Step 1：

Connect the IIC GPIO Module with Arduino UNO.

Connect way： VCC-5V GND-GND SDA-A4 SCL-A5. I/O_0.0- I/O_1.7 connected to the digital devices. When in test, we use it to drive the 16 LED lights and 16-channel relay module, that is, the positive of the 16 LED lights connected to I / O interfaces, and the negative connected to the ground.（16 channel relay module connecting way： VCC-5V、 GND-GND 、1 to16 interfaces connected to the I/O_0.0- I/O_1.7 interface in order respectively. If you are interested in our 16-channel relay module, please refer to our websitehttp://www.elecfreaks.com/store/16-channel-5v12v-minirelay-module-p-421.html

Here attached the product connection diagram ：

Step 2：

Download the Arduino test code in our WIKI， and the copy the unzipped file of Arduino Sample Code into the folder of Arduino IDE libraries.

Step 3：

Open the Arduino IDE and click File->Examples->“GPIO_Sample”, and then choose the sample, compile it and upload it until done-uploading

appears. At this time, If you see 16 LED light and operate one by one, then congratulations, the experiment you are undertaking was successful!

Step 4:

What ‘s more, IIC GPIO Module can set different module addresses to connect the 8 IIC GPIO Module in parallel， as module I/O_0.0- I/O_1.7 are readable or writable digital IO, you can set the module I/O into input/output according to your own demand. Currently we demonstrate its function with a object connecting diagram. (as picture 2)

Module internal Command Register:

Register address	function
0x00	P0 Port input，high impedance state，defaulted as high level.
0x01	P1 Port input，high impedance state，defaulted as high level.
0x02	P0 Port output register， 8 represent 8 IO state respectively
0x03	P1 Port output register， 8 represent 8 IO state respectively
0x04	P0 Port input level inversion, high active.
0x05	P1 Port input level inversion, high active.
0x06	P0 Port configurate register, set high input, set low output, defaulted as high.
0x07	P0 Port configurate register, set high input, set low output, defaulted as high.

Module Address:

	Slave address							
X	X	1	0	0	A0	A1	A2	Address	Module
X	X	1	0	0	0	0	0	0X20	1
X	X	1	0	0	0	0	1	0X21	2
X	X	1	0	0	0	1	0	0X22	3
X	X	1	0	0	0	1	1	0X23	4
X	X	1	0	0	1	0	0	0X24	5
X	X	1	0	0	1	0	1	0X25	6
X	X	1	0	0	1	1	0	0X26	7
X	X	1	0	0	1	1	1	0X27	8

Module writing output register timing order：

Module Address Setting：

From the above timing order chart, the module address is 8 bytes, including one byte of R/W, but when we operate the module, we only need 7 of them, that is, the address range from 0x20～0x27. For the R/W byte, Wire library would automatically add it according to the operation direction. A2～A0 address jumpers all connected to the ground with short caps, and the defaulted address is 0x20. If the modules need to be connected in parallel among the IIC Bus, each module need to set one address, unrepeatable, but if only one module used, the address should be 0x20. The address setting method is that A2～A0 represent the address 3 low bytes, and from 000 to 111, short caps to ground represent 0, and jumper wire to power supply 5V represent 1.

Module has another interruption output port (INT), and when IO port was set as input, if only there has change for the IO port input, the interruption will output and the low level would be active.

As the above picture demonstrated, Arduino used 4 wires（5V，GND，SCL Analog Port 5，SDA Analog Port 4）to connect with module B，and module A also connected to the module B IIC interface in parallel. Module A address was set as 0x27，A2～A0 were connected to 5V with Jumper Wire，module B address was set as 0x20，A2～A0 were connected to GND with short caps.

Their functions were that，module A perform as output，P1 port 4 high bytes high drive 4 relays, and the module B perform as input, P0 port 4 low bytes connect to keypad (one jumper wire replaced the keypad function in the above picture) When module B one of P0 port 4 low bytes connected to the ground, the module A P1 port 4 high bytes IO would cut off the display relay.

Attached Reference Code：

```
#include <Wire.h>
//   Address                    0 1 0 0     A2 A1 A0
#define KEY_ADDRESS 0x20 // Module B Address
#define LED_ADDRESS 0x27 // Module A Address
#define REGISTER_INPUT 0x00 // Register 0 set P0 input
#define REGISTER_OUTPUT 0x02 // Register 2 set P0 Output parameters
#define REGISTER_CONFIG 0x06 // Register 6 set P0 state
void setup()
{
   Wire.begin();
   gpio_write(LED_ADDRESS, 0xffff);// Set module A, IO Port as high output
   gpio_dir(LED_ADDRESS, 0x0000);// Set module A, IO Port as output
}

void loop()
{
   int bits;
   bits = gpio_read(KEY_ADDRESS) & 0x0f;// Read module B, as input
register

   gpio_write(LED_ADDRESS,
         (
            ((bits & 1) << 3) | ((bits & 2) << 1) |
            ((bits & 4) >> 1) | ((bits & 8) >> 3)
         ) << 12);// Mirror output the module B value to module A

   delay(200);
}
int gpio_read(int address)// Read function
{
```

```
    int data = 0;

    // Send input register address
    Wire.beginTransmission(address);// Module address
    Wire.send(REGISTER_INPUT);// Register address
    Wire.endTransmission();

    // Connect module and receive 2 bytes data
    Wire.beginTransmission(address);
    Wire.requestFrom(address, 2);

    if (Wire.available()) // Read P0
    {
        data = Wire.receive();// Read P0 Port 8 bytes
    }
    if (Wire.available())
    {
        data |= Wire.receive() << 8;// Read P1 Port 8 bytes

    }

    Wire.endTransmission();

    return data;
}
void gpio_dir(int address, int dir) // Configuration Functions
{
    // Send configuration register address
    Wire.beginTransmission(address);
    Wire.send(REGISTER_CONFIG);

    // Connect module and send 2 bytes data
    Wire.send(0xff & dir);    // Writing P0 Configuration Register 8 bytes

    Wire.send(dir >> 8);       // Writing P1 Configuration Register 8 bytes

    Wire.endTransmission();
}
void gpio_write(int address, int data) //Writing Function
{
    // Send Output Register Address
    Wire.beginTransmission(address);
    Wire.send(REGISTER_OUTPUT);
```

```
    //   Connect module and send two bytes of data
    Wire.send(0xff & data);   //   Writing P0 Output Register 8 bytes
    Wire.send(data >> 8);      //   Writing P1 Output Register 8 bytes

    Wire.endTransmission();
}
```

(Code directly copy into Arduino IDE，and then programme into UNO)