Renesas Synergy™ Software Package (SSP) v1.1.0

R11AN0024EU0140
Rev.1.40
Sep 1, 2016

Developer Examples

Introduction

The purpose of this Application Note is to get you started using the Developer Example applications included with SSP. A Developer Example is a simple application demonstrating the functionality of each SSP Framework or HAL module. You can call each application via a command line interface on a serial terminal.

Target Device and Software Requirements

- DK-S7G2
- Renesas Synergy e^2 studio v5.0.0.043

NOTE: This release was tested with Renesas Synergy Software Package v1.1.0. The Developer Examples and the associated project template is automatically installed with any SSP v1.1 installer.

Contents

1. Prerequisites

This guide assumes that you have installed the Synergy Software Package (SSP) and e^2 studio ISDE on your computer and have the DK-S7G2 board successfully configured and set up with the J-Link debugger. You can verify that board and e^2 studio ISDE are working correctly by running the 'Blinky' demonstration example available for all Renesas Synergy boards. In addition, you need a common PC hosted editor.

It is also helpful if you have some familiarity with the overall layout of the e^2 studio ISDE windows and with generating Synergy projects in e^2 studio, since the steps below are less 'guided' than the steps in the 'Blinky' project and they don't illustrate each window or command location used in each step.

All examples use a terminal emulator program such as Tera Term.

2. Overview

This document is intended for developers who use the Synergy DK-S7G2 Development Kit and want to get a quick start on how to use a module and its interface.

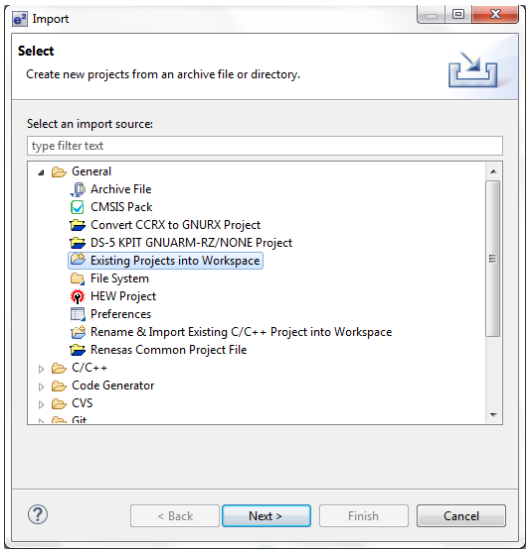
This document provides detailed information about how to exercise the module's APIs from the command line using a terminal emulator like Tera Term.

This document also explains about how to select the Root menu in command line interface and how to select the specific modules menu from Root menu. Screen shots of the terminal window show the supported commands and how to use the commands.

3. Build and Run a Developer Example application

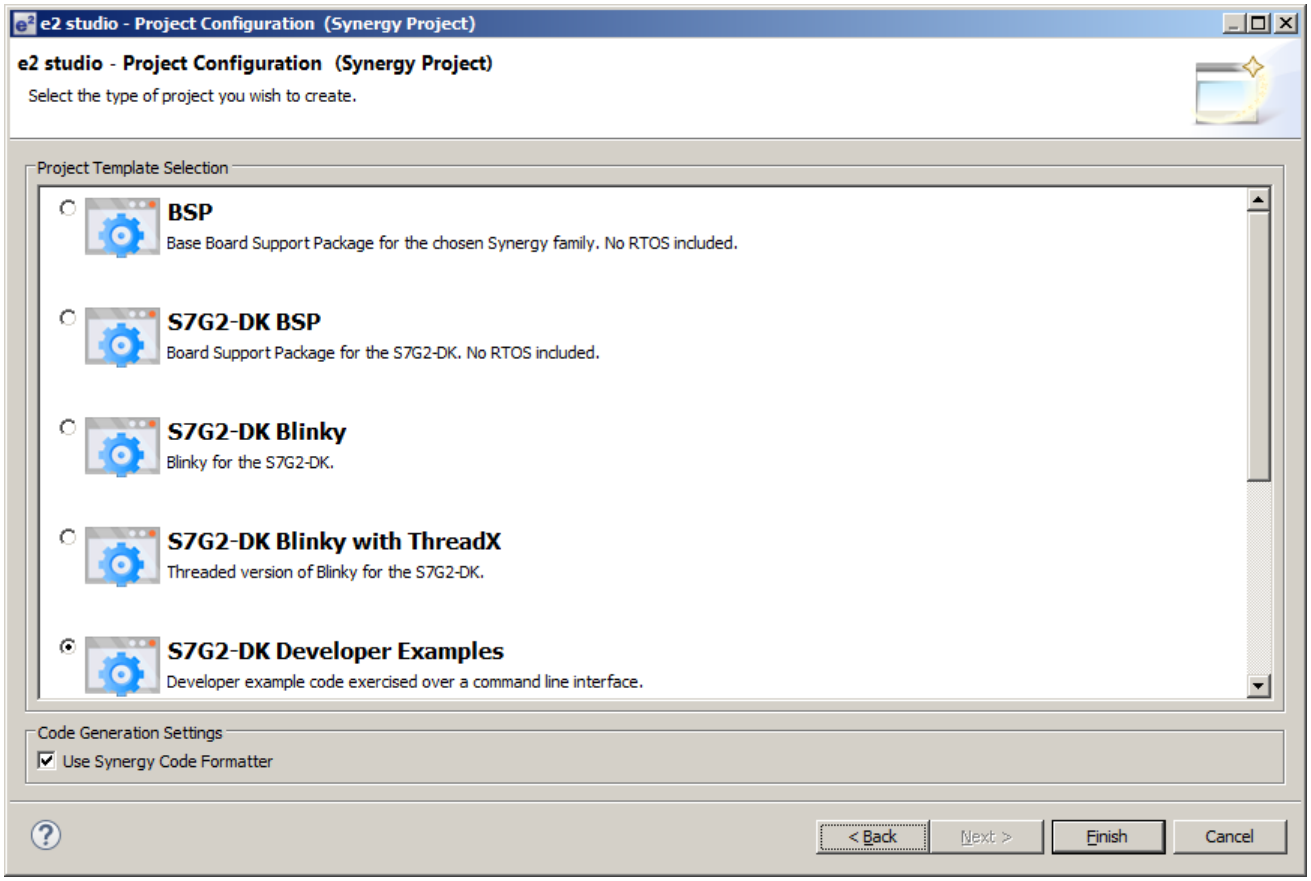
You can build and run all Developer Example applications described in this document by following these common steps:

STEP 1: Launch e^2 studio. Navigate to **File**>**New > Synergy Project**. The Import dialog box opens.



STEP 2: Enter a name for your project, select a license file, and click **Next**. Select version 1.1.0 of the SSP and the **S7G2 DK** board. Click **Next**.

STEP 3: Select the project **S7G2-DK Developer Examples**. Click **Finish**.

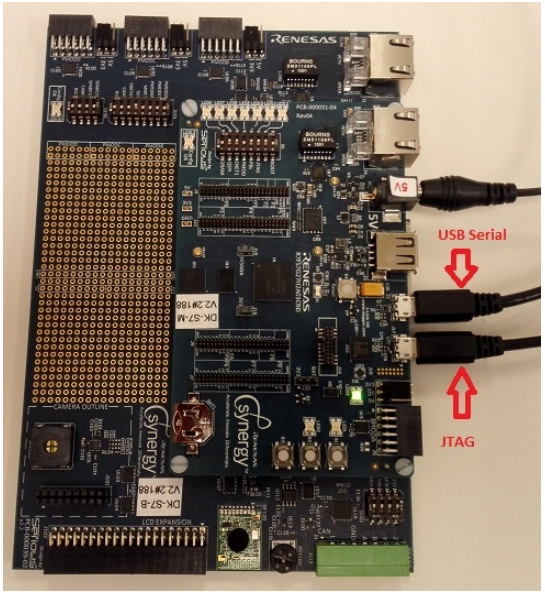


STEP 4: Build the project.

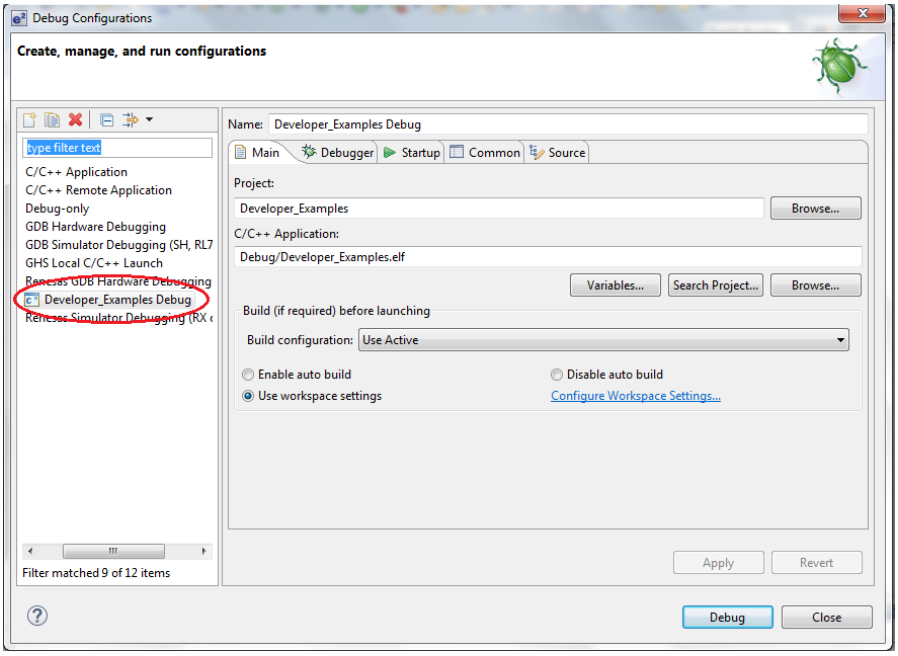
STEP 5: Connect the DK-S7G2 board to the host PC. Two connections are needed:

a) The JTAG debug connection to program and debug the board

b) The USB-CDC connection for console access



STEP 6: Power on the board. In ISDE, click **Run**>**Debug configurations**. A new debug configuration with the project name will be created. Click Debug.

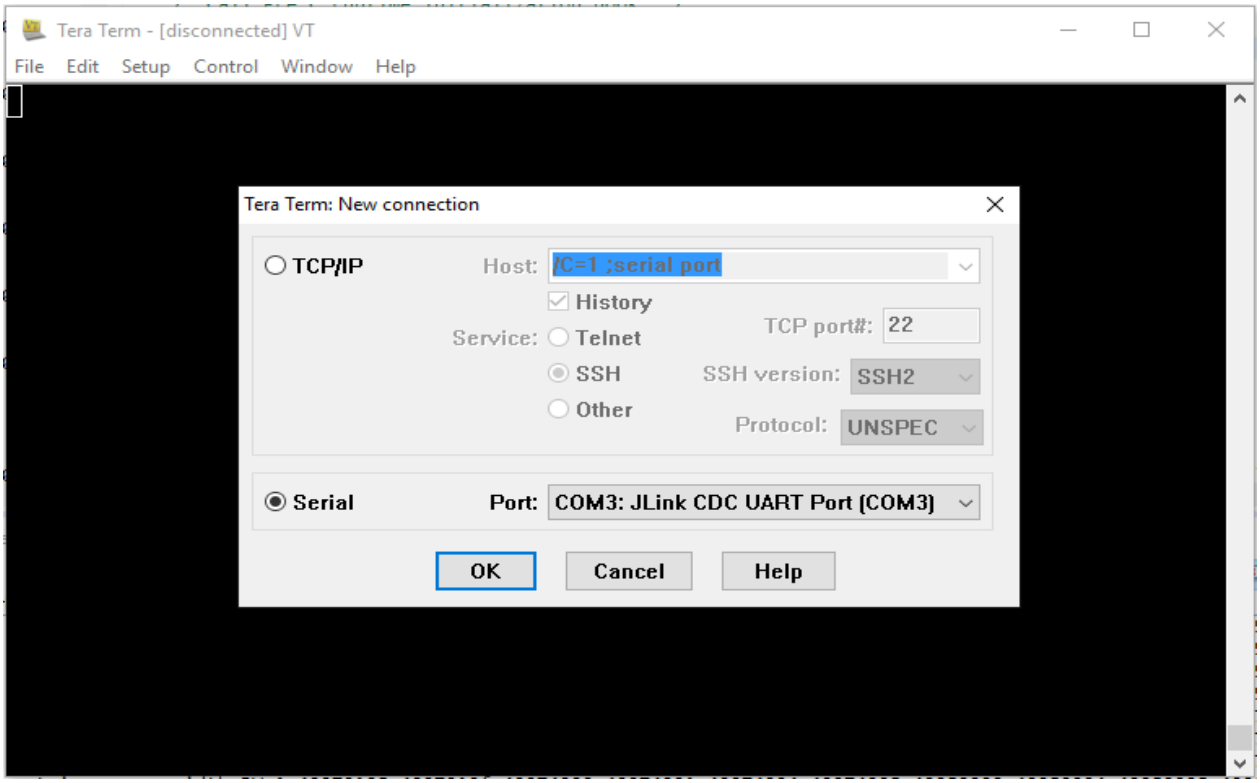


STEP 7: Click **Yes** to switch to the Debug Perspective if you are asked.

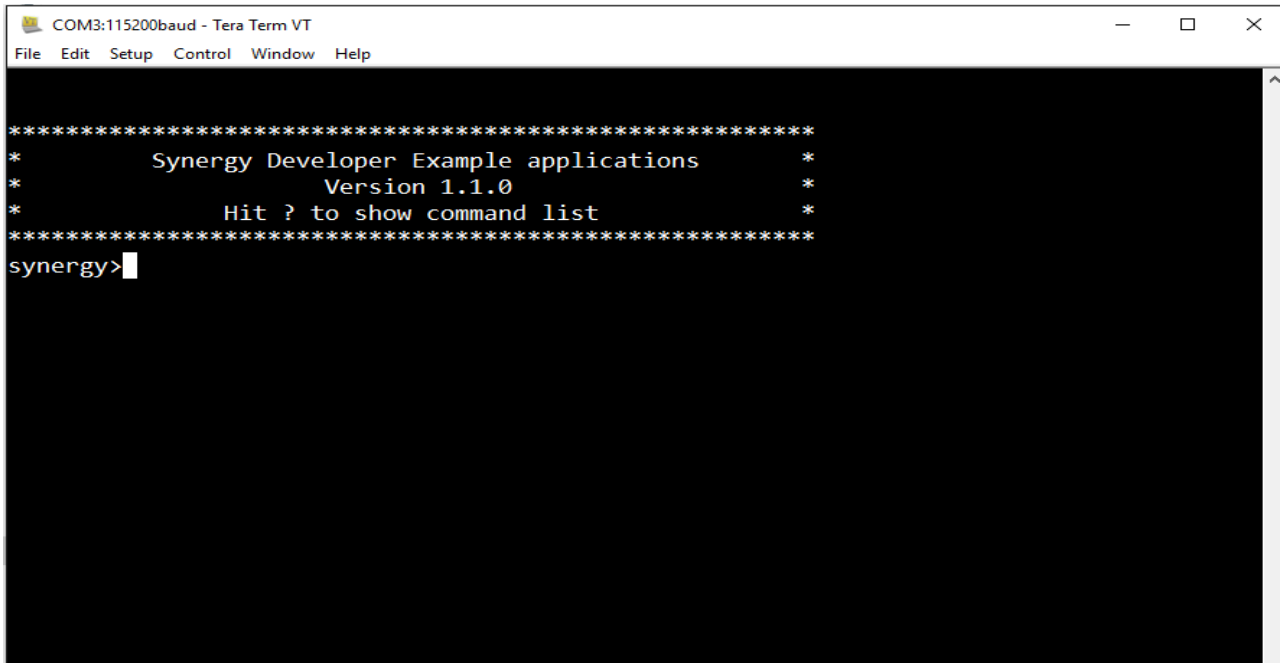
STEP 8: Click the Resume button  twice so that the application starts its scheduler.

STEP 9: If this is the first time launch, wait for the host PC to recognize the USB device as composite device and install the required driver. Once the driver is installed, launch Tera Term.

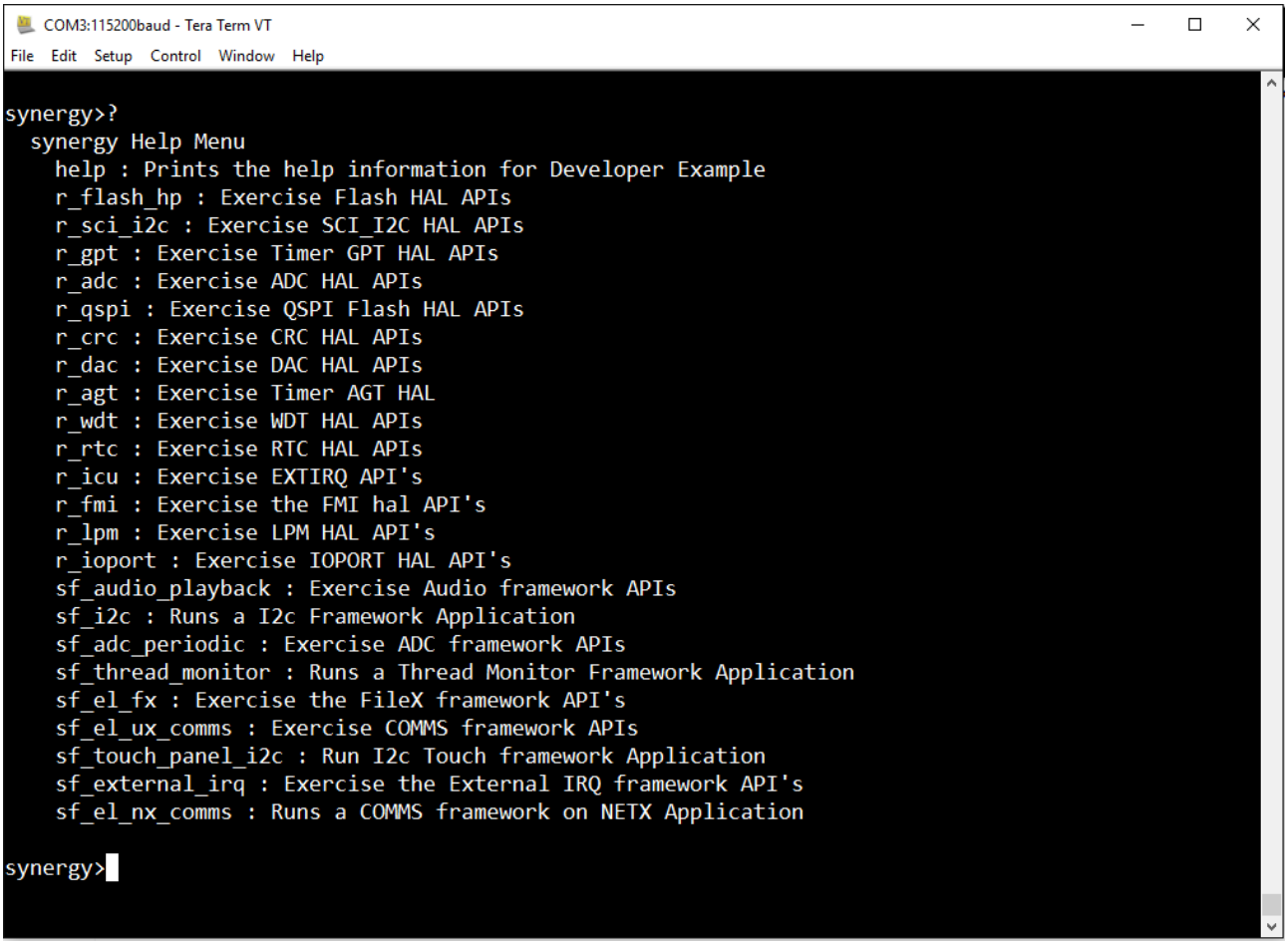
STEP 10: Choose the serial connection and choose the corresponding serial port (COM3: JLink CDC UART Port [COM3]).



STEP 11: Press the Enter key to get the console prompt.



STEP 12: Type ? and press Enter to get the Help menu showing a list of supported Developer Example applications.



To enter the submenu for any of the Developer Examples, type the name of the application and press Enter. For example, to use the ADC Framework application, type `sf_adc_periodic`. To see a list of the supported APIs, press `?` and Enter.

NOTE: Commands typed in the Tera Term window are not case sensitive

4. Developer Example: ADC Periodic Framework

4.1 Introduction

The ADC Periodic Framework operates as follows:

The GPT timer is configured to trigger an ADC group scan at periodic intervals. When the scan is complete, a DTC operation is triggered which copies the scan result to a user buffer. When completing such iterations, you are notified about the data transfer.

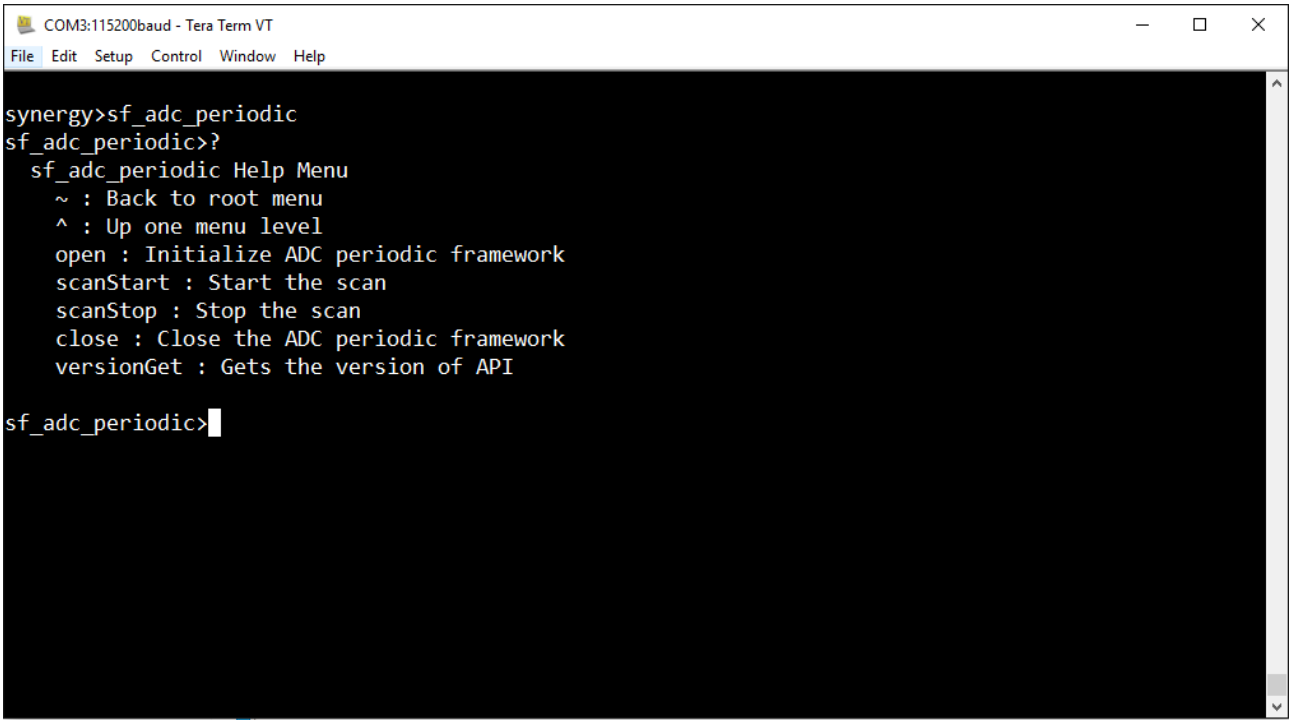
In this Developer Example, the ADC is configured to scan channel (AN000), which is connected to a potentiometer on the DK-S7G2. When the scan is complete, the DTC triggers a data transfer to the user buffer and the listening thread is notified. When receiving the notification, the thread uses I2C HAL drivers to interface to an on-board I/O expander which toggles the LEDs. This operation is repeated with the period configured for the GPT timer.

4.2 Run the ADC Periodic Framework application

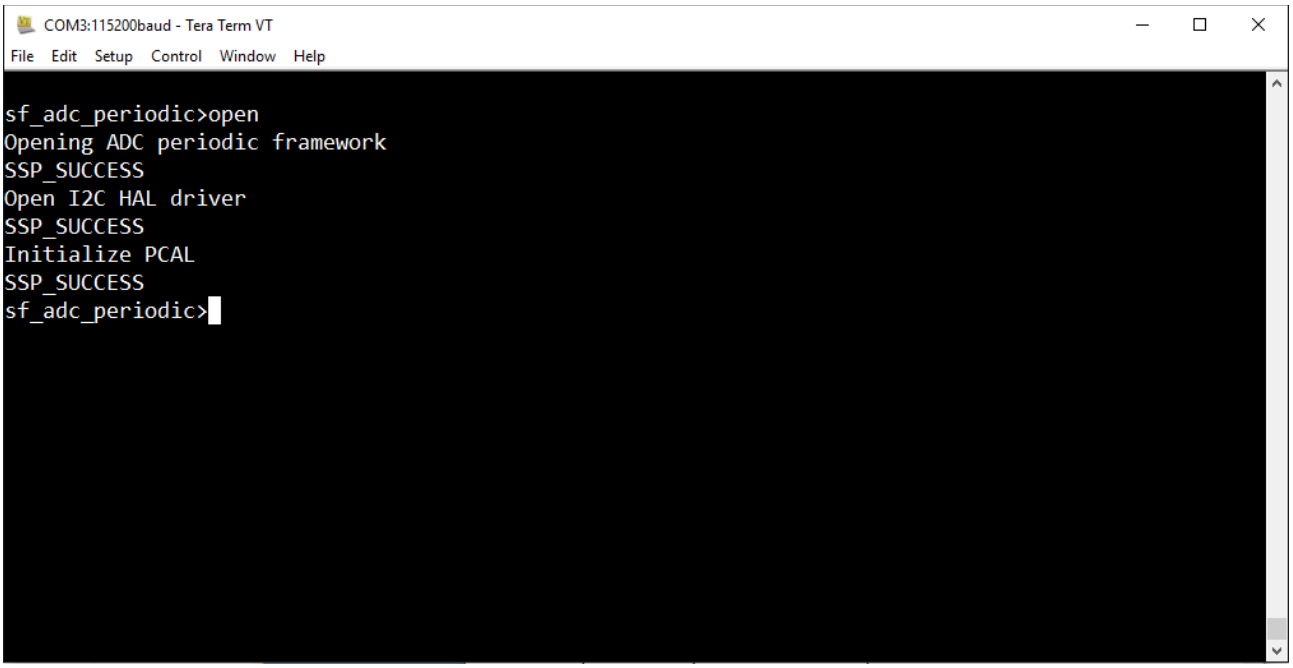
Follow the steps described in Build and Run a Developer Example application to obtain the help menu with the list of applications in the terminal window.

To run the ADC Periodic Framework application, follow these steps:

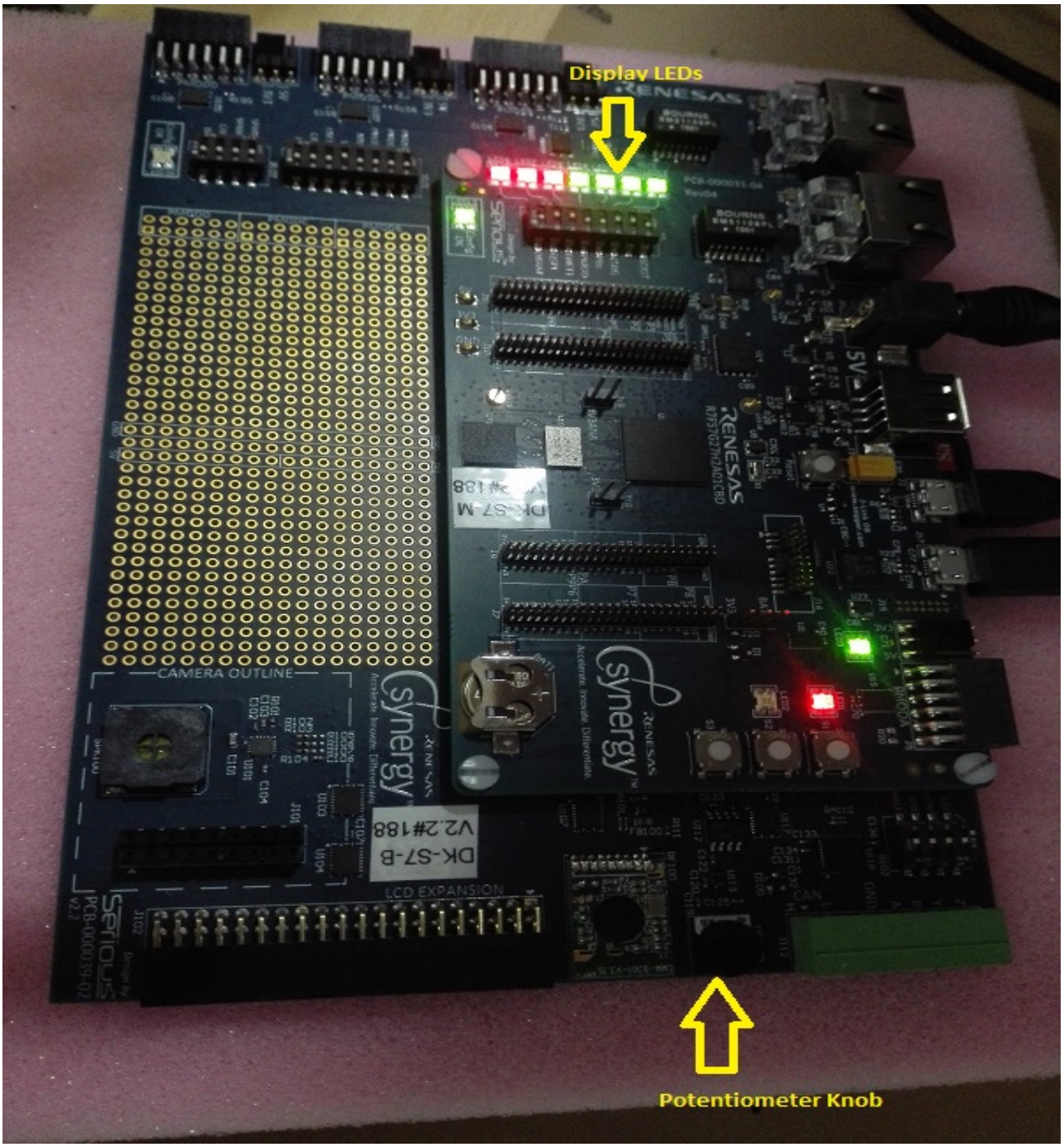
STEP 1: Type `sf_adc_periodic` in the terminal and press Enter to access the ADC Periodic Framework submenu. For help, type ? and press Enter.



STEP 2: Execute the `open` command. It opens the SCI I2C HAL driver and configures the I/O expander. It also configures the ADC Periodic Framework, but as part of the thread entry function, the ADC framework is already opened. Thus it might result in an SSP_ERROR_IN_USE. You can safely ignore that error.



STEP 3: Start the scan using the command `scanStart`. Turn the knob of the potentiometer and observe that the LEDs toggle based on the direction of the motion of the knob.



STEP 4: The `scanStop` command stops the periodic ADC scan. Once the `scanStop` command is executed, the state of the LEDs does not change with the motion of the potentiometer knob.

STEP 5: Close the ADC framework instance by typing the `close` command. Closing the ADC Framework instance also closes the I2C HAL drivers that were opened as part of the `open` command.

5. Developer Example: ADC HAL driver

5.1 Introduction

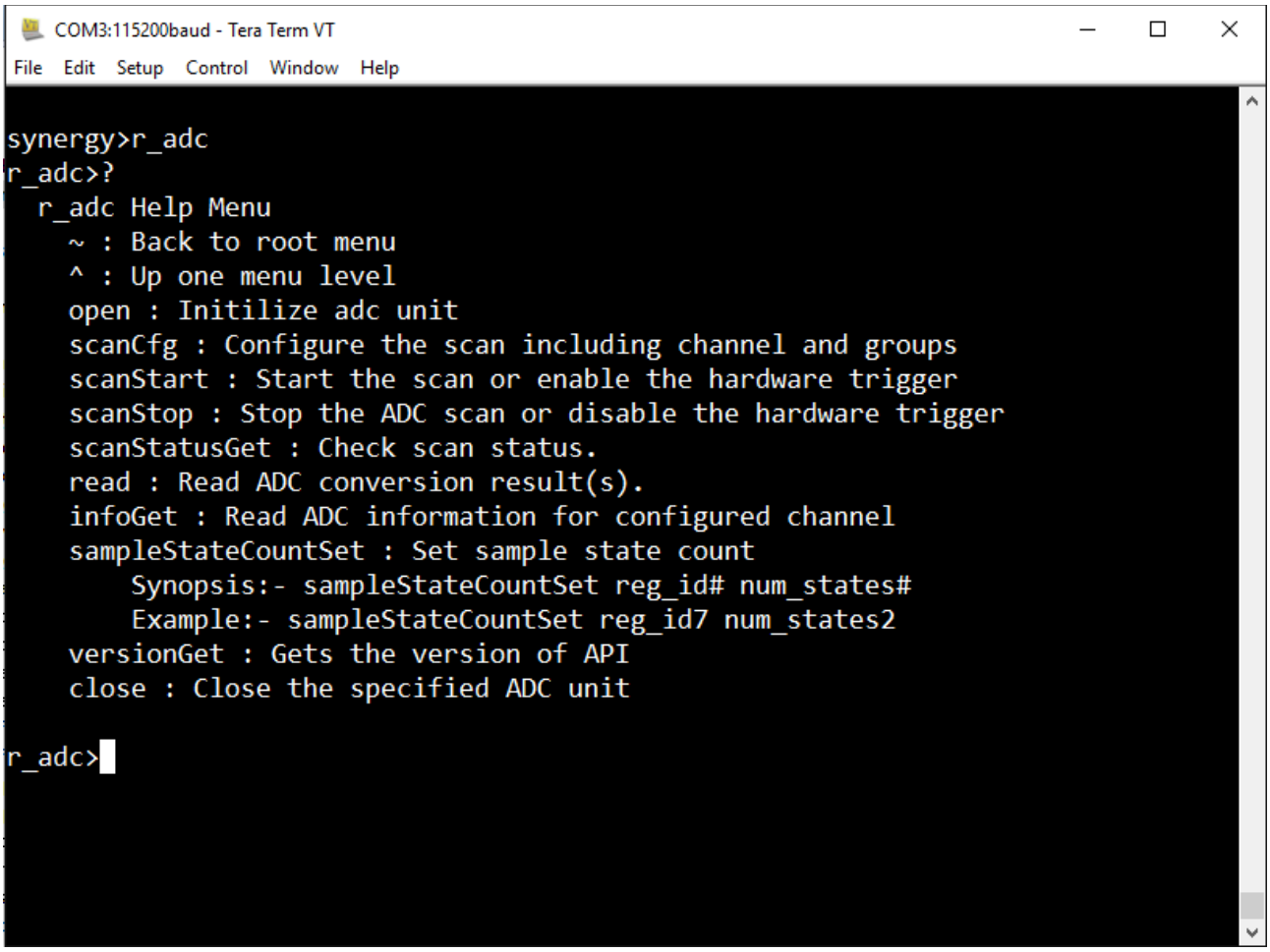
This Developer Example uses the ADC HAL APIs in single scan mode. On the DK-S7G2 board, channel 0 of the ADC is connected to a potentiometer. You can observe the changes in ADC value when the potentiometer knob is varied and a scan is performed.

5.2 Run the ADC HAL Driver application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To perform a scan and read the ADC value, follow these steps:

STEP 1: Type `r_adc` in the terminal and press enter to access the ADC HAL submenu. For help, type `?` and press Enter.

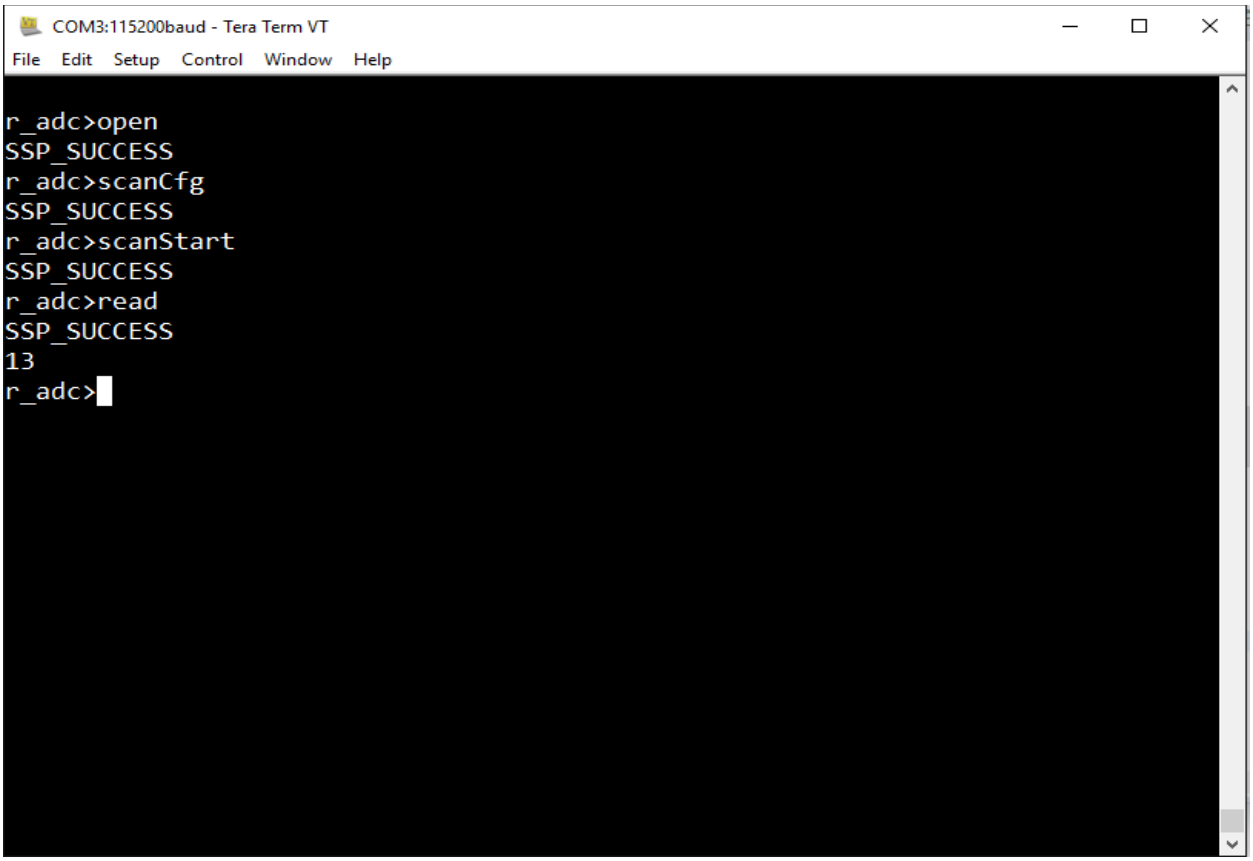


STEP 2: Invoke each menu item with the corresponding arguments to use the APIs. First configure the scan by executing the following commands in sequence:

1. `open`
2. `scanCfg`

To read the ADC value, first perform a scan using the `scanStart` command and then read the value using the `read` command:

1. `scanStart`
2. `read`



6. Developer Example: Audio Playback Framework

6.1 Introduction

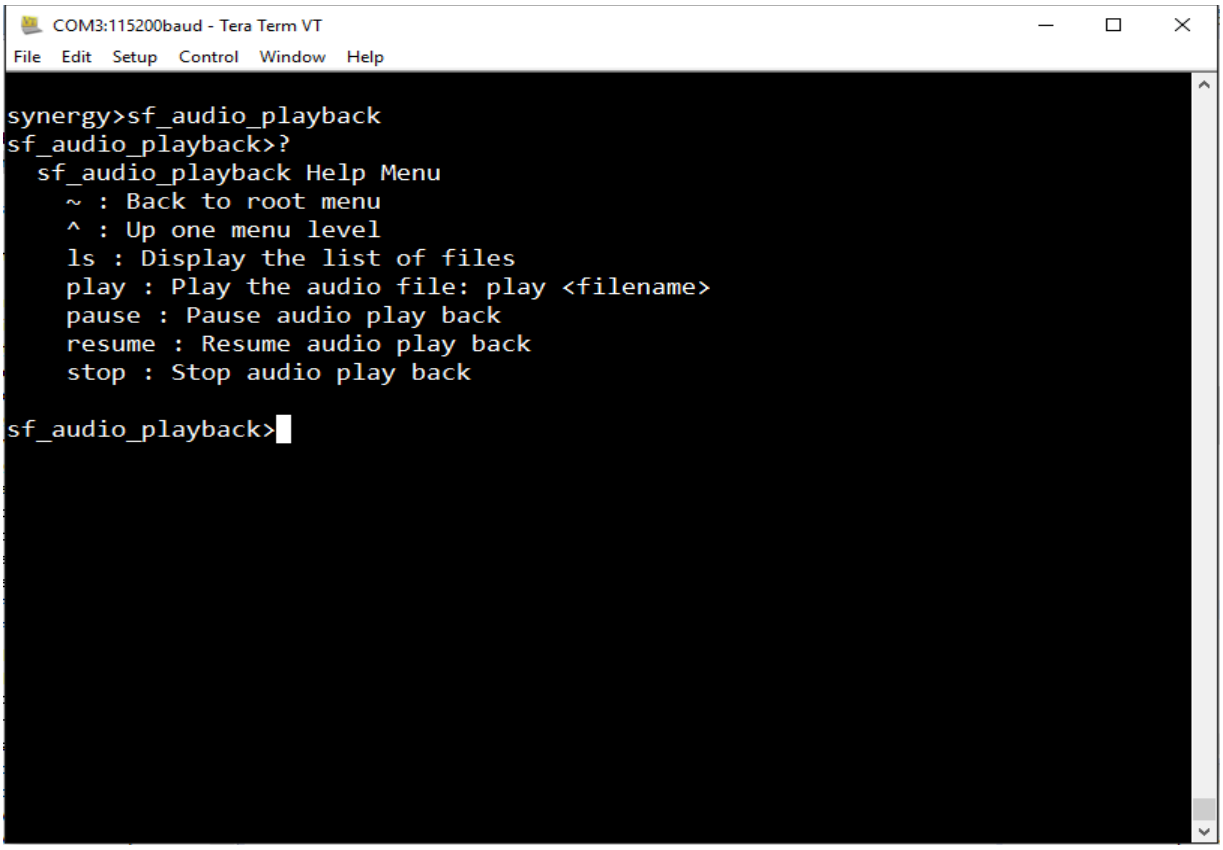
This Developer Example demonstrates the `play` API of the Audio Playback Framework. You can enable and configure the DAC to playback an audio file in .ogg format stored on an SD card and control the playback using the terminal. In addition to the Audio Playback Framework, the example also uses the FileX Adaptation Framework (FX_IO) and the SD/MMC card drivers.

6.2 Run the Audio Playback Framework application

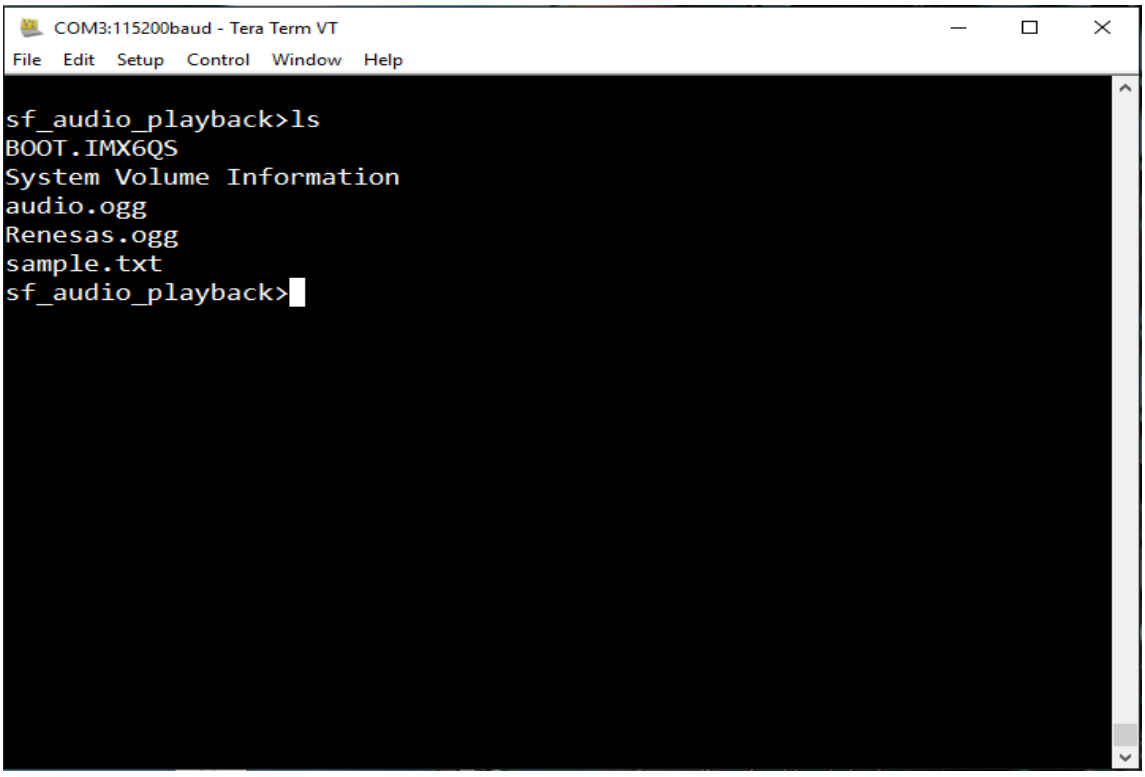
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the Audio Playback Framework application, follow these steps:

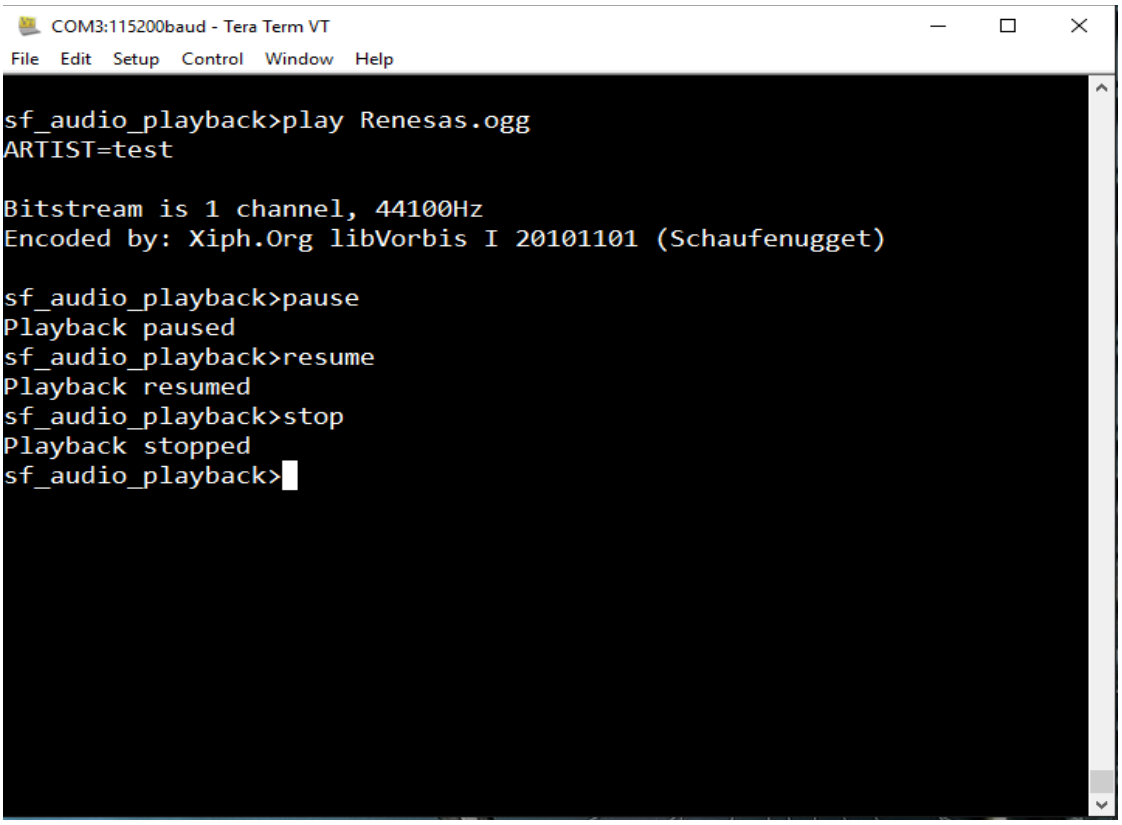
STEP 1: Type `sf_audio_playback` and press Enter to get the audio menu.



STEP 2: Type `ls` in the terminal and press enter to access the directory list of the SD card. For help, type `?` and press Enter.



STEP 3: Type `play Renesas.ogg` in the terminal and press Enter to perform audio playback of Renesas.ogg stored on the SD card. For help, type ? and press Enter. You can use `stop`, `pause` and `resume` in the same way.



6.3 Limitations

The Audio Playback Developer Example has the following limitations:

- The only supported audio format is .ogg.
- Audio files must be in mono format with a sample rate of 44.1 kHz.
- The `stop`, `pause`, and `resume` commands of this Developer Example work by manipulating the audio thread and do not use the Audio Playback APIs directly.

7. Developer Example: DAC HAL driver

7.1 Introduction

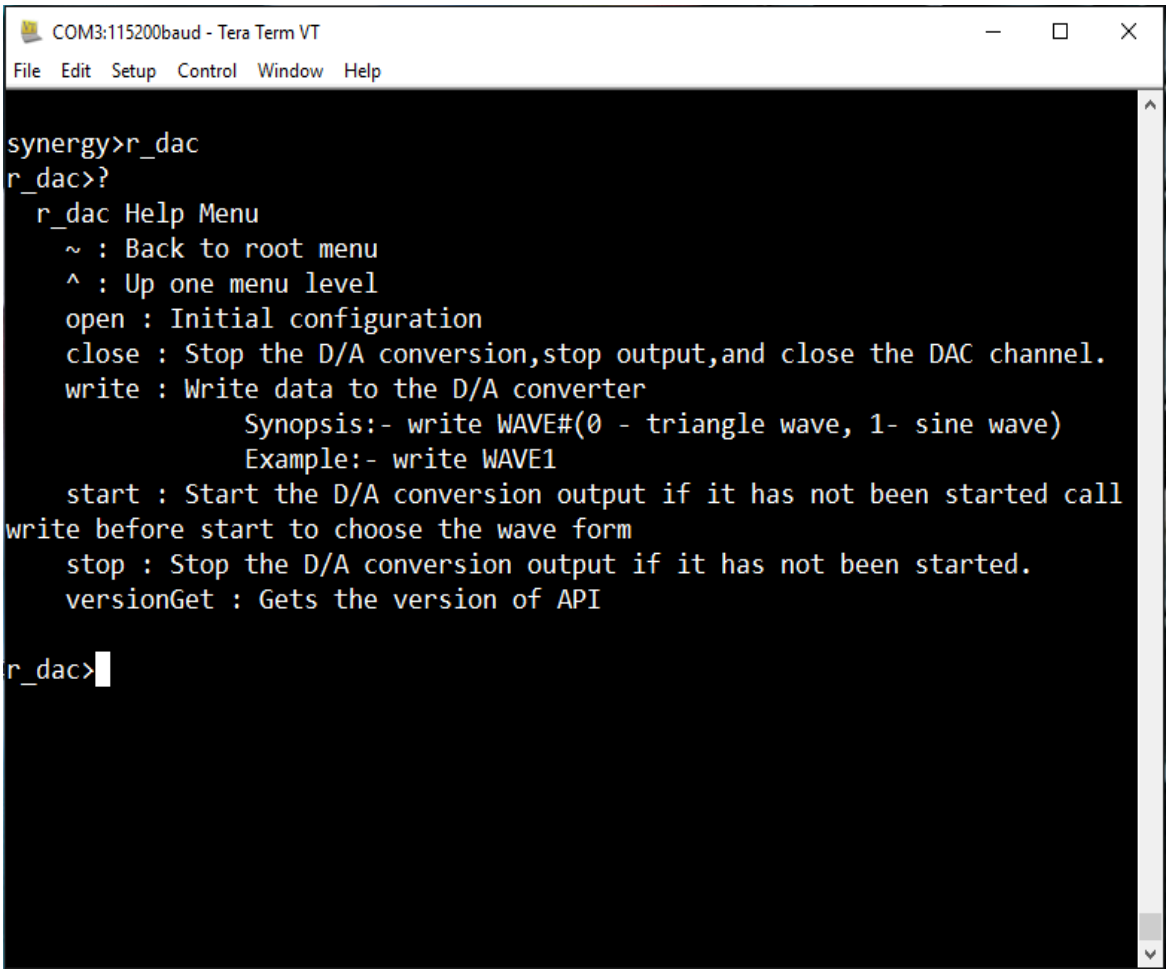
The Developer Example uses the APIs of the DAC HAL module from the terminal command line. The Developer Example generates a triangle and a sine wave on the DAC output.

7.2 Run the DAC HAL Driver application

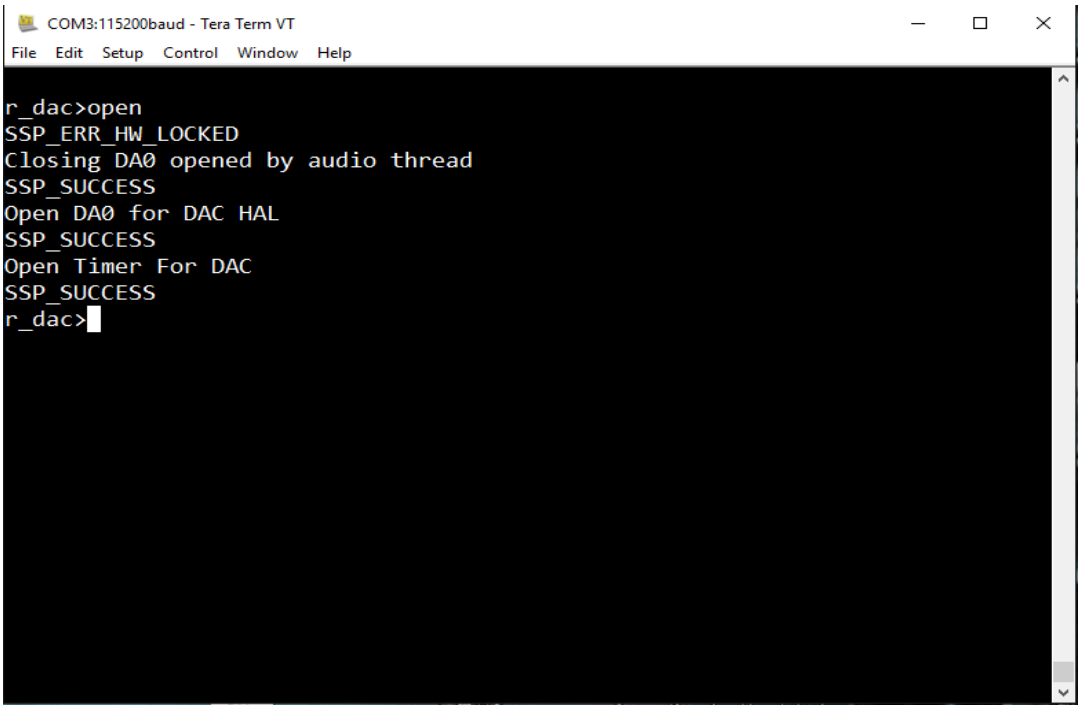
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the DAC HAL driver application, follow these steps:

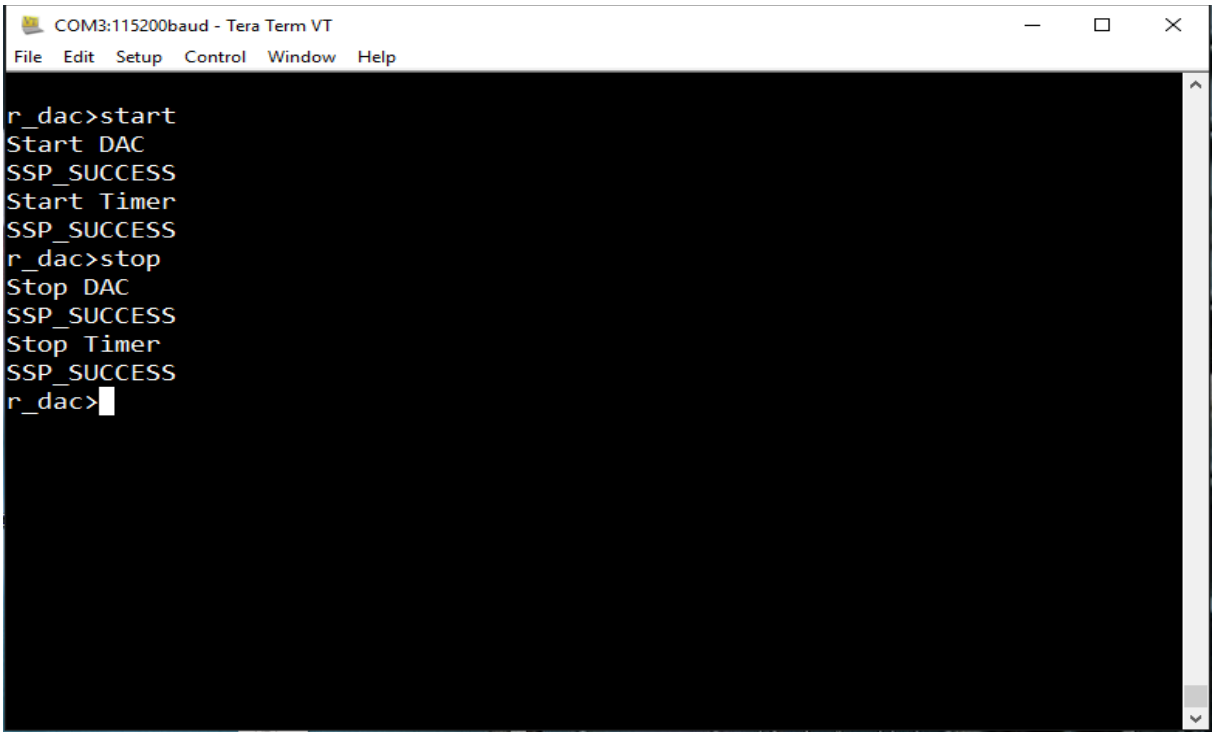
STEP 1: type `r_dac` and press Enter to get the DAC HAL menu.



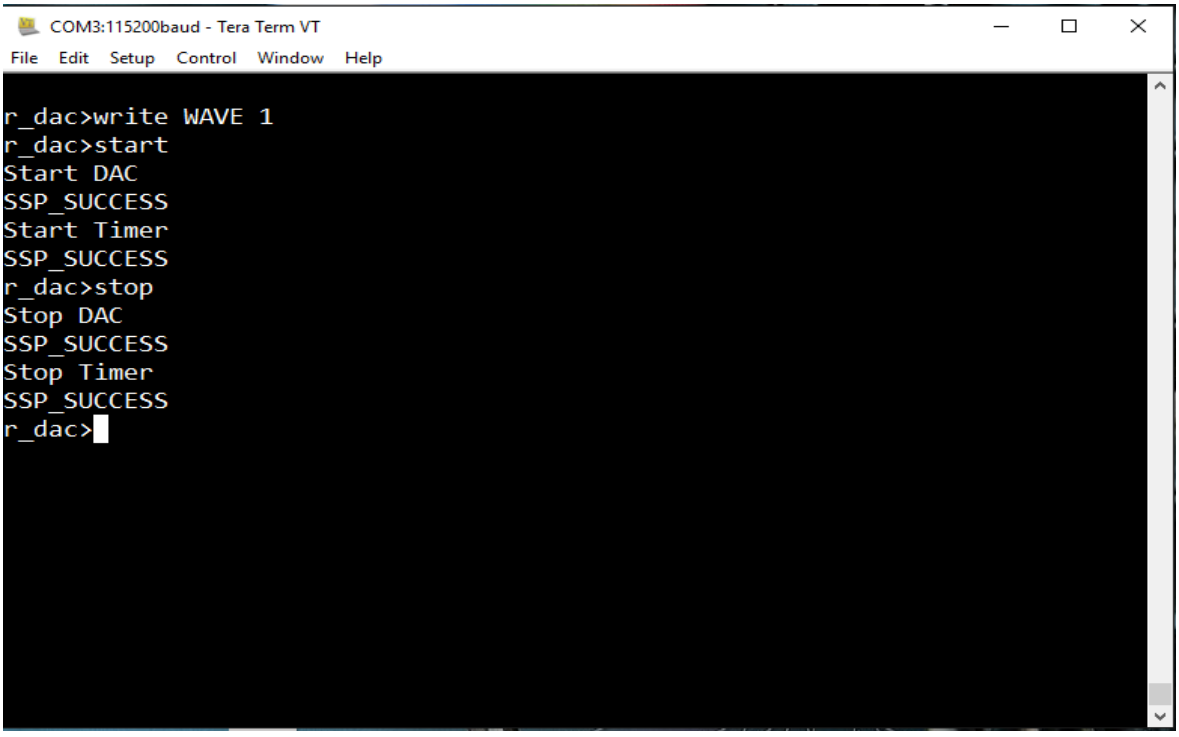
STEP 2: Type `open` in the terminal and press Enter to initialize the DAC HAL module. For help, type ? and press Enter.



STEP 3: Type `start` in terminal and press Enter to enable the DAC. For help, type ? and press Enter.



STEP 4: Type `write WAVE#` (#=0 for triangle waveform #=1 for sine wave form) in the terminal to select the waveform and then type `start` and press Enter to demonstrate the selected waveform's Digital-to-Analog output. For help, type ? and press Enter.



7.3 Limitations

The `close` command in this Developer Example does not execute the `close` API of DAC HAL module to prevent a conflict with the Developer Example for the Audio Playback Framework, which also requires the DAC HAL module.

8. Developer Example: AGT HAL driver

8.1 Introduction

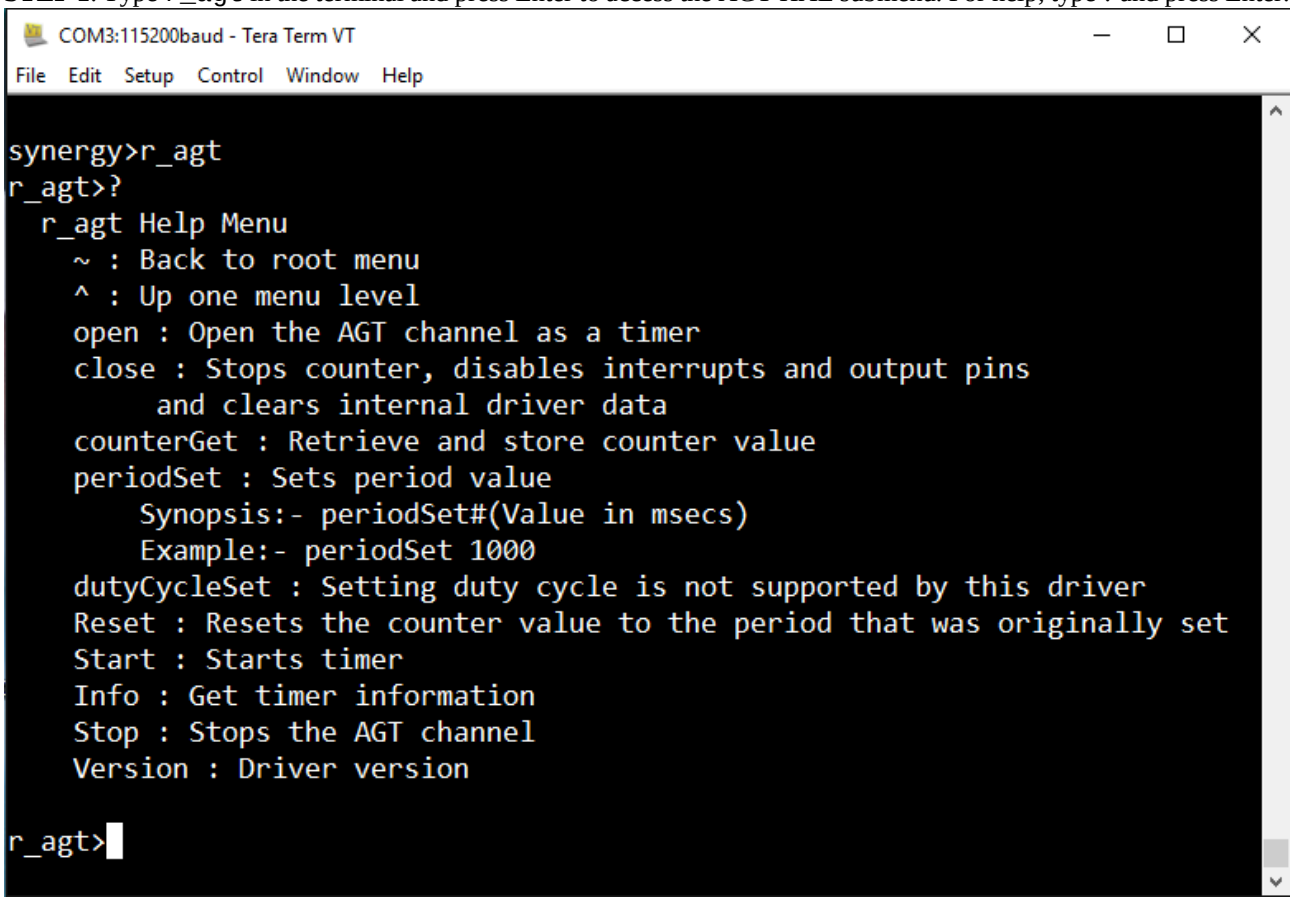
The AGT Hal driver Developer Example uses the periodic interrupt of the AGT. The interrupt toggles the LED1 on the DK-S7G2 board. The Developer Example also allows you to dynamically set the time period of the AGT timer from the command line interface while the timer is running.

8.2 Run the AGT HAL Driver application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the AGT HAL driver application, follow these steps:

STEP 1: Type `r_agt` in the terminal and press Enter to access the AGT HAL submenu. For help, type ? and press Enter.



STEP 2: Using the `open` and then `start` commands starts the AGT to run for the period that is configured in the Synergy Configuration tool and passed into `open`.

When the timer overflow interrupt occurs, LED1 on the DK-S7G2 board lights up. You can reconfigure the timer using `periodSet`

9. Developer Example: GPT HAL driver

9.1 Introduction

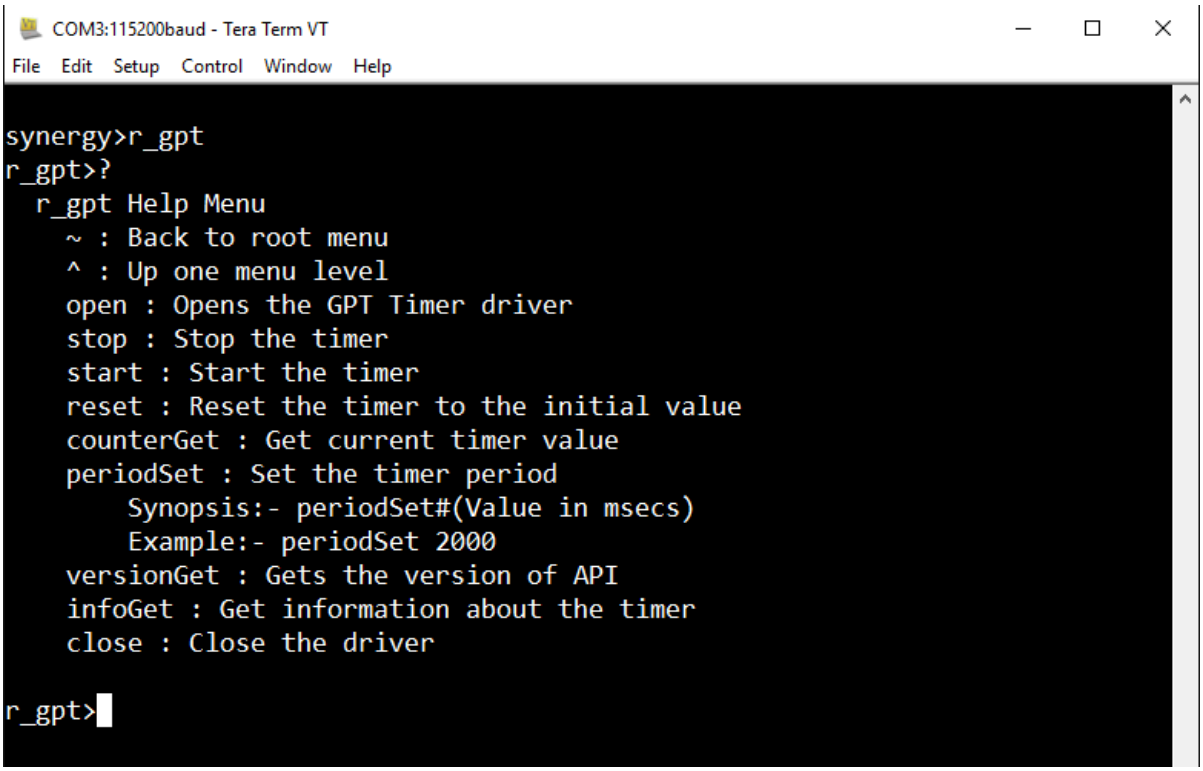
The GPT HAL driver Developer Example uses the GPT interrupt to toggle LED1 on the DK-S7G2 board.

9.2 Run the GPT HAL Driver applications

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the GPT HAL driver application, follow these steps:

STEP 1: Type `r_gpt` in the terminal and press Enter to access the GPT HAL submenu. For help, type ? and press Enter.



STEP 2: Use the `open` and then `start` commands to run the GPT for the period configured in `periodSet`. When the timer overflow interrupt occurs, LED1 on the DK-S7G2 board lights up. You can reconfigure the timer using `periodSet`.

10. Developer Example: CRC HAL driver

10.1 Introduction

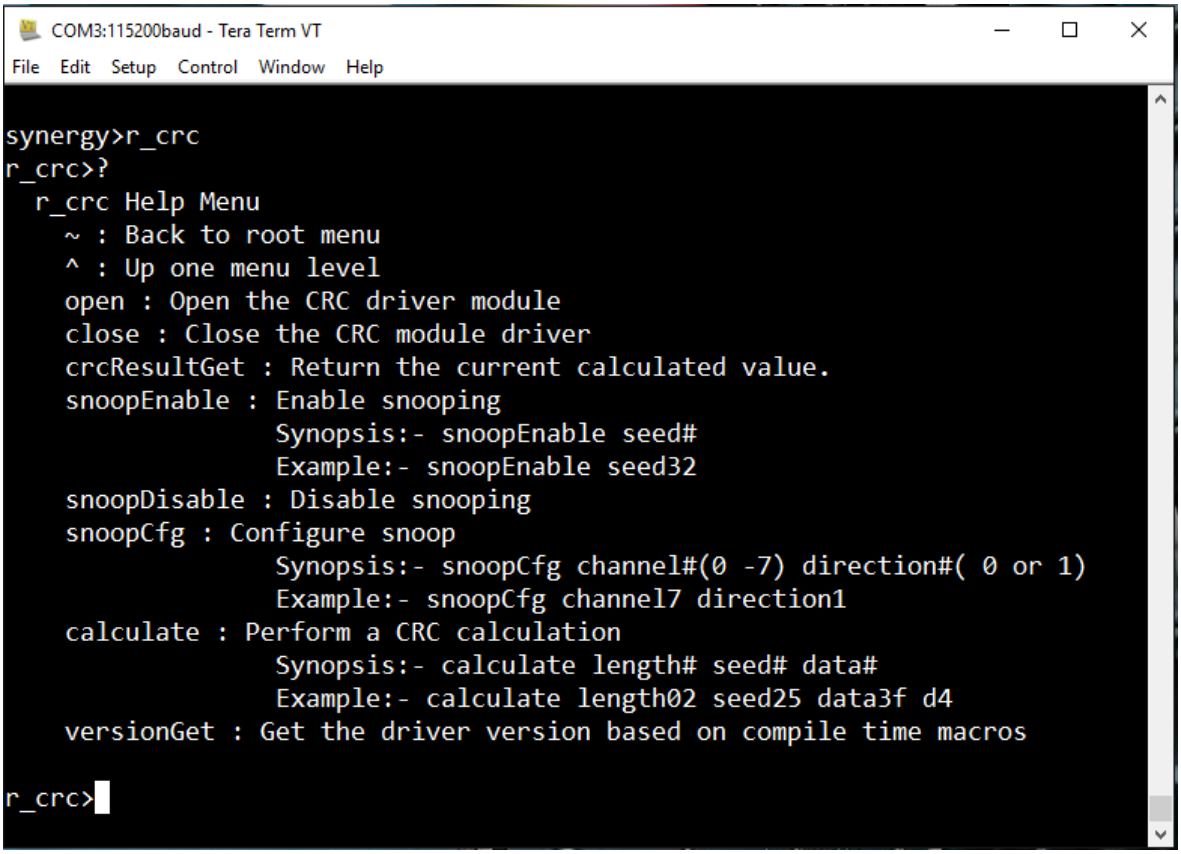
The cyclic redundancy check (CRC) detects errors in a dataset. The Developer Example uses the Snooping API function of the CRC HAL Module. The snoop function monitors read and writes to specific addresses. This function is useful in applications that require CRC code to be generated automatically in certain events, such as monitoring writes to the serial transmit buffer and reads from the serial receive buffer. The Developer Example uses the SCI I2C channels as an example.

10.2 Run the CRC HAL Driver application

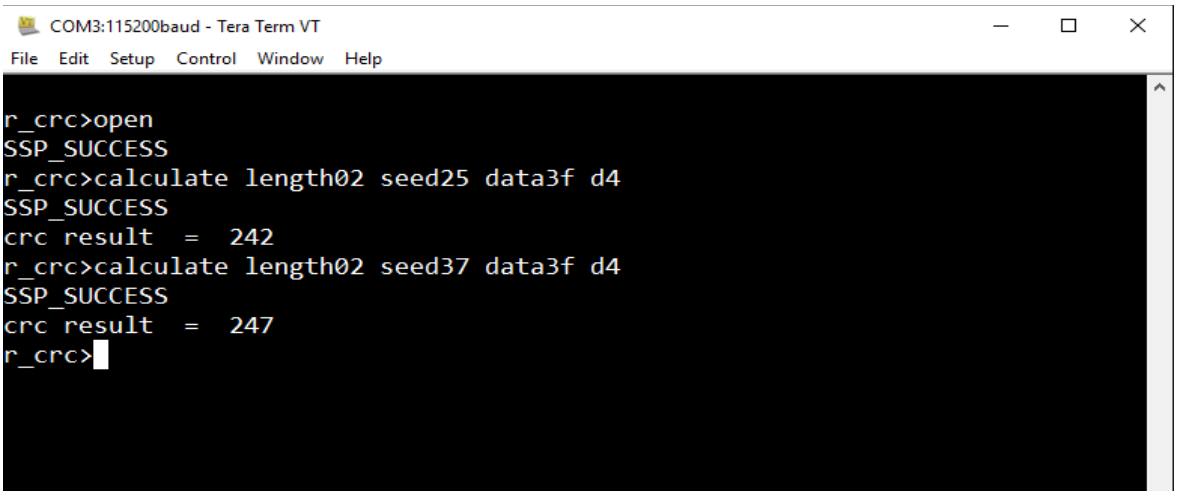
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the CRC HAL driver application, follow these steps:

STEP 1: Type `r_crc` in the terminal and press Enter to access the CRC HAL submenu. For help, type ? and press Enter.

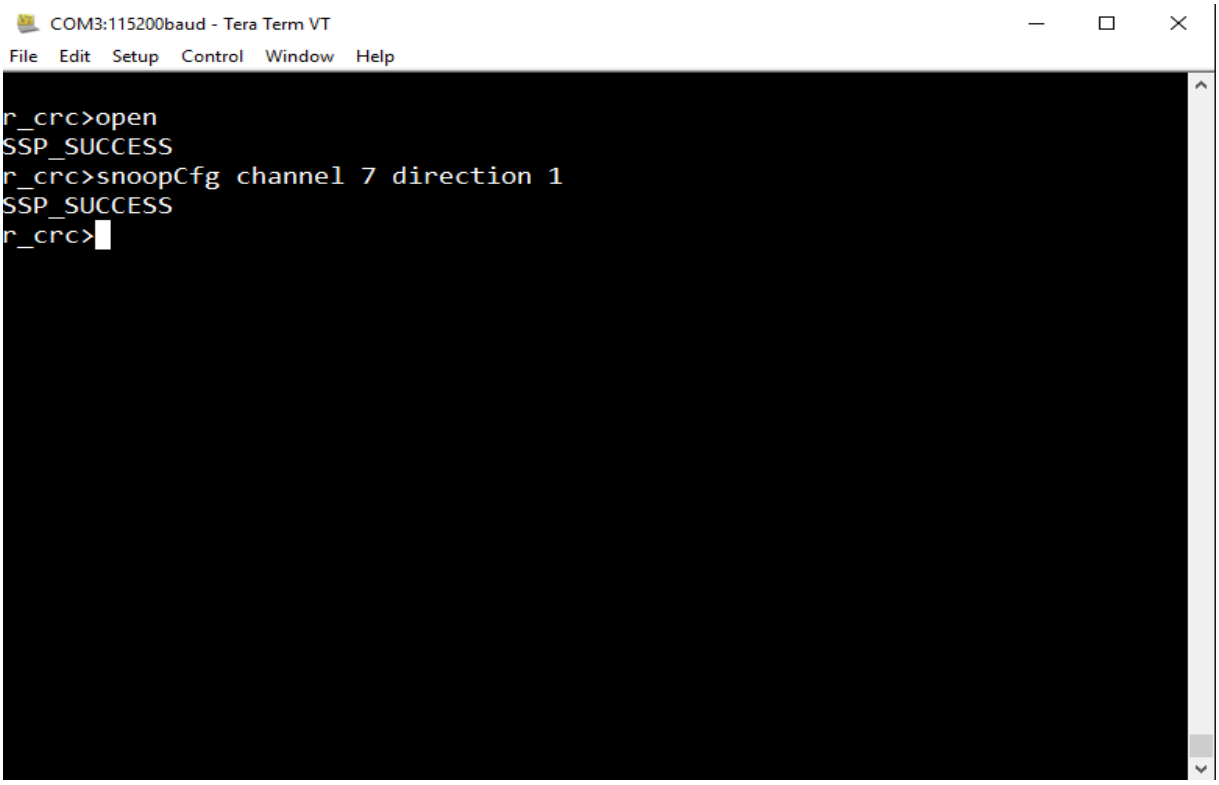


STEP 2: Invoke each menu item with the corresponding arguments to exercise the APIs. The example shows the CRC `calculate` command with length, seed and data option. It calculates CRC for the given length of data with a specific seed.

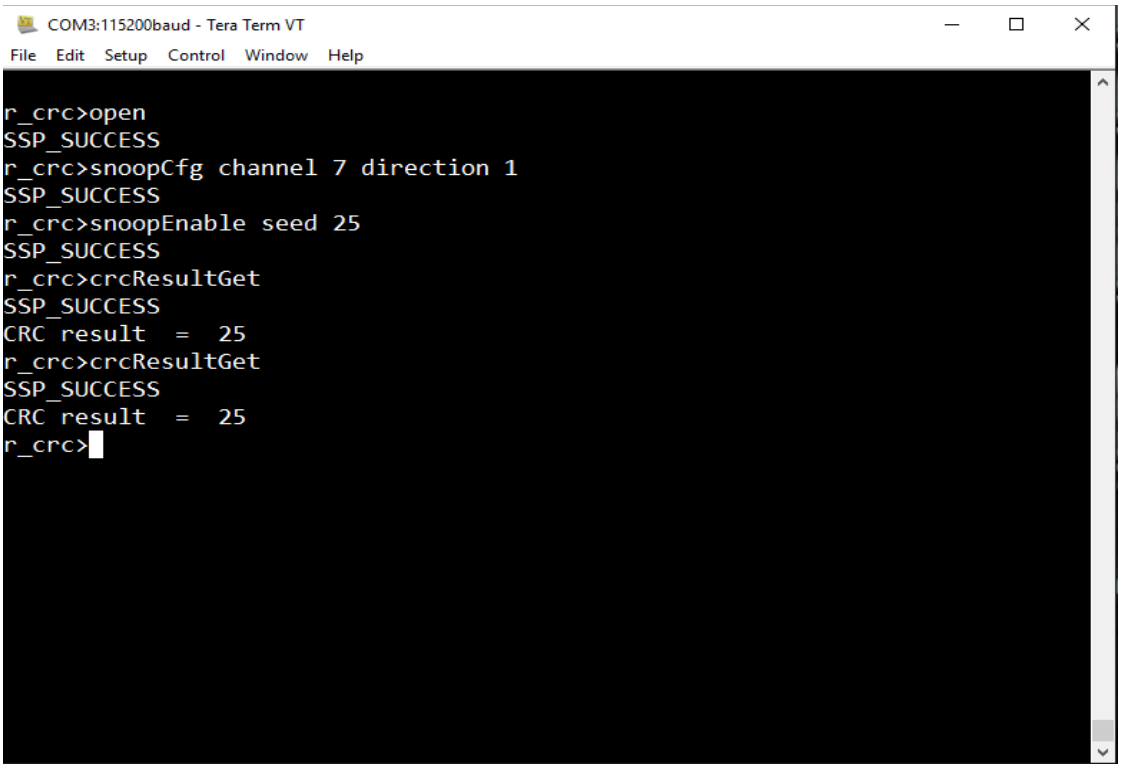


The following example shows the CRC snoop operation using `snoopCfg` and `snoopEnable` command along with running an I2Cframework application.

1. Enter `r_crc` followed by `open` and `snoopCfg` with channel and direction details



2. To validate snoop operation of r_crc any one SCI channel must be configured with respect to CRC snoop. In Developer Example sf_i2c uses SCI channel 7 so run the I2C Framework application from the sf_i2c menu and switch back to the r_crc menu and execute the following commands in sequence.
    1. `open`
    2. `snoopCfg`
    3. `snoopEnable`
    4. `crcResultGet`



11. Developer Example: Flash HAL driver

11.1 Introduction

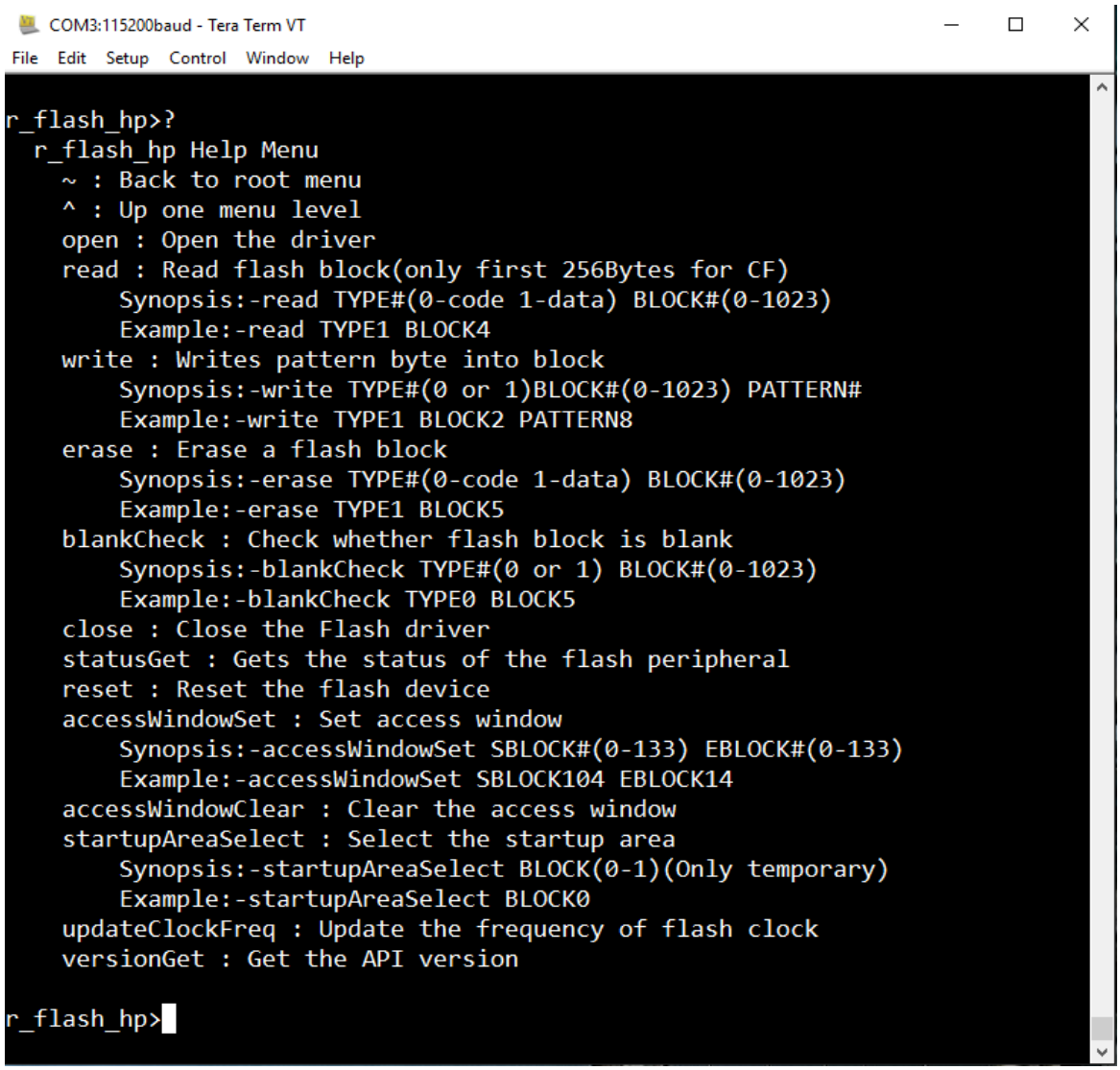
This example uses the high performance flash (flash HP) on the S7G2. The on-chip Flash consists of a code flash and a data Flash. The address range for code flash is 0x0000000 – 0x00400000 and the address range for data flash is 0x40100000 – 0x40110000. Code flash has blocks from 0-133, which is a total of 134 blocks. Data Flash has blocks from 0-1023 with a total of 1024 blocks. The code flash is defined as TYPE0 and data flash as TYPE1 in the Developer Example. The Flash Developer Example restricts the write and erase operations to certain code block regions since they may corrupt the Developer Example code itself. The Developer Example application will issue a warning message if you try to access a prohibited block and will not write to or erase that particular block.

11.2 Run the Flash HAL Driver application

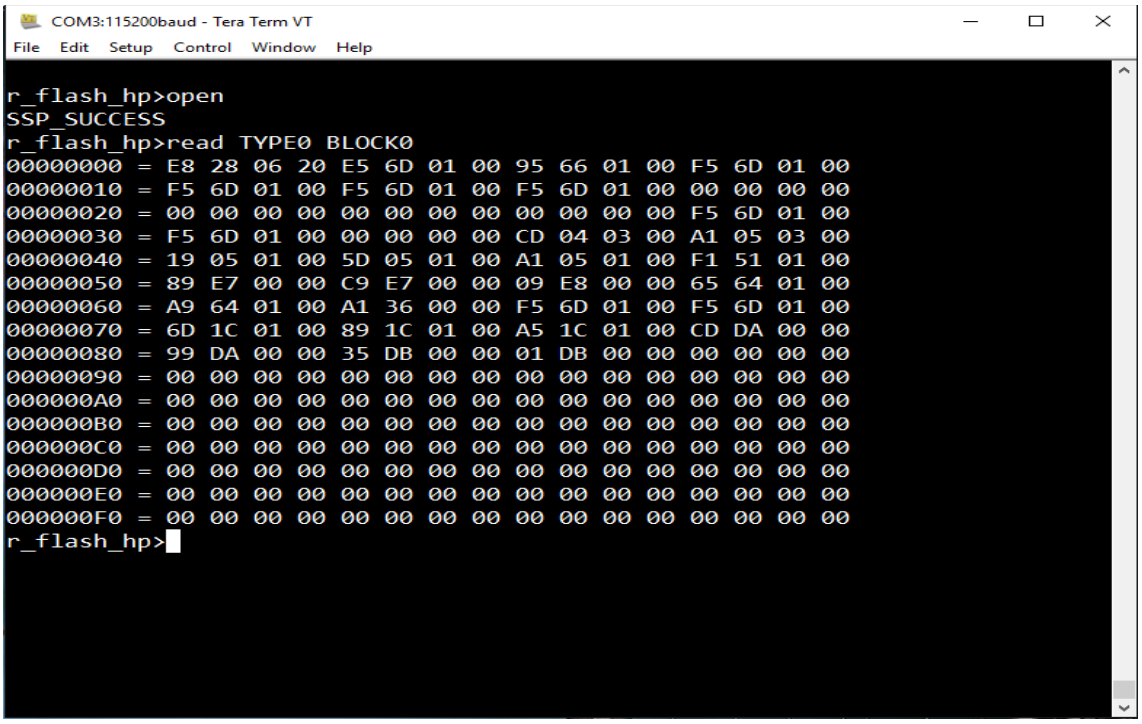
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the Flash HAL driver application, follow these steps:

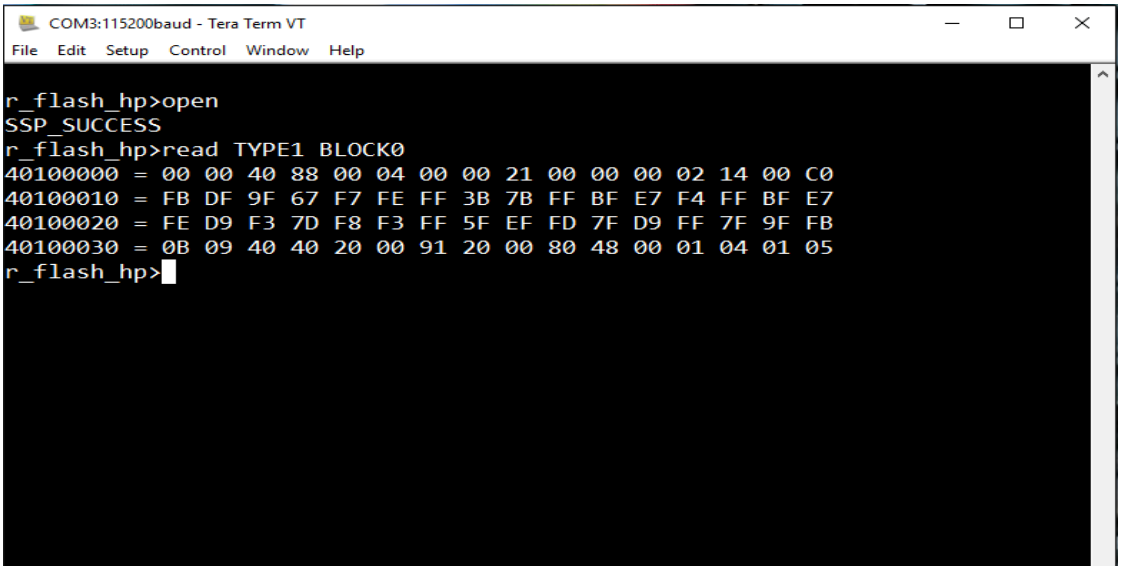
STEP 1: Type r_`flash_hp` and press Enter to access the flash HAL submenu. For help, type ? and press Enter.



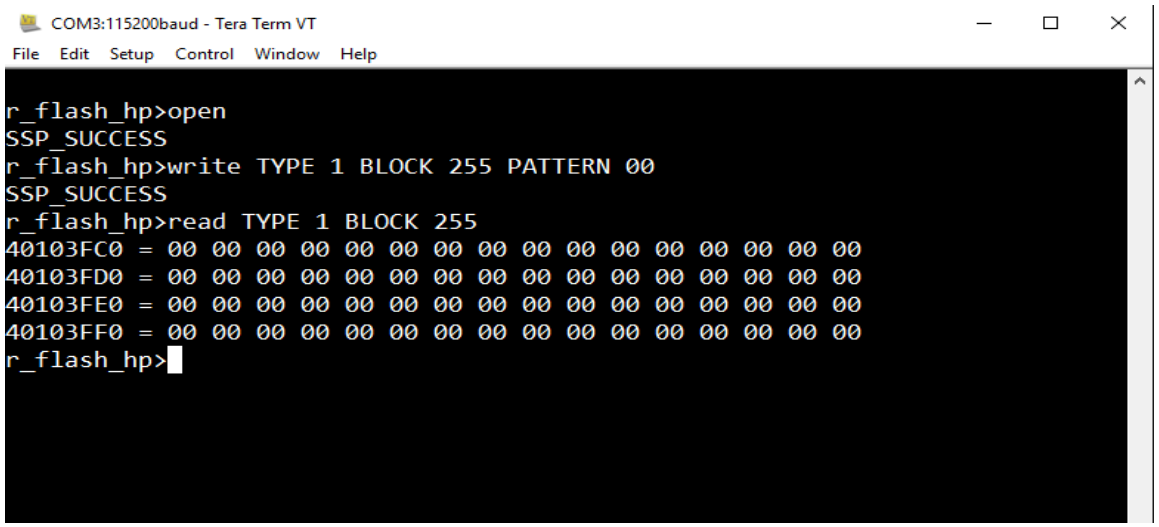
STEP 2: Use the `read` command to read from block0 of the code block.



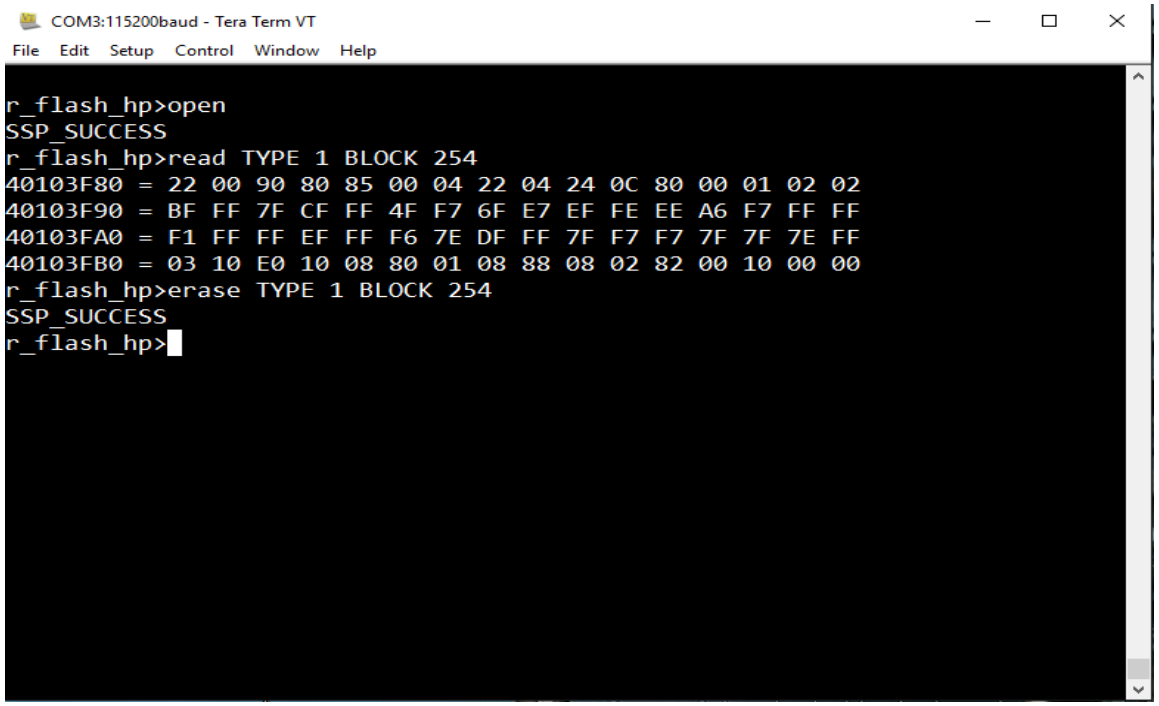
STEP 3: Use the `read` command to read from block0 of the data block.



STEP 4: Use the `write` command to write to block255 of the data block with any pattern value. The Developer Example disables the write into code flash area to protect from corrupting the code that runs the Developer Example program.



STEP 5: Use the `erase` command to erase to block255.



11.3 Limitations

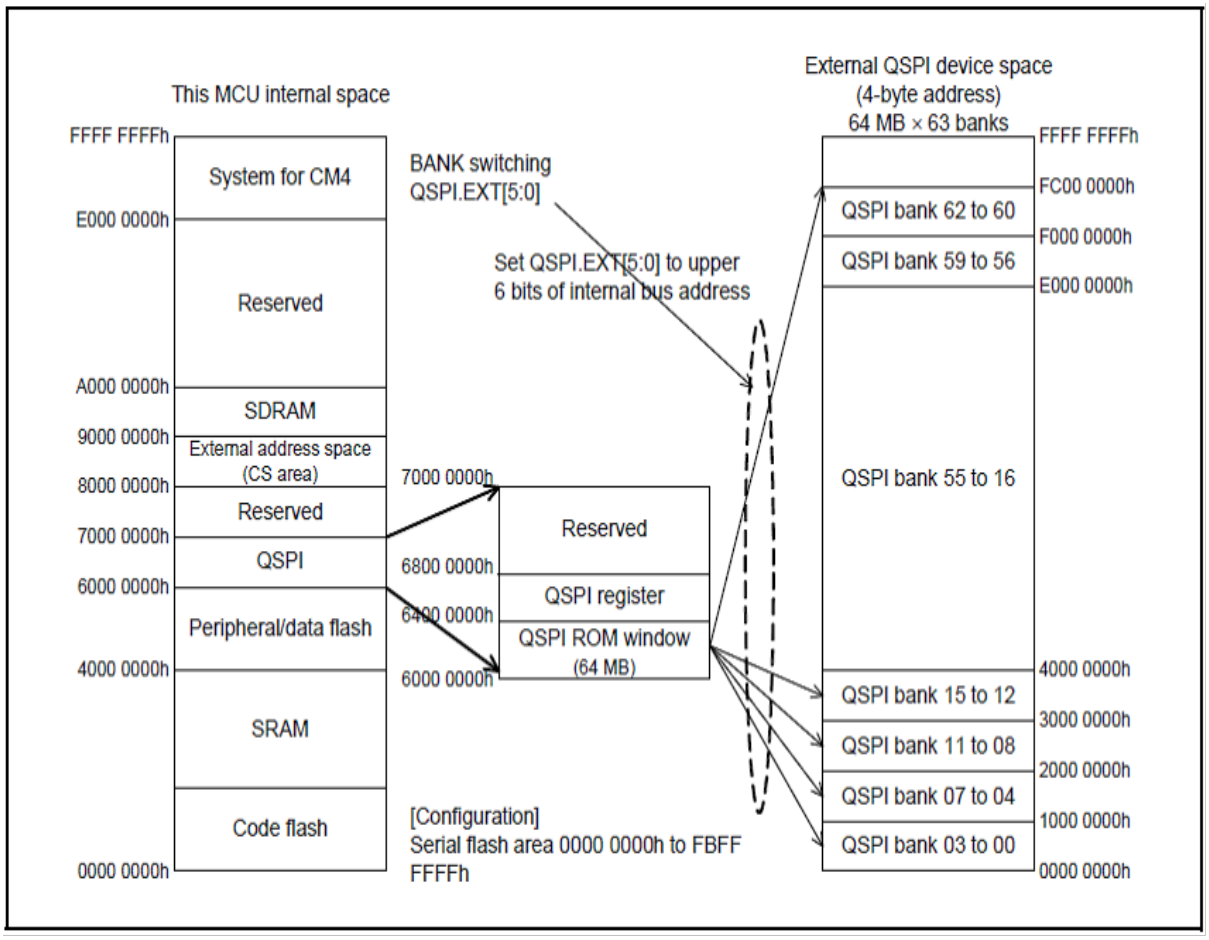
The Flash HAL Developer Example has the following limitations:

- The size of a block of data flash is 64 bytes. The size of a block of code flash is either 8 KB (BLOCK0 - BLOCK7) or 32 KB (BLOCK8 - BLOCK133). Since it is not practical to access huge volume of data (8 KB – 32 KB) over the command line interface, the write sizes are hardcoded to 256 bytes for code flash and 64 bytes for data flash. **This is a limitation in the Developer Example and not of the flash driver.**
- The current Developer Example will restrict write and erase operations to certain code block regions since it may corrupt the Developer Example code itself.

12. Developer Example: QSPI HAL driver

12.1 Introduction

The DK-S7G2 includes an external QSPI NOR flash from Micron (N25Q256A). This chip provides 256 MB of NOR flash with Execute-In-Place (XIP) capability. The SPI flash is mapped to the address 0x6000 0000 Hex to 0x63FF FFFF Hex (64 MB) of the MCU address space. Since the SPI flash is 256 MB in size, it is addressed in 4 byte address mode and accessed as 4 banks of 64 MB each.

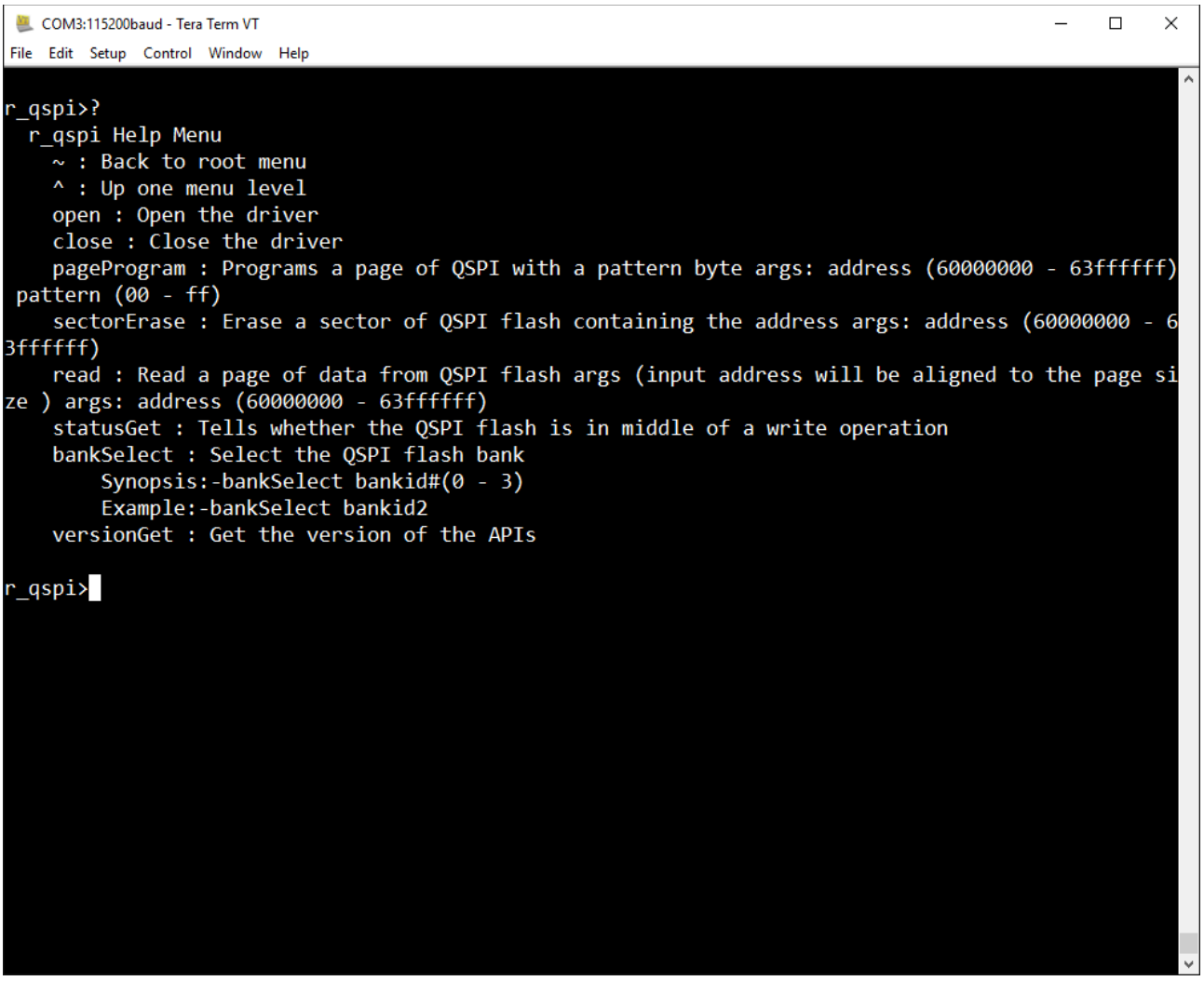


12.2 Run the QSPI HAL Driver application

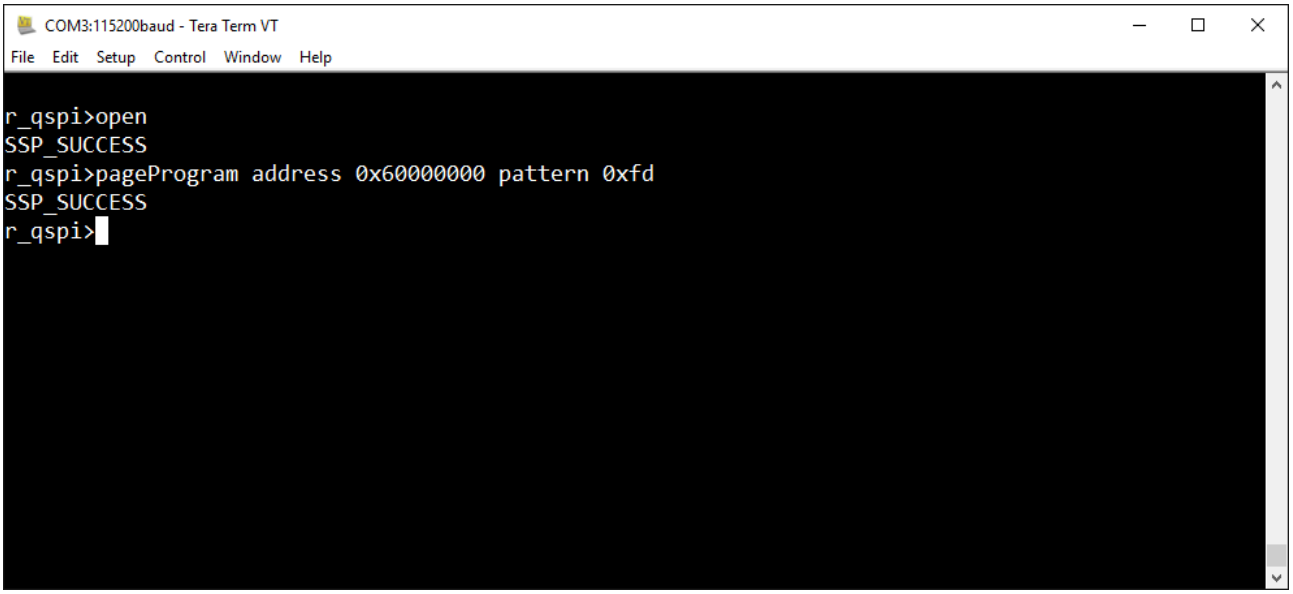
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the QSPI HAL driver application, follow these steps:

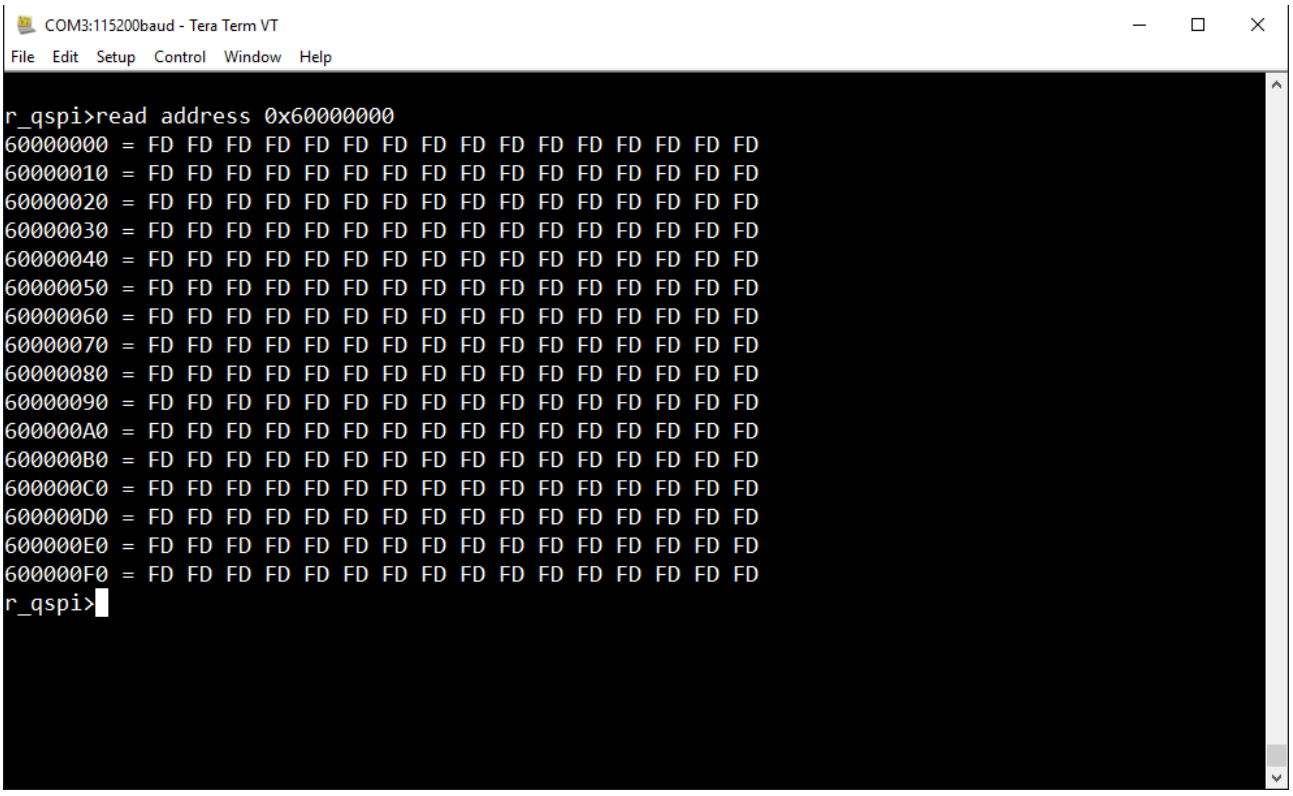
STEP 1: Type `r_qspi` and press Enter to access the QSPI flash HAL sub menu. For help, type ? and press Enter.



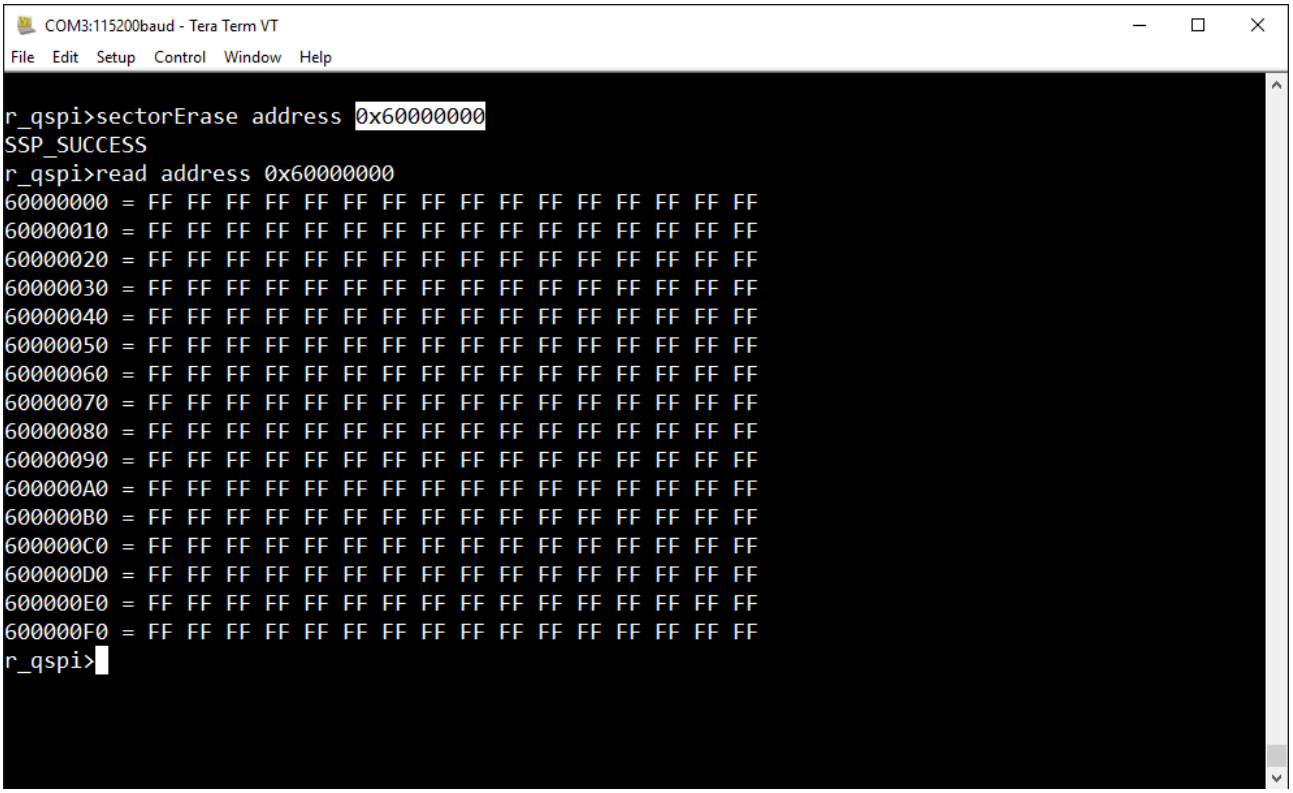
STEP 2: Use the `pageProgram` command to program a page of QSPI flash by filling it with a pattern byte. (The size of a page is 256 Bytes for this implementation of driver interface). For ease of use, the entire page containing the address will be filled with the byte pattern for a given address. This is the behavior of the command in the Developer Example and not of the driver itself.



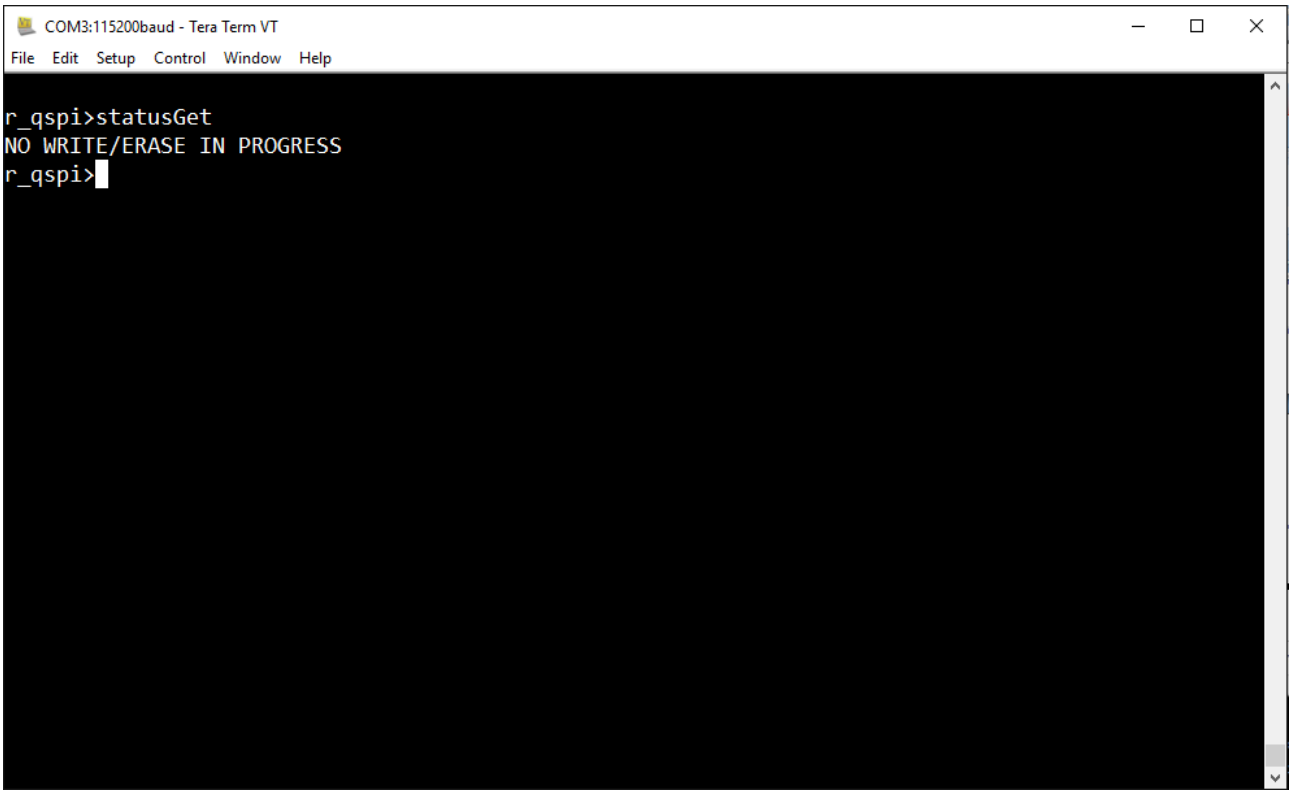
STEP 3: Use the `read` command to read a page of data from QSPI flash. For ease of use, the entire page containing the address will be read and displayed for a given address. This is a behavior of the Developer Example command and not of the driver itself.



STEP 4: Use the `erase` command to erase a sector of QSPI flash. In this implementation of the driver, the sector size is 4 KB. Passing any address within a sector will erase the whole sector.



STEP 5: Use the `statusGet` command to check the status of the QSPI erase/write. The status will return whether the device is busy doing a write/erase cycle.



12.3 Limitations

The QSPI HAL Driver Development Example has the following limitations:

- The `read` and `pageProgram` commands are aligned to the device page size. It's an implementation behavior of the command line interface command and not a driver feature.
- The device supports 256 MB of memory. It can be accessed as 4 Banks of 64 MB each. The current version of the QSPI driver allows only access to bank 0 even when the bank selects API returns success for banks 0 -3.
- To perform a successful `pageProgram` command, the pages must be erased first. If a `pageProgram` is requested on a non-erased sector, the operation will fail, but the driver nonetheless will return success error code. This is a limitation of the driver and not of the Developer Example application.

13. Developer Example: RTC HAL driver

13.1 Introduction

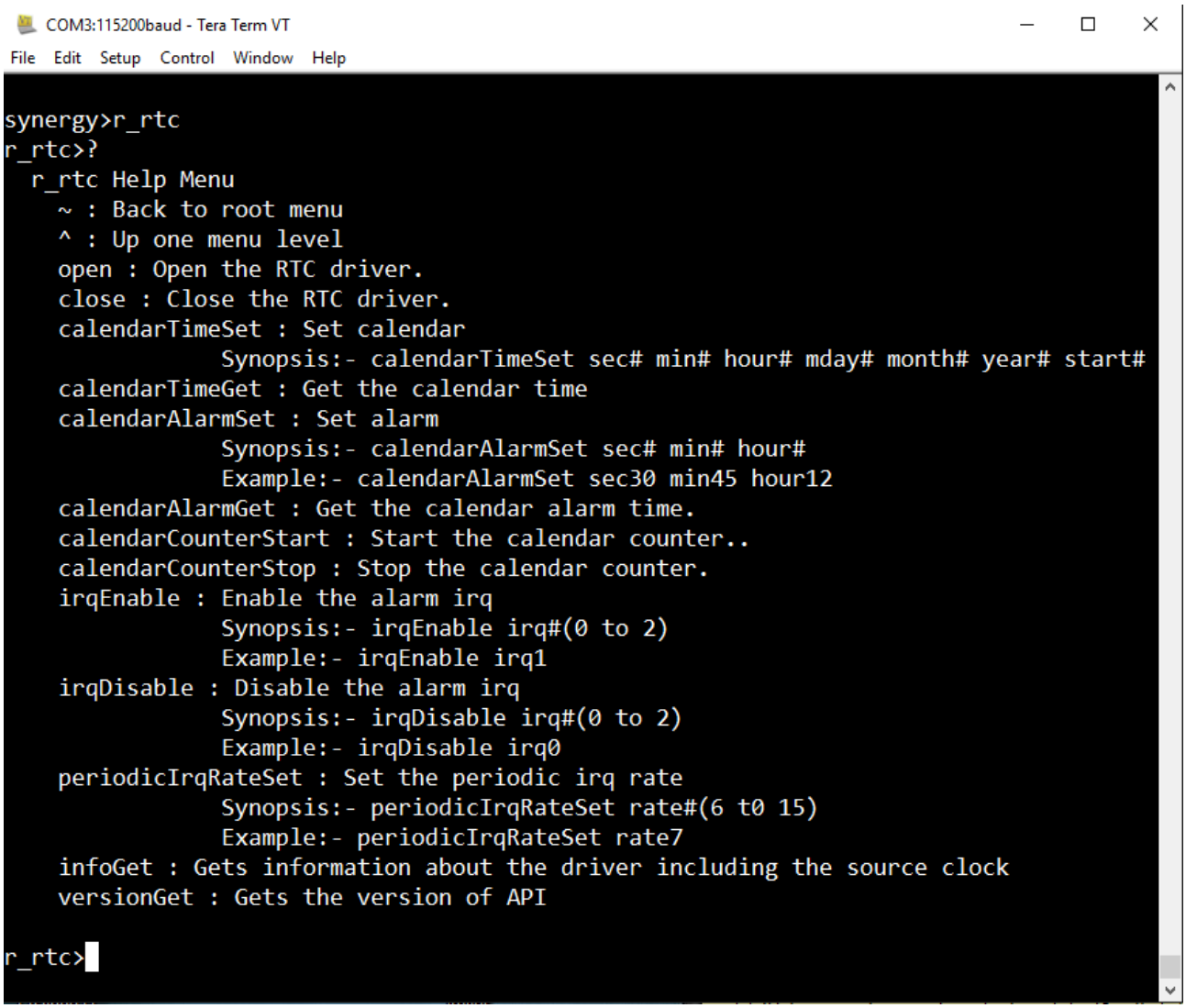
The Real Time Clock (RTC) Developer Example application uses the Low-speed on-chip oscillator as a clock source. The application allows to set the RTC configuration parameters such as time capture, alarm and periodic interrupt from the command line interface. Interrupts are handled by the callback function which inverts the state of the LED1 on the DK-S7G2 board.

13.2 Run the RTC HAL Driver application

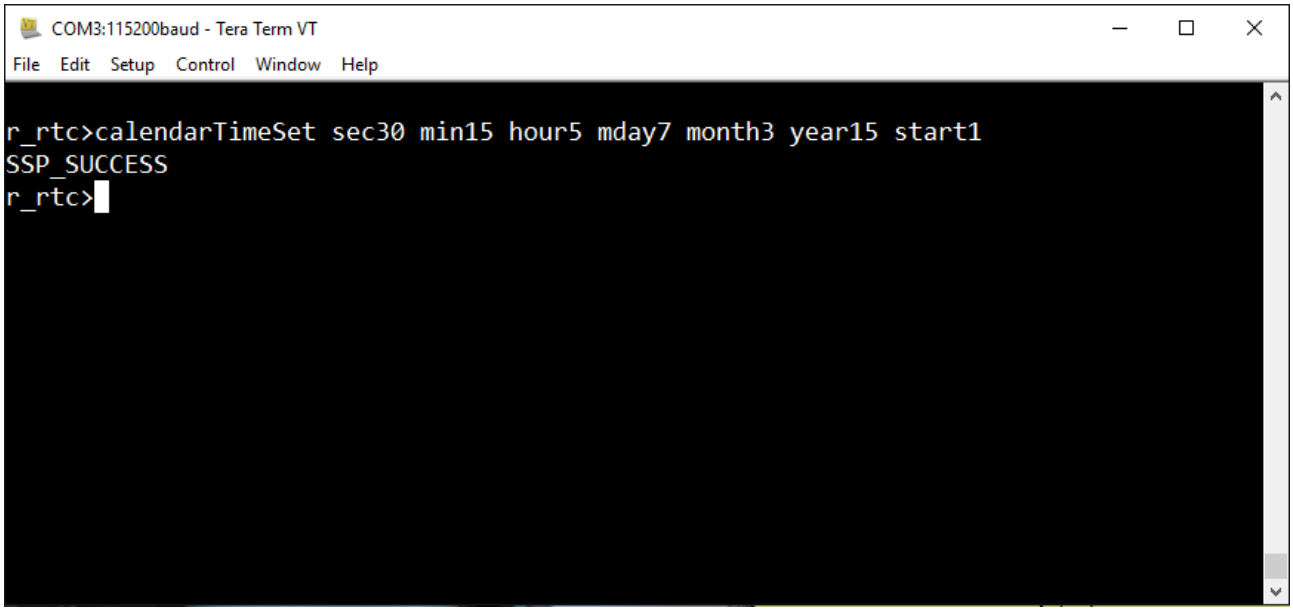
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the RTC HAL driver application, follow these steps:

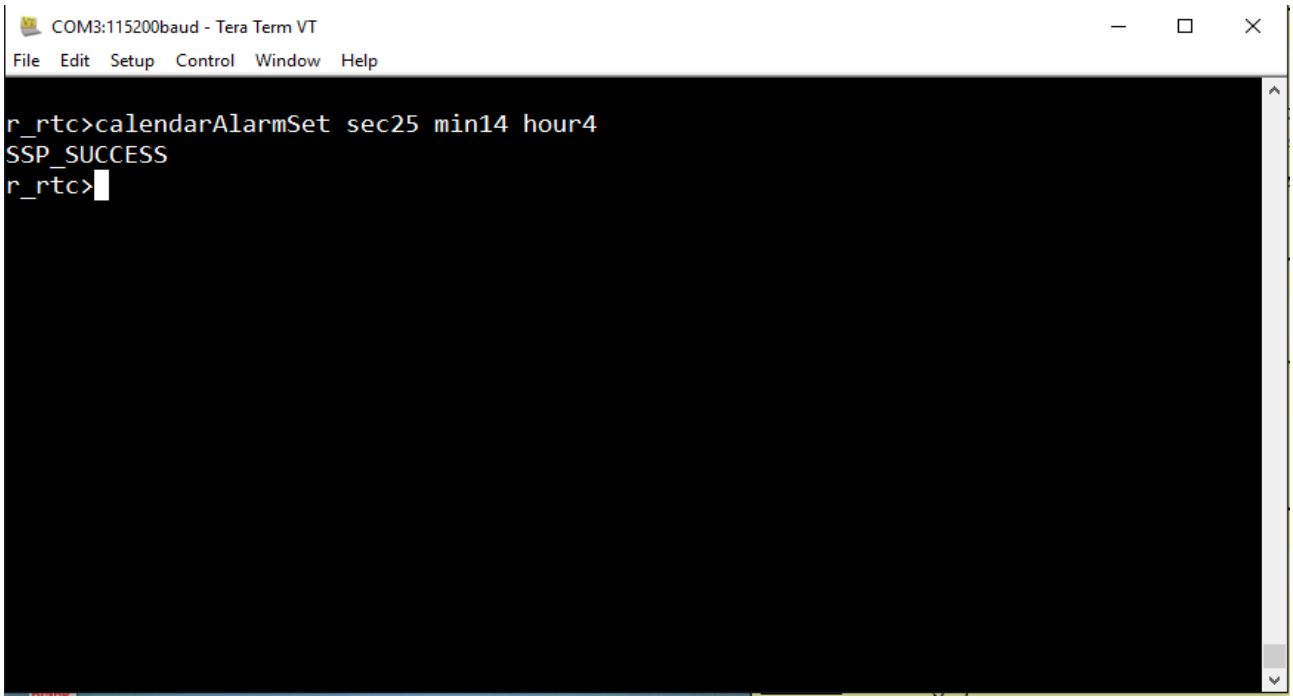
STEP 1: Type `r_rtc` in the terminal and press Enter to access the RTC HAL submenu. For help, type ? and press Enter.



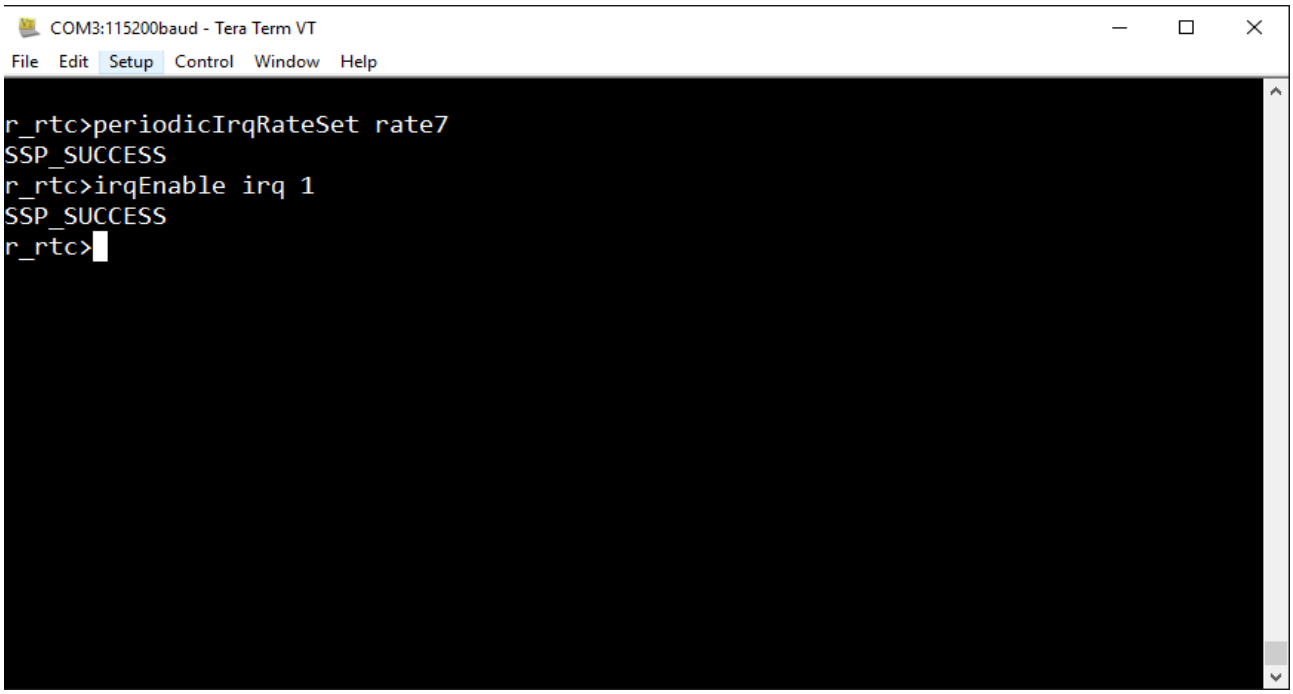
STEP 2: Type command `open` followed by `calendarTimeSet` to set the time.



STEP 3: Type `calendarAlarmSet` to set the alarm.



STEP 4: Type `periodicIrqRateSet` command followed by `irqEnable irq1` (1 for periodic).LED1 on the DK-S7G2 board blinks with the programmed period (`rate13` for a period of 0.5 sec).



14. Developer Example: SCI I2C Framework

14.1 Introduction

The SCI I2C Framework Developer Example application accesses two I2C devices (I/O expanders connected to a bank of LEDs – see Developer Example: SCI I2C HAL driver) present on S7G2-DK board which are synchronized by the SCI I2C Framework. The application uses two threads which operate on each slave device. After opening each I2C slave device, each thread performs a write to the specific registers of the I2C device. The register state is displayed by

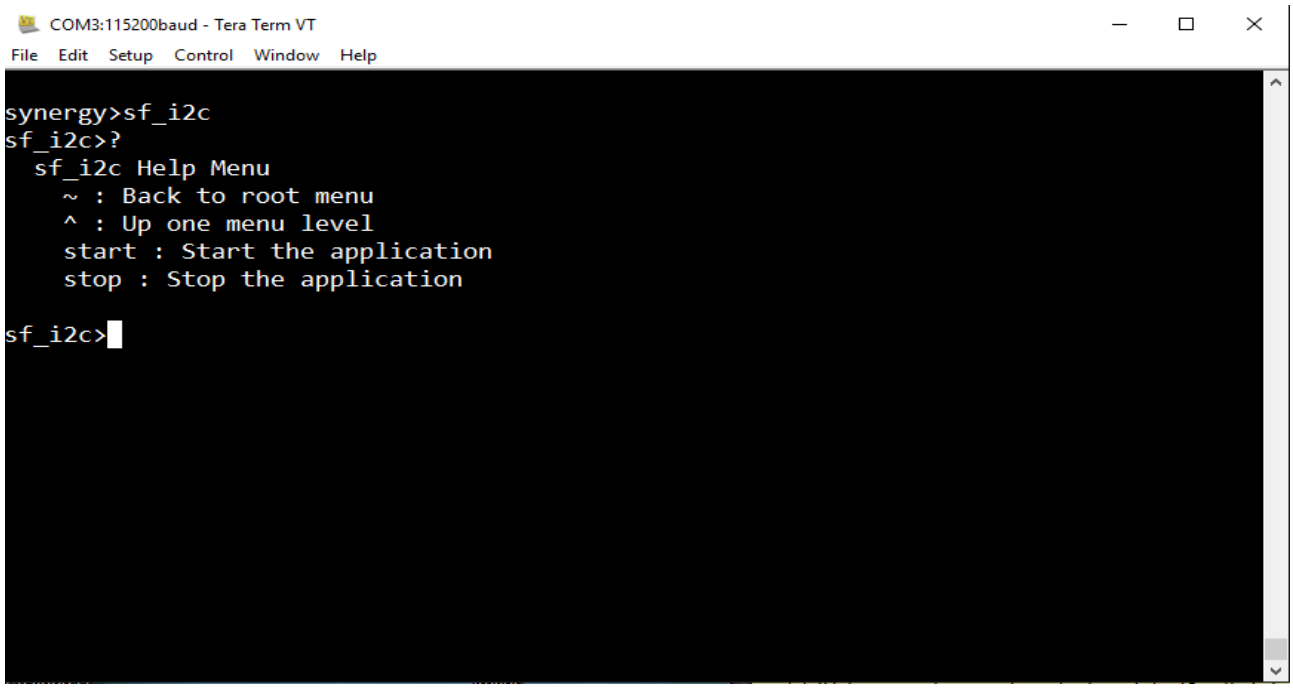
the LEDstateasON or OFF. The threads are synchronized by the I2C framework, which causes the Red and Green LEDs to light up synchronously.

14.2 Run the SCI I2C Framework application

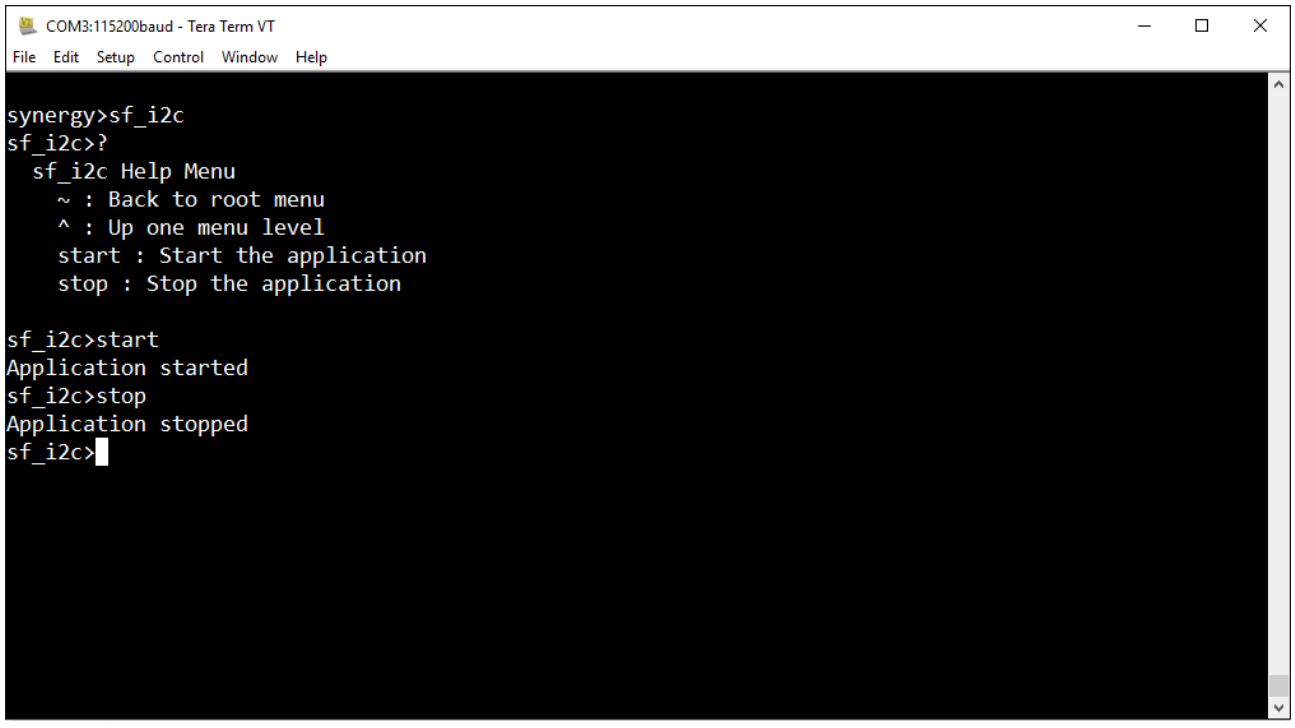
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the SCI I2C HAL driver application, follow these steps:

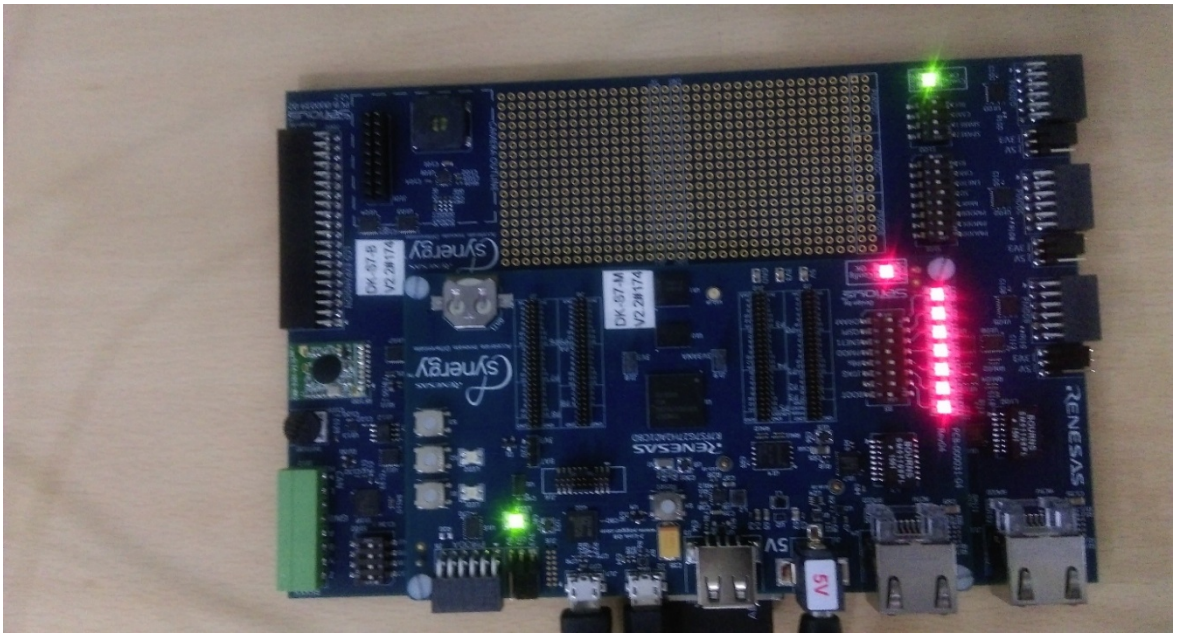
STEP 1: Type `sf_i2c` and press Enter to get the SCI I2C Framework application menu. For Help, type ? and press Enter.



STEP 2: Type `start` in the terminal and press Enter to start I2C sample application which toggles two sets of LEDs via two different I2C I/O expanders on S7G2-DK board.



STEP 3: Type `stop` in the terminal and press Enter to stop application which stops the toggling of LEDs.



15. Developer Example: SCI I2C HAL driver

15.1 Introduction

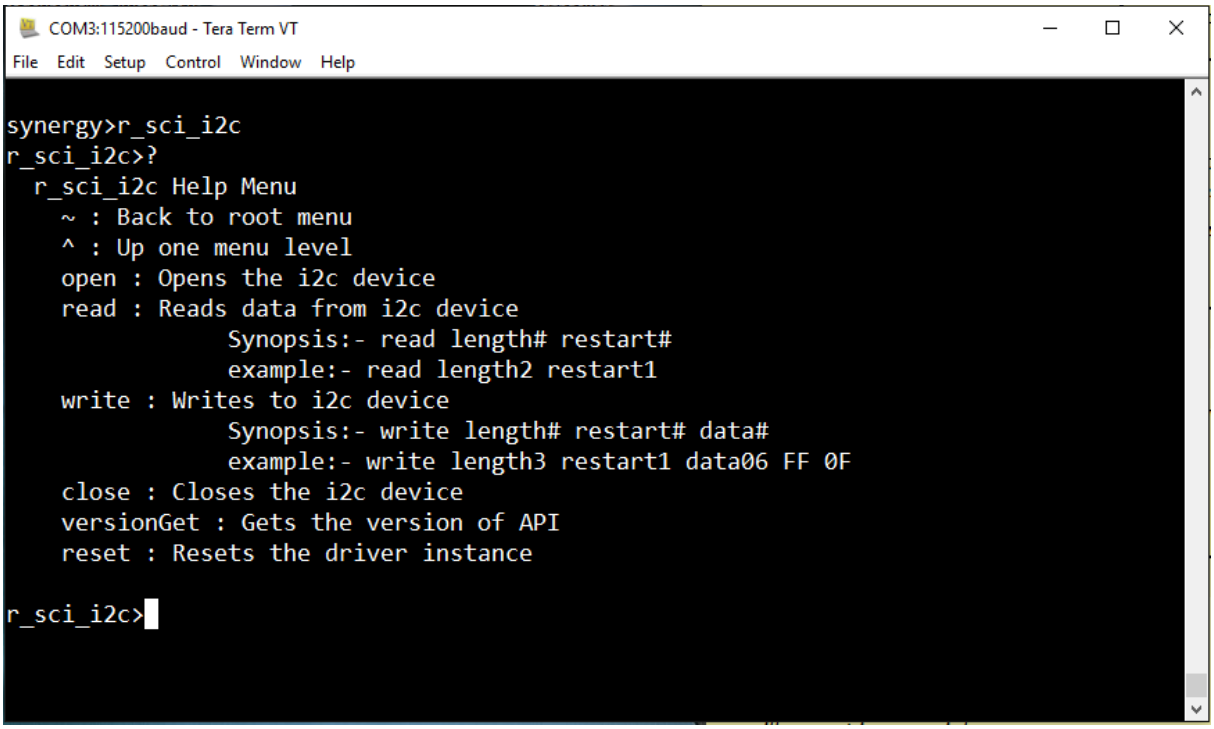
On the DK-S7G2 Development Kit, the SCI I2C bus is connected to an I2C controlled I/O expander (part PCAL9535A). This application configures the I/O expander on the DK-S7G2 as a slave device. The I/O expander toggles the LEDs on/off as the result of each write operation to the expander's registers, so you can visually see the output of the write operation. The slave address is 0x27 with 16 sub registers. For details of the I/O expander, see the PCAL9535A datasheet.

15.2 Run the SCI I2C HAL Driver application

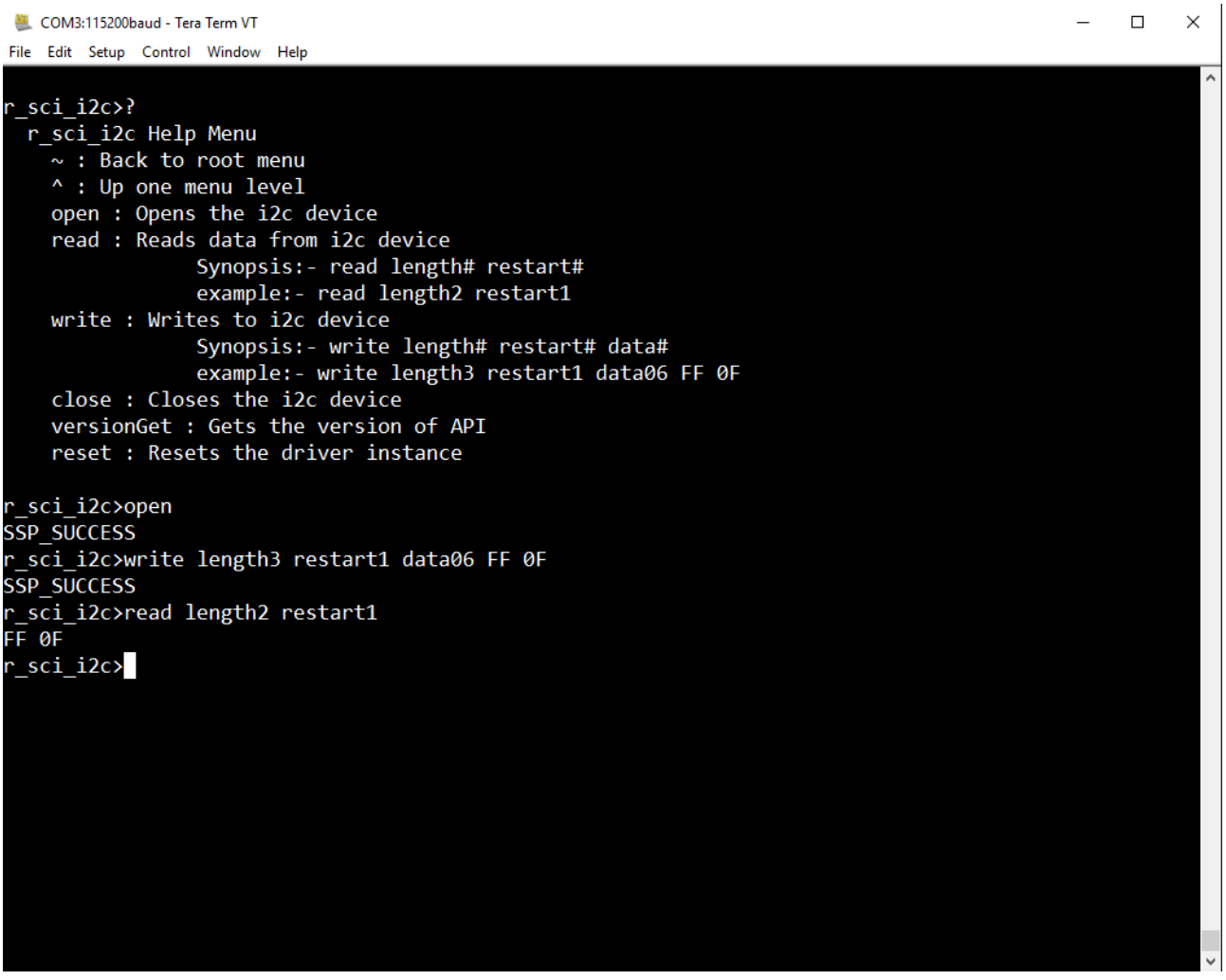
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the SCI I2C HAL driver application, follow these steps:

STEP 1: Type r`_sci_i2c` in the terminal and press Enter to access the SCI I2C HAL submenu. For help, type ? and press Enter.



STEP 2: Invoke each menu item with the corresponding arguments to exercise the APIs. The example shows the SCI I2C write operation to the command byte register6 with values HEX 0x0F and 0xFF. You will see the LED lighting up on the board as result of command output. The respective read shows the value of the registers.



16. Developer Example: Communications Framework

16.1 Introduction

The Developer Example application shows the transport-agnostic Communication Framework of the SSP.

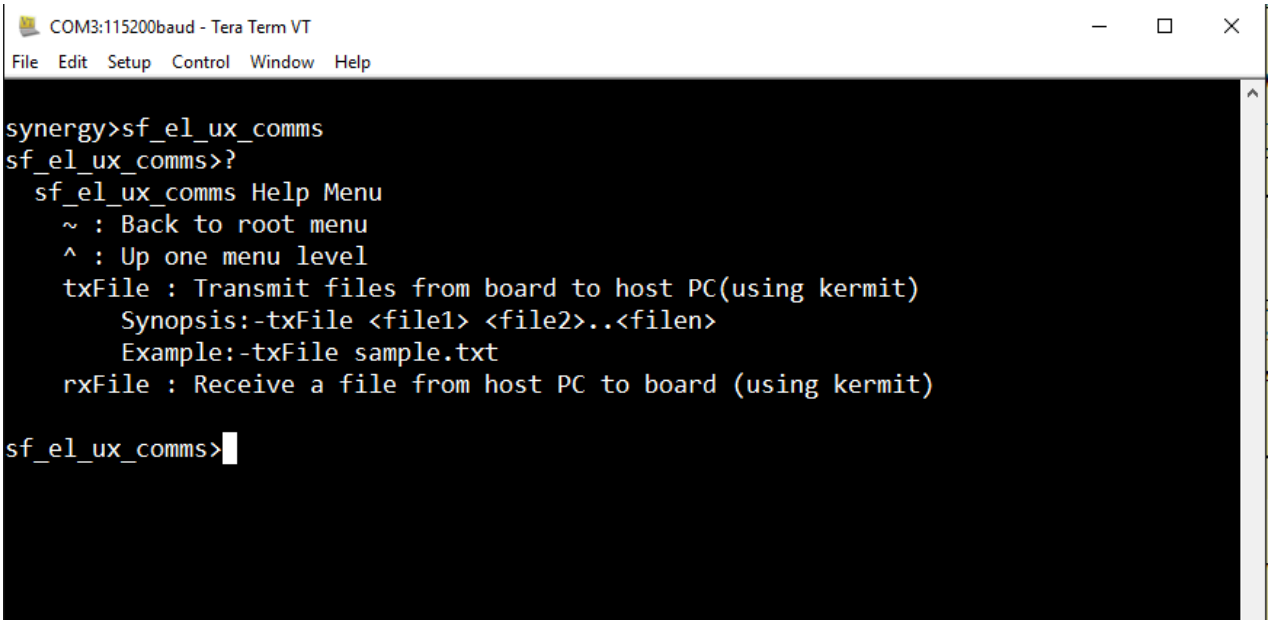
The Console Framework used to interact with Tera Termis built on top of the Communication Framework. In this example, we are using the USB Communications Framework built on top of USBx. It is the same instance that is used by console framework to interact with Tera Term. This example code provides additional file transfer capabilities between the Host PC and DK-S7G2 board using the **Kermit** protocol.

16.2 Communications Framework application

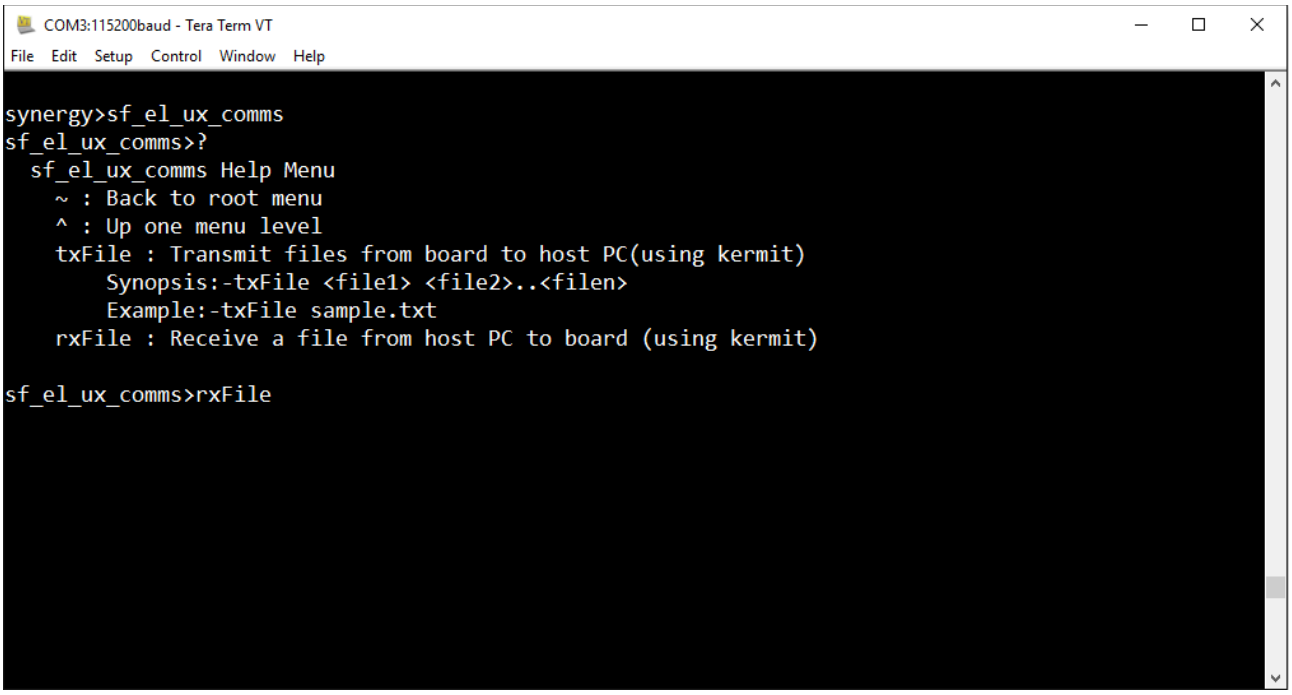
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the Communication Framework application, follow these steps:

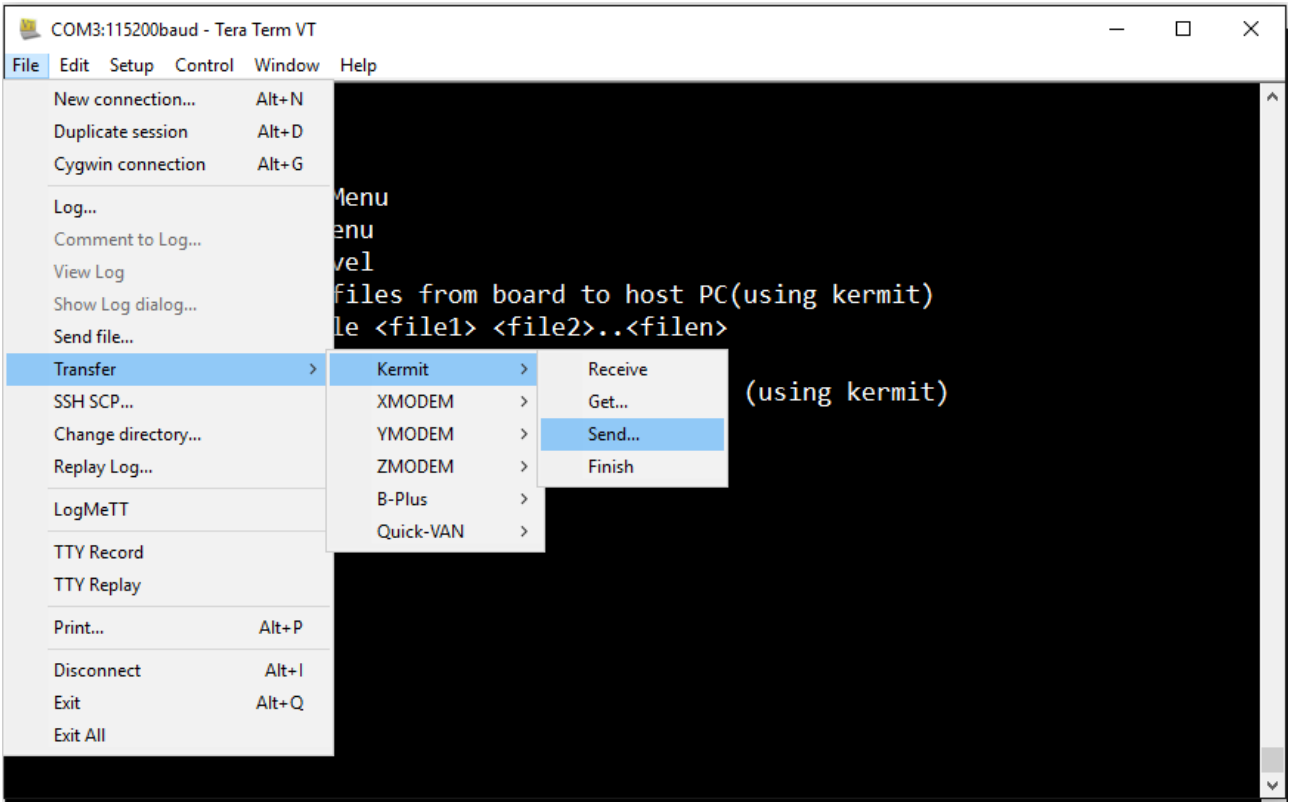
STEP 1: Type `sf_el_ux_comms` in the terminal and press Enter to access the `sf_el_ux_comms` sub menu. For help, type ? and press Enter.



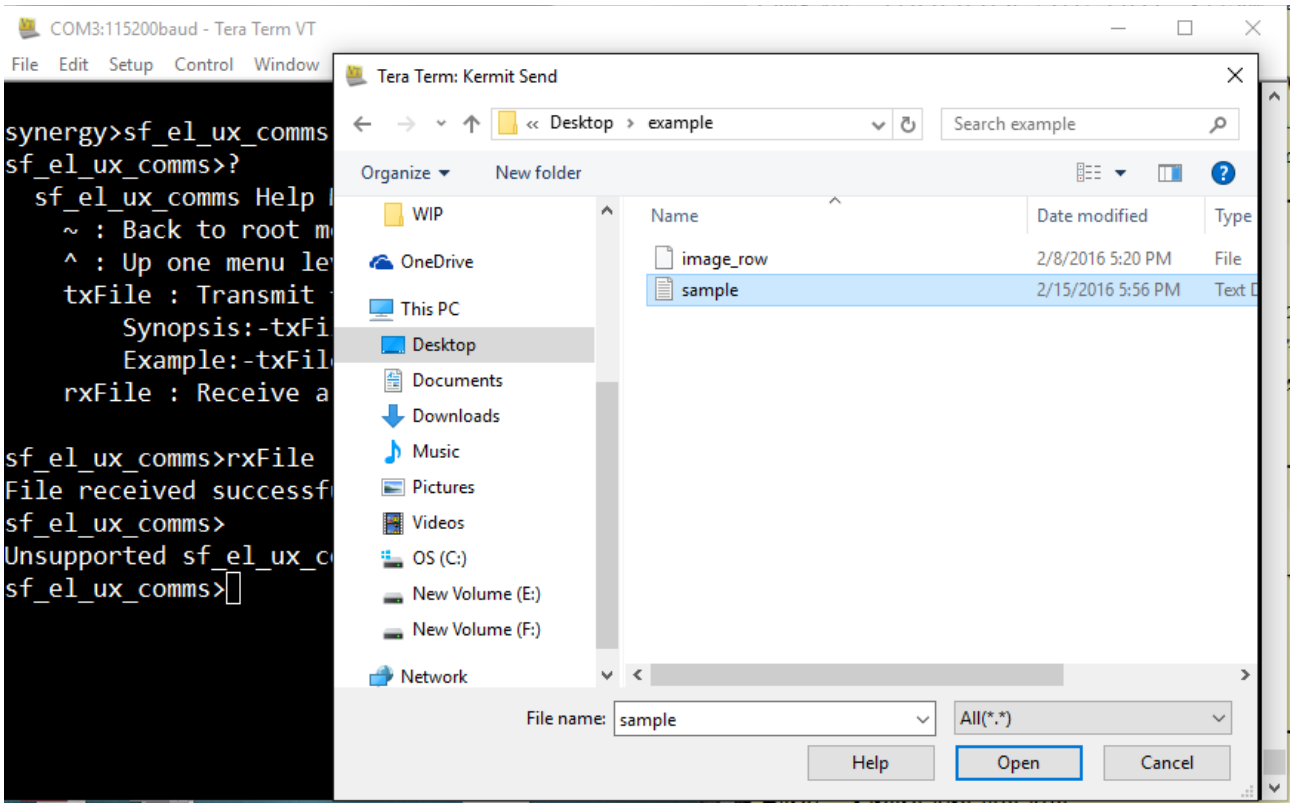
STEP 2: Use the `rxFile` command to receive file from the DK-S7G2. Type `rxFile` and press Enter. The board will then wait for the file transfer.



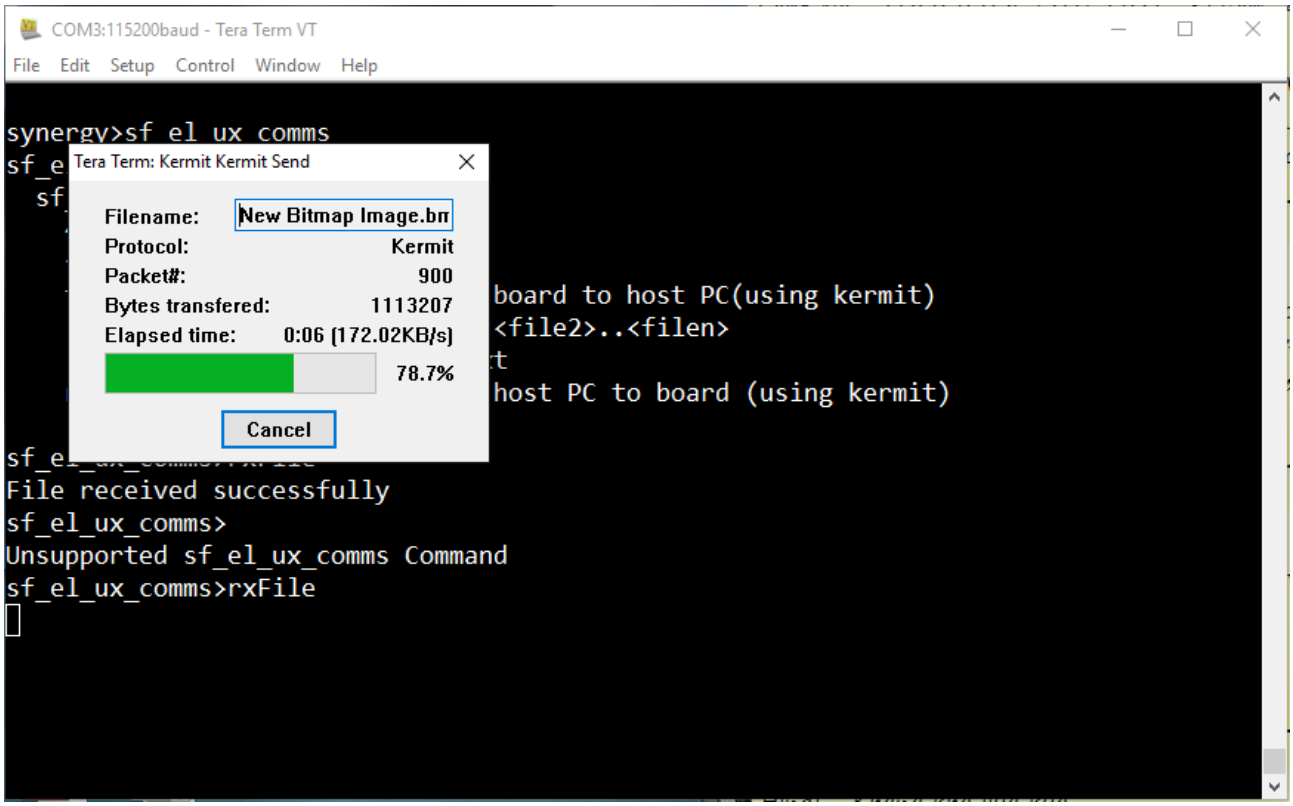
STEP 3: InTera Term, navigate to **File**>**Transfer**>**Kermit**>**Send**. The file explorer opens.



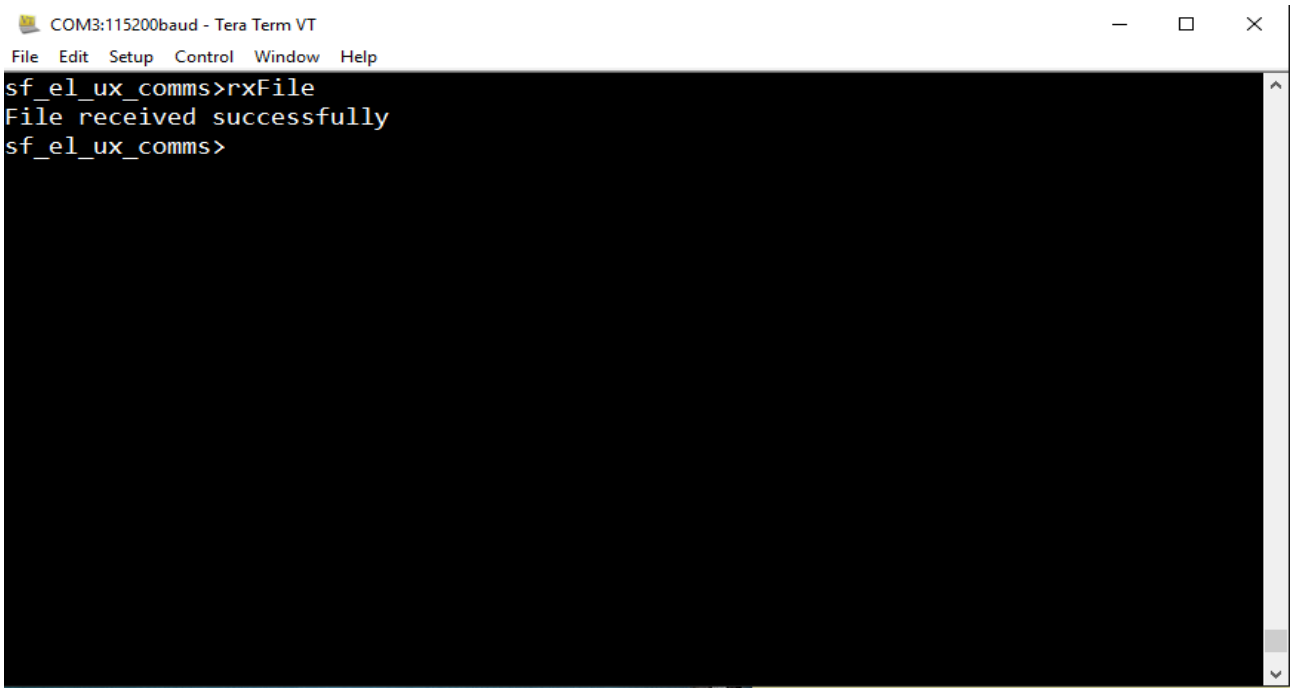
STEP 4: In the file explorer, choose the file to transmit from the host PC to the board.



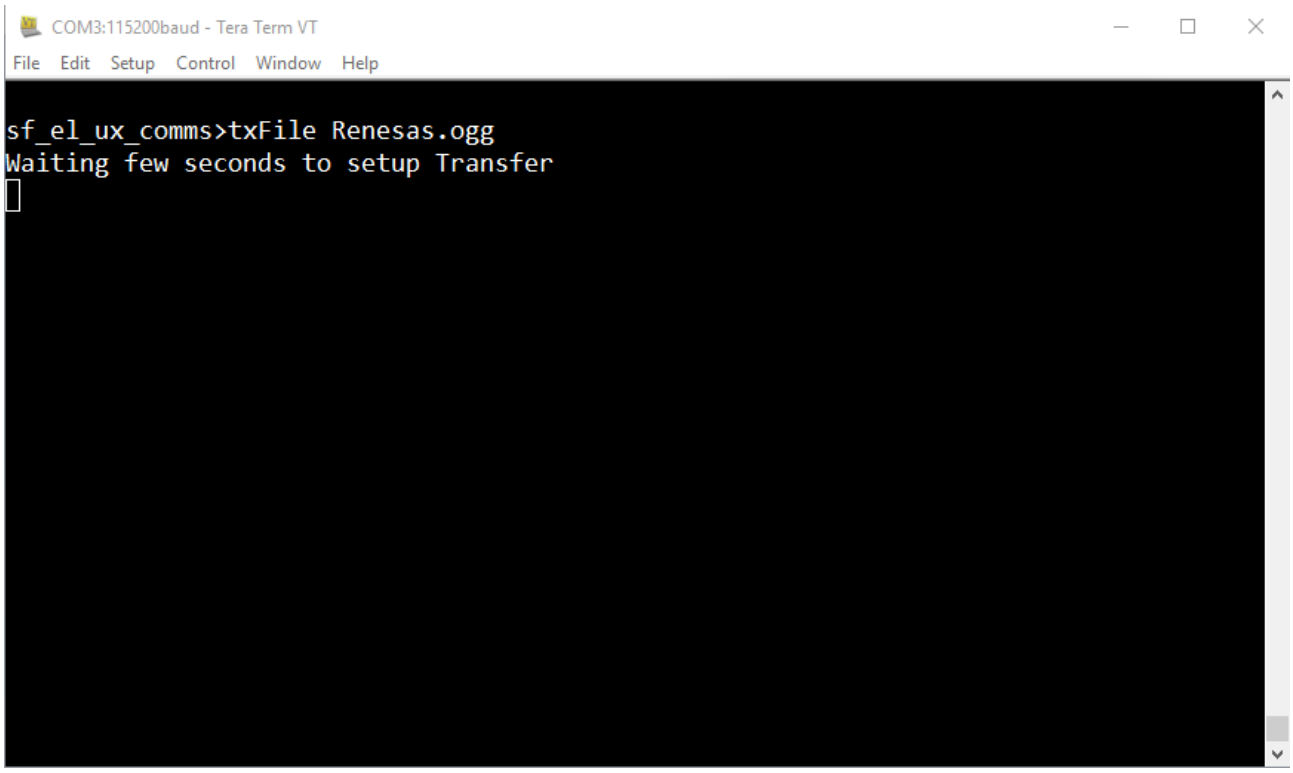
STEP 5: Once the transfer has started, you can see the transfer progress window in Tera Term.



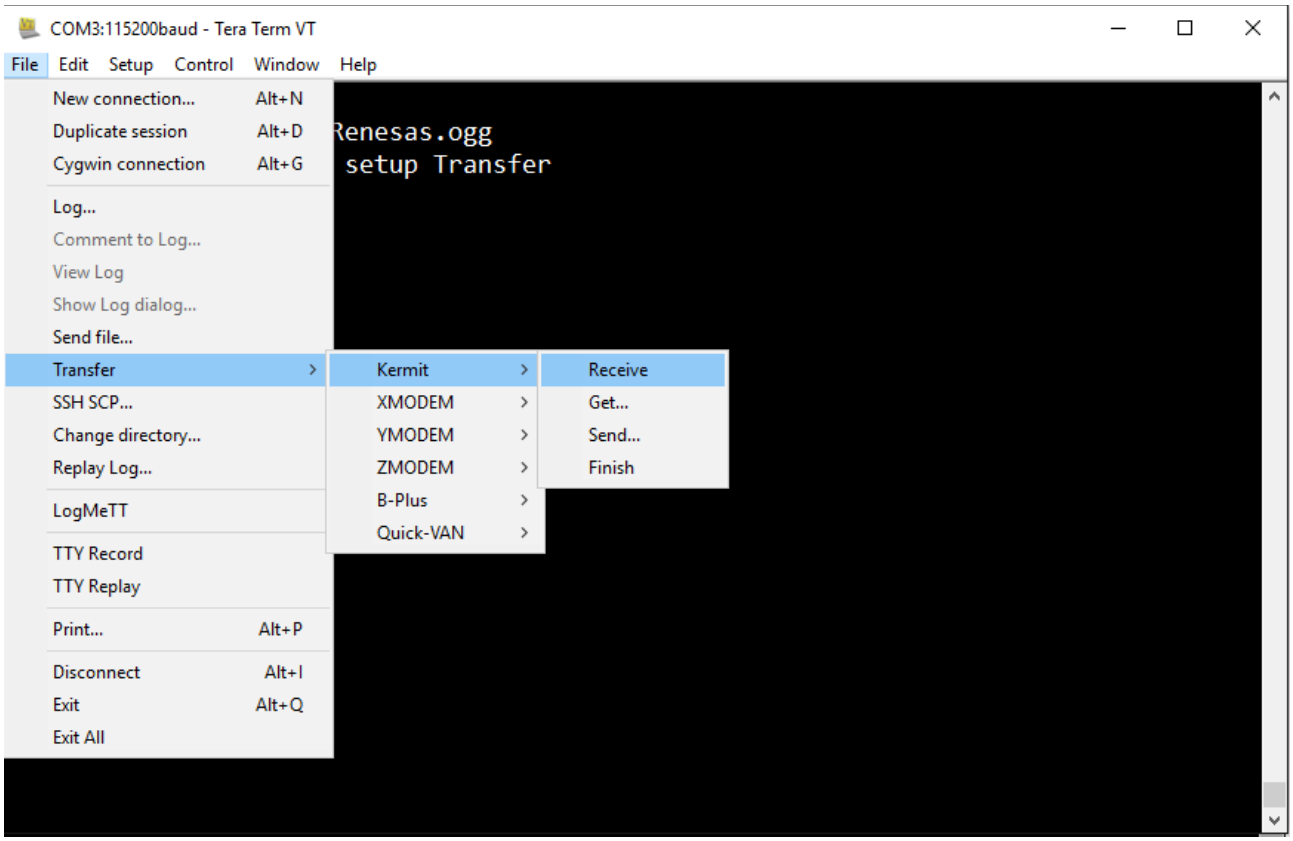
STEP 6: Once the transfer is complete, the `rxFile` command will return will with a success or failure notice.



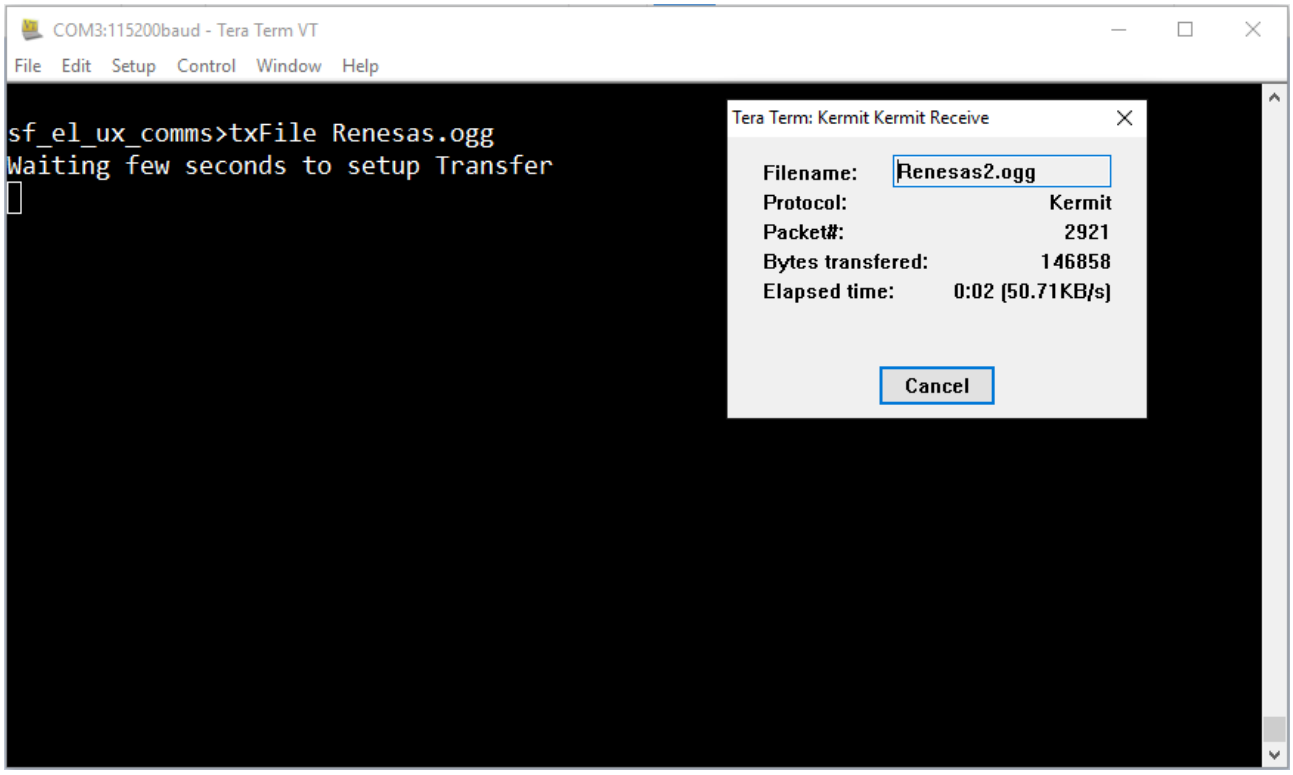
STEP 7: Use the `txFile` command to send the file from DK-S7G2 board to the PC. The command use is `txFile<filename1><filename2>`.



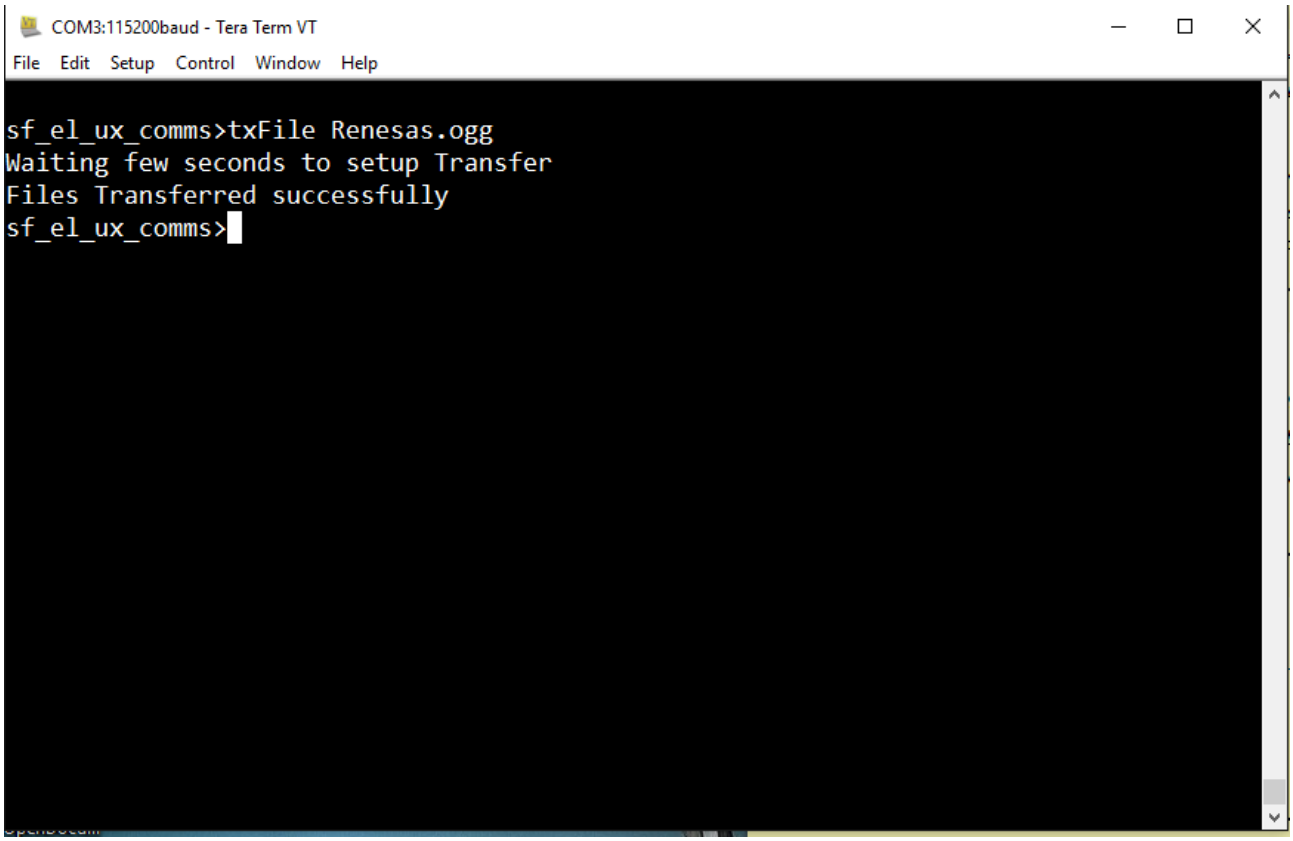
STEP 8: In Tera Term, choose **File**>**Transfer**>**Kermit**>**Receive**.



STEP 9: The application waits for about 10 seconds before it initiates the transfer. Once the transfer starts, the transfer speed is displayed in the progress window.



STEP 10: Once the transfer is closed, the transfer progress window will be closed and the console will print the status.



NOTES:

- The application uses the E-Kermit library which is licensed under the Revised 3-Clause BSD license.
- The application has only been tested using Tera Term.
- The application might not work if the file transferred is binary and contains a kermit-defined control character sequence.

17. Developer Example: FileX Framework

17.1 Introduction

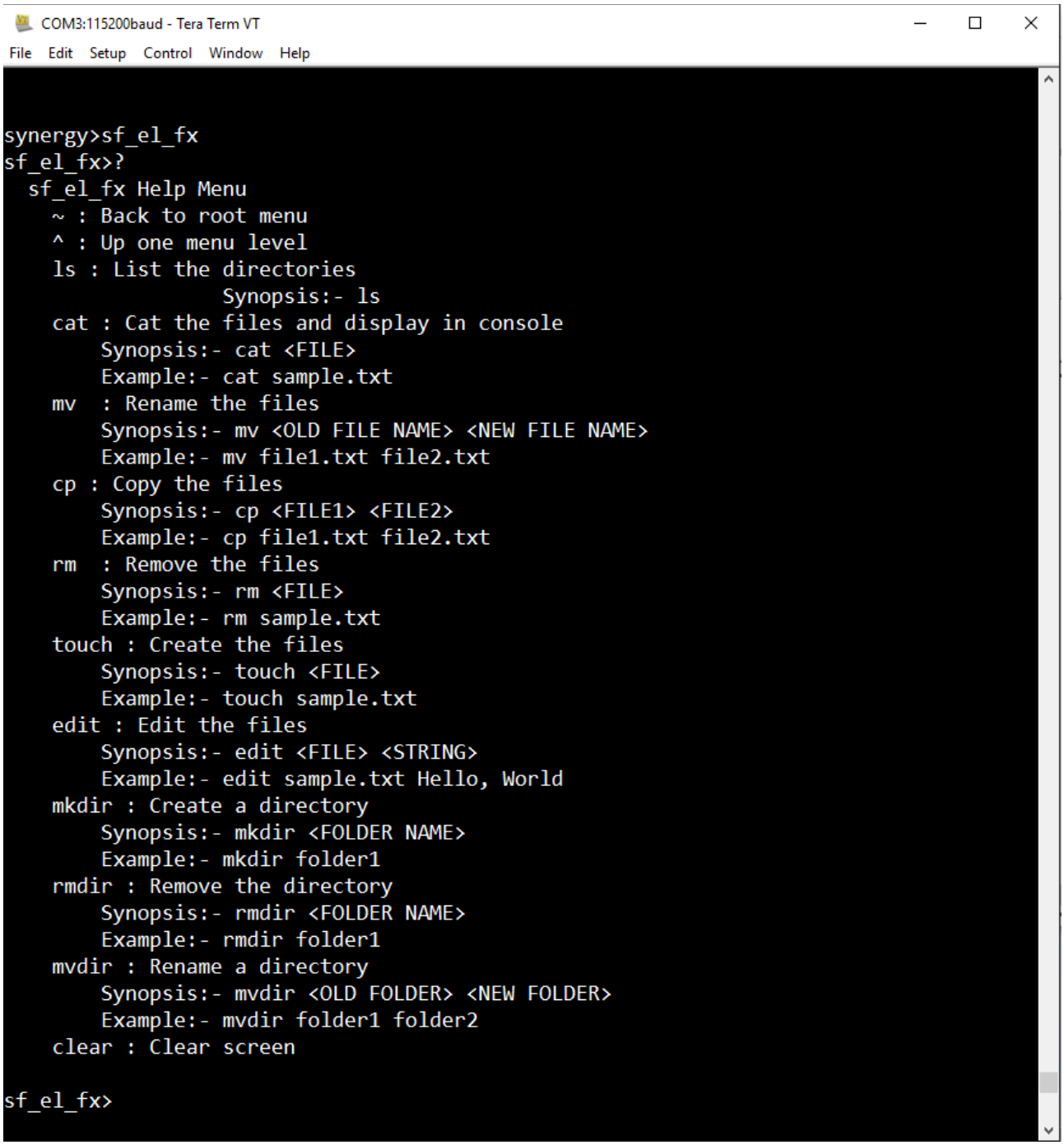
The FileX Framework is implemented with support from the Block Media Framework. The only block media device currently supported by the SSP is the SD/MMC block device. The Developer Example uses SD/MMC as the block media device and implements the commonly used file system APIs, which are similar to Linux commands, to demonstrate the FileX Framework.

17.2 Run the FileX Framework application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

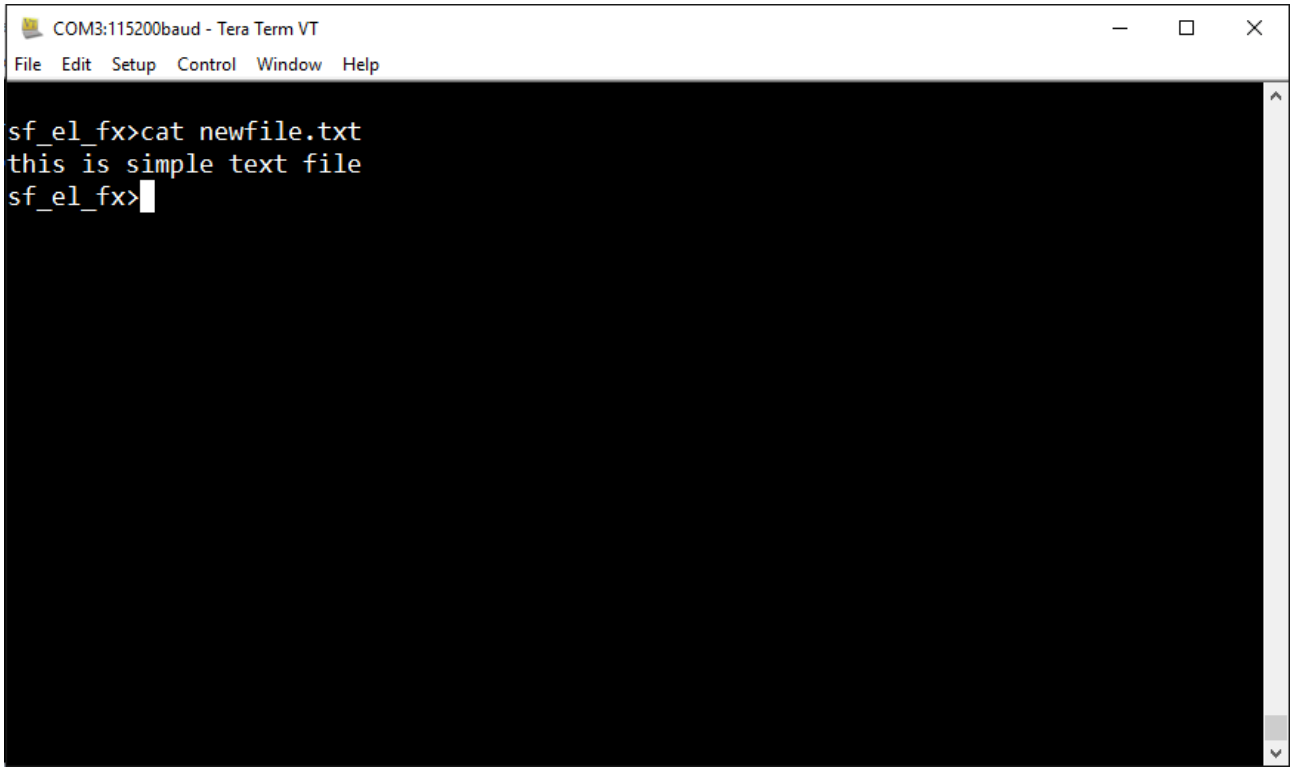
To run the FileX Framework application, follow these steps:

STEP 1: Type `sf_el_fx` in terminal and press Enter to access the sf_el_fx sub menu. For help, type ? and press Enter.

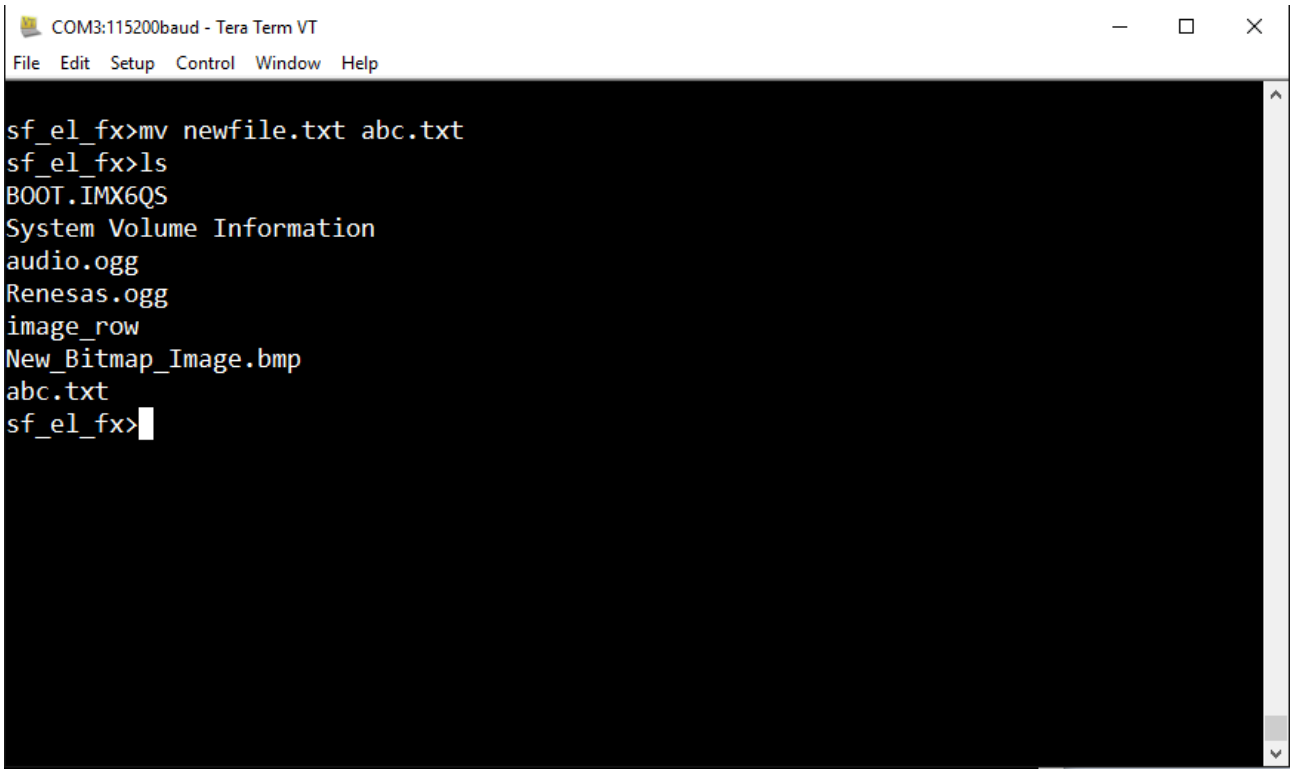


STEP 2: Type the `ls` command to list the files and directories on the SD card.

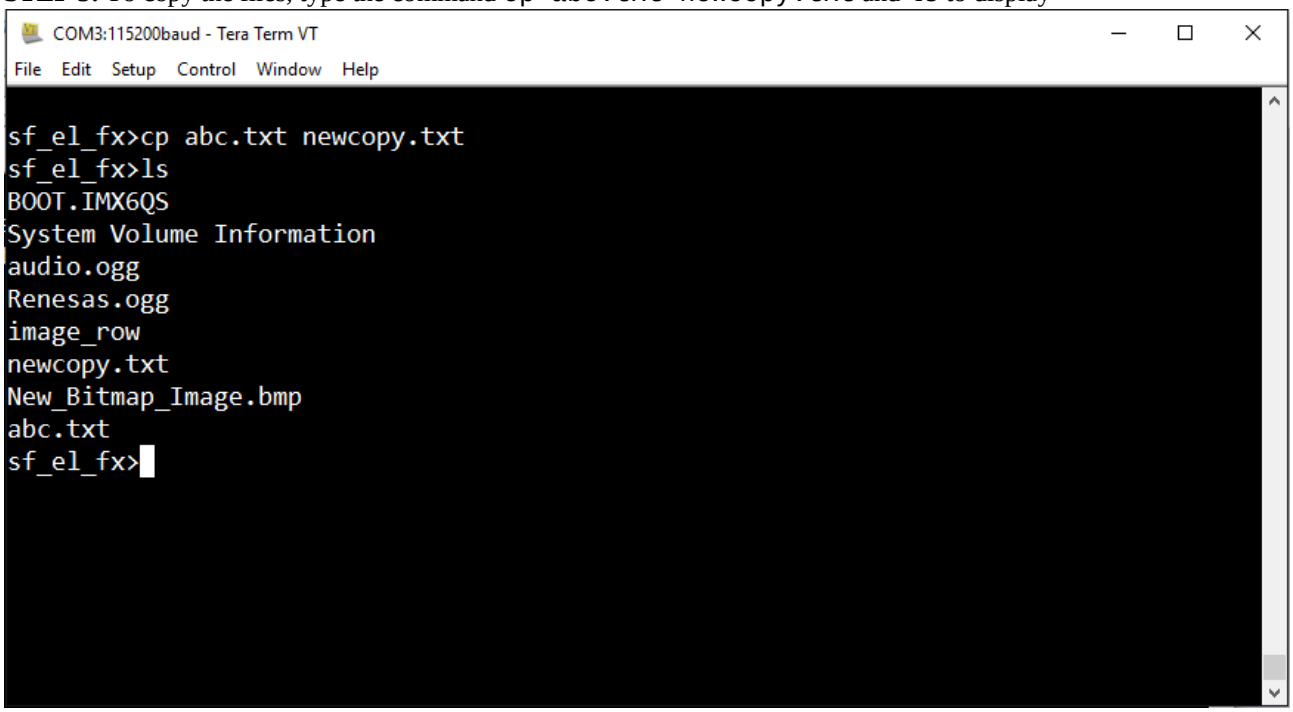
STEP 3: Type the command `cat newfile.txt` to display the content of file in the console.



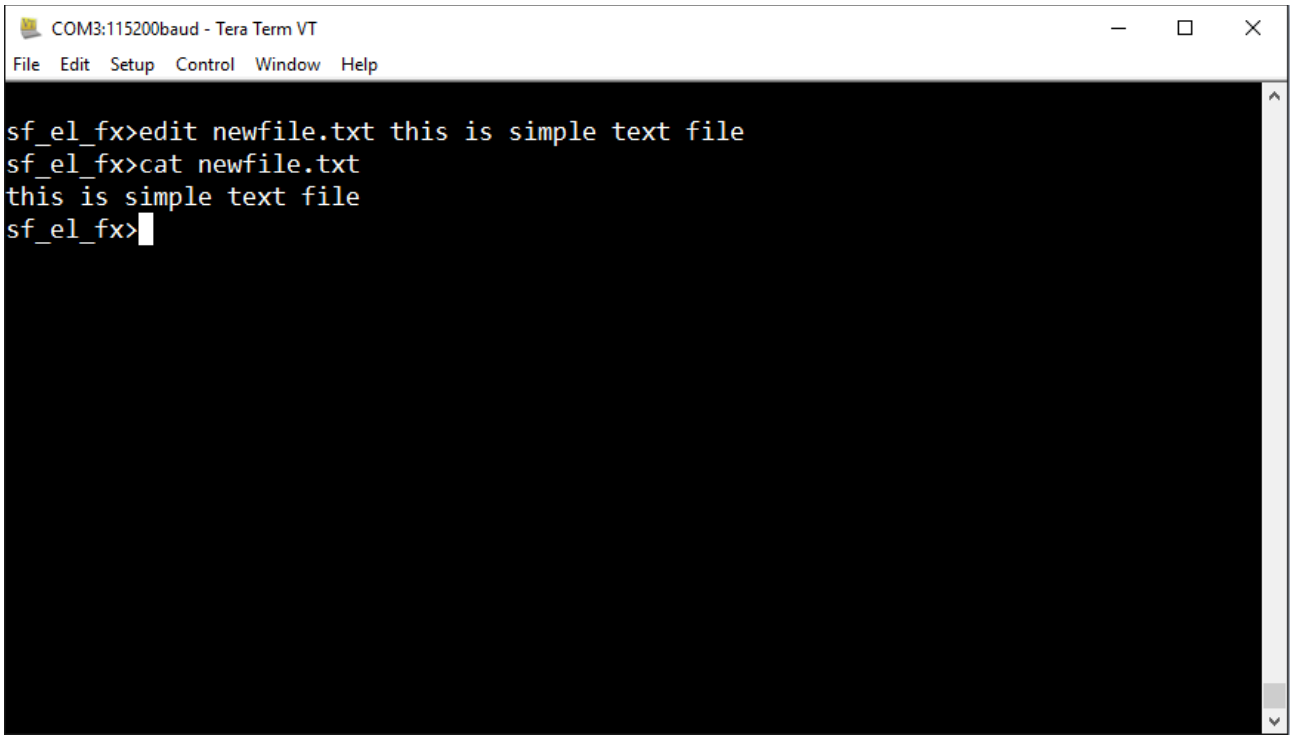
STEP 4: To rename the file, type the command `mv newfile.txt abc.txt` and `ls` to display.



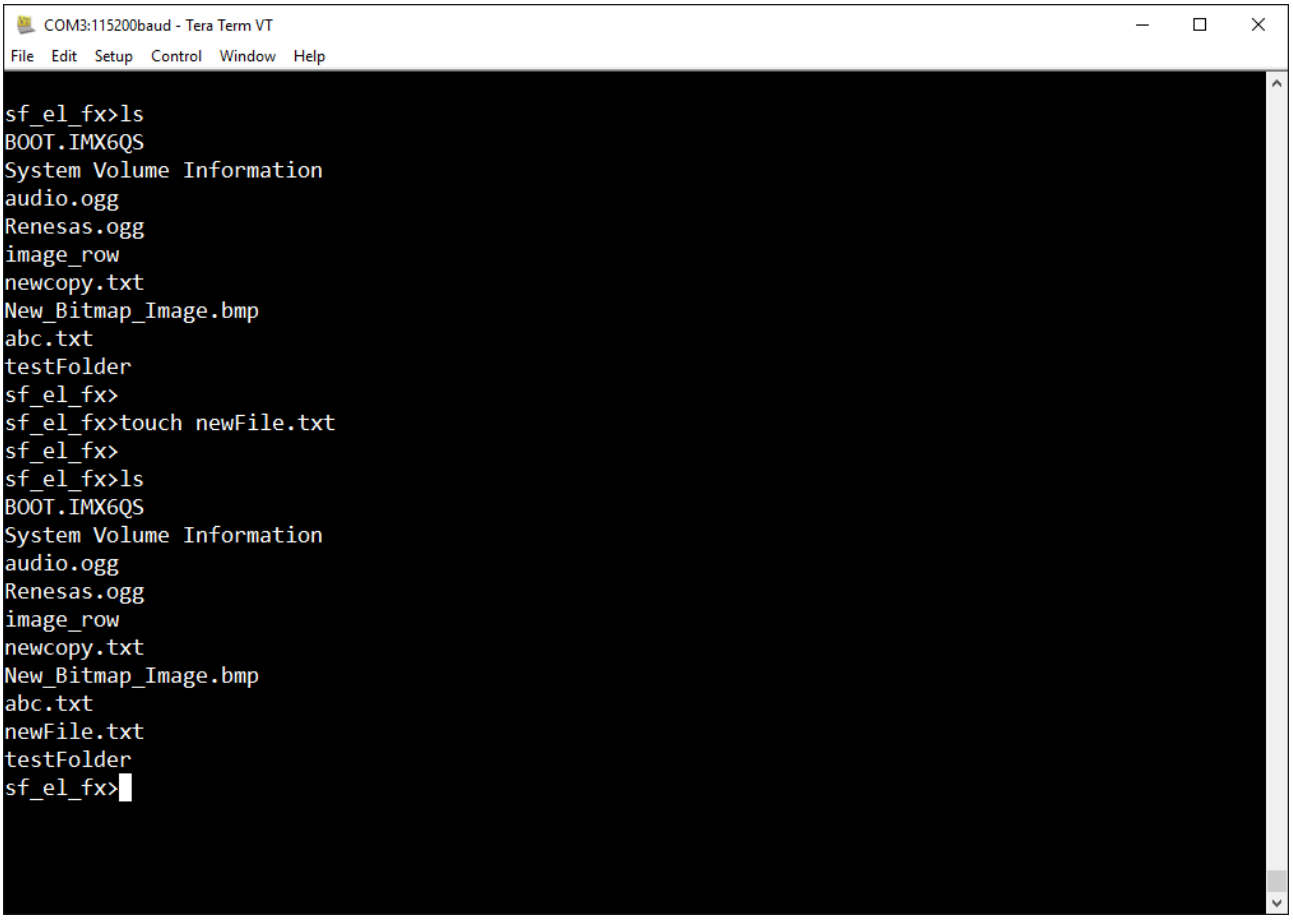
STEP 5: To copy the files, type the command `cp abc.txt newcopy.txt` and `ls` to display



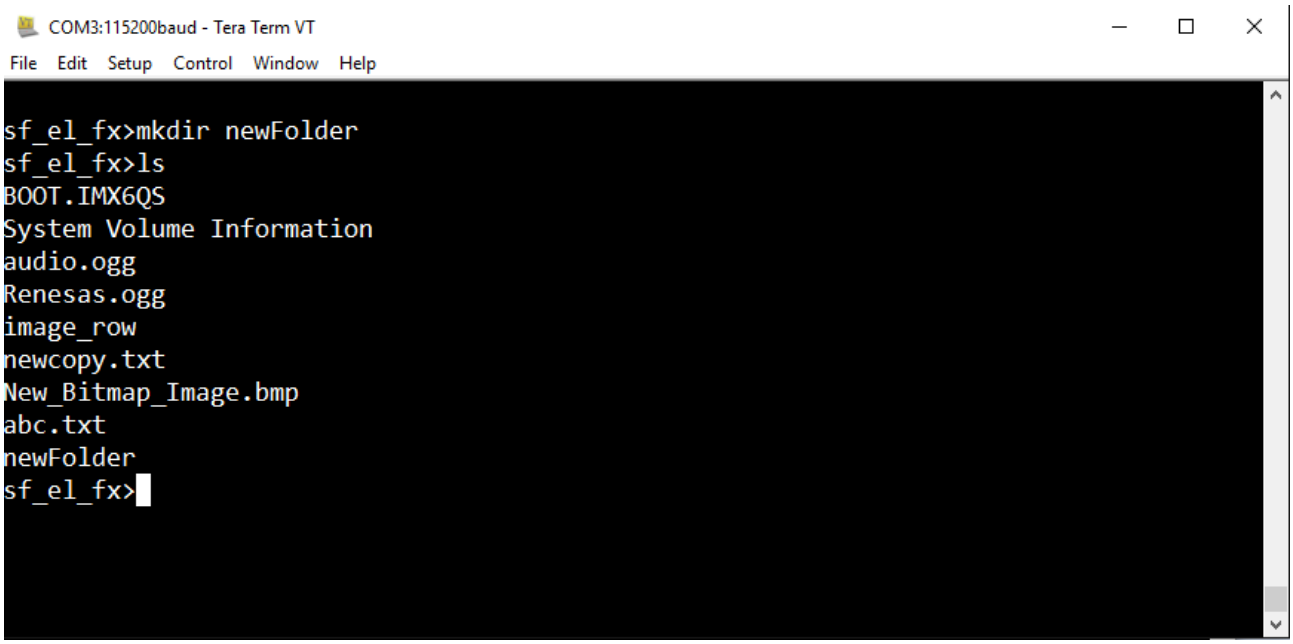
STEP 6: To edit a file, type the command `edit newfile.txt String` and use `catnewfile.txt` to display the edited content.



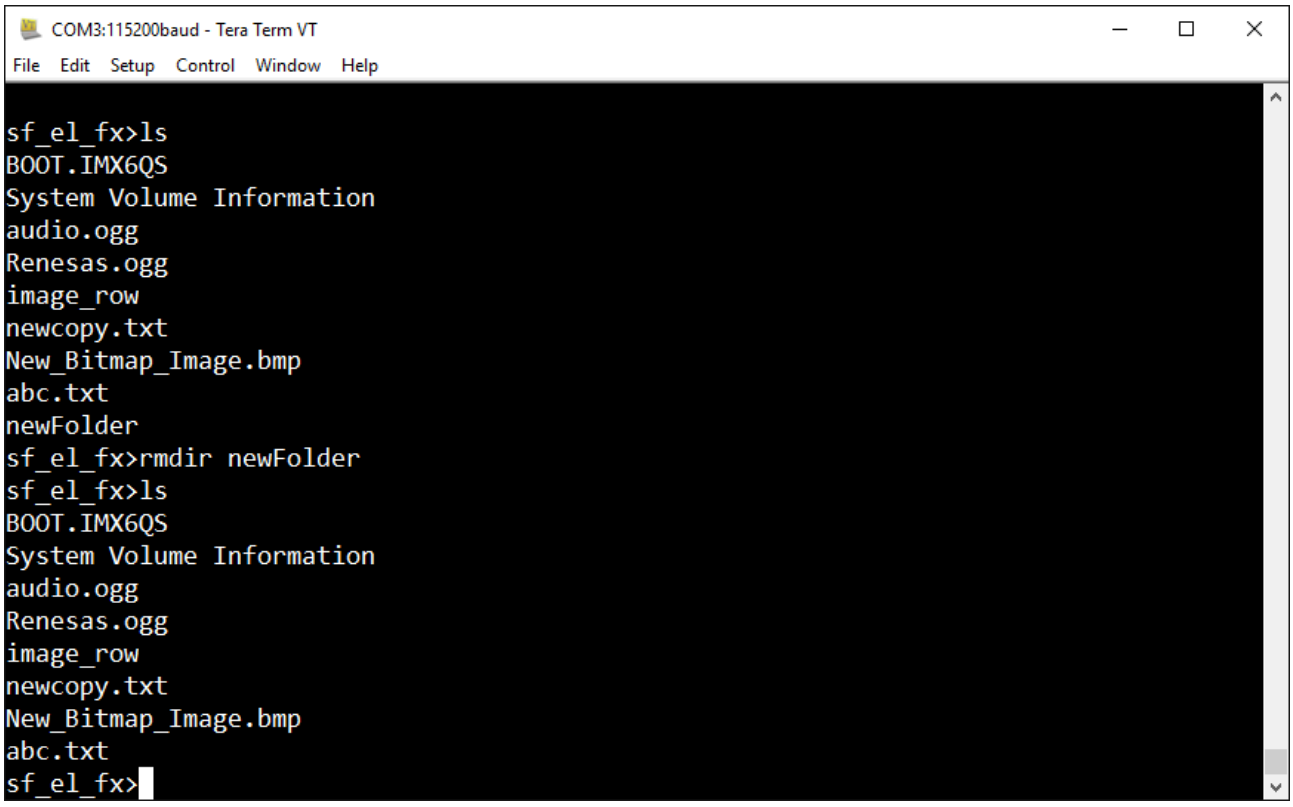
STEP 7: To create a file, type the command `touch newFile.txt`. The command creates a new file, which you can display using the `ls` command.



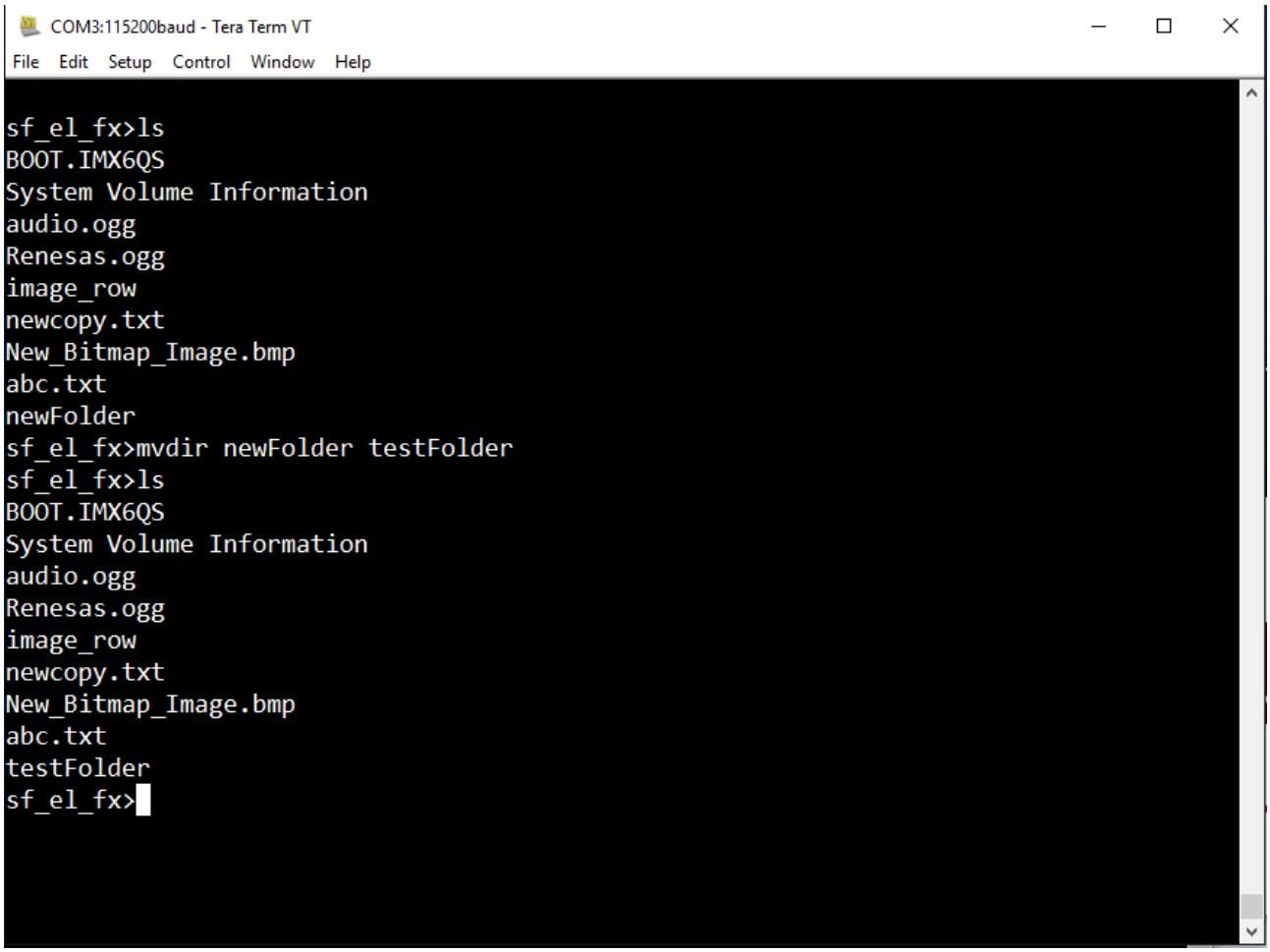
STEP 8: To create directory, type command `mkdir newFolder`.



STEP 9: To remove a directory, type command `rmdir newFolder`.



STEP 10: To rename a directory, type command `mvdir newFoldertestFolder`.



18. Developer Example: HAL ICU Driver

18.1 Introduction

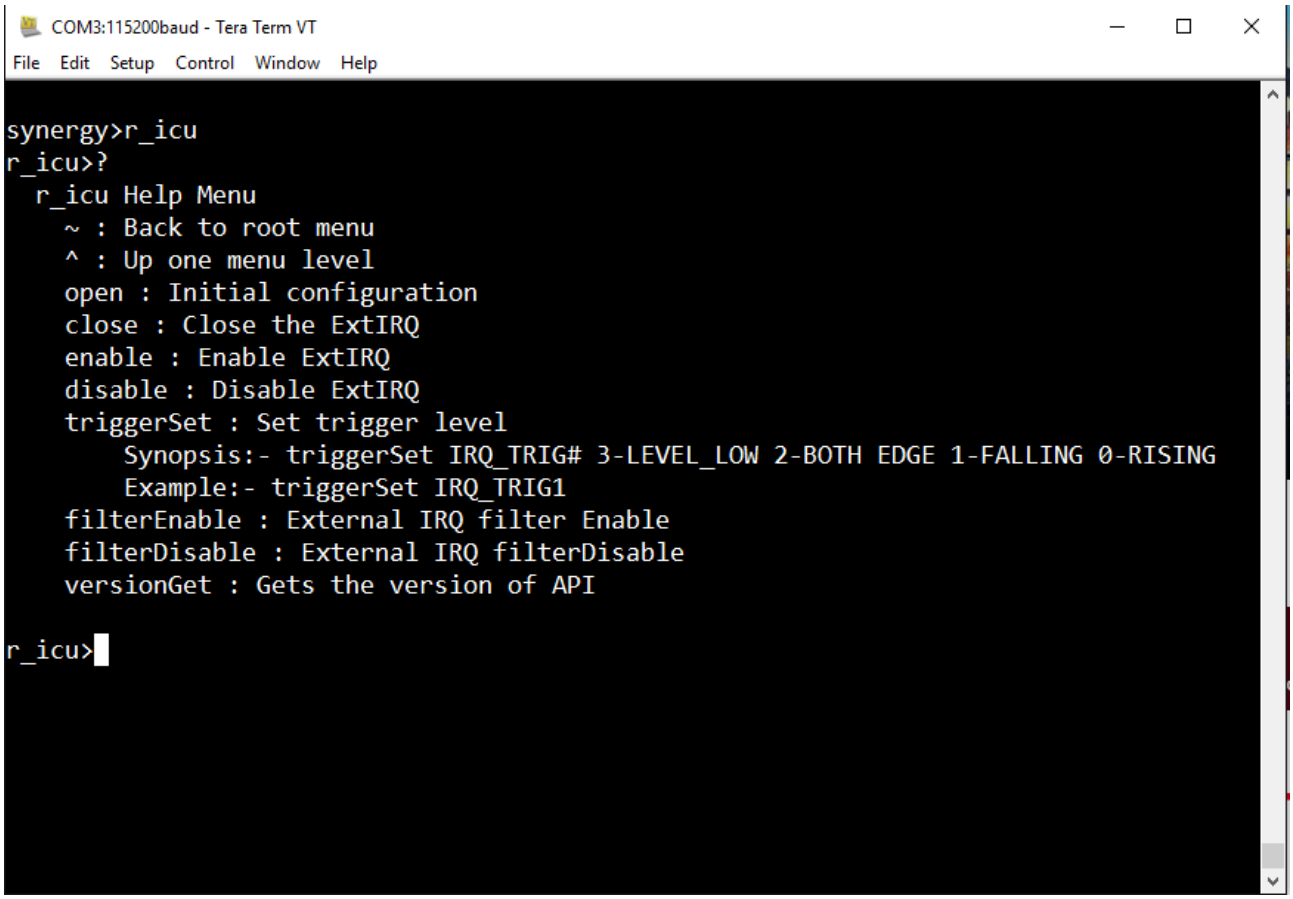
The Developer Example demonstrates the external IRQ using the ICU HAL modules. The Developer Example uses Port0 Pin6 configured as GPIO in Input mode to generate the External IRQ when you press button S1 on the DK-S7G2 Board. The application toggles LED1 each time you press the button.

18.2 Run the HAL ICU Driver application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the HAL ICU driver application, follow these steps:

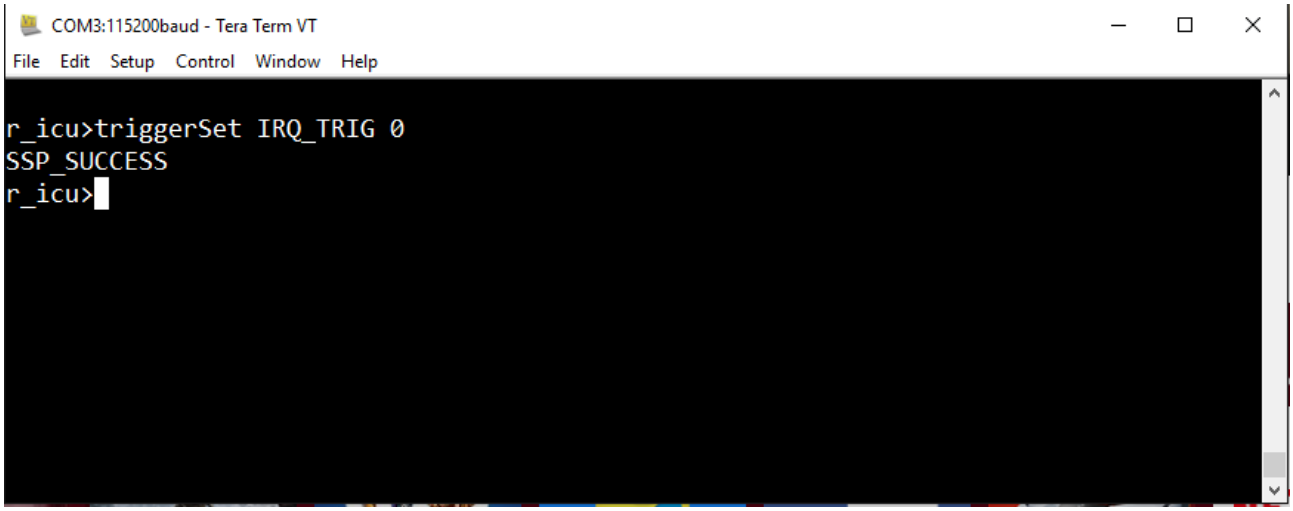
STEP 1: Type r_icu to get the ICU external IRQ menu. For help, type ? and press Enter.



STEP 2: Type the `open` command to initialize the External IRQ.

STEP 3: Type the `enable` command to enable the External IRQ.

STEP 4: Type the `triggerSet IRQ_TRIG#` command to enable the trigger for External IRQ.



STEP 5: Press button S1 on the DK-S7G2 Board to display the External IRQ trigger. LED1 toggles with each button press.

19. Developer Example: Thread Monitor Framework

19.1 Introduction

The Thread Monitor(TM) uses the Watchdog Timer as a low-level driver to reset the device if any of the threads registered in the Thread Monitor encounter an erroneous condition. The Developer Example explains how to register a thread for monitoring and how to figure out its minimum and maximum number of executions in a given window by enabling profiling mode.

The Developer Example uses a thread and registers this thread for monitoring with two possible arguments called misbehave0 and misbehave1. The thread toggles green LED of LED1 for an infinite time if misbehave0 is passed as an argument to the `demo_thread_monitor` command. If misbehave1 is passed as an argument to `demo_thread_monitor`,thethread toggles the red led of LED1 10 times and enters an erroneous condition in which the thread locks in a while(1) loop and registered thread will not indicate thread monitor that it is active, which results in device reset.

NOTES:

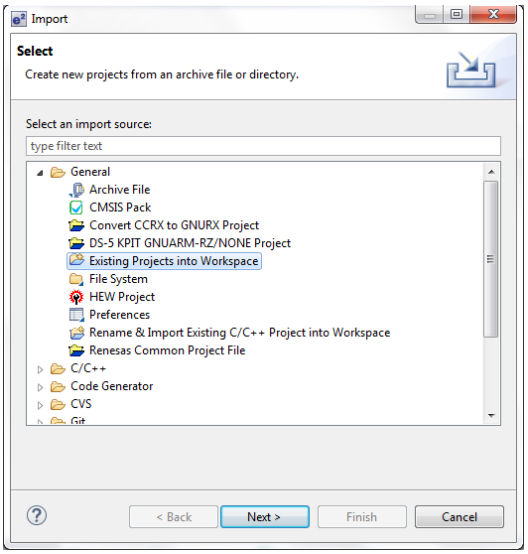
- The `close`API of the Thread Monitor does not stop the WDT. You must refresh the WDT explicitly otherwise the device is reset. Refer to Thread Monitor SSP limitation.
- Disconnect the JTAG while running the application in normal mode (that is the profiling mode is disabled). Refer WDT SSP limitation.

19.2 Build and Run the Thread Monitor Framework application

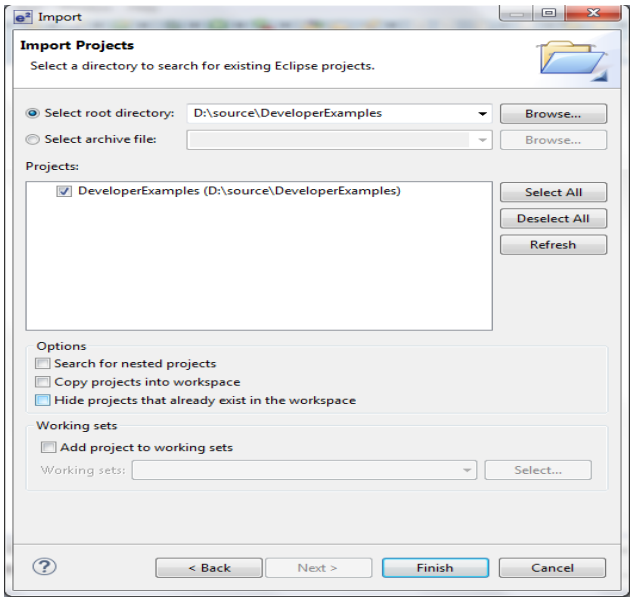
The steps required to build the application differ from the other Developer Examples because for the Thread Monitor you must configure the thread for profiling mode in the e^2 studio Project Configurator using the Threads tab.

STEP 1: Extract the downloaded project source into a work directory in the host PC.

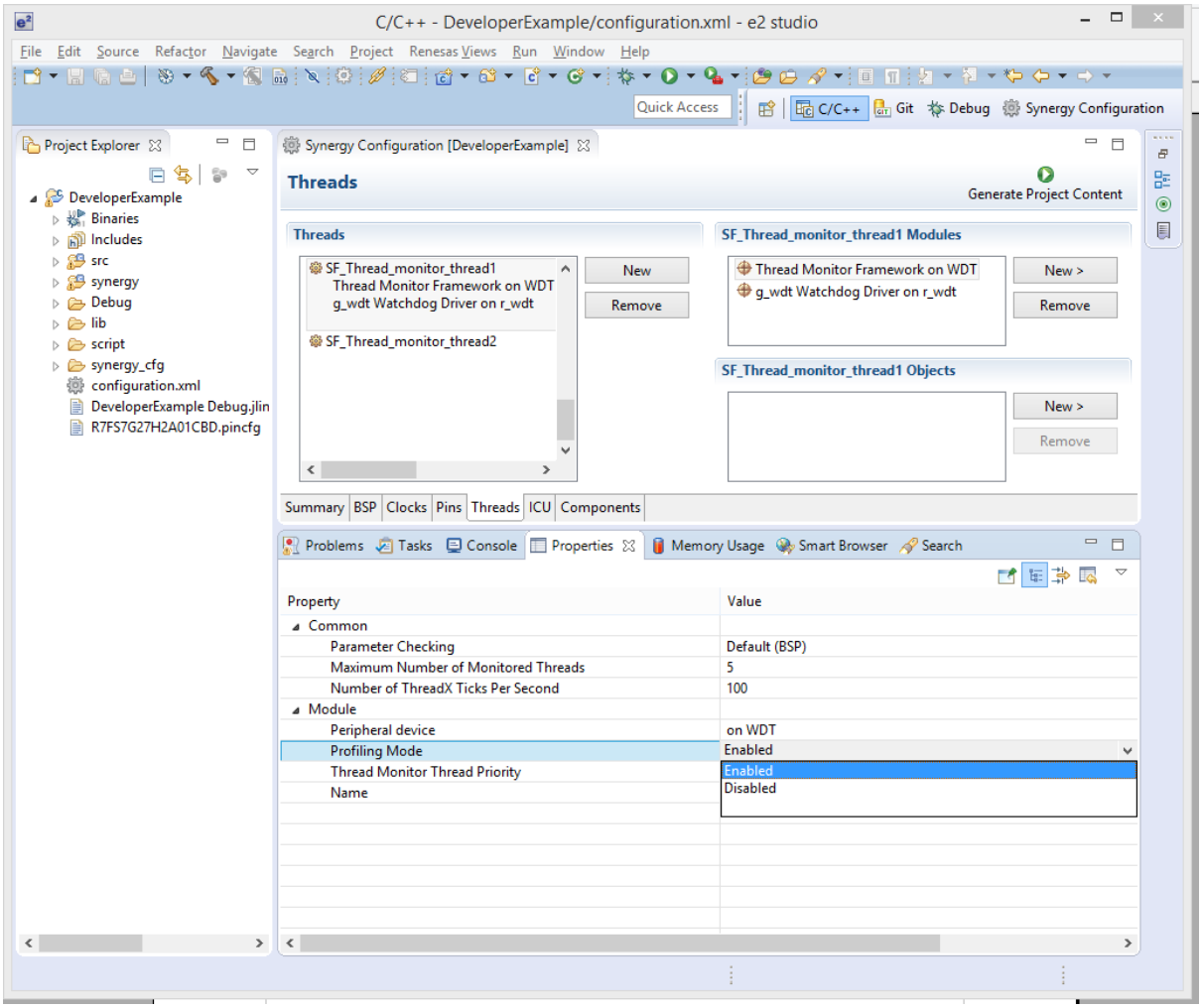
STEP 2: Launch e^2 studio. Navigate to **File**>**Import**. The Import dialog box opens.



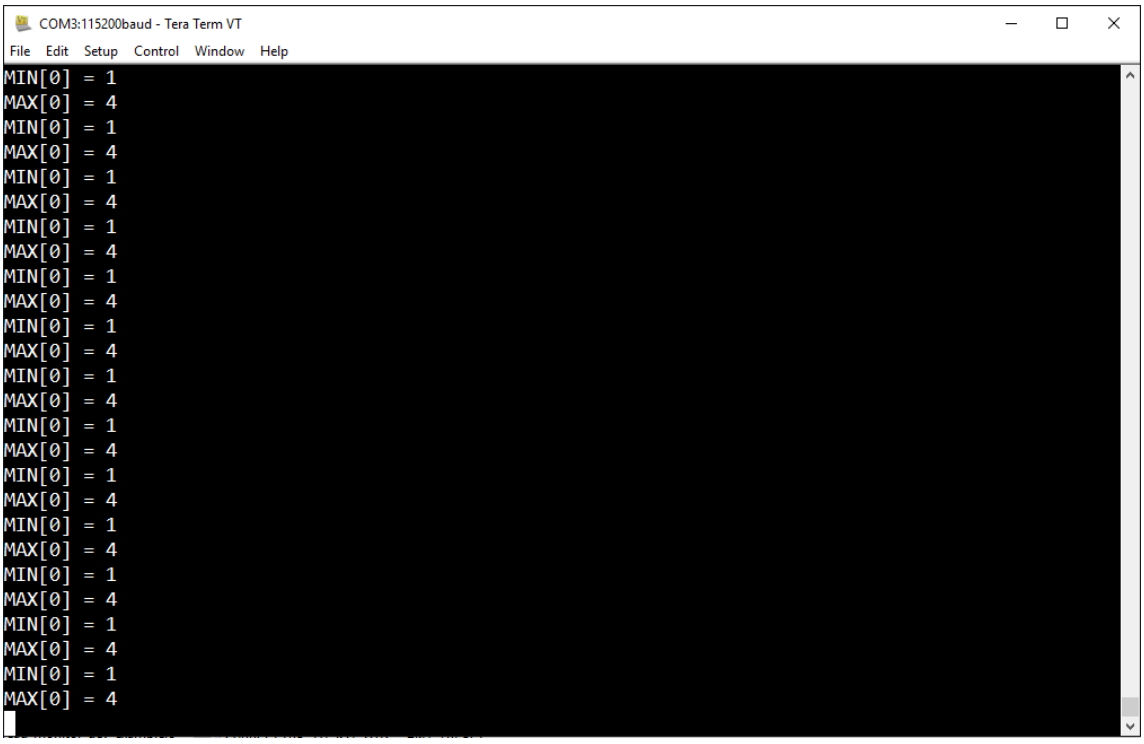
STEP 3: Select **Existing Project into Workspace** and click on **Next**. Browse to the root directory of the extracted project source and click **OK**. If e^2 studio recognizes the project, it will be shown in the **Projects** window. Make sure that checkbox next to it is checked and click the **Finish** button to import the project. (If a local working copy is required, check **Copy projects into workspace**.)



STEP 4: Open the configuration.xml file, select Thread Monitor module, and go to Properties to enable the profiling mode to capture thread's minimum and maximum count.



STEP 5: When you run application, the min and max value are printed on the console.



STEP 6: After getting the minimum and maximum values, set the values as count while registering the thread. If the count is outside of the minimum and maximum values, it is considered as misbehavior. Minimum and maximum depend on the design to design. Profiling mode helps user to get minimum and maximum count value. Once count value is extracted using profiling mode, set the same while registering for monitoring.

Disable profiling mode in the properties tab and rebuild the project.

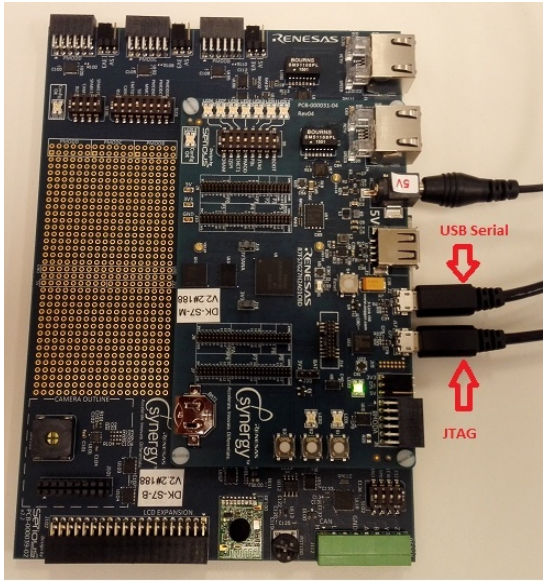
NOTE: The project must be run without the JTAG debugger for the WDT reset to work. The JTAG cable will be disconnected in the following steps for this reason.

STEP 7: Clean and build the project.

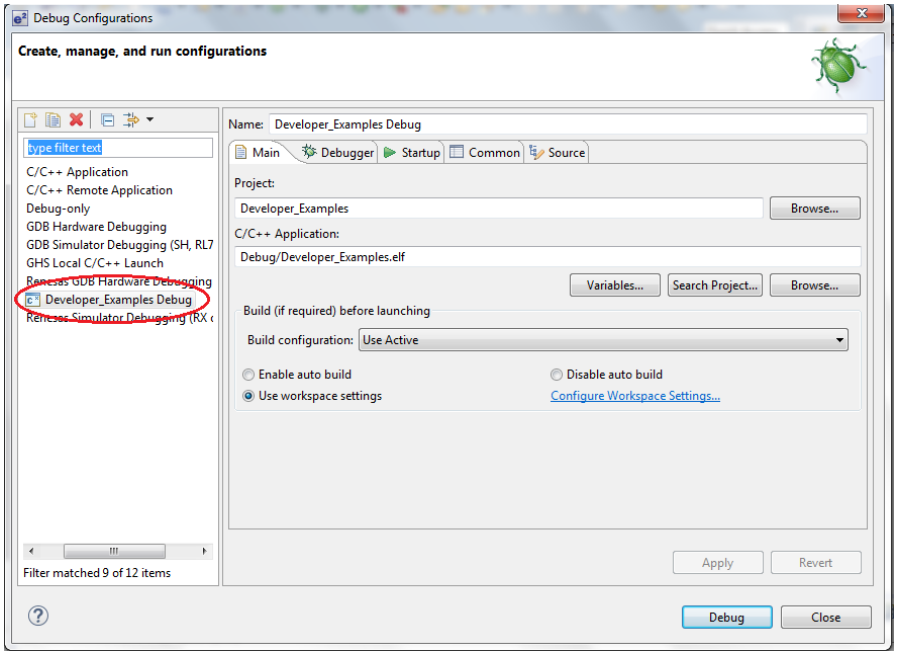
STEP 8: Connect the S7G2 DK board to the host PC. Two connections are needed:

a) The JTAG debug connection to program.

b) The USB-CDC connection for console access



STEP 9: Power on the board. In ISDE, click on **Run**>**Debug configurations**. A new debug configuration with the project name will be created. Click **Debug**.

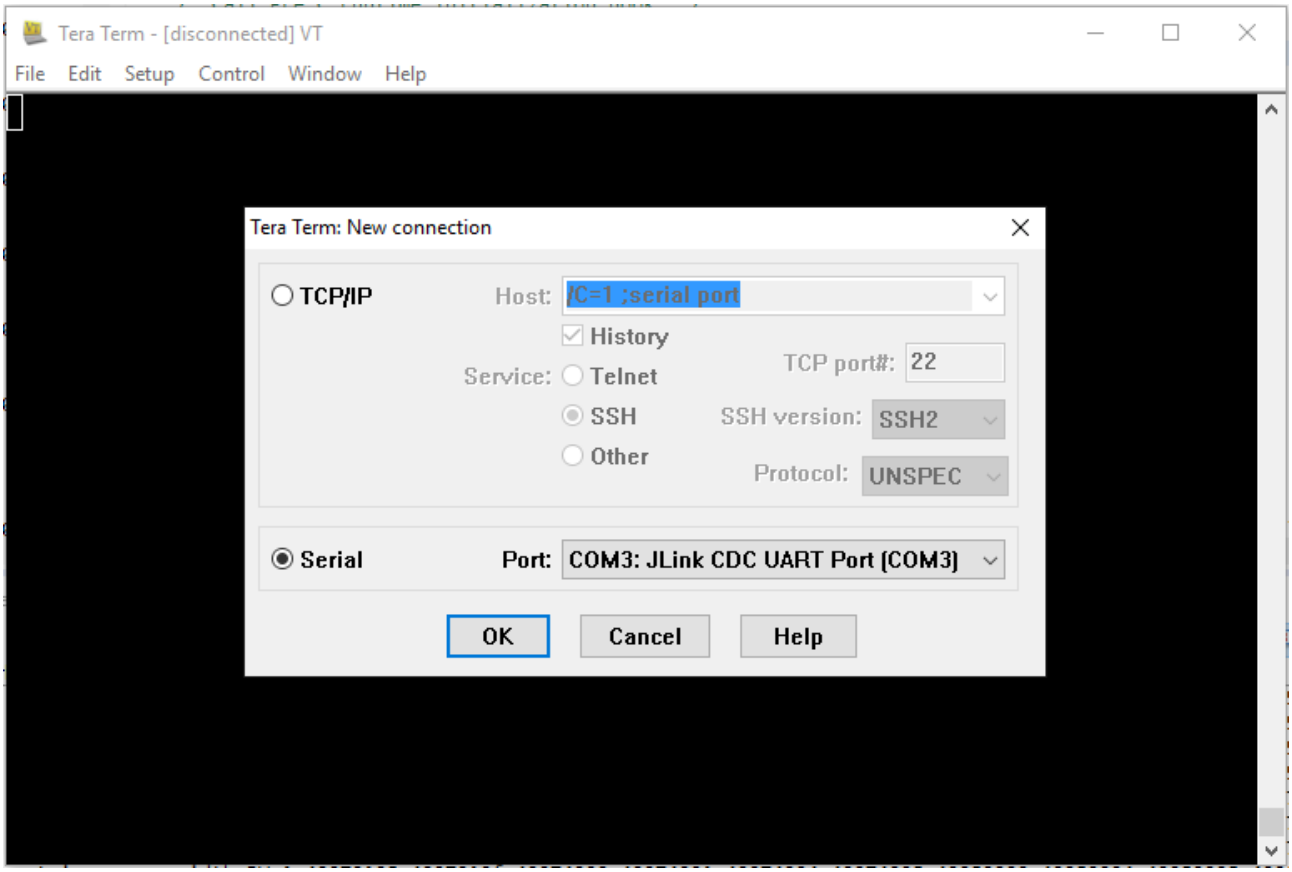


STEP 10: Click **Yes** to switch to the Debug Perspective if you are asked.

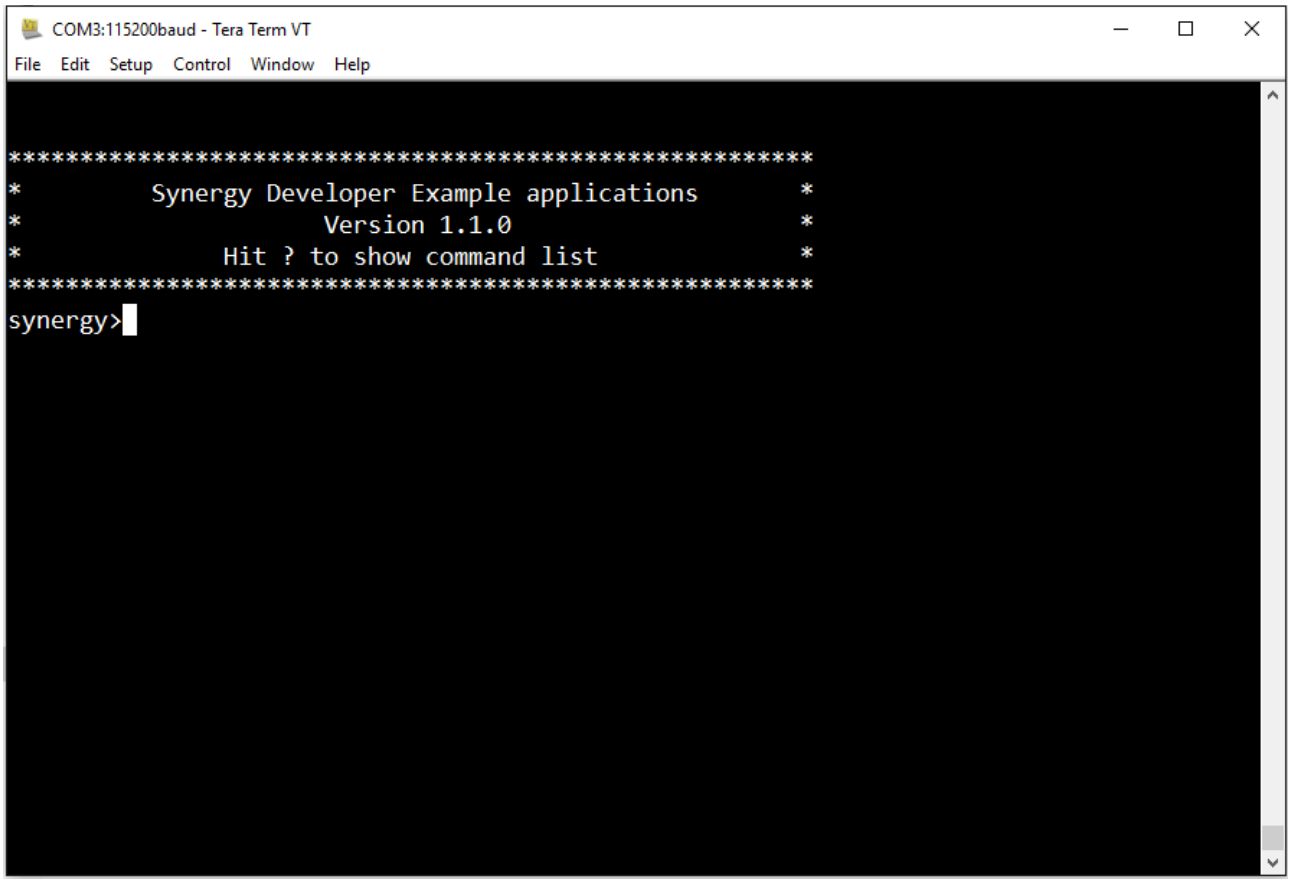
STEP 11: Click the Resume button  twice so that the application starts its scheduler.

STEP 12: Click the Stop button  and disconnect the JTAG. Press the Reset button in the board. Open Tera Term.

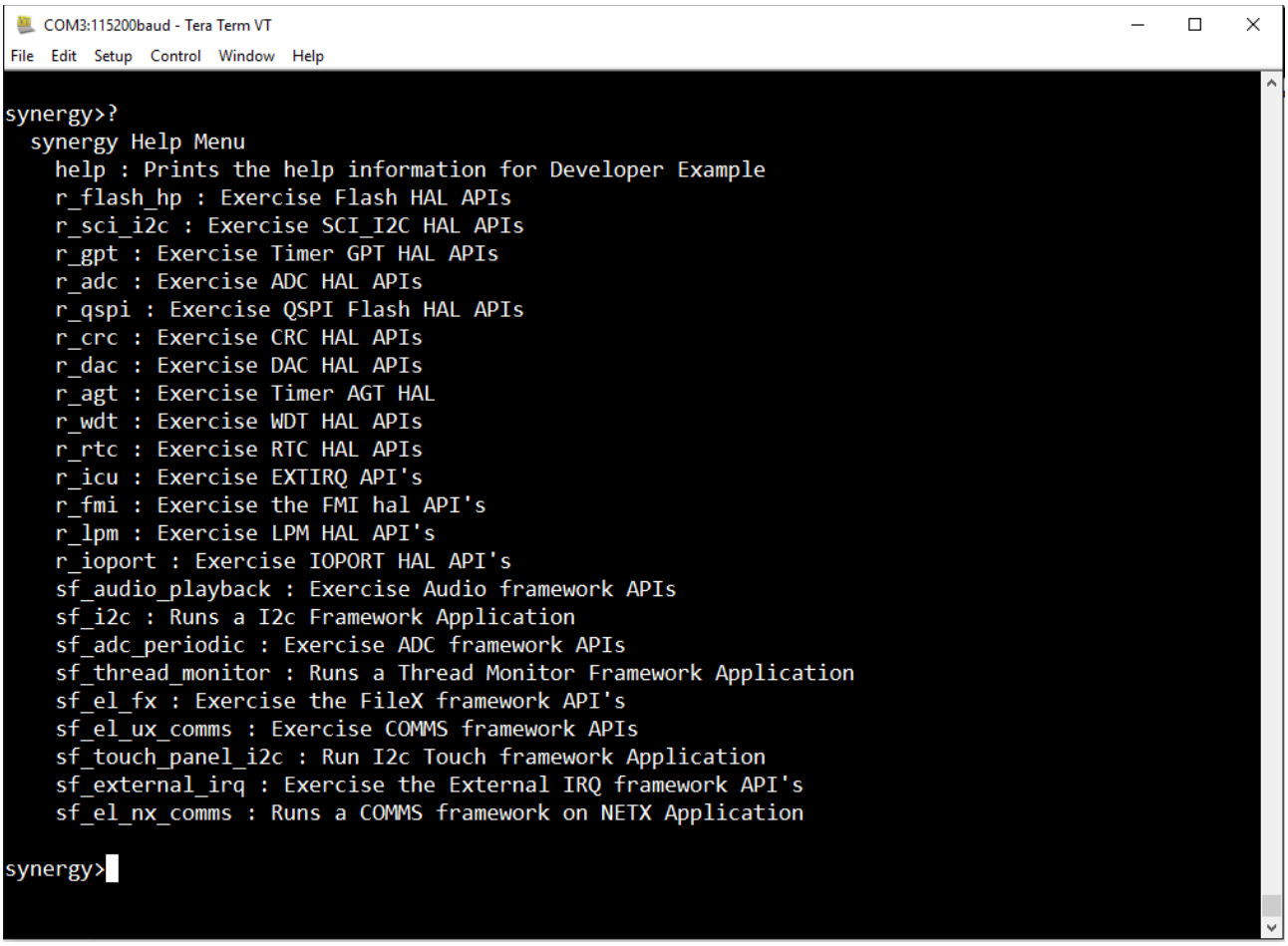
STEP 13: Choose the serial connection and choose the corresponding serial port.



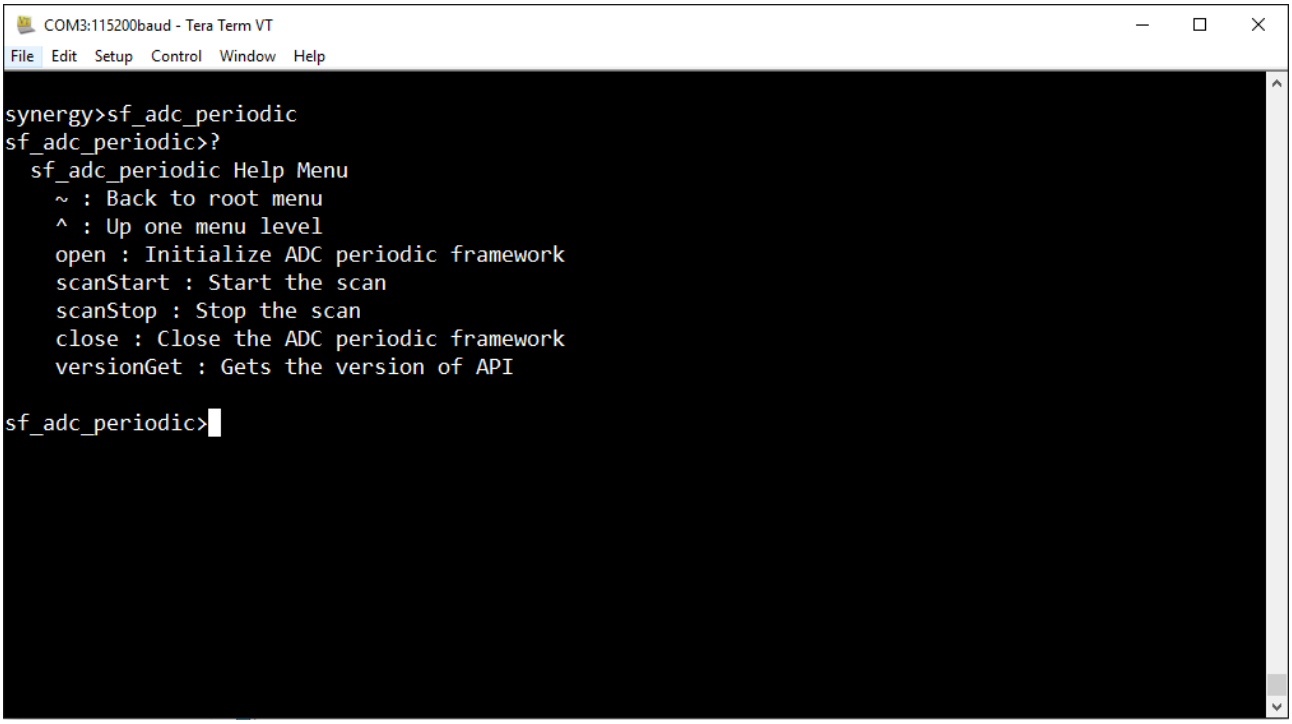
STEP 14: Press the Enter key to get the console prompt.



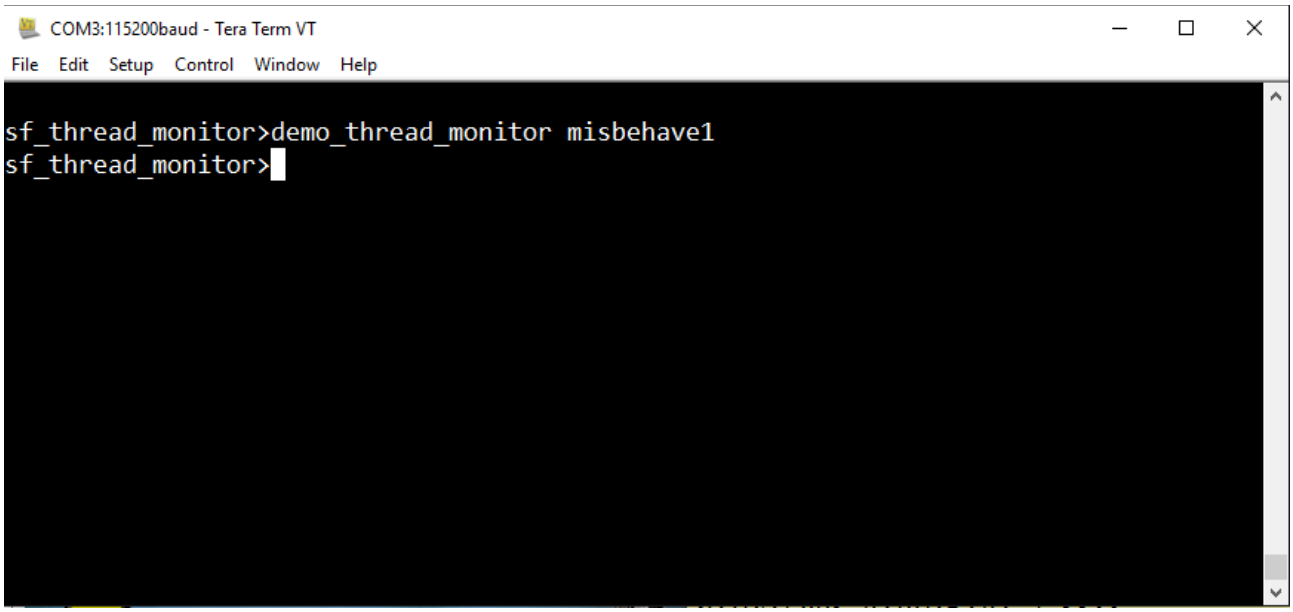
STEP 15: Type ? and press Enter to get the Help menu.



STEP 16: Type `sf_thread_monitor` in terminal and press Enter to access the Thread monitor framework submenu. For Help, type ? and press Enter.



STEP 17: Run the `demo_thread_monitor` command with parameter `misbehave1` as argument. The red LED of LED1 toggles 10 times and the LED1 turns off and the device resets. After reset LED1 lights up again.



20. Developer Example: WDT HAL driver

20.1 Introduction

The Watchdog Timer Developer Example uses the HAL APIs for WDT with output visible via an LED on the DK-S7G2 board.

The Developer Example demonstrates WDT functionality as follows:

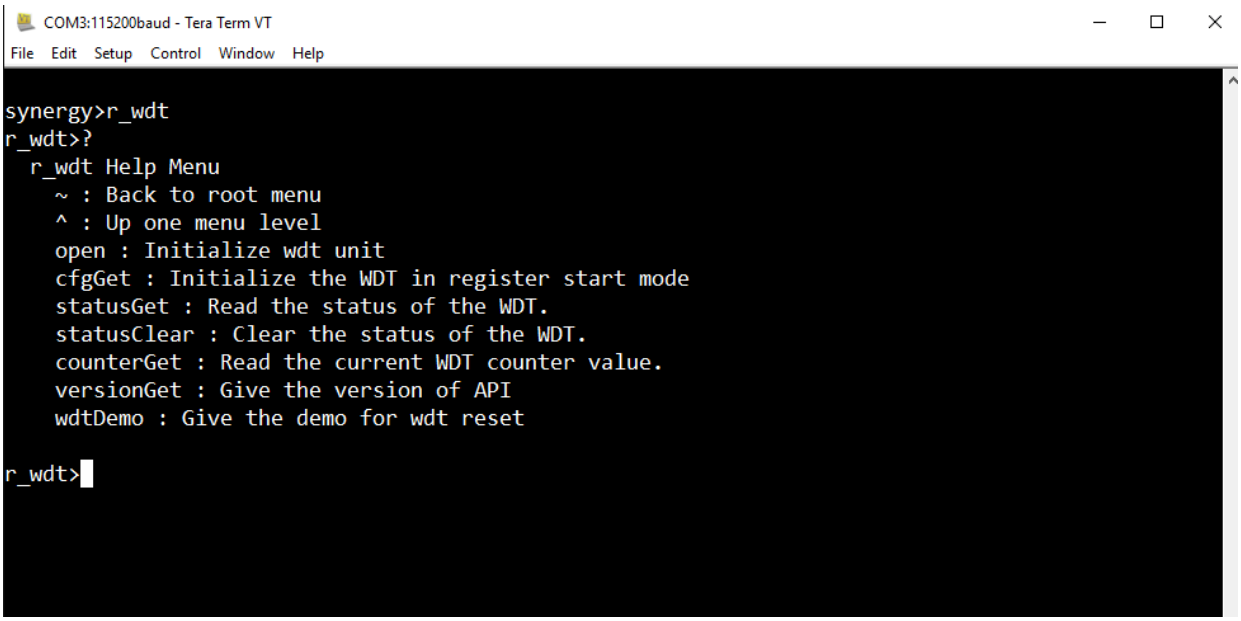
- Enables LED1 (GREEN) to demonstrate that the WDT is running.
- Enables LED1 (RED) to indicate WDT is about to expire.
- Disable LED1 to indicate that the WDT has expired.
- The WDT underflow count-down will be visible on the console.

20.2 Run the WDT HAL Driver application

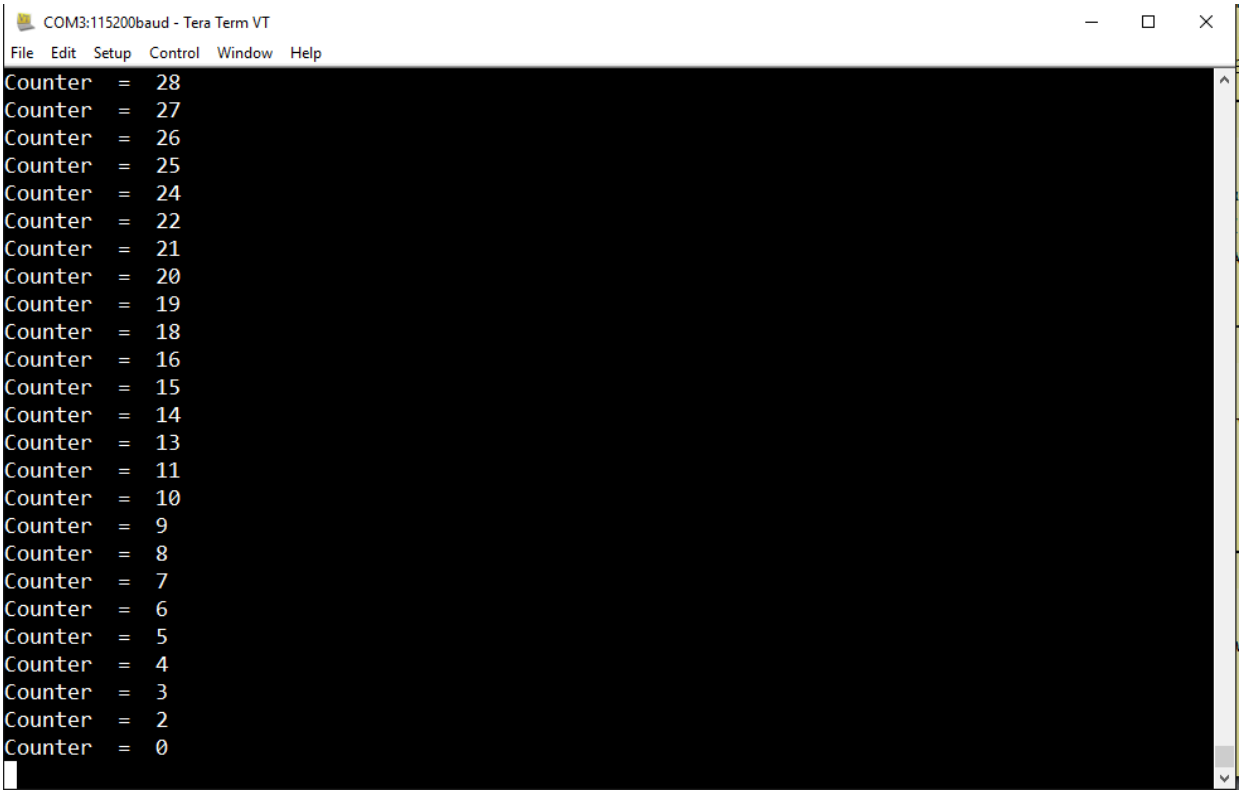
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the WDT HAL driver application, follow these steps:

STEP 1: Type `r_wdt`in the terminal and press Enter to access the WDT HAL submenu. For help, type ? and press Enter.



STEP 2: The `open` command starts the WDT (when it is in register start mode). The WDT Developer Example application will turn on the Green LED on the DK-S7G2 board. When you execute the `wdtDemo` command, the application resumes one of the sleeping thread hal_wdt_thread. The RED LED will turn on as the thread resumes. Loop in the thread takes more than the expected time to complete and the WDT resets the device.



NOTES:

- The thread monitor framework uses the WDT. In order to reset the device in the WDT demo, close the thread monitor framework. If you do not close the thread monitor, it will continually refresh the WDT and it will never reset the device.
- When using a J-Link debugger, the WDT counter does not count and therefore will not reset the device or generate an NMI (Refer Limitations of WDT in SSP User's Manual). To reset the device, remove the J-Link debugger, reset the device, and then execute the steps from STEP 9 of Build and Run a Developer Example application: Launching the terminal.

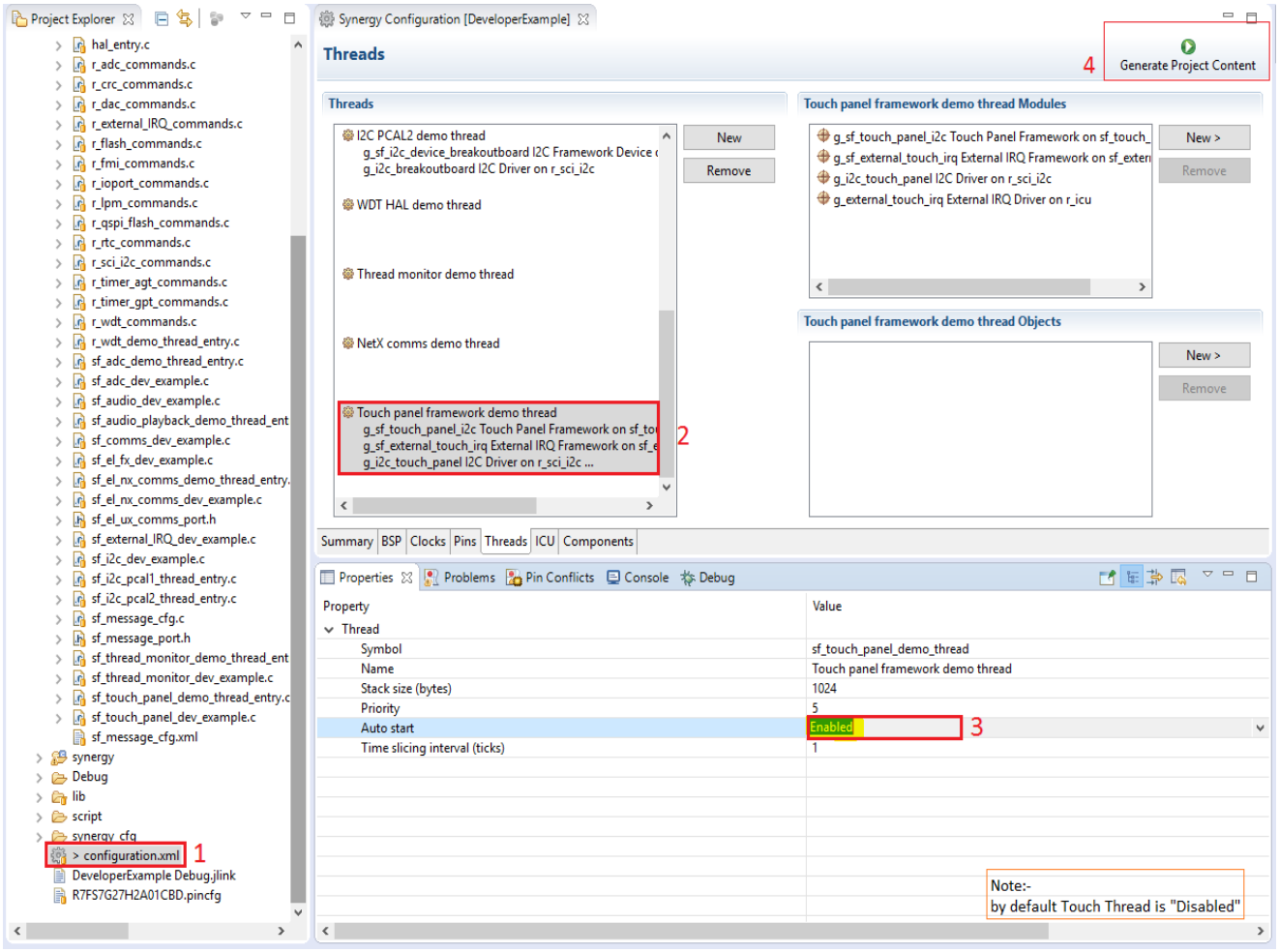
21. Developer Example: I2C Touch Panel Framework

21.1 Introduction

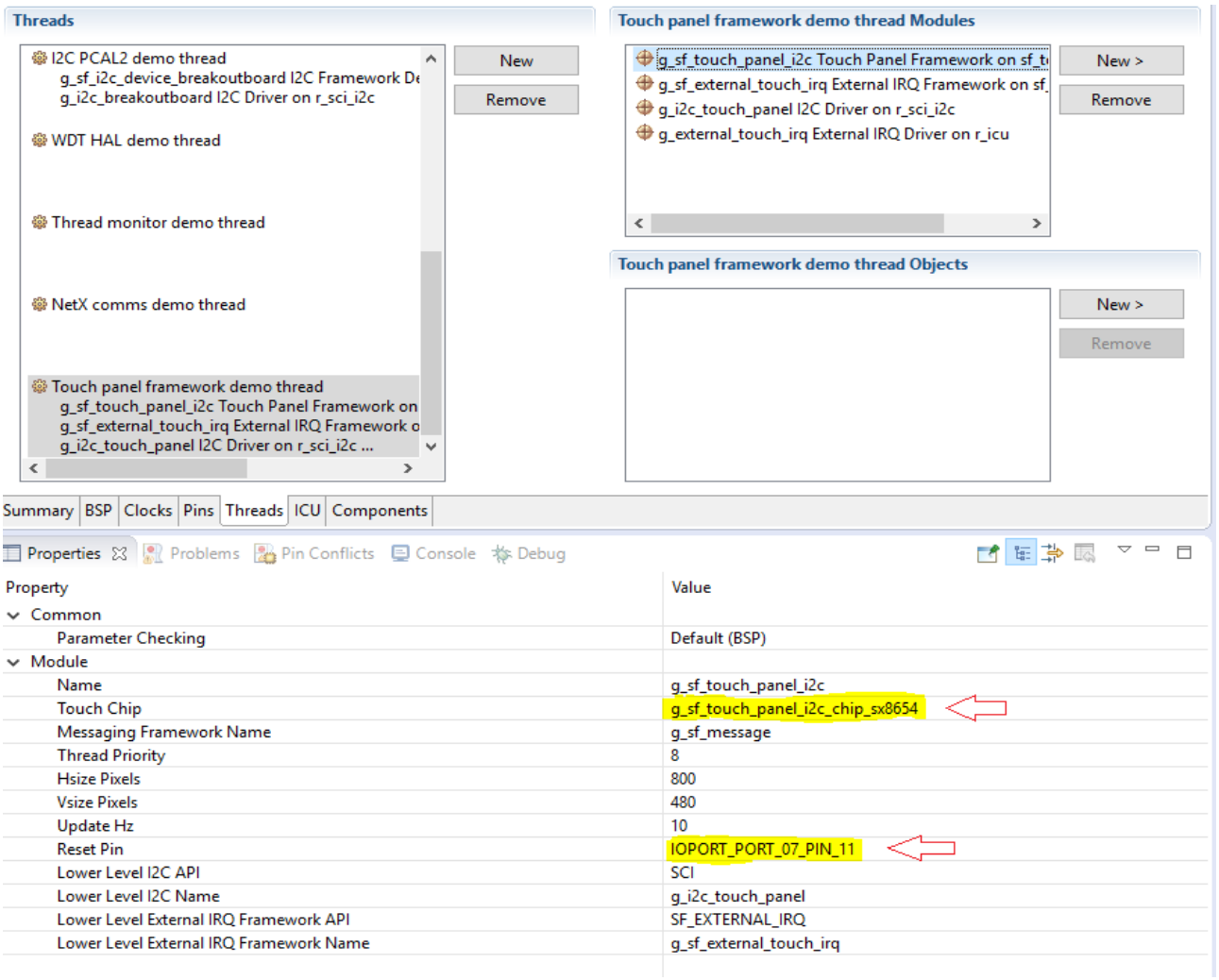
The I2C Touch Panel Framework uses External IRQ (Channel 7) and SCI I2C as low level driver and Messaging Framework to deliver the Touch Events to the respective subscribers. This Developer Example explains how to configure I2C touch panel framework instance, External IRQ framework instance, low level I2C driver, and the Messaging Framework in order to get touch event information from the DKS7 board. This document will also give step by step instructions as how to invoke the Developer Example for I2C touch panel framework on a DKS7 board. I2C Touch Panel Configuration Step

STEP 1: Open configuration.xml file, select Touch Thread, go to Properties tab, enable the Touch Thread to auto start. Click “Generate Project Content” to update configuration file and Build the project again.

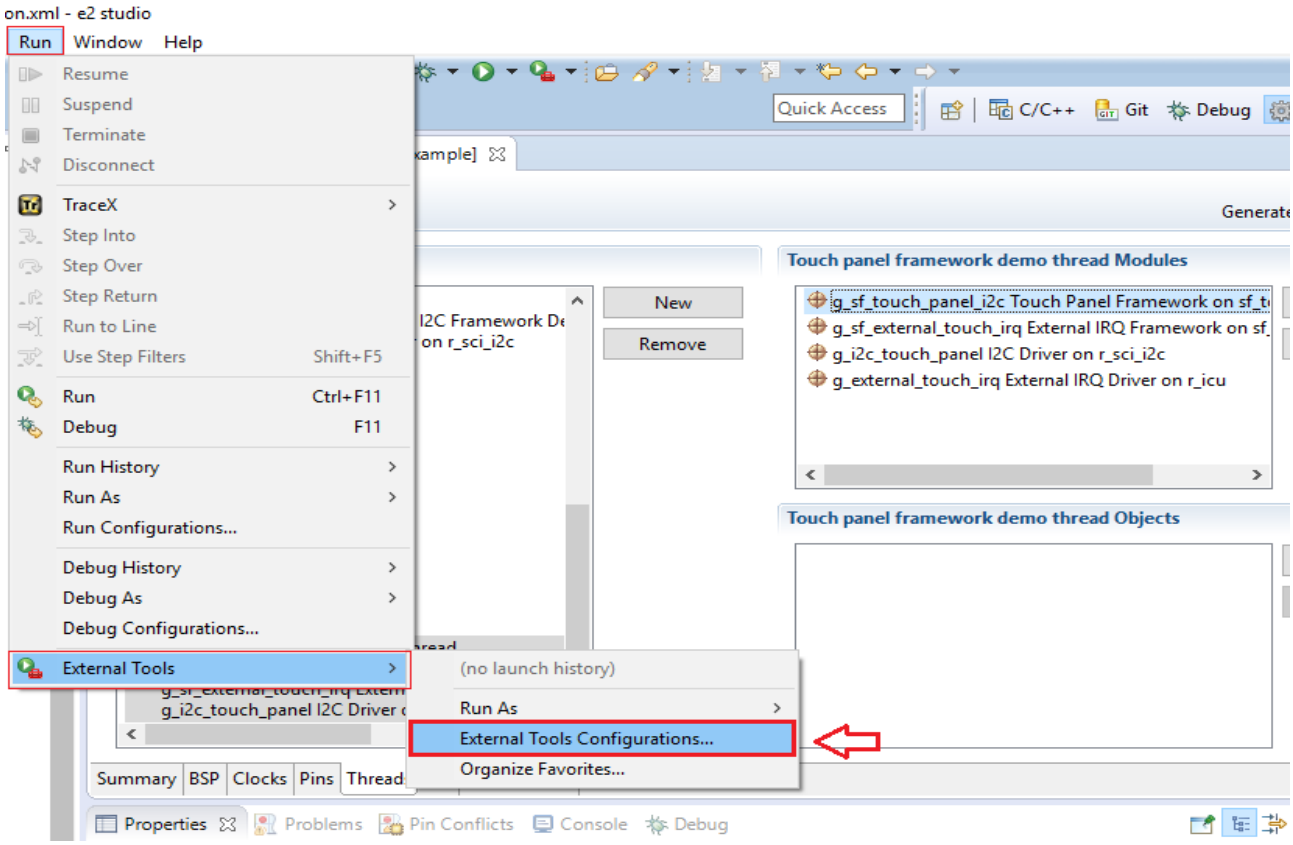
NOTE: To run the other I2C modules disable the “Auto start” thread.



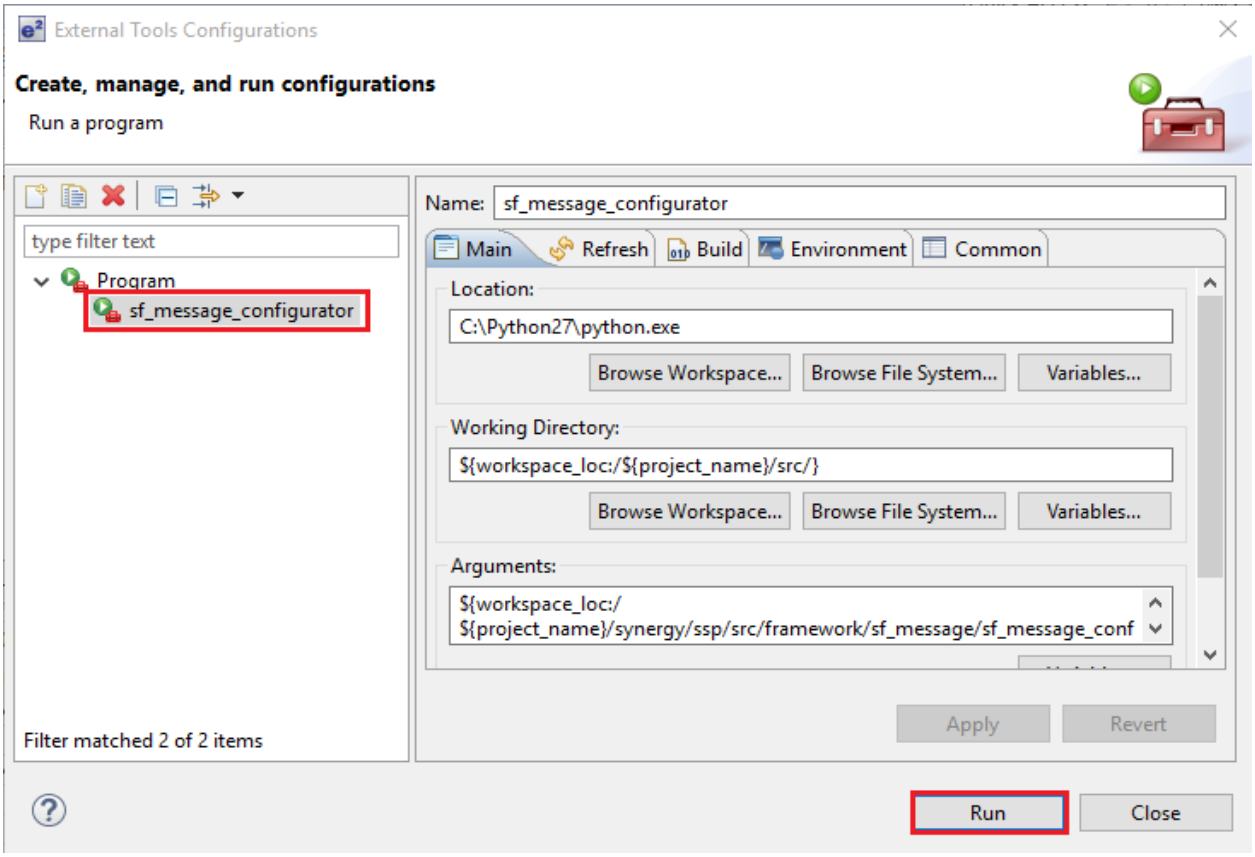
STEP 2: In Touch Thread go to Touch Thread Module section and select the I2C touch panel framework and do the following configuration as shown. Make sure the Touch Chip and Reset Pin configuration is same as in snapshot.



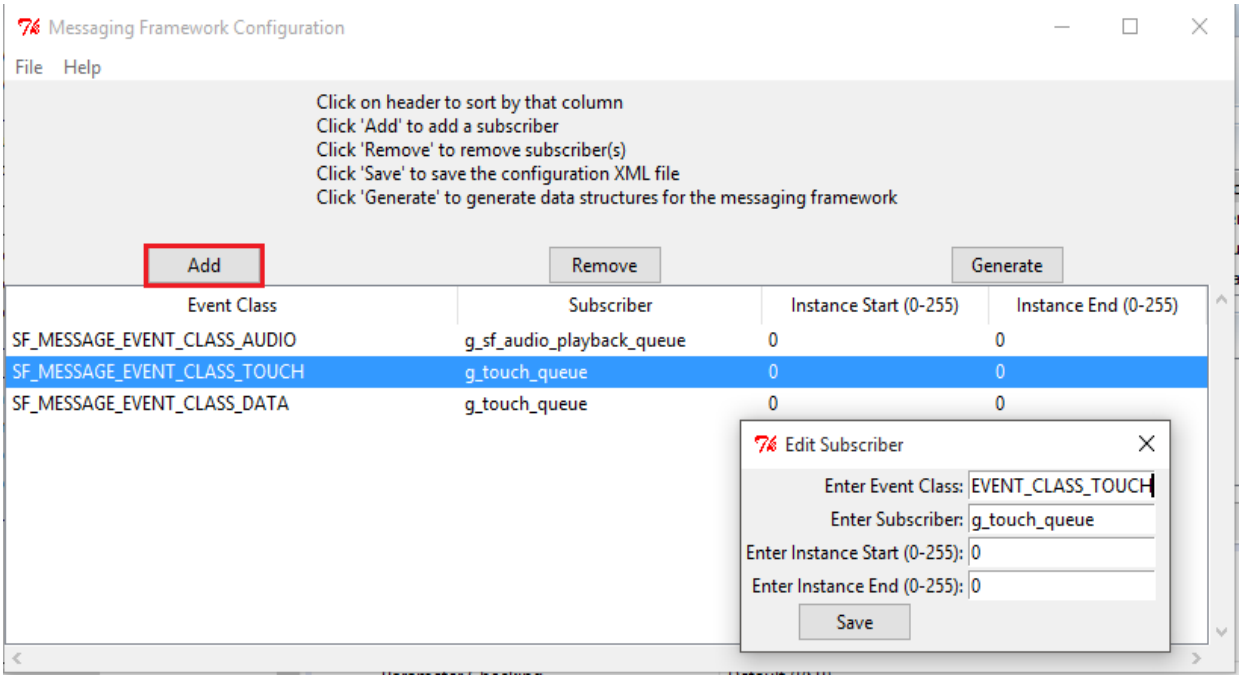
STEP 3: I2C Touch Framework require Messaging Framework to pass the touch event message. In order to configure that, go to the **Run menu** ->**External Tool**->**External Tool Configurations**.



The **External tools configuration window** displays. Select sf_message_configurator and click Run.



STEP 4: Click Add to configure the subscribers.



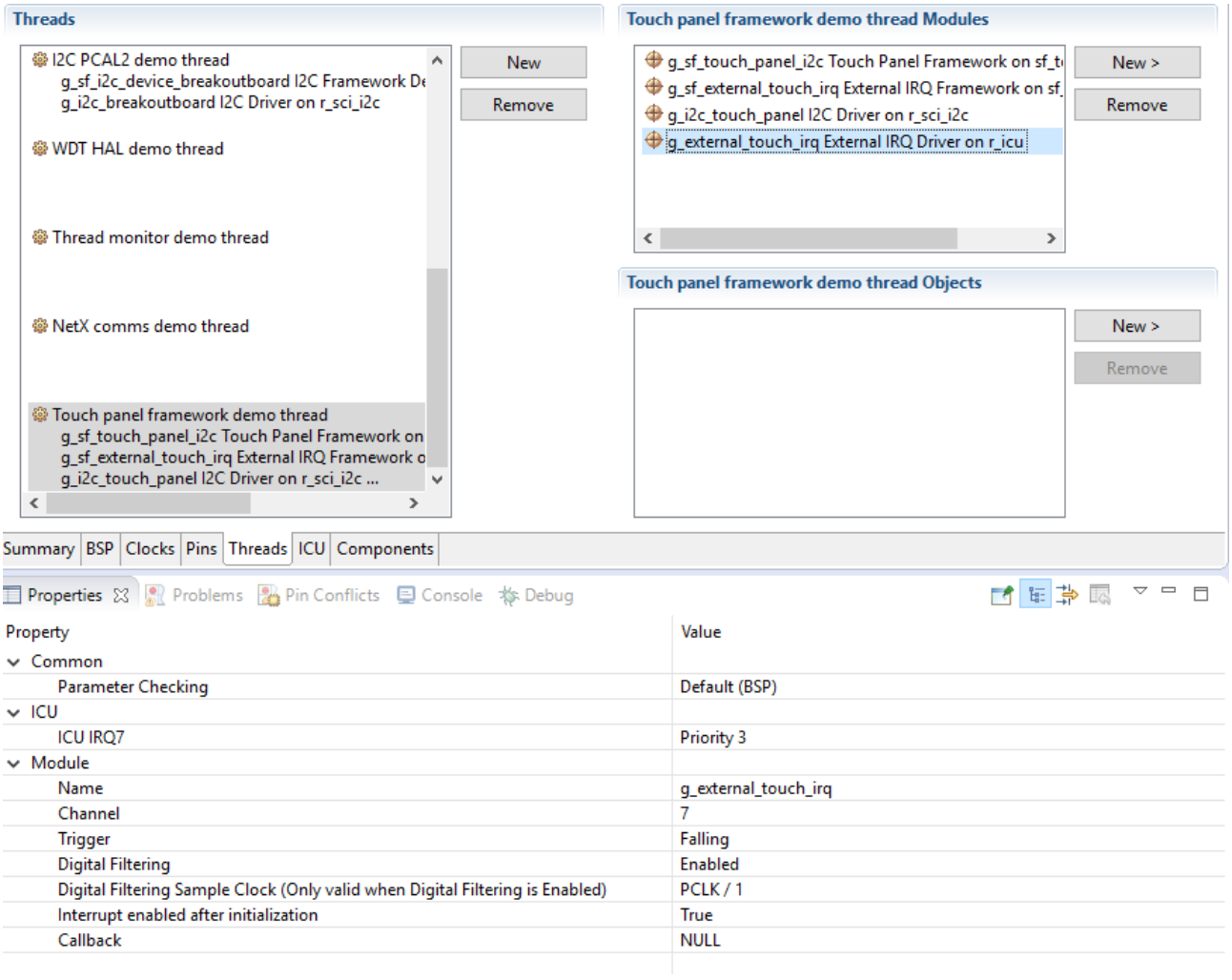
Event Class -> SF_MESSAGE_EVENT_CLASS_TOUCH

Subscriber ->q_touch_queue (queue created via configuration.xml).

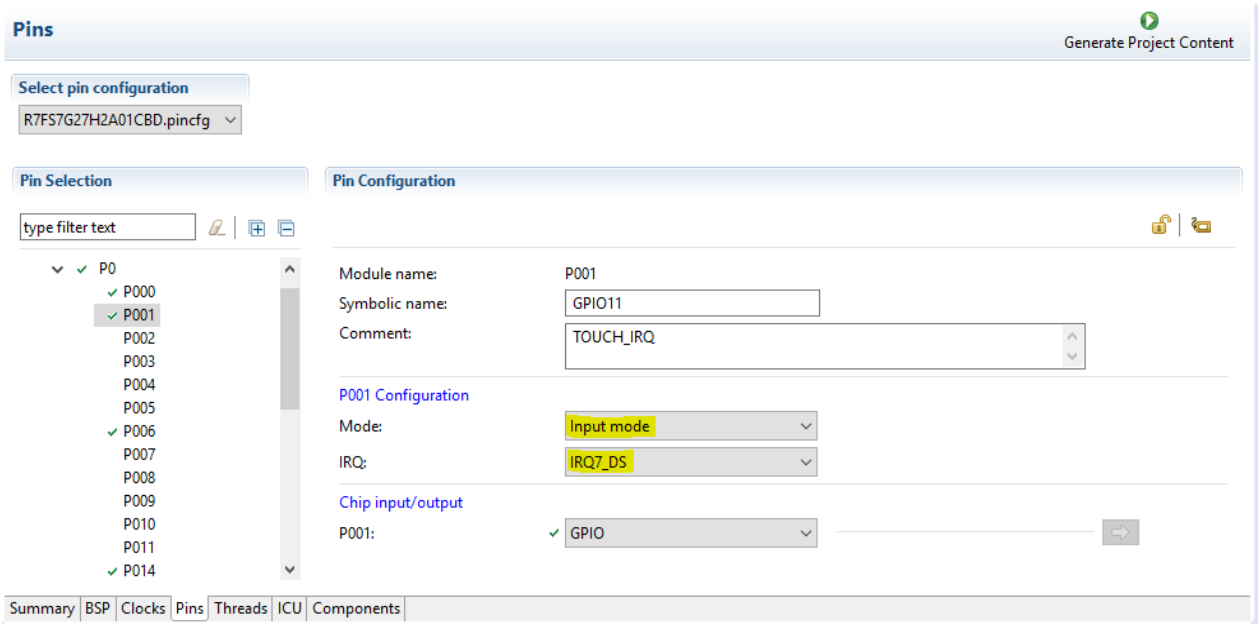
Instance -> give zero for both start and stop.

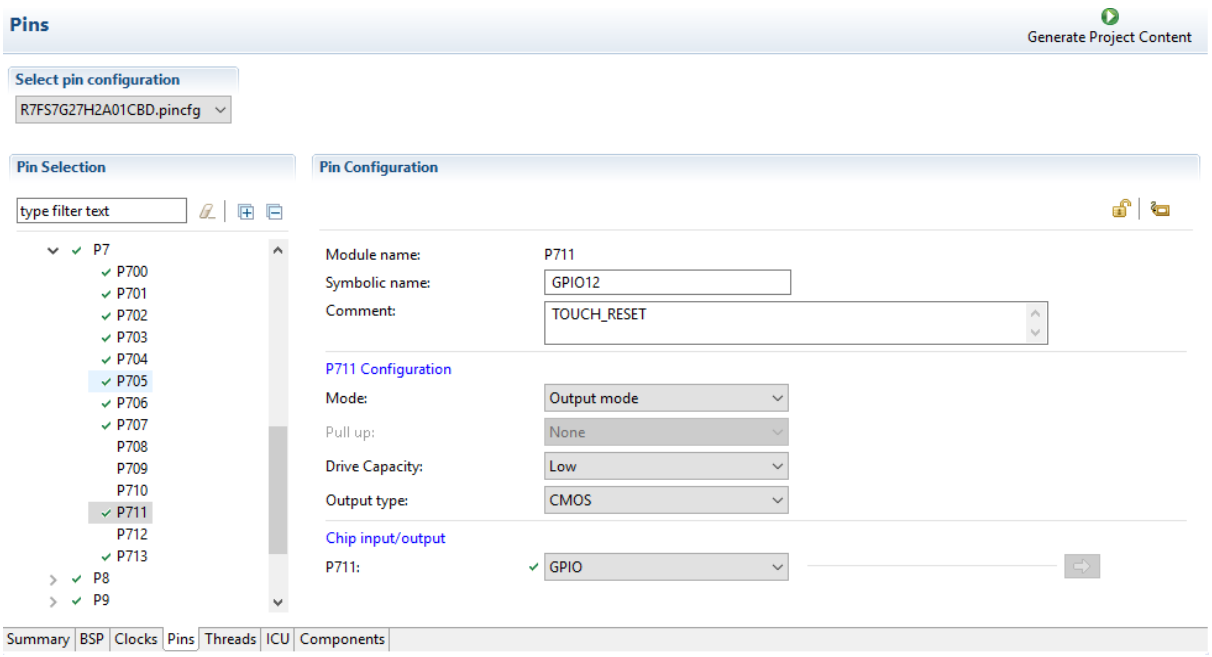
Save and Generate; this will generate sf_message_cfg.c and sf_message_port.h file in Developer Example.

STEP 5: In configuration.xml file, select Touch Thread and go to External IRQ diver andmake sure the following configuration is as shown in snapshot.

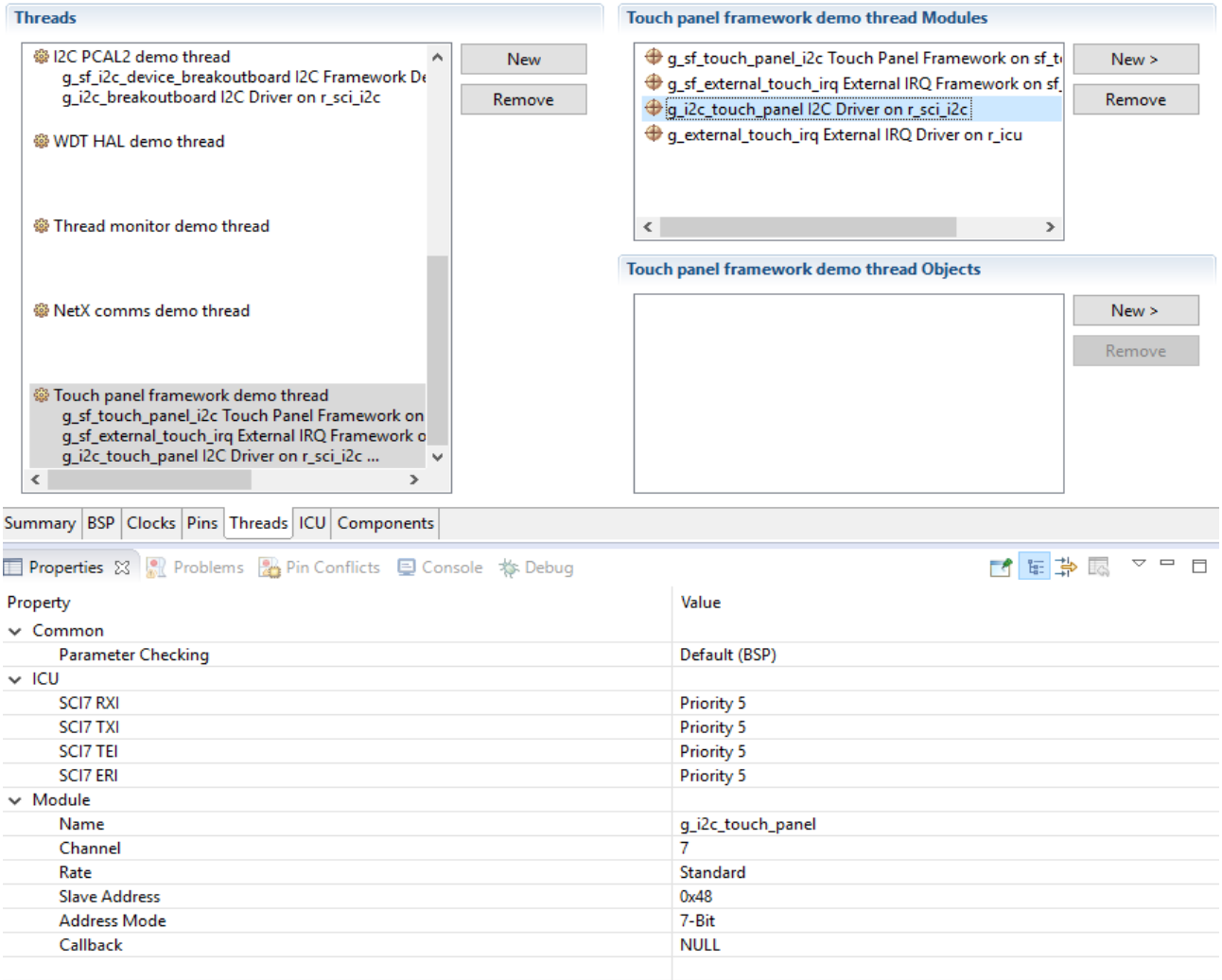


STEP 6: Open the Pins tab under configuration.xml and configure the PORT_00_PIN_01 and PORT_07_ PIN_11 as shown.





STEP 7: Select i2c driver, and check the following configuration.



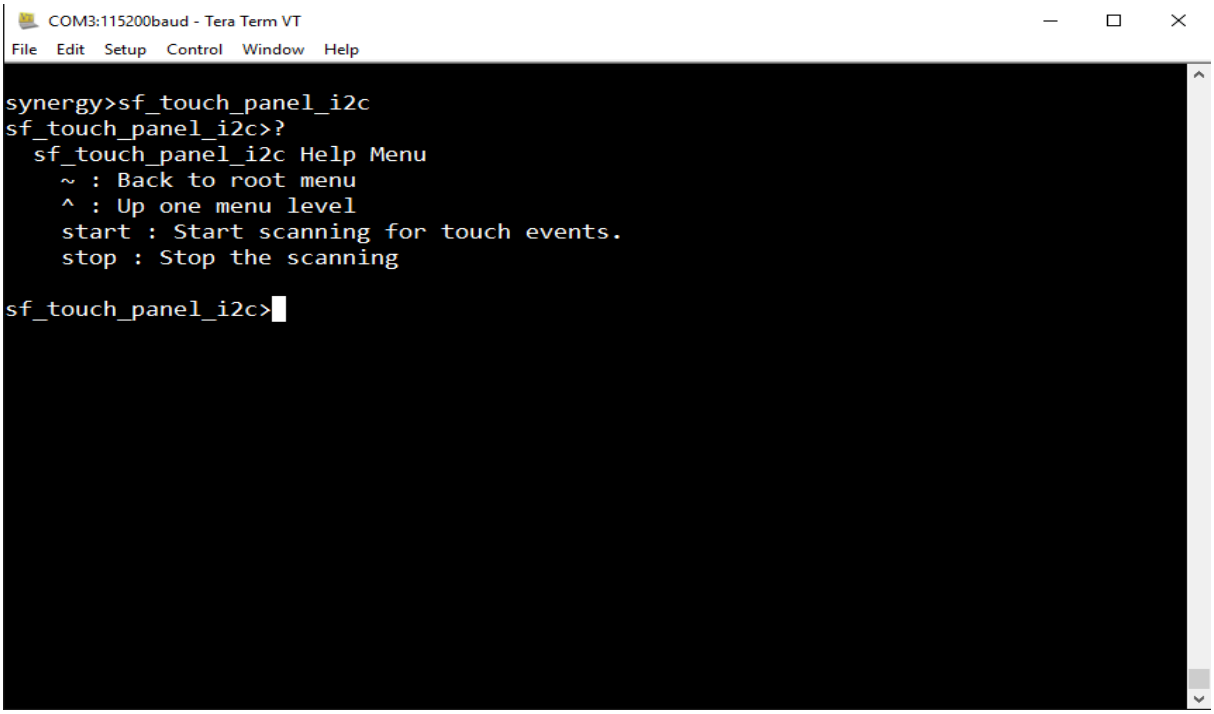
NOTE: The address of i2c touch device ic is 0x48.

21.2 Run the I2C Touch Panel Framework application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

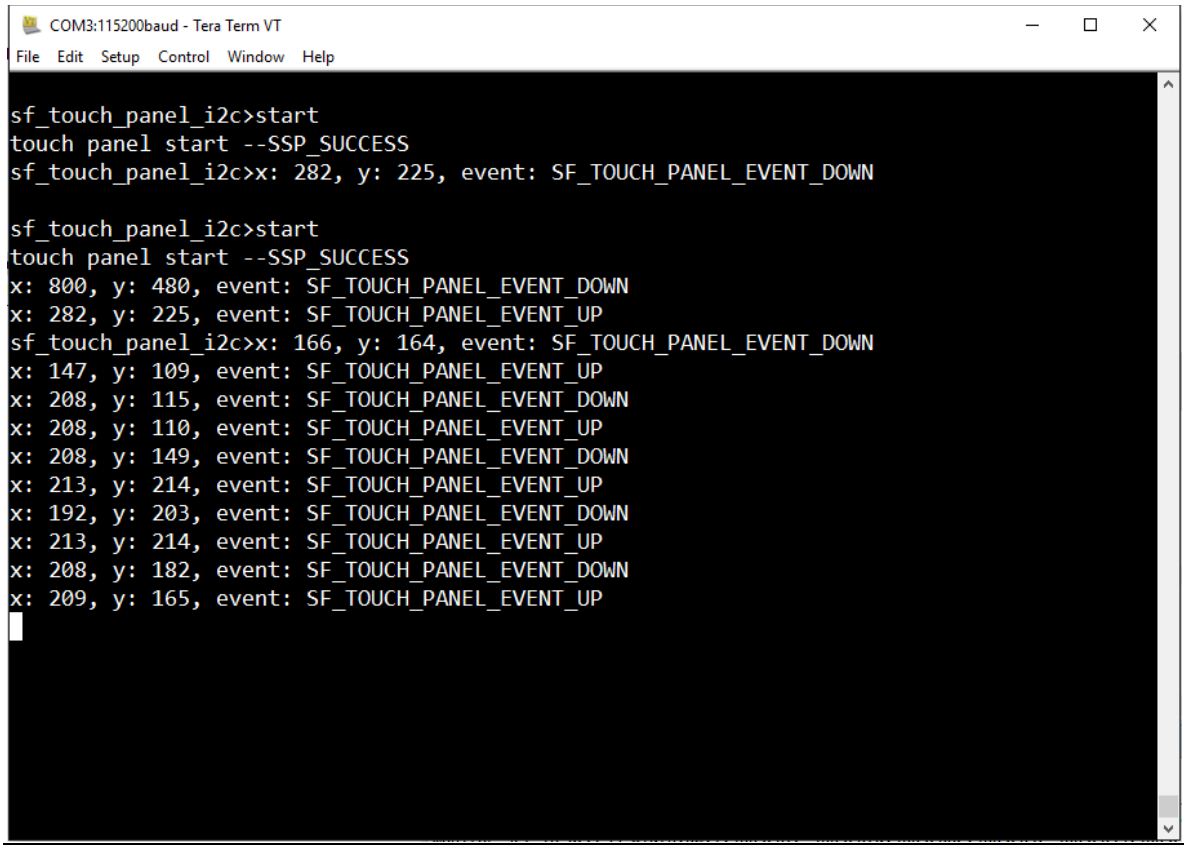
To run the I2C Touch Panel Framework application, follow these steps:

STEP 1: type `sf_touch_panel_i2c` in terminal and press Enter to access the I2C Touch Panel framework sub-menu. For Help, type "?"and press Enter.

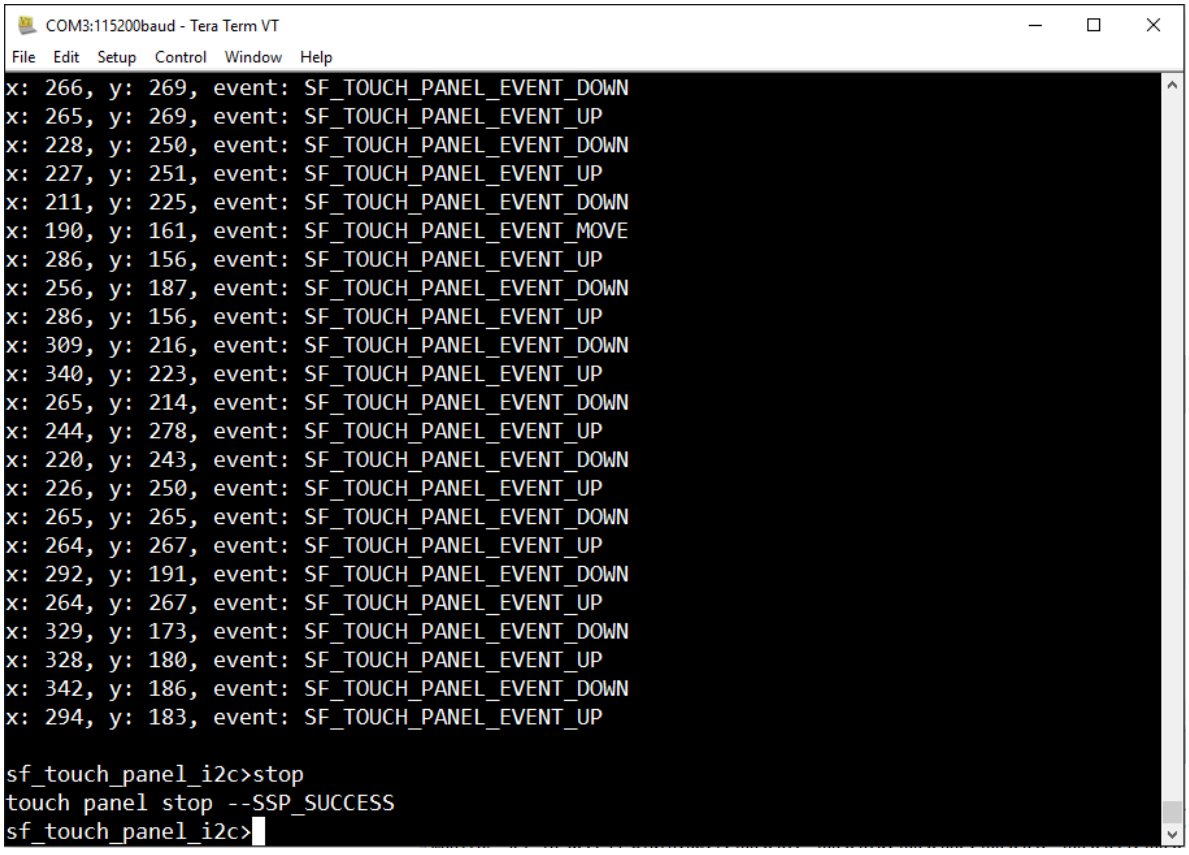


STEP 2: Enter the `start` command and touch the Touch Panel.

NOTE: For the sf_touchpanel_I2C framework to work properly there should be a touch event between `stop` and `start` command, for example, `stop` →<touch>→ `start.` However `start` →<touch>→ `start` is a valid combination and would cause the touch event to be generated. This is how the framework is implemented by the SSP and it is not a limitation of the Developer Example.



STEP 3: In order to `stop` touch sensing enter the Stop command.



22. Developer Example: FMI HAL driver

22.1 Introduction

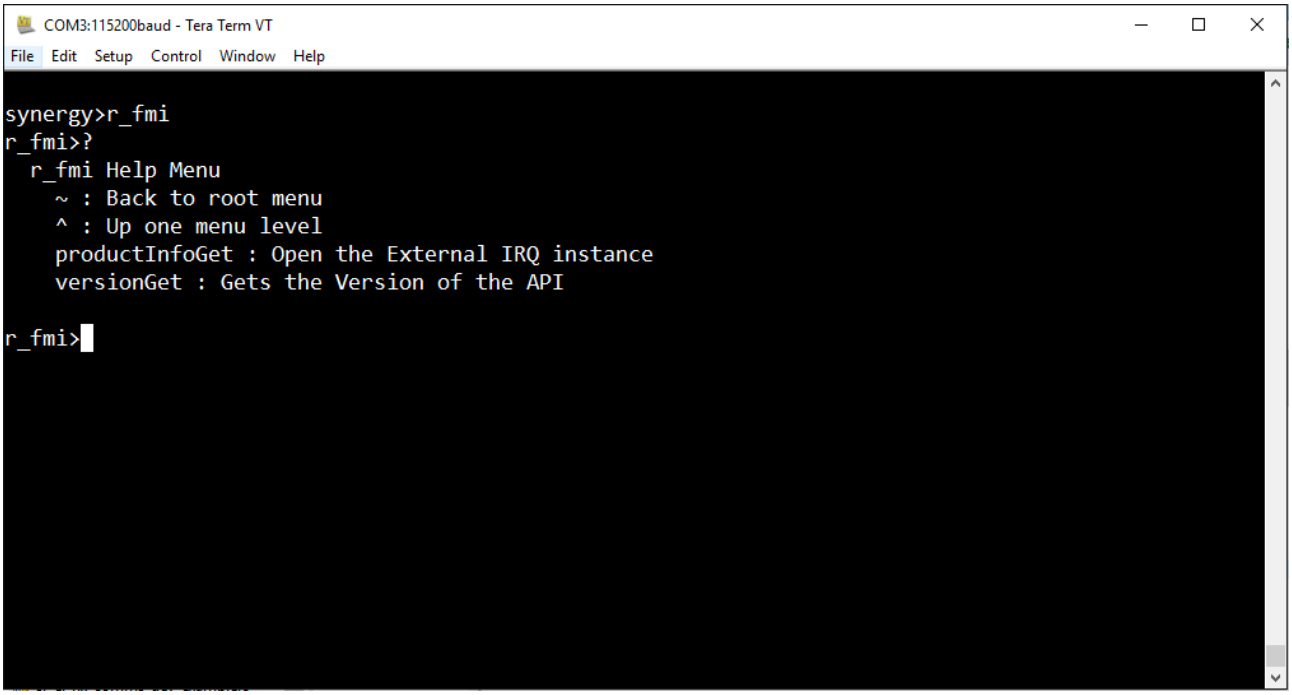
The FMI HAL Interface is a generic API for reading records from the Factory MCU Information flash table. The Developer Example demonstrates the FMI HAL by displaying the MCU information onto the console.

22.2 Run the FMI HAL Driver application

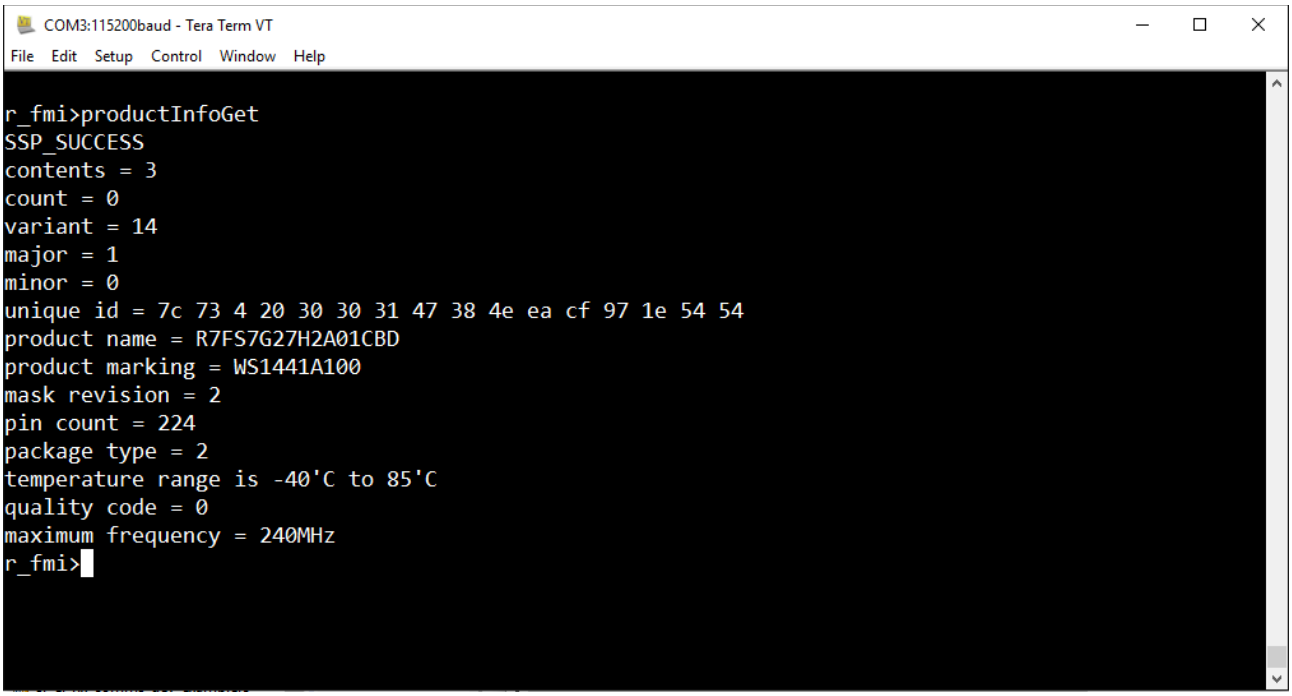
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the FMI HAL driver application, follow these steps:

STEP 1: Type `r_fmi` in the terminal and press Enter to access the FMI HAL submenu. For help, type ? and press Enter.



STEP 2: Type `productInfoGet` command to get the details of the device.



23. Developer Example: LPM HAL driver

23.1 Introduction

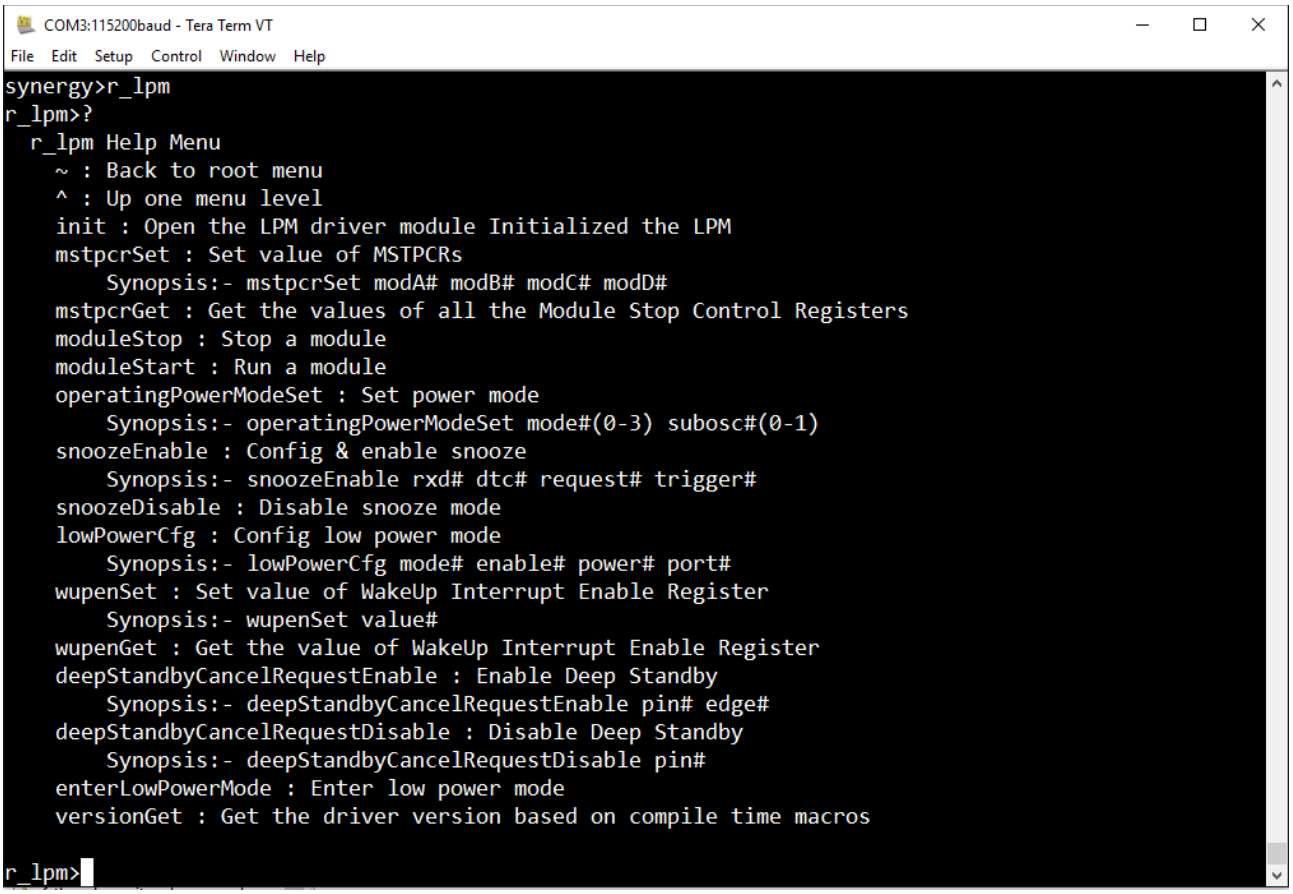
The Developer Example demonstrates the LPM HAL driver APIs. LPM module is used to put device to sleep, software standby and deep software standby mode. It is also possible to stop or start any module.

23.2 Run the LPM HAL driver application

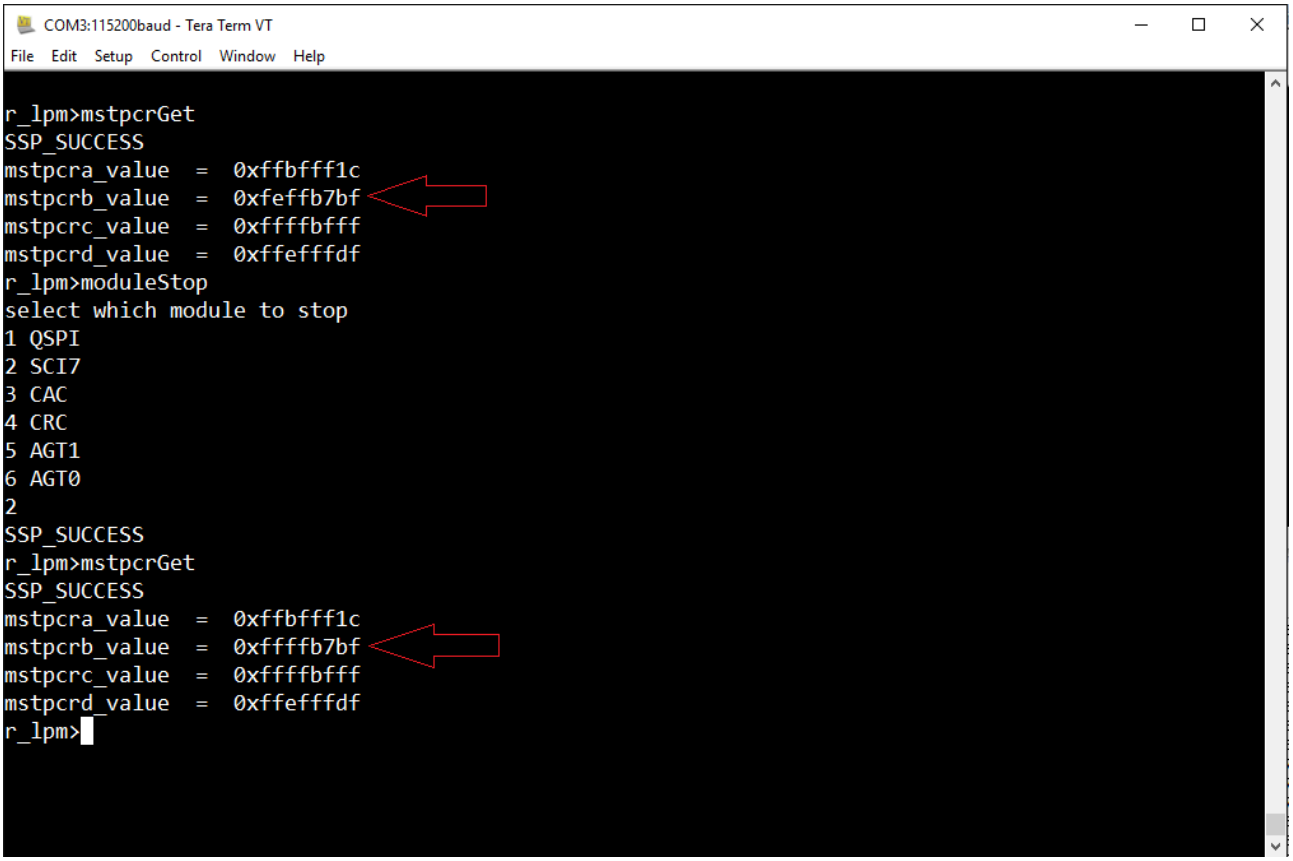
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the LPM HAL driver application, follow these steps:

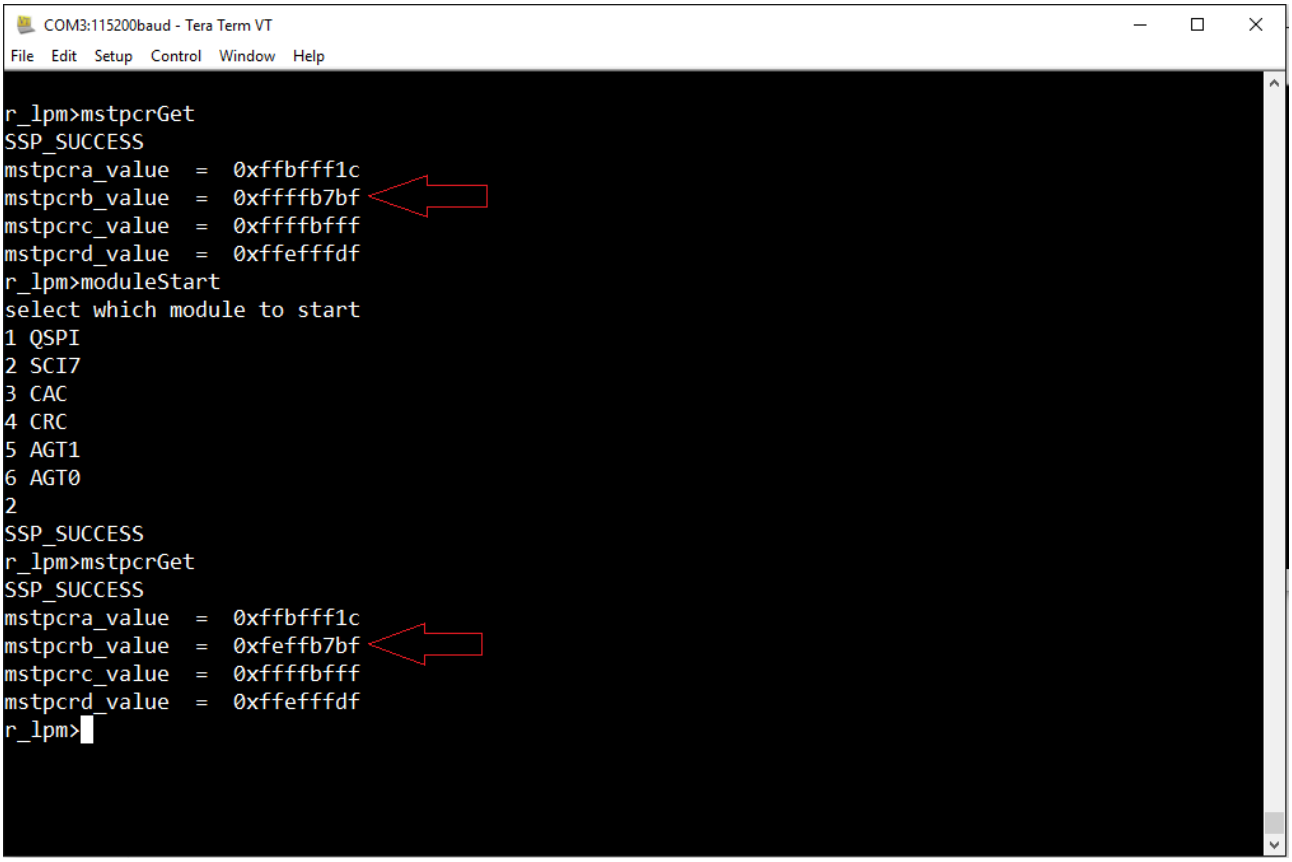
STEP 1: Type `r_lpm` in the terminal and press Enter to access the LPMHAL submenu. For help, type ? and press Enter.



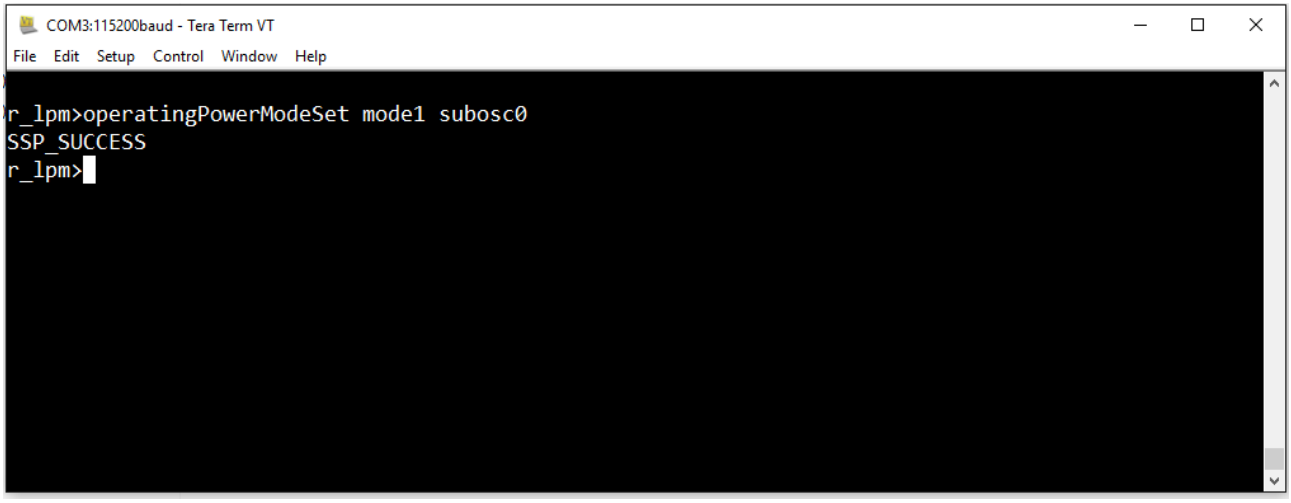
STEP 2: To stop a module type `moduleStop` command. To validate we can use `mstpcrGet` command before and after executing `moduleStop` command. In the below picture SCI7 is stopped, which can be noticed by the value in the mstpcrb value. For example, mstpcrb value before stopping SCI7 was 0xfeffb7bf and after stopping SCI7 is 0xffffb7bf



STEP 3: To stop a module type `moduleStart` command. To validate you can use `mstpcrGet` command before and after executing `moduleStart` command. In the below picture SCI7 is started, which can be noticed by the value in the mstpcrb value. For example, mstpcrb value before starting SCI7 was 0xffffb7bf and after starting SCI7 is 0xfeffb7bf.

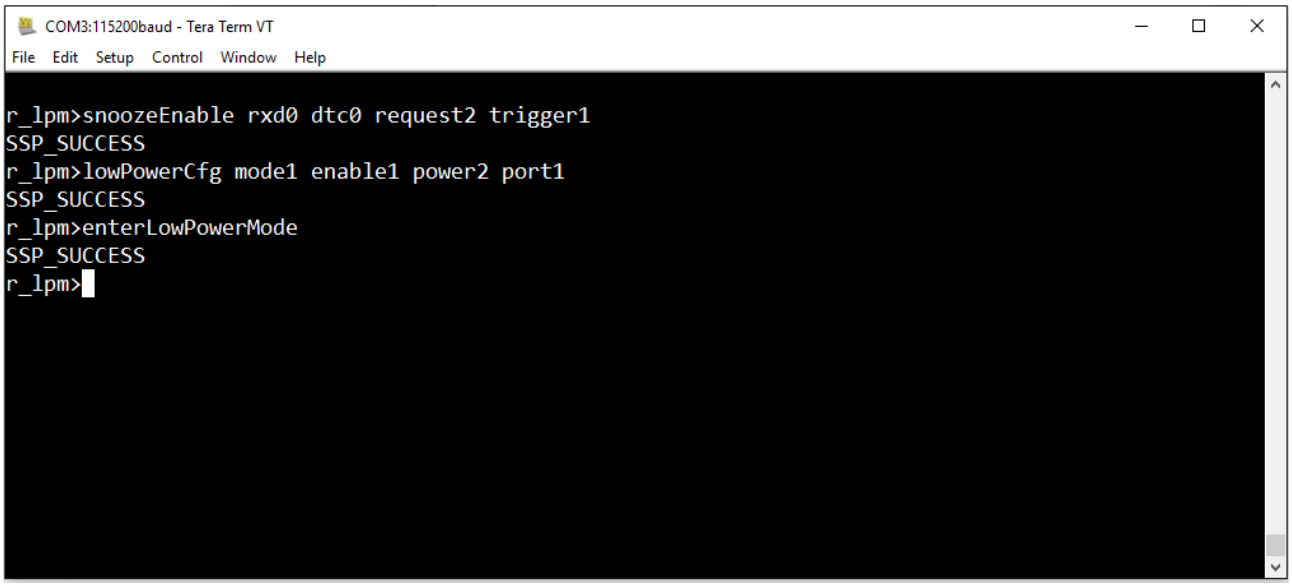


STEP 4: To set operating mode type `operatingPowerModeSet` command.



STEP 5: To snooze an interrupt in a low power mode execute following commands sequentially.

1. `snoozeEnable`
2. `lowPowerCfg`
3. `enterLowPowerMode`



NOTE:

- `moduleStart` and `moduleStop` command allows only QSPI, SCI7, CAC, CRC, AGT1 and AGT0 modules to start and stop respectively. This is not the limitation of SSP.
- If SRAM is put to sleep it resets board and if USBFS is put to sleep it blocks the console so Developer Example restricts user to put certain modules like SRAM and USBFS into sleep.
- To set the device into the low power mode, user needs to run `lowPowerCfg` command followed by `enterLowPowerMode` command.

24. Developer Example: External IRQ Framework

24.1 Introduction

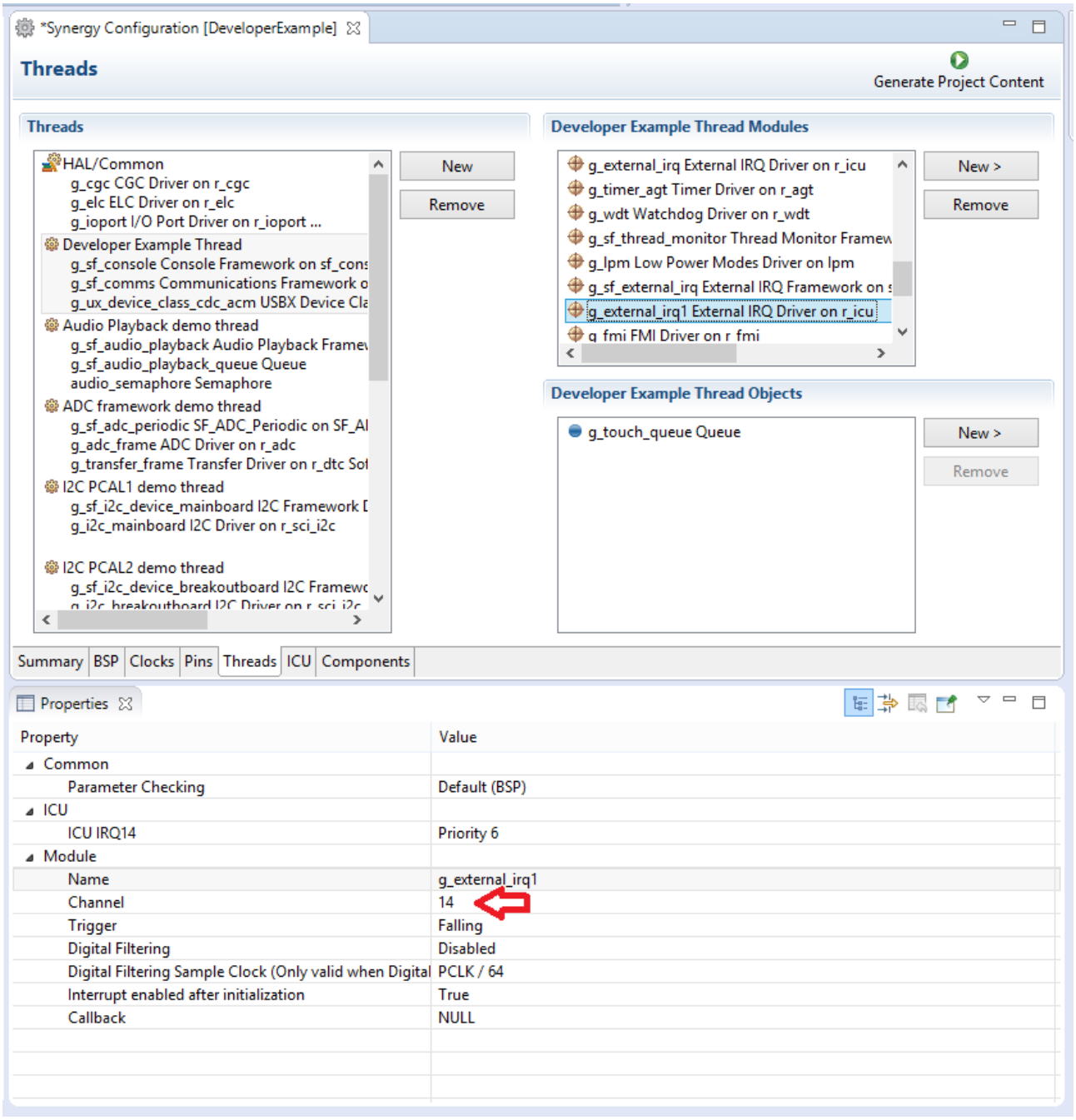
The External IRQ Framework uses External IRQ HAL module as a low level driver and waits for the user to give an external interrupt. In Developer Example, switch S2 is configured as a source of external IRQ. When the `wait` command is executed, the thread waits for an external IRQ, which can be given by pressing switch S2. The External IRQ Framework is integrated to CLI in Developer example from which all the APIs of External IRQ can be exercised.

24.2 Run the External IRQ Framework application

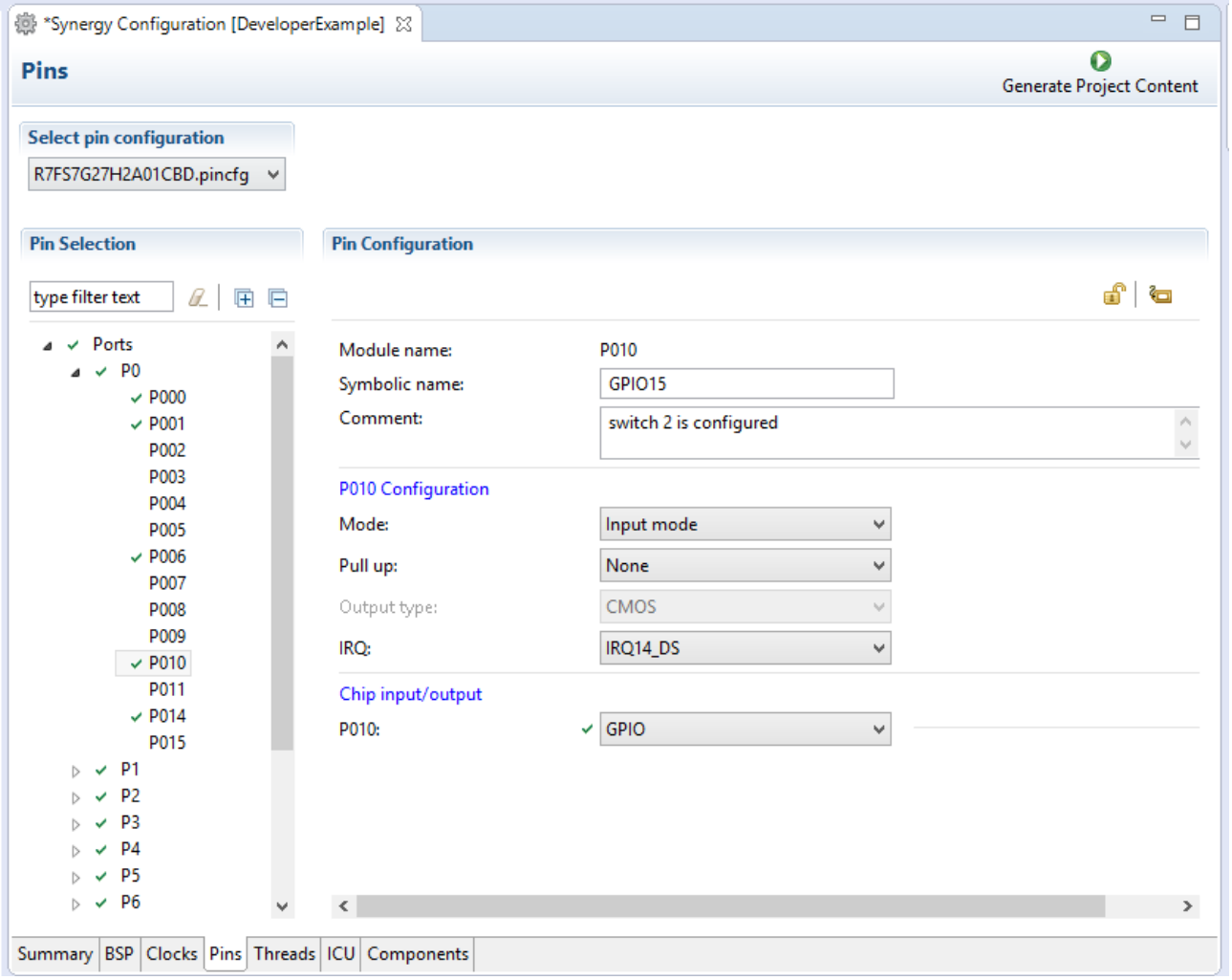
The following are the steps to configure switch S2 for an External IRQ HAL driver.

NOTE: In S7G2-DK V2.2 switch S2 is connected to P0_10 with IRQ channel 14.

The screen shot below shows the property of an external IRQ HAL module mapped to external IRQ framework.



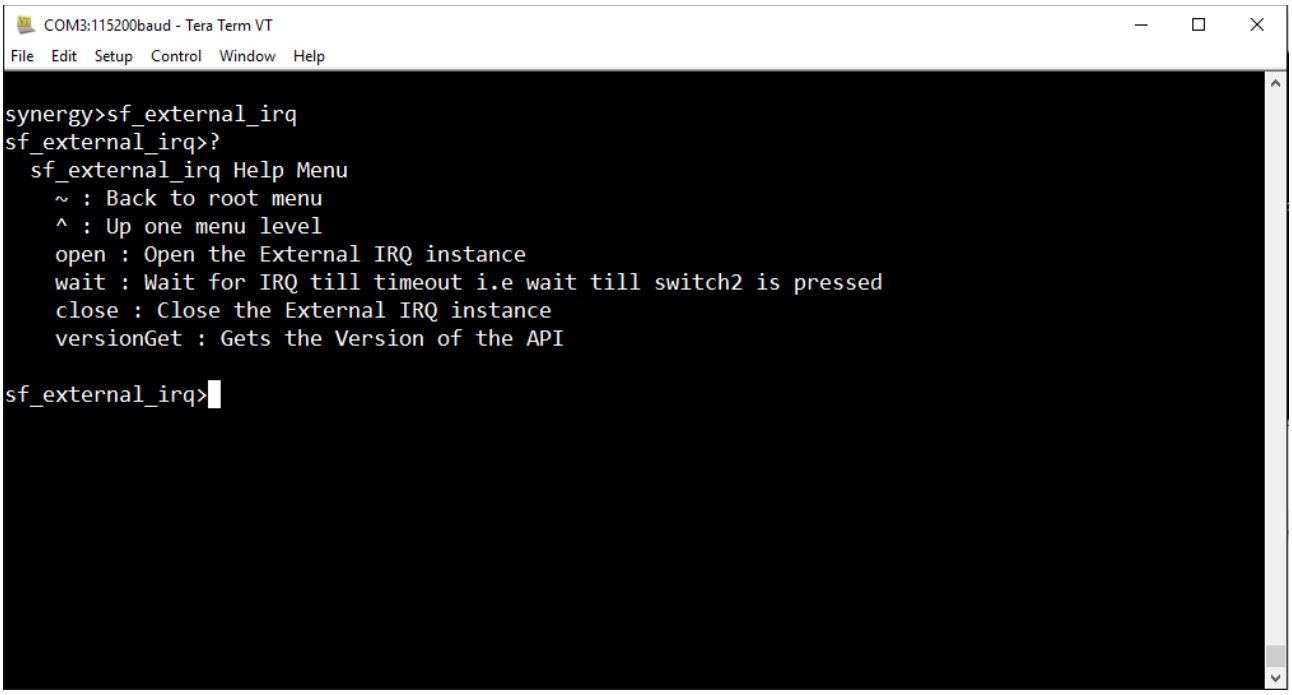
The screen shot below shows the pin configuration made for switch S2.



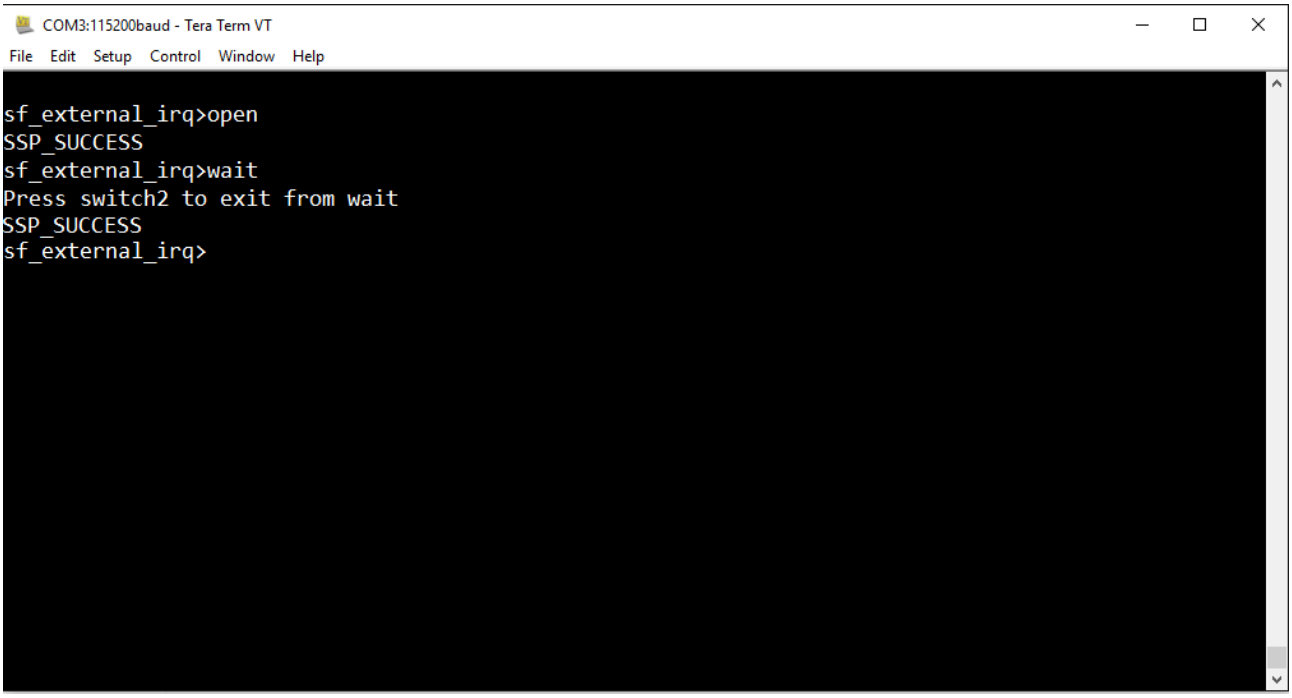
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the External IRQ Framework application, follow these steps:

STEP 1: Type `sf_external_irq` in the terminal and press Enter to access the External IRQ Framework submenu. For help, type ? and press Enter.



STEP 2: To wait for an external IRQevent, type `open` command followed by the `wait` command. Once the `wait` command is executed a message is displayed on console and LED1 turns ON indicating it is waiting for external input. Press switch S2 to generate external interrupt.



25. Developer Example: IOPort HAL driver

25.1 Introduction

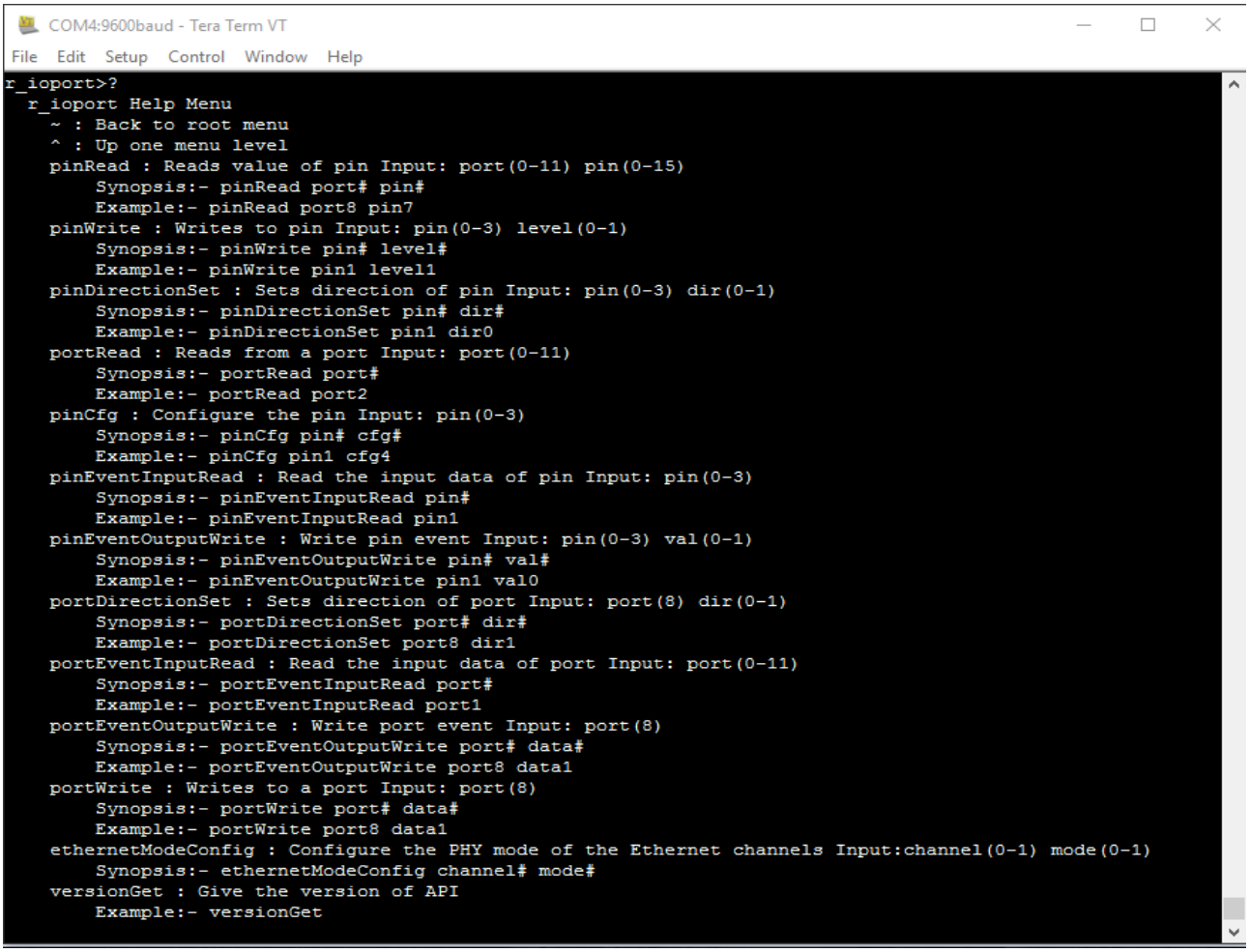
IOPort HAL Developer Example will demonstrate IOPorts HAL module and exercises the API's of IOPort HAL. This Developer Example uses only 4 pins of port8 (Pin7,8,9,10) to demonstrate the IO functionality andExercising the HAL API's of IOPort module. Other pins are mapped to different peripherals and hence not used.

25.2 Run the IOPort HAL Driver application

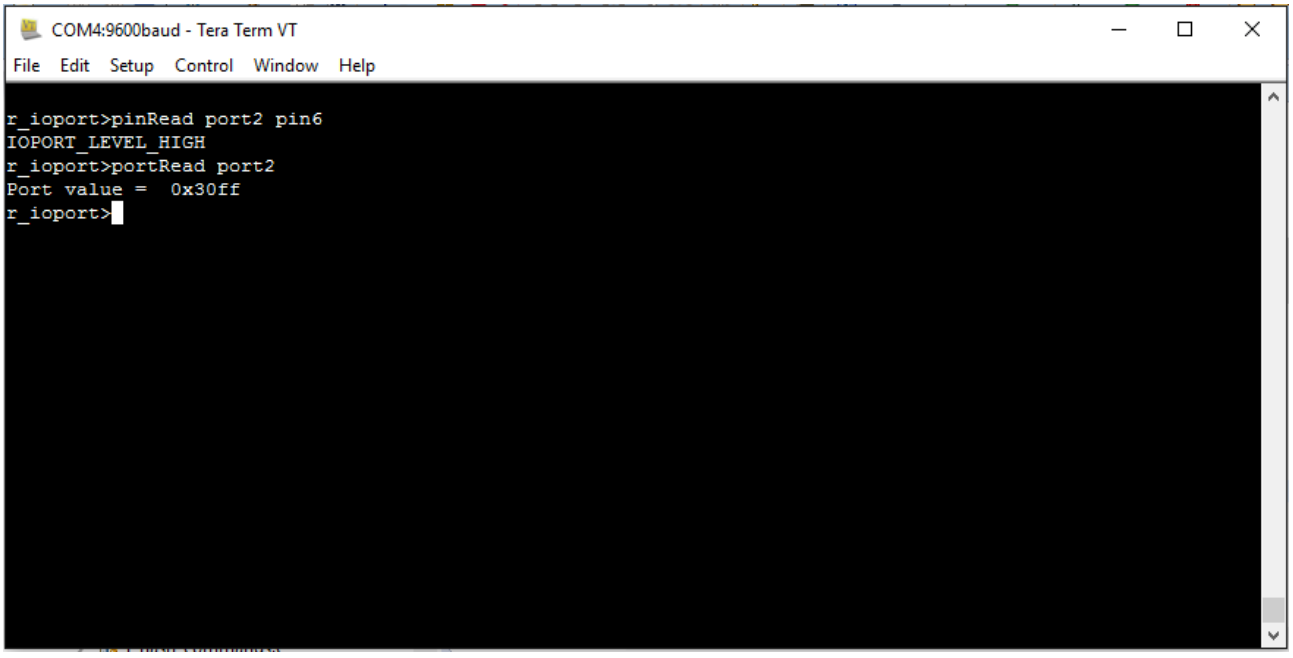
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the HAL IOPort driver application, follow these steps:

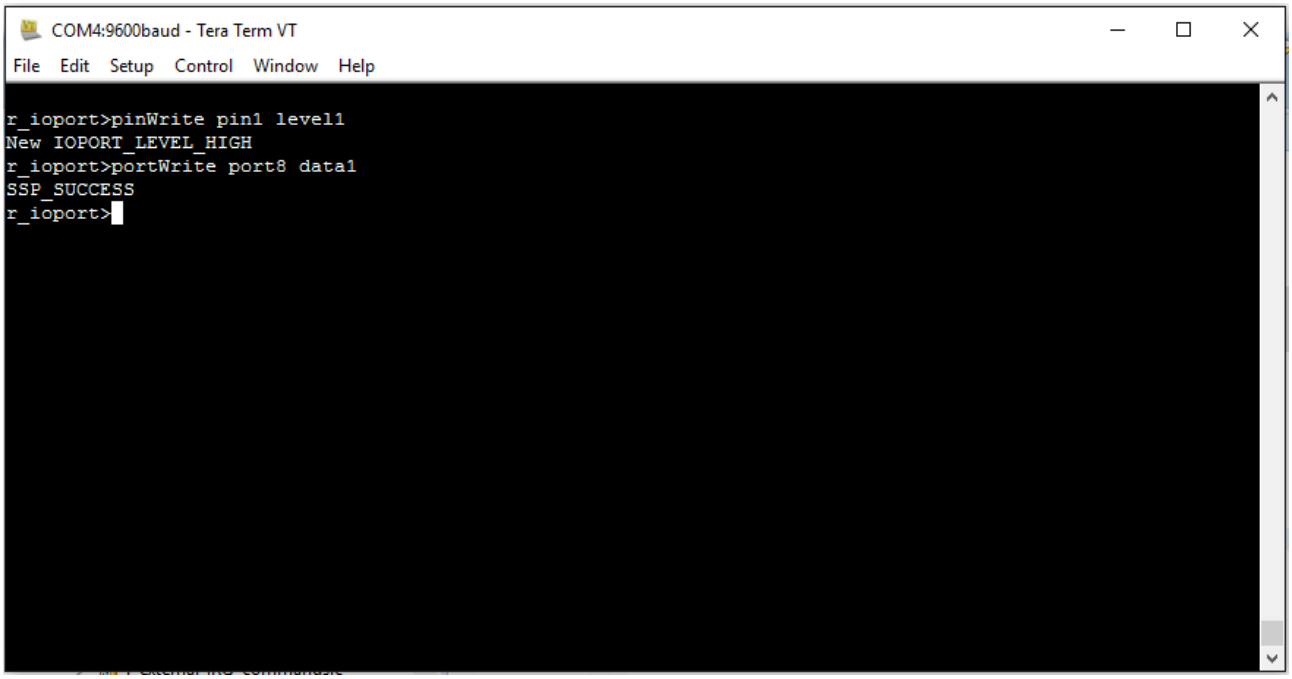
STEP 1: Type `r_ioport` in terminal and press Enter to access the r_ioportsub menu. For help, type ? and press Enter.



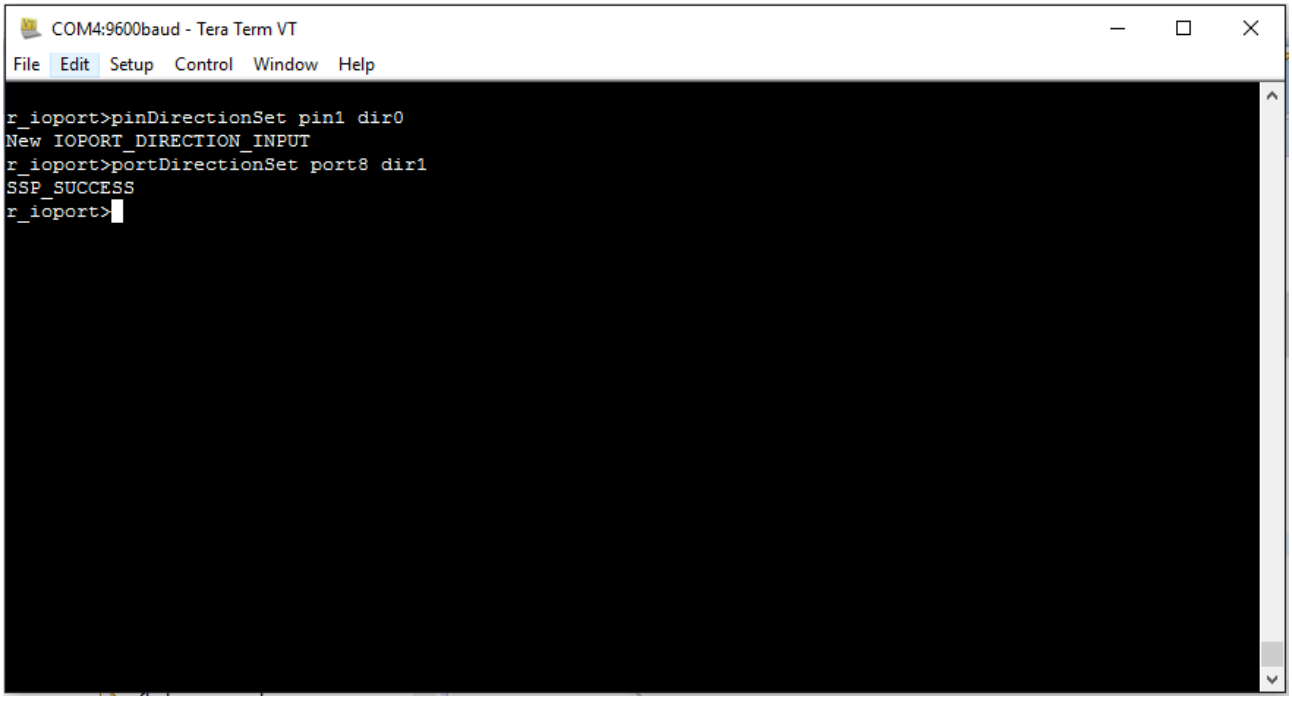
STEP 2: Type `pinRead` or `portRead` command it will display selected pin or port value in terminal.



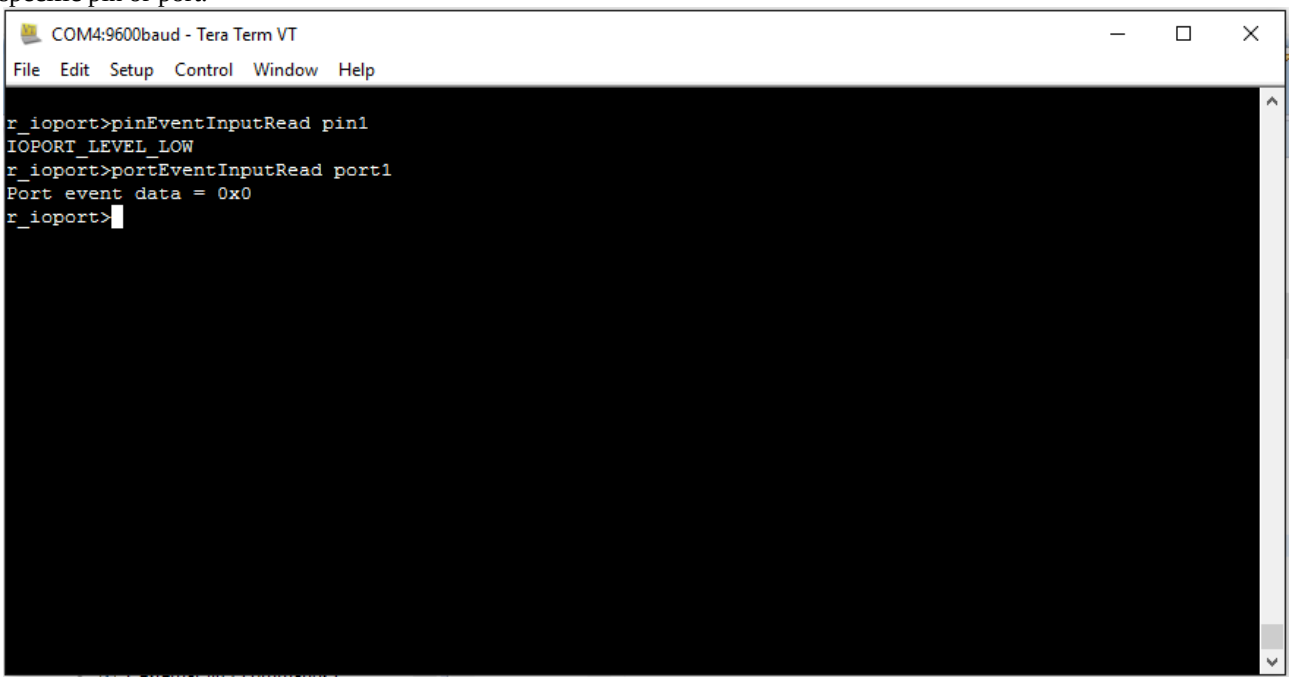
STEP 3: Type `pinWrite` or `portWrite` command to update a value to the specific pin or port.



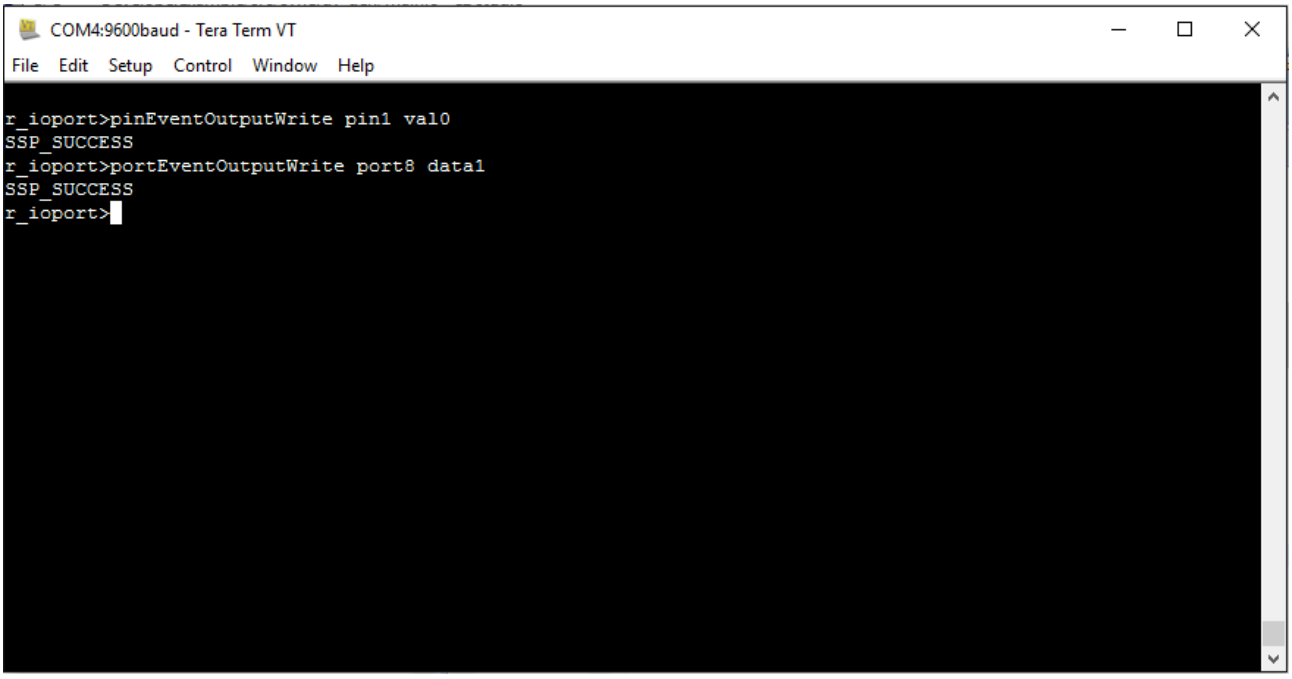
STEP 4: Type `pinDirectionSet` or `portDirectionSet` command to set direction of particular pin or port.



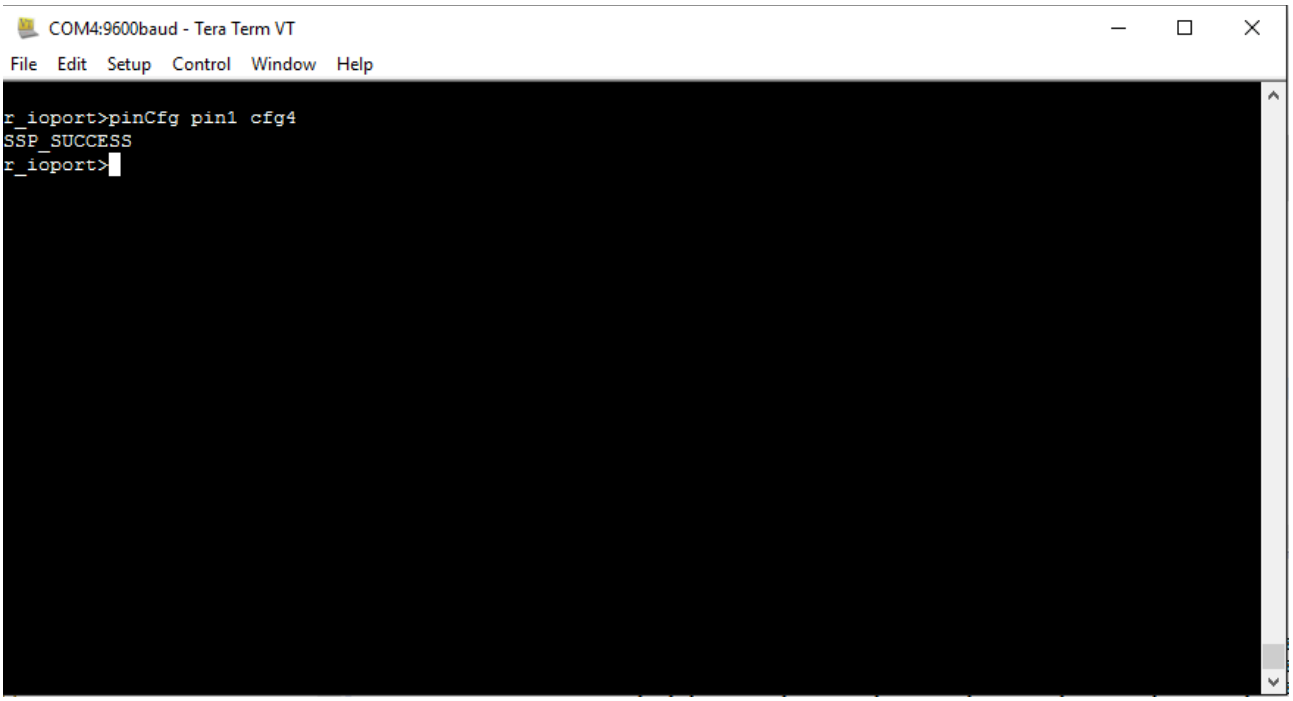
STEP 5: Type `pinEventInputRead` or `portEventInputRead` command to read the event input data of specific pin or port.



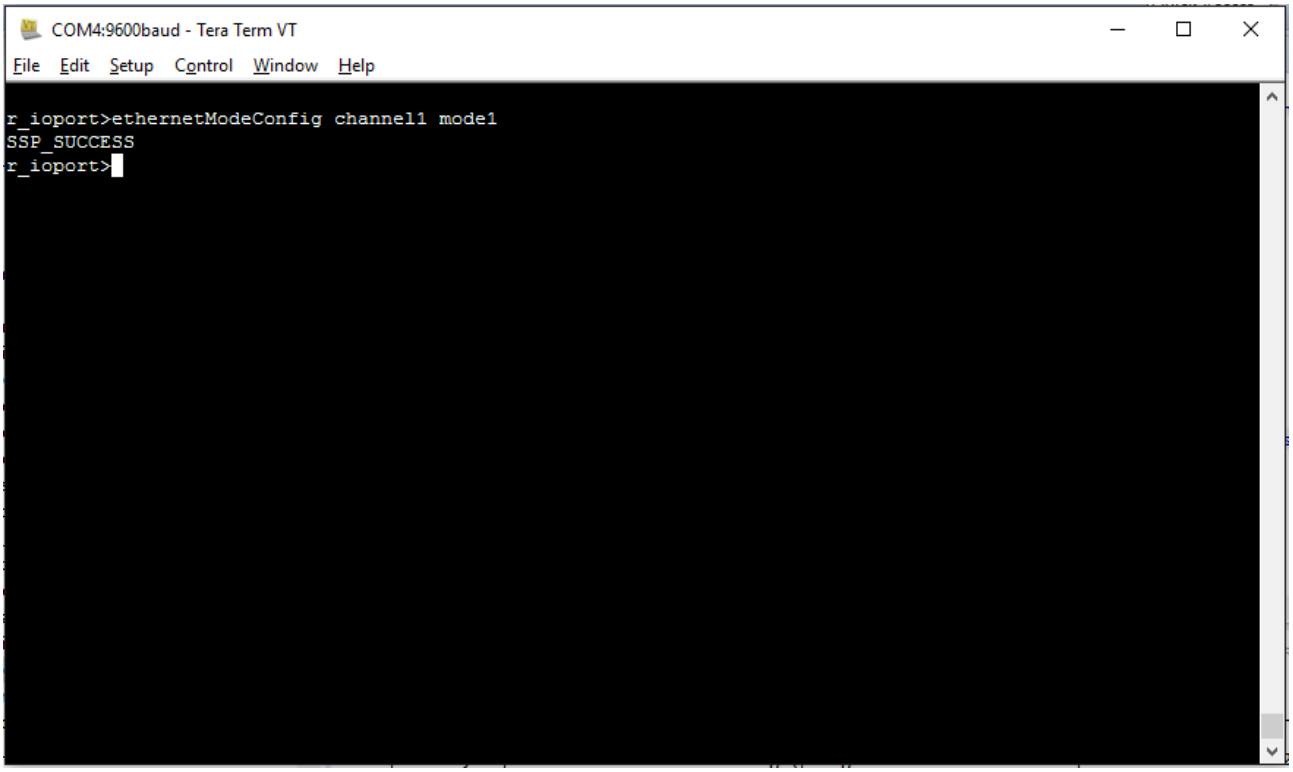
STEP 6: Type `pinEventOutputWrite` or `portEventOutputWrite` command to write the event output data value to a pin or port.



STEP 7: Type `pinCfg` command to configure the setting of a pin.



STEP 8: Type `etherNetModeConfig` command to configure Ethernet channel in PHY mode.



NOTES:

- It is recommended to reset the board before using for any other module.
- The `Init` command will reinitialize all the pins and console will not respond to any input hence not used.

26. Developer Example: Telnet Communications Framework

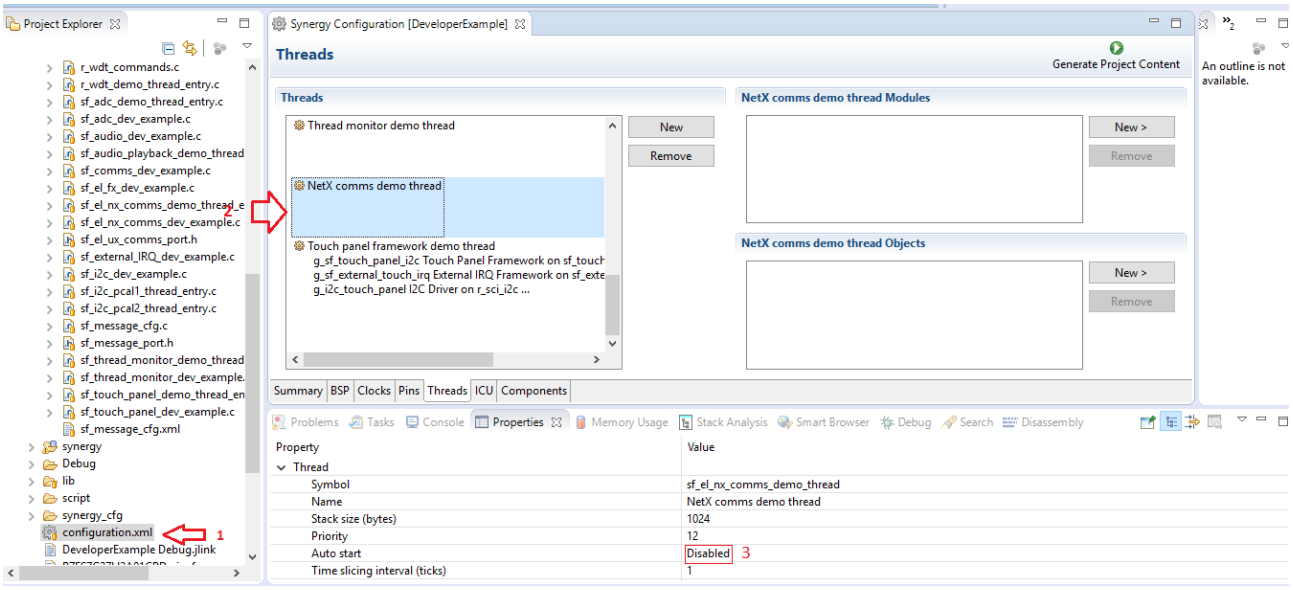
26.1 Introduction

This Developer Example uses the Telnet Communication Framework that uses NetX IPv4 TCP/IP Stack. The purpose This Telnet Communication example will demonstrate an echo server, which echo back the characters typed by you on the console.

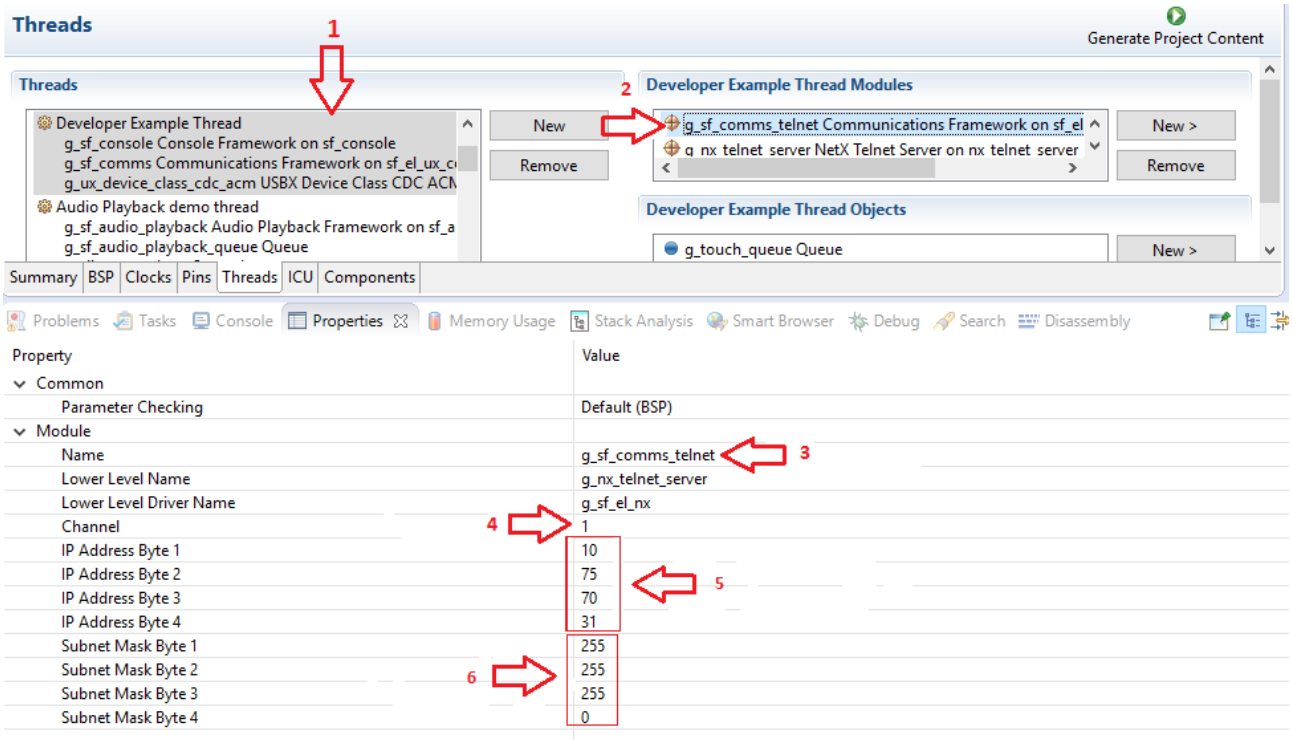
26.2 Telnet Communication Configuration Steps

To create a Telnet application, you have to make some configuration in configuration.xml file as shown below.

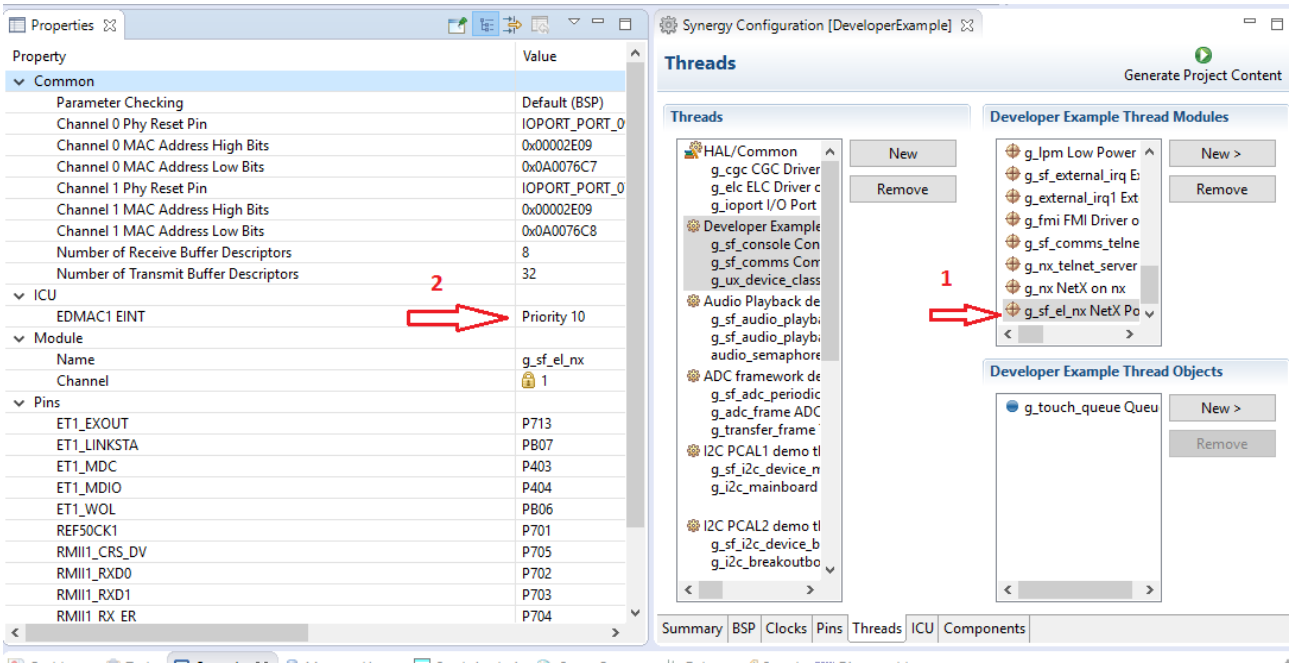
STEP 1: Open configuration.xml file, NetX comms demo thread and disable auto start.



STEP 2: In Developer Example thread, g_sf_comms_telnet_Communications Framework instance. Change the following fields.



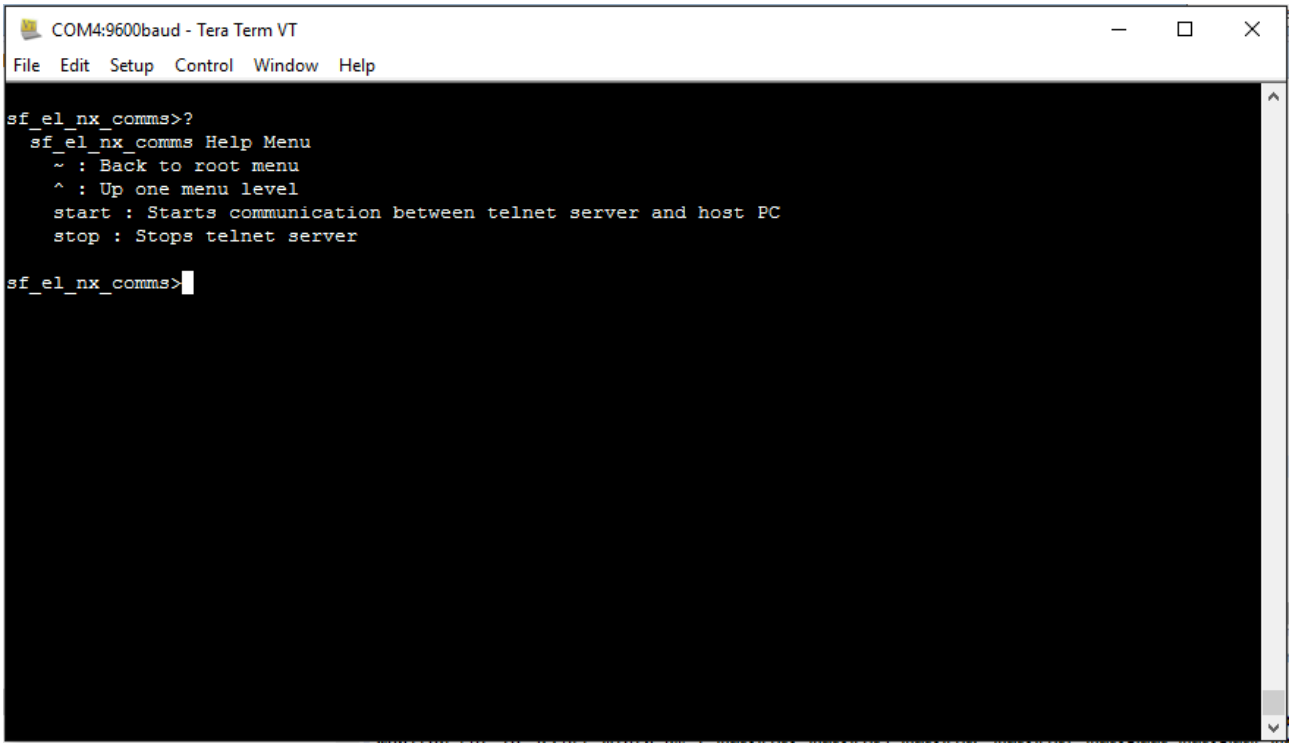
STEP 3: In the Developer Example thread, g_sf_el_nx Framework instance, enable the EDMAC1 EINT as shown and give a priority (say 10).



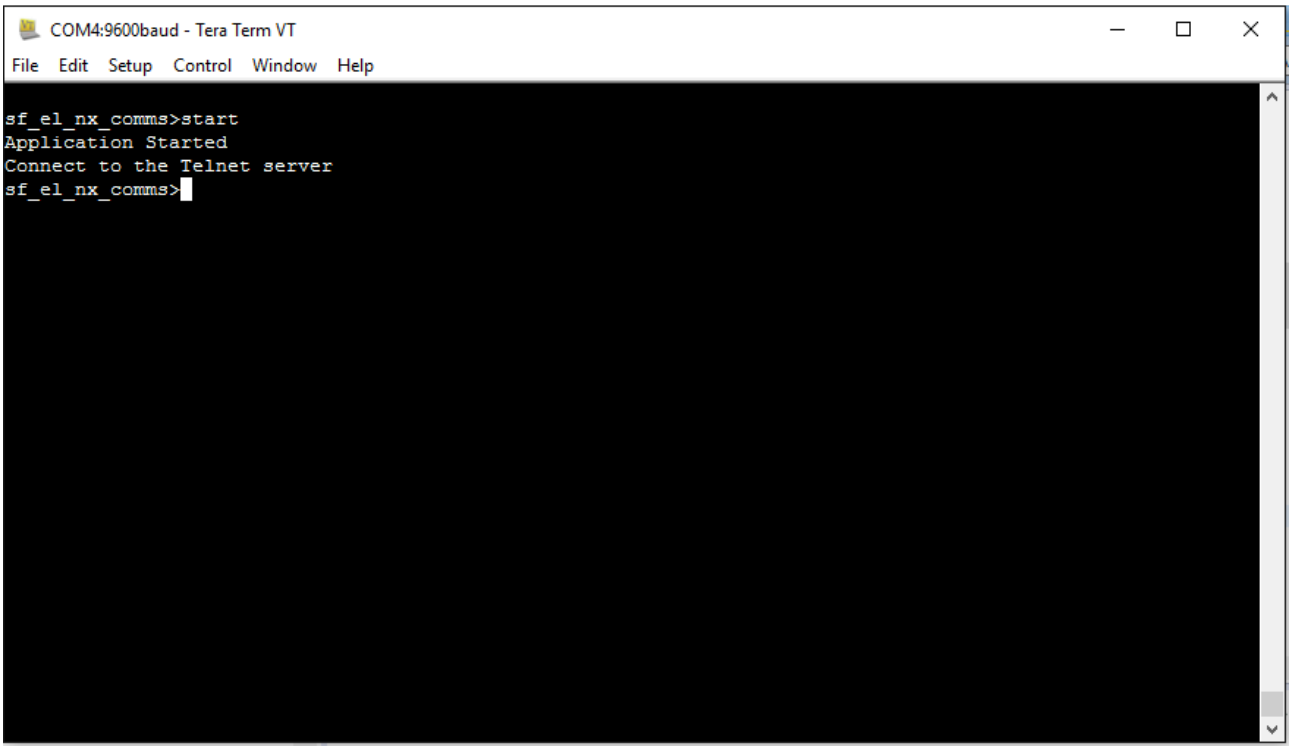
26.3 Run the Communications Framework application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

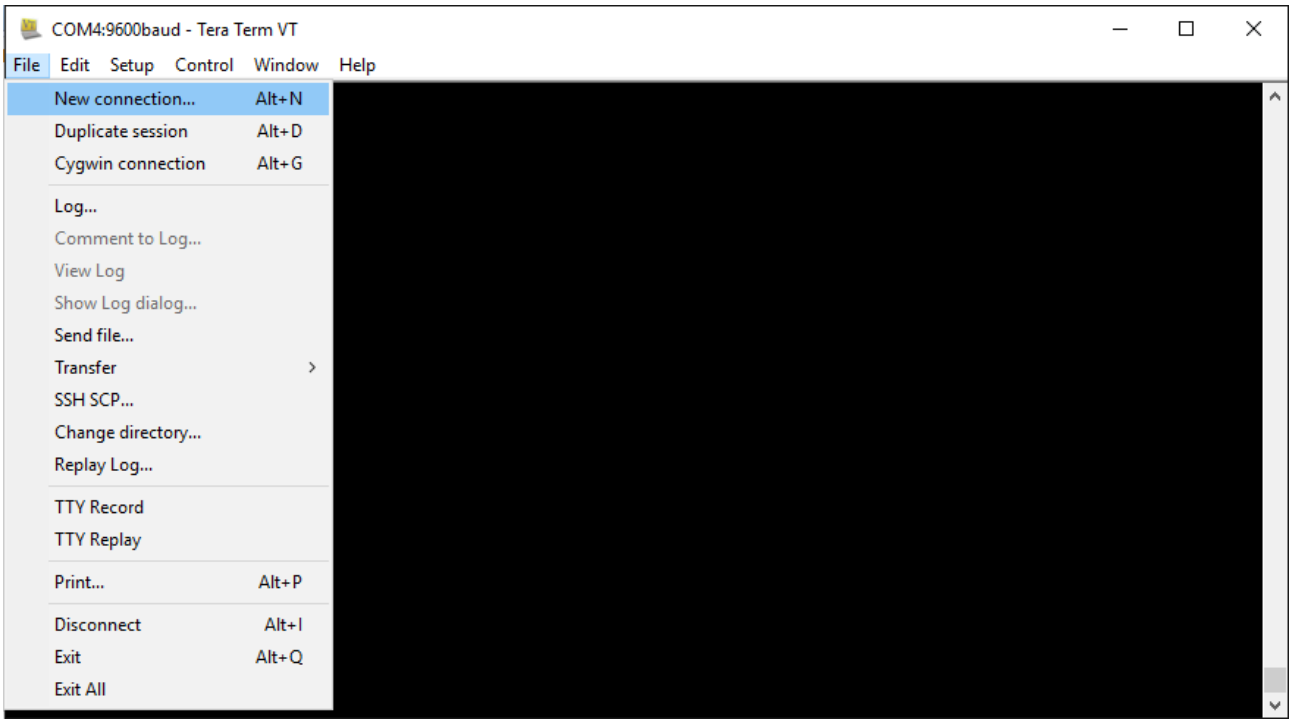
STEP 1: Type `sf_el_nx_comms` in the terminal and press Enter to access the `sf_el_nx_comms` sub menu. For Help, type ? and press Enter.



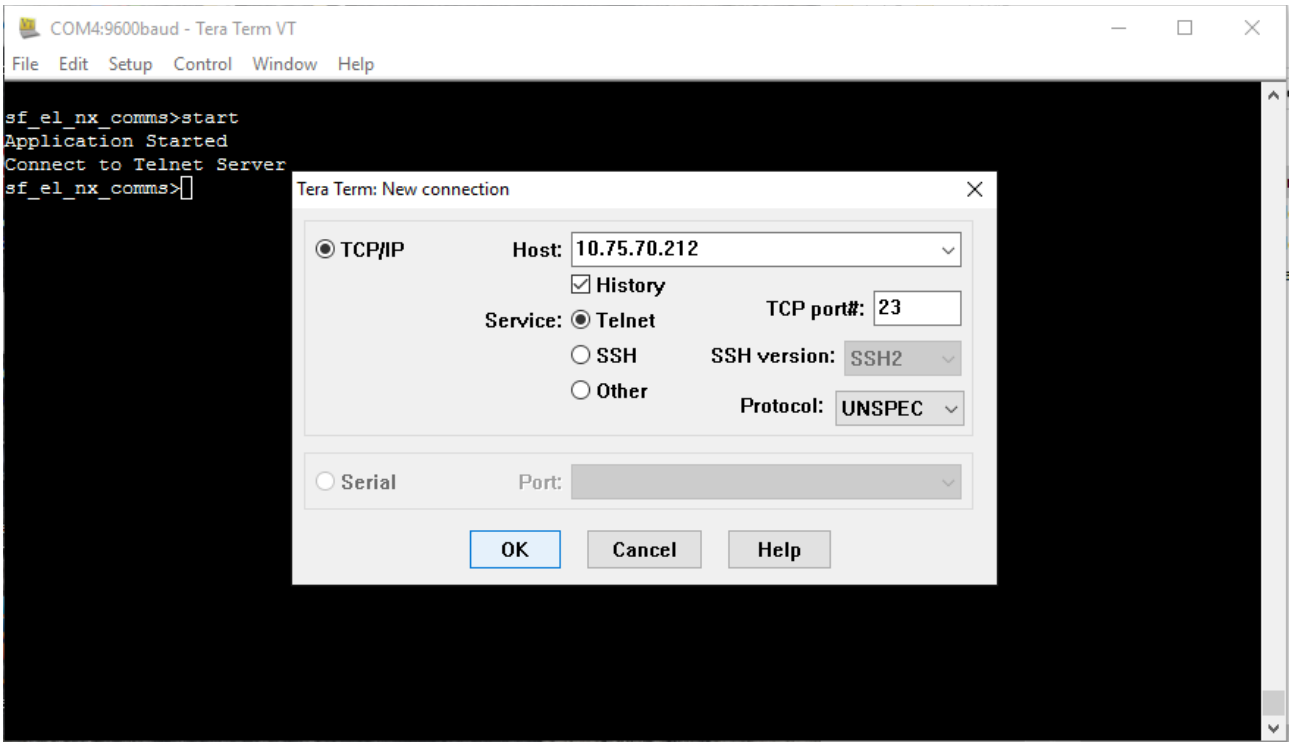
STEP 2: Use `start` command to start the application.



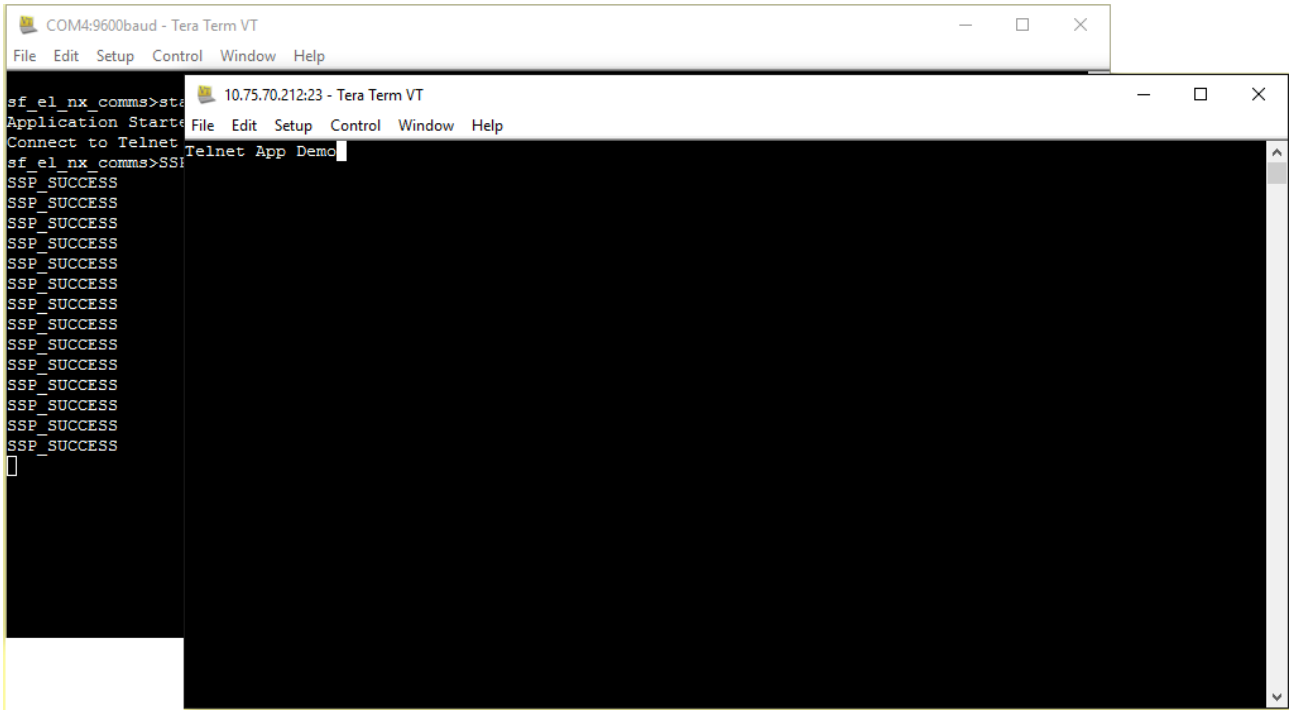
STEP 3: In Tera Term, navigate to **File**>**New connection.**



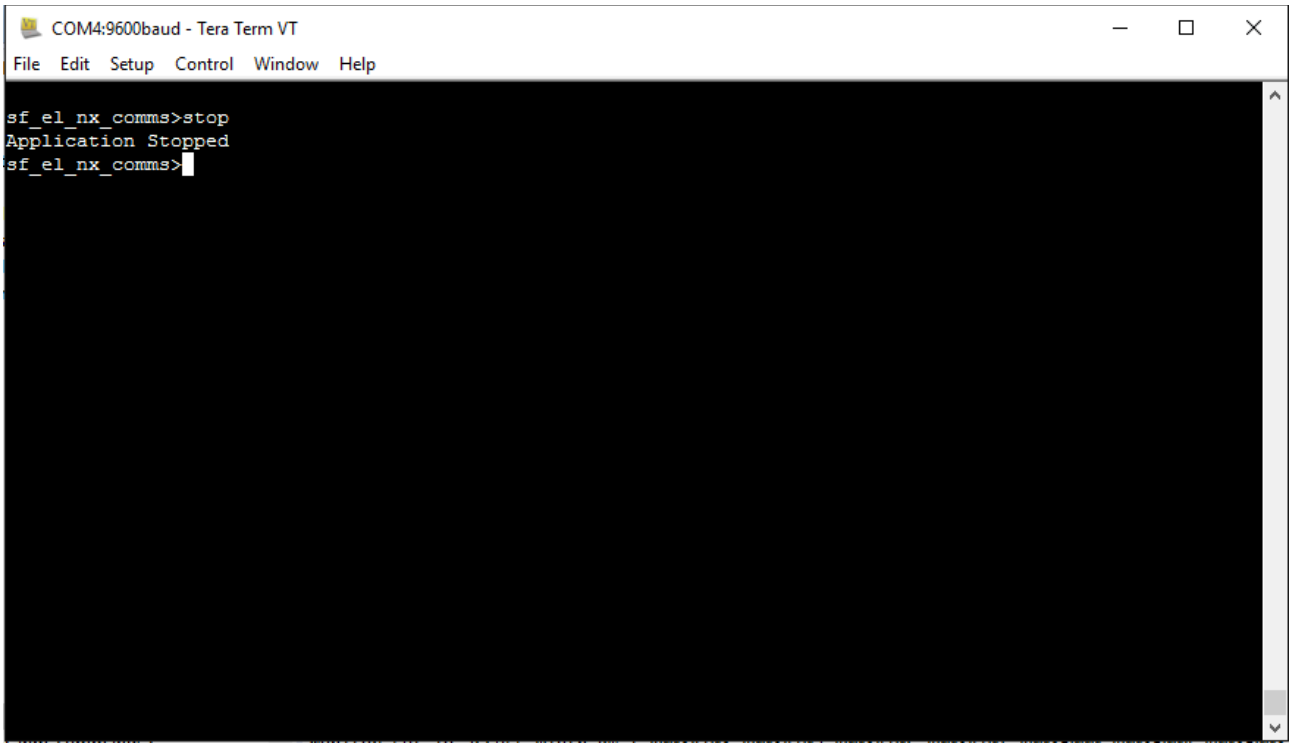
STEP 4: Select Telnet and give the IP address which is assigned to the device by configuration.xml file in the Host field and click OK.



STEP 5: A new window opens which work as an 'Echo Server'. The TELNET console will echo back given inputs, in this case Telnet App Demo is written to console which is echoed back and visible in terminal



STEP 6: Type `stop` in serial console to stop the application.



NOTES:

- It is preferable to give static IP address to the server.
- The given IP address to the device should match the default Gateway address and domain name of the system.

27. Developer Example: Power Profiles Framework

27.1 Introduction

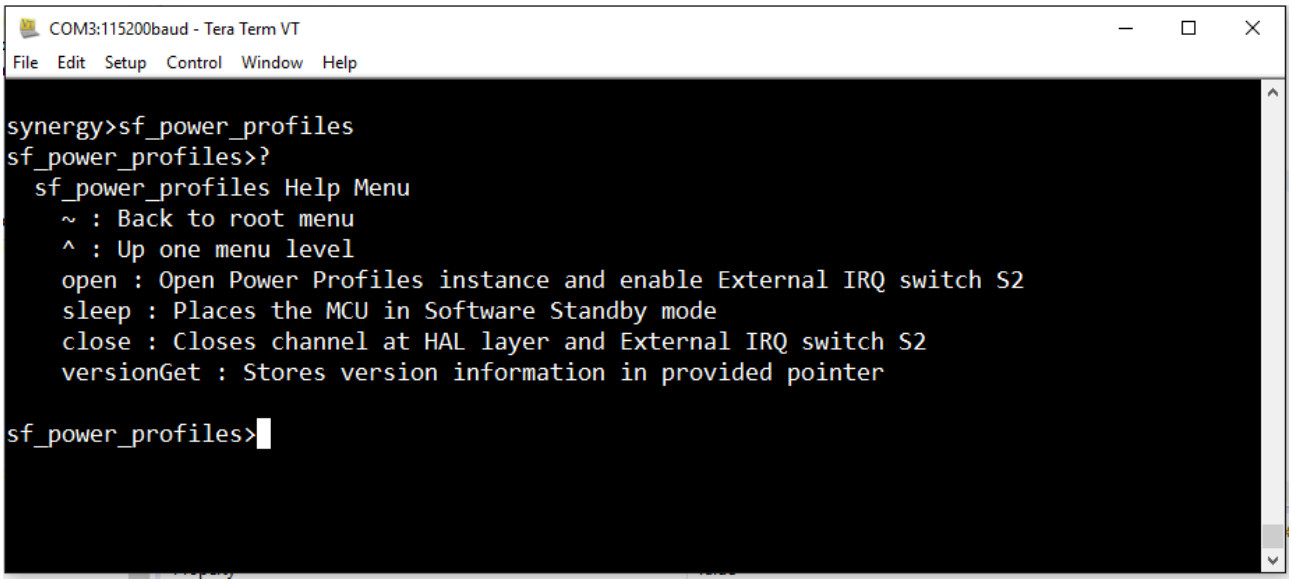
The power profiles framework supports run, RTC and external mode of operation. The Developer Example uses external mode, where switch S2 is configured and used as an external source to wake device from sleep. When sleep command is executed red LED1 glows indicating device has been put to software standby mode. Once switch S2 is pressed device wakes up and this is indicated by turning LED1 from red to green.

27.2 Run the Power Profiles Framework application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the Power Profiles Framework driver application, follow these steps:

STEP 1: Type `sf_power_profiles` in the terminal and press Enter to access the Power Profiles Framework submenu. For Help, type '?' and press Enter.



STEP 2: Use the `open` and then `sleep` command to configure and enter the device into software standby mode. Red LED1 glows will indicate device entered software standby mode. Press switch S2 to wake up the device, LED1 turns from red to green indicating the state change. This is indicated by turning LED1 from red to green.

28. Developer Example: SCI SPI HAL Driver

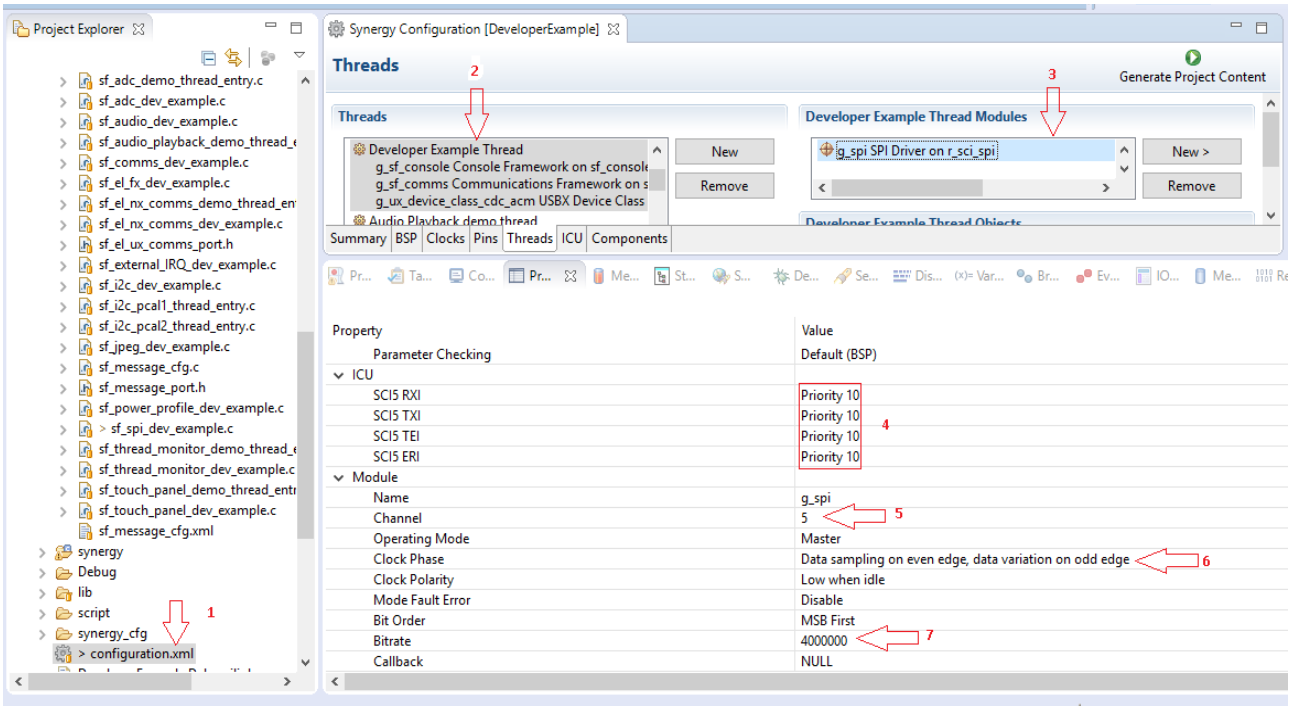
28.1 Introduction

This Developer Example exercises the SCI SPI HAL APIs using the on board Bluetooth Low Energy (BLE) device. The BLE device is connected to the SCI Channel 5.

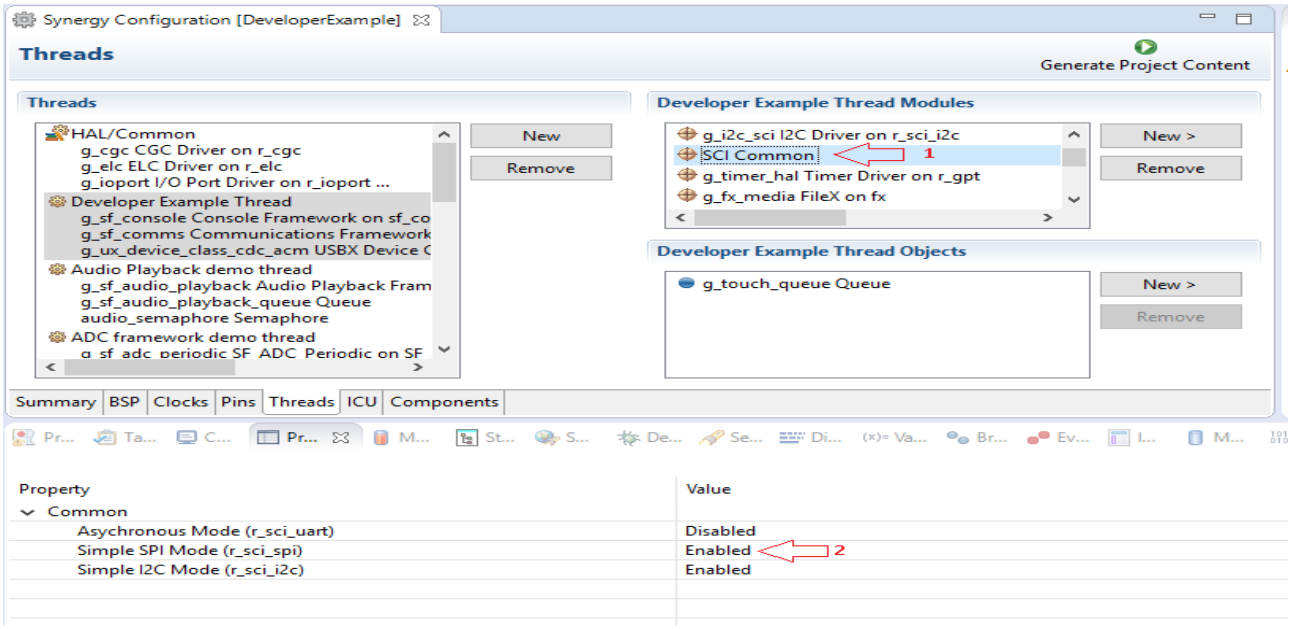
28.2 SCI SPI HAL driver configuration steps

To exercise a SPI HAL driver you need to make some configuration in configuration.xml file as mentioned below.

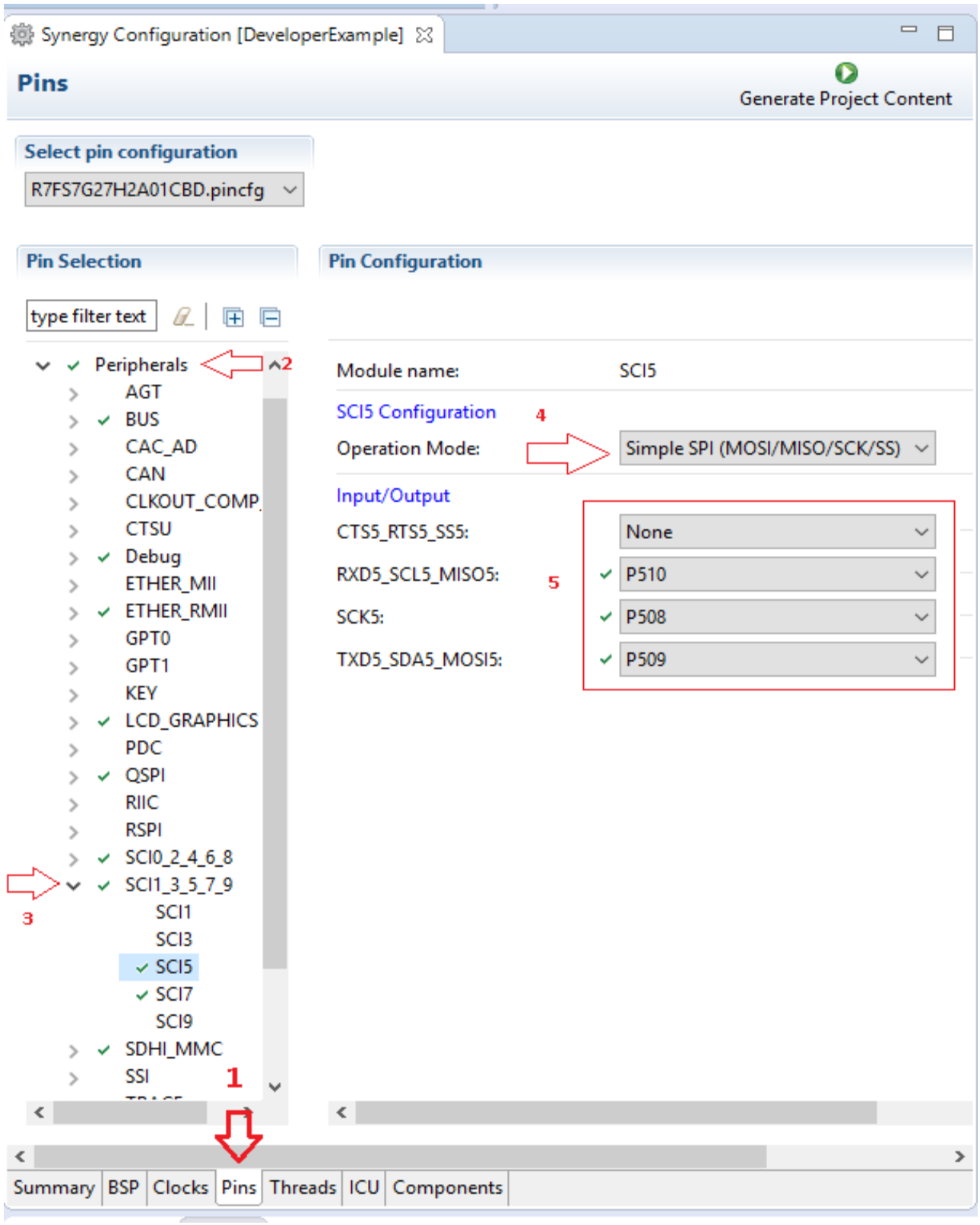
STEP 1: Open configuration.xml file, Developer Example Thread, add r_sci_spi driver and change the properties as shown below.



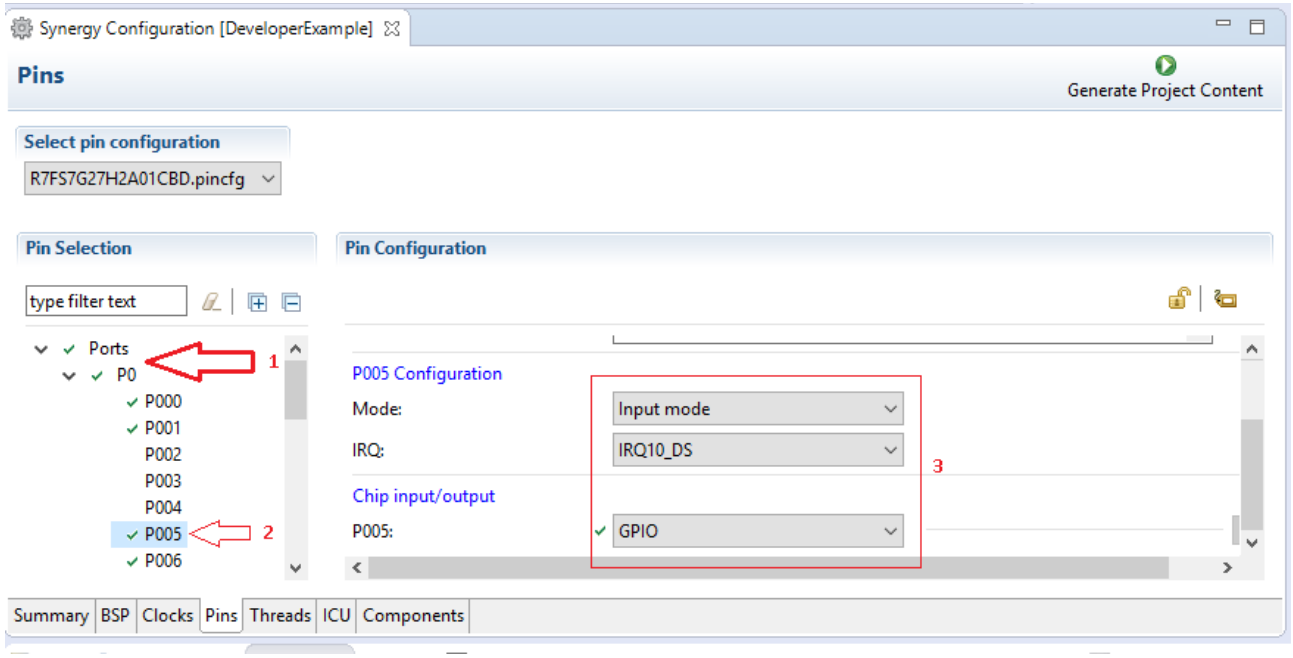
STEP 2: Next in the same Developer Example thread, SCI common, properties enable Simple SPI mode.



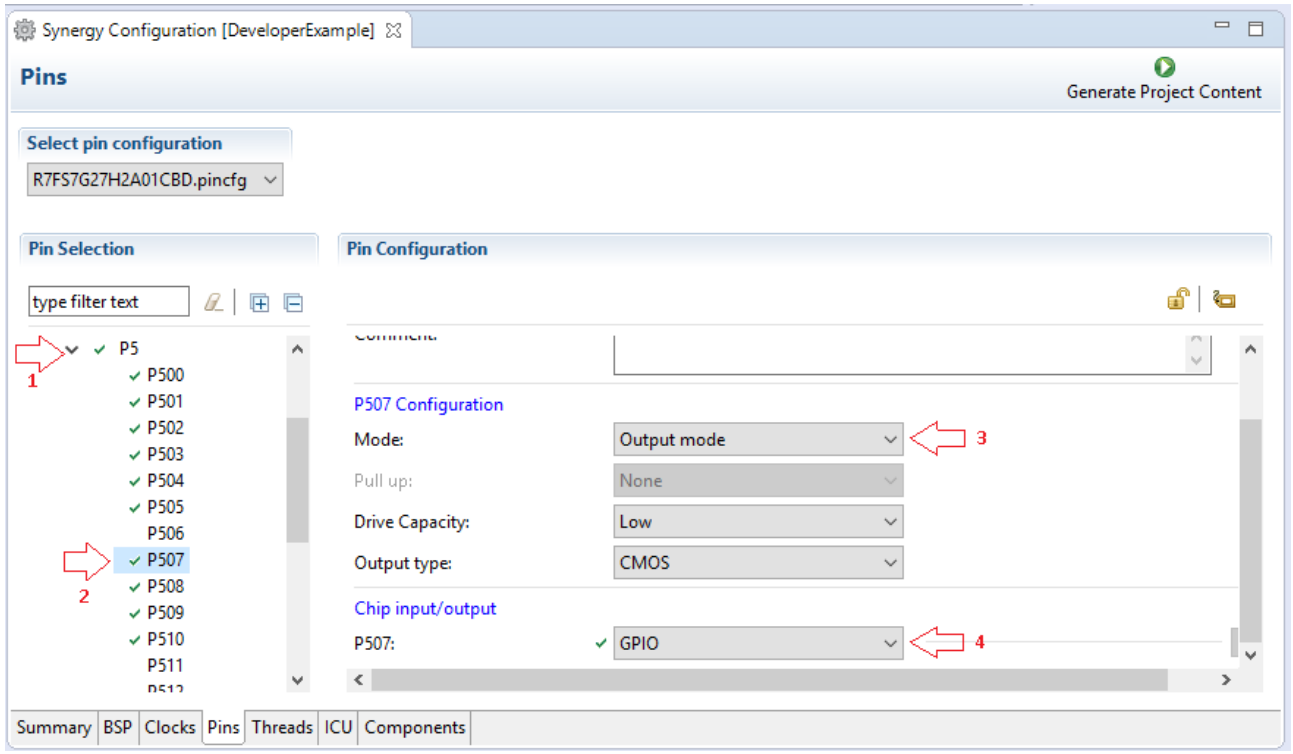
STEP 3: Next in the Pins tab, complete the following steps as shown below.



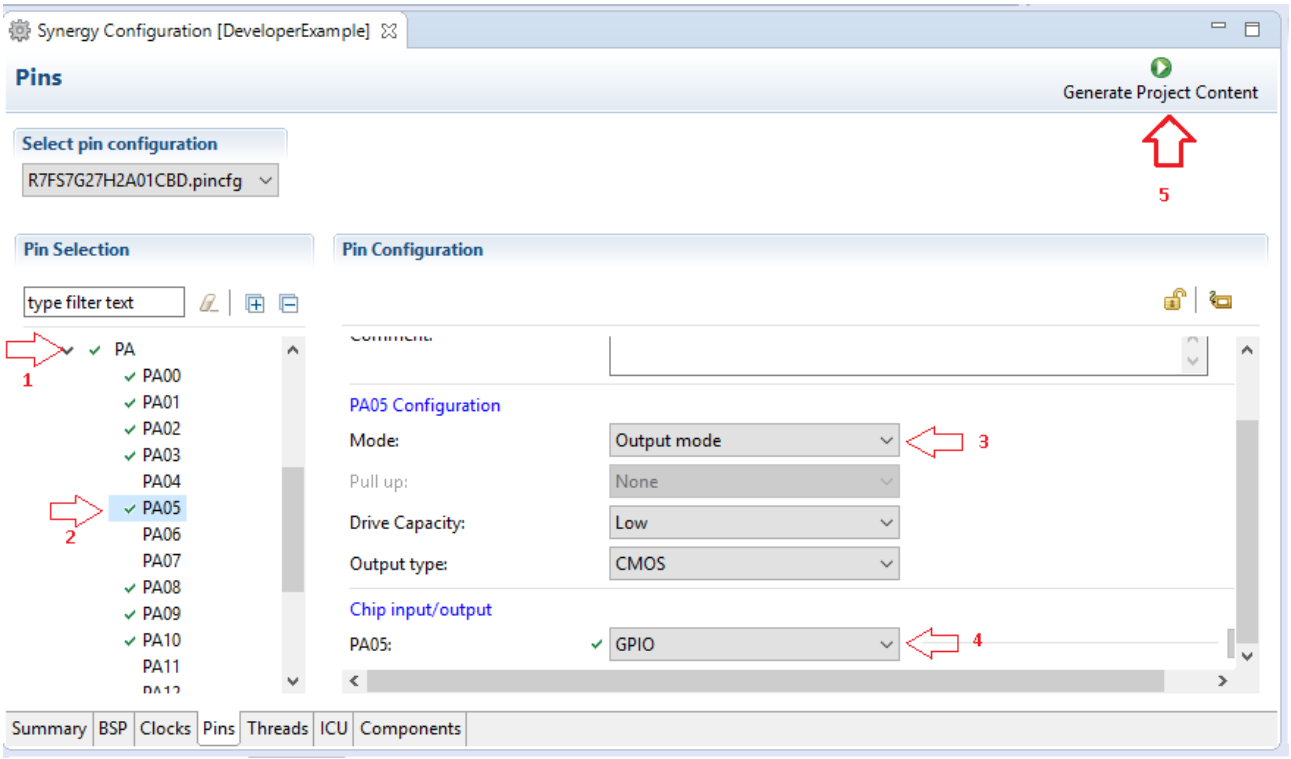
STEP 4: Next for setting the pin direction, complete the following as shown below.



STEP 5: Do the same for Port_5 Pin_7 and Port_A Pin_5.



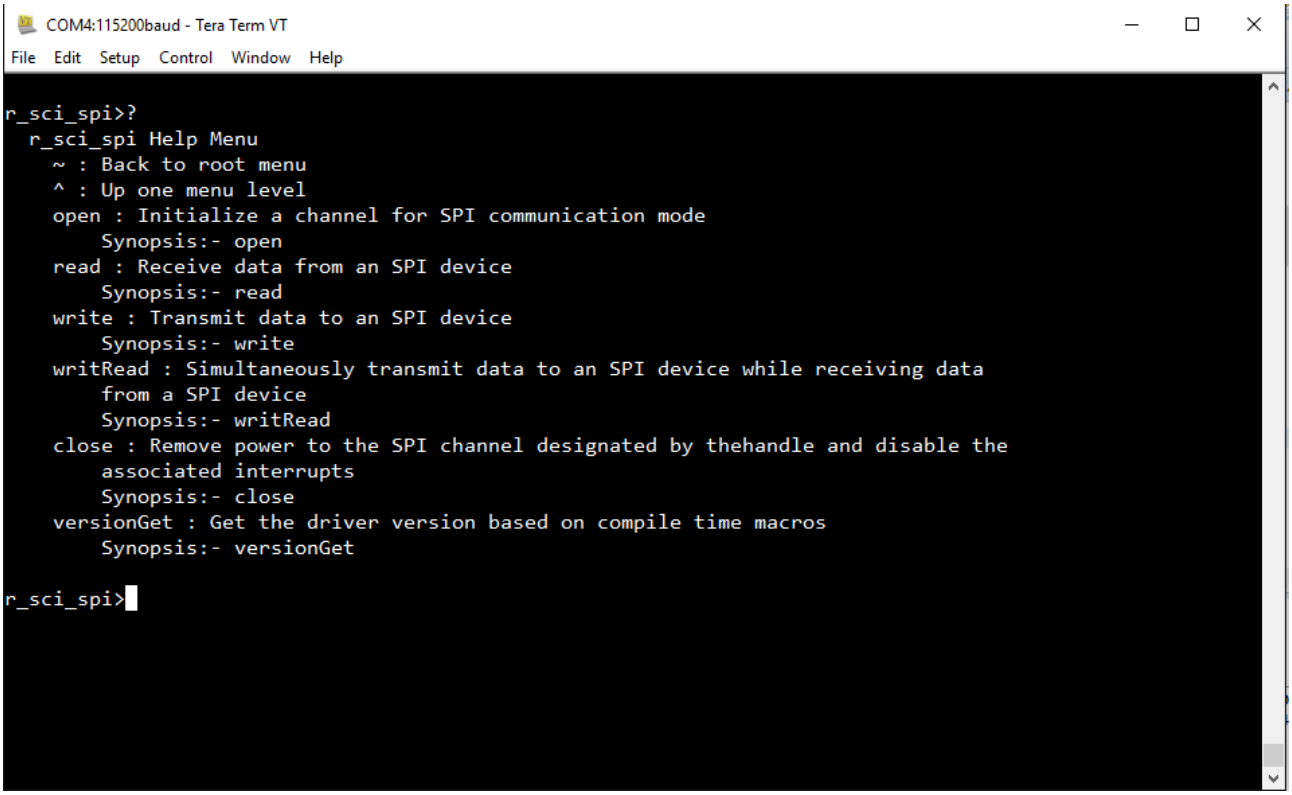
STEP 7: Save the configuration.xml file and click Generate Project Content as shown below.



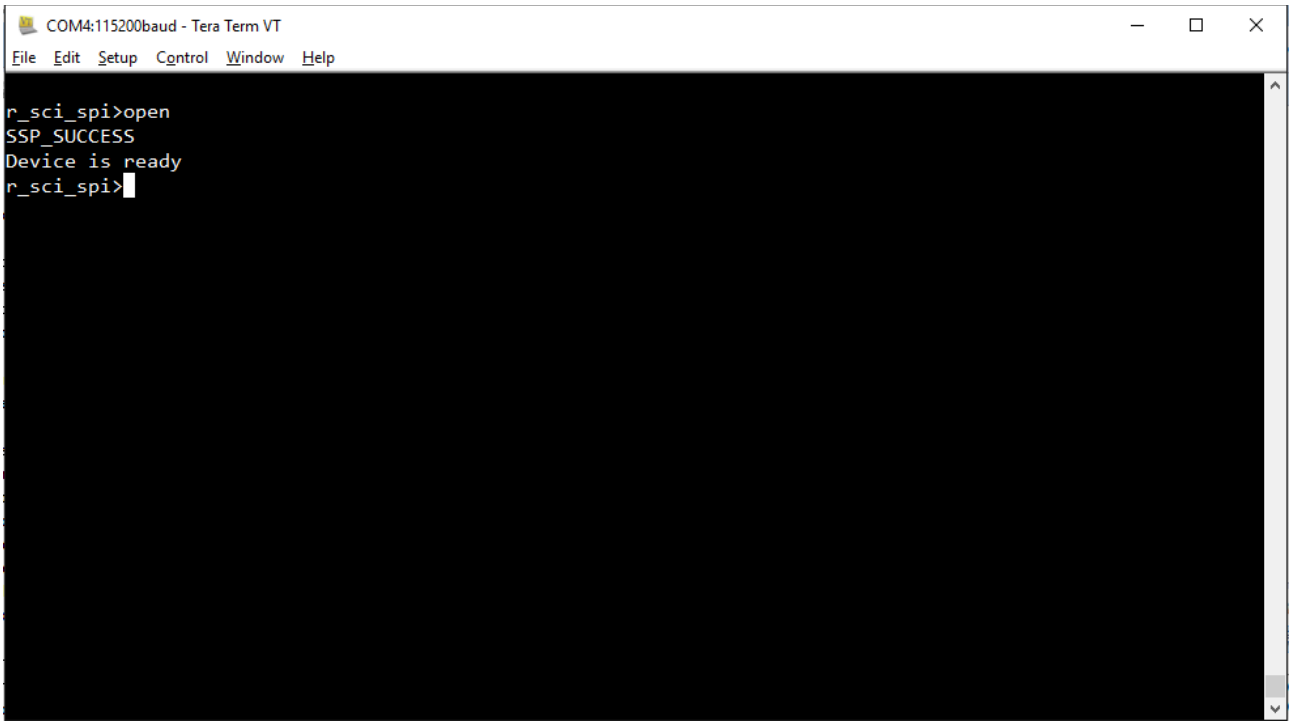
28.3 Run the SCI SPI HAL Driver application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

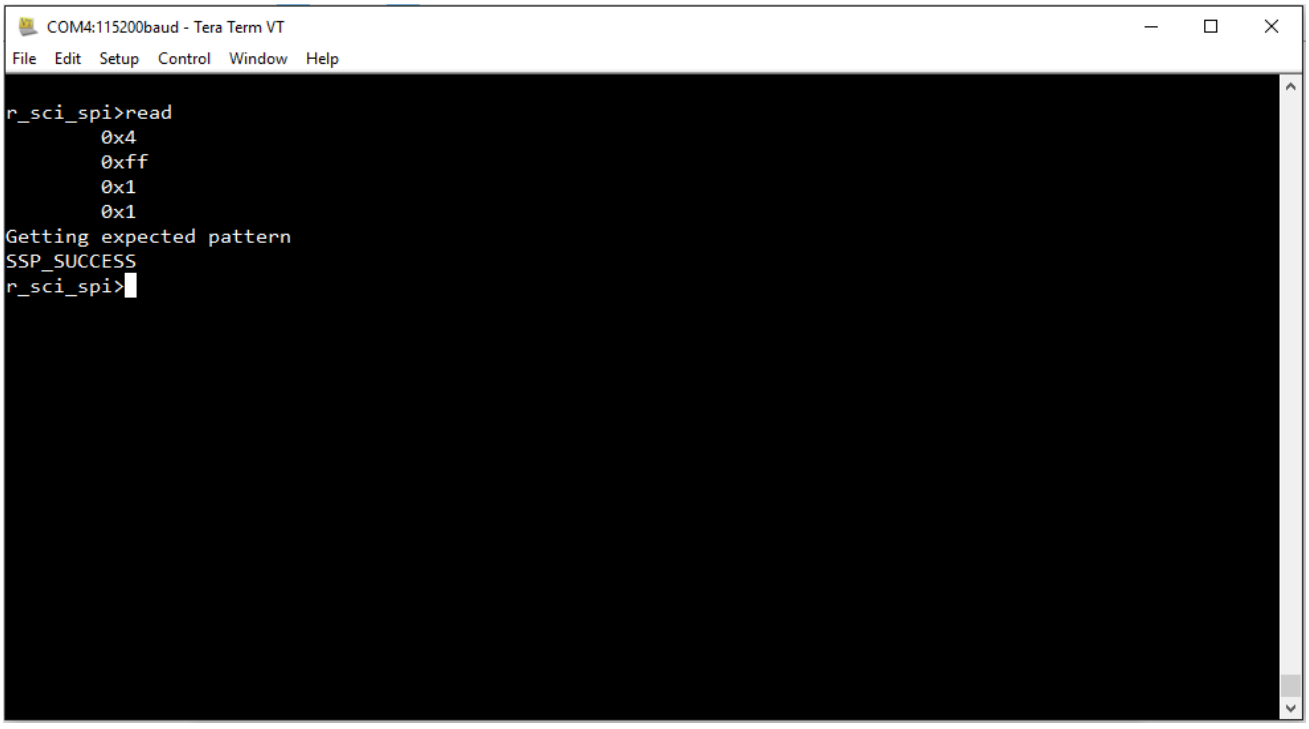
STEP 1: Type `r_sci_spi` in the terminal and press Enter to access the `r_sci_spi` sub menu. For help, type ? and press Enter.



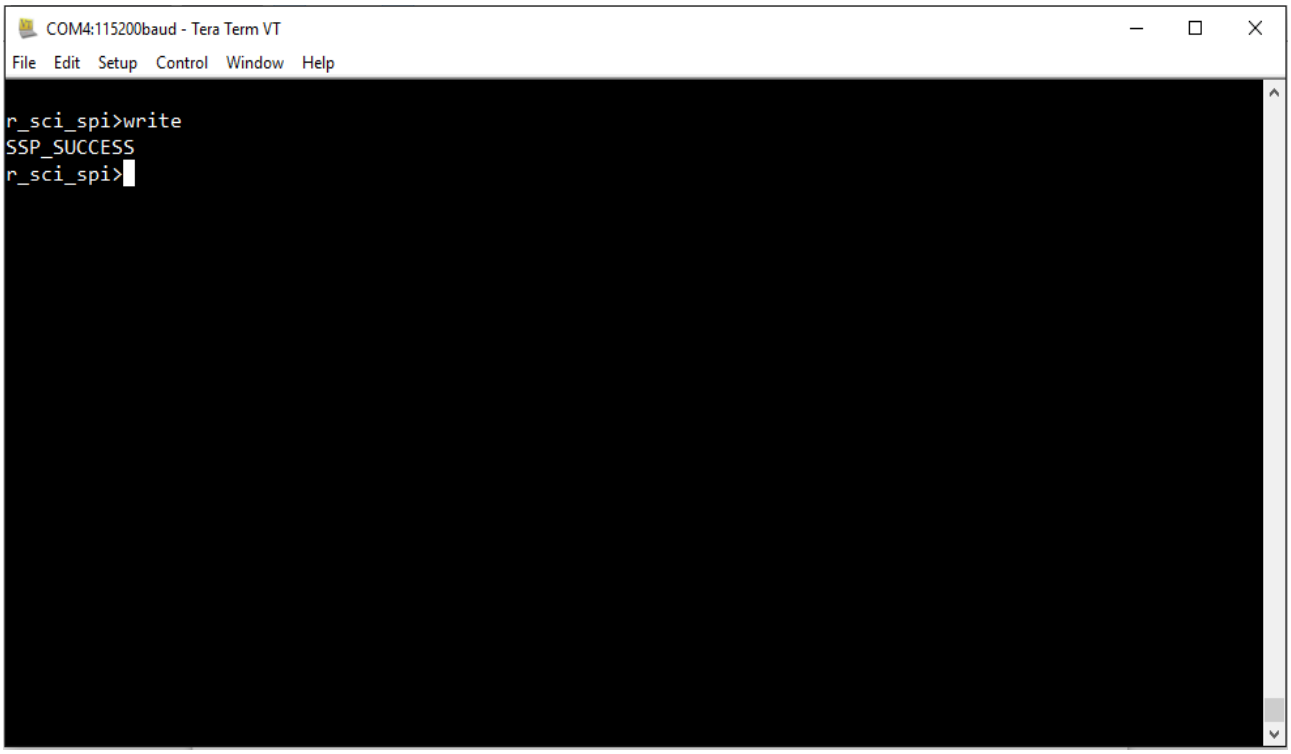
STEP 2: Use `open` command to initialize the driver.



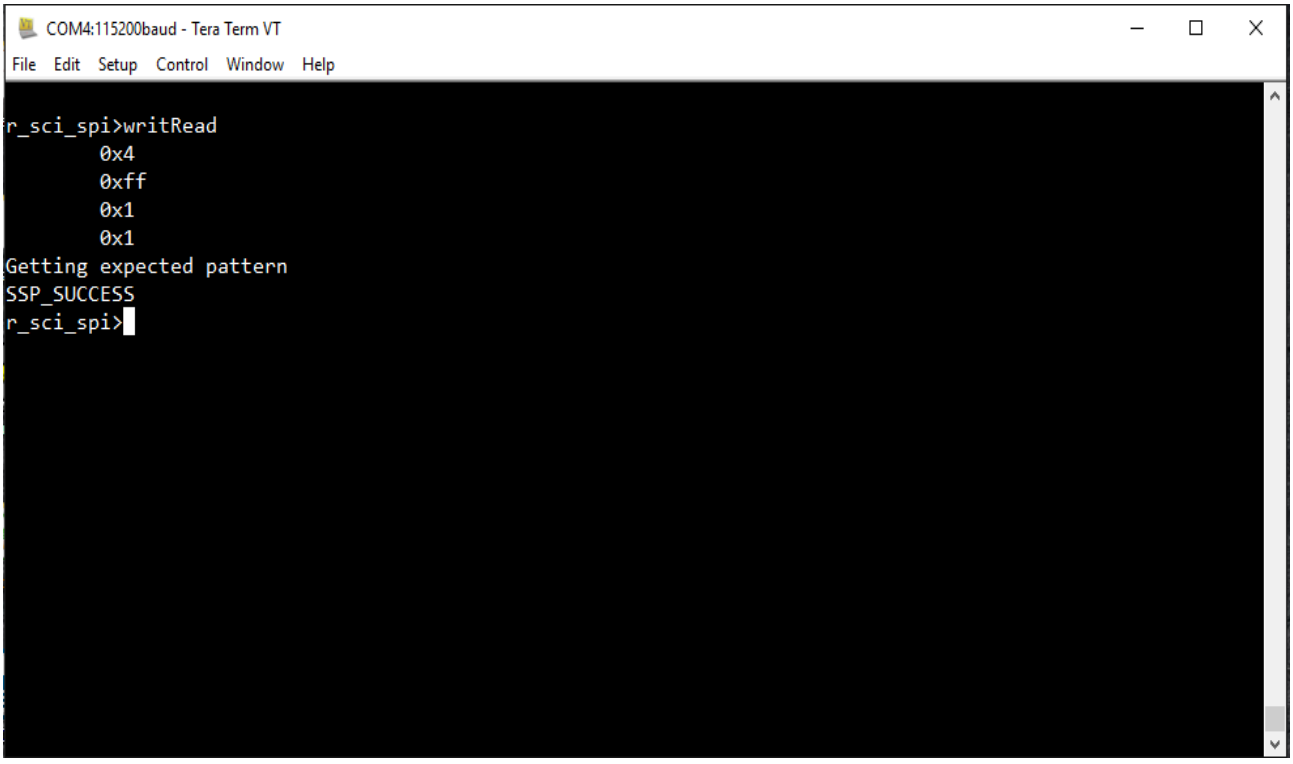
STEP 3: Type `read` command to read value from the slave device.



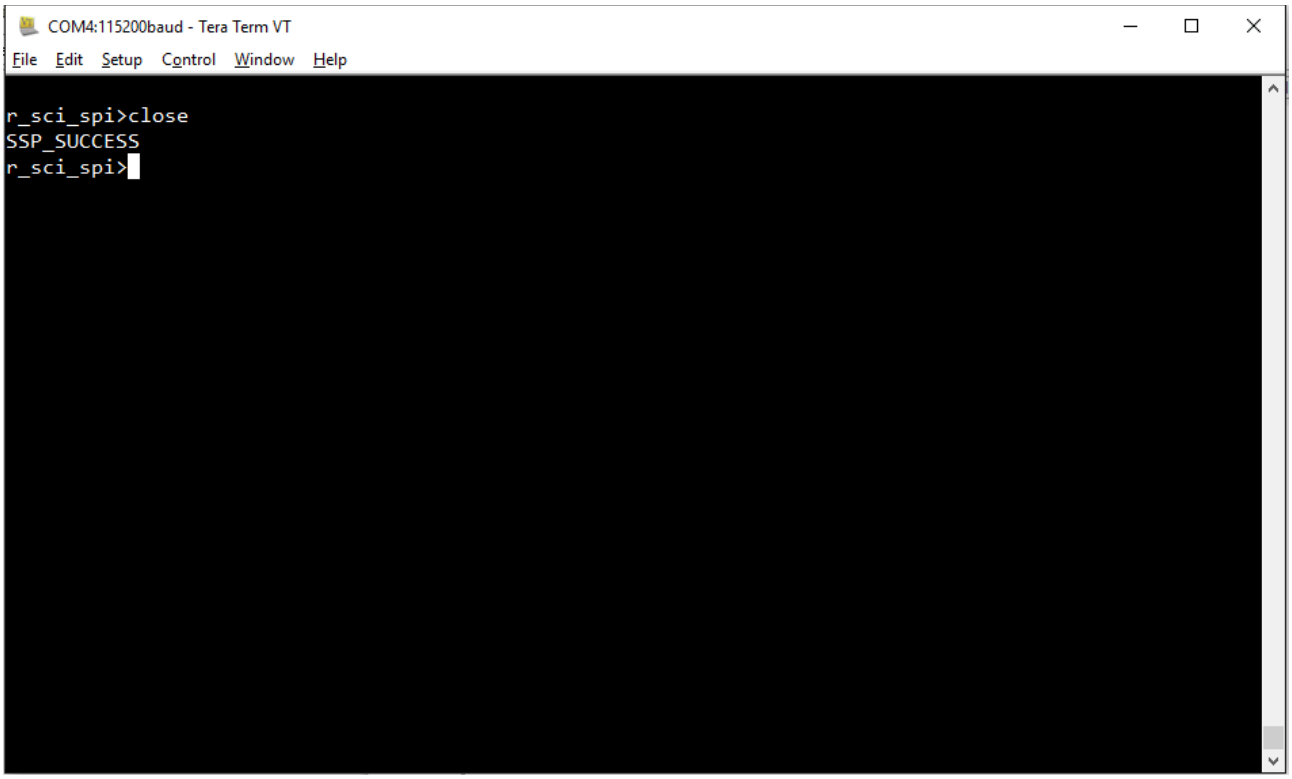
STEP 4: Type `write` command for writing to the slave device.



STEP 5: Type the writRead command to simultaneously write and read to the slave device.



STEP 6: Type `close` command to close the driver.



NOTES:

1. In Developer Example r_sci_spi the `writeRead` command is written as `writRead` command this is due to the limitation of the console framework.
2. The configuration has to be done correctly else you will not get any response from the BLE.
3. For details of BLE see EM9301 datasheet.

29. Developer Example: SPI Framework

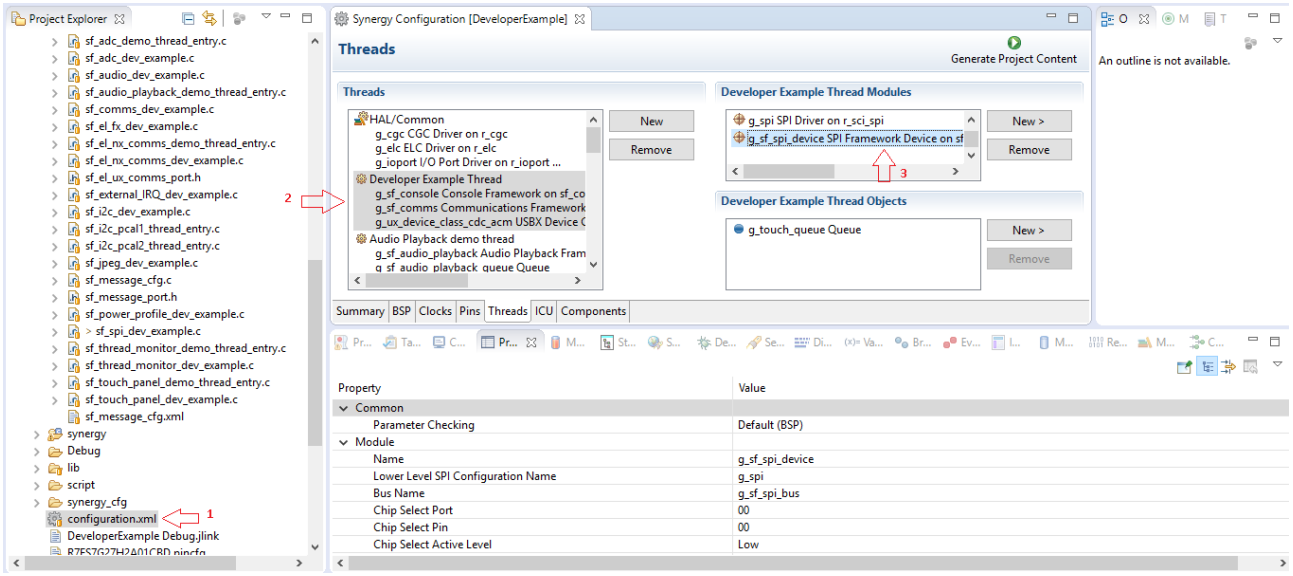
29.1 Introduction

SPI framework provides a thread safe mechanism to communicate between master and multiple slaves on the same SPI channel. Since the DK-S7G2 board has only one SPI slave device, this Developer Example for SPI Framework will just exercise the SPI Framework API's on the slave device.

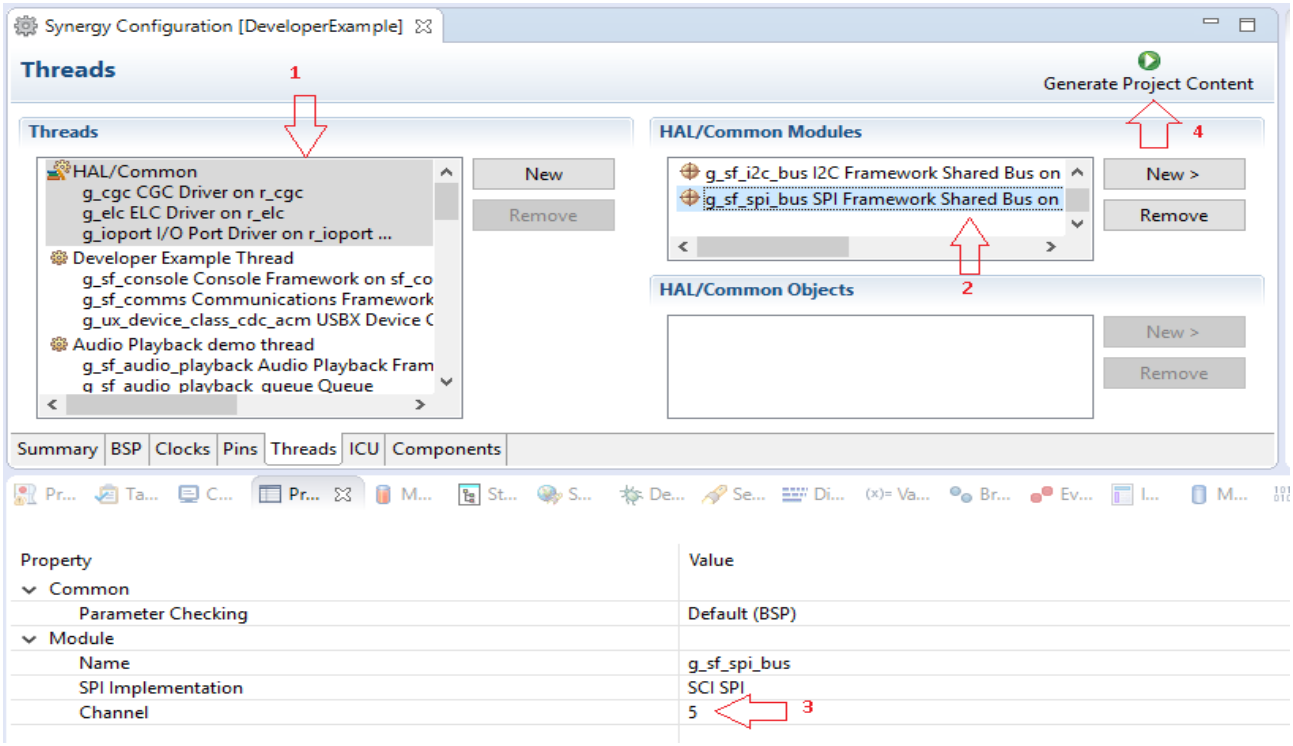
To configure the SPI Framework, you have to configure SCI SPI HAL driver first. See.SCI SPI HAL driver configuration steps follow the steps detailed below to configure the SPI Framework.

29.2 SPI Framework configuration steps

STEP 1: Add the SPI driver in Developer Example Thread.



STEP 2: SPI Framework requires a SPI shared bus. Change the channel no in spi shared bus properties as shown below. Generate the Project Content.

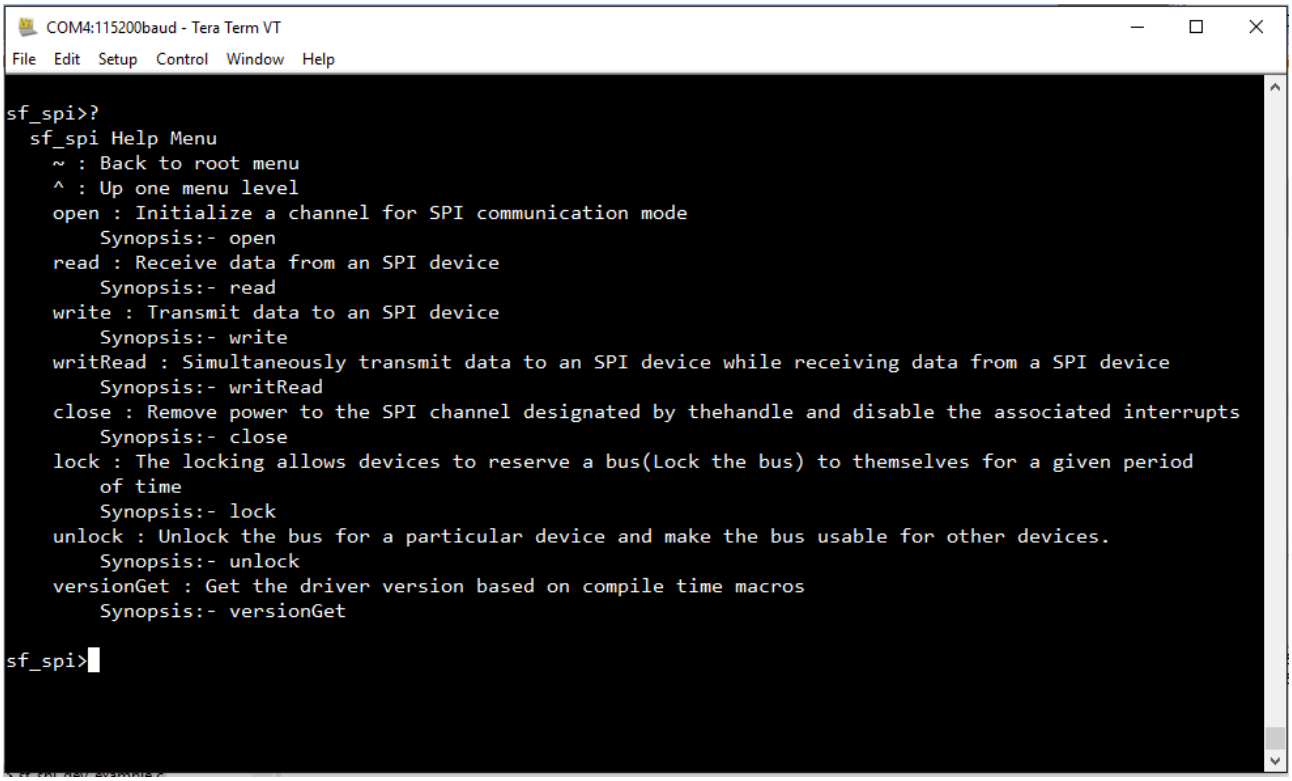


29.3 Run the SPI Framework application

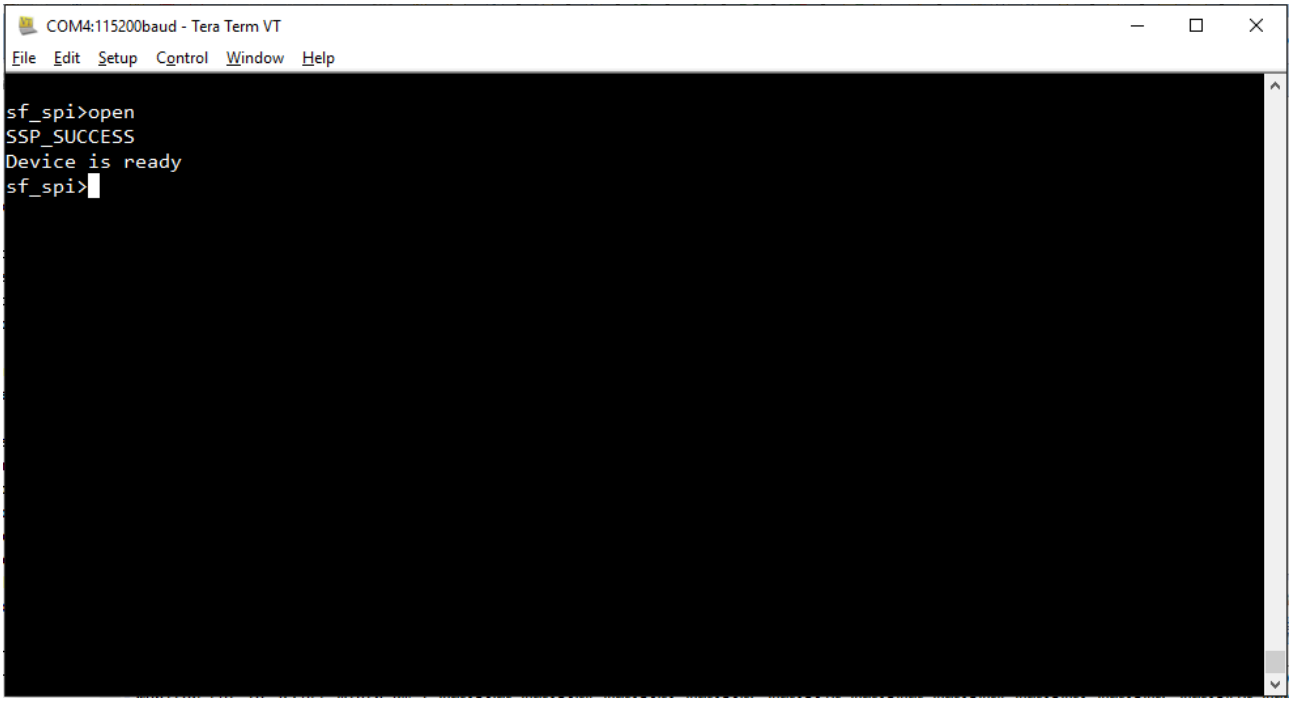
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

To run the SPI Framework application, follow these steps:

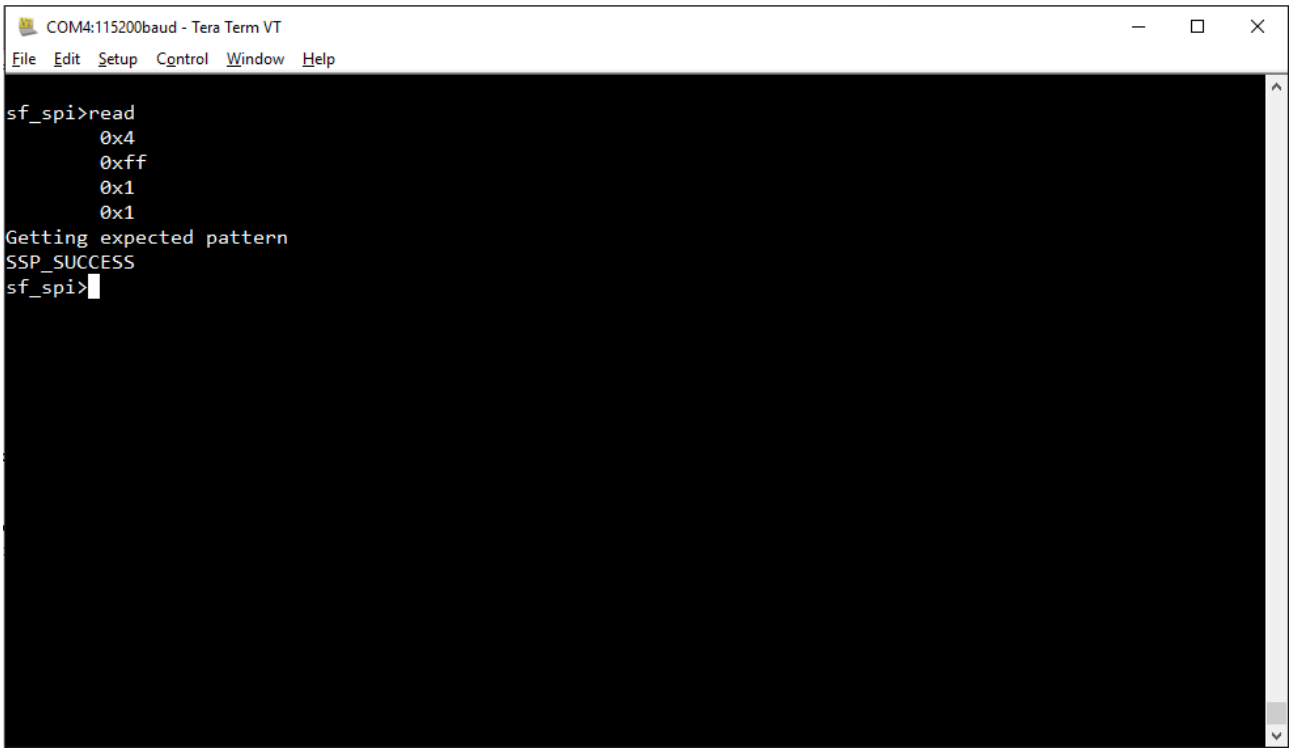
STEP 1: Type `sf_spi` in the terminal and press Enter to access the sf_`spi` sub menu. For help, type '?' and press Enter.



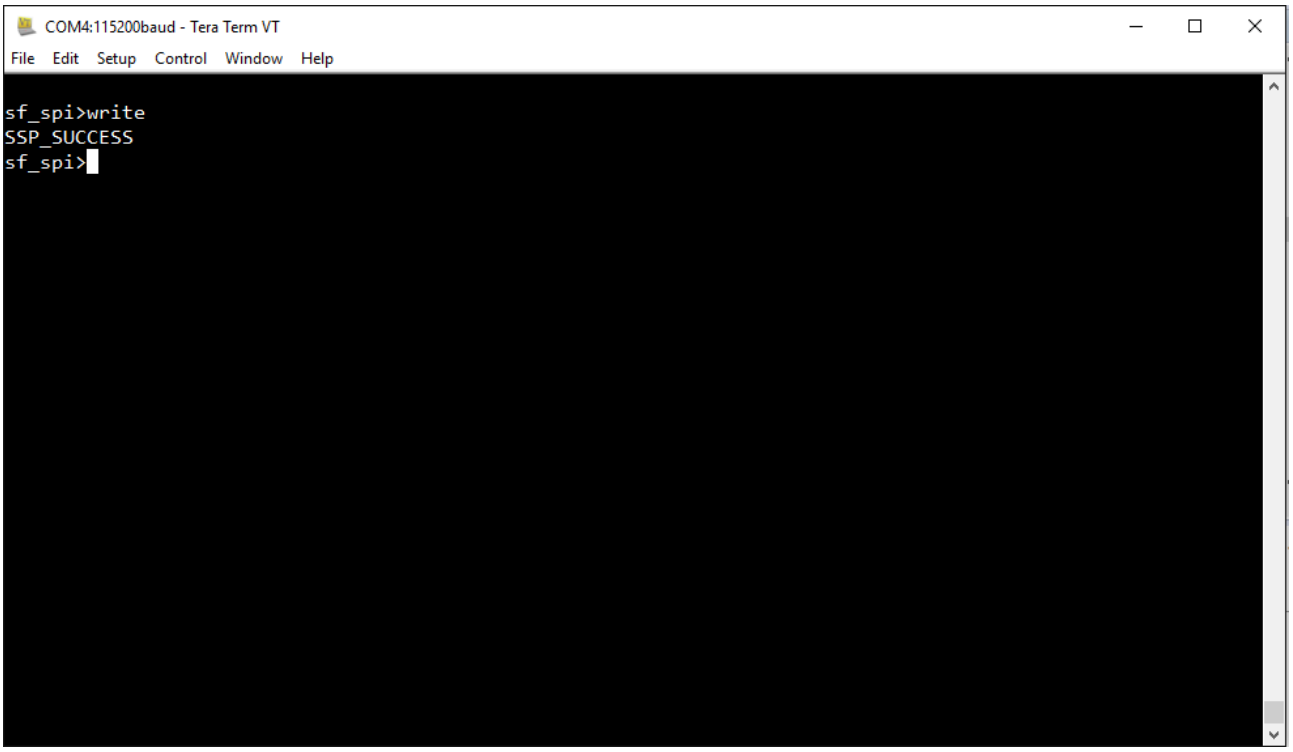
STEP 2: Type `open` in the terminal and press Enter to initialize the SPI Framework.



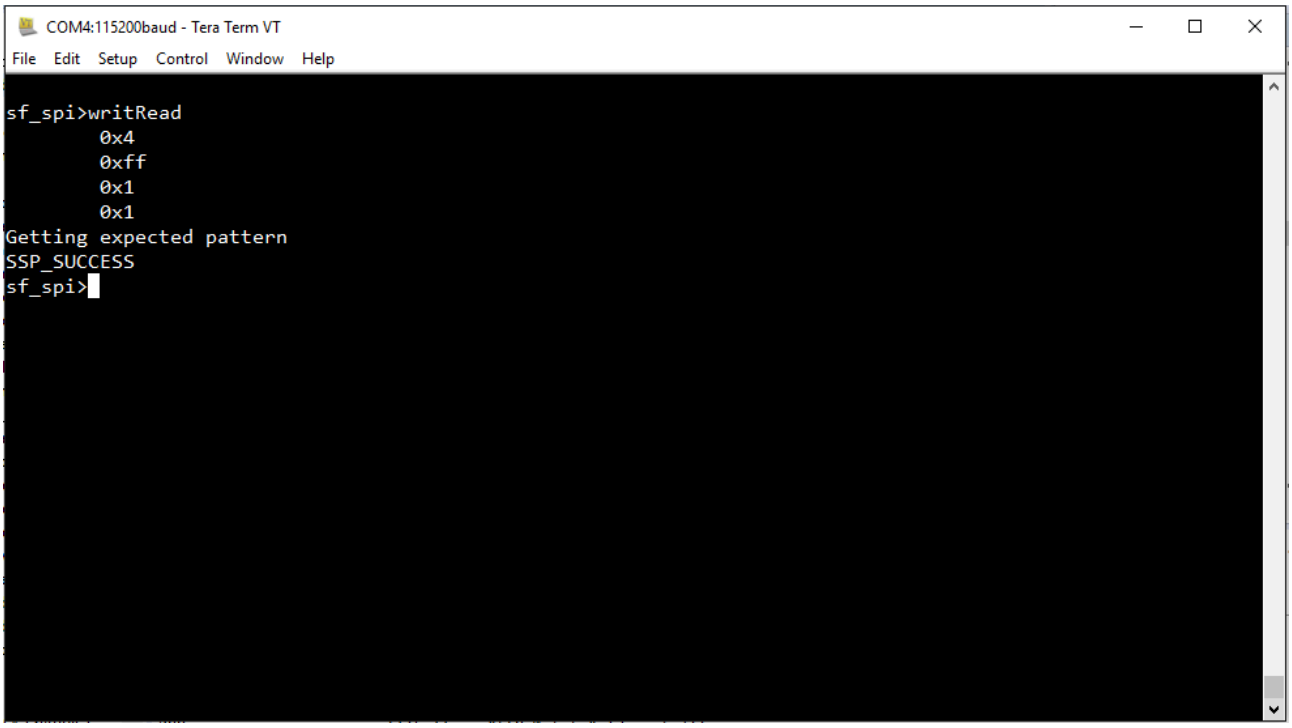
STEP 3: Type `read` command in the terminal and press Enter to read data from the slave device.



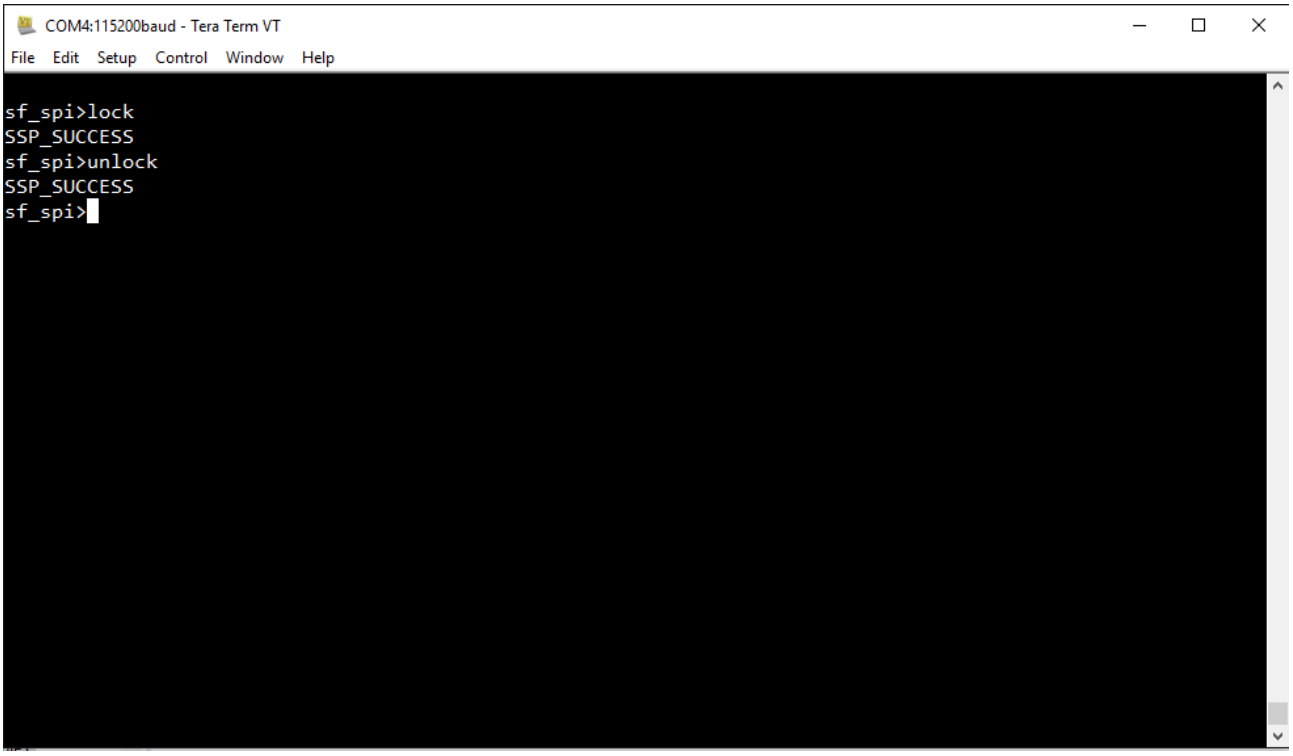
STEP 4: Type `write` command in terminal and press Enter to write data into slave device.



STEP 5: Type `writRead` command in terminal and press Enter for simultaneously write and read data from slave device



STEP 6: Type `lock` and `unlock` commands to respectively lock and unlock the bus for a device.



STEP 7: Type `close` command to close the SPI Framework.

NOTES:

1. In Developer Example, the command to perform write-read operation is writRead. This spelling mistake is introduced deliberately to bypass an issue with the console framework
2. Prior to the building and running the Developer Example, the configuration steps detailed in section ‘SPI Framework configuration steps’ should be followed to configure the BLE device correctly. Any wrong configuration will cause the device to not to respond to any of the commands.
3. During the write and write-read operations, a predetermined set of values are written to the BLE device instead of getting the data from the user. This is because BLE device expects to receive HCI commands and writing wrong or corrupt data might degrade the performance (or worse, cause damage) to the BLE module.
4. Developer example will exercise the write, read, and writRead API’s on BLE reset to observe the default expected values of BLE.
5. For details about BLE see the EM9301 datasheet.

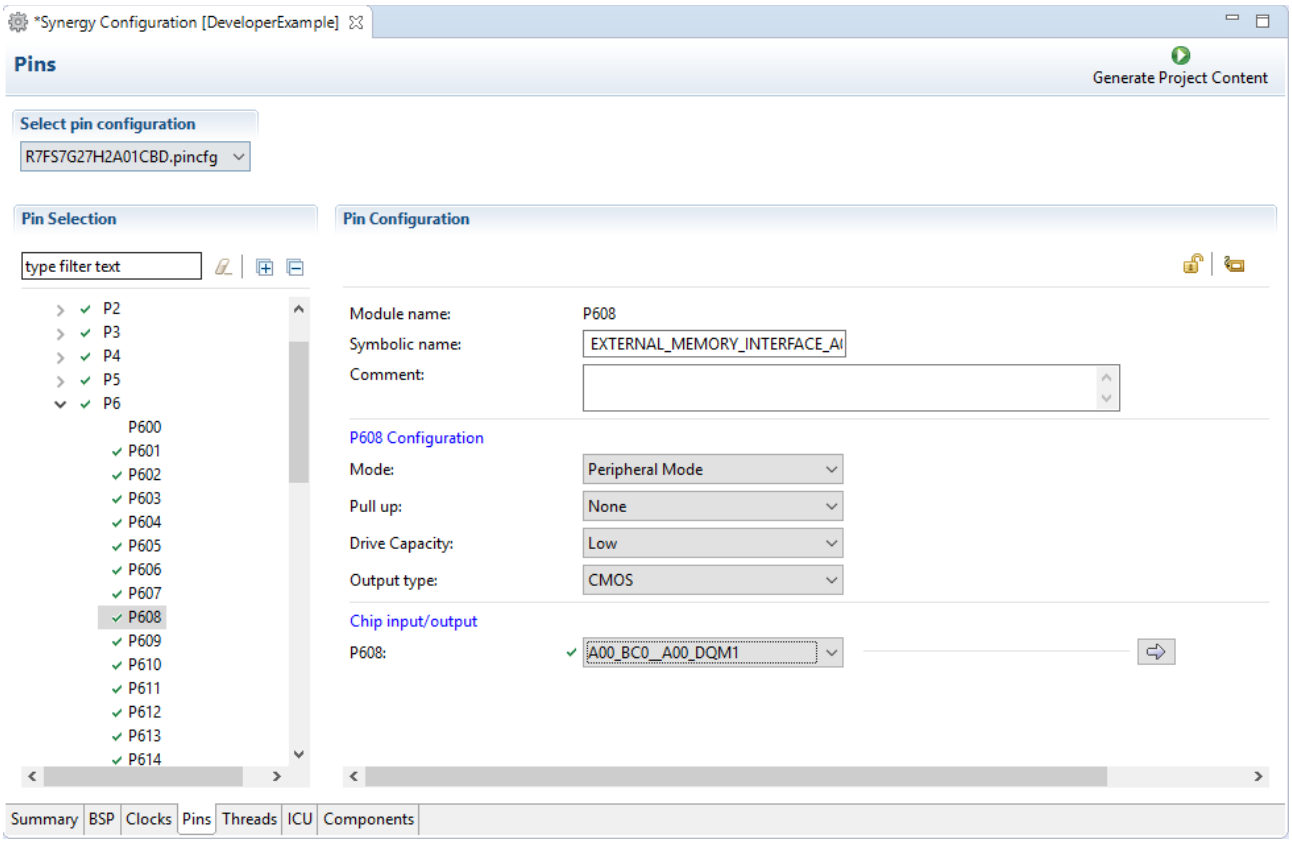
30. Developer Example: HAL JPEG Decode Driver

30.1 Introduction

The Developer Example exercises the JPEG decode driver interface to perform decode operation of a JPEG image. The resulting decoded image will be displayed in the e^2 studio debug window.

30.2 JPEG Decoder Pin Configuration

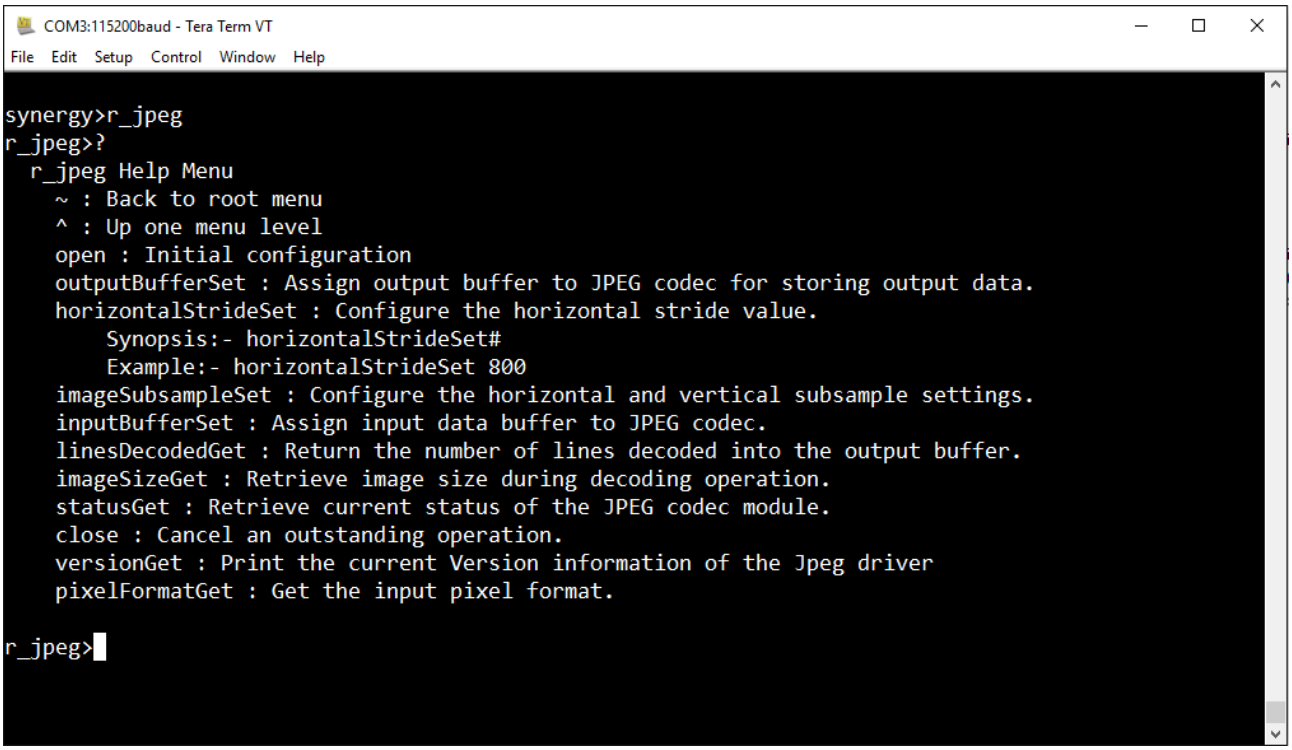
In order to run HAL JPEG and JPEG Framework change the pin (P6_8) configuration as shown below.



30.3 Run the HAL JPEG Decode Driver application

Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

Step1: Type `r_jpeg` in the terminal and press Enter to access the JPEG HAL submenu. For help, type '?' and press Enter.

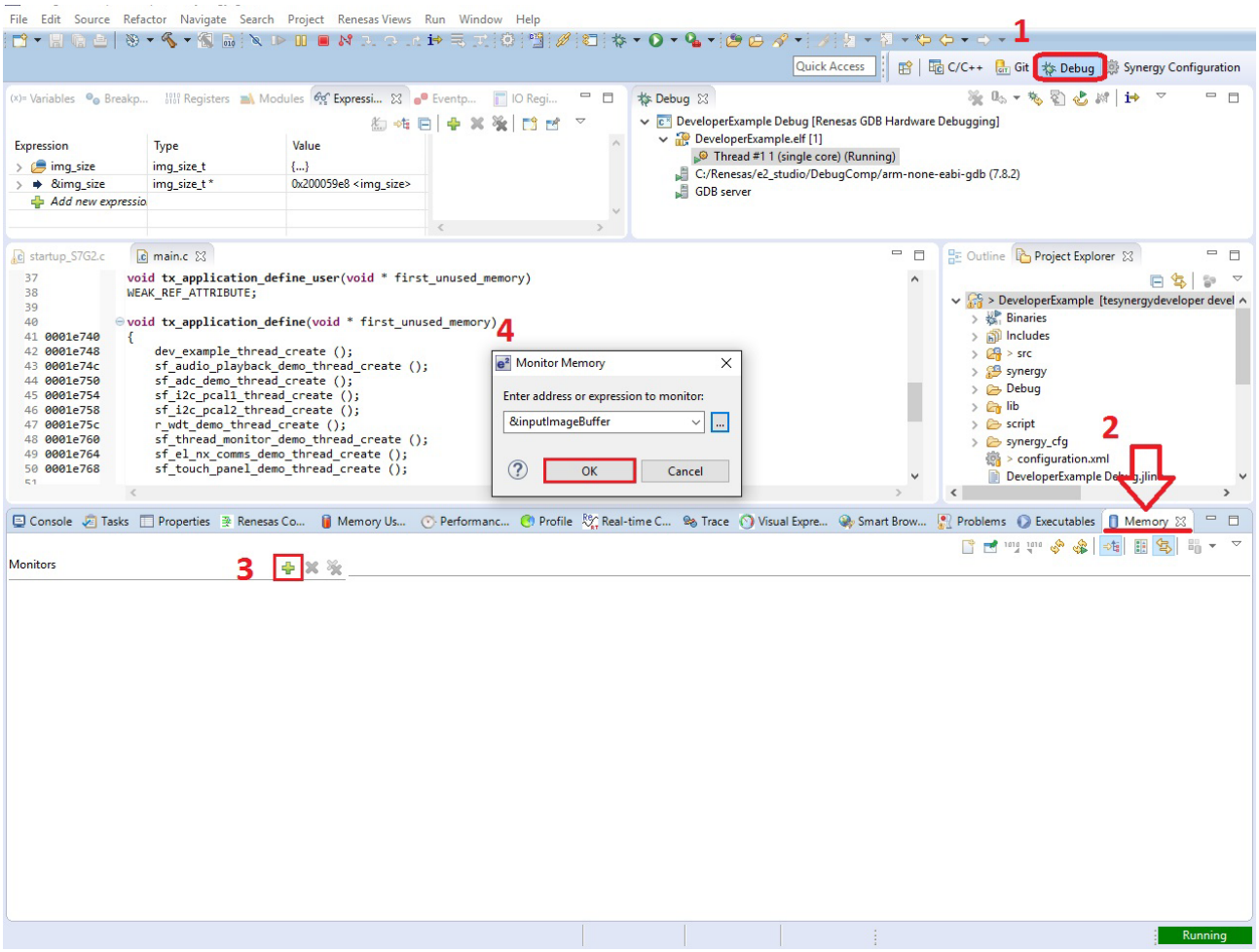


Step 2: Type `open` in terminal to open the JPEG Driver.

Step 3: In e^2 studio's debug window go to memory tab and add the input image (`inputImageBuffer`) and output image (`outputImageBuffer`) buffer address.

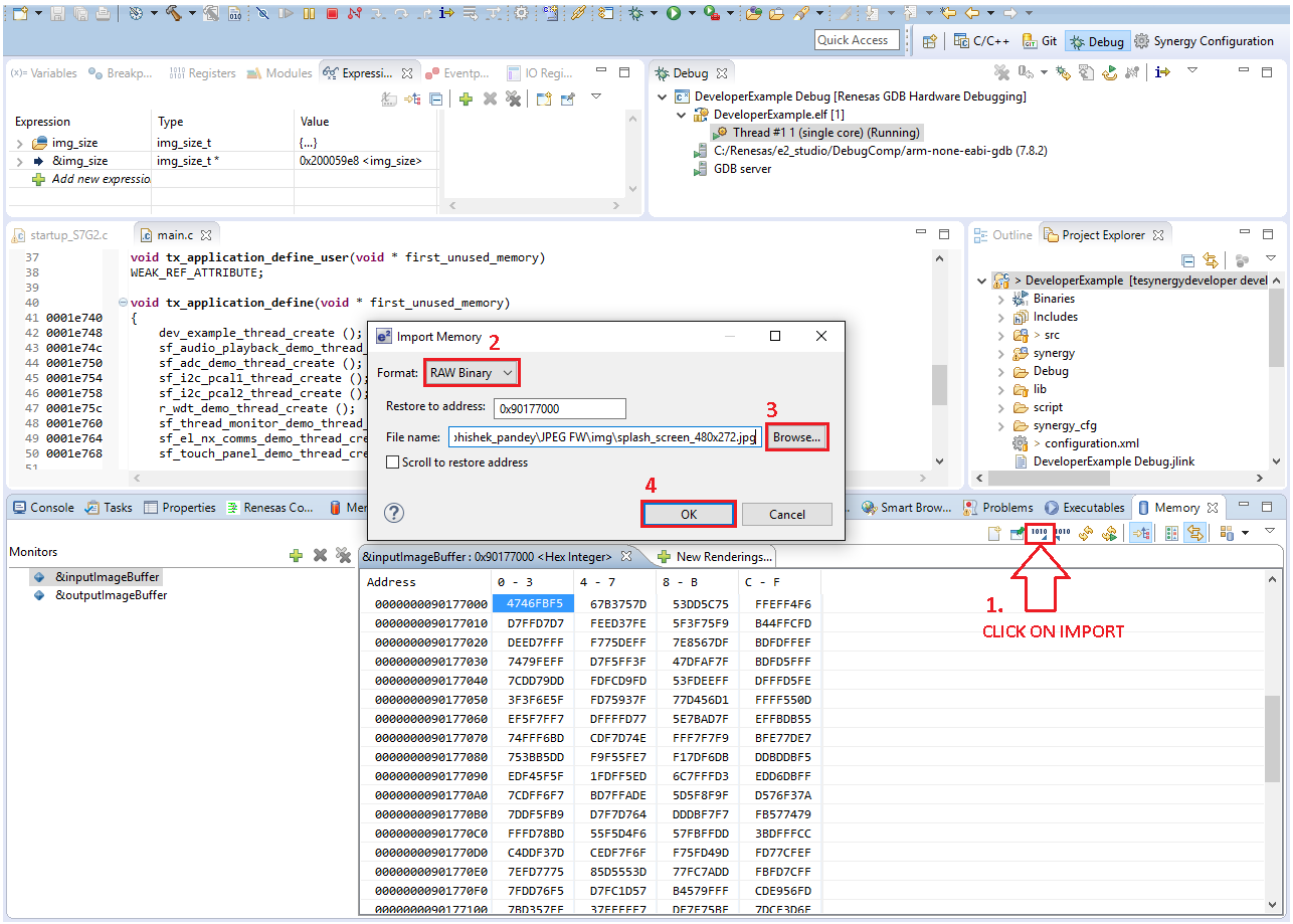
Sequence

1. Select Debug prospective
2. Select Memory tab
3. Click on add button
4. Enter the address and press **OK**.



Follow a similar sequence for setting up `outputImageBuffer` address.

Step 4: Import the JPEG image form the file explorer to the `inputImageBuffer` addresss.



NOTE: Image size should not be greater than the allocated input buffer size (750 KB), otherwise the driver will return an error. You can increase the size of input buffer from the source code of HAL JPEG (file name r_jpeg_commands.c) by updating the `INPUT_BUFF_SIZE` value.

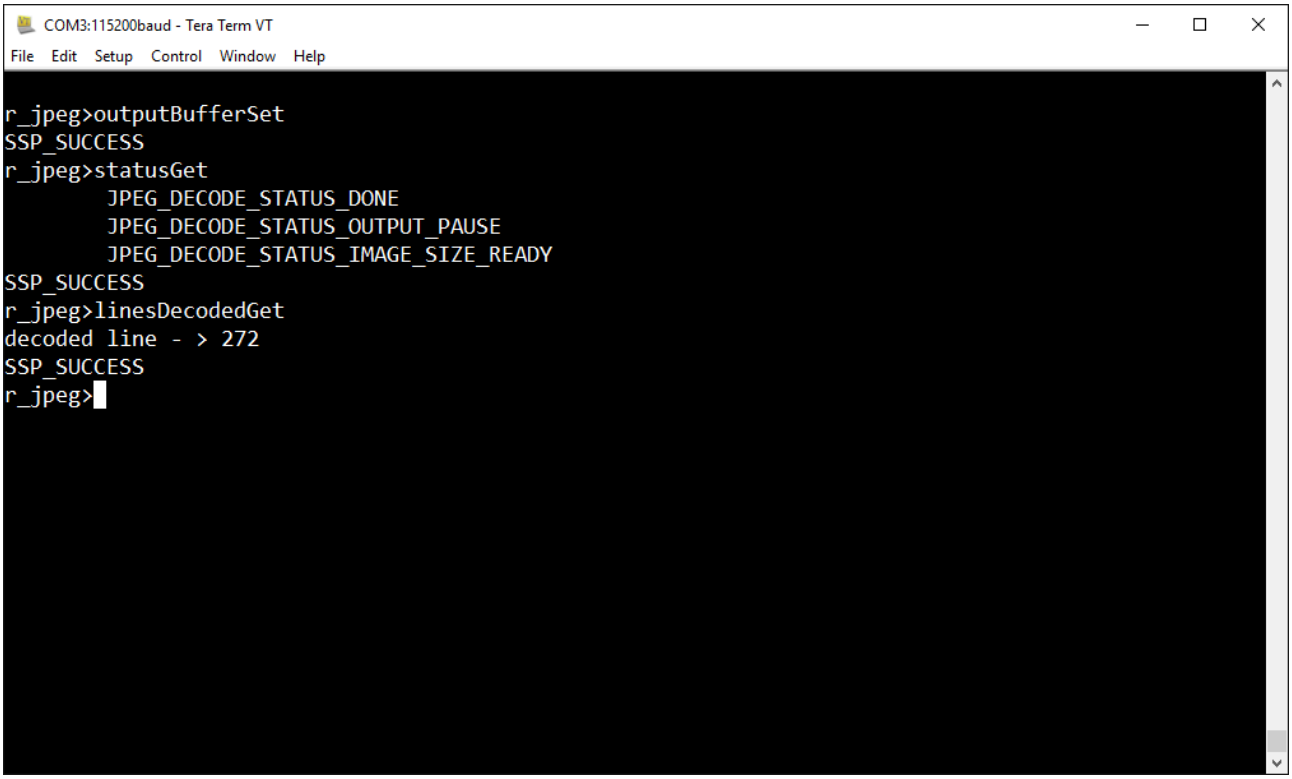
Step 4: In terminal execute the following command to set the input image and decoding parameter.

```
COM3:115200baud - Tera Term VT
File  Edit  Setup  Control  Window  Help
1 r_jpeg>inputBufferSet
SSP_SUCCESS
2 r_jpeg>imageSizeGet
Horizontal = 480
Vertical   = 272
SSP_SUCCESS
3 r_jpeg>pixelFormatGet
JPEG_DECODE_COLOR_SPACE_YCBCR420
SSP_SUCCESS
4 r_jpeg>imageSubsampleSet

Set image Subsample for horizontal and vertical
---HINT This allows an application to reduce the size of the decoded image
0.JPEG_DECODE_OUTPUT_NO_SUBSAMPLE
1.JPEG_DECODE_OUTPUT_SUBSAMPLE_HALF
2.JPEG_DECODE_OUTPUT_SUBSAMPLE_ONE_QUARTER
3.JPEG_DECODE_OUTPUT_SUBSAMPLE_ONE_EIGHTH

For horizontal - 0
For vertical   - 0
SSP_SUCCESS
5 r_jpeg>horizontalStrideSet 480
SSP_SUCCESS
r_jpeg>
```

- `inputBufferSet` command will set the address of `inputImageBuffer` to jpeg codec for decode operation.
- `imageSizeGet` and `pixelFormatGet` will print the image size(in pixel) and image pixel format on the console screen.
- Set the image sub-sample for horizontal and vertical by entering command `imageSubsampleSet` command.
- `horizontalStrideSet` sets the horizontal stride value by entering `horizontalStrideSet` along with value

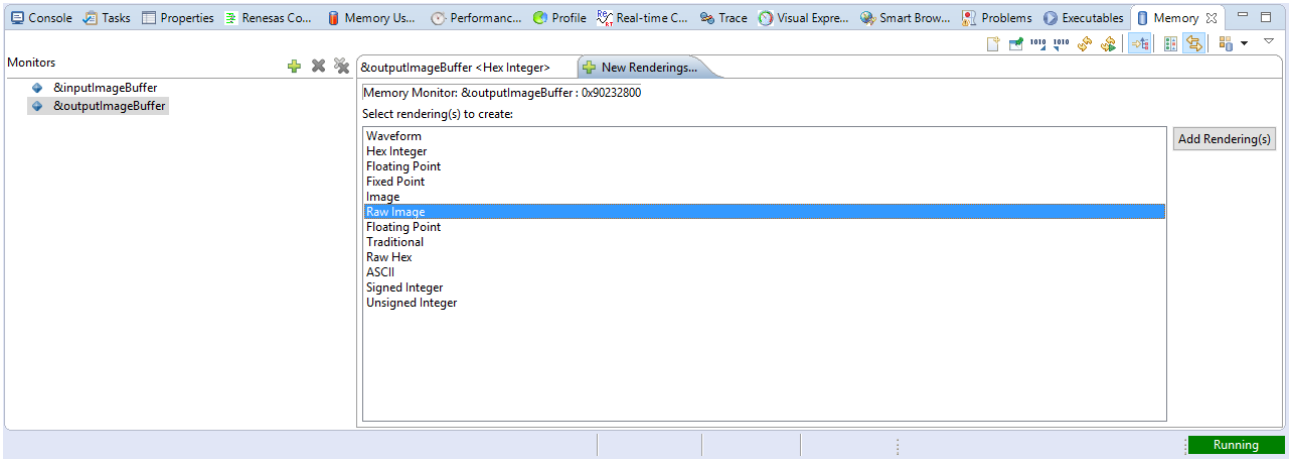
NOTE: Horizontal stride value should not be less than the horizontal pixel value.

Step 5: Set the output buffer by entering `outputBufferSet` command in console. Set up output buffer image to trigger the JPEG decode operation. You can check your current decode operation status via `statusGet command.`

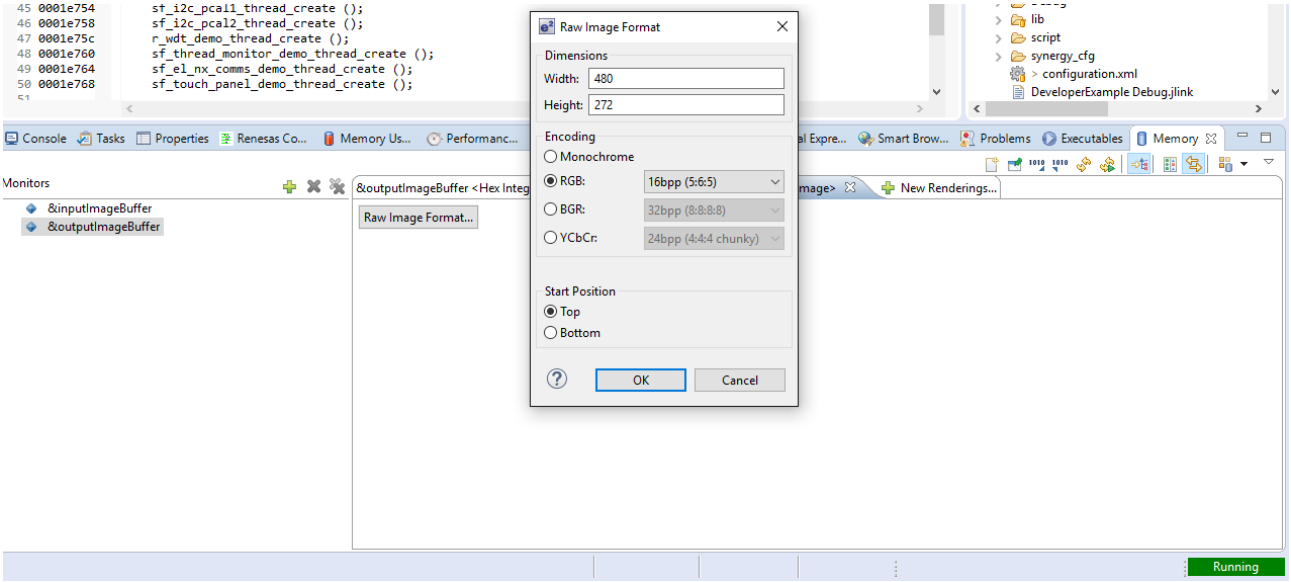


`linesDecodedGet` command will return the number of line decoded by JPEG codec.

Step 6: If `statusGet` returns `JPEG_DECODE_STATUS_DONE` it means that your current JPEG operation is completed with success. In order to see the output image, go to the e^2 studio debug window under memory tab select `outputImageBuffer` and add raw image rendering and set the horizontal and vertical pixel width as well as RBG format.



Select Encoding as RGB 565 with the Start Position at Top.



Final decode output image:

Step 6: Enter close command to close JPEG HAL driver.

NOTE: Try this Developer Example with different image sub-sample values for horizontal and vertical as described in S**tep 4** and see the effect on output image.

31. Developer Example: JPEG Decode Framework

31.1 Introduction

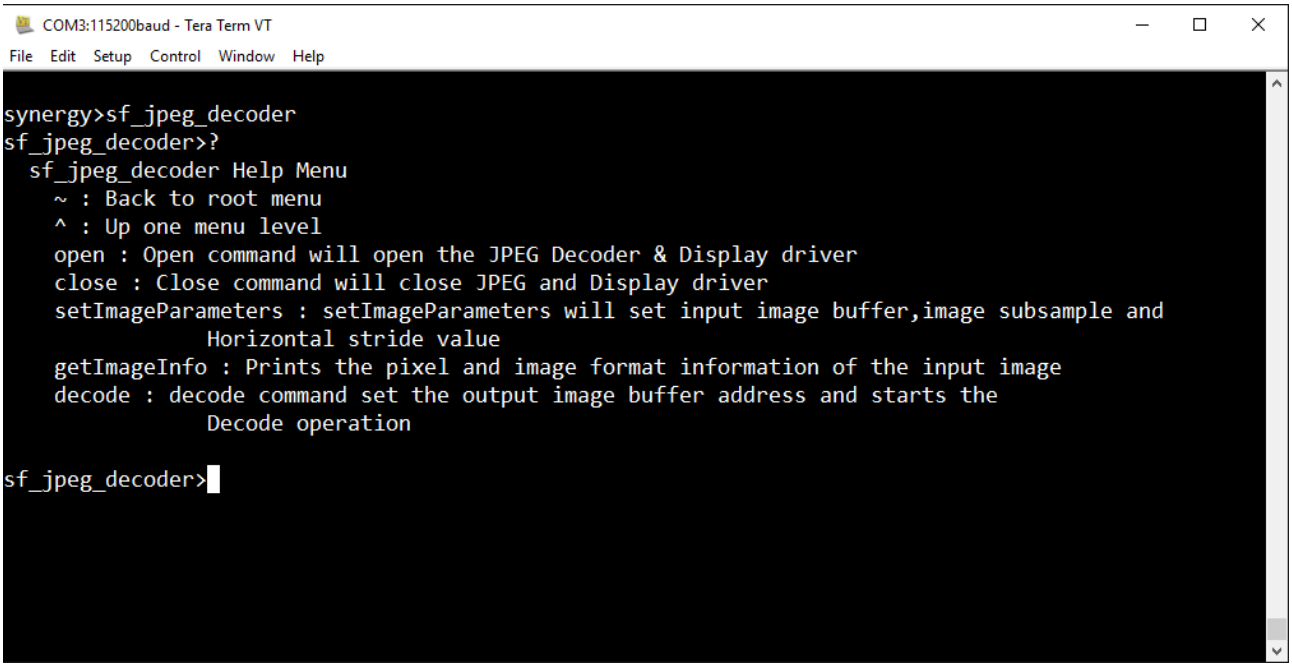
The Developer Example for JPEG Decode Framework will demonstrate the decode operation on JPEG image which include selecting image from SD card and displaying the decoded image on LCD screen.

31.2 Run the JPEG Decode Framework application

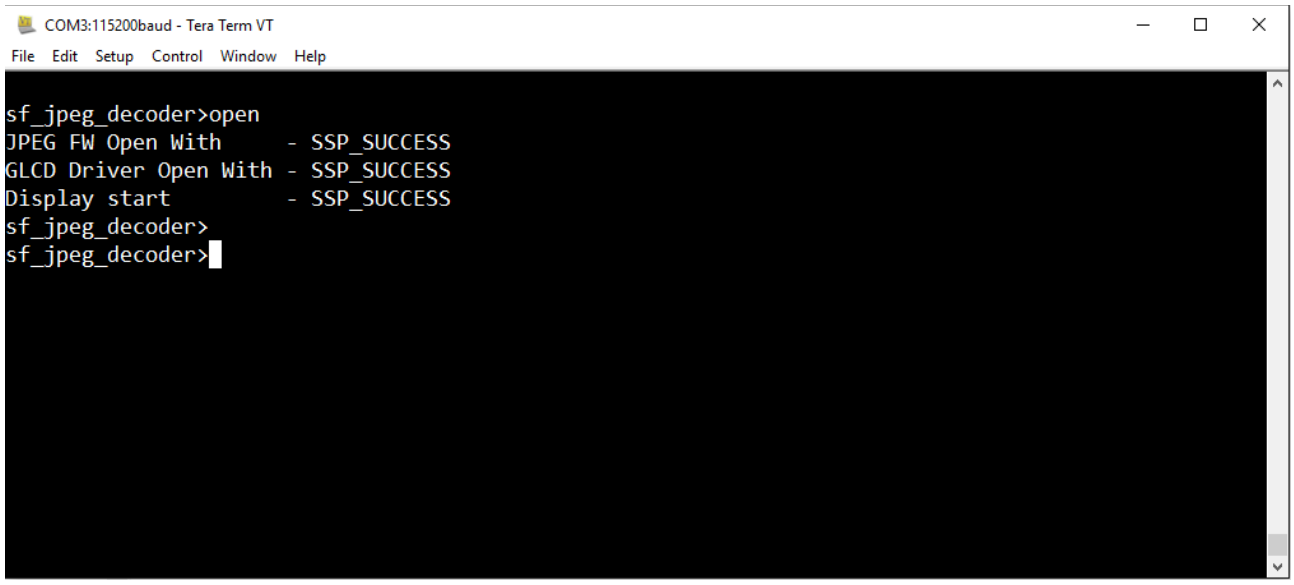
Follow the steps described in Build and Run a Developer Example application to obtain the Help menu with the list of applications in the terminal window.

NOTE: Change the pin configuration described in JPEG Decoder Pin Configuration.

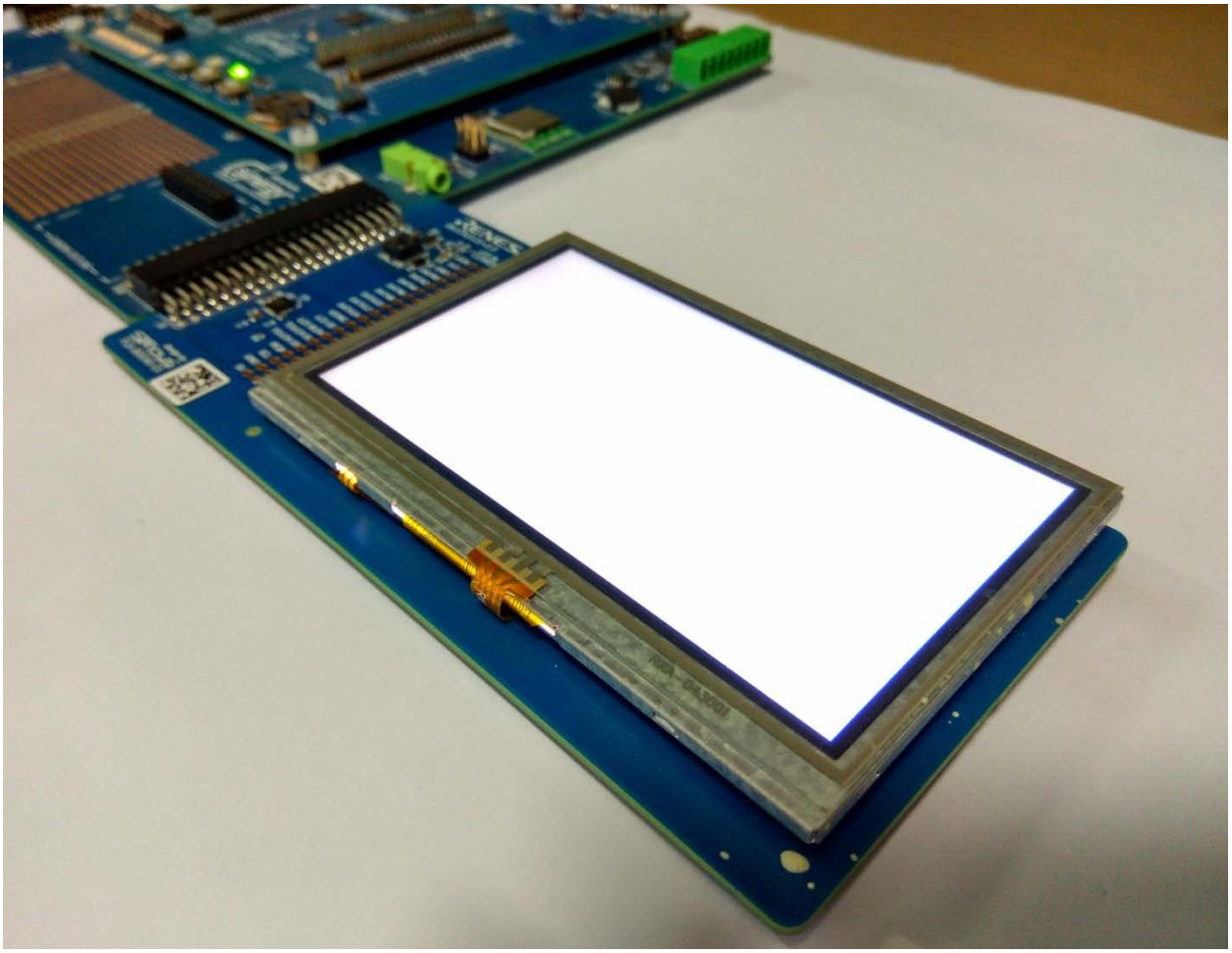
Step 1: Type `sf_jpeg_decoder` in the terminal and press Enter to access the JPEG HAL submenu. For Help, type "?" and press Enter.



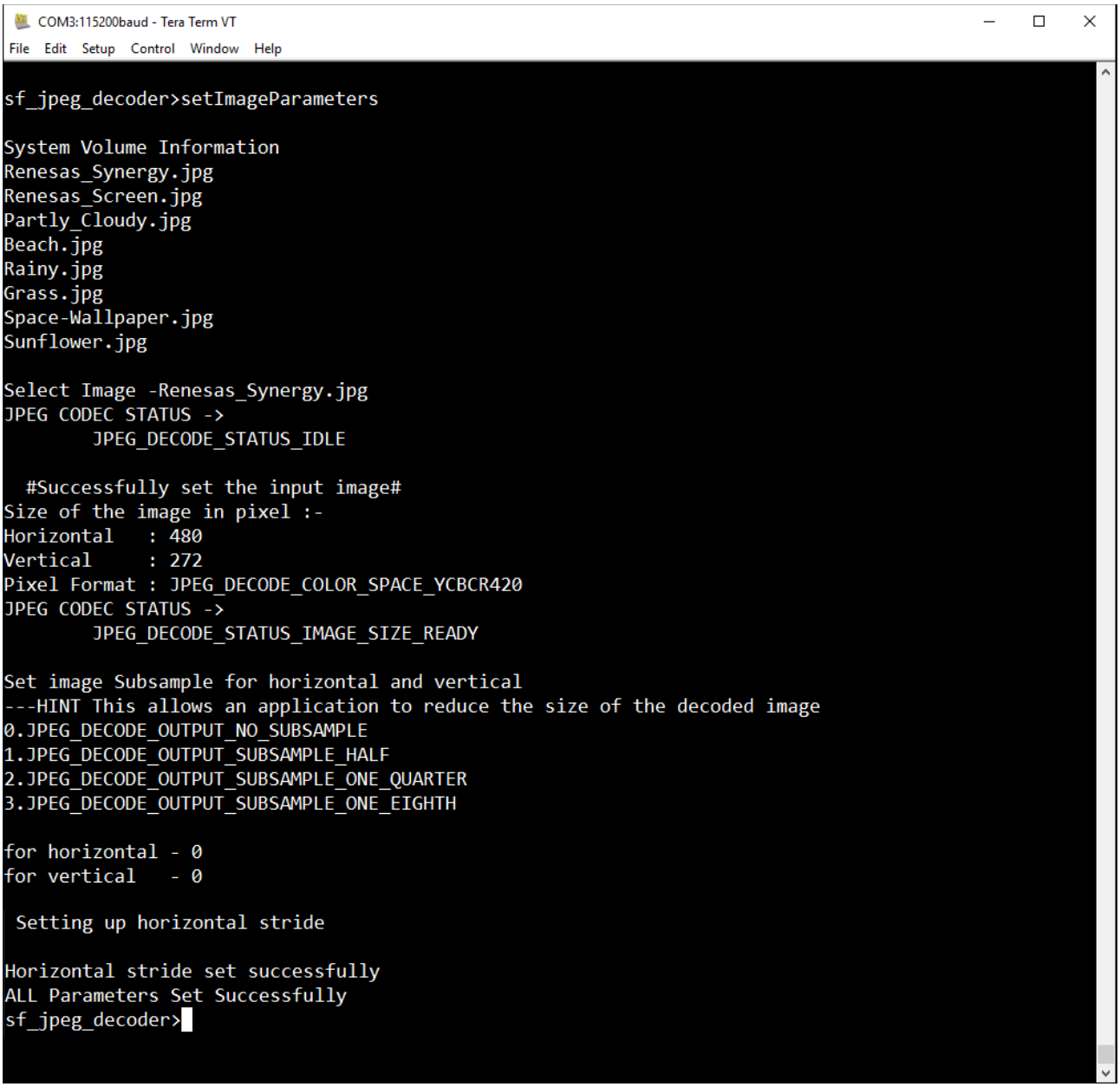
Step 2: Type `open` command to open the JPEG & GLCD Framework and Driver respectively.



Execution of `open` command will turn the LCD panel ON.

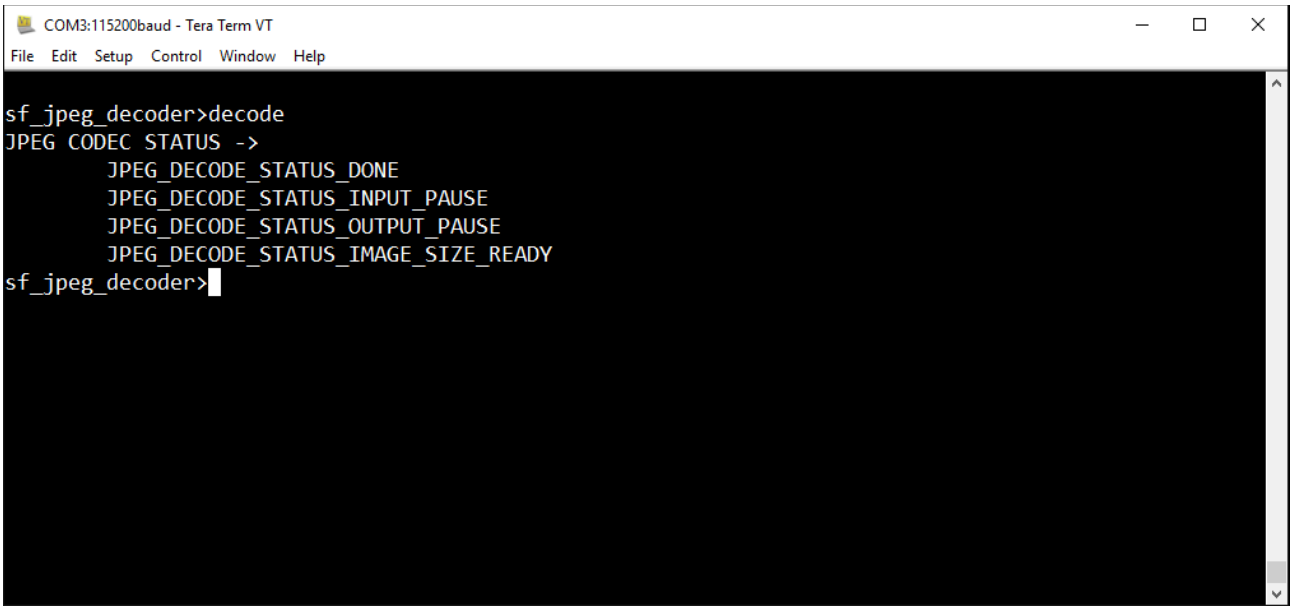


Step 3: Type `setImageParameter` command to set the input image and image parameters. Entering `setImageParameter` command will show you the available images in the SD card on the console screen. Select any image and press the ENTER key. The JPEG codec processes the JPEG header and prints the image information, for example, the image size and pixel format in the console. It prompts you to set up the image sub-sample value for the horizontal and vertical. Select the appropriate value to reduce the size of the image or set to "0"(zero) for horizontal and vertical to keep the original size.



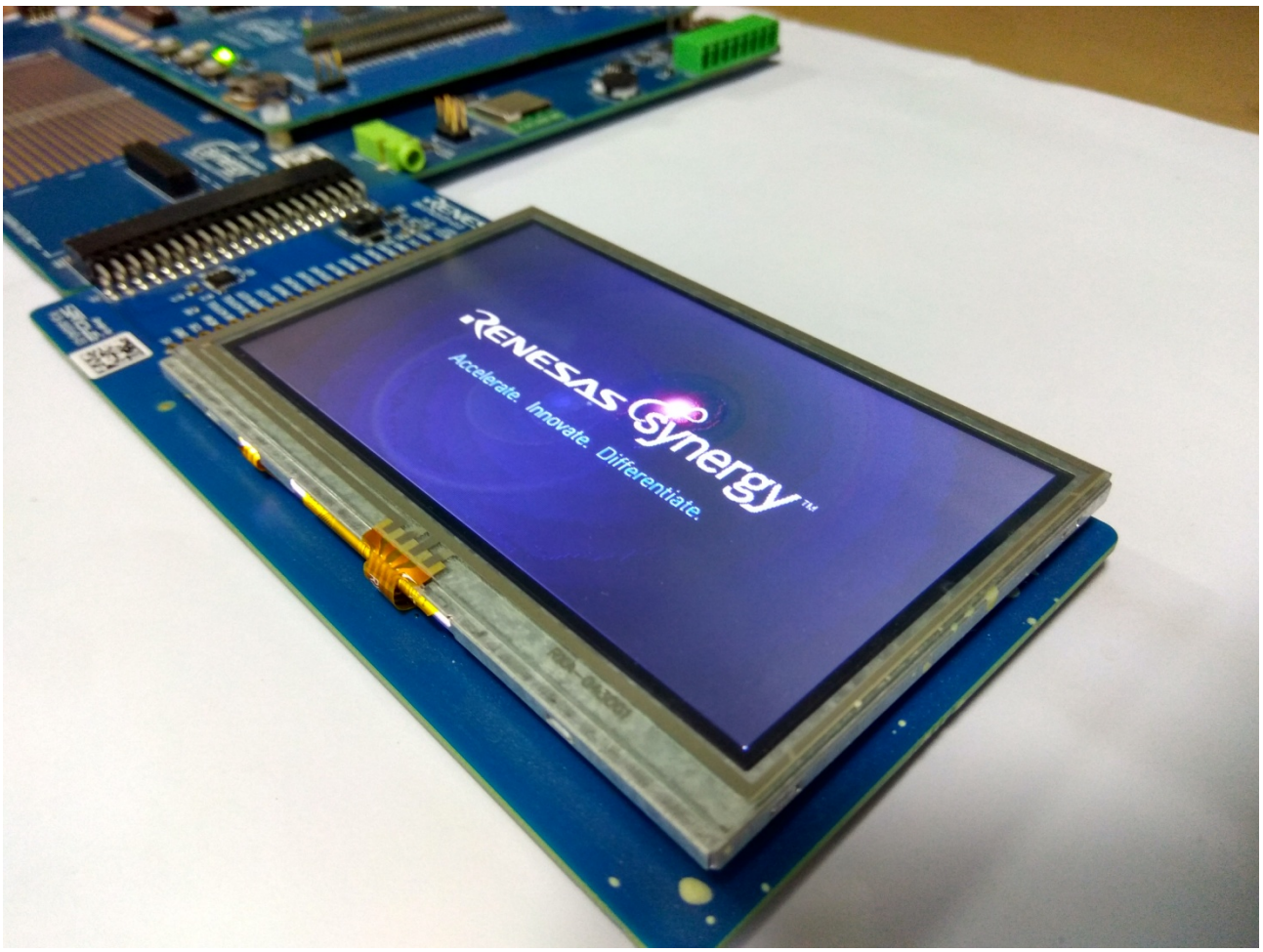
NOTE: In case If you fail to set the image or its parameters, you need to do `close>open` and then try with `setImageParameter` command.

Step 4: If the image and parameters are set with success, type the `decode` command to start the JPEG decode operation.

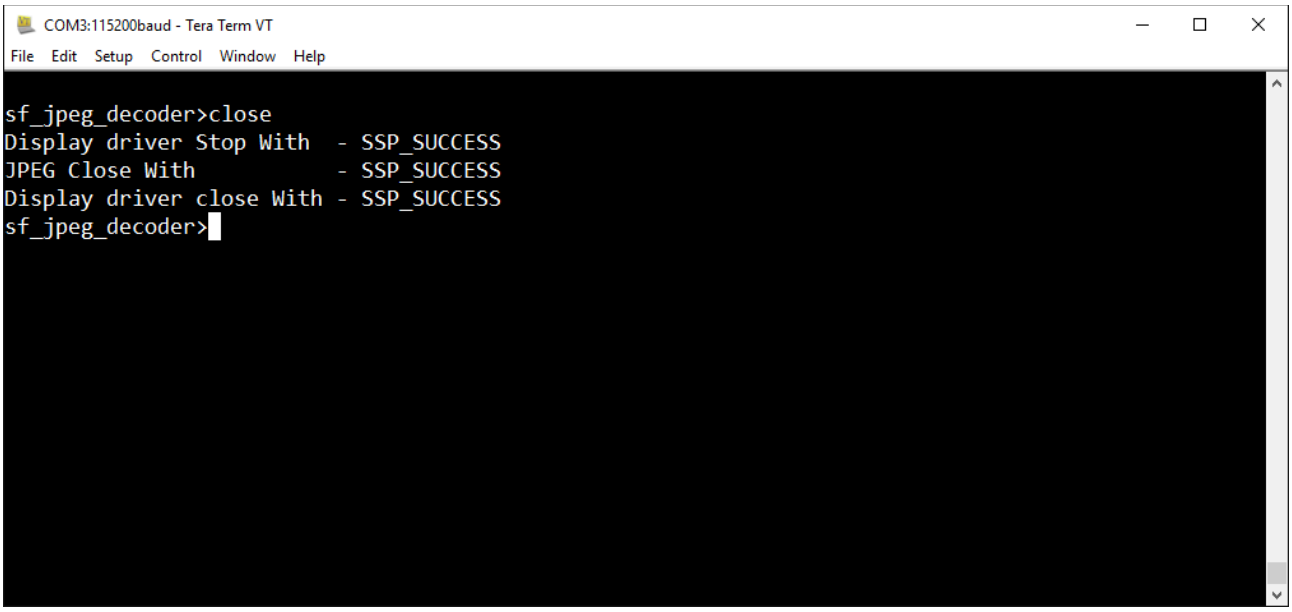


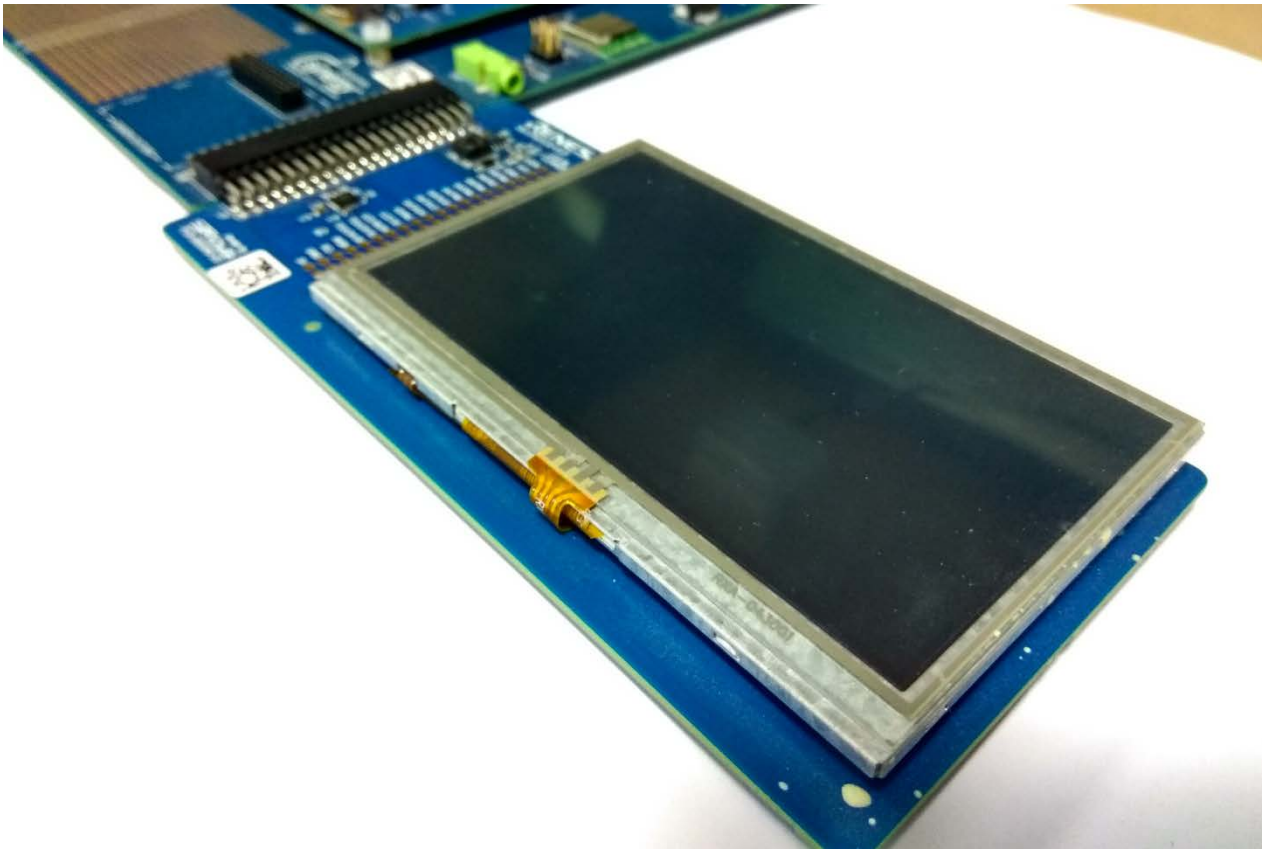
The `decode` command converts the jpeg file to a raw RBG image file that is displayed via the LCD screen.

(`decode` command will automatically display the decoded image on LCD screen)



Step 5: Enter close command to close JPEG and LCD.


COM3:115200baud - Tera Term VT
File Edit Setup Control Window Help
sf_jpeg_decoder>close
Display driver Stop With - SSP_SUCCESS
JPEG Close With - SSP_SUCCESS
Display driver close With - SSP_SUCCESS
sf_jpeg_decoder>



32. Additional Technical Notices

Subscribe to the Synergy Technical Bulletin Board to receive the latest technical news and notifications about new features, known issues, workarounds, and release announcements. To subscribe, visit http://renesasrulz.com/synergy/synergy_tech_notes/f/214.aspx. Sign in to Renesas Rulz, and click 'Email Subscribe to this forum'.

Additional technical information, including informative papers and articles on SSP and Renesas Synergy can be found at the Synergy Knowledge Base, https://knowledgebase.renesas.com/Renesas_Synergy_Platform.

Website and Support

Support: https://synergygallery.renesas.com/support

Technical Contact Details:

- America: https://renesas.zendesk.com/anonymous_requests/new
- Europe: http://www.renesas.eu/support/index.jsp
- Japan: http://japan.renesas.com/contact/index.jsp

Revision History

		Description	
Rev.	**Date**	**Page**	**Summary**
1.40	Sep 1, 2016	-	Initial version

General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

1. Handling of Unused Pins

   Handle unused pins in accordance with the directions given under Handling of Unused Pins in the manual.

   — The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible. Unused pins should be handled as described under Handling of Unused Pins in the manual.

2. Processing at Power-on

   The state of the product is undefined at the moment when power is supplied.

   — The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the moment when power is supplied.
   In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the moment when power is supplied until the reset process is completed.
   In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the moment when power is supplied until the power reaches the level at which resetting has been specified.

3. Prohibition of Access to Reserved Addresses

   Access to reserved addresses is prohibited.

   — The reserved addresses are provided for the possible future expansion of functions. Do not access these addresses; the correct operation of LSI is not guaranteed if they are accessed.

4. Clock Signals

   After applying a reset, only release the reset line after the operating clock signal has become stable. When switching the clock signal during program execution, wait until the target clock signal has stabilized.

   — When the clock signal is generated with an external resonator (or from an external oscillator) during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Moreover, when switching to a clock signal produced with an external resonator (or by an external oscillator) while program execution is in progress, wait until the target clock signal is stable.

5. Differences between Products

   Before changing from one product to another, i.e. to a product with a different part number, confirm that the change will not lead to problems.

   — The characteristics of Microprocessing unit or Microcontroller unit products in the same group but having a different part number may differ in terms of the internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
3. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from such alteration, modification, copy or otherwise misappropriation of Renesas Electronics product.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The recommended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
   "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots etc.
   "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; and safety equipment etc.
   Renesas Electronics products are neither intended nor authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems, surgical implantations etc.), or may cause serious property damages (nuclear reactor control systems, military equipment etc.). You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application for which it is not intended. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for which the product is not intended by Renesas Electronics.
6. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You should not use Renesas Electronics products or technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. When exporting the Renesas Electronics products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, who distributes, disposes of, or otherwise places the product with a third party, to notify such third party in advance of the contents and conditions set forth in this document, Renesas Electronics assumes no responsibility for any losses incurred by you or third parties as a result of unauthorized use of Renesas Electronics products.
11. This document may not be reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

RENESAS

SALES OFFICES

Renesas Electronics Corporation

http://www.renesas.com

Refer to "http://www.renesas.com/" for the latest and detailed information.

Renesas Electronics America Inc.
2801 Scott Boulevard Santa Clara, CA 95050-2549, U.S.A.
Tel: +1-408-588-6000, Fax: +1-408-588-6130

Renesas Electronics Canada Limited
9251 Yonge Street, Suite 8309 Richmond Hill, Ontario Canada L4C 9T3
Tel: +1-905-237-2004

Renesas Electronics Europe Limited
Dukes Meadow, Millboard Road, Bourne End, Buckinghamshire, SL8 5FH, U.K
Tel: +44-1628-585-100, Fax: +44-1628-585-900

Renesas Electronics Europe GmbH
Arcadiastrasse 10, 40472 Düsseldorf, Germany
Tel: +49-211-6503-0, Fax: +49-211-6503-1327

Renesas Electronics (China) Co., Ltd.
Room 1709, Quantum Plaza, No.27 ZhiChunLu Haidian District, Beijing 100191, P.R.China
Tel: +86-10-8235-1155, Fax: +86-10-8235-7679

Renesas Electronics (Shanghai) Co., Ltd.
Unit 301, Tower A, Central Towers, 555 Langao Road, Putuo District, Shanghai, P. R. China 200333
Tel: +86-21-2226-0888, Fax: +86-21-2226-0999

Renesas Electronics Hong Kong Limited
Unit 1601-1611, 16/F., Tower 2, Grand Century Place, 193 Prince Edward Road West, Mongkok, Kowloon, Hong Kong
Tel: +852-2265-6688, Fax: +852 2886-9022

Renesas Electronics Taiwan Co., Ltd.
13F, No. 363, Fu Shing North Road, Taipei 10543, Taiwan
Tel: +886-2-8175-9600, Fax: +886 2-8175-9670

Renesas Electronics Singapore Pte. Ltd.
80 Bendemeer Road, Unit #06-02 Hyflux Innovation Centre, Singapore 339949
Tel: +65-6213-0200, Fax: +65-6213-0300

Renesas Electronics Malaysia Sdn.Bhd.
Unit 1207, Block B, Menara Amcorp, Amcorp Trade Centre, No. 18, Jln Persiaran Barat, 46050 Petaling Jaya, Selangor Darul Ehsan, Malaysia
Tel: +60-3-7955-9390, Fax: +60-3-7955-9510

Renesas Electronics India Pvt. Ltd.
No.777C, 100 Feet Road, HALII Stage, Indiranagar, Bangalore, India
Tel: +91-80-67208700, Fax: +91-80-67208777

Renesas Electronics Korea Co., Ltd.
12F., 234 Teheran-ro, Gangnam-Gu, Seoul, 135-080, Korea
Tel: +82-2-558-3737, Fax: +82-2-558-5141

Colophon 5.0