Customer training workshop: Smart I/O configurator

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

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 Smart I/O Configurator as part of creating a ModusToolbox™ (MTB) application
  - The Smart I/O Configurator is part of a collection of tools included with the MTB software. It provides a GUI to configure the Smart I/O.

- MTB version: 3.0.0
- Smart I/O Configurator version: 4.0
- Device
  - The TRAVEO™ T2G CYT4BFBCCH device is used in this code example.
- Board
  - The TRAVEO™ T2G KIT_T2G-B-H_EVK board is used for testing.
Smart I/O has the following features:

- Smart I/O provides the ability to perform Boolean functions in the I/O signal path
- Path 1: Implement self-contained logic functions that directly operate on port I/O signals
- Path 2: Implement self-contained logic functions that operate on HSIOM signals
- Path 3: Operate on and modify HSIOM output signals and route the modified signals to port I/O signals
- Path 4: Operate on and modify port I/O signals and route the modified signals to HSIOM input signals
- Smart I/O can be useful when the application involves simple logic operations and routing of the signal coming from or going to the I/O pin. No CPU is required for these operations.
Introduction (contd.)

Smart I/O has the following features:

- There are several areas in the GUI to configure signals: Chip, I/O, Data Unit (DU), and LUT
- Inputs to the chip from the I/O port can be logically operated upon before being routed to the peripheral blocks and connectivity of the chip. Likewise, outputs from the peripheral blocks and internal connectivity of the chip can be logically operated upon before being routed to the I/O port.
- The programmable logic fabric of the Smart I/O can be purely combinatorial or registered with a choice of clock selection
- Each path can be selectively bypassed if certain routes are not required by the fabric
- Each Smart I/O is associated with a particular I/O port and consumes the port entirely
- If the Smart I/O is not enabled, then the Smart I/O functionality for that port is bypassed, which means that each chip terminal is routed directly to the corresponding I/O terminal.
- See the Smart I/O chapter in the Architecture technical reference manual for Smart I/O details
Launch the Smart I/O configurator

› **From Eclipse IDE**
  - You can launch the Smart I/O configurator by following either of these methods:
    a) Right-click on the project in the Project Explorer and select ModusToolbox™ > Smart I/O Configurator <version>
    b) Click the *Smart I/O Configurator* link in the Quick Panel
Launch the Smart I/O configurator (contd.)

› **From Device Configurator**

1) Open the Device configurator

2) On the Pins tab, enable the Smart I/O resource

3) On the Parameters tab, click the **Launch Smart I/O Configurator** button
Smart I/O configurator

Smart I/O configurator view – Routing tab

Port:
This parameter allows selecting a port that supports Smart I/O.

Clock:
Selects the clock source used to drive all sequential logic in the block.

When the Show routing matrix button is on, you may click on the switches in the fabric to make input connections to the LUTs.

Chip configuration:
Each has two pull-down menus; one to select the direction of the signal and the other to select the connection.

I/O configuration:
Each has a pull-down menu to select the direction of the signal.

Data Unit input configuration:
The Data Unit configuration section contains three pull-down menus to select inputs. You can click the link icon to access the tab.

LUT input configuration:
Each LUT configuration section contains three pull-down menus to select inputs from Chip, I/O, and DUT resources. You can click the link icon to access the tab.
Smart I/O configurator (contd.)

› Smart I/O configurator view – Data Unit tab

- When the DU in the Routing tab is configured to accept an input other than a Constant 0, the corresponding Data Unit configuration tab will appear.

**OpCode:**
Defines the Data Unit operation. Each opcode performs a unique function that can be controlled using the DU trigger inputs TR0, TR1, and TR2.

Refer to the pseudo verilog code in the Details window and also to the Functional Description section for more information.

**DATA0:**
Defines the DU DATA0 register source. This value is often used as the initial/reset value that is loaded into the DU working register when TR0 signal is high.

**DATA1:**
Defines the DU DATA1 register source. This value is often used as the comparison value that gets applied to the DU working register.

**Size:**
Defines the bit size operation to be performed by the data unit. Valid range is from 1 to 8 bits.

**Details:**
Pseudo Verilog code is shown for selected opcodes.
Smart I/O configurator (contd.)

› Smart I/O configurator view – LUT tab

- When a LUT in the Routing tab is configured to accept an input, the corresponding LUT configuration tab will appear.

Mode:
Defines the LUT mode.

Set Bit:
Defines the lookup truth table of the 3-to-1 LUT. The state on the three inputs (input 0, 1, and 2) are translated to an output value according to this truth table. Click to select 0 or 1.

If a LUT is used, all three inputs to the LUT must be designated. For example, even if a LUT is used to accept a single source as its input, all three inputs must accept that same signal. The lookup truth table should then be designed such that it only changes the output value when all three inputs satisfy the same condition (for example, it outputs a logic 1 only when the inputs are all 0).
Quick start

› To use the Smart I/O configurator for Smart I/O setting
  – Launch the Smart I/O 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 source code.
  – The Smart I/O 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 Smart I/O functions in your application.
Use case

› Implement a reset detection/stability circuitry on the Smart I/O

› In this use case, the circuitry has two enable signals: pin_rst_enable and gpio_rst_enable.

› The pin_rst_enable is an enable signal from an external circuitry and the gpio_rst_enable is an enable control signal by software. When both signals are enabled, the circuitry is active.

› The rst_in_n is an external reset input with Active high and rst_out_n is a reset output with Active high. The circuitry monitors rst_in_n. When rst_in_n is activated for a specific number of continuous cycles (the operation clock is input continuously for 128 cycles), the rst_out_n is output.

› The source 50 MHz clock is divided by 50 to 1 MHz. Then, count 1 μs is multiplied by 128 cycles and the time of reset activation or release is approximately 128 μs.

› Refer to the application note Smart I/O Usage Setup in Traveo II Family for details
Use case (contd.)

Connection and functional logic of each LUT [7:4] and DU in this circuitry

- In this example, four LUTs and one DU are used
  - LUT4 is used to generate the activation signal of this circuitry from two enable signals (pin_rst_enable and gpio_rst_en)
  - LUT6 and LUT7 are used to monitor the rst_in_n state and to start the counter of the DU
  - LUT5 detects the stabilization wait completion and outputs rst_out_n
  - DU is used to generate reset stability wait time, and the Tr_out of LUT5 is output synchronously by the gated output mode

- In this use case, I/O [7:5], Chip 4 is used as input or output signals
  - I/O 7 is used for “RST_IN_N”
  - I/O 6 is used for “PIN_RST_ENABLE”
  - I/O 5 is used for “RST_OUT_N”
  - Chip 4 is used for “GPIO_RST_ENABLE”
Smart I/O 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 “Smart_IO_training”.

5) Click the Create button to start application creation
Smart I/O configuration (contd.)

- **Launch the Device Configurator**
  1. Select the “Smart_IO_training” project.
  2. Click the Device Configurator in the Quick Panel.
  3. Then, open the Device configurator window.
Smart I/O configuration (contd.)

› Configure GPIO
  - In this use case, Smart I/O 13 (Port 13) is used
  - Pin assignment is as follows:
    - P13[7] (I/O 7) is used for “RST_IN_N”
    - P13[6] (I/O 6) is used for “PIN_RST_ENABLE”
    - P13[5] (I/O 5) is used for “RST_OUT_N”
    - P13[4] (Chip 4) is used for “GPIO_RST_ENABLE”
  - Configure GPIO from the pull-down list

1) Enable the signal and Smart I/O 13
2) Fill in pin name

Port13_4
 Drive mode: Strong Drive, input buffer off
 Initial Drive state: Low

Port13_5
 Drive mode: Strong Drive, input buffer off
 Initial Drive state: Low
 Digital Output: Smart I/O io_out[5]

Port13_6
 Drive mode: Digital input, input buffer on
 Initial Drive state: Low
 Digital Input: Smart I/O io_in[6]

Port13_7
 Drive mode: Digital input, input buffer on
 Initial Drive state: Low
 Digital Input: Smart I/O io_in[7]
Launch the Smart I/O configurator

1) Select Smart I/O 13
2) Click **Launch Smart I/O Configurator**
3) Click **Yes** and then open the Smart I/O Configurator window
Configure pin assignment and clock

- Pin assignment is as follows:
  - I/O 7 is used for “RST_IN_N”, set to Input (Async)
  - I/O 6 is used for “PIN_RST_ENABLE”, set to Input (Async)
  - I/O 5 is used for “RST_OUT_N”, set to Output
  - Chip 4 is used for “GPIO_RST_ENABLE”, set to none
  - I/O [3:0] is set to bypass. These pins do not use the Smart I/O fabric
- Clock setting
  - Set Clock to “Peripheral clock divider (Active)”
  - Set Clock divider to “16 bit Divider 2 clk”
Smart I/O configuration (contd.)

› Configure connection of LUTs and DU

- LUTs and DU connect to:

<table>
<thead>
<tr>
<th>LUTx/DU</th>
<th>TR0</th>
<th>TR1</th>
<th>TR2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT4</td>
<td>Chip 4</td>
<td>Chip 4</td>
<td>I/O 6</td>
</tr>
<tr>
<td>LUT5</td>
<td>DU out</td>
<td>LUT5 out</td>
<td>LUT4 out</td>
</tr>
<tr>
<td>LUT6</td>
<td>LUT7 out</td>
<td>LUT7 out</td>
<td>LUT7 out</td>
</tr>
<tr>
<td>LUT7</td>
<td>LUT4 out</td>
<td>LUT5 out</td>
<td>I/O 7</td>
</tr>
<tr>
<td>DU</td>
<td>LUT6 out</td>
<td>LUT7 out</td>
<td>“0”</td>
</tr>
</tbody>
</table>

- You can set LUTx and DU using the pull-down lists in LUT and DU box.

- If the routing matrix is shown, you can also click on the switches in the fabric to make input connections.

- All three inputs to the LUT must be designated.
  If the input signal has only one source (for example, LUT6), the same signal inputs to three inputs of LUT.

Click the switch

Set the LUTx and DU
Smart I/O configuration (contd.)

› Configure DU

- Select the Data Unit tab to configure DU

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Increment and wrap</td>
</tr>
<tr>
<td>DATA0</td>
<td>Constant 0</td>
</tr>
<tr>
<td>DATA1</td>
<td>Data Register</td>
</tr>
<tr>
<td>Register value</td>
<td>127 (Decimal)</td>
</tr>
<tr>
<td>Size</td>
<td>8-bit size/width operand</td>
</tr>
</tbody>
</table>

- When the DU in the “Routing” tab is configured to accept an input other than a Constant 0, the corresponding “Data Unit” tab will appear.
Smart I/O configuration (contd.)

› Configure LUT

- Select the Data Unit tab to configure LUTx

<table>
<thead>
<tr>
<th>LUT</th>
<th>Parameters</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUT4</td>
<td>Mode</td>
<td>TR2 gated, combination output</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>0x80</td>
</tr>
<tr>
<td>LUT5</td>
<td>Mode</td>
<td>Sequential (gated) output</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>0x60</td>
</tr>
<tr>
<td>LUT6</td>
<td>Mode</td>
<td>Combination output</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>0x7F</td>
</tr>
<tr>
<td>LUT7</td>
<td>Mode</td>
<td>TR2 gated, combination output</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>0x28</td>
</tr>
</tbody>
</table>

- When an LUT in the Routing tab is configured to accept an input, the corresponding LUT configuration tab will appear.
Smart I/O configuration (contd.)

› Close Smart I/O configurator

- Click the Save button after completing all settings and close the Smart I/O configurator

1) Click “Save” button

2) Close Smart I/O configurator

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

Click

WARNING: Invalid DU connection. DU TRD is sourced from LUT [3] but the LUT is not enabled to drive it.
Smart I/O configuration (contd.)

› Clock configuration
  - Peripheral clock configuration in the Device Configurator for Smart I/O resources
  - Divides the source clock (50 MHz) by 50

Select Peripheral-Clock tab
Divider set to 50
Assign the peripheral clock to Smart I/O

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

Code preview tab
Smart I/O configuration (contd.)

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

1) Click “Save” button
2) Close Device configurator
Smart I/O configuration (contd.)

› Configuration file

- The Smart I/O 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:

```c
#include "cycfg_pins.h"

void cycfg_pins(void) {
    // Code generated by Smart I/O Configurator
}
```

In this example, the following code is generated:

- `cycfg_pins.c`
- `cycfg_pins.h`
Implementation

This section describes how to implement the configured Smart I/O. This example will implement Smart I/O configuration in the Smart_IO_training project.

- Open main.c in Smart_IO_training project

1) Double click the main.c file

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

› Add include file

```c
#include "cyhal.h"
#include "cybsp.h"
#include "cy_pdi.h"
#include "cycfg.h"
```

Add include file in the main.c

```c
#define RST_DISABLE (Bu)
#define RST_ENABLE (3u)
```

Add macros for GPIO setting
Implementation (contd.)

› Add Smart I/O initialization and enable function

- Add Smart I/O initialization function
- Add Smart I/O enable function
- Add GPIO write function to enable “GPIO_RST_ENABLE” signal

There is structure to configure Smart I/O in the cycfg_pins.c file
Implementation (contd.)

**Smart I/O initialization**

› Call the `Cy_SmartIO_Init()` function to configure Smart I/O
  - Initialize the Smart I/O 13 include LUT4-7 and DU setting

**Smart I/O enable**

› Call the `Cy_SmartIO_Enable()` function to enable Smart I/O

**GPIO port write**

› Call the `Cy_GPIO_Write()` function to set GPIO
  - It is used to enable the “GPIO_RST_ENABLE” signal
  - “GPIO_RST_ENABLE” is configured as “`GPIO_RST_ENABLE_PORT` (= Port 13)” and “`GPIO_RST_ENABLE_PIN` (= 4 pin)” in `cycfg_pins.h` file
Compiling and programming

1. Connect to power and USB cable
2. Use Eclipse IDE for ModusToolbox™ software for compiling and programming
3. Compile
   a) Select the target application project in the Project Explorer
   b) In the Quick Panel, scroll down, and click “Build Application” in Smart_IO_training (APP KIT_T2G-B-H_EVK)

4. Programming
   a) Select the target application project in the Project Explorer
   b) In the Quick Panel, scroll down, and click “Smart_IO_training Program (KitProg3_MiniProg4)” in Launches
Run and test

1. To run this use case, use jumper wires, as shown below, on the board

<table>
<thead>
<tr>
<th>Name</th>
<th>GPIO pin</th>
<th>Connection</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIN_RST_ENABLE</td>
<td>Port13.6</td>
<td>Port 17.3</td>
<td>USER_BTN2</td>
</tr>
<tr>
<td>RST_IN_N</td>
<td>Port13.7</td>
<td>Port 21.4</td>
<td>USER_BTN1</td>
</tr>
<tr>
<td>RST_OUT_N</td>
<td>Port13.5</td>
<td>Monitor</td>
<td></td>
</tr>
</tbody>
</table>
2. After programming, the application starts automatically
3. Press USER BTN1 (RST_IN_N), you can observe the following waveform

4. When press USER BTN1 (RST_IN_N) while pressing USER BTN2 (PIN_RST_ENABLE), the output does not change.
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

Application note
› Smart I/O Usage Setup in Traveo II Family
## 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>7844899</td>
<td>2022-12-07</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.