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 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.
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 \( W[6:0] \).
The WWDG is connected to the APB1 bus and clocked by the APB1 clock. 
The WWDG early interrupt output is connected to the position 0 of the NVIC.

It is possible to select the hardware or software start, via an option byte. In hardware mode, the WWDG can be automatically enabled after reset.
The window watchdog is frozen when the system is in one of the Stop modes, but can remain active when the product is in Sleep mode.
The WWDG performs a system reset handled by the RCC block, when a timeout occurs or when the WWDG is refreshed outside the allowed window.
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 Enabling

- Enabling the WWDG in software mode:
  - In the RCC block:
    - Set the WWDGEN bit to ‘1’ in order to provide the APB clock to the watchdog
    - Set the WWDGSMEN bit to ‘1’ in order to keep the watchdog running in Sleep mode
  - In the WWDG block:
    - Set WDGA bit to ‘1’

- Enabling the WWDG in hardware mode:
  - The WWDG is running as soon as the product is in Run or Sleep.

The WWDG can work either in hardware or software mode. In software mode, the application needs to enable the APB1 watchdog clocks via the RCC, and set the bit WDGA to ‘1’ in the WWDG, in order to enable the watchdog. Note that once the watchdog is enabled, the application cannot disable it. Only a system reset can disable the watchdog clock. The low-power enable bit can be set as well if the application wishes to keep the window watchdog activated, even if the product is in Sleep mode.

In hardware mode, there is no need to enable the watchdog: the WWDG is counting-down when the product is in Run or Sleep. The bits WWDGEN and WWDGSMEN are forced to ‘1’ by the hardware.
The downcounter uses the APB1 clock, divided by 4096, and again divided by a division ratio selected by the application. This division ratio can be 1, 2, 4 or 8, 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.
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.
The window watchdog is active when the device is in Run or Sleep modes. It is not available in Stop or Standby modes. In Sleep mode, the window watchdog clock can be disabled by clearing the corresponding low-power enable bit located in the RCC block.

<table>
<thead>
<tr>
<th>Power Modes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>Active.</td>
</tr>
<tr>
<td>All Sleep modes</td>
<td>Active. Window watchdog clock can be disabled by clearing the WWDGSMEN bit in the RCC block. Peripheral interrupt causes the product to exit Sleep modes.</td>
</tr>
<tr>
<td>All Stop modes</td>
<td>Frozen. Peripheral registers content is kept.</td>
</tr>
<tr>
<td>Standby</td>
<td>Powered down. The peripheral must be initialized after exiting Standby mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>Powered down. The peripheral must be initialized after exiting Shutdown mode.</td>
</tr>
</tbody>
</table>