1.Debugging Start
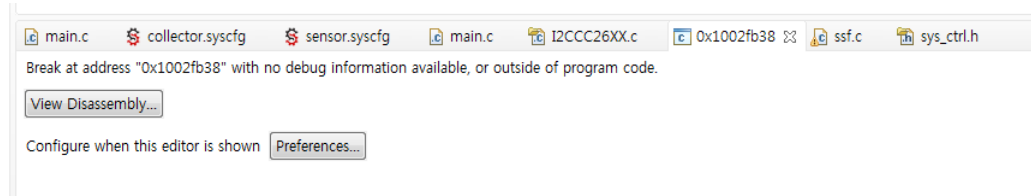
```
459 /*!
460  * @brief     "main()" function - starting point
461  */
462 int main(void)
463 {
464      Task_Params taskParams;
465
466 #ifndef USE_DEFAULT_USER_CFG
467      macUser0Cfg[0].pAssertFP = macHalAssertHandler;
468 #endif
469
470      /*
471       Initialization for board related stuff such as LEDs
472       following TI-RTOS convention
473       */
474      //PIN_init(BoardGpioInitTable);    // Changed by sklee
475      Board_init();
```

Disassembly   Enter location here

```
            main():
 0000b450:  B530            push      {r4, r5, lr}
 467             macUser0Cfg[0].pAssertFP = macHalAssertHandler;
 0000b452:  4C27            ldr       r4, [pc, #0x9c]
 0000b454:  4823            ldr       r0, [pc, #0x8c]
 0000b456:  F1AD0D3C        sub.w     sp, sp, #0x3c
 0000b45a:  3C40            subs      r4, #0x40
 0000b45c:  6660            str       r0, [r4, #0x64]
 475             Board_init();
 0000b45e:  F007FDB3        bl        #0x12fc8
 513             UART_init();
```

2.Break Point

```
523
524      UART_init();
525 /*=======================*/
526 /* I2C Initialization -Added by sklee */
527      I2CCC26XX_init(hI2CHande);
528      //I2C_init();
529      // I2C_Params_init(&pI2CParam);
530
531      // pI2CParam.bitRate = I2C_100kHz;
```

```
 0000b480:  6008            str       r0, [r1]
 524             UART_init();
 0000b482:  F005FB15        bl        #0x10ab0
 527             I2CCC26XX_init(hI2CHande);
 0000b486:  68A0            ldr       r0, [r4, #8]
 0000b488:  F008FC68        bl        #0x13d5c
 534             hI2CHande = I2CCC26XX_open(Board_I2C0, &pI2CParam);   // I2C Open
 0000b48c:  F104050C        add.w     r5, r4, #0xc
 0000b490:  2000            movs      r0, #0
 0000b492:  4629            mov       r1, r5
 0000b494:  F7FFF87C        bl        #0x0500
```

3.Step Into(F5) run

```
573  */
574 void I2CCC26XX_init(I2C_Handle handle)
575 {
576      I2CCC26XX_Object        *object;
577
578      /* Get the pointer to the object */
579      object = handle->object;
580
581      /* Initially the drivers is not open */
582      object->isOpen = false;
583
584 }
585
```

```
  582           object->isOpen = false;
              I2CCC26XX_init():
► 00013d5c:   6840              ldr       r0, [r0, #4]
  00013d5e:   2100              movs      r1, #0
  00013d60:   F88010DC          strb.w    r1, [r0, #0xdc]
  00013d64:   4770              bx        lr
```

4.Continuous Step Into run

Break at address "0x1002fb38" with no debug information available, or outside of program code.

View Disassembly...

Configure when this editor is shown   Preferences...

```
              ti_sysbios_family_arm_m3_Hwi_excHandlerAsm__I:
► 1002fb38:   F01E0F04          tst.w     lr, #4
  1002fb3c:   BF14              ite       ne
  1002fb3e:   F3EF8009          mrs       r0, psp
  1002fb42:   4668              mov       r0, sp
  1002fb44:   4685              mov       sp, r0
```

➜   Run until to Address 1002de1a(step into)

->Step Into run

```
  426 */
  427 xdc_Void Main_excHandler(UInt *excStack, UInt lr)
  428 {
  429     /* User defined function */
► 430     Main_assertHandler(MAIN_ASSERT_HWI_TIRTOS);
  431 }
  432
  433 /*!
```

```
  430           Main_assertHandler(MAIN_ASSERT_HWI_TIRTOS);
              Main_excHandler():
► 00013e56:   2004              movs      r0, #4
  00013e58:   F7FDB842          b.w       Main_assertHandler
```

->Step Into run

```
           Main_assertHandler():
00010ee0:   B51F                push        {r0, r1, r2, r3, r4, lr}
  1545    {
00010ee2:   F88D0000            strb.w      r0, [sp]
  1546        if((pNV != NULL) && (pNV->writeItem != NULL))
00010ee6:   480B                ldr         r0, [pc, #0x2c]
```

->Step Into run

```
1542  Public function defined in ssf.h
1543  */
1544 void Ssf_assertInd(uint8_t reason)
1545 {
1546      if((pNV != NULL) && (pNV->writeItem != NULL))
1547      {
1548          /* Attempt to save reason to read after reset */
1549          (void)pNV->writeItem(nvResetId, 1, &reason);
1550      }
1551 }
1552
```

```
00010ee0:   B51F                push        {r0, r1, r2, r3, r4, lr}
  1545    {
00010ee2:   F88D0000            strb.w      r0, [sp]
  1546        if((pNV != NULL) && (pNV->writeItem != NULL))
00010ee6:   480B                ldr         r0, [pc, #0x2c]
00010ee8:   6800                ldr         r0, [r0]
```

->Step Into run

```
1543   /
1544 void Ssf_assertInd(uint8_t reason)
1545 {
1546      if((pNV != NULL) && (pNV->writeItem != NULL))
1547      {
1548          /* Attempt to save reason to read after reset */
1549          (void)pNV->writeItem(nvResetId, 1, &reason);
1550      }
1551 }
1552
```

```
00010ee2:   F88D0000            strb.w      r0, [sp]
  1546        if((pNV != NULL) && (pNV->writeItem != NULL))
00010ee6:   480B                ldr         r0, [pc, #0x2c]
00010ee8:   6800                ldr         r0, [r0]
00010eea:   B168                cbz         r0, #0x10f08
```

->Step Into run

```
461 __STATIC_INLINE void
462 SysCtrlSystemReset( void )
463 {
464     // Disable CPU interrupts
465     CPUcpsid();
466     // Write reset register
467     HWREGBITW( AON_PMCTL_BASE + AON_PMCTL_O_RESETCTL, AON_PMCTL_RESETCTL_SYSRESET_BITN ) = 1;
468     // Finally, wait until the above write propagates
469     while ( 1 ) {
470         // Do nothing, just wait for the reset (and never return from here)
471     }
472 }
```

```
           $C$L763:
00010f08:   F002FE7C            bl          #0x13c04
  467          HWREGBITW( AON_PMCTL_BASE + AON_PMCTL_O_RESETCTL, AON_PMCTL_RESETCT
00010f0c:   4903                ldr         r1, [pc, #0xc]
```

->Step Into run


```
NOROM_CPUcpsid():
00013c04:    F3EF8010         mrs       r0, primask
00013c08:    B672             cpsid     i
00013c0a:    4770             bx        lr
00013c0c:    2000             movs      r0, #0
```

->Step Into run

```
460 //*************************************************************************
461 __STATIC_INLINE void
462 SysCtrlSystemReset( void )
463 {
464     // Disable CPU interrupts
465     CPUcpsid();
466     // Write reset register
467     HWREGBITW( AON_PMCTL_BASE + AON_PMCTL_O_RESETCTL, AON_PMCTL_RESETCTL_SYSRESET_BITN ) = 1;
468     // Finally, wait until the above write propagates
469     while ( 1 ) {
470         // Do nothing, just wait for the reset (and never return from here)
471     }
472 }
473
```

```
465          CPUcpsid();
             $C$L763:
00010f08:    F002FE7C         bl        #0x13c04
467          HWREGBITW( AON_PMCTL_BASE + AON_PMCTL_O_RESETCTL, AON_PMCTL_RESETCT
00010f0c:    4903             ldr       r1, [pc, #0xc]
00010f0e:    2001             movs      r0, #1
00010f10:    6008             str       r0, [r1]
469          while ( 1 ) {
             $C$L764:
```

->Step Into run


```
459 //
460 //*************************************************************************
461 __STATIC_INLINE void
462 SysCtrlSystemReset( void )
463 {
464     // Disable CPU interrupts
465     CPUcpsid();
466     // Write reset register
467     HWREGBITW( AON_PMCTL_BASE + AON_PMCTL_O_RESETCTL, AON_PMCTL_RESETCTL_SYSRESET_BITN ) = 1;
468     // Finally, wait until the above write propagates
469     while ( 1 ) {
470         // Do nothing, just wait for the reset (and never return from here)
471     }
472 }
473
474 //*************************************************************************
475 //
476 //! \brief Enables reset if OSC clock loss event is asserted.
477 //!
478 //! Clock loss circuit in analog domain must be enabled as well in order to
479 //! actually enable for a clock loss reset to occur
480 //! \ref OSCClockLossEventEnable().
481 //!
482 //! \note This function shall typically not be called because the clock loss
483 //! reset functionality is controlled by the boot code (a factory configuration
484 //! defines whether it is set or not).
485 //!
486 //! \return None
487 //!
488 //! \sa \ref SysCtrlClockLossResetDisable(), \ref OSCClockLossEventEnable()
489 //
```
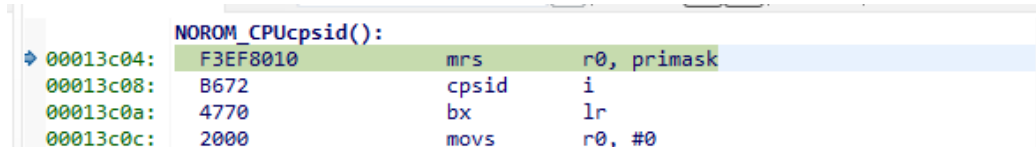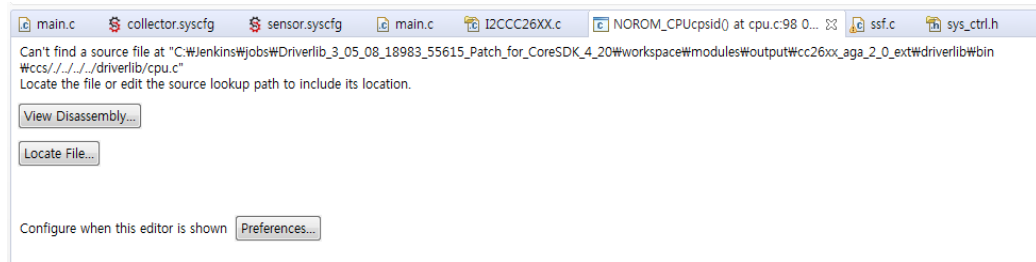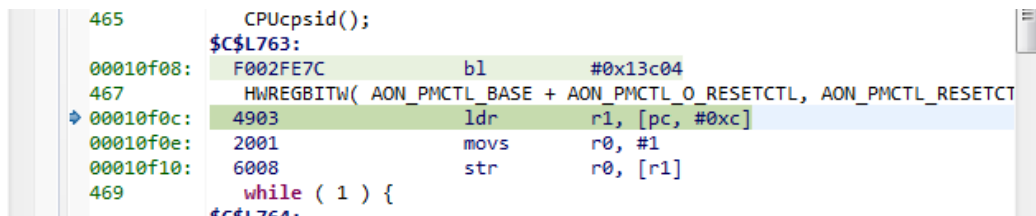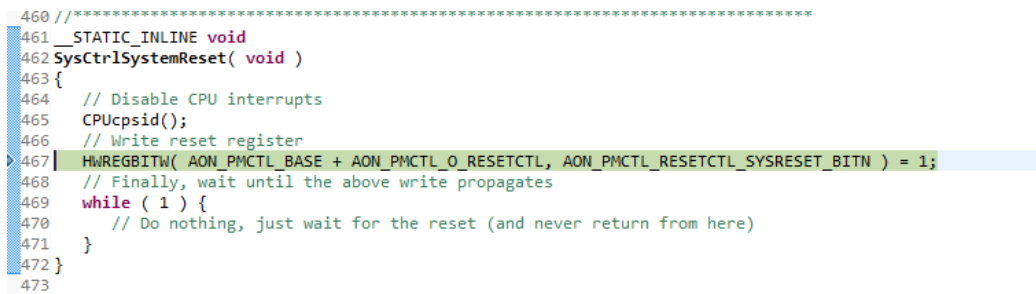
Console

sensor_CC1352P1_LAUNCHXL_tirtos_ccs_syscfg
```
Cortex_M4_0: GEL Output: Memory Map Initialization Complete.
Cortex_M4_0: GEL Output: Board Reset Complete.
Cortex_M4_0: Can't Single Step Target Program: (Error -2134 @ 0x0) Unable to control device execution state. Reset the device, and retry the
Cortex_M4_0: Error: (Error -1170 @ 0x0) Unable to access the DAP. Reset the device, and retry the operation. If error persists, confirm confi
Cortex_M4_0: Error: (Error -1170 @ 0x0) Unable to access the DAP. Reset the device, and retry the operation. If error persists, confirm confi
Cortex_M4_0: Error: (Error -1170 @ 0x0) Unable to access the DAP. Reset the device, and retry the operation. If error persists, confirm confi
Cortex_M4_0: Trouble Halting Target CPU: (Error -2062 @ 0x0) Unable to halt device. Reset the device, and retry the operation. If error persi
Cortex_M4_0: Error: (Error -1170 @ 0x0) Unable to access the DAP. Reset the device, and retry the operation. If error persists, confirm confi
Cortex_M4_0: Trouble Halting Target CPU: (Error -2064 @ 0x0) Unable to read device status. Reset the device, and retry the operation. If erro
Cortex_M4_0: Error: (Error -1170 @ 0x0) Unable to access the DAP. Reset the device, and retry the operation. If error persists, confirm confi
```