Digital Port

1.0

Deprecation of Digital Port

The Digital Port component and its associated data sheet have been deprecated and replaced by the Pins component and data sheet. This was due to the creation of the Pins component to provide a more flexible solution for pins and ports. The Digital Port component remains in the Component Catalog to support legacy designs; however, it will be hidden by default for new designs, and it has been moved to a "deprecated" folder.

You should update your designs that use the Digital Port component to use the Pins component. To replace the component:

1. Based on the direction of your Digital Port, select the appropriate Pins component from the catalog (Digital Input, Digital Output, or Digital Bidirectional), from under the **Ports and Pins** section.
2. To have the terminals drawn as a bus, as they did on the Digital Port, go to the **Mapping** tab and check the "Display as Bus" option.

   Note When displayed as a bus only one OE will be used for all the output signals (which is consistent with how the Digital Port worked). If you do not check Display as Bus then a separate OE will be provided per output signal.
3. Configure the Pins component to match your Digital Port settings using the following conversion table.

Port Parameter	Pins Component Equivalent Setting
AccessMode – SW	The equivalent setting would be to make the pins component Contiguous from the **Mapping** tab (this will cause instance level APIs to be generated) and to uncheck HW Connection from the **Pins/Type** tab (this will remove the terminals so that no connection is required in the schematic).
AccessMode - HW	The equivalent setting would be to set CY_SUPPRESS_API_GEN to true on the **Built-in** tab (or make the component non-contiguous from the **Mapping** tab) so that instance level APIs will not be generated and to make sure HW Connection from the **Pins/Type** tab is checked so that terminals will be displayed on the schematic
Contiguous	This can be set from the **Mapping** tab by the Contiguous check box.
Direction	Select the pin/s you want to change the direction of in the tree on the left side of the **Pins** tab. From the Type subtab you can use the check boxes to set the "direction" of the pins. **Note** OE is now optional. If your direction was Output before you will need to check the Digital Output option along with the OE option to get what you had with the Digital Port.

DEPRECATED

Port Parameter	Pins Component Equivalent Setting

PowerOnResetState	This can be set from the **Reset** tab.
	InDisabledOutHiZ = High-Z Analog
	InEnabledOut1 = Pulled Up
	InEnabledOut0 = Pulled Down
	InEnabledOutHiZ = No longer supported

StandardLogic	This can be set from the **Pins/Input** tab (as long as the pin type has Input or Bidirectional used). It is now called Threshold.
UseInterrupts	This is no longer used. Just set the interrupt mode for the pin directly.
Width	From the **Pins** tab there is a Num Pins text box on the toolbar in the upper left of the tab.
Alias	From the **Pins** tab select a pin from the tree on the left side of the tab then either click the **Rename** button, press [**F2**], or double-click the pin in the tree. This will open a dialog where the alias can then be specified.

Pin Mode	This is now set from the **Pins/General** tab from the Drive Mode drop down list.
	CMOS_Out = Strong Drive
	Hi_Z = High Impedance Digital
	ResPull_Up = Resistive Pull Up
	ResPull_Down = Resistive Pull Down
	ResPull_UpDown = Resistive Pull Up/Down
	OpenDrain_Lo = Open Drain, Drives Low
	OpenDrain_Hi = Open Drain Drives High

Slew Rate	This can be set from the **Pins/Output** tab (as long as the pin type has Output or Bidirectional used).
Interrupt Mode	This can be set from the **Pins/Input** tab (as long as the pin type has Input or Bidirectional used).

4. If you were using the APIs provided when in SW_ONLY mode they have been altered as follows in the new Pins component:

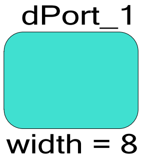
Port API	Change in Pins
[InstanceName]_WriteDM(uint8 mode, uint8 mask)	Replaced by: [InstanceName]_SetDriveMode(uint8 mode) It no longer allows a mask to be passed in. It will always affect all the pins in the component. If you were using the mask to set only some of the pins, you will now need to use the per-pin APIs provided in cy_boot/cypins.h instead. CyPins_SetPinDriveMode(pinPC, mode) For pinPC pass in the #define (in the form of [InstanceName]__[Index] or [InstanceName]__[Alias]) of the pin you want to set the drive mode of.

DEPRECATED

Port API	Change in Pins
#define [InstanceName]_BIT_SET #define [InstanceName]_BIT_CLEAR #define [InstanceName]_MODE_MASK #define [InstanceName]_MODE_BIT_0 #define [InstanceName]_MODE_BIT_1 #define [InstanceName]_MODE_BIT_2 #define [InstanceName]_PC_DM_MASK #define [InstanceName]_PC_DM_SHIFT	Removed. These all pertained to passing in a mask to [InstanceName]_WriteDM which has been removed.
[InstanceName]_ReadDR(void)	Renamed to [InstanceName]_ReadDataReg(void)
[InstanceName]_GetLastInterrupt(void)	RemovedThis feature is no longer supported.
#define [InstanceName]_WriteDR(prtValue)	This macro has been removed. Just call [InstanceName]_Write for equivalent behavior.
#define [InstanceName]_ReadPS()	This macro has been removed. Just call [InstanceName]_Read for equivalent behavior.
#define [InstanceName]_[Index]_ReadDM()	These macros have been removed from the component APIs. The per-pin APIs in cy_boot_cypins.h can be used to get this information. #define CyPins_ReadPinDriveMode(pinPC) For pinPC pass in the #define (in the form of [InstanceName]__[Index] or [InstanceName]__[Alias]) of the pin you want to get the drive mode of.
#define [InstanceName]_HI_Z #define [InstanceName]__RES_PULL_UP #define [InstanceName]_RES_PULL_DOWN #define [InstanceName]_OPEN_DRAIN_LO #define [InstanceName]_OPEN_DRAIN_HI #define [InstanceName]_CMOS_OUT #define [InstanceName]_RES_PULL_UPDOWN	Renamed to: #define [InstanceName]_DM_DIG_HIZ #define [InstanceName]_DM_RES_UP #define [InstanceName]_DM_RES_DWN #define [InstanceName]_DM_OD_LO #define [InstanceName]_DM_OD_HI #define [InstanceName]_DM_STRONG #define [InstanceName]_DM_RES_UPDWN

5. Delete your Digital Port and move your new Pins component to its old location.
6. Right-click on your project in the Workspace Explorer and select **Update Components**; use the Component Update Tool to update the latest version of the cy_boot component.

Features



- Variable widths
- Configurable interrupts
- Configurable drive modes

General Description

A digital port provides access to external data via an appropriately configured IO. All ports allow for the creation of per-pin aliases which may be viewed in the PSoC Creator Pin Editor and used in the generated port APIs.

When to use a Port

Use a port when a design needs to generate or access an off-device signal. Use an appropriate port for the type of signal being accessed (digital or analog).

Input/Output Connections

This section describes the various input and output connections for the port components. An asterisk (*) in the list of I/O's states that the I/O may be hidden on the symbol under the conditions listed in the description of that I/O.

i – Input *

Provides access to the digital input signal. This connection is only visible if all of the following criteria are met:

- AccessMode is PortAccessMode_HW
- Direction is PortDirection_Input or PortDirection_InOut

o – Output *

Enables PSoC to drive a digital signal off the device. This connection is only visible if all of the following criteria are met:

- AccessMode is PortAccessMode_HW
- Direction is PortDirection_Output or PortDirection_InOut

oe – Input *

Output enable determines whether the signal connected to the "o" terminal is actually driven off the device. This connection is only visible if all of the following criteria are met:

- AccessMode is PortAccessMode_HW
- Direction is PortDirection_Output or PortDirection_InOut

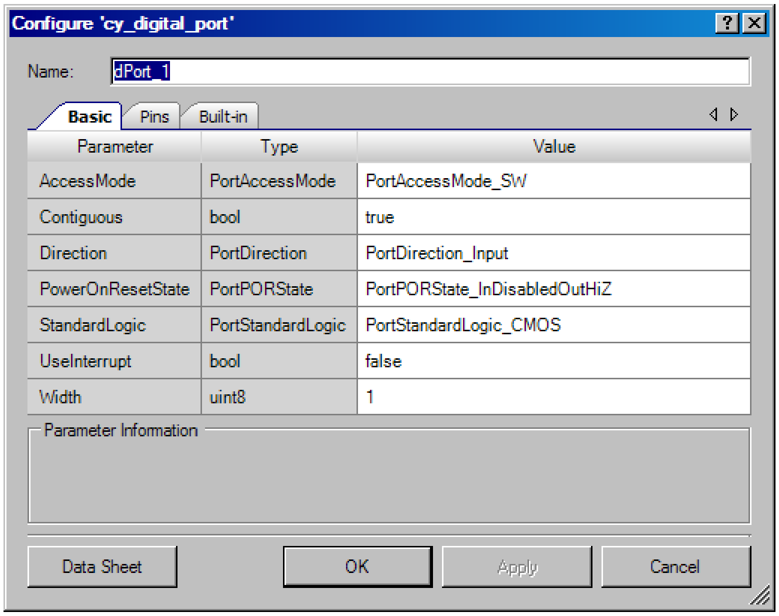
irq – Output *

The signal driven out of this connection is high when the conditions under which the port should generate an interrupt have been met. Connect this to an Interrupt component to define the interrupt handler.

Component Parameters

Drag a Digital Port onto your design and double-click it to open the Configure dialog.

Basic Tab



Configure 'cy_digital_port'

Name: dPort_1

Parameter	Type	Value
AccessMode	PortAccessMode	PortAccessMode_SW
Contiguous	bool	true
Direction	PortDirection	PortDirection_Input
PowerOnResetState	PortPORState	PortPORState_InDisabledOutHiZ
StandardLogic	PortStandardLogic	PortStandardLogic_CMOS
UseInterrupt	bool	false
Width	uint8	1

AccessMode

AccessMode configures the access method of the port.

AccessMode Value	Description
PortAccessMode_SW	The port may only be accessed via firmware (default).
PortAccessMode_HW	The port is accessed via hardware. Hardware only ports do not have APIs.

Contiguous

If true (default), all pins in the port are assigned to consecutive pins in a physical port. If false, the logical pins may be assigned to different physical ports. In software access mode (PortAccessMode_SW), the logical port is always contiguous within a physical port.

Direction

Enables signal flow in a given direction.

Direction Value	Description
PortDirection_Input	Enables a design to receive signals into the device (default).
PortDirection_InOut	Enables a design to drive signals off the device or receive signals into the device.
PortDirection_Output	Enables a design to drive signals off the device.
PortDirection_Bidirectional	Enables signal flow in both direction on a given pin.

PowerOnResetState

Specifies the power on reset (POR) state of the port. The POR setting places a value in the PRT* data register as follows:

POR Value	Description	{PRT*.DR}
InDisabledOutHiZ	Input disabled and the output is Hi-Z (default)	0
InEnabledOut0	Input enabled and the output is a logic zero	0
InEnabledOut1	Input enabled and the output is a logic one	1
InEnabledOutHiZ	Input enabled and the output is Hi-Z	0

StandardLogic

Specifies the voltage level at which a device changes state:

- **PortStandardLogic_CMOS** – CMOS logic levels (default)
- PortStandardLogic_LVTTL – TTL logic levels

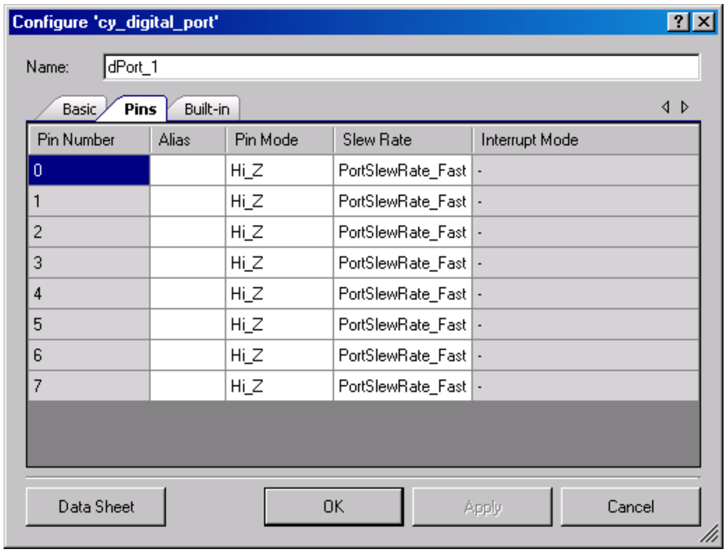
UseInterrupt

If true, the port may generate an interrupt. The “irq” terminal will become visible, and must be connected to an Interrupt component. The conditions under which an interrupt will be generated is specified on the “Pins” tab. If **false** (default), the port will not generate an interrupt.

Width

Specifies the width in bits of the logical port (default is 1).

Pins Tab



Name: dPort_1

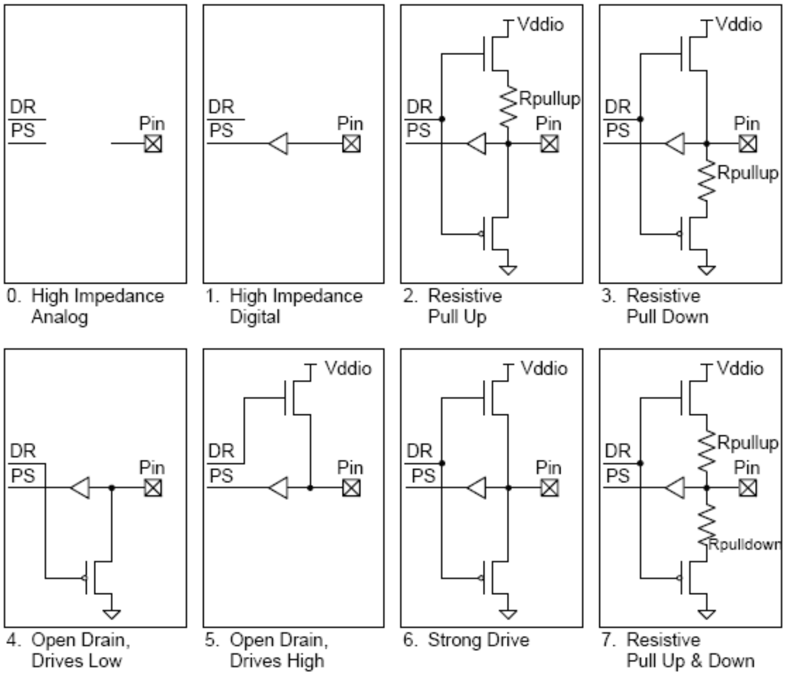
Pin Number	Alias	Pin Mode	Slew Rate	Interrupt Mode
0		Hi_Z	PortSlewRate_Fast	-
1		Hi_Z	PortSlewRate_Fast	-
2		Hi_Z	PortSlewRate_Fast	-
3		Hi_Z	PortSlewRate_Fast	-
4		Hi_Z	PortSlewRate_Fast	-
5		Hi_Z	PortSlewRate_Fast	-
6		Hi_Z	PortSlewRate_Fast	-
7		Hi_Z	PortSlewRate_Fast	-

Alias

Allows an alias (in conjunction with the actual port pin assignment) to be assigned to each pin in the port. The alias is presented in the Pin Editor and in the generated APIs for the port.

Pin Mode (Drive Mode)

This parameter allows you to configure the pin mode. Each pin is individually configurable into one of eight modes, shown in the following figure.



0. High Impedance Analog
1. High Impedance Digital
2. Resistive Pull Up
3. Resistive Pull Down
4. Open Drain, Drives Low
5. Open Drain, Drives High
6. Strong Drive
7. Resistive Pull Up & Down

DEPRECATED

The following table shows the IO pin’s drive state based on the port data register value or digital array signal if bypass mode is selected. Note that the actual IO pin voltage is determined by a combination of the selected drive mode and the load at the pin.

Drive Mode	PRTxDR = 1	PRTxDR = 0
High Impedance	High-Z	High-Z
Resistive pull up	Res High (5K)	Strong Low
Resistive pull down	Strong High	Res Low (5K)
Open drain, drives low	High-Z	Strong Low
Open drain, drive high	Strong High	High-Z
Strong drive	Strong High	Strong Low
Resistive pull up and pull down	Res High (5K)	Res Low (5K)

Three configuration bits are used for each pin (DM[2:0]) and set in the PRTxDM[2:0] registers.

- **High Impedance –** The input buffer is enabled for digital signal input. This is the standard high impedance (Hi-Z) state recommended for digital inputs.
- **Resistive Pull Up or Resistive Pull Down –** Resistive pull up or pull down, respectively, provides a series resistance in one of the data states and strong drive in the other. Pins can be used for digital input and output in these modes. Interfacing to mechanical switches is a common application for these modes.
- **Open Drain, Drives High and Open Drain, Drives Low –** Open drain modes provide high impedance in one of the data states and strong drive in the other. Pins can be used for digital input and output in these modes. A common application for these modes is driving the I2C bus signal lines.
- **Strong Drive –** Provides a strong CMOS output drive in either high or low state. This is the standard output mode for pins. Strong Drive mode pins must not be used as inputs under normal circumstances. This mode is often used to drive digital output signals or external FETs.
- **Resistive Pull Up and Pull Down –** Similar to the resistive pull up and resistive pull down modes except the pin is always in series with a resistor. The high data state is pull up while the low data state is pull down. This mode is most often used when other signals that may cause shorts can drive the bus.

Slew Rate

The slew rate of a device is the rate of change of its output. The available rates are:

- **Fast** (default)
- Slow

Interrupt Mode

Indicates the conditions under which the pin will trigger the port interrupt. The interrupt mode can only be set if **UseInterrupt** is true. The available conditions are:

- **None** (default)
- Rising Edge
- Falling Edge
- On Change

Resources

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

Application Programming Interface

Application Programming Interface (API) routines allow you to control the port component using software. The following table lists and describes the interface to each function. The subsequent sections cover each function in more detail.

By default, PSoC Creator assigns the instance name "dPort_1" to the first instance of a 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 "dPort_1".

APIs are only generated if the method of access to the logical port is software. Also the generated APIs are a function of the logical port direction, i.e., read, write or both.

Function	Description
uint8 dPort_1_Read(void)	Reads the associated physical port and mask the correct bits according to the width and bit position of the defined logical port. The result is right justified into a contiguous group of bits.
void dPort_1_Write(uint8 prtValue)	Write to the associated physical port and mask the correct bits according to the width and bit position within the physical port. Avoid changing other bits in the port by using the appropriate method (read-modify-write or bit banding).

DEPRECATED

Function	Description
uint8 dPort_1_ReadDR(void)	Read the current value of the port's data output register.
uint8 dPort_1_X_ReadDM(void)	Reads the current drive mode of Pin X, where X is between 0 and port width – 1.
void dPort_1_WriteDM(uint8 mode, uint8 mask)	Assign the drive mode, "mode", to each pin of the port which is masked by "mask".
uint8 dPort_1_ClearInterrupt(void)	Clears any active interrupts attached to port. Returns value of interrupt status register.
uint8 dPort_1_GetLastInterrupt(void)	Returns the last read value of the interrupt status register from the snap shot register.

uint8 dPort_1_Read (void)

Description: Reads the associated physical port's pin states and masks the correct bits according to the width and bit position of the defined logical port. The result is right justified into a contiguous group of bits.

Parameters: None

Return Value: Right justified group of bits equal to the width of the logical port.

Side Effects: None

void dPort_1_Write(uint8 prtValue)

Description: Write to the associated physical port's data output register. This function will mask and shift the bits appropriately for the underlying physical port. Avoid changing other bits in the port by using the appropriate method (read-modify-write or bit banding).

Parameters: uint8 prtValue: Value to write to the data output register of the physical port.

Return Value: None

Side Effects: None

uint8 dPort_1_ReadDR(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 defined logical port.

Parameters: Void

Return Value: Right justified group of bits defining the digital port's current data output register value.

Side Effects: None

uint8 dPort_1_ClearInterrupt(void)

Description: Clears any active interrupts attached to port 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** the bits of the port's interrupt status register. Not just those associated with the digital port.

uint8 dPort_1_GetLastInterrupt(void)

Description: Gets the last read value of the interrupt status register by reading the snapshot register.

Parameters: None

Return Value: uint8: The last read value of the interrupt status register.

Side Effects: None

uint8 dPort_1_X_ReadDM(void)

Description: Reads the current drive mode of pin "X" where X is a number representing pin 0 to (Width -1).

Parameters: None

Return Value: uint8: The numerical value representing that pin's current drive mode.

Side Effects: None

void dPort_1_WriteDM(uint8 mode, uint8 mask)

Description: Write the value of "mode" to be the drive mode for each of the digital port's pins specified by "mask"

Parameters: uint8 mode: mode for the selected signals
uint8 mask: mask for the selected signals

Return Value: None

Side Effects: None

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	

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