Analog I/O Board

The Analog I/O Board is an accessory board that allows the implementation of an analog interface to Elexol's 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.

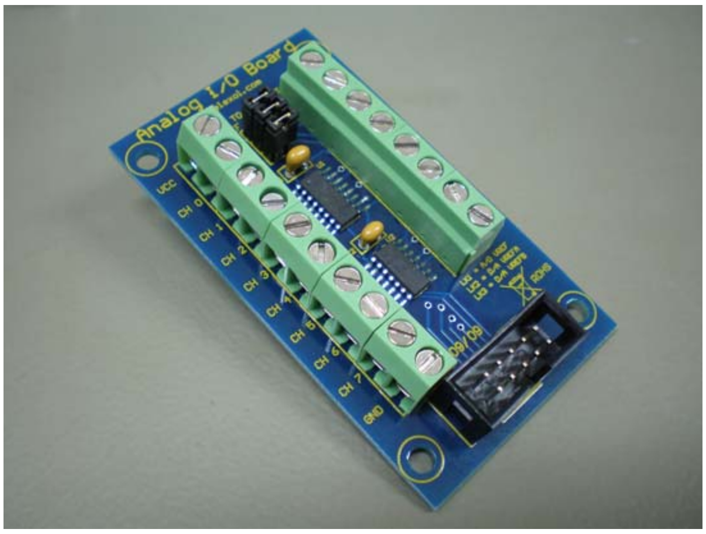


Diagram 1

The board consists of an 8 channel 10 bit Analog to Digital Converter (MCP3008) and a 12 bit Digital to Analog Converter with 2 buffered outputs (MCP4922). Communication to the A/D and D/A Converters 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.

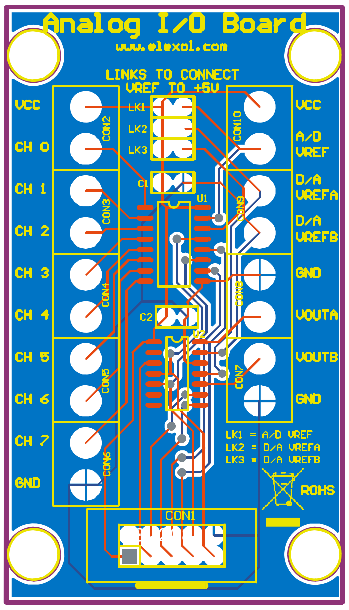
The 8 channels of the A/D, 2 channels outputs of the D/A, and all the VREF voltages are connected to screw terminals for easy access. These screw terminals will accept cables 0.5 – 2mm[1]. The connection between the I/O24 module and the Analog I/O board is via a 30 cm IDC connection cable. This cable 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.

BOARD FEATURES

- 1 x 8 Channel 10 bit Analog to Digital Converter (MCP3008)
- 1 x 12 bit Digital to Analog Converter with 2 Buffered Outputs
- Screw Terminal Blocks for Analog Inputs and Outputs
- Easy connection by 10-way box header to suit standard IDC connector for connection to the I/O port.
- 72mm Standard width for DIN Rail Modules

BOARD LAYOUT AND PHYSICAL DIMENSIONS

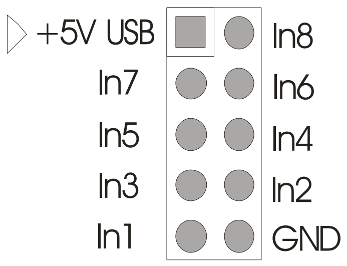


Dimensions - 3.1 X 2.8 X 1 inches (78.9 X 72 X 25.4mm)

BOARD CONNECTIONS

10 pin Box Header Pin out

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



Note: Pin1 Marked on I/O Accessory with ▷

Listed in Table 1 below are the connections for the 10 Pin Box Header

10 Pin Box Header Pin out Table

PIN #	SIGNAL	TYPE	DESCRIPTION
1	+5V USB	PWR	+3.3V to +5V drawn from I/O module powers (Supplies power to the Analog I/O Board)
2	N/C	I	Not Connected (N/C)
3	CS D/A	I	CS is the chip select input for D/A, which requires an active-low signal to enable serial clock and data functions.
4	SHD D/A	I	SHDN is the hardware shutdown input for D/A, that requires an active-low input signal to configure the D/A's in their low-power Standby mode.
5	LDAC D/A	I	LDAC (the latch DAC syncronization input) transfers the input latch registers to the DAC registers (output latches) when low. Can also be tied low if transfer on the rising edge of CS is desired.
6	CS A/D	I	The CS/SHDN input pin for the A/D is used to initiate communication with the device when pulled low. When pulled high, it will end a conversion and put the device in low power standby. The CS/SHDN pin must be pulled high between conversions.
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

TABLE 1

SCREW TERMINAL CONNECTIONS

SIGNAL	TYPE	DESCRIPTION
+5V	PWR	+3.3V to +5V drawn from I/O module powers (Supplies power to the Analog I/O Board)
CH 7	I	Analog Input for Channel 7
CH 6	I	Analog Input for Channel 6
CH 5	I	Analog Input for Channel 5
CH 4	I	Analog Input for Channel 4
CH 3	I	Analog Input for Channel 3
CH 2	I	Analog Input for Channel 2
CH 1	I	Analog Input for Channel 1
CH 0	I	Analog Input for Channel 0
GND	PWR	Ground signal from I/O module
VOB	O	DAC Channel B Output Voltage
GND	PWR	Ground signal from I/O module
VOA	O	DAC Channel A Output Voltage
D/A VREFA	I	DAC Channel A Reference Voltage Input (AVSS to VDD)
D/A VREFB	I	DAC Channel B Reference Voltage Input (AVSS to VDD)
A/D VREF	I	ADC Reference Voltage Input (AVSS to VDD)
GND	PWR	Ground signal from I/O module

JUMPER CONNECTIONS (VREF DEFAULT RANGE JUMPERS)

There are 3 jumpers located on the Analog I/O Board designated LK1 to LK3. These jumpers are placed so that the Analog Input/output range (VREF) of the Channel can be set to read or output the full voltage range of the device.

The table below outlines which

JUMPER	DESCRIPTION
LK1	A/D VREF connection to 5V
LK2	D/A VREFB connection to 5V
LK3	D/A VREFA connection to 5V

If you are wishing to convert analog inputs/outputs at another voltage please check with the electrical characteristics of the device.

Note: For the A/D and D/A when VREF is reduced the LSB size is reduced accordingly.

COMMUNICATION

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 Analog I/O board.

USB I/O 24 Commands (This example is implemented on PORT A of the USB I/O 24.)

Control Byte Register for A/D channels

```
// Control Byte A/D         // SGL/DIFF  D2  D1  D0  X   X   X   X
// Control Byte CH0 = 0x80  //  1        0   0   0  X    X   X   X
// Control Byte CH1 = 0x90
// Control Byte CH2 = 0xA0
// Control Byte CH3 = 0xB0  //  1        0   1   1   x   x   x   x
// Control Byte CH4 = 0xC0
// Control Byte CH5 = 0xD0
// Control Byte CH6 = 0xE0
// Control Byte CH7 = 0xF0
```

Perform a Read on the Analog channel (CH0)

```
//Initialize Port directions
WriteDirection('A', 0x04); //set Port A to initialise A/D Board !A 0x04
//Initialise Port Values for A/D Board
// Set up SPI pins, CS
WriteValue('A', 0x79); //Enable the CS_n for A/D and D/A to idle state
(0111 1000)

//Perform Read on CH 0 A/D

WriteValue('A', 0x71); //Lower CS for U1
WriteValue('S', 0x01); //Send the Start Byte for U1 (0x01)
WriteValue('S', 0x80); //Send the Configure Byte to Read CH 0 on A/D
WriteValue('S', 0x00); //Send Null byte to read A/D value LSB
WriteValue('A', 0x79); //Idle CS for U1 and U2
```

```
temp = ReadValue_SPI(6); // Read the null bytes returned by SPI

portA = temp[3];
portA = portA & (byte)0x07; // ADC[11]-ADC[8]
portA = portA * 256 + temp[5]; // ADC[7]-ADC[0]
VoltageADC = (double)portA * 0.00488; // Conversion to Voltage from ADC
// with VRef set to 5V
TestTextBox.AppendText("The voltage read from PD 1 is : " + VoltageADC
+ " V" + "\r\n");

unsafe private byte[] ReadValue_SPI(UInt32 rxCount)
{
   UInt32 dwRet = 0;
   FT_STATUS ftStatus = FT_STATUS.FT_OTHER_ERROR;
   byte[] cBuf = new byte[rxCount];

   dwRet = 0;
   long timeout;
   timeout = 0;

   do
   {
      ftStatus = FT_GetQueueStatus(m_hPort, ref dwRet);
      timeout++;
   } while ((dwRet < rxCount) && (timeout < 100000));

   if (timeout == 100000)
   {
      MessageBox.Show("Timeout while reading " +
      Convert.ToString(ftStatus));
      return (cBuf);
   }

   fixed (byte* pBuf = cBuf)
   {
      ftStatus = FT_Read(m_hPort, pBuf, rxCount, ref dwRet);
   }

   if (ftStatus != FT_STATUS.FT_OK)
   {
     MessageBox.Show("Failed To Read " + Convert.ToString(ftStatus));
   }
   return (cBuf);
}

unsafe private void WriteValue(char port, byte value)
{
   UInt32 dwRet = 0;
   FT_STATUS ftStatus = FT_STATUS.FT_OTHER_ERROR;
   byte[] cBuf = new byte[5];

   cBuf[0] = (byte)port;
   cBuf[1] = value;
   fixed (byte* pBuf = cBuf)
   {
      ftStatus = FT_Write(m_hPort, pBuf, 2, ref dwRet);
   }
}
```

```
unsafe private void WriteDirection(char port, byte value)
{
   UInt32 dwRet = 0;
   FT_STATUS ftStatus = FT_STATUS.FT_OTHER_ERROR;
   byte[] cBuf = new byte[5];

   cBuf[0] = (byte)'!';
   cBuf[1] = (byte)port;
   cBuf[2] = value;
   fixed (byte* pBuf = cBuf)
   {
      ftStatus = FT_Write(m_hPort, pBuf, 3, ref dwRet);
   }
}
```

Write voltage level on D/A channel (VOUTA and VOUTB)

```
// Idle state  = 0x78
// Lower CS for D/A = 0x70
// Write Command Register MSB   // A/B  BUF  GA  SHDN  D11  D10  D9  D8
// Write Command Register LSB   // D7   D6   D5   D4   D3   D2   D1  D0

//Initialize Port directions
WriteDirection('A', 0x04); //set Port A to initialise A/D Board
// Set up SPI pins, CS
WriteValue('A', 0x78); //Enable the CS_n for A/D and D/A to idle state



//This will output 2.5V on both D/A channels VOUTA and VOUTB

//Write 2.5V VOUTB
WriteValue('A', 0x38); //Lower CS for D/A and set LDAC to high
WriteValue('S', 0xF7); //Send the MSB of Write Command Register for D/A
WriteValue('S', 0xFF); //Send the LSB of Write Command Register for D/A
temp = ReadValue_SPI(4); // Read the null bytes returned by SPI
WriteValue('A', 0x78); //Idle CS D/A and A/D
WriteValue('A', 0x68); //Lower LDAC pin
WriteValue('A', 0x78); //Idle CS D/A and A/D raise LDAC to high

//Write 2.5V VOUTA
WriteValue('A', 0x38); //Lower CS for D/A and set LDAC to high
WriteValue('S', 0x77); //Send the MSB of Write Command Register for D/A
WriteValue('S', 0xFF); //Send the LSB of Write Command Register for D/A
temp = ReadValue_SPI(4); // Read the null bytes returned by SPI
WriteValue('A', 0x78); //Idle CS D/A and A/D
WriteValue('A', 0x68); //Lower LDAC pin
WriteValue('A', 0x78); //Idle CS D/A and A/D raise LDAC to high
```

Ether I/O 24 Commands

Perform a Read on the Analog channel (CH0) on Ether with IP address 10.10.10.10

```
System.Net.IPEndPoint RemoteIpEndPoint = new
System.Net.IPEndPoint(System.Net.IPAddress.Any, 0);
byte[] receiveBytes;
string returnData;

double VoltageADC;


//Lower CS pin for A/D
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x71; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Perform Read on A/D CH 0 will receive UDP packet back from SPI
//containing data from conversion. This will need to be dealt with in
//data received thread

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] = 0x01; //first byte  //start byte
data[4] = 0x80; //second byte //command byte
data[5] = 0x00; //third byte //null byte
//udpClient.Send(data, 6, EtherIP);
udpClient.Send(data, 6, "10.10.10.10", 2424);
//Raise CS to idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x79; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

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

ReturnUDPData(receiveBytes, RemoteIpEndPoint);
```

```
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[4];
     value = value & (byte)0x07; // ADC[11]-ADC[8]
     value = (value * 256) + UDPData[5]; // ADC[7]-ADC[0]
     VoltageADC = (double)value * 0.00488;
// Conversion to Voltage from ADC // with VRef set to 5V
     TextBox.AppendText("The voltage read from PD 1 is : " + VoltageADC
+ " V" + "\r\n");
   }

}
```

Write voltage level on D/A channel (VOUTA and VOUTB)

```
//Lower CS pin for D/A
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x38; //"0x98"
//udpClient.Send(data, 6, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);
// VOUTB set to 2.5V

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('A'); //"A"
data[2] = 0x02; //number of bytes to be sent out via SPI command
data[3] = 0xF7; //first byte  //Write command register MSB
data[4] = 0xFF; //second byte //Write command register LSB
//udpClient.Send(data, 6, EtherIP);
udpClient.Send(data, 5, "10.10.10.10", 2424);

//Raise CS to idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x78; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Lower LDAC to output values
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x68; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);
```

```
//Raise CS to idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x78; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Lower CS pin for D/A
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x38; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

// VOUTA set to 2.5V

data[0] = Convert.ToByte('S'); //"S"
data[1] = Convert.ToByte('A'); //"A"
data[2] = 0x02; //number of bytes to be sent out via SPI command
data[3] = 0x77; //first byte  //Write command register MSB
data[4] = 0xFF; //second byte //Write command register LSB
//udpClient.Send(data, 5, EtherIP);
udpClient.Send(data, 5, "10.10.10.10", 2424);

//Raise CS to idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x78; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Lower LDAC to output values
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x68; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);

//Raise CS to idle state
data[0] = Convert.ToByte('A');//"A"
data[1] = 0x78; //"0x98"
//udpClient.Send(data, 2, EtherIP);
udpClient.Send(data, 2, "10.10.10.10", 2424);
```

Control Byte Register MCP3008 A/D for Communications

```
// Control Byte A/D         // SGL/DIFF  D2  D1  D0  X   X   X   X
// Control Byte CH0 = 0x80  //  1        0   0   0  X    X   X   X
// Control Byte CH1 = 0x90
// Control Byte CH2 = 0xA0
// Control Byte CH3 = 0xB0  //  1        0   1   1   x   x   x   x
// Control Byte CH4 = 0xC0
// Control Byte CH5 = 0xD0
// Control Byte CH6 = 0xE0
// Control Byte CH7 = 0xF0
```

Configure Bits for the MCP3008 A/D for Communications

Control Bit Selections				**Input Configuration**	**Channel Selection**
Single /Diff	**D2**	**D1**	**D0**		
1	0	0	0	Single-ended	CH0
1	0	0	1	Single-ended	CH1
1	0	1	0	Single-ended	CH2
1	0	1	1	Single-ended	CH3
1	1	0	0	Single-ended	CH4
1	1	0	1	Single-ended	CH5
1	1	1	0	Single-ended	CH6
1	1	1	1	Single-ended	CH7
0	0	0	0	Differential	CH0 = IN+ CH1 = IN-
0	0	0	1	Differential	CH0 = IN- CH1 = IN+
0	0	1	0	Differential	CH2 = IN+ CH3 = IN-
0	0	1	1	Differential	CH2 = IN+ CH3 = IN-
0	1	0	0	Differential	CH4 = IN+ CH5 = IN-
0	1	0	1	Differential	CH4 = IN+ CH5 = IN-
0	1	1	0	Differential	CH6 = IN+ CH7 = IN-
0	1	1	1	Differential	CH6 = IN+ CH7 = IN-

Write Command Register for the MCP4922 D/A for Communications

```
// Idle state  = 0x78
// Lower CS for D/A = 0x70
// Write Command Register MSB   // A/B  BUF  GA  SHDN  D11  D10  D9  D8
// Write Command Register LSB   // D7   D6   D5  D4    D3   D2   D1  D0
```

Write Command Register MSB

Write Command Register							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
A/B	BUF	GA	SHDN	D11	D10	D9	D8
Bit 15							Bit 8

Write Command Register LSB

Write Command Register							
W-x	W-x	W-x	W-x	W-x	W-x	W-x	W-x
D7	D6	D5	D4	D3	D2	D1	D0
Bit 7							Bit 0

bit 15 A/B: DACA or DACB Select bit
1 = Write to DACB
0 = Write to DACA
bit 14 BUF: VREF Input Buffer Control bit
1 = Buffered
0 = Unbuffered
bit 13 GA: Output Gain Select bit
1 = 1x (VOUT = VREF * D/4096)
0 = 2x (VOUT = 2 * VREF * D/4096)
bit 12 SHDN: Output Power Down Control bit
1 = Output Power Down Control bit
0 = Output buffer disabled, Output is high impedance
bit 11-0 D11:D0: DAC Data bits
12 bit number "D" which sets the output value. Contains a value between 0 and 4095.

APPLICATIONS

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

- Voltage level sensing
- Wave Form Generation
- Home Automation

Absolute Maximum Ratings

MCP3008

VDD	7.0V
All inputs and outputs w.r.t. VSS	-0.6V to VDD +0.6V
Storage temperature	-65°C to +150°C
Ambient temp. with power applied	-65°C to +125°C
Soldering temperature of leads (10 seconds)	+300°C
ESD protection on all pins	> 4 kV

MCP4922

VDD	6.5V
All inputs and outputs w.r.t	AVSS –0.3V to VDD+0.3V
Current at Input Pins	±2 mA
Current at Supply Pins	±50 mA
Current at Output Pins	±25 mA
Storage temperature	-65°C to +150°C
Ambient temp. with power applied	-55°C to +125°C
ESD protection on all pins	≥ 4 kV (HBM), ≥ 400V (MM)
Maximum Junction Temperature (TJ) .	+150°C

Further Reading

Information about the Analogue to Digital and Digital to Analogue Converters 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

- Analog I/O Board Datasheet V1.0– Initial document created