GREENWAVES TECHNOLOGIES

Application Note
AN004

GAP8 E-FUSE SETTINGS FOR RELIABLE BOOT

Version : Rel.1.2
Date : Feb., 2021

---

1. INTRODUCTION

By default, GAP8 expects code to be downloaded into its internal RAM through its JTAG interface and will boot from there.
Whilst this is convenient for initial board bring-up, in a product GAP8 would normally boot from another source.
Three main alternative boot mechanisms are offered with GAP8 :
- boot from an external HyperFlash connected to GAP8's HyperBus interface
- boot from an external SPI Flash connected to GAP8's SPIM0 interface (this would typically be a Quad-SPI capable Flash to get decent throughput)
- boot from SPI Slave interface (this is an unsecured boot, although it may be handy in some cases – more details later).

Upon power-up, a primary boot code located in ROM is executed. This code performs a number of initial checks and configurations and then gets a secondary boot code (start of user's application) from the selected external source, copies it into the internal L2 RAM and executes it.

Selection of the source for booting is done through the programming of a specific e-fuse. In addition, a minimal set of additional e-fuses must be programmed for reliable boot, they control the behavior of some important checks and configurations performed by the primary boot code.

Programming an e-fuse bit consists of burning the associated fuse; once burnt the e-fuse is permanently set to the selected value and cannot be modified anymore. Fuse programming typically occurs at board production or commissioning. Some ‘user” fuses are also available to be used freely by the application.

GAP8’s SDK includes a fuse programming utility that enables the user to specify an array of fuses to be burnt.

The next sections in this document detail what fuses should be programmed for proper boot.
Reference is made to e-fuse bits listed in a map provided as Appendix. In the interest of clarity, only those e-fuses useful to customers are made explicit. Others are marked “Reserved”; they typically correspond to options reserved for alternative configurations of GAP8 or for future use. The table will be updated if some of those reserved fuses actually need to be activated in the future.

IMPORTANT NOTES:

1) A blank (non-programmed) e-fuse bit returns a Logic-0 when read. **Programming, i.e. burning, an e-fuse bit permanently sets it to Logic-1. There is then no means to set it back to Logic-0**. Therefore, if an 8-bit fuse register is programmed to, say, 0x01 (so bit0 is set), then it will not be possible to later program it to e.g. 0x02 (which would require to set bit1 and clear bit0) -- the setting of bit1 will be **incremental**, bit0 will stay set and the valid programming value will be 0x03.
2) To burn e-fuses, GAP8 requires a 2.5V+-10% voltage to be provided on pin VQPS (pin B28). In general (i.e., except for cases where GAP8 e-fuses would have been pre-programmed), the application board must provision the means to provide this voltage, either from some on-board power supply or from an external source through a connector.

2. E-FUSE PROGRAMMING SEQUENCE

- a - Specify boot source :

> Program **bits 3-5** (only those bits!) of fuse **INFO1** (ID #0) to specify Flash or SPI Slave source :
0x2 for Flash
0x3 for SPI Slave

> If booting from Flash, specify SPI or HyperBus interface through **bit 0** (only this bit!) of fuse **INFO3** (ID #37)
‘0’ for [Quad-]SPI
‘1’ for Hyperbus

- b - Configure Oscillator Convergence Checks

Case 1: with GAP8 Rev.B (early lot produced before year 2020)
> Program **bit 1** (only this bit!) of fuse **INFO2** (ID #1) to **'1'** (configure_fll)
> Program fuse **FLL_ASSERT_CYCLES** (ID #33) to **0x1F** (31 decimal)

Case 2: with GAP8 Rev.C (chips produced from year 2020 onwards)
> Program fuse **INFO2** (ID #1) to **0x40**
> Program fuse **XTAL_MAX** (ID #30,31) to **0xFF,0xFF**
> Program fuse **XTAL_MIN** to **0x0010** -- i.e. 0x10 into Fuse ID #28
> Program fuse **XTAL_DELTA** to **0x4000** -- i.e. 0x40 into Fuse ID #27
> Program **bit 7** (only this bit!) of fuse **INFO1** (ID #0) to **'1'**
Beware: the last setting indicated must be performed last – i.e., do **not** set bit 7 of INFO1 (fuse ID#0) before setting the other fuses

- c - [Optional] Boot source locking mechanism

By default, even after specifying an alternative boot source it remains possible to boot from JTAG. This is convenient for board bring-up and debug, but is often not acceptable in a product, as this means the complete memory map of GAP8 is accessible from outside GAP8 – which in many cases would be a serious security concern.
It is possible to permanently disable boot from JTAG and SPI Slave. To do so :
> Program **bit 5** (only this bit!) of fuse **INFO2** (ID #1) to **'1'**

Note setting this bit disables both boot from JTAG and boot from SPI Slave. This entails that this fuse cannot be used in an application that would need to boot from SPI Slave interface. As a result, boot from SPI Slave is highly unsecured and should be implemented only with the full knowledge of those implications. The full memory space of GAP8 is then visible to any agent able to tap the SPI slave interface.

- d – [Optional] Encrypted code

It is possible to store code in AES-encrypted form in the external Flash. Specific e-fuses can be used to store encryption keys. Please contact GreenWaves technologies for further information, should you require this feature.

APPENDIX I – GAP8 E-FUSE MAP

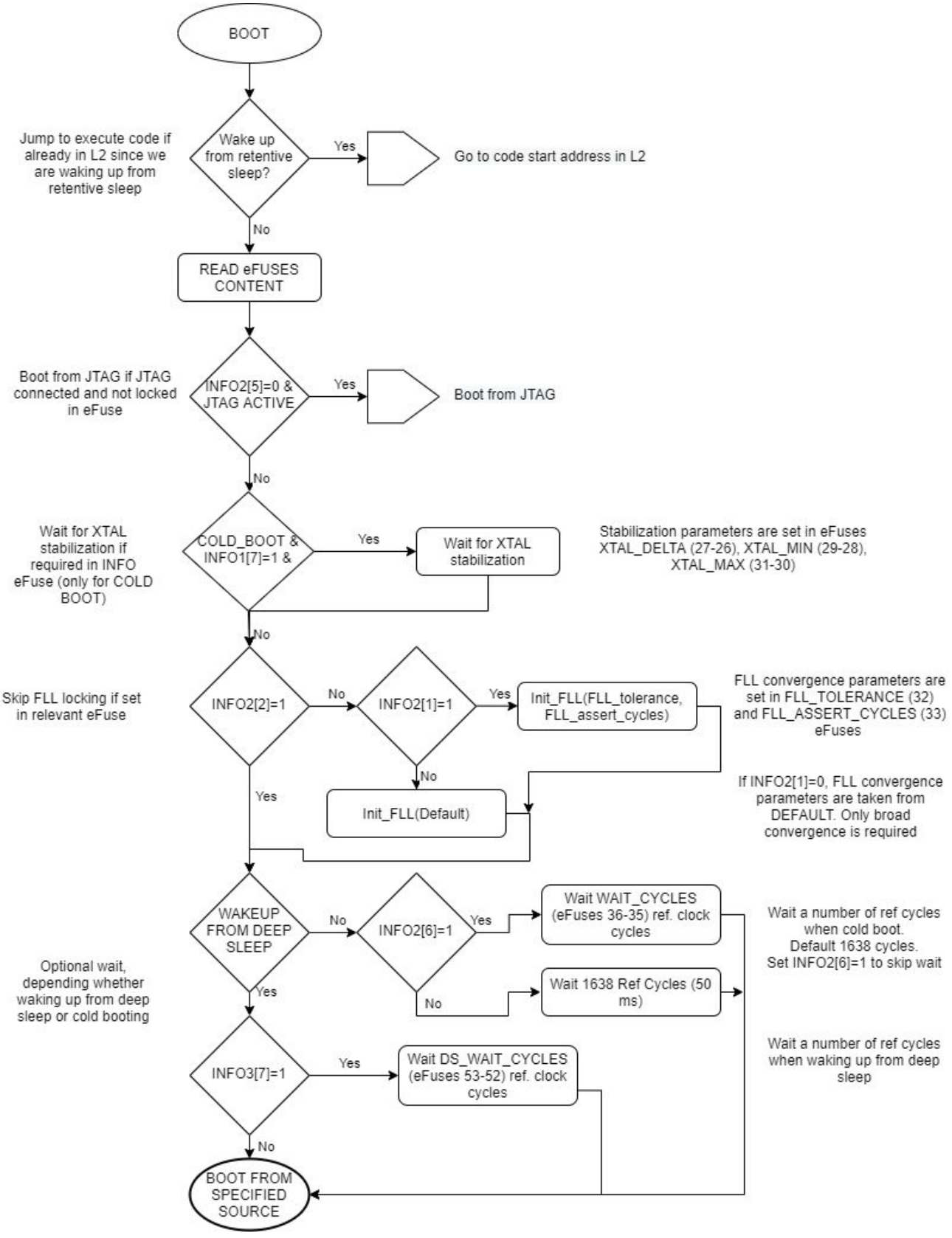
ID	NAME	Bits	Bits description
0	**INFO1**		
	GAP8 REV.B ONLY	7	Reserved e-fuse bit, Do not modify
	GAP8 REV.C ONLY	7	Check crystal oscillator stabilization
		6	Reserved e-fuse bit, Do not modify
		5-3	Boot mode (0: jtag mode, 1: stop command, 2:flash, 3: spis, 4: wait command, 5: wait end command)
		2-0	Reserved e-fuse bits, Do not modify
1	**INFO2**		
		7	Reserved e-fuse bits, Do not modify
		6	Ref_clock_wait (at cold boot only, ie. after power up) If '1' (enabled), the boot code will wait for a programmable number of ref clock cycles, specified in efuse WAIT_CYCLES (ID #35-36), before fetching application code from external Flash. If '0' (disabled) then the boot code will first wait for exactly 1638 ref clock cycles -- which at 32.768KHz is 50ms.
		5	JTAG and SPI Slave Lock (1=enabled i.e. jtag & spis are not accessible anymore)
		4-3	SPIM clock divider
		2	Bypass FLL lock (1=enabled, i.e. FLL is configuring without being locked).
		1	Configure FLL (1=enabled) according to Reg.32 and 33 for GAP8 Rev.B, Reg.58 and 59 for GAP8 Rev.C.
		0	Reserved e-fuse bit, Do not modify
2-17	**RESERVED**		Reserved e-fuse bits, Do not modify
18-25	**RESERVED**		Reserved e-fuse bits, Do not modify
GAP8 REV.B ONLY			
27-26	**RESERVED**		Reserved e-fuse bits, Do not modify
28	**RESERVED**		Reserved e-fuse bits, Do not modify
29	**RESERVED**		Reserved e-fuse bits, Do not modify
30	**RESERVED**		Reserved e-fuse bits, Do not modify
31	**FLL FREQ**		Reserved e-fuse bits, Do not modify

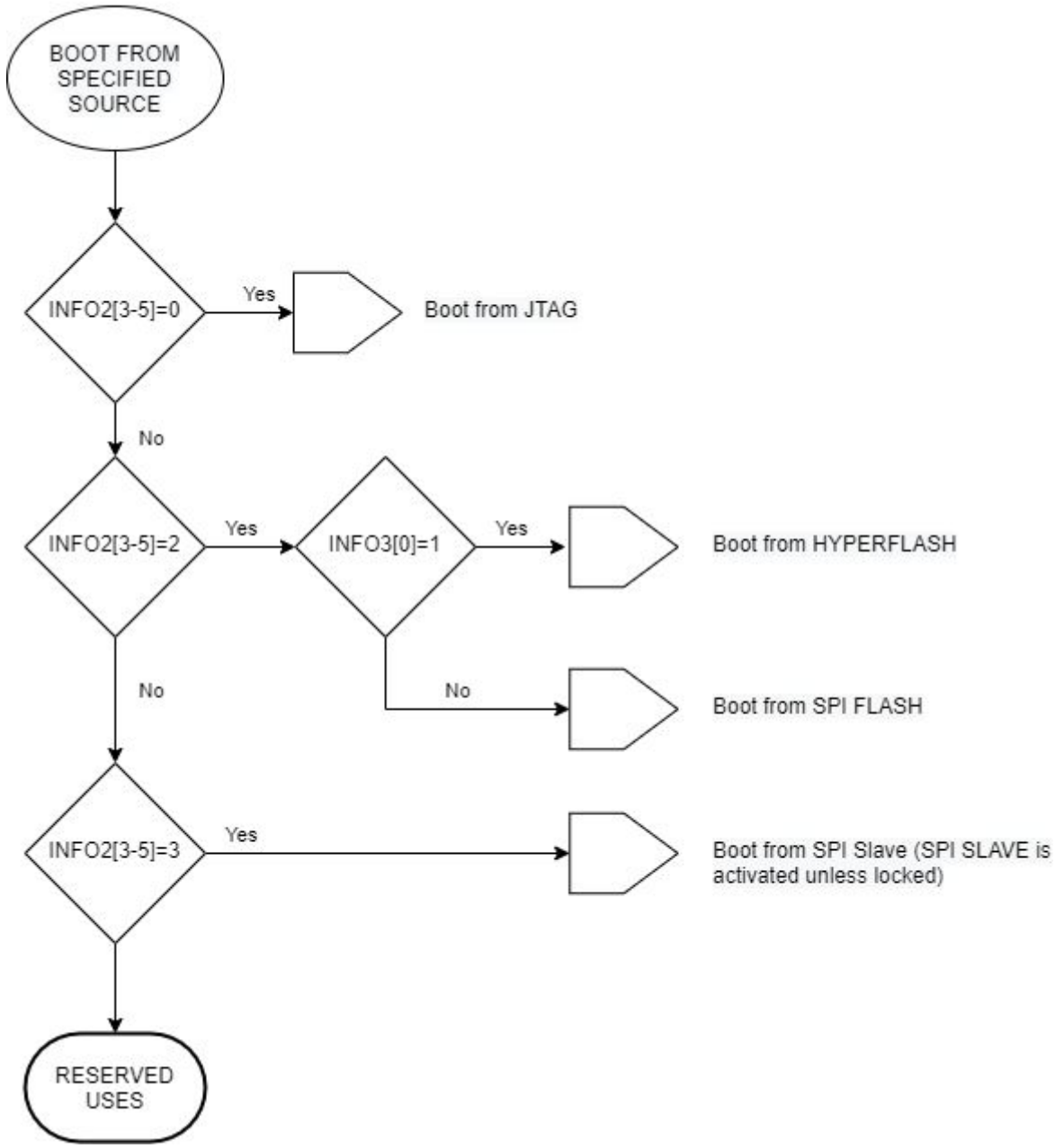
32	**FLL TOLERANCE**	7-0	Defines the excursion allowed on FLL DCO code while the 32KHz oscillator is converging. The smaller the value, the tighter the tolerance.
33	**FLL ASSERT CYCLES**	7-0	Minimum number of reference clock cycles (32KHz nom.) during which the FLL frequency (50MHz nom.) must stay within the limits defined by FLL_TOLERANCE for convergence check to be deemed successful.
GAP8 REV.C ONLY			
27-26	**XTAL DELTA**	15-0	Maximum allowed deviation in number of FLL clock cycles between two consecutive reference clock edges (used in conjunction with XTAL_MIN below). Is a percentage expressed in Q15 fractional format.
29-28	**XTAL MIN**	15-0	Minimum number of reference clock cycles (32KHz nom.) during which the oscillator frequency must stay within the limits defined by XTAL_DELTA for convergence check to be deemed successful.
31-30	**XTAL MAX**	7-0	Maximum number of reference cycles (32KHz nom.) during which convergence is checked – after that procedure is aborted and boot will fail.
32	**RESERVED**		Reserved e-fuse bits, Do not modify
34	**RESERVED**		Reserved e-fuse bits, Do not modify
35-36	**WAIT CYCLES**	15-0	Added latency (in number of ref clock cycles, 32KHz nom.) before going on with the boot procedure, taken into account if bit ref_clock_wait of INFO2 is set. Done before accessing external Flash and only at cold boot.
37	**INFO3**		
		7	Ref_clock_wait (at deep sleep wake up only) If '1' (enabled), the boot code will wait for a programmable number of ref clock cycles, specified in efuse WAIT_CYCLES (ID #52-53), before fetching application code from external Flash
		6	Flash reset wait. Wait for a certain number of ref clock edges (specified in efuse FLASH_RESET_WAIT @0x50) after the flash has been resetted.
		5	Flash init. Issue a few specific flash commands to initialize it. Only on hyperflash, this issues commands 0xAA, 0x55, 0x38, 0x8e0b
		4	Flash wakeup. Issue a flash wake-up command during flash setup.
		3	Flash wait. Once the flash setup is done, wait for a certain number of ref clock edges (specified in efuse FLASH_WAIT @0x45).

		2	Flash reset. Issue a flash reset before using it.
		1	Set SPIM clock divider (taken from efuse INFO2 bits 3 and 4). If not set the divider is 2.
		0	Flash type (0=[Quad-]SPI, 1=HyperBus)
38	**INFO4**		Reserved e-fuse bits, Do not modify
39	**INFO5**		
		7	Hyper latency. Hyper latency in the hyper controller.
		6	Hyper delay. Hyper delay in the hyper controller.
		5	Hyperchip size. Give the 4th byte of the offset to get the start address of the chip connected to chip select 1.
		4-3	Pad config. Pad configuration for the flash,spis.
		2-1	Flash_itf. Chip interface ID for the flash.
		0	Flash chip-select. For SPI, 0=CS0, 1=CS1. For hyperflash, 1 means hyperflash is on cs 0, 0 for the reverse.
40	**INFO6**		Reserved e-fuse bits, Do not modify
41	**RESERVED**		Reserved e-fuse bits, Do not modify
42	**RESERVED**		Reserved e-fuse bits, Do not modify
43	**RESERVED**		Reserved e-fuse bits, Do not modify
44	**RESERVED**		Reserved e-fuse bits, Do not modify
45	**FLASH_WAIT**	7-0	Added waiting time after the external Flash has been configured, expressed as number of reference clock cycles (32KHz nom.) -- enabled by bit 3 of INFO3 @0x37
46	**HYPERCHIP_SIZE**	7-0	Size in bytes of the chip connected to hyper chip select 0
48-47	**RESERVED**		Reserved e-fuse bits, Do not modify
49	**RESERVED**		Reserved e-fuse bits, Do not modify
50	**FLASH_RESET_WAIT**	7-0	Added waiting time after the external Flash has been reset, expressed as number of reference clock cycles (32KHz nom.) -- meaningful only if bits 2 and 6 of INFO3 @0x37 are set
51	**HYPER_LATENCY**		Reserved e-fuse bits, Do not modify
53-52	**WAIT CYCLES**		Added latency (in number of ref clock cycles, 32KHz nom.) before going on with the boot procedure, taken into account if bit ref_clock_wait of INFO2 is set. Done before accessing external Flash and only at wake up from deep sleep.
55-54	**FLASH ID**	7-0	Expected flash ID when checking it.

GAP8 REV.B ONLY			
56-127	**USER_FUSES**		“User” e-Fuse bits. Freely available for private use by the final application.
GAP8 REV.C ONLY			
57	**RESERVED**		Reserved e-fuse bits, Do not modify
58	**FLL TOLERANCE**	7-0	Defines the excursion allowed on FLL DCO code while the 32KHz oscillator is converging. The smaller the value, the tighter the tolerance.
59	**FLL ASSERT CYCLES**	7-0	Minimum number of reference clock cycles (32KHz nom.) during which the FLL frequency (50MHz nom.) must stay within the limits defined by FLL_TOLERANCE for convergence check to be deemed successful.
60	**INFO7**		Reserved e-fuse bits, Do not modify
62-61	**BURST_SIZE**		Hyper burst size (default is 1024).
63	**MODE_GPIO**		Reserved e-fuse bits, Do not modify
64	**MODE_PAD**		Reserved e-fuse bits, Do not modify
65-127	**USER_FUSES**		“User” e-Fuse bits. Freely available for private use by the final application.

APPENDIX II – GAP8 REV.C BOOT SEQUENCE AND eFUSE ACCESS





DOCUMENT HISTORY

v1.0 (XC) - June 2020
Creation

v1.1 (XC) - Feb.2021
Clarified behaviour of ref_clock_wait in INFO2 and relationship with WAIT_CYCLES

v1.2 (MG) - Feb.2021
Added description of eFuses 53,52. Added Appendix II