Using Serial Flash on CC3120 and CC3220 SimpleLink™ Wi-Fi® and Internet-of-Things Devices

ABSTRACT

This application note is divided into two parts. The first part provides important guidelines and best-practice design techniques to consider when choosing and embedding a serial flash paired with the CC3120 and CC3220 (CC3x20) devices. The second part describes the file system, along with guidelines and considerations for system designers working with the CC3x20 devices.

The CC3120 and CC3220 devices are part of the SimpleLink™ microcontroller (MCU) platform, which consists of Wi-Fi®, *Bluetooth*® low energy, Sub-1 GHz and host MCUs. All share a common, easy-to-use development environment with a single core software development kit (SDK) and rich tool set. A one-time integration of the SimpleLink platform lets you add any combination of devices from the portfolio into your design. The ultimate goal of the SimpleLink platform is to achieve 100 percent code reuse when your design requirements change. For more information, visit www.ti.com/simplelink.

Contents

Trademarks

SimpleLink is a trademark of Texas Instruments.
Bluetooth is a registered trademark of Bluetooth SIG.
Wi-Fi is a registered trademark of Wi-Fi Alliance.
All other trademarks are the property of their respective owners.

1 Factors to Consider When Designing With Serial Flash

Many embedded systems contain a serial flash component to store firmware, configuration files, and user data for usage by a microcontroller (MCU) or processor. The processor sporadically writes data into this serial flash when updating the contents. Inclusion of serial flash memory poses unique challenges for the system designer.

- A typical serial flash has a data endurance of 100K write cycles per sector and a 20-year data retention. The write endurance and data retention characteristics must be considered by the application developer.
- The serial nature of reads and writes results in long access times, increasing the challenge of maintaining a stable system-supply voltage for the duration of the access.

1.1 Serial Flash Vendor and Part Number Selection

Serial flash components from some vendors may appear to be equivalent in memory capacity, but close examination of the serial flash data sheets can reveal significant parametric differences between components in areas such as operating voltage and access times. The serial flash components listed in TI BOM tables for the CC3x20 reference designs can be used, because TI has system-tested these serial flash components. Table 1 lists the different parts tested with the CC3x20 devices. TI found these parts reliable through a series of system-level tests, to ensure robustness under various operating conditions. However, this does not ensure data integrity under extreme operating conditions, as specified in subsequent sections of this document.

More information can be found in the *CC3x20 SimpleLink Network Processor Programmer's Guide* (see the *Design Consideration* chapter).

Table 1. Serial Flash Parts Tested With the CC3x20 Devices

Vendor	Part Number	Size	Voltage Power	Recommendations
Macronix	MX25R3235FM1IL0	32Mb	1.65 V to 3.6 V	Battery and line-powered systems
ISSI	IS25LQ016B	16Mb	2.3 V to 3.6 V	Battery and line-powered systems
ISSI	IS25LQ032B	32Mb	2.3 V to 3.6 V	Battery and line-powered systems

1.2 Supported Flash Types

For compatibility with the CC3x20 device, the serial flash device must support the following commands and format:

- Uniform sector erase size of 4K.
- Command 0x9F (read the device ID [JEDEC]). Procedure: SEND 0x9F, READ 3 bytes.
- Command 0x05 (read the status of the SFLASH). Procedure: SEND 0x05, READ 1 byte. Bit 0 is busy and bit 1 is write enable.
- Command 0x06 (set write enable). Procedure: SEND 0x06, read status until write-enable bit is set.
- Command 0xC7 (chip erase). Procedure: SEND 0xC7, read status until busy bit is cleared.
- Command 0x03 (read data). Procedure: SEND 0x03, SEND 24-bit address, read n bytes.
- Command 0x02 (write page). Procedure: SEND 0x02, SEND 24-bit address, write n bytes ($0<n\leq256$).
- Command 0x20 (sector erase). Procedure: SEND 0x20, SEND 24-bit address, read status until busy bit is cleared.

1.3 Serial Flash Write or Erase Endurance Limitations

The lifespan of a serial flash component is subject to a maximum number of write/erase cycles. This should be considered when designing the system application software.

General guidelines for increasing flash endurance follow:

- Minimize the number of application writes to flash, especially after reset. For example, ensure that application configuration writes to flash occur only at initial reset, and not every time the CC3x20 is reset.
- The creation and deletion of files requires updates to the File Allocation table (FAT). When an update to an existing file is required, do not delete and then recreate the file, because unnecessary access to the FAT should be avoided.]
- More information can be found in the CC3x20 SimpleLink Network Processor Programmer's Guide (see the *Design Consideration* section in the *File System* chapter).

A typical serial flash ensures a data endurance of 100K write cycles per sector, and a 20-year data retention. Table 2 details the maximum number of writes-per-day to the same sector, allowing the device to operate for a given number of years.

Table 2. Serial Flash Endurance

Desired Product Life [Years]	Maximum Writes-per-Day (1)
20	14
15	18
10	27
5	55
2	137

(1) Maximum number of writes each day = 100000 / (product life in years × 365)

In the CC3x20 system, the serial flash may be written by the user application or by periodic activity of the CC3x20 on-chip firmware. The total number of writes from these two sources should not exceed the budget for the maximum number of flash writes calculated for the desired product lifetime.

1.4 Best Practice Design Techniques for System Robustness

1.4.1 Overview

The CC3x20 devices are proven, robust WLAN solutions when operated in accordance with the supply and signal parameters described in the data sheet. This section describes design techniques to maximize system robustness. These techniques include minimizing the possibility of inadvertent corruption of the serial flash memory connected to the CC3x20 devices for battery-powered and hybrid line/battery-powered designs. The techniques described focus on ensuring the integrity of the power supply to the serial flash, to avoid situations where the supply voltage falls below the minimum threshold specified by the serial flash manufacturer, causing corruption of flash data during write or erase operations. The nature of such corruption systems determines whether or not systems continue to operate, and in some cases a physical access to the system may be required to recover.

1.4.2 General Guidelines

General guidelines for achieving system robustness follow:

- Apply primarily to systems that are powered from batteries or by a hybrid line and battery scheme. This includes designs where the CC3x20 device connects directly to the battery, and systems where a DC-DC converter is used between the battery and the CC3x20 device.
- If the application uses a DC-DC converter, designers should ensure that the output of the DC-DC converter satisfies the supply requirements of both the CC3x20 device and the attached serial flash device.
- The CC3x20 devices should be enabled only when the supply voltage is greater than or equal to 2.3 V. This minimum is typically determined by the serial flash minimum supply voltage and not the CC3x20

minimum supply voltage, which is defined in the data sheet as 2.1 V. As specified in the data sheet, the supply must tolerate a CC3x20 transmit current or calibration current load without sagging below 2.3 V; therefore, the unloaded supply voltage may have to be greater than 2.3 V to account for internal resistance effects of the supply.

- The supply voltage applied to the CC3x20 device should never exceed 3.8 V, which is specified as the absolute maximum supply voltage in the data sheet. The corresponding absolute maximum voltage constraints from the chosen serial flash data sheet must also be followed.
- For better flash endurance, follow the guidelines in Section 1.3.
- For maximum system robustness, use a serial flash such as the Macronix MX25R6435. This device supports a wide supply voltage range, which tends to improve system immunity to supply fluctuations.
- The CC3x20 WLAN transmission can result in sudden increases in the loading on the power supply. The sudden increases may result in a momentary decrease in supply voltage. Consult the CC3x20 data sheet section that describes how to handle supply brownout.

1.4.3 Sudden Power Off

All systems using serial flash are vulnerable to the effects of sudden power removal. As noted in most serial flash data sheets, a data corruption may occur if the power is removed while a write or erase operation is in progress. This can occur if the system operating voltage goes below V_{min} of the serial flash (2.3 V typical) before the erase or write operation is completed. Figure 1 shows a typical scenario.

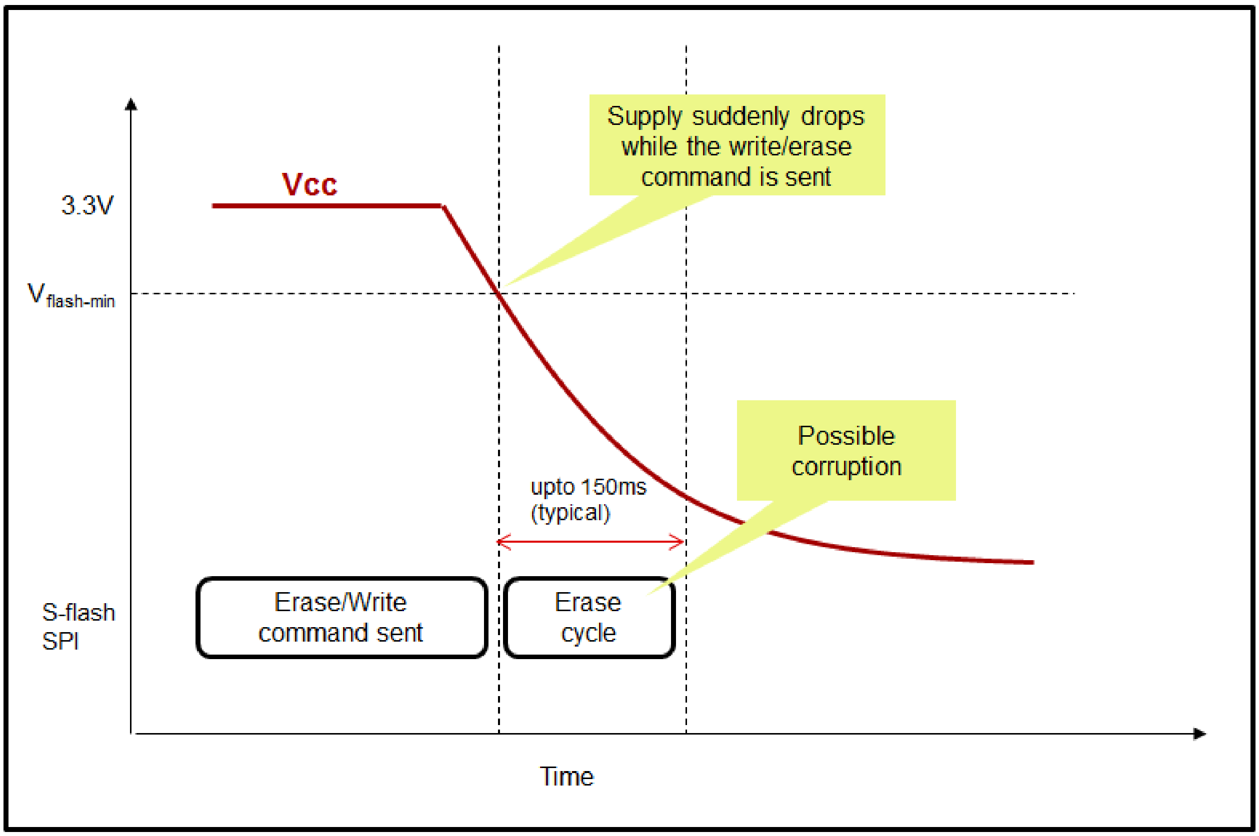

Figure 1. Sudden Power Off During Flash Erase

This scenario may occur in battery or line-powered end equipment:

- Battery-powered products: Removal of battery from the product without a soft shutdown
- Line-powered products: Electrical supply failure or sudden unplugging without a soft shutdown

The following sections explain how the potential for serial flash corruption can be minimized in both types of end equipment.

1.4.3.1 Battery-Powered Systems

In a battery-powered system, a sudden removal of power could coincide with an ongoing serial flash access. The chance of this occurring can be minimized by battery removal difficult while the product is in use, such as in a Wi-Fi weighing scale, where the user stands on the scale when the product is in use. Alternatively, product instructions could discourage users from removing the batteries while the product is in use. In systems desiring a higher degree of protection, a soft power-down push-button that maps to a GPIO of the system processor could also be provided to give a warning in advance that the user wants to remove the battery. In this system, the processor could use an LED to indicate that soft shutdown has completed, and that the batteries can now be removed.

1.4.3.2 Line-Powered Systems

In an AC main line-powered system, the failure of the grid can cause the supply of the Wi-Fi subsystem to suddenly drop. In the unlikely event that a grid failure and corresponding sudden drop in the Wi-Fi subsystem would coincide with the erase operation of the serial flash, data corruption may occur. One of the ways to minimize the chance of a flash corruption is to ensure that the DC voltage ramps down slowly after the input power is removed. Ensure that the DC voltage ramps down slowly with the help of bulk capacitors, which hold the charge while the input supply is removed. The value of the capacitor must be estimated from the maximum time for the SFLASH erase operation, voltage thresholds, and the current drawn by the system. The embedded system senses the sudden fall in the input voltage and initiates a soft shutdown of the Wi-Fi subsystem, thus safely completing all serial flash operations before $V_{flash\text{-}min}$ is reached. The charge stored on the capacitor would be used during this brief interval. Figure 2 shows this sequence.

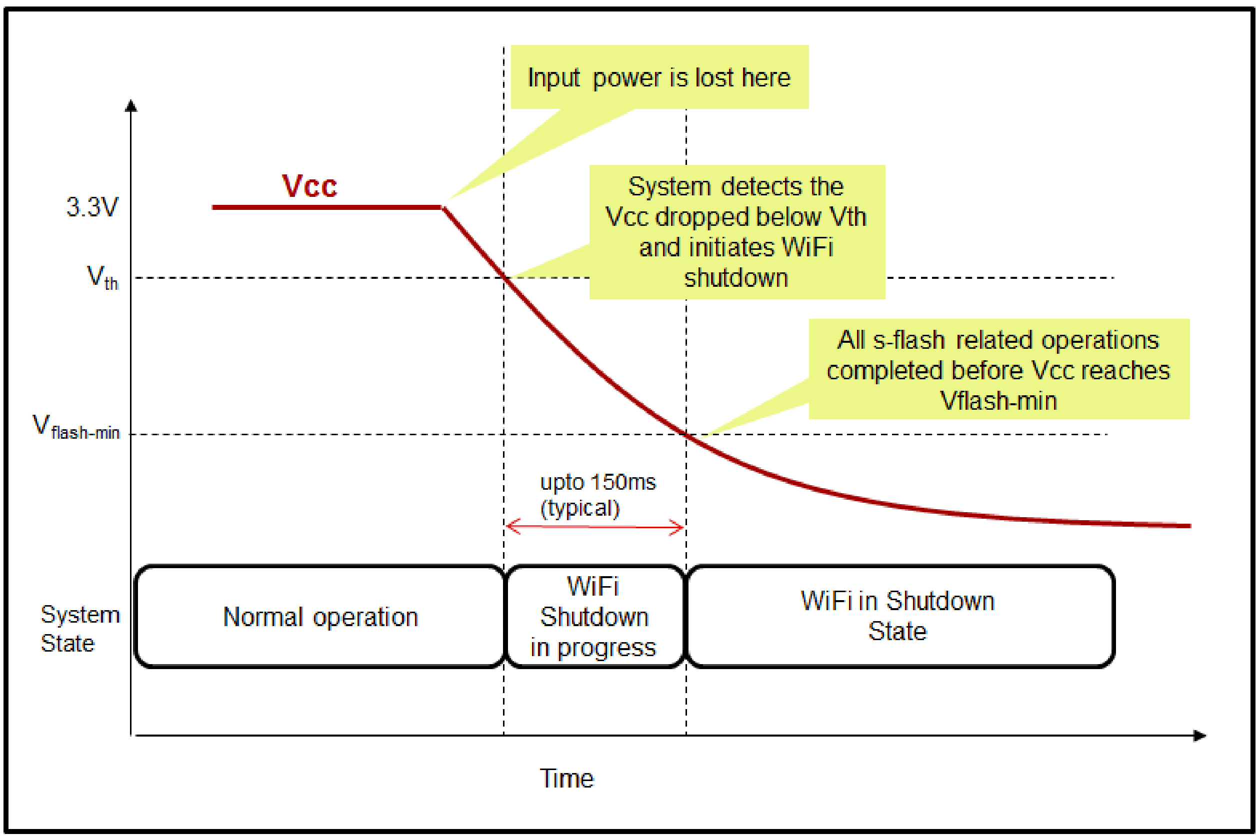

Figure 2. Sudden Power Off in Line-Powered Device

1.4.4 Brownout Mitigation Techniques for Self-Hosted CC3220 Designs

1.4.4.1 Overview

This section describes how to mitigate the effects of a substantially discharged battery on a self-hosted CC3220 system. It is assumed that the device is the main controller and is able to control all high power components in the system.

1.4.4.2 Problem Description

The brownout problem can occur when operating with a substantially discharged battery that has enough energy to power up the CC3220 processor, but not enough energy to power the Wi-Fi transmitter. This situation can cause a loop where the CC3220 device powers up and reaches the point where it does some high-power activity. This activity causes the battery voltage to drop below the brownout threshold, causing a reset. Once reset, the device consumes no power, the voltage rises back above the brownout threshold, and the device powers up again.

1.4.4.3 Suggested Solution

As a generic approach, CC3220 initialization does not turn on the internal networking processor function, so the CC3220 application can start running before Wi-Fi activity.

In this scenario, the application code can use the CC3200 A2D converter to monitor battery voltage, but the A2D accuracy must be factored into any decision about whether or not to activate the NWP for normal operation. To avoid the device getting reset repeatedly, use a secondary bootloader to load the user application and keep track of whether the application was loaded successfully (without causing another brownout event). Tracking the successful application load is done by keeping a counter in an on-chip register (OCR), which will likely be retained if voltage drops due to excessive power usage. Figure 3 describes the power-up flow.

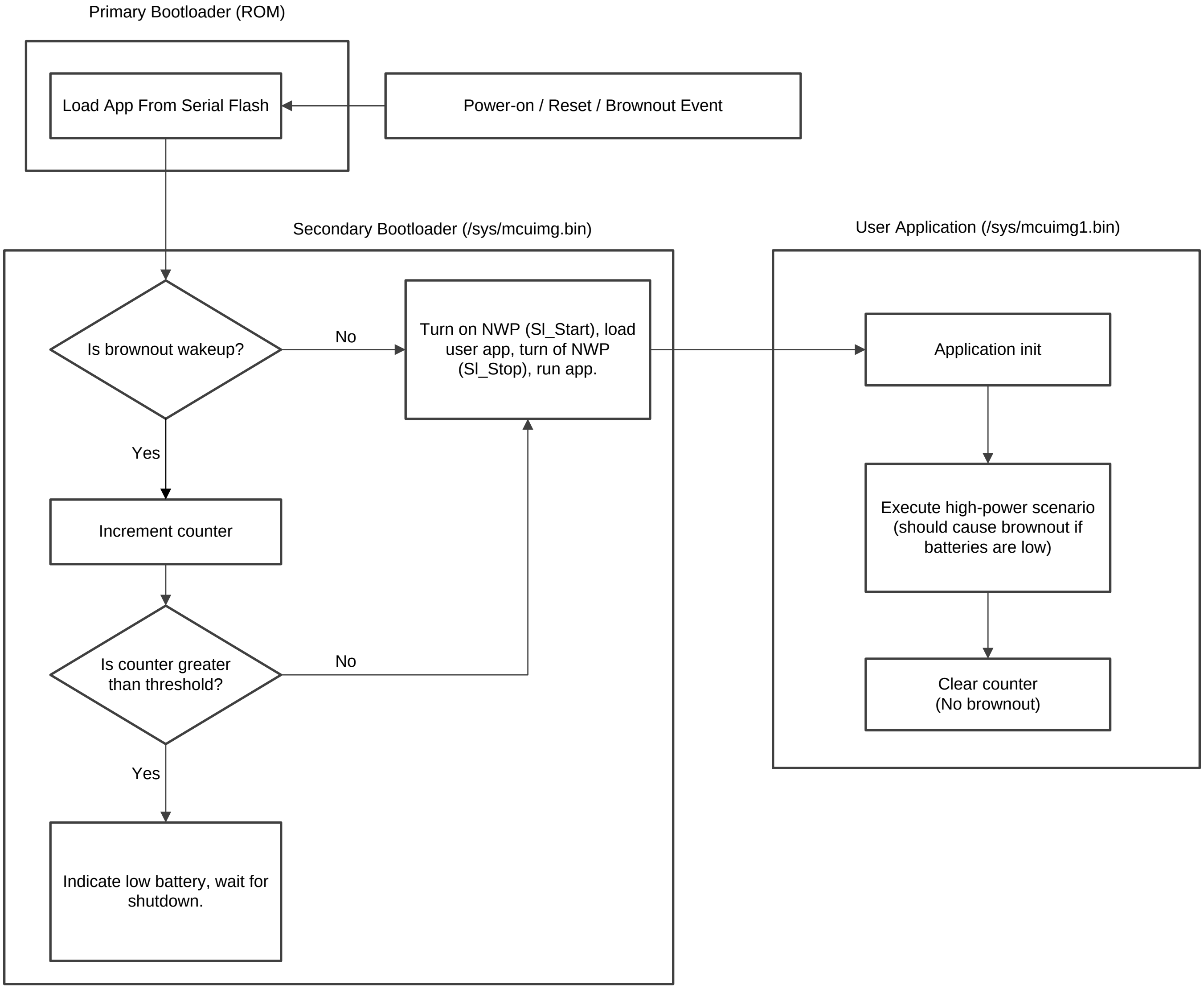

Figure 3. Brownout Suggested Flow

2 Factors to Consider When Using CC3x20 File System

2.1 Overview

The second part of this guide discusses the CC3x20 file system implemented on the serial flash. Unlike the serial flash, which can be chosen by system designers, the CC3x20 file system is common for all devices. System and configurations files can only be created and maintained using the CC3x20 file system. More detailed information on the CC3x20 file system can be found in the *CC3x20 SimpleLink Network Processor Programmer's Guide* (see the *File System* chapter).

2.2 File System Guidelines

Most of file system guidelines can be found in *CC3x20 SimpleLink Network Processor Programmer's Guide* (see the *Design Consideration* section in the *File System* chapter). In addition to those guidelines, system designers should also consider the following:

- The file system allocation table consumes five blocks (20KB).
- The serial flash storage type that the CC3x20 device supports has a minimal block size of 4096 bytes.
- Each file consumes at least:
  - One block (4KB) for a file with no fail-safe support
  - Two blocks (8KB) for a file with fail-safe support
- File size cannot be enlarged after the file has been created. To increase the file content during the device life cycle, the maximum size attribute should be set when the file is created (the file system reserves space).
- File attributes cannot be modified after the file has been created (apart from commit and rollback attributes).
- The file system does not handle fragmentation.

2.3 File Memory Space Mathematics

Because users may use the file system to store their own files, it is important to be able to accurately calculate the occupied memory space per file. The total occupied size on the flash is a function of the file content length (or the maximum size attribute upon creation), file attributes, and file system metadata.

Figure 4 shows the process for calculating how much memory is consumed on the serial flash.

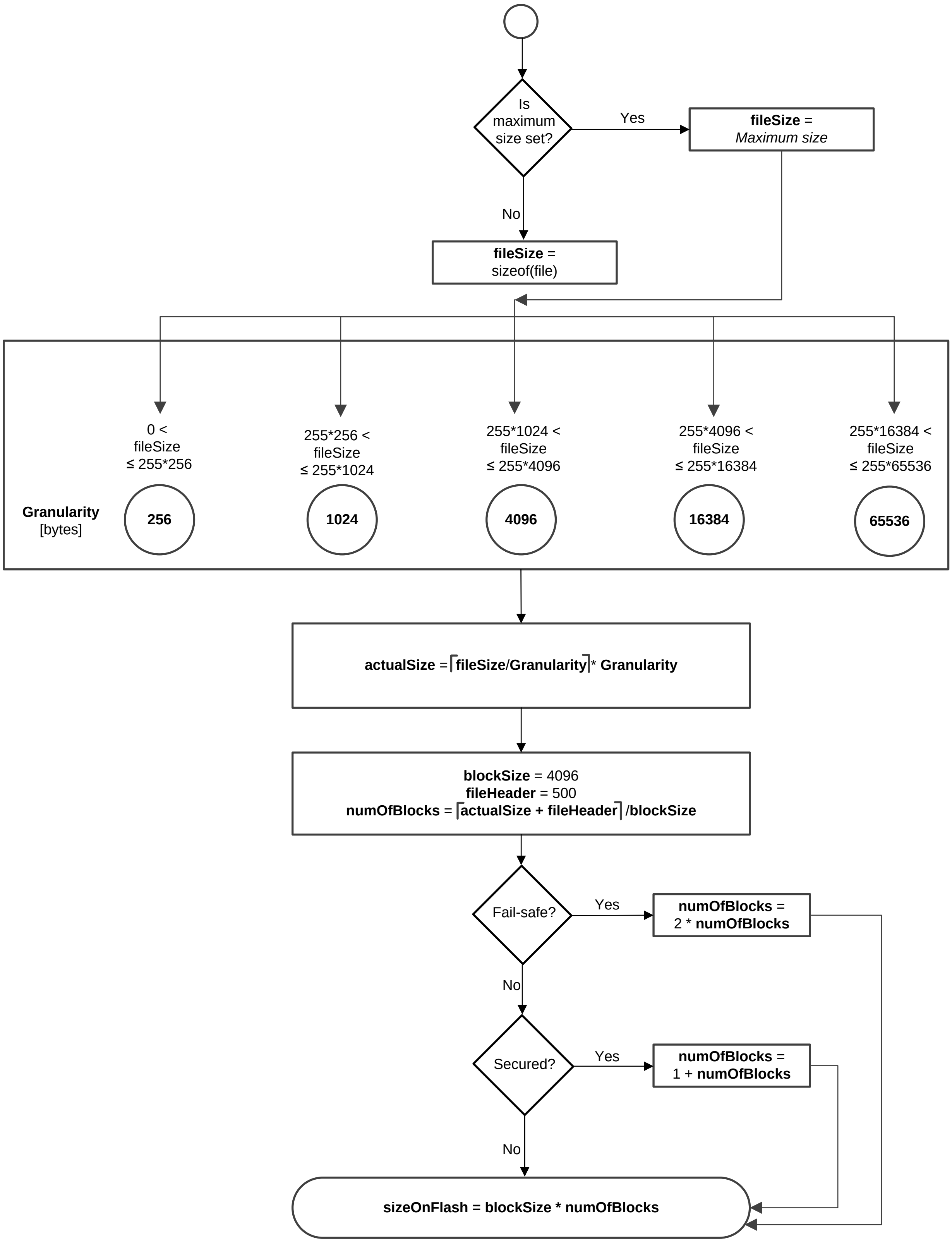

Figure 4. File Memory Consumption on Flash

2.4 System Files

2.4.1 Overview

System files include either a subset or all configuration files that set the device into a predefined state. System files can be implicitly created by the user (such as through invoking a host driver API), or internally created by the device. Because these files are essential for proper operation of the device, system designers must understand when these files are created and how to monitor the serial flash occupancy.

Failing to preserve enough space for the system files may cause unexpected behavior in the system.

2.4.2 Host Driver Mapping

Most of the system files are created implicitly by users when invoking a host driver API. Some system files are internally created by the network processor subsystem, and the user does not have control over these files (for example, calibration files, ARP table files, and so forth).

For detailed information on the host driver API implicitly attached with the system files, see the *CC3x20 SimpleLink Network Processor Programmer's Guide* (see the *Persistency* appendix).

2.4.3 How to Profile Serial Flash Content

Designing and monitoring the content on serial flash is mandatory, especially in embedded devices where memory resources are limited. New generation SimpleLink devices provide some options for system designers to both help design the system and monitor it.

2.4.3.1 Using UniFlash

2.4.3.1.1 Monitor the Storage Breakdown (Available Only During Operational Mode)

For this option to work, the device must be opened in development mode. All that is required is to connect to the device and click the file listing button.

Figure 5 is taken from the Out-of-Box project and shows the file listing (the file listing button is marked with a blue circle).

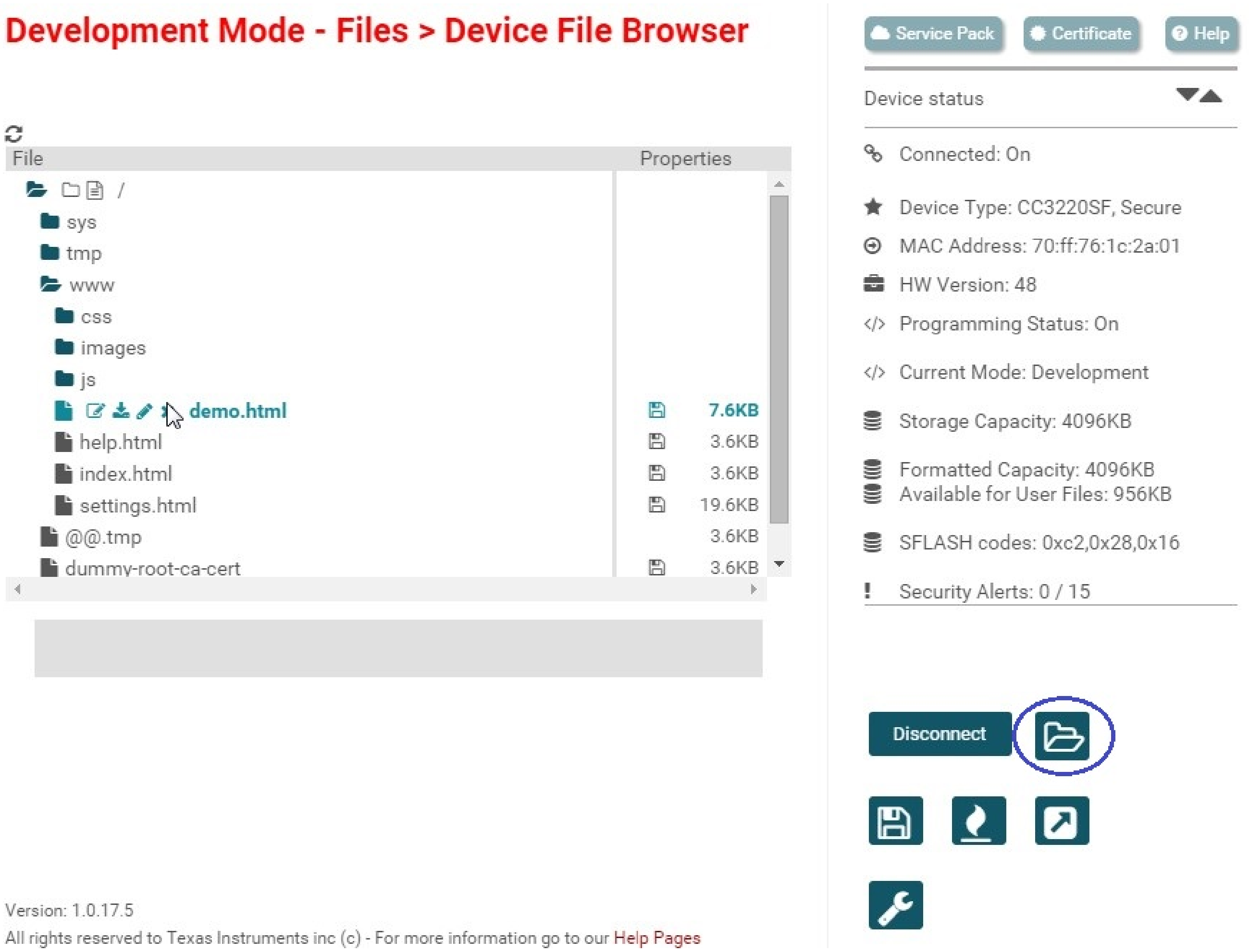

Figure 5. UniFlash File Listing

2.4.3.2 Using Host Driver APIs

Storage content can be monitored in real time using a host driver API.

There are two APIs for this purpose:

- The file list interface displays information regarding the existing files and the number of allocated blocks per file.
- The Get storage information command contains information about the device usage; it contains information regarding the device memory usage, information about security alerts in the system, the number of times FAT has been accessed, and so forth.

Figure 6 shows the file listing feature for the out-of-box invoked through the file system API and printed on the terminal. Listing the files can be invoked by executing the sl_FsGetFileList() API.

```
user@CC3220: fslist -lh
ca.der                                        3K         0x020
/sys/mcubootinfo.bin                          3K         0x2d0
/www/demo.html                                7K         0x000
/www/help.html                                3K         0x000
/www/index.html                               3K         0x000
/sys/ipcfg.ini                                3K         0x040
/www/ota.html                                11K         0x000
/sys/stacfg.ini                               3K         0x040
/sys/pref.net                                 7K         0x050
/sys/ap.cfg                                   3K         0x040
/sys/mdns.cfg                                 3K         0x040
/sys/dhcpsrv.cfg                              3K         0x040
/sys/mode.cfg                                 3K         0x040
/sys/devname.cfg                              3K         0x040
/tmp/phy.cal                                 11K         0x040
/www/settings.html                           19K         0x000
/sys/certstore.lst                            7K         0x2d0
/sys/servicepack.ucf                        131K         0x2d0
/sys/mcuflashimg.bin                        179K         0x2d0
/sys/mcuflashimghash.bin                      3K         0x070
/sys/factory.img                           1347K         0x070
/sys/ucf_signatures.bin                       3K         0x060
/www/info.html                               15K         0x000
/www/css/style.css                           31K         0x000
/www/images/rotate360.jpg                  1011K         0x020
/www/images/tilogo.gif                        7K         0x000
/www/images/icons/help.png                    3K         0x000
/www/images/icons/menu.png                    3K         0x000
/www/images/icons/wireless.png                3K         0x000
/www/images/icons/wirelessfull.               3K         0x000
/www/js/jquery.min.js                        83K         0x000
/www/js/scripts.js                            3K         0x000
/tmp/crashminidump.bin                       27K         0x020

 File properties flags description:
 0x001 - Open file commit
 0x002 - Open bundle commit
 0x004 - Pending file commit
 0x008 - Pending bundle commit
 0x010 - Secure file
 0x020 - No file safe
 0x040 - System file
 0x080 - System with user access
 0x100 - No valid copy
 0x200 - Public write
 0x400 - Public read
```

Figure 6. File Listing Through Host API

Figure 7 shows the storage info feature for the out-of-box invoked through the file system API and printed on the terminal. Storage information can be invoked by executing sl_FsCtl() API with the SL_FS_CTL_GET_STORAGE_INFO command.

```
user@CC3220: fsdu
Total space: 4096K

Filestsyem      Size    Used    Avail
User            4044K   3588K   456K
System          0K
Reserved        52K


Max number of files             : 240
Max number of system files      : 51
Number of user files            : 17
Number of system files          : 16
Number of alert                 : 0
Number Alert threshold          : 3
FAT write counter               : 38
Bundle state                    : Stopped
```

Figure 7. Get Storage Information Through Host API

2.4.4 Flash Recommended Size

The recommended size of the serial flash is determined by the memory space occupied by all system files, and by the programming image if restore-to-factory is enabled.

System designers can either reserve the entire space, or customize and reserve only the required space. In the latter case, take extra caution because not preserving enough space may cause unexpected behavior in the system.

Table 3 lists the minimal required memory consumption under the following assumptions:

- System files in use consume 64 blocks (256KB).
- Vendor files are not considered.
- MCU code is taken as the maximal possible size for the CC3220 with fail-safe enabled to account for future updates, such as through OTA.
- Gang image:
  - Storage for the gang image is rounded up to 32 blocks (meaning 128KB resolution).
  - Gang image size depends on the actual content size of all components. Service pack, system files, and the 32-block resolution are assumed to occupy 256KB.
- All calculations consider that the restore-to-default is enabled.

Table 3. Recommended Flash Size

Element	CC3120 [KB]	CC3220 [KB]	CC3220SF [KB]
File system allocation table	20	20	20
System and configuration files	256	256	256
Service pack	264	264	264
MCU code	N/A	512 (1)	2048 (1)
Gang image size	256	256 + MCU	256 + MCU
Total	796	1308 + MCU	2844 + MCU
Minimal flash size (2)	8Mb	16Mb	32Mb
Recommended flash size (2)	16Mb	16Mb	32Mb

(1) Including fail-safe.
(2) For maximum MCU size.

Revision History

NOTE: Page numbers for previous revisions may differ from page numbers in the current version.

Changes from February 8, 2017 to February 27, 2017	Page
• Added *Listing the files can be invoked by executing sl_FsGetFileList() API.*	12
• Added *Storage information can be invoked by executing sl_FsCtl() API with the SL_FS_CTL_GET_STORAGE_INFO command.*	13