FreeRTOS BSP i.MX 7Dual Demo Applications User's Guide

Freescale Semiconductor, Inc.

Document Number: FRTOS7DDAUG
Rev. 0
August 2015

Contents

Section number Title Page

Chapter 1
Introduction

FreeRTOS BSP 1.0.0 i.MX 7Dual includes applications which provide examples that show how to use this BSP for i.MX 7Dual Processor. This document describes the applications and provides instructions to configure each application (if available). The document also describes the required board setup and steps to run the applications.

Chapter 2
LED Blinking Demo on i.MX device

This demo demonstrates how to toggle a blinking LED with different frequencies.

2.1 Overview

This demo uses GPIO and GPT drivers, as well as an RDC SEMAPHORE driver to implement a blinking LED with different frequencies. If there is no user LED BOARD_GPIO_LED_CONFIG defined in board.-h, the demo outputs "+" and "-" on the terminal, which represent the LED on and off status. In this demo, a safe shared peripheral access way is introduced with RDC SEMAPHORE.

This demo has two tasks: one for switching the blinking frequency of the LED, and the other for toggling the GPIO to power on or off the LED (or output "+" and "-" instead). The switching task waits for the user to push the button on the board, and changes the blinking frequency accordingly. The toggling task waits for a time determined by the blinking frequency, then toggles the LED status.

If there is no user button BOARD_GPIO_KEY_CONFIG defined in board.h, the demo switches frequency every 5 seconds.

2.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

2.3 System Requirement

2.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

2.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench®
- ARM® GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

2.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/blinking_imx_demo/<toolchain>.

2.4 Getting Started

2.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to the target board using U-Boot.
4. Boot auxiliary ARM® Cortex® -M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

2.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information if both user LED and button are available on board:

```
================= Blinking Demo ==================

====== Blinking interval 100ms ======

Press the (FUNC1) key to switch the blinking frequency:
```

After the user pushes the "FUNC1" (the string here varies among boards) button, something new appears on the terminal, and the blinking runs much slower:

```
====== Blinking interval 200ms ======

Press the (FUNC1) key to switch the blinking frequency:
```

When this operation can be repeated, the LED blinks slower and slower until the blinking interval increases to 1000 ms. The user can push the button again to recover the blinking interval to 100 ms.

Chapter 3
ECSPI Flash Demo

This demo application demonstrates how to use the ECSPI drivers to access SPI flash memory on FreeRTOS OS.

3.1 Overview

This demo application demonstrates the ECSPI drivers working on FreeRTOS OS and how to use the ECSPI drivers to access SPI flash memory. The demo provides following features:

- Read memory status
- Set memory writing protection
- Erase memory
- Read data from memory
- Write data to memory

Note:

1. Write data to memory (Page Program) instruction allows up to 256 bytes to be written at a time. The written address should be consecutive, and on the same page of memory. If more than 256 bytes are sent to the device, previously latched data is discarded, and only the last 256 data bytes are guaranteed to be programmed correctly within the same page.
2. To ensure that other demos are not be affected, the user need to erase all chips to ensure the successful execution of future demos.

3.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual Validation board

3.3 System Requirement

3.3.1 Hardware requirements

- SD Card
- Mini USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

3.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

3.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/ecspi_flash_demo/<toolchain>.

3.4 Getting Started

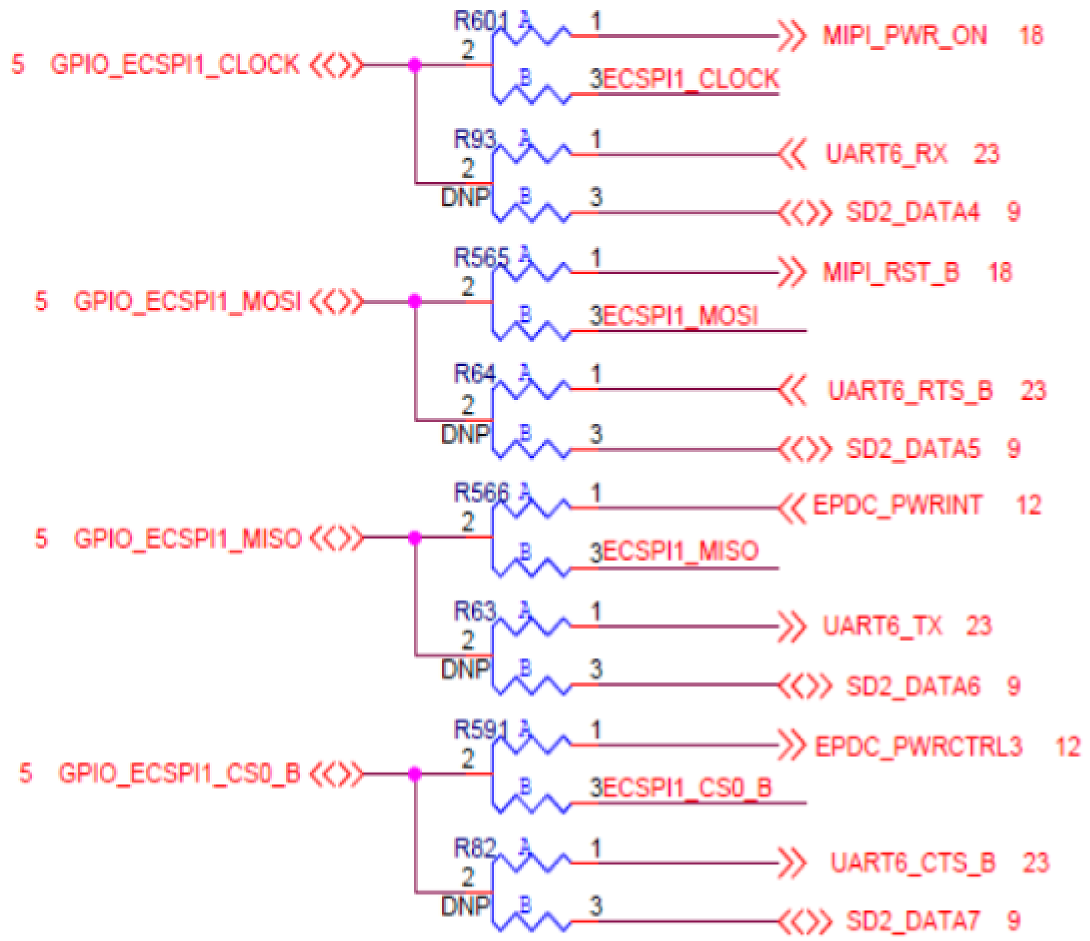
3.4.1 Hardware settings

To run this demo, enable ECSPI flash on the board. The following is the rework steps required for some boards.

i.MX 7Dual Validation board

Route ECSPI1 signals to "ECSPI1_CLOCK", "ECSPI1_MOSI", "ECSPI1_MISO", and "ECSPI1_CS0_-B".

Populate R601, R565, R566, and R591 to position B.



3.4.2 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

3.4.3 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
--------------------------- ECSPI Flash Demo ----------------------------

This demo application demonstrates usage of ECSPI driver based on FreeRTOS.
It transfers data to/from external memory over SPI bus.

Start Flash Memory Operating!
Read memory status ... 0x00
Enable write latch in memory ...OK
Read memory status ... 0x02
Write unprotect memory ... OK
Read memory status ... 0x00
Enable write latch in memory ...OK
Page write 32 bytes to location 0x00000120 in memory:
Read memory status ... 0x00
0x00    0x01    0x02    0x03    0x04    0x05    0x06    0x07    0x08    0x09   0x0a     0x0b    0x0c
      0x0d    0x0e    0x0f
0x0f    0x0e    0x0d    0x0c   0x0b     0x0a    0x09    0x08    0x07    0x06    0x05    0x04    0x03
      0x02   0x01     0x00
Reading 32 bytes from location 0x00000120 in memory:
0x00    0x01    0x02    0x03    0x04    0x05    0x06    0x07    0x08    0x09   0x0a     0x0b    0x0c
      0x0d    0x0e    0x0f
0x0f    0x0e    0x0d    0x0c   0x0b     0x0a    0x09    0x08    0x07    0x06    0x05    0x04    0x03
      0x02   0x01     0x00
Finish Flash Memory Operating!
```

Chapter 4
Hello World Demo

This demo application demonstrates the Hello World demo.

4.1 Overview

The Hello World project is a simple demonstration program that uses the BSP software. It prints the "Hello World" message to the ARM Cortex-M4 terminal using the BSP UART drivers. The purpose of this demo is to show how to use the UART and to provide a simple project for debugging and further development.

The Hello World QSPI project is same as Hello World, except they use different code regions. The project mainly shows that it can use external flash, such as QSPI, to run the program.

4.2 Supported Platforms

These Freescale i.MX boards are supported by this demo.

- i.MX 7Dual SABRE board (SDB)
- i.MX 7Dual Validation board

4.3 System Requirement

4.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

4.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

4.3.3 Software requirements

- The hello world project files are in: <BSP_Install>/examples/<board>/demo_apps/hello_-world/<toolchain>.
- The hello world qspi project files are in: <BSP_Install>/examples/<board>/demo_apps/hello_-world_qspi/<toolchain>.
- To kick off the hello world qspi demo, U-Boot must be configured with QSPI flash enabled.

4.4 Getting Started

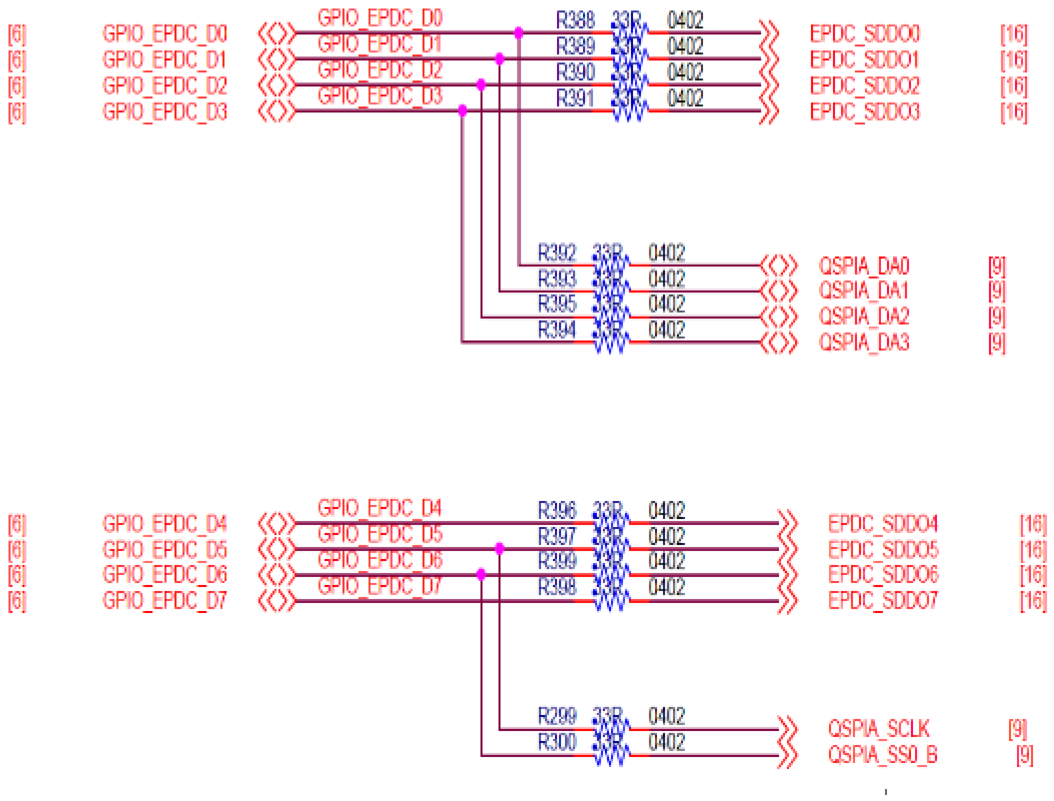
4.4.1 Hardware Settings

For Hello World QSPI demo, enable the QSPI flash on the board. The following is the rework steps required on some boards.

i.MX 7Dual SABRE board(SDB)

To enable FreeRTOS application running on the QSPI flash, remove some resistors.

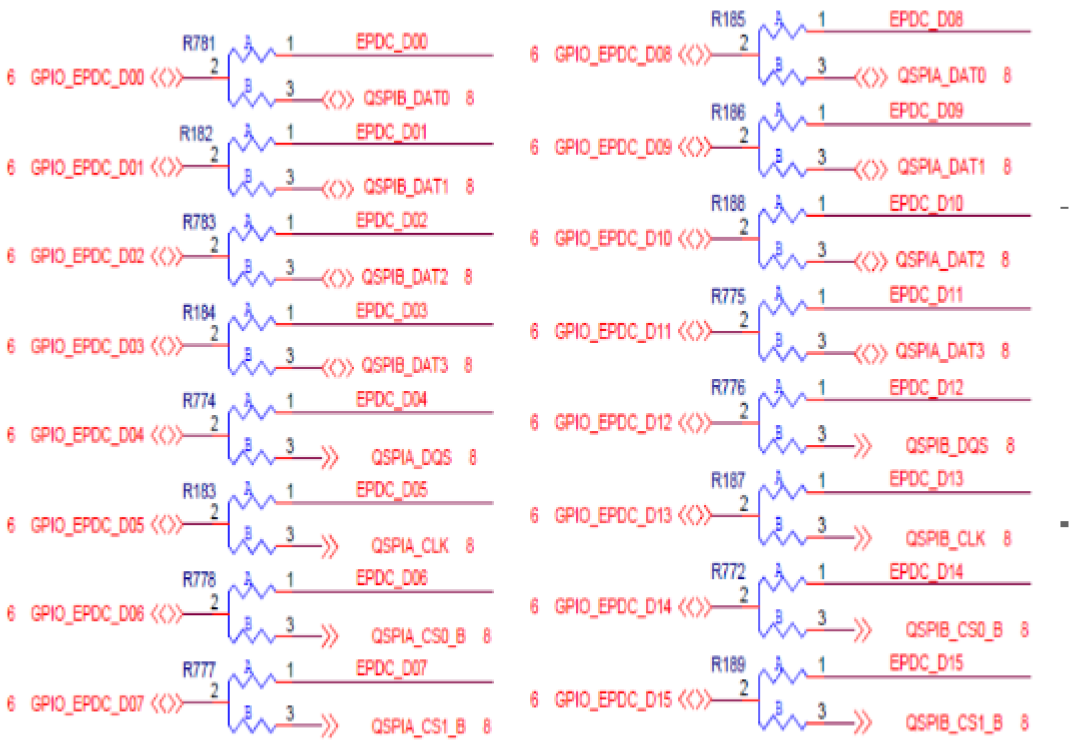
1. Remove R388, R389, R390, R391, R397, and R399.
2. Populate R392, R393, R394, R395, R299, and R300.



i.MX 7Dual Validation board

To enable FreeRTOS application running on the QSPI flash, route QSPIA and QSPIB signals from EPDC by resistors.

1. Populate R781, R182, R783, R184, R774, R183, R778, and R777 to position B.
2. Populate R185, R186, R188, R775, R776, R187, R772, and R189 to position B.



4.4.2 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

4.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
Hello World!
```

You can puts some inputs like this :

```
Hello World!
test
```

Chapter 5
RPMsg PingPong Demo

This demo application demonstrates how to use the RPMsg stack to communicate with Linux OS on FreeRTOS OS. FreeRTOS plays a role "remote" and Linux OS is the "master" in the RPMsg protocol.

5.1 Overview

This demo application demonstrates the RPMsg remote stack working on FreeRTOS OS. It works with Linux RPMsg master peer to transfer integer values back and forth. The name service handshake is performed first to create the communication channels. Next, Linux OS transfers the first integer to FreeRTOS OS. The receiving peer adds 1 to the integer and transfers it back. The loop continues to demonstrate the stability of RPMsg stack.

5.2 Supported Platforms

The following board is supported by this demo

- i.MX 7Dual SABRE board (SDB)

5.3 System Requirement

5.3.1 Hardware requirements

- SD Card
- Mini USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

5.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- ARM DS-5

For toolchain version, see *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007-DRN).

5.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/rpmsg/pingpong/<toolchain>.
- Linux RPMsg master side pingpong demo

5.4 Getting Started

5.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open two serial terminals with these settings for each of the virtual serial instances:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.
5. Boot the Linux kernel, and run the pingpong master side demo.

For detailed instructions to run this demo on Cortex-M4, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

5.4.2 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
RPMSG PingPong Demo...
RPMSG Init as Remote
init M4 as REMOTE
```

After the Linux pingpong master side demo is running, the ARM Cortex-M4 terminal displays the following information:

```
Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]
Get Data From A7 : 0
Get Data From A7 : 2
Get Data From A7 : 4
Get Data From A7 : 6
Get Data From A7 : 8
...
```

As is shown on the log, the RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform a name service handshake to create the communication channel. The M4 channel address is 1, and the A7 channel address is 2014. A7 begins to send the first data to M4. After receiving the data, M4 adds 1 to it and sends it back to A7. A7 responds with the same behaviour. The loop continues to demonstrate the stability of RPMsg communication.

Chapter 6
RPMsg String Echo Demo

This demo application demonstrates how to use the RPMsg stack to communicate with Linux OS on FreeRTOS OS. FreeRTOS OS plays the role "remote" and Linux OS is the "master" in the RPMsg protocol.

6.1 Overview

This demo application demonstrates the RPMsg remote stack working on FreeRTOS OS. It works with Linux RPMsg master peer to transfer string content back and forth. The name service handshake is performed first to create the communication channels. Next, Linux OS waits for user input to the RPMsg virtual tty. Anything which is received is sent to M4. M4 displays what is received, and echoes back the same message as an acknowledgement. After acknowledging it, A7 can wait for next user input to the RPMsg virtual tty. The demo demonstrates RPMsg's ability to send arbitrary content back and forth.

Note: The maximum message length supported by RPMsg is now 496 bytes.

6.2 Supported Platforms

The following board is supported by this demo:

- i.MX 7Dual SABRE board (SDB)

6.3 System Requirement

6.3.1 Hardware requirements

- SD Card
- Mini USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

6.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007DRN).

6.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/rpmsg/str_echo/<toolchain>.
- Linux RPMsg master side string echo demo

6.4 Getting Started

6.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open two serial terminals with these settings for each of the virtual serial instances:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Load the demo binary to the target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.
5. Boot the Linux kernel, run the string echo master side demo in it.

For detailed instructions to run this demo on ARM Cortex-M4, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

6.4.2 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
RPMSG String Echo Demo...
RPMSG Init as Remote
init M4 as REMOTE
```

After the Linux string echo master side demo is running, the user can input an arbitrary string to the virtual RPMsg tty using the following echo command:

```
echo test > /dev/ttyRPMSG
imx_rpmsg_tty rpmsg0: msg(<- src 0x1) test len 4
imx_rpmsg_tty rpmsg0: msg(<- src 0x1) len 2

echo deadbeaf > /dev/ttyRPMSG
imx_rpmsg_tty rpmsg0: msg(<- src 0x1) deadbeaf len 8
imx_rpmsg_tty rpmsg0: msg(<- src 0x1) len 2

...
```

On the M4 terminal, the received string content and its length is output, as shown in the log.

```
Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]
Get Message From A7 : "test" [len : 4] From Slot 0
Get New Line From A7 From Slot 1
Get Message From A7 : "deadbeaf" [len : 8] From Slot 2
Get New Line From A7 From Slot 0
...
```

The RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform name service handshake to create the communication channel. The M4 channel address is 1, and the A7 channel address is 1024. A7 then waits for user input to RPMsg virtual tty and sends the content to M4. On receiving the data, M4 outputs the content and its length on the terminal and echoes back the same message to A7. A7 outputs what is sent back as a verification, then waits for the next user input. The loop continues to demonstrate RPMsg's ability to send arbitrary content.

As seen in the log, there are 3 slots that are used in the remote side. The Master side may send a bundle of messages faster than the remote can consume them. To resolve this problem, synchronization is used to prevent Master from sending too many messages. Even when this is used, there is still a possibility Master can send up to 3 messages before Remote consumes them. Therefore, the remote application layer adds a size 3 buffer to hold these messages. Each entry of the buffer is called a slot. See the code for the detailed explanation.

Chapter 7
SEMA4 mutex Demo on i.MX device

This demo demonstrates how to implement a multicore mutex with SEMA4.

7.1 Overview

This demo use SEMA4 driver to implement a multicore mutex without spinning with CPU. The mutex is event driven, and one core can get unlock an event from another core. The user can trigger a mutex lock and unlock by clicking 'm' on the terminal, or let the lock and unlock occur every 5 seconds by clicking 'a'. To verify the multicore functionality, other U-Boot commands are also needed.

In this demo, SEMA4 gate 3 is used.

7.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

7.3 System Requirement

7.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

7.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For toolchain version, please refer to the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007DRN).

7.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/sema4_demo/<toolchain>.

7.4 Getting Started

7.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

7.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
================== SEMA4 demo ==================
Enter command:
----- 'm' to manually trigger a SEMA4 lock
----- 'a' to automatically trigger SEMA4 lock every 5 seconds
```

If 'm' is pressed in terminal, the following information appears:

```
m
...SEMA4 mutex lock successfully!
Enter command:
----- 'm' to manually trigger a SEMA4 lock
----- 'a' to automatically trigger SEMA4 lock every 5 seconds
```

This shows the program succeeded in locking and freeing the SEMA4 gate. This operation can be repeated. Try locking the same SEMA4 gate on U-Boot. For example, on U-Boot terminal of i.MX 7Dual:

```
=> mw.b 0x30ac0003 1
```

Then click ’m’ on SEMA4 demo Terminal to see what happens:

```
m
...Lock pending, waiting for the other core unlock the gate
```

Now unlock the SEMA4 gate on U-Boot. For example on U-Boot terminal of i.MX 7Dual:

```
=> mw.b 0x30ac0003 0
```

You can immediately see on SEMA4 demo Terminal:

```
...SEMA4 mutex lock successfully!
Enter command:
----- ’m’ to manually trigger a SEMA4 lock
----- ’a’ to automatically trigger SEMA4 lock every 5 seconds
```

Chapter 8
Sensor Demo on i.MX 7Dual device

This demo application demonstrates the on board sensor of i.MX 7Dual SDB board.

8.1 Overview

The Sensor Demo i.MX 7Dual is a simple demonstration program that uses the FreeRTOS and a set of drivers Provided by Freescale. It can get the current gravitational acceleration, temperature, altitude and magnetic field strength of the board. The purpose of this demo is to show how to use the I2C driver as a Master to communication with other I2C Slaves.

8.2 Supported Platforms

i.MX 7Dual SABRE-SDB board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

8.3 System Requirement

8.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

8.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

8.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/demo_apps/sensor_demo/sensor_-demo_imx7d/<toolchain>.

8.4 Getting Started

8.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

8.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
-------------- iMX7D SDB on board sensor example --------------
Please select the sensor demo you want to run:
```

The user is prompted to enter which sensor they want to communicate with:

```
    [1].FXAS21002 3-axes Gyro sensor
    [2].FXOS8700 6-axes Acc+Mag sensor
    [3].MPL3115 Pressure sensor
```

After entering a valid input, the selected sensor data is sampled and displays the result in the terminal:

The FXAS21002 Gyro sensor continuously monitors current angular velocity in 3 axes, and prints to the terminal if the velocity on any axis exceeds 5.0 degrees per second, like this:

```
[FXAS21002] Rotate detected: X:  2.1dps, Y: -1.0dps, Z:  5.3dps
```

The FXOS8700 6-axes Acc+Mag sensor reads the current Acc and Mag of the board in 3 axes every 1 second, and prints the current value to the terminal, like this:

```
[FXOS8700]Current Acc:X=    -.0g Y=     .0g Z=    1.0g
[FXOS8700]Current Mag:X=  26.3uT Y=  29.7uT Z=   3.7uT
```

The MPL3115 Pressure sensor reads the current altitude and temperature of the board in 3 axes every 100 ms, and prints the current value to the terminal if the altitude delta exceed 0.5 m, or the temperature exceeds 0.3 centigrade:

```
[MPL3115]Current Height =   86.5Meter, Current Temp =  31.4Celsius
```

8.6 Note

The I2C2 instance on i.MX 7Dual SDB board is assigned to M4 Core when this demo starts. Do not use this I2C instance at A7 side when this demo is running.

Chapter 9
ADC Example

This example demonstrates how to use ADC drivers with continuous mode.

9.1 Overview

This ADC example is a demonstration program that uses the BSP software. The microcontroller is set to generate an interrupt every 1000 ms to wakes up the ADC module. Then ADC module will convert the analog input ADC1_IN3 to digital output displayed in Terminal.

9.2 Supported Platforms

These Freescale i.MX boards are supported by this demo.

- i.MX 7Dual SABRE board (SDB)

9.3 System Requirement

9.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

9.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For toolchain version, see *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007-DRN).

9.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/adc_imx7d/<toolchain>.

9.4 Getting Started

9.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Connect ADC1_IN3(JP10) with external input through dupont line with cable.
3. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
4. Load the demo binary to target board using U-Boot.
5. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

9.5 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
-------------- ADC imx7d driver example --------------

This example demonstrates usage of ADC driver on i.MX processor.
It Continuous convert Analog Input, and print the result to terminal
Current analog value: 1.06v
Current analog value: 1.04v
```

Chapter 10
ECSPI Example

10.1 Overview

This example application demonstrates how to use the ECSPI driver to transfer data between two boards with blocking mode.

- ECSPI board to board:
  - Transfer data through ECSPI2 instance, ECSPI2 pins of master board are connected with ECSPI2 pins of slave board. CS0 (chip select) of master board are connected with CS0 of slave board.
  - Master sends an array to slave and receives the array back from slave. Slave receives an array from master and sends back another array to master.

10.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual Validation board

10.3 System Requirement

10.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

10.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007DRN).

10.3.3 Software requirements

- The master project files are in: <BSP_Install>/examples/<board>/driver_examples/ecspi/ecspi_-interrupt/master/<toolchain>.
- The slave project files are in: <BSP_Install>/examples/<board>/driver_examples/ecspi/ecspi_-interrupt/slave/<toolchain>.

10.4 Getting Started

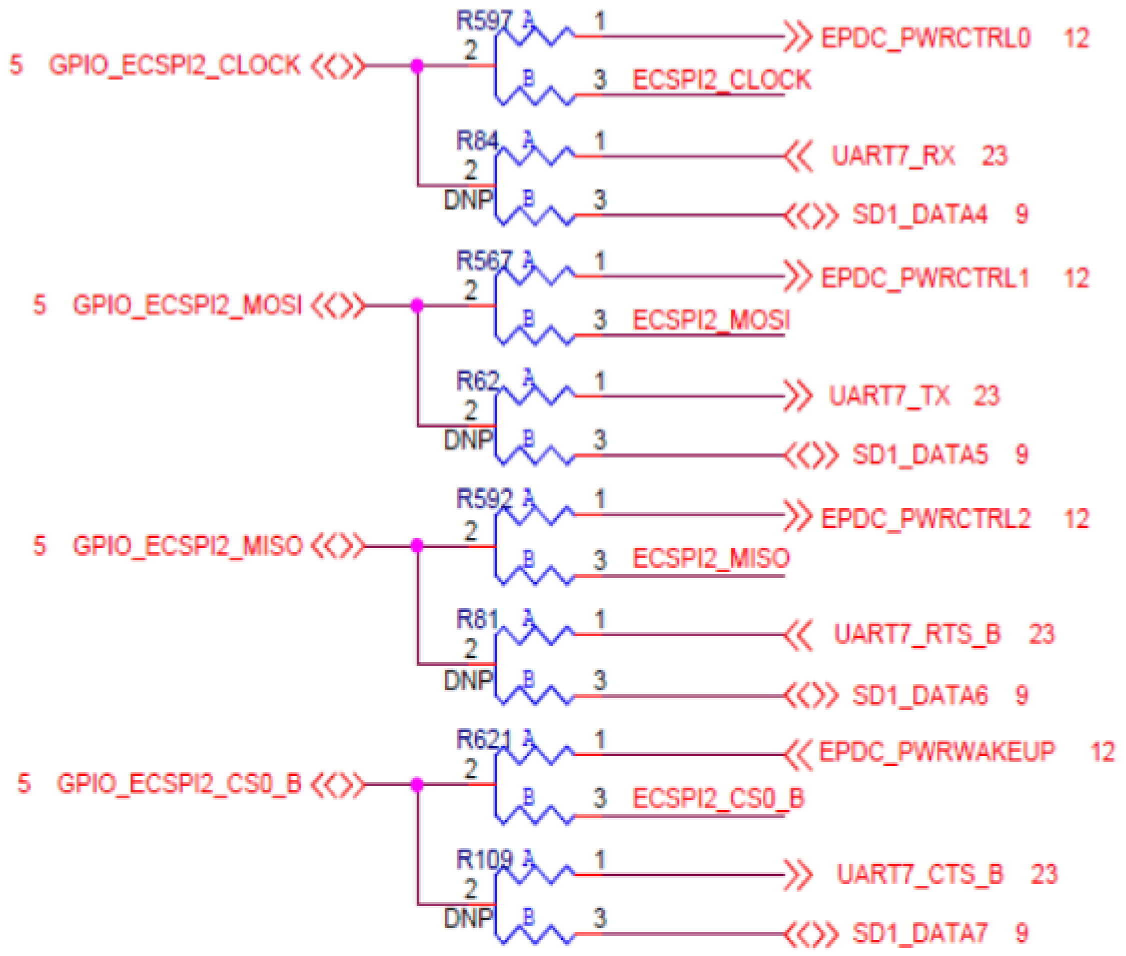
10.4.1 Hardware settings

To run this example, the ECSPI signals need to be routed on the board. The following is the rework steps required for some boards.

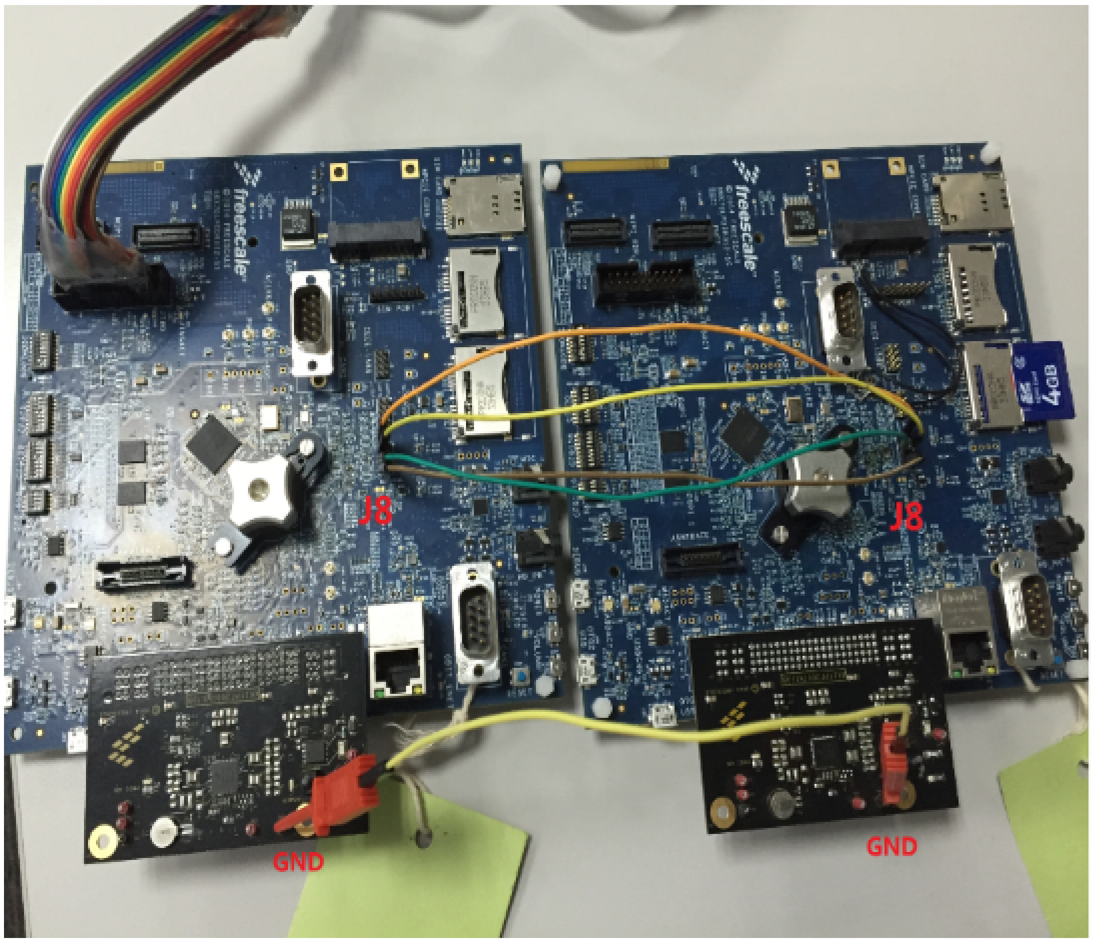
i.MX 7Dual Validation board

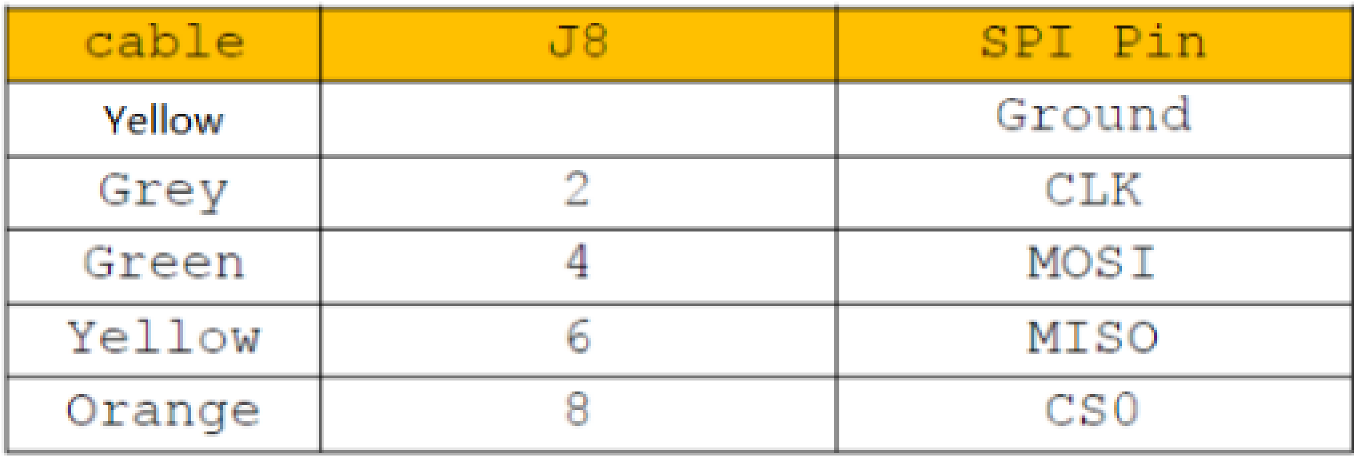
To test ECSPI master and slave example, route ECSPI2 signals to "UART7_RX", "UART7_RTS_B", "UART7_TX", and "UART7_CTS_B".

1. Remove R597, R567, R592, R621.
2. Populate R84, R62, R81, R109 to position A.



This example requires two separate boards. You can connect ECSPI2 signals on jumper "J8".





cable	J8	SPI Pin
Yellow		Ground
Grey	2	CLK
Green	4	MOSI
Yellow	6	MISO
Orange	8	CS0

J8 in the table specifies the sequential pin position (From 1 to 8) on the board.

10.4.2 Prepare the example

1. Connect 2 USB cable between the PC host and the Debug UART port on each the board.
2. Open two serial terminal for the 2 boards with following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Connect these 2 board to ECSPI2.

1. Load the demo binary to target board using U-Boot.
2. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

10.4.3 Run the example

The master example must be run before slave example, or the initialization of master SPI would cause slave example data loss. The master board prints on ARM Cortex-M4 terminal:

```
-------------- ECSPI master driver example --------------
This example application demonstrates usage of SPI driver in master mode.
It transfers data to/from remote MCU in SPI slave mode.
Press "s" when spi slave is ready.
```

Run the slave example on another board. The slave board prints on ARM Cortex-M4 terminal:

```
-------------- ECSPI slave driver example --------------
This example application demonstrates usage of ECSPI slave driver.
It responding to master via SPI bus.
SLAVE: Initial transmit data: 255
```

After ECSPI slave example is executed, press “s” to start the communication. The master board will print on terminal:

```
MASTER: Transmited data: 1
      : Received data: 255

MASTER: Transmited data: 2
      : Received data: 0

MASTER: Transmited data: 3
      : Received data: 1

...

MASTER: Transmited data: 20
      : Received data: 18
```

The slave board will print on terminal:

```
SLAVE: Next step transmit data: 0
     : Currently received data: 1

SLAVE: Next step transmit data: 1
     : Currently received data: 2

SLAVE: Next step transmit data: 2
     : Currently received data: 3

...

SLAVE: Next step transmit data: 19
     : Currently received data: 20
```

Chapter 11
FlexCAN Loopback Example

This example demonstrates how to use FlexCAN driver.

11.1 Overview

This FlexCAN Loopback example demonstrates the FlexCAN module loopback operating mode.

This example use two message buffers: one is to transmit data, and the other is to receive data. When the example starts, the example sends data from tx message buffer to its own rx message buffer, and prints the received data to terminal.

11.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

11.3 System Requirement

11.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

11.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

11.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/flexcan/flexcan_-loopback/<toolchain>.

11.4 Getting Started

11.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

11.5 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
       FLEXCAN LOOPBACK TEST *********
Message format: Standard (11 bit id)
Message buffer 9 used for Rx.
Message buffer 13 used for Tx.
Interrupt Mode: Enabled
Operating Mode: TX and RX --> LoopBack
```

After that the data will be sent through transmit MB and received from its own receive MB every 1 second. At the beginning the received data will be print to the terminal like this:

```
DLC=1, mb_idx=0x123
RX MB data: 0x0

DLC=1, mb_idx=0x123
RX MB data: 0x1

DLC=1, mb_idx=0x123
RX MB data: 0x2

DLC=1, mb_idx=0x123
RX MB data: 0x3

DLC=1, mb_idx=0x123
RX MB data: 0x4

DLC=1, mb_idx=0x123
RX MB data: 0x5
```

When the received data is up to 0xff, it will be back to 0x0.

```
DLC=1, mb_idx=0x123
RX MB data: 0xfe

DLC=1, mb_idx=0x123
RX MB data: 0xff

DLC=1, mb_idx=0x123
RX MB data: 0x0

DLC=1, mb_idx=0x123
RX MB data: 0x1

DLC=1, mb_idx=0x123
RX MB data: 0x2

DLC=1, mb_idx=0x123
RX MB data: 0x3
```

Chapter 12
FlexCAN Network Example

This example demonstrates how to use FlexCAN driver.

12.1 Overview

This FlexCAN Network example demonstrates the FlexCAN module in normal operating mode.

This example use two boards. Each board transfers data to the other, and receives data at the same time. Two message buffers are used in this example. One is used to transmit data, and the other is used to receive data. When the example starts, the example sends data from tx message buffer to the other board, and receives data from rx message buffer and prints the received data to terminal.

12.2 Supported Platforms

These Freescale i.MX boards are supported by this demo.

- i.MX 7Dual SABRE board (SDB)
- i.MX 7Dual Validation board

12.3 System Requirement

12.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

12.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007DRN).

12.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/flexcan/flexcan_-network/<toolchain>.

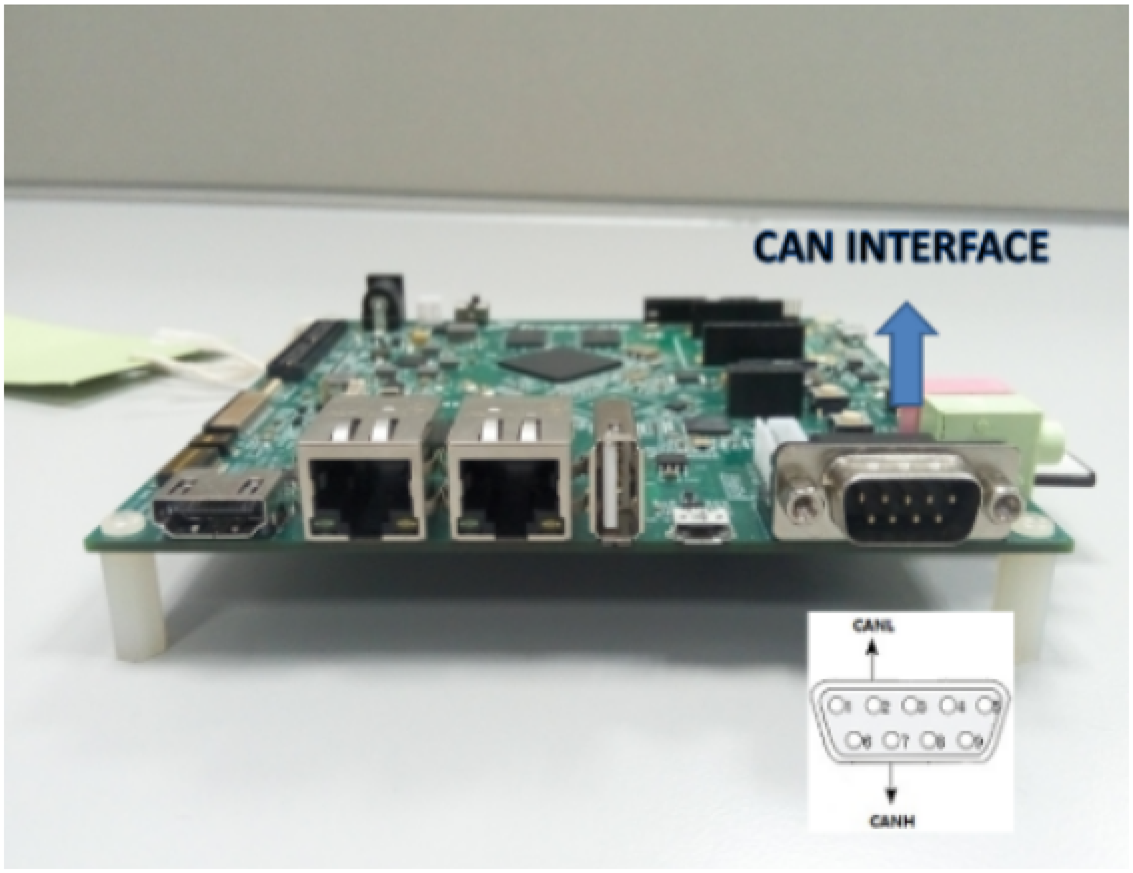
12.4 Getting Started

12.4.1 Hardware settings

To run this example, connect two board through the CAN interface:

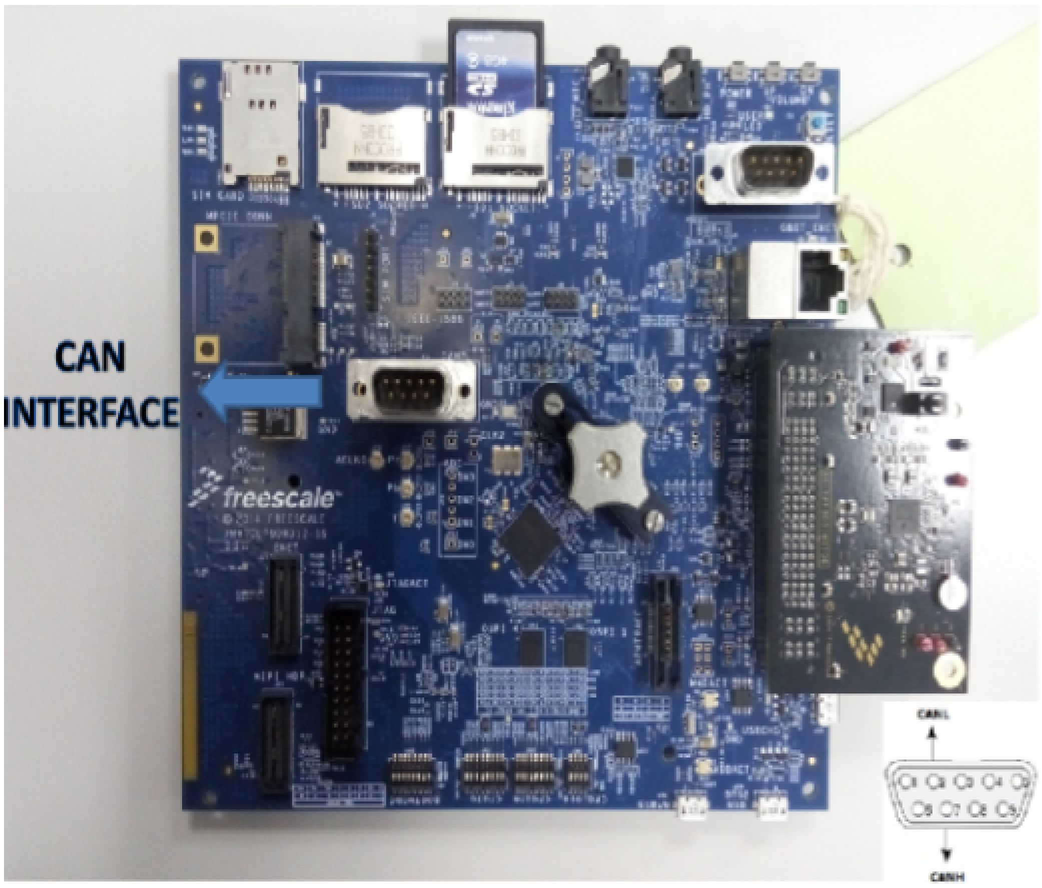
For i.MX 7Dual SABRE board

The DB-9 Connector on i.MX 7Dual SABRE board is used as the CAN interface, PIN2 connect to CANL and PIN7 connect to CANH:



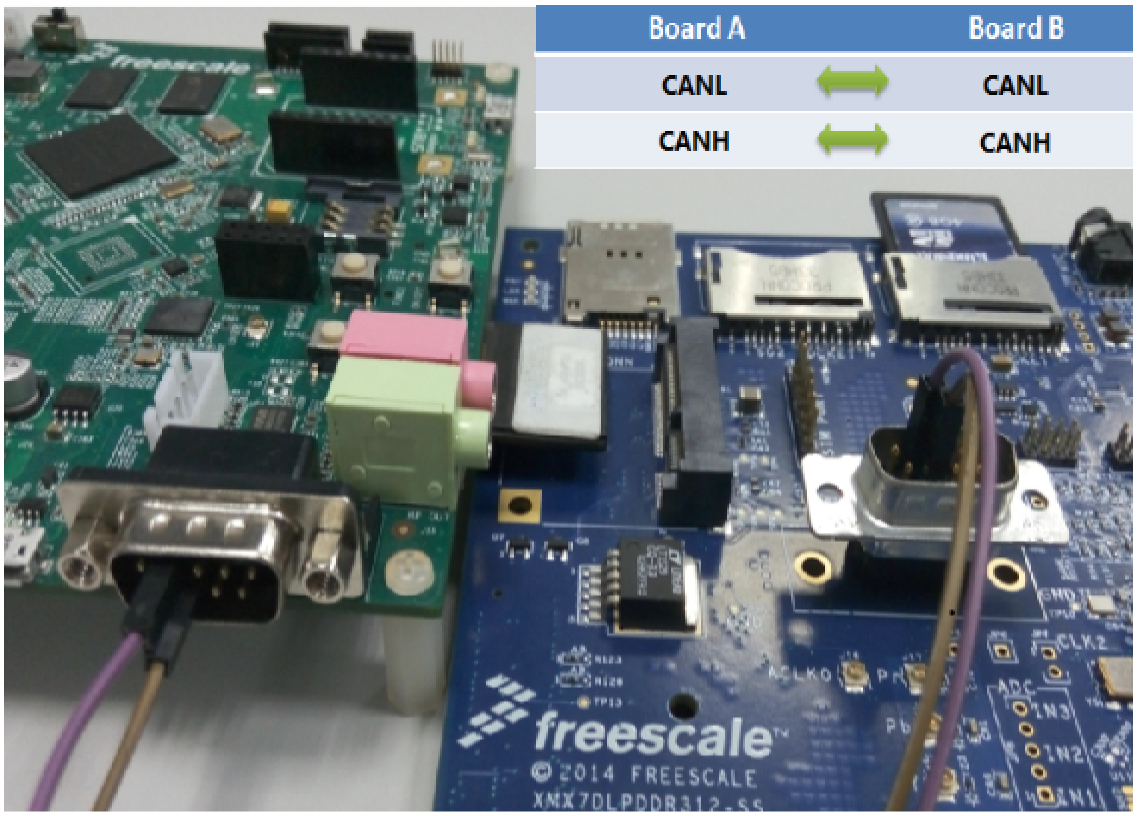
For i.MX 7Dual Validation board

The CAN2 DB-9 Connector on i.MX 7Dual SABRE board is used as the CAN interface, PIN2 connect to CANL and PIN7 connect to CANH:



Board Connection

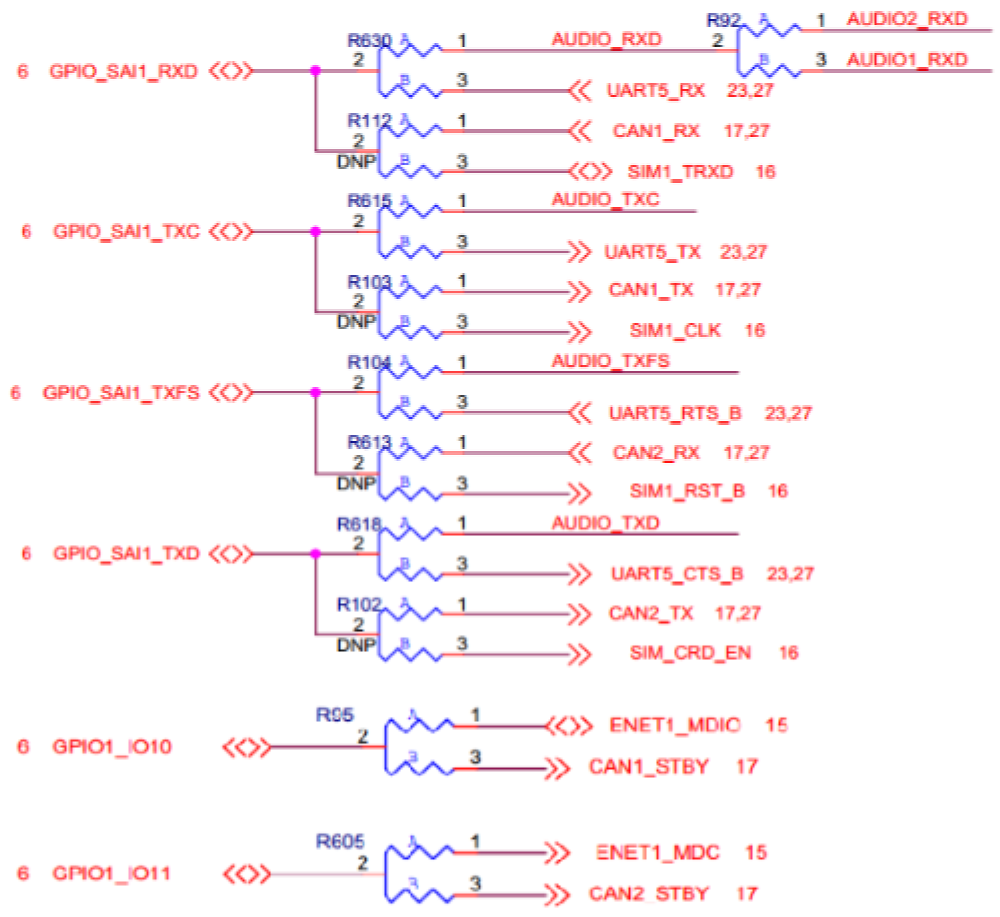
Connect two boards to CAN Bus through DB-9 Connector like this(CANH <-> CANH, CANL <-> CANL):



Board Rework for i.MX 7Dual Validation board

To test flexcan network example with i.MX 7Dual Validation board, route the following signals to "CA-Nn_RX", "CANn_TX" and "CANn_STBY"

1. Remove R630, R615, R104, R618;
2. Populate R112, R103, R613, R102 to position A;
3. Populate R112, R103, R613, R102 to position A.



12.4.2 Prepare the Demo

1. Set the Note configuration in main.c: one board set to NODE 1 (#define NODE 1) and the other set to NODE 2 (#define NODE 2).
2. Build project with different NODE configuration for these 2 board.
3. Connect 2 USB cable between the PC host and the Debug UART port on each the board.
4. Open two serial terminal for these 2 boards with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
5. Connect these 2 boards to CAN Bus.
6. Load the demo binary to target board using U-Boot.
7. Boot auxiliary Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

12.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information on each board:

```
          FLEXCAN NETWORK TEST *********
   Message format: Standard (11 bit id)
   Message buffer 9 used for Rx.
   Message buffer 8 used for Tx.
   Interrupt Mode: Enabled
   Operating Mode: TX and RX --> Normal



NODE is 2 (the NODE number you set)
```

After both of the boards are ready, the data is sent through transmit MB to the other board and receive data from the other board from its receive MB every 1 second. At the beginning, the received data prints to the terminal like this:

```
DLC=1, mb_idx=0x123
RX MB data: 0x0

DLC=1, mb_idx=0x123
RX MB data: 0x1

DLC=1, mb_idx=0x123
RX MB data: 0x2

DLC=1, mb_idx=0x123
RX MB data: 0x3

DLC=1, mb_idx=0x123
RX MB data: 0x4

DLC=1, mb_idx=0x123
RX MB data: 0x5
```

When the received data is up to 0xff, it will be back to 0x0.

```
DLC=1, mb_idx=0x123
RX MB data: 0xfe

DLC=1, mb_idx=0x123
RX MB data: 0xff

DLC=1, mb_idx=0x123
RX MB data: 0x0

DLC=1, mb_idx=0x123
RX MB data: 0x1

DLC=1, mb_idx=0x123
RX MB data: 0x2

DLC=1, mb_idx=0x123
RX MB data: 0x3
```

Chapter 13
GPIO i.MX 7Dual Example

13.1 Overview

This example application demonstrates how to use the GPIO driver to access LEDs (if existing) and Buttons on the board. If the user LED macro BOARD_GPIO_LED_CONFIG is defined in board.h, the application switches the LED on board as well.

The example provides following features:

- Control the LED(if existing) on the board.
- Configure the button as interrupt mode.
- Configure the button as general purpose IO mode and manipulate the button to control LED.

13.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

13.3 System Requirement

13.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

13.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRTOS1007DRN).

13.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/gpio_imx/<toolchain>.

13.4 Getting Started

13.4.1 Hardware settings

The GPIO Example project does not call for any special hardware configurations. Although not required, the recommendation is to leave the development board's jumper settings and configurations in default state when running this example.

13.4.2 Prepare the example

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

13.4.3 Run the example

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
====================== GPIO Example ========================

=================== GPIO Interrupt =====================
The (FUNC1) button is configured to trigger the GPIO interrupt.
Press the (FUNC1) button 3 times to continue.
```

The LED (if existing on board) is light. Press the specific button 3 times. The board prints on terminal:

```
Button pressed 1 time.
Button pressed 2 time.
Button pressed 3 time.

================= GPIO Functionality==================
The (FUNC1) button state is now polled.
Press the (FUNC1) button to switch LED on or off
```

Press the specific button on the board. The board prints on terminal:

```
Button pressed
Button released
Button pressed
Button released
Button pressed
Button released
```

If BOARD_GPIO_LED_CONFIG is defined, When the button pressed, the LED on board extinguished. When the button released, the LED is lighted.

Chapter 14
GPT Example

This example demonstrates how to use GPT driver.

14.1 Overview

This GPT example application demonstrates the GPT driver working with interrupt.

This example uses different clock sources for 2 GPT instances and capture each other's counter every second. If they both work correctly, the captured counter should be close to 0, or close to the counter represent 1 second of that GPT instance. The clock source's frequency is not 100% accurate, and the divider could also affect the clock error so that the captured number would be further than the expected value.

14.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board (SDB)

14.3 System Requirement

14.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

14.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

14.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/gpt/<toolchain>.

14.4 Getting Started

14.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

14.5 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information. The frequency and counter value might change on different hardware, but the ratio should be close to 0.0 or 1.0:

```
GPT timer will now start
counter/freq ratio should be close to 0.0 or 1.0 ...
        GPT A freq 6000000, counter 7.
        GPT B freq 49090907, counter 15.
        GPT A freq 6000000, counter 7.
        GPT B freq 49090907, counter 25.
        GPT A freq 6000000, counter 2.
        GPT B freq 49090907, counter 59.
        GPT A freq 6000000, counter 1.
        GPT B freq 49090907, counter 60.
        GPT A freq 6000000, counter 0.
        GPT B freq 49090907, counter 60.
GPT example finished...
```

Chapter 15
I2C Example

This example demonstrates how to use I2C with interrupt.

15.1 Overview

This I2C example application demonstrates the I2C driver working with interrupt.

Programming on board EEPROM through I2C bus and read back to compare if the data wrote to EEPROM are correct.

15.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual Validation board

15.3 System Requirement

15.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

15.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

15.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/i2c_imx/i2c_-interrupt/<toolchain>.

15.4 Getting Started

15.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

15.5 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
++++++++++++++++ I2C Send/Receive Interrupt Driven Example +++++++++++++++++
This example writes data to the board EEPROM through I2C Bus,
and reads them back to see if the EEPROM is programmed successfully.
```

After printing the information mentioned above, the example writes a piece of data to EEPROM and reads it back to verify if the data is wrote successfully:

```
[1].Initialize the I2C module with initialize structure.
[2].Launch a I2C write action to 0x0000 address.
[3].Prepare Data for Sending.
[4].Write data to EEPROM.
[5].Wait until transmission is finished.
[6].Launch a I2C read action from 0x0000 address.
[7].Read data from EEPROM.
[8].Wait until transmission is finished.
[9].Compare data between txBuf and rxBuf:
    txBuf and rxBuf are same, example passed!!!
```

15.6 Note

The I2C1 instance on i.MX 7Dual Validation board is assigned to M4 Core when this example start. Do not use this I2C instance at A7 side when this demo is running.

Chapter 16 UART Example

This example demonstrates how to use UART with interrupt.

16.1 Overview

This UART example demonstrates the UART driver working with interrupt.

Transfer data between the board and PC. The board transfers and receives characters with the PC through UART interface. Type characters from keyboard, and the board receives and then echoes them to terminal screen. Look for instructions output to the terminal.

16.2 Supported Platforms

These Freescale i.MX Serial SoCs SDB boards and Validation boards are supported by this demo.

- i.MX 7Dual SABRE board(SDB)

16.3 System Requirement

16.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

16.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

16.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/uart_imx/uart_-interrupt/<toolchain>.

16.4 Getting Started

16.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

16.5 Running the demo

After the boot process succeeds, the Cortex-M4 terminal displays the following information:

```
++++++++++++++++ UART Send/Receive Interrupt Driven Example +++++++++++++++++

The user needs to type characters from keyboard so the board receives and then echoes them to terminal
      screen
```

The user needs to type characters from the keyboard and the board receives and then echoes them to terminal screen.

Chapter 17
WDOG Example on i.MX

This example demonstrates how to use WDOG driver on i.MX device.

17.1 Overview

This WDOG example application demonstrates the WDOG driver working on i.MX device.

This example enables WDOG with timeout 1.5 seconds, and at the same time, an interrupt is enabled to trigger interrupt service route (ISR) 0.5 seconds before watchdog expires. In the ISR, the WDOG timer is refreshed fours times, so the WDOG does not timeout until 4 + 1.5 = 5.5 seconds.

17.2 Supported Platforms

This Freescale i.MX board is supported by this demo.

- i.MX 7Dual SABRE board(SDB)

17.3 System Requirement

17.3.1 Hardware requirements

- SD Card
- Micro USB cable
- 5V DC adapter
- Development board
- Personal Computer with USB port

17.3.2 Toolchain requirements

One of following toolchains is required:

- IAR embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.0 for i.MX 7Dual Release Notes* (document FRT-OS1007DRN).

17.3.3 Software requirements

- The project files are in: <BSP_Install>/examples/<board>/driver_examples/wdog_imx/<toolchain>.

17.4 Getting Started

17.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the Debug UART port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Boot auxiliary Cortex-M4 Core to begin running the demo.

For detailed instructions, see *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7D-GSUG).

17.5 Running the demo

After boot process succeeds, the ARM Cortex-M4 terminal displays the following information:

```
WDOG with timeout 1.5 seconds will now start
WDOG was refreshed 4
WDOG was refreshed 3
WDOG was refreshed 2
WDOG was refreshed 1
WDOG was refreshed 0
Counter down to 0, WDOG is starved now...
```

A CPU reset occurs, and the reset cannot rerun this WDOG example successfully, as this is not a full system reset and the WDOG peripheral's ENABLE control is a one-shot control. After reset, the WDOG counter cannot be reloaded, and continues running (rollover).

For example, on the i.MX 7Dual device, the reset reruns the WDOG example and shows:

```
WDOG with timeout 1.5 seconds will now start
```

Wait for more than 2 minutes to see the left information because the WDOG counter is a maximum of 128 seconds. After reset, the WDOG counter counts down from 128 seconds and triggers the interrupt again when it arrives to the timeout number of the interrupt.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: FRTOS7DDAUG
Rev. 0
08/2015