AN14954

Fuse Usage on i.MX Devices

Rev. 1.1 — 6 March 2026 Application note

Document information

Information	Content
Keywords	AN14954, i.MX, Fuses
Abstract	On i.MX processors, the eFuses (or fuses) are permanent, nonvolatile bits. They can be programmed once (0 → 1) and then read many times.

1 Introduction

On i.MX processors, the eFuses (or fuses) are permanent, nonvolatile bits. They can be programmed once (0 → 1) and then read many times. They store the immutable configuration and identity, such as boot configuration, MAC addresses, silicon ID, secure boot keys, and device life cycle flags. Programming is irreversible. Incorrect programming can permanently disable features or brick a device.

2 Fuse control IP

This chapter describes the fuse control IP.

2.1 OCOTP_CTRL

The OCOTP_CTRL module is a hardware IP block that manages access to the eFuses. It provides read/write access to internal eFuses and it loads the eFuse values into shadow registers for normal software access.

OCOTP_CTRL consists of the IP control/status registers and shadow registers. When powering on, OCOTP_CTRL reads the values from the fuse memory into shadow registers, so the software can access the fuse data without directly accessing the fuse array. The shadow registers must be reloaded manually after fuse writing.

OCOTP_CTRL is accessible in different ways for each i.MX device, as shown in Table 1.

Table 1. OCOTP_CTRL ownership

Device	Security subsystem	Accessible by
i.MX 6/7/8M	None	System memory
i.MX 8/8X/8XLite	SECO	SECO and SCU
i.MX 8ULP/9	ELE-AP	ELE-AP

2.2 Fuse status block

For devices with ELE-AP, eFuses are managed by ELE-AP and ELE-AP provides an MU interface. Using this interface to access the global fuse and status information is slow. The FSB is a simple interface to allow the SoC to directly read public fuses and ELE status information without using the MU.

Note: *The FSB interface exposes only a limited set of non-security-related fuses. Access the security-sensitive fuses through the ELE-AP MU.*

3 Fuse map

The eFuses are organized by OCOTP_CTRL in three dimensions. OCOTP_CTRL is commonly used in fuse read/write.

- Bank - a group of eight or four words (different for each device; most of them have eight)
- Word - 32 bits (32 fuses)
- Fuse - one bit

The fuse definition and address are documented in the **Fuse map** chapter in the device reference manual.

Note:

1. *For devices with SCFW, like i.MX 8XL, only bank 0 is used. The fuses are located with a word index.*

2. *Some fuses have different interpretations, depending on the software (what the reference manual describes and what the software interprets). See the correct reference manual or software documentation, such as i.MX Linux User's Guide (document UG10163), for the target variant. For example, NUM_OF_MACS is not used as the maximum MAC number in U-Boot.*

3.1 Devices without ELE-AP

For devices without ELE-AP, the fuse is accessed directly through OCOTP_CTRL. The **OCOTP_CTRL** chapter in the reference manual describes the fuse definition and address in the OCOTP_CTRL shadow fuse memory, which starts from offset 0x400. The **Fuse map** chapter in the reference manual provides detailed information about the boot-configuration fuse, lock fuse, and other fuses.

To locate the fuse bank/word address, either:

- Check the OCOTP_CTRL shadow fuse definition.
  For example, Figure 1 shows the bank/word definition in the **i.MX 8M Mini OCOTP_CTRL** chapter of the reference manual:

3035_0400	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_HW_OCOTP_LOCK)	32	R/W	0000_0000h	6.3.4.9/ 882
3035_0410	Value of OTP Bank0 Word1 (Tester Info.) (OCOTP_HW_OCOTP_TESTER0)	32	R/W	0000_0000h	6.3.4.10/ 884
3035_0420	Value of OTP Bank0 Word2 (tester Info.) (OCOTP_HW_OCOTP_TESTER1)	32	R/W	0000_0000h	6.3.4.11/ 884
3035_0430	Value of OTP Bank0 Word3 (Tester Info.) (OCOTP_HW_OCOTP_TESTER2)	32	R/W	0000_0000h	6.3.4.12/ 885
3035_0440	Value of OTP Bank1 Word0 (Tester Info.) (OCOTP_HW_OCOTP_TESTER3)	32	R/W	0000_0000h	6.3.4.13/ 885
3035_0450	Value of OTP Bank1 Word1 (Tester Info.) (OCOTP_HW_OCOTP_TESTER4)	32	R/W	0000_0000h	6.3.4.14/ 886
3035_0460	Value of OTP Bank1 Word2 (Tester Info.) (OCOTP_HW_OCOTP_TESTER5)	32	R/W	0000_0000h	6.3.4.15/ 886
3035_0470	Value of OTP Bank1 Word3 (Boot Configuration Info.) (OCOTP_HW_OCOTP_BOOT_CFG0)	32	R/W	0000_0000h	6.3.4.16/ 887

Figure 1. Fuse address in i.MX 8M Mini

  For i.MX 8M Mini, there are four words for each bank. Figure 2 shows the bank/word definition of i.MX 6UL:

21B_C400	Value of OTP Bank0 Word0 (Lock controls) (OCOTP_LOCK)	32	R	0000_0000h	35.5.11/ 2185
21B_C410	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (OCOTP_CFG0)	32	R/W	0000_0000h	35.5.12/ 2188
21B_C420	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (OCOTP_CFG1)	32	R/W	0000_0000h	35.5.13/ 2188
21B_C430	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (OCOTP_CFG2)	32	R/W	0000_0000h	35.5.14/ 2189
21B_C440	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (OCOTP_CFG3)	32	R/W	0000_0000h	35.5.15/ 2189
21B_C450	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (OCOTP_CFG4)	32	R/W	0000_0000h	35.5.16/ 2190
21B_C460	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (OCOTP_CFG5)	32	R/W	0000_0000h	35.5.17/ 2190
21B_C470	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (OCOTP_CFG6)	32	R/W	0000_0000h	35.5.18/ 2191
21B_C480	Value of OTP Bank1 Word0 (Memory Related Info.) (OCOTP_MEM0)	32	R/W	0000_0000h	35.5.19/ 2191
21B_C490	Value of OTP Bank1 Word1 (Memory Related Info.) (OCOTP_MEM1)	32	R/W	0000_0000h	35.5.20/ 2192
21B_C4A0	Value of OTP Bank1 Word2 (Memory Related Info.) (OCOTP_MEM2)	32	R/W	0000_0000h	35.5.21/ 2192

Figure 2. Fuse address in i.MX 6UL

For i.MX 6UL, there are eight words for each bank.

- Calculate it according to the offset address. If the fuse offset is given, calculate the fuse bank/word index as:
  - Bank = ((offset – 0x400) / 0x10) / words_per_bank
  - Word = ((offset – 0x400) / 0x10) % words_per_bank

  Where **words_per_bank** is either 4 or 8, depending on the device.

The fuse definition is in the corresponding OCOTP_CTRL shadow fuse register definition and the fuse map description table in the **Fuse map** chapter of the reference manual.

3.2 Devices with ELE-AP

For devices with ELE-AP, access the fuses by the ELE-AP MU API or FSB. The fuse map is provided as a spreadsheet attached in the device reference manual (for public fuses) and security reference manual (for security fuses).

The fuse map spreadsheet provides details about the fuse number, such as bank, word index, fuse index, and fuse definition, as shown in Figure 3.

BankNum	Word	Packed_FuseIndex	Packed_NumFuses	Packed_Bit_Names	Fuse_Function	Fuse_Setting
0	0					
		0	3	BOOT_CFG_LOCK[2:0]	Boot Configuration Fuse Lock for Boot_CFG 0-3 fus	`000` - Unlocked (fu
		12	3	LOCK_CFG_LOCK[2:0]	Lock for Fuse Lock bits	`000` - Unlocked (fu
0	1					
		35	3	COM_DEVICE_ID0_LOCK[2:0]	Communication Device ID0 Lock - Ethernet	`000` - Unlocked (fu
		38	3	COM_DEVICE_ID1_LOCK[2:0]	Communication Device ID1 Lock - USB, Ethernet	`xx1` - WP (Write Pr
0	2					
		64	3	OEM_SW_CFG_LOCK[2:0]	OEM SW Fuse Lock bits	`000` - Unlocked (fu
		79	1	BOOT2_CFG_LOCK_0[0]	Boot Configuration Fuse Lock for Boot_CFG 8-11 fu	`0` - Unlocked (fuse
0	3					
		111	1	BOOT2_CFG_LOCK_1[1]	Boot Configuration Fuse Lock for Boot_CFG 8-11 fu	`0` - Unlocked (Fus
0	4					
		143	1	BOOT2_CFG_LOCK_2[2]	Boot Configuration Fuse Lock for Boot_CFG 8-11 fu	`0` - Unlocked (fuse
1	9					
		291	3	GPR2_LOCK[2:0]	GPR2_CFG lock bits	`000` - Unlocked (fu
		294	3	GPR3_LOCK[2:0]	GPR3_CFG lock bits	`000` - Unlocked (fu
		297	3	GPR4_LOCK[2:0]	GPR4_CFG lock bits	`000` - Unlocked (fu
1	12			ECID[127:96]	Device Unique ID	UNIQUE_ID[127:96],
1	13			ECID[95:64]	Device Unique ID	UNIQUE_ID[95:64], L
1	14			ECID[63:32]	Device Unique ID	UNIQUE_ID[63:56], D UNIQUE_ID[55:48], D UNIQUE_ID[47:43], V UNIQUE_ID[42:32]
1	15			ECID[31:0]	Device Unique ID	UNIQUE_ID[31:0], CF

Figure 3. Fuse map spreadsheet in i.MX 93 reference manual

Calculate the fuse bank/word by the fuse word index in the spreadsheet column C, as shown in the above example:

- Bank = word_index / 8
- Word = word_index % 8

The bit fields for certain fuses are calculated by **Packed_FuseIndex** and **Packed_NumFuses**:

- Start bit in fuse word = Packed_FuseIndex % 32
- End bit in fuse word = Start bit in fuse word + Packed_NumFuses - 1

For example, the GPR3_LOCK fuse is in bank 1 word 1 [6:8]:

- Bank = 9 / 8 = 1
- Word = 9 % 8 =1
- Start bit = 294 % 32 = 6
- End bit = 6 + 3 – 1 = 8

If using the MU API, see the device ELE-AP user guide for details. The following commands are used and each API has different usage and parameters:

- READ FUSE(0x97)
- WRITE FUSE(0xD6)
- WRITE SECURE FUSE(0x91)
- WRITE SHADOW FUSE(0xF2)
- READ SHADOW FUSE(0xF3)

If using the FSB, since the fuse value register in the FSB starts from offset 0x8000, the corresponding fuse offset in the FSB is:

- Fuse value register offset = (8 * bank + word) * 4 + 0x8000

Note:

1. *All devices with ELE-AP use eight words per bank.*
2. *The fuse register offset in the FSB for i.MX 8ULP is 0x800.*

3. *The WRITE SECURE FUSE API is used to program a security sensitive fuse in a secure manner with a signed message.*

4 Fuse usage

There are four fuse operations, as shown in Table 2:

Table 2. Fuse operations

Operation	Function
Read/sense fuse	Get the physical fuse value
Write/burn/program fuse	Burn the physical fuse, only from 0 to 1, irreversible
Read shadow	Read the shadow fuse value
Write/override shadow	Write to the shadow fuse register

You can use the fuses in several software components and tools:

- U-Boot
- SM
- Linux user space
- SCFW
- SPSDK

Note: *Other software components, such as SPL or OP-TEE, can access the fuses. However, this document focuses on usage through command-line interfaces.*

4.1 U-Boot

U-Boot provides a unified command-line tool for fuse usage (**fuse**):

```
u-boot=> fuse
fuse - Fuse sub-system

Usage:
fuse read <bank> <word> [<cnt>] - read 1 or 'cnt' fuse words,
    starting at 'word'
fuse cmp <bank> <word> <hexval> - compare 'hexval' to fuse
    at 'word'
fuse readm <bank> <word> <addr> [<cnt>] - read 1 or 'cnt' fuse words,
    starting at 'word' into memory at 'addr'
fuse sense <bank> <word> [<cnt>] - sense 1 or 'cnt' fuse words,
    starting at 'word'
fuse prog [-y] <bank> <word> <hexval> [<hexval>...] - program 1 or
    several fuse words, starting at 'word' (PERMANENT)
fuse override <bank> <word> <hexval> [<hexval>...] - override 1 or
    several fuse words, starting at 'word'
```

Once the fuse bank, word, or bits are identified, you can use the fuse command to sense/program the physical fuse or read/override the shadow fuse.

The example is as follows:

```
u-boot=> fuse read 47 0 8
Reading bank 47:
```

```
Word 0x00000000: 00000001 00000000 00000000 00000000
Word 0x00000004: 00000000 00000000 00000000 00000000
u-boot=> fuse prog 47 0 0x2
Programming bank 47 word 0x00000000 to 0x00000002...
Warning: Programming fuses is an irreversible operation!
         This may brick your system.
         Use this command only if you are sure of what you are doing!

Really perform this fuse programming? <y/N>
y
u-boot=> fuse sense 47 0 8
Sensing bank 47:

Word 0x00000000: 00000003 00000000 00000000 00000000
Word 0x00000004: 00000000 00000000 00000000 00000000
```

Note: *The fuse value to burn can be a mask value. Only the bit fields with ones are programmed.*

4.2 SM

For devices that have the SM enabled, such as i.MX 95 and i.MX 943, there is a fuse tool in the SM debug monitor.

Table 3. Fuse tool in SM debug monitor

Command	Description
fuse.r *wordIdx*	display fuse *wordIdx* (if possible)
fuse.w *wordIdx value*	write *value* to fuse *wordIdx* (if possible)

Note:

- *This command uses the fuse word index only without the bank and word concept.*
- *For read operations, it tries the FSB read, and proceeds to the ELE-AP read fuse(0x97) if the FSB read fails.*
- *For write operations, it tries the ELE-AP write fuse(0xD6) command.*

The output example on i.MX 95 is as follows:

```
>$ fuse.r 376
Fuse[376] = 0x00000000

>$ fuse.w 376 0xcafe

>$ reset
>$ fuse.r 376
Fuse[376] = 0x0000cafe
```

4.3 Linux

Linux kernel contains the **nvmem** driver for fuse access in `drivers/nvmem/`. When the driver is installed, a system node is created:

- For devices without ELE-AP: `/sys/bus/nvmem/devices/imx-ocotp0/nvmem`
- For devices with ELE-AP: `/sys/bus/nvmem/devices/fsb_s400_fuse0/nvmem`

Note: *The device node name may be changed across releases.*

The fuse can be accessed through this system node with the read/write operation. The Linux **dd** command can be used to read from or write to the node. The system calls (like **pread** and **pwrite**) are also helpful.

Note: *See the **One-Time Programmable Controller Driver Using NVMEM Subsystem** chapter in the i.MX Linux User's Guide (document UG10163) for additional information.*

4.3.1 Read

To read the fuse with **dd** for i.MX 6UL, use the following command:

```
dd if=/sys/bus/nvmem/devices/imx-ocotp0/nvmem of=fuse.bin bs=4 conv=fsync
hexdump fuse.bin
```

The output example is as follows:

```
root@imx6ul7d:~# dd if=/sys/bus/nvmem/devices/imx-ocotp0/nvmem of=fuse.bin bs=4
 conv=fsync
144+0 records in
144+0 records out
576 bytes copied, 0.0592357 s, 9.7 kB/s
root@imx6ul7d:~# hexdump fuse.bin
0000000 8003 0032 ffd7 d963 79d4 2617 0023 7032
0000010 0002 0242 0000 0000 0040 0008 0000 0000
0000020 00c0 0000 0000 0000 0014 0000 0000 0000
0000030 0000 0000 0000 0000 0d7d 5ae5 0000 0000
0000040 bada bada bada bada bada bada bada bada
*
0000060 0000 0000 0000 0000 0000 0000 0000 0000
*
0000080 0000 0000 0000 0000 4311 9f08 0004 4312
0000090 9f08 0004 bada bada 4100 0000 0000 0000
```

The example for i.MX 93 is as follows:

```
dd if=/sys/bus/nvmem/devices/fsb_s400_fuse0/nvmem of=fuse.bin bs=4 conv=fsync
hexdump fuse.bin
```

The output example is as follows:

```
root@imx93evk:~# dd if=/sys/bus/nvmem/devices/fsb_s400_fuse0/nvmem of=fuse.bin
 bs=8 conv=fsync
256+0 records in
256+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.138063 s, 14.8 kB/s
root@imx93evk:~# clear
root@imx93evk:~# hexdump fuse.bin
0000000 26d8 0000 2493 0000 0490 0000 75c6 0000
0000010 009b 0000 7e00 0000 2692 0000 0002 0000
0000020 3600 0000 3493 0000 36db 0000 0000 0000
0000030 bc7c 7aef 0000 0000 05f7 78e5 0000 0000
0000040 0000 0000 0000 0000 0001 0000 81a0 0000
0000050 0000 0000 0000 0000 0000 0000 0000 0000
*
00000c0 9705 f512 8ad0 404b 568c a05f eaef 0761
00000d0 0000 0000 0000 0000 0000 0000 0000 0000
```

The fuse offset address in the **hexdump** output is calculated as follows:

- Offset = fuse_word_index * 4
- Offset = (fuse_address – OCOTP_CTRL_OFFSET) % 0x10 * 4

For example:

- MAC1_ADDR on i.MX6UL is: offset = (0x620 – 0x400) % 0x10 * 4 = 0x88
- GPR2_CFG0 on i.MX 93 is: offset = 376 * 4 = 0x5E0

Note:

- *The **bs** value must be a multitude of four, because the fuse is accessed in word granularity.*
- *The output of **hexdump** is in little endianness.*

4.3.2 Write

To write a fuse with **dd**, make sure that you use correct offset and endianness when writing fuses. The example for i.MX 93 is as follows:

- Write 0x12345678 to bank 16 word 1:

```
echo -n -e "\x78\x56\x34\x12" > fuse.bin
dd if=fuse.bin of=/sys/bus/nvmem/devices/fsb_s400_fuse0/nvmem bs=4 seek=129
 conv=notrunc status=none
```

  Where **seek** is the fuse word index.
  seek = bank * words_per_bank + word, 129 = 16 * 8 + 1

Note:

- *Use bs = 4.*
- *For ELE-AP devices, the Linux driver must have this commit included. Otherwise, the **dd** command cannot be used.*

4.4 SCFW

For devices that have SECO, like i.MX 8XL, the SCFW on the SCU manages the fuse operations. The SCFW debug monitor has a fuse tool.

Table 4. Fuse tool in the SCFW debug monitor

Command	Description
fuse.r word	Reads OTP fuse word
fuse.w word value	Writes value to OTP fuse word

The output example on i.MX 8QXP is as follows:

```
>$ fuse.r 765
Fuse[765] = 0x00000000 lock = 0
>$ fuse.w 765 0xcafe
Confirm write 0xCAFE -> word[765] [y/n]: y
Fuse[765] <- 0x0000cafe
>$ fuse.r 765
Fuse[765] = 0x0000cafe lock = 1
```

Note:

- *This command uses the fuse word index only without the bank and word concept.*
- *The lock bit for one fuse word is in bit 16 of the fuse word index.*

4.5 SPSDK

SPSDK is a unified Python SDK library. It can connect and communicate with NXP products to perform various operations, including fuse-related operations.

Note: *SPSDK supports fuse operations only for devices with ELE-AP.*

It provides two applications for fuse operations:

- nxpele
  - Use the following subcommands:
    1. read-common-fuse
    2. read-shadow-fuse
    3. write-fuse
    4. write-shadow-fuse
- nxpfuses

These two applications interact with U-Boot using the serial port or the fast boot port and use the **ele_message** U-Boot command.

4.5.1 Requirements

SPSDK can communicate with U-Boot through the serial or fast boot interfaces. Steps 3 and 4 vary depending on the method:

1. Enable **CONFIG_AHAB_BOOT** in U-Boot. This adds the **ele_message** command in the U-Boot console.
2. Stop at the U-Boot prompt.

Either:

1. Enter the fast boot mode using **fastboot 0**.
2. Use **-d uboot_fastboot** in the SPSDK command.

Or:

1. Exit the U-Boot serial port.
2. Use **-d uboot_serial** in the SPSDK command.

4.5.2 Read

To read a fuse, you can use several commands:

A.
```
nxpele -f mimx9352 -d uboot_fastboot read-common-fuse -i 129
```
The output example is as follows:
```
(venv) PS C:\SPSDK> nxpele -f mimx9352 -d uboot_fastboot read-common-fuse -i
 129
Read common fuse ends successfully.
Fuse ID_129: 0xBE2EE7C4
```
This command uses the read common fuse API to read the fuse word 129 (bank 16, word 1).
```
nxpfuses get-config -f mimx9352 -d uboot_fastboot -o current.yaml
```
B. This command reads all the fuses from the device and parses the fuse value into a YAML file. The fuse value is mapped to the corresponding functional fuse name, for example, SRKH. The fuse name can be used for other commands using the **-n** parameter.

The result example is as follows:

```
family: mimx9352
revision: latest
registers:
  LOCK_CFG0:
    BOOT_CFG_LOCK: UNLOCKED
    HSW_CFG_LOCK: OP_WP
    reserved_fuse0-bits6-8: '0x3'
    reserved_fuse0-bits9-11: '0x3'
    LOCK_CFG_LOCK: OP
……
  SRKH: 175125E6C4E72EBE3459B2FACFF70FED4C5C12E8A4DF1039FDCE3B498D1BF45B
  OTFAD1_KEY: '00000000000000000000000000000000'
  OTFAD1_CFG4:
    OTFAD1_ENABLE: DISABLED
    OTFAD1_DISABLE_OVERRIDE: '0x0'
    OTFAD1_KEY_BLOB_EN: '0x0'
    OTFAD1_KEY_SCRAMBLE_CRC_EN: '0x0'
    OTFAD1_KEY_SCRAMBLE_EN: DISABLED
    OTFAD1_KEY_SCRAMBLE_ALIGN: '0x00'
    OTFAD1_RESTRICT_OTFAD_IPS: '0x0'
  OTFAD1_KEY_SCRAMBLE: '0x00000000'
  COM_DEVICE_ID_CFG0:
    USB_VID: '0x0000'
    USB1_PID: '0x0000'
```

C. This command prints a rich version of the SRKH fuse:

```
nxpfuses print -f mimx9352 -d uboot_fastboot -n SRKH --rich
```

-n SRKH is the fuse name captured from the option **B** output.
The output example is as follows:

```
(venv) PS C:\SPSDK> nxpfuses print -f mimx9352 -d uboot_fastboot -n SRKH --
rich
Fuse name:         SRKH
Fuse value:
 175125E6C4E72EBE3459B2FACFF70FED4C5C12E8A4DF1039FDCE3B498D1BF45B
Fuse access:       Read/Write
Fuse locks:        No locks
Fuse raw value:
 5BF41B8D493BCEFD3910DFA4E8125C4CED0FF7CFFAB25934BE2EE7C4E6255117
Fuse description: SHA256 hash digest of hash of four SRK keys
Fuse address:      0x00000000
Fuse width:        256 bits
```

Note: *The fuse value is in little endianness, while the fuse raw value is in big endianness.*

4.5.3 Write

To write a fuse, use the following commands:

A.

```
nxpele -f mimx9352 -d uboot_fastboot write-fuse -d 0x4 -i
        376
```

The output example is as follows:

```
(venv) PS C:\SPSDK> nxpele -f mimx9352 -d uboot_fastboot write-fuse -d 0x4 -i
 376
```

```
ELE write fuse ends successfully.
```

B.
```
nxpfuses write-single -f mimx9352 -d uboot_fastboot -n GPR2_CFG0 -v
          0x8
```

The output example is as follows:

```
(venv) PS C:\Users\nxg13213\SPSDK> nxpfuses write-single -f mimx9352 -d
 uboot_fastboot -n GPR2_CFG0 -v 0x8
Writing of fuses is irreversible operation. Type y/yes if you prefer to
 continue.: y
Writing value 0x8 to the fuse 'GPR2_CFG0' succeeded.
```

To write a fuse with the YAML configuration file, you may first get the current device configuration by option b in the previous section or get a configuration template. Then, customize the configuration and execute it. The configuration example is as follows:

```
family: mimx9352
revision: latest
registers:
  GPR2_CFG0: '0x00000008'
  GPR2_CFG1: '0x00000001'
```

C. Then, run the following command:

```
nxpfuses write -d uboot_fastboot -c fuse.yaml
```

The output example is as follows:

```
(venv) PS C:\SPSDK> nxpfuses write -d uboot_fastboot -c fuse.yaml
Writing of fuses is irreversible operation. Type y/yes if you prefer to
 continue.: y
Writing of fuses succeeded.
```

Note: *All of the above commands write to the i.MX 93 fuse word GPR2_CFG0 (word 376) or GPR2_CFG0 (word 377).*

5 Fuse protection

The fuse can be protected for integrity and availability purposes by several means.

5.1 Lock

The lock bits protect fuses against write, override, or read. Typical controls include the following:

- Unlocked (fuses can be read, sensed, burned, or overridden in the corresponding OTP shadow register)
- WP (Write Protect, the controlled field cannot be burned)
- OP (Over-ride Protect, the controlled field shadow registers cannot be overwritten)
- RP (Read Protect, the controlled field can be sensed only, but cannot be read from shadow registers)

The lock fuses are usually located at the start of a fuse map.

Note: *For devices with SCFW, the lock bit for one fuse word is in the bit 16 of the fuse word index. For example, write fuse word 730 with the **0xdada1212** value, and set the lock bit.*

Use the following command in U-Boot:

```
fuse write 0 66266 0xdada1212
```

Where 66266 = 730 + 65536. (bit 16 = 1).

5.2 Redundancy and ECC

Starting from i.MX 8/8X families, the ECC (word-level integrity) and redundancy (bit-level mirroring) are used by the fuse block to ensure that its values are correct.

ECC-protected words can be programmed at the word granularity only once. Redundancy regions support incremental bit programming with mirrored bits for reliability purposes.

5.2.1 Redundancy

Fuses can be set only from 0 to 1 and the most common failure is a 1 appearing as a 0. The simplest mechanism to verify these failures is redundancy. Redundancy is when two bits are OR'd together so that if either is set, the output value is a 1, where:

- Bits [15:0] are the functional fuses.
- Bits [31:16] are the redundant fuses. Bits [15:0] are automatically copied to bits [31:16].

5.2.2 ECC

The more complex ECC mechanism is used when higher numbers of bits must be stored and configured at the same time in a group of 32. In this case, when written, calculate the ECC to allow an ECC correction. When reading, any bits of failure are corrected. Two bits will be detected (and an error flag asserted) by the controller.

Note: *Because the ECC is generated for the whole word, the whole word must be set up at one time and individual bits cannot be modified independently.*

5.3 Life cycle

For devices with ELE-AP, the fuse usage policy may be changed for a different device life cycle.

- READ FUSE (0x97) - there is no life cycle restriction for this API.
- WRITE FUSE (0xD6) - OEM fuses can be written in OEM OPEN life cycles. OEM fuses can be written in the OEM CLOSE life cycle only if the request is coming from a secure privilege application.
- WRITE SECURE FUSE(0x91) - this API can be called in the OEM CLOSED life cycle. It uses a signed message to program security sensitive fuses.
- WRITE SHADOW FUSE (0xF2) - this API can be called only in OEM OPEN life cycles.
- READ SHADOW FUSE (0xF3) - this API can be called only in OEM OPEN life cycles.

6 Summary

Table 5 lists the fuse features for different i.MX devices.

Table 5. Fuse summary

Device	ECC present	Access path	Words/bank	ELE-AP	Notes
i.MX 6	No	OCOTP_CTRL	8	No	Legacy fuse block; no ECC
i.MX 7Dual/i.MX 7Solo	No	OCOTP_CTRL	4	No	Legacy fuse block; no ECC
i.MX 7ULP	No	OCOTP_CTRL	8	No	Legacy fuse block; no ECC
i.MX 8M	No	OCOTP_CTRL	4	No	Legacy fuse block; no ECC
i.MX 8/8X/8XLite	Yes	SCFW	Bank0 only	No	ECC; SCFW-controlled fuse operations
i.MX 8ULP/9	Yes	FSB + ELE MU	8	Yes	ECC + ELE-AP protection

7 Abbreviations

Table 6. Abbreviations

Abbreviation	Description
API	Application Programming Interface
BSD	Berkeley Software Distribution
ECC	Error Correcting Code
ELE-AP	EdgeLock Secure Enclave (Advanced Profile)
FSB	Fuse/Status Block
IP	Intellectual Property
MAC	Media Access Control
MU	Messaging Unit
OEM	Original Equipment Manufacturer
OTP	One-Time Programmable
OP-TEE	Open Portable Trusted Execution Environment
SECO	Security Controller
SCU	System Controller
SDK	Software Development Kit
SM	System Manager
SoC	System-on-Chip
SPSDK	Secure Provisioning Software Development Kit
SPL	Secondary Program Loader
SCFW	System Controller Firmware

8 Note about the source code in the document

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

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN

ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

9 Revision history

Table 7. Revision history

Document ID	Release date	Description
AN14954 v.1.1	6 March 2026	• Small fixes
AN14954 v.1.0	27 February 2026	• Initial release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Suitability for use in automotive applications — This NXP product has been qualified for use in automotive applications. If this product is used by customer in the development of, or for incorporation into, products or services (a) used in safety critical applications or (b) in which failure could lead to death, personal injury, or severe physical or environmental damage (such products and services hereinafter referred to as "Critical Applications"), then customer makes the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, safety, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. As such, customer assumes all risk related to use of any products in Critical Applications and NXP and its suppliers shall not be liable for any such use by customer. Accordingly, customer will indemnify and hold NXP harmless from any claims, liabilities, damages and associated costs and expenses (including attorneys' fees) that NXP may incur related to customer's incorporation of any product in a Critical Application.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Trademarks

Contents

1	**Introduction**	**2**
2	**Fuse control IP**	**2**
2.1	OCOTP_CTRL	2
2.2	Fuse status block	2
3	**Fuse map**	**2**
3.1	Devices without ELE-AP	3
3.2	Devices with ELE-AP	4
4	**Fuse usage**	**6**
4.1	U-Boot	6
4.2	SM	7
4.3	Linux	7
4.3.1	Read	8
4.3.2	Write	9
4.4	SCFW	9
4.5	SPSDK	10
4.5.1	Requirements	10
4.5.2	Read	10
4.5.3	Write	11
5	**Fuse protection**	**12**
5.1	Lock	12
5.2	Redundancy and ECC	13
5.2.1	Redundancy	13
5.2.2	ECC	13
5.3	Life cycle	13
6	**Summary**	**13**
7	**Abbreviations**	**14**
8	**Note about the source code in the document**	**14**
9	**Revision history**	**15**
	Legal information	**16**