
USB 25IO Digital/ADC/PWM - V1.1

Guide

9/11/07

The Mirrorbow USB 25IO Digital/ADC/PWM allows easy interfacing of custom devices from a PC without requiring any USB knowledge.

With the loss of serial and dedicated parallel ports from PCs it has got much harder to develop all those fun, challenging and wonderful projects that were accessible a few years ago. USB has been great for the average PC user but anyone wanting a simple interface to bridge their computer program with straightforward electronics has realised that in most cases, it is more complicated than the project itself, requiring in depth knowledge of windows DDK and substantial firmware! The Mirrorbow USB 25IO removes all of this complexity, allowing you to concentrate on your project rather than the interfaces.

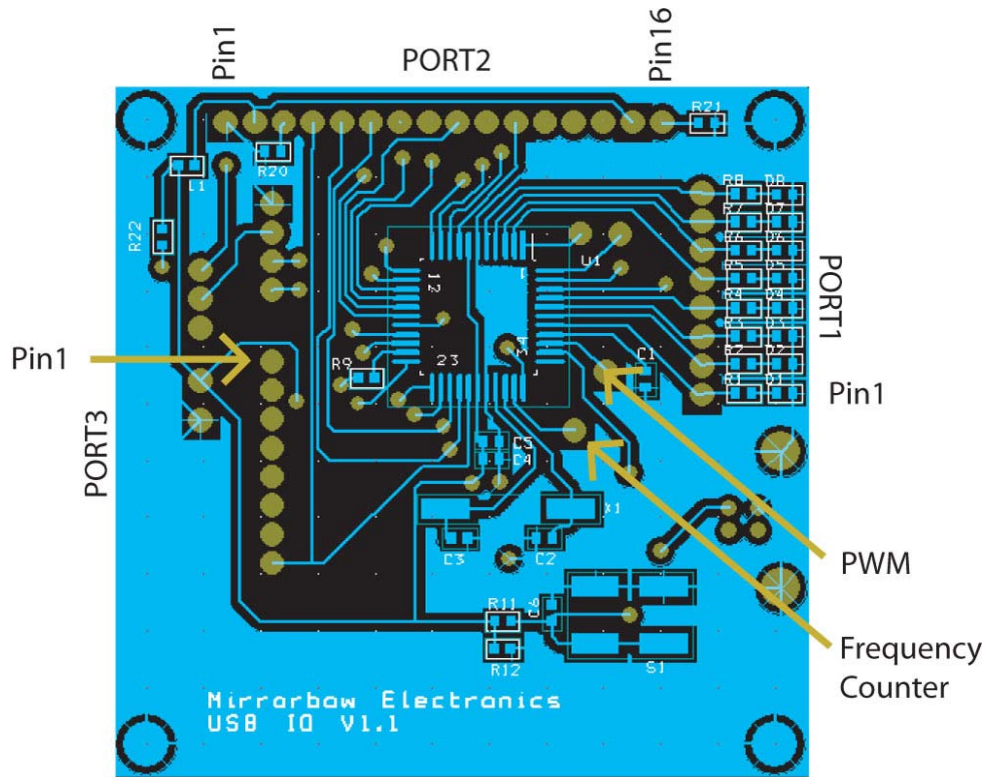
The Mirrorbow USB IO board interfaces via USB, but pretends to be a simple com port. Hence the host PC software simply sends commands over the serial interface, which is easily programmed and understood.

Warning: *As with any bare electronics It is wise to take static precautions when handling this board. Overloading IO outputs or exceeding 5V on inputs may cause permanent damage to the board. The maximum loading for any single output is 10mA, however total loading on all IO pins must not exceed 100mA. The 5V supply provided on the board is the USB supply, care must be taken not to exceed the available USB power which on a desktop PC is usually at least 400mA, but on a laptop may be much less.*

Compatible with Windows 2000 and Windows XP

Mirrorbow
Electronics

The Board



The board consists of 3 ports marked PORT1, PORT2 and PORT3.

Ports 1 and 3 are simple 8 bit ports, however Port 2 is extended to include power, 3 data bits from Port 3 as well as all 8 data bits from Port 2. This is to make it easier to connect a daughter board with the maximum flexibility. Port2 can also be turned into a servo controller for up to 8 servos.

Port3 bits 0 and 1 can be configured to convert the analogue voltage to a digital number (ADCs)

	PORT1	PORT2	PORT3
Pin1	1D0 - LED	GND (0V)	3D0 /A0
Pin2	1D1 - LED	+5V	3D1 /A1
Pin3	1D2 - LED	Do not connect	3D2 /A2
Pin4	1D3 - LED	3D7 (PORT3 copy) /A6	3D3
Pin5	1D4	3D6 (PORT3 copy) /A5	3D4 /A3
Pin6	1D5	3D5 (PORT3 copy) /A4	3D5 /A4
Pin7	1D6	2D0 (also Servo0) /A7	3D6 /A5
Pin8	1D7	2D1 (also Servo1)	3D7 /A6
Pin9		2D2 (also Servo2)	
Pin10		2D3 (also Servo3)	
Pin11		2D4 (also Servo4) /A8	
Pin12		2D5 (also Servo5) /A9	
Pin13		2D6 (also Servo6) /A10	
Pin14		2D7 (also Servo7) /A11	
Pin15		Do not connect	
Pin16		Do not connect	

Installation

Put the CD supplied in your PC's drive, then plug in the Mirrorbow USB IO board using a standard USB cable (not supplied). Windows will recognise the new device and prompt for a driver. Select the Mirrorbow CD using browse, and then hit next. Windows will then install the required driver.

Simple Commands and Testing

After installation above, go to My Computer and View System Information (top left in winxp, or right click MyComputer in win2k). Select the Hardware tab, and then push the button marked "Device Manager". This will show a list of the devices on your system. Scroll down to "Ports (COM & LPT)" and click on the + to open the folder. You should now see the USB IO Port. Note the COM port allocated to the Mirrorbow USB IO board as you will need this later.

Command Set:

Command	Format	Returns	Description
DIR	DIR1 00	A as an acknowledge	Sets Port Direction: The port number (1,2 or 3) comes directly after the command, followed by a <space> and the hexadecimal value to write. A bit set to 0 is an output, to 1 and input. DIR3 00 sets all port 3 to output DIR3 0F sets the top of port 3 to output and the bottom 4 bits to input
OUT	OUT1 AA	A as an acknowledge	Outputs an ASCII hexadecimal value to a port: OUT2 55 puts the hexadecimal number 55 onto port 2
IN	IN2	Byte as two characters	Inputs the value of the port: IN1 The board responds with an ASCII hex value e.g. 0F IN3 INA returns either 2 ADC readings or 12 depending on the command below. E.g. when 2 AD inputs are active, 11112222 is returned, where 1111 is the ADC value on pin1 and 2222 the ADC from pin2. 0000 is 0V and 03FF is 5V.
AD	AD+ or AD- or AD*	A as an acknowledge	AD+ turns on the two ADC1 and ADC2 analogue to digital converter inputs, AD- turns off the ADC function. AD* turns on all 12 ADC inputs

Formatting commands: Note the commands always start with the command and port number without spaces, e.g. (OUT1,DIR2,IN1). When writing to the board with OUT or DIR commands there must be ONE space between the port number and the hexadecimal value to be written e.g. OUT1 AA
At any time an invalid input at a position causes the USB IO board to start looking for a fresh command, so you can use a carriage return or two spaces for example, to ensure you are at the start of a command. It is good practice when writing software to send a carriage return or two spaces to the board before sending a command, this ensures that the board is always ready to receive the command and cannot get out of synchronisation.

Simple test using Windows HyperTerminal: For this simple test open windows Hyperterminal (Windows, applications, communications, HyperTerminal). Choose the COM port of the Mirrorbow USB IO Board, and select 115200,N,8,1. Then under ASCII options click "send LF" and "Echo". You are now ready for testing.
(A full explanation of the use of HyperTerminal is out of scope for this guide).

PORT1 is equipped with 4 LEDs for simple testing.

Now type:

DIR1 00

If you are communicating with the board you should see an A appear after the command thus:

DIR1 00A

This is an *acknowledge*. DIR1 sets the direction of PORT1 to all outputs, 0 is output, and 1 input. You can also use DIR2 and DIR3 for the other ports.

Now you want to output a value. *Use the command:*

OUT1 0A

You will get an *acknowledge* so will see **OUT1 0AA**.

0A is the hexadecimal value which is then sent to the port, and should light alternate LEDs. Try OUT1 05, to see the opposite pattern.

To input data, use **IN1**. This will immediately give a response of the ASCII hexadecimal value on the port (no ack is given for input as the response is enough). Before you do this, set **DIR1 FF** to set all inputs on that port. You should get **IN100** for example. Where 00 is the value read. Try connecting pin1 on this port to 5V and type **IN101**, where 01 is the value read.

And for the ADCs:

AD+ turns the ADCs on for pins 1 and 2 of Port3. You will see **AD+A** with the acknowledge.

IN3 will now return the ADC values as well as the digital value of the port.

You will see **IN32003FF0000** in this case the 6th bit (bit 5) is set ADC0 is at 5V (3FF) and ADC0 at 0V.

PWM Operation:

Prescale "T"	valid values are: 04 = prescale of 1 05 = prescale of 4 07 = prescale of 16
Period "P"	00 to FF in hex
Mark/Space Ratio "M"	00 to FF in hex (must always be less than "P")

For a 50/50 mark/space ratio P should be twice M + 1, see formula below. For 25% mark, P is 4xM +1

The output frequency is calculated with the following formula:

$F_{out} = 12000000 / ((P+1) * \text{pre-scale})$	e.g. for P=FF and prescale of 16 $F_{out} = 12000000 / ((256) * 16) = 2.93\text{kHz}$
---	--

Inputting the values on the commands line:

To set the Period to FF hex (256) enter PFF, you will see PFFA with the Ack

To set the Mark to 50%, enter M80, you will see M80A with the Ack

To set the prescale to 1, enter T04, you will see T04A with the Ack (note valid values are only T04, T05 or T07, other values will cause unpredictable results.

Servo Mode:

This feature turns Port2 into 8 servo outputs, with programmable pulse widths from under 1mS to greater than 2mS, designed for driving standard servos. Before using the servo port, ensure that DIR2 00 command is sent, to set the port to outputs and S+ is issued to turn on servo mode. Default duration for each servo is 00 which is off, logic 0 state (command S1 00 can be used to turn say servo 1 off after it was previously active). Example values are S0 C0 which places a 2mS pulse on servo output 0 (Port2 bit0), S7 05 places a 1mS pulse on servo output 7 (Port2 bit7). Each increment in the hex value represents an increase of 5.3uS. Total cycle time seen by each servo will range from around 16mS to 24mS (or 26mS if you set all the servo output beyond 1mS (FF)).

S	S+ or S- or S0 C0	A as an acknowledge	S+ turns Port2 into Servo Mode (see section under Servo Mode), while S- turns Port2 back to IO mode. S7 05 places a 1mS pulse on servo output 7 (Port2 bit7)
---	-------------------------	------------------------	--

Frequency counting:

F	F+ or F-	A as an acknowledge	F+ turns the frequency counter on F- turns the frequency counter off
FD	RDab	A as an acknowledge	RDab sets the input divider (prescaler). Permissible ab values are 00 - No divide 01 - Divide by 2 02 - Divide by 4 03 - Divide by 8
FP	FPab	A as an acknowledge	FPab sets the measurement period. Permissible values are 04 to FF in hex. The measurement period is 25mS times the ab value. So for a measurement period of 1 second set 40 decimal which is a hex value of 28 (FP28)
FI	FI	Fabcd	A number is returned that represents the count during the period. This is a hexadecimal value e.g. F0100 is 256 decimal Note an F prefixes the reading, this helps the host software identify the reading and differentiate it from other messages returned

Example of setting up the frequency counter to read in Hz.

FP28 – sets the measurement period to hex 28 which is decimal 40. 40*25mS is 1 second. So the interval is 1 second.

FD00 – sets the divider to 0 (no prescale)

So we are now set up to count the frequency of the input measuring over a period of 1 second. If the input is say 2kHz, then entering **FI** will return **F07D0** which is 2000 in hexadecimal. The max frequency with this setting is 64,535Hz.

Setting **FD01** for example will divide the input by 2, so for 2kHz input will now read 1000 decimal. But of course the maximum input frequency is now 2x64,535Hz.

Setting **FD00** (no divide) and the **FP14** (20 in hex), changes the sample period to half a second, so again having the effect of doubling the maximum frequency, but halving the reading.

It is best when increasing the frequency coverage to use the divider first.

It can be seen that by decreasing the measurement period with FP along with the divider increases the frequency coverage, while reducing resolution.

Note if the frequency causes the counter to wrap during the specified measurement period the reading will be in error.

Additionally if you have slow inputs you can increase the measurement period up to 255x25mS (6.375seconds), thus gaining more resolution.

Remember that only setting **FP28** and **FD00** will return the frequency in Hz, if you change these values you will need to make an allowance.

Remember that this is a logic input as requires a normal clean 5V logic waveform. External signal conditioning will be required if analogue signals are to be connected.

Port1 Change Polling mode:

L	L+ or L-	A as an acknowledge	L+ turns on Port1 polling mode L- turns off Port1 polling mode
---	-------------	------------------------	---

-.

When polling is activated Port1 is automatically polled every 25mS, and if there is a change a message **Pab** is returned. P stands for Poll and is included to allow this message to be passed correctly in the host software, and differentiate the message from others, and ab is the new value.

By reacting to these messages the host software can respond to changes without having to poll from the host. Note any change in port pins activates this, so if you have programmed any of the pins as outputs and change their state you will automatically receive a Pab message if enabled.

Example: If port 1 changes to a value of say FB in hex, the USB 25IO will send the message **PFB**

Port2 Change Interrupt:

C	C+ or C-	A as an acknowledge	C+ turns on Port2 low nibble interrupts C- turns off Port2 low nibble interrupts
---	-------------	------------------------	---

This mode causes an interrupt on the board each and every time one of the lower 4 port2 bits change. Unlike polling above, this will guarantee that changes that occur within a 25mS period are seen.

Warning: Each change on the lower 4 bits of port2 causes an interrupt on the USB25IO board, it is possible to overload the board and cause a loss in USB communication if these inputs change at an uncontrolled rapid rate.

Only changes to port2 pins configured as inputs cause an interrupt, so using **DIR2 01** for example, will restrict the interrupt to the first pin.

The message returned is in the form of **lab** where 'a' is the 4 bit value read at the time of the interrupt, an 'b' is the 4 bit value at the time the message is sent (there will be a delay in queuing up the message). If the two are not the same there is some evidence that interrupts are being generated faster than the USB25IO can return change messages.

Data acquisition mode:

The USB25IO may be placed in a data acquisition mode. The board will then automatically send to the host data sampled at up to 16kHz. This data can be digital, from Port1 and Port2 or analogue data in 1,2 or 12 channels.

Note that when the board is in data acquisition other functions are disabled, this is to gain the best acquisition rate for data. You should not send any commands to the board except Q0. Unpredictable behaviour may be experience if other commands are sent.

Q	Q0	A as an acknowledge for all command	Q0 turns off the acquisition mode (note any pre-acquisition settings are not guaranteed to be re-instated)
	Q1		Q1 is digital only acquisition mode – each sample consists of the value of Port1 and Port2
	Q2		Q2 is 1 channel analogue to digital mode – each sample is a value representing the signal on A0
	Q3		Q3 is 2 channel analogue to digital mode – alternate A0 and A1 samples
	Q4		Q4 is 12 channel analogue to digital mode – each of A0 to A11 is transmitted
	QA		QA sets a master sample rate of 16kHz and is the default
	QB		QB sets a master sample rate of 8kHz
	QC		QC sets a master sample rate of 4kHz
	QD		QD sets a master sample rate of 2kHz
	QE		QE sets a master sample rate of 1kHz
	QF	QF sets a master sample rate of 500Hz	

You should always set the master sample rate before turning on the acquisition mode. If you wish to change acquisition mode you must turn acquisition mode off with Q0 then select your new mode.

Example1: To set up digital only acquisition at 16Khz send the commands

QA
Q1

Example2: To set up 16khz sampling of A0 analogue to digital input

QA
Q2

Example3: To set up 8kHz sampling of A0 analogue to digital input

QB
Q2

Example4: To set up 8kHz sampling of A0 and A1 analogue to digital inputs. Note we want to sample both channels at 8kHz, and there are two so we need to set the master sample rate to 16kHz

QA

Q3

Example5: To set up 12 analogue to digital channels. Lets set the master sample rate to 16kHz, so each channel will be sampled at 1/12 of the, i.e. 1.33kHz.

QA

Q4

The data is sent to the host PC in the form of 54 byte packets. The header of the packet is 6 bytes, leaving the remaining 48 for sample data. A sample is 2 bytes, and in digital only mode these bytes are Port1 and Port2 values, in analogue to digital they are the high and low bytes of the sample.

When in analogue to digital mode, the header also contains the value of Port1 and Port2. This allows the state of these ports to be watched. These are updated once for each packet and so have an effective sample rate of approx 1/24th of the master rate. However, unlike the normal samples which are accurately timed samples, the header values may have considerable jitter.

The packet format:

'Q'	0x00	0xFF	0xFF	Port1	Port2	Two byte samples x 24
-----	------	------	------	-------	-------	-----------------------

Software examples are provided that demonstrate reading data in both digital and analogue modes.

The first example called "acquisition" acquires half a second of data and writes it to a comma delimited file, which could for example be imported into a spreadsheet.

The second example called "continuous acquisition" continually outputs data from the analogue to digital inputs. For continuous acquisition it is important that the host PC program can process the data fast enough, otherwise the PC's buffer will eventually overflow. You can check for this by, when reading data periodically, checking that you regularly read nothing...reading nothing means you're keeping up with the data. If you always read something then data is coming in faster than you are processing it! If your program is not keeping up, reduce the master sample rate and/or dump part of the data as soon as you read it (only process part of the data).

Host Software

An example of host software is provided on the accompanying disk. It is currently setup to use COM5 so must be changed to reflect the COM port the Mirrowbow USB 24IO appears in your system. The software is documented, please refer to the code for more information.

Specification

General

- 26 IOs – 24 general IOs + PWM output and Frequency measure input
- Power – USB Powered
- Maximum loading on any output 10mA subject to not exceeding 100mA total on all output pins
- All IOs logic level except for inputs configured as Analogue to Digital which have a range of 0 to 5V
- Maximum read speed (from host PC) approx 1mS (Windows on the host usually defaults to 500uS for a command write and 500uS for a response, thus a tight software loop on the host can switch the state of a port every 1mS)

PWM

- Output frequency range from 2.9KHz to 1MHz with programmable mark/space

Frequency Counter

- Input frequency range 0.2Hz to 5.25MHz logic level

Port1 change poll mode

- Any change returns a message (including changes to output state on the port)
- Polled every 25mS

Port2 change interrupt mode

- Any change on lower 4 bits of port2, restricted to changes on pins defined as inputs
- Will catch fast changes, however each change interrupts the processor on the USB 25IO so subjecting this input to continued rapid changes can result in loss of communication with the board...care must be taken with this mode
- Recommended max change rate – 40 times/sec
- Use this mode if you have short duration pulses which could be missed in poll mode above

Analogue to Digital Converter mode

- Selectable as
 - No ADC inputs (all logic IOs)
 - 2 ADC inputs
 - 12 ADC inputs
- Measurement in the range of 0V to 5V
- Resolution 10 bit , returning in hex 0000 to 03FF

Servo Mode

- Port2 can be configured as up to 8 servo outputs
- Designed to drive the signal line of standard servos
- Each output can be either off or programmed to output a pulse of width between less than 1mS to greater than 2mS. Each servo output will continue to transmit the programmed pulse chain continually.

Data Acquisition

- Master sample frequencies of 16kHz, 8kHz, 4kHz, 2kHz, 1kHz and 500Hz
- Port1 and Port2 digital acquisition up to 16Khz
- Analogue to Digital acquisition up to 16Khz
- Analogue to digital data channel modes of 1 channel, 2 channels, 12 channels with Port1 and Port2 state

