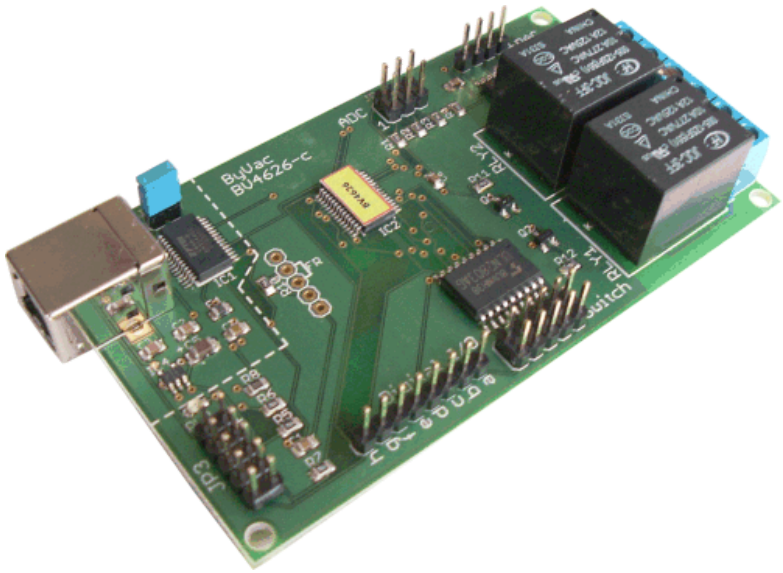
Multi-I/O BV4626



BV4626

General Purpose I/O

Product specification Mar 2010 V0.a

Contents

Rev	Change
Mar 2010	Preliminary
Sept. 2010	Note about digital I/O
Sept. 2010	Firmware update V9 & 10 see section 13
April 2011	Added relay specification
Feb 2014	I2C address corrected
Aug 2016	Default is now 115200 Baud, autobaud removed.

1. Introduction

This is a general purpose input / output board that has **three** interface options.

A USB option that presents itself as a COM port. A serial interface using escape commands and an I2C interface. All three interfaces can control the various I/O devices.

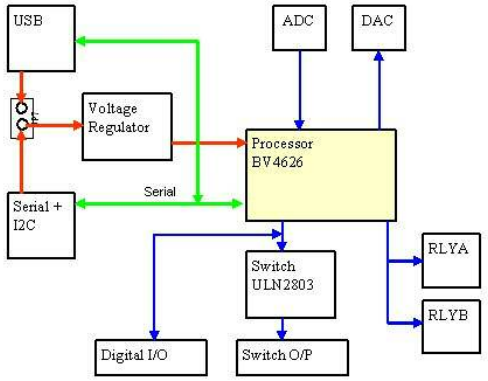
2. Features

- Twin 10/6A relays @ 250V AC
- Two channel Digital to Analogue with 64 steps
- Four channel 10 bit Analogue to digital
- On board adjustable precise voltage reference
- Eight channel digital input or
- Eight channel digital output
- Eight channel high power outputs
- USB interface
- Serial Interface
- I2C Interface
- Current: 16mA @ 5V (relays off), relays consume about 60mA each.
- Size: 94 x 53mm

The relays fitted will have the following minimum specification:

- Normally Open 10A @ 240V AC
- Normally Closed 6A @ 240V AC
- 10A @ 24V DC

3. Physical Specification



Block Diagram

The device has fairly comprehensive input / output options as can be seen by the block diagram.

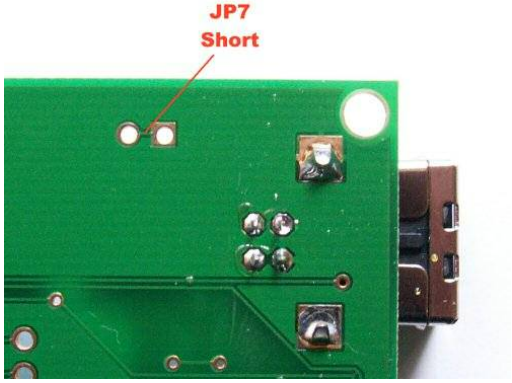
Power is derived from either the USB **or** the serial/I2C interface (not both). It is then passed to a voltage regulator. This has the dual purpose firstly of protecting the USB from inadvertent short circuits and secondly when not being powered from the USB, up to 10V can be applied without damage to the device, provided JP7 is NOT in place.

The serial interface and USB share the same I/O to the processor and so one or the other can be used. The following text describes each section of the device.

3.1. JP7

Note, where the board is supplied without the USB interface this section about JP7 is not relevant.

It may not be clear form the above exactly what this does. JP7, when closed supplies power to the device from the USB interface. Most of the time this can be in place, and to this end, by default it is shorted out by a PCB track.



As can be seen there is a thin track in place on the underside of the board that shorts out JP7, this can be cut if required. The following rules will help decide if this needs to be in place or not.

- Power supplied by USB connector: JP7 should be in place as above.
- Power supplied by serial connector which is 5V. JP7 can be in place as above or it can be cut – optional. Cutting will stop power to the USB chip and thus save a little power.
- Power supplied by serial interface which is greater than 5V. JP7 MUST be cut otherwise damage to the USB chip WILL occur.

3.2. Control Interface

The device comes optionally with a USB interface for connecting directly to a PC or Linux computer or with a serial interface for connecting to a microcontroller. Either a serial or I2C option can be chosen at start up.

3.3. Serial Interface & USB

The default Baud rate is now 115200

Where the USB circuitry is fitted, this uses the FDTI USB to COM port IC. The drivers for this are freely available at http://www.ftdichip.com/ The driver required is the 'VCP' driver which stands for virtual comm. port.

When the driver is installed (it is already installed for Linux users) the board will present itself as a COM port. This is then directly connected to the serial interface and so the USB and serial share the same interface, a description of which follows.

Serial Connector

The serial connector is a 2x5 pin at the side of the device designated JP3. Pin 1 is marked with 1. The odd pins are down one side of the connector and the even pins down the other as shown.

Pin	Function	Function	Pin
2	nDD	RX	1
4	Reset	TX	3
6	SCL	+V	5
8		SDA	7
10	nRTS	Gnd	9

This bVT100 serial connector will mate directly with the BV101-3.

The processor will operate from 3V3 and there will be some I/O functionality but the relays and voltage reference require 5V and so the device should be supplied with a 5V supply.

RX This is the serial receive and expects signals 0 to +V. The idle state is high. The input is a standard byte frame of 1 start bit 8 data bits and 1 or 2 stop bits. In other words a standard asynchronous serial signal.

NOTE the interface will NOT directly accept voltages from a standard COM port that output +ve and negative voltages. There must be some kind of voltage translation, see the BV103 at www.byvac.com which is an ideal convertor.

TX This is the output from the device.

+V Is the main power supply for the device which should be capable of providing sufficient current. This device requires a voltage of between +5V and +10V, there is an on board regulator that will supply the correct voltage to the internal circuits.

IMPORTANT: If the USB interface is not being used and it is fitted then remove JP7, supplying a voltage of greater then +5V without this jumper removed will damage the USB chip.

nRTS (output) Hardware handshaking is possible with this device and in some circumstances can be essential. The device will buffer up to 30 bytes and so provided that it can be guaranteed that the buffer will not become full then this line can be ignored. See also ACK.

This line is held low (0V) by the device when it is ready to accept data, it will put this line high when it is busy, the transmitter should monitor this and only send characters when the line is low. This signal is normally connected to the CTS line of the transmitting device.

nDD This is called the Device Disable and if left disconnected the line is pulled high by an internal pull up resistor, thus enabling the device. If the line is taken low then the device becomes disconnected from the serial interface. This is so that more than one device can be connected to the same serial bus.

If the line is pulled low, the devices UART will be disabled and the RTS line will also be disconnected, effectively removing the device from the circuitry. This will take a finite time to accomplish (approximately 10uS) and so this must be taken into account when switching between devices.

SDA & SCL

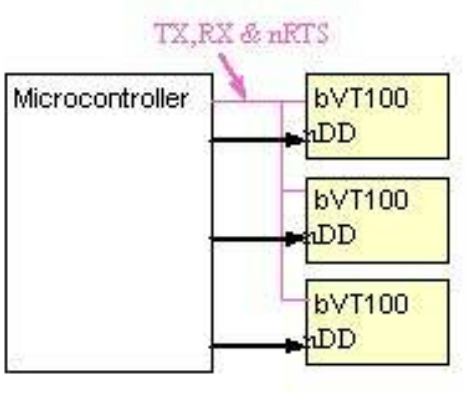
(See the section on the I2C interface)

Reset

Pulling this line low for 10ms and then releasing will reset the device. There is no need to reset the device on start up as this is done automatically. This line is provided for resetting during use, for whatever reason. If it is not intended to do this then it should be left unconnected.

3.4. Multiple Devices

As previously mentioned it is possible to control more than one device on the same bus by using the nDD signal line. This is similar to Chip Select except it does the opposite, when asserted (taken low) it deselects the device.



In this example 3 GPIO lines are used from a microcontroller, at any time two of them will be low and one high to select one device.

4. Handshake and ACK

The ideal interface would implement the hardware handshaking mechanism using RTS/CTS. This is where the devices RTS is connected to the host CTS, when the device is busy, it raises the RTS line and the host temporarily stops transmitting.

Where it is not possible to do this and also where software can be simplified, the ACK mechanism is available. This is switched off by default and so must be enabled using the appropriate command.

Because the device takes a finite time to carry out a command, say turning on a relay, it is useful to have an indication of when that has happened and more accurately when the device is ready to accept another command. This is what the ACK mechanism is for. When enabled and when the device has completed a command and is ready to accept another command it will send an ACK character. The actual character (value from 1 to 255) can be specified by the user.

Not all commands will send an ACK where this is so it is indicated on the command table.

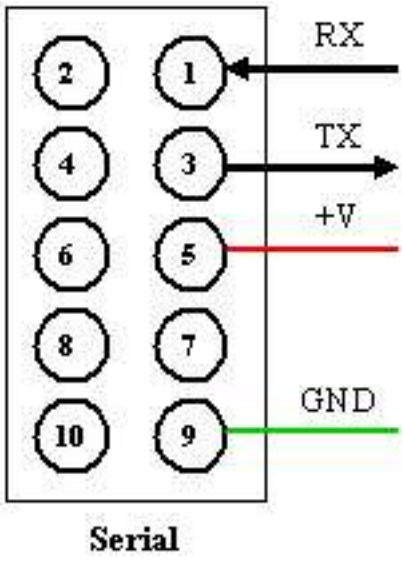
5. Control Commands

There are TWO distinct methods of controlling the device, one is by serial commands and the other is by I2C. Only one can be active at any one time, see the section on I2C for how to activate this method of command.

The device when used serially is controlled by issuing control commands that always begin with and escape byte (0x1b). The escape is followed by various sequences to make up individual commands.

The command is accepted as soon as the last byte of the command is entered, so for example when turning on relay 'A', the sequence is esc[1A. As soon as the 'A' is entered the relay will energies.

6. Serial



Serial

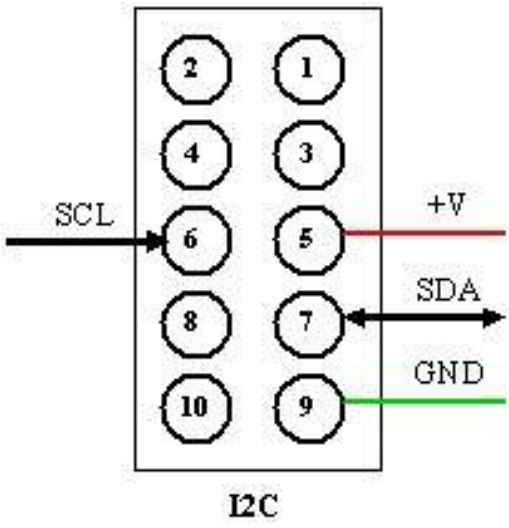
The serial interface uses 1 start bit, 8 data bits and 1 or 2 stop bits, no parity. By default the Baud rates

2400, 4800, 9600, 14400, 19200, 38400, 56700 and **115200**

No other rates are acceptable and so the host must be set to one of these. It is possible to fix one of these rates (see the command table).

7. I2C

(Default address 0x62, 8 bit) (0x31, 7 bit)



The device can operate from serial OR I2C. This is determined by the SDA line which is held low by a high value resistor. The mode is determined at reset or switch on.

With no I2C bus connected the SDA line is low and thus the device detects this and starts up in serial mode. With an I2C bus connected the pull up resistors on the master or bus hold this line high and the device goes into I2C mode. The mode remains until reset.

It should be obvious that this device does not have any pull-up resistors, these are normally provided on the master device or on the bus by two discrete pull up resistors.

The commands are shown in the command table and instead of the escape sequence a single command byte is used. All I2C communication follows the same pattern which is:

7.1.1.Write

1. Send start condition
2. Send device address with write
3. Send command
4. Send stop condition

7.1.2.Read

1. Send start with write
2. Send device address with write
3. Send command
4. Send restart condition with read
5. Read bytes
6. Send stop condition

There is a notation that is used in the examples as follows:

's' means send the start condition and device address with write. i.e. the least significant bit is set to 0 telling the I2C device that the next byte should be written to the device.

'number' A number will simply be sent to the device.

'r' This is the restart command; it sends a stop condition followed by a start condition and the device address with the read bit set (LSB set to 1). The device will now send out bytes that are clocked by the master.

'g-n' will clock 'n' number of bytes out of the device, 'g-3' will get three bytes.

'p' is the stop condition.

As an example, to send a command to a device and receive a single byte:

s 22 r g-1 p

This reads as send start and address, send byte with a value of 22, send a restart condition with device address and read bit set, clock out (get) 1 byte and send stop condition.

8. Relays

There are two relays that can be individually turned on or off. Each relay can switch up to 10A at 277V.

WARNING: This device must NOT be connected to AC Mains voltages without the proper advice from a qualified electrician. Mains voltages are Lethal you have been warned.

9. Digital I/O

The digital input output has 8 channels each individually being configured as either an input or output. When configured as an output an individual channel can be either fully on*, fully off or somewhere between.

*see output

9.1. Input

By default the port of 8 channels 'a' through 'h' are configured as input. This port is also connected to the digital switch and so does have an effect on the input impedance. The port is read as a whole using the appropriate command.

NOTE: The digital switch input impedance is quite low and so this must be taken into account if using these pins for input.

9.2. Setting In or Out

One command provides the control for if a channel is either an input or output. A byte value is provides where 1 is input and 0 is output. The value must be converted to decimal before issuing the command.

Example:

To set channels a,c and f to input and the rest as output:

h	g	f	e	d	c	b	a
0	0	1	0	0	1	0	1

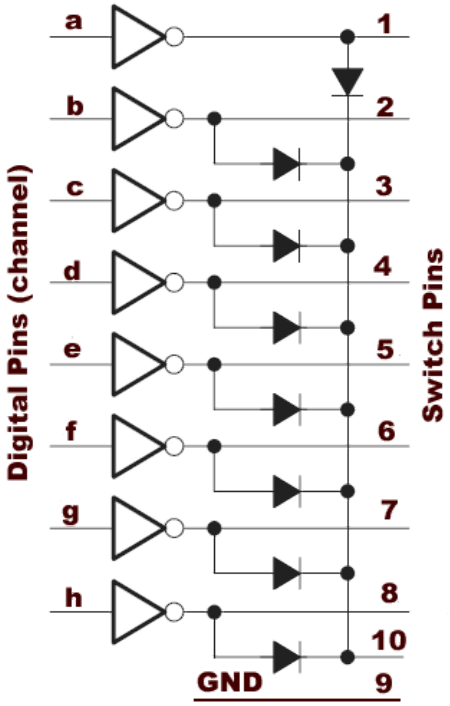
This gives a binary value of:

00100101 which is 0x25 in hex

This can be put into a scientific or PC calculator and will return a result of 37 which is the value that should be used in the command.

9.3. Output

There are two physical output ports, one is directly off of the microcontroller and is marked as 'Digital' The other is via Darlington transistor array and marked as 'Switch'.



Pin	Chan	Chan	Pin
1	a	b	2
3	c	d	4
5	e	f	6
7	g	h	8
9	GND	COM	10

Physical Connector Layout

The connector used for this output is a 2x5 way and the physical layout is show above, the odd pins are down one side and the even down the other, pin 1 can be located on the PCB.

The IC used for the Darlington o/p is a ULN2803A, the data sheet can be found on the ByVac website. The markings on the PCB 'a'-'h' of the Digital connector correspond to the input to the Darlington, these are also connecting to the microcontroller and so can be used as either input or output.

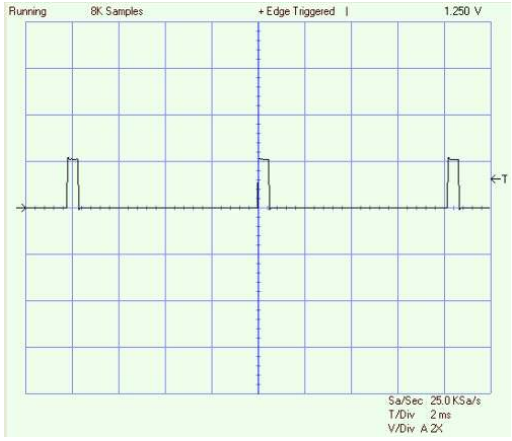
When used as an output the microcontroller will also drive the Darlington array which is capable of up to 500mA per channel. Pin 10 can be connected to the drive voltage if inductive loads are used.

The Darlington's act as a switch and so setting a high at the input will cause the selected Darlington to switch on. A low will switch it off.

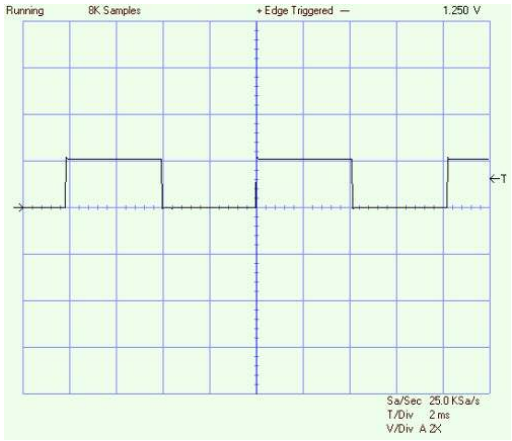
9.4. PWM

The output from the microcontroller is modulated within a fixed duty cycle. A number can be specified when using the output command, the number ranges from 0 to 255. At 0 the channel will be fully off and at 255 the channel will be fully on. A number specified between these two will modulate the signal.

The duty cycle is approximately 8ms and the on time of the channel can be set by using the number in the command.



As an example this first image has the value set to 15, the graticlue squares are 2mS apart and it can be seen that the on time is (15/256)*8ms which is approximately 0.5ms.



This next image has the channel set to 128 which is half of the duty cycle.

All 8 channels can be individually set to a different value.

NOTE: The timings are approximate and may vary if there is considerable activity on the serial port.

(pre V9) Also When the output is fully on there is still a modulating signal present that may effect digital circuits.

10. Analogue to Digital

CH1	CH3	+V
2	4	6
1	3	5
CH0	CH2	GND

Connector, pin 1 is marked on PCB

There is a 4 channel analogue to digital input port that is continually scanned for values. The input for these are on the ADC connector. Each channel has a 100k resistor connected to +V and so will read 1023 when not connected to anything.

The acquisition and conversion time for each channel is approximately 8ms and each channel takes it in turn to sample the value. The average time therefore is 32mS, however the last value is always available as this is stored in internal RAM.

To get the value of a particular channel the command is specified using the channel number 0 to 3. The output is in ASCII coded format and is in the range 0 to 1023.

10.1. Reference Voltage

There is an accurate reference voltage on board that is used for the ADC. The Voltage can be set to one of three values: 1.024V, 2.048V and 4.096V. The latter assumes at least a 5V power supply.

The voltage reference by default is 4.096 but this can be altered using the appropriate command.

11. Digital to Analogue

GND	Y	X	+V
4	3	2	1

Connector pin 1 is marked on PCB

There are two digital to analogue (DAC) channels that provide a voltage output from 0 to +5V in 64 steps (0-63). This is implemented by two 5k digital potentiometers; one side is connected to +V, the other to ground with the wiper being the output.

Each channel can be individually controlled by the appropriate command.

12. Factory Reset

Using the appropriate commands defaults can be set that in certain circumstances make the device difficult to communicate with. For example if an unknown Baud rate has been inadvertently set.

In this case there is a hardware option for setting all of the defaults back to a known condition. To do this use the following procedure:

1. Remove the power
2. Short out the two pads on the PCB marked FR, this will be somewhere on a row of 5 holes.
3. Apply power
4. Remove power and remove the short.

The device will now be restored to its factory defaults.

13. Firmware Version

Changes to firmware: The firmware version can be checked using the command esc[?31f a whole number will be returned.

13.1. Pre- Version 9

The input buffer is 70 bytes and there is a ripple on the PWM output when full on.

13.2. Version 9 & 10

The input buffer has been reduced to 30 bytes and there is no ripple when selecting an on value, on the digital ports of 255.

Version 10 will also allow user firmware upgrades via the USB.

13.3. Implemented Escape Codes & I2C commands

In the table columns indicate:

S is the VT100 standard, Y is standard, P is partially standard, N is non-standard

Code is the escape sequence to invoke the command

DEF column shows the default values from the factory, and F in this column will indicate that this value will be stored when writing the system to block 0

ACK column indicates that this command is capable of sending an ACK character when it has finished.

I2C column is the I2C command when in this mode ** Note the command values are decimal **

Grayed out cells are not implemented on this device. Unless otherwise stated all numbers are in decimal.

S	Code	DEF	ACK	I2C[3][4]	Name	Description
	Relay					
N	esc[<num>A		Y	10 0xa	Relay A	Turns on or off relay A, when <num> is 0 the relay is turned off, when it is 1 the relay is turned on
N	esc[<num>B		Y	11 0xb	Relay B	Turns on or off relay B, when <num> is 0 the relay is turned off, when it is 1 the relay is turned on
					I2C Notes	I2C requires two bytes to be sent, the command follows by either 1 or 0, for example: s 10 1 p (using the BV4221 in hex mode: s a 1 p)
	Digital Input / Output					
N	esc[<num>s		Y	21 0x15	Set input/output	Sets the digital port to either input or output. A single byte is provided to determine all of the channels, setting a bit to 1 is input and 0 is output. The least significant byte is channel 1 (a). Thus to set channels 1 to 4 as input and 5 to 8 as output the hex byte would be: 0x0f, this must be converted to decimal and supplied to the command: Example: To set channels 1-4 (a,b,c,d) as input and the other channels as output, the value in binary is 00001111, this is 0xof in hex and 15 in decimal, thus: esc[15s
					I2C Notes	The byte following the command will set the i/o as in the above example to set the lower 4 bits to input then:

						a 21 15 p (see text for use on this notation)
N	esc[r		Y	22 0x16	Return port value	Gets value from digital port as a single byte, individual port bits that are set as outputs will return the values that have been set to.
					I2C Notes	The byte is fetched after the command is sent: s 22 r g-1 p (see text for use on this notation)
N	esc[<num>a …. esc[<num>h		Y	23 0x17	Sets o/p channel A-H	Sets an output value on channel A, the value should be between 0 and 255. 0 is off and 255 is fully on (high). A value between these will set a proportionate period within the duty cycle. See text for further information. **The channel must be set to output first for this to work**.
					I2C Notes	Instead of a letter 'a' – 'h' a number is used after the value to specify the channel number. Channels are numbered from 0 to 7 representing 'a'=0 … 'h'=7. As an example to set channel 'b' to 128: s 23 128 1 p (see text for use on this notation)
	Analogue to Digital					
N	esc[<num>D		Y	31 0x1f	Get ADC channel	Gets the value of the specified channel (0-3) in decimal. This will be a number 0 to 1023
					I2C Notes	The channel 0-3 is specified following the command and two bytes are returned, high byte first. The following example will return the value of channel 0. s 31 0 r g-2 p (see text for use on this notation)
N	esc[<num>V		Y	32 0x20	Set ref. Voltage	The reference voltage for the ADC can be set to one of three voltages, specify a number as follows: 0 = 1.024V 1 = 2.048V 3 = 4.096V (default) Values specified outside this range will revert to the default value.
					I2C Notes	The reference follows the command, to set a reference voltage of 2.048 then: s 32 1 p (see text for use on this notation)
	Digital to Analogue					
N	esc[<num>X		Y	41	Output to DAC 1	Outputs the value given by <num> to the digital to analogue convertor (0-63)

				0x29		
N	esc[<num>Y		Y	42 0x2a	Output to DAC 2	Outputs the value given by <num> to the digital to analogue convertor (0-63)
					I2C Notes	The value to place on the DAC is given in the second byte, for example to set the Y DAC value to 20: s 42 20 p (see text for use on this notation)
	System Settings					
N	esc[<num>E	Off F		n/a	ACK	Sets and activates the ACK character as an alternative to hardware handshake. Some commands will send a character to the receiving device, this can be used to determine when the command has finished. The host can wait for this character before sending the next command. To set up the ACK to send 'X for example the following is used: esc[88E 88 Is the ASCII code for 'X'. To turn off the ACK system use esc[0E (esc[zeroE).
N	esc[?27D		Y	82 0x52	Writes defaults to EEPROM	Any variable indicated by an 'F' in the 'DEF' column of this table will be written to the EEPROM. This means that when the device is reset or switched on the new values will take effect (whatever the user has set them to) rather then the factory defaults.
					I2C Notes	Simply send the command: s 82 p
N	esc[<num>I		Y	87 0x57	Set I2C Address	This will set the I2C address and must be followed by EEPROM write. The new address will not take effect until reset. The default address is 0x62 (98). As an example to set the address to 0x42 (66) : <esc>[66I<esc>[?27D Both of the commands must be specified although the write to EEPROM command does not need to directly follow this command. The new address will take effect when the device has been restarted. IMPORTANT An even number MUST be specified to give a valid 8 bit address, the device will not work if an odd address is specified, the device does not check the address you enter.
N	esc[?29a	On F	Y	83 0x53		Sets the baud rate to be automatic so that when the device resets or is first switched on, it will wait for the first CR to determine the rate. The rate is selected from the following: 2400; 4800; 9600; 14400; 19200; 38400; 56700; 115200 NOTE: the right EEPROM command must be used after this command in order for it to take effect on the next reset.

						Example: to set the device back to auto Baud: esc[?29a esc?[27D Removed release 13 onwards
					I2C Notes	Simply send the command: s 83 p
N	esc[?29b	F	Y	84 0x54		The Baud rate will be set (fixed) to the current Baud rate so that when the device is next switched on or reset the new Baud rate will be in effect, by-passing the Auto Baud mechanism. NOTE: the write EEPROM command must be used after this command in order for it to take effect on the next reset. Example: esc[?29b esc?[27D This will fix the Baud rate to the current rate.
					I2C Notes	Simply send the command: s 84 p
	Esc[<num>k			43 0x2b		Permanently Changes Baud rate where <num> is: (Release 13) 1 115200 <default> 2 56700 3 38400 4 19200 5 14400 6 9600 7 4800 8 2400 Command will take effect after next reset.
N	esc[?31d		Y	85 0x55	Device ID	Returns device ID
					I2C Notes	The device ID is returned as 2 bytes, this is a 16bit number with the high byte being sent first. s 85 r g-2 p (see text for use on this notation)
N	esc[?31y	On	Y	n/a	Com Flag (0n)	When communication is established with the device, i.e. at switch on or when the auto Baud

		F				mechanism has selected the correct Baud rate, ASCII code 42 '*' is sent to indicate this has happened. This commands switches this on, it is on by default
N	esc[?31n		Y	n/a	Com Flag (off)	Switches the flag off see esc[?31y
					I2C Notes	The com flag is not applicable to I2C
N	esc[?31f		Y	81 0x51	Firmware version	Returns firmware version, this is a 16 bit number
					I2C Notes	The firmware version is returned as 2 bytes, this is a 16bit number with the high byte being sent first. s 81 r g-2 p (see text for use on this notation)
P	escc			86 0x56	Reset device	Carries out a software reset
					I2C Notes	Simply send the command: s 86 p

Notes

[3] In the examples given for I2C 's' represents a start condition, 'r' a restart condition, 'p' is a stop condition and 'g-n' is 'get' bytes where 'n' is the number of bytes to read see text for more details.

[4] Numbers are decimal.