FreeRTOS BSP for i.MX 6SoloX Demo User's Guide

NXP Semiconductors

Document Number: FRTOS6X101DAUG
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 6SoloX SABRE-SD 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 6SoloX SABRE-AI 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
Hello World DDR Demo

3.1 Overview . . . 7

3.2 Supported Platforms . . . 7
3.2.1 i.MX 6SoloX SABRE-SD board . . . 7
3.2.1.1 Hardware requirements . . . 7
3.2.1.2 Toolchain requirements . . . 7
3.2.1.3 Software requirements . . . 7

Section number	Title	Page
3.2.1.4	Getting Started	8
3.2.1.4.1	Prepare the Demo	8
3.2.1.4.2	Running the demo	8
3.2.2	i.MX 6SoloX SABRE-AI board	8
3.2.2.1	Hardware requirements	8
3.2.2.2	Toolchain requirements	8
3.2.2.3	Software requirements	9
3.2.2.4	Getting Started	9
3.2.2.4.1	Prepare the Demo	9
3.2.2.4.2	Running the demo	9

Chapter 4
Hello World Demo

4.1	**Overview**	**11**
4.2	**Supported Platforms**	**11**
4.2.1	i.MX 6SoloX SABRE-SD 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 6SoloX SABRE-AI 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 OCRAM Demo

5.1	**Overview**	**15**
5.2	**Supported Platforms**	**15**
5.2.1	i.MX 6SoloX SABRE-SD board	15
5.2.1.1	Hardware requirements	15
5.2.1.2	Toolchain requirements	15
5.2.1.3	Software requirements	15

Section number	Title	Page
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 6SoloX SABRE-AI board	16
5.2.2.1	Hardware requirements	16
5.2.2.2	Toolchain requirements	16
5.2.2.3	Software requirements	17
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 QSPI Demo

6.1	**Overview**	**19**
6.2	**Supported Platforms**	**19**
6.2.1	i.MX 6SoloX SABRE-SD 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 6SoloX SABRE-AI board	20
6.2.2.1	Hardware requirements	20
6.2.2.2	Toolchain requirements	21
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
Low Power CAN wakeup Demo

7.1	**Overview**	**23**
7.2	**Supported Platforms**	**23**
7.2.1	i.MX 6SoloX SABRE-SD board	23
7.2.1.1	Hardware requirements	23
7.2.1.2	Toolchain requirements	23
7.2.1.3	Software requirements	23

Section number	Title	Page
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 6SoloX SABRE-AI board	25
7.2.2.1	Hardware requirements	25
7.2.2.2	Toolchain requirements	26
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 Periodic WFI Demo

8.1	**Overview**	**29**
8.2	**Supported Platforms**	**29**
8.2.1	i.MX 6SoloX SABRE-SD 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
8.2.2	i.MX 6SoloX SABRE-AI board	31
8.2.2.1	Hardware requirements	31
8.2.2.2	Toolchain requirements	31
8.2.2.3	Software requirements	31
8.2.2.4	Getting Started	32
8.2.2.4.1	Prepare the Demo	32
8.2.2.4.2	Running the demo	32

Chapter 9
RPMsg PingPong Bare Metal Demo

9.1	**Overview**	**33**
9.2	**Supported Platforms**	**33**
9.2.1	i.MX 6SoloX SABRE-SD board	33
9.2.1.1	Hardware requirements	33

Section number	Title	Page
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 6SoloX SABRE-AI 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

Chapter 10
RPMsg PingPong FreeRTOS Demo with RTOS API

10.1	**Overview**	**37**
10.2	**Supported Platforms**	**37**
10.2.1	i.MX 6SoloX SABRE-SD 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 6SoloX SABRE-AI 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
RPMsg String Echo Bare Metal Demo

11.1	**Overview**	**41**
11.2	**Supported Platforms**	**41**
11.2.1	i.MX 6SoloX SABRE-SD board	41
11.2.1.1	Hardware requirements	41

Section number	Title	Page
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 6SoloX SABRE-AI 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

Chapter 12
RPMsg String Echo FreeRTOS Demo with RTOS API

12.1	**Overview**	**47**
12.2	**Supported Platforms**	**47**
12.2.1	i.MX 6SoloX SABRE-SD 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 6SoloX SABRE-AI 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 6SoloX SABRE-SD board	53
13.2.1.1	Hardware requirements	53

Section number	Title	Page
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 6SoloX SABRE-AI 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

Chapter 14
Sensor Demo

14.1	**Overview**	**59**
14.2	**Supported Platforms**	**59**
14.2.1	i.MX 6SoloX SABRE-SD 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
14.2.2	i.MX 6SoloX SABRE-AI board	61
14.2.2.1	Hardware requirements	61
14.2.2.2	Toolchain requirements	61
14.2.2.3	Software requirements	61
14.2.2.4	Getting Started	62
14.2.2.4.1	Prepare the Demo	62
14.2.2.4.2	Running the demo	62

Chapter 15
ADC Example

15.1	**Overview**	**65**
15.2	**Supported Platforms**	**65**
15.2.1	i.MX 6SoloX SABRE-SD board	65
15.2.1.1	Hardware requirements	65

Section number	Title	Page
15.2.1.2	Toolchain requirements	65
15.2.1.3	Software requirements	65
15.2.1.4	Getting Started	66
15.2.1.4.1	Prepare the Demo	66
15.2.1.4.2	Running the demo	66
15.2.2	i.MX 6SoloX SABRE-AI board	66
15.2.2.1	Hardware requirements	66
15.2.2.2	Toolchain requirements	66
15.2.2.3	Software requirements	67
15.2.2.4	Getting Started	67
15.2.2.4.1	Prepare the Demo	67
15.2.2.4.2	Running the demo	67

Chapter 16
ECSPI Interrupt Example

16.1	**Overview**	**69**
16.2	**Supported Platforms**	**69**
16.2.1	i.MX 6SoloX SABRE-SD board	69
16.2.1.1	Hardware requirements	69
16.2.1.2	Toolchain requirements	69
16.2.1.3	Software requirements	70
16.2.1.4	Getting Started	70
16.2.1.4.1	Hardware settings	70
16.2.1.4.2	Prepare the demo	71
16.2.1.4.3	Running the demo	72

Chapter 17
ECSPI Polling Example

17.1	**Overview**	**73**
17.2	**Supported Platforms**	**73**
17.2.1	i.MX 6SoloX SABRE-SD board	73
17.2.1.1	Hardware requirements	73
17.2.1.2	Toolchain requirements	73
17.2.1.3	Software requirements	74
17.2.1.4	Getting Started	74
17.2.1.4.1	Hardware settings	74
17.2.1.4.2	Prepare the demo	75
17.2.1.4.3	Running the demo	76

Section number Title Page

Chapter 18
EPIT Example

18.1 Overview . . . 77

18.2 Supported Platforms . . . 77
18.2.1 i.MX 6SoloX SABRE-SD board . . . 77
18.2.1.1 Hardware requirements . . . 77
18.2.1.2 Toolchain requirements . . . 77
18.2.1.3 Software requirements . . . 77
18.2.1.4 Getting Started . . . 78
18.2.1.4.1 Prepare the Demo . . . 78
18.2.1.4.2 Running the demo . . . 78
18.2.2 i.MX 6SoloX SABRE-AI board . . . 78
18.2.2.1 Hardware requirements . . . 78
18.2.2.2 Toolchain requirements . . . 79
18.2.2.3 Software requirements . . . 79
18.2.2.4 Getting Started . . . 79
18.2.2.4.1 Prepare the Demo . . . 79
18.2.2.4.2 Running the demo . . . 79

Chapter 19
FlexCAN Loopback Example

19.1 Overview . . . 81

19.2 Supported Platforms . . . 81
19.2.1 i.MX 6SoloX SABRE-SD board . . . 81
19.2.1.1 Hardware requirements . . . 81
19.2.1.2 Toolchain requirements . . . 81
19.2.1.3 Software requirements . . . 81
19.2.1.4 Getting Started . . . 82
19.2.1.4.1 Prepare the Demo . . . 82
19.2.1.4.2 Running the demo . . . 82
19.2.2 i.MX 6SoloX SABRE-AI board . . . 83
19.2.2.1 Hardware requirements . . . 83
19.2.2.2 Toolchain requirements . . . 83
19.2.2.3 Software requirements . . . 83
19.2.2.4 Getting Started . . . 84
19.2.2.4.1 Prepare the Demo . . . 84
19.2.2.4.2 Running the demo . . . 84

Section number | Title | Page

Chapter 20
FlexCAN Network Example

20.1 Overview . . . 87

20.2 Supported Platforms . . . 87
20.2.1 i.MX 6SoloX SABRE-SD board . . . 87
20.2.1.1 Hardware requirements . . . 87
20.2.1.2 Toolchain requirements . . . 87
20.2.1.3 Software requirements . . . 87
20.2.1.4 Getting Started . . . 88
20.2.1.4.1 Hardware settings . . . 88
20.2.1.4.2 Prepare the Demo . . . 89
20.2.1.4.3 Running the demo . . . 90
20.2.2 i.MX 6SoloX SABRE-AI board . . . 91
20.2.2.1 Hardware requirements . . . 91
20.2.2.2 Toolchain requirements . . . 91
20.2.2.3 Software requirements . . . 91
20.2.2.4 Getting Started . . . 91
20.2.2.4.1 Hardware settings . . . 91
20.2.2.4.2 Prepare the Demo . . . 93
20.2.2.4.3 Running the demo . . . 93

Chapter 21
GPIO Example

21.1 Overview . . . 95

21.2 Supported Platforms . . . 95
21.2.1 i.MX 6SoloX SABRE-SD board . . . 95
21.2.1.1 Hardware requirements . . . 95
21.2.1.2 Toolchain requirements . . . 95
21.2.1.3 Software requirements . . . 95
21.2.1.4 Getting Started . . . 96
21.2.1.4.1 Prepare the demo . . . 96
21.2.1.4.2 Running the demo . . . 96
21.2.2 i.MX 6SoloX SABRE-AI board . . . 96
21.2.2.1 Hardware requirements . . . 96
21.2.2.2 Toolchain requirements . . . 97
21.2.2.3 Software requirements . . . 97
21.2.2.4 Getting Started . . . 97
21.2.2.4.1 Prepare the demo . . . 97
21.2.2.4.2 Running the demo . . . 97

Chapter 22
I2C Interrupt Example

22.1 Overview . . . 99

22.2 Supported Platforms . . . 99
22.2.1 i.MX 6SoloX SABRE-SD board . . . 99
22.2.1.1 Hardware requirements . . . 99
22.2.1.2 Toolchain requirements . . . 99
22.2.1.3 Software requirements . . . 99
22.2.1.4 Getting Started . . . 100
22.2.1.4.1 Prepare the Demo . . . 100
22.2.1.4.2 Running the demo . . . 100
22.2.2 i.MX 6SoloX SABRE-AI board . . . 100
22.2.2.1 Hardware requirements . . . 100
22.2.2.2 Toolchain requirements . . . 101
22.2.2.3 Software requirements . . . 101
22.2.2.4 Getting Started . . . 101
22.2.2.4.1 Prepare the Demo . . . 101
22.2.2.4.2 Running the demo . . . 101

Chapter 23
I2C Polling Example

23.1 Overview . . . 103

23.2 Supported Platforms . . . 103
23.2.1 i.MX 6SoloX SABRE-SD 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
23.2.2 i.MX 6SoloX SABRE-AI board . . . 104
23.2.2.1 Hardware requirements . . . 104
23.2.2.2 Toolchain requirements . . . 105
23.2.2.3 Software requirements . . . 105
23.2.2.4 Getting Started . . . 105
23.2.2.4.1 Prepare the Demo . . . 105
23.2.2.4.2 Running the demo . . . 105

Section number Title Page

Chapter 24
UART Interrupt Example

24.1 Overview . . . 107

24.2 Supported Platforms . . . 107
24.2.1 i.MX 6SoloX SABRE-SD board . . . 107
24.2.1.1 Hardware requirements . . . 107
24.2.1.2 Toolchain requirements . . . 107
24.2.1.3 Software requirements . . . 107
24.2.1.4 Getting Started . . . 108
24.2.1.4.1 Prepare the Demo . . . 108
24.2.1.4.2 Running the demo . . . 108
24.2.2 i.MX 6SoloX SABRE-AI board . . . 108
24.2.2.1 Hardware requirements . . . 108
24.2.2.2 Toolchain requirements . . . 108
24.2.2.3 Software requirements . . . 109
24.2.2.4 Getting Started . . . 109
24.2.2.4.1 Prepare the Demo . . . 109
24.2.2.4.2 Running the demo . . . 109

Chapter 25
UART Polling Example

25.1 Overview . . . 111

25.2 Supported Platforms . . . 111
25.2.1 i.MX 6SoloX SABRE-SD board . . . 111
25.2.1.1 Hardware requirements . . . 111
25.2.1.2 Toolchain requirements . . . 111
25.2.1.3 Software requirements . . . 111
25.2.1.4 Getting Started . . . 112
25.2.1.4.1 Prepare the Demo . . . 112
25.2.1.4.2 Running the demo . . . 112
25.2.2 i.MX 6SoloX SABRE-AI board . . . 112
25.2.2.1 Hardware requirements . . . 112
25.2.2.2 Toolchain requirements . . . 112
25.2.2.3 Software requirements . . . 113
25.2.2.4 Getting Started . . . 113
25.2.2.4.1 Prepare the Demo . . . 113
25.2.2.4.2 Running the demo . . . 113

Chapter 26
WDOG Example

26.1 Overview . . . 115

26.2 Supported Platforms . . . 115
26.2.1 i.MX 6SoloX SABRE-SD board . . . 115
26.2.1.1 Hardware requirements . . . 115
26.2.1.2 Toolchain requirements . . . 115
26.2.1.3 Software requirements . . . 115
26.2.1.4 Getting Started . . . 116
26.2.1.4.1 Prepare the Demo . . . 116
26.2.1.4.2 Running the demo . . . 116
26.2.2 i.MX 6SoloX SABRE-AI board . . . 116
26.2.2.1 Hardware requirements . . . 116
26.2.2.2 Toolchain requirements . . . 117
26.2.2.3 Software requirements . . . 117
26.2.2.4 Getting Started . . . 117
26.2.2.4.1 Prepare the Demo . . . 117
26.2.2.4.2 Running the demo . . . 117

Chapter 1
Introduction

FreeRTOS BSP 1.0.1 for i.MX 6SoloX includes applications which provide examples that show how to use this BSP for i.MX 6SoloX Processor. This document describes these 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 at the Linux OS side. To run the examples with Linux OS together, pay attention to the following:

- By default, Linux BSP does not come with Cortex-M4 enabled. The device tree needs to be changed in U-Boot with the command setenv fdt_file zImage-imx6sx-sdb-m4.dtb for SABRE-SD or setenv fdt_file zImage-imx6sx-sabreauto-m4.dtb for SABRE-AI.
- 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 EPIT 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's not recommended to run this demo with default Linux OS kernel together.

2.2 Supported Platforms

2.2.1 i.MX 6SoloX SABRE-SD board

2.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

2.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/blinking_imx_demo/<toolchain>.

2.2.1.4 Getting Started

2.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 pushes 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 blinking interval increases to 1000 ms. The user can push the button again to recover the blinking interval to 100 ms.

2.2.2 i.MX 6SoloX SABRE-AI board

2.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable

- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

2.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_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 PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 ======

Wait 5 seconds to switch blinking frequency:

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

Wait 5 seconds to switch blinking frequency:

The Debug LED(D11) 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
Hello World DDR Demo

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

NOTE: The DDR memory is system resource, so make sure there is no conflict in memory allocation between the Cortex® -A9 core and the Cortex-M4 core. The code space of the FreeRTOS demo is defined in platform/devices/MCIMX6X/linker/<toolchain>/MCIMX6X_M4_ddr.<ext> and the base address and length of the code region can be changed per requirement.

3.2 Supported Platforms

3.2.1 i.MX 6SoloX SABRE-SD board

3.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD 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 6SoloX Release Notes* (document FRTOS1016XRN).

3.2.1.3 Software requirements

- The hello world ddr project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/hello_world_ddr/<toolchain>.

3.2.1.4 Getting Started

3.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 6SoloX* (document FRTOS6XGSUG).

3.2.1.4.2 Running the demo

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

```
Hello World!
```

You can puts some inputs like this :

```
Hello World!
test
```

3.2.2 i.MX 6SoloX SABRE-AI board

3.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main board
- Personal Computer with USB port

3.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

3.2.2.3 Software requirements

- The hello world ddr project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/hello_world_ddr/<toolchain>.

3.2.2.4 Getting Started

3.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 6SoloX* (document FRTOS6XGSUG).

3.2.2.4.2 Running the demo

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

```
Hello World!
```

You can puts some inputs like this :

```
Hello World!
test
```

Chapter 4
Hello World Demo

4.1 Overview

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

4.2 Supported Platforms

4.2.1 i.MX 6SoloX SABRE-SD board

4.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

4.2.1.3 Software requirements

- The hello world project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/hello_world/<toolchain>.

4.2.1.4 Getting Started

4.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

4.2.2 i.MX 6SoloX SABRE-AI board

4.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

4.2.2.3 Software requirements

- The hello world project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/hello_world/<toolchain>.

4.2.2.4 Getting Started

4.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

Chapter 5
Hello World OCRAM Demo

5.1 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 additon to that, RDC memory protection is introduced in hardware_init.c. The FreeRTOS code space read/write operation is allowed by both Cortex-A9 and Cortex-M4 core.

NOTE: The OCRAM memory is system resource, so make sure there is no conflict in memory allocation between the Cortex-A9 core and the Cortex-M4 core. The code space of FreeRTOS demo is defined in platform/devices/MCIMX6X/linker/<toolchain>/MCIMX6X_M4_ocram.<ext> and the base address and length of the code region can be changed per requirement.

5.2 Supported Platforms

5.2.1 i.MX 6SoloX SABRE-SD board

5.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

5.2.1.3 Software requirements

- The hello world ocram project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/hello_world_ocram/<toolchain>.

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(J16) on the i.MX 6SoloX 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 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

5.2.2 i.MX 6SoloX SABRE-AI board

5.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

5.2.2.3 Software requirements

- The hello world ocram project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/hello_world_ocram/<toolchain>.

5.2.2.4 Getting Started

5.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

Chapter 6
Hello World QSPI Demo

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

NOTE: The QSPI memory is system resource, so make sure there is no conflict in memory allocation between the Cortex-A9 core and the Cortex-M4 core. The code space of FreeRTOS demo is defined in platform/devices/MCIMX6X/linker/<toolchain>/MCIMX6X_M4_qspi<N>b.<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 6SoloX SABRE-SD board

6.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

6.2.1.3 Software requirements

- The hello world qspi project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/hello_world_qspi/<toolchain>.

- To kick off the hello world qspi demo, U-Boot must be configured with QSPI flash enabled.

6.2.1.4 Getting Started

6.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

6.2.2 i.MX 6SoloX SABRE-AI board

6.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

6.2.2.3 Software requirements

- The hello world qspi project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/hello_world_qspi/<toolchain>.
- To kick off the hello world qspi demo, U-Boot must be configured with QSPI flash enabled.

6.2.2.4 Getting Started

6.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 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 6SoloX* (document FRTOS6XGSUG).

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 puts some inputs like this :

```
Hello World!
test
```

Chapter 7
Low Power CAN wakeup Demo

This document explains the low power CAN wakeup example for i.MX 6SoloX; The demo consists of two applications to demonstrate how to wakeup SoC in low power mode (with CAN in stop mode) when it detects CAN message from the CAN bus.

7.1 Overview

The demo contains two applications: one for low power CAN receiver and the other for CAN transmitter. The CAN receiver runs in low power and support the whole SoC to enter SUSPEND mode (depending on the status of Cortex A9). And the CAN transmitter could wake up the CAN receiver at a 5 second interval.

7.2 Supported Platforms

7.2.1 i.MX 6SoloX SABRE-SD board

7.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- 2 i.MX 6SoloX SABRE-SD boards
- Personal Computer with USB port
- 2 Pins electric cable

7.2.1.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

7.2.1.3 Software requirements

- The CAN wakeup receiver project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/low_power_imx6sx/can_wakeup/receiver_qspi/<toolchain>.

- The CAN wakeup transmitter project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/low_power_imx6sx/can_wakeup/transmitter_qspi/<toolchain>.
- The default linker script is for QSPI flash. In order to kick off this demo, U-Boot must be configured with QSPI flash enabled.
- Running Uboot script listed in "Prepare the Demo" part to clean Shared Clock Memory before boot M4 image and Cortex-A9 image.

7.2.1.4 Getting Started

7.2.1.4.1 Hardware settings

To run this example, connect two board through the CAN interface: CANH to CANH and CANL to CANL.

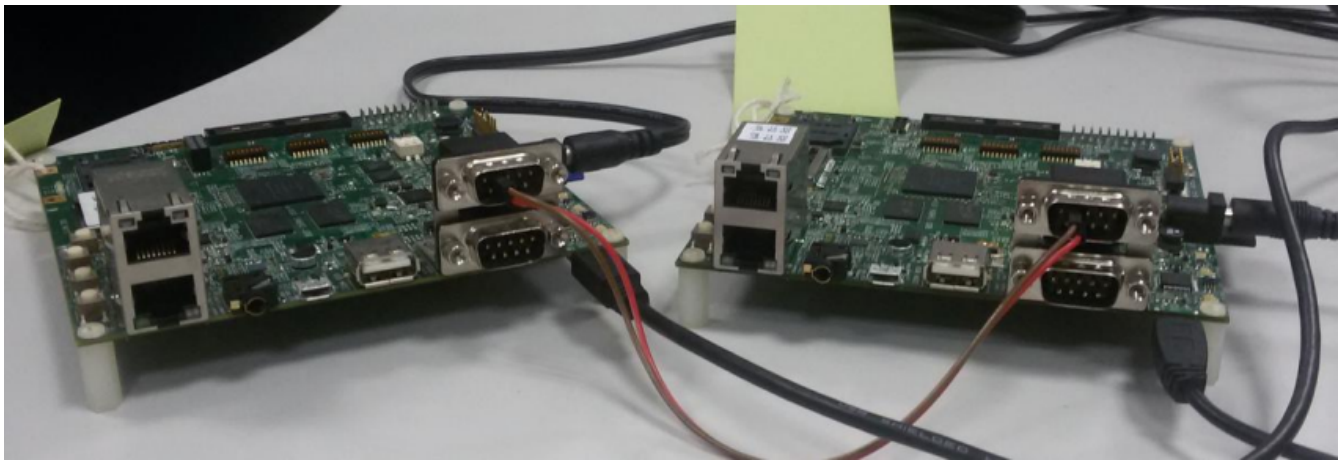


Figure 7.2.1: Connecting two board through the CAN interface

7.2.1.4.2 Prepare the Demo

1. Connect 2 USB cable between the PC host and the Debug UART port(J16) on each the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Connect these 2 boards to CAN Bus.
4. Load the demo binary to target board using U-Boot.
5. Execute U-Boot Command 'mw.w 0x0091F000 0x0 4; dcache flush;' to clean the Shared Clock Memory Magic Number of Cortex-A9 and Cortex-M4 before boot any one of them.
6. Boot M4 image and A9 image to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 6SoloX* (document FRTOS6XGSUG).

7.2.1.4.3 Running the demo

To run the example, the corresponding compiler and a terminal program are needed. First start the CAN receiver program, and when "Enter flexcan stop mode" appears on the terminal, you can start the CAN transmitter program. To verify SoC wakeup from SUSPEND mode, you can run CAN receiver (on Cortex-M4) and Linux OS (on Cortex-A9) on same board, and run CAN transmitter on the other one. After CAN receiver and CAN transmitter start, make Linux OS kernel suspend at any time to see the system (both A9 and M4) could be woken up by the CAN transmitter.

The expected console output is as follows when running with Linux OS:

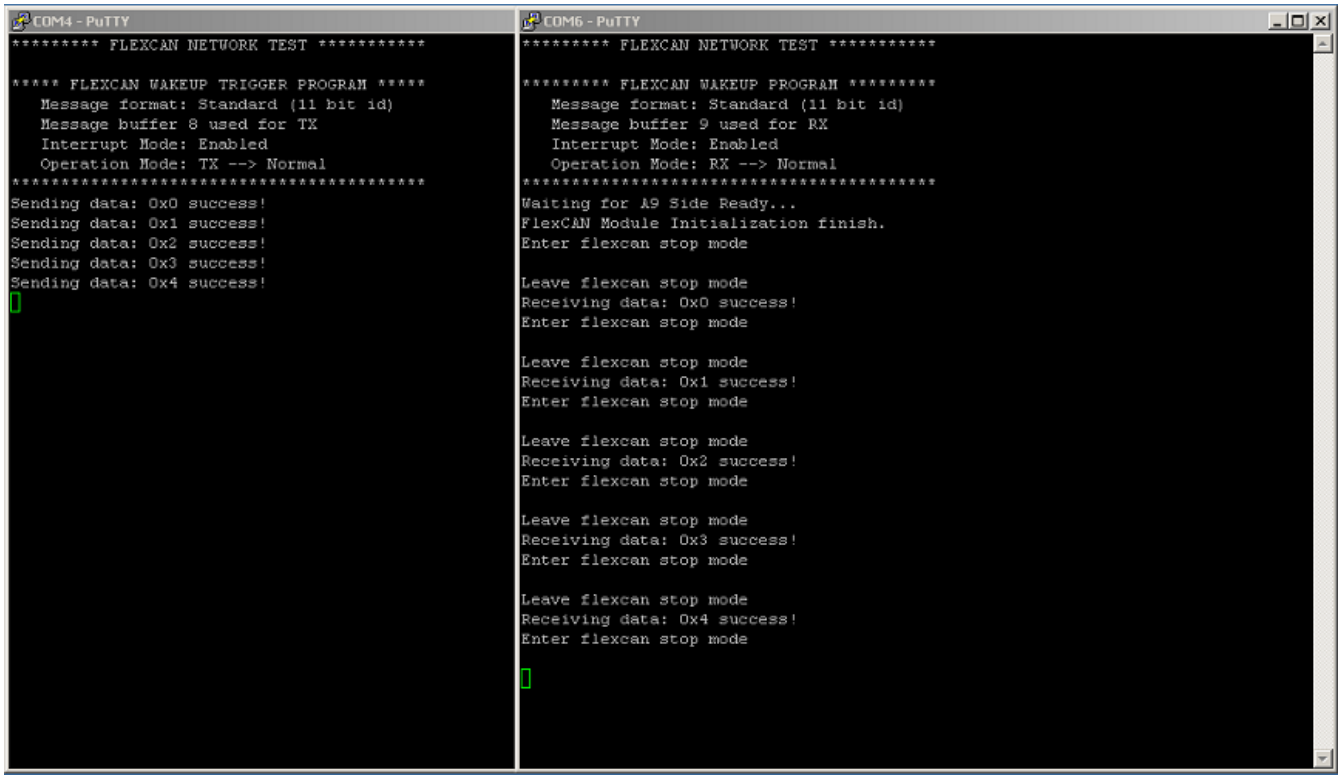


Figure 7.2.2: Expected console output

7.2.2 i.MX 6SoloX SABRE-AI board

7.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable

- 5V DC adapter
- 2 i.MX 6SoloX SABRE-AI main boards
- 2 i.MX 6SoloX SABRE Automotive Expansion boards
- Personal Computer with USB port
- 2 Pins electric cable

7.2.2.2 Toolchain requirements

One of following toolchains is required:

- IAR Embedded Workbench
- ARM GCC
- DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 6SoloX Release Notes* (document FRTOS1016XRN).

7.2.2.3 Software requirements

- The CAN wakeup receiver project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/low_power_imx6sx/can_wakeup/receiver_qspi/<toolchain>.
- The CAN wakeup transmitter project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/low_power_imx6sx/can_wakeup/transmitter_qspi/<toolchain>.
- The default linker script is for QSPI flash. In order to kick off this demo, U-Boot must be configured with QSPI flash enabled.

7.2.2.4 Getting Started

7.2.2.4.1 Hardware settings

To run this example, connect two board through the CAN interface: CANH to CANH and CANL to CANL.

7.2.2.4.2 Prepare the Demo

1. Connect 2 USB cable between the PC host and the Debug UART port(J16) on each the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Connect these 2 boards to CAN Bus.
4. Load the demo binary to target board using U-Boot.

5. Boot Cortex-M4 Core and Cortex-A9 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 6SoloX* (document FRTOS6XGSUG).

7.2.2.4.3 Running the demo

To run the example, the corresponding compiler and a terminal program are needed. First start the CAN receiver program, and when "Enter flexcan stop mode" appears on the terminal, you can start the CAN transmitter program. To verify SoC wakeup from SUSPEND mode, you can run CAN receiver (on Cortex-M4) and Linux OS (on Cortex-A9) on same board, and run CAN transmitter on the other one. After CAN receiver and CAN transmitter start, make Linux OS kernel suspend at any time to see the system (both A9 and M4) could be woken up by the CAN transmitter.

The expected console output is as follows when running with Linux OS:

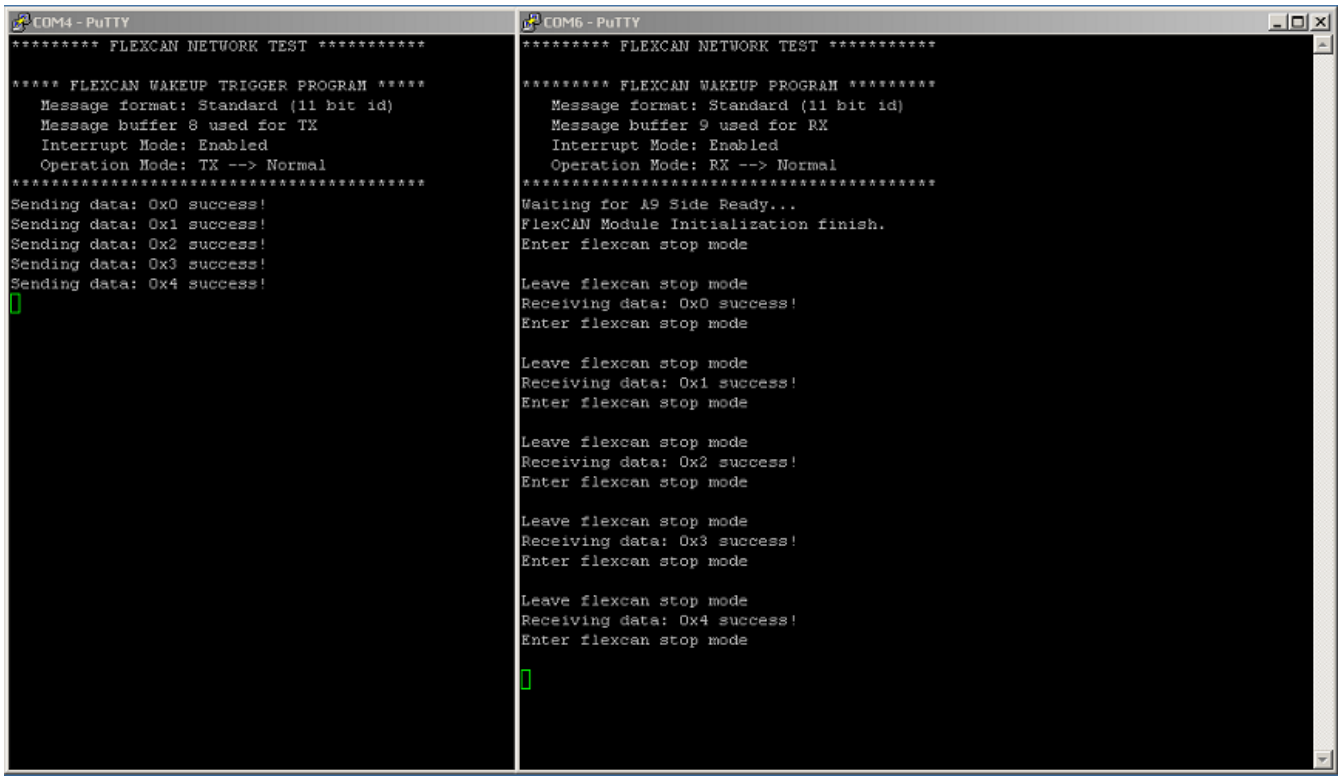


Figure 7.2.3: Expected console output

Chapter 8
Low Power Periodic WFI Demo

This demo application exhibit the Low-power feature of i.MX 6SoloX SoC.

8.1 Overview

This demo exhibits dual core low power management. i.MX 6SoloX is a dual-core chip, including a Cortex-A9 core and a Cortex-M4 core. The A9 Core is the power management arbiter, it controls the chip level power mode. The M4 Core and its peripherals act as a general device attached to A9. The application on M4 should supervise the state of all its peripherals, when all of them are running in low speed, M4 can enter its own power saving mode. In this mode, M4 can release all the high power resources, including PLL, QSPI clock, etc. M4 can then inform A9 about this status, A9 can then actually shut down these resources based on the system level situation (including M4 and other peripherals directly managed by A9).

This demo is based on EPIT timer on M4 side. With the timer, the M4 can switches between high power running mode and low power running mode continuously.

8.2 Supported Platforms

8.2.1 i.MX 6SoloX SABRE-SD board

8.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD boards
- Personal Computer with USB port

8.2.1.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

8.2.1.3 Software requirements

- The periodic WFI project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/low-_power_imx6sx/periodic_wfi_qspi/<toolchain>.
- The default linker script is for QSPI flash. In order to kick off this demo, U-Boot must be configured with QSPI flash enabled.
- Running Uboot script listed in "Prepare the Demo" part to clean Shared Clock Memory before boot M4 image and Cortex-A9 image.

8.2.1.4 Getting Started

8.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 target board using U-Boot.
4. Execute U-Boot Command 'mw.w 0x0091F000 0x0 4; dcache flush;' to clean the Shared Clock Memory Magic Number of Cortex-A9 and Cortex-M4 before boot any one of them.
5. Boot Cortex-M4 Core and Cortex-A9 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 6SoloX* (document FRTOS6XGSUG).

8.2.1.4.2 Running the demo

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

```
          i.MX 6SoloX Dual Core Low Power Demo - M4 side              *
                                                                      *
       A EPIT will change the M4 running speed every 5s               *
                                                                      *

Waiting for A9 Side Ready...
```

After Cortex-A9 Core boot up, Cortex-M4 will switch between high power mode and low power mode every 5 seconds:

```
Waiting for A9 Side Ready...

Allow deep sleep
```

```
Deny deep sleep

Allow deep sleep

Deny deep sleep

Allow deep sleep

Deny deep sleep

Allow deep sleep

Deny deep sleep

...
```

8.2.2 i.MX 6SoloX SABRE-AI board

8.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main board
- Personal Computer with USB port

8.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

8.2.2.3 Software requirements

- The periodic WFI project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/low_power_imx6sx/periodic_wfi_qspi/<toolchain>.
- The default linker script is for QSPI flash. In order to kick off this demo, U-Boot must be configured with QSPI flash enabled.

8.2.2.4 Getting Started

8.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Load the demo binary to target board using U-Boot.
4. Execute U-Boot Command 'mw.w 0x0091F000 0x0 4; dcache flush;' to clean the Shared Clock Memory Magic Number of Cortex-A9 and Cortex-M4 before boot any one of them.
5. Boot Cortex-M4 Core and Cortex-A9 Core to begin running the demo.

For the detailed instructions, see the *Getting Started with FreeRTOS BSP for i.MX 6SoloX* (document FRTOS6XGSUG).

8.2.2.4.2 Running the demo

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

```
          i.MX 6SoloX Dual Core Low Power Demo - M4 side              *
                                                                      *
        A EPIT will change the M4 running speed every 5s              *
                                                                      *

Waiting for A9 Side Ready...
```

After Cortex A9 Core boot up, Cortex-M4 will switch between high power mode and low power mode every 5 seconds:

```
Waiting for A9 Side Ready...

Allow deep sleep

Deny deep sleep

Allow deep sleep

Deny deep sleep

Allow deep sleep

Deny deep sleep

Allow deep sleep

Deny deep sleep

...
```

Chapter 9
RPMsg PingPong Bare Metal Demo

9.1 Overview

This demo application demonstrates the RPMsg remote peer stack working on bare metal with OpenAMP RPMsg API. It works with Linux RPMsg master peer to transfer integer values back and forth. The name service handshake is performed first to create the communication channels. Next, Linux OS transfers the first integer to 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 6SoloX SABRE-SD board

9.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

9.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/rpmsg/pingpong_bm/<toolchain>.
- Linux RPMsg master side pingpong module

9.2.1.4 Getting Started

9.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-SD board.
2. Open two serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 pingpong 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-A9 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform a name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 begins to send the first data to M4. After receiving the data, M4 adds 1 to it and sends it back to A9. A9 responds with the same behavior.

9.2.2 i.MX 6SoloX SABRE-AI board

9.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

9.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/rpmsg/pingpong_bm/<toolchain>.
- Linux RPMsg master side pingpong module

9.2.2.4 Getting Started

9.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI board.
2. Open two serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 pingpong 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-A9 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform a name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 begins to send the first data to M4. After receiving the data, M4 adds 1 to it and sends it back to A9. A9 responds with the same behavior.

Chapter 10
RPMsg PingPong FreeRTOS Demo with RTOS API

10.1 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 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 RPMsg RTOS API extension is a set of easy to use APIs that support multi-task receive/send messages.

If the raw RPMsg API is preferred, it's also easy to integrate with FreeRTOS by referring to the pingpong_bm demo.

10.2 Supported Platforms

10.2.1 i.MX 6SoloX SABRE-SD board

10.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

10.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/rpmsg/pingpong_freertos/<toolchain>.

Supported Platforms

- Linux RPMsg master side pingpong module

10.2.1.4 Getting Started

10.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-SD board.
2. Open two serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

10.2.1.4.2 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 pingpong 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-A9 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform a name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 begins to send the first data to M4. After receiving the data, M4 adds 1 to it and sends it back to A9. A9 responds with the same behavior.

10.2.2 i.MX 6SoloX SABRE-AI board

10.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

10.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/rpmsg/pingpong_freertos/<toolchain>.
- Linux RPMsg master side pingpong module

10.2.2.4 Getting Started

10.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI board.
2. Open two serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 pingpong 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-A9 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform a name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 begins to send the first data to M4. After receiving the data, M4 adds 1 to it and sends it back to A9. A9 responds with the same behavior.

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 which is received is sent to M4. M4 displays what is received, and echoes back the same message as an acknowledgement. The tty reader on A9 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 rpmsg_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 will be divided by virtual tty into several messages.

11.2 Supported Platforms

11.2.1 i.MX 6SoloX SABRE-SD board

11.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

11.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/rpmsg/str_echo_-bm/<toolchain>.
- Linux RPMsg master side RPMsg 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 PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-SD board.
2. Open TWO serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

11.2.1.4.2 Running the demo

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

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

After the Linux RPMsg 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 RPMsg tty using the following echo command on Cortex-A9 terminal:

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

```
echo deadbeaf > /dev/ttyRPMSG

...
```

On the 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-A9 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 then waits for user input to RPMsg virtual tty and sends the content to M4. On receiving the data, M4 outputs the content and its length on the terminal and echoes back the same message to A9. If some application is reading from /dev/ttyRPMSG on A9, 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 6SoloX SABRE-AI board

11.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main board
- Personal Computer with USB port

11.2.2.2 Toolchain requirements

One of following toolchains is required:

- IAR Embedded Workbench
- ARM GCC

Supported Platforms

- ARM DS-5

For the toolchain version, see the *FreeRTOS BSP v.1.0.1 for i.MX 6SoloX Release Notes* (document FRTOS1016XRN).

11.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_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 PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI board.
2. Open TWO serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

11.2.2.4.2 Running the demo

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

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

After the Linux RPMsg 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 RPMsg tty using the following echo command on Cortex-A9 terminal:

```
echo test > /dev/ttyRPMSG

echo deadbeaf > /dev/ttyRPMSG

...
```

On the 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-A9 Linux OS) and Remote (Cortex-M4 Bare Metal Application) perform name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 then waits for user input to RPMsg virtual tty and sends the content to M4. On receiving the data, M4 outputs the content and its length on the terminal and echoes back the same message to A9. If some application is reading from /dev/ttyRPMSG on A9, 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.

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 M4. M4 displays what is received, and echoes back the same message as an acknowledgement. The tty reader on A9 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 multi-task 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's also easy to integrate with FreeRTOS by referring to the str_echo_bm 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 6SoloX SABRE-SD board

12.2.1.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

12.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/rpmsg/str_echo_-freertos/<toolchain>.
- Linux RPMsg master side RPMsg 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 PC host and the Debug UART port (J16) on the i.MX 6SoloX SABRE-SD board.
2. Open TWO serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

12.2.1.4.2 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 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 RPMsg tty using the following echo command on Cortex-A9 terminal:

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

```
echo deadbeaf > /dev/ttyRPMSG

...
```

On the 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-A9 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 then waits for user input to RPMsg virtual tty and sends the content to M4. On receiving the data, M4 outputs the content and its length on the terminal and echoes back the same message to A9. If some application is reading from /dev/ttyRPMSG on A9, 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 Cortex-A9 could be well buffered for future receiving.

12.2.2 i.MX 6SoloX SABRE-AI board

12.2.2.1 Hardware requirements

- SD Card with Linux OS image for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

12.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_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 PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI board.
2. Open TWO serial terminals 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 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 RPMsg tty using the following echo command on Cortex-A9 terminal:

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

```
echo deadbeaf > /dev/ttyRPMSG

...
```

On the 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-A9 Linux OS) and Remote (Cortex-M4 FreeRTOS OS) perform name service handshake to create the communication channel. The M4 channel address is 1, and the A9 channel address is 1024. A9 then waits for user input to RPMsg virtual tty and sends the content to M4. On receiving the data, M4 outputs the content and its length on the terminal and echoes back the same message to A9. If some application is reading from /dev/ttyRPMSG on A9, 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 Cortex-A9 could be well buffered for future receiving.

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 6SoloX SABRE-SD board

13.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

13.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/sema4_demo/<toolchain>.

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(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 6SoloX:

```
=> mw.b 0x2290003 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 6SoloX:

```
=> mw.b 0x2290003 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 6SoloX SABRE-AI board

13.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

13.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/sema4_demo/<toolchain>.

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(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 6SoloX:

```
=> mw.b 0x2290003 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 6SoloX:

```
=> mw.b 0x2290003 0
```

You can immediately see on SEMA4 demo Terminal:

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

Chapter 14
Sensor Demo

14.1 Overview

The Sensor Demo i.MX 6SoloX is a simple demonstration program that uses the FreeRTOS and a set of drivers provided by NXP. It can get the current gravitational acceleration 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.

14.2 Supported Platforms

14.2.1 i.MX 6SoloX SABRE-SD board

14.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

14.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/demo_apps/sensor_demo/<toolchain>.

14.2.1.4 Getting Started

14.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

14.2.1.4.2 Running the demo

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

```
-------------- i.MX 6SoloX SABRE 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].MMA8451Q 3-Axis Digital Accelerometer Polling Demo
[2].MMA8451Q 3-Axis Digital Accelerometer Interrupt Demo
[3].MAG3110  3-Axis Digital Magnetometer Polling Demo
[4].MAG3110  3-Axis Digital Magnetometer Interrupt Demo
```

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

For MMA8451Q acceleration sensor(U26), it will continuous monitor current acceleration of gravity in 3 axes every 1 seconds in polling mode and every 1.58 seconds(0.63hz) in interrupt mode. And print to the terminal like this:

```
[MMA8451Q]Current Acc:X=-0.01537g Y=-0.00268g Z=-0.98827g
```

If you rotate the board the terminal print the different data like this:

```
[MMA8451Q]Current Acc:X=-0.13622g Y=-0.96459g Z=-0.05126g
```

For MAG3110 magnetic sensor(U27), it will read current magnetic field intensity of the board in 3 axes every 1 seconds in polling mode and every 0.63 seconds(1.58hz) in interrupt mode. And print current value to the terminal like this:

```
[MAG3110]Current Mag:X=  -6.8uT Y=  47.0uT Z=  45.7uT
```

If you put the headset with sounds close to the MAG3110 sensor(U27) the terminal print the different data like this:

```
[MAG3110]Current Mag:X= 124.2uT Y=   8.0uT Z= 229.4uT
```

14.2.2 i.MX 6SoloX SABRE-AI board

14.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 12V DC adapter
- i.MX 6SoloX SABRE-AI main board
- i.MX 6SoloX SABRE Automotive Expansion board
- Personal Computer with USB port

14.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

14.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/demo_apps/sensor_demo/<toolchain>.

14.2.2.4 Getting Started

14.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

14.2.2.4.2 Running the demo

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

```
-------------- i.MX 6SoloX SABRE 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].MMA8451Q 3-Axis Digital Accelerometer Polling Demo
[2].MMA8451Q 3-Axis Digital Accelerometer Interrupt Demo
[3].MAG3110  3-Axis Digital Magnetometer Polling Demo
[4].MAG3110  3-Axis Digital Magnetometer Interrupt Demo
```

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

For MMA8451Q acceleration sensor(U33), it will continuous monitor current acceleration of gravity in 3 axes every 1 seconds in polling mode and every 1.58 seconds(0.63hz) in interrupt mode. And print to the terminal like this:

```
[MMA8451Q]Current Acc:X=-0.00902g Y=-0.03905g Z=-0.95287g
```

If you rotate the board the terminal print the different data like this:

```
[MMA8451Q]Current Acc:X=-0.91967g Y=0.30933g Z=0.15332g
```

For MAG3110 magnetic sensor(U31), it will read current magnetic field intensity of the board in 3 axes every 1 seconds in polling mode and every 0.63 seconds(1.58hz) in interrupt mode. And print current value to the terminal like this:

```
[MAG3110]Current Mag:X=  -6.8uT Y=  47.0uT Z=  45.7uT
```

If you put the headset with sounds close to the MAG3110 sensor(U31) the terminal print the different data like this:

```
[MAG3110]Current Mag:X= 124.2uT Y=   8.0uT Z= 229.4uT
```

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 5 seconds to wakes up the ADC module. Then ADC module will convert the analog input to digital output displayed in Terminal.

15.2 Supported Platforms

15.2.1 i.MX 6SoloX SABRE-SD board

15.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

15.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/adc_imx6sx/<toolchain>.

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(J16) on the i.MX 6SoloX SABRE-SD board.
2. Connect ADC1_IN3(J20-PIN4) 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 DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

15.2.1.4.2 Running the demo

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

```
-------------- ADC imx6sx 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 6SoloX SABRE-AI board

15.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI 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 6SoloX Release Notes* (document FRTOS1016XRN).

15.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/adc_imx6sx/<toolchain>.

15.2.2.4 Getting Started

15.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J19) on the i.MX 6SoloX SABRE-AI board.
2. Connect ADC1_IN3(J11-PIN9) 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 DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

15.2.2.4.2 Running the demo

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

```
-------------- ADC imx6sx driver example --------------

This example demonstrates usage of ADC driver on i.MX 6SoloX processor.
It Continuous convert Analog Input, and print the result to terminal
Current analog value: 1.06v
Current analog value: 1.04v
```

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.

- ECSPI board to board:
  - Transfer data through ECSPI4 instance, ECSPI4 pins of master board are connected with ECSPI4 pins of slave board. CS0 (chip select) of master board are connected with CS0 of slave board.
  - Master sends an array to slave and receives the array back from slave. Slave receives an array from master and sends back another array to master.
- Note:
  - The SD2_CLK, SD2_CMD, SD2_DATA0, SD2_DATA3 pins used in this example are overlapped with SD WIFI function in Linux on Cortex-A9 Core. So this example cannot run with Linux running on Cortex-A9 Core at the same time.
  - You need to customize the code, if eCSPI and SD WIFI are needed in your application.

16.2 Supported Platforms

16.2.1 i.MX 6SoloX SABRE-SD board

16.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5 V DC adapter
- i.MX 6SoloX SABRE-SD 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 6SoloX Release Notes* (document FRTOS1016XRN).

16.2.1.3 Software requirements

- The master project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/ecspi/ecspi_interrupt/master/<toolchain>.
- The slave project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/ecspi/ecspi_interrupt/slave/<toolchain>.

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. This example requires two separate boards. You can connect ECSPI4 signals using shielded wire like this:

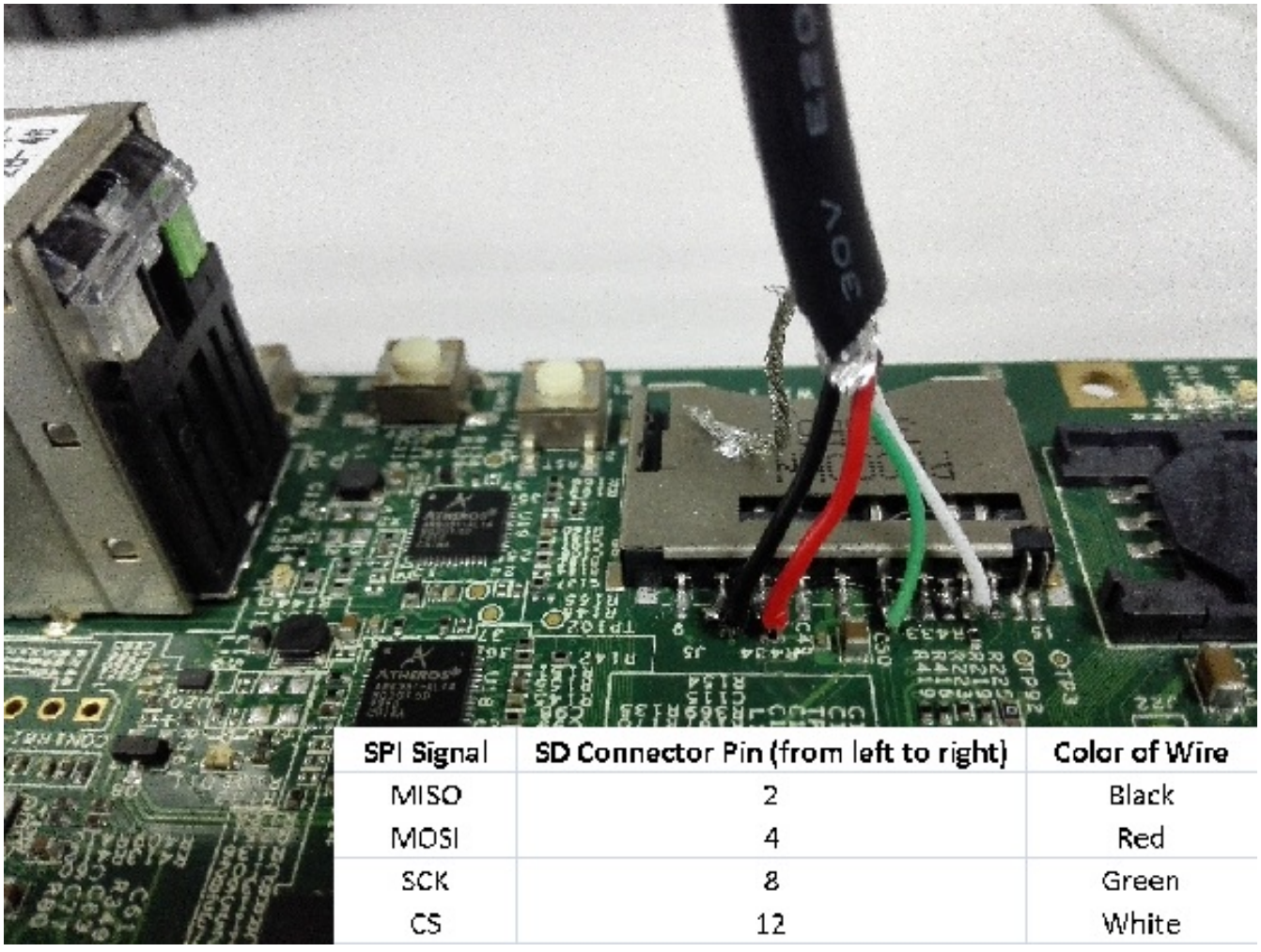


Figure 16.2.1: Connecting ECSPI4 signals using shielded wire (1)

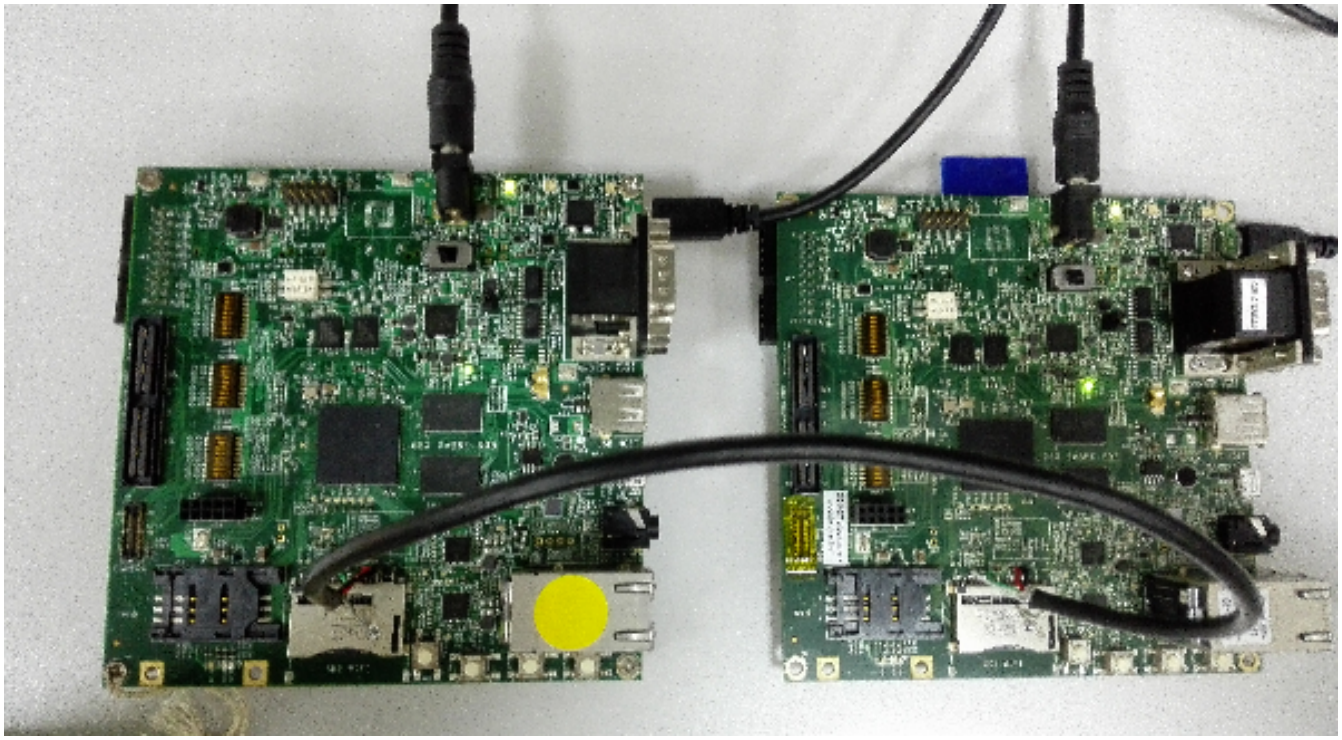


Figure 16.2.2: Connecting ECSPI4 signals using shielded wire (2)

Note: Users need to ensure that the cable connected 2 boards should be shielded and as short as possible.

16.2.1.4.2 Prepare the demo

1. Connect two micro USB cable for each PC host and Debug UART port(J16) on the i.MX 6SoloX SABRE-SD board.
2. Open two 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. Connect these 2 board to ECSPI4.
4. Load the demo binary to the DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 will print on terminal:

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

MASTER: Transmited data: 2
      : Received data: 0

MASTER: Transmited data: 3
      : Received data: 1

...

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

The slave board will print on terminal:

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

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

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

...

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

Chapter 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.

- ECSPI board to board:
  - Transfer data through ECSPI4 instance, ECSPI4 pins of master board are connected with ECSPI4 pins of slave board. CS0 (chip select) of master board are connected with CS0 of slave board.
  - Master sends an array to slave and receives the array back from slave. Slave receives an array from master and sends back another array to master.
- Note:
  - The SD2_CLK, SD2_CMD, SD2_DATA0, SD2_DATA3 pins used in this example are overlapped with SD WIFI function in Linux on Cortex-A9 Core. So this example cannot run with Linux running on Cortex-A9 Core at the same time.
  - You need to customize the code, if eCSPI and SD WIFI are needed in your application.

17.2 Supported Platforms

17.2.1 i.MX 6SoloX SABRE-SD board

17.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD 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 6SoloX Release Notes* (document FRTOS1016XRN).

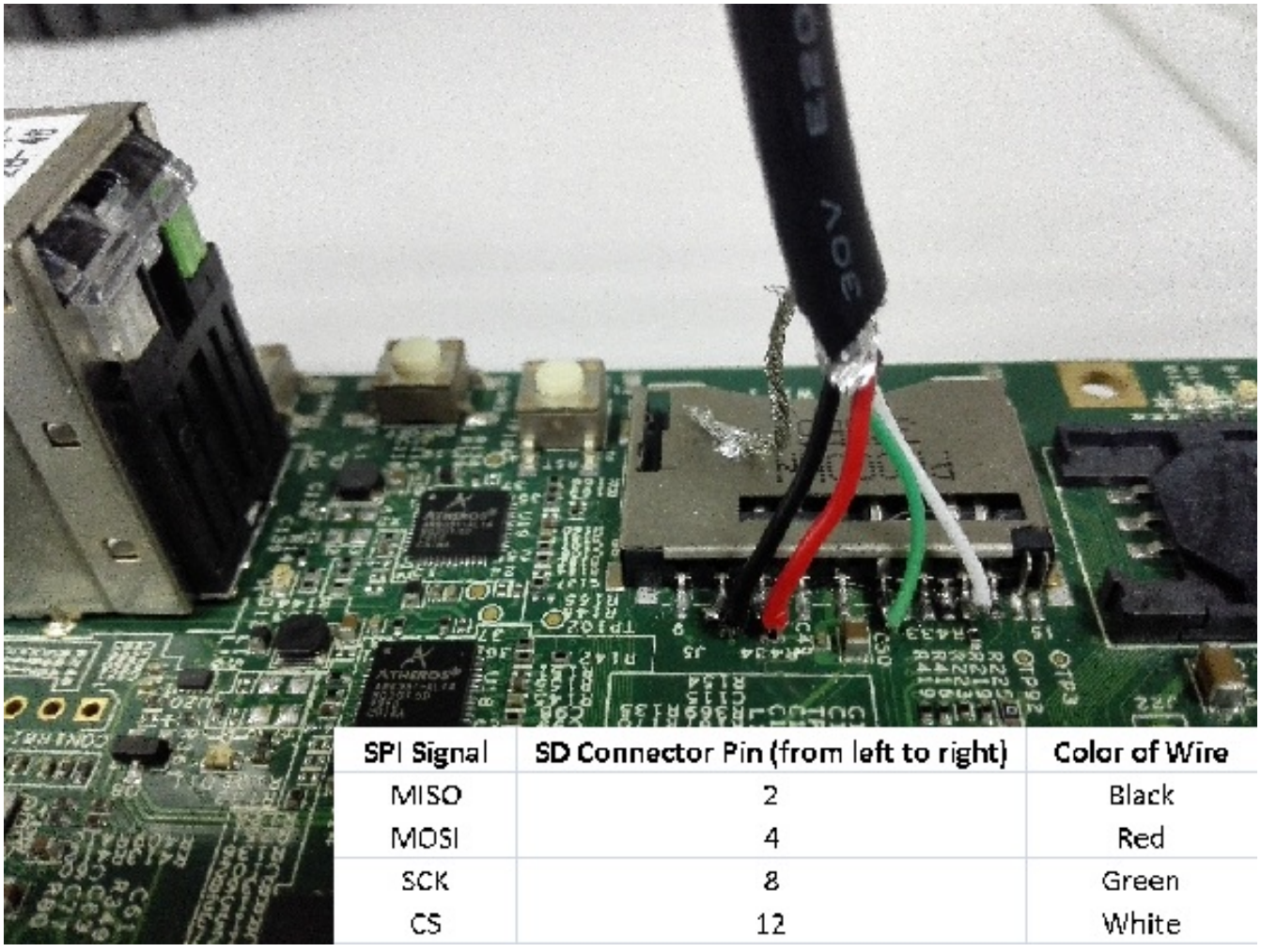
17.2.1.3 Software requirements

- The master project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/ecspi/ecspi_polling/master/<toolchain>.
- The slave project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/ecspi/ecspi_polling/slave/<toolchain>.

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. This example requires two separate boards. You can connect ECSPI4 signals using shielded wire like this:



SPI Signal	SD Connector Pin (from left to right)	Color of Wire
MISO	2	Black
MOSI	4	Red
SCK	8	Green
CS	12	White

Figure 17.2.1: Connecting ECSPI4 signals using shielded wire (1)

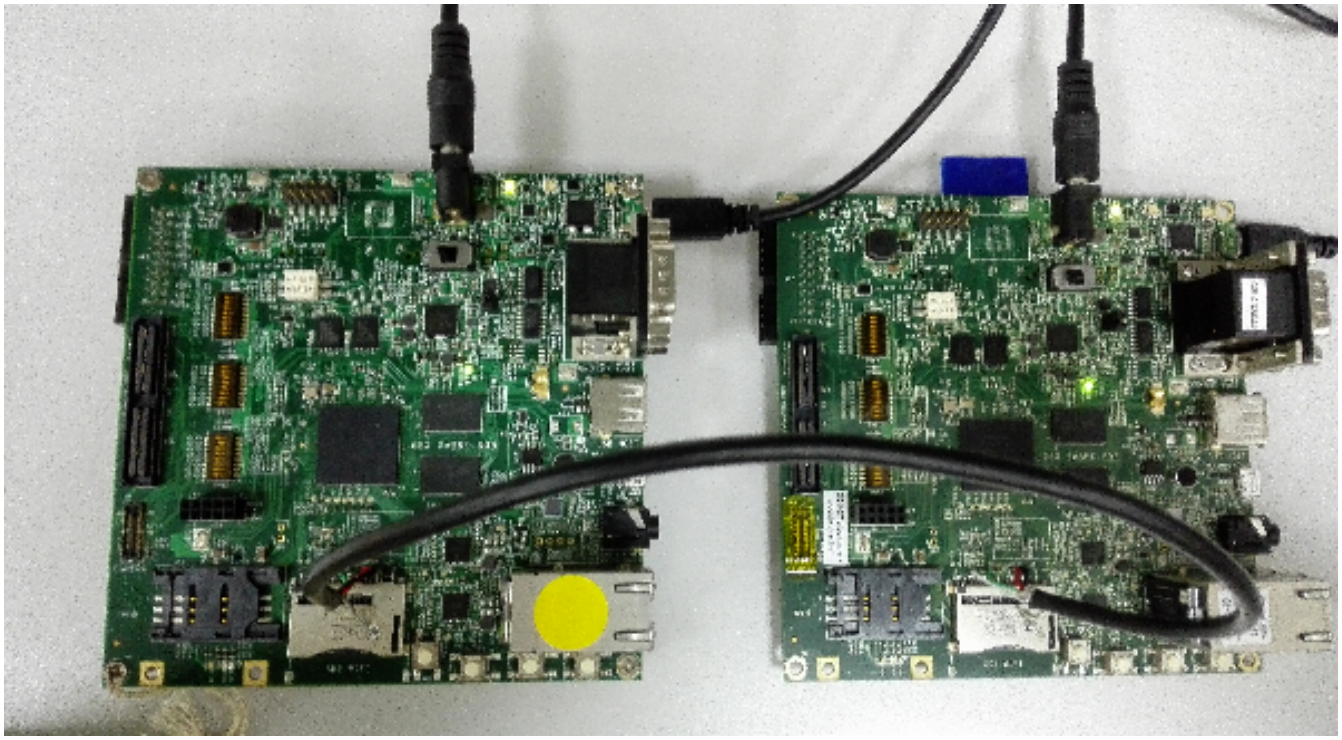


Figure 17.2.2: Connecting ECSPI4 signals using shielded wire (2)

Note: User need to ensure that the cable connected 2 boards should be shielded and as short as possible.

17.2.1.4.2 Prepare the demo

1. Connect two micro USB cable for each PC host and Debug UART port(J16) on the i.MX 6SoloX SABRE-SD board.
2. Open two 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. Connect these 2 board to ECSPI4.
4. Load the demo binary to the DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 will print on terminal:

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

MASTER: Transmited data: 2
      : Received data: 0

MASTER: Transmited data: 3
      : Received data: 1

...

MASTER: Transmited data: 20
      : Received data: 18

Example finish!!!
```

The slave board will print on terminal:

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

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

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

...

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

Example finish!!!
```

Chapter 18
EPIT Example

18.1 Overview

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

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

18.2 Supported Platforms

18.2.1 i.MX 6SoloX SABRE-SD board

18.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

18.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/epit/<toolchain>.

18.2.1.4 Getting Started

18.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

18.2.1.4.2 Running the demo

After the boot process succeeds, the ARM Cortex-M4 terminal displays the following information. The frequency and counter value might differ on different hardware, but the ratio should be close to 0.5:

```
EPIT timer will now start
counter/freq ratio should be close to 0.5 ...
        EPIT A freq 12000000, counter 5999990.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999990.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
EPIT example finished...
```

18.2.2 i.MX 6SoloX SABRE-AI board

18.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI 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 6SoloX Release Notes* (document FRTOS1016XRN).

18.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/epit/<toolchain>.

18.2.2.4 Getting Started

18.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

18.2.2.4.2 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 0.5:

```
EPIT timer will now start
counter/freq ratio should be close to 0.5 ...
        EPIT A freq 12000000, counter 5999990.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999990.
        EPIT B freq 16384, counter 8191.
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
```

```
        EPIT A freq 12000000, counter 5999989.
        EPIT B freq 16384, counter 8191.
EPIT example finished...
```

Chapter 19
FlexCAN Loopback Example

19.1 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.

19.2 Supported Platforms

19.2.1 i.MX 6SoloX SABRE-SD board

19.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

19.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/flexcan/flexcan_loopback/<toolchain>.

19.2.1.4 Getting Started

19.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

19.2.2 i.MX 6SoloX SABRE-AI board

19.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 12V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

19.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/flexcan/flexcan_loopback/<toolchain>.

19.2.2.4 Getting Started

19.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Chapter 20
FlexCAN Network Example

20.1 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.

20.2 Supported Platforms

20.2.1 i.MX 6SoloX SABRE-SD board

20.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD board
- Personal Computer with USB port

20.2.1.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

20.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/flexcan/flexcan_network/<toolchain>.

20.2.1.4 Getting Started

20.2.1.4.1 Hardware settings

To run this example, connect two boards through the CAN interface: The CAN1 DB-9 connector on the i.-MX 6SoloX SABRE-SD board is used as the CAN interface. PIN2 connects to CANL and PIN7 connects to CANH:

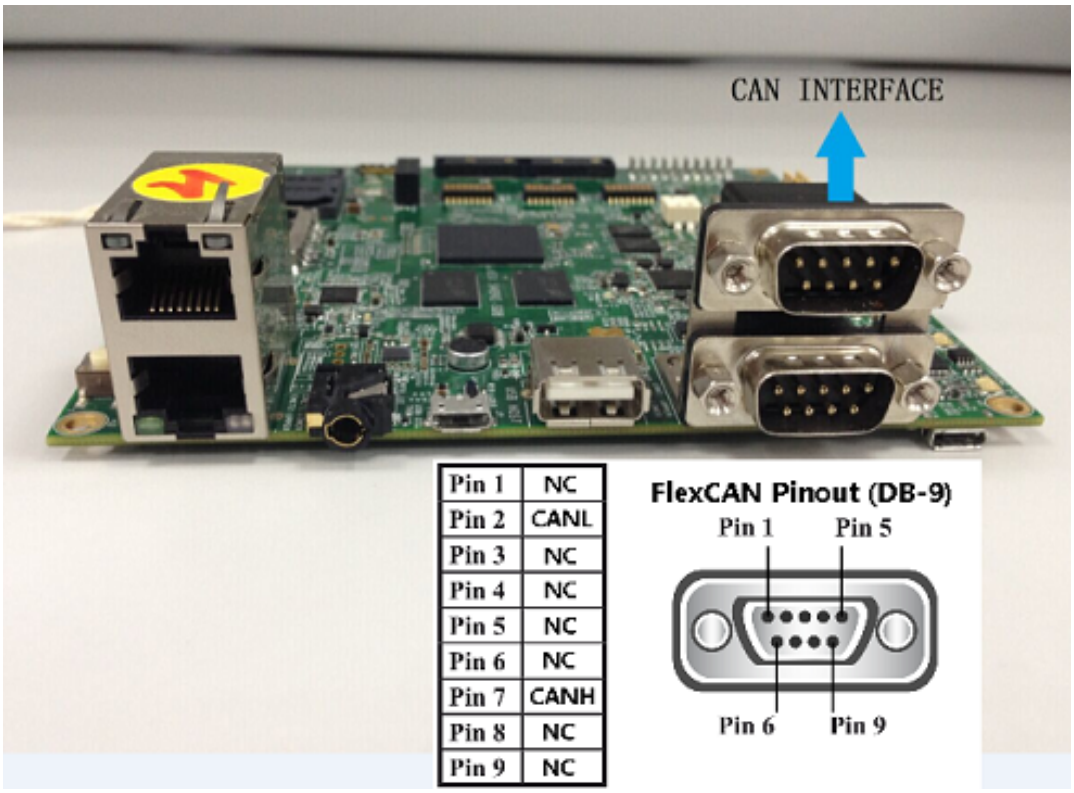


Figure 20.2.1: Connecting two boards through the CAN interface

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

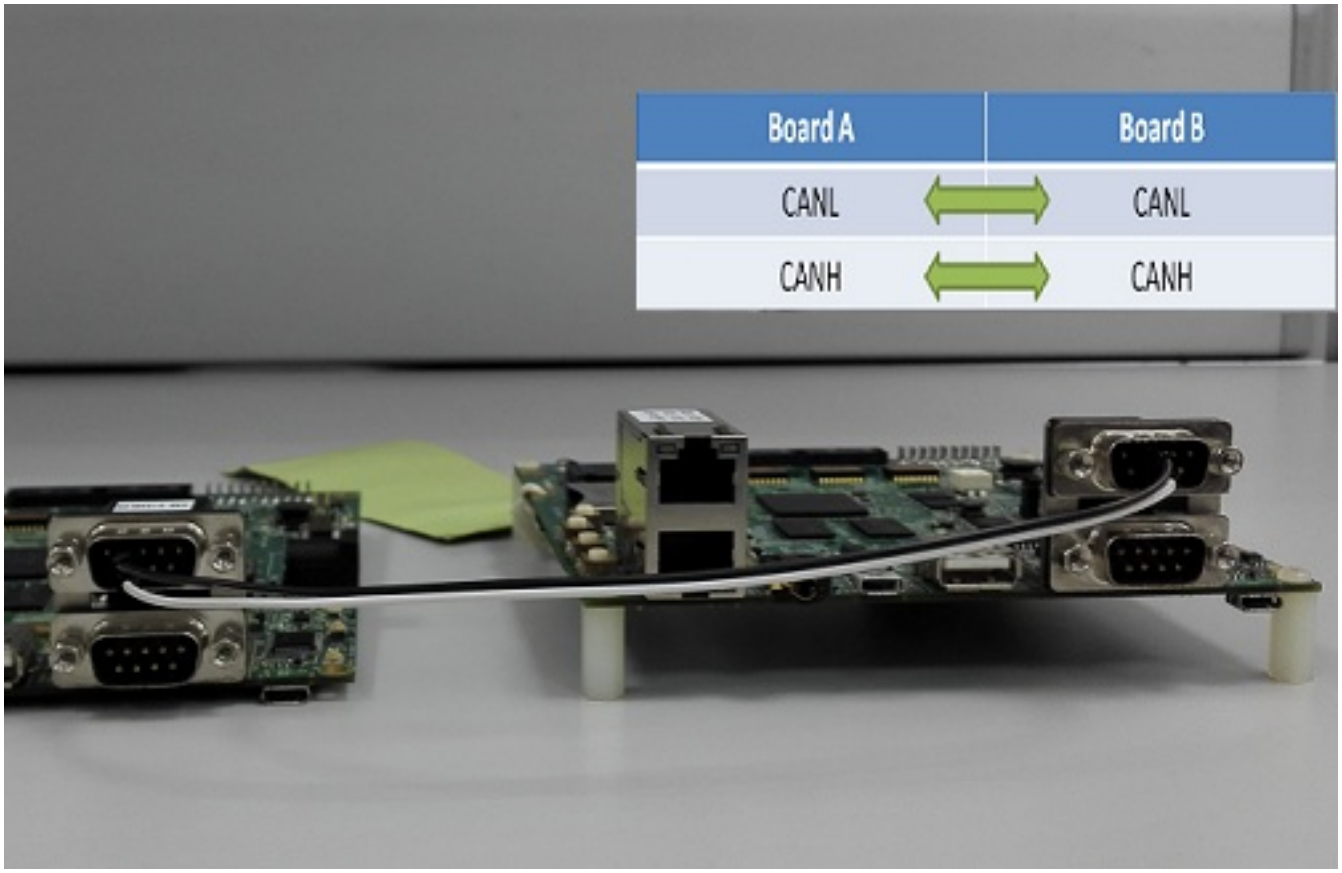


Figure 20.2.2: Connecting two boards to the CAN Bus through the DB-9 connector

20.2.1.4.2 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(J16) on the i.MX 6SoloX 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 DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

20.2.1.4.3 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 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
```

20.2.2 i.MX 6SoloX SABRE-AI board

20.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 12V DC adapter
- i.MX 6SoloX SABRE-AI main board
- i.MX 6SoloX SABRE Automotive Expansion 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 6SoloX Release Notes* (document FRTOS1016XRN).

20.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/flexcan/flexcan_network/<toolchain>.

20.2.2.4 Getting Started

20.2.2.4.1 Hardware settings

To run this example, connect two boards through the CAN interface: The CAN1 DB-9 connector on the i.MX 6SoloX SABRE-AI board is used as the CAN interface. PIN2 connects to CANL and PIN7 connects to CANH:

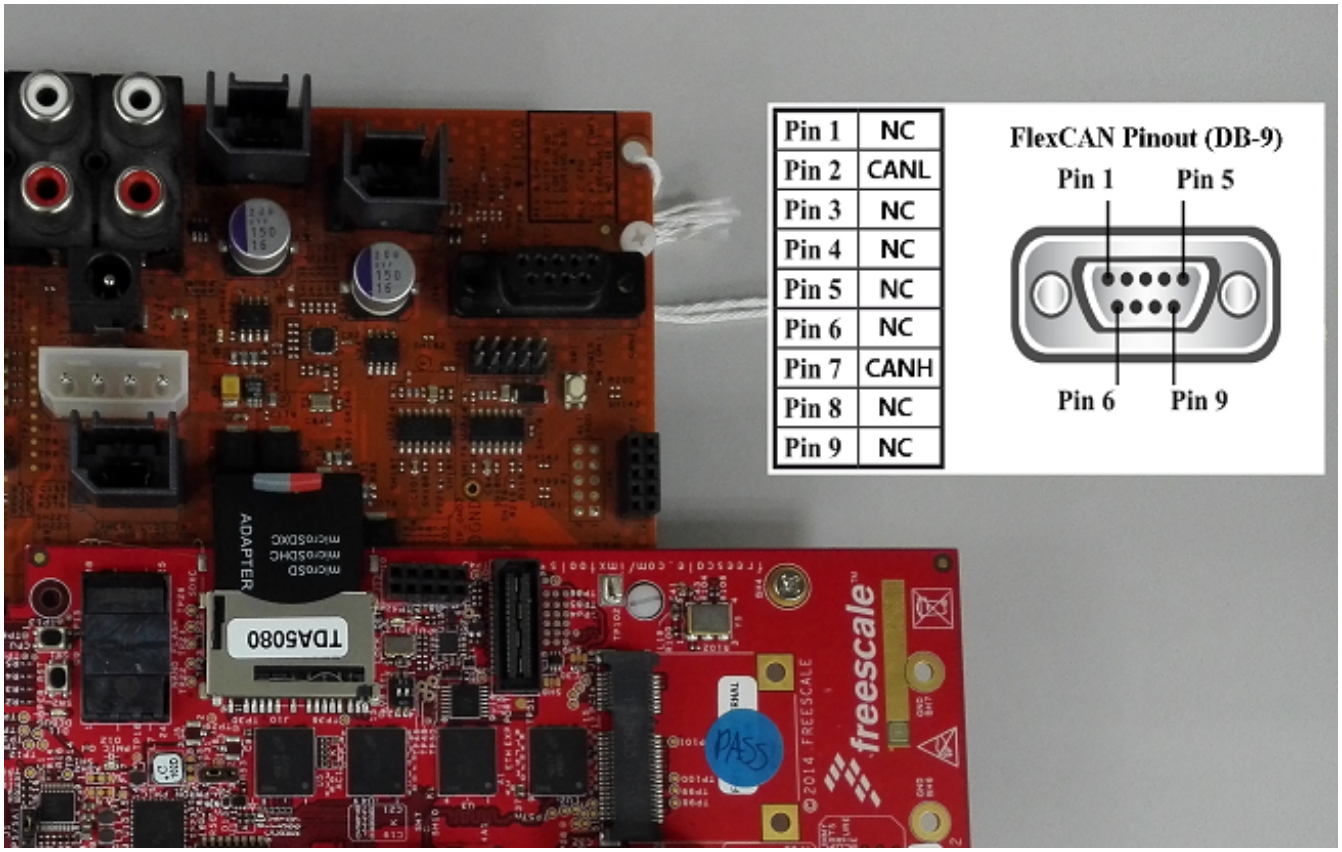


Figure 20.2.3: Connecting two boards through the CAN interface

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

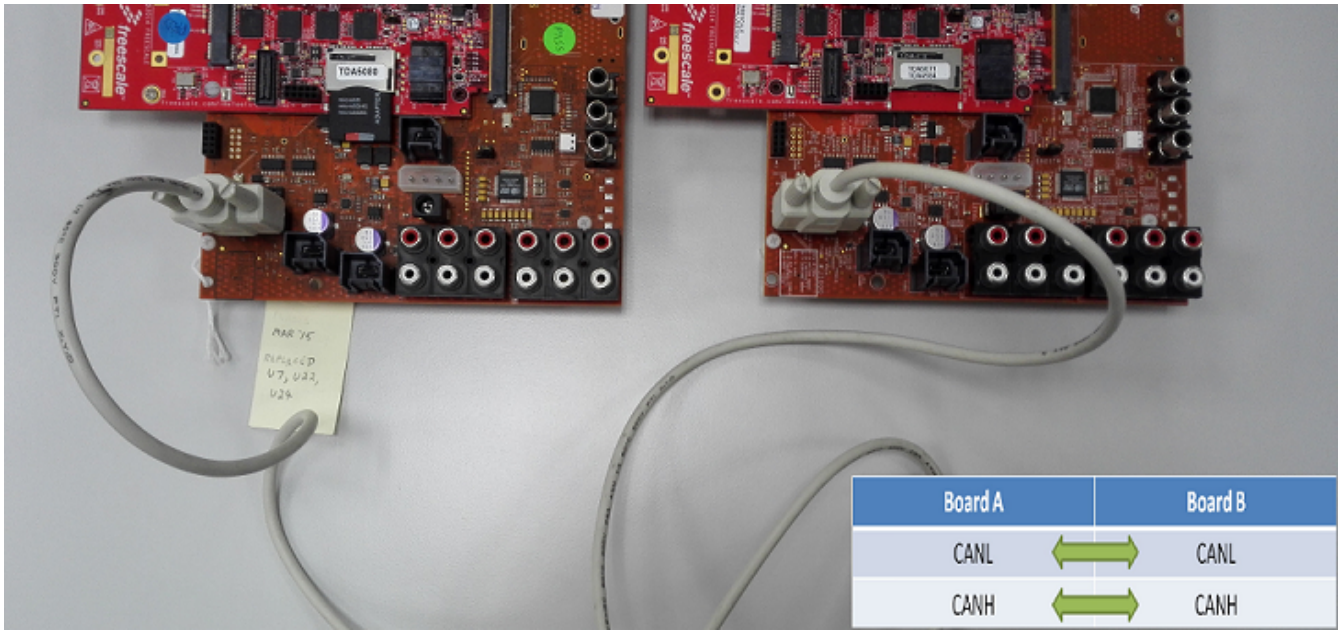


Figure 20.2.4: Connecting two boards to the CAN Bus through the DB-9 connector

20.2.2.4.2 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(J16) on the i.MX 6SoloX SABRE-AI 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 DDR first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

20.2.2.4.3 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 will be back to 0x0.

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

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

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

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

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

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

Chapter 21
GPIO Example

21.1 Overview

This example application demonstrates how to use the GPIO driver to access LEDs or Buttons on the board. On the i.MX 6SoloX SABRE-SD board, the application support the GPIO key function. And on the i.MX 6SoloX SABRE-AI board, the application switches the LED on board.

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

21.2 Supported Platforms

21.2.1 i.MX 6SoloX SABRE-SD board

21.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

21.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/gpio_imx/<toolchain>.

21.2.1.4 Getting Started

21.2.1.4.1 Prepare the demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

21.2.1.4.2 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 the 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:

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

21.2.2 i.MX 6SoloX SABRE-AI board

21.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX

- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI 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 6SoloX Release Notes* (document FRTOS1016XRN).

21.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/gpio_imx/<toolchain>.

21.2.2.4 Getting Started

21.2.2.4.1 Prepare the demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

21.2.2.4.2 Running the demo

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

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

```
============ Use key to simulate GPIO button ==============
Input any data from terminal 3 times to continues.
```

Input the any data from terminal 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
```

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

```
Button pressed 1 times
Button pressed 2 times
Button pressed 3 times
Button pressed 4 times
Button pressed 5 times
Button pressed 6 times
Button pressed 7 times
Button pressed 8 times
Button pressed 9 times
```

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

Chapter 22
I2C Interrupt Example

22.1 Overview

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

Programming on MMA8451Q acceleration sensor through I2C bus and read the 3-Axis acceleration of gravity.

22.2 Supported Platforms

22.2.1 i.MX 6SoloX SABRE-SD board

22.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD board
- Personal Computer with USB port

22.2.1.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

22.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/i2c_imx/i2c_interrupt_sensor_imx6sx/<toolchain>.

22.2.1.4 Getting Started

22.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

22.2.1.4.2 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=-0.013g Y= 0.026g Z=-0.982g
2G MODE: X=-0.019g Y= 0.020g Z=-0.986g
2G MODE: X=-0.013g Y= 0.021g Z=-0.977g
2G MODE: X=-0.017g Y= 0.020g Z=-0.985g
2G MODE: X=-0.020g Y= 0.022g Z=-0.982g
2G MODE: X=-0.021g Y= 0.018g Z=-0.980g
2G MODE: X=-0.018g Y= 0.022g Z=-0.981g
2G MODE: X=-0.018g Y= 0.026g Z=-0.992g
2G MODE: X=-0.019g Y= 0.020g Z=-0.979g
2G MODE: X=-0.020g Y= 0.021g Z=-0.980g

Example finished!!!
```

22.2.2 i.MX 6SoloX SABRE-AI board

22.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX

- Micro USB cable
- 12V DC adapter
- i.MX 6SoloX SABRE-AI main board
- i.MX 6SoloX SABRE Automotive Expansion board
- Personal Computer with USB port

22.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

22.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/i2c_imx/i2c_interrupt_sensor/<toolchain>.

22.2.2.4 Getting Started

22.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

22.2.2.4.2 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 :

```
++++++++++++++++ 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.

[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= 0.004g Y= 0.034g Z=-0.982g
2G MODE: X= 0.007g Y= 0.033g Z=-0.988g
2G MODE: X= 0.006g Y= 0.034g Z=-0.983g
2G MODE: X= 0.005g Y= 0.034g Z=-0.982g
2G MODE: X= 0.003g Y= 0.037g Z=-0.985g
2G MODE: X= 0.004g Y= 0.032g Z=-0.981g
2G MODE: X= 0.010g Y= 0.036g Z=-0.982g
2G MODE: X= 0.009g Y= 0.035g Z=-0.982g
2G MODE: X= 0.009g Y= 0.034g Z=-0.983g
2G MODE: X= 0.006g Y= 0.035g Z=-0.988g

Example finished!!!
```

Chapter 23
I2C Polling Example

23.1 Overview

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

Programming on MMA8451Q acceleration sensor through I2C bus and read the 3-Axis acceleration of gravity.

23.2 Supported Platforms

23.2.1 i.MX 6SoloX SABRE-SD board

23.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

23.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/i2c_imx/i2c_polling_sensor_imx6sx/<toolchain>.

23.2.1.4 Getting Started

23.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

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 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 fast read and active mode
[4].Acc sensor WHO_AM_I check... OK
[5].Acquire 10 samples from Acc sensor
2G MODE: X=-0.013g Y= 0.026g Z=-0.982g
2G MODE: X=-0.019g Y= 0.020g Z=-0.986g
2G MODE: X=-0.013g Y= 0.021g Z=-0.977g
2G MODE: X=-0.017g Y= 0.020g Z=-0.985g
2G MODE: X=-0.020g Y= 0.022g Z=-0.982g
2G MODE: X=-0.021g Y= 0.018g Z=-0.980g
2G MODE: X=-0.018g Y= 0.022g Z=-0.981g
2G MODE: X=-0.018g Y= 0.026g Z=-0.992g
2G MODE: X=-0.019g Y= 0.020g Z=-0.979g
2G MODE: X=-0.020g Y= 0.021g Z=-0.980g

Example finished!!!
```

23.2.2 i.MX 6SoloX SABRE-AI board

23.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX

- Micro USB cable
- 12V DC adapter
- i.MX 6SoloX SABRE-AI main board
- i.MX 6SoloX SABRE Automotive Expansion board
- Personal Computer with USB port

23.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

23.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/i2c_imx/i2c_polling_sensor/<toolchain>.

23.2.2.4 Getting Started

23.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

23.2.2.4.2 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 :

```
++++++++++++++++ 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.

[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= 0.004g Y= 0.034g Z=-0.982g
2G MODE: X= 0.007g Y= 0.033g Z=-0.988g
2G MODE: X= 0.006g Y= 0.034g Z=-0.983g
2G MODE: X= 0.005g Y= 0.034g Z=-0.982g
2G MODE: X= 0.003g Y= 0.037g Z=-0.985g
2G MODE: X= 0.004g Y= 0.032g Z=-0.981g
2G MODE: X= 0.010g Y= 0.036g Z=-0.982g
2G MODE: X= 0.009g Y= 0.035g Z=-0.982g
2G MODE: X= 0.009g Y= 0.034g Z=-0.983g
2G MODE: X= 0.006g Y= 0.035g Z=-0.988g

Example finished!!!
```

Chapter 24
UART Interrupt Example

24.1 Overview

This UART example demonstrates the UART driver working with interrupt.

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

24.2 Supported Platforms

24.2.1 i.MX 6SoloX SABRE-SD board

24.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-SD board
- Personal Computer with USB port

24.2.1.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

24.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/uart_imx/uart_interrupt/<toolchain>.

24.2.1.4 Getting Started

24.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

24.2.1.4.2 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.

24.2.2 i.MX 6SoloX SABRE-AI board

24.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main board
- Personal Computer with USB port

24.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

24.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/uart_imx/uart_interrupt/<toolchain>.

24.2.2.4 Getting Started

24.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

24.2.2.4.2 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.

Chapter 25
UART Polling Example

25.1 Overview

This UART example demonstrates the UART driver working with polling.

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

25.2 Supported Platforms

25.2.1 i.MX 6SoloX SABRE-SD board

25.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

25.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/uart_imx/uart_polling/<toolchain>.

25.2.1.4 Getting Started

25.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

25.2.1.4.2 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.

25.2.2 i.MX 6SoloX SABRE-AI board

25.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main board
- Personal Computer with USB port

25.2.2.2 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 6SoloX Release Notes* (document FRTOS1016XRN).

25.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/uart_imx/uart_polling/<toolchain>.

25.2.2.4 Getting Started

25.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

25.2.2.4.2 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.

Chapter 26
WDOG Example

26.1 Overview

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

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

26.2 Supported Platforms

26.2.1 i.MX 6SoloX SABRE-SD board

26.2.1.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX 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 6SoloX Release Notes* (document FRTOS1016XRN).

26.2.1.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_sdb_m4/driver_examples/wdog_imx/<toolchain>.

26.2.1.4 Getting Started

26.2.1.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

26.2.1.4.2 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 coninues for ever.

26.2.2 i.MX 6SoloX SABRE-AI board

26.2.2.1 Hardware requirements

- SD Card with U-Boot for i.MX 6SoloX
- Micro USB cable
- 5V DC adapter
- i.MX 6SoloX SABRE-AI main 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 6SoloX Release Notes* (document FRTOS1016XRN).

26.2.2.3 Software requirements

- The project files are in: <BSP_Install>/examples/imx6sx_ai_m4/driver_examples/wdog_imx/<toolchain>.

26.2.2.4 Getting Started

26.2.2.4.1 Prepare the Demo

1. Connect a micro USB cable between the PC host and the Debug UART port(J16) on the i.MX 6SoloX SABRE-AI 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 first and then copy them 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 6SoloX* (document FRTOS6XGSUG).

26.2.2.4.2 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...
```

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 coninues for ever.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: FRTOS6X101DAUG
Rev. 0
07/2016