Application Note

Live Firmware Update (Without Reset) of C2000 F29x MCUs

Sira Rao, Alex Wasinger

ABSTRACT

This document presents details on Live Firmware Update (LFU) without device reset on devices with A/B swappable flash banks, such as the F29H85x.

Table of Contents

1 Introduction

In applications like server power supplies, the system is desired to be run continuously to reduce downtime. Typically, during firmware upgrades - due to bug fixes, new features, and/or improvements - the system is removed from service causing downtime. This can be handled with redundant modules but with an increase in total system cost. An alternate approach, called Live Firmware Update (LFU), allows firmware to be updated while the system is still operating.

On F29x MCUs, switching to new firmware can be done both with and without a device reset. This user guide is for switching without reset. For an implementation with reset, see Firmware-Over-The-Air Upgrade Example for F29H85x. **Please note that this example is for HS-FS (i.e. nonsecure) devices.**

2 Building Blocks for LFU

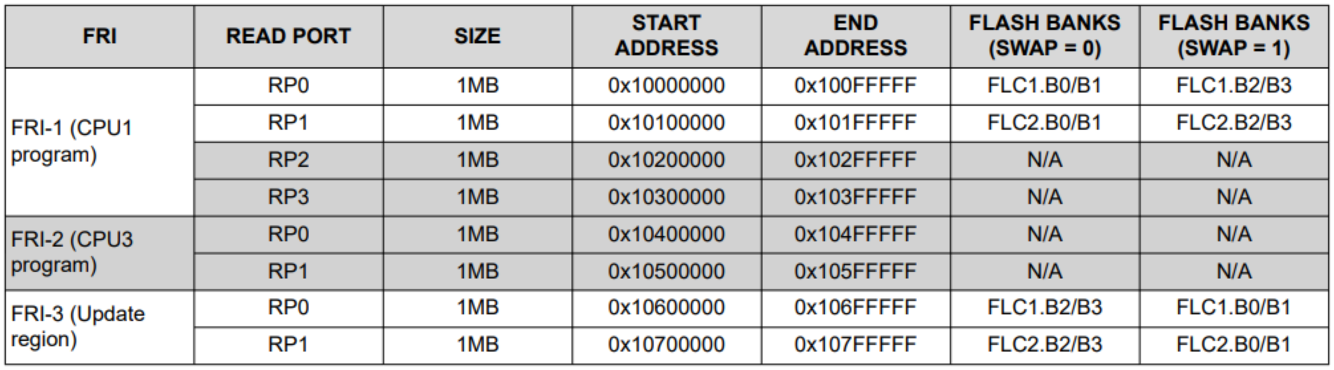
The LFU design consists of several building blocks:

- A desktop host application that issues the LFU command and sends the application image
- An LFU secondary bootloader (SBL) in the target device's flash to communicate with the host and program the application image
  - See the Serial Flash Programming of F29H85x application note for a discussion of an example implementation
- An MCU application that has implemented LFU support
- The target MCU with support for A/B swapping flash banks
- Compiler with LFU support (this will be made available in a future compiler release)

3 Details of Proposed Design

3.1 Flash Bank Organization

LFU is currently only available in bank mode 1 for HS-FS devices, which has the following flash MAIN region address mapping:



FRI	READ PORT	SIZE	START ADDRESS	END ADDRESS	FLASH BANKS (SWAP = 0)	FLASH BANKS (SWAP = 1)
FRI-1 (CPU1 program)	RP0	1MB	0x10000000	0x100FFFFF	FLC1.B0/B1	FLC1.B2/B3
	RP1	1MB	0x10100000	0x101FFFFF	FLC2.B0/B1	FLC2.B2/B3
	RP2	1MB	0x10200000	0x102FFFFF	N/A	N/A
	RP3	1MB	0x10300000	0x103FFFFF	N/A	N/A
FRI-2 (CPU3 program)	RP0	1MB	0x10400000	0x104FFFFF	N/A	N/A
	RP1	1MB	0x10500000	0x105FFFFF	N/A	N/A
FRI-3 (Update region)	RP0	1MB	0x10600000	0x106FFFFF	FLC1.B2/B3	FLC1.B0/B1
	RP1	1MB	0x10700000	0x107FFFFF	FLC2.B2/B3	FLC2.B0/B1

Figure 3-1. Flash MAIN Region Address Mapping

When SWAP = 0, the active flash banks are FLC1.B0/B1 and FLC2.B0/B1. When SWAP = 1, the active flash banks are FLC1.B2/B3 and FLC2.B2/B3. Both the active and inactive flash banks are split into sections for the SBL and the application space, shown below:

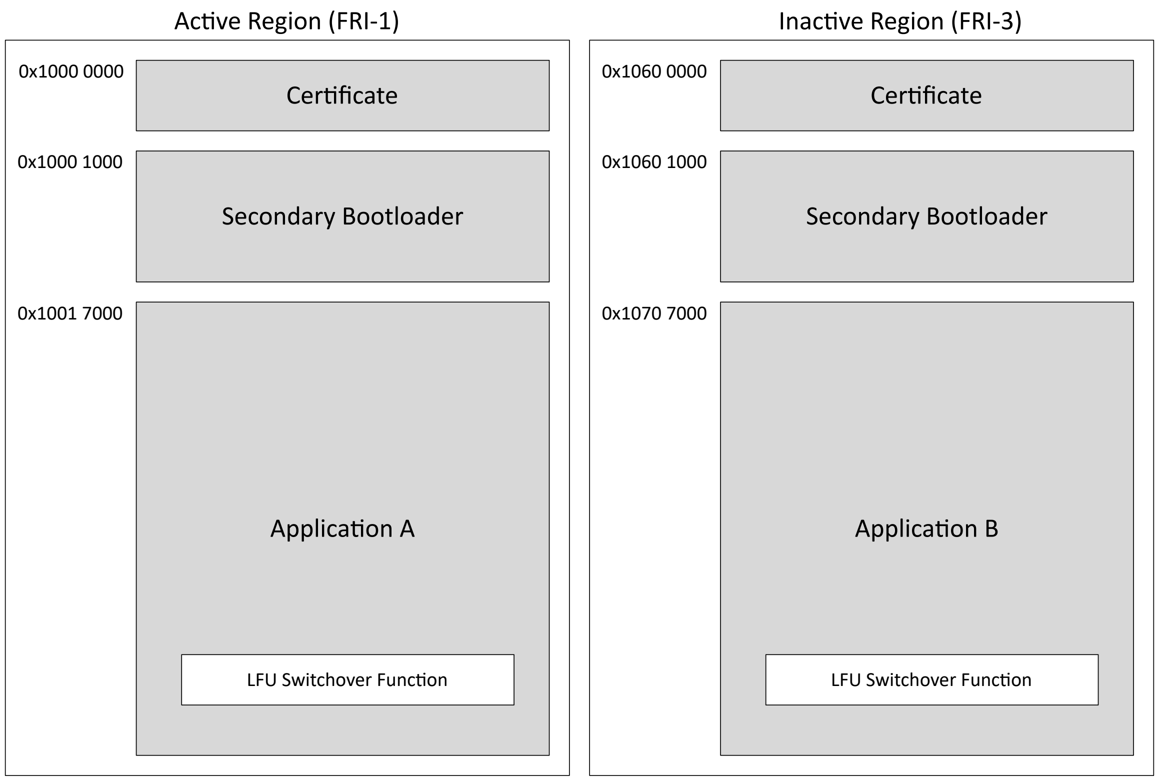


Figure 3-2. Flash Banks

When performing a firmware update, the inactive flash bank is erased and programmed with a new certificate, SBL, and application image.

3.2 LFU Concepts and Factors Impacting Performance

The key considerations when creating LFU-ready firmware are operational continuity during LFU and LFU switchover time. These two are closely related. Operational continuity is achieved through the persistence of state, keeping existing static and global variables in RAM at the same addresses between firmware versions and avoiding re-initialization of those variables when the new firmware is activated. LFU switchover time is the period the device spends with interrupts disabled while changing the firmware and is minimized by A/B swapping flash banks and vector tables.

3.3 Hardware Support for LFU

3.3.1 A/B Swappable Flash Banks

As shown in *Figure 3-1*, there is hardware support for an active and inactive flash bank. These banks can be swapped at any time by XORing the SSU_GEN_REGS.BANKMAP register.

3.3.2 Interrupt Vector Table Swap

Updating the interrupt vector table is one of the most time-consuming parts of the LFU routine. To reduce the switchover time, a shadow vector table is updated while application execution is in progress. During the switchover period, while interrupts are disabled, the shadow and active vector tables are swapped using the PIPE_REGS.INT_VECT_MAPPING register.

4 Application LFU Flow

In the following, when referring to code and the example projects, SBL = flash_based_uart_sbl_with_lfu and LFU Application = lfu_uart_cpu1_application.

The detailed steps associated with LFU are outlined below:

1. *Boot to the SBL and execute the application*: On a device reset, execution starts in the SBL. The communication peripheral is initialized and execution branches to the application's codestart routine. In code, this occurs in the SBL's main() function (flash_based_uart_sbl.c).
2. *Initiate LFU*: The user invokes LFU on the target MCU through a host initiated LFU command.
3. *Receive LFU command in the application*: The application receives the LFU command in its communication peripheral ISR. In code, this occurs in the LFU Application's SBL_uartNotifyISR() (uart_led_blinky_cpu1.c), which sets a flag to tell the application to jump back into the SBL to process the command.
4. *Process the LFU command*: The application branches back into the SBL to parse and process the LFU command. In code, this occurs in the SBL's commandJumpTable() function (flash_based_uart_sbl.c), which reads the command and executes the appropriate operations.
5. *Download the new firmware and program to flash*: The LFU flow in the SBL receives the application image from the host and programs it into the inactive flash bank. Background task functions in the old firmware have stopped executing, but control ISRs continue to be available to keep the application functionality unaffected. In code, this happens in the SBL's cpu1LFUFlow() function (sbl_command_flow.c).
6. *Return to application and report success*: If the firmware image was programmed successfully, the SBL branches back to the application and reports a successful LFU command
7. *Copy the LFU switchover function into RAM*: The application copies the new firmware's LFU switchover function from inactive flash, which is not executable, into RAM, then branches to it. As the old firmware performs the copy of the new firmware's switchover function, it must be located at a fixed address. In code, the LFU Application's performLFUSwitchover() function (lfu_switchover.c) is responsible for this.
8. *Initialize the new PIPE vector table*: The shadow PIPE vector table is populated with the new firmware's ISRs. In code, this is done in the LFU Application's lfuSwapBanks() function (lfu_switchover.c).
9. *Wait for the optimal LFU switchover point*: A simple state machine with software flags is used to determine the end of a control ISR and the beginning of idle time. This is the optimal time to perform the switchover as it maximizes the use of idle time between control loop interrupts. In code, the lfuSwapBanks function (lfu_switchover.c) accomplishes this by reading the lfu_switchover_proceed flag, which is set at the end of the INT_myCPUTIMER0_ISR() routine (uart_led_blinky_cpu1.c).
10. *Execute the LFU switchover*:
    a. Interrupts are disabled
    b. The interrupt tables are swapped
    c. The active and inactive flash banks are swapped
    d. The stack pointer is reinitialized
    e. Interrupts are re-enabled
    f. Execution branches to the new firmware's main() routine

    In code, this is the section of the LFU Application's lfuSwapBanks() function (lfu_switchover.c) marked as the time-critical phase.
11. Device initialization is skipped in the new firmware, and the background control loop starts executing immediately

5 Results and Conclusion

Figure 5-1 shows the actual LFU switchover. The top waveform represents the LFU switchover and the bottom waveform shows the ISR CPU load. The switchover occurs in 300ns, or about 60 CPU cycles.

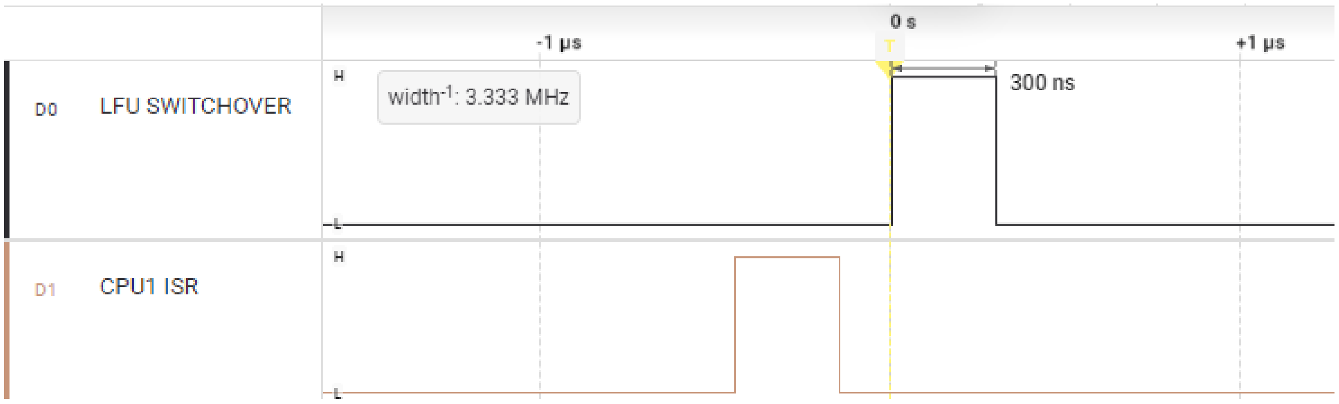


Figure 5-1. LFU Switchover

6 Example Implementations

F29H85x

Available in the F29 SDK at: F29_SDK/examples/driverlib/single_core/flash/flash_based_sbl_with_lfu

7 Summary

This application note demonstrates the systematic implementation of LFU for real-time control applications and specifically high availability systems needing operation without downtime. Switchover to new firmware is able to be completed within 60 CPU clock cycles with the available LFU building blocks, including a novel application LFU software flow and hardware LFU support.

8 References

1. Texas Instruments, F29H85x and F29P58x Real-Time Microcontrollers Technical Reference Manual , technical reference manual.
2. Texas Instruments, F29H85x, F29P58x, and F29P32x Real-Time Microcontrollers Datasheet, datasheet.
3. Texas Instruments, Serial Flash Programming of F29H85x, application note.
4. Texas Instruments, Live Firmware Update Without Device Reset on C2000 MCUs, application note.