

Offline Flash Programmer for Kinetis K- and L-series MCUs

By: Xi Yang

1 Introduction

Effective and convenient tools for the flash programming are very useful in the production phase of the product design. The traditional offline programming tools are often expensive and may include various limitations such as the restricted image size, unsupported devices, and other.

Therefore, it's valuable to provide an easy and simple way of production flash programming for the Kinetis MCUs.

This application note describes how to implement a proof-of-concept offline flash programmer for nearly all Kinetis devices. The source code and the pre-compiled binary image file are provided in the attached software. This document is intended for users developing the production flash programming hardware, production flash programming software, or a debugger interface that supports flash programming.

Contents

1	Introduction	1
2	Overview	2
2.1	Terms.....	2
2.2	Supported target MCUs	3
3	Hardware Setup	3
3.1	Hardware platform selection.....	3
3.2	Pin connections to target MCU.....	3
4	Software Setup.....	5
4.1	Software structure.....	5
4.2	Flash algorithm.....	6
5	Running the Demo.....	7
5.1	Loading firmware to FRDM-K64.....	7
5.2	Selecting flash algorithm	7
5.3	Adding target image file	8
5.4	Connecting to target.....	9
5.5	Running the demo.....	9
6	Consideration.....	10
6.1	FlexMemory partitioning and EEPROM support ..	10
6.2	Unsecure Kinetis	10
6.3	Target image file.....	11
7	References	11
8	Revision History	11

2 Overview

This document describes how to implement a proof-of-concept offline flash programmer for the Kinetis MCUs and the low-cost methods for the production flash programming. The key features include:

- NXP FRDM-K64 as the hardware platform.
- Support for most of the Kinetis K and Kinetis L series MCUs.
- Support for the program data flash.
- The image binary file and the flash algorithm configuration file stored on the SD card.

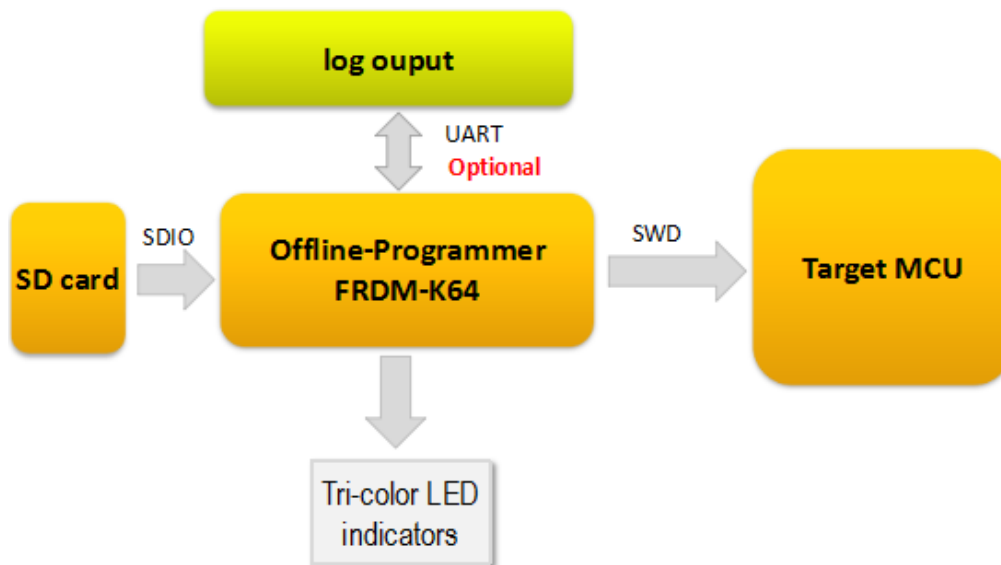


Figure 1. Offline programmer overview

Before reading this document, see *Production Flash Programming Best Practices for Kinetis K- and L-series MCUs* (document [AN4835](#)) to gain knowledge of the SWD debug and flash programming for the Kinetis series.

2.1 Terms

This table summarizes the meaning of the terms used in this document:

Table 1. Terms

Term	Meaning
Offline programmer	The tools that can store the target image file and load the file into the target MCU's internal program flash or data flash.
Target MCU	The MCU on the target board to download the image to.
TRST	The target MCU's reset pin.
Target image file	The binary file to load into the target MCU.

2.2 Supported target MCUs

The demo software attached to this document supports these target MCUs:

- Kinetis K0/1/2/3/4/5/6/7/8/x MCUs.
- Kinetis KL1/2x MCUs.

3 Hardware Setup

3.1 Hardware platform selection

The NXP Freedom development platform FRDM-K64 board is selected as the offline programmer hardware platform. This platform is popular among the developers. This is the list of the on-board peripherals used in this demo:

- SD card socket—used to store the target image and configuration files.
- GPIO pin socket—used to lead out the SWDIO/SWDCLK and TRST pins.
- Tricolored LED—used as the system status indicator.
- OpenSDA—used as the debugger and flash downloader for the K64; it can be also used as the USB-UART bridge.

3.2 Pin connections to target MCU

Four wires need to connect the offline programmer to the target MCU, as shown in this table:

Table 2. Target pin connection

Name	FRDM-K64	Target	Comments
SWD_DIO	PTB3 (J4-4)	SWD_DIO	Connected to the target MCU's SWD_DIO pin
SWD_CLK	PTB2 (J4-2)	SWD_CLK	Connected to the target MCU's SWD_CLK pin
TRST	PTB10 (J4-6)	RESET	Connected to the target MCU's reset pin
GND	GND (J3-14)	GND	GND must be connected

Besides the connections, make sure that:

- The target MCU must be powered before connecting the SWD and TRST pins.
- The target MCU's reset pin signal is clean and is not controlled by any other external signal (another debugger, key button).
- If you use the NXP Freedom and Tower System modular development platform boards, cut the connection between the target MCU and the on-board OpenSDA (SWD_IO, SWD_CLK, and TRST). You can do this in any of these ways:
 - If the FRDM/TWR board has a hardware jumper/0-Ω resistor, just change the jumper setting or remove the resistor. For example; on the FRDM-K22 board, remove R58 and disconnect J13 and J10:

Hardware Setup

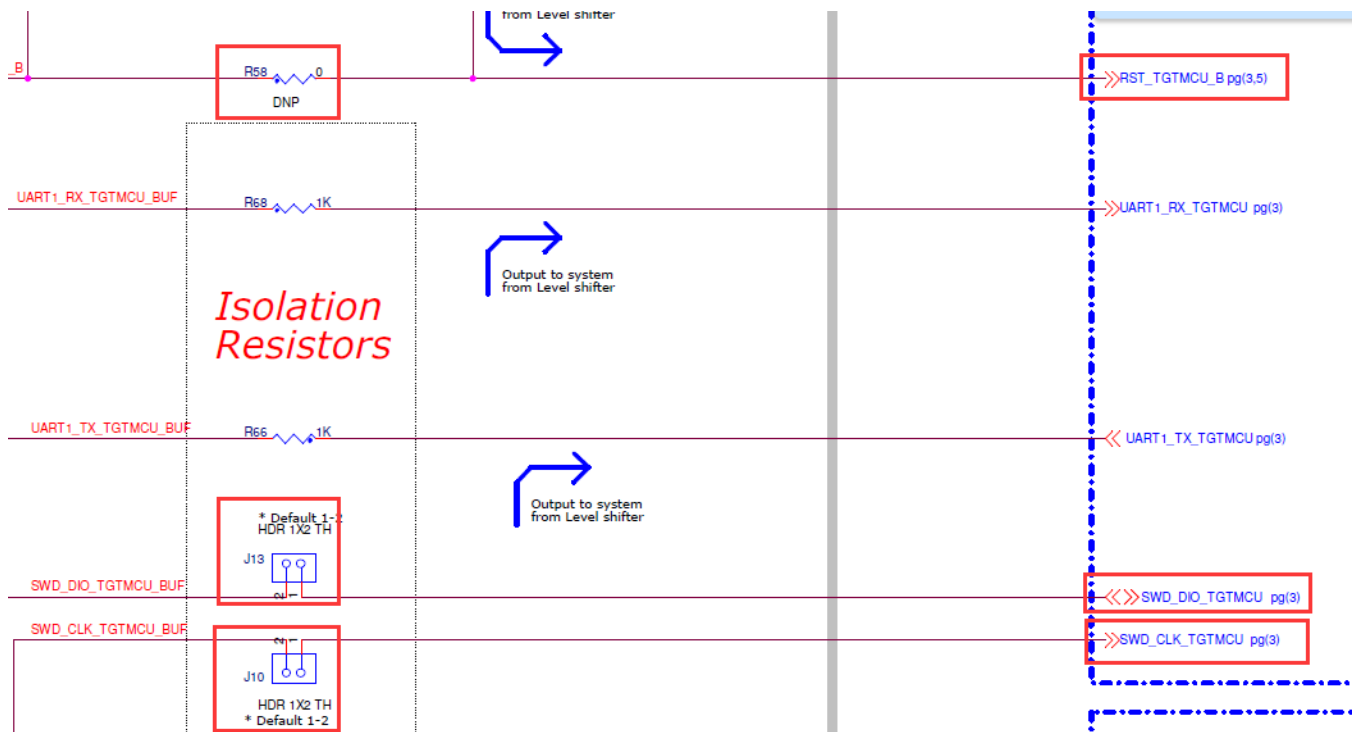


Figure 2. Disconnecting OpenSDA from target MCU

- If the FRDM/TWR board does not have the hardware jumper/resistor, erase the OpenSDA's firmware or put the OpenSDA into the bootloader mode. This sets the OpenSDA's target debug pins into the input states.

This figure shows a typical connection of the offline programmer (the target MCU board is the FRDM-KL26):

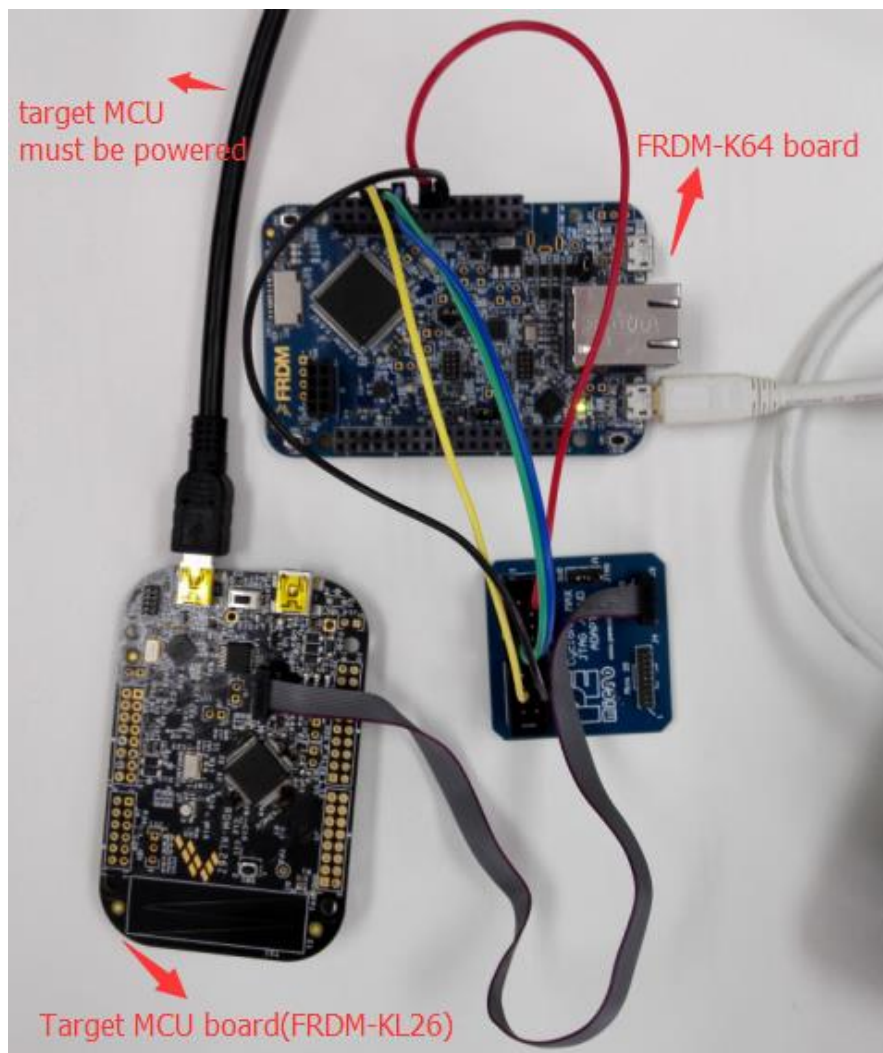


Figure 3. Hardware connection

4 Software Setup

4.1 Software structure

The offline programmer software is developed using KSDK2.0. It can be divided into several functional modules:

- SWD protocol implementation and DAP access—these codes are ported from the CMSIS-DAP source code (see github.com/mbedmicro/CMSIS-DAP).
- Flash algorithm—includes the target flash programming code and the WDOG disable sequence. This position-independent code is downloaded to the target RAM and executed by the target MCU when initializing and programming the target flash.

- The SD card driver, FAT32 driver code, and all other chip peripheral drivers, startup file, and header files are reused from the *SDK2.0\frdmk64f\driver_examples\sdcard_fatfs* demo.

The overall software workflow is shown in this figure:

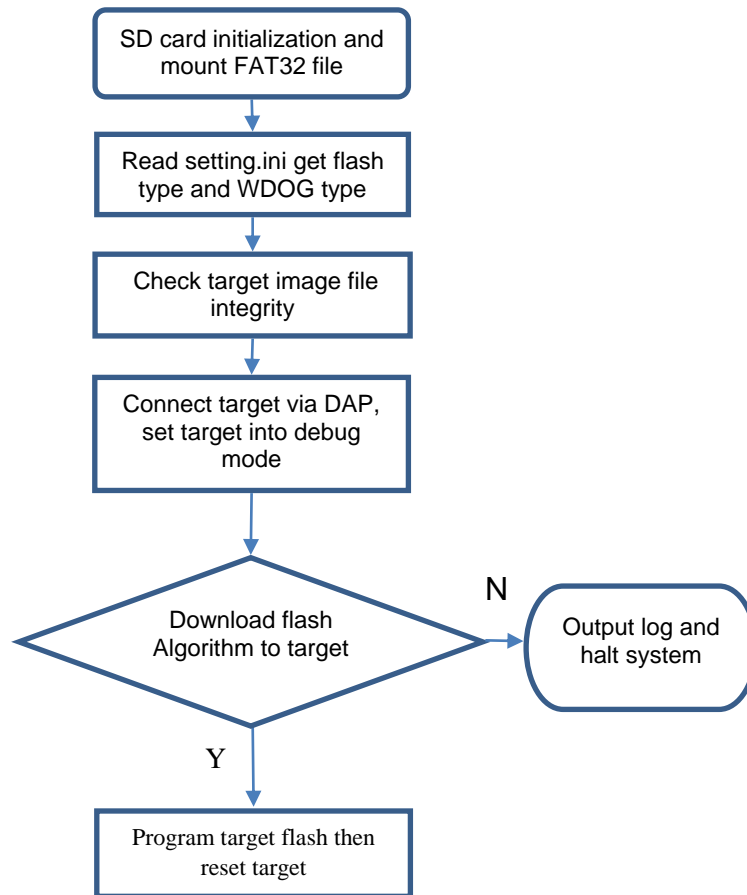


Figure 4. Software workflow

4.2 Flash algorithm

The common flash algorithm that supports most MCUs from the Kinetis K and KL series is needed for the offline programmer. You must have control over the target MCU's core to run the flash algorithm. Therefore, disable the target MCU's WDOG. There are some differences between the Kinetis series WDOG and flash modules:

- WDOG disable sequence—most Kinetis series use four different types of WDOG modules, as shown in the following table.

Table 3. WDOG type

Option	Name	Description	Examples
1	WDOG8	8-bit WDOG module	KE0x
2	WDOG16	16-bit WDOG module	Most of the Kinetis K/V/M series MCUs
3	WDOG32	32-bit WDOG module	KE1x and KL28
4	COP	WDOG integrated in the SIM module (called COP)	Most of the Kinetis KL series MCUs

- Flash sector size—program flash sector size. If the target MCU has the FlexNVM, the data flash sector size is also included. In most parts, the data flash size and the program flash size are the same. The flash sector size can be either 512, 1024, 2048, or 4096 bytes.
- Flash program command—after a flash sector (or the entire flash) is erased, the programming can start. There are three different flash-programming commands:
 - Program section.
 - Program longword (32-bit, PGM4).
 - Program phrase (64-bit, PGM8).

Although the program section command is the fastest way to program the flash, it is only supported on certain devices. All devices support either the PGM4 command or the PGM8 command, but not both. To maintain compatibility, the demo uses either the PGM4 or PGM8 commands to program the flash.

Different combinations of the WDOG type, flash sector size, and flash program command can be applied to one or many specific target MCUs. In the source code, this combination is pre-defined as the flash option. See the following section for more information.

5 Running the Demo

5.1 Loading firmware to FRDM-K64

Load the firmware provided in the attached software package into the K64. It's the same as with all SDK2.0 demo applications.

5.2 Selecting flash algorithm

Select the correct target MCU flash algorithm and the WDOG disable sequence. You can do this by modifying (creating) the *setting.ini* file on the SD card:

- Create/modify the *setting.ini* file in the root directory of the SD card.
- Type this into the file:

```
[TARGET]
FLASH_OPTION=X
```

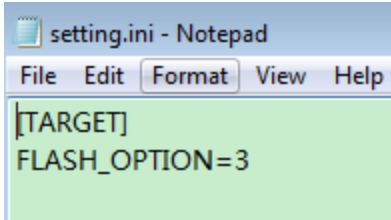


Figure 5. Flash algorithm configuration file

where X is the flash option.

The flash options are summarized in this table:

Table 4. Flash algorithm selection

Flash option	WDOG type	WDOG base	Sector size	Program command	Examples
0	WDOG16	0x40052000	2048	PGM4	KS22FN256, KS22FN128, MK60DN512
2	WDOG32	0x40076000	2048	PGM4	KL28
3	WDOG16	0x40052000	1024	PGM4	MK20DX128/DN128
4	WDOG32	0x40052000	2048	PGM8	MKE15
5	COP	0x40047000	1024	PGM4	KL15/16/17/25/26/27
6	WDOG16	0x40052000	4096	PGM8	MK2/6xFN1M/FX512
7	WDOG16	0x40052000	4096	PGM4	MK8xFN256

For the COP, the WDOG base refers to the SIM module base address. If you don't find the target MCU in the "Examples" column, just find a similar part where the WDOG, sector size, and program command are all the same as on the target MCU.

5.3 Adding target image file

The offline programmer can accept the raw binary files only in the current version. These two image files must be copied into the SD card root directory with fixed file names:

- *PIMAGE.BIN*—program flash binary image.
- *DIMAGE.BIN*—data flash binary image.

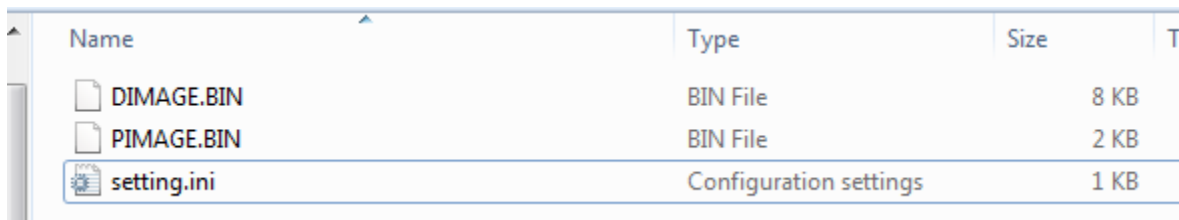


Figure 6. SD card root directory screenshot

The *.bin* files can be generated by the IDE post-compile command. If target MCU does not have a data flash or there is no need to program the data flash, just remove the *DIMAGE.BIN* file.

Put the SD card into the FRDM-K64's SD socket, connect the OpenSDA's USB to the PC, open the serial terminal with 115200, N8N1 and power on the board.

5.4 Connecting to target

Connect the SWDIO/SWDCLK/TRST and GND signals to the target MCU. Power the target board as described in [Section 3, "Hardware Setup"](#).

5.5 Running the demo

Press the reset pin on the FRDM-K64 to start the demo and the on-board three-color LED indicates the system status. When running the demo for the first time, look at the PC terminal log to debug and trace the errors.

The blue LED lights and the system halts if one of these conditions occurs:

- The SD card initialization failed.
- The FAT32 file system mount failed.
- No target image file (wrong file name/path) or configuration file detected.

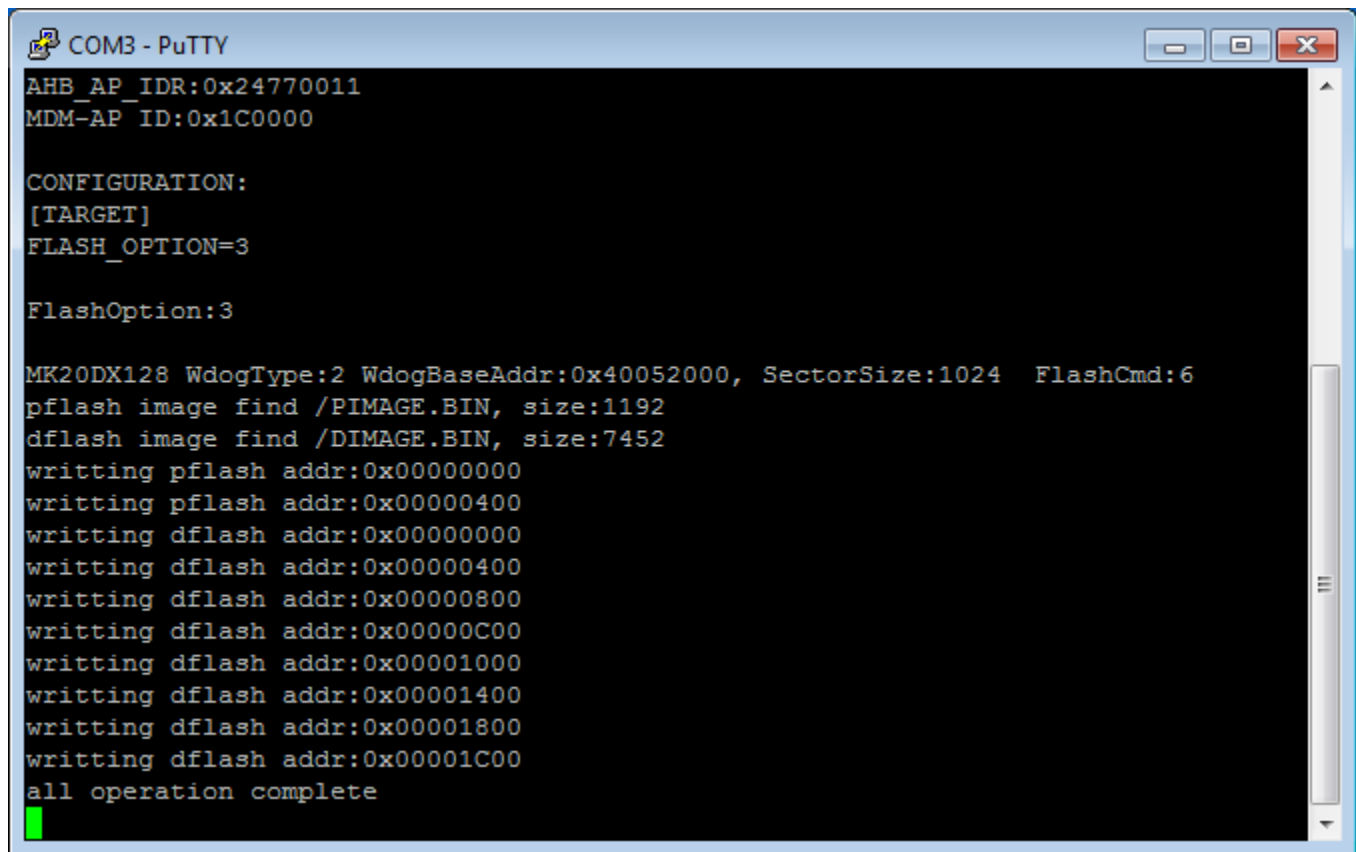
The red LED lights and the system halts if one of these conditions occurs:

- Failed to connect to the target MCU.
- Failed to download the flash-programming algorithm into the target RAM. This can be caused by a wrong flash algorithm option selection, a secured chip (mass erase is disabled), or a hardware connection issue.
- Failed to program the image file data into the target flash.

The green LED blinks if one of these conditions occurs:

- The offline programmer is programming the target flash. The green LED blinks at a frequency of 3~20 Hz, according to the SWD clock cycle and the target flash sector size.
- The offline programmer finishes all tasks successfully. The green LED blinks at a frequency of 1 Hz to indicate that all tasks completed successfully and you can move to the next target and press the FRDM-K64's reset pin to start a new load task.

This figure shows the log information when the load into the target MCU is successful:



```

COM3 - PuTTY
AHB_AP_IDR:0x24770011
MDM-AP ID:0x1C0000

CONFIGURATION:
[TARGET]
FLASH_OPTION=3

FlashOption:3

MK20DX128 WdogType:2 WdogBaseAddr:0x40052000, SectorSize:1024 FlashCmd:6
pflash image find /PIMAGE.BIN, size:1192
dflash image find /DIMAGE.BIN, size:7452
writting pflash addr:0x00000000
writting pflash addr:0x00000400
writting dflash addr:0x00000000
writting dflash addr:0x00000400
writting dflash addr:0x00000800
writting dflash addr:0x00000C00
writting dflash addr:0x00001000
writting dflash addr:0x00001400
writting dflash addr:0x00001800
writting dflash addr:0x00001C00
all operation complete

```

Figure 7. Output log

6 Consideration

6.1 FlexMemory partitioning and EEPROM support

Use the flash program partition command (PGMPART) to set up the memory partitioning and configure the EEPROM for use. After the EEPROM is configured and (optionally) loaded, the data can be programmed into the remaining data flash. Be careful when programming the data flash and do not attempt to program the area which is already initialized as the EEPROM. See the flash section in the chip reference manual for more information about the program partition.

6.2 Unsecure Kinetis

The source code provided with this document contains unlocked Kinetis sequences. The secure Kinetis can't accept most flash commands, such as the erase sector or program flash commands. If a secure Kinetis is detected, the offline programmer tries to unlock the target MCU (by performing a mass erase and deleting all data) without a notice. If the target MCU Mass Erase Enabled bit = 0, the offline programmer can't perform the mass erase operation and the system halts.

6.3 Target image file

Make sure that the target image files are correct. If not, the loading process may secure the target MCU. The demo software does not check the file size or the target MCU flash size. If the target image file is bigger than the target MCU flash, a loading error occurs and the system halts.

7 References

1. *Production Flash Programming Best Practices for Kinetis K- and L-series MCUs* (document [AN4835](#))
2. *ARM CoreSight™ Components Technical Reference Manual* available at www.arm.com
3. KINETIS-SDK: Software Development Kit for Kinetis MCUs available at www.nxp.com/ksdk

8 Revision History

This table summarizes the changes done to this document since the initial release:

Table 5. Revision history

Revision number	Date	Substantive changes
0	09/2016	Initial release.

How to Reach Us:

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including "typicals," must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address:

nxp.com/SalesTermsandConditions

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale, the Freescale logo, Kinetis, Tower, and Freedom are trademarks of NXP B.V. All other product or service names are the property of their respective owners.

ARM, the ARM Powered logo, Cortex, and CoreSight are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved.

© 2016 NXP B.V.

Document Number: AN5331

Rev. 0

09/2016

