AN3703

Live Update Application on PIC32MZ MCUs Using MPLAB Harmony v3

Introduction

The dual-panel Flash on the PIC32MZ microcontroller (MCU) allows the application to implement the Live Update features. The Live Update feature is a piece of code used to program the application code (firmware) to the inactive bank in the internal Flash or non-volatile memory (NVM). The Live Update feature updates a newer version of the firmware without affecting the working application on the Active bank. While the Live update of the current application is under progress, the existing version of the application continues to run.

The following steps provides the flow of the Live Update application:

1. The Live Update application must be programmed using a bootloader into the specific memory location.
2. The Live Update enabled application communicates to the host program (typically running on a PC) to receive the firmware through a serial interface, such as the UART.
3. The piece of code implementing the Live Update runs only when a request for a new firmware upgrade, otherwise, it will be in the idle state allowing other application functions to run.
4. The PIC32MZ MCU divides the memory region into two sections: one for the bootloader and another for the application code.
5. The bootloader is responsible to check whether the user is intending to update the firmware or run the existing firmware.
6. The application sitting in the user application space is capable of updating the firmware at runtime without triggering the bootloader (Live Update feature), in addition to performing the regular application specific functions.

This document discusses the implementation of the Live Update-enabled application on the PIC32MZ MCU with the usage of dual-bank Flash.

Table of Contents

1. Hardware and Software Requirements

1.1 PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit

The PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit is a development kit for evaluating the PIC32MZ microcontroller, which is based on a MIPS® Core running at 200 MHz. The high-performance PIC32MZ EF series of microcontrollers offer industry-leading connectivity and peripheral options that empower embedded designers to rapidly build complex applications.

The following are key features of the PIC32MZ EF MCU:

- 200 MHz/330 DMIPS performance
- Dual-Panel Live Update Flash Up to 2 MB and 512 KB RAM
- Excellent connectivity options (Hi-Speed USB, CAN, and 10/100 Ethernet)
- Integrated double-precision FPU accelerates performance in process-intensive applications
- Optional full-featured hardware crypto accelerator
- Rich peripheral set

The PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit is available at Microchip Direct.

1.2 MPLAB X Integrated Development Environment (IDE) and XC Compilers

The MPLAB® X Integrated Development Environment (IDE) is an expandable, highly configurable software program that incorporates powerful tools to discover, configure, develop, debug, and qualify embedded designs for most of the Microchip's microcontrollers.

- The MPLAB X IDE is available at Microchip Website. This document describes the MPLAB X IDE version 5.40.
- The MPLAB XC Compilers are available at Microchip Website. This document describes MPLAB XC32 version 2.41.

1.3 MPLAB Harmony v3

MPLAB Harmony v3 is a fully integrated embedded software development framework that provides flexible and interoperable software modules that enables dedicating resources to create applications for 32-bit PIC® and SAM devices, rather than dealing with device details, complex protocols, and library integration challenges.

It includes the MPLAB Harmony Configurator (MHC), an easy-to-use development tool with a graphical user interface (GUI) that simplifies device set up, library selection, configuration, and application development. The MHC is available as a plug-in that integrates with the MPLAB X IDE and has a separate Java executable for stand-alone use with other development environments.

The application discussed in this document uses the following MPLAB Harmony v3 repositories. These repositories can be downloaded from GitHub:

- CSP (Chip Support Package)
- DEV_PACKS (Harmony 3 Product Database)
- MHC (Harmony 3 Configurator)
- bootloader (Bootloader)
- UART Bootloader Application
  Or
- Use the MPLAB Harmony 3 Framework Downloader to download the above mentioned repositories.

1.4 Python

This document describes Python scripts (Python v2.7) for sending the application binary from the HOST-PC to the target (PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit).

2. Design

The design of the Live Update application on the PIC32MZ MCU uses the dual-bank Flash memory feature. The banks are named as BANK1 and BANK2. At any point in time, the application considers the bank on which it is currently executing as an active bank while the other bank is marked as an inactive bank.

The dual-bank Flash enables programming the inactive bank with a new version of the firmware, while running the current version of the firmware from the active bank.

The Live Update application is divided into two tasks.

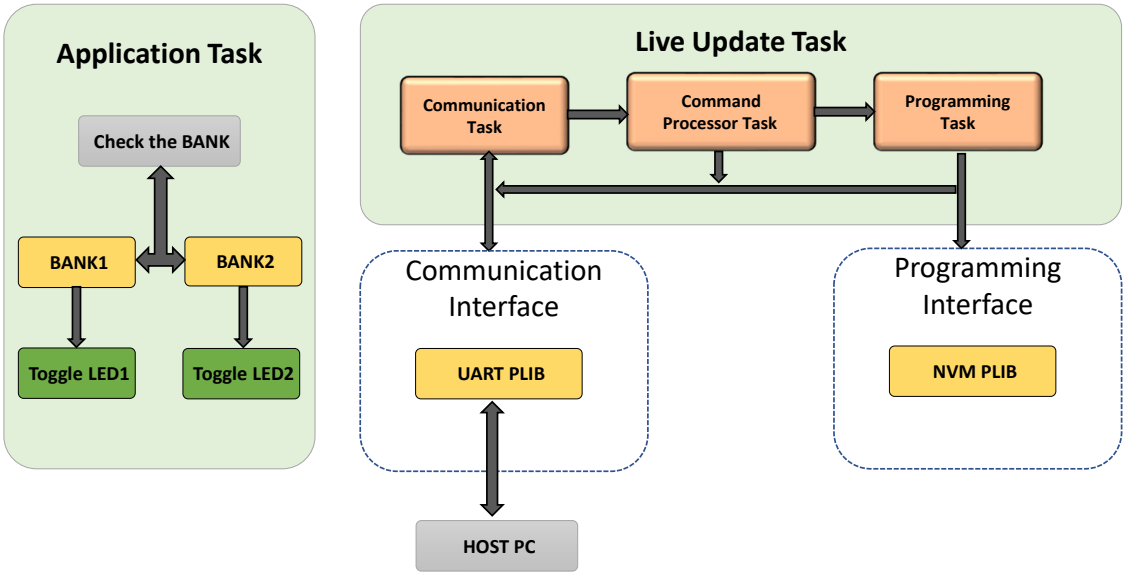
1. Application Task: Represents the end-user application. The current version of the end-user application would be running from the active bank of the dual-blank Flash, while the inactive bank would be available to upgrade the end-user application with the latest version of the firmware. The Application Task discussed in this document is minimalistic in nature. It identifies the Flash bank which has the latest version of the firmware (Active bank), and toggles a specific LED at a periodic rate. For example, it starts with toggling LED1 and when there is a bank swap (active bank changed following a successful upgrade), it toggles LED2. Similarly, the process repeats for every successful application firmware update.
2. Live Update Task: Communicates with the personal computer host application through a predefined communication protocol (discussed in the UART Live update Protocol) and performs the firmware update at runtime when there is a request from the HOST-PC. Otherwise, it will be in an idle state and allows the application functions to run.

The Live update task is divided into three sub-tasks:

- Communication Task
- Command Processor Task
- Programming Task

The following diagram illustrates the Live Update application design.

Figure 2-1. Design of the Live Update Application



2.1 UART Live Update Protocol

The Live Update firmware communicates with the personal computer host application by using a predefined communication protocol to exchange the data between the target and the host.

The UART Live update protocol comprises a Guard, Data Size, Command, and Data bytes as shown in the following figure.

Figure 2-2. UART Live Update Protocol

- **GUARD**
  - The Guard is a constant value 0x5048434D
  - This value provides the protection against the spurious commands
  - Live update firmware always checks for the Guard value at the start of packet reception and proceeds further accordingly
- **Data Size**
  - This field indicates the number of data bytes to be received
  - This value varies for different commands
- **Command**
  - Indicates the command to be processed. Each command is of 1 byte width.
  - Below are the supported commands
    - Unlock (0xA0)
    - Data (0xA1)
    - Verify (0xA2)
    - Bank Swap (0xA3)
- **Data**
  - Contains the actual data to be processed based on the command
  - Length of the data to be received is indicated by a Data Size field
  - Bootloader receives the data in the size of words (4 bytes)
  - All data words must be sent in a little-endian (LSB first) format

Response Codes

The Live update will send a single character response code in response to each command. The sequential commands can only be sent after the response code is received for a previous command, or after 100 ms timeout without a response.

The valid response codes are aas follows:

- OK (0x50): Command was received and processed successfully
- Error (0x51): There were errors during the processing of the command
- Invalid (0x52): Invalid command is received
- CRC OK (0x53): CRC verification was successful
- CRC Fail (0x54): CRC verification failed

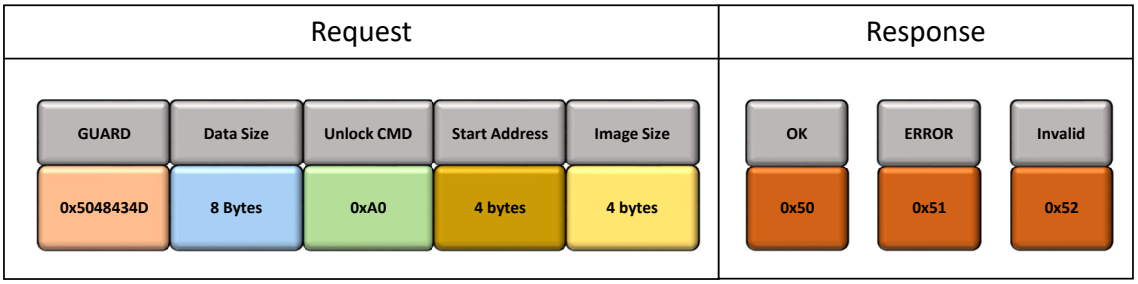
The following Command Description section provides a detailed description on the request and response codes used in the UART Live Update protocol.

2.2 Command Description

Unlock Command

The Unlock command sequence is shown in the following figure with corresponding responses.

Figure 2-3. UART Live Update Unlock Command

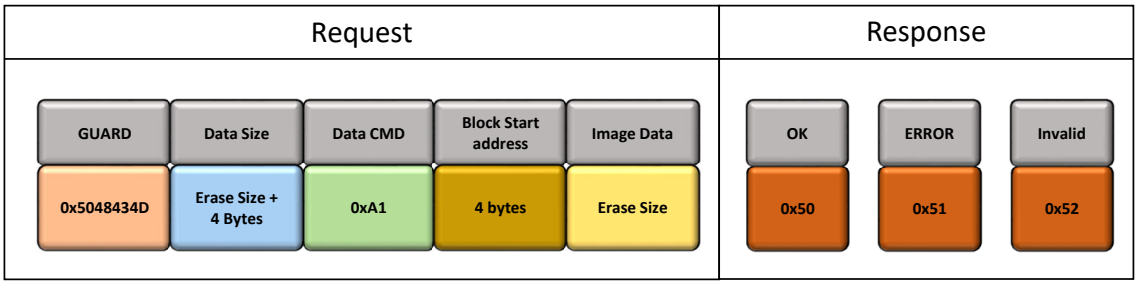


- The Unlock command must be issued before the first Data command
  - It is used to calculate the application start address and end address
  - This information will be used to validate if the addresses sent are within the range of the Flash memory
  - It is used to validate the addresses coming with the data packet to be programmed are within the region for which the unlock command is invoked
- Number of bytes of data to be received is 8 Bytes (Start Address + Image Size)
- Start Address:
  - It is the application Start Address of the Flash memory
  - It is device dependent and should be always greater than or equal to the bootloader end address
  - It must be aligned at an Erase Unit Size boundary, which is also device dependent
  - To upgrade the bootloader itself this value must be set to 0
- Image size must be in increments of Erase Unit bytes, which is also device dependent

Data Command

The Data command sequence is shown in the following figure with corresponding responses.

Figure 2-4. UART Live Update Data Command

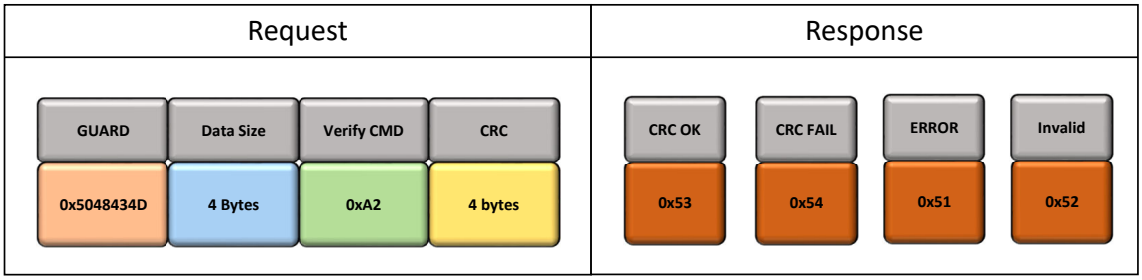


- Data command is used to send the image data
- Data size is equal to the sum of the block start address (4 Bytes) and Erase Unit Size, which is device dependent
- Block start address must be located inside the region previously unlocked through the Unlock command
- Attempts to request the write outside of the unlocked region will result in an error and supplied data will be discarded

Verify Command

The Reset command sequence is shown in the following figure with corresponding responses.

Figure 2-5. UART Live Update Verify Command

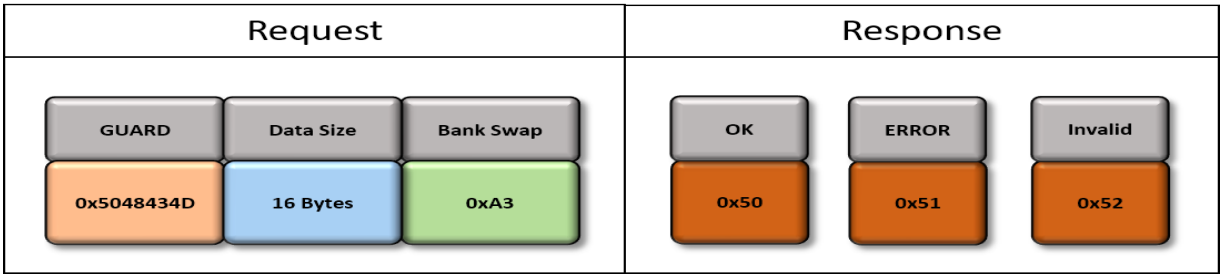


- The Verify command is used to verify the image data sent and programmed
- Image CRC is a standard IEEE CRC32 with a polynomial of 0xEDB88320
- Internal CRC is calculated based on the values read from the Flash memory after programming, hence it verifies the whole chain
- Image CRC is calculated over the previously unlocked region

Bank Swap Command

The Bank Swap Command sequence is shown in the following figure with corresponding responses.

Figure 2-6. UART Live Update Bank Swap Command



- The Bank Swap command is used to swap the Inactive bank to the Active bank
- The device reset runs the new application programmed in the Inactive bank

2.3 Communication Task

This task is responsible for receiving data from the Host PC or Embedded Host through the selected communication interface in interrupt mode. It validates the incoming packet from the host with expected header information before passing it to the command processor task.

2.4 Command Processor Task

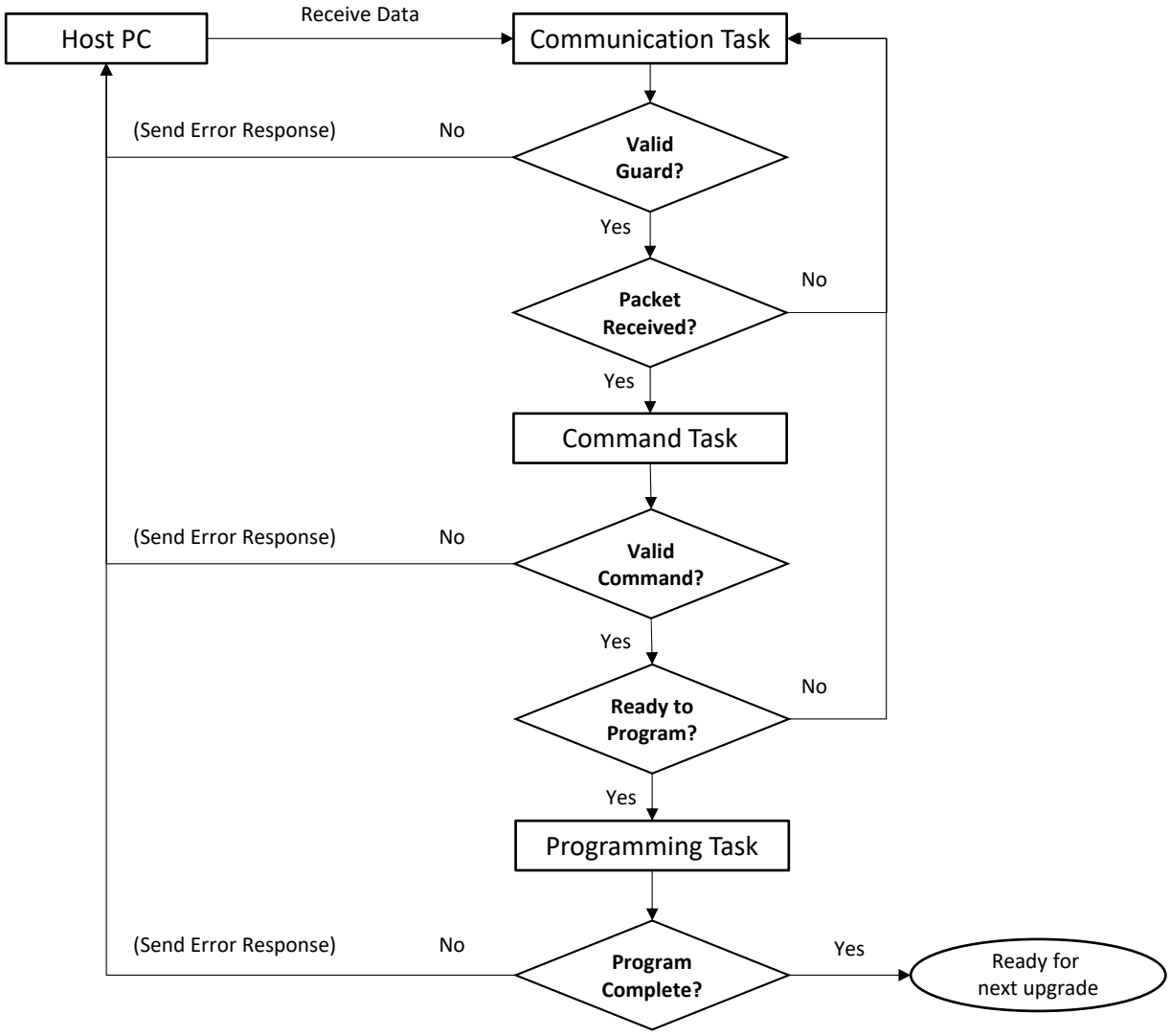
This task processes the commands received from communication tasks and acts upon them, and provides the response back to the host PC accordingly. If the received command is a program command, then it gives control to the programming task.

2.5 Programming Task

This task is responsible for programming the internal Flash memory with a data packet received. It uses the NVM peripheral library to perform the Unlock, Erase, or Write operations and invokes the communication task in parallel to receive the next packet while waiting for the Flash operation to complete.

The following figure shows the Firmware Upgrade Execution Flow chart.

Figure 2-7. Firmware Upgrade Execution Flowchart



2.6 Memory Layout

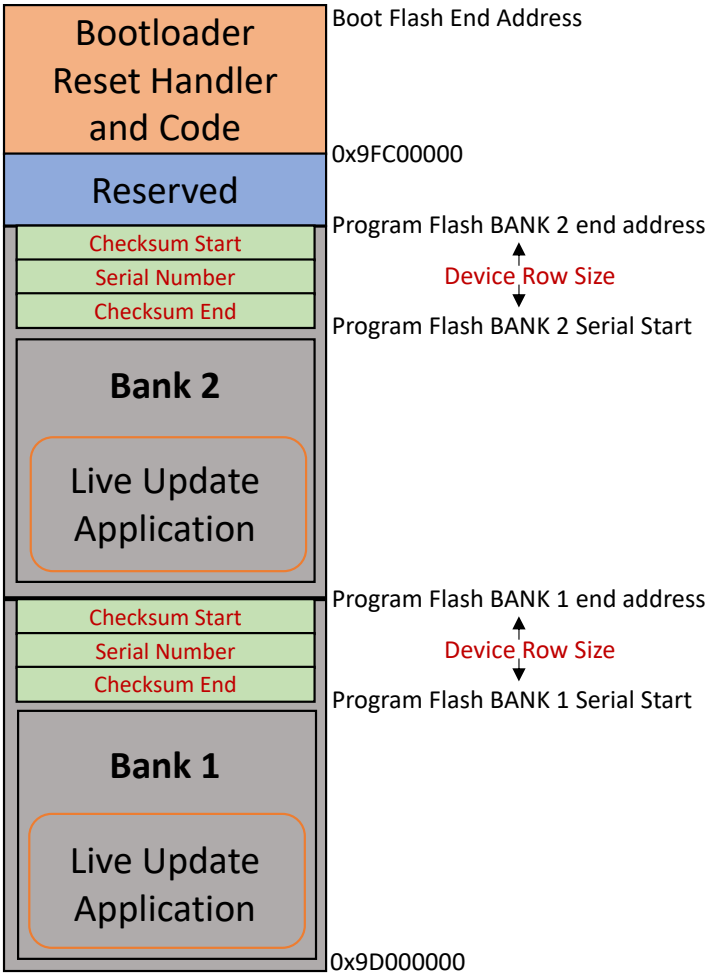
The Live Update application feature is supported for MCU devices with dual-bank support in Flash memory. The following points represent the memory layout for the PIC32MZ dual-bank Flash and the placement and configuration of the Bootloader and Live update application.

- The Bootloader code is always placed at the start of the Boot Flash memory either at KSEG0 (cacheable virtual address 0x9FC00000) or KSEG1 (not-cacheable virtual address 0xBFC00000). In this application, the bootloader code is placed at KSEG0 virtual address. Upon reset, the device runs from the start of boot Flash memory.
- The program Flash memory is divided into two equal banks (BANK1 and BANK2). Either of the banks will be mapped to a lower region (0x1D000000) from which the application must run:
  - The start address of the Active Bank which is mapped to a lower region is always 0x9D000000
  - The start address of the Inactive Bank is from the middle of the Program Flash memory, which can vary from device to device. Refer to the respective data sheets for details of Flash memory layout.
- The start address of the Inactive Bank is from the middle of the Program Flash memory, which can vary from device to device. Refer to the respective data sheets for details of Flash memory layout.

- Row size number of bytes is reserved at the end of each bank for storing a serial number. This serial number will be used by the Live update application to map the appropriate bank to a lower memory region and run the Live application from there.
- The application start address must always fall into the lower mapped region (0x9D000000 to middle of Flash). The size of the application in the linker script must not exceed the middle of the Flash.
- The address passed to the bootloader during programming must fall either in an Active bank or Inactive bank based on the update performed.

The following figure shows the PIC32MZ dual-bank Flash memory layout for the Live Update application memory:

Figure 2-8. Live Update Application in Dual Bank Memory



2.7 Execution Flow

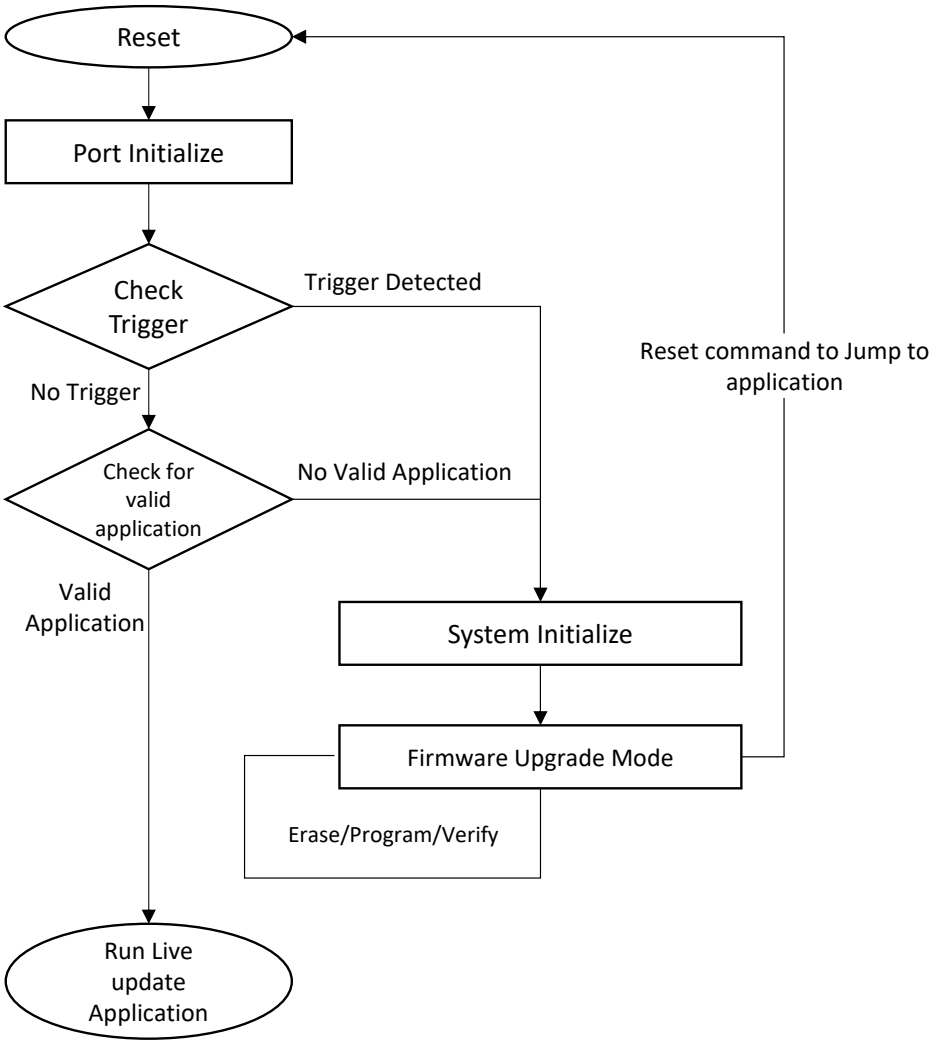
The Live Update application is programmed first time by using either the MPLAB Harmony v3 Basic Bootloader or the UART Fail-Safe Bootloader. The following section describes the MPLAB Harmony v3 Basic Bootloader and Live update application system-level execution flow.

Basic Bootloader System Level Execution Flow:

The Basic Bootloader code starts executing on a device reset. If there are no conditions to enter the firmware upgrade mode, the Basic Bootloader starts executing the user application. The Basic Bootloader performs Flash/Erase/Program operations while in the firmware upgrade mode.

The following diagram illustrates the MPLAB Harmony v3 Basic Bootloader system-level execution flow for programming of the Live Update application for the first time.

Figure 2-9. MPLAB Harmony v3 Bootloader Flow for Programming of the Live Update Application



Refer to the following link for additional information on the MPLAB Harmony v3 Basic or Fail-Safe Bootloaders.

<Harmony framework download folder>\bootloader_apps_uart\apps

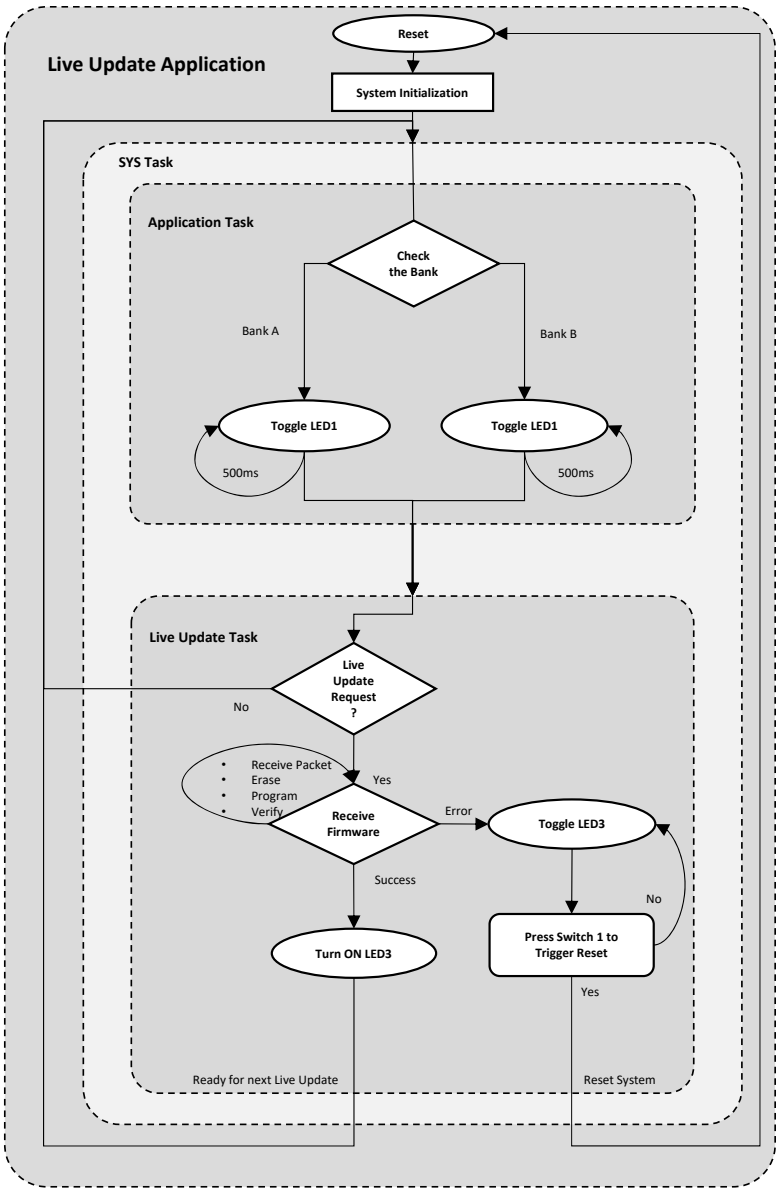
Live Update System Level Execution Flow:

After successful programming of the Live update application through the MPLAB Harmony v3 Bootloader, the Bootloader starts executing the Live update application.

The Live update application receives the new application image from the host. Once the Live update performs a successful upgrade, it moves to an idle state and waits for the next Live update request. The newly updated firmware will be loaded on the next power-on cycle or system reset.

The following diagram illustrates the Live update application execution flow.

Figure 2-10. Live Update Application Execution Flow



Flow chart sequence:

1. Upon reset, the device Bootloader loads the Live Update application to the Active bank of the dual bank.
2. The Live Update application does the system initialization and calls the Application Task followed by the Live Update Task.
3. The Application Task checks the active bank from which the Live Update application is executing, and toggles the LED 1 or LED 2 accordingly.
4. The Live Update Task runs only when there is a request for firmware upgrade otherwise it will be in idle state and the Application Task will continue to run.
5. If the application receives a request for the firmware upgrade, the Live Update Task does the following:
   – Starts receiving the packets only when the UNLOCK command is received from the host PC.
   – Receives the firmware image when the DATA command is received from the host PC and programs it in the inactive bank.
   – Verifies the received firmware image when the VERIFY command is received from the host PC.

   – When the SWAP Bank command is received, it reads the serial number from the Active bank, increments by 1, and then writes to the Bank 2 serial number.

6. If the host application requests to update the Active Bank and the address falls into the active bank serial sector then the Live Update Task sends an error response, aborts the programming operation, and toggles to LED3.
7. If any error occurs in Step 5 then it returns from Live Update mode and toggles an LED3 for every 1000 ms until the Switch1 is pressed to trigger the system reset for restarting the Live update again.
8. If the Live update is a success, then it turns LED3 ON and jumps to Step 5 for the next Live update.

3. Configuration

The Live Update application is comprised of these applications:

- **Bootloader** (`uart_fail_safe_bootloader_pic32mz_ef_sk`): Used to upgrade the Live Update application.
- **Live update application** (`pic32mz_uart_live_update`): This application consists of implementation of the Live Update feature.

Note: The MPLAB Harmony v3 bootloader project is available in the MPLAB Harmony v3 Bootloader Repository in the following path:

<Harmony framework download folder>\bootloader_apps_uart\apps\uart_fail_safe_bootloader

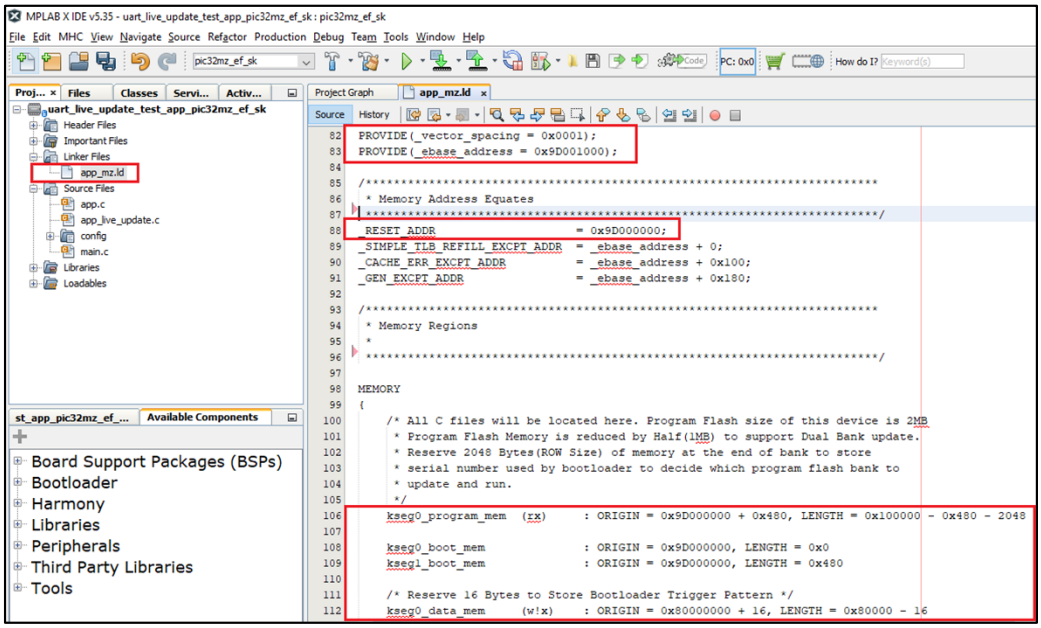
Check the following path for more information on how the UART Fail-Safe Bootloader is configured.

<Harmony framework download folder>\bootloader\doc\help_bootloader.chm

3.1 Linker Script

The XC32 linker script for the PIC32MZ device comes with a default reset address and memory region address. The Live Application code including the Reset handler and Interrupt Vector Tables (IVT's) must be placed in the Program Flash memory as the bootloader is residing in the boot Flash memory, refer to the Memory Layout section for additional information. Therefore, the custom linker script (`app_mz.ld`) must be created and added to the application project for the Live Update application project. The following figure shows the application custom linker script for the PIC32MZ device and the highlighted changes.

Figure 3-1. Live Update Application Custom Linker Script



The following changes were made in the PIC32MZ custom linker script:

- The vector address of a given interrupt is calculated using the Exception Base (EBASE) CPU register, which provides a 4 KB page-aligned base address value located in the kernel segment (kseg) address space.
- The MIPS32 core requires that the IVTs be placed on a 4 KB boundary.
- The address is calculated by using the EBASE and VS values. The VS bits provide the vector spacing between the adjacent vector addresses.
- Each vector in the table is created as an output section located at an absolute address based on values of the `_ebase_address` and `_vector_spacing` symbols.

- The Reset Address for the application loaded through the Bootloader should match the Live application start address mentioned in the Bootloader project.
- For devices with a larger boot Flash memory where the Bootloader resides completely in the boot Flash memory, the application start address by default will be the start of program Flash memory 0x9D000000.

```
_RESET_ADDR                 = 0x9D000000;
kseg0_program_mem      (rx) : ORIGIN = 0x9D000000 + 0x480, LENGTH = 0x200000 - 0x480
kseg0_data_mem        (w!x) : ORIGIN = 0x80000000, LENGTH = 0x80000
kseg1_boot_mem              : ORIGIN = 0x9D000000, LENGTH = 0x480
/* Boot Sections */
.reset _RESET_ADDR :
{
  KEEP(*(.reset))
  KEEP(*(.reset.startup))
} > kseg1_boot_mem
```

Notes:

- The device configuration bits are removed in the Live update application custom linker script, as it must be updated by the bootloader
- Device configurations must be discarded from the final hex file for the application project.

  ```
  /DISCARD/ : { *(.config_*) }
  ```

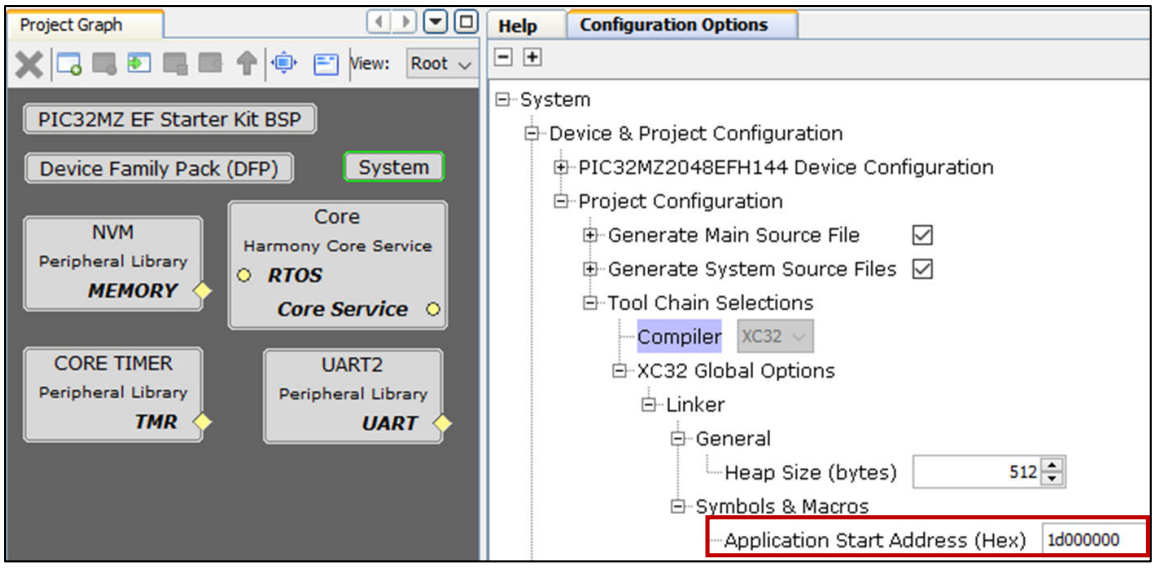
- Refer to the pre-developed Live update application custom linker script placed in *<Live update application folder>/firmware/src/config/pic32mz_ef_sk/app_mz.ld*.

3.2 MHC Configuration

The following steps describes the MHC configurations required to implement the Live Update application:

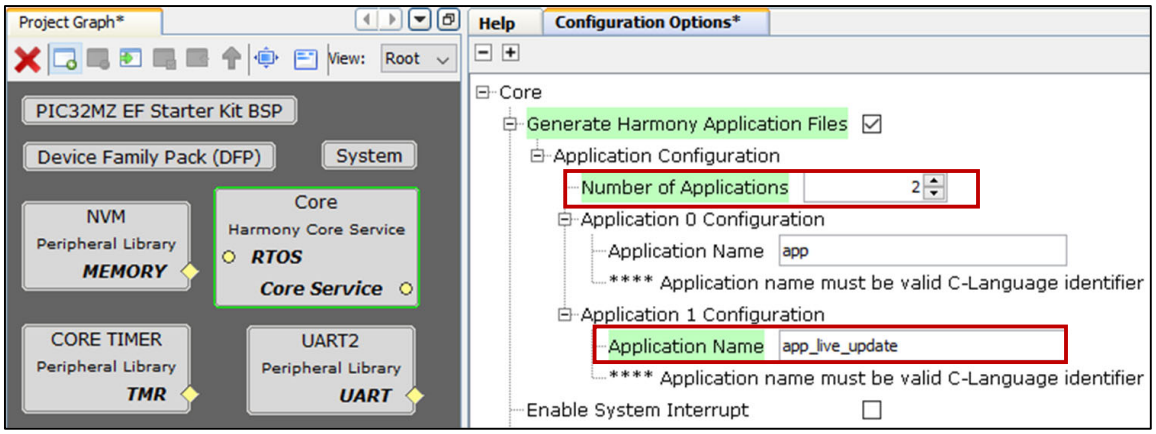
1. Remove the Device Fuse configurations from the custom linker script as it will be updated by the bootloader project.
2. Provide custom Live application linker script for the application start address matching the Application start address in the bootloader project.

   Figure 3-2. Live Update Application Start Address Configuration



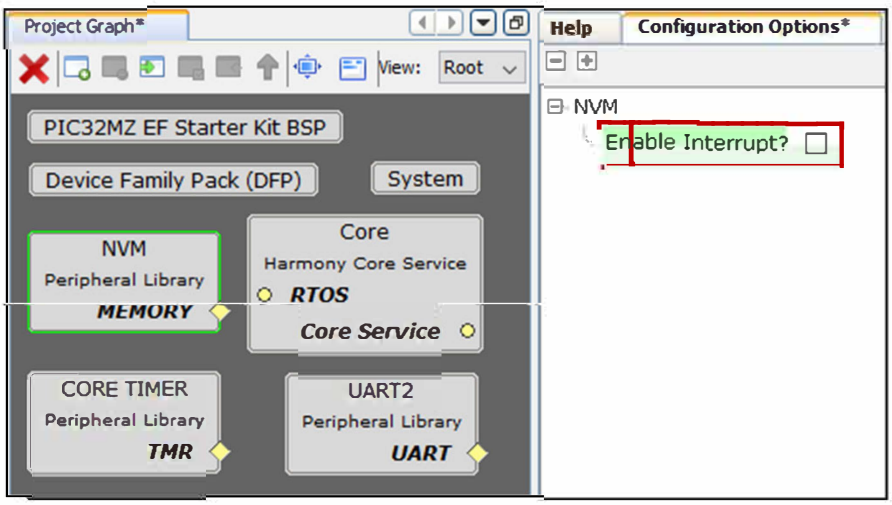
3. Add MPLAB Harmony v3 core component and configure the application and Live Update tasks. Generate the source files, such as `tasks.c`, which can be used to add any sub-tasks and `user.h` for adding any user configurations.

Figure 3-3. Live Update Application MPLAB Harmony v3 Core Configuration



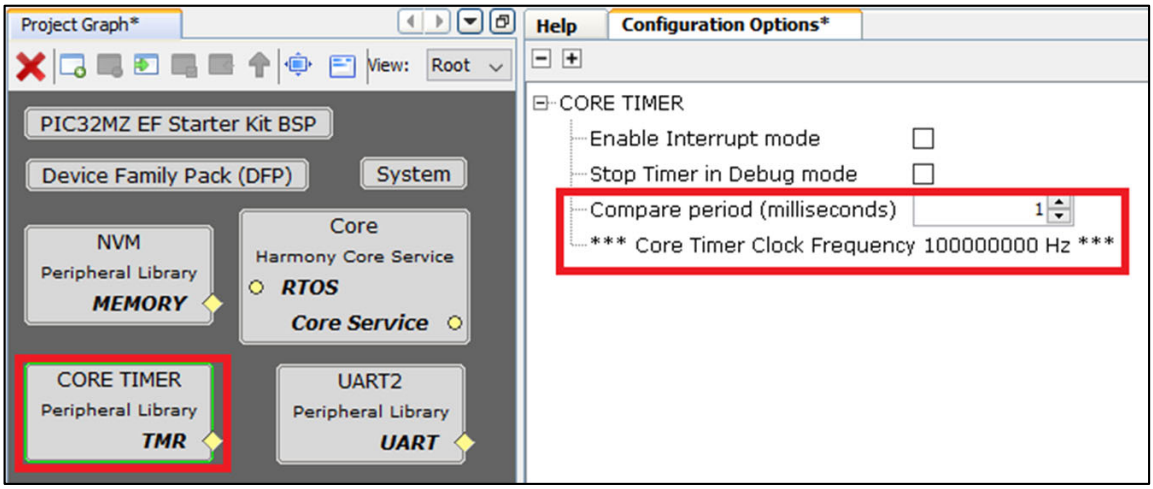
4. Add the NVM module and disable the interrupt. The Live Update application uses the NVM PLIB in polling mode, therefore the interrupt must be disabled as shown in the following figure.

Figure 3-4. Live Update Application NVM Configuration



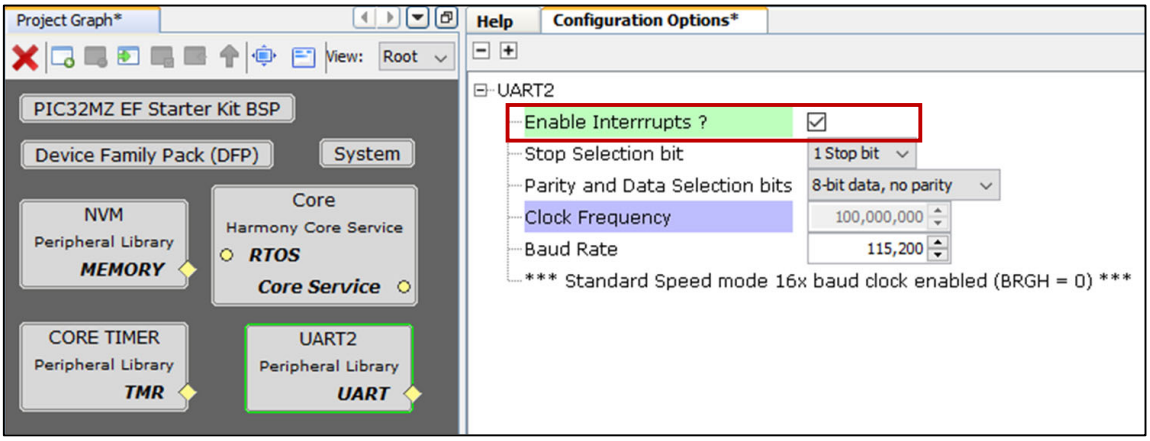
5. Configure the CORE TIMER with a Compare period of 1 ms to blink an LED at different rates.

Figure 3-5. Live Update Application Core Timer Configuration



6. Add the UART module to receive the data from the host and enable interrupt as shown in the following figure.

Figure 3-6. Live Update Application UART Configuration

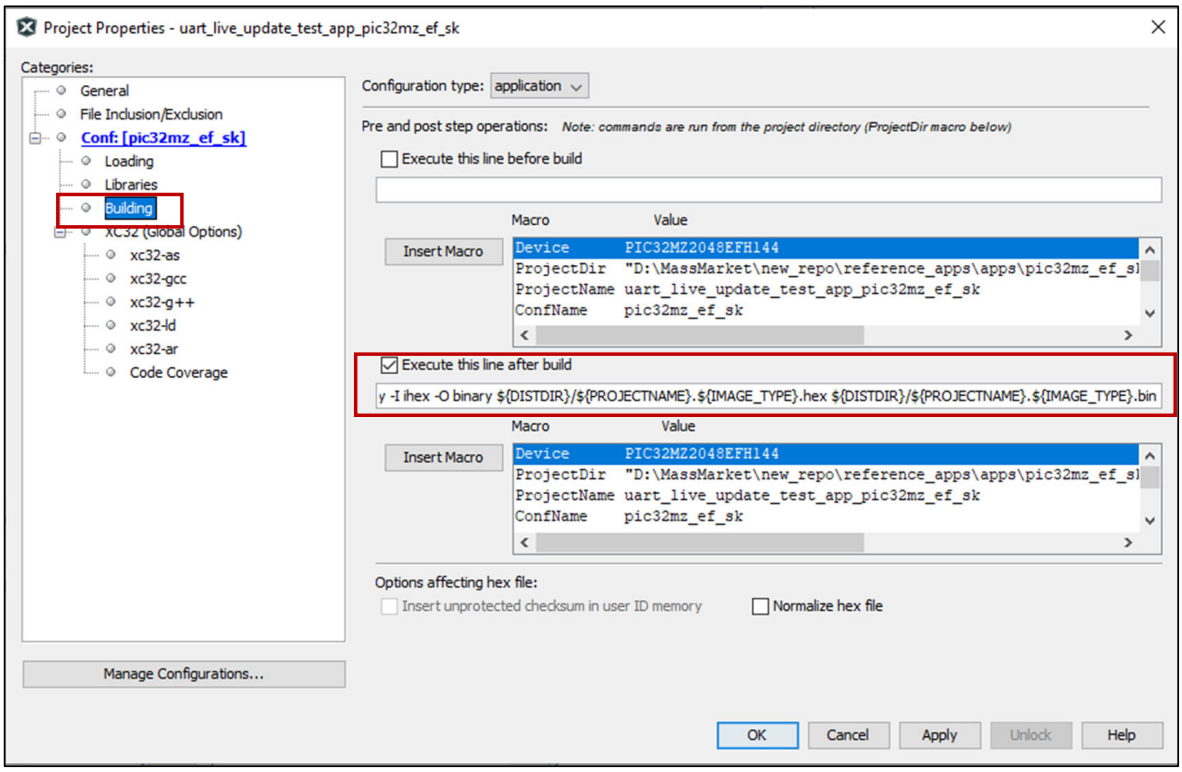


3.3 Project Settings

- **Preprocessor macro definitions:**
  - ROM_ORIGIN and ROM_LENGTH are the XC32 linker variables which will be overridden with provided values.
  - The application start address is auto-populated in the linker script with the value of the application start address provided in the MHC after regeneration (See Live Update Application Custom Linker Script).
- **Execute the line after Build:**
  - The following build options can be used to automatically generate the binary file from the hex file once the build is complete:

```
${MP_CC_DIR}/xc32-objcopy -I ihex -O binary ${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.hex
${DISTDIR}/${PROJECTNAME}.${IMAGE_TYPE}.bin
```

Figure 3-7. Live Update Application Binary Generation Setting



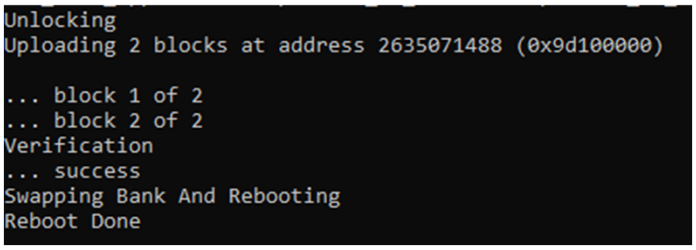
4. Running the Application

4.1 Running the Bootloader Application

The first time the Live Update application is programmed by using either the MPLAB Harmony v3 Basic Bootloader or the UART Fail-Safe Bootloader by following these steps:

1. Download the MPLAB Harmony v3 Bootloader package.
2. Download the MPLAB Harmony v3 UART Bootloader Applications package.
3. Connect a mini USB cable to the DEBUG port of the PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit.
4. Build and program the UART fail safe bootloader (Dual-bank bootloader) avaialbe at the (*<Harmony Framework download folder>/bootloader_apps_uart/apps/uart_fail_safe_bootloader/bootloader/firmware/pic32mz_ef_sk.X*) using MPLAB X IDE.
5. Build the Live Update application (*<Live Update application folder>/firmware/pic32mz_ef_sk.X*) using the MPLAB X IDE, but do not program.
6. From the command prompt, run the bootloader host script `btl_host.py` available at *<Harmony Framework download folder>/bootloader/tools_archive* to program the application binary:
   - `python <Your Harmony Framework download folder>/bootloader/tools_archive/btl_host.py -v -s -i <COM PORT> -d pic32mz -a 0x9D100000 -f <Live Update application folder>/firmware/pic32mz_ef_sk.X/dist/pic32mz_ef_sk/production/pic32mz_ef_sk.X.production.bin`
     Note: For additional information on the bootloader host script, refer to the help files for setting up the host script, which is available at < *Harmony Framework download folder>/bootloader/doc/help_bootloader.chm*.
7. The following figure shows the successful programming of the Live Update application binary. The messages, *'Swapping Bank And Rebooting' and 'Reboot Done'*, signify the bootloading was successful.

Figure 4-1. First Time Live Update Application Programming Using Bootloader



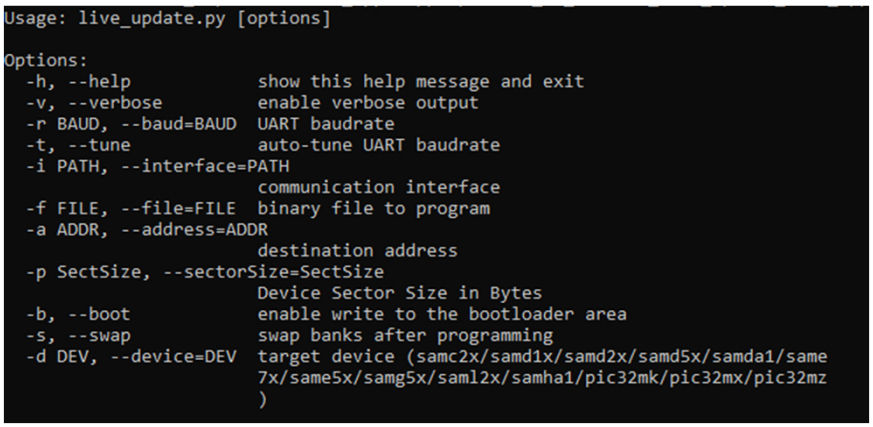
4.2 Running the Live Update Application

1. Perform the Running the Bootloader Application steps, if not done already.
2. If the above step is successful, the LED1 or LED2 on the PIC32MZ Embedded Connectivity with FPU (EF) Starter Kit will start blinking, which indicates the application is running from BANK 1 or BANK 2 based on where the program is running.
3. Reset or power cycle the device.
4. Run the host script `live_update.py` for the Live Update application from a command prompt available at *<Live Update application folder>/scripts* to program the latest version of the firmware to an inactive panel while the current version of the application is running on the active panel.

```
python <Live Update application folder>/scripts/live_update.py
 -v -s -i <COM PORT> -d pic32mz -a 0x9D100000 -f <Live Update application folder>/
firmware/pic32mz_ef_sk.X
/dist/pic32mz_ef_sk/production/pic32mz_ef_sk.X.production.bin
```

– The following figure shows the Live Update script help.

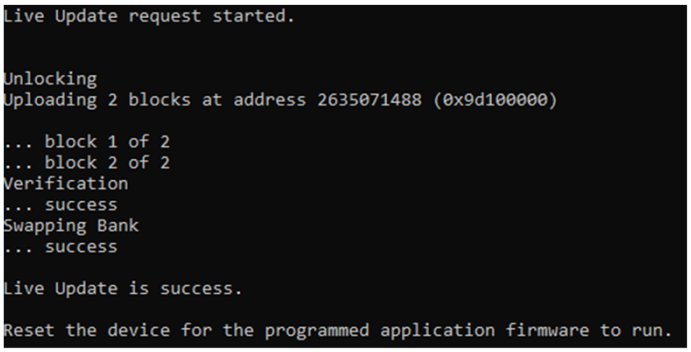
Figure 4-2. Live Update Application Host Script Help Window




```
Command: python <python script> -v -s -i <COM PORT> -d <Device Name> -a <Address> -f
<live_update_application_image>
<python script>: live_update.py
<COM PORT>: Serial Communication Port
<Device Name>: PIC32MZ
<Address>: Application start address 0x9D100000
Example: python < Live update application folder >\scripts\live_update.py -v -s -i COM6 -
d pic32mz -a 0x9D100000 -f pic32mz_ef_sk.X.production.bin
Note: Running the help command provides a brief overview of options available as shown
below.
Command: python <python script> --help
<python script>: live_update.py
```

5. The following figure shows the Live Update firmware upgrade output.

Figure 4-3. Live Update Application Firmware Upgrade Output



6. If the Live Update is success, the LED3 will turn ON.
7. In case of error during the Live update in Step 4, the LED3 will turn OFF. Use the following steps to recover from the error:
   7.1. Press and hold the switch SW1 to reset the system, then repeat the step 4 to program the Live Update application.
   Or
   7.2. Reset or power cycle the device and repeat from the Step 4 to program the Live Update application.
8. If Live Update is successful, the LED1 or LED2 will start blinking which indicates the Live Update application is running from BANK 1 or BANK 2 based on where the program is running.
   – LED1 for BANK 1
   – LED2 for BANK 2
9. Reset or power cycle the device for the newly programmed Live Update application to run.

– Observe the change in the LED toggling compared to the LED which was toggling in the previous step. This indicates that the newly updated image is running properly.

5. Conclusion

This document discussed the implementation of a Live Update application. The Live Update application enables enhancing the application functionality, making it easier to introduce new features while the device continues to run the existing version of the application software. This allows devices to stay longer in the field and helps lower the maintenance cost of the product. It also enables scheduling periodic updates, which aides in easier deployment of the product.

This document can be used as a reference to implement the Live Update features in the end user application.

6. References

- MPLAB Harmony v3 bootloader:
  < Harmony framework download folder>\bootloader\doc\help_bootloader.chm
- MPLAB Harmony v3 Bootloader applications:
  < Harmony framework download folder>\bootloader_apps_uart\apps
- Getting Started with Harmony v3 Peripheral Libraries on PIC32MZ EF MCUs:
  microchipdeveloper.com/harmony3:pic32mzef-getting-started-training-module
- MPLAB® Harmony v3 Getting Started Articles and Other Documents:
  www.microchip.com/mplab/mplab-harmony/mplab-harmony-articles-and-documentation
- MPLAB® Harmony v3 Configurator Overview:
  microchipdeveloper.com/harmony3:mhc-overview
- Getting Started with Harmony v3 Drivers and Middleware on PIC32MZ EF MCUs Using FreeRTOS:
  microchipdeveloper.com/harmony3:pic32mz-get-start-tm-drvr-middlware-freertos
- Getting Started with Harmony v3 Peripheral Libraries on PIC32MZ EF MCUs:
  microchipdeveloper.com/harmony3:pic32mzef-getting-started-training-module
- MPLAB® Harmony v3 SD Card Audio Player/Reader Tutorial:
  microchipdeveloper.com/harmony3:audio-player
- Graphics Quick Start Applications for PIC32MZ and SAM MCUs:
  github.com/Microchip-MPLAB-Harmony/gfx/wiki/Application-QuickStart
- MPLAB Harmony USB Stack:
  microchip-mplab-harmony.github.io/usb/frames.html?frmname=topic&frmfile=index.html
- Create Your First USB Device CDC Single Application:
  github.com/Microchip-MPLAB-Harmony/usb/wiki/Create-your-first-usb-device-cdc-single-application
- Create Your First USB Host MSD Application:
  github.com/Microchip-MPLAB-Harmony/usb/wiki/Create-your-first-usb-host-msd-application
- MPLAB Harmony TCP/IP Help:
  microchip-mplab-harmony.github.io/net/frames.html?frmname=topic&frmfile=index.html
- Create Your First TCP/IP Application:
  github.com/Microchip-MPLAB-Harmony/net/wiki/Create-your-first-tcpip-application

The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods being used in attempts to breach the code protection features of the Microchip devices. We believe that these methods require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Attempts to breach these code protection features, most likely, cannot be accomplished without violating Microchip's intellectual property rights.
- Microchip is willing to work with any customer who is concerned about the integrity of its code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is "unbreakable." Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-7062-5

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4485-5910 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820