5

DOS C/C++ Programming

5.1 How to Program with C/C++

Some data types defined in HSL.H are used for the HSLink DOS C/C++ library. We suggest you to use these data types in your application programs. The following table shows the names and their corresponding 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 integer	-2147483648 to 2147483647
U32	32-bit unsigned 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

Functions of the HSLink DOS C/C++ drivers are using full-names to represent their meanings. The naming convention is:

_{hardware_model}_{action_name}. **e.g. `_HSL_Initial()`.**

The followings are the detailed descriptions of each function provided by HSLink DOS C/C++ drivers.

5.1.1

- ***Syntax***

  U16 _HSL_Initial (U16 *existCards, PCI_INFO *info);

- ***Description***

  The HSLink master cards are initialized by this function. The software library could be used to control multiple HSLink master cards. Because HSLink master is in PCI bus architecture and meets the plug-and-play specifications, the ***IRQ*** and ***I/O address*** are assigned by system BIOS directly.

- ***Parameter***

  existCards: The numbers of installed HSLink master cards. The returned value shows how many HSL Master cards are installed in your system.

  info: It is adata structure used to memorize the PCI bus plug-and-play initialization information which is decided by PnP BIOS. The PCI_INFO structure is defined in HSL.H. The base I/O address and the interrupt channel number is stored in info for reference.

- ***Return Number***

  ERR_NoError, ERR_PCIBiosNotExist

- ***Example***

  result = _HSL_Initial (&bn, &info);

5.1.2

- ***Syntax***

  void _HSL_Software_Reset (U16 card_ID);

- ***Description***

This function is used to reset all the H/W registers in the master card.

- ***Parameter***

card_ID	The sequence number of HSLink master card. For example, if there are two HSLink master cards on your PC, the HSLink master card in prior slot should be registered as card_ID = 0, and the second one is registered as card_ID = 1. This parameter assigns the card to perform operation.

- ***Return Number***

none

- ***Example***

```
_HSL_Software_Reset (cardNo);
```

5.1.3

- ***Syntax***

```
U16 _HSL_Start (U16 card_ID, U16 cnnctr_Index, U16 slave_Num);
```

- ***Description***

This function is used to set the total number of the slave I/O modules that connected to HSLink master card' s (PCI-7851/52) connector and start to scan these slave I/O modules.

- ***Parameter***

card_ID	The card ID used to perform this operation.
cnnctr_Index	In the PCI-7851, it contains only one connector. The valid value is 0. For the PCI-7852, the valid value is 0 or 1.
slave_Num	The maximum slave number connect to the connector. The valid value is between 1 ~ 63.

- ***Return Number***

ERR_NoError, ERR_SatelliateNumber, ERR_ConnectIndex, ERR_InvalidBoardNumber

- ***Example***

result = _HSL_Start (cardNo, 0, 63);

5.1.4

- ***Syntax***

  U16 _HSL_Stop (U16 card_ID, U16 cnnctr_Index);

- ***Description***

  This function stops the scan of connected slave I/O modules connected to the HSLink master card.

- ***Parameter***

card_ID	The card ID used to perform this operation.
cnnctr_Index	In the PCI-7851, it contains only one connector. The valid value is 0. For the PCI-7852, the valid value is 0 or 1.

- ***Return Number***

  ERR_NoError, ERR_ConnectIndex, ERR_InvalidBoardNumber

- ***Example***

  result = _HSL_Stop (cardNo, 0);

5.1.5

- ***Syntax***

  void _HSL_Close (U16 card_ID);

- ***Description***

  This function isused to close the HSLink master card with card_ID. This function is used to tell library that this registered card is no longer used 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 used to perform this operation.

- ***Return Number***

  none

- ***Example***

  _HSL_Close (0);

5.1.6

- ***Syntax***

  void _HSL_Get_IRQ_Channel (U16 card_ID, U16 *irq_no);

- ***Description***

  This function is used to get the IRQ number of the HSLink master card with card_ID

- ***Parameter***

card_ID	card_ID The card ID used to perform this operation.
irq_no	The IRQ number of this card.

- ***Return Number***

  none

- ***Example***

  _HSL_Get_IRQ_Channel (0, &irq_no);

5.1.7

- ***Syntax***

  void _HSL_Get_Base_Address (U16 card_ID, U16 *base_addr);

- ***Description***

  This function isused to get the base address of the HSLink master card

with card_ID

- ***Parameter***

card_ID	card_ID The card ID used to perform this operation.
base_addr	The base address of this card.

- ***Return Number***

none

- ***Example***

```
_HSL_Get_Base_Address (0, &base_address);
```

5.1.8

- ***Syntax***

```
U16 _HSL_DIO_In (U16 card_ID, U16 slave_ID, U32* in_data);
```

- ***Description***

This function read the digital input value of the slave I/O module. The slave I/O module' s address is slave_ID.

- ***Parameter***

card_ID	The card ID used to perform this operation.
slave_ID	Specify the slave I/O module which wants to perform this function. For the first connector, the valid value is between 1 ~ 63; for the second connector, the valid value is between 64 ~ 126.
in_data	The digital input data of this slave I/O module. Channel 0' s data is assigned to bit 0, channel 1' s data is assigned to bit 1..etc.

- ***Return Number***

ERR_NoError, ERR_SatelliateNumber

- ***Example***

```
result = _HSL_DIO_In (cardNo, 1, &in_data);
```

5.1.9

- ***Syntax***

  U16 _HSL_DIO_Out (U16 card_ID, U16 slave_ID, U32* in_data);

- ***Description***

  This function writes the digital output value to the slave I/O module. The slave I/O module‘ s address is slave_ID.

- ***Parameter***

slave_ID	Specify the slave I/O module which wants to perform this function. For the first connector, the valid value is between 1 ~ 63; for the second connector, the valid value is between 64 ~ 126.
out_data	the digital output data that want to write to the slave I/O module. Channel 0’ s data will be assigned to bit 0, channel 1’ s data will be assigned to bit 1…etc.

- ***Return Number***

  ERR_NoError, ERR_SatelliteNumber

- ***Example***

  result = _HSL_DIO_Out (cardNo, 1, out_data);

5.1.10

- ***Syntax***

  U16 _HSL_DI16DO16(U16 card_ID, U16 slave_ID, U16* in_data, U16 out_data);

- ***Description***

This function reads the digital input value of the slave I/O module and writes the digital output value to the slave I/O module at the same time. The slave I/O module' s address is slave_ID.

- ***Parameter***

card_ID	The card ID used to perform this operation.
slave_ID	Specify the slave I/O module which wants to perform this function. For the first connector, the valid value is between 1 ~ 63; for the second connector, the valid value is between 64 ~ 126.
in_data	The digital input data of this slave I/O module. Channel 0' s data is assigned to bit 0, channel 1' s data is assigned to bit 1…etc.
out_data	the digital output data to write to the slave I/O module. Channel 0' s data will be assigned to bit 0, channel 1' s data will be assigned to bit 1…etc.

- ***Return Number***

ERR_NoError, ERR_SatelliateNumber

- ***Example***

```
result = _HSL_DI16DO16 (cardNo, 1, &in_data, out_data);
```

5.1.11

- ***Syntax***

```
void _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 can not exceed 20KHZ. So the multiplication of the dividers must larger than 300. Besides, the value of c1 and c2 must greater then 1. When c1=0 or c2=0, the timer interrupt will be stopped.

- ***Parameter***

card_ID	The card ID used to perform this operation.
c1	frequency divider of timer#1
c2	frequency divider of timer#2

- ***Return Number***

none

- ***Example***

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