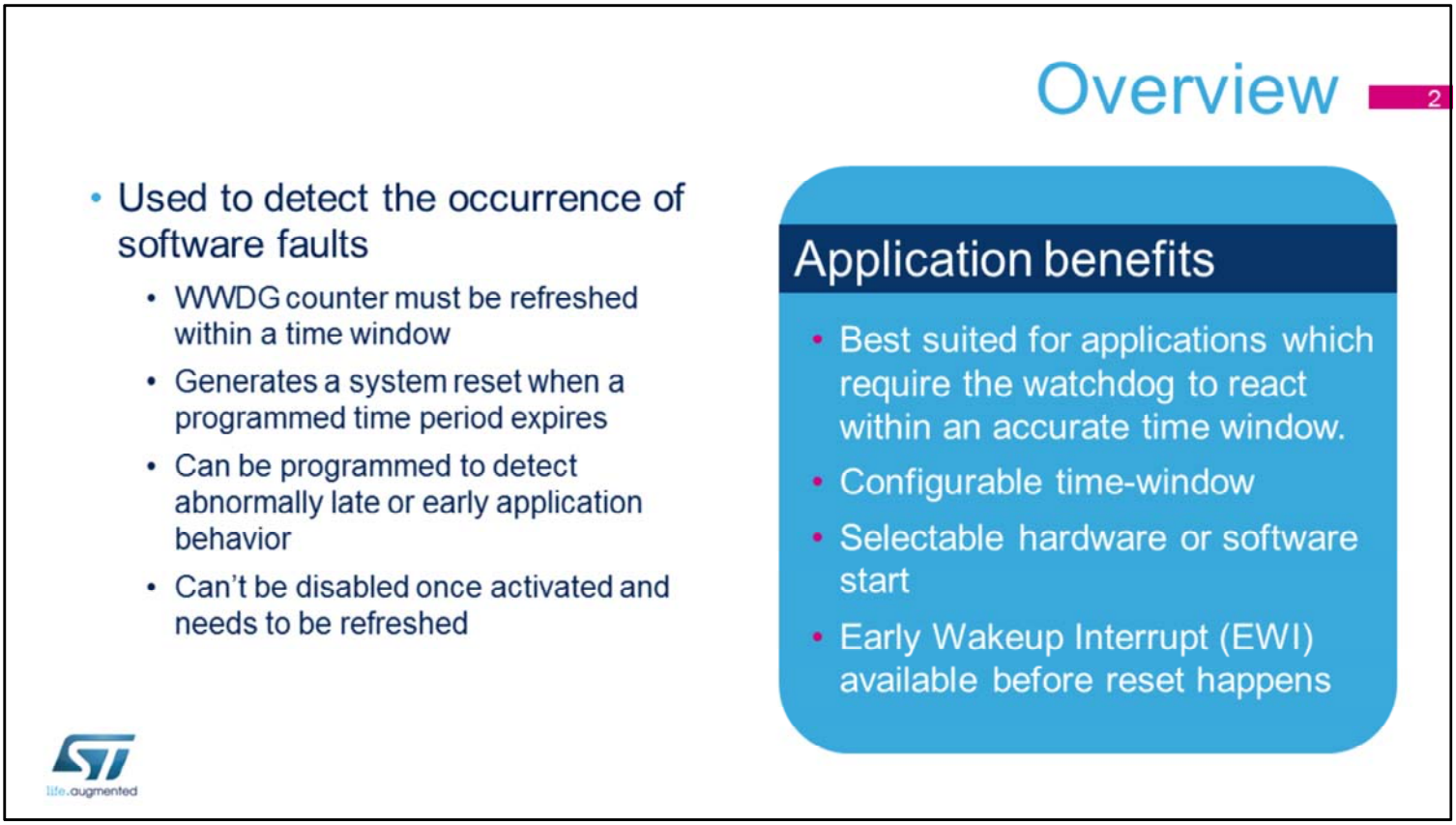
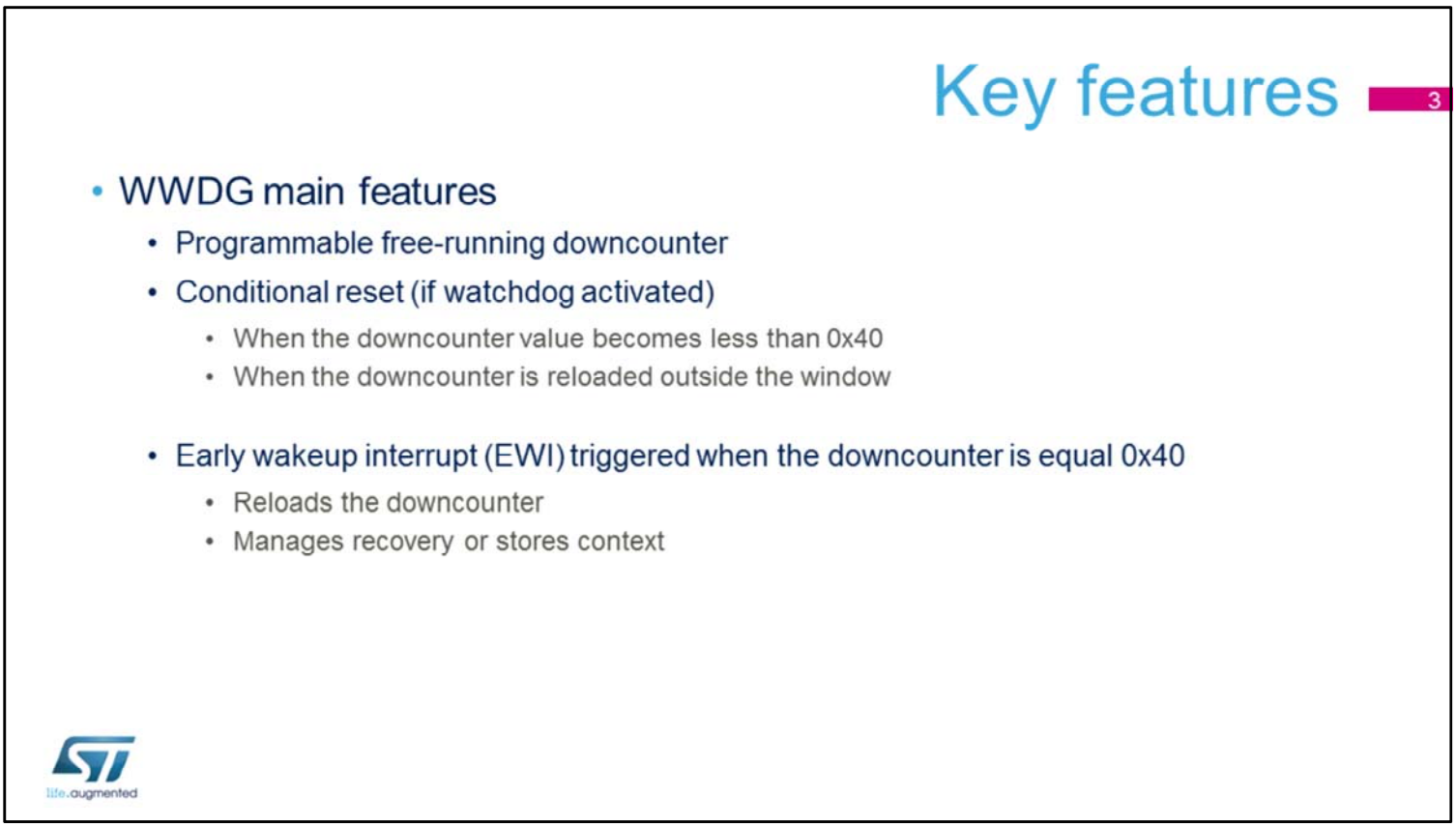

Hello, and welcome to this presentation of the STM32 system window watchdog. It will cover the main features of this peripheral used to detect software faults.




The window watchdog is used to detect the occurrence of software faults.
The window watchdog has a programmable free-running downcounter that must be refreshed within a window period that guarantees proper software execution. If a problem occurs and the programmed time period expires, the window watchdog generates a system reset.
The window watchdog can be programmed to detect abnormally late or early application behavior. Once enabled, it can only be disabled by a device reset.
The window watchdog is best suited for applications required to react within an accurate timing window. This time-window is configurable and can be adjusted according to various use cases.
The window watchdog can be configured to start either by hardware or software via the option bytes.
An Early Wakeup Interrupt can be generated before a reset happens to perform a system recovery or manage certain

actions before a system restart.

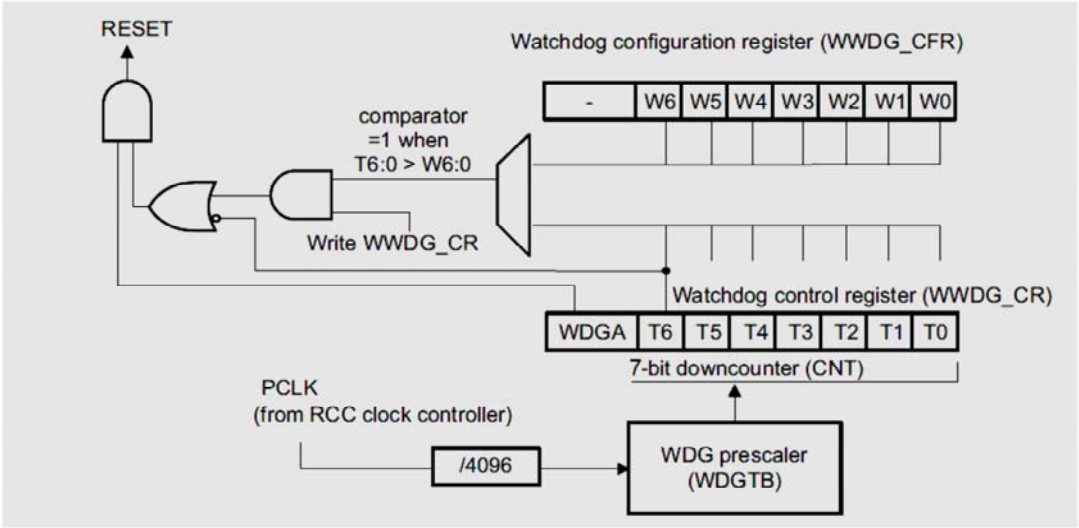



When the window watchdog is activated, a reset can occur when the downcounter value becomes less than 0x40 or when the downcounter is reloaded outside the time window.

An Early Wakeup interrupt can trigger any action when the downcounter reaches 0x40. The EWI status register can be used to reload the downcounter, to avoid generating a reset, or to manage system recovery and context backup operations.

# Block diagram





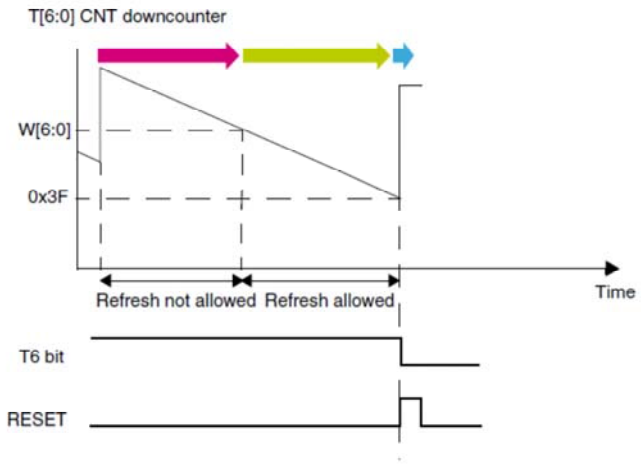

The PCLK clock from RCC clock controller is used to clock the watchdog peripheral. Bits T[6:0] from the watchdog control register count down until they roll over from 0x40 to 0x3F, which then generates a reset.

Bits W[6:0] from the watchdog configuration register contain the window value. Bits T[6:0] and W[6:0] are compared in order to evaluate the time to refresh the downcounter in the configurable window.

If the downcounter is reloaded too early or too late, the window watchdog will initiate a reset.

# WWDG functional description






If the software reloads the counter while the counter is greater than the value stored in W[6:0], then a reset is generated.

To prevent a WWDG reset, write the reload value T[6:0] while the counter value is lower than the time-window value W[6:0].

When the 7-bit downcounter T[6:0] rolls over from 0x40 to 0x3F, it initiates a reset.

life.augmented

This diagram illustrates how the window watchdog operates. When the 7-bit downcounter T[6:0] bits roll over from 0x40 to 0x3F, it initiates a reset when the T6 bit is cleared. This happens when the application software did not react within the expected time window.
If the software reloads the counter while the counter is greater than the value stored in W[6:0] bits, then a reset is generated. This happens when the application refreshes the counter too early.
To prevent a window watchdog reset, the reload value T[6:0] bits must be written while the counter value is lower than the time-window value W[6:0] bits located in the green area.

# WWDG settings and reset flag



- Enable the window watchdog clock:
  - Set the WWDGEN bit to "1" in the RCC_APB1ENR1 register
- Setting the WWDG time base:
  - WWDG time base pre-scaled from PCLK1 clock
    - 4096 internal divider and 4 pre-dividers: 1, 2, 4, or 8 selectable by WWDG_CFR register
  - Setting the WWDG timeout by using the following formula:

    t_WWDG (ms) = t_PCLK1 (ms) x 4096 x 2^WDGTB[1:0] x (T[5:0] +1)
  - Min. and max. timeout values from 51.2 µs to 26.2 ms
- Checking the WWDG reset source:
  - The WWDGRSTF reset flag indicates when a WWDG reset occurs (after device reset)


To enable the window watchdog clock, set the WWDGEN bit in the RCC_APB1ENR1 register.
The window watchdog time base is pre-scaled from PCLK1/APB1 whose maximum frequency can go up to 80 MHz.
This clock frequency is first pre-divided by 4096 and the window watchdog pre-scaler can divide it again by 1, 2, 4 or 8 as defined in the WWDG_CFR register.
The formula shown in this slide lets you determine the watchdog timeout, which is derived from the PCLK1 period and the WDGTB pre-scaler as well as the selected watchdog counter reload value.
The minimum and maximum timeout values can be between 51.2 µs and 26.2 ms.
Once the window watchdog generates a reset, a status flag WWDGRSTF is set in the RCC_CSR register identifying the source of the reset.

# Interrupts



| Interrupt event | Description |
|---|---|
| EWI | Early Wakeup Interrupt. It can be used in specific safety operations or when data logging must be performed before the actual reset is generated. |

- EWI interrupt occurs when the downcounter value reaches 0x40
- EWI interrupt is enabled by setting the EWI bit in the WWDG_CFR register
- EWI interrupt is cleared by writing “0” to the EWIF bit in the WWDG_SR register

life.augmented

The Early Wakeup Interrupt can be used for specific safety operations or when data logging must be performed before the actual reset is generated.
The EWI interrupt occurs whenever the downcounter value reaches 0x40.
It is enabled by setting the EWI bit in the WWDG_CFR register.
The EWI interrupt is cleared by writing “0” to the EWIF bit in the WWDG_SR register.

# Low-power modes



| Mode | Description |
|---|---|
| **Run** | Active*. |
| **Sleep** | Active*. Window watchdog clock can be disabled by clock gating by clearing WWDGLPEN bit in RCC_APB1LPENR register. |
| **Stop** | Not available. |
| **Standby** | Not available. |

* If WWDG enabled

The window watchdog is active in Run and Sleep modes. It is not available in Stop or Standby modes.
In Sleep and Low-power sleep modes, the window watchdog clock can be disabled by clock gating by clearing the WWDGLPEN bit in the RCC_APB1LPENR register.

# Debug information

9

- The WWDG counter can be stopped when the core is halted, depending on the DBG_WWDG_STOP configuration bit in the DBG module.

When the microcontroller enters Debug mode with the core halted, the window watchdog counter either continues to work normally or stops, depending on the DBG_WWDG_STOP configuration bit in the DBG module.

Thank you.