High Speed Link System

Master-Slave Distributed Solution

User's Manual

Manual Rev. 2.05

Revision Date: October 15, 2007

Part No: 50-12100-2040

Recycled Paper

Advance Technologies; Automate the World.

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.

Trademarks

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

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

Getting Service from ADLINK

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

ADLINK TECHNOLOGY INC.

Web Site: http://www.adlinktech.com

Sales & Service: Service@adlinktech.com

TEL: +886-2-82265877

FAX: +886-2-82265717

Address: 9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan

Please email or FAX this completed service form for prompt and satisfactory service.

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

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

Table of Contents

List of Tables

List of Figures

How to Use This Manual

This manual helps you in configuring, installing, and using the HSL series products, and describes the functions and the operational theorem of the high-speed link technology. This manual is divided into the following chapters:

Chapter 1 - HSL Introduction: Provides an overview of the HSL system, including the system features, specifications, and communication technology.

Chapter 2 - HSL Master Controller: Presents detailed information on the HSL master.

Chapter 3 - HSL Slave Module: Presents detailed information on the HSL slave modules.

Chapter 4 - HSL LinkMaster Utility: Provides instructions on how to install and use the ADLINK LinkMaster utility for testing and debugging the slave modules.

Chapter 5 - HSL Function Library: Presents the function library usage and syntax.

Chapter 6 - Programming with HSL Function Library: Provides a broad concept and knowledge of how to implement the application with the HSL library.

Appendix A - Scan Time Table: Presents the HSL cycle time based on different transmission speeds and modes.

Appendix B - Mapping Table: Provides a comparison table between old and new functions.

Appendix C - HSL-AI16AO2 Calibration: Outlines the calibration procedures for HSL-AI16AO2-M-VV and HSL-AI16AO2-M-AV.

Appendix D - HSL-HUB/Repeater information: Presents the adding time information and extension limitations.

References

Master board. HSL is a master-slave communication system. In host side, we call the control board as master board.

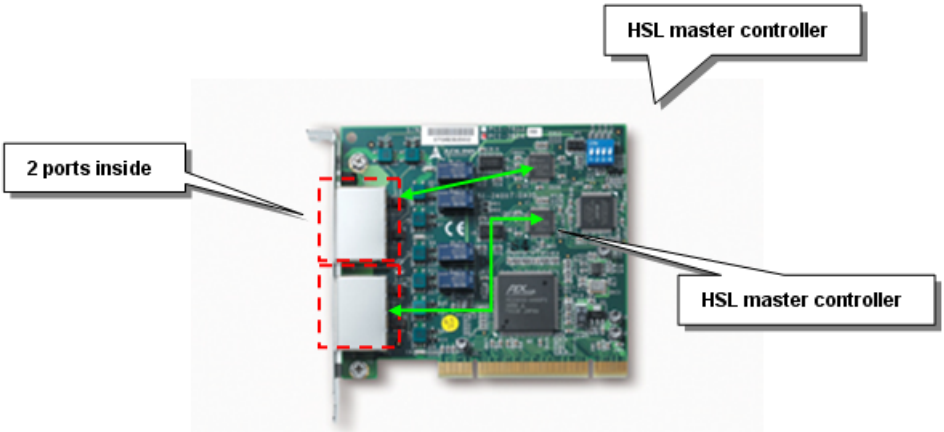
Slave module. HSL is a master-slave communication system. In remote side, the slave module can connect a variety of sensors.

Slave index. The basic unit in HSL system. One HSL slave module may occupy 1, 2 or 4 slave indexes. This depends on the design of slave modules.

Full duplex. Data transmission and receiving at the same scanning time.

Half duplex. Data transmission and receiving at the consecutive scanning time.

HSL master controller. One HSL ASIC plays the role of master controller. For example, PCI-7853 has one on-board HSL ASIC; it can connect a maximum of 63 slave indexes. For convenient connection, the HSL master has two ports. Using the same technology, the PCI-7854 can connect a maximum 126 slave indexes and has four ports.



Transmission speed. The data speed is between master board and slave modules. The unit is bit per second.

1 Introducing HSL

1.1 The HSL System

The HSL is an innovative distributed I/O technology that enables time-deterministic scanning of thousands of I/O points in milliseconds using master-slave architecture. The HSL master board comes in PCI or PMC form factors. The PMC board is used in embedded controllers. By using commercial Ethernet cable with RJ-45 connector, you can easily set up the HSL slave modules as close as possible to the sensor devices, reducing wiring effort. Aside from the I/O modules, ADLINK provides the remote motion control module with 4-axis pulse train type. The HSL network suits a variety of machine-making applications as it integrates discrete I/O, analog I/O, thermocouple module, and motion control. This local network delivers rapid response time, time-deterministic scanning and multiple-axis control. With PMC module, you may also integrate the HSL network with embedded solution platforms.

The HSL system features:

- Distributed solution based on PC architecture or embedded platform
- Convenient wiring for remote distributed I/O modules, including discrete I/Os and analog I/Os
- Space-saving and discrete low-profile U-series form factor
- Hundreds of discrete I/O points
- Time-deterministic, fast scanning
- High-speed data acquisition
- Up to 120 axes of remote motion control with two HSL master controller of master board
- Motion control features point table management and motion script download to enhance execution efficiency

1.1.1 Product Overview

The illustration shows the basic HSL system topology.

Figure 1-1: HSL topology

1.1.2 Product Highlights

High-speed performance

With scanning speed as high as 1000 points per ms, it takes only 1.895 ms for an HSL master to scan all the discrete I/O points of slave modules under 6 Mbps. For example, a distributed control system with 63 slave I/O modules of HSL-DI16DO16-DB-NN with 2016 discrete I/O points can be scanned or updated within 1.895 ms.

Time-deterministic scanning

The HSL master controller implements a deterministic time period when scanning all slave I/O modules. The total scanning cycle time is exactly proportional to the number of slave indexes. At 6 Mbps, every 30.33 µs is added for another slave index. For an HSL system with 30 discrete I/O slave modules (where every discrete I/O module occupies one slave index), the scanning time period is precisely 30 X 30.33 µs = 909.9 µs. The scan time unit based on transmission rate is illustrated below.

	3 Mbps	6 Mbps	12 Mbps
Full Duplex	60.67 µs	30.33 µs	15.17 µs
Half Duplex	118 µs	59 µs	29.5 µs

Convenient wiring

The HSL master controller connects to all slave I/O modules using Ethernet cables. This dramatically reduces the wiring costs and effort. With Ethernet cables, hundreds or even thousands of I/O data can transmitted between the HSL master and slave I/O modules. The HSL wiring is the easiest and most cost-effective solution to date. For low profile series, you can make the connection by direct wiring.

Multiple I/O points

The PCI-7853 offers one HSL master controller while the PCI-7854 offers two HSL master controllers. For maximum installation, users can have eight PCI-7853 and PCI-7854 in one system. That means users can have 1512 slave indexes in HSL network system. If choosing all connected modules as HSL-DI16DO16-DB-NN, a total of 24,192 digital input and 24,192 digital output points are supported. For embedded solution, users can choose the PMC-7852/G.

Easy I/O expansion

Expanding I/O points for centralized configuration requires more I/O boards and available PCI or ISA slots. Problems occur when system needs more I/O points while there are no

available slot. In contrast with centralized configuration, the distributed I/O configuration eliminates this limitation of a centralized I/O configuration. With the HSL system, adding more I/O points only requires one more slave I/O module and an Ethernet cable for communication link.

Self-diagnostic function

The HSL provides a self-diagnostic function that eliminates communication failures. This function continuously monitors the network status while a status register keeps the accumulated slave-no-response count for every individual slave I/O module. Also, the HSL system features the CRC12 to eliminate any communication error.

Modular design of slave I/O

ADLINK offers a variety of slave module types Including the metal-cased M series, non-metal DB series, and U series for compact systems. The M and DB series require a terminal board for connection. Terminal boards act as carrier of slave I/O module with wiring function. The Ethernet port and screw terminal on the terminal boards make it easier to replace I/O modules without turning and wiring off the system.

Remote motion control compatibility

ADLINK also offers remote motion control solution based on the HSL network Including the HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P that could connect up to four axes. The HSL-4XMO-CG-N/P features a general-type interface for use with stepper or linear motors, while the HSL-4XMO-CD-N/P has a D-sub interface. By using a transfer cable, you can connect to specific servo amplifier. You can easily make a distributed control application that includes discrete I/O, analog I/O, and remote motion control.

Easy to program

Every HSL master card comes with 32 KB SRAM that carries all the I/O status information of the HSL system. The ASIC on the HSL master board communicates with all remote slave I/O

modules at fixed scanning period and keeps the most updated I/O status information on the SRAM. You may read and write the data in the 32 KB SRAM on HSL master card through the PCI or PMC bus. You can easily read/write the most updated I/O information and never worry with the HSL protocol.

1.1.3 HSL Applications

HSL as a Distributed PLC

The distributed PLC is an important system in the field of industry automation. Via communication modules, such as RS232, RS485, PLC also performs distributed control. The traditional architecture of distributed PLC application is shown in Figure 1.2. In this setup, the MPC (Monitoring PC) takes over as the medium for data transmission from field to MIS.

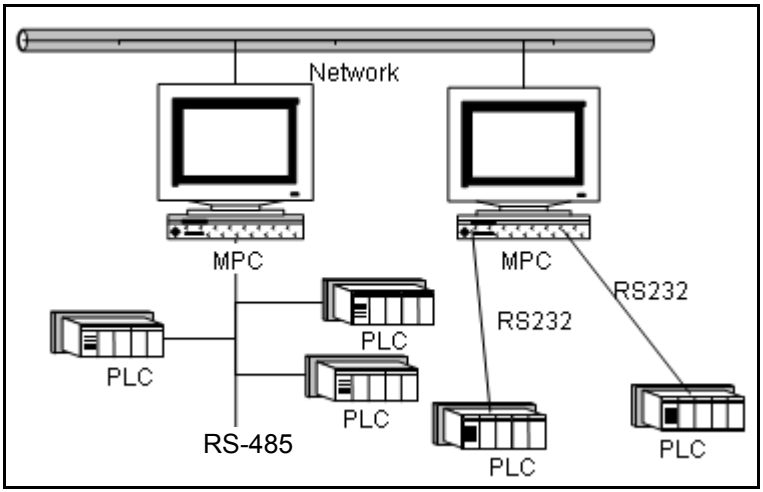


Figure 1-2: Traditional distributed PLC architecture

With the development of communication technology and popularity of networking, networking modules with Ethernet interface became available. This improvement evolved as shown in the architecture below. The medium character of the MPC was replaced.

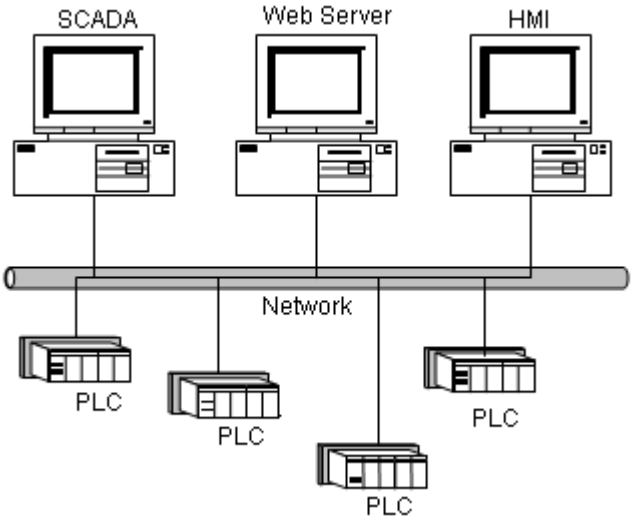


Figure 1-3: Networking PLC

PLCs that are capable of network communications are usually very expansive. And since the PLC is not an open architecture, only hardware vendors are capable of producing it.

The HSL distributed control architecture is illustrated in Figure 1.4. With HSL, there is no need for an extra PC for Ethernet communication. You may use only one IPC to control the entire system.

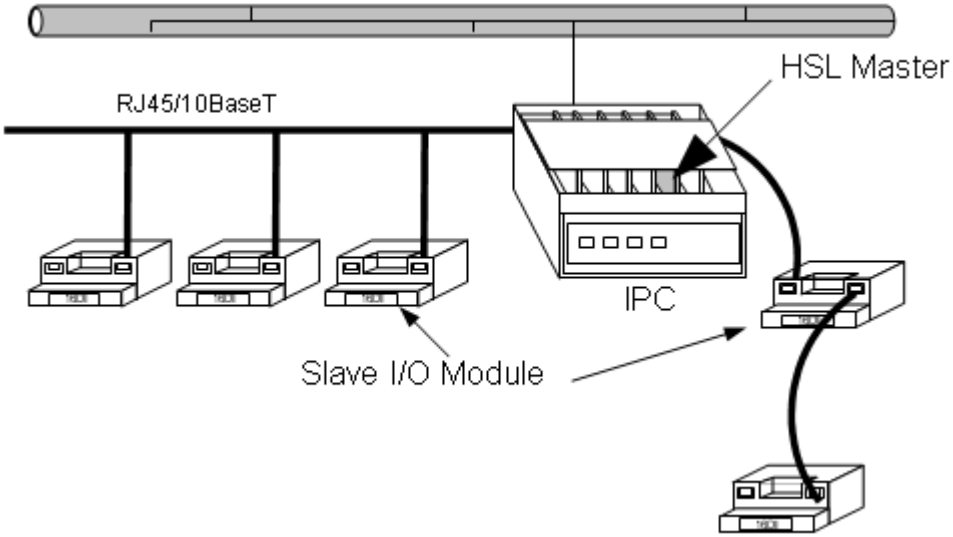


Figure 1-4: HSL as distributed PLC

Comparison between traditional PLC systems and HSL as distributed PLC

- The MPC is replaced by a PC with HSL Master
- The HSL slave I/O module is replaced by a remote side PLC
- The RS485 or RS232 cable is replaced by simple Ethernet cable
- The protocol handling is replaced by simple memory read/write.

HSL as Remote Time-deterministic DAQ

The HSL system, with high-speed performance and deterministic time-deterministic scanning, is also applicable for remote time-deterministic data acquisition.

The time-deterministic characteristic of an HSL system is an important factor when implementing a DAQ application. With an HSL system, all I/O data are refreshed in time-deterministic. The sampling rate (or scan rate) is linearly dependent on the number of slave indexed occupied, ranging from 91 μs (less than three slave indexes) to 1.911 ms (63 slave indexes) under 6 Mbps. These two features go with HSL's remote capability to make it suitable for remote DAQ applications, especially when time-deterministic is of utmost concern.

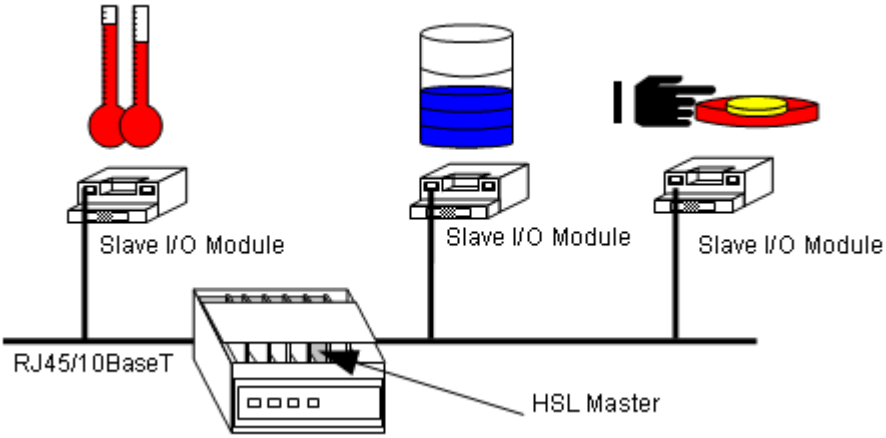


Figure 1-5: Time-deterministic DAQ using HSL

1.2 HSL System Specifications

Platform

- Hardware platform: Industrial PC with PCI Bus/Embedded SBC with PMC connector
- Operating system platform: Windows® 98/2000/NT/XP or Linux Redhat

Software support

- Windows XP/2k library
- Linux: Kernel 2.4.x

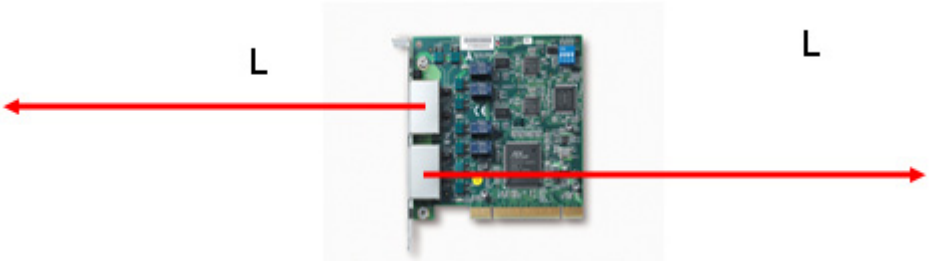
HSL Master Board

- PCI -7853 single HSL master controller board with two ports
- PCI -7854 dual-HSL master controller board with four ports
- PMC-7852/G dual-HSL master controller board with four ports and PMC connector

Remote operation

One master controller has two ports. One port uses the RJ-45 phone jack as connector. One phone jack can drive a maximum 32 modules at maximum. One master controller can connect maximum 63 slave indexes.

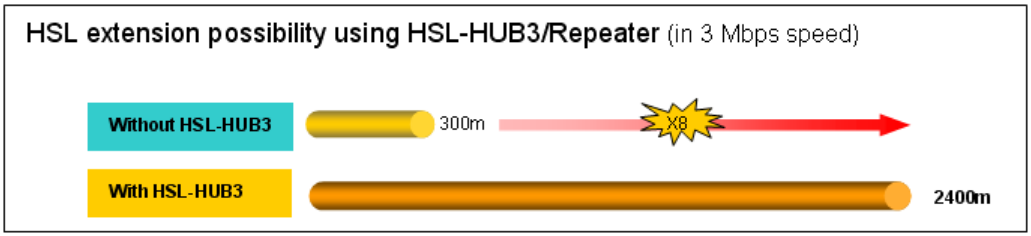
The maximum wiring distance for each RJ-45 connector (one port) is 200 m @ 6 Mbps (serial wiring from master to last slave module). The maximum length of port connection may be 400 m @ 6 Mbps since both sides are 200 m in length.



Transmission Speed	L (m)
3 Mbps	300
6 Mbps	200
12 Mbps	100

Table 1-1: Remote Operation

Supports maximum 2.4 km wiring via seven HSL-HUB/ Repeater modules



	Without HUB	HUBX1	HUBX2	HUBX5	HUBX7
12 Mbps	100 m	200 m	300 m	600 m	800 m
6 Mbps	200 m	400 m	600 m	1200 m	1600 m
3 Mbps	300 m	600 m	900 m	1800 m	2400 m

Wiring

- Connector: RJ-45 (on master controller and some of slave modules)
- Cable: Cat-5 100 Base/TX Ethernet cable with shielding

Communications

- Multi-drop full-duplex RS-422 with transformer isolation scheme
- Transmission speed: 3/6/12 Mbps (6 Mbps is factory default setting).
- I/O refresh rate: scan time unit × numbers of slave indexes (minimum is 3; maximum is 63)

	3 Mbps	6 Mbps	12 Mbps
Full Duplex	60.67 µs	30.33 µs	15.17 µs
Half Duplex	118 µs	59 µs	29.5 µs

- Communication model: single master to multi-slave
- Communication method: command/response type hand-shaking
- CRC12 and dedicated protocol for eliminating communication errors

1.3 HSL Series Products

HSL Master controller boards

See HSL Master Board on the previous section.

At least one master controller card is needed for an HSL system. With PCI-7854 or PMC-7852/G, two master controllers are available. A maximum of 12 cards are supported for a single computer system.

Slave I/O modules

A variety of HSL slave I/O modules are available.

Series	Model	Discrete Input	Discrete Output	Analog Input	Analog Output	Start Index Setting Range	Slave Index Occupation
DB	HSL-DI32-DB-N/P	32				(1,3, 5, …,61)	2
	HSL-DO32-DB-N/P		32			(1,3, 5, …,61)	2
	HSL-DI16DO16-DB-N/P	16	16			1-63	1
M	HSL-DI32-M-N/P	32				(1,3,…,61)	2
	HSL-DO32-M-N/P		32			(1,3,…,61)	2
	HSL-DI16DO16-M-NN/NP/PN//PP	16	16			1-63	1
	HSL-R8DI16-M-N/P	16	8 relay			1-63	1
	HSL-AI16AO2-M-VV			16	2	1-61	2
	HSL-AI16AO2-M-AV			16	2	1-61	2
U	HSL-DI16DO16-US/UJ	16	16			1-63	1
	HSL-DI16-UL	16				1-63	1
	HSL-AO4				4	1-62	2

Table 1-2: Slave I/O modules

Note: **Start Index Setting Range** means range of the start index address of DIP switch setting. Full duplex and half duplex mode have different ranges.

The following remote motion control modules are also supported:

Series	Model	Axes	Interface	Start Index Setting Range	Slave Index Occupation
Motion	HSL-4XMO-CG-N/P	4	General series	1~60 for Half Duplex 1~57 for Full Duplex	4
	HSL-4XMO-CD-N/P	4	D-sub		

Table 1-3: Remote Motion modules

Note: **Start Index Setting Range** means range of the start index address of DIP switch setting. Full duplex and half duplex mode have different ranges.

Terminal Base

A variety of HSL terminal base are also available.

Model Numbers	Module Type Support	Module Number Support
HSL-TB64-DIN	All the HSL DB series	2
HSL-TB32-U-DIN	All the HSL DB series	1
HSL-TB32-M-DIN	All the HSL M series	1
HSL-TB32-MD	All the HSL M series	1

Table 1-4: Terminal Base

1.4 Technical Information

1.4.1 HSL Technology Introduction

Inside an HSL system, a single master controller communicates with multi-slave through a command-response. The master controller sends commands to slave I/O modules for setting output values and requesting input information. Every slave module responds after receiving commands with address ID. The responses may either be to set output according to the received values or to reply requested input information to the master controller.

The illustration below shows the HSL working theory as regards the **setting of output values**.

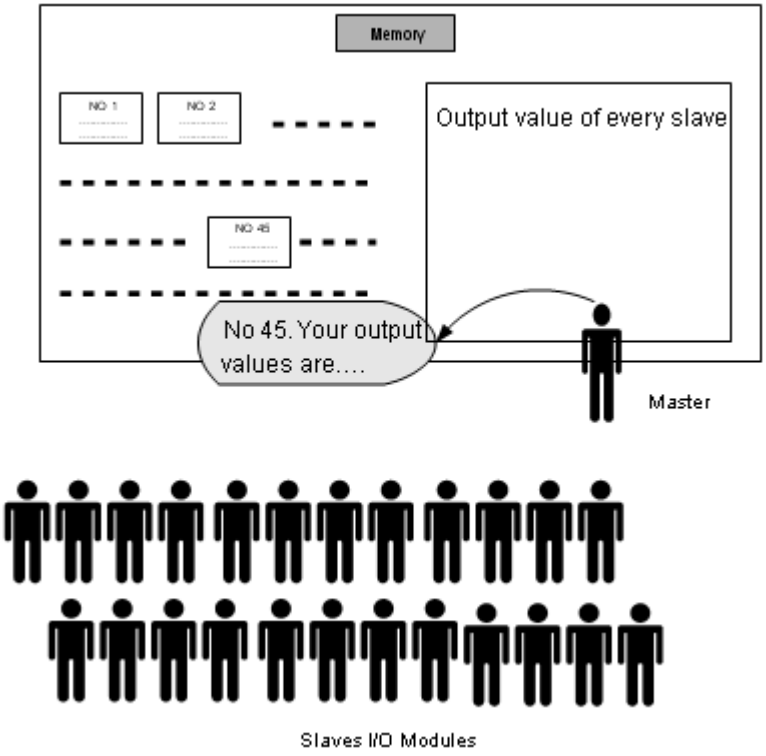


Figure 1-6: HSL technology brief -1

The teacher (master) sends message “ID.#, your output values are XXX” to all students (slave I/O modules). Every student (with ID.#) then sets its output channels according to the values heard. The values that the teacher announced to the students are written on the blackboard (RAM on master cards), and can be easily modified.

The following illustration shows the working theory for **gathering input information**.

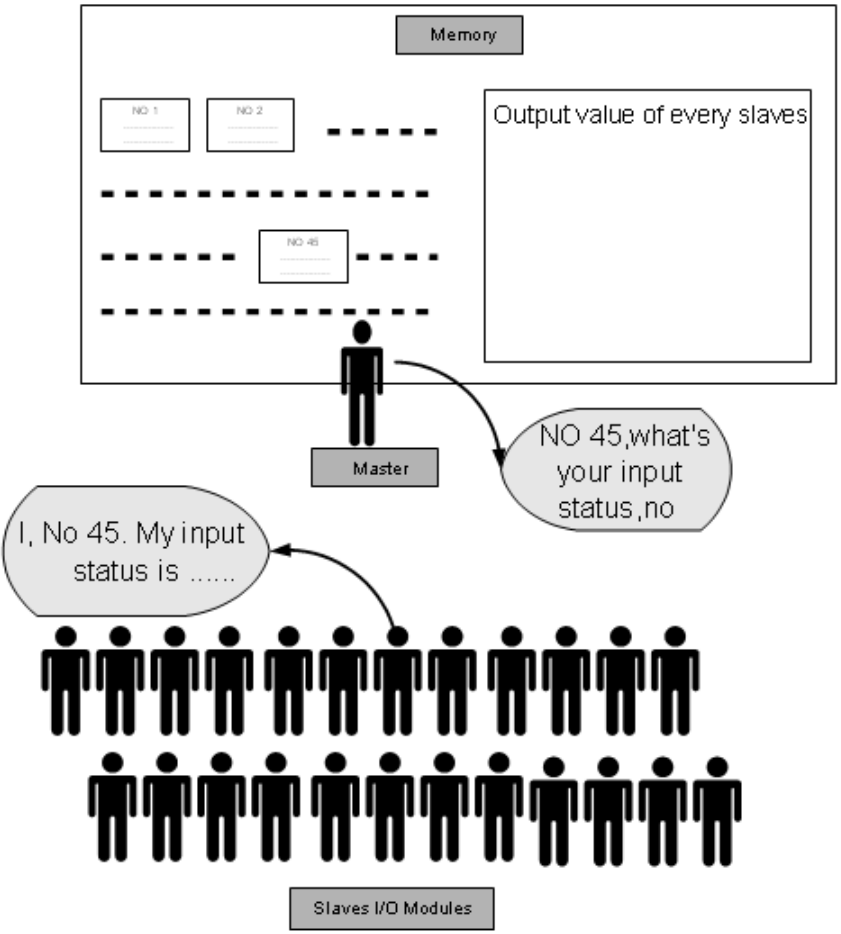


Figure 1-7: HSL technology brief-2

The teacher (master) sends the message “ID.#, what is your latest input status” to all students (slave I/O modules). Every student (with ID.#) then gives his answer. The teacher writes the answers on the blackboard (RAM on master cards). When someone (user’s AP) wants to know the students’ answers, he refers to the blackboard. All input information are saved in the memory.

These two procedures take turn and repeat on every slave module. After each cycle, each slave module sets its newest output status and the master gathers all these information from the memory. We simulate the polling communication cycle through a teacher-student conversation:

Teacher: Student No. 1, your output vales are ##, what’s your latest input status?

Student No. 1: My input status is ##

Teacher writes the answer on the blackboard.

Teacher: Student No. 2, your output vales are ##, what’s your latest input status?

Student No. 2: My input status is ##

Teacher writes the answer on the blackboard.

Until…

Teacher: Student No. 63, your output vales are ##, what’s your latest input status?

Student No. 63: My input status is ##

Teacher writes the answer on the blackboard.

The polling cycle is now complete. The process repeats from Student No. 1.)

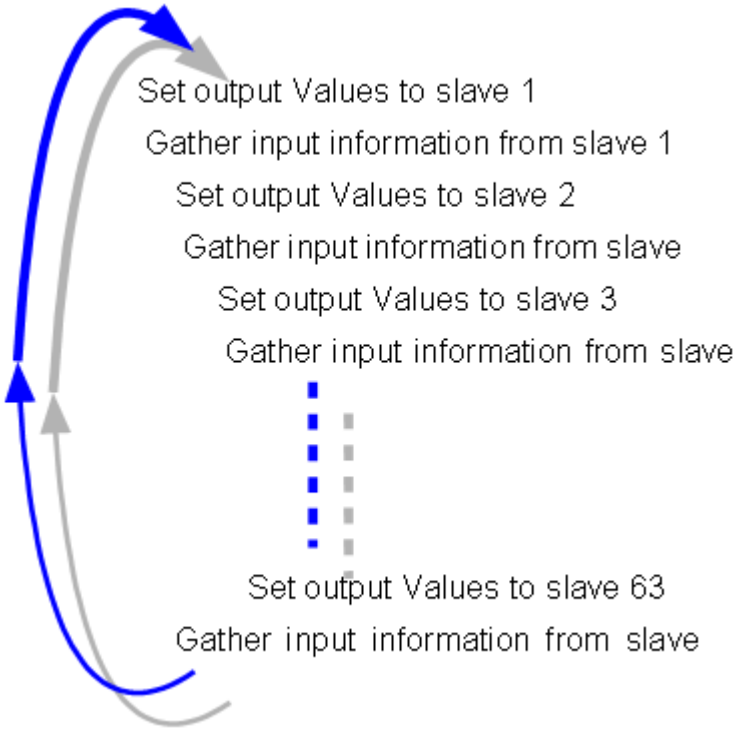


Figure 1-8: HSL I/O polling cycle

The HSL master-slave communication architecture is illustrated below:

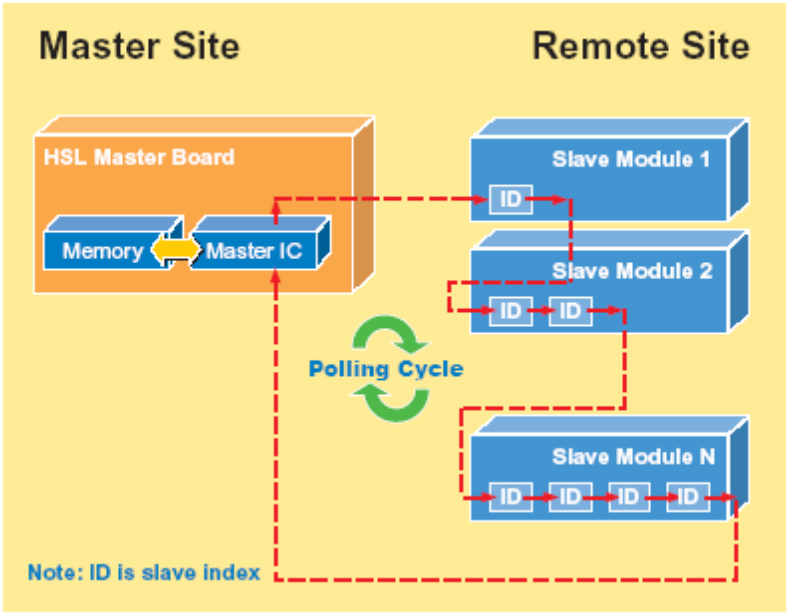


Figure 1-9: Master-slave communication architecture

1.4.2 HSL Terminology

In addition to the input/output polling mechanism shown above, here are some HSL-related syntax for your reference.

HSL Master. Master is defined as the teacher in Figure 1.9 and 1.10. The master takes charge of giving commands, including output value announcements and latest input status requests.

Slave I/O Module. Slave I/O modules are defined are the students in Figure 1.9 and 1.10. Slave I/O modules are passive components in an HSL system. They receive commands from the master, then respond by telling their newest input status or by setting output values. The slave I/O modules may take one or two address indexes depending on the I/O module type.

Polling Cycle. When communicating with slave I/O modules, the master takes turn setting the output for and gathering input from every slave module. When all slave modules are updated, a polling cycle is completed. The polling cycle infinitely repeats when the master is working properly.

I/O Refreshing Rate. The time needed to complete an I/O updating cycle. This may also be the longest time needed for any digital I/O channel to obtain its latest status. The refreshing rate is decided linearly by the total number of slaves used in an HSL system, therefore no interference occurs between any two HSL systems or PC/IPC at the same time.

Transmission Speed. May refer to the speed of digital signal transfer or I/O refreshing rate. When referring to the speed of digital signal transfer inside the cable, the transmission speed is known as data rate. The unit of transmission speed is bits per second (bps). The unit of I/O refreshing rate speed is expressed in mini-second (ms).

1.4.3 System Configurations

To develop an HSL application, you must know how to configure the HSL cards and slave I/O modules. The following sections describe the configuration concepts for an HSL system. For detailed information, refer to individual chapters.

Master Card Index (card_ID). You can install one or more HSL master boards in an IPC system. The PCI BIOS assigns the card index for each HSL master board. You need to specify the card index for programming purposes. The card index is starts from 0.

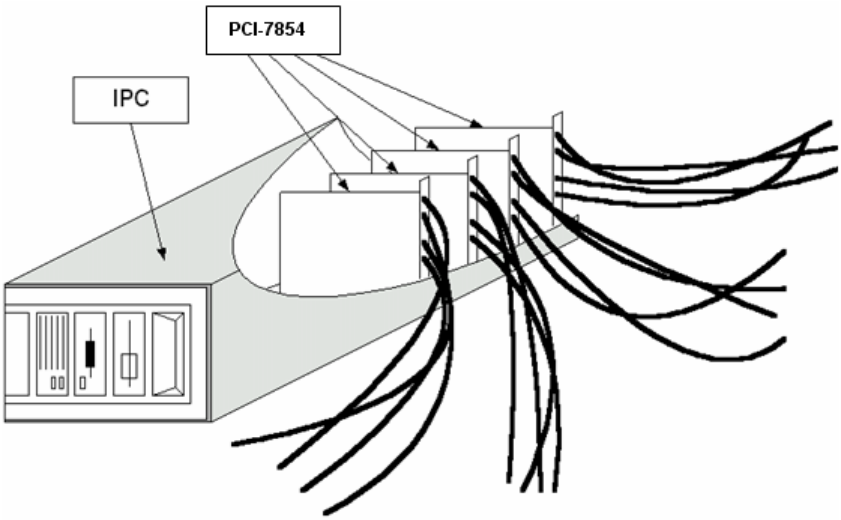


Figure 1-10: Multiple master cards in one IPC

Refer to Chapter 2: HSL Master Controller for more information.

HSL Connect Index (connect_index). The PCI-7853 provides only one HSL master controller (connect_index=0), while PCI-7854 or PMC-7852/G provides two HSL master controllers (connect_index=0 or 1). Connect index distinguishes the master controllers.

Ports. Port refers to the RJ-45 connector on the HSL master board. A connector is a loop of wiring that starts from the master card and connects to up to 32 slave I/O modules. There are two ports in one HSL master controller, both carrying the same signals sent from the master.

Slave Index. A complete HSL system is composed of one master and 1 to 63 slave indexes. The following diagram illustrates an HSL system.

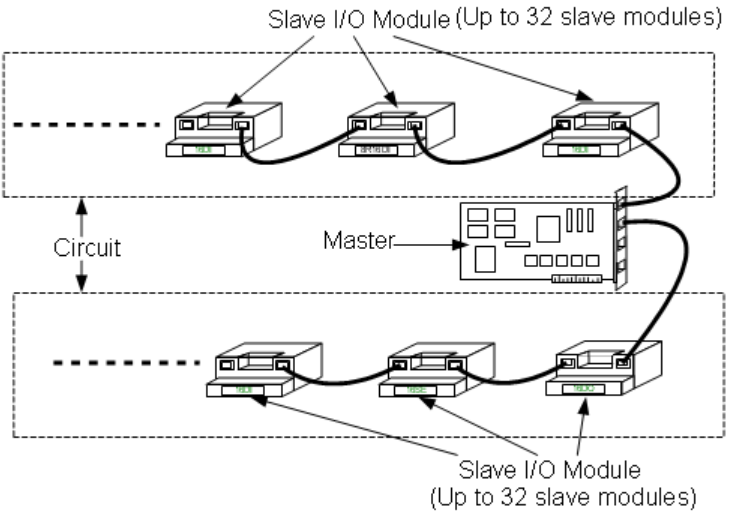


Figure 1-11: HSL system layout example-serial wiring

Every master circuit can support up to 32 slaves. However, since the slave address is assigned by a 6-bit DIP-switch on the I/O module's termination board with the value '0' reserved, the maximum number of slaves is 63, not 64. **The slave I/O modules in an HSL system must have different slave index**. Though the slave index may not necessarily continue from 1, continuous addressing is more efficient. When an HSL system has only two slave modules addressed 1 and 63, the I/O refreshing rate is the same with an HSL system with 63 slave modules addressed from 1 to 63.

Refer to Chapter 3: HSL Slave I/O Module for procedures on how to set the addresses of slave I/O modules.

1.4.4 Wiring

The HSL network follows a modified RS-422 electrical specification.

Wiring

The wire cables of an HSL system are carefully selected for installation convenience and standardization without sacrificing communication quality. A 100BaseTX cable with RJ-45 connectors is used on HSL systems.

Full-duplex RS-422 with multi-drop

The typical RS-422 is not a networking specification. However, HSL modified the specification to suit network applications. The TXD of the master is connected to the RXD of every slave I/O modules, while all TXD of slaves are connected to the RXD of the master. Only the master uses TXD (of master) to RXD (of slave) channel. Through design, only one slave at a time sends message with TXD (of slave) to RXD (of master) channel. This kind of networking solution is called RS-422 with multi-drop.

Ports

An HSL master supports a 2-port segment. Inside the ports, the TXD (of master) to RXD (of slave) channels are of the same signal sent by master. In contrast, the TXD (of slave) to RXD (of master) channels are isolated from each circuit and signals inside do not pass through the master from one circuit to another.

The diagram below illustrates RS-422 with multi-drop:

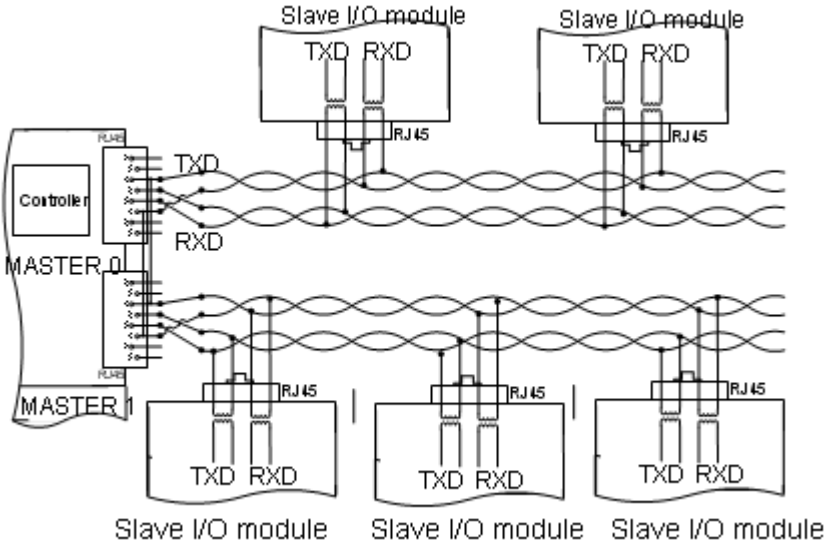


Figure 1-12: HSL wiring – RS-422 with multi-drop

- There are two RJ-45 notches in each master. Each notch supports one port of wiring.
- Only four lines of the RJ-45 cable are used: two for transmission and two for reception.
- All slave modules are connected in parallel.
- The isolation between connection cable and individual slave I/O modules protects the signals from interference by other slaves.

1.4.5 Networking Topology

Base on the RS-422/RS-485 architecture, a variety of topologies are available for an HSL circuit, including serial, multi-drop, star, etc. The following sections illustrate some applicable network topologies for your reference.

Serial wiring

All slave I/O modules are connected using a twin-head 100BaseTX cables.

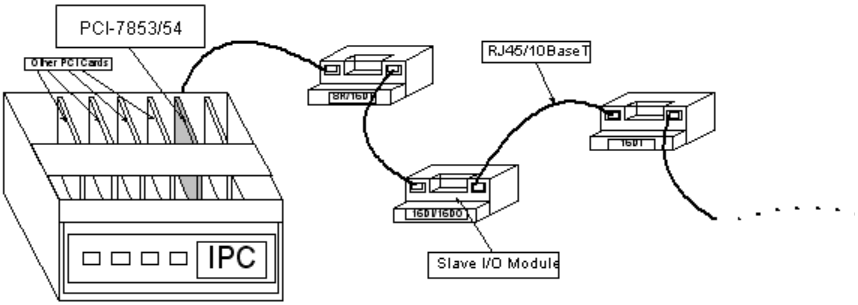


Figure 1-13: HSL networking topology – Serial

Since two notches of any slave I/O module's termination board are short circuit, this type of wiring provides the most convenient and intuitive way to form a HSL network. The longest wiring length is 200 m with under 6 Mbps.

1.4.6 I/O refreshing rate of an HSL system

The scan time unit for one slave index is set in 3/6/12 Mbps transmission rate. Once the maximum slave address is set, the polling cycle time of the HSL system is using the following formula:

*Polling cycle time = maximum-address-of-slave * scan time unit*

Maximum address	Cycle Time under 3 Mbps	Cycle Time under 6 Mbps	Cycle Time under 12 Mbps
5	303.33 µs	151.67 µs	75.83 µs
10	606.67µs	303.33 µs	151.67 µs
20	1.213 ms	606.67 µs	303.33 µs
30	1.820 ms	910.00 µs	455.00 µs
40	2.427 ms	1.213 ms	606.67 µs
50	3.033 ms	1.516 ms	758.33 µs
60	3.640 ms	1.820 ms	910.00 µs
63	3.822 ms	1.911 ms	955.50 µs

Table 1-5: Polling cycle time of HSL (Full Duplex Mode)

Note: Regardless of the transmission rate you select, the minimum polling cycle time is 3×scan time unit, even when the maximum address is less than 3. Refer to Appendix A.

1.4.7 Communication error handling

The HSL communication protocol is designed to eliminate any error, there may be some chances of communication errors such as light striking, sudden off-line, etc. In an HSL system, the master holds an **accumulated slave-no-response count** for every individual slave I/O module. The count value is updated for each slave module in every polling cycle.

- If communication with certain slave I/O module is successful, the no-response count value for this slave is set to 0.
- If the communication failed, the no-response count value increases by 1.
- When the count is larger than or equal to 3, a binary flag indexing communication error is set to **True**.
- The maximum value of no-response count is 7. The value is retained even if the error continues to occur.

The *no-response count value* and the *communication error flag status* may be obtained by software function call. These error handling data are also returned every time the user wants to set or get the I/O values.

In addition, the HSL, through a software communication error-handling driver, supports a self-diagnosis function that detects off-line or out-of-communication slave modules. In a programmed period of time (default 20 ms), the master sends an IRQ to trigger the driver to check the slave's no-response count value. If the count value is 7, the driver informs the system that the slave module is off-line.

1.5 Software Support

Window® 2000/XP DLL

The provided Windows® 2K/XP DLL (Dynamic Link Library) is a programming interface for systems using Microsoft® Windows®. The driver works with any Windows® programming language that integrates DLL such as Visual C/C++ (6.0 or above), Borland C++(5.0 or above), or Visual Basic (6.0 or above).

Linux Driver

The HSL Linux driver includes device drivers and shared library for Linux-based systems. The developing environment can be GNU C/C++ or any programming language that allows linking to a shared library. The Linux driver is included in the ADLINK All-in-one CD.

2 HSL Master Controller

The HSL master is the key component in charge of communicating with slave I/O modules. The master sets output values to and gathers input information from slaves.

ADLINK presents four types of HSL master cards: PCI-7853, PCI-7854 and PMC-7852, and PMC-7852/G. The main difference between these cards is the number of supported HSL master.

- PCI-7853: Single HSL Master Controller Interface Card
- PCI-7854: Dual-HSL Master Controller Interface Card
- PMC-7852/G: Dual-HSL Master Controller Interface Card with PMC connector

2.1 Board Overview

Figure 2-1: PCI-7854 front view

2.2 Specifications

PCI Bus

- PCI local bus specification Rev. 2.1-compliant

Master Controller

- HSL ASIC master controller
- 48 MHz external clock

Interface

- RS-422/RS-485 with transformer isolation
- Half/Full duplex communication
- 3/6/12 Mbps transmission rate through a S/W function setting (default is 6 Mbps)
- Two ports for each master controller

Connector

- RJ-45 connector x 2 (CN1 for PCI-7853)
- RJ-45 connector x 4 (CN1, CN2 for PCI-7854; H1A, H1B, H2A, H2B for PMC-7852/G)

Interrupt

- 16-bit programmable timer with 5μs resolution

LED Indicator

- Link status

Dimension

- PCI-7853/7854: 122 (L) × 107 (W) mm
- PMC-7852/G: 74 (W) × 149 (W) mm

Operating Temperature: 0°C to 70°C

Storage Temperature: -20°C to 80°C

Power Consumption: +5V @ 500 mA typical

2.2.1 PCI-7853/7854 Layout

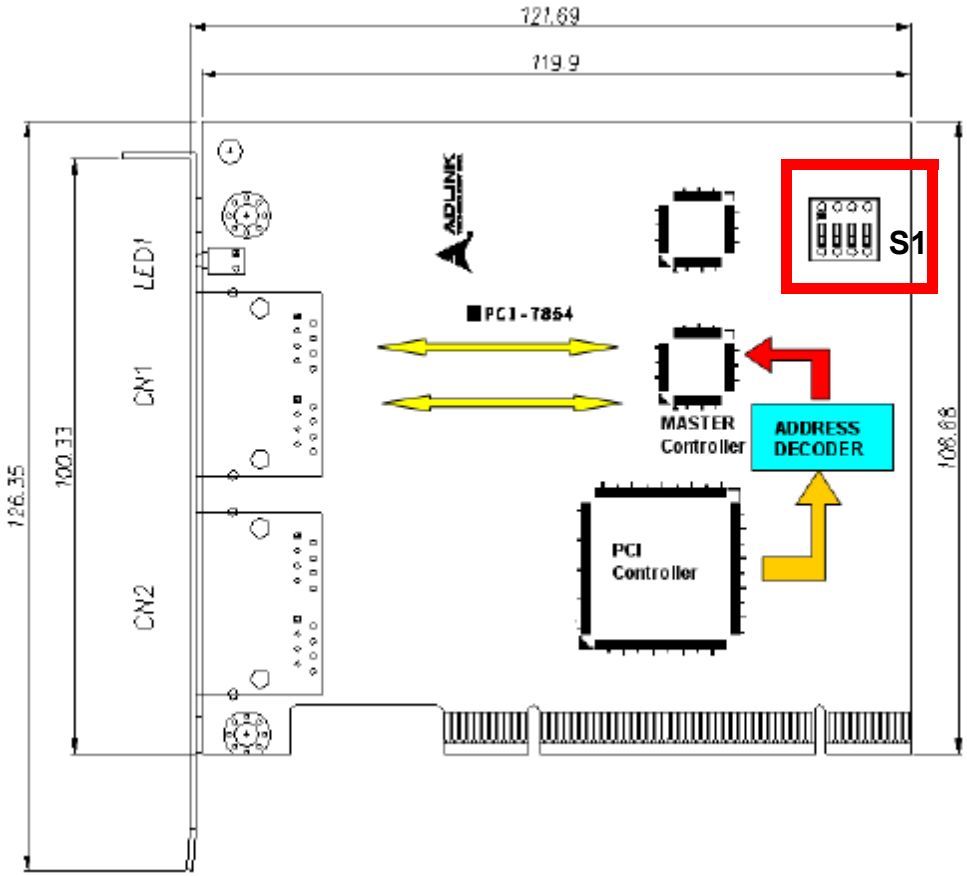


Figure 2-2: PCI-7853/7854 Layout

CN1: RJ-45 connector with first HSL master controller

CN2: RJ-45 connector with second HSL master controller (PCI-7854 only)

S1: Card ID switch selection

2.2.2 PMC-7852/G Layout

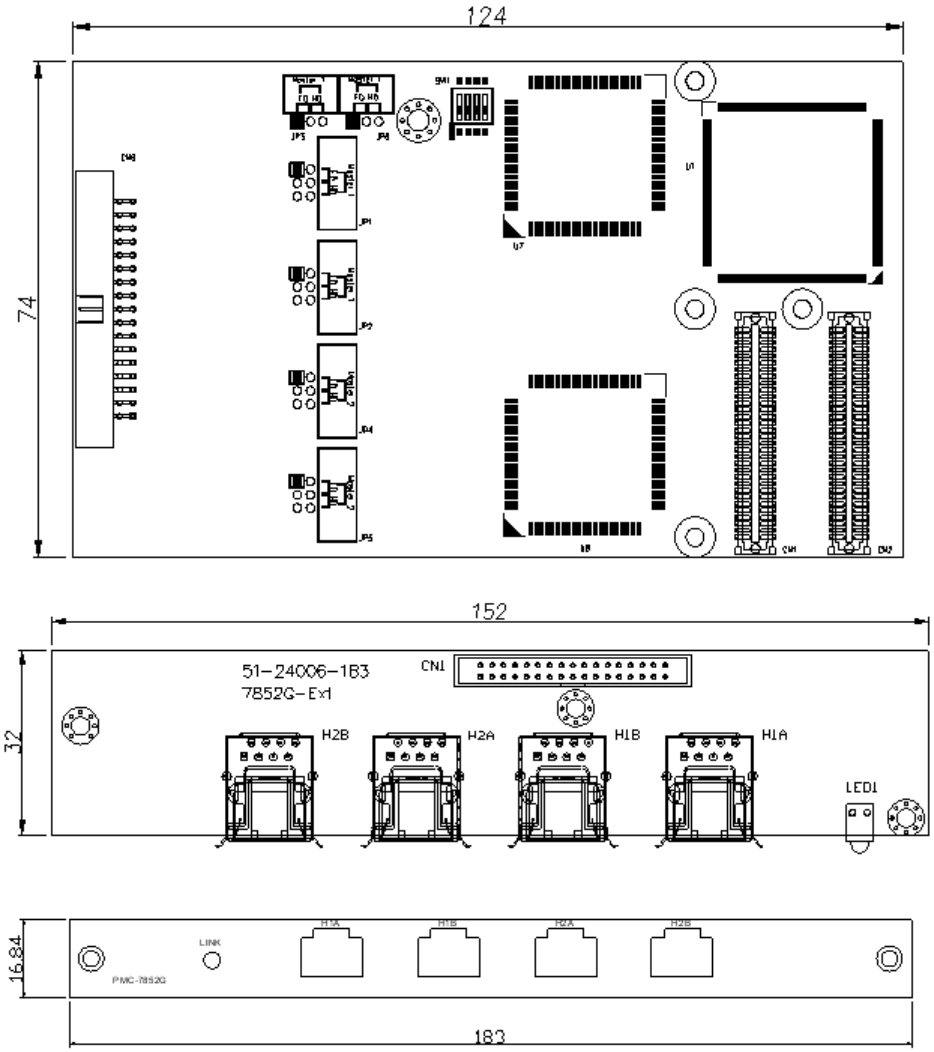


Figure 2-3: PMC-7852/G Layout

- **H1A, H1B**: RJ-45 connector with first HSL master controller
- **H2A, H2B**: RJ-45 connector with second HSL master controller (PMC-7852 only)

- **JP1, 2, 3, 6**: Full/Half duplex mode with first master controller (Default: Full duplex mode)
- **JP4, 5**: Full/Half duplex mode with second master controller (Default: Full duplex mode)
- **SW1**: Transmission speed option. (Default: 6 Mbps)

2.3 Configuration

2.3.1 SW1 (PMC-7852/G only)

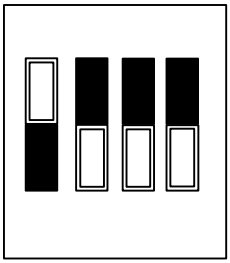


Figure 2-4: SW1 – Transmission Rate Setting

No.	Set transmission rate				
1	OFF	ON	OFF	ON	1st Master Controller
2	OFF	OFF	ON	ON	
3	OFF	ON	OFF	ON	2nd Master Controller
4	OFF	OFF	ON	ON	
Rate	12M	6M	3M	EXC	

Default: 6 Mbps.

2.3.2 JP 1, 2, 3, 6 / JP 4, 5 (PMC-7852/G only)

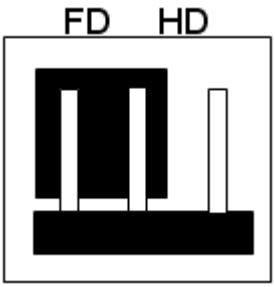
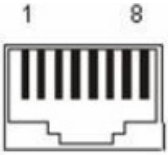


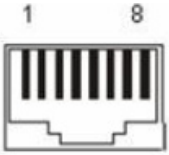
Figure 2-5: Top View of PMC-7852/G, for JP 1, 2, 3, 4, 5, 6 jumper settings.

2.4 PIN Assignment (female)



RJ45 Female Connector for PCI-7851 / 7852 / 7853 / 7854

PIN No.	PINOUT
1	NC
2	NC
3	RX+
4	TX-
5	TX+
6	RX-
7	NC
8	NC



RJ45 Female Connector for HSL-TB64-DIN, HSL-TB32-U-DIN, HSL-TB32-DIN, HSL-TB32-DO-DIN, HSL-TB32-M-DIN, HSL-TB32-MD

PIN No.	PINOUT
1	NC
2	NC
3	TX+
4	RX-
5	RX+
6	TX-
7	NC
8	NC

2.5 Software Architecture Description

The PCI-7853/PCI-7854/PMC-7852/G comes with one or two HSL master ASICs that control the HSL communication. The purpose of communicating with HSL I/O modules is to gather input data from or set output value to them. To achieve this purpose, each HSL master controller manages a 2Kbyte SRAM on PCI-7853/ PCI-7854 boards for data storage.

For every polling cycle, the master refreshes all input data from the I/O modules and sets the latest output data to the I/O modules. The SRAM keeps these data for the software drivers to read/write I/O information.

2.5.1 Functional Block Diagram

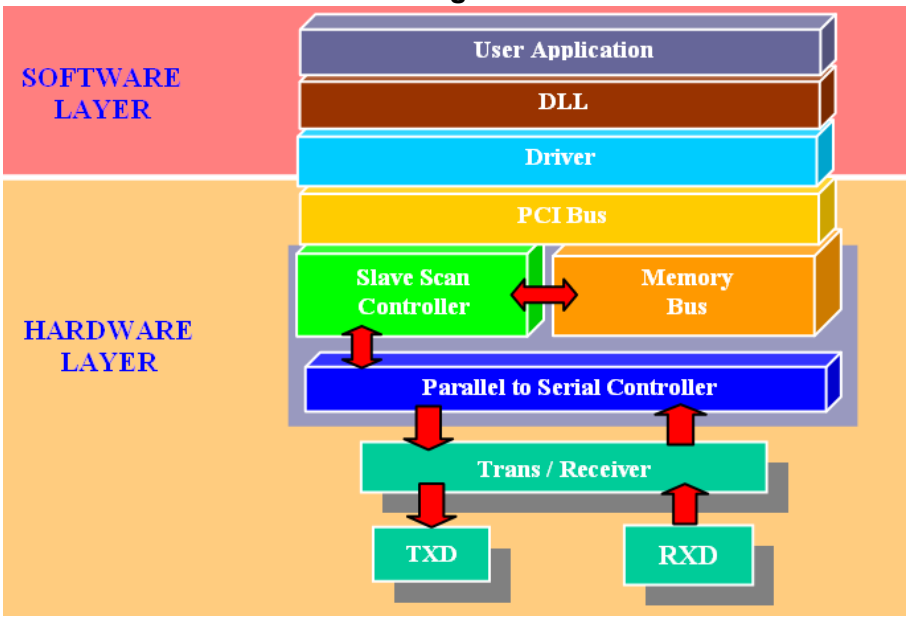


Figure 2-6: Functional Block diagram of HSL master

The diagram above shows how the HSL communicates with the user's AP. The SRAM acts with a buffer-like characteristic.

2.6 Installation

2.6.1 Hardware Installation

The PCI-7853/PCI-7854 is a plug and play device, with the system BIOS automatically assigning the interrupt channel and memory mapping address. You only need to adjust the SW1 to the desired transmission speed. For PMC-7852/G, refer to GEME user manual for details.

2.6.2 Software Installation

Refer to chapter 4.1.

3 HSL Slave Module

The HSL is a master-slave network system that features an innovative distributed architecture that modularizes the communication, I/O functions and signal termination. ADLINK provides a complete line of slave I/O modules and terminal bases including discrete I/O, analog I/O, and motion control to meet your application requirements. For motion control modules, refer to the HSL-4XMO user's manual.

Slave I/O Module. There are three groups of slave I/O modules with varied dimensions. The slave I/O modules provide the terminal base with different levels of I/O capability. To identify each slave I/O module in an HSL network, an electronic data sheet is embedded in the module, and each module is identified by an address ID configurable via the 6-bit DIP switch. Depending on the I/O support, each slave I/O module may be assigned one or two address IDs. Since the highest ID number in an HSL master is 63 (6-bit and '0' reserved for master), up to 63 slave I/O modules are supported in one HSL master.

Terminal Base. Offers an easy wiring media. Both power and signal wiring go from the terminal base to the slave I/O modules. Also, master links to all slave I/O modules via the terminal base using an RJ-45 cable. The TB makes the slave I/O modules hot-swappable without interfering other modules in the same HSL network.

HUB/Repeater. Provides sub-system support for various network topologies.

Wiring Cable. The cables connecting the HSL master and slave I/O modules are standard 100 Base/TX with RJ-45 connectors. These are exactly the same with commercial Ethernet cables.

3.1 Slave I/O Module

3.1.1 Discrete I/O Module

ADLINK provides three I/O module series: DB, M and L.

- DB: Daughterboard form factor
- M: Daughterboard form factor with aluminum cover
- U: U-series

Series	Model	Discrete Input	Discrete Output	Relay Output	Slave Index Occupation
DB	HSL-DI32-DB-N/P	32			2 (Consecutive from odd number)
	HSL-DO32-DB-N/P		32		2 (Consecutive from odd number)
	HSL-DI16DO16-DB-N/P	16	16		1
M	HSL-DI32-M-N/P	32			2 (Consecutive from odd number)
	HSL-DO32-M-N/P		32		2 (Consecutive from odd number)
	HSL-DI16DO16-M-NN/NP/PN//PP	16	16		1
	HSL-R8DI16-M-N/P	16		8	1
U	HSL-DI16DO16-US/UJ-NN/NP/PN/PP	16	16		1
	HSL-DI16-UL	16			1

Below is the selection guide.

HSL	-	DIxDOx	-	x	-	XY

Discrete I/O Type:
DI16DO16: 16 discrete inputs and 16 discrete outputs
DI32: 32 discrete inputs
DO32: 32 discrete outputs
R8DI16: 8 relay outputs and 16 discrete inputs

Series:
DB: Daughter board form factor
M: Daughter board with aluminum cover
U: U-Series

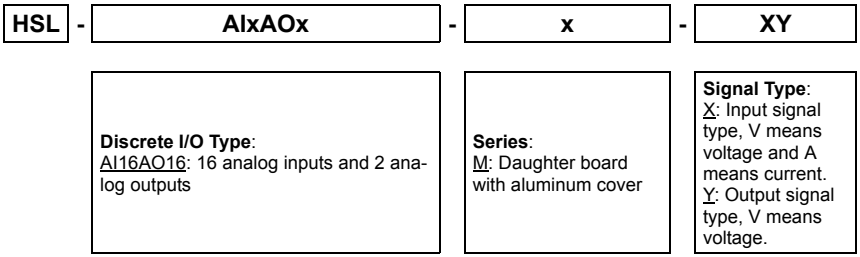
Signal Type:
X: Input Signal Type: **N**PN sinking and **P**NP sourcing support
Y: Output Signal Type: **N**PN sinking and **P**NP sourcing support

3.1.2 Analog I/O Module

ADLINK provides M series analog I/O module.

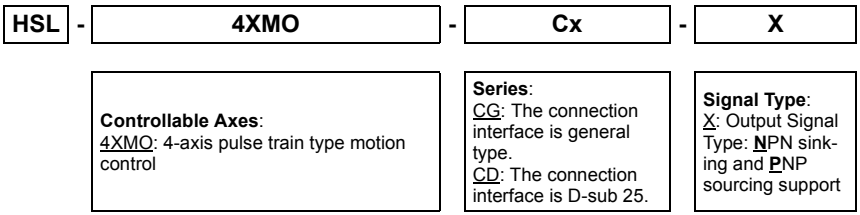
Series	Model	Analog Input	Analog Output	Slave Index Occupation
M	HSL-AI16AO2-M-VV	16	2	2 (Leap number)
	HSL-AI16AO2-M-AV	16	2	2 (Leap number)
U	HSL-AO4		4	2

Below is the selection guide.



3.1.3 Motion Control

ADLINK provides the HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P remote motion control cards. Below is the selection guide.



The HSL-4XMO-CG-N/P is suitable for applications using stepper and linear motors. For HSL-4XMO-CD-N/P, ADLINK provides the accessories and transfer cable for direct connection to a servo amplifier. For details, refer to the HSL-4XMO user's manual.

3.1.4 General Specifications

Discrete I/O Module

Discrete Input	Photo couple isolation	2500 V_{RMS}	
	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(5): -50mA/ch at 24V_{DC}
		For PNP(4)	All channels: +50mA/ch at 24V_{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 rated load	
	Response Time	ON to OFF: 3 μs (max)	
		OFF to ON: 6 μs (max)	

(1): NPN sinking type sensor input module

(2): PNP sourcing type sensor input modules

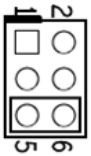
(3): NPN sinking type sensor output module

(4): PNP sourcing type sensor output modules

(5): U-series all channels: -90 mA at 24 V_{DC}

***Note**: The HSL-DI16-UL supports 5 V, 12 V, and 24 V, selected by a jumper for each channel:

- JDI0 - JDI15 (input voltage setting)



5 V
12 V
24 V (default)

Discrete Input	Input impedance	4.7 kΩ (@24 V_{DC}, 2.74 k (@12 V_{DC}), 1.1 k (@5 V_{DC})	
	Operation Voltage	DI_COM @ 24 V_{DC}	ON: 14.0 V_{DC} (max) OFF: 18.0 V_{DC} (min)
		DI_COM @ 12 V_{DC}	ON: 6.0 V_{DC} (min) OFF: 8.0 V_{DC} (max)
		DI_COM @ 5 V_{DC}	ON: 1.0 V_{DC} (min) OFF: 3.0 V_{DC} (max)

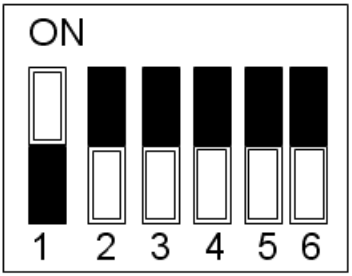
Analog I/O Module

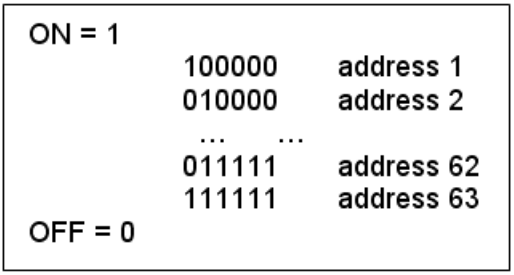
Analog Input	A/D Resolution	16-bit (14-bit guaranteed)
	Input Range	For VV type: ±10, ±5, ±2.5, ±1.25 V
		For AV type: 20 mA, 10 mA, 5 mA
	A/D Conversion	10 µs
	Signal Type	16-CH single-ended; 8-CH differential
Analog Output	D/A Resolution	16-bit
	DA Settling Time	10 µs

Motion Control

Refer to the HSL-4XMO user's manual.

3.1.5 DIP Switch Setting:





Notes

(1) The address (or slave index) 0 is reserved.

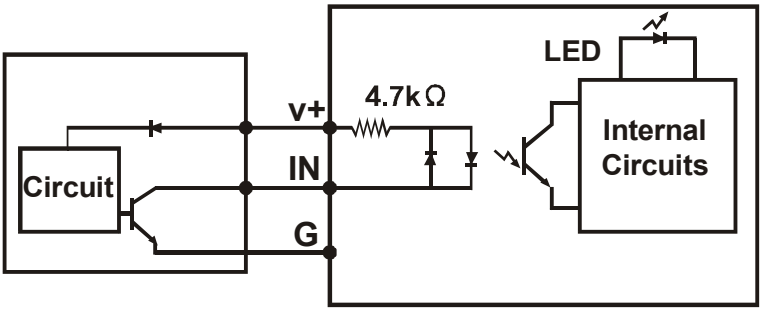
(2) The HSL-DI32-M, HSL-DO32-M, HSL-DI32-DB, and HSL-DO32-DB require two consecutive addresses starting from an odd number. For example, if the DIP switch is set to 3, it occupies slave index 3 and 4.

(3) The HSL-AI16AO2-M-VV/AV requires two leap addresses at full duplex mode. For example, if the DIP switch is set to 2, the module occupies addresses 2 and 4.

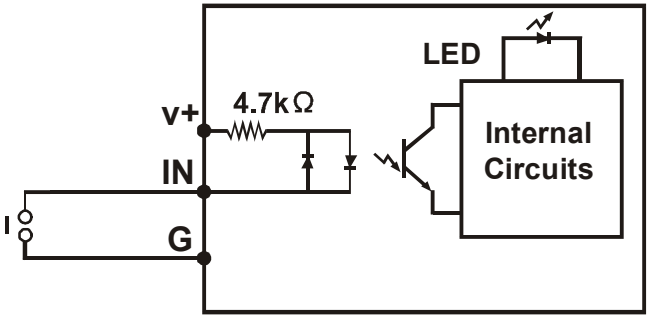
(4) The HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P require four leap addresses at full duplex mode. For example, if the DIP switch is set to 2, these modules occupy 2, 4, 6, and 8. At half duplex mode, it requires 4 consecutive addresses.

3.1.6 Wiring Diagram

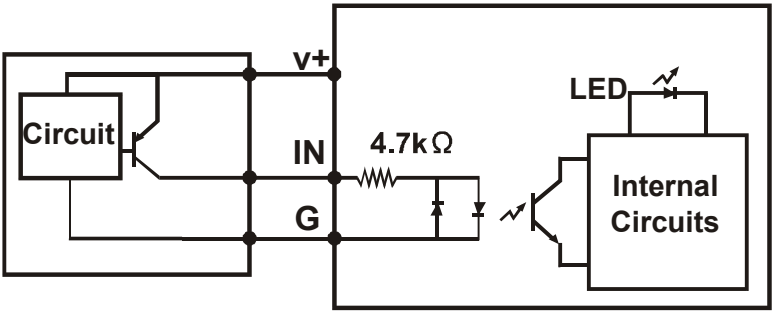
-N NPN Sinking type sensor Input



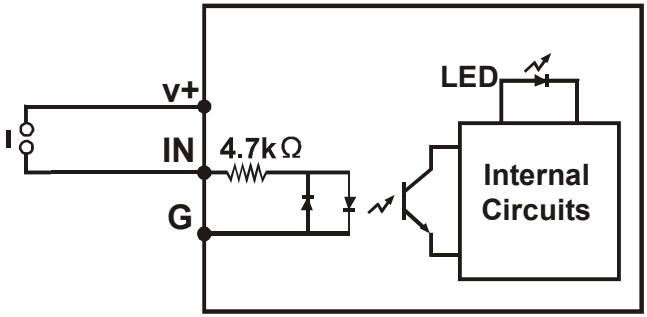
-N Dry Contact Input



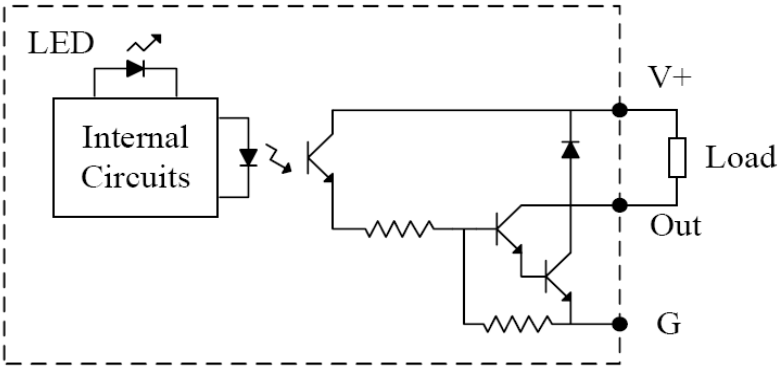
-P PNP Sourcing type sensor Input



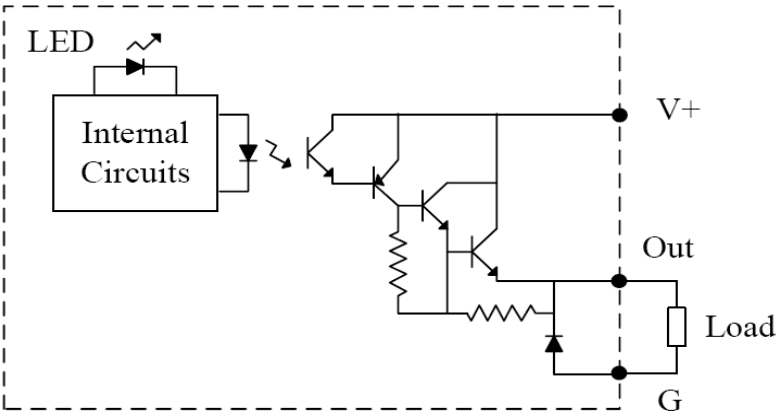
-P Wet Contact Input



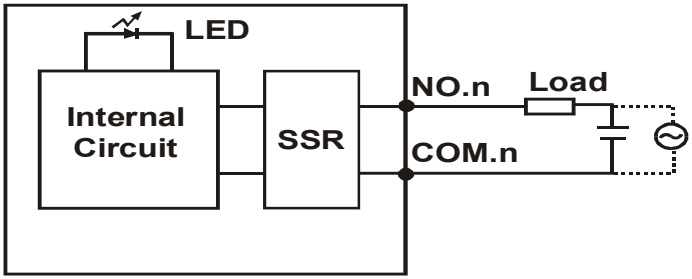
-N NPN Sinking Output



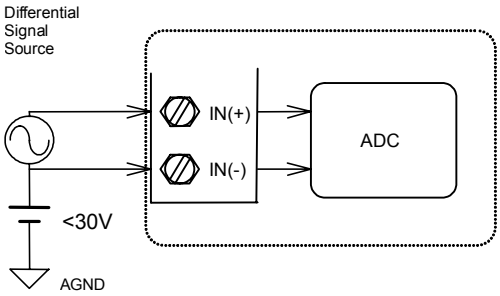
-P PNP Sourcing Output



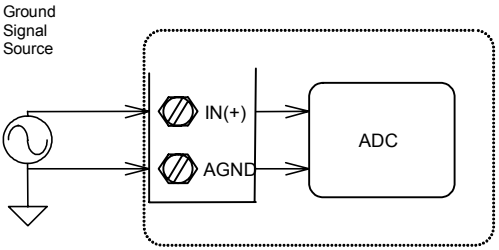
-R Relay Output



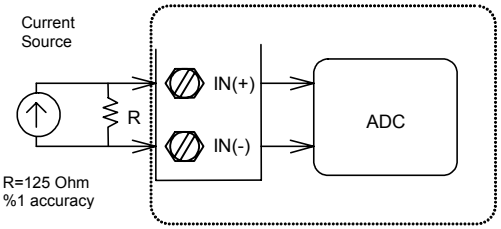
Analog Input (Differential Voltage Input)



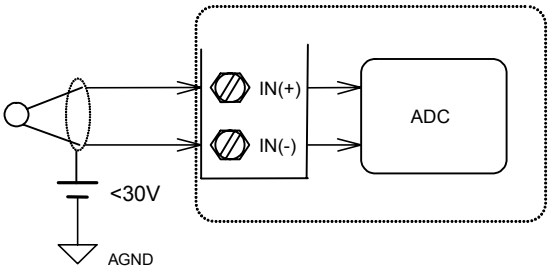
Analog Input (Single-End Voltage Input)



Analog Input (Current Measure)

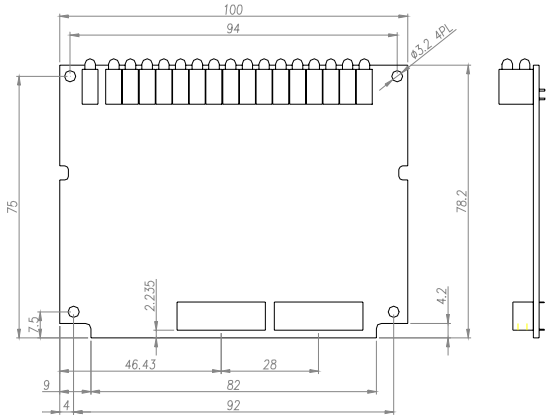


Thermocouple Measurement

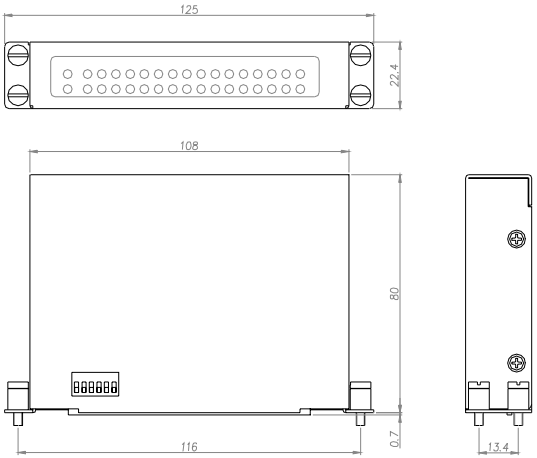


Dimension

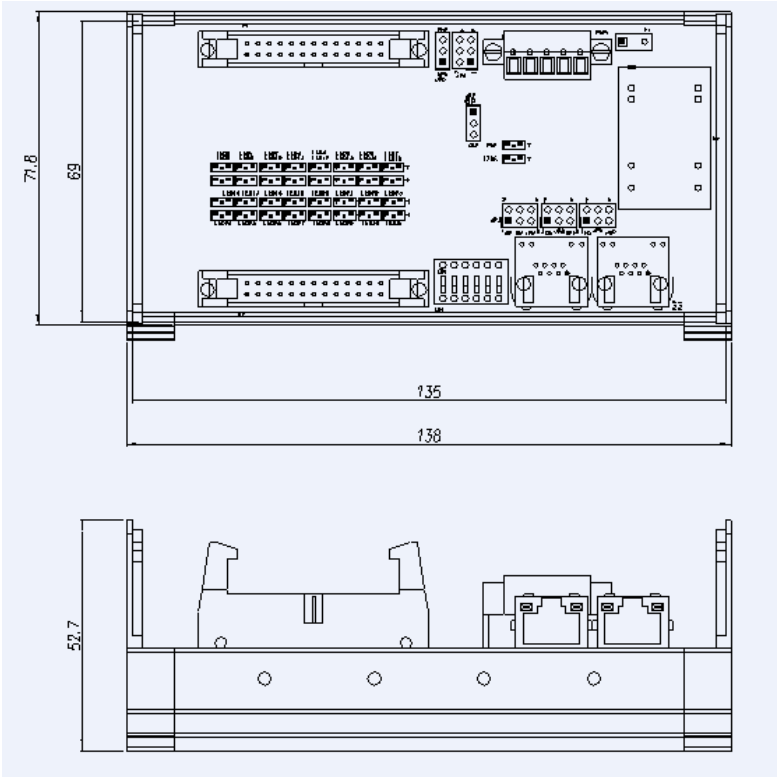
-DB Daughterboard form factor (100 mm X 78.2 mm)



-M Daughterboard with aluminum cover (125 mm X 80 mm)



-U U-series slave I/O module (71.8 mm X 138 mm)



3.2 Terminal Base

Available terminal bases 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-type terminal for easy field wiring
- Power and ground connections for each signal channel
- Interlocking design for installation in rugged environments
- Power LED indicator
- DIN rail mounting
- Onboard terminator resistor

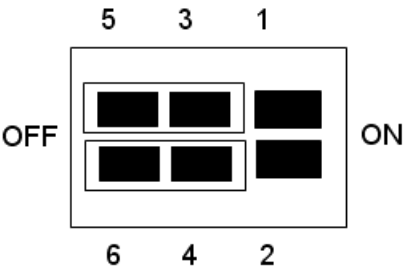
3.2.1 General Description

	Model Name	Specifications	Supports
For DB Series	HSL-TB32-U	(1): 32 channels direct connected terminal base (2): One DB slot	All HSL DB-series modules
	HSL-TB64	(1): 64 channels direct connected terminal base (2): Two DB slots	All HSL DB-series modules
For M Series	HSL-TB32-M HSL-TB32-MD	32 channels direct connected terminal base for HSL M-series module	All HSL M-series modules

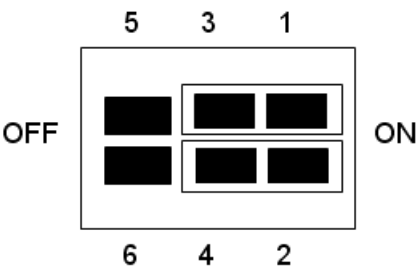
3.2.2 Jumper Settings

Since HSL is a serial transmission system, a terminator must be placed at the end of the cable. Each TB has an onboard jumper selectable terminator. **The terminator must be enable only by the last module**.

Not the last module (Default)

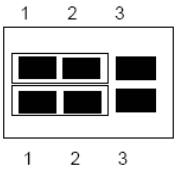


The last module



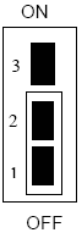
3.2.3 HSL-TB32-MD Jumper Settings

JP1,2 (External Power Option)



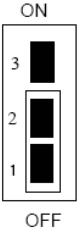
1, 2 short: Different Power (Default)
2, 3 short: Common Power

JP3 (Tx Terminal Resistor)



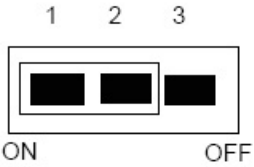
OFF is default setting

JP4 (Rx Terminal Resistor)



OFF is default setting

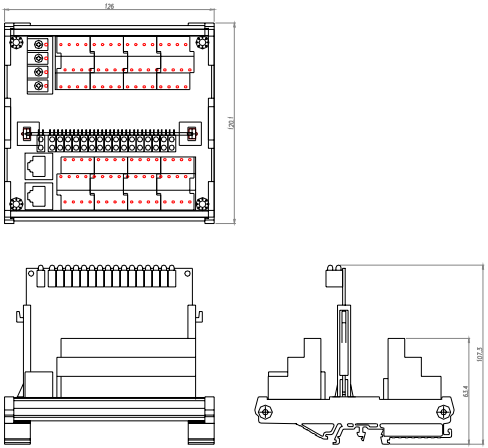
JP5 (Fuse Option)



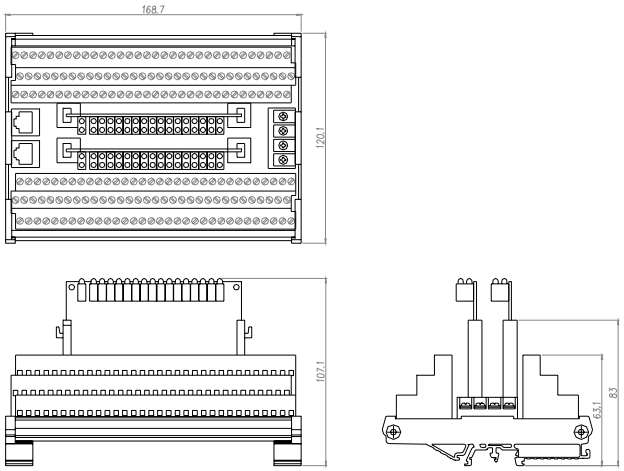
1, 2 short: With Fuse
2, 3 short: Without Fuse (Default)

3.2.4 Dimensions

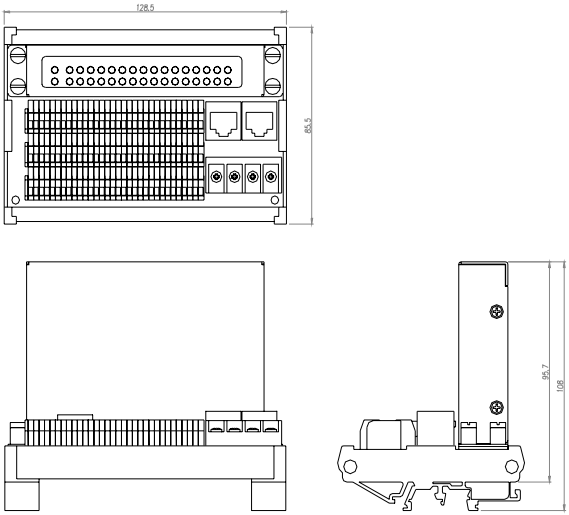
-DB with HSL-TB32-U-DIN (126 mm x 120.1 mm x 107.3 mm)



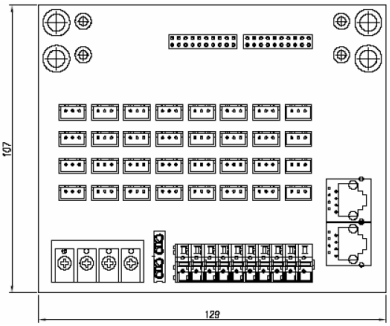
-DB with HSL-TB64-DIN (168.7 mm x 120.1 mm x 107.3 mm)



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



-HSL-TB32-MD (129 mm x 107 mm)



3.3 HSL-HUB/Repeater

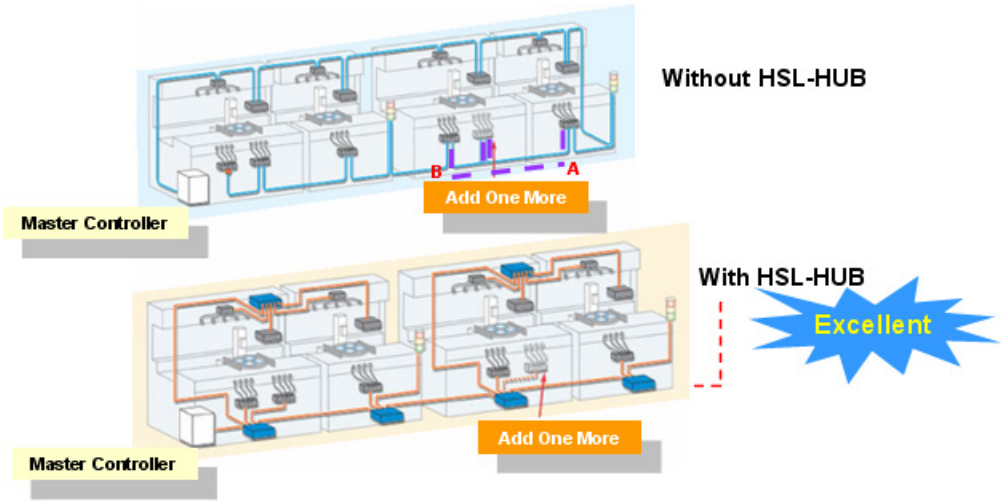
Available HSL-HUB/Repeater includes:

- HSL-HUB
- HSL-Repeater

Features

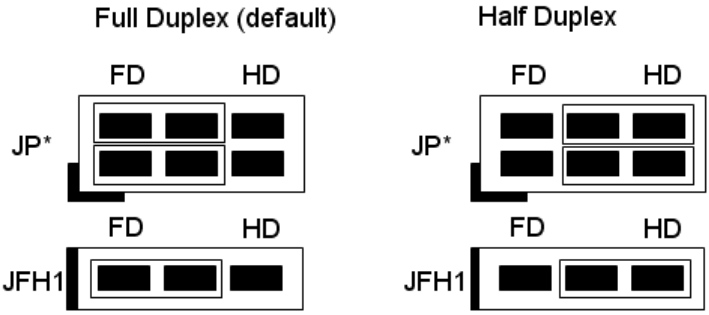
- Master to HUB, HUB to HUB, HUB to Slave link styles
- Supports T-bracing connection/Star connection (subsystem concept)
- Supports up to 2.4 km wiring distance via seven HSL-HUB/ Repeater modules
- One input port with three output segment ports
- Jumper configurable 3/6/12 Mbps transmission speed
- Jumper configurable full and half duplex transmission modes
- RJ-45 phone jack for easy installation
- 24 VDC input

3.3.1 General Description

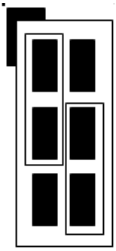


3.3.2 Jumper Setting

FD/HD setting JP*(0 to 3), JFH1



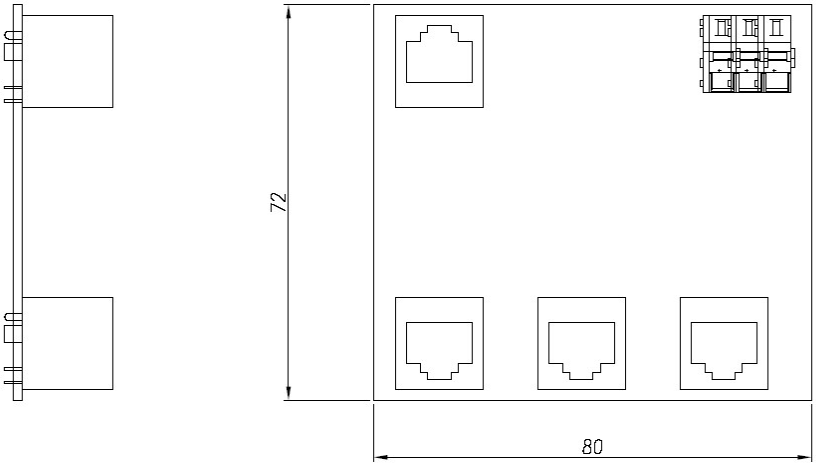
3/6/12 Mbps setting: JBPS1



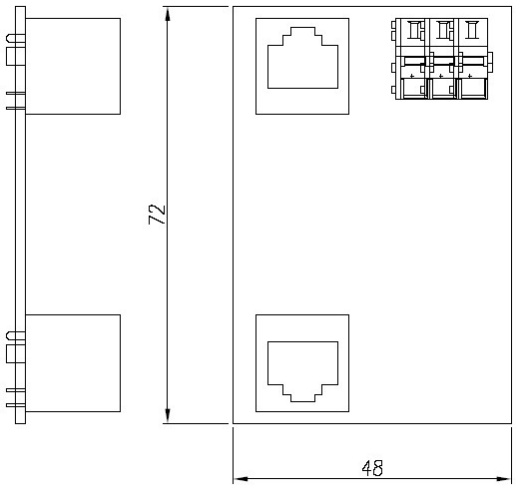
1 – 3 and 2 – 4	12 Mbps
1 – 3 and 4 – 6	6 Mbps (default)
3 – 5 and 2 – 4	3 Mbps
3 – 5 and 4 – 6	EXC

3.3.3 Dimensions

HSL-HUB



HSL-Repeater



3.4 Managing Slave Index in an HSL Network

3.4.1 Before you proceed

Before powering on the slave modules, you have to adjust the DIP switch. For more information, refer to section 3.1.6. Take note of the following:

1. A master controller can connect up to 63 slave indexes. For example, the PCI-7852 has two master controllers. Therefore, it supports a maximum 126 slave indexes.
2. The more compact the slave addresses are in an HSL network, the greater efficiency.
3. Observe the discrete I/O and relay module rule.

Module	Slave Index Occupation	Transmission Mode	Transmission Speed
HSL-DI16DO16-M-NN/NP/PN/PP	1 (Any address)	Full Duplex (Fixed)	6 Mbps (Fixed)
HSL-DI16DO16-DB-NN/NP/PN/PP			
HSL-R8DI16-M-N/P			
HSL-DI8-L-N/P			
HSL-DO8-L-N/P			
HSL-DI4DO4-L-NN/NP/PN/PP			
HSL-DI16-UL			
HSL-DI16DO16-UJ/US			
HSL-DI32-M-N/P	2 (Consecutive from odd number)		
HSL-DI32-DB-N/P			
HSL-DO32-M-N/P			
HSL-DO32-DB-N/P			

4. Observe the analog I/O and thermocouple module rule.

Module	Slave Index Occupation	Transmission Mode	Transmission Speed
HSL-AI16AO2-M-VV	2 (Leap number)	Full Duplex (Fixed)	3/6/12 Mbps Selectable
HSL-AI16AO2-M-AV			
HSL-AO4-U			

Observe the motion control module rule

Module	Slave Index Occupation	Transmission Mode	Transmission Speed
HSL-4XMO-CG-N/P	4 (Leap) / 4(Consecutive)	Full Duplex / Half Duplex	3/6/12 Mbps Selectable
HSL-4XMO-CD-N/P			

5. Special rules

   ▷ If you will install only one HSL-AI16AO2-M-VV or HSL-AI16AO2-M-AV and the DIP switch is set to 1 (HSL-AI16AO2-M-VV/AV only supports full duplex mode), the occupied indexes will be 1 and 3. You must assign the parameter “max_slave_No” of HSL_start(…) as 4 to ensure correct communication. You may ignore this rule when using the HSL_auto_start function.

   ▷ If you will install only one HSL-4XMO-CG-N/P or HSL-4XMO-CD-N/P and the DIP switch is set as to 1 and full duplex mode, the occupied indexes will be 1, 3, 5, and 7. You must assign the parameter “max_slave_No” of HSL_start(…) as 8 to ensure correct communication. You may ignore this rule when using the HSL_auto_start function. In half duplex mode, these modules occupy 1, 2, 3 and 4. Therefore, you must assign the “max_slave_No” of HSL_start(…) as 4 or call the HSL_auto_start function.

3.4.2 Examples

The following examples are provided for user reference. All modules used are set in full duplex mode.

Example 1

Provided you installed two HSL-DI16DO16, two HSL-DI32-M-N, and an HSL-AI16AO2-VV with all slave modules in full duplex mode, you can have two conditions as follows:

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

We recommended that you use the provided 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 #1	6	6
HSL-DI16DO16 #2	8	8

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

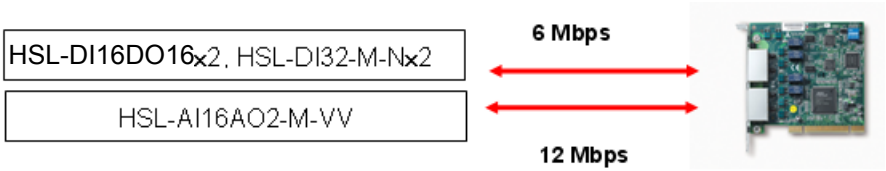
Condition 2: HSL-AI16AO2-VV operates at 12 Mbps.

We recommended that you use the provided 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 #1	5	5
HSL-DI16DO16 #2	6	6

Another example of a compact composition. The scan time needs 30.33 µs x 6 at 6 Mbps, full duplex mode. You may connect these modules with one master controller. The HSL-AI16AO2-M-VV module connects to another master controller. The DIP switch of

HSL-AI6AO2-M-VV is assigned as 1. Because of the special rule, users have to assign the “max_slave_No” of HSL_start(…) as 3 or call HSL_auto_start by connect_index #1. The illustration below explains this.



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

Example 2

Provided you installed two HSL-DI16DO16-UJ, one HSL-DI16DO16-M-NN, two HSL-DO32-M-N, one HSL-AI16AO2-VV, and two HSL-4XMO-CG-N with all slave modules in full duplex mode, you can have the following conditions:

Condition 1: The HSL-AI16AO2-VV and two HSL-4XMO-CG-N operate in 6 Mbps.

We recommended that you use the provided slave index configuration.

Item	DIP Switch	Index Occupation in HSL
HSL-4XMO-CG-N #1	1	1, 3, 5, 7
HSL-4XMO-CG-N #2	2	2, 4, 6, 8
HSL-DO32-M-N #1	9	9, 10
HSL-DO32-M-N #2	11	11, 12
HSL-AI16AO2M-VV	13	13, 15
HSL-DI16DO16-UJ #1	14	14
HSL-DI16DO16-UJ #2	16	16
HSL-DI16DO16-M-NN	17	17

The scan time needs 30.33µ x 17 at 6 Mbps, full duplex mode. You can connect these modules with one master controller.

Condition 2: An HSL-AI16AO2-VV and two HSL-4XMO-CG-N modules operate at 12 Mbps.

We recommended that you use the provided slave index configuration.

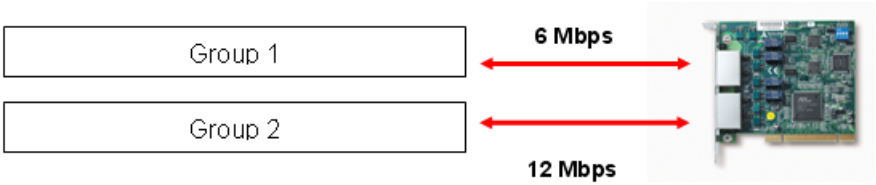
Group 1	DIP Switch	Index Occupation in HSL
HSL-DO32-M-N #1	1	1, 2
HSL-DO32-M-N #2	3	3, 4
HSL-DI16DO16-UJ #1	5	5
HSL-DI16DO16-UJ #2	6	6
HSL-DI16DO16-M-NN	7	7

The scan time needs 30.33µs x 7. You may connect these modules with one master controller. The HSL-AI16AO2-M-VV

and two HSL-4XMO-CG-N modules connect to another master controller. The management table below is for reference.

Group 2	DIP Switch	Index Occupation in HSL
HSL-4XMO-CG-N #1	1	1, 3, 5, 7
HSL-4XMO-CG-N #2	2	2, 4, 6, 8
HSL-AI16AO2-M-VV	9	9, 11

Refer to the illustration below.



The cycle time of the first master controller is 30.33μs x 7, while the cycle time of second master controller is 15.17μs x 11 at 12 Mbps, full duplex mode.

4 HSL LinkMaster Utility

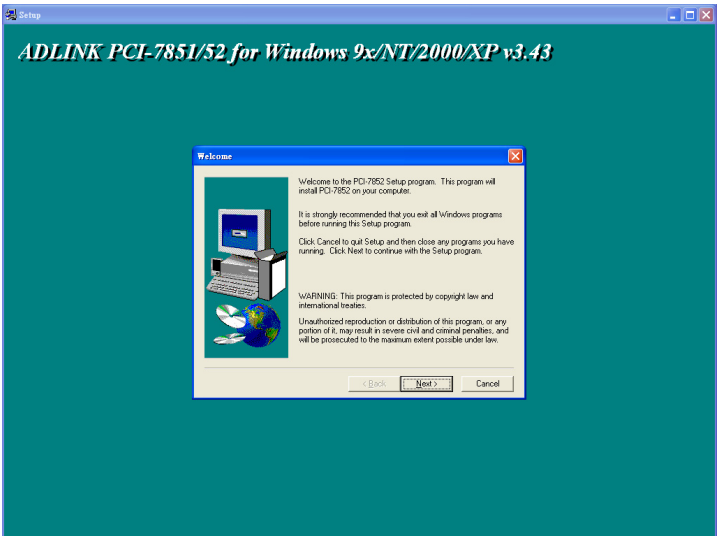
After installing the master controller and slave modules, you are now ready to install the HSL driver and the LinkMaster utility for system testing and debugging. This utility features a user-friendly interface that enables you to easily test I/O statuses, including read/write the I/O data, calibration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

4.1 Software Installation

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

To install the HSL drivers:

1. Locate, then double-click the SETUP.exe file from the All-in-One CD. The installation window appears. Click Next.

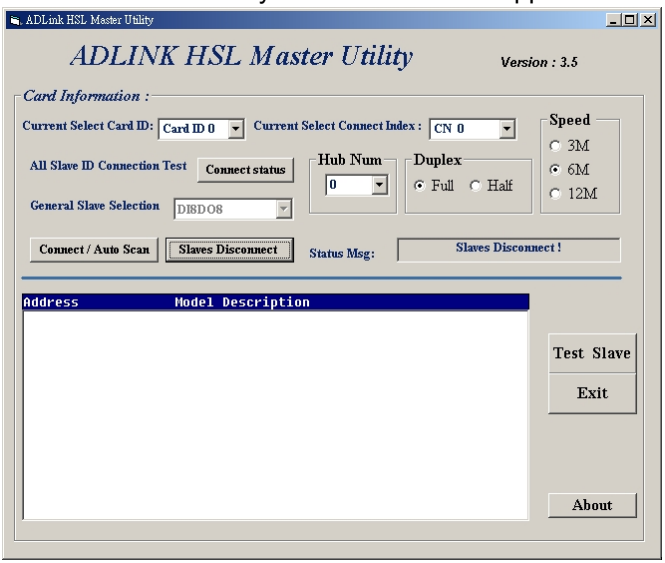


2. Follow screen instruction to install.
3. Restart the system when the installation process is completed.

4.2 ADLINK HSL LinkMaster Utility

4.2.1 Launching the LinkMaster Utility

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

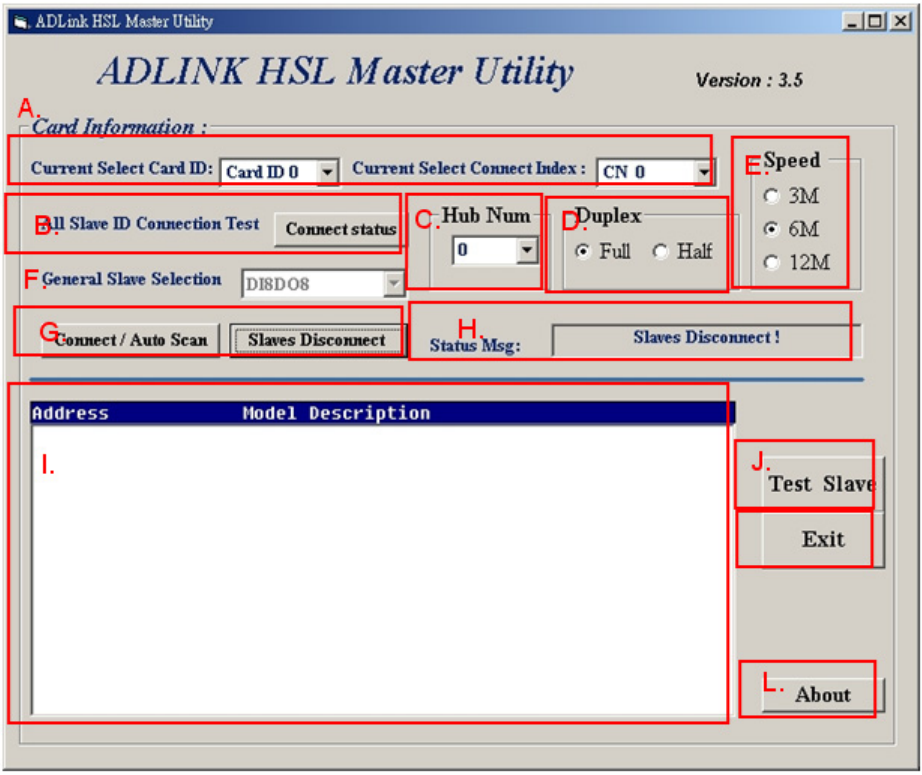


4.2.2 Before you proceed

1. LinkMaster is a testing and debugging program based on VB 6.0 and is only available for Windows® 98/NT/2000/XP 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 control modules. For motion control utility and manipulation, refer to the HSL-4XMO user's manual.

4.2.3 LinkMaster Utility Introduction

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

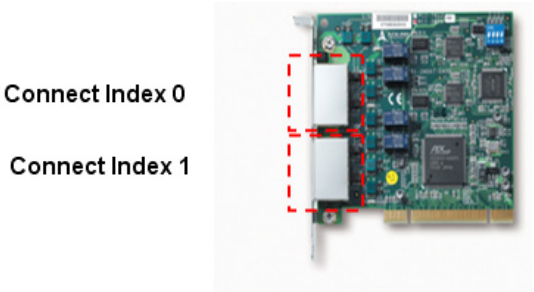


- A. Select card
- B. Network quality test
- C. Set hub number (Only for 7853/54)
- D. Set duplex mode (Only for 7853/54)
- E. Set speed mode (Only for 7853/54)
- F. General slave selection
- G. Auto scan slave modules
- H. Show software information
- I. Show module information

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

Below are descriptions of the main interface buttons.

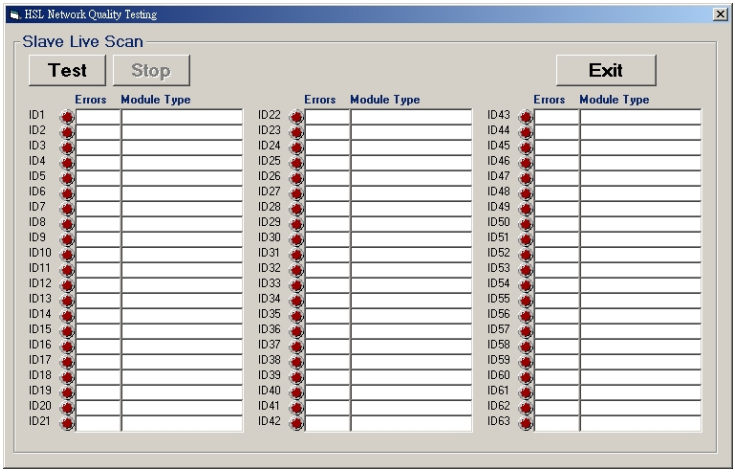
1. **Current Select Card ID**. When LinkMaster is activated, it searches all HSL master control cards installed in the system, such as PCI-7853, PCI-7854 and PMC-7852/G. Every card shows its index (ID) ranging from 0 to?. You can use this function to specify which card you want to operate.
2. **Current Select Connect Index**. For cards with two master controllers such as PCI-7854 and PMC-7852/G, the connect index ranges from 0 to 1. For single master controller such as PCI-7853, the connect index is 0. Refer to the diagram below.



PCI-7854

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.

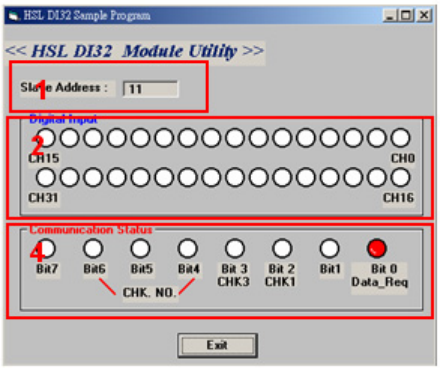
   Test Slave: While all connected slave modules list in 9th block, you can use this function to activate the testing dialog. For example, when you connect the HSL-DI16DO16-M-NN, you will see this module from the screen. Clicking on it will show a window from where you can test and debug the modules.
7.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

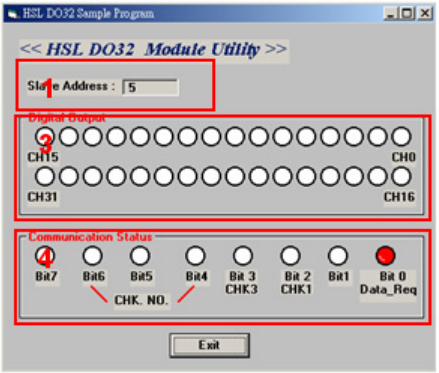
The succeeding sections outline the usage of the slave module utility.

4.2.4 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.

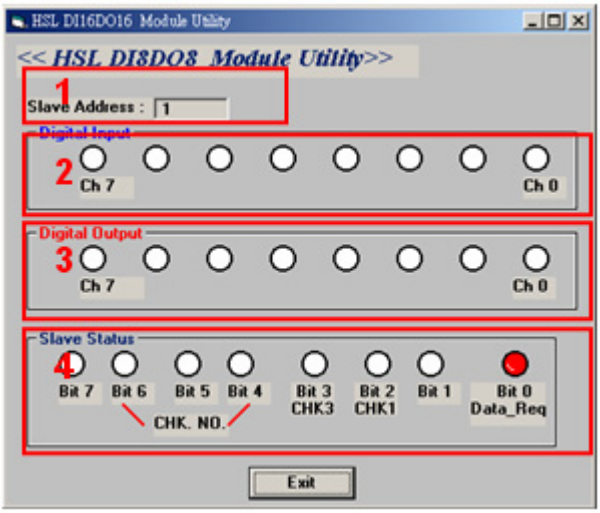
4.2.5 HSL-DI32 and HSL-DO32 Utility





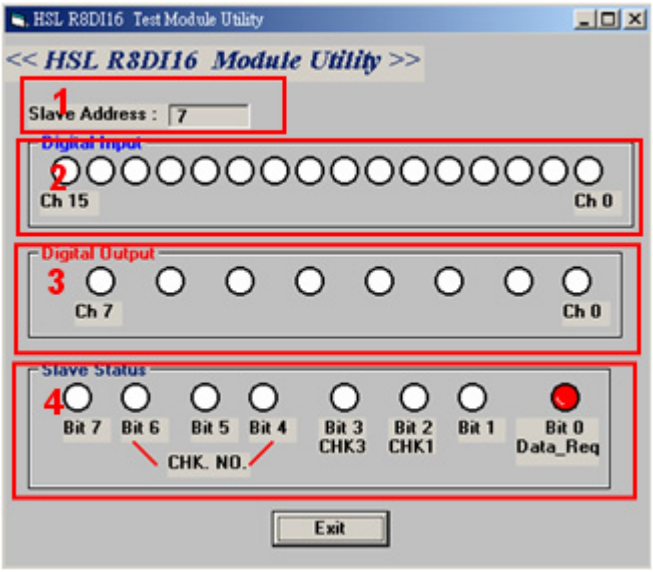
1. **Slave Address**. Shows the slave index occupied by the module. These modules occupy two slave indexes starting from an odd number. For example, when you adjust the DIP switch to 3, the modules are assigned indexes 3 and 5.
2. **Digital Input**. A white circle indicates no digital input; a red icon 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.

4.2.6 HSL-DI8/HSL-DO8/HSL-DI4DO4 Utility



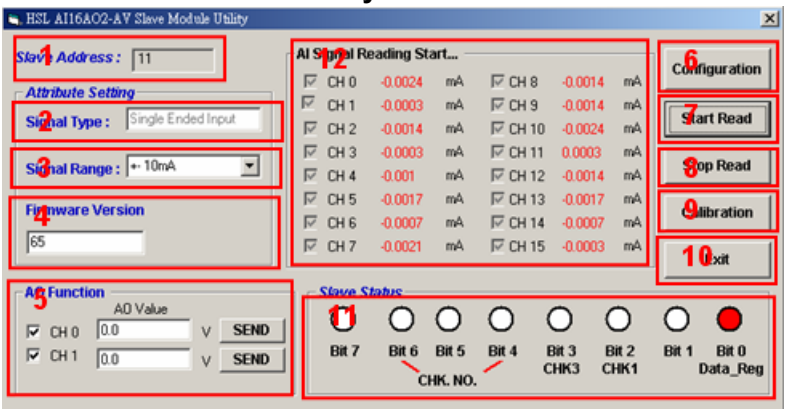
1. **Slave Address**. Shows the slave index occupied by the module. These modules occupy only one slave index.
2. **Digital Input**. A white circle indicates no digital input; a red icon indicates that the digital input is not activated.
3. **Digital Output**. Click on the icon to activate the digital 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.

4.2.7 HSL-R8DI16 Utility



1. **Slave Address**. Shows the slave index occupied by the module. These modules occupy only one slave index.
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 the icon to activate digital output. This function turns the relay ON or OFF. A red circle indicates that the relay is 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.

4.2.8 HSL-AI16AO2 Utility



1. **Slave Address**. Shows the slave index occupied by the module. The module occupies two consecutive indexes. For example, when you adjust the DIP switch to 4, the module obtains slave indexes 4 and 6.
2. **Signal Type**. Indicates the module's signal type.
3. **Signal Range**. Allows selection of the signal range. The utility offers four ranges including ±10V, ±5V, ±2.5V and ±1.25V for HSL-AI16AO2-M-VV. For HSL-AI16AO2-M-AV, the signal ranges are 20 mA, 10 mA, and 5 mA.
4. **Firmware Version**. Shows the latest firmware version.
5. **AO Function**. Key in the analog output value in the text box, then press SEND to trigger the AO. The range is ±10 V.
6. **Configuration**. Allows you to check if the signal range is correct before clicking on the Start Read button. The Configuration button allows you to save the information and complete the configuration task.
7. **Start Read**. Enables the A/D conversion task to read back the analog input values. The values are shown in the 12th block.
8. **Stop Read**. Disables the A/D conversion task.

9. **Calibration**. Calibrates the module. The modules are shipped with correct calibration. Refer to Appendix C if you want to recalibrate the module.
10. **Exit**. Closes the utility.
11. **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.

4.2.9 HSL-4XMO Utility

Refer to the HSL-4XMO user's manual.

5 HSL Function Library

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

5.1 List of Functions

This section presents all the functions. The function prototypes and common data types are declared in HSL.h. 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 5.2

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 5.3

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 5.4

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 5.5

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

Pulse Stretcher Function (HSL-DI16-UL only), Section 5.6

Function Name	Description
HSL_D_set_di_latch_function	Set DI-ltech function for one channel
HSL_D_set_di_latch_functionA	Set DI-latch function for all channels
HSL_D_get_di_latch_function	Retrieve DI-latch function

5.2 Initialization and System Information

@ Name

HSL_initial – Master board initialization

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 (7853/54 only)

HSL_get_scan_condition – Get scanning conditions (7853/54 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 master card

@ Description

HSL_initial:

Initializes the hardware and software states of the HSL master card (PCI-7851/52 or PMC-7852/G). 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 (live or die).

HSL_ get_irq_channel:

This function is used to get IRQ assigned by system.

@ Syntax

C/C++ (DOS, Windows 98/NT/2K/XP)

```
I16 HSL_initial (U16 card_ID);
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 (Windows 98/NT/2K/XP)

```
HSL_initial (ByVal card_ID 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.

connect_index: For PCI-7851, the valid value is 0. For PCI-7852 and PMC-7852/G, the valid value is 0 or 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
```

5.3 Timer Control

@ Name

HSL_enable_timer_interrupt (7851/52 only) – Enable timer interrupt of master card

HSL_disable_timer_interrupt (7851/52 only) – Disable timer interrupt of master card

HSL_set_timer (7851/52 only)– Set the resolution of timer

HSL_set_int_timer (7853/54 only) – Set the timer parameters

HSL_set_int_timer_enable (7853/54 only) – Enable\Disable timer interrupt of master card(7853/54 only)

HSL_wait_timer_interrupt (7853/54 only) – Wait timer event

@ Description

HSL_enable_timer_interrupt (7851/52 only):

Enables the hardware timer interrupt of the master card.

HSL_disable_timer_interrupt (7851/52 only):

Disables the hardware timer interrupt of the master card.

HSL_set_timer (7851/52 only):

This function sets up Timer 1 and Timer 2. Timer 1 and Timer 2 are used as frequency dividers to generate a dedicated constant timer interrupt sampling rate. The highest timer interrupt sampling rate of the master card may not exceed 20 KHz on Windows NT platform because of system limitation. The following example is set at 6 Mbps:

If you want to have a sampling rate of 15 kHz, the function must be

```
HSL_set_timer (0, 20, 20)
```

If you want to have a sampling rate of 1.2 kHz, the function must be

```
HSL_set_timer (0, 100, 50)
```

The formula used is:

Transmission speed / (c1 x c2)

In addition, the values of c1 and c2 must be greater than 1. When c1=0 or c2=0, the timer interrupt stops.

HSL_set_int_timer (7853/54 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) = $\frac{48MHz}{256 \cdot (p1+1)}$

HSL_set_int_timer_enable (7853/54 only):

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

HSL_wait_timer_interrupt (7853/54 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 codes illustrate this 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
```

5.4 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 (7853/54 only) – Set DI renewal check type

HSL_D_set_int_renewal_bit (7853/54 only) – Set the data bits of DI renewal check for each DI slave module

HSL_D_set_int_control (7853/54 only) – Set DI interrupt enable or disable

HSL_D_wait_di_interrupt (7853/54 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 (7853/54 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)

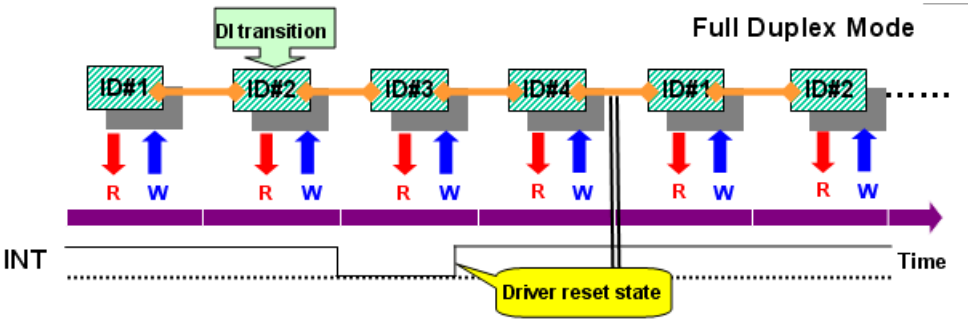


Figure 5-1: Type 1

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

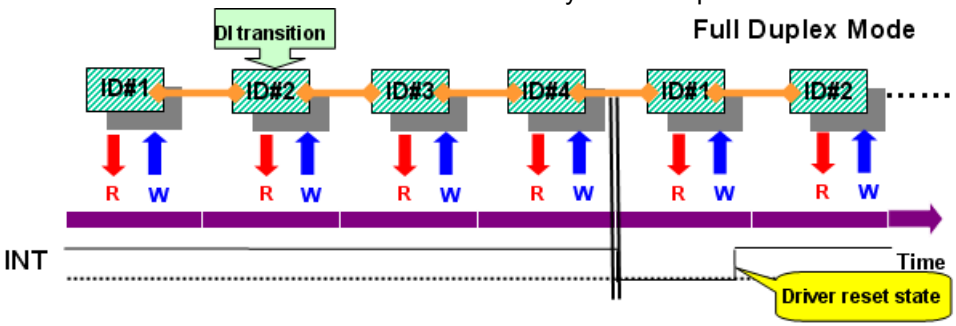


Figure 5-2: Type 2

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.

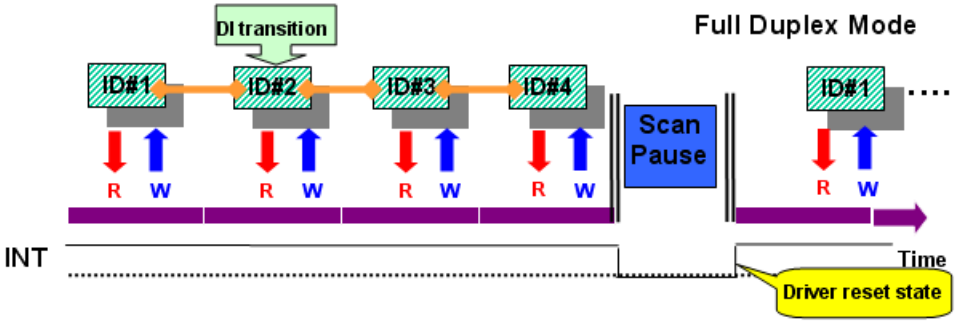


Figure 5-3: Type 3

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` (7853/54 only):

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

`HSL_D_set_int_control` (7853/54 only):

Enables or disables the DI interrupt.

`HSL_D_wait_di_interrupt` (7853/54 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 codes illustrate this 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++ (DOS, Windows 98/NT/2000/XP)

```
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 (Windows 98/NT/2000/XP)

```
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.

connect_index: For PCI-7851, the valid value is 0. For PCI-7852 and PMC-7852/G, the valid value is 0 or 1.

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-R8DI16: 0 to 15
- 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 hardward 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
```

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

```
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 (Windows 98/NT/2000/XP)

```
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.

connect_index: For PCI-7851, the valid value is 0. For PCI-7852 and PMC-7852/G, the valid value is 0 or 1.

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

For HSL-AI16AO2-M-AV

0: ± 5 mA

1: ± 10 mA

2: ± 20 mA

3: ± 20 mA

***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

For HSL-AI16AO2-M-AV

0: ± 5 mA

1: ± 10 mA

2: ± 20 mA

3: ± 20 mA

***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/AV

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 and mA for HSL-AI16AO2-M-AV 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
```

5.6 Pulse Stretcher Function (HSL-DI16-UL Only)

@ Name

HSL_D_set_di_latch_function – Set DI latch function for a specified DI channel

HSL_D_set_di_latch_functionA – Set DI latch function for all DI channels

HSL_D_get_di_latch_function – Get DI latch function for a specified DI channel

@ Description

HSL_D_set_di_latch_function:

The DI-latch function can be set for one single channel by this function. Note that when this function is executing, it will disable the Di input signal, and the Di state is unknown.

HSL_D_set_di_latch_functionA:

Set the same parameters of DI-latch function to all channels by this function. Note that when this function is executing, it will disable the Di input signal, and the Di state is unknown.

HSL_D_get_di_latch_function:

Retreive DI-latch settings. This function is used to confirm the setting which you set previously. Note that when this function is executing, it will disable the Di input signal, and the Di state is unknown.

@ Syntax

C/C++ (DOS, Windows 98/NT/2K/XP)

```
I16 HSL_D_set_di_latch_function(I16 card_ID, I16
      connect_index, I16 slave_No, I16 channel,
      I16 active_mode, I16 duration);
I16 HSL_D_set_di_latch_functionA(I16 card_ID, I16
      connect_index, I16 slave_No, I16
      active_mode, I16 duration);
```

```
I16 HSL_D_get_di_latch_function(I16 card_ID, I16
      connect_index, I16 slave_No, I16 channel,
      I16 * active_mode, I16 *duration);
```

Visual Basic (Windows 98/NT/2K/XP)

```
HSL_D_set_di_latch_function (ByVal card_ID As
      Integer, ByVal connect_index As Integer,
      ByVal slave_No As Integer, ByVal channel As
      Integer, ByVal active_mode As Integer, ByVal
      duration As Integer) As Integer
HSL_D_set_di_latch_functionA (ByVal card_ID As
      Integer, ByVal connect_index As Integer,
      ByVal slave_No As Integer, ByVal active_mode
      As Integer, ByVal duration As Integer) As
      Integer
HSL_D_get_di_latch_function (ByVal card_ID As
      Integer, ByVal connect_index As Integer,
      ByVal slave_No As Integer, ByVal channel As
      Integer, active_mode As Integer, duration 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.

connect_index: For the PCI-7851, the valid value is 0. For the PCI-7852 and PMC-7852/G, the valid value is 0 or 1.

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

channel: Specifiy the DI channel. The valid value is from 0 to 15.

active_mode: Latch the DI signal

0: active ON

1: active OFF

duration: Latch time, unit: millisecond. The valid value is from 0 to 127

@ Return Code

```
ERR_Invalid_Board_Number
ERR_Connect_Index
ERR_Satellite_Number
ERR_Board_No_Init
ERR_Channel_Number
ERR_Slave_Number
ERR_Set_Di_Latch_Failed
ERR_Di_Latch_time
ERR_No_Error
```

6 How to Program with HSL Function Library

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

6.1 Programming with HSL DLL

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

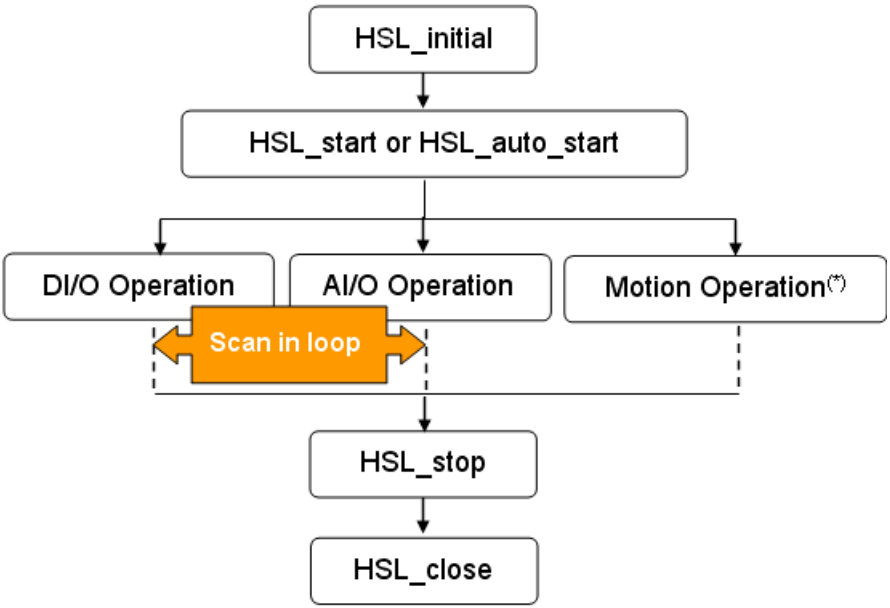


Figure 6-1: Programming Flow

6.1.1 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.

6.1.2 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.

6.1.3 Motion Operation:

Refer to HSL-4XMO user's manual.

Appendix A Scan Time Table

A.1 Full Duplex Mode

Slave Index Number	Cycle Time under 3 Mbps	Cycle Time under 6 Mbps	Cycle Time under 12 Mbps
Base Unit	60.67 µs	30.33 µs	15.17 µs
< 3(*)	182.00µs	91.00 µs	45.50 µs
5	303.33 µs	151.67 µs	75.83 µs
10	606.67µs	303.33 µs	151.67 µs
20	1.213 ms	606.67 µs	303.33 µs
30	1.820 ms	910.00 µs	455.00 µs
40	2.427 ms	1.213 ms	606.67 µs
50	3.033 ms	1.516 ms	758.33 µs
60	3.640 ms	1.820 ms	910.00 µs
63	3.822 ms	1.911 ms	955.50 µs

(*) The minimum scan time for full duplex mode at different transmission speeds.

A.2 Half Duplex Mode

Slave Index Number	Cycle Time under 3 Mbps	Cycle Time under 6 Mbps	Cycle Time under 12 Mbps
Base Unit	118 µs	59 µs	29.5 µs
< 3(*)	354 µs	177 µs	88.5 µs
5	590 µs	295 µs	147.5 µs
10	1.180 ms	590 µs	295 µs
20	2.360 ms	1.180 ms	590 µs
30	3.540 ms	1.770 ms	885 µs
40	4.720 ms	2.360 ms	1.180 ms
50	5.900 ms	2.950 ms	1.475 ms
60	7.080 ms	3.540 ms	1.770 ms
63	7.434 ms	3.717 ms	1.859 ms

(*) The minimum scan time for half duplex mode at different transmission speeds.

Appendix B Mapping Table

HSL has two types of function library in the HSL.h. The following is the mapping table for new and old versions of the codes.

B.1 Initialization and System Information

New Version	Old Version
HSL_initial	W_HSL_Initial
HSL_close	W_HSL_Close
HSL_start	W_HSL_Start
HSL_auto_start	W_HSL_Auto_Start
HSL_stop	W_HSL_Stop
HSL_connect_status	W_HSL_Connect_Status
HSL_slave_live	W_HSL_Slave_Live
HSL_get_irq_channel	W_HSL_Get_IRQ_Channel

B.2 Timer Control 3

New Version	Old Version
HSL_enable_timer_interrupt	W_HSL_TMRINT_Enable
HSL_disable_timer_interrupt	W_HSL_TMRINT_Disable
HSL_set_timer	W_HSL_Timer_Set

B.3 Discrete I/O

New Version	Old Version
HSL_D_read_input	W_HSL_DIO_In
HSL_D_read_channel_input	W_HSL_DIO_Channel_In
HSL_D_write_output	W_HSL_DIO_Out
HSL_D_write_channel_output	W_HSL_DIO_Channel_Out
HSL_D_read_ouput	W_HSL_Read_DIO_Out
HSL_D_read_all_slave_input	W_HSL_DIO_Memory_In
HSL_D_write_all_slave_output	W_HSL_DIO_Memory_Out
HSL_D_set_input_logic	W_HSL_Set_In_Out_Logic
HSL_D_set_output_logic	

B.4 Analog I/O

New Version	Old Version
HSL_A_start_read	W_HSL_AI_Start_Read
HSL_A_stop_read	W_HSL_AI_Stop_Read
HSL_A_set_signal_range	W_HSL_AI_SetConfig
HSL_A_get_signal_range	W_HSL_AI_GetConfig
HSL_A_get_input_mode	
HSL_A_set_last_channel	W_HSL_AI_Set_Last_Channel
HSL_A_get_last_channel	W_HSL_AI_Get_Last_Channel
HSL_A_read_input	W_HSL_AI_Channel_In
HSL_A_write_output	W_HSL_AO_Channel_Out
HSL_A_read_output	W_HSL_AO_Channel_In
HSL_A_sync_rw	W_HSL_AIO_Channel_InOut
HSL_A_get_version	W_HSL_AI_Get_Version

Appendix C HSL-AI16AO2 Calibration

C.1 Before you proceed

Before calibrating the HSL-AI16AO2-M-VV and HSL-AI16AO2-M-AV, take note of the following:

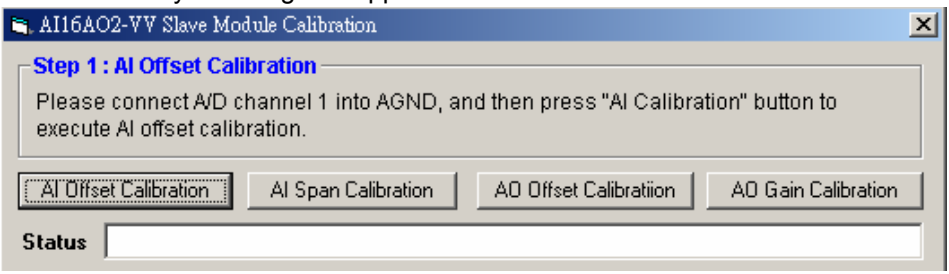
1. Make sure that the signal type is single-ended. You may set this via the jumper.
2. Use a precise calibrator that can generate a precise 5 V.
3. Check the status text to know if the calibration is successful or not.
4. Refer to the analog input field configuration below.

Single-ended mode																
Terminal No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Signal Name	AI0	AI1	AI2	AI3	AI4	AI5	AI6	AI7	AI8	AI9	AI10	AI11	AI12	AI13	AI14	AI15
Terminal No.	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Signal Name	AO0	AO1	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND	AGND

C.2 Calibrating the modules

To calibrate the modules:

1. Press the Calibration button from the HSL-AI16AO2 utility. A dialog box appears.



2. Connect AI channel 1 to AGND. The AI channel index is from 0 to 15. Take note of the index. After wiring, press the AI Offset Calibration button.
3. Connect AI channel 0 to the calibrator, then, press the AI Span Calibration button.
4. Connect AI channel 12 to AO channel 0, and AI channel 14 to AO channel 1, then press the AO Offset Calibration button.
5. If the previous step is successful, press the AO Gain Calibration button to finish the calibration.

If calibration is successful, the module is ready for use. If not, check the wiring and calibrator, then repeat the calibration procedures.

Appendix D HSL-HUB/Repeater Information

D.1 Recommended transfer rates, total extension distance, and number of installed HSL-HUB/Repeater

Transmission rate	Number of inserted Hubs (Repeater)							
	Basic configuration	1	2	3	4	5	6	7
3Mbps	300 m	600 m	900 m	1.2 km	1.5 km	1.8 km	2.1 km	2.4 km
6Mbps	200 m	400 m	600 m	800 m	1 km	1.2 km	1.4 km	1.6 km
12Mbps	100 m	200 m	300 m	400 m	500 m	600 m	700 m	800 m

D.2 Scan time table

D.2.1 Full duplex/12 Mbps

Number of inserted Hubs (Repeater)	Slave Index Number		
	3 (Min.)	30	63 (max)
Basic configuration (0)	45.50 us	455.00 us	955.50 us
1	82.00 us	820.00 us	1722.00 us
2	118.00 us	1180.00 us	2478.00 us
3	154.00 us	1540.00 us	3234.00 us
4	190.00 us	1900.00 us	3990.00 us
5	226.00 us	2260.00 us	4746.00 us
6	262.00 us	2620.00 us	5502.00 us
7	298.00 us	2980.00 us	6258.00 us

D.2.2 Full duplex/6 Mbps

Number of inserted Hubs (Repeater)	Slave Index Number		
	3 (Min.)	30	63 (max)
Basic configuration (0)	91.00 us	910.00 us	1911.00 us
1	164.00 us	1640.00 us	3444.00 us
2	236.00 us	2360.00 us	4956.00 us
3	308.00 us	3080.00 us	6468.00 us
4	380.00 us	3800.00 us	7980.00 us
5	452.00 us	4520.00 us	9492.00 us
6	524.00 us	5240.00 us	11004.00 us
7	596.00 us	5960.00 us	12516.00 us

D.2.3 Full duplex/3 Mbps

Number of inserted Hubs (Repeater)	Slave Index Number		
	3 (Min.)	30	63 (max)
Basic configuration (0)	182.00 us	1820.00 us	3822.00 us
1	328.00 us	3280.00 us	6888.00 us
2	472.00 us	4720.00 us	9912.00 us
3	616.00 us	6160.00 us	12936.00 us
4	760.00 us	7600.00 us	15960.00 us
5	904.00 us	9040.00 us	18984.00 us
6	1048.00 us	10480.00 us	22008.00 us
7	1192.00 us	11920.00 us	25032.00 us

D.2.4 Half duplex/12 Mbps

Number of inserted Hubs (Repeater)	Slave Index Number		
	1 (Min.)	30	63 (max)
Basic configuration (0)	29.50 us	885.00 us	1858.50 us
1	39.33 us	1180.00 us	2478.00 us
2	51.33 us	1540.00 us	3234.00 us
3	63.33 us	1900.00 us	3990.00 us
4	75.33 us	2260.00 us	4746.00 us
5	87.33 us	2620.00 us	5502.00 us
6	99.33 us	2980.00 us	6258.00 us
7	111.33 us	3340.00 us	7014.00 us

D.2.5 Half duplex/6 Mbps

Number of inserted Hubs (Repeater)	Slave Index Number		
	1 (Min.)	30	63 (max)
Basic configuration (0)	59.00 us	1770.00 us	3717.00 us
1	78.67 us	2360.00 us	4956.00 us
2	102.67 us	3080.00 us	6468.00 us
3	126.67 us	3800.00 us	7980.00 us
4	150.67 us	4520.00 us	9492.00 us
5	174.67 us	5240.00 us	11004.00 us
6	198.67 us	5960.00 us	12516.00 us
7	222.67 us	6680.00 us	14028.00 us

Warranty Policy

Thank you for choosing ADLINK. To understand your rights and enjoy all the after-sales services we offer, please read the following carefully.

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

3. Our repair service is not covered by ADLINK's guarantee in the following situations:
   - Damage caused by not following instructions in the User's Manual.
   - Damage caused by carelessness on the user's part during product transportation.
   - Damage caused by fire, earthquakes, floods, lightening, pollution, other acts of God, and/or incorrect usage of voltage transformers.
   - Damage caused by unsuitable storage environments (i.e. high temperatures, high humidity, or volatile chemicals).
   - Damage caused by leakage of battery fluid during or after change of batteries by customer/user.
   - Damage from improper repair by unauthorized ADLINK technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - This warranty is not transferable or extendible.
   - Other categories not protected under our warranty.
4. Customers are responsible for shipping costs to transport damaged products to our company or sales office.
5. To ensure the speed and quality of product repair, please download an RMA application form from our company website: http://rma.adlinktech.com/policy. Damaged products with attached RMA forms receive priority.

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