Application Report
SPMA074A–November 2015–Revised April 2016

Implementation of Programmer for Serial Bootloaders on TM4C12x Microcontroller

Amit Ashara

ABSTRACT

TM4C12x microcontrollers from Texas Instruments feature ROM and Flash-based bootloaders. The bootloaders are used to download an application image to the microcontroller over universal asynchronous receiver/transmitter (UART), inter-integrated circuit (I2C), synchronous serial interface (SSI), universal serial bus (USB) or Ethernet (TM4C129x device family only) without the need to connect a JTAG or Serial Wire Debug (SWD) programmer. The Flash-based bootloaders allow the ability to customize the bootloader itself to perform additional operations.

The LMFlashProgrammer and the ICDI device, on the evaluation kits and evaluation modules (EVM's) for the TM4C12x microcontrollers, can be used to program a target device. However, the details of the LMFlashProgrammer and the ICDI device are not available.

This application report demonstrates how a programmer can be implemented by using the EK-TM4C123GXL LaunchPad to communicate with the serial bootloader (SBL) over UART, I2C or SSI serial peripherals of the target device with a windows application performing the role of the control software.

Project collateral and source code discussed in this document can be downloaded from the following URL: http://www.ti.com/lit/zip/spma074.

Contents

List of Figures

List of Tables

1 Introduction

The TM4C12x family of devices and TivaWare™ from Texas Instruments integrates serial bootloaders with the following features:

- ROM-based serial bootloader for UART0, I2C0, SSI0, USB0 and Ethernet (Ethernet is supported on TM4C129x device family only)
- Flash-based serial bootloader source code that is fully customizable by the end user for UART, I2C, SSI, USB, CAN and Ethernet (Ethernet is supported on TM4C129x device family only).
- Boot Configuration Register (BOOTCFG) that can be programmed by the user application to direct the core to execute the ROM bootloader even if the application resides in the flash
- Simple packet-based protocol with identical data and communication format (for UART, I2C and SSI interfaces) to download an application image to the TM4C12x devices

2 Description of Setup and Protocol

Figure 1 is referenced to illustrate the SBL setup and protocol. It is assumed that you are familiar with UART, I2C and SSI protocols.

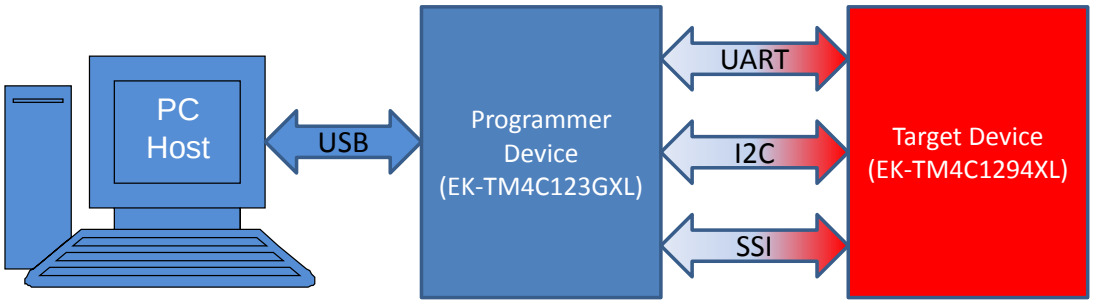

Figure 1. System Setup

2.1 Hardware Setup Description

There are three important components (see Figure 1):

- Target Device: This is the TM4C12x device to which a binary image needs to be programmed. In this application report, the target device is a TM4C1294NCPDT on the EK-TM4C1294XL Connected Launchpad. The target device runs the ROM or Flash-based bootloader.
- Programmer Device: This TM4C12x device runs an embedded application that performs the role of a protocol converter from PC to target device. It receives a binary image from a PC host and communicates with the target device over the serial interface to program the target device. In this application report, the Programmer Device is demonstrated by a TM4C123GH6PM on the EK-TM4C123GXL Launchpad.
- PC Host: The PC Host runs the applications that configure, control and download a binary image generated by an IDE to the target device by using services provided by the Programmer Device. Also, the PC Host can be used to update the embedded application of the Programmer Device. In this application report, the PC Host is a Windows 7 machine.

2.2 Device Bootloader Description

The TM4C12x bootloader are always available in ROM. A custom bootloader can still be programmed in the flash, so that additional features not provided by ROM-based bootloaders can be implemented in a customer's end system. Some of these features (not exhaustive) can be:

- Interface instances other than UART0, SSI0 and I2C0 for downloading application images.
- CRC check sum of the image
- Support for higher baud rates or serial clock rates

The ROM-based bootloader can be invoked by the application by calling the ROM_UpdateUART, ROM_UpdateI2C, ROM_UpdateSSI, ROM_UpdateUSB and ROM_UpdateEMAC or by using the BOOTCFG register.

To call the flash-based bootloader, the application must have a condition like a switch press that can be detected by the software or a software flag to call the SVCall handler. More details on the flash-based bootloader are available in the *TivaWare™ Bootloader User's Guide* (SPMU301).

2.3 Bootloader Protocol Description

The SBL uses a simple packet-based protocol to download the binary image. The target device is the slave executing the SBL. The Programmer Device is the master executing the control interface of the SBL. This is essential since the I2C and SSI in the target device are configured in slave mode by the ROM or flash based bootloader. The UART peripheral does not follow the concept of master and slave, but still it requires an external entity that can provide the packets of information and process the requests asynchronously.

2.3.1 Generic Packet Structure

The SBL uses a well defined packet structure for commands and responses to establish communication between the programmer and the target device. The size and response for each of the commands is fixed, so that the programmer and target can perform the most efficient communication.

Each command packet is made up of three fields, as shown in Figure 2.

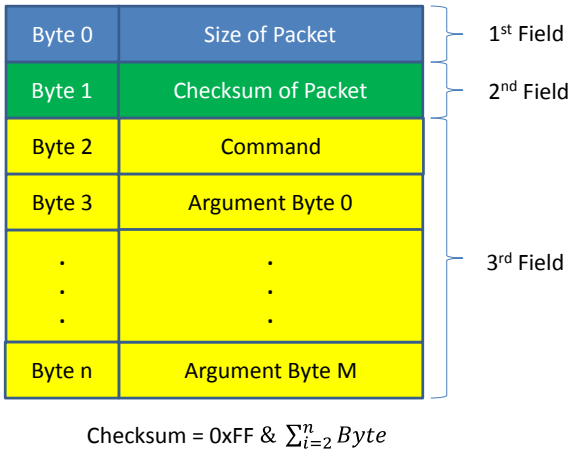

Figure 2. Generic Command Packet Structure

The first field of the packet is one byte that contains the size of the packet including the first field. The second field of the packet is also one byte and contains the checksum formed over the rest of the packet. The third field is the command followed by its arguments. This field can have a variable size based on the command and the arguments applicable to the specific command. This is described in Section 2.3.2.

2.3.2 Command Packet Structure

The following section describes each of the command packets, the intent of the command with respect to the SBL and the response it may generate. The response packet structure is described in Section 2.3.3.

- PING: The PING command is sent by the programmer to the target device to ascertain whether the target device is in bootloader mode or not. If the target device is not in bootloader mode it will not send a response, which causes the programmer to timeout. If the target device is in bootloader mode, it sends a response of ACK or NAK.

Byte 0	Size = 0x03
Byte 1	Checksum = 0x20
Byte 2	Command = 0x20

Figure 3. PING Command

- DOWNLOAD: The DOWNLOAD command is sent by the programmer to inform the target device about the address where the application image is to be programmed and the size of the application image. The target device confirms whether the start address to be programmed is a valid 32-bit word address and that the size of the image does not exceed the available flash size. Then, it sends an ACK, otherwise, it should respond with a NAK.

NOTE: Note that the checksum value is not specified as this should be computed per the equation in Figure 2 on Byte-2 to Byte-10 and placed in the Byte-1 location.

Byte 0	Size = 0x0B
Byte 1	Checksum = 0xXX
Byte 2	Command = 0x21
Byte 3	Program Address [31:24]
Byte 4	Program Address [23:16]
Byte 5	Program Address [15:8]
Byte 6	Program Address [7:0]
Byte 7	Program Size [31:24]
Byte 8	Program Size [23:16]
Byte 9	Program Size [15:8]
Byte 10	Program Size [7:0]

Figure 4. DOWNLOAD Command

- RUN: The RUN command is sent by the programmer to inform the target device to execute an application image as specified by the 32-bit argument that follows the command byte. If the address to execute is a valid address in the TM4C12x, then the target device should send an ACK. Then, it begins execution from the address specified, otherwise, it should respond with a NAK.

NOTE: Note that the checksum value is not specified as this should be computed per the equation in Figure 2 on Byte-2 to Byte-6 and placed in the Byte-1 location.

Byte 0	Size = 0x07
Byte 1	Checksum = 0xXX
Byte 2	Command = 0x22
Byte 3	Run Address [31:24]
Byte 4	Run Address [23:16]
Byte 5	Run Address [15:8]
Byte 6	Run Address [7:0]

Figure 5. RUN Command

- GET_STATUS: The GET_STATUS command is sent by the programmer to get the status of the last command issued to the target device. The target device responds by first sending an ACK response and then by sending a single byte packet with the current status code. On receiving the single byte response, the programmer sends an ACK.

Byte 0	Size = 0x03
Byte 1	Checksum = 0x23
Byte 2	Command = 0x23

Figure 6. GET_STATUS Command

- SEND_DATA: The SEND_DATA command is sent by the programmer to download the application image to the target device. The programmer device should always send data in multiples of 32-bit words. If the downloaded application binary overshoots the program size given by the DOWNLOAD command or if the checksum does not match, the target device will send a NAK, otherwise, it will respond with an ACK.

NOTE: Note that the size is not specified as this is dependent on the transfer size in the target device. Also the checksum value is not specified as this should be computed per the equation in Figure 2 on Byte-2 to Byte-N*4+10 and placed in Byte-1 location.

Byte 0	Size = 0xYY
Byte 1	Checksum = 0xXX
Byte 2	Command = 0x24
Byte 3	Data0 [7:0]
Byte 4	Data0 [15:8]
Byte 5	Data0 [23:16]
Byte 6	Data0 [31:24]
.	.
Byte N*4+3	DataN [7:0]
Byte N*4+4	DataN [15:8]
Byte N*4+5	DataN [23:16]
Byte N*4+6	DataN [31:24]

Figure 7. SEND_DATA Command

- RESET: The RESET command is sent by the programmer to the target device. The command instructs the bootloader to reset the target device to cause the new application to start from a reset. The target device sends an ACK to the programmer before executing the reset.

Byte 0	Size = 0x03
Byte 1	Checksum = 0x25
Byte 2	Command = 0x25

Figure 8. RESET Command

2.3.3 Response Packet Structure

The following section describes each of the response packets. There are two types of response packets generated. The first type of response packet (Type-1) is an ACK or NAK response, which is generated as a single byte packet when a command packet is sent. The second type of response packet (Type-2) is specific to the GET_STATUS command packet and has the same structure as that of a command packet.

- ACK Response packet: The ACK response packet is sent by either the target device in response to a command packet or by the programmer in response to the Type-2 response packet. It is generated when the command packet or a Type-2 packet has no checksum error and the parameters are defined correctly.

Byte 0	ACK = 0xCC

Figure 9. ACK Response

- NAK Response packet: The NAK response packet is sent by either the target device in response to a command packet or by the programmer in response to the Type-2 response packet. It is generated when the command packet or a Type-2 packet has either a checksum error or the parameters are not defined correctly.

Figure 10. NAK Response

- Type-2 response packet: The Type-2 response packet is sent by the target device to the programmer in response to the GET_STATUS command packet. The valid values for the Type-2 response packet are defined in Table 1. On receiving a Type-2 response packet from the target device, the programmer must send an ACK response packet.

  Note that the checksum value is not specified as this should be computed per the equation in Figure 2 on Byte-2 and placed in Byte-1 location.

Table 1. Type-2 Response Packet Value

Response Packet	Value	Comments
COMMAND_RET_SUCCESS	0x40	If the previous command was defined and format was correct
COMMAND_RET_UNKNOWN_CMD	0x41	If the previous command was an unknown command
COMMAND_RET_INVALID_CMD	0x42	If the previous command had a format error
COMMAND_RET_INVALID_ADR	0x43	If the previous download address was an invalid address
COMMAND_RET_FLASH_FAIL	0x44	If the attempt to program or erase the flash failed
COMMAND_RET_CRC_FAIL	0x45	If the CRC check of the image failed, if CRC check has been enabled

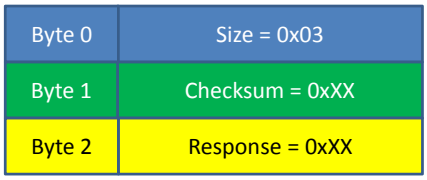

Figure 11. Type 2 Response

2.3.4 SBL Protocol Sequence

Having described the command and response packet types and structures, this section elaborates on the SBL protocol sequence that is used between the programmer and the target device to download and execute an application image. While the steps are identical for all three serial interfaces of UART, I2C and SSI, there is one additional step that is handled in UART called Auto-Baud. This step is done so that the programmer and target device can synchronize on the baud rate to be used.

Figure 12 through Figure 15 show how the programmer communicates with the target device to download the application image along with required error handling.

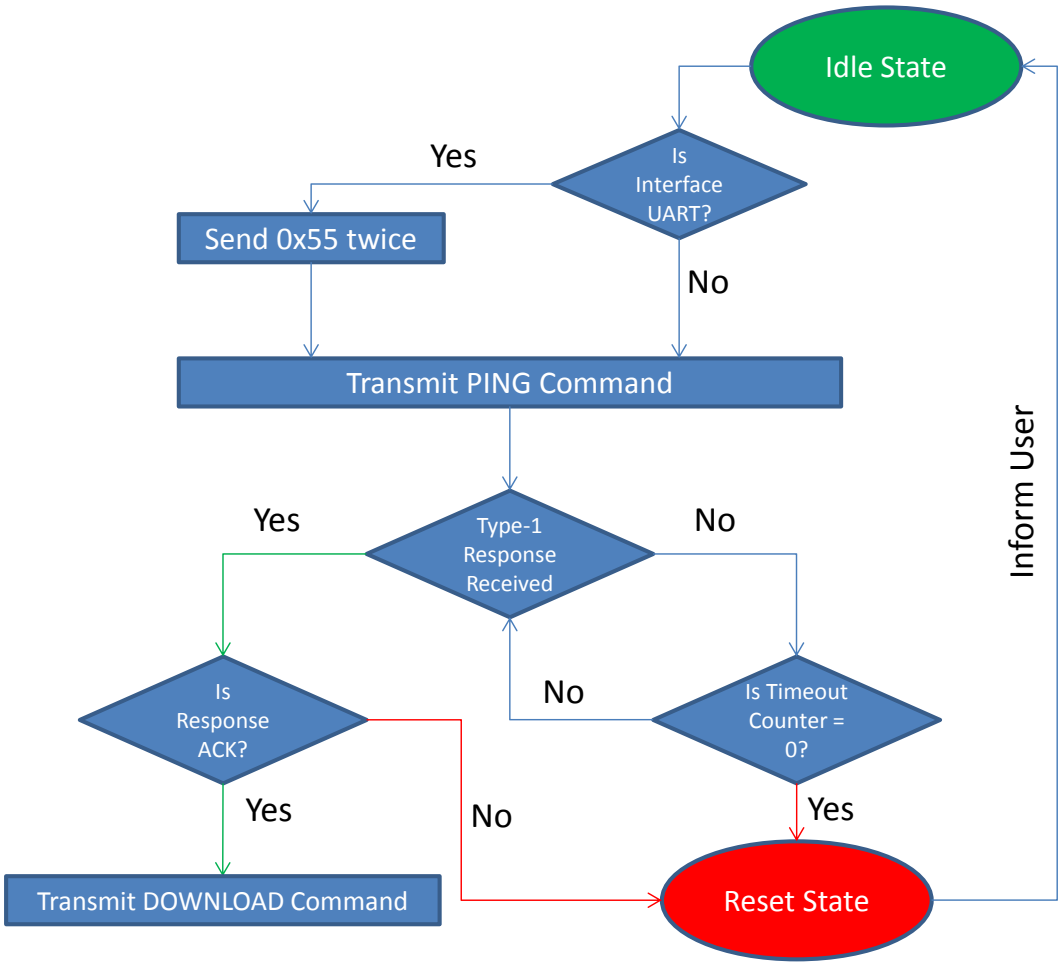

Figure 12. SBL Protocol Sequence-1

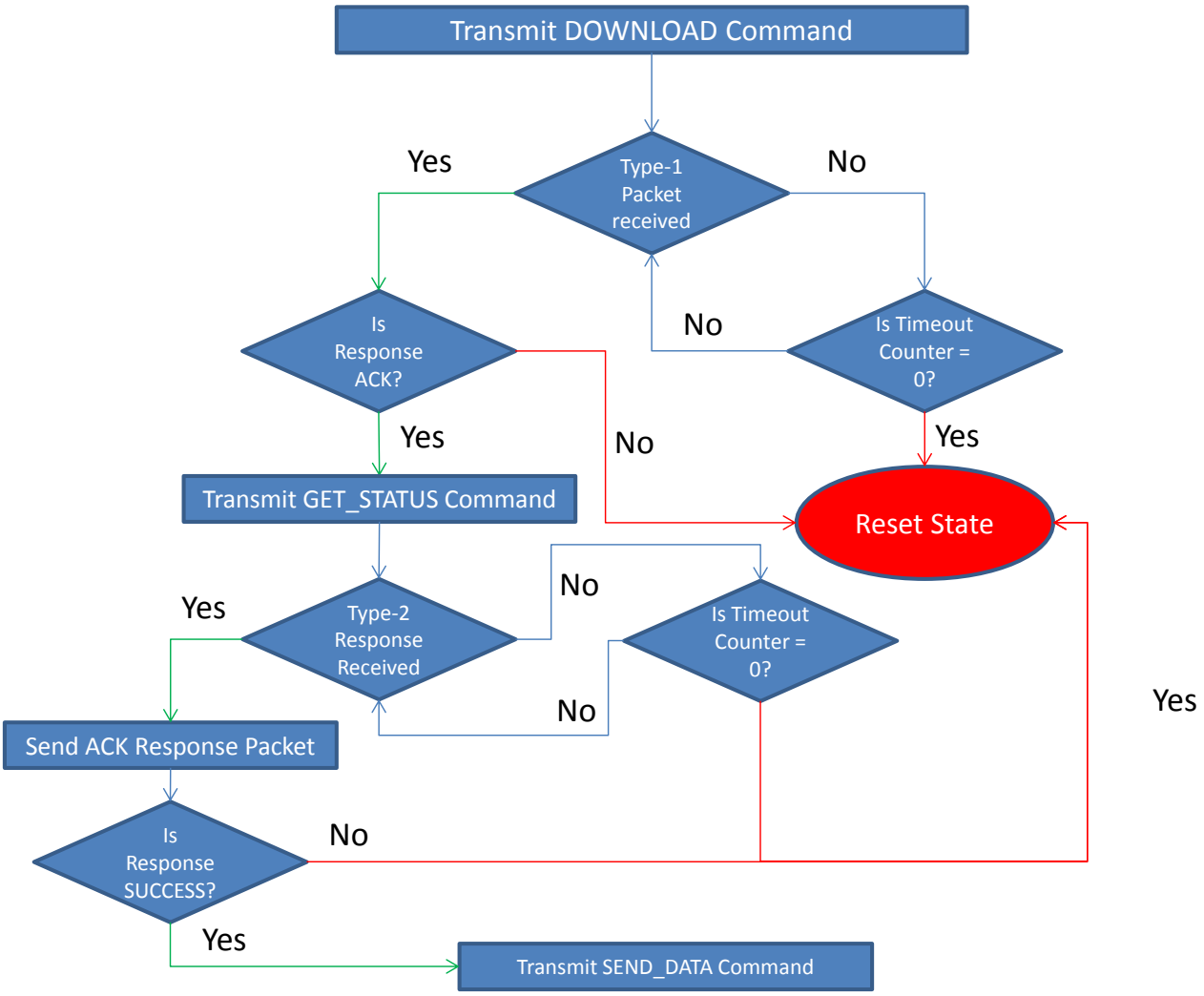

Figure 13. SBL Protocol Sequence-2

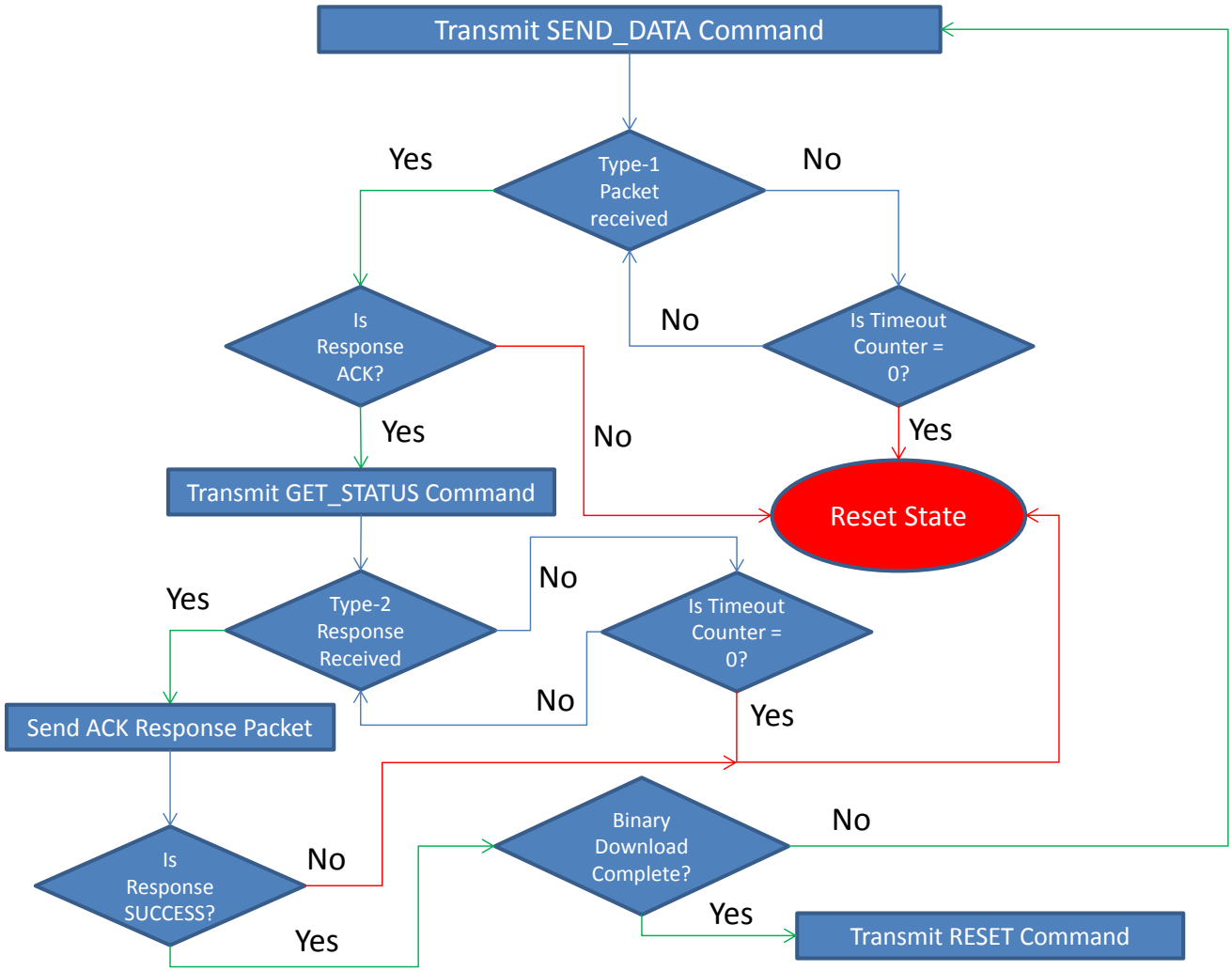

Figure 14. SBL Protocol Sequence-3

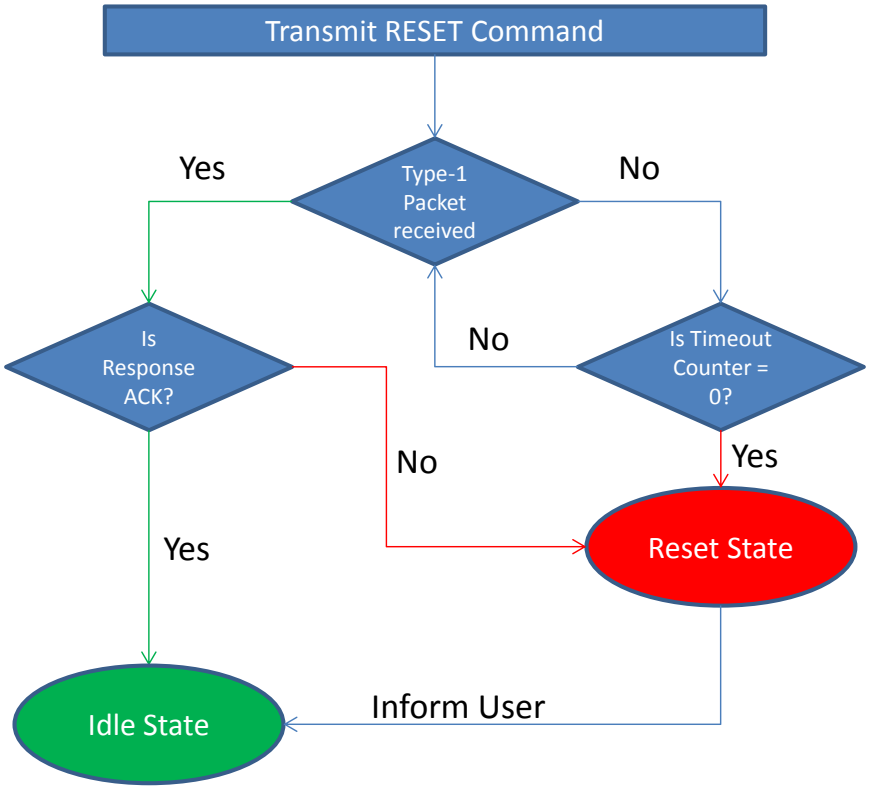

Figure 15. SBL Protocol Sequence-4

3 Getting Started Hardware

3.1 Programmer Board

As illustrated in Section 2.1, the programmer device is a TM4C123GH6PM on the EK-TM4C123GXL LaunchPad.

3.1.1 Pin Out Information

Table 2 shows the pinout from the EK-TM4C123GXL that is used to form the programming interface pins for a target device, status pins for LED indication and USB device function for a PC Host.

Table 2. Programmer Board Pin Out

Connector	Port Pin	Type	Direction	Peripheral	Comment
J1	PA6	Open Drain	Bidirectional	I2C-1	Connect to I2C Slave SCL Pin on Target Board
	PA7				Connect to I2C Slave SDA Pin on Target Board
J2	PA2	Push Pull	Output	GPIOA	Connect to BOOT Pin on Target Board
	PA3	Open Drain			Connect to RST_N Pin on Target Board
J3	PD0	Push Pull	Output	SSI-1	Connect to SSI Slave SCLK Pin on Target Board
	PD1		Output	SSI-1	Connect to SSI Slave FSS Pin on Target Board
	PD2		Input	SSI-1	Connect to SSI Slave MISO Pin on Target Board
	PD3		Output	SSI-1	Connect to SSI Slave MOSI Pin on Target Board
	PF1		Output	GPIOF	LED Red Indication on Programmer Board
	VBUS	Analog	Output	Power	5 V Supply to be Connected to Target Board
J4	PC4	Push Pull	Input	UART-1	Connect to UART TX Pin on Target Board
	PC5		Output	UART-1	Connect to UART RX Pin on Target Board
	PF2		Output	GPIOF	LED Blue Indication on Programmer Board
	PF3		Output	GPIOF	LED Green Indication on Programmer Board
J9	PD4	Analog	Bidirectional	USB-0	USB DM-DP Signals for PC Host Communication
	PD5				

3.1.2 Basic Setup Instructions

To setup the communication and control for the Programmer Device the following assumption has been made.

1. It is assumed that the programmer board is reset to factory condition.

NOTE: If not, reset the device to factory condition by using “Debug Port Unlock” utility in “LM Flash Programmer” tool from Texas Instruments.

2. Configure the board to perform the role of a programmer. This can be done by moving the “Power Select” switch SW3 to Device side, as shown in Figure 16.

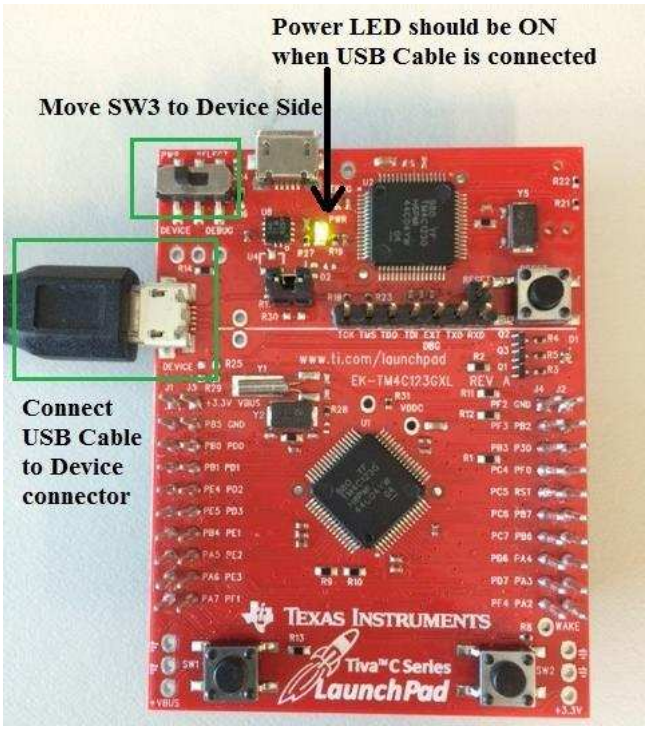

Figure 16. Programmer Board Setup

3.2 Target Board Setup

As illustrated in Section 2.1, the target device is a TM4C1294NCPDT on the EK-TM4C1294XL Connected LaunchPad.

3.2.1 Pin Out Instructions

Table 3 shows the pin out on the EK-TM4C1294XL that is used to communicate with the programmer for SBL interface, boot pin control and reset control.

Table 3. Target Board Pin Out

Connector	Port Pin	Type	Direction	Peripheral	Comment
JP1 Pin 2	VBUS	Analog	Input	Power	5 V Supply to be Connected From Programmer Board
JP4 Pin 2	PA0	Push Pull	Input	UART-0	Target Device UART Receive Pin
JP5 Pin 2	PA1	Push Pull	Output	UART-0	Target Device UART Transmit Pin
X8	PB2	Open Drain	Bidirectional	I2C-0	Target Device I2C Slave SCL Pin
X8	PB3	Open Drain	Bidirectional	I2C-0	Target Device I2C Slave SDA Pin
X7	PA2	Push Pull	Input	SSI-0	Target Device SSI Slave SCLK Pin
X7	PA3	Push Pull	Input	SSI-0	Target Device SSI Slave FSS Pin
X6	PA4	Push Pull	Output	SSI-0	Target Device SSI Slave MISO Pin
X6	PA5	Push Pull	Input	SSI-0	Target Device SSI Slave MOSI Pin
X6	PB4	Push Pull	Input	GPIOB	Boot Pin for Target Device
X7	RST_N	Pull Up	Input	NONE	Reset Pin for Target Device

3.2.2 Basic Setup Instructions

To setup the communication and control for the Target Device the following assumption has been made:

1. It is assumed that the target board is reset to factory condition.

NOTE: If not, reset the device to factory condition by using “Debug Port Unlock” utility in “LM Flash Programmer” tool from Texas Instruments. Since all settings on the EK-TM4C1294XL should be erased; make a note of the MAC Address for the EK-TM4C1294XL before running the “Debug Port Unlock” utility.

2. Configure the board to perform the role of a target, as shown in Figure 17.
3. Remove the Power Shunt from JP1 header.
4. Remove the ICDI UART0 Shunt from JP4 and JP5 header.

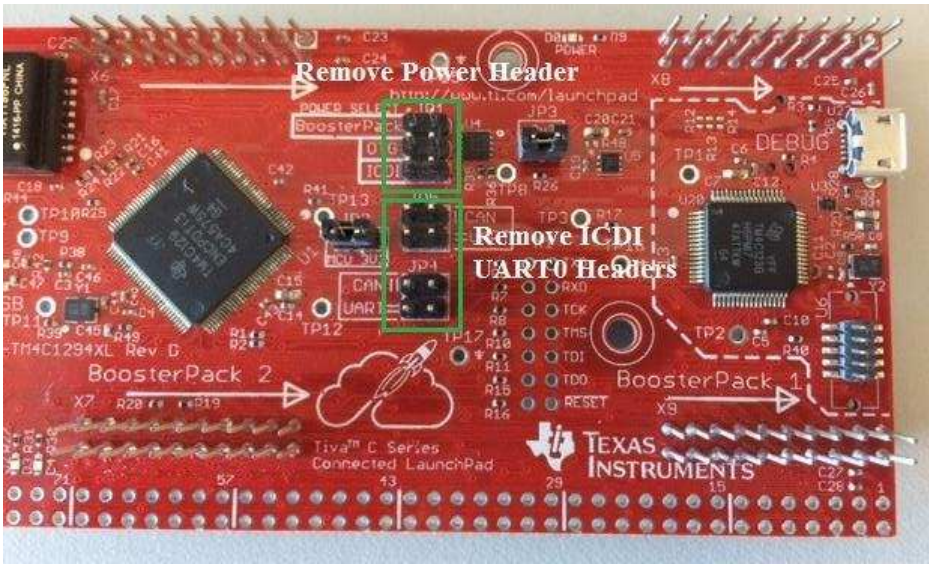

Figure 17. Target Board Setup

3.3 Connecting Using a Connector Board

3.3.1 Schematic for Connector Board on EK-TM4C123GXL

The schematic of the connector board that can be mounted on the EK-TM4C123GXL booster pack header is shown in Figure 18.

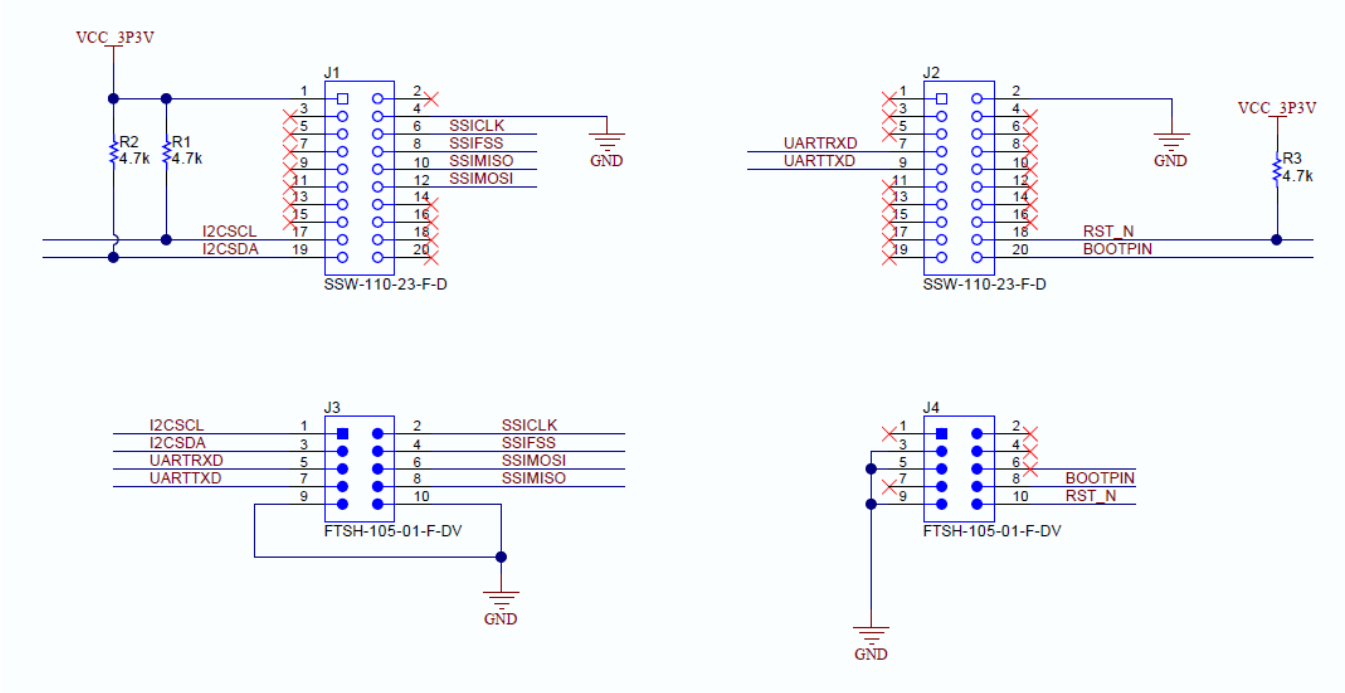

Figure 18. Schematic of EK-TM4C123GXL Connector Board

3.3.2 Schematic for Connector Board on EK-TM4C1294XL

The schematic of the connector board that can be mounted on the EK-TM4C1294XL bread board connector is shown in Figure 19.

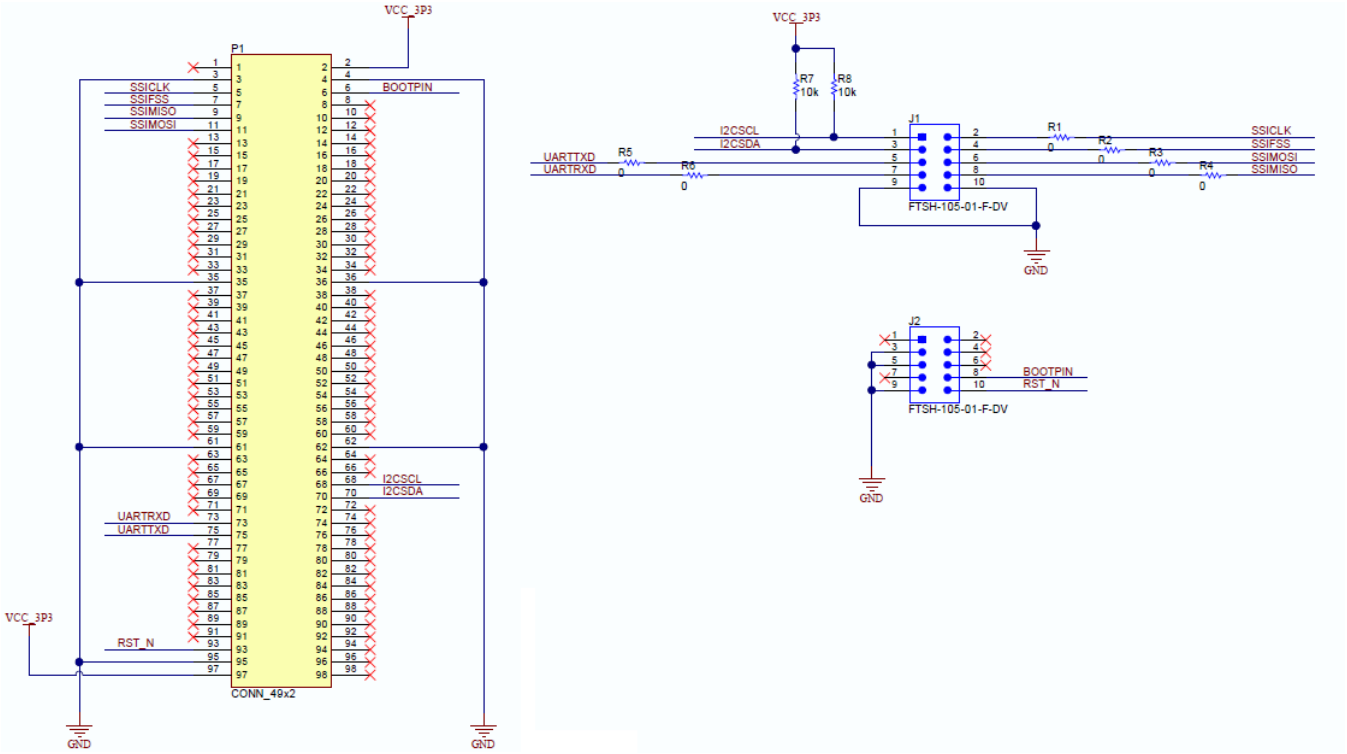

Figure 19. Schematic of EK-TM4C1294XL Connector Board

3.3.3 Connecting the Power, Control and Data Pins

Figure 20 shows the connection that needs to be made between the connector boards to interface the UART, I2C, SSI pins and the control pins for reset and boot mode. The setup uses two 10-pin 50 mil headers with a 10 wire flat ribbon cable. The GND is made is common by the ribbon cable. Only the 5V power pin needs to be connected, which can be done as given in Section 3.4.1 or using an external 5V supply for the EK-TM4C1294XL. When the EK-TM4C1294XL is supplied 5 V externally (either via a power supply or by connecting USB to the debug connector), the EK-TM4C123GXL does not need to provide a 5 V supply.

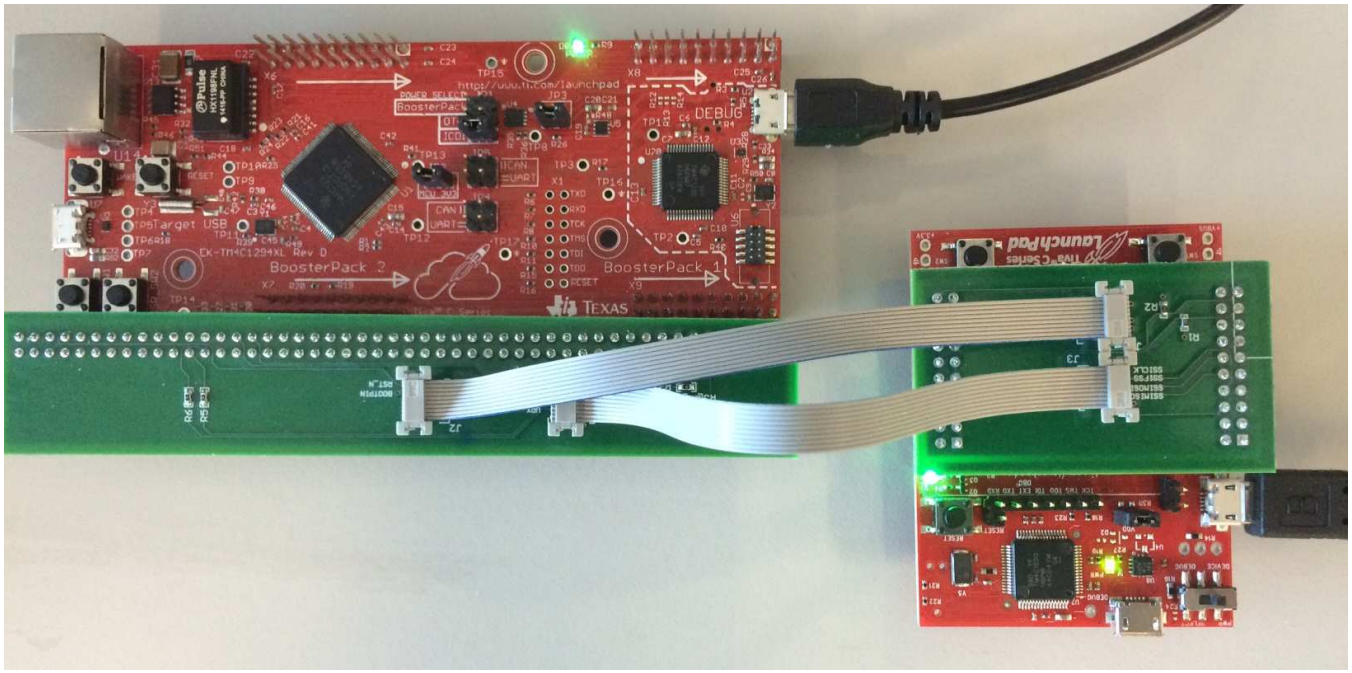

Figure 20. Connector Board Setup

3.4 Connecting Using Direct Wires

3.4.1 Connecting Power and Control Signals

The next step is to setup the common control signals between the target board and the programmer board, as shown in Figure 21.

- 5 V power from programmer board to target board
- Common GND between the programmer board and target board
- RST_N signal from the programmer board to target board
- Boot Control signal from the programmer board to target board

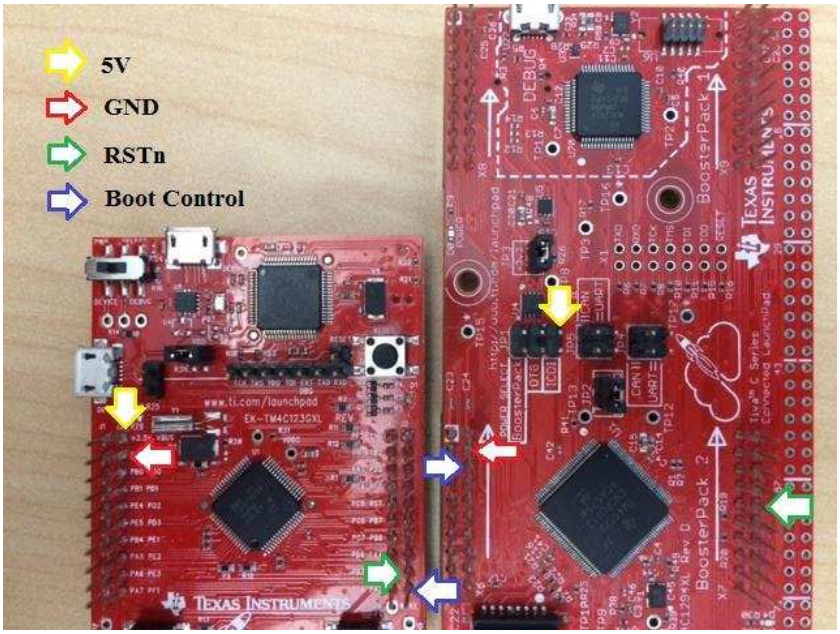

Figure 21. Power and Control Signal Setup

3.4.2 Connecting UART SBL Pins

If the SBL interface to be used is UART, connect the UART transmit and receive pins between the target board and the programmer board, as shown in Figure 22.

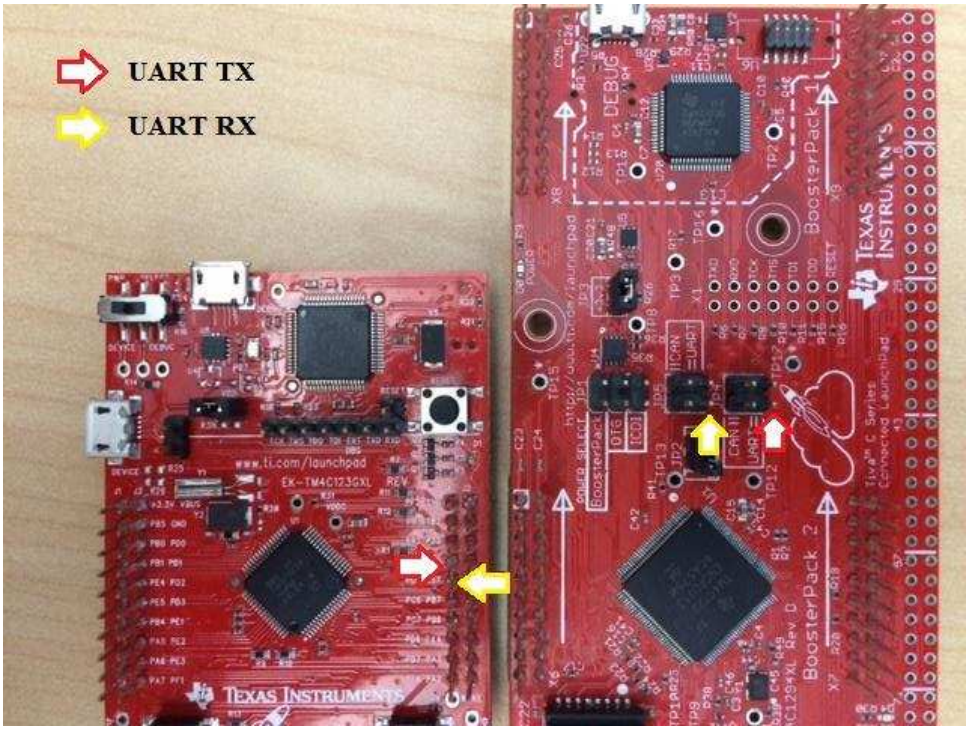

Figure 22. UART SBL Pin Setup

3.4.3 Connecting SSI SBL Pins

If the SBL interface to be used is SSI, connect the SSI Serial Clock (SCLK), Frame Select (FSS), SSI master transmit/slave receive (MOSI) and SSI master receive/slave transmit (MISO) pins between the target board and the programmer board, as shown in Figure 23.

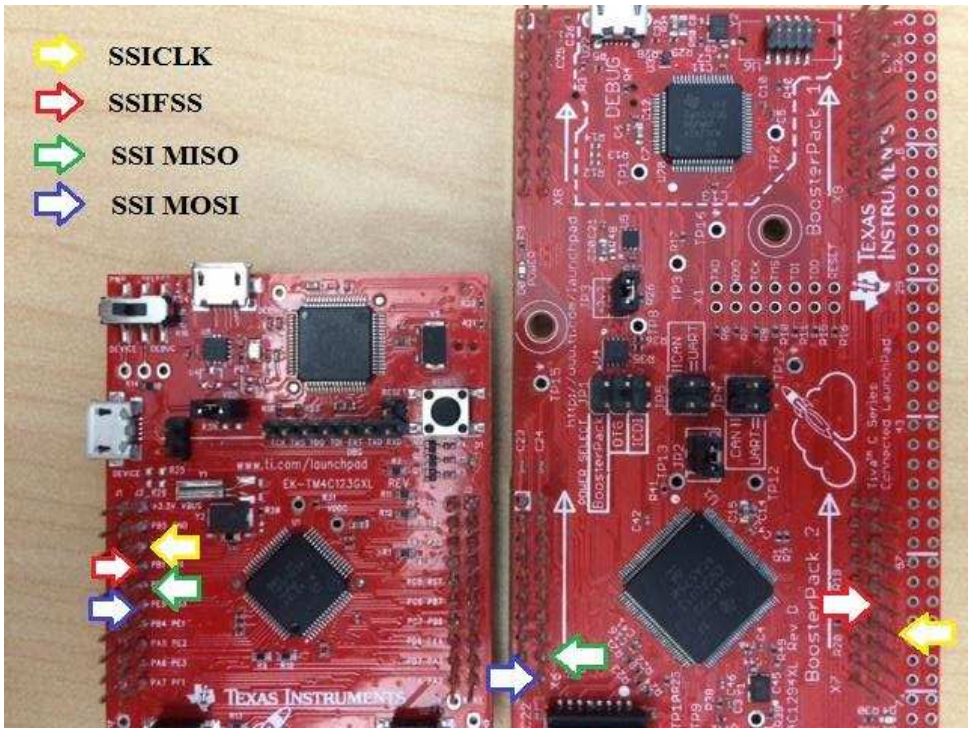

Figure 23. SSI SBL Pin Setup

3.4.4 Connecting I2C SBL Pins

If the SBL interface to be used is I2C, connect the I2C serial clock (SCL) and serial data (SDA) pins between the target board and the programmer board, as shown in Figure 24. Note that the pull up required for I2C Bus communication is provided using the weak pullup in the I2C I/O on the programmer device. This is just sufficient for the I2C communication when the line capacitance is not too high. Stronger pull may be needed for higher speed or if the line capacitance is larger in the end user application.

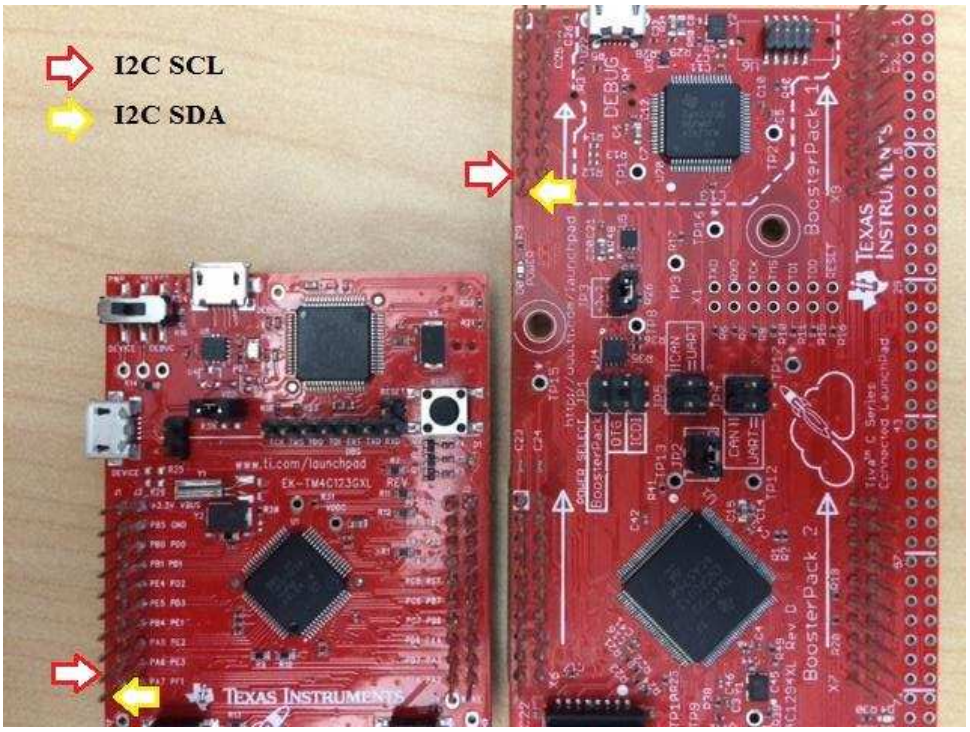

Figure 24. I2C SBL Pin Setup

4 Getting Started Software

Before installing the associated software that comes along with the application report, TivaWare for C Series software from Texas Instruments must be installed.

The serial bootloader software installer installs the following data base under “C:\Program Files (x86)\Texas Instruments\TM4C\TM4CSBL-1.0”.

- “bin” directory containing the “dfuprog.exe” application
- “examples” directory containing the source and binary for the programmer board and demo examples for the target board.
- “Serial Bootloader” directory containing the source for PC Host Application that can be imported updated and compiled in Visual Studio 2015
- “windows_drivers” directory containing the INF and CAT files to register the device drivers for the programmer board
- “Serial Bootloader.exe” binary as programmer user interface for existing examples.

4.1 Installing the Software

1. After downloading the serial bootloader software on the Windows PC, right click the .exe file and “Run as administrator”. You will be prompted by the installer, as shown in Figure 25. Click “Next”.

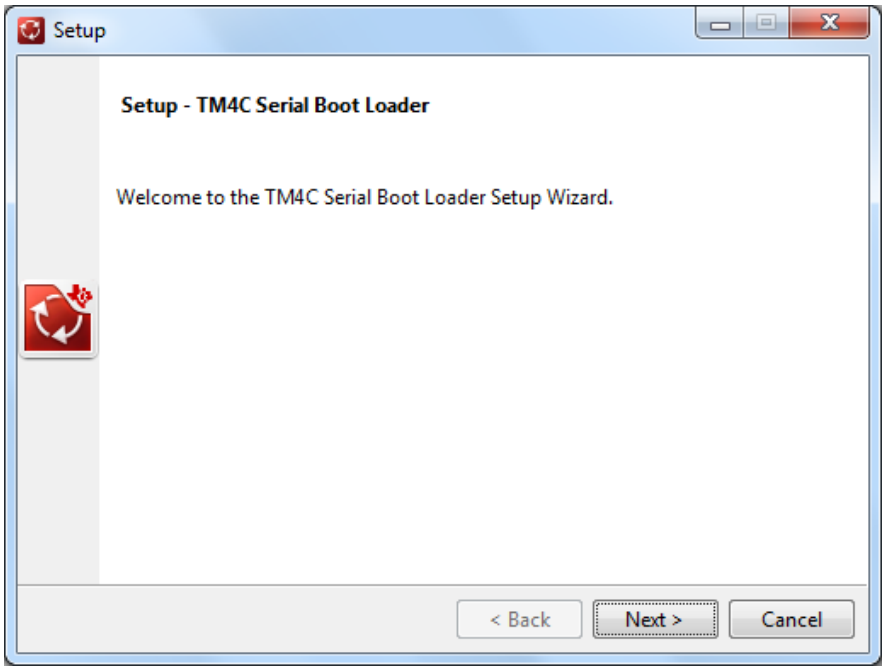

Figure 25. Software Installation Step-1

2. Accept the “License Agreement” and click “Next”. You will be prompted to the “Installation Directory” by the installer. Use the default path for installation or select another installation directory and click on “Next”.

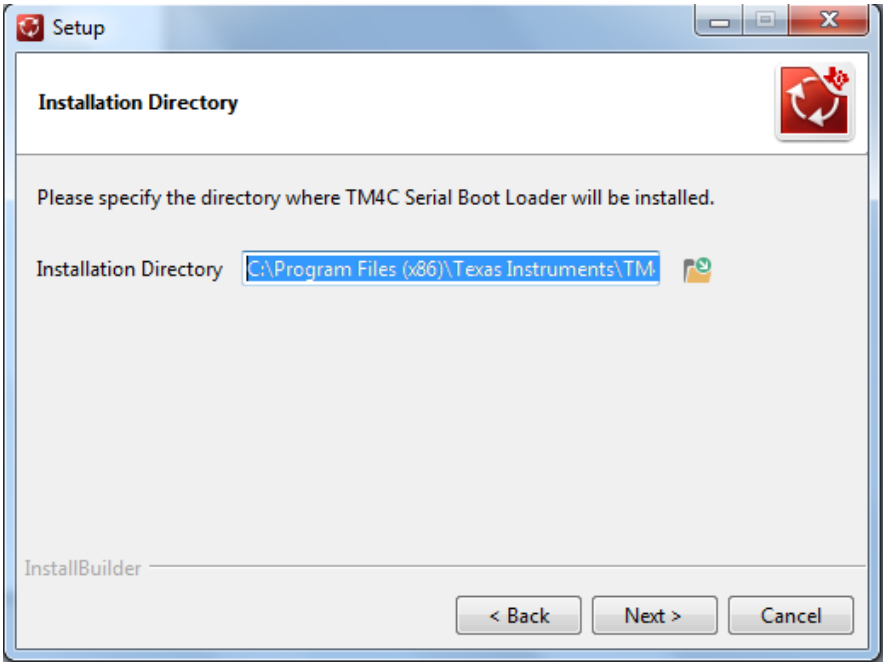

Figure 26. Software Installation Step-2

3. Once the path is setup, the installer is “Ready to Install”. Click “Next”.

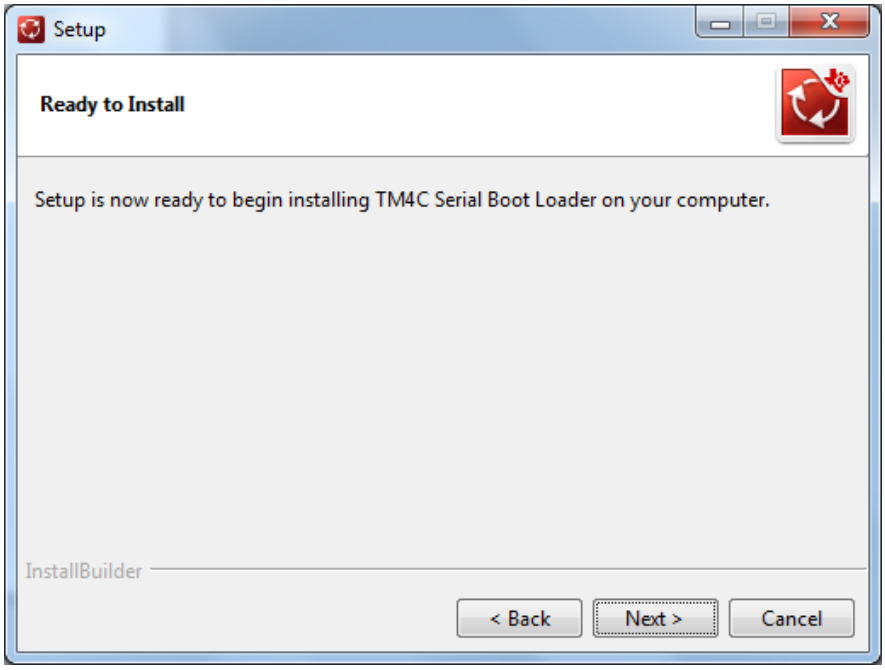

Figure 27. Software Installation Step-3

4. It may take up to two minutes for the installation to complete. Click “Finish” to complete the installation. The PC Host UI application can be accessed from “Programs” → “Texas Instruments” → “TM4C Serial Bootloader”.

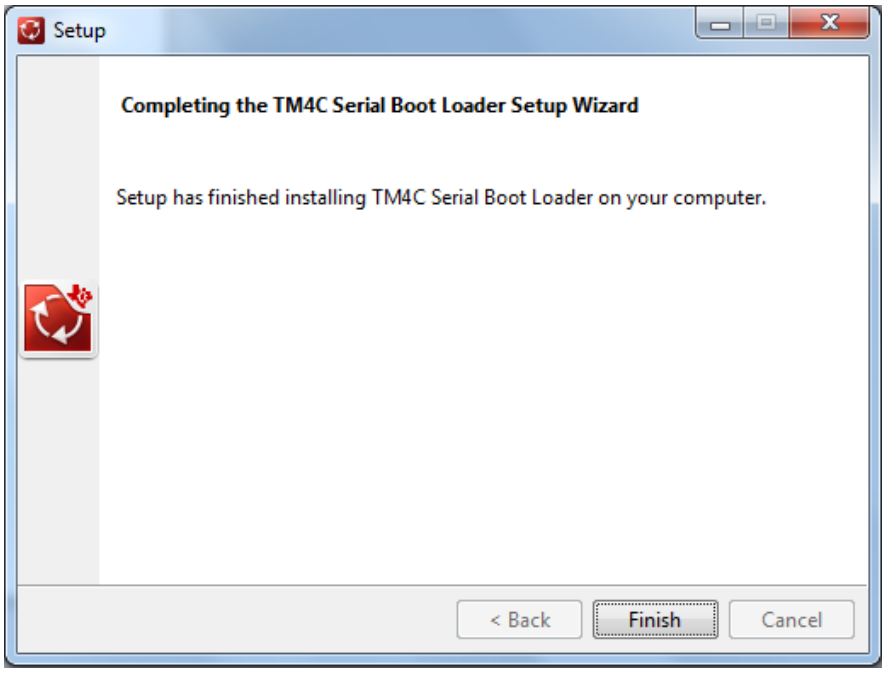

Figure 28. Software Installation Step-4

4.2 Example Software

4.2.1 Programmer Board Example Code

After the installation, the programmer board example code is kept under “examples\ek-tm4c123gxl”. The example code is called usb_dev_sbl and utilizes the USB bulk device driver to interface to the PC. The main code file usb_dev_sbl.c contains the SBL master code. The code has been written as a modular function so that you can modify the code to use with a SD card instead of a PC to function as a programmer.

4.2.2 Target Board Example Codes

After the installation, the target board example code is kept under “\examples\ek-tm4c1294xl\”. There are four examples:

- tm4c_all_led_blink: This example is used to blink two LED’s in a simultaneous mode.
- tm4c_alt_led_blink: This example is used to blink two LED’s in an alternating mode.
- tm4c_device_recovery_code: This example is used to program the BOOTCFG register to make GPIO PB4 as a Boot Configuration Pin. When the pin is asserted low and RST_N is asserted, the target device is forced into ROM bootloader mode.
- tm4c_faulty_code: This example shows how a faulty code downloaded through the SBL (that may make the device inoperable) can be recovered by the SBL programmer using the GPIO PB4 and ROM bootloader.

NOTE: The example codes have been written for both TM4C123 and TM4C129 class of devices. However, by default the TM4C129 is defined for the application demonstration.

4.2.3 Serial Bootloader

After the installation, the Visual Studio source code for the PC Host Application is installed under “Serial Bootloader”. It contains the UI application code and the functions associated with the buttons and edit boxes. The source code can be modified. Note that the lmsubdll.lib is required for linking the USB function calls; this can be imported from the TivaWare software installation.

4.2.4 Window Drivers

After installation, the window INF and CAT files for enumerating the programmer board are installed under the “window_drivers”, as shown in Section 5.

4.2.5 Device Firmware Upgrade (DFU) Programmer

After installation, the “dfuprog.exe” should be placed in the “bin” directory. Use this utility to download and update the programmer board application “usb_dev_sbl”. The source code for the DFU Programmer is already available as part of TivaWare software installation, as shown in Section 5.

5 Test Setup

Once the required connections have been made (see Section 3) and the software installed (see Section 4), use the hardware setup and example software to evaluate the SBL.

5.1 Setting Up the Programmer

Use the following steps to setup the Programmer Device.

1. Connect the programmer board USB device to the PC Host. Only the Power ON LED is lit on the programmer board. On the PC Device Manager, the programmer device enumerates the “Stellaris Device Firmware Upgrade”, as shown in Figure 29.

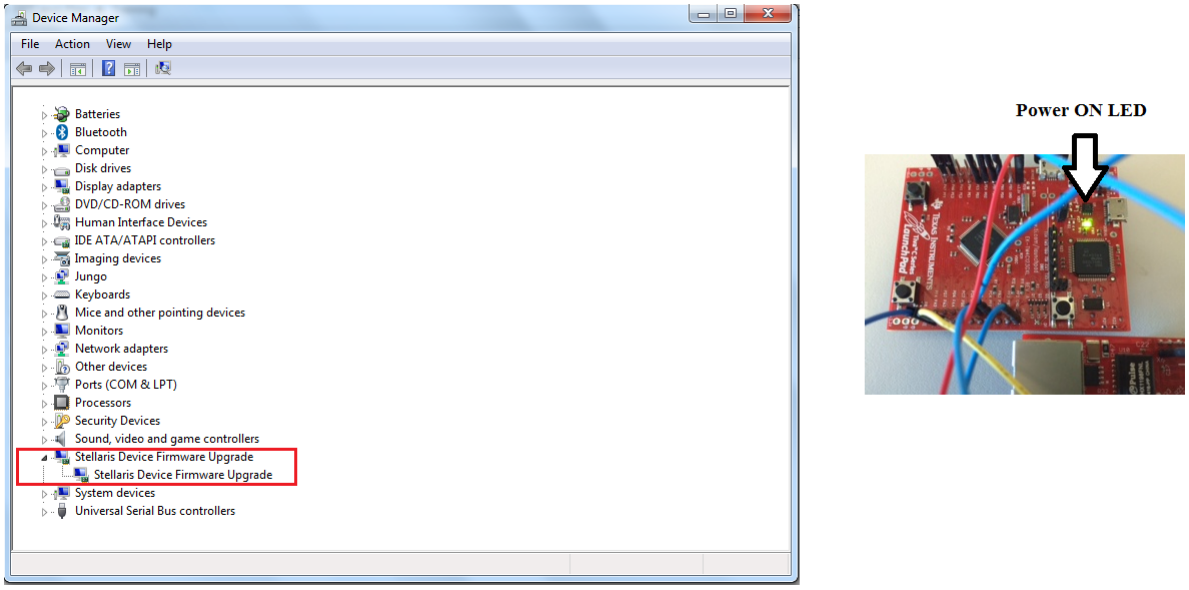

Figure 29. Programmer Default State

2. Open a command window and go to the installation directory folder where the “dfuprog.exe” is kept. Run the command “dfuproge.exe –e” to show all the USB devices connected to PC host as the DFU class device. Since there may be multiple DFU class devices, make a note of the device number of the programmer board, as shown in Figure 30.

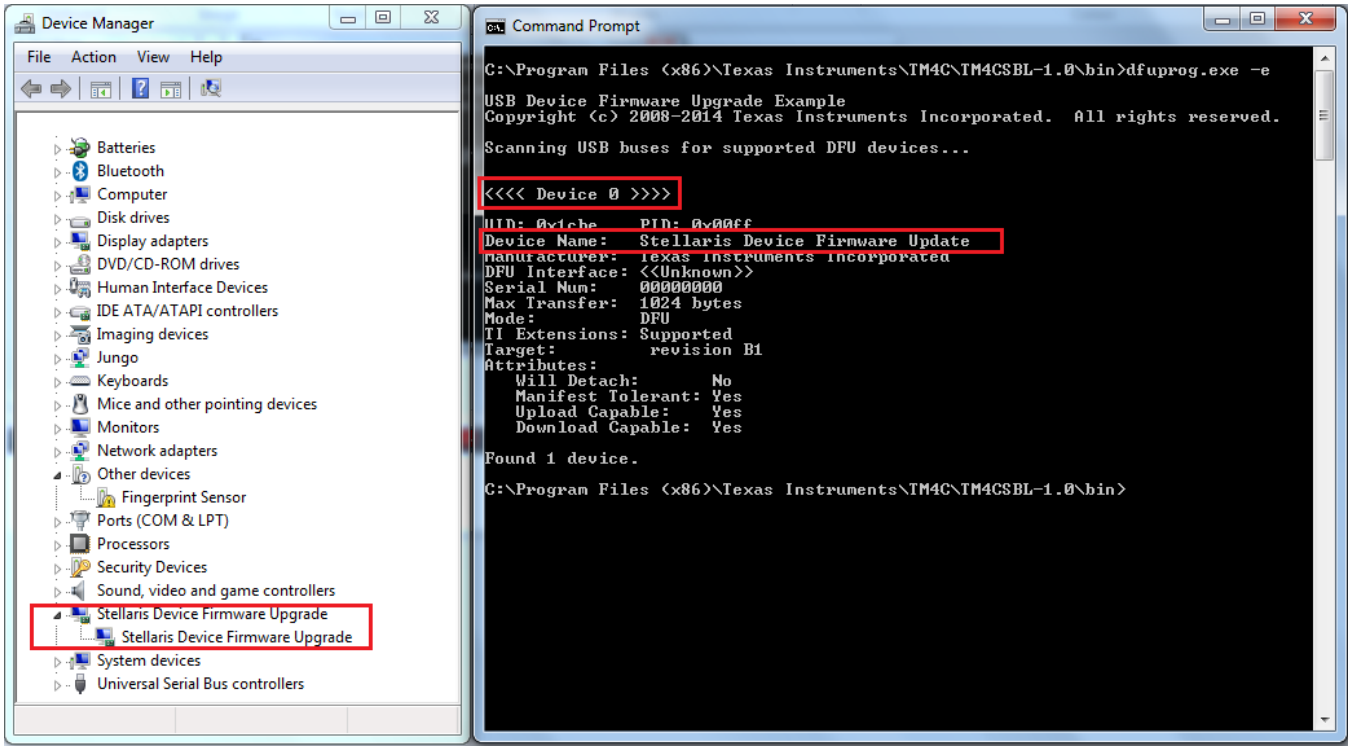

Figure 30. Getting the DFU Device Number

3. Upload the programmer application image, using the USB DFU, once the device number is known for the programmer board.
4. In the command window, run the command “dfuprog.exe –i <DEV NUMBER> -f ../examples/ek-tm4c123gxl/usb_dev_sbl/Debug/usb_dev_sbl.bin –a 0x0 –r”. This step uploads the programmer application image to the programmer board, as shown in Figure 31. There must not be any error message in the command window.

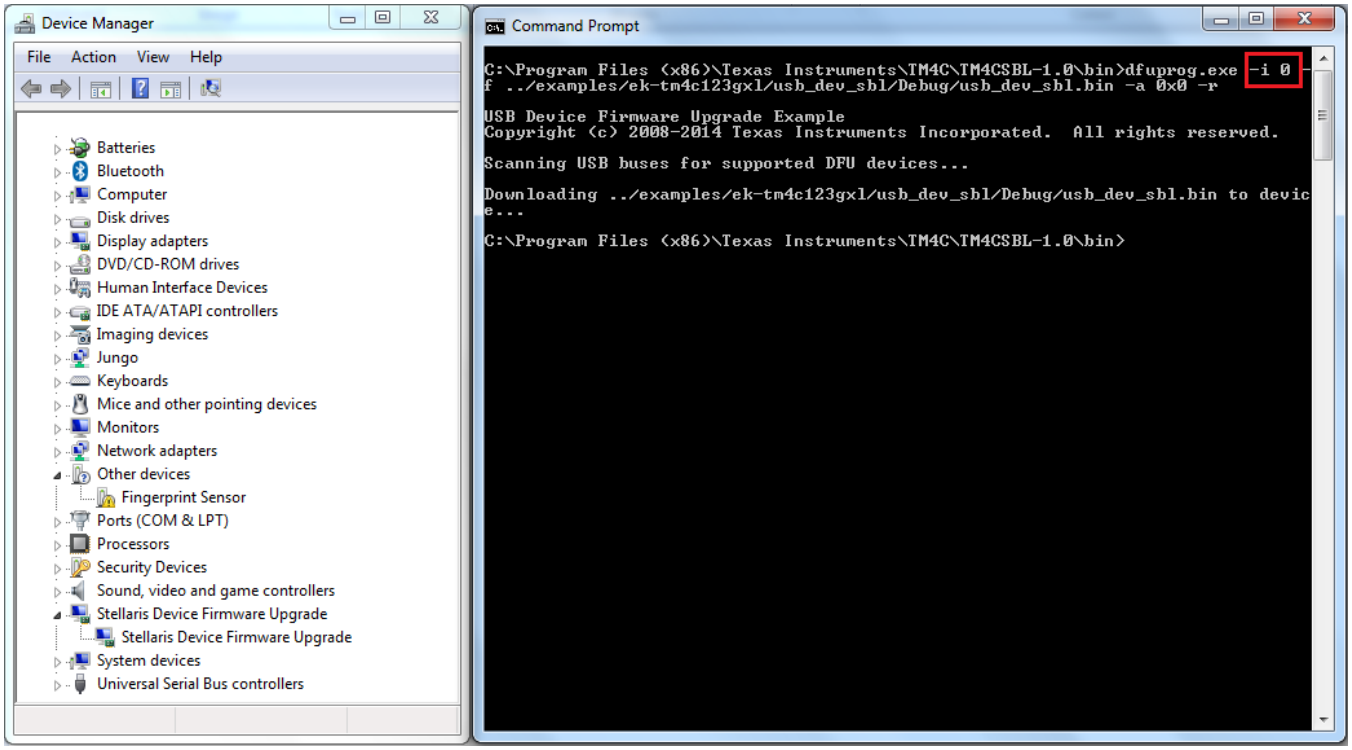

Figure 31. Downloading Programmer Application Image

5. After the successful download of the programmer application image, the PC re-enumerates the USB device and the device manage shows “TM4C Serial Bootloader”. Note that the device drivers have not yet been updated, so it is shown as “Other devices” with a “?” in the icon. Also after the last step of downloading the programmer application image, the RED LED on the board is lit as well. This is to indicate that while the firmware is running successfully on the programmer board, the USB device has not been enumerated yet.

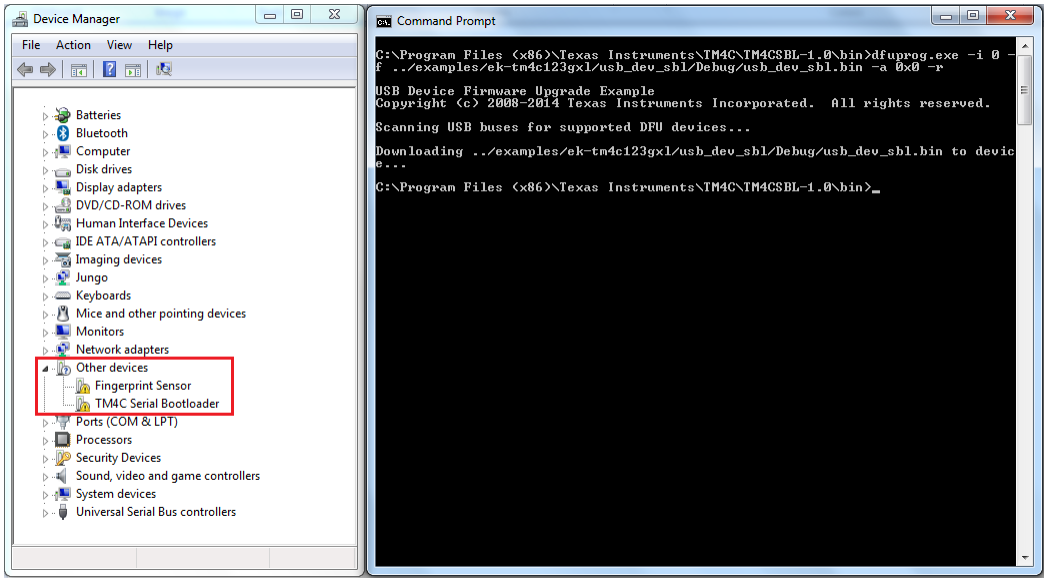

Figure 32. Device Manager View After Download

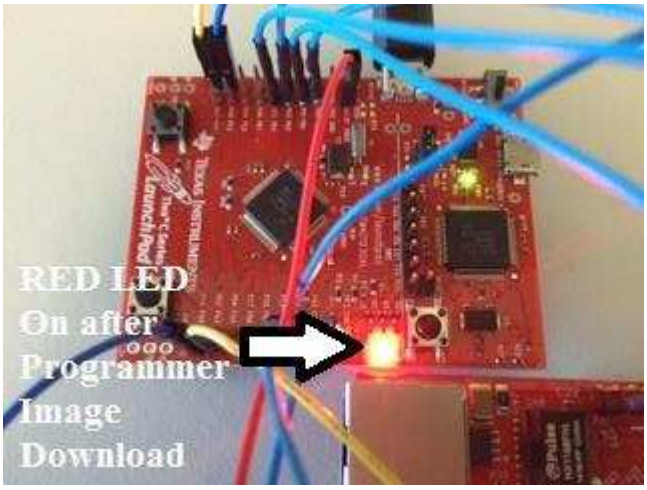

Figure 33. Programmer Board View after Download

6. Right click on the Device Manager entry for “TM4C Serial Bootloader”, then click on “Update Driver Software”. Navigate to the path where the installer has kept the drivers, as shown in Figure 34.

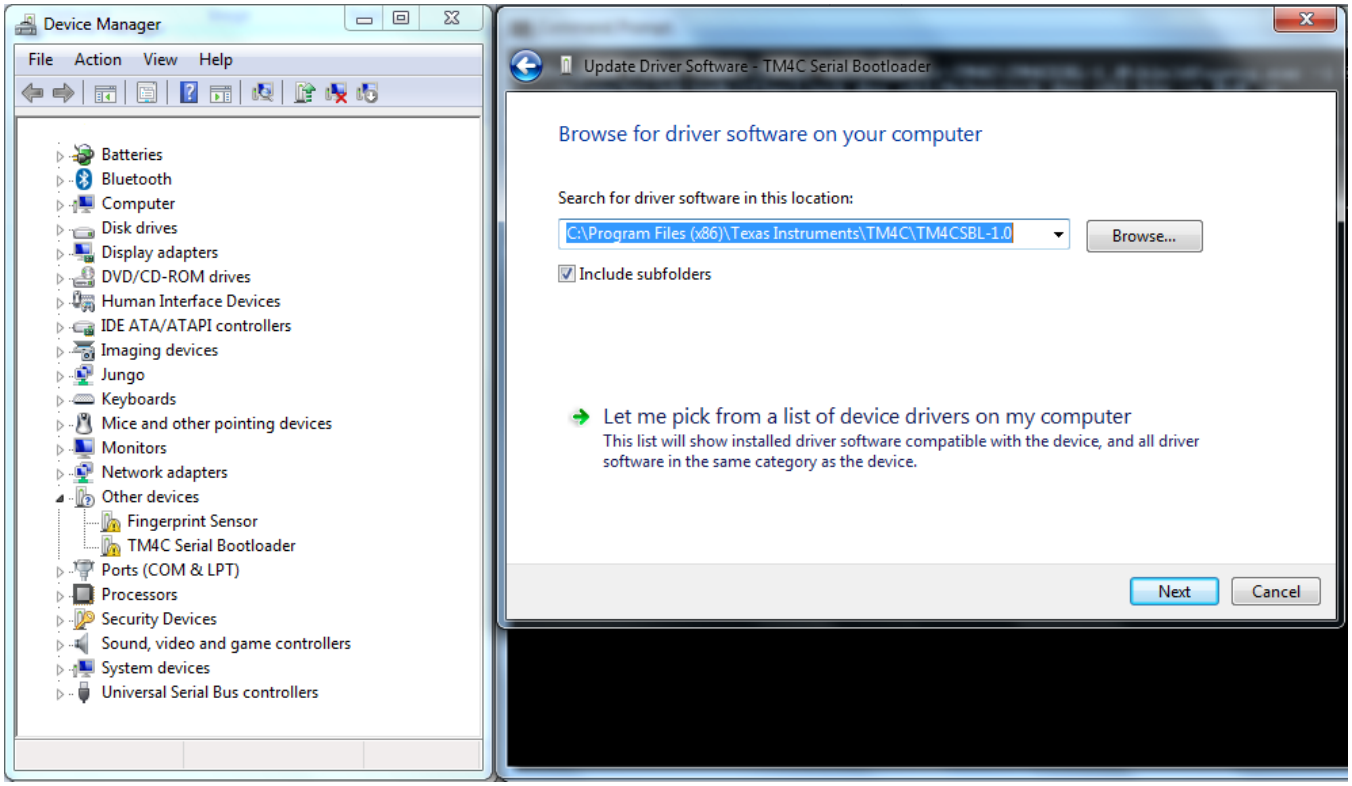

Figure 34. Updating the Device Drivers

7. You will get a prompt by the Windows OS because the drivers are not signed. Accept the warning message. Once the Windows OS updates the drivers, the device is re-enumerated as “TM4C Serial Bootloader” and the Green LED on the board is lit, which indicates that the device is now enumerated and connected to the PC Host, as shown in Figure 35.

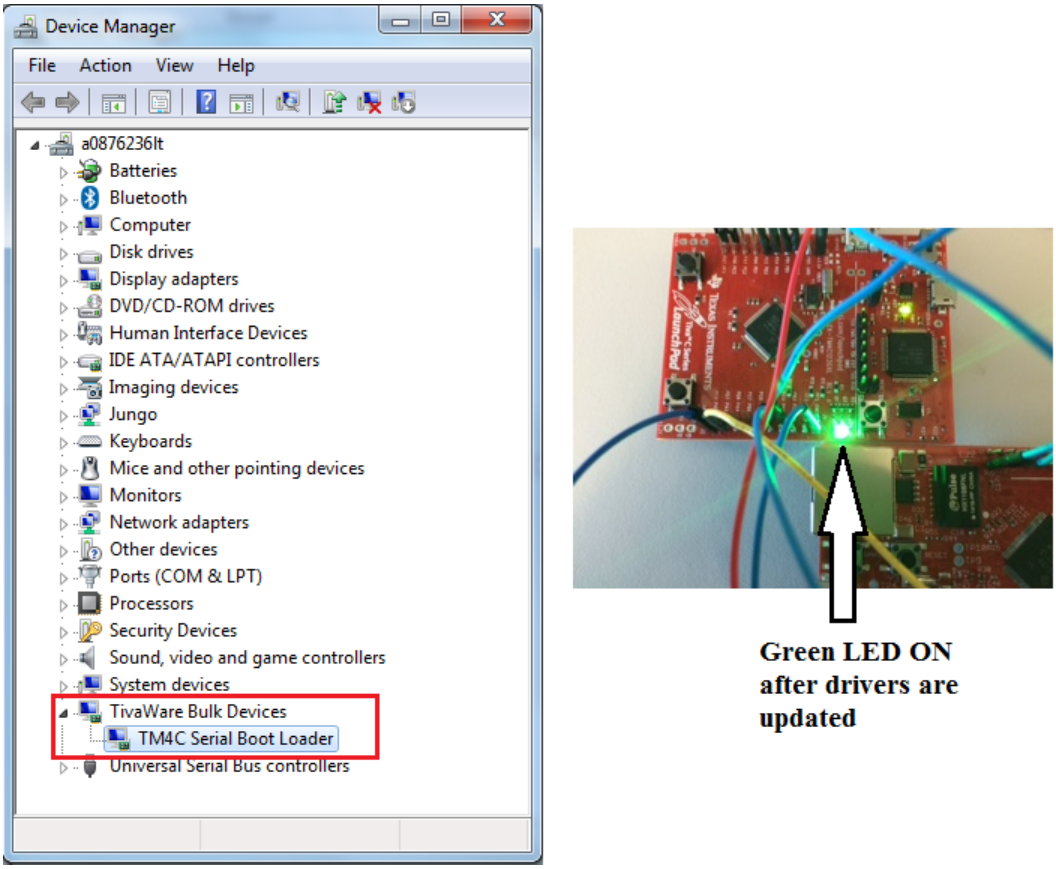

Figure 35. Device Manager and Board View After Driver Installation

5.2 Set up of SBL Graphical User Interface (GUI) Application

The user can now launch the “TM4C Serial Bootloader” application from the Windows “Start Menu”. On launching, the application the GUI would appear, as shown in Figure 36.

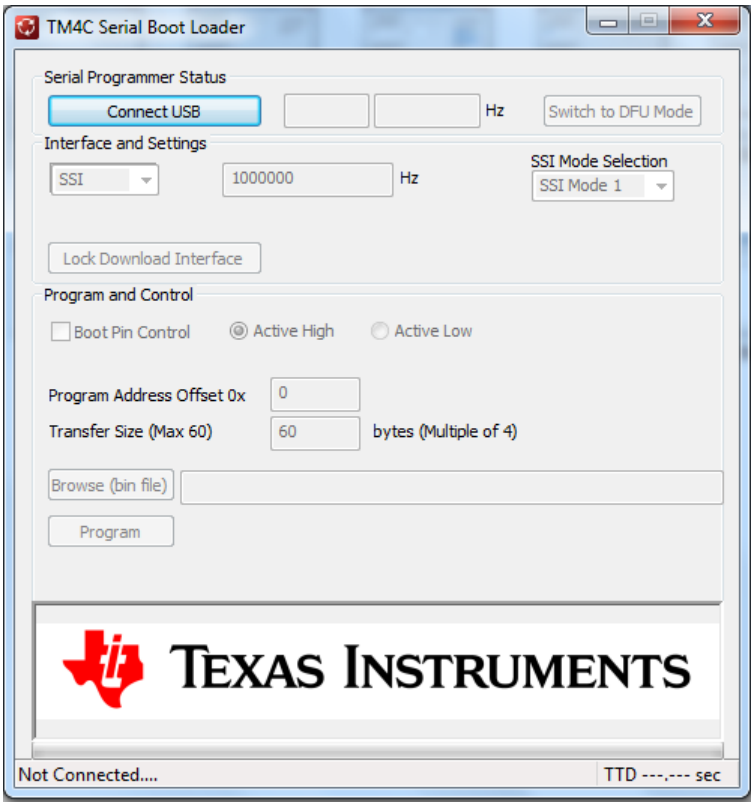

Figure 36. GUI Launch View

There are four distinct sections in the GUI application:

- Serial Programmer Status
  - “Connect USB” button is used to connect the GUI to the programmer board.
  - If the USB is connected, the programmer board sends information to the programmer device and system frequency that is populated in the two grayed edit boxes.
  - Switch to DFU Mode” button is used to switch the programmer board back to DFU mode in case a new programmer application binary has to be uploaded to the programmer.
- Interface and Settings
  - Use the drop down menu to select one of three interfaces: UART, I2C or SSI. The default is set to SSI.
  - The frequency of the interface or additional options for the interface is populated based on the interface selected from the drop down menu. The default values are the best fit when using the ROM Bootloader. The serial clock or baud rate can still be modified, but the additional options must not be changed when using ROM Bootloaders.
    - When SSI is selected, the options are the SCLK frequency and the Mode Selection.
    - When UART is selected, the options are the Baud Rate and the Auto Baud feature.
    - When I2C is selected, the options are SCL frequency and the Slave Address.
    - The “Lock Download Interface” button is used to lock the interface selected for download to prevent any unintentional change of download interface or its properties. When it is selected, the programmer board LED glows white.

- Program and Control
  - "Boot Pin Control" check box is to be used if the boot pin has been configured.
  - The polarity can be controlled by "Active High" or "Active Low" radio buttons, when the check box for "Boot Pin Control" is enabled. This ensures that when the target device is reset, the boot pin polarity is correctly set for the ROM Bootloader to be invoked.
  - The "Program Address Offset" edit box accepts hexadecimal address where the target device binary has to be downloaded.
  - The "Transfer Size" edit box is used to set the transfer size of the serial interface when the SEND_DATA command packet has to be sent. This decides the number of data bytes to be sent to the target device when the SEND_DATA command packet is sent.
  - The "Browse" button or the edit box is used to select the binary image that has to be downloaded to the target device.
  - The "Program" button begins the operation of reading the binary file and using the programmer board to download the image to the target device:
    - If the SSI interface is selected during program operation, the red LED will be lit.
    - If the I2C interface is selected during program operation, the green LED will be lit.
    - If the UART interface is selected during program operation, the blue LED will be lit.
- Status and Time To Download (TTD) bar
  - The Status bar on the bottom left of the GUI application shows the current state of the programmer and is used to indicate:
    - If the programmer is connected or not
    - If the programmer is Idle when the USB is connected
    - The number of bytes that were read from the file to be downloaded and sent to the target device
  - The TTD bar shows in the format of seconds and millisecond that the SBL took to download the image from the PC to the target device over the selected serial interface at the selected serial frequency. This is meant only for statistics.

5.3 Using the SBL GUI Application

After you are familiar with the layout of the application, begin using the application with the examples provided in the software package. Section 5.4 elaborates the error codes that the user may receive from the SBL GUI application and possible corrective actions.

1. Click on the “Connect USB” button. The GUI changes if the programmer board connects, as shown in Figure 37. The LED’s on the programmer board are not lit anymore.

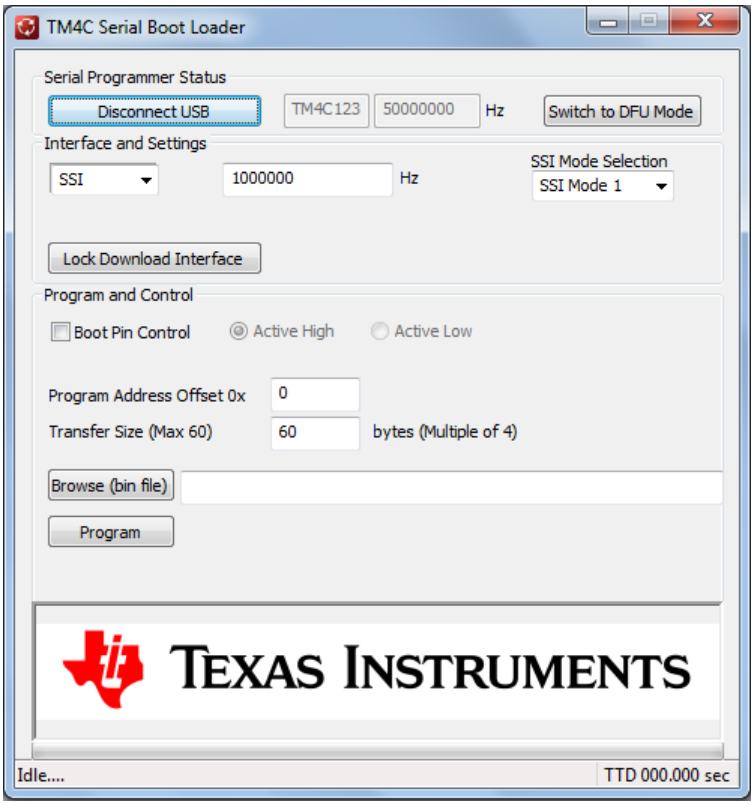

Figure 37. GUI View on USB Connect

2. If SSI interface is to be selected, set the frequency and SSI Mode of Operation. Note that SSI Mode 1 is the default mode and must be used when using the ROM-based bootloader.

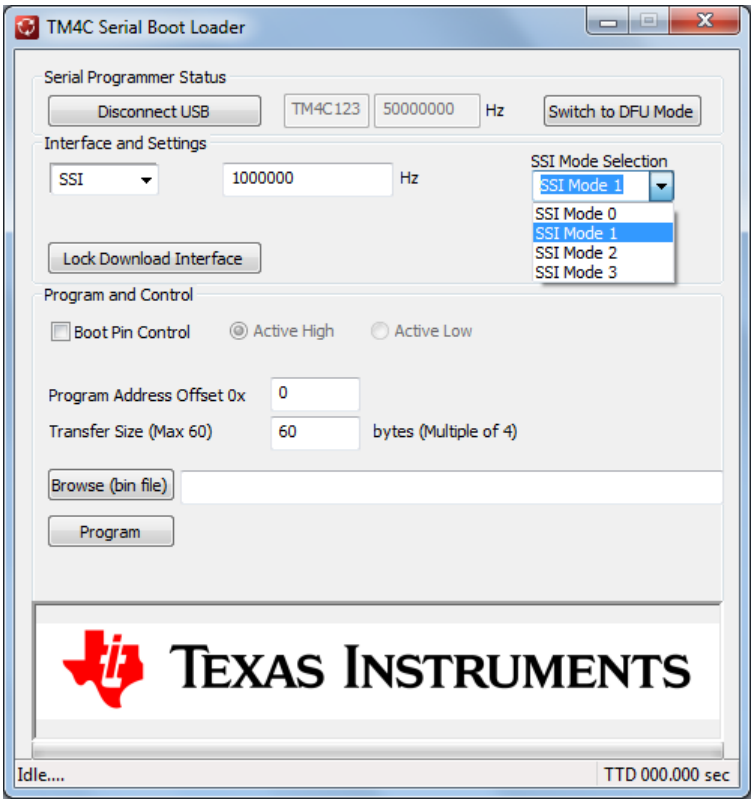

Figure 38. SSI Interface Selection

3. If I2C interface is to be selected, set the frequency and I2C Slave Address. Note that the I2C Slave address of 0x42 is the default address when using the ROM-based bootloader.

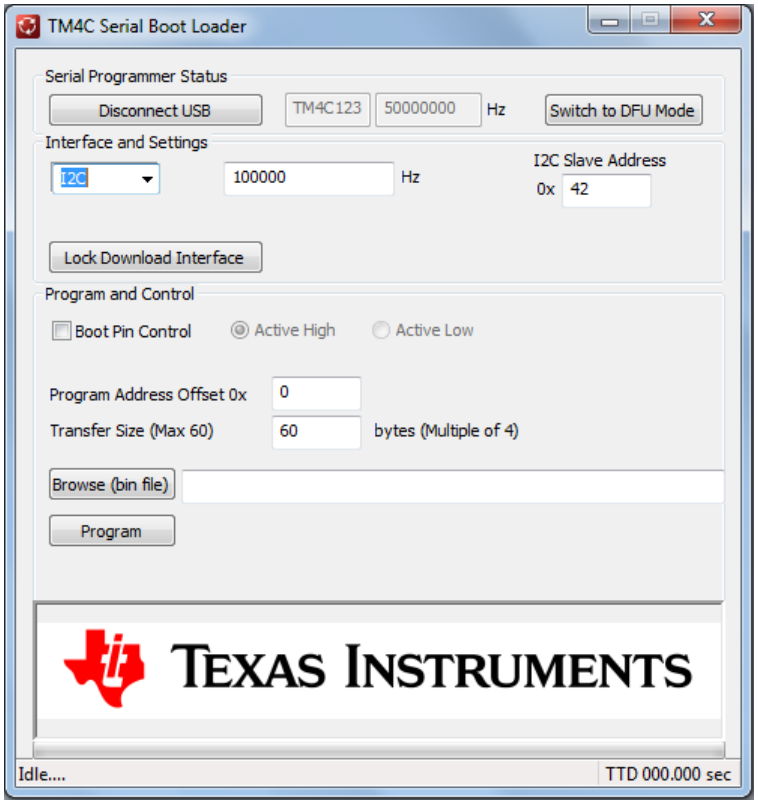

Figure 39. I2C Interface Selection

4. If UART interface is to be selected, set the baud rate and Auto Baud Enable. Note that the UART Auto Baud Enable must be set when using the ROM-based bootloader.

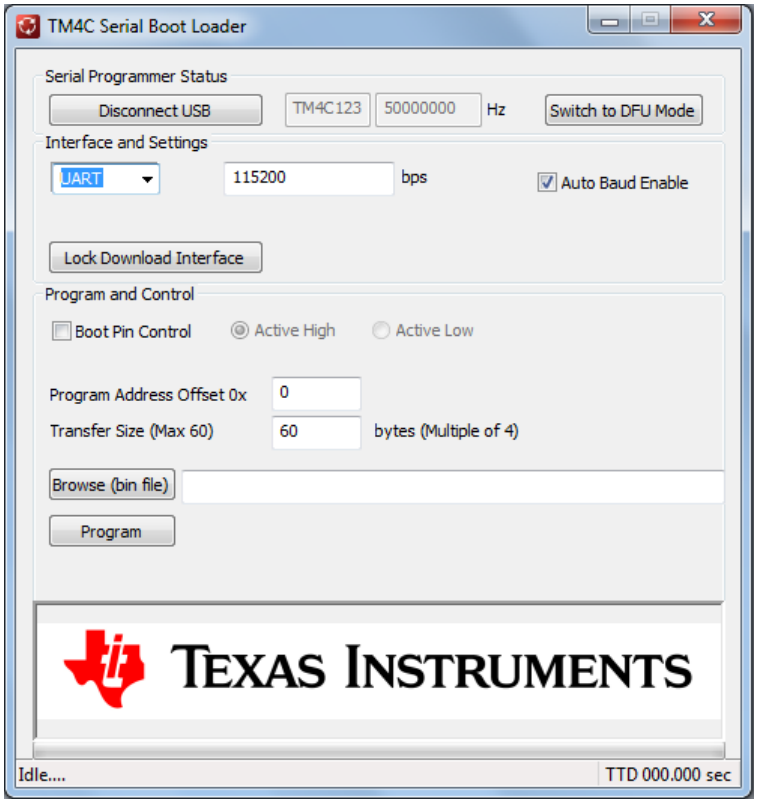

Figure 40. UART Interface Selection

5. If you want to prevent any inadvertent change to the selected SBL interface, then click on the “Lock Download Interface”. This deactivates the interface selection. Clicking on “Unlock Download Interface” reactivates the interface selection, as shown in Figure 41.

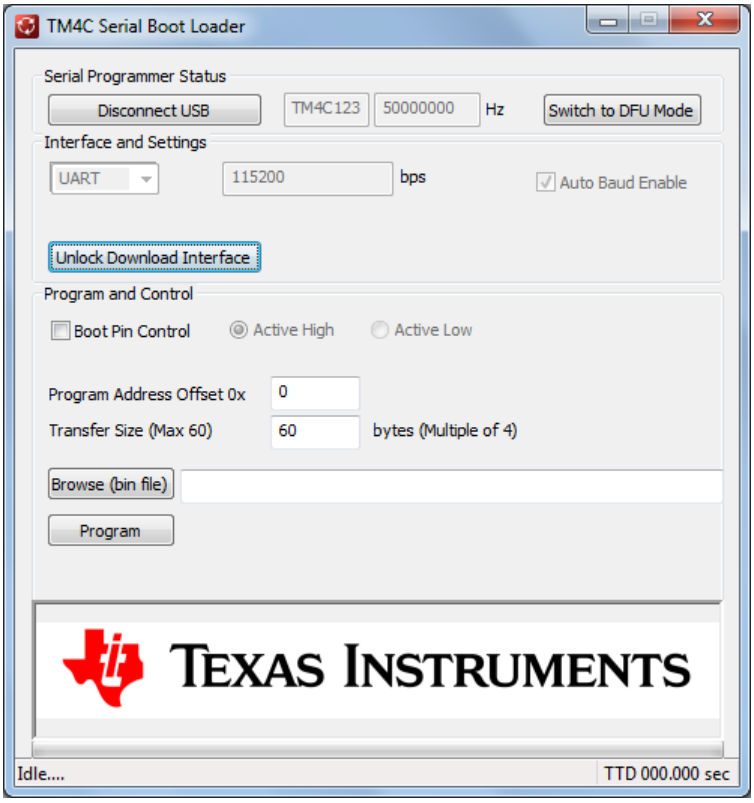

Figure 41. Locking the Interface Selection

6. Set the check box for “Boot Pin Control” and radio button for “Active Low”. The default examples use the settings as "Boot Pin Control" check box enabled and radio button set for "Active Low".

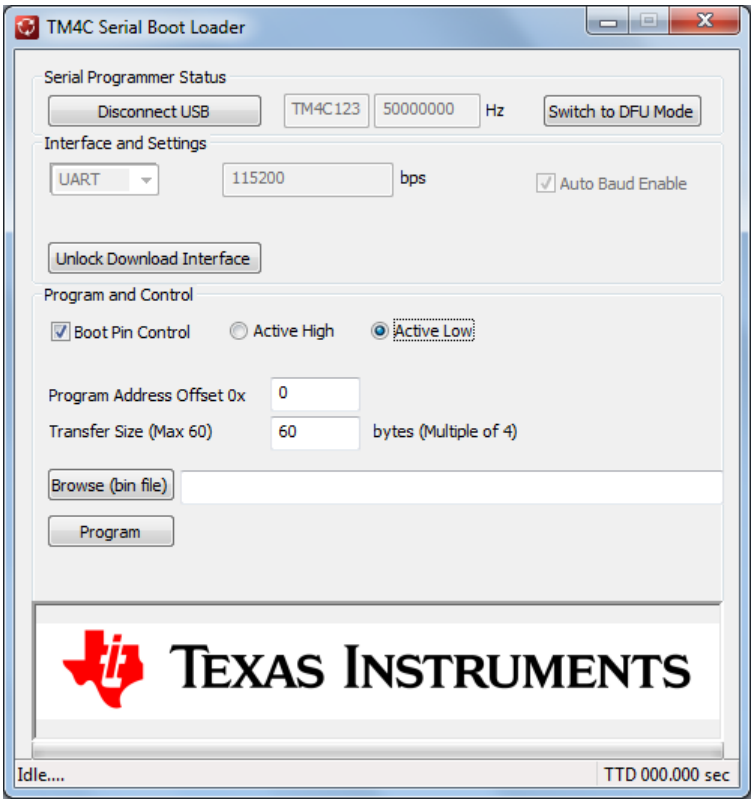

Figure 42. Boot Pin and Polarity Selection

7. In case of a wrong program being downloaded to the target device, there must be a recovery mechanism for the SBL. The example software “tm4c_device_recovery_code.bin” must first be programmed to the target device. Browse the PC for the “tm4c_device_recovery_code.bin” and click “Program” to download the image to the target device. You must get a dialog box indicating success; status bars will be updated for the number of bytes downloaded and time it takes to download as well.

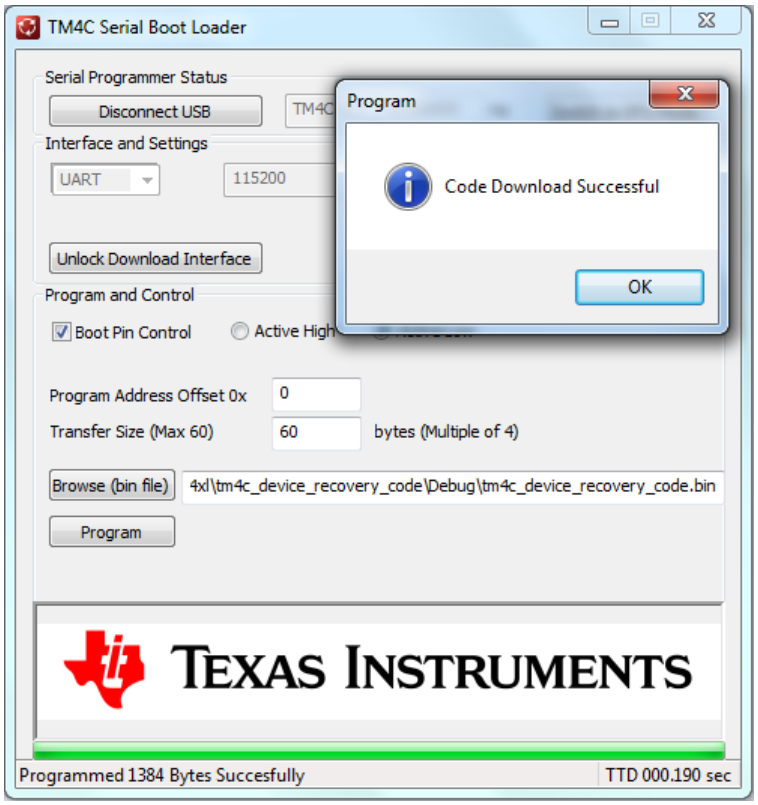

Figure 43. Downloading Recovery Code

8. Browse the PC for the “tm4c_all_led_blink.bin” and click on “Program” to download the file to the target device. The two LED’s on the target board will blink together and the status bar should be updated.

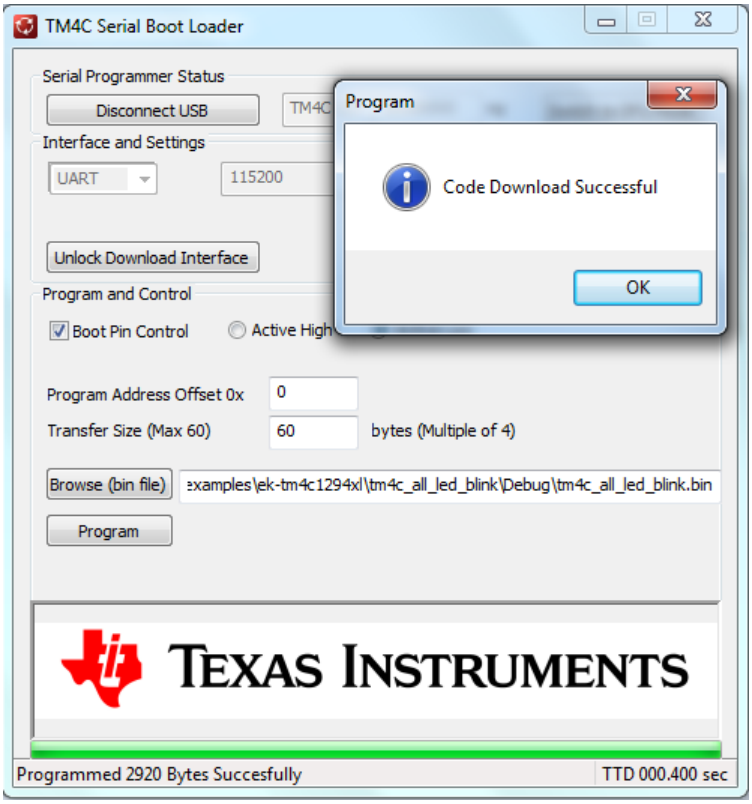

Figure 44. Downloading Blink All LED Code

9. Browse the PC for the “tm4c_faulty_code.bin” and click on “Program” to download the file to the target device. The status bar is updated showing programming completion. This code causes the device to lock out; any connection attempt via a debugger will not work.

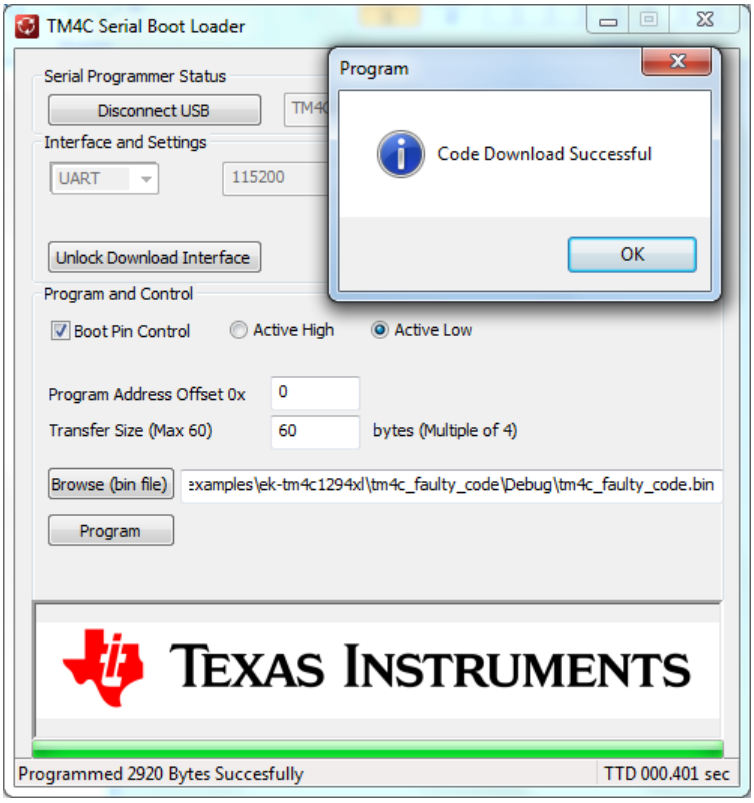

Figure 45. Downloading Faulty Code

10. Browse the PC for the “tm4c_alt_led_blink.bin” and click on “Program” to download the file to the target device. The two LED’s on the target board blink in an alternating manner and the status bar is updated. At this point, a connection attempt via a debugger should work again.

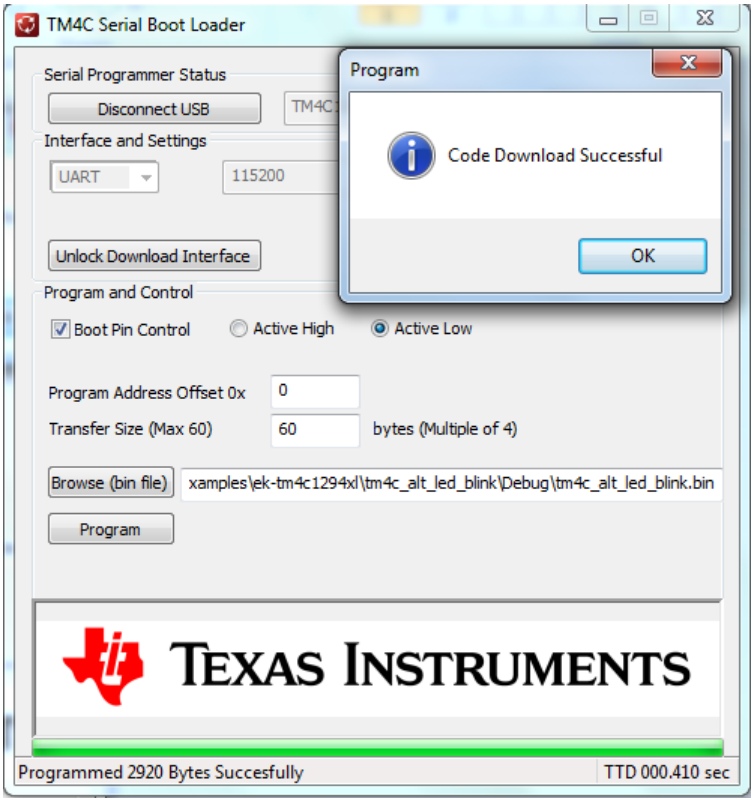

Figure 46. Downloading Blink Alternate LED Code

5.4 Error Code From SBL GUI

The SBL GUI application has error codes that are associated with different actions performed on the GUI interface or during communication of the PC host (with programmer device or during the communication of the programmer device with the target device). The messages sent to you on the GUI are self-explanatory. This section provides additional details for the user to be able to better diagnose the issue.

Table 4. Error Code From SBL GUI

Error Code	Error Description	Causes of Failure	Potential Corrective Action
ERR_PGM00	No Device Found	(a) USB cable between PC and Programmer is not connected (b) Programmer device does not have the usb_dev_sbl.bin and is in DFU Mode (c) Programmer device does not have the correct binary image to function.	(a) Check the cable connection or replace the cable (b) Update the programmer firmware using dfuprog.exe (c) Use the LMFlashProgrammer to erase EK-TM4C123GXL main MCU
ERR_PGM01	Error Locking/Unlocking Interface	Programmer device may be stuck because of another operation.	Reset the Programmer Device and Restart the SBL GUI
ERR_PGM02	Error Selecting SSI	Programmer device may be stuck because of another operation.	Reset the Programmer Device and Restart the SBL GUI
ERR_PGM03	Error Selecting I2C	Programmer device may be stuck because of another operation.	Reset the Programmer Device and Restart the SBL GUI
ERR_PGM04	Error Selecting UART	Programmer device may be stuck because of another operation.	Reset the Programmer Device and Restart the SBL GUI
ERR_PGM05	Max Transfer Size Exceeded	The GUI Transfer Size has been set to more than 60 bytes	Set the Transfer Size greater or equal to 4, less than or equal to 60 and integral multiple of 4
ERR_PGM06	Zero Size Transfer Not Supported	The GUI Transfer Size has been set to 0 byte	Set the Transfer Size greater or equal to 4, less than or equal to 60 and integral multiple of 4
ERR_PGM07	Number of Bytes must be a multiple of 4	The GUI Transfer Size has been set to a byte size which is not a multiple of 4	Set the Transfer Size greater or equal to 4, less than or equal to 60 and integral multiple of 4
ERR_PGM08	Error in Interface Selection	The GUI application Combo Box for interface selection has parsed the wrong value	Reset the Programmer Device and Restart the SBL GUI
ERR_PCH00	File Not Found	The Target Binary is not valid when “Browse” button was used to select a binary image	Check the file path for the target application image
ERR_PCH01	File Not Found	The Target Binary is not valid when the download of the binary image is to be performed	Check the file path for the target application image, Reset the Programmer Device and Restart the SBL GUI
ERR_PCH02	File Not Found	The Target Binary is not valid when the edit box is used to select a new binary	Check the file path for the target application image
ERR_FREQ00	SSI Clock Frequency Error	The SSI clock frequency selected is 0 Hz	Set the SSI Clock Frequency to 1 MHz
ERR_FREQ01	SSI Clock Frequency Error	The SSI clock frequency selected on the programmer is more than half of the system clock frequency of the programmer.	Set the SSI Clock Frequency to 1 MHz
ERR_FREQ02	I2C Clock Frequency Error	The I2C clock frequency selected is 0 Hz	Set the I2C Clock Frequency to 100 KHz or 400 KHz
ERR_FREQ03	I2C Clock Frequency Error	The I2C clock frequency selected on the programmer is more than 400Khz.	Set the I2C Clock Frequency to 100 KHz or 400 KHz
ERR_FREQ04	UART Baud Rate Error	The UART baud rate selected is 0 bps	Set the UART Baud Rate to 115200 bps
ERR_FREQ05	UART Baud Rate Error	The UART baud rate selected is less than 921600 bps but more than 1/16 of the system clock frequency of the programmer	Set the UART Baud Rate to 115200 bps
ERR_FREQ06	UART Baud Rate Error	The UART baud rate selected is more than 921600 bps	Set the UART Baud Rate to 115200 bps

Table 4. Error Code From SBL GUI (continued)

Error Code	Error Description	Causes of Failure	Potential Corrective Action
ERR_DNLD011	NAK from Target	NAK received from Target device during PING command packet	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD012	TIMEOUT from Target	TIMEOUT on the selected interface when waiting for a response for PING command packet	Check if the Interface is correctly connected between the programmer and target board, or replace the wires, or erase the Target device using LMFlashProgrammer using the Unlock Sequence
ERR_DNLD021	NAK from Target	NAK received from Target device during DOWNLOAD command packet	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD022	TIMEOUT from Target	TIMEOUT on the selected interface when waiting for a response for DOWNLOAD or GET_STATUS command packet	Check if the Interface is correctly connected between the programmer and target board, or replace the wires
ERR_DNLD023	CHECKSUM Error from Target	CHECKSUM error on the selected interface when GET_STATUS command is issued.	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD024	Data Size Error from Target	Extra Data bytes received during DOWNLOAD command on the selected interface when GET_STATUS command is issued.	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD0xx	Unknown Error from Target	Unknown error code from target device over serial interface or USB packet error from programmer when waiting for a response for PING, DOWNLOAD or GET_STATUS command.	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD101	NAK from Target	NAK received from Target device during SEND_DATA command packet except for last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD102	TIMEOUT from Target	TIMEOUT on selected interface when waiting for SEND_DATA or GET_STATUS command packet except for last data packet when the size of the target application binary is not a multiple of transfer size	Check if the Interface is correctly connected between the programmer and target board, or replace the wires
ERR_DNLD103	CHECKSUM Error from Target	CHECKSUM error on the selected interface when GET_STATUS command is issued except for last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD104	Data Size Error from Target	Extra Data bytes received during SEND_DATA command on the selected interface when GET_STATUS command is issued except for last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD1xx	Unknown Error from Target	Unknown error code from target device over serial interface or USB packet error from programmer when waiting for a response for SEND_DATA or GET_STATUS command during target application binary download phase. This is valid except for last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer

Table 4. Error Code From SBL GUI (continued)

Error Code	Error Description	Causes of Failure	Potential Corrective Action
ERR_DNLD201	NAK from Target	NAK received from Target device during SEND_DATA command packet for the last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD202	TIMEOUT from Target	TIMEOUT on selected interface when waiting for SEND_DATA or GET_STATUS command packet for last data packet when the size of the target application binary is not a multiple of transfer size	Check if the Interface is correctly connected between the programmer and target board, or replace the wires
ERR_DNLD203	CHECKSUM Error from Target	CHECKSUM error on the selected interface when GET_STATUS command is issued for the last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD204	Data Size Error from Target	Extra Data bytes received during SEND_DATA command on the selected interface when GET_STATUS command is issued for the last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_DNLD2xx	Unknown Error from Target	Unknown error code from target device over serial interface or USB packet error from programmer when waiting for a response for SEND_DATA or GET_STATUS command during target application binary download phase. This is valid for the last data packet when the size of the target application binary is not a multiple of transfer size	Check the Interface transaction of a scope or logic analyzer
ERR_SRST01	NAK from Target	NAK received from Target device during RESET command packet	Check the Interface transaction of a scope or logic analyzer
ERR_SRST02	TIMEOUT from Target	TIMEOUT on the selected interface when waiting for a response for RESET command packet	Check if the Interface is correctly connected between the programmer and target board, or replace the wires
ERR_SRSTxx	Unknown Error from Target	Unknown error code from target device over serial interface or USB packet error from programmer when waiting for a response for RESET command phase	Check the Interface transaction of a scope or logic analyzer

6 Performance Data

Since the serial bootloaders do not use the Debug Architecture or JTAG port, the high-level command packet structure can be used to download large application binaries in comparatively shorter time. Table 5 shows the performance of this programmer for the ROM bootloader. This can be improved further by using a Flash-based bootloader that may be configured to run the system clock and peripheral at higher clock frequencies on the target device. Also the programmer can utilize a scheduler using the uDMA channel for copying the data from the USB to the SBL interface, allowing the CPU to get the next data frame ahead of the current data frame completion. For evaluation purposes, the performance was measured with a 1MB binary image downloaded to a TM4C1294NCPDT on the EK-TM4C1294XL.

Table 5. Performance Data

SBL Interface	Serial Clock Frequency or Baud (Hz/bps)	Transfer Size (bytes)	Time To Download (sec)
SSI	5,000,000	60	36.707
UART	921,600	60	49.171
I2C	400,000	60	72.384

7 Summary

The Serial Bootloader feature and the application provide an easy-to-use tool and lightweight interface to factory program the TM4C12x class of devices using the ROM Bootloader. The availability of source code for the programmer application can be used by Texas Instruments customers to design programmers that are more specific to their needs. With a little effort, it would also be possible to use this approach for an over-the-air update application using Texas Instruments wireless connectivity solutions for the TM4C12x-based products.

8 References

- *Tiva™ TM4C123GH6PM Microcontroller Data Sheet* (SPMS376)
- *Tiva™ TM4C1294NCPDT Microcontroller Data Sheet* (SPMS433)
- *Tiva™ C Series TM4C123G LaunchPad Evaluation Board User's Guide* (SPMU296)
- *Tiva™ C Series TM4C1294 Connected LaunchPad Evaluation Kit User's Guide* (SPMU365)
- *TivaWare™ Boot Loader User's Guide* (SPMU301)
- TivaWare™ for C Series
- Flash Programmer, GUI and Command Line

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from Original (November 2015) to A Revision	Page
• Added new Section 2.2.	3
• Updated information in Section 2.3.4.	9
• Added new Section 3.3.	17

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve 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 (also referred to herein as “components”) 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 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.

TI does not warrant or represent that any license, either express or implied, is granted under 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.

Reproduction of significant portions of TI information in TI data books or data sheets 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.

Resale of TI components or services with statements different from or beyond the parameters stated by TI for that component or service voids all express and any implied warranties for the associated TI component or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

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 which anticipate dangerous consequences of failures, monitor failures and their consequences, lessen the likelihood of failures that might cause harm 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 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 a special agreement specifically governing such use.

Only those TI components which 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 which have ***not*** been so designated is solely at the Buyer's risk, and that 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.

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		

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