NXP Semiconductors
Application Note

Document Number: AN12003
Rev. 0, July 2017

Using S32K148 FlexNVM Memory

by: NXP Semiconductors

Contents

1 Introduction

This application note describes how the FlexNVM memory can be used in the S32K148 device. All S32K1xx devices have a FlexNVM section. However, it has a different implementation in the S32K148 device. Therefore, specific considerations must be taken. More details about the FlexNVM are described in the Reference Manual. This application note focuses on the different usages that this memory can have in the S32K148, considerations that must be taken with each configuration and software recommendations.

The application note is supported by software considerations, handling common errors, and practical code examples.

2 Overview

2.1 Glossary

This section provides a brief definition of terms that are used through all the document to help the reader have a better understanding of the document.

Table 1. Glossary

Term	Description
P-Flash	Program flash memory, One or multiple NVM read partitions that can execute program code.
FlexNVM	Nonvolatile flash memory that can execute program code, store data or back up emulated EEPROM data. This memory is located in a separate partition of the P-Flash.
D-Flash	Data Flash, nonvolatile flash memory for user data, boot code, and additional code store. DFlash can be partitioned from the FlexNVM block to store data or execute code.
E-Flash	Nonvolatile flash memory as part of the FlexNVM that is used by the Flash controller FTFC to emulate EEPROM (EEPROM backup).

Table continues on the next page...

Table 1. Glossary (continued)

Term	Description
EEE	The Flash module (FTFC) emulates the characteristics of an EEPROM by effectively providing a RAM buffer (FlexRAM) as virtual EEPROM as an Interface to the user/ EEPROM driver. The data is automatically programmed into the FlexNVM block using a hardware built-in filing system.
FlexRAM	RAM that can be used as SRAM or as high-endurance emulated EEPROM storage.
EERAM	RAM as part of FlexRAM used for EEPROM emulation.
Bank	read partition
Sector	Smallest erasable Flash area.
Section	definable Flash size for Program/Read Section Command.
Interleaved block	On S32K two 64-bit wide Flash blocks can be connected to a 128-bit wide interleaved memory block.

2.2 FlexNVM description

All S32K1xx family devices have a section called FlexNVM. This section can be used as Pflash (program flash), Dflash (data flash), and emulation EEPROM backup (E-Flash). In most of the family devices this section is 64 KB length. However, The S32K148 device has a FlexNVM of 512 KB length, which added to the 1.5 MB of Pflash generate a total of 2 MB. Special considerations must be given to FlexNVM. It can be read from, programmed, erased, Emulated EEPROM update, or used for CSEc cryptographic operation but only one of these at one time. Next section shows different use cases for FlexNVM.

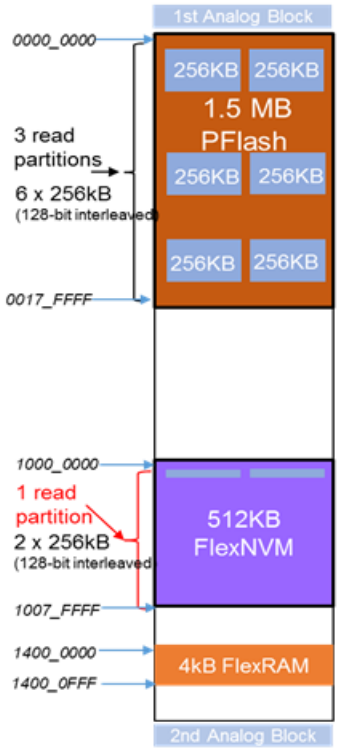

Figure 1. S32K148 Memory overview

NOTE

The **512 KB** FlexNVM partition is not contiguous with the **1.5 MB** PFlash.This involves special considerations in linker file and compiling properties, for example the FAR attribute.

3 Use cases

FlexNVM section can be used in three different configurations:

- No EEPROM nor CSEc enabled.
- EEPROM enabled and CSEc disabled.
- EEPROM and CSEc enabled.

The main characteristics of each configurations along with how the FlexNVM section is partitioned is described in sections below.

3.1 No EEPROM nor CSEc enabled

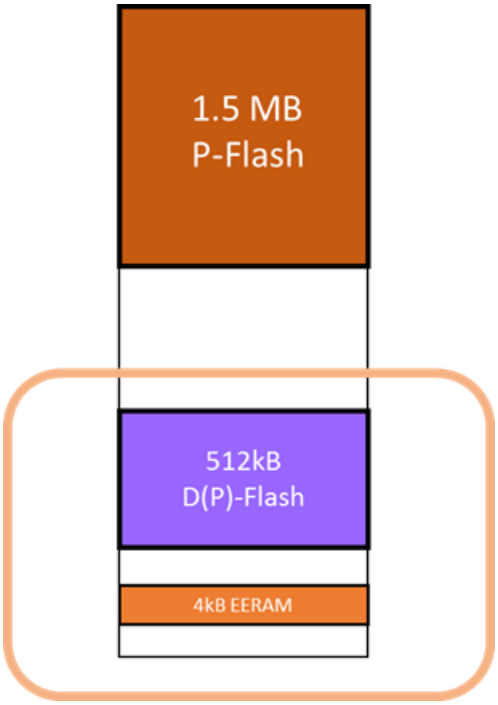

Figure 2. FlexNVM structure (No EEPROM nor CSEc)

The main characteristics of this configurations are:

- The 512 kB FlexNVM can be used as either PFlash or DFlash. In other words, The FlexNVM can be used for program or data storage**. If used as PFlash keep in mind that the FlexNVM section is not cacheable**. Therefore, some performance degradation is expected.
- The 4 kB FlexRAM can be used as SRAM, but it does not have ECC as the main SRAM does and it operates at Flash clock's speed.
- As EEPROM mechanism is not enabled. Endurance of the FlexNVM will follow the same characteristics of the PFlash memory specified in datasheet.

When to use this configuration:

- This configuration is especially useful when data flash requires **infrequently** updates of data and more than 4 KB (Maximum size available for EEPROM) are needed.
- If more than 1.5 MB of PFlash are needed, this section can be used as either PFlash and/or DFlash divided as the application requires. In case more than 2 MB of PFlash are required it is possible to use an external memory via the QuadSPI interface.

3.2 EEPROM enabled

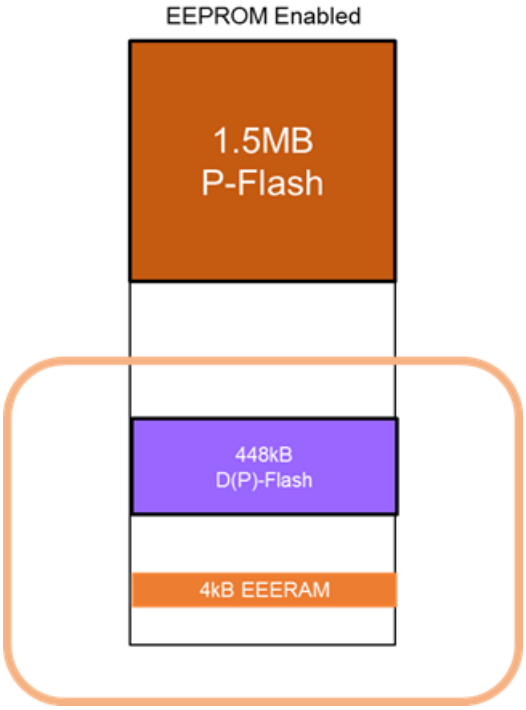

Figure 3. FlexNVM structure (EEERPOM enabled)

The main characteristics of this configurations are:

- In this configuration **64 KB** of the FlexNVM are used as **E-Flash (**EEPROMbackup**)** and **448 KB** remain as PFlash or DFlash.
- The E-Flash is not memory mapped. In other words, this memory cannot be accessed by any means after enabling EEPROM.
- **4 KB EERAM** with 32-bit access are used for **EEPROM** emulation.

When to use this configuration:

- This configuration is especially useful when a data partition with high endurance (**up to 100K cycles**) is required.
- Best for frequent update of records. For example:
  - Key on/off updates

3.3 EEPROM and CSEc enabled

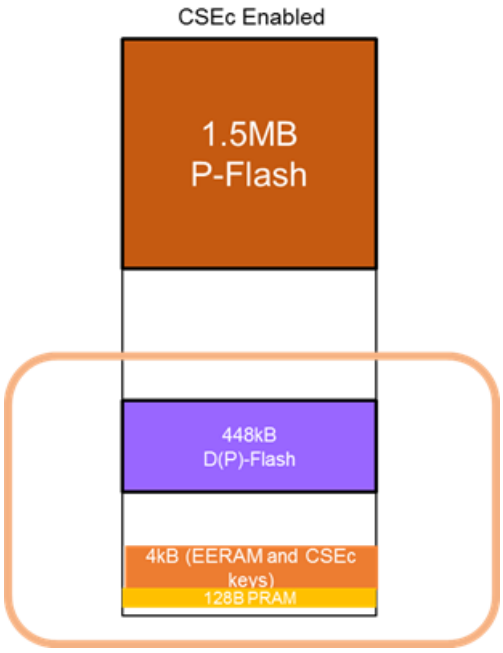

Figure 4. FlexNVM structure (CSEC enabled)

The main characteristics of this configurations are:

- In this configuration **64 KB** of the FlexNVM are used as **E-Flash (EEPROM backup and key storage)** and **448 KB** remain as PFlash or DFlash.
- The E-Flash is not mapped memory. In other words, this memory cannot be accessed by any means after enabling EEPROM.
- To use CSEc operations it is **required** that EEPROM is **enabled** as E-Flash is also used for saving CSEc data.
  - E.g. Keys
- CSEc uses up to **512 Bytes** for key storage, leaving EEPROM to use the remaining **3.5KB** EERAM

When to use this configuration:

- This configuration is necessary when CSEc operations are executed.

4 Restrictions

The following figure shows the restrictions for simultaneous operations in the different flash sectors:

		Pflash			FlexNVM			FlexRAM			CSEc
		Read	Program Phrase	Erase Flash Sector	Read	Program Phrase	Erase Flash Sector	Read	EEPROM Write	RAM write	Any command
Program Flash	Read					Allowed	Allowed		Allowed		Allowed
	Program Phase				Allowed			Allowed		Allowed	Allowed
	Erase Flash Sector				Allowed			Allowed		Allowed	Allowed
FlexNVM	Read		Allowed	Allowed							
	Program Phase	Allowed						Allowed		Allowed	
	Erase Flash Sector	Allowed						Allowed		Allowed	
FlexRAM	Read		Allowed	Allowed		Allowed	Allowed				
	EEPROM Write	Allowed									
	RAM Write		Allowed	Allowed		Allowed	Allowed				
CSEc	Any command	Allowed	Allowed	Allowed							

Figure 5. Flash sectors simoultaneous operations

From Figure 5, it is very important to notice that **only one** FlexNVM operation can be executed on the 512 KB FlexNVM partition. These operations are:

- Read
- Program
- Erase
- EEPROM write
- CSEc.

In consequence, there are software considerations that must be taken. Those considerations are discussed in the following section.

5 SW recommendations

User must be sure to follow these recommendations in order to avoid problems with the FlexNVM. Any access problem in the FlexNVM will trigger a collision and the operation that was being done will be invalidated. Collision handling will be explained in more detail later in this section.

The recommendations that the user must follow are:

- Any software driver that uses CSEc, EEPROM (writes only) or Flash controller commands must not be placed in FlexNVM's PFlash. For example:
  - Encryption/decryption CSEc commands.
  - Storing calibration information.
- Any Configuration Data (constant parameters) that must be read during a CSEc or EEPROM write or program/erase operation must not be placed in FlexNVM's Dflash.
- Any ISR associated to an interrupt that has to be served during CSEc or EEPROM write or program/erase operation must not be placed in FlexNVM's Pflash. The same restriction applies to the functions called from ISRs.

Most of the previous constraints can be eliminated if:

- Interrupts are disabled during FlexNVM Operations.
- The SW APIs that use the FlexNVM are working synchronous (return after the requested operation finished).

Another recommendation is to poll for the CCIF flag before initiating any operation that involves CSEc, EEPROM (writes only) or Flash controller commands. The following routine is an example of how can the CCIF should be polled:

```
/* The following routines writes data into EEEPROM */
         voidEEEPROM_write( uint32_t  data){
         uint32_t  * FlexRAMptr = ( uint32_t  *) 0x14000000U;  /* Pointer to FlexRAM
memory section */
        while ((FTFC-> FSTAT  & FTFC_FSTAT_CCIF_MASK) == 0){} /* Wait for anyFlexNVM
operation to be finished */
        *FlexRAMptr = 0x00AA00BB;  /* Writes into EEEPROM */
}
```

Previous routine writes into EEPROM memory. As can be observed, before writing into the section it polls for the CCIF flag to be sure that there was no previous operation running. If one operation is running it would wait until the operation has finished (CCIF flag set to 1).

In case user skip one of these recommendations and cause a simultaneous access, the event will cause a **collision** (reported and interruptible through the FTFC Read Collision Error Flag **RDCOLERR**). This collision will **always** trigger a hard fault error. If the user enables the interruption for the Read Collision Error Flag, the interruption will be triggered after the hard fault has been handled.

The following routines are examples of how these errors must be handled:

```
voidHardFault_Handler(void){
        /* Handling of error must be
        placed here.         * an error condition flag can
        be set         * to trigger some recovery
        routine */
        }
voidRead_Collision_IRQHandler(void){
          FTFC-> FSTAT  |= FTFC_FSTAT_RDCOLERR_MASK;/* clear interruptflag */
          while ((FTFC-> FSTAT  & FTFC_FSTAT_CCIF_MASK) == 0){}  /* wait for collision
condition to be done */
         /* any other routine that hanndles  the issue */
          }
```

It is important to remember that if a collision occurs. The program will always jump to the hard fault routine **first**, then, if the Read Collision Error Flag interruption is enabled, program will attend to the Read Collision ISR **after the hard fault routine**. These are only examples and user can use any recovery/correction mechanism that they developed for their application. However, keep in mind that having collision is not recommended as it will invalidate the previous operation that was being done in the FlexNVM.

6 Applications examples

This section provides some examples of software blocks or drivers that can be stored or executed from FlexNVM's PFlash section. However, this will always be dependent of user's application. User can use this section to satisfy any need of their application as long as it does not violate conditions that were described above.

- Every peripheral configuration/initialization code that does not require EEPROM (writing), CSEc nor Flash commands
- Bootloader code (assuming flash commands are launched from RAM).
- General application level code.
- Middleware/stacks.
- Post processing algorithms.
- Core self-test.
- Constant data (if copied to RAM at startup or not used during FLEXNVM operations).

7 References

- NXP semiconductors. S32K1xx Data Sheet. 2017.
- NXP semiconductors. S32K14x Series Reference Manual. 2017.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Document Number: AN12003
Rev. 0, July 2017