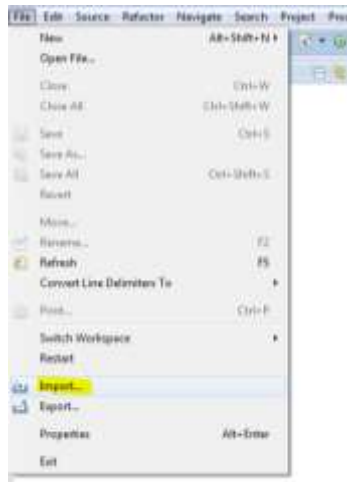


AMF-DES-T1679

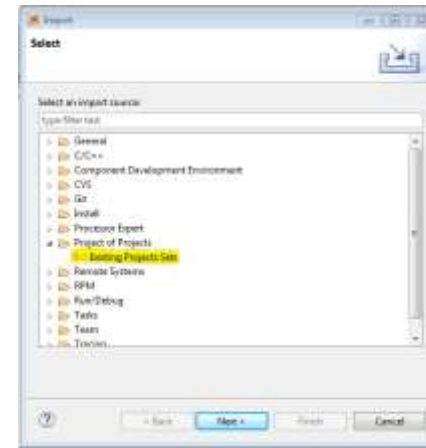
Hands-On Workshop: Getting Started with Kinetis SDK - Basic

Lab 1: Importing KSDK demos

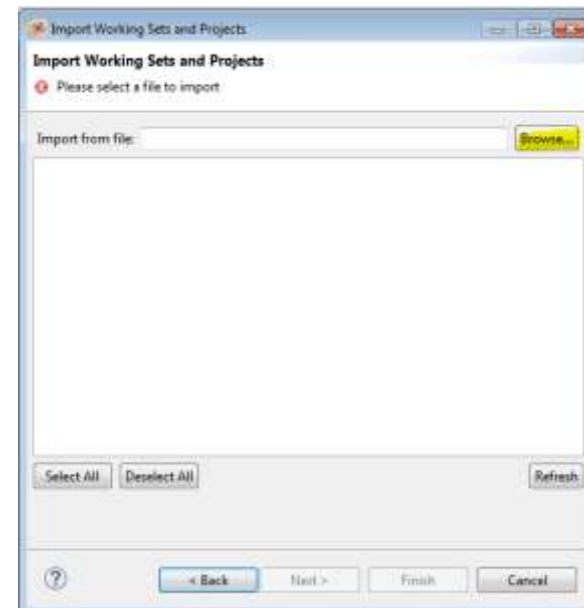
1. Click on 'File', then click on 'Import...'



2. Expand the 'Project of Projects' folder, and select 'Existing Projects Sets'. Click 'Next'.



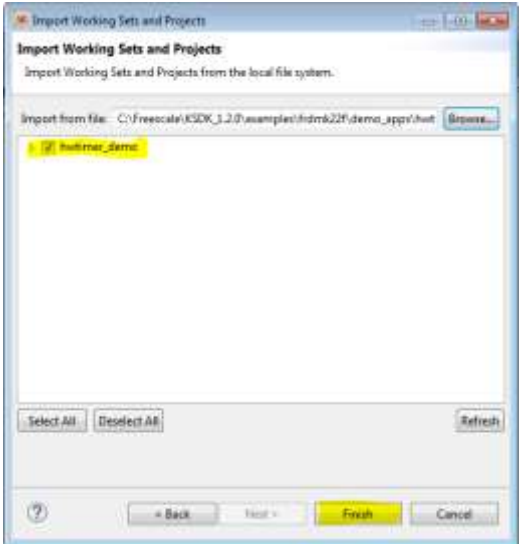
3. Click 'Browse' to navigate to the KSDK_1.2.0 installation.



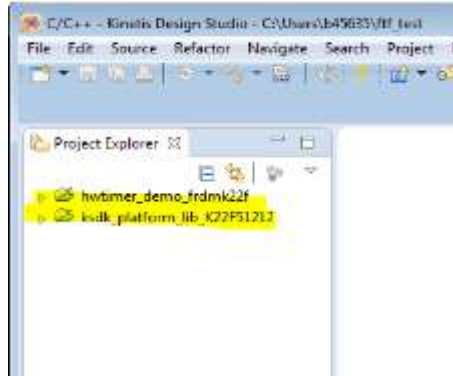
4. Select 'hwtimer_demo.wsd' under 'KSDK_1.2.0/examples/frdmk22f/demo_apps/hwtimer_demo/kds'.



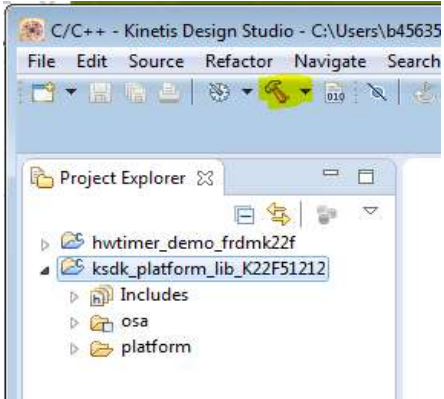
5. Make sure the working set is selected. Then click 'Finish'.

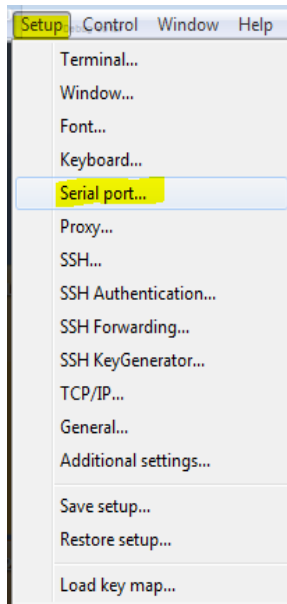


6. The platform library for K22F51212 and the hwtimer_demo will now appear in the 'Project Explorer' view.

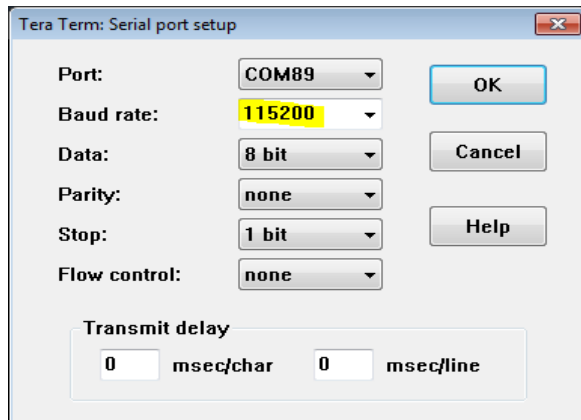


7. Build the library by clicking the 'Hammer' button.

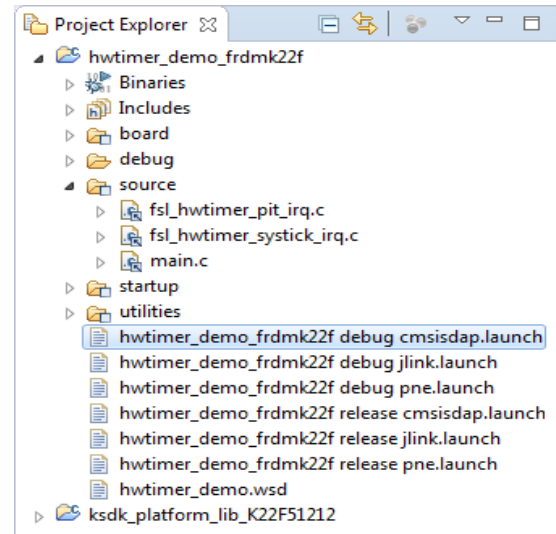




13. Make sure that the 'Baud rate' is set to 115200. Click 'OK'.



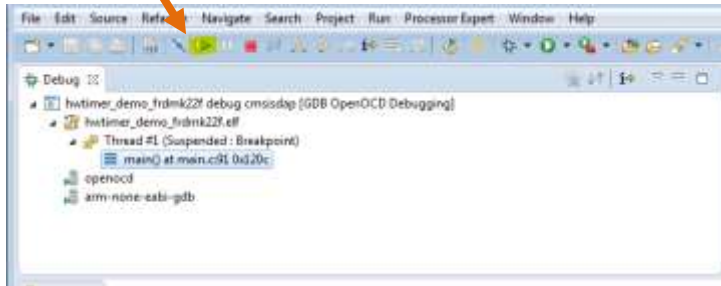
14. Now select the 'hwtimer_demo_frdmk22f120m debug cmsispad.launch' file from the project tree.



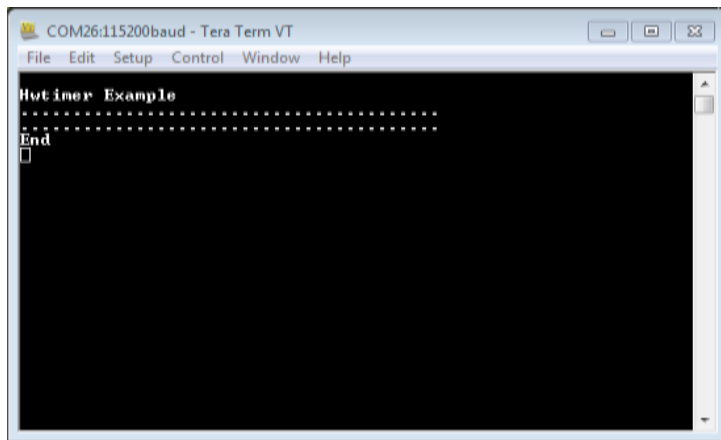
15. Select 'Debug As->hwtimer_demo_frdmk22f debug cmsisdap' from the 'Debug' drop down menu.



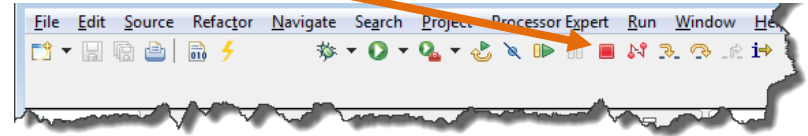
16. The debug session is launched. Code will execute until the first line inside 'main'. Press 'Resume' to continue executing code.



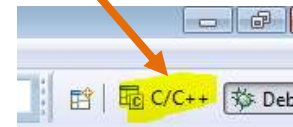
17. In the 'Tera Term' window you should see the following appear. The dots will continue to print for a while, then the 'End' message will appear.



18. The lab is now complete, please terminate the debug session.

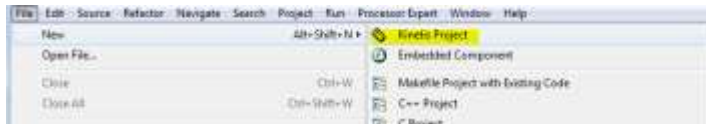


19. Change back to C/C++ perspective by clicking on the C/C++ perspective button.

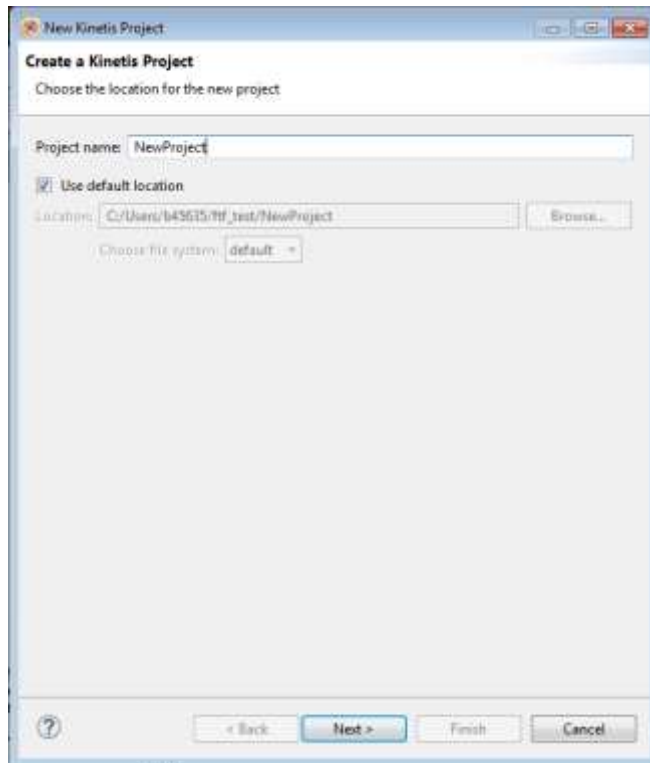


Lab 2: PEx Device Initialization + SDK Drivers

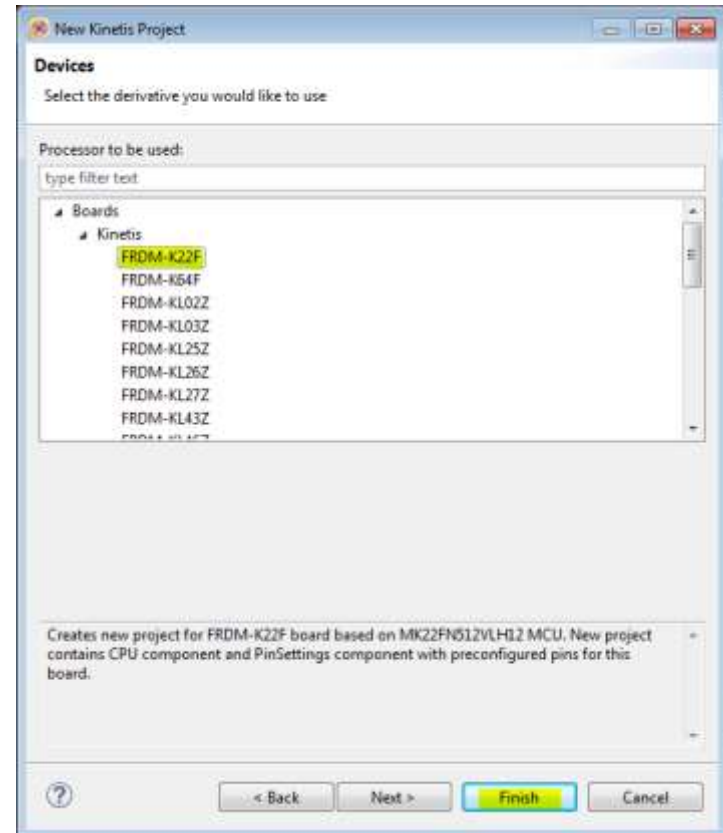
1. Begin a new project. Click 'File', then 'New' and select 'Kinetic Project'.



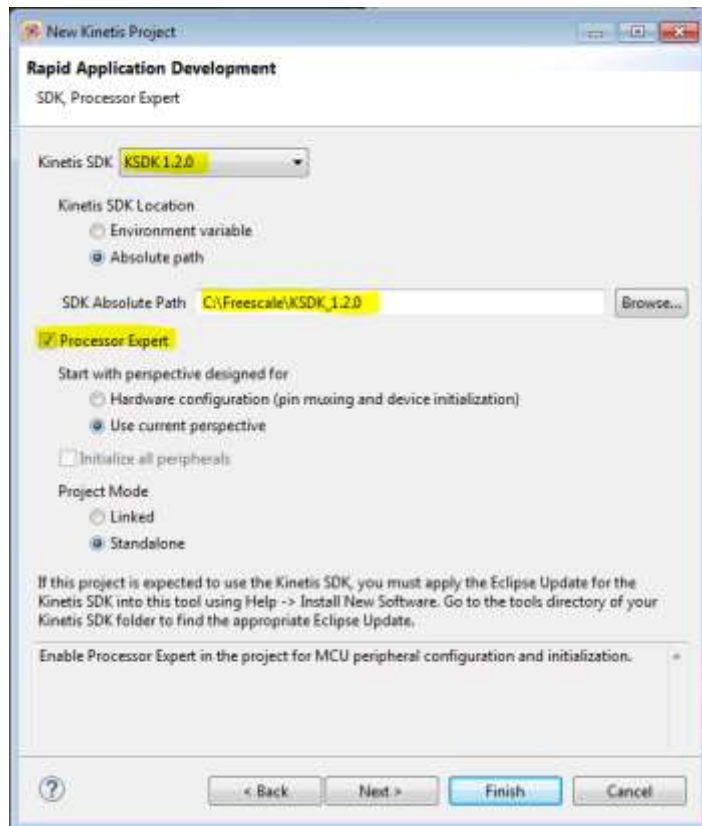
2. Type in the name for the project, 'NewProject'. Click 'Next'.



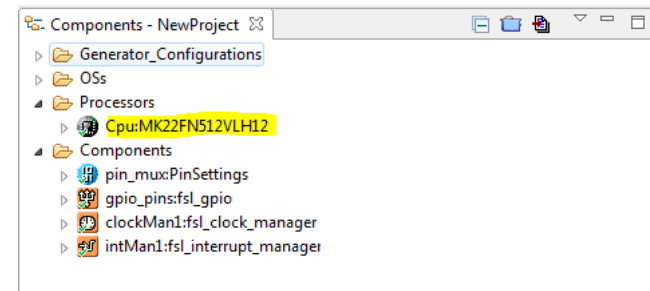
3. Expand the 'Boards' menu, then expand the 'Kinetic' menu. Select 'FRDM-K22F'. Click 'Next'.



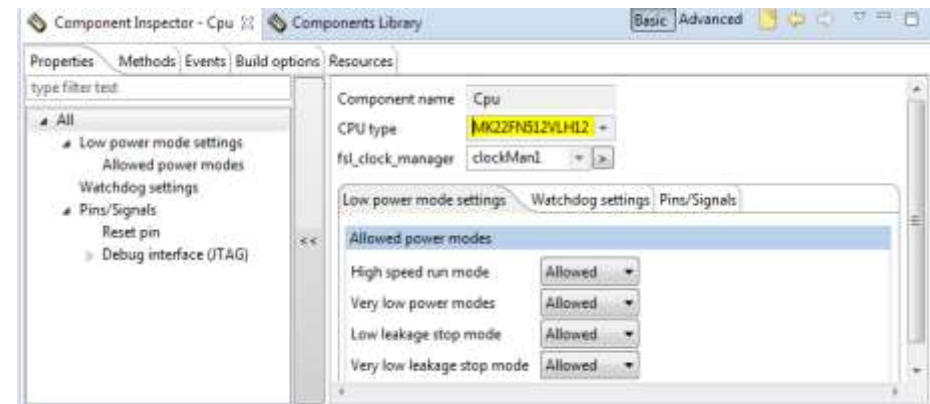
4. Make sure that 'Kinetic SDK' is checked. Now, make sure that 'Processor Expert' is also checked, with 'Use Current Perspective' selected and 'Standalone' mode selected. Clicked 'Finished'.



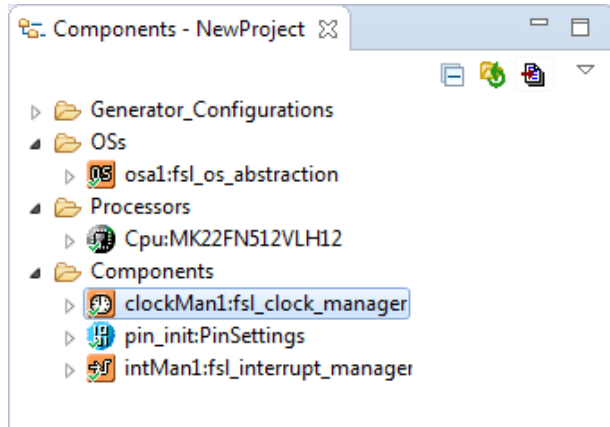
5. In the bottom left of the screen is the 'Components' view. Select the 'Cpu:MK22FN512VLH12' component.



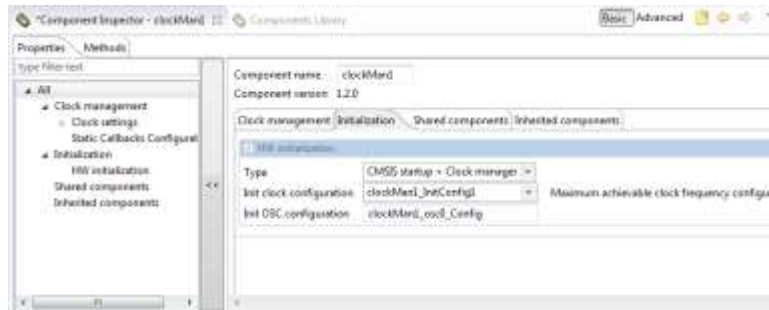
6. At the top of the Kinetis Design Studio window we have the 'Component Inspector' and 'Component Library' views. Select 'Component Inspector' and make sure that the 'CPU type' selection is the 64-pin package.



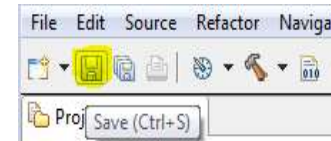
7. Select 'clockMan1:fsl_clock_manager' from the 'Components' view.



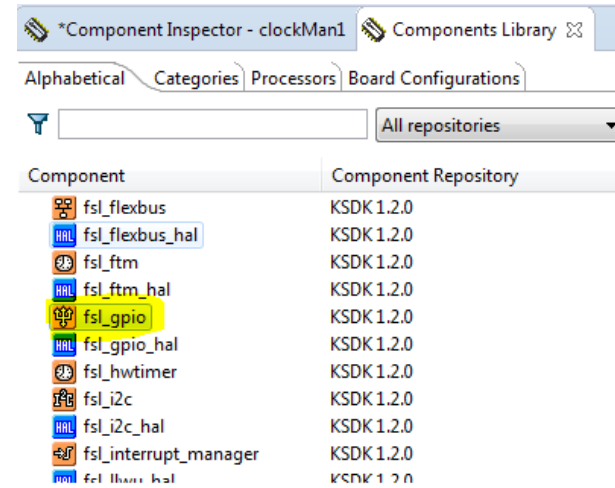
8. Next expand the 'Init clock configuration' menu and select option '1 – Maximum achievable clock frequency configuration'.



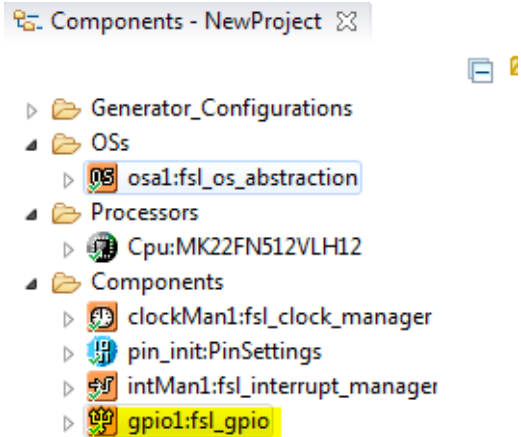
9. Click 'Save' to make sure our changes take effect.



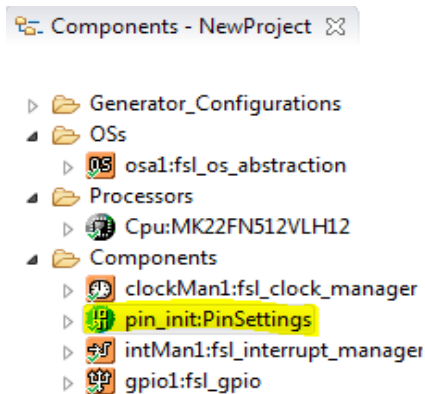
10. Now we are going to import the GPIO component. Select the 'Components Library' tab. Navigate to 'fsl_gpio'. Double-click on the component.



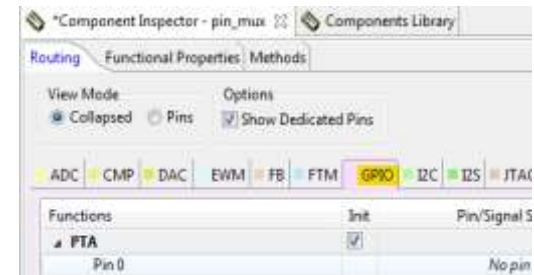
11. A new gpio component is now in the components view.



12. Return to the 'Components' view and select the 'pin_init:PinSettings' component.



13. In the 'Component Inspector' select the 'GPIO' tab.



14. Scroll down to PTA-Pin2.

Signals	Init
PTA	<input checked="" type="checkbox"/>
Pin 0	
Pin 1	
Pin 2	
Pin 3	
Pin 4	

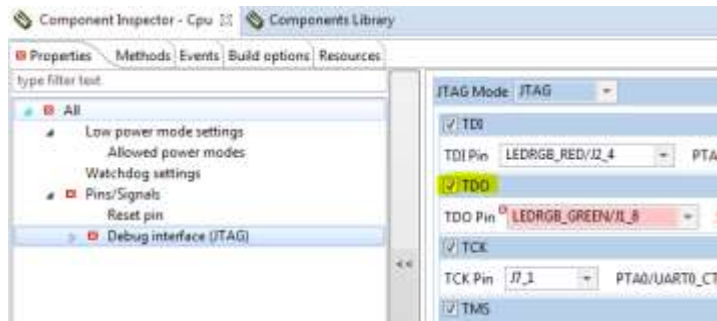
15. Select direction to 'Output'.

Signals	Init	Pin/Signal Selection	Direction
PTA	<input checked="" type="checkbox"/>		
Pin 0		No pin routed	No pin routed
Pin 1		No pin routed	No pin routed
Pin 2		No pin routed	Output
Pin 3		No pin routed	Input
Pin 4		No pin routed	No pin routed

16. Select the 'LEDRGB_GREEN...' Pin/Signal. There will be an error, we will fix it. Save.

Signals	Init	Pin/Signal Selection	Direction	Selected Pin/Signal Name
PTA	<input checked="" type="checkbox"/>			
Pin 0		No pin routed	No pin routed	
Pin 1		No pin routed	No pin routed	
Pin 2		LEDRGB_GREEN[2]_PTA[UART0_TX/PTW_CH/ITAC_IDC/TRAC2_SIBO/EZF_DG]	Output	Selected value is in conflict with other confi...
Pin 3		No pin routed	Input	
Pin 4		No pin routed	No pin routed	
Pin 5		No pin routed	No pin routed	
Pin 15		No pin routed	No pin routed	

17. Return to the CPU component. Select 'Debug interface' in the Component Inspector. Then un-check the 'TDO' box. Save.

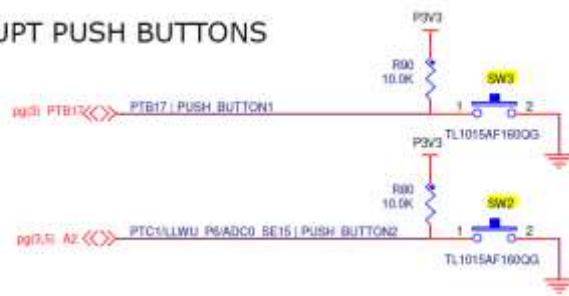


18. Return to 'gpio1' component and edit the output pin configuration. Select the LEDRB_GREEN... pin from the drop down, and configure as below.



19. We are going to set input GPIO for 'SW2' and 'SW3'.

INTERRUPT PUSH BUTTONS



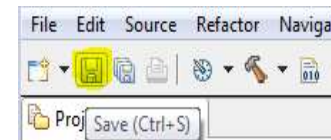
20. In the Component Inspector for pin_init, edit PTB Pin 17 as below for PUSH_BUTTON1.

Pin 16	No pin routed	No pin routed
Pin 17	PUSH_BUTTON1	Input
Pin 18	No pin routed	No pin routed

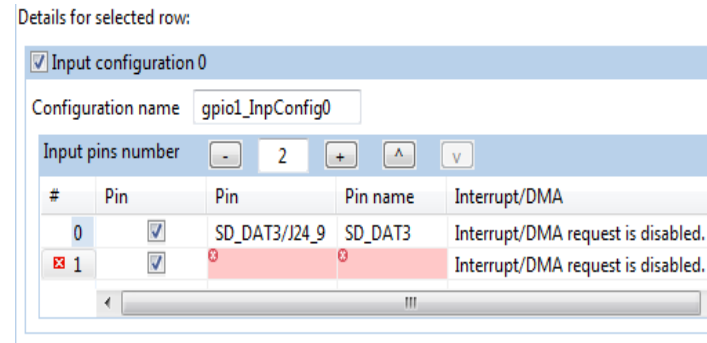
21. Repeat for PTC Pin 1. Setting to PUSH_BUTTON2....

PTC Pin 0	No pin routed	Output
Pin 1	PUSH_BUTTON2/J24_8	Input
Pin 2	No pin routed	No pin routed

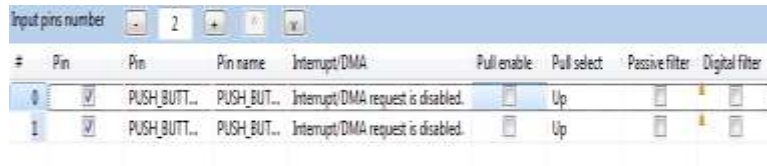
22. Save to keep our changes.



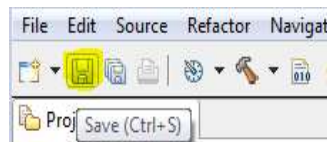
23. Return to gpio1 component. For the 'Input pins' in the Component Inspector, add another pin for Input Configuration 0.



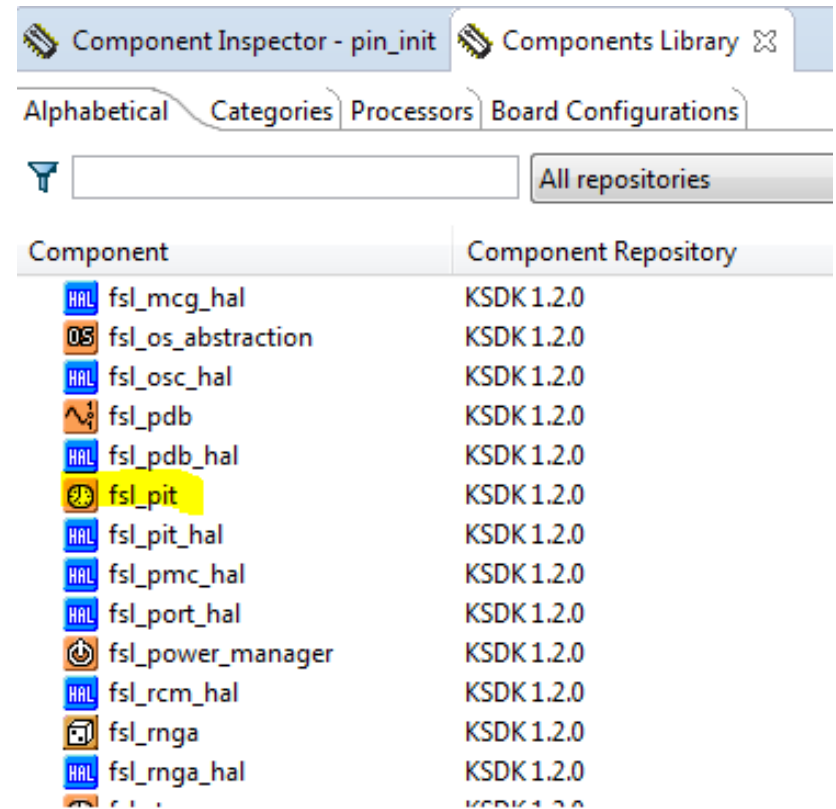
24. Select the input pins that we configured in the pin_init component. Make sure the pins are enabled without pull resistors, and no filters. Save.



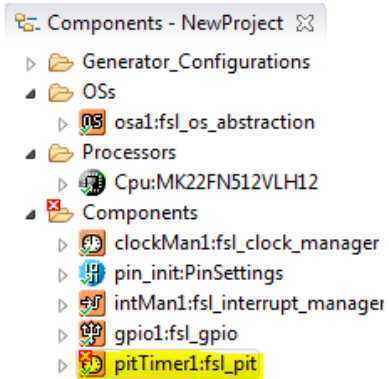
25. Save the component.



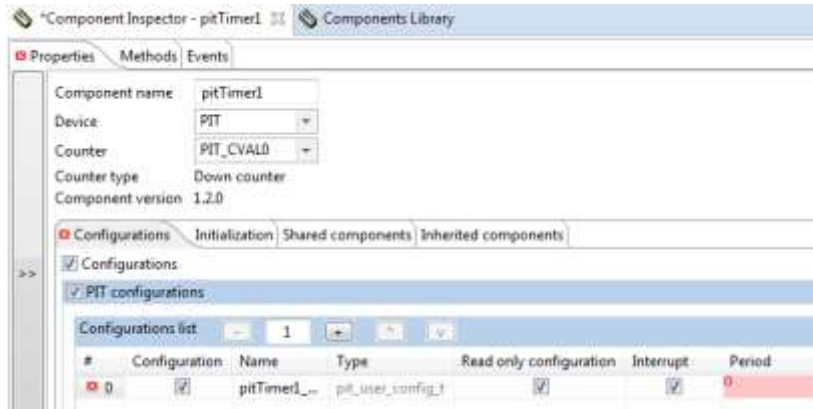
26. Now we are going to add a new component. Go to the 'Component Library' tab at the top of the Kinetis Design Studio window, next to 'Component Inspector' tab. Expand the 'Kinetis SDK' menu, then expand the 'Peripheral Drivers' menu, then the 'Timers' menu. Double-click on the 'fsl_pit' component.



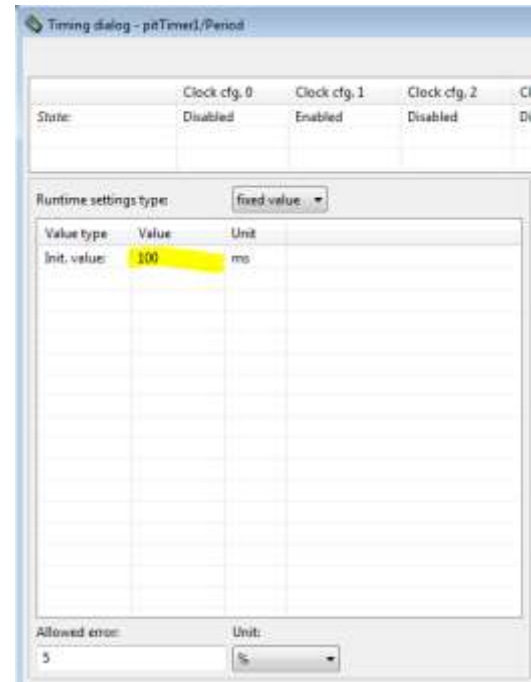
27. Select the 'pitTimer1:fsl_pit' component from the 'Components' view.



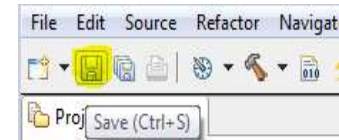
28. In the 'Component Inspector' tab, click on the glyph in the 'Period' field.



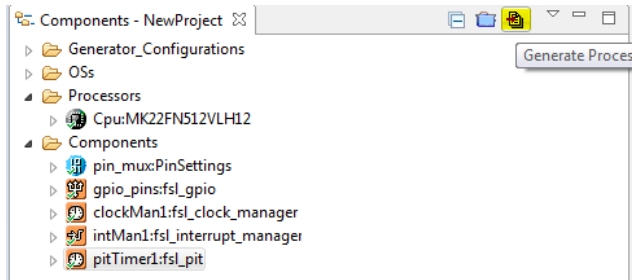
29. A new window will appear. In the 'Value' column, enter '100' ms. Click 'OK'.



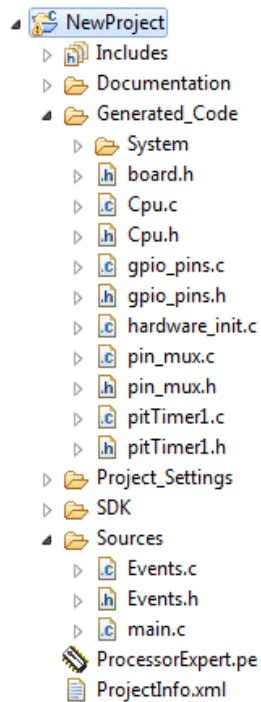
30. Save the component changes.



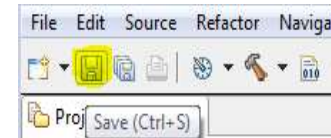
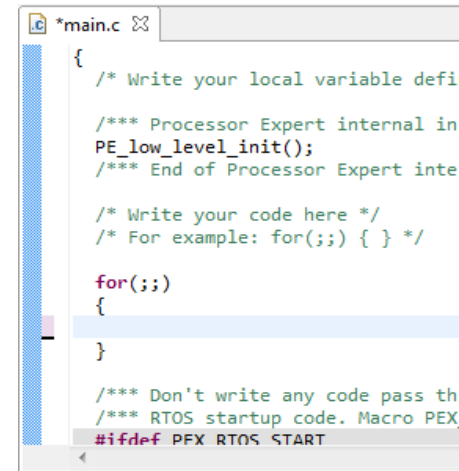
31. Now generate code by clicking on the 'Generate Code' button in the 'Components' view.



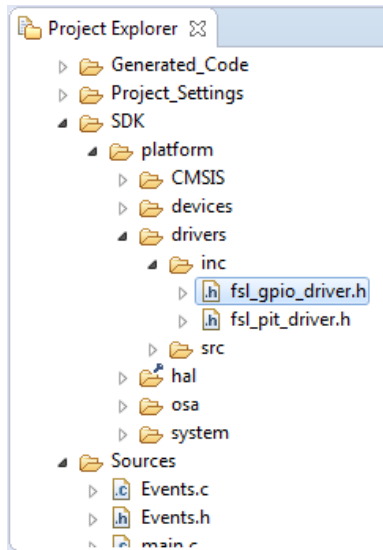
32. The 'NewProject' tree will now contain 'Generate Code' for all the components we have in our project.



33. Open up 'main.c' by double-clicking on it in the project tree. Inside 'main' scroll down to the section where it says 'Write your code here' and place a 'for' loop.



35. Now locate 'fsl_gpio_driver.h' from the project tree.



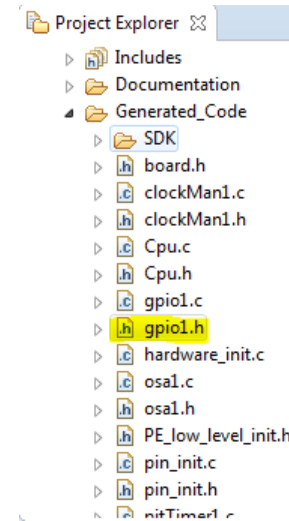
36. Use the 'Outline' view or scroll down to find the 'GPIO_DRV_TogglePinOutput' function. Copy this function call into 'Events.c' by pasting it into the 'pitTimer1_OnTimeOut' callback function.

```

/* User includes (#include below this line is not maintained by Processor
void PIT0_IRQHandler(void)
{
    /* Clear interrupt flag.*/
    PIT_HAL_ClearIntFlag(g_pitBase[FSL_PITTIMER1], FSL_PITTIMER1_CHANNEL);
    /* Write your code here ... */
    GPIO_DRV_TogglePinOutput(UINT32_T pinName);
}
/* END Events.*/

```

37. Now we need to provide a pin name for this function call. Open up the 'gpio1.h' file from the project tree.



38. Use the 'Outline' view or scroll to find 'LEDRGB_GREEN'. Copy this name into the parameter field for 'GPIO_DRV_TogglePinOutput'.

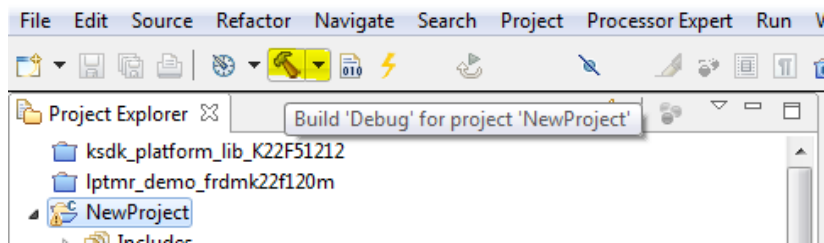
```

void PIT0_IRQHandler(void)
{
    /* Clear interrupt flag.*/
    PIT_HAL_ClearIntFlag(g_pitBase[FSL_PITTIM
    /* Write your code here ... */
    GPIO_DRV_TogglePinOutput(LEDRGB_GREEN);
}

```

39. Save 'Events.c'.

40. Build 'NewProject' by using the 'Hammer' tool.

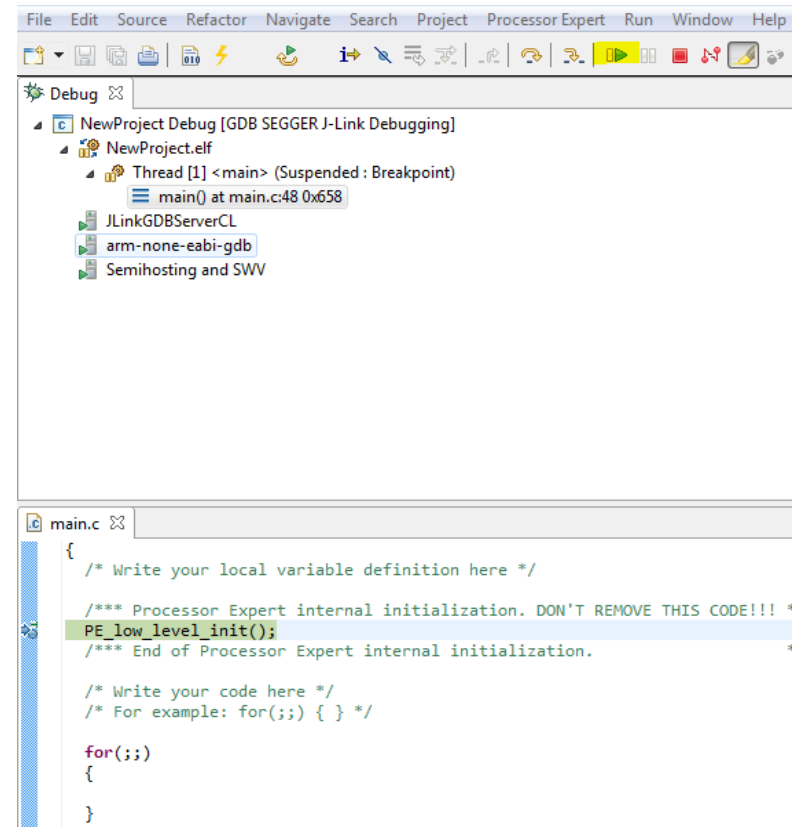


41. This should produce a 'NewProject.elf' file.

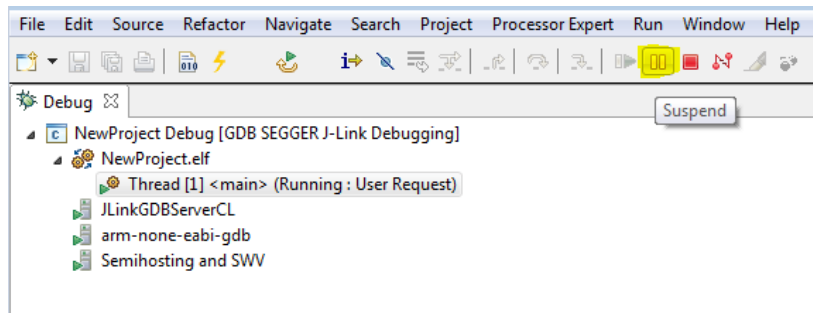
```
Problems Tasks Console Properties
CDT Build Console [NewProject]
'Finished building: ../Generated_Code/pin_mu
'
'Building file: ../Generated_Code/pitTimer1.
'Invoking: Cross ARM C Compiler'
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -m
'Finished building: ../Generated_Code/pitTim
'
'Building target: NewProject.elf'
'Invoking: Cross ARM C++ Linker'
arm-none-eabi-g++ -mcpu=cortex-m4 -mthumb -m
'Finished building target: NewProject.elf'
'
11:50:48 Build Finished (took 52s.153ms)
```

42. Now select the CMSIS-DAP debug configuration for 'NewProject'.

43. Once the debug session is launched, it will run to 'main' and halt on a breakpoint by the 'PE_low_level_init' function call. Press the 'Resume' button to continue code execution.



44. The Green LED (in the RGB LED) on the FRDM-K22F should now be flashing. To catch the code entering the PIT callback function we will set a breakpoint. First suspend the debug session by pressing the suspend button.

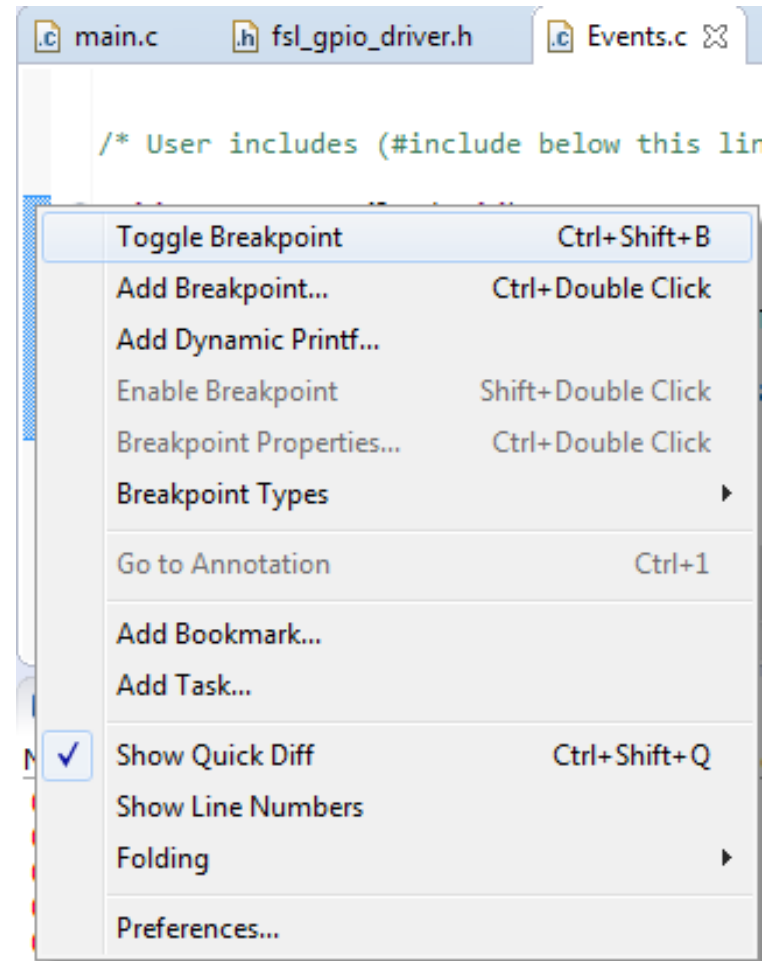


45. Navigate to the 'Events.c' file. Scroll to the 'PIT0_IRQHandler' function.

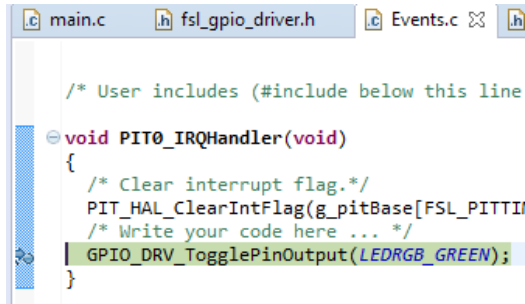
```

void PIT0_IRQHandler(void)
{
    /* Clear interrupt flag.*/
    PIT_HAL_ClearIntFlag(g_pitBase[FSL_PITTIM
    /* Write your code here ... */
    GPIO_DRV_TogglePinOutput(LED_RGB_GREEN);
}
  
```

46. Right-click in margin, next to the 'GPIO_DRV_TogglePinOutput' function call. This will bring up a list of options. Click on 'Toggle Breakpoint' to set the breakpoint.



47. You can now resume the Debug session, and see the code halt on this breakpoint.

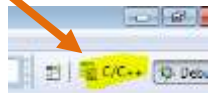


```
main.c | fsl_gpio_driver.h | Events.c | .h  
  
/* User includes (#include below this line  
void PIT0_IRQHandler(void)  
{  
    /* Clear interrupt flag.*/  
    PIT_HAL_ClearIntFlag(g_pitBase[FSL_PITTI  
    /* Write your code here ... */  
    GPIO_DRV_TogglePinOutput(LED_RGB_GREEN);  
}
```

48. After you have finished testing the code and breakpoint you can terminate the debug session.

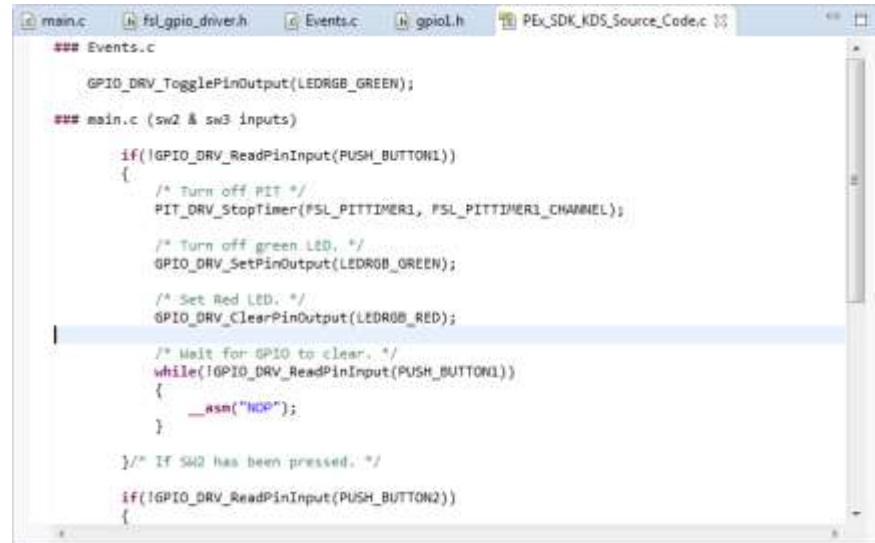


49. Change back to the C/C++ perspective by pressing the C/C++ perspective button.



50. Now we will begin to add more code to this application. In 'main.c' we are going to add some code into the 'for' loop we created earlier. The code we are going to add already written for you. Navigate to the folder '3. Lab Source Code' and then drag the file 'PEx_SDK_KDS_Source_Code.c' into the 'Editor'. Note that you must be in C/C++ perspective for this to work.

51. Once the file is open in the 'Editor', highlight the source code from the section '### main.c (sw2 & sw3 inputs)'.



```
main.c | fsl_gpio_driver.h | Events.c | gpioL.h | PEx_SDK_KDS_Source_Code.c  
  
### Events.c  
GPIO_DRV_TogglePinOutput(LED_RGB_GREEN);  
  
### main.c (sw2 & sw3 inputs)  
if(!GPIO_DRV_ReadPinInput(PUSH_BUTTON1))  
{  
    /* Turn off PIT */  
    PIT_DRV_StopTimer(FSL_PITTIMER1, FSL_PITTIMER1_CHANNEL);  
  
    /* Turn off green LED. */  
    GPIO_DRV_SetPinOutput(LED_RGB_GREEN);  
  
    /* Set Red LED. */  
    GPIO_DRV_ClearPinOutput(LED_RGB_RED);  
  
    /* Wait for GPIO to clear. */  
    while(!GPIO_DRV_ReadPinInput(PUSH_BUTTON1))  
    {  
        __asm("NOP");  
    }  
}  
  
/* If SW0 has been pressed. */  
if(!GPIO_DRV_ReadPinInput(PUSH_BUTTON2))  
{
```

52. Copy and paste this code into the 'for' loop in 'main.c'.

```

main.c 33 PEx_SDK_KDS_Source_Code.c
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(;;) { } */

for(;;)
{
if(!GPIO_DRV_ReadPinInput(kGpioSW2))
{
/* Turn off PIT */
PIT_DRV_StopTimer(FSL_PITTIMER1, FSL_PITTIMER1_CHANNEL);

/* Turn off green LED. */
GPIO_DRV_SetPinOutput(kGpioLED1);

/* Set Red LED. */
GPIO_DRV_ClearPinOutput(kGpioLED2);

/* Wait for GPIO to clear. */
while(!GPIO_DRV_ReadPinInput(kGpioSW2))
{
__asm("NOP");
}

printf("\r\nSW2 has been pressed.\r\n");
}/* If SW2 has been pressed. */

if(!GPIO_DRV_ReadPinInput(kGpioSW3))
{
/* Turn off red LED. */
GPIO_DRV_SetPinOutput(kGpioLED2);

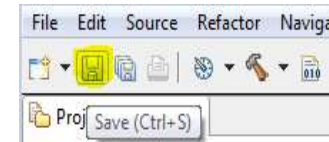
/* Turn on PIT */
PIT_DRV_StartTimer(FSL_PITTIMER1, FSL_PITTIMER1_CHANNEL);

/* Wait for GPIO to clear. */
while(!GPIO_DRV_ReadPinInput(kGpioSW3))
{
__asm("NOP");
}

printf("\r\nSW3 has been pressed.\r\n");
}/* If SW3 has been pressed. */
}
}

```

53. Save 'main.c'.

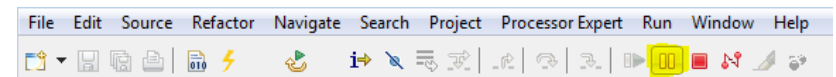


54. Before the project will build we need to add the RED led to our components. The RED led is on pin PTA Pin1, follow the steps used previously to add GPIO to add LEDRGB_RED to the project. Then re-generate code.

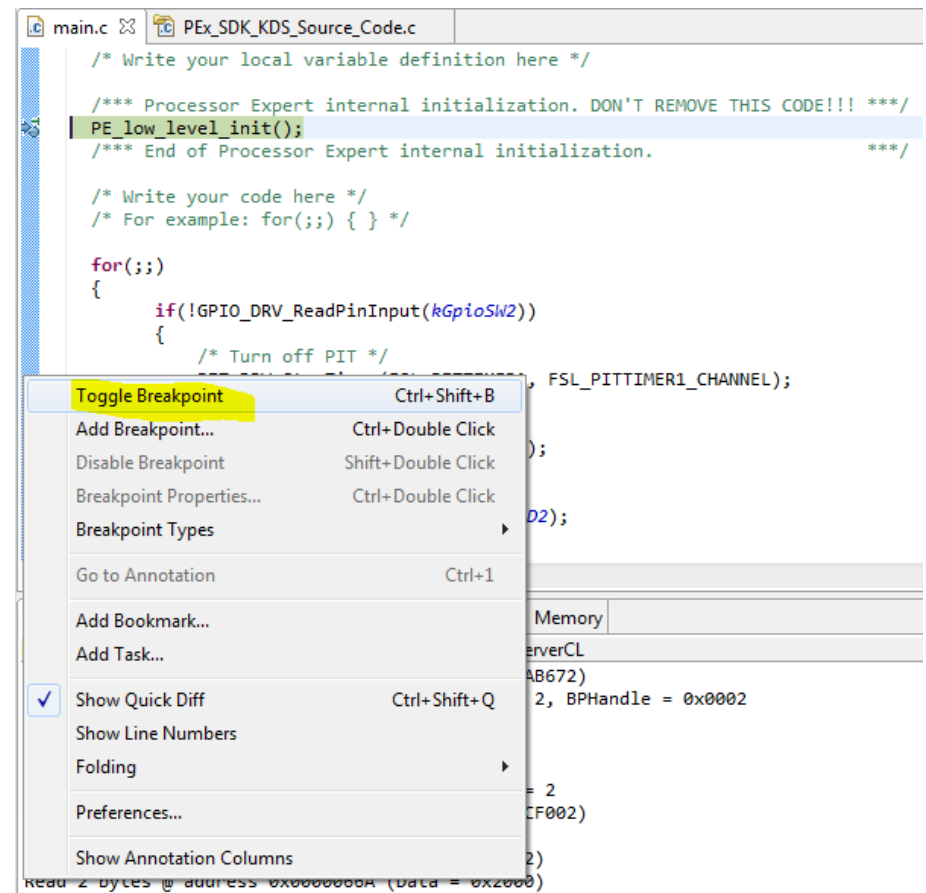
Signals	Init	Pin/Signal Selection	Direction
PTA	<input checked="" type="checkbox"/>		
Pin 0		No pin routed	No pin routed
Pin 1		LEDRGB_RED/I2_4	Not Specified
Pin 2		LEDRGB_GREEN/I1_8	Output
Pin 3		No pin routed	No pin routed

55. Now let's begin another debug session to test what we have added. Repeat steps 38 through 41 to build the project, start a debug session, and begin executing code. Once the debug session is started the Green LED will begin flashing. To halt the flashing LED, press SW2 and the Red LED will come on. To restart the flashing, and turn off the Red LED, press SW3.

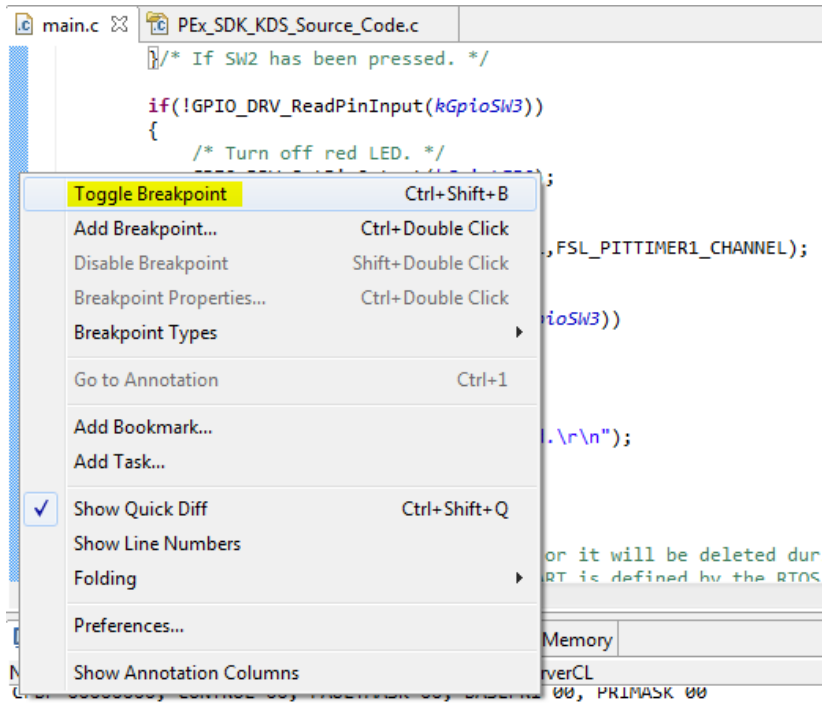
56. Suspend the debug session.



57. Place a couple breakpoints in the code to catch each button being pressed. Right-click in the margin next to 'PIT_DRV_StopTimer' function call to set a breakpoint to wait for SW2 to be pressed.



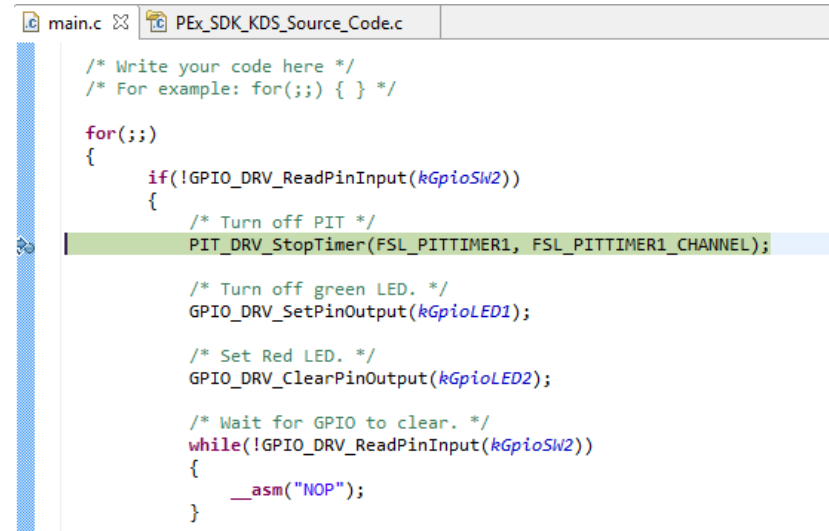
58. Right-click in the margin next to the 'GPIO_DRV_SetPinOutput(LED_RGB_RED)' function call to catch when SW3 has been pressed.



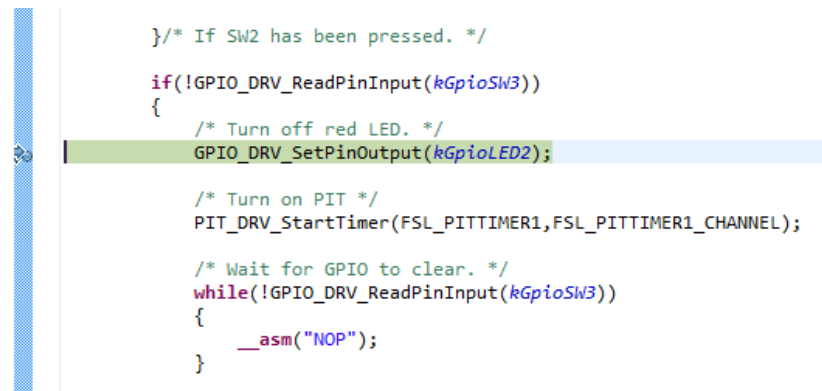
59. When ready, press Resume to restart the Debug session.



60. When SW2 is pressed the Debug session should halt at the following breakpoint:



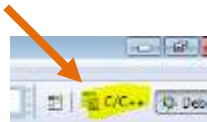
61. When SW3 is pressed the code will halt at the breakpoint shown below:



62. When you have finished testing the application, terminate the debug session.



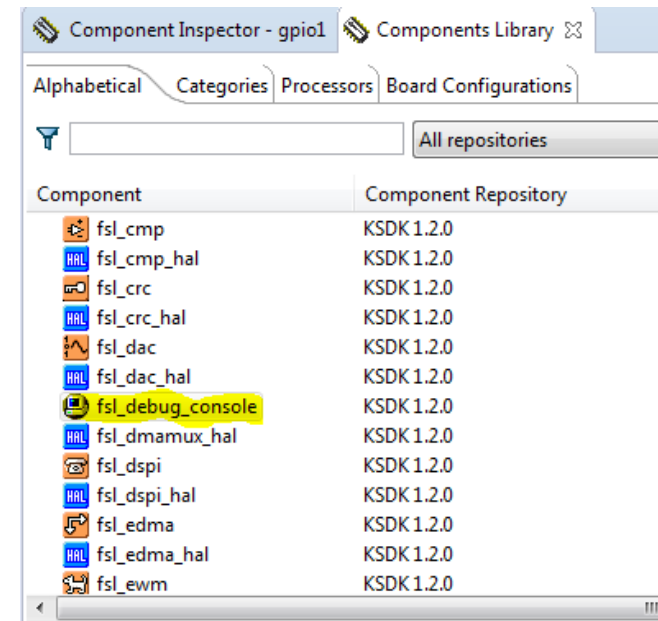
63. Return to the C/C++ perspective by pressing the C/C++ perspective button.



64. This concludes Lab 4.

Going Further: Adding Debug Console

65. Now let us add the fsl_debug_console component to our project. Using the Components Library, scroll down to find the fsl_debug_console component. Double-click to bring the component into your project.



- 66. Using the Component Inspector, configure the Debug Console to use UART1 @ 115200 bps on Pins PTE0 and PTE1.
- 67. Once added you will now be able to use PRINTF statements in the code to signal when each button is pressed. Use your terminal program to view the output from the development board. See PEx_SDK_KDS_Source_Code.c for PRINTF examples.