FTF FREESCALE TECHNOLOGY FORUM 2015

# Hands-On Workshop: Kinetis Bootloaders - How Do I Interface With It?

## FTF-DES-F1299

Eric Ocasio | Freescale Application Engineer
Michael Steffen | Freescale AMR FAE
Roy Wu | Freescale Technical Marketer

JUNE.2015

freescale™

External Use

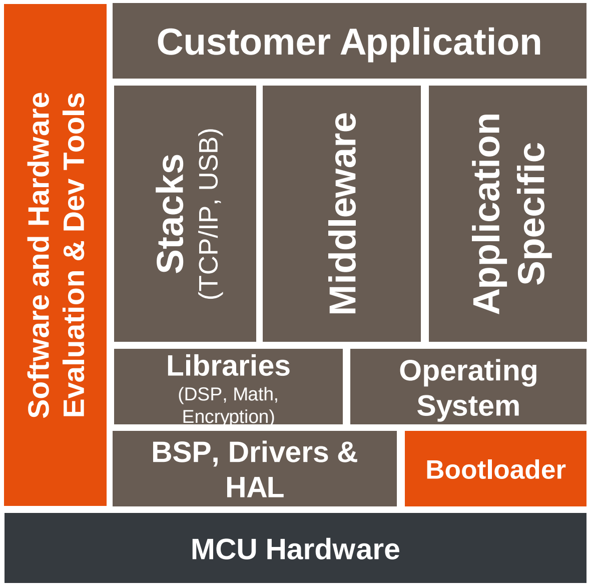
# Agenda

- What is Kinetis Bootloader?
- What is Supported? – Release Update
- Hands-on: Interface with Flash-Resident Bootloader
- Hands-on: Interface with ROM Bootloader
- Hands-on: Bootloader Minimal Configuration
- Q&A

# Kinetis Bootloader

Flash programming over a serial connection: erase, program, verify

Flash, ROM or RAM based bootloader with open-source software and host-side programming utilities.

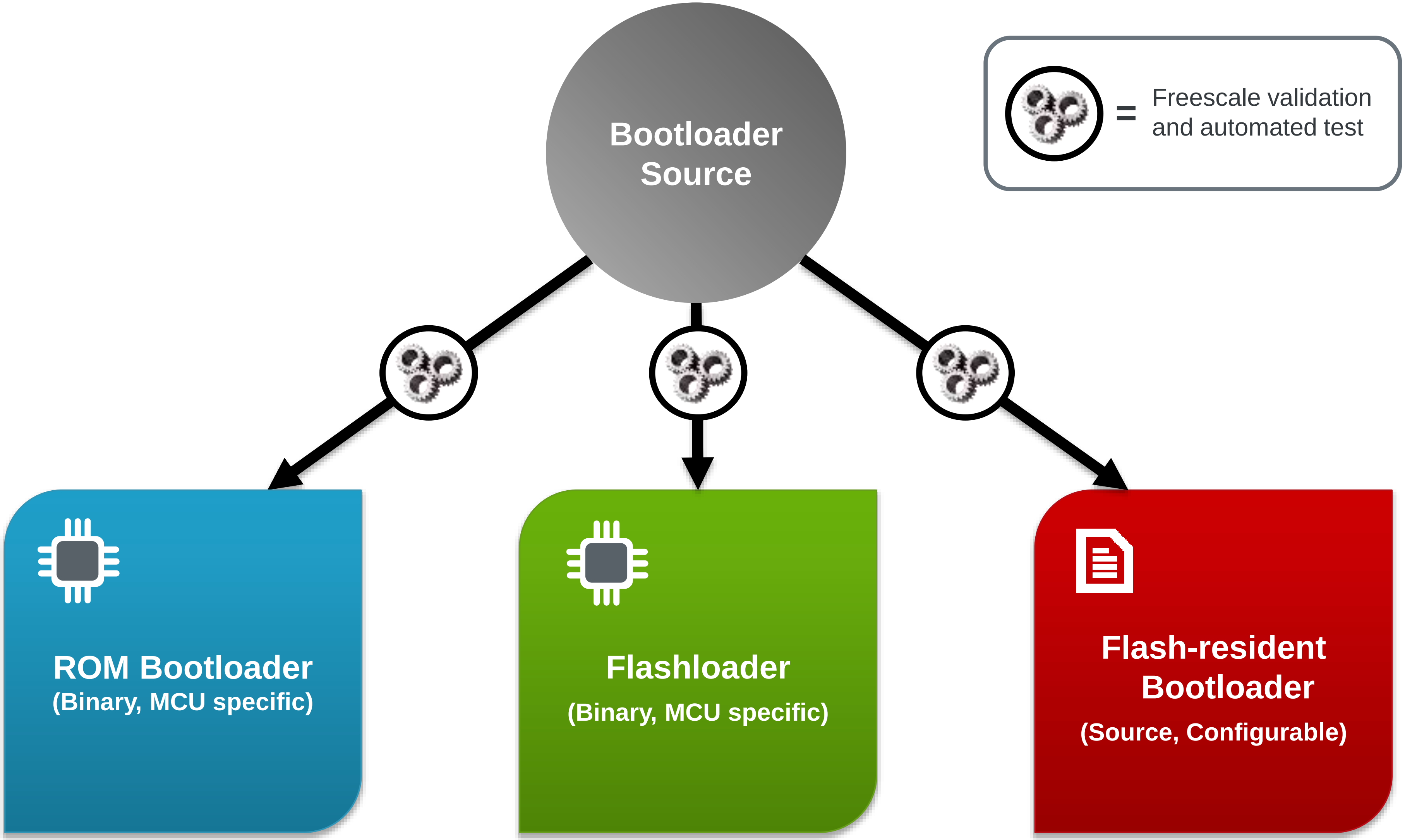


## Product Features

- A common bootloader for Kinetis MCUs
- C/C++ Source code provided under a permissive BSD open source license
- Serial communications with a host via UART, SPI, I2C, CAN and USB HID
  - Active peripheral detection
  - Common packet-based protocol for all peripherals
- Packet error detection and retransmission
- Configurable options for executing bootloader at startup or application runtime
- Command-line and GUI tools provided
- Designed to be flash, ROM or RAM resident
  - Failsafe boot mechanism on Kinetis devices with ROM
  - Pre-programmed into flash (on devices without a dedicated ROM) and executed from RAM for built-in factory programming
  - Fully customizable for use in customer applications providing reliable field updates

The OSI logo trademark is the trademark of Open Source Initiative.

# Kinetis Bootloader Configurations



Bootloader Source

= Freescale validation and automated test

ROM Bootloader
(Binary, MCU specific)

Flashloader
(Binary, MCU specific)

Flash-resident Bootloader
(Source, Configurable)

# Entering the Bootloader

## ROM-based Bootloader

- Bootloader is executed out of reset if either of these are true:
  - FOPT[BOOTSRC_SEL] = 0b11 or 0b10
    - FOPT is a Flash Option byte at 0x40D in flash — flash erased state defaults to 0b11
  - BOOTCFG0 pin is enabled and asserted (FOPT[BOOTPIN_OPT] = 0b0)
- A user application may execute the bootloader by calling the bootloader entry point (can be determined programmatically)

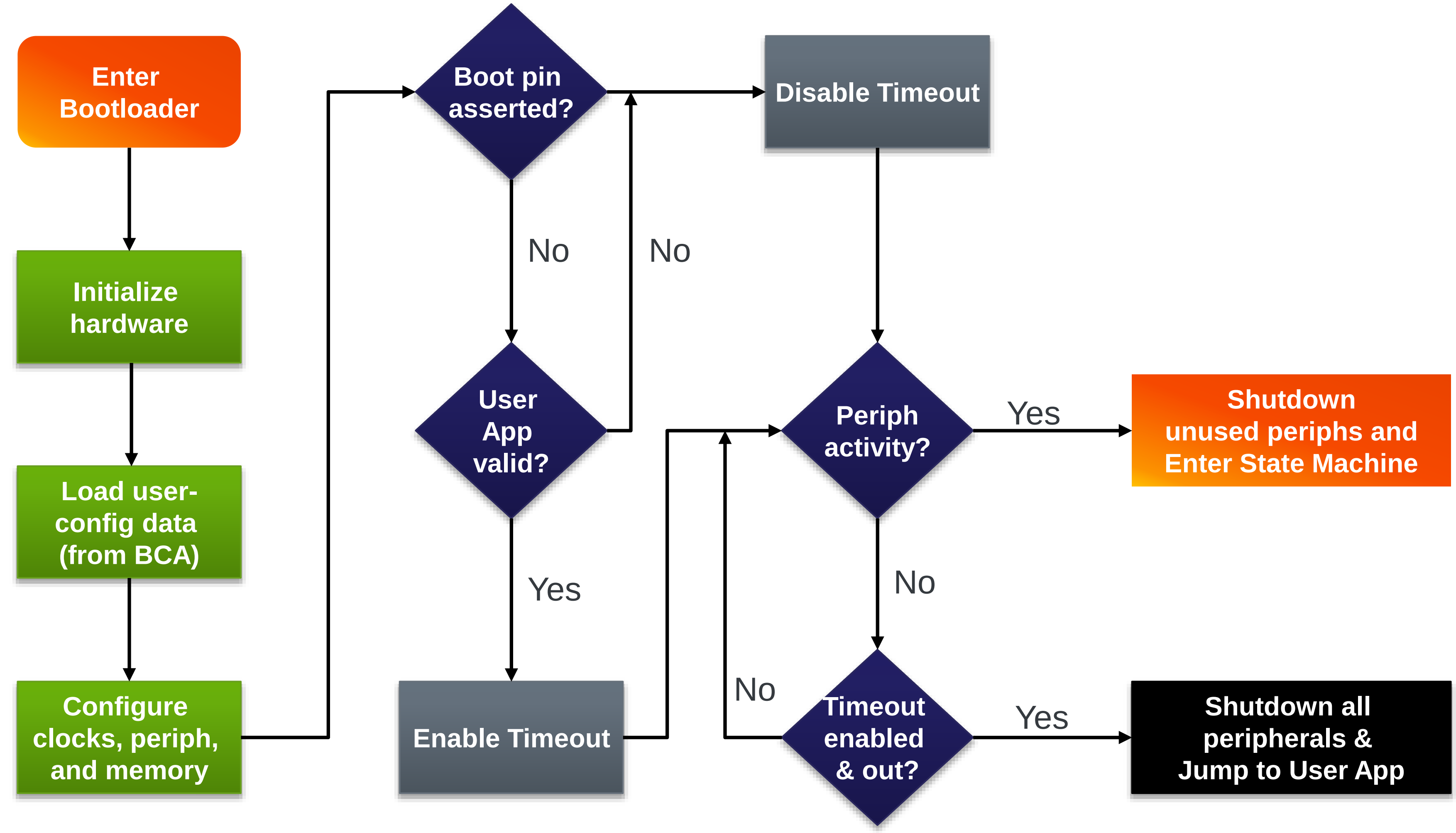
## Flash-based Bootloader

- Completely configurable by the user
- Bootloader is executed out of reset if the Reset Vector points to the bootloader
- is_boot_pin_asserted() can be customized to utilize a user-selected GPIO pin
- A user application may execute the bootloader by calling the bootloader entry point (can be determined programmatically)

## RAM-based Bootloader (Flashloader)

- Always executes out of reset
- Is intended as a "one-shot" flash programming method

# Bootloader Startup Flow (ROM/Default Flash)


Enter Bootloader
Initialize hardware
Load user-config data (from BCA)
Configure clocks, periph, and memory
Boot pin asserted?
No
User App valid?
No
Yes
Enable Timeout
Disable Timeout
Periph activity?
Yes
Shutdown unused periphs and Enter State Machine
No
Timeout enabled & out?
No
Yes
Shutdown all peripherals & Jump to User App

# Kinetis Bootloader Release Schedule

**July, 2014**

FRDM-K64F

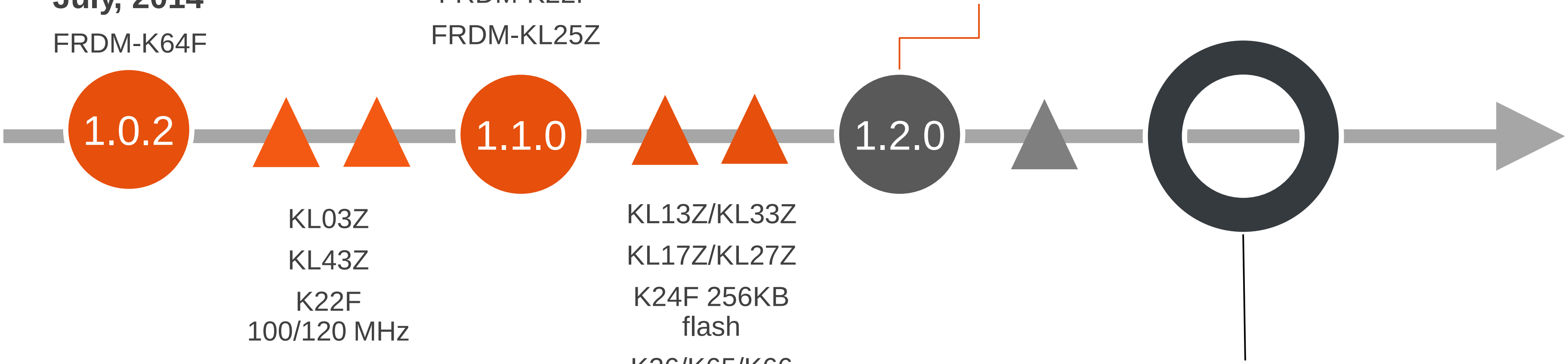
**Dec, 2014**

FRDM-K22F

FRDM-KL25Z

**1.2.0 –** July, 2015

- Add support for K24F120M, K65F, KV11, KV31, KV46
- Add support for CAN interface
- Add KDS project examples
- Add Mac OS/Linux support for command-line host tool



KL03Z

KL43Z

K22F 100/120 MHz

…

KL13Z/KL33Z

KL17Z/KL27Z

K24F 256KB flash

K26/K65/K66

…

**Security Release –** Q4 2015

- Encrypted image download support in select devices
- Security tool to generate a encrypted .sb image
- Updated host tool with commands to program encrypted image
- QSPI flash support

# Kinetis Bootloader v1.2.0 Release — What's New?

## Peripheral Support

- UART
- USB-HID
- I2C Example
- SPI Example
- CAN Example

## Toolchain Project

- IAR
- KDS

## Host Tool Update

- Command-line support Windows XP/7/8/8.1, Mac OS and Linux
- Enhanced GUI Tool

## New Devices Support*

- K26F, K65F and K66F MCUs via the TWR-K65F180M
- K24F 120MHz 256KB flash MCUs via the TWR-K24F120M
- KV11Z MCUs via the FRDM-KV11Z and TWR-KV11Z128M
- KV31Z MCUs via the TWR-KV31F120M
- KV40Z, KV43Z, KV44Z and KV46Z MCUs via the TWR-KV46F150M

# All Supported Devices

- Flash-resident Bootloader Project/Source Code
  - K24F, K63F and K64F MCUs via the FRDM-K64F and TWR-K64F120M
  - K22F 100MHz/120MHz and K02F MCUs via the FRDM-K22F and TWR-K22F120M
  - KL25Z MCUs via the FRDM-KL25Z and TWR-KL25Z48M
  - K26F, K65F and K66F MCUs via the TWR-K65F180M
  - K24F 120MHz 256KB flash MCUs via the TWR-K24F120M
  - KV11Z MCUs via the FRDM-KV11Z and TWR-KV11Z128M
  - KV31Z MCUs via the TWR-KV31F120M
  - KV40Z, KV43Z, KV44Z and KV46Z MCUs via the TWR-KV46F150M

- ROM Bootloader
  - KL03Z, KL13Z and KL33Z MCUs
  - KL17Z, KL27Z and KL43Z MCUs

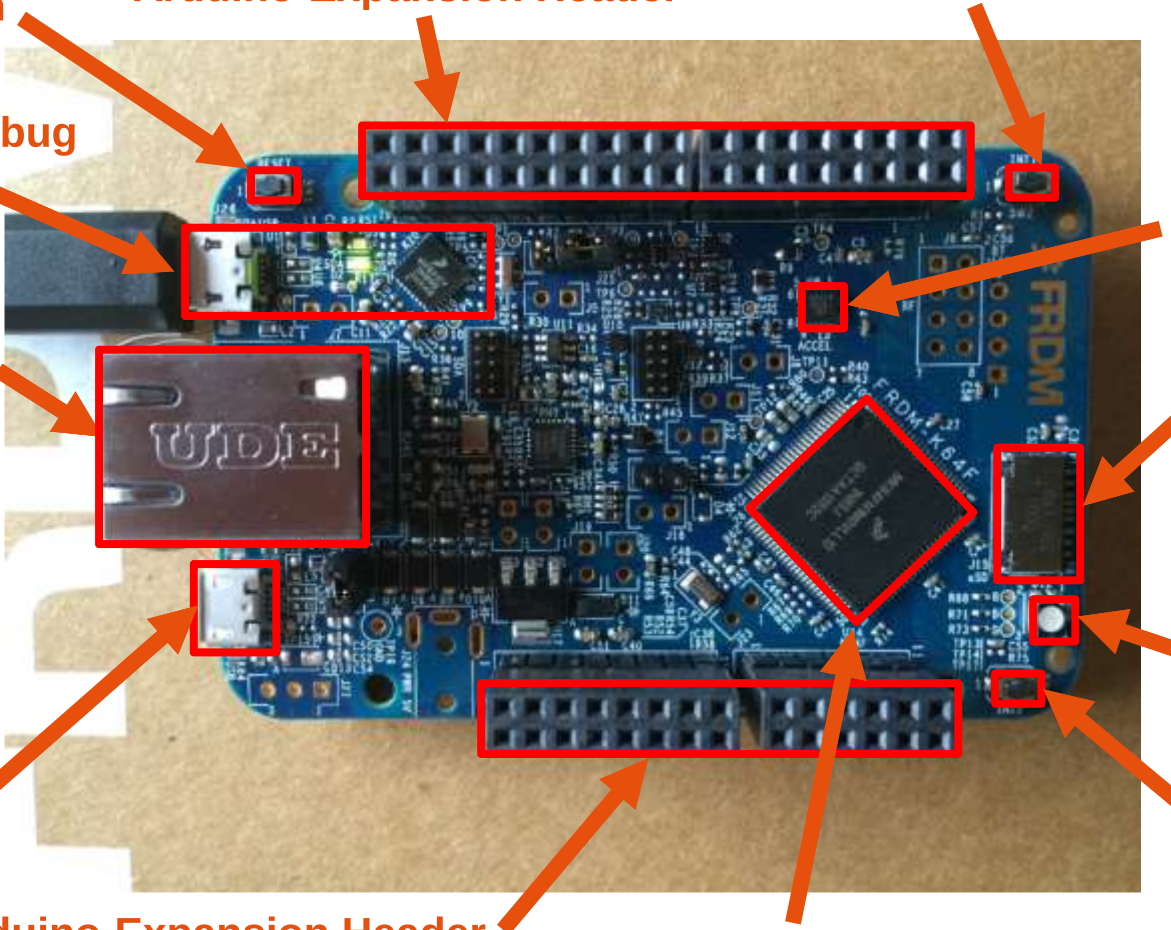
- Flashloader
  - K02F and K22F 100MHz/120MHz MCUs,
  - K24F 256KB flash MCUs
  - K26F, K65F and K66F MCUs
  - KV30, KV31 MCUs

# Hands-on: Lab #1 – Program the FRDM-K64F board with Freescale's Flash based Bootloader

✓ **Lab objective 1 -** Learn how to program the FRDM-K64F board with the latest on-line available bootloader.

✓ **Lab objective 2 –** Download via USB HID using the "Kinetis Updater" and program an application to flash the red LED.

# FRDM-K64F Hardware Overview



Reset Button

Arduino Expansion Header

Push Button

OpenSDA Debug

FXOS8700CQ
Accel + Mag

K64 Ethernet

MicroSD Card

Tri-Color LED

K64 USB

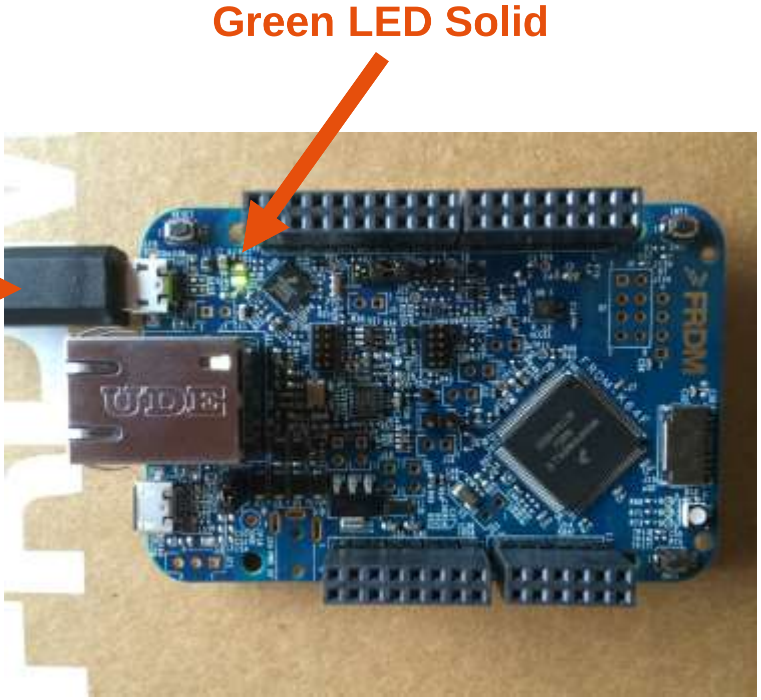
Push Button

Arduino Expansion Header

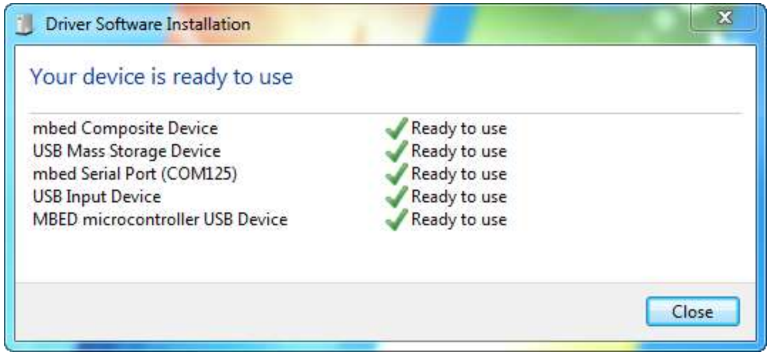
MK64FN1M0VLL12
120 MHz, Cortex-M4, 1MB Flash,
256K SRAM, Ethernet, USB, 16-bit ADC,
12-bit DAC, SDHC

# Setting up FRDM-K64F Board

- Plug-in the USB cable (OpenSDA port)



- Green LED should be solid
- Let the drivers install…you should see this similar message before continuing the lab.

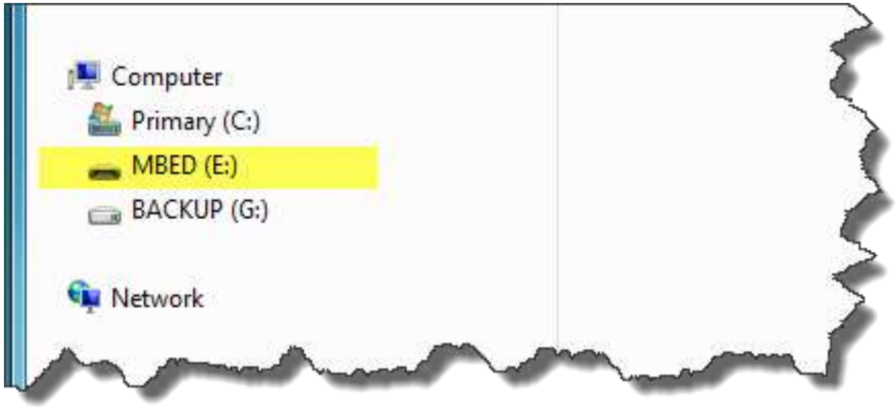


Your device is ready to use

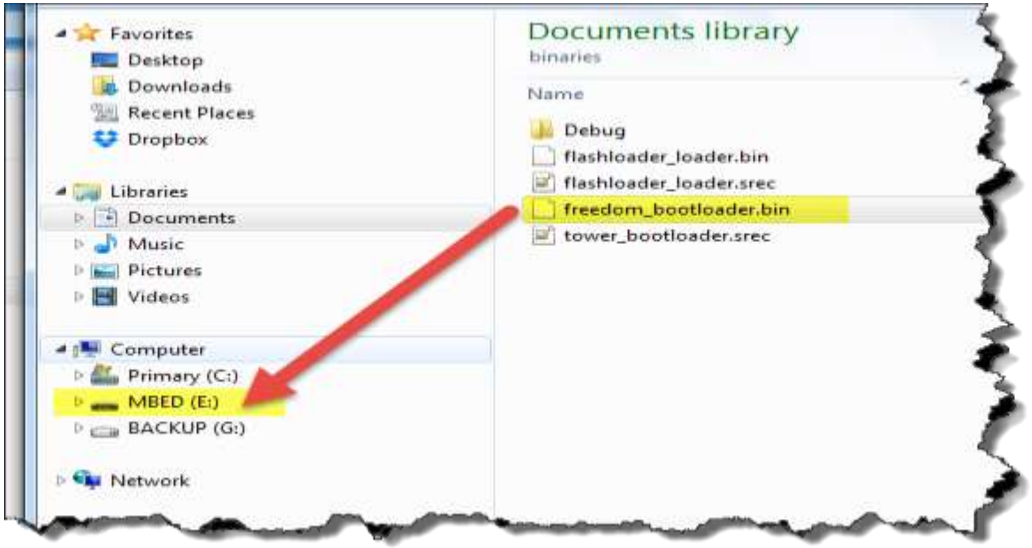
```
mbed Composite Device                     ✓ Ready to use
USB Mass Storage Device                   ✓ Ready to use
mbed Serial Port (COM125)                 ✓ Ready to use
USB Input Device                          ✓ Ready to use
MBED microcontroller USB Device           ✓ Ready to use
```

# Loading Bootloader Firmware on FRDM-K64F Board

- You should now see the MBED (drive:) appear

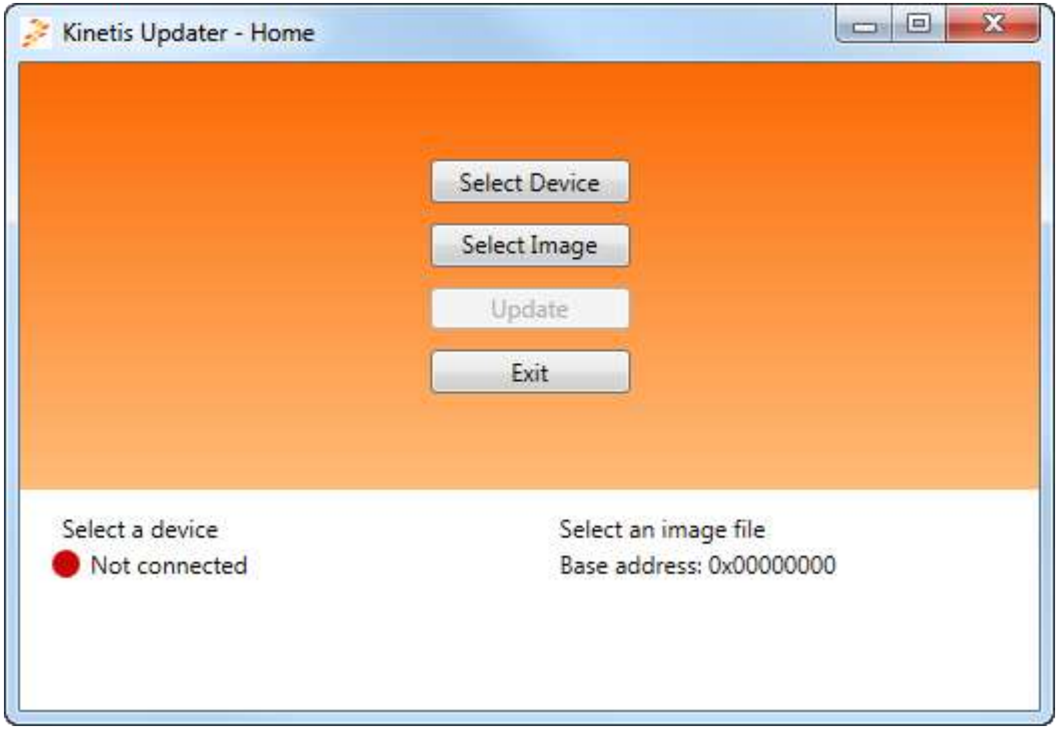


- Located the file called "freedom_bootloader.bin", located here: *C:\FTF-DES-1299\Kinetis_Bootloader_1_2_0\targets\MK64F12\binaries*
- Drag and drop the file "freedom_bootloader.bin" onto MBED(drive:)



# Connecting to the “Kinetis Updater”

- Open the program KinetisUpdater.exe located at: *C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win\KinetisUpdater*

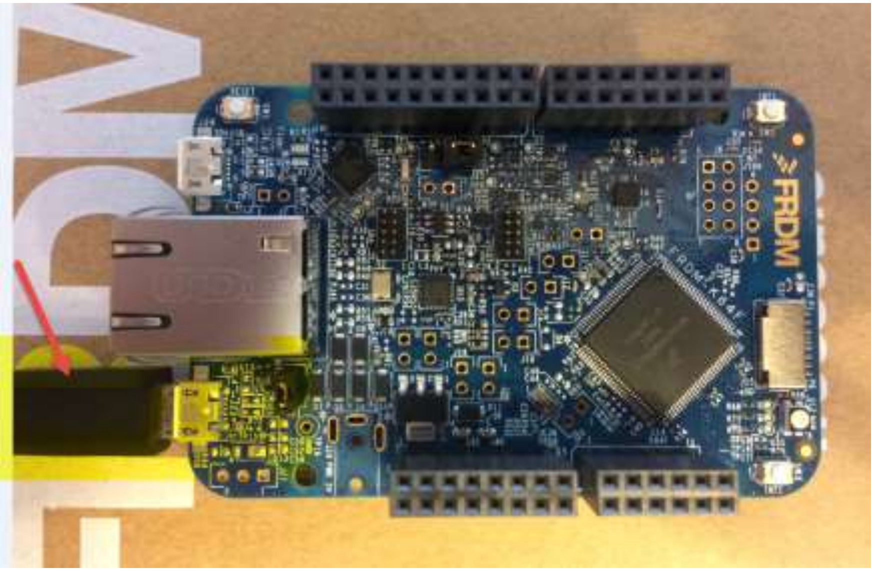


Select a device
● Not connected

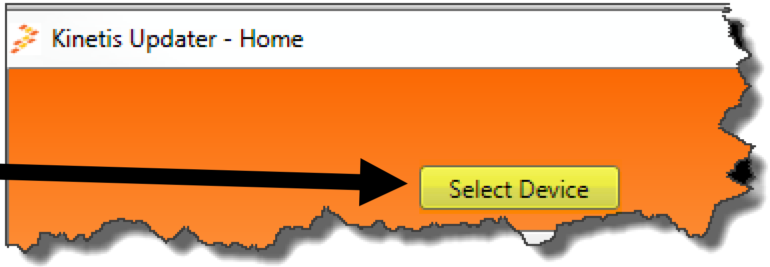
Select an image file
Base address: 0x00000000

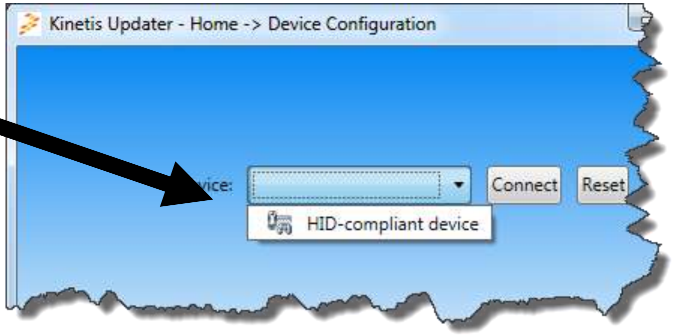
# Connecting to the “Kinetis Updater” – using HID

- Move the USB micro cable from the top USB (OpenSDA) connector to the bottom USB connector (J22). We are now communicating with the USB port on the K64F MCU directly.



- On the “KinetisUpdater” GUI, follow these steps:
  1. Press the “Select Device”
  2. Drop down the “Device” box and select “HID-compliant device”

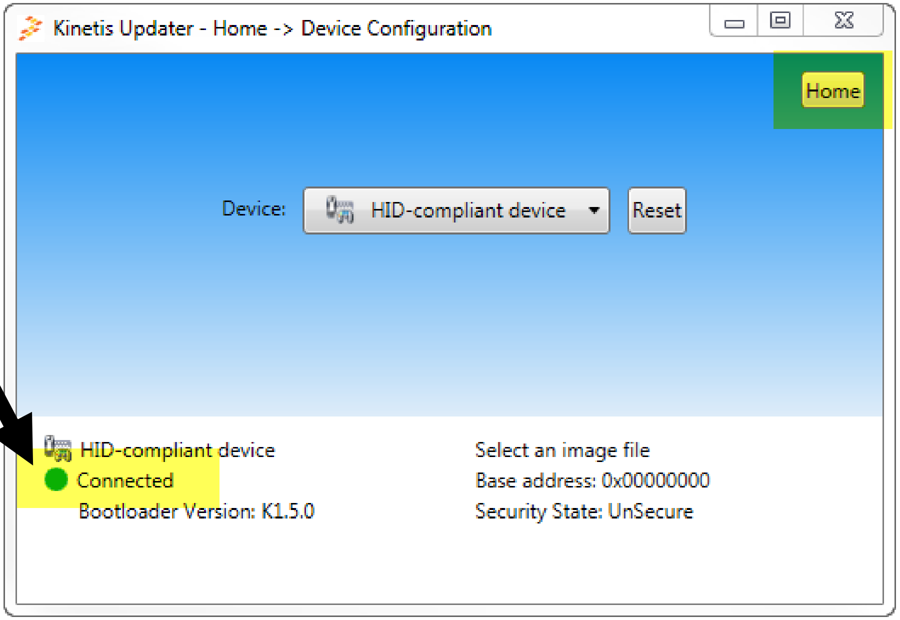




# Programming an Application using the “Kinetis Updater”

- You should now see the “Kinetis Update GUI is now “Connected”.

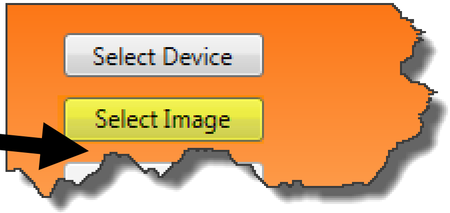
  Do not continue the lab if the green dot does not appear connected.


Kinetis Updater - Home -> Device Configuration

HID-compliant device
● Connected
Bootloader Version: K1.5.0

Select an image file
Base address: 0x00000000
Security State: UnSecure

- Press the “Home” in the upper right corner. Home
- On the “KinetisUpdater” GUI, follow these steps:
  1. Press the “Select Image”
  2. Press the “Browse” and locate this file:
     ***C:\Kinetis_Bootloader_1_2_0_rc1.1\Kinetis_Bootloader_1_2_0\apps\led_demo\ binaries* led_demo_FRDM-K64F_a000.bin**
  3. You should have this .bin file appear as the “Image file:”



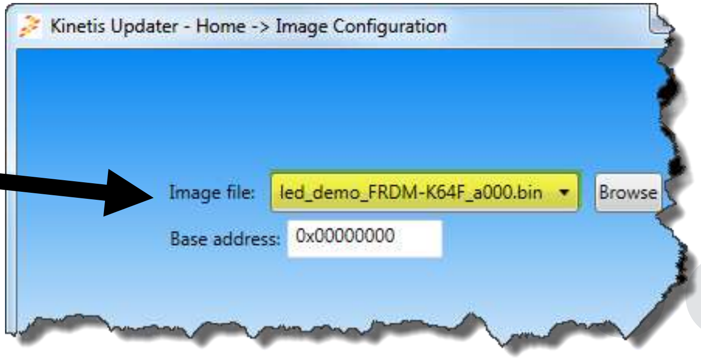
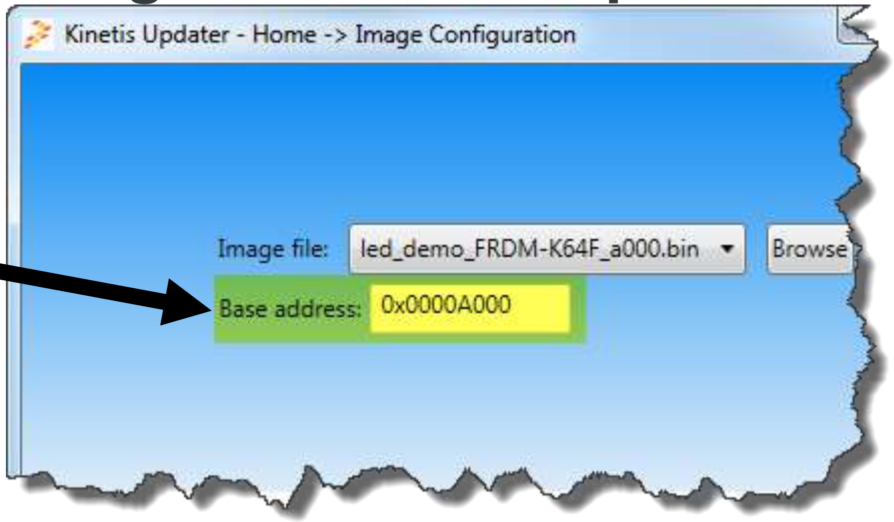

Kinetis Updater - Home -> Image Configuration

Image file: led_demo_FRDM-K64F_a000.bin

Base address: 0x00000000

# Programming an Application using the “Kinetis Updater”

3. Change the “Base address” to 0x0000A000



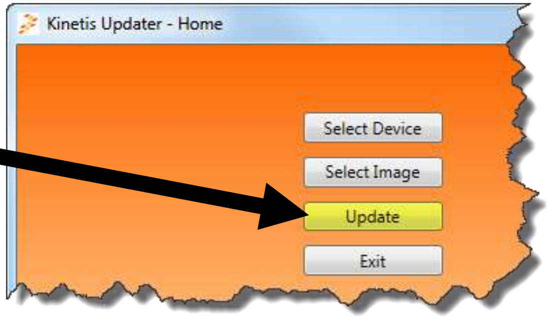
Kinetis Updater - Home -> Image Configuration

Image file: led_demo_FRDM-K64F_a000.bin

Base address: 0x0000A000

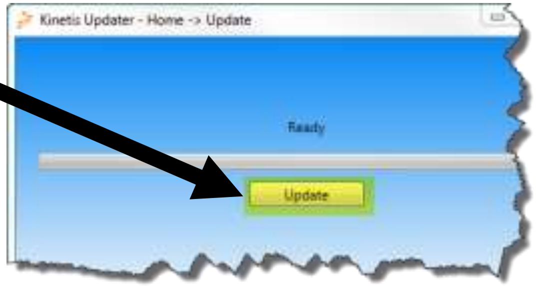
3. Press the “Home” Button in the upper right corner. Home

3. Press the “Update” button



Kinetis Updater - Home

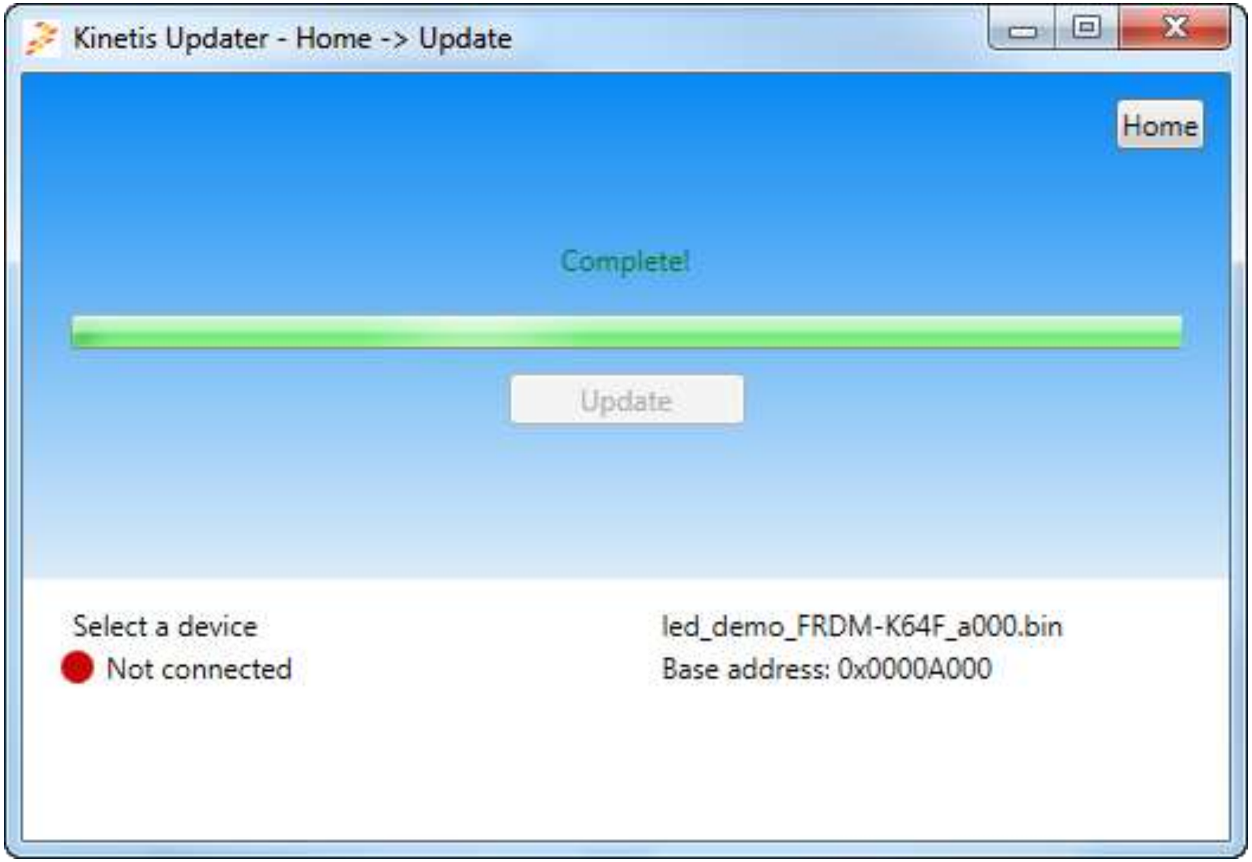
4. Press the “Update” on the next screen



Kinetis Updater - Home -> Update

Ready

# Programming an Application using the “Kinetis Updater”



Complete!

Select a device

● Not connected

led_demo_FRDM-K64F_a000.bin

Base address: 0x0000A000

- The Red/Blue LED should be flashing in the board, and the “Kinetis Updater” GUI should be disconnected now.
- Please close the GUI and proceed to next lab.

# Hands-on: Lab #2 – How to interface with the FRDM-KL03Z board with Freescale's ROM based Bootloader

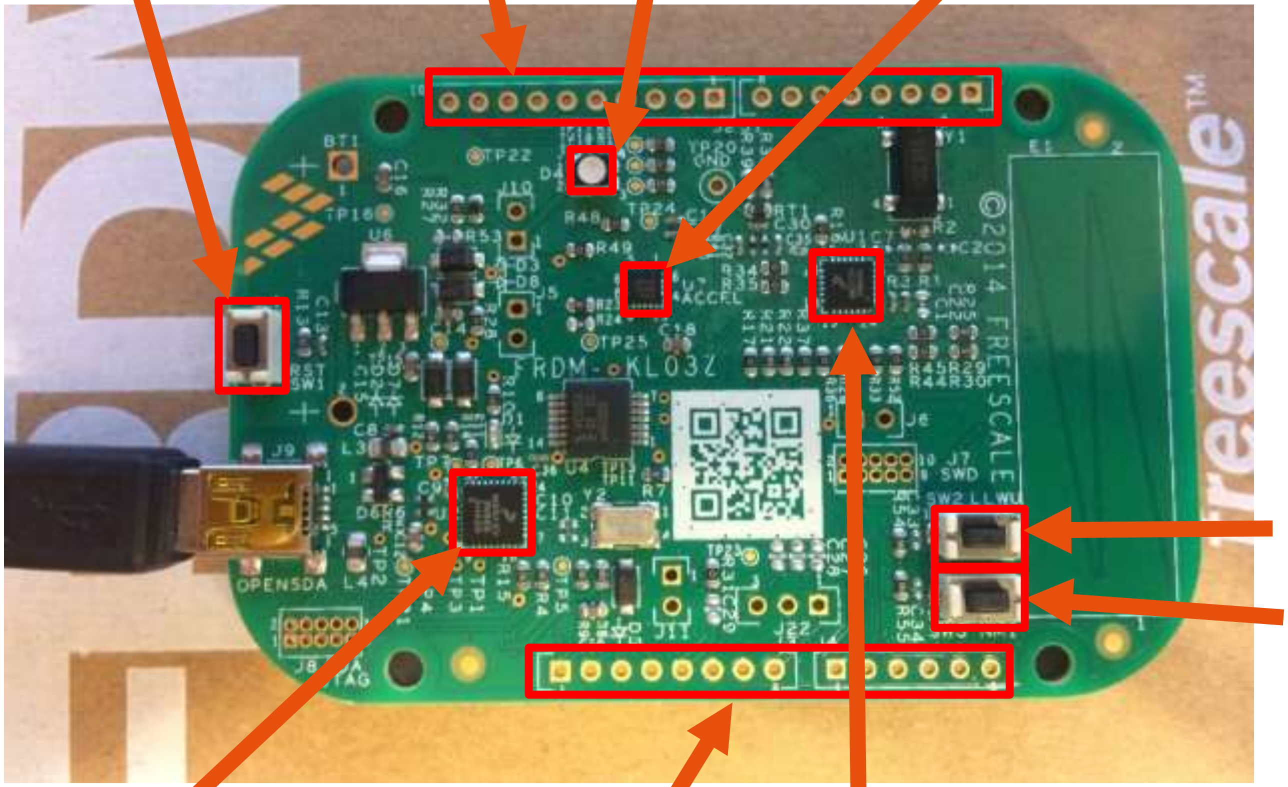
- ✓ **Lab objective 1 -** Learn how to interface with the FRDM-KL03Z board's built-in ROM based bootloader.

- ✓ **Lab objective 2 –** Download via UART using "Command line" to ping, erase, and program an application to flash the red LED.

- ✓ **Lab objective 3 –** Understand how the FOPT (flash option register ) settings are used to gain entry into the bootloader.

- ✓ **Lab objective 4 –** Understand how gain entry into the bootloader from user application code.

# FRDM-KL03Z Hardware Overview



Arduino Expansion Header

Reset Button

Tri-Color LED

MMA8491 Accelerometer

SW2 Push Button

SW3 Push Button

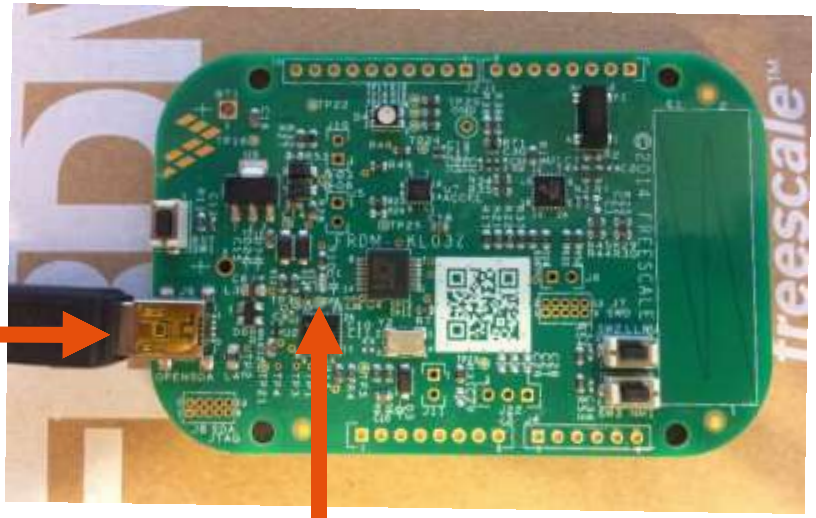
OpenSDA Debug

Arduino Expansion Header

MKL03Z16VFK4
48 MHz, Cortex-M0+, 16KB Flash, 2K SRAM, Low Power, ROM based Bootloader

# Setting up FRDM-KL03Z Board

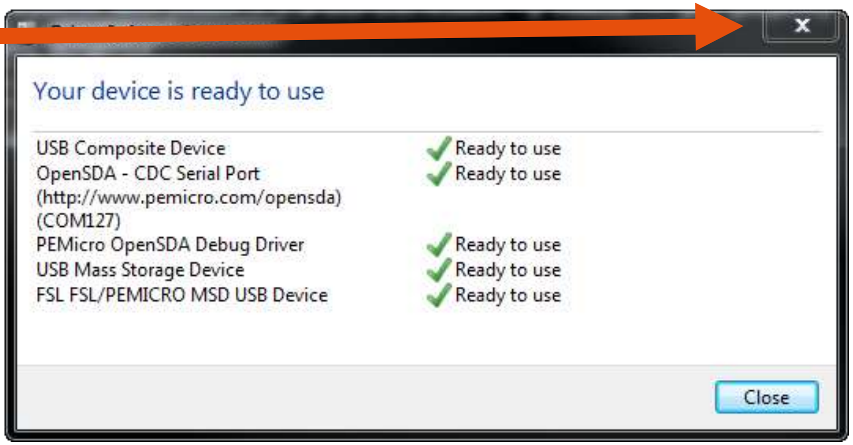
- Plug-in the USB cable (OpenSDA port)



USB Connection

Green LED Solid

- Green LED should be solid
- Let the drivers install…you should see this similar message before continuing the lab. NOTE: Before you close this window, write down the COM port #.  In the example to the right, it's COM127.

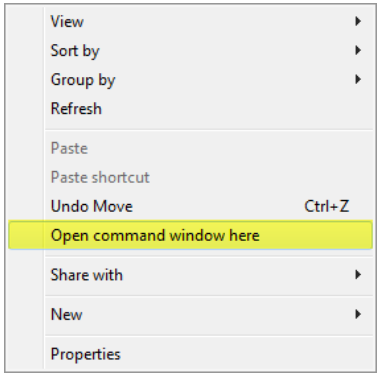


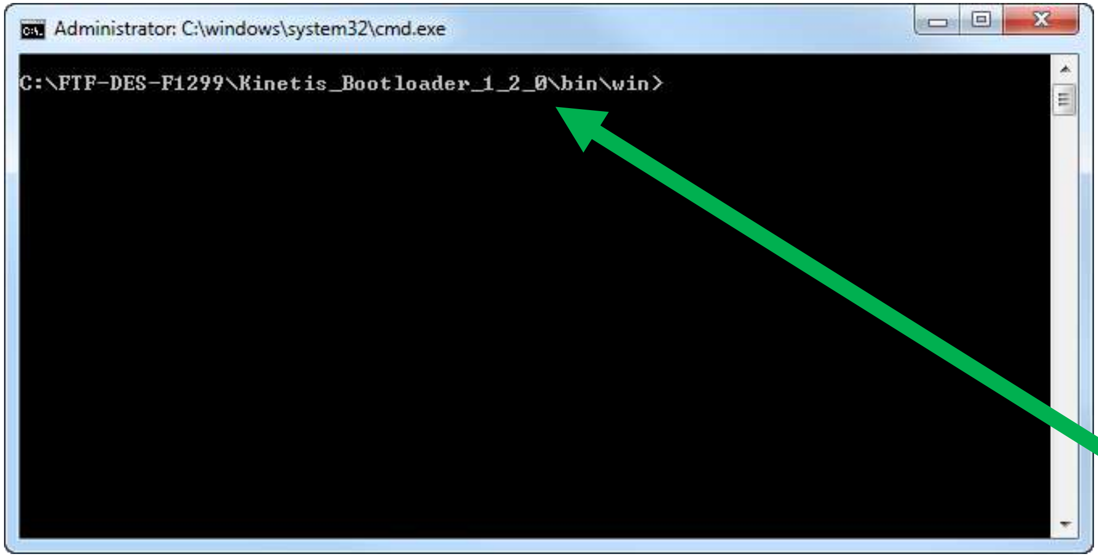
Your device is ready to use

```
USB Composite Device                         ✓ Ready to use
OpenSDA - CDC Serial Port                    ✓ Ready to use
(http://www.pemicro.com/opensda)
(COM127)
PEMicro OpenSDA Debug Driver                 ✓ Ready to use
USB Mass Storage Device                      ✓ Ready to use
FSL FSL/PEMICRO MSD USB Device               ✓ Ready to use
```

# Loading Bootloader Firmware on FRDM-KL03Z Board

- Open Windows Explorer and migrate to this directory in Windows, *C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin*
- Open a terminal window in this location by holding down the SHIFT key + mouse right click



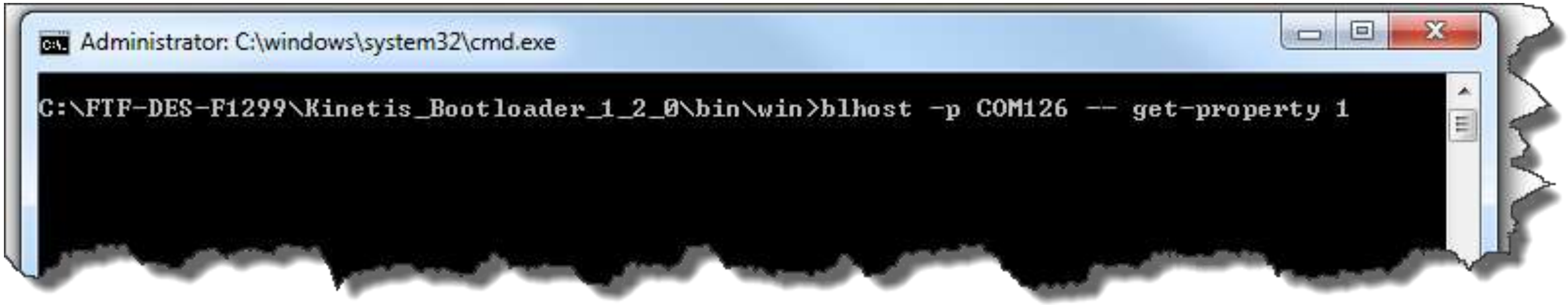


```
C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>
```

- You should have this CMD window in this directory:

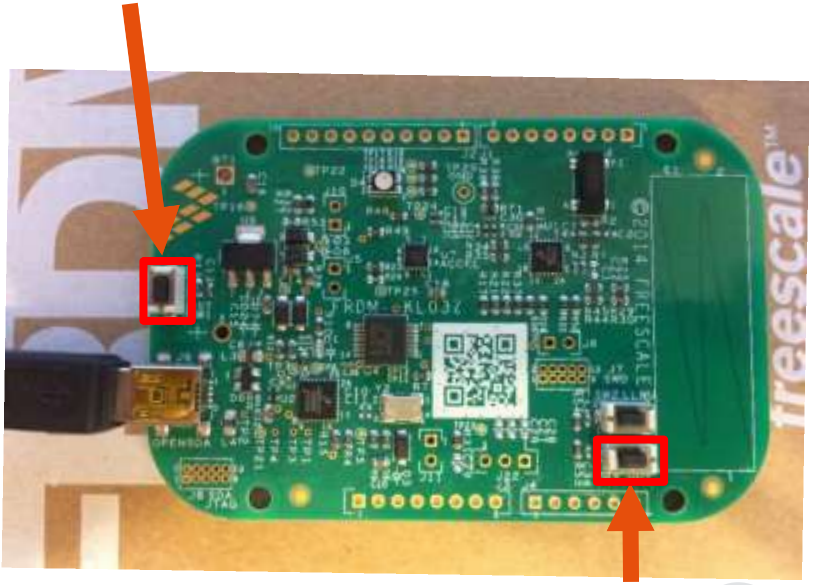
# Communicating via UART to the KL03 Bootloader

- In the command window, type this command (don't press ENTER yet):
  blhost –p com<#> -- get-property 1

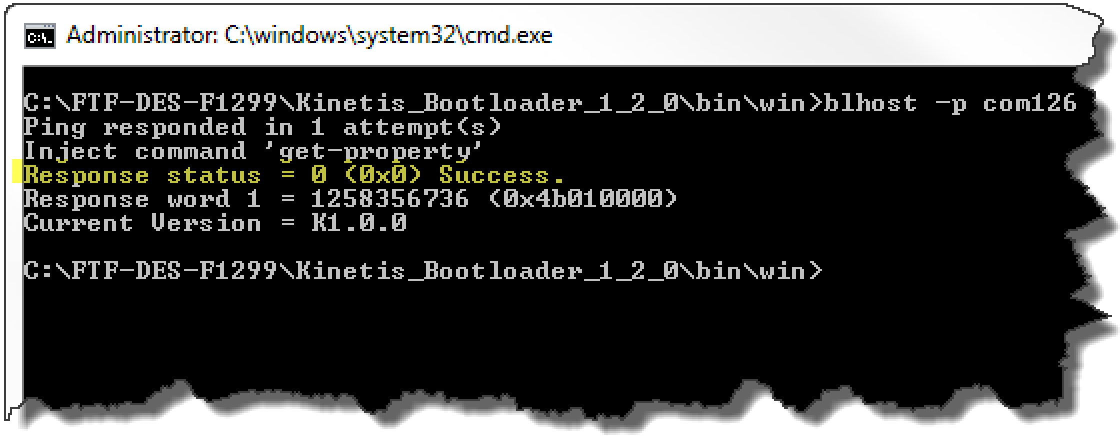


```
C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>blhost -p COM126 -- get-property 1
```

- Hold down the "RESET Switch" and the "SW3 Switch", then release "RESET Switch", and then the release "SW3 Switch".
- Now press **ENTER** in the CMD window.



# Communicating via UART to the KL03 Bootloader



Administrator: C:\windows\system32\cmd.exe

```
C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>blhost -p com126
Ping responded in 1 attempt(s)
Inject command 'get-property'
Response status = 0 (0x0) Success.
Response word 1 = 1258356736 (0x4b010000)
Current Version = K1.0.0

C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>
```

- You should see this message: Response status" = 0(0x0) Success

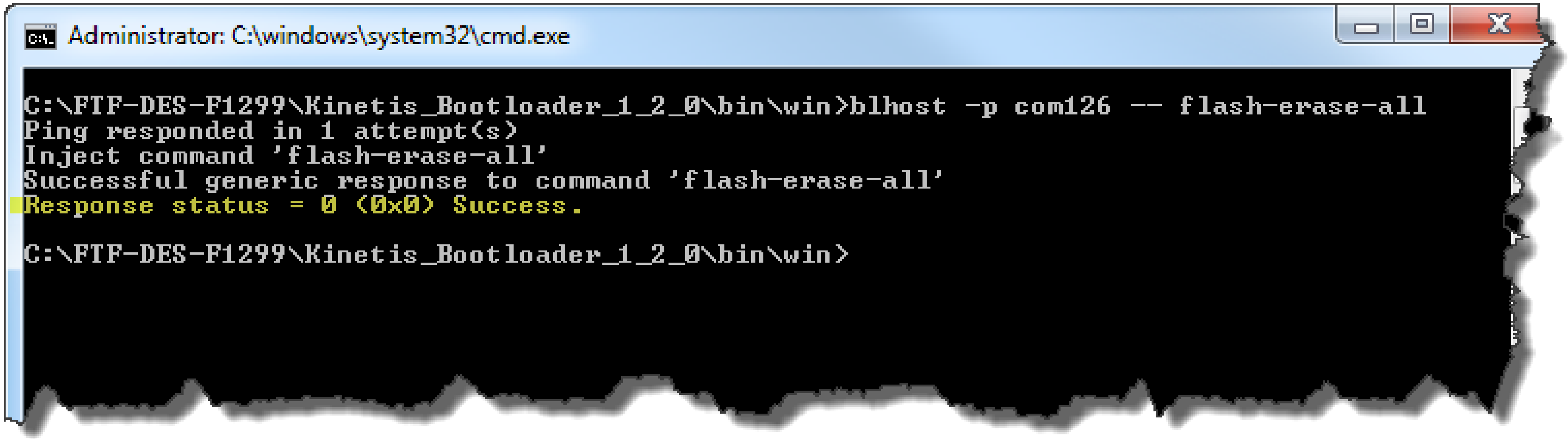
  This means you are successfully communicating to the ROM bootloader.

If you get any other error messages, retry the last three steps on the previous slide.

# ERASE Flash via UART to the KL03 Bootloader

- Now, lets erase the entire contents of flash…
  In the command window, type this command

  blhost –p com<#> -- flash-erase-all



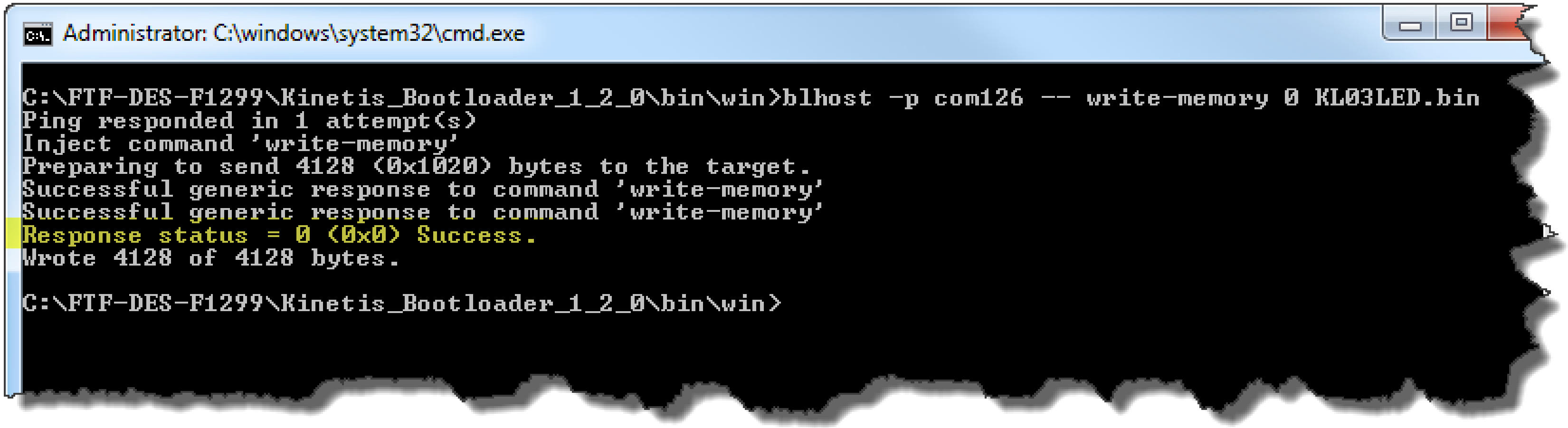
```
C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>blhost -p com126 -- flash-erase-all
Ping responded in 1 attempt(s)
Inject command 'flash-erase-all'
Successful generic response to command 'flash-erase-all'
Response status = 0 (0x0) Success.

C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>
```

- You should see this message: Response status” = 0(0x0) Success
  This means you have successfully erased the entire contents of flash.

# PROGRAM Flash via UART to the KL03 Bootloader

- Now, lets program the contents of flash from a binary file called “KL03LED.bin”…
  In the command window, type this command:

  blhost –p com<#> -- write-memory 0 KL03LED.bin


Administrator: C:\windows\system32\cmd.exe

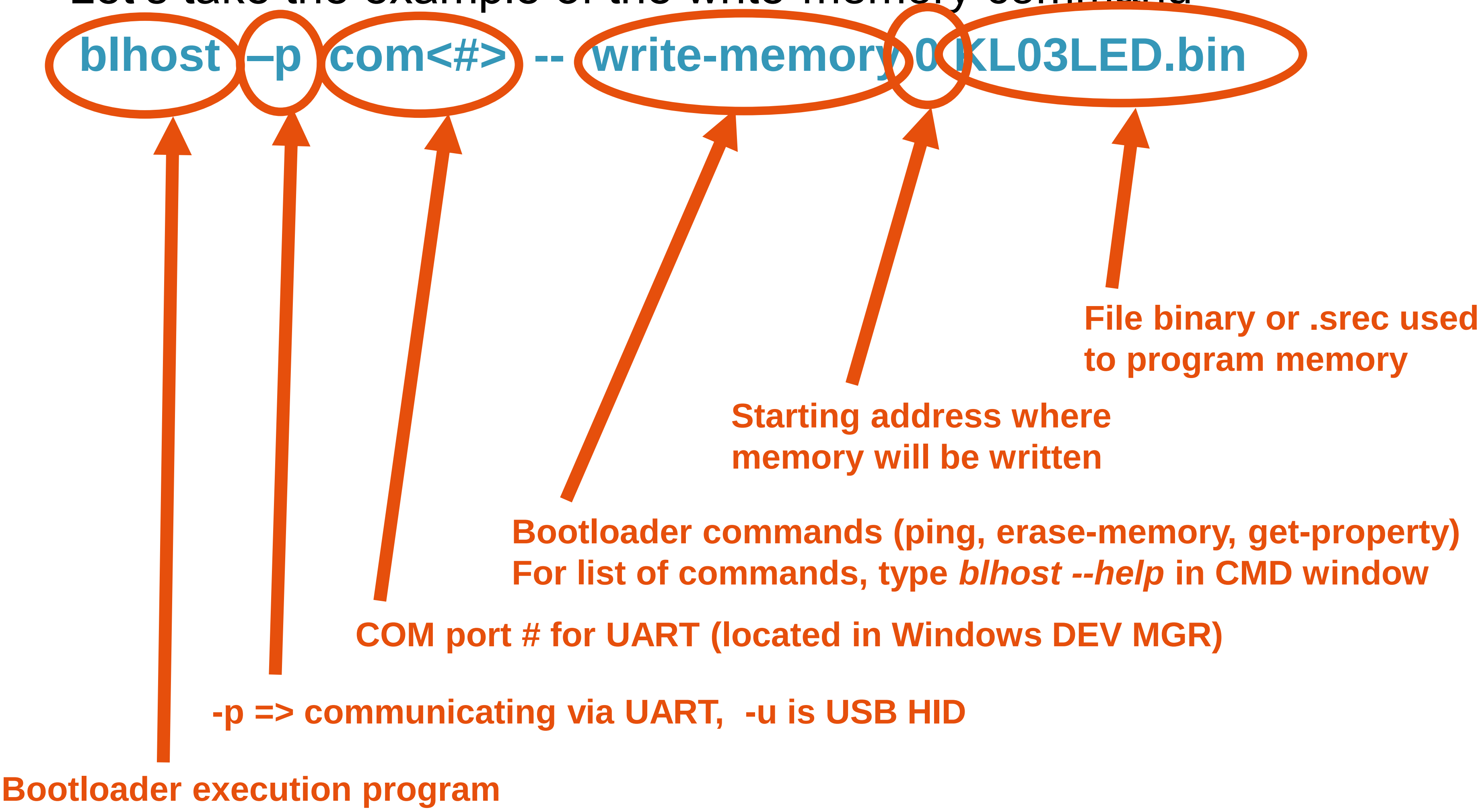
```
C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>blhost -p com126 -- write-memory 0 KL03LED.bin
Ping responded in 1 attempt(s)
Inject command 'write-memory'
Preparing to send 4128 (0x1020) bytes to the target.
Successful generic response to command 'write-memory'
Successful generic response to command 'write-memory'
Response status = 0 (0x0) Success.
Wrote 4128 of 4128 bytes.

C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\bin\win>
```

- You should see this message:  Response status” = 0(0x0) Success

  This means you have successfully programmed the file “KL03LED.bin to the flash memory.
- Now, to execute the code from flash, press the RESET button on the FRDM-KL03Z board.  This will disconnect the bootloader from the PC.
- The board will wait 5 seconds, then start executing from flash memory at 0x0000000, or the reset vector.  The red, green, and blue, LED should flash one second intervals.

# Explanation of the bootloader commands:

- Let’s take the example of the write-memory command



```
blhost –p com<#> -- write-memory 0 KL03LED.bin
```

File binary or .srec used to program memory

Starting address where memory will be written

Bootloader commands (ping, erase-memory, get-property)
For list of commands, type *blhost --help* in CMD window

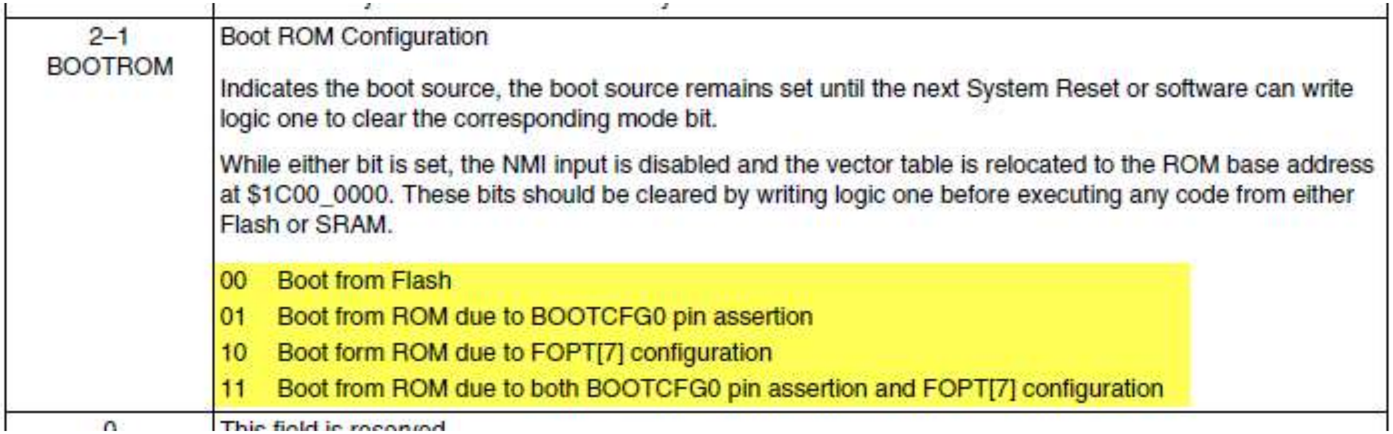
COM port # for UART (located in Windows DEV MGR)

-p => communicating via UART, -u is USB HID

Bootloader execution program

# Understanding the different methods of entry into the bootloader

- ✓For the labs, we were entering the Bootloader using the NMI pin from the MCU (connected to switch SW3).
- ✓The bootloader also has a timeout mechanism.  Meaning, the flash application code will not execute until the five seconds (configurable, default is 5 seconds) after a reset has occurred.
- ✓In the datasheet, section 21.2.6,  RCM Module, there are 4 options for entering the bootloader from RESET.



| | |
|---|---|
| 2–1<br>BOOTROM | Boot ROM Configuration<br>Indicates the boot source, the boot source remains set until the next System Reset or software can write logic one to clear the corresponding mode bit.<br>While either bit is set, the NMI input is disabled and the vector table is relocated to the ROM base address at $1C00_0000. These bits should be cleared by writing logic one before executing any code from either Flash or SRAM.<br>00 Boot from Flash<br>01 Boot from ROM due to BOOTCFG0 pin assertion<br>10 Boot form ROM due to FOPT[7] configuration<br>11 Boot from ROM due to both BOOTCFG0 pin assertion and FOPT[7] configuration |
| 0 | This field is reserved |

# Program the bootloader with different/no entry methods

Option 1: Program the application to **ALWAYS** execute from flash immediately upon reset.

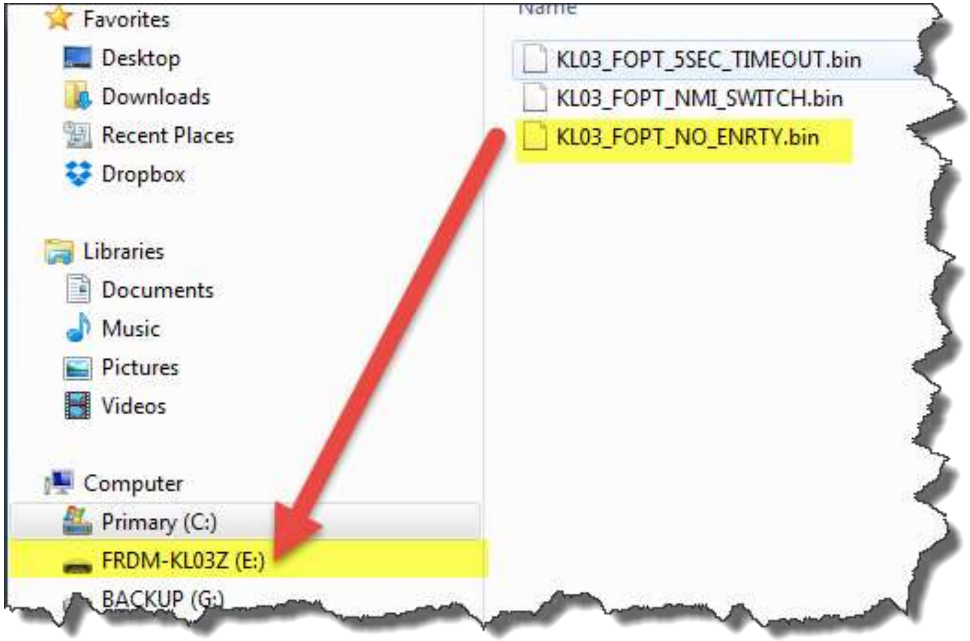
Option 2: Program the application to only execute from the bootloader if the RESET switch and NMI pin is asserted low upon RESET.

Option 3: Program the application to always jump into the bootloader and timeout after 5 seconds from RESET if communications have not been established.

Option 4: Program the application to jump into the bootloader from the user’s code without using RESET or the NMI switch.

# OPTION 1 – LAB:  No bootloader entry

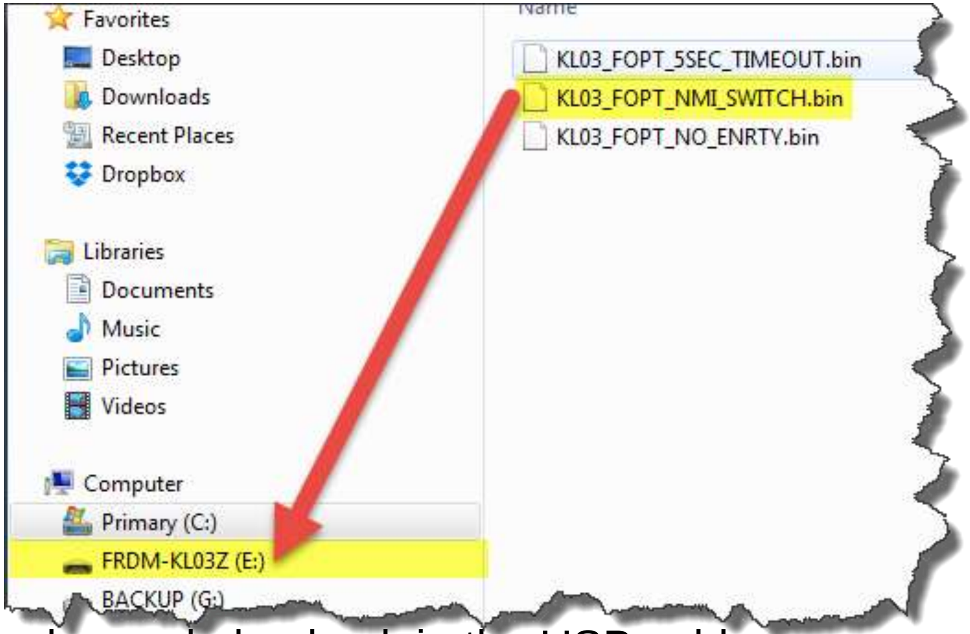
- Open Windows Explorer, and drag and drop the file called: C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\FOPT\ KL03_FOPT_NO_ENRTY.bin



- Unplug and plug back in the USB cable
- Notice that the code starts executing immediately and ignores any method of entry into the bootloader.
- Hold down the “RESET Switch” and the “SW3 Switch”, then release “RESET Switch”, and then release “SW3 Switch”. NOTHING HAPPENS!  Code continues to execute.

# OPTION 2 – LAB:  NMI Switch bootloader entry

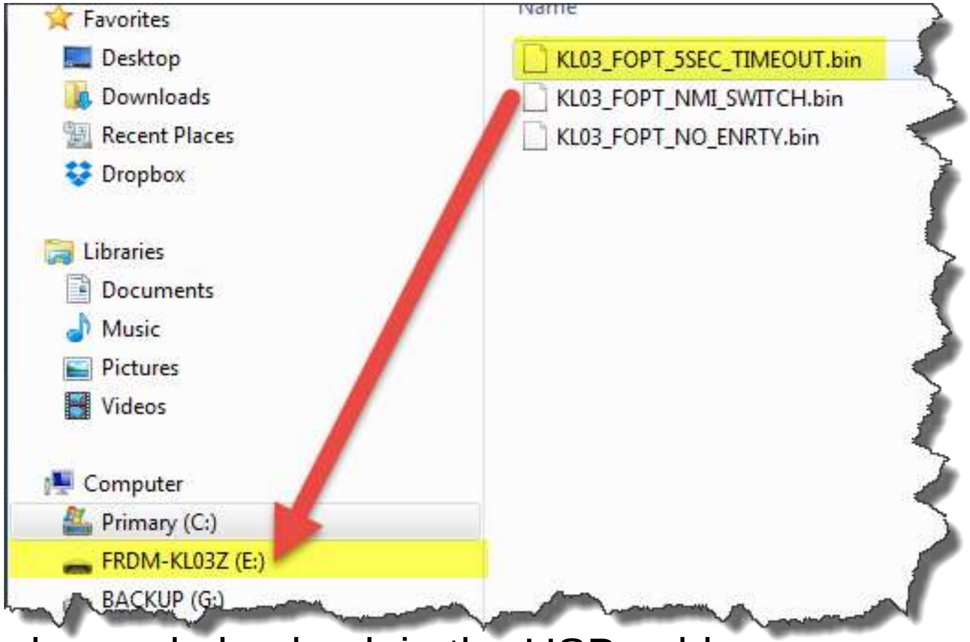
- Open Windows Explorer, and drag and drop the file called:
  C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\FOPT\ KL03_FOPT_NMI_SWITCH.bin



- Unplug and plug back in the USB cable
- Notice that the code starts executing immediately, but will enter bootloader mode if NMI switch is low at reset.
- Hold down the “RESET Switch” and the “SW3 Switch”, then release “RESET Switch”, and then release “SW3 Switch”. Notice the LEDs are not flashing.  Now it jumps into the bootloader until the reset switch is hit again.

# OPTION 3 – LAB: Timeout bootloader entry

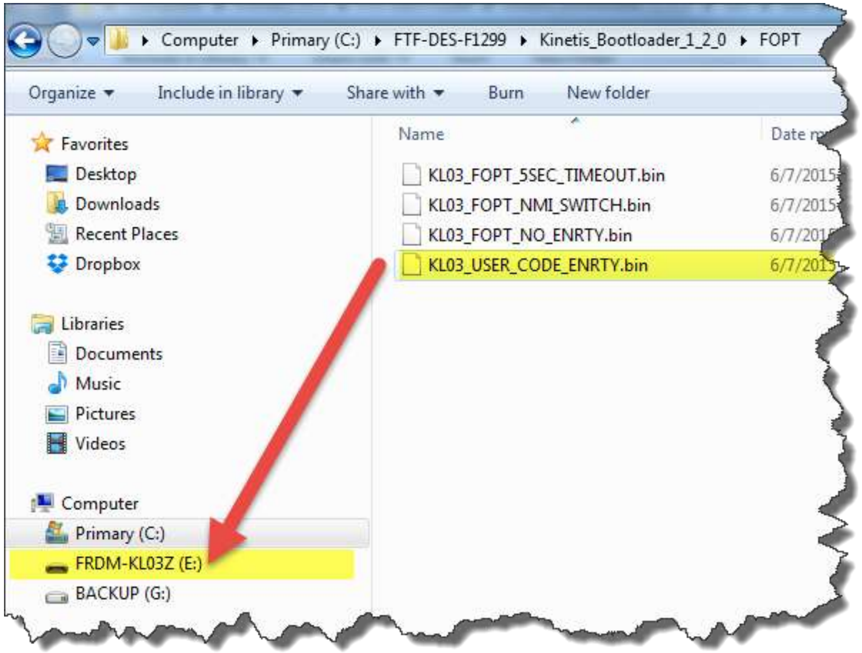
- Open Windows Explorer, and drag and drop the file called: C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\FOPT\ KL03_FOPT_5SEC_TIMEOUT.bin

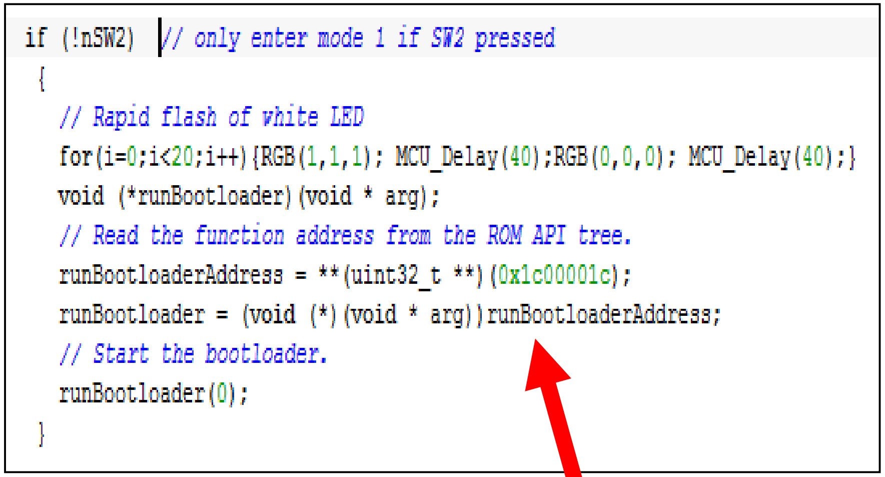


- Unplug and plug back in the USB cable
- Notice now that the code starts executing after 5 seconds of timeout. It will enter bootloader mode if NMI switch is low at reset.
- Hold down the “RESET Switch” and the “SW3 Switch”, then release “RESET Switch”, and then release “SW3 Switch”. Notice the LEDs are not flashing. Now it jumps into the bootloader until the reset switch is hit again.

# OPTION 4 – LAB:  Bootloader entry from user code

- Open Windows Explorer, and drag and drop the file called: C:\FTF-DES-F1299\Kinetis_Bootloader_1_2_0\FOPT\KL03_USER_CODE_ENTRY.bin





```
if (!nSW2)  // only enter mode 1 if SW2 pressed
 {
  // Rapid flash of white LED
  for(i=0;i<20;i++){RGB(1,1,1); MCU_Delay(40);RGB(0,0,0); MCU_Delay(40);}
  void (*runBootloader)(void * arg);
  // Read the function address from the ROM API tree.
  runBootloaderAddress = **(uint32_t **)(0x1c00001c);
  runBootloader = (void (*)(void * arg))runBootloaderAddress;
  // Start the bootloader.
  runBootloader(0);
 }
```

- Unplug and plug back in the USB cable
- Notice now that the code starts executing immediately after RESET.
- Hold down the “SW2 Switch” until the LED starts flashing white.  This is simulating entry into the bootloader from the user /application code memory location. Notice the LEDs are not flashing anymore. You can use the Get-Property command (in the pervious sides) to check if the bootloader is communicating.

# Hands-on: Lab #3 – Bootloader Minimal Configuration for FRDM-K64F Flash-resident Bootloader

- ✓ **Lab objective 1 –** Analyze standard flash-resident bootloader code size
- ✓ **Lab objective 2 –** Understand configuration macro options
- ✓ **Lab objective 3 –** Learn how to customize the flash-resident bootloader and reduce the code size

# Default Flash-Resident Bootloader Configuration

- All peripheral interfaces enabled:
  - UART
  - I2C
  - SPI
  - USB HID

- All supported features and commands enabled:
  - All erase options (region, all, unsecure)
  - Read, Write and Fill memory
  - Flash security
  - Secure binary (where applicable)
  - Run-time control (reset, execute)
  - More...

- Default configuration code size (IAR 7.30.3, high optimization):
  - Code: **22146**
  - RO data: **1322**
  - RW data: **4090**

# Configuration Macro Examples

- Peripheral interface macros:
  - BL_CONFIG_SCUART
  - BL_CONFIG_I2C
  - BL_CONFIG_DSPI
  - BL_CONFIG_USB_HID

- Feature/behavior configuration macros:
  - BL_MIN_PROFILE
  - BL_FEATURE_ENCRYPTION
  - BL_FEATURE_READ_MEMORY
  - BL_FLASH_VERIFY_DISABLE
  - BL_HAS_MASS_ERASE
  - BL_FEATURE_POWERDOWN
  - BL_ENABLE_CRC_CHECK

# BL_MIN_PROFILE Macro

- Used to reduce flash-resident bootloader to the lowest code size configuration.

  Removes the following functionality:
  - Read and Fill memory
  - Secure binary / encryption
  - Mass erase with unsecure
  - Call command (execute a function at specific address)

- UART-only with BL_MIN_PROFILE code size (IAR 7.30.3, high opt.):
  - Code: 9540
  - RO data: 572
  - RW data: 1737

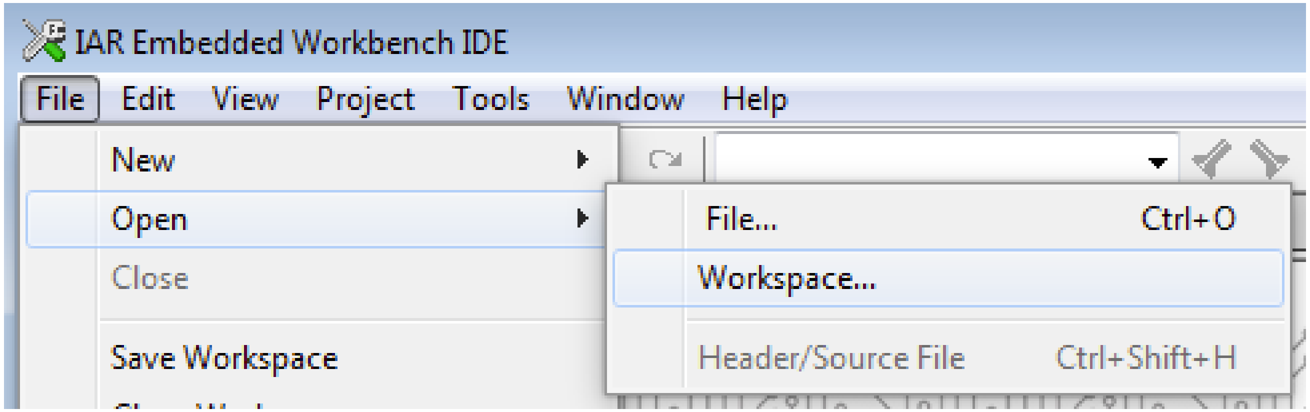
# Let’s Do It - Lab Steps

- Open bootloader project in IAR
- Build bootloader as-is, evaluate code size
- Create UART-only bootloader, evaluate size
- Create UART-only minimal profile bootloader, evaluate size

# Open Bootloader in IAR

- **Open** IAR Embedded Workbench by browsing to Start > All Programs > IAR Systems

- Select File > Open > Workspace...

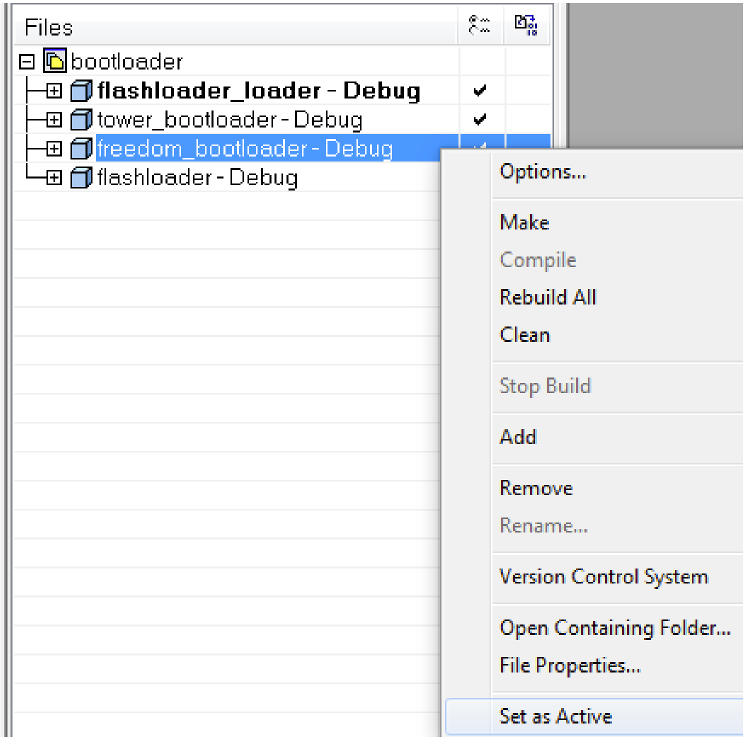


- **Select** the workspace file (bootloader.eww) in:

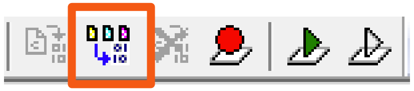
  *C:\FTF-DES-1299\Kinetis_Bootloader_1_2_0/targets/MK64F12*

# Build the Bootloader “as-is”

- Set the “freedom_bootloader” project as active

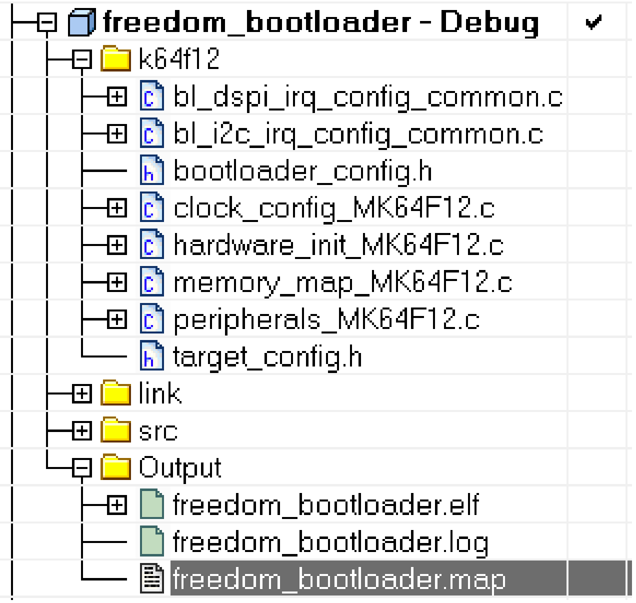


- Click on the “Make” icon to **build** the application



# Evaluate Code Size

- After the build completes, **open the MAP file**. For convenience, the MAP file is linked in the "Output" folder of the project.



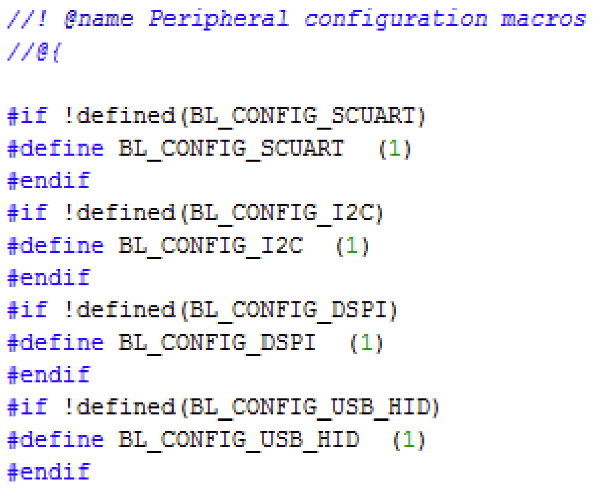
```
freedom_bootloader - Debug  ✔
  k64f12
    bl_dspi_irq_config_common.c
    bl_i2c_irq_config_common.c
    bootloader_config.h
    clock_config_MK64F12.c
    hardware_init_MK64F12.c
    memory_map_MK64F12.c
    peripherals_MK64F12.c
    target_config.h
  link
  src
  Output
    freedom_bootloader.elf
    freedom_bootloader.log
    freedom_bootloader.map
```

- **Browse** to the very bottom of the MAP file and note the size.

# Create UART-only Bootloader

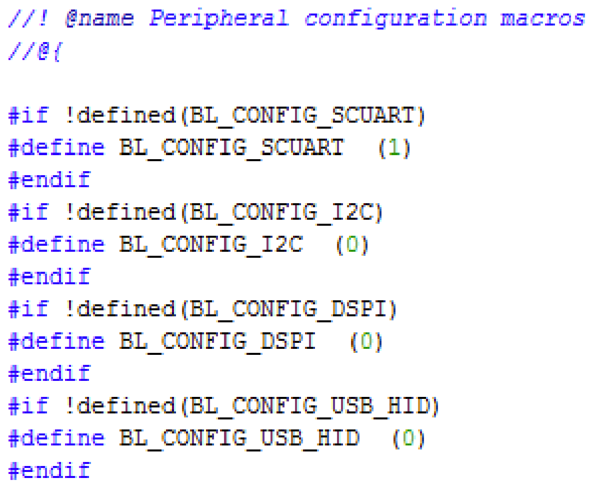
- Open the bootloader_config.h file in the “k64f12” folder of the project
- Change the peripheral configuration macros as shown below

**From:**



```
//! @name Peripheral configuration macros
//@{

#if !defined(BL_CONFIG_SCUART)
#define BL_CONFIG_SCUART   (1)
#endif
#if !defined(BL_CONFIG_I2C)
#define BL_CONFIG_I2C   (1)
#endif
#if !defined(BL_CONFIG_DSPI)
#define BL_CONFIG_DSPI   (1)
#endif
#if !defined(BL_CONFIG_USB_HID)
#define BL_CONFIG_USB_HID   (1)
#endif
```

**To:**

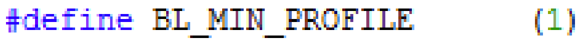


```
//! @name Peripheral configuration macros
//@{

#if !defined(BL_CONFIG_SCUART)
#define BL_CONFIG_SCUART   (1)
#endif
#if !defined(BL_CONFIG_I2C)
#define BL_CONFIG_I2C   (0)
#endif
#if !defined(BL_CONFIG_DSPI)
#define BL_CONFIG_DSPI   (0)
#endif
#if !defined(BL_CONFIG_USB_HID)
#define BL_CONFIG_USB_HID   (0)
#endif
```

- **Save** the changes, and **re-build** the application
- **Evalute** the new code size in the MAP file

# Create UART-only Minimal Profile Bootloader

- Go back to the bootloader_config.h file in the “k64f12” folder of the project
- Keep the changes from the UART-only configuration, but add the following code to the header file:



```
#define BL_MIN_PROFILE        (1)
```

- **Save** the changes, and **re-build** the application
- **Evalute** the new code size in the MAP file

# Q&A

- ✓ Do you fully understand - what is Kinetis Bootloader? How does it work? Any question on what is supported?

- ✓ Do you feel that you have the necessary skill to interface with the bootloader under various settings?

- ✓ Are you able to re-create the lab and configure the bootloader by yourself?

freescale™

www.Freescale.com