Customer training workshop: Device Configurator_DMA

TRAVEO™ T2G CYT4BF series Microcontroller Training
V1.0.0 2022-12

Scope of work

- This document helps application developers understand how to use the Device Configurator DMA as part of creating a ModusToolbox™ (MTB) application
  - The Device Configurator DMA is part of a collection of tools included with the MTB software. It provides configuration of the DMA channel and transaction descriptors.

- ModusToolbox™ tools package version: 3.0.0
- Device Configurator version: 4.0
- Device
  - The TRAVEO™ T2G CYT4BFBCH device is used in this code example.
- Board
  - The TRAVEO™ T2G KIT_T2G-B-H_EVK board is used for testing.

Introduction

- **TRAVEO™ T2G has two types of DMA :**
  - Peripheral DMA (P-DMA)
  - Memory DMA (M-DMA)
- **P-DMA has the following features:**
  - Focuses on achieving low latency for a large number of channels.
  - Focuses on peripheral-to-memory and memory-to-peripheral data transfers (but it can also perform memory-to-memory data transfers).
  - Uses a single data transfer engine that is shared by all channels.
  - A descriptor specifies the following data transfer specifications:
    - The source and destination address locations and the size of the transfer.
    - The actions of a channel; for example, generation of output triggers and interrupts.
    - Data transfer types: single, 1D, 2D, or CRC as defined in the descriptor structure. These types essentially define the address sequences generated for source and destination
  - It is called DMA DataWire in this document

Introduction (contd.)

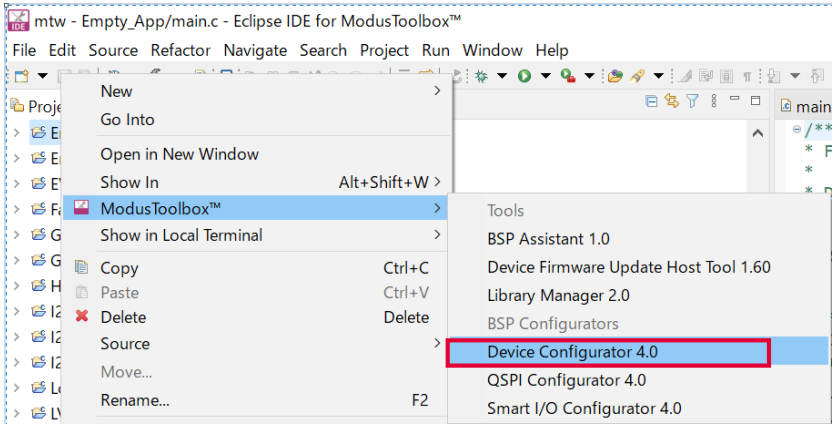
- **M-DMA has the following features:**
  - Focuses on achieving high memory bandwidth for a small number of channels.
  - Focuses on memory-to-memory data transfers (but it can also perform peripheral-to-memory and memory-to-peripheral data transfers).
  - Uses a dedicated data transfer engine for each channel.
  - A descriptor specifies the following data transfer specifications:
    - The source and destination address locations and the size of the transfer.
    - The actions of a channel; for example, generation of output triggers and interrupts.
    - Data transfer types can be single, 1D, or 2D as defined in the descriptor structure. These types essentially define the address sequences generated for source and destination. 1D and 2D transfers are used for “scatter gather” and other useful transfer operations.
    - It is called DMA Controller in this document

Launch the Device Configurator

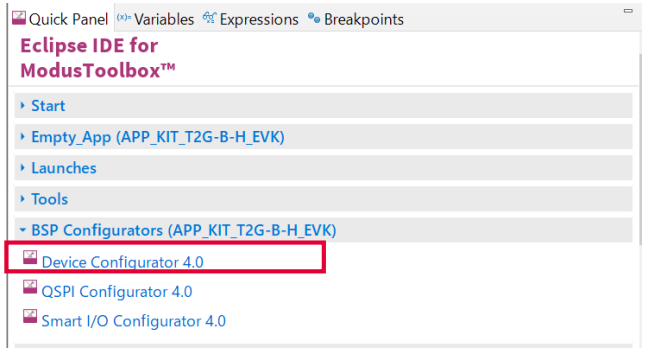
From Eclipse IDE

- You can launch the Device Configurator by either method a) or b)

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 DMA config view

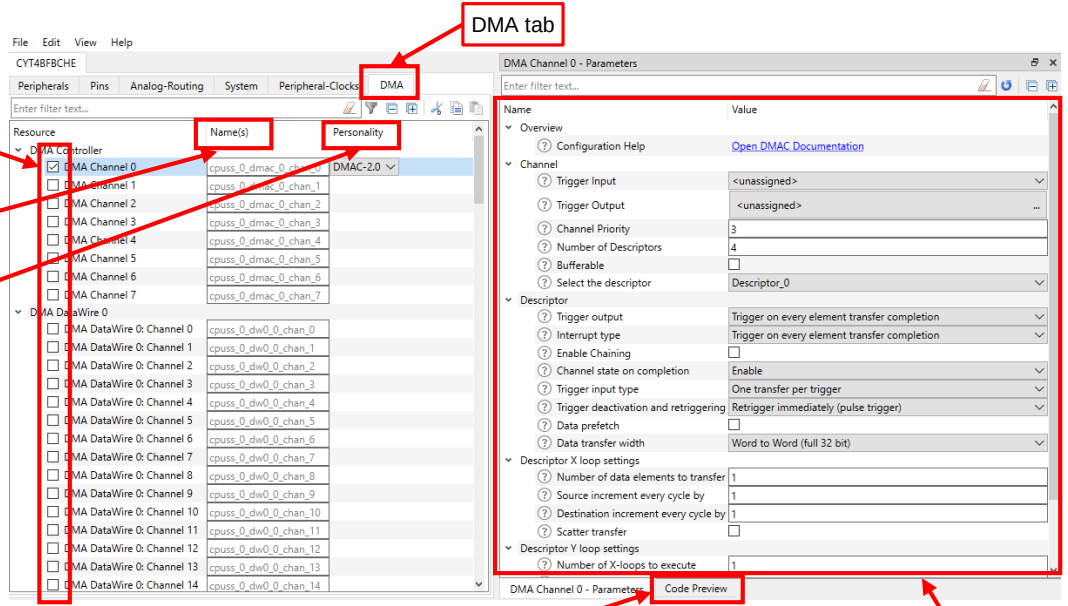
› Device Configurator DMA

- On the DMA tab, you can select and configure each DMA channel

All available DMA are shown in an expandable tree.
You can check the DMA channels that should be enabled setting.

This tab allows you to enter Names for the resource.

Personality shows the selected Personality, where applicable.



Code Preview is a read-only preview of the code that will be generated for the currently selected resource when you save the *.modus file. As you update configuration options, the Code Preview pane updates the code shown. This code will be written to the appropriate file(s) located in the GeneratedSource folder of your application.

You can set parameters for the specified DMA

Quick start

To use the DMA Device Configurator for DMA setting

- Launch the Device Configurator.
- Check the DMA channels to use
- Select the parameter from the various pull-down menus to configure signals.
- The DMA 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. 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 DMA functions in your application.

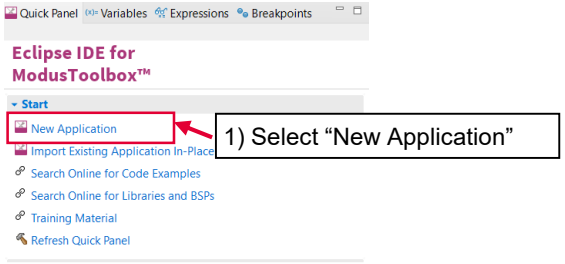
Use case

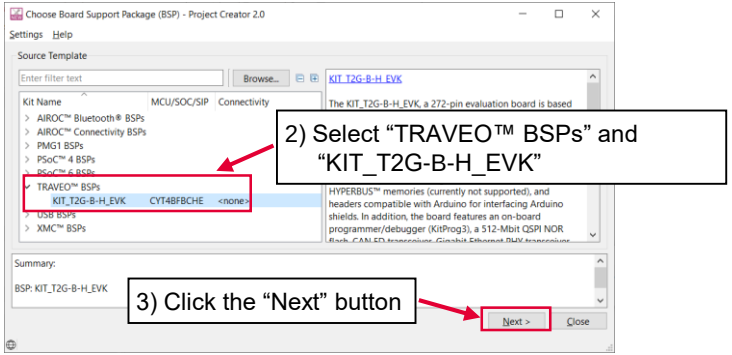
- This use case uses two DMA channels: txDma and rxDma
- txDma initiates transfer by a SCB2 transmit request event, and rxDma initiates transfer by a SCB8 receive event
- txDma
  - DMA DataWire#1 channel 20
  - In the first event, data of tx_buff (A) in RAM is transferred to the SCB2 Tx FIFO. In the second event, data of tx_buff (B) in RAM is transferred to the SCB2 Tx FIFO. In the third event, data of tx_buff (A) in RAM is transferred again.
  - Transfer size: Source byte / Destination word
  - Descriptor type: 1D transfer
  - Transfer count: 12
- rxDma
  - DMA DataWire#1 channel 33
  - In first receive event, data of SCB8 RX FIFO is transferred to the rx_buff (A) in RAM. In second event, data of SCB8 RX FIFO is transferred to the data of rx_buff (B) in RAM. In the third event, the data is transferred to rx_buff (A) in RAM again.
  - Transfer size: Source word / Destination byte
  - Descriptor type: 1D transfer
  - Transfer count: 12
- See the SCB_SPI_Master_DMA application for operation

DMA configuration

› Create project

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



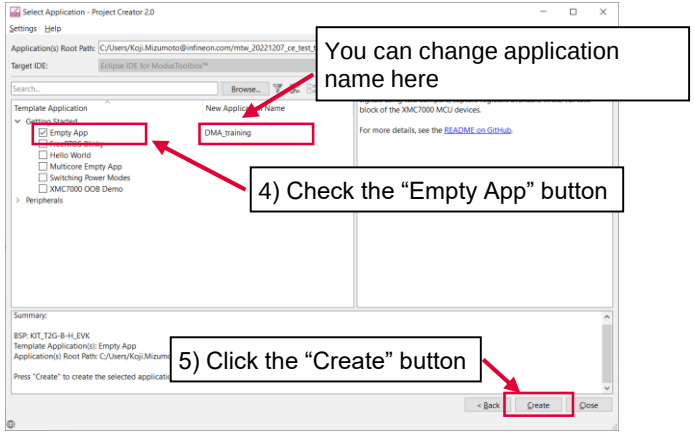


2) Select **TRAVEO™ BSPs** and **KIT_T2G-B-H_EVK**

3) Click the **Next** button and open the Application window

4) Check the **Empty App** option.
In this use case, change the application name to DMA_**training**.

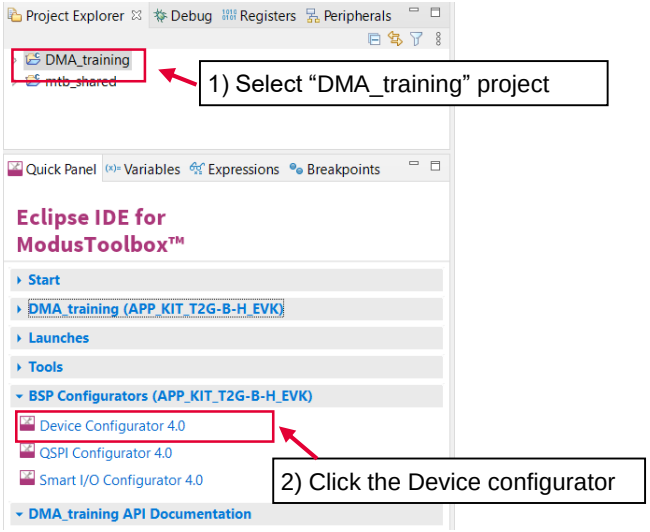
5) Click the **Create** button to start application creation

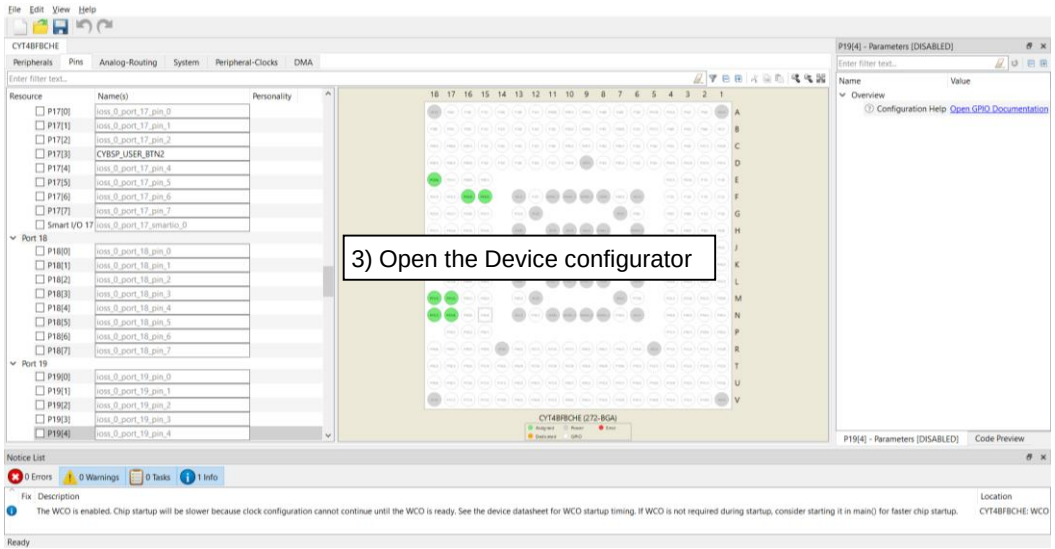


DMA configuration (contd.)

› Launch the Device Configurator

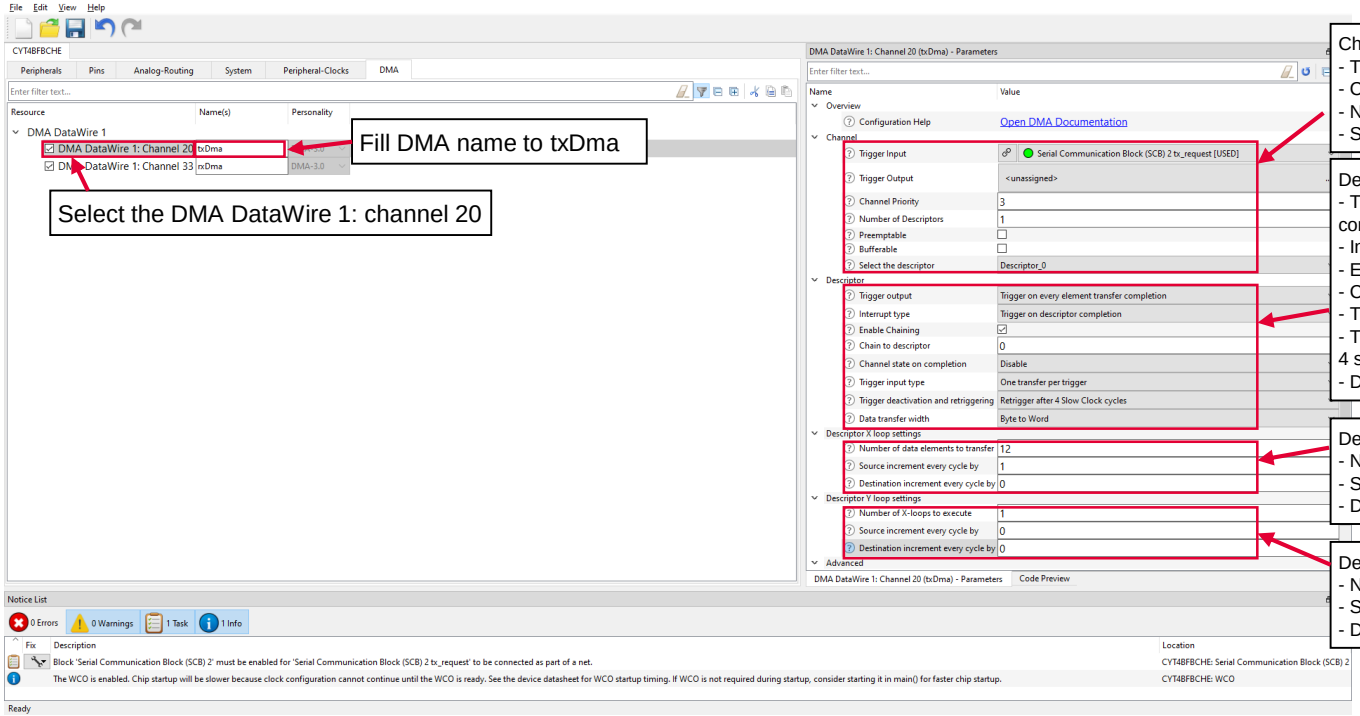
1) Select the **DMA_training** project.
2) Click the Device Configurator in the Quick Panel
3) Then, open the Device configurator window





DMA configuration (contd.)

› Configure txDMA



Channel
- Trigger Input: SCB 2 tx_request
- Channel Priority: 3
- Number of Descriptors: 1
- Select the descriptors: Descriptor_0

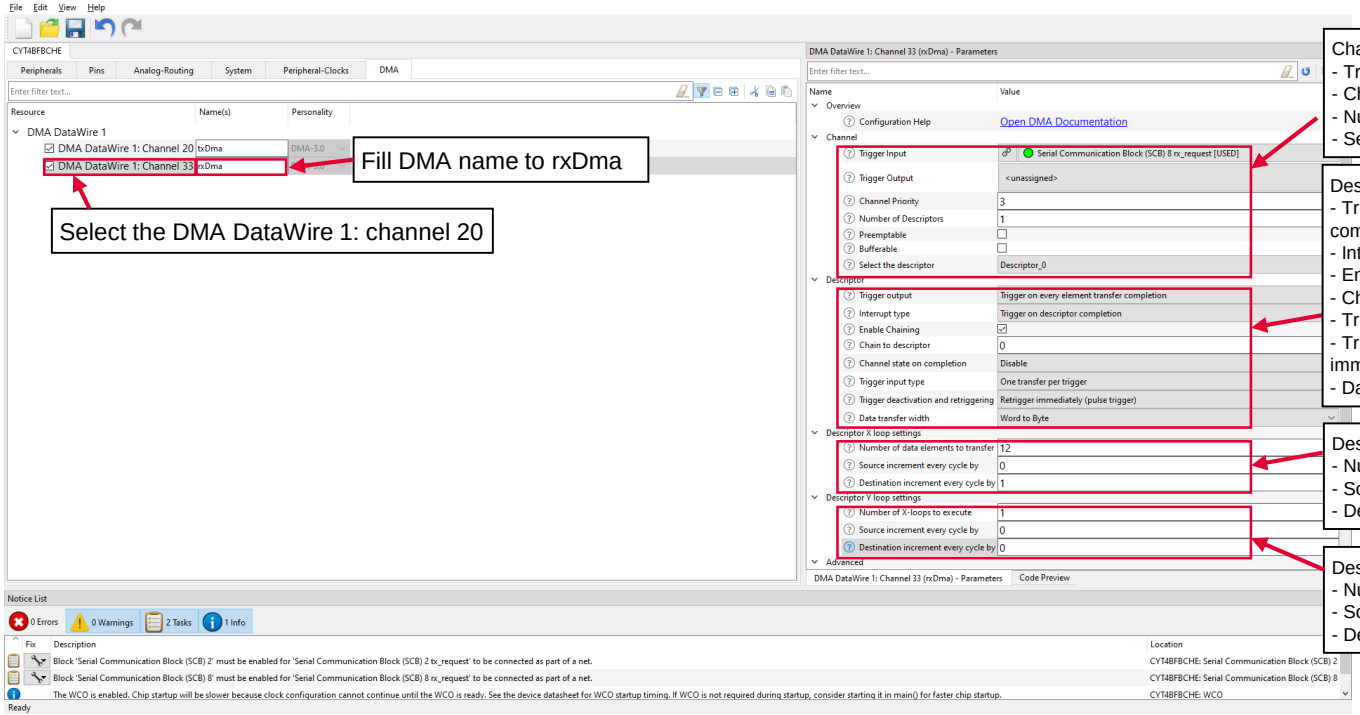
Descriptor
- Trigger output: Trigger on every element transfer completion
- Interrupt type: Trigger on descriptor completion
- Enable Chaining: On
- Channel state on completion: Disable
- Trigger input type: One transfer per trigger
- Trigger deactivation and retriggering: Retrigger after 4 slow Clock cycles
- Data transfer width: Byte to Word

Descriptor X loop settings
- Number of data elements to transfer: 12
- Source increment every cycle by: 1
- Destination increment every cycle by: 0

Descriptor Y loop settings
- Number of X-loops to execute: 1
- Source increment every cycle by: 0
- Destination increment every cycle by: 0

DMA configuration (contd.)

› Configure rxDMA



Channel
- Trigger Input: SCB 8 rx_request
- Channel Priority: 3
- Number of Descriptors: 1
- Select the descriptors: Descriptor_0

Descriptor
- Trigger output: Trigger on every element transfer completion
- Interrupt type: Trigger on descriptor completion
- Enable Chaining: On
- Channel state on completion: Disable
- Trigger input type: One transfer per trigger
- Trigger deactivation and retriggering: Retrigger immediately (pulse trigger)
- Data transfer width: Word to Byte

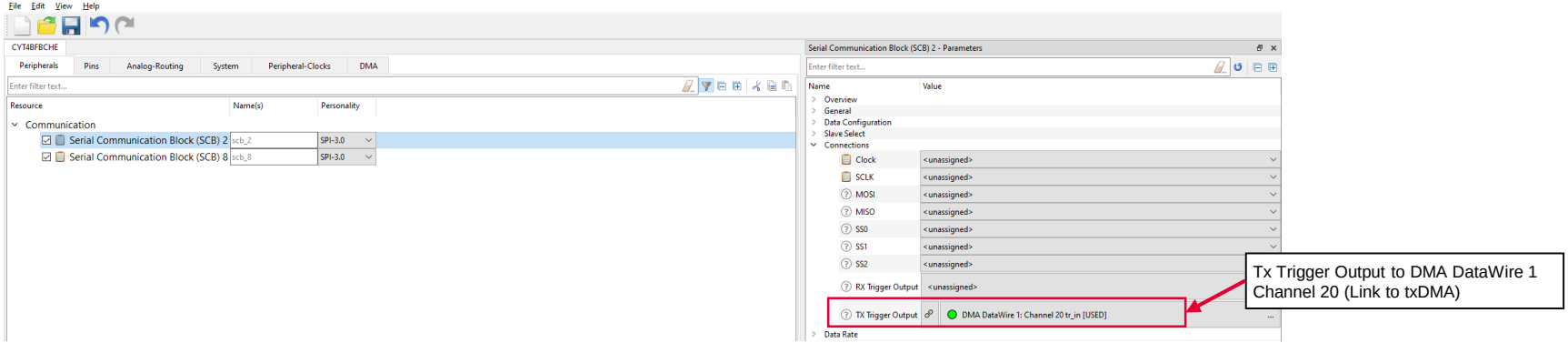
Descriptor X loop settings
- Number of data elements to transfer: 12
- Source increment every cycle by: 1
- Destination increment every cycle by: 0

Descriptor Y loop settings
- Number of X-loops to execute: 1
- Source increment every cycle by: 0
- Destination increment every cycle by: 0

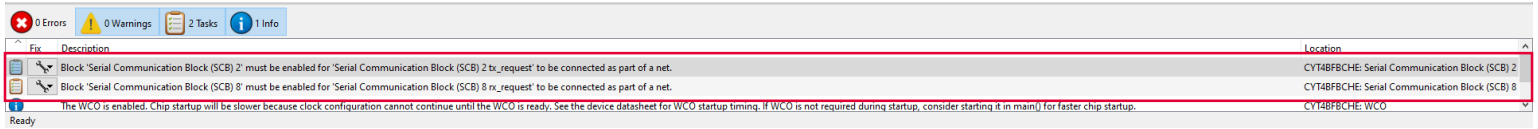
DMA configuration (contd.)

› Configure peripheral for DMA trigger

- When the trigger is selected from the peripheral, it is automatically linked to the target peripheral



- When the input trigger is set in the DMA configuration, peripheral configuration is required. If the peripheral is not valid, it will be displayed as follows.



DMA configuration (contd.)

› Confirm configuration result

- You can check the configuration result in the “Code Preview” tab of the Device Configurator

txDma

Code Preview

Enter search text...

```
#define txDma_HW DW1
#define txDma_CHANNEL 20U
#define txDma_IRQ cpuss_interrupts_dw1_20_IRQn

const cy_stc_dma_descriptor_config_t txDma_Descriptor_0_config =
{
    .retrigger = CY_DMA_RETRIG_4CYC,
    .interruptType = CY_DMA_DESCR,
    .triggerOutType = CY_DMA_1ELEMENT,
    .channelState = CY_DMA_CHANNEL_DISABLED,
    .triggerInType = CY_DMA_1ELEMENT,
    .dataSize = CY_DMA_BYTE,
    .srcTransferSize = CY_DMA_TRANSFER_SIZE_DATA,
    .dstTransferSize = CY_DMA_TRANSFER_SIZE_WORD,
    .descriptorType = CY_DMA_1D_TRANSFER,
    .srcAddress = NULL,
    .dstAddress = NULL,
    .srcXincrement = 1,
    .dstXincrement = 0,
    .xCount = 12,
    .srcYincrement = 0,
    .dstYincrement = 0,
    .yCount = 1,
    .nextDescriptor = &txDma_Descriptor_0,
};
CY_ALIGN(32) cy_stc_dma_descriptor_t txDma_Descriptor_0 =
{
    .ctl = 0UL,
    .src = 0UL,
    .dst = 0UL,
    .xCtl = 0UL,
    .yCtl = 0UL,
    .nextPtr = 0UL,
};
const cy_stc_dma_channel_config_t txDma_channelConfig =
{
    .descriptor = &txDma_Descriptor_0
```

DMA DataWire 1: Channel 20 (txDma) - Parameters | Code Preview

rxDma

Code Preview

Enter search text...

```
#define rxDma_HW DW1
#define rxDma_CHANNEL 33U
#define rxDma_IRQ cpuss_interrupts_dw1_33_IRQn

const cy_stc_dma_descriptor_config_t rxDma_Descriptor_0_config =
{
    .retrigger = CY_DMA_RETRIG_IM,
    .interruptType = CY_DMA_DESCR,
    .triggerOutType = CY_DMA_1ELEMENT,
    .channelState = CY_DMA_CHANNEL_DISABLED,
    .triggerInType = CY_DMA_1ELEMENT,
    .dataSize = CY_DMA_BYTE,
    .srcTransferSize = CY_DMA_TRANSFER_SIZE_WORD,
    .dstTransferSize = CY_DMA_TRANSFER_SIZE_DATA,
    .descriptorType = CY_DMA_1D_TRANSFER,
    .srcAddress = NULL,
    .dstAddress = NULL,
    .srcXincrement = 0,
    .dstXincrement = 1,
    .xCount = 12,
    .srcYincrement = 0,
    .dstYincrement = 0,
    .yCount = 1,
    .nextDescriptor = &rxDma_Descriptor_0,
};
CY_ALIGN(32) cy_stc_dma_descriptor_t rxDma_Descriptor_0 =
{
    .ctl = 0UL,
    .src = 0UL,
    .dst = 0UL,
    .xCtl = 0UL,
    .yCtl = 0UL,
    .nextPtr = 0UL,
};
const cy_stc_dma_channel_config_t rxDma_channelConfig =
{
    .descriptor = &rxDma_Descriptor_0
```

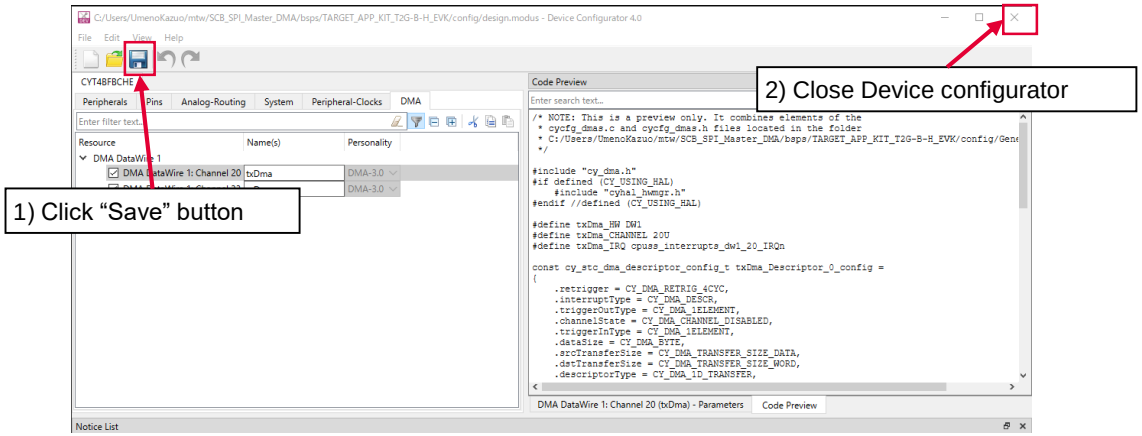
DMA DataWire 1: Channel 33 (rxDma) - Parameters | Code Preview

Code preview tab

DMA configuration (contd.)

› Close Device configurator

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

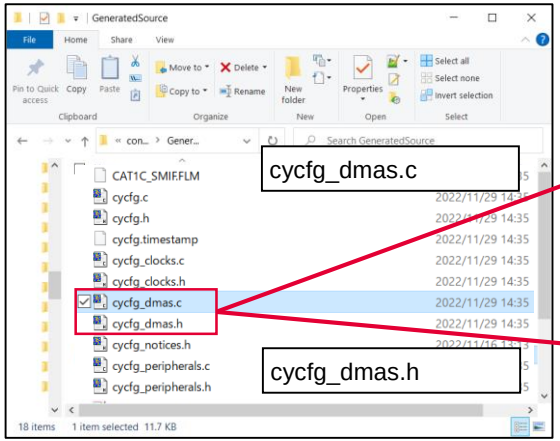


- If an **Errors/Tasks** message appears, it should be resolved according to the instructions

DMA configuration (contd.)

Configuration file

- The DMA 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.
- This example has the following code:

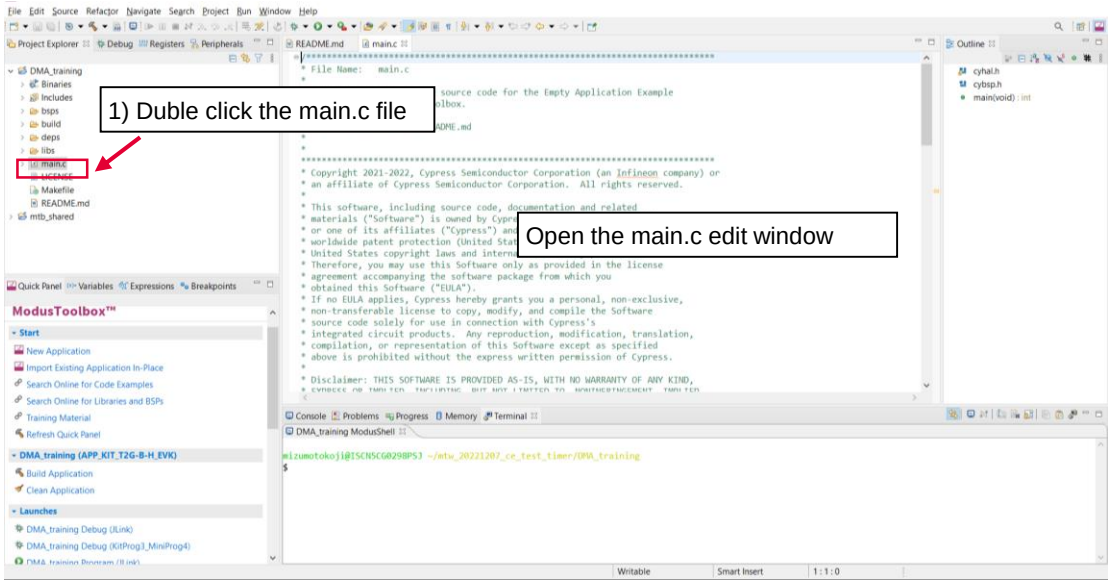



```
#include "cycfg_dmas.h"
const cy_stc_dma_descriptor_config_t txDma_Descriptor_0_config =
{
    .retrigger = CY_DMA_RETRIG_4CYC,
    .interruptType = CY_DMA_DESCR,
    .triggerOutType = CY_DMA_1ELEMENT,
    .channelState = CY_DMA_CHANNEL_DISABLED,
    .triggerInType = CY_DMA_1ELEMENT,
    .dataSize = CY_DMA_BYTE,
    .srcTransferSize = CY_DMA_TRANSFER_SIZE_DATA,
    .dstTransferSize = CY_DMA_TRANSFER_SIZE_WORD,
    .descriptorType = CY_DMA_1D_TRANSFER,
    .srcAddress = NULL,
    .dstAddress = NULL,
    .srcXincrement =
    .dstXincrement =
    .xCount = 12,
    .srcYincrement =
    .dstYincrement =
    .yCount = 1,
    .nextDescriptor
};
CY_ALIGN(32) cy_stc_
{
    .ctl = 0UL,
    .src = 0UL,
    .dst = 0UL,
    .xCtl = 0UL,
    .yCtl = 0UL,
    .nextPtr = 0UL,
};
```

```
#if !defined(CYCFG_DMAS_H)
#define CYCFG_DMAS_H

#include "cycfg_notices.h"
#include "cy_dma.h"
#if defined (CY_USING_HAL)
    #include "cyhal_hwmgr.h"
#endif //defined (CY_USING_HAL)

#if defined(__cplusplus)
extern "C" {
#endif

#define txDma_ENABLED 1U
#define txDma_HW DW1
#define txDma_CHANNEL 20U
#define txDma_IRQ cpuss_interrupts_dw1_20_IRQn
#define rxDma_ENABLED 1U
#define rxDma_HW DW1
#define rxDma_CHANNEL 33U
#define rxDma_IRQ cpuss_interrupts_dw1_33_IRQn
```

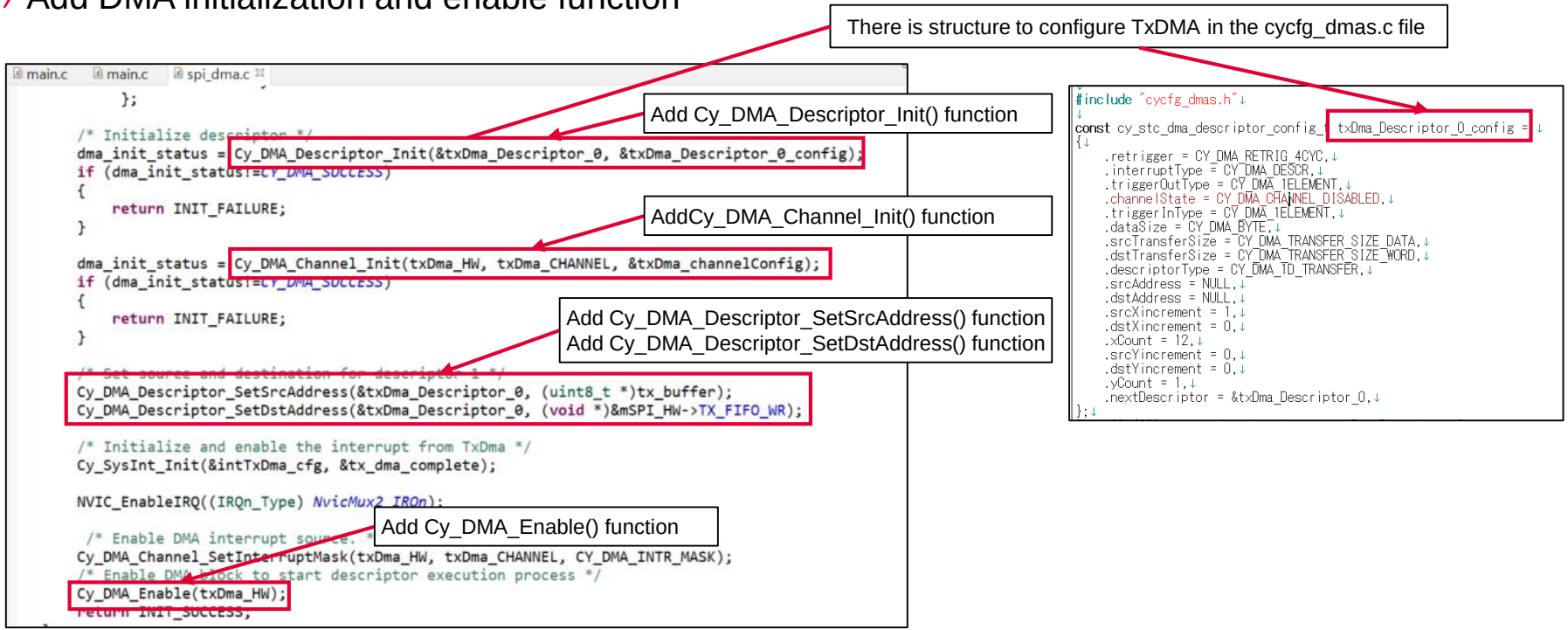
Implementation

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



Implementation (contd.)

› Add DMA initialization and enable function



Implementation (contd.)

DMA initialization

- Call the ***Cy_DMA_Descriptor_Init()*** function to initialize the DMA descriptor
  - Configure DMA with the parameters in ***txDma_Descriptor_0*** and ***txDma_Descriptor_0_config*** structure for txDma
  - Configure DMA with the parameters in ***rxDma_Descriptor_0*** and ***rxDma_Descriptor_0_config*** structure for rxDma
- Call the ***Cy_DMA_Channel_Init()*** function to initialize the DMA channel
  - Configure DMA with the parameters in ***txDma_channelConfig*** structure for txDma
  - Configure DMA with the parameters in ***rxDma_channelConfig*** structure for rxDma
- Call the ***Cy_DMA_Descriptor_SetSrcAddress()*** and ***Cy_DMA_Descriptor_SetDstAddress()*** function to set the source and destination address of the DMA transfer
  - Source address sets to RAM (*tx_buffer*), and destination address sets to TX FIFO of SCB2
  - Source address sets to RX FIFO of SCB8, and destination address sets to RAM (*rx_buffer*)

DMA enable:

- Call the ***Cy_DMA_Enable()*** function to enable DMA

Data transfer

- Data transfer is initiated by serial communication send and receive events

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

Revision	ECN	Submission Date	Description of Change
**	7847414	2022/12/13	Initial release

Important notice and warnings

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

Edition 2022-12
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Go to:
www.infineon.com/support

Document reference
002-36715 Rev. **

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.