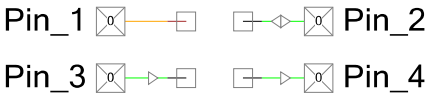
Pins
1.60

Features



- Rapid setup of all pin parameters and drive modes
- Allows PSoC Creator to automatically place and route signals
- Allows interaction with one or more pins simultaneously

General Description

The Pins component is the preferred way for hardware resources to connect to a physical port-pin. It provides access to external signals through an appropriately configured physical IO pin. It also allows electrical characteristics to be associated with one or more pins; these characteristics are then used by PSoC Creator to automatically place and route the signals within the component.

Pins can be used with schematic wire connections, software, or both. To access a Pins component from component APIs, the component must be contiguous and nonspanning. This ensures that the pins are guaranteed to be mapped into a single physical port. Pins components that span ports or are not contiguous can only be accessed from a schematic or with the global per-pin APIs.

Note There are #defines created for each pin in the Pins component to be used with global APIs.

A Pins component can be configured into any legal combination of types. For convenience, the Component Catalog provides four preconfigured Pins components: Analog, Digital Bidirectional, Digital Input, and Digital Output.

When to Use a Pins Component

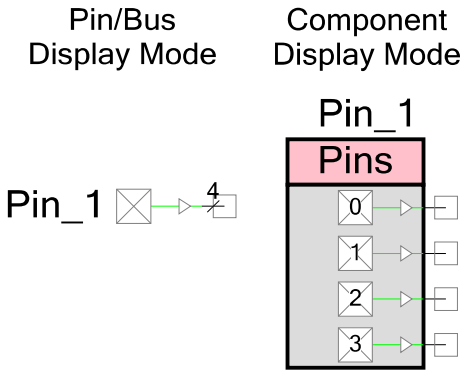
Use the Pins component when a design must generate or access an off-device signal through a physical IO pin. Pins are the most commonly used component in the Catalog. For example, they are used to interface with potentiometers, buttons, LEDs, peripheral sensors such as proximity detectors, and accelerometers.

Input/Output Connections

This section describes the various input and output connections for the Pins component.

Display of Pins

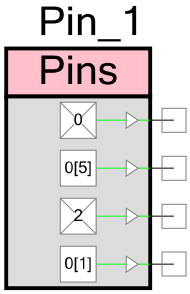
Pins can be configured into complex combinations of input, output, bidirectional, and analog. Simple configurations with less than two internal hardware connections are generally shown as single pins. More complex types of pins, arrays of pins, or buses are shown as standard components with a bounding box.



The default, and most common, configurations are shown in the following sections.

Display of Locked Pins

When you assign a Pins component to a physical GPIO or SIO pin using the PSoC Creator Design-Wide Resources Pin Editor, the tooltip for the Pins component shows the specific pin assignments. If you lock a pin assignment, the display of the component indicates the assignment, as shown in the following example:



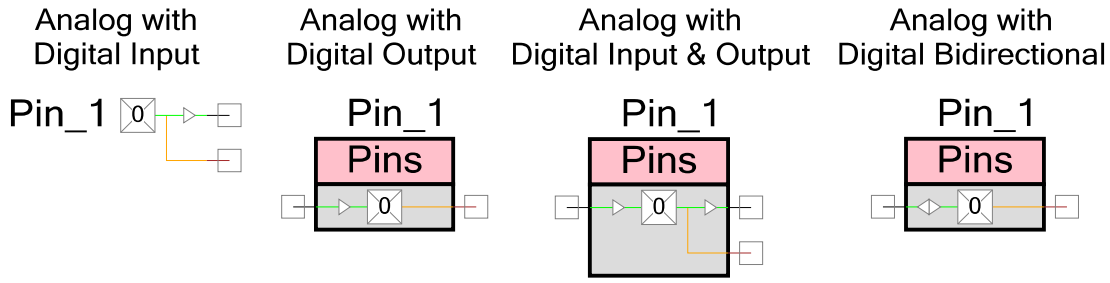
Note If the Pins component is set to **Display as Bus,** the display of the component does not display any locked pin assignments; however, the tooltip still displays this information.

Analog

Configure your Pins component as Analog any time your design requires a connection between a device pin and an internal analog terminal connected with an analog wire. When configured as analog, the terminal is shown on the right side of the symbol with the connection drawn in the color of an analog wire.



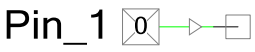
An analog Pins component may also support digital input or output connections, or both, as well as bidirectional connections. It is possible to short together digital output and analog signals on the same pin. This can be useful in some applications; however, it is an advanced topic and should be used with care.



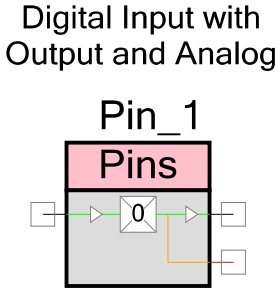
Digital Input

Configure a Pins component as digital input any time your design requires a connection between a device pin and an internal digital input terminal, or if the pin's state is read by the CPU/DMA. In all cases using digital-input pins, the pin state is readable by the CPU/DMA. Additionally, if the terminal is displayed it can be routed to other components in the schematic.

When visible, the terminal is shown on the right side of the symbol. The connection is drawn in the color of a digital wire with a small input buffer to show signal direction.



A digital-input Pins component may also support digital output and analog connections.

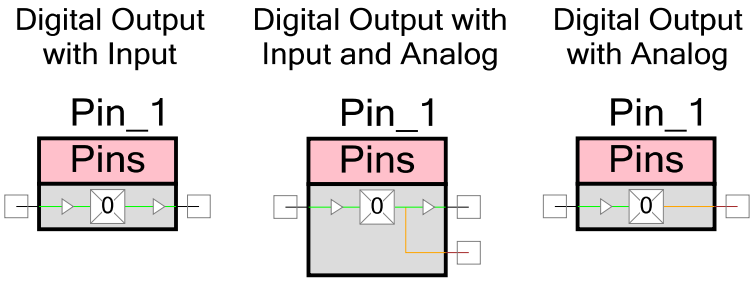


Digital Output

Configure a Pins component as digital output any time a device pin is to be driven to a logic high or low. In all such cases, the pin state is writeable by the CPU/DMA. Additionally, if the terminal is displayed it can be routed from other components in the schematic. When visible, the terminal is shown on the left side of the symbol. The connection is drawn in the color of a digital wire with a small output buffer to show signal direction.



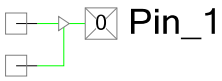
A digital-output Pins component may also support digital input and analog connections.



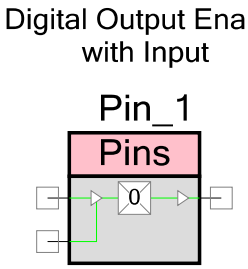
Digital Output Enable

Select digital output enable when digital logic is to be used to quickly control the pin output driver without CPU intervention. A high logic level on this terminal enables the pin output driver as configured by the **Drive Mode** parameter on the **General** subtab. A logic low level on this terminal disables the pin output driver and makes the pin assume the HI-Z drive mode. This terminal is shown when a component is configured with digital output using a schematic connection, and when the digital output enable has been selected. The digital output enable appears on the left side of the symbol and connects to the digital output buffer. It is drawn in the color of a digital wire.

When the pin is set to **Display as Bus**, only one output enable is provided regardless of the Pins component width because all of the pins share the same output enable. When not displayed as a bus, individual output enables are provided per pin.



A digital output enable Pins component may also support input and analog connections.



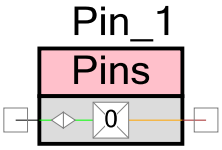
Digital Bidirectional

Configure a Pins component as digital bidirectional any time your design requires a connection between a device pin and an internal digital bidirectional terminal. Digital bidirectional mode is most often used with a communication component like I^2C. When configured as digital bidirectional, the terminal is shown on the left side of the symbol with the connection drawn in the color of a digital wire with input and output buffers showing that the signal is bidirectional.



A bidirectional Pins component may also support analog connections.

Digital Bidirectional
with Analog



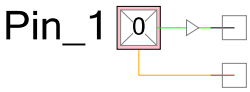
Vref

To configure a Pins component to use a Vref signal:

- Use a digital input or bidirectional terminal and set the **Threshold** parameter to **Vref** on the **Input** subtab, or
- Use a digital output or bidirectional terminal and configure the **Drive Level** to **Vref** on the **Output** subtab

Using a Vref requires an SIO pin, indicated with a pink outline. All pins can supply their respective V_{DDIO} supply voltages. SIO pins can also supply a programmable or analog-routed voltage for interface with devices at a different potential than the SIO's Vddio voltage. The Vref terminal provides the analog routed voltage supplied to the SIO pin. SIO pins can also use the Vref input as the input threshold for an SIO.

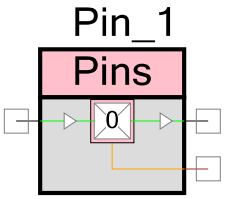
The Vref signal displays on the right side of the component, extending from the bottom of the SIO single pin or the SIO pin pair, depending on how it is configured. Each SIO pin pair shares a single Vref input.



Vref can only be used in conjunction with another digital input or output connection.

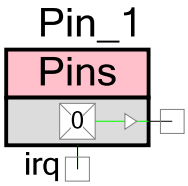
Note When using Vref, you cannot select **Analog**.

Vref with
Digital Input & Output



IRQ

To configure a Pins component with an interrupt, you must use a digital input and configure the **Interrupt** parameter on the **Input** subtab. When interrupts are used, the Pins component displays with a bounding box, and the IRQ is displayed extending from the bottom of the component. The typical use case is to connect an Interrupt component to this terminal.



An Interrupt can be used in all configurations of the Pins component, as long as you include digital input.

- **Interrupt –** This parameter selects whether the pin can generate an interrupt and, if selected, the interrupt type. The pin interrupt may be generated with a rising edge, falling edge, and both edges. If set to anything but **None**, the component must be configured to be contiguous to ensure it is mapped into a single physical port. A single port is required because all pins in a port logically OR their interrupts together and generate a single interrupt signal and symbol terminal. The **Interrupt** parameter uses dedicated pin interrupt logic, which latches the pins that generated interrupted events. After an interrupt occurs, the Pin_ClearInterrupt() function must be called to clear the latched pin events to enable detection of future events. If more than one pin in the Pins component can generate an interrupt, the Pin_ClearInterrupt() return value can be decoded to determine which pins generated interrupt events.

While not the preferred method, any digital input hardware connection can also be connected to an isr component, providing the ability to generate a pin interrupt on high or low logic level versus on an edge event. Using the digital input connection for a level interrupt does not use the dedicated pin interrupt logic configured with this parameter.

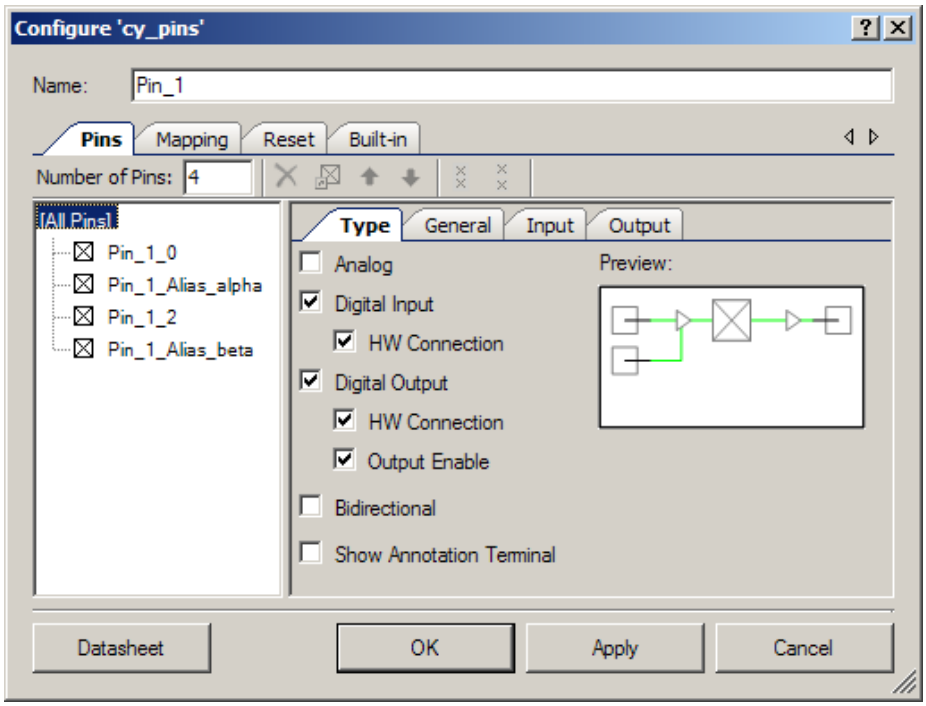
- **None** - Default
- **Rising Edge**
- **Falling Edge**
- **Both Edges**

Component Parameters

Drag a Pins component onto the design schematic and double click it to open the **Configure** dialog. This dialog is used to set component-wide parameters, such as the power-on reset state and physical pin mapping constraints. The parameters are organized into separate tabs called subtabs.

Pins Tab

The **Pins** tab has three areas: a toolbar, pin tree, and another set of subtabs. The toolbar is used to determine how many physical pins are managed by the component and determine their order. The subtabs are used to set the pin-specific attributes, such as type, direction, drive mode, and initial state. The pin tree works with the subtabs to allow you to choose the specific pins to which these attributes are applied.



Toolbar

Contains these commands:

- **Number of Pins** – The number of device pins controlled by the component. Valid values are between 1 and 64. The default value is **1**.

  Note Some configurations can only be placed into a single physical port; therefore, the default maximum number of pins is limited to 8 or less. When the component is configured as noncontiguous and spanning, the maximum number of pins can be set up to 64 because they no longer need to be placed into a single physical port.

- **Delete Pin –** Deletes selected pins from the tree.

- **Add/Change Alias –** Opens a dialog to add or change the alias name for a selected pin in the tree. You can also double-click a pin or press [**F2**] to open the dialog.

- **Move Up/Down –** Moves the selected pins up or down in the tree.

- **Pair/Unpair SIOs –** Pairs or unpairs selected SIO pins (identified by a pink outline) in the tree.

  This control specifies whether pins that require SIO should be placed in the same SIO pair on the device. Pairing pins results in fewer physical SIO pins being "wasted." This is because an unpaired pin that requires SIO cannot share its SIO pair on the device with another pin that requires SIO. For pins to share an SIO pair on the device, they must have their per-pair settings configured the same way and be adjacent.

  A pin requires SIO if **Hot Swap** is set to true, **Threshold** is set to anything but **LVTTL** or **CMOS**, **Drive Level** set to **Vref**, and/or **Drive Current** is set to **25mA sink**.

Pin Tree

This area displays all of the pins for the component. You can individually select one or more pins to use with the toolbar commands and subtabs. Each pin displays its name which consists of the Pins component name + '_' + individual pin alias.

Type Subtab

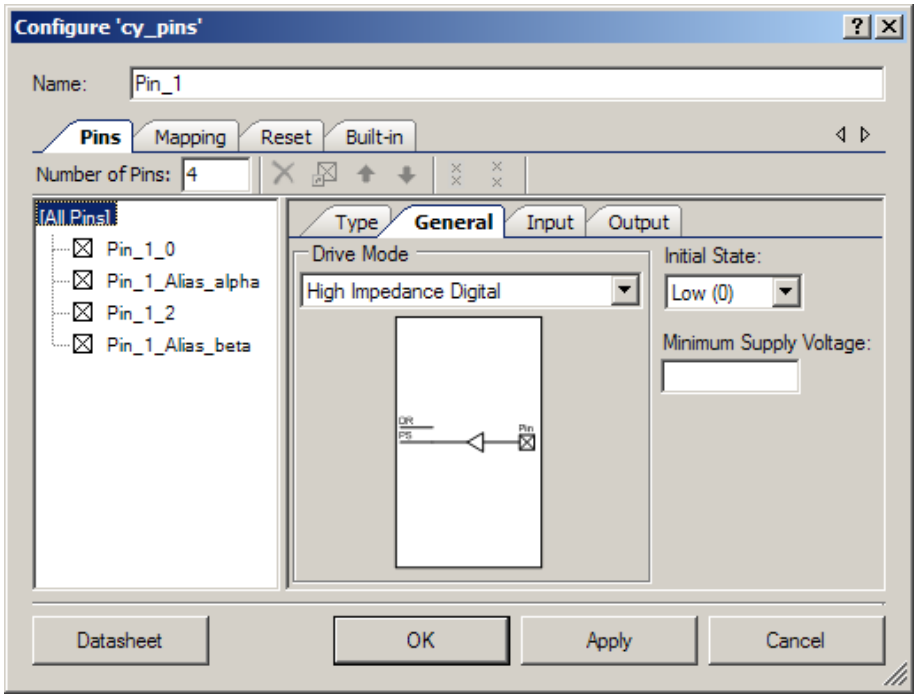
This is the default subtab displayed for the **Pins** tab. This is where you choose the type of pins for your component using the check boxes. The preview area shows what the selected Pins component symbol will look like with various options selected for that specific pin.

- **Analog** – Select **Analog** to enable the analog pin terminal to allow analog signal routing to other components. Selecting analog forces the pin to be physically placed on a GPIO pin and not an SIO pin.
- **Digital Input** – Select **Digital Input** to enable the digital input pin terminal (optional) and enable the **Input** subtab for additional configuration options related to inputs.
  - **HW Connection** – This parameter determines whether the digital input terminal for an input pin is displayed in the schematic. If displayed, the pin provides a digital signal to the digital system interconnect (DSI) for use with hardware components. Independent of this selection, all pins can always be read by the CPU through registers or APIs. If this option is not selected, the terminal is not displayed and it is controlled only by software APIs.
- **Digital Output** – Select **Digital Output** to enable the digital output pin terminal (optional) and enable the **Output** subtab for additional configuration options related to outputs.
  - **HW Connection** – This parameter determines whether the digital output terminal for a given output pin is displayed in the schematic. If displayed, the pin outputs the digital signal supplied by hardware components through the DSI. If not displayed, the output logic level is determined by CPU register or API writes. If this option is not selected, the terminal is not displayed and it is controlled only by software APIs.
  - **Output Enable** – This parameter allows the use of the output enable feature of pins and displays the output enable input terminal. The output enable feature allows a

hardware signal to control the pin's output drivers without requiring the CPU to write registers. A high logic level configures the output drivers, as set in the **Drive Mode** parameter. A low logic level disables the output drivers and places the pin into the HI-Z drive mode.

- **Bidirectional** – Enabling the **Bidirectional** parameter is functionally equivalent to enabling the **Digital Input** with **HW Connection** and the **Digital Output** with **HW Connection** parameters. The difference is that only a single bidirectional terminal is displayed on the component symbol rather than separate input and output terminals. Both **Input** and **Output** subtabs are enabled for further configuration.
- **Show Annotation Terminal** – Allows connections to Annotation Library Components to illustrate circuitry external to PSoC.

General Subtab

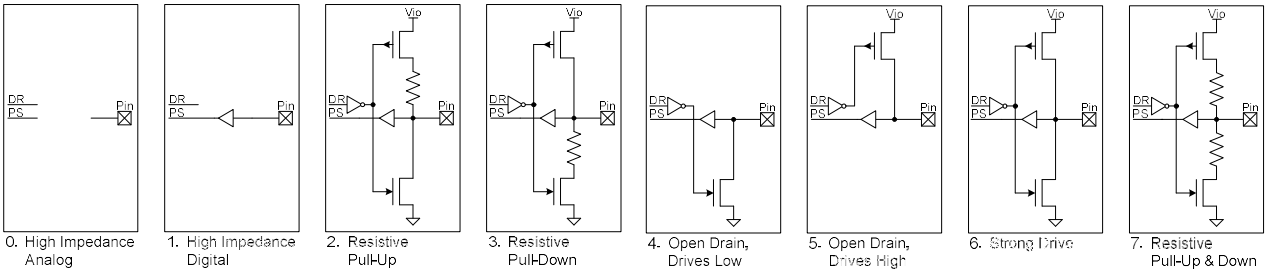


The **General** subtab allows you to set up parameters that apply to all pins, such as the drive move, initial state, and minimum supply voltage of the selected pin. The settings on this subtab include:

- **Drive Mode** – This parameter configures the pin to provide one of the eight available pin drive modes. The defaults and legal choices are influenced from the selections on the **Type** subtab. Refer to the device datasheet for more details on each drive mode. A diagram shows the circuit representation for each drive mode as it is selected.
  - If the type is **Digital Input** or **Digital Input/Analog**, the default is **High Impedance Digital**.

- If the pin type is **Analog**, the default is **High Impedance Analog**.
- If the pin type is **Bidirectional** or **Bidirectional**/**Analog**, the default is **Open Drain, Drives Low**.
- All other pin types default to **Strong Drive**.

The diagram for each drive mode is as follows:



Note If any of the three resistive drive modes (**Resistive Pull Up**, **Resistive Pull Down**, **Resistive Pull Up/Down**) is used, setting the output drive level to **Vref** does not work.

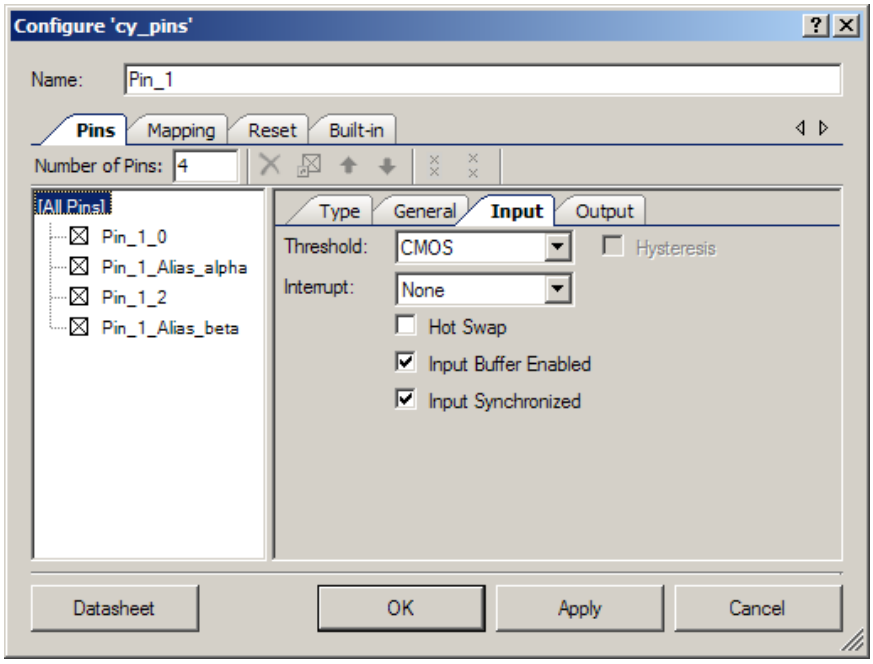
- **Initial State** – This parameter specifies the pin-specific initial value written to the pin's data register after power-on reset (POR). All pins default to a logic low (0) in hardware at POR. The initial state is written to the pin just after the drive mode is configured, which occurs as part of the configuration of the entire device. The initial state is configured high by default only for the **Resistive Pull Up** and **Resistive Pull Up/Down** drive modes to ensure the pull-up resistor is active.

  Note This should not be confused with the reset state under the main **Reset** tab. That attribute affects the state of the whole port of which the pin is a member, from the moment of reset, before any other device configuration.

- **Minimum Supply Voltage** – This parameter selects the requested minimum high logic level output voltage. The requested voltage must be provided by one of the V_{DDIO} supply inputs. This selection ensures that the Pins component will be mapped onto pins that can support its required output voltage. If left blank, the component has no voltage requirements, allowing placement to a pin supplied by any of the available V_{DDIO} voltages.

  Valid values are determined by the settings in the **System** tab of the *<project>.cydwr* file for V_{IO0}/V_{IO1}/V_{IO2}/V_{IO3}, V_{IO3}, and to a lesser extent V_{DDD}. Depending on the selected device, you could have two USB pins that will use V_{DDD} as their voltage available for placement. The pin cannot be placed if this value is not less than or equal to the maximum value set for those settings. This range check is performed outside this dialog; the results appear in the Notice List window if the check fails.

Input Subtab

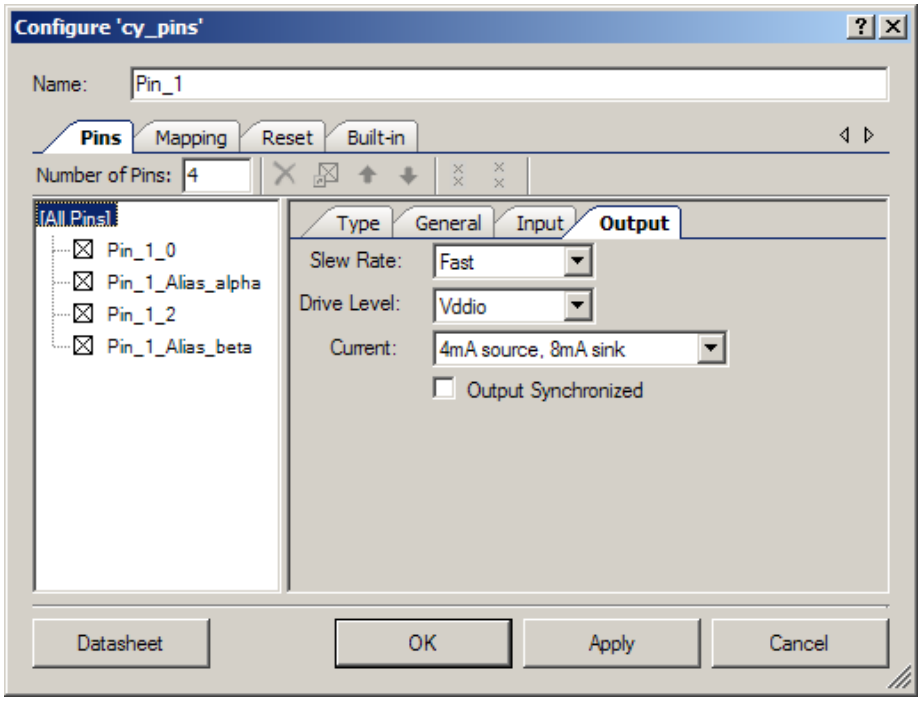


The **Input** subtab specifies input settings. If the pin type is not **Digital Input** or **Bidirectional** in the **Type** subtab, this subtab is disabled because you do not need to specify input information.

- **Threshold** – This parameter selects the threshold levels that define a logic high level (1) and a logic low level (0). **CMOS** is the default and should be used for the vast majority of application connections. The other threshold levels allow for easy interconnect with devices with custom interface requirements that differ from that of CMOS. Thresholds that are derived from Vddio or Vref require the use of an SIO pin.
  - **CMOS** – Default
  - **LVTTL**
  - **CMOS or LVTTL**
  - **0.4 x Vddio** – Requires SIO
  - **0.5 x Vddio** – Requires SIO
  - **0.5 x Vref** – Requires SIO
  - **Vref** – Requires SIO
- **Hysteresis** – Enables or disables the SIO differential hysteresis for the pin. This feature is disabled if the **Threshold** is **CMOS**, **LVTTL**, or **CMOS or LVTTL**. Hysteresis control requires you to use an SIO pin. GPIO pins always have hysteresis enabled.
  - Disabled – Default
  - Enabled

- **Interrupt** – This parameter selects whether the pin can generate an interrupt and, if selected, the interrupt type. The pin interrupt can be generated with a rising edge, falling edge, or both edges. If set to anything but **None**, you must configure the component to be contiguous so that it is mapped into a single physical port. A single port is required because all pins in a port logically OR their interrupts together and generate a single interrupt signal and symbol terminal.
  - **None** - Default
  - **Rising Edge**
  - **Falling Edge**
  - **Both Edges**
- **Hot Swap** – A pin configured for hot swap capability is mapped to an SIO pin that supports this capability in hardware. Hot swap capability allows the voltage present on the pin to rise above the pin's V_{DDIO} voltage, up to 6.0 V. Hot swap also does not allow a pin with any voltage up to 6.0 V present to leak current into the PSoC device even when the PSoC device is not powered. Hot swap is useful for connecting the PSoC device when unpowered to a communications bus like I^2C without shorting the bus or back powering the PSoC device.
  - Disabled – Default
  - Enabled – Requires SIO
- **Input Buffer Enabled** – This parameter enables or disables the pin's digital input buffer. The digital buffer is needed to read or use the logic level present on a pin through DSI routing or a CPU read. The input buffer is needed to use the pin as a digital input. Analog pins disable the digital input buffer by default to reduce pin leakage in low-power modes. If the pin type is **Analog**, the default is **Disabled**. All other pin types, including combinations that include **Analog**, default to **Enabled**. You should disable the input buffers to reduce current when not needed, especially with analog signals.
  - Enabled
  - Disabled
- **Input Synchronized** – Input synchronization occurs at pins to synchronize all signals entering the device to bus_clk. Input synchronization can be optionally disabled at the pin in limited cases in which an asynchronous signal is required for application performance and does not violate device operational requirements. Refer to the TRM or device datasheet for use details.
  - Enabled – Default
  - Disabled

Output Subtab



The **Output** subtab specifies output settings. If the pin type is not **Digital Output** or **Bidirectional** this tab is disabled because you do not need to specify output information.

- **Slew Rate** – The slew rate parameter determines the rise and fall ramp rate for the pin as it changes output logic levels. Fast mode is required for signals that switch at greater than 1 MHz. You can select slow mode for signals less than 1 MHz switching rate and benefit from slower transition edge rates, which reduce radiated EMI and coupling with neighboring signals.
  - Fast – Default
  - Slow
- **Drive Level** – This parameter selects the output drive voltage supply sourced by the pin. All pins can supply their respective V_{DDIO} supply voltages. SIO pins can also supply a programmable or analog routed voltage for interface with devices at a different potential than the SIOs V_{DDIO} voltage.
  - **Vddio** – Default
  - **Vref** – Requires SIO

  Note If any of the three resistive drive modes (**Resistive Pull Up**, **Resistive Pull Down**, **Resistive Pull Up/Down**) is used, setting the output drive level to **Vref** does not work..
- **Current** – The drive current selection determines the maximum nominal logic level current required for a specific pin. Pins can supply more current at the cost of logic level compliance

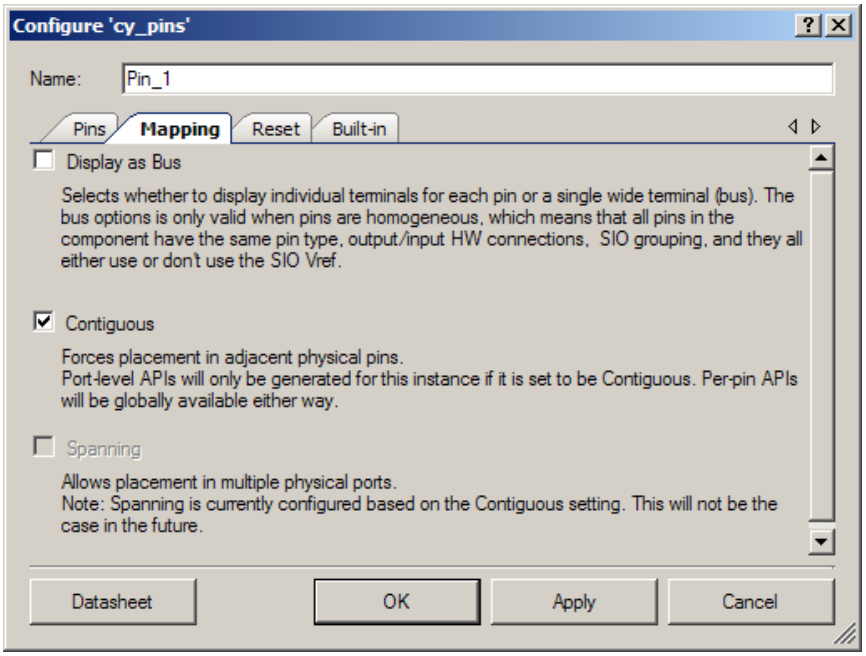
or can have a maximum value that is less than listed, based on system voltages. See the device datasheet for more details on drive currents.

- **4mA source, 8mA sink** – Default
- **4mA source, 25mA sink** – Requires SIO

- **Output Synchronized** – Output synchronization reduces pin-to-pin output signal skew in high-speed signals requiring minimal signal skew. The output signal is synchronized to bus_clk. See the TRM or device datasheet for use details.
  - Disabled – Default
  - Enabled

Mapping Tab

The **Mapping** tab contains parameters that define how the Pins component is displayed in the schematic view and mapped on to physical pins.



Display as Bus

This parameter selects whether to display individual terminals for each pin or a single wide terminal (bus). The bus option is only valid when pins are homogeneous. That means all pins in the component have the same pin type, output/input HW connections, and SIO grouping. They also must all either use or not use the SIO Vref. Displaying as a bus is useful when many of the same types of pin are required. This saves schematic space and time to configure and route.

Contiguous

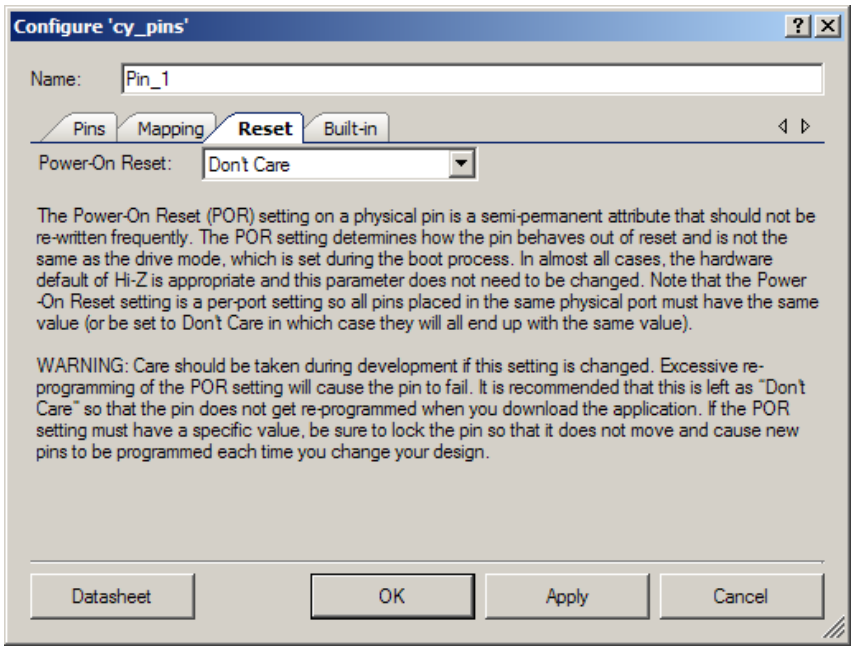
This parameter forces placement in adjacent physical pins within a port. Actual pin placement is package dependent according to the device datasheet. This option has the following restrictions:

- If contiguous, port level APIs are generated for the component. If noncontiguous, port level APIs are not generated.
- If contiguous, the number of pins in the component must be less than or equal to 8.

Spanning

This parameter enables placement in multiple physical ports. This is currently controlled by the contiguous selection, where contiguous implies nonspanning and noncontiguous implies spanning. A future release of the software will support separate control of the **Spanning** parameter.

Reset Tab



Power-On Reset

The Power-On Reset (POR) setting on a physical pin is a semipermanent attribute that you should not rewrite frequently. The POR setting determines how the pin behaves out of reset. It is not the same as the drive mode, which is set during the boot process. In almost all cases, the hardware default of HI-Z is appropriate and you do not need to change this parameter. Note that the Power-On Reset setting is a per-port setting, so all pins placed in the same physical port must have the same value (or be set to **Don't Care**, in which case they will all end up with the same value). Power-On Reset cannot be specified on PSoC 5 devices.

Warning: Be careful during development if this setting is changed. Excessive reprogramming of the POR setting causes the pin to fail. See the device datasheet for the maximum number of NVL write cycles. It is best to leave this as **Don't Care**, so that the pin is not reprogrammed when you download the application. If the POR setting must have a specific value, be sure to lock the pin so that it does not move and cause new pins to be programmed each time you change your design.

- **Don't Care** – Default. When left set to **Don't Care**, the POR is determined by the physical port in which this component is placed. If all of the placed pins in the port are set to **Don't Care**, the default POR of the part is used. Otherwise, whatever POR is specified for the other pins placed in that physical port (they must all match) is used for the ones set to **Don't Care**.
- **High-Z Analog**
- **Pulled-Up**
- **Pulled-Down**

Resources

Each Pins component consumes one physical pin per bit of the **Number of Pins** parameter.

	Digital Blocks					API Memory (Bytes)		
Analog Blocks	Datapaths	Macro cells	Status Registers	Control Registers	Counter7	Flash	RAM	Pins (per External I/O)
N/A	N/A	N/A	N/A	N/A	N/A	110	0	N/A

Application Programming Interface

Application Programming Interface (API) routines allow you to configure and use the component using software. The Pins component enables access on a per-pin and component-wide basis.

Per-Pin APIs

You can access individual pins in the component by using the global APIs defined in the *cypins.h* generated file (in the cy_boot directory). These APIs are documented in the *System Reference Guide* (Help > Documentation) and include:

- CyPins_ReadPin()
- CyPins_SetPin()
- CyPins_ClearPin()

- CyPins_SetPinDriveMode()
- CyPins_ReadPinDriveMode()

These APIs can be used with either physical pin register names or the pin alias from the component. Accessing physical pins directly from software is not recommended because there is no safeguard against the same pins being allocated to other functions by the tool. Even if a pin is only accessed from software, Cypress strongly recommends the use of a Pins component. You can use the generated aliases from the component with the above APIs to safely access individual pins without a performance or memory penalty.

To use the above APIs, the component generates aliases for the pin registers in the *CyPins_aliases.h* file. By default the alias is the component name with the pin number appended to it:

CyPins_x - x is the pin within the component (0 based)

If you provide an alias name in the Pins configuration dialog, then an additional #define is created with the form:

CyPins_<AliasName>

Component APIs

These APIs access all pins in the component in a single function call. Efficient implementation of component-wide APIs is only possible if all pins are placed in a single physical port on the device. They are only generated if the component is configured to be contiguous. Noncontiguous Pins components only allow access on the per-pin basis described above.

By default, PSoC Creator assigns the instance name “Pin_1” to the first instance of a Pins component in a given design. You can rename it to any unique value that follows the syntactic rules for identifiers. The instance name becomes the prefix of every global function name, variable, and constant symbol. For readability, the instance name used in the following table is “Pin.”

On PSoC 3 ES2 and PSoC 5, pins P15[7:6] are not available for software control.

The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

Function	Description
Pin_Read()	Reads the physical port and returns the current value for all pins in the component
Pin_Write()	Writes the value to the component pins while protecting other pins in the physical port if shared by multiple Pins components
Pin_ReadDataReg()	Reads the current value of the port’s data output register and returns the current value for all pins in the component

Function	Description
Pin_SetDriveMode()	Sets the drive mode for each of the Pins component's pins
Pin_ClearInterrupt()	Clears any active interrupts on the port into which the component is mapped. Returns value of interrupt status register

uint8 Pin_Read(void)

Description: Reads the associated physical port (pin status register) and masks the required bits according to the width and bit position of the component instance. The pin's status register returns the current logic level present on the physical pin.

Parameters: None

Return Value: The current value for the pins in the component as a right justified number.

Side Effects: None

void Pin_Write(uint8 value)

Description: Writes the value to the physical port (data output register), masking and shifting the bits appropriately. The data output register controls the signal applied to the physical pin in conjunction with the drive mode parameter. This function avoids changing other bits in the port by using the appropriate method (read-modify-write or bit banding).

Parameters: uint8 value: Value to write to the component instance.

Return Value: None

Side Effects: If you use read-modify-write operations that are not atomic; the Interrupt Service Routines (ISR) can cause corruption of this API. An ISR that interrupts this API and performs writes to the Pins component data register can cause corrupted port data. To avoid this issue, you should either use the Per-Pin APIs (primary method) or disable interrupts around this API.

uint8 Pin_ReadDataReg(void)

Description: Reads the associated physical port's data output register and masks the correct bits according to the width and bit position of the component instance. The data output register controls the signal applied to the physical pin in conjunction with the drive mode parameter. This is not the same as the preferred Pin_Read() API because the Pin_ReadDataReg() reads the data register instead of the status register. For output pins this is a useful API to determine the value just written to the pin.

Parameters: None

Return Value: The current value of the data register masked and shifted into a right justified number for the component instance.

Side Effects: None

void Pin_SetDriveMode(uint8 mode)

Description: Sets the drive mode for each of the Pins component's pins.

Parameters: uint8 mode: Mode for the selected signals. Defined legal options are:

Pin_1_DM_STRONG	Strong Drive
Pin_1_DM_OD_HI	Open Drain, Drives High
Pin_1_DM_OD_LO	Open Drain, Drives Low
Pin_1_DM_RES_UP	Resistive Pull Up
Pin_1_DM_RES_DWN	Resistive Pull Down
Pin_1_DM_RES_UPDWN	Resistive Pull Up/Down
Pin_1_DM_DIG_HIZ	High Impedance Digital
Pin_1_DM_ALG_HIZ	High Impedance Analog

Return Value: None

Side Effects: If you use read-modify-write operations that are not atomic, the Interrupt Service Routines (ISR) can cause corruption of this API. An ISR that interrupts this API and performs writes to the Pins component Drive Mode registers can cause corrupted port data. To avoid this issue, you should either use the Per-Pin APIs (primary method) or disable interrupts around this API.

uint8 Pin_ClearInterrupt(void)

Description: Clears any active interrupts attached with the component and returns the value of the interrupt status register allowing determination of which pins generated an interrupt event.

Parameters: None

Return Value: uint8: The right-shifted current value of the interrupt status register. Each pin has one bit set if it generated an interrupt event. For example, bit 0 is for pin 0 and bit 1 is for pin 1 of the Pins component.

Side Effects: Clears all bits of the physical port's interrupt status register, not just those associated with the Pins component.

Sample Firmware Source Code

PSoC Creator provides many example projects that include schematics and example code in the Find Example Project dialog. For component-specific examples, open the dialog from the Component Catalog or an instance of the component in a schematic. For general examples, open the dialog from the Start Page or **File** menu. As needed, use the **Filter Options** in the dialog to narrow the list of projects available to select.

Refer to the "Find Example Project" topic in the PSoC Creator Help for more information.

DC and AC Electrical Characteristics

The following values indicate of expected performance and are based on initial characterization data.

Note For PSoC 5 silicon under certain conditions, an SIO pin may cause up to 1 mA of additional current to be drawn from the related V_{DDIO} pin. If an SIO pin's voltage exceeds its V_{DDIO} supply by 0.5 V, the trigger condition is set. After the trigger condition is set, the SIO pin causes increased current when its voltage is between V_{SSD} + 0.5 V and V_{DDIO} – 0.5 V. The trigger condition is reset when the SIO pin is brought within the range of V_{SSD} to V_{SSD} + 0.5 V. The trigger condition may unknowingly be met during device power up because of differences in supply ramps.

Pins DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
V_{INMAX}	Maximum input voltage	All allowed values of V_{DDIO} and V_{DDD}	–	–	5.5	V
V_{INREF}	Input voltage reference (Differential input mode)		0.5	–	0.52 × V_{DDIO}	V
V_{OUTREF}	Output voltage reference (Regulated output mode)					
		V_{DDIO} > 3.7	1	–	V_{DDIO} – 1	V
		V_{DDIO} < 3.7	1	–	V_{DDIO} – 0.5	V
V_{IH}	Input voltage high threshold					
	GPIO mode	CMOS input	0.7 × V_{DDIO}	–	–	V
	Differential input mode	Hysteresis disabled	SIO_ref + 0.2	–	–	V
V_{IL}	Input voltage low threshold					
	GPIO mode	CMOS input	–	–	0.3 × V_{DDIO}	V
	Differential input mode	Hysteresis disabled	–	–	SIO_ref – 0.2	V
V_{OH}	Output voltage high					
	Unregulated mode	I_{OH} = 4 mA, V_{DDIO} = 3.3 V	V_{DDIO} – 0.4	–	–	V
	Regulated mode	I_{OH} = 1 mA	SIO_ref – 0.65	–	SIO_ref + 0.2	V
	Regulated mode	I_{OH} = 0.1 mA	SIO_ref – 0.3	–	SIO_ref + 0.2	V
V_{OL}	Output voltage low					
		V_{DDIO} = 3.30 V, I_{OL} = 25 mA	–	–	0.8	V
		V_{DDIO} = 1.80 V, I_{OL} = 4 mA	–	–	0.4	V
R_{PULLUP}	Pull-up resistor		3.5	5.6	8.5	kΩ

Parameter	Description	Conditions	Min	Typ	Max	Units
$R_{PULLDOWN}$	Pull-down resistor		3.5	5.6	8.5	kΩ
I_{IL}	Input leakage current (Absolute value) [1]					
	$V_{IH} \le V_{DDSIO}$	25 °C, V_{DDSIO} = 3.0 V, V_{IH} = 3.0 V	–	–	14	nA
	$V_{IH} > V_{DDSIO}$	25 °C, V_{DDSIO} = 0 V, V_{IH} = 3.0 V	–	–	10	µA
C_{IN}	Input Capacitance [1]		–	–	7	pF
V_H	Input voltage hysteresis (Schmitt-Trigger) [1]	Single-ended mode (GPIO mode)	–	40	–	mV
		Differential mode	–	35	–	mV
I_{DIODE}	Current through protection diode to V_{SSIO}		–	–	100	µA

Pins AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
TriseF	Rise time in fast strong mode (90/10%)[1]	Cload = 25 pF, V_{DDIO} = 3.3 V	–	–	12	ns
TfallF	Fall time in fast strong mode (90/10%)[1]	Cload = 25 pF, V_{DDIO} = 3.3 V	–	–	12	ns
TriseS	Rise time in slow strong mode (90/10%)[1]	Cload = 25 pF, V_{DDIO} = 3.0 V	–	–	75	ns
TfallS	Fall time in slow strong mode (90/10%)[1]	Cload = 25 pF, V_{DDIO} = 3.0 V	–	–	60	ns
Fsioout	SIO output operating frequency					
	3.3 V < V_{DDIO} < 5.5 V, Unregulated output (GPIO) mode, fast strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	33	MHz
	1.71 V < V_{DDIO} < 3.3 V, Unregulated output (GPIO) mode, fast strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	16	MHz
	3.3 V < V_{DDIO} < 5.5 V, Unregulated output (GPIO) mode, slow strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	5	MHz
	1.71 V < V_{DDIO} < 3.3 V, Unregulated output (GPIO) mode, slow strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	4	MHz
	3.3 V < V_{DDIO} < 5.5 V, Regulated output mode, fast strong drive mode	Output continuously switching into 25 pF	–	–	20	MHz

[1] Based on device characterization (Not production tested).

Parameter	Description	Conditions	Min	Typ	Max	Units
	1.71 V < V_{DDIO} < 3.3 V, Regulated output mode, fast strong drive mode	Output continuously switching into 25 pF	–	–	10	MHz
	1.71 V < V_{DDIO} < 5.5 V, Regulated output mode, slow strong drive mode	Output continuously switching into 25 pF	–	–	2.5	MHz
Fsioin	SIO input operating frequency					
	1.71 V < V_{DDIO} < 5.5 V	90/10% V_{DDIO}	–	–	66	MHz

Component Changes

This section lists the major changes in the component from the previous version.

Version	Description of Changes	Reason for Changes / Impact
1.60.a	Minor datasheet edits and updates	
1.60	Added Annotation Terminal capability	Allows pins to connect to Annotation Components.
	Added note about power-on reset for PSoC 5 to datasheet	Clarification
	Added note about API availability for P15[7:6] on PSoC 3 ES2 and PSoC 5 to datasheet	Clarification
1.50.a	The summary has been changed for each of the four pin macros.	Improved readability.
	Added characterization data to datasheet	
	Improved interrupt information in datasheet	
	Added note regarding Vref drive level to datasheet	
	Minor datasheet edits and updates	
1.50	Added Keil function reentrancy support to the APIs.	Add the capability for customers to specify individual generated functions as reentrant.
	Added a sentence to the Reset tab in the Configure dialog clarifying that Power-On Reset applies to an entire physical port.	Clarification.
1.20	Display as Bus now gives an error if checked and the Pins component is not homogeneous. The homogeneous check has been extended to include the HW connections settings. The only changes needed to go from the older version to the new would come from having 'Display as Bus' checked and having some HW connections unchecked.	

© Cypress Semiconductor Corporation, 2011. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

PSoC® is a registered trademark, and PSoC Creator™ and Programmable System-on-Chip™ are trademarks of Cypress Semiconductor Corp. All other trademarks or registered trademarks referenced herein are property of the respective corporations.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Purchase of I^2C components from Cypress or one of its sublicensed Associated Companies, conveys a license under the Philips I^2C Patent Rights to use these components in an I^2C system, provided that the system conforms to the I^2C Standard Specification as defined by Philips.

Use may be limited by and subject to the applicable Cypress software license agreement.