AN0015.1: EFM32 and EFR32 Series 1 Watchdog

This application note demonstrates how to use the Watchdog module on EFM32 and EFR32 Series 1 devices. For watchdog information for EFM32 and EZR32 Wireless MCU Series 0 devices, refer to *AN0015.0: EFM32 and EZR32 Wireless MCU Series 0 Watchdog*.

This document discusses initializing the Watchdog, a basic setup for operation, and ways to utilize the added Watchdog functionality in more advanced applications.

This application note includes:

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

KEY POINTS

- Clock source from selectable oscillators.
- Configurable timeout period.
- Selection to keep running or freeze when entering debug mode.
- Selection to block the CPU from entering EM4H or EM4S.
- Individual selection to keep running or freeze when entering EM2 DeepSleep or EM3 Stop.
- Selection to block the CMU from disabling the selected watchdog clock.
- Advanced features like timeout interrupt without MCU reset, PRS event monitoring or resets, and warning or window interrupt generation.
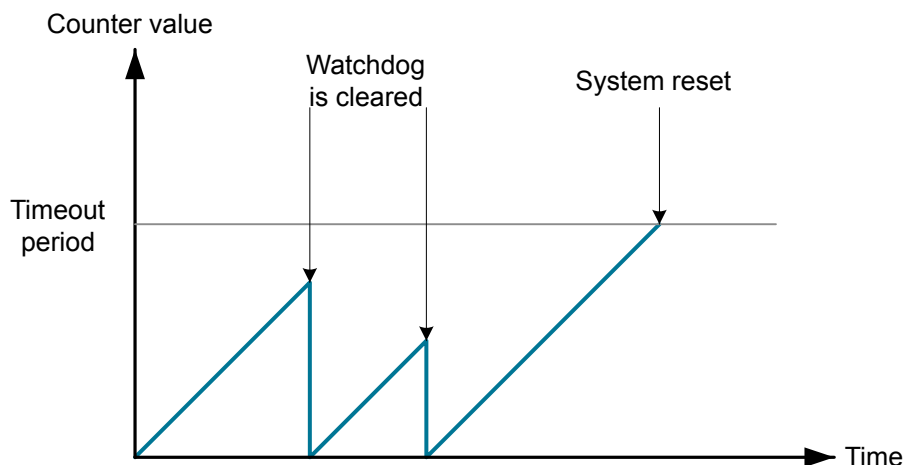
1. Device Compatibility

This application note supports multiple device families, and some functionality is different depending on the device.

MCU Series 1 consists of:
- EFM32 Jade Gecko (EFM32JG1/EFM32JG12/EFM32JG13)
- EFM32 Pearl Gecko (EFM32PG1/EFM32PG12/EFM32PG13)
- EFM32 Giant Gecko (EFM32GG11)

Wireless SoC Series 1 consists of:
- EFR32 Blue Gecko (EFR32BG1/EFR32BG12/EFR32BG13)
- EFR32 Flex Gecko (EFR32FG1/EFR32FG12/EFR32FG13)
- EFR32 Mighty Gecko (EFR32MG1/EFR32MG12/EFR32MG13)

2. Watchdog Theory

2.1 General Theory

The purpose of the Watchdog timer is to generate a reset in case of a system failure to increase application reliability. The failure may be caused by an external event such as an ESD pulse or by a software failure.

The Watchdog circuit is a timer which (when enabled) must be cleared by software regularly. If software does not clear it, a Watchdog reset is activated. This functionality provides recovery from a software stalemate. Refer to the WDOG chapter of the reference manual covering the device for more extensive specifications and descriptions.
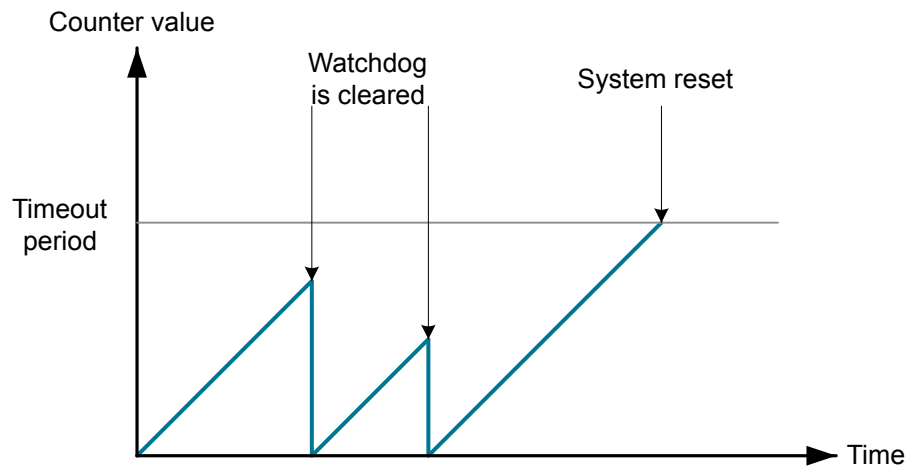
Figure 2.1. Watchdog Timer Operation

2.2 Watchdog Reset

The Watchdog timer on the EFM32 (WDOG) is a Low Energy Peripheral module that can be configured to generate a system reset in the case that it is not cleared by software before the given deadline.

When the Watchdog is cleared (often also referred to as feeding or petting the Watchdog), it means that the value counted to by the timer is reset to zero. This is done by software and is the normal procedure when the system is running correctly.

When the Watchdog timer reaches a certain threshold value, it will generate a system reset. This indicates that the system is not running correctly and that the CPU has failed to reset the Watchdog in a timely manner. In this case, the Watchdog offers a safe way to return the system to a known state through system reset.
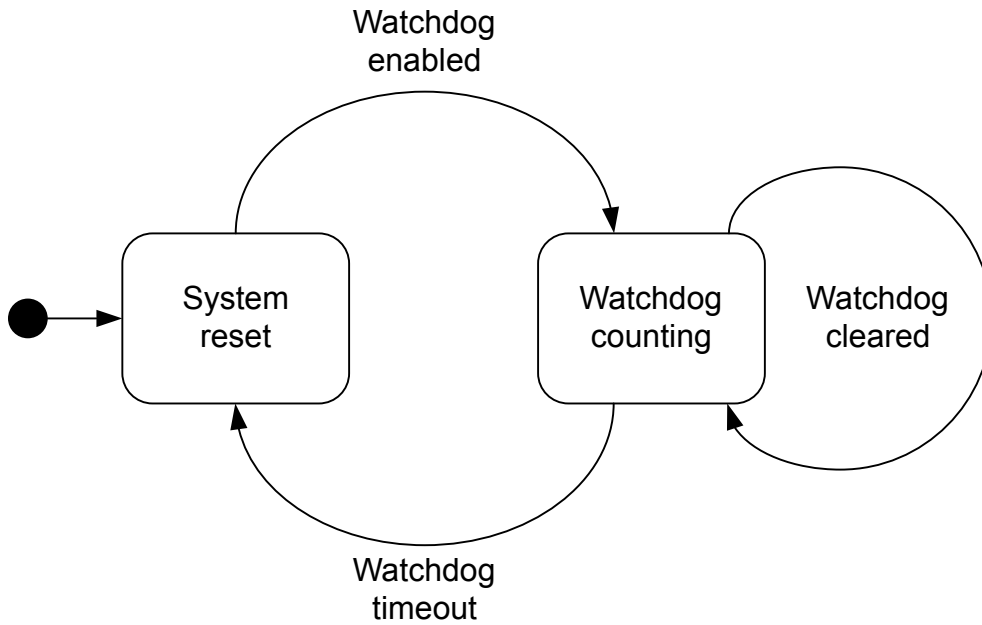

Figure 2.2. Watchdog State Diagram

3. Watchdog Configuration

3.1 Choice of Clock Source

On all devices, the WDOG can be configured to use the following clock sources: LFRCO, LFXO or ULFRCO. The EFM32xG12/13 and EFR32xG12/13 devices can additionally use the HFCORECLK as a Watchdog clock source, which is available in EM0 and EM1.

ULFRCO (Ultra Low Frequency RC Oscillator) is a separate and dedicated Watchdog 1 kHz RC oscillator that also runs in EM3. This oscillator is always enabled and is included in all current consumption numbers.

It is important to take into consideration the ULFRCO accuracy. The oscillator offers extremely low energy consumption, but has reduced accuracy. It is therefore advisable to time the Watchdog resets with considerable margin to avoid unwanted system resets. For example, the EFM32PG1 Pearl Gecko data sheet specifies that the ULFRCO can vary between 0.95 and 1.07 kHz. This means that the timeout period may vary from -6.6% to +5.3% of the desired period, device to device. This is important to remember when testing software on one device that will later run on other devices. A specific timeout period might give good results on one device, but may generate unwanted system resets on a device with a slightly different ULFRCO frequency. The ULFRCO is available down to EM4S.

The LFRCO and LFXO have higher accuracy, but in turn need more energy to operate. They are also shared with other system devices and first need be enabled in software. For EFM32PG1 devices, the LFRCO (32.768kHz) can vary between 30.474 and 34.243 kHz (ENVREF = 1 in CMU_LFRCOCTRL, ambient temperature = 85°C), which means that the timeout period may vary from -7.5% and 4.5% of the desired period. The LFRCO is available down to EM2. The LFXO clock (32.768kHz) accuracy will be dependent on the crystal specification, but is generally more accurate than either the LFRCO or the ULFRCO.

Refer to the specific device data sheet for more extensive information on oscillator accuracy.

All of the Watchdog oscillators have uncertainty that should be accounted for during system design to guarantee predictable system behavior. The clock sources for the Watchdog peripheral are also shared with other system devices and first need to be enabled in software. When using them as Watchdog clock sources, it is critical that they are not disabled in other parts of the software.

In order to access the WDOG from the CPU, the HFCORECLK_LE clock gate must be enabled in the CMU (Clock Management Unit).

3.2 Timeout Periods

The Watchdog offers a wide range of timeout periods to select from during system design. The PERSEL field in WDOGn_CTRL is used to divide the selected watchdog clock, and the timeout for the watchdog timer can be calculated with the formula:

$$T_{TIMEOUT} = \frac{2^{3+PERSEL} + 1}{f}$$

A total of 16 levels offer periods between 9 to 256k cycles of the chosen Watchdog clock source. (i.e. from ~274 µs for 32768 Hz clock source to ~262 s for 1000 Hz clock source).

3.3 Energy Mode Integration

The Watchdog contains extensive support for integration with the different Energy Modes and energy consumption minimization. Different register values toggle if the Watchdog is to keep running in respectively EM2 and/or EM3, and whether or not the system is allowed to enter EM4H or EM4S. For example, the Watchdog can be set to keep counting in EM2, but not during EM3. When resuming from EM3, the Watchdog will keep counting from the previous counter value.

The registers are:

- `WDOG_CTRL_EM2RUN` — toggles if the Watchdog timer is to keep running when the system has entered EM2.
- `WDOG_CTRL_EM3RUN` — toggles if the Watchdog timer is to keep running when the system has entered EM3.
- `WDOG_CTRL_EM4BLOCK` — toggles if the system is disabled by the Watchdog from entering EM4H or EM4S, where all the Watchdog clock sources are disabled.

3.4 Debug Functionality

The watchdog timer can either keep running or be frozen when the device is halted by a debugger. This configuration is done through the DEBUGRUN bit in WDOGn_CTRL. When code execution is resumed, the watchdog will continue counting from the previous value.

3.5 Watchdog Clearing Considerations

All the available Watchdog clocks are asynchronous to the CPU clock except for the HFCORECLK on EFM32xG12/13 or EFR32xG12/13 devices. When using clock sources other than HFCORECLK, synchronization between the CPU and the Watchdog clock domains should therefore be taken into consideration when using the Watchdog. It generally takes 3 WDOGCLK cycles from the instruction is executed until the correct value enters the WDOG registers. To guarantee that the Watchdog timer always is cleared correctly, all clear operations of the Watchdog timer should be written 4 WDOGCLK cycles before timeout (3 + 1 clock of uncertainty). This is critical to remember especially when using short timeout periods. For example, when operating with a Watchdog period of 9 WDOGCLK cycles, the first clear should be no later than 3.3 WDOGCLK cycles (9 – 4 – 33%) after the Watchdog is enabled. The interval between all subsequent Watchdog instructions should be no more than 5.3 WDOGCLK cycles (9 – 1 uncertainty – 33%). When using a clock source different from the ULFRCO, the safety margin should be changed according to the accuracy of that oscillator. This should encourage the use of caution when implementing strict timing schemes for Watchdog resetting. For more information on accessing asynchronous registers and on oscillator accuracy, refer to the Reference Manual for the device.
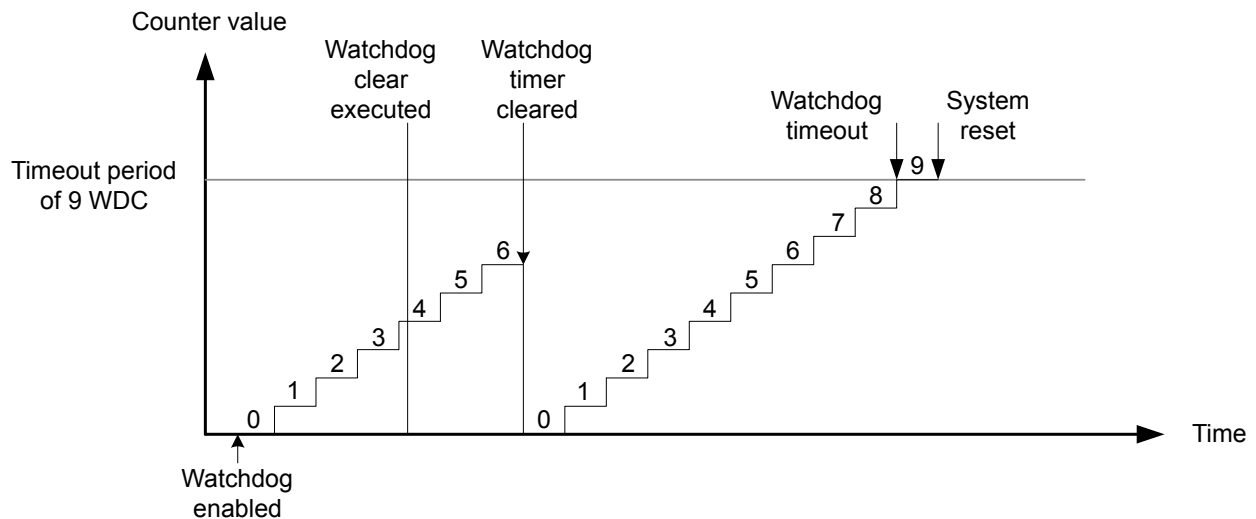

Figure 3.1. Discrete Representation of the Watchdog Timer Operation

To effectively determine when to issue a Watchdog reset in software, it is important to have good knowledge of its timing characteristics. The Watchdog timeout period need be longer than the longest possible execution path through the software initialization and main loop. The execution path length must also include the expected interrupts and their handler's run times. This is to ensure that a large number of interrupts is not regarded by the Watchdog as a system stalemate.

3.6 Reset Cause Detection

The RMU (Reset Management Unit) contains a register that holds information on what caused the last system reset. This register provides a method to detect if the Watchdog has triggered a reset. This information can be very useful in applications using the Watchdog, or to check if the Watchdog generates unwanted system resets. A basic use of the functionality is shown in the software examples. This can also be used to check if the Watchdog triggered a reset even in applications where the watchdog is not in use. This is a secure way to ensure safe operation even when the Watchdog is enabled accidentally. Alternatively, the watchdog register can be locked during startup. For more information, refer to the RMU and WDOG chapters in the Reference Manual for the device.

3.7 Register Locking

Some functionality is added to keep other parts of the system from interfering with the Watchdog. Different register entries can be set to prevent disablement of the chosen clock source, and the Watchdog register can be locked so that no modifications can be made to it. The registers are:

- `WDOG_CTRL_SWOSCBLOCK` — when set, the Watchdog blocks all software from disabling the oscillator driving the WDOG timer.
- `WDOG_CTRL_LOCK` — when set, the WDOG_CTRL register is locked and cannot be altered. The register should be set using the function `void WDOG_Lock(void)` from emlib. The lock can only be disabled by a system reset.

3.8 Watchdog Timeout Interrupt

The reset behavior of the Watchdog timeout is configurable. When the WDOGRSTDIS field of the WDOGn_CTRL register is set to 1, the Watchdog reset is disabled, meaning that the Watchdog counter reaching the timeout value will not cause a device reset. If the TOUT field of the WDOG_IF register is set to 1, then the Watchdog timeout can instead generate an interrupt, as shown in the following figure.
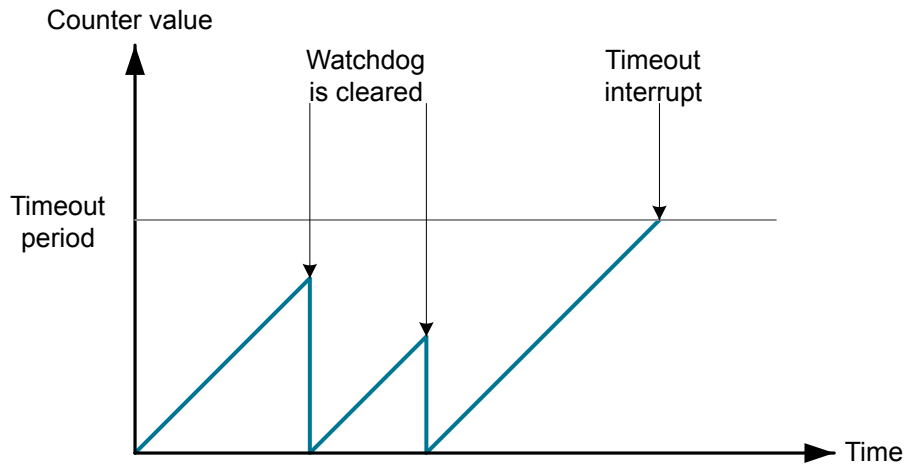

Figure 3.2. Watchdog Timeout Interrupt

3.9 PRS as a Watchdog Clear Source

The first PRS channel (selected by the WDOGn_PCH0_PRSCTRL register) can be used to clear the watchdog counter. To enable this feature, set the CLRSRC field of the WDOGn_CTRL register to 1. The following figure shows how the PRS channel takes over the Watchdog clear function. Clearing the Watchdog with the PRS is mutually exclusive of clearing the Watchdog by software.
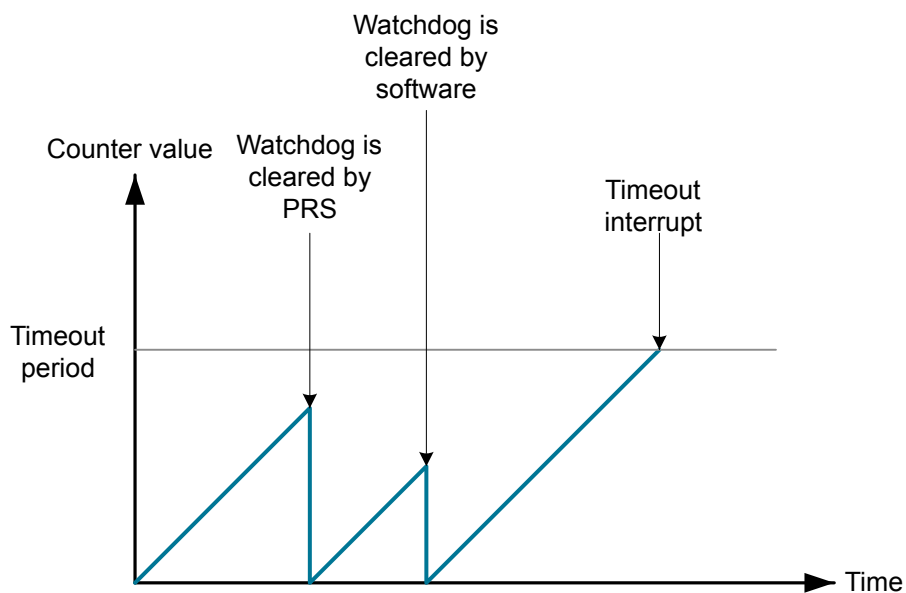

Figure 3.3. PRS as a Watchdog Clear Source

3.10 PRS Rising Edge Monitoring

PRS channels can be used to monitor multiple processes. If enabled, the PRS channels are checked every time the Watchdog timer is cleared, and any channel which has not seen an event can trigger an interrupt. The following figure illustrates these monitoring interrupts and shows where the PRS rising edge must occur.
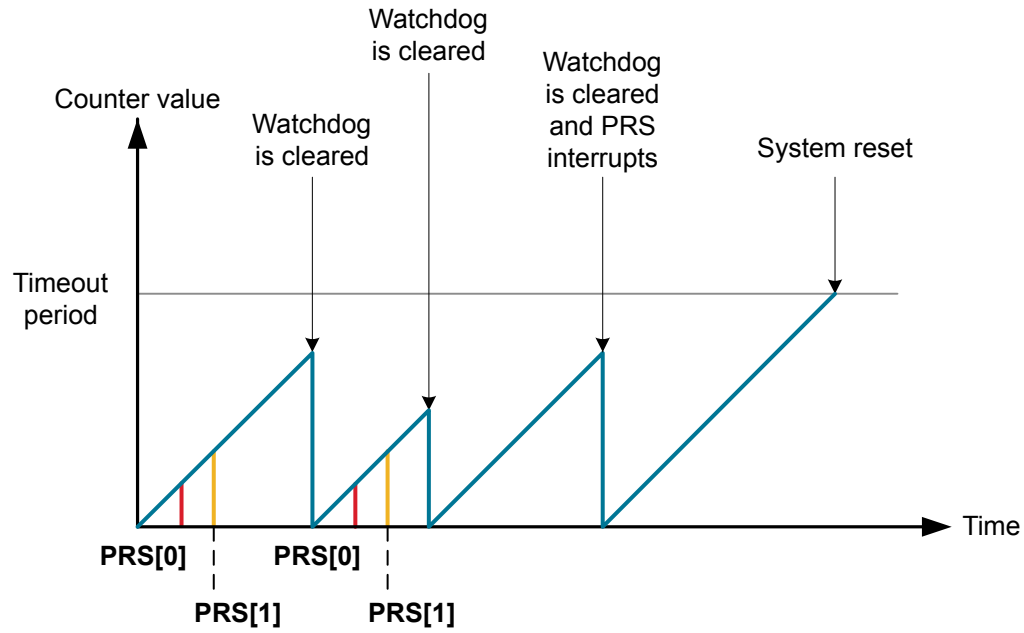
Figure 3.4. Watchdog PRS Edge Monitoring

3.11 Watchdog Warning Interrupt

The Watchdog implements a warning interrupt that can be configured to occur at approximately 25%, 50%, or 75% of the timeout period through the WARNSEL field of the WDOGn_CTRL register. This interrupt can be used to wake up the CPU for clearing the watchdog. The warning point for the watchdog timer can be calculated with the formula:

$$T_{WARNING} = \frac{2^{3+PERSEL} \times \frac{WARNSEL}{4} + 1}{f}$$

where f is the frequency of the selected clock.

The following figure illustrates the warning interrupt. Once warning interrupt was enabled, the Watchdog should be cleared before the Watchdog counter counted up to the threshold (line in red); otherwise, the warning interrupt will fire.

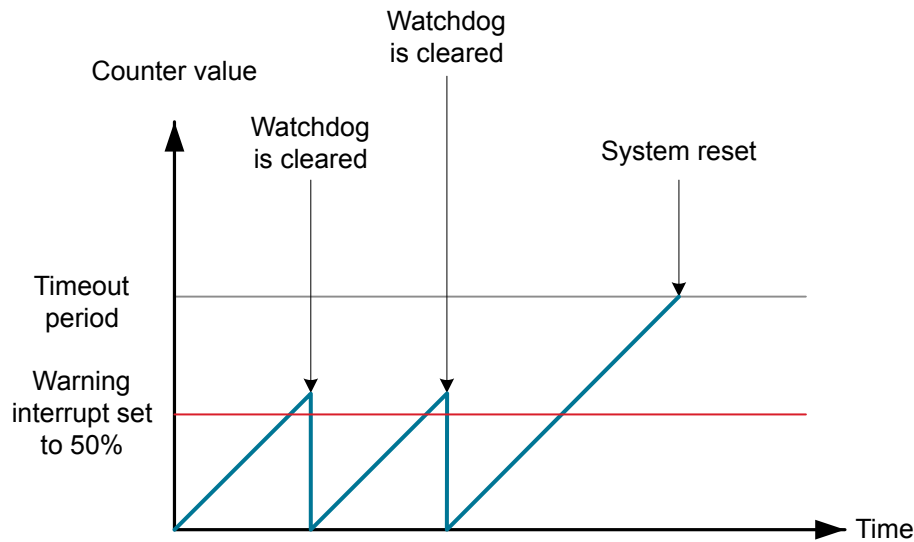When the watchdog is enabled, it is recommended to clear the Watchdog before changing WARNSEL.

Figure 3.5. Watchdog Warning Interrupt Generation

3.12 Watchdog Window Interrupt

This interrupt occurs when the Watchdog is cleared below a certain threshold. This threshold is given by the formula:

$$T_{WINDOW} = \frac{2^{3+PERSEL} \times \frac{WINSEL}{4} + 1}{f}$$

where *f* is the frequency of the selected clock.

This value will be approximately 12.5%, 25%, 37.5%, 50%, 62.5%, 75%, or 87.5% of the timeout value based on the WINSEL field of the WDOGn_CTRL register. The following figure illustrates the window and the watchdog timeout period. Once the window interrupt is enabled, clearing the Watchdog too early causes the window interrupt to fire.

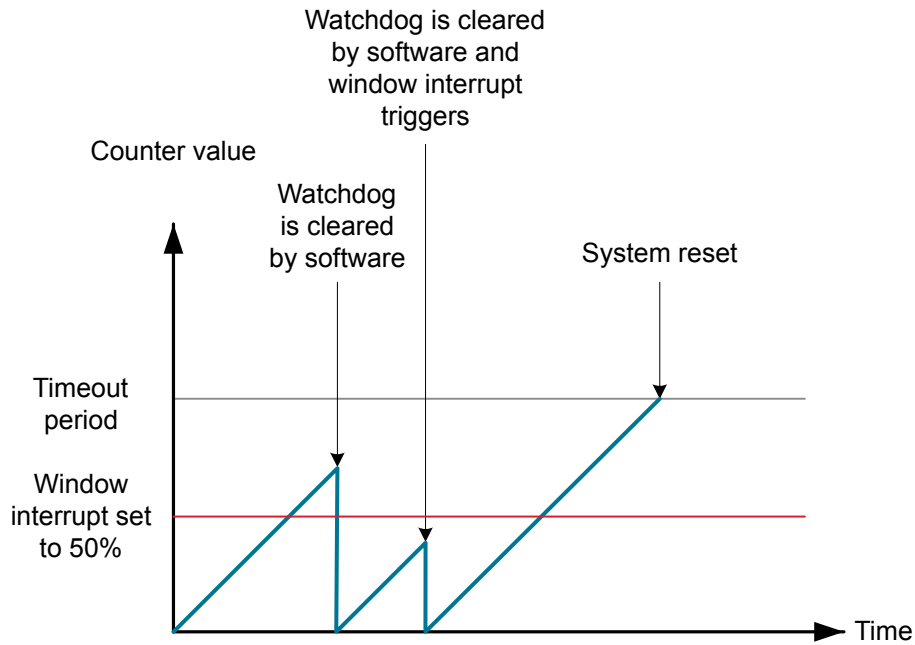When the watchdog is enabled, it is recommended to clear the Watchdog before changing WINSEL.

Figure 3.6. Watchdog Window Interrupt Generation

4. Software Examples

The software example associated with this document demonstrates how to use the main functionality included in the Watchdog module of the device. The example can be used on a Starter Kit (STK).

The LCD module is used to provide feedback. The examples uses PB1 to choose the example mode, and PB0 starts the mode. LED0 (blinking) on the STK is used to indicate the status of the example mode.

4.1 Mode 0: Basic Watchdog Operation

This example mode uses the ULFRCO as the Watchdog clock source. The software feeds the Watchdog every 50 ms using the usTimer driver. When running the mode, LED0 toggles whenever the firmware feeds the Watchdog. The reset source number will be displayed on the LCD after the Watchdog causes a reset.

4.2 Mode 1: Lock Operation

This example mode uses the LFXO as the Watchdog clock source. The mode demonstrates the Watchdog lock feature. When running the example, LED0 toggles whenever the firmware feeds the Watchdog. The reset source number will be displayed on the LCD after the Watchdog causes a reset.

4.3 Mode 2: Watchdog Interrupt

This mode uses the LFRCO as the Watchdog clock source. This mode demonstrates the Watchdog interrupt feature by generating an interrupt instead of resetting the MCU whenever the Watchdog counter overflows.

4.4 Mode 3: PRS as a Watchdog Clear Source

This example mode uses the ULFRCO as the Watchdog clock source. In this mode, the LETIMER0 module through PRS is used to clear the Watchdog regularly (every ~0.5 s).

4.5 Mode 4: PRS Rising Edge Monitoring

This example mode uses the ULFRCO as the Watchdog clock source. In this example mode, usTIMER feeds the Watchdog every 1 s. The LETIMER0 generates PRS events every 3 s. The Watchdog interrupt then triggers since PRS event does not happen before software resets the Watchdog.

4.6 Mode 5: Warning Interrupt

This mode uses the ULFRCO as the Watchdog clock source. The software configures the warning threshold to be 25% of the Watchdog interval period (~1 s). Software then resets the Watchdog whenever the warning interrupt triggers.

4.7 Mode 6: Window Interrupt

This mode uses the ULFRCO as the Watchdog clock source, and the usTimer driver feeds the Watchdog every 1 second. The window threshold is set to elapse after 2 seconds, or 50% of the Watchdog interval period. This causes the interrupt to occur, since the Watchdog is cleared below the window threshold.

5. Revision History

5.1 Revision 1.09

2017-09-12

Initial revision.

Disclaimer

Silicon Labs intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Labs products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Labs reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Labs shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Labs. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Labs products are not designed or authorized for military applications. Silicon Labs products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc.® , Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOmodem®, Micrium, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress®, Zentri and others are trademarks or registered trademarks of Silicon Labs. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

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

SILICON LABS

http://www.silabs.com