USBIO24 R Digital I/O Module

The USBIO24 R is the ROHS Compliant version of our previous USB I/O 24 range. The USB I/O 24 R is a low-cost integrated module for the input and/or output of digital signals from a computer system by connection to the USB port. The modules I/O Port pin out and firmware are compatible with the previous versions of the USB I/O24 modules.

The module features 24 5V level signal lines individually programmable as input or output. As the module connects to the USB port, multiple modules can be connected to a single PC by the use of a USB hub or hubs. Each module features a serial number and the PC can identify each module uniquely allowing for multiple modules to be connected for a single application. The outputs of the module are able to source or sink up to 30mA per I/O, up to a maximum 50mA per port, to allow for direct connection to a variety of devices.

The USB I/O 24 R Module

MODULE FEATURES

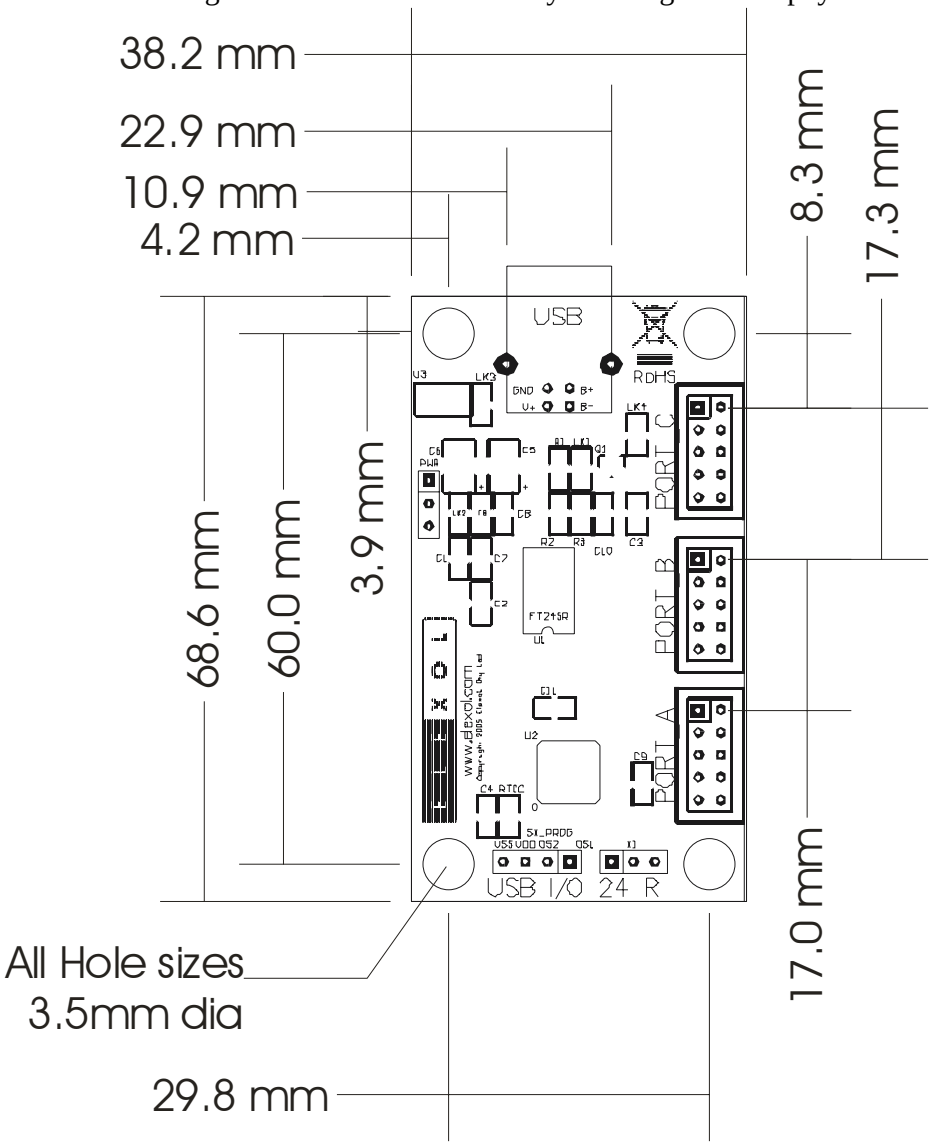
- 24 independently programmable Input / Output Pins Grouped into 3 ports.
- Single module High-Speed Digital Input / Output solution.
- Integrated Type-B USB Connector.
- On-board unique serial number and custom programmable FLASH microcontroller.
- Both USB Enumeration information and Microcontroller can be re-programmed to suit customer needs.
- Module powered by the USB from the PC or by an external power supply.

Module Standard Firmware & Software

- Virtual COM Port driver allows access as a regular serial port.
- Optional DLL based driver available.
- Easy to program using popular development languages C, Basic, Delphi, VB, etc.
- Simple command set for easy control of ports and data transfer.

Module Layout and Physical Dimensions

Shown in the diagram below is the module layout along with the physical dimensions of the unit.

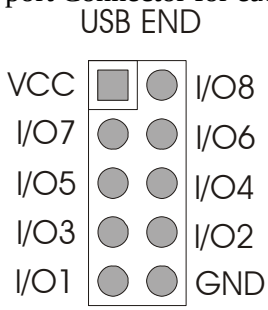


Module Connections

The USB I/O 24 module has 5 user connections and 1 programming connection.

I/O Port Connector Pin out

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



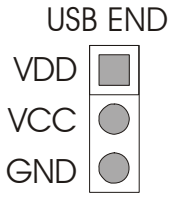
USB I/O 24 I/O Connector (PORT A, B, C)

Listed in Table 1 below are the connections for the I/O port connector. Ports A,B and C have identical connections.

PIN #	SIGNAL	TYPE	DESCRIPTION
1	VCC	PWR	+3.3V to +5V depends on powering SX48 option – May be used to supply your circuitry Note A maximum of 50mA can be drawn when powered through USB
2	I/O8	I/O	Programmable I/O pin with bit value of 128
3	I/O7	I/O	Programmable I/O pin with bit value of 64
4	I/O6	I/O	Programmable I/O pin with bit value of 32
5	I/O5	I/O	Programmable I/O pin with bit value of 16
6	I/O4	I/O	Programmable I/O pin with bit value of 8
7	I/O3	I/O	Programmable I/O pin with bit value of 4
8	I/O2	I/O	Programmable I/O pin with bit value of 2
9	I/O1	I/O	Programmable I/O pin with bit value of 1
10	GND	PWR	Ground signal USB BUS and all I/O

PWR Connector Pinout

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



USB I/O 24 Port Expansion Connector (PORT EXP)

PIN #	SIGNAL	TYPE	DESCRIPTION
1	VDD	PWR	+3.3V to +5V depends on powering SX48 option – May be used to supply your circuitry Note A maximum of 50mA can be drawn when powered through USB
2	VCC	PWR	+3.3V to 5V VCC supply to FT245R when in self-powered mode. Note: In USB powered mode this pin is linked to pin 1 ** **NO** External power needs to be applied when in Bus powered mode**
3	GND	PWR	Ground signal USB BUS and all I/O

Communication Protocol

Listed in the tables below are the communication protocols used to operate the module. The commands that are sent to the unit are in ASCII format and all data that is sent is in Binary format.

Standard Firmware

COMMAND	DATA	FUNCTION
'?'	Responds 'USB I/O 24f2'	Identify Device
'A'	1 Byte Port Data	Write to Port A
'B'	1 Byte Port Data	Write to Port B
'C'	1 Byte Port Data	Write to Port C
'a'	Responds with 1 Byte Port Data	Read from Port A
'b'	Responds with 1 Byte Port Data	Read from Port B
'c'	Responds with 1 Byte Port Data	Read from Port C
'!A'	1 Byte Port I/O Data	Write to Port A Direction Register
'!B'	1 Byte Port I/O Data	Write to Port B Direction Register
'!C'	1 Byte Port I/O Data	Write to Port C Direction Register

The commands in the above table are in ASCII format. All Data is sent in Binary format.

Additional VERSION 3 and R Firmware

- Transmits on Pin Change without Software Polling (Mode 2)
- Enables setting of the pin pull up feature on SX48
- Enables setting of inputs as CMOS Level or TTL Level or Schmitt Trigger Inputs
- Single I/O pin commands
- SPI interface (available on all ports.)
- Temperature interface to a DS18B20 (available on all I/O pins)

COMMAND	DATA	FUNCTION
'?'	Responds 'USB I/O 24'	Identify Device
'A'	1 Byte Port Data	Write to Port A
'B'	1 Byte Port Data	Write to Port B
'C'	1 Byte Port Data	Write to Port C
'a'	Responds with 1 Byte Port Data	Read from Port A
'b'	Responds with 1 Byte Port Data	Read from Port B
'c'	Responds with 1 Byte Port Data	Read from Port C
'!A'	1 Byte Port I/O Data	Write to Port A Direction Register
'!B'	1 Byte Port I/O Data	Write to Port B Direction Register
'!C'	1 Byte Port I/O Data	Write to Port C Direction Register
'H'	1 Byte specifying which I/O pin (0-23)	Set specific I/O Pin High
'L'	1 Byte specifying which I/O pin (0-23)	Set specific I/O Pin Low
'#'	Follow with Port write and binary data	Port Pull up feature for SX48
'@'	Follow with Port write and binary data	Set inputs to CMOS/ TTL level
'$'	Follow with Port write and binary data	Set Port Schmitt trigger enables
'2'		Sets unit to Mode 2 (Enables transmit on pin change)
'1'		Sets unit to Mode 1 (Exits Mode 2)

The commands in the above table are in ASCII format. All Data is sent in Binary format.

Pull Up

The Pull Up configuration applies to those lines that are set as inputs, writing a 0 to the corresponding bit applies a pull up resistor to the line so that if it is not driven low it will be pulled to a known high state, this is very useful if sensing contact closures or open collector outputs

Threshold

The threshold function sets the threshold at which a line reads as high or low. When the corresponding bit is set as 1 then the threshold is set at 1.4V and any voltage above this reads as a high level. When the corresponding threshold bit is set to 0 the threshold is set at 2.5V and any voltage above this reads as a high level.

Schmitt

Schmitt trigger inputs means that the input line is compared to 2 voltages, 0.75V and 4.25V. When the line's voltage drops below 0.75V it will read as a low until the line's voltage rises above 4.25V at which time the line will read as a high. When the line's voltage is in between 0.75V and 4.25V, the value will remain stable at its previous level. To enable the Schmitt trigger on any input a 0 must be written to the corresponding bit

Mode 2 Functional Changes

All reads have a port designator ('a', 'b' or 'c') before the data.
All auto sends have a port designator ('a', 'b' or 'c') before the data.
All writes to the port that change the port will results in a port data auto send.

SPI Interface

Send / Receive SPI Byte on PORT A

COMMAND	DATA	REPLY	FUNCTION
'S'	1 Byte of data sent on SPI Bus	1 Byte of data received on SPI Bus	Send / Receive SPI Byte on PORT A

Send / Receive SPI Byte on PORT B

COMMAND	DATA	REPLY	FUNCTION
'T'	1 Byte of data sent on SPI Bus	1 Byte of data received on SPI Bus	Send / Receive SPI Byte on PORT B

Send / Receive SPI Byte on PORT C

COMMAND	DATA	REPLY	FUNCTION
'U'	1 Byte of data sent on SPI Bus	1 Byte of data received on SPI Bus	Send / Receive SPI Byte on PORT C

Notes on SPI
Bit 1 is always Clock and must be set as an output from the SPI to function
Bit 2 is always Serial Data Out (MOSI) and must be set as an output from the SPI to function
Bit 3 is always Serial Data In (MISO) and must be set as an input from the SPI to function
All other port pins act as normal
Setting the port pin as high or low will set the clock as normally high or low before the SPI transaction begins.
Any of the other pins can be used as SS or CE when SPI is used.

NOTE: This interface will not work with the 50MHz upgrade.

Setting up SPI

1. Set bit 3 for input (Serial Input) rest of port pins output
   example for VCP MSComm1.Output = "!A" + Chr$(4)

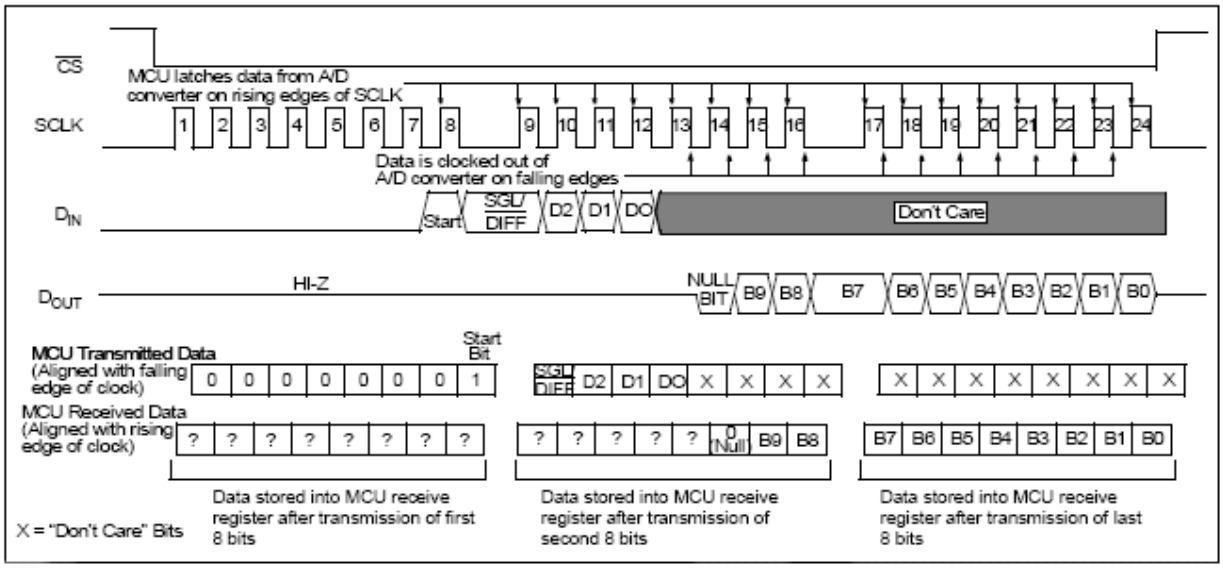
2. Sending bytes out via SPI
For every byte that is sent out via SPI it requires and "S" prefix. If you wanted to send out the following bytes $FF $AA $55 you would do the following

```
MSComm1.Output = "S" + Chr$(255)    //S FF
MSComm1.Output = "S" + Chr$(170)    //S AA
MSComm1.Output = "S" + Chr$(85)     //S 55
```

3. Each byte returned from the SPI will have a preceding "S" character and then the byte value from the SPI device.

4. To read the value returned from the device you will have to send the SPI device null characters to keep the SPI clock going. For example if you are expecting a certain number of bytes in return to a command which was sent, you send "S" + null character for the number of bytes you wish to read. In the example image, this can be seen in the last two bytes which are sent (implemented with don't cares) and on the input pin you will have the valid data from the device.

Note: you may have to send out leading zero's and don't cares depending on device you are communicating with. e.g. seven leading zero's then a start bit as per the image below.



Temperature Sensor Interface using DS18B20

Convert Start

COMMAND	DATA	REPLY	FUNCTION
'J'	1 Byte specifying which pin 0 to 23	1 Byte Status of Command	Start Conversion

Read Temperature

COMMAND	DATA	REPLY	FUNCTION
'K'	1 Byte specifying which pin 0 to 23	3 Bytes Byte 1 – Status (see below) Byte 2 – MSB of Temperature Byte 3 – LSB of Temperature	Read Temperature

Notes on the DS18B20

Status Bytes

G	Command execute normally
C	A CRC error occurred while reading the data
N	No sensor found
E	Bad pin number (only pins 0 to 23 are valid)
S	Bus Shorted (or pull up resistor missing)

The DS18B20 requires a resistor between 1K and 4.7K between the data line and the +5V rail. A DS18S20 may also be used if the higher resolution of the DS18B20 is not required.

NOTE: This interface will not work with the 50MHz upgrade.

50MHz Upgrades

Upgrades are available to run the SX48 at 50MHz clock speed instead of using the internal 4MHz clock.

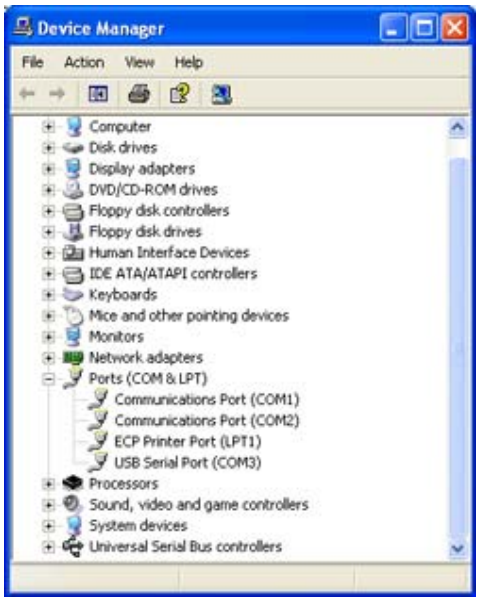
Accessory Boards

Accessory boards are available for this product. Please check the Elexol website for further information.

Driver Installation

Your first choice when using the USBIO24 is whether you want to use the Virtual COM Port driver or the Direct DLL driver.

For programming simplicity the best driver is the Virtual COM Port and when installed the USBIO24 will appear in the System Properties / Device Manager as an USB Serial Port (COMn) as follows.



The COM port number will vary depending on the number of existing COM ports on your computer and the number of USBIO24 s or USBMODs connected to your system.

To install the Virtual Com Port drivers, download the driver from our website or the ftdichip.com website and unzip the files to a local directory. Connect the USBIO24 and windows will automatically ask for the driver, select to specify a location and browse to the directory where you have unzipped the files. (Use of the Non Plug & Play driver for the USBIO24 is recommended to avoid a delay on connecting the USBIO24)

Once the Virtual COM Port is installed it can be programmed exactly as a regular serial COM port using the MSComm control from within VB or API calls from C or other languages. Set the COM port to the same number as appears in the Device Manager, the baud rate, stop bits, parity etc are not used as the device always runs at full speed.

The Direct DLL driver is installed in a similar manner but using the alternative download from the website.

Programming the Direct DLL driver is by call to the DLL Library functions.
Please download the Direct DLL programmers guide from the FTDI website.

NOTE: Only one driver for the device can be installed at any one time.

Programmers Reference Documentation

Programming the USBIO24 from Visual Basic using the Virtual COM Port.

To operate the USBIO24 from within Visual Basic it is best to use the Microsoft MSComm control to access the COM port. To input data from the USBIO24 you must use the port in binary mode and receiving the data is a bit convoluted.

Opening the Port

As the USBIO24 unit uses binary data transfer we must use the port in binary mode.
If the port number is incorrect or the USBIO24 module is not connected then VB will generate an error.

```
MSComm1.CommPort = 3 ‘ Set this number as shown in the Device Manager
MSComm1.InputMode = comInputModeBinary ‘ Set Binary Input Mode
MSComm1.PortOpen = True ‘ Open the Port
```

Setting the pins as Inputs or Outputs

Setting the Ports as Input or Output you must determine the value for the pins you want set as inputs.
To set pins I/O1, I/O2 and I/O3 as inputs and the remaining pins as outputs you simply add the bit values of the input pins 1 + 2 + 4 = 7 and thus the value to be placed in the IOValx variable in the following example code is 7.

```
‘ Set I/O1, I/O2 & I/O3 of port A to inputs and the rest as outputs.

IOValA = 7 ‘ First 3 inputs all the rest as outputs
IOValB = 0 ‘ All outputs
IOValC = 0 ‘ All outputs
MSComm1.Output = "!A" + Chr$(IOValA) ' Write to Port A Direction Register
MSComm1.Output = "!B" + Chr$(IOValB) ' Write to Port B Direction Register
MSComm1.Output = "!C" + Chr$(IOValC) ' Write to Port C Direction Register
```

Writing to the Ports

To write to the Output Pins simple repeat the Above without the ! character.
The following example code sets the I/O8 pin on port B to high and the remaining pins as low.

```
IOValB = 128 ‘ I/O8 high, all the rest low
MSComm1.Output = "B" + Chr$(IOValB)
```

Reading from the Ports

To Read the Input Pins is a bit more complicated as we must request the data from the module, wait for it to arrive and then retrieve the data from the port. The following example reads the value from port A.

```
MSComm1.Output = "a" ‘ Request data from port A

T = Timer ‘ Use the Timer to allow the program to continue if there is an error
   While Timer < T + 0.4 And MSComm1.InBufferCount < 1
   Wend
If MSComm1.InBufferCount <> 1 Then
    Call MsgBox("Read Timeout", vbInformation, "USBIO24 VERSION 3 Error")
    Exit Sub
    End If
Dim TempBuffer As Variant
Dim ByteBuffer() As Byte
TempBuffer = MSComm1.Input
ByteBuffer = TempBuffer
PortAVal = ByteBuffer(0)
```

Hardware Configurations outlined on the schematic for the USB I/O 24 R

The USB I/O 24 R has a number of different hardware configurations that can be implemented. Each of the configurations are listed below:

Bus Powered (FTDI Chip)

The bus powered mode draws power from the USB Bus. This option is the current configuration that is used when the USB I/O 24 R is sent out.

Self Powered (FTDI Chip)

The self-powered mode of operation draws power from its own external Power supply and doesn't draw any current from the USB Bus.

Note: The FTDI chip can only be bus powered or self powered

Powering SX 48 Options

There are a number of different ways that the UBICOM SX48 micro controller can be powered. These various options are shown below.

Note: When the SX48 is powered from the USB this also means that the USB Bus will also power the FTDI FT245BM.

USB 5V

The SX48 draws it power from the USB Bus.

USB 3.3V

The SX48 draws it power from the USB Bus, but a 3.3V regulator regulates the voltage provided by the USB. Note: This 3.3V regulator is not currently placed on the board and is only provided as a separate option.

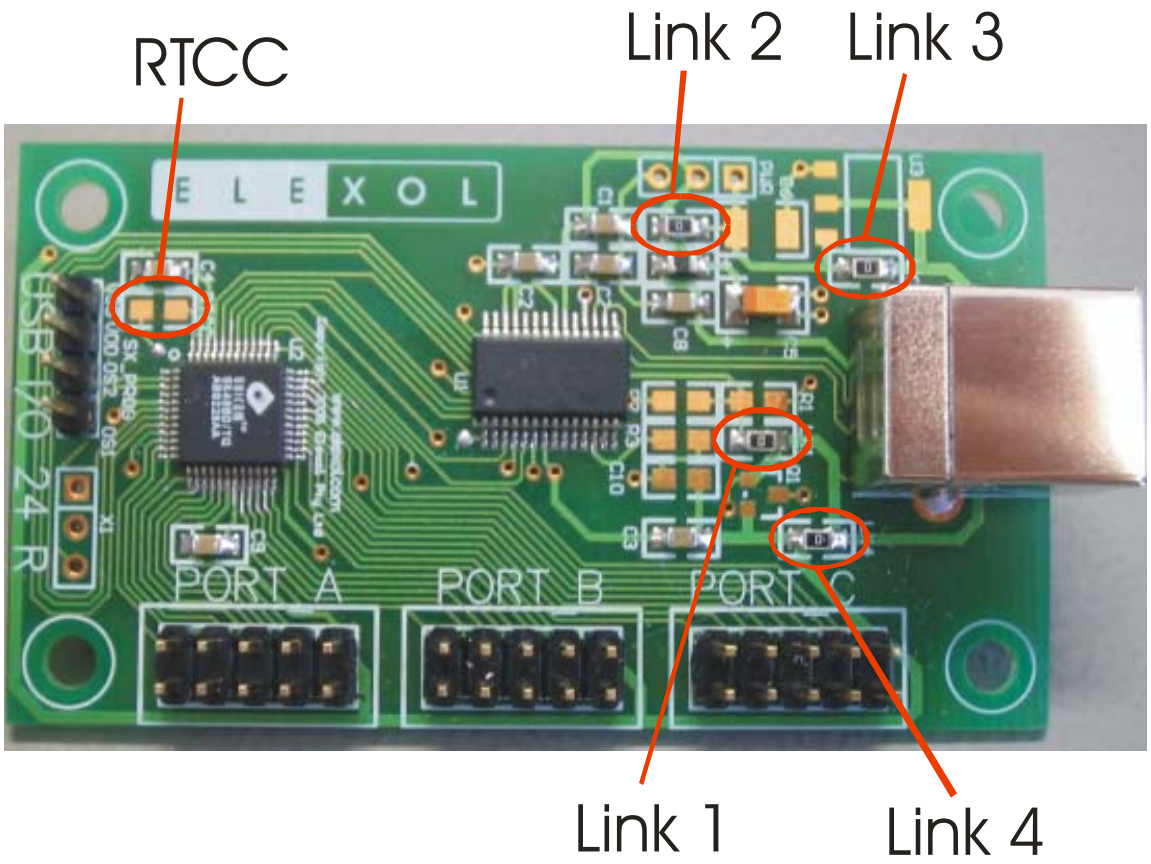
External 5V

The SX48 draws it power from its own external power supply

External 3.3V

The SX48 draws it power from its own external power supply, but the voltage provided by the power supply can be either set to the desired 3.3V or regulated by a 3.3V regulator. Note: This 3.3V regulator is not currently placed on the board and is only provided as a separate option.

Location of Links on USB I/O 24 R PCB for various configuration setups



Link 1 = remove link to use FTDI chip in Self Powered mode. Also remove components C8 and FB. Place components R1, R2. Apply an external power supply (5V MAX) to pin 2 of PWR Connector**.**

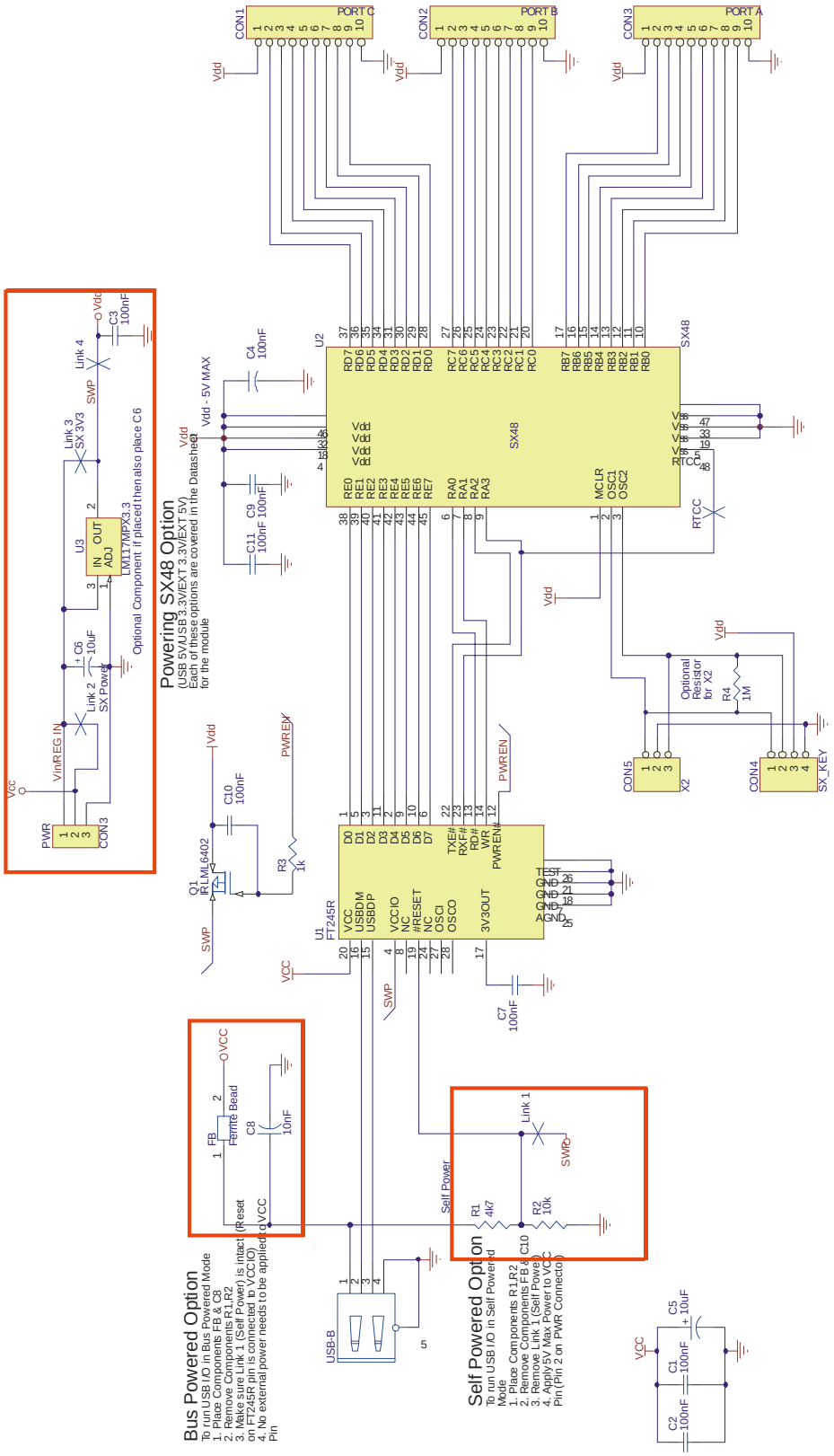
Link 2 = remove if the SX48 and the ports are to be powered by and external Power Supply. Apply an external power supply (5V MAX) to pin 1 of PWR Connector once the link has been removed**.**

Link 3 = used for to provide a direct connection to the External Power supply for the SX48 and ports. Remove link if placing the 3.3V Regulator

Link 4 =Links VSX Power rail, can be removed if the unit requires power switching to be used during USB Suspend. Note when using Power switching the soft start up circuit needs to be place (R3 and C10)

RTCC = Connected to the RTCC pin on the SX48.

Schematic


Powering SX48 Option
(USB 5V/USB 3.3V/EXT 3.3V/EXT 5V)
Each of these options are covered in the Datasheet
for the module
Optional Component if placed then also place C6
Bus Powered Option
To run USB I/O in Bus Powered Mode
1. Place Components FB & C8
2. Remove Components R1, R2
3. Make sure Link 1 (Self Power) is intact
on FT245R pin is connected to VCC/IO.
4. No external power needs to be applied to VCC
Pin
Self Powered Option
To run USB I/O in Self Powered
Mode
1. Place Components R1,R2
2. Remove Components FB & C10
3. Remove Link 1 (Self Power)
4. Apply 5V Max Power to VCC
Pin (Pin 2 on PWR Connector)
Optional
Resistor
for X2
SX48
FT245R
LM117MP3.3
IRLML6402
USB-B

Absolute Maximum Ratings

Subjecting the device to conditions outside these rating will invalidate the product warranty and may cause irreparable damage the device.

Maximum Voltage on Any Pin (referenced to GND)	5.1 V DC
Minimum Voltage on Any Pin (referenced to GND)	-0.1V DC
Supply current when USB enumerating	55mA
Supply current when running with no load on I/O pins	32mA
Total Current to or from Any Single I/O Pin	30mA
Total Current to or from Any PORT (8 Single I/O make a PORT)	50mA
Total Current in or out of all the module's I/O pins	200mA