HSL System

User’s Guide

Recycled Paper

Manual Rev. 1.00: March. 26, 2001

Part No : 50-12100-200

Getting service from ADLINK

♦Customer Satisfaction is always the most important thing for ADLINK Tech Inc. If you need any help or service, please contact us and get it.

ADLINK Technology Inc.			
Web Site	http://www.adlink.com.tw		
	http://www.adlinktechnology.com		
Sales & Service	service@adlink.com.tw		
Technical Support	NuDAQ + USBDAQ	nudaq@adlink.com.tw	
	NuDAM, HSL	nudam@adlink.com.tw	
	NuIPC	nuipc@adlink.com.tw	
	NuPRO	nupro@adlink.com.tw	
	Software	sw@adlink.com.tw	
	AMB	Amb@adlink.com.tw	
TEL	+886-2-82265877	FAX	+886-2-82265717
Address	9F, No. 166, Jian Yi Road, Chungho City, Taipei, 235 Taiwan, R.O.C.		

♦Please inform or FAX us of your detailed information for a prompt, satisfactory and constant service.

Detailed Company Information			
Company/Organization			
Contact Person			
E-mail Address			
Address			
Country			
TEL		FAX	
Web Site			

Questions		
Product Model		
Environment to Use	☐OS: ________ ☐Computer ______________ ☐M/B: ☐Chipset: ☐Video Card: ☐Network Interface Card: ☐Other:	 ☐CPU: ☐BIOS:
Challenge Description		
Suggestions for ADLINK		

Table of Contents

How to Use This Guide

This manual is written to help you to use the HSL series products. It describes the versatile functions and the operation theorem of the HSL. This manual is divided into 3 chapters:

Chapter 1, "HSL Introduction" gives an overview of the HSL system regarding system features, applications, and specifications.

Chapter 2, "HSL Master Controller" describes detail information of HSL master.

Chapter 3, "HSL Slave Module" describes detail information of HSL slave modules.

Chapter 4, "NT DLL Programming" describes installation procedures of DLL driver and functions supported.

Chapter 5, "HSL Products Selection guides" tell you how to find out what you want.

1

HSL Introduction

1.1 What Is 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 mater-slave architecture. The HSL master is a PCI-bus add-on card plugged into PCI slot of computer system, and by using commercial Ethernet cable with RJ45 connector, users may easily settle the HSL slaves modules as close as possible to sensor devices. Thus the wiring effort is dramatically reduced.

This product series is to satisfy those who want:

- ♦ **Remote control based on PC architecture**
- ♦ **Simple wiring solution for remote I/O**
- ♦ **A vast, up to hundreds or more, number of I/O**
- ♦ **Real-time sensitive I/O**
- ♦ **High speed data acquisition**

1.1.1 Product Overview

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

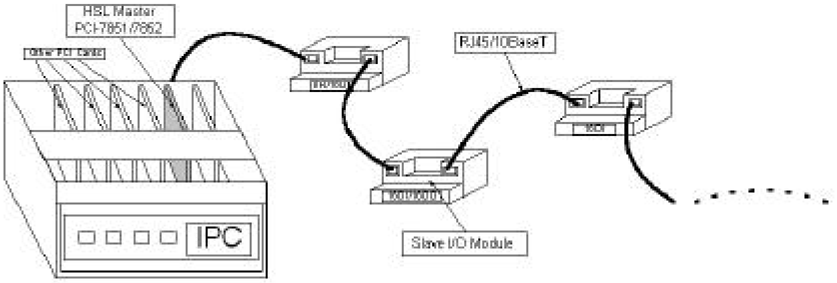


Figure 1.1 HSL topology

1.1.2 Features of HSL System

- **High speed**

It takes at most 1.895ms for a HSL master to scan all the I/O points of slave modules. 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 I/O points could be scanned (or updated) in 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 to scan all slave I/O modules is deterministic. It is exactly proportional to the number of slave I/O modules. Every 30.1μs is added for one more slave module. For a HSL system with 30 slave I/O modules, the scanning time period is precisely 30 X 30.1μs = 0.903 ms.

- **Easy wiring**

The connection among the HSL master and all slave I/O modules requires merely the 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.

♦ **Huge number of I/O points**

There are at most 63 slave I/O modules with one HSL master, and there are at most 16 HSL master (8 pieces of PCI-7852 or 8 pieces of PCI-7852) in one computer system. That is 63X16=1008 modules. Just choose all modules as HSL-DI16DO16-DB-NN, and the total number of I/O points is 16,128 DI and 16,128 DO.

♦ **Easy I/O expansion**

To expand I/O points of add-on card needs not only the card itself but also a free PCI or ISA slot. The nightmare is that there are still I/O points needed when no more free slot exists. To lift the constraint of add-in-like I/O, just choose HSL. What you need to expand I/O points is one more slave I/O module and an Ethernet cable with RJ45 for communication link.

♦ **Self-diagnostic function**

For those who concern the communication failure, the HSL provides an answer. Once power 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. Still the CRC12 is another proven to eliminate any communication error.

♦ **Modular design of slave I/O**

All of the slave I/O modules are mechanically of the same size (except –C modules) and follow the identical connecting address ID assignment. To fit into various wiring situations and keep flexibility, the HSL Terminal Base (TB) is introduced. HSL TB acts as a carrier of salve I/O module with wiring function. The Ethernet connector and screw terminal on HSL TB makes it easier for users to replace I/O module without power off and wire off when necessary.

♦ **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 card takes charge of communication with all remote slave I/O modules at constant scanning period and keeps the most updated I/O status information in the SRAM. What users do is to read and write the data in the 32K SRAM on HSL master card through PCI bus. So, no protocol is needed.

1.1.3 Application of HSL

A. HSL as Distributed PLC

First we review the evolution history of distributed PLC:
In the past decades, PLC had token an important character in the field of industry automation. With the help of communication modules, such as RS232, RS485, PLC could also perform distributed control. The traditional architecture of distributed PLC application is showed at Figure 1.2, in which the MPC(Monitoring PC) take the character of medium for data transmission from field to MIS.

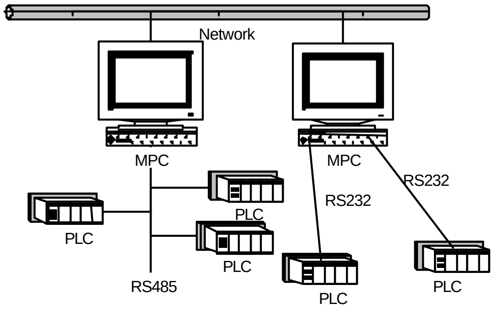


Figure 1.2 Traditional architecture of distributed PLC

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

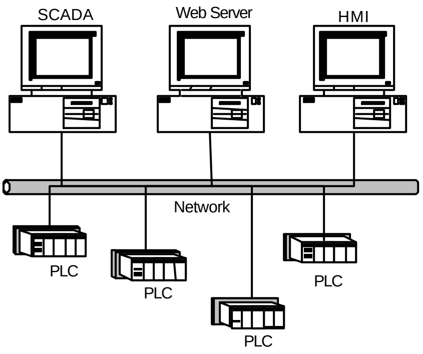


Figure 1.3 Networking PLC

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

With HSL, the distributed control architecture can become the figure 1.4. User does not need one extra PC for Ethernet communication. User can use one IPC to control the whole system.

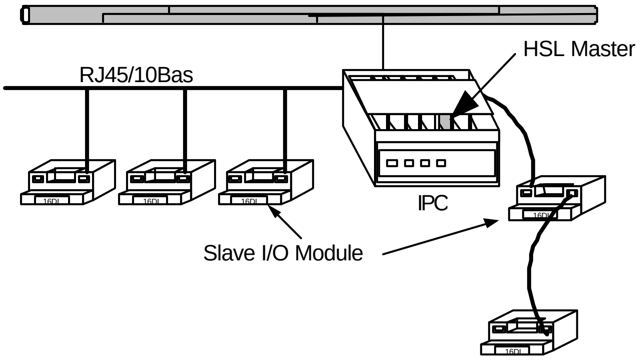


Figure 1.4 HSL as distributed PLC

Compare Figure 1.2, and Figure 1.4

- The MPC is replace by PC with HSL Master.
- The remote side PLC is replaced by HSL slave I/O module.
- The RS485 or RS232 cable is replaced by simple Ethernet cable.

- The protocol handling is replaced by simple memory read/write.

♦ HSL as Remote Real-time DAQ

HSL system gets many beneficial features as described on sections before. Among those features, two are worthy of being particularly notified.

- High Speed
- Real time

To implement a DAQ application, the real-time characteristic is the most important issue. With HSL system, all I/O data are time-deterministic refreshed. The sampling rate (or scan rate) is linearly dependent on the number of slave I/O modulesused, ranges from 90µs (less than 3 modules) to 2 ms (63 modules). These two features with HSL's born-nature remoteness make HSL very suitable for remote DAQ applications, especially when real-time is concerned.

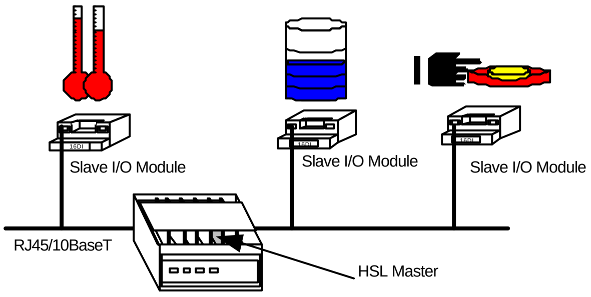


Figure 1.5 HSL as real-time DAQ

♦ HSL for SCADA system

HSL is suitable for SCADA (supervisory control and data acquisition) system. The character of HSL in a SCADA system is DAQ hardware but HMI toolkit. Users still need some SCADA software to accomplish the completely SCADA function. The reason why HSL is suitable for SCADA is described below:

- HSL is based on an open Architecture of PC.
- HSL is able to support Mass I/O points.
- HSL is capable of real-time remote DAQ

The PC-based characteristic of HSL system opens a highway to the "e" world. User just set up his 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-bus.

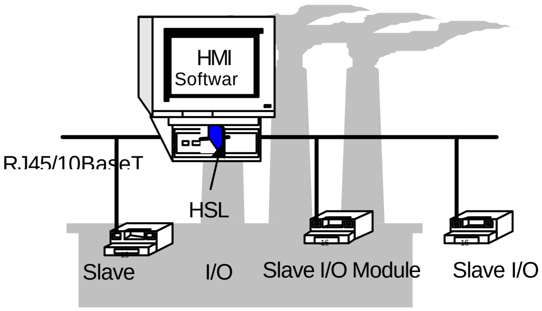


Figure 1.6 HSL for SCADA

1.2 Specifications of HSL System

The detail specifications of HSL system are listed as following:

- ♦ **Platform:**

  Computer hardware: Standard **PC** or **IPC** with **PCI-Bus**

  OS: **Windows 95/98/2000/NT** or **Linux**

- ♦ **Programming tool:**

  Any Windows programming language that could integrates DLL; DDE; Server; LabVIEW; ISaGRAF (IEC1131-3 PLC standard).

- ♦ **HSL Master Interface Card:**

  PCI-7851: HSL master controller card

  PCI-7852: Dual HSL master controller card

- ♦ **Remoteness:**

  One master has two HSL ports

  One port can drive maximum 32 modules

  One master can control maximum 63 slave I/O modules

  Maximum wiring distance for each port: 200m (serial wiring from master to last slave module)

- ♦ **Wiring:**

  Connector: **RJ45** (on both master controller and slave modules)

  Cable: Cat.5 100 Base/TX Ethernet cable, shielded one is better

- ♦ **Communication:**

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

  Data Rate: **6Mbps**

  I/O refresh rate: 30.1 µs x numbers of slave I/O modules (min: 3; max: 63)

  Communication model: single-master/multi-slave
  Communication method: command – response
  CRC12 and dedicate protocol for eliminating any communication error

1.3 HSL Series Products

♦ **HSL Master controller cards**

There are two Master cards supported:

- **PCI-7851**: HSL master controller card.
- **PCI-7852**: Dual HSL master controller card.

At least one master controller card is necessary for HSL system. With PCI-7852, two master controllers are available on one card. 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.

Model Numbers	Discrete Input	Discrete Output	Address ID Occupied
HSL-DI32-DB	32		2
HSL-DO32-DB		32	2
HSL-R8DI16-DB	16	8 relay	1
HSL-DI16DO16-DB	16	16	1
HSL-DI32-M	32		2
HSL-DO32-M		32	2
HSL-R8DI16-M	16	8 relay	1
HSL-DI16DO16-M	16	16	1
HSL-DI8DO8-C	8	8	1

Table 1 Slave I/O modules

More slave I/O modules and intelligent master with different interface will be available soon.

♦ **Terminal Base**

A variety of HSL terminal base are available. Here are the lists.

Model Numbers	Module Type Support	Module Number Support
HSL-TB64-DIN	All the HSL-DB	2
HSL-TB32-DIN	HSL-DI16DO16-DB-NN/PN; HSL-DO32-DB-N	1
HSL-TB32-U-DIN	All the HSL-DB	1
HSL-TB32-DO-DIN	HSL-DO32-DB-N	1
HSL-TB32-M-DIN	All the HSL-M	1
HSL-TB16-C	All the HSL-C	1

Table 2 Terminal base

More terminal base will be available soon and OEM/ODM of any kinds are welcome.

Please always contact with ADLINK for newest HSL series product information.

1.4 HSL Technical Information

1.4.1 HSL Technology Brief

HSL is a single master multi–slave command–response communication system. The master sends commands to slave I/O modules for setting output values and requesting input information. Every slave responses when receiving commands with address ID of its. 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**.

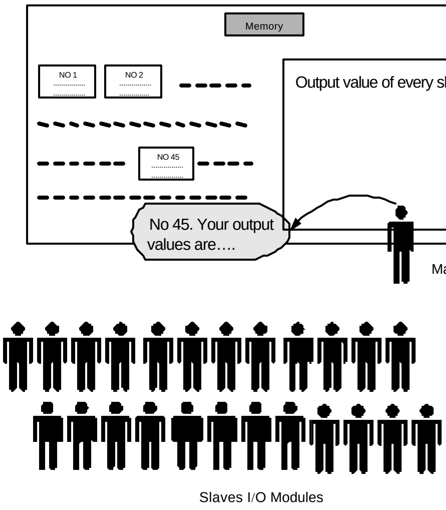


Figure 1.8 HSL technology brief -1

The only one teacher(the master) sends 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 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**.

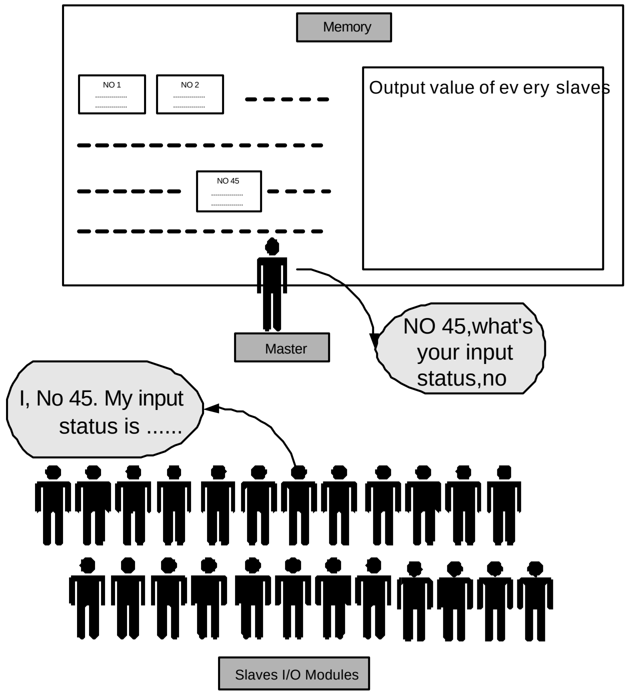

Memory
NO 1
NO 2
NO 45
Output value of ev ery slaves
Master
NO 45,what's your input status,no
I, No 45. My input status is ......
Slaves I/O Modules

Figure 1.9 HSL technology brief-2

The only one 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 some slave’s newest 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: NO1, your output vales are ##, what's your newest input status?

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

Teacher: NO2, 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: NO63, 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)

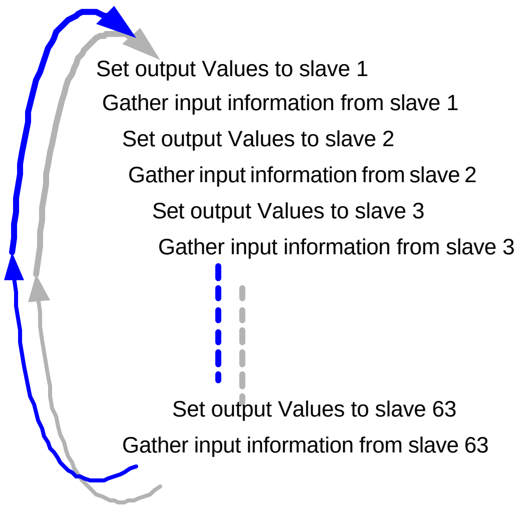

Set output Values to slave 1
Gather input information from slave 1
Set output Values to slave 2
Gather input information from slave 2
Set output Values to slave 3
Gather input information from slave 3
Set output Values to slave 63
Gather input information from slave 63

Figure 1.10 HSL I/O polling cycle

1.4.2 Terminology of HSL

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 Figure 1.9 & 1.10. 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 Figure 1.9 & 1.10. 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. Each slave I/O module might take 1 or 2 address ID depend 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 is 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, it could be defined as the longest time needed for any digital I/O channel to get it’s 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.

Data Rate: 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 Data Rate is bps, Bits Per Second, but unit of I/O refreshing rate is ms, mini-second.

1.4.3 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 ID: The master card is a PCI bus add-on card with HSL master on it. In one IPC system, there may be more than one Master card. You need to give the card number for your programming. The card number is assigned by PCI BIOS.

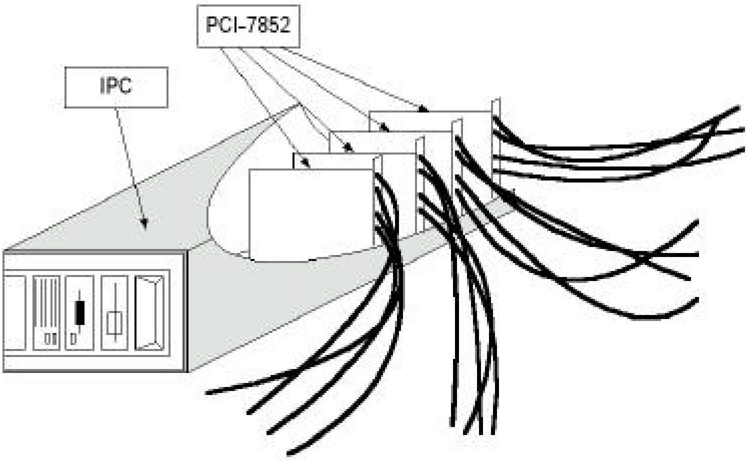


Figure 1.11 Multiple master cards in one IPC

Please refer to Chapter 2 "HSL Master Controller" to get advanced information.

HSL Set ID: There are two kind of HSL master card supported by ADLINK, PCI-7851& PCI-7852. The PCI-7851 supports 1 HSL master, while the PCI-7852 supports 2 HSL masters. In 7851, the Set ID of the only HSL set is '0'. In 7852, the Set ID is '0' and '1' to distinguish the two HSL master

Ports: A port is a loop of wiring that starts from the master card and connects at most 32 slave I/O modules. There are two ports in one HSL master, and the two ports will carry the same signals sent from master. (For detail description of circuit, please refer to chapter 2: HSL Master Controller)

Slave Number: A complete HSL system must be composed of one master and 1~63 slaves. In the following graph, a layout example of HSL system is presented.

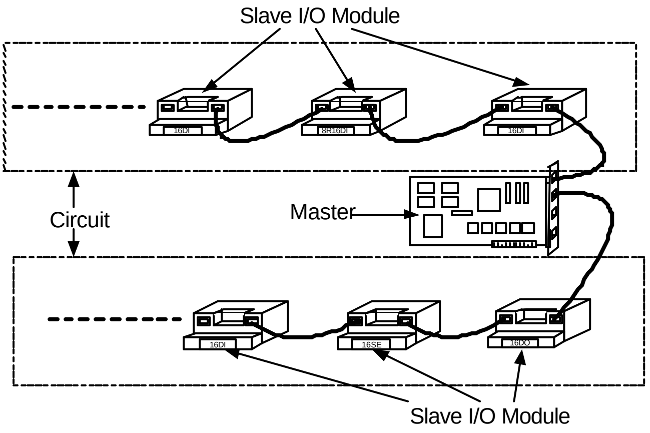


Figure 1.12 HSL system layout example-serial wiring

Each circuit from master could 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 addresses ID.** These addresses ID aren't necessary to be continuous from 1. However, continuous addressing will be more efficient. Actually, if some user 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 HSL system.

Please refer to chapter 3 "HSL-Slave I/O Module" for completed descriptions regarding how to set addresses of slave I/O modules.

1.4.4 Wiring

The HSL network follows RS422 electrical spec, 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 RS422 with Multi-drop**

Typical RS422 is not actually a networkingspec. We do make some change. 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) → RXD(of slave) channel, and ,by well designed, only one slave at one time sends message with TXD(of slave)→ RXD(of master) channel. So, 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) → RXD(of slave) channels are actually of the same signal sent by master, but the TXD(of slave)→ RXD(of master) channels are not that case. TXD(of slave)→ 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:

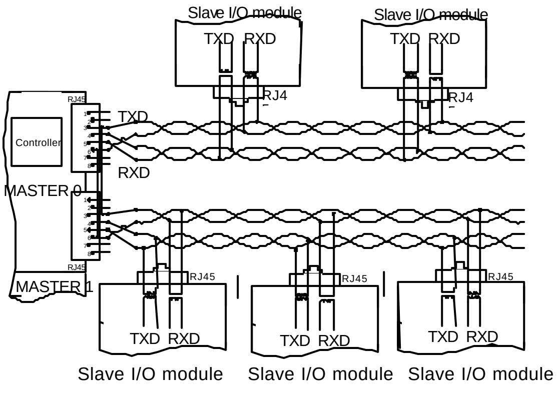


Figure 1.13 HSL wiring – RS422 with multi-drop

- There are two RJ45 notches in each master. Each notch supports one port of wiring.
- Only 4 of 100BaseTX cable’s 8 lines 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.

1.4.5 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.

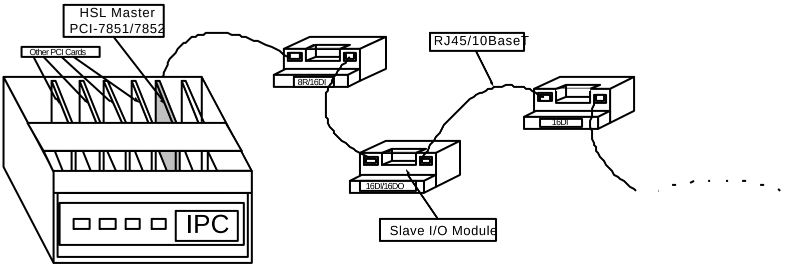


Figure 1.14 HSL networking topology – Serial

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.

1.4.6 I/O refreshing rate of HSL system

Once the maximum slave address used in one HSL system is decide. The polling cycle time of this HSL system will be known, also. Here is the formula:

Polling cycle time = maximum-address-of-slave * 30.1 μ sec.

Maximum address	Polling time
5	150.5 μ sec.
10	301.0 μ sec.
20	602.0 μ sec.
30	903.0 μ sec
40	1.204 m sec
50	1.505 m sec
60	1.806 m sec
63	1.895 m sec

Table 2 Polling cycle time of HSL

Note : The minimum polling cycle time is 90.3μsec, even the maximum address is less than 3.

1.4.7 Communication error handling

Though the HSL communication protocol is dedicatedly designed to avoid any error, there is inevitably still some chance that communication error may occur, e.g.: light striking, suddenly off-line… etc. If this is the case, never be too worried, 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 send IRQ to trigger this driver to check every slave's*no-response count value.* If the count value is '7', the drive will inform system, by event, that some slave I/O is in mal-communication situation.

1.5 Software Supporting

- **Window 95/98/2000/NT DLL**

The HSL Windows 95/98/2000/NT 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++(4.0 or above), Borland C++(5.0 or above), or Microsoft Visual Basic (4.0 or above), etc.

- **Linux Driver**

The HSL Linux Driver Include device drivers and 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 CD.

- **DDE Server**

DDE stands for Dynamic Data Exchange specifications. The HSL-DDE includes the HSL system DDE server. The HSL-DDE server is included in the ADLINK CD. It is free shipped with the board. The DDE server can be used conjunction with any DDE client under Windows NT/2000.

- **LabVIEW Driver**

HSL-LVIEW contains the Vis, which are used to interface with NI's LabVIEW® software package. The HSL-LVIEW supports Windows NT/2000. The LabVIEW® drivers are free shipped with the board. You can install and use them without license. For detail information about HSL-LVIEW, please refer to the ADLINK CD.

- **ISaGRAF Driver**

The ISaGRAF WorkBench is an IEC1131-3 SoftPLC control program development environment. The HSL-ISG includes ADLink products' target drivers for ISaGRAF under Windows NT environment. The HSL-ISG is included in the ADLINK CD. It needs license.

- **OPC Server**

HSL-OPC is an OPC Server, which can link with the OPC clients. There are many software packages on the market can provide the OPC clients now. The HSL-OPC supports the Windows 95/98/NT/2000. The HSL-OPC is included in the ADLINK CD. It needs license.

1.6 How to Install a HSL System

To successfully setup a HSL system, the following steps are some guidelines.

1. Analyze your specific application. How many I/O points and which I/O type?

2. Select suitable slave I/O modules and quantities for your application.

3. Select HSL master cards:

Total number of address ID Occupied	Recommended HSL master card	Number of HSL Master
1 ~ 63	PCI-7851	1
64 ~ 126	PCI-7852	2
127 ~ 189	One PCI-7852 + one PCI-7851	3
190 or more	Optional combination of PCI-7852 & PCI-7851	4 or more

4. Make order from local distributor or ADLINK.

5. Install HSL master into PCI-bus PC or IPC. Please read carefully the installation procedures of HSL master.

6. Setup slave modules on remote side. Please read carefully the PIN OUT information of individual I/O modules. Set the dip switch for individual slave address ID.

7. Connect master and slaves by RJ45/100Base/TX cable. Note: Do not mix different sets of HSL system.

8. Connect the power distribution of slave modules.

9. Testing and debugging.

10. User's application developing.

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 output values to and gather input information from slaves by communication.

There are two kind of HSL master card supplied by ADLINK: PCI-7851, PCI-7852. The difference between these two is the number of HSL master supported.

✓ PCI-7851: High Speed Link Master Controller Interface Card
✓ PCI-7852: Dual High Speed Link Master Controller Interface Card

2.1 Board Overview

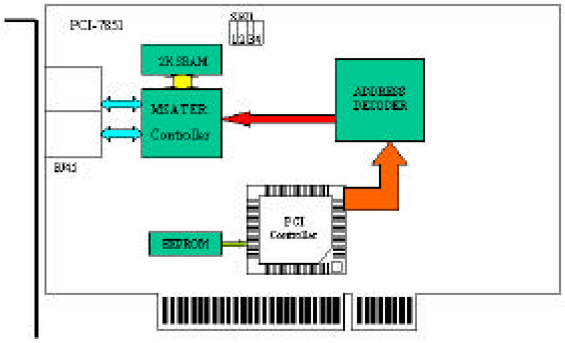
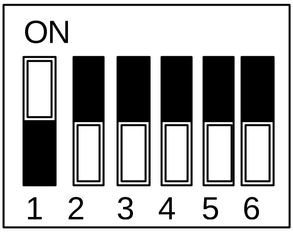


Figure 2.1 PCI-7851 front view

2.2 Specifications

- ♦ **PCI Controller:**
  - PCI local bus specification Rev. 2.1 compliance
- ♦ **Master Controller:**
  - Master controller: ASIC
  - External Clock: 48MHz
- ♦ **Memory:**
  - 32KB SRAM – 12ns
- ♦ **Interface:**
  - RS-422 with transformer isolation
  - Full duplex communication
  - Selectable transfer speed by dip switch (Default 6Mbps)
  - Two ports for one control master
- ♦ **Connector:**
  - RJ45 connector x 2 (7851)
  - RJ45 connector x 4 (7852)
- ♦ **Interrupt:**
  - 32 bits Programmable timer
- ♦ **LED Indicator:**
  - Power status
- ♦ **PCB Dimension:**
  - 176 (L) x 107 (W) mm
- ♦ **Operating Temperature:** 0 to 60℃
- ♦ **Storage Temperature:** -20 to 80℃
- ♦ **Power Consumption:** +5V @500 mA typical

2.3 Configuration

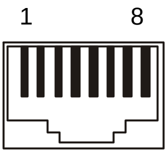


Sw1 Setting – Baud rate set

NO	Set baud rate				
1	OFF	ON	OFF	ON	Master 1
2	OFF	OFF	ON	ON	
3	OFF	ON	OFF	ON	Master 2
4	OFF	OFF	ON	ON	
Bps	12M	6M	3M	EXC	

The default is 6M.

2.4 PIN Assignment CN1



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

2.5 Function Description

PCI-7851/7852 is equipped with one/two HSL masters 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 doing that, each HSL master manages a 32K byte SRAM on 7851/7852 boards to keep 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 mine I/O information for users.

Functional Block Diagram

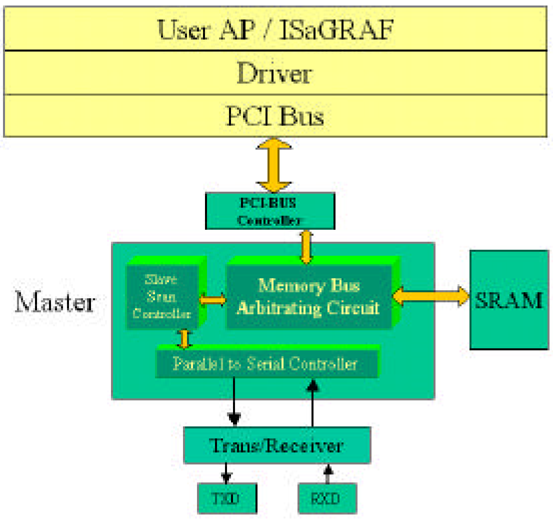


Figure 2.2 Functional Block diagram of HSL master

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

2.6 Installation

Hardware Configuration

The PCI-7851/PCI-7852 is a plug and play device, the interrupt number and memory address mapping are assigned by system BIOS. You only need to configure the SW1 for communication transfer speed (baud rate) setting.

Installation Procedure:

1. Turn off your computer.
2. Turn off all the accessories (printer, modem, monitor, etc.) connected to the computer.
3. Remove the cover from your computer.
4. Select a 32-bit PCI expansion slot.
5. Before handling the PCI-7851/PCI-7852, discharge any static buildup on your body by touching the metal case of the computer. Hold the edge and do not touch the components.
6. Position the PCI-7851/PCI-7852 board into the PCI slot you selected.
7. Secure the PCI-7851/7852 in place at the rear panel of the system unit using screw removed from the slot.

3

HSL Slave Module

The HSL is a modular system which features an innovative distributed architecture that modularizes the communication, I/O functions and signal termination. You can independently choose the slave I/O modules and terminal base to best your particular applications requirement.

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

There are three groups of the slave I/O modules:

- -DB: Daughter board form factor (100mm x 78.2mm)
- -M: Daughter board with aluminum cover (125mm x 80mm)
- -C: Compact sized slave I/O module (48mm x 72mm)

The slave I/O module of the daughter board form factor series include:

- HSL-DI16DO16-DB-NN
- HSL-DI16DO16-DB-NP
- HSL-DI16DO16-DB-PN
- HSL-DI16DO16-DB-PP
- HSL-DI32-DB-N
- HSL-DI32-DB-P
- HSL-DO32-DB-N
- HSL-DO32-DB-P
- HSL-R8DI16-DB-N
- HSL-R8DI16-DB-P

The slave I/O module of the daughter board with aluminum cover series include:

- HSL-DI16DO16-M-NN
- HSL-DI16DO16-M-NP
- HSL-DI16DO16-M-PN
- HSL-DI16DO16-M-PP
- HSL-DI32-M-N
- HSL-DI32-M-P
- HSL-DO32-M-N
- HSL-DO32-M-P

- HSL-R8DI16-M-N
- HSL-R8DI16-M-P

The slave I/O module of the compact sized slave I/O module series include:

- HSL-DI8DO8-C-NN
- HSL-DI8DO8-C-NP
- HSL-DI8DO8-C-PN
- HSL-DI8DO8-C-PP

3.1.1 Slave I/O Module Numbering Guide:

HSL	DI16DO16	DB	NN
↓	↓	↓	↓
Product Code: HSL: High Speed Link System	***Product Line:*** DI16DO16: 16 discrete inputs and 16 discrete outputs DI32: 32 discrete inputs DO32: 32 discrete outputs R8DI16: 8 relay outputs and 16 discrete inputs DI8DO8: 8 discrete inputs and 8 discrete outputs	***Form Factor:*** DB: Daughter board form factor M: Daughter board with aluminum cover C: Compact sized slave I/O module	***I/O Type:*** First digit: Input type, 'N' means NPN sinking type sensor input, 'P' means PNP sourcing type sensor input Second digit: output type, 'N' means NPN type sinking output, 'P' means PNP type sourcing output

3.1.2 General Specifications:

- **Discrete Input:**

  Input current (max.):

  - -10mA (NPN sinking type sensor input modules)
  - +10mA (PNP sourcing type sensor input modules)

  Input impedance: 4.7K ohm

  Operation voltage (with 24Vdc on V+):

  - ON: 9.6Vdc max. OFF: 19.0Vdc min.
    (NPN sinking type sensor input modules)
  - ON: 14.4Vdc max. OFF: 5.0Vdc min
    (PNP sourcing type sensor input modules)

  Response time:

  - ON: 5µs typical.
  - OFF: 10µs typical.

- **Transistor Output:**

  Switching capacity (max.):

  - Single channel–500mA; all channels-60mA at 24Vdc (NPN sinking type output modules)
  - Single channel +500mA; all channels +60mA at 24Vdc (PNP sourcing type output modules)

  Response time:

  - ON: 1.2µs typical.
  - OFF: 180µs typical.

- **Relay Output:**

  Channel number: 8

  Relay type: SPST, normally open, non-latching

  Switch frequency (max.): 20 cpm (@3A, 250Vac)

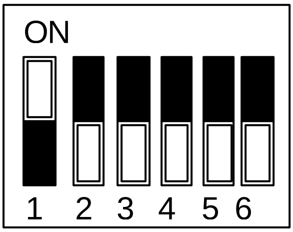
  Switching capacity (max.): 3A/250Vac; 3A/30Vdc

Response time:

- Operate: 6ms max.
- Release: 7ms max.

♦ **External Power requirement:** 10 ~ 30Vdc

3.1.3 DIP Switch Setting:



ON = 1

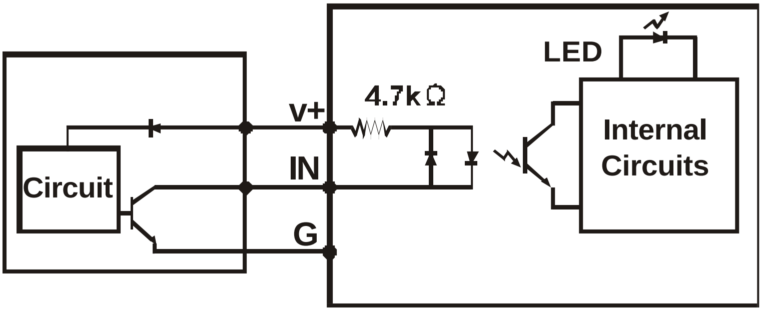
100000	address 1
010000	address 2
…	…
011111	address 62
111111	address 63

OFF = 0

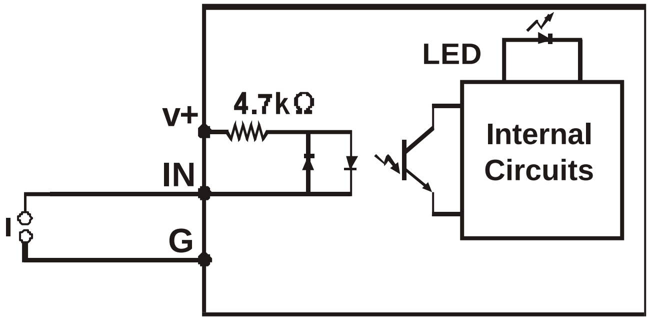
Note that address '0' is reserved and each HSL-DI32 or HSL-DO32 needs two consecutive addresses that start from an odd number.

3.1.4 Wiring Diagram:

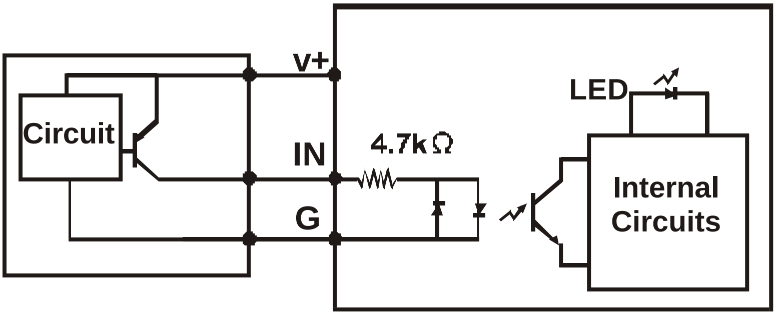
-N NPN Sinking type sensor Input



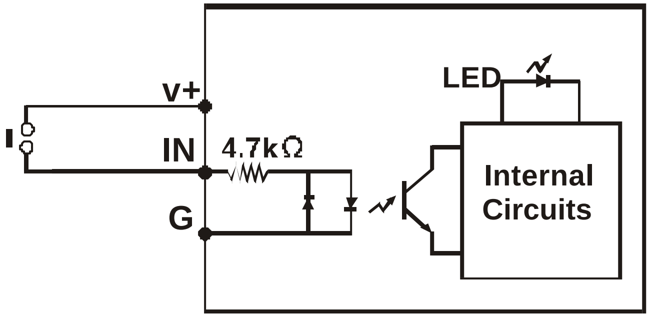
-N Dry Contact Input



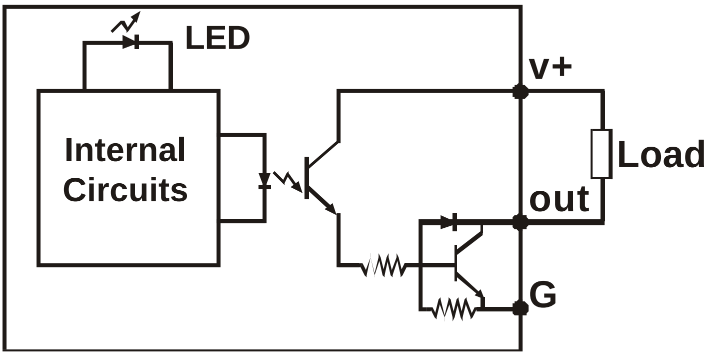
-P PNP Sourcing type sensor Input



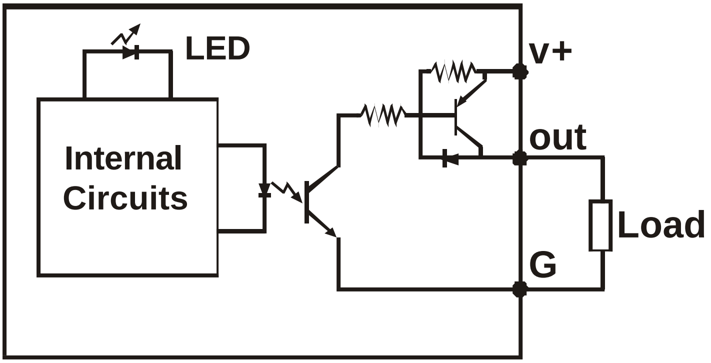
-P Wet Contact Input



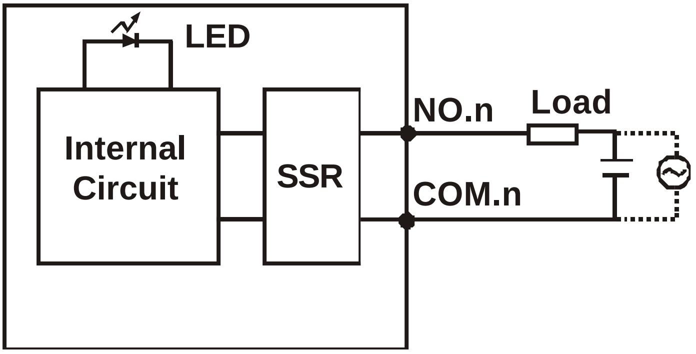
-N NPN Sinking Output



-P PNP Sourcing Output

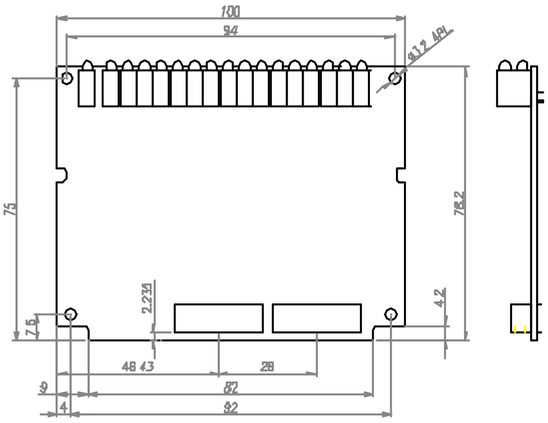


-R Relay Output

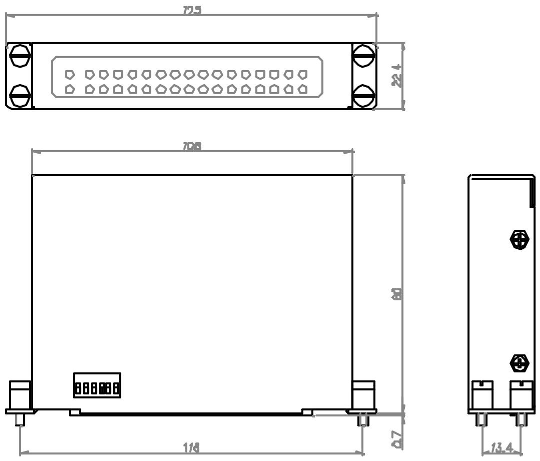


3.1.5 Dimension:

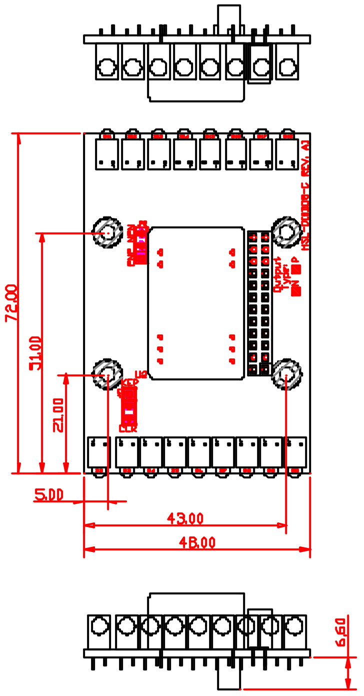
-DB Daughter board form factor (100mm X 78.2mm)



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



-C Compact sized slave I/O module (48mm X 72mm)



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
- HSL-TB16-C

3.2.1 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

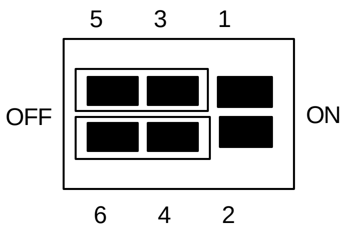
3.2.2 General Description

	Model Name	**Description**	**Module Support**
Standard	HSL-TB32-U	32 channels direct connected terminal base	All HSL –DB modules
	HSL-TB64	64 channels direct connected terminal base	All HSL –DB modules
	HSL-TB32	16 channels short circuit protected current driver (up to 300mA/ch) and 16 channels direct connected terminal base	HSL-DI16DO16-DB-NN HSL-DI16DO16-DB-PN HSL-DO32-DB-N
	HSL-TB32-DO	32 channels short circuit protected current driver (up to 300mA/ch) terminal base	HSL-DO32-DB-N
With Cover	HSL-TB32-M	32 channels direct connected terminal base for HSL – M modules	All HSL –M modules
Compact	HSL-TB16-C	16 channels direct connected terminal base for HSL – C modules	All HSL –C modules

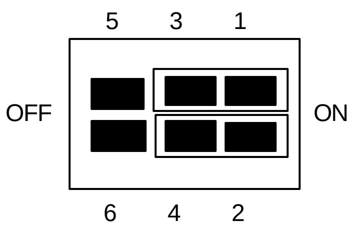
3.2.3 Jumper Setting

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.

Not the last module (Default)

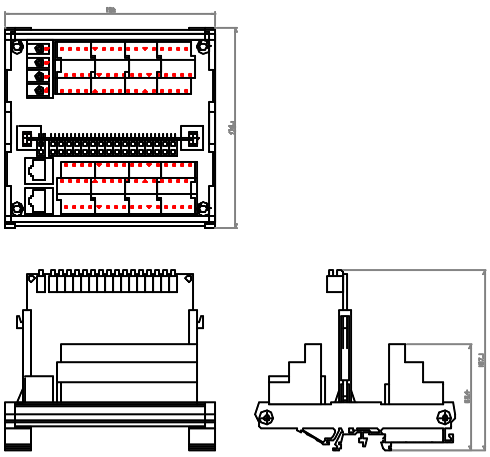


The last module

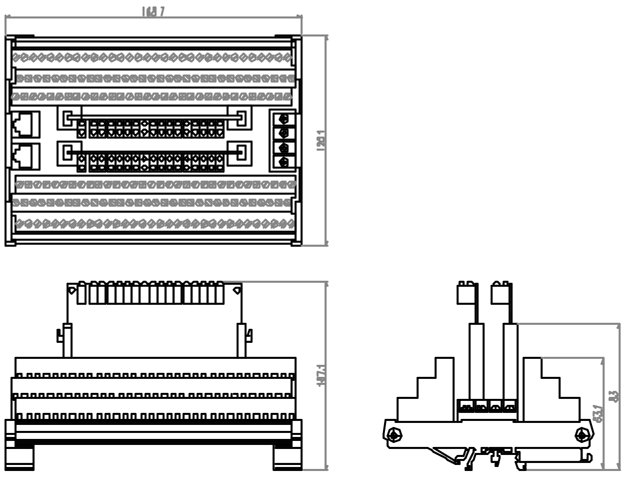


3.2.4 Dimension

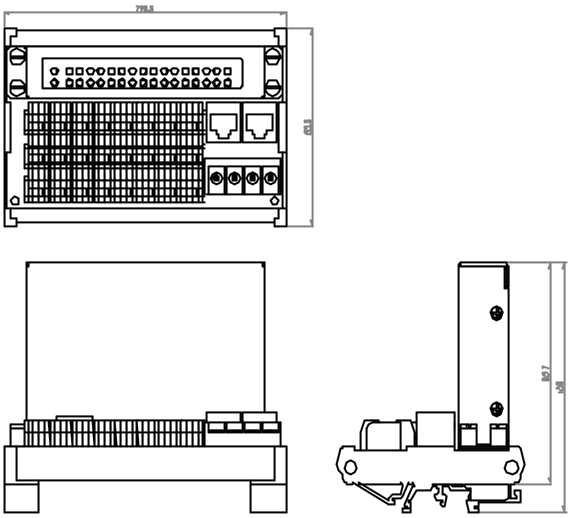
-DB with HSL-TB32-DIN or HSL-TB32-U-DIN (126x120.1x107.3) mm



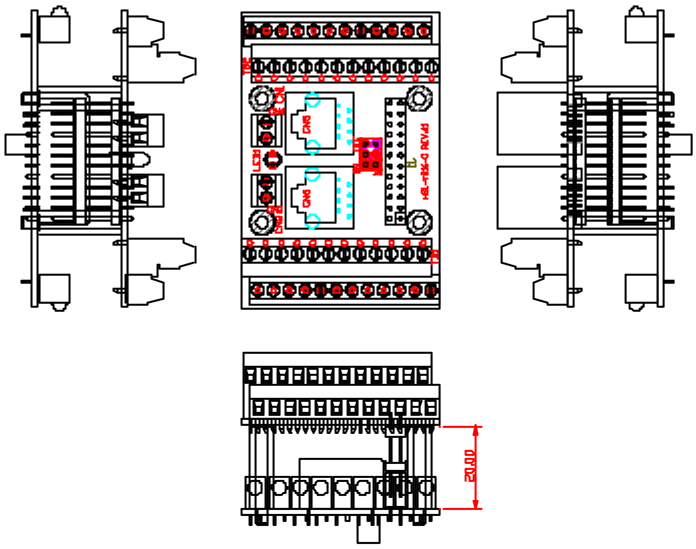
-DB with HSL-TB64-DIN (168.7x120.1x107.3) mm



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



-C module with HSL-TB16-C (48x72x65) mm



4

HSL Library Programming

4.1 Installation of HSL DLL Driver

- **System Requirements**

  HSL Windows DLL requires the following minimum requirement:

- **An IBM PC/AT or compatible system, running Windows NT/95/98 2000.**
- **A hard disk with enough disk space to install HSL Windows NT DLL, about 1M.**
- **A 1.44 MB 3.5'' floppy disk drive or a CD-ROM drive.**
- **Application development system: Any Windows programming language that allows to call DLL, such as Microsoft Visual C/C++, Microsoft Visual Basic, etc.**
- **HSL master cards and slave modules.**
- **Installation procedures**

With "ADLINK All-In-One Compact Disc":

Step 1. Place "ADLINK All-In-One Compact Disc" in the CD-ROM drive.

Step 2. If auto run setup program is not invoked, execute x:\setup.exe (x: indicates the CD-ROM drive).

Step 3. Select “Driver Installation”→“HSL”→“PCI-7851”→ “WinNT/2000 Driver” or “Win95/98 Driver” to install the DLL and driver.

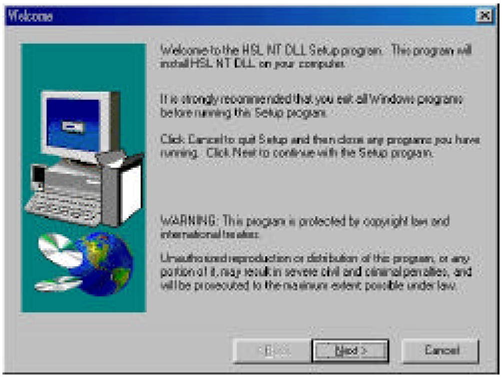
With “NuDAM-HSL Name Card Size Compact Disc”:

Step 1. Place “NuDAM-HSL Name Card Size Compact Disc” in the CD-ROM drive.

Step 2. If autorun setup program is not invoked, execute x:\setup.exe (x indicates the CD-ROM drive).

Step 3. Select “HSL”→ “Software Installation” → “WinNT/2000 Driver” or “Win95/98 Driver” to install the DLL and driver.

Setup first displays a Welcome dialog box. Please click “Next” button to go to the next step. Then, the Setup wizard prompts the following dialog box for you to specify the destination directory for HSL Windows DLL. The default path is c:\ADLINK\HSL.



Welcome to the HSL NT DLL Setup program. This program will install HSL NT DLL on your computer.

It is strongly recommended that you exit all Windows programs before running this Setup program.

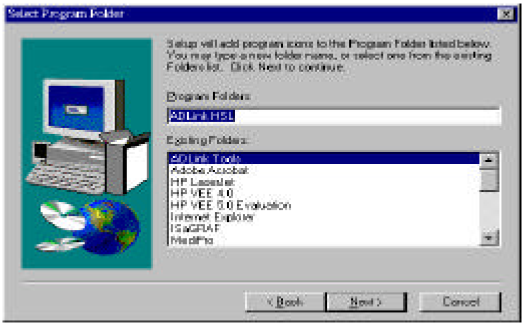
Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

If you want to install this DLL in another directory, please enter the directory you would like to install.

Then user can assign the “Program Folder”. The default “Program Folder” is “HSL”.


Select Program Folder

Setup will add program icons to the Program Folder listed below. You may type a new folder name, or select one from the existing Folders list. Click Next to continue.

Program Folders:

ADLink HSL

Existing Folders:

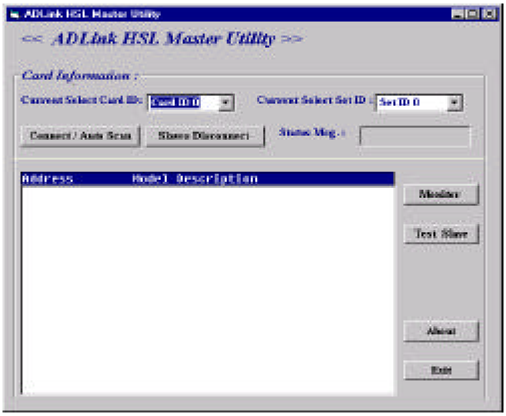
ADLink Tools
Adobe Acrobat
HP LaserJet
HP VEE 4.0
HP VEE 5.0 Evaluation
Internet Explorer
ISaGRAF
MediPro

After complete the installation processes, you have to restart Windows NT system, so that the HSL drivers could work normally.

4.2 ADLINK HSL Link Master Utility

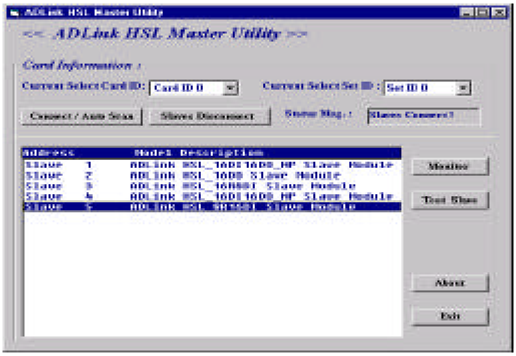
In the “ADLINK All-In-One Compact Disc” or “NuDAM-HSL Name Card Size Compact Disc”, ADLINK provide a HSL Link Master Utility. User can install the utility. This utility is a software diagnostic system. So user can use this utility to test his HSL system and slave I/O modules.

After run the ADLINK HSL Link Master Utility, It will show the main operation window as below.



About the operation items in the “ADLink HSL Master Utility”, they will be described as following.

- ♦ **“Current Select Card ID”:** This utility will detect all HSL master cards (include PCI-7851 and PCI-7852) in this computer. The total master cards will display in the **“Current Select Card ID”** item. User can use it to specify the master card he wants to operate.
- ♦ **“Current Select Set ID”:** When user have selected the master card, then the **“Current Select Set ID”** item will show the set ID number. For the PCI-7851 master card, the “Current Select Set ID” is “Set ID0”. For the PCI-7852 master card, the “Current Select Set ID” is “Set ID0” or “Set ID1”.
- ♦ **“Connect / Auto Scan”**: When user press this button, the utility will connect the master card and slave I/O modules, then scan all the slave I/O modules in the **“**Current Select Set ID**”** of the “Current Select Card ID”. This utility will show all the slave I/O module’s attributes (include the address and slave type) on the screen.


ADLink HSL Master Utility

<< ADLink HSL Master Utility >>

Card Information :

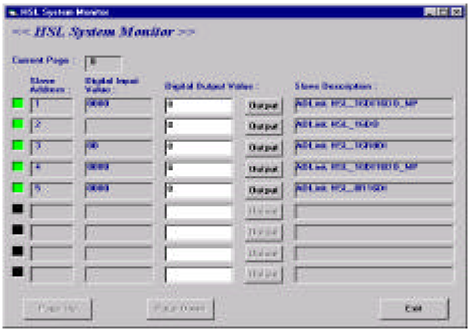
Current Select Card ID: Card ID 0 Current Select Set ID : Set ID 0

Status Msg. : Slaves Connect!

```
Address          Model Description
Slave    1       ADLink HSL_16DI16DO_HP Slave Module
Slave    2       ADLink HSL_16DO Slave Module
Slave    3       ADLink HSL_16R8DI Slave Module
Slave    4       ADLink HSL_16DI16DO_HP Slave Module
Slave    5       [illegible]
```

♦ **"Slaves Disconnect":** When user press this button, the utility will disconnect the master card and slave I/O modules.

♦ **"Monitor":** When user press **"Connect / Auto Scan"** button, all slave I/O modules in the **"Current Select Set ID"** of the **"Current Select Card ID"** will display in the screen. Now user can press the **"Monitor"** button to enter the "**HSL System Monitor**" window.

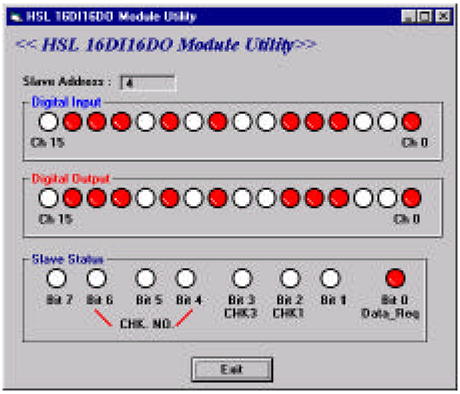

HSL System Monitor

<< HSL System Monitor >>

Current Page : 0

	Slave Address :	Digital Input Value :	Digital Output Value :		Slave Description :
🟩	1	0000	0	Output	ADLink HSL_16DI16DO_HP
🟩	2		0	Output	ADLink HSL_16DO
🟩	3	00	0	Output	ADLink HSL_16R8DI
🟩	4	0000	0	Output	ADLink HSL_16DI16DO_HP
🟩	5	0000	0	Output	[illegible]
⬛				Output	
⬛				Output	
⬛				Output	
⬛				Output	

♦ **"Test Slave":** When user press **"Connect / Auto Scan"** button, all slave I/O modules in the "Current Select Set ID" of the **"Current Select Card ID"** will display in the screen. Now user can select a slave I/O module in the screen, then press the **"Test Slave"** button to enter the Slave testing window (The following is a testing utility for HSL DI16DO16 module)



About the operation items in the "HSL System Monitor", they will be described as following.

- **"Page Up":** When user presses this button, the utility will display previous 9 slave I/O modules.
- **"Page Down":** When user presses this button, the utility will display next 9 slave I/O modules.
- **"Output":** When user key in the value in the "Digital Output Value" item, then h e have to press this button. The value will output to the specified slave I/O module.

4.3 List of Functions

Initialization

W_HSL_Initial(card_ID)	Interface card initialization
W_HSL_Close(card_ID)	Release card
W_HSL_Start(card_ID, set_ID, slave_No)	Start scan
W_HSL_Auto_Start(card_ID, set_ID)	Auto star scan
W_HSL_Stop(card_ID, set_ID)	Stop scan

System Information

W_HSL_Connect_Status(card_ID, set_ID, slave_No, sts_data)	Get slave module's status
W_HSL_Slave_Live(card_ID, set_ID, slave_No, live_data)	Check if the slave alive
W_HSL_Get_IRQ_Channel(card_ID, *irq_no)	Get IRQ channel

Digital I/O

W_HSL_DIO_In(card_ID, set_ID, slave_No, *in_data)	Digital input from all bits
W_HSL_DIO_Channel_In(card_ID, set_ID, slave_No, channel_No, *in_data)	Digital input from one bit
W_HSL_DIO_Out(card_ID, set_ID, slave_No, out_data)	Digital output for all bits
W_HSL_DIO_Channel_Out(card_ID, set_ID, slave_No, channel_No, out_data)	Digital output for one bit

System Reset

W_HSL_Software_Reset(card_ID)	Reset card

Timer

W_HSL_TMRINT_Enable(card_ID, HANDLE *phEvent)	Enable timer interrupt
W_HSL_TMRINT_Disable(card_ID)	Disable timer interrupt
W_HSL_Timer_Set(card_ID, c1, c2)	Set timer

Memory I/O

W_HSL_DIO_Memory_Out(card_ID, set_ID, unsigned short *data_out)	Set output values of all slave modules
W_HSL_DIO_Memory_In(card_ID, set_ID, unsigned short *data_in)	Get input values from all slave modules

4.4 The HSL Function Reference

4.4.1 Initial

- **Syntax**

  U16 _stdcall W_HSL_Initial(U16 card_ID);

- **Description**

  Initialize the hardware and software states of an HSL master card(PCI-7851 or PCI-7852), and then return a status that corresponds to the card initialized. W_HSL_Initial must be called before any other HSL DLL functions can be called for the card. The function initializes the card and variables internal to HSL DLL. Because HSL master card meets the plug-and-play design, the base address and IRQ level are assigned by BIOS directly.

- **Parameter**

card_ID	The sequence number of HSL master card (PCI-7851 or PCI-7852). This first master card (in the most prior slot) is with card_ID =0. For example, if there are two HSL master card plugged on your PC, the HSL master card in prior slot should be registered with card_ID = 0, the other HSL master card with card_ID = 1.

- **Return Number**

  ERR_NoError, ERR_OpenDriverFail, ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_Initial(0);

4.4.2 Close

- **Syntax**

  U16 _stdcall W_HSL_Close(U16 card_ID);

- **Description**

  This function is to close the HSL master card with card_ID. This function is used to tell library that this registered card is not used currently and can be released. By the end of a program, you need to use this function to release all cards that were registered.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.

- **Return Number**

  ERR_NoError,ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_Close(0);

4.4.3 Start

- **Syntax**

  U16 _stdcall W_HSL_Start(U16 card_ID, U16 set_ID, U16 slave_No);

- **Description**

  This function is used to set the total connected slave I/O module numbers of the HSL master card (PCI-7851 or PCI-7852) with set_ID and start to scan these slave I/O modules.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	The maximum slave numbers connect to the HSL master card by the set_ID set. The valid value is between 1 ~ 63.

- **Return Number**

  ERR_NoError,ERR_SatelliteNumber,ERR_ConnectIndex,ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_Start(cardNo, 0, 63);

4.4.4 Auto Start

- **Syntax**

  U16 _stdcall W_HSL_Auto_Start(U16 card_ID, U16 set_ID);

- **Description**

  This function is used to auto-detect the total connect slave I/O module numbers of the HSL master card (PCI-7851 or PCI-7852) with set_ID and start to scan these slave I/O modules.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI- 7852, the valid value is 0 or 1.

- **Return Number**

  ERR_NoError, ERR_ConnectIndex, ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_Auto_Start(cardNo, 0);

4.4.5 Stop

- **Syntax**

  U16 _stdcall W_HSL_Stop(U16 card_ID, U16 set_ID);

- **Description**

  This function stop to scan the connected slave I/O modules of the HSL master card (PCI-7851 or PCI-7852) with set_ID value.

- **Parameter**

card_ID	The card ID of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.

- **Return Number**

  ERR_NoError, ERR_ConnectIndex, ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_Stop(cardNo, 0);

4.4.6 Connect Status

- **Syntax**

U16 _stdcall W_HSL_Connect_Status(U16 card_ID, U16 set_ID, U16 slave_No, U8 sts_data);

- **Description**

This function is to read the communication status of the slave I/O module. The slave I/O module's address is slave_No and set value is set_ID.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
sts_data	The communication status of this slave I/O module. Bit 0 is Data_Req bit. Bit 2 is for CHK1. (If Bit2 is 1. It mean that there is 1 time communication error). Bit 3 is for CHK3. (If Bit3 is 1. It mean that there are 3 times communication error). Bit 4, BIT 5 and BIT 6 bits are for CHK7. (If Bit4, Bit5 and Bit6 all are 1. It mean that there are 7 times communication error).

- **Return Number**

ERR_NoError,ERR_InvalidBoardNumber,ERR_ConnectIndex,ERR_SatelliteNumber, ERR_NotADLinkSlaveType

- **Example**

```
result = W_HSL_Connect_Status (cardNo, 0, 1, &sts_data);
```

4.4.7 Slave Live

- **Syntax**

  U16 _stdcall W_HSL_Slave_Live(U16 card_ID, U16 set_ID, U16 slave_No, U8 live_data);

- **Description**

  This function is to read the module status of the slave I/O module (live or die). The slave I/O module's address is slave_No and set value is set_ID.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
live_data	The module status of this slave I/O module. 1: The module is live 0: The module is die

- **Return Number**

  ERR_NoError,ERR_InvalidBoardNumber,ERR_ConnectIndex,ERR_SatelliteNumber, ERR_NotADLinkSlaveType

- **Example**

  result = W_HSL_Slave_Live (cardNo, 0, 1, &live_data);

4.4.8 Get IRQ Channel

- **Syntax**

  void _stdcall W_HSL_Get_IRQ_Channel(U16 card_ID, U16 *irq_no);

- **Description**

  This function is to get the IRQ number of the HSL master card with cardNo

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
irq_no	The IRQ number of this card.

- **Example**

  W_HSL_Get_IRQ_Channel (0, &irq_no);

4.4.9 DIO In

♦ **Syntax**

U16 _stdcall W_HSL_DIO_In (U16 card_ID, U16 set_ID,U16 slave_No,U32* in_data);

♦ **Description**

This function is to read the digital input value of the slave I/O module. The module's address is slave_No and set value is set_ID.

♦ **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
in_data	the digital input data of this slave I/O module. In this value. Channel 0 data is assign to bit 0, channel 1 data is assign to bit 1….etc.

♦ **Return Number**

ERR_NoError,ERR_InvalidBoardNumber,ERR_SatelliteType,ERR_ConnectIndex,ERR_SatelliteNumber,ERR_NotADLinkSlaveType

♦ **Example**

result = W_HSL_DIO_In (cardNo, 0, 1, &in_data);

4.4.10 DIO Channel In

- **Syntax**

  U16 _stdcall W_HSL_DIO_Channel_In (U16 card_ID,U16 set_ID,U16 slave_No ,U16 channel_No, U16* in_data);

- **Description**

  This function is to read the digital input value of the specified channel on the slave I/O module. The slave I/O module's address is slave_No, set value is set_ID and the the specified channel value is channel_No.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI- 7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
channel_No	Specify the digital input channel of the slave I/O module which want to perform this function. The valid value is described as below: HSL-R8DI16 : 0 ~ 15 . HSL-DI16DO16 : 0 ~ 15 . HSL-DI32 : 0 ~ 31 .
in_data	The digital input data of the channel.

- **Return Number**

  ERR_NoError,ERR_InvalidBoardNumber,ERR_SatelliteType,ERR_ConnectIndex,ERR_SatelliteNumber,ERR_NotADLinkSlaveType,ERR_ChannelNumber

- **Example**

  result = W_HSL_DIO_Channel_In (cardNo, 0, 1, 1, &in_data);

4.4.11 DIO Out

- **Syntax**

  U16 _stdcall W_HSL_DIO_Out (U16 card_ID, U16 set_ID, U16 slave_No,U32 out_data);

- **Description**

  This function is to write the digital output value to the slave I/O module. The slave I/O module's address is slave_No and set value is set_ID.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
out_data	the digital output data want to write to the slave I/O module. In this value. channel 0 data is assign to bit 0, channel 1 data is assign to bit 1….etc.

- **Return Number**

  ERR_NoError,ERR_InvalidBoardNumber,ERR_ConnectIndex,ERR_SatelliteNumber,
  ERR_SatelliteType,ERR_NotADLinkSlaveType

- **Example**

  result = W_HSL_DIO_Out (cardNo, 0, 1, out_data);

4.4.12 DIO Channel Out

- **Syntax**

  U16 _stdcall W_HSL_DIO_Channel_Out (U16 card_ID, U16 set_ID, U16 slave_No, U16 channel_No, U16 out_data);

- **Description**

  This function is to write the digital output value to the specified digital channel of slave I/O module. The slave I/O module's address is slave_No, set value is set_ID and the specified channel value is channel_No.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
slave_No	Specify the slave I/O module with slave_No address which want to perform this function. The valid value is between 1 ~ 63.
channel_No	Specify the digital output channel of the slave I/O module which want to perform this function. The valid value is described as below: HSL-R8DI16 : 0 ~ 7. HSL-DI16DO16 : 0 ~ 15. HSL-DO32 : 0 ~ 31.
out_data	The digital output data want to write to the channel.

- **Return Number**

  ERR_NoError,ERR_InvalidBoardNumber,ERR_ConnectIndex,ERR_SatelliteNumber,ERR_SatelliteType,ERR_NotADLinkSlaveType,ERR_ChannelNumber,ERR_NotADLinkSlaveType

- **Example**

  result=W_HSL_DIO_Channel_Out(cardNo, 0, 1, 2, out_data);

4.4.13 Software Reset

♦ **Syntax**

void _stdcall W_HSL_Software_Reset(U16 card_ID);

♦ **Description**

This function is used to reset all the H/W register value in the master card. User can use it to dynamic reset all the H/W register value.

♦ **Parameter**

card_ID	The card id of the card that want to perform this operation.

♦ **Return Number**

none

♦ **Example**

W_HSL_Reset (cardNo);

4.4.14 Timer Interrupt Enable

- **Syntax**

  U16 _stdcall W_HSL_TMRINT_Enable(U16 card_ID, HANDLE *phEvent);

- **Description**

  This function is used to enable the H/W timer interrupt of this master card.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
phEvent	Return the handle of the timer interrupt event . The interrupt event is used to indicate an interrupt which is generated for the card's timer interrupt system.

- **Return Number**

  ERR_NoError, ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_TMRINT_Enable (cardNo, &phEvent);

4.4.15 Timer Interrupt Disable

- **Syntax**

  U16 _stdcall W_HSL_TMRINT_Disable(U16 card_ID);

- **Description**

  This function is used to disable the H/W timer interrupt of this master card.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.

- **Return Number**

  ERR_NoError, ERR_InvalidBoardNumber

- **Example**

  result = W_HSL_TMRINT_Disable(cardNo);

4.4.16 Timer Set

- **Syntax**

  void _stdcall W_HSL_Timer_Set(U16 card_ID, U16 c1, U16 c2);

- **Description**

  This function is used to setup the Timer#1 and Timer#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 Win NT platform. So the multiplication of the dividers must be larger than 300. The value of c1 and c2 must be greater then 1. When c1=0 or c2=0, the timer interrupt will be stopped.

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
c1	frequency divider of timer#1
c2	frequency divider of timer#2

- **Return Number**

  none

- **Example**

```
 /* set the timer interrupt sampling rate to 6MHZ / (20* 20) =15 KHZ*/
  W_HSL_Timer_Set(cardNo, 20, 20);
/* set the timer interrupt sampling rate to 6MHZ / (100 * 50) =1.2 KHZ */
  W_HSL_Timer_Set(cardNo, 100, 50);
```

4.4.17 DIO Memory Out

- **Syntax**

  void _stdcall W_HSL_DIO_Memory_Out(U16 card_ID, U16 set_ID,unsigned short *data_out);

- **Description**

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

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

- **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
data_out	It is a unsigned short array pointer. Before use this function, user must create a unsigned short array which contain 63 items. In this array, index 0's value will be written to slave_No=1 I/O modules, index 1's value will be written to slave_No=2 I/O modules, index 2's value will be written to slave_No=3 I/O modules…… and index 62's value will be written to slave_No=63 I/O modules.

- **Return Number**

  ERR_NoError, ERR_InvalidBoardNumber, ERR_ConnectIndex,

- **Example**

```
unsigned short data_out[63]; // Create the Array

data_out[0] = 0x000f; // Set the output value of slave_No=1 I/O
                        modules to 0x000f

data_out[62] = 0x00ff; // Set the output value of slave_No=63 I/O
                        modules to 0x00ff

W_HSL_DIO_Memory_Out(cardNo, 0, data_out); // Output data
```

4.4.18 DIO Memory In

♦ **Syntax**

void _stdcall W_HSL_DIO_Memory_In(U16 card_ID, U16 set_ID, unsigned short *data_in);

♦ **Description**

This function is used to read the digital input values from all slave I/O modules which the set value is set_ID 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.

♦ **Parameter**

card_ID	The card id of the card that want to perform this operation.
set_ID	In PCI-7851, because it contain only one connector so the valid value is 0, for PCI-7852, the valid value is 0 or 1.
data_in	It is a unsigned short array pointer. Before use this function, user must create a unsigned short array which contain 63 items. After calling this function, the index 0 item value of this array will be the slave_No=1 I/O modules digital input value, the index 1 item value of this array will be the slave_No=2 I/O modules digital input value, the index 2 item value of this array will be the slave_No=3 I/O modules digital input value… and the index 62 item value of this array will be the slave_No=63 I/O modules digital input value.

♦ **Return Number**

ERR_NoError, ERR_InvalidBoardNumber, ERR_ConnectIndex,

♦ **Example**

unsigned short data_in[63]; // Create the Array

unsigned index1_value, index63_value;

W_HSL_DIO_Memory_In(cardNo, 0, data_in); // Read all Input data

```
index1_value = data_in[0]; // Read the digital input value of
                              slave_No=1 I/O modules
Index63_value = data_in[62]; // Read the digital input value of
                              slave_No=63  I/O modules
```

4.5 How to Program with HSL DLL

In this section, it introduces the programming sequence about the HSL system under Windows platform. The sequence is described as below:

Step1. Use W_HSL_Initial(…) to initial the HSL master card.

Step2. Use W_HSL_Auto_Start(….) or W_HSL_Start(….) to start the HSL slave modules operation which connect with the master card.

Step3. After Step2, the HSL system is enable. So now you can use some functions as below for HSL operation.

a. W_HSL_Slave_Live(…) : It is used to detect the status of the slave module(Live or Die).

b. W_HSL_DIO_In(…),W_HSL_DIO_Channel_In(….),W_HSL _DIO_Memory_In(….) : It is for digital input operation of slave module.

c. W_HSL_DIO_Out(…)orW_HSL_DIO_Channel_Out(….),W_ HSL_DIO_Memory_Out(….) : It is for digital output operation of slave module.

d. W_HSL_Connect_Status(…) : It is used to detect the communication status of the slave module.

Step4. Use W_HSL_Stop(….) to stop the HSL slave modules operation which connect with the master card.

Step5. Use W_HSL_Close(….) to release the HSL master card.

For more programming information, please refer to the example programs which is located in the software installation directory.

5

HSL Products Selection Guides

System Requirement: (I/O Point)

System I/O Point	PCI-7851
# < 2000	X 1
2000 < # < 4000	X 2
4000 < # < 6000	X 3
6000 < # < 8000	X 4
8000 < # < 10000	X 5
10000 < # < 12000	X 6
12000 < # < 14000	X 7
14000 < # < 16000	X 8

System Requirement: (I/O Refreshing Rate)

Refreshing Rate	Slave ID Occupy	PCI-7851	Max. I/O Point
< 100 us	3	X 1	96
	3	X 2	192
	3	X 8	768
< 500 us	16	X 1	512
	16	X 2	1.024
	16	X 8	4,096
< 1000 us	33	X 1	1,056
	33	X 2	2,112
	33	X 8	8,448
< 2000 us	63	X 1	2,016
	63	X 2	4,032
	63	X 8	16,128

I/O Model Selection:

-DB	-M	-C
● Daughter Board form factor ● Suitable for machine builders or user with their own cabinet ● Suitable for OEM/ODM ● Lower cost	● Daughter with aluminum cover ● Rugged ● Well dust and electromagnetic protection ● Higher cost	● Compact sized slave I/O module ● Stackable ● Suitable for application with small I/O usage

I/O Type Selection Terminology:

•	DI:	Discrete input	
•	DO:	Discrete output	
•	R:	Relay output	
•	-M:	Module with metal housing	
•	-DB:	Daughter board without metal case	
•	-N:	NPN sinking	
		for input:	input switch couple with input channel and ground.
		for output:	output load couple with output channel and power.
•	-P:	PNP sourcing	
		for input:	input switch couple with input channel and power.
		for output:	output load couple with output channel and ground.
•	-DIN:	DIN socket support	

-DB Discrete I/O Selection						
Model	-N Input #	-P Input #	-N Output #	-P Output #	Relay Output #	ID Occupy
HSL-DI32-DB-N	32	NA	NA	NA	NA	2
HSL-DI32-DB-P	NA	32	NA	NA	NA	2
HSL-DO32-DB-N	NA	NA	32	NA	NA	2
HSL-DO32-DB-P	NA	NA	NA	32	NA	2
HSL-DI16DO16-DB-NN	16	NA	16	NA	NA	1
HSL-DI16DO16-DB-NP	16	NA	NA	16	NA	1
HSL-DI16DO16-DB-PN	NA	16	16	NA	NA	1
HSL-DI16DO16-DB-PP	NA	16	NA	16	NA	1
HSL-R8DI16-DB-N	16	NA	NA	NA	8	1
HSL-R8DI16-DB-P	NA	16	NA	NA	8	1

-DB Terminal Base Selection				
Model	HSL-TB32-DIN	HSL-TB32-U-DIN	HSL-TB32-DO-DIN	HSL-TB64-DIN
HSL-DI32-DB-N	NA	1 –DB support	NA	2 –DB support
HSL-DI32-DB-P	NA	1 –DB support	NA	2 –DB support
HSL-DO32-DB-N	NA	1 –DB support	1 –DB support Lift output to 300mA/ch Output short circuit protection	2 –DB support
HSL-DO32-DB-P	NA	1 –DB support	NA	2 –DB support
HSL-DI16DO16-DB-NN	1 –DB support Lift output to 300mA/ch Output short circuit protection	1 –DB support	NA	2 –DB support
HSL-DI16DO16-DB-NP	NA	1 –DB support	NA	2 –DB support
SL-DI16DO16-DB-PN	1 –DB support Lift output to 300mA/ch Output short circuit protection	1 –DB support	NA	2 –DB support
HSL-DI16DO16-DB-PP	NA	1 –DB support	NA	2 –DB support
HSL-R8DI16-DB-N	NA	1 –DB support	NA	2 –DB support
HSL-R8DI16-DB-P	NA	1 –DB support	NA	2 –DB support

-M Discrete I/O Selection						
Model	-N Input #	-P Input #	-N Output #	-P Output #	Relay Output #	ID Occupy
HSL-DI32-M-N	32	NA	NA	NA	NA	2
HSL-DI32-M-P	NA	32	NA	NA	NA	2
HSL-DO32-M-N	NA	NA	32	NA	NA	2
HSL-DO32-M-P	NA	NA	NA	32	NA	2
HSL-DI16DO16-M-NN	16	NA	16	NA	NA	1
HSL-DI16DO16-M-NP	16	NA	NA	16	NA	1
HSL-DI16DO16-M-PN	NA	16	16	NA	NA	1
HSL-DI16DO16-M-PP	NA	16	NA	16	NA	1
HSL-R8DI16-M-N	16	NA	NA	NA	8	1
HSL-R8DI16-M-P	NA	16	NA	NA	8	1

-M Terminal Base Selection	
Model	HSL-TB32-M-DIN
HSL-DI32-M-N	1 –M module support
HSL-DI32-M-P	1 –M module support
HSL-DO32-M-N	1 –M module support
HSL-DO32-M-P	1 –M module support
HSL-DI16DO16-M-NN	1 –M module support
HSL-DI16DO16-M-NP	1 –M module support
HSL-DI16DO16-M-PN	1 –M module support
HSL-DI16DO16-M-PP	1 –M module support
HSL-R8DI16-M-N	1 –M module support
HSL-R8DI16-M-P	1 –M module support

-C Discrete I/O Selection						
Model	-N Input #	-P Input #	-N Output #	-P Output #	Relay Output #	ID Occupy
HSL-DI8DO8-C-NN	8	NA	8	NA	NA	1
HSL-DI8DO8-C-NP	8	NA	NA	8	NA	1
HSL-DI8DO8-C-PN	NA	8	8	NA	NA	1
HSL-DI8DO8-C-PP	NA	8	NA	8	NA	1

-C Terminal Base Selection	
Model	HSL-TB16-C
HSL-DI8DO8-C-NN	1 –C module support
HSL-DI8DO8-C-NP	1 –C module support
HSL-DI8DO8-C-PN	1 –C module support
HSL-DI8DO8-C-PP	1 –C module support

Product Warranty/Service

ADLINK warrants that equipment furnished will be free from defects in material and workmanship for a period of one year from the date of shipment. During the warranty period, we shall, at our option, either repair or replace any product that proves to be defective under normal operation.

This warranty shall not apply to equipment that has been previously repaired or altered outside our plant in any way as to, in the judgment of the manufacturer, affect its reliability. Nor will it apply if the equipment has been used in a manner exceeding its specifications or if the serial number has been removed.

ADLINK does not assume any liability for consequential damages as a result from our product uses, and in any event our liability shall not exceed the original selling price of the equipment. The remedies provided herein are the customer's sole and exclusive remedies. In no event shall ADLINK be liable for direct, indirect, special or consequential damages whether based on contract of any other legal theory.

The equipment must be returned postage-prepaid. Package it securely and insure it. You will be charged for parts and labor if the warranty period is expired or the product is proves to be misuse, abuse or unauthorized repair or modification.