Kinetis SDK Demo Applications User's Guide

Freescale Semiconductor, Inc.

KSDKDEMOUG

1.0.0

Jul 2014

Contents

Chapter 1
ADC Hardware Trigger Demo

Chapter 2
ADC Low Power Demo

Chapter 10
Hardware Timer Demo

Chapter 11
I2C Communication 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)

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 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK ADC Hardware Trigger demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

1.3 Getting Started

1.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

1.4 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.

1.5 Customization Options

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

1.5.1 Default configurations

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

1.5.1.1 ADC configurations

1. Use ADC0 instance.
2. Use ADC_SE23 input pin as sample pin.
3. Use VREFH/L as reference voltage.

1.5.1.2 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.

1.5.2 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
```

1.5.3 Configure the signal frequency

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

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

1.5.4 Configure the ADC instance

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

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

1.5.5 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.

2.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the Kinetis software development kit ADC Low Power demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

2.3 Getting Started

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

2.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

2.3.2 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 green or white LED light turns on.
5. Increment or decrement the temperature to see the changes.

Chapter 3
DAC ADC Demo

This demo application demonstrates the DAC and ADC demo.

3.1 Overview

This application demonstrates how to configure the DAC and set the output on the DAC using software. It also demonstrates how to configure the ADC in 'Blocking Mode' and read ADC values.

3.2 Supported Platforms

This demo supports these Freescale Freedom development platforms and Tower System modules:

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

3.3 Getting Started

3.3.1 Hardware Settings

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

Platform	DAC Out		ADC In	
	Pin Name	**Board Location**	**Pin Name**	**Board Location**
FRDM-K22F	DAC0_OUT	J24-11	PTB0/ADC0_SE8	J24-2 (Arduino:A0)
FRDM-K64F	DAC0_OUT	J4-11	PTB2/ADC0_SE12	J4-2 (Arduino:A0)
TWR-K22F120M	DAC0_OUT	Primary Elevator - A32	PTB0/ADC0_SE8	Primary Elevator - B27
TWR-K64F120M	DAC0_OUT	Primary Elevator - A32	PTB4/ADC1_SE10	Primary Elevator - B27
TWR-KV31F120-M	DAC0_OUT	Primary Elevator - A32	PTE2/ADC1_SE6a	Primary Elevator - B27

3.3.2 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.

2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

3.4 Run the demo

This example shows how to run the demo on TWR-K22F120M:

```
DAC ADC Demo!
-------------------------------------------------------------------------
 These pins are used for this demo.
-------------------------------------------------------------------------
-----------------------------------------------------------------
-----------------------------------------------------------------
  DAC Out
-----------------------------------------------------------------
  DAC0_OUT (J24A-A32)
-----------------------------------------------------------------
-----------------------------------------------------------------
  ADC In
-----------------------------------------------------------------
  PTB0/ADC0_SE8 (J24A-B27)
-----------------------------------------------------------------
-----------------------------------------------------------------
Press space bar to start the demo.
```

The user is prompted to enter a voltage to output on the DAC:

```
Select DAC output level:
        1. 1.0 V
        2. 1.5 V
        3. 2.0 V
        4. 2.5 V
        5. 3.0 V
->
```

After entering a valid input, the ADC captures the voltage set by the DAC and displays the result in the terminal:

```
Select DAC output level:
        1. 1.0 V
        2. 1.5 V
        3. 2.0 V
        4. 2.5 V
        5. 3.0 V
->3


ADC Value: 2471


ADC Voltage: 1.99


What next?:
        1. Test another DAC output value.
        2. Terminate demo.
->
```

At this point, the user can test another DAC output value or terminate the demo.

This configuration exhibits up to 2% error when reading back voltage.

3.5 Key Functions

uint8_t demo_start(demo_state_t *prevState)

Prints out a welcome message and pins required by the demo.

Parameters

**prevState*	Pointer to previous state for state machine.

Returns

msg Returns the character entered into the terminal by user.

uint8_t device_config(demo_state_t *prevState)

Configures the DAC and the ADC. The DAC is configured for software updates. The ADC is set in 'Blocking Mode'.

Parameters

∗*prevState*	Pointer to previous state for state machine.

Returns

msg Returns 0.

uint8_t dac_set(demo_state_t ∗prevState)

Sets output level on the DAC.

Parameters

∗*prevState*	Pointer to previous state for state machine.

Returns

msg Returns the character entered into the terminal by user.

uint8_t wait_state(demo_state_t ∗prevState)

Performs a wait and possible state change based on the ∗prevState.

Parameters

∗*prevState*	Pointer to previous state for state machine.

Returns

msg Returns 0.

uint8_t adc_get(demo_state_t ∗prevState)

Gets ADC values from a channel connected to the DAC output.

Parameters

∗*prevState*	Pointer to previous state for state machine.

Returns

msg Returns the character entered into the terminal by user.

uint8_t device_deinit(demo_state_t *prevState)

Deinitializes the DAC and the ADC module following a user command to terminate the demo. Also frees allocated memory.

Parameters

∗prevState	Pointer to previous state for state machine.

Returns

msg Returns 0.

uint8_t demo_end(demo_state_t ∗prevState)

Indicates to the user that the demo has been terminated.

Parameters

∗prevState	Pointer to previous state for state machine.

Returns

msg Returns 0.

Chapter 4
DSPI eDMA Demo

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

4.1 Overview

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

4.2 Supported Platforms

This demo supports the following Freescale Freedom development platforms and Tower System modules:

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

4.3 Getting Started

4.3.1 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.

FRDM-K22F:

	Master		Connects To	Slave	
Signal	**Pin Name**	**Board Location**		**Pin Name**	**Board Location**
SS	PTD0	J2 - Pin 6	->	PTD4	J6 - Pin 4
SCK	PTD1	J2 - Pin 12	->	PTD5	J6 - Pin 5
Data In	PTD2	J2 - Pin 8	->	PTD7	J6 - Pin 7
Data Out	PTD3	J2 - Pin 10	->	PTD6	J6 - Pin 6

FRDM-K64F:

	Master	Connects To	Slave

Signal	Pin Name	Board Location		Pin Name	Board Location
SS	PTD0	J2 - Pin 6	->	PTD4	J6 - Pin 4
SCK	PTD1	J2 - Pin 12	->	PTD5	J6 - Pin 5
Data In	PTD2	J2 - Pin 8	->	PTD7	J6 - Pin 7
Data Out	PTD3	J2 - Pin 10	->	PTD6	J6 - Pin 6

TWR-K22F120M:

	Master		Connects To	Slave	
Signal	**Pin Name**	**Board Location**		**Pin Name**	**Board Location**
SS	PTD0	Primary Elevator - B46	->	PTB10	Primary Elevator - B9
SCK	PTD1	Primary Elevator - B48	->	PTB11	Primary Elevator - B7
Data In	PTD2	Primary Elevator - B45	->	PTB17	Primary Elevator - B11
Data Out	PTD3	Primary Elevator - B44	->	PTB16	Primary Elevator - B10

TWR-K64F120M:

	Master		Connects To	Slave	
Signal	**Pin Name**	**Board Location**		**Pin Name**	**Board Location**
SS	PTD0	Primary Elevator - B46	->	PTB10	Primary Elevator - B9
SCK	PTD1	Primary Elevator - B48	->	PTB11	Primary Elevator - B7
Data In	PTD2	Primary Elevator - B45	->	PTB17	Primary Elevator - B11
Data Out	PTD3	Primary Elevator - B44	->	PTB16	Primary Elevator - B10

TWR-KV31F120M :

	Master		Connects To	Slave	
Signal	**Pin Name**	**Board Location**		**Pin Name**	**Board Location**

SS	PTE16	Primary Elevator - B46	->	PTB10	Primary Elevator - B9
SCK	PTE17	Primary Elevator - B48	->	PTB11	Primary Elevator - B7
Data In	PTE18	Primary Elevator - B45	->	PTE3	Primary Elevator - B11
Data Out	PTE19	Primary Elevator - B44	->	PTE1	Primary Elevator - B10

4.3.2 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

4.4 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)
```

Run the demo

```
 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.
```

4.5 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 5
Flash Demo

This demo application demonstrates how to use the Flash drivers.

5.1 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)

5.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK Flash demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

5.3 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.

5.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

5.4 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 6
FlexCAN and UART communication Demo

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

6.1 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.

6.2 Supported Hardware

This Tower System module is supported by the KSDK FlexCAN and UART demo.

- TWR-K64F120M

6.3 Getting Started

6.3.1 Hardware configuration

TWR-SER Tower System module configuration

1. Short J5(1-2), J5(3-4), J5(5-6), J5(7-8), and J5(9-10) to enable CAN connection.
2. Connect the two TWR-SER modules through the CAN port (J7).

6.3.2 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

6.4 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 7
FlexTimer PWM Demo

This demo application demonstrates the FlexTimer PWM demo.

7.1 Overview

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

7.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK FlexTimer demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

7.3 Getting Started

Hardware configuration

This table lists which FTM channels and MCU pins control which LEDs 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	LED/Color	Jumper Config
FRDM-K22F	FTM0 - CH5	PTD5	D14 - Blue	N/A
FRDM-K64F	FTM0 - CH0	PTC1	D12 - Green	J1/P5 - J2/P1
TWR-K22F120M	FTM0 - CH4	PTD4	D7 - YEL/GRN	J16/P1-P2
TWR-K64F120M	FTM3 - CH1	PTE6	D5 - YEL/GRN	J30/P1-P2
TWR-KV31F120-M	FTM0 - CH7	PTD7	D7 - YEL/GRN	J17/P7-P8

NOTE: The FRDM-K64 requires an external jumper wire to make the appropriate connections.

7.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.

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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

7.3.2 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. The LED on board increases/decreases intensity according to PWM pulse width changes. The LED affected is indicated in the table in the Hardware Configurations part.

Chapter 8
GPIO I2C Demo

This demo application demonstrates the GPIO I2C demo.

8.1 Overview

This demo application is a simulator which uses the GPIO to simulate the I2C protocol and control the MMA8451 or the FXOS8700CQ microcontrollers on the board. The UART console shows values received from the MMA8451 or the FXOS8700CQ.

8.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK GPIO I2C demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

8.3 Getting Started

8.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

Hardware configuration

These pins are used to connect to the accelerometer on the board:

Platform	SCL	SDA
FRDM-K22F	PTB2	PTB3
FRDM-K64F	PTE24	PTE25
TWR-K22F120M	PTC10	PTC11
TWR-K64F120M	PTC10	PTC11
TWR-KV31F120M	PTD2	PTD3

8.3.2 Run the demo

1. Download the program to a target broad.
2. Connect the USB cable to the PC host and open a serial terminal configured as shown in the table.
3. For the MMA8451, the UART outputs values in the X, Y, and Z direction, received from the MMA8451.

```
== Accelerometer Test ==

Reading acceleration data...
x=4, y=1, z=1
```

4. For the FXOS8700CQ, UART provides output values in the X, Y, Z direction and magnetism.

```
== Accelerometer Test ==

Jan  8 2014    15:12:40
Initializing FXOS...
Resetting FXOS...
Reading acceleration data...
xAcc = 28, yAcc = -100, zAcc = 2075, xMag = 89, yMag = -55, zMag = 336
```

8.3.3 Customization Options

Parameters are available and can be configured to meet other requirements.

SCL and SDA configurations

The BOARD_I2C_GPIO_SCL and BOARD_I2C_GPIO_SDA macros define which GPIO pins are used to emulate I2C pins for the MMA8451 or FXOS8700CQ I2C interface.

These macros can be configured in mcu-sdk\boards\<board_name>\board.h

```
#define BOARD_I2C_GPIO_SCL              GPIO_MAKE_PIN(HW_GPIOE, 24)
#define BOARD_I2C_GPIO_SDA              GPIO_MAKE_PIN(HW_GPIOE, 25)
```

Baudrate configuration

To get the correct I2C baudrate, use an oscilloscope to look at the baudrate waveform.

A delay is used in this demo application for a correct baudrate. This is defined in the mcu-sdk\boards\<board_name>\board.h file.

```
#define BOARD_I2C_DELAY \
do \
{ \
    int32_t i; \
      for (i = 0; i < 500; i++) \
    { \
        __asm("nop"); \
    } \
} while (0)
```

Important things to remember:

- For a higher baudrate, reduce the delay time and loop count.
- For a lower baudrate, increase the delay time and loop count.
- Keep in mind code execution time.
- Use an oscilloscope to get the correct baudrate waveform.
- There is no delay in code execution in this demo application. Therefore, this might be the highest baudrate.
- See the I2C protocol and the MMA8451 and the FXOS microcontroller details for more information.

Chapter 9
Hello World Demo

This demo application demonstrates the Hello World demo.

9.1 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.

9.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK Hello World demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

9.3 Getting Started

9.3.1 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.

9.3.2 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

Communication Interface Settings:

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

9.4 Run the demo

This is an example how to run the demo.

```
Hello World!
```

9.5 Communication Interface Settings:

This part provides the information to customize the Hello World demo. The Hello World demo is configured to use these port pins for the platforms by default. If applicable for the board, jumpers are specified to select between serial output via OpenSDA and serial output via TWR-SER.

Platform	TX MCU Pin (Board Pin)	RX MCU Pin (Board Pin)	Module Instance
FRDM-K22F	PTE0 (N/A)	PTE1 (N/A)	UART1
FRDM-K64F	PTB17 (N/A)	PTB16 (N/A)	UART0
TWR-K22F120M	PTE0 (J30)	PTE1 (J29)	UART1
TWR-K64F120M	PTC4 (J15)	PTC3 (J10)	UART1
TWR-KV31F120M	PTB17 (J23)	PTB16 (J22)	UART0

9.5.1 Customization Options

The USE_STDIO_FUNCTIONS define determines if the demo uses standard I/O functions, such as printf. If it is not defined, then the demo accesses the UART driver directly.

Chapter 10
Hardware Timer Demo

This demo application demonstrates using the hardware timer driver.

10.1 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.

10.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the Kinetis SDK Hardware Timer demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

10.3 Getting Started

10.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

10.3.2 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 ∗ HWTIMER_LINES_COUNT (defined in hwtimer_demo.c) interrupts occur.
4. Finally, the "End" message is displayed.

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

10.4 Customization Options

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

10.4.1 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
```

10.4.2 Configure which clock is used by the hardware timer

Determine which clock source is used by the hardware timer.

```
#define HWTIMER_LL_SRCCLK   kCoreClock
```

10.4.3 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
```

10.4.4 Hardware Timer Period

Determine the timer period (in microseconds).

```
#define HWTIMER_PERIOD          100000
```

Chapter 11
I2C Communication Demo

This demo application demonstrates the I2C demo.

11.1 Overview

The I2C communication application demonstrates I2C data communication between two boards. It also features low power wakeup of the slave board by using I2C address matching.

First, the I2C slave board enters the low power wait mode. An LED on the I2C slave board is on to indicate that the MCU is in sleep mode and no code is running.

Then, the I2C slave board is woken up by the I2C address matching interrupt when the I2C master boards sends the proper address. The LED on the I2C slave board is toggled during the data communication.

After power on, the I2C master starts reading data from the I2C slave data buffer. The I2C slave has "sub" addresses to access a specific byte of data on the slave board. The master prints this data out via the serial terminal. The master can then modify the data at a specific "sub" address on the slave board. When the data is received, the I2C slave changes the content at that requested "sub" address. This change is reflected when the master reads the slave data buffer again.

11.2 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK I2C Communication demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

11.3 Getting Started

11.3.1 Hardware configuration

This demo requires two separate boards. Make these connections between the two boards by using external wires:

FRDM-K22F:

Master Board		Connects To	Slave Board	
Pin Name	Board Location		Pin Name	Board Location
PTB2/I2C0_SCL	J24 Pin 12	->	PTB2/I2C0_SCL	J24 Pin 12
PTB3/I2C0_SDA	J24 Pin 10	->	PTB3/I2C0_SDA	J24 Pin 10
GND	J2 Pin 14	->	GND	J2 Pin 14

FRDM-K64F:

Master Board		Connects To	Slave Board	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	J2 Pin 20	->	PTE24/I2C0_SCL	J2 Pin 20
PTE25/I2C0_SDA	J2 Pin 18	->	PTE25/I2C0_SDA	J2 Pin 18
GND	J2 Pin 14	->	GND	J2 Pin 14

TWR-K22F120M & TWR-KV31F120M:

Master Board		Connects To	Slave Board	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	Primary Elevator A7	->	PTE24/I2C0_SCL	Primary Elevator A7
PTE25/I2C0_SDA	Primary Elevator A8	->	PTE25/I2C0_SDA	Primary Elevator A8
GND	Primary Elevator A6	->	GND	Primary Elevator A6

TWR-K64F120M:

Master Board		Connects To	Slave Board	
Pin Name	Board Location		Pin Name	Board Location
PTC10/I2C1_SCL	Primary Elevator A75	->	PTC10/I2C1_SCL	Primary Elevator A75
PTC11/I2C1_SDA	Primary Elevator A60	->	PTC11/I2C1_SDA	Primary Elevator A60
GND	Primary Elevator A65	->	GND	Primary Elevator A65

11.3.2 Terminal configuration

Configure the PC host serial console as shown:

- 115200 baud rate
- 8 data bits
- No parity
- One stop bit
- No flow control

11.3.3 Run the demo

1. Connect the I2C slave board to the master board using the connections listed above.
2. Power on the I2C slave board.
3. Download and run the i2c_comm_slave project to the I2C slave board.
4. The terminal of the I2C slave board prints out a "====== I2C Slave ======" message.
5. Power on the I2C master board.
6. Download and run the i2c_comm_mstr project to the I2C master board.
7. The terminal of the I2C master board prints out a "====== I2C Master ======" message and the data received from the I2C slave.
8. The I2C slave project creates some "sub" addresses to access a specific byte of data on the slave board. The master reads all these "sub" addresses and prints out the data.

Slave Sub Address	Character
[0]	I
[1]	2
[2]	C
[3]	-
[4]	C
[5]	O
[6]	M
[7]	M

9. To change the I2C slave sub address content, input a new character in the I2C master command line:

```
Input slave sub address and the new character.
Slave Sub Address: 5
Input New Character: F
```

10. The master then displays the updated content on the terminal output.

Slave Sub Address	Character
[0]	I
[1]	2

[2]	C
[3]	-
[4]	C
[5]	F
[6]	M
[7]	M

Chapter 12
I2C Demo with RTOS

This demo application demonstrates the I2C demo on different RTOS.

12.1 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 RED/GREEN/BLUE color 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.

12.2 Supported RTOS

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

12.3 Supported Platforms

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK I2C demo with RTOS.

- FRDM-K22F
- FRDM-K64F

- TWR-K22F120M
- TWR-K64F120M

12.4 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.

12.4.1 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.

12.4.2 Hardware configuration

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

Freescale Freedom FRDM-K22F

FRDM-K22F Single Board				
Master		Connects To	Slave	
Pin Name	Board Location		Pin Name	Board Location
PTB2/I2C0_SCL	J24 - Pin 12	->	PTC10/I2C1_SCL	J1 - Pin 13
PTB3/I2C0_SDA	J24 - Pin 10	->	PTC11/I2C1_SDA	J2 - Pin 7

FRDM-K22F Two Boards				
Master (Board #1)		Connects To	Slave (Board #2)	
Pin Name	Board Location		Pin Name	Board Location
PTB2/I2C0_SCL	J24 - Pin 12	->	PTC10/I2C1_SCL	J1 - Pin 13
PTB3/I2C0_SDA	J24 - Pin 10	->	PTC11/I2C1_SDA	J2 - Pin 7
GND	TP21	->	GND	TP21

Freescale Freedom FRDM-K64F

FRDM-K64F Single Board

Master		Connects To	Slave	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	J2 - Pin 20	->	PTC10/I2C1_SCL	J4 - Pin 12
PTE25/I2C0_SDA	J2 - Pin 18	->	PTC11/I2C1_SDA	J4 - Pin 10

FRDM-K64F Two Boards				
Master (Board #1)		Connects To	Slave (Board #2)	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	J2 - Pin 20	->	PTC10/I2C1_SCL	J4 - Pin 12
PTE25/I2C0_SDA	J2 - Pin 18	->	PTC11/I2C1_SDA	J4 - Pin 10
GND	J2 - Pin 14	->	GND	J2 - Pin 14

TWR-K22F120M Tower System module

TWR-K22F120M Single Board				
Master		Connects To	Slave	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	Primary Elevator - Pin A7	->	PTC10/I2C1_SCL	Primary Elevator - Pin B50
PTE25/I2C0_SDA	Primary Elevator - Pin A8	->	PTC11/I2C1_SDA	Primary Elevator - Pin B51

TWR-K22F120M Two Boards				
Master (Board #1)		Connects To	Slave (Board #2)	
Pin Name	Board Location		Pin Name	Board Location
PTE24/I2C0_SCL	Primary Elevator - Pin A7	->	PTC10/I2C1_SCL	Primary Elevator - Pin B50
PTE25/I2C0_SDA	Primary Elevator - Pin A8	->	PTC11/I2C1_SDA	Primary Elevator - Pin B51
GND	Primary Elevator - Pin A65	->	GND	Primary Elevator - Pin A65

TWR-K64F120M Tower System module

TWR-K64F120M Single Board				
Master		Connects To	Slave	
Pin Name	Board Location		Pin Name	Board Location

PTD8/I2C0_SCL	Primary Elevator - Pin A7	->	PTC10/I2C1_SCL	Primary Elevator - Pin A75
PTD9/I2C0_SDA	Primary Elevator - Pin A8	->	PTC11/I2C1_SDA	Primary Elevator - Pin B71

TWR-K64F120M Two Boards				
Master (Board #1)		Connects To	Slave (Board #2)	
Pin Name	Board Location		Pin Name	Board Location
PTD8/I2C0_SCL	Primary Elevator - Pin A7	->	PTC10/I2C1_SCL	Primary Elevator - Pin A75
PTD9/I2C0_SDA	Primary Elevator - Pin A8	->	PTC11/I2C1_SDA	Primary Elevator - Pin B71
GND	Primary Elevator - Pin A65	->	GND	Primary Elevator - Pin A65

12.4.3 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

12.5 Run the demo

This menu displays in the terminal window:

```
Available Commands:
LED Red Toggle (1)    - Red Light toggles on/off
LED Green Toggle (2)  - Green Light toggles on/off
LED Blue Toggle (3)  - Blue Light toggles on/off
Read Temperature (4)   - Get temperature of client
Read Id     (5)    - Read client unique ID
```

Enter your choice (1 - 5):

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

Chapter 13
LPTMR Demo

This demo application demonstrates the LPTMR demo.

13.1 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.

13.2 Supported Platforms

This demo supports the following Freescale Freedom development platforms and Tower System modules:

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

13.3 Getting Started

13.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

13.4 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:

Run the demo

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

Chapter 14
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.

14.1 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.

14.2 Supported RTOS

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

14.3 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the Kinetis software development kit HTTPServer demo.

- FRDM-K64F
- TWR-K64F120M

14.4 Getting Started

See the *lwIP TCPIP Stack and Kinetis SDK Integration User's Guide* (document KSDKLWIPUG) for more information about the setup and requirements.

14.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions steps, see a Kinetis SDK User's Guide for your board.

14.4.2 Network Configuration

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

14.4.3 Run the demo

1. Download the program to target board, which should be installed in TWR or FRDM.
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 15
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.

15.1 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.

15.2 Supported RTOS

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

15.3 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK Ping demo.

- FRDM-K64F
- TWR-K64F120M

15.4 Getting Started

See the *lwIP TCPIP Stack and Kinetis SDK Integration User's Guide* (document KSDKLWIPUG) for instructions and requirements.

15.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
    - 115200 baud rate
    - 8 data bits
    - No parity
    - One stop bit
    - No flow control

3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

15.4.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

15.5 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 16
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.

16.1 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.

16.2 Supported RTOS

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

16.3 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK TCP Echo demo.

- FRDM-K64F
- TWR-K64F120M

16.4 Getting Started

See the *lwIP TCPIP Stack and Kinetis SDK Integration User's Guide* (document KSDKLWIPUG) for instructions and requirements.

16.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

16.4.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

16.5 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 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 17
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.

17.1 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.

17.2 Supported RTOS

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

17.3 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK UDP Echo demo.

- FRDM-K64F
- TWR-K64F120M

17.4 Getting Started

See the *lwIP TCPIP Stack and Kinetis SDK Integration User's Guide* (document KSDKLWIPUG) for instructions and requirements.

17.4.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

17.4.2 Network Configuration

Configure the IP address of PC network adapters as shown:

- 192.168.2.100

17.5 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 18
RTC Function Demo

This demo application demonstrates how to use the RTC driver.

18.1 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.

18.2 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK RTC Function demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M

18.3 Getting Started

18.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

18.4 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 19
SAI Demo

This demo application demonstrates how to use the SAI drivers.

19.1 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, or from the line-in on a TWR-AUDIO-SGTL Tower System module using the KSDK I2S and I2C drivers. On the TWR-K22F120M and the TWR-K64F120M Tower System modules, the project also uses the CMSIS-DSP library to perform a Fast Fourier Transform, and return the fundamental frequency of the line-in audio.

19.2 Supported Hardware

This demo supports the following Freescale Freedom development platforms and Tower System modules:

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

19.3 Getting Started

19.3.1 GCC Compiler notes

When building the demo with GCC, ensure that the demo and platform library are built with this option:

```
CHOOSE_FLOAT=HARD_FP
```

Otherwise, the project does not use the Kinetis device's hardware floating point when using the CMSIS--DSP library.

19.3.2 Hardware Settings

These Tower System modules are required to run the sai_demo:

- TWR-ELEV
- TWR-AUDIO-SGTL

19.3.3 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
2. Open a serial terminal with these settings:
   - 115200 baud rate
   - 8 data bits
   - No parity
   - One stop bit
   - No flow control
3. 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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

19.4 Run the demo

To hear the audio playback, connect a set of headphones to the headphone output on the TWR-AUDIO--SGTL card. For input to the codec, connect an audio source to the Line-In on the TWR-AUDIO-SGTL.

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 either play back audio from the line-in source, or play a .wav file stored in the Flash.

The line-in option plays the audio gathered from the codec line-in for approximately 15 seconds.

```
Select player:
        1. Line-In Playback
        2. Wav File Playback
->1
```

If selecting playback from the line-in source, decide whether to perform an FFT analysis to find the fundamental frequency of the audio input. Finding the fundamental frequency is best suited for pure tones played into the line-in of the TWR-AUDIO-SGTL card.

```
Select filter:
        1. FFT - Find Fundamental Frequency
        2. None
->1
```

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

```
Choose headphone dB level:
        1. +3.0 dB
        2. 0.0 dB
        3. -3.0 dB
        4. -6.0 dB
        5. -12.0 dB
        6. -24.0 dB
        7. -48.0 dB
->5
Frequency is 93 Hz
```

The table shows the terminal display after playback has completed and the FFT option was selected.

These are the options for the .wav file option:

```
Select player:
        1. Line-In Playback
        2. Wav File Playback
->2
Select Wav file:
        1. Audio Demo
->1
Choose headphone dB level:
        1. +3.0 dB
        2. 0.0 dB
        3. -3.0 dB
        4. -6.0 dB
        5. -12.0 dB
        6. -24.0 dB
        7. -48.0 dB
->5
```

The quality of the .wav file PCM data depends on the demo system and the compiler.

The table below shows the audio sample rate, channels and bit depth of the .wav file for the various platforms and compilers.

Hardware System	Sample Rate (kHz)				Bit Depth				Channels			
	IAR	ARM	GNU-GCC	KDS-GCC	IAR	ARM	GNU-GCC	KDS-GCC	IAR	ARM	GNU-GCC	KDS-GCC
TWR--K64-F120-M	44.1	44.1	44.1	44.1	32	32	32	32	2	2	2	2

TWR--K22-F120-M	44.1	44.1	11.-025	11.-025	16	16	16	16	2	2	2	2
TWR--K22-F120-M256	11.-025	11.-025	11.-025	11.-025	16	16	16	16	2	2	1	1
TWR--K22-F120-M128	11.-025	11.-025	11.-025	11.-025	16	16	16	16	2	2	1	1

Quality differences of the .wav playback depend on the size constraints of the target device, the Flash size, and the density of the code generated by the compiler.

Note that all supported platforms play audio from the line-in option with the same quality: 16-bit, 44.1 kHz, 2 channels.

19.5 Key Functions

void audio_stream_init(void)

Initializes the I2S, I2C, and TWR-AUDIO-SGTL Tower System module for streaming audio from Line-In.

void audio_wav_init(wave_file_t ∗newWav)

Initializes the I2S, I2C, and TWR-AUDIO-SGTL Tower System 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 kStatusFail if function failed.

snd_status_t stream_audio(dsp_types_t dspType, uint8_t volumeCtrl)

Plays a stream of audio.

Parameters

dspType	Used to select one DSP function to perform on the data.
volumeCtrl	Value used to set decibel level on codec.

Returns

Returns soundcard status

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 kStatusFail if function failed.

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 kStatusFail if function failed.

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.

float32_t do_fft(sai_data_format_t *dataFormat, uint8_t *buffer, float32_t *fftData, float32_t *fftResult)

Performs frequency analysis and finds fundamental frequency of the PCM data.

Parameters

dataFormat	Pointer to audio data format structure.
buffer	Pointer to data array to store modulated PCM data.
fftData	Pointer to data array for storing Fast Fourier Transform data.
fftResult	Point to data array for storing real frequency bins from FFT.

Returns

float32_t Returns fundamental frequency in Hz.

Chapter 20
SD Card Demo

This demo application demonstrates the SD Card demo.

20.1 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.

20.2 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK SD Card demo.

- FRDM-K64F
- TWR-K64F120M

20.3 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.

20.3.1 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

20.4 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 21
Watchdog Timer Reset Demo

This demo application demonstrates the Watchdog Timer Reset demo.

21.1 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.

21.2 Supported Hardware

These Freescale Freedom development platforms and Tower System modules are supported by the KSDK Watchdog Timer Reset demo.

- FRDM-K22F
- FRDM-K64F
- TWR-K22F120M
- TWR-K64F120M
- TWR-KV31F120M

21.3 Getting Started

21.3.1 Hardware configuration

These switch buttons are used by this demo:

Platform	Switch	Notes
FRDM-K22F	SW2	
FRDM-K64F	SW2	
TWR-K22F120M	SW1	
TWR-K64F120M	SW1	
TWR-KV31F120M	SW1	Jumper J26 should be removed

Note that on the some boards, the reset source information printed out to the terminal may be incorrect and may sometimes display an "External Pin Reset" message. This occurs when the Watchdog resets the system, and the OpenSDA circuit also sends a pin reset to the target MCU through the voltage level translator. Sometimes this translator can't pull up the pin in time and the duration of the low level of this pin is long. When this occurs, the Watchdog reset status is overridden by the external pin reset and the message "External Pin Reset" is output to the terminal.

21.3.2 Prepare the Demo

1. Connect a USB cable between the PC host and the OpenSDA USB port on the board.
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.

For more detailed instructions, see a Kinetis SDK User's Guide for your board.

21.4 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
FRDM-K22F	Red
FRDM-K64F	Red
TWR-K22F120M	Yellow
TWR-K64F120M	Yellow
TWR-KV31F120M	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.

/*!

Chapter 22
Revision History

This table summarizes revisions to this document.

Revision History		
Revision number	Date	Substantial changes
1.0.0	7/2014	Initial release

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: KSDKDEMOUG
Rev. 1.0.0
07/2014