MAPS-K64 Demo Applications User's Guide

Freescale Semiconductor, Inc.

1.0.0
Feb 2015

Contents

Chapter 7
I2C EEPROM Demo

Chapter 8
Acceptance Test Demo

Chapter 9
Flash Demo

Chapter 14
Hardware Timer Demo

Chapter 15
I2C Demo with RTOS

Chapter 16
LPTMR Demo

Section number	Title	Page

Chapter 20
UDP Echo Demo on lwIP TCP/IP Stack

Chapter 21
RTC Function Demo

Chapter 22
SAI Demo

Chapter 23
SD Card Demo

Chapter 24
SPI Flash Demo

Chapter 25
Watchdog Timer Reset Demo

Chapter 1
ADC Hardware Trigger Demo

This demo application demonstrates how to use the ADC drivers with different hardware triggers.

1.1 Overview

This is an ADC demo application which shows how to use different hardware trigger sources to handle the ADC hardware trigger function. These trigger sources are supported:

- PIT (Periodic Interrupt Timer)
- PDB (Programmable Delay Block)
- LPTMR (Low Power Timer)

Demo Location

- PIT: demos/adc_hw_trigger/pit
- PDB: demos/adc_hw_trigger/pdb
- LPTMR: demos/adc_hw_trigger/lptmr

1.1.1 Trigger by PIT

The Periodic Interrupt Timer (PIT) is a period timer source and the ADC hardware trigger event. Because the PIT trigger event can only be used to trigger one of the ADC channels (channel 0 or 1), this demo uses PIT as a trigger source for the ADCx channel 0. The PIT triggers the ADC in a fixed frequency and the demo gets the ADC conversion result in the ADC Conversion Complete (COCO) interrupt.

1.1.2 Trigger by PDB

The Programmable Delay Block (PDB) is a continuous trigger event for ADC. It uses the software trigger as the first trigger input event and turns on the PDB continuous mode to generate a period trigger source. Because the PDB can trigger different channels inside one ADC instance, this demo shows the Ping-Pong triggering which occurs by sampling the channel 0/1 with the PDB Pre-trigger A/B channel.

1.1.3 Trigger by LPTMR

The Low Power Timer (LPTMR) is a period timer source and the ADC hardware trigger event. Because the LPTMR trigger event can only be used to trigger one of the ADC channels (channel 0 or 1), this demo uses the LPTMR as a trigger source for the ADCx channel 0. The LPTMR triggers the ADC in a fixed frequency and the demo gets the ADC conversion result in the ADC Conversion Complete (COCO) interrupt.

1.1.4 Input signal for ADC

Use the DAC module to generate a sine wave as the ADC input on the DAC0_OUT pin. Because the DAC0_OUT is internally connected to the ADC0_SE23 (DAC0_OUT is a source of ADC0_SE23), there is no need to connect any external signals for this demo.

This demo samples the input digital signal from the ADC0_SE23 pin and records each sample point with the appropriate amplitude. After 2 period samples are complete, it prints out the rough shape of the signal wave on the debug console like a primitive oscilloscope.

1.2 Getting Started

1.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

1.3 Run the demo

1. Select and open one project from the three projects available: adc_pit_trigger, adc_lptmr_trigger and adc_pdb_trigger.
2. Open the UART console on a PC.
3. Download and run the program on the target.
4. The signal waveform is displayed on the console.

1.4 Customization Options

This demo application is customizable to show different kinds of input signal waves.

1.4.1 Default configurations

The configuration macro is located in the adc_hw_trigger.h header file.

1.4.1.1 ADC configurations

1. Use ADC0 instance.

2. Use ADC_SE23 input pin as sample pin.
3. Use VREFH/L as reference voltage.

1.4.1.2 Sample frequency

The default sample rate is 20 Hz $*$ 100 / 2, which enables the demo application to get 100 samples per two periods. To change the sample rate, see the next section.

1.4.2 Configure the number of samples

Printing of the signal wave shape depends on the console size. A console can be 100x40. To get the best printing effect, align the number of samples to the console column numbers and convert the amplitude range to the [0, row - 1] range. The console column number should be same as sample numbers. Configuring the number of samples means configuring the console column size:

```
#define CHART_ROWS 30U // chart row for sampled data
#define CHART_COLS 100U // chart column for sampled data
#define NR_SAMPLES 100U // number of samples in one period
```

1.4.3 Configure the signal frequency

Change the following macro to configure the desired frequency in Hz units.

```
#define INPUT_SIGNAL_FREQ 20U // in Hz
```

1.4.4 Configure the ADC instance

Change the ADC_INST macro to configure the ADC instance you want to use.

```
#define ADC_INST 0U  // ADC instance
```

1.4.5 Configure the ADC input pin

If you do not use the DAC0_OUT as a input signal, disable the macro in the project:

```
//#USE_DAC_OUT_AS_SOURCE
```

After disabling the DAC output, configure one ADC input source pin to get the signal:

```
#define ADC_INPUT_CHAN    23U // default input signal channel
```

Chapter 2
ADC Low Power Demo

This demo application demonstrates how to use the ADC drivers in low power modes.

2.1 Overview

The ADC Low Power Demo project is a demonstration program that uses the KSDK software. The microcontroller is set to a very low power stop (VLPS) mode, and every 500 ms an interrupt wakes up the ADC module and takes the current temperature of the microcontroller. While the temperature remains within boundaries, both LEDs are off. If the temperature is higher than average, a red LED comes on. If it is lower, a blue LED comes on. This demo provides an example to show how ADC works during a VLPS mode and a simple debugging, "golden" project.

Demo Location

- demos/adc_low_power

2.2 Getting Started

The ADC Low Power project is designed to work with the Tower System or in a stand alone setting.

2.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

2.2.2 Run the demo

1. Set your target board in a place where the temperature is constant.
2. Press the reset button on your development board.
3. "ADC LOW POWER DEMO" message and some instructions should be displayed on the terminal.
4. Wait until the red or white LED light turns on.
5. Increment or decrement the temperature to see the changes.

Chapter 3
ADC Potentiometer Demo

This demo application demonstrates how to sample the voltage output by the potentiometer.

3.1 Overview

This application demonstrates how to configure the ADC as Single-End mode and differential mode and sample the voltage output by the potentiometer.

Demo Location

- demos/adc_potentiometer

3.2 Getting Started

3.2.1 Hardware Settings

This table shows the connections that are required for the supported platforms:

Platform	Jump Setting
MAPS-K64	JP10 1-2
	JP11 1-2

3.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

3.3 Run the demo

1. Power on the boards, download the application into Flash, and run.
2. Open the PC console to connect to the board.
3. When successfully connected, the output is like as follows:

```
Potentiometer demo start...

Select Potentiometer mode:
1: Potentiometer -- Single End(RV2).
2: Potentiometer -- Single End(RV1)JP10 connect first).
3: Potentiometer -- Difference.
Please enter your choice (1 - 3):
```

4. select the choice, then the voltage sampled would output to the console, like this "voltage: 3.2984V". If the choice is not 1, 2 and 3, the "Invalid choice." would be output to the console.

Chapter 4
DAC Speaker Demo

This demo application demonstrates how to control the speaker by DAC output.

4.1 Overview

This application demonstrates how to configure the DAC and set the output on the DAC to control the sound of speaker.

Demo Location

- demos/dac_speaker

4.2 Getting Started

4.2.1 Hardware Settings

This table shows the connections that are required for the supported platforms:

Platform	Jump Setting
MAPS-K64	JP18 1-2

4.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

4.3 Run the demo

1. Power on the boards, download the application into Flash, and run.
2. Open the PC console to connect to the board.
3. When successfully connected, the output is like as follows:

```
DAC output to control the sound of speaker.
```

```
Input value(0--2047):
```

4. Input the value, for example: 1024, then the speaker will sound. Input the different value, then the loud of speaker is different.

Chapter 5
DSPI eDMA Demo

This demo application demonstrates how to use the DSPI and eDMA drivers.

5.1 Overview

This application demonstrates how to configure the DSPI transfers using eDMA in both send and receive modes.

Demo Location

- demos/dspi_edma_demo

5.2 Getting Started

5.2.1 Hardware Settings

Make these connections between the signals listed by using external wires . This demo uses the same board for both master and slave connections.

MAPS-K64:

	Master		Connects To	Slave	
Signal	**Pin Name**	**Board Location**		**Pin Name**	**Board Location**
SS	PTD0	CN8 - 15	->	PTB10	CN9 - 11
SCK	PTD1	CN8 - 16	->	PTB11	CN9 - 12
Data In	PTD2	CN8 - 17	->	PTB17	CN9 - 14
Data Out	PTD3	CN8 - 18	->	PTB16	CN9 - 13

5.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control

3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

5.3 Run the demo

This output appears in the terminal window:

```
----------------------------------------------------------------
----------------------------------------------------------------
DSPI eDMA Demo
----------------------------------------------------------------
DSPI Transfers using eDMA
----------------------------------------------------------------
----------------------------------------------------------------~


----------------------------------------------------------------
----------------------------------------------------------------
Configuration:
----------------------------------------------------------------
Setting         | Value
:---------------------------- | :----------------------------
Terminal Baud:  | 115200 bps
 DSPI Bit rate: | User defined (3 kHz to 2 MHz)
 DSPI Master:   | DSPI0
 DSPI Slave:    | DSPI1
```

The user enters the desired bit rate when this prompt appears:

```
Enter a SPI clock frequency between 3,000 Hz & 2,000,000 Hz below.
NOTE: If the number is less than 1,000,000 press enter after typing the number.~


->
```

Upon successful completion of the demo this message appears in the terminal window:

```
Demo Complete.~
```

```
128 loops completed.~

g_slaveRxBuffer~

00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210
00000123        00004567        000089AB        0000CDEF
0000FEDC        0000BA98        00007654        00003210~

~

Success! Slave received all the bytes from the Master.~

masterRxBuffer~

00000100        00000302        00000504        00000706
00000908        00000B0A        00000D0C        00000F0E
00001110        00001312        00001514        00001716
00001918        00001B1A        00001D1C        00001F1E
00002120        00002322        00002524        00002726
00002928        00002B2A        00002D2C        00002F2E
00003130        00003332        00003534        00003736
00003938        00003B3A        00003D3C        00003F3E
00004140        00004342        00004544        00004746
00004948        00004B4A        00004D4C        00004F4E
00005150        00005352        00005554        00005756
00005958        00005B5A        00005D5C        00005F5E
00006160        00006362        00006564        00006766
00006968        00006B6A        00006D6C        00006F6E
00007170        00007372        00007574        00007776
```

```
00007978        00007B7A        00007D7C        00007F7E~


Final value of g_slaveTxCount sent:  0x7F~


Success! The Master received all the bytes from the slave correctly.~


End of demo.
```

5.4 Key Functions

void dspi_edma_master_setup(uint8_t instance, uint32_t baudRateHz, uint8_t transferSizeBits)

This function takes in the DSPI module instance, the desired data rate of DSPI transfers, the frame size of the data transfer, and initializes the DSPI master instance.

Parameters

instance	DSPI module instance
baudRateHz	Pass in the desired baud rate of DSPI transfers in Hz.
transferSizeBits	Pass data frame size (8 or 16 bit)

void dspi_edma_slave_setup(uint8_t instance, uint8_t transferSizeBits)

This function takes in the DSPI module instance, the frame size of the data transfer, and initializes the DSPI slave instance.

Parameters

instance	DSPI module instance
transferSizeBits	Pass data frame size (8 or 16 bit)

dspi_status_t data_source(uint8_t ∗ sourceWord, uint32_t instance)

This is a callback function for the DSPI slave which transmits data from the slave. In this application, the function generates the slave data out which is a count.

Parameters

sourceWord	8-bit data variable to be passed to slave PUSHR register.
instance	Instance of the DSPI module.

void on_error(dspi_status_t error, uint32_t instance)

This is a callback function for the DSPI slave which handles errors. In this application, the function sets the error flag to signal the end of the demonstration.

Parameters

error	Uses dspi_status_t value given by driver interrupt handler.
instance	Instance of the DSPI module.

void setup_edma_loop(edma_loop_setup_t ∗loopSetup, uart_state_t ∗uart)

This function configures an eDMA transfer loop by using a loopSetup structure. If a valid uart_state_t is passed, the function prints out the TCD for that loop.

Parameters

loopSetup	Data structure defined by user containing all parameters for the eDMA loop.
uart	Pointer to a valid uart_state_t for debug printing.

void disable_edma_loop(edma_loop_setup_t ∗loopSetup, uart_state_t ∗uart)

This function disables the user-configured eDMA transfer loop. It also frees memory allocated by the eDMA transfer loop. If a valid uart_state_t is passed to it, it prints out the TCD for that loop.

Parameters

loopSetup	Data structure defined by user containing all parameters for the eDMA loop.
uart	Pointer to a valid uart_state_t for debug printing.

void ∗mem_align(size_t ptrSize, uint32_t alignment)

This function performs the aligned data memory allocation and is useful when the mem_align is not available.

Parameters

ptrSize	size_t variable to pass size of memory to be allocated
alignment	uint32_t variable to pass byte size to align data with

Returns

pointer to aligned & allocated memory

void free_align(void *ptr)

This function frees the memory allocated by the mem_align.

Parameters

ptr	Pointer that has been allocated with the mem_align.

Chapter 6
SPI LCD Demo

This demo application demonstrates how to use lcd12864 by SPI interface.

6.1 Overview

The dspi_lcd demo project shows how to use lcd12864 by SPI. The lcd12864 controller is ST7565S.

Demo Location

- demos/dspi_lcd

6.2 Getting Started

The Dspi_lcd Demo example code shows how to use lcd12864 through SPI interface.

6.2.1 Prepare the Demo

1. Download the program to the target board.
2. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

6.2.2 Run the demo

1. Download and run the dspi_lcd code to the board.
2. See if the char can be display on lcd successful.

Chapter 7
I2C EEPROM Demo

This demo application demonstrates how to use I2C driver to communicate with AT24C02 EEPROM.

7.1 Overview

This is a EEPROM communication demo which demonstrates the communication between MCU with EEPROM IC AT24C02. The demo will write data into 256 bytes space in the EEPROM chip and then read it back.

Demo Location

- demos/eeprom_demo

7.2 Getting Started

7.2.1 Hardware configuration

Running this demo requires both MAPS-K64 and MAPS-DOCK board connected together.

MAPS-DOCK module configuration

1. Short JP4(1-2), J4(3-4) to enable I2C connection.

7.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

7.3 Run the demo

1. For a successfully run, the read back data is the same as write data. If there is error communication with I2C, then you will see write and read error messages.

Chapter 8
Acceptance Test Demo

This demo application demonstrates how to test the module hardware.

8.1 Overview

The fat demo project shows how to test the module hardware.

Demo Location

- demos/fat

8.2 Getting Started

The fat Demo example code shows how to test module hardware, including "SRAM","USB", "SPI-FLASH", "EEPROM", "SD" modules.

8.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 9600 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

8.2.2 Run the demo

1. Download and run the fat code to the board.
2. See the uart information and spilcd screen which shows the test result.

This demo application demonstrates how to test the module hardware.

8.3 Overview

The fat_dock demo project shows how to test the DOCK board module hardware.

Demo Location

- demos/fat_dock

8.4 Getting Started

The fat Demo example code shows how to test module hardware, including "SRAM","USB", "SPI-FLASH", "EEPROM", "SD" modules.

8.4.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 9600 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

8.4.2 Run the demo

1. Download and run the fat code to the board.
2. See the uart information and spilcd screen which shows the test result.

Chapter 9
Flash Demo

This demo application demonstrates how to use the Flash drivers.

9.1 Overview

The Flash demo project shows how to erase, program, and performs swap (if available) on the Flash module. Targets exist for both Flash and SRAM memory space. The features include:

1. Full Flash erase and programming support
2. Flash Erase by block or sector, including margin read options
3. Programming region defined by user
4. Flash verify and checksum support
5. Flash Swap (if supported on device)

Demo Location

- demos/flash_demo

9.2 Getting Started

The Flash Demo example code shows how to erase and program the Flash content and use the swap feature if it is supported on the device.

9.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

9.3 Commands/Directions

1. Select the Debug target from within the IDE and build the project selected for the target hardware. The default Debug target runs from flash and demonstrates the Swap feature for devices that support Swap (K64F).

2. Connect one end of the USB cable to a PC host and the other end to the OpenSDA connector on the board.
3. Open Terminal program such as TeraTerm, Putty, or Hyperterminal.
4. Configure the Terminal program to select the OpenSDA COMx port for the board using 115200 8N1: 115200 baud, 8 data bits, No parity, 1 Stop bit.
5. Connect to the board with the debugger (download & debug), run the program, and view the Terminal messages for Flash operations being performed.
6. For devices that support Swap, the Flash_Debug target copies (programs) the application that is running from the lower block to the upper block and then issues swap commands.
7. Flash memory blocks are swapped at the next reset. Disconnect debug session and hit the reset button on the board. Note: During swap, memory locations 0x7F000 & 0xFF000 are swapped and displayed on the terminal showing how the memory map changes.
8. For devices that do not support swap, view the terminal messages for Flash operations that are occurring for the demo.
9. Terminal displays the message "Flash Demo Complete!" when finished.
   Note: Callback functions are not currently supported during flash erase or program operations
   Note: For K22F, Flash erase and program operations are not allowed in High-Speed RUN modes. Therefore, the core clock speed is restricted to 80 MHz or less.

Chapter 10
FlexBus LCD Demo

This demo application demonstrates how to use lcd(ILI9341) by FlexBus.

10.1 Overview

The flexbus_lcd demo project shows how to use lcd(ILI9341) by FlexBus.

Demo Location

- demos/flexbus_lcd

10.2 Getting Started

The Dspi_lcd Demo example code shows how to use lcd(ILI9341) through flexbus interface.

10.2.1 Prepare the Demo

1. Download the program to the target board.
2. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

10.2.2 Run the demo

1. Download and run the dspi_lcd code to the board.
2. See if the picture can be display on lcd successful.

Chapter 11
FlexCAN and UART communication Demo

This demo application demonstrates how to use the FlexCAN and UART drivers.

11.1 Overview

This is a FlexCAN and UART communication demo which demonstrates the communication between two boards which is handled by FlexCAN and the UART input/output. On one board, the user inputs characters by using the UART debug terminal and sends the data with the FlexCAN interface. On the other board, the FlexCAN receives the data and prints them to the UART terminal.

Demo Location

- demos/flexcan_uart

11.2 Getting Started

11.2.1 Hardware configuration

1. Connect the two CAN modules through the CAN port CN13(Dock board).

11.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

11.3 Run the demo

1. Select the node ID for each board by typing A/a or B/b followed by Enter.
2. You may now transfer characters by using serial terminals. For more details, see the video in the //video folder.

Chapter 12
FlexTimer PWM Demo

This demo application demonstrates the FlexTimer PWM demo.

12.1 Overview

This application demonstrates the FTM edge-aligned PWM function. It outputs the PWM to control the intensity of the LED.

Demo Location

- demos/ftm_pwm

12.2 Getting Started

Hardware configuration

This table lists which FTM channels and MCU pins control for this demo application. This table also lists which connections should be made (if any) to ensure proper demo operation.

Platform	FTM Instance/Chnl	MCU Pin
MAPS-K64	FTM0 - CH4	PTA7

12.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Find Pin PTA7 and connect to the oscilloscope.
5. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

12.2.2 Run the demo

1. Download and run the ftm_pwm code to the board.

2. Terminal prints the message "Welcome to FTM PWM demo!"
3. Capture Pin PTA7 waveform and check the PWM pulse width changes.

Chapter 13
Hello World Demo

This demo application demonstrates the Hello World demo.

13.1 Overview

The Hello World project is a simple demonstration program that uses the KSDK software. It prints the "Hello World" message to the terminal using the KSDK 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.

Demo Location

- demos/hello_world

13.2 Getting Started

13.2.1 Hardware Settings

The Hello World project does not call for any special hardware configurations. Although not required, the recommendation is to leave the development board jumper settings and configurations in default state when running this demo.

13.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

13.3 Run the demo

This is an example how to run the demo.

```
Hello World!
```

Chapter 14
Hardware Timer Demo

This demo application demonstrates using the hardware timer driver.

14.1 Overview

The Hardware Timer project is a demonstration program to show how to use the Hardware Timer driver. An hwtimer interrupt is created and fires multiple times until it reaches the requested number.

Demo Location

- demos/hwtimer_demo

14.2 Getting Started

14.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

14.2.2 Run the demo

1. Press the reset button on your board.
2. "Hwtimer Example" message is displayed on the terminal.
3. A dot is printed when an hwtimer interrupt occurs until the HWTIMER_DOTS_PER_LINE ∗ HW-TIMER_LINES_COUNT (defined in hwtimer_demo.c) interrupts occur.
4. Finally, the "End" message is displayed.

```
Hwtimer Example
........................................
........................................
End
```

14.3 Customization Options

This demo application is customizable to show different types of hardware timers.

14.3.1 Configure the Hardware Timer Used

Determine which timer the hardware timer driver uses. The ARM core systick timer is used by default.

```
#define HWTIMER_LL_DEVIF    kSystickDevif
```

14.3.2 Configure which clock is used by the hardware timer

Determine which clock source is used by the hardware timer.

```
#define HWTIMER_LL_SRCCLK   kCoreClock
```

14.3.3 Configure which instance of the module is used

Determine which instance of the selected hardware module to use. For the systick timer only '0' is valid. If the PIT is used, use this to select the PIT channel.

```
#define HWTIMER_LL_ID       0
```

14.3.4 Hardware Timer Period

Determine the timer period (in microseconds).

```
#define HWTIMER_PERIOD          100000
```

Chapter 15
I2C Demo with RTOS

This demo application demonstrates the I2C demo on different RTOS.

15.1 Overview

This I2C application demonstrates the SDK Peripheral drivers working on different RTOSes. The application acts as both the I2C master and the slave device on different I2C buses, such as the I2C Master on the I2C0 bus and the I2C Slave on the I2C1 bus. It can run on a single board or on two different boards. When connecting the two I2C buses on one board, the master sends the command using the I2C0 bus to the slave using the I2C1 bus. When connecting the I2C0 bus to the I2C1 bus on the other board, the application running on the first board is a master and sends a command to the other board which acts as a slave. This means that the first board can send a command and get a response from the other board by using the I2C bus. The basic purpose of this demo is:

1. Read the Kinetis chip UID (low 32bits) from the slave board
2. Read the Kinetis chip internal temperature from the slave board
3. Control the LEDs on the slave board

The application creates three different tasks to handle events concurrently:

1. Master task: responds to the user interface interaction, runs as a I2C master, and acts as a simple UI. It accepts user's commands to read the basic chip UID, chip temperature and control the on board LED, and power mode on the slave.
2. Slave task: responds to the command received from the I2C master and returns the result to the master.
3. ADC sample task: responds to getting the chip temperature in a period.
4. For the bare metal version, the master and slave tasks are separated into two separate projects.

15.2 Supported RTOS

- Freescale MQX™ RTOS
- FreeRTOS
- µC/OS-II
- µC/OS-III
- Bare Metal (no RTOS)

Demo Location

- MQX: demos/i2c_rtos/i2c_rtos_mqx
- FreeRTOS: demos/i2c_rtos/i2c_rtos_freertos
- µC/OS-II: demos/i2c_rtos/i2c_rtos_ucosii
- µC/OS-III: demos/i2c_rtos/i2c_rtos_ucosiii
- BM: demos/i2c_rtos/i2c_rtos_bm_master, demos/i2c_rtos/i2c_rtos_bm_slave

15.3 Getting Started

The I2C RTOS application is designed to work on one single board or two different boards. Note that the bare-metal version only supports two boards.

15.3.1 Build with different RTOS support

Before running this application, build it with the RTOS you want to use. The projects for different RTOSes are differentiated by the workspace file name in the format of i2c_rtos_<rtos>.eww For example, in IAR, the i2c_rtos_ucosii.eww workspace file is the µC/OS-II version of this application. After opening the appropriate workspace, build the ksdk_<rtos>_lib project and build the application project. A binary named i2c_rtos_<rtos>.out is generated.

15.3.2 Hardware configuration

Make the connections between the listed signals by using the external wires.

Freescale MAPS-K64

Shunt JP1 on MAPS-DOCK for I2S signals Shunt JP4 on MAPS-DOCK for I2C signals

15.3.3 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

15.4 Run the demo

This menu displays in the terminal window:

1 - LED One Toggle – LED1 toggles on/off 2 - LED Two Toggle – LED2 toggles on/off 3 - LED Three Toggle – LED3 toggles on/off 4 - LED Four Toggle – LED4 toggles on/off 5 - Read Temperature – Get temperature of client 6 - Read Id – Read client unique id Please enter your choice (1 - 6): Enter your choice (1 - 5):

You can select to toggle the LEDs, read the temperature of the client board, and read the client unique ID.

Chapter 16
LPTMR Demo

This demo application demonstrates the LPTMR demo.

16.1 Overview

The LPTMR (Low Power Timer) project is a simple demonstration program to show how to use the LPTMR driver. It triggers an LPTMR interrupt once every second, and prints out the number of interrupts that have occurred since the program started running.

Demo Location

- demos/lptmr_demo

16.2 Getting Started

16.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

16.3 Run the demo

An LPTMR interrupt occurs every second and prints out to the serial terminal the number of interrupts that have occurred since the program started running. The LPTMR module uses the internal 1 kHz Low Power Oscillator (LPO) as its clock source for this demo.

The output on the serial terminal is as shown:

```
Low Power Timer Example
Started LPTMR
1 2 3 4 5 6 7
```

Chapter 17
HTTP Server Demo on lwIP TCP/IP Stack

This demo application demonstrates the HTTPServer demo on lwIP TCP/IP stack with bare metal SDK or different RTOSes.

17.1 Overview

This is an HTTPServer set up on lwIP TCP/IP stack with bare metal SDK or different RTOSes. The user uses an Internet browser to send a request for connection. The board acts as an HTTP server and sends a Web page back to the PC.

17.2 Supported RTOS

- Freescale MQX™ RTOS
- FreeRTOS
- µC/OS-II
- µC/OS-III
- Bare Metal (no RTOS)

Demo Location

- MQX: demos/lwip_httpserver_demo/httpserver_rtos/httpserver_mqx
- FreeRTOS: demos/lwip_httpserver_demo/httpserver_rtos/httpserver_freertos
- µC/OS-II: demos/lwip_httpserver_demo/httpserver_rtos/httpserver_ucosii
- µC/OS-III: demos/lwip_httpserver_demo/httpserver_rtos/httpserver_ucosiii
- BM: demos/lwip_httpserver_demo/httpserver_bm

17.3 Getting Started

17.3.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

17.3.2 Network Configuration

Configure the IP address of PC network adapters as shown: IP address - 192.168.2.100 Subnet Mask - 255.255.255.0

17.3.3 Run the demo

1. Download the program to target board.
2. Connect the Ethernet cable between the PC and the board.
3. When successfully connected, reset the board to run the demo.
4. Open the PC command window, type in "ping 192.168.2.102" to test whether lwIP stack is running. If successful,
   four echo request packets are successfully replied.
5. Input "192.168.2.102" in the URL of an Internet browser on a PC. If successful, the web page the board returns opens in the browser.

Chapter 18
Ping Demo on lwIP TCP/IP Stack

This demo application demonstrates the Ping demo on lwIP TCP/IP stack with bare metal SDK or different RTOSes.

18.1 Overview

This is a Ping Demo on the lwIP TCP/IP stack which uses the ICMP protocol. The application on board periodically sends the ICMP echo request to a PC and processes the PC reply. Type the "ping $board_address" in the PC command window to send an ICMP echo request to the board. The lwIP stack sends the ICMP echo reply back to the PC.

18.2 Supported RTOS

- Freescale MQX™ RTOS
- FreeRTOS
- µC/OS-II
- µC/OS-III
- Bare Metal (no RTOS)

Demo Location

- MQX: demos/lwip_ping_demo/ping_rtos/ping_mqx
- FreeRTOS: demos/lwip_ping_demo/ping_rtos/ping_freertos
- µC/OS-II: demos/lwip_ping_demo/ping_rtos/ping_ucosii
- µC/OS-III: demos/lwip_ping_demo/ping_rtos/ping_ucosiii
- BM: demos/lwip_ping_demo/ping_bm

18.3 Getting Started

18.3.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

18.3.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

18.4 Run the demo

1. Download the program to the target board.
2. Connect the Ethernet cable between the PC and the board.
3. When successfully connected, reset the board to run the demo.
4. Open the terminal. Ping send and ping receive are successful.
5. Type in "ping 192.168.2.102" in PC command window. If the operation is successful, four packets are
   successful replied.

Chapter 19
TCP Echo Demo on lwIP TCP/IP Stack

This demo application demonstrates the TCP Echo demo on lwIP TCP/IP stack with bare metal SDK or different RTOSes.

19.1 Overview

This is a TCP echo demo on the lwIP TCP/IP stack with bare metal SDK or different RTOSes, which uses the TCP protocol and acts as an echo server. The application on board sends back the TCP packets from the PC, which can be used to test whether the TCP connection is available.

19.2 Supported RTOS

- Freescale MQX™ RTOS
- FreeRTOS
- µC/OS-II
- µC/OS-III
- Bare Metal (no RTOS)

Demo Location

- MQX: demos/lwip_tcpecho_demo/tcpecho_rtos/tcpecho_mqx
- FreeRTOS: demos/lwip_tcpecho_demo/tcpecho_rtos/tcpecho_freertos
- µC/OS-II: demos/lwip_tcpecho_demo/tcpecho_rtos/tcpecho_ucosii
- µC/OS-III: demos/lwip_tcpecho_demo/tcpecho_rtos/tcpecho_ucosiii
- BM: demos/lwip_tcpecho_demo/tcpecho_bm

19.3 Getting Started

19.3.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

19.3.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

19.4 Run the demo

1. Download the program to the target board.
2. Connect the Ethernet cable between the PC and the board.
3. When successfully connected, reset the board to run the demo.
4. Open the command window on PC, type in "tcpecho 192.168.2.102" to test whether the LwIP is running.
5. If it is running, use an external echo tool to perform the echo request. This tool sends TCP packets to the board and checks whether the content sent back from board is the same. A similar tool named "echotool" can be downloaded from the: `http://bansky.net/echotool/` [example: echotool 192.168.2.102 /p tcp /r 7 /d hello]
6. If the operation is successful, all packets sent back are same as the packets sent to the board.

Chapter 20
UDP Echo Demo on lwIP TCP/IP Stack

This demo application demonstrates the UDP Echo demo on lwIP TCP/IP stack with bare metal SDK or different RTOSes.

20.1 Overview

This is a UDP echo demo on the lwIP TCP/IP stack with bare metal SDK or different RTOSes, which uses the UDP protocol and acts as an echo server. The application on board sends back the UDP packets from the PC, which can be used to test whether the UDP connection is available.

20.2 Supported RTOS

- Freescale MQX™ RTOS
- FreeRTOS
- µC/OS-II
- µC/OS-III
- Bare Metal (no RTOS)

Demo Location

- MQX: demos/lwip_udpecho_demo/udpecho_rtos/udpecho_mqx
- FreeRTOS: demos/lwip_udpecho_demo/udpecho_rtos/udpecho_freertos
- µC/OS-II: demos/lwip_udpecho_demo/udpecho_rtos/udpecho_ucosii
- µC/OS-III: demos/lwip_udpecho_demo/udpecho_rtos/udpecho_ucosiii
- BM: demos/lwip_udpecho_demo/udpecho_bm

20.3 Getting Started

20.3.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

20.3.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

20.4 Run the demo

1. Download the program to the target board.
2. Connect the Ethernet cable between the PC and the board.
3. When successfully connected, reset the board to run the demo.
4. Open the command window on PC, type in "ping 192.168.2.102" to test whether the lwIP is running.
5. If it is running, use an external echo tool to perform the echo request. This tool sends UDP packets to the board and checks whether the content sent back from board is the same. A similar tool named "echotool" can be downloaded from the: `http://bansky.net/echotool/` [example: echotool 192.168.2.102 /p udp /r 7 /d hello]
6. If the operation is successful, all packets sent back are the same as the packets sent to the board.

Chapter 21
RTC Function Demo

This demo application demonstrates how to use the RTC driver.

21.1 Overview

This RTC demo application demonstrates the important features of the RTC Module by using the RTC Periperhal Driver. It supports these features:

- Calendar
  - Get the current date time with Year, Month, Day, Hour, Minute and Second.
  - Set the current date time with Year, Month, Day, Hour, Minute and Second.
- Alarm
  - Set the alarm based on the current time.
  - Application prints a notification when the alarm expires.
- Seconds interrupt
  - Use second interrupt function to display a digital time blink every second.
- Compensation
  - Configure the compensation with cycles.
  - The 1 Hz RTC clock with compensation configured is output to a pin. Use an oscilloscope to check the compensation result.

Demo Location

- demos/rtc_func

21.2 Getting Started

21.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

21.3 Run the demo

This menu is displayed on the serial terminal:

```
Please choose the sub demo to run:
1) Get current date time.
2) Set current date time.
3) Alarm trigger show.
4) Second interrupt show (demo for 20s).
5) Set RTC compensation.

Select:
```

Chapter 22
SAI Demo

This demo application demonstrates how to use the SAI drivers to communicate with WM8960 audio codec.

22.1 Overview

The SAI Demo project is a digital audio demonstration program that uses the KSDK software. It performs audio playback from either a .wav file, stored in Flash.

Demo Location

- demos/sai_demo

22.2 Getting Started

22.2.1 Hardware Settings

MAPS-K64 and MAPS-DOCK modules are required to run the sai_demo.

22.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

22.3 Run the demo

To hear the audio playback, connect a set of headphones to the headphone output on the MAPS-DOCK board CN10.

When the demo starts, this message is displayed in the terminal output window:

```
Audio Demo!~
```

```
Press spacebar to start demo.~

Demo begin...
```

The user can select play back a .wav file stored in the Flash.

```
Select player:
        1. Wav File Playback
->1
```

The user is prompted to select from a list of headphone output levels:

```
Choose headphone dB level:
        1. 0.0 dB
        2. -3.0 dB
        3. -6.0 dB
        4. -12.0 dB
        5. -24.0 dB
```

22.4 Key Functions

void audio_wav_init(wave_file_t *newWav)

Initializes the I2S, I2C, and audio codec module for playing back WAV file in Flash.~

Parameters

newWav & Pointer to wave file data structure.

uint32_t config_volume(sgtl_handler_t *handler, sgtl_module_t module, uint32_t volumeCtrl)

```
Sets volume from the user input.
```

Parameters

handler	pointer to codec handler structure.
module	name of module on codec to set the volume for.
volumeCtrl	user input data from terminal menu.

Returns

status_t Return kStatus_Success if function completed successfully, return kStatusFai

snd_status_t get_wav_data(wave_file_t *waveFile)

Collects data from WAV file header.

Parameters

waveFile	Data structure of pcm data array.

Returns

status_t Return kStatus_Success if function completed successfully, return kStatusFai

snd_status_t play_wav(uint32_t *pcmBuffer, uint8_t volumeCtrl)

Plays the PCM audio data from the WAV format array.

Parameters

pcmBuffer	Pointer to data array containing WAV formatted audio data.
volumeCtrl	Value used to set decibel level on codec.

Returns

status_t Return kStatus_Success if function completed successfully, return kStatusFai

void send_wav(uint8_t *dataBuffer, uint32_t length, sai_data_format_t *dataFormat)

Sends audio data to the sound card.

Parameters

pdataBuffer	Pointer to data array containing WAV formatted audio data.
length	length of WAV file to send.
dataFormat	Point to audio_data_format_t for sound card.

Chapter 23
SD Card Demo

This demo application demonstrates the SD Card demo.

23.1 Overview

The SD Card demo application demonstrates the use of SD card driver. It displays the card information followed by a write-read compare test and the erase operation.

Demo Location

- demos/sdhc_sdcard

23.2 Getting Started

Hardware configuration

There is no specific hardware requirement. The default configuration of the supported targets is sufficient for this demo. The demo uses the on-board connector support for the card detect signal which is connected to a GPIO line for card detection.

23.2.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo

23.3 Run the demo

1. Insert an SD or a micro-SD card depending on the connector on board. Ensure that the card doesn't contain any important content because the demo will erase and overwrite some sectors.
2. After the card detection, the card-specific information, such as capacity, is shown. Then, the user is encouraged to back up data as needed. A write-read-compare access is performed to demonstrate the use case.

3. If the card was not inserted as mentioned in step 1, the demo waits for the card insertion. Once a card is inserted, it auto-detects and proceeds as shown in step 3.

Chapter 24
SPI Flash Demo

This demo application demonstrates the access to the SPI flash.

24.1 Overview

This demo application provides simple menu with a series of commands to access the SPI Flash. These commands can be used either to read, write, or erase the SPI flash.

Demo Location

- demos/spi_flash

24.1.1 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Either press the reset button on your board or launch the debugger in your IDE to begin running the demo.

24.1.2 Run the demo

```
***SPI Flash Demo***

1 - Erase entire chip
2 - Erase sectors
3 - Erase block
4 - Program one page with pattern (0x5a) and verify
5 - Read byte
Please enter your choice (1-5):
```

You can select the command you want.

24.1.2.1 1. Erase the entire chip

This command erase the entire SPI flash.

```
Please enter your choice (1-5): 1
Erase OK!
```

24.1.2.2 2. Erase sectors

This command erase the sectors specified by user. The input sector address must be HEX, and aligned with 4KB sector size. The input sector erase length must be multi of 4KB.

```
Please enter your choice (1-5): 2

Input the sector address(HEX), 0x1000
Input the sector length:4096
Erase OK!
```

24.1.2.3 3. Erase one block

This command erase one block specified by user. The input block address must be HEX, and aligned with 32KB or 64KB block size.

```
Input the block address(HEX), 0x8000
1. 32K block size
2. 64K block size
Select the block size:1
Erase OK!
```

24.1.2.4 4. Programe one page and read out to verify

This command program one page (256B) with the value 0x5A, and then read out it to verify the program result. The input block address must be HEX, and aligned with 32KB or 64KB block size.

```
Please enter your choice (1-5): 4

Input the program address(HEX), 0x100
Program and verify done!
```

24.1.2.5 5. Read out one byte

This command read one byte from the address specified by user. The input read address must be HEX.

```
Please enter your choice (1-5): 5

Input the read address(HEX), 0x100
0x100 = 0x5a
```

Chapter 25
Watchdog Timer Reset Demo

This demo application demonstrates the Watchdog Timer Reset demo.

25.1 Overview

The Watchdog Timer Reset demo application demonstrates how the Watchdog module can be used to reset a device. The overflow time for a Watchdog timer is approximately 2 seconds.

Demo Location

- demos/wdog_reset

25.2 Getting Started

25.2.1 Hardware configuration

These switch buttons are used by this demo:

Platform	Switch	Notes
MAPS-K64	K5	

25.2.2 Prepare the Demo

1. Connect the mbed debugger (CN14) on dock board to PC host.
2. Open a serial terminal with the following settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. Download the program to the target board.
4. Press either the reset button on your board or launch the debugger in your IDE to begin running the demo.

25.3 Run the demo

1. When the program is running, the Watchdog is enabled. The program continuously refreshes the Watchdog to prevent the CPU reset.
2. The message: "Watchdog example running, Loop #: xx, press <SW> to start watchdog timeout..." displays on the terminal.

3. An LED also blinks. The color of the LED depends on the board:

Platform	LED Color
MAPS-K64	Red

1. When the SW button is pressed, the LED begins to blink rapidly, signifying that the Watchdog is about to expire.
2. When the Watchdog signals a reset, the "Watchdog (COP) Reset" message and "Watchdog (COP) reset count: xx" message is output to the terminal.

/∗!

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

freescale™