AN14165

Flash Memory Swap Feature on KE17Z512

Rev. 1.0 — 11 January 2024 Application note

Document information

Information	Content
Keywords	AN14165, flash swap, reliable firmware update, kinetis, KE17Z512
Abstract	This application describes how to use flash swap feature on KE17Z512 series.

1 Introduction

The flash swap function, which is accessible on certain Kinetis microcontroller components, serves as a strong tool for dependable software enhancements. This document discusses how to employ the swap feature effectively. Frequently asked queries are responded to, and a practical use case is illustrated.

This document is aimed at system coders who are building applications that can apply the flash swap feature. To obtain more critical details about the flash swap, consult the user manual of the device mentioned in this document.

2 Firmware update architecture

This section describes the differences and advantages between the traditional firmware update architecture and the firmware update architecture supported by KE17Z hardware.

2.1 Traditional remote update systems

Conventional systems for remotely upgrading firmware depend on a bootloader program that springs into action when the system resets. This program either decides which application must be run or kick starts a procedure for updating an application. A distant server transmits fresh firmware to the system, which is then inscribed into its local storage.

Usually, the process of updating the system triggers a pause in the operation of the primary application. The core application code is then deleted and rewritten. The main application has only one version, therefore, any unnoticed errors during the receipt and programming of the new code can lead to incorrect system functioning. In extreme cases, the system can cease to function until a new application is loaded. The gravest situation arises when the system becomes unresponsive, and it is impossible to force it into bootloader mode to facilitate system updates.

Advantages:

- Simple to implement
- Less flash memory is required because no backup copy of the application is maintained

Disadvantages:

- The main application must stop during the update process
- Not possible to go back to a known working application
- Cannot be tolerant of power loss during the update process

2.2 Systems with code backup (dual image)

A system can be implemented with a complete software backup held in memory. If a serious error is detected, the system can revert to the backup copy of the main application. A bootloader can be used to select the correct copy of the main application to execute.

Advantages:

- In a multitasking operating system, it is possible to continue to execute the main application tasks while background tasks are running to update the new copy of the application.
- Backup copy of code; possible to revert to the known working application.

Disadvantages:

- More memory space is required to store a backup copy.
- A bootloader is required (a startup routine that selects which application to run).

2.3 Systems using flash memory swap

In devices with two or more internal flash blocks that support swap, the memory base address of each flash block can be exchanged. The address location of each flash block can therefore be swapped in the logical memory map of the device. After a reset, the built-in flash swap system essentially selects which software executes by the location of the flash block in the logical memory map. This allows a code back-up system with the added ease of programming. You can execute out of one block while erasing/programming the other block. On Kinetis devices, the flash swap system monitors/controls all the steps on how to switch from the old application to the new. There is an added assurance of reliable operation in the case of a power loss during one of those steps.

Advantages:

- Ease of programming. The application always executes out of the lower block in the memory map.
- Power loss tolerant.
- No bootloader is required. No delay to start the main application.
- Well suited for a multi-tasking OS. Minimal application downtime. In a multi-tasking system, it is possible to continue to execute the main application tasks while background tasks are running to update the new copy of the application.
- Backup copy of code; possible to revert to the known working application.

Disadvantages:

- More flash memory space is required to store a backup copy.

3 Program flash swap overview

Kinetis devices are available with internal flash memory for program and data storage. For devices with two or more program flash (p-flash) memory blocks that have the p-flash swap feature, the system programmer can configure the logical memory map of the p-flash space such that either of the two physical p-flash blocks can exist at relative address 0x0000. The flash swap feature is not mandatory.

This is useful because on Kinetis devices, the default vector table is at address 0x0000. When the processor exits reset, it fetches the initial stack pointer (SP) from address 0x0000 and the program counter (PC) from address 0x0004. Therefore, swapping the base address of the two p-flash blocks allows the system to either boot from p-flash block 0 or p-flash block 1 because either block can be at base address 0x0000.

The swap system is controlled by issuing swap control commands that are similar to other flash commands on Kinetis devices. The swap system requires initialization before a swap is possible.

3.1 Key facts about p-flash swap

The following are the facts about the p-flash swap:

- The block currently at the relative address 0x0000 is known as the active block. It is also referred to as the lower block. The other block is known as the nonactive block or upper block.
- The typical use case is for the system to execute from one block (active block) while reprogramming the other block (nonactive block). In most applications, the system application code is limited to fit within one flash block (active block).
- After reprogramming the nonactive block and when it is ready for execution, the user can complete the swap process to swap the blocks. The nonactive block becomes the active block and vice versa.
- A system reset begins code execution from the active block (the block at base address 0x0000).
- The same procedure for swapping back and forth between blocks is used.

3.2 KE17Z512 flash swap details

KE17Z512 devices with two p-flash blocks support the swapping of the two blocks. The `SWAP` bit of the `FTFL_FCNFG` register indicates which p-flash block is at address 0x0000. The swap system sets the state of the SWAP flag within the flash module during the reset sequence.

The `SWAPPFLSH` bit in the `SIM_FCFG2` register indicates whether the swap system has been initialized and therefore potentially ready to be swapped.

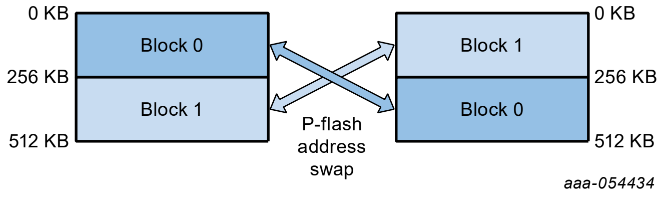

Figure 1. P-flash swap (KE17Z512)

3.3 Swap command overview

The swap command has the following four options:

- Set Swap Indicator Address (Initialize Swap system)
- Set Swap in Update (Prepare) Mode
- Set Swap in Complete Mode
- Report Swap Status

Set Swap Indicator Address Command:

This command initializes the swap system. When issuing this command, you must provide an address for the flash swap indicators. This command sets the swap enable word and swap indicator address, which is used by the swap system. This step is required only once when a swap is performed for the first time.

Set Swap in Update Mode:

This command programs the flash swap indicator value to indicate the system that content updates of the upper block have been planned. Also, it unprotects the sector holding the flash swap indicator within the nonactive block to allow it to be erased. Erasing this sector during the Update (or Update-Erased) state is a requirement of the swap process.

Set Swap in Complete Mode:

This command programs the flash swap indicator value to indicate the system that reprogramming the nonactive block is complete and ready to swap the flash blocks. This command is to be issued after the contents of the nonactive (upper) block are erased/reprogrammed as required to update the system software. It includes erasing the sector in the nonactive block that contains the flash swap indicator.

Report Swap Status:

This command verifies the status of the swap system, providing the following:

- Current swap state
- Status of the current swap block (the block currently at address 0x0000)
- Status of the next swap block (the block that is going to be at 0x0000 after a system reset)
- Any reported errors

Issue this command in the system initialization code that executes on a system reset. It helps the system software identify any errors with the swap system, which the power loss can potentially cause during a swap command execution.

4 Swap steps

This section describes the steps for the detailed swap using the flash swap feature of the KE17Z.

4.1 Swap procedure

The procedure for swapping the flash blocks is simple. The same procedure is used for swapping back and forth between the blocks.

4.2 First swap

To perform the first swap, perform the following steps:

1. Initialize the system by issuing the initialization command. This step is required only once when a swap is performed for the first time. When first initialized, the swap system sets to Update-Erased.
2. Erase the nonactive (upper) block.
3. Reprogram the nonactive (upper) block with new software.
4. Issue the command to set the system to the Complete state. The swap takes effect after a reset (including a software reset).
5. After a reset, the blocks are swapped, and the swap system is set to the Ready state.

4.3 Procedure for more swaps

After the first swap has been completed, the process starts with the Ready state.

1. Issue the command to set the system to the Update state.
2. Erase the nonactive (upper) block. Once erase is complete, the system automatically moves to the Update-Erased state.
3. Reprogram the software in the nonactive block.
4. Issue the command to set the system to the Complete state.
5. Reset the microcontroller (which is any reset including software reset).
6. After a reset, the blocks are swapped, and the swap system is set to the Ready state.

4.4 Erasing the nonactive upper block

You can use block erase or sector erases. It is only required to erase the flash swap indicator sector in the nonactive block. However, to update the software in the nonactive block, ensure to first erase it. Once erase is complete, the system automatically moves to the Update-Erased state.

Swapping between blocks without erasing/reprogramming the entire nonactive block is possible. Ensure to erase the sector with the flash swap indicator (when in the Update state). It is useful for swapping back to a known good application in the nonactive block.

4.5 Order of the steps

NXP recommends that the new code must be uploaded to the nonactive (upper) block when the swap system is in the Update-Erased state and before the system is moved to the Complete state. Therefore, if a power loss occurs during the process, the swap system knows that it has been in the middle of updating and must revert to the previous swap state. The previous swap state implies the good state.

5 Software example

A software example is provided with this application note to demonstrate the swap feature.

5.1 Running the demo

This section describes the steps to run the demo.

5.1.1 Prerequisites

To run the demo, the following prerequisites are required:

- The KE17Z512 flash has a total size of 512 kB; divided into 2 physical blocks each of 256 kB, with a sector size of 2 K.
- Physical address: 0x00000 - 0x40000 (0 K - 256 K) is called lower block (LB).
- Physical address: 0x40000 - 0x80000 (256 K - 512 K) is called upper block (UB).
- Logical address: The address the CPU perceives; if there is no FlashSwap function, then the logical address is the same as the physical address.

5.1.2 FlashSwap

The FlashSwap implies that the user can execute an operation to map the logical address to the LB or UB. For instance, by default, the code resides in the LB region starting from address 0, and the logical address is also 0.

Once a swap is initiated, on the next startup, the logical address 0 is already mapped to UB (physical: 0x40000). For the CPU, the address 0 is already the physical address (0x40000).

If the SWAP is executed again, it switches back, ping-pong each other, and so on.

5.1.3 Running the code

After burning the code to the board, connect the serial port and press reset, the following will be printed:

~~~
```
PFlash Swap Example Start
APP VERSION location offset:0x24
APP VERSION:0
CURRENT SWAP STAT: LowerAtZero(Physical Address:0x0, mapped to Logical
 0x00000000)
Current swap system status: Uninitialized
swapIndicatorAddress:0x3F800
PLEASE SELECT...
1 - Show flash info
2 - Simulate update firmware
3 - Swap flash and reboot
4 - Reboot
```
~~~

Where:

- `CURRENT SWAP STAT: LowerAtZero(Physical Address:0x0, mapped to Logical 0x00000000)` indicates that the current logical address of the CPU is mapped to physical the address: 0x0000_0000 (equivalent to no FlashSwap)
- Enter 1: Print the basic information of Flash
- Enter 2: Simulate firmware upgrade behavior, which is copying the entire LB IMAGE to UB, only incrementing by 1 at the VPP_VERSION location

- Enter 3: Enable FlashSwap and reboot (takes effect)
- Enter 4: Reboot

6 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

7 Revision history

Table 1 summarizes the revisions to this document.

Table 1. Revision history

Document ID	Release date	Description
AN14165 v.1	11 January 2024	Initial public release

Legal information

Contents

Date of release: 11 January 2024
Document identifier: AN14165