TI Designs

BLE Enabled IoT Node on High-Performance Microcontrollers

TI Designs

TI Designs provide the foundation that you need including methodology, testing and design files to quickly evaluate and customize the system. TI Designs help *you* accelerate your time to market.

Design Resources

TIDM-TM4C129XBLE	Design Folder
TM4C1294NCPDT	Product Folder
SimpleLink CC2650	Product Folder
EK-TM4C1294XL	Tools Folder
CC2650EMK	Tools Folder
BOOST-CCEM ADAPTER	Tools Folder
BLE-STACK-2	Tools Folder
CC2650DK	Tools Folder

TI E2E™ Community

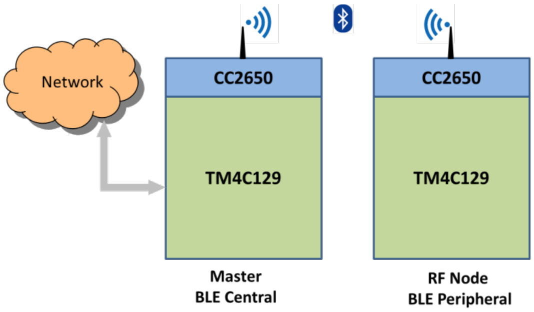
ASK Our E2E Experts
WEBENCH® Calculator Tools

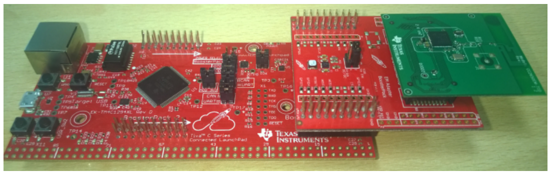
Design Features

- The Master is a TM4C1294 MCU and the CC2650 as BLE Central.
- The Slave is a TM4C1294 MCU and the CC2650 as BLE Peripheral.
- LWIP-Based Web Server Runs on the Master Side TM4C1294 MCU.
- The BLE Stack Runs on CC2650.
- The TM4C1294 Works as the Host Processor Performing a Demo Application.
- HTML Code Remotely Controls Slave Operation of the TM4C1294 From a Web Browser.
- Code Composer Studio™ is for Development and Debugging.
- TI RTOS is for Task Scheduling and Peripheral Access.

Featured Applications

- Industrial Application and Automation
- Home Automation
- Smart Grid and Energy
- Test and Measurement





An IMPORTANT NOTICE at the end of this TI reference design addresses authorized use, intellectual property matters and other important disclaimers and information.

Code Composer Studio, LaunchPad, SimpleLink, BoosterPack, TivaWare, Tiva are trademarks of Texas Instruments.
Stellaris is a registered trademark of Texas Instruments.
ARM, Cortex are registered trademarks of ARM Limited.
Bluetooth is a registered trademark of Bluetooth SIG.
ZigBee is a registered trademark of Zigbee Alliance.
All other trademarks are the property of their respective owners.

1 System Description

Bluetooth® low energy, also known as *Bluetooth* Smart (BLE), is one of the most common low-power wireless connectivity technologies. BLE is the intelligent, power-friendly version of *Bluetooth* wireless technology. While the power-efficiency of BLE is perfect for devices that run off a tiny battery for long periods, the magic of BLE is its ability to work with an application on a smartphone or tablet you already own. BLE lets developers and OEMs easily create solutions that work with the billions of *Bluetooth*-enabled products already in the market.

This reference design shows how to create a BLE node using the TM4C129 high-performance microcontroller and the single-mode CC2650. For more information about this BLE node application, refer to BLE-Enabled IoT Node With High-Performance MCU Reference Design.

The software accompanying this design works on an EK-TM4C1294XL LaunchPad™ integrated with a CC2650EMK.

TIRTOS schedules various tasks. TI recommends using RTOS to distribute the load and make the application easily scalable.

1.1 *TM4C1294NCPDT*

The TM4C1294NCPDT device is a 120-MHz high-performance microcontroller with 1MB of on-chip flash and 256KB on-chip SRAM, and features an integrated Ethernet MAC+PHY for connected applications. The device has high-bandwidth interfaces such as a memory controller and a high-speed USB2.0 digital interface. With the integration of numerous low-to-mid speed serials (up to 4 million samples per second [MSPS]), a 12-bit ADC, and motion control peripherals, the TM4C1294NCPDT microcontroller is ideal for use with industrial communication equipment applications to Smart Energy or Smart Grid applications.

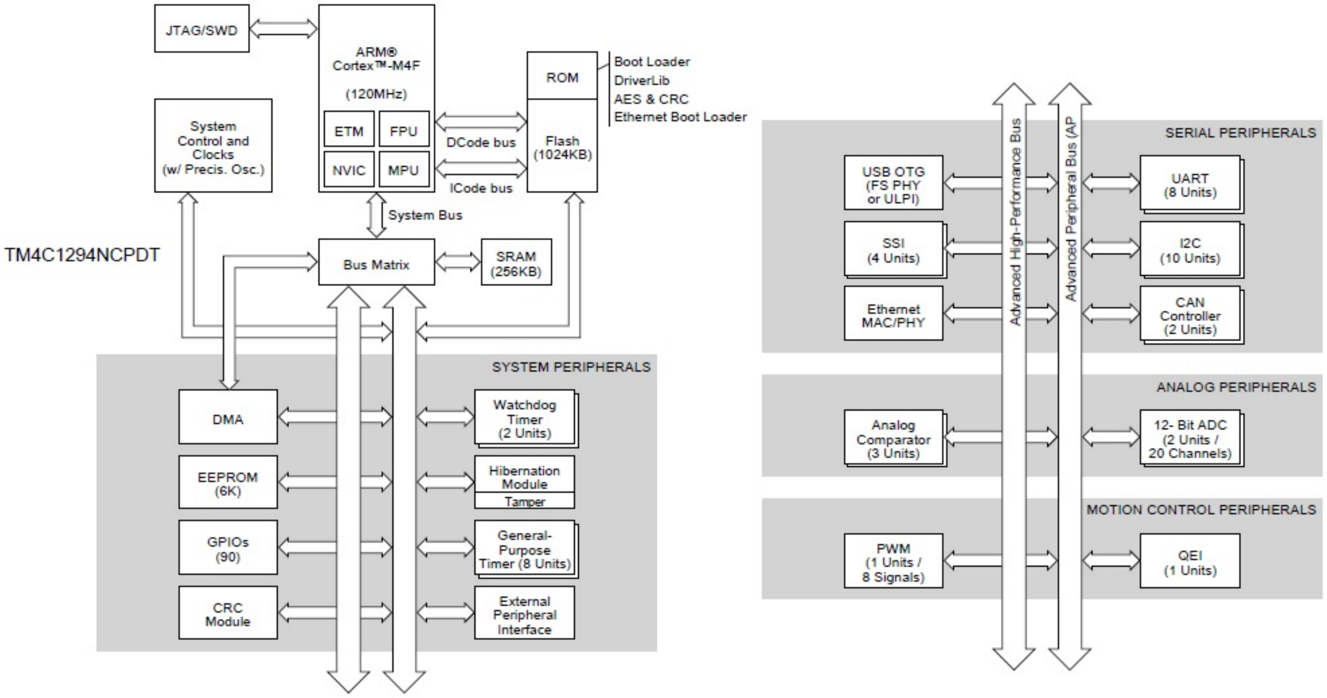


Figure 1. TM4C1294NCPDT Microcontroller High-Level Block Diagram

1.2 CC2650

The CC2650 is a cost-effective, ultralow power, 2.4-GHz RF wireless MCU targeting *Bluetooth* Smart, ZigBee® and 6LoWPAN, and ZigBee RF4CE remote control applications. A very low active RF and MCU current and low-power mode current consumption provides excellent battery lifetime, operates on small coin-cell batteries, and operates in energy-harvesting applications.

The CC2650 contains a 32-bit ARM® Cortex®-M3 running at 48MHz as the main processor and has a rich peripheral feature set, including an ultralow power sensor controller. The ultralow power sensor controller is ideal for interfacing external sensors or collecting analog and digital data while the rest of the system is in sleep mode. The *Bluetooth* low-energy controller and the IEEE 802.15.4 MAC are embedded into ROM and are running partially on a separate ARM Cortex®-M0 processor. This architecture improves overall system performance and power consumption and frees up flash memory for the application.

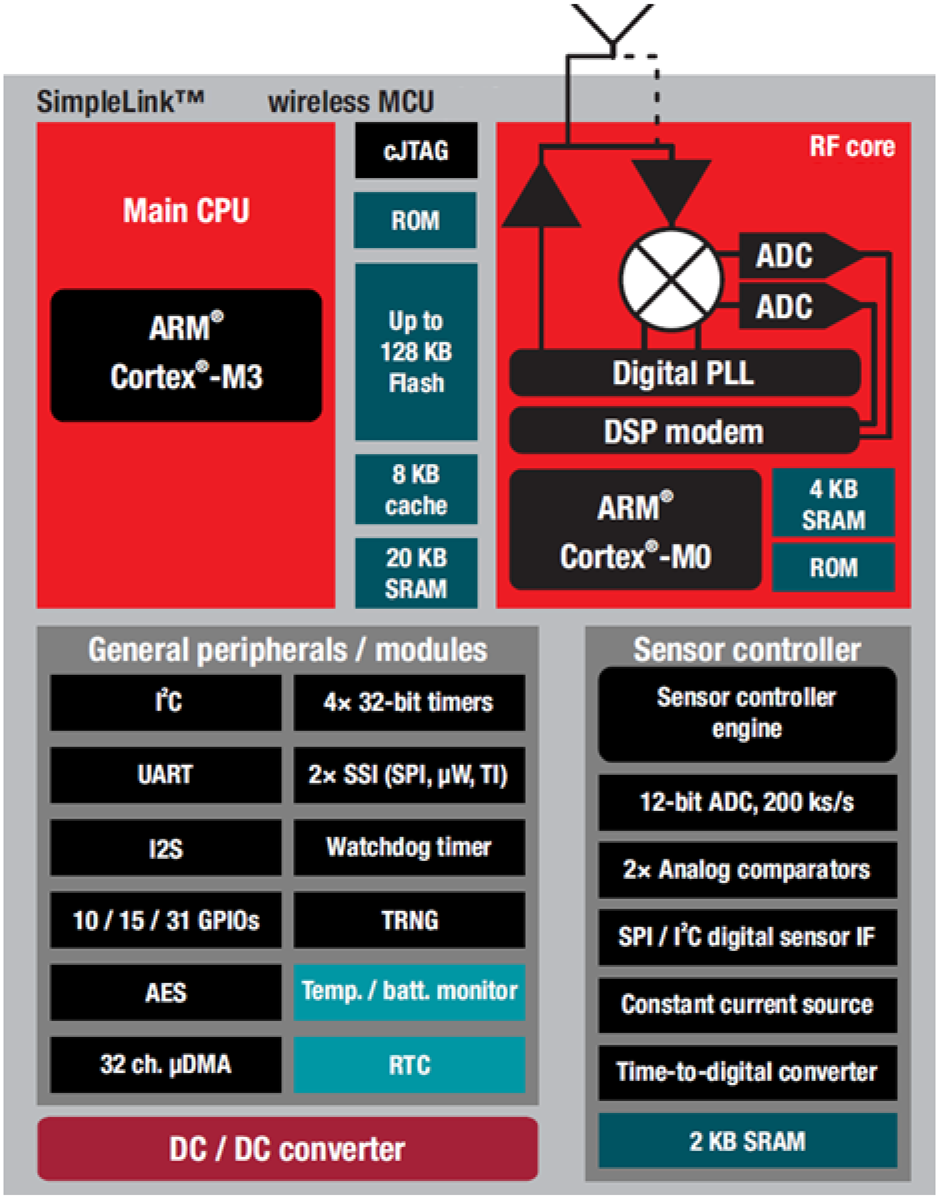


Figure 2. CC2650 Hardware Overview

1.3 TM4C129 and CC265 Interface

Figure 3 illustrates the interface between the TM4C1294 and the CC2650. The TM4C1294 communicates to the CC2650 through UART. A simple command response protocol is implemented for this demonstration. UART0 is used in CC2650 and UART7 is used in TM4C of this demonstration.

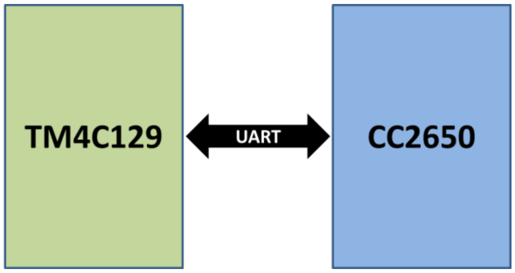


Figure 3. TM4C129 and CC2650 Interface Overview

2 System Functionality Block Diagram

The TM4C BLE node can be configured in two modes: central and peripheral. The wired master node acts as BLE central and RF node acts as BLE peripheral. For this demonstration, the CC2650 is configured in single-processor mode and the TM4C acts as the application controller. The demo BLE profiles and services are in the CC2650. Network processor architecture is beyond the scope for this software design.

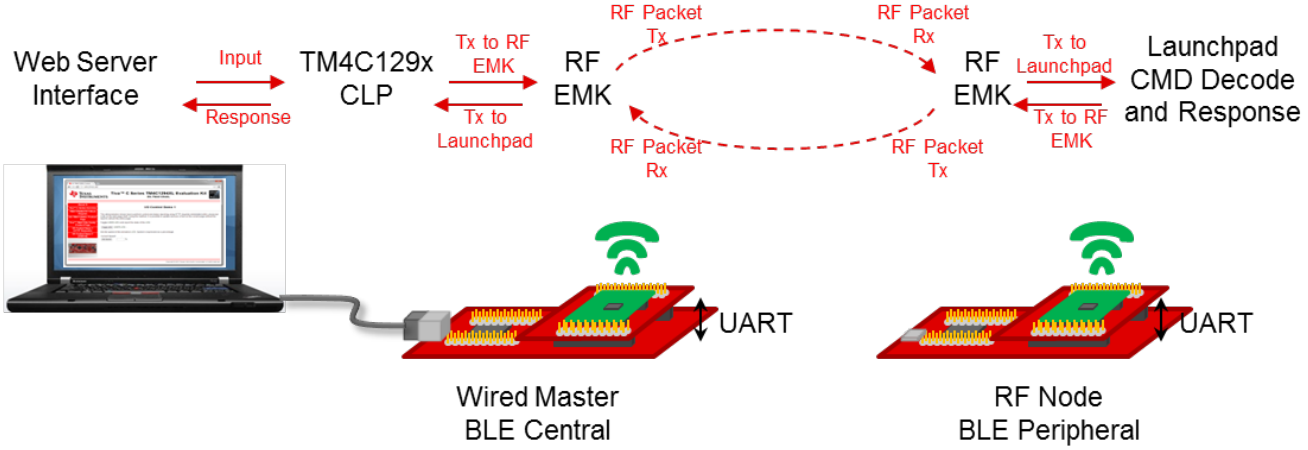


Figure 4. General Setup and Dataflow for BLE-Enabled IoT Node

2.1 TM4C BLE Node as a Central

The TM4C performs the following tasks:

- Runs LWIP Ethernet-based Web server.
- Displays assigned IP addresses through UART0.
- Receives user commands requested through a hosted Web page.
- Sends user commands to the CC2650 device through UART, and waits for a response.
- Responds to the web page http request for the selected demo command.

The CC2650 performs the following tasks:

- Waits for commands from the TM4C through UART.
- Discovers and connects to the BLE Peripheral node after receiving a Connect command.
- Interacts with the BLE peripheral node depending on the requested command. Read and Write characteristics value requests are sent to the BLE peripheral node.
- Responds to the TM4C with the data read from the TM4C BLE peripheral node.

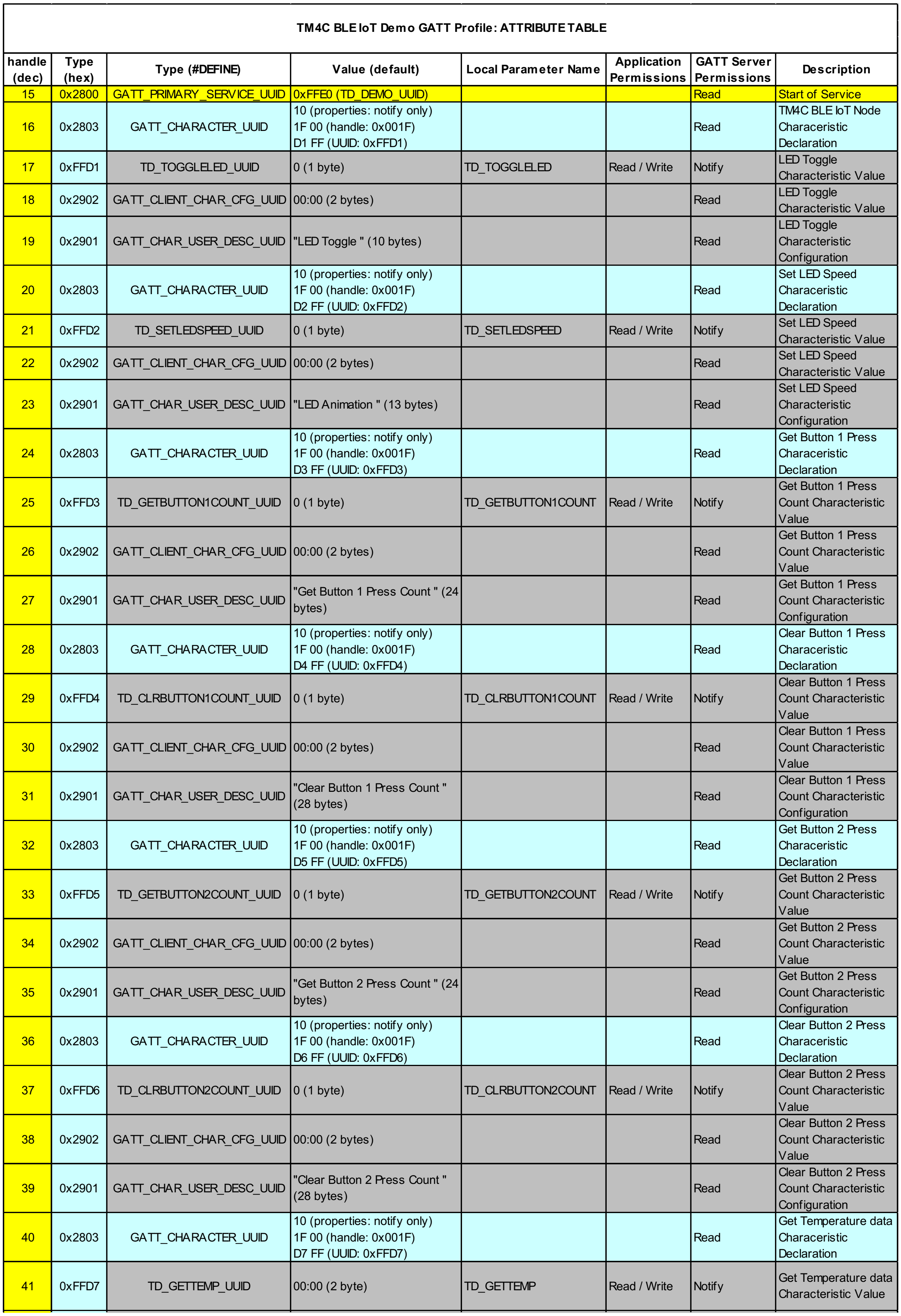
2.2 TM4C BLE Node as a Peripheral

The CC2650 performs the following tasks:

- Advertises primary characteristics for connection.
- Connects to the TM4C BLE central on request.
- Receives Read or Write characteristics value request coming from the TM4C BLE central.
- Sends user commands to the TM4C device through UART, and waits for a response.
- Responds to the Read or Write characteristic value request coming from the TM4C BLE central

The TM4C performs the following tasks:

- Receives demo commands coming through the BLE peripheral.
- Performs actions or tasks associated with the user demo command.
- Responds with the appropriate data or acknowledgment to the CC2650 BLE peripheral through TM4C UART7.



TM4C BLE IoT Demo GATT Profile: ATTRIBUTE TABLE

handle (dec)	Type (hex)	Type (#DEFINE)	Value (default)	Local Parameter Name	Application Permissions	GATT Server Permissions	Description
15	0x2800	GATT_PRIMARY_SERVICE_UUID	0xFFE0 (TD_DEMO_UUID)			Read	Start of Service
16	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D1 FF (UUID: 0xFFD1)			Read	TM4C BLE IoT Node Characeristic Declaration
17	0xFFD1	TD_TOGGLELED_UUID	0 (1 byte)	TD_TOGGLELED	Read / Write	Notify	LED Toggle Characteristic Value
18	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	LED Toggle Characteristic Value
19	0x2901	GATT_CHAR_USER_DESC_UUID	"LED Toggle " (10 bytes)			Read	LED Toggle Characteristic Configuration
20	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D2 FF (UUID: 0xFFD2)			Read	Set LED Speed Characeristic Declaration
21	0xFFD2	TD_SETLEDSPEED_UUID	0 (1 byte)	TD_SETLEDSPEED	Read / Write	Notify	Set LED Speed Characteristic Value
22	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	Set LED Speed Characteristic Value
23	0x2901	GATT_CHAR_USER_DESC_UUID	"LED Animation " (13 bytes)			Read	Set LED Speed Characteristic Configuration
24	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D3 FF (UUID: 0xFFD3)			Read	Get Button 1 Press Characeristic Declaration
25	0xFFD3	TD_GETBUTTON1COUNT_UUID	0 (1 byte)	TD_GETBUTTON1COUNT	Read / Write	Notify	Get Button 1 Press Count Characteristic Value
26	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	Get Button 1 Press Count Characteristic Value
27	0x2901	GATT_CHAR_USER_DESC_UUID	"Get Button 1 Press Count " (24 bytes)			Read	Get Button 1 Press Count Characteristic Configuration
28	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D4 FF (UUID: 0xFFD4)			Read	Clear Button 1 Press Characeristic Declaration
29	0xFFD4	TD_CLRBUTTON1COUNT_UUID	0 (1 byte)	TD_CLRBUTTON1COUNT	Read / Write	Notify	Clear Button 1 Press Count Characteristic Value
30	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	Clear Button 1 Press Count Characteristic Value
31	0x2901	GATT_CHAR_USER_DESC_UUID	"Clear Button 1 Press Count " (28 bytes)			Read	Clear Button 1 Press Count Characteristic Configuration
32	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D5 FF (UUID: 0xFFD5)			Read	Get Button 2 Press Characeristic Declaration
33	0xFFD5	TD_GETBUTTON2COUNT_UUID	0 (1 byte)	TD_GETBUTTON2COUNT	Read / Write	Notify	Get Button 2 Press Count Characteristic Value
34	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	Get Button 2 Press Count Characteristic Value
35	0x2901	GATT_CHAR_USER_DESC_UUID	"Get Button 2 Press Count " (24 bytes)			Read	Get Button 2 Press Count Characteristic Configuration
36	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D6 FF (UUID: 0xFFD6)			Read	Clear Button 2 Press Characeristic Declaration
37	0xFFD6	TD_CLRBUTTON2COUNT_UUID	0 (1 byte)	TD_CLRBUTTON2COUNT	Read / Write	Notify	Clear Button 2 Press Count Characteristic Value
38	0x2902	GATT_CLIENT_CHAR_CFG_UUID	00:00 (2 bytes)			Read	Clear Button 2 Press Count Characteristic Value
39	0x2901	GATT_CHAR_USER_DESC_UUID	"Clear Button 2 Press Count " (28 bytes)			Read	Clear Button 2 Press Count Characteristic Configuration
40	0x2803	GATT_CHARACTER_UUID	10 (properties: notify only) 1F 00 (handle: 0x001F) D7 FF (UUID: 0xFFD7)			Read	Get Temperature data Characeristic Declaration
41	0xFFD7	TD_GETTEMP_UUID	00:00 (2 byte)	TD_GETTEMP	Read / Write	Notify	Get Temperature data Characteristic Value

Figure 5. TM4C BLE Peripheral Profile Table

3 Getting Started Hardware

For both the master central and slave peripheral nodes, the hardware is the EK-TM4C129XL-connected LaunchPad and the CC2650 EMK board. The EK-TM4C129XL-connected LaunchPad board is connected to the SimpleLink™ CC2650 BoosterPack™ board through the BoosterPack connector 1 and an adapter board. The communication channel is UART in 2-pin standard mode. Table 1 lists the necessary signal mapping for the demonstration.

Table 1. Signal Mapping

BoosterPack Connector	TM4C1294 LaunchPad	CC2650EMK	EM Adapter BoosterPack
A1-1	3.3 V	3.3 V	VDD_LP
A1-2	PE4	Unused	PE4
A1-3	PC4_U7RX	IOID_2	LP1-3
A1-4	PC5_U7TX	IOID_3	LP1-4
A1-5	PC6	Unused	Unused
A1-6	PE5	Unused	Unused
A1-7	PD3_SSI2CLK	Unused	Unused
A1-8	PC7	Unused	Unused
A1-9	PB2	Unused	Unused
A1-10	PB3	Unused	Unused
D1-1	GND	GND	GND
D1-2	PM3	Unused	Unused
D1-3	PH2	Unused	Unused
D1-4	PH3	Unused	Unused
D1-5	RESET	Unused	Unused
D1-6	PD1_I2C7SDA	Unused	Unused
D1-7	PD0_I2C7SCL	Unused	Unused
D1-8	PN2	Unused	Unused
D1-9	ON3	Unused	Unused
D1-10	PP2	Unused	Unused

4 Getting Started Software

4.1 TM4C BLE Node as a Central Software Architecture

Figure 6 illustrates the architecture of the TM4C1294 BLE central node.

TM4C software blocks:

- TivaWare™ C – Allows for TM4C hardware register access and UART communication.
- LWIP Ethernet stack for Web server
- Demo packet handler
  - Converts http requests to UART demo commands.
  - Converts UART command responses to string format for Web page display.

CC2650 software blocks:

- TI RTOS – for general scheduling
  (a) Manages the demo command or response handling over UART.
  (b) Operates the BLE central.
- CC26xxWare – Performs CC2650 hardware access and UART operation.
- BLE Stack – Supports BLE protocol.

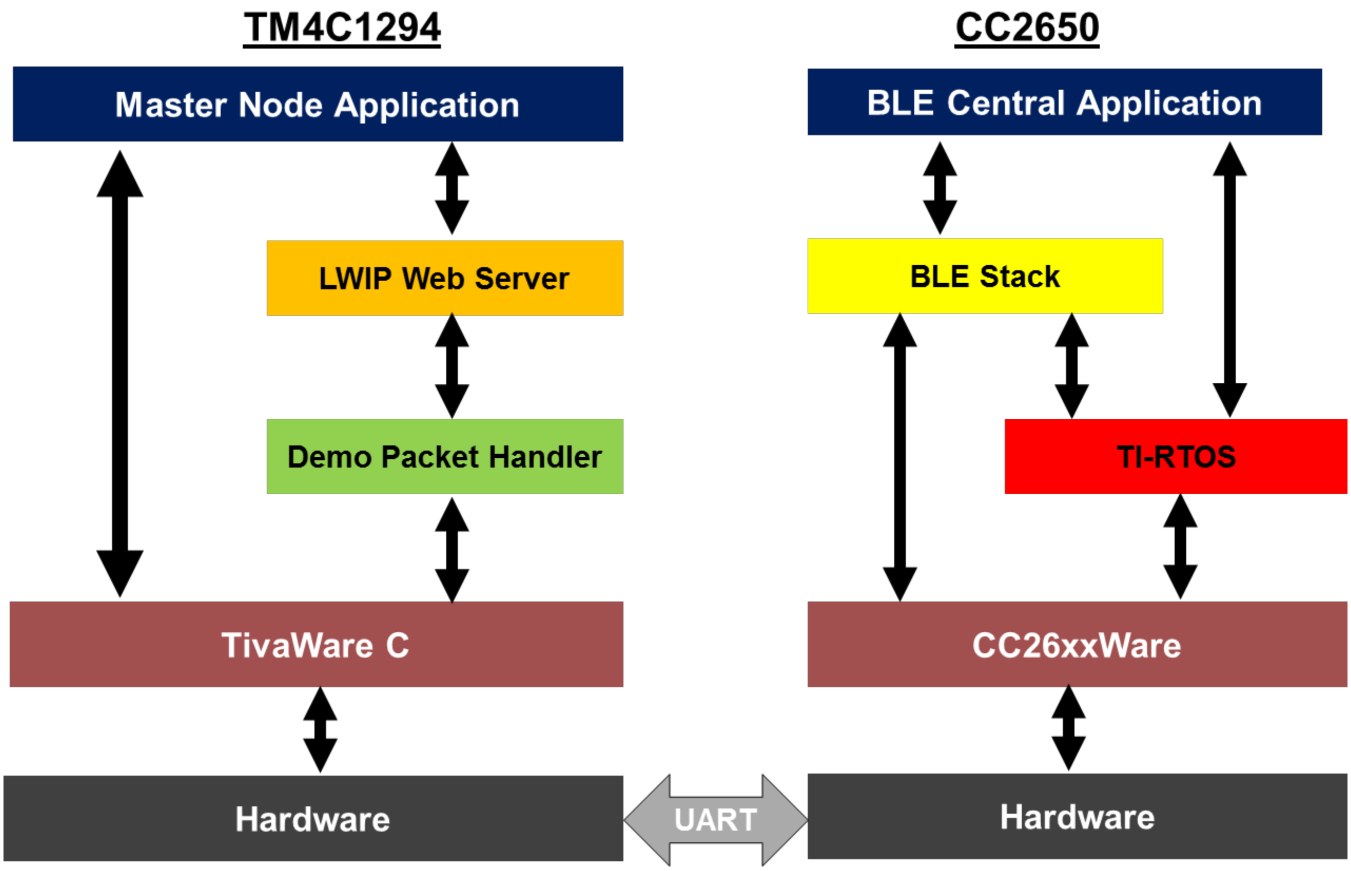


Figure 6. TM4C BLE Central Architecture Block Diagram

The following TI RTOS functions are statically-configured in the TI RTOS configuration file:

- cmdReceived_sem: Waits until the demo command is received.
- cmdResponse_sem: Waits until the demo response is ready.
- Task_ externalMCUComm: Performs UART command receive and response transmit.

4.2 TM4C BLE Node as a Peripheral Software Architecture

Figure 7 illustrates the architecture of the TM4C1294 BLE peripheral node.

CC2650 software blocks:

- TI RTOS – for general scheduling
  (a) Manages the demo command or response handling over UART.
  (b) Operates the BLE peripheral.
- CC26xxWare – Performs CC2650 hardware access and UART operation.
- BLE Stack – Supports the BLE protocol.

TM4C software blocks:

- TI RTOS – for general scheduling
  (a) Manages the demo command and response handling over UART.
  (b) Runs the demo application that performs LED control, temperature, button counts, and so forth
- TivaWare C – Performs TM4C hardware register access and UART communication.
- Demo packet handler – Decodes UART-based demo commands to demo tasks.

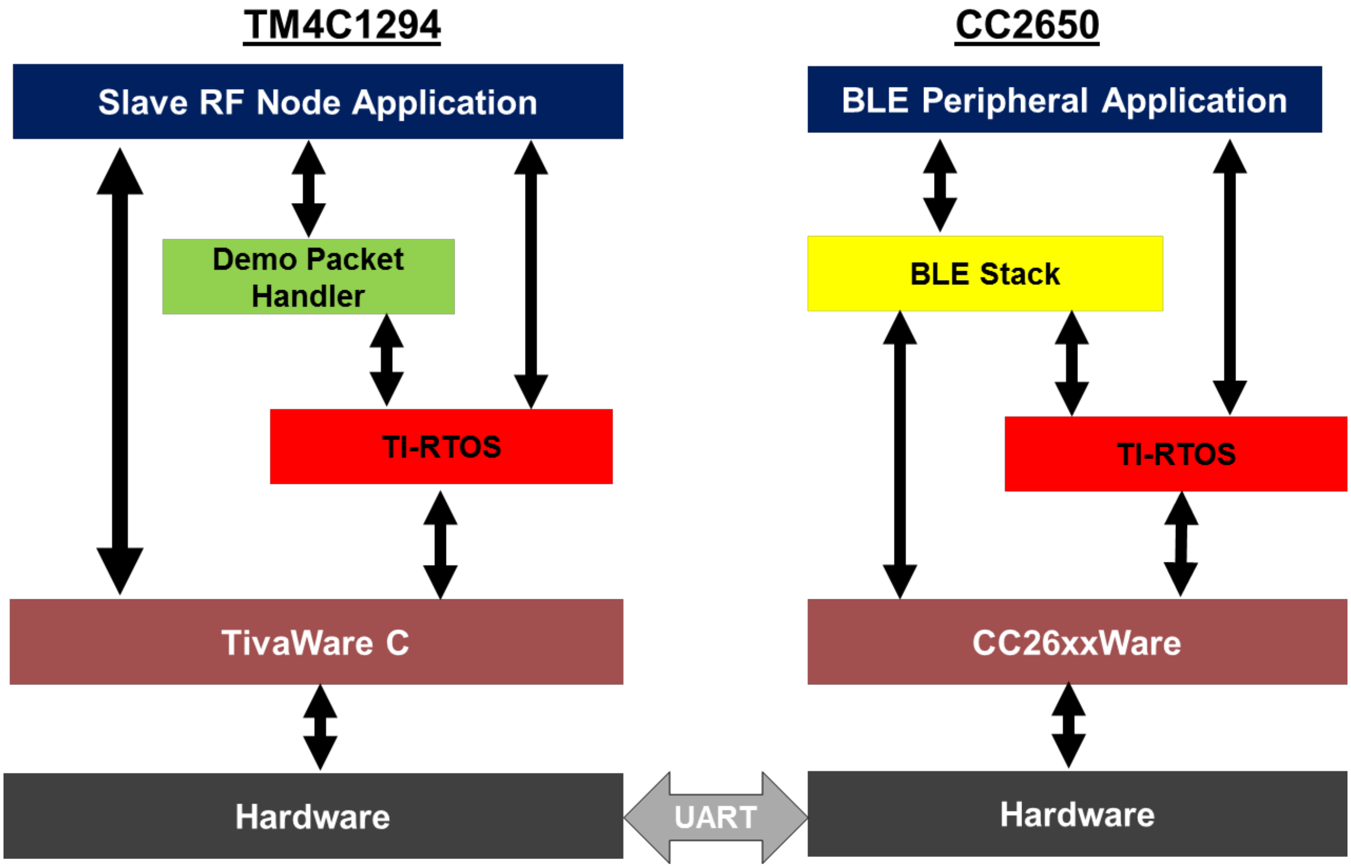


Figure 7. TM4C BLE Peripheral Architecture Block Diagram

The following TI RTOS functions are statically configured in the TI RTOS configuration file:

- TM4C:
  - ledAnimationClock: LED animation control
  - updateTempClock: Updates the temperature value every second.
  - updateButtonCountclock: Updates the button press status every 10 ms.
  - cmdReceived_sem: Waits until the demo command is received.

  - cmdResponse_sem: Waits until the demo response is ready.
  - Task_ uartCommand: Performs UART command receive and response transmit.
  - Task_ demoRFnode: Performs demo applications such as LED control, temperature, update button counts, and so forth.
- CC2650:
  - cmdReceived_sem: Waits until demo command is received.
  - cmdResponse_sem: Waits until demo response is ready.
  - Task_ externalMCUComm: Performs UART command receive and response transmit.

5 Software Setup

These tools and software packages are required to build and test access point and station projects:

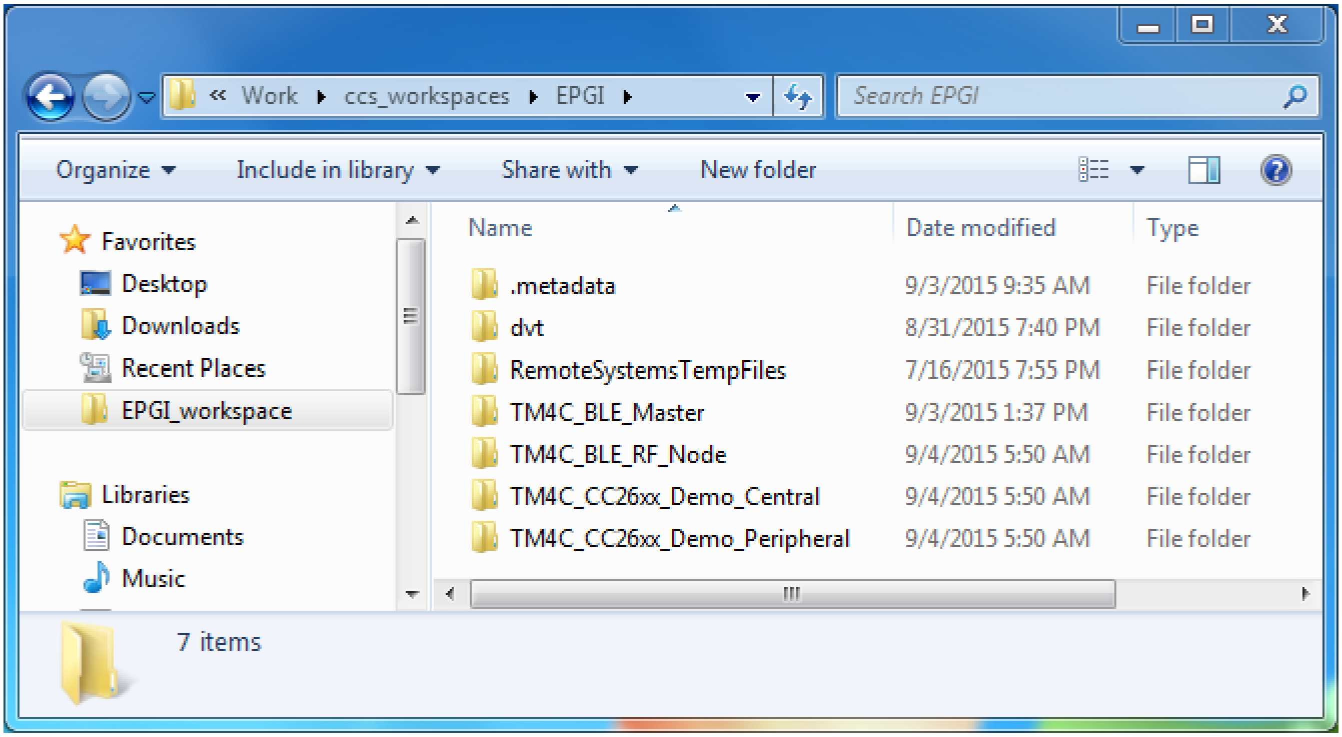
- Composer Studio (http://www.ti.com/tool/ccstudio)
- CC2650 BLE Stack-2 (http://www.ti.com/tool/ble-stack-archive)
- TI-RTOS for CC2650 v2_11_01_0910 (Part of CC2650 BLE Stack-2 Installer). CC26xxWare is included.
- TivaWare_C v2.1.1.71 (http://www.ti.com/tool/sw-tm4c)
- TI-RTOS for TIVA v2.14.0.10 (Resource Explorer in CCS)

NOTE: The demonstration is not compatible to BLE-STACK-2-1(http://www.ti.com/tool/ble-stack).

The demonstration is not compatible with tirtos_simplelink version 2_12_x, 2_13_x, 2_14_x due to the UART driver changes in these releases. TI recommends using 2_11_01_0910 for this demonstration.

TI recommends installing these packages in the default location under C:\ti to avoid making any changes in the CCS project. When the previous tools are installed, follow these steps:

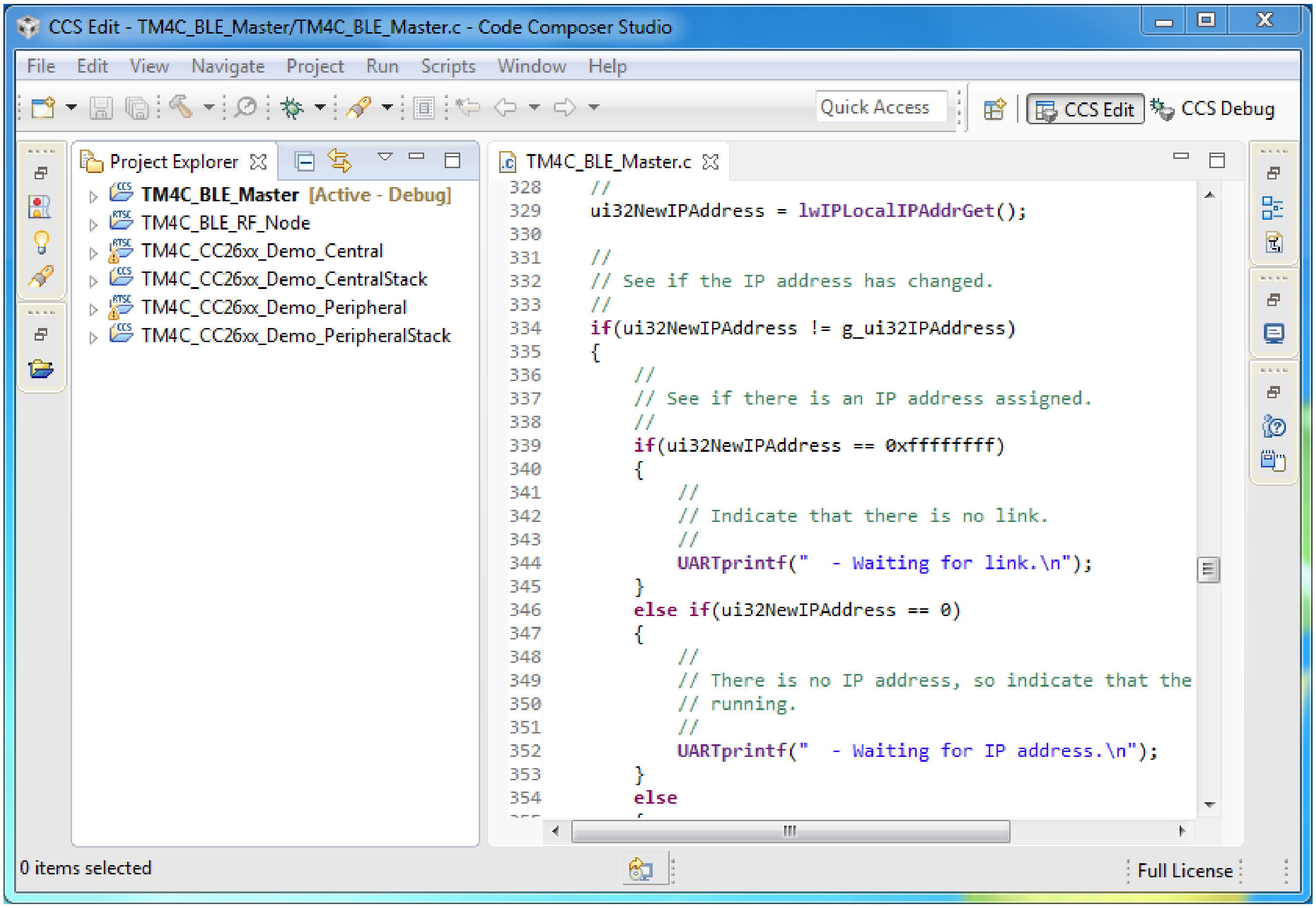
1. Unzip the software release zip file.
2. Place the extracted TM4C_CC26xx_Demo_Central, TM4C_CC26xx_Demo_Peripheral, TM4C_BLE_Master, and TM4C_BLE_RF_Node directories in your workspace. See Figure 8.



Name	Date modified	Type
.metadata	9/3/2015 9:35 AM	File folder
dvt	8/31/2015 7:40 PM	File folder
RemoteSystemsTempFiles	7/16/2015 7:55 PM	File folder
TM4C_BLE_Master	9/3/2015 1:37 PM	File folder
TM4C_BLE_RF_Node	9/4/2015 5:50 AM	File folder
TM4C_CC26xx_Demo_Central	9/4/2015 5:50 AM	File folder
TM4C_CC26xx_Demo_Peripheral	9/4/2015 5:50 AM	File folder

Figure 8. CCS Workspace

3. Import all projects into CCS (see Figure 9).


CCS Edit - TM4C_BLE_Master/TM4C_BLE_Master.c - Code Composer Studio
File Edit View Navigate Project Run Scripts Window Help
Quick Access
CCS Edit
CCS Debug
Project Explorer
TM4C_BLE_Master.c
Full License

- TM4C_BLE_Master [Active - Debug]
- TM4C_BLE_RF_Node
- TM4C_CC26xx_Demo_Central
- TM4C_CC26xx_Demo_CentralStack
- TM4C_CC26xx_Demo_Peripheral
- TM4C_CC26xx_Demo_PeripheralStack

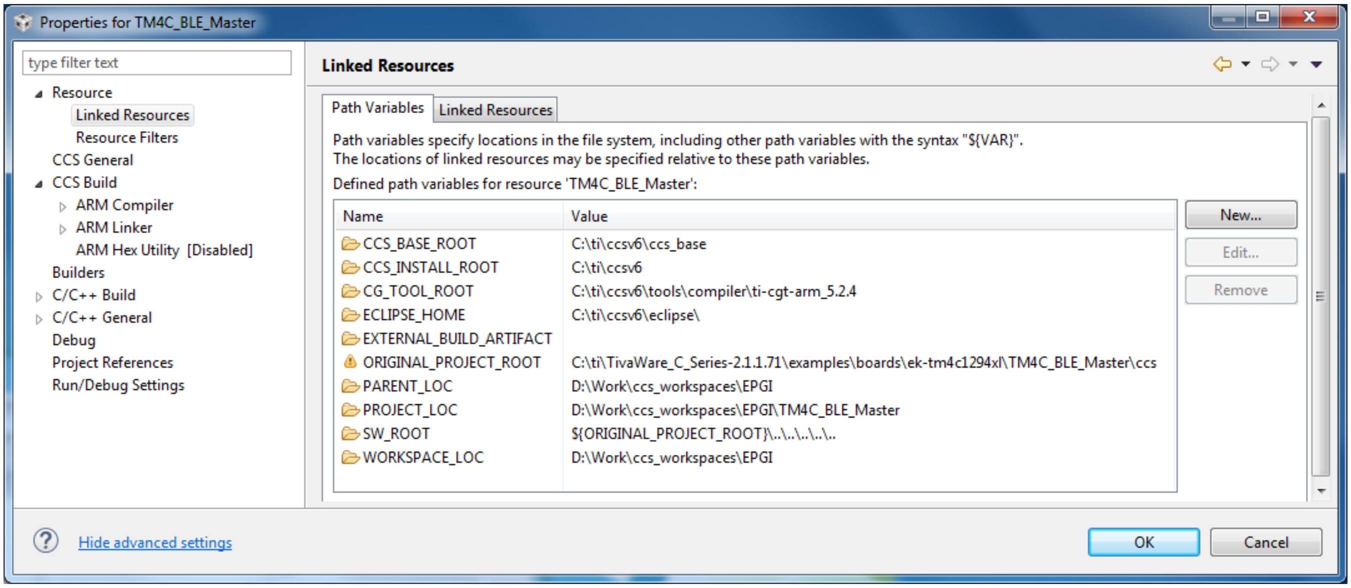
```
    //
    ui32NewIPAddress = lwIPLocalIPAddrGet();

    //
    // See if the IP address has changed.
    //
    if(ui32NewIPAddress != g_ui32IPAddress)
    {
        //
        // See if there is an IP address assigned.
        //
        if(ui32NewIPAddress == 0xffffffff)
        {
            //
            // Indicate that there is no link.
            //
            UARTprintf("  - Waiting for link.\n");
        }
        else if(ui32NewIPAddress == 0)
        {
            //
            // There is no IP address, so indicate that the
            // running.
            //
            UARTprintf("  - Waiting for IP address.\n");
        }
        else
```

0 items selected

Figure 9. CCS Projects

4. Check the Linked Resources Path Variables (see Figure 10) to confirm that they correspond to the actual folders in the current setup.

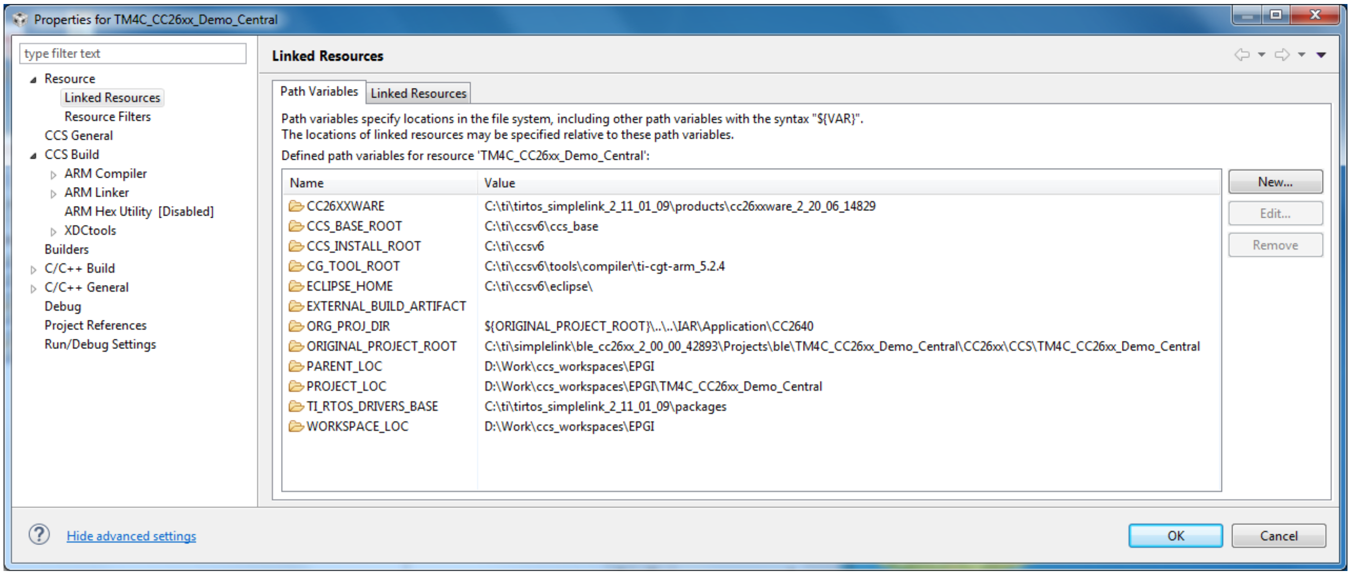


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_BLE_Master':

Name	Value
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIFACT	
⚠ ORIGINAL_PROJECT_ROOT	C:\ti\TivaWare_C_Series-2.1.1.71\examples\boards\ek-tm4c1294xl\TM4C_BLE_Master\ccs
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_BLE_Master
SW_ROOT	${ORIGINAL_PROJECT_ROOT}\..\..\..\..\..
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 10. Path Variables – TM4C Master

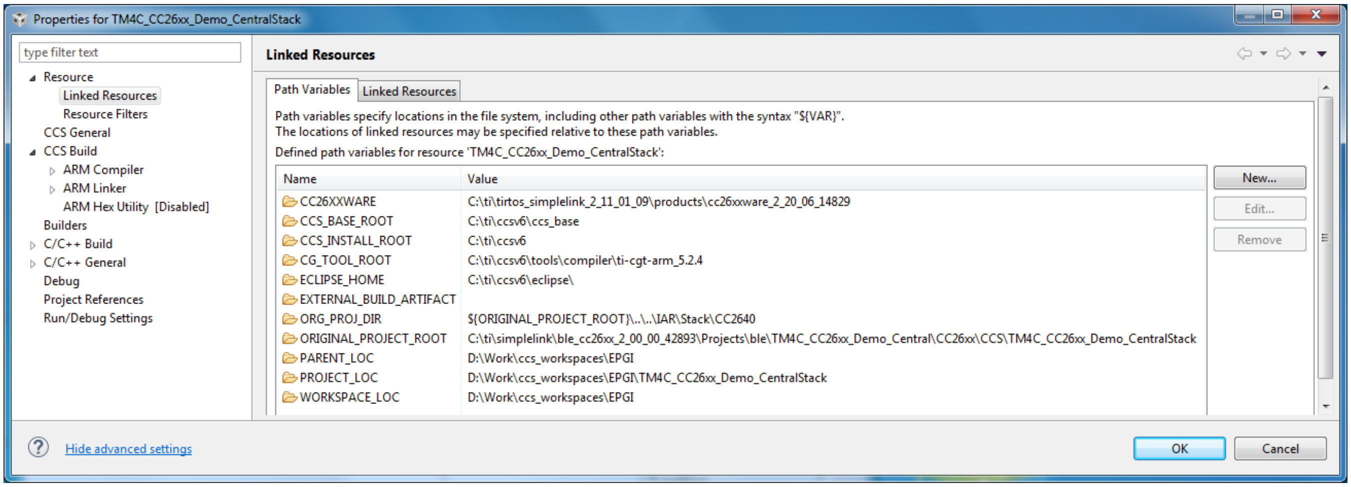


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_CC26xx_Demo_Central':

Name	Value
CC26XXWARE	C:\ti\tirtos_simplelink_2_11_01_09\products\cc26xxware_2_20_06_14829
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIFACT	
ORG_PROJ_DIR	${ORIGINAL_PROJECT_ROOT}\..\..\IAR\Application\CC2640
ORIGINAL_PROJECT_ROOT	C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble\TM4C_CC26xx_Demo_Central\CC26xx\CCS\TM4C_CC26xx_Demo_Central
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_CC26xx_Demo_Central
TI_RTOS_DRIVERS_BASE	C:\ti\tirtos_simplelink_2_11_01_09\packages
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 11. Path Variables – CC2650 Central

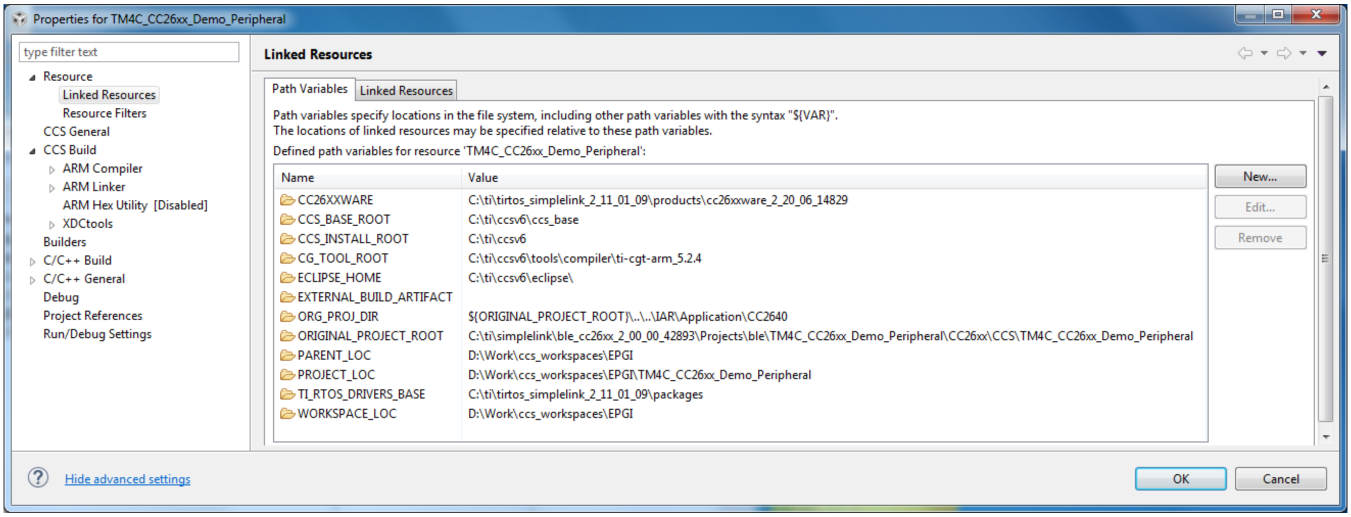


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_CC26xx_Demo_CentralStack':

Name	Value
CC26XXWARE	C:\ti\tirtos_simplelink_2_11_01_09\products\cc26xxware_2_20_06_14829
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIFACT	
ORG_PROJ_DIR	${ORIGINAL_PROJECT_ROOT}\..\..\IAR\Stack\CC2640
ORIGINAL_PROJECT_ROOT	C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble\TM4C_CC26xx_Demo_Central\CC26xx\CCS\TM4C_CC26xx_Demo_CentralStack
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_CC26xx_Demo_CentralStack
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 12. Path Variables – CC2650 Central Stack

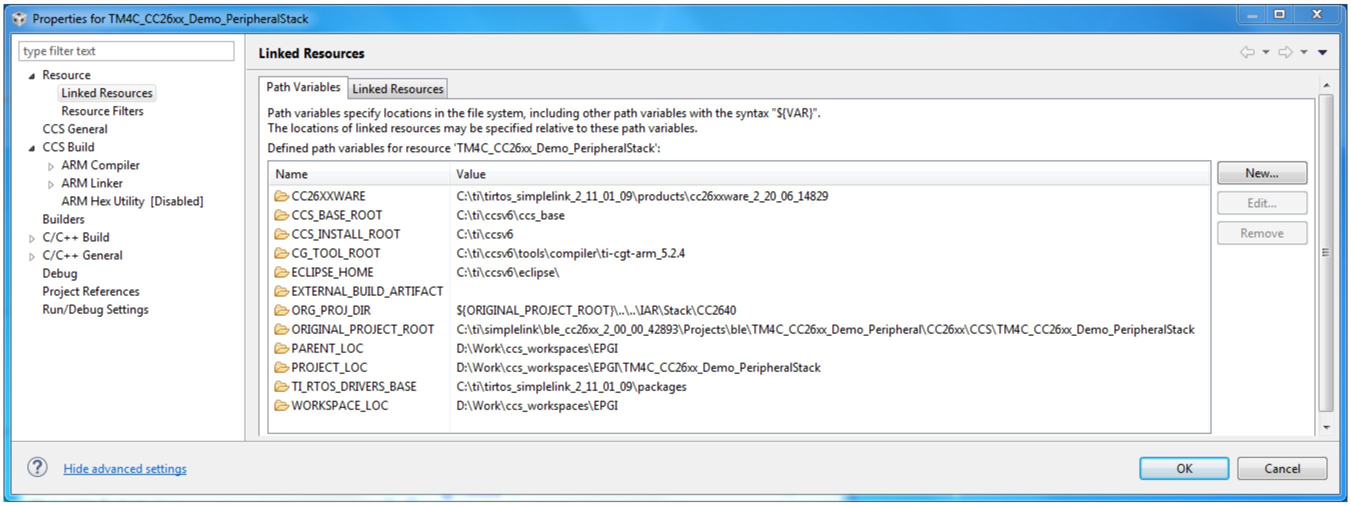


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_CC26xx_Demo_Peripheral':

Name	Value
CC26XXWARE	C:\ti\tirtos_simplelink_2_11_01_09\products\cc26xxware_2_20_06_14829
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIFACT	
ORG_PROJ_DIR	${ORIGINAL_PROJECT_ROOT}\..\..\IAR\Application\CC2640
ORIGINAL_PROJECT_ROOT	C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble\TM4C_CC26xx_Demo_Peripheral\CC26xx\CCS\TM4C_CC26xx_Demo_Peripheral
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_CC26xx_Demo_Peripheral
TI_RTOS_DRIVERS_BASE	C:\ti\tirtos_simplelink_2_11_01_09\packages
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 13. Path Variables – CC2650 Peripheral

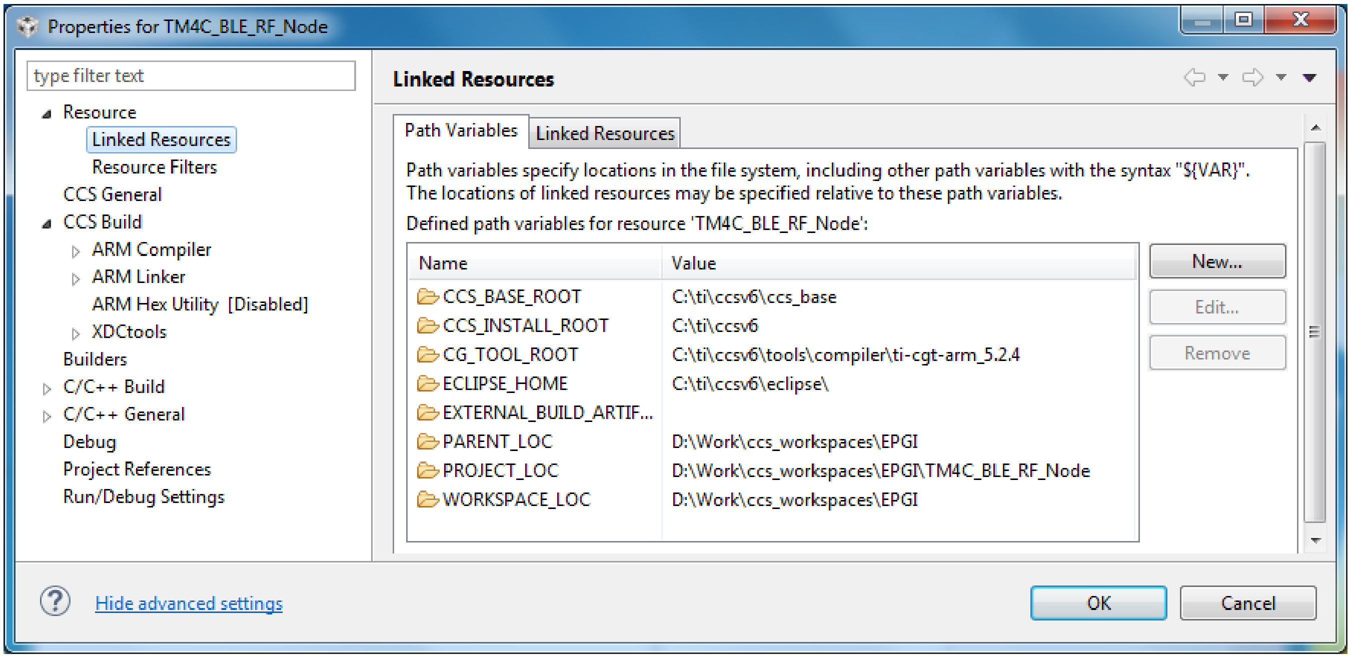


Path variables specify locations in the file system, including other path variables with the syntax "${VAR}".
The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_CC26xx_Demo_PeripheralStack':

Name	Value
CC26XXWARE	C:\ti\tirtos_simplelink_2_11_01_09\products\cc26xxware_2_20_06_14829
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIFACT	
ORG_PROJ_DIR	${ORIGINAL_PROJECT_ROOT}\..\..\IAR\Stack\CC2640
ORIGINAL_PROJECT_ROOT	C:\ti\simplelink\ble_cc26xx_2_00_00_42893\Projects\ble\TM4C_CC26xx_Demo_Peripheral\CC26xx\CCS\TM4C_CC26xx_Demo_PeripheralStack
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_CC26xx_Demo_PeripheralStack
TI_RTOS_DRIVERS_BASE	C:\ti\tirtos_simplelink_2_11_01_09\packages
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 14. Path Variables – CC2650 Peripheral Stack



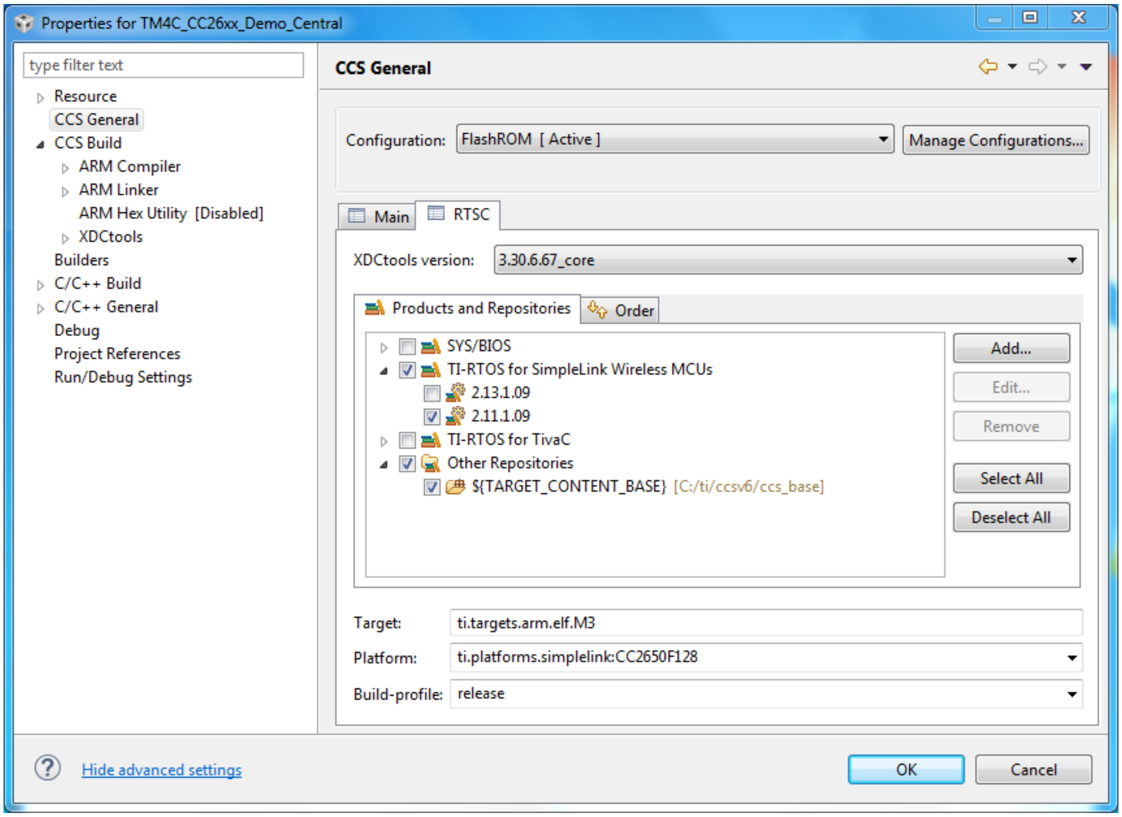
Path variables specify locations in the file system, including other path variables with the syntax "${VAR}". The locations of linked resources may be specified relative to these path variables.

Defined path variables for resource 'TM4C_BLE_RF_Node':

Name	Value
CCS_BASE_ROOT	C:\ti\ccsv6\ccs_base
CCS_INSTALL_ROOT	C:\ti\ccsv6
CG_TOOL_ROOT	C:\ti\ccsv6\tools\compiler\ti-cgt-arm_5.2.4
ECLIPSE_HOME	C:\ti\ccsv6\eclipse\
EXTERNAL_BUILD_ARTIF...	
PARENT_LOC	D:\Work\ccs_workspaces\EPGI
PROJECT_LOC	D:\Work\ccs_workspaces\EPGI\TM4C_BLE_RF_Node
WORKSPACE_LOC	D:\Work\ccs_workspaces\EPGI

Figure 15. Path Variables – TM4C Slave

5. Check the TI-RTOS version and platform selection (see Figure 16 and Figure 17).



Configuration: FlashROM [Active]

XDCtools version: 3.30.6.67_core

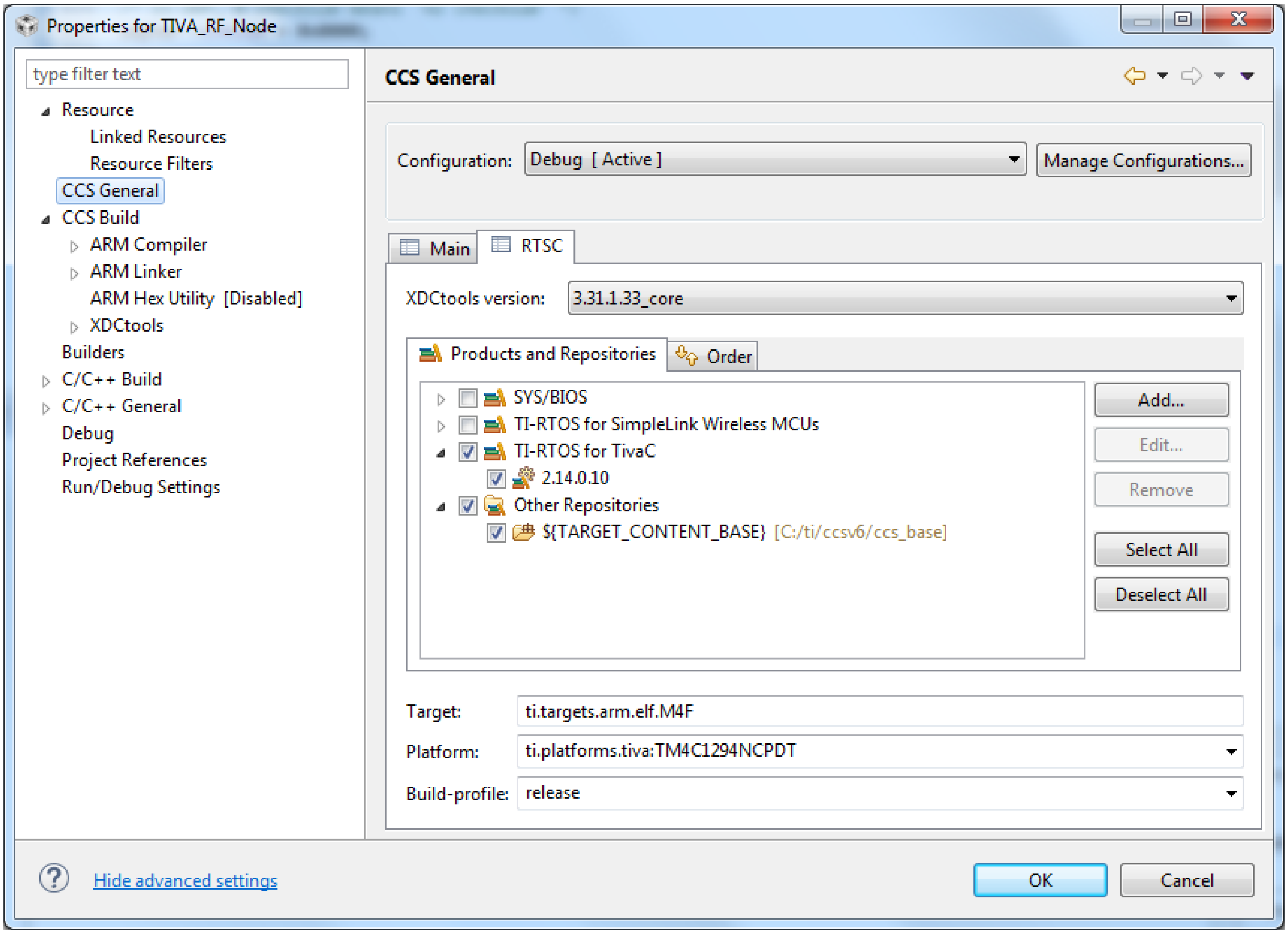
- □ SYS/BIOS
- ☑ TI-RTOS for SimpleLink Wireless MCUs
  - □ 2.13.1.09
  - ☑ 2.11.1.09
- □ TI-RTOS for TivaC
- ☑ Other Repositories
  - ☑ ${TARGET_CONTENT_BASE} [C:/ti/ccsv6/ccs_base]

Target: ti.targets.arm.elf.M3

Platform: ti.platforms.simplelink:CC2650F128

Build-profile: release

Figure 16. TI-RTOS Product Selection – CC2650



Properties for TIVA_RF_Node

CCS General

Configuration: Debug [Active]

XDCtools version: 3.31.1.33_core

- ☐ SYS/BIOS
- ☐ TI-RTOS for SimpleLink Wireless MCUs
- ☑ TI-RTOS for TivaC
  - ☑ 2.14.0.10
- ☑ Other Repositories
  - ☑ ${TARGET_CONTENT_BASE} [C:/ti/ccsv6/ccs_base]

Target: ti.targets.arm.elf.M4F

Platform: ti.platforms.tiva:TM4C1294NCPDT

Build-profile: release

Figure 17. TI-RTOS Product Selection – TM4C Slave

6. Demo executable list
   - TM4C:
     – Wired Master – TM4C_BLE_Master.out
     – RF Node Slave – TM4C_BLE_RF_Node.out
   - CC2650:
     – Central
       - TM4C_CC26xx_Demo_CentralStack.out
       - TM4C_CC26xx_Demo_Central.out
     – Peripheral RF Node
       - TM4C_CC26xx_Demo_PeripheralStack.out
       - TM4C_CC26xx_Demo_Peripheral.out

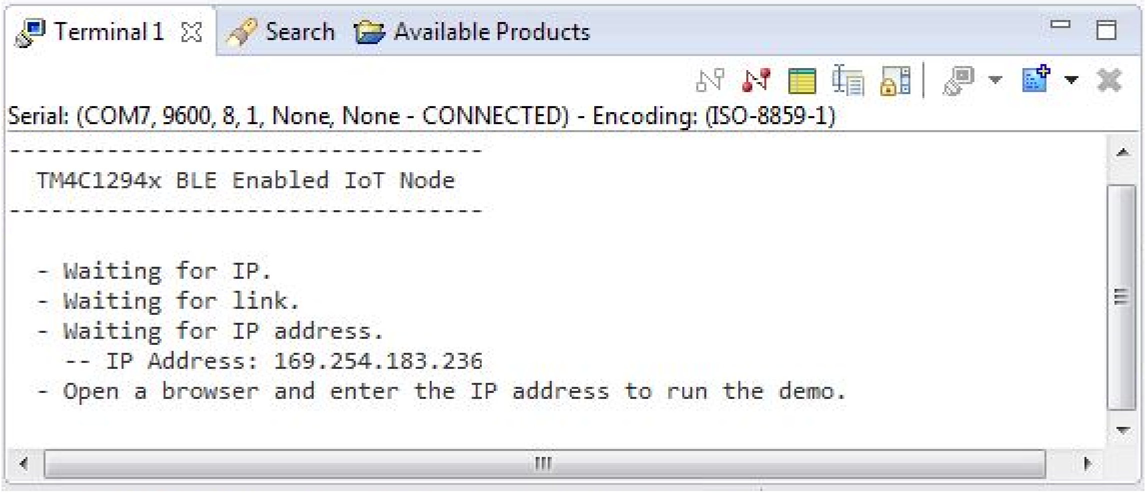
NOTE: <project_name>Stack.out must be be loaded before loading the <project_name>.out.

6 Demonstration Execution

For details about demonstration execution, see BLE-Enabled IoT Node With High-Performance MCU Reference Design.

6.1 Debug Port Setup for TM4C Wired Master Device

1. Open a terminal window (like Hyperterminal or TeraTerm) and connect to the Stellaris® Virtual Serial Port COM port corresponding to the TM4C-wired master device.
2. Select the baud rate as 9600, data bits as 8, parity as none, stop bits as 1, and flow control as none.
3. When the Ethernet cable is connected to TM4C-wired master device, the IP address is acquired and displayed on the debug terminal.



Serial: (COM7, 9600, 8, 1, None, None - CONNECTED) - Encoding: (ISO-8859-1)

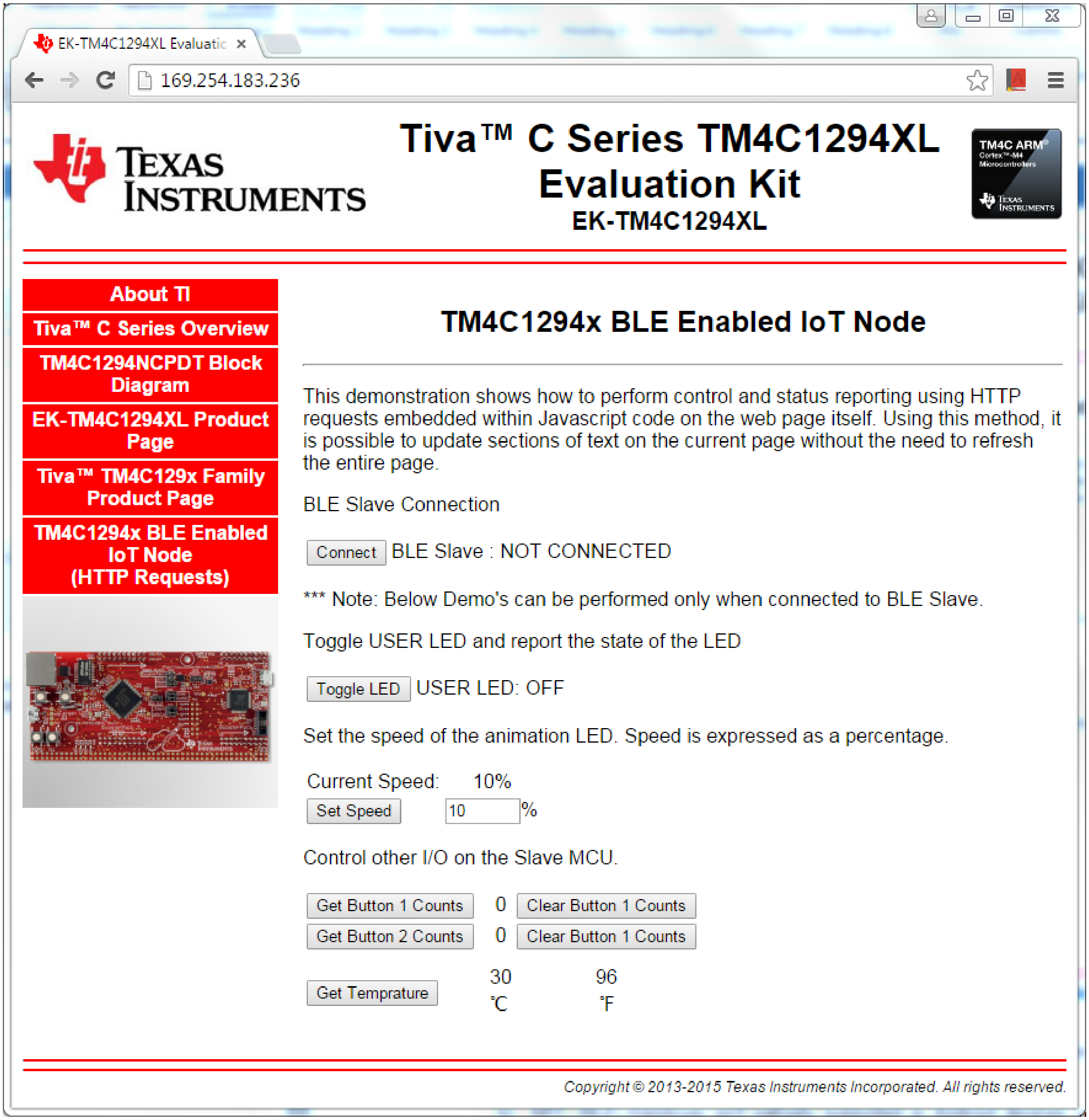
```
----------------------------------
  TM4C1294x BLE Enabled IoT Node
----------------------------------

  - Waiting for IP.
  - Waiting for link.
  - Waiting for IP address.
    -- IP Address: 169.254.183.236
  - Open a browser and enter the IP address to run the demo.
```

Figure 18. Debug Terminal

6.2 Demonstration Execution on the Web Page

1. Power on the peripheral RF node.
2. From a PC connected to the same network as the central node, open a browser and enter the assigned IP address to open the Web page.
3. Press Connect and wait until the status displays as CONNECTED (can take up to 8 seconds).
4. Demo 1: **Toggle LED** – Pressing Toggle LED toggles the LED1 on the Slave TM4C-Connected LaunchPad.
5. Demo 2: **LED Animation** – Controls the LED animation speed on the Slave TM4C. 0% is 1-s period and 100% is 20 ms.
6. Demo 3: **Get and Clear Button Press Count**
   - Get Button 1 Count displays the press count of the button SW1 on the slave TM4C board.
   - Clear Button 1 Count clears the press count of the button SW1 on the slave TM4C board.
   - Get Button 2 Count displays the press count of the button SW2 on the slave TM4C board.
   - Clear Button 2 Count clears the press count of the button SW2 on the slave TM4C board.
7. Demo 4: **Get Temperature** – Get Temperature gives the device junction temperature of the Slave TM4C microcontroller.



Tiva™ C Series TM4C1294XL Evaluation Kit

EK-TM4C1294XL

- About TI
- Tiva™ C Series Overview
- TM4C1294NCPDT Block Diagram
- EK-TM4C1294XL Product Page
- Tiva™ TM4C129x Family Product Page
- TM4C1294x BLE Enabled IoT Node (HTTP Requests)

TM4C1294x BLE Enabled IoT Node

This demonstration shows how to perform control and status reporting using HTTP requests embedded within Javascript code on the web page itself. Using this method, it is possible to update sections of text on the current page without the need to refresh the entire page.

BLE Slave Connection

Connect BLE Slave : NOT CONNECTED

*** Note: Below Demo's can be performed only when connected to BLE Slave.

Toggle USER LED and report the state of the LED

Toggle LED USER LED: OFF

Set the speed of the animation LED. Speed is expressed as a percentage.

Current Speed: 10%

Set Speed 10 %

Control other I/O on the Slave MCU.

Get Button 1 Counts 0 Clear Button 1 Counts

Get Button 2 Counts 0 Clear Button 1 Counts

Get Temprature 30 ℃ 96 ℉

Copyright © 2013-2015 Texas Instruments Incorporated. All rights reserved.

Figure 19. Tiva™ C Series TM4C1294XL Evaluation Kit

7 Resources

To download the resource files for this reference design, refer to http://www.ti.com/tool/TIDM-TM4C129XBLE.

8 References

1. *TivaWare for C Series:* http://www.ti.com/tool/SW-TM4C
2. *Stellaris In-Circuit Debug Interface (ICDI) and Virtual COM Port Driver Installation Instructions* (SPMU287)
3. *TI RTOS:* http://www.ti.com/tool/ti-rtos
4. *BLE Stack-2:* http://www.ti.com/tool/ble-stack-archive
5. *EK-TM4C1294XL LaunchPad:* http://www.ti.com/tool/ek-tm4c1294xl
6. *CC2650 Development Kit:* http://www.ti.com/tool/cc2650dk

IMPORTANT NOTICE FOR TI REFERENCE DESIGNS

Texas Instruments Incorporated ("TI") reference designs are solely intended to assist designers (“Buyers”) who are developing systems that incorporate TI semiconductor products (also referred to herein as “components”). Buyer understands and agrees that Buyer remains responsible for using its independent analysis, evaluation and judgment in designing Buyer’s systems and products.

TI reference designs have been created using standard laboratory conditions and engineering practices. **TI has not conducted any testing other than that specifically described in the published documentation for a particular reference design.** TI may make corrections, enhancements, improvements and other changes to its reference designs.

Buyers are authorized to use TI reference designs with the TI component(s) identified in each particular reference design and to modify the reference design in the development of their end products. HOWEVER, NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY THIRD PARTY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT, IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI components or services are used. Information published by TI regarding third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI REFERENCE DESIGNS ARE PROVIDED "AS IS". TI MAKES NO WARRANTIES OR REPRESENTATIONS WITH REGARD TO THE REFERENCE DESIGNS OR USE OF THE REFERENCE DESIGNS, EXPRESS, IMPLIED OR STATUTORY, INCLUDING ACCURACY OR COMPLETENESS. TI DISCLAIMS ANY WARRANTY OF TITLE AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, QUIET ENJOYMENT, QUIET POSSESSION, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS WITH REGARD TO TI REFERENCE DESIGNS OR USE THEREOF. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY BUYERS AGAINST ANY THIRD PARTY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON A COMBINATION OF COMPONENTS PROVIDED IN A TI REFERENCE DESIGN. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, SPECIAL, INCIDENTAL, CONSEQUENTIAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ON ANY THEORY OF LIABILITY AND WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, ARISING IN ANY WAY OUT OF TI REFERENCE DESIGNS OR BUYER’S USE OF TI REFERENCE DESIGNS.

TI reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All semiconductor products are sold subject to TI’s terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its components to the specifications applicable at the time of sale, in accordance with the warranty in TI’s terms and conditions of sale of semiconductor products. Testing and other quality control techniques for TI components are used to the extent TI deems necessary to support this warranty. Except where mandated by applicable law, testing of all parameters of each component is not necessarily performed.

TI assumes no liability for applications assistance or the design of Buyers’ products. Buyers are responsible for their products and applications using TI components. To minimize the risks associated with Buyers’ products and applications, Buyers should provide adequate design and operating safeguards.

Reproduction of significant portions of TI information in TI data books, data sheets or reference designs is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Buyer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products, and any use of TI components in its applications, notwithstanding any applications-related information or support that may be provided by TI. Buyer represents and agrees that it has all the necessary expertise to create and implement safeguards that anticipate dangerous failures, monitor failures and their consequences, lessen the likelihood of dangerous failures and take appropriate remedial actions. Buyer will fully indemnify TI and its representatives against any damages arising out of the use of any TI components in Buyer’s safety-critical applications.

In some cases, TI components may be promoted specifically to facilitate safety-related applications. With such components, TI’s goal is to help enable customers to design and create their own end-product solutions that meet applicable functional safety standards and requirements. Nonetheless, such components are subject to these terms.

No TI components are authorized for use in FDA Class III (or similar life-critical medical equipment) unless authorized officers of the parties have executed an agreement specifically governing such use.

Only those TI components that TI has specifically designated as military grade or “enhanced plastic” are designed and intended for use in military/aerospace applications or environments. Buyer acknowledges and agrees that any military or aerospace use of TI components that have ***not*** been so designated is solely at Buyer's risk, and Buyer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI has specifically designated certain components as meeting ISO/TS16949 requirements, mainly for automotive use. In any case of use of non-designated products, TI will not be responsible for any failure to meet ISO/TS16949.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265