ADLINK TECHNOLOGY INC.

High Speed Link System

Master-Slave

Distributed Solutions

User's Manual

Manual Rev. 2.01

Revision Date: April 7, 2005

Part No: 50-12100-2010

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

Microsoft®, Windows NT®, Windows 98®, Windows 2000®, Windows XP® are registered trademarks of Microsoft Corporation. Borland C++ Builder® is a registered trademark of Borland International, 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):

ADLINK TECHNOLOGY INC.

Table of Contents

List of Tables

List of Figures

1 HSL Introduction

Please note before reading:

Master board: HSL is a master-slave communication system. The control board in the host side is called the master board.

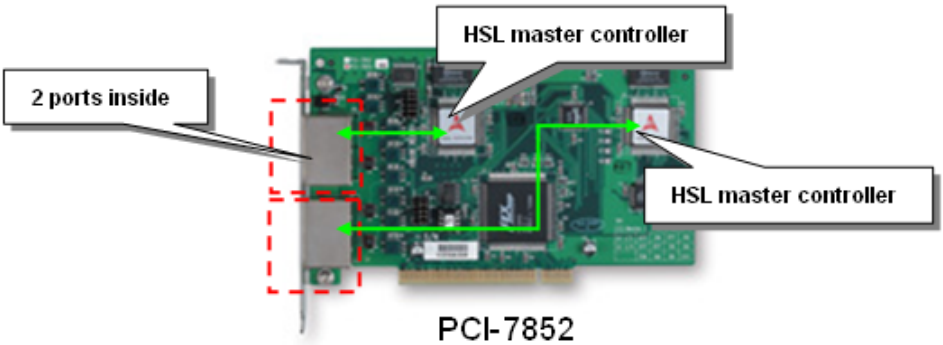
Slave module: HSL is a master-slave communication system. From the remote side, the slave module can connect a variety of sensors.

Slave index: The basic unit in a HSL system is called the slave index. One HSL slave module may occupy 1, 2, or 4 slave indexes, depending on slave module type.

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

Half duplex: Data transmission and receiving occurs at consecutive scanning time.

HSL master controller: One HSL ASIC plays the role of the master controller. For example, the PCI-7851 has one on-board HSL ASIC which can connect up to 63 slave indexes. Two ports are provided for convenient connection. The same rule applies to the PCI-7852; it can connect up to 126 slave indexes and provides four ports.



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

1.1 What Is an HSL System

HSL is an innovative distributed I/O technology which allows thousands of I/O points to be scanned in millisecond-level real time by using a mater-slave architecture. The HSL master board has two form factors: PCI and PMC. PMC boards can be used in embedded controllers. By using a commercial Ethernet cable with RJ45 connector, users can easily set up HSL slave modules as close as possible to sensor devices. Thus the wiring effort is dramatically reduced. Besides I/O modules, ADLINK also provides remote motion control modules with 4-axis pulse train type. With motion control, a variety of machine makers can benefit from HSL networks because it integrates discrete I/O, analog I/O, thermocouple module, and motion control. This local network features rapid response, real-time scanning and multiple-axis control. With PMC module, users can also make the integration with embedded solution platform. HSL solutions are ready to use.

HSL solutions are for those who want:

- Distributed solutions based on the PC architecture or embedded platforms
- Easy wiring solutions for remote I/O, including discrete I/O and analog I/O modules
- Low-profile discrete I/O modules for fitting into limited space
- A vast (hundreds or more) number of discrete I/O points
- Real-time and fast scanning
- High speed data acquisition
- Remote motion control of up to 120 axes with two HSL master controllers of a master board.
- Motion control features point table management and motion script download to enhance execution efficiency

Product Overview

The following drawing shows the basic topology of the HSL system.

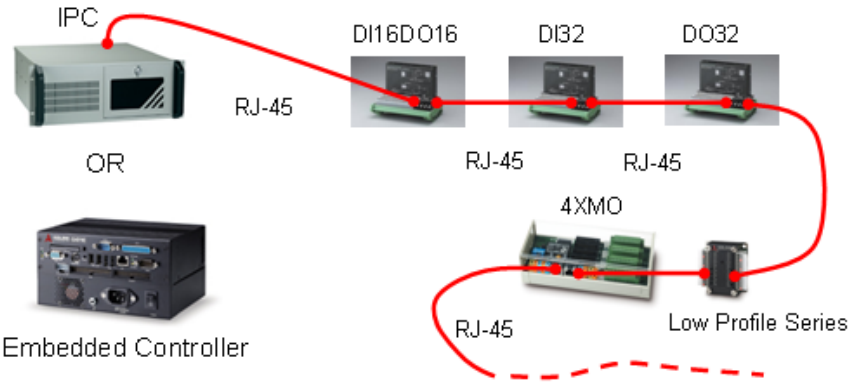


Figure 1-1: HSL topology

HSL System Features

High speed

It takes at most 1.895ms for a HSL master to scan all the discrete I/O points of slave modules under 6Mbps. Take a distributed control system with 63 slave I/O modules of type HSL-DI16DO16-DB-NN as an example, this slave I/O module supports 16DI and 16DO, and these 2016 discrete I/O points could be scanned (or updated) within 1.895ms. In other words, the scanning speed is as high as 1000 points per ms.

Real time

The time period for a HSL master controller to scan all slave I/O modules is deterministic. The total scanning cycle time is exactly proportional to the number of slave indexes. At 6Mbps, every 30.4μs is added for one more slave index. For a HSL system with 30 discrete I/O slave modules (every discrete I/O module occupies one slave index), the scanning time period is precisely 30 X

30.4μs = 0.912 ms. The scan time unit based on transmission rate is as follows.

	3 Mbps	6 Mbps	12 Mbps
Full Duplex	60.7μs	30.4μs	15.2μs
Half Duplex	118μs	59μs	29.5μs

Easy wiring

The connection between the HSL master controller and all slave I/O modules requires only commercial Ethernet cables, which dramatically reduce the wiring effort. With just Ethernet cables, hundreds even thousands of I/O data can transmit between the HSL master and slave I/O modules. This is absolutely the easiest and most cost effective wiring solution. For low profile series, users make the connection by direct wiring.

Large number of I/O points

PCI-7851 offers one HSL master controller and PCI (PMC)-7852 offers two HSL master controllers. For maximum installation, users can have 8 PCI -7851 and PCI-7852 in one PC system. That means users can have 1512 slave indexes in HSL network system. If choosing all connected modules as HSL-DI16DO16-DB-NN, the total numbers of I/O points are 24,192 DI and 24,192 DO at maximum. For embedded solutions, users can select the PMC-7852/G.

Easy I/O expansion

For centralized configuration, expanding I/O points requires more I/O boards and free PCI or ISA slots. What happens if there are still I/O points needed when no more free slot exists? Unlike centralized configurations, the distributed I/O configuration lifts the constraint of centralized I/O. All you need is to add one more slave I/O module and an Ethernet cable with RJ45 for communication link.

Self-diagnostic function

Once powering on, the network status will be monitored continuously, and a status register will keep the accumulated slave-no-response count for every individual slave I/O module. Additionally,

CRC12 is another proven method to eliminate communication error.

Modular design of the slave I/O

ADLINK offers a variety of slave module types for users. M series have the metal case; DB series are without metal case and L series are low-profile design for compact size use. For M and DB series, users also need terminal board for connection. Terminal boards act as a carrier of salve I/O module with wiring function. The Ethernet connector and screw terminal on terminal boards make it easier for users to replace I/O module without power off and wire off when necessary.

Remote motion control

Besides slave I/O modules, ADLINK also offers remote motion control solutions based on the HSL network. HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P are currently available and can connect up to 4 axes. HSL-4XMO-CG-N/P has general type interfaces, especially for those who use stepper or linear motors. HSL-4XMO-CD-N/P has D-sub interface. By using transfer cables, users can connect to specific servo amplifier. Consequently, no matter what you choose, you can easily make a distributed control application and include discrete I/O, analog I/O, and remote motion control.

Easy to program

Every HSL master card is equipped with 32K SRAM; the 32K SRAM carries all the I/O status information of this HSL system. The ASIC on every HSL master board takes charge of communication with all remote slave I/O modules at fixed scanning period and keeps the most updated I/O status information to the SRAM. What users do is to read and write the data in the 32K SRAM on HSL master card through PCI (PMC) bus.

HSL Applications

HSL as a Distributed PLC

In the past decades, PLC had taken an important character in the field of industry automation. With the help of communication modules, such as RS232 and RS485, PLC can also perform distributed control. The traditional architecture of distributed PLC applications is shown below, in which the MPC (Monitoring PC) takes the character of medium for data transmission from field to MIS.

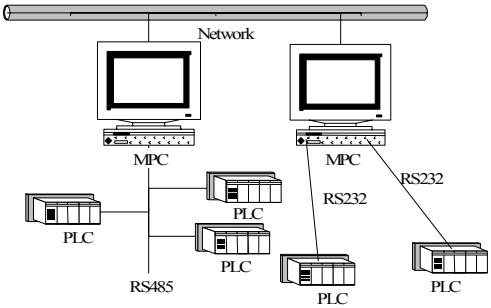


Figure 1-2: Traditional Architecture of Distributed PLC

Then, with the developing of communication technology and popularization of networking, networking modules such as Ethernet became available. This improvement evolved the following architecture. The medium character of MPC is replaced.

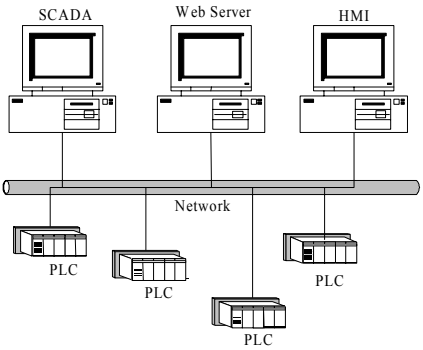


Figure 1-3: Networking PLC

PLCs are capable of network communicating, but it is usually very expansive. This is because PLCs are not an open architecture, only the hardware venders can make it.

With HSL, the distributed control architecture can become as the figure below. Users do not need one extra PC for Ethernet communication. Users can use one IPC to control the whole system.

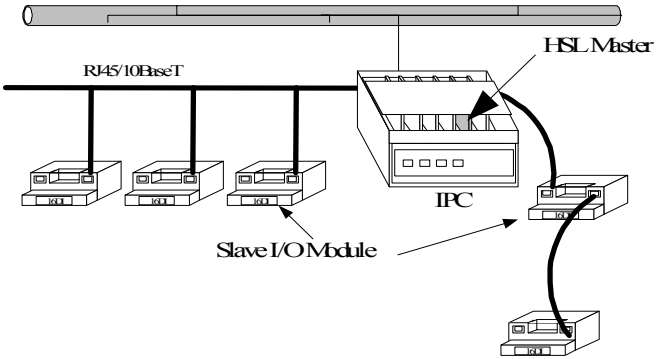


Figure 1-4: HSL as distributed PLC

Comparing the figures above:

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

HSL as a Remote Real-time DAQ

HSL systems provide many beneficial features as described above. Among those features, two are worthy of being particularly notified.

- High Speed
- Real time (time-deterministic)

To implement a DAQ application, the real-time characteristic is the most important issue. With HSL systems, all I/O data are time-deterministic refreshed. The sampling rate (or scan rate) is linearly dependent on the number of slave indexed occupied, ranges from

90μs (less than 3 slave indexes) to 2 ms (63 slave indexes) under 6Mbps. These two features with HSL's remoteness make HSL very suitable for remote DAQ applications, especially when real-time is concerned.

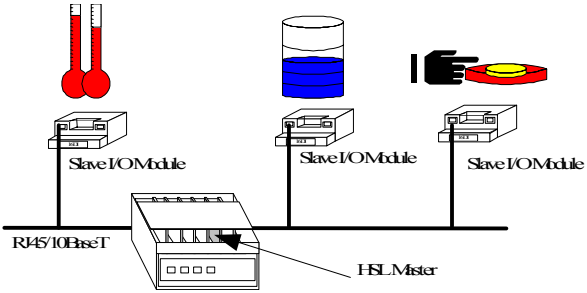


Figure 1-5: HSL as real-time DAQ

HSL for SCADA system

HSL is suitable for SCADA (supervisory control and data acquisition) systems. The characteristics of a HSL in a SCADA system are DAQ hardware and a HMI toolkit. Users still need some SCADA software to fully accomplish SCADA functionality. The reason why HSL is suitable for SCADA is described below:

- HSL is based on the PC's open architecture.
- HSL is able to support mass I/O points.
- HSL is capable of real-time remote DAQs.

The PC-based characteristic of HSL systems opens a highway to the "e" world. Users simply set up their own HMI software on the PC where HSL master is installed. All I/O data is collected in a constant period from slave I/O modules to master. Then, HMI could access these data through local PCI (PMC)-bus

.

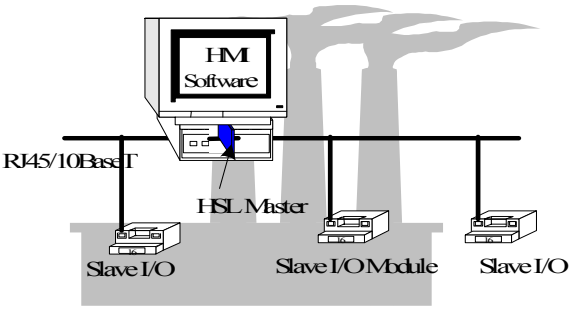


Figure 1-6: HSL for SCADA

1.2 HSL System Specifications

The detail specifications of HSL system are listed as following:

Platform:

Hardware platform: Industrial PC with PCI Bus/ Embedded SBC with PMC connector

Operating system platform: Windows 98/2K/NT/XP or Linux Redhat Distribution

Software Supported:

Windows: C library with DLL

Linux: Kernel 2.4.x

HSL Master Boards:

PCI -7851: Single HSL master controller board, two ports

PCI -7852: Dual HSL master controller board, four ports

PMC-7852/G: Dual HSL master controller board, four ports and PMC connector.

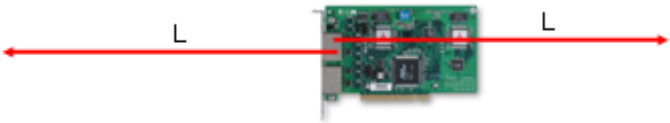
Remoteness:

One master controller has two ports. One port uses RJ-45 phone jack as connector.

One phone jack can drive 32 modules at maximum

One master controller can connect 63 slave indexes at maximum

Maximum wiring distance for each RJ-45 connector (one port): 200m@6Mbps (serial wiring from master to last slave module). The maximum length of port connection could be 400m@6Mbps because one side is 200m and the other side is also 200m.



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

Wiring:

Connector: RJ45 (on master controller and some of slave modules)

Cable: Cat-5 100 Base/TX Ethernet cable with shielding

Direct Wiring: for low-profile modules

Communication:

Multi-drop full-duplex RS-422 with transformer isolation scheme

Transmission speed: 3/6/12 Mbps, 6Mbps is the factory default setting.

I/O refresh rate: scan time unit × numbers of slave indexes (min: 3; max: 63)

	3 Mbps	6 Mbps	12 Mbps
Full Duplex	60.7µs	30.4µs	15.2µs
Half Duplex	118µs	59µs	29.5µs

Communication model: single master to multi-slave

Communication method: command/ response type handshaking

CRC12 and dedicate protocol for eliminating any communication error

1.3 HSL Series Products

HSL Master controller boards

Three models are provided:

- PCI-7851: HSL master controller card with PCI interface
- PCI-7852: Dual HSL master controller card with PCI interface
- PMC-7852/G: Dual HSL master controller card with PMC interface

At least one master controller card is necessary for HSL system. With PCI (PMC)-7852, two master controllers are available. There are at most 8 cards support for one computer system.

Slave I/O modules

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

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
TC	HSL-TC08			8(TC)	2	1~61	2
L	HSL-DI8-L-N/P	8				1~63	1
	HSL-DO8-L-N/P		8			1~63	1
	HSL-DI4DO4-L-N/P	4	4			1~63	1

Table 1-1: 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 range.

Remote motion control modules are also supported. Here are the lists:

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	4
	HSL-4XMO-CD-N/P	4	D-sub	1-57 for Full Duplex	

Table 1-2: 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 range.

Terminal Base

A variety of HSL terminal bases are available.

Model Numbers	Module Type Support	Module Number Support
HSL-TB64-DIN	All the HSL DB series	2
HSL-TB32-DIN	HSL-DI16DO16-DB-NN/PN and HSL-DO32-DB-N	1
HSL-TB32-U-DIN	All the HSL DB series	1
HSL-TB32-DO-DIN	HSL-DO32-DB-N	1
HSL-TB32-M-DIN	All the HSL M series	1

Table 1-3: Terminal Base

1.4 HSL Technical Information

HSL Technology Brief

In a HSL system, a single master controller can communicate with multiple slaves via a command-response method. The master controller sends commands to slave I?O modules for setting output values and requesting input information. Every slave module responds when receiving commands with its address ID. The response is either to set output according received values or to reply requested input information to master.

The following graph shows the working theory of HSL regarding how to **set output values**.

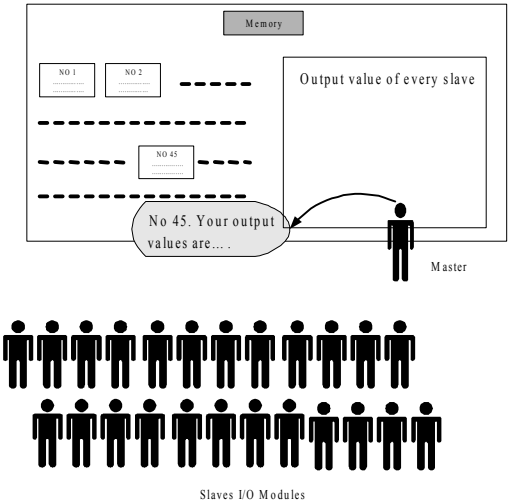


Figure 1-7: HSL Technology Brief -1

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

And, the following graph shows working theory of gathering input information.

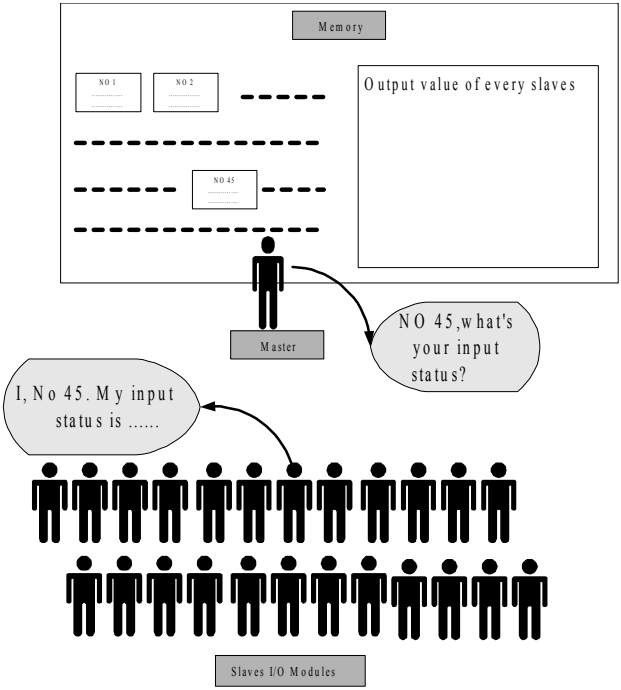


Figure 1-8: HSL Technology Brief-2

The single teacher (the master) sends the message “ID.#, what’s your newest input status” to all students, and the very student (with ID.#) gives his answer. Then, teacher writes the answer on the blackboard (RAM on master cards). If someone (user’s AP) is interested in a slave’s current input status, he just looks up the blackboard. All input information is presented there.

The two procedures above will take turn and repeat for every slave module. And after a cycle, every slave module sets its newest output status and master gathers every slave module’s newest input information in the memory. We simulate the polling communication cycle by the following personalized conversation of teacher and student:

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

Student No.1: My input status is ## (Teacher writes data on blackboard)

Teacher: No.2, your output vales are ##, what's your newest input status?

Student No.2: My input status is ## (Teacher writes data on blackboard)

(if 63 slave modules are equipped)

Teacher: No.63, your output vales are ##, what's your newest input status?

Student No.63: My input status is ## (Teacher writes data on blackboard)

(A polling cycle is completed)

(Repeat from NO1)

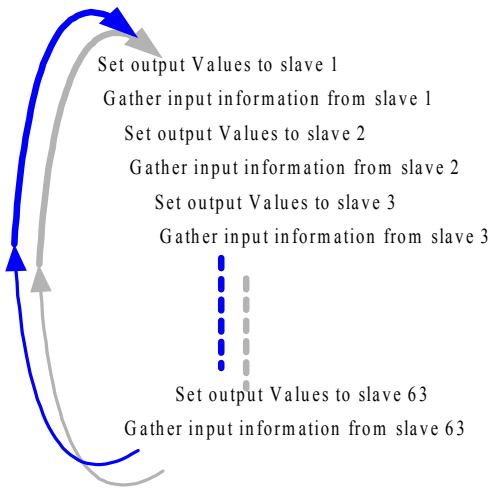
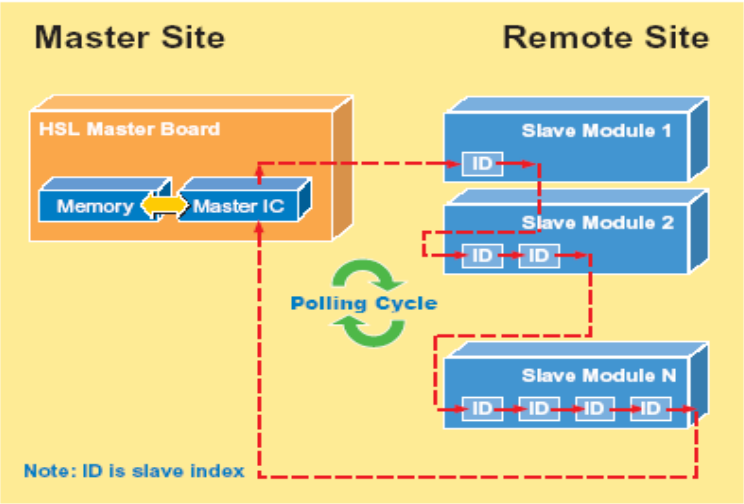


Figure 1-9: HSL I/O polling cycle

The architecture of HSL master-slave communication is as the following diagram:



HSL Terminology

In addition to input/output polling mechanism showed above, some syntax related to HSL should also be kept in mind.

HSL Master: Master is the “teacher” character in the figures above. The master takes charge of giving commands, including output value announcing and newest input status requesting.

Slave I/O Module: Slave I/O modules are the “student” characters in the figures above. Slave I/O modules are passive components in HSL system. They just receive commands from master and then response – replying newest input status or setting output values. The slave I/O module may take 1 or 2 address index depending on the I/O module type.

Polling Cycle: While communicating with slave I/O modules, the master takes turns to set output for and gather input from every slave module. If all slave modules are updated, we say that a polling cycle has completed. If only the master is working normally, the polling cycle will infinitely repeat.

I/O Refreshing Rate: The “I/O Refreshing Rate” is defined as the time needed to run a completed I/O updating cycle. Also, you can

think of it as the longest time needed for any digital I/O channel to get its newest status. The refreshing rate is decided linearly by total number of slaves used in individual HSL system, but no interference in any two HSL systems at the same PC or IPC.

Transmission Speed: This characteristic is quite ambiguous with I/O refreshing rate. But they are totally different matters. Data rate is referring to speed of digital signal transferred inside the wire cable. The unit of Transmission Speed is bps, but the unit of I/O refreshing rate is ms.

System Configurations

To establish a HSL application, users need to know how to configure all these HSL cards and slave I/O modules. Here are some concepts regarding configurations of HSL system. For detail information, please refer to individual chapter.

Master Card Index (card_ID): You may have more than one HSL master boards in IPC system. PCI BIOS will assign the card index for each HSL master board. Users need to give the card index for your programming. The card index is from 0.

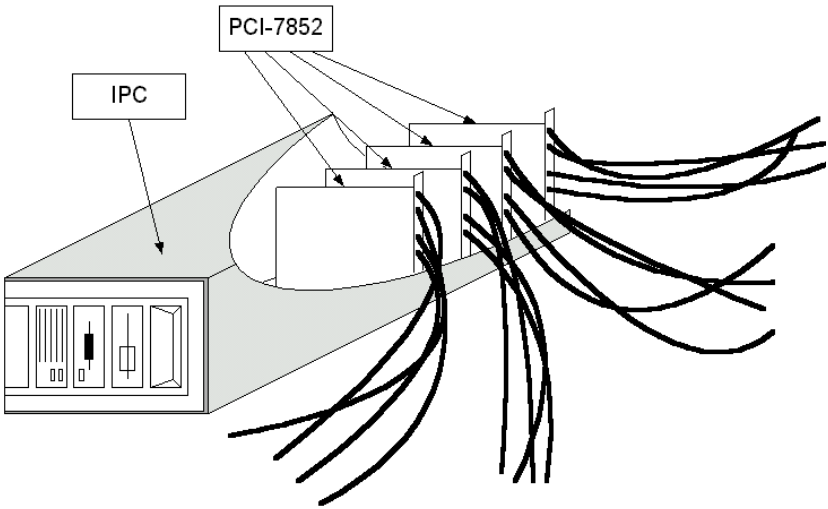


Figure 1-10: Multiple master cards in one IPC

Please refer to Chapter 2 “HSL Master Controller” for more information.

HSL Connect Index (connect_index): PCI-7851 provide purely one HSL master controller (connect_index = 0); PCI-7852 / PMC-7852/G provides two HSL master controllers (connect_index = 0 and 1). Connect index is used to distinguish the master controllers.

Ports: Port means the RJ-45 connector on HSL master board. A connector is a loop of wiring that starts from the master card and connects at most 32 slave I/O modules. There are two connectors (ports) in one HSL master controller, and these two connectors will carry the same signals sent from master.

Slave Index: A complete HSL system must be composed of one master and 1-63 slave indexes. In the following graph, a layout example of HSL system is presented.

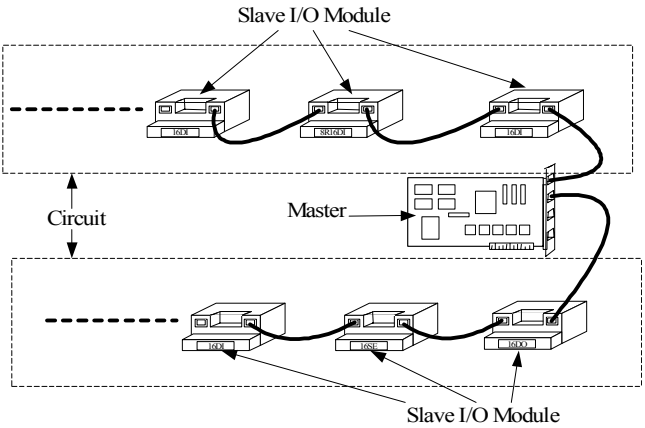


Figure 1-11: HSL system layout example-serial wiring

Each circuit from a master can equip at most 32 slaves. However, the maximum number of slaves is 63, not 64. This is because the slave address is decided by a 6-bit DIP-switch on I/O module’s termination board, and the value ‘0’ is reserved. All slave I/O modules at the same HSL set must be configured different slave index. The slave index is not necessary to be continuous from 1. How-

ever, continuous addressing will be more efficient. Actually, if some users set up a HSL system with only 2 slave modules addressed 1 & 63, the I/O refreshing rate is exactly the same as a 63-slave, addressed from 1 to 63 within HSL system.

Note: Please refer to Chapter 3 “HSL Slave I/O Module” for a complete description regarding how to set addresses of slave I/O modules.

Wiring

The HSL network follows RS422 electrical specifications, but not exactly the same.

Wiring

The wire cables of HSL system are carefully chosen to satisfy both easiness and standardization requirement without sacrificing communication quality. A 100BaseTX cable with RJ45 connectors is selected for HSL system.

Full-duplex RS-422 with Multi-drop

Typical RS-422 is not actually a networking specification. However, HSL does network application based on RS-422 with modification. The TXD of master is connected to RXD of every slave I/O modules. And all TXD of slaves are connected to RXD of master. Only master use TXD (of master) to RXD (of slave) channel, and by well designed, only one slave at one time sends message with TXD (of slave) to RXD (of master) channel. Consequently, there could be 63 slave I/O modules in a set of HSL network. This kind of networking solution is called RS422 with Multi-drop.

Ports

Every HSL master can support 2 ports segment. Inside the two ports, the TXD (of master) to RXD (of slave) channels are actually of the same signal sent by master, but the TXD (of slave) to RXD (of master) channels are not that case. TXD (of slave) to RXD (of master) channels are isolated from each circuit, and signals inside wouldn’t pass through master from one circuit to the other.

As the sketch map below showed:

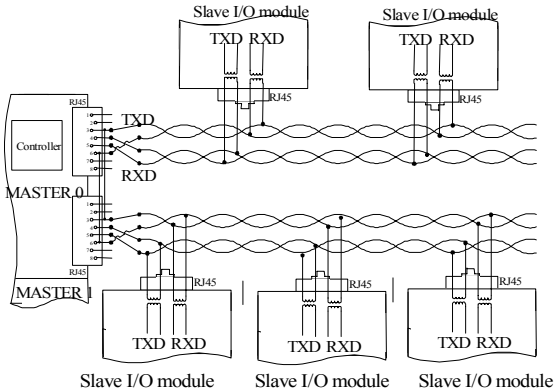


Figure 1-12: HSL wiring – RS422 with multi-drop

- There are two RJ45 notches in each master. Each notch supports one port of wiring.
- Only 4 lines of RJ-45 cable are used, 2 for transmission and 2 for reception.
- All slave modules are parallel connected.
- With isolation between connection cable and individual slave I/O module, signal is well protected from being interfered by any other slaves.

Networking Topology

As described above, the HSL system is of full-duplex RS422 with multi-drop architecture. Base on this architecture, there are a variety of methods to networking a HSL circuit, e.g.: serial, multi-drop, star …etc. It is very difficult to enumerate all the possible network topologies and the detail specifications. Instead, some real cases are presented here and they should be helpful to give senses.

Serial wiring:

All the slave I/O modules are connected one by one with twin-head 100BaseTX cables.

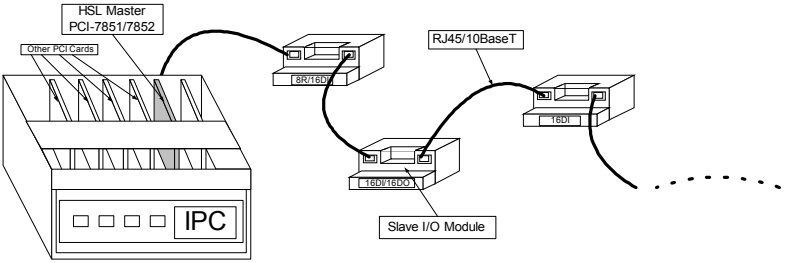


Figure 1-13: HSL networking Topology – Serial

Because the 2 notches of any slave I/O module's termination board are actually short circuit, this kind of wiring provides the easiest and most intuitive way to form a HSL network. And, the longest wiring length is 200m under 6Mbps.

I/O refreshing rate of HSL system

The scan time unit for one slave index is decided by transmission rate: 3/6/12 Mbps. Once the maximum slave address used in one HSL system is decided, the polling cycle time of this HSL system will be determined. The formula is as follows:

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

Maximum Address	Cycle Time under 3Mbps	Cycle Time under 6Mbps	Cycle Time under 12Mbps
5	303.5 μsec.	152.0 μsec.	76.0 μsec.
10	607 μsec	304.0 μsec.	152.0 μsec
20	1.214 msec.	608.0 μsec.	304.0 μsec.
30	1.821 msec.	912.0 μsec	456.0 μsec
40	2.428 msec.	1.216 msec	608.0 μsec
50	3.035 msec.	1.520 msec	760.0 μsec
60	3.642 msec.	1.824 msec	912.0 μsec
63	3.824 msec.	1.915 msec	957.6 μsec

Table 1-4: Polling cycle time of HSL (Full Duplex Mode)

Note: No matter which transmission rate you choose, the minimum polling cycle time is 3×scan time unit, even the max-

imum address is less than 3. Please also refer to the table of Appendix A.

Communication Error Handling

Though the HSL communication protocol is dedicatedly designed to avoid any errors, there is inevitably still some chance that communication error may occur, e.g.: light striking, suddenly off-line, etc. If this is the case, the master will know it. In HSL, the master holds an accumulated slave-no-response count for every individual slave I/O module. The count value will be updated for every slave module during every polling cycle.

- If communication to certain slave I/O module is successful, the no-response count value for this slave will be set to ‘0’.
- If the communication failed, the no-response count value increases by ‘1’.
- When the count is larger then or equal to ‘3’, a binary flag indexing communication error will be set to true.
- The maximum value of no-response count is ‘7’. It retained at ‘7’ even if error continue to occur.

The no-response count value and the communication error flag status can be obtained by software function call, or every time when user want to set or get I/O values, these error handling data will also be returned.

In addition to no-response count, the HSL supports a self-diagnosis function to detect off-line or out-of-communication of any slave module. A software communication error-handling driver does this. In a programmable period of time (default 20 ms), the master sends IRQ to trigger this driver to check every slave’s no-response count value. If the count value is ‘7’, the drive will inform the system, by event that some slave I/O is in mal-communication situation.

1.5 Software Support

Window 98/NT/2K/XP DLL

The HSL Windows 98/NT/2K/XP DLL (Dynamic Link Library) is provided as a programming interface under Microsoft Windows

environment. The driver can work with any Windows programming language that could integrates DLL, such as Microsoft Visual C/C++ (6.0 or above), Borland C++(5.0 or above), or Microsoft Visual Basic (6.0 or above), etc.

Linux Driver

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

2 HSL Master Controller

The HSL master is the key character that takes charge of the communication with slave I/O modules. The master set outputs values to and gathers input information from slaves by communication.

ADLINK supplies three HSL master card models: PCI-7851, PCI-7852 and PMC-7852/G, supporting different numbers of HSL masters.

- PCI-7851: Single HSL Master Controller Interface Card
- PCI-7852: Dual HSL Master Controller Interface Card
- PMC-7852/G: Dual HSL Master Controller Interface Card with PMC connector

2.1 Board Overview

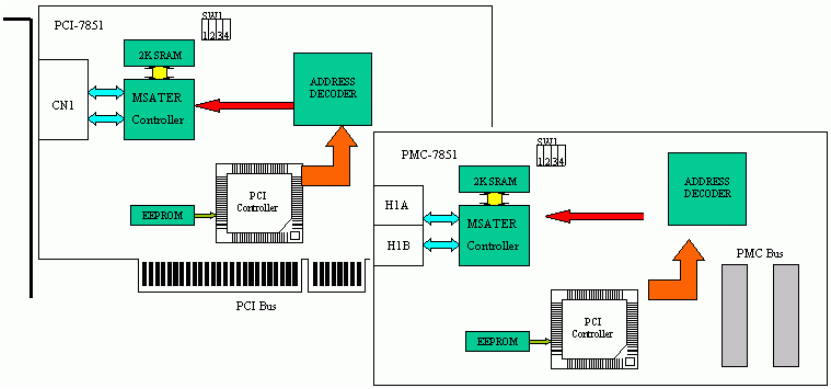


Figure 2-1: PCI-7851, PMC-7852/G sketch

2.2 Specifications

- PCI Bus:
  - PCI local bus specification Rev. 2.1 compliance
- Master Controller:
  - Master controller: HSL ASIC
  - External Clock: 48MHz
- Memory:
  - 32KB SRAM – 12ns
- Interface:
  - RS-422 with transformer isolation
  - Half / Full duplex communication
  - Selectable transmission rate by DIP switch (3/6/12Mbps, factory default is at 6Mbps)
  - Two ports for one master controller
- Connector:
  - RJ45 connector x 2 (CN1 for PCI-7851)
  - RJ45 connector x 4 (CN1, CN2 for PCI-7852; H1A, H1B, H2A, H2B for PMC-7852/G)
- Interrupt:
  - 32 bits Programmable timer
- LED Indicator:
  - Power status
- PCB Dimension:
  - For PCI-7851/7852: 176 (L) × 107 (W) mm
  - For PMC-7852/G: 74 (W) × 149 (W) mm
- Operating Temperature: 0 to 60°C
- Storage Temperature: -20 to 80°C
- Power Consumption: +5V @500mA typical

2.3 PCI-7851/7852 Outline Drawing

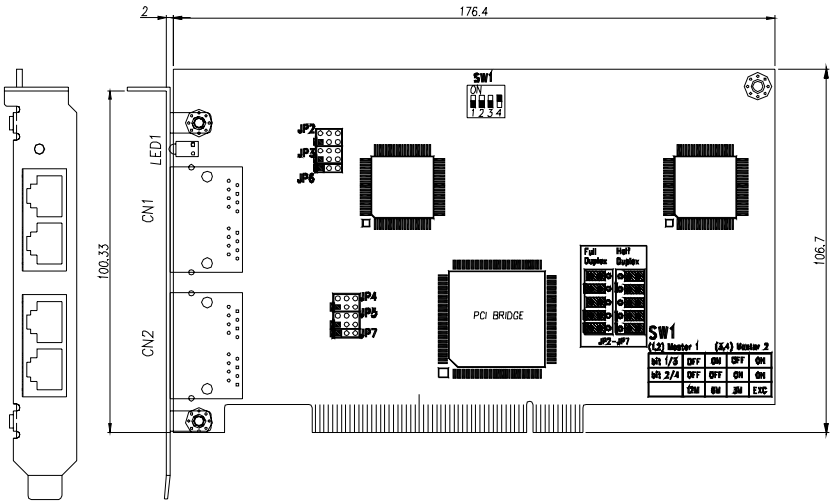


Figure 2-2: PCI-7851/7852 Outline

CN1	RJ45 connector with first HSL master controller
CN2	RJ45 connector with second HSL master controller (Only available for PCI-7852)
JP2, 3, 6	Full/Half duplex mode with first master controller (Factory default: Full duplex mode)
JP4, 5, 7	Full/Half duplex mode with second master controller (Factory default: Full duplex mode)
SW1	Transmission speed option. (Facctory default: 6 Mbps)

2.4 PMC-7852/G Outline Drawing

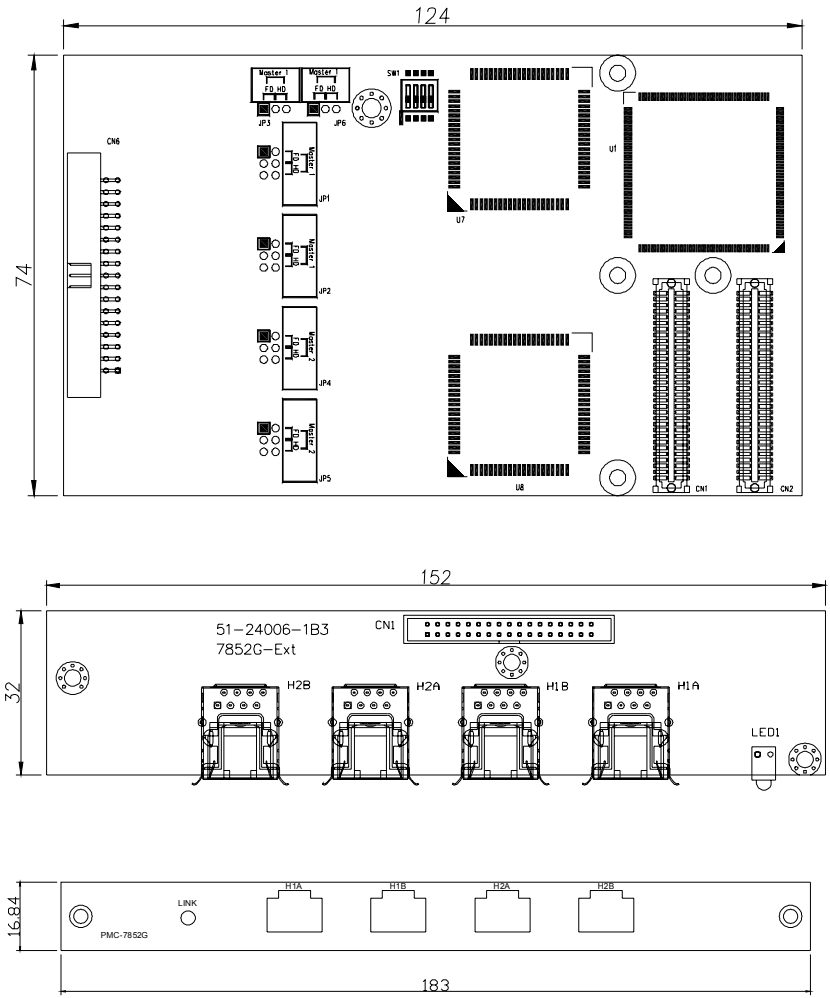


Figure 2-3: PMC-7852/G Outline

H1A, H1B	RJ45 connector with first HSL master controller

H2A, H2B	RJ45 connector with second HSL master controller (Only available for PCI-7852)
JP1, 2, 3, 6	Full/Half duplex mode with first master controller (Factory default: Full duplex mode)
JP4, 5	Full/Half duplex mode with second master controller (Factory default: Full duplex mode)
SW1	Transmission speed option. (Factory default: 6 Mbps)

2.5 Configuration

SW1

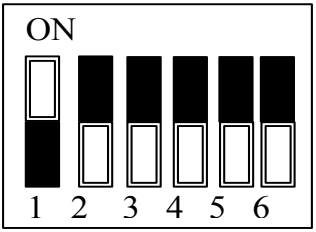
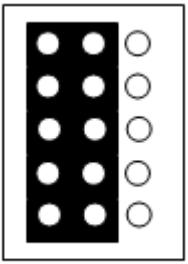


Figure 2-4: SW1 – Transmission Rate Setting

No.	Set transmission rate				Notes
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	Factory default: 6Mbps.

2.4.2 JP 2, 3, 6 / JP4, 5, 7 (For PCI-7851/PCI-7852)



FD (Default) HD

2.4.3 JP 1, 2, 3, 6 / JP 4, 5 (For PMC-7852/G)

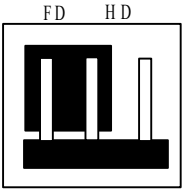
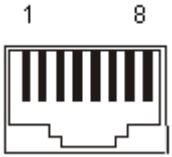


Figure 2-5: PMC-7852/G Top View, JP 1-6 Jumper Settings

2.6 PIN Assignment



RJ45 Female Connector

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

Table 2-1: Ethernet Connector

2.7 Software Architecture Description

PCI-7851 / PCI-7852 / PMC-7852/G is equipped with one or two HSL master ASICs that control the communication inside HSL system. The purpose to communicate with HSL I/O modules is to gather input data from or set output value to them. In order to achieve that purpose, each HSL master controller manages a 32Kbyte SRAM on PCI-7851/ PCI-7852 / PMC-7852/G boards to store data.

In every polling cycle, master refreshes all input data from I/O modules and set newest output data to I/O modules. The 32K SRAM keep these data, and the software drivers read/write I/O information for users.

Functional Block Diagram

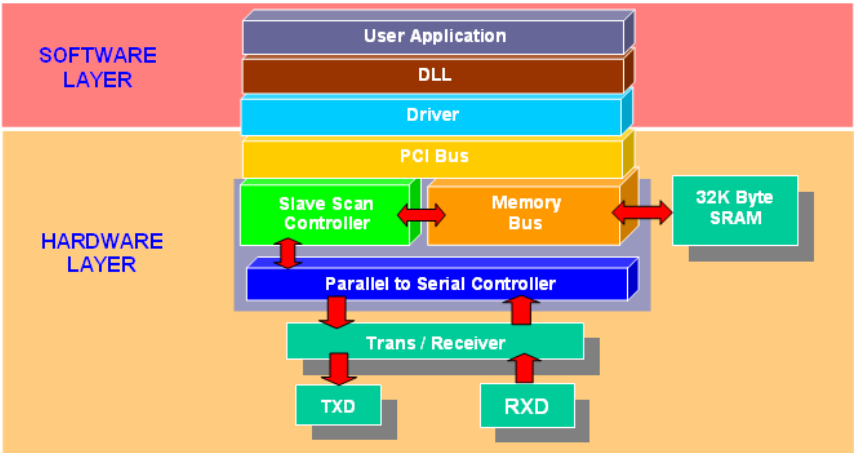


Figure 2-6: HSL Master Functional Block Diagram

The Block Diagram above shows how HSL communicates with user's AP. The SRAM act a buffer-like character.

2.8 Installation

Hardware Configuration

The PCI-7851/PCI-7852 is a plug-and-play device, so the interrupt channel and memory mapping address are assigned by system BIOS. You only need to configure the SW1 for transmission speed. For PMC-7852/G, please users refer to GEME user manual for details.

Software Configuration

Please refer to Chapter 4.

3 HSL Slave Module

The HSL is a master-slave network system, which features an innovative distributed architecture that modularizes the communication, I/O functions and signal termination. You can individually choose the slave I/O modules and terminal base to meet your particular applications requirement. ADLINK provides discrete I/O, analog I/O, thermocouple module and motion control. As for motion control module, please refer to HSL-4XMO user manuals.

The slave I/O module is a combination of following components

Slave I/O Module: There are three groups of the slave I/O module. The slave I/O module gives the terminal base different I/O capability. Regardless of the various I/O types, all slave I/O modules in the same group have the same mechanical dimension. To identify each slave I/O module in a HSL network, a module type electronic data sheet is inherent in the module itself. And the slave I/O module located by address ID, which is selectable by a 6-bit DIP-switch. Depending on different I/O supported, every slave I/O module may consume 1 or 2 address ID. Since the greatest ID number in a HSL master is 63 (6 bit and ‘0’ reserved for master), there are at most 63 slave I/O modules in one HSL master.

Terminal Base: The job of terminal base (TB) is offering an easy wiring media. Both power and signal wiring go from terminal base into the slave I/O modules. Also, the RJ-45 connector used to link the masters to all the slave I/O modules on it. With the help of TB, the slave I/O modules are hot swappable without interfering with other modules in the same HSL network.

Wiring Cable: The communication wiring cables among the HSL master and I/O modules are standard 100 Base/TX with RJ-45 connectors. There are exactly the same as commercial Ethernet cables.

3.1 Slave I/O Module

Discrete I/O Module

ADLINK provides three series: DB, M and L series.

- DB: Daughter board form factor
- M: Daughter board form factor with aluminum cover
- L: Low-profile design

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
L	HSL-DI8-L-N/P	8			1
	HSL-DO8-L-N/P		8		1
	HSL-DI4DO4-L-N/P	4	4		1

Table 3-1: I/O Module Series

HSL	-	DI#DO#	-	*	-	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 **DI8**: 8 discrete inputs **DO8**: 8 discrete outputs **DI4DO4**: 4 discrete inputs and 4 discrete outputs		Series: **DB**: Daughter board form factor **M**: Daughter board with aluminum cover **L**: Low-profile design		Signal Type: **X**: Input Signal Type: NPN sinking and PNP sourcing support **Y**: Output Signal Type: NPN sinking and PNP sourcing support

Table 3-2: I/O Module Series Selection Guide

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)

Table 3-3: I/O Module M Series

HSL	-	AI#AO#	-	*	-	XY
		Analog I/O Type: **AI16AO16**: 16 analog inputs and 16 analog outputs		**M**: Daughter board with aluminum cover		Signal Type: **X**: Input Signal Type: V for voltage and A for current **Y**: Output Signal Type: V for voltage

Table 3-4: I/O Module Series M election Guide

Thermocouple Input Module

ADLINK provides HSL-TC08 for temperature measurement. The measuring temperature range is as follows.

Type	Input Range	Type	Input Range
J	0°C-760°C	K	0°C-1370°C
T	-100°C-400°C	E	0°C-1000°C
R	500°C-1750°C	S	500°C-1750°C
B	500°C-1800°C	N	-270°C-1300°C
C	0°C-2320°C		

Motion Control

ADLINK provides two models for users to do remote motion control. The models are HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P.

HSL	-	4XMO	-	*	-	X
		Controllable Axes: **4XMO**: 4-axis pulse train type motion control		Series: **CG**: The connection interface is general type. **CD**: The connection interface is D-sub 25.		Signal Type: **X**: Output Signal Type: NPN sinking and PNP sourcing support

Table 3-5: I/O Module Series Selection Guide

HSL-4XMO-CG-N/P is suitable for those who use stepper and linear motor. For HSL-4XMO-CD-N/P, ADLINK can provide the accessory, transfer cable, to direct connect to servo amplifier. For details, please refer to HSL-4XMO manual and function library.

General Specifications

Discrete I/O Module

Discrete Input	Photo couple isolation	2500VRMS	
	Input impedance	4.7kΩ	
	Input Voltage	±40V (Max.)	
	Input Current	For NPN(1)	-10mA
		For PNP(2)	+10mA
	Operation Voltage (@ 24VDC Power Supply)	For NPN(1)	ON: 9.6VDC(Max.) OFF: 19.0VDC (Min.)
		For PNP(2)	ON: 14.4 VDC(Max.) OFF: 5.0 VDC (Min.)
	Response Time	ON: 5μs(Typical); OFF: 10μs(Typical)	
Discrete Output	Switch capacity	For NPN(3)	Single channel: -500mA All channels: -60mA at 24 VDC
		For PNP(4)	Single channel: +500mA All channels: +60mA at 24 VDC
	Response Time	ON to OFF: 180μs	
		OFF to ON: 1.2μs	
Relay	Relay Type	SPST, normally open, non-latching	
	Rating	30 VDC/2A; 250 VAC/2A	
	Switching Frequency	20 times/minute at rating load	
	Response Time	ON to OFF: 3ms(Max.)	
		OFF to ON: 6ms(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

Analog I/O Module

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

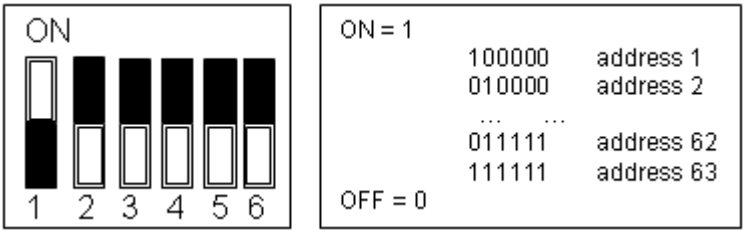
Thermocouple Module

Type	Input Range	Type	Input Range
J	0°C-760°C	K	0°C-1370°C
T	-100°C-400°C	E	0°C-1000°C
R	500°C-1750°C	S	500°C-1750°C
B	500°C-1800°C	N	-270°C-1300°C
C	0°C-2320°C		

Motion Control

Please refer to the HSL-4XMO user manual.

DIP Switch Setting:



Please note the following:

1. The address (or slave index) ‘0’ is reserved.
2. HSL-DI32-M, HSL-DO32-M, HSL-DI32-DB, and HSL-DO32-DB need two consecutive addresses that start

from an odd number. For example, if the DIP switch is set as 3, it would occupy slave index 3 and 4.

3. HSL-AI16AO2-M-VV/AV needs two leap addresses at full duplex mode. For example, if the DIP switch is 2, this module will occupy 2 and 4.
4. HSL-4XMO-CG-N/P and HSL-4XMO-CD-N/P need four leap addresses at full duplex mode. For example, if the DIP switch is 2, this module will occupy 2, 4, 6 and 8. However, at half duplex mode, it needs 4 consecutive numbers.

Wiring Diagrams

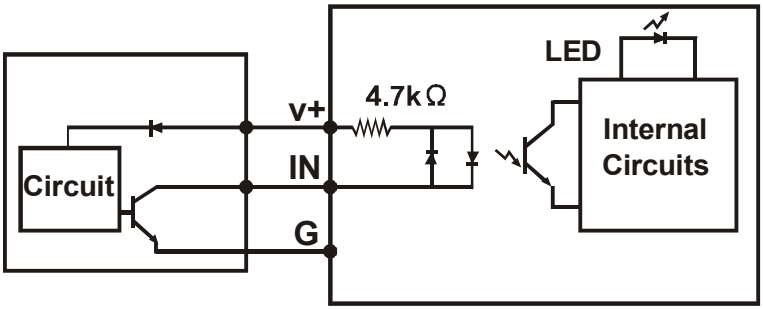


Figure 3-1: -N NPN Sinking type sensor Input

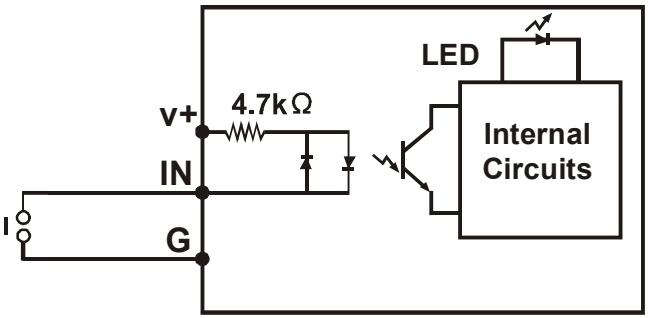


Figure 3-2: -N Dry Contact Input

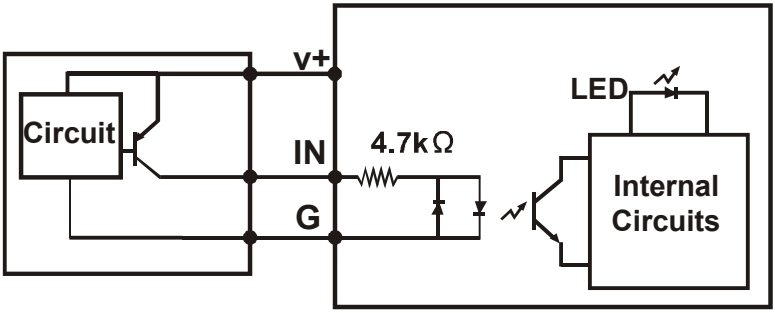


Figure 3-3: -P PNP Sourcing type sensor Input

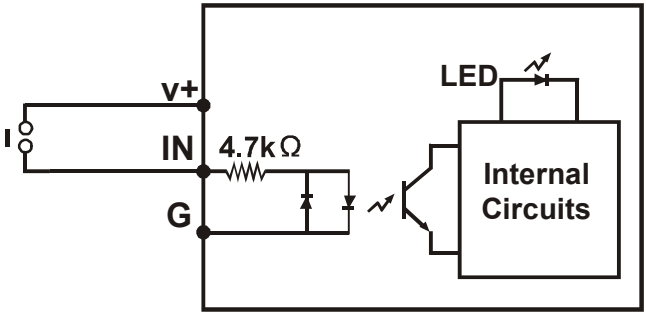


Figure 3-4: -P Wet Contact Input

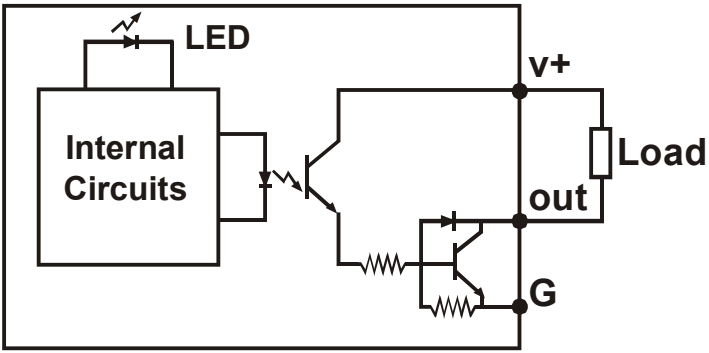


Figure 3-5: -N NPN Sinking Output

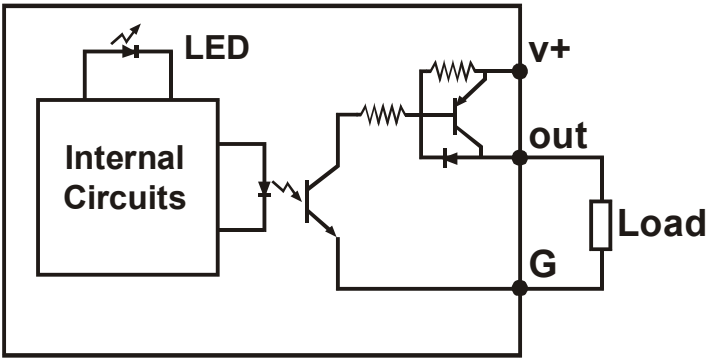


Figure 3-6: -P PNP Sourcing Output

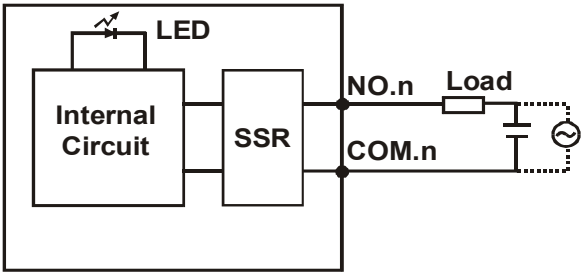


Figure 3-7: -R Relay Output

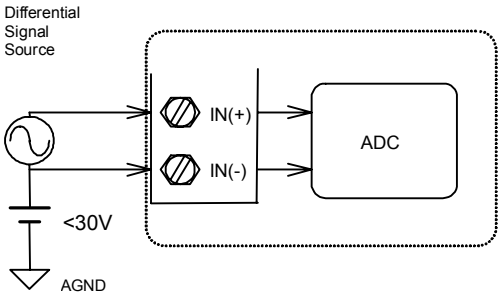


Figure 3-8: Analog Input (Differential Voltage Input)

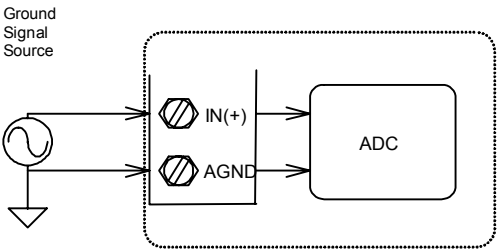


Figure 3-9: Analog Input (Single-End Voltage Input)

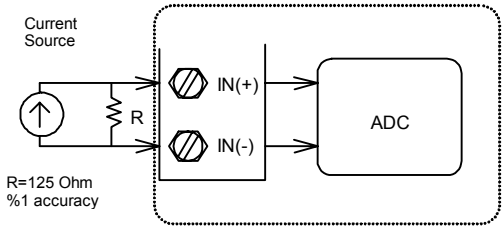


Figure 3-10: Analog Input (Current Measure)

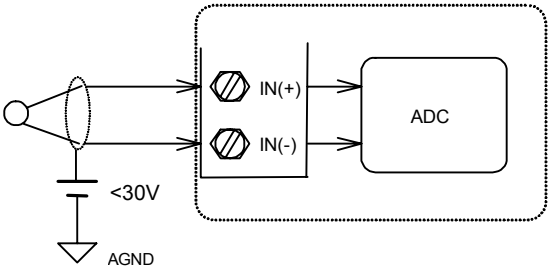
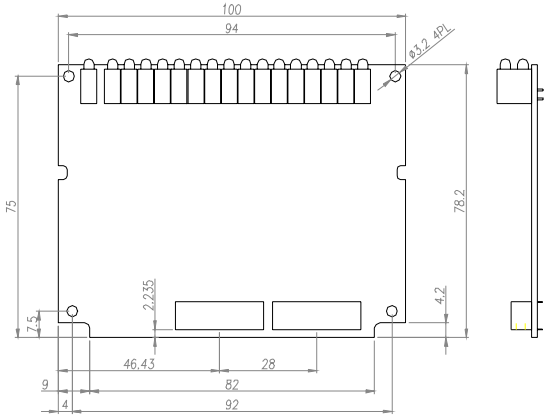


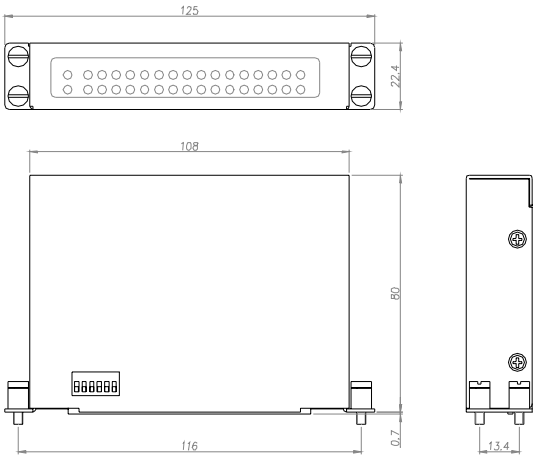
Figure 3-11: Thermocouple Measurement

Dimensions

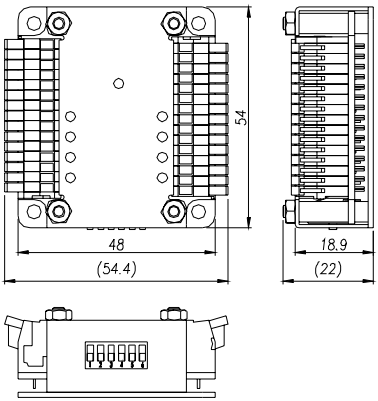
:-DB Daughter board form factor (100mm X 78.2mm)



-M Daughter board with aluminum cover (125mm X 80mm)



-L Low-profile slave I/O module (54.4 mm X 54mm)



3.2 Terminal Base

The terminal bases include:

- HSL-TB32-U-DIN
- HSL-TB64-DIN
- HSL-TB32-DIN
- HSL-TB32-DO-DIN
- HSL-TB32-M-DIN

Features

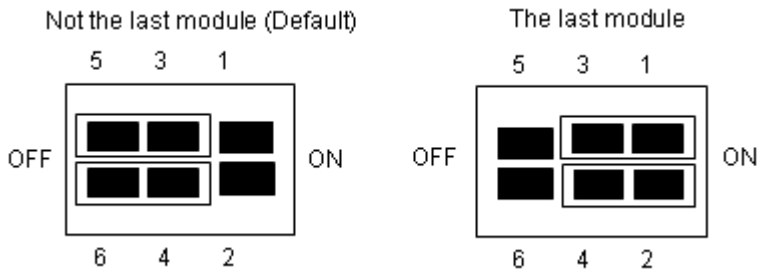
- Field I/O wiring connection for HSL I/O modules
- Screw or spring terminal for easy field wiring
- Power and ground included for each signal channel
- Interlocking design for rugged installation
- Power LED indicator
- DIN rail mounting
- Terminator resistor on board

General Description

	Model Name	Description	Module Support
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
	HSL-TB32	(1) 16 channels short circuit protected current driver (up to 300mA/ch) and 16 channels direct connected terminal base (2) One DB slot	HSL-DI16DO16-DB-NN HSL-DI16DO16-DB-PN
	HSL-TB32-DO	(1) 32 channels short circuit protected current driver (up to 300mA/ch) terminal base (2) One DB slot	HSL-DO32-DB-N
For M Series	HSL-TB32-M	32 channels direct connected terminal base for HSL M-series module	All HSL M-series modules

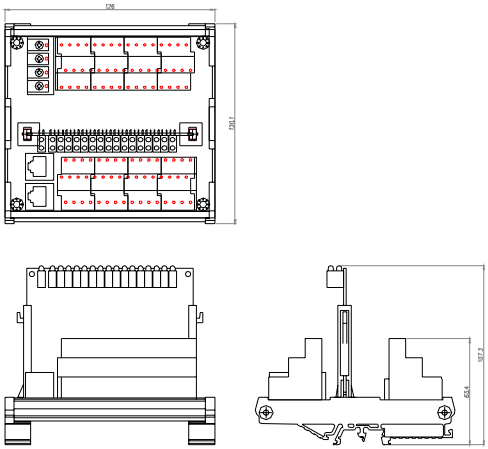
Jumper Setting

Since HSL is a serial transmition system, a terminator should be placed at the end of cable. Each TB has a jumper selectable terminator on board. Only the last module have to enable the terminator.

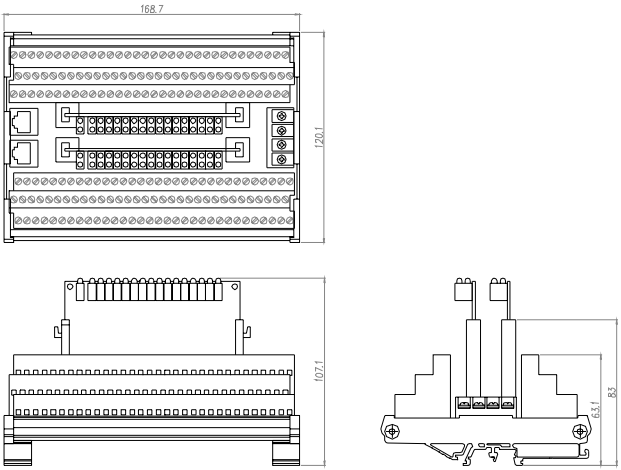


Dimensions

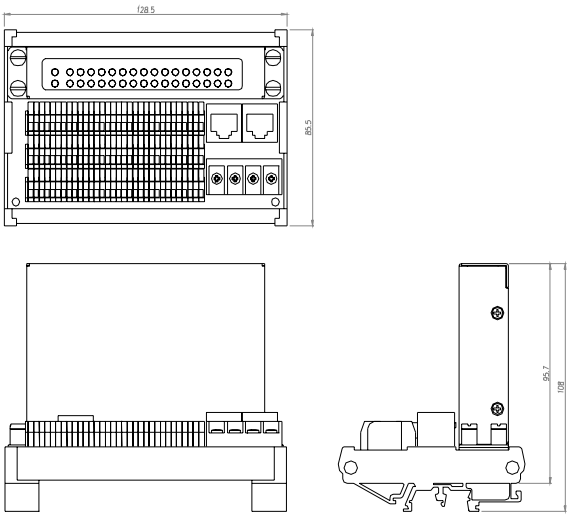
-DB with HSL-TB32-DIN or HSL-TB32-U-DIN (126x120.1x107.3) mm



-DB with HSL-TB64-DIN (168.7x120.1x107.3) mm



-M module with HSL-TB32-M-DIN (128.5x85.5x108) mm



3.3 How to Manage Slave Index within a HSL Network

Notes about managing slave modules in a HSL network

Before powering on the slave modules, users have to adjust the DIP switch. For this step, please refer to 3.1 to know how to do this.

- ▶ One master controller can connect up to 63 slave indexes. For example, PCI-7852 has two master controllers, so the maximum is 126 slave indexes for this master board.
- ▶ The more compact slave address in HSL network, the much more efficiently HSL can work.
- ▶ Discrete I/O & 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-DI32-M-N/P	2 (Consecutive from odd number)		
HSL-DI32-DB-N/P			
HSL-DO32-M-N/P			
HSL-DO32-DB-N/P			

- ▶ Analog I/O & 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-TC08			

▶ 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	3/6/12Mbps Selectable
HSL-4XMO-CD-N/P			

▶ Special Rules

1. If users only have one HSL-AI16AO2-M-VV (or HSL-AI16AO2-M-AV) and the DIP switch is set as 1 (HSL-AI16AO2-M-VV/AV only support full duplex mode), the occupied index would be 1 and 3. At this time, users have to assign the parameter “max_slave_No” of HSL_start(…) as 4 to ensure the correct communication. Except this address number, there is no other rule. If using HSL_auto_start function, users do not have to care about this.
2. If users only have one HSL-4XMO-CG-N/P (or HSL-4XMO-CD-N/P) and the DIP switch is set as 1 and full duplex mode, the occupied index would be 1, 3, 5, and 7. At this time, users have to assign the parameter “max_slave_No” of HSL_start(…) as 8 to ensure the correct communication. Except this address number, there is no other rule. If using HSL_auto_start function, users do not have to care about this. While in half duplex mode, it will occupy 1, 2, 3 and 4, therefore, users just assign the “max_slave_No” of HSL_start(…) as 4 or call HSL_auto_start.

Examples

We will give users some setting examples and help users to easily understand it. All modules in examples are used as full duplex modem.

Example 1: If users have HSL-DO8-L-N×2, HSL-DI32-M-N×2 and HSL-AI16AO2-M-VVx1 and all slave modules are in full duplex mode, we can have two conditions as follows:

Condition 1: HSL-AI16AO2-M-VV×1 is in 6Mbps.

ADLINK suggests users use the slave index configuration as follows:

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-M-VV	5	5, 7
HSL-DO8-L-N #1	6	6
HSL-DO8-L-N #2	8	8

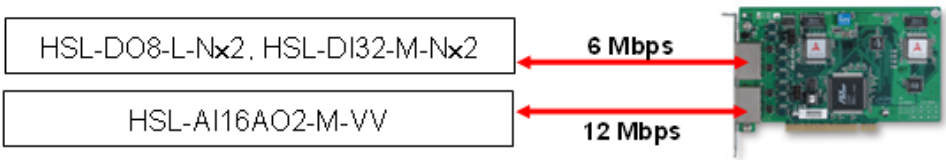
Therefore, it is the compact composition. The scan time would need 30.4μs× 8 at 6Mbps, full duplex mode. Users can connect these entire modules with one master controller.

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

ADLINK suggests users use the slave index configuration as follows:

Item	DIP Switch	Index Occupation in HSL
HSL-DI32-M-N #1	1	1, 2
HSL-DI32-M-N #2	3	3, 4
HSL-DO8-L-N #1	5	5
HSL-DO8-L-N #2	6	6

Therefore, it is the compact composition. The scan time would need 30.4μs× 6 at 6Mbps, full duplex mode. Users can connect these entire modules with one master controller. For HSL-AI16AO2-M-VV, it connects to another master controller. The DIP switch of HSL-AI6AO2-M-VV is assigned as 1. Because of the special rule description, users have to assign the “max_slave_No” of HSL_start(…) as 3 or call HSL_auto_start by connect_index #1. The illustrated diagram is as follows.



Consequently, the cycle time of first master controller is 30.4μs× 6 and the cycle time of second master controller is 45.6μs at 12Mbps, full duplex mode.

Example 2: If users have HSL-DI8-L-N×2, HSL-DI16DO16-M-NN×1, HSL-DO32-M-N×2 ,HSL-AI16AO2-M-VV×1, and HSL-4XMO-CG-N×2, and all slave modules are in full duplex mode, we can have two conditions as follows:

Condition 1: HSL-AI16AO2-M-VV×1 and HSL-4XMO-CG-N×2 is in 6Mbps.

ADLINK suggests users use the slave index configuration as follows:

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-AI16AO2-M-VV	13	13, 15
HSL-DI8-L-N #1	14	14
HSL-DI8-L-N #2	16	16
HSL-DI16DO16-M-NN	17	17

Therefore, it is the compact composition. The scan time would need 30.4μ × 17 at 6Mbps, full duplex mode. Users can connect these entire modules with one master controller.

Condition 2: HSL-AI16AO2-M-VV×1 and HSL-4XMO-CG-N×2 operate at 12 Mbps.

ADLINK suggests users use the slave index configuration as follows:

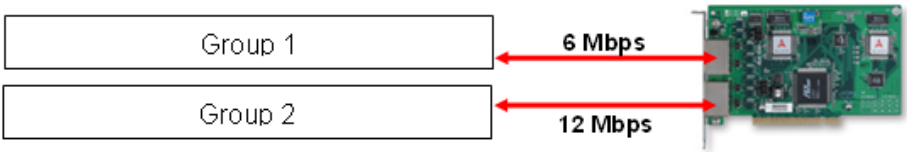
Group 1	DIP Switch	Index Occupation in HSL
HSL-DO32-M-N #1	1	1, 2
HSL-DO32-M-N #2	3	3, 4
HSL-DI8-L-N #1	5	5
HSL-DI8-L-N #2	6	6

Group 1	DIP Switch	Index Occupation in HSL
HSL-DI16DO16-M-NN	7	7

Therefore, it is the compact composition. The scan time would need 30.4μs× 7. Users can connect these entire modules with one master controller. For HSL-AI16AO2-M-VV and HSL-4XMO-CG-N×2, they connect to another master controller. The management table is as follows.

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

The illustrated diagram is as follows.



Consequently, the cycle time of first master controller is 30.4μs × 7 and the cycle time of second master controller is 15.2μs × 11 at 12Mbps, full duplex mode.

4 HSL LinkMaster Utility

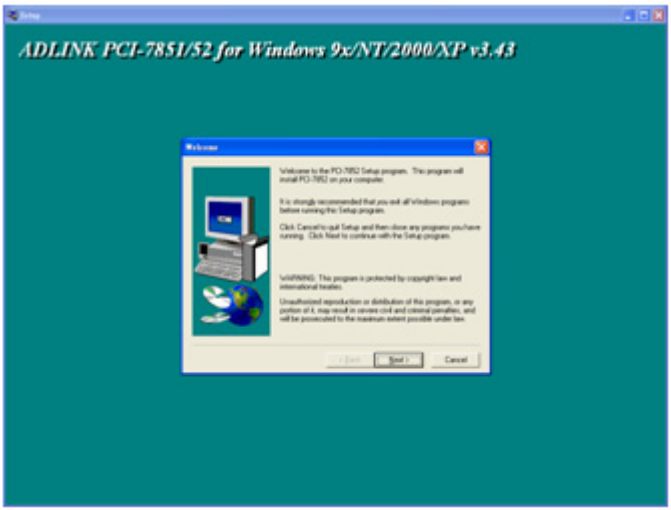
After installing master controller and slave modules, users can install the HSL driver and use the LinkMaster utility to test and debug the system. This utility provides a user-friendly interface for customers. Users can easily test I/O status, including read and write I/O data, calibration, and motion control. We strongly suggest that users use this utility before implementing the whole system.

Note that LinkMaster utility is only available for Windows 98/NT/2K/XP with screen resolutions greater than 800×600.

4.1 Software Installation

ADLINK provides two ways to get the installation driver. One is from “ADLINK All-in-One CD (Automation)” which is ready inside the package; the other is from ADLINK website. Users can access the latest version from our website. The installation steps are described as follows.

1. Find the SETUP.exe and execute the file.
2. You will see the screen as follows.



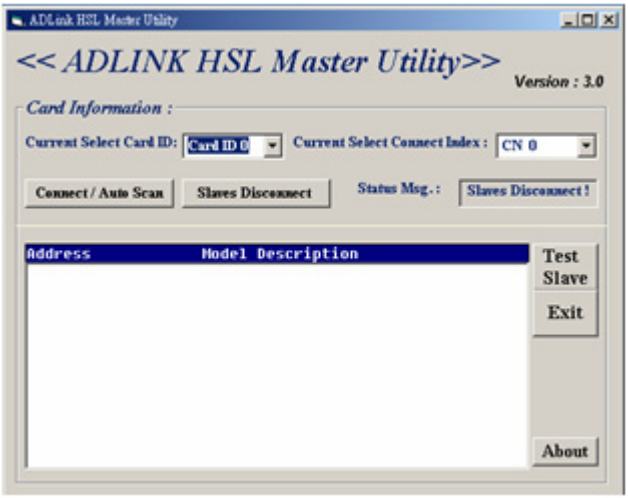
Then, follow the instructions to finish the installation.

3. After complete the installation processes, you have to restart Windows so that the HSL drivers will work normally.

4.2 ADLINK HSL LinkMaster Utility

Run the LinkMaster Utility

After driver installation, you can find the utility placed in “Start menu”->”PCI-7851”. Click on the “LinkMaster” and you will see the main menu as follows.



About the LinkMaster Utility

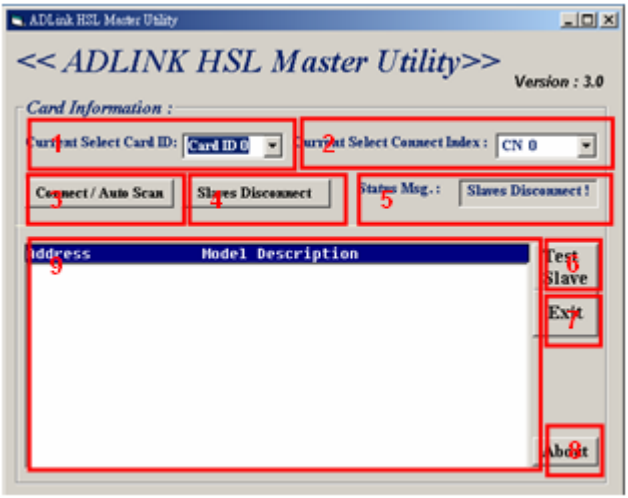
Before running LinkMaster, please review the following notes.

1. LinkMaster is a testing and debug program based on VB 6.0 and is only available for Windows 98/NT/2K/XP with

a screen resolution greater than 800×600. It is not compatible with DOS.

2. LinkMaster has version control. Users can check the version number on the right-top corner.
3. Every slave module can be tested with this utility, including discrete I/O, analog I/O, thermocouple module and motion control modules. As for motion control utility and manipulation, please refer to the HSL-4XMO manual.

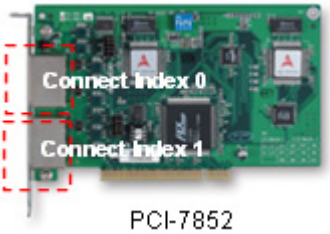
LinkMaster Utility Introduction



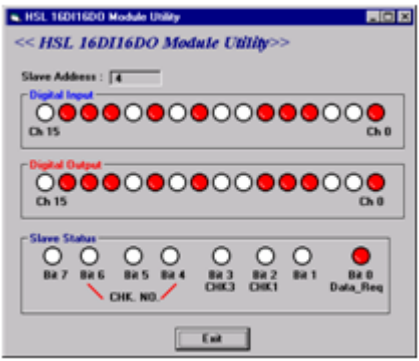
All the buttons on the main menu are described as follows.

1. Current Select Card ID: While activating this utility, it will search all HSL master control cards that exist in the system, including PCI-7851, PCI-7852 and PMC-7852/G. Every card has its own index (ID) ranging from 0. Users can specify the card index here to operate.
2. Current Select Connect Index: PCI-7852 and PMC-7852/G have two master controllers inside. The connect index is ranges from 0 to 1. For PCI-7851, it has one

master controller inside. The connect index is 0. The following diagram shows that:



3. Connect/Auto Scan: When this button is pressed, the utility will scan all the slave modules connected to master card with specified connect index. This utility will show all information on the slave modules, including the address and slave type within the 9th block.
4. Slaves Disconnect: When users press this button, the utility will stop scanning all the slave modules and disconnect them.
5. Status Msg: Users can check whether the slave modules are connected or disconnected.
6. Test Slave: While the all connected slave modules list in the 9th block, users can click on one of them to activate the testing dialog. For example, users can connect HSL-DI16DO16-M-NN and see this module shown in the screen, then click on it and will see the dialog as follows.

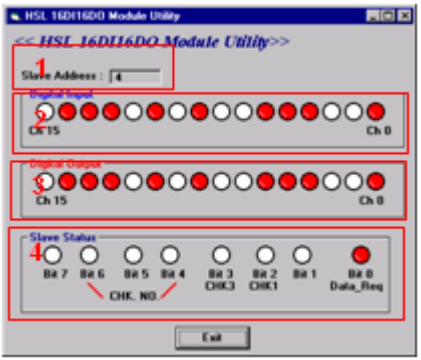


Users can test and debug modules based on this utility.

7. Exit: Press this button to exit the utility.
8. About: This contains DLL version information.

The next section will outline the usage of every slave module utility.

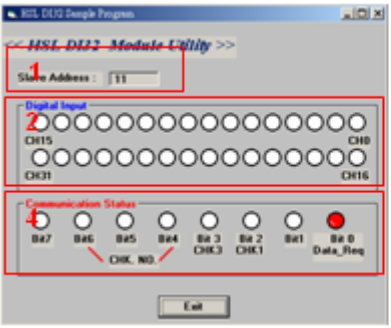
HSL-DI16DO16 Utility

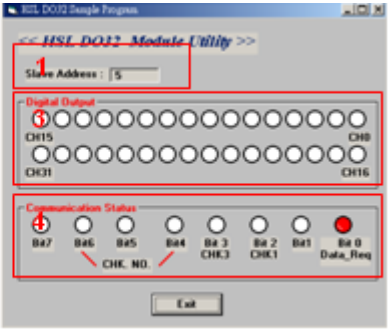


All contents are described as follows.

1. Slave Address: The slave index occupied by this module. This module just occupies one slave index.
2. Digital Input: The white icon stands for no digital input; the red icon means that the digital input is not activated.
3. Digital Output: Users can directly click on the icon to activate the digital output. Red icon means the digital output is turned on, and vice versa.
4. Slave Status: This shows the communication status between this slave module and master card. 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 one communication error).
   - ▷ Bit 3 is for CHK3. (If Bit3 is 1. It means that there are three communication errors).
   - ▷ Bit 4, Bit 5, and Bit 6 are for CHK7. (If Bit 4, Bit 5, and Bit 6 all are 1, that there are 7 communication errors).

HSL-DI32 & HSL-DO32 Utility

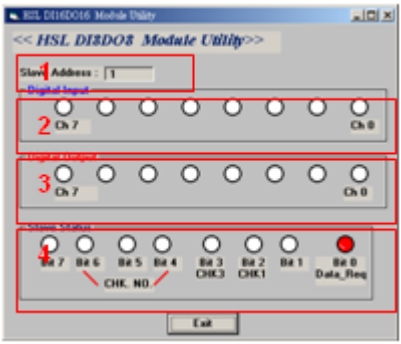




All contents are described as follows.

1. Slave Address: The slave index occupied by this module. These modules occupy two slave indexes from odd number. For example, if users adjust the DIP switch of HSL-DI32 or HSL-DO32 as 3, it will have two slave indexes as 3 and 5 actually.
2. Digital Input: The white icon stands for no digital input; the red icon means that the digital input is not activated.
3. Digital Output: Users can directly click on the icon to activate the digital output. Red icon means the digital output is turned on, and vice versa.
4. Slave Status: This shows the communication status between this slave module and master card. 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 one communication error).
   - ▷ Bit 3 is for CHK3. (If Bit3 is 1. It means that there are three communication errors).
   - ▷ Bit 4, Bit 5, and Bit 6 are for CHK7. (If Bit 4, Bit 5, and Bit 6 all are 1, that there are 7 communication errors).

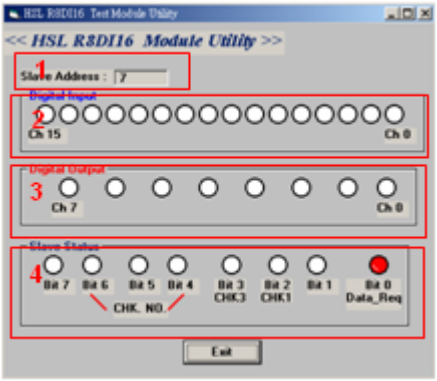
HSL-DI8, HSL-DO8, HSL-DI4DO4 Utility



HSL-DI8-L, HSL-DO8-L, and HSL-DI4DO4-L series have the same utility dialog as above. All contents are described as follows.

1. Slave Address: The slave index occupied by this module. These modules only occupy one slave index.
2. Digital Input: The white icon stands for no digital input; the red icon means that the digital input is not activated.
3. Digital Output: Users can directly click on the icon to activate the digital output. Red icon means the digital output is turned on, and vice versa.
4. Slave Status: This shows the communication status between this slave module and master card. 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 one communication error).
   - ▷ Bit 3 is for CHK3. (If Bit3 is 1. It means that there are three communication errors).
   - ▷ Bit 4, Bit 5, and Bit 6 are for CHK7. (If Bit 4, Bit 5, and Bit 6 all are 1, that there are 7 communication errors).

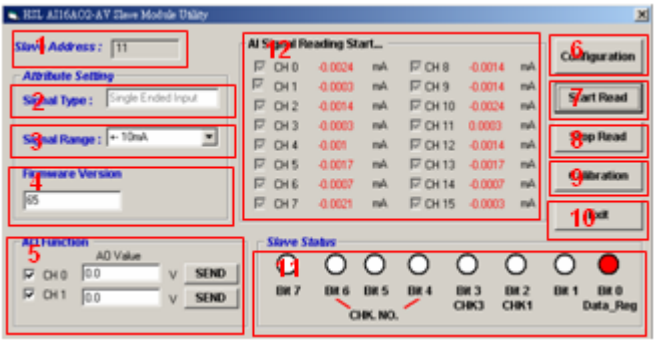
HSL-R8DI16 Utility



All contents are described as follows.

1. Slave Address: The slave index occupied by this module. This module only occupies one slave index.
2. Digital Input: The white icon stands for no digital input; the red icon means that the digital input is not activated.
3. Digital Output: Users can directly click on the icon to activate the digital output. This action will turn the relay ON or OFF. Red icon means the relay is turned on, and vice versa.
4. Slave Status: This shows the communication status between this slave module and master card. 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 one communication error).
   - ▷ Bit 3 is for CHK3. (If Bit3 is 1. It means that there are three communication errors).
   - ▷ Bit 4, Bit 5, and Bit 6 are for CHK7. (If Bit 4, Bit 5, and Bit 6 all are 1, that there are 7 communication errors).

HSL-AI16AO2 Utility



All contents are described as follows.

1. Slave Address: The slave index occupied by this module. The module will occupy two consecutive indexes. For example, if users adjust the DIP switch of HSL-AI16AO2 as 4, it will actually have two slave indexes as 4 and 6.
2. Signal Type: HSL-AI16AO2 offers two signal types, single-ended and differential. This is selectable by jumper.
3. Signal Range: Users can select the signal range. It offers 4 ranges: ±10V, ±5V, ±2.5V and ±1.25V for HSL-AI16AO2-M-VV. For HSL-AI16AO2-M-AV, the signal ranges are 20mA, 10mA and 5mA.
4. Firmware Version: The latest firmware version number of this module.
5. AO Function: Type in the analog output value in the text box and press the “SEND” button to trigger the AO. The range is ±10V.
6. Configuration: Before pressing “Start Read” button, users have to check if the signal range is correct. Press

"Configuration" button, you can save the information and complete the configuration task.

7. Start Read: Enable A/D conversion task to read back the analog input values. Users will see the values shown in 12th block.
8. Stop Read: Disable A/D conversion task.
9. Calibration: Calibrate this module. ADLINK delivers the modules with well calibration. If users want to calibrate it again, please refer to Appendix C.
10. Exit: Leave the program.
11. Slave Status: This shows the communication status between this slave module and master card. 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 one communication error).
    - ▷ Bit 3 is for CHK3. (If Bit3 is 1. It means that there are three communication errors).
    - ▷ Bit 4, Bit 5, and Bit 6 are for CHK7. (If Bit 4, Bit 5, and Bit 6 all are 1, that there are 7 communication errors).

HSL-4XMO Utility

Please refer to the HSL-4XMO manual.

5 HSL Function Library

This chapter describes the supporting software for HSL system. User can use these functions to develop programs in C, C++, or Visual Basic.

5.1 List of Functions

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

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

Table 5-1: Data Types

All the HSL function calls are revised. ADLINK proposes the new naming rule and the previous ones are also reserved. Users can refer to the mapping table in the Appendix B. All function calls have the same prefix as HSL_. If they belong to system level purpose, the function will be as follows.

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

If they belong to discrete I/O modules, the function will be as follows.

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

If they belong to analog I/O modules, the function will be as follows.

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

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

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

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

Initialization & System Information Section 5.2

Function Name	Description
HSL_initial	Master card initialization
HSL_close	Release all resources occupied by master card
HSL_start	Start to scan all the salve 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_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
HSL_disable_timer_interrupt	Disable timer interrupt of master card
HSL_set_timer	Set the resolution of timer

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

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

5.2 Initialization & 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_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:

Initialize the hardware and software states of the HSL master card (PCI-7851/52 or PMC-7852/G). Users can check the return code of this function to know if the initialization is successful or not. Because HSL master card supports plug-and-play design, the base address and IRQ level are assigned by BIOS directly.

HSL_close:

This function is to release 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 HSL master card.

HSL_start:

This function is used to scan the total connected slave modules. Users can assign how many slave indexes HSL master board should scan. Then, it will scan from 1 to the specified value.

HSL_auto_start:

This function is used to automatically detect the total connected slave modules. Every master controller can connect up to 63 slave indexes. As a result, It will scan from 1 to 63.

HSL_stop:

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

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_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_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
```

@ Arguments

`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.

`*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
```

5.3Timer Control

@Name

HSL_enable_timer_interrupt – Enable timer interrupt of master card

HSL_disable_timer_interrupt – Disable timer interrupt of master card

HSL_set_timer – Set the resolution of timer

@ Description

HSL_enable_timer_interrupt:

This function is used to enable the hardware timer interrupt of this master card.

HSL_disable_timer_interrupt:

This function is used to disable the hardware timer interrupt of this master card.

HSL_set_timer:

This function is used to setup the Timer 1 and Time 2. Timer 1 & Timer 2 are used as frequency divider for generating constant timer interrupt sampling rate dedicatedly. The highest timer interrupt sampling rate of the master card cannot exceed 20KHz on Windows NT platforms because of the Windows NT limitations. The following is an example at 6Mbps:

If users want to have sampling rate as 15kHz, the usage would be as

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

If users want to have sampling rate as 1.2kHz, the usage would be as

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

The formula is: Transmission speed / (c1 × c2)

Besides, the value of c1 and c2 must be greater then 1. When c1=0 or c2=0, the timer interrupt will be stopped.

@ Syntax

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

Visual Basic (Windows 98/NT/2K/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
```

sdd@ Arguments

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

`*phEvent`: Return the handle of the timer interrupt event . The interrupt event is used to indicate an interrupt which is generated from the master card's timer.

`c1`: Frequency divider of Timer 1.

`c2`: Frequency divider of Timer 2.

@ Return Code

```
ERR_No_Error
ERR_Invalid_Board_Number
```

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

@ Description

`HSL_D_read_input:`

This function is to read the digital input value of the discrete I/O module. Users have to specify the connect index and slave index.

`HSL_D_read_channel_input:`

This function is to read the digital input value of the discrete I/O module at the specified channel.

`HSL_D_write_output:`

This function is to write the digital output value of the discrete I/O module. Users have to specify the connect index and slave index.

`HSL_D_write_channel_output:`

This function is to write the digital output value of the discrete I/O module at the specified channel.

`HSL_D_read_ouput:`

This function is to write all digital output values to all connected discrete I/O modules. It will map all data into memory. With this function, user can write all digital output values to the all connected discrete I/O modules at one time.

`HSL_D_read_all_slave_input:`

This function is used to read the digital input values from all slave I/O modules which the set value is connect_index and card no is card_ID.

In this function, user can read all digital input values from all slave I/O modules at one time.

`HSL_D_write_all_slave_output:`

This function is to write the digital output values from all slave I/O modules which the set value is connect_index and card no is card_ID.

In this function, user can write all digital output values from all slave I/O modules at one time.

`HSL_D_set_input_logic:`

This function is to set 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:`

This function is to set 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.

@ Syntax

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

Visual Basic (Windows 98/NT/2K/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
```

@ Arguments

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.

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

out_data: The digital output of this discrete module. The definition is as follows.

- For 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.
- For HSL_D_write_channel_output: The value is digital output data of the specified channel.

***out_data**: It is a unsigned short array pointer. User have to create an unsigned short array which contain 63 cells. The cell index is corresponding to slave index. For example, cell index 0 corresponds to the module which slave index is 1. The cell index 1 corresponds to the module which slave index is 2 and so on. The last cell index 62 correpsonds to the module which slave index is 63.

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

`*in_data`: The input data of slave modules. The definition is as follows.

- 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 digital input data of the specified channel.
- For HSL_D_all_slave _index: It is a unsigned short array pointer. User have to create an unsigned short array which contain 63 cells. The cell index is corresponding to slave index. For example, cell index 0 corresponds to the module which slave index is 1. The cell index 1 corresponds to the module which slave index is 2 and so on. The last cell index 62 correpsonds to the module which slave index is 63.

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

`channel`: Specify the channel of the discrete I/O module which want to perform this function. The valid value is described as follows.

- HSL-R8DI16: 0 - 15
- HSL-DI16DO16: 0 - 15
- HSL-DI32: 0 - 31
- HSL-DO32: 0 - 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`: Set the input logic to the specified module.

`output_logic`: Set the output logic to the specified module.

@ Return Code

```
ERR_No_Error
ERR_Invalid_Board_Number
```

```
ERR_Memory_Mapping
ERR_Connect_Index
ERR_Satellite_Number
ERR_Over_Max_Address
```

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:
```

This function is used to initialize the reading operation of the analog input channels of all HSL AI/O modules which are connected to the master card.

Before using HSL_A_read_input(), HSL_A_write_output() and HSL_A_sync_rw(), the functions need to be executed first to start the A/D conversion.

`HSL_A_stop_read:`

This function is used to stop the reading operation of analog input channels of all HSL AI/O modules which are connected to the master card.

When you want to stop the A/D conversion, please use this function to stop it.

`HSL_A_set_signal_range:`

This function is used to set the input range of the specified HSL AI/O modules.

`HSL_A_get_signal_range:`

This function is used to get the input range of the specified HSL AI/O modules.

`HSL_A_get_input_mode:`

This function is used to get the signal input mode of HSL AI/O modules. This is determined by hardware jumper setting.

`HSL_A_set_last_channel:`

This function is used to set the last number of analog input channels of HSL AI/O modules. For example, HSL-AI16AO2 has 16 analog input with single-ended wiring. If users just want to read back the first 4 analog input data of this module, assign the last channel as 3. The analog input channel index is from 0. As a result, the AI channel 0 to 4 would be enabled and the others would be disabled.

`HSL_A_get_last_channel:`

This function is used to retrieve 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 with HSL_A_get_lastchannel.

`HSL_A_read_input:`

This function is used to read the specified AI channel of the slave module.

`HSL_A_write_output:`

This function is used to write the specified AO channel of the slave module.

`HSL_A_read_output:`

This function is used to read back the analog output data from the HSL AI/O modules with the specified the analog output channel.

`HSL_A_sync_rw:`

This function is used to read AI data and write AO data at the specified channel synchronously of HSL AIO module. It can let users read/write the data at one time.

`HSL_A_get_version:`

This function is to read the kernel version of the HSL AI/O modules.

@ Syntax

C/C++ (DOS, Windows 98/NT/2K/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/2K/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
```

@ Arguments

`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.

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

`signal_range`: The single range of analog input setting.

For HSL-AI16AO2-M-VV

- 0: ± 1.25V
- 1: ± 2.5V
- 2: ± 5V
- 3: ± 10V

For HSL-AI16AO2-M-AV

- 0: ± 5mA
- 1: ± 10mA
- 2: ± 20mA
- 3: ± 20mA

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

For HSL-AI16AO2-M-VV

- 0: ± 1.25V
- 1: ± 2.5V
- 2: ± 5V
- 3: ± 10V

For HSL-AI16AO2-M-AV

- 0: ± 5mA
- 1: ± 10mA
- 2: ± 20mA
- 3: ± 20mA

`*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`: User 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`: Specify the AI channel of the slave module which want 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`: Specify the AI channel of the slave module which want 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 unit of 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
```

6 How to Program with the HSL DLL

The programming flow chart is as follows:

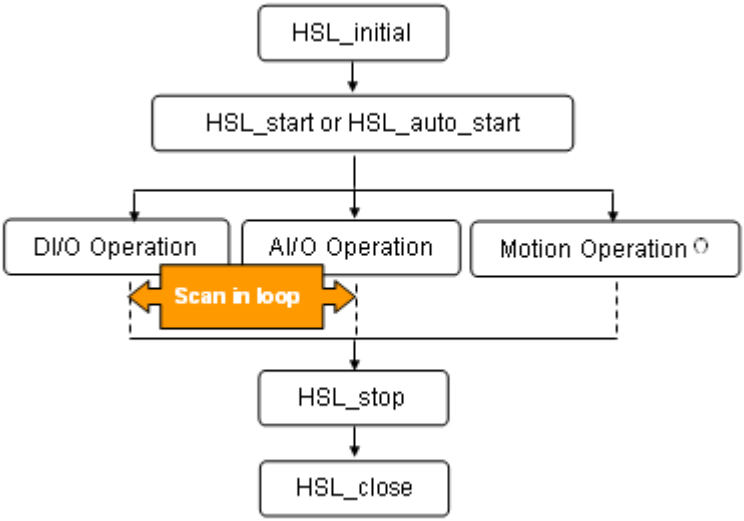


Figure 6-1: Programming Flow Chat

6.1 DI/O Operations

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

- HSL_slave_live (…): It is used to detect the status of the slave module(Live or Die).
- HSL_connect_status(…): It is used to detect the communication status of the slave module.
- HSL_D_read_input(…),HSL_D_read_channel_input(….), HSL_D_read_all_slave_input(….): It is for digital input operation of slave module.
- HSL_D_write_output(…), HSL_D_write_channel_output(…),

HSL_D_write_output(…): It is for digital output operation of slave module.

- HSL_D_read_output(…): It is for reading output data in memory.
- HSL_D_set_input_logic(…), HSL_D_set_output_logic(…): It is for setting DIO logic.

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

6.2 AI/O Operations

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

- If the module needs to be calibrated, please refer to Appendix C for
- Use HSL_A_set_signal_range(…), HSL_A_set_last_channel(..) to setting the AI/O configuration of the slave module. If users want to check AI/O configuration, please use HSL_A_get_signal_range(…), HSL_A_get_input_mode(…), and HSL_A_get_last_channel(…).
- Use HSL_A_start_read(…) to initialize the AIO channels reading operation.
- Now, the HSL AD conversion is activated. You can use some functions as below for HSL operation.
  - HSL_slave_live(…): It is used to detect the status of the slave module(Live or Die).
  - HSL_connect_status(…): It is used to detect the communication status of the slave module.
  - HSL_A_read_input(…): It is for analog value reading operation of slave module.
  - HSL_A_write_output(…): It is for analog value writing operation of slave module.
  - HSL_A_sync_rw(…): It is for analog input and output synchronously.
- Use HSL_A_stop_read(….) to stop the AIO channels reading operation.

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

6.3 Motion Operations

Please refer to HSL-4XMO user manual for details.

7 Appendix

7.1 Scan Time Table

Slave Index Number	Cycle Time under 3Mbps	Cycle Time under 6Mbps	Cycle Time under 12Mbps
Base Unit	60.7μs	30.4μs	15.2μs
< 3*	182.1s	91.2μs	45.6μs
5	303.5μs	152.0μs	76.0μs
10	607μs	304.0μs	152.0μs
20	1.214ms	608.0μs	304.0μs
30	1.821ms	912.0μs	456.0μs
40	2.428ms	1.216ms	608.0μs
50	3.035ms	1.520ms	760.0μs
60	3.642ms	1.824ms	912.0μs
63	3.824ms	1.915ms	957.6μs

Table 7-1: Full Duplex Mode

*The minimum scan time for full duplex mode at different transmission speed.

Slave Index Number	Cycle Time under 3Mbps	Cycle Time under 6Mbps	Cycle Time under 12Mbps
Base Unit	118μs	59μs	29.5μs
< 3*	354μs	177μs	88.5μs
5	590μs	295μs	147.5μs
10	1.18ms	590μs	295μs
20	2.36ms	1.18ms	590μs
30	3.54ms	1.77ms	885μs
40	4.72ms	2.36ms	1.18ms
50	5.9ms	2.95ms	1.475ms
60	7.08ms	3.54ms	1.77ms
63	7.434ms	3.717ms	1.859ms

Table 7-2: Half Duplex Mode

*The minimum scan time for half duplex mode at different transmission speed.

7.2 Mapping Table

HSL has two kinds of function library existing in HSL.h. The following is the mapping table between new and old one.

Initialization & 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

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

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	

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

7.3 HSL-AI16AO2 Calibration

Before Calibration

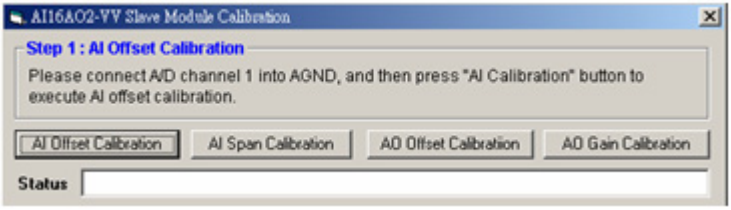
Before calibrating the HSL-AI16AO2-M-VV and HSL-AI16AO2-M-AV, please note the following:

- Make sure the signal type is single ended. It is selectable by jumper.
- Prepare a precise calibrator that can generate precise 5 volt.
- Users can check the status text to know if the calibration is successful or not.
- The analog input field configuration is as follows.

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

Start to Calibrate

Press the “Calibration button” in the HSL-AI16AO2 utility to display the following dialog.



1. Connect AI channel 1 with AGND. The AI channel index is from 0 to 15. Please remember it. After well wiring, please press the first button, “AI Offset Calibration”.
2. Connect AI channel 0 into calibrator with a precise 5 volts. Then, press "AI Span Calibration" button to execute AI span calibration.
3. Connect AI channel 12 with AO channel 0 and AI channel 14 with AO channel 1. Then, press "AO Offset Calibration" button to execute AO offset calibration.
4. If step 3 is successful, just press “AO Gain Calibration” to finish the calibration.

After step 1 to 4 is executed successfully, the module is ready to use. If not, please check the wiring and calibrator. Follow the procedures to do again.

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 two-year guarantee:
   - The warranty period starts from the product's shipment date 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 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 two-year 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 technicians.
   - Products with altered and/or damaged serial numbers are not entitled to our service.
   - 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.