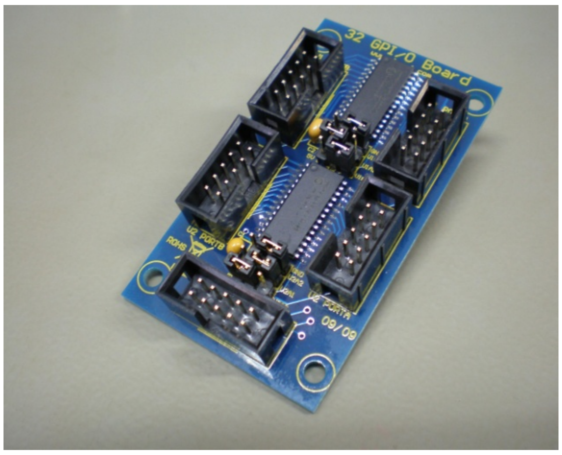
GPIO32 EXPANDER BOARD

FEATURES

- 2 X MCP23S17 GPIO Expander IC's
- 4 x I/O ports matching configuration of I/O 24
- Hardware addressable pins for each SPI device
- Easy connection to the I/O port via a 10-way box header that suits a standard IDC connector.
- 72mm Standard width for DIN Rail Modules



GENERAL DESCRIPTION

The 32 GPIO Expander Board is an accessory board that allows the implementation of an additional 32 GPIO (General purpose Input / Output) to be controlled from a single port on the existing 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 board consists of two MCP23S17 GPIO expanders, which are split up into four 8 bit Ports (2 Ports per MCP23S17).

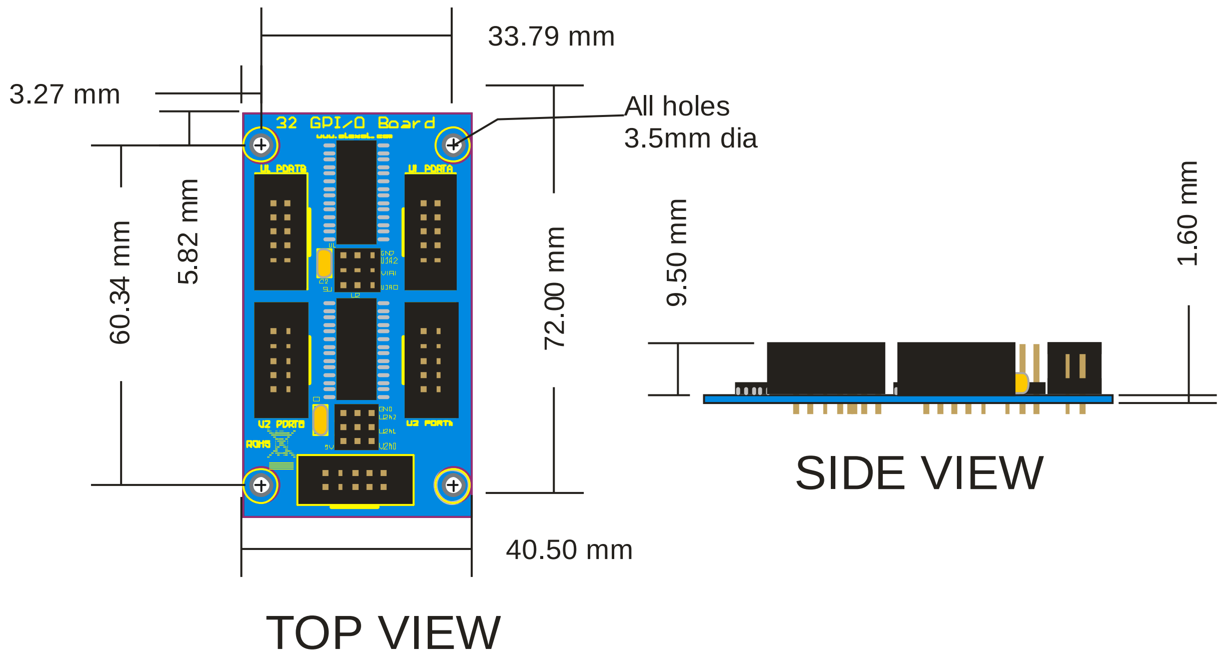
The four 8 bit ports use the same hardware configuration as the I/O 24 range. This makes it easier to connect other Elexol I/O boards to the 32 GPIO Expander board.

Communication to the GPIO's is via the SPI interface of the I/O 24 module which is implemented in the firmware. For further information on implementing the SPI interface please refer to the user manual for the I/O 24 module.

The connection between the I/O 24 module and the 32 GPIO Expander board is via a 30 cm IDC connection cable. This cable is provided with the board.

The board has been designed to a 72mm standard width so that it can easily be mounted in DIN rail mounting modules.

LAYOUT AND MECHANICALS

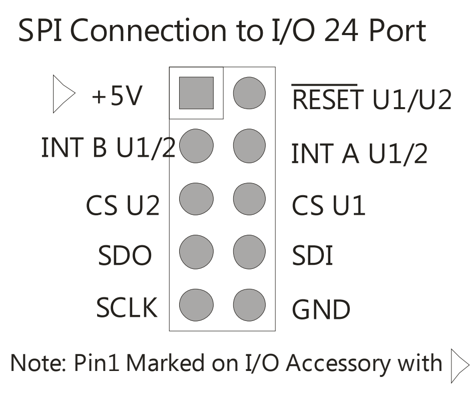


Dimensions: 1.59 X 2.8 X 0.43 inches (40.5 X 72 X 11mm)

PINOUTS AND BOARD CONNECTIONS

10 PIN BOX HEADER

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



SPI Connection to I/O 24 Port
+5V
RESET U1/U2
INT B U1/2
INT A U1/2
CS U2
CS U1
SDO
SDI
SCLK
GND
Note: Pin1 Marked on I/O Accessory with

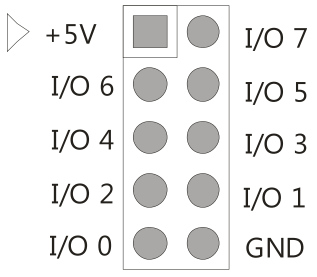
Listed in the table below are the connections for the 10 Pin Box Header

PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+3.3V to +5V drawn from I/O module powers (Supplies power to the I/O Board)
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 U1	I	Chip Select U1 of MCP23S17
7	SDO	O	Serial Data Out
8	SDI	I	Serial Data In
9	CLK	I	Serial Clock Input
10	GND	PWR	Ground signal from I/O module

10 PIN BOX HEADER

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

I/O 24 Port Connection



Note: Pin1 Marked on I/O Accessory with ▷

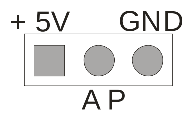
I/O 24 BOX HEADER CONNECTIONS

PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+3.3V to +5V drawn from I/O module powers (Supplies power to the connected I/O Board)
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

HARDWARE ADDRESS CONFIGURATION (U1/U2 HARDWARE ADDRESS JUMPERS)

There are 6 hardware address jumpers in total, located on the GPIO Expander Board. These jumpers are split into two lots of three, with each group handling the hardware addressing of each MCP23S17.

Hardware Address Configuration
Jumper



PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V	PWR	+5V Power pin from I/O module powers
2	AP	I	Address Pin connected to MCP23S17 for Hardware addressing. This pin needs to be jumpered to either 5V or GND depending on the hardware address required
3	GND	PWR	Ground signal from I/O module

These Address pins will take effect when the HAEN bit on the MCP23S17 is enabled not before hand.

NOTE: Both HAEN bits in the IOCON command register need to be set in the MCP23S17 before the hardware address takes effect.

COMMUNICATIONS

SETTING UP THE PORT ON THE I/O 24 FOR SPI COMMUNICATIONS TO THE 32 GPIO EXPANDER BOARD

Communication to the GPIO Expander board is via SPI from the port of I/O 24. However the port direction and pins will need to be setup before communication can begin with the board. Below are the configurations that need to be setup on the I/O 24 for the GPIO Expander board.

I/0 24 COMMAND	DESCRIPTION
!A 0x64	Initialise PORT Direction on I/O 24
A 0x99	Set CS for GPIO’s to idle state
A 0x91	Lower CS for GPIO U1 and U2 to idle state
A 0x89	Lower CS for GPIO U2 and U1 to idle state

CONTROL BYTES AND COMMAND REGISTER

There are a number of control bytes / command registers that are used in order to communicate with the GPIO Expander board. These control bytes /command registers along with their configuration bits are outlined below.

CONTROL BYTE CONFIGURATION FOR MCP23S17

CONTROL BYTE FOR MCP23S17								
Bit	7	6	5	4	3	2	1	0
Configuration Bit	0	1	0	0	A2	A1	A0	R/W

Configuration bit 7-1 SLAVE ADDRESS FOR GPIO

Bits 7-4 are fixed values
Bits 3-1 are defined from the Hardware Address jumpers

Configuration bit 0 READ / WRITE ENABLE

R/W = 1 = Read
R/W = 0 = Write

COMMAND REGISTER CONFIGURATIONS FOR MCP23S17 COMMUNICATIONS

ADDRESS IOCON.BANK = 1	ADDRESS IOCON.BANK = 0 (Default)	COMMAND REGISTER
0x00	0x00	IODIRA
0x10	0x01	IODIRB
0x01	0x02	IPOLA
0x11	0x03	IPOLB
0x02	0x04	GPINTENA
0x12	0x05	GPINTENB
0x03	0x06	DEFVALA
0x13	0x07	DEFVALB
0x04	0x08	INTCONA
0x14	0x09	INTCONB
0x05	0x0A	IOCON
0x15	0x0B	IOCON
0x06	0x0C	GPPUA
0x16	0x0D	GPPUB
0x07	0x0E	INTFA
0x17	0x0F	INTFB
0x08	0x10	INTCAPA
0x18	0x11	INTCAPB
0x09	0x12	GPIOA
0x19	0x13	GPIOB
0x0A	0x14	OLATA
1Ah	0x15	OLATB

The command registers listed in the table above are used in the SPI communication to the GPIO Expander device.

BASIC OVERVIEW OF SPI COMMUNICATIONS WITH THE GPIO EXPANDER BOARD

Listed below is a basic overview of the procedure that needs to be followed in order to communicate with the board.

1. Setup Direction Register on I/O 24 for GPIO Board (Not required for Ether IO24 PIC R)
2. Setup Port Values on I/O 24 for idle state on GPIO Board (Not required for Ether IO24 PIC R)
3. Lower the CS pin for chip communication
4. Send Control Byte and command registers
5. Send Null Bytes if response is required. This is done to keep the SPI clock going or no responses will be clocked in
6. Raise the CS pin back to idle state.
7. Repeat from Step 3 for other commands.

When using the Ether IO24 PIC R, steps 1 and 2 are handled by setting the Port Mode to SPI Mode in the Power-up Settings.

The coding examples below go into more detail of what commands are issued to the board.

CODING EXAMPLES

Outlined below are some standard commands that can be sent to the GPIO Expander board for the Ether I/O 24. The example code has been written in Visual C# Express.

These examples are available for download from our website www.elexol.com

Ether IO 24 Commands

Setting up Ether IO24 R Port A for GPIO Expander Board

```
private void Send_Enable_SPI()
{
UdpClient udpClient = new UdpClient();
IPEndPoint EtherIPEndPoint = new IPEndPoint(_deviceIP, _devicePort);
byte[]data = new byte[16];

// 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, EtherIPEndPoint);

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

udpClient.Send(data, 2, EtherIPEndPoint);

//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, EtherIPEndPoint);
}
```

(The above code is not required for the Ether IO24 PIC R. For this module you will need to set the Port Mode to SPI Mode in the Power-up Settings)

Write 0xAA to Port B of GPIO U1

```
private void button1_Click(object sender, EventArgs e)
{
UdpClient udpClient = new UdpClient();
IPEndPoint EtherIPEndPoint = new IPEndPoint(_deviceIP, _devicePort);
byte[]data = new byte[16];

// 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 CS1
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x91; //lower CS U1 = "0x91" or U2 "0x89
udpClient.Send(data, 2, EtherIPEndPoint);

//Write to Port Direction of U1 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; // command byte
data[4] = 0x01; // register byte
data[5] = 0x00; // register data byte
udpClient.Send(data, 6, EtherIPEndPoint);

//Raise CS1 both CS are in idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x99; //"0x99"
udpClient.Send(data, 2, EtherIPEndPoint);

//Read 6 SPI bytes back from stream
data = udpClient.Receive(ref EtherIPEndPoint);

//Lower CS1
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x91; //"0x91" or "0x89
udpClient.Send(data, 2, EtherIPEndPoint);

//Write to Value of U1 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; // command byte
data[4] = 0x13; // register byte
data[5] = 0xAA; // register value to be written
udpClient.Send(data, 6, EtherIPEndPoint);

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

//Read 6 SPI bytes back from stream
data = udpClient.Receive(ref EtherIPEndPoint);
}
```

Read Port B of GPIO U1

```
// 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, EtherIPEndPoint);
```

```
//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] = 0x41; // command byte
data[4] = 0x12; // register byte
data[5] = 0x00; // data byte
udpClient.Send(data, 6, EtherIPEndPoint);

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

//Read 6 SPI bytes back from stream
data = udpClient.Receive(ref EtherIPEndPoint);

//Port B Value is returned in data[5]

TextBox.AppendText("The value of PORTB U2 is : " +
Convert.ToString(data[5], 16) + "\r\n");

}
```

ABSOLUTE MAXIMUM RATINGS

<u>MCP23S17</u>

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

NOTE: 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 GPIO Expanders (MCP23S17) 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

- 32 GPIO Expander Board Datasheet Revision 1 – Initial document created