

# CC2640 Porting Projects

---

[Bluetooth Low Energy Main Page \(http://processors.wiki.ti.com/index.php/Category:BluetoothLE\)](http://processors.wiki.ti.com/index.php/Category:BluetoothLE)

This wiki page explains necessary steps to port projects from an older version of the TI *Bluetooth<sup>(R)</sup>* low energy software stack (BLE-Stack) v2.x SDK installer to a newer version. As always, it is recommended to make a backup copy of all changes prior to porting your project. In addition to making a backup of your project, TI recommends making a backup (zip) of the BLE-Stack SDK, typically installed to C:\ti\simplelink\ble\_cc26xx\_2\_ox\_oy\_nnnnn, prior to developing your Bluetooth Smart application.

The procedures listed below assume the developer is familiar with the IDE and build environment concepts described in the CC2640 BLE SW Developer's Guide (SWRU393 (<http://www.ti.com/lit/pdf/swru393>)). A copy of the Developer's Guide can be found in the Documents folder of the BLE-Stack SDK installer.

## Contents

---

### Porting from BLEv2.2.1 to BLEv3.x

#### LE Secure Connections Recommended Parameters

#### Porting BLEv2.2.1 Projects to BLEv2.2.2

- Porting Steps

#### Porting BLEv2.2.0 Projects to BLEv2.2.1

- Porting Steps

- Optimizing Flash Memory

- Known Issues / Workarounds for BLEv2.2.1

#### Porting BLEv2.1.1 Projects to BLEv2.2

- BLE Stack changes

  - icall\_api.c

  - peripheral.c

  - central.c

  - main.c

- TI-RTOS changes

  - Display Driver

  - Porting BLE Bridge Project to LaunchPad

- Known Issues / Workarounds for BLEv2.2.0

#### Porting BLEv2.1.0 Projects to BLEv2.1.1

- IAR & CCS Project Porting Directions

#### Porting BLEv2.0.0 Projects to BLEv2.1.0

- IAR Project Porting Directions

- CCSProject Porting Directions
  - Import the Old Project
  - Application Project Modifications
  - Stack Project Modifications
  - Compiling Steps
- API Changes

### Porting Drivers From TI-RTOS 2.20 or higher To TI-RTOS 2.18

- Steps For Importing Drivers
  - Continue For CCS User
  - Continue For IAR User

## Porting from BLEv2.2.1 to BLEv3.x

See the Porting chapter of the software developer's guide.

## LE Secure Connections Recommended Parameters

The underlying ECC software used to generate the Public/Private Key pair and the DH Key check will lock the BLE-Stack task for approximately 160ms. This lock will occur twice during secure connections pairing. When using a LE Secure Connections pairing method, it is recommend to configure a supervision timeout of at least **165ms** to prevent the link from dropping during ECDH key operations. Additionally the application should be prepared to be preempted/blocked during this time.

Note: This only affects stacks using the LE Secure Connections feature

## Porting BLEv2.2.1 Projects to BLEv2.2.2

This section will describe how to update from BLEv2.2.1 to BLE v2.2.2. Application source code previously developed or updated to BLE 2.2.1 can be ported to BLE 2.2.2 with minimal effort due to the fact that no BLE APIs were modified between these releases. Notable differences include:

- Updated to use TI-RTOS SDK 2.21.01.08 (included with BLE-Stack SDK installer). It's recommended to use the updated board files for this TI-RTOS version
- Updated BLE libraries and supporting stack source code. This includes GAPBondMgr, OAD, the ECCROMCC26XX driver and ble\_user\_config.c which as updated RF overrides and ROM patch parameters. Refer to BLE-Stack release notes for a full list of changes
- Example application source files (e.g., simple\_peripheral.c) updated to handle and assert upon HCI\_BLE\_HARDWARE\_ERROR\_EVENT\_CODE in \*\_processStackMsg handler. This is optional.

## Porting Steps

---

1. Copy your project files, source files, and any modified BLE Stack source files from the 2.2.1 installer to the 2.2.2

installer, ensuring to merge any changes with common (non-application) SDK files.

2. Verify your application's .customargvars (IAR) / .cproject or .projectspec (CCS) point to resources in the TI-RTOS 2.21.01.08 & XDCTools default installation path. These can be inspected by importing an unmodified sample application from the BLE 2.2.2 SDK. These are the same parameters, but different values, as listed below for BLE 2.2.1.
3. Open/import your project in IAR 7.80.4 or CCS v7.4. Note that CCS projects now use the TI ARM Compiler 16.9.4.LTS and the RTSC option needs to specify the new TI-RTOS version.

## Porting BLEv2.2.0 Projects to BLEv2.2.1

This section will describe one way port a project from BLEv2.2.0 to BLE v2.2.1. It assumes that your project is constructed similar to the default stack projects. That is, the project files exist under \$INSTALL\$/examples/cc2650em/ and source files exist under the respective folder in \$INSTALL\$/src/. It also assumes that you are using the default .customargvars (IAR) / .cproject or .projectspec (CCS) which use the relative location of the IAR / CCS project files to point to BLE Stack components. If your project exists outside of the standard directory structure of the source installer and uses absolute paths for these variables, you'll need to manually modify these to point to the respective directories in the 2.2.1 installer. In this case, the .cfg file in the application project will also need to be manually modified to point to the 2.2.1 .cfg (\$INSTALL\$/src/common/cc26xx/kernel/cc2640/config/cc2640.cfg)

### Porting Steps

1. Copy your project files, source files, and any modified BLE Stack source files from the 2.2.0 installer to the 2.2.1 installer, ensuring to merge any changes with common files.
2. Since many of the environment variables defined in the .customargvars / .project or .projectspec use the relative location of the project file, these should successfully be updated from step 1. However, the paths relating to the new RTOS need to be manually updated. The following should be used based on the type of project:

- IAR .custom\_argvars:

```

CC26xx-TI-RTOS
TI_RTOS_DRIVERS_BASE = C:\ti\tirtos_cc13xx_cc26xx_2_20_01_08\products\tidrivrs_cc13xx_cc26xx_2_20_01_10\packages
BIOS_BASE = C:\ti\tirtos_cc13xx_cc26xx_2_20_01_08\products\bios_6_46_01_38\packages
XDCPATH = C:\ti\tirtos_cc13xx_cc26xx_2_20_01_08\products\tidrivrs_cc13xx_cc26xx_2_20_01_10\packages;C:\ti\tirtos_cc13xx_cc26xx_2_20_01_08\products\bios_6_46_01_38\packages
CC26XXWARE = C:\ti\tirtos_cc13xx_cc26xx_2_20_01_08\products\cc26xxware_2_24_02_17393
XDCROOT = C:\ti\xdctools_3_32_00_06_core

```

- CCS .project\_spec:

```

<!-- Env Vars -->
<pathVariable name="TI_RTOS_DRIVERS_BASE" path="C:/ti/tirtos_cc13xx_cc26xx_2_20_01_08/products/tidrivrs_cc13xx_cc26xx_2_20_01_10/packages" scope="project"></pathVariable>
<pathVariable name="TI_BLE_SDK_BASE" path="C:/ti/simplelink/ble_sdk_2_02_01_18" scope="project"></pathVariable>
<pathVariable name="SRC_EX" path="${TI_BLE_SDK_BASE}/src" scope="project"></pathVariable>
<pathVariable name="SRC_COMMON" path="${TI_BLE_SDK_BASE}/src/components" scope="project"></pathVariable>
<pathVariable name="SRC_BLE_CORE" path="${TI_BLE_SDK_BASE}/src/" scope="project"></pathVariable>
<pathVariable name="ROM" path="${TI_BLE_SDK_BASE}/src/rom" scope="project"></pathVariable>
<pathVariable name="TOOLS_BLE" path="${TI_BLE_SDK_BASE}/tools" scope="project"></pathVariable>
<pathVariable name="CC26XXWARE" path="/C:/ti/tirtos_cc13xx_cc26xx_2_20_01_08/products/cc26xxware_2_24_02_17393" scope="project"></pathVariable>
<pathVariable name="PROJECT_IMPORT_LOC" path="." scope="project"></pathVariable>

```

- CCS: .project:

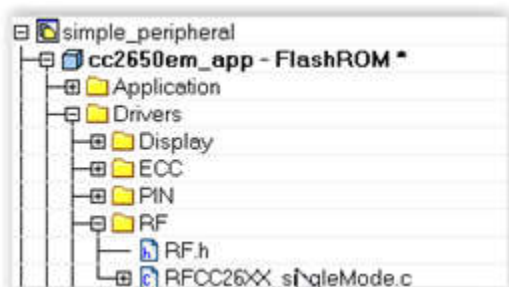
```

</linkedResources>
<variableList>
  <variable>
    <name>CC26XXWARE</name>
    <value>file:/C:/ti/tirtos_cc13xx_cc26xx_2_20_01_08/products/cc26xxware_2_24_02_17393</value>
  </variable>
  <variable>
    <name>ORG_PROJ_DIR</name>
    <value>${PARENT-2-PROJECT_LOC}/iar/stack</value>
  </variable>
  <variable>
    <name>ROM</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/src/rom</value>
  </variable>
  <variable>
    <name>SRC_BLE_CORE</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/src</value>
  </variable>
  <variable>
    <name>SRC_COMMON</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/src/components</value>
  </variable>
  <variable>
    <name>SRC_EX</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/src</value>
  </variable>
  <variable>
    <name>TI_RTOS_BOARD_BASE</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/src/components</value>
  </variable>
  <variable>
    <name>TI_RTOS_DRIVERS_BASE</name>
    <value>file:/C:/ti/tirtos_cc13xx_cc26xx_2_20_01_08/products/tidrivrs_cc13xx_cc26xx_2_20_01_10/packages</value>
  </variable>
  <variable>
    <name>TOOLS_BLE</name>
    <value>${PARENT-5-ORG_PROJ_DIR}/tools</value>
  </variable>
</variableList>

```

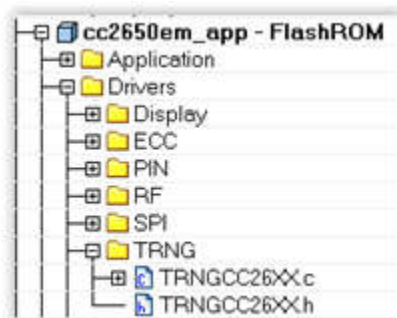
3. Open in the new IAR (7.70.2) / CCS (6.2.0 Build 50 or later)

4. In the application project, under Drivers --> RF...remove RF.c and add \$TI\_RTOS\_DRIVERS\_BASE\$\ti\drivers\rf\RFCC26XX\_singleMode.c



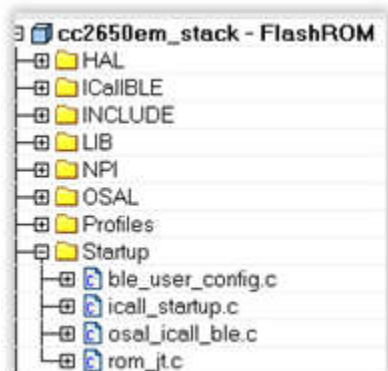
5. In the application project, under Drivers, add a TRNG folder and add the following files:

- \$SRC\_COMMON\$\hal\src\target\\_common\TRNGCC26XX.c
- \$SRC\_COMMON\$\hal\src\target\\_common\TRNGCC26XX.h

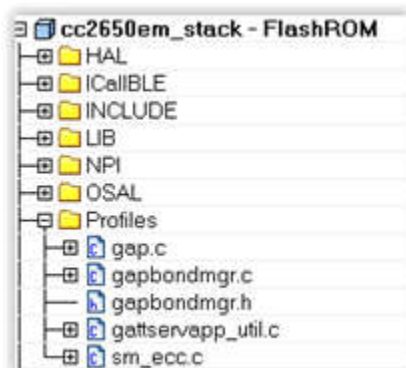


6. In the stack project, add `$SRC_BLE_CORE$/rom` to preprocessor search path

7. Remove `common_rom_init.c` and `rom_init.c` from the Startup folder and add `$ROM$/rom_jt.c`



8. If building with the Secure Connections feature defined (`-DBLE_V42_FEATURES=SECURE_CONNS_CFG` or similar), add `$SRC_EX$/host/sm_ecc.c` to the Profiles folder:



## Optimizing Flash Memory

The following tips for maximizing the amount of flash memory to the application is a supplement to Section 10.4 *Optimizing Bluetooth low energy Stack Memory Usage* from the BLE SW Developer's Guide (docID SWRU393 rev D).

- When not enabling Pairing/Bonding support with the GAP Bond Manager (GAP\_BOND\_MGR) in the stack build\_config.opt, set the following Predefined Symbols in the Stack project:
  - NO\_BLE\_SECURITY
    - If OSAL\_SNV=x is defined, set OSAL\_SNV=0, otherwise, define NO\_OSAL\_SNV. This will remove allocation of flash page(s) for SNV storage
- The use of the BT4.1 L2CAP\_COC\_CFG is not required for GATT and has limited or no smartphone compatibility. Enabling a BLE\_V41\_FEATURES=V41\_CTRL\_CFG is the minimum configuration needed for Peripheral + Central configurations or when multiple slave connections are required.
- If the GAP\_BOND\_MGR is not required or if the application does not need to send a Service Changed Indication, support for this feature can be disabled to save flash memory space. Service Changed requires use of L2CAP stack build\_config.opt feature. Set the following in the Stack Predefined Symbols to disable support for Service

Changed: GATT\_NO\_SERVICE\_CHANGED

## Known Issues / Workarounds for BLEv2.2.1

- CCS cannot find target config file for simple\_peripheral\_cc2650em\_app project. Open your project properties and select "General". Check the box marked "Manage the project's target-configuration automatically" and change "Connection" to Texas Instruments XDS100V3 USB Debug.
- Host Test does not automatically respond to Connection Parameter Update Requests and instead allows the application to specify the response parameters. Responses must manually sent within 40 seconds of reception of the GAP\_LinkParamUpdateRequest event 0x0612 using HCI\_LERemoteConnectionParameterRequestReply or HCI\_LERemoteConnectionParameterRequestNegativeReply. See log below for an example of handling this request using BTool:

```
-----
[22] : <Rx> - 01:30:27.499
-Type           : 0x04 (Event)
-EventCode      : 0x00FF (HCI_LE_ExtEvent)
-Data Length    : 0x0D (13) bytes(s)
Event           : 0x0612 (1554) (GAP_LinkParamUpdateRequest)
Status          : 0x00 (0) (Success)
ConnHandle      : 0x0000 (0)
MinConnInt     : 0x0006 (6)
MaxConnInt     : 0x0006 (6)
ConnLatency    : 0x0000 (0)
ConnTimeout    : 0x07D0 (2000)
Dump (Rx) :
0000:04 FF 0D 12 06 00 00 00 06 00 06 00 00 00 D0 07 .....
-----
[23] : <Tx> - 01:30:33.803
-Type           : 0x01 (Command)
-OpCode        : 0x2021 (HCI_LERemoteConnectionParameterRequestNegativeReply)
-Data Length    : 0x03 (3) byte(s)
Handle         : 0x0000 (0)
Status         : 0x3B (59) (Unacceptable Connection Interval)
Dump (Tx) :
0000:01 21 20 03 00 00 3B .! ...;
-----
[24] : <Rx> - 01:30:33.827
-Type           : 0x04 (Event)
-EventCode      : 0x000E (HCI_CommandCompleteEvent)
-Data Length    : 0x06 (6) bytes(s)
Packets        : 0x01 (1)
OpCode         : 0x2021 (HCI_LERemoteConnectionParameterRequestNegativeReply)
Status         : 0x00 (0) (Success)
Handle         : 0x0000 (0)
Dump (Rx) :
0000:04 0E 06 01 21 20 00 00 00 .! ...
-----
```

- Advertising (ADV) may stop when performing extended periods (i.e., more than 1 hour) of continuous Advertising due to an anomaly in the TI-RTOS RF Driver used by BLE-Stack v2.2.0 and v2.2.1. The workaround is to stop and restart ADV on a periodic basis using a TI-RTOS Clock instance. See attached [File:simple\\_peripheral\\_ble\\_221\\_advRestart.zip](#) with modifications denoted by "ADV-Restart" code comments. Please use Rev 1.1 of the patch. The suggested restart period is 30 minutes but can be adjusted by changing the ADV\_RESTART\_EVT\_PERIOD #define. This file is compatible with BLE-Stack v2.2.1 and the same procedure can be applied to other sample applications in the SDK. **Note: TI strongly recommends using BLE 2.2.2 which includes a fix for this issue.**
- The CC2650 Remote Control (CC2650RC) "hid\_adv\_remote\_cc2650rc\_app" application project in CCS will fail to

link due to incorrect TI-RTOS project configuration. Open your project properties, General -> RTSC tab, select 2.20.1.08 under "TI-RTOS for CC13XX and CC26XX". Rebuild the project.

- Error in stack project build: symbol "ECC\_initialize" redeclared with incompatible type. This error will occur when projects using Secure Connections are built in CCS with a TI ARM Compiler other than TI ARM Compiler v5.2.6. To install TI ARM Compiler v5.2.6 if it is not already installed please refer to section 2.6.3.2 of the TI BLE Software Developer's Guide.
- The Sleep Clock Accuracy (SCA) setting for CC2650 LaunchPad sample applications does not match the actual 32 kHz crystal accuracy for this development kit. This may result in an unexpected connection drop, especially with longer connection intervals. To set the correct SCA, call the following API after ICall\_registerApp in the Application's initialization function: HCI\_EXT\_SetSCACmd(100);

# Porting BLEv2.1.1 Projects to BLEv2.2

## BLE Stack changes

---

BLE Stack 2.2 is a major release that incorporates several bug fixes as well as new features from the BLE 2.2 specifications. See Software Developer's Guide(SDG)v2.2 for more details. The new features include:

- LE Secure Connections
- LE Data Length Extension
- LE Privacy 1.2

New APIs associated with their respective APIs is described in the SDG. The following APIs have changed from 2.1 to 2.2:

### icall\_api.c

```
void GAPBondMgr_SlaveReqSecurity(uint16 connHandle, uint8 authReq)
```

```
bStatus_t GAP_Bond(uint16 connectionHandle, uint8 authenticated,  
                  uint8 secureConnections, smSecurityInfo_t *pParams,  
                  uint8 startEncryption)
```

### peripheral.c

```
// Connection parameter update parameters.  
static gapRole_updateConnParams_t gapRole_updateConnParams =  
{  
    .paramUpdateEnable = FALSE,  
    .minConnInterval = DEFAULT_MIN_CONN_INTERVAL,  
    .maxConnInterval = DEFAULT_MAX_CONN_INTERVAL,  
    .slaveLatency = MIN_SLAVE_LATENCY,  
    .timeoutMultiplier = DEFAULT_TIMEOUT_MULTIPLIER  
};  
  
static bStatus_t gapRole_startConnUpdate(uint8_t handleFailure,  
                                         gapRole_updateConnParams_t *pConnParams);
```

### central.c

```
uint8_t authReq = ((gapSlaveSecurityReqEvent_t *)pMsg)->authReq;  
GAPBondMgr_SlaveReqSecurity(connHandle, authReq);
```

Most of these APIs changes are due to additional parameters that have been added to support new features.

Other than the new features, the major change in BLE SDK 2.2 is the new directory structure. In addition, a new naming convention is used to a common naming standard for file and folder names. See Section 2.4 Directory Structure in the Software Developer's Guide v2.2 for more information.

Note: To port from a release prior to BLE v2.1.1, we recommend following the porting guides incrementally up to this version.

To port from a project based on 2.1 to 2.2, we recommend manually adding the changes into the 2.2 directory files due to the extensive amount of changes incorporated in the 2.2 release.

### main.c

Copying the reset vectors was removed from main.c since it is now configured in the RTOS configuration file. The options are set in the Pre-build command option: "--cfgArgs NO\_ROM=1,OAD\_IMG\_E=1" to make RTOS automatically copy the reset vectors into RAM before main.c for OAD projects.

## TI-RTOS changes

---

There is significant amount of change from the TI-RTOS used in BLE SDK v2.1 to v2.2. Some of the RTOS changes are detailed in this guide: [http://processors.wiki.ti.com/index.php/TI-RTOS\\_Migration\\_2\\_15](http://processors.wiki.ti.com/index.php/TI-RTOS_Migration_2_15)

See the following watchdog consideration: [https://e2e.ti.com/support/wireless\\_connectivity/bluetooth\\_low\\_energy/f/538/p/543091/1984008#1984008](https://e2e.ti.com/support/wireless_connectivity/bluetooth_low_energy/f/538/p/543091/1984008#1984008)

### Display Driver

A new display driver has been added. For example, to add the UART display to the Launchpad, define/modify these in preprocessor settings:

```
xDisplay_DISABLE_ALL  
xBOARD_DISPLAY_EXCLUDE_UART  
BOARD_DISPLAY_EXCLUDE_LCD
```

and open the display to use UART in simple\_peripheral.c:

```
dispHandle = Display_open(Display_Type_UART, NULL);
```

To not use the display at all, define

```
Display_DISABLE_ALL
```



The `Display_print()` macros can be redefined in the project to output to another debug module. For example, in the SPP BLE project, it is routed to the UART via the `DEBUG` macro:

```
# define Display_print0(handle, line, col, fmt) DEBUG(fmt); \
    DEBUG_NEWLINE()
# define Display_print1(handle, line, col, fmt, a0) DEBUG(fmt); \
    DEBUG_NEWLINE()
```

## Porting BLE Bridge Project to LaunchPad

This section shows some examples of changes needed to port an existing project from v2.1 to v2.2.

The following files need to be added:

- Add SDI folder in the project
- Exclude Simple GATT Profile and add Serial Port Profile in the project

Note: Make sure to disable all displays since Launchpad does not include a display and the SDI uses the UART for sending data.

The following modifications has to be made:

- Change definition of `MRDY_PIN` and `SRDY_PIN` to match the board file in `sdi_config.h`:

```
# elif defined(SDI_USE_UART)
#     define MRDY_PIN Board_BUTTON0
#     define SRDY_PIN Board_BUTTON1
# endif
```

- Add in new power drivers:

```
#include <ti/drivers/Power.h>
#include <ti/drivers/power/PowerCC26XX.h>
```

- Change definition names of power constraints:

```
// set constraints for Standby and idle mode
Power_setConstraint(PowerCC26XX_SB_DISALLOW);
Power_setConstraint(PowerCC26XX_IDLE_PD_DISALLOW);
```

- Change definition of UART struct name:

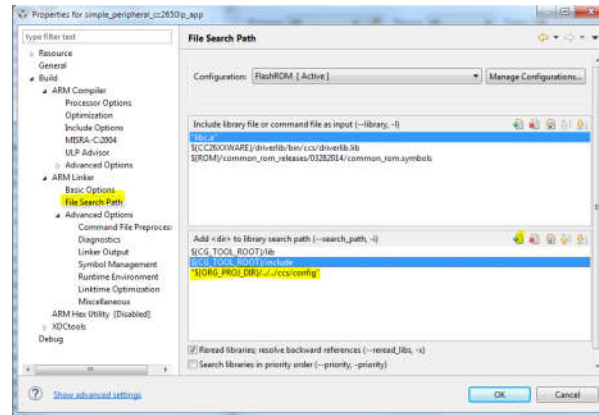
```
if (!UARTCharsAvail(((UARTCC26XX_HWAttrsV1 const *) (uartHandle->hwAttrs))->baseAddr))
```

## Known Issues / Workarounds for BLEv2.2.0

- Building protocol stack projects in CCS with TI Compiler v15.12.2.LTS results in linker error / build failure. This is fixed in the v15.12.3.LTS compiler update or by updating the project to use the TI Compiler v5.2.7. Details on how to update to TI Compiler v5.2.7 are provided in the SW Developer's Guide (SWRU393).
- SensorTag Application project fails to build on CCS when the project is copied to the Workspace. Workaround: Do

not copy the project to the Workspace when importing the project -or- update the project with the procedure from this E2E post ([https://e2e.ti.com/support/wireless\\_connectivity/bluetooth\\_low\\_energy/f/538/t/525660](https://e2e.ti.com/support/wireless_connectivity/bluetooth_low_energy/f/538/t/525660)).

- Link order in CCS v6.1.x projects may be incorrect which may result in an incorrectly linked Application project. This may cause undefined behaviour at run time. To workaroud this condition, the following procedure can be applied to the Application project:
  - In the Application project, expand the TOOLS folder, right-click on 'ccs\_linker\_define.cmd' and select Exclude from Build
  - Add linker search path `${ORG_PROJ_DIR}/.././ccs/config` to the Project Properties -> ARM Linker -> File Search Path as shown below



- Open the linker file, typically 'cc26xx\_app.cmd', and add the line "-l ccs\_linker\_defines.cmd" (no quotes, lower case dash L) near the top of the file but before the MEMORY define. See below for reference.

```

40 /* Retain interrupt vector table variable */
50 --retain_g_vectors
51 /* Override default entry point.
52 --entry_point ResetISR
53 /* Suppress warnings and errors:
54 /* - 10053: Warning about entry point not being _v_start
55 /* - 10011, 10012: 8-byte alignment errors, Observed when linking in object
56 /* Files compiled using tool chain compiler)
57 --diag_suppress=1000,1001,10012
58
59 -l ccs_linker_defines.cmd

```

- SNV data may be lost following reset after **first time power up** when using OSAL\_SNV=1 (default setting for stack projects). To apply the fix, edit nvocop.c in ble\_sdk\_2\_02\_00\_31\src\components\services\src\nv\cc26xx and add {FF}; to the as shown in this post ([https://e2e.ti.com/support/wireless\\_connectivity/bluetooth\\_low\\_energy/f/538/p/530696/1942788#1942788](https://e2e.ti.com/support/wireless_connectivity/bluetooth_low_energy/f/538/p/530696/1942788#1942788)) such that the NV\_FLASH array is defined as follows:

```
const uint8 NV_FLASH[FLASH_PAGE_SIZE] = {0xFF};
```

- Potential memory leak caused by using non-atomic Queue APIs that are not thread-safe in util.c. To apply the fix, edit util.c found in ble\_sdk\_2\_02\_00\_31\src\common\cc26xx as shown in this post ([https://e2e.ti.com/support/wireless\\_connectivity/bluetooth\\_low\\_energy/f/538/p/544221/1994563#1994563](https://e2e.ti.com/support/wireless_connectivity/bluetooth_low_energy/f/538/p/544221/1994563#1994563)).
- Using "#pragma FUNCTION\_OPTIONS(<func name>, "opt\_level=X" ) in CCS to selectively deoptimize functions for debugging purposes can result in non functional application code. The work around for this is to set the whole project's optimization level to the desired level. Note, if additional problems appear, upgrade to newer TI compilers for debugging purposes. TI does not recommend releasing production code without optimizations.

## Porting BLEv2.1.0 Projects to BLEv2.1.1

The BLE-Stack v2.1.1 release is a maintenance release to the existing v2.1.0 release. Most v2.1.1 changes are incorporated in library form, therefore porting from v2.1.0 uses a simplified procedure. For porting earlier v2.0 projects, it is recommended to first port to v2.1.0, then follow the below instructions to incorporate the v2.1.1 changes. For a detailed list of the changes in v2.1.1, please refer to the release notes included in the BLE-Stack v2.1.1 SDK installer.

# IAR & CCS Project Porting Directions

---

1. Move (or Copy) the project files from `$2.1.0_INSTALL$\Projects\ble\PROJECT$` to `$INSTALL\Projects\ble\PROJECT$` where `$2.1.0_INSTALL$` is the top level installation directory of the v2.1.0 stack, `$INSTALL$` is the top level installation directory of the 2.1.1 stack, and `$PROJECT$` is your project folder.
2. When porting from v2.1.0, the only source file required to be merged to a v2.1.1 based project is `$INSTALL$\Projects\ble\ICall\Application\bleUserConfig.c`. It can be accessed from the IDE in the Application project under the ICallBLE virtual folder. Under normal conditions, this file is not modified during project development. However, if you have made changes to `bleUserConfig.c`, merge your changes to the v2.1.1 `bleUserConfig.c` version.
  - 3a. For IAR: Open the updated v2.1.1 project workspace from `$PROJECT$\CC26xx\IAR`.
  - 3b. For CCS: Import the Application and Stack projects from `$PROJECT$\CC26xx\CCS` into your workspace using Project-->Import CCS Projects.
4. Rebuild & save your project.

## Porting BLEv2.0.0 Projects to BLEv2.1.0

### IAR Project Porting Directions

---

1. Move (or Copy) the project files from `$2.0.0_INSTALL$\Projects\ble\PROJECT$` to `$INSTALL\Projects\ble\PROJECT$` where `$2.0.0_INSTALL$` is the top level installation directory of the 2.0.0 stack, `$INSTALL$` is the top level installation directory of the 2.1.0 stack, and `$PROJECT$` is your project folder.
2. If you modified any files from anywhere outside of the application folder, such as in the profiles at `$2.1.0_INSTALL$\Projects\Profiles`, you will need to merge your changes with the new 2.1.0 version.
3. Open your project (now in the 2.1.0 folder) with IAR  $\geq 7.40.2$
4. When prompted, choose yes to convert for use with new IAR version.
5. The device information profile path has changed. If this is part of your project, remove the old file (`devinfoservice.c`) from the project (it will be in the PROFILES group). Then add the file from the correct path (`$INSTALL\Projects\ble\Profiles\DevInfo\CC26xx\devinfoservice.c`)
6. There are several files in the `$PROJECT$\Config` folder which need to be updated for 2.1.0. The easiest way to do this is to delete this folder and copy the Config folder from the SimpleBLEPeripheral sample project (`$INSTALL$\Projects\ble\SimpleBLEPeripheral\CC26xx\IAR\Config`) for use. Then, if there were any modifications to any of these files in the original 2.0.0 project, replicate the modifications here.
7. Similarly, the stack configuration file needs to be updated. The easiest way to do this is to replace the `$PROJECT$\Stack\CC2640\buildConfig.opt` file with `$INSTALL\Projects\ble\SimpleBLEPeripheral\CC26xx\IAR\Stack\CC2640\buildConfig.opt`. Then make any desired modifications.
8. The custom argument variables must be updated to use the new RTOS. The easiest way to do this is to replace the 2.0.0 `.custom_argvars` file in `$PROJECT$` with the `.custom_argvars` from the simpleBLEPeripheral project: `$INSTALL\Projects\ble\SimpleBLEPeripheral\CC26xx\IAR\SimpleBLEPeripheral.custom_argvars`). Close and re-open the IAR workspace after changing these variables.

9. Delete the `$PROJECT$\Application\CC2640\configPkg` and `$PROJECT$\CC26xx\IAR\Config\src` folders to remove any remnants of the old RTOS.
10. Compile and save your project.

## CCSProject Porting Directions

---

### Import the Old Project

1. Move (or Copy) the project files from `$2.0.0_INSTALL$\Projects\ble\PROJECT$` to `$INSTALL\Projects\ble\PROJECT$` where `$2.0.0_INSTALL$` is the top level installation directory of the 2.0.0 stack, `$INSTALL$` is the top level installation directory of the 2.1.0 stack, and `PROJECT$` is your project folder.
2. If you modified any files from anywhere outside of the application folder, such as in the profiles at `$2.1.0_INSTALL$\Projects\Profiles`, you will need to merge your changes with the new 2.1.0 version.
3. Using CCS 6.1.0, import the application and stack projects from `PROJECT$\CC26xx\CCS` into your workspace using Project-->Import CCS Projects. The relevant project folder in your folder will be referred to as `WORKSPACE$`.

### Application Project Modifications

1. The device information profile path has changed. If this is part of your project, remove the old file (`devinfoservice.c`) from the project (it will be in the PROFILES group). Then add the file from the correct path (`$INSTALL\Projects\ble\Profiles\DevInfo\CC26xx\devinfoservice.c`)
2. There are several files in the `PROJECT$\CC6xx\CCS\Config` folder which need to be updated for 2.1.0. The easiest way to do this is to delete this folder and copy the Config folder from the SimpleBLEPeripheral sample project (`$INSTALL$\Projects\ble\SimpleBLEPeripheral\CC26xx\CCS\Config`) for use. Then, if there were any modifications to any of these files in the original 2.0.0 project, replicate the modifications here.
3. In the project properties, under Resource->Linked Resources in the Path Variables tab, update the following names:

```
CC26XXWARE = C:\ti\tirtos_simplelink_2_13_00_06\products\cc26xxware_2_21_01_15600
TI_RTOS_DRIVERS_BASE = C:\ti\tirtos_simplelink_2_13_00_06\packages
```

4. Under General in the Main tab, select the TI v5.2.4 compiler
5. Under General in the RTSC tab, select the 2.13.0.06 TI-RTOS and choose version 3.31.1.33 of XDCTools from the drop down menu. Then reselect `ti.platforms.simplelink:CC2640F128` from the Platforms dropdown menu.
6. Under Build->ARM Compiler->Advanced Options->Predefined Symbols, remove the `__TI_COMPILER_VERSION=1` define
7. Clean out the old RTOS by deleting the `PROJECT$\CC25xx\CCS\Config\src` and `WORKSPACE$\FlashROM\configPkg` folders if they exist.
8. Remove all uncommented lines from the `ccsLinkerDefines.cmd` and `ccsCompilerDefines.bcfg` files. These are located under the TOOLS folder.
9. Do not compile yet.

### Stack Project Modifications

1. The stack configuration file needs to be updated. The easiest way to do this is to replace the \$PROJECT\$\CC26xx\IAR\Stack\CC2640\buildConfig.opt file with \$INSTALL\Projects\ble\SimpleBLEPeripheral\CC26xx\IAR\Stack\CC2640\buildConfig.opt. Then make any desired modifications.
2. In the project properties, under Resource->Linked Resources in the Path Variables tab, update the following name:

```
CC26XXWARE = C:\ti\tirtos_simplelink_2_13_00_06\products\cc26xxware_2_21_01_15600
```

3. Delete the pwrmon.c and pwrmon.h files from the project. They can be found in HAL\Target\CC2650\Drivers
4. Under General in the Main tab, select the TI v5.2.4 compiler
5. Under Build->ARM Compiler->Advanced Options->Predefined Symbols, remove the \_\_TI\_COMPILER\_VERSION=1 define
8. Remove all uncommented lines from the ccsLinkerDefines.cmd and ccsCompilerDefines.bcfg files. These are located under the TOOLS folder.

## Compiling Steps

1. Compile the Stack Project. If there is a Boundary error after the first compilation, compile again. See the Boundary Tool in the software developer's guide for more information.
2. Compile the Application Project.
3. Load the Application Project.
4. Load the Stack Project.

## API Changes

---

1. The following array elements of structures defined in att.h have changed from uint8 to uint16:

- numInfo
- len
- numPairs
- numHandles
- numGrps

2. The GAP\_RegisterForHCIMsgs() command has changed to GAP\_RegisterForMsgs().

3. The gattService\_t structure has an additional encKeySize field: <syntaxhighlight lang='c'> typedef struct {

```
uint16 numAttrs; //!< Number of attributes in attrs
uint8 encKeySize; //!< Minimum encryption key size required by service (7-16 bytes)

/** Array of attribute records.
 * NOTE: The list must start with a Service attribute followed by
 * all attributes associated with this Service attribute.
 */
gattAttribute_t *attrs;
```

3. The gattService\_t; </syntaxhighlight> Because of this, the GATTServApp\_RegisterService() command has an additional encryption key size parameter: <syntaxhighlight lang='c'> bStatus\_t GATTServApp\_RegisterService( gattAttribute\_t

\*pAttrs,

```
uint16 numAttrs, uint8 encKeySize,
CONST gattServiceCBs_t *pServiceCBs )
```

4. The connection event callback command can now work with multiple connections. Therefore a connHandle parameter has been added: `hciStatus_t HCI_EXT_ConnEventNoticeCmd( uint16 connHandle, uint8 taskID, uint16 taskEvent );` 5. The `HCI_EXT_GetNumConnsCmd()` command has been replaced with the `HCI_EXT_GetConnInfoCmd()` command. See the software developer's guide for more information.

6. A connection handle parameter has been added to the `ggsAttrValueChange_t` function type: `typedef void (*ggsAttrValueChange_t)( uint16 connHandle, uint8 attrId );` 7. The read and write profile callback function types have been modified: the pLen and maxlen parameters have been changed from uint8 to uint16: `typedef bStatus_t (*pfnGATTReadAttrCB_t)( uint16 connHandle, gattAttribute_t *pAttr,`

```
uint8 *pValue, uint16 *pLen, uint16 offset,
uint16 maxlen, uint8 method );
```

`typedef bStatus_t (*pfnGATTWriteAttrCB_t)( uint16 connHandle, gattAttribute_t *pAttr,`

```
uint8 *pValue, uint16 len, uint16 offset,
uint8 method );
```

8. RSSI functionality has been moved from the GAPRole to the application. See the `simpleBLEcentral` project (and central GAPRole) for an example.

## Porting Drivers From TI-RTOS 2.20 or higher To TI-RTOS 2.18

The BLE-Stack v2.2 comes with `tirtos_cc13xx_cc26xx_2_18_00_03` and all the software testing was done with this combination. Therefore, it's not recommended to change TI-RTOS version if you want to use BLE-Stack. Here we are providing a workaround for users who would like to use the addition drivers which release in `tirtos_cc13xx_cc26xx_2_20_00_06` or later.

Please noted, here we are using `simple_peripheral` launchpad as example. You can use the same guide to port the drivers to work with different board files.

### Steps For Importing Drivers


1. Rename the PWM folder, driver files and board file folder in TI-RTOS 2.18(C:\ti\tirtos\_cc13xx\_cc26xx\_2\_18\_00\_03\products\tidrivens\_cc13xx\_cc26xx\_2\_16\_01\_13\packages\ti\drivers) to something different (optional, but recommended)

 pwm_origin	9/27/2016 3:40 PM	File folder
 PWM_origin.c	4/22/2016 12:42 AM	C File
 PWM_origin.h	4/22/2016 12:42 AM	H File
 Watchdog_origin.c	4/22/2016 12:42 AM	C File
 Watchdog_origin.h	4/22/2016 12:42 AM	H File

C:\ti\tirtos\_cc13xx\_cc26xx\_2\_18\_00\_03\products\tidrivrs\_cc13xx\_cc26xx\_2\_16\_01\_13\packages\ti\boards














 CC2650_LAUNCHXL_origin	9/27/2016 3:42 PM	File folder
--	-------------------	-------------

2. Rename the board files in BLE stack(C:\ti\simplelink\ble\_sdk\_2\_02\_00\_31\src\boards)

 CC2650_LAUNCHXL_origin	9/27/2016 3:47 PM	File folder
--	-------------------	-------------

3. Copy the needed driver and board files from TI-RTOS 2.20 over to TI-RTOS 2.18(we copied all the new drivers for convenience, so we don't need to do any more modification on the board files other than copy&paste)

From C:\ti\tirtos\_cc13xx\_cc26xx\_2\_20\_00\_06\products\tidrivrs\_cc13xx\_cc26xx\_2\_20\_00\_08\packages\ti\drivers to C:\ti\tirtos\_cc13xx\_cc26xx\_2\_18\_00\_03\products\tidrivrs\_cc13xx\_cc26xx\_2\_16\_01\_13\packages\ti\drivers

 adc	9/27/2016 3:50 PM	File folder
 adcbuf	9/27/2016 3:50 PM	File folder
 pwm	9/27/2016 3:50 PM	File folder
 timer	9/27/2016 3:50 PM	File folder
 watchdog	9/27/2016 3:50 PM	File folder
 ADC.c	6/22/2016 10:47 PM	C File
 ADC.h	6/22/2016 10:47 PM	H File
 ADCBuf.c	6/22/2016 10:47 PM	C File
 ADCBuf.h	6/22/2016 10:47 PM	H File
 PWM.c	6/22/2016 10:47 PM	C File
 PWM.h	6/22/2016 10:47 PM	H File
 Watchdog.c	6/22/2016 10:47 PM	C File
 Watchdog.h	6/22/2016 10:47 PM	H File

From C:\ti\tirtos\_cc13xx\_cc26xx\_2\_20\_00\_06\products\tidriv... \ti\boards to C:\ti\tirtos\_cc13xx\_cc26xx\_2\_18\_00\_03\products\tidriv... \ti\boards

CC2650\_LAUNCHXL

9/27/2016 3:52 PM

File folder

4. Copy the needed board files from TI-RTOS 2.20 to BLE stack

From C:\ti\tirtos\_cc13xx\_cc26xx\_2\_20\_00\_06\products\tidriv... \ti\boards to C:\ti\simplelink\ble\_sdk\_2\_02\_00\_31\src\boards

CC2650\_LAUNCHXL

9/27/2016 3:52 PM

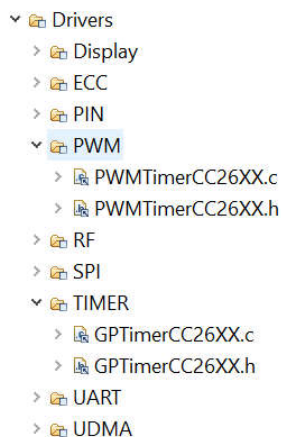
File folder

Comment out the following line in the CC2650\_LAUNCHXL.h(Only needed for the board files in BLE stack)

```
<syntaxhighlight lang='c'> #define CC2650_LAUNCHXL </syntaxhighlight>
```

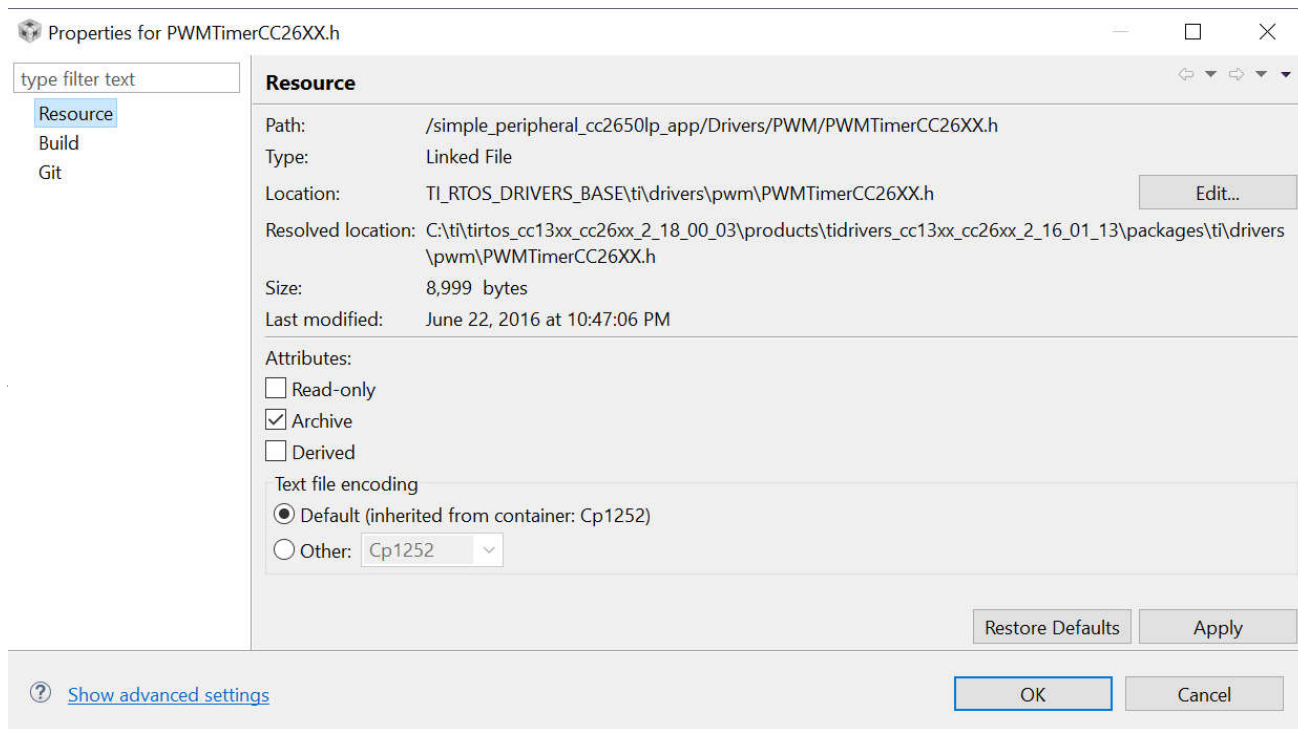
## Continue For CCS User

5. Import simple\_peripheral to CCS, make sure the compiler version for both stack and app >= 5.2.6
6. Add virtual folders under the Drivers folder and drag the files into the new folders you made



7. Make sure the location of the files are relative to TI\_RTOS\_DRIVERS\_BASE





8. In  
file

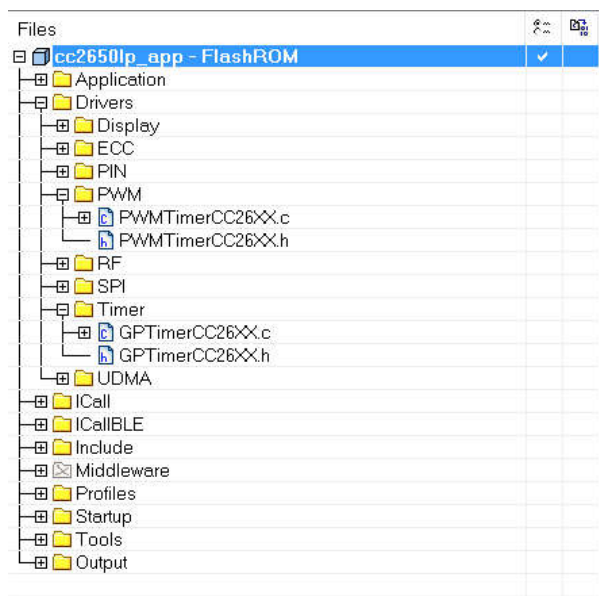
main.c, add the following code `<syntaxhighlight lang='c'> #include <ti/drivers/PWM.h>; </syntaxhighlight>`

9. In `int main()`, add `<syntaxhighlight lang='c'> PWM_init(); </syntaxhighlight>`

10. Delete the FlashROM folder in app, then build stack and application.

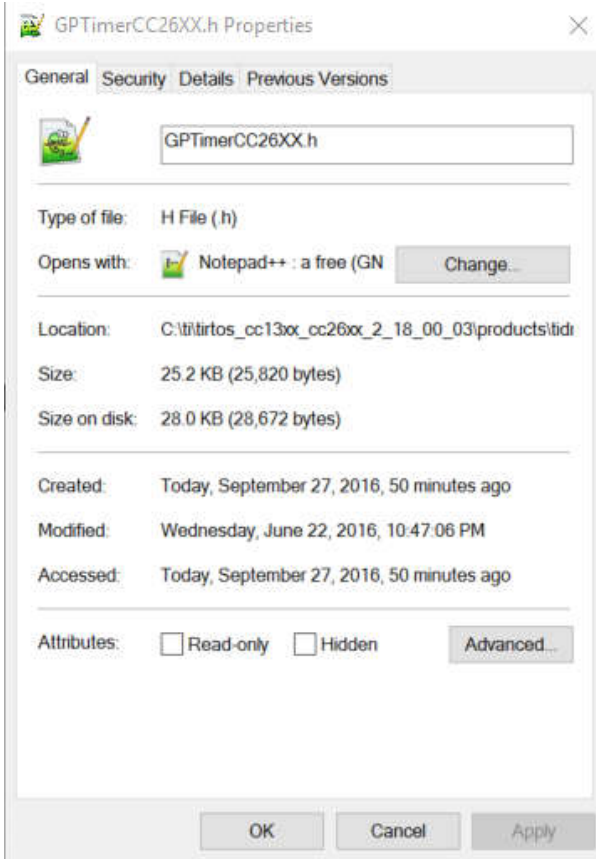
## Continue For IAR User

5. Add virtual folders under the Drivers folder and drag the files into the new folders you made



6. Make sure the location of the files are relative to

TI\_RTOS\_DRIVERS\_BASE



7. In file main.c, add the following code <code><pre><code>

```

<code>#include <ti/drivers/PWM.h>; </code>
<code>PWM_init(); </code>

```
8. In int main(), add <code><pre><code>

```

<code>PWM_init(); </code>

```

</code></pre></code>

9. Rebuild the cc2650lp\_app

<p>1. switchcategory:MultiCore=</p> <ul style="list-style-type: none"> <li>■ For technical support on MultiCore devices, please post your questions in the <a href="#">C6000 MultiCore Forum</a></li> <li>■ For questions related to the BIOS MultiCore SDK (MCSDK), please use the <a href="#">BIOS Forum</a></li> </ul> <p>Please post only comments related to the article <b>CC2640 Porting Projects</b> here.</p>	<p><b>Keystone=</b></p> <ul style="list-style-type: none"> <li>■ For technical support on MultiCore devices, please post your questions in the <a href="#">C6000 MultiCore Forum</a></li> <li>■ For questions related to the BIOS MultiCore SDK (MCSDK), please use the <a href="#">BIOS Forum</a></li> </ul> <p>Please post only comments related to the article <b>CC2640 Porting Projects</b> here.</p>	<p><b>C2000=For technical support on the C2000 please post your questions on The C2000 Forum.</b> Please post only comments about the article <b>CC2640 Porting Projects</b> here.</p>	<p><b>DaVinci=For technical support on the DaVinci Forum.</b> Please post only comments about the article <b>CC2640 Porting Projects</b> here.</p>	<p><b>MSP430=For technical support on the MSP430 please post your questions on The MSP430 Forum.</b> Please post only comments about the article <b>CC2640 Porting Projects</b> here.</p>
--	--	--	--	---

## Links

<a href="#">Amplifiers &amp; Linear</a>	<a href="#">DLP &amp; MEMS</a>	<a href="#">Processors</a>	<a href="#">Switches &amp; Multiplexers</a>
<a href="#">Audio</a>	<a href="#">High-Reliability</a>	<ul style="list-style-type: none"><li>■ <a href="#">ARM Processors</a></li></ul>	<a href="#">Temperature Sensors &amp;</a>
<a href="#">Broadband RF/IF &amp; Digital</a>	<a href="#">Interface</a>	<ul style="list-style-type: none"><li>■ <a href="#">Digital Signal Processors (DSP)</a></li></ul>	<a href="#">Control ICs</a>
<a href="#">Radio</a>	<a href="#">Logic</a>	<ul style="list-style-type: none"><li>■ <a href="#">Microcontrollers (MCU)</a></li></ul>	<a href="#">Wireless Connectivity</a>
<a href="#">Clocks &amp; Timers</a>	<a href="#">Power Management</a>	<ul style="list-style-type: none"><li>■ <a href="#">OMAP Applications Processors</a></li></ul>	
<a href="#">Data Converters</a>			

---

Retrieved from "[https://processors.wiki.ti.com/index.php?title=CC2640\\_Porting\\_Projects&oldid=234371](https://processors.wiki.ti.com/index.php?title=CC2640_Porting_Projects&oldid=234371)"

---

**This page was last edited on 23 April 2018, at 19:43.**

Content is available under [Creative Commons Attribution-ShareAlike](#) unless otherwise noted.