Freescale Semiconductor
User's Guide

Document Number: EMBEDDEDHOSTUG
Rev. 0, 04/2016

Embedded Host User's Guide

Contents

1 Introduction

The Kinetis bootloader package provides several tools for ease of use for the customer. The tools range from command line-driven host tools, such as blhost, to fully interactive GUI based tools such as FlashTool, MfgTool, and so on. These tools run on host PC and provide ways and means to communicate to the Kinetis bootloader running on the target device in order to exercise bootloader commands and properties, and to program the flash with the application image.

The Embedded host application or tool is new and different from PC-based tools. It provides an example implementation of code running on an embedded platform host and communicating over serial connection to flash the target device. The Embedded host tool implementation uses the TWR-K65F180M Tower System module as the host platform. The target can be any Kinetis MCU platform running Kinetis bootloader. The Embedded host tool can communicate over I2C, SPI, UART, and CAN interfaces with the Kinetis bootloader running on the target platform.

2 Overview

This user's guide describes in detail how to configure and use the Embedded Host tool in communicating with the target Kinetis bootloader device.

Embedded Host provides three interfaces for the user to interact with. The first interface is the command prompt interface. The user can exercise various bootloader commands using a PC connected to the host platform over UART. Any PC terminal applications, such as TeraTerm, can be used to communicate over the serial port using the command prompt interface, and send bootloader supported commands to the target device.

The second interface is USB MSD interface. Embedded host enumerates on the PC as a USB Mass Storage Class device when connected to the USB port, facilitating the download of application images to flash the target platform.

The third interface is the host platform's buttons interface. Using the button press on the host platform, the user can initiate programming the image to the target device. The LEDs are used for reporting status of the button press operation. The image to program should be available in the host platform flash memory for this interface to work. The image can be transferred to the host memory using Embedded host's USB MSD interface itself or any other user preferred method.

All three interfaces are discussed later in this user's guide.

The Embedded host implementation can be used as a reference example to implement embedded host solutions on various hybrid platforms involving multiple SoCs. Applications running on master processor can flash the Kinetis MCU present on the same device or on other device connected over one of the serial interfaces.

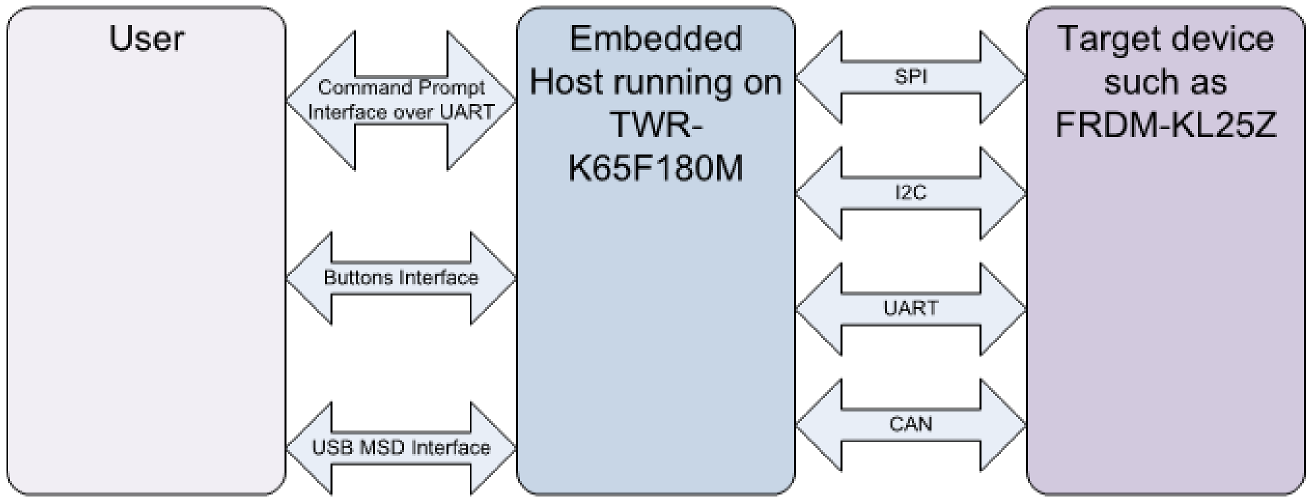

User
Command Prompt Interface over UART
Buttons Interface
USB MSD Interface
Embedded Host running on TWR-K65F180M
SPI
I2C
UART
CAN
Target device such as FRDM-KL25Z

Figure 1. Embedded host setup

3 Requirements

Table 1. System requirements

Component	Requirements
Processor	1 gagahertz (GHz) or faster x86 or x64 processor
Memory	1 GB RAM (32-bit); 2 GB RAM (63-bit)

Table continues on the next page...

Table 1. System requirements (continued)

Component	Requirements
Operating system	Windows® OS XP, Windows® OS 7, Windows® OS 8, Windows® OS 10
Toolchain	IAR Embedded Workbench for ARM® v7.50.2 or later
Hardware	Host platform: TWR-K65F180M, TWR-ELEV, TWR-SER Target platform: Device having Kientis bootloader, e.g., K82, KL25
Other tools	TeraTerm or any PC host serial terminal application

3.1 Hardware description

The TWR-K65F180M is a Tower MCU Module featuring the K65FN2M0VMI18, an ARM® Cortex®-M4F-based MCU with 2 MB on-chip flash, 256 KB on-chip SRAM, SDRAM controller, and dual USB controllers in a 169-pin MAPBGA package. It has a maximum core operating frequency of 180 MHz. It is intended for use in the Tower System, but can operate as a standalone module. An on-board debug circuit, OPENSDA, provides the SWD debug interface and power supply input through a single USB mini-AB connector

4 Embedded Host Utility Application

As mentioned in the introduction chapter, the Embedded host is a brand new tool first introduced with the v2.0.0 package of the Kinetis bootloader, available with the complete source code at www.nxp.com/KBOOT. This section provides a description of each command supported over the serial port, and how to transfer application image file using the USB MSD interface of the Embedded host.

4.1 Embedded host commands

This section provides brief description and usage of commands supported by the Embedded host. Embedded host transports most of the commands to the Kinetis bootloader running on the target device.

Not all commands are supported on all Kinetis bootloader platforms. If a command is not supported by the Kinetis bootloader, it returns *k_StatusUnkownCommand*.

See the ROM bootloader or flashloader chapter of the specific Kinetis device reference manual for a list of supported commands and properties

4.1.1 help

Example: *help*

This command is used to show all the commands supported by the embedded host command prompt interface.

4.1.2 exit

Example: *exit*

This command is used to exit embedded host. After exiting, only a manual reset returns back to embedded host.

4.1.3 reset

Example: *reset*

The reset sends the reset bootloader command to the Kinetis bootloader running on the target device.

4.1.4 get-property <tag>

Example: *get-property 10*

This command is used to query the bootloader about various properties and settings. Each supported property has a unique integer tag value.

tag:

- 0x01 The current version of the Kinetis bootloader.
- 0x02 A mask of the Available Peripherals.
- 0x03 The starting address of the on-chip flash.
- 0x04 The size of the on-chip flash.
- 0x05 The size in bytes of one sector of program flash. This is the minimum erase size.
- 0x06 The number of blocks in the on-chip flash.
- 0x07 A mask of the Available Commands.
- 0x08 Status code from the last CRC check operation. Available only if the CRC check feature is supported.
- 0x09 Reserved for future use.
- 0x0a Verify Writes Flag - Boolean controlling whether the bootloader verifies writes to flash. A value of 0 means no verification is done, a value of 1 enables verification. This feature is enabled by default.
- 0x0b The maximum supported packet size for the currently active peripheral interface.
- 0x0c Reserved Regions - List of memory regions reserved by the bootloader. Returned as value pairs (<start-address-of-region>, <end-address-of-region>). If HasDataPhase flag is not set, response packet parameter count indicates number of pairs. If HasDataPhase flag is set, second parameter is the number of bytes in the data phase.
- 0x0e The starting address of the on-chip RAM.
- 0x0f The size in bytes of the on-chip RAM.
- 0x10 The value of the Kinetis System Device Identification register.
- 0x11 Flash Security Enabled Flag - Boolean indicating whether flash security is enabled. 0 means disabled, 1 means enabled.
- 0x12 The values of the Device Unique ID. The number of ID words is indicated by the parameter count in the response packet.
- 0x13 FAC Supported Flag - Boolean indicating whether Flash Access Control (FAC) module is supported. 0 means not supported, 1 means supported.
- 0x14 The number of bytes per FAC segment.
- 0x15 The number of segments available in the FAC.
- 0x16 The flash read margin setting. A value of 0 indicates Normal read margin, a value of 1 indicates User read margin, and a value of 2 indicates Factory read margin. The default is User.
- 0x17 Reserved for future use.
- 0x18 Target Version - the target build version number.

4.1.5 set-property <tag> <value>

Example: *set-property 10 0*

This command changes properties and options in the bootloader. The command accepts the same property tags used with the get-property command; however, only some properties are writable. If an attempt to write a read-only property is made, an error is returned indicating the property is read-only and cannot be changed.

Properties that can be changed all have 32-bit values.

tag:

- 0x0a Verify Writes Flag - Boolean controlling whether the bootloader verifies writes to flash. A value of 0 means no verification is done, a value of 1 enables verification. This feature is enabled by default.
- 0x16 The flash read margin setting. A value of 0 indicates Normal read margin, a value of 1 indicates User read margin, and a value of 2 indicates Factory read margin. The default is User.

4.1.6 flash-erase-region <addr> <byte>

Example: *flash-erase-region 0xa000 1024*

Erases one or more sectors of flash memory.

The start address and count must be a multiple of the word size. The entire sector(s) containing the start and end address is erased.

If the VerifyWrites property is enabled, the command performs a flash verify erase operation.

4.1.7 flash-erase-all

Example: *flash-erase-all*

Performs an erase of the entire flash memory.

If any flash regions are protected, the command fails with an error. If any flash regions are reserved by the bootloader, they are ignored (not erased).

If the VerifyWrites property is enabled, the flash-erase-all command performs a flash verify erase all operation, or multiple flash verify erase options if decomposed due to reserved regions.

4.1.8 flash-erase-all-unsecure

Example: *flash-erase-all-unsecure*

This command is only supported in new versions of the flash controller. Most Kinetis devices do not support this command, and the bootloader sends a kStatus_UnknownCommand error in response.

Performs a mass erase of the flash memory, including protected sectors and any reserved regions in flash. Flash security is immediately disabled if it was enabled and the FSEC byte in the Flash Configuration Field at address 0x40C is programmed to 0xFE.

The Mass Erase Enable option in the FSEC field is honored by this command. If mass erase is disabled, then this command fails.

This command is only useful and only present in ROM configurations of the bootloader because it erases reserved regions in flash.

4.1.9 read-memory <addr> <byte_count>

Example: *read-memory 0x3c0 32*

Read memory and print memory data.

Returns the contents of memory at the given address, for a specified number of bytes. This command can read any region of memory accessible by the CPU and not protected by security. This includes flash, RAM, and peripheral registers. Note that the minimum profile does not support reading the peripheral register space.

4.1.10 write-memory <addr> [{{<hex-data>}}]

Example: *write-memory 0xa000 {{aabbccddeeff0001020304}}*

Write memory at addr from string of hex values.

Writes a provided buffer to a specified range of bytes in memory. Can write to all accessible memory, including flash, RAM, and peripheral registers. However, if flash protection is enabled, it writes to protected sectors fails. The data specified by file is treated as binary data.

Any flash sector written to must be previously erased with either a flash-erase-all, flash-erase-region, or flash-erase-all-unsecure command.

Writing to flash requires the start address to be word-aligned. The byte count is rounded up to a multiple of the word size, and trailing bytes are filled with the flash erase pattern (0xff).

Word and halfword-aligned and sized writes to RAM and peripheral registers use appropriately sized writes. This enables writing to registers larger than a byte in a single bus transaction. Note that the minimum profile does not support writing to the peripheral register space.

If the VerifyWrites property is enabled, writes to flash performs a flash verify program operation.

4.1.11 fill-memory <addr> <byte_count> <pattern> [word | short | byte]

Example: *fill-memory 0x3c0 32 0xff byte*

Fill memory with a pattern; size is word (default), short or byte. To fill 32-bit memory words with a repeating byte pattern, specify a byte-sized pattern argument followed by the 'byte' keyword. To fill memory words with a repeating 16-bit pattern, specify a short-sized pattern followed by the 'short' keyword.

This follows the same rules as the write-memory command.

4.1.12 execute <address> <arg> <stackpointer>

Example: *execute 0x6000 0x21 0x1fff8400*

Jumps to code at the provided address and does not return to the bootloader. The system is returned to reset state prior to the jump. The function arg parameter is passed in R0 to the called code. The main stack pointer and process stack pointer registers are set to the stackpointer parameter, which can be zero. If set to zero, the code being called should set the stack pointer before using the stack.

The effective prototype of the called function is:

void function (uint32_t arg);

4.1.13 call <address> <arg>

Example: *call 0x6000 0x21*

This command invokes code at the provided address with the expectation that control returns to the bootloader.

The function that is called has the same prototype as for the one called by the execute command.

4.1.14 flash-security-disable <key>

Example: *flash-security-disable 0102030405060708*

Performs the flash security disable operation by comparing the provided 8-byte backdoor key against the backdoor key stored in the Flash Configuration Field at address 0x400 in flash.

If the backdoor key comparison fails, further attempts to disable security with this command fails until the system is reset.

Backdoor key access must be enabled by setting the KEYEN bitfield of the FSEC byte in the Flash Configuration Field to 0b10. It is disabled by default. The backdoor key in the Flash Configuration Field must also be set to a value other than all zeros or all ones.

4.1.15 flash-program-once <index> <byteCount> <data>

Example: *flash-program-once 0 4 0x01020304*

This writes provided data to a specific program once field.

Special care must be taken when writing to program once field. The program once field only supports programming once.

Any attempts to reprogram a program once field gets an error response. The number of bytes to be written must be 4-byte aligned for non-FAC fields, and be 8-byte aligned for FAC fields.

4.1.16 flash-read-once <index> <byteCount>

Example: *flash-read-once 0 4*

Returns the contents of a specific program once field.

4.1.17 flash-read-resource <addr> <byteCount> <option>

Example: *flash-read-resource 0 4 1*

Reads the contents of Flash IFR or Flash Firmware ID as specified by option and print result.

byteCount: The number of bytes to read and returns to the caller. Must be 4-byte aligned.

option: Indicates which area to be read. 0 means Flash IFR, 1 means Flash Firmware ID.

4.1.18 flash-image

Example: *flash-image*

The programs application image stored in the host flash to the target. The image should be SB, S-record, or Intel Hex format, and previously flashed into the fixed flash region of embedded host using the USB MSD interface. See Section 5.1, "Typical Usage", for how to download image file to the embedded host.

4.1.19 set-bus –b<bus> –f<frequency/baud rate>

Example: *set-bus –bspi –f100*

Sets current transfer bus and frequency/baud rate.

Options:

- *–b<bus>* Transfer bus type, ‘spi’, ‘i2c’, ‘uart’, or ‘can’, and default is ‘spi’.
- *–f<frequency/baud rate>* Bus frequency or baud rate, frequency should between 100 to 1000 KHz, default is 100. Baud rate should be 19200, 38400, 57600, 115200, or 230400, and the default is 57600. The FlexCAN speed should be 125, 250, 500, or 1000 KHz, and is automatically converted to the supported speed if the input is unsupported speed.

4.2 Programming Target Device

There are a few methods supported by the Embedded host in programming the image to the target device. The first method, mentioned in the previous chapter, is using the flash-image command on the command prompt interface. The second method is to use the buttons interface on the host platform. Both methods require the image to be present in the fixed flash region of the host platform. This can be easily done using the USB MSD interface of the embedded host, which is discussed first. The chapter "Typical Usage" provides step-by-step procedures in preparing the setup and programming the device using both methods.

After the programming is initiated, the embedded host performs the following operations to complete programming the target device:

1. Read from the flash region at start address 0x10000 and find the image data.
2. Determine the image format.
3. Parse and traverse the image to the record image start address and size for target.
4. Erase the target flash region for programming.
5. Parse and traverse image again. Read image records into buffer and set address, then flash the target with the Kinetis bootloader command.
6. Finish flashing the target.

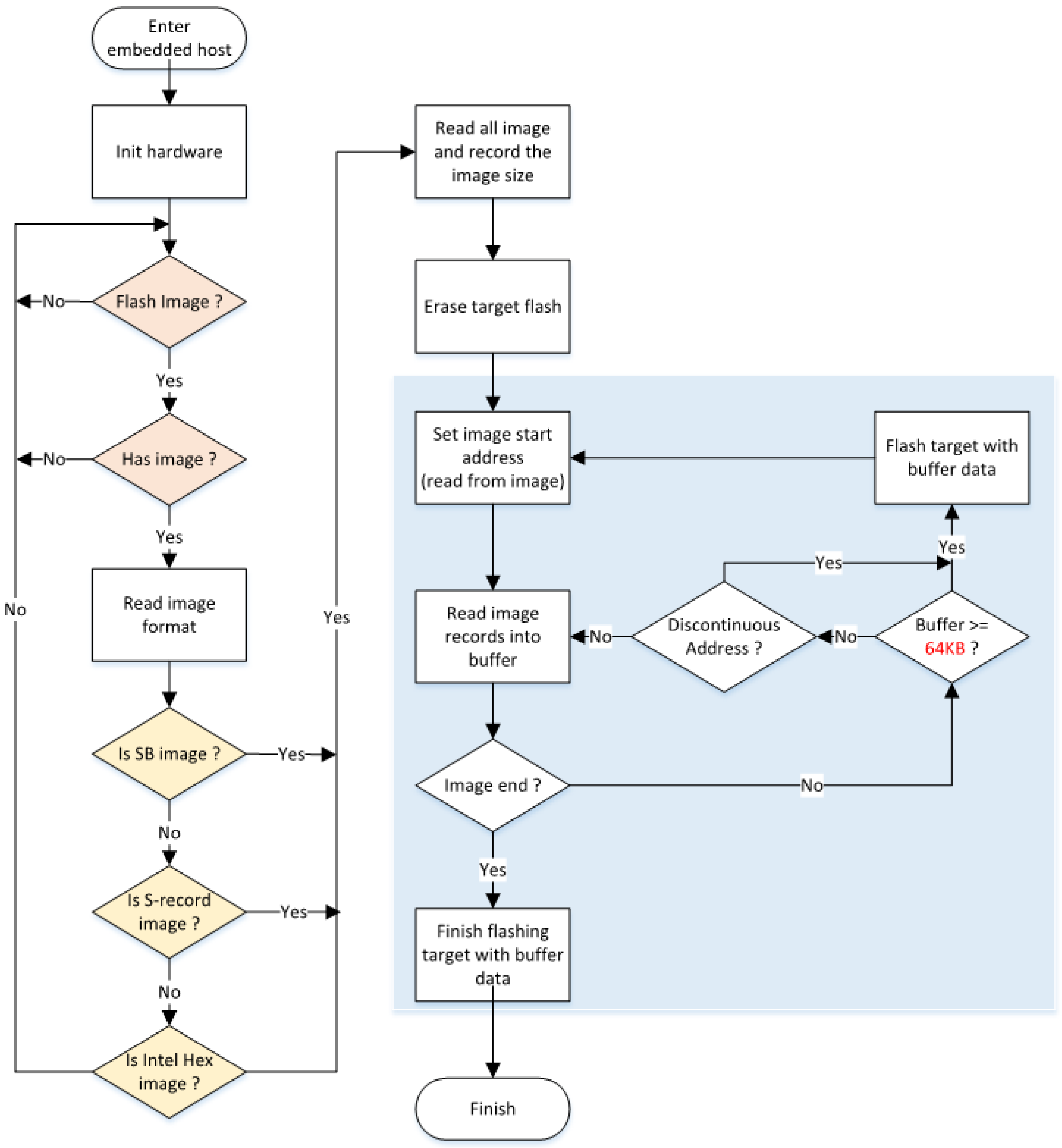


Figure 2. Lib folder

4.3 Downloading Image using USB MSD Interface

The first step is to download the firmware image in SB, S-record, or Intel Hex format to the embedded host.

Embedded host on TWR-K65F180M is equipped with 2 MB on-chip flash. After connecting USB cable to host computer, a thumb drive called “FSL_Loader” appears in Windows Explorer. The image file can drag-and-drop into “FSL_Loader” to end up at the internal flash memory location 0x10000.

NOTE

1. On drag-and-drop of the image on the USB MSD drive, embedded host automatically erases its own flash region from start address 0x10000.
2. The size of the image file supported should be smaller than 1 MB.

For all the steps involved, see the *Typical Usage* section below.

5 Typical Usage

This section describes the step-by-step usage of the Embedded host.

5.1 Pinmux configuration

Embedded host supports th efollowing peripherals.

Table 2. Embedded host peripheral pinmux

Peripheral	Port	Instance	Signal	Alt	Pinout for TER-ELEV	Comments
UART	PTE16	2	UART2_TX	ALT3		OpenSDA
	PTE17	2	UART2_RX	ALT3		OpenSDA
	PTE24	4	UART4_TX	ALT3	B29	
	PTE25	4	UART4_RX	ALT3	B30	
I2C	PTE18	0	I2C0_SDA	ALT4	A8	
	PTE19	0	I2C0_SCL	ALT2	A7	
SPI	PTD11	2	SPI0_PCS0	ALT2	B46	
	PTD12	2	SPI0_SCK	ALT2	B48	
	PTD13	2	SPI0_SIN	ALT2	B45	
	PTD14	2	SPI0_SOUT	ALT2	B44	
CAN	PTA30	0	CAN0_TX	ALT2	B42	Need to use pinout J7 on TWR_SET
	PTA31	0	CAN0_RX	ALT2	B41	

5.2 Boards setup

Table 3. TWR-K65F180M jumper setting

Jumper	Setting	Comments
J35	1-2 OFF	Disconnect I2CO with accelerometer.

Table continues on the next page...

Table 3. TWR-K65F180M jumper setting (continued)

Jumper	Setting	Comments
	3-4 OFF	
J23	1-2 OFF	Avoid USB regulator break issue.
	3-4 OFF	

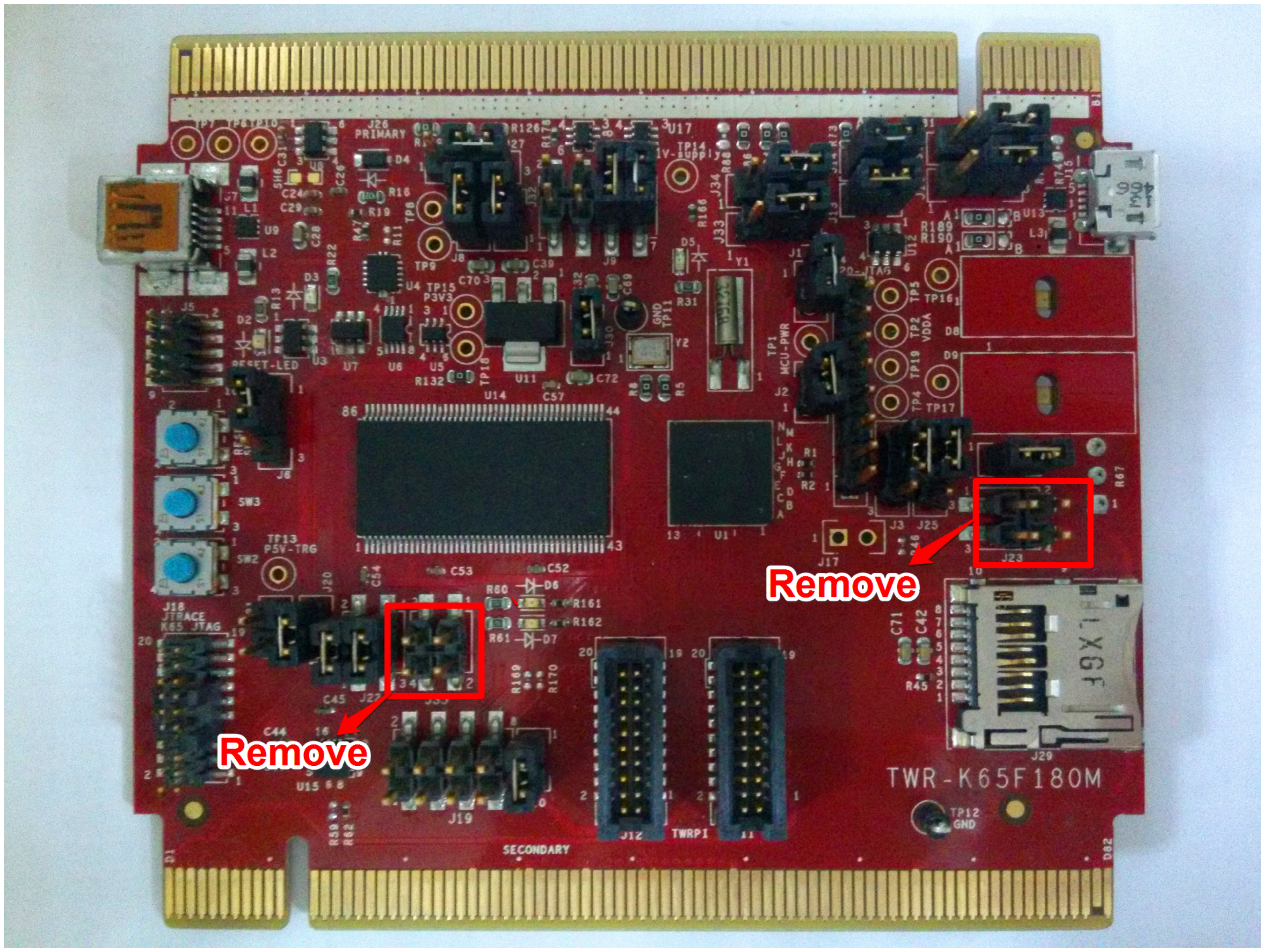


Figure 3. TWR-K65F180M jumper setting

Table 4. TWR-SER jumper setting

Jumper	Setting	Comments
J16	1-2 OFF	USB device mode
	3-4 OFF	
	5-6 OFF	

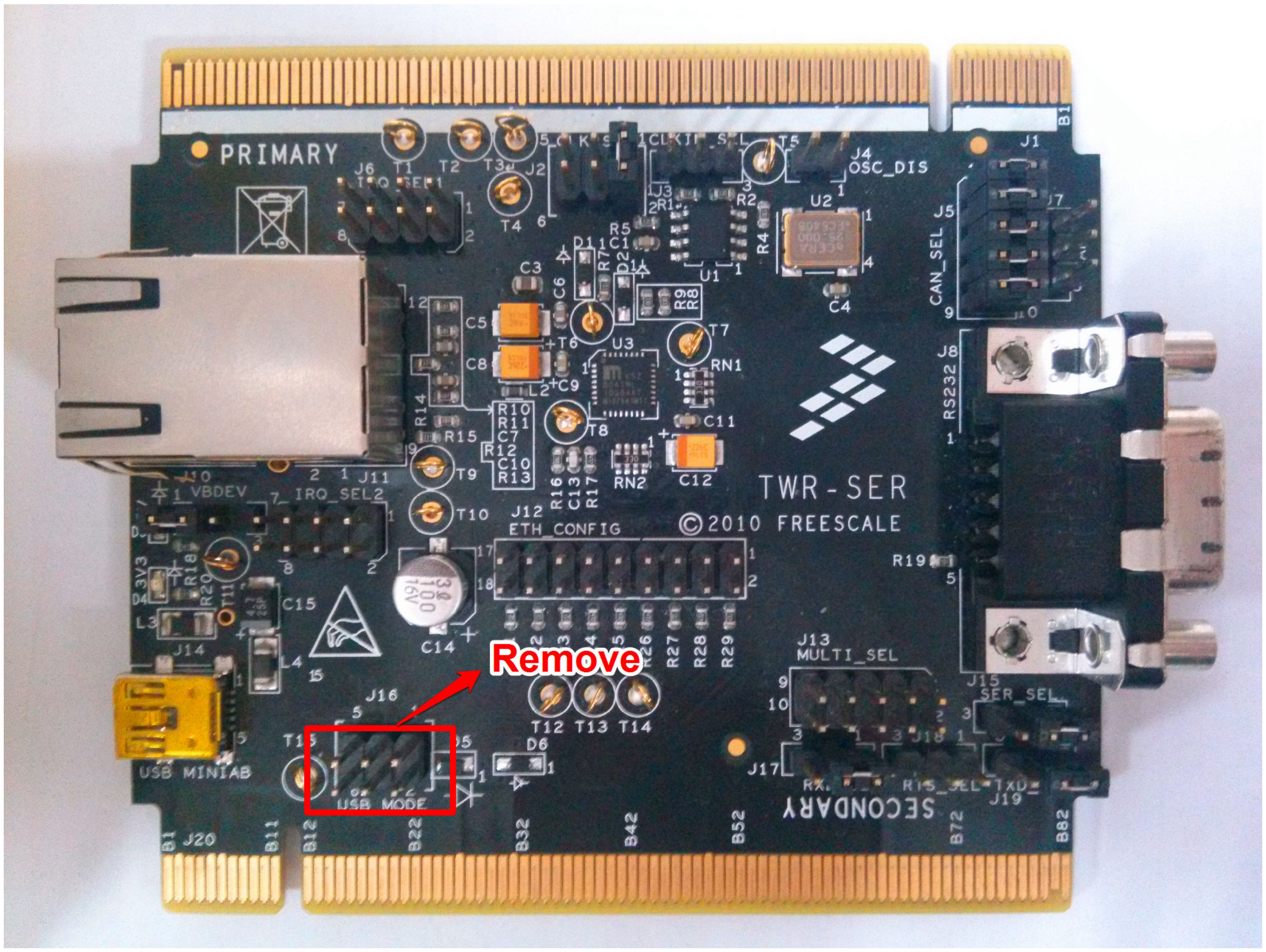
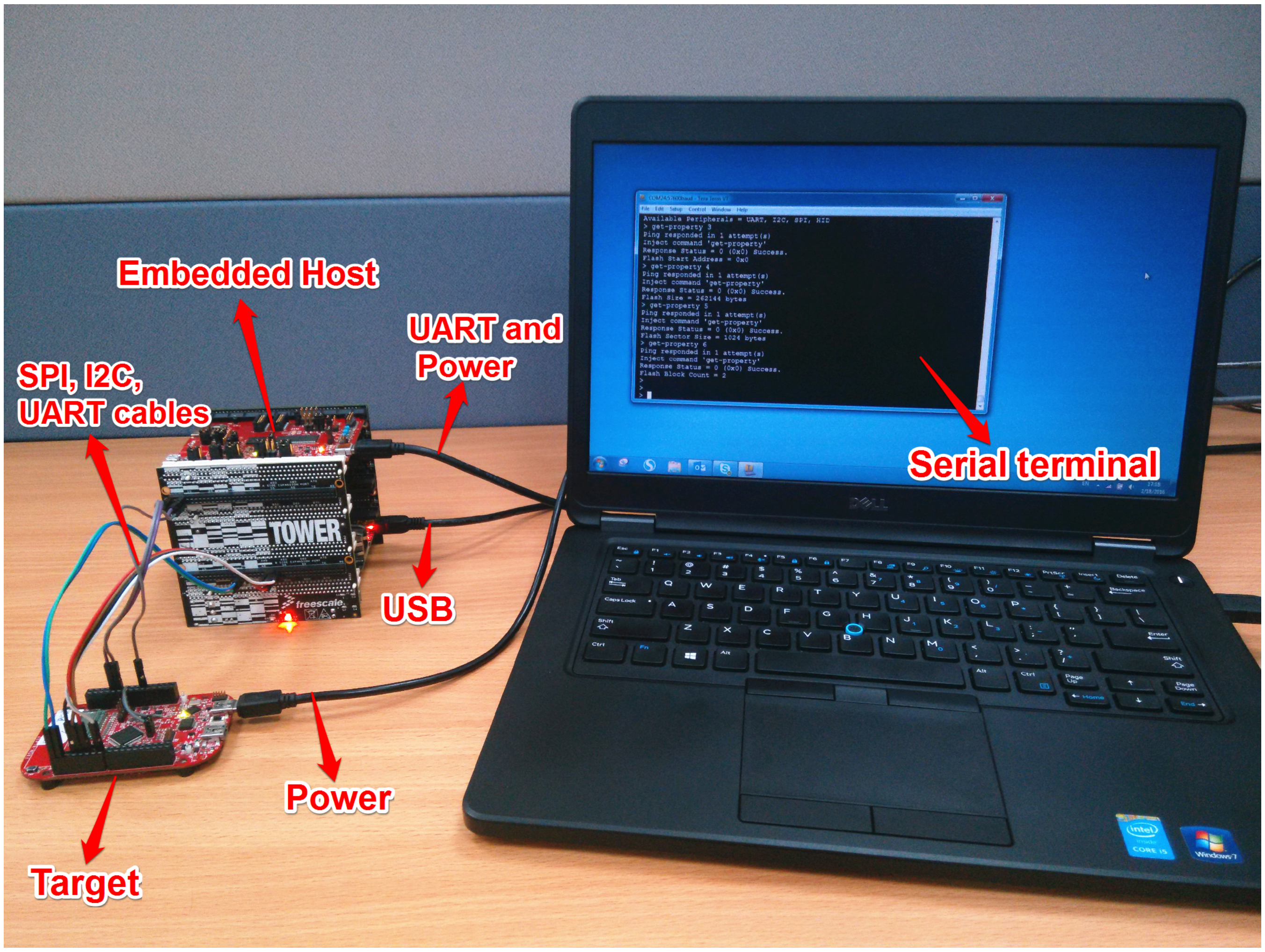


Figure 4. TWR-SER jumper setting

5.3 Command Prompt Interface

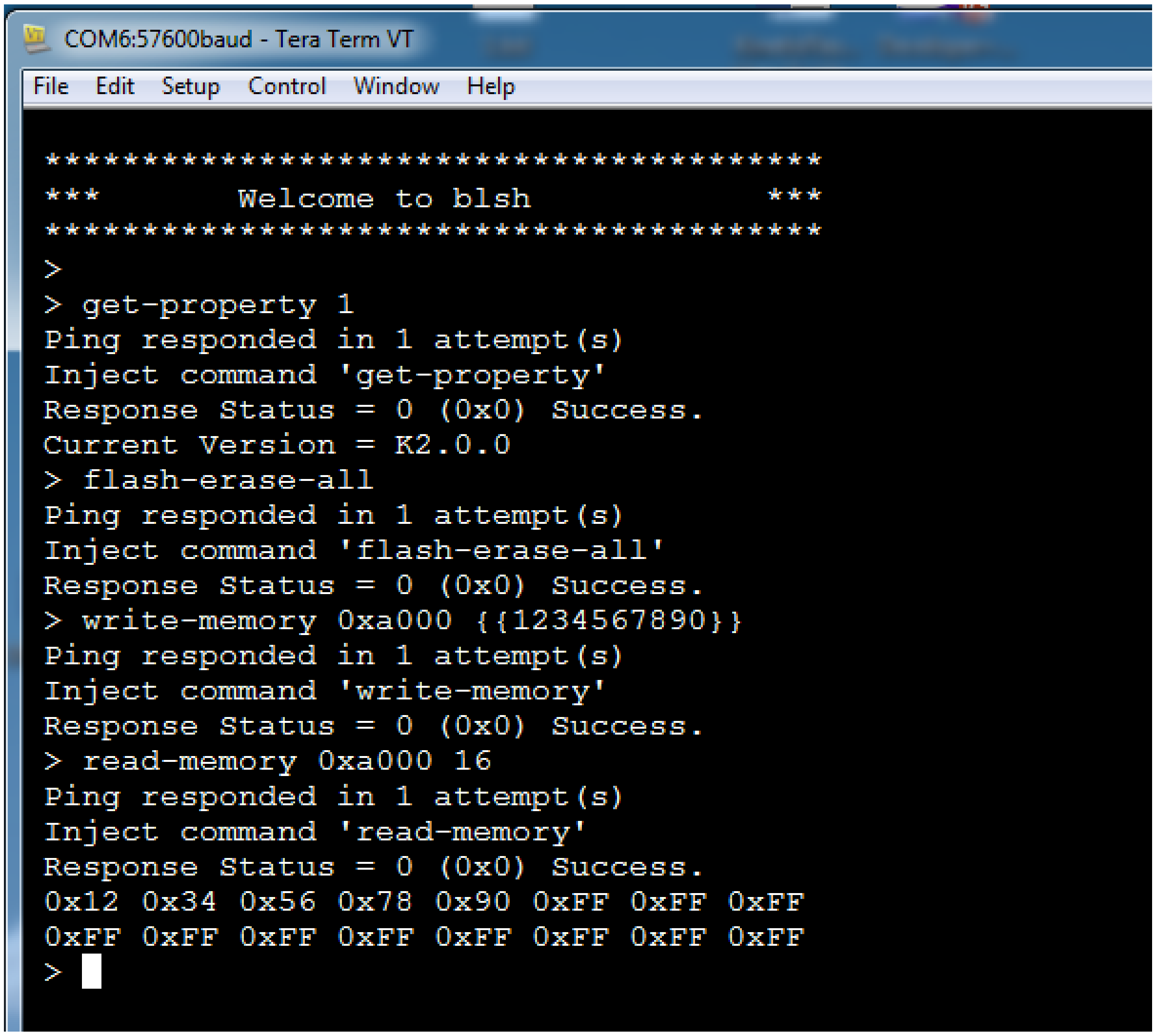
- Step 1: Connect SPI, I2C, UART, or CAN cable between the embedded host and target.
- Step 2: Connect UART cable between the embedded host and computer.



```
Available Peripherals = UART, I2C, SPI, HID
> get-property 3
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response Status = 0 (0x0) Success.
Flash Start Address = 0x0
> get-property 4
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response Status = 0 (0x0) Success.
Flash Size = 262144 bytes
> get-property 5
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response Status = 0 (0x0) Success.
Flash Sector Size = 1024 bytes
> get-property 6
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response Status = 0 (0x0) Success.
Flash Block Count = 2
>
>
>
```

Figure 5. Embedded host connection

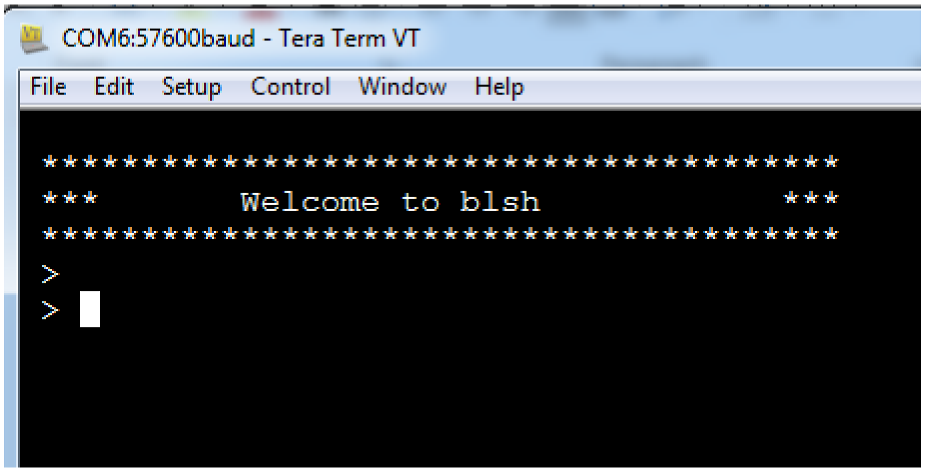
- Step 3: Open one of the serial terminal applications on your computer and open the correct port. Set the baud rate to 57600.
- Step 4: Press the 'Enter' key in the terminal, or the reset button on the TWR-K65 module to ensure the embedded host is running successfully and can receive echo in the terminal.


COM6:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
****************************************
***        Welcome to blsh           ***
****************************************
>
> get-property 1
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response Status = 0 (0x0) Success.
Current Version = K2.0.0
> flash-erase-all
Ping responded in 1 attempt(s)
Inject command 'flash-erase-all'
Response Status = 0 (0x0) Success.
> write-memory 0xa000 {{1234567890}}
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Response Status = 0 (0x0) Success.
> read-memory 0xa000 16
Ping responded in 1 attempt(s)
Inject command 'read-memory'
Response Status = 0 (0x0) Success.
0x12 0x34 0x56 0x78 0x90 0xFF 0xFF 0xFF
0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF 0xFF
>
```

Figure 6. Embedded host command prompt on the terminal window

- Step 5: Input any supported commands in the terminal and check the result.

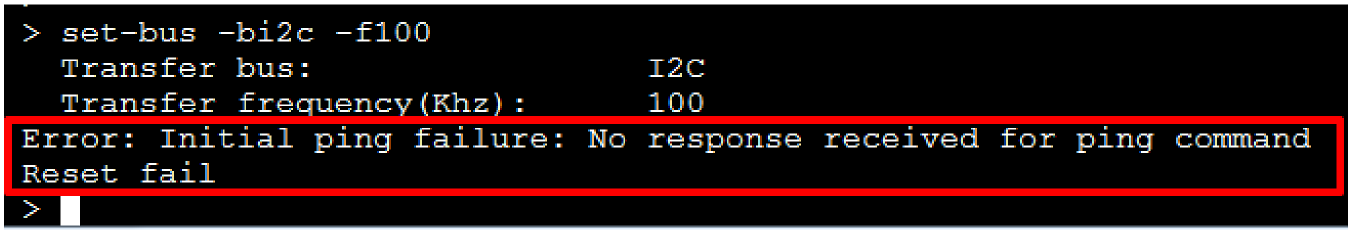

COM6:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
****************************************
***        Welcome to blsh           ***
****************************************
>
>
```

Figure 7. Embedded host commands

1. The default transfer but is SPI with 100 KHz frequency.

2. Input "set-bus –b<bus> –f<frequency/baud rate>" to change the bus or frequency. Embedded host resets the target with the current bus and change transfer bus, so there is no need to reset the target manually.
3. If the current transfer bus (default is SPI) is not available, the reset command fails. This error can be safely ignored.



```
> set-bus -bi2c -f100
  Transfer bus:                 I2C
  Transfer frequency(Khz):      100
Error: Initial ping failure: No response received for ping command
Reset fail
>
```

Figure 8. Error prompt when reset command fails

5.4 USB-HID interface

- Step 1: Connect the UART and USB cable between embedded host and your computer.
- Step 2: Hold down the SW3 button, and power the board by connecting host to PC via USB cable.
- Step 3: Release the SW3 button. Normally, a removable storage label can be seen on the PC as shown in the below figure.

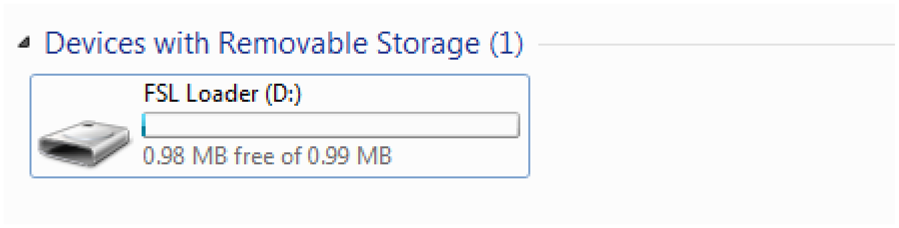
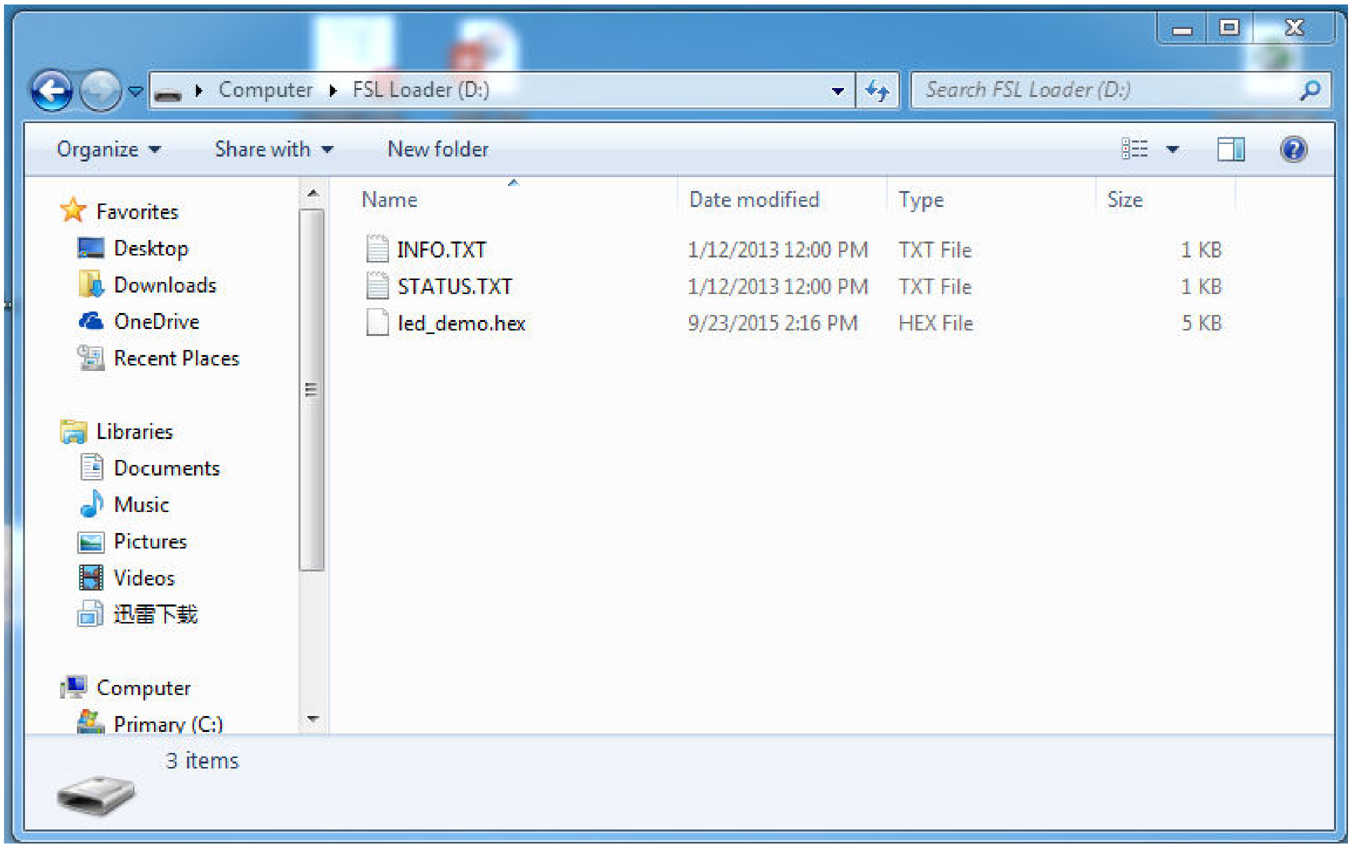


Figure 9. Removable storage

- Step 4: Open the removable storage and drag the image file for the target into it.



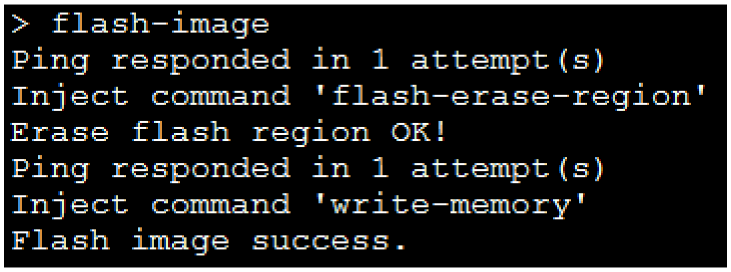
Name	Date modified	Type	Size
INFO.TXT	1/12/2013 12:00 PM	TXT File	1 KB
STATUS.TXT	1/12/2013 12:00 PM	TXT File	1 KB
led_demo.hex	9/23/2015 2:16 PM	HEX File	5 KB

Figure 10. Dragging image file

The next two sections describe the two methods available with the Embedded host to program the downloaded image to the target device.

5.5 Programming image using flash-image command

- Step 1 to 4: Follow section *Downloading image to embedded host flash* to download the image first.
- Step 5: Open the terminal and input command "flash-image". The host automatically identifies the type of image and flash it to the target.
- Step 6: Input the command "reset" or press the reset button for the target. The image runs after 5 seconds of waiting time.



```
> flash-image
Ping responded in 1 attempt(s)
Inject command 'flash-erase-region'
Erase flash region OK!
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Flash image success.
```

Figure 11. Flash image command

5.6 Programming image using buttons interface on board

Embedded host also provides transferring of image to the target device interactively, using the buttons available on the board and LED lights to provide the status.

- Step 1 to 4: Follow the section *Downloading image to embedded host flash* to download the image first.
- Step 5: Press SW3 to select transfer bus. The select options is displayed by lights:
  - LED_GREEN_OFF, LED_BLUE_OFF --> SPI
  - LED_GREEN_ON, LED_BLUE_OFF --> I2C
  - LED_GREEN_OFF, LED_BLUE_ON --> UART
  - LED_GREEN_ON, LED_BLUE_ON --> FlexCAN

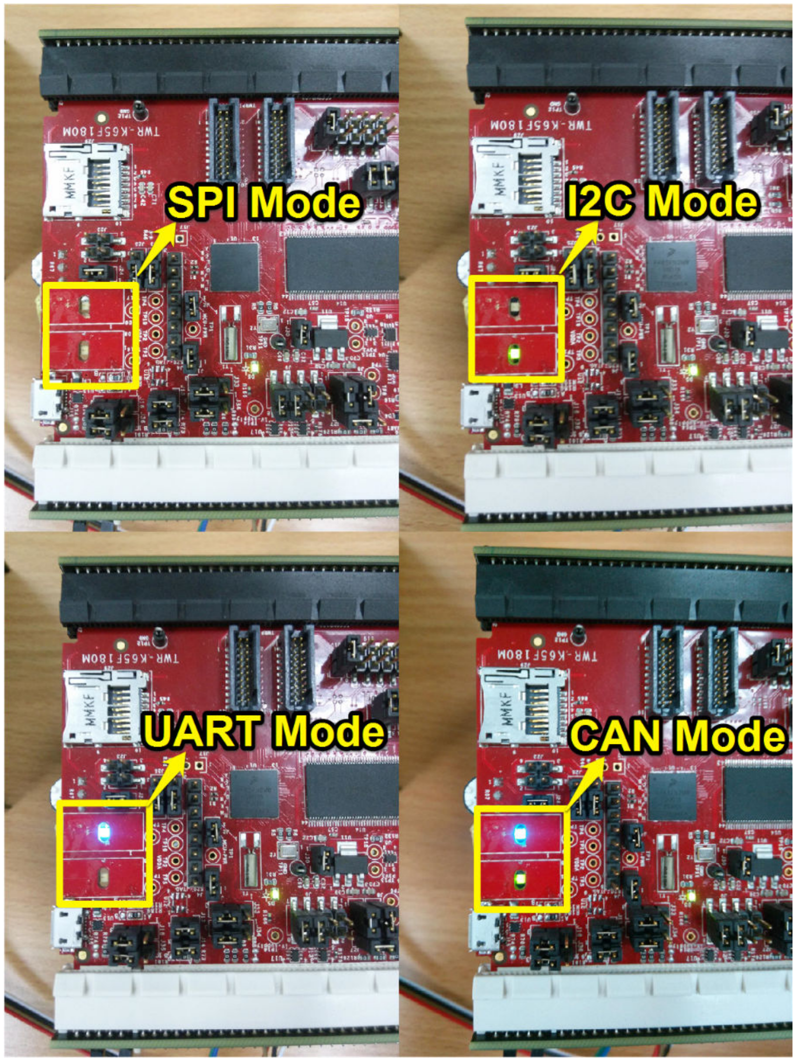


Figure 12. Light status with different transfer mode

- Step 6: Press SW2 to lock your selection. LED_YELLOW (D6) on the Tower module is on.
- Step 7: Press SW2 again if you would like to confirm the program target. If not, press SW3 to unlock the selection and reselect the transfer bus, following step 5.
- Step 8: If the image is programmed successfully to the target device, all lights (LED_GREEN, LED_BLUE, LED_YELLOW, and LED_RED) on the board are displayed as off. Otherwise, LED_RED (D7) glows.

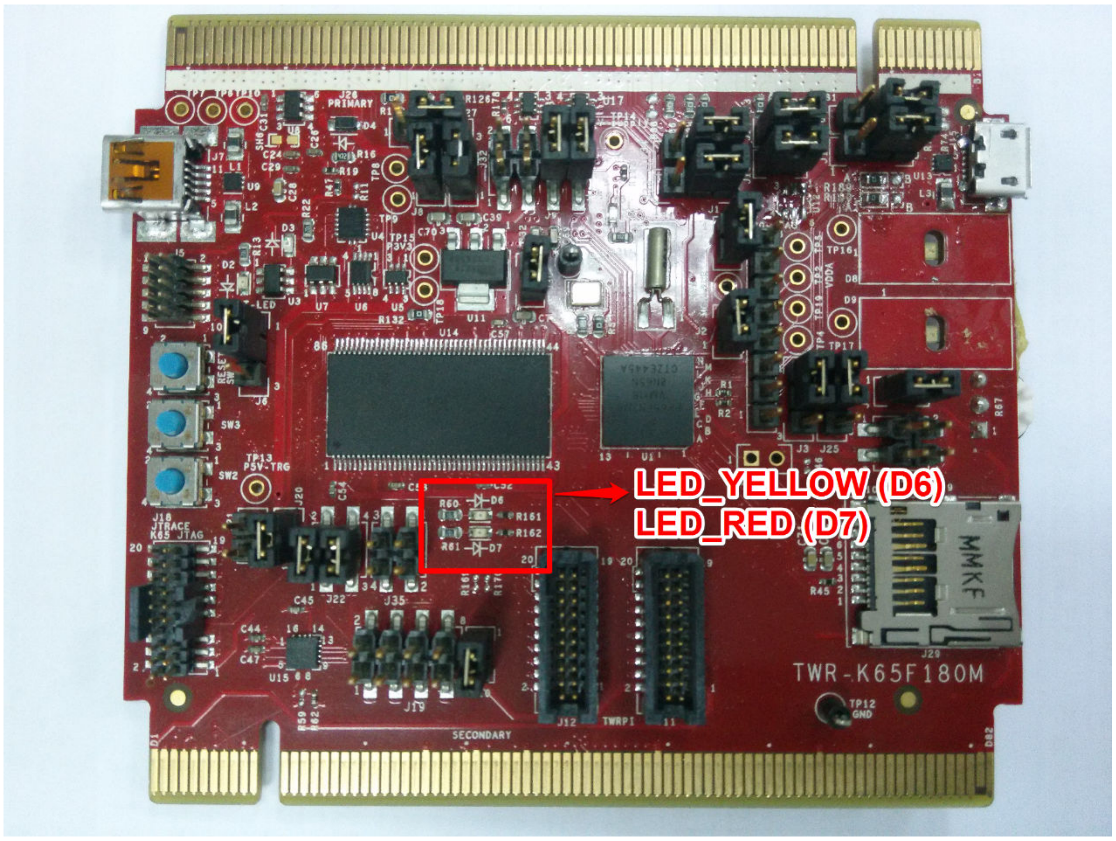


Figure 13. Indicator lights of programming result

6 Revision history

The following table contains a history of changes made to this document.

Table 5. Revision history

Revision number	Date	Substantive changes
0	04/2016	Kinetis bootloader v2.0.0 initial release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: EMBEDDEDHOSTUG
Rev. 0
04/2016