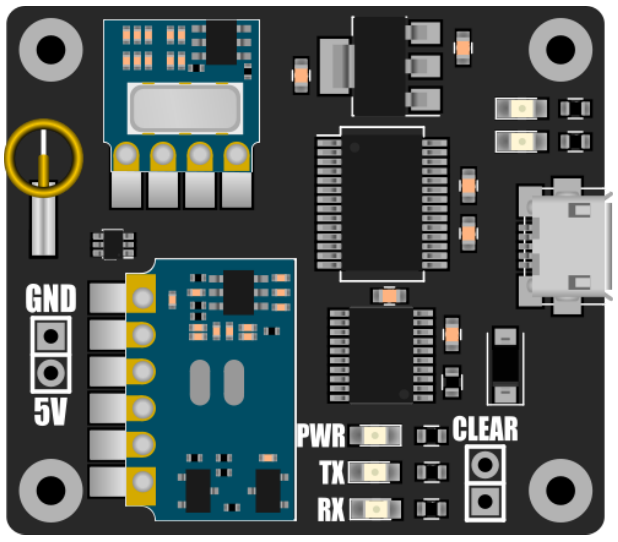
# SmartRFy Digital USB Module Manual



# HCMODU0144

Revision 1.0.0

## DISCLAIMER

**SmartRFy modules are micro power short range devices (SRD) using the 433MHz frequency band. This frequency band is licensed exempt (within certain restrictions) in many countries including the UK, Europe, Asia, and the US. It is the end users responsibility to ensure that it is legal to operate devices in this frequency band within your own country before use.**

As with all wireless devices, external factors such as range and interference may cause transmissions to be corrupted or blocked. Therefore these devices should not be used in applications where control or monitoring is a critical requirement.

Mercury is a brand name of AVSL group Ltd. Mercury branded energy saving sockets are a third party product and Hobby Components Ltd has no affiliation with the Mercury brand or AVSL group Ltd. When using Mercury devices any safety instructions, requirements, and specifications supplied with the product must be followed. Hobby Components accepts no responsibility for and loss or damage relating to the use of these third party products. Any technical questions relating to the use of SmartRFy modules with Mercury branded products should be directed at Hobby Components only.

## COPYRIGHT NOTICE

# Table of Contents

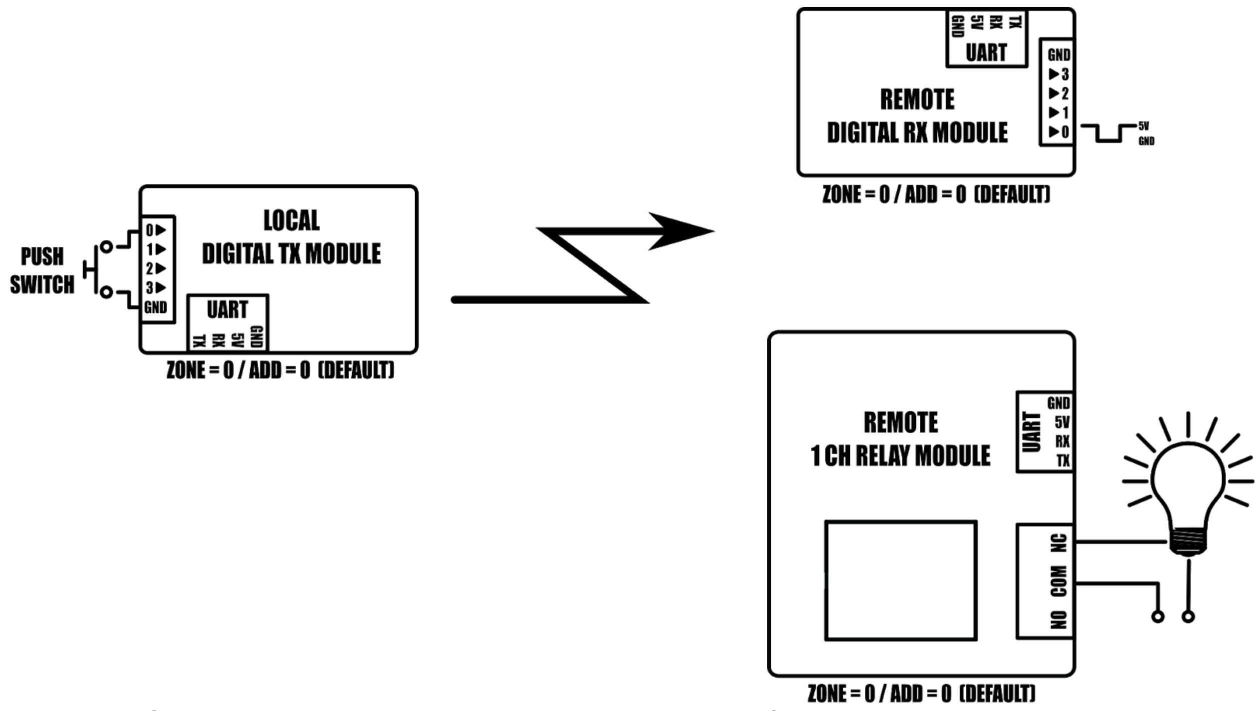
## SmartRFy system overview

SmartRFy modules are a range of wireless modules that provide a simple low cost way of controlling and monitoring remote devices and sensors. They are designed to work at their basic level with no programming required. However, when reconfigured or controlled via their serial interface they can also provide more complex and even autonomous functions.

SmartRFy modules work by wirelessly communicating with one another using simple text based commands - for example, to turn on a remote relay a command such as RLY=1 can be sent, and to turn it back off RLY=0. These commands can be automatically sent from one module to another or can be sent by a user via a module's serial interface to manually control modules on the SmartRF network.

They are addressable and can be grouped into one of 255 zones, with each zone capable of individually addressing 255 devices, giving a total of over 65,000 unique addresses. SmartRFy modules are designed to be both easy and flexible to use by providing three levels of control:

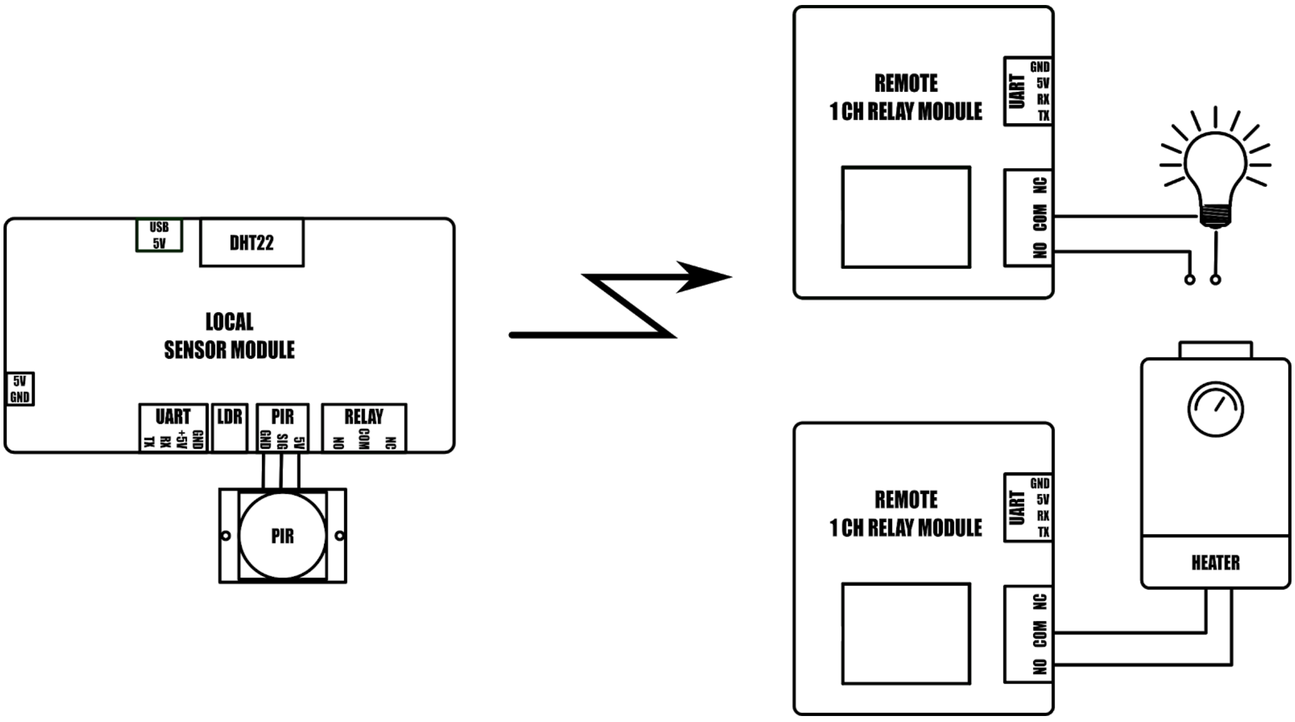
**Zero configuration (out-of-the-box)**



***Example: Controlling a remote digital pin and light from a push switch.***

With no configuration at all the SmartRFy range of modules allow basic functionality such as remotely switching relays, digital pins, or monitoring sensors, such as temperature, humidity, light, motion etc. All SmartRFy modules default to the same zone and address and so can respond to, or control, other modules without the need to modify any zone or address settings.

## Serial port configuration

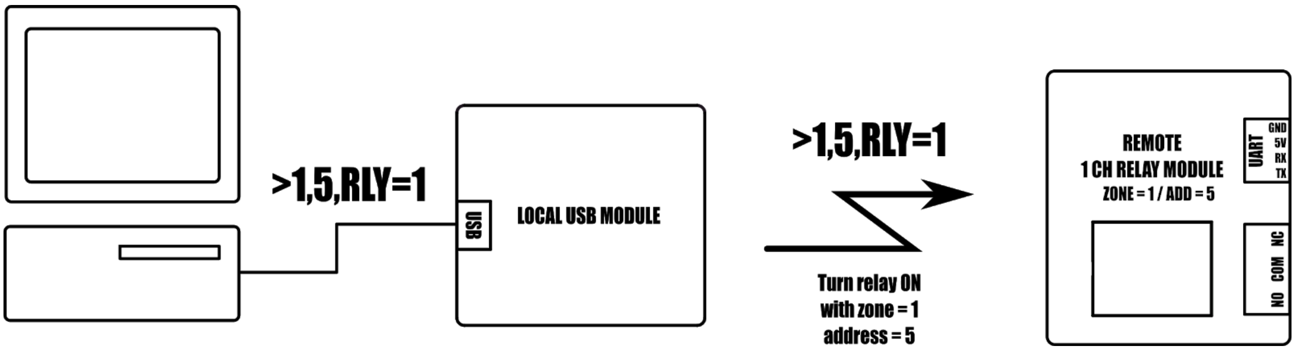


*Example: Remote 1CH relay module configured to turn on light when local modules' PIR is triggered and second remote relay module is configured to control a heating device based on local modules' temperature sensor readings.*

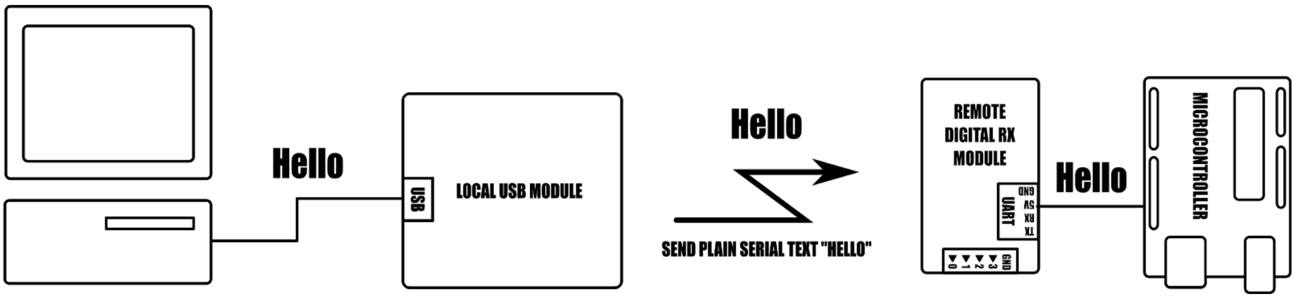
All smartRFy modules include a serial port which allows them to be re-configured using simple text based commands. These text commands allow changing of configuration settings and control of specific features of each module. Any setting changes are stored by the module's non-volatile memory and so are retained even when power is removed from the module. When reconfigured they are able to perform more complex functions and even some basic autonomous tasks. For example, a SmartRFy relay module can be reconfigured to control a heating system based on the temperature transmitted from a remote sensor or, turn on a flood light for a set amount of time when triggered by a remote

PIR – all without any additional hardware. Serial port configuration also allows a module's zone and address (all SmartRFy modules default to zone and address 0) to be changed. This provides the option for modules to be placed into groups, which will then only respond to other modules with a matching zone or address.

**Slave controlled**



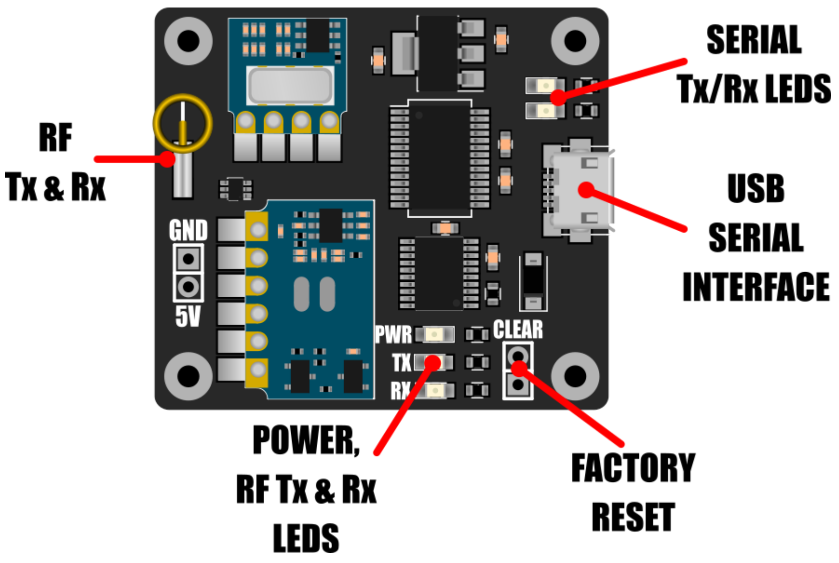
*Example 1: Turning on a remote relay with zone = 1 and address = 5*



*Example 2: Send plain serial text “Hello” to a remote microcontroller*

SmartRFy modules can also be controlled via their serial port by other devices such as microcontrollers or computer based automation software such as Node-RED. Using a SmartRFy module as a slave device (must be capable of transmitting/receiving) gives a master device the ability to monitor and control any other SmartRFy devices on the network, regardless of their zone and address. SmartRFy modules can even be used as a passive wireless serial port for passing non-SmartRFy data to other remotely connected devices.

# SmartRFy USB module



The SmartRFy USB module adds the ability to monitor and control a SmartRFy network directly from a computer. Connection to the computer is simply a case of connecting the module to the computer's USB port (via a micro USB cable - not supplied). Once connected the USB module will appear as a standard serial communication port which can then be used to send and receive text based SmartRFy commands to any remote SmartRFy module. By appearing as a standard serial port many types of third party software can be used with the USB module - from serial terminal programs to dedicated automation software such as Node-RED. This opens up the potential for far more advanced control and automation of SmartRFy modules.

The SmartRFy USB module also has the additional feature of being able to directly control Mercury branded energy saving mains socket adapters (see Hobby Components SKU: HCPOWE0010). These are remotely controlled adapters that sit between your electrical appliance and the mains socket and allow mains power to the appliance to be remotely controlled. Using the SmartRFy USB module you can individually control up to a total of 192 Mercury sockets.

Power for the module is supplied via the USB connection and so an external power supply is not required.

## Features

Standard microUSB interface

Add computer control of a SmartRFy network

Appears as a standard virtual com port (VCP) serial interface

Compatible with common operating systems (Windows, OSX, Linux, etc)

Can be used with most software supporting communication via serial interface including serial terminal software, Node-RED, etc.

Compatible with Mercury branded energy saving sockets

Powered directly from the computer USB interface.

Control monitor and control SmartRFy device using simple text based commands.

Wireless serial communication with 8 different baud rates (9600 default).

## Specification

| | |
|---|---|
| Model number: | HCMODU0144 |
| Supply Voltage: | 4.5 to 5.5V |
| Supply current min: | 19mA (idle) |
| Supply current max: | 30.5mA (transmit) |
| Operating frequency: | 433MHz (OOK) |
| Operating range: | 30 Meters (unobstructed) |
| Interfaces: | RF Tx & Rx, and USB serial |
| Module dimensions (WxDxH): | 41.3mm x 36.6mm x 4.7mm |

## Driver Installation

The SmartRFy USB module uses an FTDI Chip FT232RL to interface to a computer via its USB port. This IC is well supported on all popular operating systems and in most cases the driver will be automatically installed the first time the module is plugged into the PC. When connected the USB module should appear as a standard virtual COM port (VCP) serial device

Windows: COMx where x is the COM port number.

Linux: This may vary depending on your distribution but will normally appear as /dev/tty/USBx where x is the device number.

MAC OSX: /dev/cu.usbserial-xxxxxxxx or /dev/tty.usbserial-xxxxxxxx where xxxxxxxx is the device's serial number.

Should you experience issues with driver support or accessing the serial interface please see our support forum (forum.hobbycomponents.com) for more information and links to drivers. Alternatively you can download the latest driver directly from the FTDI Chip website here:

https://www.ftdichip.com/FTDrivers.htm

Note: Please ensure when downloading drivers from the FTDI chip website that you select the VCP version of the driver.

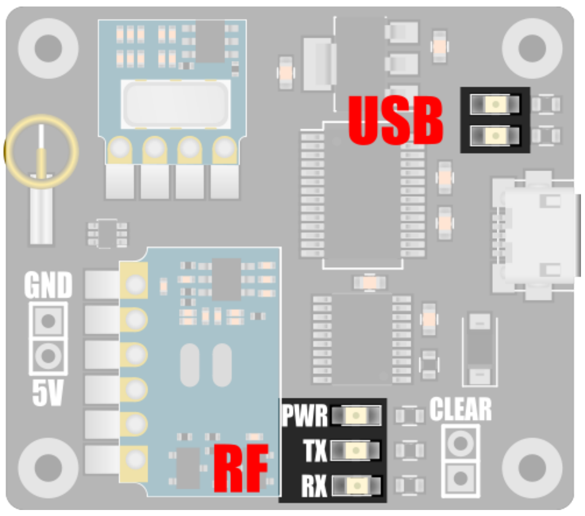
# USB Module Features

## Power

Power is supplied from the computer via the USB connection and so no external power supply is required.

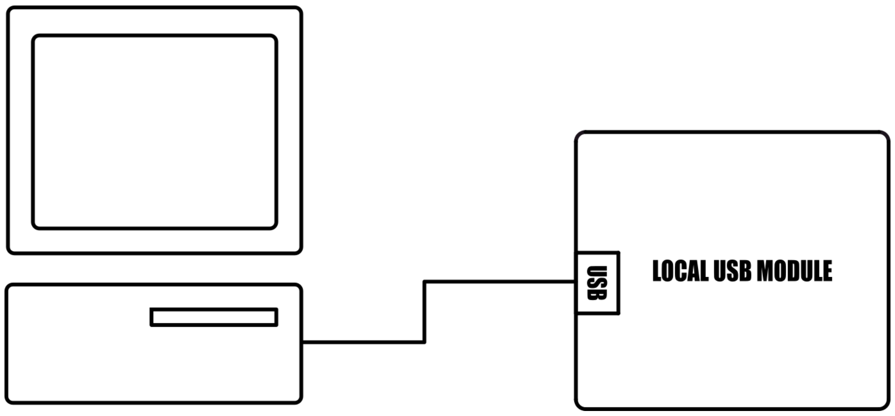
## LEDs

The sensor module includes 5 LEDs which indicate the current state of the module.



| | |
|---|---|
| PWR LED (green) | Indicates the module is currently powered. |
| RF TX LED (red) | Indicates the module is currently transmitting data to other SmartRFy modules. |
| RF RX LED (amber) | Indicates the module is currently receiving data from other SmartRFy modules regardless of zone or address. |
| USB TX (red) | Indicates the module is currently sending serial data to the PC |
| USB RX (amber) | Indicates the module is currently receiving serial data from the PC |

## USB Serial Interface



When connected to a host PC the module will appear as a standard virtual COM port (VCP). Using suitable software such as a serial terminal program on the host PC the USB module can be controlled using simple text based commands.

### Command interface feature

The USB serial interface can be used for issuing local text based commands and settings directly to the module via the connected PC (local commands). The interface can also be used for sending and receiving text based commands to other remote SmartRFy devices (remote commands).

### Wireless serial port feature

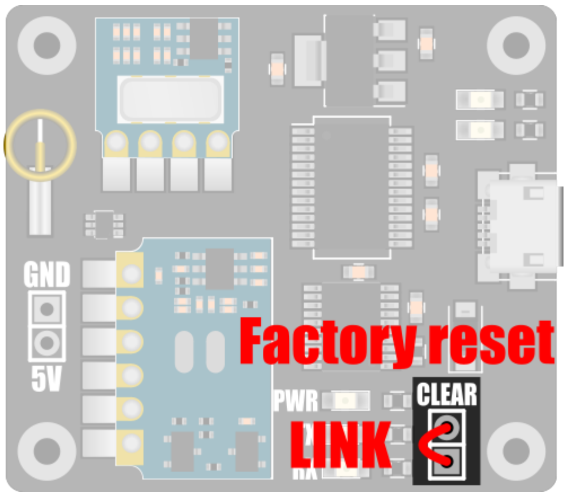
Additionally, the serial interface can also act as a passive wireless serial port for transmitting non-SmartRFy related data to other devices connected to remote SmartRFy modules. Note that when sending passive serial data the USB modules serial buffer is limited to 25 bytes and maximum data transfer rate to a remote device is limited by the RF transfer rate and not the actual serial connection speed from the computer to the USB module.

### Default configuration settings

By default the USB serial interface is configured to 9600 baud, 8 data bits, no parity, and 1 stop bit. See the USB serial interface control section for more information on setting the baud rate, verbose mode, and issuing commands.

Note: To connect the module to a PC a suitable micro USB cable will be required (see Hobby Components SKU: HCCABL0012).

## Clear (Factory Reset)



The module is capable of storing a number of user configuration settings in its non-volatile memory. This allows it to retain these settings even after power is removed. The two pads marked 'CLEAR' provide a means of restoring the module back to its original factory default settings. Should you wish to reset the module, connect power to the module whilst shorting the two pads together.

Alternatively the module can be restored back to factory default settings by issuing the factory reset (FTRE) command via the serial interface – see the SmartRFy commands section for more information.

# SmartRFy commands

## Command formats

SmartRFy modules are capable of responding to an array of text based commands sent via the USB serial interface. These commands can be either module configuration settings or commands to control the locally connected module itself, or commands to be transmitted to a remote module.

### Command format requirements

In all cases the command must always be terminated with a carriage return (ASCII code 13) and a line feed (ASCII code 10).

The relay modules do not have transmit capability. Therefore commands can only be processed by the module itself (local commands) and cannot be transmitted to other modules (remote commands). To tell the module that you are issuing a local command, the command must be prefixed with an asterisk (*). For example *SW0=1 will cause relay 0 to be energised. Remote commands, i.e. commands prefixed with a greater than symbol (>) will be ignored.

Commands must always be in uppercase and no part of the command should include white space characters, including spaces.

Some commands require more than one parameter. In this case parameters should be separated by commas (,) with no spaces.

Commands are always 4 characters in length. Commands that require one or more parameters will have an equals (=) as the last character (e.g. XXX=), and commands that are querying information will have a question mark (?) as the last character (e.g. XXX?).

The device will respond to a successful local command with the response ‘OK’. An unsuccessful command will respond with an ‘ERROR’. There is no response by the local device when issuing remote commands. All responses are terminated with a carriage return and line feed.

No serial data sent to the local device, command or otherwise, may exceed 25 characters in length (including carriage return / linefeed termination).

## Local command format

| Prefix | Command | Optional parameter | Optional parameter | Optional parameter | Carriage return | Line feed |
|---|---|---|---|---|---|---|
| * | XXXX | <parameter 1> | ,<parameter 2> | ,<parameter n> | 0x0D (\r) | 0x0A (\n) |

Example: *Setting and confirming the zone of the local device*

```
*ZON=1<CR><LF>
```

Sets the zone of the connected module to 1.

Response:

```
OK<CR><LF>
```

```
*ZON?<CR><LF>
```

Requests the current zone of the local device.

Response:

```
1<CR><LF>
```

## Command quick lookup table

## SmartRFy generic commands – quick lookup table

| Command | Description | Parameter(s) | Response |
|---|---|---|---|
| FWV? | Get firmware version | None | Vx.x<br>(C) HobbyComponents.com |
| ZON? | Get zone number (default: 0) | None | Zone number (0 to 255) |
| ADD? | Get address (default: 0) | None | Address (0 to 255) |
| STO=*Time* | Sets the timeout time for the serial interface (default: 2000ms) | *Time*: Timeout time in milliseconds (0 to 65535) | OK |
| BUR=*Baud* | Sets the BAUD rate for the serial interface | Baud: 0 = 1200 BAUD<br>Baud: 1 = 2400 BAUD<br>Baud: 2 = 4800 BAUD<br>Baud: 3 = 9600 BAUD (default)<br>Baud: 4 = 19200 BAUD<br>Baud: 5 = 38400 BAUD<br>Baud: 6 = 57600 BAUD<br>Baud: 7 = 115200 BAUD | OK |
| TRS=*Resends* | Sets the number of times any Tx data will be retransmitted | *Resends*: Number of resends (0 to 5) | OK |
| VBM=*Mode* | Sets the verbose mode which specifies what received data is output to the serial interface (default: 0) | Mode: 0 = Both passive serial & commands from all zones and addresses<br>Mode: 1 = Only passive serial data from all zones and addresses<br>Mode: 2 = Only control from all zones and addresses<br>Mode: 3 = Both passive serial & commands with matching zone and address<br>Mode: 4 = Only passive serial data with matching zone and address<br>Mode: 5 = Only commands with matching Zone and address | OK |
| ZON=*Zone* | Sets the devices zone number | *Zone*: Zone number (0 to 255) | OK |
| ADD=Address | Sets the devices address | *Address*: Devices address (0 to 255) | OK |
| TRP=*State* | Turns on or off repeater mode (default: off) | *State*: Repeater state (0 = off, 1 = on) Note: Only available on modules with both RF Tx & Rx capability | OK |
| FTRE | Performs a factory reset – all settings a restored to their default values | None | OK |

## Digital USB module specific commands – quick lookup table

| Command | Description | Parameter(s) | Response |
|---|---|---|---|
| SWM=*MerAdd, MerChan, MerState* | Sets the ON/OFF state of a Mercury energy saving socket | *MerAdd*: The address of the Mercury socket (0 to 31)<br><br>*MerChan*: The channel number of the Mercury socket (0 to 5)<br><br>MerState: The On/Off state of the socket (0 = off, 1 = on) | OK |
| SWx=*State* | Sets the state Mercury energy saving socket x | *x*: The energy saving socket number (0 to 3). E.g. SW0 is socket 0<br><br>*State*: Floating point value which determines the state of the digital output pin depending on TxH, TxL, MxH, & MxL settings. Default: 0 = socket off, 1 = socket on | OK |
| SxN=*Name* | Sets an alternative three letter command name the Mercury socket will respond to (default SWx) | *x*: The energy saving socket number (0 to 3). E.g. S0N is socket 0<br><br>*Name*: Three letter name of the alternate command | OK |
| SxO=*Time* | Sets the amount of time the Mercury socket will remain on when triggered. The socket will automatically turn off when time has elapsed | *x*: The energy saving socket number (0 to 3). E.g. S0O is socket 0<br><br>*Time*: Mercury socket on time in seconds (0 to 65525). Default = 0 (disabled). | OK |
| TxH=*Threshold* | Sets the threshold value required to turn the Mercury socket on (default: 1) | *x*: The energy saving socket number (0 to 3). E.g. T0H is socket 0<br><br>*Threshold*: A decimal number which sets the level required turn the socket on | OK |
| TxL=*Threshold* | Sets the threshold value required to turn the Mercury socket off (default: 1) | *x*: The energy saving socket number (0 to 3). E.g. T0L is socket 0<br><br>*Threshold*: A decimal number which sets the level required turn the socket off | OK |
| MxH=*Mode* | Sets the test condition under which the Mercury socket will be turned on (default: =) | *x*: The energy saving socket number (0 to 3). E.g. M0H is socket 0<br><br>*Mode*: > The value passed by the SWx= command must be greater than the threshold level set by the TxH= command<br><br>*Mode*: = The value passed by the SWx= command must equal the threshold level set by the TxH= command<br><br>*Mode*: < The value passed by the SWx= command must be less than the threshold level set by the TxH= command | OK |
| MxL=*Mode* | Sets the test condition under which the Mercury socket will be turned off (default: =) | x: The energy saving socket number (0 to 3). E.g. M0L is socket 0<br><br>Mode: > The value passed by the SWx= command must be greater than the threshold level set by the TxH= command<br><br>Mode: = The value passed by the SWx= command must equal the threshold level set by the TxH= command<br><br>Mode: < The value passed by the SWx= command must be less than the threshold level set by the TxH= command | OK |
| SxA=*Add* | Sets the Mercury socket address assigned to SWx | *x*: The energy saving socket number (0 to 3). E.g. SxA is socket 0<br><br>*Add*: The address of the Mercury socket (0 to 31) | OK |
| SxC=*Chan* | Sets the Mercury socket address assigned to SWx | *x*: The energy saving socket number (0 to 3). E.g. SxC is socket 0<br><br>*Chan*: The channel number of the Mercury socket (0 to 5) | OK |

## Generic SmartRFy commands

Commands listed below are common to all SmartRFy modules, including the digital USB module. Note that all commands must be proceeded with a carriage return and line feed, but for clarity purposes this is omitted from the examples in this manual.

### Firmware version (FWV?)

Gets the firmware version of the module.

Example: *FWV?

Returns:
```
Vx.x
(C) HobbyComponents.com
```

Where Vx.x is the version number

### Get the modules zone (ZON?)

Gets the modules current zone number

Example: *ZON?

Returns: The current zone number as a decimal value (0 to 255)

### Get the modules address (ADD?)

Gets the modules current address

Example: *ADD?

Returns: The current address as a decimal value (0 to 255)

### Serial serial timeout time (STO=)

Sets the timeout time for the serial interface in milliseconds. Maximum timeout time is 65535ms (65.535 seconds). If a complete command (including CR + LF termination) is not received within this time the command will be ignored and an 'ERROR' will be returned.

Default: 2000 (2 seconds)

Example: *STO=1000

The above example will set the serial port timeout time to 1000ms (1 second).

Returns: OK

## Set the baud rate (BUR=)

Sets the communication baud rate for the serial UART interface. There are 8 (0 to 7) possible settings for the baud rate:

0 = 1200 BAUD
1 = 2400 BAUD
2 = 4800 BAUD
3 = 9600 BAUD (default)
4 = 19200 BAUD
5 = 38400 BAUD
6 = 57600 BAUD
7 = 115200 BAUD

NOTE: This command sets the serial communication speed between the connected device and the SmartRFy module – it does not set the wireless RF communication speed between modules.

Default: 3

Example: *BUR=5

The above example will set the serial port baud rate to 38400 baud.

Returns: OK

## Set number of transmit resends (TRS=)

Sets the number of times the module will automatically re-transmit data. Increasing the number of times data is retransmitted will reduce the chances of the data not being received due to interference but will also increase transmission times and network traffic. The number of resends can be set from 0 (no resends) to 5.

| | |
|---|---|
| Default: | 2 |
| Example: | *TRS=3 |

The above example will set the number of resends to 3. Therefore any data wirelessly transmitted by the module will be transmitted a total of 4 times.

Returns: OK

## Set the serial verbose mode (VBM=)

Sets what type of data received by the module is automatically output to its serial port.
There are 6 possible modes (0 to 5):

0 = Both passive serial and commands from all zones and addresses
1 = Only passive serial data from all zones and addresses
2 = Only control from all zones and addresses
3 = Both passive serial and commands with matching zone and address
4 = Only passive serial data with matching zone and address
5 = Only commands with matching zone and address

| | |
|---|---|
| Default: | 0 |
| Example: | *VBM=5 |

The above example will set the module to only output received commands that have been transmitted from a module with the same zone and address as itself.

Returns: OK

## Set the modules zone number (ZON=)

Sets the modules zone number. The zone number can be between 0 and 255.

Note that zone 255 is a special zone number. Setting the module to this zone number means that it will treat received data from any zone as if it were in the same zone as itself. Therefore this zone number can be used when you wish the module to respond to commands from modules in multiple zones. Additionally, when transmitting data with a zone of 255 all modules within range will respond to the data as if it is in the same zone as itself. Therefore this zone number can also be used when you require the module to control multiple remote modules in different zones.

| | |
|---|---|
| Default: | 0 |
| Example: | *ZON=2 |

The above example will set the modules zone number to 2.

Returns: OK

## Set the modules address (ADD=)

Sets the module's address. The address can be between 0 and 255.

Note that address 255 is a special address. Setting the module to this address means that it will treat received data from any address as if it were the same address as itself. Therefore, this address can be used when you wish the module to respond to commands from modules with different addresses. Additionally, when transmitting data with an address of 255 all modules within range will respond to the data as if it has the same address as itself. Therefore this address can also be used when you require the module to control multiple remote modules with different addresses.

| | |
|---|---|
| Default: | 0 |
| Example: | *ADD=5 |

The above example will set the modules address to 5.

Returns: OK

## Turn on/off repeater mode (TRP=)

Turns repeater mode on or off. When turned on the module will retransmit any data it receives regardless of its zone and address. This mode can be used when a remote module is outside the range of a module that needs to communicate with it, but the repeating module is within range of both modules. Setting the repeater mode to 1 will turn on repeater mode and setting it to 0 will turn it off.

Note:

> This mode is only supported by SmartRFy modules with both RF receive and transmit capabilities.
>
> Turning on the repeater mode will double network traffic for any modules within range.

| | |
|---|---|
| Default: | 0 (off) |
| Example: | *TRP=1 |

The above example will turn on repeater mode.

Returns: OK

## Factory reset (FTRE)

Performs a factory reset of the module. All module settings will be restored to their factory defaults.

Example: *FTRE

The above example will restore all settings to their defaults.

Returns: OK

## SmartRFy USB module commands

Commands below are specific to the SmartRFy digital USB module and can be used to configure parameters relating to the various hardware features of this module.

### Set the state of a Mercury socket (SWM=)

Sets the on/off state of a Mercury socket by specifying its address and channel number. Note that Mercury sockets use their own addressing system which is separate to SmartRFy module addresses. See the Mercury socket section for more information.

The command is formatted in the form of SWM=*add*,*chan*,*state* where *add* is the Mercury address of the socket (0 to 31), *chan* is the sockets channel number (0 to 5) and *state* is the required on / off state.

**Examples:**

*SWM=1,2,1

The above example will turn on a Mercury socket with an address of 1 and channel of 2.

*SWM=1,2,0

The above example will turn off a Mercury socket with a Mercury address of 1 and channel of 2.

### Set the state of a Mercury socket using a preset command (SWx=)

The USB module has 4 presets (SW0, SW1, SW2, & SW3) which can be assigned to any Mercury socket address and channel. Once a socket is assigned to one of the SWx commands it can be turned on or off by simply issuing the appropriate SWx command.

**Examples:**

Assign a Mercury socket with an address of 1 and channel of 2 to command SW0

*S0A=1 Command SW0 is now assigned to a Mercury socket with an address of 1

*S0C=2 Command SW0 is now assigned to a Mercury socket with a channel of 2

Mercury socket with address 1 and channel 2 can now be turned on with preset command...

*SW0=1

...and turned off with command

*SW0=0

**Returns:** OK

## Set an alternative preset name (SxN=)

Sets an alternative 3 letter command name for one of the 4 preset switch commands (SW0, SW1, SW2, SW3). Setting an alternate command name allows a socket that is assigned to one of the 4 presets to be controlled by other devices such as temperature sensors, PIR's, switches etc. (Also see pin commands SxO, TxH, TxL, MxH, MxL, and Appendix A & B for more options and examples of configuring the preset switch commands).

To specify which preset the command is intended for substitute x for the preset number number (0 to 3). For example S0N= for preset switch SW0, S1N= for preset switch SW1, etc.

| | |
|---|---|
| Default: | SW0 for preset 0 |
| | SW1 for preset 1 |
| | SW2 for preset 2 |
| | SW3 for preset 3 |

Example: *S0N=ABC

The mercury socket assigned to preset 0 will now respond to a command in the format of ABC=xxxx

Returns: OK

## Set a preset on time (SxO=)

Sets the amount of time in seconds that a socket assigned to one of the 4 presets (SW0, SW1, SW2, or SW3) will remain on after receiving a trigger command. Setting the on time to 0 (default) will disable the timer (for that particular preset) and the socket will not turn back off unless it receives a command to do so. Setting the on time to anything between 1 and 65535 will cause the socket to remain on until that amount of time, in seconds, has elapsed. To specify which preset the command is intended for substitute x for the preset number (0 to 3). For example S0O= for pin preset SW0, S1O= for preset SW1, etc.

| | |
|---|---|
| Default: | 0 (disabled) |
| Example: | *S0O=60 |

In the above example the Mercury socket assigned to preset 0 (SW0) will automatically turn off after 60 seconds from receiving a trigger command.

Returns: OK

## Set the on threshold value for one of the presets (TxH=)

Sets the threshold value which will be used to determine if a socket assigned to one of the 4 presets (SW0, SW1, SW2, or SW3) should be turned on when a SWx= command is received. The on threshold can be a positive or negative decimal number. The module will compare this value with the value received from an SWx command and together with the on mode command (MxH) will determine if the socket should be turned on. (Also see preset commands SxO, TxH, TxL, MxH, MxL, and Appendix A & B for more options and examples of configuring the preset switch commands). To specify which preset the command is intended for substitute x for the preset number (0 to 3). For example T0H= for preset SW0, T1H= for preset SW1, etc.

|  |  |
|---|---|
| Default: | 1 |
| Example: | *T0H=100 |

In the above example a Mercury socket assigned to preset SW0 will be turned on if the module receives the preset command SW0=100 (assuming M0H is set to default).

Returns: OK

## Set the off threshold value for one of the presets (TxL=)

Sets the threshold value which will be used to determine if a socket is assigned to one of the 4 presets (SW0, SW1, SW2, or SW3) should be turned off when a SWx= command is received. The off threshold can be a positive or negative decimal number. The module will compare this value with the value received from an SWx command and together with the off mode command (MxL) will determine if the socket should be turned off. (Also see preset commands SxO, TxL, TxL, MxH, MxL, and Appendix A & B for more options and examples of configuring the preset switch commands). To specify a preset the command is intended to substitute x for the preset number (0 to 3). For example T0L= for preset SW0, T1L= for preset SW1, etc.

|  |  |
|---|---|
| Default: | 0 |
| Example: | *T0L=-50 |

In the above example a Mercury socket assigned to preset SW0 will be turned off if the module receives the preset command SW0=-50 (assuming M0L is set to default).

Returns: OK

## Set the on test condition for one of the presets (MxH=)

Sets the threshold on condition used to determine if a Mercury socket assigned to one of the 4 presets (SW0, SW1, SW2, or SW3) should be turned on. When a preset (SWx) is received its parameter value is tested against the appropriate on threshold high (TxH) value, and this threshold condition determines if the socket should be turned on. To specify which preset the command is intended for substitute x for the preset number (0 to

3). For example M0H= for preset SW0, M1H= for preset SW1, etc. The MxH threshold condition can be set to one of three values:

MxH=< The socket will be tuned on if the value received by the SWx command is *less than* the preset threshold high (TxH) parameter.

MxH== The socket will be tuned on if the value received by the SWx command is *equal to* the preset threshold high (TxH) parameter.

MxH=> The socket will be turned on if the value received by the SWx command is *greater than* the socket threshold high (TxH) parameter.

| | |
|---|---|
| Default: | = |
| Example: | *M0H=> |

In the above example the Mercury socket assigned to preset 0 will be turned on if the module receives the preset switch command SW0 with a value greater than the current pin threshold high (T0H) level.

Returns: OK

## Set the off test condition for one of the presets (MxL=)

Sets the threshold off condition used to determine if a Mercury socket assigned to one of the 4 presets (SW0, SW1, SW2, or SW3) should be turned off. When a preset (SWx) is received its parameter value is tested against the appropriate on threshold low (TxL) value, and this threshold condition determines if the socket should be turned off. To specify which preset the command is intended for substitute x for the preset number (0 to 3). For example M0L= for preset SW0, M1L= for preset SW1, etc. The MxL threshold condition can be set to one of three values:

MxL=< The socket will be turned off if the value received by the SWx command is *less than* the preset threshold low (TxH) parameter.

MxL== The socket will be turned off if the value received by the SWx command is *equal to* the preset threshold low (TxH) parameter.

MxL=> The socket will be turned off if the value received by the SWx command is *greater than* the socket threshold low (TxH) parameter.

| | |
|---|---|
| Default: | = |
| Example: | *M0H=< |

In the above example the Mercury socket assigned to preset 0 will be turned off if the module receives the preset switch command SW0 with a value less than the current pin threshold low (T0L) level.

Returns: OK

## Set a Mercury socket preset address (SxA=)

Sets a Mercury socket address (0 to 31) for one of the 4 presets (SW0, SW1, SW2, or SW3). Note that Mercury sockets use their own addressing system which is separate to SmartRFy module addresses. See the Mercury socket section for more information. When a mercury socket address is assigned to one of the 4 presets, sending an on/off trigger to that preset will turn on or off a Mercury socket with that address (and matching channel number – see command SxC). To specify which preset the command is intended for substitute x for the preset number (0 to 3). For example S0A= for preset SW0, S1A= for preset SW1, etc.

| | |
|---|---|
| Default: | 0 |
| Example: | *S0A=5 |

In the above example sending a preset switch command SW0=1 will turn on a Mercury socket with a socket address of 5 and will turn off the socket with a switch command SW0=0. Note that the socket channel number must also match – see preset channel command SxC.

## Set a Mercury socket preset channel number (SxC=)

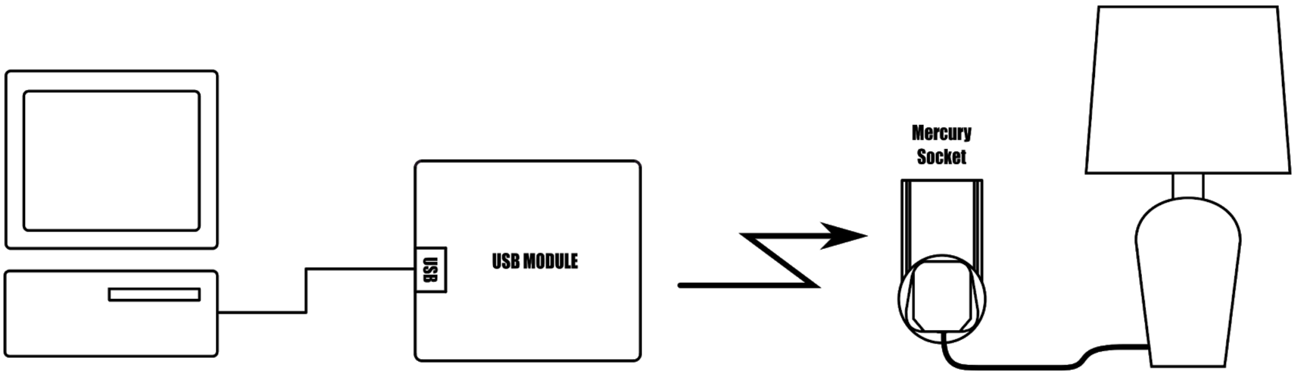
Sets a Mercury socket channel number (0 to 5) for one of the 4 presets (SW0, SW1, SW2, or SW3). When a mercury channel number is assigned to one of the 4 presets, sending an on/off trigger to that preset will turn on or off a Mercury socket with that channel number (and matching socket address – see command SxA). To specify which preset the command is intended for substitute x for the preset number (0 to 3). For example S0C= for preset SW0, S1C= for preset SW1, etc.

| | |
|---|---|
| Default: | 0 |
| Example: | *S0C=2 |

In the above example sending a preset switch command SW0=1 will turn on a Mercury socket with a channel number of 2 and will turn off the socket with a switch command SW0=0. Note that the socket address must also match – see preset socket address command SxA.

## Mercury sockets

The SmartRFy USB module has the additional feature of being able to directly control Mercury branded energy saving mains socket adapters (see Hobby Components SKU: HCPOWE0010). These are remotely controlled adapters that sit between your electrical appliance and the mains socket and allow mains power to the appliance to be remotely switched using an RF remote control.



Mercury sockets are addressable and each socket can be programmed with an address between 0 and 31 (note the use of an addressing system that is separate to the one used by SmartRFy modules). Additionally each addressed socket can be assigned a channel number between 0 and 5. This means that a total of 192 sockets can be individually controlled by the SmartRFy USB module.

### Controlling a socket by address and channel

The state of a socket can be controlled by sending the SWM= command followed by the socket's address, channel number and required state to the USB module via the PC. For example to turn on a socket with an address of 5 and channel of 2 you would send the following command:

```
*SWM=5,2,1
Response: OK
```

To turn the socket back off send the following command:

```
*SWM=5,2,0
Response: OK
```

## Controlling a socket using preset commands

The USB module has 4 preset commands (SW0, SW1, SW2, & SW3) each of which can be assigned to an individual Mercury socket. Once a socket is assigned to one of these presets it can then be controlled via other remote SmartRFy modules such as the digital Tx module. The 4 preset commands can also be configured to work in toggle, timer, and threshold modes (see appendix A & B).

By default the 4 presets are assigned to Mercury socket address 0 and channel numbers 0 to 3 respectively. To assign one of the presets to a Mercury socket with a different address & channel number you must program the preset with the address and channel number of the required socket. For example to assign a socket with an address of 5 and channel of 2 to preset 0 (SW0):

```
*S0A=5        Assigns a Mercury socket with an address of 5 to preset SW0
Response: OK
*S0C=2        Assigns a Mercury socket with a channel number of 2 to preset SW0
Response: OK
```

Once the above commands have been issued the socket with address 5 and channel 2 can be turned on with the following preset command:

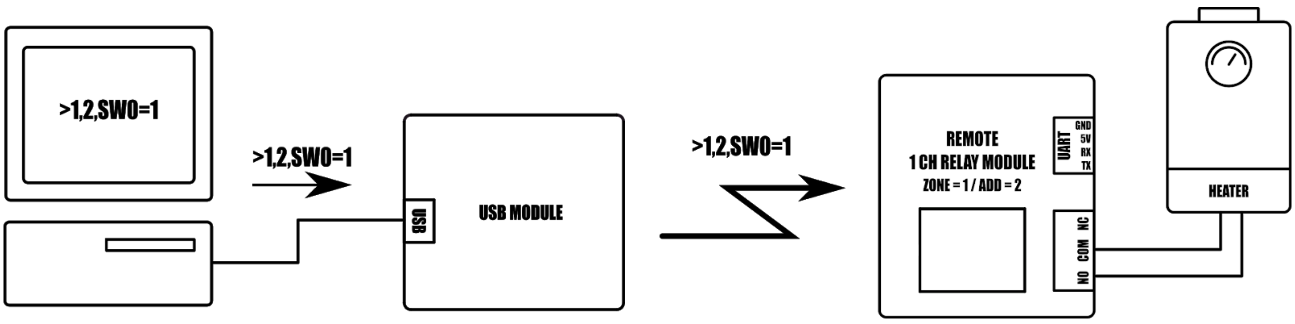
```
*SW0=1
```

To turn the socket back off send the following command:

```
*SW0=0
```

# Appendix A: Examples

## Controlling a remote SmartRFy module



This example demonstrates how to control remote SmartRFy devices by using the example of controlling a central heating boiler via a remote SmartRFy relay module with a zone of 1 and address of 2. All other settings within the relay and USB modules are assumed to be at their defaults.

1) From the computer the remote switch command is sent to the USB module (terminated with a CR & LF):

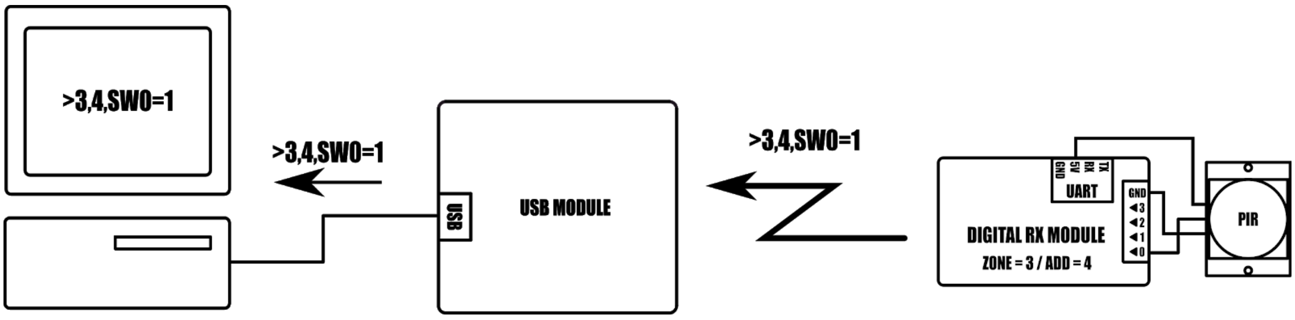
```
>1,2,SW0=1
```

2) The USB module transmits this command to all SmartRFy modules within range.

3) The relay module receives the command and as the commands zone and address match its zone and address, it processes the switch command.

4) The relay module interprets the command SW0=1 as ‘turn relay channel 0’ on and therefore energises its relay turning the boiler on.

## Monitoring a Remote SmartRFy module



This example demonstrates receiving commands from a remote SmartRFy module. In the example a PIR sensor is connected to digital input 0 of a remote digital Tx module with a zone of 3 and address of 4.

1) When the PIR is triggered the remote digital Tx module will transmit the following command:

>3,4,SW0=1

2) The USB module will receive this command and regardless of whether the commands zone and address match that of itself, the USB module will output the command to the computer via the USB interface (terminated with a CR & LF).

3) The command >3,4,SW0=1 will then be displayed on the terminal software running on the computer (or can be processed by whatever automation software is monitoring the serial interface) which can be interpreted as digital pin 0 on a remote module with a zone of 3 and address of 4 has been pulled high.

Note:

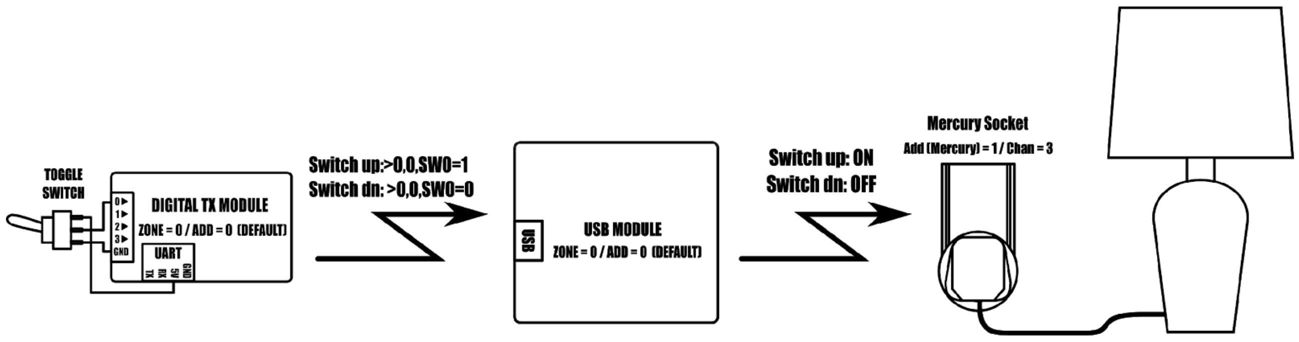
When the PIR returns to its un-triggered state a second command (SW0=0) will be transmitted by the digital Tx module to reflect this new state.

By default the USB module will output to the computer any command it receives regardless of its zone or address. This behaviour can be changed using the verbose command VBM. See SmartRFy generic commands section for more information.

## Appendix B: Mercury preset modes

When using the USB module to control Mercury energy saving sockets the 4 preset functions can be used to perform some basic automation functions without the need of a host computer. In these examples the USB module instead acts as a hub to allow other remote SmartRFy modules to control the Mercury sockets.

### Default preset mode



°

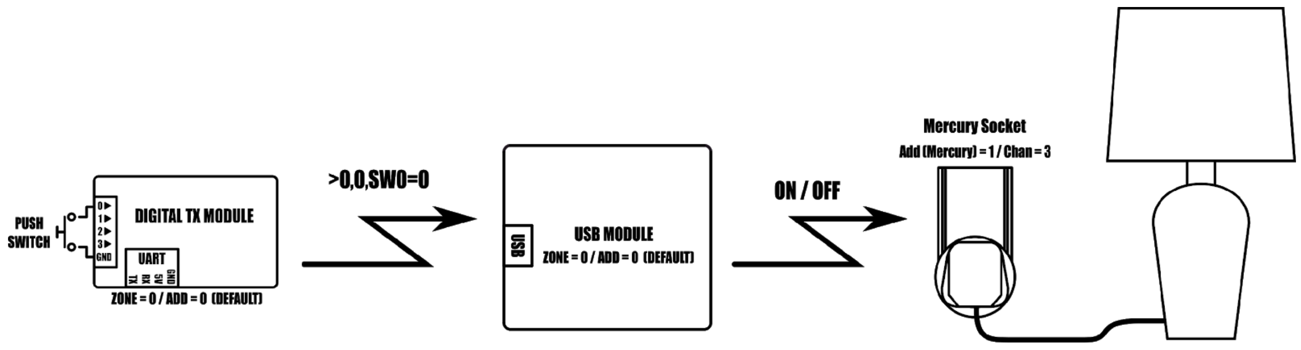
This example demonstrates the use of one of the 4 the Mercury socket preset commands to allow a toggle switch connected to a remote digital Tx module to control a lamp plugged into a Mercury socket. Note that the zone and address of the digital Tx module must match that of the USB module. In this example both zone and addresses are left at their default (zone=0, address=0).

On the USB module configure preset SW0 to control a Mercury socket with a mercury address of 1 and channel of 3:

| | |
|---|---|
| *S0A=1 | Preset SW0 will now control a Mercury socket with an address of 1 |
| *S0C=3 | Preset SW0 will now control a Mercury socket with a channel number of 3 |

With all other settings in both the digital Tx and USB modules left at their defaults the lamp will turn on when the toggle switch is in the up position and off when it is in the down position.

## Toggle mode



This example demonstrates the preset toggle mode feature. This feature allows the state of a Mercury socket to be toggled whenever an appropriate trigger command is received. In this example a lamp plugged into a Mercury socket can be toggled on and off by pressing a push switch connected to a remote digital Tx module. Note that the zone and address of the digital Tx module must match that of the USB module. In this example both zone and addresses are left at their default (zone=0, address=0). Each time the push button is pressed it will pull digital input 0 on the Tx module low and the module will transmit a SW0=0 command. The USB module is configured to toggle the state of preset 0 when it receives SW0=0 command:

On the USB module configure preset SW0 to control a Mercury socket with a mercury address of 1 and channel of 3:

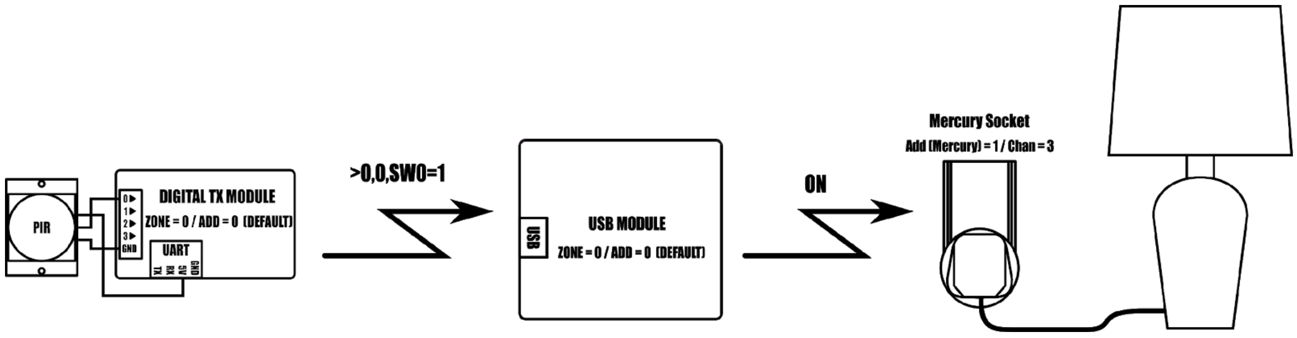
| | |
|---|---|
| *S0A=1 | Preset SW0 will now control a Mercury socket with an address of 1 |
| *S0C=3 | Preset SW0 will now control a Mercury socket with a channel number of 3 |

Next configure preset SW0 to toggle its state whenever it receives the SW0=0 command by setting its low and high thresholds to the same value (0):

| | |
|---|---|
| *T0H=0 | Preset SW0 threshold high is now set to 0 |
| *T0L=0 | Preset SW0 threshold low is now set to 0 |

With all other settings in both the digital Tx and USB modules left at their defaults the lamp will now turn on or off whenever the push button on the remote digital Tx module is pressed.

## Timer mode



This example demonstrates the preset timer mode feature. This feature allows a Mercury socket to be turned on for a fixed amount of time whenever an appropriate trigger command is received. In this example A lamp plugged into a Mercury socket will turn on for 60 seconds whenever a PIR connected to a remote digital Tx module is triggered. Note that the zone and address of the digital Tx module must match that of the USB module. In this example both zone and addresses are left at their default (zone=0, address=0).

On the USB module configure preset SW0 to control a Mercury socket with a mercury address of 1 and channel of 3:

| | |
|---|---|
| *S0A=1 | Preset SW0 will now control a Mercury socket with an address of 1 |
| *S0C=3 | Preset SW0 will now control a Mercury socket with a channel number of 3 |

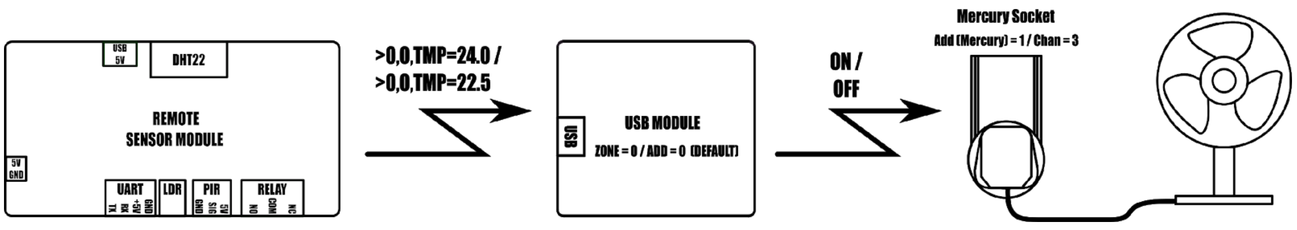
Next configure the USB module to enable timer mode for preset SW0:

| | |
|---|---|
| *S0O=60 | The on timer for preset SW0 is now set to 60 seconds |

With all other settings in both the digital Tx and USB modules left at their defaults the lamp will now turn for 60 seconds each time the PIR connected to the remote digital Tx is triggered. If at any point during the 60 second period the PIR is re-triggered the timer will be reset.

Note: To disable timer mode set the on time to 0 seconds (S0O=0).

## Threshold mode



This example demonstrates the threshold mode feature. Threshold mode allows a Mercury socket to be turned on or off whenever the value passed by the trigger command goes above or below a predefined value. In this example a desk fan plugged into a Mercury socket will turn on whenever the temperature sensed by the remote sensor module goes above 24oC and will turn off when it goes below 22.5oC. Note that the zone and address of the sensor module must match that of the USB module. In this example both zone and addresses are left at their default (zone=0, address=0).

On the USB module configure preset SW0 to control a Mercury socket with a mercury address of 1 and channel of 3:

| | |
|---|---|
| *S0A=1 | Preset SW0 will now control a Mercury socket with an address of 1 |
| *S0C=3 | Preset SW0 will now control a Mercury socket with a channel number of 3 |

Next configure the USB modules mode and threshold levels for preset SW0:

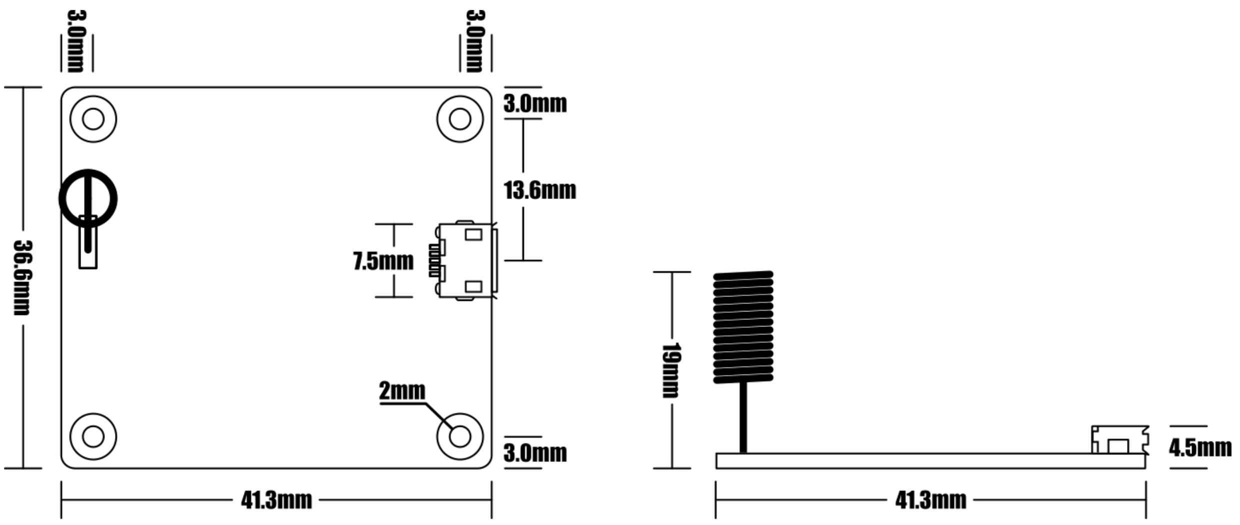
| | |
|---|---|
| *M0H=> | Preset SW0 will now turn on whenever it receives a value above T0H level |
| *M0L=< | Preset SW0 will now turn off whenever it receives a value below T0L level |
| *T0H=24.0 | Preset SW0 will now turn on whenever it receives a value above 24.0 |
| *T0H=22.5 | Preset SW0 will now turn off whenever it receives a value below 22.5 |

Finally configure the USB modules preset SW0 to listen to the TMP command:

| | |
|---|---|
| *S0N=TMP | Preset SW0 will now respond to TMP commands |

With all other settings in both the sensor and USB modules left at their defaults the fan will now turn on when the temperature sensed by the remote sensor module goes above 25.0oC and will turn off when it goes below 22.5oC.

## Dimensions


3.0mm
3.0mm
3.0mm
13.6mm
7.5mm
36.6mm
2mm
3.0mm
41.3mm
19mm
4.5mm
41.3mm