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

NXP Semiconductors

Document Number: FRTOS7D101DAUG
Rev. 0
Jul 2016



Contents

Chapter 1 Introduction

Chapter 2 LED Blinking Demo

2.1	Overview	3
2.2	Supported platforms	3
2.2.1	i.MX 7Dual SABRE board	3
2.2.1.1	Hardware requirements	3
2.2.1.2	Toolchain requirements	3
2.2.1.3	Software requirements	3
2.2.1.4	Getting started	4
2.2.1.4.1	Prepare the Demo	4
2.2.1.4.2	Running the demo	4
2.2.2	i.MX 7Dual Validation board	4
2.2.2.1	Hardware requirements	4
2.2.2.2	Toolchain requirements	5
2.2.2.3	Software requirements	5
2.2.2.4	Getting started	5
2.2.2.4.1	Prepare the Demo	5
2.2.2.4.2	Running the demo	5

Chapter 3 eCSPI Flash Demo

3.1	Overview	7
3.2	Supported platforms	7
3.2.1	i.MX 7Dual Validation board	7
3.2.1.1	Hardware requirements	7
3.2.1.2	Toolchain requirements	7
3.2.1.3	Software requirements	8

Section number	Title	Page
3.2.1.4	Getting started	8
3.2.1.4.1	Prepare the Demo	9
3.2.1.4.2	Running the demo	9

Chapter 4 Hello World DDR Demo

4.1	Overview	11
4.2	Supported platforms	11
4.2.1	i.MX 7Dual SABRE board	11
4.2.1.1	Hardware requirements	11
4.2.1.2	Toolchain requirements	11
4.2.1.3	Software requirements	11
4.2.1.4	Getting started	12
4.2.1.4.1	Prepare the Demo	12
4.2.1.4.2	Running the demo	12
4.2.2	i.MX 7Dual Validation board	12
4.2.2.1	Hardware requirements	12
4.2.2.2	Toolchain requirements	12
4.2.2.3	Software requirements	13
4.2.2.4	Getting started	13
4.2.2.4.1	Prepare the Demo	13
4.2.2.4.2	Running the demo	13

Chapter 5 Hello World Demo

5.1	Overview	15
5.2	Supported platforms	15
5.2.1	i.MX 7Dual SABRE board	15
5.2.1.1	Hardware requirements	15
5.2.1.2	Toolchain requirements	15
5.2.1.3	Software requirements	15
5.2.1.4	Getting started	16
5.2.1.4.1	Prepare the Demo	16
5.2.1.4.2	Running the demo	16
5.2.2	i.MX 7Dual Validation board	16
5.2.2.1	Hardware requirements	16
5.2.2.2	Toolchain requirements	16
5.2.2.3	Software requirements	17

Section number	Title	Page
5.2.2.4	Getting started	17
5.2.2.4.1	Prepare the Demo	17
5.2.2.4.2	Running the demo	17

Chapter 6 Hello World OCRAM Demo

6.1	Overview	19
6.2	Supported platforms	19
6.2.1	i.MX 7Dual SABRE board	19
6.2.1.1	Hardware requirements	19
6.2.1.2	Toolchain requirements	19
6.2.1.3	Software requirements	19
6.2.1.4	Getting started	20
6.2.1.4.1	Prepare the Demo	20
6.2.1.4.2	Running the demo	20
6.2.2	i.MX 7Dual Validation board	20
6.2.2.1	Hardware requirements	20
6.2.2.2	Toolchain requirements	20
6.2.2.3	Software requirements	21
6.2.2.4	Getting started	21
6.2.2.4.1	Prepare the Demo	21
6.2.2.4.2	Running the demo	21

Chapter 7 Hello World QSPI Demo

7.1	Overview	23
7.2	Supported platforms	23
7.2.1	i.MX 7Dual SABRE board	23
7.2.1.1	Hardware requirements	23
7.2.1.2	Toolchain requirements	23
7.2.1.3	Software requirements	23
7.2.1.4	Getting started	24
7.2.1.4.1	Hardware settings	24
7.2.1.4.2	Prepare the Demo	24
7.2.1.4.3	Running the demo	25
7.2.2	i.MX 7Dual Validation board	25
7.2.2.1	Hardware requirements	25
7.2.2.2	Toolchain requirements	25

Section number	Title	Page
7.2.2.3	Software requirements	26
7.2.2.4	Getting started	26
7.2.2.4.1	Hardware settings	26
7.2.2.4.2	Prepare the Demo	26
7.2.2.4.3	Running the demo	27

Chapter 8 Low Power Random WFI Demo

8.1	Overview	29
8.2	Supported platforms	29
8.2.1	i.MX 7Dual SABRE board	29
8.2.1.1	Hardware requirements	29
8.2.1.2	Toolchain requirements	29
8.2.1.3	Software requirements	30
8.2.1.4	Getting started	30
8.2.1.4.1	Prepare the Demo	30
8.2.1.4.2	Running the demo	30

Chapter 9 RPMMsg Ping-Pong Bare Metal Demo

9.1	Overview	33
9.2	Supported platforms	33
9.2.1	i.MX 7Dual SABRE board	33
9.2.1.1	Hardware requirements	33
9.2.1.2	Toolchain requirements	33
9.2.1.3	Software requirements	33
9.2.1.4	Getting started	34
9.2.1.4.1	Prepare the Demo	34
9.2.1.4.2	Running the demo	34
9.2.2	i.MX 7Dual Validation board	35
9.2.2.1	Hardware requirements	35
9.2.2.2	Toolchain requirements	35
9.2.2.3	Software requirements	35
9.2.2.4	Getting started	35
9.2.2.4.1	Prepare the Demo	35
9.2.2.4.2	Running the demo	36

Section number	Title	Page
----------------	-------	------

Chapter 10
RPMMsg Ping-Pong FreeRTOS Demo with RTOS API

10.1	Overview	37
10.2	Supported platforms	37
10.2.1	i.MX 7Dual SABRE board	37
10.2.1.1	Hardware requirements	37
10.2.1.2	Toolchain requirements	37
10.2.1.3	Software requirements	37
10.2.1.4	Getting started	38
10.2.1.4.1	Prepare the Demo	38
10.2.1.4.2	Running the demo	38
10.2.2	i.MX 7Dual Validation board	39
10.2.2.1	Hardware requirements	39
10.2.2.2	Toolchain requirements	39
10.2.2.3	Software requirements	39
10.2.2.4	Getting started	39
10.2.2.4.1	Prepare the Demo	39
10.2.2.4.2	Running the demo	40

Chapter 11
RPMMsg String Echo Bare Metal Demo

11.1	Overview	41
11.2	Supported platforms	41
11.2.1	i.MX 7Dual SABRE board	41
11.2.1.1	Hardware requirements	41
11.2.1.2	Toolchain requirements	41
11.2.1.3	Software requirements	42
11.2.1.4	Getting started	42
11.2.1.4.1	Prepare the Demo	42
11.2.1.4.2	Running the demo	42
11.2.2	i.MX 7Dual Validation board	43
11.2.2.1	Hardware requirements	43
11.2.2.2	Toolchain requirements	43
11.2.2.3	Software requirements	44
11.2.2.4	Getting started	44
11.2.2.4.1	Prepare the Demo	44
11.2.2.4.2	Running the demo	44

Section number	Title	Page
Chapter 12		
RPMMsg String Echo FreeRTOS Demo with RTOS API		
12.1	Overview	47
12.2	Supported platforms	47
12.2.1	i.MX 7Dual SABRE board	47
12.2.1.1	Hardware requirements	47
12.2.1.2	Toolchain requirements	47
12.2.1.3	Software requirements	48
12.2.1.4	Getting started	48
12.2.1.4.1	Prepare the Demo	48
12.2.1.4.2	Running the demo	48
12.2.2	i.MX 7Dual Validation board	49
12.2.2.1	Hardware requirements	49
12.2.2.2	Toolchain requirements	49
12.2.2.3	Software requirements	50
12.2.2.4	Getting started	50
12.2.2.4.1	Prepare the Demo	50
12.2.2.4.2	Running the demo	50

Chapter 13

SEMA4 Mutex Demo

13.1	Overview	53
13.2	Supported platforms	53
13.2.1	i.MX 7Dual SABRE board	53
13.2.1.1	Hardware requirements	53
13.2.1.2	Toolchain requirements	53
13.2.1.3	Software requirements	53
13.2.1.4	Getting started	54
13.2.1.4.1	Prepare the Demo	54
13.2.1.4.2	Running the demo	54
13.2.2	i.MX 7Dual Validation board	55
13.2.2.1	Hardware requirements	55
13.2.2.2	Toolchain requirements	55
13.2.2.3	Software requirements	55
13.2.2.4	Getting started	56
13.2.2.4.1	Prepare the Demo	56
13.2.2.4.2	Running the demo	56

Section number	Title	Page
Chapter 14		
Sensor Demo		
14.1	Overview	59
14.2	Supported platforms	59
14.2.1	i.MX 7Dual SABRE board	59
14.2.1.1	Hardware requirements	59
14.2.1.2	Toolchain requirements	59
14.2.1.3	Software requirements	59
14.2.1.4	Getting started	60
14.2.1.4.1	Prepare the Demo	60
14.2.1.4.2	Running the demo	60

Chapter 15

ADC Example

15.1	Overview	63
15.2	Supported platforms	63
15.2.1	i.MX 7Dual SABRE board	63
15.2.1.1	Hardware requirements	63
15.2.1.2	Toolchain requirements	63
15.2.1.3	Software requirements	63
15.2.1.4	Getting started	64
15.2.1.4.1	Prepare the Demo	64
15.2.1.4.2	Running the demo	64
15.2.2	i.MX 7Dual Validation board	64
15.2.2.1	Hardware requirements	64
15.2.2.2	Toolchain requirements	64
15.2.2.3	Software requirements	65
15.2.2.4	Getting started	65
15.2.2.4.1	Prepare the Demo	65
15.2.2.4.2	Running the demo	65

Chapter 16

eCSPI Interrupt Example

16.1	Overview	67
16.2	Supported platforms	67

Section number	Title	Page
16.2.1	i.MX 7Dual Validation board	67
16.2.1.1	Hardware requirements	67
16.2.1.2	Toolchain requirements	67
16.2.1.3	Software requirements	67
16.2.1.4	Getting started	68
16.2.1.4.1	Hardware settings	68
16.2.1.4.2	Preparing the demo	70
16.2.1.4.3	Running the demo	70

Chapter 17 eCSPI Polling Example

17.1	Overview	73
17.2	Supported platforms	73
17.2.1	i.MX 7Dual Validation board	73
17.2.1.1	Hardware requirements	73
17.2.1.2	Toolchain requirements	73
17.2.1.3	Software requirements	73
17.2.1.4	Getting started	74
17.2.1.4.1	Hardware settings	74
17.2.1.4.2	Preparing the demo	76
17.2.1.4.3	Running the demo	76

Chapter 18 FlexCAN Loopback Example

18.1	Overview	79
18.2	Supported platforms	79
18.2.1	i.MX 7Dual SABRE board	79
18.2.1.1	Hardware requirements	79
18.2.1.2	Toolchain requirements	79
18.2.1.3	Software requirements	79
18.2.1.4	Getting started	80
18.2.1.4.1	Prepare the Demo	80
18.2.1.4.2	Running the demo	80
18.2.2	i.MX 7Dual Validation board	81
18.2.2.1	Hardware requirements	81
18.2.2.2	Toolchain requirements	81
18.2.2.3	Software requirements	81
18.2.2.4	Getting started	82

Section number	Title	Page
18.2.2.4.1	Prepare the Demo	82
18.2.2.4.2	Running the demo	82

Chapter 19 FlexCAN Network Example

19.1	Overview	85
19.2	Supported platforms	85
19.2.1	i.MX 7Dual SABRE board	85
19.2.1.1	Hardware requirements	85
19.2.1.2	Toolchain requirements	85
19.2.1.3	Software requirements	85
19.2.1.4	Getting started	86
19.2.1.4.1	Hardware settings	86
19.2.1.4.2	Prepare the Demo	87
19.2.1.4.3	Running the demo	88
19.2.2	i.MX 7Dual Validation board	89
19.2.2.1	Hardware requirements	89
19.2.2.2	Toolchain requirements	89
19.2.2.3	Software requirements	89
19.2.2.4	Getting started	89
19.2.2.4.1	Hardware settings	89
19.2.2.4.2	Prepare the Demo	91
19.2.2.4.3	Running the demo	92

Chapter 20 GPIO Example

20.1	Overview	93
20.2	Supported platforms	93
20.2.1	i.MX 7Dual SABRE board	93
20.2.1.1	Hardware requirements	93
20.2.1.2	Toolchain requirements	93
20.2.1.3	Software requirements	93
20.2.1.4	Getting started	94
20.2.1.4.1	Preparing the demo	94
20.2.1.4.2	Running the demo	94
20.2.2	i.MX 7Dual Validation board	94
20.2.2.1	Hardware requirements	94
20.2.2.2	Toolchain requirements	95

Section number	Title	Page
20.2.2.3	Software requirements	95
20.2.2.4	Getting started	95
20.2.2.4.1	Preparing the demo	95
20.2.2.4.2	Running the demo	95

Chapter 21 GPT Example

21.1	Overview	97
21.2	Supported platforms	97
21.2.1	i.MX 7Dual SABRE board	97
21.2.1.1	Hardware requirements	97
21.2.1.2	Toolchain requirements	97
21.2.1.3	Software requirements	97
21.2.1.4	Getting started	98
21.2.1.4.1	Prepare the Demo	98
21.2.1.4.2	Running the demo	98
21.2.2	i.MX 7Dual Validation board	98
21.2.2.1	Hardware requirements	98
21.2.2.2	Toolchain requirements	99
21.2.3	Software requirements	99
21.2.3.1	Getting started	99
21.2.3.1.1	Prepare the Demo	99
21.2.3.1.2	Running the demo	99

Chapter 22 I2C Interrupt EEPROM Example

22.1	Overview	101
22.2	Supported platforms	101
22.2.1	i.MX 7Dual Validation board	101
22.2.1.1	Hardware requirements	101
22.2.2	Toolchain requirements	101
22.2.2.1	Software requirements	101
22.2.2.2	Getting started	102
22.2.2.2.1	Prepare the Demo	102
22.2.2.2.2	Running the demo	102

Section number	Title	Page
Chapter 23		
I2C Interrupt Sensor Example		
23.1	Overview	103
23.2	Supported platforms	103
23.2.1	i.MX 7Dual SABRE board	103
23.2.1.1	Hardware requirements	103
23.2.1.2	Toolchain requirements	103
23.2.1.3	Software requirements	103
23.2.1.4	Getting started	104
23.2.1.4.1	Prepare the Demo	104
23.2.1.4.2	Running the demo	104
Chapter 24		
I2C Polling EEPROM Example		
24.1	Overview	105
24.2	Supported platforms	105
24.2.1	i.MX 7Dual Validation board	105
24.2.1.1	Hardware requirements	105
24.2.2	Toolchain requirements	105
24.2.2.1	Software requirements	105
24.2.2.2	Getting started	106
24.2.2.2.1	Prepare the Demo	106
24.2.2.2.2	Running the demo	106
Chapter 25		
I2C Polling Sensor Example		
25.1	Overview	107
25.2	Supported platforms	107
25.2.1	i.MX 7Dual SABRE board	107
25.2.1.1	Hardware requirements	107
25.2.1.2	Toolchain requirements	107
25.2.1.3	Software requirements	107
25.2.1.4	Getting started	108
25.2.1.4.1	Prepare the Demo	108
25.2.1.4.2	Running the demo	108

Section number	Title	Page
Chapter 26		
UART Interrupt Example		
26.1	Overview	109
26.2	Supported platforms	109
26.2.1	i.MX 7Dual SABRE board	109
26.2.1.1	Hardware requirements	109
26.2.1.2	Toolchain requirements	109
26.2.1.3	Software requirements	109
26.2.1.4	Getting started	110
26.2.1.4.1	Prepare the Demo	110
26.2.1.4.2	Running the demo	110
26.2.2	i.MX 7Dual Validation board	110
26.2.2.1	Hardware requirements	110
26.2.2.2	Toolchain requirements	110
26.2.2.3	Software requirements	111
26.2.2.4	Getting started	111
26.2.2.4.1	Prepare the Demo	111
26.2.2.4.2	Running the demo	111

Chapter 27
UART Polling Example

27.1	Overview	113
27.2	Supported platforms	113
27.2.1	i.MX 7Dual SABRE board	113
27.2.1.1	Hardware requirements	113
27.2.1.2	Toolchain requirements	113
27.2.1.3	Software requirements	113
27.2.1.4	Getting started	114
27.2.1.4.1	Prepare the Demo	114
27.2.1.4.2	Running the demo	114
27.2.2	i.MX 7Dual Validation board	114
27.2.2.1	Hardware requirements	114
27.2.2.2	Toolchain requirements	114
27.2.2.3	Software requirements	115
27.2.2.4	Getting started	115
27.2.2.4.1	Prepare the Demo	115
27.2.2.4.2	Running the demo	115

Section number	Title	Page
Chapter 28		
WDOG Example		
28.1	Overview	117
28.2	Supported platforms	117
28.2.1	i.MX 7Dual SABRE board	117
28.2.1.1	Hardware requirements	117
28.2.1.2	Toolchain requirements	117
28.2.1.3	Software requirements	117
28.2.1.4	Getting started	118
28.2.1.4.1	Prepare the Demo	118
28.2.1.4.2	Running the demo	118
28.2.2	i.MX 7Dual Validation board	118
28.2.2.1	Hardware requirements	118
28.2.2.2	Toolchain requirements	119
28.2.2.3	Software requirements	119
28.2.2.4	Getting started	119
28.2.2.4.1	Prepare the Demo	119
28.2.2.4.2	Running the demo	119

Chapter 1

Introduction

FreeRTOS™ BSP 1.0.1 i.MX 7Dual includes applications, which provide examples that show how to use this BSP for i.MX 7Dual Processor. This document describes applications and provides instructions to configure each application (if available). The document also describes the required board setup and steps to run the applications.

This document does not cover the details about compiling and running using the Linux® OS. To run the examples with Linux OS, note the following:

- By default, Linux BSP does not come with ARM® Cortex® -M4 enabled. The device tree needs to be changed in U-Boot with the command "setenv fdt_file zImage-imx7d-sdb-m4.dtb" for SABRE--SD board or "setenv fdt_file zImage-imx7d-12x12-lpddr3-arm2-m4.dtb" for validation board.
- Add "uart_from_osc" to the U-Boot "bootargs" variable to make sure that the Linux UART driver uses OSC as the clock source to comply with the FreeRTOS application. Otherwise, it gets messy output messages.
- To run RPMsg demos, configure and build the corresponding RPMsg kernel modules in Linux OS.



Chapter 2 LED Blinking Demo

2.1 Overview

This demo use GPIO and GPT driver as well as RDC SEMAPHORE driver to demonstrates how to toggle a blinking LED or printing "+" and "-" on the Terminal with different frequencies. In this demo, a safe shared peripheral access way is introduced with RDC SEMAPHORE.

NOTE: Sharing of GPIO need deep customization in Linux OS kernel, so it is not recommended to run this demo with the default Linux OS kernel together.

2.2 Supported platforms

2.2.1 i.MX 7Dual SABRE board

2.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal Computer with USB port

2.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

2.2.1.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_sdb_m4/demo_apps/blinking_imx_demo/<toolchain>`.

Supported platforms

2.2.1.4 Getting started

2.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

2.2.1.4.2 Running the demo

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

```
===== Blinking Demo =====  
  
===== Blinking interval 100ms =====  
  
Press the (FUNC1) key to switch the blinking frequency:  
- + - + - + - + - + - + - + - + -
```

After the user presses the "FUNC1" 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 UART print slower and slower until the print interval increases to 1000 ms. The user can press the button again to recover the print interval to 100 ms.

2.2.2 i.MX 7Dual Validation board

2.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

2.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

2.2.2.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_val_m4/demo_apps/blinking_imx_demo/<toolchain>`.

2.2.2.4 Getting started

2.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

2.2.2.4.2 Running the demo

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

```
===== Blinking Demo =====
===== Blinking interval 100ms =====
Press the (VOLUME UP) key to switch the blinking frequency:
```

After the user presses the "VOLUME UP" button, something new appears on the terminal, and the blinking runs much slower:

Supported platforms

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

Press the (VOLUME UP) key to **switch** the blinking frequency:

The Debug LED blinks become slower every 5 seconds. And it will recover the blinking interval to 100 ms until the blinking interval increases to 1000 ms.

Chapter 3 eCSPI Flash Demo

3.1 Overview

This demo application demonstrates the eCSPI driver working with FreeRTOS OS and how to use the eCSPI driver to access SPI flash memory. The demo provides following SPI flash features:

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

NOTE: The eCSPI1 instance on i.MX 7Dual Validation board is assigned to the ARM Cortex-M4 core when this example starts. Do not use this demo when Linux OS is running at the ARM Cortex-A7 core, because eCSPI1 is occupied by ARM Cortex-A7 when Linux OS kernel boots.

3.2 Supported platforms

3.2.1 i.MX 7Dual Validation board

3.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

3.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

Supported platforms

3.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/ecspi_flash_demo/<toolchain>.

3.2.1.4 Getting started

Hardware settings

To run this demo, enable the eCSPI flash on this board. The following rework steps is required: Route eCSPI1 signals to "ECSPI1_CLOCK", "ECSPI1_MOSI", "ECSPI1_MISO", and "ECSPI1_CS0_B".

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

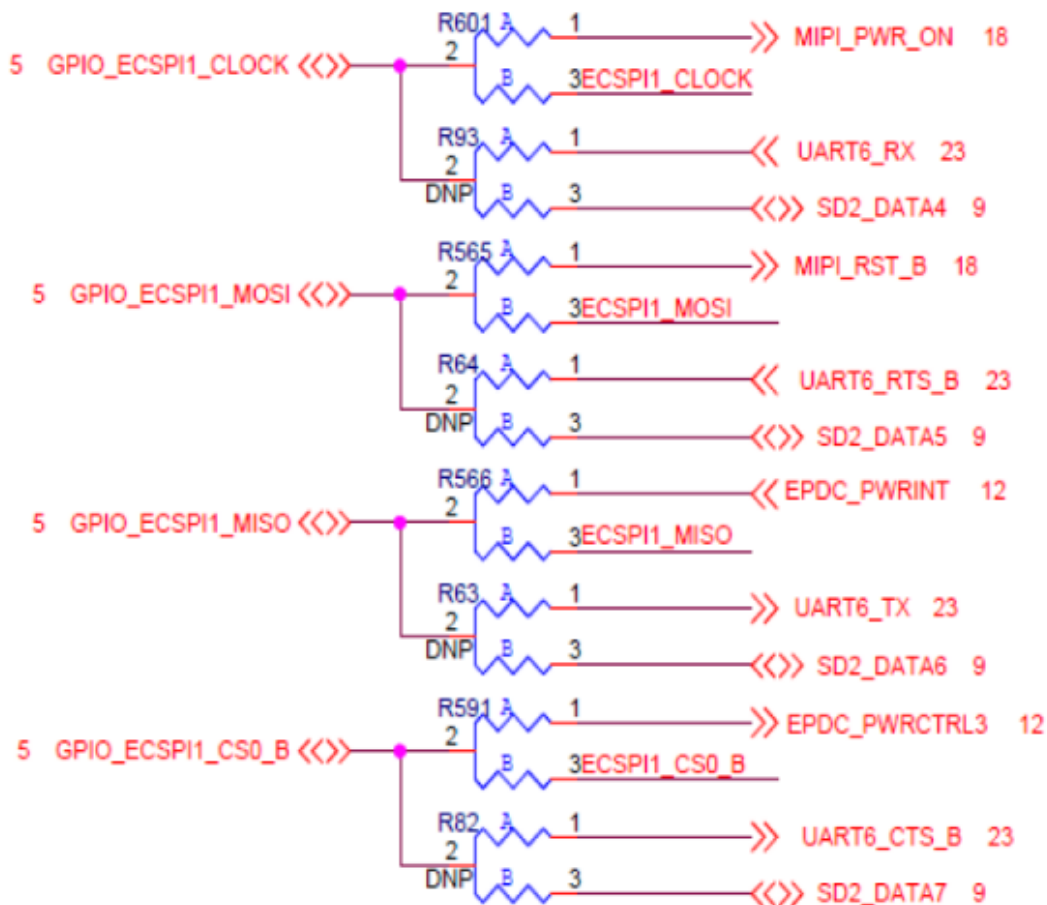


Figure 3.2.1: eCSPI rework guidance schematic diagram for i.MX 7Dual Validation board

3.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).


3.2.1.4.2 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!
```



Supported platforms

Chapter 4

Hello World DDR Demo

4.1 Overview

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

In addition to that, RDC memory protection is introduced in hardware_init.c. The FreeRTOS code space read/write operation is only allowed by the ARM Cortex-M4 core.

NOTE: The DDR memory is system resource and make sure there is no conflict in memory allocation between the ARM Cortex-A7 core and ARM Cortex-M4 core. The code space of FreeRTOS demo is defined in platform/devices/MCIMX7D/linker/<toolchain>/MCIMX7D_M4_ddr.<ext> and the base address and length of the code region can be changed per requirement.

4.2 Supported platforms

4.2.1 i.MX 7Dual SABRE board

4.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

4.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

4.2.1.3 Software requirements

- The Hello World DDR project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/hello_world_ddr/<toolchain>.

Supported platforms

4.2.1.4 Getting started

4.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the DDR using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

4.2.1.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```

4.2.2 i.MX 7Dual Validation board

4.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

4.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

4.2.2.3 Software requirements

- The Hello World DDR project files are in: `<BSP_Install>/examples/imx7d_val_m4/demo_apps/hello_world_ddr/<toolchain>`.

4.2.2.4 Getting started

4.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port(J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the DDR using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

4.2.2.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```



Supported platforms

Chapter 5 Hello World Demo

5.1 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.

5.2 Supported platforms

5.2.1 i.MX 7Dual SABRE board

5.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

5.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

5.2.1.3 Software requirements

- The hello world project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/hello_world/<toolchain>.

Supported platforms

5.2.1.4 Getting started

5.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

5.2.1.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```

5.2.2 i.MX 7Dual Validation board

5.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

5.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

5.2.2.3 Software requirements

- The hello world project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/hello_world/<toolchain>.

5.2.2.4 Getting started

5.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).


5.2.2.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```



Supported platforms

Chapter 6

Hello World OCRAM Demo

6.1 Overview

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

In addition to that, RDC memory protection is introduced in hardware_init.c. The FreeRTOS code space read/write operation is allowed by both Cortex-A7 and Cortex-M4 core.

NOTE: The OCRAM memory is a system resource and make sure there is no conflict in memory allocation between the ARM Cortex-A7 core and ARM Cortex-M4 core. The code space of FreeRTOS demo is defined in platform/devices/MCIMX7D/linker/<toolchain>/MCIMX7D_M4_ocram.<ext> and the base address and length of the code region can be changed per requirement.

6.2 Supported platforms

6.2.1 i.MX 7Dual SABRE board

6.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

6.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

6.2.1.3 Software requirements

- The hello world ocram project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/hello_world_ocram/<toolchain>.

Supported platforms

6.2.1.4 Getting started

6.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the OCRAM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

6.2.1.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```

6.2.2 i.MX 7Dual Validation board

6.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

6.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

6.2.2.3 Software requirements

- The hello world ocram project files are in: `<BSP_Install>/examples/imx7d_val_m4/demo_apps/hello_world_ocram/<toolchain>`.

6.2.2.4 Getting started

6.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the OCRAM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).


6.2.2.4.2 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```



Supported platforms

Chapter 7

Hello World QSPI Demo

7.1 Overview

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

In addition to that, RDC memory protection is introduced in hardware_init.c. The FreeRTOS code space read/write operation is only allowed by the ARM Cortex-M4 core.

NOTE:

1. Do not use this demo and Linux OS on the ARM Cortex-A7 core at the same time, because the IO Pins for QSPI Flash on i.MX 7Dual Validation board are occupied by Linux OS for EPDC display.
2. To start this demo, the QSPI Flash in U-Boot should be enabled.

7.2 Supported platforms

7.2.1 i.MX 7Dual SABRE board

7.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

7.2.1.2 Toolchain requirements

One of the following toolchains is required:

- IAR Embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

7.2.1.3 Software requirements

- The Hello World QSPI project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/hello_world_qspi/<toolchain>.

Supported platforms

- To start the Hello World QSPI demo, U-Boot must be configured with QSPI flash enabled.

7.2.1.4 Getting started

7.2.1.4.1 Hardware settings

To enable FreeRTOS application running on the QSPI flash, perform the following board rework:

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

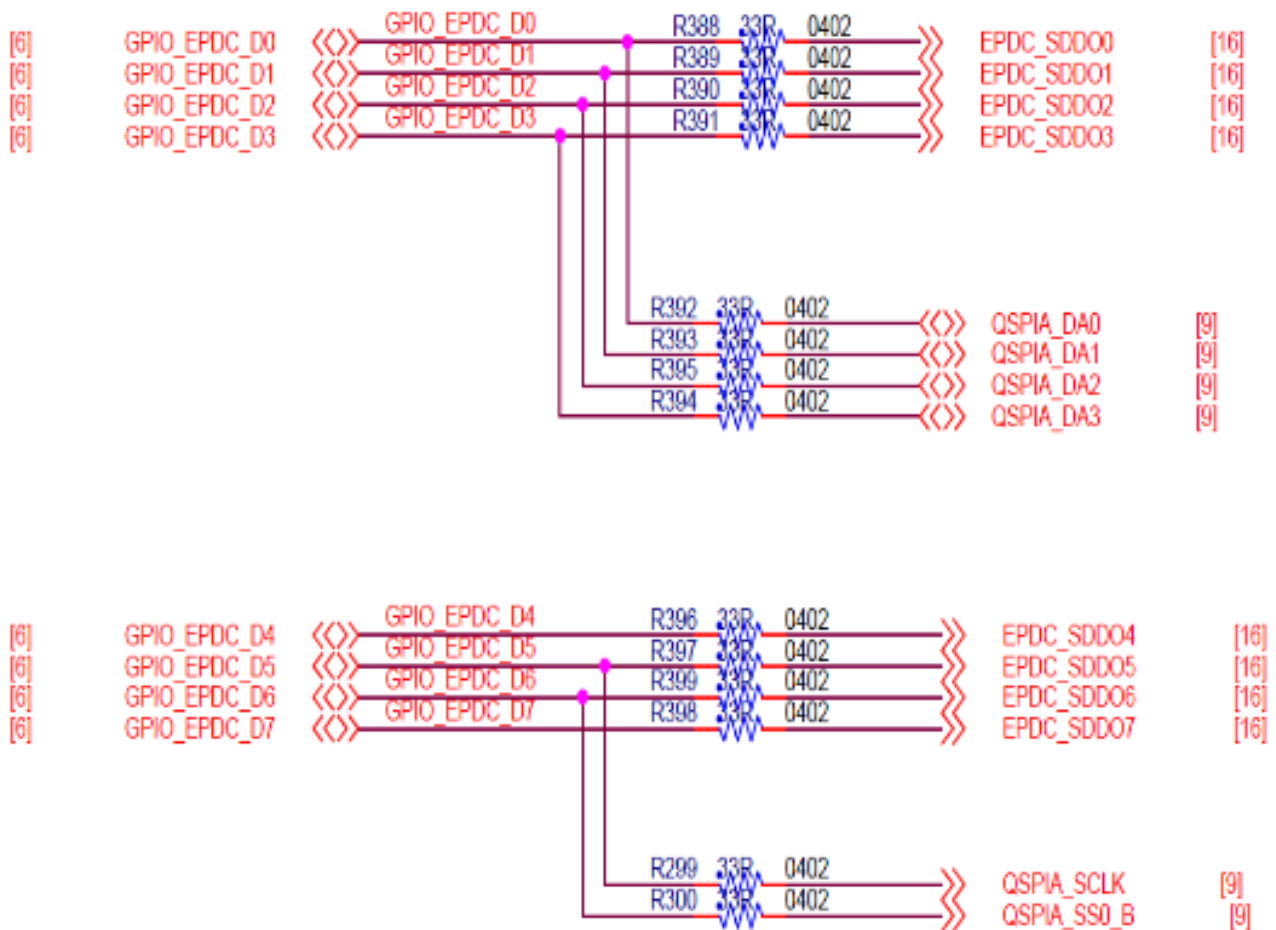


Figure 7.2.1: QSPI rework guidance schematic diagram for i.MX 7Dual SDB board

7.2.1.4.2 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:

- 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to DDR using U-Boot. Write the image from DDR to QSPI flash.
 4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

7.2.1.4.3 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```

7.2.2 i.MX 7Dual Validation board

7.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

7.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

Supported platforms

7.2.2.3 Software requirements

- The Hello World QSPI project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/hello_world_qspi/<toolchain>.
- To start the Hello World QSPI demo, U-Boot must be configured with QSPI flash enabled.

7.2.2.4 Getting started

7.2.2.4.1 Hardware settings

To enable FreeRTOS application running on the QSPI flash, perform the following board rework: -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.

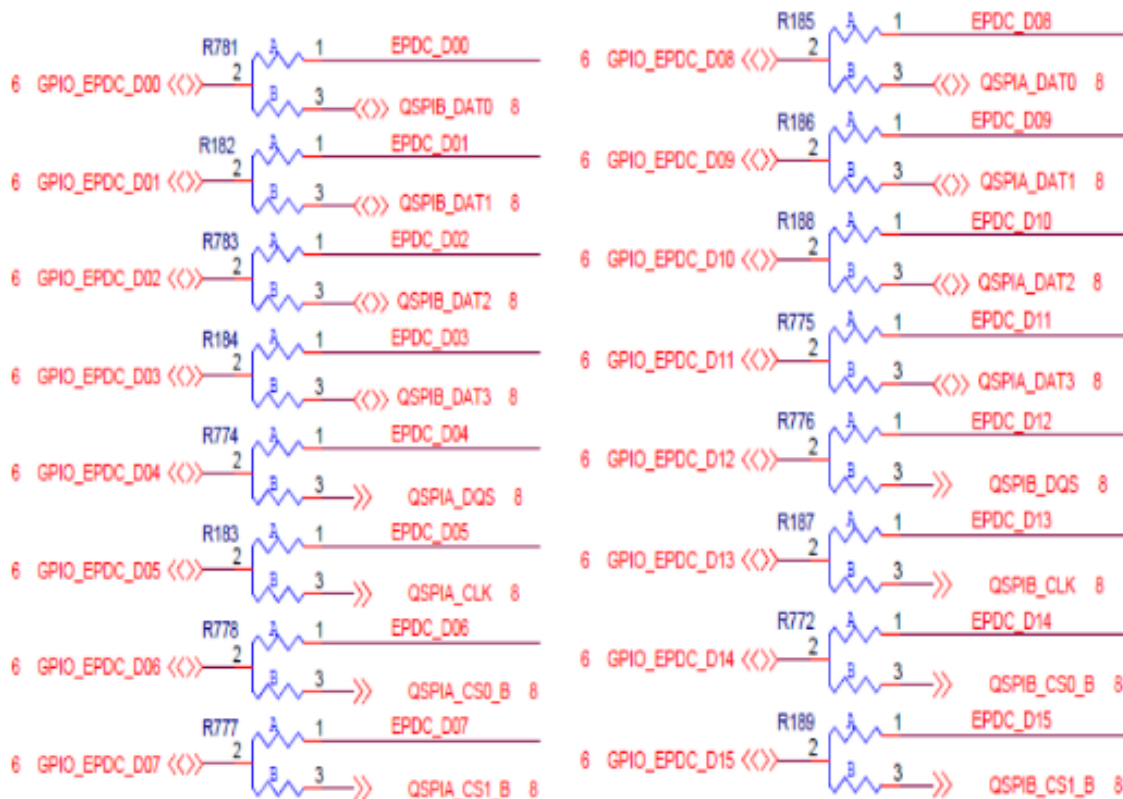


Figure 7.2.2: QSPI rework guidance schematic diagram for i.MX 7Dual Validation board

7.2.2.4.2 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.

2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to DDR using U-Boot. Write the image from DDR to QSPI flash.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

7.2.2.4.3 Running the demo

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

```
Hello World!
```

You can put some inputs like this :

```
Hello World!  
test
```



Supported platforms

Chapter 8

Low Power Random WFI Demo

This demo application exhibits the low-power feature of i.MX 7Dual SoC.

8.1 Overview

This demo exhibits dual-core low-power management. i.MX 7Dual is a dual-core chip, including an ARM Cortex-A7 core and an ARM Cortex-M4 core. Both the ARM Cortex-A7 core and the ARM Cortex-M4 core have different power modes - RUN, WAIT, and STOP. During WAIT and STOP, the core clock is gated off. The core stops running and enters low-power mode. Peripheral interrupt can be registered and wake up the core from low-power mode. Other than that, ARM Cortex-A7 core can do more power saving by shutting down the PLL, making DDR in self-refresh, etc. ARM Cortex-M4 core and its peripherals may need these resources, so the application on ARM Cortex-M4 should supervise the state of all its peripherals. When all of them do not need the high-power resources, ARM Cortex-M4 can inform ARM Cortex-A7 about it. ARM Cortex-A7 can then actually shut down these resources based on the system-level situation (including ARM Cortex-M4 and other peripherals directly managed by ARM Cortex-A7).

This demo is based on the GPT timer on the ARM Cortex-M4 side. With the timer, the ARM Cortex-M4 can switch between RUN, WAIT, and STOP modes continuously. ARM Cortex-M4 core also switches its clock source from PLL2 to OSC. When ARM Cortex-M4 and all its peripheral relinquish the high-power resources, ARM Cortex-M4 runs a part of code in TCM and lets ARM Cortex-A7 shut down these resources safely. When both ARM Cortex-M4 and ARM Cortex-A7 come into the STOP mode. The whole system enters a deep sleep mode (DSM). In DSM mode, ARM Cortex-M4 platform including the TCM are totally powered off. After exiting from DSM, ARM Cortex-A7 cold boots ARM Cortex-M4 again.

8.2 Supported platforms

8.2.1 i.MX 7Dual SABRE board

8.2.1.1 Hardware requirements

- SD Card with Linux image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD boards
- Personal computer with USB port

8.2.1.2 Toolchain requirements

One of the following toolchains is required:

Supported platforms

- IAR Embedded Workbench[®]
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

8.2.1.3 Software requirements

- The random WFI project files are in: `<BSP_Install>/examples/imx7d_sdb_m4/demo_apps/lowpower_imx7d/rand_wfi/<toolchain>`.
- The default linker script is for TCM.

8.2.1.4 Getting started

8.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD 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 ARM Cortex-M4 core and ARM Cortex-A7 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

8.2.1.4.2 Running the demo

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

```
Low Power Demo

Please wait :
  1) A7 peer is ready
Then press "S" to start the demo

Press "S" to start the demo :
```

After ARM Cortex-A7 core boots up, press 's' in the terminal. ARM Cortex-M4 core begins to switch between RUN/WAIT/STOP with a random time interval between 5 seconds to 10 seconds. The random time interval can be adjusted by changing the 2 macro "PERIOD_MIN" and "PERIOD_MAX" defined in

<BSP_Install>/examples/src/demo_apps/lowpower_imx7d/rand_wfi/main.c. The CPU clock switches to OSC when in WAIT / STOP mode and switches to PLL2 when in RUN mode. The terminal displays the following information.

```
GPT will trigger interrupt in 16s
go to mode WAIT
GPT Event! Total time 16s
GPT will trigger interrupt in 16s
go to mode STOP
GPT Event! Total time 32s
GPT will trigger interrupt in 18s
go to mode RUN
GPT Event! Total time 50s
GPT will trigger interrupt in 8s
...

```

If the LPM_MCORE_PRINT_DEBUG_INFO macro in lpm_mcore.h is enabled, the terminal will also display the debug information of the low-power demo.

After ARM Cortex-M4 switch to OSC and get into WAIT/STOP mode, ARM Cortex-A7 side can shut down high-bus resources. Running bus freq switch test script in ARM Cortex-A7, related information can be seen.

After ARM Cortex-M4 enters STOP mode, if ARM Cortex-A7 also enters STOP mode, then the whole system enters DSM. After exiting from DSM, ARM Cortex-M4 core is cold booted. Running MEM suspend script on the ARM Cortex-A7 core, related information can be seen.

```
Change M4 clock freq to 24M
Verify M4 Speed : 10 of 10 ... Done.
GPT will trigger interrupt in 16s
go to mode WAIT
GPT Event! Total time 16s
Verify M4 Speed : 10 of 10 ... Done.
GPT will trigger interrupt in 16s
go to mode STOP
GPT Event! Total time 32s
Change M4 clock freq to SysPLL Div2 (240M)
Verify M4 Speed : 10 of 10 ... Done.
GPT will trigger interrupt in 18s
go to mode RUN
GPT Event! Total time 50s
Change M4 clock freq to 24M
Verify M4 Speed : 10 of 10 ... Done.
GPT will trigger interrupt in 8s
...

```

Whenever ARM Cortex-M4 wakes up from WAIT or STOP, it uses MU to inform ARM Cortex-A7 to prepare the high-bus resources. This wakes up ARM Cortex-A7 as a consequence.



Supported platforms

Chapter 9

RPMMsg Ping-Pong Bare Metal Demo

9.1 Overview

This demo application demonstrates the RPMMsg remote peer stack working on bare metal with OpenAMP RPMMsg API. It works with Linux RPMMsg 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 ARM Cortex-M4 bare metal application. The receiving peer adds 1 to the integer and transfers it back. The loop continues infinitely.

9.2 Supported platforms

9.2.1 i.MX 7Dual SABRE board

9.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

9.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

9.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/rpmsg/pingpong_bm/<toolchain>.
- Linux RPMMsg master side ping-pong module

Supported platforms

9.2.1.4 Getting started

9.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open two serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, and install the pingpong master side module.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

9.2.1.4.2 Running the demo

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

```
RPMSG PingPong Bare Metal Demo...
RPMSG Init as Remote
```

After the Linux ping-pong master side module is installed, 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 Master Side : 0
Get Data From Master Side : 2
Get Data From Master Side : 4
Get Data From Master Side : 6
Get Data From Master Side : 8
.....
```

As is shown on the log, the RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform a name service handshake to create the communication channel. The ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core begins to send the first data to ARM Cortex-M4 core. After receiving the data, ARM Cortex-M4 core adds 1 to it and sends it back to ARM Cortex-A7 core. ARM Cortex-A7 core responds with the same behaviour.

9.2.2 i.MX 7Dual Validation board

9.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

9.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

9.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/rpmsg/pingpong_bm/<toolchain>.
- Linux RPMsg master side ping-pong module

9.2.2.4 Getting started

9.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open two serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, and install the ping-pong master side module.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

Supported platforms

9.2.2.4.2 Running the demo

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

```
RPMSG PingPong Bare Metal Demo...
RPMSG Init as Remote
```

After the Linux ping-pong master side module is installed, 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 Master Side : 0
Get Data From Master Side : 2
Get Data From Master Side : 4
Get Data From Master Side : 6
Get Data From Master Side : 8
.....
```

As is shown on the log, the RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform a name service handshake to create the communication channel. The ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core begins to send the first data to ARM Cortex-M4 core. After receiving the data, ARM Cortex-M4 core adds 1 to it and sends it back to ARM Cortex-A7 core. ARM Cortex-A7 core responds with the same behaviour.

Chapter 10

RPMMsg Ping-Pong FreeRTOS Demo with RTOS API

10.1 Overview

This demo application demonstrates the RPMMsg remote peer stack working on FreeRTOS OS with RPMMsg RTOS API extension. It works with Linux RPMMsg 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 infinitely.

The RPMMsg RTOS API extension is a set of easy to use APIs that support multitask receive/send messages. If the raw RPMMsg API is preferred, it is easy to integrate with FreeRTOS OS by referring to the RPMMsg Ping-Pong Bare Metal Demo.

10.2 Supported platforms

10.2.1 i.MX 7Dual SABRE board

10.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

10.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

10.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/rpmsg/pingpong_freertos/<toolchain>.

Supported platforms

- Linux RPMsg master side ping-pong module

10.2.1.4 Getting started

10.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open two serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, and install the ping-pong master side module.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

10.2.1.4.2 Running the demo

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

```
RPMMSG PingPong FreeRTOS RTOS API Demo...
RPMMSG Init as Remote
```

After the Linux ping-pong master side module is installed, 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 Master Side : 0
Get Data From Master Side : 2
Get Data From Master Side : 4
Get Data From Master Side : 6
Get Data From Master Side : 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 ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core begins to send the first data to ARM Cortex-M4 core. After receiving the data, ARM Cortex-M4 core adds 1 to it and sends it back to ARM Cortex-A7 core. ARM Cortex-A7 core responds with the same behaviour.

10.2.2 i.MX 7Dual Validation board

10.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

10.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

10.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/rpmsg/pingpong_freertos/<toolchain>.
- Linux RPMsg master side ping-pong module

10.2.2.4 Getting started

10.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open two serial terminals on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, and install the ping-pong master side module.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

Supported platforms

10.2.2.4.2 Running the demo

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

```
RPMSG PingPong FreeRTOS RTOS API Demo...
RPMSG Init as Remote
```

After the Linux ping-pong master side module is installed, 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 Master Side : 0
Get Data From Master Side : 2
Get Data From Master Side : 4
Get Data From Master Side : 6
Get Data From Master Side : 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 ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core begins to send the first data to ARM Cortex-M4 core. After receiving the data, ARM Cortex-M4 core adds 1 to it and sends it back to ARM Cortex-A7 core. ARM Cortex-A7 core responds with the same behaviour.

Chapter 11

RPMsg String Echo Bare Metal Demo

11.1 Overview

This demo application demonstrates the RPMsg extension API working on Bare Metal application. 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 received is sent to ARM Cortex-M4 core. ARM Cortex-M4 core displays what is received, and echoes back the same message as an acknowledgement. The tty reader on ARM Cortex-A7 core can get the message, and start another transaction. The demo demonstrates RPMsg's ability to send arbitrary content back and forth.

The RPMsg extension API is introduced in `rpmmsg_ext.h` to achieve zero-copy receive/send operation which can improve performance significantly in busy transaction.

Note: The maximum message length supported by RPMsg is now 496 bytes. String longer than 496 is divided by virtual tty into several messages.

11.2 Supported platforms

11.2.1 i.MX 7Dual SABRE board

11.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

11.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

Supported platforms

11.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/rpmsg/str_echo_bm/<toolchain>.
- Linux RPMMsg master side RPMMsg tty module and application

11.2.1.4 Getting started

11.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open TWO serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, install the RPMMsg tty module.
6. Run RPMMsg tty receive program `"/unit_tests/mxc_mcc_tty_test.out /dev/ttyRPMMSG 115200 R 100 1000 &"`

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

11.2.1.4.2 Running the demo

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

```
RPMMSG String Echo Bare Metal Demo...
RPMMSG Init as Remote
```

After the Linux RPMMsg tty module is installed, the ARM Cortex-M4 terminal displays the following information:

```
Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]
```

The user can then input an arbitrary string to the virtual RPMMsg tty using the following echo command on Cortex-A7 terminal:

```
echo test > /dev/ttyRPMMSG
```

```
echo deadbeaf > /dev/ttyRPMMSG
```

```
...
```

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

```
Get Message From Master Side : "test
                               " [len : 5] From Slot 0
Get Message From Master Side : "deadbeaf
                               " [len : 9] From Slot 1
...
```

The RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform name service handshake to create the communication channel. The ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core then waits for user input to RPMsg virtual tty and sends the content to ARM Cortex-M4 core. On receiving the data, ARM Cortex-M4 core outputs the content and its length on the terminal and echoes back the same message to ARM Cortex-A7 core. If some application is reading from /dev/ttyRPMMSG on ARM Cortex-A7 core, it could get the echo message. 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.

11.2.2 i.MX 7Dual Validation board

11.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation main board
- Personal computer with USB port

11.2.2.2 Toolchain requirements

One of following toolchains is required:

- IAR Embedded Workbench

Supported platforms

- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

11.2.2.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_val_m4/demo_apps/rpmsg/str_echo_-bm/<toolchain>`.
- Linux RPMsg master side RPMsg tty module and application

11.2.2.4 Getting started

11.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open TWO serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, and install the RPMsg tty module.
6. Run `rpmsg tty receive program "/unit_tests/mxc_mcc_tty_test.out /dev/ttyRPMMSG 115200 R 100 1000 &"`

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRT-OS7DGSUG).

11.2.2.4.2 Running the demo

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

```
RPMMSG String Echo Bare Metal Demo...
RPMMSG Init as Remote
```

After the Linux RPMsg tty module is installed, the ARM Cortex-M4 terminal displays the following information:

Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]

The user can then input an arbitrary string to the virtual RPMsg tty using the following echo command on Cortex-A7 terminal:

```
echo test > /dev/ttyRPMSG
```

```
echo deadbeaf > /dev/ttyRPMSG
```

...

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

```
Get Message From Master Side : "test
                               " [len : 5] From Slot 0
Get Message From Master Side : "deadbeaf
                               " [len : 9] From Slot 1
...
```

The RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform name service handshake to create the communication channel. The ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core then waits for user input to RPMsg virtual tty and sends the content to ARM Cortex-M4 core. On receiving the data, ARM Cortex-M4 core outputs the content and its length on the terminal and echoes back the same message to ARM Cortex-A7 core. If some application is reading from /dev/ttyRPMSG on ARM Cortex-A7 core, it could get the echo message. 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.



Supported platforms

Chapter 12

RPMsg String Echo FreeRTOS Demo with RTOS API

12.1 Overview

This demo application demonstrates the RPMsg remote peer stack working on FreeRTOS OS with RPMsg RTOS API extension. 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 ARM Cortex-M4 core. ARM Cortex-M4 core displays what is received, and echoes back the same message as an acknowledgement. The tty reader on ARM Cortex-A7 core can get the message, and start another transaction. The demo demonstrates RPMsg's ability to send arbitrary content back and forth.

The RPMsg RTOS API extension is a set of easy to use APIs that support multitask receive/send messages. In this demo, zero-copy operation in RTOS API is leveraged to improve transaction performance.

If the raw RPMsg API is preferred, it is easy to integrate with FreeRTOS OS by referring to the RPMsg String Echo Bare Metal Demo.

Note: The maximum message length supported by RPMsg is now 496 bytes. String longer than 496 is divided by the virtual tty into several messages.

12.2 Supported platforms

12.2.1 i.MX 7Dual SABRE board

12.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal Computer with USB port

12.2.1.2 Toolchain requirements

One of following toolchains is required:

- IAR Embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain versions, see the *FreeRTOS BSP v.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

Supported platforms

12.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/rpmsg/str_echo_freertos/<toolchain>.
- Linux RPMMsg master side RPMMsg tty module and application

12.2.1.4 Getting started

12.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open TWO serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, install the RPMMsg tty module.
6. Run RPMMsg tty receive program `"/unit_tests/mxc_mcc_tty_test.out /dev/ttyRPMMSG 115200 R 100 1000 &"`

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

12.2.1.4.2 Running the demo

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

```
RPMMSG String Echo FreeRTOS RTOS API Demo...
RPMMSG Init as Remote
```

After the Linux RPMMsg tty module was installed, the ARM Cortex-M4 terminal displays the following information:

```
Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]
```

The user can then input an arbitrary string to the virtual RPMMsg tty using the following echo command on Cortex-A7 terminal:

```
echo test > /dev/ttyRPMMSG
```



```
echo deadbeaf > /dev/ttyRPMSG
```

```
...
```

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

```
Get Message From Master Side : "test
                               " [len : 5]
Get Message From Master Side : "deadbeaf
                               " [len : 9]
...
```

The RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform name service handshake to create the communication channel. The ARM Cortex-M4 channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core then waits for user input to RPMsg virtual tty and sends the content to ARM Cortex-M4 core. On receiving the data, ARM Cortex-M4 core outputs the content and its length on the terminal and echoes back the same message to ARM Cortex-A7 core. If some application is reading from /dev/ttyRPMSG on ARM Cortex-A7 core, it could get the echo message. The loop continues to demonstrate RPMsg's ability to send arbitrary content.

The RTOS API implementation leverages queue to store received messages so that all the messages from ARM Cortex-A7 core could be well buffered for future receiving.

12.2.2 i.MX 7Dual Validation board

12.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

12.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

Supported platforms

12.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/rpmsg/str_echo_freertos/<toolchain>.
- Linux RPMsg master side RPMsg tty module and application

12.2.2.4 Getting started

12.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open TWO serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.
5. Boot the Linux OS kernel, install the RPMsg tty module.
6. Run RPMsg tty receive program `"/unit_tests/mxc_mcc_tty_test.out /dev/ttyRPMSG 115200 R 100 1000 &"`

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

12.2.2.4.2 Running the demo

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

```
RPMSG String Echo FreeRTOS RTOS API Demo...
RPMSG Init as Remote
```

After the Linux RPMsg tty module is installed, the ARM Cortex-M4 terminal displays the following information:

```
Name service handshake is done, M4 has setup a rpmsg channel [1 ---> 1024]
```

The user can then input an arbitrary string to the virtual RPMsg tty using the following echo command on ARM Cortex-A7 terminal:

```
echo test > /dev/ttyRPMSG
```

```
echo deadbeaf > /dev/ttyRPMMSG
```

...

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

```
Get Message From Master Side : "test
                               " [len : 5]
Get Message From Master Side : "deadbeaf
                               " [len : 9]
...
```

The RPMsg Master (Cortex-A7 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform name service handshake to create the communication channel. The ARM Cortex-M4 core channel address is 1, and the ARM Cortex-A7 channel address is 1024. ARM Cortex-A7 core then waits for user input to RPMsg virtual tty and sends the content to ARM Cortex-M4 core. On receiving the data, ARM Cortex-M4 core outputs the content and its length on the terminal and echoes back the same message to ARM Cortex-A7 core. If some application is reading from /dev/ttyRPMMSG on ARM Cortex-A7 core, it could get the echo message. The loop continues to demonstrate RPMsg's ability to send arbitrary content.

The RTOS API implementation leverages a queue to store received messages so that all the messages from ARM Cortex-A7 core could be well buffered for future receiving.



Supported platforms

Chapter 13

SEMA4 Mutex Demo

13.1 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 an "unlocked" 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.

13.2 Supported platforms

13.2.1 i.MX 7Dual SABRE board

13.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

13.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

13.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/sema4_demo/<toolchain>.

Supported platforms

13.2.1.4 Getting started

13.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

13.2.1.4.2 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
```

13.2.2 i.MX 7Dual Validation board

13.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

13.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

13.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/demo_apps/sema4_demo/<toolchain>.

Supported platforms

13.2.2.4 Getting started

13.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

13.2.2.4.2 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
```



Supported platforms

Chapter 14 Sensor Demo

14.1 Overview

The Sensor Demo i.MX 7Dual is a simple demonstration program that uses the FreeRTOS OS and a set of drivers provided by NXP. 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.

NOTE: The I2C2 instance on i.MX 7Dual SDB board is assigned to ARM Cortex-M4 core when this demo starts. Do not use this I2C instance at ARM Cortex-A7 core side when this demo is running.

14.2 Supported platforms

14.2.1 i.MX 7Dual SABRE board

14.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

14.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

14.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/demo_apps/sensor_demo/<toolchain>.

Supported platforms

14.2.1.4 Getting started

14.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port(J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

14.2.1.4.2 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 Polling Demo
[2].FXAS21002 3-axes Gyro sensor Interrupt Demo
[3].FXOS8700 6-axes Acc+Mag sensor Polling Demo
[4].FXOS8700 6-axes Acc+Mag sensor Interrupt Demo
[5].MPL3115 Pressure sensor Polling Demo
[6].MPL3115 Pressure sensor Interrupt Demo
```

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 500 ms, 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 500 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
```



Supported platforms

Chapter 15

ADC Example

15.1 Overview

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

15.2 Supported platforms

15.2.1 i.MX 7Dual SABRE board

15.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

15.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

15.2.1.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_sdb_m4/driver_examples/adc_imx7d/<toolchain>`.

Supported platforms

15.2.1.4 Getting started

15.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Connect ADC1_IN3 (J22-PIN8) with external input through dupont line with cable.
3. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
4. Load the demo binary to the TCM using U-Boot.
5. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

15.2.1.4.2 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
```

15.2.2 i.MX 7Dual Validation board

15.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation main board
- Personal computer with USB port

15.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

15.2.2.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_val_m4/driver_examples/adc_imx7d/<toolchain>`.

15.2.2.4 Getting started

15.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Connect ADC1_IN3(JP8-PIN5) with external input through dupont line with cable.
3. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
4. Load the demo binary to the TCM using U-Boot.
5. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRT-OS7DGSUG).

15.2.2.4.2 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 7Dual processor.
It Continuous convert Analog Input, and print the result to terminal
Current analog value: 1.06v
Current analog value: 1.04v
```



Supported platforms

Chapter 16

eCSPI Interrupt Example

16.1 Overview

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

NOTE: Do not use this example and Linux OS on ARM Cortex-A7 core at the same time, because the IO Pins for eCSPI2 on i.MX 7Dual Validation board is occupied by Linux OS for EPDC display.

16.2 Supported platforms

16.2.1 i.MX 7Dual Validation board

16.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5 V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port
- 4-pin metal-shielded wire

16.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

16.2.1.3 Software requirements

- The master project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/ecspi/ecspi-_interrupt/master/<toolchain>.
- The slave project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/ecspi/ecspi-_interrupt/slave/<toolchain>.

Supported platforms

16.2.1.4 Getting started

16.2.1.4.1 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.

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.

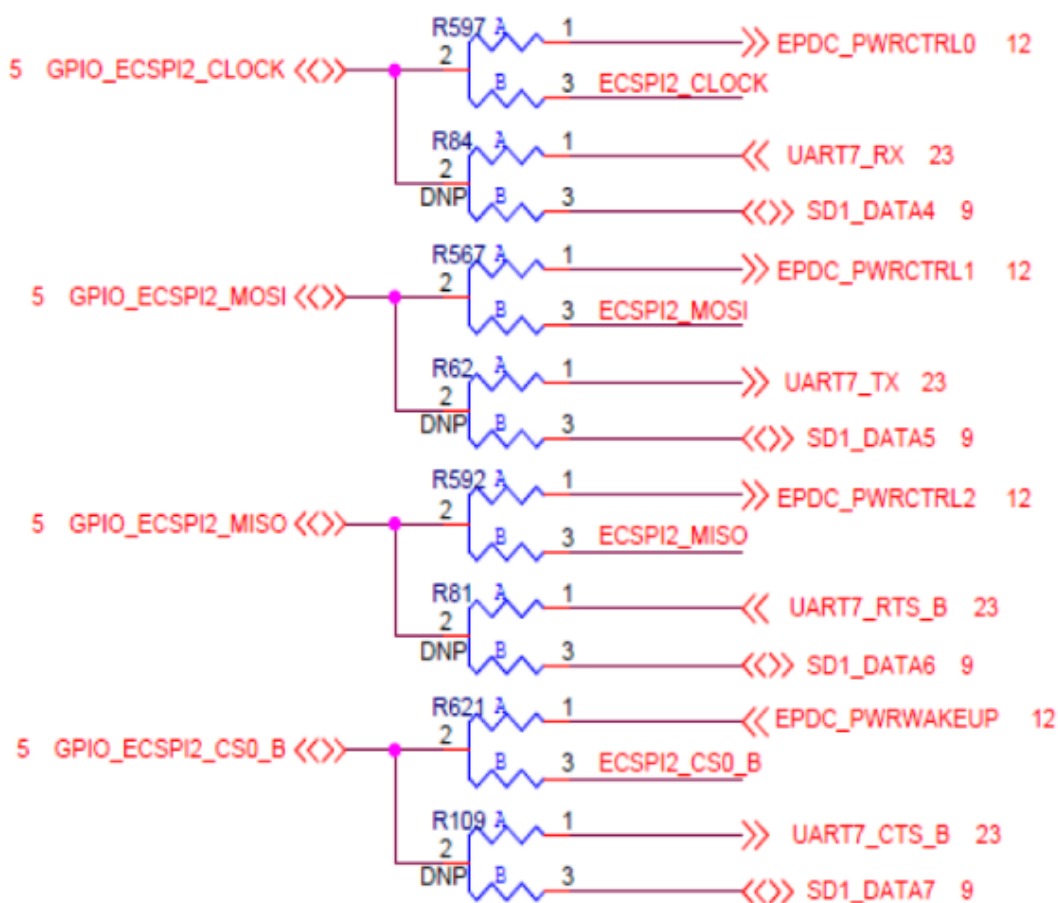


Figure 16.2.1: eCSPI rework guidance schematic diagram for i.MX 7Dual Validation board

This example requires two separate boards. You can connect eCSPI2 signals on jumper "J8".

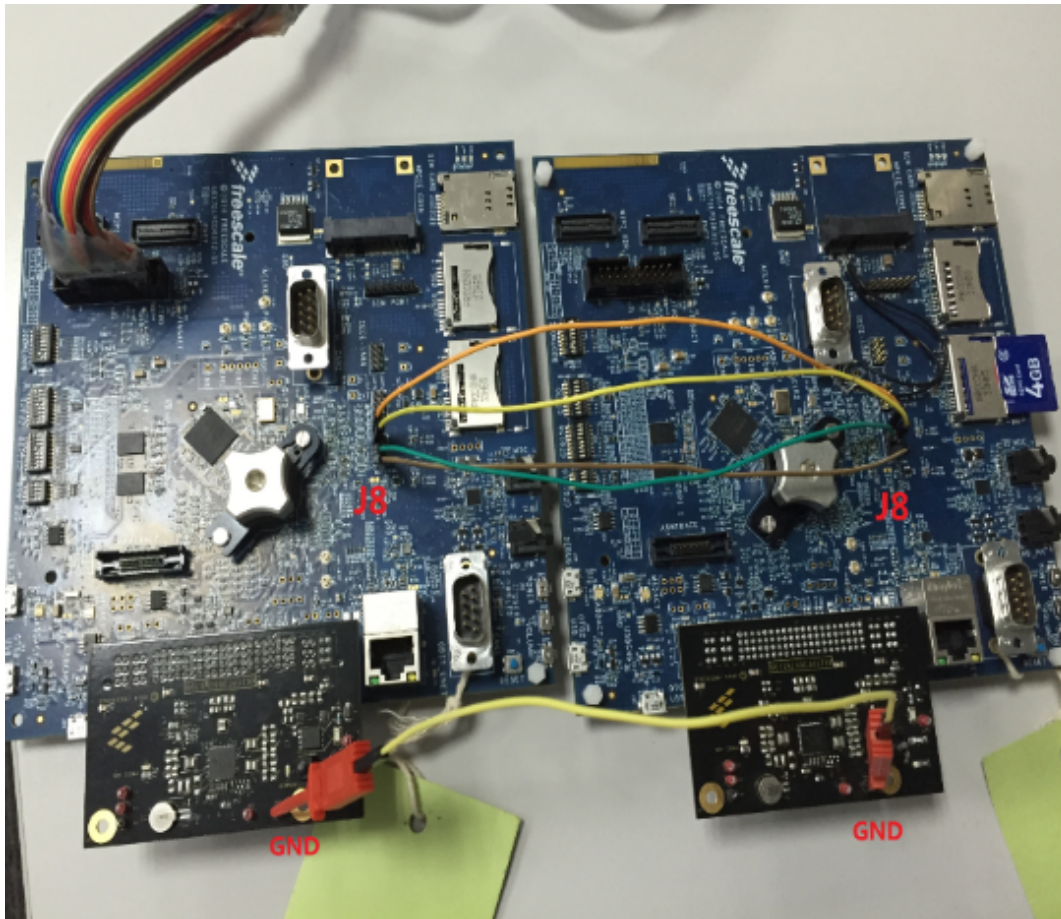


Figure 16.2.2: eCSPI Interrupt example board connection

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

Figure 16.2.3: eCSPI pin assignment table

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

Supported platforms

16.2.1.4.2 Preparing the demo

1. Connect two micro USB cable for each computer host and Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open two serial terminals on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Connect these 2 board to eCSPI4.
4. Load the demo binary to the TCM using U-Boot.
5. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

16.2.1.4.3 Running the demo

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 prints on terminal:

```
MASTER: Transmitted data: 1  
       : Received data: 255  
  
MASTER: Transmitted data: 2  
       : Received data: 0  
  
MASTER: Transmitted data: 3  
       : Received data: 1  
  
...  
  
MASTER: Transmitted data: 20  
       : Received data: 18
```

The slave board prints 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
```



Supported platforms

Chapter 17

eCSPI Polling Example

17.1 Overview

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

NOTE: Do not use this example and Linux OS on Cortex-A7 Core at the same time, because the IO Pins for eCSPI2 on i.MX 7Dual Validation board are occupied by Linux OS for EPDC display.

17.2 Supported platforms

17.2.1 i.MX 7Dual Validation board

17.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5 V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port
- 4-pin metal-shielded wire

17.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

17.2.1.3 Software requirements

- The master project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/ecspi/ecspi-_polling/master/<toolchain>.
- The slave project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/ecspi/ecspi-_polling/slave/<toolchain>.

Supported platforms

17.2.1.4 Getting started

17.2.1.4.1 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.

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.

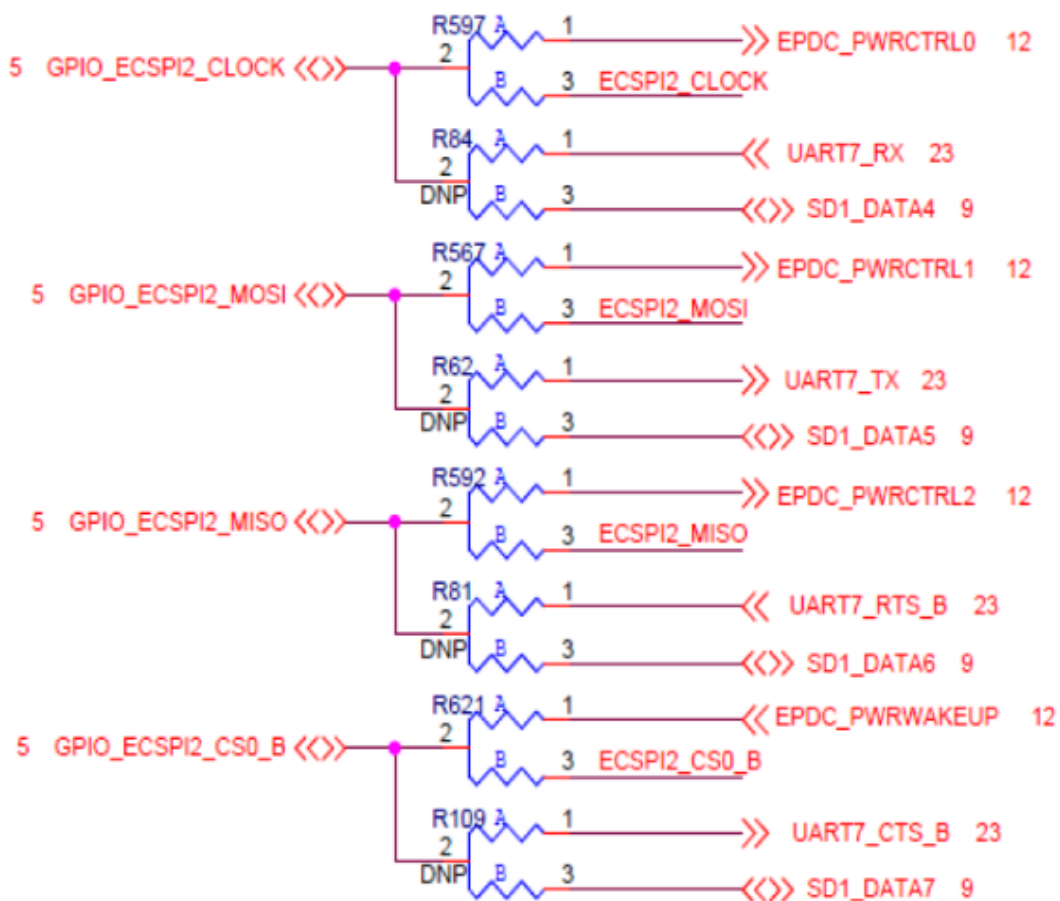


Figure 17.2.1: eCSPI rework guidance schematic diagram for i.MX 7Dual Validation board

This example requires two separate boards. You can connect ECSPi2 signals on jumper "J8".

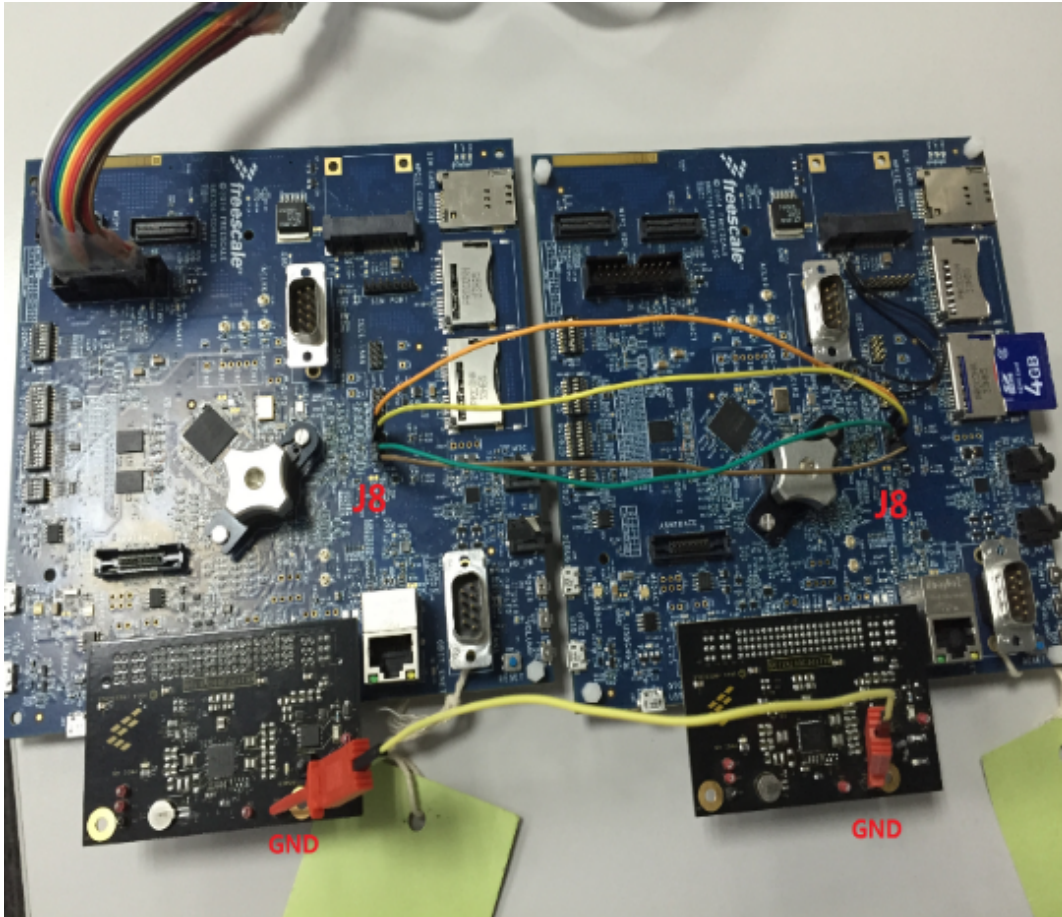


Figure 17.2.2: eCSPI Polling example board connection

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

Figure 17.2.3: eCSPI pin assignment table

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

Supported platforms

17.2.1.4.2 Preparing the demo

1. Connect two micro USB cable for each computer host and Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open two serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Connect these 2 board to ECSPI4.
4. Load the demo binary to the TCM using U-Boot.
5. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

17.2.1.4.3 Running the demo

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 prints on terminal:

```
MASTER: Transmitted data: 1  
       : Received data: 255  
  
MASTER: Transmitted data: 2  
       : Received data: 0  
  
MASTER: Transmitted data: 3  
       : Received data: 1  
  
...  
  
MASTER: Transmitted data: 20  
       : Received data: 18  
  
Example finish!!!
```

The slave board prints 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
```

```
Example finish!!!
```



Supported platforms

Chapter 18

FlexCAN Loopback Example

18.1 Overview

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

This example use two message buffers: one is for transmitting data, and the other is for receiving 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.

18.2 Supported platforms

18.2.1 i.MX 7Dual SABRE board

18.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

18.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

18.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/flexcan/flexcan-_loopback/<toolchain>.

Supported platforms

18.2.1.4 Getting started

18.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

18.2.1.4.2 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 is sent through transmit MB and received from its own receive MB every 1 second. At the beginning the received data is 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
```

18.2.2 i.MX 7Dual Validation board

18.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 12V DC adapter
- i.MX 7Dual Validation main board
- Personal computer with USB port

18.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

18.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/flexcan/flexcan-_loopback/<toolchain>.

Supported platforms

18.2.2.4 Getting started

18.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

18.2.2.4.2 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 is sent through transmit MB and received from its own receive MB every 1 second. At the beginning the received data is 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 is 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



Supported platforms

Chapter 19

FlexCAN Network Example

19.1 Overview

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

This example uses 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.

19.2 Supported platforms

19.2.1 i.MX 7Dual SABRE board

19.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

19.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

19.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/flexcan/flexcan-network/<toolchain>.

Supported platforms

19.2.1.4 Getting started

19.2.1.4.1 Hardware settings

To run this example, connect two boards through the CAN interface: The CAN1 DB-9 Connector on the i.MX 7Dual SABRE-SD board is used as the CAN interface, PIN2 connects to CANL and PIN7 connects to CANH:



Figure 19.2.1: CAN bus interface DB-9 connector usage for i.MX 7Dual SDB board

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

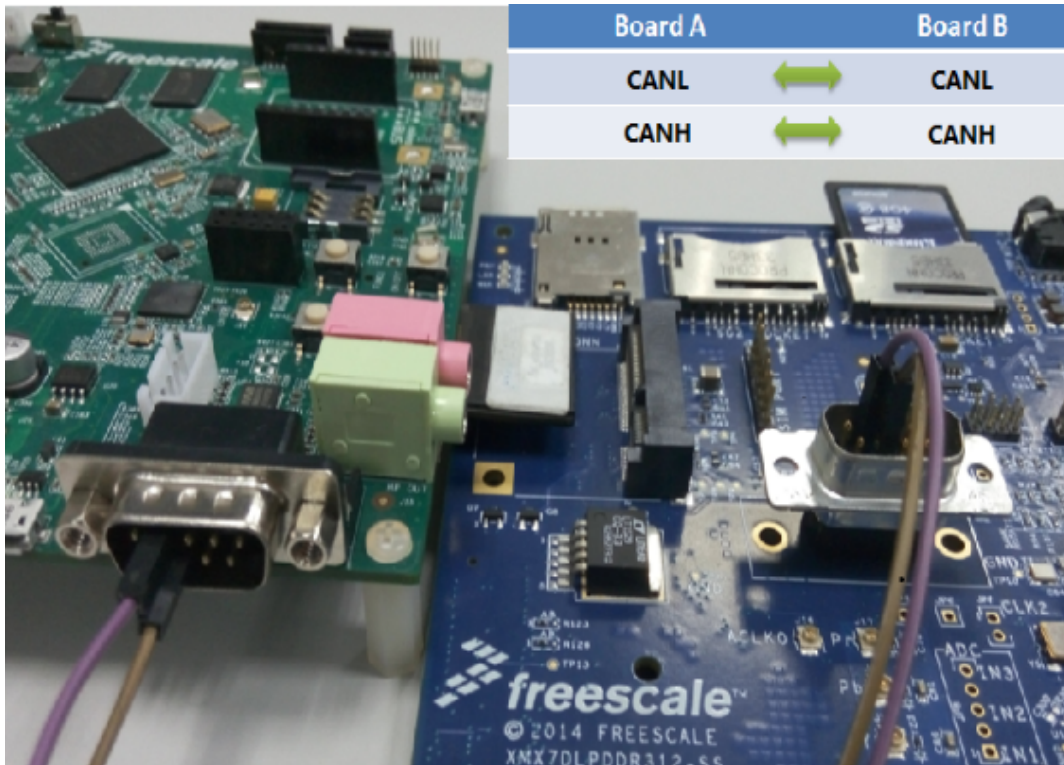


Figure 19.2.2: FlexCAN network example board connection

19.2.1.4.2 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 configurations for these two boards.
3. Connect two micro USB cable for each PC host and Debug UART port(J11) on the i.MX 7Dual SABRE-SD board.
4. Open two serial terminals for these two boards with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
5. Connect these two boards to the CAN Bus.
6. Load the demo binary to the TCM using U-Boot.
7. Boot auxiliary Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

Supported platforms

19.2.1.4.3 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 is 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
```


19.2.2 i.MX 7Dual Validation board

19.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 12V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

19.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

19.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/flexcan/flexcan-network/<toolchain>.

19.2.2.4 Getting started

19.2.2.4.1 Hardware settings

To run this example, connect two boards through the CAN interface: The CAN1 DB-9 Connector on the i.MX 7Dual Validation board is used as the CAN interface, PIN2 connects to CANL and PIN7 connects to CANH:

Supported platforms

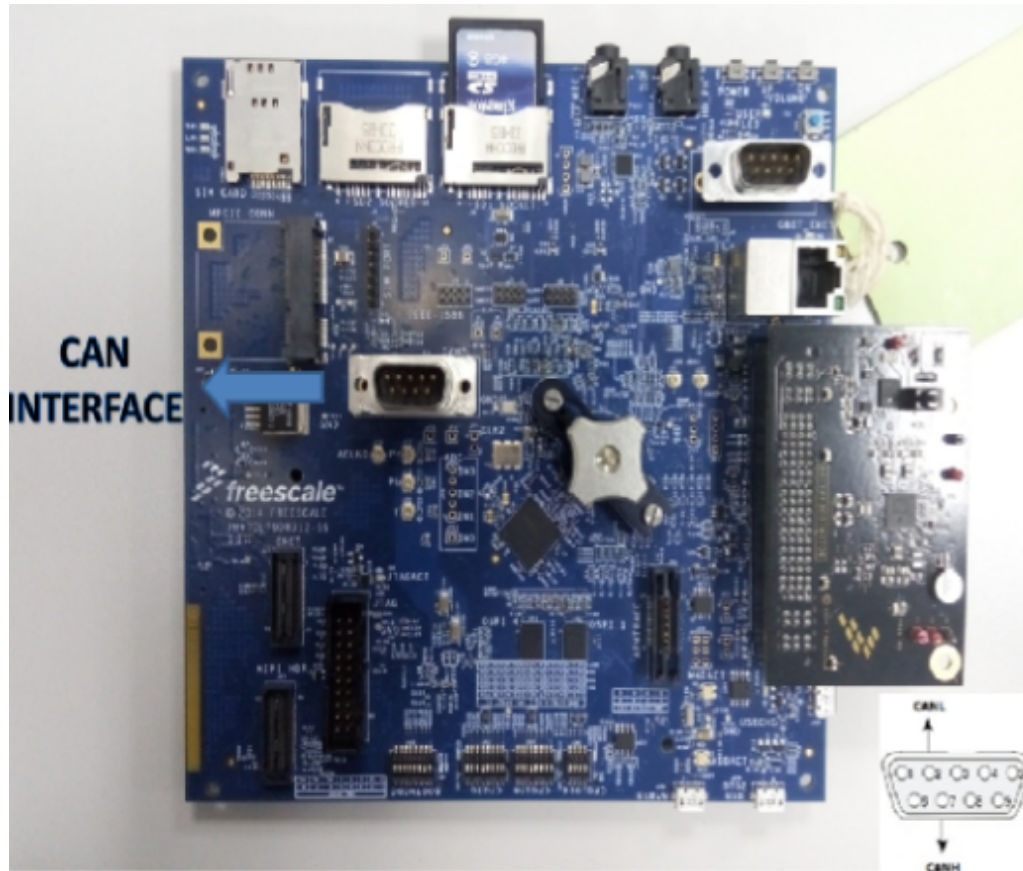


Figure 19.2.3: FlexCAN interface DB-9 connector usage for i.MX 7Dual Validation board

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

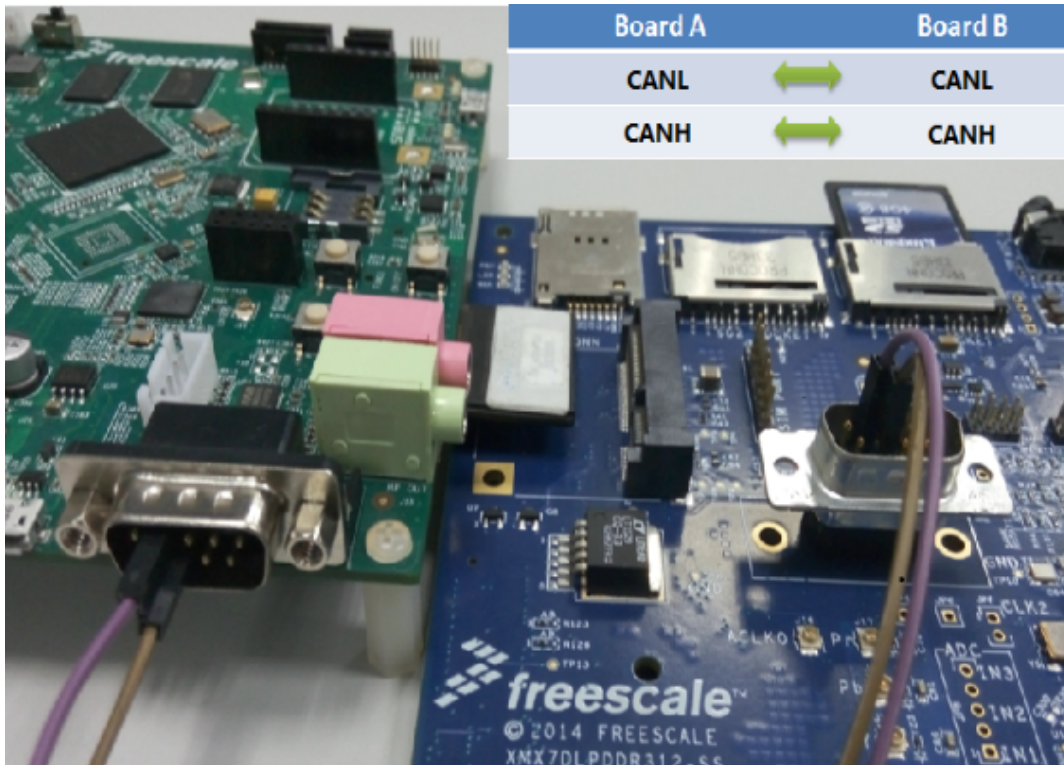


Figure 19.2.4: FlexCAN Network example board connection

19.2.2.4.2 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 two board.
3. Connect two micro USB cable for each PC host and Debug UART port (J28) on the i.MX 7Dual Validation board.
4. Open two serial terminal for these two boards with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
5. Connect these two boards to CAN Bus.
6. Load the demo binary to the TCM using U-Boot.
7. Boot auxiliary Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

Supported platforms

19.2.2.4.3 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 is 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 20

GPIO Example

20.1 Overview

This example application demonstrates how to use the GPIO driver to access LEDs or Buttons on the board. On the i.MX 7Dual SABRE-SD board, the application support the GPIO key function. And on the i.MX 7Dual Validation board, the application switches the LED on board when user press GPIO key.

NOTE: Sharing of GPIO need deep customization in Linux OS kernel, so it is not recommended to run this example with default Linux OS kernel together.

20.2 Supported platforms

20.2.1 i.MX 7Dual SABRE board

20.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

20.2.1.2 Toolchain requirements

One of the following toolchains is required:

- IAR Embedded Workbench
- ARM GCC
- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

20.2.1.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_sdb_m4/driver_examples/gpio_imx/<toolchain>`.

Supported platforms

20.2.1.4 Getting started

20.2.1.4.1 Preparing the demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

20.2.1.4.2 Running the demo

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

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

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 button state is now polled.  
Press the button to switch LED on or off
```

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

```
+ - + - + - + - + - + - + - + - + -
```

20.2.2 i.MX 7Dual Validation board

20.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual

- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

20.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

20.2.2.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_val_m4/driver_examples/gpio_imx/<toolchain>`.

20.2.2.4 Getting started

20.2.2.4.1 Preparing the demo

1. Connect a micro USB cable between the PC host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

20.2.2.4.2 Running the demo

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

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

Supported platforms

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

Press the specific button 3 times. The board prints following information on terminal:

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

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

Input the any data from terminal. The board prints on terminal:

```
Button pressed 1 time.  
Button pressed 2 time.  
Button pressed 3 time.  
Button pressed 4 time.  
Button pressed 5 time.  
Button pressed 6 time.
```

It switches the LED each time you input any data from terminal.

Chapter 21

GPT Example

21.1 Overview

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

This example uses different clock sources for 2 GPT instances and capture the counter of each clock source every 0.5 seconds. If they both work correctly, the captured counter should be close to half of the GPT frequency. The clock source frequency is not 100% accurate, and the divider could also affect the clock error so that the captured number would be different from the expected value.

21.2 Supported platforms

21.2.1 i.MX 7Dual SABRE board

21.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

21.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

21.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/gpt/<toolchain>.

Supported platforms

21.2.1.4 Getting started

21.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

21.2.1.4.2 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...
```

21.2.2 i.MX 7Dual Validation board

21.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation main board
- Personal Computer with USB port

21.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

21.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/gpt/<toolchain>.

21.2.3.1 Getting started

21.2.3.1.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

21.2.3.1.2 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.
```

Supported platforms

```
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 22

I2C Interrupt EEPROM Example

22.1 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.

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

22.2 Supported platforms

22.2.1 i.MX 7Dual Validation board

22.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

22.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

22.2.2.1 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/i2c_imx/i2c_interrupt_eeprom/<toolchain>.

Supported platforms

22.2.2.2 Getting started

22.2.2.2.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

22.2.2.2.2 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!!!
```

Chapter 23

I2C Interrupt Sensor Example

23.1 Overview

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

Programming on FXOS8700 acceleration sensor through I2C bus and read the 3-Axis acceleration of gravity.

23.2 Supported platforms

23.2.1 i.MX 7Dual SABRE board

23.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal Computer with USB port

23.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

23.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/i2c_imx/i2c_interrupt_sensor_imx7d/<toolchain>.

Supported platforms

23.2.1.4 Getting started

23.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

23.2.1.4.2 Running the demo

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

```
+++++ I2C Send/Receive interrupt Example +++++
This example will configure on board accelerometer through I2C Bus
and read 10 samples back to see if the accelerometer is configured successfully.
```

After printing the information mentioned above, the example reads acceleration sensor's data and print it to terminal like this:

```
[1].Initialize the I2C module with initialize structure.
[2].Set on-board Acc sensor range to 2G
[3].Set on-board Acc sensor working at fast read and active mode
[4].Acc sensor WHO_AM_I check... OK
[5].Acquire 10 samples from Acc sensor
2G MODE: X= 0.058g Y= 0.039g Z= 0.977g
2G MODE: X= 0.058g Y= 0.023g Z= 1.004g
2G MODE: X= 0.056g Y= 0.031g Z= 0.995g
2G MODE: X= 0.063g Y= 0.027g Z= 0.980g
2G MODE: X= 0.057g Y= 0.036g Z= 0.979g
2G MODE: X= 0.054g Y= 0.032g Z= 0.981g
2G MODE: X= 0.058g Y= 0.035g Z= 0.985g
2G MODE: X= 0.055g Y= 0.034g Z= 0.978g
2G MODE: X= 0.057g Y= 0.028g Z= 0.999g
2G MODE: X= 0.059g Y= 0.038g Z= 0.972g
```

```
Example finished!!!
```


Chapter 24

I2C Polling EEPROM Example

24.1 Overview

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

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

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

24.2 Supported platforms

24.2.1 i.MX 7Dual Validation board

24.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

24.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

24.2.2.1 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/i2c_imx/i2c_-polling_eeprom/<toolchain>.

Supported platforms

24.2.2.2 Getting started

24.2.2.2.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

24.2.2.2.2 Running the demo

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

```
+++++ I2C Send/Receive polling 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 and wait until transmission finished.
[5].Launch a I2C read action from 0x0000 address.
[6].Read data from EEPROM and wait until transmission finished.
[7].Compare data between txBuf and rxBuf:
    txBuf and rxBuf are same, example passed!!!
```

Chapter 25

I2C Polling Sensor Example

25.1 Overview

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

Programming on FXOS8700 acceleration sensor through I2C bus and read the 3-Axis acceleration of gravity.

25.2 Supported platforms

25.2.1 i.MX 7Dual SABRE board

25.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

25.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

25.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/i2c_imx/i2c_polling_sensor_imx7d/<toolchain>.

Supported platforms

25.2.1.4 Getting started

25.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

25.2.1.4.2 Running the demo

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

```
+++++ I2C Send/Receive polling Example ++++++
This example will configure on board accelerometer through I2C Bus
and read 10 samples back to see if the accelerometer is configured successfully.
```

After printing the information mentioned above, the example reads acceleration sensor's data and print it to terminal like this:

```
[1].Initialize the I2C module with initialize structure.
[2].Set on-board Acc sensor range to 2G
[3].Set on-board Acc sensor working at active mode
[4].Acc sensor WHO_AM_I check... OK
[5].Acquire 10 samples from Acc sensor
2G MODE: X= 0.060g Y= 0.022g Z= 1.010g
2G MODE: X= 0.058g Y= 0.036g Z= 0.977g
2G MODE: X= 0.062g Y= 0.024g Z= 0.998g
2G MODE: X= 0.058g Y= 0.038g Z= 0.970g
2G MODE: X= 0.059g Y= 0.024g Z= 1.018g
2G MODE: X= 0.058g Y= 0.036g Z= 0.987g
2G MODE: X= 0.056g Y= 0.024g Z= 1.002g
2G MODE: X= 0.057g Y= 0.031g Z= 0.987g
2G MODE: X= 0.060g Y= 0.022g Z= 1.002g
2G MODE: X= 0.061g Y= 0.034g Z= 0.990g
```

```
Example finished!!!
```

Chapter 26

UART Interrupt Example

26.1 Overview

This UART example demonstrates the UART driver working with interrupt.

Transfer data between the board and computer. 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.

26.2 Supported platforms

26.2.1 i.MX 7Dual SABRE board

26.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

26.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

26.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/uart_imx/uart_interrupt/<toolchain>.

Supported platforms

26.2.1.4 Getting started

26.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

26.2.1.4.2 Running the demo

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

```
+++++ UART Send/Receive Interrupt Driven Example +++++  
Type characters from keyboard, the board will receive and then echo them to terminal screen
```

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

26.2.2 i.MX 7Dual Validation board

26.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

26.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

26.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/uart_imx/uart_interrupt/<toolchain>.

26.2.2.4 Getting started

26.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.


For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

26.2.2.4.2 Running the demo

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

```
+++++++ UART Send/Receive Interrupt Driven Example ++++++
Type characters from keyboard, the board will receive and then echo them to terminal screen
```

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



Supported platforms

Chapter 27

UART Polling Example

27.1 Overview

This UART example demonstrates the UART driver working with polling.

Transfer data between the board and computer. The board transfers and receives characters with the computer 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.

27.2 Supported platforms

27.2.1 i.MX 7Dual SABRE board

27.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

27.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

27.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_sdb_m4/driver_examples/uart_imx/uart-polling/<toolchain>.

Supported platforms

27.2.1.4 Getting started

27.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

27.2.1.4.2 Running the demo

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

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

```
Type characters from keyboard, the board will receive and then echo them to terminal screen
```

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

27.2.2 i.MX 7Dual Validation board

27.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal Computer with USB port

27.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

27.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx7d_val_m4/driver_examples/uart_imx/uart_polling/<toolchain>.

27.2.2.4 Getting started

27.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on PC for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM using U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).


27.2.2.4.2 Running the demo

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

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

```
Type characters from keyboard, the board will receive and then echo them to terminal screen
```

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



Supported platforms

Chapter 28

WDOG Example

28.1 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 four times, so the WDOG does not timeout until $4 + 1.5 = 5.5$ seconds.

28.2 Supported platforms

28.2.1 i.MX 7Dual SABRE board

28.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual SABRE-SD board
- Personal computer with USB port

28.2.1.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRT-OS1017DRN).

28.2.1.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_sdb_m4/driver_examples/wdog_imx/<toolchain>`.

Supported platforms

28.2.1.4 Getting started

28.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J11) on the i.MX 7Dual SABRE-SD board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM by U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FR-TOS7DGSUG).

28.2.1.4.2 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, as this is not a full system reset and many register fields of WDOG peripheral can write once only, so the second run will keep the WDOG setting unchanged. The WDOG counter will keep running and work just like the first time. The loop continues for ever.

28.2.2 i.MX 7Dual Validation board

28.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 7Dual
- Micro USB cable
- 5V DC adapter
- i.MX 7Dual Validation board
- Personal computer with USB port

28.2.2.2 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.1.0.1 for i.MX 7Dual Release Notes* (document FRTOS1017DRN).

28.2.2.3 Software requirements

- The project files are in: `<BSP_Install>/examples/imx7d_val_m4/driver_examples/wdog_imx/<toolchain>`.

28.2.2.4 Getting started

28.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the computer host and the Debug UART port (J28) on the i.MX 7Dual Validation board.
2. Open a serial terminal on the computer for Debug UART port with these settings:
 - 115200 baud rate
 - 8 data bits
 - No parity
 - One stop bit
 - No flow control
3. Load the demo binary to the TCM by U-Boot.
4. Boot auxiliary ARM Cortex-M4 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 7Dual* (document FRTOS7DGSUG).

28.2.2.4.2 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...
```

Supported platforms

A CPU reset occurs, as this is not a full system reset and many register fields of WDOG peripheral can write once only, so the second run keeps the WDOG setting unchanged. The WDOG counter keeps running and works just like the first time. The loop continues for ever.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein. Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

Freescale and the Freescale logo are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. All other product or service names are the property of their respective owners. ARM, ARM Powered, and Cortex are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.
© 2016 Freescale Semiconductor, Inc.

Document Number: FRTOS7D101DAUG
Rev. 0
07/2016

