Embedded Data Systems

HA7Net Ethernet to 1-Wire Bus Master

Users Manual and Programmers Guide

Revised 10/13/2004

Introduction

Welcome to the HA7Net. A product that is designed to simplify the integration of distributed 1-Wire / iButton networks into your application or system. By acting as an Ethernet to 1-Wire bridge, the HA7Net serves as an efficient tool that can be used to overcome a variety of challenges related to the successful implementation of 1-Wire MicroLans.

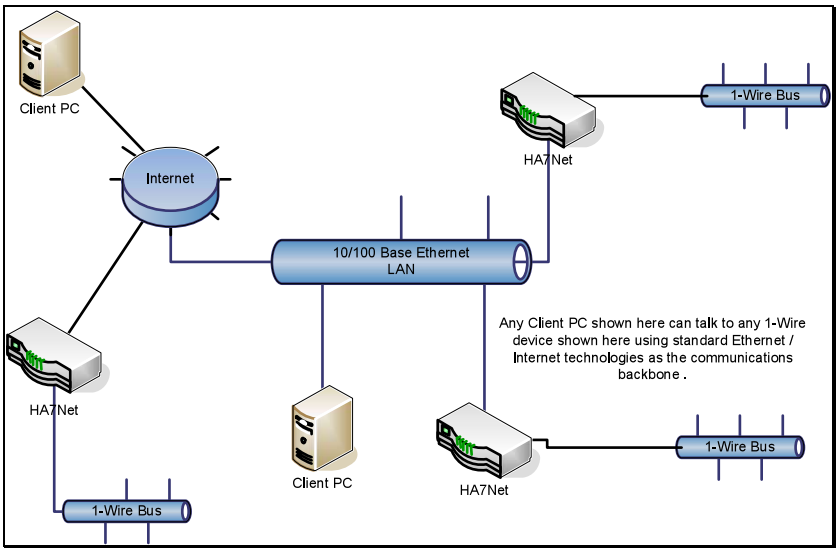


Figure 1

The HA7Net effectively allows you to use industry standard Ethernet networking products to build out the backbone of your system, providing several important benefits: First, the individual 1-Wire MicroLans can be kept to a smaller physical size, thereby improving reliability and decreasing installation costs. Second, since the HA7Net communicates via TCP/IP over standard Ethernet, you can take advantage of existing corporate LANs, the Internet, and in-house MIS expertise for building out the backbone of your sensor network, as shown in figure 1. Another advantage is the number of readily available libraries and developer tools that can be used to communicate with the HA7Net that are widely available for common development platforms.

Intended Audience

This document is intended for the developer / integrator charged with the task of designing the 1-Wire sensor MicroLans and writing the software that will interface with the HA7Net

HA7Net Hardware Overview

Form Factor

The HA7Net is available in both panel mount and DIN rail form factors. The panel mount version suitable for desktop or wall mounting while the DIN rail version is compatible with industry standard DIN-35 mounting rails typically found in industrial environments.

Power Supply Requirements

The HA7Net is fitted with a 2.5mm power jack suitable for connection to any 6-12 Volt DC power supply having a F2 (2.5mmID x 5.5mmOD) female type plug wired for positive polarity on the center conductor. The power supply should capable of delivering 500mA.

Internal Clock

The HA7Net is equipped with an internal battery backed DS2417 real-time clock, which itself is a 1-Wire device connected to the primary 1-Wire bus. The clock is used for such things as managing security certificate lifetimes as related to the SSL portion of the internal http server, and is also used as a reference for time stamping each result page returned from the HA7Net. Since the clock is connected to the primary 1-Wire bus, it can be used in simple diagnostics to determine the correct operation of the HA7Net. This also means that it can be completely accessed by the integrator. Note that the HA7Net uses this clock for proper operation and setting the clock to arbitrary values will almost certainly cause undesired side-effects in the HA7Net. The DS2417 clock has a specified accuracy of +/- 2 minutes per month.

Internal Battery

The HA7Net is equipped with a user replaceable Lithium CR2032 coin cell battery. This battery serves to power the internal real-time clock such that the HA7Net's clock settings are not lost during power failures, etc.

Ethernet Interface

The HA7Net is equipped with a standard 8-Wire RJ-45 Ethernet jack that meets the ISO 8877 requirements for 10/100BASE-T. Both half and full duplex modes of operation are supported on the Ethernet interface.

1-Wire Interface

For ease of use, the HA7Net is equipped with 3 parallel 1-Wire ports. Each port is a standard 6-Wire RJ11 jack, pinned for use with 1-Wire devices as follows:
<Insert Pin-out Table/Diagram Here>
The Vdd signal can be configured to one of three modes of operation by setting jumper J6 located inside of the HA7Net. The possible configurations are as follows:

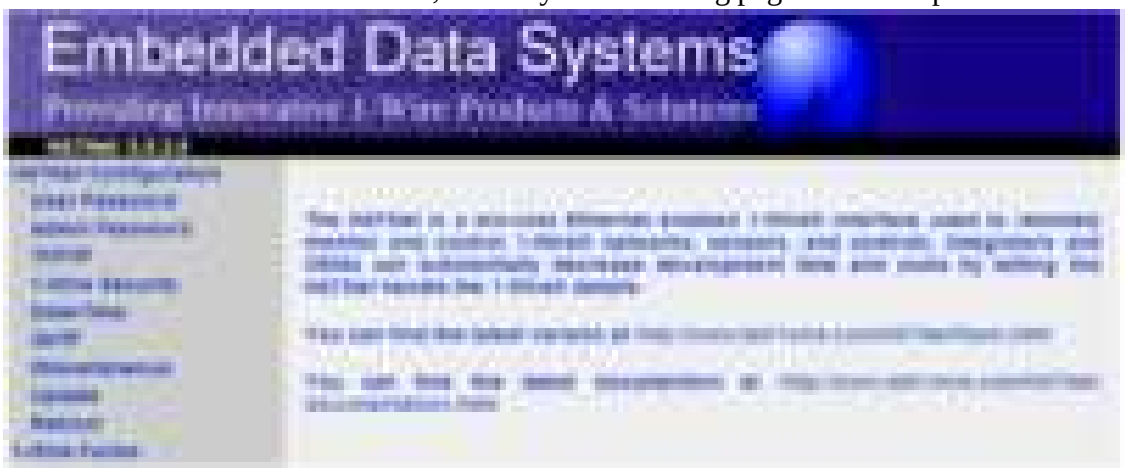
- +5 Volt DC power supplied, up to 200mA. This might be useful if you want to provide external power to sensors such as the DS18x20.

- GND- Pin will be held to the common 1-Wire GND level
- Floating- The pin will be physically disconnected inside the HA7Net. This might be useful if you have other devices on the bus that need to control this line.

The HA7Net supports up to 2000 feet of cabling and 100 1-Wire devices on a CAT-5 twisted pair net, and automatically provides smart strong-pull-up for sensors. ESD protection to more than 27kV (IEC801-2 Reference Model) is provided on the 1-Wire bus.

Configuration / Setup

The HA7Net is generally usable right out of the box, but in most cases you will want to configure certain options, as discussed here. Configuration is performed by using a web browser to connect to the HA7Net, whereby the following page should be presented:



The left hand side of this page contains a menu of the various configuration options available. To enter into any of the configuration screens, you must be able to supply the current administration password. The default administrator username / password combination is as follows:

User: Admin
Password: eds

Note that we highly recommend changing the default passwords.

User Password

If set, this password will be required in order to access any of the 1-Wire Forms or API. By default, this password is not set. This password is intended to be used to provide access control to your 1-Wire bus. The username associated with this password is 'user'.

Admin Password

The administrative password is required in order to access any of the HA7Net configuration screens, as well as the telnet interface. Additionally, the administrative password can be used in lieu of the 'User' password to gain access to the 1-Wire bus. The default administrator username / password is as follows:

User: Admin (can not be changed.)
Password: eds

TCP/IP Settings

The HA7Net ships with the following default TCP/IP configuration: First, the device will attempt to obtain an IP addresses from a local DHCP server. If that fails, then the HA7Net will default to a static IP address of 192.168.0.250, with a netmask of

255.255.255.0. Using the TCP/IP configuration screen, the following parameters can be specified:

- Obtain IP Configuration via DHCP
  This instructs the HA7Net to load all TCP/IP settings from a local DHCP server.
- Use Static on DHCP Failure:
  By enabling this option, the HA7Net will "fallback" to its statically configured parameters if it is unable to negotiate a DHCP lease. If this is not enabled, then the HA7Net will continuously attempt to locate a DHCP server.
- Static IP Address
- Network Mask
- Default Route (Gateway)
- Primary DNS server
- Secondary DNS server

Note that if the device successfully configures itself via DHCP, then you will either have to consult your DHCP server logs, or take advantage of the HA7Net's multicast listener in order to discover the HA7Net's IP address.

Date/Time

This form can be used to configure the time and date stored in the HA7Net's real-time clock. We recommend setting this to GMT, but this is not a requirement and you may want to use another time zone depending on your application. Pressing the "Guess" button will populate the fields by converting your PC's time to GMT. Note that the HA7Net can automatically keep its clock synchronized to a network time server, by configuring the SNTP settings.

SNTP

SNTP is a simple network time protocol, which is supported by many commonly available servers. By enabling the "Sync Time using SNTP" option and configuring at least one SNTP server, the HA7Net will periodically synchronize its internal clock to that of the SNTP server. Note that using this option will force the HA7Net's clock to GMT. For more information on SNTP, please consult your favorite Internet search engine.

Miscellaneous

The miscellaneous tab is used to configure the following parameters:

- Lock idle timeout (seconds):
  This timeout discusses the amount of time that a 1-Wire bus lock can remain idle before the HA7Net forcibly releases it. For a complete discussion of 1-Wire bus locking, see the section entitled "Concurrency Management", later in this document.

Update

The update form is used to flash the HA7Net with new versions of its firmware. Instructions for using this form will likely accompany any new firmware updates you receive.

Reboot

Some configuration settings require restarting the HA7Net before they will take effect. This form allows you to reboot the HA7Net from remote.

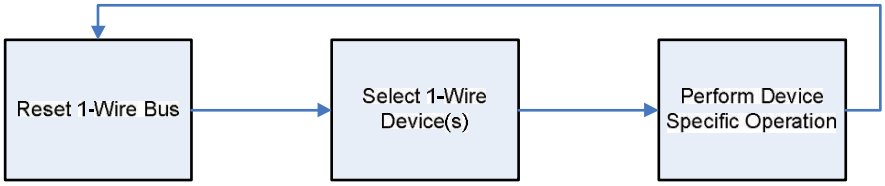
1-Wire Technology Overview

1-Wire is a technology created by Dallas Semiconductor which is centered around a data bus that implements a simple signaling scheme used to perform two-way communications between a single master and multiple peripheral devices over a single connection. A powerful feature common to all 1-Wire bus devices is that each and every device, whether in a chip or an iButton form, has a factory-lasered serial number that will never be repeated in any other device. That is to say, every device is unique. This allows any single device to be individually selected from among many that can be connected to the same bus wire. Because one, two, or even dozens of 1-Wire devices can share a single wire for communications, a binary searching algorithm is used to find each device in turn. Once each device serial number is known, any device can be uniquely selected for communication using that serial number to address it.

Device Selection

The first part of any 1-Wire communication generally involves the bus master issuing a "reset" which synchronizes the entire bus. A slave device is then selected for subsequent communications. This can be done by selecting all slaves, or by selecting a specific slave (using the serial number of the device). These commands are referred to collectively as "network" or ROM (Read-Only-Memory) commands, and are implemented on the HA7Net in the form of the 'Search', 'Address', and 'Match' commands. Once a specific device has been selected, all other devices drop out and ignore subsequent communications until the next reset is issued.

Because each 1-Wire device type performs different functions and serves a different purpose, each has a unique protocol once it has been selected. Even though each device type may have different protocols and features, they all have the same selection process and follow the command flow as seen in the following figure:



Device Interaction

Once a device is isolated for bus communication (selected) the master can issue device-specific commands to it, send data to it, or read data from it. This is generally accomplished using combinations of the following HA7Net commands: 'Write Block', 'Write Bit', 'Read Bit', 'Read Pages', 'Read Records', and 'Write Record'. Since each 1-Wire device type performs different functions, they each implement their own protocol for interacting with it after the selection process. The HA7Net commands above will allow you to effectively interact with any 1-Wire device manufactured, present or future, regardless of its protocol.

Family Codes

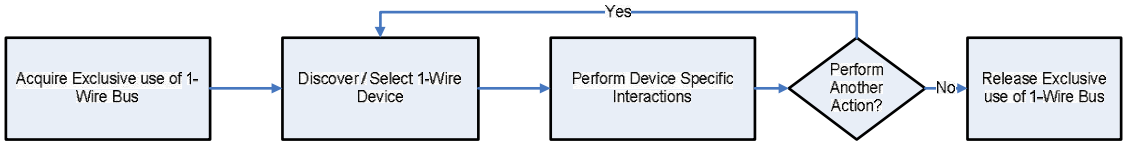
An integral part of the unique serial number in each 1-Wire device is an 8-bit family code. This code is specific to the device's model type. Because each device model performs different functions, this code can be used to select the protocol that will be used to control or interrogate it. The following table provides a partial mapping of family codes to Dallas Semiconductor part numbers:

Family Code	**Part Number** ()iButton Package	**Description** (Memory size in bits unless specified)
01 (hex)	(DS1990A)*, DS2401	1-Wire net address (serial number) only
04	(DS1994), DS2404	4k NV RAM memory and clock, timer, alarms
05	DS2405	Single addressable switch
06	(DS1993)	4k NV RAM memory
08	(DS1992)	1k NV RAM memory
09	(DS1982), DS2502	1k EPROM memory
0A	(DS1995)	16k NV RAM memory
0B	(DS1985), DS2505	16k EPROM memory
0C	(DS1996), (DS1996x2), (DS1996x4)	64k to 256k NV RAM memory
0F	(DS1986), DS2506	64k EPROM memory
10	(DS1920), DS1820, DS18S20	Temperature with alarm trips
12	DS2406, DS2407	1k EPROM memory, two channel addressable switch
14	(DS1971), DS2430A	256-bit EEPROM memory and 64-bit OTP register
18	(DS1963S)	4k NV RAM memory and SHA-1 engine
1A	(DS1963L)	4k NV RAM memory with write cycle counters
1D	DS2423	4k NV RAM memory with external counters
1F	DS2409	2 channel addressable coupler for sub-netting
20	DS2450	4 channel A/D
21	(DS1921), (DS1921H), (DS1921Z)	Thermochron™ temperature logger
22	DS1822	Econotemperature
23	(DS1973), DS2433	4k EEPROM memory
24	(DS1904), DS2415	Real-time clock (RTC)
26	DS2438	Temperature, A/D
27	DS2417	RTC with interrupt
28	DS18B20	Adjustable resolution temperature

33	(DS1961S), DS2432	1k EEPROM memory with SHA-1 engine

Typical 1-Wire Communication Process

Putting this altogether, communications with devices on a 1-Wire bus typically follow a flow similar to the following:



HA7Net Ethernet Communication Interfaces

The HA7Net provides 3 interfaces exposed to the Ethernet.

1. HTTP Server -The primary means of interfacing with the HA7Net is via the HTTP protocol, available by default on port 80 (non-SSL) and port 443 (SSL).
2. Telnet Server -A debugging facility is also provided via a Telnet interface located on the standard telnet port (23).
3. Multicast Listener –A multicast listener is integrated into the HA7Net for ease of device discovery in dynamically configured networks. When the HA7Net hears the appropriate multicast packet, it will respond with a directed UDP packet containing information about how to contact the HA7Net.

Telnet Interface

As an aid in both troubleshooting and the initial integration process, the HA7Net offers a telnet server that can be used to monitor what is happening internal to the device. This interface is available on the standard telnet port (23), and is secured by the HA7Net's administrative level password.

The telnet interface provides access to messages and logging from each of the individual subsystems in the HA7Net, currently:

- Http Server
- Debug / Logging Facility
- 1-Wire Network Activity
- Telnet Server
- Clock Maintenance

The messages are formatted in the standard Unix syslog format, and each subsystem can be individually configured for the desired logging level ranging from practically none (Emergency) to nearly everything (Debug).

Multicast Listener

In order to provide a mechanism for discovering HA7Nets on a dynamically addressed (i.e. DHCP) networks, the HA7Net has an integrated multicast listener that will respond to properly formatted multicast packets with a directed UDP packet. This allows the client to discover all of the HA7Nets within reach of a single multicast packet.

Packet Format

The multicast packet transmitted from the client to the HA7Net should be constructed as follows:

```
    Char      signature[2];  -> ”HA”
    WORD16    command;       -> 0x0001
i.e
    0x48410001     -> “HA\000\001”
```

This packet should be transmitted to group (IP): 224.1.2.3, port 4567.

When this packet is detected, the HA7Net will respond with a UDP packet directed back to the same IP address and originating port of the client that transmitted the multicast. The format of the response packet is as follows:

```
char      signature[2];  -> "HA"
WORD16    command;       -> 0x8001
WORD16    port;          -> Non-SSL http port (80)
WORD16    sslport;       -> SSL http port (443)
char      serial_num[12] -> HA7Net serial (MAC)
char      dev_name[64]   -> HA7Net device name
```

Note that the serial_num and dev_name fields will not be null terminated if the data occupies the entire allocation, otherwise they will be null terminated. The client should be able to locate each of the HA7Nets by querying the client's underlying TCP/IP stack for the remote IP information of the received UDP packet. As shown above, the response packet will include the port numbers of both the non-SSL and SSL http servers. These servers default to being located on ports 80 and 443 respectively, but can be re-configured by the integrator. If either of the http servers is disabled, then the port number will be returned as 0.

The following is an excerpt from a Visual Basic code sample for transmitting the multicast packet and receiving the response:

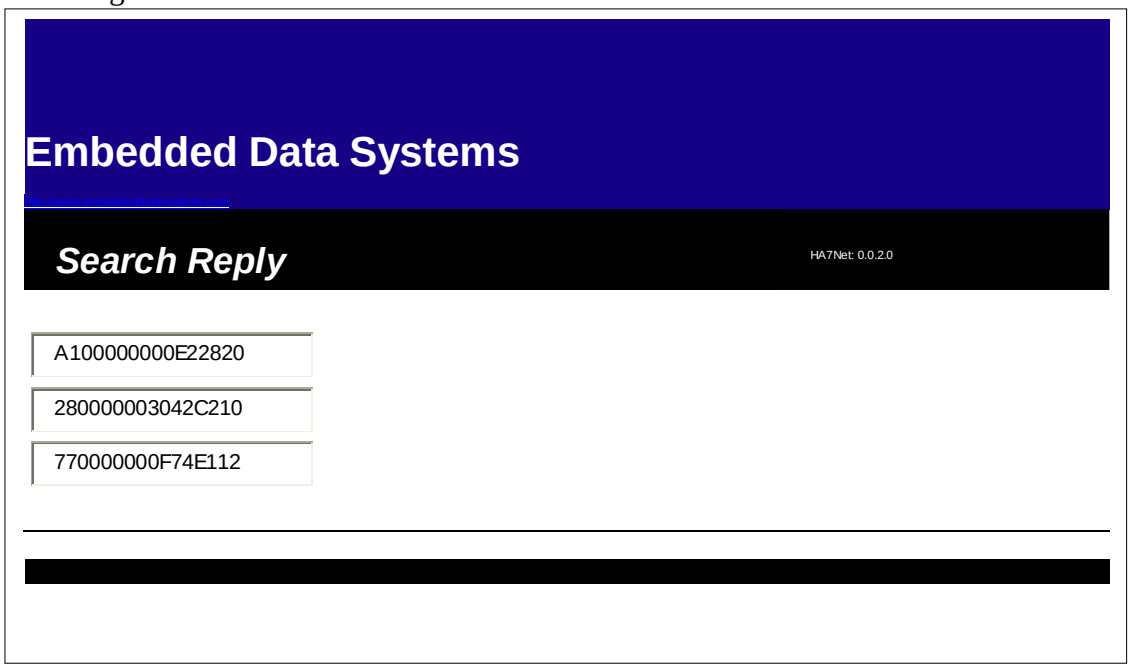
```
Private Sub Form_Load()
    Dim sckHndl As Long
    Winsock1.Protocol = sckUDPProtocol
    sckHndl = Winsock1.SocketHandle
    setsockopt sckHndl, SOL_SOCKET, SO_BROADCAST, 1, 1

    With Winsock1
           .RemotePort = 4567
           .RemoteHost = "224.1.2.3"
           .SendData "HA" & Chr$(0) & Chr$(1)
    End With
End Sub

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
    Dim responseData As String
    Winsock1.GetData responseData, vbString, bytesTotal
    Debug.Print responseData & Winsock1.RemoteHostIP & Winsock1.RemotePort
End Sub
```

HTTP Interface

Primary communication with the HA7Net is via the http protocol. The actual data is exchanged in the form of html documents which have been designed to accommodate both human readability and 100% reliable machine parsing. This is accomplished through the use of unique, predictably named form fields that can be automatically parsed by most DOM parsers, and easily digested via regular expressions on lighter weight platforms. Data is passed to the HA7Net in the form of parameters placed in the URL.

Since the result pages are human readable, both proof of concept and integration time is reduced since you can effectively interact with the HA7Net using a standard web browser.

For example, to discover all of the devices that are connected to the 1-Wire bus, the client would request the following URL from the HA7Net: http://HA7.net/1Wire/Search.html. The HA7Net will then respond with an html document that renders similar to the following in a standard web browser:



You can see that the serial number of each 1-Wire device appears in a form field. Taking a look at the source of this html document, you can see how the data is also easily machine parseable.

Each html response document is typically divided into three primary sections that are organized into html tables. The three sections are used to discuss:

- The data portion of the response.
- Exceptions that may occur during the request.
- Statistical information about the request.

Data Table

The following is the section of the html document shown above that discusses the serial numbers of the 1-Wire devices:

```
<table name="Addresses" id="Addresses">

<tr><td><INPUT CLASS="HA7Value" NAME="Address_0" ID="ADDRESS_0" TYPE="text" VALUE="A100000000E22820">
</td>
</tr>

<tr><td><INPUT CLASS="HA7Value" NAME="Address_1" ID="ADDRESS_1" TYPE="text" VALUE="280000003042C210">
</td>
</tr>

<tr><td><INPUT CLASS="HA7Value" NAME="Address_2" ID="ADDRESS_2" TYPE="text" VALUE="770000000F74E112">
</td>
</tr>
</table>
```

From this, you can see that all of the 1-Wire addresses are contained in a table named "Addresses", and each of the individual addresses are contained in a text field named "Address_x", where x is a 0 based sequential number. This can data can be parsed automatically by DOM parsers, or by simply by using a regular expression similar to the following:

```
's/.*<INPUT.*NAME="Address_\(.*\)".*VALUE="\(.*\)".*./\2/p'
```

In fact, the following is a sample Linux command line that can be used to communicate to an HA7Net and print the address of all of the 1-Wire devices connected to that HA7Net to stdout:

```
lynx –source 'http://192.168.253.136/1Wire/Search.html' | sed --silent –e \
's/.*<INPUT.*NAME="Address_\(.*\)".*VALUE="\(.*\)".*./\2/p'
```

Exception Table

The following is the section of the html document shown above that discusses any exceptions that may have occurred during the request:

```
<table name="Exceptions" ID="Exceptions">
<tr>
<td><INPUT CLASS="HA7Value" NAME="Exception_Code_0" ID="Exception_Code_0" TYPE="hidden" VALUE="0"></td>
<td><INPUT CLASS="HA7Value" NAME="Exception_String_0" ID="Exception_String_0" TYPE="hidden" VALUE="None"></td>
</tr>
</table>
```

In this example, there where no exceptions encountered as indicated by the form field named "Exception_Code_0" having its value equal to "0". When this is the case, the exception fields are type hidden to prevent rendering in the web browser. If there had been an actual exception the fields would be type text. It is recommended that client applications always check for exceptions and handle them as appropriate. A complete listing of exceptions and possible causes are discussed later in this document.

Statistics Table

The following is the section of the html document show above that discusses statistics regarding the specific request:

```
<table name="Statistics" ID="Statistics">
<tr>
<td><INPUT TYPE="HIDDEN" NAME="Completed_0" VALUE="101446">
</td>
</tr>
</table>
```

Shown here is the only statistics data currently implemented, which is the timestamp when the html document was created on the HA7Net. The value represents the number of seconds since January 1, 1970, and is stored in the field named "Completed_0". This timestamp can be used to determine the when the request was completed regardless of any latency between the client and the HA7Net. This can be useful when calculating the rate at which a piece of data is changing across multiple requests.

Concurrency Management

The physical layer of the 1-Wire MicroLan is designed such that only one data request or operation can be made on the 1-Wire bus at a time. It is also common for a single logical transaction to span multiple 1-Wire requests. Given this modal nature of the 1-Wire bus, and the multi-user nature of the http interface, it is necessary for the HA7Net to provide a means of concurrency and bus contention management. Concurrency management is

provided on the HA7Net via a traditional locking mechanism wherein a client will request a lock on the 1-Wire bus and no other clients will be allowed access to the bus until the lock is either explicitly released or expires. The default maximum idle lifetime of the locks can be configured per HA7Net, and is discussed later in this document.

The general process for interacting with the HA7Net in a multi-user environment looks like this:

- Client requests lock of the 1-Wire bus
  - http://HA7.Net/1-Wire/GetLock.html
- HA7Net responds with a new LockId when the bus is available
- Client performs transactions while referencing the LockID
  - http://HA7.Net/1-Wire/Temperature.html?LockID=xxxxxxx&Address=1234567812345678
  - http://HA7.Net/1-Wire/Search?LockID=xxxxxx&FamilyCode=10&Conditional=True
- Client releases previously acquired lock
  - http://HA7.Net/1-Wire/ReleaseLock.html?LockID=xxxxxxx

For the duration of the lock, no other clients will be allowed to access the 1-Wire bus.

In summary, all 1-Wire activities are handled atomically within the HA7Net so all requests made that do not reference a LockID are guaranteed to have uninterrupted access to the 1-Wire bus for the duration of that particular request. By requesting and referencing an explicit lock, uninterrupted access to the 1-Wire bus can be guaranteed across multiple requests.

HTTP Interface API Reference

Command List Overview

The following is an overview of each http request that can be made to the HA7Net. Detailed reference on each command can be found on the pages that follow.

Search – Implements the 1-Wire search algorithm including the regular search, family search, and conditional search. This function is used to discover the 64-bit ROM codes (addresses) of all the devices connected to the 1-Wire bus. The search function can optionally restrict the returned list of addresses to those devices belonging to a particular family, and/or those that are in a device defined conditional state.

Address Device –Used to select the particular 1-Wire device on the 1-Wire bus that you want to talk to.

Match ROM –Provides a shortcut method to reset the 1-Wire bus and reselect the 1-Wire device that was selected with the last 'Address Device 'command.

Reset – Used to reset the 1-Wire bus.

Power Down Bus –Used to completely power down the 1-Wire bus.

Read Bit –Used to read a single bit from the 1-Wire bus.

Write Bit –Used to write a single bit to the 1-Wire bus.

Write Block –Used to write and simultaneously read up to 32 bytes of data to the 1-Wire bus.

Read Pages –Used to read one or more consecutive pages of memory from 1-Wire memory devices.

Read File Records –Used to read one or more consecutive TMEX formatted file records from 1-Wire memory devices.

Write File Record – Used to write a single TMEX formatted file record to a 1-Wire memory device.

Search ROM Command

Base URL:

http://HA7Net.com/1Wire/Search.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Optional	FamilyCode	Two byte hex number used to restrict the results to devices of a given family.
Optional	Conditional	A '0' or '1', with '1' indicating that only devices in a conditional state are to be returned.

Returns:

'Addresses' –Table that contains a list of 8 byte 1-Wire ROM address codes, with each ROM code residing a text field named 'Address_x' where x is a 0 based sequential integer.

Description

The search ROM command implements the 1-Wire search algorithm including each of the regular search, family search, and conditional search functionalities. This function is used to discover the 64-bit ROM codes (addresses) of devices connected to the 1-Wire bus. The search function can optionally restrict the returned list of addresses to include only those devices belonging to a particular family, and/or those that are in a device defined conditional state.

Examples:

To retrieve a list of every device on the 1-Wire bus:
http://HA7Net.com/1Wire/Search.html

To retrieve a list of all DS18S20 temperature sensors on the 1-Wire bus:
http://HA7Net.com/1Wire/Search.html?FamilyCode=10

To retrieve a list of all devices on the 1-Wire bus in a conditional state:
http://HA7Net.com/1Wire/Search.html?Conditional=1

To retrieve a list of all DS18S20 temperature sensors on the 1-Wire bus that are in an alarm state:
http://HA7Net.com/1Wire/Search.html?FamilyCode=10&Conditional=True

Address Device Command

Base URL:

http://HA7Net.com/1Wire/AddressDevice.html

Parameters:

Required	*Address*	8 byte hex 1-Wire ROM Address
Optional	*LockID*	Ten byte decimal number previously returned by GetLock.

Returns:

'Addresses' –Table that contains the single 8 byte 1-Wire ROM address code. This value is stored in a text field named 'Address_0', and is the same value that was passed in the 'Address' parameter.

Description

This command will reset the 1-Wire bus, and then select the particular 1-Wire device on the 1-Wire bus that you want to talk to. Generally, communications on the 1-Wire bus occur between the bus master (HA7Net), and a single 1-Wire device. Before you can have communications with that particular device, you must select the device which accomplishes two things:

1. It tells the device you are addressing that you intend to communicate with it.
2. All other devices drop off of the bus until the next bus reset.

Examples:

To address a 1-Wire temperature sensor having ROM code '280000003042C210':
http://HA7Net.com/1Wire/AddressDevice.html?Address=280000003042C210

This address might have been previously discovered using the 'Search' command, stored in a database from an initial system installation, etc.

See Also:

The 'Match ROM' command provides a shortcut to the 'Address Device' command, in that the 'Address' parameter is optional... the previously selected ROM code will be automatically selected.

Match ROM Command

Base URL:

http://HA7Net.com/1Wire/MatchRom.html

Parameters:

Optional	LockID	**Ten byte decimal number previously returned by GetLock.**

Returns:

'Addresses' –Table that contains the single 8 byte 1-Wire ROM address code. This value is stored in a text field named 'Address_0', and is the ROM Id of the matched device.

Description

This command is used to simultaneously reset the 1-Wire bus and then reselect the 1-Wire device that was most recently selected with the 'Address Device' command.

Examples:

To address a 1-Wire temperature sensor having ROM code '280000003042C210', perform some actions on the bus, then reset the bus and reselect the same sensor:
http://HA7Net.com/1Wire/AddressDevice.html?Address=280000003042C210
... Do some work with the Sensor here...
http://HA7Net.com/1Wire/MatchRom.html

See Also:

The 'Address Device' command provides a method to directly select a device given its ROM id.

Reset Command

Base URL:

http://HA7Net.com/1Wire/Reset.html

Parameters:

Optional	LockID	**Ten byte decimal number previously returned by GetLock.**

Returns:

This command does not return anything other than the page statistics and an exception, if applicable.

Description

This command is used to reset the 1-Wire bus. Resetting the bus returns all devices on the bus to the addressing mode, where they wait for you to select the next device(s) that you want to talk to.

Examples:

To reset the 1-Wire bus:
http://HA7Net.com/1Wire/Reset.html

Power Down Bus Command

Base URL:

http://HA7Net.com/1Wire/PowerDownBus.html

Parameters:

Optional	LockID	**Ten byte decimal number previously returned by GetLock.**

Returns:

This command does not return anything other than the page statistics and an exception, if applicable.

Description

This command is used to completely power down the 1-Wire bus. During normal operation, the HA7Net holds the voltage level on the 1-Wire bus high. For power sensitive applications, the Power Down Bus command can be called to reduce the 1-Wire bus voltage to 0. Any 1-Wire activity from the HA7Net will cancel the Power Down mode, although the suggested method for coming out of power down is via the Reset command.

Examples:

To power down the 1-Wire bus:
http://HA7Net.com/1Wire/PowerDownBus.html

Write Block Command

Base URL:

http://HA7Net.com/1Wire/WriteBlock.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Optional	Address	8 byte hex 1-Wire ROM Address
Required	Data	1-32 bytes of data formatted as HEX

Returns:

'ResultData' –Table that contains the data read back from the 1-Wire bus. This value is stored in a text field named 'ResultData_0'.

Description

Perhaps the most important HA7Net command, the Write Block command allows you to write and read any raw data to the 1-Wire bus under any context. This will allow you to implement any device specific protocols.

If the optional 'Address' parameter is specified, the HA7Net will first reset the 1-Wire bus, then select the device having that address prior to writing the block of data to the bus. If the 'Address' parameter is not present, then the HA7Net will simply write the data and reads data back from the bus without regard for the current state of the bus.

While a complete understanding of the 1-Wire electrical interface is not necessary, there is an important concept for you to understand in order to effectively use this command. Every 1-Wire device can only talk back to the bus master during a read cycle on the bus. The only time that read cycles are created on the bus is immediately after each bit that the host master writes to the bus. Therefore, if the response you expect to read back from the 1-Wire bus is longer than the data you intend to write to the bus, you must pad the data you are writing with 1 bits in order to generate the necessary time slots for the devices to write their entire response back to you. In short, you can only read as many bits from the 1-Wire bus as you write. For a complete explanation of 1-Wire bus timing and communication, please see chapter one of the Book of DS19xx iButton Standards from Dallas Semiconductor.

Examples:

To tell the 1-Wire temperature sensor having ROM code '280000003042C210' to perform a temperature conversion:
http://HA7Net.com/1Wire/WriteBlock.html?Address=280000003042C210&Data=44

To read the 9 byte scratchpad from the temperature sensor above, in order to obtain the temperature information:

http://HA7Net.com/1Wire/WriteBlock.html?Address=280000003042C210&Data=BEFFFFFFFFFFFFFFFFF
Notice the 9 'FF 'bytes. This will create the necessary time slots on the 1 wire bus in order for the sensor to write back the response, as discussed in the description above.

To select the DS2406 with ROM code 2400000007377212, issue the Channel Access command and read the Channel Info Byte which contains the input latches, output latches, and sensed levels of the two IO lines PIOA and PIOB:
http://HA7Net.com/1Wire/WriteBlock.html?Address=2400000007377212&Data=F5CFFFFF

Note that 4 bytes were written, and 4 bytes were read.

Read Bit Command

Base URL:

http://HA7Net.com/1Wire/ReadBit.html

Parameters:

Optional	LockID	**Ten byte decimal number previously returned by GetLock.**

Returns:

'Bits' –Table that contains the bit read from the 1-Wire bus. This value is stored in a text field named 'Bit_0'.

Description

This command is used to read a single bit from the 1-Wire bus regardless of context.

Examples:

To read a single bit from the 1-Wire bus:
http://HA7Net.com/1Wire/ReadBit.html

Write Bit Command

Base URL:

http://HA7Net.com/1Wire/WriteBit.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Required	Bit	Bit to write to the 1-Wire bus (Should be either 0 or 1)

Returns:

'Bits' –Table that contains the bit read from the 1-Wire bus. This value is stored in a text field named 'Bit_0'.

Description

This command is used to write a single bit from the 1-Wire bus regardless of context. Any response from a 1-Wire device on the bus is read and returned.

Examples:

To read a single bit from the 1-Wire bus:
http://HA7Net.com/1Wire/ReadBit.html

Read Pages Command

Base URL:

http://HA7Net.com/1Wire/ReadPages.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Required	StartPage	Page number to begin reading on. Should be an integral value between 0 and 255.
Optional	PagesToRead	Number of pages to read. Defaults to 1 if not specified.
Optional	Address	8 byte hex 1-Wire ROM Address

Returns:

'Pages' –Table that contains the pages read from the 1-Wire device. This value is stored as hex in a text field named 'Page_x', where x is a 0 based sequentially numbered integer.

Description

This command is used to read one or more consequtive memory pages from 1-Wire devices that have memory pages. If the Optional 'Address' parameter is given, then the 1-Wire bus will be reset and the devices specified by 'Address' will be selected prior to reading the memory pages. If 'NumPages' is not specified, then only one page will be read. 'StartPage' must be specified.

Examples:

To read a 4 consecutive memory pages beginning with page 1 from the from the 1-Wire device having ROM code '2400000007377212':
http://HA7Net.com/1Wire/ReadPages.html?Address=2400000007377212&StartPage=1&PagesToRead=4

Read File Records Command

Base URL:

http://HA7Net.com/1Wire/ReadFileRecords.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Required	StartRecord	Page number to reading the record from. Should be an integral value between 0 and 255.
Optional	RecordsToRead	Number of records to read. Defaults to 1 if not specified.
Optional	Address	8 byte hex 1-Wire ROM Address

Returns:

'Records' –Table that contains the records read from the 1-Wire device. This value is stored as hex in a text field named 'Record_x', where x is a 0 based sequentially numbered integer.

Description

The Read File Record command is used to read one or more consecutive TMEX formatted file records from 1-Wire devices that have memory pages. The CRC16 is automatically checked on the records, and if any record is not a valid Touch Memory File Record, an exception will be returned and all good records up to the point of error will be returned.

The byte count, continuation code, and CRC16 bytes of the TMEX record are stripped from the file record before it is returned. See Chapter 7 of the Book of DS19xx iButton Standards from Dallas Semiconductor for a complete discussion of the Touch Memory File Structure.

If the Optional 'Address' parameter is given, then the 1-Wire bus will be reset and the device specified by 'Address' will be selected prior to reading the file records. If 'RecordsToRead' is not specified, then only one record will be read. 'StartRecord' must be specified.

Examples:

To read a 4 consecutive file records beginning with the record located at page 1 from the from the 1-Wire device having ROM code '2400000007377212':
http://HA7Net.com/1Wire/ReadFileRecords.html?Address=2400000007377212&StartRecord=1&RecordsToRead=4

Write File Record Command

Base URL:

http://HA7Net.com/1Wire/WriteFileRecord.html

Parameters:

Optional	LockID	Ten byte decimal number previously returned by GetLock.
Required	RecordNumber	Page number to write the record to. Should be an integral value between 0 and 255.
Required	Data	Up to 28 bytes in hex (56 hex characters)
Optional	Address	8 byte hex 1-Wire ROM Address

Returns:

This command does not return anything other than the page statistics and an exception, if applicable.

Description

The Write File Record command is used to write a TMEX formatted file record to the memory page specified by 'RecordNumber'. During the write process, the HA7Net will automatically set the continuation pointer in the file record to the next memory page, which limits its use to files having contiguous records. The HA7Net will always write 28 bytes of data to the file record, even if less than 28 bytes are supplied. The record is validated as it is written to the device, and an exception will be returned if the record fails to write correctly.

Please see Chapter 7 of the Book of DS19xx iButton Standards from Dallas Semiconductor for a complete discussion of the Touch Memory File Structure.

If the Optional 'Address' parameter is given, then the 1-Wire bus will be reset and the device specified by 'Address' will be selected prior to reading the file records. 'StartRecord' must be specified.

Examples:

To write the record "HA7 is Easy to USE" into the file that contains page 21h of the DS1996 with ROM code EF00000003B7890C:
http://HA7Net.com/1Wire/WriteFileRecord.html?Address=EF00000003B7890C&RecordNumber=21&Data=484137206973204561737920544F20555345