GPI/O 32 Board

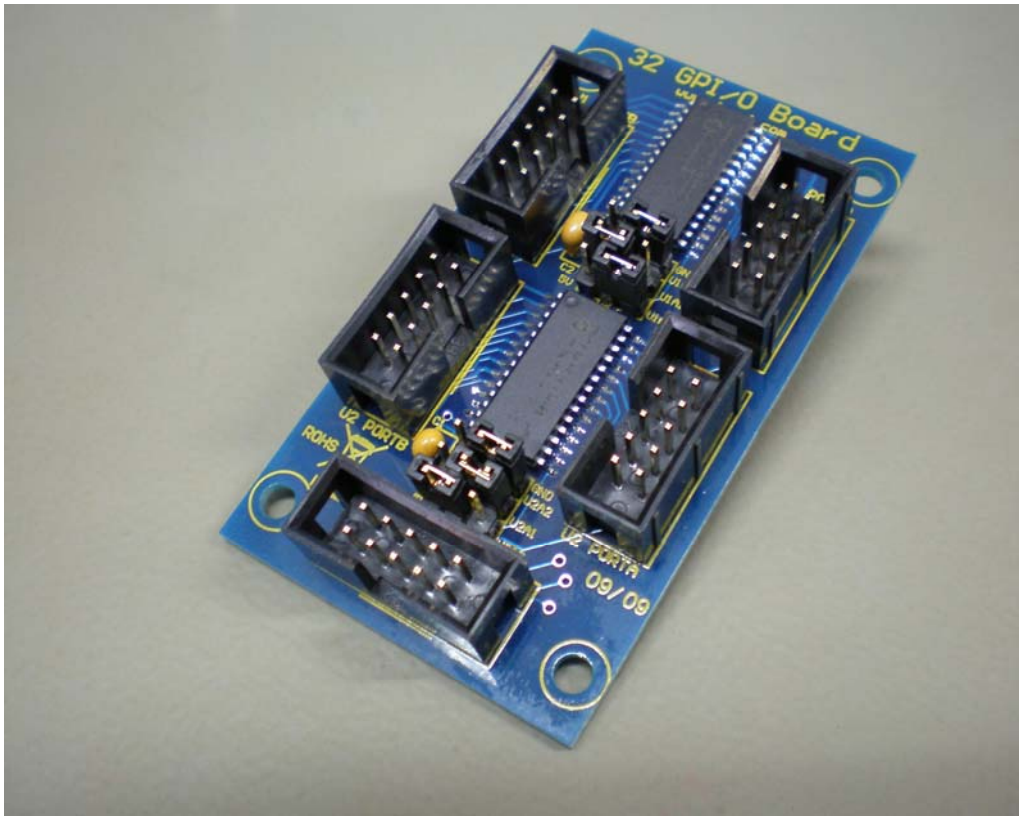


Diagram 1

The GPI/O 32 Board shown in diagram 1 above, is an accessory board that allows up to 32 I/O pins to be controlled through a single Port connection on Elexol's I/O 24 Range. The Elexol I/O 24 Range consists of Ether I/O 24 R, Ether I/O 24 DIP R, USB I/O 24 R and the USB I/O 24 DIP R.

The GPI/O 32 Board incorporates 2 x MCP23S17 GPIO expanders, which are split up into four 8 bit Ports (2 Ports per MCP23S17). This allows up to 32 I/O pins to be controlled via the Elexol I/O 24 SPI Protocol.

BOARD FEATURES

- 2 x MCP23S17 GPIO expander IC's
- 4 x I/O ports matching configuration of I/O 24 Port.
- Hardware addressable pins for each SPI device
- 72mm Standard width for DIN Rail Modules
- Compact measurements 40 x 72 x 11mm (DIN Rail mountable with adaptor).

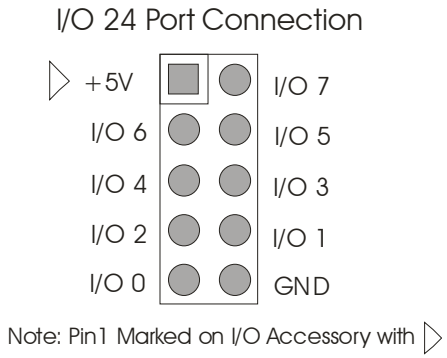
BOARD LAYOUT AND PHYSICAL DIMENSIONS

Dimensions – 1.58 X 2.8 X 0.4 inches (40.2 X 72 X 11mm)

BOARD CONNECTIONS

10 pin Box Header Pin out for U1 Port A & B, U2 Port A & B

Shown in the diagram below is the I/O port Connector for each of the Ports on the module.



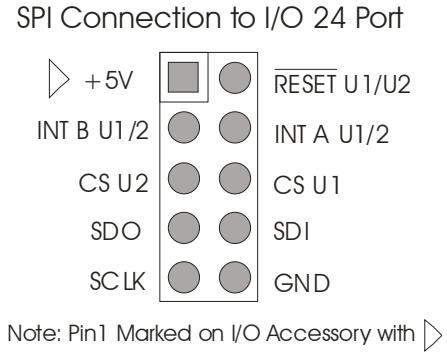
Listed in Table 1 below are the connections for the 10 Pin Box Header for U1,2 Port A & B

TABLE 1

PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+5V drawn from I/O module powers
2	I/O 7	I/O	Input / Output pin 7
3	I/O 6	I/O	Input / Output pin 6
4	I/O 5	I/O	Input / Output pin 5
5	I/O 4	I/O	Input / Output pin 4
6	I/O 3	I/O	Input / Output pin 3
7	I/O 2	I/O	Input / Output pin 2
8	I/O 1	I/O	Input / Output pin 1
9	I/O 0	I/O	Input / Output pin 0
10	GND	PWR	Ground signal from I/O module

10 pin Box Header Pin out for SPI Connection to I/O 24 Board

Shown in the diagram below is the SPI port Connection to the I/O 24. The SDI and SDO pins correspond to the GPIO module.



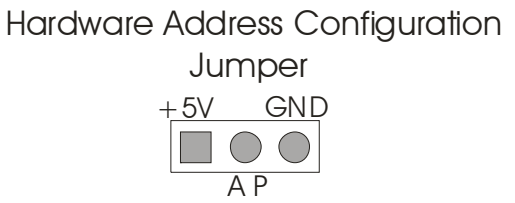
Listed in Table 2 below are the connections for the 10 Pin Box Header for the SPI connection to the I/O 24 port

TABLE 2

PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+5V drawn from I/O module powers
2	Reset U1/U2	I	Hardware Reset control line for U1/U2. Reset signal is active low in order to keep U1/U2 out of reset the line must be held high
3	INT B U1/U2	O	Interrupt Output for PORTB of U1/U2. Can be configured as active high, active low or open drain depending on the register setup for the MCP23S17
4	INT A U1/U2	O	Interrupt Output for PORTA of U1/U2. Can be configured as active high, active low or open drain depending on the register setup for the MCP23S17
5	CS U2	I	Chip Select U2 of MCP23S17
6	CS U2	I	Chip Select U1 of MCP23S17
7	SDO	O	Serial Data Output pin from MCP23S17
8	SDI	I	Serial Data Input pin from MCP23S17
9	SCLK	I	Serial Clock Input
10	GND	PWR	Ground signal from I/O module

Hardware Address U1/U2 Configuration Jumpers

There are three hardware address jumpers per MCP23S17, the jumper configuration is set out below.



PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+5V drawn from I/O module powers
2	AP	I	
3	GND	GND	Ground signal from I/O module

OPERATION

Communication to the module is via SPI from the port of I/O 24. Outlined below are some standard commands that can be sent to the GPI/O 32 Board.

Ether I/O 24 Commands

Setting up Ether I/O Port A for GPIO Expander Board

```
private void Send_Enable_SPI()
{
// Send out an SPI enable command
//Need to set up the various CS and other pins associated with the board
data[0] = Convert.ToByte('!'); //"!"
data[1] = Convert.ToByte('A');//"A"
data[2] = 0x64; //"0x48"

udpClient.Send(data, 3, "10.10.10.10", 2424);

data[0] = Convert.ToByte('A');//"A"
data[1] = 0x99; //"0x31" was 0x98

udpClient.Send(data, 2, "10.10.10.10", 2424);

//Enable the SPI by sending S1A

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('1');//"1"
data[2] = Convert.ToByte('A');//"A"

udpClient.Send(data, 3, "10.10.10.10", 2424);
}
```

Write 0xAA to Port B of GPIO U2

```
private void button1_Click(object sender, EventArgs e)
{

//set CS value u1
// IdleCS = 0x99;
// U1 LowerCS = 0x91; //U2 LowerCS = 0x89;
// Send out an SPI enable command
//Need to set up the various CS and other pins associated with the board

//Lower CS2
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x91; //lower CS u1 = "0x91" or u2 "0x89
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Write to Port Direction of U2 Port B

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('A'); //"A"
data[2] = 0x03; //number of bytes to be sent out via SPI command
data[3] = 0x40; //first byte  //command byte
data[4] = 0x01; //second byte //register byte
data[5] = 0x00; //third byte  //register data byte
udpClient.Send(data, 6, "10.10.10.10", 2424);
```

```
//Raise CS2 both CS are in idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x99; //"0x99"
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Lower CS2
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x91; //"0x91" or "0x89
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Write to Value of U2 Port B

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('A'); //"A"
data[2] = 0x03; //number of bytes to be sent out via SPI command
data[3] = 0x40; //first byte  //command byte
data[4] = 0x13; //second byte //register byte
data[5] = 0xAA; //third byte // register value to be written
//udpClient.Send(data, 6, EtherIP);
udpClient.Send(data, 6, "10.10.10.10", 2424);

//Raise CS2 both CS are in idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x99; //"0x90" or "0x88
udpClient.Send(data, 2, "10.10.10.10", 2424);

}
```

Read Port B of GPIO U2

```
//set CS value u1
// IdleCS = 0x99;
// U1 LowerCS = 0x91; //U2 LowerCS = 0x89;
// Send out an SPI enable command
//Need to set up the various CS and other pins associated with the board

//Lower CS2
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x91; //lower CS u1 = "0x91" or u2 "0x89
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Read to Port Value of U2 Port B

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('A'); //"A"
data[2] = 0x03; //number of bytes to be sent out via SPI command
data[3] = 0x45; //first byte  //command byte
data[4] = 0x12; //second byte //register byte
data[5] = 0x00; //third byte  //register data byte
udpClient.Send(data, 6, "10.10.10.10", 2424);

receiveBytes = udpClient.Receive(ref RemoteIpEndPoint);
returnData = System.Text.Encoding.ASCII.GetString(receiveBytes);

ReturnUDPData(receiveBytes, RemoteIpEndPoint);

//Raise CS2 both CS are in idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x99; //"0x99"
udpClient.Send(data, 2, "10.10.10.10", 2424);
```

```
public void ReturnUDPData(byte[] UDPData, IPEndPoint RemoteIP)
{

string MacString;
string VersionNumber;
string UDPDataString;
double VoltageADC;

if ((UDPData.Length == 6))
//SPI packets that we are recieving back are this long
{
  //need to check for S first, then A, number of bytes, ignore two bytes
and then data byte to read
  if (UDPData[0] == 0x53)
//we are dealing with an SPI packet coming back
  {
     value = 0;
     value = UDPData[5];
      TextBox.AppendText("The value of PORTB U2 is : " +
      Convert.ToString(UDPData[5], 16) + "\r\n");
   }

}
```

Other function can be written to implement other registers on the device using the above two examples. The other registers are shown in the table below. By default IOCON.BANK is set to 0.

Register Addresses for the MCP23S17 for Communications

Address IOCON.BANK = 1	Address IOCON.BANK = 0	Access to:
00h	00h	IODIRA
10h	01h	IODIRB
01h	02h	IPOLA
11h	03h	IPOLB
02h	04h	GPINTENA
12h	05h	GPINTENB
03h	06h	DEFVALA
13h	07h	DEFVALB
04h	08h	INTCONA
14h	09h	INTCONB
05h	0Ah	IOCON
15h	0Bh	IOCON
06h	0Ch	GPPUA
16h	0Dh	GPPUB
07h	0Eh	INTFA
17h	0Fh	INTFB
08h	10h	INTCAPA
18h	11h	INTCAPB
09h	12h	GPIOA
19h	13h	GPIOB
0Ah	14h	OLATA
1Ah	15h	OLATB

APPLICATIONS

Listed below are just a few applications the Relay board could be used for:

- Home/industrial automation
- Remote monitoring systems

ABSOLUTE MAXIMUM RATINGS

MCP23S17

Voltage on VDD with respect to VSS..... -0.3V to +5.5V
Voltage on all other pins with respect to VSS…..... -0.6V to VDD +0.6V
Storage temperature ..... -65°C to +150°C
Ambient temperature with power applied .....-40°C to +125°C
Total Power dissipation 700mW
Maximum Current out of VSS pin 150mA
Maximum Current into VDD pin 125mA
Maximum output current sunk by any output pin…….. 25mA
Maximum output current sourced by any output pin…….. 25mA

Care will need to be taken when connecting this device to the USB I/O 24 as you may exceed the maximum current draw allowed via the USB specifications.

FURTHER READING

Information about the 16 bit I/O Expander that is implemented on the GPIO Expander board can be found on the Microchip product datasheets for the devices. These can be downloaded from the Microchip website at www.microchip.com

Document Revision History

- GPIO 32 Board Datasheet V1.0– Initial document created

For further information regarding the release of this product please visit our website at **http://www.elexol.com** or contact us via email at **enquires@elexol.com**