STM32MP1 – WWDG

System Window Watchdog

Revision 1.0

life.augmented

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.

Overview

2

- Used to detect the occurrence of software faults
  - WWDG counter must be refreshed within a time window
  - Generates a system reset when a programmed time period expires
  - Can be programmed to detect abnormally late or early application behavior
  - Can't be disabled once activated and needs to be refreshed

Application benefits

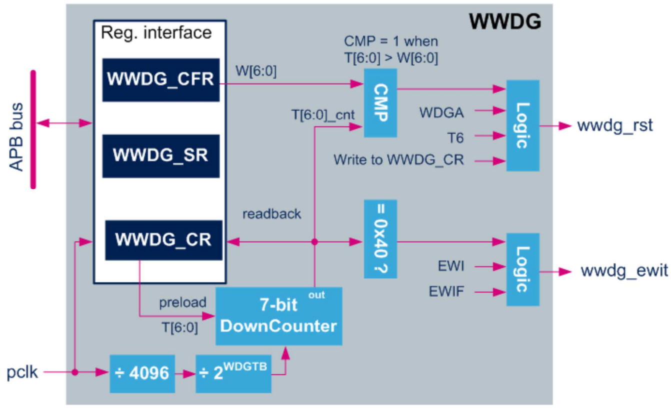
- Best suited for applications which require the watchdog to react within an accurate time window.
- Configurable time window
- Early Wakeup Interrupt (EWI) available before reset happens

life.augmented

The window watchdog is used to detect the occurrence of software faults.
The window watchdog can be programmed to detect abnormally late or early application behavior.
It is best suited for applications required to react within an accurate timing window.
Once enabled, it can only be disabled by a device reset.
An Early Wakeup Interrupt can be generated before a reset happens to perform a system recovery or manage certain actions before a system restart.

Key features

- WWDG main features
  - Programmable timeout value
  - Programmable time window width
  - Reset generation:
    - When the timeout value is reached
    - When it is refreshed outside the time-window
  - Early wakeup interrupt (EWI)
    - Generated before the timeout value is reached



life.augmented

The window watchdog offers several features:

- The user can program the timeout value and the window width according to application needs.
- It can generate a reset under two conditions:
  - when the downcounter value becomes less or equal to 0x3F, or
  - when the watchdog is refreshed outside the time-window.
- It can generate an early wakeup interrupt when the downcounter reaches 0x40.

  The early wakeup interrupt can be used to reload the downcounter in order to avoid a reset generation, or to manage system recovery and context backup operations.

As shown in the figure, the window watchdog uses the APB clock (pclk), as reference clock for its time-base.
The pclk is provided by the RCC block.

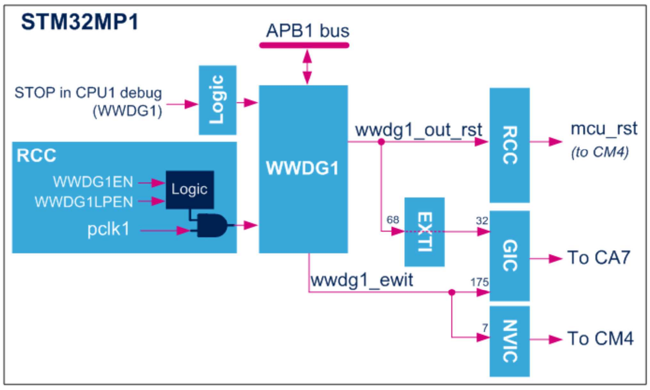
This clock is divided by 4096, and by a value programmed by the application.

The application can also program the reload value of the downcounter bits T[6:0].

The window width is controlled by bits W[6:0].

WatchDog Integration

- The WWDG1 is dedicated to the MCU (Cortex M4)
- The WWDG1 can generate an early interrupt for the Cortex M4 and an for the Cortex A7.
- The WWDG1 can generate an interrupt to the Cortex A7 (via EXTI) when the watchdog reset is activated.
- The WWDG1 generates a local reset to the Cortex M4.



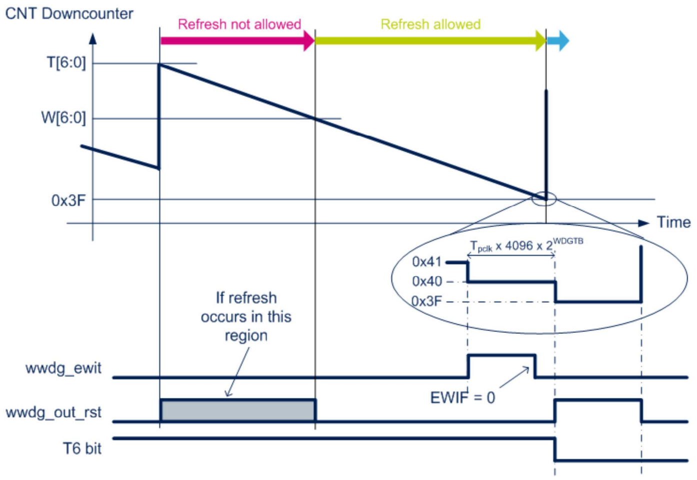
The STM32MP1 includes a window watchdog dedicated to Cortex M4 usage.
WWDG1 is connected to the APB1 bus, and the APB clock must be enabled via WWDGEN bit, prior to use the WWDG1.

The WWDG1 early interrupt output is connected to the NVIC and to the GIC, so that both Cortex M4 and A7 can handle the interrupt.
The WWDG1 reset output is connected to the RCC, and generates a reset of the Cortex M4 only. In addition this reset output is connected to the EXTI in order to activate the Cortex A7 when a window watchdog reset occurs. This feature allows the Cortex A7 to restart properly the Cortex M4 if needed.

It is possible to select if the watchdog will be frozen when the Cortex M4 is in Debug mode (core halted) . Please refer to the MPU debug unit (DBGMCU) description for details.

Note as well that window watchdog is frozen when the Cortex M4 is in CStop or Standby mode, but can remain active when it is in CSleep mode.

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.

This diagram illustrates how the window watchdog operates.
When the 7-bit downcounter rolls over from 0x40 to 0x3F, it initiates a reset. This happens if the application software does not refresh the window watchdog on time.
The early interrupt, if enabled can be generated when the downcounter reaches 0x40.

If the software refreshes the watchdog while the downcounter is greater than the value stored in bits W[6:0], a reset is generated.
This happens when the application refreshes the watchdog too early. No interrupt is generated in this case.

To prevent a window watchdog reset, the watchdog refresh must happen while the downcounter value is

lower than the time-window value, and greater than 0x3F. This is illustrated by the green area.
The refresh operation consists on reloading the downcounter with bits T[6:0].

WWDG settings and reset flag

- Enable the window watchdog clock:
  - In the RCC block:
    - Set the WWDG1EN bit to ‘1’ in order to provide the APB clock to the watchdog
    - Set the WWDG1LPEN bit to ‘1’ in order to keep the watchdog running in Sleep mode
- Setting the WWDG time base:
  - WWDG time base pre-scaled from PCLK clock
    - 4096 internal divider and 8 pre-dividers: 1, 2, 4, 16 ... 128 selectable by register WWDG_CFR
  - Setting the WWDG timeout by using the following formula:

$$t_{WWDG}\ (ms) = t_{PCLK} \times 4096 \times 2^{WDGTB} \times (T[5:0] + 1)$$

- Checking the WWDG reset source:
  - A Reset flag in the RCC block indicates when a WWDG1 reset occurs (after device reset)

life.augmented

To enable the window watchdog clock, the corresponding window watchdog enable bit in the RCC block must be set to 1.
Note that once the APB clock for the watchdog is enabled, the application cannot disable it. Only a system reset can disable the watchdog clock.
A low-power enable bit can be set as well if the application wishes to keep the window watchdog activated, even if the CPU is in Sleep mode.
The downcounter uses the APB clock PCLK divided by 4096, and again divided by a division ratio selected by the application.
It can be 1, 2, 4, 8, 16, 32, 64 or 128 as defined in the WWDG_CFR register.
The formula shown in this slide lets you determine the watchdog timeout value.

When a system reset occurs, it is possible to identify which

parts causes the reset, thanks to status flags provided by the RCC block.
The window watchdogs can be one of the sources.

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 in order to perform emergency tasks before the reset occurs, such as:

- Data logging,
- Data protection,
- Watchdog refresh in order to prevent the reset, or
- Other emergency tasks...

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

CM4 Mode	Description
CRun	Active*.
CSleep	Active*. Window watchdog clock can be disabled by clock gating when the WWDGxLPEN bit in the RCC block is cleared.
CStop	Not available.

* If WWDG enabled

The window watchdog is active when the Cortex M4 is in **CRun and CSleep** modes. It is not available in Stop or Standby modes.
In CSleep mode, the window watchdog clock can be disabled by clearing the corresponding low-power enable bit located in the RCC block.