

## Table of Contents

1. Overview .....	2
2. Physical Connections .....	2
Table 1 GPIO Electrical Specifications .....	2
Table 2 GPIO Digital Input/Output Connector .....	3
Figure 1 External DIO Connections .....	3
3. Accessing the GPIO Ports .....	5
Figure 3 General Purpose Input (GPI) Register .....	5
Figure 4 General Purpose Output (GPO) Register .....	5
3.1. Linux Reference Code .....	6
3.2. Windows Reference Code .....	7

## 1. Overview

The model LPC-81x, LPC-83x Little PC and SR-283xx Rack Mount Computer provides 4 isolated digital inputs and 4 isolated digital outputs via the GPIO (Digital Input/Output) Connector located on the back of the unit. This GPIO connector is a 1x10 10-pin, 3.5mm pitch terminal block, located on the rear of the system. The GPIO ports can be accessed via software, as described in **Section 3** of this document.

## 2. Physical Connections

The electrical specifications for the inputs and outputs are shown in **Table 1**.

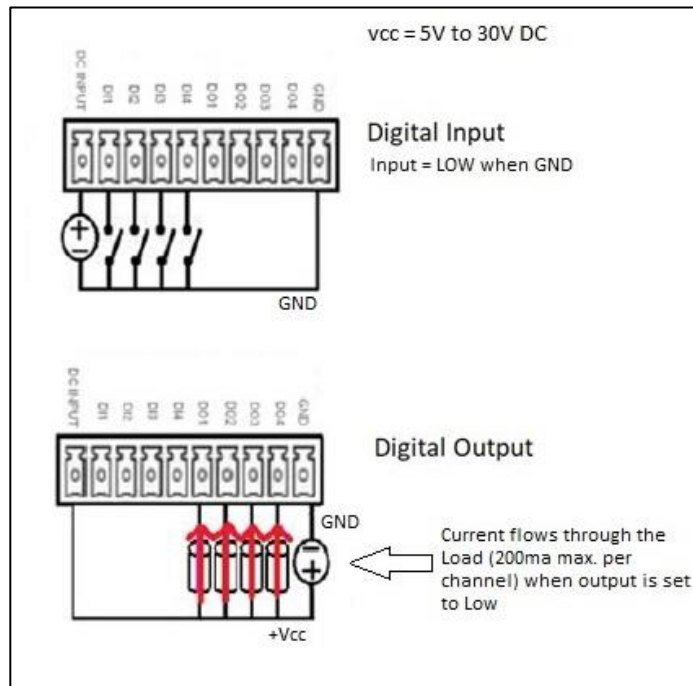
INPUT	
Item	Specification
Number of input channels	4
Input format	Opto-isolated input
Input voltage	5~30VDC
Input current	200 mA max
OUTPUT	
Number of output channels	4
Output format	Opto-isolated output
Output voltage	5~30VDC
Output current	200 mA max
Surge protector	Zener diode (BZX84C51-TG-WS/WILLAS)
Wire Gauge	
Conductor cross-section, solid (AWG/mm2)	28~14/0.2~1.5
Conductor cross-section, flexible (AWG/mm2)	28~14/0.2~1.5

**Table 1 GPIO Electrical Specifications**

**Table 2** and **Figure 1** show the pinouts and external connections for the DIO Connector.

Pin	Definition	Pin	Definition
1	DC INPUT	6	DO1
2	DI1	7	DO2
3	DI2	8	DO3
4	DI3	9	DO4
5	DI4	10	GND

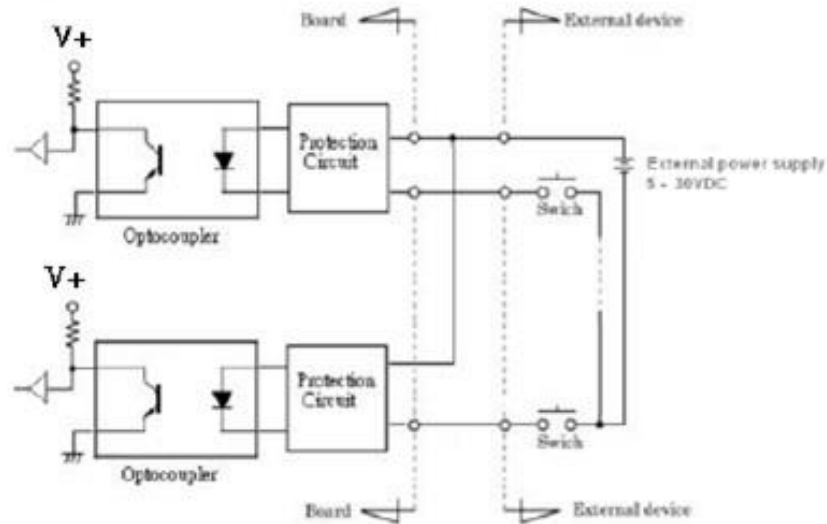
**Table 2 GPIO Digital Input/Output Connector**



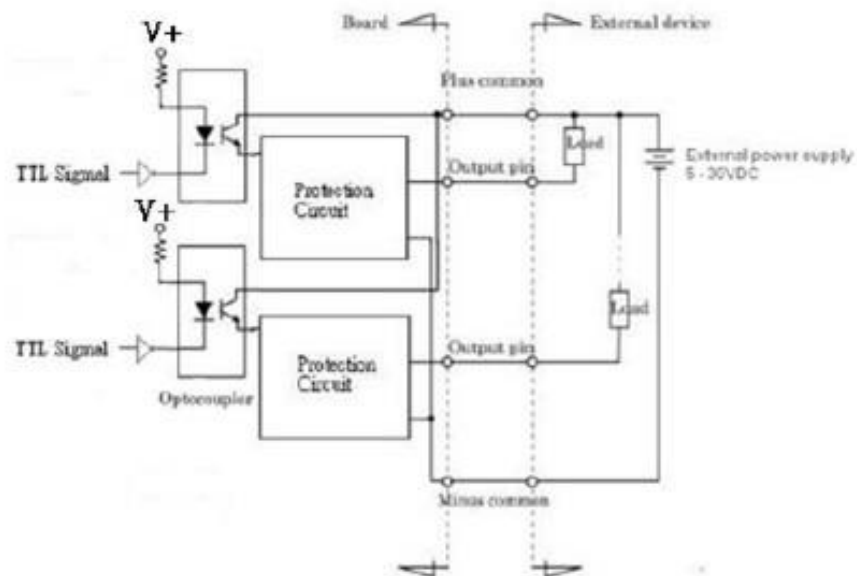
**Figure 1 External DIO Connections**

For reference, **Figure 2** shows the input and output electrical circuits.

### Input Circuit



### Output Circuit



**Figure 2 Equivalent Input and Output Circuits**

### 3. Accessing the GPIO Ports

General-purpose input/output (GPIO) functionality is provided by the Fintek F81866 Super I/O (SIO) controller and this controller can be accessed through the GPIO index/data ports. The Fintek SDK (Software Development Kit) can be used to program the GPIO. Sample code is included with the Fintek SDK. Download the Fintek SDK here:

<http://www.fintek.com.tw/index.php/mnu-swdevelopkitdl>

The GPIO register is accessed by writing an index to the index port, followed by a read/write from/to the data port. The configuration of the GPIO for the LPC-81x, LPC-83x and SR-283xx is described below:

Figure 3 and Figure 4 provide the GPIO Address and bit references for each of the General Purpose Inputs (GPI) and the General Purpose Outputs (GPO).

	GPI 0	GPI 1	GPI 2	GPI 3
GPI Register Address	0xA03h	0xA03h	0xA03h	0xA03h
Bit	4	5	6	7
Connector Reference	D11	D12	D13	D14

**Figure 3 General Purpose Input (GPI) Register**

	GPO 0	GPO 1	GPO 2	GPO 3
GPO Register Address	0xA02h	0xA02h	0xA02h	0xA02h
Bit	0	1	2	3
Connector Reference	DO1	DO2	DO3	DO4

**Figure 4 General Purpose Output (GPO) Register**

These constant values are used to support the configuration and manipulation of the GPIO port (as referenced in the psuedo code) sample below:

```
SIO_INDEX_Port = 0x04Eh
SIO_DATA_Port = 0x04Fh
SIO_UnLock_Value = 0x087h
SIO_Lock_Value = 0x0AAh
SIO_LDN_GPIO = 0x06h
```

```
GPI_ADDR = 0xA03h
GPO_ADDR = 0xA02h
GPO_0 = 00010000b
```

### 3.1. *Linux Reference Code*

In linux, **ioperm**, **inb**, and **outb** system function calls are used to program the GPIO. This sample C program (**gpio** command) takes 3 input arguments, and either reads a GPI pin value or sets a GPO pin (high or low).

**gpio usage:**

```
gpio -i <DI_IDX> -i | -o <DO_IDX> -v <0|1 >

-i <DI_IDX> // DI index, ex. DI1, use 1
-o <DO_IDX> // DO index, ex. DO2, use 2
-v <0 | 1> // 0 Pull DO pin low, 1, pull DO pin high
```

**Example:**

```
gpio -i 1 // Read from DI1
gpio -i 2 -v 1 // Pull DO2 high
```

**Sample code snippet:**

Set DO2 (GPO 1) pin to high

```
// set access permission
ioperm(SIO_INDEX_Port, 2, 1);

// enable config mode, switch GPIO configuration
outb(SIO_INDEX_Port, SIO_UNLOCK_VALUE);
usleep(4000);
outb(SIO_INDEX_Port, SIO_UNLOCK_VALUE);
outb(SIO_INDEX_Port, 0x07); // Enter selecting mode
outb(SIO_DATA_Port, SIO_LDN_GPIO);

// set GPO pin to high
outb(SIO_INDEX_Port, GPO_ADDR);
data=inb(SIO_DATA_Port);
data=data|(1<<1);
outb(SIO_DATA_Port, data);

// close config mode
outb(SIO_INDEX_Port, SIO_LOCK_VALUE);
```

### 3.2. *Windows Reference Code*

When programming in the Windows environment, the GPIO is programmed using InpOut32 or InpOut64 to set registers and read/write the ports. InpOut32 and InpOut64 are open source Windows DLLs and drivers. More information about the open source DLLs and drivers can be found at <http://www.highrez.co.uk/downloads/inpout32/>.

The sample code is built using Visual Studio 2017, and is a Visual C++ Console Program which uses inpoutx64.dll. To build a 32-bit version, simply change your Platform target and use inpout32.dll.

Since this is a console program, it must be run from the command prompt. Inpoutx64.dll (or inpout32.dll is 32-bit) MUST be in the same directory as the executable. The first time the program is run, execute it in an elevated command prompt (such as an administrator) so the DLL can install the appropriate driver.

The sample console application (**gpio** command) takes 3 input arguments, and either reads a GPI pin value or sets a GPO pin (high or low).

**gpio usage:**

```
gpio <PIN_IDX> <Direction> <Value>

<PIN_IDX>    // DI Idx.  ex. DI1 or DO1, use 1
<Direction> // 0 for DO output pin, 1 for DI input pin
<Value>     // Pull high or low ex. 0 for low, 1 for high
```

**Example:**

```
gpio 1 1 0    // Read from DI1
gpio 2 0 1    // Pull DO2 high
gpio 2 0 0    // Pull DO2 low
```

**Sample code snippet:**

Set DO2 (GPO 1) to high

```
hinstLib = LoadLibrary("Inpoutx64.DLL");
gfpOut32 = (lpOut32)GetProcAddress(hinstLib, "Out32");
gfpInp32 = (lpInp32)GetProcAddress(hinstLib, "Inp32");

// enable config mode, switch GPIO configuration for SIO LDN
gfpOut32(SIO_INDEX_Port, SIO_UnLock_Value);
Sleep(4);
gfpOut32(SIO_INDEX_Port, SIO_UnLock_Value);
gfpOut32(SIO_INDEX_Port, 0x07);
gfpOut32(SIO_DATA_Port, SIO_LDN_GPIO);

// pull DO 2 pin high
gfpOut32(SIO_INDEX_Port, GPO_ADDR);
```

```
data = gfpInp32(SIO_DATA_Port);  
data = data | (1 << 1);  
gfpOut32(SIO_DATA_Port, data);  
  
// close config mode  
gfpOut32(SIO_INDEX_Port, SIO_Lock_Value);
```