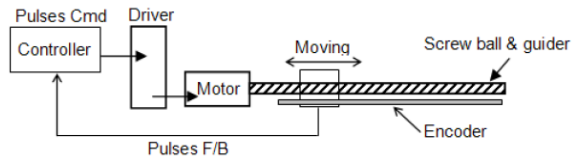


The motion controller sends incremental pulses to motor drivers. It means that we can only send relative command to motor driver. But we can solve this problem by calculating the difference between current position and target position first. Then send the differences to motor driver. For example, if current position is 1000 and we want to move a motor to 9000, you can use an absolute command to set a target position of 9000. Inside the motion controller, it will get current position 1000 first then calculate the difference from target position. The result is +8000. So, the motion controller will send 8000 pulses to motor driver to move the position of 9000.

Sometimes, users need to install a linear scale or external encoder to check machine's position. But how do you to build this coordinate system? If the resolution of external encoder is 10,000 pulses per 1mm and the motor will move 1mm if the motion controller send 1,000 pulses, It means that when we want to move 1 mm, we need to send 1,000 pulses to motor driver then we will get the encoder feedback value of 10,000 pulses. If we want to use an absolute command to move a motor to 10,000 pulses position and current position read from encoder is 3500 pulses, how many

pulses will it send to motor driver? The answer is  $(10000 - 3500) / (10,000 / 1,000) = 650$  pulses.

The motion controller will calculate it automatically if users set “move ratio” already. The “move ratio” means the (feedback resolution/command resolution).

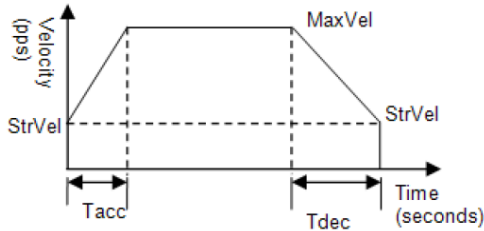


## **Absolute and relative position move**

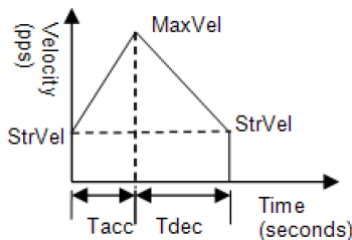
In the coordinate system, we have two kinds command for users to locate the target position. One is absolute and the other is relative. Absolute command means that user give the motion controller a position, then the motion controller will move a motor to that position from current position. Relative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

## Trapezoidal speed profile

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



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

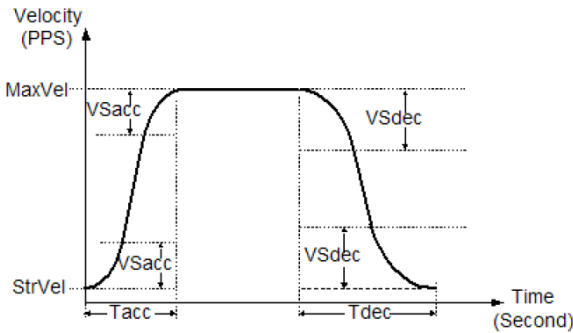


This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.

## S-curve and Bell-curve speed profile

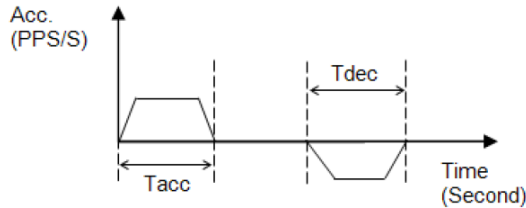
S-curve means the speed profile in accelerate/decelerate area follows a 2nd order curve. It can reduce vibration at the beginning of motor start and stop. In order to speed up the acceleration/deceleration during motion, we need to insert a linear part into these areas. We call this shape as “Bell” curve. It adds a linear curve between the upper side of s-curve and lower side of s-curve. This shape improves the speed of acceleration and also reduces the vibration of acceleration.

For a bell curve, we define its shape parameters as below:



- ▶ **Tacc**: Acceleration time in second
- ▶ **Tdec**: Deceleration time in second
- ▶ **StrVel**: Starting velocity in PPS
- ▶ **MaxVel**: Maximum velocity in PPS
- ▶ **VSacc**: S-curve part of a bell curve in deceleration in PPS
- ▶ **VSdec**: S-curve part of a bell curve in deceleration in PPS

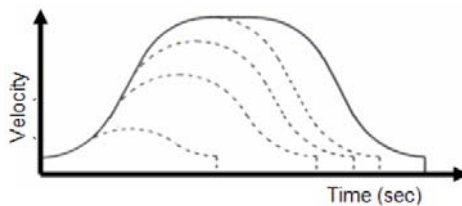
If  $VS_{acc}$  or  $VS_{dec}=0$ , it means acceleration or deceleration use pure S-curve without linear part. The Acceleration chart of bell curve is shown below:



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

The rule is to lower the value of  $MaxVel$  and the  $T_{acc}$ ,  $T_{dec}$ ,  $VS_{acc}$ ,  $VS_{dec}$  automatically, and keep  $StrVel$ , acceleration, and jerk unchanged. This is also applicable to Trapezoidal profile motion.

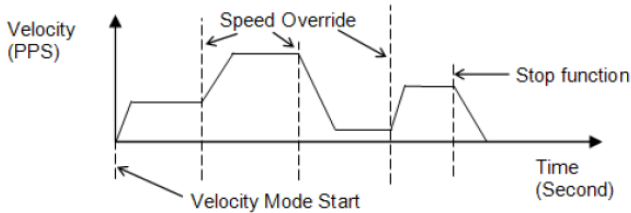
This kind of speed profile could be applied on velocity mode, position mode in one axis or multi-axes linear interpolation and two axes circular interpolation modes.



## Velocity mode

Velocity mode means the pulse command is continuously outputting until a stop command is issued. The motor will run without a target position or desired distance unless it is stopped by other reasons. The output pulse accelerates from a starting velocity to a specified maximum velocity. It can follow a linear or S-curve acceleration shape. The pulse output rate is kept at maximum velocity until another velocity command is set or a stop command is issued. The velocity could be overridden by a new speed setting.

Notice that the new speed could not be a reversed speed of original running speed. The speed profile of this kind of motion is shown as below:

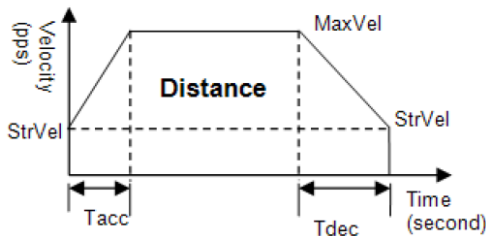




## One axis position mode

Position mode means the motion controller will output a specific amount of pulses which is equal to users' desired position or distance. The unit of distance or position is pulse internally on the motion controller. The minimum length of distance is one pulse.

However, in MNET motion control system, we provide a floating point function for users to transform a physical length to pulses. Inside our software library, we will keep those distance less than one pulse in register and apply them to the next motion function. Besides positioning via pulse counts, our motion controller provides three types of speed profile to accomplish positioning. There are 1st order trapezoidal, 2nd order S-curve, and mixed bell curve. Users can call respective functions to perform that. The following chart shows the relationship between distance and speed profile. We use trapezoidal shape to show it.



The distance is the area of the V-t diagram of this profile.

## Home Return Mode

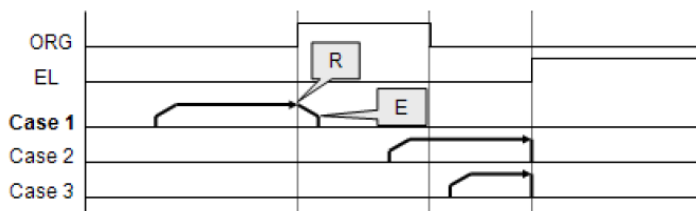
Home return means searching a zero position point on the coordinate. Sometimes, users use a ORG, EZ or EL pin as a zero position on the coordinate. At the beginning of machine power on, the program needs to find a zero point of this machine. Our motion controller provides a home return mode to make it.

We have many home modes and each mode contents many control phases. All of these phases are done by ASIC. No software efforts or CPU loading will be taken. After home return is finished, the target counter will be reset to zero at the desired condition of home mode. For example, a raising edge when ORG input. Sometimes, the motion controller will still output pulses to make machine show down after resetting the counter. When the motor stops, the counter may not be at zero point but the home return procedure is finished. The counter value you see is a reference position from machine's zero point already.

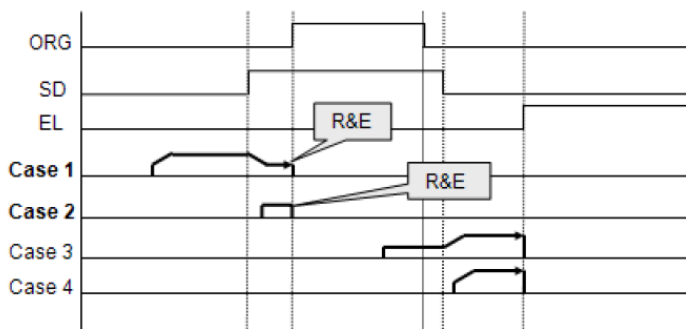
The following figures show the various home modes: R means counter reset (command and position counter). E means ERC signal output.

## Home mode=0: (ORG Turn ON then reset counter)

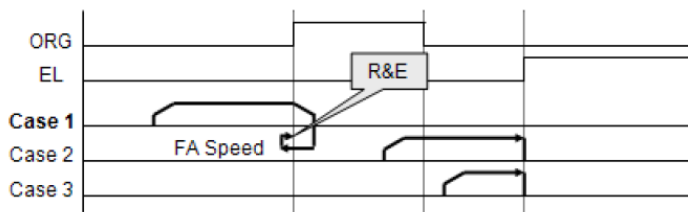
- When SD is not installed



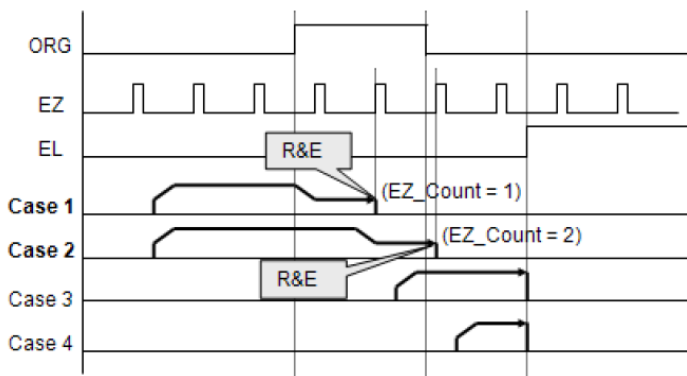
- When SD is installed and SD is not latched



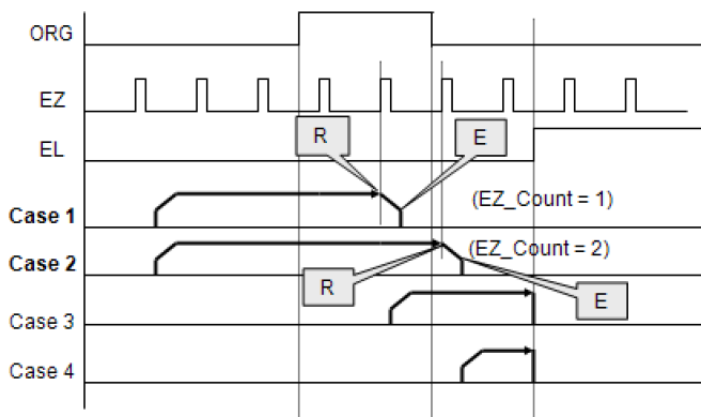
## Home mode=1: (Twice ORG turn ON then reset counter)



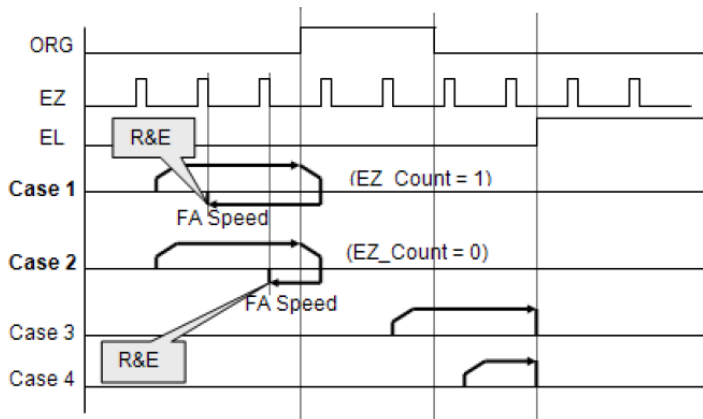
**Home mode=2: (ORG ON then Slow down to count EZ numbers and reset counter)**



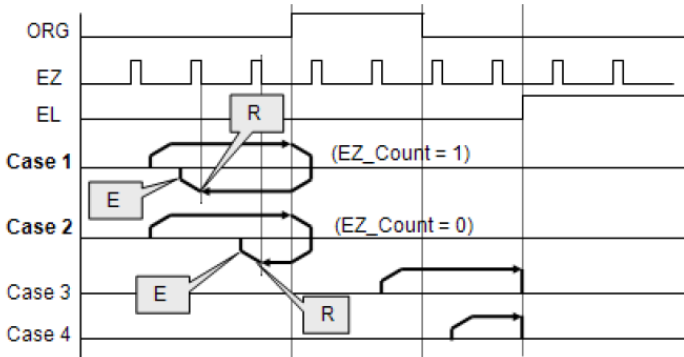
**Home mode=3: (ORG ON then count EZ numbers and reset counter)**



# **Home mode=4: (ORG On then reverse to count EZ number and reset counter)**

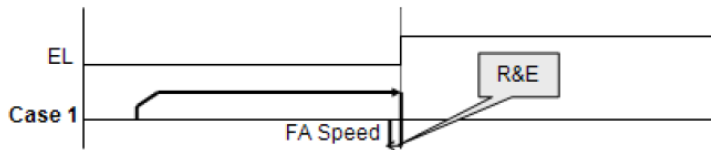


**Home mode=5: (ORG On then reverse to count EZ number and reset counter, not using FA Speed)**

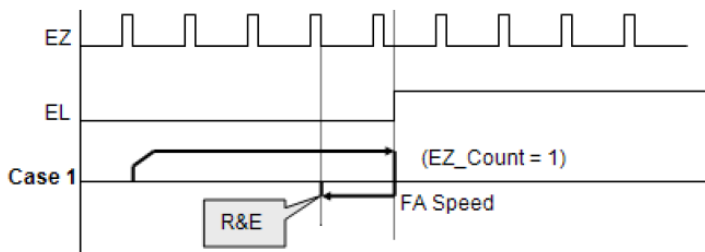




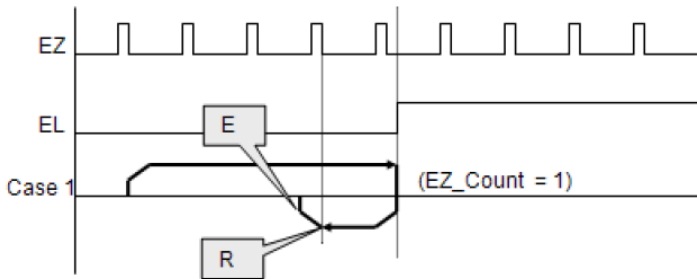
**Home mode=6: (EL On then reverse to leave EL and reset counter)**



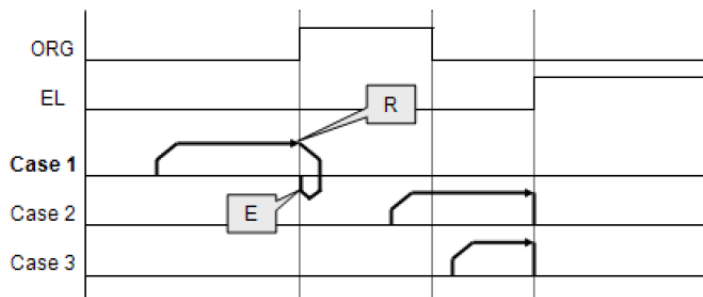
**Home mode=7: (EL On then reverse to count EZ number and reset counter)**



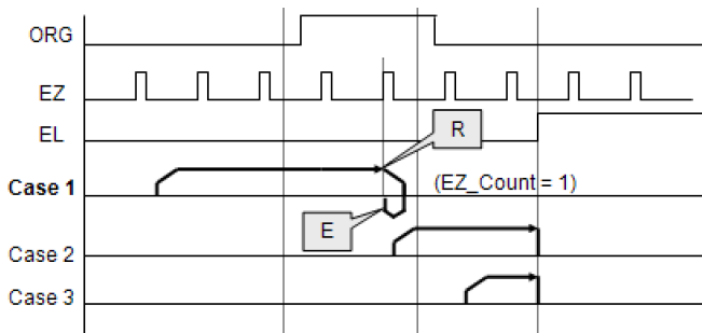
**Home mode=8: (EL On then reverse to count EZ number and reset counter, not using FA Speed)**



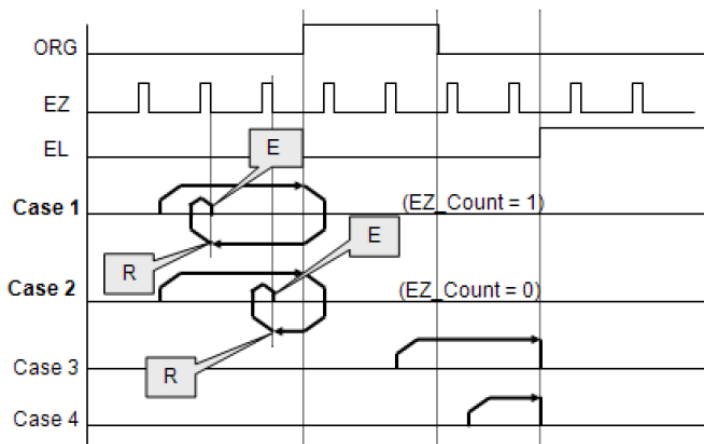
**Home mode=9: (ORG On then reverse to zero position, an extension from mode 0)**



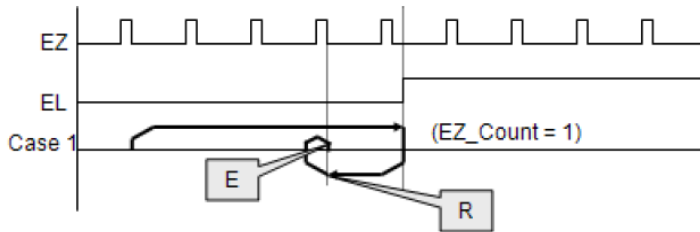
**Home mode=10: (ORG On the counter EZ and reverse to zero position, an extension from mode 3)**



**Home mode=11: (ORG On then reverse to counter EZ and reverse to zero position, an extension from mode 5)**

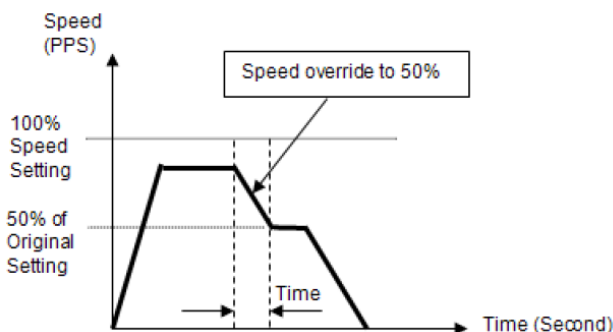


**Home mode=12: (EL On then reverse to count EZ number and reverse to zero position, an extension from mode 8)**



## Speed Override Function

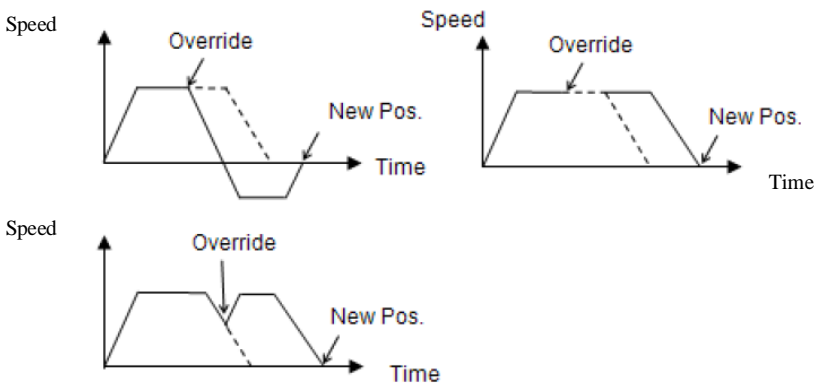
Speed override means that users can change command's speed during the operation of motion. The change parameter is a percentage of original defined speed. Users can define a 100% speed value then change the speed by percentage of original speed when motion is running. If users didn't define the 100%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





## Position Override Function

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



### 3.4 The motor Driver Interface

We provide several dedicated I/Os which can be connected to motor driver directly and have their own functions. Motor drivers have many kinds of I/O pins for external motion controller to use.

We classify them to two groups. One is pulse I/O signals including pulse command and encoder interface. The other is digital I/O signals including servo ON, alarm, INP, servo ready, alarm reset and emergency stop inputs. The following sections will describe the functions these I/O pins.

#### 3.4.1 Pulse Command Output Interface

The motion controller uses pulse command to control servo/step-per 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:

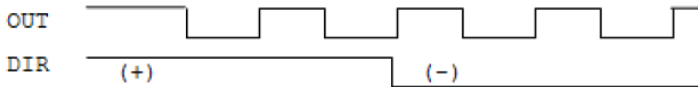
Mode	Output of OUT pin	Output of DIR pin
Dual pulse output (CW/CCW)	Pulse signal in plus (or CW) direction	Pulse signal in minus (or CCW) direction
Single pulse output (OUT/DIR)	Pulse signal	Direction signal (level)

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

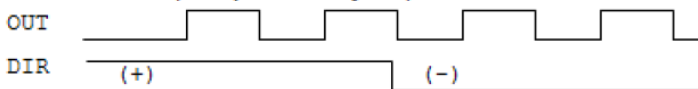
In this mode, the OUT pin is for outputting command pulse chain.

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)

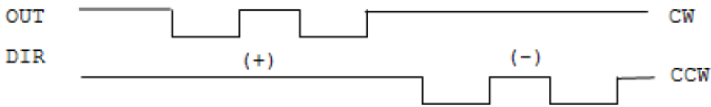


## Dual Pulse Output Mode (CW/CCW Mode)

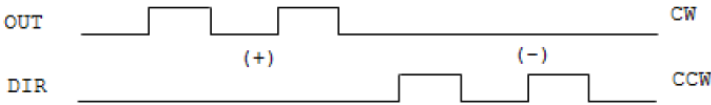
In this mode, the waveform of the OUT and DIR pins represent CW (clockwise) and CCW (counter clockwise) pulse output respectively. The numbers of pulse represent distance in pulse. The frequency of the pulse represents speed in pulse per second.

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.

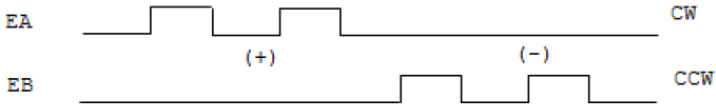
### 3.4.2 Pulse feedback input interface

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)

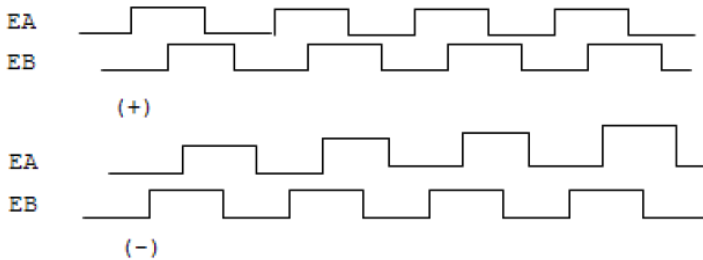


The pattern of pulses in this mode is the same as the Dual Pulse Output Mode (CW/CCW Mode) portion in the Pulse Command Output section except that the input pins are EA and EB.

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.

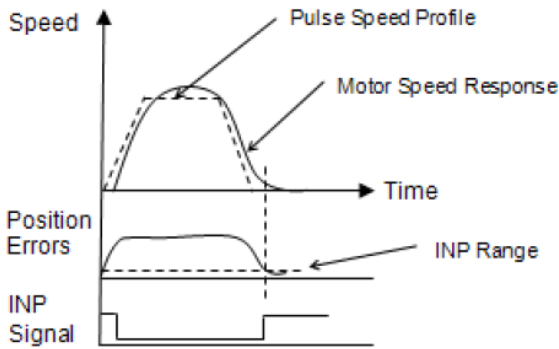


The index input (EZ) signal is as the zero reference in linear or rotary encoder. The EZ can be used to define the mechanical zero position of the mechanism. The logic of signal must also be set correctly to get correct result.

### 3.4.3 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.





### **3.4.4 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 motor driver's manual about the 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.

### **3.4.5 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 large the error counter value is, the faster the motor speed command will be set. If the error counter is zero, it means that 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

### **3.4.6 Servo ON/OFF switch**

The servo on/off switch is a general digital output signal on motion controller. We define it as SVON pin on the connector. It can be used for switching motor driver's controlling state. Once it is turned on, the motor will be held because the control loop of driver is active. Be careful that when the axis is vertically installed and the servo signal is turned off, the axis will be in uncontrolled state. It could fall on the ground. Some situations like servo alarm and emergency signal ON will result in the same trouble.

### **3.4.7 Servo Ready Signal**

The servo ready signal is a general digital input on motion controller. It has no relative purpose to motion controller. Users can connect this signal to motor driver's RDY signal to check if the motor driver is in ready state. It lets users to check something like the motor driver's power has been inputted or not. Or users can connect this pin as a general input for other purpose. It doesn't affect motion control.

### **3.4.8 Servo alarm reset switch**

The servo driver will raise an alarm signal if there is something wrong inside the servo driver. Some alarm situations like servo motor over current, over speed, over loading and so on. Power reset can clear the alarm status but users usually don't want to power off the servo motor when operating. There is one pin from servo driver for users to reset the alarm status. Our motion controller provides one general output pin for each axis. Users can use this pin for resetting servo alarm status.

### **3.4.9 Gain Selection switch**

This is connected to the gain select input (GAIN) terminal on the servo amplifier, and it is used to select the 2nd level of gain.

### **3.5 Mechanical switch interface**

MNET module provides some dedicated input pins for mechanical switches like original switch (ORG), plus and minus end-limit switch ( $\pm$ EL), slow down switch (SD), emergency stop input (EMG). These switches' response time is very short, only a few LSI clock times. There is no real-time problem when using these signals. All functions are done by MNET module. The software can just do nothing and only need to wait the results. For detailed information, please refer to MNET module's user guide.

#### **3.5.1 Original or Home (Zero position) signal**

Our controller provides one original or home signal for the motion 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. Please refer to home mode section for details.

#### **3.5.2 End-Limit switch signal**

The end-limit switches are usually installed on both ending sides of one motion axis. We must install plus EL at the positive position of the axis and minus EL at the negative position of the axis. These two signals are for safety reason. If they are installed reversely, the protection will be invalid. Once the motor's moving part touches one of the end-limit signals, the MNET module will stop sending pulses and output an ERC signal. It can prevent machine crash when miss operation.

### **3.5.3 Slow down switch signal**

The slow down signal is used to force the command pulse to decelerate to the starting velocity when it is active. This signal is used to protect a mechanical moving part under high speed movement toward the mechanism's limit. The SD signal is effective for both plus and minus directions.

### **3.5.4 Emergency stop input**

MNET module specifies an input signal to identify emergency situation. Once the input is turned on, MNET module will stop this axis motion immediately to prevent machine's damage. Usually, users can connect an emergency stop button to this input on their machine.

## 3.6 The Counters

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

They are described in this section.

- ▶ Command position counter: counts the number of output pulses
- ▶ Feedback position counter: counts the number of input pulses
- ▶ Position error counter: counts the error between command and feedback pulse numbers.
- ▶ Target position recorder: A software-maintained

### 3.6.1 Command position counter

The command position counter is a 28-bit binary up/down counter.

Its input source is the output pulses from the MNET module. It provides the information of the current command position. It is useful for debugging the motion system.

Our motion system is an open loop type. The motor driver receives pulses from MNET module 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 seen 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.

### **3.6.2 Feedback position counter**

The feedback position counter is a 28-bit binary up/down counter.

Its input source is the input pulses from the EA/EB pins. It counts the motor position from motor's encoder output.

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

### **3.6.3 Command and Feedback error counter**

The command and feedback error counter is used to calculate the error between the command position and the feedback position.

The value is calculated from command subtracting feedback position.

If the ratio between command and feedback is not 1:1, the error counter is meaningless. This counter is a 16-bit binary up/down counter.

### **3.6.4 Target position recorder**

The target position recorder is used for providing target position information. It is used in continuous motion because motion controller need to know the previous motion command's target position and current motion command's target position in order to calculate relative pulses of current command.

## 3.7 The Comparators

There are 3 counter comparators of each axis. Each comparator has dedicated functions. They are:

1. Positive soft end-limit comparator to command counter
2. Negative soft end-limit comparator to command counter
3. Command and feedback error counter comparator

### 3.7.1 Soft end-limit comparators

There are two comparators for end-limit function of each axis. We call them for the soft end-limit comparators. One is for plus or positive end-limit and the other is for minus or negative end-limit. The end-limit is to prevent machine crash when over traveling. We can use the soft limit instead of a real end-limit switch. Notice that these two comparators only compare the command position counter. Once the command position is over the limited set inside the positive or negative comparators, it will stop moving as it touches the end-limit switch.

### 3.7.2 Command and feedback error counter comparators

This comparator is only for command and feedback counter error.

Users can use this comparator to check if the error is too big. It can be set a action when this condition is met. The actions include generating interrupt, immediately stop, and deceleration to stop.



## 3.8 Other Motion Functions

We provide many other functions on MNET motion module. Such as backlash compensation, vibration restriction, speed profile calculation and so on. The following sections will

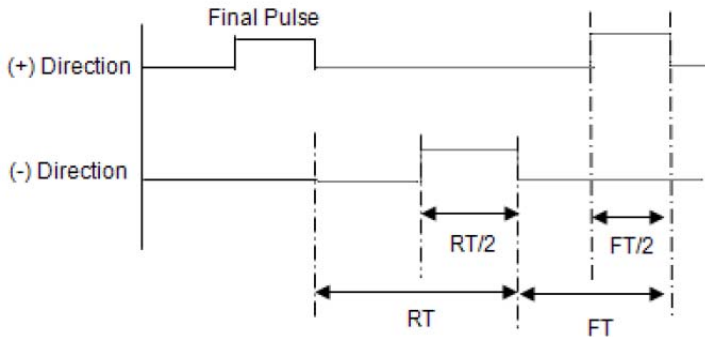
### 3.8.1 Backlash compensation and slip corrections

The motion controller has backlash and slip correction functions.

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

### 3.8.2 Vibration restriction function

The method of vibration restriction of the motion controller is by adding one pulse of reverse direction and then one pulse of forward direction shortly after completing a motion command. The timing of these two dummy pulses are shown below: (RT is reverse time and FT is forward time).

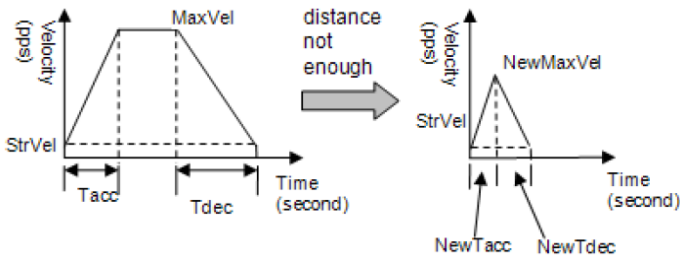


### 3.8.3 Speed profile calculation function

Our motion function needs several speed parameters from users.

Some parameters are conflict in speed profile. For example, if users input a very fast speed profile and a very short distance to motion function, the speed profile is not exist for these parameters.

At this situation, motion library will keep the acceleration and deceleration rate. It tries to lower the maximum speed from users automatically to reform a speed profile feasible. The following diagram shows this concept. Our motion library has a series of functions to know the actual speed profile of the command from users.



## 4 DB-8153 Motionnet Master Utility

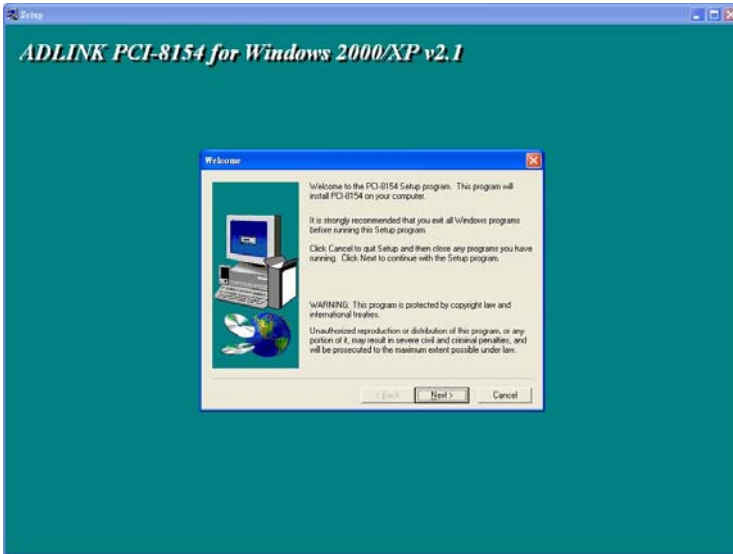
After installing master controller and slave modules, users can install the PCI-8154/58 driver and use Motionnet Master utility to test and debug the system. This utility provides a user-friendly interface for customers. Users can easily test the related motion control. We strongly suggest users can use this utility before implementing the whole system.

Note: LinkMaster is only available for Windows 2k/XP with screen resolution higher than 800×600.

## 4.1 Software Installation

You can install the drivers from the ADLINK All-in-One CD that comes with the package or you may download the drivers from the ADLINK website. The latest driver version are available from the website.

1. Find the SETUP.exe and execute the file.
2. The following screen will appear.



**Figure 4-1: PCI-8154/58 Installation Page**

Follow the instruction to complete installation.

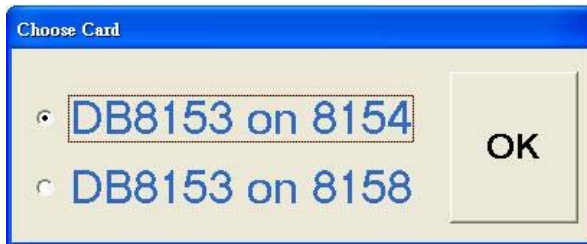
3. After complete the installation processes, you have to restart Windows system so that the PCI-8154/58 drivers could work normally.

## 4.2 ADLINK DB8153 Motionnet Master Utility

### 4.2.1 Select the type of carrier card

Use this utility to test MNET system and slave modules. After running the ADLINK DB8153 Motionnet Master Utility, It will show a selection screen for choosing the card as shown below.

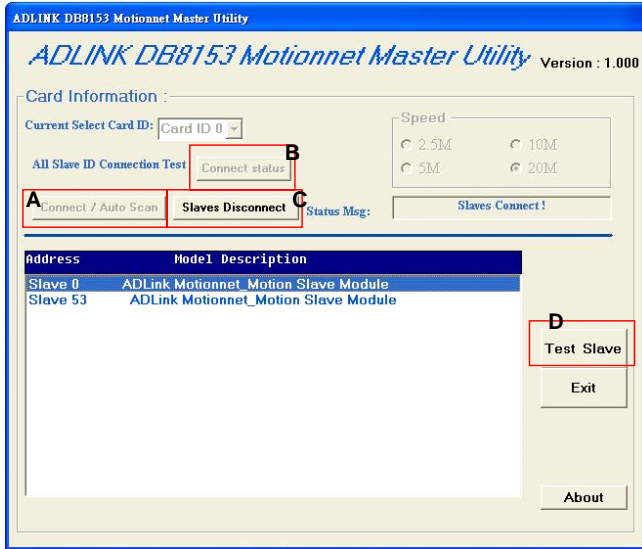
After driver installation, you can find the utility placed in “Start” > “PCI-8154”. Click on the “ADLINK DB8153 Motionnet Master Utility” and you will see the main menu as follows.



**Figure 4-2: Carrier Selection Dialog**

## 4.2.2 Run Motionnet Master Utility

Enter the operation window to run this system. The operation items in the “ADLINK DB8153 Motionnet Master Utility” are described as follows.

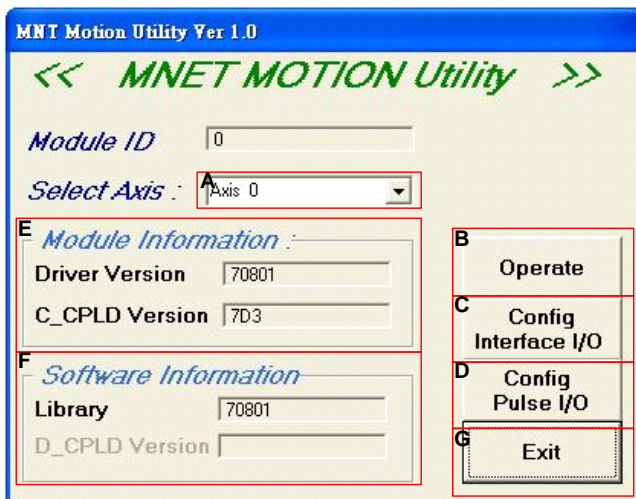


- A. **“Connect / Auto Scan”**: When this button is pressed, the utility will connect the master card and slave modules, then scan all the slave modules. This utility will show all the slave module’s attributes (include the slave id and slave type) on the screen.
- B. **“Connect status”**: Press the “Connect status” button to enter the “Slave Live Scan” window.
- C. **“Slaves Disconnect”**: When this button is pressed, the utility will disconnect the master card and slave modules.
- D. **“Test Slave”**: When the **“Connect / Auto Scan”** button is pressed, all slave modules in the **“Current Select Set ID”** of the **“Current Select Card ID”** will display in the screen. Now user can select a slave module in the screen, then press the **“Test Slave”** button to enter the Slave testing window (The following is a testing utility for MNET motion module)

## 4.2.3 MNET motion Utility

### Main Menu

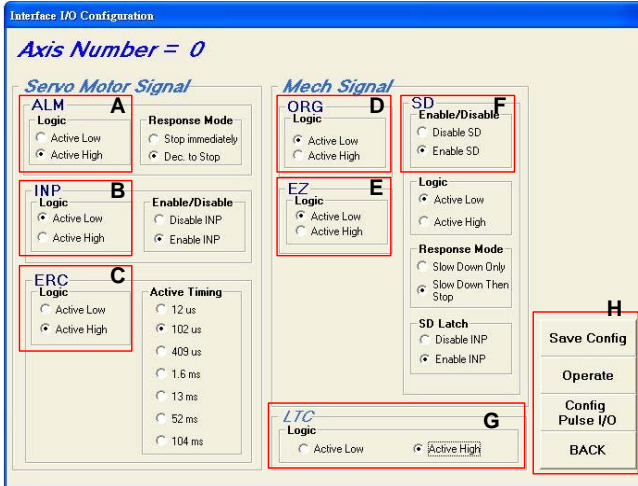
The main menu is shown as below. It is used to:



- A. Select Axis
- B. Go to Operate menus
- C. Go to Interface I/O configuration menus
- D. Go to Config Pulse I/O menus
- E. Show Module information
- F. Show Software information
- G. Exit

## Interface I/O Configuration Menu

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



**Interface I/O Configuration**

*Axis Number = 0*

**Servo Motor Signal**

**ALM** **A**

**Logic**

☐ Active Low

☒ Active High

**Response Mode**

☐ Stop immediately

☒ Dec. to Stop

**INP** **B**

**Logic**

☒ Active Low

☐ Active High

**Enable/Disable**

☐ Disable INP

☒ Enable INP

**ERC** **C**

**Logic**

☐ Active Low

☒ Active High

**Active Timing**

☐ 12 us

☐ 102 us

☐ 409 us

☐ 1.6 ms

☐ 13 ms

☐ 52 ms

☐ 104 ms

**Mech Signal**

**ORG** **D**

**Logic**

☒ Active Low

☐ Active High

**EZ** **E**

**Logic**

☒ Active Low

☐ Active High

**SD** **F**

**Enable/Disable**

☐ Disable SD

☒ Enable SD

**Logic**

☒ Active Low

☐ Active High

**Response Mode**

☐ Slow Down Only

☒ Slow Down Then Stop

**SD Latch**

☐ Disable INP

☒ Enable INP

**LTC** **G**

**Logic**

☐ Active Low

☒ Active High

**Save Config** **H**

**Operate**

**Config Pulse I/O**

**BACK**

- A. ALM Logic and Response mode: Select logic and response modes of ALM signal. The related function call is `_815x_db53_M_set_alm()`.
- B. INP Logic and Enable/Disable selection: Select logic, and Enable/ Disable the INP signal. The related function call is `_815x_db53_M_set_inp()`
- C. ERC Logic and Active timing: Select the Logic and Active timing of the ERC signal. The related function call is `_815x_db53_M_set_erc()`.
- D. ORG Logic: Select the logic of the ORG signal. The related function call is `_815x_db53_M_set_home_config()`.
- E. EZ Logic: Select the logic of the EZ signal. The related function call is `_815x_db53_M_set_home_config()`.
- F. SD Configuration: Configure the SD signal. The related function call is `_815x_db53_M_set_sd()`.
- G. LTC Logic: Select the logic of the LTC signal. The related function call is `_815x_db53_M_set_ltc_logic()`.

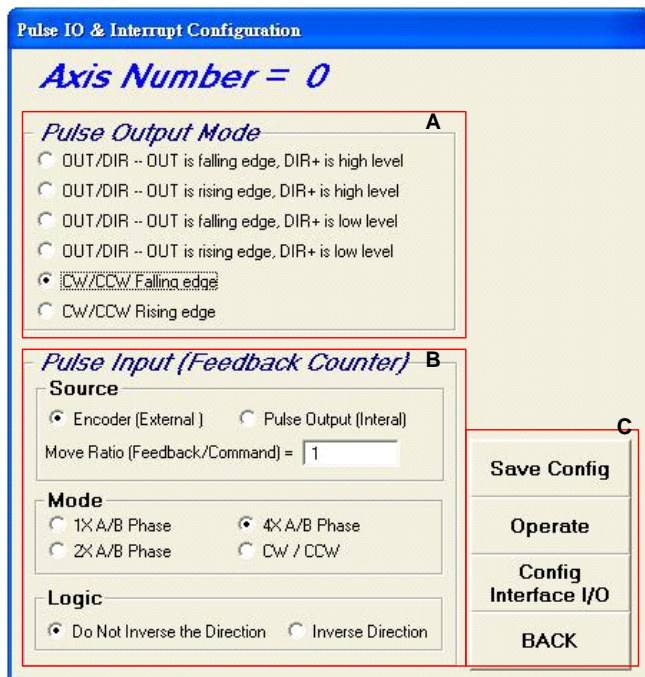


#### H. Buttons:

- ▷ **Save Config:** Save current configuration to MNETMO.cfg.
- ▷ **Operate:** Go to the operation menu
- ▷ **Config Pulse:** Go to the Pulse IO Configuration menu
- ▷ **Back:** Return to the main menu.

## Pulse IO Configuration Menu

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



**Pulse IO & Interrupt Configuration**

*Axis Number = 0*

**A Pulse Output Mode**

- ☐ OUT/DIR -- OUT is falling edge, DIR+ is high level
- ☐ OUT/DIR -- OUT is rising edge, DIR+ is high level
- ☐ OUT/DIR -- OUT is falling edge, DIR+ is low level
- ☐ OUT/DIR -- OUT is rising edge, DIR+ is low level
- ☒ CW/CCW Falling edge
- ☐ CW/CCW Rising edge

**B Pulse Input (Feedback Counter)**

**Source**

- ☒ Encoder (External)
- ☐ Pulse Output (Internal)

Move Ratio (Feedback/Command) =

**Mode**

- ☐ 1X A/B Phase
- ☐ 2X A/B Phase
- ☒ 4X A/B Phase
- ☐ CW / CCW

**Logic**

- ☒ Do Not Inverse the Direction
- ☐ Inverse Direction

**C**

Save Config

Operate

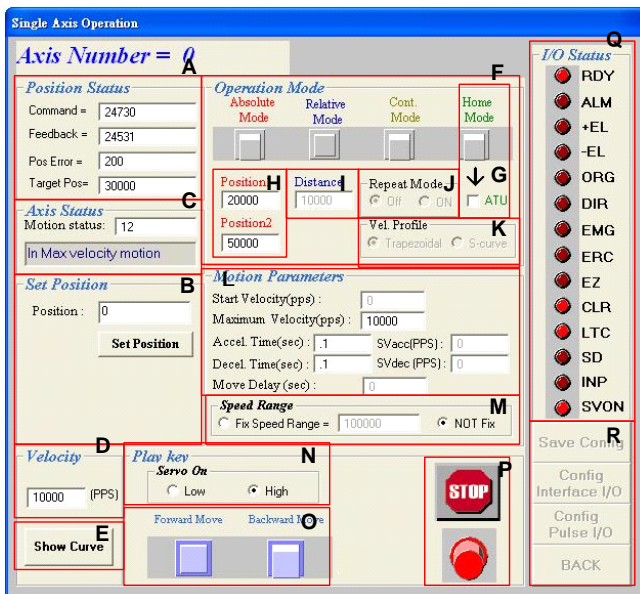
Config Interface I/O

BACK

- A. Pulse Output Mode: Select the output mode of the pulse signal (OUT/ DIR). The related function call is `_815x_db53_M_set_pls_outmode()`.
- B. Pulse Input: Sets the configurations of the Pulse input signal(EA/EB). The related function calls are `_815x_db53_M_set_pls_iptmode()`, `_815x_db53_M_set_feedback_src()`.
- C. Buttons:
- ▷ **Save Config:** Save current configuration to MNETMO.cfg.
  - ▷ **Operate:** Go to the operation menu
  - ▷ **Config Pulse:** Go to the Pulse IO Configuration menu
  - ▷ **Back:** Return to the main menu.

## Operation Menu

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



The screenshot shows the 'Single Axis Operation' window. It includes sections for Position Status (Command, Feedback, Pos Error, Target Pos), Axis Status (Motion status, In Max velocity motion), Set Position (Position, Set Position button), Velocity (10000 PPS), Play key (Servo On, Low/High), Forward/Backward Move buttons, and a STOP button. The Operation Mode section has tabs for Absolute, Relative, Cont. Mode, and Home Mode. The Motion Parameters section includes Start Velocity, Maximum Velocity, Accel. Time, Decel. Time, Move Delay, and Speed Range. The I/O Status section on the right lists various status indicators like RDY, ALM, EL, ORG, DIR, EMG, ERC, EZ, CLR, LTC, SD, INP, and SVON. The bottom right has buttons for Save Config, Config Interface I/O, Config Pulse I/O, and BACK.

### A. Position:

- ▷ **Command:** displays the value of the command counter. The related function is `_815x_db53_M_get_command()`.
- ▷ **Feedback:** displays the value of the feedback position counter. The related function is `_815x_db53_M_get_position()`.
- ▷ **Pos Error:** displays the value of the position error counter. The related function is `_815x_db53_M_get_error_counter()`.
- ▷ **Target Pos:** displays the value of the target position recorder. The related function is `_815x_db53_M_get_target_pos()`.

- B. Position Reset: clicking this button will set all positioning counters to a specified value.

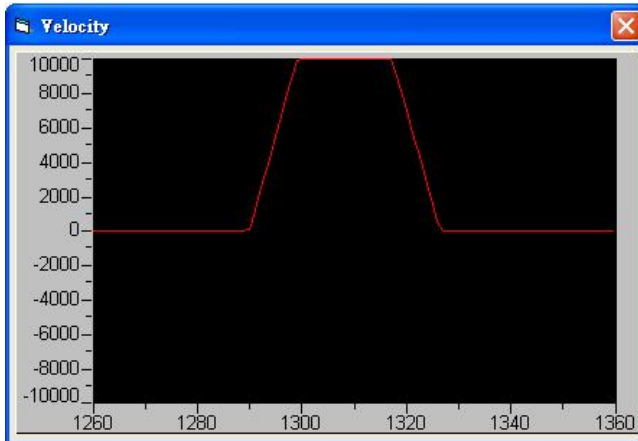
The related functions are:

```
_815x_db53_M_set_position ()
_815x_db53_M_set_command ()
_815x_db53_M_reset_error_counter ()
_815x_db53_M_reset_target_pos ()
```

- C. Axis Status: Displays the returned value of the \_815x\_db53\_M\_motion\_done function.

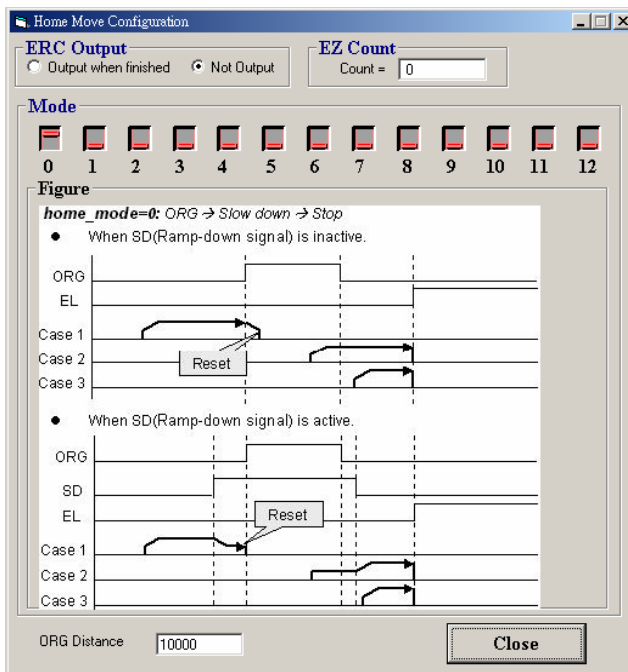
- D. Velocity: The absolute value of velocity in units of PPS. The related function is \_815x\_db53\_M\_get\_current\_speed().

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



## F. Operation Mode: Select operation mode.

- **Absolute Mode:** “Position1” and “position2” will be used as absolution target positions for motion. The related functions are: `_815x_db53_M_start_ta_move()`, `_815x_db53_M_start_sa_move()`
- **Relative Mode:** “Distance” will be used as relative displacement for motion. The related functions are: `_815x_db53_M_start_tr_move()`, `_8154_db53_M_start_sr_move()`.
- **Cont. Move:** Velocity motion mode. The related functions are: `_815x_db53_M_tv_move()`, `_815x_db53_M_sv_move()`.
- **Home Mode:** Home return motion. Clicking this button will invoke the home move configuration window. The related function is `_815x_db53_M_set_home_config()`. If the check box “ATU” is checked, it will execute auto homing when motion starts.



- ▷ **ERC Output:** Select if the ERC signal will be sent when home move completes.
  - ▷ **EZ Count:** Set the EZ count number, which is effective on certain home return modes.
  - ▷ **Mode:** Select the home return mode. There are 13 modes available.
  - ▷ **Home Mode figure:** The figure shown explains the actions of the individual home modes.
  - ▷ **Close:** Click this button close this window.
  - ▷ **ORG Distance:** The length during ORG is ON
- G. ATU: Enable/Disable ATU.
- H. Position: Set the absolute position for “Absolute Mode.” It is only effective when “Absolute Mode” is selected.
- I. Distance: Set the relative distance for “Relative Mode.” It is only effective when “Relative Mode” is selected.
- J. Repeat Mode: When “On” is selected, the motion will become repeat mode (forward<->backward or position1<->position2). It is only effective when “Relative Mode” or “Absolute Mode” is selected.
- K. Vel. Profile: Select the velocity profile. Both Trapezoidal and S-Curve are available for “Absolute Mode,” “Relative Mode,” and “Cont. Move.”
- L. Motion Parameters: Set the parameters for single axis motion.
- ▷ **Start Velocity:** Set the start velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -100.0 is the same as 100.0. In “Cont. Move,” both the value and sign are effective. -100.0 means 100.0 in the minus direction.
  - ▷ **Maximum Velocity:** Set the maximum velocity of motion in units of PPS. In “Absolute Mode” or “Relative Mode,” only the value is effective. For example, -5000.0 is the same as 5000.0. In “Cont. Move,” both the value and sing is effective. -5000.0 means 5000.0 in the minus direction.

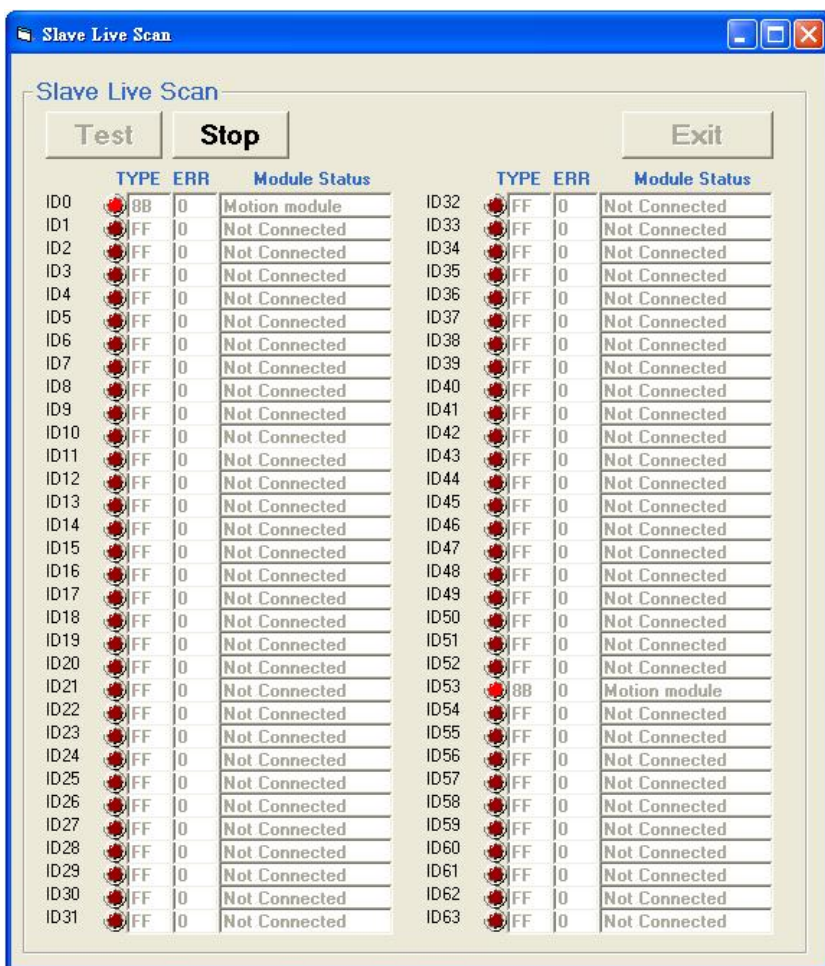
- ▷ **Accel. Time:** Set the acceleration time in units of second.
  - ▷ **Decel. Time:** Set the deceleration time in units of second.
  - ▷ **SVacc:** Set the S-curve range during acceleration in units of PPS.
  - ▷ **SVdec:** Set the S-curve range during deceleration in unit sof PPS.
  - ▷ **Move Delay:** This setting is effective only when repeat mode is set “On.” It will cause the DB8153 to delay for a specified time before it continues to the next motion.
- M. Speed Range: Set the max speed of motion. If “Not Fix” is selected, the “Maximum Speed” will automatically become the maximum speed range, which can not be exceeded by on-the-fly velocity change.
- N. Servo On: Set the SVON signal output status. The related function is `_815x_db53_M_set_servo()`.
- O. Play Keys
- ▷ Left play button: Clicking this button will cause the DB8153 start to outlet pulses according to previous setting.
  - ▷ In “Absolute Mode,” it causes the axis to move to position1.
  - ▷ In “Relative Mode,” it causes the axis to move forward.
  - ▷ In “Cont. Move,” it causes the axis to start to move according to the velocity setting.
  - ▷ Right play button: Clicking this button will cause the DB8153 start to outlet pulses according to previous setting.
  - ▷ In “Absolute Mode,” it causes the axis to move to position.
  - ▷ In “Relative Mode,” it causes the axis to move backwards.

- ▷ In “Cont. Move,” it causes the axis to start to move according to the velocity setting, but in the opposite direction.
- P. Stop Button: Clicking this button will cause the DB8153 to decelerate and stop. The deceleration time is defined in “Decel. Time.” The related function is `_815x_db53_M_sd_stop()`.
- Q. I/O Status: The status of motion I/O. Light-On means Active, while Light-Off indicates inactive. The related function is `_815x_db53_M_get_io_status()`.
- R. Buttons:
  - ▷ **Save Config:** Save current configuration to MNET-MOMC.cfg.
  - ▷ **Config Pulse:** Go to the Pulse IO Configuration menu
  - ▷ **Config Interface I/O:** Go to the Interface I/O Configuration Menu
  - ▷ **Back:** Return to the main menu.



## 4.2.4 Slave Live Scan

The utility will display status for all Motionnet slave modules.



**Operation Button:**

- ▶ **Test:** When users press this button, the information of those scanned slave modules will be shown on the screen.
- ▶ **Stop:** Stop scanning.
- ▶ **Exit:** Exit this form.

**Information of module:**

- ▶ 1. Type: 8B => Motion type; FF => Not connected
- ▶ 2. ERR: Communication error => 1; Communication ok => 0
- ▶ 3. Module Status: The module type

## 5 Function Library

This chapter describes the supporting software for the DB-8153 Motionnet system. 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, DB8153.h manually.

## 5.1 List of Functions

### System & Initialization, Section 5.3

Function Name	Description
_8154_db53_initial	Card initialization
_8154_db53_close	Card Close
_8154_db53_auto_start	Start to scan and automatically detect all the slave modules connected to DB8153 master card
_8154_db53_stop	Stop scanning the connected slave modules
_8154_db53_set_scan_condition	Set scanning conditions
_8154_db53_get_scan_condition	Get scanning conditions
_8154_db53_set_clock_rate	Set clock rate
_8154_db53_get_clock_rate	Set clock rate
_8154_db53_get_detected_modules_status	Get scanned module's information related to module type and slave id
_8154_db53_get_slave_type	Get slave type
_8154_db53_get_DBcpld_version	Get CPLD version of DB8153
_8154_db53_M_get_AxisNo_mapto_SlaveNo	Get slave id which is mapped to axis

## Pulse Input/Output Configuration, Section 5.4

Function Name	Description
_8154_db53_M_set_pls_outmode	Set pulse command output mode
_8154_db53_M_set_pls_iptmode	Set encoder input mode
_8154_db53_M_set_feedback_src	Set counter input source

## Velocity Mode Motion, Section 5.5

Function Name	Description
_8154_db53_M_tv_move	Accelerate an axis to a constant velocity with trapezoidal profile
_8154_db53_M_sv_move	Accelerate an axis to a constant velocity with S-curve profile
_8154_db53_M_sd_stop	Decelerate to stop
_8154_db53_M_emg_stop	Immediately stop
_8154_db53_M_get_current_speed	Get current speed(pulse/sec)
_8154_db53_M_set_max_override_speed	Set the maximum override speed

## Single Axis Position Mode, Section 5.6

Function Name	Description
_8154_db53_M_start_tr_move	Begin a relative trapezoidal profile move
_8154_db53_M_start_ta_move	Begin an absolute trapezoidal profile move
_8154_db53_M_start_sr_move	Begin a relative S-curve profile move
_8154_db53_M_start_sa_move	Begin an absolute S-curve profile move
_8154_db53_M_set_move_ratio	Set the ratio of command pulse and feedback pulse.

## Home Return Mode, Section 5.7

Function Name	Description
_8154_db53_M_set_home_config	Set the home/index logic configuration
_8154_db53_M_home_move	Begin a home return action
_8154_db53_M_home_search	Perform an auto search home

## Motion Status, Section 5.8

Function Name	Description
_8154_db53_M_motion_done	Return the motion status

## Motion Interface I/O, Section 5.9

Function Name	Description
_8154_db53_M_set_servo	Set On-Off state of SVON signal
_8154_db53_M_set_clr_mode	Set CLR signal's mode
_8154_db53_M_set_inp	Set INP signal's logic and operating mode
_8154_db53_M_set_alm	Set ALM signal's logic and operating mode
_8154_db53_M_set_erc	Set ERC signal's logic and timing
_8154_db53_M_set_erc_out	Output an ERC signal
_8154_db53_M_clr_erc	Clear the ERC signal
_8154_db53_M_set_sd	Set SD signal's logic and operating mode
_8154_db53_M_enable_sd	Enable SD signal
_8154_db53_M_set_limit_mode	Set EL operating mode
_8154_db53_M_get_io_status	Get all the motion I/O status of DB8153

## Position Control and Counters, Section 5.10

Function Name	Description
_8154_db53_M_get_position	Get the value of the feedback position counter
_8154_db53_M_set_position	Set the feedback position counter
_8154_db53_M_get_command	Get the value of the command position counter
_8154_db53_M_set_command	Set the command position counter
_8154_db53_M_get_error_counter	Get the value of the position error counter
_8154_db53_M_reset_error_counter	Reset the position error counter
_8154_db53_M_get_target_pos	Get the value of the target position recorder
_8154_db53_M_reset_target_pos	Reset target position recorder
_8154_db53_M_get_res_distance	Get remaining pulses accumulated from motions
_8154_db53_M_set_res_distance	Set remaining pulses record

## Position Compare and Latch, Section 5.11

Function Name	Description
_8154_db53_M_set_error_comparator	Set the error comparator
_8154_db53_M_set_trigger_comparator	Set the trigger comparator
_8154_db53_M_set_latch_source	Set the latch timing for a counter
_8154_db53_M_get_latch_data	Get the latch data

## Soft Limit, Section 5.12

Function Name	Description
_8154_db53_M_disable_soft_limit	Disable soft limit function
_8154_db53_M_enable_soft_limit	Enable soft limit function
_8154_db53_M_set_soft_limit	Set the soft limits

## Backlash Compensation/Vibration Suppression, Section 5.13

Function Name	Description
_8154_db53_M_backlash_comp	Set backlash corrective pulse for compensation
_8154_db53_M_suppress_vibration	Set suppress vibration idle pulse counts
_8154_db53_M_set_fa_speed	Set FA speed for home mode

## Speed Profile Calculation, Section 5.14

Function Name	Description
_8154_db53_M_get_tr_move_profile	Get relative trapezoidal speed profile
_8154_db53_M_get_ta_move_profile	Get absolute trapezoidal speed profile
_8154_db53_M_get_sr_move_profile	Get relative S-curve speed profile
_8154_db53_M_get_sa_move_profile	Get absolute S-curve speed profile



## 5.2 C/C++ Programming Library

This section details all the functions. The function prototypes and some common data types are decelerated in **DB8153.h**. We suggest you to use these data types in your application programs. The following table shows the data type names and their range.

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

The naming rule is defined by ADLINK. All function calls have the same prefix as **\_8154\_db53\_**. If they belong to system level purpose, the function will be as follows:

**\_8154\_db53\_{action\_name}**

e.g. **\_8154\_db53\_initial()**.

If they belong to motion control modules purpose, the function will be as follows.

**\_8154\_db53\_M\_{action\_name}**

e.g. **\_8154\_db53\_M\_start\_tr\_move()**.

## 5.3 System & Initialization

### @ Name

`_8154_db53_initial` – DB8153 board initialization

`_8154_db53_close` – Release all resource occupied by DB8153 board

`_8154_db53_auto_start` –Start to scan and automatically detect all the slave modules connected to DB8153 card

`_8154_db53_stop` –Stop scanning the connected slave modules

`_8154_db53_set_scan_condition` – Set scanning conditions

`_8154_db53_get_scan_condition` – Get scanning conditions

`_8154_db53_set_clock_rate` – Set clock rate

`_8154_db53_get_clock_rate` – Get clock rate

`_8154_db53_get_detected_modules_status` – Get scanned slave's information related to module type and slave id

`_8154_db53_get_slave_type` – Get the slave type of the slave module

`_8154_db53_get_DBCpld_version` – Get CPLD version of DB8153

`_8154_db53_M_get_AxisNo_mapto_SlaveNo` – Get slave id which is mapped to axis

### @ Description

`_8154_db53_initial:`

Initialize the hardware and software states of the `8154_DB8153` MNET master card. Before calling this function, users must call `_8154_initial` first to get resource of DB8153. Users can check the return code of this function to know if the initialization is successful or not.

#### \_8154\_db53\_close:

This function is to release the resource occupied by the **8154\_DB8153** MNET master card. When terminating the program, do not forget to call this function to release all the resource occupied by **8154\_DB8153** MNET card.

#### \_8154\_db53\_auto\_start:

This function is used to automatically detect the total connected slave modules. Every master controller can connect up to 64 slave indexes. As a result, it will scan from 0 to 63.

#### \_8154\_db53\_stop:

This function is used to stop scanning the connected slave modules.

#### \_8154\_db53\_set\_scan\_condition:

This function is used to assign the scan rate (2.5/5/10/20M). And this function needs to set up between the function **\_8154\_db53\_initial** and **\_8154\_db53\_auto\_start**.

#### \_8154\_db53\_get\_scan\_condition:

By this function, User can get the settings of communication types and scan rate which are set by "**\_8154\_db53\_set\_scan\_condition**".

#### \_8154\_db53\_set\_clock\_rate:

This function is used to assign the clock rate(40/80Mhz). And this function needs to set up between the function **\_8154\_db53\_initial** and **\_8154\_db53\_auto\_start**.

#### **`_8154_db53_get_clock_rate:`**

By this function, User can get the settings of clock rate which are set by “`_8154_db53_set_clock_rate`”.

#### **`_8154_db53_get_detected_modules_status:`**

After scanning by set `_8154_db53_auto_start`, user can use this function to get information related to motion type and slave id.

#### **`_8154_db53_get_slave_type:`**

This function is used to get slave type (motion or others) of the specified slave module.

#### **`_8154_db53_get_DBCpld_version:`**

This function is used to get CPLD version of DB8153.

#### **`_8154_db53_M_get_AxisNo_mapto_SlaveNo:`**

The function is used to get slave id which is mapped to scanned axis.

### **@ Syntax**

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

```
I16 _8154_db53_initial(I16 Card_ID);  
I16 _8154_db53_close(I16 Card_ID);  
I16 _8154_db53_auto_start(I16 Card_ID);  
I16 _8154_db53_stop(I16 Card_ID);  
I16 _8154_db53_set_scan_condition(I16 Card_ID,  
    I16 transfer_rate);  
I16 _8154_db53_get_scan_condition(I16 Card_ID,  
    I16 *transfer_rate);  
I16 _8154_db53_set_clock_rate(I16 Card_ID, I16  
    clock_rate);  
I16 _8154_db53_get_clock_rate(I16 Card_ID, I16  
    *clock_rate);
```

```
I16 _8154_db53_get_detected_modules_status(I16
Card_ID, I16 *Maximum_Satellite, I16
*Detect_Module_No, I16
*Detect_Module_Type);
I16 _8154_db53_get_slave_type(I16 Card_ID, I16
slave_No, I16 *type_data);
I16 _8154_db53_get_DBCpld_version(I16 Card_ID,
I16 *cpld_version);
I16 _8154_db53_M_get_AxisNo_mapto_SlaveNo(I16
Card_ID, I16 AxisNo, I16 *Slave_No);
```

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

```
B_8154_db53_initial (ByVal card_ID As Integer) As
Integer
B_8154_db53_close (ByVal card_ID As Integer) As
Integer
B_8154_db53_auto_start (ByVal card_ID As Integer)
As Integer
B_8154_db53_stop (ByVal card_ID As Integer) As
Integer
B_8154_db53_set_scan_condition(ByVal card_ID As
Integer, ByVal transfer_rate As Integer) As
Integer
B_8154_db53_get_scan_condition((ByVal card_ID As
Integer, ByRef transfer_rate As Integer) As
Integer
B_8154_db53_set_clock_rate(ByVal card_ID As
Integer, ByVal clock_rate As Integer) As
Integer
B_8154_db53_get_clock_rate(ByVal card_ID As
Integer, ByRef clock_rate As Integer) As
Integer
B_8154_db53_get_detected_modules_status(ByVal
card_ID As Integer, ByRef Maximum_Satellite
As Integer, ByRef Detect_Module_No As
Integer, ByRef Detect_Module_Type As
Integer) As Integer
B_8154_db53_get_slave_type(ByVal card_ID As
Integer, ByVal slave_No As Integer, ByRef
type_data As Integer) As Integer
B_8154_db53_get_DBCpld_version(ByVal card_ID As
Integer, ByRef cpld_version As Integer) As
Integer
```

```
B_8154_db53_M_get_AxisNo_mapto_SlaveNo(ByVal  
    card_ID As Integer, ByVal AxisNo As Integer,  
    ByRef Slave_No As Integer) As Integer
```

## @ Argument

**card\_ID:** Specify the MNET master card index. Normally, the board index sequence would be decided by the system. The index is from 0.

**slave\_No:** Specify the slave module with slave index which want to perform this function. The valid value is from 0 to 63.

**transfer\_rate:** transfer rate setting

- ▶ 0: 2.5M
- ▶ 1: 5M
- ▶ 2: 10M
- ▶ 3: 20M(Default)

**clock\_rate:** transfer rate setting

- ▶ 0: 40Mhz
- ▶ 1: 80Mhz(Default)

**\*Maximum\_Satellite:** Total numbers of scanned slave modules.

**\*Detect\_Module\_No:** The module's ID of scanned physical order. Use 64 elements of array "Array[64]" to get information.

- ▶ 0-63: slave ID value
- ▶ 0xFF: Not Connected

**\*Detect\_Module\_Type:** The module's type of scanned physical order. Use 64 elements of array "Array[64]" to get information.

- ▶ 0x8b: Motion
- ▶ 0xFF: Not Connect

**\*type\_data:** Module type

- ▶ 0x8b: Motion
- ▶ 0xFF: Not Connected

**\*cpld\_version:** CPLD version on DB8153

**AxisNo:** Axis number designated to configure the pulse input/output.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**\*Slave\_No:** the slave id which is mapped to axis

- ▶ 0 ~ 63: Slave id
- ▶ 0xFF: Not connected

## 5.4 Pulse Input/Output Configuration

### @ Name

`_8154_db53_M_set_pls_iptmode` – Set the configuration for feedback pulse input.

`_8154_db53_M_set_pls_outmode` – Set the configuration for pulse command output.

`_8154_db53_M_set_feedback_src` – Enable/Disable the external feedback pulse input

### @ Description

`_8154_db53_M_set_pls_iptmode:`

Configure the input modes of external feedback pulses. There are 4 types for feedback pulse input. Note that this function makes sense only when the Src parameter in `_8154_db53_M_set_feedback_src()` function is enabled.

`_8154_db53_M_set_pls_outmode:`

Configure the output modes of command pulses. There are 8 modes for command pulse output.

`_8154_db53_M_set_feedback_src:`

If external encoder feedback is available in the system, set the Src parameter in this function to an Enabled state. Then, the internal 28-bit up/down counter will count according to the configuration of the `_8154_db53_M_set_pls_iptmode()` function. Else, the counter will count the command pulse output.



## @ Syntax

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

```
I16 _8154_db53_M_set_pls_iptmode(I16 Card_ID, I16
    AxisNo, I16 pls_iptmode, I16 pls_logic);
I16 _8154_db53_M_set_pls_outmode(I16 Card_ID, I16
    AxisNo, I16 pls_outmode);
I16 _8154_db53_M_set_feedback_src(I16 Card_ID,
    I16 AxisNo, I16 Src);
```

### Visual Basic6 (Windows 2000/XP)

```
B_8154_db53_M_set_pls_iptmode(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    pls_iptmode As Integer, ByVal pls_logic As
    Integer) As Integer
B_8154_db53_M_set_pls_outmode(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    pls_outmode As Integer) As Integer
B_8154_db53_M_set_feedback_src(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal Src
    As Integer) As Integer
```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to configure the pulse input/output.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**pls\_ipmode:** Encoder feedback pulse input mode setting (EA/EB signals).

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

**pls\_logic:** Logic of encoder feedback pulse.

Value	Meaning
0	Not inverse direction
1	inverse direction

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

Value meaning					
Value	Type	Positive Direction		Negative Direction	
0	OUT/DIR				
1	OUT/DIR				
2	OUT/DIR				
3	OUT/DIR				
4	CW / CCW				
5	CW / CCW				
6	AB	OUT DIR	OUT DIR	OUT DIR	OUT DIR
7	AB	OUT DIR	OUT DIR	OUT DIR	OUT DIR

**src:** Counter source

Value	Meaning
0	External signal feedback
1	Command pulse
2	PA/PB

## 5.5 Velocity Mode Motion

### @ Name

\_8154\_db53\_M\_tv\_move – Accelerate an axis to a constant velocity with trapezoidal profile

\_8154\_db53\_M\_sv\_move – Accelerate an axis to a constant velocity with S-curve profile

\_8154\_db53\_M\_emg\_stop – Immediately stop

\_8154\_db53\_M\_sd\_stop – Decelerate to stop

\_8154\_db53\_M\_get\_current\_speed – Get current speed

### @ Description

\_8154\_db53\_M\_tv\_move:

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

\_8154\_db53\_M\_sv\_move:

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

\_8154\_db53\_M\_emg\_stop:

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

### **\_8154\_db53\_M\_sd\_stop:**

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

### **\_8154\_db53\_M\_get\_current\_speed:**

This function is used to read the current pulse output rate (pulse/sec) of a specified axis. It is applicable in any time in any operation mode.

## **@ Syntax**

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

```
I16 _8154_db53_M_tv_move(I16 Card_ID, I16 AxisNo,
    F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8154_db53_M_sv_move(I16 Card_ID, I16 AxisNo,
    F64 StrVel, F64 MaxVel, F64 Tacc, F64
    SVacc);
I16 _8154_db53_M_emg_stop(I16 Card_ID, I16
    AxisNo);
I16 _8154_db53_M_sd_stop(I16 Card_ID, I16 AxisNo,
    F64 Tdec);
I16 _8154_db53_M_get_current_speed(I16 Card_ID,
    I16 AxisNo, F64 *speed);
```

### **Visual Basic6 (Windows 2000/XP)**

```
B_8154_db53_M_tv_move(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal StrVel As
    Double, ByVal MaxVel As Double, ByVal Tacc
    As Double) As Integer
B_8154_db53_M_sv_move (ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal StrVel As
    Double, ByVal MaxVel As Double, ByVal Tacc
    As Double, ByVal SVacc As Double) As Integer
B_8154_db53_M_emg_stop(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer) As Integer
```

```

B_8154_db53_M_sd_stop(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal Tdec As
    Double) As Integer
B_8154_db53_M_get_current_speed(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByRef
    Speed As Double) As Integer

```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**strVel:** Starting velocity in units of pulse per second

**MaxVel:** Maximum velocity in units of pulse per second

**Tacc:** Specified acceleration time in units of second

**svacc:** Specified velocity interval in which S-curve acceleration is performed.

► Note: SVacc = 0, for pure S-Curve

**Tdec:** specified deceleration time in units of second

**\*speed:** Variable to get current speed (pulse/sec).

## 5.6 Single Axis Position Mode

### @ Name

`_8154_db53_M_start_tr_move` – Begin a relative trapezoidal profile move

`_8154_db53_M_start_ta_move` – Begin an absolute trapezoidal profile move

`_8154_db53_M_start_sr_move` – Begin a relative S-curve profile move

`_8154_db53_M_start_sa_move` – Begin an absolute S-curve profile move

`_8154_db53_M_set_move_ratio` – Set the ration of command pulse and feedback pulse

### @ Description

#### General:

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

#### `_8154_db53_M_start_tr_move:`

This function causes the axis to accelerate form a starting velocity (StrVel), rotate at constant velocity (MaxVel), and decelerate to stop at the relative distance with trapezoidal profile. The acceleration (Tacc) and deceleration (Tdec) time is specified independently—it does not let the program wait for motion completion but immediately returns control to the program.

#### \_8154\_db53\_M\_start\_ta\_move:

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

#### \_8154\_db53\_M\_start\_sr\_move:

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

#### \_8154\_db53\_M\_start\_sa\_move:

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

#### \_8154\_db53\_M\_set\_move\_ratio:

This function configures scale factors for the specified axis. Usually, the axes only need scale factors if their mechanical resolutions are different. For example, if the resolution of feedback sensors is two times resolution of command pulse, then the parameter “move\_ratio” could be set as 2.

## @ Syntax

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

```

I16 _8154_db53_M_start_tr_move(I16 Card_ID, I16
    AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
    F64 Tacc, F64 Tdec);
I16 _8154_db53_M_start_ta_move(I16 Card_ID, I16
    AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64
    Tacc, F64 Tdec);
I16 _8154_db53_M_start_sr_move(I16 Card_ID, I16
    AxisNo, F64 Dist, F64 StrVel, F64 MaxVel,
    F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8154_db53_M_start_sa_move(I16 Card_ID, I16
    AxisNo, F64 Pos, F64 StrVel, F64 MaxVel, F64
    Tacc, F64 Tdec, F64 SVacc, F64 SVdec);
I16 _8154_db53_M_set_move_ratio(I16 Card_ID, I16
    AxisNo, F64 move_ratio);
I16 _8154_db53_M_set_max_override_speed(I16
    Card_ID, I16 AxisNo, F64 Ovrdspeed, I16
    Enable);
  
```

### Visual Basic6 (Windows 2000/XP)

```

B_8154_db53_M_start_tr_move(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal Dist
    As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
    ByVal Tdec As Double) As Integer
B_8154_db53_M_start_ta_move(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal Pos
    As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
    ByVal Tdec As Double) As Integer
B_8154_db53_M_start_sr_move(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal Dist
    As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
    ByVal Tdec As Double, ByVal SVacc As Double,
    ByVal SVdec As Double) As Integer
B_8154_db53_M_start_sa_move(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal Pos
    As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
  
```



```

ByVal Tdec As Double, ByVal SVacc As Double,
ByVal SVdec As Double) As Integer
B_8154_db53_M_set_move_ratio(ByVal Card_ID As
Integer, ByVal AxisNo As Integer, ByVal
move_ratio As Double) As Integer
B_8154_db53_M_set_max_override_speed(ByVal
Card_ID As Integer, ByVal AxisNo As Integer,
ByVal OvrSpeed As Double, ByVal Enable As
Integer) As Integer

```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

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

**Pos:** Specified absolute position to move (unit: pulse)

**strVel:** Starting velocity of a velocity profile in units of pulse per second

**MaxVel:** Maximum velocity in units of pulse per second

**Tacc:** Specified acceleration time in units of seconds

**Tdec:** Specified deceleration time in units of seconds

**svacc:** Specified velocity interval in which S-curve acceleration is performed.

- Note: SVacc = 0, for pure S-Curve. For more details, see section 2.4.4

**svdec**: specified velocity interval in which S-curve deceleration is performed.

- Note: SVdec = 0, for pure S-Curve. For more details, see section 4.2.4

**Move\_ratio**: ratio of (feedback resolution)/(command resolution)  
, should not be 0

## 5.7 Home Return Mode

### @ Name

`_8154_db53_M_set_home_config` – Set the configuration for home return move motion

`_8154_db53_M_home_move` – Perform a home return move

`_8154_db53_M_home_search` – Perform an auto search home

### @ Description

`_8154_db53_M_set_home_config` –

Configures the home return mode, origin(ORG) and index signal(EZ) logic, EZ count, and ERC output options for the `home_move()` function. Refer to section 3.3.3 for the setting of `home_mode` control.

`_8154_db53_M_home_move` –

This function will cause the axis to perform a home return move according to the `_8154_db53_M_set_home_config()` function settings. The direction of movement is determined by the sign of velocity parameter (MaxVel). Since the stopping condition of this function is determined by the `home_mode` setting, users should take care in selecting the initial moving direction. Users should also take care to handle conditions when the limit switch is touched or other conditions that are possible causing the axis to stop. For more detail discription, see section 3.3.3.

`_8154_db53_M_home_search` –

This function will cause the axis to perform a home-search move according to the `_8154_db53_M_set_home_config()` function settings. The direction of movement is determined by the sign of velocity parameter (MaxVel). Since the stopping condition of this function is determined by the `home_mode` setting, users should take care in selecting the initial moving direction. Users should also take care to handle conditions when the limit switch is

touched or other conditions that are possible causing the axis to stop. For more detail discription, see section 3.3.3.

## @ Syntax

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

```
I16 _8154_db53_M_set_home_config(I16 Card_ID, I16
    AxisNo, I16 home_mode, I16 org_logic, I16
    ez_logic, I16 ez_count, I16 erc_out);
I16 _8154_db53_M_home_move(I16 Card_ID, I16
    AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc);
I16 _8154_db53_M_home_search(I16 Card_ID, I16
    AxisNo, F64 StrVel, F64 MaxVel, F64 Tacc,
    F64 ORGOffset);
```

### Visual Basic (Windows 2000/XP)

```
B_8154_db53_M_set_home_config(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    home_mode As Integer, ByVal org_logic As
    Integer, ByVal ez_logic As Integer, ByVal
    ez_count As Integer, ByVal erc_out As
    Integer) As Integer
B_8154_db53_M_home_move(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal StrVel As
    Double, ByVal MaxVel As Double, ByVal Tacc
    As Double) As Integer
B_8154_db53_M_home_search(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    StrVel As Double, ByVal MaxVel As Double,
    ByVal Tacc As Double, ByVal ORGOffset As
    Double) As Integer
```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**home\_mode:** Stopping modes for home return, This value is between 0 to 12. Please refer to the operation theory section 3.3.3.

**org\_logic:** Action logic configuration for ORG

Value	Meaning
0	Active low
1	Active high

**ez\_logic:** Action logic configuration for EZ

Value	Meaning
0	Active low
1	Active high

**ez\_count:** 0-15 (Please refer to section 3.3.3.)

**erc\_out:** Set ERC output options.

Value	Meaning
0	No ERC out
1	ERC signal out when home-move finishes

**strVel:** Starting velocity of a velocity profile. (unit: pulse/sec)

**MaxVel:** Maximum velocity. (unit: pulse/sec)

**Tacc:** Specified acceleration time (Unit: sec)

**ORGOffset:** The escape pulse amounts when home search touches the ORG singal (Unit: pulse)

## 5.8 Motion Status

### @ Name

`_8154_db53_M_motion_done` – Get the motion status

### @ Description

`_8154_db53_M_motion_done`:

Get motion status of the db8153.

### @ Syntax

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

```
I16 _8154_db53_M_motion_done(I16 Card_ID, I16
    AxisNo , I16 *M_sts)
```

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

```
B_8154_db53_M_motion_done(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, M_sts As
    Integer) As Integer
```

### @ Argument

**Card\_ID**: Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo**: Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**\*M\_sts:** Motion status

0	Normal stopped condition
1	Reserved
2	Waiting for CSTA input
3	Reserved
4	Reserved
5	Waiting for a completion of ERC timer
6	Waiting for a completion of direction change timer
7	Correcting backlash
8	Wait PA/PB
9	At FA speed
10	At FL Speed
11	Accelerating
12	At FH Speed
13	Decelerating
14	Wait INP
15	Others(Controlling Start)
16	SALM
17	SPEL
18	SMEL
19	SEMG
20	SSTP
21	SERC



## 5.9 Motion Interface I/O

### @ Name

\_8154\_db53\_M\_set\_servo – Set the ON-OFF state of the SVON signal

\_8154\_db53\_M\_set\_clr\_mode – Set the mode of CLR signal

\_8154\_db53\_M\_set\_inp – Set the logic of INP signal and operating mode

\_8154\_db53\_M\_set\_alm – Set the logic of ALM signal and operating mode

\_8154\_db53\_M\_set\_erc – Set the logic of ERC signal and operating mode

\_8154\_db53\_M\_set\_erc\_out – Output an ERC signal

\_8154\_db53\_M\_clr\_erc – Clear the ERC signal

\_8154\_db53\_M\_set\_sd – Set the logic SD signal and operating mode

\_8154\_db53\_M\_enable\_sd – Enable SD signal

\_8154\_db53\_M\_set\_limit\_mode – Set PEL/MEL operating mode

\_8154\_db53\_M\_get\_io\_status – Get all the motion I/O statuses of each DB8153

### @ Description

\_8154\_db53\_M\_set\_servo:

You can set the ON-OFF state of the SVON signal with this function. The default value is 0(OFF), which means the SVON is open to GND.

\_8154\_db53\_M\_set\_clr\_mode

CLR inputed signal can reset specified counters(command, position and error counter). The reset action could be set by this function. The reset action mode has 4 types. For details refer to arguments description.

#### `_8154_db53_M_set_inp:`

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_db53_M_set_alm:`

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

#### `_8154_db53_M_set_erc:`

Users can set the logic and on time of the ERC with this function. It also can set the pulser width of ERC signal.

#### `_8154_db53_M_set_erc_out:`

This function is used to output the ERC signal manually.

#### `_8154_db53_M_clr_erc:`

This function is used to reset the output when the ERC signal output is specified to a level type output.

#### `_8154_db53_M_set_sd:`

Set the active logic, latch control, and operating mode of the SD signal input from a mechanical system. Users can select whether they want to enable this function by `_8154_db53_M_enable_sd`. It is disabled by default

#### `_8154_db53_M_enable_sd:`

Enable the SD signal input. Default setting is default.

#### `_8154_db53_M_set_limit_mode:`

Set the reacting modes of the EL signal.

.

## \_8154\_db53\_M\_get\_io\_status:

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

Bit	Name	Description
0	RDY	RDY pin input
1	ALM	Alarm Signal
2	+EL	Positive Limit Switch
3	-EL	Negative Limit Switch
4	ORG	Origin Switch
5	DIR	DIR output
6	EMG	EMG status
7	Reserved	Reserved
8	ERC	ERC pin output
9	EZ	Index signal
10	CLR	Clear signal
11	Reserved	Reserved
12	SD	Slow Down signal input
13	INP	In-Position signal input
14	SVON	Servo-ON output status

## @ Syntax

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

```

I16 _8154_db53_M_set_servo(I16 Card_ID, I16
    AxisNo, I16 on_off);
I16 _8154_db53_M_set_clr_mode(I16 Card_ID, I16
    AxisNo, I16 clr_mode, I16
    targetCounterInBit);
I16 _8154_db53_M_set_inp(I16 Card_ID, I16 AxisNo,
    I16 inp_enable, I16 inp_logic);
I16 _8154_db53_M_set_alm(I16 Card_ID, I16 AxisNo,
    I16 alm_logic, I16 alm_mode);
I16 _8154_db53_M_set_erc(I16 Card_ID, I16 AxisNo,
    I16 erc_logic, I16 erc_pulse_width, I16
    erc_mode);
I16 _8154_db53_M_set_erc_out(I16 Card_ID, I16
    AxisNo);

```

```

I16 _8154_db53_M_clr_erc(I16 Card_ID, I16
    AxisNo);
I16 _8154_db53_M_set_sd(I16 Card_ID, I16 AxisNo,
    I16 sd_logic, I16 sd_latch, I16 sd_mode);
I16 _8154_db53_M_enable_sd(I16 Card_ID, I16
    AxisNo, I16 enable);
I16 _8154_db53_M_set_limit_mode(I16 Card_ID, I16
    AxisNo, I16 limit_mode);
I16 _8154_db53_M_get_io_status(I16 Card_ID, I16
    AxisNo, U16 *io_sts);

```

### Visual Basic (Windows 2000/XP)

```

B_8154_db53_M_set_servo(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal on_off As
    Integer) As Integer
B_8154_db53_M_set_clr_mode(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    clr_mode As Integer, ByBal
    targetCounterInBit as Integer) As Integer
B_8154_db53_M_set_inp(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal inp_enable As
    Integer, ByVal inp_logic As Integer) As
    Integer
B_8154_db53_M_set_alm(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal alm_logic As
    Integer, ByVal alm_mode As Integer) As
    Integer
B_8154_db53_M_set_erc(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal erc_logic As
    Integer, ByVal erc_pulse_width As Integer,
    ByVal erc_mode As Integer) As Integer
B_8154_db53_M_set_erc_out(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer) As Integer
B_8154_db53_M_clr_erc(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer) As Integer
B_8154_db53_M_set_sd(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal sd_logic As
    Integer, ByVal sd_latch As Integer, ByVal
    sd_mode As Integer) As Integer
B_8154_db53_M_enable_sd(ByVal Card_ID As Integer,
    ByVal AxisNo As Integer, ByVal Enable As
    Integer) As Integer

```

```

B_8154_db53_M_set_limit_mode(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    limit_mode As Integer) As Integer
B_8154_db53_M_get_io_status(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, io_sts As
    Integer) As Integer

```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**on\_off:** ON-OFF state of SVON signal

Value	Meaning
0	OFF
1	ON

**enable:** enable or disable

Value	Meaning
0	Disable
1	Enable

**clr\_mode:** Specify a CLR input clear mode

- ▶ clr\_mode = 0, Clear on the falling edge (default)
- ▶ clr\_mode = 1, Clear on the rising edge
- ▶ clr\_mode = 2, Clear on a LOW level
- ▶ clr\_mode = 3, Clear on a HIGH level

**targetCounterInBit:** Enable/Disable clear target counter in bit

Value	Meaning
Bit	Description
0	Reset command counter when CLR input turns ON
1	Reset position counter when CLR input turns ON
2	Reset error counter when CLR input turns ON

**inp\_enable:** INP function enabled/disabled

- ▶ inp\_enable = 0, Disabled (default)
- ▶ inp\_enable = 1, Enabled

**inp\_logic:** Set the active logic for the INP signal

Value	Meaning
0	Negative logic
1	Positive logic

**alm\_logic:** Setting of active logic for ALARM signals

Value	Meaning
0	Negative logic
1	Positive logic

**alm\_mode:** Reacting modes when receiving an ALARM signal.

Value	Meaning
0	Motor immediately stops (default)
1	Motor decelerates then stops

**erc\_logic:** Set the active logic for the ERC signal

Value	Meaning
0	Negative logic
1	Positive logic

**erc\_pulse\_width:** Set the pulse width of the ERC signal

Value	Meaning
0	12 us
1	102 us
2	409 us
3	1.6 ms
4	13 ms
5	52 ms
6	104 ms
7	Level output

**erc\_mode:**

Value	Meaning
0	Disable
1	Output ERC when stopped by EL, ALM, or EMG input
2	Output ERC when complete home return
3	Both 1 and 2

**sd\_logic:**

Value	Meaning
0	Negative logic
1	Positive logic

**sd\_latch:** Set the latch control for the SD signal

Value	Meaning
0	Do not latch
1	Latch

**sd\_mode:** Set the reacting mode of the SD signal

Value	Meaning
0	Slow down only
1	Slow down then stop

**enable:** Set the ramping-down point for high speed feed.

Value	Meaning
0	Automatic setting
1	Manual setting (default)

**limit\_mode:**

Value	Meaning
0	Stop immediately
1	Slow down then stop

**\*io\_sts:** I/O status. Please refer to 6.9 function description.



## 5.10 Position Control and Counters

### @ Name

\_8154\_db53\_M\_get\_position – Get the value of feedback position counter

\_8154\_db53\_M\_set\_position – Set the feedback position counter

\_8154\_db53\_M\_get\_command – Get the value of command position counter

\_8154\_db53\_M\_set\_command – Set the command position counter

\_8154\_db53\_M\_get\_error\_counter – Get the value of position error counter

\_8154\_db53\_M\_reset\_error\_counter – Reset the position error counter

\_8154\_db53\_M\_get\_target\_pos – Get the value of target position recorder

\_8154\_db53\_M\_reset\_target\_pos – Reset target position recorder

\_8154\_db53\_M\_get\_res\_distance – Get remaining pulses accumulated from motions

\_8154\_db53\_M\_set\_res\_distance – Set remaining pulses record

### @ Description

\_8154\_db53\_M\_get\_position:

This function is used to read the feedback position counter value. Note that this value has already been processed by the move ratio setting by \_8154\_db53\_M\_set\_move\_ratio(). If the move ratio is 0.5, then the value of position will be twice. The source of the feedback counter is selectable by the function \_8154\_db53\_M\_set\_feedback\_src() to be external EA/EB or internal pulse output of DB8153.

#### `_8154_db53_M_set_position:`

This function is used to change the feedback position counter to the specified value. Note that the value to be set will be processed by the move ratio. If move ratio is 0.5, then the set value will be twice as given value.

#### `_8154_db53_M_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 DB8153.

#### `_8154_db53_M_set_command:`

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

#### `_8154_db53_M_get_error_counter:`

This function is used to read the value of the position error counter.

#### `_8154_db53_M_reset_error_counter:`

This function is used to clear the position error counter.

#### `_8154_db53_M_get_target_pos:`

This function is used to read the value of the target position recorder. The target position recorder is maintained by the DB8153 software driver. It records the position to settle down for current running motion.

#### **\_8154\_db53\_M\_reset\_target\_pos:**

This function is used to set new value for the target position recorder. It is necessary to call this function when home return completes, or when a new feedback counter value is set by function **\_8154\_db53\_M\_set\_position()**.

#### **\_8154\_db53\_M\_get\_res\_distance:**

This function is used to read the value of the residue distance recorder. The target position recorder is maintained by the DB8153 software driver. It records the position to settle down for current running motion.

#### **\_8154\_db53\_M\_set\_res\_distance:**

This function is used to change the value of the residue distance counter

### **@ Syntax**

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

```
I16 _8154_db53_M_get_position(I16 Card_ID, I16
    AxisNo, F64 *Pos);
I16 _8154_db53_M_set_position(I16 Card_ID, I16
    AxisNo, F64 Pos);
I16 _8154_db53_M_get_command(I16 Card_ID, I16
    AxisNo, I32 *Command);
I16 _8154_db53_M_set_command(I16 Card_ID, I16
    AxisNo, I32 Command);
I16 _8154_db53_M_get_error_counter(I16 Card_ID,
    I16 AxisNo, I16 *error);
I16 _8154_db53_M_reset_error_counter(I16 Card_ID,
    I16 AxisNo);
I16 _8154_db53_M_get_target_pos(I16 Card_ID, I16
    AxisNo, F64 *T_pos);
I16 _8154_db53_M_reset_target_pos(I16 Card_ID,
    I16 AxisNo, F64 T_pos);
I16 _8154_db53_M_get_res_distance(I16 Card_ID,
    I16 AxisNo, F64 *Res_Distance);
```

```
I16 _8154_db53_M_set_res_distance(I16 Card_ID,  
I16 AxisNo, F64 Res_Distance);
```

### Visual Basic (Windows 2000/XP)

```
B_db53_8154_M_get_position(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, Pos As  
Double) As Integer  
B_db53_8154_M_set_position(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByVal Pos  
As Double) As Integer  
B_db53_8154_M_get_command(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, Cmd As  
Long) As Integer  
B_db53_8154_M_set_command(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByVal Cmd  
As Long) As Integer  
B_db53_8154_M_get_error_counter(ByVal Card_ID As  
Integer, ByVal AxisNo As  
Integer, ByRef error As Integer) As Integer  
B_db53_8154_M_reset_error_counter(ByVal Card_ID  
As Integer, ByVal AxisNo As Integer) As  
Integer  
B_db53_8154_M_reset_target_pos(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByVal Pos  
As Double) As Integer  
B_db53_8154_M_get_target_pos(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByRef Pos  
As Double) As Integer  
B_db53_8154_M_set_res_distance(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByVal  
Res_Distance As Double) As Integer  
B_db53_8154_M_get_res_distance(ByVal Card_ID As  
Integer, ByVal AxisNo As Integer, ByRef  
Res_Distance As Double) As Integer
```

### @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**Pos, \*Pos:** Feedback position counter value,  
 (\_8154\_db53\_M\_get/set\_position)

► range: -134217728 to 134217727

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

► range: -134217728 to 134217727

**\*error:** Position error counter value,

► range: -32768 to 32767

**TargetPos, \*TargetPos:** Target position recorder value,

► range: -134217728~134217727

**ResDistance, \* ResDistance:** residue distance

## 5.11 Position Compare and Latch

### @ Name

`_8154_db53_M_set_trigger_comparator` – Set the trigger comparator

`_8154_db53_M_set_error_comparator` – Set the error comparator

`_8154_db53_M_set_latch_source` – Set the latch timing for a counter

`_8154_db53_M_get_latch_data` – Get the latch data from counter

### @ Description

`_8154_db53_M_set_error_comparator`:

This function is used to set the comparing method and value for the error comparator. Don't use "`_8154_db53_M_set_error_comparator`" and "`_8154_db53_M_set_trigger_comparator`" at the same time, because they use comparator 3 together.

`_8154_db53_M_set_trigger_comparator`:

This function is used to set the comparing source counter, comparing method and value for the trigger comparator. When the comparison source counter's value reaches the comparing value, the db8153 will generate a pulse output via CMP. Don't use "`_8154_db53_M_set_error_comparator`" and "`_8154_db53_M_set_trigger_comparator`" at the same time, because they use comparator 3 together.

`_8154_db53_M_set_latch_source`:

There are 4 latch triggering source. By using this function, user can choose the event source to latch counters' data.

## \_8154\_db53\_M\_get\_latch\_data:

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

### @ Syntax

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

```
I16 _8154_db53_M_set_error_comparator(I16
    Card_ID, I16 AxisNo, I16 CmpMethod, I16
    CmpAction, I32 Data);
I16 _8154_db53_M_set_trigger_comparator(I16
    Card_ID, I16 AxisNo, I16 CmpSrc, I16
    CmpMethod, I32 Data);
I16 _8154_db53_M_set_latch_source(I16 Card_ID,
    I16 AxisNo, I16 LtcSrc);
I16 _8154_db53_M_get_latch_data(I16 Card_ID, I16
    AxisNo, I16 CounterNo, F64 *Pos);
```

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

```
B_8154_db53_M_set_error_comparator(ByVal Card_ID
    As Integer, ByVal AxisNo As Integer, ByVal
    CmpMethod As Integer, ByVal CmpAction As
    Integer, ByVal Data As Long) As Integer
B_8154_db53_M_set_trigger_comparator(ByVal
    Card_ID As Integer, ByVal AxisNo As Integer,
    ByVal CmpSrc As Integer, ByVal CmpMethod As
    Integer, ByVal Data As Long) As Integer
B_8154_db53_M_set_latch_source(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    LtcSrc As Integer) As Integer
B_8154_db53_M_get_latch_data(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    CounterNo As Integer, Pos As Double) As
    Integer
```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**CmpSrc:** The comparing source counters

Value	Meaning
0	Command counter
1	Feedback counter
2	Error counter

**CmpMethod:** The comparing methods

Value	Meaning
0	No Compare(Disable)
1	Data = Source counter (direction independent)
2	Data = Source counter (Count up only)
3	Data = Source counter (Count down only)
4	Data > Source counter
5	Data < Source counter

**Data:** Comparing value (Position)



**CmpAction:**

Value	Meaning
0	No action
1	Immediate Stop
2	Slow down then stop

**ltc\_src:**

Value	Meaning
1	ORG pin input
2	Software –EL conditions are met
3	Trigger or Error comparator conditions are met

**CounterNo:** Specified the counter to latch

Value	Meaning
0	Command counter
1	Feedback counter
2	Error counter

**\*Pos:** Latch data (Position)

## 5.12 Soft Limit

### @ Name

`_8154_db53_M_disable_soft_limit` – Disable soft limit function

`_8154_db53_M_enable_soft_limit` – Enable soft limit function

`_8154_db53_M_set_soft_limit` – Set soft limit

### @ Description

`_8154_db53_M_disable_soft_limit`:

This function is used to disable the soft limit function.

`_8154_db53_M_enable_soft_limit`:

This function is used to enable the soft limit function. Once enabled, the action of soft limit will be exactly the same as physical limit.

`_8154_db53_M_set_soft_limit`:

This function is used to set the soft limit value.

### @ Syntax

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

```
I16 _8154_db53_M_disable_soft_limit(I16 Card_ID,  
    I16 AxisNo);  
I16 _8154_db53_M_enable_soft_limit(I16 Card_ID,  
    I16 AxisNo, I16 Action);  
I16 _8154_db53_M_set_soft_limit(I16 Card_ID, I16  
    AxisNo, I32 PlusLimit, I32 MinusLimit);
```

## Visual Basic (Windows 2000/XP)

```

B_8154_db53_M_disable_soft_limit(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer) As Integer
B_8154_db53_M_enable_soft_limit(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    Action As Integer) As Integer
B_8154_db53_M_set_soft_limit(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    PlusLimit As Long, ByVal MinusLimit As Long)
    As Integer

```

### @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**Action:** The reacting method of soft limit

Value	Meaning
0	INT only
1	Immediately stop
2	slow down then stop

**PlusLimit:** Soft limit value, positive direction

**MinusLimit:** Soft limit value, negative direction

## 5.13 Backlash Compensation/Vibration Suppression

### @ Name

\_8154\_db53\_M\_backlash\_comp – Set backlash corrective pulse for compensation

\_8154\_db53\_M\_suppress\_vibration – Set vibration suppressing timing

\_8154\_db53\_M\_set\_fa\_speed – Set the FA speed

### @ Description

\_8154\_db53\_M\_backlash\_comp:

Whenever direction change occurs, the DB8153 outputs backlash corrective pulses before sending commands. This function is used to set the compensation pulse numbers.

\_8154\_db53\_M\_suppress\_vibration:

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

\_8154\_db53\_M\_set\_fa\_speed:

This function is used to specify the low speed for backlash correction or slip correction. It also used as a reverse low speed for home return operation.

## @ Syntax

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

```
I16 _8154_db53_M_backlash_comp(I16 Card_ID, I16
    AxisNo, I16 CompPulse, I16 Mode);
I16 _8154_db53_M_suppress_vibration(I16 Card_ID,
    I16 AxisNo, U16
ReverseTime, U16 ForwardTime);
I16 _8154_db53_M_set_fa_speed(I16 Card_ID, I16
    AxisNo, F64 FA_Speed);
```

### Visual Basic (Windows 2000/XP)

```
B_8154_db53_M_backlash_comps (ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    CompPulse As Integer, ByVal Mode As Integer)
    As Integer
B_8154_db53_M_suppress_vibration(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    ReverseTime As Integer, ByVal ForwardTime As
    Integer) As Integer
B_8154_db53_M_set_fa_speed(ByVal Card_ID As
    Integer, ByVal AxisNo As Integer, ByVal
    FA_Speed As Double) As Integer
```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

**CompPulse:** Specified number of corrective pulses, 12 bit

**Mode:**

Value	Meaning
0	Turns off
1	Enable backlash compensation
2	Slip correction

**ReverseTime:** Specified Reverse Time, 0 - 65535, unit 1.6 us

**ForwardTime:** Specified Forward Time, 0 - 65535, unit 1.6 us

**FA\_Speed:** fa speed (unit: pulse/sec)

## 5.14 Speed Profile Calculation

### @ Name

\_8154\_db53\_M\_get\_tr\_move\_profile – Get the relative trapezoidal speed profile

\_8154\_db53\_M\_get\_ta\_move\_profile – Get the absolute trapezoidal speed profile

\_8154\_db53\_M\_get\_sr\_move\_profile – Get the relative S-curve speed profile

\_8154\_db53\_M\_get\_sa\_move\_profile – Get the absolute S-curve speed profile

### @ Description

\_8154\_db53\_M\_get\_tr\_move\_profile:

This function is used to get the relative trapezoidal speed profiles. By this function, user can get the actual speed profile before running.

\_8154\_db53\_M\_get\_ta\_move\_profile:

This function is used to get the absolute trapezoidal speed profiles. By this function, user can get the actual speed profile before running.

\_8154\_db53\_M\_get\_sr\_move\_profile:

This function is used to get the relative S-curve speed profiles. By this function, user can get the actual speed profile before running.

\_8154\_db53\_M\_get\_sa\_move\_profile:

This function is used to get the absolute S-curve speed profiles. By this function user can get the actual speed profile before running.

## @ Syntax

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

```
I16 _8154_db53_M_get_tr_move_profile(I16 Card_ID,  
    I16 AxisNo, F64 Dist, F64 StrVel, F64  
    MaxVel, F64 Tacc, F64 Tdec, F64 *pStrVel,  
    F64 *pMaxVel, F64 *pTacc, F64 *pTdec, F64  
    *pTconst );  
  
I16 _8154_db53_M_get_ta_move_profile(I16 Card_ID,  
    I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,  
    F64 Tacc, F64 Tdec, F64 *pStrVel, F64  
    *pMaxVel, F64 *pTacc, F64 *pTdec, F64  
    *pTconst );  
  
I16 _8154_db53_M_get_sr_move_profile(I16 Card_ID,  
    I16 AxisNo, F64 Dist, F64 StrVel, F64  
    MaxVel, F64 Tacc, F64 Tdec, F64 SVacc, F64  
    SVdec, F64 *pStrVel, F64 *pMaxVel, F64  
    *pTacc, F64 *pTdec, F64 *pSVacc, F64  
    *pSVdec, F64 *pTconst);  
  
I16 _8154_db53_M_get_sa_move_profile(I16 Card_ID,  
    I16 AxisNo, F64 Pos, F64 StrVel, F64 MaxVel,  
    F64 Tacc, F64 Tdec, F64 SVacc, F64 SVdec, F64  
    *pStrVel, F64 *pMaxVel, F64 *pTacc, F64  
    *pTdec, F64 *pSVacc, F64 *pSVdec, F64  
    *pTconst);
```



## Visual Basic (Windows 2000/XP)

```
B_8154_db53_M_get_tr_move_profile(ByVal Card_ID
    As Integer, ByVal AxisNo As Integer, ByVal
    Dist As Double, ByVal StrVel As Double,
    ByVal MaxVel As Double, ByVal Tacc As
    Double, ByVal Tdec As Double, ByRef pStrVel
    As Double, ByRef pMaxVel As Double, ByRef
    pTacc As Double, ByRef pTdec As Double,
    ByRef pTconst As Double) As Integer

B_8154_db53_M_get_ta_move_profile(ByVal Card_ID
    As Integer, ByVal AxisNo As Integer, ByVal
    Pos As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
    ByVal Tdec As Double, ByRef pStrVel As
    Double, ByRef pMaxVel As Double, ByRef pTacc
    As Double, ByRef pTdec As Double, ByRef
    pTconst As Double) As Integer

B_8154_db53_M_get_sr_move_profile(ByVal Card_ID
    As Integer, ByVal AxisNo As Integer, ByVal
    Dist As Double, ByVal StrVel As Double,
    ByVal MaxVel As Double, ByVal Tacc As
    Double, ByVal Tdec As Double, ByVal SVacc As
    Double, ByVal SVdec As Double, ByRef pStrVel
    As Double, ByRef pMaxVel As Double, ByRef
    pTacc As Double, ByRef pTdec As Double,
    ByRef pSVacc As Double, ByRef pSVdec As
    Double, ByRef pTconst As Double) As Integer

B_8154_db53_M_get_sa_move_profile(ByVal Card_ID
    As Integer, ByVal AxisNo As Integer, ByVal
    Pos As Double, ByVal StrVel As Double, ByVal
    MaxVel As Double, ByVal Tacc As Double,
    ByVal Tdec As Double, ByVal SVacc As Double,
    ByVal SVdec As Double, ByRef pStrVel As
    Double, ByRef pMaxVel As Double, ByRef pTacc
    As Double, ByRef pTdec As Double, ByRef
    pSVacc As Double, ByRef pSVdec As Double,
    ByRef pTconst As Double) As Integer
```

## @ Argument

**Card\_ID:** Specify the MNET master card index. Normally, the board index would be decided by the sequence of PCI slot.

**AxisNo:** Axis number designated to move or stop.

card_id	Slave_No	Slave_type	AxisNo
0	0	Motion	0
	1	Other type	No
	2	Motion	1
	3	Other type	No
	4	Motion	2
	....	....	....
	63	....	....

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

**Pos:** Specified absolute position (unit: pulse)

**StrVel:** Starting velocity (unit: pulse/sec)

**MaxVel:** Maximum velocity (unit: pulse/sec)

**Tacc:** time for acceleration (unit: sec)

**Tdec:** time for deceleration (unit: sec)

**SVacc:** S-curve region during acceleration (unit: pulse/sec)

- Note: SVacc = 0, for pure S-Curve. For more details, see section 3.3.3.

**SVdec:** S-curve region during deceleration (unit: pulse/sec)

- Note: SVdec = 0, for pure S-Curve. For more details, see section 3.3.3.

- \***pStrVel**: Starting velocity by calculation
- \***pMaxVel**: Maximum velocity by calculation
- \***pTacc**: Acceleration time by calculation
- \***pTdec**: Deceleration time by calculation
- \***pSVacc**: S-curve region during acceleration by calculation
- \***pSVdec**: S-curve region during deceleration by calculation
- \***pTconst**: constant speed time(maximum speed)

## 5.15 Return Code

The return error code is defined in "MNET\_err.h". The meaning is described in following table.

Code	Meaning
0	No error
-10500	Error Card number
-10501	Error card's ID conflict
-10502	Error Invalid SlaveNo
-10503	Error transfer rate
-10504	Error clock rate
-10505	Error DB8153 not Initialize yet
-10506	Error Invalid AxisNo
-10506	Error speed error
-10507	Error event already enabled
-10508	Error event not enable yet
-10509	Error position out of range
-10510	Error motion busy
-10511	Error speed error
-10512	Error axis range error
-10513	Error compare parameter error
-10514	Error compare method
-10515	Error Invalid CmpSrc
-10516	Error Invalid CmpAction
-10517	Error axis already stop
-10518	Error enable range
-10519	Error auto accelerate time
-10520	Error dwell time
-10521	Error dwell distance
-10522	Error counter number
-10523	Error port type
-10524	Error time out
-10525	Error read mismatch command
-10526	Error Int setting

Code	Meaning
-10527	Error axis INT wait failed
-10528	Error Invalid RW Method
-10529	Error RWCOMMAND Other MaxTry
-10530	Error RWCOMMAND Other Enter
-10531	Error RWCOMMAND MaxTry
-10532	Error RWCOMMAND NOT FINISH
-10533	Error FNPARAMETER Other MaxTry
-10534	Error FNPARAMETER Other Enter
-10535	Error RW Method
-10536	Error Unknown RWType
-10537	Error Invalid SlaveNo

