# The LinkHub™

## Users Guide

**An Advanced, Intelligent, Multiport**
**RS232 Serial Port Interface**
**for the iButton™ and 1-Wire™ Bus**

Revision C
9/27/2004

**Software Version 1.1b and above**

## Introduction:

The LinkHub™ is an advanced, intelligent, multiport serial port interface for use with Dallas Semiconductor 1-Wire™ and iButton™ components. It uses superior digital and analog methods to accomplish more reliable operation on a wide variety of network topologies. It also offers many functions that simplify network communications and some diagnostic tools as well.

There are five (5) 1-wire busses (1 internal and 4 external) which are logically or'ed into a single composite 1-wire bus. Software written for the Dallas Semiconductor DS9097U interface will work even though the 1-wire devices are physically on separate busses. Each external bus can be up to 1000 feet long. This yields a logical 1-wire network of 4000 feet with 4 times the network weight of a standard 1-wire bus.

The LinkHub is a powered device, providing +12V and +5V power on the same cable as the 1-wire bus. In addition, local EEPROM storage is provided for user configuration data. The LinkHub is available as three models, differentiated by the amount of EEPROM memory, 32KB, 256KB, and 512KB.

The LinkHub has been designed to operate reliably on long and short 1-Wire busses. This is accomplished by a special analog interface design that uses matched impedances and slew rate controls, as well as smart cable pre-charge. In addition, the firmware (programming) is adaptive and makes adjustments for a variety of network parameters automatically.

The LinkHub is based on the Link, designed in cooperation with Dallas Semiconductor Corp., Dallas, Texas, who is the manufacturer of the 1-Wire and iButton devices and various accessories. Some terms used in this document may be trademarks of Dallas Semiconductor Corporation, and are used with their permission.

## DS9097U Emulation:

The LinkHub is programmed to closely emulate a Dallas DS9097U serial port interface so that existing software can use it without modification. This allows The LinkHub to be used in place of the Dallas DS9097U in most cases where increased performance or reliability is required. Some DS9097U functions are not emulated by The LinkHub, as described below.

### *Calibration:*

Unlike the DS9097U, The LinkHub does not require calibration from the serial port because it has a crystal-controlled time base. This means that drift in the interface data rate and waveform timing is not an issue. (A calibration byte is expected by The LinkHub for DS9097U software compatibility, although it is ignored by The LinkHub.)

*Overdrive Speed:*

The LinkHub does not support Overdrive speed bus communication. Because the analog components in the bus interface are carefully tuned for optimum long- and short-line performance at standard speed, the higher speed mode cannot also be supported. However, for the vast majority of applications, this is not an issue.

*Flex Timing Modes:*

The LinkHub uses 1-Wire waveform timing that is carefully and dynamically controlled by the intelligent control algorithms. Flex mode timing changes that could be done in the DS9097U are unnecessary and are not supported in The LinkHub. However, The LinkHub will emulate the Flex Mode registers, and so will appear to behave like the DS9097U for software compatibility.

*EPROM Programming:*

The LinkHub does not support +12V pulses for programming EPROM 1-Wire or iButton devices. Programming pulses for 5V devices (EEPROM) are supported.

*Device Version:*

The LinkHub reports device version 7 when queried using the normal DS9097U method. This allows the software to distinguish The LinkHub from a standard Dallas DS9097U interface, which (at the time of this writing) returns a 3.  Since the Link (single port) also returns a device version 7, the user can use the “ “ (blank) ASCII command to determine if a LinkHub is present.

**Embedded EEPROM Memory:**

The LinkHub includes 32K Bytes of EEPROM memory.  This EEPROM is not part of the 1-wire bus structure.

**LinkHub Serial Number:**

Each LinkHub contains a DS2401 serial number chip on the internal 1-wire bus.  The serial number chip will show up on 1-wire searches unless the internal bus is deactivated.

**LED indicators:**

The LinkHub has two LED indicators on the right side.  The LED on the left is a heart beat indicator that the LinkHub is executing.  At idle, the LED will cycle from toggle once a second.  The LED on the right indicates 1-wire bus traffic.  The LED will illuminate when there is 1-wire traffic on any bus.

**ASCII Control Functions:**

In addition to the DS9097U emulation, The LinkHub also provides many 1-Wire bus functions in response to ASCII commands. This allows The LinkHub to be used with any terminal program or serial port communications API to perform iButton or 1-Wire bus functions without software drivers. The LinkHub 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. As a rule, these special commands consist of every command code that is not used in the DS9097U emulation.

The LinkHub ASCII functions are letters and symbols as follows:

Key **h** – Displays a list of LinkHub ASCII commands that are available. (These are in addition to the standard DS2480/DS9097U binary command codes that are available.)

Key **r** – Performs a 1-Wire bus reset and returns the status of the bus, 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.

Key **b** – Places the LinkHub 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.

**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.

**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.

**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.

Key **p** – Places the LinkHub 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. *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.

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.) Note that if a Family Search (started with \f command) is in progress, next may return a "?" if encountering a family code overshoot.

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)** – This will 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 **z** – Turn off the switched +5V line to the LinkHub output connectors.

Key **d** – Turn on the switched +5V line to the LinkHub output connectors

Key **&** - Report the current state of the switch +5V line on the LinkHub output connector(s).

Key **$** - Start arrival/departure scanning (See *Scanning* below.)

Key **: (colon)** - Switches the device to the 115,200 baud serial port data rate. The host terminal will be required to switch to 115,200 baud before it can communicate with The LINK 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.

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 LINK 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.

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 LINK 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.

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 LINK 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.

Key (**spacebar**) – Display the LinkHub software version number.

Key **** (**backslash**) – Escape character to the LinkHub expanded command set. The sniffer mode of the standard link is not supported by the LinkHub. The \ command by itself does nothing ... it must be followed by a expanded command character.

Key **\f** (**backslash f**) – Family search first. 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. 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 overshoot, a "?" is returned instead of the normal "+" or "-". See the First ("f") command.

Key **\h** (**backslash h**) – Display help for the expanded command set.

Key **\C** (**backslash C**) – Set the active Channel Mask. This command is follow by a two digit hex value representing a bit mask for channels which should be active. There are 5 OW busses. The bit patterns are:

```
0x37 ... All busses enabled
0x01 ... External bus 1
0x02 ... External bus 2
0x04 ... External bus 3
0x10 ... Internal bus
0x20 ... External bus 4
```

A one (1) makes the channel active. A zero (0) makes the channel inactive.

Key **\c** (**backslash c**) – Report the active Channel Mask. The LinkHub will respond with
a 2 character hex value reporting the current Channel Mask contents.

Key **\E (backslash uppercase E**) – Report maximum internal EEPROM address. The LinkHub will respond with a 6 digit Hex number which represent the address which can be specified with Read/Write EEPROM commands. The 32KB LinkHub has 32768 EEPROM addresses. A 256KB LinkHub has 262136 addresses. A 512KB LinkHub has 524272 addresses. A 32KB LinkHub will respond with 007FFF.

Key **\M** (backslash uppercase M) – Set EEPROM Read/Write Address. The \M command is followed by a 6 digit Hex Address specifying the next EEPROM address to be read/written. Example: \M000030 would specify address 30 (hex) as the next read/write address.

Key **\m** (backslash lowercase m) – Report EEPROM Read/Write Address. The next EEPROM read/write address will be returned followed by a carriage return. Example: \m might return 0001FF<CR> meaning the next byte to be read/written is 0001FF hex.

Key **\R** (**backslash uppercase R**) – Read EEPROM hex data. The \R command is followed by a 2 digit hex number specifying the number of bytes to read. The contents of each EEPROM byte will be returned as 2 digit hex numbers terminated with a carriage return

Example: **\M0003FF\R04** will return the contents of EEPROM addresses 0003FF through 000402 followed by a carriage return. The next EEPROM read/write address would be 000403 after the execution of the example.

Key `\r` **(backslash lowercase r)** – Read EEPROM ASCII data. Data is read from the EEPROM and returned as ASCII characters. The read is terminated by a carriage return from the EEPROM data stream. The carriage return is also sent to the user. Example:

```
\M000100\wThis is a test<CR>
\wThis is line two<CR>
\WThis is line three<CR>
\wThis is the last line<CR>
\M000100\r\r\r\r
```

would return:

```
This is a test<CR>
This is line two<CR>
This is line three<CR>
This is the last line<CR>
```

Key `\W` (**backslash uppercase W**) – Write EEPROM hex data. The \W command is followed by a 2 hex characters representing the next byte to be written. A carriage return terminates the write. The carriage return is NOT written to the EEPROM.

Example: `\M0000FF\W416243<CR>` will put the ASCII string "AbC" into bytes 0000FF through 0000102 of the EEPROM memory. The next EEPROM read/write address would be 0000103 after the execution of the example. Note that the <CR> was NOT written to the EEPROM.

Key `\w` **(backslash lowercase w)** – Write ASCII data to EEPROM. The \w command is followed by ASCII characters terminated by a carriage return <CR>. The carriage return IS written to the EEPROM. Example:

`\M000100\wThis is a test<CR>` writes 15 characters to EEPROM beginning at address 000100. The next EEPROM read/write address would be 00010F after the execution of the example.

Key **|** (**vertical bar**) – This will cause The LinkHub to enter a pass-thru mode. In this mode, all activity on the serial port output line 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 LINK 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 LinkHub.

***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. Pacing means waiting for the character response before sending another character. Even though the LinkHub has a bigger typeahead buffer (64 characters) than the original Link, pacing will insure that the LinkHub does not overrun the OW bus.***

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 LINK responds with a carriage return/line feed. The DPU is turned back ON by any reset of The LINK (break or power cycle).

Key **!** ... (**Exclamation point**) Cause a formatted dump of the LinkHub RAM contents and register states to be produced on the serial interface. A watchdog timer reset of the LinkHub occurs after the dump is complete.

**Power Issues:**

The LINK provides an Auxiliary I/O line on the RJ connector. The LinkHub provides switched +5V on the same pin.

**Scanning:**

It is often useful to scan the 1-Wire bus and report the ***arrival*** of a new 1-Wire device or the ***departure*** of a device from the bus. A bus scanning function is included in The LinkHub to serve this purpose. When turned ON using the '$' character, the scanning system continually performs First and Next device discovery operations and builds a table of up to eight (8) device serial numbers. As a new device appears on the 1-Wire bus, and after the same device serial number (with correct CRC8 and a non-zero family code) has been observed on two subsequent full discovery passes, an Arrival is reported as a string with a "!" (exclamation-point) character, a comma, and then the arriving device serial number. When a device has been present on the bus and then is not found in ten (10) successive complete discovery cycles, it is reported as a Departure with a "?" (question mark) followed by a comma and then the serial number of the device that departed from the bus. (This de-bounces the departure.) Scanning is terminated by a Reset, First, Next or any number of other operations.

***The scanning system will not work properly if more than 8 iButton or 1-Wire devices are present on the bus. This is due to the limited memory available in The LinkHub.***

Questions about The LinkHub should be directed to info@iButtonLINK.com.

## Appendix A – 1-Wire Communications Examples

The sequence for reading a DS18B20 temp sensor via The LinkHub™ is straightforward:

1. Issue a 1-wire reset (`r`).
2. Enter byte mode in pull-up mode (`p`) and address the ROM by sending `55` followed by the ROM address in reverse byte order (that is, if the discovered id is `E60000003DA0E128` 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 `BE`.
9. Send two read commands as `FFFF`. The echoed data will contain the temperature reading in Intel (little-endian) order as a 16-bit signed integer (as 4 hex characters)
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:**

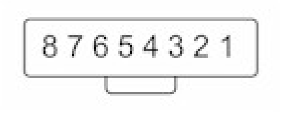
The power up default of the device is 85 degrees C (185 degrees F). Look carefully if you receive this value. This indicates that a convert has never been executed by this device. It is possible to address a device and NOT have enough power available for it to execute a convert.

Appendix B … Wiring Diagrams

## Pin outs

### LinkHub External 1-Wire Port

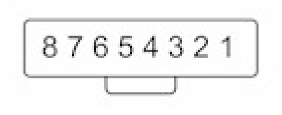
**(viewed looking into port)**



Pin 1 ... Ground
Pin 2 ... V+ (+12V)
Pin 3 ... Switched +5V (Aux)
Pin 4 … Ground
Pin 5 … OW Data
Pin 6 … Ground
Pin 7 ... +5V
Pin 8 ...Ground

### LinkHub Serial Port

**(viewed looking into port)**



Pin 1 ... Gnd
Pin 2 ... Tx (Transmit Data)
Pin 3 ... Gnd
Pin 4 … CTS (Clear to Send)
Pin 5 … n/c
Pin 6 … Rx (Receive Data)
Pin 7 ... DTR (Data Terminal Ready)
Pin 8 ...n/c