

## Standard RS232 -> 1Wire Gateway

### Features

#### ASCII Commands/Reports ...

Only ASCII commands are necessary to communicate with and retrieve data from the module.

### Installation

The module attaches to your computer or controller via a serial COM port. Locate an unused COM port and attach the module to the male DB9 connector. Connect the sensors to the module using standard Cat5 cables.

Note: while the same cable type is used as for Ethernet, do not plug your sensors or the module into your Ethernet LAN. It won't work.

The module steals power from the serial port (COM port). The amount of power the serial port supplies may limit the number of sensors you can deploy.

### ASCII Control Functions:

The device provides many 1-Wire bus functions in response to ASCII commands. This allows it to be used with any terminal program or serial port communications API to perform iButton or 1-Wire functions without software drivers. The functions that can be performed have been coded for normal ASCII keys so that all the basic 1-Wire functions can be performed manually on the keyboard of a terminal program, or using programming languages that are limited to ASCII serial port I/O.

The supported ASCII functions are letters and symbols as follows:

Key **h** – Displays a list of ASCII commands that are available.

**Example:** (output from the h command, each line is terminated with <CR><LF>)

<b>f</b>	=First	<b>\</b>	=Expanded Command Set
<b>n</b>	=Next	<b>!</b>	=Thru
<b>r</b>	=Reset	<b>l</b>	=Level
<b>(</b>	=Extend	<b>:</b>	=115200
<b>b</b>	=Byt(NN+)	<b>^</b>	=57600
<b>j</b>	=Bit(N+)	<b>`</b>	=38400
<b>p</b>	=BytW/P	<b>,</b>	=19200
<b>~</b>	=BitW/P	<b>x</b>	=BusLo
<b>d</b>	=Aux+	<b>\$</b>	=Scan
<b>z</b>	=Aux-	<b>*</b>	=List
<b>&amp;</b>	=Aux?	<b>h</b>	=Help
<b>t</b>	=Search Type		

**Key r** – Performs a bus reset and returns the status of the bus.

The **r** returns either a “P” representing the presence of one or more devices on the bus, an “N” representing no devices on the bus, or an “S” representing a shorted bus.

Example: P<CR><LF>

**Key b** – Places the device into byte mode.

The next two characters entered will be taken as a hexadecimal byte value, which is then issued onto the bus. The response byte is then displayed in hexadecimal. Subsequent pairs of hexadecimal characters will also generate bytes, allowing for streaming of bytes without intervening commands. Hitting ENTER (Carriage Return) will end the Byte mode.

The following subcommands are recognized in byte mode instead of a 2 character hex pair:

**P**... Issue 64 byte reads to the OW bus and return results. This subcommand is used to read a memory page from devices that have a page size of 64 bytes. Returns 128 characters followed by <CR><LF>

**p** ... Issue 66 byte reads to the OW bus and return results. This subcommand is used to read a memory page WITH CRC from devices that have a page size of 64 bytes. Returns 132 characters followed by <CR><LF>

**M** ... Issue 32 byte reads to the OW bus and return results. This subcommand is used to read a memory page from devices that have a page size of 32 bytes. Returns 64 characters followed by <CR><LF>

**m** ... Issue 34 byte reads to the OW bus and return results. This subcommand is used to read a memory page WITH CRC from devices that have a page size of 32 bytes. Returns 68 characters followed by <CR><LF>

**Key p** – Places the device into byte mode with strong pullup.

The next two characters entered will be taken as a hexadecimal byte value, which is then issued onto the bus. The response byte is then displayed in hexadecimal. Subsequent pairs of hexadecimal characters will also generate bytes, allowing for streaming of bytes without intervening commands. Hitting ENTER (Carriage Return) will end the Byte mode. This command differs from the “b” command in that power (strong pull-up) is applied to the bus after the last bit of the first byte is issued. (Subsequent bytes will be performed but will not be followed by strong pull- up.)

**Key j** – Performs the same type of operation as the “b” key above, but with only single bits.

Single ASCII digits of 0 or 1 value only are allowed, and single ASCII digits are returned. Hitting ENTER (Carriage Return) will end the Bit mode.

**Key ~ (tilde)** – Performs the same type of operation as the “b” key above, but with only single bits.

Single ASCII digits of 0 or 1 value only are allowed, and single ASCII digits are returned. Hitting ENTER (Carriage Return) will end the Bit mode. This command differs from the “j” command in that power (strong pull-up) is applied to the bus after the first bit is issued. (Subsequent bits will be performed but will not be followed by strong pull- up.)

**Key f** - Performs a 1-Wire bus “first” operation.

This operation searches the bus and finds the first 1-Wire or iButton device, displaying the device serial number. A single character, either a “+” or a “-“, is returned to indicate if there are more (“+”) or no more (“-“) parts remaining to be found. Similar command s: “n (next)”

**Key n** – Performs a 1-Wire bus “next” operation.

This operation searches the bus and returns the next 1-Wire or iButton device, displaying the device serial number. A single character, either a “+” or a “-“, is returned to indicate if there are more (“+”) or no more (“-“) parts remaining to be found. (If used again after a “-“ response is received, this function will find the first part again.) Similar commands: “f (first)”

Key **t** – When followed by a two-character hexadecimal value, will change the search type to this function command value. Using “tF0” will make the search normal. Using “tEC” will make the search conditional and will discover only devices for which the search conditions are satisfied. (See the data sheets for each individual Dallas iButton or 1-Wire device for specific search type command codes.)

Key **l** (lower case L) – Test the level of the 1-Wire bus and report a “0” or “1” followed by a carriage return.

Key **x** – This will take the 1-Wire bus to a low level.

Issue a reset (“r”) command, or any bit or byte command, to return the 1-Wire bus to functionality. This function is used to cause a bus-wide reset by robbing power from all the bus devices for a few seconds.

Key **^** (hat or shift-6) - Switches the device to the 57,600 baud serial port data rate.

The host terminal will be required to switch to 57,600 baud before it can communicate with the device any further. When a “break” condition is detected, The device resets and returns to the 9600 baud data rate, so sending the “^” followed by more 9600 baud data will often find the device resetting and the speed returning to 9600 baud. See note (1) below.

Key **`** (single quote under ~) - Switches the device to the 38,400 baud serial port data rate.

The host terminal will be required to switch to 38,400 baud before it can communicate with the device any further. When a “break” condition is detected, the device resets and returns to the 9600 baud data rate, so sending the “^” followed by more 9600 baud data will often find the device resetting and the speed returning to 9600 baud. See note (1) below.

Key **,** (comma) - Switches the device to the 19,200 baud serial port data rate.

The host terminal will be required to switch to 19,200 baud before it can communicate with the device any further. When a “break” condition is detected, the device resets and returns to the 9600 baud data rate, so sending the “^” followed by more 9600 baud data will often find the device resetting and the speed returning to 9600 baud. See note (1) below.

Key **|** (vertical bar) – This will cause the device to enter a pass-thru mode.

In this mode, all activity on the serial port is passed through (inverted) to the 1-Wire bus, and all activity on the 1-Wire bus is passed-through (inverted) to the serial port input line. This mode can be used to bypass the device and allow the serial port direct access to the 1-Wire bus. Because the device is no longer able to interpret serial data in this mode, the only way to get out of the pass thru mode is by a power-on-reset of the device.

**Note 1:** The 1-Wire bus with relaxed timing suitable for long lines can only process bits at a rate of about 14,000 per second. Streaming bytes using the (b) command will fail if the baud rate is set to more than 19,200 because the host will overrun the 1-Wire bus. When the baud rate is set to any value greater than 19,200 the host commands must be paced to assure that 1-Wire bus overrun does not occur.

Key **.”** (Period) – This will turn OFF the dynamic pull-up (DPU) driver in the 1-Wire bus interface.

The DPU helps extend the useable length of the 1-Wire bus by increasing the charge current at the appropriate times in the 1-Wire waveform. In the rare event that the action of the DPU causes a problem on shorter networks, this command allows it to be turned off. The device responds with a carriage return/line feed. The DPU is turned back ON by any reset of the device (break or power cycle).

Key **“(** (Left parenthesis) – Relax 1-Wire timings.

The Extended Wire “(“ command relaxes the 1-Wire timings to accommodate long bus lengths. Invoke this timing if you are experiencing bus errors due to reflections from busses over 300 feet in length.

Key `\\` (backslash backslash) – Causes the device to enter the 1-Wire sniffer mode and switch to the 57,600 baud serial port rate. The 1-Wire sniffer function listens to the 1-Wire bus and decodes data on the bus. The device ceases serving as the bus master in this mode. When data is detected on the bus, it is converted into hexadecimal bytes and displayed. Each time a 1-Wire bus reset is detected, a carriage return and line feed (CR/LF) are sent. This mode is used to debug 1-Wire master programs by capturing the actual data bytes that are observed on the bus. Overdrive speed is not supported by the sniffer mode. The switch to 57,600 baud is necessary to keep up with standard speed 1-Wire bus data. Sending a “break” condition will cause the device to be reset to default settings, and to the default 9600 baud serial port data rate.

Key `\f` (backslash f) – Family search first. The `\f` requires two hex characters as data.

Similar to the standard first command, the family search first expects the next two hex characters to specify the 1-Wire family code to be included in the search. This causes the search to start with a particular family code. The standard “n” (next) command is used to retrieve the next 1-Wire serial number from the bus search. Since the family search may not terminate on the last device on the bus, care should be taken to check the family code returned for each serial number. In the case of a family search mismatch, a “?” is returned instead of the normal “+” or “-”. Similar commands: “f” (first) “n” (next)

Example: `\f26`

A first is performed beginning with serial number  
0000000000000026.

## Power Issues:

The device obtains power with which to operate by ‘robbing’ it from two signal lines that are available in the host DB-9 serial port. These signals (DTR and RTS) should both be held in the high state for best performance (although the device will often operate just fine with only one of these lines high). The device uses an efficient switching regulator to convert the port pin power to useable levels with minimal losses. Different terminal programs handle the signal lines in different ways. If both RTS and DTR are taken low for a few seconds, the device will be robbed of all power and will be reset to its power-up state. If the host computer is a very low power type, like a lap-top or palm-top computer, it may not provide sufficient operating power for the device. In the situation where port power may not be sufficient, it is important to make sure that the host program holds both DTR and RTS in the high state. Many very small (palm-type) computers generate serial signals that are as low as 3 volts (which are not legal RS232 voltages). These computers will not work with the device. Desktop PCs and quality laptops will provide valid RS232 signal levels and more than enough power to operate the device.

## 1-Wire Communications Example

The sequence for reading a DS18B20 temp sensor is straightforward:

1. Issue a 1-wire reset (r).
2. Enter byte mode (b) and address the ROM by sending 0x55 followed by the ROM address in reverse byte order (that is, if the discovered id is E6000003DA0E128 you would address it as 28E1A03D000000E6).
3. Send the convert command 44.
4. Exit byte mode <CR>.
5. Wait at least 900ms for the conversion to complete.
6. Issue a 1-Wire reset (r).
7. Enter byte mode (b) and address the ROM as before.
8. Send the read command 0xBE.
9. Send two read commands as FFFF. The returned data will contain the temperature reading in Intel (little-endian) order as a 16-bit signed int.
10. Exit byte mode (CR).

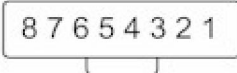
The returned value will be in 1/16 deg C increments. So a return value of 5701 represents 0x0157, or 343 in decimal. Dividing by 16 gives us 21.4 degrees C, or 70.6 degrees F. Mind the sign bit, or values below freezing will appear to be unusually warm by several thousand degrees.

### Debugging hint:

The power up default of the 18B20 is 85 degrees C (185 degrees F). Look carefully if you receive this value. This might indicate that a convert has never been executed by this device. It is possible to address an 18B20 and NOT have enough power available for it to execute a convert.

## Pin outs

Viewed looking into the RJ-45 socket.



- Pin 5 ... 1-Wire Data (Blue)
- Pin 4 ... 1-Wire Ground (Blue/White)
- Pin 3 ... Auxiliary (Green)

Wire colors are for "standard" Cat5 cable.

## Glossary of terms used in this manual:

- <CR> ... a single ASCII carriage return character
- <LF> ... a single ASCII line feed character

Contains copyright material from iButtonLink LLC. Used with permission.