AN0004: Clock Management Unit

This application note gives an overview of the CMU module with explanations on how to choose clock sources, prescaling, and clock calibration.

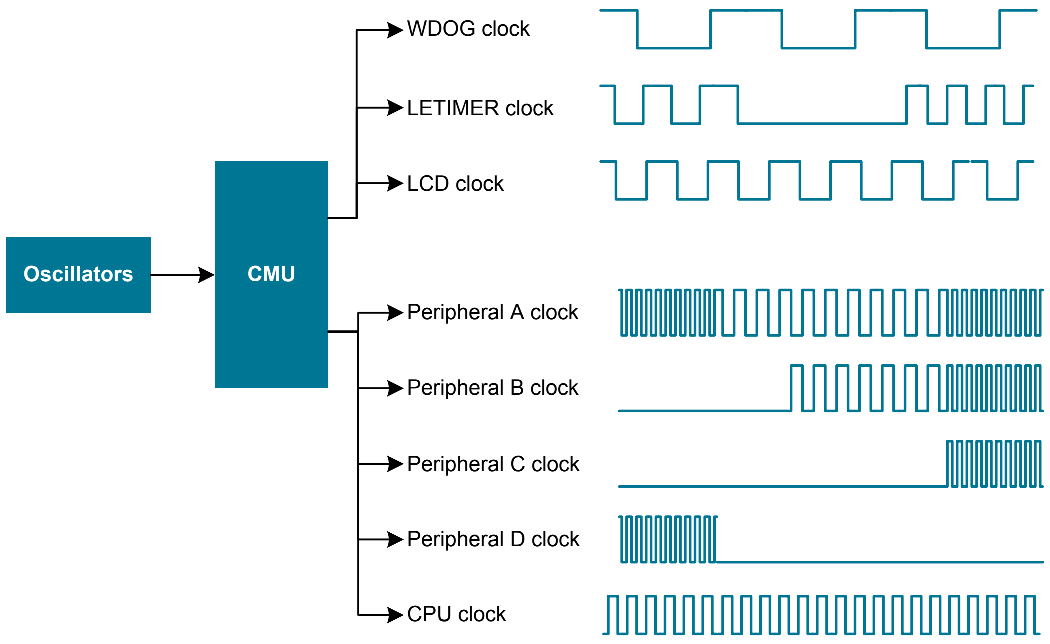
It also contains information about how to handle oscillators on wake up, external clock sources, and RC oscillator calibration.

This application note includes:

- This PDF document
- Source files (zip)
  - Example C-code
  - Multiple IDE projects

KEY POINTS

- EFM32 devices have several internal clock sources available, ranging from 32 kHz up to 28 MHz.
- EFM32 devices can also use external high-frequency and low-frequency clock sources.
- Selecting the right clock source is key for creating low energy applications.



1. Overview

The Clock Management Unit (CMU) on the EFM32 or EZR32 controls the oscillators and clocks. It can enable or disable the clock to the different peripherals individually, as well as enable, disable, or configure the available oscillators. This allows for minimizing energy consumption by disabling the clock for unused peripherals or having them run at lower frequencies.

The EFM32 or EZR32 has 3 main clock branches (HFCLK, LFACLK, and LFBCLK) which are then routed to the different peripherals. Some peripherals have dedicated prescalers such as the Low Energy peripherals. Other peripherals need to be prescaled by prescaling the clock source, which will affect all the peripherals driven by the same source.

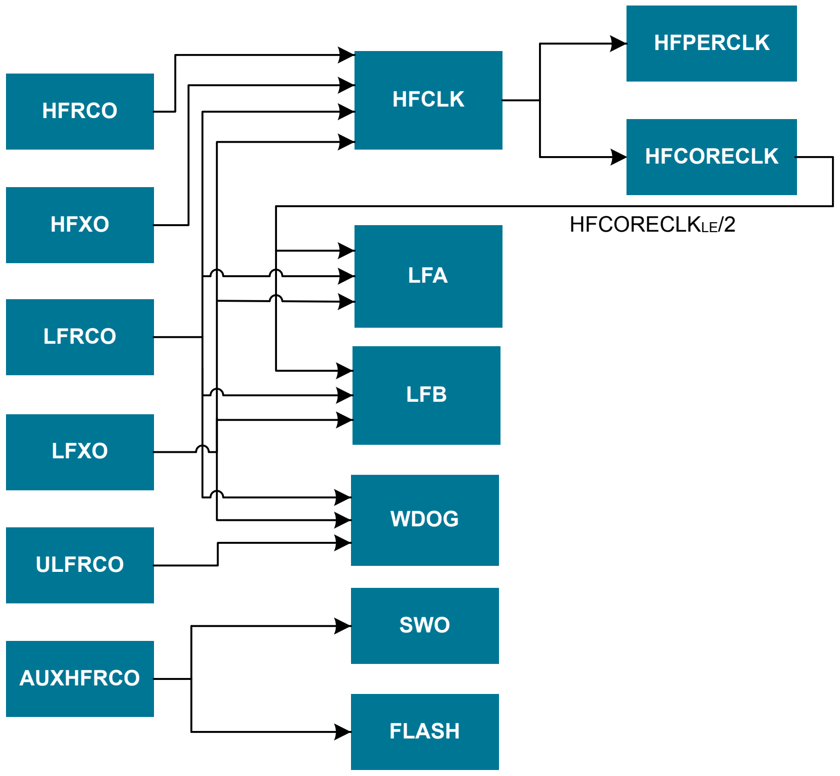


Figure 1.1. Clocks and Oscillators Overview

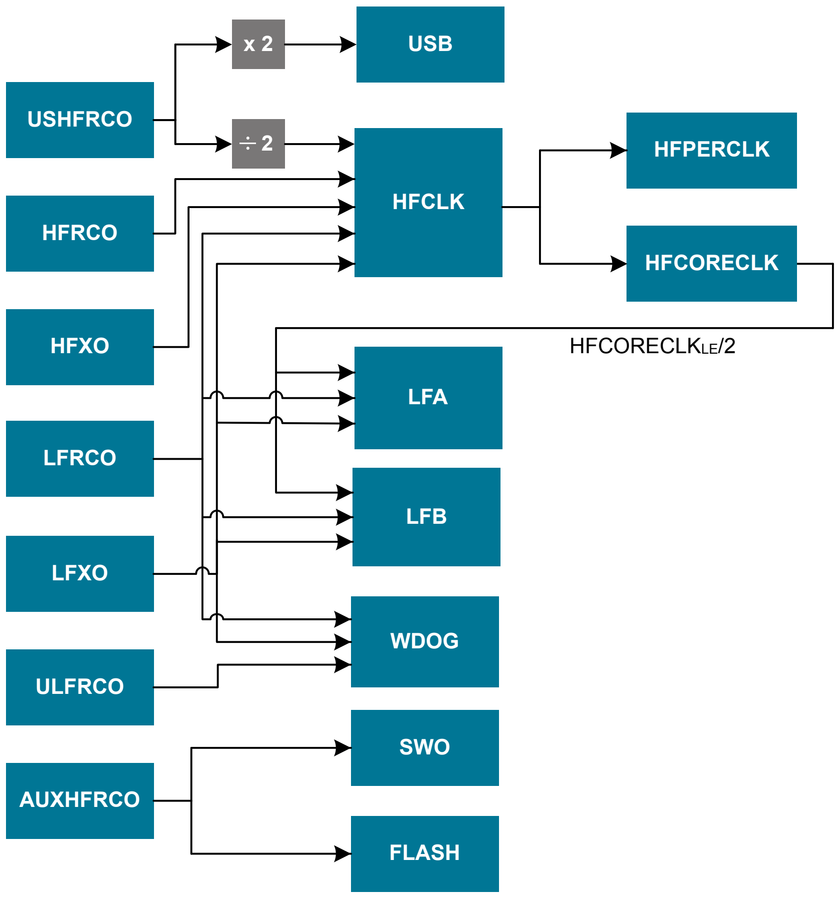


Figure 1.2. Clocks and Oscillators Overview (Happy Gecko)

There are 5 selectable clock sources:

- 1-28 MHz High Frequency RC Oscillator (HFRCO)
- 4-32 MHz High Frequency Crystal Oscillator (HFXO)
- 32.768 kHz Low Frequency RC Oscillator (LFRCO)
- 32.768 kHz Low Frequency Crystal Oscillator (LFXO)
- (Happy Gecko only) 48/24 MHz Universal Serial High Frequency RC Oscillator (USHFRCO)

The HFCLK can use any of these sources while the LFACLK and LFBCLK can only be sourced from the Low Frequency Oscillators or the $HFCORECLK_{LE}$/2, which is a prescaled version of the HFCLK for the Cortex-M3 Core and related modules (HFCORECLK), such as DMA or MSC (Memory System Controller).

In addition to these clock sources, there is a 14 MHz RC Oscillator (AUXCLK) which is used for flash programming and SWO debug output, and a 1 kHz RC Oscillator (ULFRCO) which can be selected as clock source only for the Watchdog Timer (WDOG) only.

For a detailed diagram of the EFM32 or EZR32 clock tree, refer to the reference manual.

1.1 Clock Sources

To select the clock source for a branch (HFCLK, LFA or LFB), the chosen oscillator must be enabled before the switch is done. If this is not done, the modules that are running from that clock branch will stop. In the case of selecting a disabled oscillator for the HFCLK branch, the CPU will stop and can only be recovered after a reset.

To enable or disable an oscillator, the following function from the emlib can be used:

```
CMU_OscillatorEnable(CMU_Osc_TypeDef osc, bool enable, bool wait)
```

With this function, the user can select which oscillator to enable or disable and if it should wait for the oscillator to stabilize before returning. For the HFXO and LFXO the timeout period can be configured in the CMU_CTRL register. Once enabled, the selected oscillator can be used as a clock source for one of the branches using the following function:

```
CMU_ClockSelectSet(CMU_Clock_TypeDef clock, CMU_Select_TypeDef ref)
```

This function also enables the chosen clock source in case it has not been enabled yet. The clock parameter can be one of the three clock branches HF, LFA, or LFB, and the reference can be the HFRCO, HFXO, LFRCO, LFXO, or $\text{HFCORECLK}_{\text{LE}}$/2 for the LFA and LFB branches only.

After a reset, the HF branch is clocked by the HFRCO at 14 MHz and both low frequency branches have the LFRCO selected as clock source. The LFRCO is disabled upon reset so it needs to be enabled before using the low frequency peripherals.

1.2 Prescaling

Deriving from the HF clock branch there are two sub-branches, HFPERCLK and HFCORECLK that can be prescaled individually up to a factor of 512. The HFPERCLK drives the high-frequency peripherals and the HFCORECLK drives the Core Modules which consist of the CPU and modules tightly coupled to it such as the DMA or MSC. Some peripherals allow for even further clock prescaling (such as the ADC) which is controlled by the peripheral's registers. As for the low frequency branches (LFB and LFA) all the peripherals driven from these can be prescaled individually with the exception of the Pulse Counter (PCNT). The prescaling function is the following:

```
CMU_ClockDivSet(CMU_Clock_TypeDef clock, CMU_ClkDiv_TypeDef div)
```

When using this function some attention should be made to both parameters. Not all clocks have a prescaler and the maximum prescaling value is also not the same for the different clocks (for instance the HFPERCLK has a maximum of 512 but the LETIMER clock can be divided by 32768).

1.3 HFRCO Band Selection

By default, the HFRCO band is set to 14 MHz. It can be changed to 1, 7, 11, 14, 21 or 28 MHz using the BAND bitfield in the CMU_HFRCOCTRL register. The tuning value for each band is set using the TUNING bitfield in the same register. Each band is calibrated during production, and suitable tuning values for each band can be read from the Device Information page. To make this task easier, there is also a function available to change the HFRCO band:

```
CMU_HFRCOBandSet(CMU_HFRCOBand_TypeDef band)
```

This function handles the band setting with the correct tuning value.
Note: Happy Gecko and Zero Gecko devices do not support the 28 MHz band. The highest band available on these devices is 21 MHz.

1.4 Wait States

When changing the core clock frequency (HFCORECLK) to a value above 16 MHz, the number of flash-access wait states must be set to 1 before the frequency switching. This is handled automatically by the CMU functions in the emlib, but if the user wants to make the change by writing directly into the registers, the number of wait states is modified using the MODE bitfield in the MSC_READCTRL register.

1.5 Output Clock to Pin

It is possible to configure the CMU to output clocks on two pins. This clock selection is done using CLKOUTSEL0 and CLKOUTSEL1 fields in CMU_CTRL. The pin location and enabling is configured in the CMU_ROUTE register. It is also necessary to configure the pins as outputs. Please refer to AN0012 GPIO for information on pin configuration.

- LFRCO or LFXO can be output on one pin (CMU_OUT1 on the Gecko STK)
- HFRCO, HFXO, HFCLK/2, HFCLK/4, HFCLK/8, HFCLK/16 or ULFRCO can be output on another pin (CMU_OUT0 on the Gecko STK)

Note that HFXO and HFRCO clock outputs to pin can be unstable after startup and should not be output on a pin before HFXORDY/HFRCORDY is set high in CMU_STATUS.

1.6 Software Example

This document includes a software example that demonstrates how to enable and select clock sources and also do prescaling and changing HFRCO bands. In these examples, a timer is constantly running and, on each overflow, it toggles the LEDs on the STK and all user LEDs on the DK. The timer is clocked in 4 different ways that change in a constant loop after every 5 LED toggles.

- The program starts with the HF branch being clocked by the HFRCO at 14 MHz but prescaling it by 2 for the HFPERCLK, which drives the timer. At this point the timer is clocked at 7 MHz.
- Then the prescaler is removed and the timer is clocked at 14 MHz.
- Next the HFRCO band is changed to 21 MHz and the HF clock source is changed from the HFRCO to the HFXO running at 32 MHz. For Happy Gecko devices, the example switches from the HFRCO to the USHFRCO.
  Note: The maximum clock speed for Zero Gecko devices is 24 MHz, and the maximum clock speed for Happy Gecko devices is 25 MHz.

2. Energy Modes

2.1 Active Oscillators

The energy mode of the EFM32 or EZR32 determines which oscillators are active. In EM0 and EM1, all the oscillators can be enabled and used as clock sources. Going down to EM2 causes the High Frequency Oscillators (HFXO, USHFRCO, and HFRCO) and AUXHFRCO to shut off automatically so the high frequency peripherals that run from them are no longer available. In EM3 the Low Frequency Oscillators will also stop, disabling the low frequency peripherals. The ULFRCO is always running from EM0 down to EM3 and is only disabled when the EFM32 or EZR32 goes to EM4.

2.2 Wake Up Considerations

2.2.1 Waking up from Energy Modes

All the oscillators are able to run in EM1, so the core wakes up instantly from this Energy Mode. In EM2 and EM3, the High Frequency Oscillators are disabled and need to be enabled before the core starts running code. When waking up from EM2 or EM3, the core will run from the HFRCO by default, regardless of which oscillator it was running from before entering these energy modes. The HFRCO has a very short wake-up time and it takes only 2 µs before the CPU starts running code. The previously configured HFRCO band is also restored by hardware on wake up. The wake time from EM4 is 163 µs and the core will run the HFRCO at 14 MHz.

2.2.2 Restoring the Oscillator

To use a different oscillator after wake up, the user must either enable and select the oscillator manually or use the energy mode functions from emlib, which can handle it automatically. If an oscillator (HFXO for instance) is manually selected as a clock source before the oscillator is stable, the system's behavior is undefined. The functions that send the EFM32 or EZR32 to EM2 and EM3 (see figure below) have a boolean parameter which indicates if the oscillator should be restored or not. If chosen not to be restored and the user wants to switch oscillator upon wake up, the oscillator will have to be selected manually. If chosen to be restored, the functions will handle the oscillator switching but the wake time will be longer than 2 µs. The oscillators running prior to going to EM2/3 will be enabled and the function will wait for the ready flag to be set before selecting it as a clock source. The ready flag is set when the start-up time is exceeded, which can be configured by the user in the CMU_CTRL register. The HFRCO will also be disabled after switching the clock source.

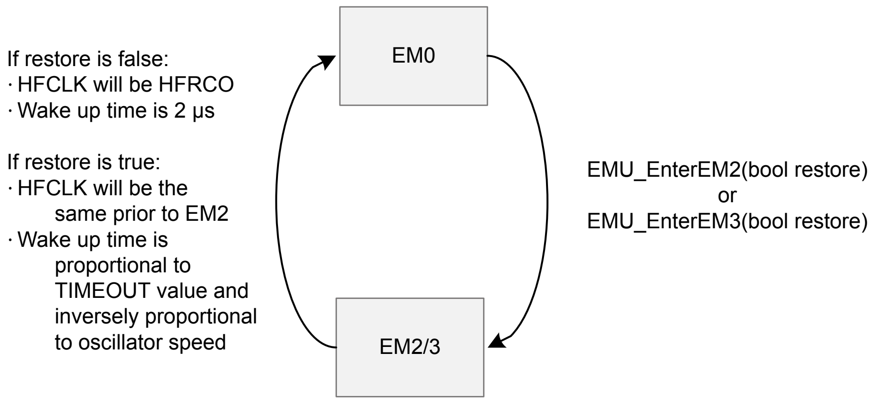

If restore is false:
· HFCLK will be HFRCO
· Wake up time is 2 µs
If restore is true:
· HFCLK will be the same prior to EM2
· Wake up time is proportional to TIMEOUT value and inversely proportional to oscillator speed
EM0
EMU_EnterEM2(bool restore) or EMU_EnterEM3(bool restore)
EM2/3

Figure 2.1. Oscillator restoring

For the LFXO or HFXO oscillators the start-up delay is set by the HFXOTIMEOUT or LFXOTIMEOUT bitfields in the CMU_CTRL register. The value in these bitfields configures the number of cycles before the ready flag is set and it can be adjusted by the user depending on the crystal characteristics (faster or slower start-up). A longer timeout (e.g. 16K cycles) will guarantee that the oscillator will be stable but it also accounts for a long wake up period if polling the oscillator ready flag. Selecting 16K cycles of timeout for a 32 MHz crystal means approximately 500 µs between start-up and the ready flag being set. The timeout can be configured to a lower value which means that the ready flag will be set sooner but the oscillator might not yet be stable.

A recommended way of switching oscillator on wake up without stopping the processor or waiting for the ready flag is using interrupts. The user can enable the HFXO Ready Interrupt before going into EM2/3 and on wake up simply enable the HFXO and continue running code from the HFRCO. When the HFXO is ready it generates an interrupt and then the HF clock source can be switched without the risk of stopping the core.

2.2.3 Start-Up Current

The start-up time can also be affected by the current supplied to the crystal oscillators, which can be adjusted in the HFXOBOOST and LFXOBOOST bitfields in the CMU_CTRL register. It is recommended to leave these bitfields at their default value which configures the EFM32 to supply the maximum current.

2.2.4 Glitch Detector

When the HFXO starts running glitches can appear on the oscillator output while it builds up the oscillation. These glitches can be filtered away if the HFXOGLITCHDETEN bit is enabled. The detector flags all glitches shorter than 1 ns and no glitches longer than 4 ns. The start-up counter is reset each time a glitch is detected.

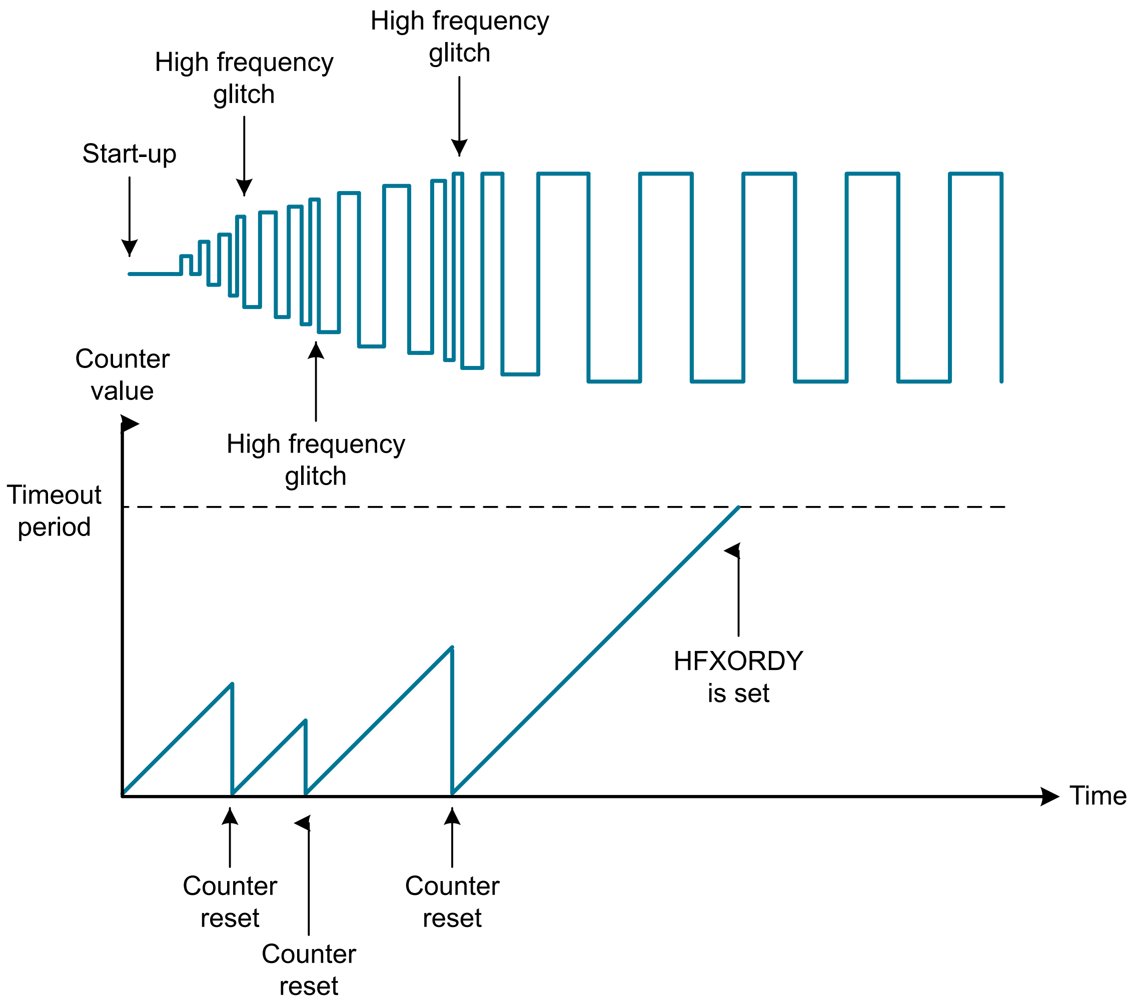


Figure 2.2. Glitch Detector

If the glitch detector is enabled the start-up delay might be longer than the configured TIMEOUT. This depends on how many glitches are detected and thus how many times the start-up counter is reset. For optimization purposes the glitch detector can be enabled or disabled while using different timeouts to achieve a faster crystal wake up.

3. External Clock Sources

The EFM32 or EZR32 can be clocked by an external clock signal such as a square or a sine wave like a crystal or ceramic oscillator. The selection is done using the HFXOMODE or LFXOMODE bitfields in the CMU_CTRL register. For the EFM32 or EZR32 to run properly, these signals should comply with the characteristics defined in the next sections.

3.1 External Sine Wave

To clock the EFM32 or EZR32 with an external sine wave, the BUFEXTCLK option should be set on either HFXOMODE or LFXOMODE. This will couple an ac-coupled buffer in series with the input pin (HFXTAL_N or LFXTAL_N), which is suitable for an external sine wave (4-32 MHz for the high frequency HFXO and 32.768 kHz for the low frequency LFXO). The sine wave should have a minimum of 200 mV amplitude single-ended. For more information, refer to application note, "AN0016: Oscillator Design Considerations".
Note: The allowable HFXO frequencies are 4-24 MHz for Zero Gecko and 4-25 MHz for Happy Gecko.

3.2 Digital External Clock

The EFM32 or EZR32 can also be clocked by an external digital signal connected to the same pins as in the previous section. The DIGEXTCLK option should be selected in HFXOMODE or LFXOMODE and this actually bypasses the oscillator. The signal should be a rail-to-rail square wave with 50% duty cycle (4-32 MHz for the high frequency oscillator and 32.768 kHz for the low frequency oscillator).

3.3 Duty Cycle

The recommended duty cycle for an external HFXO clock is 46% to 54% for correct operation. The key requirement is the minimum clock low time and minimum clock high time implied by this specification. If the minimum low and high times are not being met, then setup violations can occur in the digital logic.

The recommendation is that designs obey the 46-54% duty cycle requirement at the frequencies that determine the number of wait states used, or 16 MHz and 32 MHz.

Table 3.1. External Clock Requirements

Key Frequency	Minimum Clock Low/High Time
16 MHz	28.75 ns
32 MHz	14.38 ns

For example, for an 8 MHz external oscillator, the 16 MHz minimum clock low and high times would be met with 23% duty cycle.

4. RC Oscillator Calibration

The CMU has built-in HW support to efficiently calibrate the RC oscillators at run-time. It compares the HFCLK frequency with a selected reference clock (CALCLK). When the calibration circuit is started, one down-counter (HFCLK counter) and one up-counter (CALCLK counter) are started simultaneously. When the down-counter has reached 0, both counters are stopped and software can read out the reference counter (CALCLK counter) and compare it with the start value of the down-counter.

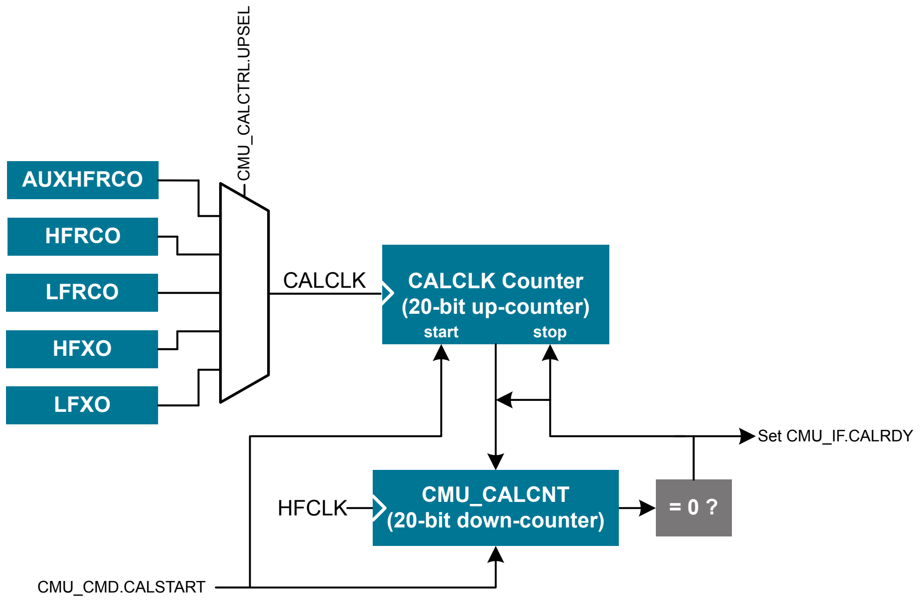


Figure 4.1. Hardware Support for Calibration

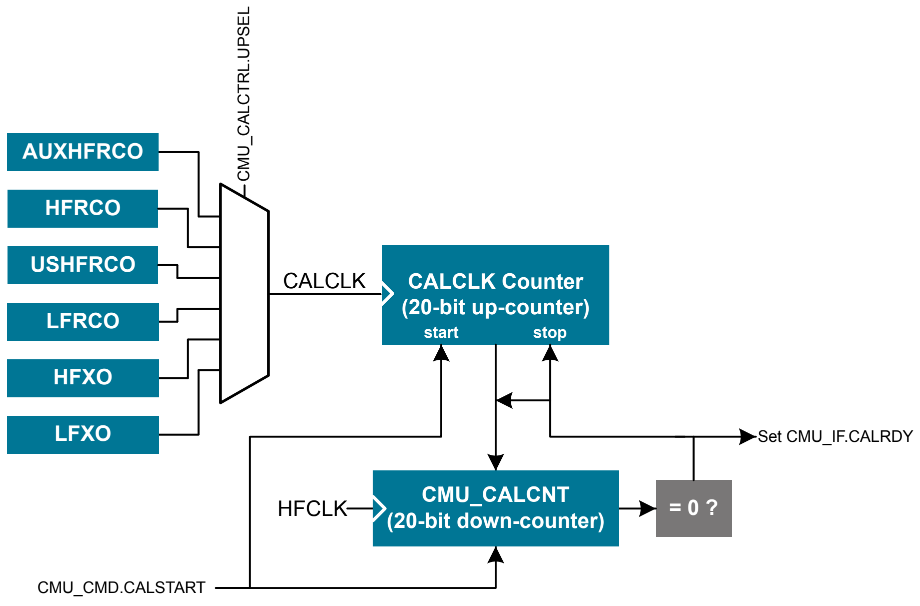


Figure 4.2. Hardware Support for Calibration (Happy Gecko)

The down-counter initial value is set by writing to the CMU_CALCTRL register before starting the calibration. When the calibration is finished, the result for the reference counter is also read from CMU_CALCTRL. There is a function in the emlib to facilitate this task:

```
uint32_t CMU_Calibrate(uint32_t HFCycles, CMU_Osc_TypeDef ref)
```

The `HFCycles` is the value that the down-counter will start from and `ref` is the selected reference oscillator for the CALCLK. The function returns the number of cycles from the reference oscillator counted by the up-counter (CALCLK counter). The following formula can be used to calibrate the RC oscillators:

$$CALCLK_{counter} = \frac{HF_{cycles} \times Ref_{cycles}}{HF_{freq}}$$

Depending on the desired frequency (HF_{freq}), frequency of the reference oscillator (Ref_{freq}) and top value for the down-counter (HF_{cycles}) the user can calculate the expected value for the up-counter ($CALCLK_{counter}$) and adjust the TUNING to obtain the desired frequency for the oscillator.

4.1 Software Example

The `cmu_calib` project exemplifies how to use the calibration routine. It calibrates the HFRCO against the LFXO and displays the several TUNING values on the LCD until it reaches the correct value. The TUNING value for the HFRCO is purposely written with the highest number possible (`0xFF`) so that the HFRCO oscillates at a frequency higher than the default of 14 MHz. The calibration routine runs in a while loop and the TUNING value can either be decremented or incremented in every iteration until the expected value for the $CALCLK_{counter}$ is reached. Then the two last counter values are compared to see which one is closer to the one resulting from the formula and adjust the TUNING value accordingly. The final TUNING value is written on the LCD. If a scope is available, the HFRCO can be probed on the clock output pin (PC12 for the Gecko STK) during the calibration process.

For Happy Gecko, the `cmu_calib` example is the same except it calibrates the USHFRCO against the LFXO.

5. Revision History

5.1 Revision 1.08

2014-02-25

Added Happy Gecko.

Updated format.

5.2 Revision 1.07

2014-05-07

Updated example code to CMSIS 3.20.5

Changed to Silicon Labs license on code examples

Added example projects for Simplicity IDE

Removed example makefiles for Sourcery CodeBench Lite

5.3 Revision 1.06

2013-11-26

New cover layout

5.4 Revision 1.05

2012-11-12

Added software projects for the Tiny and Giant Gecko STKs.

Adapted software projects to new kit-driver and bsp structure.

5.5 Revision 1.04

2012-04-20

Adapted software projects to new peripheral library naming and CMSIS_V3.

5.6 Revision 1.03

2011-10-21

Updated IDE project paths with new kits directory.

5.7 Revision 1.02

2011-05-18

Updated project to align with new bsp version

5.8 Revision 1.01

2011-05-16

Changed Figure 4.1 CALCLK mux selection bit, it was CMU_CALCTRL.REFSEL and now is CMU_CALCTRL.UPSEL.

5.9 Revision 1.00

2010-12-02

Initial revision.

Simplicity Studio

One-click access to MCU tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

www.silabs.com/simplicity

MCU Portfolio
www.silabs.com/mcu

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com