NXP Semiconductors
Application Note

Document Number: AN11983
Rev. 1, March 2018

Using the S32K1xx EEPROM Functionality

by: NXP Semiconductors

1 Introduction

There are three methods to implement EEPROM functionality. One is using real EEPROM such as KEA64 which has 256 B EEPROM. The advantage is it doesn't occupy the flash memory. But the disadvantage is, it is expensive. The second method is using software to realize the EEPROM functionality such as KEA8 and KEA128. The advantage is cheap. But the disadvantage is it occupies the flash memory including the extra code for EEPROM realization and the flash memory using to simulate EEPROM. The third method is using firmware to realize the EEPROM functionality such as S32K1xx. The advantage is the realization of EEPROM functionality is absolutely transparent to customer and does not need flash memory to store extra code for EEPROM implementation.

This application note mainly introduces S32K1xx EEPROM (EEE) functionality feature and usage. The S32K1xx EEPROM (EEE) allows users to configure some of the on-chip flash memory as enhanced EEPROM, additional flash memory, or a combination of the two.

For detailed information for any reference mentioned during this application note, please refer to S32K1xx Reference Manual and Datasheet.

2 S32K1xx EEPROM (EEE) features

The S32K1xx EEPROM (EEE) has a number of features that allow the replacement of external EEPROMs and improves upon their performance.

S32K1xx EEE features include:

- Automatic – does not require customer software development.
  - Reset: EEE image loaded into RAM.
  - Reads: Records read directly from RAM. No EEE operation.
  - Writes: Records written directly to RAM. Image automatically synchronized with EEE flash.
- Utilizes best practices to optimize reliability and cycling endurance.
  - Round robin load leveling.
  - Sector retirement.
- Brownout tolerant.

Contents

1 Introduction.....1

2 S32K1xx EEPROM (EEE) features.....1
2.1 S32K1xx EEE works.....2
2.2 How S32K1xx EEE uses memory.....2

3 Using the S32K1xx EEE.....5
3.1 S32K1xx EEE partitioning.....5
3.2 S32K1xx FlexRAM configuration.....10
3.3 S32K1xx command error handling.....14
3.4 S32K1xx EEE startup.....14
3.5 S32K1xx reading and writing the EEE.....14

4 S32K1xx EEE performance.....15

5 S32K1xx brownout detection.....16

6 S32K1xx new quick write mode.....16

7 S32K1xx EEPROM endurance.....17

8 Software considerations.....18
8.1 Simultaneous operations.....18
8.2 Enabling CSEc and EEPROM.....18
8.3 Power-On recommendation...18
8.4 Data record checking.....19
8.5 Power Modes transition.....19

9 Appendix A EEPROM examples.....19

10 Revision history.....22

  - New Quick Write mode priority-writes small amount of data before power loss.
- Already proven in production.
  - More than 150M C90TFS industrial and consumer units in the field.

2.1 S32K1xx EEE works

To provide enhanced EEPROM functionality, the S32K1xx EEE uses a RAM block (FlexRAM), a flash block (FlexNVM), and EEE state machine. When the EEE functionality is enabled, the FlexRAM becomes your EEE memory. The FlexRAM address space is where you access all of your EEE data. When the EEE is accessed, the EEE state machine keeps track of the data and backs it up as data records, stored in some portion of the FlexNVM used as an E-flash. Using a large block of E-flash to back up the data for a smaller amount of EEE data allows the S32K1xx EEE implementation to offer extremely high endurance. The EEE state machine uses 72-bit records to backup data from the EEE into the flash (E-Flash). Thirty two bits of the record are used for the data, and the other 40 bits are address, status and parity information about the data. The data records are written and erased as needed. This means that if a location within the EEPROM has never been accessed, there will be no data record for it. This helps to reduce the amount of data that needs to be backed up and can increase the memory endurance.

2.2 How S32K1xx EEE uses memory

S32K1xx have two separate blocks of flash, the P-Flash block and the FlexNVM block. The P-Flash block is intended to be used as the program flash block, but it can also be used for storing both instructions and data. The FlexNVM block is a configurable flash block that can be used as additional flash space (D-flash), as backup memory for supporting the enhanced EEPROM functionality (E-flash), or as a combination of the two.

NOTE

The portions of the FlexNVM not used as EEE backup memory (E-flash) are referred to as D-flash. This flash would typically be used for data storage space; however, like the P-flash, D-flash can actually be used for instructions or data.

S32K1xx consists of the FlexNVM block, FlexRAM, and EEE state machine. These three blocks together are required to support the EEE functionality. Figure 1. on page 3 shows the S32K1xx Memory Map.

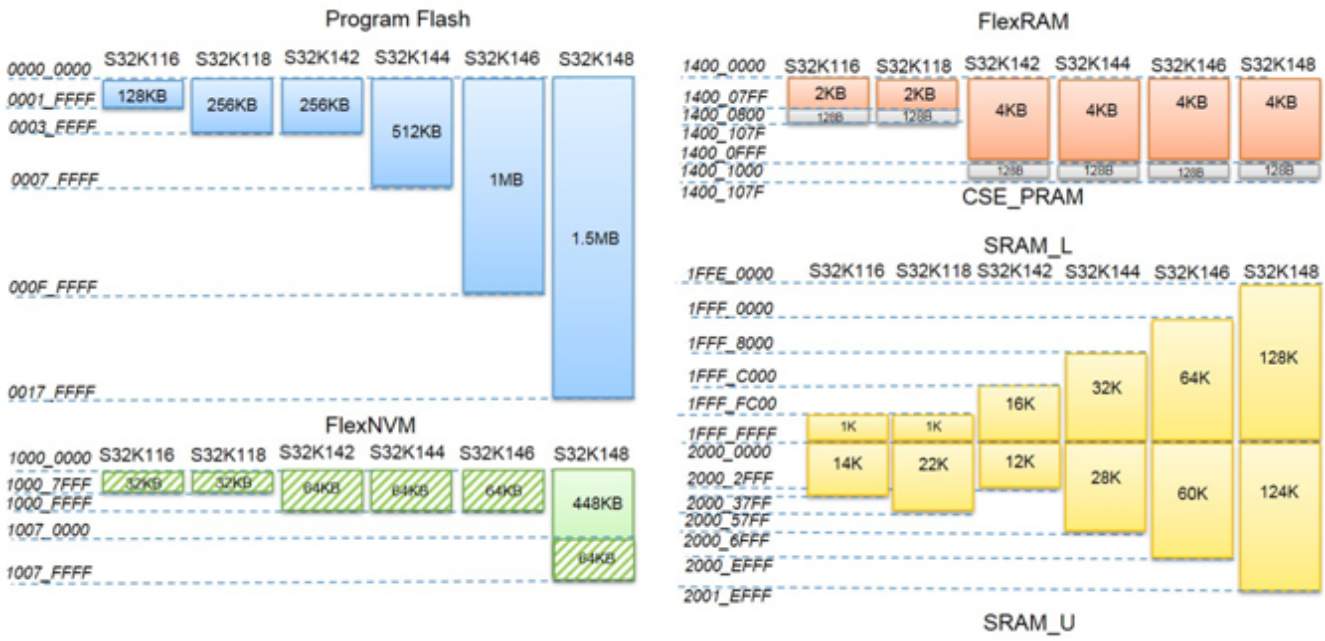


Figure 1. S32K1xx Memory Map

Special considerations need to be taken for S32K148, in which, FlexNVM block is shared with 448 kB of P/DFlash. Software considerations describes detailed information for these considerations.

The sub sections below uses S32K144's memory map as reference for different EEE configurations. Same principle applies to different S32K1xx devices considering their own memory sizes/addresses.

2.2.1 S32K1xx with EEE functionality disabled

The figure below shows how the memory blocks function about S32K144, if the EEE functionality is disabled.

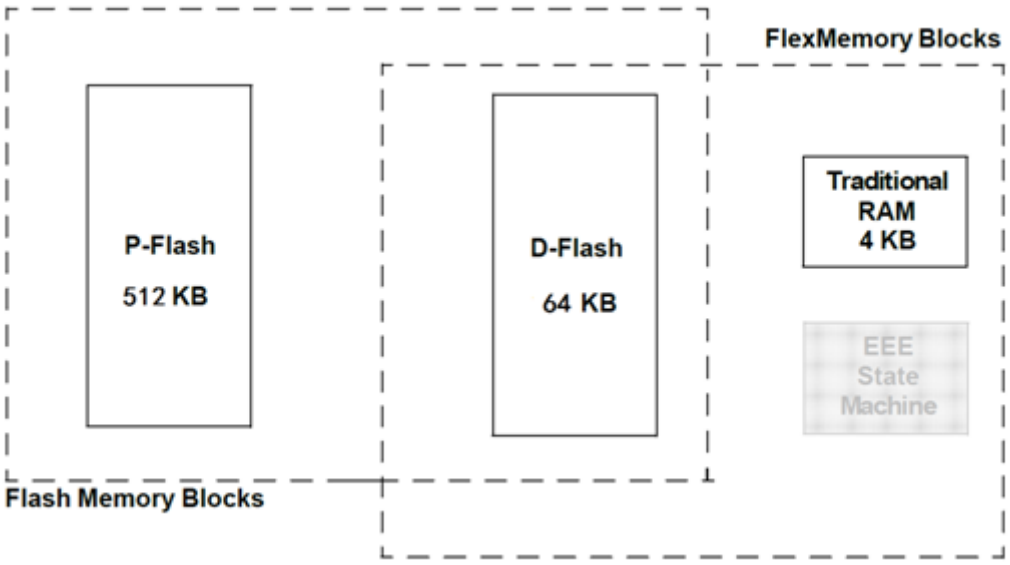


Figure 2. S32K144 Memory Block (Flash and NVM) with EEE disabled

The P-Flash memory is always a P-Flash. Its functionality does not change for any FlexMemory configuration. Because the EEE functionality is not used in this case, the entire FlexNVM is allocated as D-flash space (no E-flash is needed). The FlexRAM becomes a 4 kB traditional RAM. This means it can be used as extra memory space, but keep in mind that it does run at the flash clock speed and not the core speed (the tightly coupled memory RAM, TCM, runs at the core speed). The EEE state machine is present in the device, but not active.

NOTE

No ECC is generated for FlexRAM used as traditional RAM.

2.2.2 S32K1xx with EEE functionality enabled

The figure below shows how the memory blocks function, if the EEE functionality is enabled and the entire FlexNVM is used to back up the EEE data.

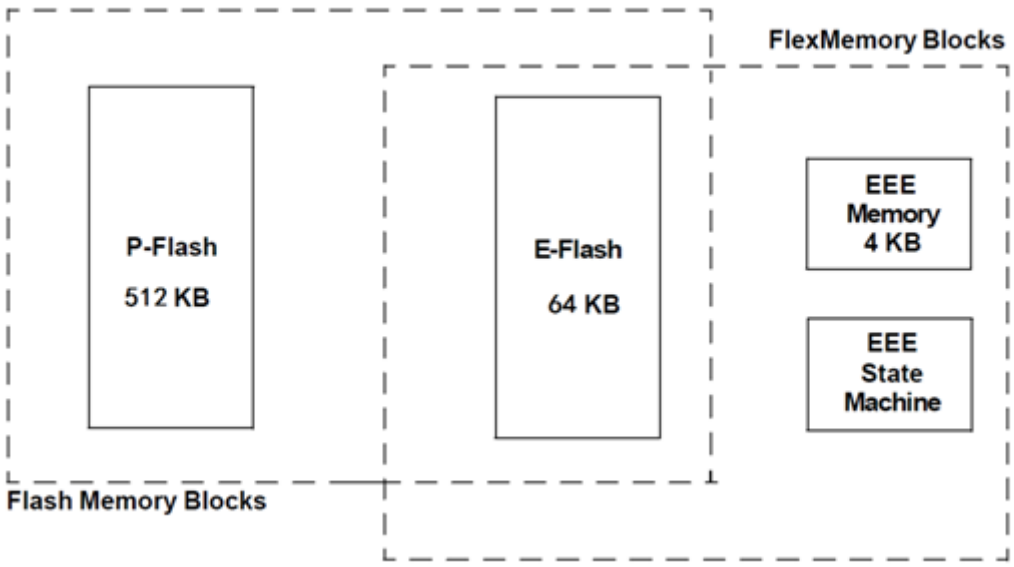


Figure 3. S32K1xx Flash EEE with EEE enabled (Entire FlexNVM used to backup EEE data)

There are a number of configuration options that can be used when the EEE functionality is enabled. Figure 3. on page 4 shows an example of where the entire FlexNVM is used as E-flash memory. The FlexRAM becomes the EEE memory space (up to 4 kB). Any reads and writes of the EEE data uses this 4 kB memory space because the E-flash is not directly accessible. The EEE state machine automatically manages all of the writes to the EEE memory space, and generates flash program and erase operations as needed into the E-flash.

2.2.2.1 E-flash and EEE memory configuration details

Figure 3 shows the memory blocks at a high level and shows the respective functionality of the configurable memory blocks. This section describes the actual use of the E-flash and FlexRAM blocks to create a EEE subsystem in more detail and provide examples of some different configurations. S32K1xx allows for many different memory configuration options. The amount of EEE data and the amount of E-flash memory used to backup that EEE data are all programmable. This allows you to make trade-offs between the memory size and endurance of the EEE.

There are two programmable options that are used to define the exact memory use for a system. These parameters are:

1. EEE size — This is the total size of the EEE data needed. The total EEE size can be 0 or 4 kB for S32K14x devices and 0 or 2 kB for S32K11x devices.
2. FlexNVM partition — This parameter defines how much of the FlexNVM is used as normal flash (D-flash) and how much is used as EEE backup memory (E-flash). If EEE is used, then at least 32 kB (For S32K14x devices) or 24 kB (For S32K11x

devices) of the FlexNVM must be allocated as E-flash. To get the maximum possible endurance from the EEE, then the entire FlexNVM can be used as E-flash.

FlexNVM endurance is impacted by the relationship between EEE backup memory and EEE size. See S32K1xx datasheet for detailed information about NVM reliability specifications.

3 Using the S32K1xx EEE

3.1 S32K1xx EEE partitioning

To use the EEE features, the memory has to be partitioned. The partitioning process tells the state machine how much EEE memory will be used and how much of the FlexNVM flash will be used to back up the EEE. The flash has a special program partition command that is used to configure the EEE. The program partition command is used to program the two EEE configuration parameters described in E-flash and EEE memory configuration details. These two parameters are programmed into a special location within the flash block itself. As this is a non-volatile storage location, the partitioning needs to be done only once in the entire lifetime of the device. Before launching the program partition command the FlexNVM and D-flash IFR must be in an erased state. It is recommended that a new device is partitioned as the first step in factory programming.

NOTE

Partitioning must only be done once. If the flash is re-partitioned for a different configuration, then recorded data is lost and you are not guaranteed to get the expected endurance.

NOTE

Partitioning information, EEE data, and EEE location information is lost if a device is mass erased. Use of a backdoor key is highly recommended when enabling security on a device where the EEE is being used. The backdoor key allows for a means of disabling security temporarily without needing to perform a mass erase. If a mass erase is used instead then the endurance of the EEE can no longer be guaranteed. CSEc enabled parts are required to do DBG_AUTH command prior to any mass erase.

3.1.1 S32K1xx program partition command

The program partition command prepares the FlexNVM block for use as data flash, emulated EEPROM backup, or a combination of both and initializes the FlexRAM. For detailed information see S32K1xx's reference manual. The table below shows parameters needed for partition command.

Table 1. Program partition command

FCCOB Number	FCCOB Contents [7:0]
0	0x80 (PGMPART)
1	CSEc Key Size
2	SFE
3	FlexRAM load during reset option (only bit 0 used): 0 - FlexRAM loaded with valid EEPROM data during reset sequence 1- FlexRAM not loaded during reset sequence
4	EEPROM Data Set Size Code
5	FlexNVM Partition Code

3.1.1.1 Program partition command requirements

Flash commands are configured through FCCOB registers in FTFC module. Each command includes its own parameters and for program partition command, six parameters are needed:

- FCCOB0 defines desired command. 0x80 refers to PGMPART command (Program partition command).
- FCCOB1 and FCCOB2 are used for CSEc configuration. These two parameters are explained in depth in AN5401 - Getting Started with CSEc Security module. For EEPROM functionality only, these two values can be set to 0x00.
- FCCOB3 (only bit 0 is used) configures if FlexRAM is loaded with EEPROM data during reset sequence or not. If this bit is cleared (0x00), then FlexRAM is loaded with EEEPROM data during reset sequence. On the other hand, if this bit is set (0x01), then FlexRAM is not loaded with EEEPROM data during reset sequence, meaning that it will be act as traditional RAM after reset until its mode is changed by issuing a FlexRAM configuration command depicted in S32K1xx FlexRAM configuration.
- FCCOB4 indicates EEPROM data size. There are only two different values for this option depending on FlexRAM size.

Table 2. EEPROM data size

EEERAMSIZE value (FCCOB4[3:0])	EEPROM data size (Bytes)
0xF	0
0x3[1]	2K
0x2[2]	4K

1 Valid size only for devices with 2KB of FlexRAM (S32K11x devices).
.
2 Valid size only for devices with 4KB of FlexRAM (S32K14x devices).
.

NOTE

EEPROM Data Size must be set to 0 Bytes when the FlexNVM Partition Code is set for no EEPROM operation (i.e. when FlexNVM is configured to be used just as D-Flash).

- FCCOB5 indicates how to split the FlexNVM block between data flash memory (D-Flash) and emulated EEPROM backup memory (E-Flash). There are different values for this option depending on FlexNVM size. Following tables show these options.

Table 3. FlexNVM partition codes for 32 kB FlexNVM devices (S32K11x)

FlexNVM Partition Code (FCCOB5[3:0])	Data flash Size (Kbytes)	EEPROM-backup Size (Kbytes)
0x0	32	0
0x3	0	32
0x8	0	32
0x9	8	24
0xB	32	0

Table 4. FlexNVM partition codes for 64 kB FlexNVM devices (S32K142, S32K144, S32K146)

FlexNVM Partition Code (FCCOB5[3:0])	Data flash Size (Kbytes)	EEPROM-backup Size (Kbytes)
0x0	64	0

Table continues on the next page...

Table 4. FlexNVM partition codes for 64 kB FlexNVM devices (S32K142, S32K144, S32K146) (continued)

FlexNVM Partition Code (FCCOB5[3:0])	Data flash Size (Kbytes)	EEPROM-backup Size (Kbytes)
0x3	32	32
0x4	0	64
0x8	0	64
0xA	16	48
0xB	32	32
0xC	64	0

Table 5. FlexNVM partition codes for 64 kB FlexNVM devices (S32K148)

FlexNVM Partition Code (FCCOB5[3:0])	Data flash Size (Kbytes)	EEPROM-backup Size (Kbytes)
0x0	512	0
0x4	448	64
0xF	512	0

Prior to launch program partition command, data flash IFR must be in an erased state. After launching program partition code, EEPROM backup memory size and EEESIZE values are saved on data flash IFR region, and DEPART field of FTFC_FCFG1 register is loaded during system reset from the data flash IFR region.

NOTE

SIM_FCFG1[DEPART] field saves current FlexNVM partition code, hence, it is recommended that prior to launch program partition command, user checks this field and validates FlexNVM is not partitioned already (DEPART field contains 0xF for non-partitioned devices) to avoid partition command error.

3.1.1.2 Program partition command examples

There are multiple EEPROM backup (E-Flash) sizes available for FlexNVM. The figures below shows different partition commands and FlexNVM configuration for S32K1xx devices. Be aware that memory used for EEPROM backup is not available (mapped) for the user.

NOTE

FCCOBx's offset is different than FCCOB's index number, it means that FCCOB0 is not the first registers in the FCCOB array. For more details, refer to FCCOB's offset description from reference manual.

- Maximum E-Flash value. (32 kB for S32K11x devices, 64 kB for S32K14x devices)

```
FTFC-> FCCOB [3] = 0x80;    /* FCCOB0: Selects the PGMPART command */
FTFC-> FCCOB [2] = 0x00;    /* FCCOB1: No CSEc operation */
FTFC-> FCCOB [1] = 0x00;    /* FCCOB2: No CSEc operation */
FTFC-> FCCOB [0] = 0x00;    /* FCCOB3: FlexRAM loaded with valid EEPROM during reset
sequence */
#if S32K11x_DEVICES
FTFC->FCCOB[7] = 0x03;    /* FCCOB4: EEPROM data set size code: EEESIZE = 3 (2 kB) */
FTFC->FCCOB[6] = 0x08;    /* FCCOB5: FlexNVM Partition code: DEPART = 8 (Data flash: 0,
EEPROM backup: 32 kB) */
#elif S32K14x_DEVICES
FTFC-> FCCOB [7] = 0x02;    /* FCCOB4: EEPROM data set size code: EEESIZE = 2 (4 kB) */
FTFC-> FCCOB [6] = 0x04;    /* FCCOB5: FlexNVM Partition code: DEPART = 4 (Data flash: 0
kB, EEPROM backup: 64 kB) */
```

```
#endif
flash_launchCommand();
```

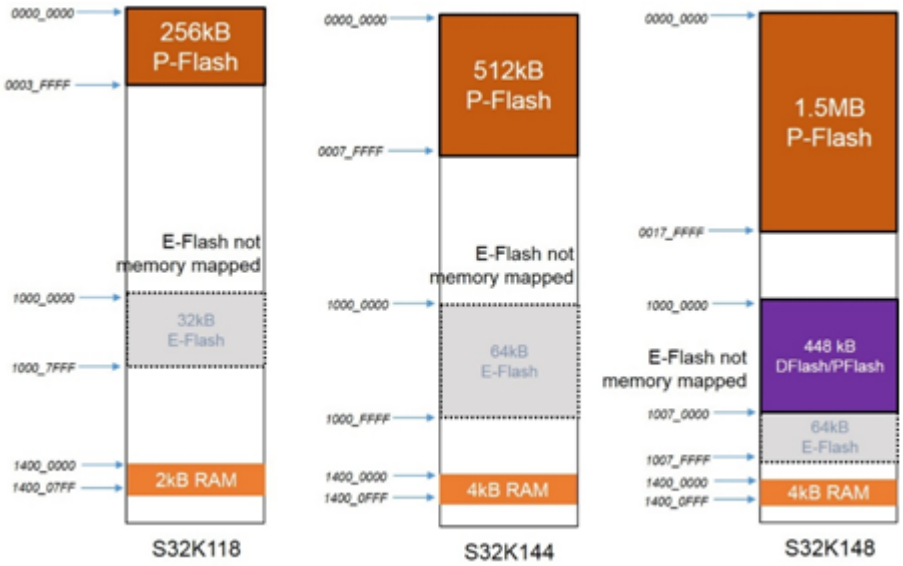


Figure 4. FlexNVM configuration for maximum E-Flash size

- Split FlexNVM for D-Flash and E-Flash. (Unavailable option for S32K148)

```
FTFC-> FCCOB [3] = 0x80;     /* FCCOB0: Selects the PGMPART command */
FTFC-> FCCOB [2] = 0x00;     /* FCCOB1: No CSEc operation */
FTFC-> FCCOB [1] = 0x00;     /* FCCOB2: No CSEc operation */
FTFC-> FCCOB [0] = 0x00;     /* FCCOB3: FlexRAM loaded with valid EEPROM during reset
sequence */
#if S32K11x_DEVICES
FTFC->FCCOB[7] = 0x03;  /* FCCOB4: EEPROM data set size code: EEESIZE = 3 (2 kB) */
FTFC->FCCOB[6] = 0x09;  /* FCCOB5: FlexNVM Partition code: 9 (Data flash 8 kB, EEPROM
backup 24 kB) */
#elif S32K14x_DEVICES
FTFC-> FCCOB [7] = 0x02;  /* FCCOB4: EEPROM data set size code: EEESIZE = 2 (4 kB) */
FTFC-> FCCOB [6] = 0x03;  /* FCCOB5: FlexNVM Partition code: 3 (Data flash 32 kB, EEPROM
backup 32 kB) */
#endif
flash_launchCommand();
```

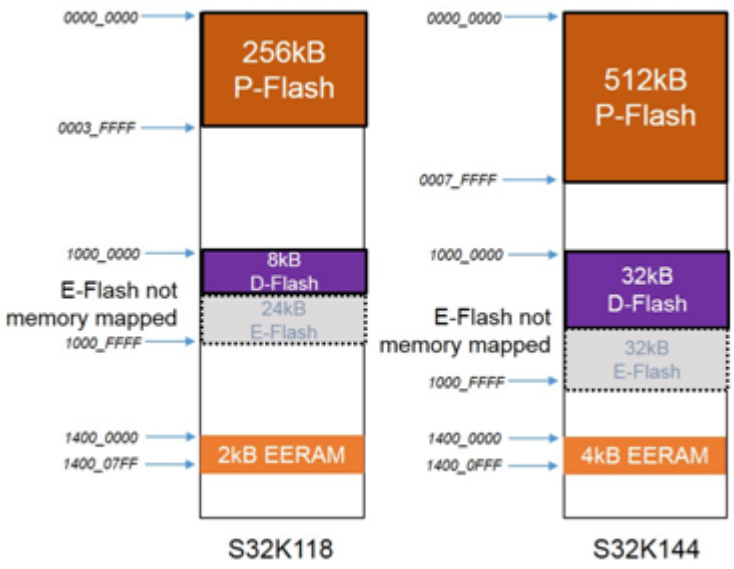


Figure 5. FlexNVM configuration for 32 kB of D-Flash and 32 kB of E-Flash

- FlexNVM for D-Flash and 0 kB of E-Flash

```
FTFC-> FCCOB [3] = 0x80;  /* FCCOB0: Selects the PGMPART command */
FTFC-> FCCOB [2] = 0x00;  /* FCCOB1: No CSEc operation */
FTFC-> FCCOB [1] = 0x00;  /* FCCOB2: No CSEc operation */
FTFC-> FCCOB [0] = 0x00;  /* FCCOB3: FlexRAM loaded with valid EEPROM during reset sequence
*/
FTFC-> FCCOB [7] = 0x0F;  /* FCCOB4: EEPROM data set size code: EEESIZE = 0x0F (0kB) */
FTFC-> FCCOB [6] = 0x00;  /* FCCOB5: FlexNVM Partition code:0 (DFlash 32/64/512 kB, EEPROM
backup 0  kB) */
flash_launchCommand();
```

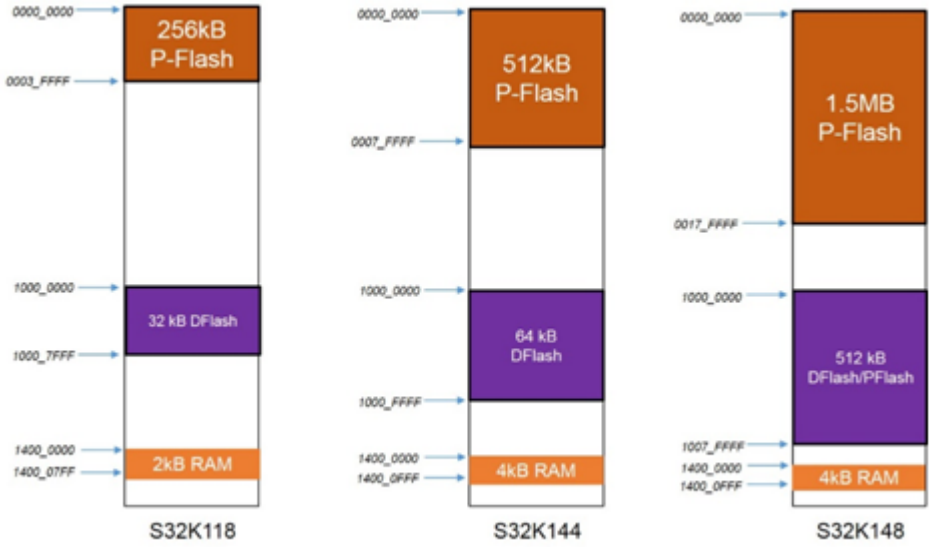


Figure 6. FlexNVM configuration for D-Flash only

Besides configuring FlexNVM and set E-Flash size, FlexRAM operation must also be configured for desired EEPROM operation.

3.2 S32K1xx FlexRAM configuration

The Set FlexRAM Function command changes the function of the FlexRAM:

- When not partitioned for emulated EEPROM (or program partition command was issued by setting FlexRAM load during reset option field to 0x01), the FlexRAM is typically used as traditional RAM.[1]
- When partitioned for emulated EEPROM, the FlexRAM is typically used to store EEPROM data.

The figure below shows the parameters needed for FlexRAM configuration command. For detailed information see FTFC chapter in Reference Manual.

FCCOB Number	FCCOB Contents [7:0]
0	0x81 (SETRAM)
1	FlexRAM Function Control Code (see Table 32-47)
2	Reserved
3	Reserved
4	Number of FlexRAM bytes allocated for EEPROM quick writes [15:8]
5	Number of FlexRAM bytes allocated for EEPROM quick writes [7:0]
5	Brown-out (BO) Detection Codes • 0x00 - No EEPROM issues detected • 0x01 - BO detected before completing EEPROM quick write maintenance • 0x02 - BO detected before completing EEPROM quick writes • 0x04 - BO detected during normal EEPROM write activity
6	Number of EEPROM quick write records requiring maintenance [15:8]
7	Number of EEPROM quick write records requiring maintenance [7:0]
8	EEPROM sector erase count [15:8]
9	EEPROM sector erase count [7:0]

Figure 7. Set FlexRAM function command

3.2.1 Set FlexRAM function command requirements

Set FlexRAM function command is used to configure FlexRAM functionality and some other quick write settings. Command will perform different operations depending on control code value usage.

- FCCOB0 defines desired command. 0x81 refers to SETRAM command (Set FlexRAM function command).
- FCCOB1. FlexRAM function control code is used to perform different actions. The table below shows different control codes and the action that each performs:

Table 6. FlexRAM function control options

FlexRAM Function Control Code	Action
0xFF	Make FlexRAM available as RAM: — Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags — Write a background of ones to all FlexRAM locations. — Set the FCNFG[RAMRDY] flag

Table continues on the next page...

[1] FlexRAM runs at different speed than SRAM, also, no ECC is available for FlexRAM.

Table 6. FlexRAM function control options (continued)

FlexRAM Function Control Code	Action
0xAA	Complete interrupted EEPROM quick write process: — Clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags. — Complete maintenance on interrupted EEPROM quick write operations. — Set the FCNFG[EEERDY] flag.
0x77	EEPROM quick write status query: — Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags. — Report emulated EEPROM status. — Set the FCNFG[EEERDY] flag.
0x55	Make FlexRAM available for EEPROM quick writes: — Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags. — Enable the emulated EEPROM system for EEPROM quick writes. — Set the FCNFG[EEERDY] flag.
0x00	Make FlexRAM available for emulated EEPROM: — Clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags. — Write a background of ones to all FlexRAM locations. — Copy-down existing EEPROM data to FlexRAM. — Set the FCNFG[EEERDY] flag.

- Remaining FCCOB parameters are used depending on which FlexRAM function control code is selected.

Set FlexRAM function command can be launched in any moment. Hence, user can be switching between using FlexRAM as traditional RAM for certain routine(s) and then switching back for some E-RAM (FlexRAM used as EEE interface) without losing EEEPROM data.

3.2.1.1 FlexRAM used as RAM

If FlexRAM is required to be used as traditional RAM, 0xFF must be used in the control code field. Other FCCOB fields are not used.

FTFC module will clear the FCNFG[EEERDY] and FCNFG[RAMRDY] flags, overwrites the content of the entire FlexRAM with a background pattern of all ones, and sets the FCNFG[RAMRDY] flag indicating that FlexRAM is ready to be used as traditional RAM, in this case, normal read and write accesses to the FlexRAM are available. Be aware that FlexRAM is slower than SRAM and they are also not in contiguous addresses between each other.

NOTE

The state of the EPROT register does not prevent the FlexRAM from being overwritten

Next snippet code shows FCCOB configuration for FlexRAM function command when FlexRAM is configured as traditional RAM.

```
FTFC-> FCCOB [3] = 0x81;   /* FCCOB0: Selects the SETRAM command */
FTFC-> FCCOB [2] = 0xFF;   /* FCCOB1: Make FlexRAM available as RAM */
flash_launchCommand();
```

3.2.1.2 FlexRAM used for normal emulated EEPROM (No quick writes)

If FlexRAM is required to be used for normal emulated EEPROM operation, 0x00 must be used in the control code field. Other FCCOB fields are not used.

NOTE

This is the default FlexRAM mode out of reset for CSEc enabled parts. So, there is no need to launch Set FlexRAM command with control code set to 0x00.

FTFC will clear the FCNFG[RAMRDY] and FCNFG[EEERDY] flags, overwrites the contents of the FlexRAM allocated for emulated EEPROM with a background pattern of all ones and copies the existing EEPROM data from the EEPROM backup record space to the FlexRAM. After completion of the EEPROM copy-down, the FCNFG[EEERDY] flag is set. When the FlexRAM is set to work as emulated EEPROM, normal read and write access to the FlexRAM are available, but writes to the FlexRAM also invoke emulated EEPROM activity.

Next snippet code shows FCCOB configuration for FlexRAM function command when FlexRAM is used for normal emulated EEPROM.

```
FTFC-> FCCOB [3] = 0x81;   /* FCCOB0: Selects the SETRAM command */
FTFC-> FCCOB [2] = 0x00;   /* FCCOB1: Make FlexRAM available for emulated EEPROM */
flash_launchCommand();
/* Wait until previous command is finished */
while ((FTFC-> FCNFG & FTFC_FCNFG_EEERDY_MASK) == 0);
```

3.2.1.3 FlexRAM used for EEPROM quick writes

If FlexRAM is required to be used for EEPROM quick writes, 0x55 must be used in the control code field. FCCOB4 and FCCOB5 fields are needed to indicate the byte amount that will be used for quick writes. These 2 fields represent a 16-bit value (FCCOB4[15:8] and FCCOB5[7:0]). Number of bytes can be chosen from values between 16 and 512 (0x0010 to 0x0200) but must be divisible by 4 otherwise FlexRAM function command will fail and FSTAT[ACCERR] flag will be set. Other FCCOB registers are not used.

NOTE

Only 32-bit quick writes are allowed, others will generate an access error.

When making the FlexRAM available for EEPROM quick writes, the FTFC will leave the contents of the FlexRAM as is and prepare the emulated EEPROM system for quick write activities. After the EEERDY flag in the FCNFG register is set, the emulated EEPROM system is ready for quick writes. When the FlexRAM is set to accept EEPROM quick writes, read and write access to the FlexRAM is available but each 32-bit write to the FlexRAM will invoke EEPROM quick write activity. The CCIF and EEERDY flag will de-assert on each 32-bit write and assert after the associated EEPROM quick write activity has completed. After the last 32-bit write to the FlexRAM, specified by the contents of the FCCOB4 and FCCOB5, the emulated EEPROM system will create the data record for the last write (~150 µs) and begin the EEPROM maintenance activities on the entire block of quick write data. The CCIF and EEERDY flag will remain negated until all EEPROM quick write maintenance activities have completed.

Next snippet code shows FCCOB configuration for FlexRAM function command when FlexRAM is configured as quick writes EEPROM.

```
FTFC-> FCCOB [3] = 0x81;   /* FCCOB0: Selects the SETRAM command */
FTFC-> FCCOB [2] = 0x55;  /* FCCOB1: Make FlexRAM available for emulated EEPROM (quick writes)
*/
FTFC-> FCCOB [1] = 0x00;   /* FCCOB2: Reserved */
FTFC-> FCCOB [0] = 0x00; /* FCCOB3: Reserved */
/* Allocate 512 bytes for quick write operation */
FTFC-> FCCOB [7] = 0x02;   /* FCCOB4: Number of bytes allocated for EEPROM quick writes [15:8]
*/
FTFC-> FCCOB [6] = 0x00;   /* FCCOB5: Number of bytes allocated for EEPROM quick writes [7:0] */
flash_launchCommand();
```

Quick write mode is described in S32K1xx New Quick Write Mode.

NOTE

Quick writes must be completed before switching to another FlexRAM functionality. Same applies if user wants to start a new quick write operation while another one is taking place; the first quick write operation must be completed prior to start a new one (there is no way to abort the ongoing operation).

3.2.1.4 FlexRAM used for querying write status on EEPROM

If FlexRAM is required to be used for querying write status, 0x77 must be used in the control code field. Other FCCOB fields are not used to launch the command, however, some of these are used to retrieve status from write operations.

After command is launched, FCNFG[EEERDY] flag is cleared, the FTFC will interrogate the emulated EEPROM system and report EEPROM cleanup requirements (on FCCOB6 and FCCOB7), EEPROM sector erase count (on FCCOB8 and FCCOB9) and FCCOB5 will save brown-out detection code depending conditions shown in the table below.

Table 7. Brownout detection code options

Brownout code	Condition
0x00	No EEPROM issues.
0x04	If EEPROM normal write activity is interrupted by a reset or loss of power before finalizing the record.
0x02	If EEPROM quick write activity is interrupted by a reset before writing all quick write records.
0x01	If the interruption occurs after all quick writes requested have been written but before quick write maintenance has completed.

The CCIF and EERDY flag will remain negated until EEPROM status has been loaded into FCCOB registers.

Next snippet code shows FCCOB configuration for FlexRAM function command when FlexRAM is querying (quick) writes status and the way to manage these status results.

```
uint8_t brownOutCode;
uint16_t quickWriteMaintenance;
uint16_t sectorEraseCount;

FTFC-> FCCOB [3] = 0x81;   /* FCCOB0: Selects the SETRAM command */
FTFC-> FCCOB [2] = 0x77;   /* FCCOB1: EEPROM quick write status query */
flash_launchCommand();   /* Launch command and wait until it finishes */
/* FCCOB5: Brown-out (BO) Detection Codes */
brownOutCode = FTFC-> FCCOB [6];
/* FCCOB6,7: Number of EEPROM quick write records requiring maintenance */
quickWriteMaintenance = FTFC-> FCCOB [5] << 8 | FTFC-> FCCOB [4];
/* FCCOB8,9: EEPROM sector erase count */
sectorEraseCount = FTFC-> FCCOB [11] << 8 | FTFC-> FCCOB [10];
```

If a reset occurs before the last quick write is completed, brownout code will be set to 0x02 meaning that previous write is ignored and no update will take effect, leaving the last records as they were. If a reset occurs after all quick writes are completed (but maintenance couldn't finish), brownout code will be set to 0x01 and these records can be completed later (by using the FlexRAM command to complete interrupted quick write process).

If a reset occurs before normal write activity finishes, Brownout code will be set to 0x04 meaning that previous write is ignored and no update will take effect.

3.2.1.5 FlexRAM used to complete interrupted EEPROM quick write process

If FlexRAM is required to be used to complete interrupted EEPROM write process, 0xAA must be used in the control code field. Other FCCOB fields are not used.

After command is launched, FCNFG[EEERDY] flag is cleared, the FTFC will process all EEPROM quick write activity not previously completed. If the EEPROM quick write activity did not complete writing all requested records, the records written will be cleared, effectively restoring previously valid records. If all EEPROM quick write records were written but maintenance activities were not completed, maintenance will complete on the remaining quick write records. The CCIF and FCNFG[EEERDY] flag will remain negated until all EEPROM maintenance activities have been completed.

Next snippet code shows FCCOB configuration for FlexRAM function command when it is used to complete interrupted EEPROM operations.

```
FTFC-> FCCOB [3] = 0x81;   /* FCCOB0: Selects the SETRAM command */
FTFC-> FCCOB [2] = 0xAA;   /* FCCOB1: Complete interrupted   */
flash_launchCommand();   /* Launch command and wait until it finishes */
```

3.3 S32K1xx command error handling

For each flash command related to EEPROM functionality (Program Partition command and set FlexRAM function), there are error conditions that cause the command to fail. Program Partition command error handling and Set FlexRAM function command error handling tables in S32K1xx Reference Manual show these conditions and affected flags for each of these conditions. Be sure to avoid these error conditions and check error flags each time these commands are launched.

NOTE

When set, Access Error (ACCER) and Flash Protection Violation (FPVIOL) flags from FSTAT register prevent the launch of any more commands or writes to the FlexRAM until the flag is cleared (by writing a one to it).

3.4 S32K1xx EEE startup

After a reset the EEE configuration options written during the partitioning process are automatically loaded. If the EEE is enabled, then the state machine loads the FlexRAM with EEE data from the E-flash during the system boot. The amount of time needed to copy data from the E-flash into the FlexRAM can vary depending on the configured size of the EEE and the amount of backup E-flash that needs to be parsed. The FTFC_FCNFG[EEERDY] flag is cleared until the load of the EEE data is complete, therefore software must wait for the EEERDY flag to set before attempting to access the EEE data in the FlexRAM. If an interrupt driven option is needed instead of software polling, then the CCIF interrupt could be used instead of polling EEERDY.

If FlexRAM load during reset option field was set to 0x01, EEE data won't be copied to FlexRAM as it is used as traditional RAM after reset. User must issue a Set FlexRAM command to change FlexRAM functionality to EEERAM when needed. When CSEc functionality is enabled, user is not able to get delayed EEE reset option (user must set reset option field to 0x00 in program partition command).

Polling for FTFC_FCNFG[EEERDY] and/or FTFC_FCNFG[RAMRDY] after reset sequence is a good practice to get current FlexRAM mode.

3.5 S32K1xx reading and writing the EEE

The EEE data is read and written by accessing the FlexRAM address space. The EEE space is allocated starting at the beginning of the FlexRAM. The addressable space is the FlexRAM base address (0x1400_0000) up to the programmed EEE size.

Because the EEE data is accessed through a RAM, the data is readable and writable at any size, byte, word, or longword. Although any access size is possible, the records used to back up the EEE data use a word sized data field. This means that byte writes are possible, but they make less efficient use of the E-flash.

3.5.1 EEE writes

Writes to the EEE space launch a EEE operation to store the data within the E-flash memory. Because this is a flash program operation, software must test the CCIF bit to determine if any other flash operations are in progress before writing to the EEE space. Because multiple concurrent writes and read-while-write operations within the same flash block are not allowable, accesses to the EEE or D-flash space are not allowed until the EEE write is complete. Check CCIF flag to determine if previous Flash or EEEPROM command is finished.

NOTE

The P-flash memory is a completely separate logical block, therefore read accesses to the P-flash can continue normally while a EEE write is in progress. It does not apply for last 448 kB of P-Flash in S32K148 device that are located in the same memory block than E-Flash.

EEE writes don't have to be to sequential location, indeed user can write to any valid location and/or write to the same address more than once.

NOTE

EEE writes cannot be performed unless device is in RUN mode. For HSRUN/VLPR only EEE reads are allowed.

3.5.2 EEE reads

When the EEE is read, the data is supplied by the FlexRAM, so no flash operations are triggered. However, EEE reads are not allowed while a EEE write is in progress. Software must wait for CCIF after a write access before allowing software to continue. This way a special function is needed for EEE writes, but a EEE read does not require any special software. Another advantage to this approach is that no additional delay or flag checking is required if you have multiple EEE read accesses with no EEE write cycles in between. A special case for a EEE read that must be considered is the first access to the EEE after a reset. For the first read of the EEE after reset, the EEERDY bit may need to be tested to make sure that the state machine has completed the initial load of data from the E-flash to the FlexRAM. If the system start-up time is long, this guarantees that the initial data load has time to complete before the first EEE read, then a test of the EEERDY flag before the first read may not be required. However, it is safer to explicitly test the EEERDY bit before the first read access to the EEE.

NOTE

Certain CSEc boot options (for example, MAC boot options) might prevent FCNFG[EEERDY] flag to be set until boot process is finished correctly.

4 S32K1xx EEE performance

In addition to the flexibility and high endurance provided by S32K1xx EEE implementation, S32K1xx EEE is faster than a typical EEPROM. A traditional external EEPROM typically requires around 5 ms for the maximum program time. By comparison, the EEE can be erased and written in 1.5 ms worst case scenario. The EEE can also be pre-erased by writing 0xFF to the EEE data locations. Pre-erasing locations help to reduce the program time because it guarantees that an erase cycle will not be needed. Typical program time to a pre-erased data location is ~100 µs. This feature allows for quick data logging in time critical situations. One typical use case is a system where fault data or operating information needs to be stored when an imminent power loss is detected. The amount of data that needs to be saved times the maximum write time determines how much decoupling must be provided in the system to maintain minimum operating power long enough for the data to be stored. The significant decrease in the EEE program time gained by pre-erasing data locations means that in this situation less decoupling is required and more data can be stored before losing power.

Additionally, using quick write mode for data records in cases where brownout is detected, ensures that critical data can be written to E-Flash in a smaller amount of time rather than normal write mode.

5 S32K1xx brownout detection

The EEE state machine includes logic that can detect if any EEE data has not been fully programmed. This feature is referred to as brownout detection in documentation, but the name is not completely accurate. Any situation where EEE data is detected as not fully programmed due to either a brownout or any reset during the write process is treated the same. If a reset happens while a EEE write is in progress, then the data can become corrupted. The EEE state machine tests vulnerable EEE data record for values that may not be fully programmed. If an incomplete record is detected, the state machine marks the data record as *compromised* and replaces it with the previous valid data record for the associated EEE address during the next EEE write. This ensures that if a EEE write is interrupted for any reason, you will get the last value that was properly written to the EEE. Depending on how far into the write the reset occurred, this value can be either the previous value or the new value, but you will not get a corrupted value.

You can check section FlexRAM used for querying write status on EEPROM. For more information about Brownout detection.

To avoid brownout conditions in S32K1xx devices, it is necessary to properly follow hardware recommendations shown in Hardware Design Guidelines to ensure that system is powered while completing any EEPROM backup sequence. As long as robust power supply hardware is included, less brownout conditions will be faced.

6 S32K1xx new quick write mode

Quick write mode priority-writes data, e.g. before imminent brownout. EEE record maintenance can be performed after reset, if necessary. Quick write mode is limited to 32-bit writes only. Normal writes cannot resume until Quick Write maintenance is completed. All requested bytes have to complete the program operation for any of them to be valid. If write operation interrupted by reset or power lost before last byte written, none of the writes are valid and last records are the valid.

The figure below shows the principle of Normal record write process to Flash and Quick write process. Normal write process performs incremental maintenance with each record write. Quick write mode writes records as fast as possible, postponing maintenance until later. This Quick write mode makes full recovery possible if power is lost during maintenance.

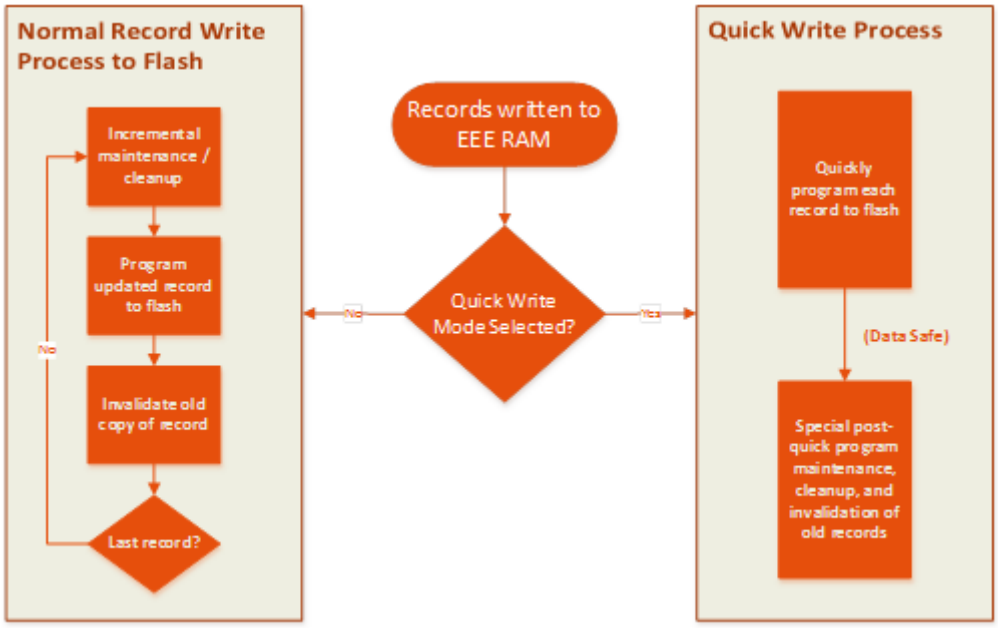


Figure 8. S32K1xx quick write process mode

Quick Write time is a 66% improvement compared to Normal Write mode (however maintenance must eventually be performed). The figure below shows the write time of Quick Write mode compared with Normal Write mode.

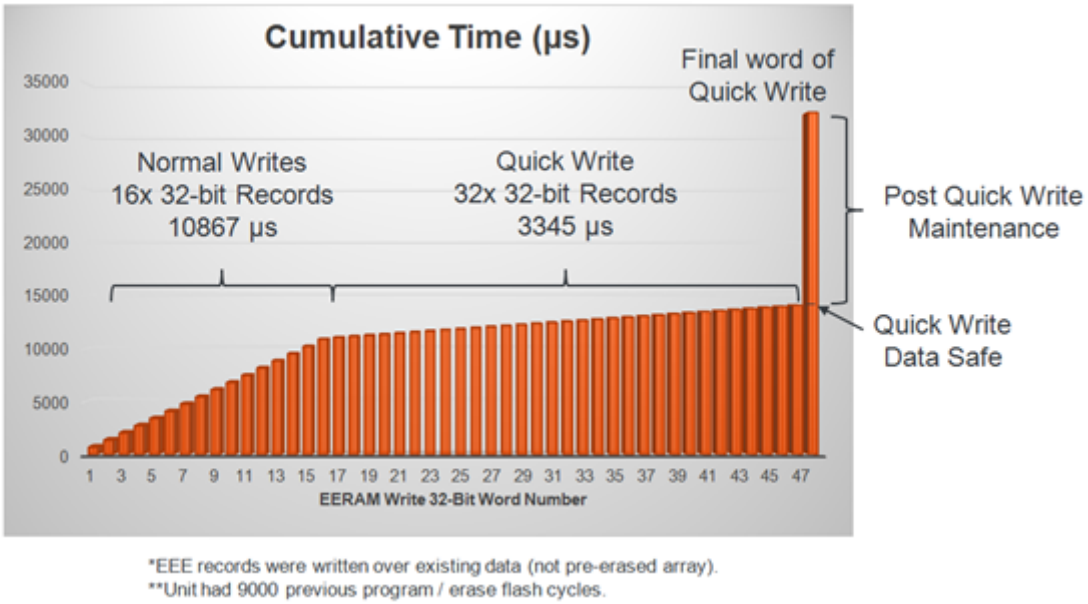


Figure 9. S32K Quick write process write time characterization

NOTE

These numbers used are for reference only, not to be taken as absolute numbers.

Quick Write process takes less time to write all records into E-Flash, however, total time (considering post quick write maintenance time) can be longer than normal writes. The advantage of Quick write is that, once last quick write data has been written (EEE reaches the quick write data safe point from previous figure) any brownout event does not cause data to be lost as maintenance can be done later. However, if brownout occurs before writing the last data, no new data will be saved.

If a brownout condition occurs before post quick write maintenance is completed, Maintenance can be resumed in the next reset cycle by issuing a Set FlexRAM command to complete interrupted quick write process. It is explained in depth in FlexRAM used for EEPROM quick writes.

There are times when quick writes could seem slower than normal writes depending on background clean up happening in the EEE system. Quick writes could take longer than normal writes for the total time because the record status update and compression process is different in both modes, however, the advantage of quick write over normal write is that it's faster to write new data since it first saves data into E-Flash before cleanup process and in case that cleanup process is not completed due a brownout situation, then you can resume the activity on the next power on reset.

7 S32K1xx EEPROM endurance

As FlexNVM partition code choices affect the endurance and data retention characteristics of the devices, it is important to select the best partition code/ EEERAM size according application's requirements.

In order to achieve specified w/e cycle endurance shown in S32K1xx's datasheet, the emulated EEPROM backup size must be at least 16 times the emulated EEPROM partition size in the FlexRAM (EEERAM). This can be achieved by using all available E-Flash (64 kB for S32K14x devices and 32 kB for S32K11x devices) with maximum EEERAM size (4 kB for S32K14x devices and 2 kB for S32K11x devices). In case that user wants to allocate less than 64kB of E-Flash, for example 48kB in S32K14x devices, EEERAM size must be less than 4kB (3kB) in order to achieve endurance shown in Datasheet. (48kB / 3Kb = 16).

If higher endurance is desired, utilizing fewer records will increase the ratio of RAM to NVM, thus, increasing the w/e endurance.

NOTE

The way to increase the ratio of RAM to NVM beyond 16 is to use less RAM addresses in the application. If user only write to 2 kB of the total 4 kB of FlexRAM for S32K14x devices, then ratio effectively changes from 1:16 to 1:32, doubling the w/e cycle endurance.

NXP offers a tool to estimate numbers of write/erase cycles performed on the FlexNVM, average time between each refresh FlexNVM cycle and worst case in number of EEPROM cycles required by the user's application. You can download the tool from NXP website (FlexMemory Endurance Calculator) to calculate these values according your application's requirements.

8 Software considerations

This section is focused to list some software considerations/recommendations when using Emulated EEPROM functionality on S32K1xx devices.

8.1 Simultaneous operations

Some operations on flash module cannot be executed simultaneously because certain hardware resources are shared by the program flash (P-Flash), data flash (D-Flash) and FlexRAM (EEERAM). In general, none of the CCOB commands must not be executed simultaneously with any CSEc command and vice versa (See CSEc chapter on Reference Manual for details).

Allowed simultaneous memory operations table on Reference Manual shows simultaneous memory operations for P-Flash, D-Flash and FlexRAM (when configured as EEPROM and traditional RAM).

For E-write, only reading from P-Flash simultaneously is allowed.

NOTE

For S32K148, last 512 kB bank is shared between E/D-Flash and P-Flash. No read-while-write access is possible in this single bank. Be sure to poll for CCIF flag before issuing another access in this bank either through CSEc, D-Flash, P-Flash or EEEPROM operation. For more details, there is a separate Application note about FlexNVM operation in S32K148 device (Using S32K148 FlexNVM memory).

Simultaneous access will cause a collision that could be reported and interruptible through FTFC Read Collision Error Flag (RDCOLERR) and previous operation that was running will be invalidated as well.

8.2 Enabling CSEc and EEPROM

When CSEc operation is also enabled, it will use up to 512 B for key storage, leaving EEPROM to use the remaining size for EEERAM (3.5 kB for S32K14x devices and 1.5 kB for S32K11x devices).

Also, if CSEc is needed, to maintain the full specified w/e endurance in the emulated EEPROM system, the number of records stored in FlexRAM should be limited to maintain the 1:16 ratio of FlexRAM to FlexNVM. To achieve this ration, it is recommended to use the maximum E-Flash space available.

Program partition command must use the maximum available E-Flash size (64kB for S32K14x devices and 32kB for S32K11x devices).

8.3 Power-On recommendation

At power-on, wait for EERDY flag to be set before reading/writing to EEERAM. Also, polling for normal/quick write status by issuing a set FlexRAM function command is highly recommended.

In cases where FlexRAM is not configured to load E-Flash data on reset sequence, user must configure FlexRAM to act as EEERAM by issuing a Set FlexRAM command for either Normal or Quick Writes mode before reading/writing from EEEPROM. After command is launched, wait for FTFC_FCNFG[EEERDY] flag to be set.

The first write after reset would potentially be longer due the need to cleanup brownout activity, so, it is recommended to use Set FlexRAM command to address query status/incomplete process as shown in Appendix section.

8.4 Data record checking

As FlexRAM does not support ECC, user must implement software check (i.e. by using CRC) to detect record errors. Besides this, backing up more than once every safety-relevant/critical data (i.e. saving 3 times or more each record and not using adjacent locations in the FlexRAM) can increase the reliability of the data.

As stated in S32K14x Series Safety Manual document, the way to check Emulated EEPROM data must be implemented based on SETRAM command. After EEE driver copies backup data to EEERAM (FlexRAM) user must implement integrity checks (such as MISR, CRC) on EEERAM, then, perform SETRAM to switch to system RAM, followed by SETRAM to switch back to EEERAM and once the content from EFlash is copied down to EEERAM, another integrity check must be done. If the two results are the same, then, the EFlash contents that were copied down to EEERAM are as expected. If they are different, then likely the first copy was corrupted in some way (power loss, etc). For more details, see the device Safety Manual.

8.5 Power Modes transition

As Emulated EEPROM's writes can only be performed when device is in RUN mode, if user desires to switch to any other mode (HSRUN mode to operate at higher frequency for example or VLPR mode to decrease the power consumption value) it is recommended that, prior to switching to any power mode, EEPROM's writes must be completed first. In case that device is running in HSRUN/VLPR mode and an EEPROM write is needed, user must switch to RUN mode to perform the write and after it finishes, user can switch back to HSRUN/VLPR mode.

EEPROM's reads can be performed in HSRUN or VLPR mode as stated in Module operation in available low power modes table included in device's reference manual.

9 Appendix A EEPROM examples

This section is focused on showing different scenarios where EEEPROM can be involved and the way to handle each of these. It provides a brief guide on different situation such as EEEPROM initialization and brownout events

1. Although EEEPROM partition command is intended to be used once in product lifetime, below figure shows recommended flow chart after reset event which includes FlexNVM partition query. It shows how the SetFlexRAM command is issued to configure FlexRAM in its multiple modes such as normal/quick write mode, query for quick write status or complete interrupted quick writes where maintenance was aborted before its completion.

   This is just a starting point specially to get involved with FlexNVM and EEEPROM functionality.

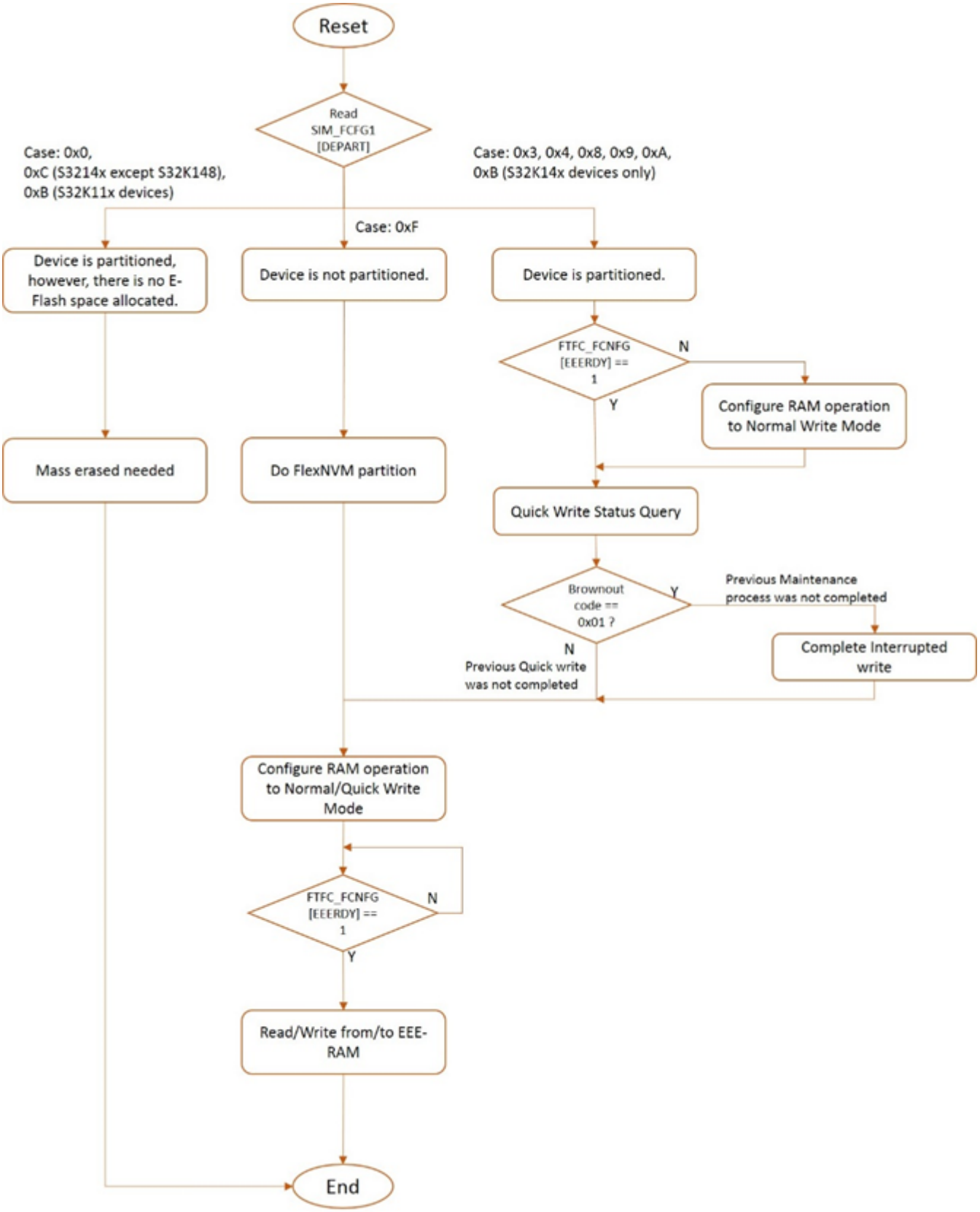


Figure 10. EEE initialization flow chart

2. EEEPROM quick write usage: As Quick write mode is intended to be used to save critical status records prior brownout events, user can implement a scheme where system switches between normal and quick writes when needed. As all records, must be written into FlexRAM before brownout condition happens, it is highly recommended to configure quick write mode using the necessary number of bytes to warranty that all bytes are written before brownout condition.

   Hardware must provide enough time to keep the system powered until last word is written. Time needed can be obtained from datasheet and it relies on number of written bytes and the time it takes for every 32-bit write.

   Following flow chart shows a way to handle quick write procedure by using Low Voltage Detect (LVD) interrupt routine. For detailed information about LVD please refer to section 39.5 Low Voltage Detect System in Reference Manual.

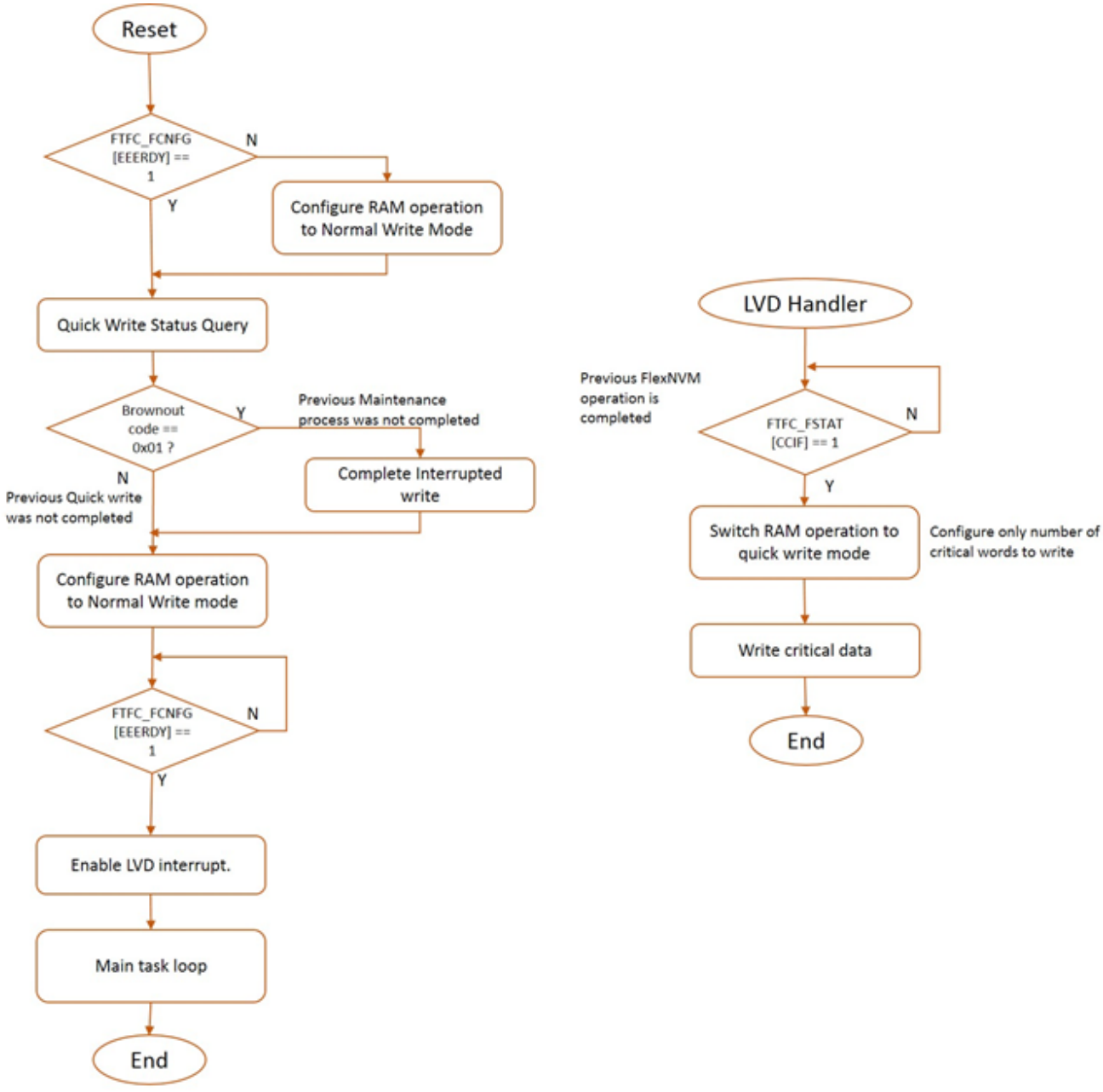


Figure 11. EEE quick write usage flowchart

10 Revision history

Rev. No.	Date	Substantive Change(s)
0	June 2017	Initial release
1	March 2018	• Updated Power-On recommendation for fixing the description for FlexRAM loaded in reset sequence • Updated Program partition command requirements • Added a note in Program Partition Command examples • Updated S32K1xx new quick write mode by adding reference for Quick write regarding Figure 11 • Added Appendix A for EEEPROM use cases • Added Power Modes transition on page 19 • Added a new note in FlexRAM used for EEPROM quick writes on page 12

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, COOLFLUX, EMBRACE, GREENCHIP, HITAG, I2C BUS, ICODE, JCOP, LIFE VIBES, MIFARE, MIFARE CLASSIC, MIFARE DESFire, MIFARE PLUS, MIFARE FLEX, MANTIS, MIFARE ULTRALIGHT, MIFARE4MOBILE, MIGLO, NTAG, ROADLINK, SMARTLX, SMARTMX, STARPLUG, TOPFET, TRENCHMOS, UCODE, Freescale, the Freescale logo, AltiVec, C-5, CodeTEST, CodeWarrior, ColdFire, ColdFire+, C-Ware, the Energy Efficient Solutions logo, Kinetis, Layerscape, MagniV, mobileGT, PEG, PowerQUICC, Processor Expert, QorIQ, QorIQ Qonverge, Ready Play, SafeAssure, the SafeAssure logo, StarCore, Symphony, VortiQa, Vybrid, Airfast, BeeKit, BeeStack, CoreNet, Flexis, MXC, Platform in a Package, QUICC Engine, SMARTMOS, Tower, TurboLink, and UMEMS are trademarks of NXP B.V. All other product or service names are the property of their respective owners. ARM, AMBA, ARM Powered, Artisan, Cortex, Jazelle, Keil, SecurCore, Thumb, TrustZone, and µVision are registered trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. ARM7, ARM9, ARM11, big.LITTLE, CoreLink, CoreSight, DesignStart, Mali, mbed, NEON, POP, Sensinode, Socrates, ULINK and Versatile are trademarks of ARM Limited (or its subsidiaries) in the EU and/or elsewhere. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2018 NXP B.V.

Document Number: AN11983
Rev. 1, March 2018