Application Note

AN_408

FT90x Telnet to UART Bridge

Version 1.1

Issue Date: 2017-02-16

This application note describes an Ethernet to serial bus (UART) implemented on the FT90x. It uses the telnet protocol for transferring data between the serial bus and a remote network client and an HTTP configuration interface.

Bridgetek Pte Limited (BRTChip)

178 Paya Lebar Road, #07-03 Singapore 409030
Tel : +65 6547 4827 Fax : +65 6841 6071
Web Site: http://brtchip.com

Table of Contents

1 Introduction

The FT900/FT901 and FT905/FT906 devices include an Ethernet interface. This can be used with a suitable networking stack to allow the device to be networked. This application note shows how the Ethernet interface can be used to bridge the built-in UART to a remote telnet client. In addition, the network information such the MAC and IP address are retrieved and stored on the onboard external I2C EEPROM, giving the advantage that subsequent application software updates will not overwrite this information.

1.1 Overview

UART Bridge on the FT90x. Data transmitted to the bridge from the remote client is forwarded onto the UART interface of the FT90x, data received on the UART is transmitted to the remote client.

The network interface supports DHCP to obtain configuration information or it can be configured using a web page served by the FT90x firmware. UART configuration is performed using the same web page.

The configuration is stored either on FT90x Flash or external EEPROM, so that it is not lost after power off:

- FT90x FlashROM: Hostname, UART configuration (baud/bits/parity/stop/flow)
- External EEPROM: MAC address, IP/ gateway/mask address, DHCP setting

If the EEPROM key is not valid, the software enables DHCP to be allocated a new address.

Third-party open source code is used to implement a TCP/IP stack and a telnet library in this application note:

- TCP/IP - lwIP (LightWeight IP) library which has been ported to the FT90x.
- Telnet – libtelnet.
- Printf – tinyprintf.

Links to resources for these libraries are in Appendix A – References.

1.2 Scope

The telnet protocol is widely used on LANs to provide bi-directional communications. It is not encrypted or otherwise secured and is not generally used on open networks. Security concerns have made the SSH protocol more widely used. This application note does not intend to provide any encryption or security and as such should be deployed on private LANs where access is controlled.

The external I2C EEPROM used in this example is the 24AA02E48T-I/OT which is a 2K serial EEPROM device. This memory is present on the MM900EVxA , excluding the MM900EV-LITE.

1.2.1 Features

The application note highlights the use of lwIP to provide a TCP/IP stack. The "arch" folder of lwIP in the source code for the application note contains the architecture-specific parts of lwIP required for the FT90x.

1.2.2 Enhancement

Enhancements to this application might be:

- To implement SSH protocol instead of telnet.
- To provide access to the UART through a web page.
- Allow only permitted IP addresses to connect to the device.
- Reduce the overall size of the application code.

This telnet to UART bridge example application should be treated as an example. Full source code is provided allowing users to build and modify if required:

http://brtchip.com/wp-content/uploads/FT90x/AN408_Telnet_to_UART_Bridge.zip

See section 5 Importing into the FT90x Toolchain for more information.

2 Project Overview

The project files for this application note are divided into the following folders.

Folder	Description
Source	Application source code and abstraction files.
httpdfs	File system for web server.
lib	Library files.
lib\tinyprintf	tinyprintf library.
lib\libtelnet	Telnet library.
lib\lwip	lwIP library.

Table 2.1 Project Files Overview

The application source code is contained within the "Sources" folder. The main() function and the high-level functions of the application are in the "main.c" file. In this folder there are 2 abstraction files, one for the network interface ("net.c") and another for the telnet library ("telnetd.c").

2.1 Main Program

The main program is responsible for handling data between the UART interface and the telnet library.

It also handles configuration by the web interface. The code generates data for the simple server side includes (SSI) functions in the lwIP httpd server. Data is returned through HTTP GET requests to update the configurations. The httpd server uses callbacks for the server size and includes the HTTP GET handler.

The dlog features described in AN_365 FT900 API Programmers Manual are used for persistent storage and retrieval of configuration data.

2.2 Network Code

The networking layer is provided by the lwIP library which has been ported to support the FT90x. The lwIP code is available separately from FTDI with the FT90x architecture extensions.

2.2.1 Network Abstraction

The network interface is abstracted from the main project in the file "net.c". All lwIP features which refer to an interface (netif) are passed through this layer.

The lwIP network interface is initialized and configured using data supplied from the main application. The function which processes incoming packets for the network interface within lwIP is called from the net_tick() function.

2.2.2 lwIP Library

The library is available with a BSD-style license. Version 2.0.0 of the code is used as the basis. No changes have been made to the code from the lwIP project page.

The FT90x architecture code is responsible for obtaining packets from the Ethernet interface and transmitting packets to the Ethernet interface.

The application makes extensive use of the callback methods in lwIP to permit sending and receiving of data.

The httpd app is used to provide an interface for configuration. This makes use of server side includes (SSI) and CGI scripts (HTTP GET requests) to present configuration data to the user and change it when requested to do so.

2.3 Telnet Code

2.3.1 Telnet Abstraction

A small abstraction layer is used to interface libtelnet to the lwIP code. Several callback functions are used to receive information from the libtelnet library which can be data or control messages. Data is passed from this layer to the UART handler in the main application code or the network interface in the lwIP code.

2.3.2 Telnet Library

The libtelnet library is used to interpret the in-band control codes and manage the telnet protocol. This is public domain code. It is not modified for this application.

A minimum set of DO and DON'T requests and WILL and WON'T instructions are used. Terminal settings to control the UART interface are not implemented, instead the configuration interface is used.

2.4 Persistent Storage

Configuration parameters including the network hostname and UART parameters are stored in FlashROM on the device. The 4 kB long datalogger partition is used to keep this data between power cycles. The data is updated each time the data is changed by the configuration interface.

When the application starts it checks the data stored for the network and UART configuration in the persistent storage. Two 32-bit keys are used (one for the network and one for the UART sections) to ensure that the data read is valid. If one value is incorrect then default values are loaded for that function.

When the FlashROM is reprogrammed by the FT90x Programming Utility it is normal for this data to be overwritten. This will result in default parameters being used until the data is reprogrammed.

Other networking parameters such as MAC address, IP/ gateway/mask address and DHCP setting are stored on external EEPROM via I2C reads and writes.

2.5 Web Pages

The configuration web pages are stored in a folder called httpdfs on the top level of the directory structure for the project. The raw html and image files are compiled into a single C file (fsdata.c) which is included in the source code of the application. An internal library in the lwIP httpd code reads this data and extracts files for the web service.

A batch files (makefs.bat) and a compilation utility (makefsdata.exe) Windows platforms are included to perform the compilation and place the fsdata.c file in the correct place. Changes to the web pages have to be compiled before the application is recompiled.

Further details on how lwIP performs Server Side Include (SSI) and acts on the HTTP GET requests used to update the configuration are to be found in the lwIP documentation.
To make changes to the webpage, edit the shtm or html files using a text editor. Knowledge of HTML code is required. The page must be rebuilt by browsing to the project's top level directory in a windows console and typing makefs.bat as shown in Figure 2.1.

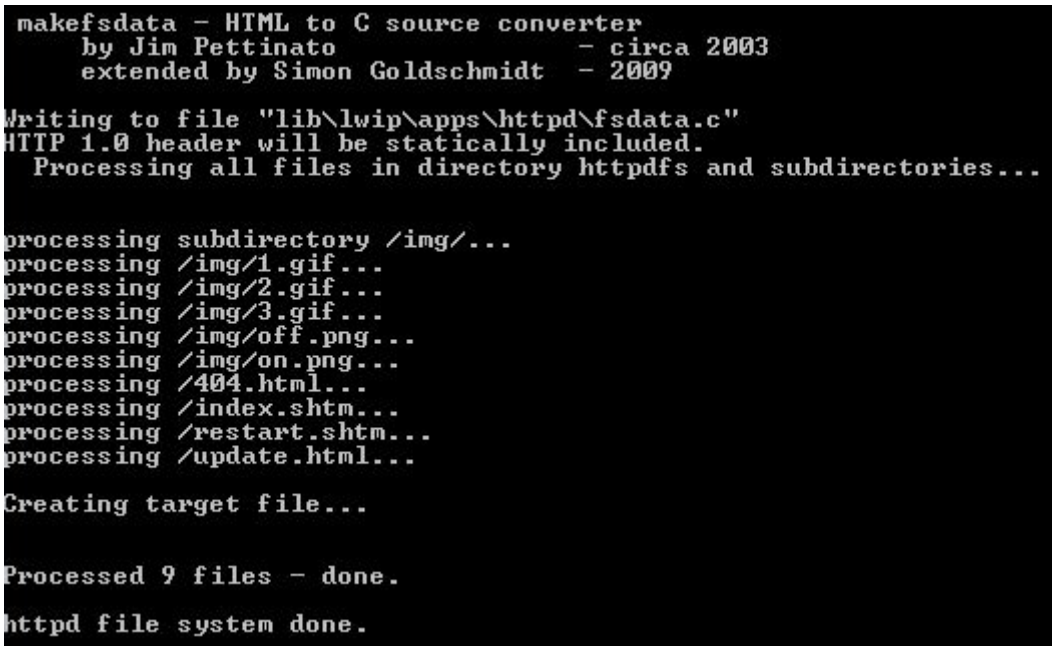

Figure 2.1 Makefs.bat Output

3 Code Structure

The code is written to use the lwIP library efficiently. Callbacks are used to link the lwIP code to the telnet abstraction layer and then to the telnet library.

Data sent via telnet is routed in both directions through the abstraction layer to the telnet library.

The main function "bridge()" polls the UART and sends any data received through the telnet library (Figure 3.1). The telnet library processes the data to encapsulate it in the telnet protocol. When this is complete a callback function in the abstraction layer routes the data to the lwIP library for transmission (Figure 3.2).

In the other direction, data received from the network by the lwIP library is sent to the telnet library for processing. Data is returned to the abstraction layer by a callback and is then sent back to the main function for transmission over the UART (Figure 3.3).

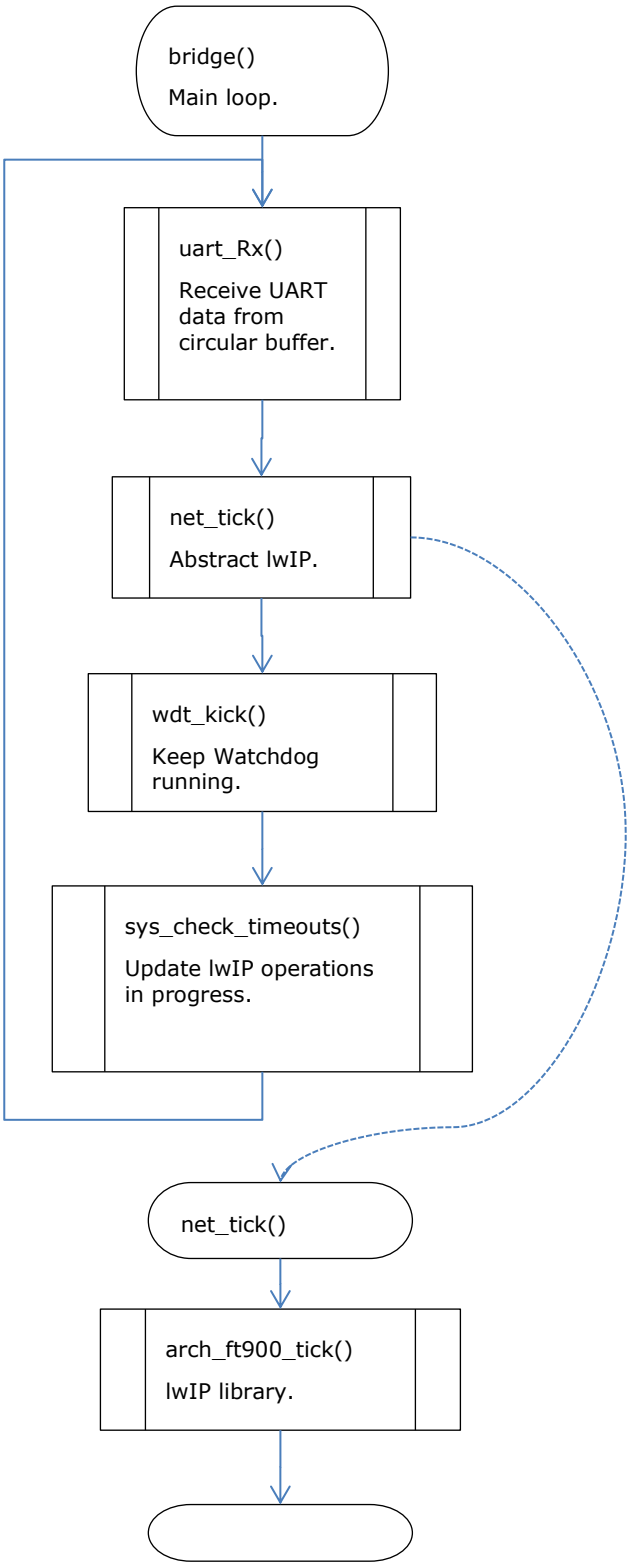

Figure 3.1 Main Function Flowchart

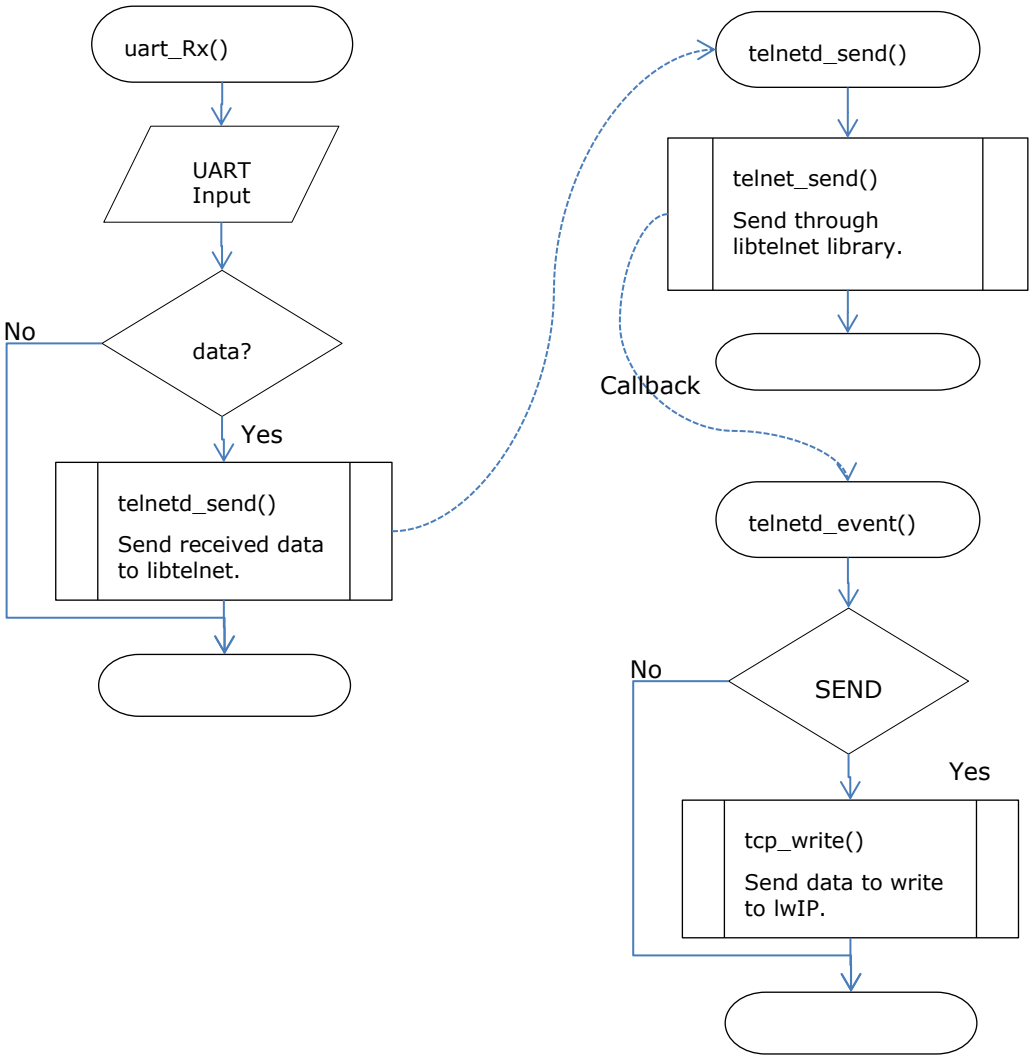

Figure 3.2 UART to Telnet Function Flowchart

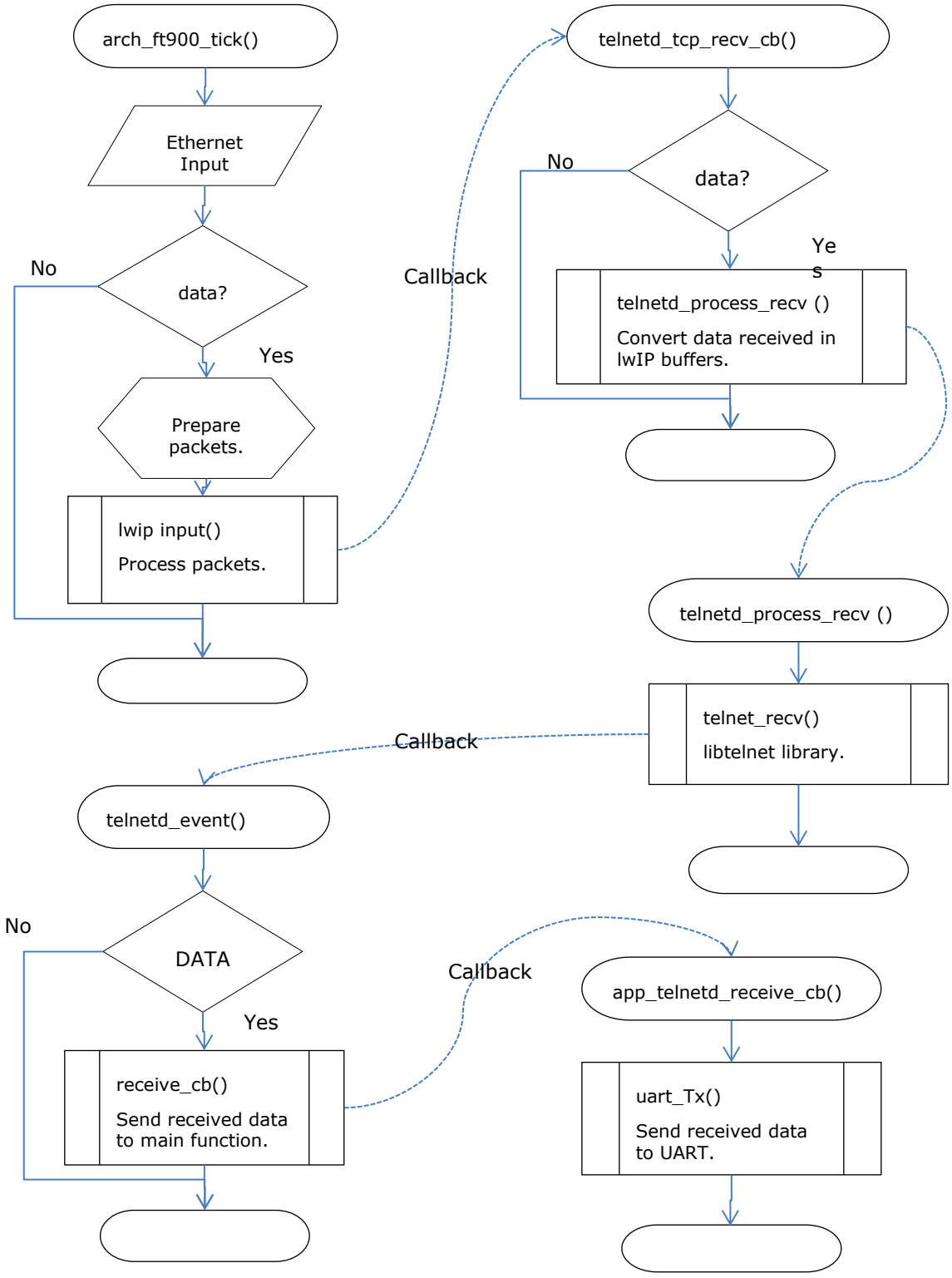

Figure 3.3 Telnet to UART Function Flowchart

4 Configuration

This section describes the default configuration used by the application and how to change the configuration.

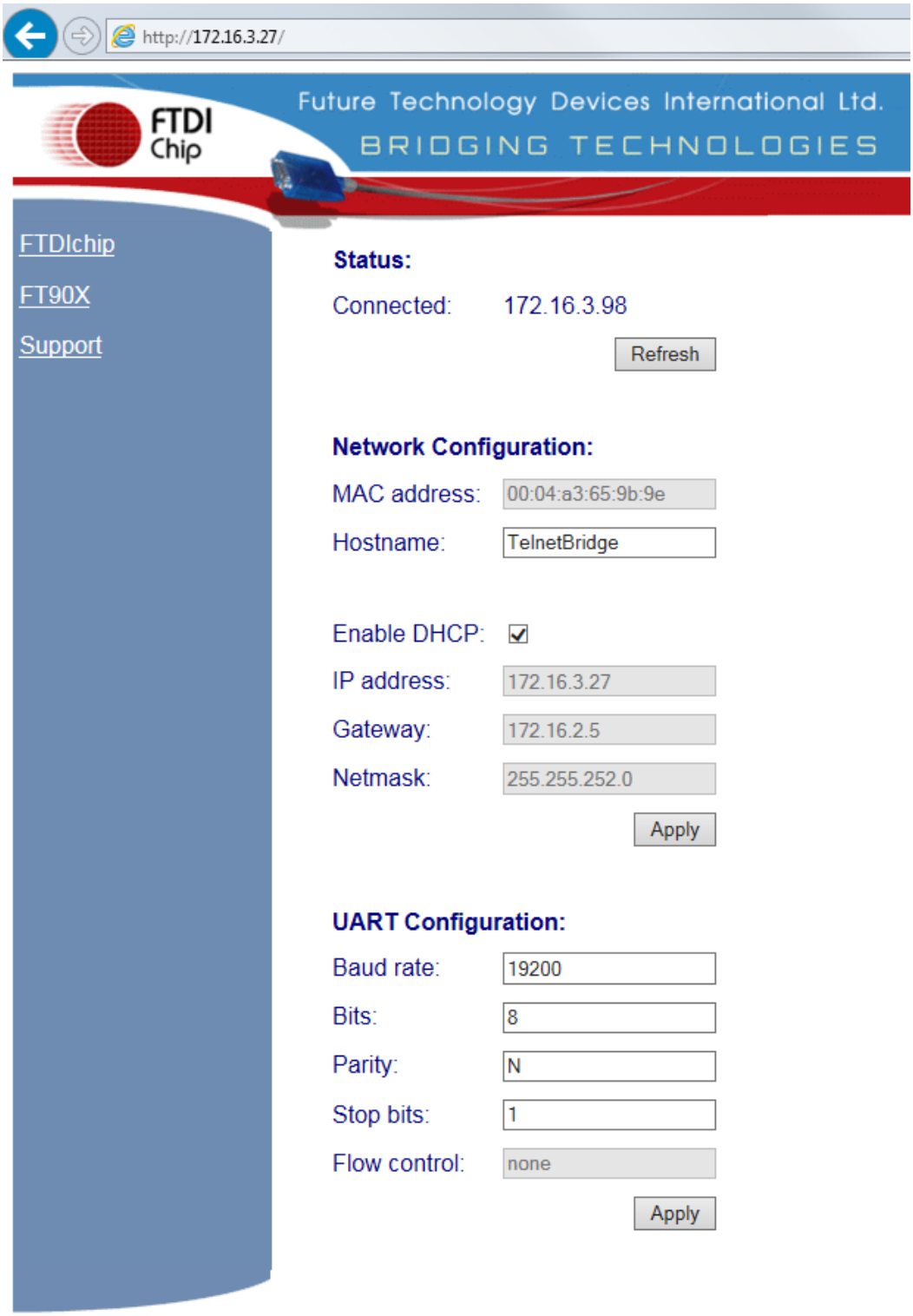

Figure 4.1 Web Configuration Interface

A web configuration page is accessible when the device connects to the network. This allows both networking and UART settings to be altered.

In order to serve the configuration web page the device must be connected to the network and have obtained (or have been configured with) an IP address. DHCP is enabled by default allowing the application to request an IP address when it is first started.

In many cases it is possible to query the DHCP server on a network to find out the IP address allocated to a particular device. In this application the device will send the DHCP server the hostname "TelnetBridge" which some DHCP servers may make available to allow users to identify the allocated address more easily.

Another way to obtain the IP address to access the web configuration page is to check the UART debug text, while NET_DEBUG is defined in net.c, for example:

```
IP config:
ipaddr = 172.16.3.115
gwaddr = 172.16.2.5
netmask = 255.255.252.0
```

4.1 Default Configuration

The MAC address is obtained from the onboard 24AA02E48T-I/OT EEPROM during initialization (see 'net_init' in net.c). This memory is present on the MM900EVxA , excluding the MM900EV-LITE. The pre-programmed permanently protected MAC address resides at the last 6 bytes of memory in the EEPROM.

Note: The code should be changed if you are using hardware that does not have this external EEPROM.

4.1.1 Network

DHCP is turned on by default. The application will ask for an IP address, gateway and netmask from any available DHCP server and use these values to configure the network interface. It will also provide a hostname of "TelnetBridge" to the DHCP server.

4.1.2 UART

The following settings are used by default for the UART:

- 19200 baud
- 8 bits data
- 1 stop bit
- no parity

There is no option for flow control at present.

4.2 Modifying the Configuration

The configuration web page is split into 3 sections:

- a 'Status' section showing the IP address of a telnet client.
- a 'Network Configuration' section.
- a 'UART Configuration' section.

There are also shortcut links on the left to visit the FTDI web site, the FT90x section and the support information within the FTDI website.

4.2.1 Telnet Client

This section provides the IP address of the telnet client connected to the bridge. If no telnet client is connected then it will display "Connected: No Connection".

It is refreshed when the page is reloaded. The user can click on the Refresh button which is the same as refreshing the webpage (F5).

It does not permit any configurable functionality.

A telnet client like PuTTY can be used for testing this application example, as shown in Figure 4.2.

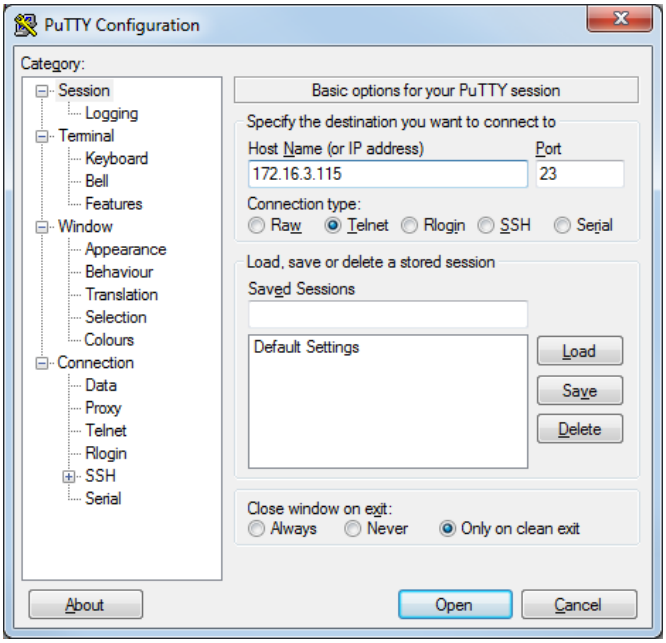

Figure 4.2 PuTTY Telnet Client

4.2.2 Network Configuration

The network configuration section has 2 sub-sections.

The first is the MAC address and hostname. These are required for configuring the device. As has been mentioned earlier the MAC address is required by Ethernet to be unique on a network. This is the hardware address used for all low-level communications on Ethernet. A network will act in an unspecified manner if this is duplicated.

It is recommended that bit 2 in the most significant byte (first octet transmitted) be set to 1 to indicate that the MAC address is locally administered.

The hostname field is sent to the DHCP server during the DHCP negotiation and may be used by the server to identify a device. In some DHCP setups it can be passed onto DNS to allow the hostname to be used seamlessly within a local network.

The third network option is to enable or disable DHCP. When enabled this will instruct the application to request an IP address, gateway address and netmask from a DHCP server. Therefore the fields that configure these values are disabled. The values obtained from the server are shown in the fields for information only.

If DHCP is disabled then the 3 fields are enabled allowing specific addresses to be configured.

There is no error checking that the values in the form are correctly formatted. The IP address, gateway and netmask are 4 decimal values separated by dots.

4.2.3 Programming

To program the Network Configuration and UART Configuration via the configuration webpage, make edits as required then click on the 'Apply' button, then click on restart on the next page. Note that the FT90x IC will be rebooted.

In Section 2.4 it is states that the configuration parameters are overwritten when programming new code to the FlashROM. This is because the FT90x Programming Utility will write the entire FlashROM when a new image is programmed. This will overwrite the datalogger partition with blank formatting.

However, when using the FT90x Programming Utility to configure one or many devices, a "Config File" can be specified and used to store preconfigured parameters into the datalogger partition.

The address of the partition will be 8 kB from the top of FlashROM (0x3E000).

An existing datalogger partition can be copied from an FT90x device using the "Data Log" tab in the FT90x Programming Utility. The binary file obtained can be modified to change any of the configurable parameters using the structs "`net_config`" and "`uart_config`" defined in the source code.

The network config is at an offset of 0x100 in the datalogger file and the UART config is at offset 0x200. Both structures have a 32-bit signature field to validate the values stored. The first and last 256 bytes of the datalogger file have required validation data, do not modify this.

4.2.3.1 Network Configuration

The network configuration (at offset 0x100 in the datalogger file) is stored as per the defined structure.

```
struct net_config {
        uint32_t key;
        char hostname[64];
};
```

The structure translates to the binary layout in Figure 4.3. The hostname is up to 64 characters long, must include a terminating NULL character and is coloured green. Only bytes from offset 0x100 to 0x180 are shown.

```
12 39 38 46 54 65 6C 6E 65 74 42 72 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figure 4.3 Network Configuration Layout

4.2.3.2 UART Configuration

The UART structure (at offset 0x200) consists of a 32-bit value for the baud rate, shown in red; the number of data bits in blue; the parity bits in green; and number of stop bits (0 = 1 stop bit) in purple.

```
struct uart_config {
        uint32_t key;
        int baud;
        uint8_t bits;
        uint8_t parity;
        uint8_t stop;
        uint8_t flow;
};
```

The flow control, orange, is not implemented. The remainder is unused.

The layout in the datalogger file is shown in Figure 4.4. Only bytes from offset 0x200 to 0x220 are shown.

```
13 29 38 49 00 4B 00 00 08 00 00 00 FF FF FF FF
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

Figure 4.4 UART Configuration Layout

5 Importing into the FT90x Toolchain

The AN_408 Firmware found at the following link can be easily imported into the FT90x Toolchain:

http://brtchip.com/wp-content/uploads/FT90x/AN408_Telnet_to_UART_Bridge.zip

Once installed, select File --> Import --> General --> Existing Projects into Eclipse, and point to the downloaded and extracted project directory.

The project will appear in Eclipse Project Explorer as shown in Figure 5.1.

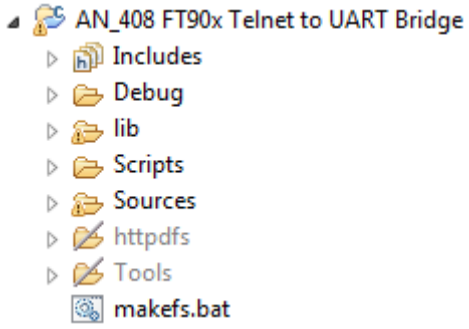

Figure 5.1 Eclipse Project Structure

5.1 Changing the Application Software

The application software provided can be altered and changed if required. The FT90x Toolchain is a free tool to enable code development and debug for the FT90x series and is based on plug-ins for the free popular IDE using the GCC compiler.

For example, the default UART configuration can be changed so a different default UART baud rate is used:

```
static struct uart_config default_uart_config = {
UART_CONFIG_KEY,
19200,
uart_data_bits_8,
uart_parity_none,
uart_stop_bits_1,
0};
```

With each software change, the project should be rebuilt and reprogrammed into the FT90x IC. Please refer to AN_325 FT900 Toolchain Installation Guide for further information.

6 Performance

The following throughput results were obtained. These values are subject to verification and change.

The measurement was made by timing data transmission (from first packet to last ACK packets) over the Ethernet. The small data sizes will therefore have a larger overhead as partial packets and delayed ACK packets will overly affect the data rates.

6.1 19200 baud

Data sample size 1720 bytes.

Telnet to UART -> 1433 bytes per second.

UART to Telnet -> 1433 bytes per second.

This is around 90% of the UART capacity.

6.2 115200 baud

Data sample size 1720 bytes.

Telnet to UART -> 3612 bytes per second.

UART to Telnet -> 3612 bytes per second.

This is around 37.5% of the UART capacity.

Larger data samples achieve higher UART utilizations but this is not recorded.

7 Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan , R.O.C.
Tel: +886 (2) 8797 5691
Fax: +886 (2) 8751 9737

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales) sales.emea@brtichip.com
E-mail (Support) support.emea@brtchip.com

Branch Office – Vietnam

Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel : 08 38453222
Fax : 08 38455222

E-mail (Sales) sales.apac@brtchip.com
E-mail (Support) support.apac@brtchip.com

Web Site

http://brtchip.com/

Distributor and Sales Representatives

Please visit the Sales Network page of the Bridgetek Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – References

Document References

FT90x Datasheet: http://brtchip.com/m-ft9/

lwIP http://savannah.nongnu.org/projects/lwip/

libtelnet https://github.com/seanmiddleditch/libtelnet

PuTTY http://www.chiark.greenend.org.uk/~sgtatham/putty/

tinyprintf http://www.sparetimelabs.com/tinyprintf/tinyprintf.php

AN_325 FT900 Toolchain Installation Guide

AN_365 FT900 API Programmers Manual

TN 157 Ethernet Explained

FT90x Toolchain

AN_408 Firmware:

http://brtchip.com/wp-content/uploads/FT90x/AN408_Telnet_to_UART_Bridge.zip

24AA02E48T-I/OT EEPROM http://www.microchip.com/wwwproducts/en/24AA02E48

MM900EVxA - http://brtchip.com/ft9xx-development-modules-landing/

Acronyms and Abbreviations

Terms	Description
CGI	Common Gateway Interface
DFU	Device Firmware Update
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
MAC	Media Access Controller
SSI	Server Side Include
USB	Universal Serial Bus
UART	Universal Asynchronous Receiver and Transmitter (Serial Port)
EEPROM	Electrically Erasable Programmable Read Only Memory
I2C	Inter-Integrated Circuit

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_408 FT90x Telnet to UART Bridge
Document Reference No.: BRT_000102
Clearance No.: BRT#060
Product Page: http://brtchip.com/m-ft9/
Document Feedback: Send Feedback

Revision	Changes	Date
1.0	Initial Release	2016-04-27
1.1	Updated Release	2017-02-16