Paris

Smart VSCP relay controller

Reversion 1.0 - 2012-01-09

Abstract

Paris is a relay module that connects to a CAN4VSCP bus and can control up to seven external relays which can be up to 1/2 a kilometer from the controller. The module can be attached to a standard DIN Rail or be mounted directly on a wall (ordered separately). The module fully adopts to the CAN4VSCP specification and can be powered directly over the bus with a 9 - 28VDC power source. It has a rich register set for configuration and many information events defined. It also have a decision matrix for easy dynamic event handling.

Grodans Paradis AB
Brattbergavägen 17
820 50 LOS
SWEDEN
web: http://www.auto.grodansparadis.com
email: info@grodansparadis.com
phone: +46 8 40011835

All boards produced by *Grodans Paradis AB* are ROHS compliant.

Contents

1 Paris - smart VSCP relay module **6**

1.1 Most current information . . . 6
1.2 The raw facts . . . 7
1.3 Hardware . . . 7
1.4 Cable and connectors . . . 9
1.4.0.1 RJ-XX pin-out . . . 9
1.4.1 Cable length . . . 11
1.4.2 Termination . . . 11
1.4.2.1 Why are terminators required? . . . 11
1.4.3 Daisy chain connector . . . 12
1.4.4 Power the module . . . 12
1.5 Installing the module . . . 12
1.5.1 Termination block pin-out . . . 13
1.6 Updating firmware . . . 14
1.6.1 Update firmware using the ICP connector . . . 14
1.6.2 Update firmware with VSCP Works . . . 14
1.7 Configure the module? . . . 15
1.7.1 Zone/sub-zone . . . 15
1.7.2 Functionality for the relays . . . 15
1.7.3 Set/get relay state with register read/writes . . . 16
1.7.4 Set/get relay state using the decision matrix . . . 16
1.7.4.1 You want to trigger on a specific event. . . . 17
1.7.4.2 You want to trigger on any event you get from a specific module. . . . 17
1.7.5 Pulsing outputs . . . 17
1.7.6 Protection timer . . . 18
1.7.7 Alarm . . . 18
1.8 Registers . . . 18
1.8.1 Zone registers . . . 18
1.8.2 Relay status registers . . . 19
1.8.3 Relay control registers . . . 19
1.8.4 Relay pulse time registers . . . 20
1.8.5 Relay protection time registers . . . 21
1.8.6 Relay zone information . . . 21
1.8.7 Registers for decision matrix . . . 22
1.9 Decision matrix . . . 22
1.9.1 Action = 0(0x00) . . . 22
1.9.2 Action = 1(0x01) . . . 22

1.9.3 Action = 2(0x02) . . . 22
1.9.4 Action = 3(0x03) . . . 22
1.9.5 Action = 4(0x04) . . . 22
1.9.6 Action = 5(0x05) . . . 23
1.9.7 Action = 6(0x06) . . . 23
1.9.8 Action = 7(0x07) . . . 23
1.9.9 Action = 8(0x08) . . . 23
1.9.10 Action = 9(0x09) . . . 23
1.9.11 Action = 10(0x0a) . . . 23
1.9.12 Action = 11(0x0b) . . . 23
1.9.13 Action = 12(0x0c) . . . 23
1.9.14 Action = 13(0x0d) . . . 23
1.9.15 Action = 14(0x0e) . . . 23
1.9.16 Action = 15(0x0f) . . . 23
1.9.17 Action = 16(0x10) . . . 23
1.10 Alarm register . . . 24
1.11 Events . . . 24
1.11.1 On Event . . . 24
1.11.1.1 Package . . . 24
1.11.2 Off Event . . . 24
1.11.2.1 Package . . . 24
1.11.3 Stopped Event . . . 25
1.11.3.1 Package . . . 25
1.11.4 Started Event . . . 25
1.11.4.1 Package . . . 25
1.11.5 Alarm Event . . . 25
1.11.5.1 Package . . . 25
1.12 Where can I find the source code? . . . 26
1.13 Appendix A - Mandatory VSCP registers. . . . 26

List of Figures

1.1 Schema for the Paris relay module 8
1.2 Road map to module 9
1.3 RJ-45 pin out 10
1.4 CAN4VSCP bus with drops and terminations 11
1.5 Termination 11
1.6 Daisy chain connector 12
1.7 Pin-out for termination block 13
1.8 Connection of relay 14

List of Tables

1.1 The raw facts 7
1.2 RJ-XX pin-out 10
1.3 class/type filter 16
1.4 VSCP mandatory registers 27

Chapter 1

Paris - smart VSCP relay module

Paris is a relay module that connects to a CAN4VSCP bus and can control up to seven external relays. The module can be attached to a standard DIN Rail or be mounted directly on a wall (ordered separately). The module fully adopts to the CAN4VSCP specification and can be powered directly over the bus with a 9-28V DC power source. It has a rich register set for configuration and many information events defined. It also have a decision matrix for easy dynamic event handling.

VSCP CAN modules are designed to work on a VSCP4CAN bus which use ordinary RJ-45 connectors and is powered with 9-28V DC over the same cable. This means there is no need for a separate power cable. All that is needed is a CAT5 or better twisted pair cable. Buss length can be a maximum of 500 meters with drops of maximum 24 meters length (up to a total of 120 meters). As for all VSCP4CAN modules the communication speed is fixed at 125 kbps.

All VSCP modules contains information of there own setup, manual, hardware version, manufacturer etc. You just ask the module for the information you need and you will get it. When they are started up they have a default functionality that often is all that is needed to get a working setup. If the module have something to report it will send you an event and if it is setup to react on a certain type of event it will do it's work when you send event(s) to it.

1.1 Most current information

You can find the most current information about the Paris relay module at http://www.auto.grodansparadis.com/paris/paris.html. On the site you can also find links to the latest firmware and Module Description File (MDF) for the device as well as schematics and recipes for its use. This information is of course pointed to from the MDF file which you can locate from the module itself reading it's standard registers.

Parameter	Value
Supply voltage	9-28VDC
PCB Size	42 mm x 72mm
Power requirements	0.1W + relay driver power if used.
Communication	VSCP4CAN (CAN), 125kbps
Max sink capacity (each relay)	500 mA, 50V
Max relays	7

Table 1.1: The raw facts

1.2 The raw facts

1.3 Hardware

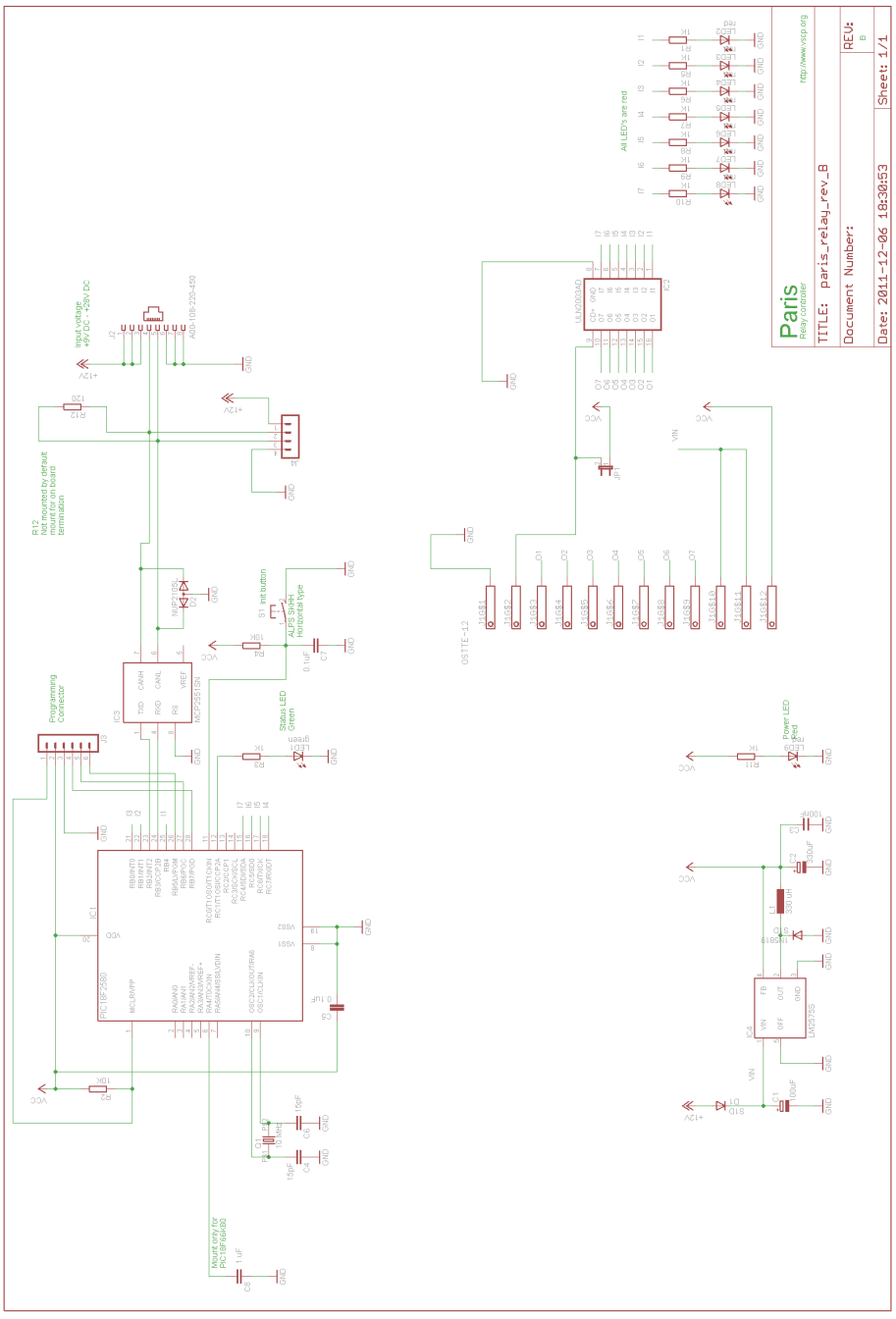

Figure 1.1: Schema for the Paris relay module

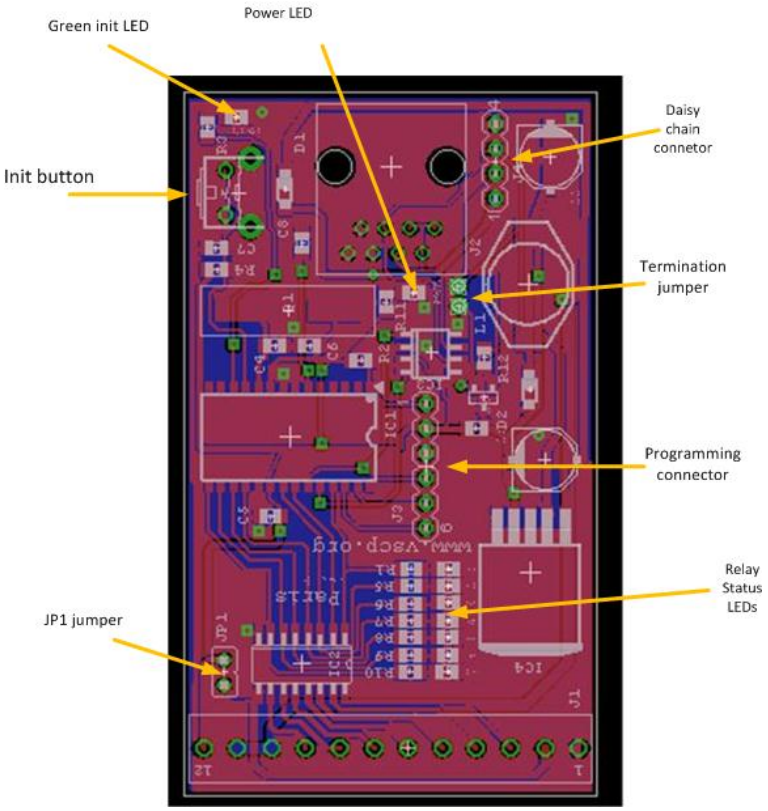

Figure 1.2: Road map to module

Some key positions on the module is outlined in the figure below

1.4 Cable and connectors

The unit is powered over the CAN4VSCP bus. The CAN4VSCP normally uses CAT5 or better twisted pair cable. You can use other cables if you which. The important thing is that the CANH and CANL signals uses a twisted cable. For connectors you can use RJ10, RJ11, RJ12 or the most common RJ45 connectors. There are different versions

1.4.0.1 RJ-XX pin-out

RJ-11/12/45 pin-out

Always use a pair of wires for CANH/CANL fort best noise immunity. If the EIA/TIA 56B standard is used this condition will be satisfied. This is good as most Ethernet networks already is wired this way.

Pin	Use	RJ-11	RJ-12	RJ-45	Patch Cable wire color T568B
1	+9-28V DC			RJ-45	Orange/White
2 1	+9-28V DC		RJ-12	RJ-45	Orange
3 2 1	+9-28V DC	RJ-11	RJ-12	RJ-45	Green/White
4 3 2	CANH	RJ-11	RJ-12	RJ-45	Blue
5 4 3	CANL	RJ-11	RJ-12	RJ-45	Blue/White
6 5 4	GND	RJ-11	RJ-12	RJ-45	Green
7 6	GND		RJ-12	RJ-45	Brown/White
8	GND			RJ-45	Brown

Table 1.2: RJ-XX pin-out

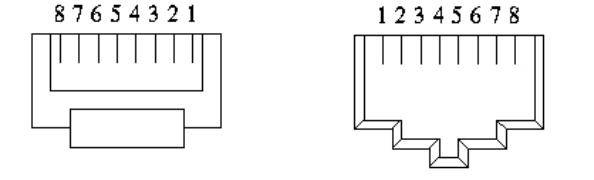

Figure 1:
End view of RJ45 Plug

Figure 2:
Looking into an RJ45 Jack

Figure 1.3: RJ-45 pin out

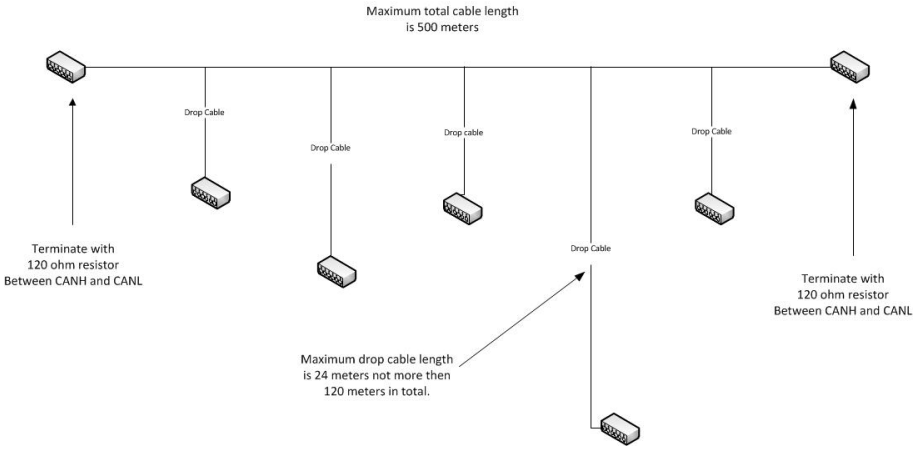

Figure 1.4: CAN4VSCP bus with drops and terminations

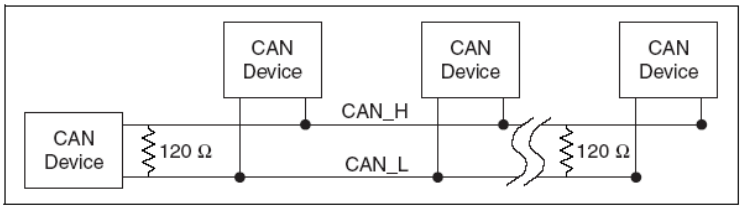

Figure 1.5: Termination

1.4.1 Cable length

CAN4VSCP always communicate with 125kbps. This means that if you use a good quality cable it can be up to a maximum of 500 meters using AWG24 or similar (CAT5) . Actual length depend on the environment and other parameters. Drops with a maximum length of 24 meters can be taken from this cable and the sum of all drops must not exceed a total of 120 meters.

1.4.2 Termination

The CAN4VSCP bus, as all CAN based networks, should be terminated with a 120 ohms resistor between CANH and CANL at both ends of the cable.

If you use CAT5 this termination should be placed between the blue - blue/white cables at both end of your bus.

On the board there is a jumper for an on-board terminator. See figure above.

1.4.2.1 Why are terminators required?

Terminators are needed to cancel signal echos in the cable. In short you get less noice in the cable if you use them. It is recommended to use them even if at

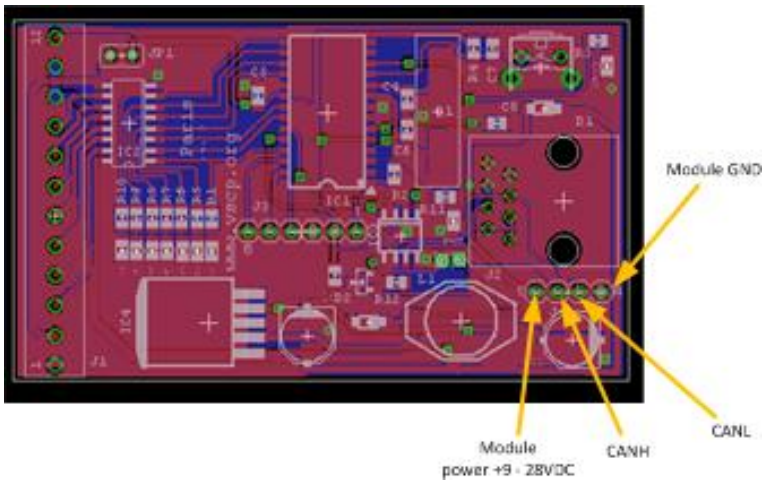

Figure 1.6: Daisy chain connector

125 kbit it is possible that your bus will work anyway.

1.4.3 Daisy chain connector

The daisy chain connector is a pin-header that can be used as an easy way to daisy chain several modules in a cabinet or similar. You just connect the modules together with a straight cable. The pin-out is

1.4.4 Power the module

You normally power the module through the RJ45 connector over the CAN4VSCP bus. See 1.4.0.1 for a description of which pins to use for power and ground. The voltage range is +9VDC - 28VDC. The current need depend on how many modules you want to power.

An alternative way to power the module is through the daisy chain connector described above. Just connect +9V - 28V to it's pin 1 and ground to pin 4. Needless to say you can't have power supplied by the CAN4VSCP bus at the same time.

1.5 Installing the module

Connect the module to the CAN4VSCP bus. The red led on the module should light up indicating that the device is powered. If this is the first time you start up the module the green lamp next to the initializing button will start to blink. This means that the module is trying to negotiate a nickname address with the rest of the modules on the bus. When it found a free nickname the green led will light steady. If the green led does not start to blink press the initialization button until it does. Now your module is ready to use.

You have to decide what power source you should use for your relays. On position 12 of the connectors on the board you can find +5V which can be used

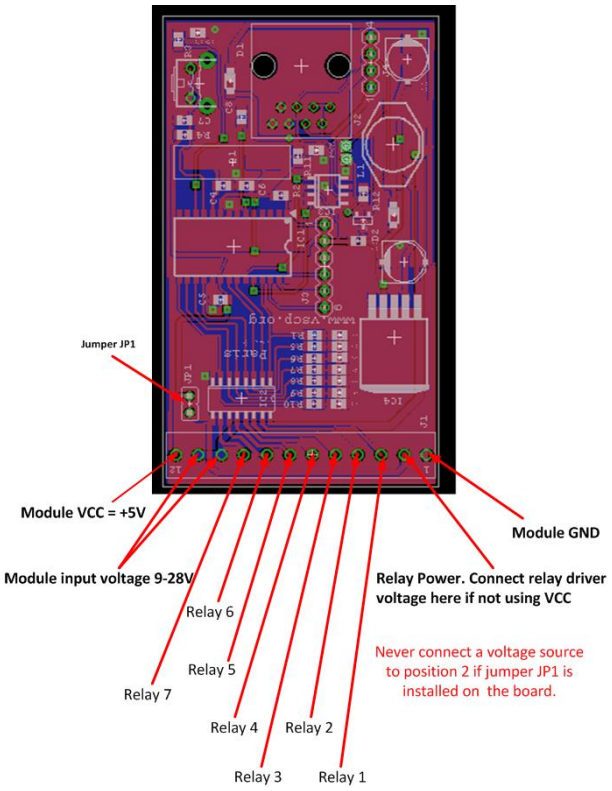

Figure 1.7: Pin-out for termination block

for this purpose. If you want to use +5V install the jumper JP1. Often however +5V is a to low voltage to control relays. You need +12V or even +24V. This is OK. With the Paris module it is possible to use up to +50V DC. **Before you connect this external voltage make sure that jumper JP1 is not installed.**

On position 10/11 you have the supply voltage for the module available. This is +9VDC - 28VDC and can also be used as your relay voltage.

1.5.1 Termination block pin-out

The individual positions for the twelve position termination block is numbered from the left (looking into it) as in the figure below.

Ground, +5V and the power for the module is available from the board and the seven relay control positions. If you plan to use relays that can be controlled with +5V just install jumper JP1 and connect +5V to the relays. Normally however you need to have a higher voltage such as +12V or +24V to control the the relays. In this case the JP1 jumper should be uninstalled. Connect the relay voltage (which can be taken from position 10/11 if a suitable voltage is used to power the module) to your relay and to position 2 of the termination block. By doing this you activate the fly-back protection diodes

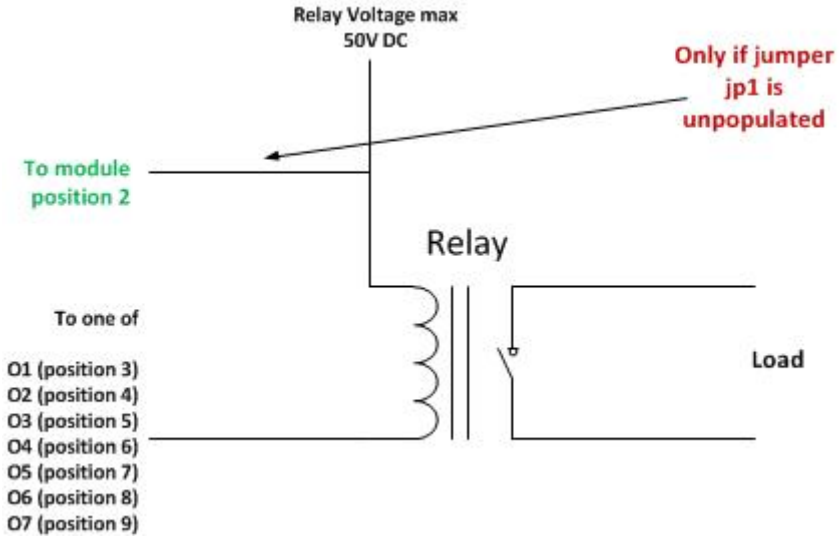

Figure 1.8: Connection of relay

which protects the circuits from inductive loads. If you just have an resistive load you don't have to do this connection but there is no harm if you use it anyway.

The power used to power the Paris module is a perfect candidate for the relay voltage. It is available on the terminal block position 10/11.

1.6 Updating firmware

There is two ways you can use to update the firmware of the module. You can program the device using the programming socket on board or you can use VSCP Works to remotely program the device.

1.6.1 Update firmware using the ICP connector

The firmware of a circuit equipped with a Microchip microprocessor usually can be programmed in circuit. That is when it is mounted on a printed circuit board. This is also true for the Paris relay module which have the programming connector on-board (J3). If you have a programmer for Microchip processors (Real ICE, ICD-2, ICD-3, PICKIT-2, PICKIT3 or other) you can program your own firmware or the latest official firmware into the module using MPLAB or similar tools. You can always find a link to the latest firmware on the Paris relay module home page (http://www.auto.grodansparadis.com/paris/paris.html).

1.6.2 Update firmware with VSCP Works

When a module is installed in a remote location or if you don't have a Microchip programmer you can program the module using the built in boot-loader. This

can be done with VSCP Works a program that can be run on the Windows or the Linux platform and can perform and can perform different maintenance, configuration and status checks of VSCP modules. If you have not installed the VSCP & Friends package it is time to do so now. You can always find the latest version on the VSCP projects download page (http://vscp.org/downloads.php).

The boot loader process using VSCP Works is described in section 16.4 of the VSCP specification. The Paris relay module uses the *PIC1* boot loader.

1.7 Configure the module?

You configure a VSCP module by writing content into the modules registers. You can do this manually or with the wizard available in VSCP works. Using the wizard is absolutely the easiest way to use.

1.7.1 Zone/sub-zone

You should always plan your overall structure. The zone and the sub-zone registers found in the first two register positions can help you here. Think of a zone as a house, floor plan or similar and sub-zone as a room or a location. Note that this is not an address. It's a way to group functionality together. Each relay can belong to it's own sub-zone. Set it in register 50-56.

1.7.2 Functionality for the relays

As you may expect the main functionality for the Paris relay module is the possibility to turn a relay on or off. Even if this is the main functionality there is some extra functionality available. Each relay have a control byte with flags (bits) which control different functionality for each relay. The bits have the following meaning

- bit 7 - Must be set to one to make it possible to control the relay. Both register writes, pulse and actions are ignored when the bit is set to zero.
- bit 6 - If set a STOPPED event (CLASS=20, TYPE=24) will be sent when the relay goes to it's inactive state.
- bit 5 - If set a STARTED event (CLASS=20, TYPE=25) will be sent when the relay goes to it's active state.
- bit 4 - If set an OFF event (CLASS=20, TYPE=4) will be sent when the relay goes to it's inactive state.
- bit 3 - If set an ON event (CLASS=20, TYPE=3) will be sent when the relay goes to it's active state.
- bit 2 - Enables the protection timer if set to one. See 1.7.6
- bit 1 - Alarm is sent if the protection timer elapses if this bit is set to one.
- bit 0 - Pulse output enabled if set to one. See 1.7.5

Mask bit n	Filter bit n	Incoming event class bit n	Accept or reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Table 1.3: class/type filter

1.7.3 Set/get relay state with register read/writes

A relay can be active (on) or inactive(off). In registers 2-8 you can read or write the status for an individual relay. A zero mean the relay is off, and a one that it is on. So writing a one to register 2 of a Paris relay module will turn relay 1 on. A better way is to use the decision matrix of the module to accomplish this.

1.7.4 Set/get relay state using the decision matrix

Normally the decision matrix of the module is the best way to to handle relay state changes. This much more flexible then to use the register writes directly. This is also why the zone/sub-zone registers should be initialized with meaningful values. The decision matrix of this module can have eight entries. Each entry is a line that look for a specified event on the bus and perform a specific action if this event is found. Typically this can be a ON-event sent to a specific zone/sub-zone. A row in the decision matrix can be setup to turn on one or more relays if this event is detected and if the zone/sub-zone is right.

At first the setup of the decision matrix can be hard to grasp. You can read all about it in the VSCP specification section 7.3. Here we will just at a simple example.

The decision matrix consist of seven bytes. The first byte is the originating address. You can set a nickname here for the node that is allowed to trigger (perform) a selected action. The control byte have configuration bits(flags) for how the information should be interpreted. One bit (bit 6) tells if the originating address should be checked or not. That is if bit 6 is set then the originating address must be the same as the byte stored in the first byte of the decision matrix to trigger the action. Bit 7 must always be set for the row to be compared ton an incoming event. Bit four and five, if set, check if byte 1 and 2 of the incoming event is the same as the zone/sub-zone stored in register3 0/1 of the module.

Byte 2/3 together with bits 0/1 of the control register is the class mask and filter. A ninth bit is needed as a class consist of nine bits. Byte 4/5 is the type mask and filter. This is something most newcomers to VSCP have problems with. But is is actually really simple.

The following table illustrates how this works

Think of the mask as having ones at positions that are of interest and the filter telling what the value should be for those bit positions that are of interest.

- So to only accept one class set all mask bits to one and enter the class in filter.

- To accept all classes set the class mask to 0. In this case filter don't care.
- To accept everything set both masks to zero and the filter to any value you like.

Common cases are

1.7.4.1 You want to trigger on a specific event.

1. Set all bits of the class mask to ones (255 plus bit 1 of the control register set to one) and all bits of the type mask to ones (255).
2. Write the class that should trigger the event into the class filter (remember that bit nine goes to bit 0 of the control byte) and the type to the type filter. So if you want to trigger on a TurnOn event which have CLASS=30, TYPE=5 you set the class filter = 30 (class bit i control byte to zero) and the type filter = 5.
3. Now if the originating address bit is not set and the row is enabled (bit 7 of the control register is set) and incoming event of the specified type will trigger the action.

1.7.4.2 You want to trigger on any event you get from a specific module.

1. Enter the nickname of the module in the first byte of the decision matrix.
2. Set the control byte to 192 (bit 6 (check originating address) and bit 7 (enable decision matrix row) set).
3. Set byte 2-5 (masks and filters) to zero. Actually class and type filters can have any value as long as the masks are zero.
4. Set the appropriate action/action parameter values. For example action=1 and action parameter = 3 (0b00000011) to activate relay 0 and 1 when any event is received from the node.

The decision matrix makes it very easy to set up systems where one event triggers actions on several modules. It is also easy to adopt your system to new demands. You can add and replace modules without effecting the system functionality.

1.7.5 Pulsing outputs

If you want your relay output to be turned on/off with a certain interval the Paris relay module automatically can handle this for you. First set the time the relay should be on/off in register 18-33. There are two registers for each relay and the time is given in seconds. The lower byte holds the most significant byte and the higher byte holds the least significant byte. As and example: If you want relay 0 to have an output that is on for five minutes and then off for five minutes and so on you first calculate how many seconds the pulse time is. In this case 5 * 60 = 300 seconds. This is what should go into register 18 and 19. 300/256 = 1 which is what should go into register 18 and the rest (300 - 1*256

= 44) 44 forms the least significant byte and should go to register 19. To start the pulse output you also have to set bit 0 in the corresponding relay control register to one.

1.7.6 Protection timer

A protection timer can be convenient to use to protect a system form malfunctions in some of it's components. Suppose you have a relay that controls a pump that fills a tank with some liquid. The pump should then be turned off when a the tanks is full which is indicated by an event from a sensor on the tank. If this sensor is broken or the cable to the module is broken this event will not be received. With the protection timer activated the pump will be turned of anyway before things get to bad. It is also possible to send an alarm when this happens to inform the rest of the system.

The protection timers are in registers 34-47 and comes, like the pulse registers, in pairs. The first byte in a pair holds the most significant byte of the timer and the second the least significant byte. Also as for the pulse time the time set is given in seconds. A maximum of 65535 seconds can thus be stored. So the range is one seconds up to around eighteen hours.

The protection timer, if activated, is started when you activate the protected relay. If the relay is active when the timer elapse it will be turned off. If you activate the relay again before the timer has elapsed the protection timer will be resettled and start to count down from the preset value again. If you turn of the relay the timer will be inactivated until the relay is activated again. Both relay control through actions and register writes affects the timer.

1.7.7 Alarm

The module can send alarm events if the protection timer elapses. When an alarm occurs a bit is set in the alarm register which is located in register position 128. You can always read this register to see if the module have sent out and alarm events. When you read the register the alarm bits will be cleared.

1.8 Registers

All VSCP modules have a set of 8-bit registers defined. Some of them (register 128-255) are predefined and the information in them are the same for all VSCP modules. See the VSCP specification for a description of their content (http://sourceforge.net/projects/m2m/files/VSCP%20Specification/). The lower 128 register positions are used for module specific registers. It is normally here you find registers with which you configure your module. You can also find registers where you typically can read status information such as measurement data from the module.

Below is a description of the registers on the Paris smart relay controller.

1.8.1 Zone registers

- **Register 0**(0x00) - Zone.
- **Register 1**(0x01) - Sub-zone.

1.8.2 Relay status registers

- **Register 2**(0x02) - Relay 1 Status register. Read/Write.
- **Register 3**(0x03) - Relay 2 Status register. Read/Write.
- **Register 4**(0x04) - Relay 3 Status register. Read/Write.
- **Register 5**(0x05) - Relay 4 Status register. Read/Write.
- **Register 6**(0x06) - Relay 5 Status register. Read/Write.
- **Register 7**(0x07) - Relay 6 Status register. Read/Write.
- **Register 8**(0x08) - Relay 7 Status register. Read/Write.
- **Register 9**(0x09) - Reserved.

Writing a value to the relay control register will activate/deactivate the relay output.

- 0 - The relay is inactivated.
- 1 - The relay is activated.

Reading a value from the relay control register is read as a one if the relay is activated and a 0 if the relay is deactivated.

1.8.3 Relay control registers

- **Register 10**(0x0A) - Relay 1 Control Register. Read/Write.
- **Register 11**(0x0B) - Relay 2 Control Register. Read/Write.
- **Register 12**(0x0C) - Relay 3 Control Register. Read/Write.
- **Register 13**(0x0D) - Relay 4 Control Register. Read/Write.
- **Register 14**(0x0E) - Relay 5 Control Register. Read/Write.
- **Register 15**(0x0F) - Relay 6 Control Register. Read/Write.
- **Register 16**(0x10) - Relay 7 Control Register. Read/Write.
- **Register 17**(0x11) - Reserved.

The relay control bits enable/disable intelligent relay functionality:

- **Bit 0** - Enable pulsed output if set to one.
- **Bit 1** - Alarm sent when protection timer triggers (if set).
- **Bit 2** - Protection timer enable if set to one.
- **Bit 3** - Send On event (CLASS=20, TYPE=3) when relay goes to active state.
- **Bit 4** - Send Off event (CLASS=20, TYPE=4) when relay goes to inactive state.

- **Bit 5** - Send Started event (CLASS=20, TYPE=25) when relay goes to active state.
- **Bit 6** - Send Stopped event (CLASS=20, TYPE=24)when relay goes to inactive state.
- **Bit 7** - If set to one the relay is enabled. If set to zero it is inactivated.

1.8.4 Relay pulse time registers

- **Register 18**(0x12) - On/off pulse time Relay 1 (seconds) MSB. Read/Write.
- **Register 19**(0x13) - On/off pulse time Relay 1 (seconds) LSB. Read/Write.
- **Register 20**(0x14) - On/off pulse time Relay 2 (seconds) MSB. Read/Write.
- **Register 21**(0x15) - On/off pulse time Relay 2 (seconds) LSB. Read/Write.
- **Register 22**(0x16) - On/off pulse time Relay 3 (seconds) MSB. Read/Write.
- **Register 23**(0x17) - On/off pulse time Relay 3 (seconds) LSB. Read/Write.
- **Register 24**(0x18) - On/off pulse time Relay 4 (seconds) MSB. Read/Write.
- **Register 25**(x019) - On/off pulse time Relay 4 (seconds) LSB. Read/Write.
- **Register 26**(0x1A) - On/off pulse time Relay 5 (seconds) MSB. Read/Write.
- **Register 27**(0x1B) - On/off pulse time Relay 5 (seconds) LSB. Read/Write.
- **Register 28**(0x1C) - On/off pulse time Relay 6 (seconds) MSB. Read/Write.
- **Register 29**(0x1D) - On/off pulse time Relay 6 (seconds) LSB. Read/Write.
- **Register 30**(0x1E) - On/off pulse time Relay 7 (seconds) MSB. Read/Write.
- **Register 31**(0x1F) - On/off pulse time Relay 7 (seconds) LSB. Read/Write.
- **Register 32**(0x20) – On/off pulse time Relay 8 (seconds) MSB. Read/Write.
- **Register 33**(0x21) – Reserved.

This is the pulse time for the each relay expressed in seconds. This can be used to have a relay turn on and off with a certain preset interval. The minimum pulse time is 1 second and the maximum time is 65535 seconds which is about 18 hours. Set to zero (default) for no pulse time i.e. the relay will be steady on/off.

To start a pulse sequence bit 0 for the corresponding relay should be set to one.

1.8.5 Relay protection time registers

- **Register 34**(0x22) - Protection time Relay 1 (seconds) MSB. Read/Write.
- **Register 35**(0x23) - Protection time Relay 1 (seconds) LSB. Read/Write.
- **Register 36**(0x24) - Protection time Relay 2 (seconds) MSB. Read/Write.
- **Register 37**(0x25) - Protection time Relay 2 (seconds) LSB. Read/Write.
- **Register 38**(0x26) - Protection time Relay 3 (seconds) MSB. Read/Write.
- **Register 39**(0x27) - Protection time Relay 3 (seconds) LSB. Read/Write.
- **Register 40**(0x28) - Protection time Relay 4 (seconds) MSB. Read/Write.
- **Register 41**(0x29) - Protection time Relay 4 (seconds) LSB. Read/Write.
- **Register 42**(0x2A) - Protection time Relay 5 (seconds) MSB. Read/Write.
- **Register 43**(0x2B) - Protection time Relay 5 (seconds) LSB. Read/Write.
- **Register 44**(0x2C) - Protection time Relay 6 (seconds) MSB. Read/Write.
- **Register 45**(0x2D) - Protection time Relay 6 (seconds) LSB. Read/Write.
- **Register 46**(0x2E) - Protection time Relay 7 (seconds) MSB. Read/Write.
- **Register 47**(0x2F) - Protection time Relay 7 (seconds) LSB. Read/Write.
- **Register 48**(x030) - Protection time Relay 8 (seconds) MSB. Read/Write.
- **Register 49**(0x31) - Reserved.

This is the relay protection time. A relay will be inactivated if not written to before this time has elapsed. Set to zero to disable (default). The max time is 65535 seconds which is about 18 hours.

The registers can for example be used as a security feature to ensure that an output is deactivated after a preset time even if the controlling device failed to deactivate the relay.

1.8.6 Relay zone information

- **Register 50**(0x32) - Relay 1 Sub Zone.
- **Register 5**1(0x33) - Relay 1 Sub Zone.
- **Register 52**(0x34) - Relay 1 Sub Zone.
- **Register 53**(0x35) - Relay 2 Sub Zone.
- **Register 54**(0x36) - Relay 3 Sub Zone.
- **Register 55**(0x37) - Relay 1 Sub Zone.
- **Register 56**(0x38) - Relay 4 Sub Zone.
- **Register 57**(0x39) - Relay 1 Sub Zone.

- **Register 5**8(0x3A) - Relay 5 Sub Zone.
- **Register 5**9(0x3B) - Relay 1 Sub Zone.
- **Register** 60(0x3C) - Relay 6 Sub Zone.
- **Register** 61(0x3D) - Relay 1 Sub Zone.
- **Register** 62(0x3E) - Relay 7 Sub Zone.
- **Register** 63(0x3F) - Relay 1 Sub Zone.
- **Register** 64(0x40) - Reserved
- **Register** 65(0x41) - Relay 1 Sub Zone.

This is the zone and sub-zone value related to a specific relay. If zero the the module Zone/sub-zone will be used.

1.8.7 Registers for decision matrix

- **Register 72**(0x48) - **127**(0x7f) - Decision Matrix with seven rows.

1.9 Decision matrix

1.9.1 Action = 0(0x00)

NOOP, No action.

1.9.2 Action = 1(0x01)

Activate relay(s) given by argument. The argument is a bit array where bit 0 is relay 1 and so on. That is a position with a set bit will activate the corresponding relay. Byte 1 is Zone and byte 2 is sub-zone and must be equal to register content to trigger action.

1.9.3 Action = 2(0x02)

Deactivate relay(s) given by argument. The argument is a bit array where bit 0 is relay 1 and so on. That is a position with a set bit will inactivate the corresponding relay. Byte 1 is Zone and byte 2 is sub-zone and must be equal to register content to trigger action.

1.9.4 Action = 3(0x03)

Pulse relay(s) given by argument. The argument is a bit array where bit 0 is relay 1 and so on. That is a position with a set bit will toggle the corresponding relay. Byte 1 is Zone and byte 2 is sub-zone and must be equal to register content to trigger action.

1.9.5 Action = 4(0x04)

Reserved.

1.9.6 Action = 5(0x05)

Reserved.

1.9.7 Action = 6(0x06)

Reserved.

1.9.8 Action = 7(0x07)

Reserved.

1.9.9 Action = 8(0x08)

Reserved.

1.9.10 Action = 9(0x09)

Reserved.

1.9.11 Action = 10(0x0a)

Send relay status. The argument is a bit array where bit 0 is relay 1 and so on. That is a position with a set bit will have a status event sent for the corresponding relay.

1.9.12 Action = 11(0x0b)

Reserved.

1.9.13 Action = 12(0x0c)

Reserved.

1.9.14 Action = 13(0x0d)

Reserved.

1.9.15 Action = 14(0x0e)

Reserved.

1.9.16 Action = 15(0x0f)

Reserved.

1.9.17 Action = 16(0x10)

Disable relay(s) given by argument. The argument is a bit array where bit 0 is relay 1 and so on. That is a position with a set bit will disable the corresponding relay. Byte 1 is Zone and byte 2 is zone page and must be equal to register content to trigger action.

1.10 Alarm register

- **Bit 0** Relay 1 protection timer has caused a relay action.
- **Bit 1** Relay 2 protection timer has caused a relay action.
- **Bit 2** Relay 3 protection timer has caused a relay action.
- **Bit 3** Relay 4 protection timer has caused a relay action.
- **Bit 4** Relay 5 protection timer has caused a relay action.
- **Bit 5** Relay 6 protection timer has caused a relay action.
- **Bit 6** Relay 7 protection timer has caused a relay action.
- **Bit 7 Reserved**

Read the register to clear alarm bits.

1.11 Events

1.11.1 On Event

If enabled the event is sent when a relay goes to its active state.

Class: 0x014 Type: 0x03

1.11.1.1 Package

- Byte 0: Index.
- Byte 1: Zone
- Byte 2: Sub-zone

Index is 0 for relay 1, 1 for relay 2 and so on. zone and sub-zone set accordingly. Sub-zone for relay is used if it's not zero.

1.11.2 Off Event

If enabled the event is sent when a relay goes to its inactive state.

Class: 0x014 Type: 0x04

1.11.2.1 Package

- Byte 0: index.
- Byte 1: Zone
- Byte 2: Sub-zone

Index is 0 for relay 1, 1 for relay 2 and so on. Zone and sub-zone set accordingly. Sub-zone for relay is used if it's not zero.

1.11.3 Stopped Event

If enabled the event is sent when a relay goes to its inactive state.

Class: 0x014 Type: 0x18

1.11.3.1 Package

- Byte 0: index.
- Byte 1: Zone
- Byte 2: Sub-zone

Index is 0 for relay 1, 1 for relay 2 and so on. Zone and sub-zone set accordingly. Sub-zone for relay is used if it's not zero.

1.11.4 Started Event

If enabled the event is sent when a relay goes to its active state.

Class: 0x014 Type: 0x19

1.11.4.1 Package

- Byte 0: index.
- Byte 1: Zone
- Byte 2: Sub-zone

Index is 0 for relay 1, 1 for relay 2 and so on. Zone and sub-zone set accordingly. Sub-zone for relay is used if it's not zero.

1.11.5 Alarm Event

If enabled the event is sent when a relay goes to its inactive state after a protection timer have timed out.

Class: 0x001 Type: 0x02

1.11.5.1 Package

- Byte 0: index.
- Byte 1: Zone
- Byte 2: Sub-zone

Index is 0 for relay 1, 1 for relay 2 and so on.

1.12 Where can I find the source code?

Most VSCP modules from Grodans Paradis AB is Open hardware/Open source meaning that both the hardware information as well as the source code is available. This means that you can modify the source code and /or the hardware to your specific needs if you want.

1.13 Appendix A - Mandatory VSCP registers.

Address	Access Mode	Description
0×00 – 0x7f	—	Device specific. *Unimplemented registers should return zero.*
128/0×80	Read Only	Alarm status register content (!= 0 indicates alarm). Condition is reset by a read operation. The bits represent different alarm conditions.
129/0×81	Read Only	VSCP Major version number this device is constructed for.
130/0×82	Read Only	VSCP Minor version number this device is constructed for.
131/0×83	Read/Write	Node control flags Bit: 7 Start-up control; 6 Start-up control; 5 r/w control of registers below 0×80. (1 means write enabled); 4 Reserved; 3 Reserved; 2 Reserved; 1 Reserved; 0 Reserved
132/0×84	Read/Write	User ID 0 – Client settable node id byte 0.
133/0×85	Read/Write	User ID 1 – Client settable node id byte 1.
134/0×86	Read/Write	User ID 2 – Client settable node id byte 2.
135/0×87	Read/Write	User ID 3 – Client settable node id byte 3.
136/0×88	Read/Write	User ID 4 – Client settable node id byte 4.
137/0×89	Read only	Manufacturer device ID byte 0.
138/0x8a	Read only	Manufacturer device ID byte 1.
139/0x8b	Read only	Manufacturer device ID byte 2.
140/0x8c	Read only	Manufacturer device ID byte 3.
141/0x8d	Read only	Manufacturer sub device ID byte 0.
142/0x8e	Read only	Manufacturer sub device ID byte 1.
143/0x8f	Read only	Manufacturer sub device ID byte 2.
144/0×90	Read only	Manufacturer sub device ID byte 3.
145/0×91	Read only	Nickname id for node if assigned or 0xff if no nickname id assigned.
146/0×92	Read/Write	Page select register MSB
147/0×93	Read/Write	Page Select register LSB
148/0×94	Read Only	Firmware major version number.
149/0×95	Read Only	Firmware minor version number.
150/0×96	Read Only	Firmware sub minor version number.
151/0×97	Read Only	Boot loader algorithm used. 0Xff for no boot loader support. Codes for algorithms are specified here VSCP_event_class_000 for Type = 12
152/0×98	Read Only	Buffer size. The value here gives an indication for clients that want to talk to this node if it can support the larger mid level Level I control events which has the full GUID. If set to 0 the default size should used. That is 8 bytes for Level I and 512-25 for Level II.
153/0×99	Read Only	Number of register pages used. If not implemented one page is assumed.
154/0x9A-207/0xcf	—	Reserved for future use. Return zero.
208/0xd0-223/0xdf	Read Only	128-bit (16-byte) globally unique ID (GUID) identifier for the device. This identifier uniquely identifies the device throughout the world and can give additional information on where driver and driver information can be found for the device. MSB for the identifier is stored first (in 0xd0).
224/0xe0-255/0xff	Read Only	Module Description File URL. A zero terminates the ASCII string if not exactly 32 bytes long. The URL points to a file that gives further information about where drivers for different environments are located. Can be returned as a zero string for devices with low memory. It is recommended that unimplemented registers read as oxff. For a node with an embedded MDF return a zero string. The CLASS1.PROTOCOL, Type=34/35 can then be used to get the information if available.

Table 1.4: VSCP mandatory registers

Index

+12V, 13
+24V, 13
+5V, 13

Action, 22, 23
action, 16
Alarm, 15, 18
alarm, 18
Alarm Event, 25
Alarm register, 24
Alarm status register, 27
AWG24, 11

boot loader, 15
Boot loader algorithm, 27
boot-loader, 14

Cable length, 11
CAN, 11
CANH, 9, 11
CANL, 9, 11
CAT5, 6, 9, 11
class, 17
class filter, 16, 17
class mask, 16, 17
class/type filter, 16
configuration, 15
configuration bits, 16
Configure, 15
Connection of relay, 14
connectors, 9
control byte, 17
control registers, 19
current, 12

Daisy chain, 12
daisy chain, 12
Decision matrix, 22
decision matrix, 16, 17, 22
Drops, 11

Events, 24

firmware, 14
fly-back protection diodes, 13
free nickname, 12

green lamp, 12
Ground, 13
GUID, 27

Hardware, 7

ICD-2, 14
ICD-3, 14
ICE, 14
ICP connector, 14
inductive loads, 14
initializing button, 12
Installing, 12

JP1, 13

key positions, 9

least significant byte, 17

maintenance, 15
Mandatory VSCP registers, 26
MDF, 6
Microchip microprocessor, 14
Module Description File, 6, 27
most significant byte, 17
MPLAB, 14

nickname, 16, 17
nickname address, 12

OFF event, 15
Off Event, 24
official firmware, 14
ON event, 15
On Event, 24
on-board terminator, 11
Open hardware, 26

Open source, 26
originating address, 16, 17

PIC1, 15
PICKIT-2, 14
PICKIT3, 14
Pin-out, 13
Power, 12
power, 12, 13
power source, 12
programming socket, 14
Protection timer, 18
protection timer, 15, 18
protection timers, 18
Pulse output, 15
pulse registers, 18
pulsed output, 19
Pulsing outputs, 17

Real ICE, 14
red led, 12
Registers, 18
registers, 15
relay control bits, 19
Relay protection time, 21
Relay pulse time, 20
relay state, 16
relay state changes, 16
relay status, 16
Relay status registers, 19
relay voltage, 13
Relay zone, 21
remotely program, 14
resistive load, 14
RJ-45, 6
RJ-45 pin out, 10
RJ-XX, 9
RJ10, 9
RJ11, 9
RJ12, 9
RJ45, 9, 12

Schema, 8
source code, 26
STARTED event, 15
Started Event, 25
status checks, 15
status information, 18
STOPPED event, 15
Stopped Event, 25
sub-zone, 15
supply voltage, 13

T568B, 10
terminated, 11
Termination, 11
termination, 11
Termination block, 13
termination block, 13
terminations, 11
Terminators, 11
trigger action, 17
TurnOn event, 17
twelve position termination block, 13
twisted cable, 9
twisted pair cable, 9
type mask, 17

Update firmware, 14
Updating firmware, 14

voltage, 12
VSCP & Friends package, 15
VSCP specification, 15, 16, 18
VSCP Works, 14, 15
VSCP works, 15

wizard, 15

Zone, 15
Zone registers, 18