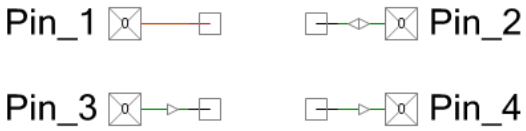
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the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

Pins
2.20

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 allows 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 (e.g., Drive Mode) to be chosen for 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 Application Programming Interfaces (APIs), the Component must be contiguous and non-spanning. 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 (see the Application Programming Interface section for details).

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 many combinations 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

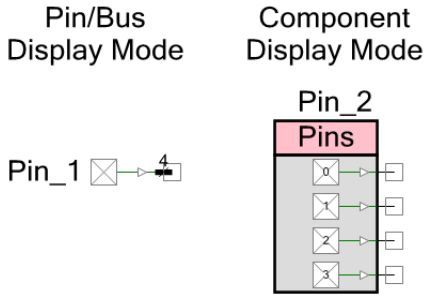
Use the Pins Component when a design must generate or access an off-device signal through a physical IO pin. Pins are the most commonly used Component in the Component Catalog. For example, they are used to interface with potentiometers, buttons, LEDs, and peripheral sensors such as proximity detectors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

Input/Output Connections

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

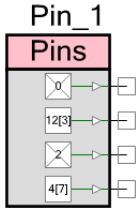
Display of Pins

Pins can be configured into complex combinations of digital input, digital output, digital bidirectional, and analog. Simple configurations are generally shown as single pins. More complex types of pins are shown as standard Components with a bounding box.



Display of Locked Pins

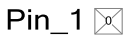
When a Pins Component is assigned to a physical General Purpose IO (GPIO) or Special IO (SIO) pin using the PSoC Creator Design-Wide Resources Pin Editor, the tooltip for the Pins Component shows the specific pin assignments. If a pin assignment is locked, the display of the Component indicates the assignment, 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.

Display of Software Pins

Software pins do not show any terminals as they can only be driven via the CPU/DMA. In order to read from the pin, the input buffer parameter should be enabled. The sync parameter has no effect on software pins.

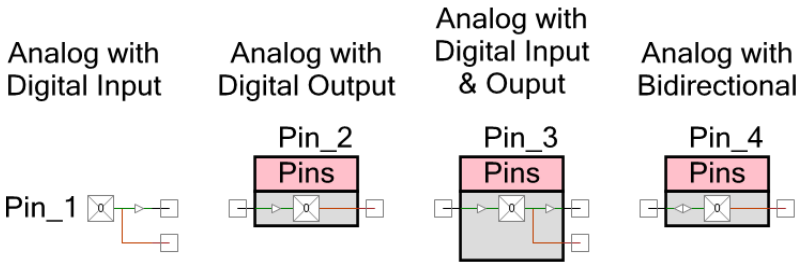


Analog

Configure the Pins Component as Analog any time the 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, but is not a general use case, and should be used with care.



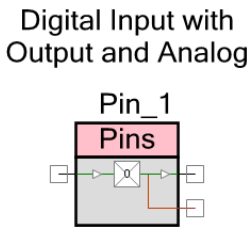
Digital Input

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 schematic terminal (HW Connection) 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.

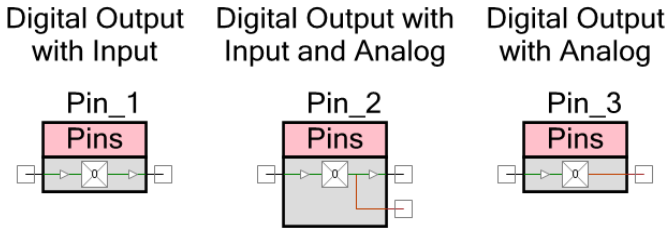


Digital Output

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.

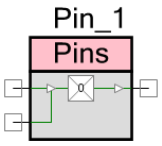
Note The digital output enable terminal is only applicable for hardware signals. It does not apply when the pin is controlled using firmware.

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 Output Enable with Input



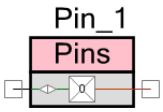
Digital Bidirectional

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



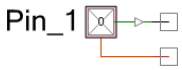
Vref

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.

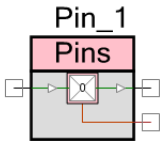
The Vref signal displays on the right side of the Component, extending from the bottom of the SIO single pin or the SIO pin pair, depending on how it is configured. Each SIO pin pair shares a single Vref input.



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

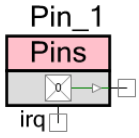


IRQ

To configure a Pins Component with a port-dedicated interrupt, you must use a digital input and configure the **Interrupt** parameter. You must also select the check box that asks whether to use the dedicated port interrupt. 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. This will allow the pin to use the dedicated Port Interrupt to trigger its interrupts.

If the check box to use the dedicated port interrupt is not selected, then the IRQ terminal will not be exposed, and you will not be able to assign an interrupt to use the port specific interrupt. However, the interrupt registers will be configured to allow you to use the Combined Port Interrupt through the Global Signal Reference Component for those devices that support this feature.

Note If the Pin Interrupt is used to wake the part up from deep-sleep or hibernate low-power mode, the Interrupt Component connected to the Pins irq terminal may not have **InterruptType** set to "RISING_EDGE;" it must be set to "LEVEL" or "Derived."



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

For more information on configuring the interrupt, see the **Interrupt** parameter description.

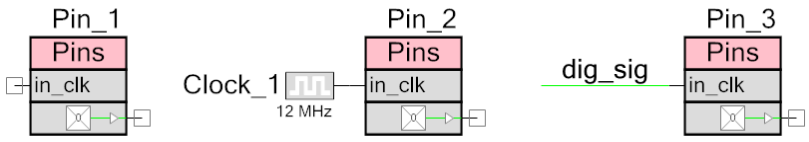
Alternatively, any digital input hardware connection can also be connected to an Interrupt Component, providing the ability to generate a pin interrupt on high or low logic level versus on an edge event. Using the digital input connection as a source for the interrupt does not use the dedicated pin interrupt logic configured with this parameter, but instead consumes digital routing resources.

PSoC 4 Specific Connections

The following terminals are only available on PSoC 4 devices. They can be used when **Sync Mode** is set to other modes besides "transparent."

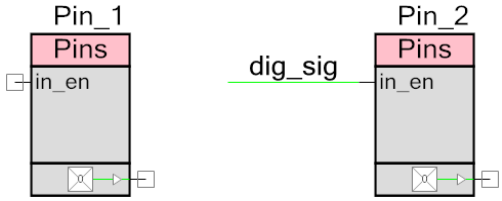
In Clock

On PSoC 4, the Pins Component can use a Clock Component or a digital signal as the clock for the input synchronization logic. If the **In Clock** parameter is specified to be "External," the in_clk terminal is exposed to allow connection on the schematic.



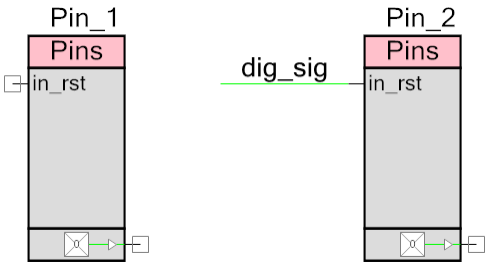
In Clock Enable

On PSoC 4, the Pins Component can use a digital signal as the clock enable for the input synchronization logic. If the **In Clk En** parameter is specified to be "External," the in_en terminal is exposed to allow connection on the schematic.



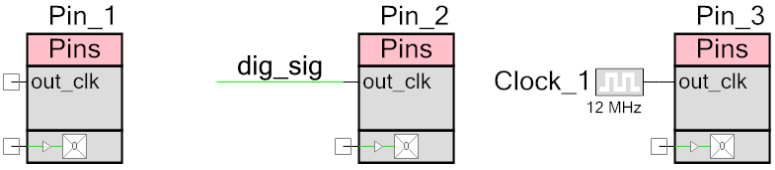
In Reset

On PSoC 4, the Pins Component can use a digital signal as the reset for the input synchronization logic. If the **In Reset** parameter is specified to be "External," the in_rst terminal is exposed to allow connection on the schematic.



Out Clock

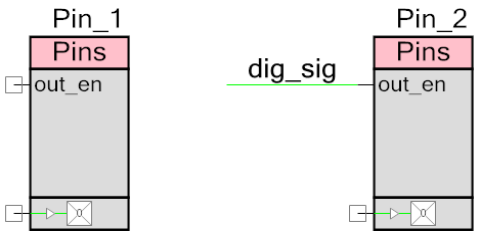
On PSoC 4, the Pins Component can use a Clock Component or a digital signal as the clock for the output synchronization logic. If the **Out Clock** parameter is specified to be "External," the out_clk terminal is exposed to allow connection on the schematic.



Note This configuration can be used to drive the clock signal to a pin on PSoC 4 devices. Refer to the **Output Mode** parameter for more information on its usage.

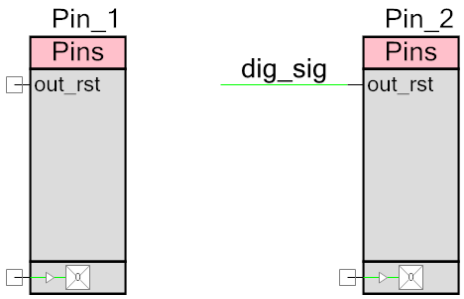
Out Clock Enable

On PSoC 4, the Pins Component can use a digital signal as the clock enable for the input synchronization logic. If the **Out Clk En** parameter is specified to be "External," the out_en terminal is exposed to allow connection on the schematic. If **Output Mode** is set to "Clock" or "Clock-Inverted," then this signal will act as an enable signal to the clock.



Out Reset

On PSoC 4, the Pins Component can use a digital signal as the reset for the output synchronization logic. If the **Out Reset** parameter is specified to be "External," the out_rst terminal is exposed to allow connection on the schematic.

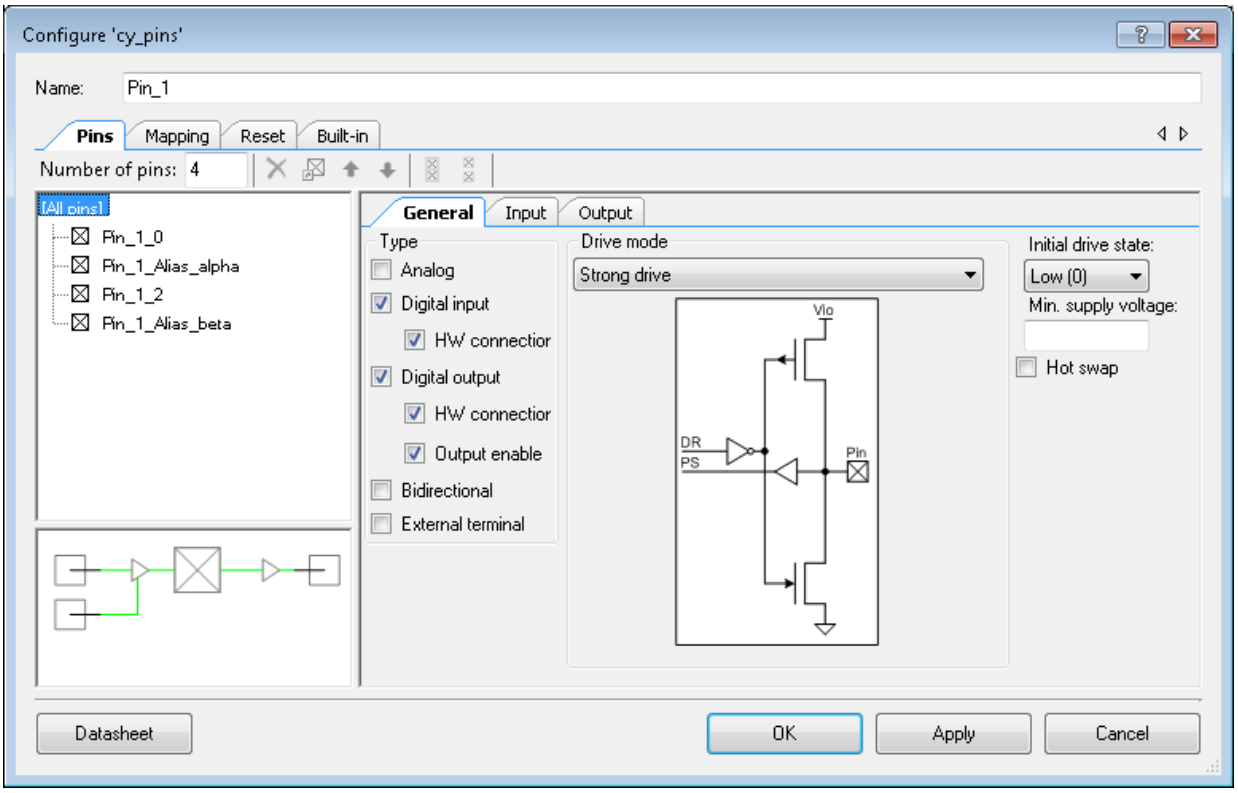


Component Parameters

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.

Pins Tab

The **Pins** tab has three areas: a toolbar, pin tree, and a 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 pin type, drive mode, and initial drive state. The pin tree works with the subtabs to allow you to choose the specific pins to which these attributes are applied.



Toolbar

The 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 selected, **Threshold** is set to anything but "LVTTL" or "CMOS," **Drive Level** set to "Vref," and/or **Current** is set to "25mA sink."

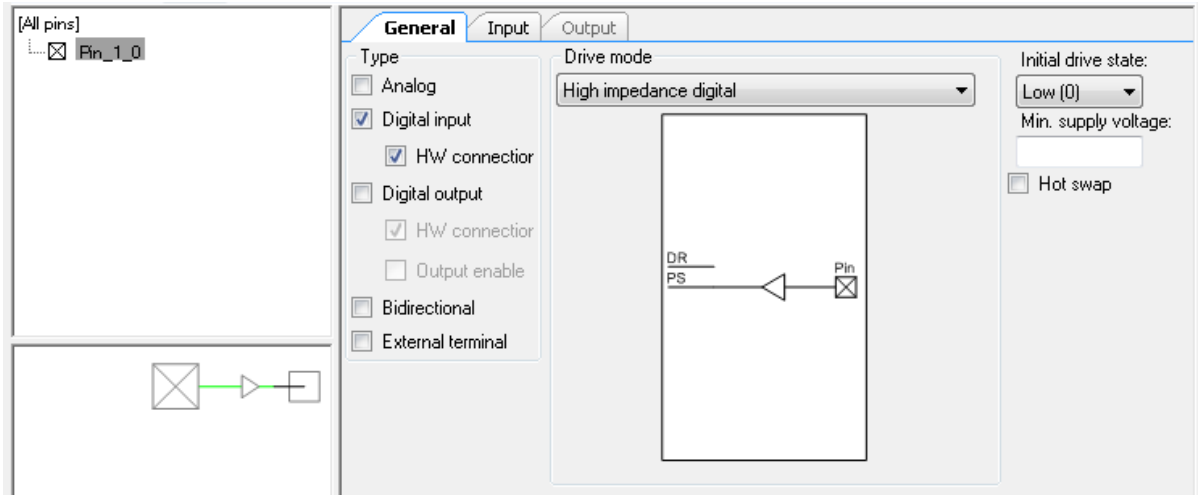
Pin Tree

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.

Below the tree is a preview area that shows what the selected Pins Component symbol will look like with various options selected for that specific pin.

General Subtab

This is the default subtab displayed for the **Pins** tab.



It contains the following parameters:

Type

This is where you choose the type of pins for your Component using the check boxes.

- **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 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.

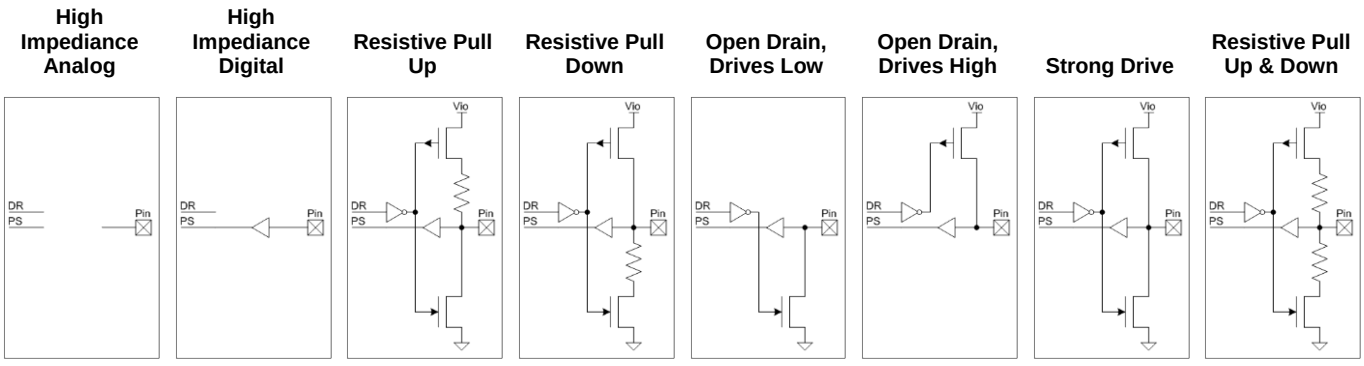
- **Bidirectional** – Enabling the **Bidirectional** parameter is functionally equivalent to enabling the **Digital Input** with **HW Connection** and the **Digital Output** with **HW Connection** parameters. The difference is that only a single bidirectional terminal is displayed on the Component symbol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

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 **Type** selections. 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 **Type** is Analog, the default is High Impedance Analog. This is the only pin drive mode that can support purely Analog pins.
- If the **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:



- The "DR" connection is driven from either the Digital System (when the Digital Output terminal is connected) or the Data Register (when HW Connection is disabled).

- The "PS" connection drives the Pin State register. It also drives the Digital System if the Digital Input terminal is enabled and connected.
- The analog connection connects directly to the pin.

Notes

- 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.
- Direct connection of pins to fixed function hardware blocks on PSoC 4 devices have drive mode limitations. Refer to the device TRM for more information on the restrictions.

  For example, PSoC 4000 devices permit only direct connections to the peripherals. The drive mode is limited for the following peripheral connections.

Peripheral	I/O	Drive Mode Limitation
TCPWM	Input	Limited to HiZ Digital. No other drive modes are allowed
	Output	No restrictions

- For over voltage tolerance to work properly, the pin must be in one of the following **Drive Modes**:
  - High Impedance Analog
  - High Impedance Digital
  - Open Drain, Drives Low
- PSoC 3 and PSoC 5LP USBIO pins only support the "Open Drain, Drives Low" and "Strong Drive" **Drive Modes**. In order to use USBIO pins as inputs, the "Open Drain, Drives Low" option should be used.
- PSoC 4 USB pins can only be software controlled digital pins. It supports all drive modes except for “Resistive Pull Down”, “Open Drain, Drives High” and “Resistive Pull Up/Down”.

Initial Drive State

This parameter specifies the pin-specific initial value written to the pin's Data Register after a device reset/power-on. This happens during port configuration process in device start-up code. Unless changed manually or automatically configured to logic high (1) by the pin **Drive Mode** parameter, all pins default to logic low (0).The initial drive state is configured high (1) by default only for the "Resistive Pull Up" and "Resistive Pull Up/Down" **Drive Modes** to ensure the pull-up resistor is active.

On PSoC 4, the initial drive state is not configurable for hardware digital output pins. This is driven by the signal attached to it and hence the initial drive state does not get set at the output.

Note PSoC 4 device pins default to Hi-Z Analog at device startup. The initial drive state will not take effect until the port configuration in the start-up code.

Note For PSoC 3 and PSoC 5LP devices, this should not be confused with the **Power-On Reset** setting 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 (including Initial drive state 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 Design-Wide Resources System Editor (in the *<project>.cydwr* file) for V_{DDIO0}/V_{DDIO1}/V_{DDIO2}/V_{DDIO3}, or V_{DDD} on devices that do not have independent V_{DDIO} settings. 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.

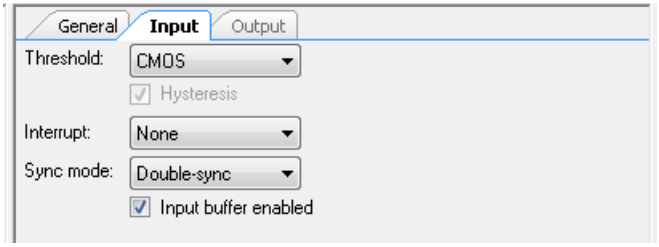
Hot Swap

A pin configured for hot swap capability is mapped to an SIO or a GPIO Over-Voltage Tolerance (GPIO_OVT) 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 or GPIO_OVT

Input Subtab

The **Input** subtab specifies input settings. If the **Type** is not "Digital Input" or "Bidirectional," 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) for the entire port on which that pin is placed. "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 those of CMOS. A pin specified as "CMOS or LVTTL" will default to CMOS, but may be configured as LVTTL in order to be placed in a port configured as LVTTL. Thresholds that are derived from Vddio, a routed Vref, or an Internal Vref (1.2 V) require the use of an SIO pin. This setting is on a port, so if it is in the same group where there are different settings, the port cannot be contiguous.

Threshold	Allowed Multiplier values			Notes
	PSoC 3 / PSoC 5LP	PSoC 4 without SIO	PSoC 4 with SIO	
CMOS	1	1	1	**Default:** Applicable for all devices
LVTTL	1	1	1	Applicable for all devices
CMOS or LVTTL	1	1	1	Applicable for all devices
CMOS 1.8V	N/A	1	1	PSoC 4 only (excluding PSoC 4000 / PSoC 4100 / PSoC 4200)
Vddio	0.4, 0.5	N/A	0.4, 0.5	Requires SIO
Vref	1	N/A	1	Requires SIO
0.5 x Vref	1	N/A	All	Requires SIO
Vref (Internal)	N/A	N/A	1	Requires SIO (PSoC 4 only)
0.5 x Vref (Internal)	N/A	N/A	All	Requires SIO (PSoC 4 only)

The list of possible multiplier values include:

- 1.00 x - Default
- 1.25 x
- 1.49 x
- 1.67 x
- 2.08 x
- 2.50 x
- 2.78 x
- 4.16 x

Note For PSoC 4 SIO, Vref and Internal Vref share resources with the Drive Level parameter in the **Output** subtab. Therefore, they need to match if using these options in a pin configured as both Digital input pin and a Digital output.

Note When using **0.5 x Vref** or **0.5 x Vref (Internal)** options, the voltage on Vref or the 1.2 V Vref (Internal) must adhere to the lower limit of 1 V and the upper limit of (VDD – 0.4 V). That is,

1 V < (Vref x multiplier) < (VDD - 0.4 V)

Operating beyond this limit will result in incorrect operation. The Vref Component in the relation above is the full Vref, **NOT** 0.5 x Vref.

Hysteresis

This parameter is used to configure the pin hysteresis. The hysteresis is always enabled for all pins except for SIOs in PSoC 3 and PSoC 5LP. In this case, the parameter allows the differential hysteresis to be enabled or disabled.

- Disabled – Default for PSoC 3 or PSoC 5LP SIOs
- 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 via dedicated Port Interrupt.

This parameter uses dedicated pin interrupt logic, which latches the pins that generated the interrupt 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.

Note Some ports on PSoC 4 devices do not have dedicated port interrupts that can wake up the device from deep-sleep. If this is a feature you wish to use on that particular port, then the Combined Port Interrupt (**AllPortInt**) signal provided by the Global Signal Reference Component can be used. To use this feature, the **Dedicated Interrupt** checkbox should be unchecked. Refer to the parameter description and the Functional Description section for more information.

The following interrupt options are supported:

- None – Default
- Rising Edge
- Falling Edge

- Both Edges

Dedicated Interrupt

Ports on certain devices do not have dedicated Port Interrupts capable of waking up the chip from deep-sleep. They can all however provide port-wide wakeup from chip sleep mode. Check the **Dedicated Interrupt** check box to be able to use the port dedicated interrupt logic, if you do not intend to use it for wakeup from deep-sleep mode.

If however you do wish to use the pin as a wakeup source from chip deep-sleep, then refer to the TRM to see if that particular port has the capability of waking up the device from chip deep-sleep.

For certain devices such as PSoC 4200M and PSoC 4200L, you may use the Global Signal Reference Component, which provides a Combined Port Interrupt (**AllPortInt**) signal that gets triggered whenever an interrupt condition is met on any of the interrupt-enabled pins on the device. This resource allows you to wake-up the device from chip deep-sleep. Uncheck the **Dedicated Interrupt** checkbox to use this feature.

Note The Combined Port interrupt triggers also for pins on all enabled ports with dedicated port interrupts. Refer to the Functional Description section for usage information.

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 **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

Sync Mode

Input synchronization occurs by default at pins to synchronize all signals entering the device to the input clock using a double-synchronizer (Double-Sync). On PSoC 3 and PSoC 5LP, the input clock is always BUS_CLK. On PSoC 4, the input clock defaults to HFCLK, but may be selected via the **In Clock** parameter.

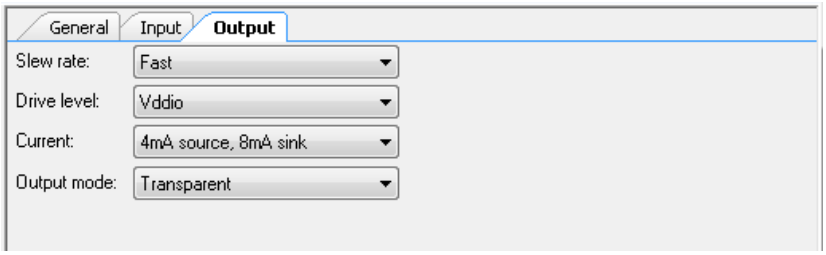
Input synchronization can be optionally disabled at the pin in limited cases in which an asynchronous signal is required for application performance and does not violate device operational requirements (Transparent). On PSoC 4, synchronization can also be performed using a single flip-flop (Single-Sync).

- Double-Sync – Default

- Single-Sync (PSoC 4 only)
- Transparent

Output Subtab

The **Output** subtab specifies output settings. If the **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.

On GPIO pins, 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.

On SIO pins, fast mode is required for signals that switch at greater than 10 MHz You can select slow mode for signals less than 10 MHz switching rate and benefit from lower power usage by the SIO output buffer.

For devices supporting GPIO_OVT, I2C FM+, and I^2C HS, options can be used for improving the slew rate to meet FM+ and I^2C HS specification. This option also requires the **Minimum Supply Voltage** to be defined.

Note On PSoC 4 devices, all pins on the same port must have the same Slew Rate.

- Fast – Default
- Slow
- I2C FM+ – Requires GPIO_OVT
- I2C HS <= 1.7 Mbps – Requires GPIO_OVT
- I2C HS > 1.7 Mbps – Requires GPIO_OVT

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. SIOs in PSoC 4 can use a routed external Vref or an internal Vref (1.2 V) as the drive voltage supply.

Threshold	Allowed Multiplier values			Notes
	PSoC 3 / PSoC 5LP	PSoC 4 without SIO	PSoC 4 with SIO	
Vddio	1	1	1	**Default** – Applicable for all devices
Vref	1	N/A	All	Requires SIO
Vref (Internal)	N/A	N/A	All	Requires PSoC 4 SIO

The list of multiplier values include:

- 1.00 x - Default
- 1.25 x
- 1.49 x
- 1.67 x
- 2.08 x
- 2.50 x
- 2.78 x
- 4.16 x

Note For PSoC 4 SIO, external Vref and Internal Vref share resources with the **Threshold** parameter. Therefore they need to match if using these options in a pin configured as both Digital input pin and a Digital output pin.

Note: When using **Vref** or **Vref (Internal)** options, the voltage on Vref or the 1.2V Vref (Internal) must adhere to the lower limit of 1V and the upper limit of (VDD – 0.4V). That is:

1V < (Vref x multiplier) < (VDD - 0.4V)

Operating beyond this limit will result in incorrect operation.

Note In PSoC 3 and PSoC 5LP devices, 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. For PSoC 4 devices, the drive mode must be Strong Drive if you are using either **Vref** or **Vref (Internal)**.

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, 10mA sink – Requires GPIO_OVT or SIO (PSoC 4 only)
- 4mA source, 25mA sink – Requires SIO

Output Mode

Output synchronization reduces pin-to-pin output signal skew in high-speed signals requiring minimal signal skew. By default, this parameter is set to "Transparent" and no synchronization occurs. If "Single-Sync" is selected, the output signal is synchronized to the output clock.

- Transparent – Default
- Single-Sync
- Clock (PSoC 4 only)
- Clock-Inverted (PSoC 4 only)

On PSoC 3 and PSoC 5LP, the output clock is always BUS_CLK.

On PSoC 4, the output clock defaults to HFCLK, but may be changed via the **Out Clock** parameter.

Choosing either "Clock" or "Clock-Inverted" output modes on PSoC 4 allows the externally connected clock or signal to drive the pin. In this configuration, the data register (DR) value is used as an enable to the out_clock terminal and must be set high either by initially setting it to 1 in the pin customizer or set using software. Note that if this is a HW output pin then the value of the signal connected to the output pin terminal does not affect the operation of the pin when in this mode. Instead, hardware control of the clock enable can be achieved if **Out Clock Enable** is used with DR set high.

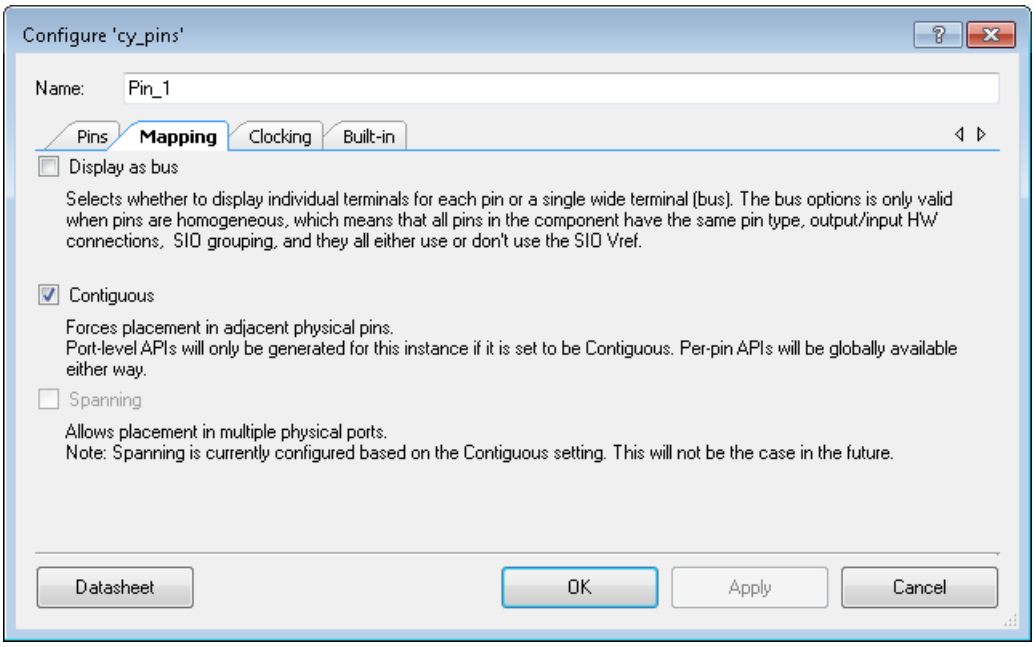
OE Synchronized (PSoC 4 only)

Output Enable synchronization allows the Output Enable signal to be synchronized to the output clock.

- 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:

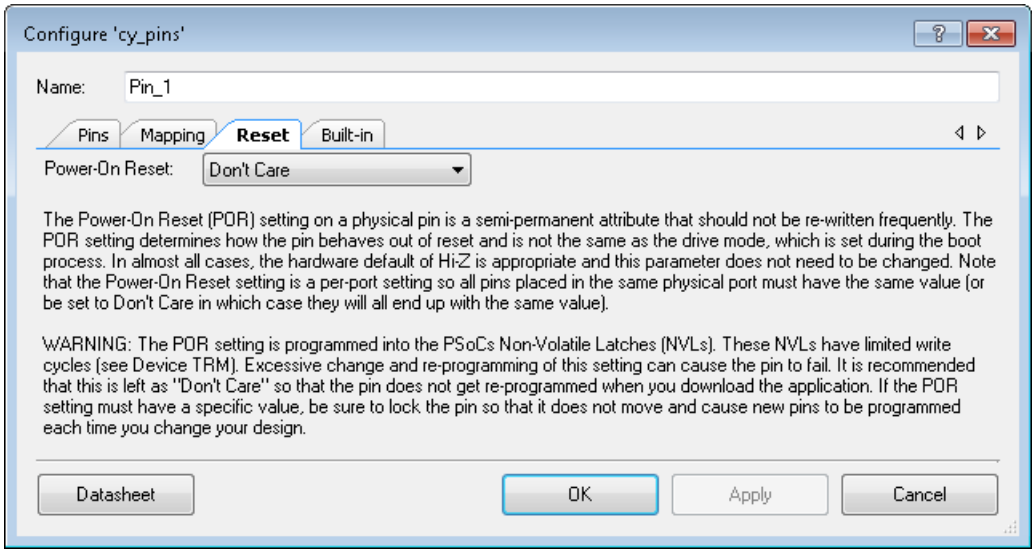
- If contiguous, port-level APIs are generated for the Component. If noncontiguous, port-level APIs are not generated.
- If contiguous, the number of pins in the Component must be less than or equal to 8.

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.

Reset Tab

The **Reset** tab is only available on PSoC 3 and PSoC 5LP.



Power-On Reset

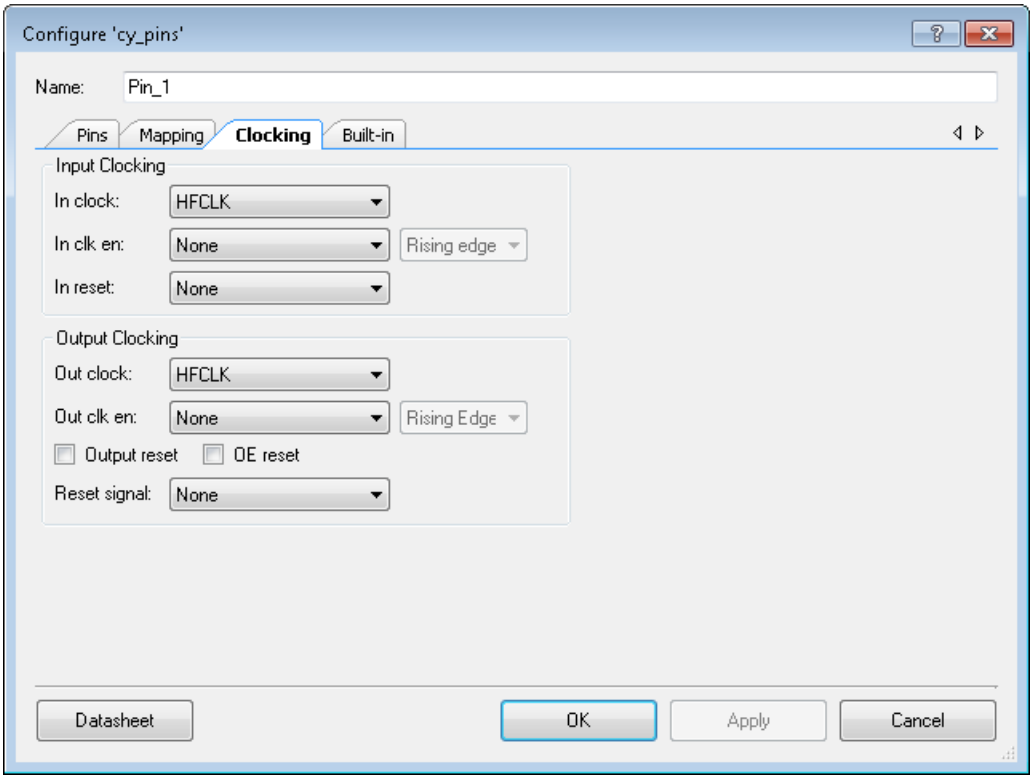
The Power-On Reset (POR) setting on a physical pin is a semi-permanent attribute that you should not rewrite frequently. The POR setting determines how the pin behaves out of reset. It is not the same as the drive mode, which is set during the boot process. In almost all cases, the hardware default of HI-Z is appropriate and you do not need to change this parameter. Note that the POR 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).

Warning: The POR setting is programmed into the PSoC's Non-Volatile Latches (NVLs). These NVLs have limited write cycles (see Device TRM). Excessive change and re-programming of this setting can cause the pin to fail. It is recommended that this is left as **Don't Care** so that the pin does not get re-programmed 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 (Hi-Z Analog) 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

Clocking Tab

The **Clocking** tab is only available on PSoC 4.



In Clock

This parameter selects the clock to use for the input synchronization logic of this Component. By default, HFCLK is used as the input synchronization clock. It is also possible to use a Clock Component or other signal by enabling the in_clk terminal (External). To use an off-chip signal as the input clock with minimal skew, any pin in this Component configured as Input or Bidirectional may be selected as the In Clock (Pin_N). Any of the options may also be inverted inside the Pins Component.

- HFCLK – Default
- HFCLK (inverted)
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

In Clk En

This parameter selects a signal to use as the enable signal for the input synchronization logic of this Component. By default, no enable signal is used, and input synchronization is always enabled. A digital signal from the schematic may be used as the enable by displaying and connecting the in_en terminal (External). To use an off-chip signal as the enable signal with minimal delay, any pin in this Component configured as Input or Bidirectional may be selected as the In Clk En (Pin_N). Any of the options may also be inverted inside the Pins Component.

- None – Default
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

In Clock Enable Mode

The drop-down to the right of the **In Clk En** parameter controls the Enable Mode of the In Clock Enable. If the Enable Mode is set to Rising Edge, the input synchronization flip flops will only transition on the clock cycle immediately after the enable signal transitions from low to high. If the Enable Mode is set to Level, the input synchronization flip flops can transition on any clock cycle when the enable signal is high.

- Rising Edge – Default
- Level

In Reset

This parameter selects a signal to use as the reset signal for the input synchronization logic of this Component. By default, reset is not used. A digital signal from the schematic may be used as the enable by displaying and connecting the in_rst terminal (External). To use an off-chip signal as the reset signal with minimal delay, any pin in this Component configured as Input or Bidirectional may be selected as the In Reset (Pin_N). Any of the options may also be inverted inside the Pins Component.

- None – Default
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

Out Clock

This parameter selects the clock to use for the output synchronization logic of this Component. By default, HFCLK is used as the output synchronization clock. It is also possible to use a Clock Component or other signal by enabling the out_clk terminal (External). To use an off-chip signal as the output clock with minimal skew, any pin in this Component configured as Input or Bidirectional may be selected as the In Clock (Pin_N). Any of the options may also be inverted inside the Pins Component.

- HFCLK – Default
- HFCLK (inverted)
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

Out Clk En

This parameter selects a signal to use as the enable signal for the output synchronization logic of this Component. By default, no enable signal is used, and output synchronization is always enabled. A digital signal from the schematic may be used as the enable by displaying and connecting the out_en terminal (External). To use an off-chip signal as the enable signal with minimal delay, any pin in this Pins Component configured as Input or Bidirectional may be selected as the In Clk En (Pin_N). Any of the options may also be inverted inside the Pins Component.

- None – Default
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

Out Clock Enable Mode

The drop-down to the right of the **Out Clk En** parameter controls the Enable Mode of the Out Clock Enable. If the Enable Mode is set to Rising Edge, the output synchronization flip flop will only transition on the clock cycle immediately after the enable signal transitions from low to high. If the Enable Mode is set to Level, the output synchronization flip flop can transition on any clock cycle when the enable signal is high.

- Rising Edge – Default
- Level

Output Reset

This allows the selected Out Reset Signal to be used to reset the output synchronization logic.

- Disabled – Default
- Enabled

OE Reset

This allows the selected Out Reset Signal to be used to reset the output enable synchronization logic.

- Disabled – Default
- Enabled

Out Reset Signal

This selects a signal to use as the reset signal for either the output synchronization logic or the output enable synchronization logic of this Component. By default, reset is not used. A digital signal from the schematic may be used as the enable by displaying and connecting the out_rst terminal (External). To use an off-chip signal as the reset signal with minimal delay, any pin in this Component configured as Input or Bidirectional may be selected as the Out Reset (Pin_N). Any of the options may also be inverted inside the Pins Component.

- None – Default
- External
- External (inverted)
- Pin_N
- Pin_N (inverted)

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 basis, as well as on Component-wide basis. The preferred method of using Pins Component is to use the Component-wide API for pins that are contiguous. If they are non-contiguous, then the Per-Pin API should be used.

The API functions access all pins in the Component in a single function call. Efficient implementation of Component-wide function is only possible if all pins are placed in a single physical port on the device. They are generated only if the Component is configured to be contiguous. Non-contiguous Pins Components only allow access on the per-pin basis described under Per-Pin API.

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."

General API

General API functions are used for run-time configuration of the component during active power mode.

These include, initializing, starting, stopping, reading from registers and writing to registers.

Functions

- uint8 Pin_Read(void)
  Reads the associated physical port (pin status register) and masks the required bits according to the width and bit position of the component instance.
- void Pin_Write(uint8 value)
  Writes the value to the physical port (data output register), masking and shifting the bits appropriately.
- uint8 Pin_ReadDataReg(void)
  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.
- void Pin_SetDriveMode(uint8 mode)
  Sets the drive mode for each of the Pins component's pins.
- void Pin_SetInterruptMode(uint16 position, uint16 mode)
  Configures the interrupt mode for each of the Pins component's pins. Alternatively you may set the interrupt mode for all the pins specified in the Pins component.
- uint8 Pin_ClearInterrupt(void)
  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.

Function Documentation

uint8 Pin_Read (void)

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.

Returns:

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

Function Usage

```
{
   uint8 pinState;
   /* Read the state of a digital pin */
   pinState = Pin_Read();
}
```

void Pin_Write (uint8 *value*)

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).

Note This function should not be used on a hardware digital output pin as it is driven by the hardware signal attached to it.

Parameters:

value	Value to write to the component instance.

Returns:

None

Side Effects

If you use read-modify-write operations that are not atomic; the Interrupt Service Routines (ISR) can cause corruption of this function. An ISR that interrupts this function 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 function.

Function Usage

```
{
   #define LED_ON      1u
   #define LED_OFF     0u
   /* Turn on an LED connected to a SW controlled pin */
   Pin_Write(LED_ON);
}
```

uint8 Pin_ReadDataReg (void)

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 function to determine the value just written to the pin.

Returns:

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

Function Usage

```
{
   uint8 dataRegVal;
   /* Write a logic 1 to pin 0 and pin 2 of "Pin" instance */
   Pin_Write(0x05);
   /* Read the data register of the Pin component */
   dataRegVal = Pin_ReadDataReg();
}
```

void Pin_SetDriveMode (uint8 *mode*)

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

Note This affects all pins in the Pins component instance. Use the Per-Pin APIs if you wish to control individual pin's drive modes.

Note USBIOs have limited drive functionality. Refer to the Drive Mode parameter for more information.

Parameters:

mode	Mode for the selected signals. Valid options are documented in Drive mode constants.

Returns:

None

Side Effects

If you use read-modify-write operations that are not atomic, the ISR can cause corruption of this function. An ISR that interrupts this function 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 function.

Function Usage

```
{
   /* Set the Pin to analog high-Z to prepare for deep-sleep */
   Pin_SetDriveMode (Pin_DM_ALG_HIZ);

   /* Call Sleep() function of a component connected to the Pin */

   /* Put the chip into deep-sleep mode */
   CySysPmDeepSleep();
}
```

void Pin_SetInterruptMode (uint16 *position*, uint16 *mode*)

Configures the interrupt mode for each of the Pins component's pins. Alternatively you may set the interrupt mode for all the pins specified in the Pins component.

Note The interrupt is port-wide and therefore any enabled pin interrupt may trigger it.

Parameters:

position	The pin position as listed in the Pins component. You may OR these to be able to configure the interrupt mode of multiple pins within a Pins component. Or you may use Pin_INTR_ALL to configure the interrupt mode of all the pins in the Pins component. • Pin_0_INTR (First pin in the list) • Pin_1_INTR (Second pin in the list) • ... • Pin_INTR_ALL (All pins in Pins component)
mode	Interrupt mode for the selected pins. Valid options are documented in Interrupt constants.

Returns:

None

Side Effects

It is recommended that the interrupt be disabled before calling this function to avoid unintended interrupt requests. Note that the interrupt type is port wide, and therefore will trigger for any enabled pin on the port.

Function Usage

```
{
   /* Turn off the interrupt capability of pin 0 and pin 1 of Pin instance */
   Pin_SetInterruptMode(Pin_0_INTR | Pin_1_INTR, Pin_INTR_NONE);
}
```

uint8 Pin_ClearInterrupt (void)

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.

Returns:

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.

Function Usage

```
{
   /* This code is placed inside the ISR of the port interrupt connected to the Pin.
      It may also be placed inside the ISR of the AllPortInt signal from the GSRef
      component for those devices that support this functionality. */

   uint8 intrSrc;

   /* Clear the port interrupt and determine which pin triggered it */
   intrSrc = Pin_ClearInterrupt();
}
```

Power Management API

Power management API functions perform the necessary configurations to the components to prepare it for entering low power modes.

These functions must be used if the intent is to put the chip to sleep, then to continue the component operation when it comes back to active power mode.

Functions

- void Pin_Sleep(void)
  Stores the pin configuration and prepares the pin for entering chip deep-sleep/hibernate modes. This function must be called for SIO and USBIO pins. It is not essential if using GPIO or GPIO_OVT pins.
- void Pin_Wakeup(void)
  Restores the pin configuration that was saved during Pin_Sleep().

Function Documentation

void Pin_Sleep (void)

Stores the pin configuration and prepares the pin for entering chip deep-sleep/hibernate modes. This function applies only to SIO and USBIO pins. It should not be called for GPIO or GPIO_OVT pins.

Note This function is available in PSoC 4 only.

Returns:

None

Side Effects

For SIO pins, this function configures the pin input threshold to CMOS and drive level to Vddio. This is needed for SIO pins when in device deep-sleep/hibernate modes.

Function Usage

```
{
   /* Place the SIO Pin in correct state before entering deep sleep */
   Pin_Sleep();
   /* Put the chip into deep-sleep mode */
```

```
        CySysPmDeepSleep();
        /* The chip woke up from deep-sleep. Restore the SIO settings */
        Pin_Wakeup();
    }
```

void Pin_Wakeup (void)

Restores the pin configuration that was saved during Pin_Sleep().This function applies only to SIO and USBIO pins. It should not be called for GPIO or GPIO_OVT pins.

For USBIO pins, the wakeup is only triggered for falling edge interrupts.

Note This function is available in PSoC 4 only.

Returns:

None

Function Usage

Refer to Pin_Sleep() for an example usage.

API Constants

Component API functions are designed to work with pre-defined enumeration values.

These values should be used with the functions that reference them.

Modules

- Drive mode constants
  Constants to be passed as "mode" parameter in the Pin_SetDriveMode() function.
- Interrupt constants
  Constants to be passed as "mode" parameter in Pin_SetInterruptMode() function.

Data Structures

Data Structures are used to group related elements.

The following data structures are used by the component API functions.

Data Structures

- struct Pin_BACKUP_STRUCT

Deprecated Code

Deprecated code that is no longer used by the component.

This component contains deprecated code that is not recommended for use but is kept to preserve backward compatibility with the existing designs. Deprecated code should no longer be used in new projects.

Macros

- #define **Pin_DRIVE_MODE_SHIFT** (0x00u)
- #define **Pin_DRIVE_MODE_MASK** (0x07u << Pin_DRIVE_MODE_SHIFT)

Drive mode constants

Constants to be passed as "mode" parameter in the Pin_SetDriveMode() function.

Macros

- #define Pin_DM_ALG_HIZ (0x00u)
  High Impedance Analog.
- #define Pin_DM_DIG_HIZ (0x01u)
  High Impedance Digital.
- #define Pin_DM_RES_UP (0x02u)
  Resistive Pull Up.
- #define Pin_DM_RES_DWN (0x03u)
  Resistive Pull Down.
- #define Pin_DM_OD_LO (0x04u)
  Open Drain, Drives Low.
- #define Pin_DM_OD_HI (0x05u)
  Open Drain, Drives High.
- #define Pin_DM_STRONG (0x06u)
  Strong Drive.
- #define Pin_DM_RES_UPDWN (0x07u)
  Resistive Pull Up/Down.

Interrupt constants

Constants to be passed as "mode" parameter in Pin_SetInterruptMode() function.

Macros

- #define Pin_INTR_NONE ((uint16)(0x0000u))
  Disabled.
- #define Pin_INTR_RISING ((uint16)(0x5555u))
  Rising edge trigger.
- #define Pin_INTR_FALLING ((uint16)(0xaaaau))
  Falling edge trigger.
- #define Pin_INTR_BOTH ((uint16)(0xffffu))
  Both edge trigger.

Data Structure Documentation

Pin_BACKUP_STRUCT Struct Reference

Data Fields

- uint32 pcState
- uint32 sioState
- uint32 usbState

Field Documentation

uint32 Pin_BACKUP_STRUCT::pcState

State of the port control register

uint32 Pin_BACKUP_STRUCT::sioState

State of the SIO configuration

uint32 Pin_BACKUP_STRUCT::usbState

State of the USBIO regulator

Per-Pin API

You can access individual pins in the Component separately by using the global API functions defined in the *cypins.h* generated file (in the cy_boot directory). These API functions are documented in the *System Reference Guide* (Help in PSoC Creator toolbar).

To use the per-pin API functions, the Pins Component generates aliases for the pin registers in the *Pin_aliases.h* file, where "Pin" is the instance name of the Pins Component. By default the alias is the Component name with the pin number appended to it:

Pin_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:

Pin_<AliasName>

Either of these aliases can be used. For example, "Pin" has an <AliasName> called "MyAlias." This generates aliases, Pin_0 and Pin_MyAlias. To read this pin using the per-pin API function, you can use either of these methods:

```
CyPins_ReadPin(Pin_0)
CyPins_ReadPin(Pin_MyAlias)
```

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.

MISRA Compliance

This section describes the MISRA-C:2004 compliance and deviations for the Component. There are two types of deviations defined:

- project deviations – deviations that are applicable for all PSoC Creator Components
- specific deviations – deviations that are applicable only for this Component

This section provides information on Component specific deviations. The project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The Pins Component is MISRA Compliant, except for following specific deviations.

MISRA rule	Rule class	Rule Description	Deviation Description
10.1	R	The value of an expression of integer type shall not be implicitly converted to a different underlying type if: a) it is not a conversion to a wider integer type of the same signedness, or b) the expression is complex, or c) the expression is not constant and is a function argument, or d) the expression is not constant and is a return expression.	Assigned const value to the unsigned variable do not have any negative impact or side effect as constant is positive and is less than max byte value. The result of the operation is explicitly casted to an unsigned type.

API Memory Usage

The Component memory usage varies significantly, depending on the compiler, device, number of APIs used and Component configuration. The following table provides the memory usage for all APIs available in the given Component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design, the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 4 (GCC)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Default with interrupt	60	0	112	0	112	0
Default with interrupt and LPM API functions	-	-	152	12	-	-

Functional Description

The pins Component allows easy configuration of common pin settings in most designs. It also provides more advanced configurations for those designs requiring settings beyond the basic functionality. This section highlights some of the more advanced pin modes that may not be obvious from the given parameter descriptions.

- **SIO Pins –** The Special Input/Output SIO pins provide differential input buffer and a means to regulate the high-level output voltage (VOH). The SIO pins are tolerant to input voltages higher than the I/O supply voltage and can sink up to 25 mA current. There are several ways to choose an SIO pin in your design. A pin requires SIO if any of the following parameters are set.
  - **Hot Swap** is set to true
  - **Threshold Level** is set to Vddio, 0.5 x External/Internal Vref, External/Internal Vref
  - **Drive Level** is set to External/Internal Vref,
  - **Drive Current** is set to 25mA sink.

  Two SIO pins can be paired from the Component configuration, which allows them to share a common reference generator block.

- **POR –** Power on Reset (POR) option is available on PSoC 3 and PSoC 5LP. The POR setting on a physical pin is a semi-permanent attribute that should not be re-written frequently. The POR setting determines how the pin behaves out of reset. The setting is port-wide and 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 this parameter does not need to be changed.

- **OVT Pins** – Some GPIOs have an Over-Voltage Tolerance (OVT) feature that allows them to withstand higher voltages than the specified levels. A GPIO_OVT pin is used if any of the following parameters is set.
  - **Hot Swap** is set to true
  - **Slew Rate** is set to I2C FM+, or I2C HS
  - **Drive Current** is set to 10mA sink

  Note: GPIO_OVT pins are functionally identical to regular GPIOs. However, they contain these extra features and allow tolerance of voltages higher than the I/O supply voltage.

- **PSoC 4 pin clocking –** The input and output values can be synchronized with an external clock using the PSoC 4 pin clocking options. Use this if the pin state needs to be synchronized with other clocks aside from HFCLK.

- **Route PSoC 4 clock to pin –** Unlike PSoC 3 and PSoC 5LP, clocks in PSoC 4 cannot be connected directly to a pin terminal unless it is specified to be a clock. To enable this mode, the **Output Mode** parameter can be set to either Clock or Clock-Inverted.

- **PSoC Port Adapter –** Some devices, such as PSoC 4000 series and specific ports on PSoC 4 devices, do not have Port Adapters. This limits some pin features such as input **Sync Mode**, **Output Mode** and PSoC 4 pin clocking options. Keep this in mind when migrating devices or ports and consult the device TRM.
- **Port Interrupts –** Port interrupts allow signal transitions on the port pins to trigger an interrupt. To use these, you must connect a LEVEL or Derived interrupt type to the IRQ terminal of the Pins Component. This interrupt can be triggered in all power modes. If you choose to connect a RISING_EDGE interrupt to either the IRQ terminal or the Pin I/O terminal, then routing resources will be used and the interrupt will be limited to active and sleep power modes.
- **PSoC 4 dedicated port interrupts –** Some ports on PSoC 4 devices such as ports 5, 6 and 7 on PSoC 4100M / 4200M do not have Port Interrupts that can provide wakeup from chip deep-sleep mode. This means that although you may connect an interrupt to the IRQ terminal on the Pins Component, the interrupt will only be able to provide wakeup functionality from chip sleep mode. If you wish to wake up the device from chip deep-sleep or hibernate using pins on these ports, you must use another hardware resource called Combined Port Interrupt (**AllPortInt**) signal provided by the Global Signal Reference Component. Refer to Combined_Port_Interrupt Code Example for sample usage.

  To use this feature, follow these instructions:

  1. Configure the pin to generate an interrupt using the Interrupt parameter.
  2. If the pin does not have a port adapter (as described in PSoC Port Adapter), then make sure that those settings are configured to not use those features.
  3. Uncheck the **Dedicated Interrupt** check box. This will configure the interrupt generation registers to be able to use the Combined Port Interrupt, while not having to use the dedicated port interrupt.

     Note It is also possible for you to use both the dedicated port interrupt and the **AllPortInt** signal. However you will need to handle both ISRs as they will both need to be serviced. That is, the **AllPortInt** signal triggers even for the interrupts triggered via dedicated port interrupt.
  4. Use the Global Signal Reference Component's **AllPortInt** signal and attach an Interrupt Component to it.

     Note As mentioned previously, the **AllPortInt** signal triggers even for the interrupts triggered via the dedicated port interrupt. Therefore, if you use the dedicated port interrupt on a port, then that interrupt should be higher priority than the **AllPortInt** interrupt so that it gets serviced first. Otherwise, the interrupt trigger sequence will not be predictable. Hence identifying and clearing the interrupt source will be difficult.
  5. In the ISR of the interrupt, insert a routine to check for the interrupt status register (Pin_INTSTAT, where Pin is the instance name of the Component) for the port that the pin is placed in. Then call Pin_ClearInterrupt() API to clear that particular port's

interrupt. Repeat the process for all ports in the ISR. You may also use the Pin_INTR_CAUSE register to determine which port caused the interrupt to trigger.

Note If the ISR for the dedicated port interrupt was already serviced and the interrupt was cleared, then you do not need to check for that port in the **AllPortInt** ISR.

Resources

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

DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ T_A ≤ 85 °C and T_J ≤ 100 °C, except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted.

Note Final characterization data for PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

PSoC 4 Pins DC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
V_{IH}	Input voltage high threshold					
	CMOS Input	Must not exceed VDDD + 0.2 V	0.7 x VDDD	–	–	V
	LVTTL Input	VDDD ≥ 2.7 V, Must not exceed VDDD + 0.2 V	2.0	–	–	V
	Differential input (SIO)	Hysteresis Disabled	Vref + 0.2	–	–	V
V_{IL}	Input voltage low threshold					
	CMOS input		–	–	0.3 x VDDD	V
	LVTTL input	VDDD ≥ 2.7 V	–	–	0.8	V
	Differential input (SIO)	Hysteresis Disabled	–	–	Vref + 0.2	V
V_{OH}	Output voltage high level					
	High level at 3V	IOH = 4 mA at 3 V VDDD	VDDD – 0.6	–	–	V
	High level at 1.8 V	IOH = 1 mA at 1.8 V VDDD	VDDD – 0.5	–	–	V
	SIO, unregulated mode	IOH = 4 mA at 3.3 V VDDD	VDDD – 0.4	–	–	V
	SIO regulated mode	IOH= 1 mA	Vref – 0.65	–	Vref + 0.2	V
		IOH= 0.1 mA	Vref – 0.3	–	Vref + 0.2	V
V_{OL}	Output voltage low level					
	Low level at 1.8 V	IOL = 4 mA at 1.8 V VDDD	–	–	0.6	V
	Low level at 3 V	IOL = 10 mA at 3 V VDDD	–	–	0.6	V

Parameter	Description	Conditions	Min	Typ	Max	Units
	Low level at 3 V, less IOL	IOL = 3 mA at 3 V VDDD	–	–	0.4	V
	OVT GPIO	IOL = 20 mA, VDDD > 2.9 V	–	–	0.4	V
	SIO, regulated mode	IOL = 25 mA, VDDIO = 3.3 V	–	–	0.8	V
		IOL= 4 mA, VDDIO= 1.8 V	–	–	0.4	V
V_{inref}	SIO, Input voltage reference		0.48	–	0.52 x VDDIO	V
V_{outref}	SIO, Output voltage reference (regulated mode)	VDDIO > 3.3 V	1	–	VDDIO – 1	V
		VDDIO < 3.3 V	1	–	VDDIO – 0.5	V
R_{PULLUP}	Pull-up resistor	–	3.5	5.6	8.5	kΩ
$R_{PULLDOWN}$	Pull-down resistor	–	3.5	5.6	8.5	kΩ
I_{IL}	Input leakage current (absolute value)					
	GPIO	25 °C, VDDD = 3.0 V	–	–	2	nA
	CTBM GPIO pins	–	–	–	4	nA
	OVT GPIO	25 °C, VDDD = 0 V, VIH = 3.0 V	–	–	10	nA
	SIO	VIH ≤ VDDSIO; 25 °C	–	–	14	nA
		VIH > VDDSIO; 25 °C	–	–	10	nA
C_{IN}	Input Capacitance	–	–	–	7	pF
V_H	Input Hysteresis					
	CMOS input	Guaranteed by characterization	0.05 x VDDD	–	–	mV
	LVTTL input	VDDD ≥ 2.7 V. Guaranteed by characterization	25	40	–	mV
	SIO input	Single-ended mode	–	40	–	mV
		Differential mode	–	35	–	mV
I_{DIODE}	Current through protection diode to VDD/Vss	Guaranteed by characterization	–	–	100	uA
I_{TOT_GPIO}	Maximum Total Source or Sink Chip Current	Guaranteed by characterization	–	–	200	mA

PSoC 4 Pins AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
TriseF	Rise time in fast strong mode	Cload = 25 pF, V_{DDIO} = 3.3 V	2	–	12	ns
	OVT GPIO rise time in fast strong mode	25-pF load, 10%–90%, VDD=3.3 V	1.5	–	12	ns
TfallF	Fall time in fast strong mode	Cload = 25 pF, V_{DDIO} = 3.3 V	2	–	12	ns

Parameter	Description	Conditions	Min	Typ	Max	Units
	OVT GPIO fall time in fast strong mode	25-pF load, 10%–90%, VDD=3.3 V	1.5	–	12	ns
TriseS	Rise time in slow strong mode	Cload = 25 pF, Vddio = 3.3 V	10	–	60	ns
	OVT GPIO rise time in Slow-Strong mode	25-pF load, 10%–90%, VDD=3.3 V	10	–	60	ns
	SIO rise time in Slow-Strong mode	Cload = 25 pF, Vddio = 3.3 V	–	–	75	ns
TfallS	Fall time in slow strong mode	Cload = 25 pF, Vddio = 3.3 V	10	–	60	ns
	OVT GPIO fall time in Slow-Strong mode	25-pF load, 10%–90%, VDD=3.3 V	10	–	60	ns
	SIO fall time in Slow-Strong mode	Cload = 25 pF, Vddio = 3.3 V	–	–	70	ns
Fgpioout	GPIO output operating frequency					
	GPIO, 3.3 V < V_{DDIO} < 5.5 V, fast strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	33	MHz
	OVT GPIO, 3.3 V < V_{DDIO} < 5.5 V, fast strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	24	MHz
	GPIO, 1.7 V < V_{DDIO} < 3.3 V, fast strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	16.7	MHz
	OVT GPIO, 1.71 V < V_{DDIO} < 3.3 V, fast strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	16	MHz
	GPIO, 3.3 V < V_{DDIO} < 5.5 V, slow strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	7	MHz
	GPIO, 1.7 V < V_{DDIO} < 3.3 V, slow strong drive mode	90/10%, 25 pF load, 60/40 duty cycle	–	–	3.5	MHz
Fgpioin	GPIO input operating frequency, 1.71 V ≤ V_{DDD} ≤ 5.5 V	90/10% V_{IO}	–	–	48	MHz
Fsioout	SIO output operating frequency					
	3.3 V < V_{DDIO} < 5.5 V, Unregulated output, fast strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	33	MHz
	1.71 V < V_{DDIO} < 3.3 V, Unregulated output, fast strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	16	MHz
	3.3 V < V_{DDIO} < 5.5 V, Unregulated output, slow strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	5	MHz
	1.71 V < V_{DDIO} < 3.3 V, Unregulated output, slow strong drive mode	90/10% V_{DDIO} into 25 pF	–	–	3.5	MHz
	3.3 V < V_{DDIO} < 5.5 V, Regulated output, fast strong drive mode	Output continuously switching into 25 pF	–	–	20	MHz
	1.71 V < V_{DDIO} < 3.3 V, Regulated output, fast strong drive mode	Output continuously switching into 25 pF	–	–	10	MHz
	1.71 V < V_{DDIO} < 5.5 V, Regulated output, slow strong drive mode	Output continuously switching into 25 pF	–	–	2.5	MHz

Parameter	Description	Conditions	Min	Typ	Max	Units
Fsioin	SIO input operating frequency, 1.71 V < V_{DDIO} < 5.5 V	90/10% V_{IO}	–	–	48	MHz

PSoC 3 and PSoC 5LP 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
		LVTTL input, V_{DDIO} ≥2.7V	2.0	–	–	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
		LVTTL input, V_{DDIO} ≥2.7V	–	–	0.8	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
$R_{PULLDOWN}$	Pull-down resistor		3.5	5.6	8.5	kΩ
I_{IL}	Input leakage current (Absolute value) [1]					
	GPIO	25 °C, V_{DDIO} = 3.0 V	–	–	2	nA

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

Parameter	Description	Conditions	Min	Typ	Max	Units
	SIO: $V_{IH} \le V_{DDSIO}$	25 °C, V_{DDSIO} = 3.0 V, V_{IH} = 3.0 V	–	–	14	nA
	SIO: $V_{IH} > V_{DDSIO}$	25 °C, V_{DDSIO} = 0 V, V_{IH} = 3.0 V	–	–	10	µA
C_{IN}	Input Capacitance[1]					
	PSoC 3		–	–	7	pF
	PSoC 5LP		–	–	9	pF
V_H	Input voltage hysteresis (Schmitt-Trigger)[1]					
	PSoC 3	Single-ended mode (GPIO mode)	–	40	–	mV
		Differential mode	–	35	–	mV
	PSoC 5LP	Single-ended mode (GPIO mode)	–	115	–	mV
		Differential mode	–	50	–	mV
I_{DIODE}	Current through protection diode to V_{SSIO}		–	–	100	µA

PSoC 3 and PSoC 5LP 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
	1.71 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

Parameter	Description	Conditions	Min	Typ	Max	Units
Fsioin	SIO input operating frequency					
	1.71 V < V_{DDIO} < 5.5 V	90/10% V_{DDIO}	–	–	33	MHz

References

For more information on the advanced features of this Component, refer to following references.

- AN86439 (PSoC 4 Using GPIO PIns)
- AN72382 (Using PSoC 3 and PSoC 5LP GPIO pins)
- AN60580 (SIO Tips and Tricks in PSoC®3 / PSoC 5LP)

Component Errata

This section lists known problems with the Pins component.

Cypress ID	Component Version	Problem	Workaround
272088	v2.20	“CMOS 1.8V” input threshold option is shown for PSoC 4 SIO pins (P12.0, P12.1). Building the design does not show any warnings or errors.	“CMOS 1.8V” is not available for PSoC 4 SIO pins. They are meant for GPIO_OVT pins. Use other available options such as Vddio or VRef to set the desired input threshold level (e.g., to 1.8V).

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
2.20.d	Updated datasheet.	Added MISRA Compliance table, rule 10.1.
		Added errata item 272088 to document an issue with the input threshold option.
2.20.c	Updated PSoC 3/PSoC 5LP Fsioin from 66 MHz to 33 MHz	Outdated characterization data
	Updated v2.20 changelog with regards to the default sync update	Missed in datasheet revision
	Added “Display of Software Pins” sub-section	Usability information regarding software pins
2.20.b	Edited datasheet.	Added final characterization data for the PSoC 4000S device.

Version	Description of Changes	Reason for Changes / Impact
2.20.a	Edited datasheet.	Final characterization data for PSoC 4000S, PSoC 4100S and PSoC Analog Coprocessor devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.
2.20	Digital software pins no longer show wires on the schematic	Terminals and wires for SW pins are not needed on the schematic since nothing can be connected to them.
	Added SetInterruptMode() API	Allows the pin interrupt trigger mode to change during run-time.
	Added PSoC 4 SIO and USBIO support	PSoC 4200L support.
	Added I2C HS slew rate options	Supports slew rate definition to meet I2C HS specification.
	Added Sleep() and Wakeup() APIs for PSoC 4 devices	PSoC 4 SIO and USBIO pins must be configured in preparation for chip deep-sleep and hibernate modes. The functions allow restoration of pin configuration upon wakeup from chip deep-sleep and hibernate modes.
	Changed default sync settings to be transparent.	Many devices and ports do not support sync functionality at the pin. The default sync setting was changed to allow greater compatibility for migrating designs.
2.10.c	Added information for using interrupts on PSoC 4 ports that do not have PICU.	PSoC 4100M and PSoC 4200M devices introduce ports that do not have dedicated PICUs. To use interrupts on these ports, you must use the Global Signal Reference Component.
	Added information regarding drive mode restrictions for direct connections of peripherals to pins.	Some peripherals impose drive mode restrictions when directly connected to dedicated pins.
	Added References section.	References to Pins application notes.
2.10.b	Updated datasheet.	Updated change history to include v2.5.
2.10.a	Updated datasheet.	Added OVT GPIO characterization data.
2.10	Added GPIO_OVT support.	New pin type supported in BLE devices.
	Added PSoC 3 and PSoC 5LP per-pin APIs compatibility for PSoC 4 designs.	Allows PSoC 3 and PSoC 5LP per-pin APIs to be used in PSoC 4 designs.
	Merged contents of Type and General subtabs into General subtab.	Easier to observe Pin type affecting the default drive mode and initial drive state values.
	Initial state parameter text changed to "Initial drive state". When the pin is a PSoC 4 HW digital output, the parameter is fixed and is not configurable.	PSoC 4 HW digital output is driven by the signal connected to it and hence the initial drive state parameter should not be used.
2.5	Fixed a firmware bug that switched the POR settings for "Pulled-Up" and "Pulled-Down" modes; removed the Errata section.	To address a glitch with POR settings at power up.

Version	Description of Changes	Reason for Changes / Impact
2.0.c	Added Errata section to the datasheet.	To address a glitch with POR settings at power up.
2.0.b	Fixed a defect with SIO pairs on PSoC 3 and PSoC 5LP. If a Pins Component was used to configure a pair of SIOs, the settings for the first pin would be silently applied to the second. As a result, it was not possible to set up SIO pairs where the parameters were not identical for both pins.	Version 2.0 of the Pins Component allows independent settings of parameters on the pins. Note, however, that certain parameters, such as the input threshold, are still required to match. Normal parameter value checking within the customizer will catch these errors and force you to make appropriate corrections.
	Updated the handling of SIO pin pairs. When migrating from an earlier version of the Pins Component, the pin pair can show as being unpaired.	If the unpairing occurs, delete the pair and add the pins back.
2.0.a	Datasheet edits.	Updated the screen capture of the Reset tab. Added a Functional Description section.
2.0	Added support for PSoC 4000 (CY8C40xx) devices.	PSoC 4000 device pins have restrictions on routing and synchronization features.
	Added documentation for PSoC 4 Per-Pin APIs.	PSoC 4 Per-Pin APIs differ from PSoC 3 and PSoC 5LP
	Added clock driving pin information for PSoC 4 devices.	This requires specific configuration
1.90.a	Clarified Drive Mode diagrams.	Clarification
	Minor datasheet edits.	
1.90	Added PSoC 4 Support.	PSoC 4 enables new pin clocking options.
1.80	Added MISRA Compliance section.	The Component does not have any specific deviations.
	Added note about interrupt type for isr terminal connection.	Clarification
	Added note about drive mode support on USBIOs.	Clarification
	Changed Input Synchronized check-box on Input page to Sync Mode drop-down.	Drop-down allows future modes to be added.
	Changed Output Synchronized check-box on Output page to Output Mode drop-down.	Drop-down allows future modes to be added.
1.70	Minor datasheet edits and updates	
1.60.a	Minor datasheet edits and updates	
1.60	Added External Terminal capability	Allows pins to connect to Off-Chip 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

Version	Description of Changes	Reason for Changes / Impact
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 (an Infineon company), 2015-2021. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical Components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical Component is any Component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.