How to update OTP with U-Boot

Contents

Stable: 16.02.2021 - 10:41 / Revision: 16.02.2021 - 10:41

A quality version of this page, approved on *16 February 2021*, was based off this revision.

This page explains how to manually update OTP with the U-Boot `fuse` command.

Contents

1 The fuse command 4
2 STM32MP15x support 5
2.1 STM32MP15x OTP 5
2.1.1 Simple OTP examples 5
2.1.2 MAC address example 6
2.2 STPMIC1 NVM 8
3 References 9

1 The fuse command

Programming fuses is an irreversible operation!
This may brick your system.
Use this command only if you are sure of what you are doing!

The `fuse` command allows you to update the OTP words in U-Boot:

- sense/program to directly access the OTP value (for a permanent update)
- read/overidde to access only the shadow cache value (for a temporary update).

```
Board $> help fuse
fuse - Fuse sub-system

Usage:
fuse read <bank> <word> [<cnt>] - read 1 or 'cnt' fuse words,
    starting at 'word'
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'.
```

See doc/README.fuse for details.

2 STM32MP15x support

The STM32MP15x lines support is implemented in drivers/misc/stm32mp_fuse.c with 2 banks:

- <bank> = SOC OTP: 0 #STM32MP15x OTP
- <bank> = PMIC NVM : 1 #STPMIC1 NVM.

2.1 STM32MP15x OTP

<bank> = 0 provides access to the 96 STM32MP15x OTP words with the BSEC driver: arch/arm/mach-stm32mp/bsec.c .

Refer to the STM32MP15 reference manuals for the OTP layout.

The OTP<index> words value and lock status (0=unlocked, 1=locked) are available with <index> = 0 to 95 :

- for value: <word> = <index>
- for lock status: <word> = 0x10000000 + <index>

not shadowed, only support operation 'sense' and 'program'.

Only 32 lower OTPs words are accessible by default, the software needs to manage exceptions to allow some upper OTPs to be accessed by the non-secure world as described in BSEC_device_tree_configuration.

2.1.1 Simple OTP examples

1) Read OTP value for OTP57 and 58 (2 OTP words)

```
Board $> fuse sense 0 57 2
Sensing bank 0:

Word 0x00000039: 42e18000 0000e448
```

2) Check lock status of OTP 57 - 60 (4 words at index 57 = 0x39)

```
Board $>  fuse sense 0 0x10000039 4
Sensing bank 0:

Word 0x10000039: 00000001 00000001 00000001 00000000
```

3) Display shadow values for all OTPs

When only the 32 lower OTPs are accessible:

```
Board $> fuse read 0 0 32
Reading bank 0:

Word 0x00000000: 00000017 00008001 00000000 00000000
Word 0x00000004: 00000000 00000000 00000000 00000000
Word 0x00000008: 00000000 82004000 00000000 00000000
Word 0x0000000c: 7d04f0db 00470022 33385115 34383330
Word 0x00000010: 22986562 27010551 7a470140 06cc1608
Word 0x00000014: 5e560054 00000000 00000000 401a300c
Word 0x00000018: ffffffff ffffffff ffffffff ffffffff
Word 0x0000001c: ffffffff ffffffff ffffffff ffffffff
```

When all the 96 OTPs are available:

```
Board $> fuse read 0 0 96
Reading bank 0:

Word 0x00000000: 00000017 00008000 00000000 00000000
Word 0x00000004: 00000000 00000000 00000000 00000000
Word 0x00000008: 00000000 00000000 00000000 00000000
Word 0x0000000c: 7cf5f0f9 00410032 33385116 34383330
Word 0x00000010: 129675aa 2931215e 7a550000 069013ec
Word 0x00000014: 5e360042 00000000 00000000 40133023
Word 0x00000018: 00000000 00000000 00000000 00000000
Word 0x0000001c: 00000000 00000000 00000000 00000000
Word 0x00000020: 00000000 00000000 00000000 00000000
Word 0x00000024: 00000000 00000000 00000000 00000000
Word 0x00000028: aa333e40 b5e90dda f15f4678 8ab41400
Word 0x0000002c: b74efe3a f0a03b1b 01e016b3 d06a79dd
Word 0x00000030: 48b96fbe 20fbd352 6732dbf4 edc395f9
Word 0x00000034: cdf15575 418fd3d0 0bb7d994 8dc929d0
Word 0x00000038: 00000000 42e18000 0000e448 12722301
Word 0x0000003c: 00000000 00000000 00000000 00000000
Word 0x00000040: 00000000 00000000 00000000 00000000
Word 0x00000044: 00000000 00000000 00000000 00000000
Word 0x00000048: 00000000 00000000 00000000 00000000
Word 0x0000004c: 00000000 00000000 00000000 00000000
Word 0x00000050: 00000000 00000000 00000000 00000000
Word 0x00000054: 00000000 00000000 00000000 00000000
Word 0x00000058: 00000000 00000000 00000000 00000000
Word 0x0000005c: 00000000 00000000 00000000 00000000
```

4) Override value for one OTP

```
Board $> fuse override 0 0x0000005c 1
Overriding bank 0 word 0x0000005c with 0x00000001...
```

```
Board $> fuse read 0 0x0000005c
Reading bank 0:

Word 0x0000005c: 00000001

Board $> fuse sense 0 0x0000005c
Sensing bank 0:

Word 0x0000005c: 00000000
```

2.1.2 MAC address example

For STM32MP15_boards, the MAC address[1] is retrieved in the 2 OTP words:

- OTP_57[31:0] = MAC_ADDR[31:0]
- OTP_58[15:0] = MAC_ADDR[47:32]

To program a MAC address on virgin OTP words above, you can use the `fuse` command on bank 0 to access internal OTP words and lock them:

Prerequisite: check if a MAC address isn't yet programmed in OTP.

1) Check OTPs: their value must be equal to 0:

```
Board $> fuse sense 0 57 2
Sensing bank 0:
Word 0x00000039: 00000000 00000000
```

2) Check environment variable:

```
Board $> env print ethaddr
## Error: "ethaddr" not defined
```

3) Check lock status of OTP 57 & 58 (at 0x39, 0=unlocked, 1=locked):

```
Board $> fuse sense 0 0x10000039 2
Sensing bank 0:
   Word 0x10000039: 00000000 00000000
```

Example to set MAC address "12:34:56:78:9a:bc"

1) Write OTP:

```
Board $> fuse prog -y 0 57 0x78563412 0x0000bc9a
```

This prog command cannot be executed twice because the values of these 2 upper OTP words are ECC-protected.
To avoid an issue for a second operation these OTPs must be locked when programmed.

2) Read OTP:

```
Board $> fuse sense 0 57 2
Sensing bank 0:
Word 0x00000039: 78563412 0000bc9a
```

3) Lock OTP:

```
Board $> fuse prog 0 0x10000039 1 1

Board $> fuse sense 0 0x10000039 2
Sensing bank 0:
   Word 0x10000039: 00000001 00000001
```

4) OTP is used after REBOOT, in the trace:

```
### Setting environment from OTP MAC address = "12:34:56:78:9a:bc"
```

5) Check env update:

```
Board $> env print ethaddr
ethaddr=12:34:56:78:9a:bc
```

2.2 STPMIC1 NVM

<bank> = 1 provides access to the non-volatile memory (NVM) of the PMIC on the board.

For STPMIC1, the NVM has 8 bytes as defined in datasheet: DS12792, with <word> = <0xf8> to <0xff>.

1) Read the values of the 8 NVM

```
Board $> fuse read 1 0xf8 8
Reading bank 1:

Word 0x000000f8: 000000ee 00000092 000000c0 00000002
Word 0x000000fc: 000000f2 00000080 00000002 00000033
```

2) Read 2 NVM shadow values

```
Board $> fuse sense 1 0xf9 2
Sensing bank 1:

Word 0x000000f9: 00000092 000000c0
```

3) Update the NVM at index 0xfc with value 0xf2

```
Board $> fuse prog 1 0xfc 0xf2
Programming bank 1 word 0x000000fc to 0x000000f2...
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
```

3 References

- https://en.wikipedia.org/wiki/MAC_address

One Time Programmed

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Power Management Integrated Circuit

Non Volatile Memory, like a flash memory

Boot and Security and OTP control

media access control address (https://en.wikipedia.org/wiki/MAC_address)

Elliptic curve cryptography

Error Correction Capability