Application Note

AN_360

FT9XX Example Applications

Version 1.8

Issue Date: 2018-11-14

This application note describes the example applications provided for the FT9XX and demonstrates the use of the FT9XX Peripheral Driver Library.

Bridgetek Pte Ltd (BRT Chip)
178 Paya Lebar Road, #07-03, Singapore 409030
Tel: +65 6547 4827 Fax: +65 6841 6071
Web Site: http://www.brtchip.com

Table of Contents

1 Introduction

The FT9XX peripheral driver libraries (libft900.a and libft930.a) are a collection of 'C' language based functions that abstract away from the bare metal and present the programmer with an easy to use API to develop applications quickly and easily.

Figure 1 - FT90X Series Interface Driver Support and Figure 2 show the overall FT9XX Interface driver support. This document focuses on the Hardware Interface Driver layer to show examples of usage. All drivers except for the D2XXdrivers and MCCI stack (FT90X only) are provided as source code for easy adaptation and modification. Please contact Bridgetek if access to proprietary source code is required.

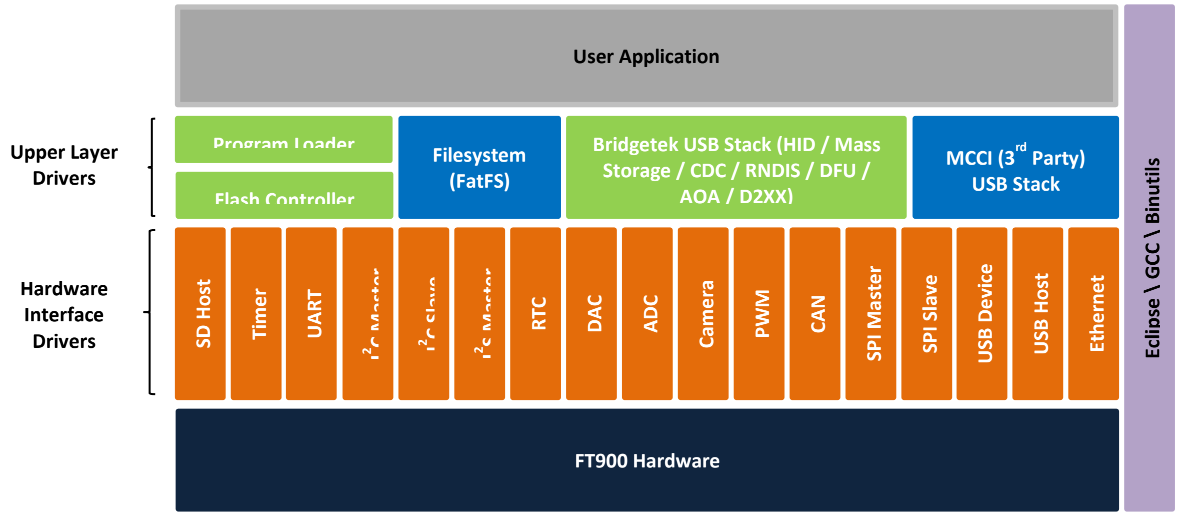


Figure 1 - FT90X Series Interface Driver Support

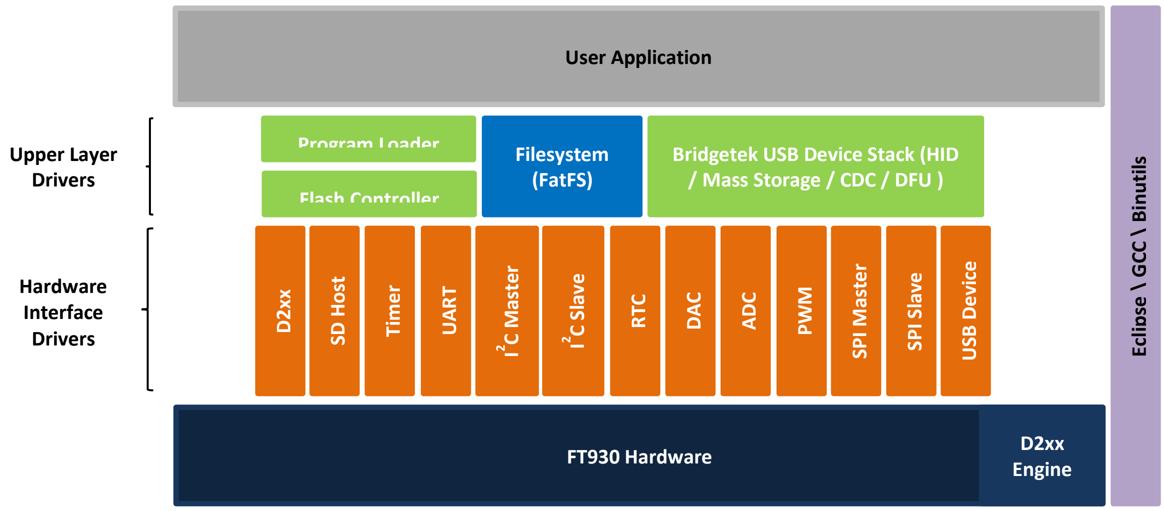


Figure 2 - FT93X Series Interface driver support

1.1 Overview

This document describes the construction and execution of the FT9XX example programs and is intended to introduce users to FT9XX programming using Bridgetek's FT9XX Driver Libraries in order to lower development time and deliver higher quality applications.

Table 1 shows the breakdown of examples between FT90X and FT93X series.

	Examples	**FT90X**	**FT93X**
1	Analogue to Digital Converter (ADC)	Yes	Yes
2	Camera Sensor Interface	Yes	No
3	Controller Area Network (CAN)	Yes	No
4	Digital to Analogue Converter (DAC)	Yes	Yes
5	Ethernet	Yes	No
6	General Purpose I/O (GPIO)	Yes	Yes
7	I2C Master	Yes	Yes
8	I2C Slave	Yes	Yes
9	I2S	Yes	No
10	Pulse Width Modulation (PWM)	Yes	Yes
11	Real Time Clock	Yes	Yes
12	SD Host	Yes	Yes
13	SPI Master	Yes	Yes
14	SPI Slave	Yes	Yes
15	Timer	Yes	Yes
16	Universal Asynchronous Receiver Transmitter (UART)	Yes	Yes
17	Watchdog	Yes	Yes

Table 1 - Examples supported by FT90X and FT93X

Additional examples can be found here: http://brtchip.com/ft90x/.

2 Installation and Programming

2.1 FT9XX Toolchain Installation

Please refer to AN_325 FT9XX Toolchain Installation Guide for instruction on how to install the FT9XX GCC toolchain.

2.2 Importing the Examples

Within Eclipse, select File → Import → General → Existing Projects into Workspace. Browse to the FT9XX examples directory (C:\Users*Username*\Documents\Bridgetek\FT9xx*version*\Examples). Select and click OK. The projects should appear in the Project Explorer.

2.3 Targets and Configurations

The example projects are preconfigured to target both the FT90X and FT93X family of microcontrollers. Each target MCU has two configurations – Debug and Release. The Debug configuration is intended to be used for debugging (with GDB) and uses the –Og optimization flag. The Release configuration uses –Os. The Debug and Release project configurations link to Debug and Release versions of the peripheral libraries, respectively.

Since there are two possible target MCUs, a project may have up to 4 possible configurations – FT900_Debug, FT900_Release, FT930_Debug and FT930_Release as shown in Figure 3. The user should set the appropriate configuration as the "active" configuration depending on the actual target microcontroller. The default active configuration is FT900_Debug. In Eclipse the active configuration can be changed by right clicking the project and selecting `Build Configurations | Set Active` or selecting the appropriate configuration from the dropdown list near the build icon () on the Eclipse main menu as shown in Figure 3.

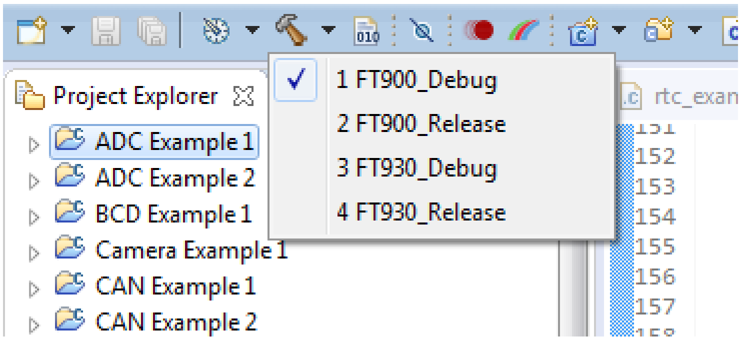


Figure 3 - Project Configurations

When creating a new project, the user can select the configurations required as shown in Figure 4 and Figure 5.

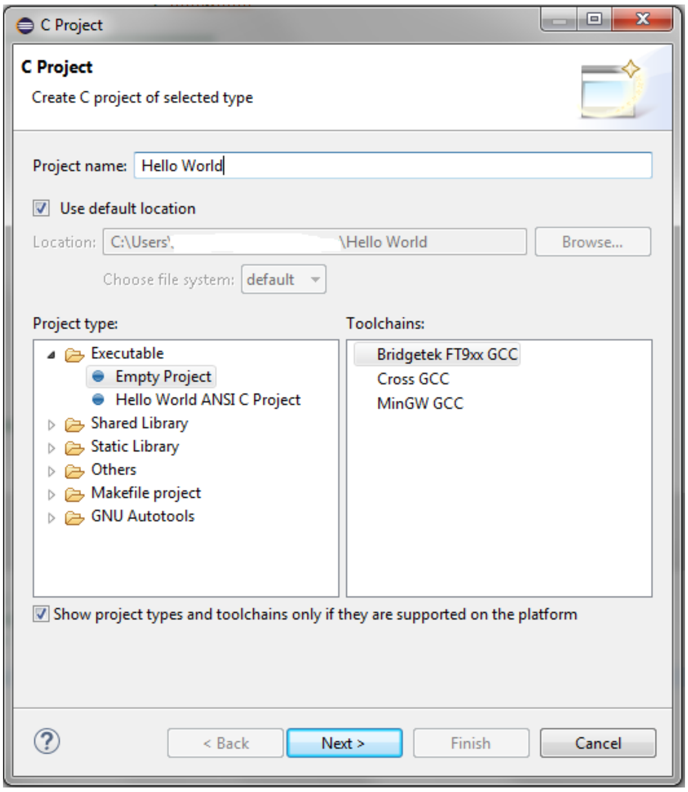


Figure 4 - Creating a new empty project

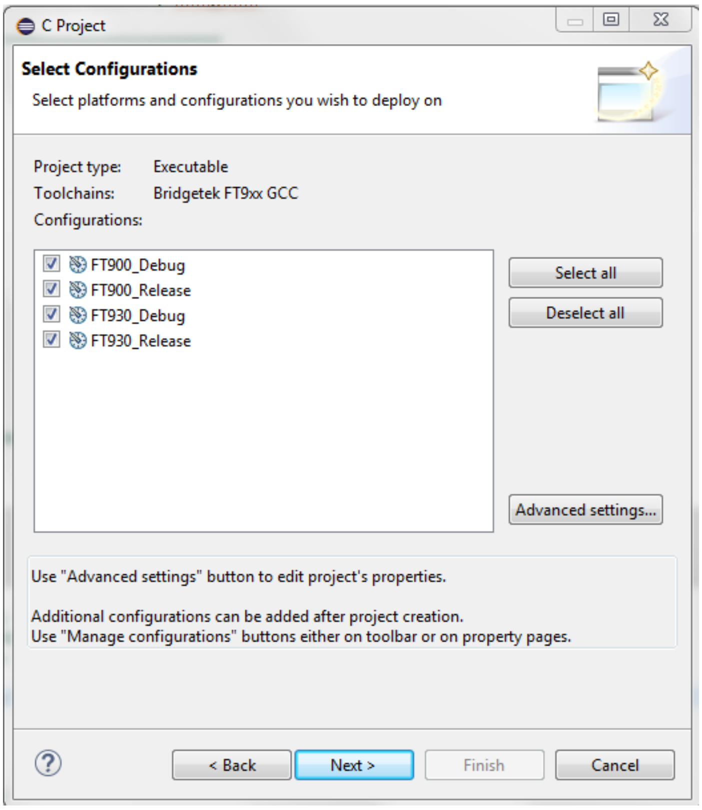


Figure 5 - Selecting the required target configurations

2.4 Supported Hardware

The examples have been verified on the following evaluation boards:

1. MM900EVxA and MM900EVxB for FT900
2. MM900EV-LITE for FT900
3. MM930Mini for FT930
4. MM930Lite for FT930

2.5 Building the FT9XX 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'.

2.6 Programming

For every example that is successfully built, a .bin and an .elf file are produced. The .bin file is the stripped version of the .elf file and it represents the binary image to be programmed into the device. For example, when UART Example 1 project is built, uart_example1.bin and uart_example1.elf files are produced. The uart_example1.bin file is programmed into the device and uart_example1.elf is used by the debugger.

To program a specific application into the memory of an FT9XX device, use the **Run → Run Configurations…**menu within Eclipse in Figure 6.

Alternatively you can use Bridgetek's free FT9XX Programming Utility ( ) shown in Figure 7. This can be accessed via the 'FT9XX Programming Utility' shortcut found on the desktop or under the 'Bridgetek Utilities' menu in Eclipse.

Please see more details on programming the binary file through Eclipse plugin or FT9XX programmer utility in AN_325 FT9XX Toolchain Installation Guide.

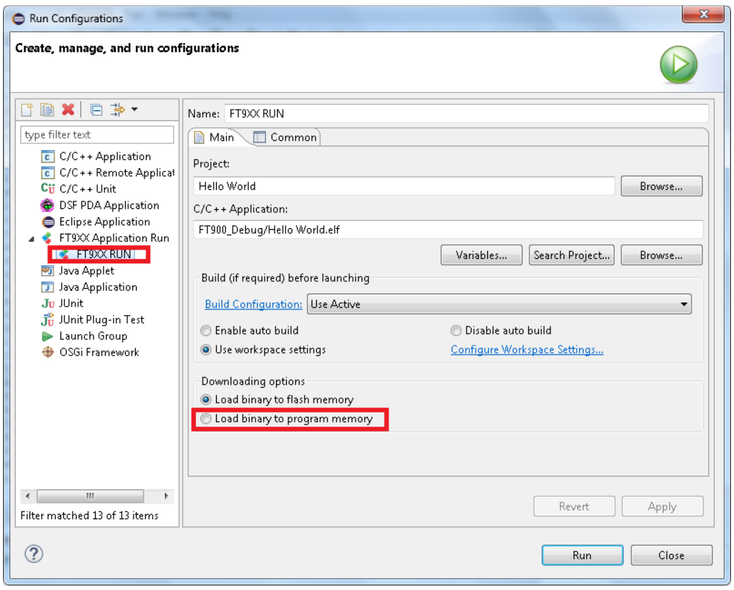


Figure 6 - Run Configuration window

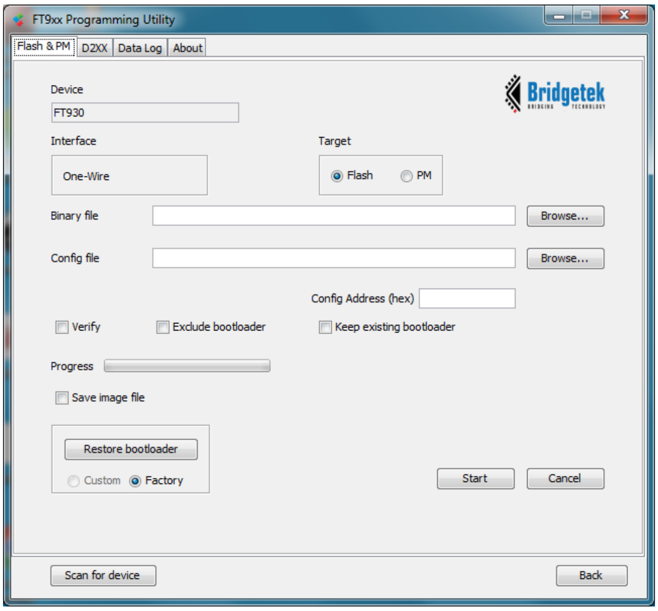


Figure 7- FT9XX Programming Utility One-Wire Option

2.7 D2XX drivers

The D2XX examples in this document depend on FTDI D2XX Windows CDM drivers. When Windows enumerates a D2XX interface, the required driver software is automatically installed by Windows Update. Note that the CDM drivers will not be installed automatically if the default VID and PID are modified. Please contact Bridgetek Pte Ltd for assistance. The D2XX interfaces are enumerated and appear as 'USB Serial Converter X' where X is an alphabet starting from A in the Windows Device Manager. The USB Serial Converter may be accessed as a regular D2XX device (through the D2XX API) or as a Virtual COM Port (VCP) device. In order to access the device as a VCP device, select the 'Load VCP' option on the property page of the USB Serial Converter. Figure 8 - USB Serial Converter Properties shows the property page and the 'Load VCP' selection.

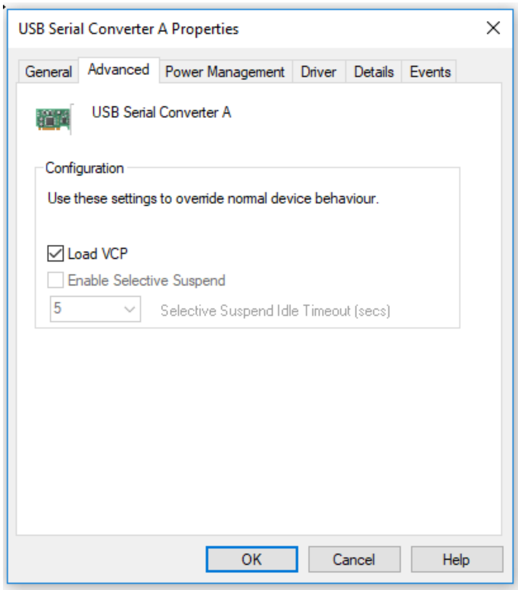


Figure 8 - USB Serial Converter Properties

See [Drivers] and [Installation Guides] for further information on the D2XX CDM driver installation.

3 Examples

Some of the examples require the user to connect a USB to Serial converter cable to UART0 as this port is used to send and receive text. The terminal settings should be 19200 baud rate, unless otherwise a specific baud rate is mentioned in the example. The rest of the settings are 8 data bits, no parity and 1 stop bit. Flow control is not enabled, unless otherwise specified in the example. Set terminal display settings to monospaced (fixed sized) font for best output.

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

Or

- UART0_TXD/GPIO23: available via CN2 Pin 3 on FT930 Mini module
- UART0_RXD/GPIO22: available via CN2 Pin 4 on FT930 Mini module

Or

- UART0_TXD/GPIO23: available via CN1 Pin 4 on MM930Lite module
- UART0_RXD/GPIO22: available via CN1 Pin 6 on MM930Lite module

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 most of the examples is PuTTY. However, any VT100 compatible terminal emulator or any simple serial port (COM) terminal emulation program such as Terminal or Teraterm can be used.

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

Below table gives the matrix of FT9XX examples that can be supported on various hardware boards.

Where,

o - Applicable

x - Not Applicable

FT9XX Examples	MM900 EV1A	MM900 EV2/3A	MM900E V-LITE	MM900E V1B	MM930 Mini	MM930 Lite
ADC Example 1	o	o	o	o	o	o
ADC Example 2	o	o	o	o	o	o
ADC Example 3	o	o	o	o	o	o
BCD Example 1	o	o	o	o	o	o
Camera Example 1	x	o	x	x	x	x
CAN Example 1	o	o	o	o	x	x
CAN Example 2	o	o	o	o	x	x
CAN Example 3	o	o	o	o	x	x
D2XX Example 1	o	o	o	o	o	o
D2XX Example UART Bridge	o	o	o	o	o	o
DAC Example 1	o	o	o	o	o	o
DAC Example 2	o	o	o	o	o	o

FT9XX Examples	MM900 EV1A	MM900 EV2/3A	MM900E V-LITE	MM900E V1B	MM930 Mini	MM930 Lite
DAC Example 3	o	o	o	o	o	o
DLOG Example	o	o	o	o	o	o
Ethernet Example 1	o	o	x	o	x	x
FreeRTOS Example 1	o	o	o	o	o	o
FreeRTOS Example 2	o	o	o	o	o	o
FreeRTOS Example 3	o	o	o	o	o	o
FreeRTOS Example 4	o	o	o	o	o	o
FreeRTOS lwIP Example	o	o	x	o	x	x
FreeRTOS D2XX Example	o	o	o	o	o	o
GPIO Example 1	o	o	o	o	o	o
GPIO Example 2	o	o	o	o	o	o
GPIO Example 3	o	o	o	o	o	o
I2C Master Example 1	x	x	o	x	o	o
I2C Master Example 2	o	o	x	o	x	x
I2C Slave Example 1	o	o	o	o	o	o
I2S Master Example 1	o	o	x	o	x	x
I2S Master Example 2	o	o	x	o	x	x
PWM Example 1	o	o	o	o	o	o
PWM Example 2	o	o	o	o	o	o
PWM Example 3	o	o	o	o	o	o
RTC Example 1	o	o	x	o	o	o
RTC Example 2	o	o	x	o	o	o
RTC External Example 1	o	o	x	x	x	x
RTC External Example 2	o	o	x	x	x	x
SD Host Example 1	o	o	o	o	x	o
SPI Master Example 1	o	o	o	o	o	o
SPI Master Example 2	o	o	o	o	o	o

FT9XX Examples	MM900 EV1A	MM900 EV2/3A	MM900E V-LITE	MM900E V1B	MM930 Mini	MM930 Lite
SPI Master Example 3	o	o	o	o	o	o
SPI Slave Example 1	o	o	o	o	o	o
Timer Example 1	o	o	o	o	o	o
Timer Example 2	o	o	o	o	o	o
Timer Example 3	o	o	o	o	o	o
UART Example 1	o	o	o	o	o	o
UART Example 2	o	o	o	o	o	o
UART Example 3	o	o	o	o	o	o
UART Example 4	o	o	o	o	o	o
UART 9bit Mode Example	o	o	o	o	x	x
GPIO DFU Example	o	o	o	o	o	o
USBD Example BOMS to SD Card	o	o	o	o	x	o
USBD Example HID	o	o	o	o	o	o
USBD Example HID Bridge	o	o	o	o	o	o
USBD Example CDCACM	o	o	o	o	o	o
USBD Example RNDIS	o	o	x	o	x	x
USBD Example UVC Webcam	x	o	x	x	x	x
USBH_Example Hub	o	o	x	o	x	x
USBH Example HID	o	o	x	o	x	x
USBH Example CDCACM	o	o	x	o	x	x
USBH Example BOMS	o	o	x	o	x	x
USBH Example File System	o	o	x	o	x	x
USBH Example FT232	o	o	x	o	x	x
AOA Examples	o	o	x	o	x	x
Watchdog Example 1	o	o	o	o	o	o

Table 2 - Examples supported in various hardware boards

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

Supported hardware and pin information:

Purpose	MM900 EVxA	MM900EVx B	MM900 EV-LITE	MM930 Mini	MM930 Lite
ADC1 pin	CN3 Pin 30	CN3 Pin 30	CN1 Pin 30	CN2 Pin 24	CN2 Pin 24

Connect a voltage source to the *ADC1* pin connect as shown in Figure 9.

This could be a potentiometer with the ends connected to 3.3V and GND, with the wiper connected to ADC1.

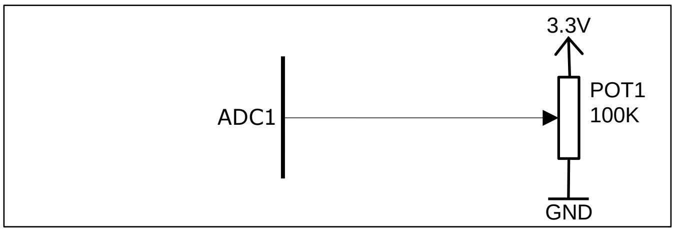


Figure 9 - Circuit Diagram for ADC Examples

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

3.1.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to ADC Example 1...

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

2. Apply an input voltage between 0 and 3.3V to *ADC1*. This should cause the value to change. For example,
3.

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to ADC Example 1...

Poll ADC 1 continuously.
--------------------------------------------------------------------

ADC 1 = 0123
```

Note that 1023 (0x3FF) is the maximum ADC value in case of 10 bit ADC in FT90X and is 255 (0xFF) in case of 8 bit ADC in FT93x.

This example enables rail-rail reference bit (bit ADC_EXT_VREF in DAC_ADC_CONF register) before starting the ADC. When rail-rail reference is enabled, the ADC sample values can be read for full input voltage range 0 to 3.3V. When rail-rail is disabled, the ADC value read is 0 or 1023 if input voltage is outside the range of 0.33V to 2.97V.

```
Resolution of the ADC          ADC Reading
-----------------------    =   ------------------
   Voltage (Vcc)               Analog Input Voltage
```

For 10 bit ADC, 3.3V Vcc, the ADC reading for input voltage of 0.2V is 62.

3.1.2 ADC Example 2

3.1.2.1 Purpose

The purpose of adc_example2.c is to use an interrupt to capture ADC channel 1 samples and store them in a circular buffer, and then print 64 samples of ADC channel 1 from circular buffer at regular interval.

3.1.2.2 Setup

Refer to 3.1.1.2 Setup Section.

3.1.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to ADC Example 2...

Use interrupt to capture samples and store them in a circular buffer,
 print 64 samples of ADC 1 from the circular buffer at a regular interval.
--------------------------------------------------------------------
```

2. Apply a voltage between 0 and 3.3V to *ADC1*. This should cause the value to change. For example,

```
143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143
143 143 143 143 143 143 146 142 141 140 140 146 146 146 141 146 146 145 141 140
143 145 146 141 142 146 144 140 144 146 142 141 141 140 142 145 146 146 143 143
143 143 143 143

145 145 145 145 145 141 141 141 141 141 141 141 141 141 141 141 141 141 141 141
141 141 141 141 141 141 141 141 141 141 146 140 140 140 142 142 146 145 145 145
145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145
145 145 145 145
```

Note that 1023 (0x3FF) is the maximum ADC value in case of 10 bit ADC in FT90X and is 255 (0xFF) in case of 8 bit ADC in FT93X.

This example does not enable rail-rail reference bit (bit ADC_EXT_VREF in DAC_ADC_CONF register) before starting the ADC.

3.1.3 ADC Example 3

3.1.3.1 Purpose

Use ADC interrupt to capture multiple channels ADC samples and stored in circular buffer, print 16 samples of different ADC channel in regular interval.

3.1.3.2 Setup

Supported hardware and pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
ADC1	CN3 Pin 30	CN3 Pin 30	CN1 Pin 30	CN2 Pin 24	CN1 Pin 36
ADC2	CN3 Pin 29	CN3 Pin 29	CN1 Pin 29	CN2 Pin 53	CN1 Pin 35
ADC3	CN3 Pin 27	CN3 Pin 27	CN1 Pin 27	CN2 Pin 52	CN1 Pin 37
ADC4	CN3 Pin 28	CN3 Pin 28	CN1 Pin 28	-na-	-na-
ADC5	CN3 Pin 31	CN3 Pin 31	CN1 Pin 31	-na-	-na-
ADC6	CN3 Pin 32	CN3 Pin 32	CN1 Pin 32	-na-	-na-
ADC7	CN3 Pin 33	CN3 Pin 33	CN1 Pin 33	-na-	-na-

Connect as shown in Figure 9

Different ADC channels should connect to different potentiometers like setup of ADC Example 1.

Only FT90X Revision C supports 12.5MHz and 6.25MHz (half-rate) frequencies and 10-bit and 8-bit resolutions using macros 'SWITCH_FREQUENCY' and 'SWITCH_RESOLUTION' in adc_example3.c. Enable the macros respectively to select 'half rate' and 8-bit resolution.

3.1.3.3 Execution

3. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to ADC Example 3...

Use interrupt to capture samples from multiple ADC channels,
 store them in a circular buffer and print 16 samples of
 different ADC channels at regular interval.
--------------------------------------------------------------------
```

4. Apply a voltage between 0 and 3.3V to different *ADC channels*. The code will output ADC1 to ADCn in round robin manner. For example, the following output is captured from MM930Lite module.

```
ADC Channel 1
0078 0078 0077 0078 0078 0078 0078 0077 0077 0078 0078 0078 0077 0078 0077 0079
ADC Channel 2
0917 0910 0911 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912
ADC Channel 3
0918 0910 0911 0911 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912
ADC Channel 4
0631 0629 0629 0629 0629 0629 0629 0629 0630 0632 0628 0630 0629 0630 0629 0628
ADC Channel 5
0641 0631 0630 0629 0629 0629 0629 0630 0629 0622 0642 0629 0631 0630 0628 0632
ADC Channel 6
0628 0629 0629 0629 0630 0629 0630 0629 0629 0631 0626 0630 0627 0630 0628 0630
ADC Channel 7
1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023
ADC Channel 1
0078 0077 0078 0078 0078 0077 0078 0077 0078 0078 0078 0078 0078 0077 0078 0078
ADC Channel 2
0918 0911 0912 0912 0912 0912 0912 0912 0912 0913 0912 0912 0912 0912 0913 0912
ADC Channel 3
0917 0912 0911 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912 0912
ADC Channel 4
0630 0628 0629 0619 0623 0626 0634 0624 0631 0631 0629 0631 0628 0631 0629 0629
ADC Channel 5
0627 0630 0629 0630 0628 0630 0629 0630 0629 0630 0629 0630 0629 0630 0630 0629
ADC Channel 6
0639 0627 0630 0629 0629 0630 0629 0629 0630 0629 0630 0630 0629 0630 0630 0629
ADC Channel 7
1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023 1023
```

Note that 1023 (0x3FF) is the maximum ADC value in case of 10 bit ADC in FT90X revision B and revision C and is 255 (0xFF) in case of 8 bit ADC in FT93X and if configured for 8 bit in case of FT90X revision C.

Also, this example does not enable rail-rail reference bit (bit ADC_EXT_VREF in DAC_ADC_CONF register) before starting the ADC.

3.2 BCD Examples

The USB ports on personal computers are convenient places for USB devices such as the MM900EVxA 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 FT9XX 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 a maximum of 500mA.
- **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. This port is capable of supplying charge currents beyond 1.5A.
- **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 up to 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 FT9xx device is connected to – a SDP, DCP or CDP port.

3.2.1.2 Setup

Connect a USB to Serial converter cable to UART0 as this port is used to send debug text. Connect the FT900/FT930 USB device port to a SDP port of USB host, or connect to DCP ports of a USB power source like Wall Warts, auto adapters and power banks.

3.2.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 sensor with the FT90X. The MM900EV2A / MM900EV3A come with an Omni vision OV9655 color camera sensor.

3.3.1.2 Setup

Connect an Omnivision OV9655 or OV7670 camera sensor to the camera sensor interface of the FT90X if not already connected to CN13 or CN14 on the FT90X EVM.

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

3.3.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
CAN1_RXD	CN3 Pin 40	CN3 Pin 40	CN1 Pin 40	-na-	-na-
CAN1_TXD	CN3 Pin 38	CN3 Pin 38	CN1 Pin 38	-na-	-na-
CAN0_RXD	CN3 Pin 37	CN3 Pin 37	CN1 Pin 37	-na-	-na-
CAN0_TXD	CN3 Pin 36	CN3 Pin 36	CN1 Pin 36	-na-	-na-

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/EV-Lite hardware does not have CAN transceivers onboard so these need to be externally connected.

Connect the following as shown in Figure 10

1. Connect *CAN1_RXD* to the *RXD* pin of IC1.
2. Connect *CAN1_TXD* to the *TXD* pin of IC1.
3. Connect *CAN0_RXD* to the *RXD* pin of IC2.
4. Connect *CAN0_TXD* 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*

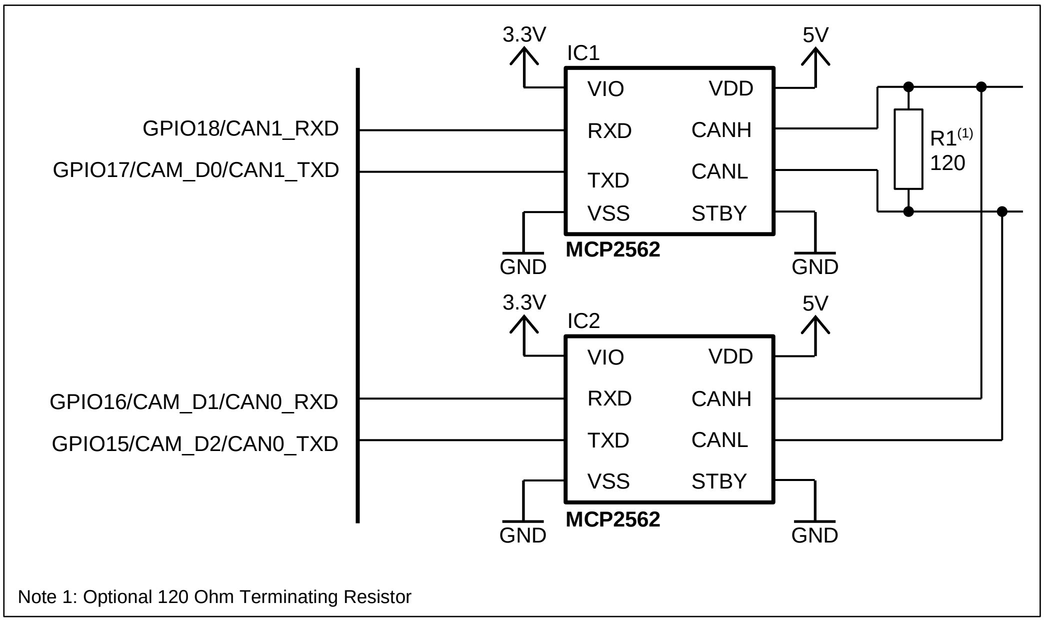


Figure 10 - Circuit diagram for CAN Examples

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

3.4.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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:

```
Copyright (C) Bridgetek Pte 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. This example can only be run on FT90X.

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:

```
Copyright (C) Bridgetek Pte 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/FT930 can be configured and enumerated as FTDI D2XX devices and additionally be enabled as Virtual Com Ports (VCP). The example demonstrates data loop back from a terminal PC application and the example firmware application over the D2XX channel.

3.5.1.2 Setup

Connect the development board programmed with D2XX_Example1.bin to the host PC via USB. When the development board has been enumerated as a D2XX device, drivers are automatically installed from Windows Update. The default VID and PID combination is included in FTDI driver release 2.12.14 and later. More details on driver installation are in Section 2.7.

Additionally, connect a USB to Serial converter cable 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, Bridgetek Pte. Ltd.
--------------------------------------------------------------------
Welcome to D2XX Example 1...

Enter any text on the D2XX port, the same is echoed back on the same port...
--------------------------------------------------------------------
D2XX_Init() returned: 0, No of Interfaces: 3
BUS_RESET
READY
Copyright (C)
```

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.

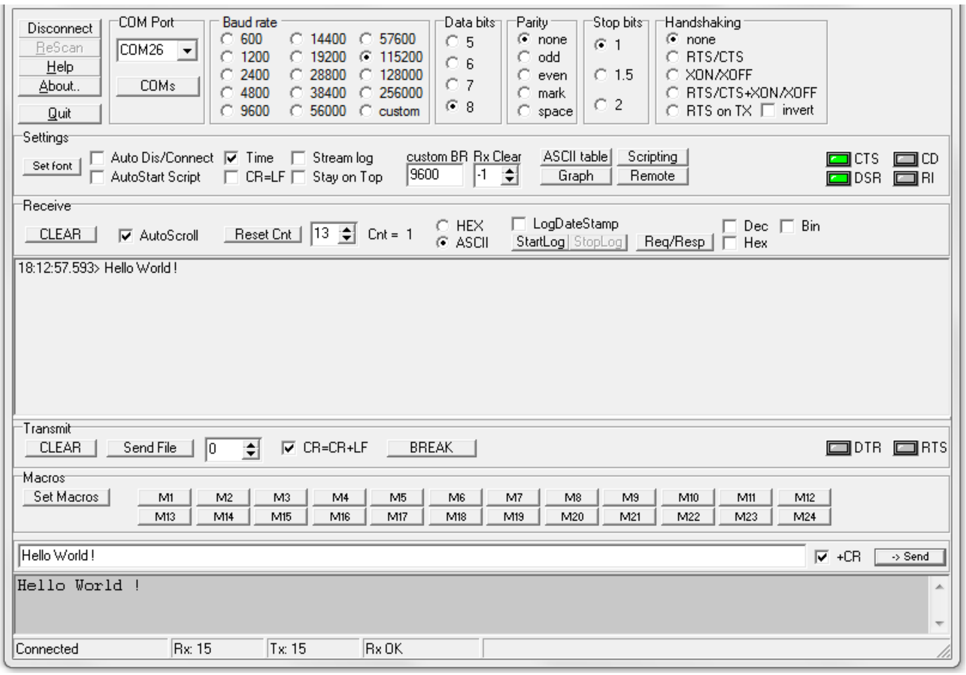
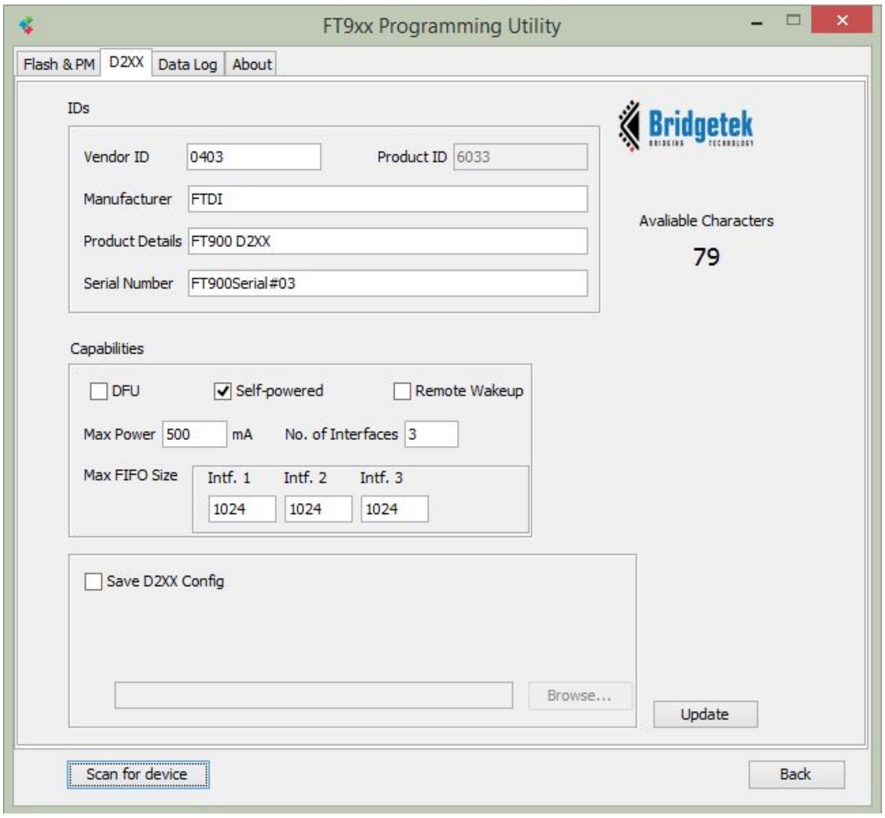


Figure 11 - D2XX Port opened in the PC Terminal application

The default D2XX device configuration settings are available in the example as ft900_d2xx_default_config.inc (or as ft930_d2xx_default_config.inc) header file. This configuration can be read from FT9xx device and changed using the FT9XX Programming GUI Utility's D2XX tab:



Product Page
Document Feedback

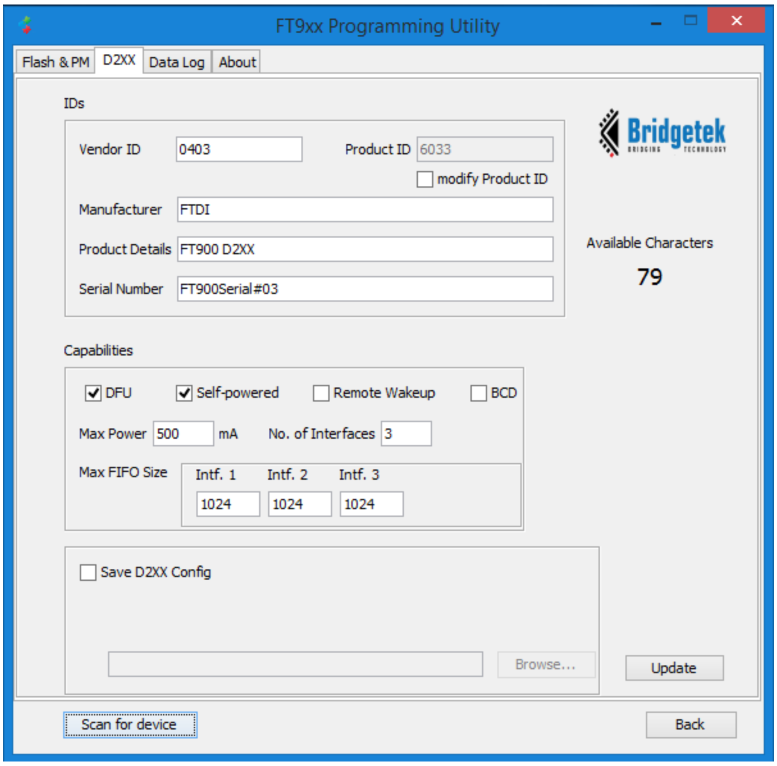


Figure 12 - FT9XX Programming Utility's D2XX tab

The D2XX configuration can be modified and programmed to device's flash using the utility. It can also be saved back as ft900_d2xx_default_config.inc (or as ft930_d2xx_default_config.inc) header file for subsequent compilation usage.

3.5.2 D2XX Example UART Bridge

3.5.2.1 Purpose

The purpose of this example is to demonstrate that FT900/FT930 can be configured and enumerated as FTDI D2XX devices and additionally be enabled as Virtual Com Ports (VCP). The example demonstrates data bridging from a terminal PC application, through the example firmware application over the D2XX channel and UARTs in FT9xx.

3.5.2.2 Setup

Connect the development board programmed with "D2XX_Example UART Bridge.bin" to the host PC via USB. D2XX PC drivers are installed automatically by Windows Update. The default VID and PID combination is included in FTDI driver release 2.12.14 and greater. More details on driver installation in Section 2.7.

The D2XX Channels 1 and 2 are mapped to UART0 and UART1 in case of FT900 and D2XX Channels 1 to 4 are mapped to UART0 to UART4 respectively, in case of FT930. Crossover any two UARTs on the device, Rx to Tx and vice versa.

The UART0 on the FT9XX doubles up as a port to send debug text from the example application. A USB to Serial converter cable can be attached to UART0 to view the debug text on a terminal application on PC.

Open up terminal PC application programs for the FT9XX's D2XX Channels appearing as Virtual COM ports.

The port settings for all the above terminal application use-cases are 19200 baud, no parity, 8 data bits, and 1 stop bit.

3.5.2.3 Execution

1. A welcome message should appear like so, if UART0 is connected to PC terminal application.

```
(C) Copyright, Bridgetek Pte. Ltd.
----------------------------------------------------------------------
Welcome to D2XX UART Bridge Example...

Enter any text on the D2XX[1,2..4] port, the same is echoed on
the UART[0,1,..3]...
----------------------------------------------------------------------
D2XX_Init() called, Result: 0 Interfaces: 3
```

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 multiple instances of the terminal applications and connect to the virtual COM ports corresponding to the D2XX channels. Enter some text on one terminal application, for e.g. D2XX channel 1, the same text is bridged to UART0, and then crossed over to, say UART1, and returns on D2XX channel2 that is associated to UART1.

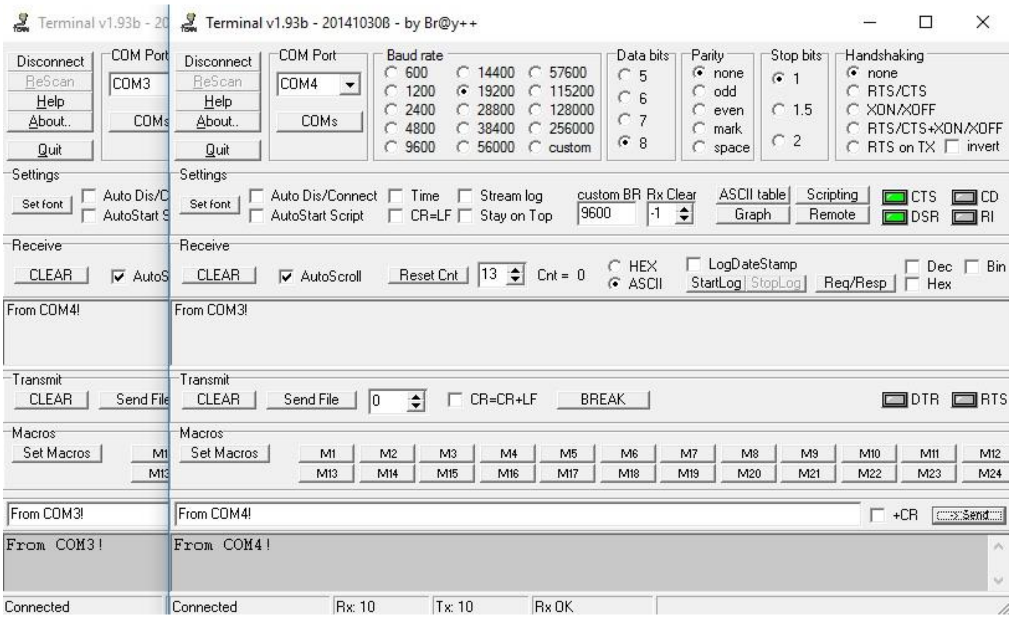


Figure 13 - D2XX Ports opened in the PC Terminal application

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

Supported hardware and pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
DAC0	CN3 Pin 35	CN3 Pin 35	CN1 Pin 35	CN2 Pin 17	CN1 Pin 9

Connect DAC0 to an oscilloscope. Also connect GND.

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

3.6.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 *DAC0* as shown in Figure 14.

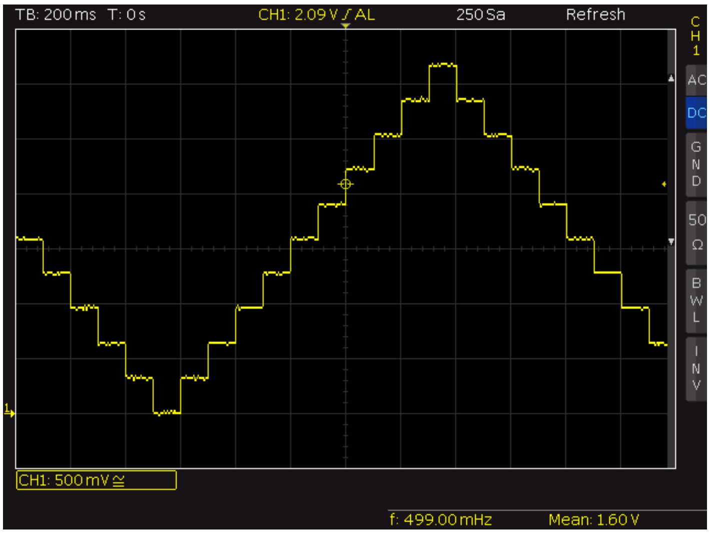


Figure 14 - Output from dac_example1.c

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 load new data.

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:

```
Copyright (C) Bridgetek Pte 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 *DAC0* as shown in Figure 15.

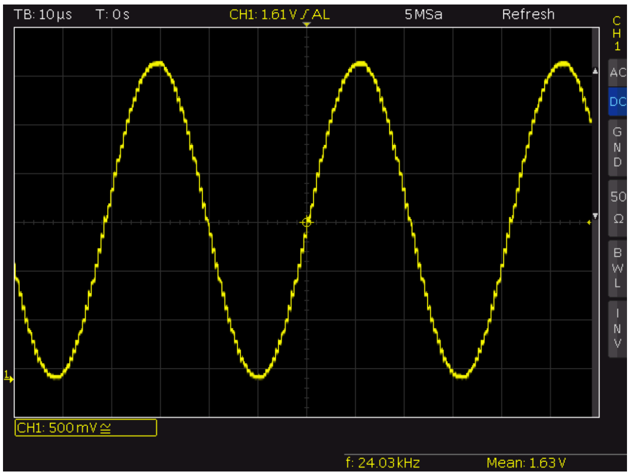


Figure 15 - Output from dac_example2.c

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:

```
Copyright (C) Bridgetek Pte 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 *DAC0* as shown in Figure 15.

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. The allocated sector has 16 pages and pages 0 and 15 are reserved by the library. Pages 1 to 14 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 FT9xx development board programmed with DLOG_Example1.bin to the host PC via USB.

Additionally, connect a USB to Serial converter cable 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,

```
Copyright (C) Bridgetek Pte 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 datalog partition address in flash, page size and the number of pages in the sector used by datalogger.

```
__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 operation 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 cable 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 3.9.6.2.1

3.8.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to Ethernet Example 1...

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

2. Details will appear showing the MAC address[1] 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:

```
ping -c 4 192.168.1.55
```

1

[1]MM900EVxA and MM900EV1B development modules provide a unique 48-bit MAC address for Ethernet applications,

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 16.

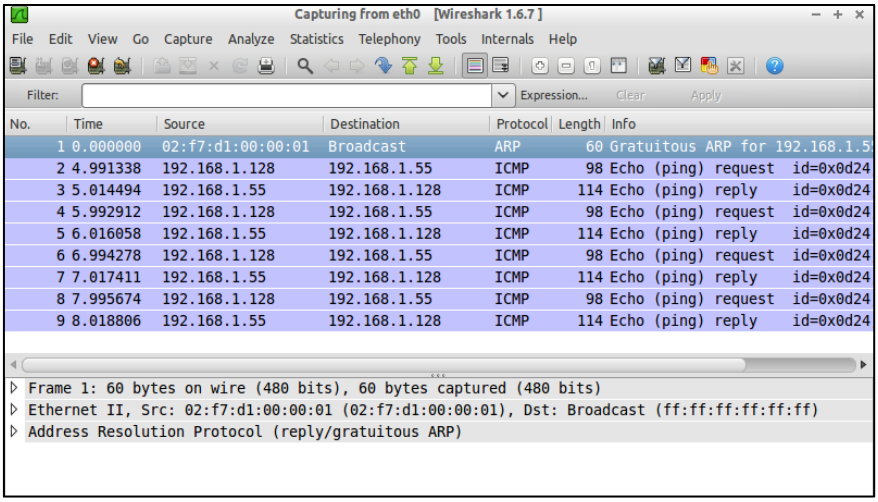


Figure 16 - Wireshark output for eth_example1.c

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.

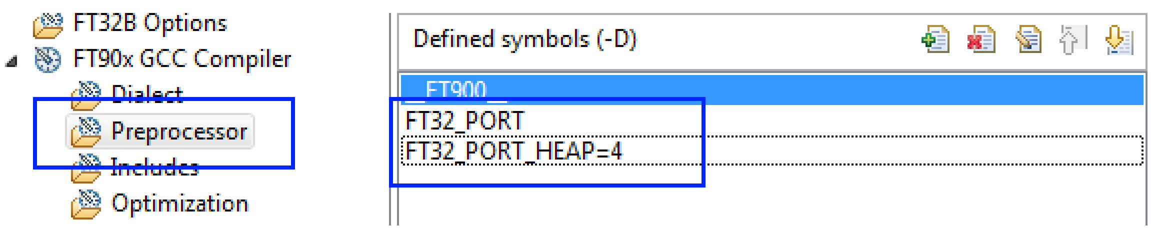
3.9.1 Setup (common for all FreeRTOS projects)

To use the FT9XX FreeRTOS port within projects, some special configurations/includes in Eclipse tool are required that are different from the usual example projects. These are already set in the example projects but are summarized here:

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

1. Preprocessor 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 alternate heap management strategies from FreeRTOS. Since strategy 4 is the most flexible, it has been selected for the default ~~FT900~~ FT9xx port. For more information on the heap management strategies available with FreeRTOS refer to http://www.freertos.org/a00111.html

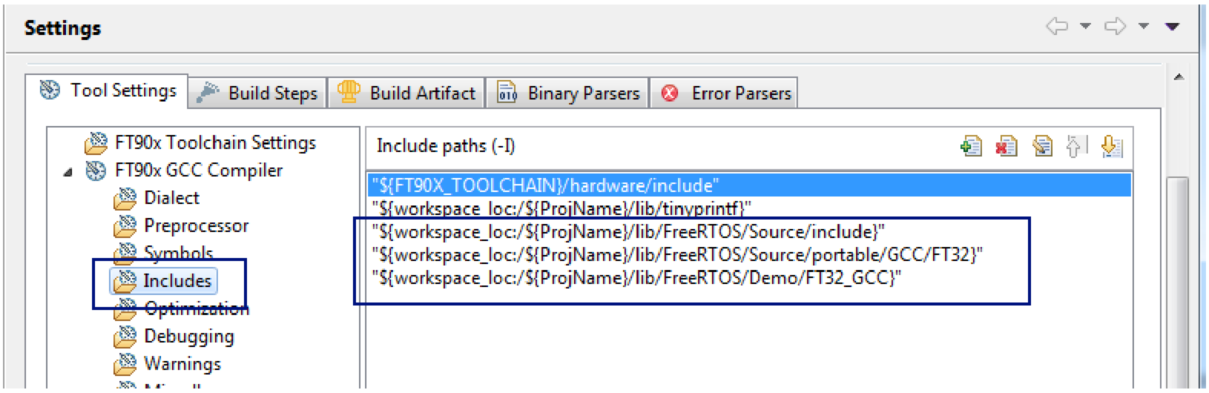


2. Set the include paths to various directories under the FreeRTOS folder structure as shown below. Key directories are:

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

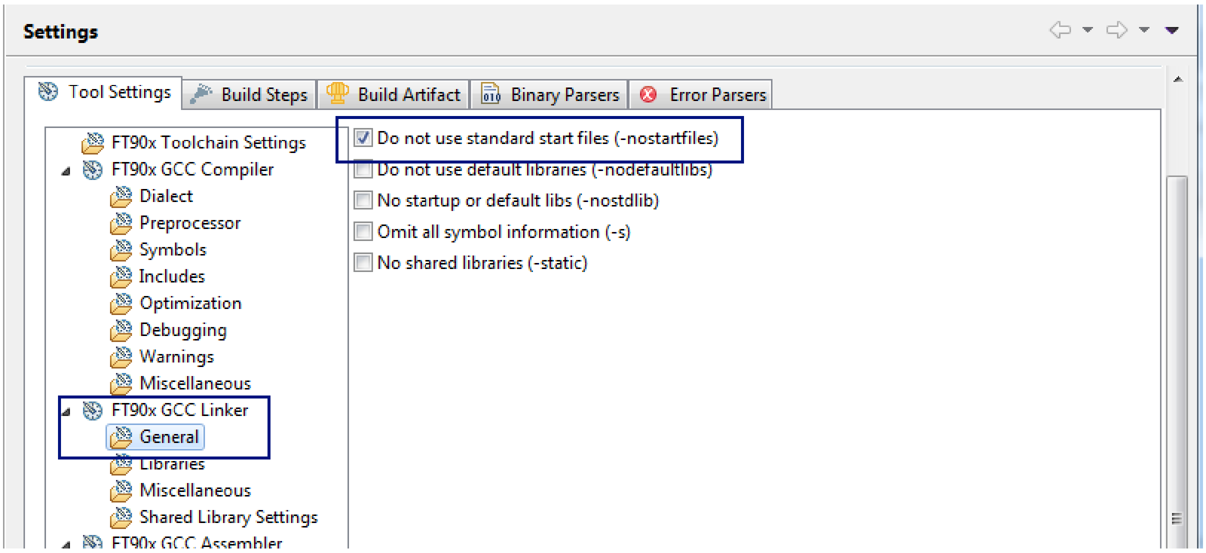
   FreeRTOS\Source\portable\GCC\FT32 - Files specific to FT900 port.

   FreeRTOS\Demo\FT32_GCC – crt0 (C runtime zero) and configuration file for the port.



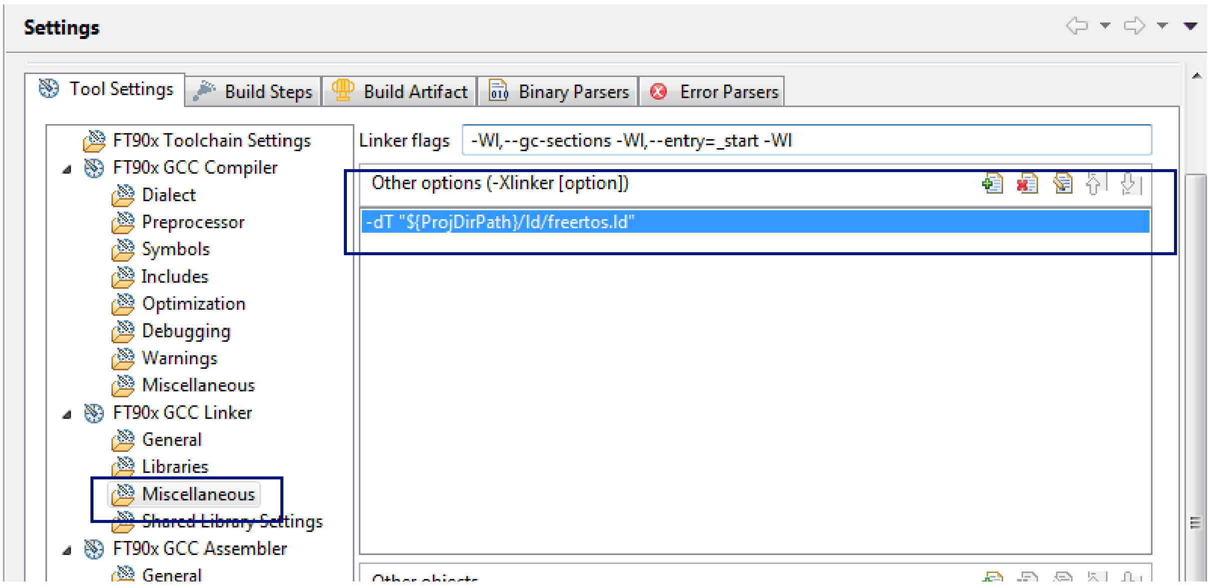
3. Select the **–nostartfiles** linker option.

   The FreeRTOS port uses a custom crt0 file available at *FreeRTOS\Demo\FT32_GCC\crt0.S*

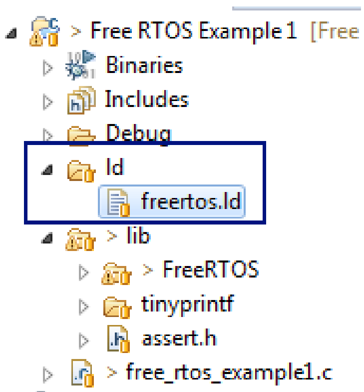


4. Set the linker options to use the custom linker script: **-dT "${ProjDirPath}/ld/freertos.ld**"

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 on FT900 and GPIO26 and GPIO27 on FT930** (CN3 pins 7 and 9 on the MM900EVxA or CN1 1 and 2 on the FT930 Mini Module or CN1 34 and 33 on the MM930Lite).

Open a COM port on the PC to UART0 on the evaluation board to view the logs printed out from FT9XX. The baud rate to be configured is **19200bps.** Set terminal display settings to monospaced (fixed sized) font for best output.

3.9.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 evaluation board to view the logs printed out from FT9xx. The baud rate to be configured is **19200bps.** Set terminal display settings to monospaced (fixed sized) font for best output.

3.9.3.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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:

```
 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 evaluation board to view the logs printed out from FT9XX. The baud rate to be configured is **19200bps.** Set terminal display settings to monospaced (fixed sized) font for best output.

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, and 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 (C) Bridgetek Pte 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
```

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

3.9.4.3.2 Demo 2

1. A welcome message will appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to Free RTOS Test Example 3...

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

2. 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
```

3. 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*
```

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

3.9.4.3.3 Demo 3

1. A welcome message will appear like so:

```
Copyright (C) Bridgetek Pte Ltd
---------------------------------------------------------------------
Welcome to Free RTOS Test Example 3...

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

2. 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
```

3. 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 ##########################################
```

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

3.9.5 FreeRTOS Example 4

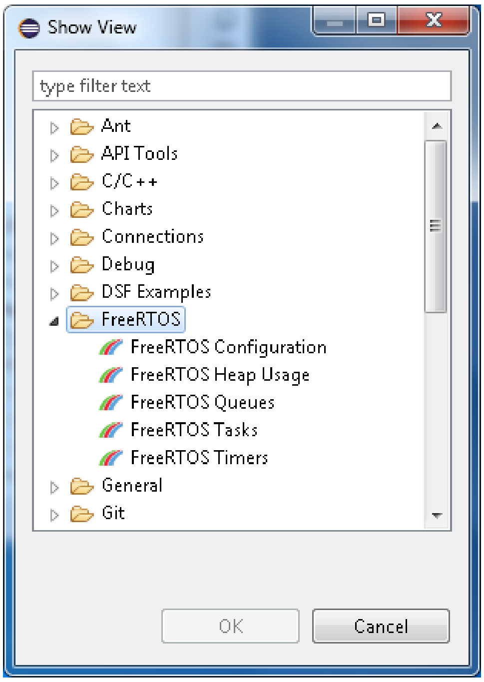
3.9.5.1 Purpose

The purpose of this example is to demonstrate the FT9XX Eclipse feature for FreeRTOS Kernel-Aware Debugging. This feature enables monitoring of FreeRTOS resources such as tasks, queues, timers and heap to help developers debug their applications faster.

3.9.5.2 Setup

Open a COM port on the PC to UART0 on the evaluation board to view the logs printed out from FT9XX. The baud rate to be configured is **19200bps.** Set terminal display settings to monospaced (fixed sized) font for best output.

Open the Eclipse views for FreeRTOS monitoring via Window -> Show View -> Other



3.9.5.3 Execution

1. Add a breakpoint inside the loop of **prvPrintTask** and Debug the application:

free_rtos_example4.c

```
static void prvPrintTask(void *pvParameters)
{
    char *pcStringToPrint;
    int max = 0x1FF;
    int random = 0;

    pcStringToPrint = (char *)pvParameters;
    for(int i=0; ; i++)
    {
        random = rand() & max;
        prvNewPrintString(pcStringToPrint);
#if ( INCLUDE_vTaskDelay == 1 )
        vTaskDelay(random);
#endif
        deleteResource(pcStringToPrint);
    }
}
```

2. The application will stop when the breakpoint is triggered. Once the breakpoint is triggered, select any of the FreeRTOS views.

FreeRTOS Tasks | FreeRTOS Queues | FreeRTOS Timers | FreeRTOS Heap Usage

Task TCB#	Task Name	Task Address	Task Priority	Task State	Task Stack Start	Task Stack Top	Task Stack Usage	Task Event Object	Task Runtime
1	Print0	0x00801d0c	1	READY	0x00800d04	0x00801c74	3952	None	0.00%
2	Print1	0x00802d80	1	READY	0x00801d78	0x00802ce8	3952	None	0.00%
3	Print2	0x00803df4	2	READY	0x00802dec	0x00803d5c	3952	None	0.00%
4	Print3	0x00804e68	3	READY	0x00803e60	0x00804dd0	3952	None	0.00%
5	Print4	0x00805edc	4	READY	0x00804ed4	0x00805e44	3952	None	0.00%
6	Print5	0x00806f50	5	RUNNING	0x00805f48	0x00806eb8	3952	None	0.00%
7	Print6	0x00807fc4	6	BLOCKED	0x00806fbc	0x00807f14	3928	None	99.97%
8	IDLE	0x0080898c	0	READY	0x00808584	0x008088f4	880	None	0.00%
9	Tmr Svc	0x00808e00	9	BLOCKED	0x008089f8	0x00808d48	848	0x008083b0	0.03%

FreeRTOS Tasks | FreeRTOS Queues | FreeRTOS Timers | FreeRTOS Heap Usage

Queue Name	Queue Address	Queue # Tx Waiting	Queue # Rx Waiting	Queue Current Length	Queue Max Length	Queue Item Size	Queue Type
CS1	0x0080816c	0	0	1	4	0	Counting ...
CS2	0x008081c4	0	0	0	5	0	Counting ...
CS3	0x0080821c	0	0	1	6	0	Counting ...
MU1	0x00808274	0	0	1	1	0	Mutex
MU2	0x008082cc	0	0	1	1	0	Mutex
MU3	0x00808324	0	0	1	1	0	Mutex
MUX	0x00800cac	0	0	1	1	0	Mutex
QU1	0x00808030	0	0	0	1	5	Queue
QU2	0x00808090	0	0	0	2	7	Queue
QU3	0x008080f8	0	0	0	3	9	Queue
TmrQ	0x008083b0	0	1	0	10	12	Queue

FreeRTOS Tasks | FreeRTOS Queues | FreeRTOS Timers | FreeRTOS Heap Usage

Timer ID	Timer Name	Timer Address	Timer Period	Timer Reload	Timer Callback
0x00000001	OneShotTmr1	0x0080837c	1000	Off	TimerCallbac...
0x00000002	OneShotTmr2	0x00808480	10000	Off	TimerCallbac...
0x00000003	OneShotTmr3	0x008084b4	30000	Off	TimerCallbac...
0x00000004	AutoReload1	0x008084e8	5000	Auto	TimerCallbac...
0x00000005	AutoReload2	0x0080851c	100000	Auto	TimerCallbac...
0x00000006	AutoReload3	0x00808550	300000	Auto	TimerCallbac...

FreeRTOS Tasks | FreeRTOS Queues | FreeRTOS Timers | FreeRTOS Heap Usage

Heap Block Base Address	Heap Block End Address	Heap Block Size	Heap Block Available Size	Heap Type
0x00800c9c	0x00808e64	33224	0	Heap 4
0x00808e64	0x0080d49c	17976	17976	Heap 4

3. Continue debugging the application by pressing F8 (Resume) to resume the debugging. The application will stop when the breakpoint is triggered again. Once the breakpoint is triggered, select the FreeRTOS Tasks view.

FreeRTOS Tasks | FreeRTOS Queues | FreeRTOS Timers | FreeRTOS Heap Usage

Task TCB#	Task Name	Task Address	Task Priority	Task State	Task Stack Start	Task Stack Top	Task Stack Usage	Task Event Object	Task Runtime
1	Print0	0x00801d0c	1	READY	0x00800d04	0x00801c74	3952	None	0.00%
2	Print1	0x00802d80	1	READY	0x00801d78	0x00802ce8	3952	None	0.00%
3	Print2	0x00803df4	2	READY	0x00802dec	0x00803d5c	3952	None	0.00%
4	Print3	0x00804e68	3	READY	0x00803e60	0x00804dd0	3952	None	0.00%
5	Print4	0x00805edc	4	READY	0x00804ed4	0x00805e44	3952	None	0.00%
6	Print5	0x00806f50	5	READY	0x00805f48	0x00806e20	3952	None	47.19%
7	Print6	0x00807fc4	6	RUNNING	0x00806fbc	0x00807f14	3928	None	52.79%
8	IDLE	0x0080898c	0	READY	0x00808584	0x008088f4	880	None	0.00%
9	Tmr Svc	0x00808e00	9	BLOCKED	0x008089f8	0x00808d48	848	0x008083b0	0.02%

Observe that the runtime percentage of the tasks changes. The active task also changed from Print5 Task to Print6 Task.

4. Press F8 several times then examine the changes in the other FreeRTOS views.

For more information regarding the FreeRTOS views, refer to the FT9XX Installation Guide: AN_325_FT9xx_Tool_Chain_Installation_Guide.pdf

3.9.6 FreeRTOS 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 of program memory and about 4kB of 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 lwIP project website.

3.9.6.1 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** or **dynamic IP** addresses. A DHCP server is required in the network if dynamic IP addresses are used. Two companion Python scripts are provided in the `/Scripts` directory which can be run from a PC to test the demos.

3.9.6.2 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 address using the preprocessor switch **`USE_DHCP`**, a value of 1 selects dynamic IP address and a value of 0 selects Static IP address. Default configuration is a server demo with static IP address.

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

3.9.6.2.1 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 an static IP address of **192.168.1.10** as shown in Figure 18. On Windows 7, the LAN Connection properties can be found in Control Panel > Network and Internet > Network and Sharing Center as shown in Figure 17.

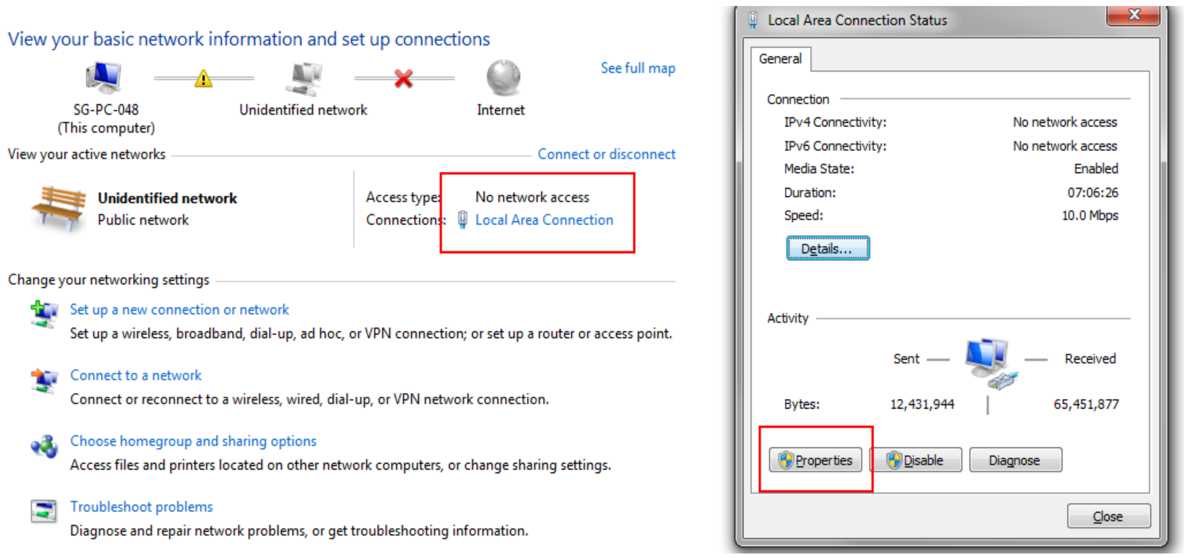


Figure 17 - Windows 7 LAN Properties

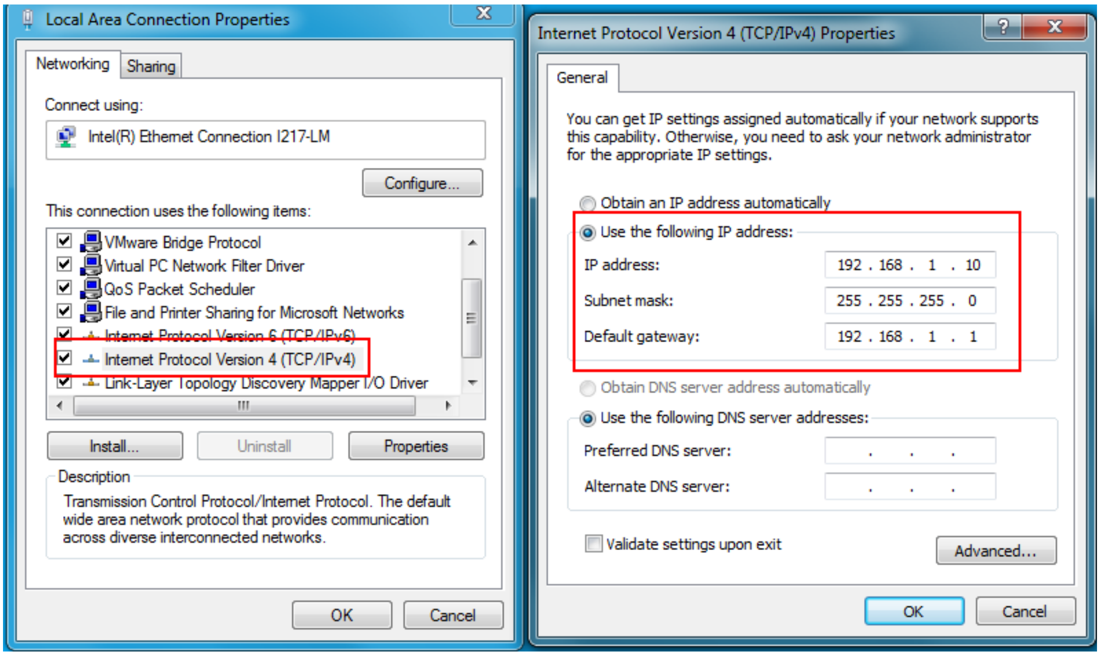


Figure 18 - Host PC static IP configuration

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

3.9.6.2.2 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.

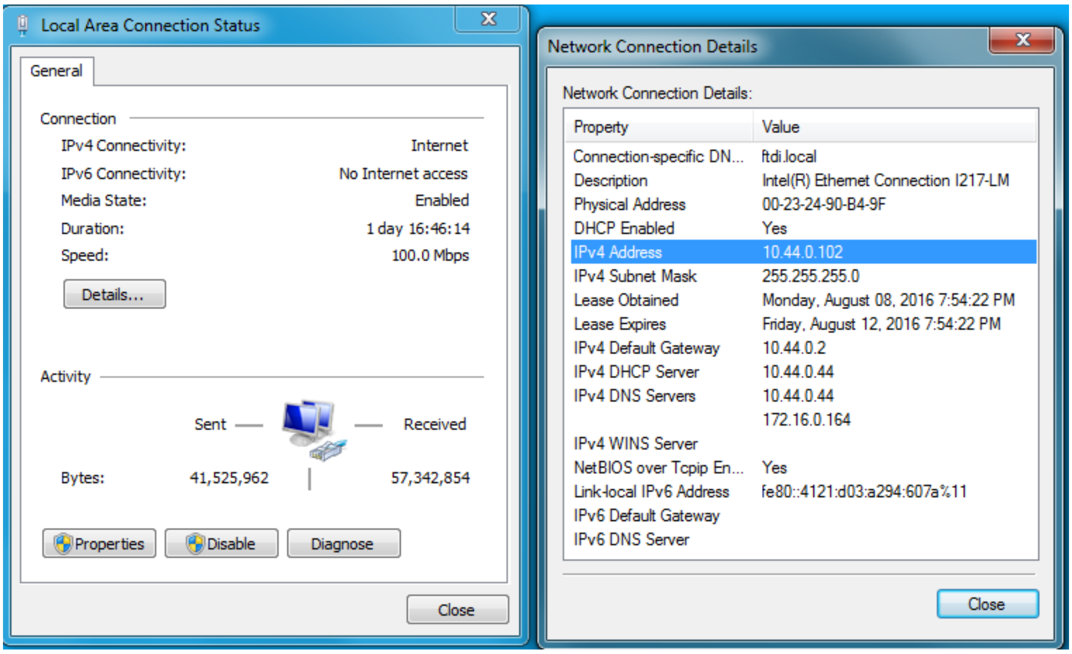


Figure 19 - Windows 7 - Find host PC IP address

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

Figure 20 - Update host PC IP address

3.9.6.3 Execution

3.9.6.3.1 Server Demo

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 address configuration the IP should be **192.168.1.190** and for a dynamic IP address configuration, the address printed in the UART logs must be used. An example is shown in Figure 21 (for a dynamic IP address 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
```

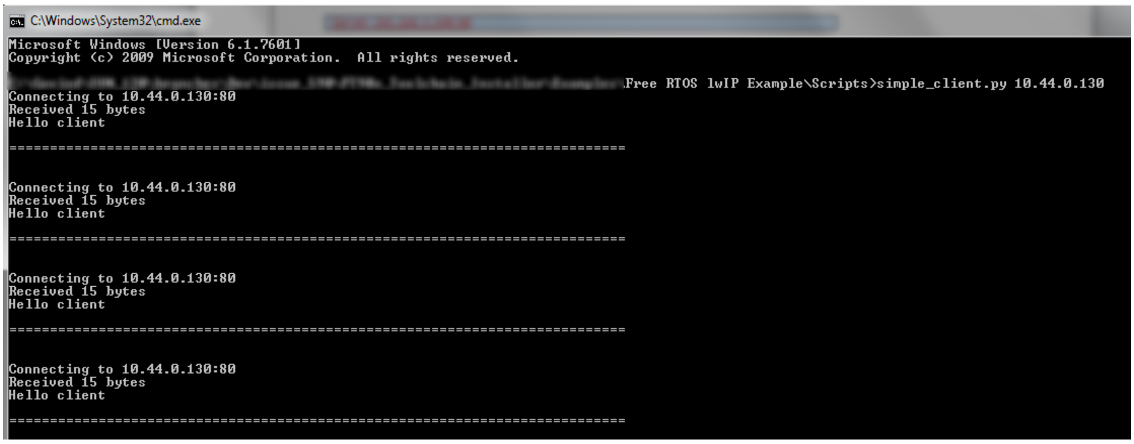


Figure 21 - Simple TCP Client running on Host PC (FT900 dynamic IP address is 10.44.0.120)

3.9.6.3.2 Client Demo

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 address configuration, the address should be **192.168.1.10** and for dynamic IP address, the appropriate host PC IP address should be used (refer section 3.9.6.2). 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 22 - Simple TCP server running on a host PC

3.9.7 FreeRTOS D2XX Example

3.9.7.1 Purpose

The purpose of this example is to demonstrate the Free RTOS port of the D2XX Example 1. The FT900/FT930 device reports 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.

3.9.7.2 Setup

Connect the development board programmed with FreeRTOS D2XX Example.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. More details on driver installation in Section 2.7.

Additionally, connect a USB to Serial converter cable 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.

3.9.7.3 Execution

1. A welcome message should appear like so,

```
(C) Copyright, Bridgetek Pte. Ltd.
--------------------------------------------------------------------
--------------------------------------------------------------
Welcome to D2XX with FreeRTOS
D2XX_Init() called, Result: 0 Interfaces: 3
D2XX tasks created
TaskScheduler started
vD2XX_UserTask: (intf=1) started
vD2XX_UserTask: (intf=2) started
vD2XX_UserTask: (intf=3) started
```

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.

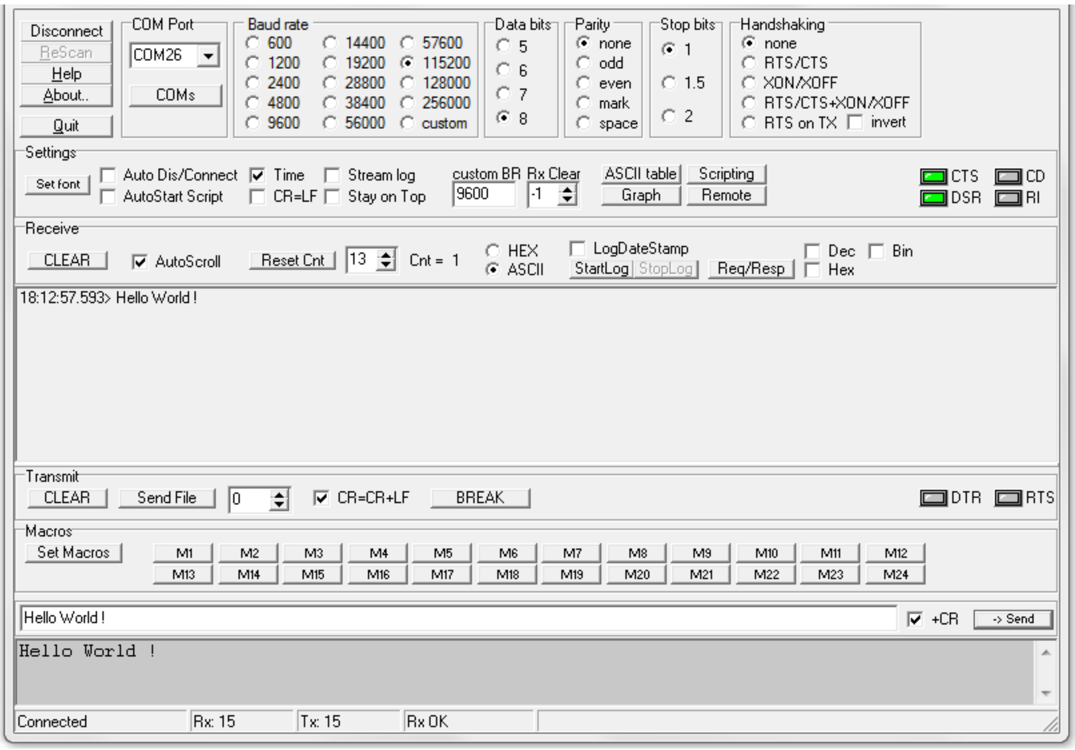


Figure 23 - D2XX Port opened in the PC Terminal application

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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
GPIO18	CN3 Pin 40	CN3 Pin 35	CN1 Pin 40	CN2 Pin 10	CN1 Pin 29

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

Connect something to *GPIO18* to monitor the state of the pin (e.g. an LED).

3.10.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to GPIO Example 1...

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

2. GPIO18 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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
GPIO18	CN3 Pin 40	CN3 Pin 35	CN1 Pin 40	CN2 Pin 10	CN1 Pin 29

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

Connect something to *GPIO18* to change the state of the pin.

3.10.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to GPIO Example 2...

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

2. The current state of *GPIO18* 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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
GPIO18	CN3 Pin 40	CN3 Pin 35	CN1 Pin 40	CN2 Pin 10	CN1 Pin 29

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

Connect something to *GPIO18* to change the state of the pin.

3.10.3.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to GPIO Example 3...

Use interrupts to inform the user of a falling edge on a GPIO pin.
--------------------------------------------------------------------
```

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

```
Pin Interrupted!
```

3.11 I²C Master Examples

3.11.1 I²C Master Example 1

3.11.1.1 Purpose

The purpose of this example is to demonstrate the use of the I²C master peripheral and how to transfer data to and from it. This test can be performed on hardware boards without on-board EEPROM.

3.11.1.2 Setup

Pin information:

Purpose			**MM900 EV-Lite**	**MM930 Mini**	**MM930 Lite**
I2C1_SCL			CN1 Pin 25	CN2 Pin 15	CN1 Pin 40
I2C1_SDA			CN1 Pin 26	CN2 Pin 16	CN1 Pin 38

This test uses a 24LC01 1Kbit EEPROM as an I²C slave device.

Connect as shown in Figure 24

1. Connect the *SCL* pin of the 24LC01 to the *I2C1_SCL*.
2. Connect the *SDA* pin of the 24LC01 to the *I2C1_SDA*.
3. Connect the *WP*, *A0*, *A1*, *A2* and *GND* pins of the 24LC01 to Ground.

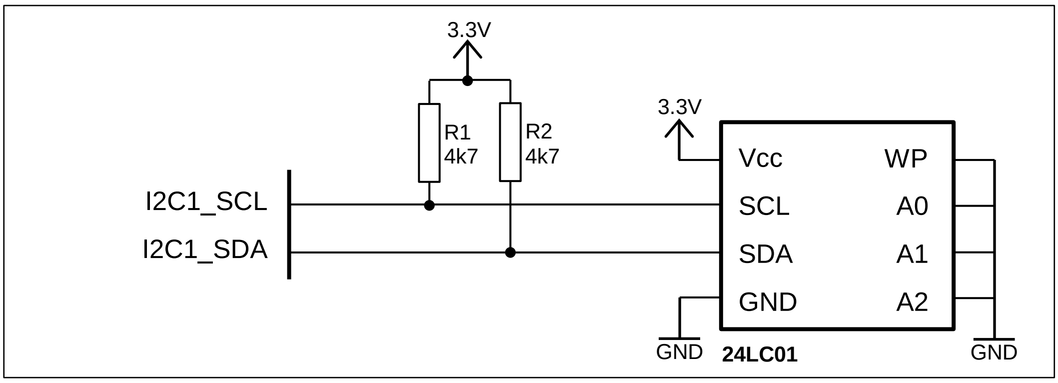


Figure 24 - Circuit Diagram for I2C Master Examples

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

3.11.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 FF  | ................
0x0010: FF 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 FF  | ................
0x0030: FF 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 FF  | ................
0x0050: FF 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 FF  | ................
0x0070: FF 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_h 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 FF  | ................
0x0010: FF 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 FF  | ................
0x0030: FF 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 FF  | ................
0x0050: FF 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 FF  | ................
0x0070: FF 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_h 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.
```

3.11.2 I²C Master Example 2

3.11.2.1 Purpose

The purpose of this example is to demonstrate the use of the I²C 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:

```
Copyright (C) Bridgetek Pte 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 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 BB  | ...............
```

3.12 I²C Slave Examples

3.12.1 I²C Slave Example 1

3.12.1.1 Purpose

The purpose of this example is to demonstrate the use of the I²C slave peripheral and how to transfer data to and from it.

3.12.1.2 Setup

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
I2C1_SCL	CN3 Pin 25	CN3 Pin 25	CN1 Pin 25	CN2 Pin 15	CN1 Pin 40
I2C1_SDA	CN3 Pin 26	CN3 Pin 26	CN1 Pin 26	CN2 Pin 16	CN1 Pin 38

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 25

1. Connect the *CLK* pin of the Bus Pirate to the *I2C1_SCL*.
2. Connect the *MOSI* pin of the Bus Pirate to the *I2C1_SDA*.
3. Connect the *GND* pin of the Bus Pirate to the *GND* pin of the FT90X

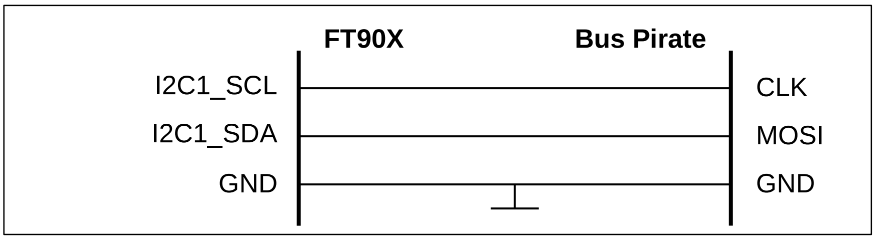


Figure 25 - Circuit Diagram for I2C Slave Examples

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

3.12.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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^{2}C mode:

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

3. On the Bus Pirate, to write data to the FT90X I^{2}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^{2}C bus (38_h).
   - The address pointer is sent on the I^{2}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^{2}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 I²C bus (38_h).
- The address pointer is sent on the I²C bus (4).

- The '[' character will cause a restart condition to occur.
- The read address is sent on the I²C bus (39_h).
- 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 I²S Slave Examples

3.13.1 I²S Master Example 1

3.13.1.1 Purpose

The purpose of this example is to demonstrate I²S in master mode, 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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
AU_OUT	CN10, CN11	CN10	-na-	-na-	-na-
HP_OUT	CN9	CN9	-na-	-na-	-na-

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

Connect a set of speakers to AU_OUT, or connect a set of headphones to HP_OUT.

3.13.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to I2S Master Example 1...

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

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

3.13.2 I²S Master Example 2

3.13.2.1 Purpose

The purpose of this example is to demonstrate the reception and transmission of data over I²S of FT90X in master mode. The FT90X EVM is fitted with a Wolfson Microelectronics WM8731 Codec.

3.13.2.2 Setup

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
AU_OUT	CN10, CN11	CN10	-na-	-na-	-na-
HP_OUT	CN9	CN9	-na-	-na-	-na-

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

Connect a set of speakers to AU_OUT, or connect a set of headphones to HP_OUT.

3.13.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to I2S Master 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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
PWM0	CN3 Pin 13	CN3 Pin 13	CN1 Pin 13	CN1 Pin 29	CN1 Pin 13

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
PWM1	CN3 Pin 14	CN3 Pin 14	CN1 Pin 14	CN1 Pin 30	CN1 Pin 14
PWM2	CN3 Pin 12	CN3 Pin 12	CN1 Pin 12	CN1 Pin 27	CN1 Pin 12

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

Connect the following signals to an oscilloscope for measurement:

- PWM0
- PWM1
- PWM2

3.14.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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. *PWM0* should have a 25% duty cycle wave output on it, *PWM1* should have a 50% duty cycle wave output on it, and *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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
PWM0	CN3 Pin 13	CN3 Pin 13	CN1 Pin 13	CN1 Pin 29	CN1 Pin 13

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

Optionally, connect an LED or oscilloscope to *PWM0.*

3.14.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
---------------------------------------------------------------------
Welcome to PWM Example 2...

Output a PWM signal to drive a fading LED on PWM0
The so called, breathing LED.
---------------------------------------------------------------------
```

2. The output on *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

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
PWM0	CN3 Pin 13	CN3 Pin 13	CN1 Pin 13	CN1 Pin 29	CN1 Pin 13
PWM1	CN3 Pin 14	CN3 Pin 14	CN1 Pin 14	CN1 Pin 30	CN1 Pin 14

A low pass filter will be needed to remove the PWM carrier frequency from the output waveform.

Figure 26 shows the circuit needed to create a 10.047 kHz Low Pass Filter with optional stereo volume control and DC offset removal.

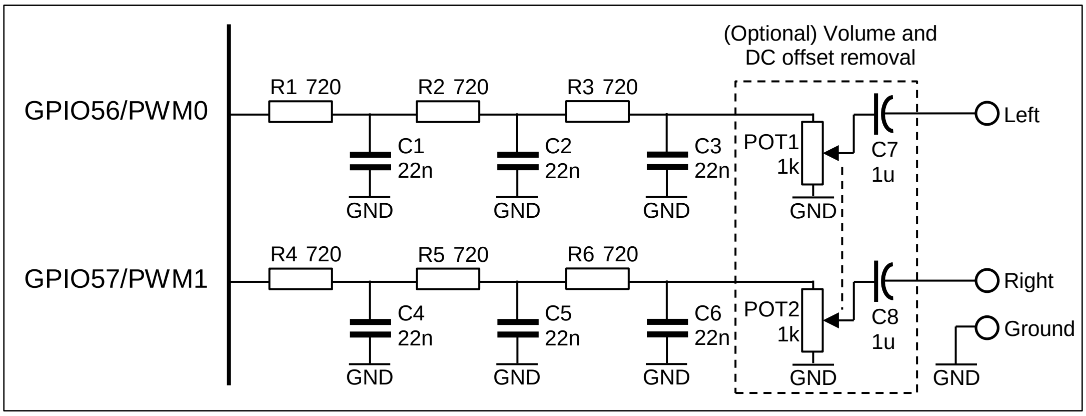


Figure 26 - PWM Low Pass Filter

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

Connect the following signals to an oscilloscope for measurement:

- PWM0.
- PWM1.

3.14.3.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to PWM Example 3...

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

2. *GPIO56/PWM0* and *GPIO57/PWM1 of FT900 (GPIO11/PWM0* and *GPIO10/PWM1 in FT930)* should be outputting PWM waves which represent the Left and Right channels of a Sine wave.

3.15 Real Time Clock (Internal) Examples

The RTC modules on FT90X revision C and revision B MCUs are greatly distinct. For toolchain versions prior to v2.5.0, the software written for FT90X revision B modules cannot be run on FT90X revision C as it is not forward compatible. However, the RTC API from toolchain version 2.5.0 onwards, is backward compatible to RTC in FT900 revision B, i.e. all three modules (MM900EVxA, MM900EVxB, & MM930EVxA) can be supported.

3.15.1 RTC Example 1

3.15.1.1 Purpose

The purpose of this example is to demonstrate the FT90X's on-chip 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 EVM 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.

No special setup is required for the MM900EVxB, MM930Mini Module and MM930Lite.

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

3.15.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 FT90X's on-chip Real Time Clock peripheral. This example demonstrates the time matching capability.

3.15.2.2 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 EVM 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.

No special setup is required for the MM900EVxB, MM930Mini Module and MM930Lite.

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

3.15.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 Real Time Clock (External) Examples

The MM900EVxA development modules provide a 32.768 KHz quartz crystal and load capacitors for an external Real Time Clock (RTC). The two zero ohm resistors (**R141, R142**) are alternatives for the FT900 microcontroller's on-chip RTC and these resistors are not populated by default.

3.16.1 RTC External Example 1

3.16.1.1 Purpose

The purpose of this example is to demonstrate the external Real Time Clock peripheral present on the MM900EVxA boards.

3.16.1.2 Setup

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

3.16.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to External RTC Example 1...

--------------------------------------------------------------------
RTC Time: 09/08/17 Wednesday  09:00:01AM
```

2. Every second, the program will update the pre-set RTC time:

```
09/08/17 Wednesday  09:00:01AM
09/08/17 Wednesday  09:00:02AM
09/08/17 Wednesday  09:00:03AM
09/08/17 Wednesday  09:00:04AM
09/08/17 Wednesday  09:00:05AM
```

3.16.2 RTC External Example 2

3.16.2.1 Purpose

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

3.16.2.2 Setup

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

3.16.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
---------------------------------------------------------------------
Welcome to External RTC Example 2...

---------------------------------------------------------------------
RTC Time: 09/08/17 Wednesday  09:00:01AM
```

2. Alarm triggers after 4 seconds, then onwards the program updates the pre-set RTC time every 2 seconds.

```
Alarm 1:  09/08/17 Wednesday  09:00:05AM
Alarm 1:  09/08/17 Wednesday  09:00:07AM
Alarm 1:  09/08/17 Wednesday  09:00:09AM
Alarm 1:  09/08/17 Wednesday  09:00:11AM
Alarm 1:  09/08/17 Wednesday  09:00:13AM
Alarm 1:  09/08/17 Wednesday  09:00:15AM
```

3.17 SD Host Examples

3.17.1 SD Host Example 1

3.17.1.1 Purpose

The purpose of this example is to demonstrate the SD host controller peripheral.

3.17.1.2 Setup

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

Insert a FAT32 formatted SD card (CN5 on the FT90X EVM or CN7 on the MM930Lite).

3.17.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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
```

3.18 SPI Master Examples

3.18.1 SPI Master Example 1

3.18.1.1 Purpose

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

3.18.1.2 Setup

Pin information:

Purpose	**MM900EVxA**	**MM900 EVxB**	**MM900 EV-Lite**	**MM930 Mini**	**MM930 Lite**
SPIM_SS0	J2 Pin 2	J2 Pin 2	J1 Pin 2	CN1 Pin 5	J1 Pin 2
SPIS0_SS	CN7 Pin 2	CN7 Pin 2	CN5 Pin 2	CN1 Pin 14	CN1 Pin 22
SPIM_SCK	J2 Pin 1	J2 Pin 1	J1 Pin 1	CN1 Pin 9	J1 Pin 1
SPIS0_SCK	CN7 Pin 1	CN7 Pin 1	CN5 Pin 1	CN1 Pin 9	CN1 Pin 18
SPIM_MOSI	J2 Pin 4	J2 Pin 4	J1 Pin 4	CN1 Pin 11	J1 Pin 4
SPIS0_MOSI	CN7 Pin 3	CN7 Pin 3	CN5 Pin 3	CN1 Pin 11	CN1 Pin 20
SPIM_MISO	J2 Pin 3	J2 Pin 3	J1 Pin 3	CN1 Pin 12	J1 Pin 3
SPIS0_MISO	CN7 Pin 4	CN7 Pin 4	CN5 Pin 7	CN1 Pin 12	CN1 Pin 19

This example uses two FT900EVx EV modules or two MM930Mini Modules or two MM930Lite modules, one module as SPI master and other module as SPI slave.

Connect as shown in Figure 27.

1. Connect the *SPIM_SS0* to *SPIS0_SS*.
2. Connect the *SPIM_SCK* to *SPIS0_SCK*.
3. Connect the *SPIM_MOSI* to *SPIS0_MOSI*.
4. Connect the *SPIM_MISO* to *SPIS0_MISO*.
5. Ensure that both the boards have a common ground.

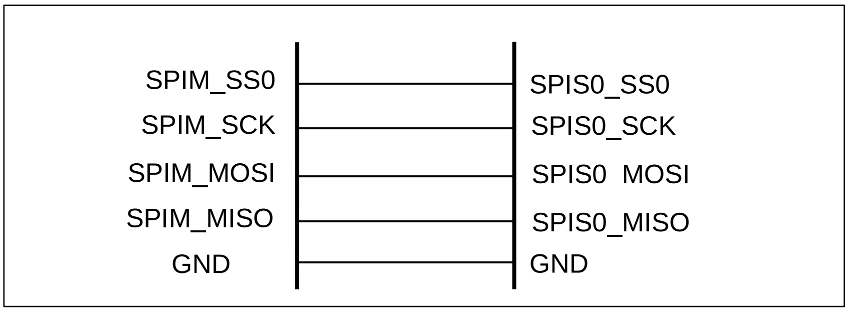


Figure 27 - Circuit Diagram for SPI Master Examples

Additionally, connect a USB to Serial converter cable 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.

3.18.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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
```

3.18.2 SPI Master Example 2

3.18.2.1 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.

3.18.2.2 Setup

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
SPIM_SS3	CN3 Pin 17	CN3 Pin 17	CN1 Pin 17	-	-
SPIM_SS0	-	-	-	CN1 Pin 5	J1 Pin 2
SPIM_SCK	J2 Pin 1	J2 Pin 1	J1 Pin 1	CN1 Pin 9	J1 Pin 1
SPIM_MOSI	J2 Pin 4	J2 Pin 4	J1 Pin 4	CN1 Pin 11	J1 Pin 4
SPIM_MISO	J2 Pin 3	J2 Pin 3	J1 Pin 3	CN1 Pin 12	J1 Pin 3

This example uses an AT93C46D 1Kbit EEPROM as a SPI device as shown in Figure 28

1. Connect the *SPIM_SS3* or *SPIM_SS0* to the *CS* pin of the AT93C46D.
2. Connect the *SPIM_SCK* to the SK pin of the AT93C46D.

3. Connect the *SPIM_MOSI* to the DI pin of the AT93C46D.
4. Connect the *SPIM_MISO* 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*.

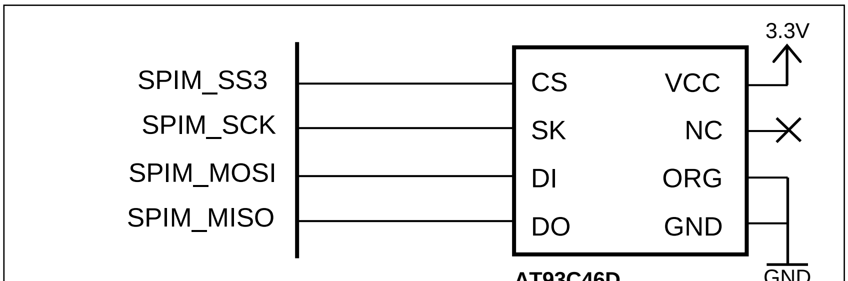


Figure 28 - Circuit Diagram for SPI master EEPROM Examples

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

3.18.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 FF  | ................
0x0010: FF 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 FF  | ................
0x0030: FF 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 FF  | ................
0x0050: FF 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 FF  | ................
0x0070: FF 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_h 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 FF  | ................
0x0010: FF 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 FF  | ................
0x0030: FF 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 FF  | ................
0x0050: FF 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 FF  | ................
0x0070: FF 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_h 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.
```

3.18.3 SPI Master Example 3

3.18.3.1 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.

3.18.3.2 Setup

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
SPIM_SS3	CN3 Pin 17	CN3 Pin 17	CN1 Pin 17	-	-
SPIM_SS0	-	-	-	CN1 Pin 5	J1 Pin 2
SPIM_SCK	J2 Pin 1	J2 Pin 1	J1 Pin 1	CN1 Pin 9	J1 Pin 1
SPIM_MOSI	J2 Pin 4	J2 Pin 4	J1 Pin 4	CN1 Pin 11	J1 Pin 4
SPIM_MISO	J2 Pin 3	J2 Pin 3	J1 Pin 3	CN1 Pin 12	J1 Pin 3
SPIM_IO2	J2 Pin 6	J2 Pin 6	J1 Pin 6	CN1 Pin 14	J1 Pin 6
SPIM_IO3	J2 Pin 5	J2 Pin 5	J1 Pin 5	CN1 Pin 13	J1 Pin 5

This example uses a HD47780 compatible LCD in 4 bit mode shown in Figure 29.

1. Connect the *SPIM_SS3* or *SPIM_SS0* to the *RS* pin of the LCD.
2. Connect *SPIM_SCK* to the *DB4* pin of the LCD.
3. Connect *SPIM_MISO* to the *DB5* pin of the LCD.
4. Connect *SPIM_IO2* to the *DB6* pin of the LCD.
5. Connect *SPIM_IO3* to the *DB7* pin of the LCD.
6. Connect the *VSS*, *R/W* and *LED–* pins of the LCD to *GND*.
7. Connect the *VCC* pin of the LCD to *5V*.
8. Connect a 100kΩ potentiometer between *5V* and *GND*, with the wiper going to pin *V0* of the LCD.

9. 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).

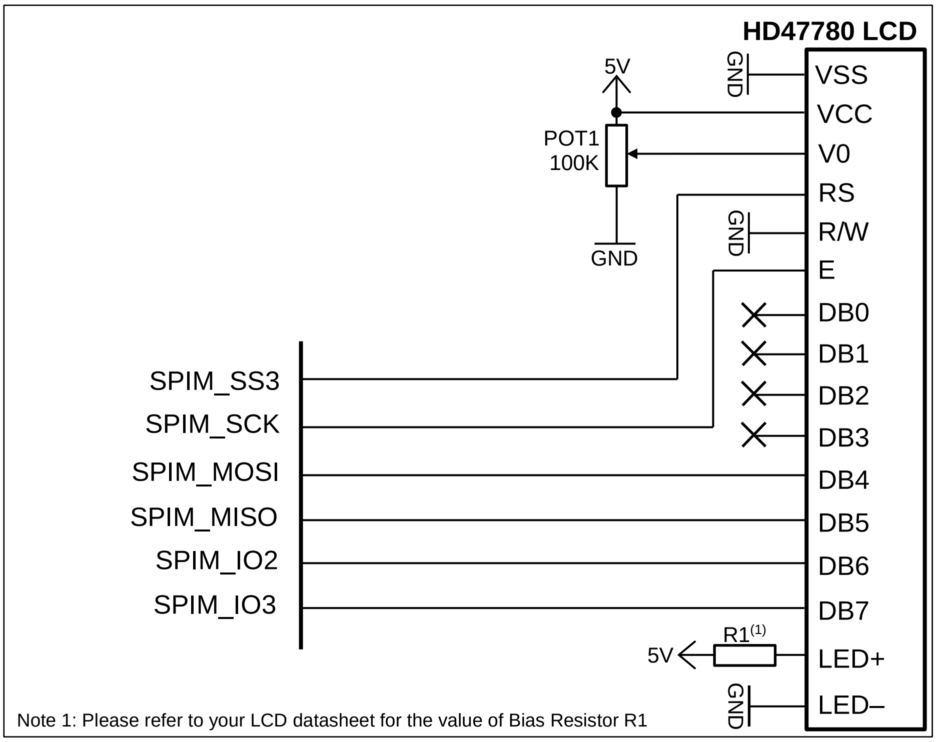


Figure 29 - Circuit Diagram for SPI Master Example 3

3.18.3.3 Execution

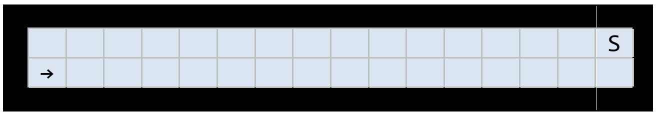
1. A welcome message should appear like so:

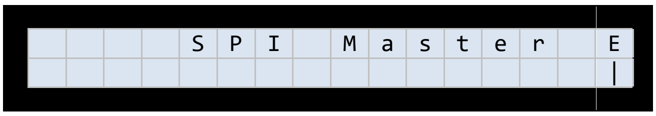
```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to SPI Master Example 3...

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

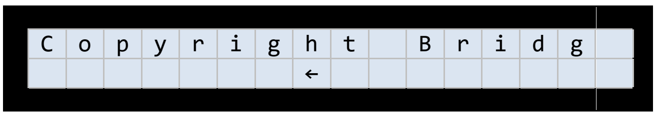
2. The program will initialize the LCD to 4-bit mode, displaying the message:

```
Initialising the LCD
```

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







3.19 SPI Slave Examples

3.19.1 SPI Slave Example 1

3.19.1.1 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.

3.19.1.2 Setup

Pin information:

Purpose	MM900EVxA	MM900 EVxB	MM900 EV-Lite	MM930 Mini	MM930 Lite
SPIM_SS0	J2 Pin 2	J2 Pin 2	J1 Pin 2	CN1 Pin 5	J1 Pin 2
SPIS0_SS	CN7 Pin 2	CN7 Pin 2	CN5 Pin 2	CN1 Pin 14	CN1 Pin 22
SPIM_SCK	J2 Pin 1	J2 Pin 1	J1 Pin 1	CN1 Pin 9	J1 Pin 1
SPIS0_SCK	CN7 Pin 1	CN7 Pin 1	CN5 Pin 1	CN1 Pin 9	CN1 Pin 18
SPIM_MOSI	J2 Pin 4	J2 Pin 4	J1 Pin 4	CN1 Pin 11	J1 Pin 4
SPIS0_MOSI	CN7 Pin 3	CN7 Pin 3	CN5 Pin 3	CN1 Pin 11	CN1 Pin 20
SPIM_MISO	J2 Pin 3	J2 Pin 3	J1 Pin 3	CN1 Pin 12	J1 Pin 3
SPIS0_MISO	CN7 Pin 4	CN7 Pin 4	CN5 Pin 7	CN1 Pin 12	CN1 Pin 19

This example uses two FT900EVx EV modules, one module as SPI master and other module as SPI slave.

Connect the following as shown in Figure 30.

1. Connect the *SPIM_SS0* to *SPIS0_SS.*
2. Connect the *SPIM_SCK* to *SPIS0_SCK*.
3. Connect the *SPIM_MOSI* to *SPIS0_MOSI*.
4. Connect the *SPIM_MISO* to *SPIS0_MISO*.
5. Ensure that both the boards have a common ground.

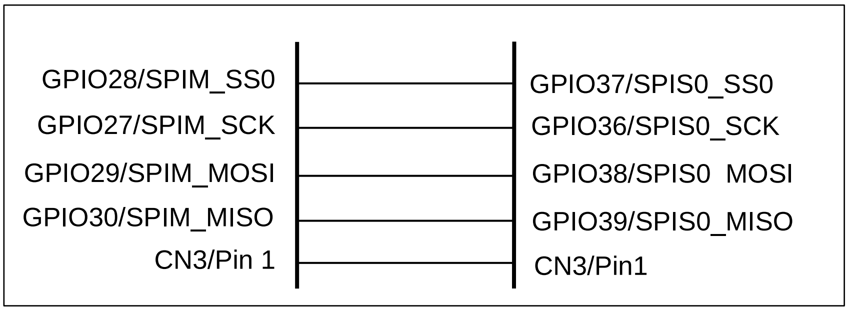


Figure 30 - Circuit Diagram for SPI Slave Examples

Additionally, connect a USB to Serial converter cable 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.

3.19.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to SPI Master Example 1...

Loopback use case between two FT900/FT930 (SPI Master) and FT900/FT930 (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.

3.20 Timer Examples

3.20.1 Timer Example 1

3.20.1.1 Purpose

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

3.20.1.2 Setup

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

3.20.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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 _ _ _ _
```

3.20.2 Timer Example 2

3.20.2.1 Purpose

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

3.20.2.2 Setup

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

3.20.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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
--------------------------------------------------------------------
```

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
```

3.20.3 Timer Example 3

3.20.3.1 Purpose

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

3.20.3.2 Setup

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

3.20.3.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to Timer Example 3...

All timers are in continuous mode with interrupts enabled.
 * 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
```

3.21 UART Examples

3.21.1 UART Example 1

3.21.1.1 Purpose

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

3.21.1.2 Setup

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

3.21.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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.

3.21.2 UART Example 2

3.21.2.1 Purpose

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

3.21.2.2 Setup

Refer to Setup Section of UART Example 1.

3.21.2.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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.

3.21.3 UART Example 3

3.21.3.1 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.

3.21.3.2 Setup

Refer to Setup Section of UART Example 1.

3.21.3.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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.

3.21.4 UART Example 4

3.21.4.1 Purpose

The purpose of this example is to demonstrate use of different UART mode - UART MODE-550, UART MODE-650, UART MODE-750, UART MODE-950 with auto flow control. Each UART mode configures Receiver & Transmitter to different FIFO trigger level.

3.21.4.2 Setup

Connect a USB to Serial converter (Gnd, Rx, Tx, CTS, RTS) to UART0 as this port is used to send debug text and for the example.

Modify Uart_example_4.c:

- Enable only desired UART mode: Default enabled UART mode is 950.

```
//#define UART_MODE_550
//#define UART_MODE_650
//#define UART_MODE_750
#define UART_MODE_950
```

- Change Baud rate to the required: UART_DIVIDER_1200_BAUD/ UART_DIVIDER_9600_BAUD/ UART_DIVIDER_19200_BAUD/ UART_DIVIDER_115200_BAUD.

Configure the host PC terminal application (for e.g Teraterm) for:

- Baud rate same as in the UART Example 4 : 1200/ 9600/ 19200/ 115200,
- Flow Control : hardware
- Set the Log file path (Tera Term - > File – Log…).

Build UART Example 4 and download the application to the MM9xx hardware board.

3.21.4.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to UART Example 4...

Any character typed here will be echoed back on the same serial port
via an interrupt. This example uses UART FIFO mode with auto flow control.
--------------------------------------------------------------------
```

2. Typing any characters here will cause them to be transmitted back.
3. Further, prepare a text file with some text message. And send the file (Tera Term - > File – Send file…).
4. Compare to check the log file contents are same as the sent file.

3.21.5 UART 9Bit Mode Example

3.21.5.1 Purpose

The purpose of this example is to demonstrate using the UART in 9-bit mode.

3.21.5.2 Setup

Supported hardware:
MM900EVxA and MM900EVxB for FT900

Connect a USB to Serial converter cable to UART0 as this port is used to send debug prints. The terminal settings is 19200 baud rate,8 data bits, no parity and 1 stop bit. Flow control is not enabled.

Loopback transmitter pin and receiver pin of UART1 (external loopback). Flow control is not enabled for UART1.

3.21.5.3 Execution

A welcome message should appear, followed by three paragraph texts. Each paragraph is sent for a configured address in the UART1.

```
Copyright (C) Bridgetek Pte Ltd
--------------------------------------------------------------------------
Welcome to UART 9-bit mode Example...

 UART1 works in 9-bit mode. When Rx and Tx pin of UART1 are externally
 connected to make a loop. UART1 receives the data from its transmitter
 and the received data is displayed in the debug UART.
-------------------------------------------------------------------------

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam dictum nunc id
ullamcorper consectetur.Vestibulum tristique egestas ligula a rutrum. In eu blandit
elit, eu ultricies risus. Vivamus vitae duiut purus vestibulum blandit. Orci varius
natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.
Pellentesque quis aliquam metus. Suspendisse sit amet bibendum felis, at iaculis
elit. Suspendisse quis enim mauris. Morbi a accumsan dolor. Aliquam dignissim,
lectus quis aliquet consequat,velit tortor volutpat tortor, a vehicula risus sem
non ipsum. Aenean et iaculis magna, quis placerat elit.

Vivamus laoreet tempor lorem sit amet lobortis. Proin in mauris rutrum, sagittis
erat ac, sodales enim.In scelerisque scelerisque enim id cursus. Morbi aliquam leo
eget ornare semper. Aliquam vitae diam ut dolor interdum tincidunt non eget velit.
Ut eros enim, pellentesque vitae est a, feugiat congue massa. Nulla facilisi.

Suspendisse interdum ligula in convallis posuere. Pellentesque quis nisl turpis. In
congue, enim at ultricies luctus, ex purus suscipit risus, quis tristique elit
massa et nisl. Cras vehicula neque nec lacus tincidunt malesuada. Mauris aliquam,
lectus a dictum hendrerit, nunc dui interdum metus, a finibus massa ipsum in
sapien. Nam sit amet sem faucibus est eleifend vehicula. Nulla sapien justo,
aliquam sed ullamcorper ut, facilisis non leo. Suspendisse elementum augue nunc,
sit amet vulputate turpis consequat sit amet. Aenean quis lacinia sem.
```

3.22 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/.

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 3.

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 [FT900] 0xfcf [FT930]

Table 3 - 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.

3.22.1 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.

3.22.1.1 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.

3.22.1.2 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.

In case of FT930, the pins GPIO 19, GPIO 20 and GPIO 21 are used for the above purpose.

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.

3.22.1.3 Execution

1. A welcome message should appear like so:

```
Copyright (C) Bridgetek Pte 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
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.

3.22.2 USBD Example BOMS to SD Card

3.22.2.1 Purpose

This example program will create FT9xx 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 FT9XX device.

3.22.2.2 Setup

Connect a USB to Serial converter cable to UART0 as this port is used to send debug text. Insert a FAT32 formatted SD card (CN5 on the MM900EVxA EVM or CN7 on MM900EV-Lite or CN7 on the MM930Lite). Connect the FT900 USB device port to a USB host. The device is enumerated as a USB mass storage device.

3.22.2.3 Execution

1. A welcome message should appear like so:

```
Copyright I Bridgetek Pte Ltd
---------------------------------------------------------------------
Welcome to USB Mass Storage to SD Card Test ApplicatiI...
---------------------------------------------------------------------
Inge...connect car79ibendum7979nitialise 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.

3.22.3 USBD 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.

3.22.3.1 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.

3.22.3.2 Setup

Connect the FT9xx 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.

3.22.3.3 Execution

1. A welcome message should appear like so,

```
CIright (C) Bridge-----------
Welcome to USBD 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 a FT9xx device... nice to meet you!
```

3. The device will not perform any further actions.

3.22.4 USBD 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.

3.22.4.1 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.

3.22.4.2 Setup

Connect the FT9xx USB device port to a USB host and a USB to UART convertor to the FT900 UART.

3.22.4.3 Execution

1. A welcome message should appear like so,

```
Iopyright (C) Bri--------------
Welcome to USBD 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. For example, typing “Hello World” on the UART terminal, will have the following keypresses appear on the PC

```
Hello World
```

3.22.5 USBD 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.

3.22.5.1 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 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.

3.22.5.2 Setup

Connect the FT9xx 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 "*FT9xx 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 FT9XX 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.

3.22.5.3 Execution

1. A welcome message should appear like I

```
Copyright (C) -----------------
Welcome to USBD 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.

3.22.6 USBD 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.

3.22.6.1 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 strea83ibendum 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.

3.22.6.2 Setup

Connect an Ethernet cable to the FT90X's Ethernet Port. Connect a USB to Serial converter cable 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.

3.22.6.3 Execution

1. A welcome message should appear liIso:

```
Copyright (C) Bridgetek Pte Ltd
---------------------------------------------------------------------
Welcome to USBD RIS 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
PIet filter ON P N M Y
..........Rx 0 I 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 1Irr 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.

3.22.7 USBD Example UVC Webcam

This example is adapted from the FT90X UVC Webcam example (AN_414) to use highbandwidth ISO interface. Data is transferred to the USB host using either bulk transfers or isochronous transfers. Using high bandwidth isochronous transfer, resolution of up to SVGA is supported in this example.

3.22.7.1 Purpose

The FT90X devices include a Camera Interface and this can be used with suitable USB code to implement a UVC webcam. This example demonstrates how data from the camera interface can be read and sent to the USB interface. A USB host can receive the video data and display the output as it would with a dedicated webcam.

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.

There is an option to use bulk or isochronous data endpoints. A macro "USB_ENDPOINT_USE_ISOC" should be defined to enable isochronous otherwise bulk will be used.

The data rate available from bulk data endpoints is higher and will ther–fore support higher resolutions - QVGA and VGA resolutions (320x240, 640x480 respectively) uncompressed YUY2 stream. When using isochronous transfers only QVGA is supported. But if highbandwidth isochronous transfer (macro USB_ENDPOINT_USE_HBW_ISOC) is selected, resolutions upto uncompressed SVGA is supported in this example. When isochronous data endpoints are enabled the SetInterface request is used to enable data transmission over the USB. Class-specific configuration settings are defined for use by the GET_CUR, GET_MIN, GET_MAX, GET_DEF, GET_RES and SET_CUR requests.

The DFU-C facility may be enabled for this application. It can be enabled at startup or can provide a separate interface for updating. To enable the DFU at startup define the USB_INTERFACE_USE_STARTUPDFU macro. This enables a call from the macro STARTUP_DFU() in the main() function. It will briefly enable the USB device on the FT90X and allow a DFU utility to update the application code. This can be removed entirely or configured to alter the number of milliseconds it will wait before closing the USB device and continuing with the application. To have a DFU interface during normal operation define the USB_INTERFACE_USE_DFU macro.

3.22.7.2 Setup

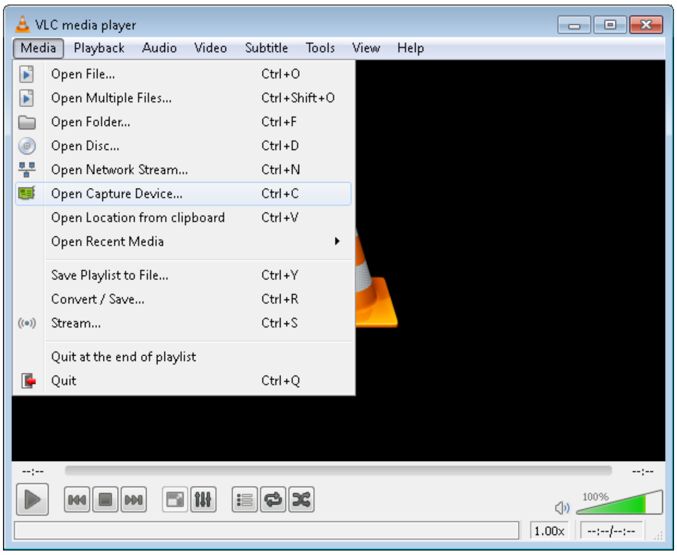
Supported hardware:

- MM900EV2A/MM900EV3A module with a front or back facing OV9657 camera module. The example has to be configured for Bulk transfer.

The module is connected to and powered by a PC running Windows. When the device enumerates as a USB imaging device (AN 414 UVC ISOC or AN 414 UVC BULK), it can be tested using VLC media player or Skype application on the windows PC.

3.22.7.3 Execution

The VLC media player uses Microsoft DirectShow to obtain video from the webcam. To connect to the FT90X webcam select "Open Capture Device…" from the Media menu. The menu item is shown in Figure 32.



Fi–ure 31 - VLC Media Menu

The Open Capture Device dialog has a drop-down box to select the device and an option for "Video Size" as shown in Figure 33. The device name of "FT900 UVC" is used by this application note.

In hi-speed mode any of the three supported resolutions can be used for the video size. If it is left blank then the highest resolution is used.

Valid options are "320x240" for QVGA, "640x480" for VGA and "800x600" for SVGA.

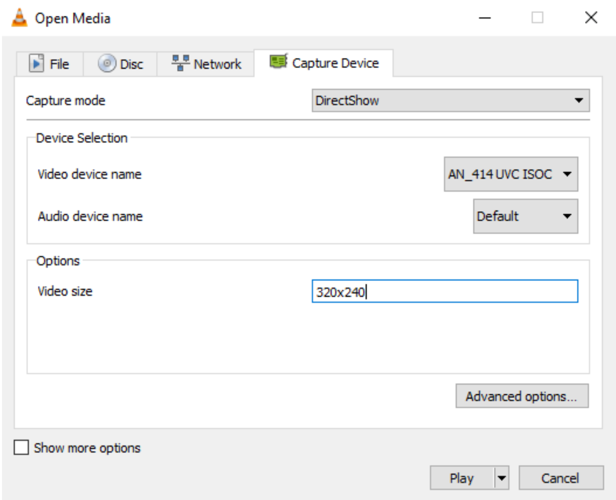


Figure 32- VLC Open Capture Device Dialog

When the device is selected in Skype it will try to open at VGA resolution (640x480).

To test this select Tools --> Options --> Video settings.

3.23 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/.

3.23.1 USBH_Example Hub

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

3.23.1.1 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.

3.23.1.2 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.

3.23.1.3 Execution

```
A welcome message shId appear like so---------------------------------
Icome to USBH Hub Tester Example 1...

List --------------------------------------
```

1. 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.

3.23.2 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.

3.23.2.1 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.

3.23.2.2 Setup

Connect the FT90X 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.

3.23.2.3 Execution

```
A welcome messI should appear l----------------------------------------
Welcome to USBH HID Tester Example 1...

Find and displays reports received from HI‘ device’ connected to
the USB Host Port

'Timeout' is displayed when the connected HID device does not
respond to the host requests; it does not indicate an error! Try to move
the HID mouse or press any key on the HID keyboard to generate HID reports
--------------------------------------------------------------------
```

1. 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.

3.23.3 USBH Example CDCACM

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

3.23.3.1 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).

3.23.3.2 Setup

Connect the FT90X 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.

3.23.3.3 Execution

```
A welcome mIage should appear------------------------------------------
Welcome to USBH CDC Tester Example 1...

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

1. 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
BeginniICDC ACM testing
Sending encapsulated reset... I 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.

3.23.4 USBH Example BOMS

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

3.23.4.1 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.

3.23.4.2 Setup

Connect the FT90X 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.

3.23.4.3 Execution

```
A welcome message should appear------------------------------------------------
---
Welcome to USBH BOMS Tester Example 1...

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

1. 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 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 00
```

3.23.5 USBH Example File System

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

3.23.5.1 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.

3.23.5.2 Setup

Connect the FT90X 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.

3.23.5.3 Execution

```
A welcome message should appear ------------------------------------------------
-------
Welcome to USBH File System Tester Example 1...

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

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

```
USB Device Detected
USB D“”ice 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 SCI.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
```

2. 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
```

3. 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. Ut92ibendum92t aliquam lectus. Phasellus non risus a nisl semper
vehicula a vitae lore92ibendue 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 sapi92ibenduporta. 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 er92ibendumr. Donec lorem justo, vestibulum vitae sagittis ac, bibendum
vitae velit. Integer ante mi, tempus sodales consect92ibendum, porta ac libero.
Maecenas dapibus orci at rhoncus bibendum. Nulla elementum lectus massa, non
varius lorem scelerisque sit amet.

Closing LOREM.TXT
```

3.23.6 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.

3.23.6.1 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.

3.23.6.2 Setup

1. Run the FT90X device with the USBH Example FT232.bin. Connect a USB to Serial converter cable 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 paritI8 data bits, and 1 stop bit, **no flow control***"

```
Copyright (C) Bridgetek Pte 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**"*.

3.23.6.3 Execution

1. Connect the other end of the crossover cable to the FT90X'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:I3; 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.

3.23.7 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 FT90X). 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.

3.23.7.1 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_208 FT311/FT312D Demo APK User Guide explains the details of the Android
- 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.

3.23.7.2 GPIO Example

3.23.7.2.1 Purpose

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

3.23.7.2.2 Setup

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

Connect a USB to Serial converter cable 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.

3.23.7.2.3 Execution

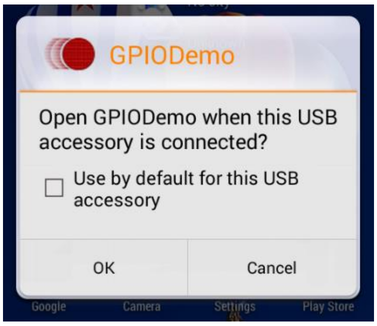
1. A welcome messageIould appear on the PC terminal application like so,

```
Copyright (C) Bridgetek Pte Ltd
-------------------------------------------------------------------
---
Welcome to USBH GPIO AOA Tester Example 1...

Tests the FTDIGPIODemo on an Android device.
-------------------------------------------------------------------
---
```

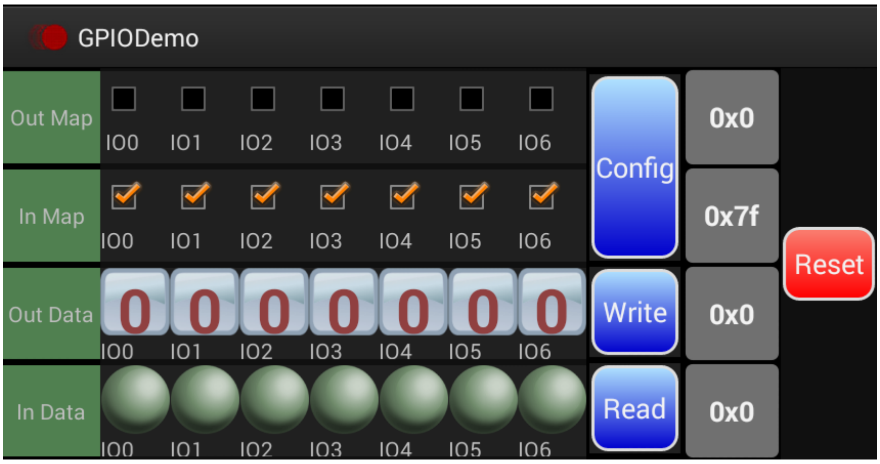
2. Connect the Android device to the MM900EV board using a USB cable. The PC terminal application should display

```
USB DevicIetected
.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.



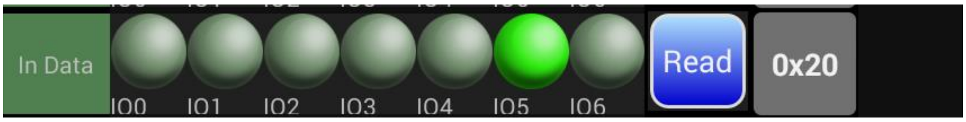
4. Click the "Config" button on the App to send some data to FT900. The following message will appear on the PC terminal.

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

5. 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.



3.23.7.3 UART Loopback Example

3.23.7.3.1 Purpose

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

3.23.7.3.2 Setup

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

Connect a USB to Serial converter cable 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.

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

3.23.7.3.3 Execution

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

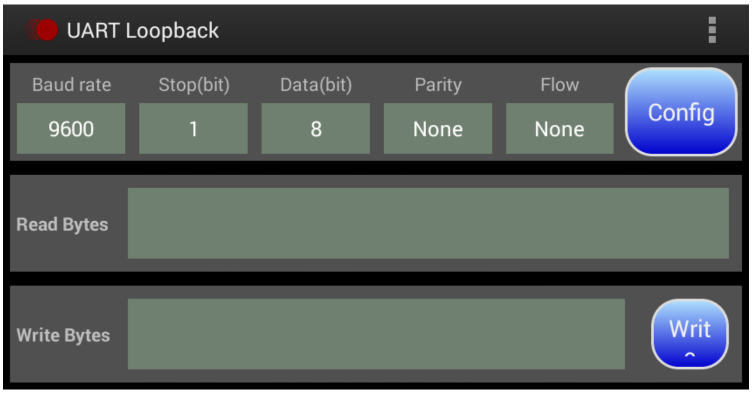
```
Copyright 2016, Bridgetek Pte Ltd
--------------------------------------------------------------------
Welcome to USBH UART AOA Tester Example 1...

Tests the FTDIUARTLoopback on an Android device.
--------------------------------------------------------------------
```

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

```
USB DeIe 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

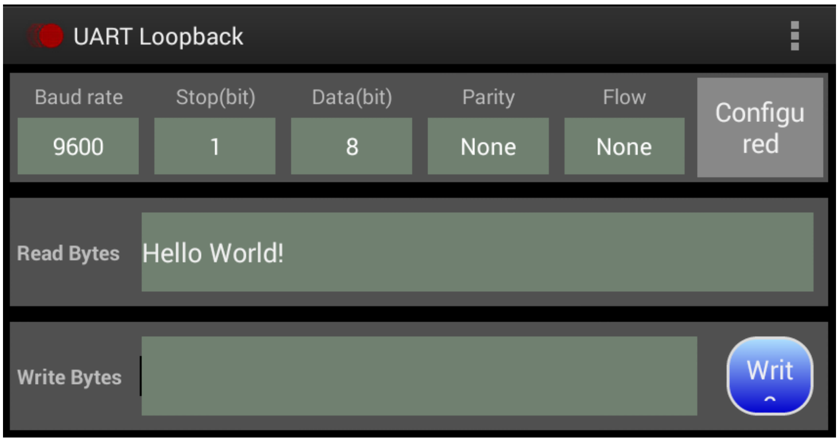
3. Select OK on the Android device. The UARTLoopback application should be automatically launched and you should be able to see the following screen.



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
```

5. 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.



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

3.23.7.4 AOA HID Example

3.23.7.4.1 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.

3.23.7.4.2 Setup

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

Connect a USB to Serial converter cable 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.

3.23.7.4.3 Execution

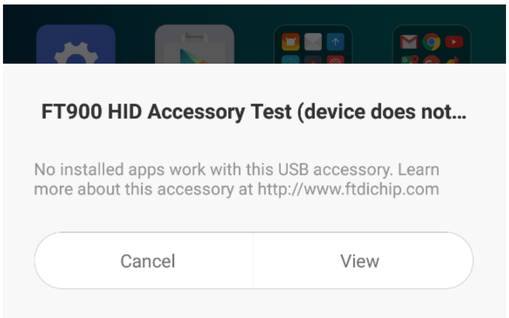
1. A welcome message should appear on the PC terminal application like so,

```
Copyright 2016, Bridgetek Pte 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

```
USBIvice 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
```

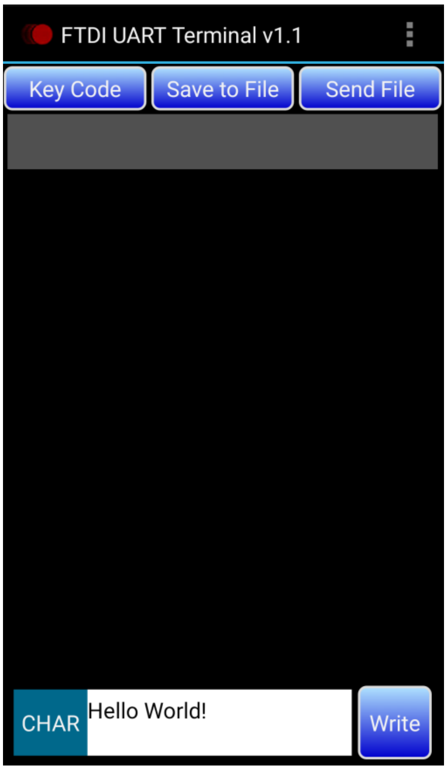
3. On the Android device the following pop-up will appear. Select Cancel to proceed.



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.

   At the PC terminal type a string that you wish to send to the Android device
5. , 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 applications 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.

3.24 Watchdog Timer Examples

3.24.1 Watchdog Example 1

3.24.1.1 Purpose

The purpose of this example is to demonstrate the usage of the Watchdog Timer in FT9XX. On a watchdog timeout, the current context of the application is saved in the first page of dlog location in flash. The current context of the application for this example is a dummy array of 256 bytes and its contents is incremented on every watchdog timeout. The program in addition displays the type of power on reset (watchdog, soft, external hardware) the hardware has undergone before it began execution.

3.24.1.2 Setup

Connect a USB to Serial converter cable to UART0 as this port is used to seIdebug text.

```
ExeLtd
----------------------------------------------------------------------------
Welcome to WDT Example 1.ut.
----------------------------------------------------------------------------
!!!EXTERNAL HW Power-On-Reset!!!
```

1. Similarly it could display "!!!WATCHDOG Power-On-Reset!!!" or "!!!SOFT Power-On-Reset!!!" based on the type of reset flag read from the hardware register. On a watchdog power on reset, the program then displays the last saved application context from dlog page in flash. The array values range from 0 – 255 and is incremented every watchdog second level timeout.

```
last saved context on a watchdog reset:

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
```

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

```
Wait for 10 Seconds before starting the test
WDT Kick 1
WDT Kick 2
WDT Kick 3
WDT Kick 4
WDT Kick 5IT Kick 6
I Kick 7
WDT Kickk Pte Ltd
----------------------------------------------------------------------------
Welcome to WDT Exa timeout.
----------------------------------------------------------------------------
!!!WATCHDOG Power-On-Reset!!!
last saved context on a watchdog reset:

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
Wait for 10 Seconds before starting the test
```

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

4 Contact Information

Head Quarters – Singapore

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

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

Branch Office – Taipei, Taiwan

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

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

Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 27[77]
Fax: +44 (0) 141 429 2758

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

Branch Office – Vietnam

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

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

Web Site

http://brtchip.com/

Distributor and Sales Representatives

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

Appendix A – References

Document References

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

FT900/FT901/FT902/FT903 Datasheet

FT905/FT906/FT907/FT908 Datasheet

FT900 User Manual

AN_325 FT9XX Toolchain Installation Guide

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

Bus Pirate: http://dangerousprototypes.com/docs/Bus_Pirate

GNU Make Manual – 9.5 Overriding Variables

AN_414 FT90x UVC Webcam

Acronyms and Abbreviations

Terms	Description
ADC	Analogue to Digital Converter
ARP	Address Resolution Protocol
CAN	Controller Area Network
DAC	Digital to Analogue Converter
EEPROM	Electronically Erasable PROgrammable Memory
EVM	Evaluation Module
GPIO	General Purpose I/O
I^{2}C	Inter-IC
I^{2}S	Inter-IC Sound
ICMP	Internet Control Messaging Protocol
MDI-X	Medium Dependent Interface Crossover
PWM	Pulse Width Modulation
RTC	Real Time Clock
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver Transmitter
USB	Universal Serial Bus
UVC	USB Video Class
WDT	WatchDog Timer

Appendix B – List of Tables & Figures

List of Tables

List of Figures

Appendix C – Revision History

Document Title: AN_360 FT9XX Example Applications

Document Reference No.: BRT_000115

Clearance No.: BRT#073

Product Page: http://brtchip.com/product/

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, USBD 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
1.4	Updated release; Migration of the product from FTDI to Bridgetek name – logo changed, copyright changed, contact information changed	2017-03-08
1.5	Updated for toolchain v2.3.1	2017-03-22
1.6	Updated the content for support of MM930Lite module; Updated output for Free RTOS examples output texts in alignment with the example code; Added Se–tion 2.7 on D2XX drivers used in D2XX Examples	2017-07-05
1.7	Updated the following - ADC examples, RTC external examples, USBD Example HID, USBD Example UVC Webcam, Section 2.6 Programming, support for MM900EV-Lite module, table for boards and the example supported in those boards, Watchdog example 1 and I2S Master examples; Added note in section 3.15 RTC Internal examples on the compatibility in FT90X Rev. B and C	2018-01-19
1.8	Added section "3.9.5 FreeRTOS Example 4". Changed the baud rate of debug UART0 for all the examples to 19200 Added documentation for UART Example 4 (fifo usage in UART), UART 9Bit Mode Example Updated ADC_Example 3 for switching resolution and frequency	2018-11-13