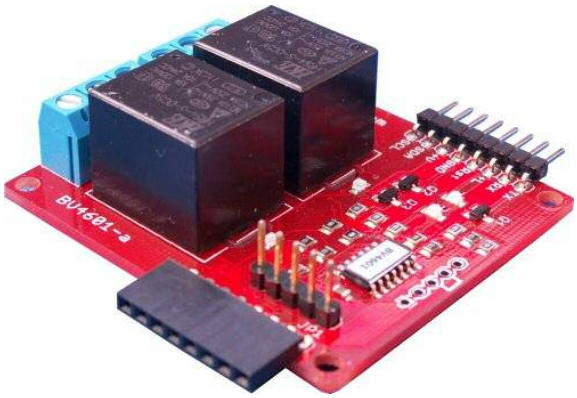
Serial & I2C Twin Relay BV4601



BV4601

Serial & I2C Twin Relay

Product specification Sep. 2013 V0.a

Contents

Rev	Change
Sep 2013	Preliminary
Sep 2013	Warning about stacking +V pin
Dec 2013	Errors in I2C command documentation

1. Introduction

The BV4601 is a twin relay that can be controlled with either a serial or I2C input. The connectors on the side allow it to be stacked without the need of additional wiring.

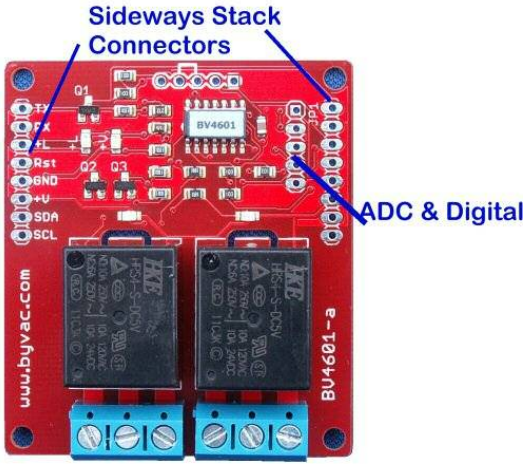
The relays can be timed on or off from 1mS to 48 days. In addition there are three 10 bit ADC inputs and one digital output.

More data and examples with free software can be found at www.pichips.co.uk

2. Features

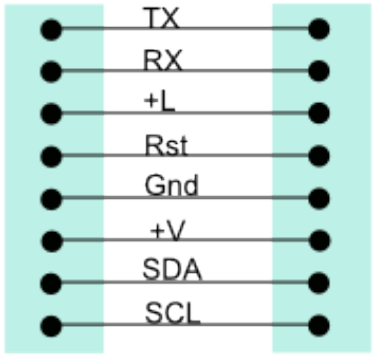
- Wide voltage range 2.5V to 5.5V
- Current relays off 5V, 6mA
- Current each relay 5V, D:\Zed\Dual\BV4601 - dual relay90mA
- Timed on / off
- Three 10 bit ADC channels
- On board temperature indication
- One digital output
- Multi Volt logic
- Addressable many devices can share a single serial bus
- Sideways stackable
- User configurable EEPROM
- 10A relays

3. Electrical interface

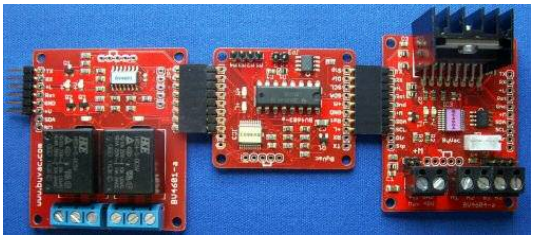


3.1. Sideways Stackable

The connectors on either side of the board are through connectors and are designed to have a plug on the left hand side and a socket on the other. This enables devices to be stacked side by side without the need for any additional wiring.



If more than two relays are required then additional BV4601's can simply be stacked side by side. This is not limited to just this device.

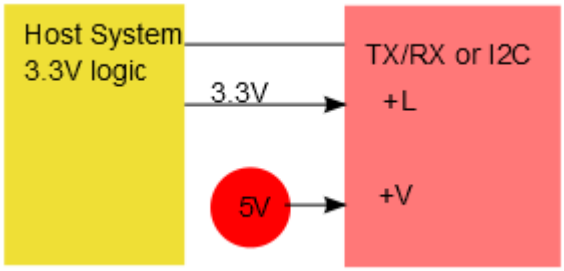


Several differing devices can be stacked in this way. ** See WARNING

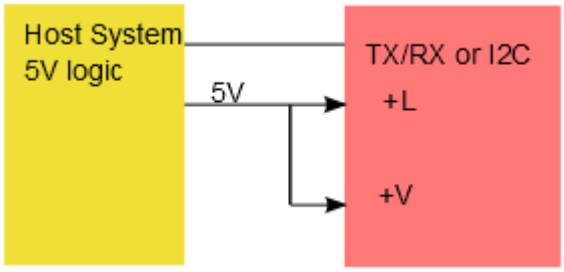
3.2. Power Supply

There are two power inputs to the connector, +L and +V. This allows 3.3V devices to be used yet the relays still powered by the required 5V.

+V should be set to 5V and +L should be set to the logic matching the host system, this will be either 3.3V or 5V.



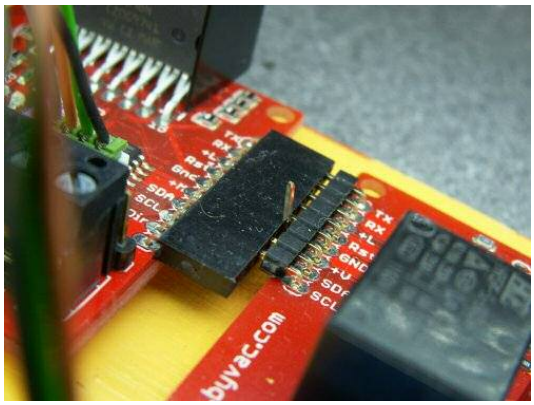
Example when connecting to a system with 3.3V logic supply



Example when connecting to a host system that uses a 5V logic supply.

The logic supply can range from 2.5 to 5.5V

** **WARNING** The maximum voltage for this device on the +V pin is 5V. In the photograph, as the +V pin is shared, some devices have differing power requirements as an example the motor controllers may need 12V but the relay maximum is 5V, so when stacking differing devices be aware of this.



In this example, a relay is stacked to a motor controller. The supply to the motor controller is 12V through the +V pin and so as can be seen the relay is stacked because it shares all of the other pins but the +V pin is bent out of the way so that it can be connected to a separate 5V supply.

4. Relays

The specification is marked on top of the relays, this may vary from device to device but it will always be at least 10A. The current varies with the type of voltage but is typically:

250V AC, NO 10A
250V AC, NC 6A
120V AC 10A
24V DC 10A

The relays require 5V to operate, this must be provided via the +V pin.

5. Serial Interface

The serial interface is a standard 1 start bit 8 data bits and 1 stop bit and is initially set to 9600 Baud. This is user changeable from 2400 to 115200 in 8 steps.

The TX has an open collector output and so many devices can be connected to the same serial bus. Each device has its own address ranging from 97 to 122 giving a maximum of 26 devices per serial bus.

6. I2C Interface

The default i2c address is 0x66(8 bit) 0x33(7 bit - RPi, Arduino)

By default the serial interface is active. That is if no connections are made to the I2C pins. If an I2C host is connected to the I2C pins then this will pull up the SDA line as the I2C specification requires pull up resistors. If this is the case and power is applied to the device then the I2C interface will be selected.

7. ADC and Digital

This interface has the following pins:

Pin	Description
1 (top)	ADC 1
2	ADC 2
3	ADC 3
4	Digital out
5	Ground

The ADC is a 10 bit input which means that values from 0 to 1023 can be retrieved. Each channel is continually scanned at approximately 1mS intervals and the 'g' command retrieves that last scan value. This frees the host from having to bother with delays and the like to get an ADC reading.

The frequency of the scan rate determines the acquisition time (the time taken for the circuit to respond to a new value) and this can be set between 1 and 256 mS by adjusting the scan rate value in EEPROM. See later in the text for EEPROM values.

Physically all three channels are connected to 10k pull up resistors and so without anything connected to the pins a value of around 1023 will be received.

A voltage reference can also be set using the 'v' command - see the command for more details.

This device has **four** ADC channels, the forth channel will give an indication of the temperature of the controller IC. This cannot be translated into degrees but the hotter the IC gets the larger the value will be.

There is just one digital output that is set low on reset. This can be set high or low by using the appropriate command.

8. Reset

This pin can be left unconnected or tied to +V, pulling this pin momentarily low will cause a hardware reset.

9. Factory Reset

The device set up including its address is determined by values that are stored in EEPROM. This enables flexibility as the user can adjust the values.

It may be possible to place some values in EEPROM that render communication with the device impossible. If this is the case there is a procedure that will set some of the EEPROM values to a conditions where the device can communicate again. The procedure is:

1) Remove the power

2) Place a jumper wire between the pins as marked on the PCB. This is the top row of 5 holes, pad 1 is the square one and so the jumper goes between pads 4 and 4. This is indicated via an image of a bridge.

3) Apply power to +L, reset will be instantaneous.

4) Remove the link and communication will be restored at 9600 using the default address.

10. Serial Command Set

Default address ('f')

The commands are sent to the serial interface byte by byte, however for convenience the byte values chosen coincide with ASCII characters. This makes debugging on a terminal very easy.

All commands will be referred to by their ASCII value but remember on a microcontroller host system, sending 'a' on a terminal is just the same as sending the value 97.

Where appropriate all of the serial commands listed in the summary table below have an I2C equivalent.

The full command details are listed later on in the text.

Command Set Summary	
a (1)	Turn on / off relay A
b (1)	Turn on / off relay B
r (2)	Read relay timer values
o (6)	Turn off all relays
g (5)	Get ADC values
v (7)	Set ADC voltage reference
d (8)	Digital output
W (0x91)	Write to EEPROM
R (0x90)	Read from EEPROM
I	Invert TX
C (0x95)	Reset
D (0xa1)	Device ID number
V (0xa0)	Firmware Version
H	Hello

Table 1 Command Set summary (I2C)

All of the above commands require a device address to be specified before sending the command and also **every command sequence must be terminated with CR** ("\r") (13) (0xd).

The device will return ACK (6) on all successful commands and NACK (21) on unsuccessful commands.

Any command beginning with an address that does not match the devices address is ignored.

11. EEPROM values

The EEPROM contains important values that control the way the device behaves. All of the values can be changed by the user using the 'W' command.

The EEPROM consists of 255 bytes and in general the first 16 bytes are used by the system, the second 16 byte are used by the device and the rest of the bytes can normally be used by the user.

Adr	Default Value	Description
0	0	System Use
1	102	Device address
2	6	ACK value
3	21	NACK value
4	3	Baud rate
5	1	Error reporting 1 = on
6	13	End of line
7	1	Invert TX 1=invert
14	102	Device address copy
16	0	Relay time scale low
17	0	Relay time scale high
18	3	Voltage reference
19	1	ADC scan rate multiplier
250	102	Device address copy

Table 2 EEPROM use

The user is free to use any locations that are not occupied by the system but for future use it is best to avoid locations below 32.

EEPROM values are only read on start up so when changing values they will not normally take effect until the device is reset.

11.1. Address

These EEPROM locations contains the device address. By convention the address is set to values between the values 97 to 122, no checking is made by the device so setting values outside this range may or may not work.

For security the address is stored in three places and to change the address of the device at least two of the locations need to be set otherwise the device will detect the anomaly at start up and revert to the majority value.

Normally to change the address of a device locations 1 and 14 are both changed. The device will detect this at start up and change the address in location 250 to match.

11.2. ACK character

By default this is 6 but can be changed using the EERPOM Write command. The effect will not be implemented until the device is reset.

11.3. NACK character

By default this is 21 but can be changed using the EERPOM Write command. The effect will not be implemented until the device is reset.

11.4. Baud Rate

The Baud rate has the following values:

0. no valid
1. Baud rate is fixed at 2400
2. Baud rate is fixed at 4800
3. Baud rate is fixed at 9600 (default*)
4. Baud rate is fixed at 14400
5. Baud rate is fixed at 19200
6. Baud rate is fixed at 38400
7. Baud rate is fixed at 57600
8. Baud rate is fixed at 115200

11.5. Turn off Error reporting

By default error reporting is enabled and this will be reported and an output prefixed by Error, for example '**Error 2**'. This may get in the way of the program trying to control the device and so it can be disabled with this command. The effect will not be implemented until the device is reset. This does not apply to I2C if available.

11.6. CR Character

By default this is 13 which is the standard ASCII CR and the whole protocol relies on this being at the end of every command. It may be that this is unsuitable in some systems and so this can be changed.

11.7. Relay time scale

By default the timing value used by the device is milliseconds. This value can be scaled using a setting in the EEPROM.

The scale value is a 16 bit word, thus if it is set to 1000 then the timing value will be 1 second, if set to 60000 then the timing value will be 1 minute. As the maximum value a 16 bit word can hold is 65000 then the scale maximum is 65 seconds (approximately). So when a timer value of 65000 is given this is gives a delay of about 48 days.

The timings are approximate and will vary. If accurate timings are needed then tests should be carried out.

Some EEPROM values for various scales

Scale	**(low)**	**(high)**
10	10	0
100	100	0
1000	232	3
60000	96	234

The formula is: divide the scale value by 256, that is the high byte, the remainder is the low byte.

11.8. Voltage Reference

This is the default voltage default reference code that will be used on reset. See the voltage reference for suitable values.

11.9. ADC Scan Rate Multiplier

The ADC channels are scanned every 1mS in turn. This value multiplies this by the given amount, so if set to 10 then each channel, in turn would be scanned every 10mS.

This is the time that will be given to the ADC circuit for acquiring the voltage and may need to be increased for higher impedance circuits. In practice 1mS is sufficient for most applications.

Setting the value to 0 will give a delay of 256mS.

The channels are scanned in turn so for three channels with this set at 1, the acquisition time will be 1mS but the channel will be updated every 3mS.

12. Commands

All serial commands are proceeded by an address and terminate with CR (0xd). In the examples given below the address is 'f' or 0x66

When a command is accepted by the device it always returns ACK which by default is the value 6. If the device rejects the command then it will return NACK, value 21

Serial	I2C	range	Default Value	EEPROM Location	Description
an,m **bn,m**	1 or 2	n=0-1 m = 0 to 65534			**Turn on / off relay** A relay can be turned on or off in a given time. The command format is: <adr><a or b><0 or 1><time><CR> The a or b refers to the particular relay, 0 or 1 is off or on respectively and the time is the time in mS when this will happen. The time value can range from 0 (immediately) to 65,534mS. **Examples**: Turn on relay 'a' immediately: **fa1,0** Turn off relay 'b' immediately: **fb0,0** Turn on relay 'a' in 20 seconds: **fa1,20000** **I2C** The relay is either command 1 or 2. Command 1 applies to relay 'a' and command 2 applies to relay 'b' **s <1 or 2> <1 or 0><timeH><timeL> p** As an example to turn relay 'b' on after 100 the following would be applied: s 2 1 0 100 p To turn off after 200 s 2 0 0 200 p
rn	4	n=1-2			**Timer value** Gets the current timer value for the relay. It may be useful for the host to know when or just how far into the action the relay has got. Some appropriate action may not be performed until the relay has energised for example. In this case the relay is specified as a number from 1 to 2 The command format is: <address><'r'><relay number 1 to 2><CR> As an example to get the relay timer value for relay 'b' would be: **fr2** **I2C** **s 4 <1 or 2><timeH><timeL> p**

o	6				**All off** This is just a convenient way of turning all of the relays off with one command. Example: **fo** **I2C** **s 6 p**
gn	5				**Get ADC Values** Command format: <adr><g><1,2 ro 3><CR> This will return a value between 0 and 1023. **Example:** Get the value of ADC1 **fg1** **I2C** **s 5 <chan> r g-2 p** Two bytes received Low first
vn	7	n=1-4	3		**Set Voltage reference** There are 4 possibilities for the +ve voltage reference and these are: 1: 1.024V 2: 2.048V 3: 4.096V 4: VDD (+L) The command will accept a value of between 1 and 4 that will set the voltage as above. Obviously if the host voltage (+L) is 3.3V then option 3 will not give the 4.096 as stated. **Example:** To set the reference voltage to the same as +L: **fv4** **I2C** **s 7 <1 to 4> p**
dn	8	n=0-1	Low		**Set Digital Output** This will set pin 4 of the output pins to be either 1 or 0, at reset the pin is 0. To set the pin to 1: **fd1** The voltage on the pin will be the same as the +L voltage. **I2C** **s 8 <1 or 0> p**
Wn,m	0x91	n=0-255 m=0-255			**Write to EEPROM** This will write a single byte to an EEPROM location, the command format is:

					<adr>W<EEPROM address>,<value> Care should be taken when using this command for two reasons: 1) The EEPROM can only be written to a certain number of times all be it a large number. 2) The EEPROM contains system information that is used at start up a wrong value could mean loss of communication with the device that would require a factory reset. **I2C Example write 23 to location 7** s 0x91 7 23 p
Rn,m	0x90	n=0-255 m=0-255			**Read from EEPROM** The EEPROM values can be read with this command given a starting address and the number of bytes to read. <adr>R<"EEPROM adr><#bytes> This example will output the first 16 bytes of EEPROM **fR0,16** The output from the device will commence after receiving CR and will consist of a string of data terminated with ACK. The sting will be in the form of text delimited by ',' and all of the values will be decimal. An example of output for the first 5 bytes of EERPOM would be: "0,97,6,21,0"<ACK> **I2C** I2C will read only single values at a time, to read from location 3: s 0x90 3 r g-1 p
D	0xa1				**Device ID** Returns a number representing the device product number as a string **fD** Output from the above would be: "4601"<ACK> **I2C** returns two bytes representing a 16 bit number, high byte first s 0xa1 r g-2 p
I	n/a		1	7	**Toggle Inverted** This command will invert the output of the TX pin and store the value to EEPROM. A value of 1 is inverted and 0 is not inverted. The command will toggle form one to the other. If the TX pin requires changing, instead of ACK being returned by the device a value of 0x3e ('>') is returned. This is easily detected and this command can be issued to correct

					it. Example **fI** This is only needed as a **one time** operation as the change is automatically written to EEPROM
C	0x95				**Reset** Resets an individual device. This is a soft reset. A soft reset will normally be the same as a reset at start-up but this may not always be the case. Obviously no ACK will be returned by this command. Example **fC** **I2C** **s 0x95 p**
V	0xa0				**Version** Returns the firmware version as a string in the format "H.L" Example **fV** **I2C** Sends two bytes, major revision first so 2.7 world be 2 and 7
H	n/a				**Hello** This command is used to check what devices are on the bus. It simply returns ACK but where there is more then one device on the bus the following sudo code will list them: for j = 97 to 122 Send(chr$(j)+"H\r") if ack received then print device j found If a device is found then the other attributes such as device ID can be obtained. Also user information could be stored in the devices EEPROM and retrieved.