ECE 491 Lab 2

GPIO's, ADC's and Serial Interface

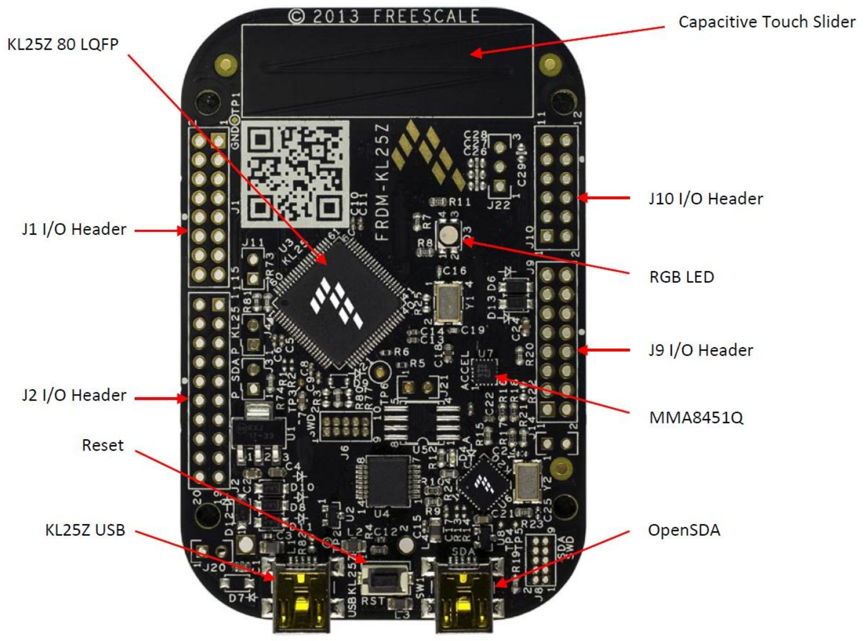
1. Introduction:

The purpose of this lab is for the groups to get familiar with Freedom Board KL25z GPIO's, ADC's and serial debugging using the Kinetics Design Studio (KDS) Integrated Development Environment (IDE) and Processor Expert (PE) for rapid code development. The groups will demonstrate the following for checkoff

- Switching the On-board LED's
- Switching an external LED using any available freedom board GPIO
- Reading a voltage using available 16 bit ADC
- Printing on a terminal (putty) the value of read voltage.

2. Background and Tools:

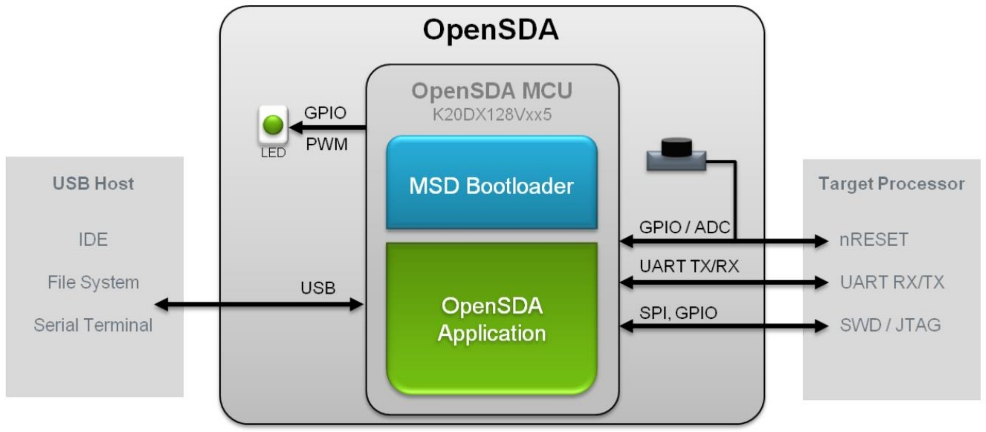
a) Freedom Board FRDM-KL25Z:



To learn more about the freedom board FRDM-KL25Z refer to user guide and pinout documents on blackboard. (Files FRDM-KL25Z User Manual (Rev 2).pdf and FRDM-KL25Z pinouts (Rev 1.0).pdf). You should always refer to these documents for relevant pinouts and schematics before you make your connections in order to protect your board.

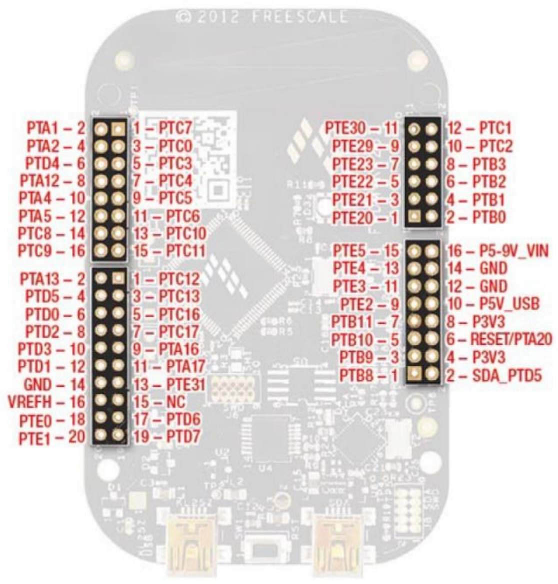
b) Serial and Debug Adapter (OpenSDA):

OpenSDA is an open-standard serial and debug adapter. It bridges serial and debug communications between a USB host and an embedded target processor as shown in figure. The hardware circuit is based on a Freescale Kinetis K20 family microcontroller (MCU) with 128 KB of embedded flash and an integrated USB controller. OpenSDA features a mass storage device (MSD) bootloader, which provides a quick and easy mechanism for loading different OpenSDA Applications such as flash programmers, run-control debug interfaces, serial-to-USB converters, and more. Refer to the OpenSDA User's Guide for more details.



c) Input/Output Connectors:

The KL25Z128VLK4 microcontroller is pac
in an 80-pin LQFP. Some pins are utilized
board circuitry, but many are directly con
to one of four I/O headers.
The pins on the KL25Z microcontroller are
for their general purpose input/output po
function. For example, the pin 1 on Port A is
referred to as PTA1. The I/O connector pin
names are given the same name as the K
connected to it, where applicable. Note th
pinout data is available in spreadsheet fo
FRDM-KL25Z Pinouts. See the Reference
Documents section for details.



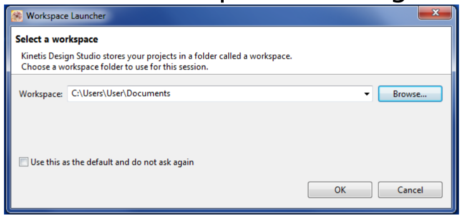
d) Kinetis Design Studio (KDS) with Processor Expert:

The Kinetis Design Studio (KDS) is used for development which is an integrated development environment for Kinetis MCUs. It enables robust editing, compiling and debugging of your designs shall use the Processor Expert integrated in KDS IDE to generate your codes with its knowledge ba helps create powerful applications with a few mouse clicks.

Steps for starting a new project in KDS with Processor expert.

Please follow the following steps to start a new project in KDS.

1. Click on Kinetics Design Studio 3.0.0 IDE on the desktop
2. Select a workspace making sure it points to a location on your system why is NOT read-only



Workspace Launcher

Select a workspace

Kinetis Design Studio stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

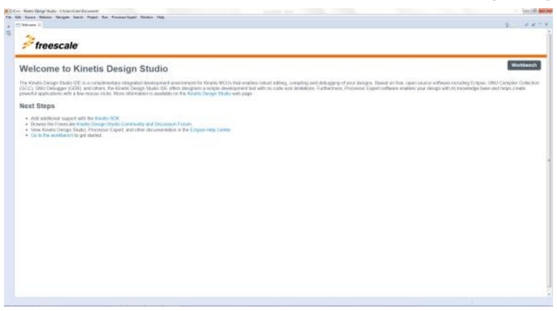
Workspace: C:\Users\User\Documents

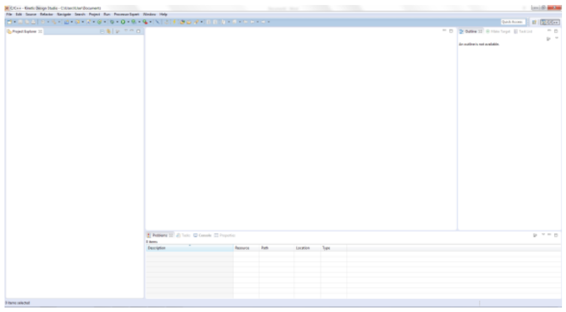
Browse...

☐ Use this as the default and do not ask again

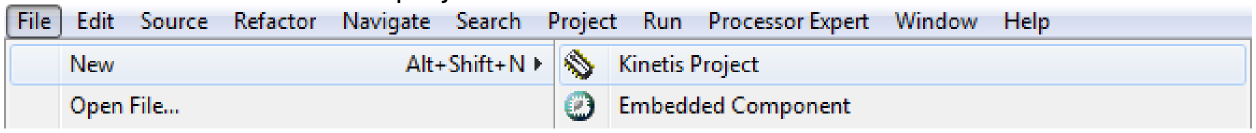
OK Cancel

3. Click on the workbench on the top-right to start

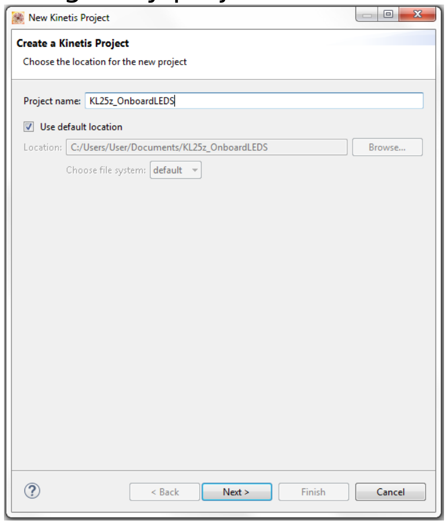




4. Click on File→ New→ Kinetis project



5. Assign any project name and click Next



New Kinetis Project

Create a Kinetis Project

Choose the location for the new project

Project name: KL25z_OnboardLEDS

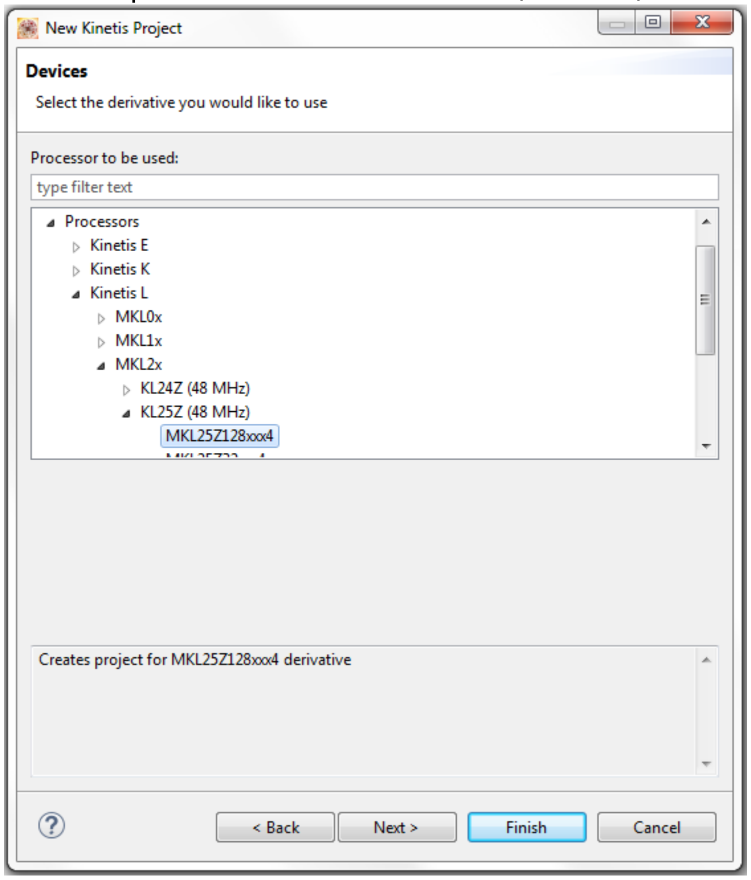
☑ Use default location

Location: C:/Users/User/Documents/KL25z_OnboardLEDS Browse...

Choose file system: default

< Back Next > Finish Cancel

6. Click on processors→Kinetis L→KL25Z (48 MHz)→MKL25Z128xx4 and then click Next



New Kinetis Project

Devices

Select the derivative you would like to use

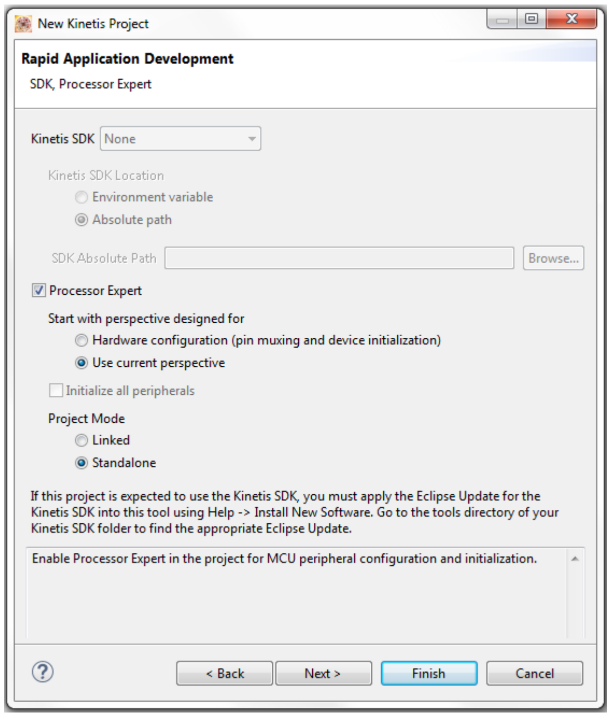
Processor to be used:

type filter text

- Processors
  - Kinetis E
  - Kinetis K
  - Kinetis L
    - MKL0x
    - MKL1x
    - MKL2x
      - KL24Z (48 MHz)
      - KL25Z (48 MHz)
        - MKL25Z128xxx4

Creates project for MKL25Z128xxx4 derivative

7. Select Processor Expert as Rapid Application Development and leave the options as is and click Next



New Kinetis Project

Rapid Application Development

SDK, Processor Expert

Kinetis SDK: None

Kinetis SDK Location
- Environment variable
- [✓ Absolute path]

SDK Absolute Path

[✓ Processor Expert]

Start with perspective designed for
- Hardware configuration (pin muxing and device initialization)
- [✓ Use current perspective]

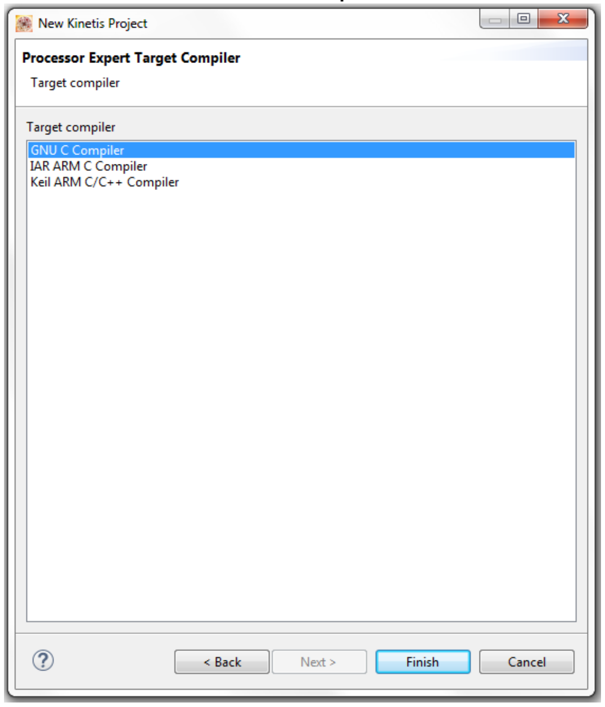
Initialize all peripherals

Project Mode
- Linked
- [✓ Standalone]

If this project is expected to use the Kinetis SDK, you must apply the Eclipse Update for the Kinetis SDK into this tool using Help -> Install New Software. Go to the tools directory of your Kinetis SDK folder to find the appropriate Eclipse Update.

Enable Processor Expert in the project for MCU peripheral configuration and initialization.

8. Select the default compiler as GNU C Compiler and click Finish



New Kinetis Project

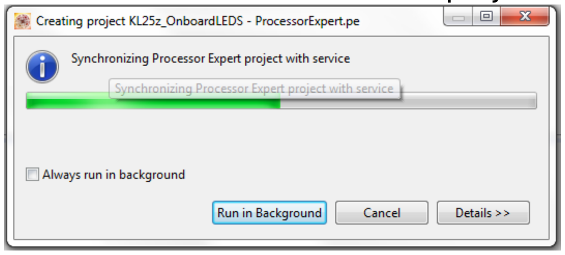
Processor Expert Target Compiler

Target compiler

Target compiler

GNU C Compiler
IAR ARM C Compiler
Keil ARM C/C++ Compiler

9. Wait until the IDE creates the project



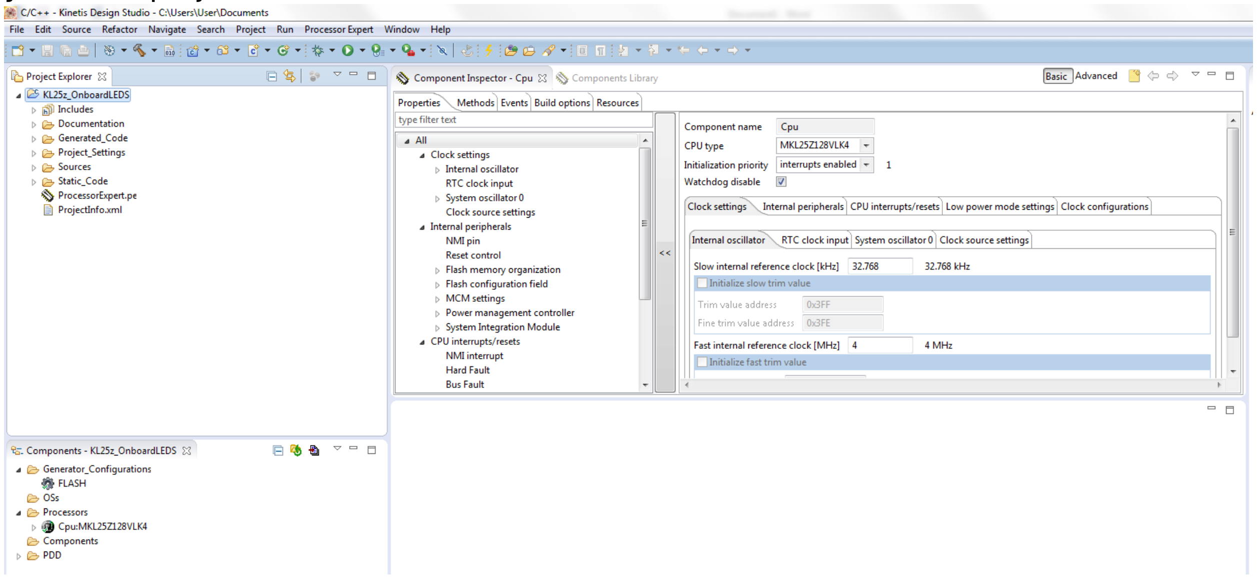
Creating project KL25z_OnboardLEDS - ProcessorExpert.pe

Synchronizing Processor Expert project with service

Synchronizing Processor Expert project with service

Always run in background

10. Click on project in the Project Explorer to reveal details as shown and you are ready to create your own projects.



C/C++ - Kinetis Design Studio - C:\Users\User\Documents

Project Explorer:

- KL25z_OnboardLEDS
  - Includes
  - Documentation
  - Generated_Code
  - Project_Settings
  - Sources
  - Static_Code
  - ProcessorExpert.pe
  - ProjectInfo.xml

Components - KL25z_OnboardLEDS:

- Generator_Configurations
  - FLASH
- OSs
- Processors
  - Cpu:MKL25Z128VLK4
- Components
- PDD

Properties tree:

- All
  - Clock settings
    - Internal oscillator
    - RTC clock input
    - System oscillator 0
    - Clock source settings
  - Internal peripherals
    - NMI pin
    - Reset control
    - Flash memory organization
    - Flash configuration field
    - MCM settings
    - Power management controller
    - System Integration Module
  - CPU interrupts/resets
    - NMI interrupt
    - Hard Fault
    - Bus Fault

Component name	Cpu	
CPU type	MKL25Z128VLK4	
Initialization priority	interrupts enabled	1
Watchdog disable	✓	

Slow internal reference clock [kHz]	32.768	32.768 kHz
Initialize slow trim value		
Trim value address	0x3FF	
Fine trim value address	0x3FE	
Fast internal reference clock [MHz]	4	4 MHz
Initialize fast trim value		

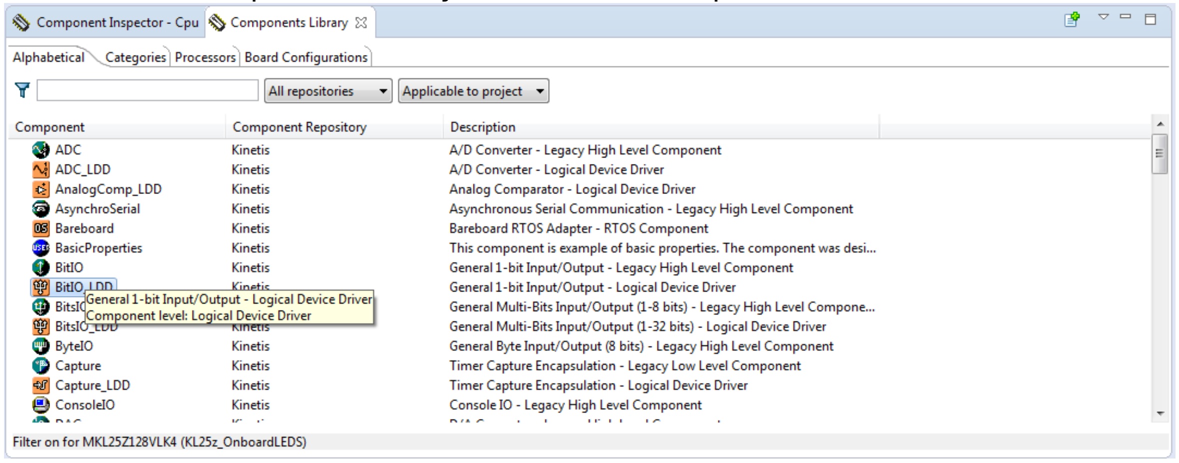
3. LED

In first part of this exercise, an On-board LED connected to one of the GPIO will be switched on and in the second part you will use any available GPIO to switch on an external LED.

e) Programming the On-board LED’s

To turn on or off or read the values of GPIO pins on the KL25Z we will use the BitIO_LDD component from the component library as shown below

1. Click on the component library and select the alphabetical tab as shown



Component Inspector - Cpu | Components Library

Alphabetical | Categories | Processors | Board Configurations

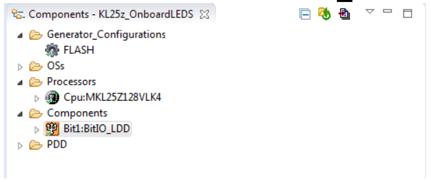
All repositories | Applicable to project

Component	Component Repository	Description
ADC	Kinetis	A/D Converter - Legacy High Level Component
ADC_LDD	Kinetis	A/D Converter - Logical Device Driver
AnalogComp_LDD	Kinetis	Analog Comparator - Logical Device Driver
AsynchroSerial	Kinetis	Asynchronous Serial Communication - Legacy High Level Component
Bareboard	Kinetis	Bareboard RTOS Adapter - RTOS Component
BasicProperties	Kinetis	This component is example of basic properties. The component was desi...
BitIO	Kinetis	General 1-bit Input/Output - Legacy High Level Component
BitIO_LDD	Kinetis	General 1-bit Input/Output - Logical Device Driver
BitsIO		General Multi-Bits Input/Output (1-8 bits) - Legacy High Level Compone...
BitsIO_LDD	Kinetis	General Multi-Bits Input/Output (1-32 bits) - Logical Device Driver
ByteIO	Kinetis	General Byte Input/Output (8 bits) - Legacy High Level Component
Capture	Kinetis	Timer Capture Encapsulation - Legacy Low Level Component
Capture_LDD	Kinetis	Timer Capture Encapsulation - Logical Device Driver
ConsoleIO	Kinetis	Console IO - Legacy High Level Component

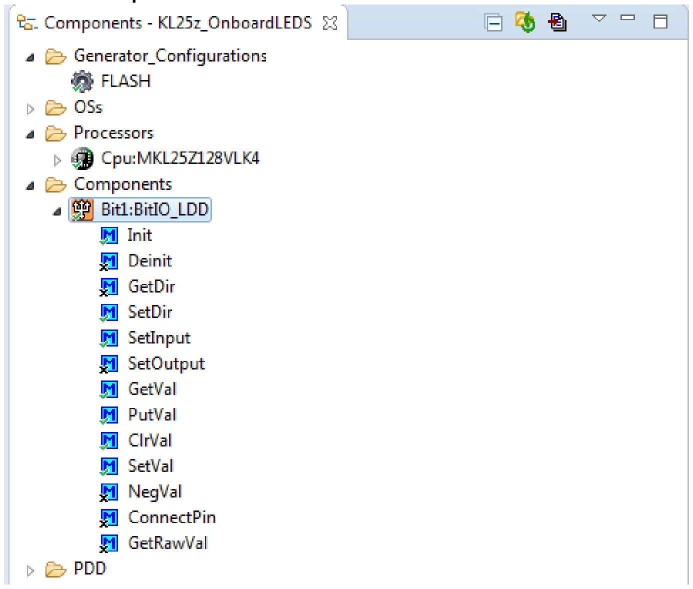
General 1-bit Input/Output - Logical Device Driver
Component level: Logical Device Driver

Filter on for MKL25Z128VLK4 (KL25z_OnboardLEDS)

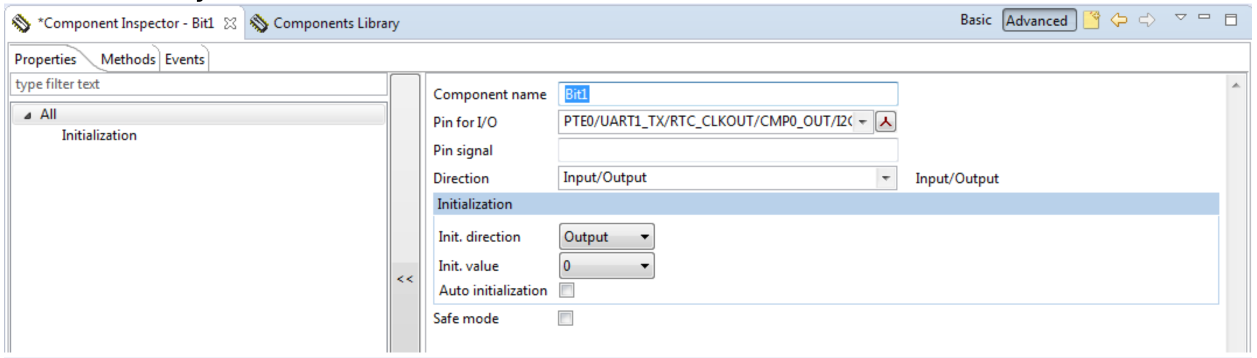
2. Double click the “BitIO_LDD” to add it to the project components



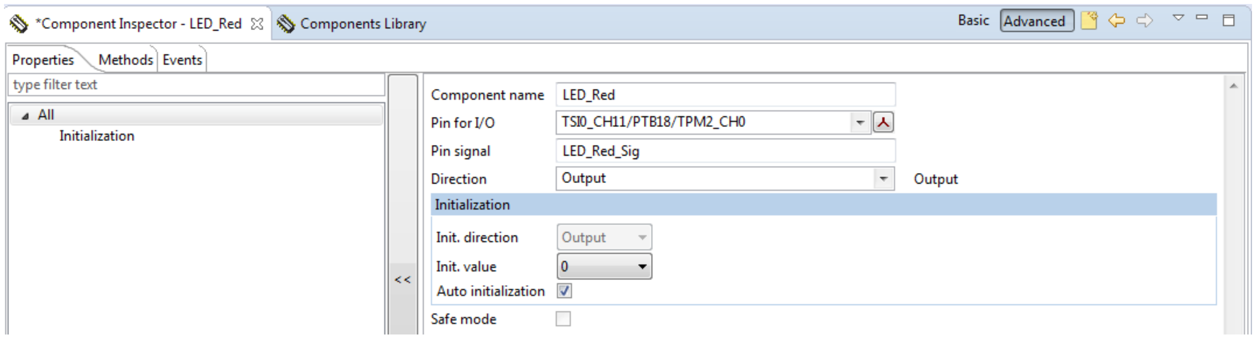
3. Double click the added “BitIO_LDD” in the components to open the component inspector for this component and click on the advanced mode to edit parameters



4. Do the following
   - Rename the ‘Component name’ to “LED_Red”
   - Select “TSI0_CH11/PTB18/TPM2_CH0” as ‘Pin for I/O’
   - Give an optional ‘Pin signal’ name as “LED_Red_Sig”
   - Select the ‘Direction’ as “Output”
   - And Finally enable ‘Auto Initialization’ to enable its initialization

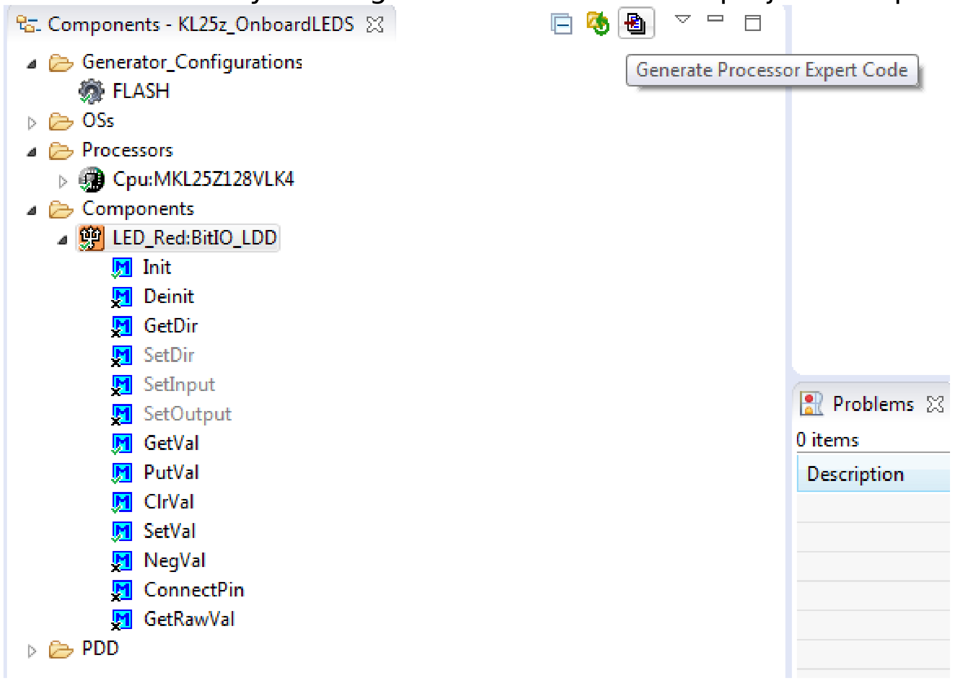


Component name	Bit1	
Pin for I/O	PTE0/UART1_TX/RTC_CLKOUT/CMP0_OUT/I2(	
Pin signal		
Direction	Input/Output	Input/Output
Initialization		
Init. direction	Output	
Init. value	0	
Auto initialization	☐	
Safe mode	☐	

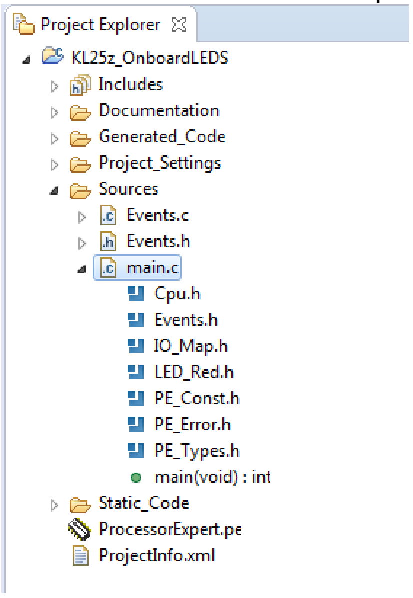


Component name	LED_Red	
Pin for I/O	TSI0_CH11/PTB18/TPM2_CH0	
Pin signal	LED_Red_Sig	
Direction	Output	Output
Initialization		
Init. direction	Output	
Init. value	0	
Auto initialization	✓	
Safe mode	☐	

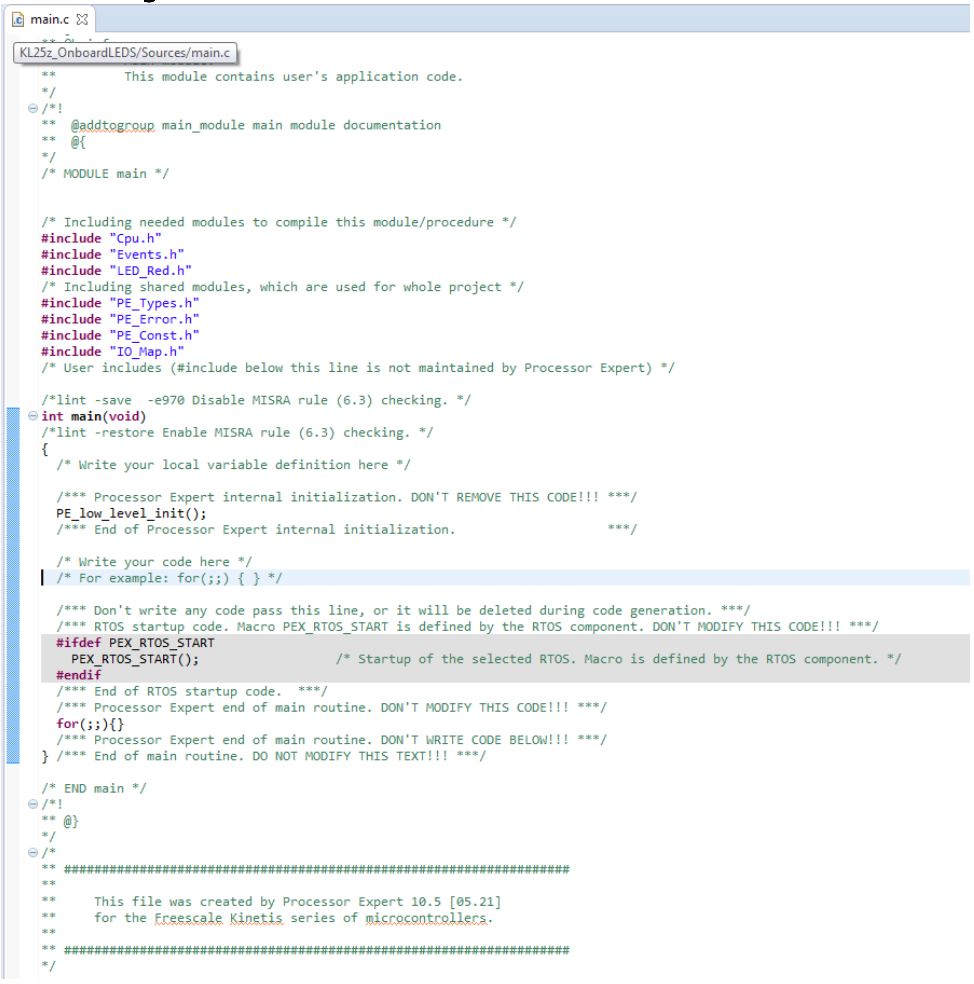
5. Generate code by clicking on the PE icon on the project components window



6. The application “main.c” in the ‘Sources’ folder under the main project in the Project Explorer window. Double click to open it

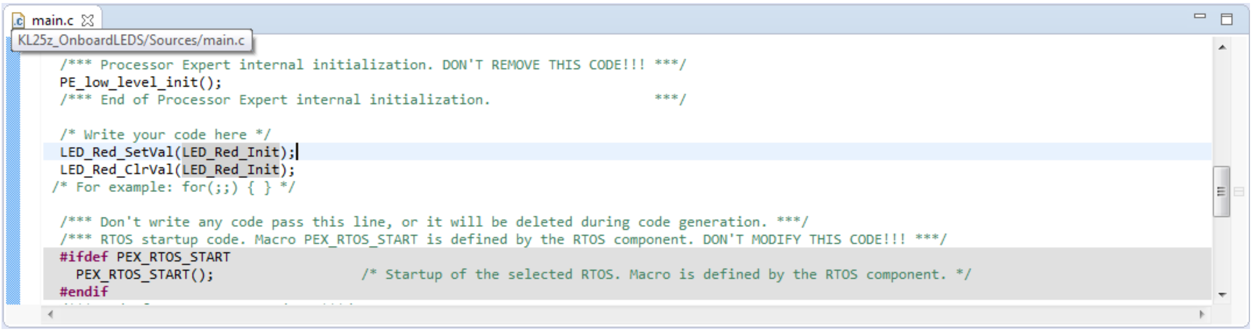


7. The auto generated code is as follows



```c
**         This module contains user's application code.
*/
/*!
**  @addtogroup main_module main module documentation
**  @{
*/
/* MODULE main */


/* Including needed modules to compile this module/procedure */
#include "Cpu.h"
#include "Events.h"
#include "LED_Red.h"
/* Including shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
/* User includes (#include below this line is not maintained by Processor Expert) */

/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  /* For example: for(;;) { } */

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of RTOS startup code.  ***/
  /*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
  for(;;){}
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
} /*** End of main routine. DO NOT MODIFY THIS TEXT!!! ***/

/* END main */
/*!
** @}
*/
/*
** ###################################################################
**
**     This file was created by Processor Expert 10.5 [05.21]
**     for the Freescale Kinetis series of microcontrollers.
**
** ###################################################################
*/
```

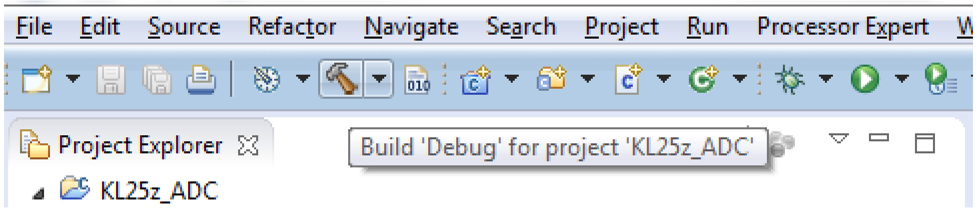
8. Add code by using drag and drop from components to the main c code the following methods ‘SetVal’ and ‘ClrVal’. Be sure to drag and drop the 'init' function into these functions and remove the '();' from them. By doing this we are treating the initialization function as a pointer to the required registers.



main.c
KL25z_OnboardLEDS/Sources/main.c

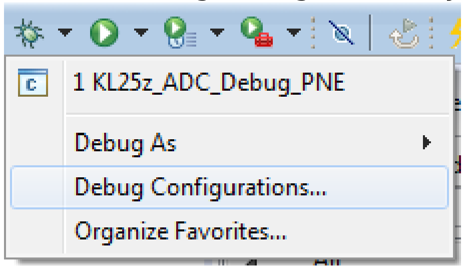
```
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  LED_Red_SetVal(LED_Red_Init);
  LED_Red_ClrVal(LED_Red_Init);
 /* For example: for(;;) { } */

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
```

9. Build the project after writing the code by clicking on the hammer icon

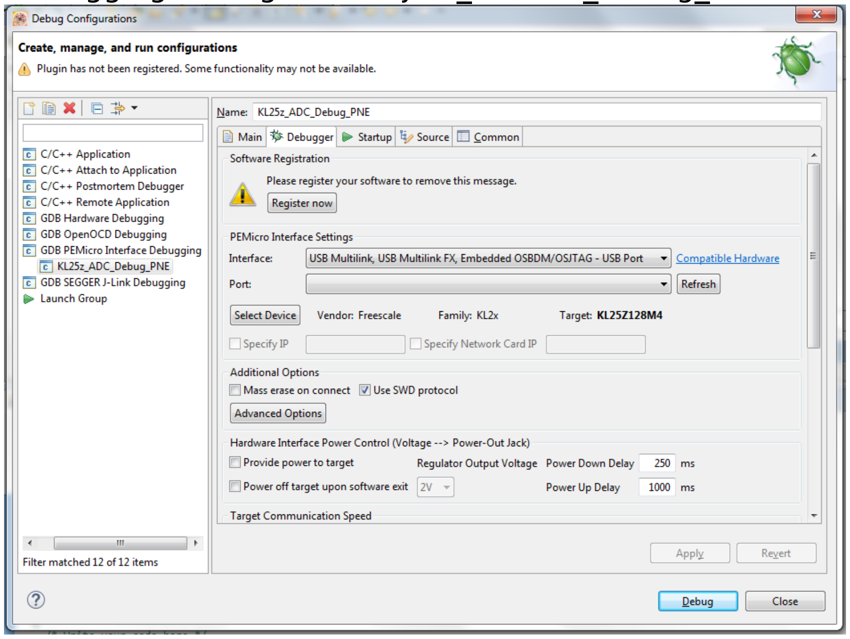


Build 'Debug' for project 'KL25z_ADC'

10. Set the debug configuration by clicking the arrow down beside the bug icon



11. Select the GDB PEMicro Interface Debugging on the left side menu and click on project debugging ending in ‘<Project_Name>_Debug_PNE” and select the Debugger tab



Debug Configurations

Create, manage, and run configurations

Plugin has not been registered. Some functionality may not be available.

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
- GDB PEMicro Interface Debugging
  - KL25z_ADC_Debug_PNE
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 12 of 12 items

Name: KL25z_ADC_Debug_PNE

Main | Debugger | Startup | Source | Common

Software Registration

Please register your software to remove this message.

Register now

PEMicro Interface Settings

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port   Compatible Hardware

Port: Refresh

Select Device   Vendor: Freescale   Family: KL2x   Target: **KL25Z128M4**

☐ Specify IP   ☐ Specify Network Card IP

Additional Options

☐ Mass erase on connect   ☑ Use SWD protocol

Advanced Options

Hardware Interface Power Control (Voltage --> Power-Out Jack)

☐ Provide power to target   Regulator Output Voltage   Power Down Delay 250 ms

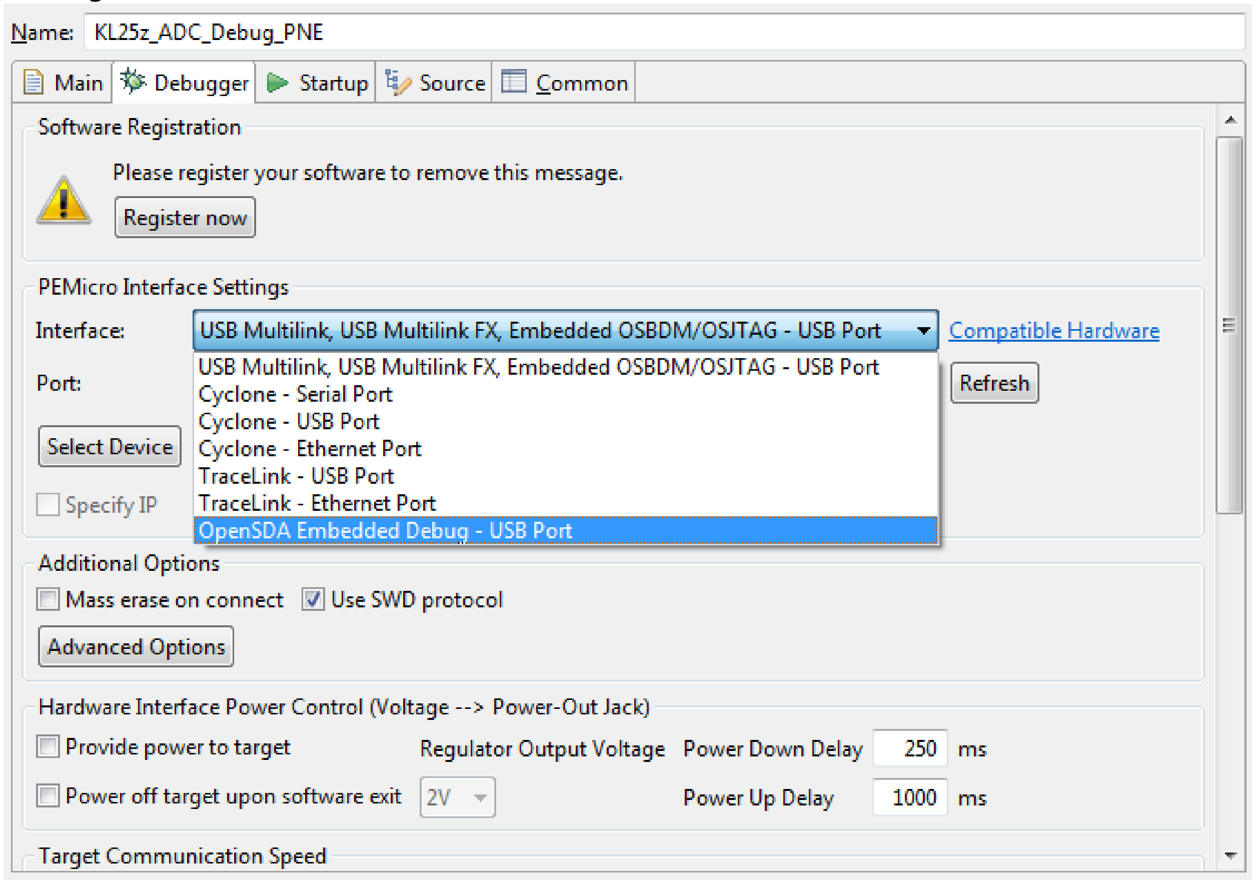
☐ Power off target upon software exit   2V   Power Up Delay 1000 ms

Target Communication Speed

Apply   Revert

Debug   Close

12.Select the interface as 'OpenSDA Embedded Debug – USB port', click apply and then click 'debug'



Name: KL25z_ADC_Debug_PNE

Software Registration

Please register your software to remove this message.

PEMicro Interface Settings

Interface: USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port

Port:

- USB Multilink, USB Multilink FX, Embedded OSBDM/OSJTAG - USB Port
- Cyclone - Serial Port
- Cyclone - USB Port
- Cyclone - Ethernet Port
- TraceLink - USB Port
- TraceLink - Ethernet Port
- OpenSDA Embedded Debug - USB Port

Additional Options

- [ ] Mass erase on connect
- [x] Use SWD protocol

Hardware Interface Power Control (Voltage --> Power-Out Jack)

- [ ] Provide power to target
- [ ] Power off target upon software exit

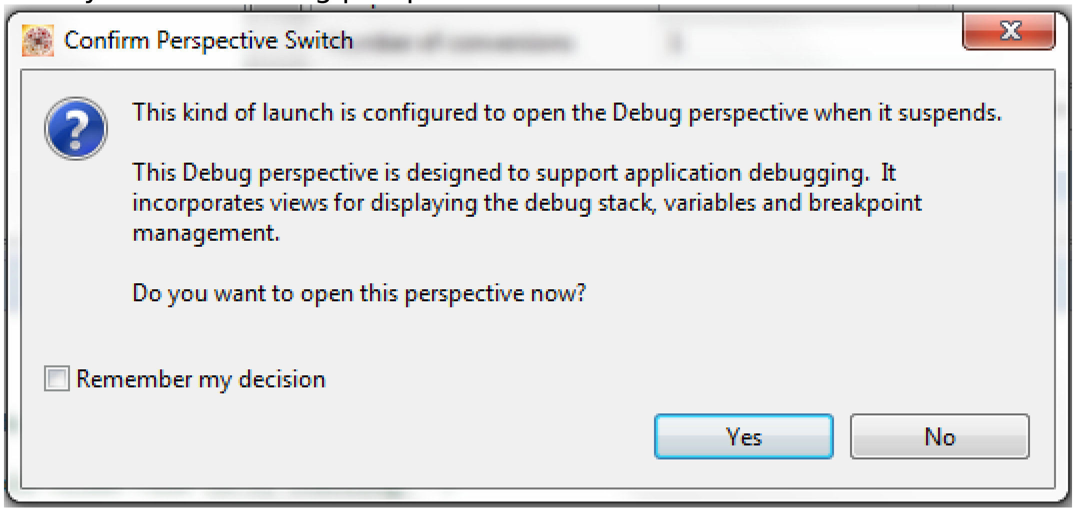
Regulator Output Voltage 2V

Power Down Delay 250 ms

Power Up Delay 1000 ms

Target Communication Speed

13.Click yes to the debug perspective view.



Confirm Perspective Switch

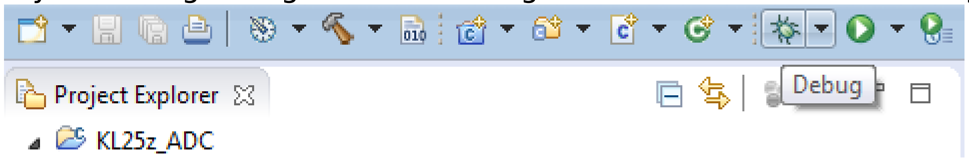
This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

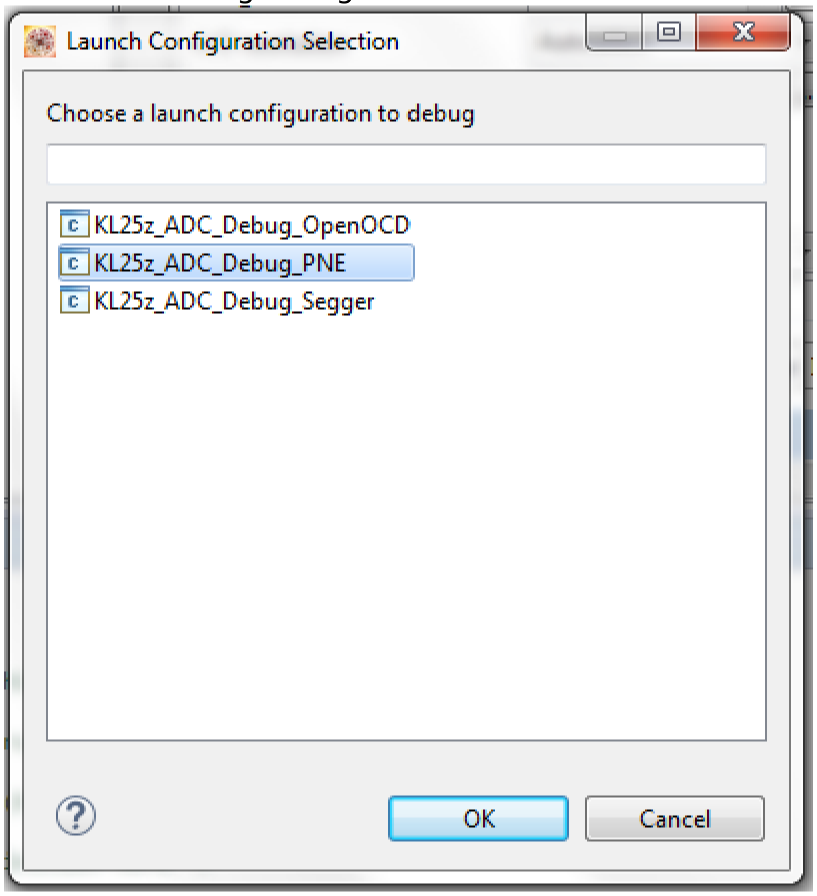
Do you want to open this perspective now?

- [ ] Remember my decision

14.If your debug configuration is debug is set then click on the debug icon



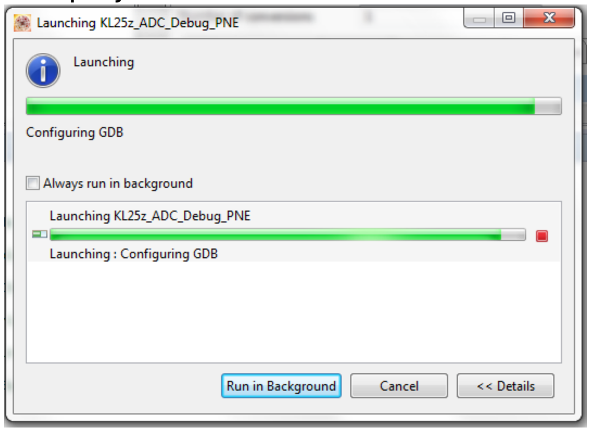
15. Select the debug configuration


Launch Configuration Selection

Choose a launch configuration to debug

KL25z_ADC_Debug_OpenOCD
KL25z_ADC_Debug_PNE
KL25z_ADC_Debug_Segger

16. The project will be launched in the debug mode


Launching KL25z_ADC_Debug_PNE

Launching

Configuring GDB

Always run in background

Launching KL25z_ADC_Debug_PNE

Launching : Configuring GDB

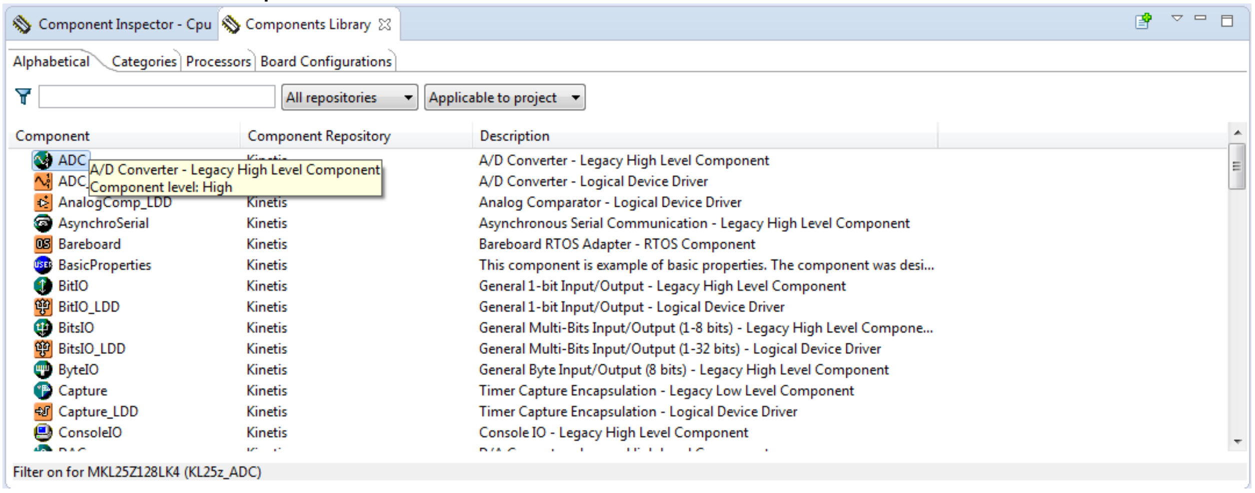
f) Programming an external LED

Repeat the above steps but with another GPIO, say PTB0, to switch on a Red generic LED with 330 ohm resistor. The breadboard will be used to connect from port pin 1 of J4 I/O header (I/O header pin 4) to LED in series with one resistor and terminating in ground.

4. ADC

Create a new project as mentioned earlier. We shall use the PTB0 to read analog voltage, this is connected to J10, pin 2 according to the Pinouts.

1. Add the ADC component

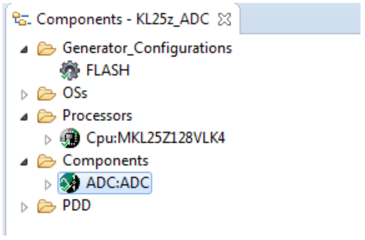


Component	Component Repository	Description
ADC	Kinetis	A/D Converter - Legacy High Level Component
ADC_LDD	Kinetis	A/D Converter - Logical Device Driver
AnalogComp_LDD	Kinetis	Analog Comparator - Logical Device Driver
AsynchroSerial	Kinetis	Asynchronous Serial Communication - Legacy High Level Component
Bareboard	Kinetis	Bareboard RTOS Adapter - RTOS Component
BasicProperties	Kinetis	This component is example of basic properties. The component was desi...
BitIO	Kinetis	General 1-bit Input/Output - Legacy High Level Component
BitIO_LDD	Kinetis	General 1-bit Input/Output - Logical Device Driver
BitsIO	Kinetis	General Multi-Bits Input/Output (1-8 bits) - Legacy High Level Compone...
BitsIO_LDD	Kinetis	General Multi-Bits Input/Output (1-32 bits) - Logical Device Driver
ByteIO	Kinetis	General Byte Input/Output (8 bits) - Legacy High Level Component
Capture	Kinetis	Timer Capture Encapsulation - Legacy Low Level Component
Capture_LDD	Kinetis	Timer Capture Encapsulation - Logical Device Driver
ConsoleIO	Kinetis	Console IO - Legacy High Level Component

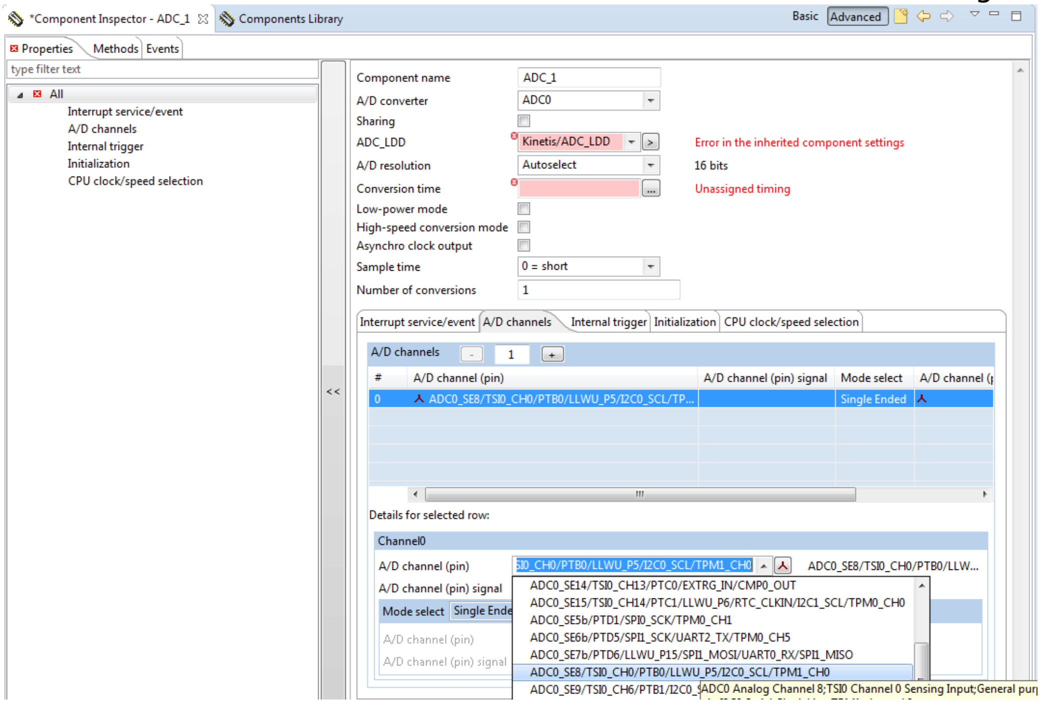
A/D Converter - Legacy High Level Component
Component level: High

Filter on for MKL25Z128LK4 (KL25z_ADC)

2. Double click the ADC component to open the component inspector



3. Scroll down and select the 'A/D Channels' Tab and set the following



Component name	ADC_1	
A/D converter	ADC0	
Sharing	☐	
ADC_LDD	Kinetis/ADC_LDD	Error in the inherited component settings
A/D resolution	Autoselect	16 bits
Conversion time		Unassigned timing
Low-power mode	☐	
High-speed conversion mode	☐	
Asynchro clock output	☐	
Sample time	0 = short	
Number of conversions	1	

A/D channels 1

#	A/D channel (pin)	A/D channel (pin) signal	Mode select	A/D channel (p
0	ADC0_SE8/TSI0_CH0/PTB0/LLWU_P5/I2C0_SCL/TP...		Single Ended	

Details for selected row:

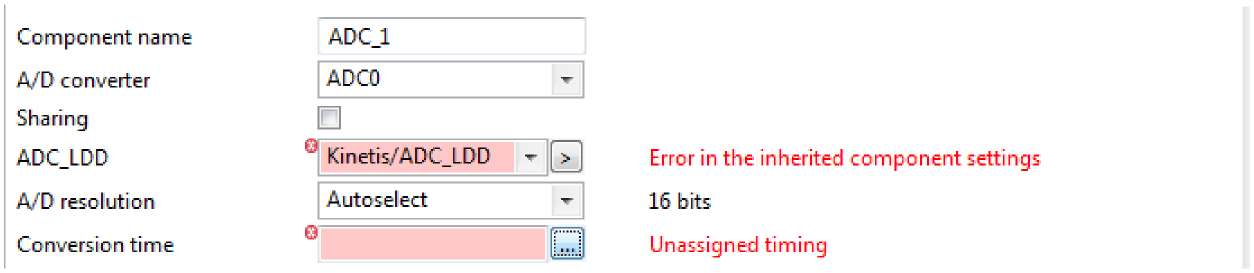
Channel0

A/D channel (pin): SI0_CH0/PTB0/LLWU_P5/I2C0_SCL/TPM1_CH0 ADC0_SE8/TSI0_CH0/PTB0/LLW...
A/D channel (pin) signal
Mode select Single Ende
A/D channel (pin)
A/D channel (pin) signal

ADC0_SE14/TSI0_CH13/PTC0/EXTRG_IN/CMP0_OUT
ADC0_SE15/TSI0_CH14/PTC1/LLWU_P6/RTC_CLKIN/I2C1_SCL/TPM0_CH0
ADC0_SE5b/PTD1/SPI0_SCK/TPM0_CH1
ADC0_SE6b/PTD5/SPI1_SCK/UART2_TX/TPM0_CH5
ADC0_SE7b/PTD6/LLWU_P15/SPI1_MOSI/UART0_RX/SPI1_MISO
ADC0_SE8/TSI0_CH0/PTB0/LLWU_P5/I2C0_SCL/TPM1_CH0
ADC0_SE9/TSI0_CH6/PTB1/I2C0_S

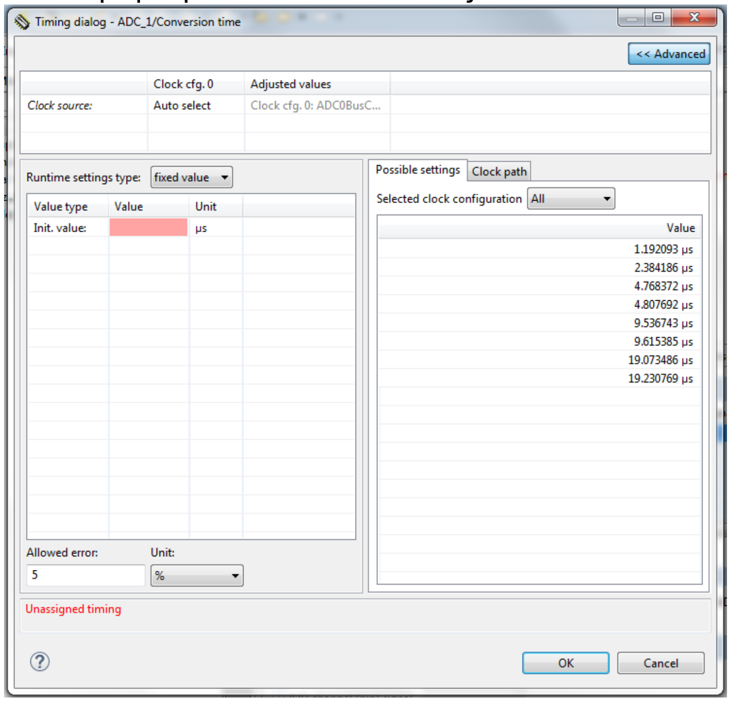
ADC0 Analog Channel 8;TSI0 Channel 0 Sensing Input;General pur

4. Click on 'conversion time' '…' more button



Component name	ADC_1	
A/D converter	ADC0	
Sharing	☐	
ADC_LDD	Kinetis/ADC_LDD	Error in the inherited component settings
A/D resolution	Autoselect	16 bits
Conversion time		Unassigned timing

5. In the pop-up window select any conversion time and click 'ok'



Timing dialog - ADC_1/Conversion time

	Clock cfg. 0	Adjusted values
Clock source:	Auto select	Clock cfg. 0: ADC0BusC…

Runtime settings type: fixed value

Value type	Value	Unit
Init. value:		μs

Allowed error: 5 Unit: %

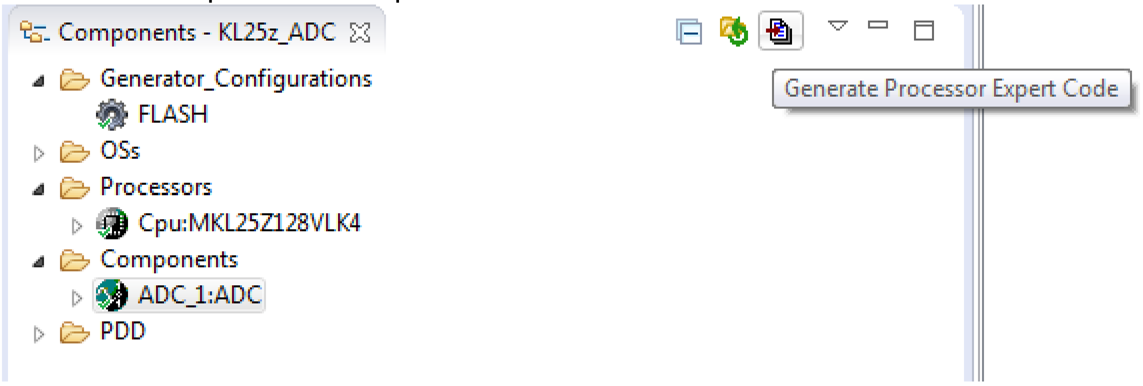
Possible settings | Clock path

Selected clock configuration: All

Value
1.192093 μs
2.384186 μs
4.768372 μs
4.807692 μs
9.536743 μs
9.615385 μs
19.073486 μs
19.230769 μs

Unassigned timing

6. Generate the processor expert code



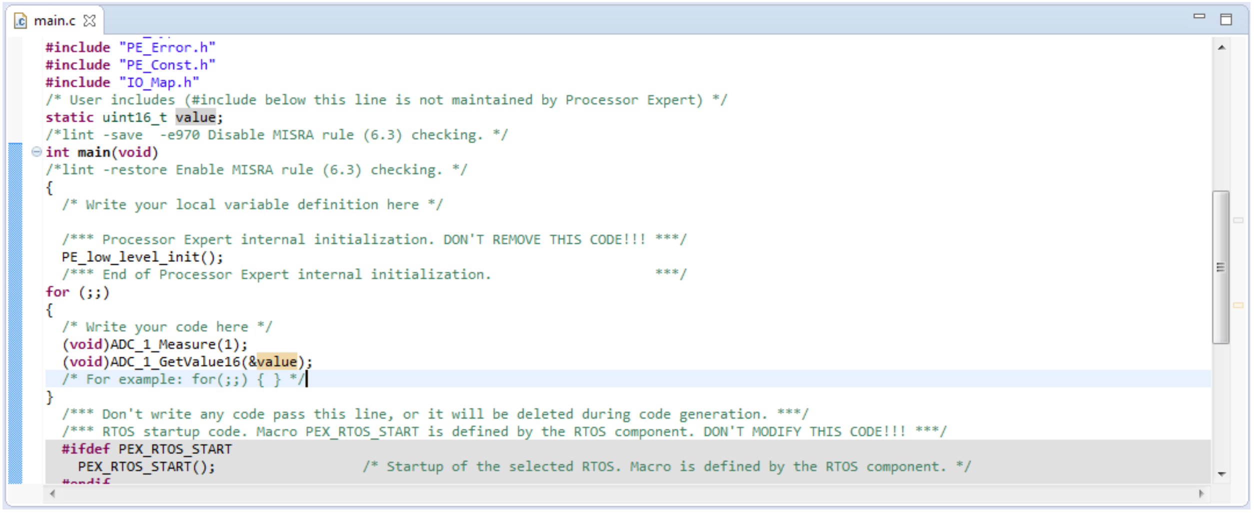
Components - KL25z_ADC

- Generator_Configurations
  - FLASH
- OSs
- Processors
  - Cpu:MKL25Z128VLK4
- Components
  - ADC_1:ADC
- PDD

7. Drag and drop the 'Measure' method for the 'ADC' component in the main.c code.

8. The method measure() needs an extra boolean argument if it shall wait for the finish of the conversion (we want to wait). Place a number 1 inside the parenthesis to indicate true. And we are not interested in the error return code, so we cast it to void

9. To get the conversion value, the method GetValue16() is used which returns the 16bit result. For the result define a 16bit global variable and do the AD conversion in an endless loop. The final code would look like this



main.c

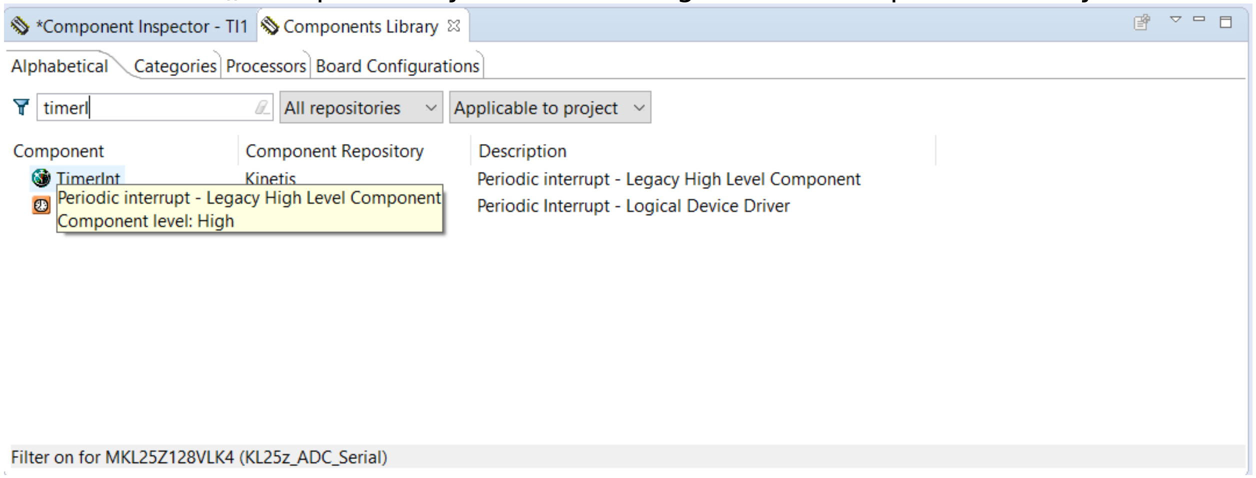
```
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"
/* User includes (#include below this line is not maintained by Processor Expert) */
static uint16_t value;
/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/
for (;;)
{
  /* Write your code here */
  (void)ADC_1_Measure(1);
  (void)ADC_1_GetValue16(&value);
  /* For example: for(;;) { } */
}
  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX_RTOS_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
  #ifdef PEX_RTOS_START
    PEX_RTOS_START();                  /* Startup of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
```

10. Save the file and Build the file by clicking on the Hammer icon

5. Serial Debugging

Now we shall add the serial interface to send the ADC value read to the serial terminal. We shall use a TimerInt() component and the Serial_LDD() component

1. Add an ADC component and configure it as in the previous section.
2. Add a TimerInt() component by double clicking it in the component library

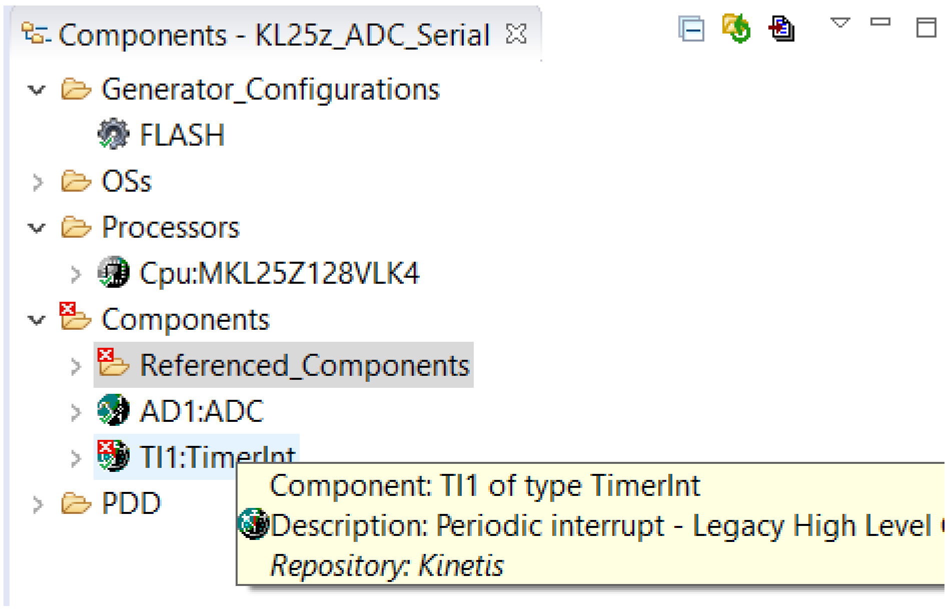


Component	Component Repository	Description
TimerInt	Kinetis	Periodic interrupt - Legacy High Level Component
		Periodic Interrupt - Logical Device Driver

Periodic interrupt - Legacy High Level Component
Component level: High

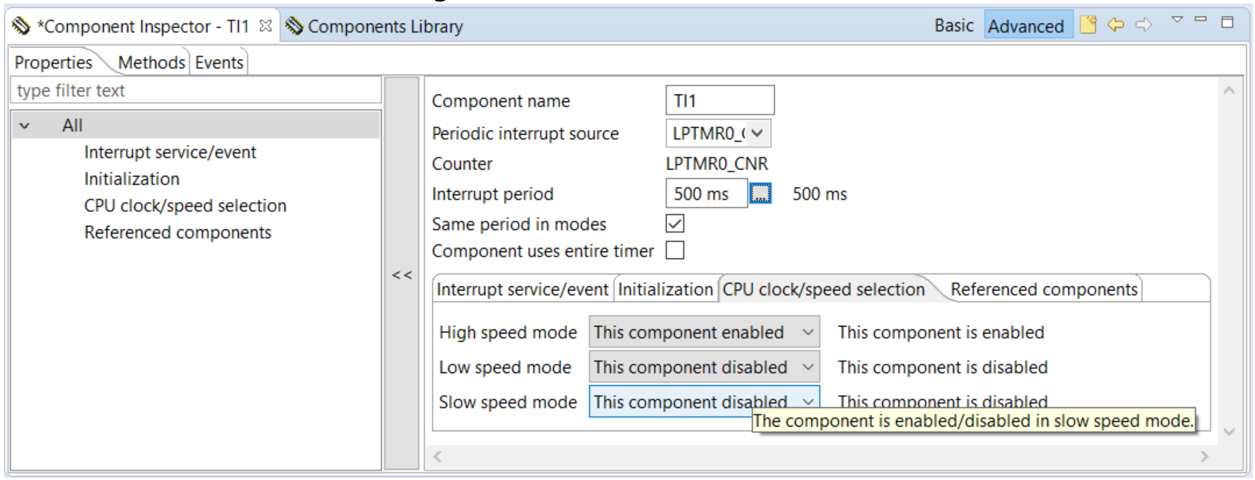
Filter on for MKL25Z128VLK4 (KL25z_ADC_Serial)

3. In the project component window, double click the TI1:TimerInt to open the component inspector



Component: TI1 of type TimerInt
Description: Periodic interrupt - Legacy High Level
Repository: Kinetis

4. In the component inspector set the interrupt period to 500ms by clicking on the '…' more button next to it and selecting 500ms

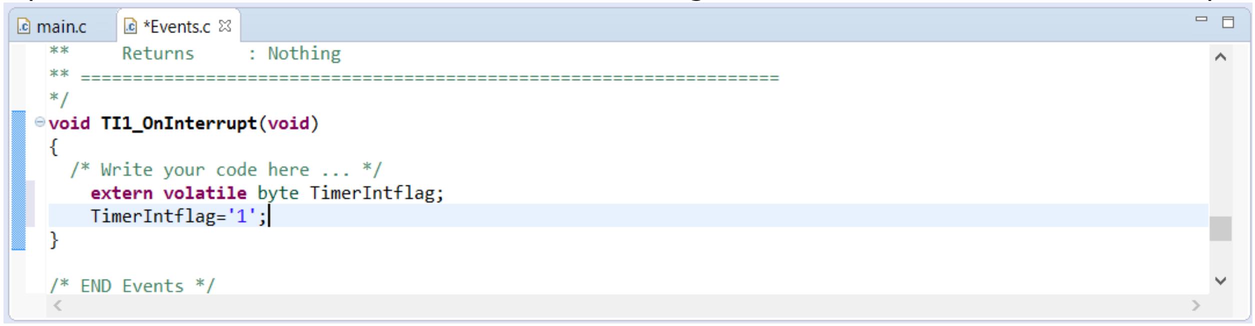


Component name	TI1	
Periodic interrupt source	LPTMR0_(	
Counter	LPTMR0_CNR	
Interrupt period	500 ms	500 ms
Same period in modes	✓	
Component uses entire timer	☐	

High speed mode	This component enabled	This component is enabled
Low speed mode	This component disabled	This component is disabled
Slow speed mode	This component disabled	This component is disabled

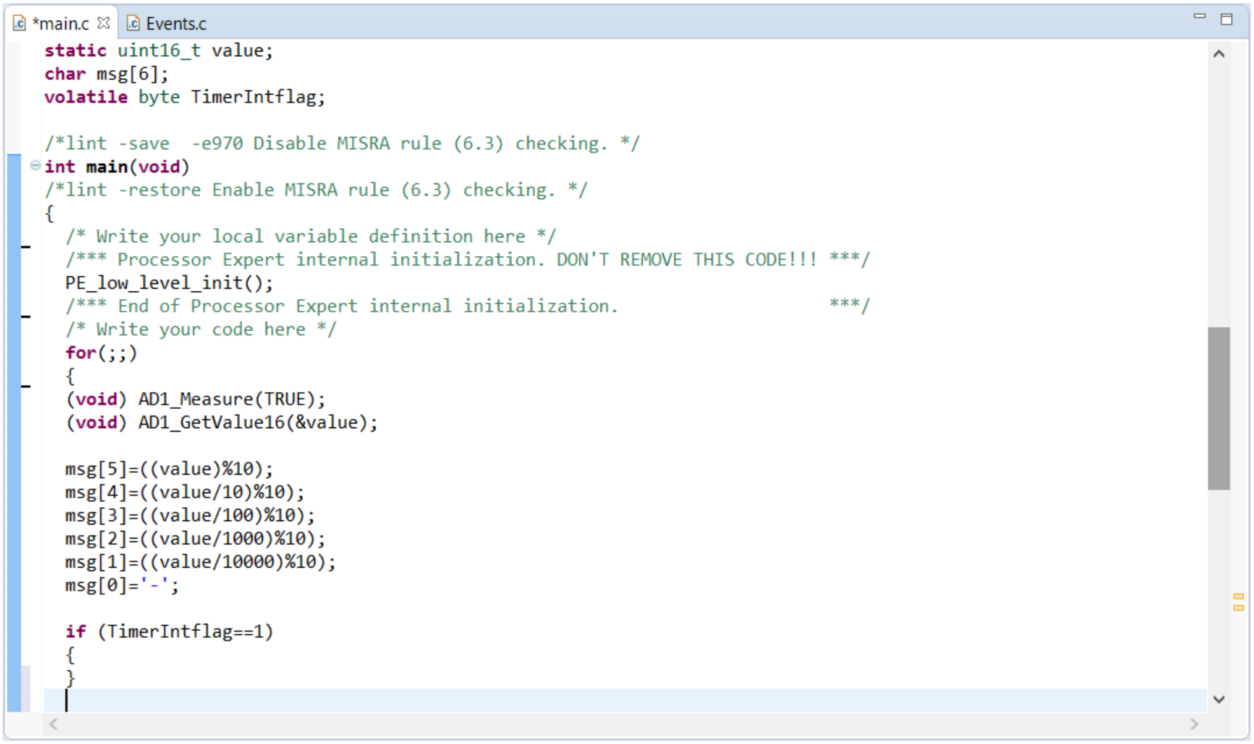
The component is enabled/disabled in slow speed mode.

5. Generate the Processor Expert Code by clicking the PE icon in the components view
6. Open the file "Events.c" and write the following code to execute when the interrupt occus



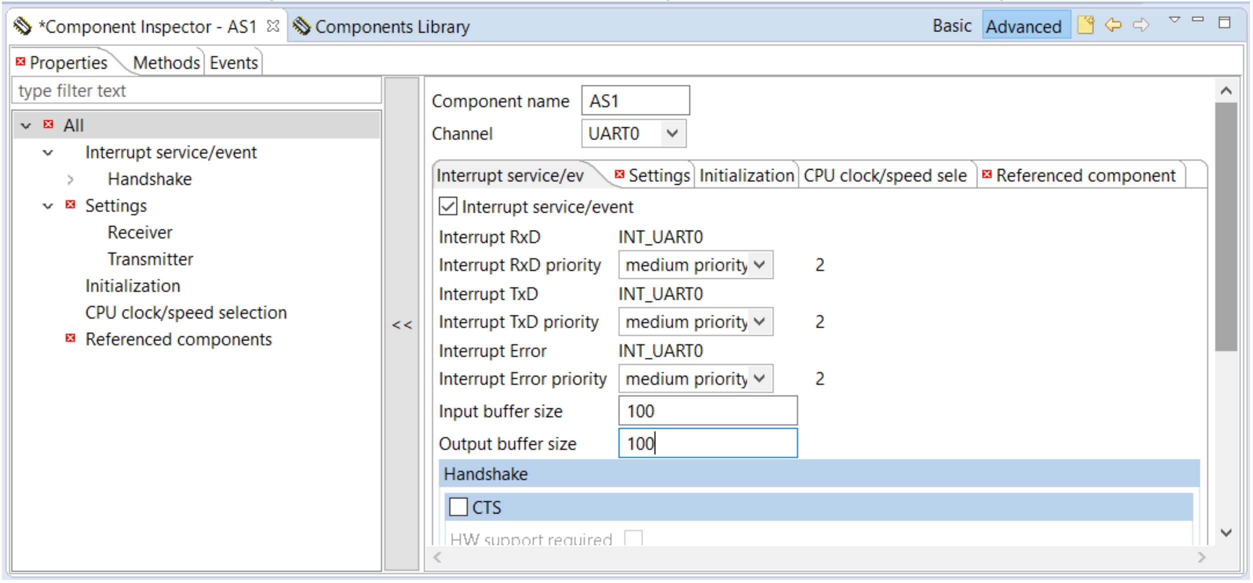
```
**     Returns      : Nothing
** ===================================================================
*/
void TI1_OnInterrupt(void)
{
  /* Write your code here ... */
    extern volatile byte TimerIntflag;
    TimerIntflag='1';
}

/* END Events */
```

7. Add the following code to the main.c



```
static uint16_t value;
char msg[6];
volatile byte TimerIntflag;

/*lint -save  -e970 Disable MISRA rule (6.3) checking. */
int main(void)
/*lint -restore Enable MISRA rule (6.3) checking. */
{
  /* Write your local variable definition here */
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/
  /* Write your code here */
  for(;;)
  {
  (void) AD1_Measure(TRUE);
  (void) AD1_GetValue16(&value);

  msg[5]=((value)%10);
  msg[4]=((value/10)%10);
  msg[3]=((value/100)%10);
  msg[2]=((value/1000)%10);
  msg[1]=((value/10000)%10);
  msg[0]='-';

  if (TimerIntflag==1)
  {
  }
  |
```

8. Now we need the ASynchroSerial() to send the value read to the serial terminal. Add from the library the AsynchroSerial() to the projects components and double click it to open the component inspector
9. Enable the interrupt service/event, set the input buffer size and output buffer size to 100 each.



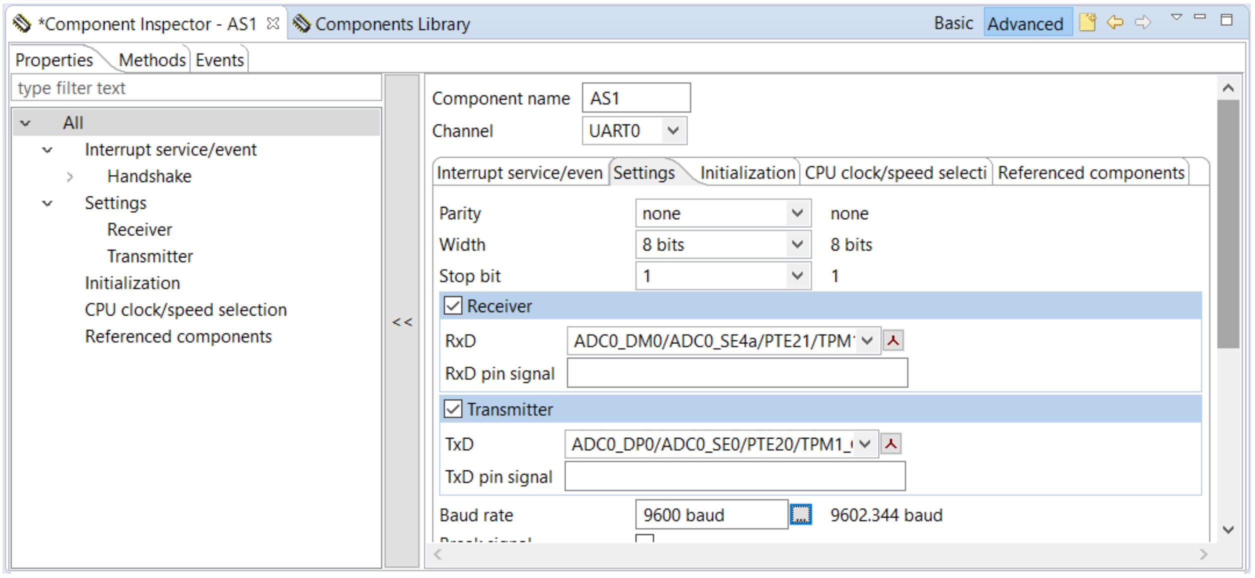
Component name AS1
Channel UART0

☑ Interrupt service/event

Interrupt RxD	INT_UART0	
Interrupt RxD priority	medium priority	2
Interrupt TxD	INT_UART0	
Interrupt TxD priority	medium priority	2
Interrupt Error	INT_UART0	
Interrupt Error priority	medium priority	2
Input buffer size	100	
Output buffer size	100	

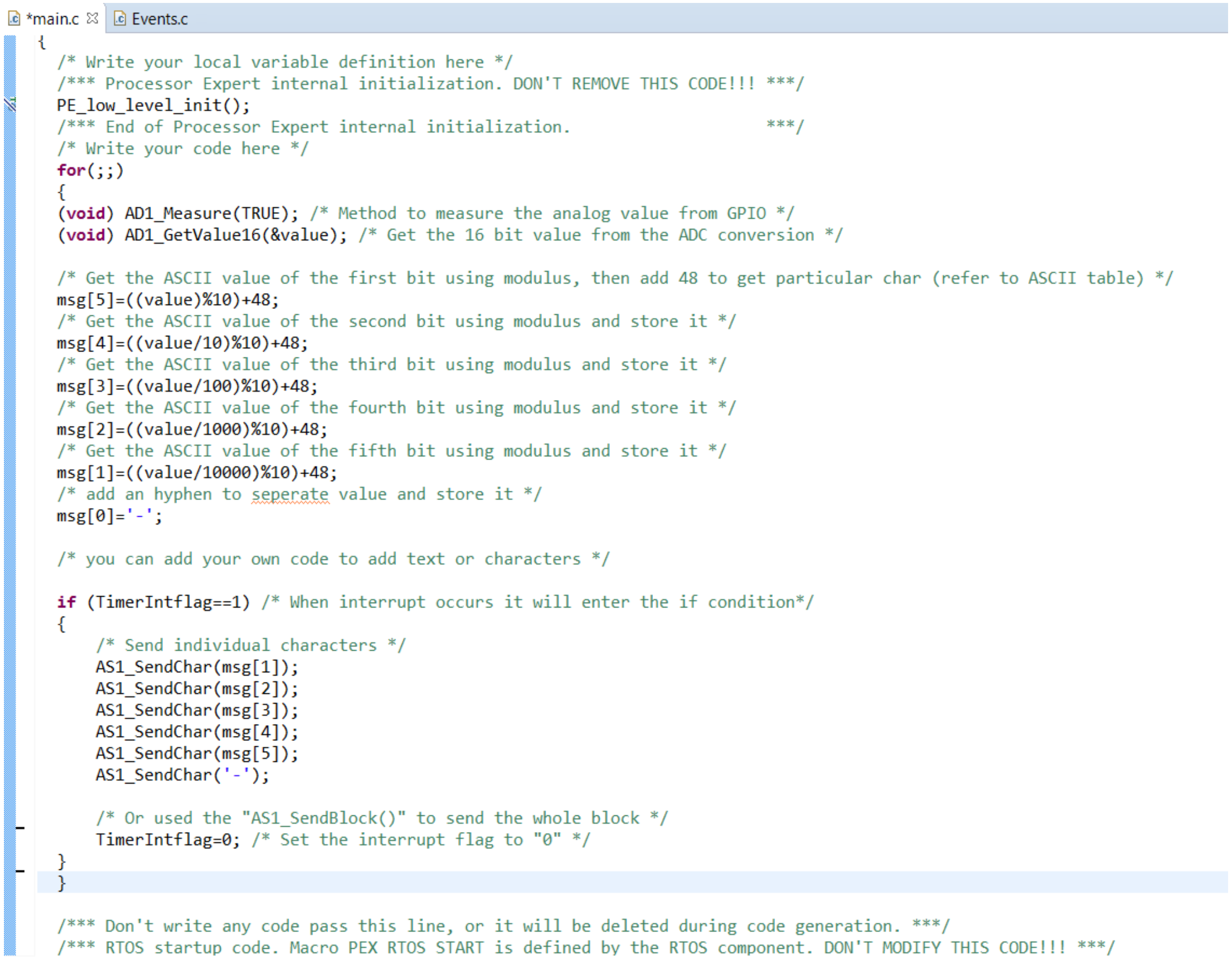
☐ CTS
HW support required ☐

10. In the setting tab, set the baud rate to be 9600, select the Receiver and transmitter from Port A i.e. PTA1 as RxD and PTA2 as TxD from the dropdown menu respectively.



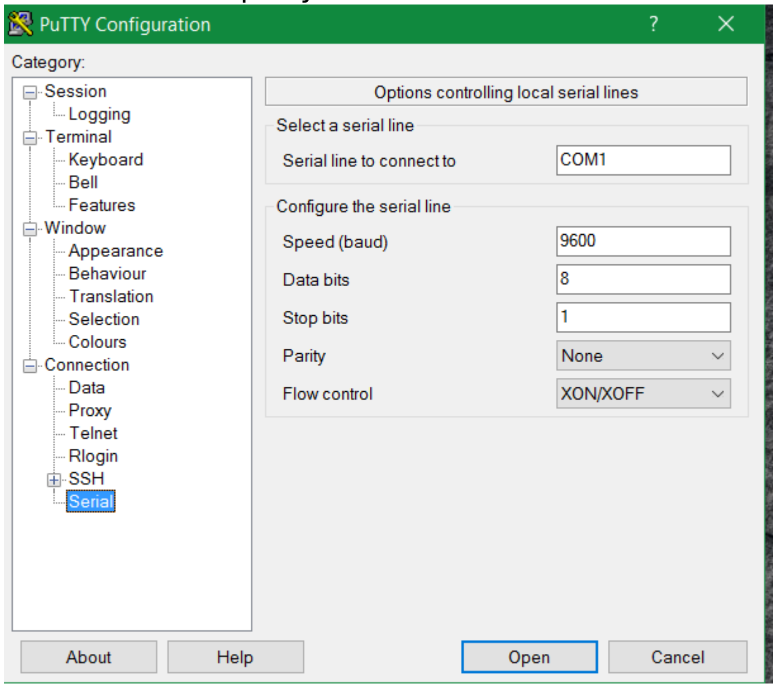
Component name	AS1	
Channel	UART0	
Parity	none	none
Width	8 bits	8 bits
Stop bit	1	1
☑ Receiver		
RxD	ADC0_DM0/ADC0_SE4a/PTE21/TPM	
RxD pin signal		
☑ Transmitter		
TxD	ADC0_DP0/ADC0_SE0/PTE20/TPM1_	
TxD pin signal		
Baud rate	9600 baud	9602.344 baud

11. Generate the Processor Expert Code by clicking on the icon.
12. Drag and drop the AS1_SendChar() method into the main.c file and write the following code. You can add



```
{
  /* Write your local variable definition here */
  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  PE_low_level_init();
  /*** End of Processor Expert internal initialization.                    ***/
  /* Write your code here */
  for(;;)
  {
  (void) AD1_Measure(TRUE); /* Method to measure the analog value from GPIO */
  (void) AD1_GetValue16(&value); /* Get the 16 bit value from the ADC conversion */

  /* Get the ASCII value of the first bit using modulus, then add 48 to get particular char (refer to ASCII table) */
  msg[5]=((value)%10)+48;
  /* Get the ASCII value of the second bit using modulus and store it */
  msg[4]=((value/10)%10)+48;
  /* Get the ASCII value of the third bit using modulus and store it */
  msg[3]=((value/100)%10)+48;
  /* Get the ASCII value of the fourth bit using modulus and store it */
  msg[2]=((value/1000)%10)+48;
  /* Get the ASCII value of the fifth bit using modulus and store it */
  msg[1]=((value/10000)%10)+48;
  /* add an hyphen to seperate value and store it */
  msg[0]='-';

  /* you can add your own code to add text or characters */

  if (TimerIntflag==1) /* When interrupt occurs it will enter the if condition*/
  {
      /* Send individual characters */
      AS1_SendChar(msg[1]);
      AS1_SendChar(msg[2]);
      AS1_SendChar(msg[3]);
      AS1_SendChar(msg[4]);
      AS1_SendChar(msg[5]);
      AS1_SendChar('-');

      /* Or used the "AS1_SendBlock()" to send the whole block */
      TimerIntflag=0; /* Set the interrupt flag to "0" */
  }
  }

  /*** Don't write any code pass this line, or it will be deleted during code generation. ***/
  /*** RTOS startup code. Macro PEX RTOS START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! ***/
```

13.Use any serial terminal like “putty.exe” or “TeraTerm.exe” to see the output



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM1

Configure the serial line

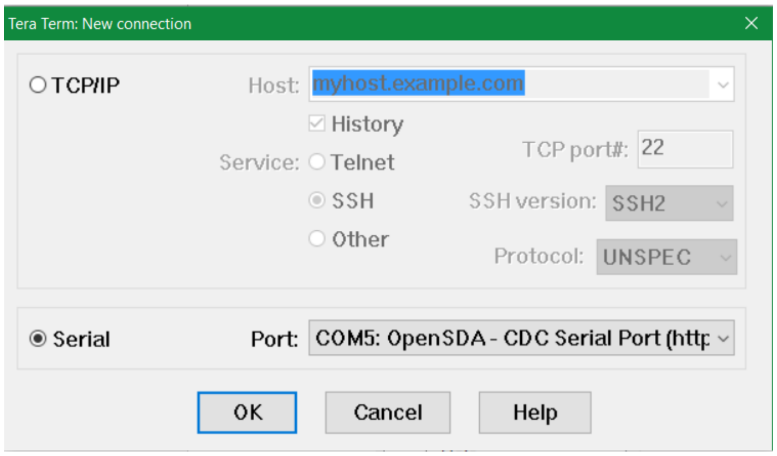
Speed (baud): 9600

Data bits: 8

Stop bits: 1

Parity: None

Flow control: XON/XOFF



TCP/IP Host: myhost.example.com

History

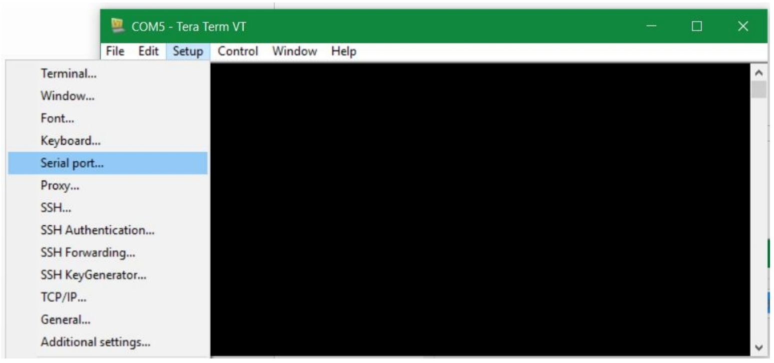
Service: Telnet / SSH / Other

TCP port#: 22

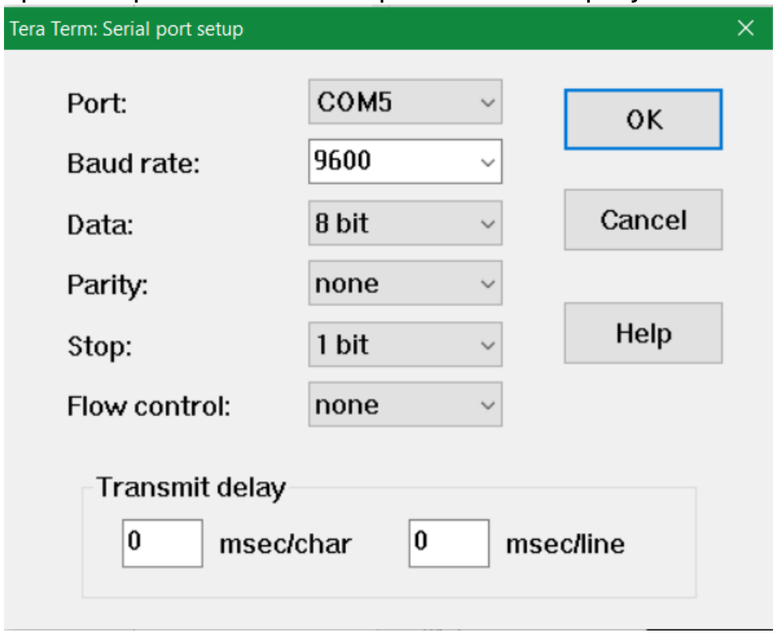
SSH version: SSH2

Protocol: UNSPEC

[✓ Serial] Port: COM5: OpenSDA - CDC Serial Port (http



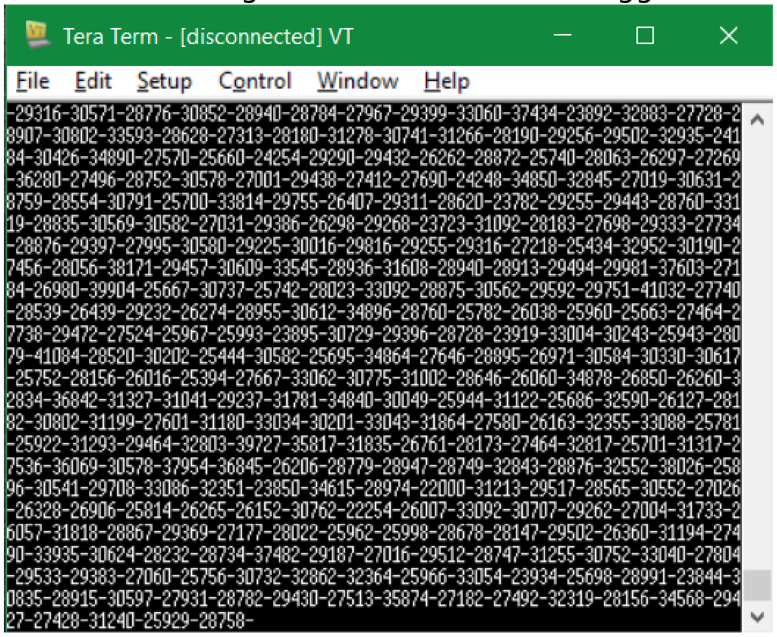
14.Set the serial port as per the AS1 component in the project



Port: COM5
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char
0 msec/line

15.Read the values at PTB0 on the terminal window. Click on the project folder and then click the debug icon to launch the debugger. a


Tera Term - [disconnected] VT
File Edit Setup Control Window Help

```
-29316-30571-28776-30852-28940-28784-27967-29399-33060-37434-23892-32883-27728-2
8907-30802-33593-28628-27313-28180-31278-30741-31266-28190-29256-29502-32935-241
84-30426-34890-27570-25660-24254-29290-29432-26262-28872-25740-28063-26297-27269
-36280-27496-28752-30578-27001-29438-27412-27690-24248-34850-32845-27019-30631-2
8759-28554-30791-25700-33814-29755-26407-29311-28620-23782-29255-29443-28760-331
19-28835-30569-30582-27031-29386-26298-29268-23723-31092-28183-27698-29333-27734
-28876-29397-27995-30580-29225-30016-29816-29255-29316-27218-25434-32952-30190-2
7456-28056-38171-29457-30609-33545-28936-31608-28940-28913-29494-29981-37603-271
84-26980-39904-25667-30737-25742-28023-33092-28875-30562-29592-29751-41032-27740
-28539-26439-29232-26274-28955-30612-34896-28760-25782-26038-25960-25663-27464-2
7738-29472-27524-25967-25993-23895-30729-29396-28728-23919-33004-30243-25943-280
79-41084-28520-30202-25444-30582-25695-34864-27646-28895-26971-30584-30330-30617
-25752-28156-26016-25394-27667-33062-30775-31002-28646-26060-34878-26850-26260-3
2834-36842-31327-31041-29237-31781-34840-30049-25944-31122-25686-32590-26127-281
82-30802-31199-27601-31180-33034-30201-33043-31864-27580-26163-32355-33088-25781
-25922-31293-29464-32803-39727-35817-31835-26761-28173-27464-32817-25701-31317-2
7536-36069-30578-37954-36845-26206-28779-28947-28749-32843-28876-32552-38026-258
96-30541-29708-33086-32351-23850-34615-28974-22000-31213-29517-28565-30552-27026
-26328-26906-25814-26265-26152-30762-22254-26007-33092-30707-29262-27004-31733-2
6057-31818-28867-29369-27177-28022-25962-25998-28678-28147-29502-26360-31194-274
90-33935-30624-28232-28734-37482-29187-27016-29512-28747-31255-30752-33040-27804
-29533-29383-27060-25756-30732-32862-32364-25966-33054-23934-25698-28991-23844-3
0835-28915-30597-27931-28782-29430-27513-35874-27182-27492-32319-28156-34568-294
27-27428-31240-25929-28758-
```

6. Checkout

The group has to successfully demonstrate programming all four functionality required by this lab.