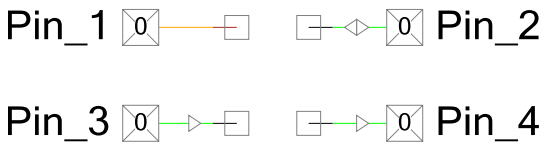
Pins
1.20

Features



- Any number of any type of pin
- Rapid setup of pin parameters
- Allows PSoC Creator to automatically place and route signals
- Allows interaction with 1 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 data via an appropriately configured physical IO pin. It 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 constrained within the component.

Pins can be used from schematics and/or software. To access a Pins component from component 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.

Note #defines are 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 and, for convenience, the Component Catalog provides four preconfigured Pins components: Analog, Digital Bidirectional, Digital Input, and Digital Output.

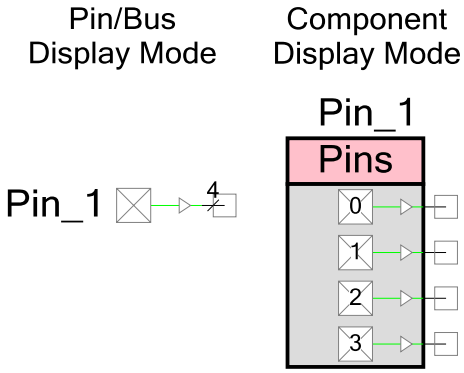
When to use a Pins Component

Use the Pins component when a design needs to generate or access an off-device signal through a physical IO pin. Pins are probably the most commonly used component in the Catalog. For example, in Cypress development and evaluation kits 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.

Note Pins can be configured into complex combinations of input, output, bidirectional, and analog. Simple configurations are generally shown as single pins or busses, but more complex types are shown as usual components.

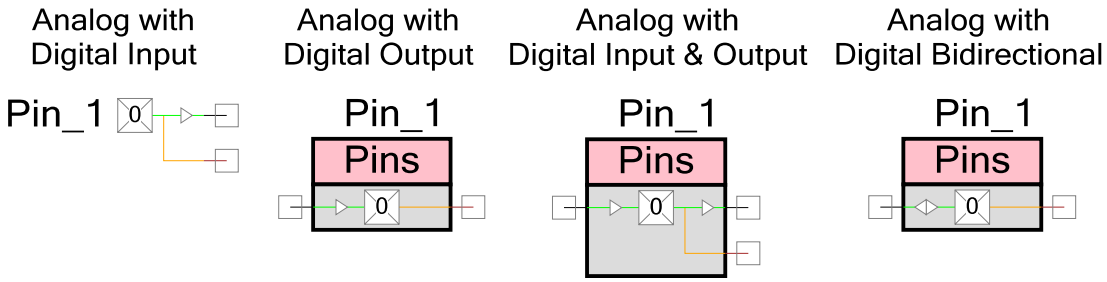


The default, and most common, configurations are shown in the following sections:

Analog

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.

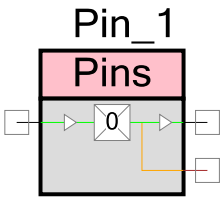


Digital Input

When configured as a digital input, the terminal can be shown for use in a schematic, or hidden for software access only. When visible, the terminal is shown on the right side of the symbol. The connection is drawn in the color of a digital wire and a small input buffer.

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

Digital Input with Output and Analog

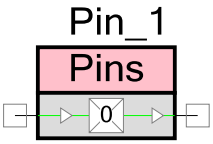


Digital Output

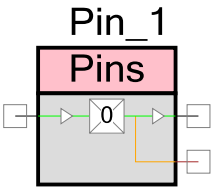
When configured as a digital output, the terminal can be shown for use in a schematic, or hidden for software access only. When visible, the terminal is shown on the left side of the symbol. The connection is drawn in the color of a digital wire and a small output buffer.

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

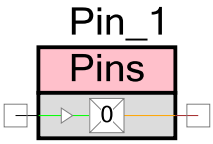
Digital Output with Input



Digital Output with Input and Analog



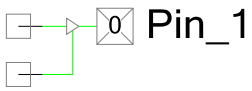
Digital Output with Analog



Digital Output Enable

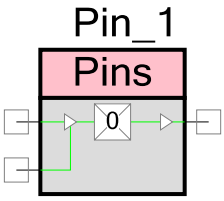
This terminal is shown when a component is configured with digital output using a schematic connection, and when the digital output enable has been selected. The digital output enable appears on the left side of the symbol into the digital output buffer. It is drawn in the color of a digital wire.

When set to Display as Bus, only one output enable is provided regardless of the Pins component width. When not displayed as a bus, individual output enables are provided per Output signal.



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

Digital Output Enable with Input

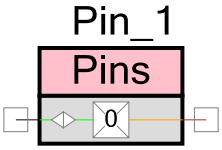


Digital Bidirectional

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.

A bidirectional Pins component may also support analog connections.

Digital Bidirectional with Analog



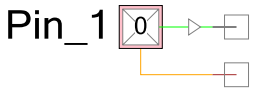
Vref

To configure a Pins component to use a Vref signal:

- use a Digital Input or Bidirectional terminal and configure 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. The Vref specifies the voltage supplied to the pin. All pins are capable of supplying their respective Vddio supply voltage. SIO pins are also able to supply a programmable or analog routed voltage for interface with devices at a different potential than the SIO’s Vddio voltage.

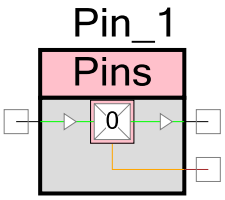
The Vref signal displays on the right side of the component, coming out of the bottom of the SIO single pin or the SIO pin pair depending on how it is configured.



Vref will be used with another digital connection.

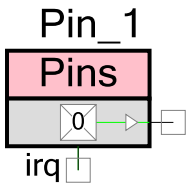
Note When using VRef, Analog cannot be used.

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 component displays in component mode, and the IRQ is displayed coming out of the bottom of the component. One typical use case is to hook an Interrupt component to this terminal.



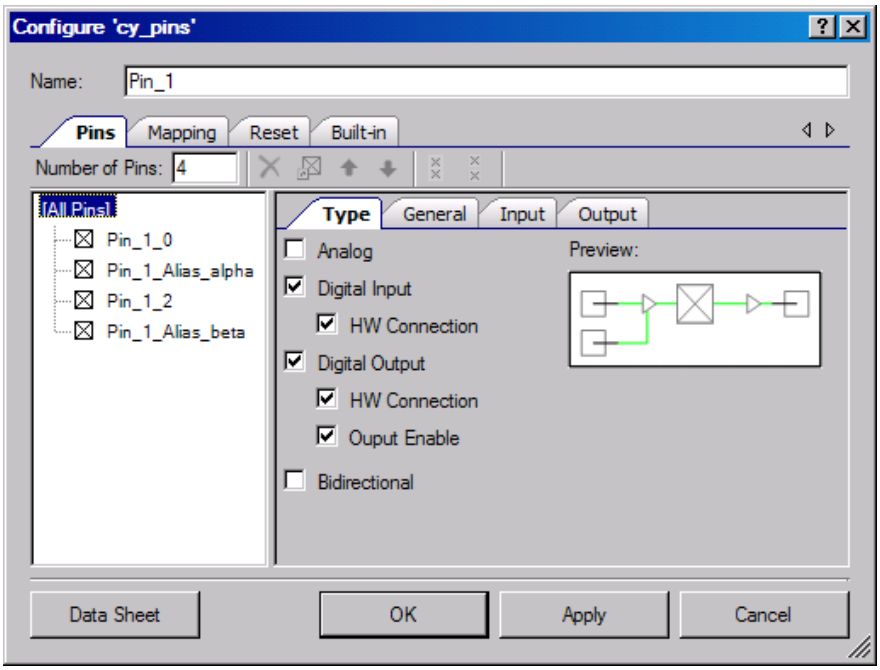
An Interrupt can be used in all types of the Pins component except analog, as long as you include Digital Input.

Component Parameters

Drag a Pin component onto your design 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 categorized into separate tabs.

Pins Tab

The **Pins** tab has three areas: a toolbar, pin tree, and another set of tabs (called 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 pin(s) 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. Default Value: 1.

  Note Some configurations can only be placed into a single physical port; therefore, the number of pins is limited to 8 or less. When the component is configured as noncontiguous and spanning, the number of pins can be set up to 64.

- **Delete Pin –** Deletes selected pin(s) 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 pin(s) up or down in the tree.
- **Pair / Unpair SIOs –** Pairs or unpairs selected SIO pins (denoted by a pink outline) in the tree.

  This control specifies whether or not 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** set to true, **Threshold Level** is set to anything but LVTTL or CMOS, **Drive Level** set to Vref, and/or **Drive Current** is set to a 25mA sink.

Pin Tree

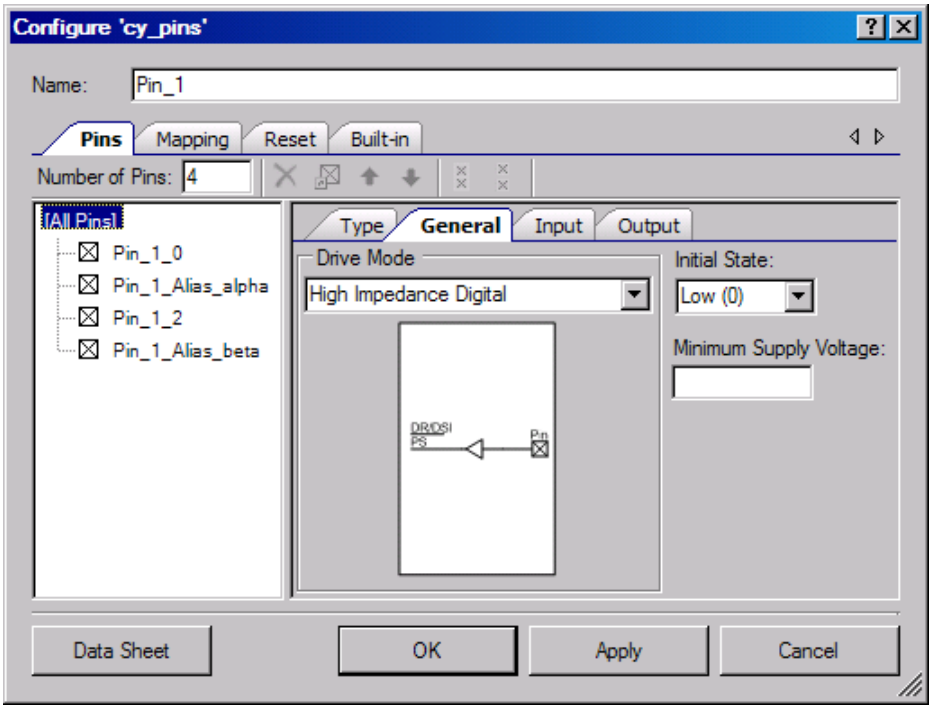
This area displays all of the pins for the component. You can individually select one or more pins to use with the toolbar commands and subtabs.

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 checkboxes. The preview area shows what the selected pin(s) type will look like with various options selected.

- Analog
- Digital Input
  - HW Connection – This parameter determines whether or not 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 may always be read by the CPU through registers or APIs. If this option is unchecked, the terminal is not displayed and it is controlled by software APIs.
- Digital Output
  - HW Connection – This parameter determines whether or not 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 unchecked, the terminal is not displayed and it is controlled by software APIs.
  - Output Enable
- Bidirectional

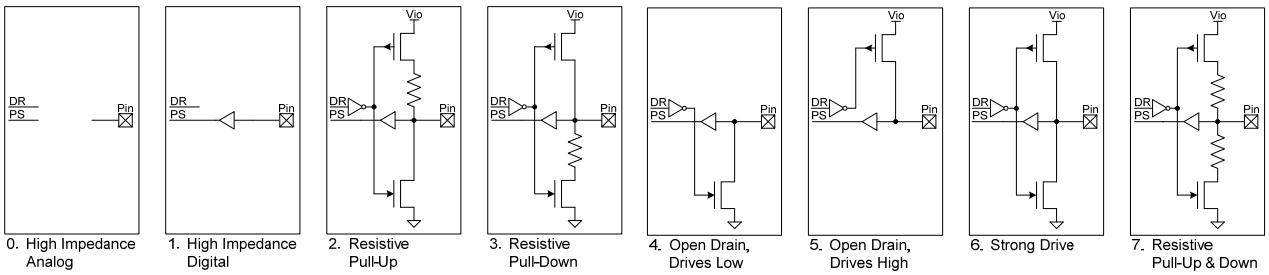
General Subtab



This subtab allows you to set up 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 data sheet for more details on each drive mode. A diagram shows the circuit representation for each drive mode as they are 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:



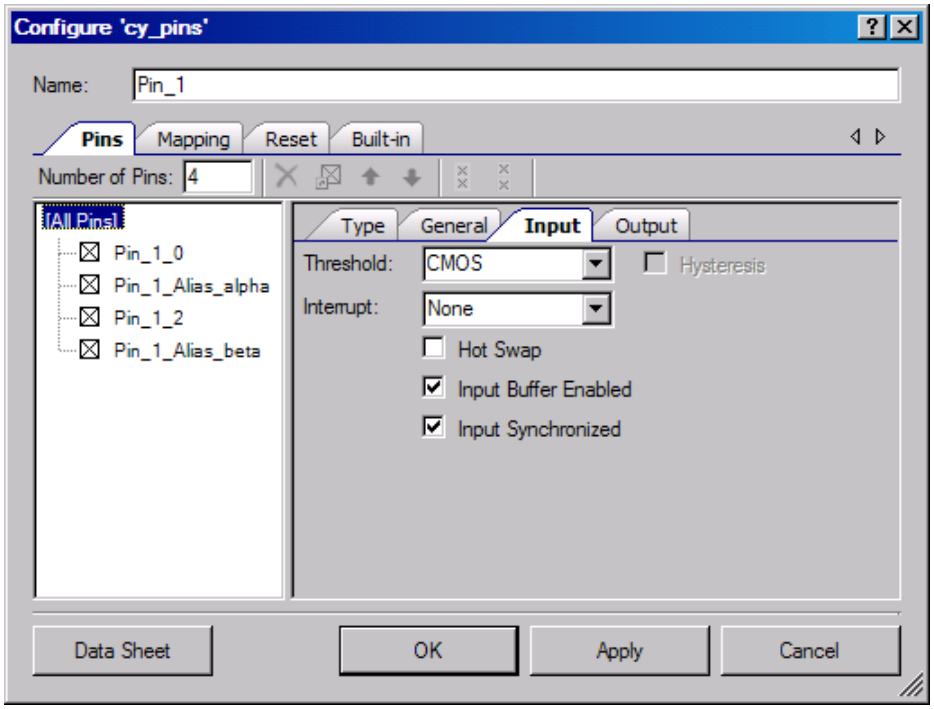
- **Initial State –** This parameter specifies the initial value written to the pin's Data Register after Power-On Reset (POR). All pins default to a logic low (0, False) in hardware at POR. The Initial State is written to the pin just after the Drive Mode is configured. The Initial State is configured high by default only for the "Resistive Pull Up" and "Resistive Pull Up/Down" drive-modes to ensure the pull up resister is active.

  Note This should not be confused with the reset state under the main **Reset** tab. That attribute affects the state of the pin from the moment of reset (and should not be modified often to avoid damaging the device) while this is the post-boot state of the pin, which is most-often the important state.

- **Minimum Supply Voltage –** This parameter selects the requested minimum high logic level output voltage. The requested voltage must be provided by one of the Vddio supply inputs. This selection ensures that the 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 Vddio voltages.

  Valid values are determined by the settings in the **System** tab of the *<project>.cydwr* file for Vio0/Vio1/Vio2/Vio3 Vio3 and to a lesser extent Vddd. Depending on the selected device, you may have a nominal number of USB pins that will use Vddd as their voltage available for placement. The pin will not be placeable 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 will appear in the Notice List window if the check fails.

Input Subtab

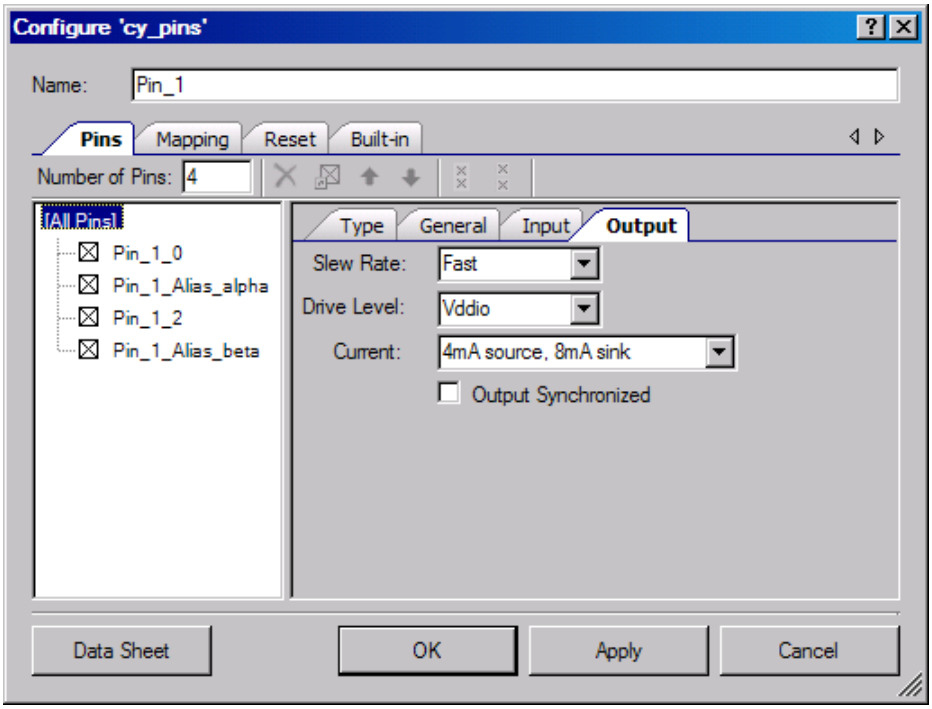


This subtab is used to specify input settings. If the pin type does not use Input or Bidirectional this tab is disabled as no input information needs to be specified.

- **Threshold –** This parameter selects the threshold levels that define a logic high level (1, True) and a logic low level (0, False). 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.
  - CMOS – Default
  - LVTTL
  - CMOS or LVTTL
  - 0.5 x Vddio – Requires SIO
  - 0.4 x Vddio – Requires SIO
  - 0.5 x Vref – Requires SIO
  - Vref – Requires SIO
- **Hysteresis –** Enables/Disables the SIO differential hysteresis for the pin. This feature is disabled if the Threshold is CMOS or LVTTL.
  - Disabled – Default
  - Enabled
- **Interrupt –** This parameter selects whether the pin is able to generate an interrupt and, if selected, the interrupt type. The pin interrupt may be generated with a rising edge, falling edge, as well as either edge or 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.
  - None - Default
  - Rising edge
  - Falling edge
  - Any edge
- **Hot Swap –** A pin configured to support hot swap capability will be mapped to an SIO pin supporting this capability in hardware. Hot Swap capability allows the voltage present on the pin to raise above the pins Vddio voltage, up to 6.0V. Hot Swap also allows a pin with any voltage up to 6.0V present to not leak current into the PSoC device even when the PSoC device is not powered.
  - No - Default
  - Yes – Requires SIO

- **Input Buffer Enabled –** This parameter determines if the pin's digital input buffer is enabled. The digital buffer is required to read or use the logic level present on a pin through DSI routing or a CPU read. The input buffer is required 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 default to Enabled.
  - Enabled
  - Disabled
- **Input Synchronized –** Input Synchronization occurs at pins to ensure all signals entering the device are synchronized throughout the system for proper device operation. Input synchronization may be optionally disabled at the pin in limited cases where an asynchronous signal is required for application performance and does not violate device operational requirements. Refer to the TRM or device data sheet for use details.
  - Yes - Default
  - No

Output Subtab

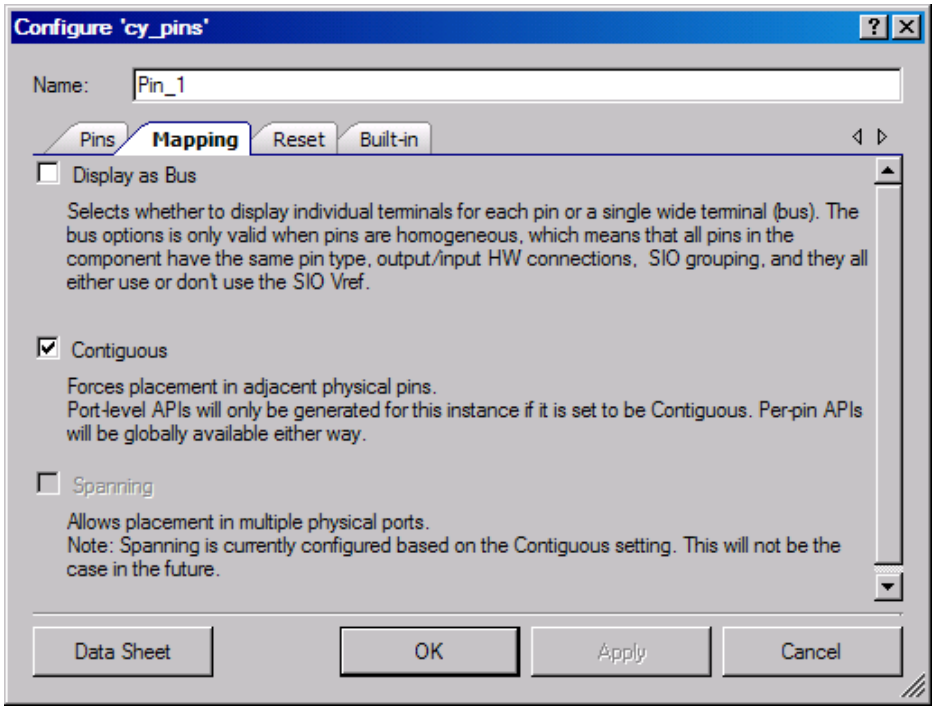


This tab is used to specify output settings. If the type is not Output or Bidirectional this tab is disabled because no output information needs to be specified.

- **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 1MHz. Slow mode may be selected for signals less than 1MHz switching rate and benefit from slower transition edge rates reducing radiated EMI and coupling with neighboring signals.
  - Fast – Default
  - Slow
- **Drive Level –** Selects the output drive voltage supplied sourced by the pin. All pins are capable of supplying their respective Vddio supply voltage. SIO pins are also able to supply a programmable or analog routed voltage for interface with devices at a different potential than the SIOs Vddio voltage.
  - Vddio – Default
  - Vref – Requires SIO
- **Drive Current –** The drive current selection determines the maximum nominal logic level current required for a specific pin. Pins may supply more current at the cost of logic level compliance or may have a maximum value that is less than listed, based on system voltages. Refer to the device data sheet for more details on drive currents.
  - 4mA source, 8mA sink – Default

  - 4mA source, 25mA sink – Requires SIO
- **Output Synchronized –** Output Synchronization can be enabled to reduce pin to pin output signal skew in high speed signals requiring minimal signal skew. Please see the TRM or device data sheet for use details.
  - Disabled - Default
  - Enabled

Mapping Tab



Display as Bus

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 all either use or don't use the SIO Vref.

Contiguous

Check box to force placement in adjacent physical pins. This option has the following restrictions:

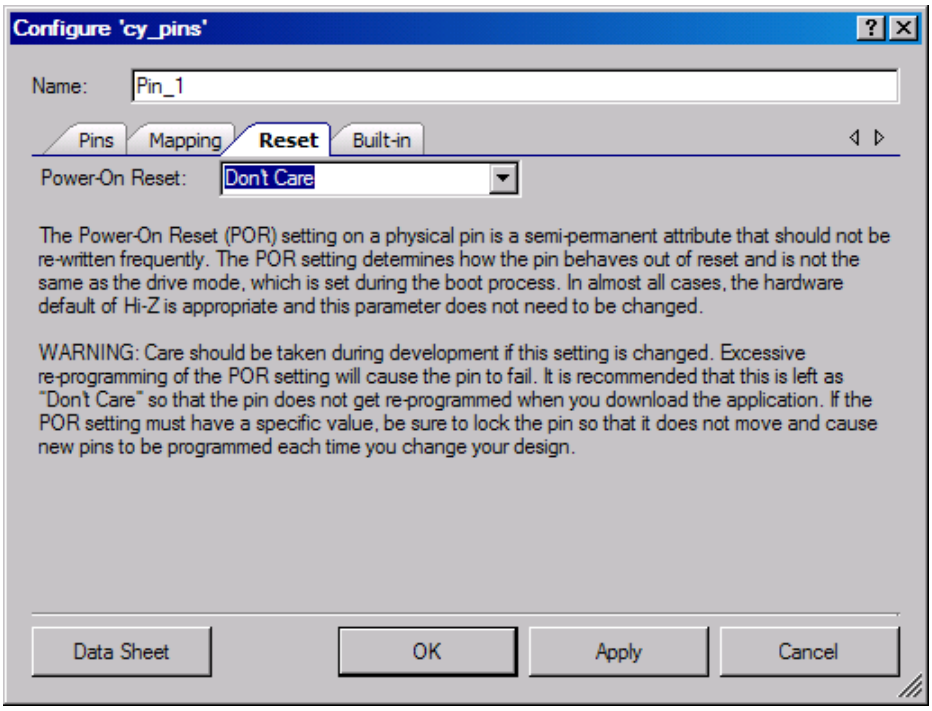
- If contiguous, analog cannot be used.
- If contiguous, port level APIs will be generated for the component. If non-contiguous port level APIs will not be generated.

- If contiguous, the number of pins in the component needs to be less than or equal to 8.

Spanning

Check box to allow placement in multiple physical ports. This is currently controlled by the contiguous selection. Until a future release of software offers greater flexibility and control, Contiguous implies non-spanning. Non-contiguous implies spanning.

Reset Tab



Power-On Reset

The Power-On Reset (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 and is not the same as the drive mode. In almost all cases, the hardware default of Don't Care is appropriate and this parameter does not need to be changed.

WARNING: Care should be taken during development if this setting is changed. Excessive re-programming of the POR setting will 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 will be determined by the physical port in which this component is placed. If all the placed pins in the port are set to Don't Care, the default

POR of the part will be used. Otherwise, whatever POR is specified for the other pins placed in that physical port (they must all match) will be used for the ones set to Don't Care.

- High-Z analog
- Pulled-up
- Pulled-down

Placement

There is no placement specific information.

Resources

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

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 ever 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 *<InstanceName>_aliases.h* file. By default the alias is the component name with the pin number appended to it:

<InstanceName>_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:

<InstanceName>_<AliaseName>

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. Non-contiguous 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.

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

Function	Description
uint8 Pin_1_Read(void)	Reads the physical port and returns the current value for all pins in the component
void Pin_1_Write(uint8 value)	Writes the value to the component pins while protecting other pins in the physical port.
uint8 Pin_1_ReadDataReg(void)	Read the current value of the port's data output register and returns the current value for all pins in the component.
void Pin_1_SetDriveMode(uint8 mode)	Sets the drive mode for each of the Pins component's pins.
uint8 Pin_1_ClearInterrupt(void)	Clears any active interrupts on the port into which the component is mapped. Returns value of interrupt status register.

uint8 Pin_1_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.

Parameters: None

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

Side Effects: None

void Pin_1_Write(uint8 value)

Description: Writes the value to the physical port (data output register), masking and shifting the bits appropriately. 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: Do to the use of read-modify write operations that are not atomic; it is possible for Interrupt Service Routines (ISR) to cause corruption of this API. An ISR that interrupts this API and performs writes to the Pins component Data register may cause corrupted port data. To avoid this issue it is recommended to either use the Per-Pin APIs (primary method) or disable interrupts around this API.

uint8 Pin_1_ReadDataReg(void)

Description: Reads the associated physical port's current data output register and masks the correct bits according to the width and bit position of the component instance. This is not the same as the preferred Read() API because it reads the data register instead of the status register. For output pins this is an occasionally 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_1_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: Do to the use of read-modify write operations that are not atomic it is possible for Interrupt Service Routines (ISR) to cause corruption of this API. An ISR that interrupts this API and performs writes to the Pins component Drive Mode registers may cause corrupted port data. To avoid this issue it is recommended to either use the Per-Pin APIs (primary method) or disable interrupts around this API.

uint8 Pin_1_ClearInterrupt(void)

Description:	Clears any active interrupts attached with the component and returns the value of the interrupt status register.
Parameters:	None
Return Value:	uint8: The current value of the interrupt status register.
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

The following is a C language example demonstrating the basic functionality of the Pins component. This example assumes the component has been placed in a design with the default name Pin_1.

Note If you rename your component you must also edit the example code as appropriate to match the component name you specify.

```
#include <device.h>

/* The following code uses both the per-pin and component          */
/* APIs to drive a pair of LEDs from a Pins instance.              */
/* The pins are aliased to RedLED (pin 0) and GreenLED (pin 1).    */

/* Set both pins to Strong drive */
Pin_1_SetDriveMode( Pin_1_DM_STRONG );

Pin_1_Write( 0 );    // Turn both LEDs off
Pin_1_Write( 1 );    // Turn pin 0 on and pin 1 off
Pin_1_Write( 2 );    // Turn pin 1 on and pin 0 off

/* Turn just GreenLED (pin 1) off, leaving RedLED alone  */
CyPins_ClearPin( Pin_1_GreenLED );

for(;;)
{
        /* Cause RedLED to flash      */
        if( CyPins_ReadPin( Pin_1_RedLED ) )
              CyPins_ClearPin( Pin_1_RedLED );
        else
              CyPins_SetPin( Pin_1_RedLED );

        delay();
}
```

DC and AC Electrical Characteristics

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

5.0V/3.3V DC and AC Electrical Characteristics

Parameter	Typical	Min	Max	Units	Conditions and Notes
Input					
Input Voltage Range	---		Vss to Vdd	V	
Input Capacitance	---		---	pF	
Input Impedance	---		---	Ω	
Maximum Clock Rate	---		67	MHz	

Component Changes

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

Version	Description of Changes
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.