PCIe-7856/7853

Master-Slave Distributed Motion
and I/O Master Controller

User's Manual

Manual Rev.:	1.1
Revision Date:	Sept. 17, 2020
Part No:	50-15113-1010

Leading **EDGE COMPUTING**

Revision History

Revision	Release Date	Description of Change(s)
1.0	2020-02-27	Initial release
1.1	2020-09-17	Add PCIe-7853; integrate HSL System documentation

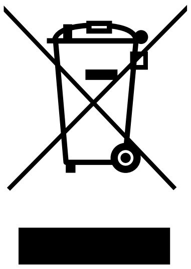
Preface

Disclaimer

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.

Environmental



ADLINK is committed to fulfill its social responsibility to global environmental preservation through compliance with the European Union's Restriction of Hazardous Substances (RoHS) directive and Waste Electrical and Electronic Equipment (WEEE) directive. Environmental protection is a top priority for ADLINK. We have enforced measures to ensure that our products, manufacturing processes, components, and raw materials have as little impact on the environment as possible. When products are at their end of life, our customers are encouraged to dispose of them in accordance with the product disposal and/or recovery programs prescribed by their nation or company.

Battery Labels (for products with battery)

Li-ion
RECYCLE
RBRC
Li·ion
1.800.822.8837
廢電池請回收

California Proposition 65 Warning

WARNING: This product can expose you to chemicals including acrylamide, arsenic, benzene, cadmium, Tris(1,3-dichloro-2-propyl)phosphate (TDCPP), 1,4-Dioxane, formaldehyde, lead, DEHP, styrene, DINP, BBP, PVC, and vinyl materials, which are known to the State of California to cause cancer, and acrylamide, benzene, cadmium, lead, mercury, phthalates, toluene, DEHP, DIDP, DnHP, DBP, BBP, PVC, and vinyl materials, which are known to the State of California to cause birth defects or other reproductive harm. For more information go to www.P65Warnings.ca.gov**.**

Trademarks

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

Conventions

Take note of the following conventions used throughout this manual to make sure that users perform certain tasks and instructions properly.

NOTE: Additional information, aids, and tips that help users perform tasks.

CAUTION: Information to prevent ***minor*** physical injury, component damage, data loss, and/or program corruption when trying to complete a task.

WARNING: Information to prevent ***serious*** physical injury, component damage, data loss, and/or program corruption when trying to complete a specific task.

Table of Contents

This page intentionally left blank.

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

1 Introduction

As industrial machine automation continues to advance, designers require not only centralized control systems but also distributed solutions to develop more complex machine applications. Distributed solutions provide many benefits such as lower maintenance, reduced wiring, and easy integration of vast numbers of modules. Motionnet (MNET) and High Speed Link (HSL) are innovative distributed motion and I/O technologies that enable time-deterministic scanning of thousands of I/O points within milliseconds using the master-slave architecture. The MNET bus further improves distributed motion control capability by providing control of up to 256 axes plus minimal command execution time for single-axis control.

The PCIe-7856 is a PCI Express interface card with two ports for MNET and HSL systems for distributed motion and I/O modules for a wide variety of machine automation applications.

HSL technology allows thousands of I/O points to be scanned at the millisecond level in real time by means of the master-slave architecture. Commercial Ethernet cables with RJ45 connectors are used for simplified setup of HSL slave modules as close as possible to sensor devices, resulting in dramatic wiring reduction. System integrators can greatly benefit from an HSL network because it integrates discrete I/O and analog I/O modules. This local network features rapid-response, real-time scanning.

An MNET system is a distributed motion solution for machine systems. MNET is an innovative distributed motion technology which provides distributed motion axis control of up to 256 axes for any servo/stepper motor controlled using mater-slave architecture. This not only facilitates general purpose 4-axis motion control, but also allows up to 64 specific single-axis motion control modules to be scanned at the millisecond level in real time.

MNET and HSL features:

- ▶ Flexible, comprehensive, extendable distributed motion and I/O solution based on PC architecture or embedded platform.
- ▶ Convenient wiring for remote distributed motion and I/O modules, inc. multiple-axis motion control modules, single-axis motion control modules, discrete I/Os, and analog I/Os.
- ▶ Saves space, reduces wiring, and lowers costs due to ease of maintenance.
- ▶ Fast, time-deterministic scanning with hundreds of discrete I/O points (up to 2,016 points).
- ▶ Rapid, real-time scanning to support high-speed and high-response motion control of up to 256 axes.

The PCIe-7856 block diagram is as follows.

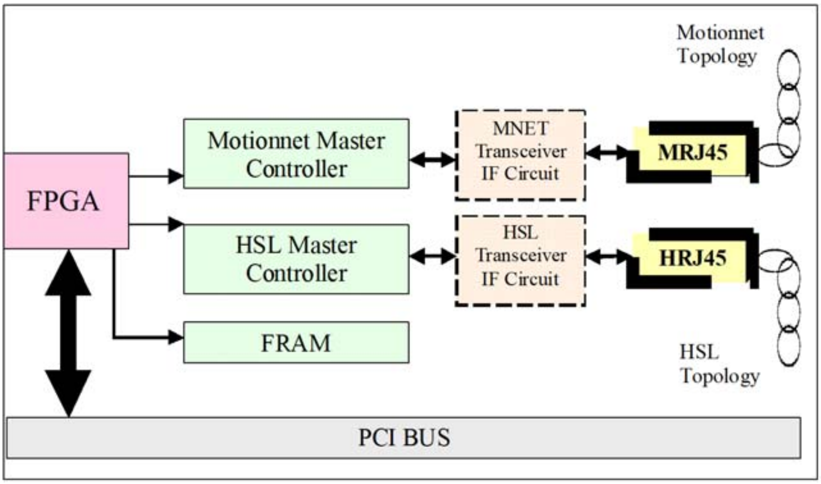

Motionnet Topology
FPGA
Motionnet Master Controller
MNET Transceiver IF Circuit
MRJ45
HSL Master Controller
HSL Transceiver IF Circuit
HRJ45
FRAM
HSL Topology
PCI BUS

Figure 1-1: PCIe-7856 Block Diagram

1.1 Specifications

		PCIe-7856	PCIe-7853
Bus		PCI Express x1, Plug and Play	
Master Controller		Dedicated Motion Controller	N/A
		Motionnet ASIC master control (80 MHz external clock)	N/A
		Dedicated I/O Controller: HSL ASIC master control (48 MHz external Clock)	
Interface	Motion	RS-485 with transformer isolation	N/A
		Half duplex communication	N/A
		2.5/5/10/20 Mbps transmission rate can be set by software (20 Mbps default)	N/A
	HSL	RS-422 with transformer isolation	
		Full duplex communication	
		3/6/12 Mbps transmission rate can be set by software (6 Mbps default)	
Connectors		RJ45 connector x4 (MRJ45 connector for Motionnet; HRJ45 connector for HSL)	RJ-45 connector x2 (HRJ45 connector for HSL)
Interrupt		Status read back	
Storage Temperature		-20°C to +80°C (-4°F to 176°F)	
Power Consumption		+3.3 V @ 1.2 A (typical)	
		+5 V @ 1.5 A (typical)	
Dimensions		119.5 mm x 100.2 mm (L x W) (4.66” x 3.9”)	
Operating Systems		Windows 10/8/7 (32/64 bit)	
Software Compatibility		VB, VC++, BCB, Delphi, VB.net, C# compatible	VB6, VC++, C# compatible
		Various sample programs with source codes	
Software Recommendations		APS SDK	HSL LinkMaster Utility
Certifications		FCC Part 15 B / EN 55032&55035	

1.2 Supported Software

Program Library

ADLINK provides Windows WDM drivers and DLL function libraries for the PCIe-7856. These function libraries are shipped with the board and they support Windows 7/8/10 (32/64-bit).

2 Installation

This chapter describes how to install and set up the PCIe-7856. Please follow these steps:

- Check the product for any sign of defect or damage (use Figure 2-1 on page 6 as a reference).
- Install software drivers (Section 2.3, page 7).
- Understand the I/O signal connections and how to use them (Section 2.5 on page 7).

2.1 Package Contents

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

- PCIe-7856: Distributed Motion and I/O Master Controller x1

or

- PCIe-7853: High Speed Link Master Controller Controller x1

If any part of the item is 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.

Signal connections of all I/O's are described later in this chapter. Refer to the contents of this chapter before wiring any cables between the PCIe-7856 and any slave module.

2.2 PCIe-7856 Mechanical Drawing

Dimensions: mm

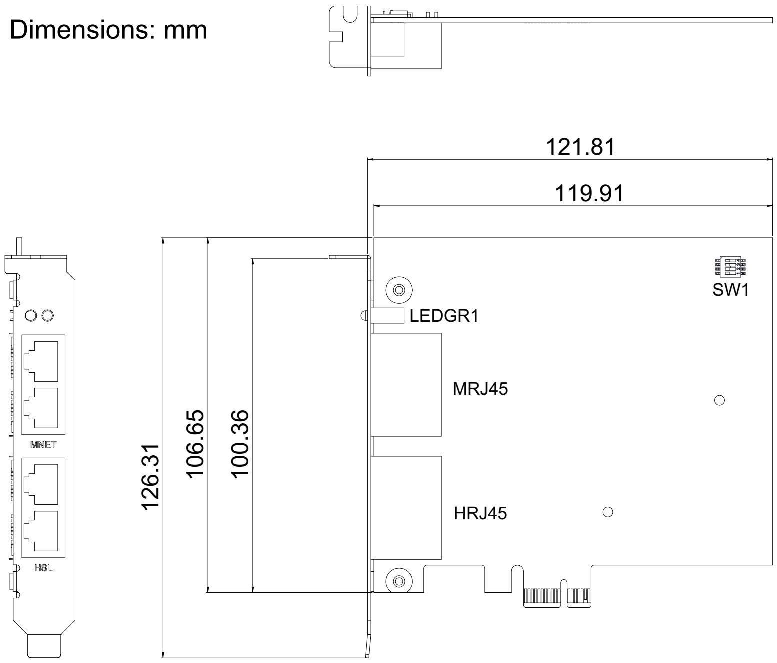


Figure 2-1: PCIe-7856 Mechanical Drawing

- **MRJ45:** MNET connection port
- **HRJ45:** HSL connection port
- **SW1:** Card identification switch
- **LEDGR1:** HSL/Motionnet Scan LEDs

2.3 Driver Installation

2.3.1 PCIe-7856

Open a web browser and navigate to the PCIe-7856/7853 product web page (https://www.adlinktech.com/Products/Motion_Control/DistributedMotionControl/PCIe-7856_7853). Under **Technical Resources**, select **Driver** to see the list of available drivers. After downloading the required ZIP file, extract and double-click the enclosed executable file to run the installer. Follow the installation steps and, after installation is complete, restart the PC.

2.3.2 PCIe-7853

Open a web browser and navigate to the PCIe-7856/7853 product web page (https://www.adlinktech.com/Products/Motion_Control/DistributedMotionControl/PCIe-7856_7853). Under **Technical Resources**, select **Driver** to see the list of available drivers. Download the HSL LinkMaster Utility for the PCIe-7853 and install it, following the onscreen instructions.

2.4 Troubleshooting

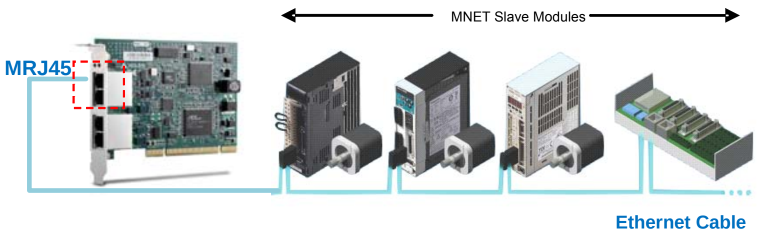
If the system doesn't boot or if the PCIe board exhibits any erratic behavior, it is most likely caused by an interrupt conflict. After confirming the issue wasn't caused by a simple oversight, the solution can be found by consulting the BIOS documentation that comes with your system. Check the Windows control panel on the connected PC to see if the card is listed by the system. If not, check the PCIe settings in the BIOS or use another PCIe slot.

2.5 Signal Connections

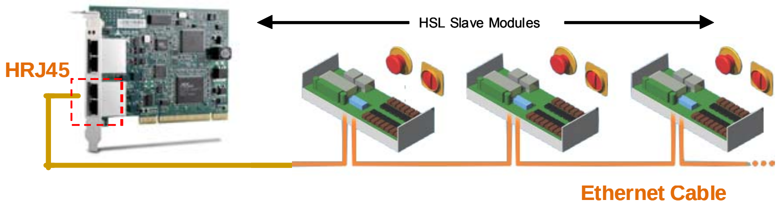
Signal connections of all I/O's are described in the following two sub-sections. Please review this information before wiring any cables between the PCIe-7856 and slave modules.

2.5.1 Connecting HSL/MNET Slave Modules

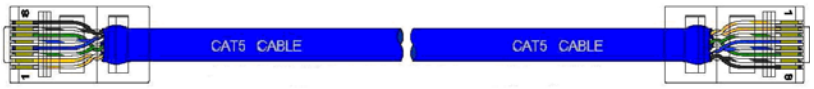
Wiring for MNET Motion Slave Modules



Wiring for HSL I/O Slave Modules

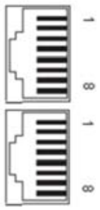


Ethernet Cable (CAT5e Recommended)



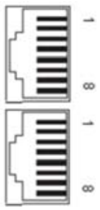
2.5.2 RJ45 Pin Assignments

The Motionnet (MNET) master is the key component in charge of communicating with slave motion modules. The master sends commands to slave motion controllers and obtains motion status from them. The PCIe-7856 provides two MNET master connection ports for greater wiring flexibility. The pin assignments of the MRJ45 connector on the PCIe-7856 are as listed below:



Pin No.	Pinout
1	NC
2	NC
3	NC
4	Data-
5	Data+
6	NC
7	NC
8	NC

The HSL master is the key component in charge of communicating with slave I/O modules. The master sends output values to, and gathers input information from, the slaves. PCIe-7856 provides two ports for HSL master connections for greater wiring flexibility. The pin assignments of the HRJ45 connector on the PCIe-7856 are as follows:



Pin No.	Pinout
1	NC
2	NC
3	RX+
4	TX-
5	TX+
6	RX-
7	NC
8	NC

2.5.3 HSL and Motionnet LED Indicators

The two LEDs on the PCIe-7856 provide communication status information. The red LED indicates MNET status and the green LED indicates HSL status. Before initialization of the PCIe-7856, both LEDs will be off. After initialization, the LEDs will begin blinking at a 1 Hz frequency.

When the PCIe-7856 connects to an HSL slave module, the green LED will turn on and remain constantly on during the scanning process, after which the green LED will continue blinking at 1 Hz.

When the PCIe-7856 connects to an MNET slave module, the red LED will turn on and remain constantly on until the scanning process stops or a communication error occurs, after which the red LED will continue blinking at 1 Hz.

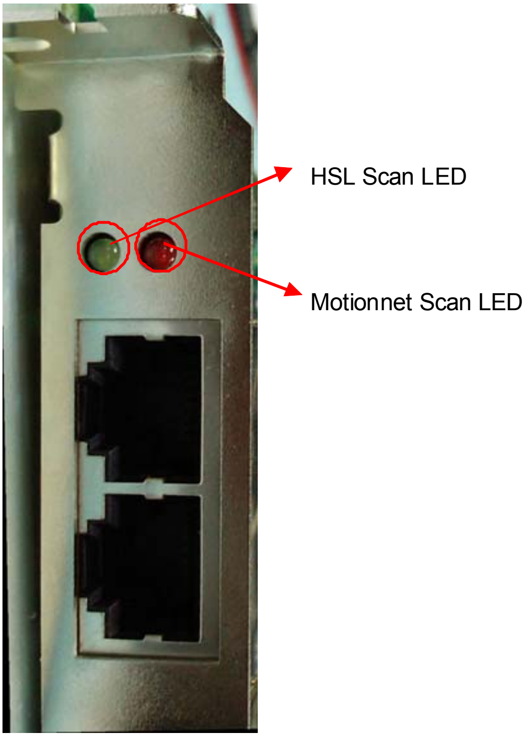
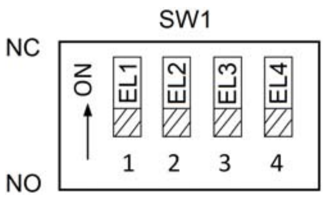


Figure 2-2: LED Indicators on the PCIe-7856

2.6 SW1 Card ID Switch Settings

The card ID can be set via the SW1 DIP switch as follows.



ON = 1
0000 Card ID 0
0001 Card ID 1
0010 Card ID 2
…   …
1110 Card ID 14
1111 Card ID 15
OFF = 0

This page intentionally left blank.

3 MNET Master-Slave Motion System

Motionnet (MNET) is an ultra-high-speed serial communication system proposed by Nippon Pulse Motor (NPM). It features strong performance with a maximum transfer speed of up to 20Mbps. The PCIe-7856 is equipped with one MNET port offering control of up to 256 axes via serial connections. ADLINK MNET solutions include not only single-axis controllers suitable for multiple PTP (point-to-point) movement applications, but also 4-axis motion controllers with support for linear and circular interpolation functions. Individual devices can control Panasonic A4 servo drivers. The controller can be used for executing continuous operations at constant speeds, performing linear as well as S-curve acceleration and deceleration, carrying out preset positioning operations, executing zero return operations, and so forth. As for connection distance, the cable length can be extended by up to 100 meters using an ordinary CAT5e LAN cable while connecting 64 axes at 20Mbps. All function library designs are compatible with ADLINK's PCIe motion controllers and MNET bus motion controllers.

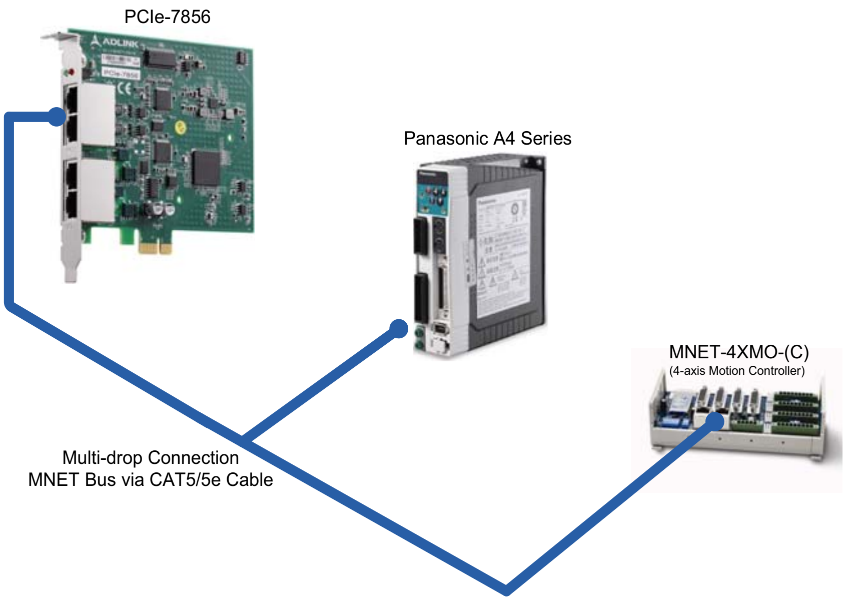


Figure 3-1: MNET Distributed Motion Control System

3.1 MNET System Specifications

The two major functions of a Motionnet (MNET) system are serial communication and motion control.

Item	Specifications
Total serial communication line length (using recommended cables)	▶ Maximum of 100m (at a data transfer speed of 20 Mbps with 32 devices connected) ▶ Maximum of 50m (at a data transfer speed of 20 Mbps with 64 devices connected) ▶ Maximum of 100m (at a data transfer speed of 10 Mbps with 64 devices connected)
Serial communication interface	Pulse transformer and RS-485 specification line transceiver
Serial communication protocol	ADLINK 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
Maximum number of MNET modules	64 (the total number of axes will be 64 if all single-axis modules are connected or 256 if all modules belong to MNET-4XMO)

Table 3-1: MNET System Specifications

3.1.1 Wiring Cables

The PCIe-7856 system guarantees enhanced quality for high-speed communication and is designed to be connected with user-provided LAN cables suitable for 100BASE-T and 1000BASE-T. Because these cables have well-known specifications and are cheap and easy to obtain, we do not provide them and do not include them in our product lines. When selecting cables, make sure they meet one of the following standards.

Wiring Standards

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

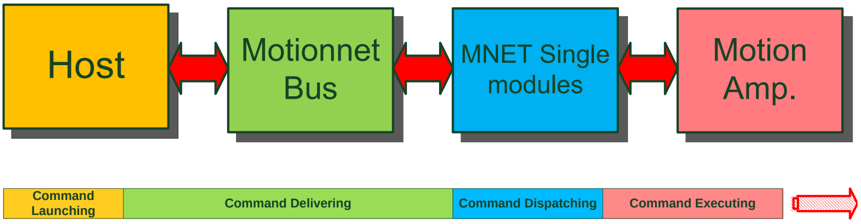
Choose UTP (Unshielded Twisted Pair) or STP (Shielded Twisted Pair) cables that meet one of the standards above. For an environment with excessive electromagnetic noise, use a shielded cable (STP).

Observe the following when connecting your system.

1. Keep the total serial line length as short as possible.
2. Maximum total serial line length will vary based on the data transfer speed and the number of local boards that are connected (this system employs a multi-drop connection method).
   - ▷ 20 Mbps with 32 modules connected: Max. 100 m
   - ▷ 20 Mbps with 64 modules connected: Max. 50 m
   - ▷ 10 Mbps with 64 modules connected: Max. 100 m
3. The shortest cable must be at least 60 cm long.
4. Do not mix cables of different types or models in the same serial line.
5. If 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.

3.1.2 MNET System Communication

The following is a communication block diagram for a Missionnet (MNET) system.



Command Launching

Within the MNET system, remote modules communicate with each other using MNET network packets, but users do not need to understand the contents of these packets. Several API functions are provided for controlling modules and these functions are easy to understand and use.

API functions can analyze parameters from user commands and pack them as MNET network packets. The packets are then passed to remote modules. The remote modules will interpret the packets and execute the commands. Before launching a packet, all commands issued by the user are written into RAM and transferred on the MNET network.

RAM, therefore, is a bridge between the MNET master controller and the host PC. The RAM access time for one packet is about 600 ns and should be quite fast on the host PC. The delivery time for one command on the network will depend on the number of modules and the operating clock rate. In addition to using RAM, users can also write data into a FIFO queue in the central device and then issue a “send” command. This communication will be sent and received automatically by interrupting the cyclic communication. Complete command delivery time will depend on the number of MNET packets. One packet command can be delivered in one MNET scan (cycle) time.

Command Delivery

For commands delivered as part of the *cyclic communication* process, the time allowed for communication by a single module is fixed. However, in a direct *data communication* the communication time will vary based on how the communication is controlled by the user's program and the time needed to access the PCIe-7856.

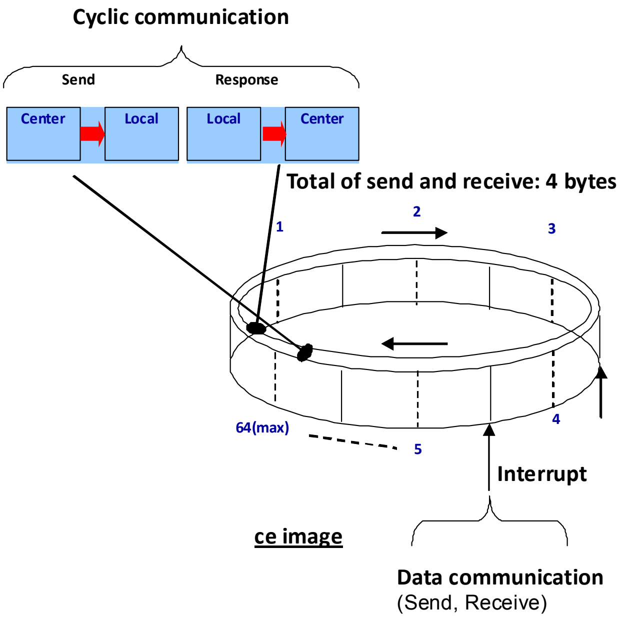


Figure 3-2: MNET System Communication Sequence

3.2 MNET Motion Modules

Motionnet (MNET) motion slave modules are wire-saving solutions. ADLINK provides two types of general purpose 4-axis modules: MNET-4XMO and MNET-4XMO-C. Both offer crucial motion functions such as point-to-point, zero-position searching, programmable acceleration/deceleration, T/S curve speed profile, etc. In addition, the MNET-4XMO-C also supports high-speed position comparison, a trigger output function, and a point table for continuous contouring applications. For additional details, please refer to the MNET-4XMO Series User’s Manual available for download at: https://www.adlinktech.com/Products/Industrial_Fieldbus/Motionnet/MNET-4XMO-(C).

ADLINK has also provided single-axis motion modules with specific drivers for connecting to Panasonic A4 servos. These single-axis modules have reached “end of life,” however, and the 4-axis modules are recommended as they can be conveniently plugged into any of those servos.

Regardless of the module and servo types involved, the servos themselves can be connected serially by the recommended cable type, greatly reducing wiring requirements.

Series	Model	Servo Driver	Axes	Mechanical I/O
MNET Single-axis Motion Modules	MNET-MIA	Panasonic A4	1	PEL, MEL, ORG, SD, EMG
MNET 4-axis Motion Modules	MNET-4XMO	General Purpose	4	PEL, MEL, ORG, SD, EMG
	MNET-4XMO-C	General Purpose	4	PEL, MEL, ORG, SD, EMG, TRG

Table 3-2: MNET Motion Module Series

The MNET-MIA can control a servomotor when I/O signals from a Panasonic (Matsushita) servo amplifier MINAS A4 series (pulse command supporting type) servo amplifier CNI/F or CNX5 are routed directly to this connector: CN4.

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

Since these modules can connect directly to the mechanical I/O signals of a servo driver, they do not need the special servo driver cable required of conventional motion control modules, thus saving time and providing the following advantages: simplified wiring, shortened wiring runs, and reduction of problems caused by faulty wiring. They also offer high noise immunity, take full advantage of high-speed signal lines to handle command pulses, and are highly compact, especially since they conserve wiring space.

Again, ADLINK also offers general purpose 4-axis motion control modules which are highly recommended, especially if using servos or stepper motors that are not described above or if more advanced motion functionality is required (such as linear/circular interpolation). By using specific or general purpose D-Sub cables, these 4-axis modules can directly connect to servo drivers, including the Panasonic MINAS A4.

3.2.1 Motion Module Mechanical Drawings

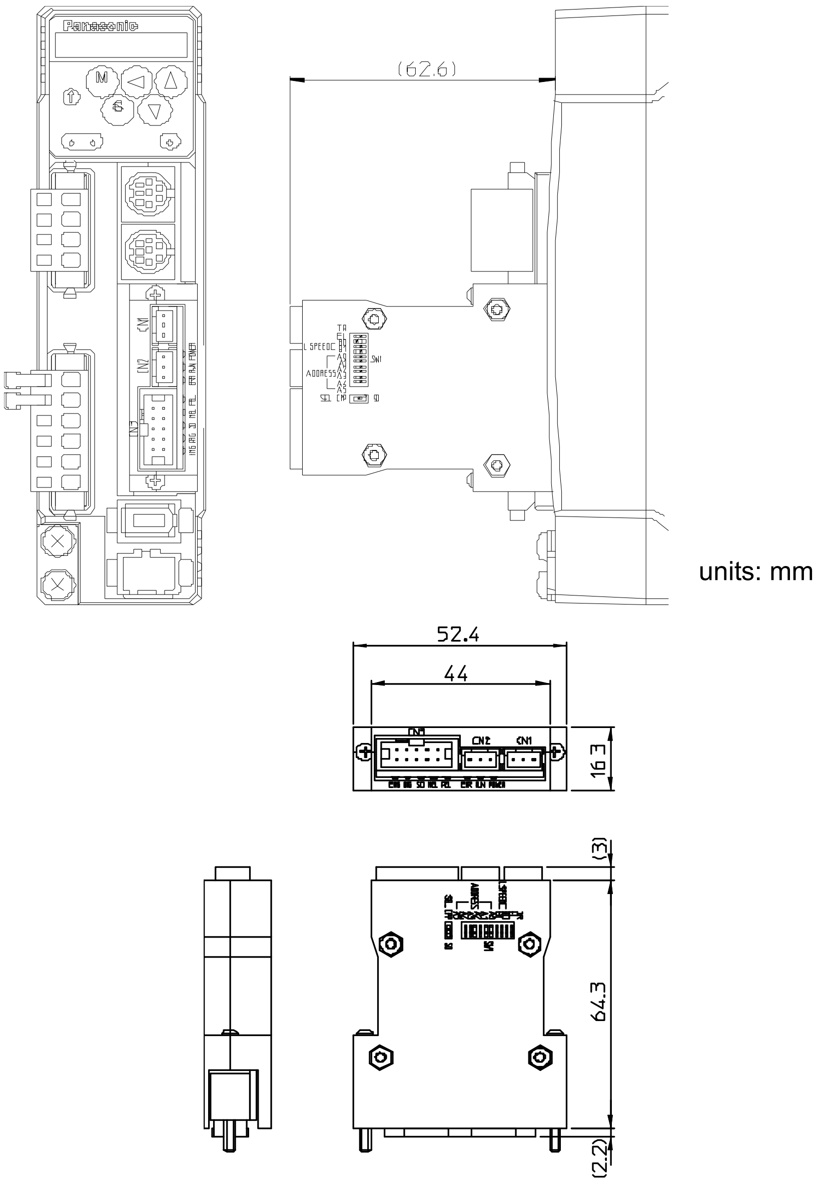
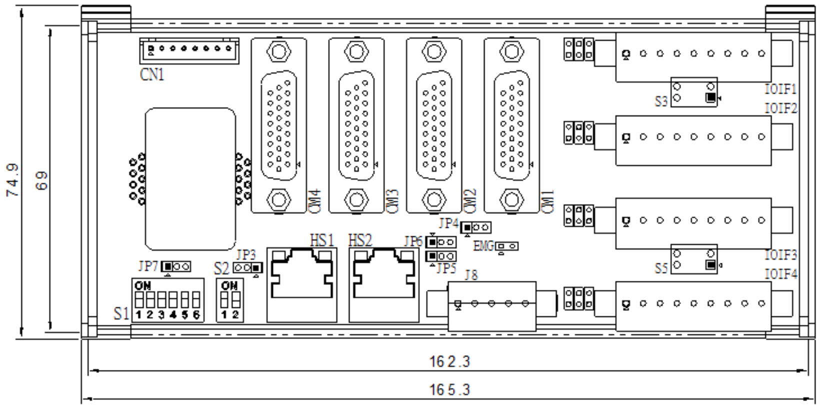


Figure 3-3: MNET-MIA with MINAS A4 Servo Driver



units: mm

Figure 3-4: MNET-4XMO-(C) Mechanical Diagram

This page intentionally left blank.

4 HSL Slave Modules

High Speed Link (HSL) is a master-slave network system featuring an innovative distributed architecture that modularizes communication, I/O functionality, and signal termination. ADLINK provides slave I/O modules and terminal bases to meet your particular application requirements, including discrete I/O, analog I/O, and motion control. For complete details about the modules, please refer to the HSL-4XMO User’s Manual, available here: https://www.adlinktech.com/Products/Industrial_Fieldbus/HighSpeedLink(HSL)/HSL-4XMO

Slave I/O Modules: There are three groups of slave I/O modules with different dimensions. Slave I/O modules give terminal bases additional I/O capability. To identify each slave I/O module in a HSL network, a module type electronic data sheet is stored in the module itself. Slave I/O modules can also be located by address ID, set by a 6-bit DIP-switch. Depending on the I/O type, each slave I/O module may consume 1 or 2 address IDs. Since the greatest ID number in a HSL master is 63 (the highest value for a 6-bit unsigned integer), and since ‘0’ is reserved for the master, there are at most 63 slave I/O modules for each HSL master.

Terminal Bases: The function of a terminal base (TB) slave module is to facilitate convenient wiring. Both power and signal wiring go from the TB into the slave I/O modules. TBs can also facilitate RJ-45 connections between masters and slave I/O modules. With the help of a TB, slave I/O modules can be hot-swapped without interfering with other modules on the same HSL network.

U-series Modules: U-series slave modules provide direct I/O signal wiring on top of the device. They are more compact than TBs and offer several I/O interface types to facilitate signal linking.

HUB/Repeaters: HSL-HUB/Repeaters provide more sub-system flexibility and thus enable a greater variety of topologies.

Wiring Cables: The communication wiring cables between an HSL master and its I/O modules are standard 100 Base/TX with RJ-45 connectors, the same as commercial Ethernet cables.

4.1 HSL Slave I/O Modules

4.1.1 Discrete I/O Module

ADLINK provides a discrete M series daughter board form factor I/O module with aluminum cover.

Series	Model	Discrete Inputs	Discrete Outputs	Relay Outputs	Slave Index Occupation
M	HSL-DI32-M-N	32			2 (consecutive from odd number)
	HSL-DO32-M-N		32		2 (consecutive from odd number)
	HSL-DI16DO16-M-NN/PN	16	16		1

Table 4-1: HSL Discrete I/O Module Series

The selection guide is as follows:

HSL	-	Discrete I/O Type	-	Series	-	Signal Type
		▶ **DI16DO16:** 16 discrete inputs and 16 discrete outputs ▶ **DI32:** 32 discrete inputs ▶ **DO32:** 32 discrete outputs		▶ **M:** Daughter board with aluminum cover		▶ **X:** Input Signal Type: NPN sinking or PNP sourcing support ▶ **Y:** Output Signal Type: NPN sinking or PNP sourcing support

Table 4-2: HSL Discrete I/O Module Selection Guide

4.1.2 Analog I/O Modules

ADLINK provides M and U analog I/O modules, as shown below.

Series	Model	Analog Input	Analog Output	Slave Index Occupation
M	HSL-AI16AO2-M-VV	16	2	2 (leap number)
U	HSL-AO4		4	2

Table 4-3: HSL Analog I/O Module Series

The selection guide is as follows.

HSL	-	Discrete I/O Type	-	Series	-	Signal Type
		▶ **AI16AO2:** 16 analog inputs and 2 analog outputs		▶ **M:** Daughter board with aluminum cover		▶ **X:** Input signal type, V for voltage and A for current ▶ **Y:** Output signal type, V means voltage

Table 4-4: HSL Analog I/O Module Selection Guide

4.2 General Specifications

4.2.1 Digital I/O Modules

Discrete Input	Photo Couple Isolation	2500 VRMS	
	Input Impedance	4.7 kΩ	
	Input Voltage	+24 V	
	Input Current	For NPN(1)	-10 mA
		For PNP(2)	+10 mA
	Operation Voltage (@ 24 V_{DC} Power Supply)	For NPN(1)	ON: 11.4 V_{DC} (max.) OFF: 14.3 V_{DC} (min.)
		For PNP(2)	ON: 12.6 V_{DC} (min.) OFF: 9.8 V_{DC} (max.)
	Response Time	ON: 8.8 µs (typical) OFF: 42 µs (typical)	
Discrete Output	Switch Capacity	For NPN(3)	All channels: -50 mA/ ch at 24 V_{DC}
		For PNP(4)	All channels: +50 mA/ ch at 24 V_{DC}
	Response Time	ON to OFF: 68 µs	
		OFF to ON: 1.1 µs	
Relay	Relay Type	SPST, normally open, non-latching	
	Rating	30 V_{DC}/2 A, 250 V_{AC}/2 A	
	Switching Frequency	20 times/minute at rating load	
	Response Time	ON to OFF: 3 µs (max.)	
		OFF to ON: 6 µs (max.)	

Table 4-5: Digital I/O Module

NOTE:

(1) NPN sinking type sensor input modules.
(2) PNP sourcing type sensor input modules.
(3) NPN sinking type sensor output modules.
(4) PNP sourcing type sensor output modules.

4.2.2 Analog I/O Modules

Analog Input	A/D Resolution	16-bit (14-bit guaranteed)
	Input Range	For VV type: ±10 V, ±5, ±2.5, ±1.25 V
		For AV type: 20 mA, 10 mA, 5 mA
	A/D Conversion	10 µs
	Signal Type	16-ch Single Ended; 8-ch Differential
Analog Output	D/A Resolution	16-bit
	D/A Settling Time	10 µs

Table 4-6: Analog I/O Modules

4.2.3 HSL Module DIP Switch

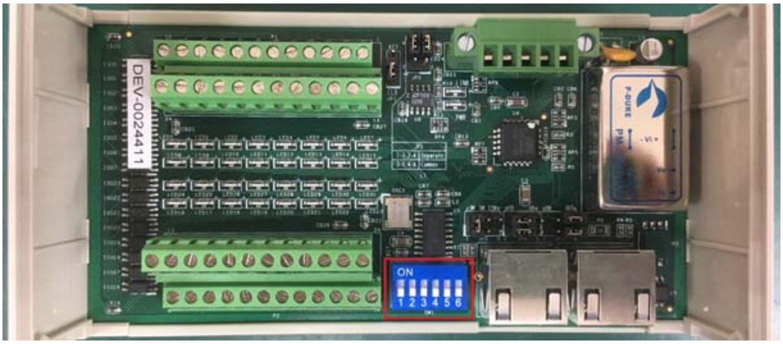
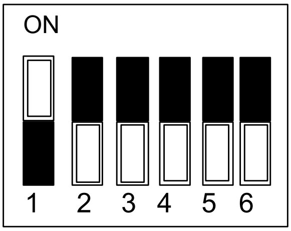


Figure 4-1: HSL Module DIP Switch Location

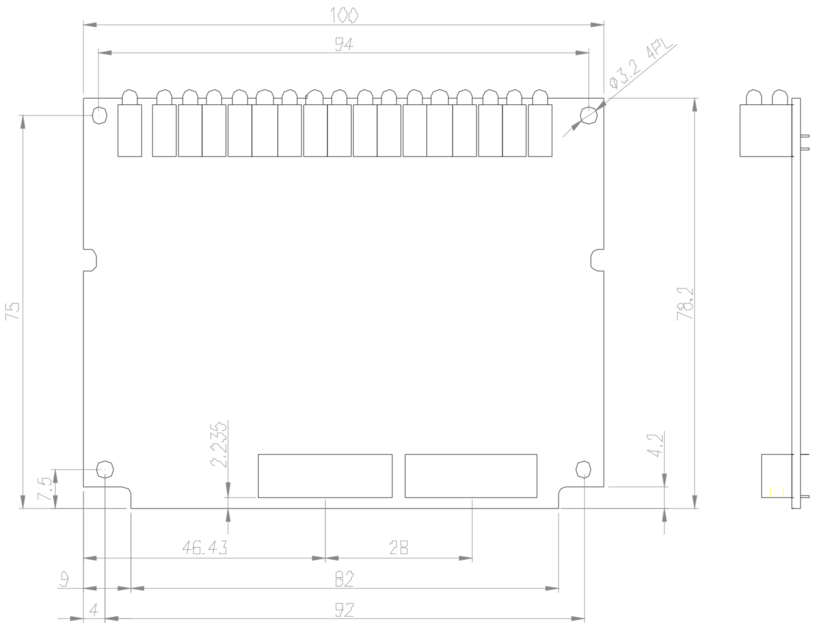


ON = 1
100000 address 1
010000 address 2
… …
011111 address 62
111111 address 63
OFF = 0

NOTE:

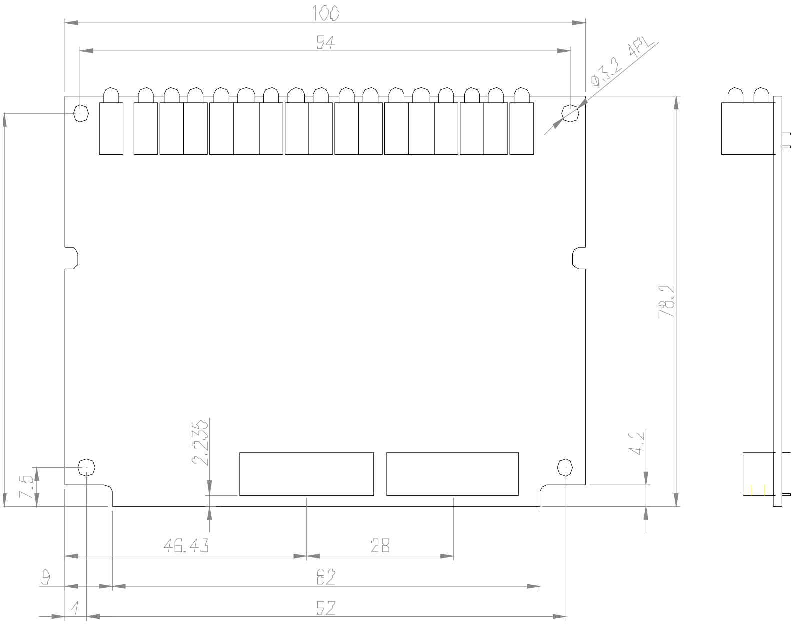
1. The address (or index) ‘0’ is reserved for the master.
2. HSL-DI32-M and HSL-DO32-M need two consecutive addresses starting from an odd number. For example, if the DIP switch is set to 3, it will occupy indexes 3 and 4.
3. HSL-AI16AO2-M-VV/AV needs two leap addresses in full-duplex mode. For example, if the DIP switch is set to 2, this module will occupy indexes 2 and 4.

4.2.4 Daughter Board/Module Dimensions



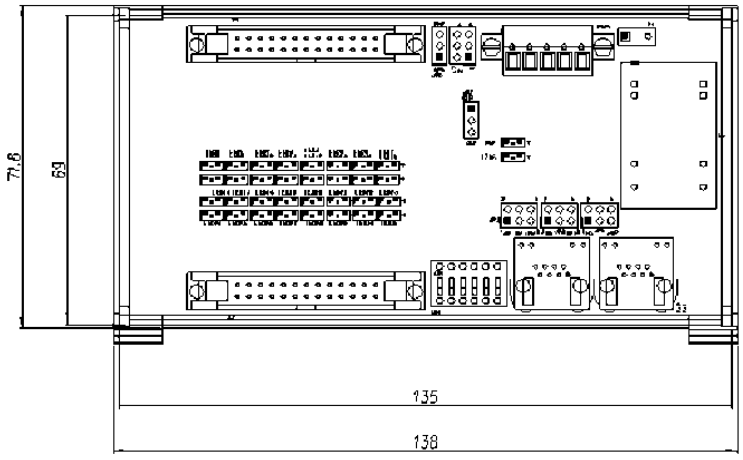
units: mm

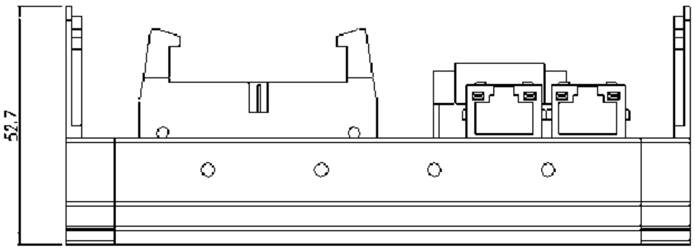
M: Daughter board with aluminum cover (125 mm × 80 mm)



units: mm

U: Low-profile I/O module (71.8 mm × 138 mm)

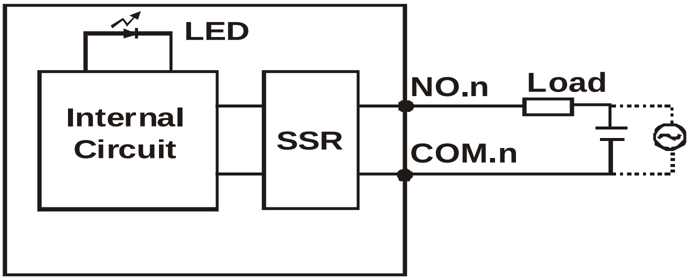




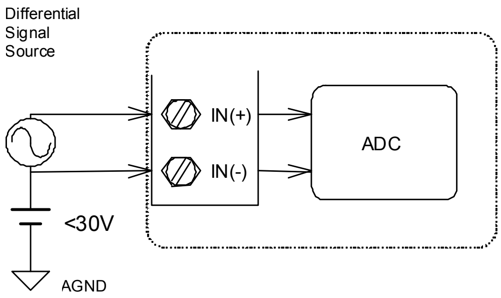
units: mm

4.2.5 Wiring Diagrams

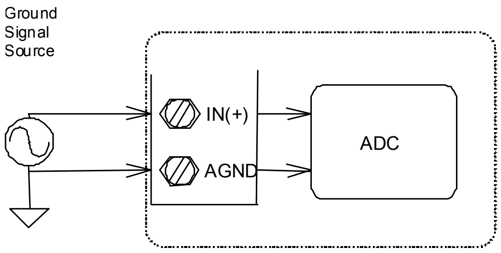
R (Relay Output):



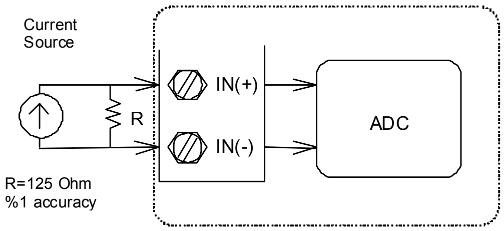
Analog Input (Differential Voltage Input):



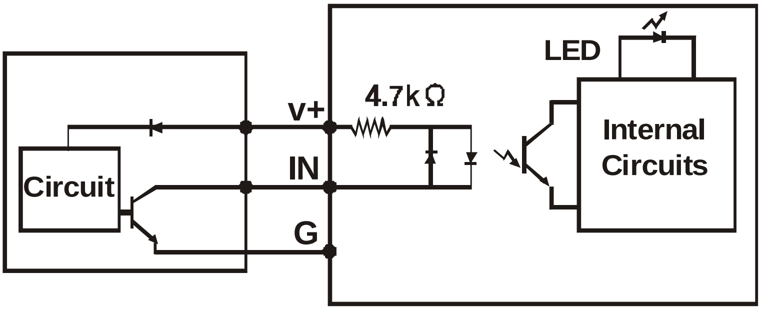
Analog Input (Single-Ended Voltage Input):



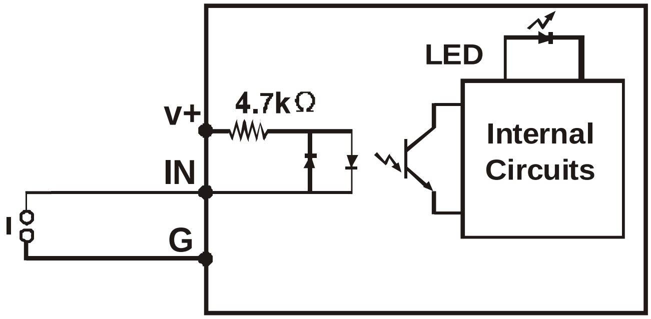
Analog Input (Current Measure):



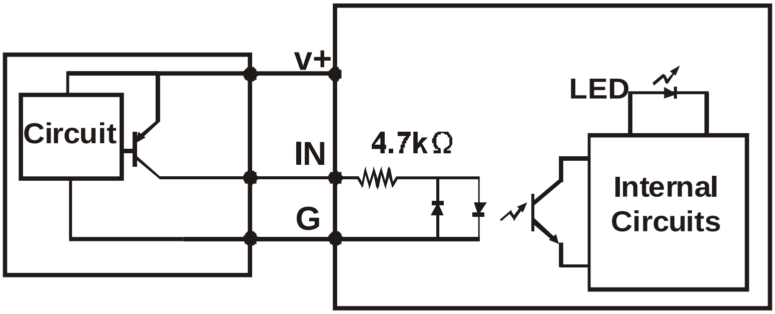
N (NPN Sinking-type Sensor Input):



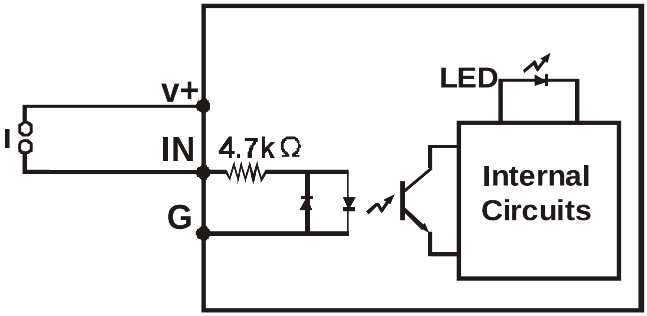
N (Dry Contact Input):



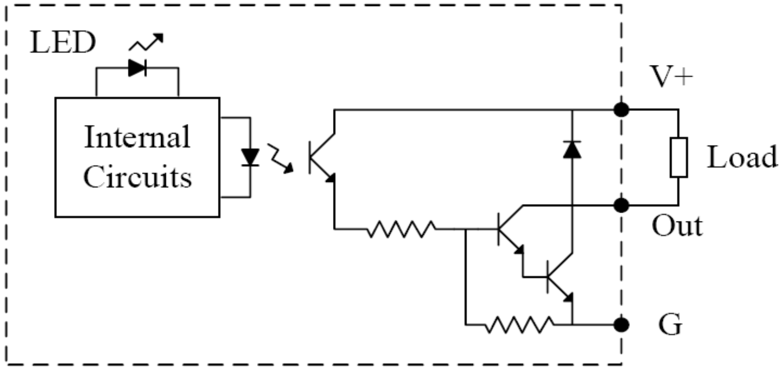
P (PNP Sourcing-type Sensor Input):



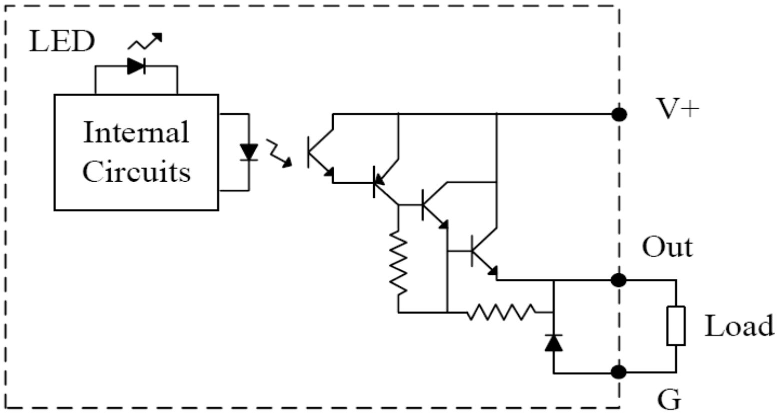
P (Wet Contact Input):



N (NPN Sinking Output):



P (PNP Sourcing Output):



4.2.6 Terminal Base Motion Control Modules

The terminal bases (TBs) include:

- HSL-TB32-U-DIN
- HSL-TB64-DIN
- HSL-TB32-M-DIN
- HSL-TB32-MD

Features

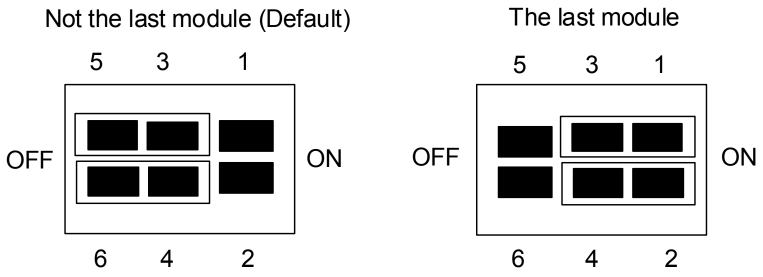
- Field I/O wiring connection for HSL I/O modules
- Screw or spring terminal for easy field wiring
- Power and ground included for each signal channel
- Interlocking design for rugged installation
- Power LED indicator
- DIN rail mount
- Onboard terminator resistor

General Descriptions

Series	Model	Description	Module Support
DB	HSL-TB32-U	(1) 32-channel direct connection terminal base (2) One DB slot	All HSL DB-series modules
	HSL-TB64	(1) 64-channel direct connection terminal base (2) Two DB slots	All HSL DB-series modules
M	HSL-TB32-M	32-channel direct connection terminal base for HSL M-series modules	All HSL M-series modules
	HSL-TB32-MD		

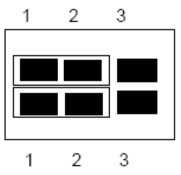
Jumper Settings

Since HSL is a serial transmission system, a terminator should be set at the end of the cable. Each TB has a jumper-selectable terminator on board. Only the last module needs to have the terminator enabled.



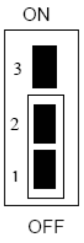
HSL-TB32-MD Jumper Settings

JP1,2 (External Power Option)



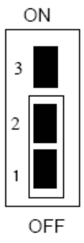
1, 2 short: Different Power (Default)
2, 3 short: Common Power

JP3 (Tx Terminal Resistor)



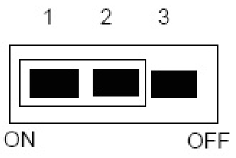
OFF is default setting

JP4 (Rx Terminal Resistor)



OFF is default setting

JP5 (Fuse Option)

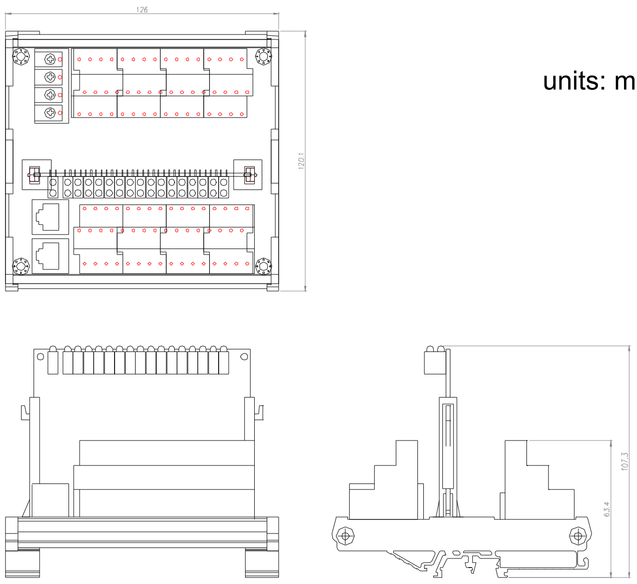


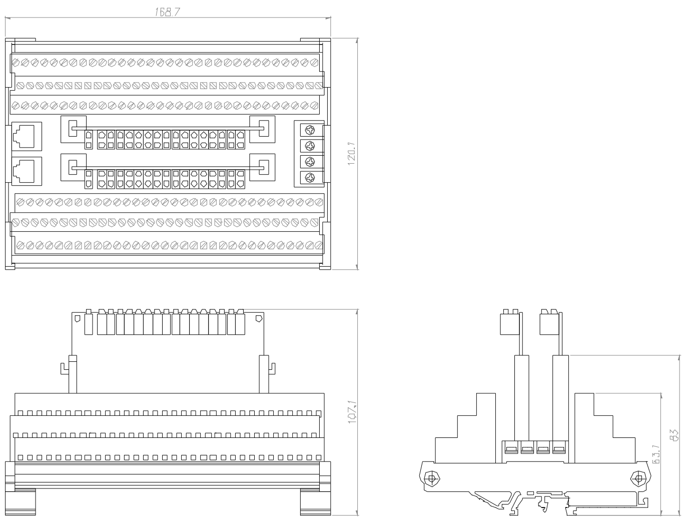
1, 2 short: With Fuse
2, 3 short: Without Fuse (Default)

Dimensions

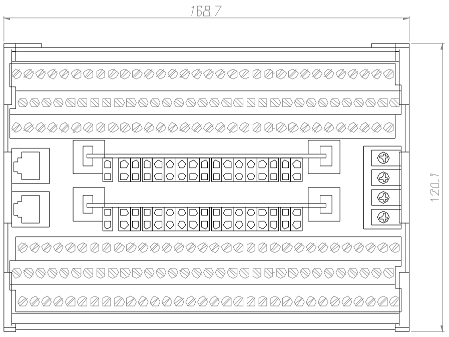
DB with HSL-TB32-U-DIN (126 × 120.1 × 107.3) mm

units: mm

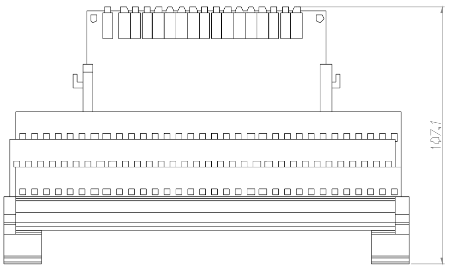


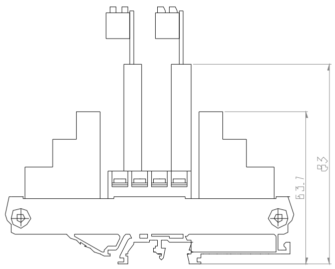


DB with HSL-TB64-DIN (168.7 × 120.1 × 107.3) mm



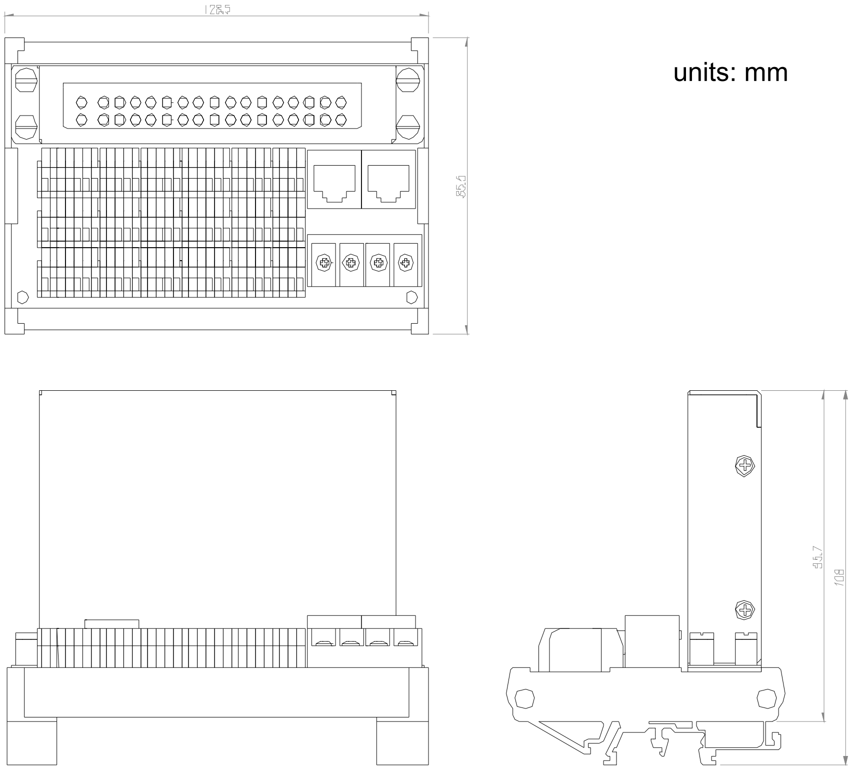
units: mm





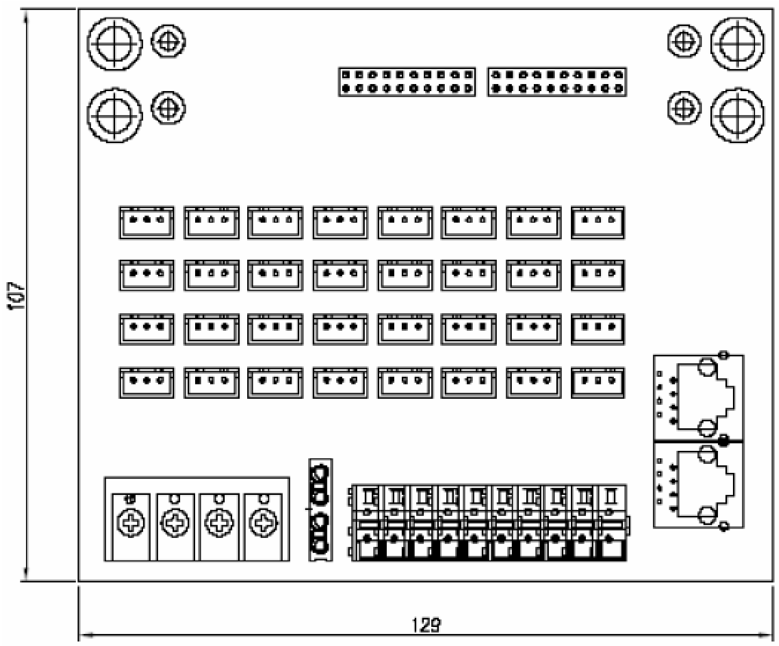
M module with HSL-TB32-M-DIN (128.5 × 85.5 × 108) mm

units: mm



HSL-TB32-MD (129 × 107) mm

units: mm



4.2.7 Managing Slave Indexes in an HSL Network

Before Proceeding

Before powering on the slave modules, ensure the DIP switch is properly set. Refer to Section 4.2.3 on page 28 and take special note of the following:

1. One master controller can connect to a maximum of 63 slave modules.
2. The more compact the slave addresses, the more efficiently the HSL system can work.
3. Discrete I/O and relay module rules:

Module	Slave Index Occupation	Transmission Mode	Transmission Speed
HSL-DI16DO16-M-NN/PN	1 (any address)	Full-duplex (fixed)	6 Mbps (fixed)
HSL-DI32-M-N/P	2 (consecutive from odd number)		
HSL-DO32-M-N/P			

4. Analog I/O and thermocoupling module rules:

Module	Slave Index Occupation	Transmission Mode	Transmission Speed
HSL-AI16AO2-M-VV	2 (leap number)	Full-duplex (fixed)	3/6/12 Mbps (selectable)
HSL-AO4-U			

5. Special rule: If installing only one HSL-AI16AO2-M-VV and the DIP switch is set to 1 (the HSL-AI16AO2-M-VV only supports full-duplex mode), the occupied indexes will be 1 and 3. You must assign a value of 4 to the parameter “MOD_No” of “APS_set_field_bus_slave_param()” to ensure correct communication (PCIe-7856 only).

Examples

The following examples are provided for user reference. All the modules involved are assumed to be in full-duplex mode.

Example 1

HSL-DI16DO16-UD×2, HSL-DI32-MN×2, and HSL-AI16AO2-VV×1 are installed (with all slave modules in full-duplex mode) under two different conditions:

Condition 1: HSL-AI16AO2-VV operating at 6 Mbps.

ADLINK suggests the following slave index configuration:

Item	DIP Switch	Index Occupation in HSL
HSL-DI32-M-N #1	1	1, 2
HSL-DI32-M-N #2	3	3, 4
HSL-AI16AO2-VV	5	5, 7
HSL-DI16DO16-UD #1	6	6
HSL-DI16DO16-UD #2	8	8

This is an example of a compact composition. The scan time will be 30.33 μs × 8 at 6 Mbps, full-duplex mode. Users can connect the modules with one master controller.

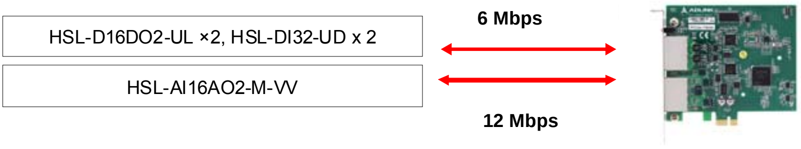
Condition 2: HSL-AI16AO2-VV×1 operating at 12 Mbps.

ADLINK recommends the following slave index configuration.

Item	DIP Switch	Index Occupation in HSL
HSL-DI32-M-N #1	1	1, 2
HSL-DI32-M-N #2	3	3, 4
HSL-DI16DO16-UD #1	5	5
HSL-DI16DO16-UD #2	6	6

This is another example of a compact composition. The scan time required is 30.33 μs × 6 at 6 Mbps, full-duplex mode. Users may connect these modules with one master controller.

The HSL-AI16AO2-M-VV module will connect to another master controller. The DIP switch of HSL-AI6AO2-M-VV will be assigned a value of 1.



Consequently, the cycle time of the first master controller is 30.33 µs × 6 and the cycle time of the second master controller is 45.5 µs at 12 Mbps, full-duplex mode.

Example 2

Two HSL-DI16DO16-UJ's, one HSLDI16DO16-M-NN, two HSL-DO32-M-N's and one HSL-AI16AO2-VV are installed (with all slave modules in full-duplex mode) under two different conditions:

Condition 1: HSL-AI16AO2-VV module operating at 6 Mbps.

ADLINK recommends the following slave index configuration:

Item	DIP Switch	Index Occupation in HSL
HSL-DO32-M-N #1	1	1, 2
HSL-DO32-M-N #2	3	3, 4
HSL-AI16AO2M-VV	5	5, 6
HSL-DI16DO16-UJ #1	7	7
HSL-DI16DO16-UJ #2	8	8
HSL-DI16DO16-M-NN	9	9

The scan time will be 30.33 µ ×17 at 6 Mbps, full-duplex mode. These modules can be connected with one master controller.

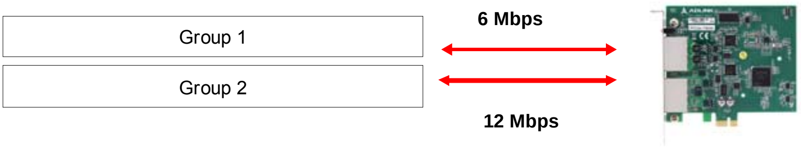
Condition 2: HSL-AI16AO2-VV module operating at 12 Mbps.

ADLINK recommends the following slave index configuration:

Group 1	DIP Switch	Index Occupation in HSL
HSL-DO32-M-N #1	1	1, 2
HSL-DO32-M-N #2	3	3, 4
HSL-DI16-UJ #1	5	5
HSL-DI16-UJ #2	6	6
HSL-DI16DO16-M-NN	7	7

The scan time required is 30.33 μs × 7. These modules may be connected with one master controller. The HSL-AI16AO2-M-VV module will connect to another master controller. The management table and illustration below are provided for reference.

Group 2	DIP Switch	Index Occupation in HSL
HSL-AI16AO2-M-VV	1	1, 2



The cycle time of the first master controller will be 30.33 μs × 7 while the cycle time of the second master controller will be 15.17 μs × 11 at 12 Mbps, full-duplex mode.

5 MotionCreatorPro 2 (MCP2)

After installing the hardware, it is necessary to correctly configure all cards and double-check the system before running. This chapter provides guidelines for establishing a control system and manually testing the PCIe-7856 to verify correct operation. The MCP2 software provides a simple yet powerful means to set up, configure, test, and debug a motion control system that uses PCIe-7856.

5.1 About MCP2

Before running MCP2, please note the following.

1. MCP2 was developed using BCB 6.0 and is available only for Windows systems with a screen resolution of 1024x768 or higher. It cannot be run under DOS.
2. The following files are required by the program:
   - ▷ MCP2.mdb, which stores parameters and graphics.
   - ▷ MCPro2.ini, which stores initialization settings.
3. MCP2 is a highly integrated program that supports many ADLINK motion control cards. Multiple cards can be used in one system.

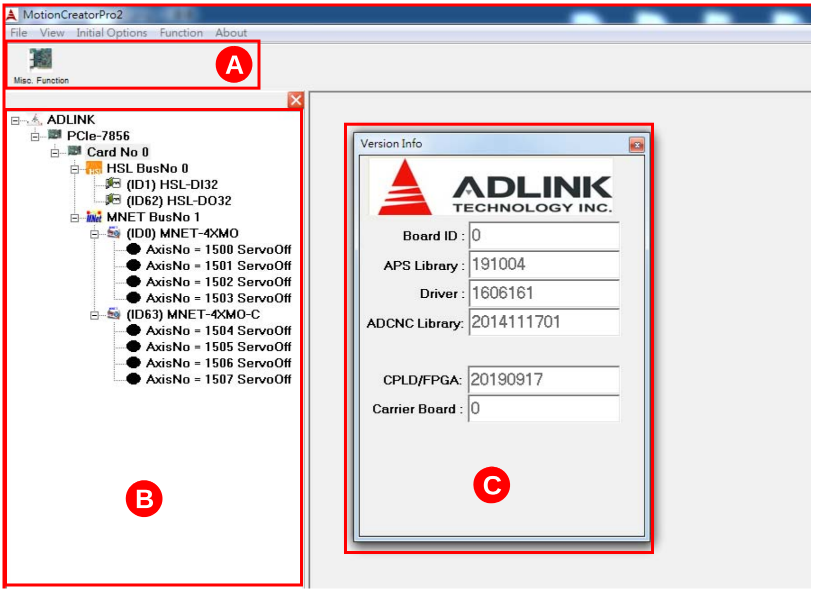
5.2 How to Run MCP2

After installing the software drivers for PCIe-7856, the MCP2 program will be located at “<chosen path>\MCP2.exe”. Double click the executable file to run the program.

5.3 MCP2 Features

5.3.1 Main Menu

Launching MCP2.exe will present the user with the main menu. Refer to the following images and descriptions for details on all the available features. To exit the program at any time, select “Exit” from the “File” menu.



```
ADLINK
  PCIe-7856
    Card No 0
      HSL BusNo 0
        (ID1) HSL-DI32
        (ID62) HSL-DO32
      MNET BusNo 1
        (ID0) MNET-4XMO
          AxisNo = 1500 ServoOff
          AxisNo = 1501 ServoOff
          AxisNo = 1502 ServoOff
          AxisNo = 1503 ServoOff
        (ID63) MNET-4XMO-C
          AxisNo = 1504 ServoOff
          AxisNo = 1505 ServoOff
          AxisNo = 1506 ServoOff
          AxisNo = 1507 ServoOff
```

ADLINK TECHNOLOGY INC.

Board ID :	0
APS Library :	191004
Driver :	1606161
ADCNC Library:	2014111701
CPLD/FPGA:	20190917
Carrier Board :	0

A. Icons for operation modes. Some will be active when a bus/motion item in the tree view is selected and some will be active when an axis item is selected.

- **Function Buttons**
  - **Configuration**

Button	Function	Description
	Axis/Board Configuration	Set axis/board parameters.

▷ Movement

Button	Function	Description
	Single Movement	Single-axis movement (PTP), including absolute and relative functions.
	Home Return Movement	Home return movement.
	Interpolation	Interpolation function.
	Sampling	Sampling function. Select this to set the source and draw its profile.
	2D Movement	Execute 2D motion.

▷ Field Bus

Button	Function	Description
	Field Bus Connect	Connect an MNET/HSL module. Select baud rate (to the right of the button) and connect.
	Field Bus Disconnect	Disconnect an MNET/HSL module.
	Field Bus Module Test	If properly connected, a module can be selected here for a module test.

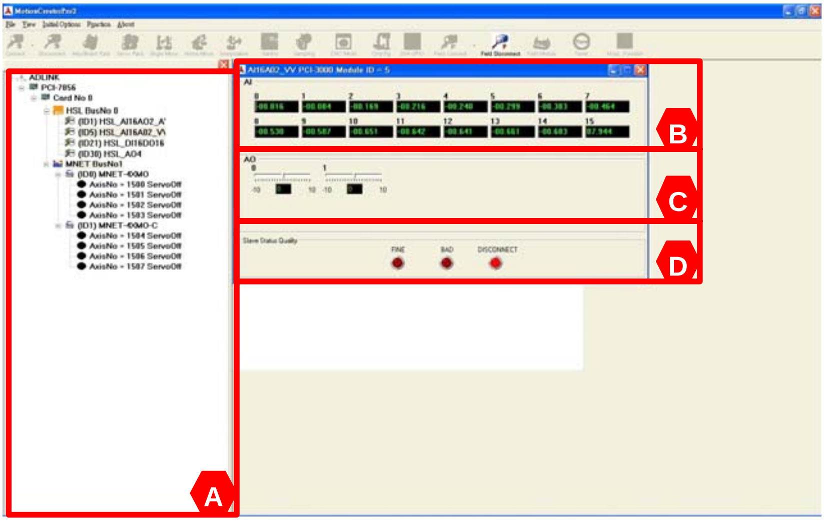
B. All automation products found by MCP2. The tree view will display motion axes as well as field bus I/O.

ICON	Function	Description
🟡 (Yellow)	Warning	Servo warning.
🔴 (Red)	Alarm	Servo alarm.
⚫ (Black)	Normal (Servo OFF)	No error and servo is off.
🟢 (Green)	Normal (Servo ON)	No error and servo is on.

C. Board information, including software, firmware, and hardware version numbers.

5.3.2 HSL Distributed I/O Manager

This page can be used to test the HSL system and slave modules. After executing the I/O management page, the main operation window shown below will appear. You can select the module for testing in the tree list of left window. The corresponding ID will also appear with each module. For example, the following figure shows the management pane for the HSL-AI16AO2-VV module. Analog input information is presented in this window and you can use the sliding bar to control the analog output.



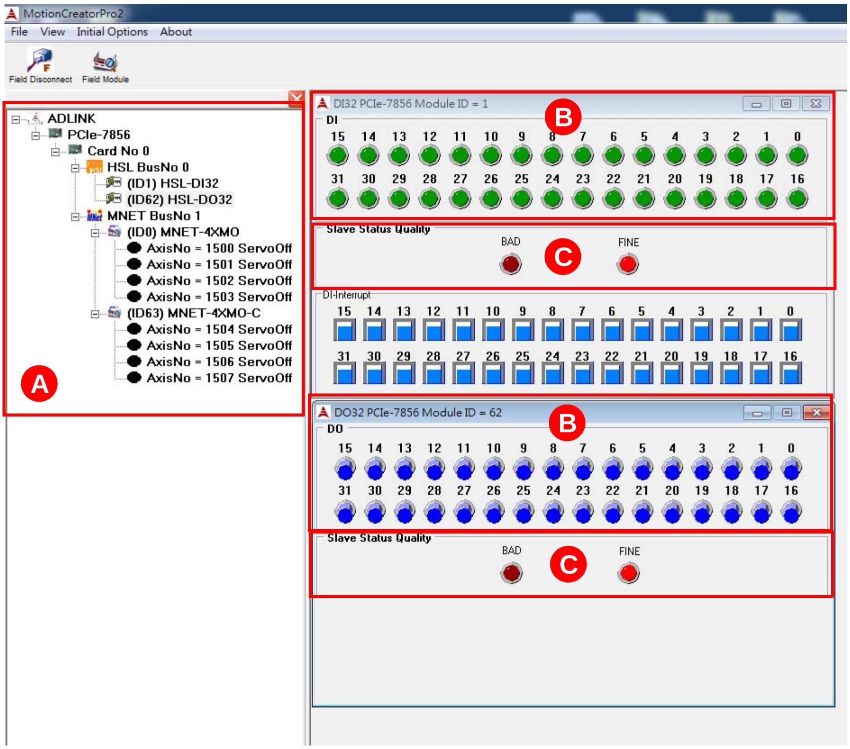
Operation Instructions

A. Tree view of all HSL and MNET modules.

B. Analog input presentation.

C. Analog output control panel.

D. Check the communication status of each module.



```
ADLINK
  PCIe-7856
    Card No 0
      HSL BusNo 0
        (ID1) HSL-DI32
        (ID62) HSL-DO32
      MNET BusNo 1
        (ID0) MNET-4XMO
          AxisNo = 1500 ServoOff
          AxisNo = 1501 ServoOff
          AxisNo = 1502 ServoOff
          AxisNo = 1503 ServoOff
        (ID63) MNET-4XMO-C
          AxisNo = 1504 ServoOff
          AxisNo = 1505 ServoOff
          AxisNo = 1506 ServoOff
          AxisNo = 1507 ServoOff
```

Operation Instructions

A. Tree view of all HSL and MNET modules.

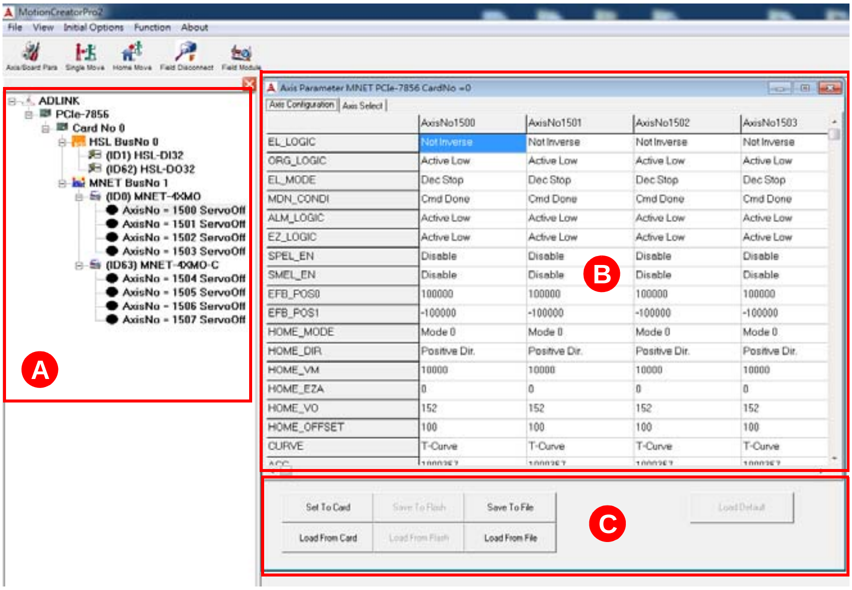
B. Digital input representation or digital output control.

C. Check the communication status of each module.

5.3.3 MNET Distributed Motion Manager

The Motionnet (MNET) manager offers several motion operations, including single-axis movement, home return, axis parameter setting, etc.

Parameter Management



```
ADLINK
  PCIe-7856
    Card No 0
      HSL BusNo 0
        (ID1) HSL-DI32
        (ID62) HSL-DO32
      MNET BusNo 1
        (ID0) MNET-4XMO
          AxisNo = 1500 ServoOff
          AxisNo = 1501 ServoOff
          AxisNo = 1502 ServoOff
          AxisNo = 1503 ServoOff
        (ID63) MNET-4XMO-C
          AxisNo = 1504 ServoOff
          AxisNo = 1505 ServoOff
          AxisNo = 1506 ServoOff
          AxisNo = 1507 ServoOff
```

	AxisNo1500	AxisNo1501	AxisNo1502	AxisNo1503
EL_LOGIC	Not Inverse	Not Inverse	Not Inverse	Not Inverse
ORG_LOGIC	Active Low	Active Low	Active Low	Active Low
EL_MODE	Dec Stop	Dec Stop	Dec Stop	Dec Stop
MDN_CONDI	Cmd Done	Cmd Done	Cmd Done	Cmd Done
ALM_LOGIC	Active Low	Active Low	Active Low	Active Low
EZ_LOGIC	Active Low	Active Low	Active Low	Active Low
SPEL_EN	Disable	Disable	Disable	Disable
SMEL_EN	Disable	Disable	Disable	Disable
EFB_POS0	100000	100000	100000	100000
EFB_POS1	-100000	-100000	-100000	-100000
HOME_MODE	Mode 0	Mode 0	Mode 0	Mode 0
HOME_DIR	Positive Dir.	Positive Dir.	Positive Dir.	Positive Dir.
HOME_VM	10000	10000	10000	10000
HOME_EZA	0	0	0	0
HOME_VO	152	152	152	152
HOME_OFFSET	100	100	100	100
CURVE	T-Curve	T-Curve	T-Curve	T-Curve

Operation Instructions

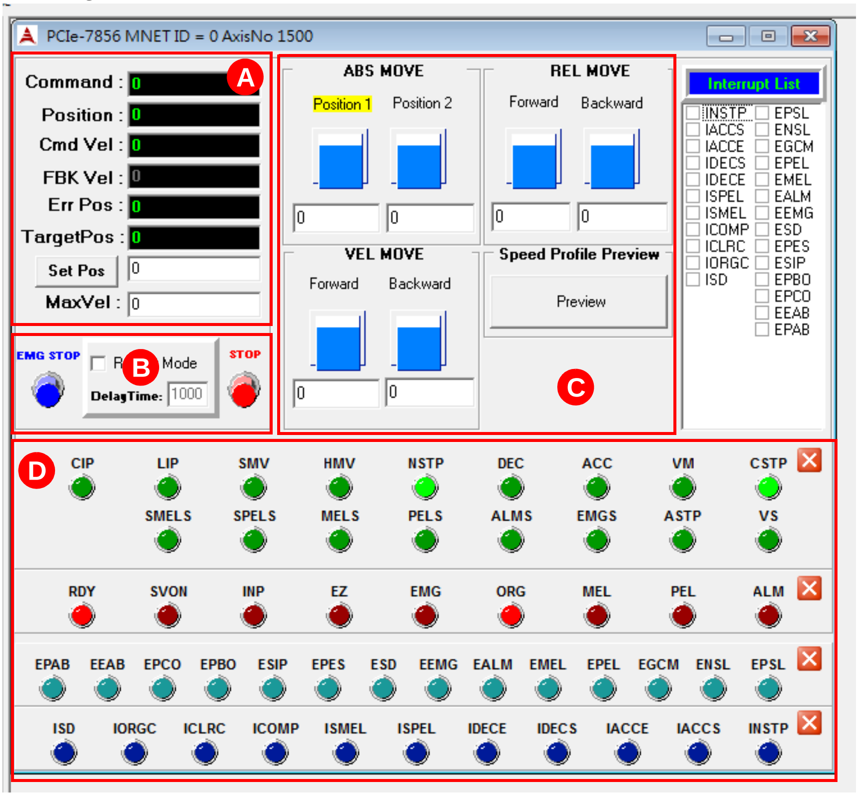
A. Tree view of all HSL and MNET modules.

B. Parameter values of each axis.

C. Parameter management buttons for saving/loading parameters in various ways. “Set To Card” must be clicked to activate any changes made to this table.

Tip: *Right-click on a parameter to apply it to all other axes*.

Single-axis Movement


PCIe-7856 MNET ID = 0 AxisNo 1500
A
Command : 0
Position : 0
Cmd Vel : 0
FBK Vel : 0
Err Pos : 0
TargetPos : 0
Set Pos 0
MaxVel : 0
B
EMG STOP
Repeat Mode
DelayTime: 1000
STOP
ABS MOVE
Position 1
Position 2
REL MOVE
Forward
Backward
VEL MOVE
Forward
Backward
Speed Profile Preview
Preview
C
Interrupt List
INSTP IACCS IACCE IDECS IDECE ISPEL ISMEL ICOMP ICLRC IORGC ISD
EPSL ENSL EGCM EPEL EMEL EALM EEMG ESD EPES ESIP EPBO EPCO EEAB EPAB
D
CIP LIP SMV HMV NSTP DEC ACC VM CSTP
SMELS SPELS MELS PELS ALMS EMGS ASTP VS
RDY SVON INP EZ EMG ORG MEL PEL ALM
EPAB EEAB EPCO EPBO ESIP EPES ESD EEMG EALM EMEL EPEL EGCM ENSL EPSL
ISD IORGC ICLRC ICOMP ISMEL ISPEL IDECE IDECS IACCE IACCS INSTP

Operation Instructions:

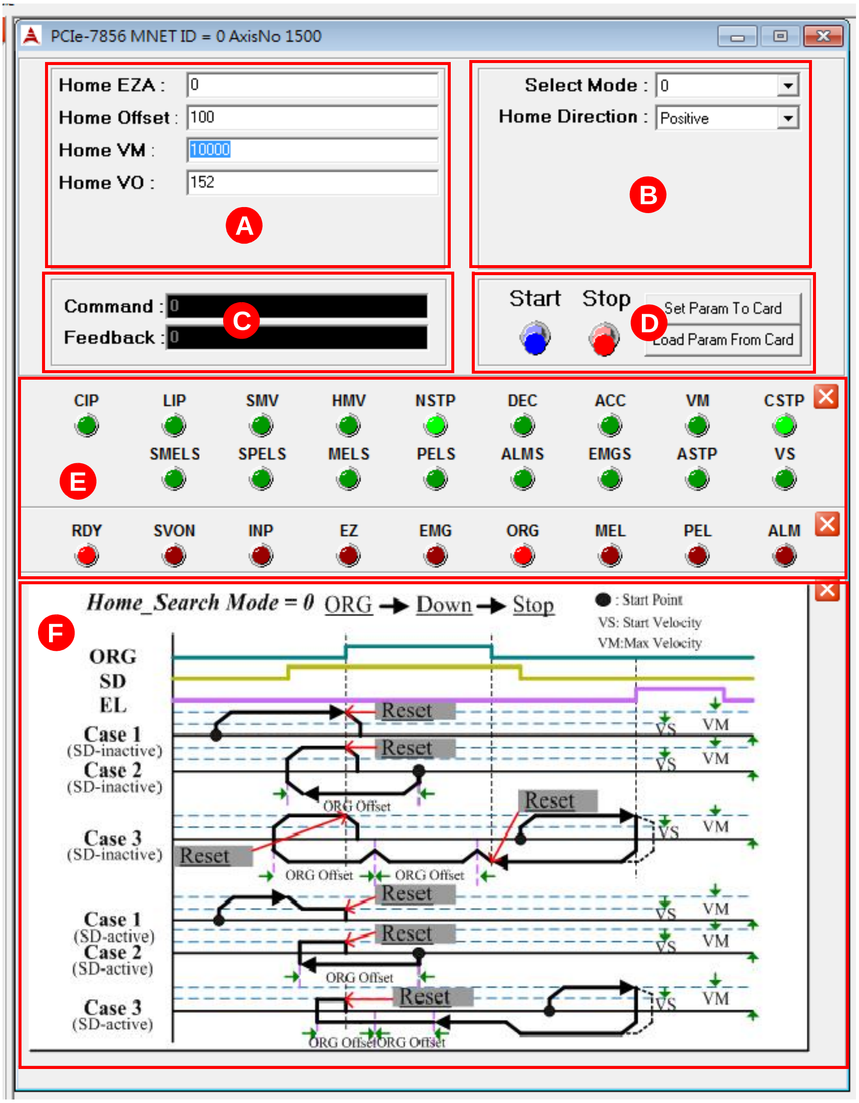
A. Command, feedback, error, and target position information. Command and feedback speed information. The minimum speed value may be limited by speed calculation cycle time for low speed display.

B. Optional operation settings and buttons. Repeat Mode can be used in both Relative and Absolute mode. The axes will move between two positions or forward/backward distance cyclically. You can set the delay time (in milliseconds) between each move. The minimum value is 1 ms. The stop button is for relative, absolute, and velocity modes.

C. Operation buttons and settings for 3 modes. You can switch operation between relative, absolute, and velocity modes. Before operation, mode parameters must be set, such as positions 1 and 2, forward/backward distance, and forward/backward velocity. Set “MaxVel” before executing relative or absolute mode. While using jog mode, the other three modes will be disabled.

D. Motion status, I/O status, and interrupt status display area.

Home Return



PCIe-7856 MNET ID = 0 AxisNo 1500

Home EZA : 0
Home Offset : 100
Home VM : 10000
Home VO : 152

Select Mode : 0
Home Direction : Positive

Command : 0
Feedback : 0

Operation Instructions

A. Speed parameter of the homing profile. Refer to area F.

B. Mode setting for the homing function, selectable via pull-down menu.

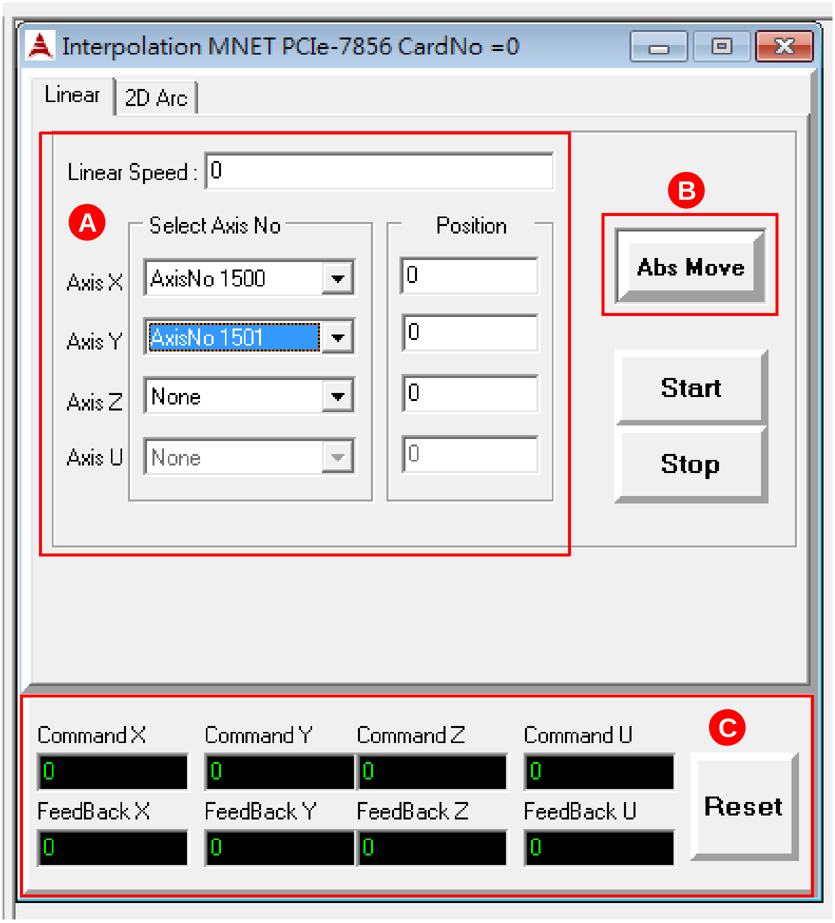
C. Command and position information while homing. After homing is complete, command will reset to zero.

D. Buttons for “starting” and “stopping/aborting” the homing function.

E. Motion and I/O status while homing.

F. Timing chart for the homing function.

Interpolation



Interpolation MNET PCIe-7856 CardNo =0

Linear | 2D Arc

Linear Speed : 0

A

	Select Axis No	Position
Axis X	AxisNo 1500	0
Axis Y	AxisNo 1501	0
Axis Z	None	0
Axis U	None	0

B

Abs Move

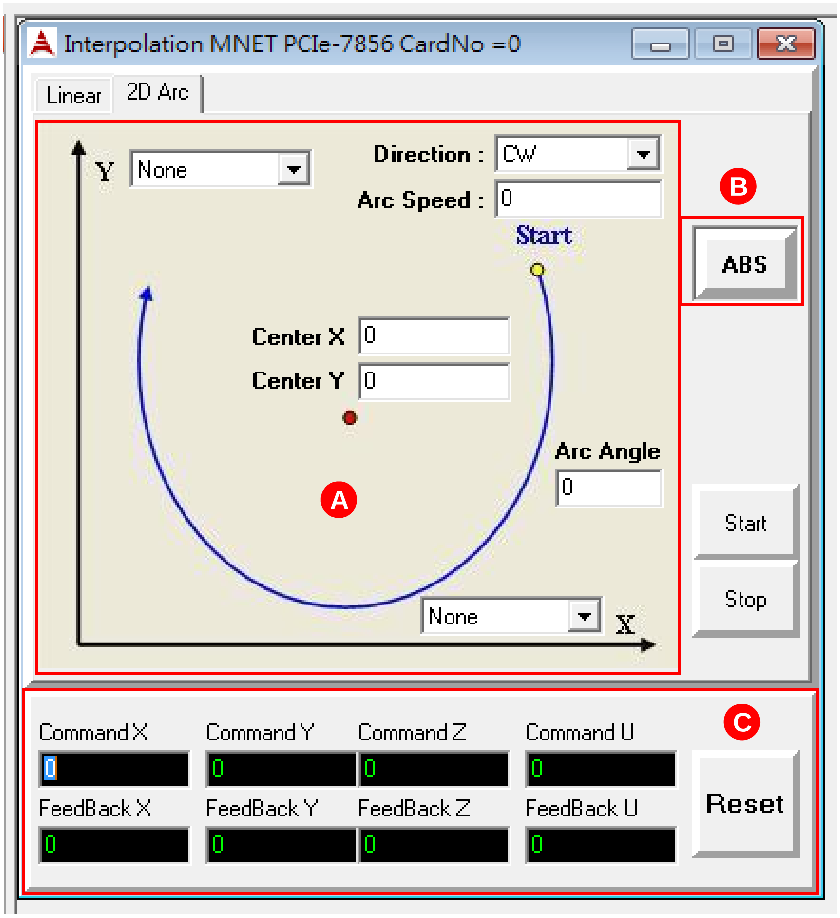
Start

Stop

C

Command X	Command Y	Command Z	Command U
0	0	0	0
FeedBack X	FeedBack Y	FeedBack Z	FeedBack U
0	0	0	0

Reset


Interpolation MNET PCIe-7856 CardNo =0
Linear
2D Arc
Y
None
Direction :
CW
Arc Speed :
0
Start
Center X
0
Center Y
0
Arc Angle
0
A
None
X
B
ABS
Start
Stop
C
Command X
0
Command Y
0
Command Z
0
Command U
0
FeedBack X
0
FeedBack Y
0
FeedBack Z
0
FeedBack U
0
Reset

Operation Instructions

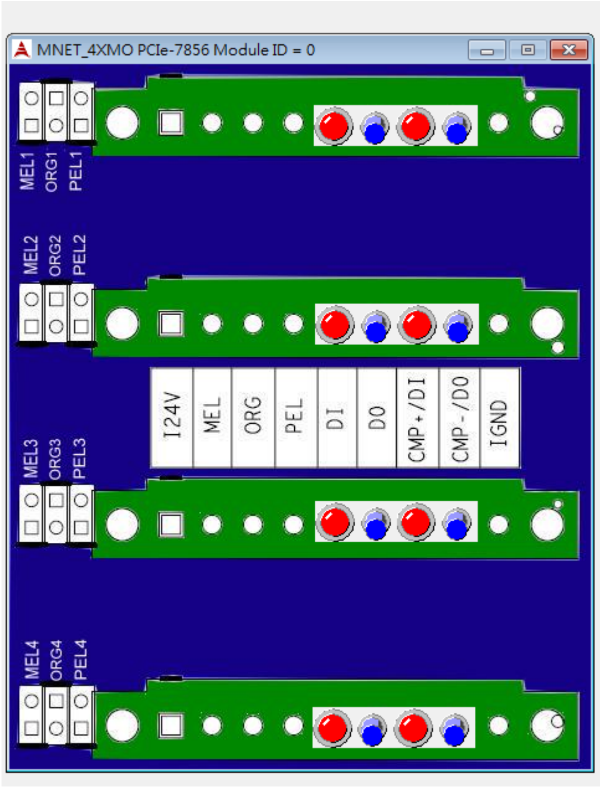
A. Interpolation axis selection and operation parameters, including center position in Arc mode or target position in Linear mode. The arc angle can be larger than 360.

B. Absolute or relative interpolation mode selection. In Arc mode, it relates to the center position. In Linear mode, it relates to the target position.

C. Command and position information. In Arc mode, only two will be active.

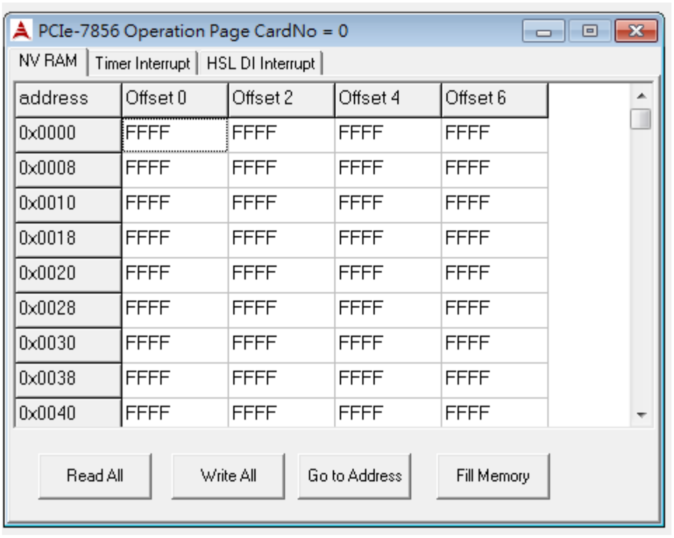
Dedicated Motion I/O status

Conveniently monitor and configure motion I/O channels.



NVRAM Read/Write Window

The PCIe-7856 is equipped with 32 kB of NVRAM. The read/write window provides direct access to this non-volatile memory.



address	Offset 0	Offset 2	Offset 4	Offset 6
0x0000	FFFF	FFFF	FFFF	FFFF
0x0008	FFFF	FFFF	FFFF	FFFF
0x0010	FFFF	FFFF	FFFF	FFFF
0x0018	FFFF	FFFF	FFFF	FFFF
0x0020	FFFF	FFFF	FFFF	FFFF
0x0028	FFFF	FFFF	FFFF	FFFF
0x0030	FFFF	FFFF	FFFF	FFFF
0x0038	FFFF	FFFF	FFFF	FFFF
0x0040	FFFF	FFFF	FFFF	FFFF

5.4 MCP2 Error Codes

The meaning of error codes that may be returned by the MCP2 program are as follows:

- (-1) Operation system type mismatch
- (-2) Open device driver failed; driver interface creation failed
- (-3) Insufficient memory
- (-4) Cards not initialized
- (-5) Cards not found (no card in your system)
- (-6) Duplicate card IDs
- (-7) Cards have been initialized, check if different software has been enabled on same hardware device
- (-8) Card interrupt events not enabled or not initialized
- (-9) Function timed out
- (-10) Invalid function input parameters
- (-11) Set data to EEPROM failed
- (-12) Get data from EEPROM failed
- (-13) Function unavailable in this step; function unsupported by device; internal process failed
- (-14) Firmware error: please reboot the system
- (-15) Previous command is in process
- (-16) Duplicate axis ID
- (-17) Slave module not found
- (-18) Number of modules insufficient
- (-51) Set data to SRAM failed
- (-52) Get data from SRAM failed
- (-1000) Invalid INT value or WIN32_API error: please contact ADLINK support staff

6 Scan Time Table

6.1 Full-duplex Mode

The following table shows minimum scan times in full-duplex mode at different transmission speeds.

Slave Index Number	Cycle Time at 2.5 Mbps	Cycle Time at 5.0 Mbps	Cycle Time at 10 Mbps	Cycle Time at 20 Mbps
Base Unit	60.67 µs	30.33 µs	15.17 µs	15.17 µs
< 3	182.00 µs	91.00 µs	45.50 µs	45.50 µs
5	303.33 µs	151.67 µs	75.83 µs	75.83 µs
10	606.67 µs	303.33 µs	151.67 µs	151.67 µs
20	1.213 ms	606.67 µs	303.33 µs	303.33 µs
30	1.820 ms	910.00 µs	455.00 µs	455.00 µs
40	2.427 ms	1.213 ms	606.67 µs	606.67 µs
50	3.033 ms	1.516 ms	758.33 µs	758.33 µs
60	3.640 ms	1.820 ms	910.00 µs	910.00 µs
63	3.822 ms	1.911 ms	955.50 µs	955.50 µs

This page intentionally left blank.

7 HSL LinkMaster Utility

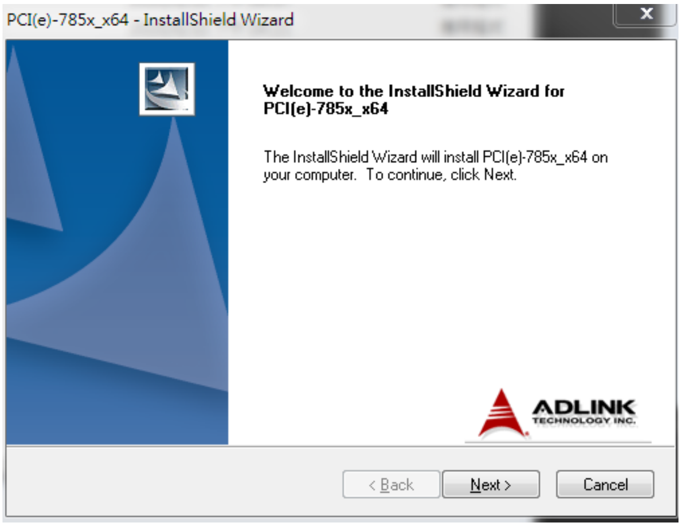
After installing the master controller and slave modules, you are now ready to install the HSL driver and the LinkMaster utility for system testing and debugging. This utility features a user-friendly interface that enables you to easily test I/O statuses, including read/write the I/O data, calibration and motion control. It is recommended that you use this utility before implementing the whole system.

7.1 Software Installation

You can install the HSL drivers from the ADLINK website at https://www.adlinktech.com/Products/Motion_Control/DistributedMotion-Control/PCIe-7856_7853?lang=zh-hant.

To install the HSL drivers:

1. Double-click the SETUP.exe file. The installation window appears. Click **Next**.


PCI(e)-785x_x64 - InstallShield Wizard

Welcome to the InstallShield Wizard for PCI(e)-785x_x64

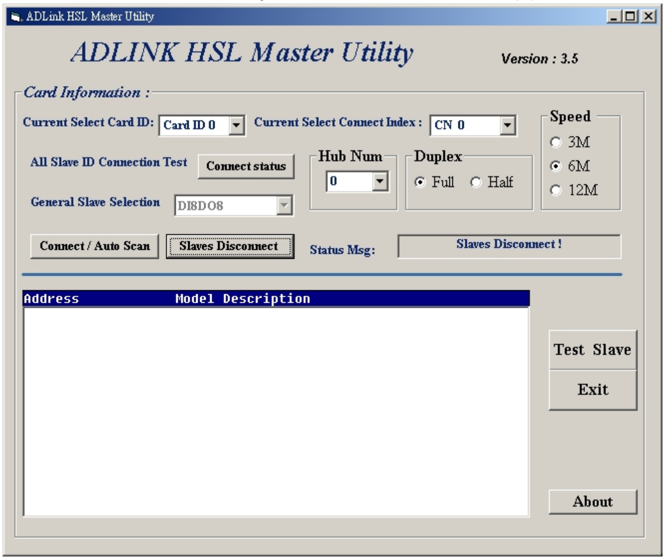
The InstallShield Wizard will install PCI(e)-785x_x64 on your computer. To continue, click Next.

2. Follow the on-screen instructions.
3. Restart the system when the installation process is completed.

7.2 ADLINK HSL LinkMaster Utility

7.2.1 Launching the LinkMaster Utility

After installing the drivers, click Start > PCI-7853 > LinkMaster to launch the LinkMaster utility. The main window appears.



Version : 3.5

Current Select Card ID: Card ID 0

Current Select Connect Index : CN 0

Speed: 3M [✓ 6M] 12M

Hub Num: 0

Duplex: [✓ Full] Half

General Slave Selection: DI8DO8

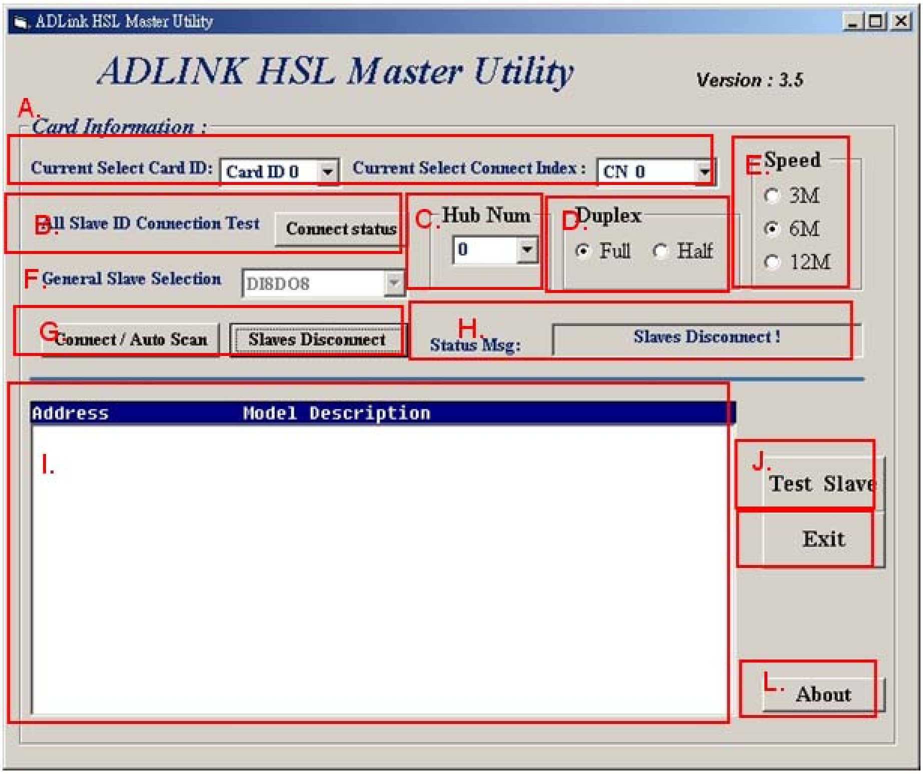
Status Msg: Slaves Disconnect !

7.2.2 Before you proceed

1. LinkMaster is a testing and debugging program based on VB 6.0 and is only available for Windows® 7/10 environments with a monitor that has a screen resolution of 800x600 or higher. The utility does not support DOS environment.
2. The LinkMaster version control may be found on the top-right corner of the main window.
3. Any slave modules may be tested with this utility, including discrete I/O, analog I/O, thermocouple module, and motion modules. For motion control utility and manipulation, refer to the HSL-4XMO user's manual.

7.2.3 LinkMaster Utility Introduction

Below is the LinkMaster main user interface labeled according to function.

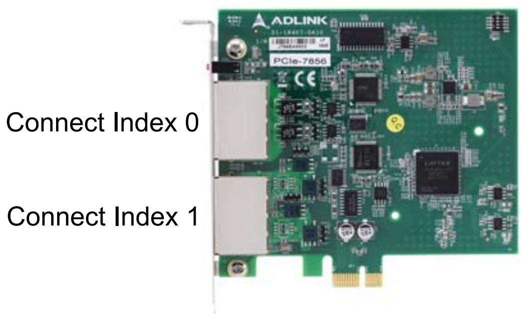


- A. Select card
- B. Network quality test
- C. Set hub number (Only for 7853)
- D. Set duplex mode (Only for 7853)
- E. Set speed mode (Only for 7853)
- F. General slave selection
- G. Auto scan slave modules
- H. Show software information
- I. Show module information

- J. Test slave module
- K. Exit motion creator
- L. Version information

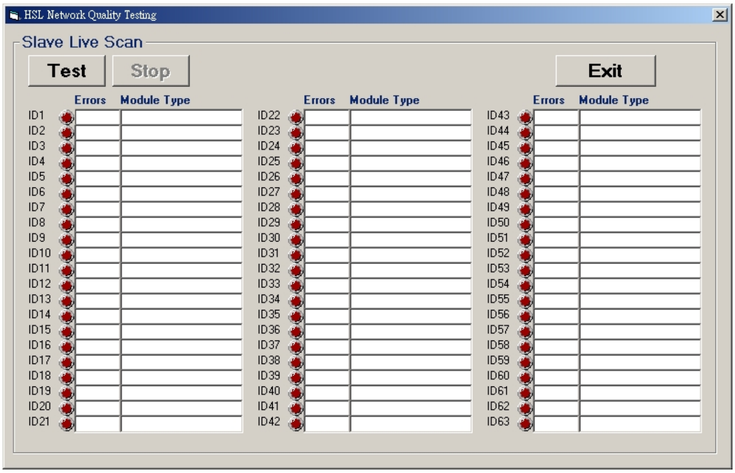
Below are descriptions of the main interface buttons.

1. **Current Select Card ID**. When LinkMaster is activated, it searches all HSL master control cards installed in the system, such as PCIe-7853. Every card shows its index (ID) ranging from 0 to ?. You can use this function to specify which card you want to operate.
2. **Current Select Connect Index**. For single master controller such as PCI-7853, the connect index is 0. Refer to the diagram below.



3. **ALL Slave ID Connection Test**. The screen capture below shows a live scan of all I/O modules for network quality test. The LinkMaster lets you check the network environment.

Start the test by clicking on the **Test** button. Press Stop to stop scanning. When you start the test, the utility continuously tests each ID and shows the module type to left-column labels. Right-column labels show the counter for communication error.



4. **Connect/Auto Scan**. Clicking this button allows the utility to scan all slave modules connected to the master card with specified connect index. The utility shows all the slave modules' information including the address and slave type within the 9th block.
5. **Slaves Disconnect**. Click this to stop the utility from scanning all the slave modules and to disconnect them.
6. **Status Msg**. Checks if the slave modules are connected or disconnected.

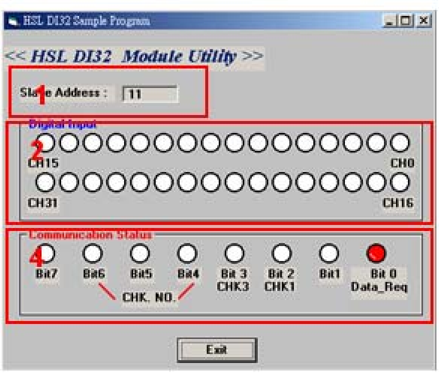
   Test Slave: While all connected slave modules list in 9th block, you can use this function to activate the testing dialog. For example, when you connect the HSL-DI16DO16-M-NN, you will see this module from the screen. Clicking on it will show a window from where you can test and debug the modules.
7. **Exit**. Click to close the utility.
8. **About**. Shows the DLL version information.

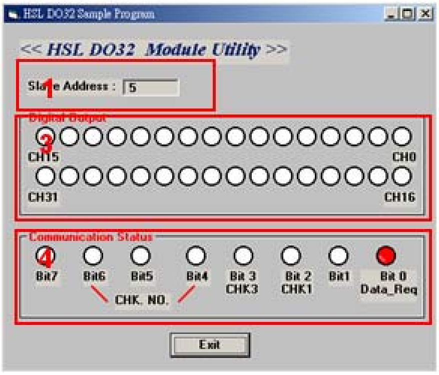
The succeeding sections outline the usage of the slave module utility.

7.2.4 HSL-DI16DO16 Utility

1. **Slave Address**. Shows the slave index occupied by the module.
2. **Digital Input**. A white circle indicates no digital input; a red icon indicates that the digital input is not activated.
3. **Digital Output**. Click on the icon to activate the digital output. Red icon indicates that the digital output is turned on, and vice-versa.
4. **Slave Status**: Shows the communication status between the slave module and the master card. The functions definition are enumerated below.
   - ▷ Bit 0 is Data_Req bit.
   - ▷ Bit 2 is for CHK1. When Bit2 is equal to 1, a communication error occurred once).
   - ▷ Bit 3 is for CHK3. When Bit3 is equal to 1, a communication error occurred three times.
   - ▷ Bit 4, Bit 5 and Bit 6 bits are for CHK7. WhenBit4, Bit5, and Bit6 are all equal to 1, a communication error occurred seven times.

7.2.5 HSL-DI32 and HSL-DO32 Utility





1. **Slave Address**. Shows the slave index occupied by the module. These modules occupy two slave indexes starting from an odd number. For example, when you adjust the DIP switch to 3, the modules are assigned indexes 3 and 5.
2. **Digital Input**. A white circle indicates no digital input; a red icon indicates that the digital input is not activated.
3. **Digital Output**. Click on the icon to activate the digital output. Red icon indicates that the digital output is turned on, and vice-versa.
4. **Slave Status**: Shows the communication status between the slave module and the master card. The functions definition are enumerated below.
   - ▷ Bit 0 is Data_Req bit.
   - ▷ Bit 2 is for CHK1. When Bit2 is equal to 1, a communication error occurred once).
   - ▷ Bit 3 is for CHK3. When Bit3 is equal to 1, a communication error occurred three times.
   - ▷ Bit 4, Bit 5 and Bit 6 bits are for CHK7. WhenBit4, Bit5, and Bit6 are all equal to 1, a communication error occurred seven times.

7.2.6 HSL-4XMO Utility

Refer to the HSL-4XMO user’s manual.

8 HSL Function Library

This chapter describes the functions for developing programs in C, C++, or Visual Basic.

8.1 List of Functions

This section presents all the functions. The function prototypes and common data types are declared in HSL. It is recommended that you use these data types in your application programs. The following table shows the data type names and their ranges.

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

All HSL function calls were revised. Refer to the mapping table in Appendix B. All function calls have the same prefix HSL_. The function belonging to a system level purpose has the following form:

HSL_{action_name}. e.g. HSL_initial().

If they belong to a discrete I/O module purpose, the function is as follows:

HSL_D_{action_name}. e.g. HSL_D_read_input()

If they belong to an analog I/O module purpose, the function is as follows.

HSL_A_{action_name}. e.g. HSL_A_write_output().

If they belong to a motion control module purpose, the function is as follows.

HSL_M_{action_name}. e.g. HSL_M_start_tr_move().

For the motion control library description, refer to the HSL-4XMO function library manual. This section contains the system level function, discrete I/O control, and analog I/O control.

Initialization and System Information, section 8.2

Function Name	Description
HSL_initial	Master card initialization
HSL_intial_sw	Initialize by system automatically (sw_enable=0) or manually via the S1 dip switch (sw_enable=1) (7853/54 only)
HSL_close	Release all resources occupied by master card
HSL_start	Start to scan all the slave modules connected to master card
HSL_auto_start	Start to scan and automatically detect all the slave modules connected to master card
HSL_stop	Stop scanning the connected slave modules
HSL_set_scan_condition	Set scanning conditions (only for 7853/54)
HSL_get_scan_condition	Get scanning conditions (only for 7853/54)
HSL_connect_status	Get the communication status of the specified slave module
HSL_slave_live	Get the module status of the slave module
HSL_get_irq_channel	Get the IRQ occupied by master card

Timer Control, section 8.4

Function Name	Description
HSL_enable_timer_interrupt	Enable timer interrupt of master card (For 7851/52)
HSL_disable_timer_interrupt	Disable timer interrupt of master card (For 7851/52)
HSL_set_timer	Set the resolution of timer (For 7851/52)
HSL_set_int_timer	Set the timer parameters (For 7853/54)
HSL_set_int_timer_enable	Enable/Disable timer interrupt of master card (For 7853/54)
HSL_wait_timer_interrupt	Wait timer event (For 7853/54)

Discrete I/O, section 8.5

Function Name	Description
HSL_D_read_input	Read back all discrete I/O with unsigned 32-bit
HSL_D_read_channel_input	Read back discrete I/O by channel selection
HSL_D_write_output	Write all discrete I/O with unsigned 32-bit
HSL_D_write_channel_output	Write discrete I/O by channel selection
HSL_D_read_ouput	Read back the output value stored in RAM
HSL_D_read_all_slave_input	Read back all inputs of slave modules
HSL_D_write_all_slave_output	Write all outputs of slave modules
HSL_D_set_input_logic	Set the logic of digital input
HSL_D_set_output_logic	Set the logic of digital output
HSL_D_set_int_renewal_type	Set DI renewal check type (Only for 7853/54)
HSL_D_set_int_renewal_bit	Set the data bits of DI renewal check for each slave (Only for 7853/54)
HSL_D_set_int_control	Set DI interrupt enable or disable (Only for 7853/54)
HSL_D_wait_di_interrupt	Wait DI renewal event(Only for 7853/54)

Analog I/O, section 8.6

Function Name	Description
HSL_A_start_read	Start A/D conversion.
HSL_A_stop_read	Stop A/D conversion
HSL_A_set_signal_range	Set the signal range of analog input channels
HSL_A_get_signal_range	Get the signal range of analog input channels
HSL_A_get_input_mode	Get the signal input mode
HSL_A_set_last_channel	Set the last channel of analog input channels
HSL_A_get_last_channel	Get the last channel of analog input channels
HSL_A_read_input	Read back the value of analog input channels
HSL_A_write_output	Send out the analog output
HSL_A_read_output	Read back the analog output data
HSL_A_sync_rw	Read and write the data synchronously
HSL_A_get_version	Get the kernel version of analog I/O module

8.2 Initialization and System Information

@ Name

`HSL_initial` – Master board initialization

`HSL_initial_sw` – Initialize by the system automatically (sw_enable=0) or manually via the S1 dip switch (sw_enable=1) (7853 only)

`HSL_close` – Release all resource occupied by master board

`HSL_start` – Start to scan all slave module connected to master board

`HSL_auto_start` – Start to scan and automatically detect all the slave modules connected to master card

`HSL_stop` –Stop scanning the connected slave modules

`HSL_set_scan_condition` – Set scanning conditions (7853 only)

`HSL_get_scan_condition` – Get scanning conditions (7853 only)

`HSL_connect_status` – Get the communication status of the specified slave module

`HSL_slave_live` – Get the module status of the slave module

`HSL_get_irq_channel` – Get the IRQ occupied by the master card

@ Description

`HSL_initial_sw`:

Like HSL_initial, it can initialize the hardware and software states of the HSL master card. This function returns the initialized card bit. Users can use get this API to initialize all HSL master cards at one time, and check their card IDs. It also supports automatically (sw_enable = 0) or manually (sw_enable=1) arranged card IDs via the S1 dip switch.

`HSL_initial`:

Initializes the hardware and software states of the HSL master card. You can check the return code of this function to know if the initialization is successful or not. Since the HSL master card is plug-and-play, the base address and IRQ level are automatically assigned by the BIOS.

`HSL_close`:

Releases the resource occupied by the HSL master card. When terminating the program, do not forget to call this function to release all the resource occupied by the HSL master card.

`HSL_start`:

Scans the total connected slave modules. You can assign the number of slave indexes the HSL master board will scan.

`HSL_auto_start`:

Automatically detects the total connected slave modules. Every master controller can connect up to 63 slave indexes.

`HSL_stop`:

Stops scanning the connected slave modules.

`HSL_set_scan_condition`:

Assigns the scan rate (3/6/12 Mbps) and communication types (full or half duplex). This function needs to be set up between the function HSL_initial and HSL_start.

`HSL_get_scan_condition`:

By this function, User can get the settings of communication types and scan rate which are set by “HSL_set_scan_condition”.

`HSL_connect_status`:

This function is used to check the communication status between master board and slave modules.

`HSL_slave_live`:

This function is used to check the status of the slave module (alive or dead).

`HSL_` get_irq_channel:

This function is used to get IRQ assigned by the system.

@ Syntax

C/C++

```
I16 HSL_initial (U16 card_ID);
I16 FNTYPE HSL_initial_sw(I32 *card_ID_inBit, I16
     sw_enable );
I16 HSL_close (U16 card_ID);
I16 HSL_start (U16 card_ID, U16 connect_index,
     U16 max_slave_No);
I16 HSL_auto_start (U16 card_ID, U16
     connect_index);
I16 HSL_stop (U16 card_ID, U16 connect_index);
I16 HSL_set_scan_condition(I16 card_ID, I16
     connect_index, I16 comm_type, I16
     transfer_rate, I16 hub_number);
I16 HSL_get_scan_condition(I16 card_ID, I16
     connect_index, I16 *comm_type, I16
     *transfer_rate, I16 *hub_number);
I16 HSL_connect_status (U16 card_ID, U16
     connect_index, U16 slave_No, U8 *sts_data);
I16 HSL_slave_live (U16 card_ID, U16
     connect_index, U16 slave_No, U8 *live_data);
void HSL_get_irq_channel (I16 card_ID, I16
     *irq_no);
```

Visual Basic

```
HSL_initial (ByVal card_ID As Integer) As Integer
HSL_initial_sw (ByRef card_ID_inBit As Integer,
     ByVal sw_enable As Integer) As Integer
HSL_close (ByVal card_ID As Integer) As Integer
HSL_start (ByVal card_ID As Integer, ByVal
     connect_index As Integer, ByVal max_slave_No
     As Integer) As Integer
HSL_auto_start (ByVal card_ID As Integer, ByVal
     connect_index As Integer) As Integer
HSL_stop (ByVal card_ID As Integer, ByVal
     connect_index As Integer) As Integer
HSL_set_scan_condition(ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     comm_type As Integer, ByVal transfer_rate As
     Integer, ByVal hub_number As Integer);
```

```
HSL_get_scan_condition((ByVal card_ID As Integer,
     ByVal connect_index As Integer, comm_type As
     Integer, transfer_rate As Integer,
     hub_number As Integer);
HSL_connect_status (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, sts_data as Byte) As
     Integer
HSL_slave_live (ByVal card_ID As Integer, ByVal
     connect_index As Integer, ByVal slave_No as
     Integer, live_data as Byte) As Integer
HSL_get_irq_channel (ByVal card_ID As Integer,
     irq_no As Integer) As Integer
```

@ Argument

`card_ID`: Specify the HSL master card index. Normally, the board index sequence would be decided by the system. The index is from 0.

`*card_ID_inBit`: Card ID information in bit format. Example: If the value of BoardID_InBits is 0x11, there are 2 cards in your system and the card's IDs are 0 and 4.

`sw_enable`: Card ID is initialized by the system automatically (sw_enable=0) or manually via the S1 dip switch (sw_enable=1).

`max_slave_No`: The maximum slave index connected to the HSL master card with the connect_index. The valid value is from 1 to 63.

`slave_No`: Specifiy the slave module with slave index which want to perform this function. The valid value is from 1 to 63.

`comm_type`: Half or Full duplex

0: Half duplex

1: Full duplex

`transfer_rate`: transfer rate setting

1: 3M

2: 6M

3: 12M

hub_number: cascaded Hub number. If no Hub in the system, the value of hub_number is set to 0.

***sts_data**: The communication status of this slave module. The definition is as follows.

- Bit 0 is Data_Req bit.
- Bit 2 is for CHK1. (If Bit2 is 1. It means that there is 1 time communication error).
- Bit 3 is for CHK3. (If Bit3 is 1. It means that there are 3 times communication errors).
- Bit 4, BIT 5 and BIT 6 bits are for CHK7. (If Bit4, Bit5 and Bit6 all are 1. It means that there are 7 times communication errors).

***live_data**: The module status.

- 1: the module is live
- 0: the module is die.

irq_no: IRQ occupied by master card.

@ Return Code

```
ERR_No_Error
ERR_Open_Driver_Fail
ERR_Invalid_Board_Number
ERR_Satellite_Number
ERR_Connect_Index
```

8.3 Error Codes

The following table provides a list of possible return values in the HSL master library. If the return value is non-zero, it means there an error or warning has occurred. The C/C++ standard header file, HSL_ErrorCode.h, defines the error codes.

Code	Description
-1	ERR_No_Device_Found
0	ERR_No_Error
1	ERR_Board_No_Init
2	ERR_Invalid_Board_Number
3	ERR_PCI_Bios_Not_Exist
4	ERR_Open_Driver_Fail
5	ERR_Memory_Mapping
6	ERR_Connect_Index
7	ERR_Satellite_Number
8	ERR_Count_Number
9	ERR_Satellite_Type
10	ERR_Not_ADLink_Slave_Type
11	ERR_Channel_Number
12	ERR_Over_Max_Address
13	ERR_AI_Range
14	ERR_AI_Signal_Type
15	ERR_AI_CJC_Status
16	ERR_CJC_Direction
17	ERR_Time_Out
18	ERR_Create_Timer
19	ERR_PID_Create_Failed
20	ERR_PID_Start_Failed
21	ERR_PID_No_Output
22	ERR_PID_No_FeedBack
23	ERR_No_PID_Controller
24	ERR_Logic_Input
25	ERR_OS_Unknown
26	ERR_AI16AO2_Signal_Range
27	ERR_AI16AO2_Read
28	ERR_AI16AO2_Last_Channel
29	ERR_AI16AO2_Set_Data
30	ERR_AI16AO2_Read_Signal_Type

Code	Description
31	ERR_AO_Channel_Input
32	ERR_AI_Channel_Input
33	ERR_DA_Channel_Input
34	ERR_Over_Voltage_Spec
35	ERR_File_Open_Fail
36	ERR_TrimDAC_Channel
37	ERR_Over_Current_Spec
38	ERR_Axis_Out_Of_Range
39	ERR_Send_Motion_Command
40	ERR_Read_Motion_HexFile
41	ERR_Flash_Data_Transfer
42	ERR_Unkown_Data_Type
43	ERR_CheckSum
44	ERR_Point_Index
45	ERR_DI_Channel_Input
46	ERR_DO_Channel_Output
47	ERR_No_GCode
48	ERR_Code_Syntax
49	ERR_Read_GC_TexTFile
50	ERR_No_Motion_Module
51	ERR_Owner_Set
52	ERR_Signal_Notify
53	ERR_Communication_Type_Range
54	ERR_Transfer_Rate
55	ERR_Hub_Number
56	ERR_Slave_Number
57	ERR_Slave_Not_Stop
58	ERR_Link_Status
59	ERR_Counter_Failed
60	ERR_Create_Event_Failed
61	ERR_DI_Renewal_Type
62	ERR_Wait_Di_Interrupt
63	ERR_Di_Event_Open_Already
64	ERR_Di_Event_Disable
65	ERR_Timer_Parameter
66	ERR_Close_Timer
67	ERR_Wait_Timer_Interrupt
68	ERR_AO_Data

Code	Description
69	ERR_Flash_Write_In
70	ERR_Motion_Busy
71	ERR_Motion_abnormal_stop
72	ERR_Di_Latch_time
73	ERR_Set_Di_Latch_Failed
74	ERR_Parameters_invalid
75	ERR_LinkIntError
76	ERR_HomeALL_Mode
77	ERR_RW_Procedure_Error
78	ERR_Handshake_Method
79	ERR_Kernel_Type_Dismatch
80	ERR_No_8ID_KernelType
81	ERR_DI_Renewal_Type_Interruptmode
82	ERR_Invalid_Setup
83	ERR_StrVelError
84	ERR_Read_ModuleType_Dismatch
85	ERR_Gantry_Axis_Counts
86	ERR_Gantry_Axis_Dismatch
87	ERR_Gantry_not_enable
88	ERR_Gantry_MotionType
89	ERR_Board_Already_Init
90	ERR_4XMO_Not_Support
91	ERR_RTX_Not_Support
92	ERR_InvalidCommand
93	ERR_Win32Error
94	ERR_Dimension_Wrong

8.4 Timer Control

@ Name

`HSL_set_int_timer` (7853 only) – Set the timer parameters

`HSL_set_int_timer_enable` (7853 only) – Enable\Disable timer interrupt of master card (7853 only)

`HSL_wait_timer_interrupt` (7853 only) – Wait timer event

@ Description

`HSL_set_int_timer` (7853 only):

Sets up the Timer parameter p1. The timer is used as frequency divider to generate a dedicated constant timer interrupt sampling rate.

The formula is: Frequency(Hz) = $\dfrac{48MHz}{256 \cdot (p1+1)}$

`HSL_set_int_timer_enable` (7853 only):

Enables or disables the hardware timer interrupt of this master card.

`HSL_wait_timer_interrupt` (7853 only):

Waits for the specific interrupt when you enabled the interrupt function by HSL_set_int_timer_enable() and set the timer parameter p1 by HSL_set_int_timer(). When this function is running, the process never stops even if it is triggered or the function has timed out.

The following code illustrates the HSL_wait_timer_interrupt function.

```
I16 ret;
HSL_set_int_timer(0, 0xffff); // set the
     parameter p1
HSL_set_int_timer_enable(0, 1); //enable the
     timer

for(int i = 0; i< 10; i++)
{
     ret = HSL_wait_timer_interrupt(g_cardId,
     10000);
     if(ret == 0)
          // do something…
     else
     // time out
}
```

@ Syntax

C/C++ (DOS, Windows 98/NT/2000/XP)

```
I16 HSL_set_timer (I16 card_ID, I16 c1, I16 c2);
I16 HSL_enable_timer_interrupt (I16 card_ID,
     HANDLE *phEvent);
I16 HSL_disable_timer_interrupt (I16 card_ID);
I16 HSL_set_int_timer(I16 card_ID, U16 p1);
I16 HSL_set_int_timer_enable(I16 card_ID, I16
     enable);
I16 HSL_wait_timer_interrupt(I16 card_ID, I32
     time_out_ms);
```

Visual Basic (Windows 98/NT/2000/XP)

```
HSL_set_timer (ByVal card_ID As Integer, ByVal c1
     As Integer, ByVal c2 As Integer) As Integer
HSL_enable_timer_interrupt (ByVal card_ID As
     Integer, phEvent As Long) As Integer
HSL_disable_timer_interrupt (ByVal card_ID As
     Integer) As Integer
HSL_set_int_timer(ByVal card_ID As Integer, ByVal
     p1 As Integer)As Integer
HSL_set_int_timer_enable(ByVal card_ID As
     Integer, ByVal enable As Integer) As Integer
```

```
HSL_wait_timer_interrupt(ByVal card_ID As
     Integer, ByVal time_out_ms As Integer)As
     Integer
```

@ Argument

`card_ID`: Specifies the HSL master card index. Typically, the board index sequence is assigned by the system. The index starts from 0.

`*phEvent`: Returns the handle of the timer interrupt event. The interrupt event indicates an interrupt which is generated from the master card's timer.

`c1`: Frequency divider of Timer 1.

`c2`: Frequency divider of Timer 2.

`p1`: Parameter of timer

The formula is : Frequency(Hz) = $\frac{48MHz}{256 \cdot (p1+1)}$

`enable`: Enables (1) or disables (0) the timer interrupt

`time_out_ms`: Specifies the time-out interval in milliseconds. The function returns if the interval elapses, even if the interrupt is non-signaled. If time_out_ms is zero, the function tests the Di state and returns immediately. If time_out_ms is -1, the function time-out interval does not elapse (infinite).

@ Return Code

```
ERR_No_Error
ERR_Invalid_Board_Number
ERR_Timer_Parameter
ERR_Close_Timer
ERR_Wait_Timer_Interrupt
```

8.5 Discrete I/O

@ Name

`HSL_D_read_input` – Read back all discrete I/O with unsigned 32-bit

`HSL_D_read_channel_input` – Read back discrete I/O by channel selection

`HSL_D_write_output` – Write all discrete I/O with unsigned 32-bit HSL_D_write_channel_output – Write discrete I/O by channel selection

`HSL_D_read_ouput` – Read back the output value stored in RAM

`HSL_D_read_all_slave_input` – Read back all inputs of slave modules

`HSL_D_write_all_slave_output` – Write all outputs of slave modules

`HSL_D_set_input_logic` – Set the logic of digital input

`HSL_D_set_output_logic` – Set the logic of digital output

`HSL_D_set_int_renewal_type` (7853 only) – Set DI renewal check type

`HSL_D_set_int_renewal_bit` (7853 only) – Set the data bits of DI renewal check for each DI slave module

`HSL_D_set_int_control` (7853 only) – Set DI interrupt enable or disable

`HSL_D_wait_di_interrupt` (7853 only) – Wait DI renewal event

@ Description

`HSL_D_read_input`:

Reads the digital input value of the discrete I/O module. You must specify the connect index and slave index.

`HSL_D_read_channel_input`:

Reads the digital input value of the discrete I/O module at a specified channel.

`HSL_D_write_output`:

Writes the digital output value of the discrete I/O module. You must specify the connect index and slave index.

`HSL_D_write_channel_output`:

Writes the digital output value of the discrete I/O module at the specified channel.

`HSL_D_read_ouput`:

Writes all digital output values to all connected discrete I/O modules. This function maps all data into memory. With this function, you can write all digital output values to all connected discrete I/O modules at one time.

`HSL_D_read_all_slave_input`:

Reads the digital input values from all slave I/O modules with set value of connect_index and card no is card_ID. This function allows you to read all digital input values from all slave I/O modules at one time.

`HSL_D_write_all_slave_output`:

Writes the digital output values from all slave I/O modules with set value of connect_index and card no is card_ID. This function allows you to write all digital output values from all slave I/O modules at one time.

`HSL_D_set_input_logic`:

Sets the digital input logic to the specified slave I/O module. The slave I/O module‘s address is slave_No and set value is connect_index.

`HSL_D_set_output_logic`:

Sets the digital output logic to the specified slave I/O module. The slave I/O module's address is slave_No and set value is connect_index.

`HSL_D_set_int_renewal_type` (7853 only):

Sets the type of hardware interrupt occurrence timing. These are.

Type 1: Generates hardware interrupt when any DI data transitions are detected. (Figure 5.1)

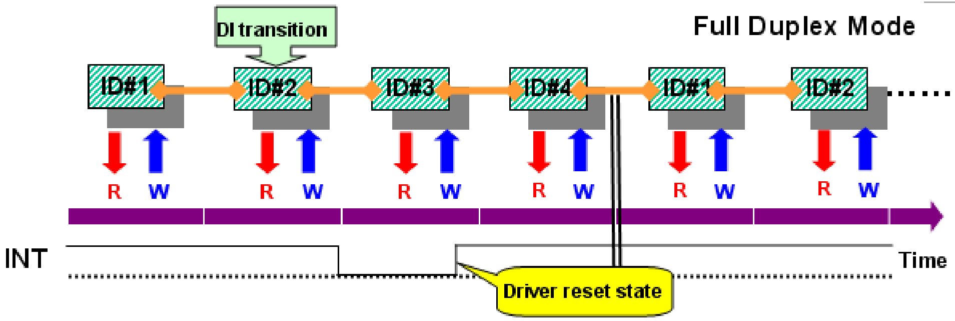


Figure 8-1: Type 1

Type 2: Generates hardware interrupt when any DI data transitions are detected and when the scan cycle is completed.

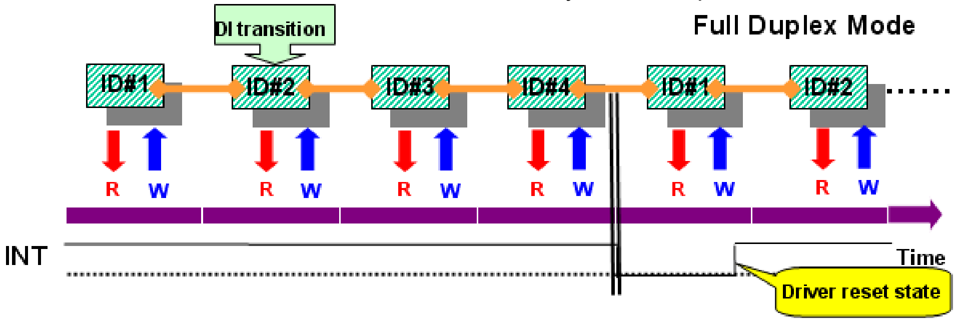


Figure 8-2: Type 2

Type 3: Generates hardware interrupt when any DI data transitions are detected and when the scan cycle is completed. When interrupt occurrs, the scan pauses until the driver resets the state.

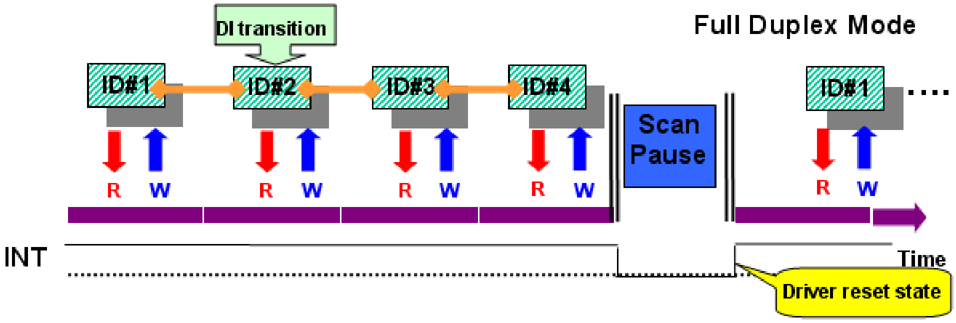


Figure 8-3: Type 3

Caution: Scanning is paused while user choice the type3 of renewal type. This pause time depends on the user system performance. Consequently, when using type3, constancy(always keeping scan cycle constant) will not be maintained between scans.

`HSL_D_set_int_renewal_bit` (7853 only):

Sets the Di data bits of specified modules that you want to monitor.

`HSL_D_set_int_control` (7853 only):

Enables or disables the DI interrupt.

`HSL_D_wait_di_interrupt` (7853 only):

Waits for the specific interrupt when you enable the Interrupt function by HSL_D_set_int_control() and set the renewal type and data bits on specified slave DI modules by HSL_D_set_int_renewal_bit(), HSL_D_set_int_renewal_type(). When this function is running, the process never stops even if triggered or the function timed out.

The following code illustrates the HSL_D_wait_di_interrupt function.

```
I16 ret;
HSL_D_set_int_renewal_type(1, 0, 1);
 // slave id = 1, monitor the states of bit 0 and
     bit 1
HSL_D_set_int_renewal_bit(1, 0, 1, 0x003);
HSL_D_set_int_control(1, 0, 1); //enable
…
// start wait
ret =HSL_D_wait_di_interrupt(1, 10000);
if(ret == ERR_No_Error)
{    // DI state trainisted and check which bits
     change states…
}else
{    // time out
}…
```

@ Syntax

C/C++

```
I16 HSL_D_write_output (I16 card_ID, I16
     connect_index, I16 slave_No, U32 out_data);
I16 HSL_D_write_channel_output(I16 card_ID, I16
     connect_index, I16 slave_No, I16 channel,
     U16 out_data);
I16 HSL_D_read_input (I16 card_ID, I16
     connect_index, I16 slave_No, U32 *in_data);
I16 HSL_D_read_channel_input (I16 card_ID, I16
     connect_index, I16 slave_No, I16 channel,
     U16 *in_data);
I16 HSL_D_read_output (I16 card_ID, I16
     connect_index, I16 slave_No, U32
     *out_data_in_ram);
I16 HSL_D_read_all_slave_input (I16 card_ID, I16
     connect_index, U16 *in_data);
I16 HSL_D_write_all_slave_output (I16 card_ID,
     I16 connect_index, U16 *out_data);
I16 HSL_D_set_input_logic (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     input_logic);
```

```
I16 HSL_D_set_output_logic (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     output_logic);
I16 HSL_D_set_int_renewal_type(I16 card_ID, I16
     connect_index, I16 type);
I16 HSL_D_set_int_renewal_bit(I16 card_ID, I16
     connect_index, I16 slave_No, U16
     bitsOfCheck);
I16 HSL_D_set_int_control(I16 card_ID, I16
     connect_index, I16 enable);
I16 HSL_D_wait_di_interrupt(I16 card_ID, I32
     time_out_ms);
```

Visual Basic

```
HSL_D_write_output (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ByVal out_data As Long)
     As Integer
HSL_D_write_channel_output (ByVal card_ID As
     Integer, ByVal connect_index As Integer,
     ByVal slave_No As Integer, ByVal channel As
     Integer, ByVal out_data As Integer) As
     Integer
HSL_D_read_input (ByVal card_ID As Integer, ByVal
     connect_index As Integer, ByVal slave_No As
     Integer, in_data As Long) As Integer
HSL_D_read_channel_input (ByVal card_ID As
     Integer, ByVal connect_index As Integer,
     ByVal slave_No As Integer, ByVal channel As
     Integer, in_data As Integer) As Integer
HSL_D_read_output (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, out_data_in_ram As
     Long) As Integer
HSL_D_read_all_slave_input (ByVal card_ID As
     Integer, ByVal connect_index As Integer,
     in_data As Integer) As Integer
HSL_D_write_all_slave_output (ByVal card_ID As
     Integer, ByVal connect_index As Integer,
     out_data As Integer) As Integer
HSL_D_set_input_logic (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
```

```
          slave_No As Integer, ByVal input_logic As
          Integer) As Integer
     HSL_D_set_output_logic (ByVal card_ID As Integer,
          ByVal connect_index As Integer, ByVal
          slave_No As Integer, ByVal output_logic As
          Integer) As Integer
     HSL_D_set_int_renewal_type(ByVal card_ID As
          Integer, ByVal connect_index As Integer,
          ByVal type As Integer)As Integer
     HSL_D_set_int_renewal_bit(ByVal card_ID As
          Integer, ByVal connect_index As Integer,
          ByVal slave_No As Integer, ByVal bitsOfCheck
          As Long) As Integer
     I16 HSL_D_set_int_control(I16 card_ID, I16
          connect_index, I16 enable);
     I16 HSL_D_wait_di_interrupt(I16 card_ID, I32
          time_out_ms);
```

@ Argument

`card_ID`: Specifies the HSL master card index. Typically, the board index sequence is assigned by the system. The index starts from 0.

`slave_No`: Specifies the slave module with slave index that wants to perform this function. The valid value is 1 to 63.

`out_data`: The digital output of the discrete module

- **`HSL_D_write_output`**: The data of channel 0 is assigned to bit 0, the data of channel 1 is assigned to bit 1, and so on.
- **`HSL_D_write_channel_output`**: The value is the digital output data of the specified channel.

`*out_data`: An unsigned short array pointer. You must create an unsigned short array containing 63 cells. The cell index corresponds to the slave index. For example, cell index 0 corresponds to the module with slave index 1. The cell index 2 corresponds to the module with slave index 2, and so on. The last cell index 62 corresponds to the module with slave index 63.

Cell index of array (Unsigned short)	Corresponding slave index
0	1
1	2
……	……
62	63

***in_data**: The input data of slave modules.

- For **HSL_D_read_input**: The data of channel 0 is assigned to bit 0, the data of channel 1 is assigned to bit 1, and so on.
- For **HSL_D_read_channel_input**: The value is the digital input data of the specified channel.
- oFor **HSL_D_all_slave_index**: An unsigned short array pointer. You must create an unsigned short array containing 63 cells. The cell index corresponds to the slave index. For example, cell index 0 corresponds to the module with slave index 1. The cell index 2 corresponds to the module with slave index 2, and so on. The last cell index 62 corresponds to the module with slave index 63.

Cell index of array (Unsigned short)	Corresponding slave index
0	1
1	2
……	……
62	63

channel: Specifies the channel of the discrete I/O module that wants to perform this function. The valid values are enumerated below.

- HSL-DI16DO16: 0 to 15
- HSL-DI32: 0 to 31
- HSL-DO32: 0 to 31

`*out_data_in_ram`: The output data stored in RAM. The data of channel 0 is assigned to bit 0; the data of channel 1 is assigned to bit 1 and so on.

`input_logic`: Sets the input logic to the specified module.

output_logic: Sets the output logic to the specified module.

`Type`: Types of hardware interrupt occurrence timing value (1 to 3).

`bitsOfCheck`: Renews data bits (16 bits).

`enable`: Enables (0) or disables (1) the Di interrupt.

`time_out_ms`: Specifies the time-out interval in milliseconds. The function returns if the interval elapses, even when the interrupt is non-signaled. If time_out_ms is zero, the function tests the Di state and returns immediately. If time_out_ms is -1, the function's time-out interval does not elapses (infinite).

@ Return Code

```
ERR_No_Error
ERR_Invalid_Board_Number
ERR_Memory_Mapping
ERR_Connect_Index
ERR_Satellite_Number
ERR_Over_Max_Address
ERR_DI_Renewal_Type
ERR_Wait_Di_Interrupt
ERR_Di_Event_Open_Already
ERR_Di_Event_Disable
```

8.6 Analog I/O

@ Name

`HSL_A_start_read` – Start A/D conversion

`HSL_A_stop_read` – Stop A/D conversion

`HSL_A_set_signal_range` – Set the signal range of analog input channels HSL_A_get_signal_range – Get the signal range of analog input channels

`HSL_A_get_input_mode` – Get the signal input mode

`HSL_A_set_last_channel` – Set the last channel of analog input channels

`HSL_A_get_last_channel` – Get the last channel of analog input channels

`HSL_A_read_input` – Read back the value of analog input channels

`HSL_A_write_output` – Send out the analog output

`HSL_A_read_output` – Read back the analog output data

`HSL_A_sync_rw` – Read and write the data synchronously

`HSL_A_get_version` – Get the kernel version of analog I/O module

@ Description

`HSL_A_start_read`:

Initializes the reading operation of the analog input channels of all HSL AI/O modules that are connected to the master card. Before using HSL_A_read_input(), HSL_A_write_output() and HSL_A_sync_rw(), the functions must be executed to start the A/D conversion.

`HSL_A_stop_read`:

Stops the reading operation of analog input channels of all HSL AI/O modules that are connected to the master card. Use this function to stop the A/D conversion.

`HSL_A_set_signal_range`:

Sets the input range of the specified HSL AI/O modules.

`HSL_A_get_signal_range`:

Obtains the input range of the specified HSL AI/O modules.

`HSL_A_get_input_mode`:

Obtains the signal input mode of HSL AI/O modules. This is determined by hardware jumper setting.

`HSL_A_set_last_channel`:

Sets the last number of analog input channels of HSL AI/O modules. For example, the HSL-AI16AO2 has 16 analog inputs with single-ended wiring. If you want to read back the first four analog input data, assign the last channel as 3. The analog input channel index starts from 0. The AI channel 0 to 4 are enabled while the rest are disabled.

`HSL_A_get_last_channel`:

Retrieves the last number of analog input channels of HSL AI/O modules. For example, if you use HSL_A_set_last_channel and set the last channel as 5, then you can read the value of the last channel using this function.

`HSL_A_read_input`:

Reads the specified AI channel of the slave module.

`HSL_A_write_output`:

Writes the specified AO channel of the slave module.

`HSL_A_read_output`:

Reads back the analog output data from the HSL AI/O modules with the specified analog output channel.

`HSL_A_sync_rw`:

Synchronously reads AI data and writes AO data at the specified channel of the HSL AIO module. It allows simultaneous data read/write.

`HSL_A_get_version`:

Reads the kernel version of the HSL AI/O modules.

@ Syntax

C/C++

```
I16 HSL_A_start_read (I16 card_ID, I16
     connect_index);
I16 HSL_A_stop_read (I16 card_ID, I16
     connect_index);
I16 HSL_A_set_signal_range (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     signal_range);
I16 HSL_A_get_signal_range (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     *signal_range);
I16 HSL_A_get_input_mode (I16 card_ID, I16
     connect_index, I16 slave_No, I16 *mode);
I16 HSL_A_set_last_channel (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     last_channel);
I16 HSL_A_get_last_channel (I16 card_ID, I16
     connect_index, I16 slave_No, I16
     *last_channel);
I16 HSL_A_read_input (I16 card_ID, I16
     connect_index, I16 slave_No, I16 ai_channel,
     F64 *ai_data);
I16 HSL_A_write_output (I16 card_ID, I16
     connect_index, I16 slave_No, I16 ao_channel,
     F64 ao_data);
I16 HSL_A_read_output (I16 card_ID, I16
     connect_index, I16 slave_No, I16 ao_channel,
     F64 *ao_data);
I16 HSL_A_sync_rw (I16 card_ID, I16
     connect_index, I16 slave_No, I16 ai_channel,
     F64 *ai_data, I16 ao_channel, F64 ao_data);
I16 HSL_A_get_version (I16 card_ID, I16
     connect_index, I16 slave_No, I16 *ver);
```

Visual Basic

```
HSL_A_start_read (ByVal card_ID As Integer, ByVal
     connect_index As Integer) As Integer
HSL_A_stop_read (ByVal card_ID As Integer, ByVal
     connect_index As Integer) As Integer
```

```
HSL_A_set_signal_range (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ByVal signal_range As
     Integer) As Integer
HSL_A_get_signal_range (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, signal_range As
     Integer) As Integer
HSL_A_get_input_mode (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, mode As Integer) As
     Integer
HSL_A_set_last_channel (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ByVal last_channel As
     Integer) As Integer
HSL_A_get_last_channel (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, last_channel As
     Integer) As Integer
HSL_A_read_input (ByVal card_ID As Integer, ByVal
     connect_index As Integer, ByVal slave_No As
     Integer, ByVal ai_channel As Integer,
     ai_data As Double) As Integer
HSL_A_write_output (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ByVal ao_channel As
     Integer, ByVal ao_data As Double) As Integer
HSL_A_read_output (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ByVal ao_channel As
     Integer, ao_data As Double) As Integer
HSL_A_sync_rw (ByVal card_ID As Integer, ByVal
     connect_index As Integer, ByVal slave_No As
     Integer, ByVal ai_channel As Integer,
     ai_data As Double, ByVal ao_channel As
     Integer, ByVal ao_data As Double) As Integer
HSL_A_get_version (ByVal card_ID As Integer,
     ByVal connect_index As Integer, ByVal
     slave_No As Integer, ver As Integer) As
     Integer
```

@ Argument

card_ID: Specifies the HSL master card index. Typically, the system assigns the board index sequence. The index starts from 0.

slave_No: Specifies the slave module with slave index that wants to perform this function. The valid value is 1 to 63.

signal_range: The single range of analog input setting.

For HSL-AI16AO2-M-VV

0: ± 1.25 V

1: ± 2.5 V

2: ± 5 V

3: ± 10 V

***signal_range**: Reads back the single range of analog input setting.

For HSL-AI16AO2-M-VV

0: ± 1.25 V

1: ± 2.5 V

2: ± 5 V

3: ± 10 V

***mode**: 0: differential type; 1: single-ended input.

last_channel: For single-ended setting, the maximum last channel is 15. For differential setting, the maximum last channel is 7.

***last_channel**: You can get the last channel depending on what you set previously. For single-ended setting, the maximum last channel is 15. For differential setting, the maximum last channel is 7.

ai_channel: Specifies the AI channel of the slave module that wants to perform this function. The valid value is described as follows.

HSL-AI16AO2-M-VV

Differential: 0 - 15

Single-ended: 0 - 7

`ao_channel`: Specifies the AI channel of the slave module that wants to perform this function. For HSL-AI16AO2-M-VV/AV, the valid value is 0 and 1.

`*ai_data`: The AI data of the specified channel. The unit is Volt for HSL-AI16AO2-M-VV module.

`ao_data`: The AO data of the specified channel in Volt.

`*ver`: kernel version number.

@ Return Code

```
ERR_No_Error
ERR_Invalid_Board_Number
ERR_Connect_Index
ERR_Time_Out
ERR_Memory_Mapping
ERR_Satellite_Number
ERR_Satellite_Type
ERR_Over_Max_Address
ERR_AI16AO2_Signal_Range
```

This page intentionally left blank.

9 How to Program with the HSL Function Library

This chapter describes how to create a program with the HSL C library using a flowchart. The C library supports Windows and Redhat Linux platforms.

9.1 Programming with HSL DLL

The programming flow chart illustrates the program creation with HSL DLL.

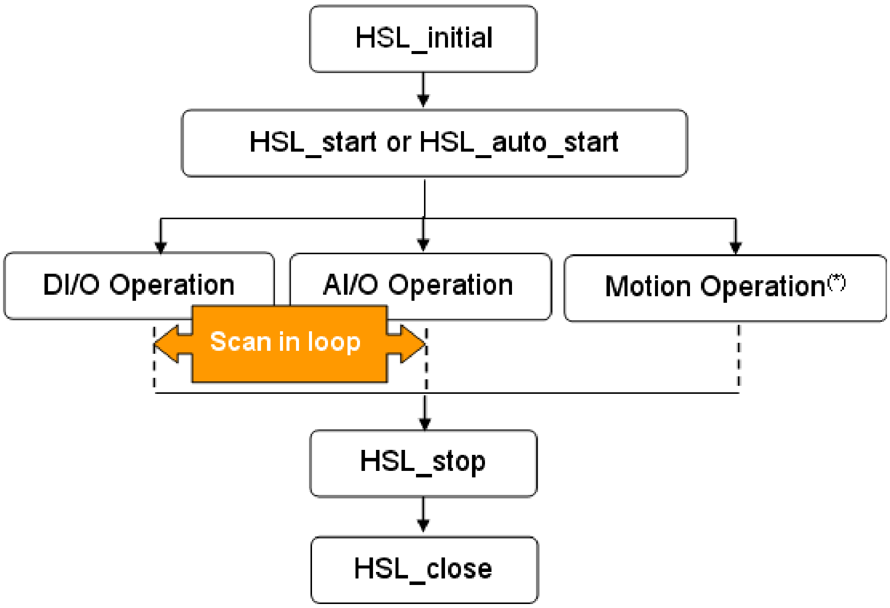


Figure 9-1: Programming Flow

9.1.1 DIO Operation

Inside DI/O Operation, the following function calls are for users' reference.

`HSL_slave_live` (…):

Detects the status of the slave module (live or die).

HSL_connect_status(…):

Detect the communication status of the slave module.

HSL_D_read_input(…)

HSL_D_read_channel_input(….)

HSL_D_read_all_slave_input(….)

Functions for the digital input operation of slave modules.

HSL_D_write_output(…)

HSL_D_write_channel_output(…)

HSL_D_write_output(…)

Functions for the digital output operation of slave modules.

HSL_D_read_output(…)

Reads the output data in memory.

HSL_D_set_input_logic(…)

HSL_D_set_output_logic(…)

Functions for setting the DIO logic.

All functions may be executed in a loop to obtain the latest information from the slave modules.

9.1.2 AI/O Operation

Inside AI/O Operation, the following function calls are provided for user reference.

1. If the module needs to be calibrated, refer to Appendix C.
2. To set the AI/O configuration of the slave module, use

HSL_A_set_signal_range(…)

HSL_A_set_last_channel(..).

If you want to check AI/O configuration, use

HSL_A_get_signal_range(…)

HSL_A_get_input_mode(…)

HSL_A_get_last_channel(…)

3. Use **HSL_A_start_read**(…) to initialize the AIO channels reading operation.
4. After activating the HSL AD conversion, use these functions for the HSL operation.

HSL_slave_live(…)

Detects the status of the slave module(Live or Die).

HSL_connect_status(…)

Detects the communication status of the slave module.

HSL_A_read_input(…)

Function for analog value reading operation of the slave modules.

HSL_A_write_output(…)

Function for analog value writing operation of the slave modules.

HSL_A_sync_rw(…)

Function for synchronous analog input and output.

5. Use **HSL_A_stop_read**(….) to stop the AIO channels reading operation.

All steps may be executed in a loop to get the latest information from the slave modules.

9.1.3 Motion Operation:

Refer to HSL-4XMO user's manual.

This page intentionally left blank.

Important Safety Instructions

For user safety, please read and follow all instructions, Warnings, Cautions, and Notes marked in this manual and on the associated device before handling/operating the device, to avoid injury or damage.

S'il vous plaît prêter attention stricte à tous les avertissements et mises en garde figurant sur l'appareil , pour éviter des blessures ou des dommages.

- Read these safety instructions carefully.
- Keep the User’s Manual for future reference.
- Read the Specifications section of this manual for detailed information on the recommended operating environment.
- The device can be operated at an ambient temperature of 55ºC.
- When installing/mounting or uninstalling/removing device, or when removal of a chassis cover is required for user servicing:
  - Turn off power and unplug any power cords/cables.
  - Reinstall all chassis covers before restoring power.
- To avoid electrical shock and/or damage to device:
  - Keep device away from water or liquid sources.
  - Keep device away from high heat or humidity.
  - Keep device properly ventilated (do not block or cover ventilation openings).
  - Always use recommended voltage and power source settings.
  - Always install and operate device near an easily accessible electrical outlet.
  - Secure the power cord (do not place any object on/over the power cord).
  - Only install/attach and operate device on stable surfaces and/or recommended mountings.
- If the device will not be used for long periods of time, turn off and unplug it from its power source
- Never attempt to repair the device, which should only be serviced by qualified technical personnel using suitable tools

- A Lithium-type battery may be provided for uninterrupted backup or emergency power.

CAUTION:	Risk of explosion if battery is replaced with one of an incorrect type; please dispose of used batteries appropriately. *Risque d'explosion si la pile est remplacée par une autre de type incorrect. Veuillez jeter les piles usagées de façon appropriée.*

- The device must be serviced by authorized technicians when:
  - The power cord or plug is damaged.
  - Liquid has entered the device interior.
  - The device has been exposed to high humidity and/or moisture.
  - The device is not functioning or does not function according to the User's Manual.
  - The device has been dropped and/or damaged and/or shows obvious signs of breakage.
- Disconnect the power supply cord before loosening the thumbscrews and always fasten the thumbscrews with a screwdriver before starting the system up.
- It is recommended that the device be installed only in a server room or computer room where access is:
  - Restricted to qualified service personnel or users familiar with restrictions applied to the location, reasons therefor, and any precautions required.
  - Only afforded by the use of a tool or lock and key, or other means of security, and controlled by the authority responsible for the location.

	BURN HAZARD Touching this surface could result in bodily injury. To reduce risk, allow the surface to cool before touching. ***RISQUE DE BRÛLURES*** *Ne touchez pas cette surface, cela pourrait entraîner des blessures. Pour éviter tout danger, laissez la surface refroidir avant de la toucher.*

Getting Service

Ask an Expert: http://askanexpert.adlinktech.com

ADLINK Technology, Inc.
9F, No.166 Jian Yi Road, Zhonghe District
New Taipei City 235, Taiwan

Tel:	+886-2-8226-5877
Fax:	+886-2-8226-5717
Email:	service@adlinktech.com

Ampro ADLINK Technology, Inc.
5215 Hellyer Avenue, #110
San Jose, CA 95138, USA

Tel:	+1-408-360-0200
Toll Free:	+1-800-966-5200 (USA only)
Fax:	+1-408-360-0222
Email:	info@adlinktech.com

ADLINK Technology (China) Co., Ltd.
300 Fang Chun Rd., Zhangjiang Hi-Tech Park
Pudong New Area, Shanghai, 201203 China

Tel:	+86-21-5132-8988
Fax:	+86-21-5132-3588
Email:	market@adlinktech.com

ADLINK Technology GmbH
Hans-Thoma-Straße 11
D-68163 Mannheim, Germany

Tel:	+49-621-43214-0
Fax:	+49-621 43214-30
Email:	emea@adlinktech.com

Please visit the Contact page at www.adlinktech.com for information on how to contact the ADLINK regional office nearest you: