Customer training workshop:
Device Configurator MCWDT

TRAVEO™ T2G CYT4BF series Microcontroller Training
V1.0.0 2023-7

Please read the Important notice and warnings at the end of this document.
Scope of work

- This document helps application developers understand how to use the MCWDT configuration of the Device Configurator as part of creating a ModusToolbox™ application
  - The Device Configurator is part of a collection of tools included with the ModusToolbox™ software. It provides a GUI to configure the target device.

- ModusToolbox™ tools package version
  - 3.1.0

- Device Configurator version
  - 4.10

- Device
  - This code example uses the TRAVEO™ T2G CYT4BF8CDS device.

- Board
  - The TRAVEO™ T2G KIT_T2G-B-H_LITE board is used for testing.
Introduction

– MCWDT has the following features:
  – Up to four MCWDTs, each supporting the following:
    – LFCLK (ILO0, ILO1, WCO, LPECO, or ECO) as the input clock source
    – Fault and device reset generation, if not serviced, within a configurable interval
    – Periodic interrupt/wakeup generation in Active, Sleep, and DeepSleep power modes
    – Three independent counters: two 16-bit counters and one 32-bit counter
    – Warning threshold generates an interrupt to request servicing
    – Window mode
    – Running and freezing timers during DeepSleep mode
    – Debug
Launch the Device Configurator

- **From Eclipse IDE**
  You can launch the Device Configurator by either of the following methods:
  
a. Right-click on the project in the Project Explorer and select **ModusToolbox™ > Device Configurator <version>**

b. Click the Device Configurator link in the Quick Panel
Device Configurator view for MCWDT config

- **Peripherals tab**
  - Set each MCWDT on the Peripheral tab
Quick start

- **To use the Device Configurator for MCWDT setting**
  - Launch the Device Configurator.
  - Use the various pull-down menus to configure signals. Refer to the descriptions in the Routing tab section for more details.
  - Save the file to generate the source code.
  - The Device Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location where you saved the *.modus file for non-IDE applications. That directory contains the necessary source (.c) and header (.h) files for the generated firmware, which uses the relevant driver APIs to configure the hardware.
  - Use the generated structures as input parameters for MCWDT configuration functions in your application.
Use case

- **Use case operation overview**
  - Use MCWDT Subcounter0, 1, and 2. Subcounter0 and 1 count up to the Warn limit (32000). Then an interrupt is generated and the count value restarts from zero. When Subcounter2 counts up to the setting value in the toggle bit, an interrupt is generated. The count value continuously counts up to a maximum value (0xFFFFFFFF). Then the count value restarts from zero.
  - When each interrupt generates, the LED is repeatedly turned on and off. Combinations of Subcounter and LED are shown in the following diagrams.
    - Subcounter0: LED3, Subcounter1: LED4, Subcounter2: LED5

- **Subcounter operation**

  ![Subcounter operation diagram](image-url)
Use case (contd.)

- CLK_LF frequency set to 32.8 kHz
  - Source clock uses ILO
- MCWDT0 Subcounter0, Subcounter1, and Subcounter2 operations with interrupt.
  - Subcounter0
    - Upper limit: 0xFFFF, Lower limit: 0, Warn limit: 32000, Warn limit action: Interrupt
    - Enabling Auto service (Count up to Warn limit, then restart at zero), Pausing in DeepSleep mode, Enabling debugger connection run
  - Subcounter1
    - Upper limit: 0xFFFF, Lower limit: 0, Warn limit: 32000, Warn limit action: Interrupt
    - Enabling Auto service (Count up to Warn limit, then restart at zero), Pausing in DeepSleep mode, Enabling debugger connection run
  - Subcounter2
    - Toggle bit position value: 15, Toggle bit position action: Interrupt
    - Pausing in DeepSleep mode, Enabling debugger connection run
- See the MCWDT_Interrupt application for operation
MCWDT configuration

Create project

1. Click New Application in the Quick Panel and open the Choose Board Support Package (BSP) window

2. Select TRAVEO™ BSPs and KIT_T2G-B-H_LITE

3. Click the Next button and open the Application window

4. In this use case, it changes to “MCWDT_training”

5. Click the Create button, and then start application creation
MCWDT configuration (contd.)

- **Launch the Device configurator**
  1. Select the **MCWDT_training** project.
  2. Click the Device configurator in the Quick Panel
  3. Then, open the Device Configurator window
MCWDT configuration (contd.)

- Configure MCWDT Subcounter0 and Subcounter1
  - Open the Peripherals tab and make the following settings
  - The values without description are default settings.

1) Select Multi-Counter Watchdog Timer (MCWDT)0

2) Fill the Name to “MCWDT_0”

- Set C0 Upper Limit to 0xFFFF
- Set C0 Warn limit to 32000
- Select C0 Warn Action to Interrupt
- Check all items (Auto service, DeepSleep pause, Debug run)

- Set C1 Upper Limit to 0xFFFF
- Set C1 Warn limit to 32000
- Select C1 Warn Action to Interrupt
- Check all items (Auto service, DeepSleep pause, Debug run)
MCWDT configuration (contd.)

- Configure MCWDT Subcounter2
  - The values without description are default settings.

Options:
- Set C2 Toggle Bit to 15
- Set C2 Action to Interrupt
- Check all items (DeepSleep pause, Debug run)
- Select MCWDT Core Select to CM7_0
MCWDT configuration (contd.)

- **System tab**
  - Set Low frequency clock (CLK_LF) on the System tab.
MCWDT configuration (contd.)

- Confirm configuration result
  - Check the configuration result in the “Code Preview” tab of the Device Configurator
MCWDT configuration (contd.)

- Close Device configurator
  - Click the Save button after completing all settings, then close the Device configurator

- If an Errors/Tasks message appears, resolve that according to the instructions as shown in the following screenshot
MCWDT configuration (contd.)

- **Configuration file**
  - The Device Configurator generates code into a "GeneratedSource" directory in your Eclipse IDE application, or in the same location you saved the *.modus file for non-IDE applications.
  - In this example, the following code is generated:

```
const cy_stc_mcwdt_config_t MCWDT_0_config = { 
  .cLowerLimit = 0u, 
  .cUpperLimit = 0xFFFFU, 
  .cWarnLimit = 32000U, 
  .cWarnAction = CY_MCMDT_ACTION.NONE, 
  .cUpperAction = CY_MCMDT_ACTION.NONE, 
  .cDebugRun = CY_MCMDT_ENABLE, 
  .cSleepDeepPause = CY_MCMDT_ENABLE, 
  .cCpuSleepBit = 15U, 
  .cAction = CY_MCMDT_CNT2_ACTION_INT, 
  .cSleepDeepPause = CY_MCMDT_DISABLE, 
  .cDebugRun = CY_MCMDT_DISABLE, 
  .coreSelect = CY_MCMDT_PAUSED_BY_DSPMP.CopyTo, 
};
```

- In this example, the following code is generated:

```
cyclg_peripherals.h
```
The structure generated by the Device Configurator can be used by implementing the following function in your application code.

1) Double-click the main.c file

Open the main.c edit window
Implementation (contd.)

- Add MCWDT initialization enable function

You can use the "MCWDT_0_HW" (MCWDT#0) to specify the hardware.
Implementation (contd.)

Initialize the MCWDT_0

- Call the `Cy_MCWDT_Init()` function to initialize the MCWDT_0.
  - The MCWDT configuration that is set in `MCWDT_0_config` can be changed using Device Configurator
  - Subcounter0/1 is set to generate interrupt when the counter matches to warning threshold value (=32000)
  - Subcounter2 is set to generate an interrupt when bit15 of the counter toggles

Set up the interrupt handler

- Call the `Cy_SysInt_Init()` function to set up `ISR_MCWDT_0()` as the ISR.
  - The ISR reads the status of the interrupt by calling the `Cy_MCWDT_GetInterruptStatusMasked()` function
  - Then it controls each user LED by calling `Cy_GPIO_Inv()` depends on cause of interrupt
    - If the cause is MCWDT Subcounter0, user LED3 (P5.0) is toggled.
    - If the cause is MCWDT Subcounter1, user LED4 (P5.1) is toggled.
    - If the cause is MCWDT Subcounter2, user LED5 (P5.2) is toggled.
  - Clear interrupt by calling the `Cy_MCWDT_ClearInterrupt()` function.
Implementation (contd.)

Enable the MCWDT_0 counters

- Unlocks the MCWDT_0 configuration registers by calling the `Cy_MCWDT_Unlock()` function.
- Set MCWDT_0 interrupt mask register by calling the `Cy_MCWDT_SetInterruptMask()` function.
- Enables MCWDT_0 counters by calling the `Cy_MCWDT_Enable()` function.
- Locks out configuration changes to the MCWDT_0 registers by calling the `Cy_MCWDT_Lock()` function.
References

Datasheet
- CYT4BF datasheet 32-bit Arm® Cortex®-M7 microcontroller TRAVEO™ T2G family

Architecture Technical reference manual
- TRAVEO™ T2G automotive body controller high family architecture technical reference manual

Registers Technical reference manual
- TRAVEO™ T2G Automotive body controller high registers technical reference manual

PDL/HAL
- PDL
- HAL

Training
- TRAVEO™ T2G Training
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>ECN</th>
<th>Submission Date</th>
<th>Description of Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>7933288</td>
<td>2023/07/27</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
**Important notice and warnings**

All referenced product or service names and trademarks are the property of their respective owners.

**IMPORTANT NOTICE**

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie").

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer’s compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer’s products and any use of the product of Infineon Technologies in customer’s applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer’s technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

**WARNINGS**

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies’ products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.