

DRIVER TUTORIAL EXAMPLES

FRDM-KEAZ128, FRDM-
KEAZ64 AND FRDM-KEAZN
BOARDS

ULTRA-RELIABLE MICROCONTROLLERS
(MCU) FOR INDUSTRIAL AND AUTOMOTIVE
FEB, 2020

USING S32DS



PUBLIC



SECURE CONNECTIONS
FOR A SMARTER WORLD

Index

Drivers and Examples

[1. General Purpose Input Output \(GPIO\)](#)

[2. Internal Clock Source \(ICS\)](#)

[Nested Vectored Interrupt Controller \(NVIC\)](#)

[PRINT](#)

[3. Universal Asynchronous Receiver Transmitter](#)

[4. Key Board Interrupt \(KBI\)](#)

[5. FlexTimer Module \(FTM\)](#)

[6. Analog to Digital Converter \(ADC\)](#)

[7. Analog Comparator \(ACMP\)](#)

[8. Periodic Interrupt Timer \(PIT\)](#)

[9. Real Timer Clock \(RTC\)](#)

[10. Pulse Width Timer \(PWT\)](#)

[11. Watchdog Timer \(WDOG\)](#)

[12. Serial Peripheral Interface \(SPI\)](#)

[13. Inter Integrated Circuit \(I2C\)](#)

[14. Aliased Bit Band Domain](#)

[15. Bit Manipulation Engine \(BME\)](#)

[16. Cyclic Redundancy Check \(CRC\)](#)

[17. Flash Memory Module \(FLASH\)](#)

[18. Power Management Controller \(PMC\)](#)

[19. Local Interconnect Interface \(LIN\)](#)

DRIVERS AND EXAMPLES

1. General Purpose Input Output (GPIO)

Overview

- There are up to 71 GPIOs in the Kinetis EA series MCUs. Each GPIO in the MCU includes an internal pull-up resistor. Efficient bit manipulation of the general-purpose outputs is supported through the addition of set, clear, and toggle write-only registers for each port output data register.



SW2
SW3
LEDs

Example:

- This example uses polling to manipulate GPIO pins.
- This Example Turns on the RED & GREEN LEDs up on pressing SW2(BTN0) & SW3(BTN1) respectively
- LEDs remains turned on as long as the SW is pressed

Hardware Connections:

- No Hardware connections required

Library : gpio.h

- With the GPIO library you will be able to configure the GPIOs as inputs or outputs, enable the internal pull up resistors, toggle, set or clear outputs, and read inputs

2. Internal Clock Source (ICS)

Overview

- The Internal Clock Source module is used for main system clock generation. The ICS module controls which clock sources (internal references, external crystals or external clock signals) generate the source of the system clocks.



```
COM8 - PuTTY
Running the FRDM_KEA128_ICS_FEE project.
Dual edge capture end.
The input counts are 16
The input wave frequency is 156.25 kHz
```

FEE Example:

For, FRDM-KEAZ128:

- FLL Engaged External (FEE). Bus clock is set at 20 MHz using external crystal of 8MHz. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH4 (PTG6 –J1_13) in Dual Edge Capture mode.
- Clock out = $20\text{MHz}/128 = 156.25\text{ KHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J1_13

Library: ics.h& ics.c

- This library allows to select the reference clock that will be used for system clock generation. Also it allows to switch between 7 operation modes of the ICS : FEI, FEE, FBI, FBILP, FBE, FBELP and STOP.



2. Internal Clock Source (ICS)



FEE Example:

For, FRDM-KEAZ64

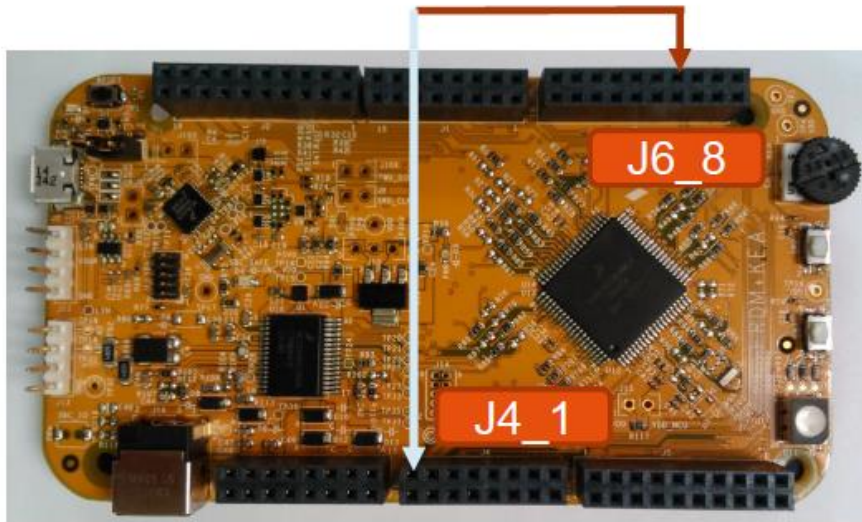
- FLL Engaged External (FEE). Bus clock is set at 20 MHz using external crystal of 8MHz. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH0 (PTB4-J2_4) in Dual Edge Capture mode.
- $\text{Clock out} = 20\text{MHz}/128 = 156.25 \text{ KHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J2_4

```
COM16 - PuTTY
Running the FRDM_KEA64_ICS_FEE project.
Dual edge capture end.
The input counts are 16
The input wave frequency is 156.25 kHz
```

2. Internal Clock Source (ICS)



FEE Example: For, FRDM-KEAZN32

- FLL Engaged External (FEE). Bus clock is set at 16 MHz using external crystal of 8MHz. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH0 (PTC0 –J4_1) in Dual Edge Capture mode.
- $\text{Clock out} = 16\text{MHz}/128 = 125 \text{ KHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

```
COM14 - PuTTY
Running the FRDM_KEA32_ICS_FEE project.
Dual edge capture end.
The input counts are 8
The input wave frequency is 125.0 kHz
```

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J4_1

2. Internal Clock Source (ICS)



```
COM8 - PuTTY
Running the FRDM_KEA128_ICS_FEE project.
Dual edge capture end.
The input counts are 16

The input wave frequency is 187.50 kHz
```

FEI Example:

For, FRDM-KEAZ128

- FLL Engaged Internal (FEI). By default KEAZ128/64 internal reference clock is trimmed at 37.5 kHz. This example sets a bus clock of 24 MHz using internal reference clock. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH4 (PTG6 –J1_13) in Dual Edge Capture mode.
- Clock out = $24\text{MHz}/128 = 187.5\text{ kHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J1_13

2. Internal Clock Source (ICS)



```
COM16 - PuTTY
Running the FRDM_KEA64_ICS_FEI project.
Dual edge capture end.
The input counts are 16
The input wave frequency is 187.50 kHz
```

FEI Example:

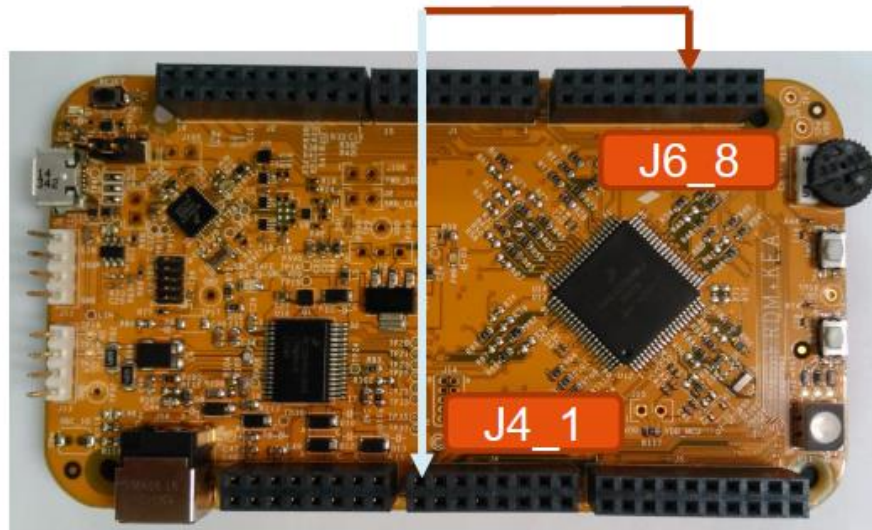
For, FRDM-KEAZ64

- FLL Engaged Internal (FEI). By default KEAZ128/64 internal reference clock is trimmed at 37.5 kHz. This example sets a bus clock of 24 MHz using internal reference clock. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH0 (PTB4 –J2_4) in Dual Edge Capture mode.
- Clock out = $24\text{MHz}/128 = 187.5 \text{ kHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J2_4

2. Internal Clock Source (ICS)



FEI Example: For, FRDM-KEAZN32

- FLL Engaged Internal (FEI). By default KEAZN32 internal reference clock is trimmed at 32 kHz. This example sets a bus clock of 24 MHz using internal reference clock. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH0 (PTC0 – J4_1) in Dual Edge Capture mode.
- Clock out = $24\text{MHz}/128 = 128\text{ kHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

```
COM15 - PuTTY
Running the FRDM_KEA32_ICS_FEI project
Dual edge capture end.
The input counts are 8
The input wave frequency is 128.0 kHz
```

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J4_1

2. Internal Clock Source (ICS)

Overview

- The Internal Clock Source module is used for main system clock generation. The ICS module controls which clock sources (internal references, external crystals or external clock signals) generate the source of the system clocks.



```
COM8 - PuTTY
Running the FRDM_KEA128_ICS_FEE project.
Dual edge capture end.
The input counts are 16
The input wave frequency is 156.25 kHz
```

FEE Example:

For, FRDM-KEAZ128:

- FLL Engaged External (FEE). Bus clock is set at 20 MHz using external crystal of 8MHz. Bus clock output is enabled on PTH2(J6_8) with a prescaler of 128. Measures this clock frequency on FTM2 CH4 (PTG6 –J1_13) in Dual Edge Capture mode.
- Clock out = $20\text{MHz}/128 = 156.25\text{ KHz}$.
- FTM2 captures both rising/falling edge of the input waveform and prints the counts and freq value on the terminal window.

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Connect J6_8 to J1_13

Library: ics.h& ics.c

- This library allows to select the reference clock that will be used for system clock generation. Also it allows to switch between 7 operation modes of the ICS : FEI, FEE, FBI, FBILP, FBE, FBELP and STOP.



Nested Vectored Interrupt Controller (NVIC)

Library: nvic.h& nvic.c

- This library can be used to mask and unmask the particular module's interrupt.
- Non-Maskable Interrupt (NMI) can not be masked.
- It includes functions to Enable and Disable interrupts from desired module.
- The other functions related to NVIC, like setting interrupt priority, can be found in “**core_cm0plus.h**” header.

PRINT

Library: print.h& print.c

- This library provides functions, like in_char(), printf, sprintf etc, to print/get the char/string, on the terminal window using UART2 as a communication medium.

3. Universal Asynchronous Receiver Transmitter (UART)

Overview

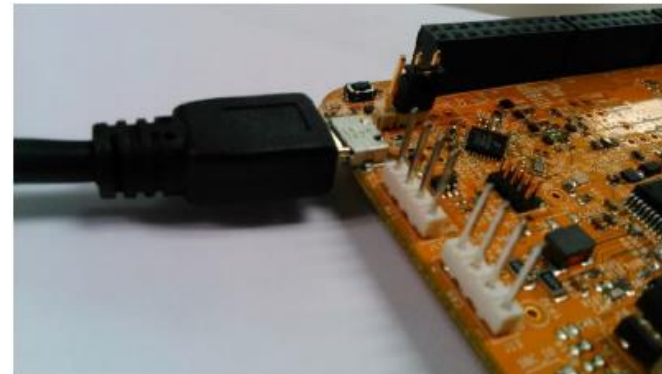
- The UART module inside the KEA MCUs allows the user to communicate with other devices via UART protocol.
- This UART module is also capable to communicate with other devices via LIN protocol.
- The UART comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. During normal operation, the MCU monitors the status of the UART, writes the data to be transmitted, and processes received data.

Hardware Connections:

- Setup Terminal window at baud rate of 115200

Library: `uart.h` & `uart.c`

- UART is the serial communication protocol. This library includes functions to initialize, send and receive data using Polling, Interrupt and Loopback mode.



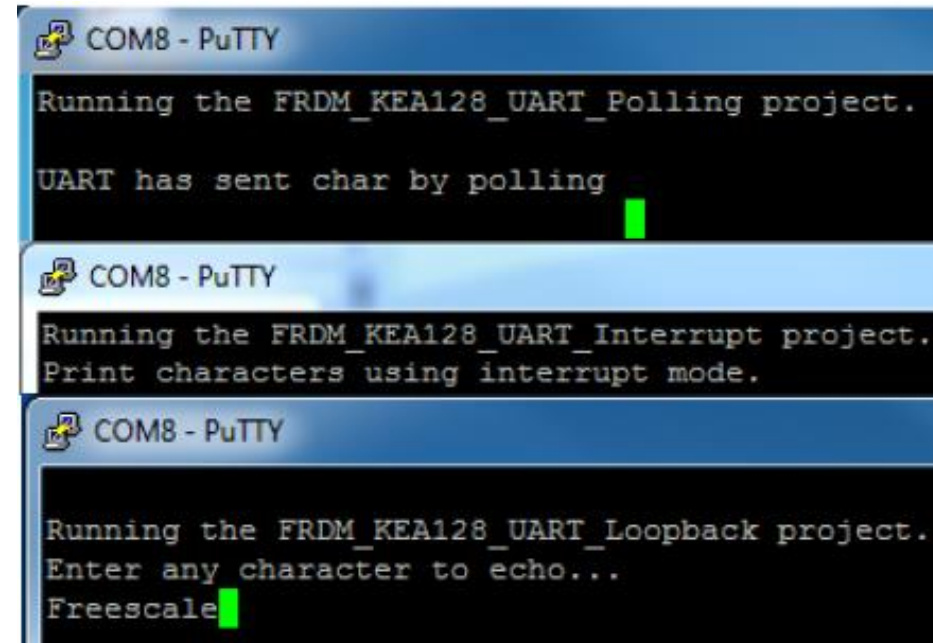
3. Universal Asynchronous Receiver Transmitter (UART)

Examples:

- **Polling:** This example demonstrates how to send characters using UART Polling.
- **Interrupt:** This example shows how to send and echo back the characters using UART interrupt.
- **Loopback:** This example shows the loopback mode in UART0 by creating an echo in the hyper terminal. When writing an character in the hyper terminal, UART2 will receive the char then the same character will be send and received by UART0 in loopback mode then char received by UART0 will be sent to UART2.

Hardware Connections:

- Setup Terminal window at baud rate of 115200



The image shows three stacked screenshots of a PuTTY terminal window titled 'COM8 - PuTTY'. The first screenshot shows the text 'Running the FRDM_KEA128_UART_Polling project.' followed by 'UART has sent char by polling' and a green cursor. The second screenshot shows 'Running the FRDM_KEA128_UART_Interrupt project.' followed by 'Print characters using interrupt mode.' and a green cursor. The third screenshot shows 'Running the FRDM_KEA128_UART_Loopback project.' followed by 'Enter any character to echo...' and 'Freescale' with a green cursor.

```
COM8 - PuTTY
Running the FRDM_KEA128_UART_Polling project.
UART has sent char by polling

COM8 - PuTTY
Running the FRDM_KEA128_UART_Interrupt project.
Print characters using interrupt mode.

COM8 - PuTTY
Running the FRDM_KEA128_UART_Loopback project.
Enter any character to echo...
Freescale
```

4. Key Board Interrupt (KBI)

Overview

- The KEAZ128/64 and KEAZN32 MCUs includes up to 64 and 16 interrupt pins respectively, divided in two instances of KBI, each of 32 and 8 keyboard interrupt pins respectively. KBI is useful for Human Machine Interface (HMI).



SW2 Pressed!



SW3 Pressed!



```
COM8 - PuTTY
Running the FRDM_KEA128_KBI project.
SW3 pressed!
SW2 pressed!
```

Example:

- This examples demonstrates how to configure the pins connected to SW2 & SW3 in KBI mode to generate the interrupt.
- Whenever a key is pressed the blue led will be toggled and the a message will be displayed on Terminal window as shown

Hardware Connections:

- Setup Terminal window at baud rate of 115200

Library: kbi.h& kbi.c

- This Library includes the function to Initialize, deinitialize and set up call back routine.
- This Library functions can se up KBI in falling-edge sensitivity only, rising-edge sensitivity only, both falling-edge and low-level sensitivity, both rising-edge and high-level sensitivity



5. FlexTimer Module (FTM)

Overview

- The KEAZ128/64 and KEAZN32 series MCUs includes 3 FlexTimers that represents up to 10 PWM/ input capture channels.
- The FlexTimer module is a 16 bit timer that supports input capture, output compare and generation of PWM signals, suitable to control electric motors and power management applications.
- Each channel can be configured for input capture, output compare, or edge-aligned PWM mode.
- PWM signal can be generated as edge-aligned, center-aligned, combine and complementary.

Application

- This timer is great option for BLDC motor control applications,
- DC motor control, read encoder signals, lighting control, RGB
- ledscontrol and much more.

Library: `ftm.h` & `ftm.c`

- This Library contains functions to configure FTM in all three modes by utilizing various functions available with FTM module.

5. FlexTimer Module (FTM)

Output Compare Example:

- When the counter matches the value in the CnVregister of an output compare channel, the channel (n) output can be set, cleared, or toggled. Otherwise the Internal subroutine can be called to perform some operations periodically.
- In this example FTM2 CH0 is configured to toggle REDLED connected to PTH0 at every 200ms.

EPWM Example:

- This example shows the dimming and brightening of Blue LED periodically using the Edge-Aligned Pulse Width Modulation. It basically changes value in CnVregister to generate different pulse width.

Hardware Connections:

- Setup Terminal window at baud rate of 115200



```
COM8 - PuTTY
Running the FRDM_KEA128_FTM_OutputCompare project.
FTM counter value.
0x927C
FTM counter value.
0x24F8
FTM counter value.
0xB774
FTM counter value.
0x49F0
FTM counter value.
0xDC6C

COM14 - PuTTY
Running the FRDM_KEA32_FTM_OutputCompare project.
FTM counter value.
0x6400
FTM counter value.
0xC800
FTM counter value.
0x2C00
```



5. FlexTimer Module (FTM)

Dual Edge Example:

For, FRDM-KEAZ128

This example demonstrates FTM dual edge capture feature.

FTM0 CH0 is configured in output compare mode & generates toggled waveform on PTA0. The period is 12000 FTM counts (2ms), positive pulse width is 6000 (1ms).

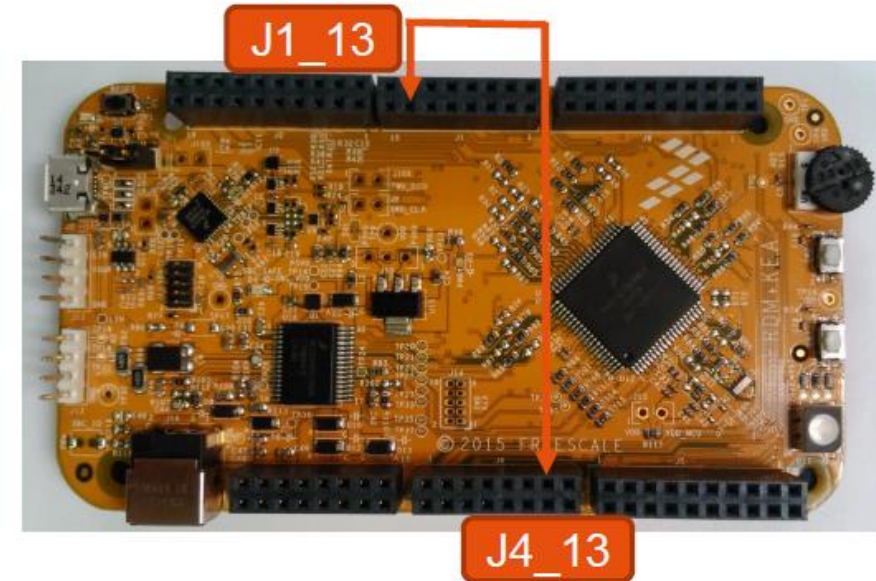
FTM2 is configured in Dual Edge Capture mode. The output generated at PTA0 is fed as an input of FTM2 CH4 and CH5 to measure the input counts and the wave period. FTM2 captures both rising/falling edge of the input waveform.

Hardware Connections:

Terminal : Same as of UART

Connect PTA0 – J4_13 to PTG6 – J1_13

Observe the Count value & period on terminal Window



```
COM8 - PuTTY
Running the FRDM_KEA128_FTM_DualEdge project.
Dual edge capture end. The input counts are 6000
The input wave period is 2 ms
```


5. FlexTimer Module (FTM)

Dual Edge Example:

For, FRDM-KEAZ64

This example demonstrates FTM dual edge capture feature.

FTM0 CH0 is configured in output compare mode & generates toggled waveform on PTA0. The period is 12000 FTM counts (2ms), positive pulse width is 6000 (1ms).

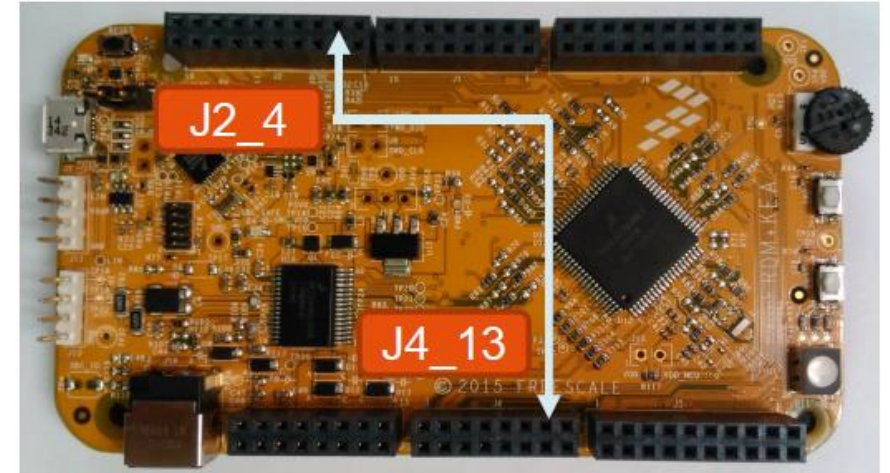
FTM2 is configured in Dual Edge Capture mode. The output generated at PTA0 is fed as an input of FTM2 CH0 and CH1 to measure the input counts and the wave period. FTM2 captures both rising/falling edge of the input waveform.

Hardware Connections:

Terminal : Same as of UART

Connect PTA0 – J4_13 to PTB4 – J2_4

Observe the Count value & period on terminal Window



```
COM16 - PuTTY
Running the FRDM_KEA64_FTM_DualEdge project.
Dual edge capture end. The input counts are 6000

The input wave period is 2 ms
```

5. FlexTimer Module (FTM)

Dual Edge Example:

For, FRDM-KEAZN32

This example demonstrates FTM dual edge capture feature.

FTM0 CH0 is configured in output compare mode & generates toggled waveform on PTB2. The period is 8196 FTM counts (2ms), positive pulse width is 4098 (1ms).

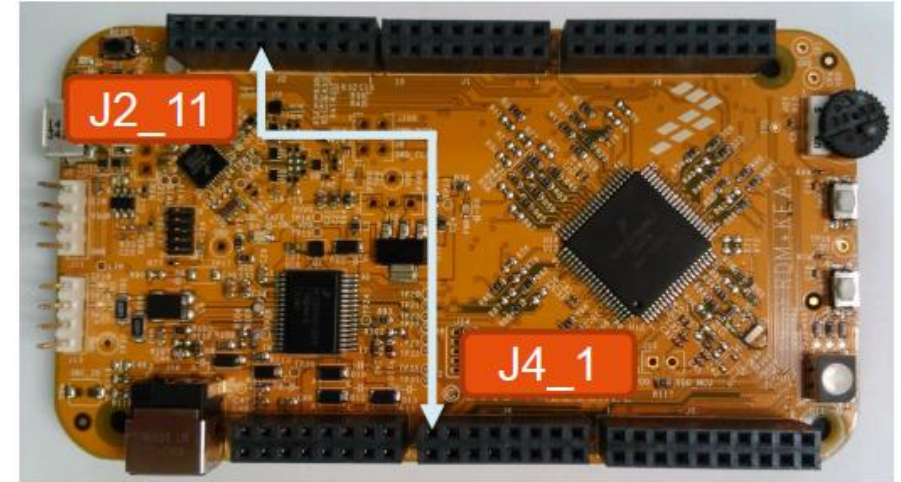
FTM2 is configured in Dual Edge Capture mode. The output generated at PTB2 is fed as an input of FTM2 CH0 and CH1 to measure the input counts and the wave period. FTM2 captures both rising/falling edge of the input waveform.

Hardware Connections:

Terminal : Same as of UART

Connect PTB2 – J2_11 to PTC0 – J4_1

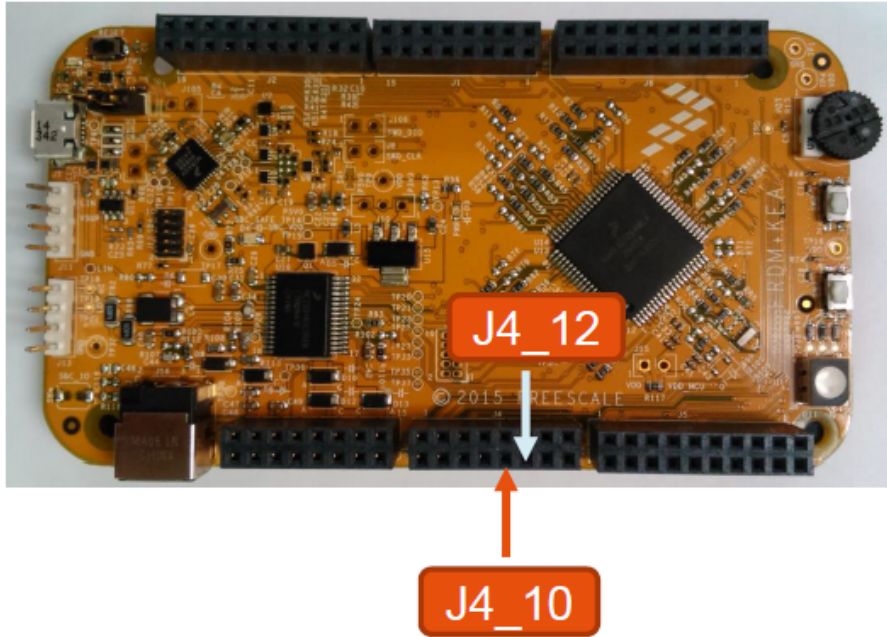
Observe the Count value & period on terminal Window



```
COM14 - PuTTY
Running the FRDM_KEA32_FTM_DualEdge project.
Dual edge capture end. The input counts are 4098

The input wave period is 2 ms
```


5. FlexTimer Module (FTM)



Combine Example:

This example shows how to generate PWM signal
Combining two channels.

FTM2 Ch0 and adjacent Ch1 are combined to generate
PWM signal on RED LED (PTH0) with positive pulse width
is of 4.5 ms.

BLUE LED (PTE7 – J4_12) will toggle every 100ms and at
the start of PWM signal.

Use an oscilloscope to measure the PWM signal of RED
LED (PTH0 – J4_10)

Hardware Connections:

Terminal : Same as of UART

Observe the RED and BLUE LED

Use Oscilloscope to measure PWM signal.

6. Analog to Digital Converter(ADC)

Overview:

The KEAZ28/64 and KEAZN32 series MCUs include a 12 bit analog to digital converter (ADC). The ADC implements a linear successive approximation algorithm with 8-, 10- or 12-bit resolution

The KEAZ28/64 and KEAZN32 includes up to 16 external analog inputs, external pin inputs, and 5 internal analog inputs including internal band gap, temperature sensor, and references.



Potentiometer

Library: adc.h & adc.c

This Library includes the function to initialize and de initialize ADC.

It also includes functions to read value, to set up channel, for voltage reference select, for clock divide and for interrupt call back



6. Analog to Digital Converter(ADC)

Example:

- These examples demonstrate how to configure the ADC CH10 in 12-bit resolution mode to read and display the ADC value on the terminal screen.
- **Polling:** In this example MCU continuously reads the Potentiometer, but waits until the user press any key on the computer keyboard. As soon as the key is pressed, the values are displayed on the terminal window.
- **Interrupt:** This demo generate an interrupt whenever the ADC module finishes reading Potentiometer. The conversions are shown on the terminal window. It also waits for input from Keyboard.
- **FIFO:** This demo reads the Potentiometer, VREFH and VREFL and saving the conversion in an array that is shown on the screen when the user presses a key on computer keyboard

Hardware Connections:

- Setup Terminal window at baud rate of 115200
- Vary the Potentiometer for Different values

```
Running the FRDM_KEA128_ADC_Polling project.  
POTENTIOMETER conversion value:0x7E1  
Input any character to start a new conversion!
```

```
Running the FRDM_KEA128_ADC_Interrupt project.  
ADC conversion result as below:  
POTENTIOMETER conversion value:0x7E0  
Input any character to start a new conversion!
```

```
Running the FRDM_KEA128_ADC_FIFO project.  
ADC conversion result as below  
POTENTIOMETER conversion value:0x7E3  
VREFH conversion value:0xFFC  
VREFL conversion value:0x0  
Input any character to start a new conversion!
```


7. Analog Comparator (ACMP)

Overview

- Two analog comparators are included in the Kinetis EA. The analog comparator (ACMP) is a module inside the KEA that provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage. The ACMP can achieve analog comparison between positive input and negative input, producing a digital output and/or an interrupt afterwards.

Library: `acmp.h` & `acmp.c`

- This library includes functions to initialize, deinitialize and for interrupt callback routine

Example

- ACMP is configured in Interrupt mode.
- Changing the voltage level of ACMP external pin0 -PTA6 (J5_7) between VDD (J5_10) and GND (J5_12), ACMP ISR happens.
- Every time that the external voltage on PTA6 cross the 2.5V threshold an interrupt occurs and "Callback happens" message will appear on the terminal window.

Hardware Connections:

- Terminal : Same as of UART
- Discrete Potentiometer connected to VDD, PTA6, and GND



8. Periodic Interrupt Timer (PIT)

Overview:

The Kinetis EA series MCUs includes two 32-bit timer. The periodic interrupt timer is a 32 bit timer that can be used to raise interrupts and triggers. Both 32-bit timer can be chained to generate a 64-bit timer.

Library: pit.h & pit.c

This library includes functions to initialize, deinitialize and for interrupt callback routine

Example:

This demo sets up PIT timer 1 in chain mode with PIT timer 0 to generate interrupt each 1s. PIT timer 0 runs 2,400,000 cycle, PIT timer 1 run 10 cycle, so total cycle is 24,000,000.

Bus clock in the code is 24MHz, so it generates interrupt every 1s.

So, BLUE LED will toggle at every 1s.

Note: To see GREEN LED toggling at every 100ms, uncomment the corresponding code and comment the code for BLUE LED.



9. Real Time Clock (RTC)

Overview:

The Kinetis EA series MCUs includes one Real-Time Counter (RTC). The RTC consists of one 16 – bit counter, one 16 – bit comparator, several binary-based and decimal-based prescaler dividers, three clock sources, one programmable periodic interrupt, and one programmable external toggle pulse output.

Library: rtc.h & rtc.c

This library includes functions to initialize, deinitialize and for interrupt callback routine

Example:

RTC produces 1s interrupt with LPO clock source, then toggle BLUE LED (PTC0) to generate blinking pattern.

Hardware Connections:

Terminal : Same as of UART

Applications:

This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake-up from low-power modes.

10. Pulse Width Timer (PWT) (KEAZ128/64 Only)

Overview:

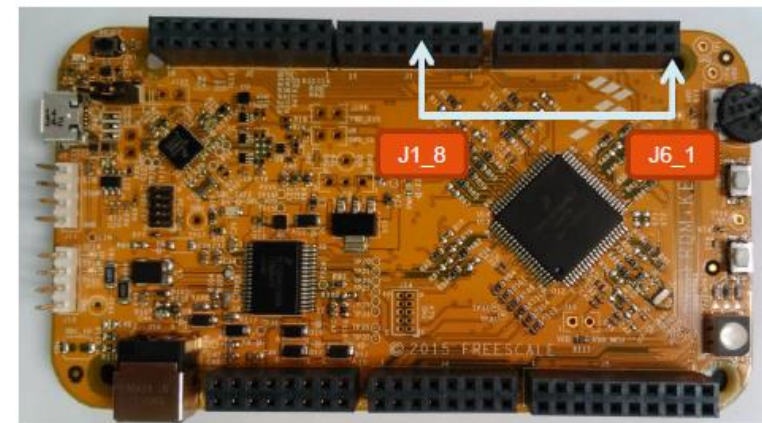
The Kinetis EA series MCUs includes a Pulse Width Timer (PWT), that allows you to automatically measure pulse widths in 16-bit resolution. Some of the features of this PWT are: programmable triggering edge for starting measurement, four selectable pulse inputs, separate positive and negative pulse width measurements and much more.

Note: This Feature is not available with FRDM-KEAZN32

Library: pwt.h & pwt.c

This library includes functions to initialize, deinitialize and for interrupt callback routine

```
Running the FRDM_KEA128_PWT project.  
Positive Pulse Width =18751, Negative Pulse Width =18751, Frequency = 9.9 Hz  
Positive Pulse Width =18752, Negative Pulse Width =18751, Frequency = 9.9 Hz
```



Example:

This example generates a toggle pulse on PTD0 (J6_1) at 10 Hz frequency with RTC library. The period and frequency is measured with PWT library on pin PTD5 (J1_8).

Hardware Connections:

Terminal : Same as of UART

Connecting J6_1 with J1_8 will generate a message

It prints Positive Pulse Width, Negative Pulse Width and Frequency on Terminal window

11. Watchdog Timer (WDOG)

Overview

- The Kinetis EA series MCUs includes a Watchdog timer (WDOG). The WDOG module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the MCU.

Example

- The watchdog is initialized to be enabled in low power state as well as in debug state and generate watchdog reset at around 1s.
- It toggles BLUE LED periodically. When a NMI pin is asserted to ground the processor goes into infinite loop and it will then generate watchdog reset so the RED LED will blink.



Library: `acmp.h` & `acmp.c`

- This library includes functions to initialize, deinitialize and for interrupt callback routine

Hardware Connections:

- Terminal : Same as of UART
- To create a NMI interrupt, first load the KEA with the WDOG program. Disconnect board from computer and reconnect so that it is in run mode. Connect PTB4 (J2_4) to the ground (J2_13) for just a split moment, otherwise MCU can sense NMI and go to infinite loop before watchdog initialization and one may never see RED LED blinking. If the first time, execution just pauses upon NMI interrupt, push the reset button (SW1) or power-on-reset (POR) and repeat test.



12. Serial Peripheral Interface (SPI)

Overview:

The Kinetis EA series MCUs includes up to 2 instances of SPI. The serial peripheral interface(SPI) module provides for full-duplex, synchronous, serial communication between the MCU and peripheral devices.

SPI module features are: master or slave operation, serial clock phase and polarity options, receive data buffer hardware match feature, among others.

Library: spi.h & spi.c

This library includes functions to initialize & deinitialize SPI module. It also includes functions to run SPI using interrupt or polling, in both Master and Slave configuration.

Application:

This Library can be used to communicate between other MCUs, sensors, memories, ADCs, shift registers etc.

Examples:

Master Interrupt: This example provides a template for SPI master transfer with interrupt mode. It generates an echo with the other board board using the Slave project. The Master sends a counter value and waits for the slave to respond with the same value to increase the counter.

Blue Light indicates, the board is working as master.

12. Serial Peripheral Interface (SPI)

Example:

Master Polling: This project as well sends a counter value and waits for the slave to respond with the same value to increase the counter. But this is done by polling method.

Blue Light indicates, the board is working as master.

Slave: In this project the slave responds to the master with the same value received from master.

RED light indicates, the board is working as slave.

For all projects: After receiving that value it prints that value on the terminal window using UART.

Hardware Connections:

Terminals : Same as of UART

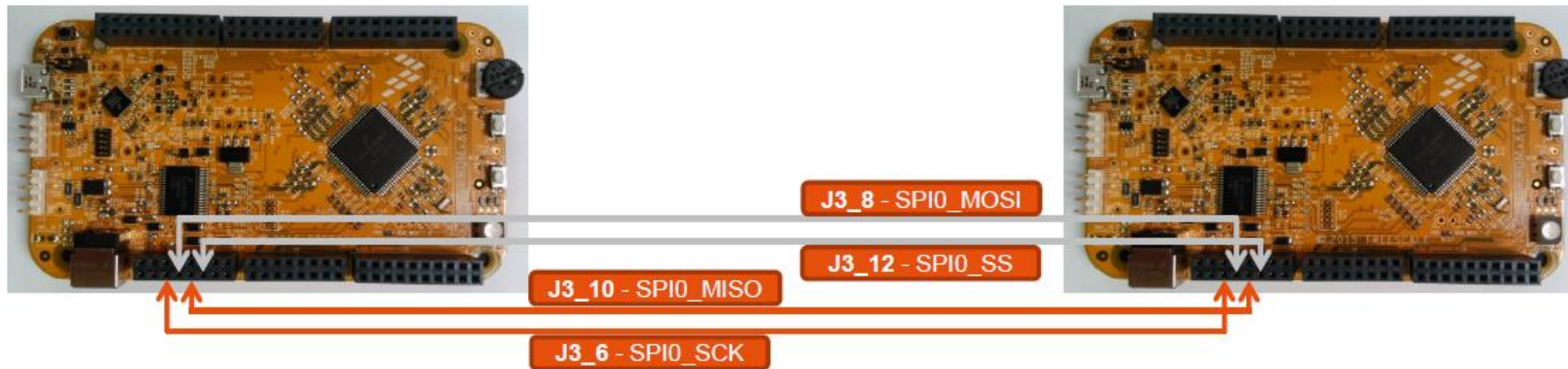
Connect both, master & slave board's following pins.

J3_6 – PTE0 – SPI0_SCK

J3_8 – PTE1 – SPI0_MOSI

J3_10 – PTE2 – SPI0_MISO

J3_12 – PTE3 – SPI0_SS



13. Inter-Integrated Circuit (I2C)

Overview:

The Kinetis EA series MCUs includes up to 2 instances of I2C. The inter-integrated circuit (I2C or IIC) module provides a method of communication between numbers of devices.

Some of the I2C module features are: multi-master operation, software-selectable acknowledge bit, calling address identification interrupt

Library: i2c.h & i2c.c

This library includes functions to initialize & deinitialize I2C module. It also includes library for Starting, stopping, reading data and writing data in Interrupt as well as polling mode and many more functions.

Applications:

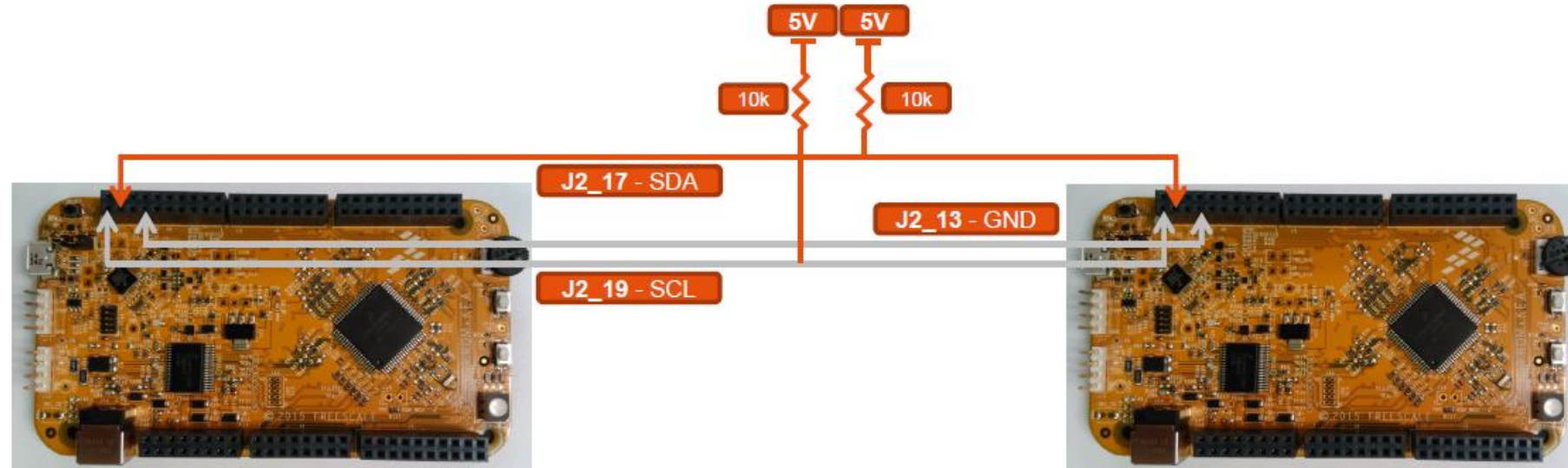
This Library can be used to communicate with different sensors like gyroscopes, accelerometers, pressure sensors and more.

Examples:

Master Interrupt: This example provides a template for I2C master with interrupt mode.

It sends the data to slave and receives the data from slave. It prints the both data on terminal window using UART.

13. Inter-Integrated Circuit (I2C)



Examples:

Master Polling: this example does the same as of “Master Interrupt” Example but using polling.

Slave: Slave receives the data from the master and sends back the same data to the master. It uses Interrupts.

For all projects: After receiving that value it prints that value on the terminal window using UART.

Hardware Connections:

Terminals : Same as of UART

Connect both, master & slave board's following pins.

J2_17 – PTA2 – SDA

J2_19 – PTA3 – SCL

J2_13 – GND

Place Pull up resistors of 10kΩ between connections

14. Aliased Bit-Band Domain (KEAZ128/64 Only)

Overview:

The on-chip SRAM is split into two ranges: SRAM_L and SRAM_U. The SRAM_U range supports bit operation on this device through two ways:

- Aliased bit-band region
- Bit Manipulation Engine (BME)

Here we will discuss the aliased bit-band region.

The device supports aliased SRAM_U bit-band region with Cortex M0+ core. A 32-bit write in the alias region has the same result as a read-modify-write operation on the targeted bit in the bit-band region, but with only one cycle time. Aliased bit-band region is much more efficient for bit operation. The aliased bit-band region only supports simple set or clear operation.

Library: bitband.h

This library includes function to initialize pointer to invoke the pointer to access aliased Bit -Band domain.

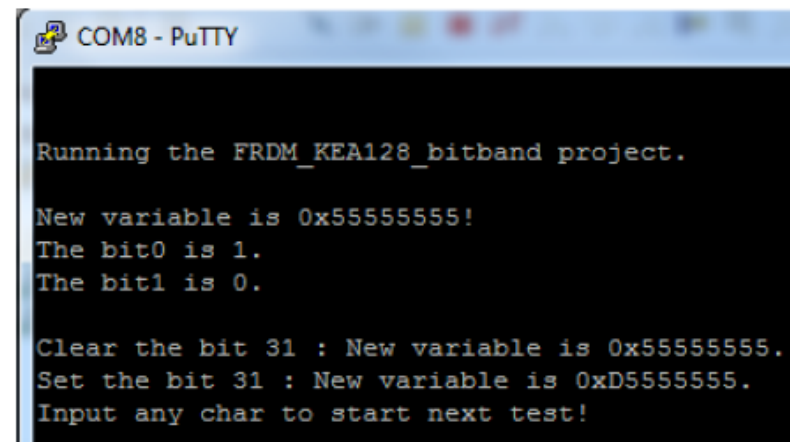
Example:

This example demonstrates the bit wise read-modify-write operations in the SRAM_U region. The 0th and 1st bit of variable (stored in SRAM_U region) is read, then new value is written to 31st bit and at the end the variable's value is increased by one.

Hardware Connections:

Terminal : Same as of UART

Observe the output on the screen



```
COM8 - PuTTY

Running the FRDM_KEA128_bitband project.

New variable is 0x55555555!
The bit0 is 1.
The bit1 is 0.

Clear the bit 31 : New variable is 0x55555555.
Set the bit 31 : New variable is 0xD5555555.
Input any char to start next test!
```

15. Bit Manipulation Engine (BME)

Overview:

The other module supporting the bit operations on these devices is Bit Manipulation Engine (BME)

The Bit Manipulation Engine (BME) provides hardware support for atomic read-modify-write memory operations to the peripheral address space in Cortex-M0+ based microcontrollers.

This architectural capability is also known as "decorated storage" as it defines a mechanism for providing additional semantics for load and store operations to memory mapped peripherals beyond just the reading and writing of data values to the addressed memory locations.

Library: bme.h

This library contains macros for AND, OR and XOR operations. It has also macros for Set and clear bits. It has also macros for inserting or extracting bit fields. All these functions are available for 32-bit, 16-bit and 8-bit data sizes.

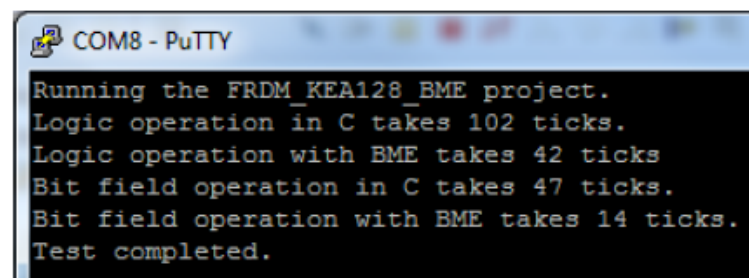
Example:

This demo compares the bit manipulation using simple C code and BME. It has examples for simple read-modify-write operation on one bit and bit field using simple C and BME. And results of both method is compared using system tick timer module in terms of time.

Note: the compiler optimization level will affect the code execution time; it is recommended that high level optimization for speed to be used.

Hardware Connections:

Terminal : Same as of UART



```
COM8 - PuTTY
Running the FRDM_KEA128_BME project.
Logic operation in C takes 102 ticks.
Logic operation with BME takes 42 ticks
Bit field operation in C takes 47 ticks.
Bit field operation with BME takes 14 ticks.
Test completed.
```

16. Cyclic Redundancy Check (CRC)

Overview:

The cyclic redundancy check (CRC) module generates 16/32-bit CRC code for error detection.

The CRC module provides a programmable polynomial, WAS, and other parameters required to implement a 16-bit or 32-bit CRC standard.

The 16/32-bit code is calculated for 32 bits of data at a time.

Library: `crc.h` & `crc.c`

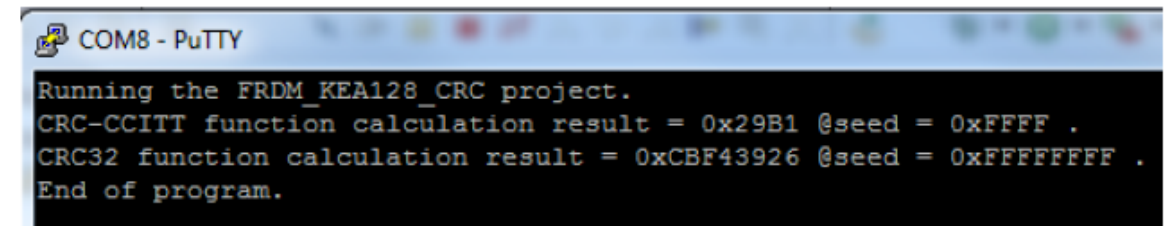
This library includes functions to initialize & deinitialize CRC module. It also includes library for 16 and 32-bit of CRC code calculations.

Example:

This example demonstrates the working of CRC module in 16-bit and 32-bit mode. It generates the 16 and 32 bit CRC results for a given data.

Hardware Connections:

Terminal : Same as of UART

A screenshot of a terminal window titled "COM8 - PuTTY". The terminal displays the following text:

```
Running the FRDM_KEA128_CRC project.  
CRC-CCITT function calculation result = 0x29B1 @seed = 0xFFFF .  
CRC32 function calculation result = 0xCBF43926 @seed = 0xFFFFFFFF .  
End of program.
```

17. Flash Memory Module (FLASH)

Overview

- The flash memory is ideal for single-supply applications allowing for field reprogramming without requiring external high voltage sources for program or erase operations. The flash module includes a memory controller that executes commands to modify flash memory contents. It has automated program and erase algorithm with verification, fast sector erase and long word program operation and flexible protection scheme to prevent accidental programming or erasing of flash memory.
- It is called FTMRH in KEA64 sub-family and FTMRE in KEA128 sub-family.

Library: acmp.h & acmp.c

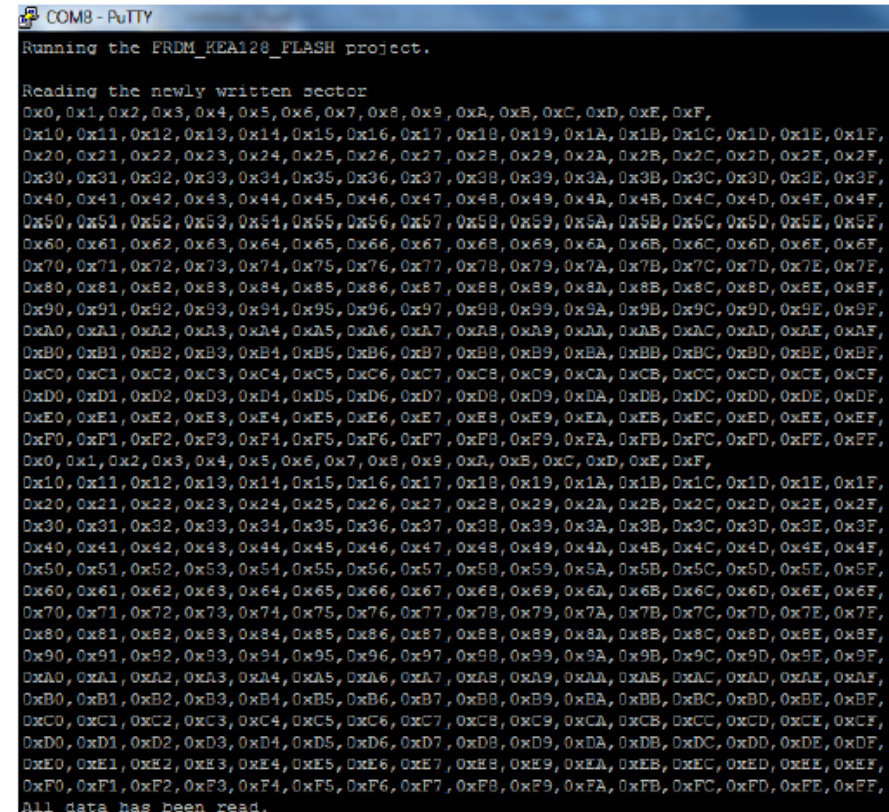
- This library includes functions to initialize, erase, verify and read flash. It also includes library to program flash in different word length and many more functions.

Example

- Here the sector on the Flash memory is erased and the new data has been written. And then the sector is read again to ensure the data is written perfectly

Hardware Connections:

- Terminal : Same as of UART



```
COM8 - PuTTY
Running the FRDM_KEA128_FLASH project.

Reading the newly written sector
0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xA,0xB,0xC,0xD,0xE,0xF,
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,
0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,
0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,0x3D,0x3E,0x3F,
0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C,0x4D,0x4E,0x4F,
0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5A,0x5B,0x5C,0x5D,0x5E,0x5F,
0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,0x6B,0x6C,0x6D,0x6E,0x6F,
0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A,0x7B,0x7C,0x7D,0x7E,0x7F,
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,
0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0x9D,0x9E,0x9F,
0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,
0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,
0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,
0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF,
0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,
0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF,
0x0,0x1,0x2,0x3,0x4,0x5,0x6,0x7,0x8,0x9,0xA,0xB,0xC,0xD,0xE,0xF,
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1A,0x1B,0x1C,0x1D,0x1E,0x1F,
0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,0x28,0x29,0x2A,0x2B,0x2C,0x2D,0x2E,0x2F,
0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39,0x3A,0x3B,0x3C,0x3D,0x3E,0x3F,
0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,0x48,0x49,0x4A,0x4B,0x4C,0x4D,0x4E,0x4F,
0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,0x58,0x59,0x5A,0x5B,0x5C,0x5D,0x5E,0x5F,
0x60,0x61,0x62,0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x6A,0x6B,0x6C,0x6D,0x6E,0x6F,
0x70,0x71,0x72,0x73,0x74,0x75,0x76,0x77,0x78,0x79,0x7A,0x7B,0x7C,0x7D,0x7E,0x7F,
0x80,0x81,0x82,0x83,0x84,0x85,0x86,0x87,0x88,0x89,0x8A,0x8B,0x8C,0x8D,0x8E,0x8F,
0x90,0x91,0x92,0x93,0x94,0x95,0x96,0x97,0x98,0x99,0x9A,0x9B,0x9C,0x9D,0x9E,0x9F,
0xA0,0xA1,0xA2,0xA3,0xA4,0xA5,0xA6,0xA7,0xA8,0xA9,0xAA,0xAB,0xAC,0xAD,0xAE,0xAF,
0xB0,0xB1,0xB2,0xB3,0xB4,0xB5,0xB6,0xB7,0xB8,0xB9,0xBA,0xBB,0xBC,0xBD,0xBE,0xBF,
0xC0,0xC1,0xC2,0xC3,0xC4,0xC5,0xC6,0xC7,0xC8,0xC9,0xCA,0xCB,0xCC,0xCD,0xCE,0xCF,
0xD0,0xD1,0xD2,0xD3,0xD4,0xD5,0xD6,0xD7,0xD8,0xD9,0xDA,0xDB,0xDC,0xDD,0xDE,0xDF,
0xE0,0xE1,0xE2,0xE3,0xE4,0xE5,0xE6,0xE7,0xE8,0xE9,0xEA,0xEB,0xEC,0xED,0xEE,0xEF,
0xF0,0xF1,0xF2,0xF3,0xF4,0xF5,0xF6,0xF7,0xF8,0xF9,0xFA,0xFB,0xFC,0xFD,0xFE,0xFF,
All data has been read.
```



18. Power Management Controller (PMC)

Overview

- The PMC provides the user with multiple power options. Multiple modes are supported that allow the user to optimize power consumption for the level of functionality needed. Includes power on reset (POR) and integrated low voltage detect (LVD) with reset (brownout) capability and selectable LVD trip points.

Library: pmc.h & pmc.c

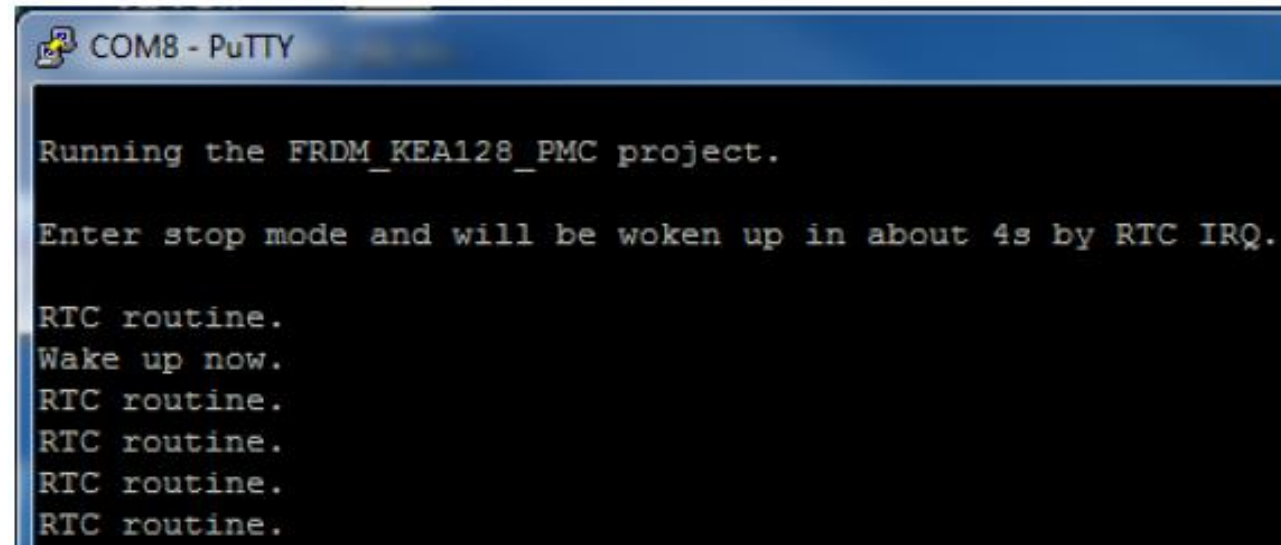
- This library includes functions to initialize & deinitialize PMC module. It also includes library for setting it in different modes (run, wait and stop modes).

Example

- In this example, configuring PMC, puts MCU in stop mode and LVD is still on in stop mode. But, RTC is also on so the RTC Interrupt will wake up MCU after 4 seconds.

Hardware Connections:

- Terminal: Same as of UART
- Observe the Blue LED toggling
- Observe that even though printf nWake up now. n”) is in the is in the sequence of the program it is executed only after wake up initiated by RTC module.



```
COM8 - PuTTY

Running the FRDM_KEA128_PMC project.

Enter stop mode and will be woken up in about 4s by RTC IRQ.

RTC routine.
Wake up now.
RTC routine.
RTC routine.
RTC routine.
RTC routine.
```

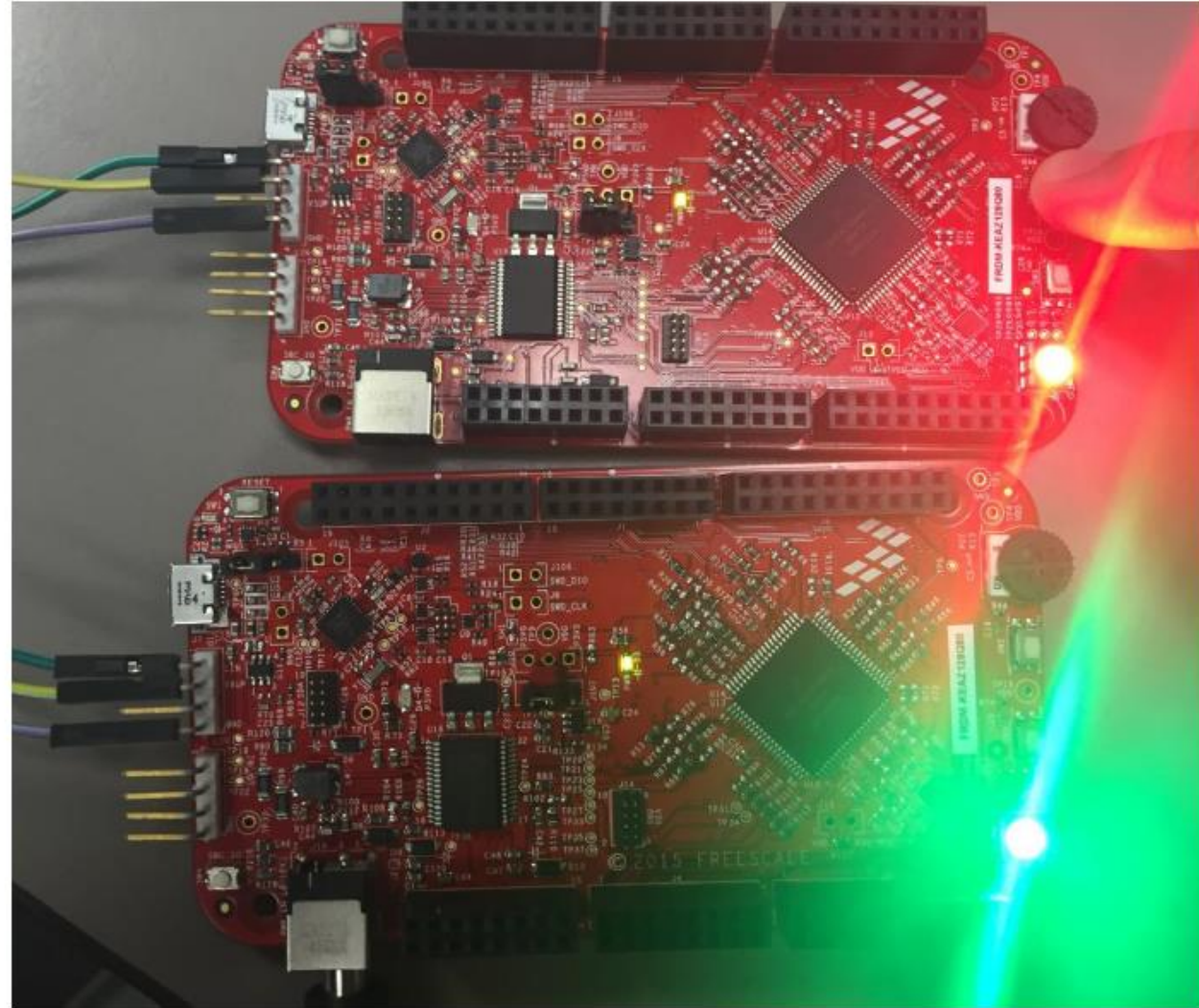
19. Local Interconnect Interface (LIN) (KEAZ1284/64 Only)

Overview:

The Local Interconnect Network protocol is a cost effective communications method between individual ECUs, similar to CAN. It is popular among automotive manufacturers for non-safety critical applications because of its low cost and hardware requirements compared to CAN.

Library: LIN Driver UART

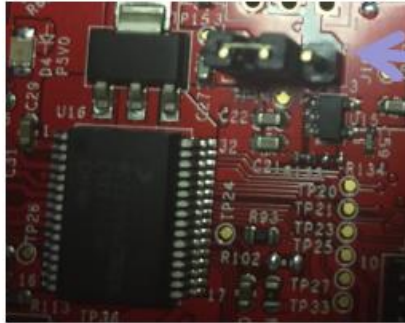
The LIN driver headers and source files are in the “LIN_Driver_UART” subfolder within the SDK drivers folder of the Quick Start Package. They are separate from the “headers” and “sources” folder because the complexity of LIN protocol requires more than a single header and source file. There is no LIN peripheral in KEA like there are for SPI or I2C; it uses the UART in a way that conforms to LIN standard.



19. Local Interconnect Interface (LIN) (KEAZ1284/64 Only)



Example:



This example requires two FRDM KEA EVBs. One is loaded with the LIN Master program while the other board runs the LIN Slave code. Wire the LIN terminals of the respective boards together then plug an external power supply to either master or slave. Watch the slave's LED light red in response to the potentiometer reading of the master; and watch the master's green and blue LEDs light in when SW2 and SW3 of the slave, respectively, are pressed.

Hardware Connections:

Connect both, master & slave board's following pins.

J11_1 – LIN

J11_2 – Vsup

J11_4 – GND

Jumper: J107_1 and J107_2 – Turns on System Basis Chip to run on external power supply



SECURE CONNECTIONS
FOR A SMARTER WORLD