



# Application Note

## AN\_360

# FT90x Example Applications

**Version 1.3**

**Issue Date: 2016-09-19**

This note describes the example applications provided for the FT90x and demonstrates the use of the FT90x Peripheral Driver Library.

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold FTDI harmless from any and all damages, claims, suits or expense resulting from such use.

**Future Technology Devices International Limited (FTDI)**

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

Web Site: <http://ftdichip.com>

Copyright © Future Technology Devices International Limited

## **Table of Contents**

<b>1</b>	<b>Introduction .....</b>	<b>5</b>
1.1	<b>Overview .....</b>	<b>5</b>
<b>2</b>	<b>Installation and Programming .....</b>	<b>6</b>
2.1	<b>FT90x Toolchain Installation .....</b>	<b>6</b>
2.2	<b>Importing the Examples .....</b>	<b>6</b>
2.3	<b>Building the FT900 Example Applications .....</b>	<b>6</b>
2.4	<b>Programming .....</b>	<b>6</b>
<b>3</b>	<b>Examples .....</b>	<b>8</b>
3.1	<b>ADC Examples .....</b>	<b>8</b>
3.1.1	ADC Example 1 .....	8
3.1.2	ADC Example 2 .....	9
3.2	<b>BCD Examples .....</b>	<b>10</b>
3.2.1	BCD Example 1 .....	10
3.3	<b>Camera Examples .....</b>	<b>11</b>
3.3.1	Camera Example 1 .....	11
3.4	<b>CAN Examples .....</b>	<b>12</b>
3.4.1	CAN Example 1 .....	12
3.4.2	CAN Example 2 .....	14
3.4.3	CAN Example 3 .....	14
3.5	<b>D2XX Examples .....</b>	<b>16</b>
3.5.1	D2XX Example 1 .....	16
3.6	<b>DAC Examples .....</b>	<b>18</b>
3.6.1	DAC Example 1 .....	18
3.6.2	DAC Example 2 .....	19
3.6.3	DAC Example 3 .....	19
3.7	<b>DLOG Example .....</b>	<b>21</b>
3.7.1	Purpose .....	21
3.7.2	Setup .....	21
3.7.3	Execution .....	21
3.8	<b>Ethernet Examples .....</b>	<b>24</b>

---

3.8.1	Ethernet Example 1 .....	24
<b>3.9</b>	<b>FreeRTOS Examples .....</b>	<b>27</b>
3.9.1	Setup (common for all FreeRTOS projects) .....	27
3.9.2	FreeRTOS Example 1 .....	29
3.9.3	FreeRTOS Example 2 .....	30
3.9.4	FreeRTOS Example 3 .....	32
3.9.5	FreeRTOS and lwIP Example .....	35
<b>3.10</b>	<b>GPIO Examples .....</b>	<b>40</b>
3.10.1	GPIO Example 1 .....	40
3.10.2	GPIO Example 2 .....	40
3.10.3	GPIO Example 3 .....	41
<b>3.11</b>	<b>I<sup>2</sup>C Master Examples .....</b>	<b>42</b>
3.11.1	I <sup>2</sup> C Master Example 1 .....	42
3.11.2	I <sup>2</sup> C Master Example 2 .....	43
<b>3.12</b>	<b>I<sup>2</sup>C Slave Examples .....</b>	<b>45</b>
3.12.1	I <sup>2</sup> C Slave Example 1 .....	45
<b>3.13</b>	<b>I<sup>2</sup>S Examples .....</b>	<b>47</b>
3.13.1	I <sup>2</sup> S Example 1 .....	47
3.13.2	I <sup>2</sup> S Example 2 .....	47
<b>3.14</b>	<b>PWM Examples .....</b>	<b>49</b>
3.14.1	PWM Example 1 .....	49
3.14.2	PWM Example 2 .....	49
3.14.3	PWM Example 3 .....	50
<b>3.15</b>	<b>Real Time Clock Examples .....</b>	<b>51</b>
3.15.1	RTC Example 1 .....	51
3.15.2	RTC Example 2 .....	51
<b>3.16</b>	<b>SD Host Examples .....</b>	<b>53</b>
3.16.1	SD Host Example 1 .....	53
<b>3.17</b>	<b>SPI Master Examples .....</b>	<b>55</b>
3.17.1	SPI Master Example 1 .....	55
3.17.2	SPI Master Example 2 .....	56
3.17.3	SPI Master Example 3 .....	58
<b>3.18</b>	<b>SPI Slave Examples .....</b>	<b>60</b>
3.18.1	SPI Slave Example 1 .....	60

---

<b>3.19</b>	<b>Timer Examples .....</b>	<b>62</b>
3.19.1	Timer Example 1 .....	62
3.19.2	Timer Example 2 .....	62
3.19.3	Timer Example 3 .....	63
<b>3.20</b>	<b>UART Examples .....</b>	<b>65</b>
3.20.1	UART Example 1 .....	65
3.20.2	UART Example 2 .....	65
3.20.3	UART Example 3 .....	66
<b>3.21</b>	<b>USB Device Examples .....</b>	<b>67</b>
3.21.1	GPIO DFU Example .....	67
3.21.2	USB D Example BOMS to SD Card .....	69
3.21.3	USB D Example HID .....	70
3.21.4	USB D Example HID Bridge .....	71
3.21.5	USB D Example CDCACM .....	72
3.21.6	USB D Example RNDIS .....	73
<b>3.22</b>	<b>USB Host Examples .....</b>	<b>75</b>
3.22.1	USB H_Example Hub .....	75
3.22.2	USB H Example HID .....	76
3.22.3	USB H Example CDCACM .....	77
3.22.4	USB H Example BOMS .....	78
3.22.5	USB H Example File System .....	79
3.22.6	USB H Example FT232 .....	81
3.22.7	AOA Examples .....	82
<b>3.23</b>	<b>Watchdog Timer Examples .....</b>	<b>90</b>
3.23.1	Watchdog Example 1 .....	90
<b>4</b>	<b>Contact Information .....</b>	<b>92</b>
<b>Appendix A</b>	<b>– References .....</b>	<b>93</b>
	<b>Document References .....</b>	<b>93</b>
	<b>Acronyms and Abbreviations .....</b>	<b>93</b>
<b>Appendix B</b>	<b>– List of Tables &amp; Figures .....</b>	<b>94</b>
	<b>List of Tables .....</b>	<b>94</b>
	<b>List of Figures .....</b>	<b>94</b>

---

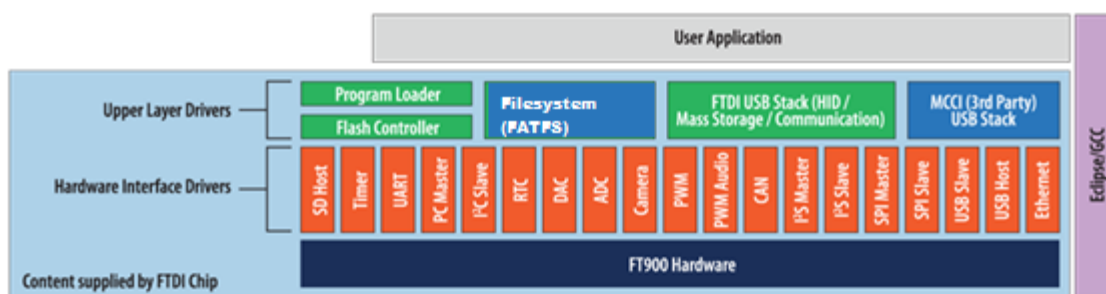


## **Appendix C – Revision History ..... 95**

## 1 Introduction

The FT90x Peripheral Driver Library (libft900) is a collection of 'C' language based functions that abstract away from the bare metal and presents the programmer with an easy to use API which will allow the programmer to develop applications quickly and easily.

Figure 1 shows the overall FT90x Interface driver support. This document focuses on the Hardware Interface Driver layer to show examples of usage. All drivers will be provided as source code for easy adaptation and modification.



**Figure 1: FT90x Interface driver support**

### 1.1 Overview

This document describes the construction and execution of the FT90x example programs and is intended to educate end users in the programming of the FT90x using FTDI's FT90x Driver Libraries in order to lower development time and deliver higher quality applications.

A number of examples are given for various FT90x Peripherals:

- Analogue to Digital Converter (ADC)
- Camera Interface
- CAN
- Digital to Analogue Converter (DAC)
- Ethernet
- General Purpose I/O (GPIO)
- I2C Master
- I2C Slave
- I2S
- Pulse Width Modulation (PWM)
- Real Time Clock
- SD Host
- SPI Master
- SPI Slave
- Timer
- Universal Asynchronous Receiver Transmitter (UART)
- Watchdog

Additional examples can be found here:

<http://www.ftdichip.com/Support/SoftwareExamples/FT90X.htm>

## 2 Installation and Programming

### 2.1 FT90x Toolchain Installation


Please refer to [AN\\_325\\_FT900\\_Toolchain\\_Installation\\_Guide](#) for instruction on how to install the FT90x GCC toolchain.

### 2.2 Importing the Examples

Within Eclipse, select File → Import → General → Existing Projects into Workspace. Browse to the FT90x examples directory (C:\Users\Username\Documents\FTDI\FT90x\version\Examples). Select and click OK. The projects should appear in the Project Explorer.

### 2.3 Building the FT900 Example Applications

To build the example applications:

- Right click the Project and click 'Build Project'.  
OR
- With the Project selected, click the Build Icon ().

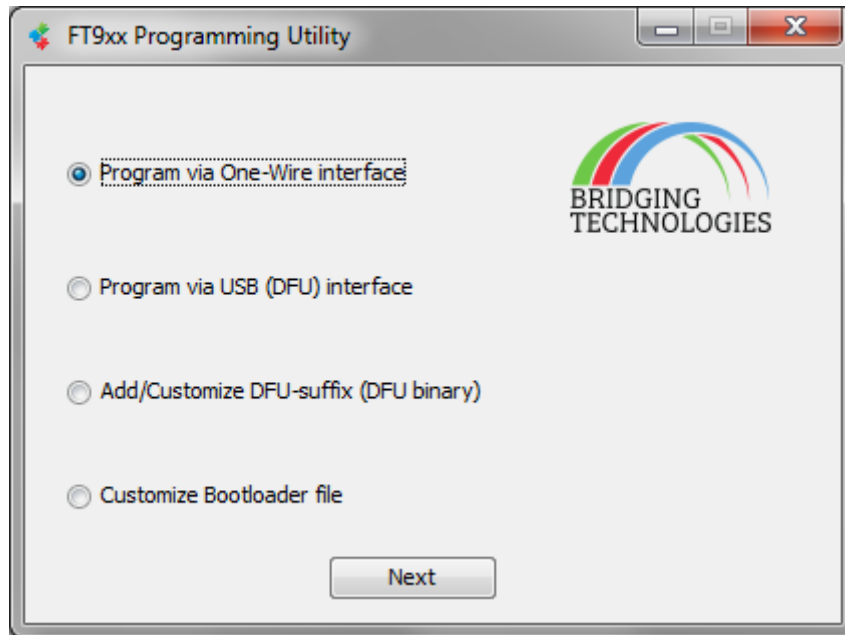
Note that you can also clean the project by right clicking on the project and selecting 'Clean'.

### 2.4 Programming

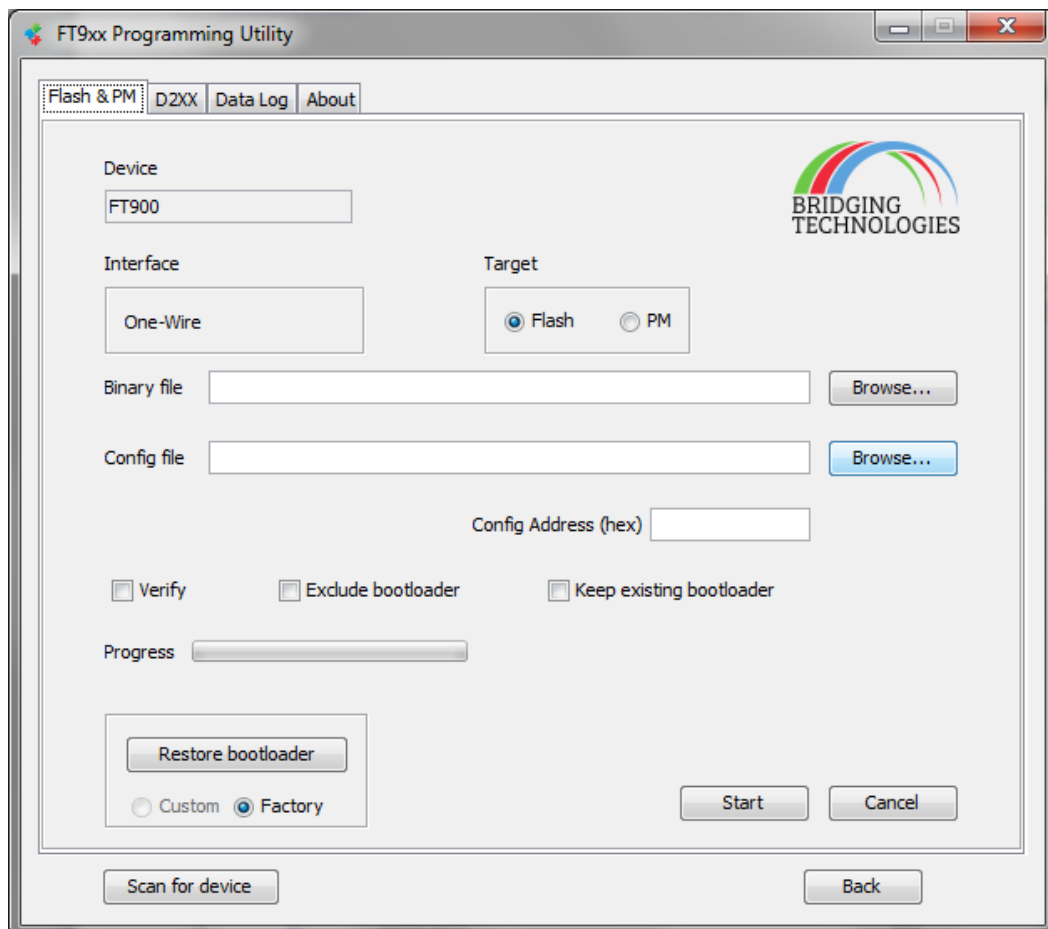
For every example there will be a corresponding binary file (e.g. uart\_example1.c will compile to uart\_example1.bin). Note that an .elf file is also created but this should not be used for programming as this used for debugging purposes.

To program a specific application into the memory of an FT90x device, create an 'External Tool Configuration' via the Run menu within Eclipse. Please see [AN\\_325\\_FT900\\_Toolchain\\_Installation\\_Guide](#) for details.

Alternatively you can use FTDI's free FT9xx Programming Utility shown in Figure 2 and Figure 3. This can be accessed via the 'FT900 Programming Utility' shortcut created on the desktop or under the 'FTDI Utilities' menu in Eclipse.



**Figure 2: FT90x Programming Utility Start Screen**



**Figure 3: FT90x Programming Utility One-Wire Option**



## 3 Examples

Some of the examples require the user to connect a USB to Serial converter to UART0 as this port is used to send and receive text. The terminal settings should be 19200 baud rate, unless otherwise a specific baudrate is mentioned in the example. Rest of the settings is 8 data bits, no parity and 1 stop bit. Flow control is not enabled.

Minimum UART0 connections are:

- UART0\_TXD/GPIO48: available via CN3 Pin 4 on FT90x EVM
- UART0\_RXD/GPIO49: available via CN3 Pin 6 on FT90x EVM

FTDI have a range of suitable cables available like the TTL-232R-3V3 available from <http://www.ftdichip.com/Products/Cables.htm>.

The terminal program used in these tests was PuTTY. However, any VT100 compatible terminal emulator can be used.

Additionally, some examples require the use of a Bus Pirate ([http://dangerousprototypes.com/docs/Bus\\_Pirate](http://dangerousprototypes.com/docs/Bus_Pirate)) in order to provide stimulus.

### 3.1 ADC Examples

#### 3.1.1 ADC Example 1

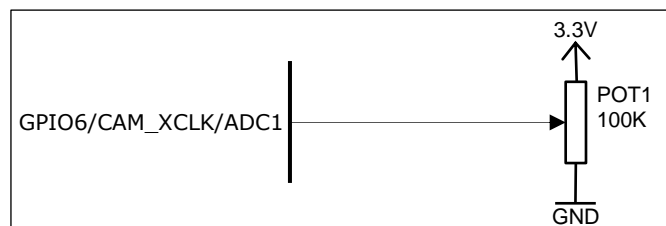
##### 3.1.1.1 Purpose

The purpose of `adc_example1.c` is to continuously poll the ADC for a new value and display it to the user.

##### 3.1.1.2 Setup

Connect as shown in Figure 4

1. Connect a voltage source to the *GPIO6/CAM\_XCLK/ADC1* pin which can be found on connector CN3 Pin 30 of the FT90x EVM. This could be a potentiometer with the ends connected to 3.3V and GND, with the wiper connected to *GPIO6/CAM\_XCLK/ADC1*.



**Figure 4: Circuit Diagram for ADC Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

### 3.1.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to ADC Example 1...

Poll ADC 1 continuously.
-----
ADC 1 = 0000
```

2. Apply a voltage between 0 and 3.3V to *GPIO6/CAM\_XCLK/ADC1*. This should cause the value to change. For example,
- 3.

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to ADC Example 1...

Poll ADC 1 continuously.
-----
ADC 1 = 0123
```

Note that 1023 (0x3FF) is the maximum value since this is a 10 bit ADC.

## 3.1.2 ADC Example 2

### 3.1.2.1 Purpose

The purpose of `adc_example2.c` is to use an interrupt to update a variable with the latest reading from the ADC and display it to the user.

### 3.1.2.2 Setup

Refer to 3.1.1.2 Setup Section.

### 3.1.2.1 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to ADC Example 2...

Use an interrupt to capture the current value of ADC 1 and print it out.
-----
ADC 1 = 0000
```

2. Apply a voltage between 0 and 3.3V to *GPIO6/CAM\_XCLK/ADC1*. This should cause the value to change. For example,
- 3.

```
ADC 1 = 0123
```

Note that 1023 (0x3FF) is the maximum value since this is a 10 bit ADC.

## 3.2 BCD Examples

The USB ports on personal computers are convenient places for USB devices such as the MM900EVx module to draw power. This convenience has led to the creation of USB chargers that simply expose a USB standard-A receptacle. This allows USB devices to use the same USB cable to either be powered from either a personal computer or from a USB charger.

The various charging ports that **BCD Device mode** in FT900 can detect are:

- **Standard Downstream Port (SDP)** – Typically found in desktop and laptop computers. The USB devices will be enumerated to be USB compliant. This type of port can supply maximum of 500mA only when configured for that current.
- **Dedicated Charging Port (DCP)** – Power sources like AC adapters and Auto/Car adapters that do not enumerate so that powering can occur with no digital communication at all.
- **Charging Downstream Port (CDP)** – Battery Charging Specification 1.1 defines this new higher current USB port for PCs, laptops and other hardware. This type of port can supply upto 1.5A before enumeration.

### 3.2.1 BCD Example 1

#### 3.2.1.1 Purpose

The purpose of the example is to display the type of charging port that the FT900 device is connected to – a SDP or a DCP or a CDP port.

#### 3.2.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text. Connect the FT900 USB device port to a SDP port of USB host, or connect to DCP ports of a USB power sources like Wall Warts, auto adapters and power banks.

**Note:** The MM900EVx module should be powered via CN2 (JST connector) for the example to operate correctly.

#### 3.2.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to BCD Example..

Displays the charging port that the USB device is connected to...
-----
```

2. The type of charging port detected is displayed. When connected to USB host like laptops or desktops, following is displayed

```
SDP mode found
```

Or as

```
DCP mode found
```

when connected to power sources like power bank or wall warts.

## 3.3 Camera Examples

### 3.3.1 Camera Example 1

#### 3.3.1.1 Purpose

The purpose of this example is to demonstrate how to interface an 8-bit camera with the FT90x. The MM900EV2A / MM900EV3A come with an Omni vision OV9655 Color Camera.

#### 3.3.1.2 Setup

Connect an Omni vision OV9655 or OV7670 camera to the camera interface of the FT90x if not already connected to CN13 or CN14 on the FT90x EVM.

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.3.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Camera Example 1...

Get a frame from an OV7670 and print it out to the console.
-----
```

2. A countdown should appear to allow the camera to auto adjust,

```
3...2...1...
```

3. The size of the resulting image will be displayed followed by an ASCII art output of the captured frame,

```
36 by 62 ASCII Art
```



## 3.4 CAN Examples

### 3.4.1 CAN Example 1

#### 3.4.1.1 Purpose

The purpose of this example is to transmit CAN messages between CAN0 and CAN1 and poll for the response.

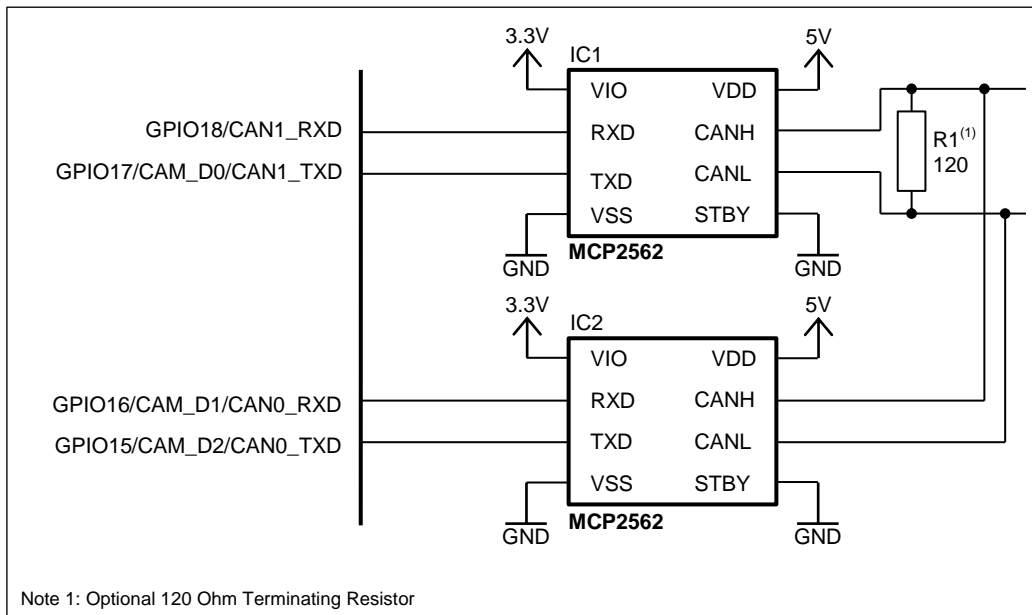
#### 3.4.1.2 Setup

Connect CAN0 and CAN1 together through CAN transceivers (e.g., Microchip MCP2562) to allow for each CAN interface to send messages to each other.

Note that the FT90x EVM hardware does not have CAN transceivers onboard so these need to be externally connected.

Connect the following as shown in Figure 5

1. Connect *GPIO18/CAN1\_RXD* (found via connector CN3 Pin 40 on the FT90x EVM) to the *RXD* pin of IC1.
2. Connect *GPIO17/CAM\_D0/CAN1\_TXD* (found via connector CN3 Pin 38 on the FT90x EVM) to the *TXD* pin of IC1.
3. Connect *GPIO16/CAM\_D1/CAN0\_RXD* (found via connector CN3 Pin 37 on the FT90x EVM) to the *RXD* pin of IC2.
4. Connect *GPIO15/CAM\_D2/CAN0\_TXD* (found via connector CN3 Pin 36 on the FT90x EVM) to the *TXD* pin of IC2.
5. Connect the *CANH* pins of IC1 and IC2.
6. Connect the *CANL* pins of IC1 and IC2.
7. Connect the *VIO* pins of IC1 and IC2 to 3.3V.
8. Connect the *VDD* pins of IC1 and IC2 to 5V.
9. Connect the *VSS* and *STBY* pins of IC1 and IC2 to *GND*.
10. (Optionally) Connect a 120Ω Resistor between *CANH* and *CANL*



**Figure 5: Circuit diagram for CAN Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

### 3.4.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to CAN Example 1...

Send and Receive messages between CAN0 and CAN1 and poll the response.
-----
```

2. Messages should begin to be transmitted between CAN0 and CAN1,

```
CAN0 TX-> ID=____0x123      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
CAN1 RX<- ID=____0x123      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
CAN1 TX-> ID=____0x123      {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01}
CAN0 RX<- ID=____0x123      {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01}
```

Errors can occur when running this example. If an error occurs, a prompt will output the current error code in a readable format. For example:

```
Error whilst Transmitting :
RX_WRN: Receive Warning. The number of receive errors is >= 96
TX_WRN: Transmit Warning. The number of transmit errors is >= 96
ACK_ERR: Acknowledge Error Occurred
FRM_ERR: Form Error Occurred
CRC_ERR: CRC Error Occurred
STF_ERR: Stuff Error Occurred
BIT_ERR: Bit Error Occurred
```

## 3.4.2 CAN Example 2

### 3.4.2.1 Purpose

The purpose of this example is to transmit CAN messages from CAN0 and show that they can be filtered on CAN1.

### 3.4.2.2 Setup

Refer to 3.4.1.2 Setup Section.

### 3.4.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to CAN Example 2...

Filter CAN messages arriving at CAN1.
-----
```

2. Messages should begin to be transmitted between CAN0 and CAN1,

```
Transmitting 50 unwanted messages
Transmitting 1 wanted messages
There is 1 message available on CAN1
CAN1 RX-> ID= 0x123 {0x48,0x45,0x4c,0x4f,0x57,0x52,0x4c,0x44}
```

## 3.4.3 CAN Example 3

### 3.4.3.1 Purpose

The purpose of this example is to show how CAN messages can be received and processed using interrupts.

### 3.4.3.2 Setup

Refer to 3.4.1.2 Setup Section.

### 3.4.3.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to CAN Example 3...

Receive messages via an interrupt on CAN1.
-----
```

2. Messages should begin to be transmitted between CAN0 and CAN1,

```
CAN0 TX-> ID=_____0x123      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
CAN1 RX<- ID=_____0x123      {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

CAN0 TX-> ID=_____0x123      {0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00}
CAN1 RX<- ID=_____0x123      {0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x00}

CAN0 TX-> ID=_____0x123      {0x02,0x01,0x00,0x00,0x00,0x00,0x00,0x00}
CAN1 RX<- ID=_____0x123      {0x02,0x01,0x00,0x00,0x00,0x00,0x00,0x00}

CAN0 TX-> ID=_____0x123      {0x03,0x02,0x01,0x00,0x00,0x00,0x00,0x00}
CAN1 RX<- ID=_____0x123      {0x03,0x02,0x01,0x00,0x00,0x00,0x00,0x00}
```



## 3.5 D2XX Examples

### 3.5.1 D2XX Example 1

#### 3.5.1.1 Purpose

The purpose of this example is to demonstrate that FT900 device brings itself as a FTDI D2XX device to the host PC and the data is sent back and forth on the D2XX channel, between a terminal PC application and the User Firmware application.

#### 3.5.1.2 Setup

Connect the FT900 development board programmed with D2XX\_Example1.bin to the host PC via USB. Install the drivers required. The default VID and PID combination is included in FTDI driver release 2.12.14 and greater. See [Drivers](#) and [Installation Guides](#) for further information.

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

Open up terminal PC application program for UART0 with following port setting 19200 baud, no parity, 8 data bits, and 1 stop bit.

#### 3.5.1.3 Execution

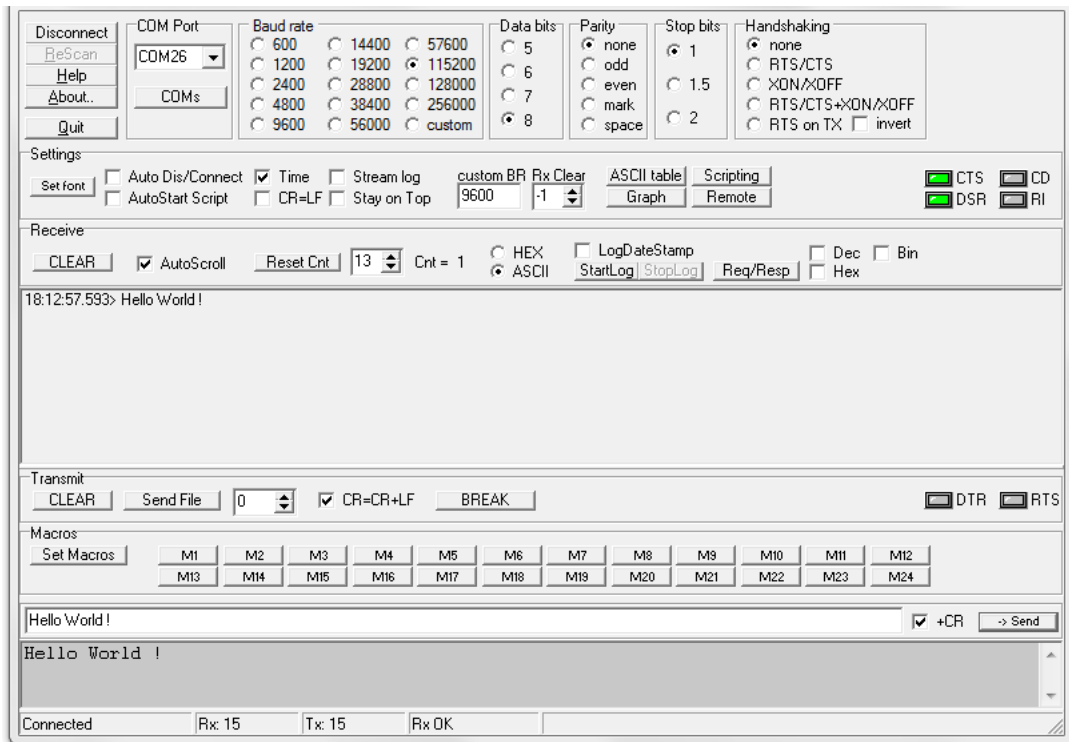
1. A welcome message should appear like so,

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to D2XX Example 1...

Enter any text on the D2XX port, the same is echoed back on the same port...
-----
D2xx_loopback_test1: D2XX_Init() called, Result: 0

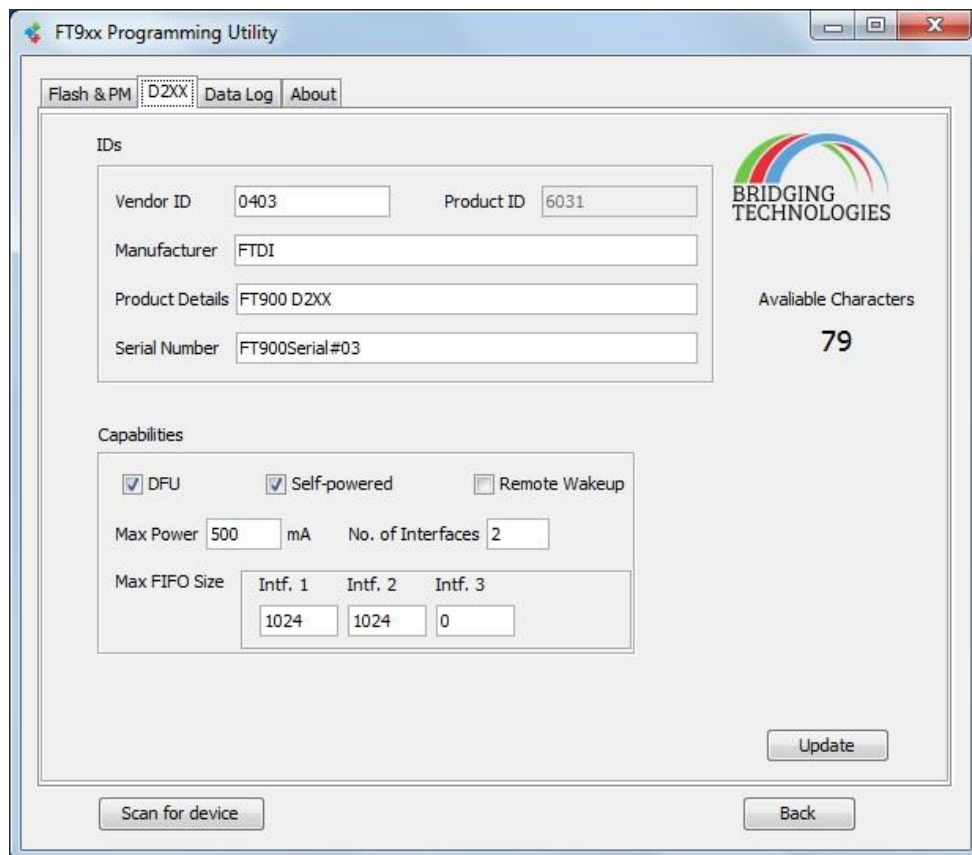
D2xx_loopback_test1: D2XX_EVT_READY is received
D2xx_loopback_test1: D2XX_EVT_RESET is received
```

2. When the host D2XX drivers are installed and D2XX interfaces are detected. This should cause the USB serial ports corresponding to the D2XX channels to appear.
3. Open the serial port corresponding to the D2XX channel in the terminal application. Enter some text and the same text is received on the USB serial port.



**Figure 6: D2XX Port opened in the PC Terminal application**

The default settings can be changed using the FT900 Programming GUI Utility's D2XX tab:



**Figure 7: FT900 Programming Utility D2XX Tab**

## 3.6 DAC Examples

### 3.6.1 DAC Example 1

#### 3.6.1.1 Purpose

The purpose of this example is to demonstrate the DAC operating in a single shot mode.

#### 3.6.1.2 Setup

Connect *GPIO14/CAM\_D3/DAC0* (found via connector CN3 Pin 35 on the FT90x EVM) to an oscilloscope. Also connect GND.

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

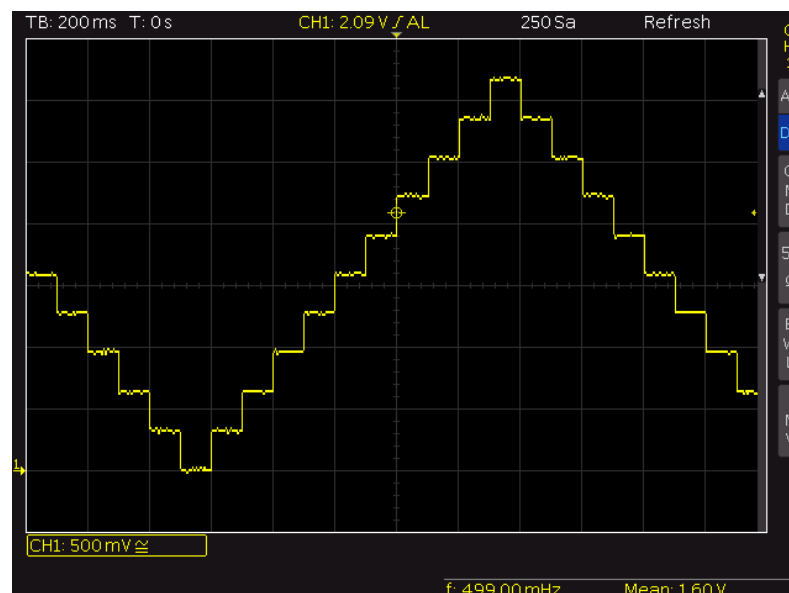
#### 3.6.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to DAC Example 1..

Cycle through a series of values outputting them to the DAC in
single shot mode.
The values will be output on DAC0
-----
```

2. An analogue wave should appear on *GPIO14/CAM\_D3/DAC0* as shown in Figure 8.



**Figure 8: Output from dac\_example1.c**

## 3.6.2 DAC Example 2

### 3.6.2.1 Purpose

The purpose of this example is to demonstrate the DAC operating in a continuous mode, and polling the DAC to see if new data is needed.

### 3.6.2.2 Setup

Refer to 3.6.1.2 Setup Section.

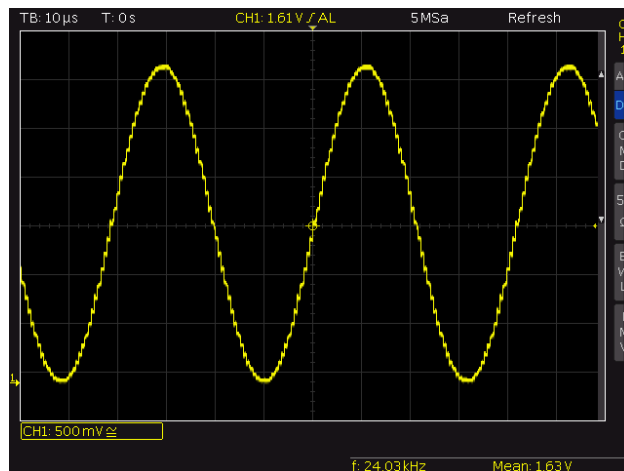
### 3.6.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to DAC Example 2...

Output a series of values in continuous mode by polling the DAC.
The values will be output on DAC0
-----
```

2. A 24 kHz Sine wave should appear on *GPIO14/CAM\_D3/DAC0* as shown in Figure 9.



**Figure 9: Output from dac\_example2.c**

## 3.6.3 DAC Example 3

### 3.6.3.1 Purpose

The purpose of this example is to demonstrate the DAC operating in a continuous mode, and using an interrupt to supply new data.

### 3.6.3.2 Setup

Refer to 3.6.1.2 Setup Section.

### 3.6.3.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to DAC Example 3...

Output a series of values in continuous mode by interrupt.
The values will be output on DAC0
-----
```

2. An analogue wave should appear on *GPIO14/CAM\_D3/DAC0* as shown in Figure 9.

## 3.7 DLOG Example

### 3.7.1 Purpose

This example program demonstrates use of the datalogger APIs. A 4KB sector is allocated as the datalog partition. One sector has 16 pages and page 0 is reserved by the lib. Pages 1 to 15 are available for use by the user. Once a page is programmed, that page may not be overwritten. Pages may not be erased individually and the entire sector has to be erased. Therefore, users shall ensure that pages are completely filled before programming into the datalogger. APIs are provided to read, program and erase. Page management is left to the user application requirements.

### 3.7.2 Setup

Connect the FT900 development board programmed with DLOG\_Example1.bin to the host PC via USB.

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

Open up terminal PC application program for UART0 with following port setting 19200 baud, no parity, 8 data bits, and 1 stop bit.

### 3.7.3 Execution

1. A welcome message should appear like so,

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to the Datalogger Example ...
This example will erase the datalogger partition and fill all pages
from 1 to 14 (14 pages) with repeated values of 0x00 to 0x0D and end.
-----
```

2. Details will appear showing the dlog partition address in flash, page size and the number of pages in the sector used by dlogger.

```
_dlog_partition: 0003E000
dlog_init: passed, pgsz=0x100, pages=14
```

3. The example then erases each page from 1 to 14 and fills each page with the repeated values of 1 to 14.

```
0x00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x80: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x90: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xa0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xb0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xc0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0xf0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0x00: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x10: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x20: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x30: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x40: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x50: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x60: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x70: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x80: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0x90: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xa0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xb0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xc0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xd0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xe0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
0xf0: 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01

0x00: 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
0x10: 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
0x20: 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
...
...
0xd0: 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c
0xe0: 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c
0xf0: 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c 0c

0x00: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x10: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x20: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x30: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x40: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x50: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x60: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x70: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x80: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0x90: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xa0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xb0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xc0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xd0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xe0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
0xf0: 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d 0d
```

4. Then the example ends.

```
program ended
```



## 3.8 Ethernet Examples

### 3.8.1 Ethernet Example 1

#### 3.8.1.1 Purpose

The purpose of this example is to demonstrate the operating of the Ethernet module, implementing ARP and ICMP Echo support in order to allow the user to "Ping" the device.

#### 3.8.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect an Ethernet cable to the FT90x's Ethernet Port, connecting the other end into a network or directly into PC with a crossover cable or into an Ethernet Port which supports Auto MDI-X. Alternatively, connect both a PC and an FT90x to an Ethernet switch or hub.

If connecting the MM900 directly to a PC, the PC should be configured for Static IP as described in 3.9.5.2.1

#### 3.8.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Ethernet Example 1..

Allow the user to "Ping" (ICMP Echo) to the device.
-----
```

2. Details will appear showing the MAC address<sup>1</sup> and IP address of the device.

```
MAC address = 02:F7:D1:00:00:01
IP address = 192.168.1.55
```

3. The program will wait until the Ethernet cable is plugged in.

```
Please plug in your Ethernet cable
Ethernet Link Up
```

4. Standard network traffic will occur. The FT90x will report when ARP packets are received on its Ethernet Interface and when it sends a response.

```
Got an ARP Packet
Sending Reply ARP
Got an ARP Packet
Sending Reply ARP
Got an ARP Packet
Sending Reply ARP
```

5. On a PC on the same network, "ping" the FT90x:

On Windows:

```
ping 192.168.1.55
```

---

<sup>1</sup> This MAC Address is one within the Locally Administered address set and should not clash with any Global MAC Address that is assigned to other hardware. If you are running more than one Ethernet Example, make sure that they have different MAC Addresses (e.g. 02:F7:D1:00:00:02).

On Linux:

```
ping -c 4 192.168.1.55
```

6. The FT90x will report when ICMP packets arrive at the Ethernet Interface. The “ping” program uses ICMP Echo and Echo Replies in order to determine if a device is present on the network and the time it takes for that device to respond:

```
Got an ICMP Packet
Sending ICMP Echo Reply
Got an ICMP Packet
Sending ICMP Echo Reply
Got an ICMP Packet
Sending ICMP Echo Reply
Got an ICMP Packet
Sending ICMP Echo Reply
```

7. On a PC on the same network running the “ping” program, the program should be reporting successful responses:

On Windows:

```
> ping 192.168.1.55

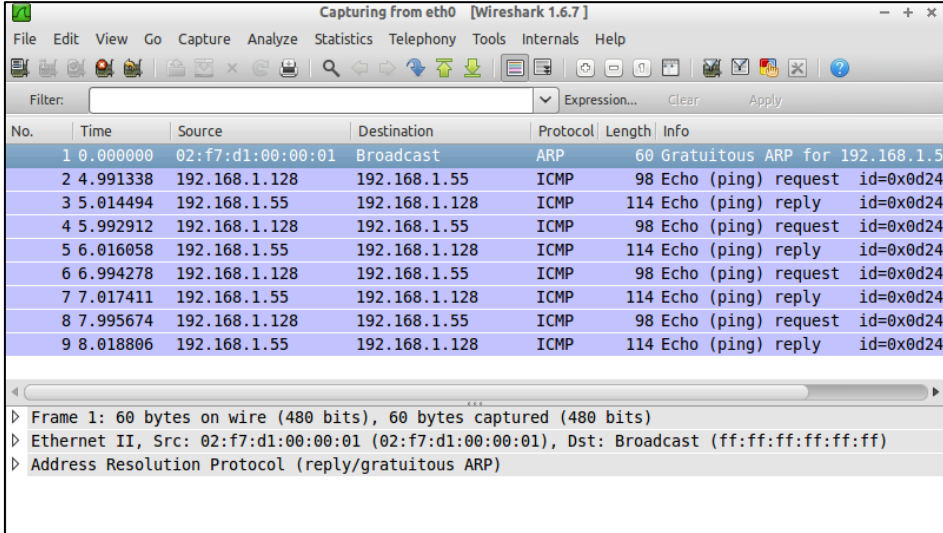
Pinging 192.168.1.55 with 32 bytes of data:
Reply from 192.168.1.55: bytes=32 time=23ms TTL=64
Reply from 192.168.1.55: bytes=32 time=23ms TTL=64
Reply from 192.168.1.55: bytes=32 time=23ms TTL=64
Reply from 192.168.1.55: bytes=32 time=23ms TTL=64

Ping statistics for 192.168.1.55:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 23ms, Maximum = 23ms, Average = 23ms
```

On Linux:

```
$ ping -c 4 192.168.1.55
PING 192.168.1.55 (192.168.1.55) 56(84) bytes of data.
64 bytes from 192.168.1.55: icmp_req=1 ttl=64 time=23.2 ms
64 bytes from 192.168.1.55: icmp_req=2 ttl=64 time=23.1 ms
64 bytes from 192.168.1.55: icmp_req=3 ttl=64 time=23.1 ms
64 bytes from 192.168.1.55: icmp_req=4 ttl=64 time=23.1 ms
--- 192.168.1.55 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 23.176/23.188/23.208/0.152 ms
```

A network analyzer tool like Wireshark (a free and open-source packet analyzer) can be used to look at the raw network traffic travelling between the PC and the FT90x, as shown in Figure 10.



Capturing from eth0 [Wireshark 1.6.7]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	02:f7:d1:00:00:01	Broadcast	ARP	60	Gratuitous ARP for 192.168.1.5
2	4.991338	192.168.1.128	192.168.1.55	ICMP	98	Echo (ping) request id=0x0d24
3	5.014494	192.168.1.55	192.168.1.128	ICMP	114	Echo (ping) reply id=0x0d24
4	5.992912	192.168.1.128	192.168.1.55	ICMP	98	Echo (ping) request id=0x0d24
5	6.016058	192.168.1.55	192.168.1.128	ICMP	114	Echo (ping) reply id=0x0d24
6	6.994278	192.168.1.128	192.168.1.55	ICMP	98	Echo (ping) request id=0x0d24
7	7.017411	192.168.1.55	192.168.1.128	ICMP	114	Echo (ping) reply id=0x0d24
8	7.995674	192.168.1.128	192.168.1.55	ICMP	98	Echo (ping) request id=0x0d24
9	8.018806	192.168.1.55	192.168.1.128	ICMP	114	Echo (ping) reply id=0x0d24

▶ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)  
 ▶ Ethernet II, Src: 02:f7:d1:00:00:01 (02:f7:d1:00:00:01), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
 ▶ Address Resolution Protocol (reply/gratuitous ARP)

**Figure 10: Wireshark output for eth\_example1.c**

## 3.9 FreeRTOS Examples

FreeRTOS (<http://www.freertos.org/>) is a popular real-time operating system with a modified GPL license allowing it to be used freely in commercial applications. It supports multiple scheduling strategies for tasks like pre-emptive, cooperative and time-slicing. Queues, events, semaphores and mutexes are also available for inter task communication and synchronization. On FT900 the FreeRTOS port consumes about 9KB of Flash (under **-Os** optimization). More information on FreeRTOS is available on the FreeRTOS website.

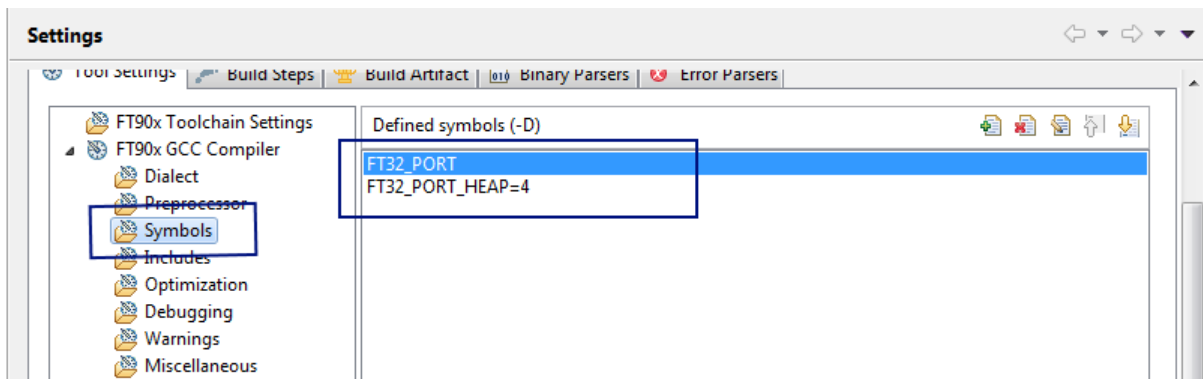
### 3.9.1 Setup (common for all FreeRTOS projects)

To use the FT900 FreeRTOS port in projects, some special configurations/includes in Eclipse are required that are different from the usual example projects. These are already set in the example projects but are summarized next.

To access these in Eclipse, go to Project → Properties → C/C++ Build → Settings.

1. Symbols **FT32\_PORT** and **FT32\_PORT\_HEAP=4**.

The user can set **FT32\_PORT\_HEAP** to 1, 2 or 3 if he or she wishes to use Heap management strategies 1/2/3 available from FreeRTOS. Since strategy 4 is the most flexible, it has been selected for the default FT900 port. For more information on the Heap management strategies available with FreeRTOS refer <http://www.freertos.org/a00111.html>

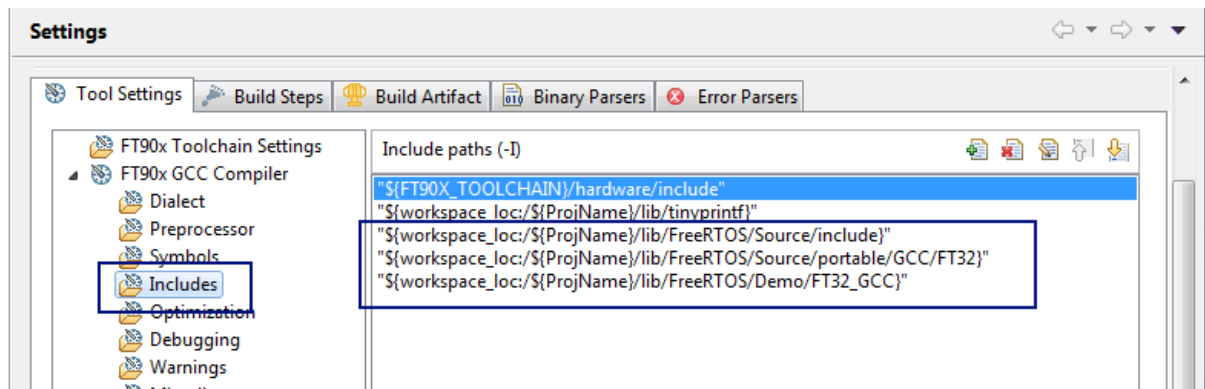


2. Set the include paths to various directories under the FreeRTOS folder structure as shown below. Key directories are:

**FreeRTOS\Source** - The source code for the FreeRTOS kernel.

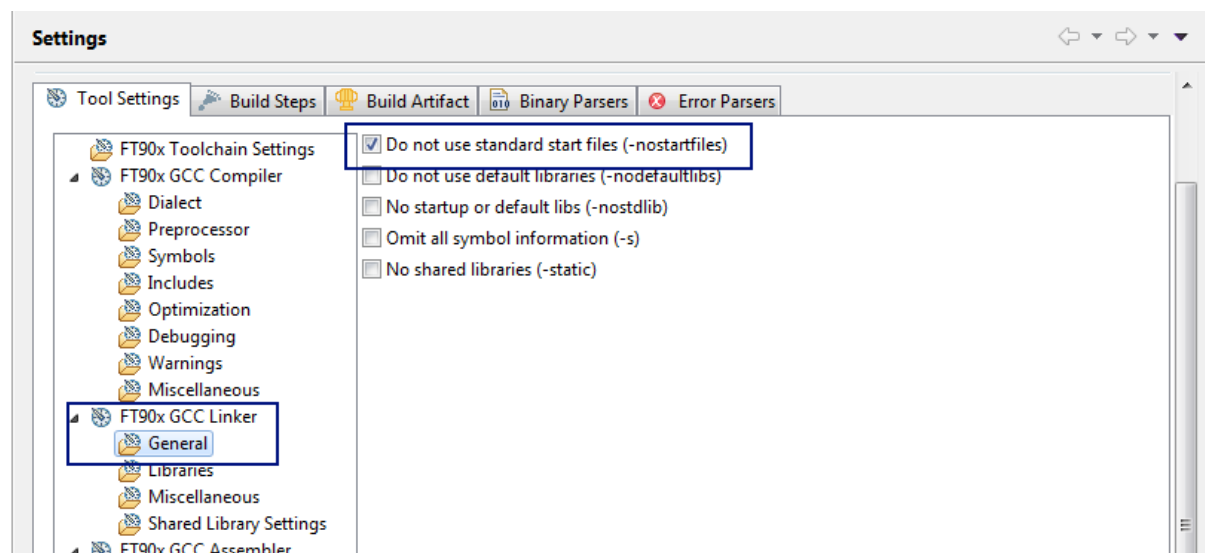
**FreeRTOS\Source\portable\GCC\FT32** - Files specific to FT900 port.

**FreeRTOS\Demo\FT32\_GCC** - The crt0 (C runtime zero) and configuration file for the port.



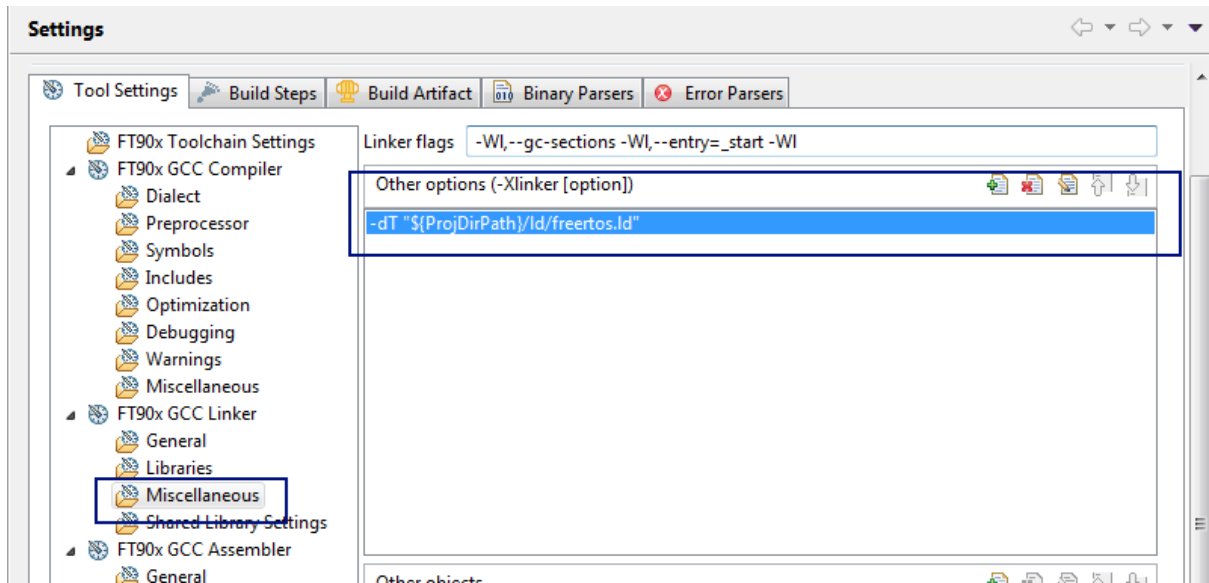
3. Select the **-nostartfiles** linker option.

The FreeRTOS port uses a custom crt0 available at *FreeRTOS\Demo\FT32\_GCC\crt0.S*

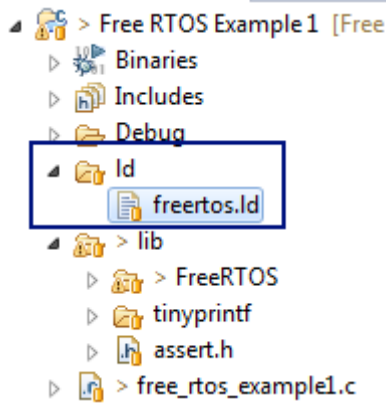


4. Set the linker options to use the custom linker script: **-dT**  
**"\${ProjDirPath}/ld/freertos.ld"**

The FreeRTOS port uses a custom linker script (available at **ld/freertos.ld**)



The linker script is available in the example project:



## 3.9.2 FreeRTOS Example 1

### 3.9.2.1 Purpose

The purpose of this example is to test the FreeRTOS port by running a series of tasks doing mathematical operations that are interrupted by the pre-emptive scheduler. It also illustrates the use of Queues for passing data from the interrupt context to tasks, along with context switching from ISRs. The examples are ported from the demos available within the FreeRTOS distribution.

### 3.9.2.2 Setup

Connect the UART1 RX and TX lines together to generate a loopback on UART1. These are pins **GPIO52** and **GPIO53** (CN3 pins 7 and 9 on the MM900EVx).

Open a COM port on the PC to UART0 on the MM900 board to view the logs printed out from FT900. The baud rate to be configured is **230400bps**

### 3.9.2.3 Execution

1. A welcome message should appear like so:

```
Copyright 2014-2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS Test Example 1...

Communication tasks which will send and receive data over UART1 using FreeRTOS
Queues
Please ensure that UART1 RX and TX lines are connected to each other.
-----
```

2. This message will be followed by lines indicating the successful creation of 8 IntMath and Math tasks and the COMRx and Tx tasks like so:

```
C IntMath1 ec8 e40 acc
C IntMath2 1340 12b8 f44
C IntMath3 17b8 1730 13bc
C IntMath4 1c30 1ba8 1834
C IntMath5 20a8 2020 1cac
C IntMath6 2520 2498 2124
C IntMath7 2998 2910 259c
C IntMath8 2e10 2d88 2a14
C Math1 3688 3600 2e8c
C Math2 3f00 3e78 3704
C Math3 4778 46f0 3f7c
C Math4 4ff0 4f68 47f4
C Math5 5868 57e0 506c
C Math6 60e0 6058 58e4
C Math7 6958 68d0 615c
C Math8 71d0 7148 69d4
Setup of UART1 Complete !C COMTx 7904 787c 7508
C COMRx 7d7c 7cf4 7980
C Check 81f4 816c 7df8
C IDLE 866c 85e4 8270
C Tmr Svc 8be4 8b5c 87e8
COM Rx task started.
C MEM_CHECK 905c 8fd4 8c60
COM Tx task started.
```

3. Following this the logs will indicate the running of the memory check task (every 3 seconds) like so:

```
C MEM_CHECK c3f4 c36c bff8
C MEM_CHECK 905c 8fd4 8c60
C MEM_CHECK 905c 8fd4 8c60
C MEM_CHECK c3f4 c36c bff8
C MEM_CHECK 905c 8fd4 8c60
```

4. Additional logs will be printed in case any of the running tests fails. If there are no logs other than the creation of the MEM\_CHECK task, it means that the tests are running successfully.

### 3.9.3 FreeRTOS Example 2

#### 3.9.3.1 Purpose

The purpose of this example is to test the FreeRTOS port by running a series of tasks that cover the use of Semaphores, Queues, Events and Dynamic Priority assignment to tasks. The code is ported from the demos available within the FreeRTOS distribution.

#### 3.9.3.2 Setup

Open a COM port on the PC to UART0 on the MM900 board to view the logs printed out from FT900. The baud rate to be configured is **230400bps**.

### 3.9.3.3 Execution

1. A welcome message should appear like so:

```
Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS Test Example 2...

Test use of Semaphores, Queues, Events, Dynamic Priority Assignment
-----
```

2. This message will be followed by lines indicating the successful creation of the various test tasks like so:

```
C PolSEM1 f88 f00 b8c
C PolSEM2 1400 1378 1004
C BlkSEM1 18f4 186c 14f8
C BlkSEM2 1d6c 1ce4 1970
C QConsNB 2258 21d0 1e5c
C QProdNB 26d0 2648 22d4
C CONT_INC 2bac 2b24 27b0
C LIM_INC 3024 2f9c 2c28
C C_CTRL 349c 3414 30a0
C SUSP_SEND 3914 388c 3518
C SUSP_RECV 3d8c 3d04 3990
C 1st_P_CHANGE 4204 417c 3e08
C 2nd_P_CHANGE 467c 45f4 4280
C QConsB1 4b7c 4af4 4780
C QProdB2 4ff4 4f6c 4bf8
C QProdB3 54f4 546c 50f8
C QConsB4 596c 58e4 5570
C QProdB5 5e74 5dec 5a78
C QConsB6 62ec 6264 5ef0
C EvntCTRL 67d0 6748 63d4
C Event0 6c48 6bc0 684c
C Event1 70c0 7038 6cc4
C Event2 7538 74b0 713c
C Event3 79b0 7928 75b4
C Check 7e28 7da0 7a2c
C IDLE 82a0 8218 7ea4
C Tmr Svc 8818 8790 841c
```

3. Following this the logs will indicate the running of the memory check task (every 3 seconds) like so:

```
C MEM_CHECK 9108 9080 8d0c
C MEM_CHECK 8c90 8c08 8894
C MEM_CHECK 9108 9080 8d0c
C MEM_CHECK 9108 9080 8d0c
C MEM_CHECK 8c90 8c08 8894
C MEM_CHECK 9108 9080 8d0c
C MEM_CHECK 8c90 8c08 8894
```

4. Additional logs will be printed in case any of the running tests fails. If there are no logs other than the creation of the MEM\_CHECK task, it means that the tests are running successfully.



### 3.9.4 FreeRTOS Example 3

#### 3.9.4.1 Purpose

The purpose of this example is to give a simple illustration of time-slicing and pre-emptive scheduling and using a mutex to synchronize access.

#### 3.9.4.2 Setup

Open a COM port on the PC to UART0 on the MM900 board to view the logs printed out from FT900. The baud rate to be configured is **230400bps**.

The example contains 3 demos that can be compiled by changing the **FRT\_DEMO** preprocessor define switch to 1, 2 or 3, found in `free_rtos_example3.c`. The behavior of each demo is:

**Demo 1** - Illustrates time-slicing by creating tasks of the same priority. FreeRTOS will try to give all three of them equal execution time.

**Demo 2** - 3 Tasks of different priorities (3, 2, 1) are created. The task with priority 2 runs constantly, never yielding. The task with priority 3 (highest priority) prints its name and yields every 500mS. The net result is that Task 2 runs constantly, while being interrupted by Task 1 every 500mS. Task 3 never gets to run.

**Demo 3** - 4 Tasks are created with different priorities. Each of the tasks prints lines onto UART0. Since the tasks have different priorities, preemption will occur and a mutex is used to synchronize access to UART0, keeping the printed strings uninterrupted.

#### 3.9.4.3 Execution

##### 3.9.4.3.1 Demo 1

1. A welcome message will appear like so:

```
Copyright 2014-2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS Test Example 3...

Demonstrate FreeRTOS Time-slicing
-----
```

2. This message will be followed by lines indicating the successful creation of the various test tasks like so:

```
C Task 1 15fc 1574 660
C Task 2 2614 258c 1678
C Task 3 362c 35a4 2690
C IDLE 3aa4 3a1c 36a8
C Tmr Svc 401c 3f94 3c20
```

3. Once the tasks have been created their names will be printed out equally on average like so:

```
Task1
Task2
Task3
Task1
Task2
Task3
Task1
Task2
Task3
Task1
Task2
```

- This indicates that all three tasks get an opportunity to run.

### 3.9.4.3.2 Demo 2

- A welcome message will appear like so:

```
Copyright 2014-2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS Test Example 3...

Demonstrate FreeRTOS Task Priority handling
-----
```

- This message will be followed by lines indicating the successful creation of the various test tasks like so:

```
Demo 2
C Task 1 1610 1588 674
C Task 2 2628 25a0 168c
C Task 3 3640 35b8 26a4
C IDLE 3ab8 3a30 36bc
C Tmr Svc 4030 3fa8 3c34
```

- Once the tasks have been created Task1 will be scheduled every 500mS and Task 2 will run in the intervening time like so:

```
*Task1*
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
Task2
*Task1*
```

- Task 3 is of priority 1 and never gets to run.

### 3.9.4.3.3 Demo 3

- A welcome message will appear like so:

```
Copyright 2014-2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS Test Example 3...

Demonstrate FreeRTOS mutex based synchronization
-----
```

- This message will be followed by lines indicating the successful creation of the various test tasks like so:

```
Demo 3
C Print1 1b60 1ad8 bc4
C Print2 2b78 2af0 1bdc
C Print3 3b90 3b08 2bf4
C Print4 4ba8 4b20 3c0c
C IDLE 5020 4f98 4c24
C Tmr Svc 5598 5510 519c
```

- Once the tasks have been created their names messages are printed out without breaks like so:

```
Task 3 #####
Task 4 ~~~~~
Task 2 -----
Task 1 *****
Task 3 #####
Task 1 *****
Task 4 ~~~~~
Task 4 ~~~~~
Task 2 -----
Task 1 *****
Task 2 -----
Task 4 ~~~~~
Task 4 ~~~~~
Task 3 #####
Task 2 -----
Task 2 -----
Task 1 *****
Task 4 ~~~~~
Task 2 -----
Task 3 #####
```

- This indicates that all three tasks get an opportunity to run.

### 3.9.5 FreeRTOS and lwIP Example

Light Weight IP (lwIP) is an open TCP/IP stack suitable for use in small embedded systems on account of its low resource footprint. For instance the lwIP stack compiled for the current example (with TCP/IP, UDP and DHCP enabled) consumes about 64kB Program Memory and about 4K static Data Memory. Additional memory required for TCP/IP buffers is allocated dynamically (these configurations are specified in the file `ports/v3/include/lwipopts.h`). lwIP may be used with or without an OS. More details and the latest source code for lwIP can be found on the [project website](#).

#### 3.9.5.1 Purpose

The purpose of this example is to demonstrate the usage of the lwIP stack integrated with FreeRTOS. The example contains two demos – one with FT900 running a **TCP Server** and the other with FT900 running a **TCP Client**. Both demos can be configured to use either **Static IP** or **Dynamic IP** (DHCP). Two companion python scripts are provided in the `/Scripts` directory which can be run from a PC to test the demos.

#### 3.9.5.2 Setup

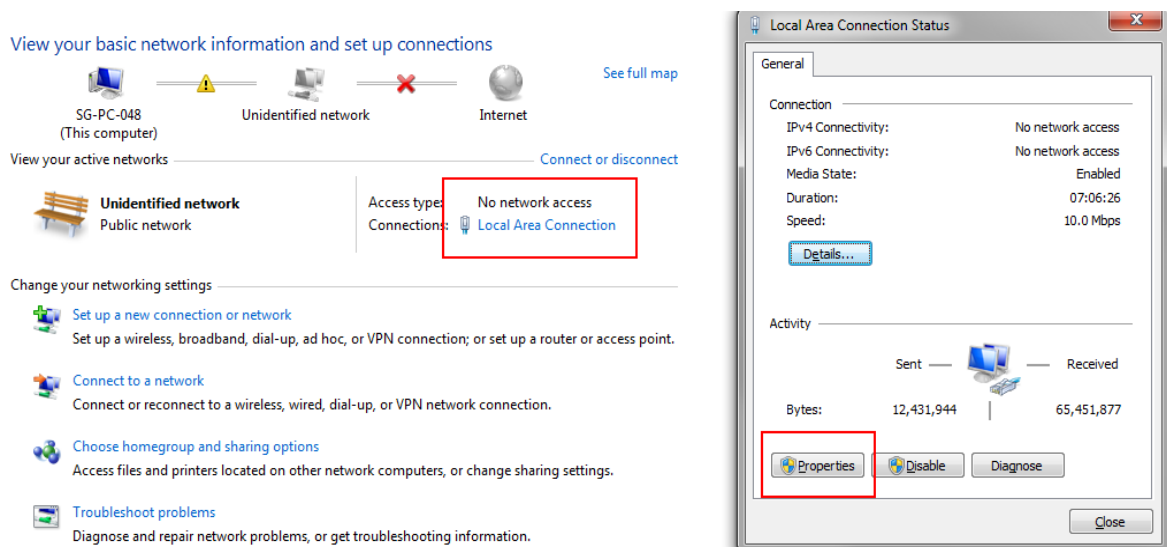
The example source code contains two demos which are switched using the `DEMO_TYPE` preprocessor switch. A value of `SERVER` selects the Server demo where FT900 acts as a TCP Server and `CLIENT` selects the Client demo where FT900 acts as a TCP Client. Furthermore the demos can be configured to use either Static or Dynamic IP using the preprocessor switch `USE_DHCP`, a value of 1 selects Dynamic IP and a value of 0 selects Static IP. Default configuration is a Server demo with Static IP.

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

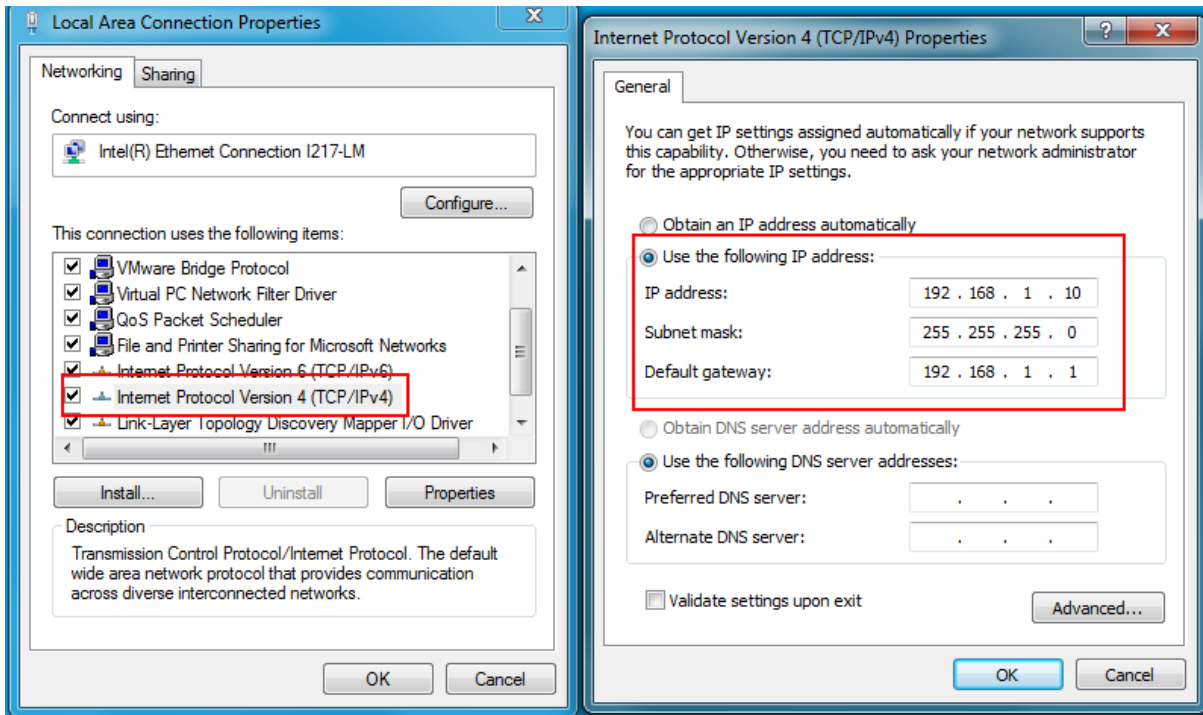
##### 3.9.5.2.1 Setup for Static IP configuration (`USE_DHCP == 0`)

Connect an Ethernet Cable between the FT90x EVM board and a host PC.

Configure the Host PC to have a Static IP of **192.168.1.10** as shown in Figure 12. On Windows 7, the LAN Connection properties can be found in Control Panel > Network and Internet > Network and Sharing Center as shown in Figure 11



**Figure 11: Windows 7 LAN Properties**

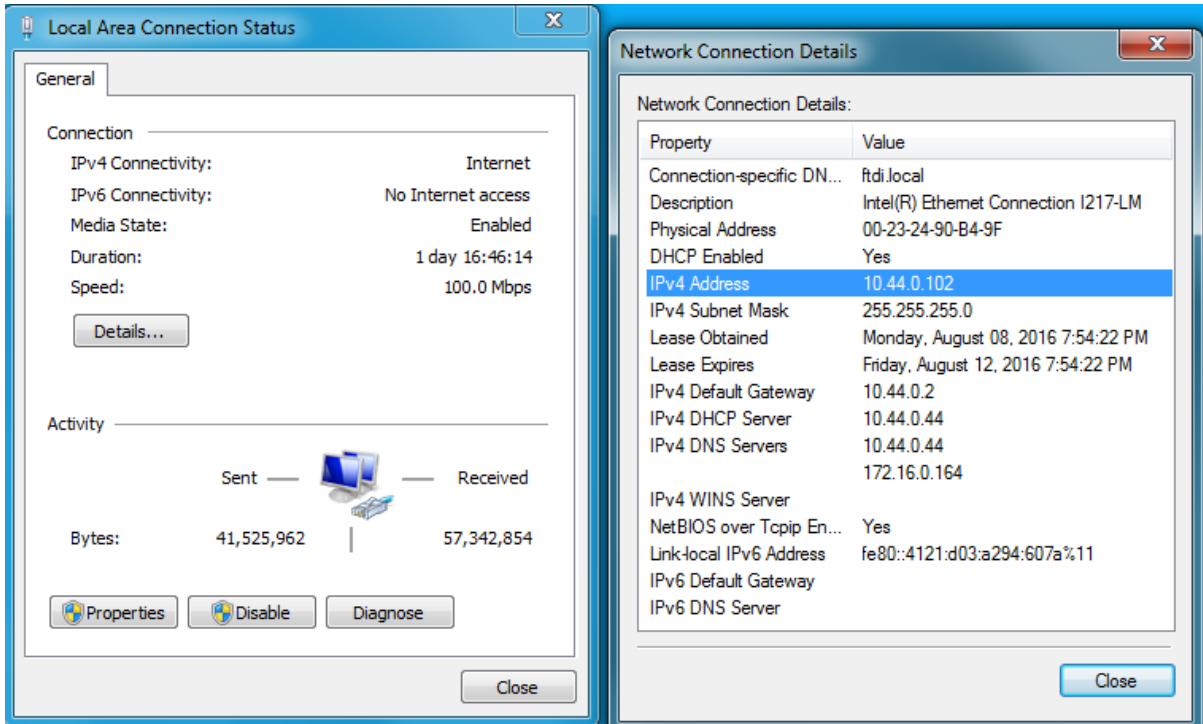


**Figure 12: Host PC Static IP Configuration**

If a different IP address is used, the example source code (for the Client demo) must be updated accordingly.

### 3.9.5.2.2 Setup for Dynamic IP Configuration (USE\_DHCP == 1)

Connect both the Host PC and FT900 Ethernet ports to the same Local Area Network (LAN). Find the IP address of the Host PC and update it in the example source code. This is shown below.



**Figure 13: Windows 7 - Find Host PC IP address**

```
#if DEMO_TYPE == CLIENT
#define IP_ADDR_SERVER "10.44.0.102" // The "server" should run elsewhere, eg: on a PC
```

**Figure 14: Update Host PC IP address**

### 3.9.5.3 Execution

#### 3.9.5.3.1 Server Demo

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS LWIP Example...

Demonstrate a TCP Server using the LWIP Stack running on FreeRTOS
-----
```

2. The TCP server IP address and listening port is displayed in the logs, make a note of it:

```
Server 192.168.1.190:80
```

3. On the Host PC run the python script /Scripts/simple\_client.py from a command prompt passing the appropriate FT900 TCP Server IP address as a parameter. For a Static IP configuration the IP should be **192.168.1.190** and for a Dynamic IP configuration, the address printed in the UART logs must be used. An example is shown in Figure 15 (for a Dynamic IP configuration)

```
$ python simple_client.py 192.168.1.190
```

4. The messages "Hello Client" and "Hello Server" should appear repeatedly at the Host PC and FT900 side respectively.

```
process_server[0]
Hello server
[00]Terminated on end-of-string

process_server[1]
Hello server
[00]Terminated on end-of-string

process_server[2]
Hello server
[00]Terminated on end-of-string

process_server[3]
Hello server
[00]Terminated on end-of-string

process_server[4]
Hello server
[00]Terminated on end-of-string
```

```

C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\FreeRTOS\lwIP\Example\Scripts>python simple_client.py 10.44.0.130
Connecting to 10.44.0.130:80
Received 15 bytes
Hello client
=====

Connecting to 10.44.0.130:80
Received 15 bytes
Hello client
=====

Connecting to 10.44.0.130:80
Received 15 bytes
Hello client
=====

Connecting to 10.44.0.130:80
Received 15 bytes
Hello client
=====
    
```

**Figure 15: Simple TCP Client running on Host PC (FT900 dynamic IP is 10.44.0.120)**

### 3.9.5.3.2 Client Demo

1. A welcome message should appear like so:

```

(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Free RTOS LWIP Test Example...

Demonstrate a TCP Client using the LWIP Stack running on FreeRTOS
-----
    
```

2. On the Host PC run the python script Scripts/simple\_server.py by opening it in the Python editor IDLE and pressing the F5 key. The script is configured to run a TCP server listening on PORT **9990**. Confirm that the example source code has been updated with the correct Server IP address – for Static IP configuration the address should be **192.168.1.10** and for Dynamic IP the appropriate Host PC IP address should be used (refer section 3.9.5.2). Note that firewalls running on the Host PC might block all incoming connections, it would be best to disable the firewall when testing.
3. The messages “Hello Client” and “Hello Server” should appear repeatedly at the FT900 and Host PC side respectively.

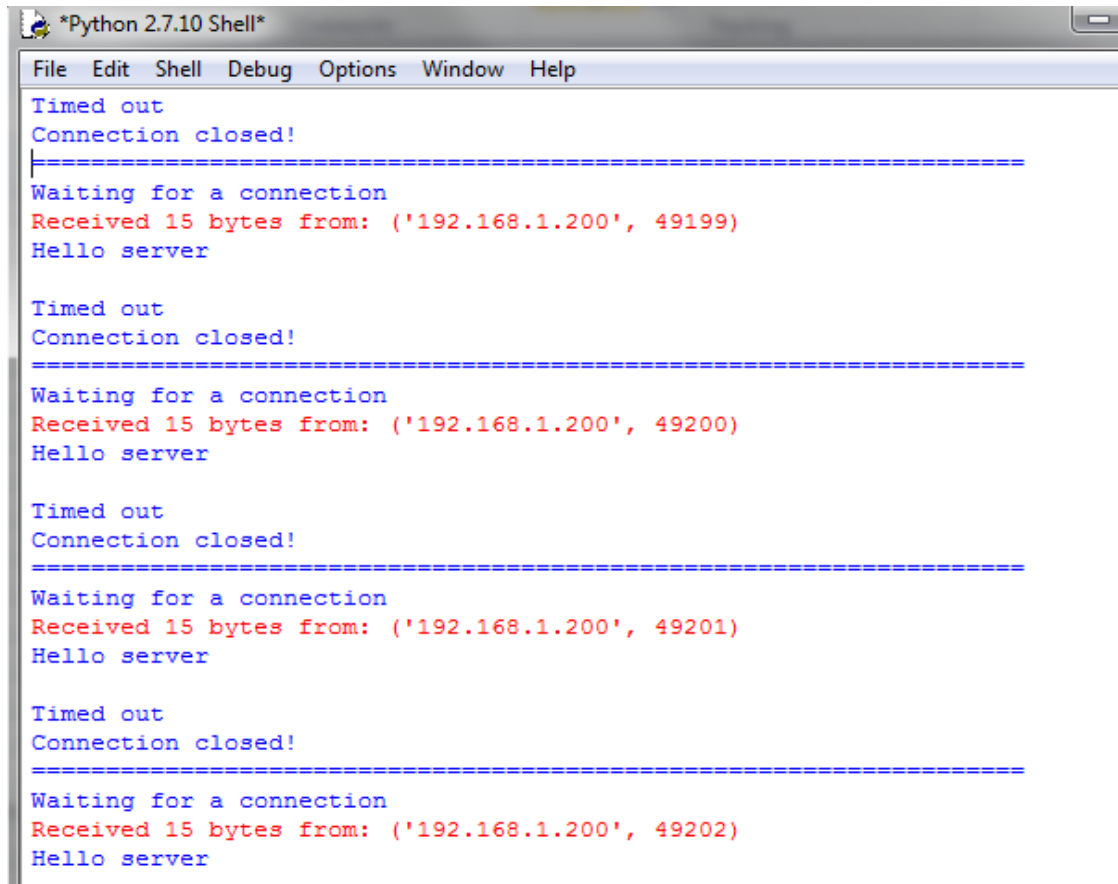
```

process_client[0]
Sock 0
Hello client
[00]Terminated on end-of-string

process_client[1]
Sock 0
Hello client
[00]Terminated on end-of-string

process_client[2]
Sock 0
Hello client
[00]Terminated on end-of-string

process_client[3]
Sock 0
Hello client
[00]Terminated on end-of-string
    
```



```
*Python 2.7.10 Shell*
File Edit Shell Debug Options Window Help
Timed out
Connection closed!
=====
Waiting for a connection
Received 15 bytes from: ('192.168.1.200', 49199)
Hello server

Timed out
Connection closed!
=====
Waiting for a connection
Received 15 bytes from: ('192.168.1.200', 49200)
Hello server

Timed out
Connection closed!
=====
Waiting for a connection
Received 15 bytes from: ('192.168.1.200', 49201)
Hello server

Timed out
Connection closed!
=====
Waiting for a connection
Received 15 bytes from: ('192.168.1.200', 49202)
Hello server
```

**Figure 16: Simple TCP server running on a Host PC**



## 3.10 GPIO Examples

### 3.10.1 GPIO Example 1

#### 3.10.1.1 Purpose

The purpose of this example is to demonstrate using GPIO functions.

#### 3.10.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect something to *GPIO18/CAN1\_RXD* (found via connector CN3 Pin 40 on the FT90x EVM) to monitor the state of the pin (e.g. an LED).

#### 3.10.1.3 Execution

5. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to GPIO Example 1..
```

```
Toggle a pin on and off.  
-----
```

6. *GPIO18/CAN1\_RXD* will toggle on and off every second.

### 3.10.2 GPIO Example 2

#### 3.10.2.1 Purpose

The purpose of this example is to demonstrate using GPIO pins.

#### 3.10.2.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect something to *GPIO18/CAN1\_RXD* (found via connector CN3 Pin 40 on the FT90x EVM) to change the state of the pin.

#### 3.10.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to GPIO Example 2..
```

```
Read the value of a pin.  
-----
```

2. The current state of *GPIO18/CAN1\_RXD* will be reported:

```
Pin is High
```

### 3.10.3 GPIO Example 3

#### 3.10.3.1 Purpose

The purpose of this example is to demonstrate using GPIO pins.

#### 3.10.3.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect something to *GPIO18/CAN1\_RXD* (found via connector CN3 Pin 40 on the FT90x EVM) to change the state of the pin.

#### 3.10.3.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to GPIO Example 2..
Use interrupts to inform the user of a falling edge on a GPIO pin.
-----
```

2. Changing the state of *GPIO18/CAN1\_RXD* from High to Low will cause the pin to be interrupted, which will display the message:

```
Pin Interrupted!
```

## 3.11 I<sup>2</sup>C Master Examples

### 3.11.1 I<sup>2</sup>C Master Example 1

#### 3.11.1.1 Purpose

The purpose of this example is to demonstrate the use of the I<sup>2</sup>C Master Peripheral and how to transfer data to and from it.

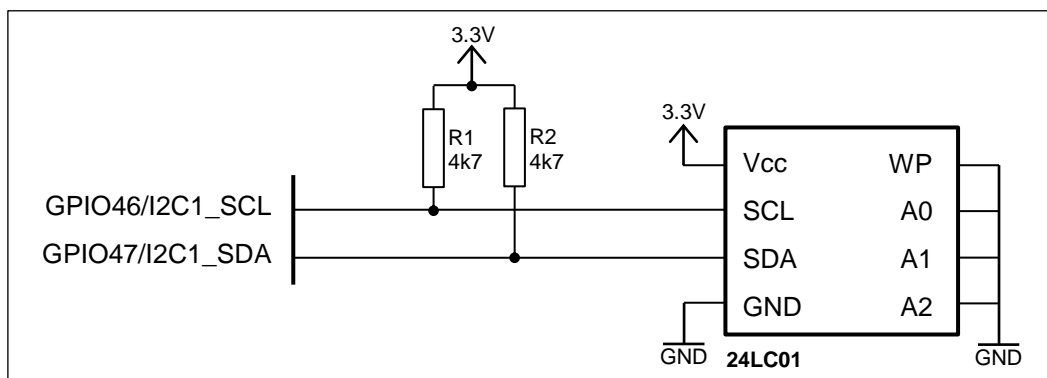
#### 3.11.1.2 Setup

This test uses a 24LC01 1Kbit EEPROM as an I<sup>2</sup>C device.

Connect as shown in Figure 17

Connect the *SCL* pin of the 24LC01 to the *GPIO46/I2C1\_SCL* found via connector CN3 Pin 25 on the FT90x EVM.

1. Connect the *SDA* pin of the 24LC01 to the *GPIO47/I2C1\_SDA* found via connector CN3 Pin 25 on the FT90x EVM.
2. Connect the *WP*, *A0*, *A1*, *A2* and *GND* pins of the 24LC01 to Ground.



**Figure 17: Circuit Diagram for I2C Master Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.11.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to I2C Master Example 1..
```

```
Read and write to an I2C EEPROM (24LC01)  
-----
```

2. The program will start by dumping the contents of EEPROM:

```

Reading all 128 bytes of EEPROM
0x0000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
  
```

- The program will then set all locations in EEPROM to FF<sub>h</sub> then dump the contents of EEPROM:

```

Setting the EEPROM to 0xFF
0x0000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
  
```

- The program will then set all even locations to 01<sub>h</sub> then dump the contents of EEPROM:

```

Set all even numbered locations to 0x01
0x0000: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0010: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0020: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0030: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0040: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0050: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0060: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0070: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
  
```

- The program will then fill EEPROM with a block of example text and dump the contents of EEPROM:

```

Filling the EEPROM with example text
0x0000: 4C 6F 72 65 6D 20 69 70 73 75 6D 20 64 6F 6C 6F | Lorem ipsum dolo
0x0010: 72 20 73 69 74 20 61 6D 65 74 2C 20 63 6F 6E 73 | r sit amet, cons
0x0020: 65 63 74 65 74 75 72 20 61 64 69 70 69 73 63 69 | ectetur adipisci
0x0030: 6E 67 20 65 6C 69 74 2E 20 41 6C 69 71 75 61 6D | ng elit. Aliquam
0x0040: 20 69 6E 74 65 72 64 75 6D 20 65 72 6F 73 20 73 | interdum eros s
0x0050: 69 74 20 61 6D 65 74 20 6C 6F 72 65 6D 20 70 75 | it amet lorem pu
0x0060: 6C 76 69 6E 61 72 2C 20 76 65 6C 20 70 6F 73 75 | lvinar, vel posu
0x0070: 65 72 65 20 6C 65 6F 20 70 6F 73 75 65 72 65 2E | ere leo posuere.
  
```

### 3.11.2 I<sup>2</sup>C Master Example 2

#### 3.11.2.1 Purpose

The purpose of this example is to demonstrate the use of the I<sup>2</sup>C Master Peripheral and how to transfer data to and from it.

#### 3.11.2.2 Setup

This test uses the on-board MAC address EEPROM (24AA02E48T). No external setup is required.

### 3.11.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to I2C Master Example 2..

Read and write to the on-board MAC address EEPROM (24AA02E48T)
-----
```

2. The program will start by reading the current content of the EEPROM:

```
Reading all 16 bytes of EEPROM
0x0000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
```

3. The program will then write some new data to the EEPROM:

```
Setting the EEPROM to 0xBB
0x0000: BB BB BB BB BB BB BB BB BB BB BB BB BB BB BB | .....
```

## 3.12 I<sup>2</sup>C Slave Examples

### 3.12.1 I<sup>2</sup>C Slave Example 1

#### 3.12.1.1 Purpose

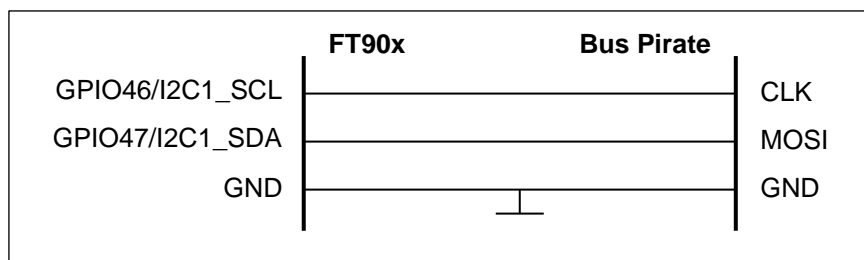
The purpose of this example is to demonstrate the use of the I<sup>2</sup>C Slave Peripheral and how to transfer data to and from it.

#### 3.12.1.2 Setup

This test is best carried out using a Bus Pirate which is an open hardware tool to program and interface with communication buses, available to buy from multiple sources online.

Connect the following as shown in Figure 18

1. Connect the *CLK* pin of the Bus Pirate to the *GPIO46/I2C1\_SCL* found via connector CN3 Pin 25 on the FT90x EVM.
2. Connect the *MOSI* pin of the Bus Pirate to the *GPIO47/I2C1\_SDA* found via connector CN3 Pin 26 on the FT90x EVM.
3. Connect the *GND* pin of the Bus Pirate to the *GND* pin of the FT90x



**Figure 18: Circuit Diagram for I2C Slave Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.12.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to I2C Slave Example 1...

Have a block of memory act as registers on an I2C bus.
Read Address = 0x39, Write Address = 0x38
-----
```

This is followed by instructions on screen for quick reference.

2. On the Bus Pirate, enter I<sup>2</sup>C mode:

```
>m4
(1)>3
```

3. On the Bus Pirate, to write data to the FT90x I<sup>2</sup>C Slave, see the following commands and return data:

```
I2C>[ 0x38
I2C START BIT
WRITE: 0x38 ACK
I2C>4
WRITE: 0x04 ACK
I2C>0xA5
WRITE: 0xA5 ACK
I2C>]
I2C STOP BIT
```

- The '[' character will cause a start condition occur.
- The write address is sent on the I<sup>2</sup>C bus (38<sub>h</sub>).
- The address pointer is sent on the I<sup>2</sup>C bus (4).
- Data is written on the I2C bus which loads data in starting from the given address pointer (i.e. Location 0 = 1, Location 1 = 2, Location 2 = 3, Location 3 = 4).
- The ']' character will cause a stop condition to occur.

4. On the Bus Pirate, to read data from the FT90x I<sup>2</sup>C Slave, execute the following

```
I2C>[ 0x38
I2C START BIT
WRITE: 0x38 ACK
I2C>4
WRITE: 0x04 ACK
I2C>[
I2C START BIT
I2C>0x39
WRITE: 0x39 ACK
I2C>r
READ: 0xA5
I2C>]
NACK
I2C STOP BIT
```

- The '[' character will cause a start condition occur.
- The write address is sent on the I<sup>2</sup>C bus (38<sub>h</sub>).
- The address pointer is sent on the I<sup>2</sup>C bus (4).
- The '[' character will cause a restart condition to occur.
- The read address is sent on the I<sup>2</sup>C bus (39<sub>h</sub>).
- The 'r' character will cause a byte to be read from the slave device and return the result.
- The ']' character will cause a stop condition to occur.

## 3.13 I<sup>2</sup>S Examples

### 3.13.1 I<sup>2</sup>S Example 1

#### 3.13.1.1 Purpose

The purpose of this example is to demonstrate I<sup>2</sup>S transmitting a block of data held in RAM to the Wolfram Codec. The FT90x EVM is fitted with a Wolfram WM8731 Codec.

#### 3.13.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect a set of speakers to CN10 and CN11 of the FT90x EVM, or connect a set of headphones to CN9.

#### 3.13.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to I2S Example 1...

Play a Fs/64 (44100/64 = 689) Hertz Sine Wave using a Wolfram WM8731.
-----
```

2. A 689 Hz Sine wave will play on the output of the Codec.

### 3.13.2 I<sup>2</sup>S Example 2

#### 3.13.2.1 Purpose

The purpose of this example is to demonstrate the receiving and transmitting data over I<sup>2</sup>S. The FT90x EVM is fitted with a Wolfram WM8731 Codec.

#### 3.13.2.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect a set of speakers to CN10 and CN11 of the FT90x EVM, or connect a set of headphones to CN9.

#### 3.13.2.3 Execution

1. A welcome message should appear like so:



```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to I2S Example 2...
```

```
Play the microphone input from the WM8731 codec to the output.  
-----
```

2. Any sounds heard at the microphone input (P1 on FT90x EVM) will be output from the Codec via the FT90x.

## 3.14 PWM Examples

### 3.14.1 PWM Example 1

#### 3.14.1.1 Purpose

The purpose of this example is to demonstrate using the PWM module to output a fixed duty cycle.

#### 3.14.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect the following signals to an oscilloscope for measurement:

- GPIO56/PWM0 (CN3 Pin 13 on FT90x EVM)
- GPIO57/PWM1 (CN3 Pin 14 on FT90x EVM)
- GPIO58/PWM2 (CN3 Pin 12 on FT90x EVM)

#### 3.14.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to PWM Example 1...

Output a number of PWM levels on various pins:
* PWM0 will output 25% duty cycle
* PWM1 will output 50% duty cycle
* PWM2 will output 75% duty cycle
-----
```

2. *GPIO56/PWM0* should have a 25% duty cycle wave output on it, *GPIO57/PWM1* should have a 50% duty cycle wave output on it, and *GPIO58/PWM2* should have a 75% duty cycle wave output on it.

### 3.14.2 PWM Example 2

#### 3.14.2.1 Purpose

The purpose of this example is to demonstrate using the PWM module to output a variable duty cycle PWM wave. This example will exponentially fade PWM0 up and down in order to demonstrate an LED fading.

#### 3.14.2.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Optionally, connect an LED or oscilloscope to *GPIO56/PWM0* (CN3 Pin 13 on FT90x EVM).

#### 3.14.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  

Welcome to PWM Example 2...
```

```
Output a PWM signal to drive a fading LED on PWM0  

The so called, breathing LED.
```

- The output on *GPIO56/PWM0* should vary between 100% and 0% duty cycle in an exponential fashion.

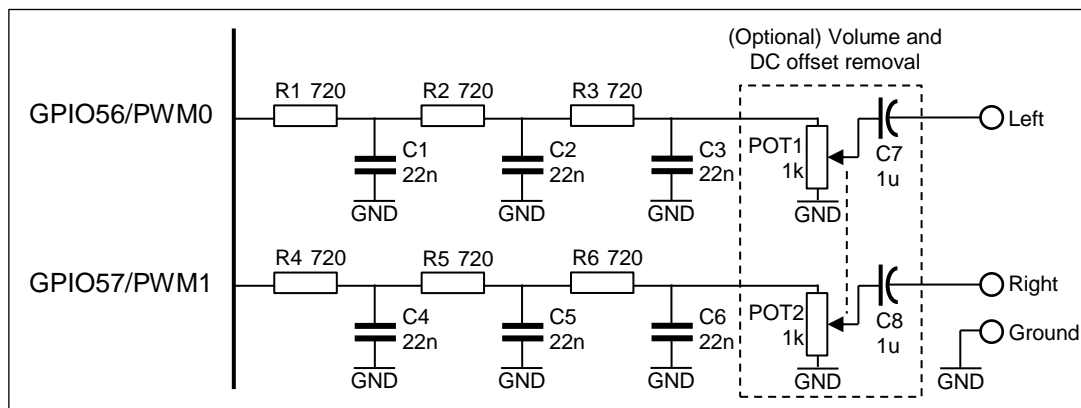
### 3.14.3 PWM Example 3

#### 3.14.3.1 Purpose

The purpose of this example is to demonstrate using the PWM module to output audio.

#### 3.14.3.2 Setup

A low pass filter will be needed to remove the PWM carrier frequency from the output waveform. Figure 19 shows the circuit needed to create a 10.047 kHz Low Pass Filter with optional stereo volume control and DC offset removal.



**Figure 19: PWM Low Pass Filter**

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Connect the following signals to an oscilloscope for measurement:

- GPIO56/PWM0 (CN3 Pin 13 on FT90x EVM)
- GPIO57/PWM1 (CN3 Pin 14 on FT90x EVM)

#### 3.14.3.3 Execution

- A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  

Welcome to PWM Example 3...
```

```
Output a sine wave on the PWM audio channel (PWM0 and PWM1)
```

- GPIO56/PWM0* and *GPIO57/PWM1* should be outputting PWM waves which represent the Left and Right channels of a Sine wave.

## 3.15 Real Time Clock Examples

### 3.15.1 RTC Example 1

#### 3.15.1.1 Purpose

The purpose of this example is to demonstrate the Real Time Clock Peripheral.

#### 3.15.1.2 Setup

On the MM900EVxA board two 0-ohm resistors [**R141** and **R142**] must be connected and R143 and R144 should be removed before the FT90x internal RTC can be used. These resistors connect the external crystal to the RTC. By default the crystal is connected to MCP7940M External RTC module.

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.15.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to RTC Example 1...

Display the current elapsed time using the RTC.
-----
```

2. The program should display the current elapsed time:

```
Uptime 0 h 0 m 1 s
```

### 3.15.2 RTC Example 2

#### 3.15.2.1 Purpose

The purpose of this example is to demonstrate the Real Time Clock Peripheral. This example demonstrates the time matching capability.

#### 3.15.2.2 Setup

On the MM900EVx board two 0-ohm resistors [**R141** and **R142**] must be connected and R143 and R144 should be removed before the FT90x internal RTC can be used. These resistors connect the external crystal to the RTC. By default the crystal is connected to MCP7940M External RTC module.

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.15.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to RTC Example 2...
```

```
Display a message every two seconds via an RTC interrupt.  
-----
```

2. Every two seconds, the program should update the currently elapsed time:

```
2 seconds elapsed
```

## 3.16 SD Host Examples

### 3.16.1 SD Host Example 1

#### 3.16.1.1 Purpose

The purpose of this example is to demonstrate the SD Host Peripheral.

#### 3.16.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Insert a FAT32 formatted SD card (CN5 on the FT90x EVM).

#### 3.16.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to SD Host Example 1..

Read and write some files using FatFS
-----
```

2. The program will prompt for an SD Card to be inserted:

```
Please Insert SD Card
```

3. After inserting an SD Card, the program will mount the file system:

```
SD Card Inserted
Mounted
```

4. The program will list the contents of the drive's root directory:

```
ls(path = ""):
DD/MM/YYYY HH:MM          Size Filename
01/08/2014 10:57          333878 SCR01.BMP
01/08/2014 10:57          333878 SCR02.BMP
20/08/2014 10:32              0 SCR03.BMP
20/08/2014 10:33              0 SCR04.BMP
22/07/2014 13:51          207360 TMCAPP~1.EXE
. . .
17/07/2014 16:56   <DIR>          0 FT900
30/10/2014 16:09          114146 ETH_EX~1.PNG
28/10/2014 10:48        151328281 V100~1.ZIP
04/11/2014 13:16              210 GCCVARS.BAT
42 File(s)          290935952 bytes
```

5. The program will write some data to a text file:

```
LOREM.TXT already exists. Deleting
Opening LOREM.TXT for writing
Wrote 1658 bytes
Closing LOREM.TXT
```

## 6. The program will read back the file:

```
Opening LOREM.TXT for reading
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi dictum mi eget  
malesuada auctor. Cras tellus ligula, feugiat ac ante eu, tincidunt consectetur  
mauris. Phasellus in mollis enim, dapibus venenatis est. Sed urna tellus, varius a  
dui sed, scelerisque commodo lectus. In pretium lobortis tortor, semper ultricies  
odio viverra a. Ut sit amet aliquam lectus. Phasellus non risus a nisl semper  
vehicula a vitae lorem. Fusce suscipit, purus nec facilisis lacinia, lacus massa  
aliquet augue, in feugiat neque nibh a lacus. Curabitur pharetra viverra massa  
quis efficitur.
```

```
Mauris posuere nisl vel aliquam finibus. Aenean ac fringilla justo. Nulla eu  
sollicitudin erat. Duis in ligula at quam pretium hendrerit. Fusce quis egestas  
metus. In hac habitasse platea dictumst. Fusce tincidunt enim at tempus  
ullamcorper. Aenean pellentesque condimentum sapien vel porta. In sollicitudin  
tempor pulvinar. Pellentesque aliquet justo lacus, scelerisque feugiat augue  
commodo viverra.
```

```
Etiam pulvinar quam a pulvinar aliquam. Cras rutrum quis tortor ut ultrices.  
Curabitur sit amet odio eros. Mauris auctor erat non risus interdum, at venenatis  
urna interdum. Nam eget auctor risus, auctor fringilla leo. Quisque sit amet  
ligula mattis, gravida tortor quis, ullamcorper odio. Nullam semper mauris at leo  
aliquam, quis mollis tortor iaculis. Mauris ut tempor elit, sed sodales magna.  
Donec non eros tortor. Donec lorem justo, vestibulum vitae sagittis ac, bibendum  
vitae velit. Integer ante mi, tempus sodales consectetur vel, porta ac libero.  
Maecenas dapibus orci at rhoncus bibendum. Nulla elementum lectus massa, non  
varius lorem scelerisque sit amet.
```

```
Closing LOREM.TXT
```

## 3.17 SPI Master Examples

### 3.17.1 SPI Master Example 1

#### 3.17.1.1 Purpose

The purpose of this example is to demonstrate the use of the SPI Master Peripheral and how to transfer data.

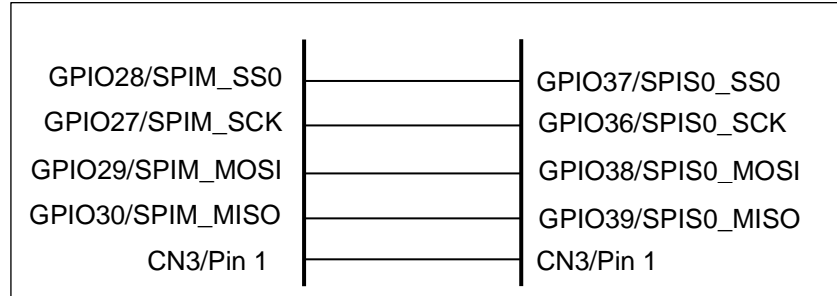
#### 3.17.1.2 Setup

This example uses two FT90x EVM modules, one module as SPI master and other module as SPI slave.

Connect as shown in Figure 20.

Connect the *GPIO28/SPIM\_SS0* (accessible via J2 Pin 2 on FT90x EVM) to *GPIO37/SPIS0\_SS* (accessible via CN7 Pin 1 on FT90x EVM) .

1. Connect the *GPIO27/SPIM\_SCK* (accessible via J2 Pin 1 on FT90x EVM) to *GPIO36/SPIS0\_SCK* (accessible via CN7 Pin 2 on FT90x EVM).
2. Connect the *GPIO29/SPIM\_MOSI* (accessible via J2 Pin 4 on FT90x EVM) to *GPIO38/SPIS0\_MOSI* (accessible via CN7 Pin 3 on FT90x EVM).
3. Connect the *GPIO30/SPIM\_MISO* (accessible via J2 Pin 3 on FT90x EVM) to *GPIO39/SPIS0\_MISO* (accessible via CN7 Pin 4 on FT90x EVM).
4. Ensure that both the boards have a common ground (CN3 Pin 1 of the two boards can be tied together)



**Figure 20: Circuit Diagram for SPI Master Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to dumping the exchange data.

Note that the Slave device should be started (powered up) first. If not, the master may send data when the slave is not running.

#### 3.17.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to SPI Master Example 1...

Loopback use case between two MM900EVxA module, FT900 (SPI Master) and FT900 (SPI
Slave)
-----
```

2. The program will start exchanging the data between SPI master and SPI Slave.
3. The program will start by dumping the contents of exchange:



```

Data sent and received 70 6e 71 6f 72 70 73 71 74 72 75 73 76 74 77 75
Data sent and received 78 76 79 77 7a 78 7b 79 7c 7a 7d 7b 7e 7c 7f 7d
Data sent and received 80 7e 81 7f 82 80 83 81 84 82 85 83 86 84 87 85
Data sent and received 88 86 89 87 8a 88 8b 89 8c 8a 8d 8b 8e 8c 8f 8d
Data sent and received 90 8e 91 8f 92 90 93 91 94 92 95 93 96 94 97 95
Data sent and received 98 96 99 97 9a 98 9b 99 9c 9a 9d 9b 9e 9c 9f 9d
Data sent and received a0 9e a1 9f a2 a0 a3 a1 a4 a2 a5 a3 a6 a4 a7 a5
Data sent and received a8 a6 a9 a7 aa a8 ab a9 ac aa ad ab ae ac af ad
Data sent and received b0 ae b1 af b2 b0 b3 b1 b4 b2 b5 b3 b6 b4 b7 b5
Data sent and received b8 b6 b9 b7 ba b8 bb b9 bc ba bd bb be bc bf bd
Data sent and received c0 be c1 bf c2 c0 c3 c1 c4 c2 c5 c3 c6 c4 c7 c5
Data sent and received c8 c6 c9 c7 ca ce cb 4f cc fa cd cb ce cc cf cd
Data sent and received d0 d9 d1 ff d2 d0 d3 d1 d4 d2 d5 d3 d6 d4 d7 d5
Data sent and received d8 d6 d9 d7 da d8 db d9 dc da dd db de dc df dd
Data sent and received e0 de e1 df e2 e0 e3 e1 e4 e2 e5 e3 e6 e4 e7 e5
Data sent and received e8 e6 e9 e7 ea e8 eb e9 ec ea ed eb ee ec ef ed
Data sent and received f0 ee f1 ef f2 f0 f3 f1 f4 f2 f5 f3 f6 f4 f7 f5
  
```

### 3.17.2 SPI Master Example 2

#### 3.17.2.1 Purpose

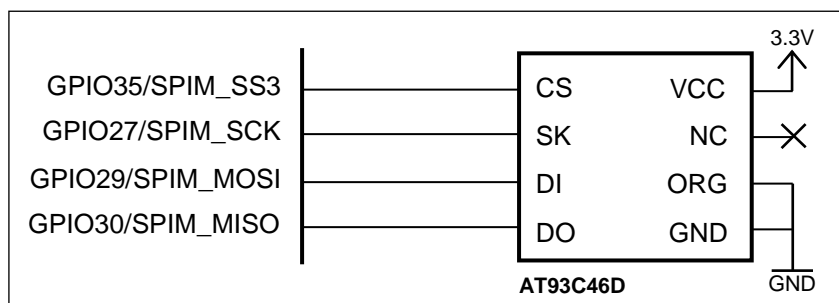
The purpose of this example is to demonstrate the use of the SPI Master Peripheral and how to transfer data by using FIFOs to buffer the transfers.

#### 3.17.2.2 Setup

This example uses an AT93C46D 1Kbit EEPROM as a SPI device.

Connect as shown in Figure 21

1. Connect the *GPIO35/SPIM\_SS3* (accessible via CN3 Pin 17 on FT90x EVM) to the CS pin of the AT93C46D.
2. Connect the *GPIO27/SPIM\_SCK* (accessible via CN3 Pin 18 on FT90x EVM) to the SK pin of the AT93C46D.
3. Connect the *GPIO29/SPIM\_MOSI* (accessible via CN3 Pin 20 on FT90x EVM) to the DI pin of the AT93C46D.
4. Connect the *GPIO30/SPIM\_MISO* (accessible via CN3 Pin 19 on FT90x EVM) to the DO pin of the AT93C46D.
5. Connect the VCC pin of the AT93C46D to 3.3V.
6. Connect the ORG and GND pin of the AT93C46D to GND.



**Figure 21: Circuit Diagram for SPI Master EEPROM Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to send debug text.

### 3.17.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to SPI Master Example 2...

Read and Write from a serial EEPROM (AT93C46D) using FIFOs to
Streamline transfers
-----
```

2. The program will start by dumping the contents of EEPROM:

```
Reading all 128 bytes of EEPROM
0x0000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
```

3. The program will then set all locations in EEPROM to FF<sub>h</sub> then dump the contents of EEPROM:

```
Setting the EEPROM to 0xFF
0x0000: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0010: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0020: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0030: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0040: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0050: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0060: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
0x0070: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF | .....
```

4. The program will then set all even locations to 01<sub>h</sub> then dump the contents of EEPROM:

```
Set all even numbered locations to 0x01
0x0000: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0010: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0020: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0030: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0040: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0050: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0060: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
0x0070: 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF 01 FF | .....
```

5. The program will then fill EEPROM with a block of example text and dump the contents of EEPROM:

```
Filling the EEPROM with example text
0x0000: 4C 6F 72 65 6D 20 69 70 73 75 6D 20 64 6F 6C 6F | Lorem ipsum dolo
0x0010: 72 20 73 69 74 20 61 6D 65 74 2C 20 63 6F 6E 73 | r sit amet, cons
0x0020: 65 63 74 65 74 75 72 20 61 64 69 70 69 73 63 69 | ectetur adipisci
0x0030: 6E 67 20 65 6C 69 74 2E 20 41 6C 69 71 75 61 6D | ng elit. Aliquam
0x0040: 20 69 6E 74 65 72 64 75 6D 20 65 72 6F 73 20 73 | interdum eros s
0x0050: 69 74 20 61 6D 65 74 20 6C 6F 72 65 6D 20 70 75 | it amet lorem pu
0x0060: 6C 76 69 6E 61 72 2C 20 76 65 6C 20 70 6F 73 75 | lvinar, vel posu
0x0070: 65 72 65 20 6C 65 6F 20 70 6F 73 75 65 72 65 2E | ere leo posuere.
```

### 3.17.3 SPI Master Example 3

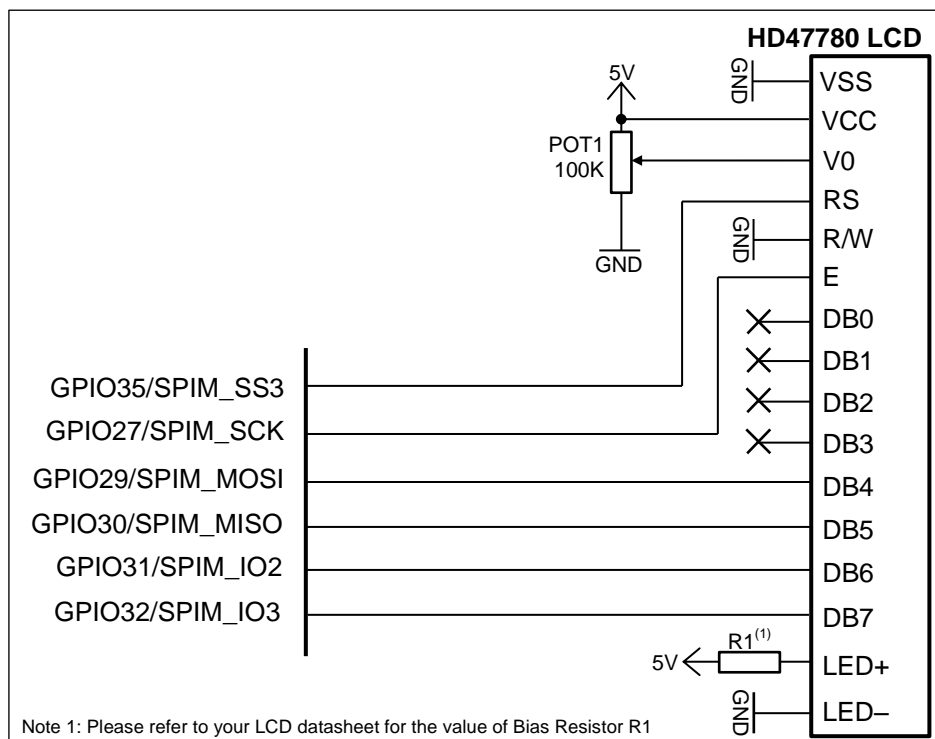
#### 3.17.3.1 Purpose

The purpose of this example is to demonstrate the use of the SPI Master Peripheral and how to transfer data in 4-bit mode.

#### 3.17.3.2 Setup

This example uses a HD47780 compatible LCD in 4 bit mode shown in Figure 22.

1. Connect *GPIO35/SPIM\_SS3* (accessible via CN3 Pin 17 on FT90x EVM) to the *RS* pin of the LCD.
2. Connect *GPIO27/SPIM\_SCK* (accessible via J2 Pin 1 on FT90x EVM) to the *E* pin of the LCD.
3. Connect *GPIO28/SPIM\_MOSI* (accessible via J2 Pin 4 on FT90x EVM) to the *DB4* pin of the LCD.
4. Connect *GPIO30/SPIM\_MISO* (accessible via J2 Pin 3 on FT90x EVM) to the *DB5* pin of the LCD.
5. Connect *GPIO31/SPIM\_IO2* (accessible via J2 Pin 6 on FT90x EVM) to the *DB6* pin of the LCD.
6. Connect *GPIO32/SPIM\_IO3* (accessible via J2 Pin 5 on FT90x EVM) to the *DB7* pin of the LCD.
7. Connect the *VSS*, *R/W* and *LED-* pins of the LCD to *GND*.
8. Connect the *VCC* pin of the LCD to *5V*.
9. Connect a 100k $\Omega$  potentiometer between *5V* and *GND*, with the wiper going to pin *V0* of the LCD.
10. Connect a Resistor between *LED+* of the LCD and *5V* (This Resistor is used to bias the LED backlight in the LCD module, please refer to the LCD's documentation to determine this value).



**Figure 22: Circuit Diagram for SPI Master Example 3**

### 3.17.3.3 Execution

1. A welcome message should appear like so:

```

(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to SPI Master Example 3..

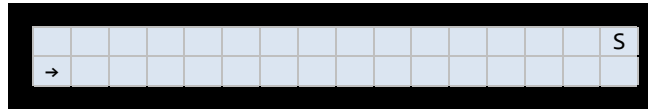
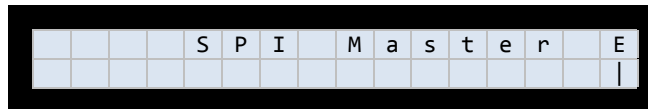
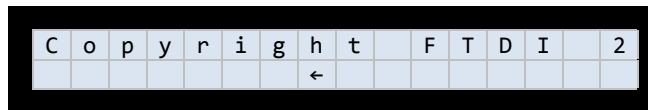
Use 4-bit mode to drive a HD44780 compatible LCD
-----
```

2. The program will initialize the LCD to 4-bit mode, displaying the message:

```

Initialising the LCD
```

3. The program will scroll through the text "SPI Master Example 3 Copyright FTDI 2014" on the first line and will show a bouncing animation on the second line. For example:

## 3.18 SPI Slave Examples

### 3.18.1 SPI Slave Example 1

#### 3.18.1.1 Purpose

The purpose of this example is to demonstrate the use of the SPI Slave Peripheral and how to transfer data.

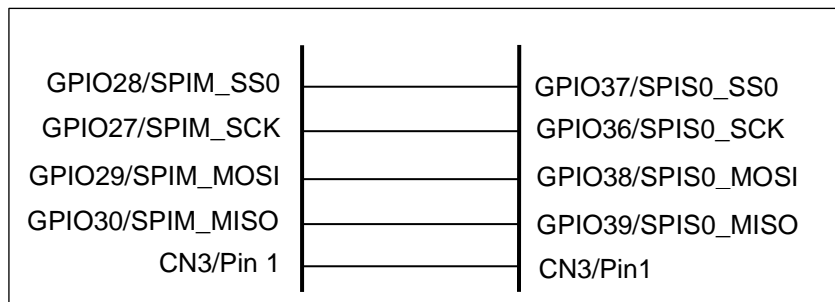
The default buffer size for this example is 8 bytes deep. This can be changed by editing the APP\_BUFFER\_SIZE definition and recompiling the example.

#### 3.18.1.2 Setup

This example uses two FT90x EVM modules, one module as SPI master and other module as SPI slave.

Connect the following as shown in Figure 23.

1. Connect the *GPIO28/SPIM\_SS0* (accessible via J2 Pin 2 on FT90x EVM) to *GPIO37/SPIS0\_SS* (accessible via CN7 Pin 1 on FT90x EVM) .
2. Connect the *GPIO27/SPIM\_SCK* (accessible via J2 Pin 1 on FT90x EVM) to *GPIO36/SPIS0\_SCK* (accessible via CN7 Pin 2 on FT90x EVM).
3. Connect the *GPIO29/SPIM\_MOSI* (accessible via J2 Pin 4 on FT90x EVM) to *GPIO38/SPIS0\_MOSI* (accessible via CN7 Pin 3 on FT90x EVM).
4. Connect the *GPIO30/SPIM\_MISO* (accessible via J2 Pin 3 on FT90x EVM) to *GPIO39/SPIS0\_MISO* (accessible via CN7 Pin 4 on FT90x EVM).
5. Ensure that both the boards have a common ground (CN3 Pin 1 of the two boards can be tied together)



**Figure 23: Circuit Diagram for SPI Slave Examples**

Additionally, connect a USB to Serial converter to UART0 as this port is used to dump debug text.

Note that the Slave device should be started (powered up) first. If not, the master may send data when the slave is not running.

### 3.18.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to SPI Master Example 1...

Loopback use case between two MM900EVxA module, FT900 (SPI Master) and FT900 (SPI
Slave)
-----
```

This is followed by instructions on screen for quick reference.

2. At the setup time predefined data is written into the transmission FIFO, at run time read and write back the data to SPI master.

## 3.19 Timer Examples

### 3.19.1 Timer Example 1

#### 3.19.1.1 Purpose

The purpose of this example is to demonstrate using the Timers in single shot mode.

#### 3.19.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.19.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to Timer Example 1...

All timers are in one-shot mode and are polled in the main loop.
* Timer A will expire after 5 seconds.
* Timer B will expire after 6 seconds.
* Timer C will expire after 7 seconds.
* Timer D will expire after 8 seconds.
The current state of the timer will be shown every second
-----
```

2. Every second, the status of the timers will be printed. The output should be:

```
Timer _ _ _ _
Timer _ _ _ _
Timer _ _ _ _
Timer _ _ _ _
Timer _ _ _ _
Timer A _ _ _ _
Timer _ B _ _
Timer _ _ C _
Timer _ _ _ D
Timer _ _ _ _
```

### 3.19.2 Timer Example 2

#### 3.19.2.1 Purpose

The purpose of this example is to demonstrate using the Timers in continuous mode.

#### 3.19.2.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.19.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to Timer Example 2...
```

```
All timers are in continuous mode and are polled in the main loop.
```

```
* Timer A will fire every 2 seconds.
```

```
* Timer B will fire every 3 seconds.
```

```
* Timer C will fire every 4 seconds.
```

```
* Timer D will fire every 5 seconds.
```

```
The current state of the timer will be shown every second
```

- Every second, the status of the timers will be printed. The output should be:

```
Timer _ _ _ _  
Timer _ _ _ _  
Timer A _ _ _  
Timer _ B _ _  
Timer A _ C _  
Timer _ _ _ D  
Timer A B _ _  
Timer _ _ _ _  
Timer A _ C _  
Timer _ B _ _  
Timer A _ _ D  
Timer _ _ _ _  
Timer A B C _  
Timer _ _ _ _  
Timer A _ _ _  
Timer _ B _ D
```

### 3.19.3 Timer Example 3

#### 3.19.3.1 Purpose

The purpose of this example is to demonstrate using the Timers in continuous mode and being serviced by an interrupt.

#### 3.19.3.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.19.3.3 Execution

- A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to Timer Example 3...
```

```
All timers are in continuous mode and are polled in the main loop.
```

```
* Timer A will fire every 2 seconds.
```

```
* Timer B will fire every 3 seconds.
```

```
* Timer C will fire every 4 seconds.
```

```
* Timer D will fire every 5 seconds.
```

```
The current state of the timer will be shown every second
```



2. Every second, the status of the timers will be printed. The output should be:

```
Timer _ _ _ _  
Timer _ _ _ _  
Timer A _ _ _  
Timer _ B _ _  
Timer A _ C _  
Timer _ _ _ D  
Timer A B _ _  
Timer _ _ _ _  
Timer A _ C _  
Timer _ B _ _  
Timer A _ _ D  
Timer _ _ _ _  
Timer A B C _  
Timer _ _ _ _  
Timer A _ _ _  
Timer _ B _ D
```

## 3.20 UART Examples

### 3.20.1 UART Example 1

#### 3.20.1.1 Purpose

The purpose of this example is to demonstrate using the UART to transmit and receive characters.

#### 3.20.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text and for the example.

#### 3.20.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to UART Example 1...

Any character typed here will be echoed back on the same serial port.
-----
```

2. Typing any characters here will cause them to be transmitted back.

Note that some terminal programs have a 'local echo' function. If enabled, then the transmitted and received characters are displayed. If disabled, only the receive character is displayed.

### 3.20.2 UART Example 2

#### 3.20.2.1 Purpose

The purpose of this example is to demonstrate using the UART to transmit and receive characters using an interrupt.

#### 3.20.2.2 Setup

Refer to 3.20.1.2 Setup Section.

#### 3.20.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to UART Example 2...

Any character typed here will be echoed back on the same serial port
via an interrupt
-----
```

2. Typing any characters here will cause them to be transmitted back.

Note that some terminal programs have a 'local echo' function. If enabled, then the transmitted and received characters are displayed. If disabled, only the receive character is displayed.

### **3.20.3 UART Example 3**

#### **3.20.3.1 Purpose**

The purpose of this example is to demonstrate using the UART to transmit and receive characters using an interrupt which stores data in local memory and then the local data transmitted back every second.

#### **3.20.3.2 Setup**

Refer to 3.20.1.2 Setup Section.

#### **3.20.3.3 Execution**

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to UART Example 3..

Any character typed here will be echoed back on the same serial port
via an interrupt and polled every second.
-----
```

2. Typing any characters here will cause them to be transmitted back in one second bursts.

## 3.21 USB Device Examples

The USB device examples emulate a particular class of device or implement a function on the FT900. The examples typically show emulating a device class.

Data can be received from or sent to the USB host permitting 'bridge' devices to be made which transfer data from one interface to the USB device interface.

The specifications of the devices emulated in this section may be obtained from the USB -IF website at [http://www.usb.org/developers/docs/devclass\\_docs/](http://www.usb.org/developers/docs/devclass_docs/).

FTDI have reserved a range of USB PIDs (from 0x0fd0 to 0x0fdf) which have been allocated to different device classes to facilitate testing. Currently allocated PID values are listed in Table 1.

Class	VID	PID
CDC ACM	0x0403 (FTDI)	0x0fd1
RNDIS Networking	0x0403 (FTDI)	0x0fd3
CDC NCM	0x0403 (FTDI)	0x0fd4
Mass Storage	0x0403 (FTDI)	0x0fd5
HID	0x0403 (FTDI)	0x0fda
DFU	0x0403 (FTDI)	0xfde

**Table 1 USB device example VIDs and PIDs.**

For example, all HID devices for the FT900 will use a VID of 0x0403 and a PID of 0x0fda.

### 3.21.1 GPIO DFU Example

Provides a Device Firmware Update (DFU) interface, allowing a USB host to update firmware on the device. This will allow a ROM image to be downloaded to a device from a utility program running on a USB host. The firmware will wait for a GPIO line to be pulled down or a character to be received from the UART to enable a DFU mode.

#### 3.21.1.1 Purpose

The purpose of this example is to demonstrate using the USB device to download new firmware to a device. A DFU interface is instantiated on the device allowing a utility program on the host to connect to the device and download or upload firmware to or from the device. The DFU mode can be activated programmatically when some input source is signaled. This example uses a GPIO line or the UART interface.

The example complies with the Microsoft WCID specifications to automatically install a WinUSB driver when the device is plugged into a Windows system. This simplifies installing the drivers required by a DFU utility on the host to communicate with the DFU interface.

### 3.21.1.2 Setup

Connect the FT900 USB device port to a USB host. Pull GPIO 18 low (to GND) or send a carriage return over the UART and DFU mode will be activated. Pull GPIO 17 low or send a space character over the UART and DFU mode will be activated for 5 seconds. Pull GPIO 16 low or send any other character over the UART and DFU mode will be activated for the default timeout which is around 1 second.

When the device is enumerated for the first time it will require a driver to be installed. Windows will install the WinUSB device driver automatically.

The utility program dfu-util (<http://dfu-util.sourceforge.net/>) can be used to send firmware to the device.

### 3.21.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to GPIO_DFU Example...

Start a DFU interface on the USB Device Port when a GPIO is pulled low.
-----
On GPIO trigger GPIO18 for infinite DFU timeout
                GPIO17 for 5 seconds DFU timeout
                GPIO16 for default timeout
On UART press <CR> for infinite DFU timeout
                <space> for 5 seconds DFU timeout
                Any other key for default timeout
```

2. Pull GPIO18 low to enable the DFU interface.

```
GPIO18 interrupted
Starting DFU - never to return
```

3. Send the new firmware image to the device using the dfu-util utility:

```
C:\>dfu-util.exe -D dfu_test_file.bin.bin
dfu-util 0.8

Copyright 2005-2009 Weston Schmidt, Harald Welte and OpenMoko Inc.
Copyright 2010-2014 Tormod Volden and Stefan Schmidt
This program is Free Software and has ABSOLUTELY NO WARRANTY
Please report bugs to dfu-util@lists.gnumonks.org

Invalid DFU suffix signature
A valid DFU suffix will be required in a future dfu-util release!!!
Opening DFU capable USB device...
ID 0403:0fde
Run-time device DFU version 0110
Claiming USB DFU Interface...
Setting Alternate Setting #0 ...
Determining device status: state = dfuIDLE, status = 0
dfuIDLE, continuing
DFU mode device DFU version 0110
Device returned transfer size 256
Copying data from PC to DFU device
Download      [=====] 100%      258048 bytes
Download done.
state(6) = dfuMANIFEST-SYNC, status(0) = No error condition is present
unable to read DFU status after completion
```

The “unable to read DFU status after completion” message is not indicative of a failure. It is expected as the device resets itself when the firmware download is complete and therefore dfu-util cannot receive an answer for a status request.

## 3.21.2 USB Example BOMS to SD Card

### 3.21.2.1 Purpose

This example program will create FT900 as a USB Bulk-Only Mass Storage (BOMS) device to the USB host. Data for the mass storage device are read from or written to an SD card inserted in the FT900 device.

### 3.21.2.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text. Insert a FAT32 formatted SD card (CN5 on the FT90x EVM). Connect the FT900 USB device port to a USB host. The device is enumerated as a USB Mass Storage Device.

### 3.21.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USB Mass Storage to SD Card Test Application ...
-----
Change...connect card 2
Card initialise 0
Restarting
```

2. The FT900 device, after enumeration at the Windows host, should appear as a Removable Disk Drive. Using Windows Explorer, Open the folder to view files in the drive. All the file operations – create, open, write, read, close, delete can be performed.

### 3.21.3 USB Example HID

Emulates a keyboard when connected to a USB host and sends a fixed sequence of key presses to the operating system on the host.

#### 3.21.3.1 Purpose

The purpose of this example is to demonstrate using the USB device to emulate a keyboard by instantiating a HID device. The HID device is a boot mode keyboard which will enumerate then send a predetermined string to the host as simulated key presses. The operating system on the host can receive and treat these key presses as if they were from a real keyboard.

The example presents 2 USB interfaces to the host, a keyboard interface and a DFU (Device Firmware Update) interface. This is therefore considered to be a composite USB device. The DFU interface allows firmware updates to the device under the control of a utility running on the host, as per the previous example.

The method for handling standard, class and vendor requests for a composite device are shown. Device, configuration and string descriptors for the device are defined and handled in the example code.

An interrupt IN endpoint is on the keyboard interface which is polled by the USB host. Simulated key presses are sent to the host using a report descriptor. The format of the descriptor is given in the code and is available to the host to aid it in interpreting the key press 'reports'. The required functions of a HID class device are covered including the SetIdle request.

The string to send to the host demonstrates pressing normal alpha-numeric keys, the Caps Lock key, and the carriage return key.

#### 3.21.3.2 Setup

Connect the FT900 USB device port to a USB host. When the device is enumerated as a USB keyboard it will start sending a text string to the host which will appear as if typed in by a real keyboard.

#### 3.21.3.3 Execution

1. A welcome message should appear like so,

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USB HID Tester Example 1.,,
Emulate a HID device connected to the USB Device Port
-----
```

2. After enumeration this string will appear on the debug console and the host system.

```
Hello, I am an FTDI FT900 device... nice to meet you!
```

3. The device will not perform any further actions.

### 3.21.4 USB Example HID Bridge

Emulates a keyboard when connected to a USB host and sends characters received from the UART interface as key presses to the operating system on the host.

#### 3.21.4.1 Purpose

The purpose of this example is to demonstrate using the USB device to emulate a keyboard by instantiating a HID device. The HID device is a boot mode keyboard which will enumerate then convert characters received from the UART interface into simulated key presses. The operating system on the host can receive and treat these key presses as if they were from a real keyboard.

The example presents 2 USB interfaces to the host, a keyboard interface and a DFU (Device Firmware Update) interface. This is therefore considered to be a composite USB device. The DFU interface allows firmware updates to the device under the control of a utility running on the host, as per the previous example.

The method for handling standard, class and vendor requests for a composite device are shown. Device, configuration and string descriptors for the device are defined and handled in the example code.

An interrupt IN endpoint is on the keyboard interface which is polled by the USB host. Simulated key presses are sent to the host using a report descriptor. The format of the descriptor is given in the code and is available to the host to aid it in interpreting the key press 'reports'. The required functions of a HID class device are covered including the SetIdle request.

The characters translated from the UART interface include several control characters and escape sequences which are produced by popular PC terminal emulation programs.

#### 3.21.4.2 Setup

Connect the FT900 USB device port to a USB host and a USB to UART convertor to the FT900 UART.

#### 3.21.4.3 Execution

1. A welcome message should appear like so,

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USB HID Bridge Example 1.,,
Emulate a HID device connected to the USB Device Port, bridge buffer
from the UART to the HID device.
-----
```

2. After enumeration characters received from the UART interface will appear as key presses on the host system.

```
Hello, I am an FTDI FT900 device... nice to meet you!
```



### 3.21.5 USB Example CDCACM

Emulates a Communications Device Class (CDC) Abstract Control Model (ACM) device when connected to a USB host. This example allows the operating system on the host to open a virtual COM port to the CDC ACM device. Data is read from the UART interface and transmitted to the host, data received from the host is sent to the UART interface.

#### 3.21.5.1 Purpose

The purpose of this example is to demonstrate using the USB device to emulate a Communications Device Class (CDC) device. The type of CDC device emulated is an Abstract Control Model (CDC ACM).

The example shows a composite device presenting 3 USB interfaces to the host, a CDC CONTROL interface, a CDC DATA interface and a DFU interface.

The method for handling standard, class and vendor requests for the composite device are shown. Device, configuration and string descriptors for the device are defined and handled in the example code.

The CDC CONTROL interface has an interrupt IN endpoint to receive notifications which is polled by the USB host. A notification structure is sent to the host periodically when the state of the UART changes.

The CDC DATA interface has 2 BULK endpoints, one for IN packets and one for OUT packets. This forms a bi-directional data pipe to the virtual COM port on the USB host for both receiving data and transmitting data. A circular buffer on the device is used to turn the data stream to and from the UART into packets used by the USB interface.

A DFU interface is also provided device. The example also complies with the Microsoft WCID specifications to automatically install a WinUSB driver when the device is plugged into a Windows system. This simplifies installing the drivers required by a DFU utility on the host to communication with the DFU interface.

#### 3.21.5.2 Setup

Connect the FT900 USB device port to a USB host. When the device is enumerated it will require a driver to be installed. This is only required the first time it is connected.

To install the Windows built-in CDC ACM driver start Device Manager and right-click on the "FT900 CDC ACM" device in "Other Devices". The click on "Update device driver...", then "Browse my computer for driver software" and navigate to the examples directory. The next step may be a Dialog box stating that "Windows can't verify the publisher of this driver software", click on "Install this driver software anyway". The CDC ACM driver will install and a virtual COM port will appear in Device Manager under "COM Ports".

When running a terminal program on the USB host (such as PuTTY or CoolTerm) a virtual COM port can be opened to allow direct communication with the emulated CDC device. If the UART interface on the FT900 is also connected to a PC terminal program as well then data can be sent both ways between the terminal programs. It is also possible to connect to an external device such as a modem.

### 3.21.5.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USB CDC ACM Tester Example 1..

Emulate a CDC ACM device connected to the USB Device Port.
-----
```

2. The device will continue to bridge from the USB CDC ACM interface to the UART.

### 3.21.6 USB Example RNDIS

Emulates a Remote Network Driver Interface Specification (RNDIS) compliant device when connected to a USB host. The FT900 device is the network device that provides network connectivity to the host PC over USB.

A USB RNDIS device is implemented as a USB Communication Device Class (CDC) device with two interfaces. A Communication Class interface, of type Abstract Control, and a Data Class interface combined to form a single functional unit representing the USB Remote NDIS device. The Communication Class interface includes a single endpoint for event notification and uses the shared bidirectional Control endpoint for control messages. The Data Class interface includes two bulk endpoints for data traffic.

#### 3.21.6.1 Purpose

The purpose of this example is to demonstrate using the USB device to emulate a network adapter. The type of CDC device emulated is an Abstract Control Model (CDC ACM).

The example shows a composite device presenting 3 USB interfaces to the host, a CDC CONTROL interface, a CDC DATA interface and a DFU runtime interface.

The method for handling standard, class and vendor requests for the composite device are shown. Device, configuration and string descriptors for the device are defined and handled in the example code.

The CDC CONTROL interface has an interrupt IN endpoint to receive notifications which is polled by the USB host. A notification structure is sent to the host periodically when the state of the UART changes.

The CDC DATA interface has 2 BULK endpoints, one for IN packets and one for OUT packets. This forms a bi-directional data pipe to the USB host for both receiving and transmitting network data packets. A circular buffer on the device is used to turn the data stream to and from the ethernet into packets used by the USB RNDIS device interface.

A DFU interface is also provided in the device. The example also complies with the Microsoft WCID specifications to automatically install a WinUSB driver when the device is plugged into a Windows system. This simplifies installing the drivers required by a DFU utility on the host to communication with the DFU interface.

#### 3.21.6.2 Setup

Connect an Ethernet cable to the FT90x's Ethernet Port. Connect a USB to Serial converter to UART0 as this port is used to send debug text. Connect the FT900 USB device port to a USB host. When the device is enumerated, the windows host will install the default device driver (Rndismp<y>.sys Usb8023<y>.sys). The FT900 RNDIS device appears as one of the Network Adapters in the Windows Device manager.

#### 3.21.6.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
```

```
-----  
Welcome to USBD RNDIS Tester Example 1...
```

```
Emulate a RNDIS device connected to the USB Device Port.
```

```
-----  
MAC a0:00:00:c0:ff:ee
```

```
Restarting
```

```
Restarting
```

```
Enumerated
```

```
Please plug in your Ethernet cable
```

```
.Rx 0 Err 0 Dr 0 Tx 0 Err 0 Dr 0
```

```
Packet filter ON P N M N
```

```
Packet filter ON P N M Y
```

```
Packet filter ON P N M Y
```

```
.....Rx 0 Err 0 Dr 0 Tx 0 Err 0 Dr 0
```

```
.....Rx 0 Err 0 Dr 0 Tx 0 Err 0 Dr 0
```

2. Connect the other end of the Ethernet cable at the FT90x's Ethernet Port into a LAN network. The host gets network connectivity over FT900 RNDIS device.

```
..Media connected 100 Mb/sec
```

```
.....Rx 15 Err 0 Dr 0 Tx 68 Err 0 Dr 0
```

```
.....Rx 102 Err 0 Dr 0 Tx 128 Err 0 Dr 0
```

```
.....Rx 259 Err 1 Dr 11 Tx 238 Err 0 Dr 0
```

```
.....Rx 341 Err 1 Dr 12 Tx 276 Err 0 Dr 0
```

```
.....Rx 456 Err 1 Dr 12 Tx 426 Err 0 Dr 0
```

```
.....Rx 740 Err 1 Dr 22 Tx 706 Err 0 Dr 0
```

3. The device continues to bridge Ethernet to USB. This could be checked by issuing ping requests for IP addresses in the network. Other devices and PCs in the LAN can be discovered through the windows explorer and shared folders can be accessed.

## 3.22 USB Host Examples

The USB host examples demonstrate connecting to and controlling devices connected to the host USB.

Data can be received from or sent to USB devices and control operations can be performed on these devices.

The specifications of the devices connected to in this section may be obtained from the USB -IF website at [http://www.usb.org/developers/docs/devclass\\_docs/](http://www.usb.org/developers/docs/devclass_docs/).

### 3.22.1 USBH\_Example Hub

Lists devices connected to the USB host port of the FT900. The output is sent to the UART interface.

#### 3.22.1.1 Purpose

The purpose of this example is to demonstrate using the USB host to find and identify devices on the USB. It will send queries to the devices found to request additional information.

#### 3.22.1.2 Setup

Connect the FT900 USB host port to a USB device or a USB hub with multiple devices. When the devices are enumerated the program will list all detected devices.

#### 3.22.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH Hub Tester Example 1...

List devices connected to the USB Host Port
-----
```

2. When a device is connected then the program will output information about the device. This example is for a generic USB flash disk:

```
USB Device Detected
USB Device Enumerated

Device found at level 1

Device Descriptors:
bcdUSB:           0200
bDeviceClass:    00
bDeviceSubClass: 00
bDeviceProtocol: 00
bMaxPacketSize0: 40
VID:             1043
PID:             8012
bcdDevice:       0100

iManufacturer:   Generic
iProduct:        Flash Disk
iSerialNumber:   0604260938510038

Configuration Descriptors:
wTotalLength:    0020
bNumInterfaces:  01
bConfigurationValue: 01
iConfiguration:  00
bmAttributes:    80
MaxPower:        32

Interface Descriptors:
bInterfaceNumber: 00
bAlternateSetting: 00
bNumEndpoints:    02
bInterfaceClass:  08
bInterfaceSubClass: 06
bInterfaceProtocol: 50
iInterface:        00

Endpoint Descriptors:
bEndpointAddress: 81
Transfer Type:     Bulk
wMaxPacketSize:   0000
bInterval:         02

Endpoint Descriptors:
bEndpointAddress: 02
Transfer Type:     Bulk
wMaxPacketSize:   0000
bInterval:         02

Please remove the USB Device
```

Devices can be removed once they have been queried and other devices then inserted. Devices connected to USB hubs will be queried as well, however, the example code does not check for connection or removal events on USB hubs and will not update the output if new devices are added or devices removed from a downstream hub.

### 3.22.2 USBH Example HID

Display report data received from a Human Interface Device (HID) over the USB. The output is sent to the UART interface. The data is not interpreted to decode the meaning of the HID reports.

### 3.22.2.1 Purpose

The purpose of this example is to demonstrate receiving USB HID reports from a HID device. This is done by polling an interrupt IN endpoint for data from the device under test.

A simple blocking read is made of the interrupt endpoint with a 1000 ms timeout. If data has been received then the data is displayed in hexadecimal format.

### 3.22.2.2 Setup

Connect the FT900 USB host port to a USB HID device such as a keyboard or a mouse. When the HID device is enumerated the program will start displaying HID reports as they are received from the HID device.

### 3.22.2.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH HID Tester Example 1...

Find and displays reports received from HID devices connected to
the USB Host Port
-----
```

2. When a device is connected then the program will output information from the device:

```
USB Device Detected
USB Device Enumerated
HID device found at level 1
VID: 03f0 PID: 0024
Speed: 0 low
Address: 1
Setting idle
Reports from device 8 bytes:
20 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
20 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00
Timeout
```

The HID reports can be decoded into key presses and displayed on the terminal. However, this is out with the scope of this example code. This example shows a low-speed keyboard with 8 bytes of HID report data. The "Timeout" occurs when the device does not respond to the host requests within an arbitrary length of time; it does not indicate an error.

### 3.22.3 USBH Example CDCACM

Implement a UART to Communication Device Class (CDC) bridge over the USB.

#### 3.22.3.1 Purpose

This example code demonstrates bridging bi-directional data from the UART interface to a CDC device on the USB. The CDC device must support Abstract Control Model (CDC ACM).

### 3.22.3.2 Setup

Connect the FT900 USB host port to a USB CDC ACM device such as a modem or mobile phone. When the CDC device is enumerated the program will start sending data received from the UART interface to the CDC device and returning data from the CDC device to the UART interface.

### 3.22.3.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH CDC Tester Example 1...

Bridge data from the UART to a CDC ACM device on the USB host port.
-----
```

2. When a device is connected then the program will output information about the device then bridge data:

```
USB Device Detected
USB Devices Enumerated
Beginning CDC ACM testing 0 0
Beginning CDC ACM testing
Sending encapsulated reset... not supported -3
Sending reset to DATA interface...Received OK
ATZ
OK
ATDT07778889999
ERROR
```

The example code will send an encapsulated reset command ("ATZ"). This may or may not be supported by the device.

In the example output the reset command "ATZ" was sent via the UART as the encapsulated command was not accepted by the device.

Then an attempt to dial was made which resulted in an error – the device did not have a SIM card.

## 3.22.4 USBH Example BOMS

Implement a simple tester for USB Bulk-Only Mass Storage (BOMS) devices.

### 3.22.4.1 Purpose

This example code will query a flash disk which supports the BOMS specification and read sectors 0 and 1025 with the contents displayed in hexadecimal format on the UART interface. It will then read the entire first cluster of the disk and display that similarly on the UART interface.

### 3.22.4.2 Setup

Connect the FT900 USB host port to a USB BOMS device such as a Flash disk. When the BOMS device is enumerated the program will start sending data received from the sectors read on the BOMS device to the UART interface.

### 3.22.4.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH BOMS Tester Example 1..

Find and exercises BOMS (Flash Disk) devices connected to the USB
Host Port
-----
```

2. When a device is connected, the program will output information about the device followed by data from the device:

```
USB Device Detected
USB Devices Enumerated
BOMS device found at level 1

Sector 0
33 c0 8e c0 8e d8 8e d0 bc 00 7c fc 8b f4 bf 00
06 b9 00 01 f2 a5 ea 44 06 00 00 8b d5 58 b4 10
f6 e4 05 ae 04 8b f0 8a 74 01 8b 4c 02 bb 00 7c
b8 01 02 cd 13 72 16 81 bf fe 01 55 aa 75 0e ea
00 7c 00 00 80 fa 81 74 02 b2 80 8b ea bf be 07
b9 04 00 32 f6 8a 45 04 3c 00 74 0b 3c 05 74 07
80 3d 80 74 19 fe c6 83 c7 10 e2 e9 0a f6 74 06
be 9c 06 eb 04 90 be b4 06 e8 0e 00 eb fe 8a c6
04 31 50 be 99 06 bb 1b 06 53 fc ac 50 24 7f b4
0e cd 10 58 a8 80 74 f2 c3 0d 0a a0 0d 0a 4e 6f
20 61 63 74 69 76 65 20 70 61 72 74 69 74 69 6f
6e 2e 2e ae 0d 0a 50 61 72 74 69 74 69 6f 6e 20
6e 6f 74 20 66 6f 75 6e 64 2e 2e ae 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 81 24 18 11
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 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 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 0a 0b 0c 0d 00 00 80 01
01 00 06 0f e0 ff 20 00 00 00 e0 ff 07 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 55 aa

Sector 1025
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

### 3.22.5 USBH Example File System

Implement a simple file system tester for USB Bulk-Only Mass Storage (BOMS) devices. This uses the FatFS library.



### 3.22.5.1 Purpose

This example code will query a flash disk which supports the BOMS specification and read sectors 0 and 1025 with the contents displayed in hexadecimal format on the UART interface. It will then read the entire first cluster of the disk and display that similarly on the UART interface.

### 3.22.5.2 Setup

Connect the FT900 USB host port to a USB BOMS device such as a Flash disk. When the BOMS device is enumerated the program will start performing tests on the file system on the BOMS device and sending the result to the UART interface.

### 3.22.5.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH File System Tester Example 1...

Find and exercises Flash Disks devices connected to the USB
Host Port
-----
```

2. When a device is connected the program will display the files in the root directory of the disk:

```
USB Device Detected
USB Device Enumerated
BOMS device found at level 1
ls(path = ""):
DD/MM/YYYY HH:MM          Size Filename
01/08/2014 10:57          333878 SCR01.BMP
01/08/2014 10:57          333878 SCR02.BMP
20/08/2014 10:32              0 SCR03.BMP
20/08/2014 10:33              0 SCR04.BMP
22/07/2014 13:51          207360 TMCAPP~1.EXE
...
17/07/2014 16:56   <DIR>          0 FT900
30/10/2014 16:09          114146 ETH_EX~1.PNG
28/10/2014 10:48          151328281 V100~1.ZIP
04/11/2014 13:16              210 GCCVARS.BAT
42 File(s)          290935952 bytes
```

3. Then the program will write some data to a text file:

```
LOREM.TXT already exists. Deleting
Opening LOREM.TXT for writing
Wrote 1658 bytes
Closing LOREM.TXT
```

- The program will read back the file:

```
Opening LOREM.TXT for reading
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi dictum mi eget  
malesuada auctor. Cras tellus ligula, feugiat ac ante eu, tincidunt consectetur  
mauris. Phasellus in mollis enim, dapibus venenatis est. Sed urna tellus, varius a  
dui sed, scelerisque commodo lectus. In pretium lobortis tortor, semper ultricies  
odio viverra a. Ut sit amet aliquam lectus. Phasellus non risus a nisl semper  
vehicula a vitae lorem. Fusce suscipit, purus nec facilisis lacinia, lacus massa  
aliquet augue, in feugiat neque nibh a lacus. Curabitur pharetra viverra massa  
quis efficitur.
```

```
Mauris posuere nisl vel aliquam finibus. Aenean ac fringilla justo. Nulla eu  
sollicitudin erat. Duis in ligula at quam pretium hendrerit. Fusce quis egestas  
metus. In hac habitasse platea dictumst. Fusce tincidunt enim at tempus  
ullamcorper. Aenean pellentesque condimentum sapien vel porta. In sollicitudin  
tempor pulvinar. Pellentesque aliquet justo lacus, scelerisque feugiat augue  
commodo viverra.
```

```
Etiam pulvinar quam a pulvinar aliquam. Cras rutrum quis tortor ut ultrices.  
Curabitur sit amet odio eros. Mauris auctor erat non risus interdum, at venenatis  
urna interdum. Nam eget auctor risus, auctor fringilla leo. Quisque sit amet  
ligula mattis, gravida tortor quis, ullamcorper odio. Nullam semper mauris at leo  
aliquam, quis mollis tortor iaculis. Mauris ut tempor elit, sed sodales magna.  
Donec non eros tortor. Donec lorem justo, vestibulum vitae sagittis ac, bibendum  
vitae velit. Integer ante mi, tempus sodales consectetur vel, porta ac libero.  
Maecenas dapibus orci at rhoncus bibendum. Nulla elementum lectus massa, non  
varius lorem scelerisque sit amet.
```

```
Closing LOREM.TXT
```

### 3.22.6 USBH Example FT232

This example is used to list an FTDI device connected to the USB host root hub port of the FT900. It then bridges the FTDI device such as FT232R or FT232H detected on the USB to the UART0 interface.

#### 3.22.6.1 Purpose

The purpose of this example is to demonstrate using the USB host to find and identify FTDI devices connected to the USB root hub port. It will send queries to the devices found to request additional information.

#### 3.22.6.2 Setup

- Run the FT900 device with the USBH Example FT232.bin. Connect a USB to Serial converter to UART0 as this port is used to send debug text. Open up terminal PC application program for UART0 with following port setting 19200 baud, no parity, 8 data bits, and 1 stop bit, no flow control:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH FT232 Tester Example 1...

Send and receive data from an FT232 connected to the USB host port.
-----

Please plug in a USB Device
```

2. Crossover two FT232R devices (RTS to CTS and vice versa; Rx to Tx and vice versa, GND to GND). Connect one USB end of the crossover cable to the PC. Open the terminal application for the VCP port enumerated with following port setting 19200 baud, no parity, 8 data bits, and 1 stop bit, RTS/CTS flow control.

### 3.22.6.3 Execution

1. Connect the other end of the crossover cable to the FT900's USB host root hub port. When the device is enumerated the program will list the detected device.

```
USB Device Detected
USB Device Enumerated
0403\6001 device found
FT232 device found
E2Addr:0x12; E2Data:0054
E2Addr:0x13; E2Data:0054
E2Addr:0x14; E2Data:004C
Beginning FT testing... latency: 16, modemstat: 6011
```

2. Typing any characters from the FT232R port in the PC will cause them to be transmitted to the UART0 of FT900 (observed on the terminal application of UART0) and similarly typing from the terminal application of UART0 appears in terminal application of FT232R.

### 3.22.7 AOA Examples

The Android AOA feature allows Android devices to attach to USB Host devices as an accessory. In this mode, the Android device is powered by the Host device (such as FT900). Data exchange is over custom Bulk End Points or over USB HID (available in AOAv2). AOAv1 support is available from Android 3.1 and AOAv2 support is available from Android 4.1.

The FT900 USB Peripheral Driver Library provides an AOA driver (usbh\_aoa.c) that implements the AOA protocol.

#### 3.22.7.1 Installing Example Apps onto an Android Device

- The GPIO and UARTLoopback examples require companion Apps on the Android device to interact with. The HID example does not require any specific App to run.
- To install the Apps download and extract the "Android.zip" folder from <http://www.ftdichip.com/Support/SoftwareExamples/Android/Android.zip>
- An App note explaining the details of the Android Apps is available online: [AN\\_208 FT311/FT312D Demo APK User Guide](#)
- Copy or send the GPIODemoActivity.apk and UARTLoopbackActivity.apk onto the Android device.
- On the Android device, go to Settings > Security > Unknown Sources and select "Allow installation of apps which are not from the electronic market".

- Using a file browser App (such as "File Explorer" or "ES File Explorer") on the Android device navigate to the apk location and open them. Install both the GPIODemoActivity and UARTLoopbackActivity.

**Note:** Ensure Developer Options > USB Debugging is not enabled on the Android device; otherwise the examples will not function correctly.

### 3.22.7.2 GPIO Example

#### 3.22.7.2.1 Purpose

The purpose of this demo is to illustrate data exchange between an Android App and the FT900.

#### 3.22.7.2.2 Setup

Program the FT900 development board with "USBH Example AOA GPIO 1.bin".

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Open up terminal PC application program for UART0 with following port setting: 19200 baud, no parity, 8 data bits, 1 stop bit.

Connect FT900 USB Host Port to an Android device that contains the GPIODemoActivity.apk pre-installed.

#### 3.22.7.2.3 Execution

1. A welcome message should appear on the PC terminal application like so,

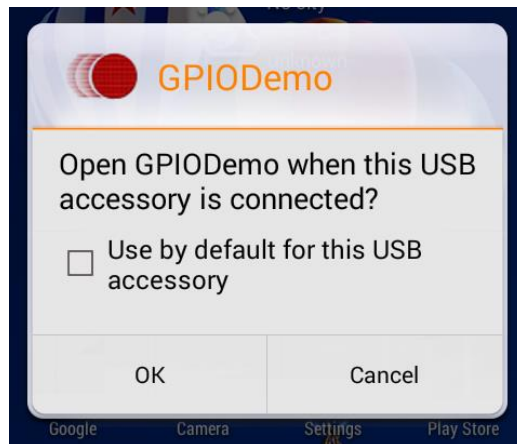
```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH GPIO AOA Tester Example 1..
Tests the FTDIGPIODemo on an Android device.
-----
```

2. Connect the Android device to the MM900EV Board using a USB cable. The PC terminal application should display

```

USB Device Detected
.USB Devices Enumerated
Init AOA
Will re-enumerate as AOA
.....
.....
.....
.....USB Devices
Enumerated
Init AOA
Attaching AOA
Testing AOA
VID: 18d1 PID: 2d01
Speed: 2 high
Address: 1
Accessory: yes
Audio: no
Adb: yes
Write: 4
  
```

On the Android device, the following pop-up should appear. Select OK to continue.



3. Select OK on the Android device. The GPIODemo application should be automatically launched and you should be able to see the following screen.



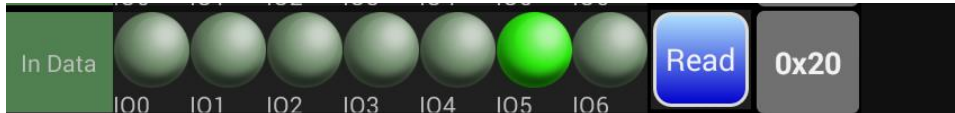
- Click the "Config" button on the App to send some data to FT900. The following message will appear on the PC terminal.

```
Configure command: Outmap 0000000 Inmap 1111111
```

- Click the "Read" button on the App to read the "Bitmap" status. The current Bitmap status is displayed on the PC terminal as:

```
Read Port: Bitmap 0000010
```

The appropriate bit in the "In Data" list will be set according to the current bitmap.



### 3.22.7.3 UART Loopback Example

#### 3.22.7.3.1 Purpose

The purpose of this demo is to illustrate data exchange between an Android App and the FT900.

#### 3.22.7.3.2 Setup

Program the FT900 development board with "USBH Example AOA UARTLoopback 1.bin".

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Open up terminal PC application program for UART0 with following port setting: 19200 baud, no parity, 8 data bits, 1 stop bit.

Connect FT900 USB Host Port to an Android device that contains the UARTLoopbackActivity.apk pre-installed.

#### 3.22.7.3.3 Execution

- A welcome message should appear on the PC terminal application like so,

```
Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH UART AOA Tester Example 1..
Tests the FTDIUARTLoopback on an Android device.
-----
```

- Connect the Android device to the MM900EV Board using a USB cable. The PC terminal application should display

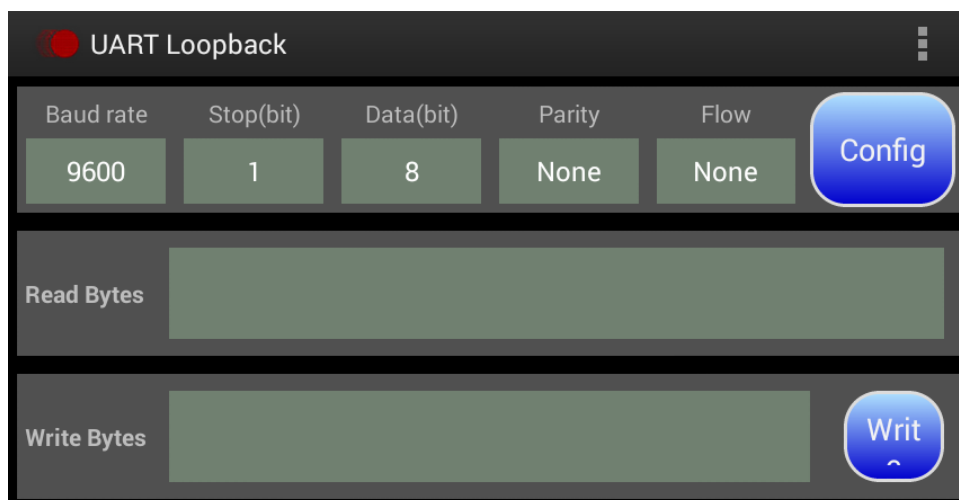
```

USB Device Detected
.USB Devices Enumerated
Init AOA
Will re-enumerate as AOA
.....
.....
.....
.....USB Devices Enumerated
Init AOA
Attaching AOA
Testing AOA
VID: 18d1 PID: 2d01
Speed: 2 high
Address: 1
Accessory: yes
Audio: no
Adb: yes
  
```

On the Android device, the following pop-up should appear



3. Select OK on the Android device. The UARTLoopback application should be automatically launched and you should be able to see the following screen

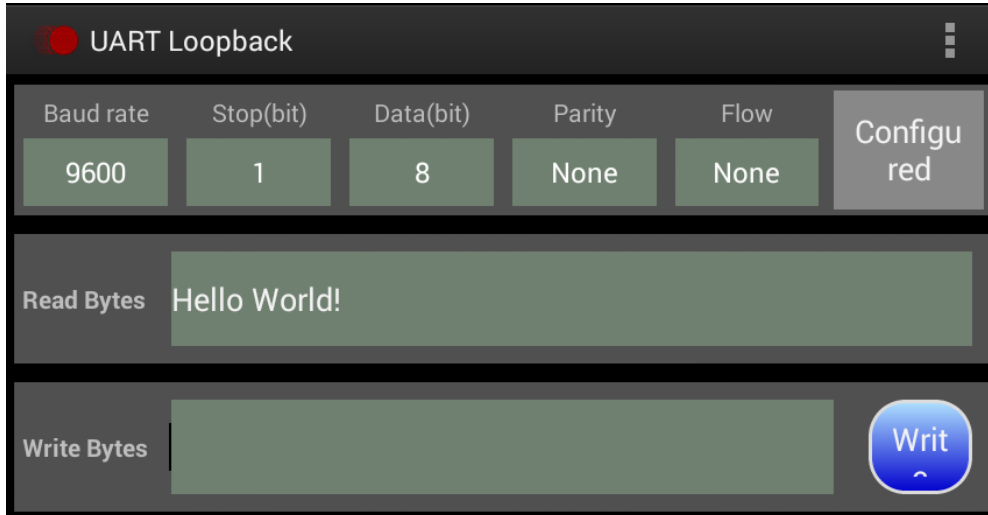


4. Press the "Config" button on the App. The following message will appear on the PC terminal. Note that none of the UART parameters are actually changed.

```

Config: baud 9600 stop 8 data 1 parity None flow None
  
```

- Send some data by typing into the PC terminal, for example the string "Hello World!" The string should appear in the "Read Bytes" text box in the Android App as shown below.



- To transmit data to FT900 type something in the "Write Bytes" text box in the App and click on "Write"

### 3.22.7.4 AOA HID Example

#### 3.22.7.4.1 Purpose

AOAv2 allows AOA Hosts to register as one or more Human Interface Device (HID) with the Android device. The HID example uses this feature to make FT900 register as a HID keyboard with the Android device. This demo can be used with any android App that can accept keyboard input.

#### 3.22.7.4.2 Setup

Program the FT900 development board with "USBH Example AOA HID 1.bin".

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

Open up terminal PC application program for UART0 with following port setting: 19200 baud, no parity, 8 data bits, 1 stop bit.

Connect FT900 USB Host Port to an Android device with android v4.1 and above.

#### 3.22.7.4.3 Execution

- A welcome message should appear on the PC terminal application like so,



```

Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to USBH AOA HID Tester Example 1..

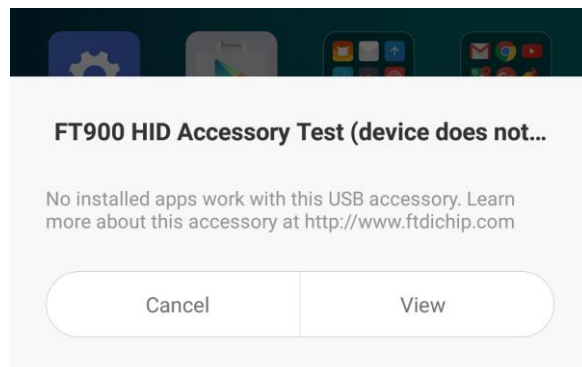
Bridges UART input to a HID keyboard on an Android device.
-----
  
```

2. Connect the Android device to the MM900EV Board using a USB cable. The PC terminal application should display

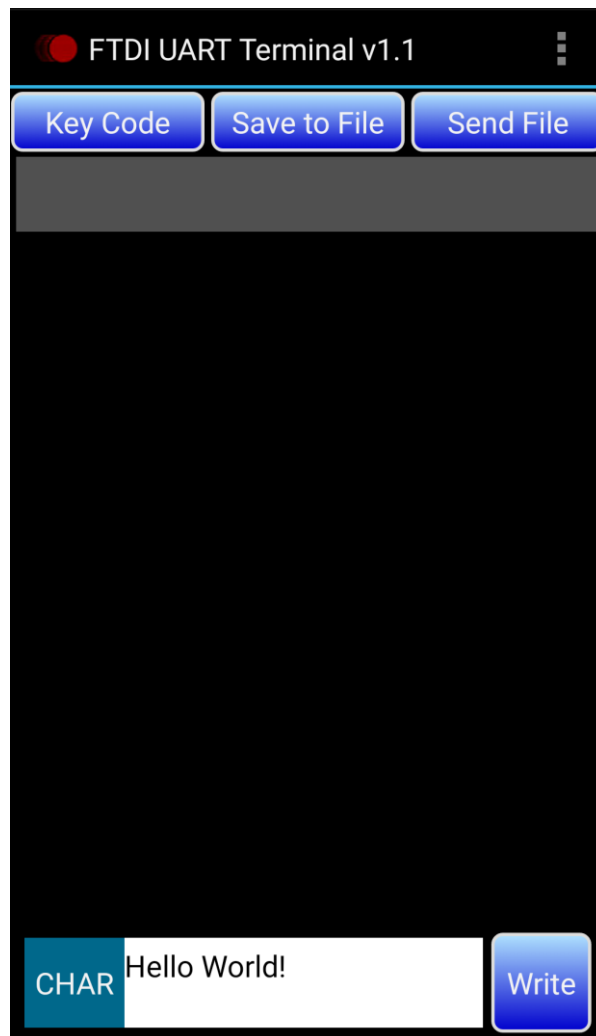
```

USB Device Detected
.USB Devices Enumerated
Init AOA
Will re-enumerate as AOA
.....U
SB Devices Enumerated
Init AOA
Attaching AOA
Testing AOA
VID: 18d1 PID: 2d01
Speed: 2 high
Address: 1
Accessory: yes
Audio: no
Adb: yes
  
```

3. On the Android device the following pop-up will appear. Select Cancel to proceed.



4. On the Android device, launch any App that can accept keyboard inputs like the [FTDI UART Terminal App](#) from the Google Play store or any text editor app.
5. At the PC terminal type a string that you wish to send to the Android device, e.g.: "Hello World!"
6. The string should be displayed on the Android device.



Note that even though the AOA specification says that Hosts that are not associated with Apps can enumerate without sending the manufacturer and model names (see section “Connecting to AOAv2 without an Android app” at <https://source.android.com/accessories/aoa2.html>). As of Android 5.x, such devices are not allowed to enumerate. This is currently still an open [bug](#).

## 3.23 Watchdog Timer Examples

### 3.23.1 Watchdog Example 1

#### 3.23.1.1 Purpose

The purpose of this example is to demonstrate using the Watchdog Timer.

#### 3.23.1.2 Setup

Connect a USB to Serial converter to UART0 as this port is used to send debug text.

#### 3.23.1.3 Execution

1. A welcome message should appear like so:

```
(C) Copyright 2016, Future Technology Devices International Ltd.
-----
Welcome to WDT Example 1...

Kick the watchdog for 10 seconds then let it expire.
-----
```

2. The program will kick the watchdog for 10 seconds and then resets the device:

```
WDT Kick 1
WDT Kick 2
WDT Kick 3
WDT Kick 4
WDT Kick 5
WDT Kick 6
WDT Kick 7
WDT Kick 8
WDT Kick 9
WDT Kick 10
```

The welcome message is shown after the device is reset and the timer example should run again if the program is stored in flash.

Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold FTDI harmless from any and all damages, claims, suits or expense resulting from such use.

**Future Technology Devices International Limited (FTDI)**

Unit 1, 2 Seaward Place, Glasgow G41 1HH, United Kingdom

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

Web Site: <http://ftdichip.com>

Copyright © Future Technology Devices International Limited

## 4 Contact Information

### Head Office – Glasgow, UK

Future Technology Devices International Limited  
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) [sales1@ftdichip.com](mailto:sales1@ftdichip.com)  
E-mail (Support) [support1@ftdichip.com](mailto:support1@ftdichip.com)  
E-mail (General Enquiries) [admin1@ftdichip.com](mailto:admin1@ftdichip.com)

### Branch Office – Tigard, Oregon, USA

Future Technology Devices International Limited  
(USA)  
7130 SW Fir Loop  
Tigard, OR 97223-8160  
USA  
Tel: +1 (503) 547 0988  
Fax: +1 (503) 547 0987

E-Mail (Sales) [us.sales@ftdichip.com](mailto:us.sales@ftdichip.com)  
E-Mail (Support) [us.support@ftdichip.com](mailto:us.support@ftdichip.com)  
E-Mail (General Enquiries) [us.admin@ftdichip.com](mailto:us.admin@ftdichip.com)

### Branch Office – Taipei, Taiwan

Future Technology Devices International Limited  
(Taiwan)  
2F, No. 516, Sec. 1, NeiHu Road  
Taipei 114  
Taiwan, R.O.C.  
Tel: +886 (0) 2 8797 1330  
Fax: +886 (0) 2 8751 9737

E-mail (Sales) [tw.sales1@ftdichip.com](mailto:tw.sales1@ftdichip.com)  
E-mail (Support) [tw.support1@ftdichip.com](mailto:tw.support1@ftdichip.com)  
E-mail (General Enquiries) [tw.admin1@ftdichip.com](mailto:tw.admin1@ftdichip.com)

### Branch Office – Shanghai, China

Future Technology Devices International Limited  
(China)  
Room 1103, No. 666 West Huaihai Road,  
Shanghai, 200052  
China  
Tel: +86 21 62351596  
Fax: +86 21 62351595

E-mail (Sales) [cn.sales@ftdichip.com](mailto:cn.sales@ftdichip.com)  
E-mail (Support) [cn.support@ftdichip.com](mailto:cn.support@ftdichip.com)  
E-mail (General Enquiries) [cn.admin@ftdichip.com](mailto:cn.admin@ftdichip.com)

### Web Site

<http://ftdichip.com>

### Distributor and Sales Representatives

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

System and equipment manufacturers and designers are responsible to ensure that their systems, and any Future Technology Devices International Ltd (FTDI) devices incorporated in their systems, meet all applicable safety, regulatory and system-level performance requirements. All application-related information in this document (including application descriptions, suggested FTDI devices and other materials) is provided for reference only. While FTDI has taken care to assure it is accurate, this information is subject to customer confirmation, and FTDI disclaims all liability for system designs and for any applications assistance provided by FTDI. Use of FTDI devices in life support and/or safety applications is entirely at the user's risk, and the user agrees to defend, indemnify and hold harmless FTDI from any and all damages, claims, suits or expense resulting from such use. This document is subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Neither the whole nor any part of the information contained in, or the product described in this document, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow G41 1HH, United Kingdom. Scotland Registered Company Number: SC136640

## Appendix A – References

### Document References

<http://www.ftdichip.com/Products/ICs/FT90x.html>

[FT900/FT901/FT902/FT903 Datasheet](#)

[FT905/FT906/FT907/FT908 Datasheet](#)

[FT900 User Manual](#)

[AN\\_325 FT900 Toolchain Installation Guide](#)

Serial Cables: <http://www.ftdichip.com/Products/Cables/USBTTLSerial.htm>

Bus Pirate: [http://dangerousprototypes.com/docs/Bus\\_Pirate](http://dangerousprototypes.com/docs/Bus_Pirate)

GNU Make Manual - 9.5 Overriding Variables

([http://www.gnu.org/software/make/manual/html\\_node/Overriding.html#Overriding](http://www.gnu.org/software/make/manual/html_node/Overriding.html#Overriding))

### Acronyms and Abbreviations

Terms	Description
ADC	Analogue to Digital Converter
ARP	Address Resolution Protocol
CAN	Controller Area Network
DAC	Digital to Analogue Converter
EEPROM	Electrically Erasable PROGRAMMABLE Memory
GPIO	General Purpose I/O
I <sup>2</sup> C	Inter-IC
I <sup>2</sup> S	Inter-IC Sound
ICMP	Internet Control Messaging Protocol
MDI-X	Medium Dependent Interface Crossover
PWM	Pulse Width Modulation
RTC	Real Time Clock
SD	Secure Digital
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
WDT	WatchDog Timer

## Appendix B – List of Tables & Figures

### List of Tables

Table 1 USB device example VIDs and PIDs. ....	67
--	----

### List of Figures

Figure 1: FT90x Interface driver support.....	5
Figure 2: FT90x Programming Utility Start Screen.....	7
Figure 3: FT90x Programming Utility One-Wire Option.....	7
Figure 4: Circuit Diagram for ADC Examples.....	8
Figure 5: Circuit diagram for CAN Examples.....	13
Figure 6: D2XX Port opened in the PC Terminal application.....	17
Figure 7: FT900 Programming Utility D2XX Tab.....	17
Figure 8: Output from dac_example1.c.....	18
Figure 9: Output from dac_example2.c.....	19
Figure 10: Wireshark output for eth_example1.c.....	26
Figure 11: Windows 7 LAN Properties.....	35
Figure 12: Host PC Static IP Configuration.....	36
Figure 13: Windows 7 - Find Host PC IP address.....	36
Figure 14: Update Host PC IP address.....	37
Figure 15: Simple TCP Client running on Host PC (FT900 dynamic IP is 10.44.0.120).....	38
Figure 16: Simple TCP server running on a Host PC.....	39
Figure 17: Circuit Diagram for I2C Master Examples.....	42
Figure 18: Circuit Diagram for I2C Slave Examples.....	45
Figure 19: PWM Low Pass Filter.....	50
Figure 20: Circuit Diagram for SPI Master Examples.....	55
Figure 21: Circuit Diagram for SPI Master EEPROM Examples.....	56
Figure 22: Circuit Diagram for SPI Master Example 3.....	58
Figure 23: Circuit Diagram for SPI Slave Examples.....	60

## Appendix C – Revision History

Document Title: AN\_360 FT90x Example Applications  
 Document Reference No.: FT\_001149  
 Clearance No.: FTDI# 451  
 Product Page: <http://www.ftdichip.com/FTProducts.htm>  
 Document Feedback: [Send Feedback](#)

Revision	Changes	Date
1.0	Initial Release	2015-06-29
1.1	Addition of USB Examples, I2C Master	2015-10-08
1.2	Added examples for USBH AOA, FreeRTOS, VFW Loader, D2XX, Datalogger feature, USB D BOMS to SD Card and BCD Device	2016-02-24
1.3	Update for toolchain v2.2.0 – NEW: MCCI section USBH-RNDIS, USBH-FT232; Update SPIM Example 2 to use SS3 on CN2 to match code, also CN2 is easier to access on the EVM; Update SPIM Example 2 to use CN3 for all SPI lines; Changes made based on new Programming Utility plus other minor changes; Updated Taiwan contact details; Add FreeRTOS + lwIP example; Update lwIP example for DHCP; SPIM example 3 uses GPIO35/SS3 to match code; Removed MCCI examples as they are to be shipped separately	2016-09-19