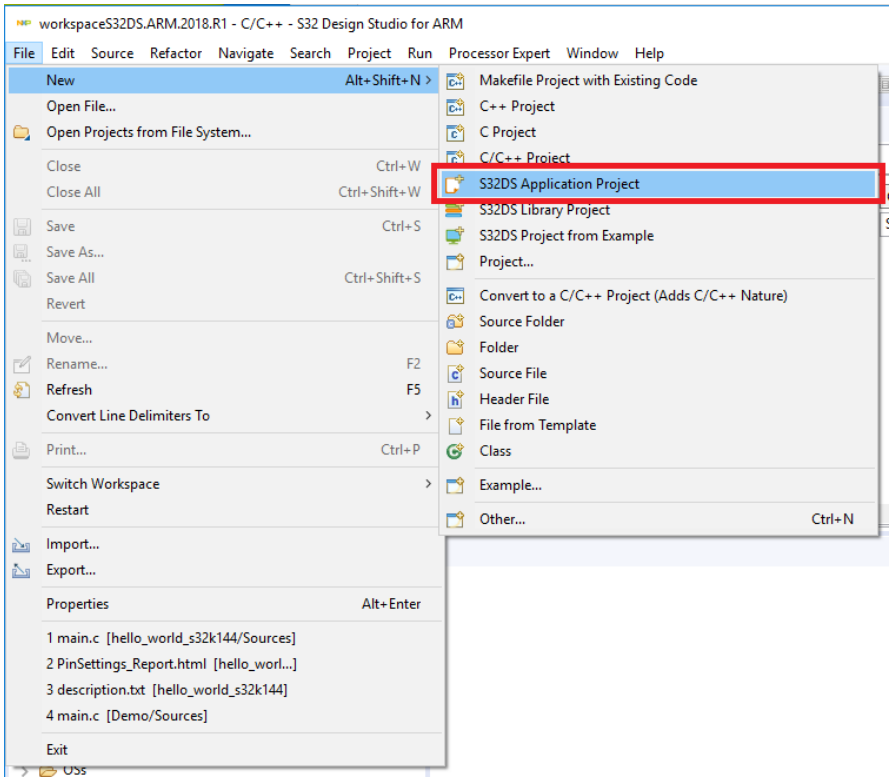
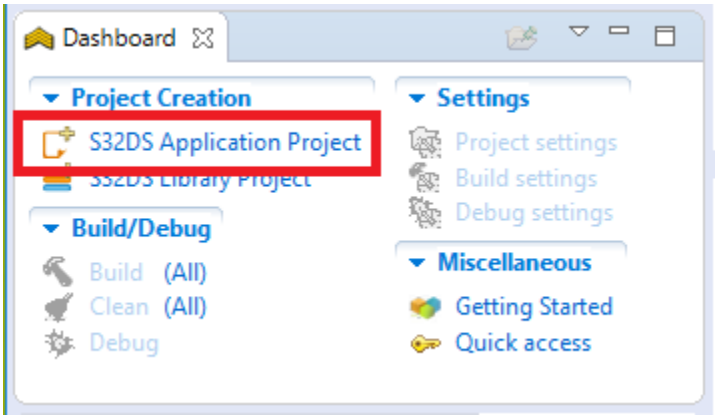
HOWTO: Create a Blinking LED example project using S32K144 SDK

This document shows the step-by-step process to create a simple 'Blinking_LED' project. There is also a video which demonstrates the same steps. This project uses the S32K144EVB-Q100 EVB, connected to a PC through USB (OpenSDA) connection.

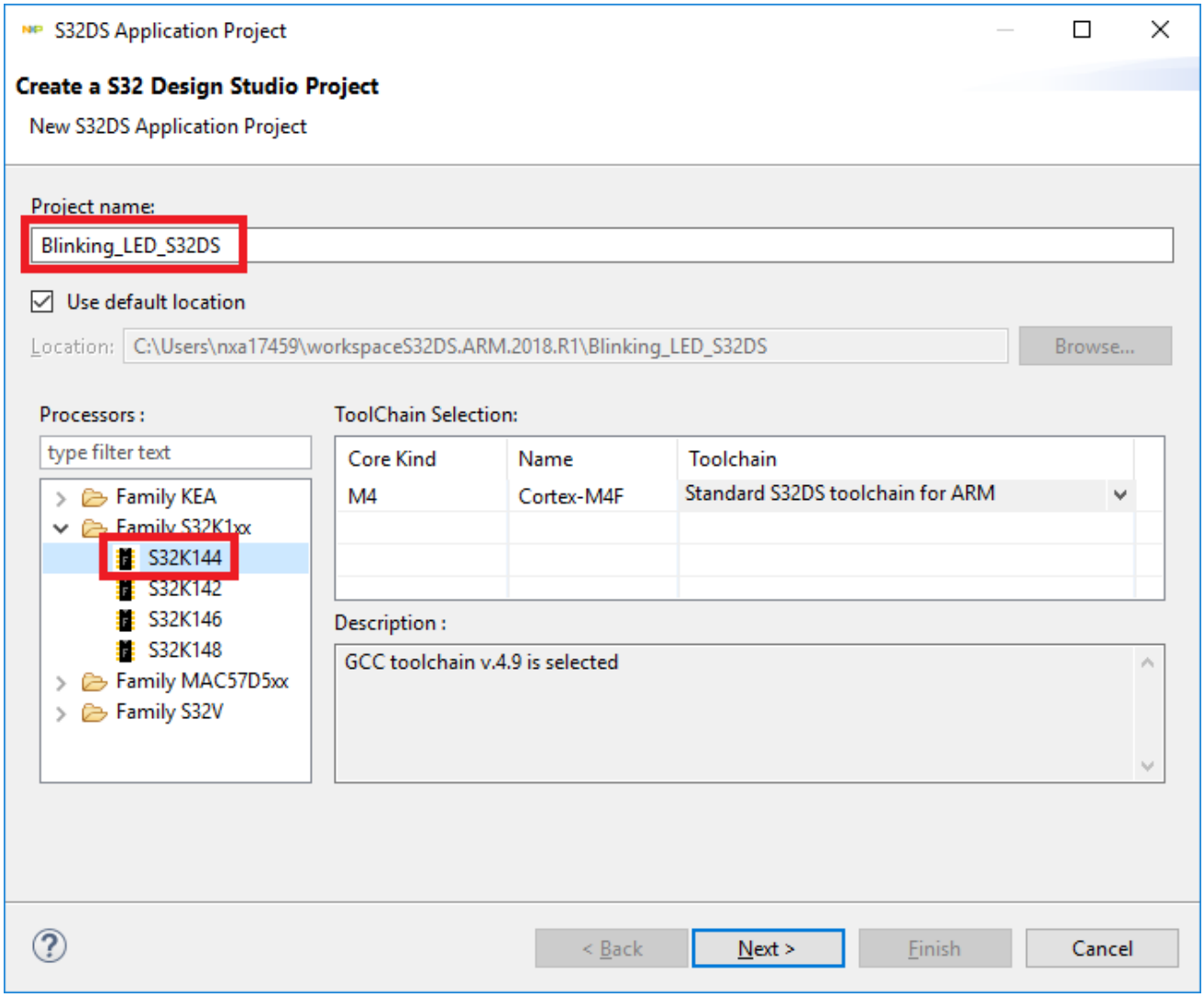
1. New S32DS Project



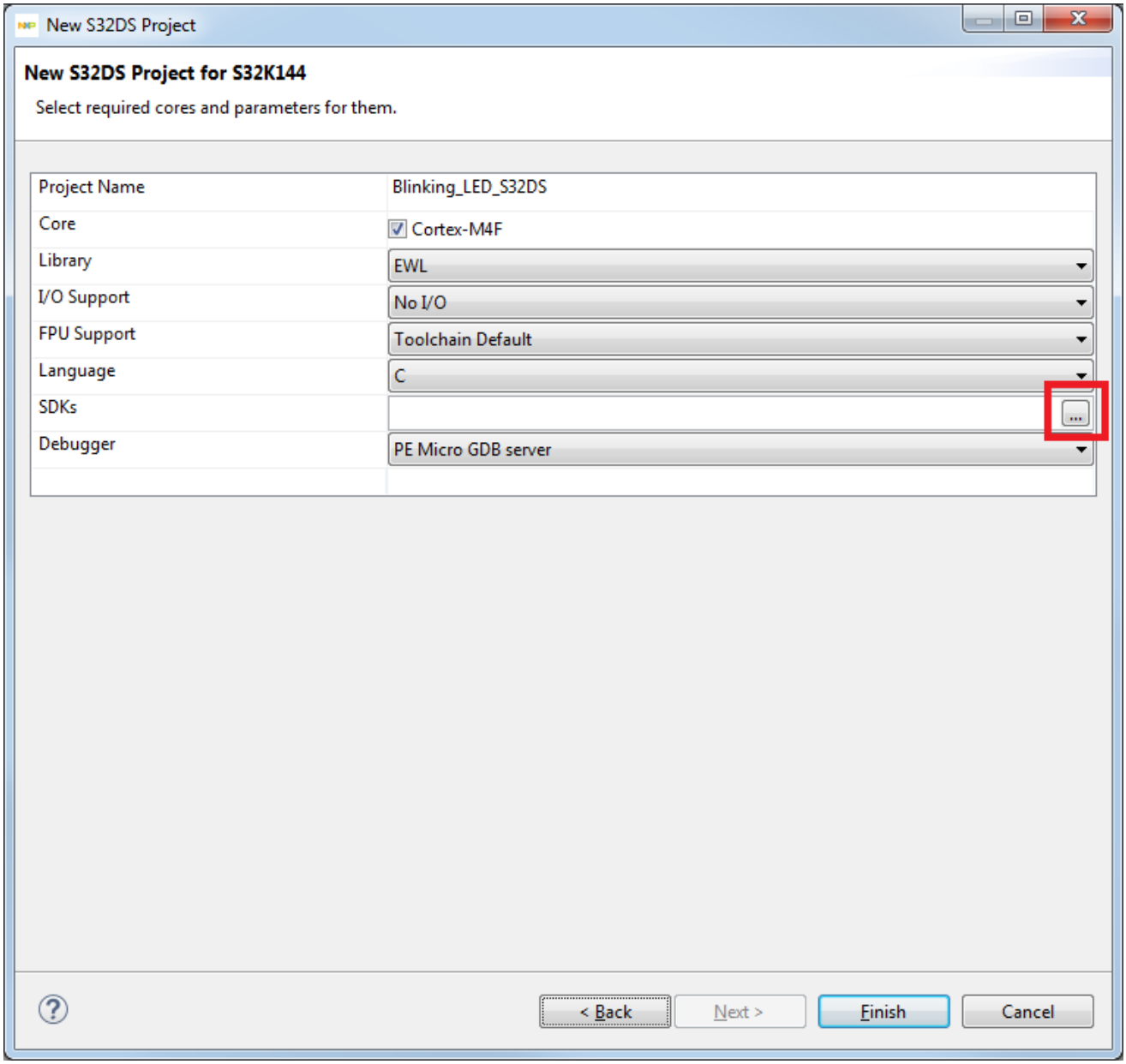
OR



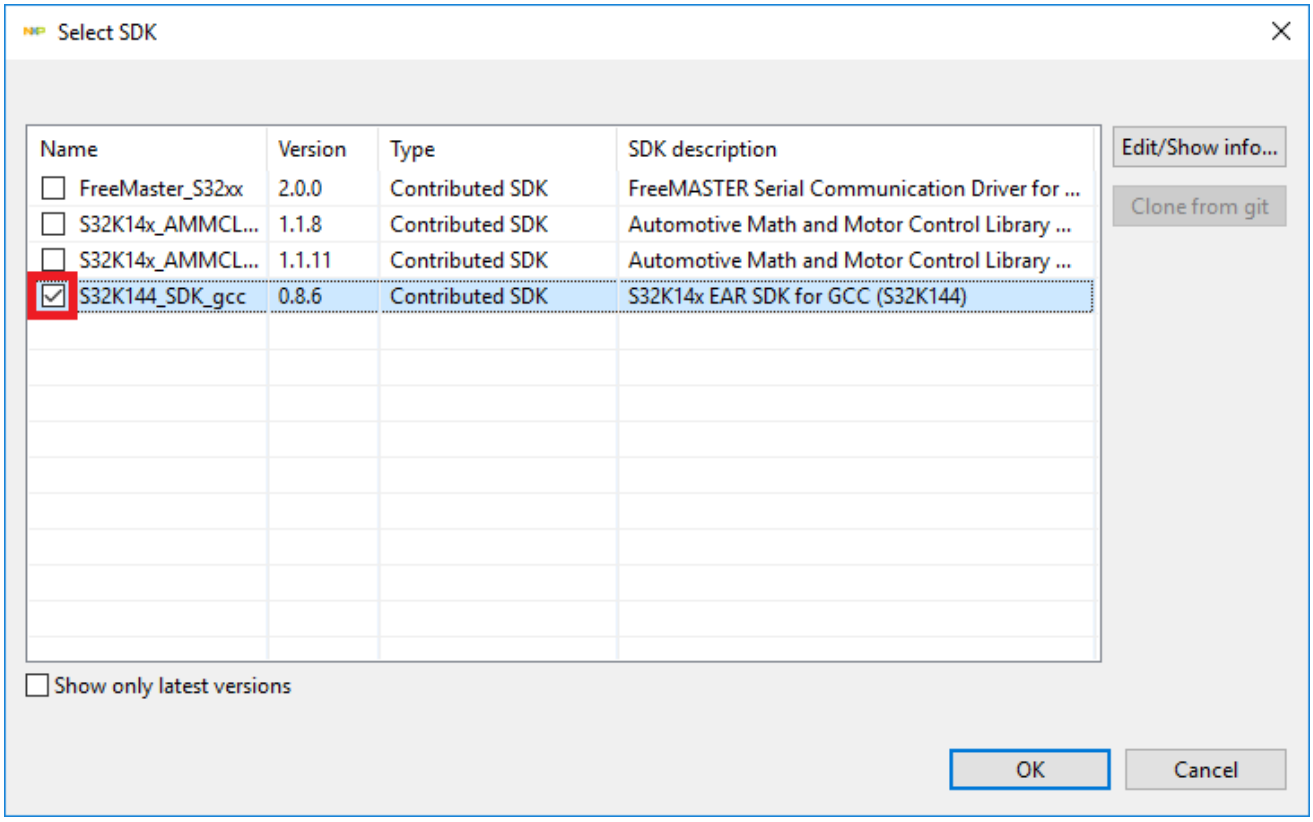
2. Provide a name for the project, for example 'Blinking_LED_S32DS'. The name must be entered with no space characters.
3. Expand Family S32K1xx, Select S32K144
4. Click **Next**



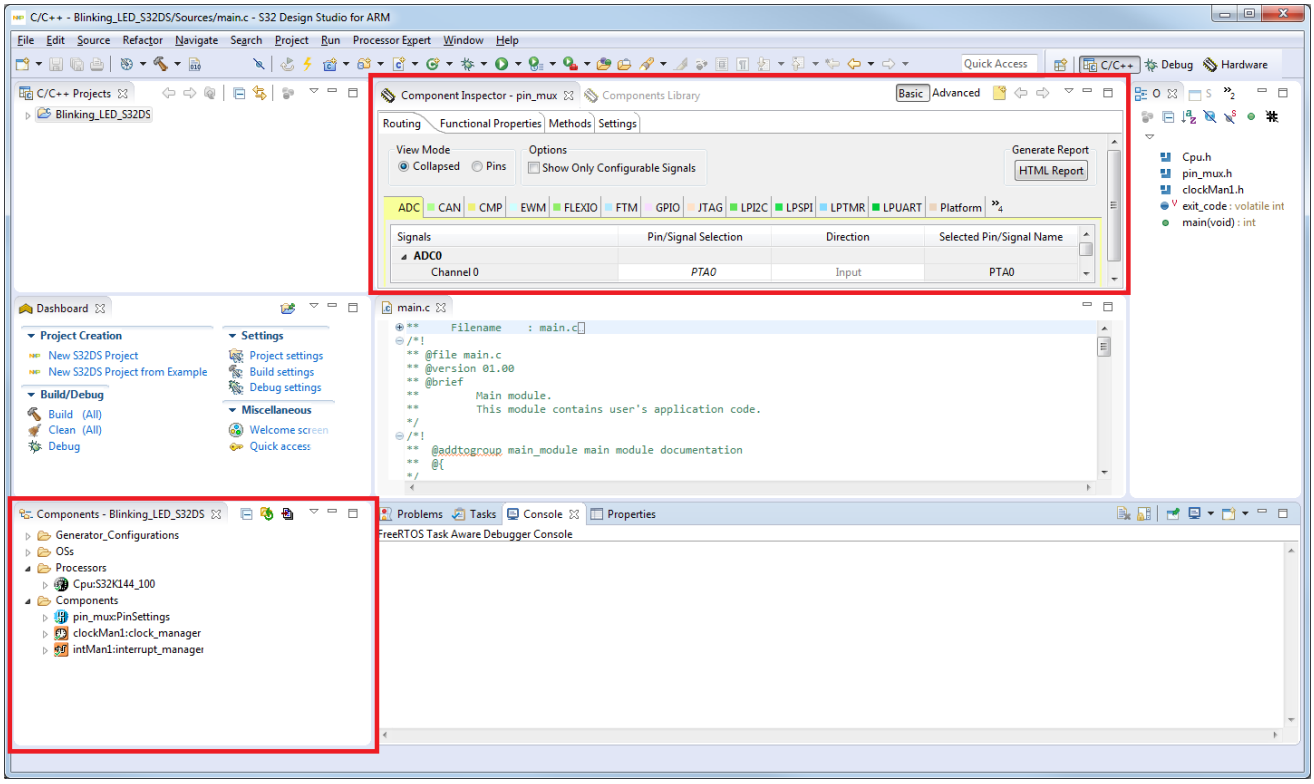
5. Click '…' button next to SDKs



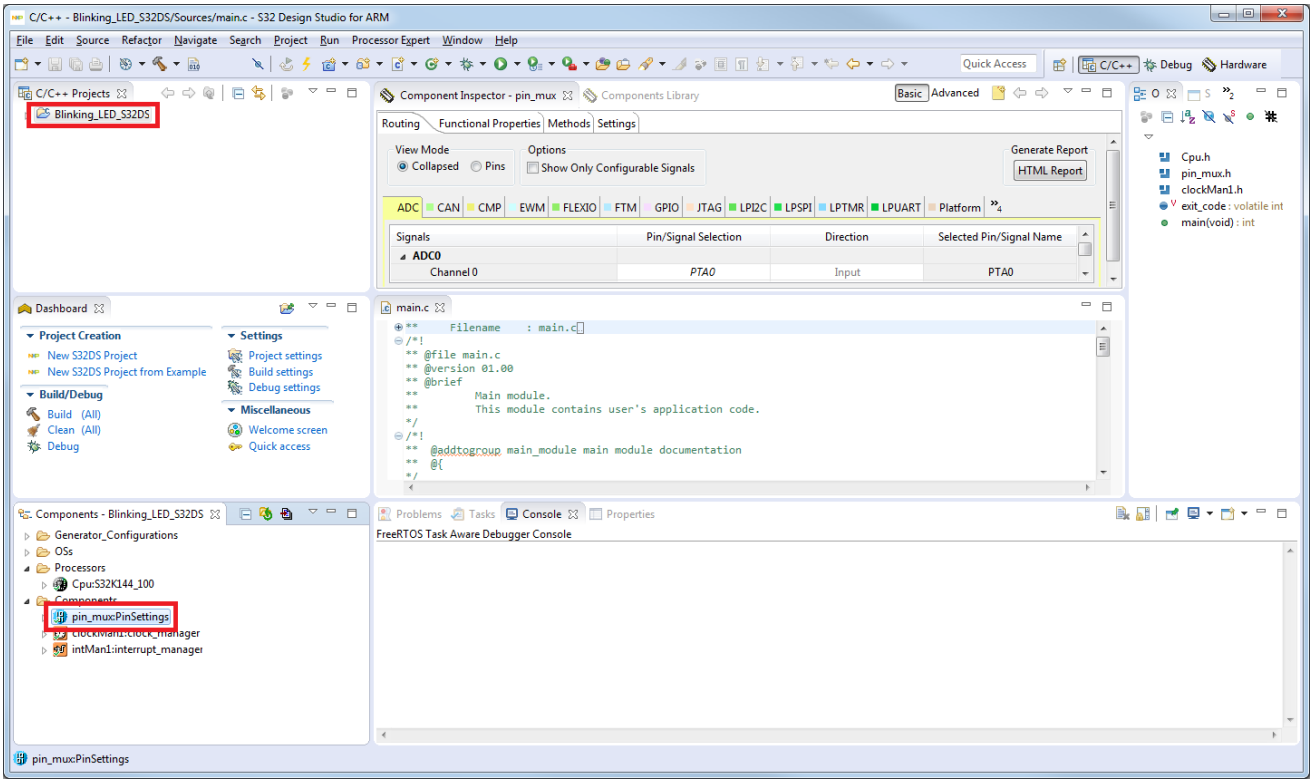
6. Check box next to S32K144_SDK_gcc. It may be necessary to check the release notes for the SDK to confirm which version to select.
7. Click **OK**



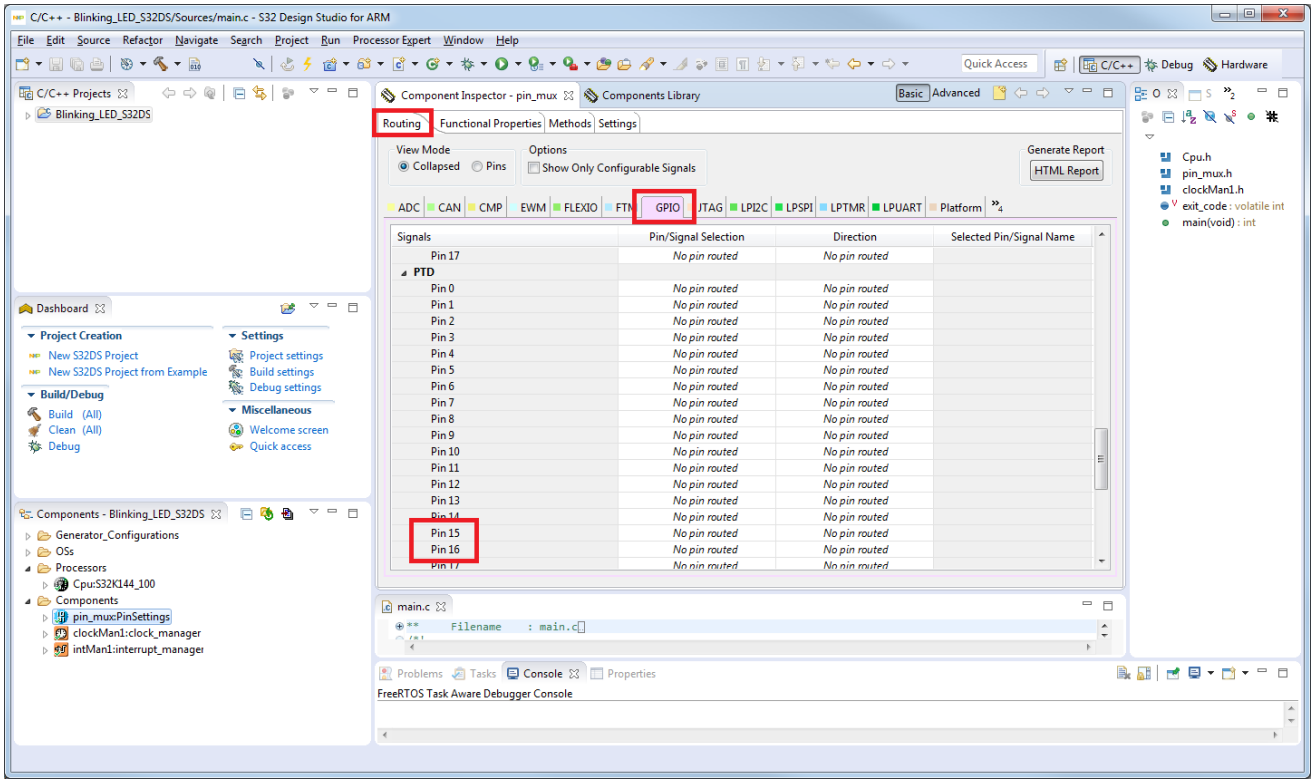
8. Click Finish, wait for project generation wizard to complete
9. Notice Processor Expert views have opened.



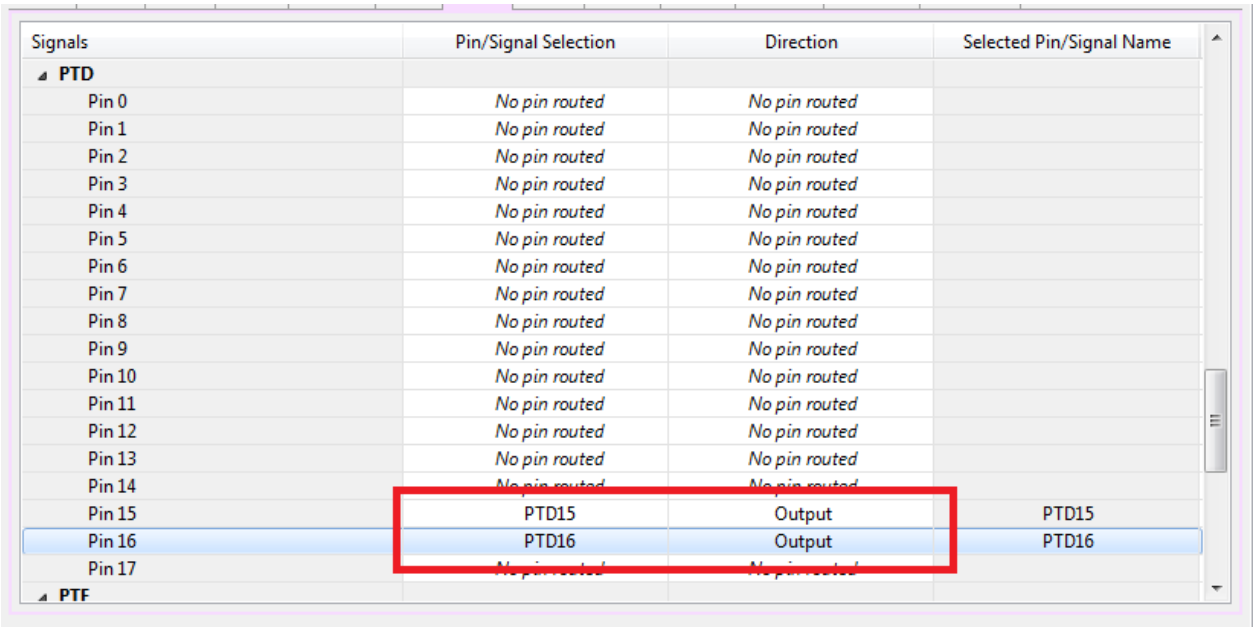
10. Make sure Blinking_LED_S32DS is selected in Project Explorer view, then from the Components- Blinking_LED_S32DS view, select: Components -> **pinmux:PinSettings**



11. From the Routing tab, select the GPIO pin routing group and scroll the list until PTD Pin 15 and Pin 16 are visible

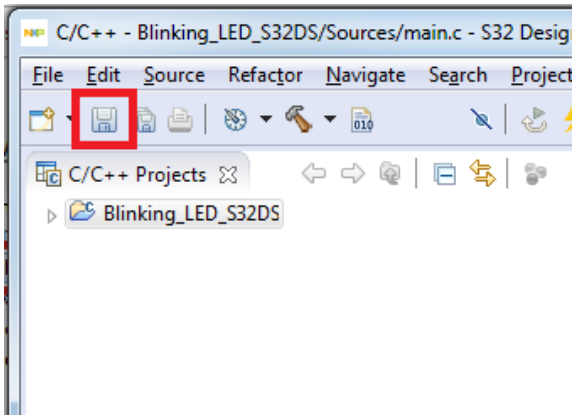


12. On the row Pin 15, in the column Pin/Signal Selection set to PTD15, in the column Direction set to Output
13. On the row Pin 16, in the column Pin/Signal Selection set to PTD16, in the column Direction set to Output

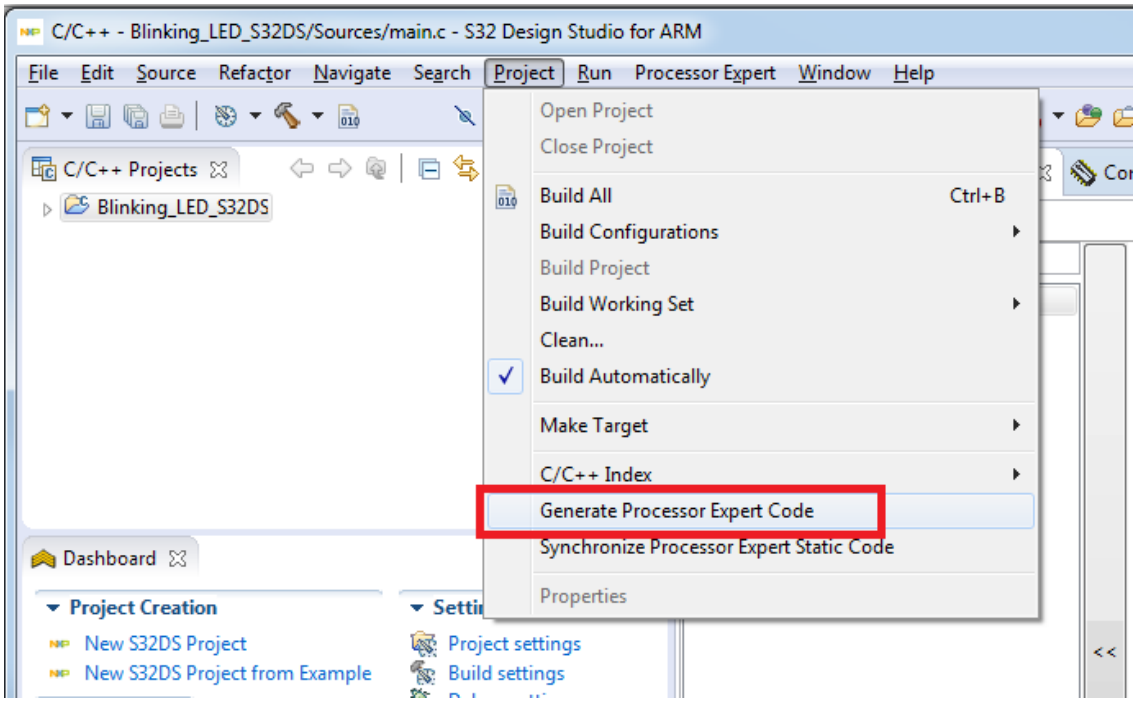


Signals	Pin/Signal Selection	Direction	Selected Pin/Signal Name
PTD			
Pin 0	No pin routed	No pin routed	
Pin 1	No pin routed	No pin routed	
Pin 2	No pin routed	No pin routed	
Pin 3	No pin routed	No pin routed	
Pin 4	No pin routed	No pin routed	
Pin 5	No pin routed	No pin routed	
Pin 6	No pin routed	No pin routed	
Pin 7	No pin routed	No pin routed	
Pin 8	No pin routed	No pin routed	
Pin 9	No pin routed	No pin routed	
Pin 10	No pin routed	No pin routed	
Pin 11	No pin routed	No pin routed	
Pin 12	No pin routed	No pin routed	
Pin 13	No pin routed	No pin routed	
Pin 14	No pin routed	No pin routed	
Pin 15	PTD15	Output	PTD15
Pin 16	PTD16	Output	PTD16
Pin 17	No pin routed	No pin routed	
PTE			

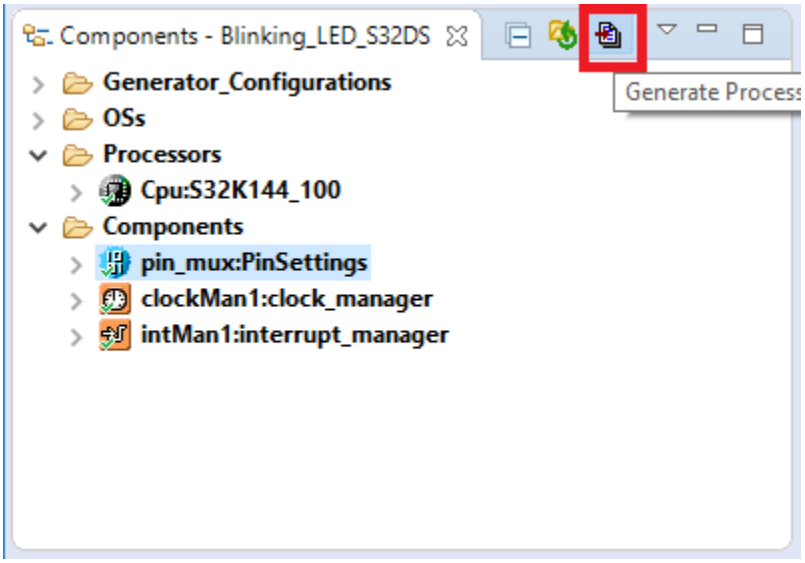
14. Click **Save**



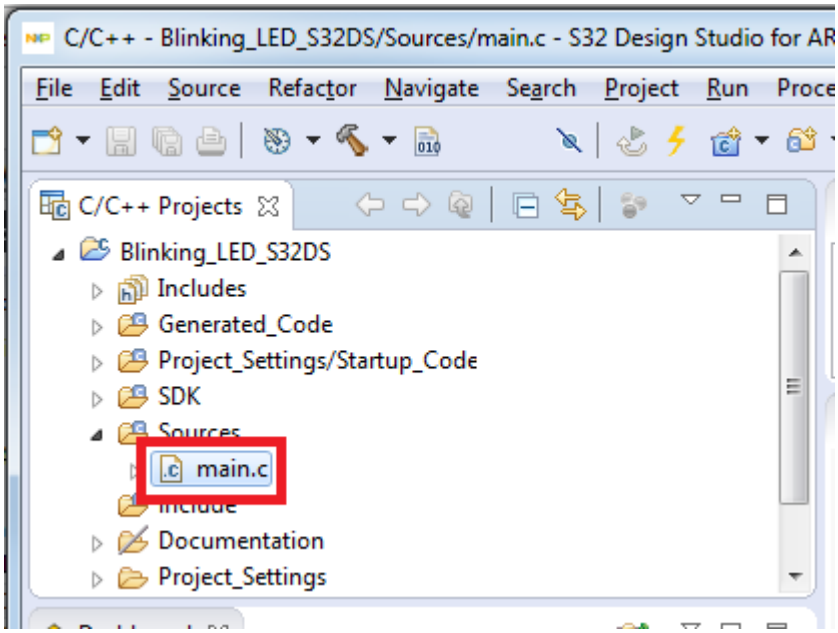
15. Project -> Generate Processor Expert Code



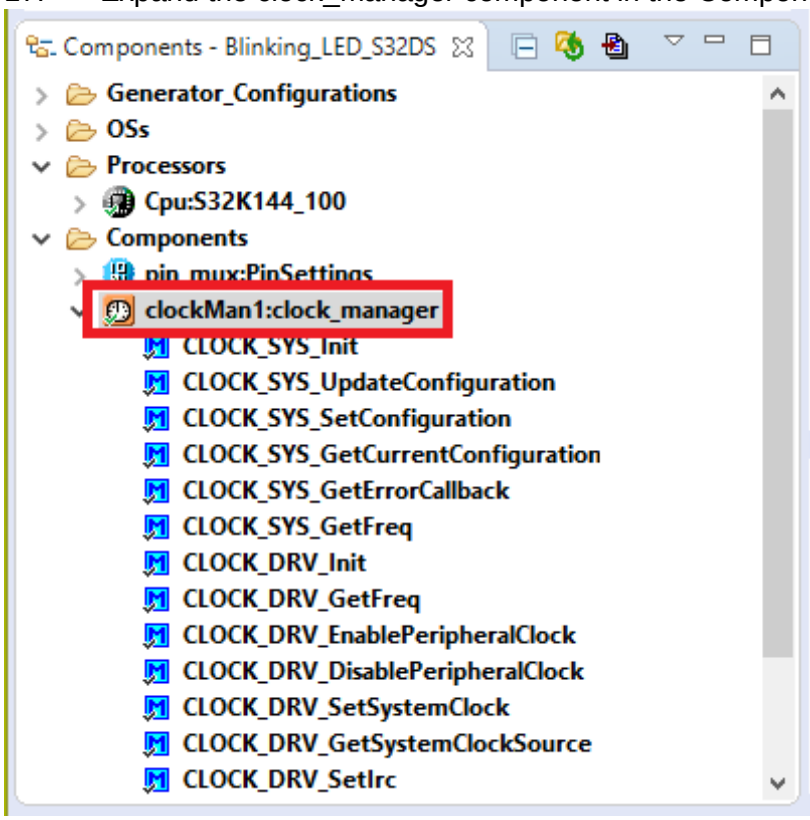
OR



16. If not already open, in the project window double click the main.c file to open it.



17. Expand the clock_manager component in the Components Window



18. Drag and drop the **CLOCK_SYS_Init** function into main, after the comment 'Write your code here'

```
*main.c
 * - startup asm routine
 * - main()
 */
int main(void)
{
  /* Write your local variable definition here */

  /*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
  #ifdef PEX_RTOS_INIT
    PEX_RTOS_INIT();                  /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
  #endif
  /*** End of Processor Expert internal initialization.                    ***/

  /* Write your code here */
  CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT, g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
  /* 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(;;) {
    if(exit_code != 0) {
      break;
```

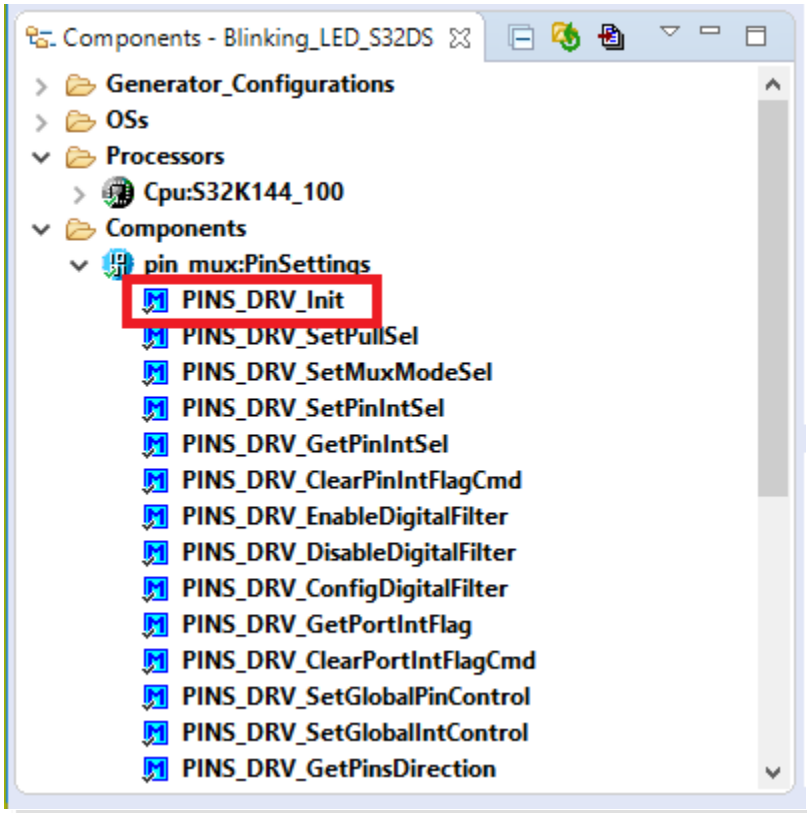
19. Drag and drop the **CLOCK_SYS_UpdateCo**nfiguration function into main
20. In the **CLOCK_SYS_UpdateConfiguration** add the following parameters.

    0U,
    CLOCK_MANAGER_POLICY_FORCIBLE

Notice after the second parameter is added, it turns blue and when the mouse pointer is hovered over it the value is displayed. This shows it is recognized as a defined macro.

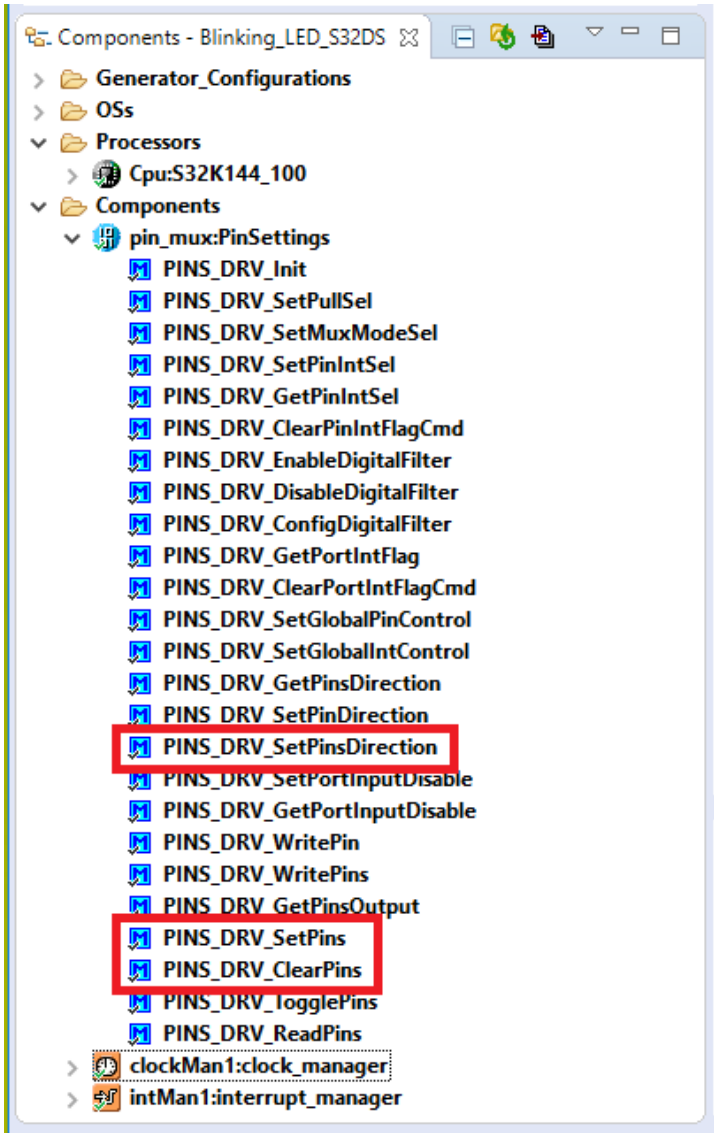
```
/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
#ifdef PEX_RTOS_INIT
  PEX_RTOS_INIT();                  /* Initialization of the selected RTOS. Macro is defined by the RTOS component. */
#endif
/*** End of Processor Expert internal initialization.                    ***/

/* Write your code here */
CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT, g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);
/* For example: for(;;) { } */
                                  1
                                  Press 'F2' for focus
/*** 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.  ***/
```

21. Expand the pin_mux component in the Components Window
22. Drag and drop the **PINS_DRV_Init** function into main, below the clock configuration




```
/*** End of Processor Expert internal initialization.                    ***/

/* Write your code here */
CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT, g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);

/* For example: for(;;) { } */

/*** Don't write any code pass this line, or it will be deleted during code generation. ***/
```

23. Expand the pin_mux:PinSettings component in the Components Window and add the following functions in sequence
24. Drag and drop the **PINS_DRV_SetPinsDirection** function into main immediately after **PINS_DRV_Init**
25. Drag and drop the **PINS_DRV_SetPins** function into main
26. Drag and drop the **PINS_DRV_ClearPins** function into main




```
/* Write your code here */
  CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
          g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
  CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

  PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);

  PINS_DRV_SetPinsDirection();
  PINS_DRV_SetPins();
  PINS_DRV_ClearPins();

/* For example: for(;;) { } */
```

27. For each of the **PINS_DRV** functions, there are 2 arguments, first is always PTD (which is macro defined in SDK), the second is defined as follows:

PINS_DRV_SetPinsDirection: OR-ing of LEDRGB_RED and LEDRGB_GREEN = 1 << 15U | 1 << 16U

PINS_DRV_SetPins: Bit shift of LEDRGB_RED = 1 << 15U

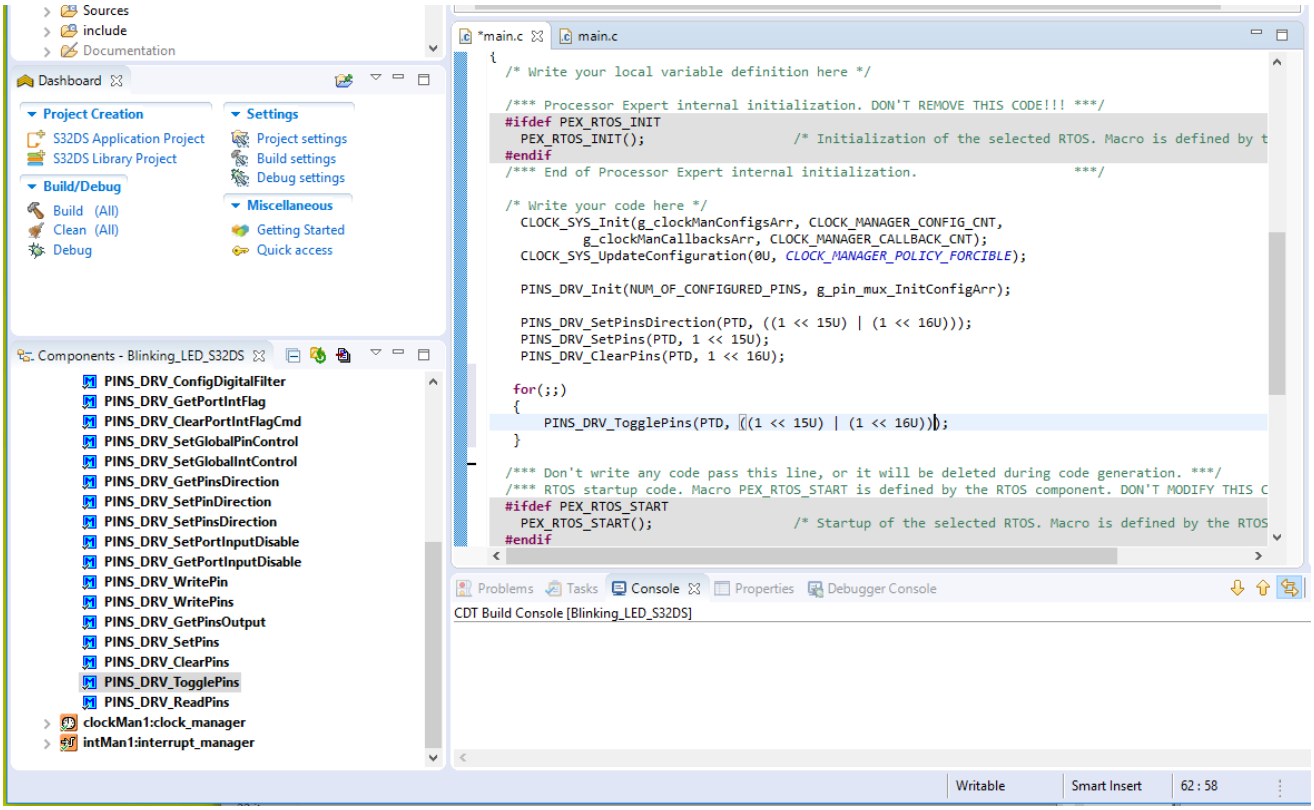
PINS_DRV_ClearPins: Bit shift of LEDRGB_GREEN = 1 << 16U

```
/* Write your code here */
  CLOCK_SYS_Init(g_clockManConfigsArr, CLOCK_MANAGER_CONFIG_CNT,
          g_clockManCallbacksArr, CLOCK_MANAGER_CALLBACK_CNT);
  CLOCK_SYS_UpdateConfiguration(0U, CLOCK_MANAGER_POLICY_FORCIBLE);

  PINS_DRV_Init(NUM_OF_CONFIGURED_PINS, g_pin_mux_InitConfigArr);

  PINS_DRV_SetPinsDirection(PTD, ((1 << 15U) | (1 << 16U)));
  PINS_DRV_SetPins(PTD, 1 << 15U);
  PINS_DRV_ClearPins(PTD, 1 << 16U);

/* For example: for(;;) { } */
```

28. Include an infinite loop after these functions

```
PINS_DRV_SetPinsDirection(PTD, ((1 << 15U) | (1 << 16U)));
PINS_DRV_SetPins(PTD, 1 << 15U);
PINS_DRV_ClearPins(PTD, 1 << 16U);

for(;;)
{
    |
}

/*** Don't write any code pass this line, or it will be delet
```

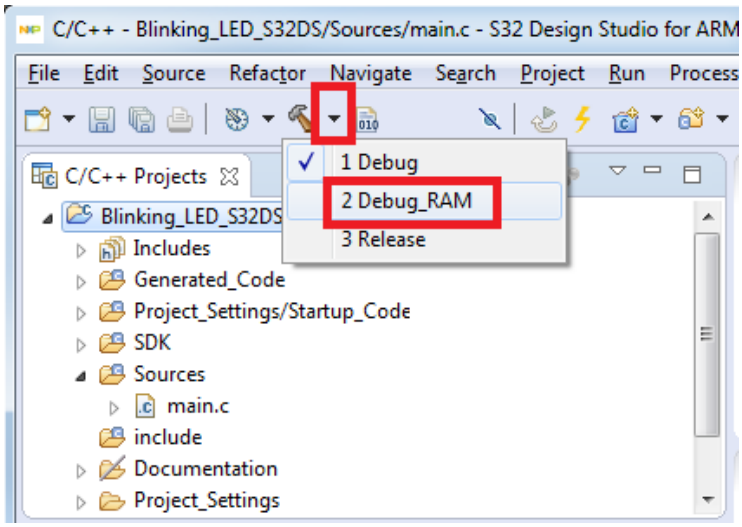
29. Drag and drop the **PINS_DRV_TogglePins** function in to main, and place it inside the 'for' loop.
30. Again, the first argument will be PTD and the second is the same as for PINS_DRV_SetPinsDirection above.



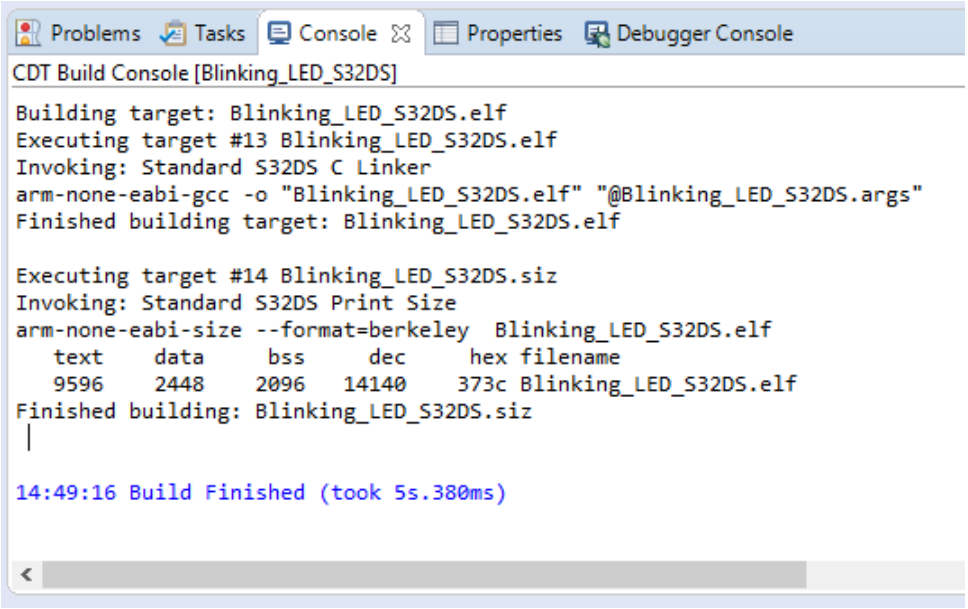
31. Within the 'for' loop, and prior to the **GPIO_HAL_TogglePins** function, add a delay of 720000 cycles int cycles = 720000;
    while(cycles--);

```
PINS_DRV_SetPinsDirection(PTD, ((1 << 15U) | (1 << 16U)));
PINS_DRV_SetPins(PTD, 1 << 15U);
PINS_DRV_ClearPins(PTD, 1 << 16U);

for(;;)
{
    int cycles = 720000;
    while(cycles--);

    PINS_DRV_TogglePins(PTD, ((1 << 15U) | (1 << 16U)));
}

/*** Don't write any code pass this line, or it will be deleted during code generation. ***/
```

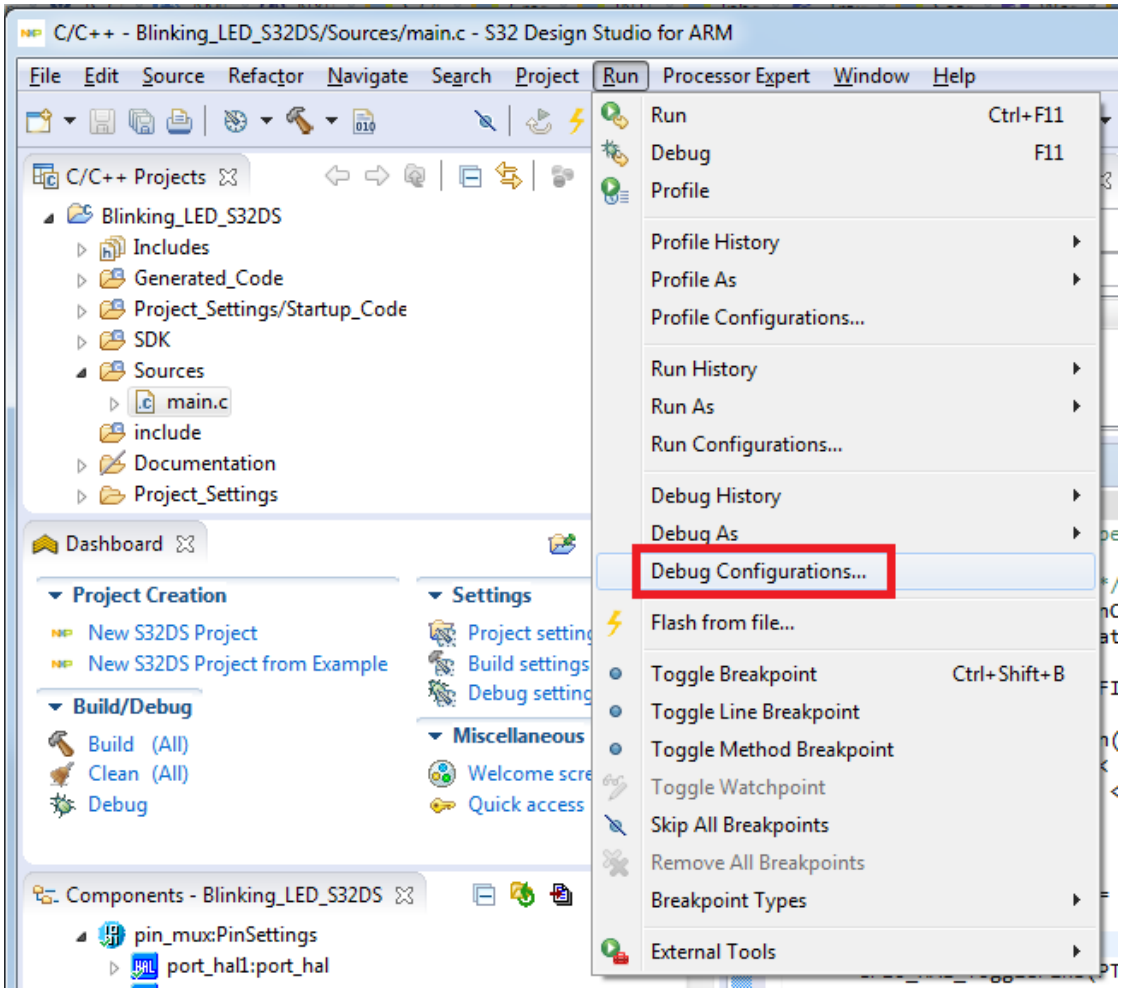
32. Build 'Blinking_LED_S32DS'. Select the project name in 'C/C++ Projects' view and then press 'Debug_RAM'



33. After the build completes, check that there are no errors.

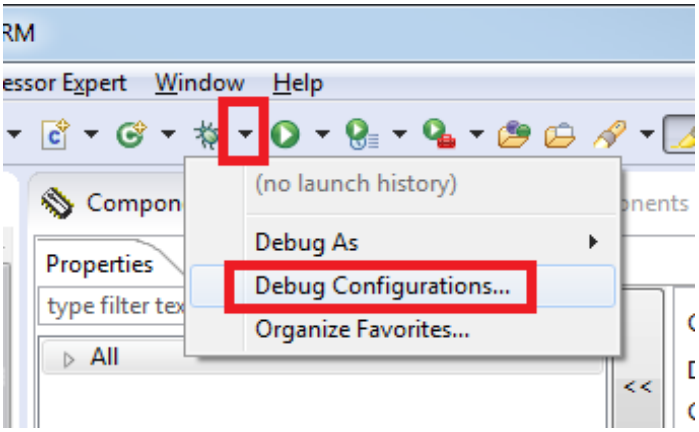


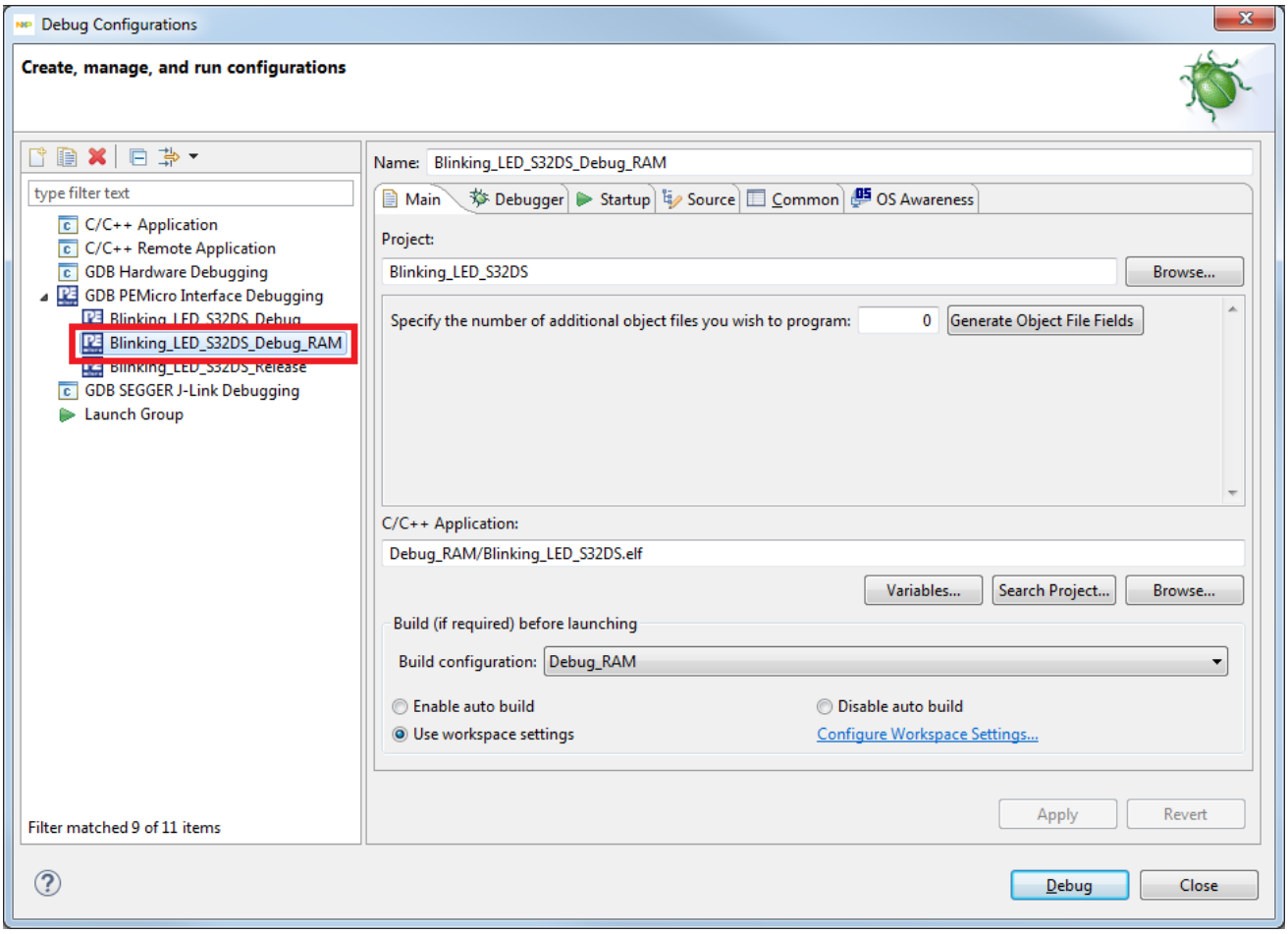
34. Open Debug Configurations and select 'Blinking_LED_S32DS_Debug_RAM'



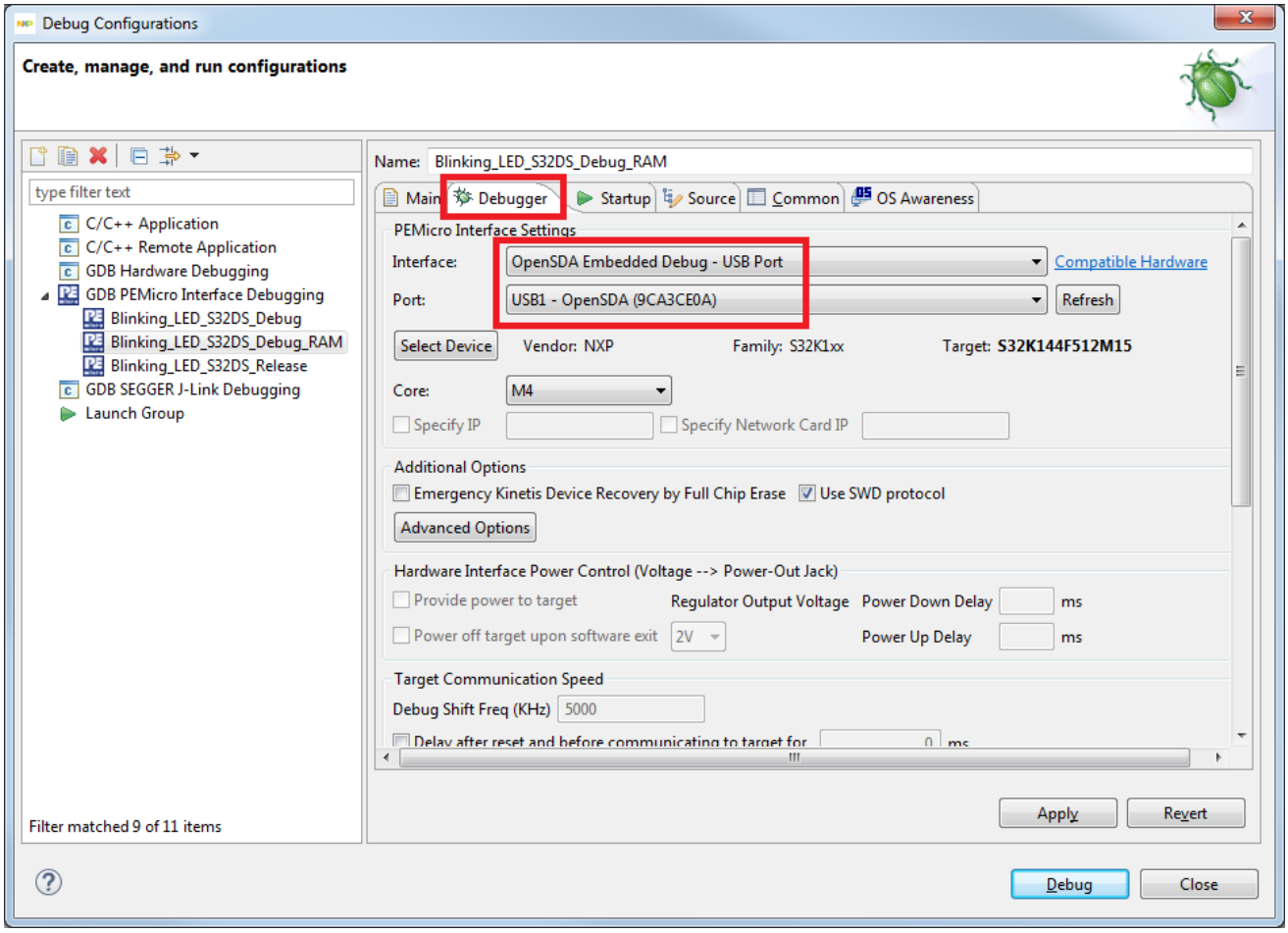
HOWTO: Create the Blinking LED example project using SDK

OR

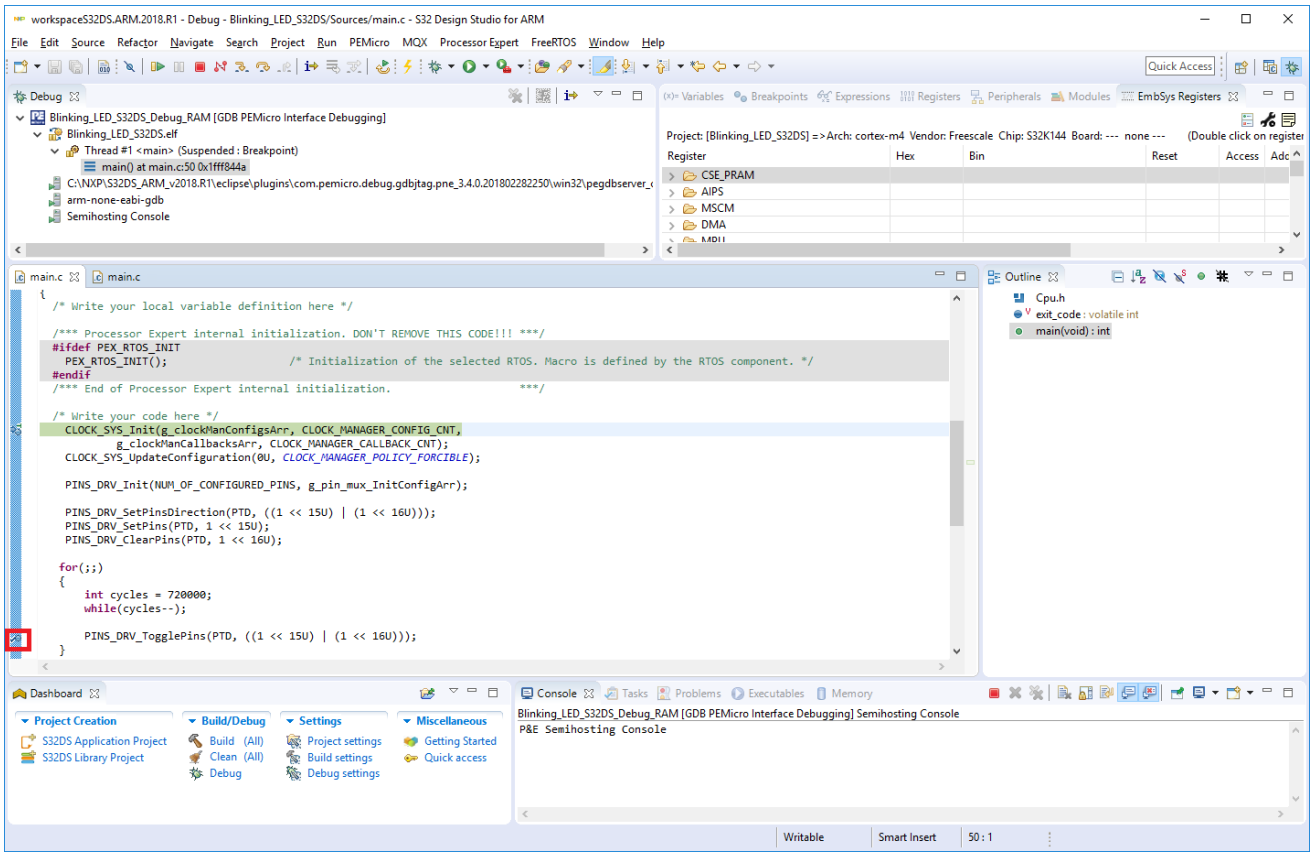




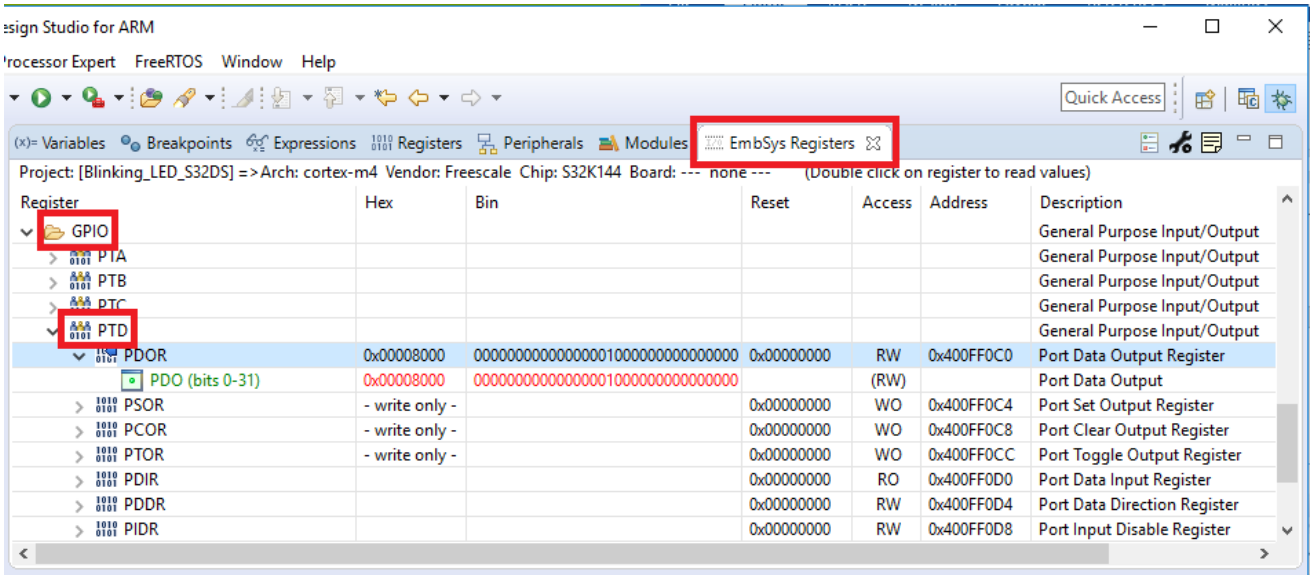
35. Check the Debugger settings and ensure that 'OpenSDA Embedded Debug - USB Port' is selected for interface.
36. Click **Debug**



37. Set breakpoint on PINS_DRV_TogglePins



38. Step through initialization calls
39. To see the output register bits change, go to 'EmbSys Registers' tab and expand 'GPIO', then 'PTD' and 'PDOR'. Double-click on PDOR to enable reading of the values.



40. Click **resume** to advance to the breakpoint, see the LED on board change color
41. Click **resume** again and see LED change to other color