Please note that Cypress is an Infineon Technologies Company.

The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content

The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers

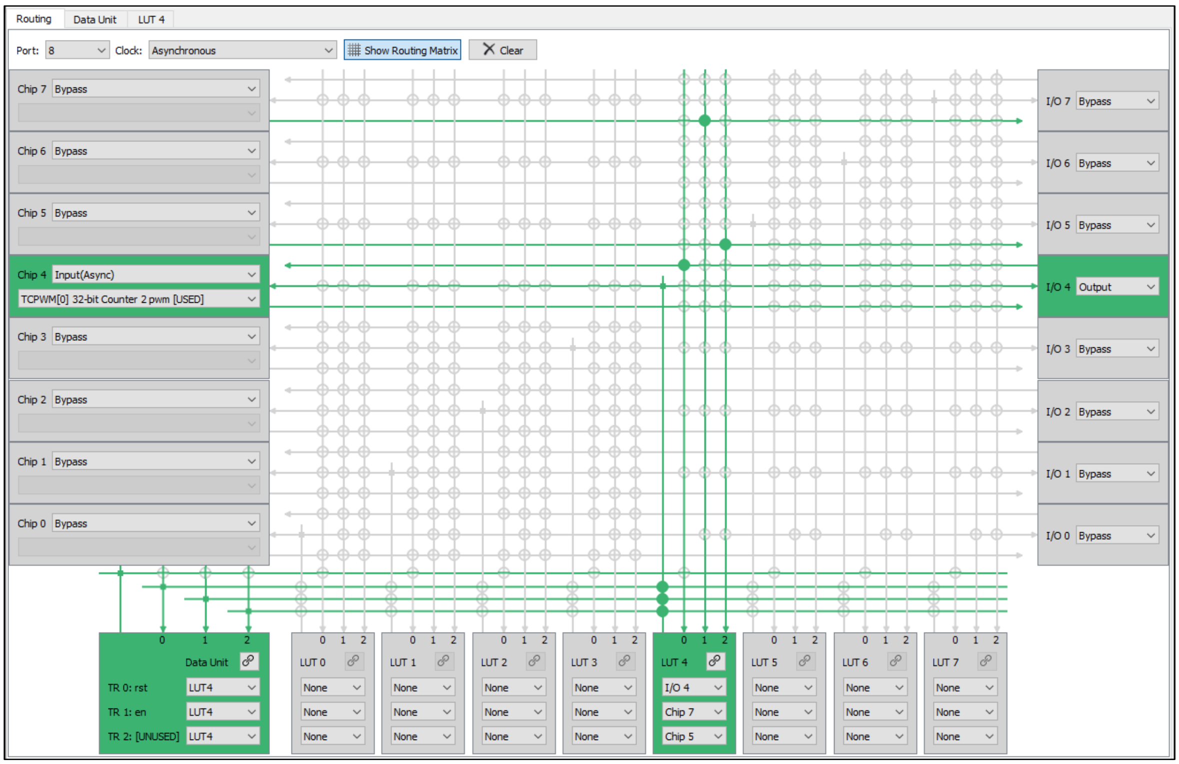
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

ModusToolbox®
Smart I/O™ Configurator Guide

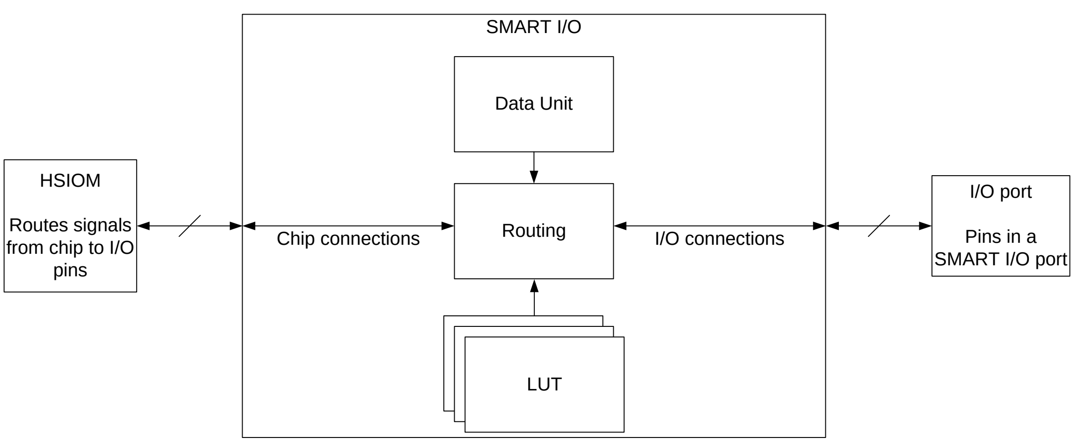
Version 2.20

Overview

The Smart I/O Configurator is part of a collection of tools included with the ModusToolbox software. It provides a GUI to configure the Smart I/O.



The Smart I/O block adds programmable logic to an I/O port. It is positioned in the signal path between the high-speed I/O matrix (HSIOM) and the I/O port. The HSIOM multiplexes the output signals from fixed-function peripherals and the CPU to a specific port pin and vice-versa. The Smart I/O block is placed on this signal path, acting as a bridge that can process signals between port pins and the HSIOM, as shown in the following diagram. For more information about the Smart I/O block, refer to the device *Technical Reference Manual*.



There are several areas in the GUI to configure signals: Chip, I/O, Data Unit, 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. The functionality is completely user-defined, and 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.

Note Bypassed means each chip terminal is routed directly to the corresponding I/O terminal.

Launch the Smart I/O Configurator

You can launch the Smart I/O Configurator various ways: as a stand-alone tool, from the Eclipse IDE for ModusToolbox, from the Device Configurator or from the command line. The Smart I/O Configurator GUI contains menus and tabs to configure I/O settings. The command line tool has various options. Then, you can either use the generated source with an Eclipse IDE application, or use it in any software environment you choose.

As a Stand-Alone Tool

You can launch the Smart I/O Configurator as a stand-alone tool without the Eclipse IDE. By default, it is installed here:

<install_dir>/ModusToolbox/tools_<version>/usbdev-configurator

On Windows, launch the tool from the **Start** menu.

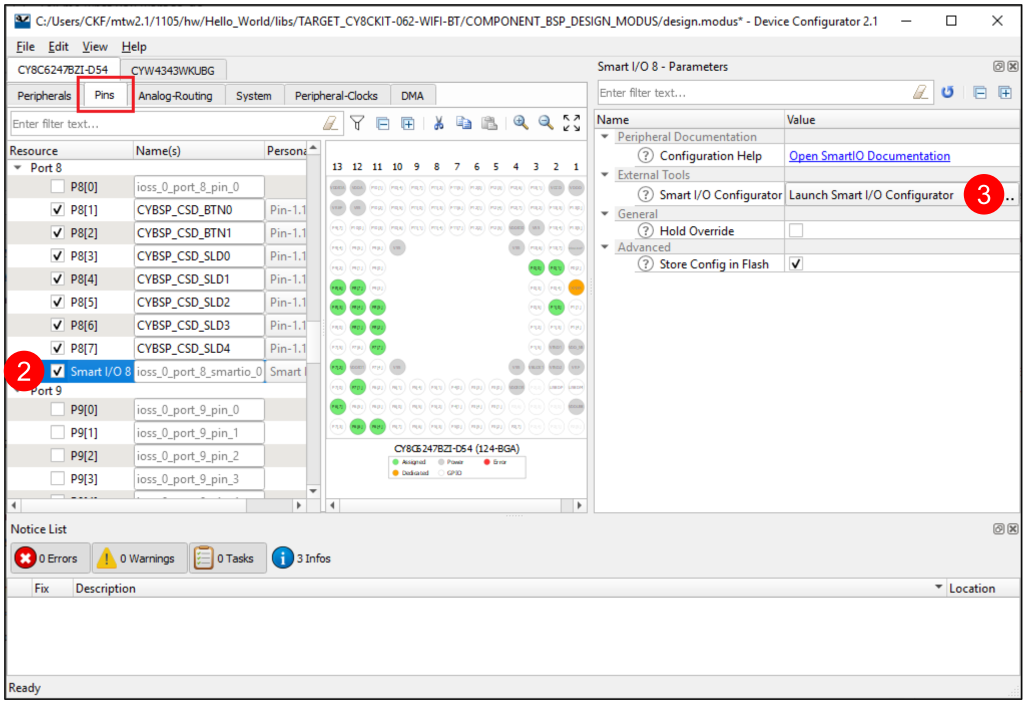
For other operating systems, navigate to the install location and run the executable.

When opened this way, the Smart I/O Configurator GUI opens without any information. You must open an existing **.modus* file or create a new one for the application in which you want to configure Smart I/O.

From the Device Configurator

To launch the Smart I/O Configurator GUI from the Device Configurator:

1. Open the Device Configurator. See the Device Configurator User Guide for details.
2. On the **Pins** tab, enable the Smart I/O resource.
3. On the **Parameters** tab, click the **Launch Smart I/O Configurator** button.



From the Eclipse IDE

To run the Smart I/O Configurator GUI from an application within the Eclipse IDE, right-click on the project and select **ModusToolbox > Smart I/O Configurator**. You can also open the Smart I/O Configurator GUI by clicking the link in the Eclipse IDE Quick Panel:

This opens the Smart I/O Configurator GUI using the application's *design.modus* file that is shared by the Device Configurator. This file contains all the required hardware configuration information about the device for the application. When you save updates to the *design.modus* file, the tool generates/updates source code in the “GeneratedSource” folder. Eclipse IDE applications use the *design.modus* file and generated source code in future application builds.

From the Command Line

You can run the configurator from the command line. However, there are only a few reasons to do this in practice. The primary use case would be to re-generate source code based on the latest configuration settings. This would often be part of an overall build script for the entire application.

For information about command line options, run the configurator using the `-h` option.

Quick Start

The Smart I/O is a port-wide resource that has a close relationship with the port to which it is dedicated. Hence the connectivity of peripherals to the Smart I/O will differ based on which device and which port is used. To use the Smart I/O:

1. Launch the Smart I/O Configurator.
2. Use the various pull-down menus to configure signals. Refer to the descriptions in the Routing Tab section for more details.
3. 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.

4. Use the generated structures as input parameters for Smart I/O functions in your application.

   For more information, refer to the Peripheral Driver Library (PDL) API Reference documentation.

GUI Description

Menus

File

- **New** – Creates a new *.modus file.
- **Open** – Opens and loads an existing *.modus file.
- **Close** – Closes the current file (leaves the application open).
- **Save** – Saves changes to the file. If the file does not exist, the Save file dialog opens.
- **Save As…** – Saves changes to a new file.
- **Open in System Explorer** – This opens your computer's file explorer tool to the folder that contains the *design.modus* file.
- **Change Library** – Opens a dialog to select a different library (devicesupport.xml) file used for resource Personalities. See the Device Configurator Guide for more details.
- **Change Devices** – This opens the Change Devices dialog to select an alternate MCU device and/or add/remove Companion devices. See the Device Configurator Guide for more details.
- **Recent Files** – Lists up to five recently opened *.modus files, which you can select to re-open.
- **Exit** – Closes the configurator.

View

- **Notice List** – Shows/hides the **Notice List** pane, which contains any errors, warnings, tasks, and information notes. See the *Device Configurator Guide* for more details.
- **Toolbar** – Shows/hides the **Toolbar**.

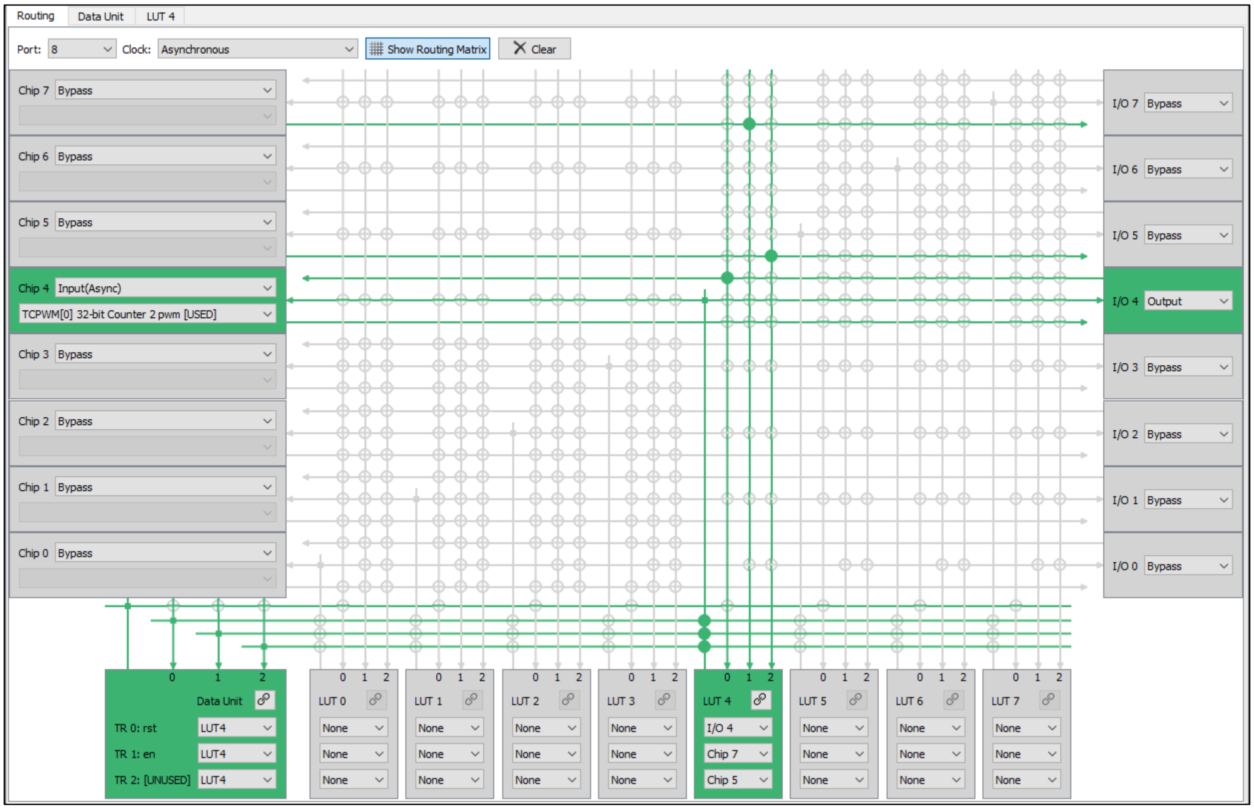
Help

- **View Help** – Opens this document.
- **About** – Opens the **About** box for version information.

Tabs

Routing Tab

The **Routing** tab is used to define the general settings and the routing configuration of the Smart I/O.



This tab contains the following parameters:

Port

Valid peripheral connections of a Smart I/O are port- and device-specific. 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. This clock is global within the Smart I/O and has differing constraints suited to different applications. Refer to the <u>Functional Description</u> section for more information.

- **Signal on I/O terminal 7…0** – Uses the selected I/O signal as the clock source. This signal may not be used in LUT inputs if used as the clock source.
- **Signal on Chip terminal 7…0** – Uses the selected Chip (peripheral or UDB) signal as the clock source. This signal may not be used in LUT inputs if used as the clock source. This clock can only be used during chip active and sleep modes.
- **Peripheral clock divider (Active)** – Divided clock from HFCLK. This clock is operational only in chip Active and Sleep modes. Sequential elements will be reset when entering Deep-Sleep or Hibernate mode and at POR.
- **Peripheral clock divider (Deep-Sleep)** – Divided clock from HFCLK. This clock is operational only in chip Deep Sleep modes. Sequential elements will be reset when entering Hibernate mode and at POR.

- **Peripheral clock divider (Hibernate)** – Divided clock from HFCLK. This clock is operational only in chip Hibernate modes. Sequential elements will be reset only at POR.
- **Clk_LF** – Low Frequency clock (either from ILO or WCO). This clock operates during chip Active, Sleep and Deep-sleep and allows the Smart I/O sequential logic to be clocked during those power modes.
- **Asynchronous** – If the Smart I/O is used purely for combinatorial logic, this option allows the block to conserve power by not using a clock. There are no constraints to power modes with this selection.

Show Routing Matrix

This button hides or displays the Smart I/O routing matrix. If the routing matrix is shown, you may click on the switches in the fabric to make input connections to the LUTs.

Clear

Click on this button to reset the routing matrix. All DU and LUT inputs will be cleared and the Chip and I/O terminal directions will become bypassed.

Chip Configuration

There are eight **Chip** signals that can be configured. Each has two pull-down menus; one to select the direction of the signal and the other to select the connection.

Chip 7…0 direction

Defines the direction of the specified **Chip** terminal. Valid options include:

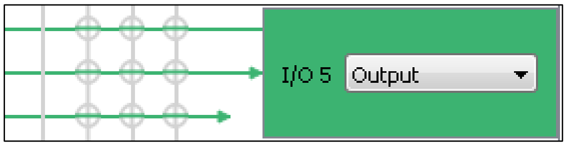
- **Bypass** – The line is bypassed and do not go through the Smart I/O routing fabric. i.e. Connection between the chip resource and the pin are directly connected. **Bypass** option frees this line and allows ModusToolbox to use this pin location for placing resources not meant to be used in the Smart I/O fabric.
- **Input (Sync/Async)** – Changes the direction of the terminal to be input type and allows connecting the matching peripheral/UDB signal to it. Input signals can be used as inputs to the LUTs. These can be either synchronized to the clock or remain asynchronous.
- **Output** – Changes the direction of the terminal to be output type and allows connecting the matching peripheral/UDB signal to it. Output signals can only be driven by the corresponding LUT outputs.
- **None** – The Chip terminal is consumed and cannot be used for connecting chip resources to it. This option is chosen automatically if the corresponding I/O terminal on the channel is specified as either **Input** or **Output**. You may use the terminal if the direction is set to **Output** or **Input** (only allowed if corresponding I/O is **Output**).

Chip 7…0 connection

When a **Port** is specified, these parameters allow each of the **Chip** terminals to connect to another peripheral on the chip.

I/O Configuration

There are eight **I/O** signals that can be configured. Each has a pull-down menu to select the direction of the signal.



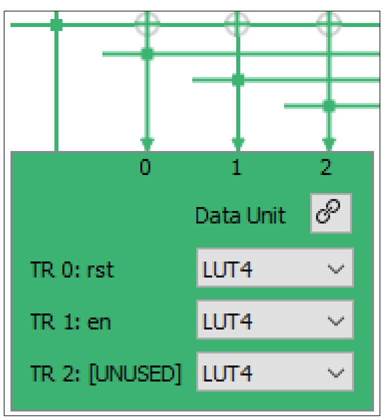
I/O 7…0 direction

Defines the direction of the specified **I/O** terminal. Valid options are:

- **Bypass** – The line is bypassed and do not go through the Smart I/O fabric. i.e. Connection between the chip resource and the pin are directly connected. **Bypass** option frees this line and allows ModusToolbox to use this pin location for placing resources not meant to be used in the Smart I/O fabric.
- **Input (Sync/Async)** – Changes the direction of the terminal to be input type and allows connecting an input I/O pin to it. Input signals can be used as inputs to the LUTs. These can be either synchronized to the clock (**Sync**) or remain asynchronous (**Async**).
- **Output** – Changes the direction of the terminal to be output type and allows connecting an output I/O pin to it. Output signals can only be driven by the corresponding LUT outputs.
- **None** – The I/O terminal is consumed and cannot be used for connecting a pin to it. This option is chosen automatically if the corresponding Chip terminal on the channel is specified as either **Input** or **Output**. You may use the terminal if the I/O direction is set to **Output** or **Input** (only allowed if corresponding Chip is **Output**).

Data Unit Input Configuration

The Data Unit configuration section contains three pull-down menus to select inputs. As you make various selections, the **Data Unit** tab becomes available for selection. You can also click the link icon to access the tab.



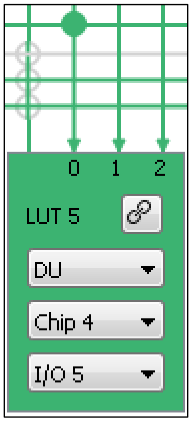
DU TR0, TR1, TR2

Defines the inputs for Data Unit triggers TR0, TR1 and TR2. The purpose of the triggers are dependent on the chosen Opcode as specified in the **Data Unit** tab. The triggers are active high.

- **Constant 0** – Default input selection. This effectively makes the trigger input to not perform any operation on the DU.
- **Constant 1** – Ties the trigger high, which activates and maintains the chosen DU operation.
- **DU** – Feeds back the single 1-bit DU output.
- **LUT 0…7** – Accepts any of the LUT outputs as an input.

LUT Input Configuration

There are eight **LUT** signals that can be configured. Each LUT configuration section contains three pull-down menus to select inputs from Chip, I/O, and DUT resources. As you make various selections, the corresponding **LUT** tab becomes available for selection. You can also click the link icon to access the tab.



LUT 7…0, Input 0

Defines the input 0 of the specified LUT. The actual valid selection depends on the enabled/used resources and terminals in your design. The LUT input 0 may accept any LUT outputs as an input **with the exception of LUT 0**. Instead, it may accept the DU output as an input.

- **DU** – The 1-bit output from the DU.
- **LUT 1…7** – Accepts the outputs of LUT1 to LUT 7.
- If using LUT 0…3, **I/O/Chip 0…3** signals are allowed as potential inputs.
- If using LUT 4…7, **I/O/Chip 4…7** signals are allowed as potential inputs.

LUT 7…0, Input 1

Defines the input 1 of the specified LUT. The actual valid selection depends on the enabled/used resources and terminals in your design.

- **LUT 0…7** – Accepts the outputs of LUT0 to LUT 7.
- If using LUT 0…3, **I/O/Chip 0…3** signals are allowed as potential inputs.
- If using LUT 4…7, **I/O/Chip 4…7** signals are allowed as potential inputs.

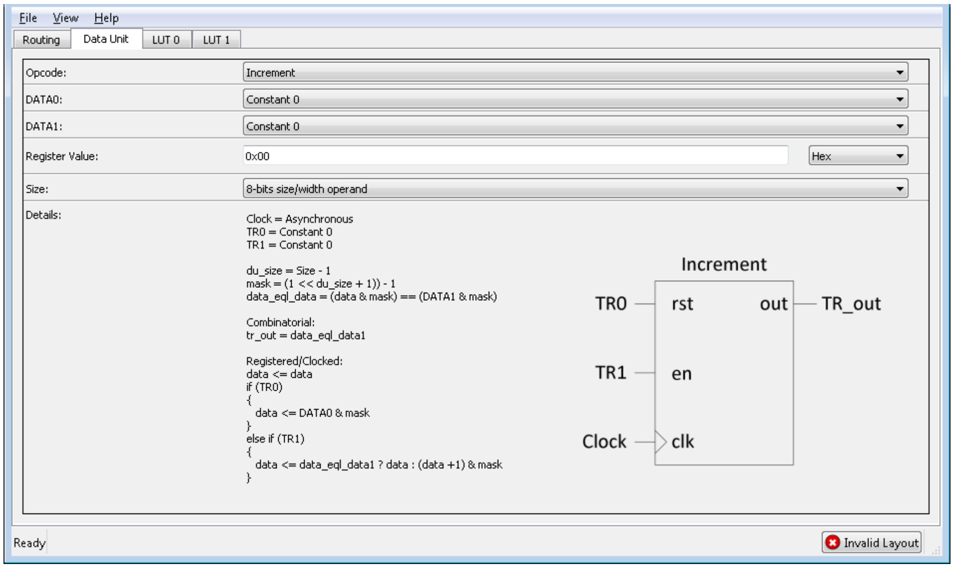
LUT 7…0, Input 2

Defines the input 2 of the specified LUT. The actual valid selection depends on the enabled/used resources and terminals in your design.

- **LUT 0…7** – Accepts the outputs of LUT0 to LUT 7.
- If using LUT 0…3, **I/O/Chip 0…3** signals are allowed as potential inputs.
- If using LUT 4…7, **I/O/Chip 4…7** signals are allowed as potential inputs.

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.



This tab contains the following parameters:

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.

Note Not all trigger inputs are required for a chosen Opcode. Refer to the pseudo verilog code in the **Details** window and also to the Functional Description section for more information.

- **Increment** – Implements an 8-bit One-shot Up Counter. The DU output goes high when the internal DU working data register is equal to DU DATA1 value.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the increment operation
- **Decrement** – Implements an 8-bit One-shot Down Counter. The DU output goes high when the internal DU working data register is equal to 0.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the decrement operation
- **Increment and wrap** – Implements an 8-bit Up Counter that wraps when it reaches DU DATA1 value. The DU output is a single clock pulse when the internal DU working data register is equal to DU DATA1 value.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the increment operation
- **Decrement and wrap** – Implements an 8-bit Down Counter that wraps when it reaches 0. The DU output is a single clock pulse when the internal DU working data register is equal to 0.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the decrement operation

- **Increment/Decrement** – Implements an 8-bit Up/Down Counter. The DU output goes high when the internal DU working data register is equal to either DU DATA1 register or if it is equal to 0.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the increment operation
  - TR2 = enable signal to start the decrement operation
- **Increment/Decrement and wrap** – Implements an 8-bit Up/Down Counter that wraps when it reaches either DU DATA1 (count up) or when it reaches 0 (count down). The DU output goes high when the internal DU working data register is equal to either DU DATA1 register or if it is equal to 0.
  - TR0 = reset signal that resets the working register to DU DATA0 value
  - TR1 = enable signal to start the increment operation
  - TR2 = enable signal to start the decrement operation
- **Rotate Right** – Implements a right circular shift register. The DU output is the LSB of the working register, which also gets fed back to the MSB of the working register.
  - TR0 = load signal that loads the working register with DU DATA0 value
  - TR1 = enable signal to start the shift right and rotate operation
- **Shift Right** – Implements a right shift register. The DU output is the LSB of the working register.
  - TR0 = load signal that loads the working register with DU DATA0 value
  - TR1 = enable signal to start the shift right operation
  - TR2 = shift in value that gets inserted into the MSB of the working register
- **AND, OR** – Implements a bitwise AND operation on the DU working register and DU DATA1. The DU output is high if the result of the operation is true.
  - TR0 = load signal that loads the working register with DU DATA0 value
- **Shift right and Majority 3** – Implements a shift register with a majority 3 comparison. The DU output will go high if the contents of the working register is equal to 0x03, 0x05, 0x06 or 0x007.
  - TR0 = load signal that loads the working register with DU DATA0 value
  - TR1 = enable signal to start the shift right operation
  - TR2 = shift in value that gets inserted into the MSB of the working register
- **Shift right and Compare** – Implements a shift register with a match DU DATA1 value comparison. The DU output will go high if the contents of the working register is equal to DU DATA1.
  - TR0 = load signal that loads the working register with DU DATA0 value
  - TR1 = enable signal to start the shift right operation
  - TR2 = shift in value that gets inserted into the MSB of the working register

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.

- **Constant 0** – Source is constant 0x00
- **Chip signal [7:0]** – Sourced from all 8 Chip terminals of the Smart I/O, allowing internal chip signals to be directly loaded into DATA0
- **I/O Signal [7:0]** – Sourced from all 8 I/O terminals of the Smart I/O, allowing external signals to be directly loaded into DATA0
- **DATA Register** – Sourced from the DU Register, which is accessible by the CPU

DATA1

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

Note DU DATA1 is not necessary for all Opcodes.

- **Constant 0** – Source is constant 0x00
- **Chip signal [7:0]** – Sourced from all 8 Chip terminals of the Smart I/O, allowing internal chip signals to be directly loaded into DATA1
- **I/O Signal [7:0]** – Sourced from all 8 I/O terminals of the Smart I/O, allowing external signals to be directly loaded into DATA1
- **DATA Register** – Sourced from the DU Register, which is accessible by the CPU

Register Value

Defines the 8-bit DU Reg value. This value is used as a source for DATA0 and/or DATA1.

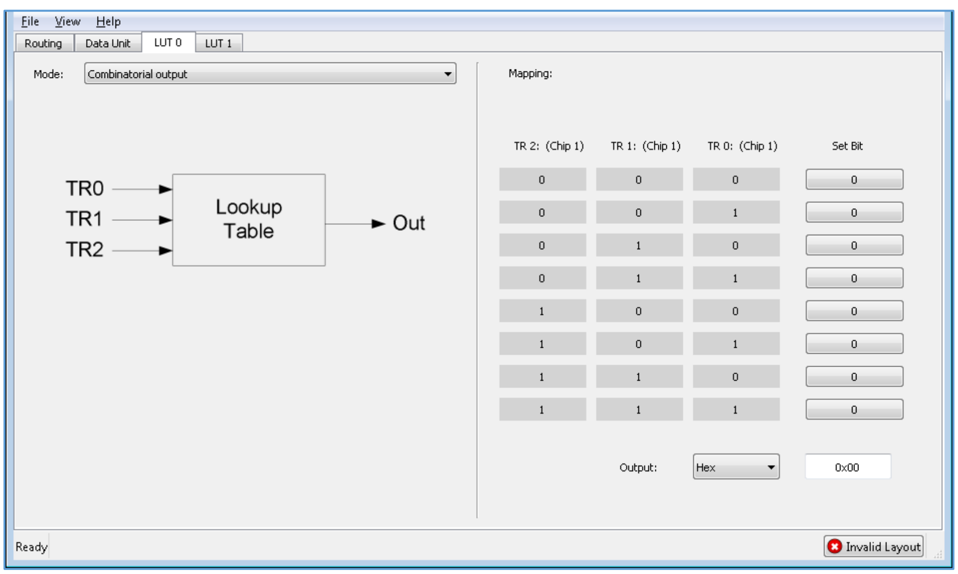
Note DU Reg is available only if either DATA0 or DATA1 are configured to be sourced from it.

Size

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

LUT Tabs

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



This tab contains the following parameters:

LUT 7…0 Mode

The LUTs can be configured in one of four modes:

- **Combinatorial** – The LUT is purely combinatorial. The LUT output is the result of the LUT mapping truth table, and will only be delayed by the LUT combinatorial path.

- **TR2 gated, combinatorial output** – The LUT input 2 is registered. The other inputs are direct connects to the LUT. The LUT output is combinatorial. You may use the output to feed back into input 2.
- **Sequential (gated) output** – The inputs are direct connects to the LUT but the output is registered.
- **Asynchronous Set/Reset mode** – The inputs and the LUT truth table are used to control an asynchronous S/R flip-flop.

LUT 7…0 Output mapping

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.

Note If the LUT is used to operate on a single signal (e.g. to invert a signal), then that signal must be connected to all 3 inputs of the LUT.

Functional Description

The Smart I/O implements a port-wide logic array that can be used to perform routing and logic operations to peripheral and I/O signals. The following sub sections describe the block restrictions and application critical information.

Routing Fabric

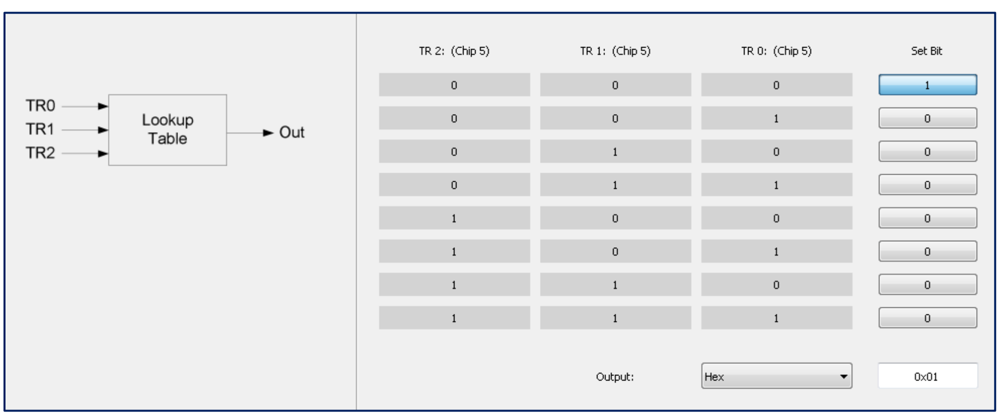
The Smart I/O routing fabric is divided into two portions, where each portion is capable of accepting half of the Chip or I/O signals. The LUTs have the following structure.

- LUT 7…4 are capable of accepting signals from I/O / Chip 7…4 as inputs.
- LUT 3…0 are capable of accepting signals from I/O / Chip 3…0 as inputs.
- The LUTs can accept any LUT output as an input.
- Each LUT output is dedicated to the corresponding output I/O and Chip terminals. For example, LUT 0 can go to either I/O 0 terminal (output type) or Chip 0 terminal (output type). The LUT output cannot be routed to an input terminal type.

Single Source LUT Input

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, consider the case where the signal on Chip 5 must be inverted before being passed to I/O 5. LUT 5 accepts Chip 5 as input 0, 1 and 2. The truth table is defined such that it outputs a logic 1 only when the inputs are all 0.



SCB Restriction

The SCB routing paths are restricted and can only be connected to the dedicated pin. They can however go through the dedicated LUT. For example, an SCB SPI slave select line on Chip 2 may be an input to LUT2, and then the output go to I/O 2. It cannot go to any other LUTs.

Clock and Reset Behavior

The Smart I/O Configurator drives its synchronous elements using a single peripheral-wide clock. Depending on the clock source, the Configurator will have different reset behaviors, which will reset all the flip-flops in the LUTs and synchronizers to logic 0. The configuration registers will retain their values unless coming out of Power on Reset (POR).

Note If the Configurator is only disabled, the values in the LUT flip-flips and I/O synchronizers are held as long as the chip remains in a valid power mode.

Note The selected clock for the fabric's synchronous logic is not phase aligned with other synchronous logic on the chip operating on the same clock. Therefore, communication between the Smart I/O and other synchronous logic should be treated as asynchronous (just as the communication between I/O input signals and other synchronous logic should be treated as asynchronous).

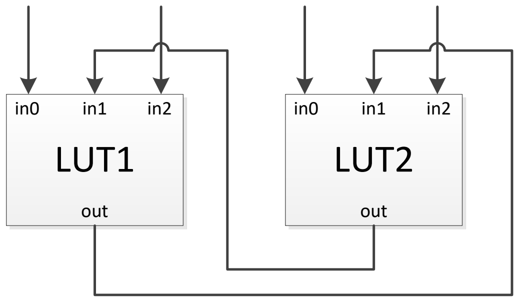
Clock Source	Reset Behavior	Enable Delay	Description
I/O 7…0	Reset on POR	2 clock edges	If chosen as the clock source, that particular signal cannot also be used as an input to a LUT as it may cause a race condition. The fabric will be enabled after 2 clock edges of the signal on the I/O terminal.
Chip 7…0	Reset on POR	2 clock edges	If chosen as the clock source, that particular signal cannot also be used as an input to a LUT as it may cause a race condition. The fabric will be enabled after 2 clock edges of the signal on the Chip terminal.
Peripheral Clock Divider (Active)	Reset when going to Deep Sleep, Hibernate or POR	2 clock edges	The fabric will be enabled after 2 clock edges of the divided clock. Any synchronous logic in the LUTs will be reset to 0 when in chip deep-sleep or hibernate modes.
Peripheral Clock Divider (Deep-Sleep)	Reset when going to Hibernate or POR	2 clock edges	The fabric will be enabled after 2 clock edges of the divided clock. Any synchronous logic in the LUTs will be reset to 0 when in hibernate mode.
Peripheral Clock Divider (Hibernate)	Reset on POR	2 clock edges	The fabric will be enabled after 2 clock edges of the divided clock.
Clk_LF	Reset when going to Hibernate and POR	2 clock edges	The fabric will be enabled after 2 clock edges of the low frequency clock (LFCLK). Any synchronous logic in the LUTs will be reset to 0 when in hibernate mode.
Asynchronous	Reset on POR	3 clock edges of SYSCLK	The fabric will be enabled after 3 clock edges of the system clock (SYSCLK).

Signal Synchronization Requirement

If any of the signals coming in through the Smart I/O are meant to be used in sequential elements in the LUTs, the terminal synchronizer must first be used to synchronize that signal to the peripheral clock. For example, if the signal on I/O 0 must be used in LUT0 in Sequential output mode, the synchronization for I/O 0 terminal should be enabled for reliable operation.

LUT Combinatorial Feedback

Since the LUTs can be configured as purely (or partially) combinatorial elements and since they can chain to each other in any fashion, combinatorial timing loops can occur. This causes oscillations that burn power and create unpredictable behavior. If a feedback is required, the signals should always go through a flip-flop before feeding back. For example, the following is a potentially problematic design. LUT1 and LUT2 are configured in **Combinatorial** mode. This will result in oscillations. To prevent it, one of the LUTs should be configured to **Gated Output** mode.



References

Refer to the following documents for more information, as needed:

- ModusToolbox Device Configurator Guide
- Eclipse IDE for ModusToolbox User Guide
- PDL API Reference Guide
- Device Datasheets
- Device Technical Reference Manuals

Version Changes

This section lists and describes the changes for each version of this tool.

Version	Change Descriptions
1.0	New tool.
1.1	Updated to incorporate changes to the back end.
1.2	Implemented various defect fixes and performance enhancements.
	Added **Open Containing Folder** menu item.
2.1	Updated to incorporate changes to the back end; version updated to be in sync with Device Configurator. Added Open System Explorer menu item. Added Change Devices menu item and dialog. Added major/minor version number to the title bar.
2.20	Added Copy feature to the Notice List. Added feature to support incremental patch updates.