TIMER

First we will see what are timers, their working and later we will configure the PIC16f877a timers to generate delay of 100ms and 500ms respectively. At the end we will see how to use the Explore Emdedded Timer library.

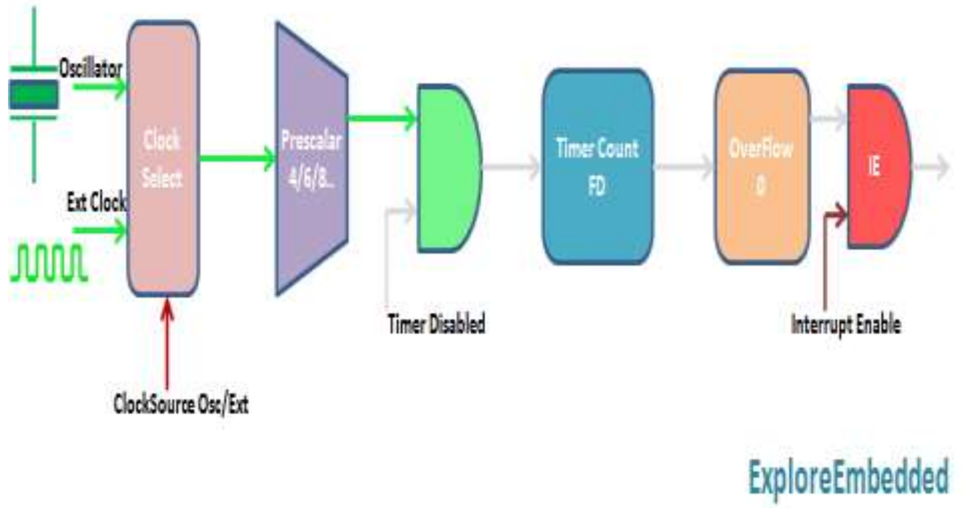
Timer Basics

As the name suggests these are used to measure the time or generate the accurate time delay. The microcontroller can also generate/measure the required time delays by running loops, but the timer/counter relieves the CPU from that redundant and repetitive task, allowing it to allocate maximum processing time for other tasks.

Timer is nothing but a simple binary counter that can be configured to count clock pulses(Internal/External). Once it reaches the Max value, it will roll back to zero setting up an **OverFlow** flag and generates the interrupt if enabled.

TIMER

Timer Block Diagram



TIMER

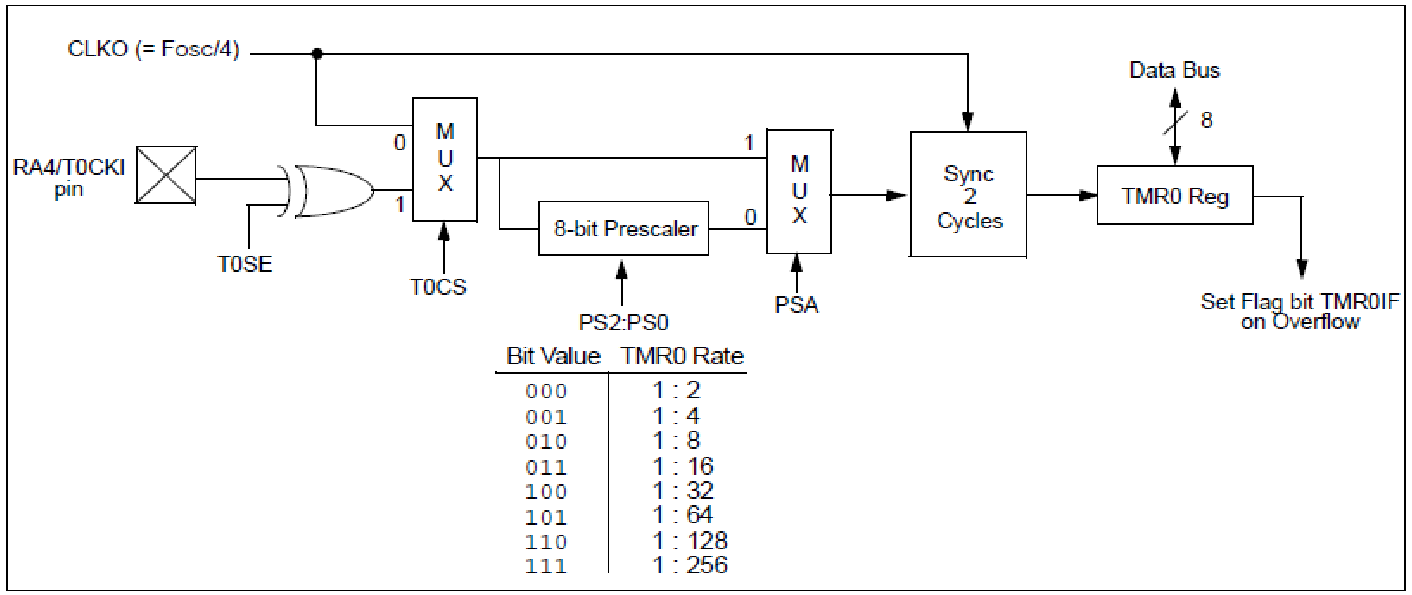
Timer 0

The TMR0 module is an 8-bit timer/counter with the following features:

- 8-bit timer/counter
- Readable and writable
- 8-bit software programmable prescaler
- Internal or external clock select
- Interrupt on overflow from FFh to 00h
- Edge select for external clock

TIMER

Timer0 Block Diagram Excluding WDT



Bit Value	TMR0 Rate
000	1 : 2
001	1 : 4
010	1 : 8
011	1 : 16
100	1 : 32
101	1 : 64
110	1 : 128
111	1 : 256

TIMER

Below are the steps for configuring and using the Timer0 for delay generation:

1.Calculate the Timer Count for the required delay.

2.Set the Presaclar bits in **OPTION_REG** as per the delay calculations.

3.Clear the **PSA**bit for using the prescalar.

4.Select the Clock Source Internal/External using **TOCS** bit.

5.Load the timer value into **TMRO** register.

6.Enable the Timer0 Interrupt by setting **TMR0IE** bit

7.Enable the Global and Peripheral interrupts by setting **GIE** and **PIE** bits

Below is the sample code to blink the LEDs with **1ms** delay.

TIMER

Timer 1

The timer TMR1 module is an 16-bit timer/counter with the following features:

•16-bit timer/counter with two 8-Bit register TMR1H/TMR1L

•Readable and writable

•software programmable prescaler upto 1:8

•Internal or external clock select

•Interrupt on overflow from FFFFh to 00h

•Edge select for external clock

TIMER

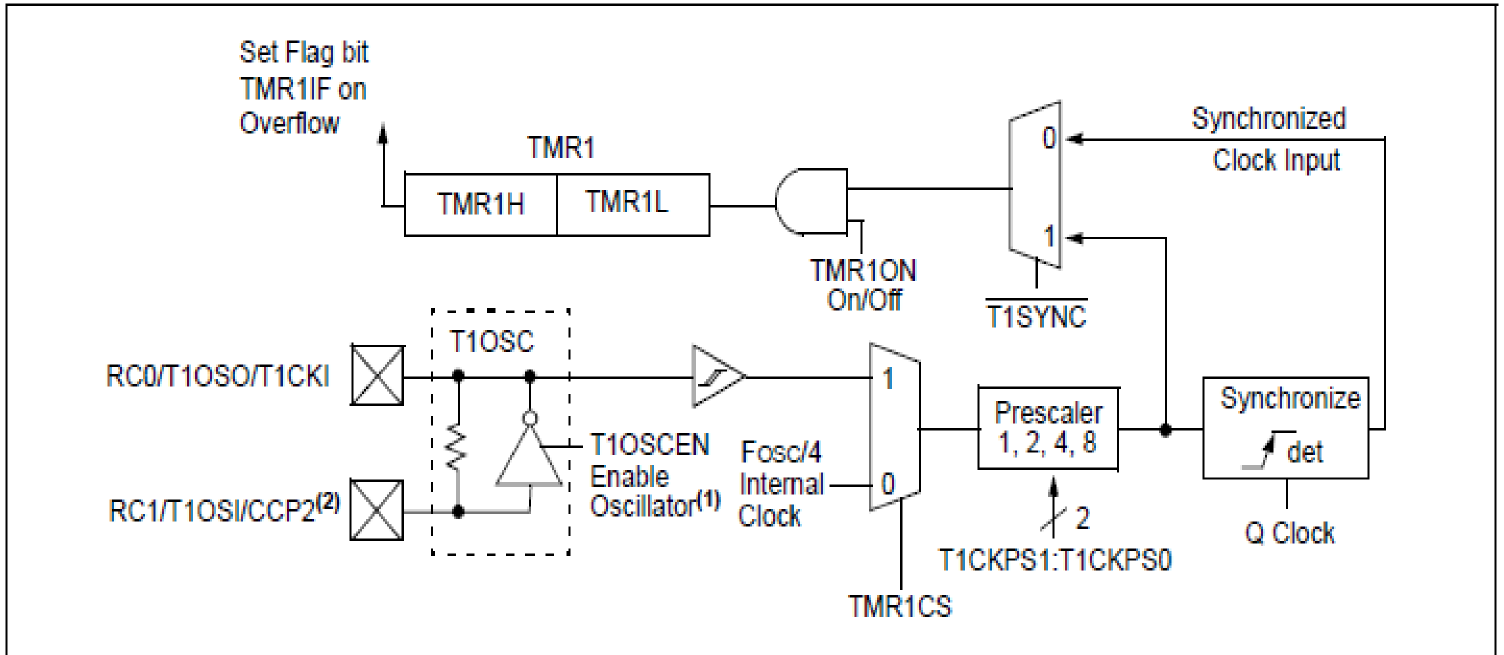
Below are the steps for configuring and using the Timer1 for delay generation:

1.Calculate the Timer Count for the required delay.
2.Set the Presaclar bits in **T1CON** as per the delay calculations.
3.Select the Clock Source Internal/External using **TMR1CS** bit.
4.Load the timer value into **TMR1H,TMR1L** register.
5.Enable the Timer1 Interrupt by setting **TMRIE** bit
6.Enable the Global and Peripheral interrupts by setting **GIE** and **PIE** bits
7.Finally start the timer by setting **TMR1ON** bit

Below is the sample code to blink the LEDs with **100ms** delay.

TIMER

TIMER1 BLOCK DIAGRAM



Note 1: When the T1OSCEN bit is cleared, the inverter is turned off. This eliminates power drain.

TIMER

Timer 2

The TImer2 module is an 8-bit timer/counter with the following features:

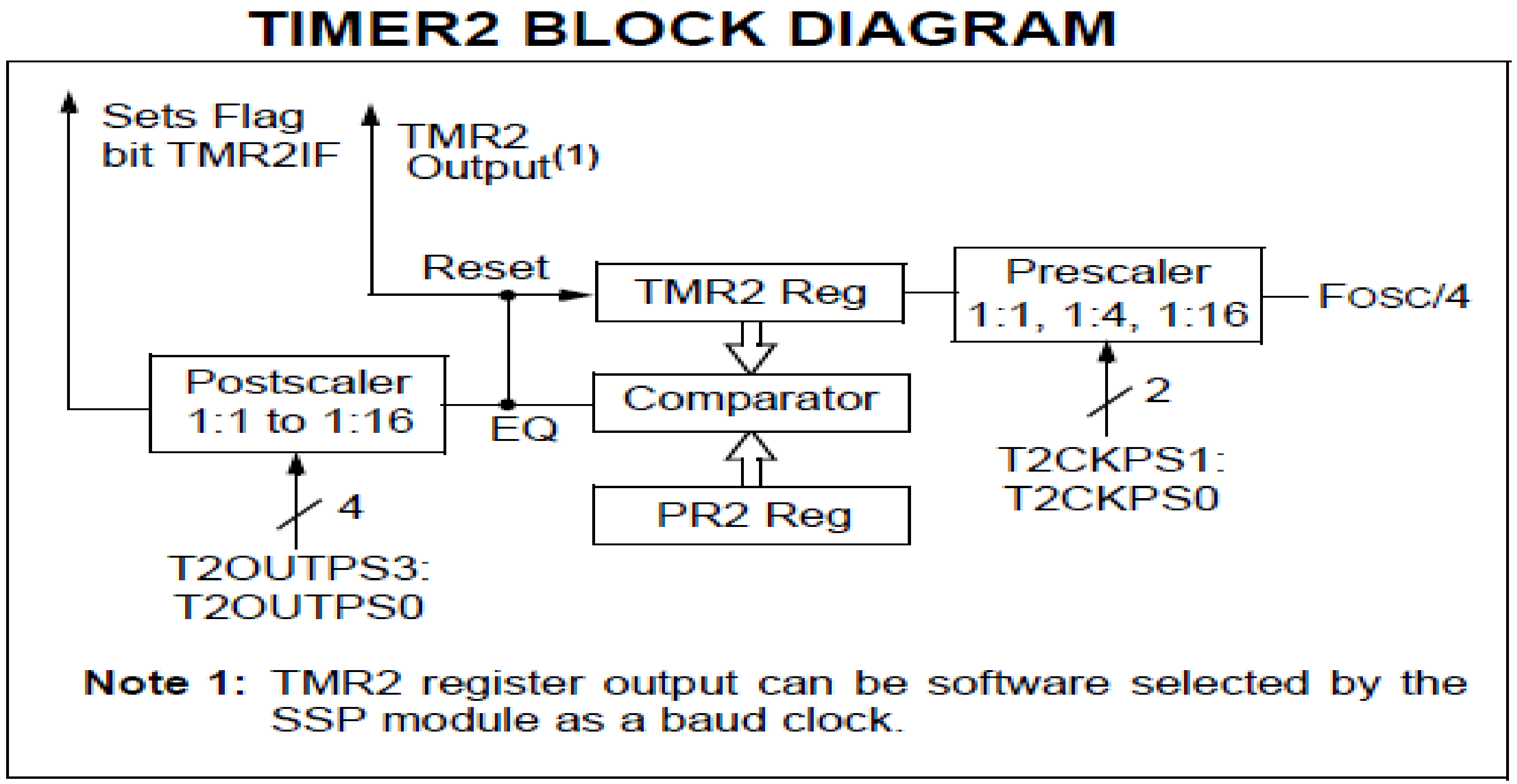
•8-bit timer/counter

•Readable and writable

•Software programmable prescaler/PostScaler upto 1:16

•Interrupt on overflow from FFh to 00h

TIMER



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

TIMER

Below are the steps for configuring and using the Timer2 for delay generation:

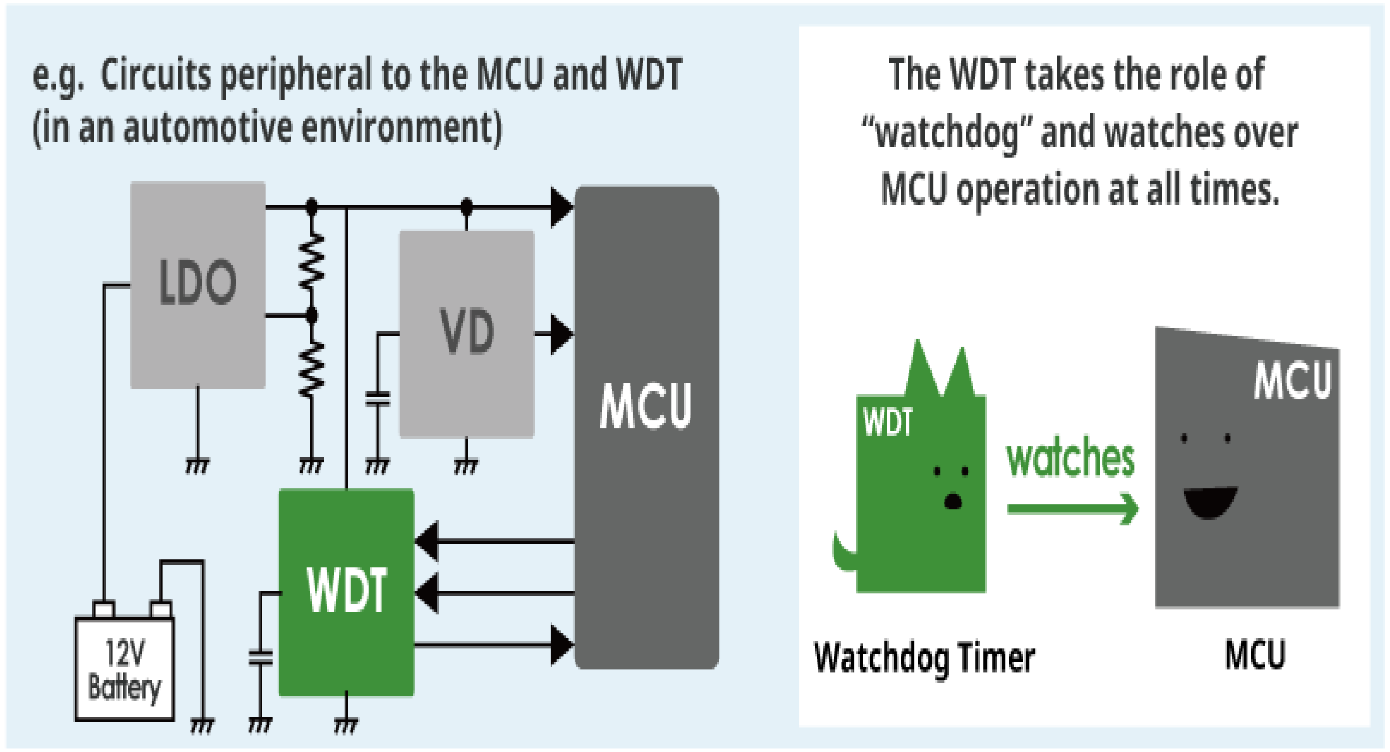
1. Calculate the Timer Count for the required delay.
2. Set the Prescaler bits in **T2CON** as per the delay calculations.
3. Load the timer value into **TMR2** register.
4. Enable the Timer2 Interrupt by setting **TMR2IE** bit
5. Enable the Global and Peripheral interrupts by setting **GIE** and **PIE** bits
6. Finally start the Timer2 by setting **TMR2ON** bit

Watchdog Timer (WDT)

A watchdog timer (WDT) is a timer that monitors microcontroller (MCU) programs to see if they are out of control or have stopped operating. It acts as a “watchdog” watching over MCU operation.
A microcontroller (MCU) is a compact processor for controlling electronic devices. Integrated into a wide variety of electronic devices, MCUs come pre-loaded with program software whose commands are used to control electronic devices.
This makes safeguarding normal MCU operation essential. Should the MCU program, for some reason, run out of control or stop running altogether, the electronic device may behave erratically, which in the worst case could cause damage or an accident.
To proactively prevent such incidents, **it falls to the role of the watchdog timer to constantly watch over the MCU to ensure it is operating normally.**
The watchdog timer function can be inside the MCU, but here we are introducing “external” watchdog timers, the safer kind.

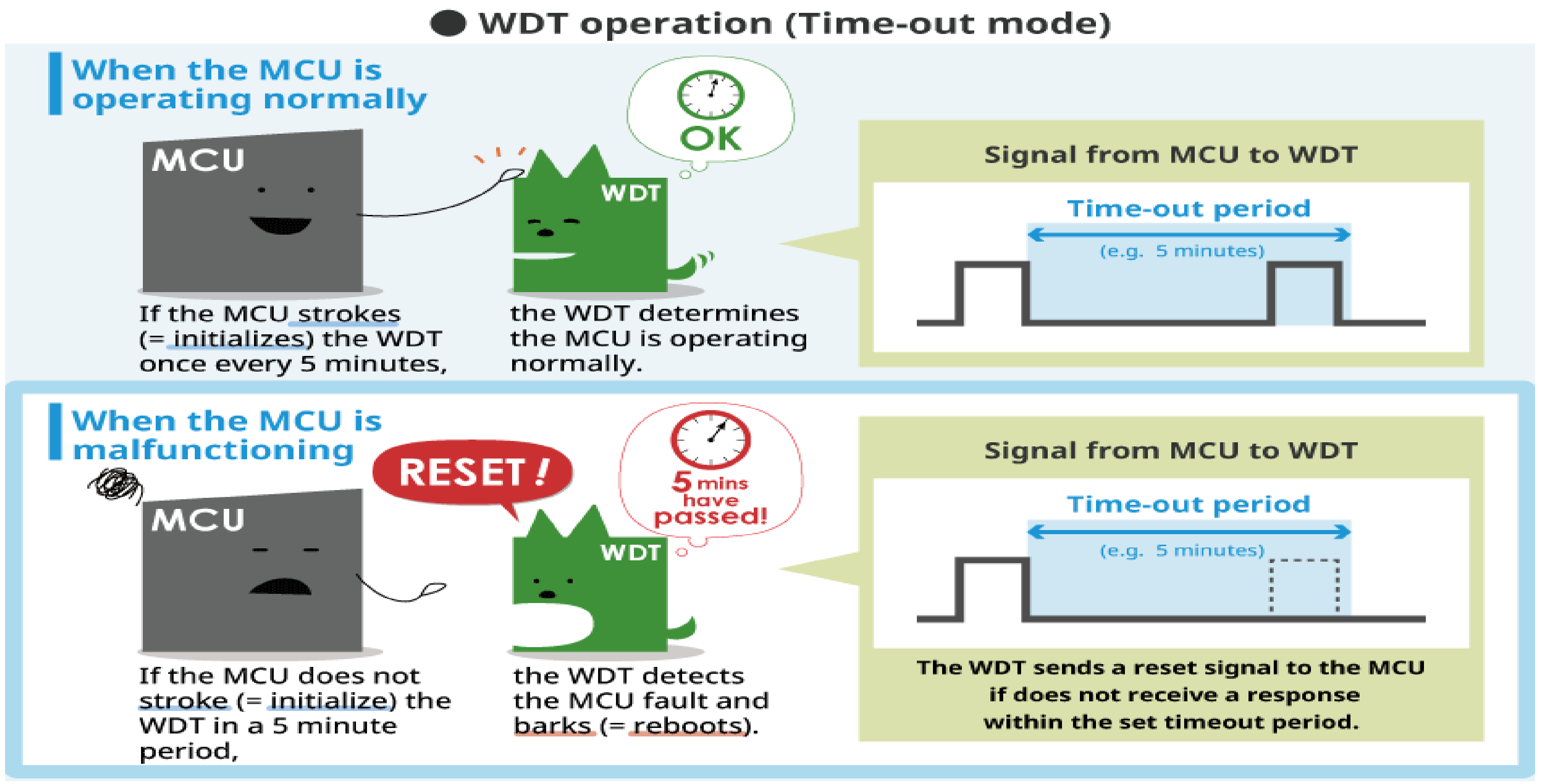
Watchdog Timer (WDT)

The watchdog timer communicates with the MCU at a set interval. If the MCU does not output a signal, outputs too many signals or outputs signals that differ from a predetermined pattern, the timer determines that the MCU is malfunctioning and sends a reset signal to the MCU.


e.g. Circuits peripheral to the MCU and WDT
(in an automotive environment)
LDO
VD
MCU
WDT
12V Battery
The WDT takes the role of "watchdog" and watches over MCU operation at all times.
WDT
watches
MCU
Watchdog Timer
MCU

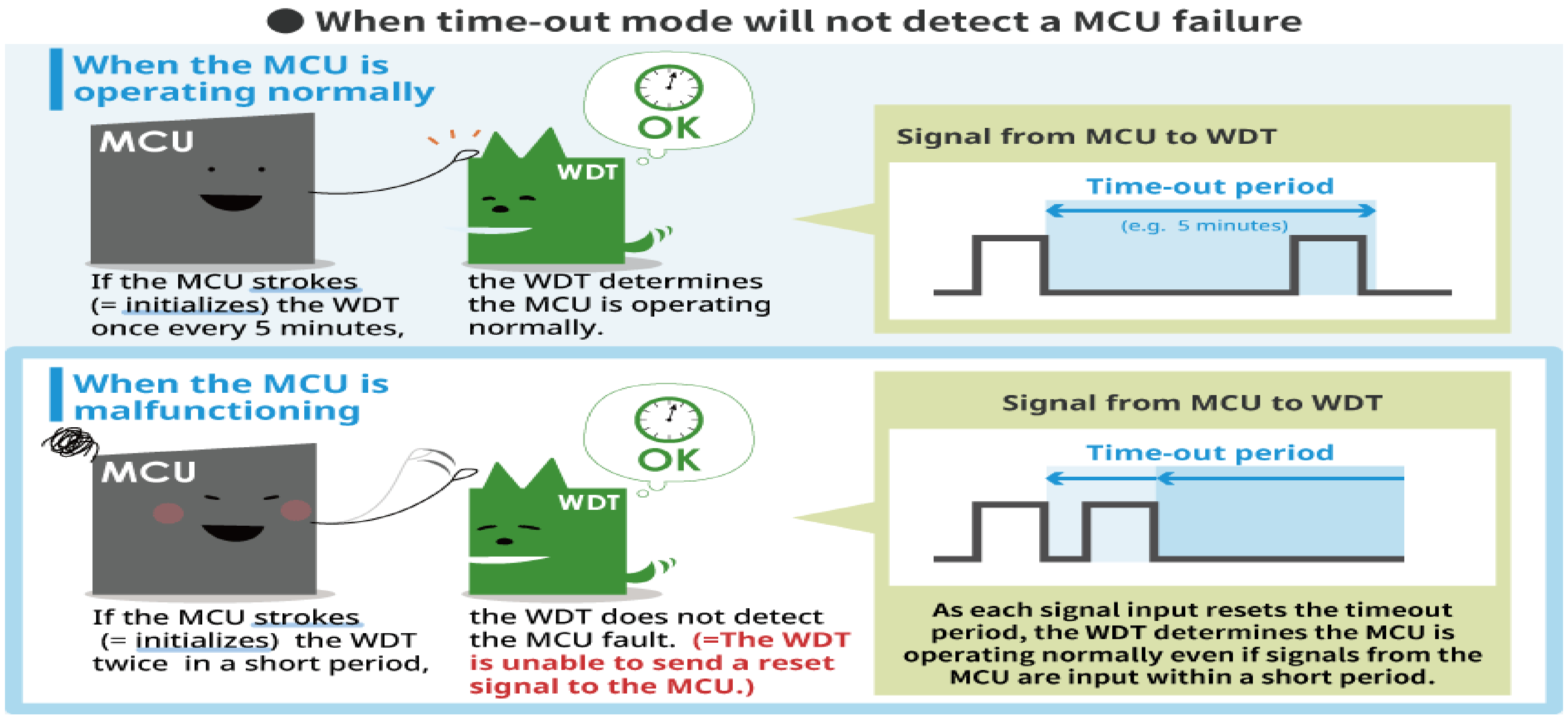
Watchdog Timer (WDT)

Time-out mode, the watchdog timer determines the MCU is malfunctioning and outputs a reset signal **if it does not receive a signal from the MCU within the set interval.**


WDT operation (Time-out mode)
When the MCU is operating normally
MCU
WDT
OK
If the MCU strokes (= initializes) the WDT once every 5 minutes,
the WDT determines the MCU is operating normally.
Signal from MCU to WDT
Time-out period
(e.g. 5 minutes)
When the MCU is malfunctioning
RESET !
MCU
WDT
5 mins have passed!
If the MCU does not stroke (= initialize) the WDT in a 5 minute period,
the WDT detects the MCU fault and barks (= reboots).
Signal from MCU to WDT
Time-out period
(e.g. 5 minutes)
The WDT sends a reset signal to the MCU if does not receive a response within the set timeout period.

Watchdog Timer (WDT)

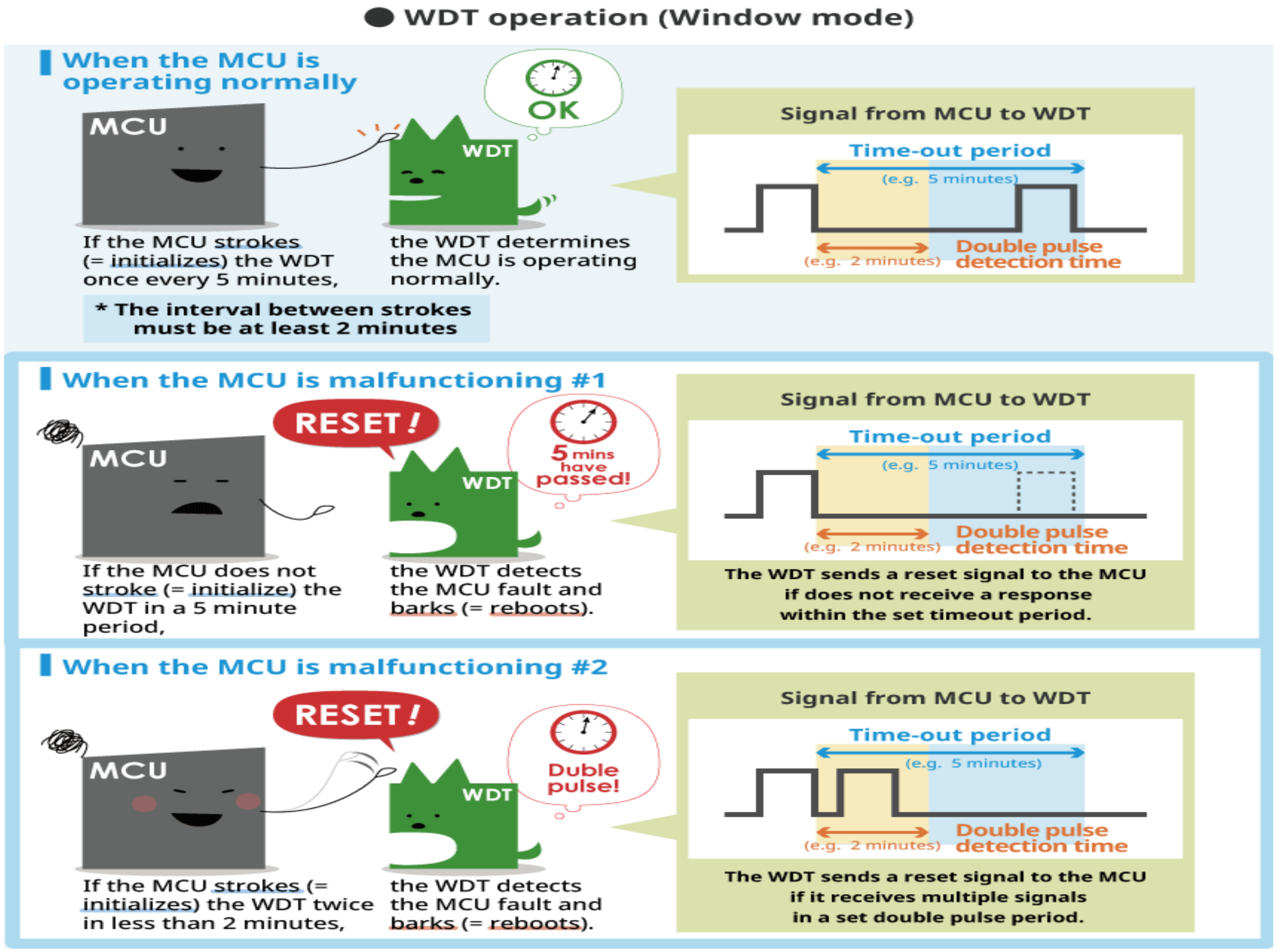
The time-out mode is a major WDT monitoring mode or method, but it sometimes fails to detect MCU faults.
In Time-out mode, the WDT will not detect an MCU fault if the MCU inputs multiple signals (= double pulse) in the set period.


● When time-out mode will not detect a MCU failure
When the MCU is operating normally
MCU
WDT
OK
If the MCU strokes (= initializes) the WDT once every 5 minutes,
the WDT determines the MCU is operating normally.
Signal from MCU to WDT
Time-out period
(e.g. 5 minutes)
When the MCU is malfunctioning
MCU
WDT
OK
If the MCU strokes (= initializes) the WDT twice in a short period,
the WDT does not detect the MCU fault. (=The WDT is unable to send a reset signal to the MCU.)
Signal from MCU to WDT
Time-out period
As each signal input resets the timeout period, the WDT determines the MCU is operating normally even if signals from the MCU are input within a short period.

Watchdog Timer (WDT)

The **window mode** enables more accurate detection of faults than the Time-out mode.
In window mode, the watchdog timer determines that the MCU is malfunctioning and outputs a reset signal **if it does not receive a signal, or receives multiple signals (= double pulse) from the MCU within the set interval.**
A window mode watchdog timer may be more suitable for applications such as automotive devices that require greater safety.

Watchdog Timer (WDT)


● WDT operation (Window mode)
When the MCU is operating normally
MCU
WDT
OK
If the MCU strokes (= initializes) the WDT once every 5 minutes,
the WDT determines the MCU is operating normally.
* The interval between strokes must be at least 2 minutes
Signal from MCU to WDT
Time-out period
(e.g. 5 minutes)
(e.g. 2 minutes)
Double pulse detection time
When the MCU is malfunctioning #1
RESET!
5 mins have passed!
If the MCU does not stroke (= initialize) the WDT in a 5 minute period,
the WDT detects the MCU fault and barks (= reboots).
The WDT sends a reset signal to the MCU if does not receive a response within the set timeout period.
When the MCU is malfunctioning #2
RESET!
Duble pulse!
If the MCU strokes (= initializes) the WDT twice in less than 2 minutes,
the WDT detects the MCU fault and barks (= reboots).
The WDT sends a reset signal to the MCU if it receives multiple signals in a set double pulse period.