


# Clock

## 1.0

## Features

Clock\_1  24 MHz

- Quickly define new clocks
- Refer to system or design-wide clocks
- Configure the tolerance

## General Description

The Clock component provides two key features: it provides the means to create local clocks, and it provides the means to connect designs to system and design-wide clocks. All clocks are shown in the Design-Wide Resources (DWR) Clock Editor. For more information, refer to the PSoC Creator Help, Clock Editor section.

Clocks may be defined in a variety of ways, for example:

- as a frequency with an automatically selected source clock
- as a frequency with a user-selected source clock
- as a divider and user-selected source clock.

If a desired frequency is specified, PSoC Creator will automatically select a divider that yields the most accurate resulting frequency. If allowed, PSoC Creator will also examine all system and design-wide clocks and select a source and divider pair that yields the most accurate resulting frequency.

## When to use a Clock

The Clock component should be used whenever a design requires access to a system clock, design-wide clock, or a specific clock frequency.

## Appearance

The color of the Clock component waveform symbol will change based on the clock's Domain (as shown in the DWR Clock Editor), as follows:

- Digital – the waveform color is the same as a digital wire, with a black outline.
- Analog – the waveform color is the same as an analog wire, with a black outline.
- Indeterminate – the waveform color is white, with no outline.

**PRELIMINARY**

## Input/Output Connections

Clocks have a single output terminal which provides access to the resulting clock signal.

## Component Parameters

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

### Configure Clock Tab

The **Configure Clock** tab contains the **Clock Type** and **Source** parameters. Based on your selections, this tab will contain various other parameters as shown in the following figures:

Figure 1 Clock Type: New / Source: <Auto>

The screenshot shows the 'Configure 'cy\_clock'' dialog box. The 'Name' field contains 'Clock\_1'. The 'Configure Clock' tab is active, showing 'Clock Type' as 'New' and 'Source' as '<Auto>'. Under 'Specify', 'Frequency' is set to 24 MHz and 'Tolerance' is set to 5%. The 'Summary' section shows 'API Generated: Yes' and 'Uses Clock Tree Resource: Yes'. The 'Data Sheet', 'OK', 'Apply', and 'Cancel' buttons are at the bottom.

PRELIMINARY



**Figure 2 Clock Type: New / Source: Specific Clock**

The screenshot shows the 'Configure 'cy\_clock'' dialog box with the 'Configure Clock' tab selected. The 'Name' field is 'Clock\_1'. Under 'Clock Type', 'New' is selected. The 'Source' dropdown is 'MASTER\_CLK (24.000 MHz)'. Under 'Specify', 'Frequency' is selected with a value of '24' and unit 'MHz'. The 'Summary' section shows 'API Generated: Yes' and 'Uses Clock Tree Resource: Yes'. The 'Source Clock Info' section shows 'Name: MASTER\_CLK', 'Enabled: Yes', 'Frequency: 24.000 MHz', 'Accuracy: ±1', and 'Divider: 1'. Buttons at the bottom include 'Data Sheet', 'OK', 'Apply', and 'Cancel'.

**Figure 3 Clock Type: Existing**




The screenshot shows the 'Configure 'cy\_clock'' dialog box with the 'Configure Clock' tab selected. The 'Name' field is 'Clock\_1'. Under 'Clock Type', 'Existing' is selected. The 'Source' dropdown is 'BUS\_CLK (24.000 MHz)'. The 'Summary' section shows 'API Generated: No' and 'Uses Clock Tree Resource: No'. The 'Source Clock Info' section shows 'Name: BUS\_CLK', 'Enabled: Yes', 'Frequency: 24.000 MHz', 'Accuracy: ±1', and 'Divider: 1'. Buttons at the bottom include 'Data Sheet', 'OK', 'Apply', and 'Cancel'.

The following sections describe the Clock component parameters:

## Clock Type

There are two clock types: New and Existing. For new clocks, you can specify a clock **Source** to use or allow PSoC Creator to choose by selecting <Auto>. If you select <Auto>, you can also enter a specific **Frequency** and optional **Tolerance**. If you specify a Source, you can either specify a **Frequency** or choose a **Divider**. For existing clocks, you can only select the clock **Source**.

For different configurations, the clock symbol displays differently on the schematic, as shown in the following examples.

New/Desired Frequency	New/Divider	Existing
Clock_1  24 MHz	Clock_2  ILO / 1	BUS_CLK 

Clock components configured as "New" consume resources in the device and have APIs generated for them. Clock components configured as "Existing" to a system or design-wide clock do not consume any physical resources on the device and no APIs are generated for them. Instead, they act as an alias or proxy for the selected system or design-wide clock.

## Source

Select <Auto> (default) if PSoC Creator should automatically locate an available source clock that, when divided down, provides the most accurate resulting frequency. Clocks with a source of <Auto> may only enter a desired frequency. A tolerance may also optionally be provided.

Select a system or design-wide clock from the list provided to force PSoC Creator to use that clock as the source.

## Frequency

Enter the desired frequency and units (default = 24 MHz). PSoC Creator will then calculate the divider that will create a clock signal that is as close as possible to the desired frequency.

## Tolerance

If <Auto> is selected as the clock source, you can enter the desired tolerance values for the clock (default is +/- 5%). PSoC Creator will ensure that the accuracy of the resulting clock falls within the given tolerance range or produce a DRC warning if the desired clock is not achievable. Clock tolerances are specified as a percentage. (**Note** Entering ppm will cause the value entered to be converted to the corresponding percent value.) If there is no desired tolerance range, then the check box next to the tolerance can be unchecked and no warning will be generated for this clock.

PRELIMINARY



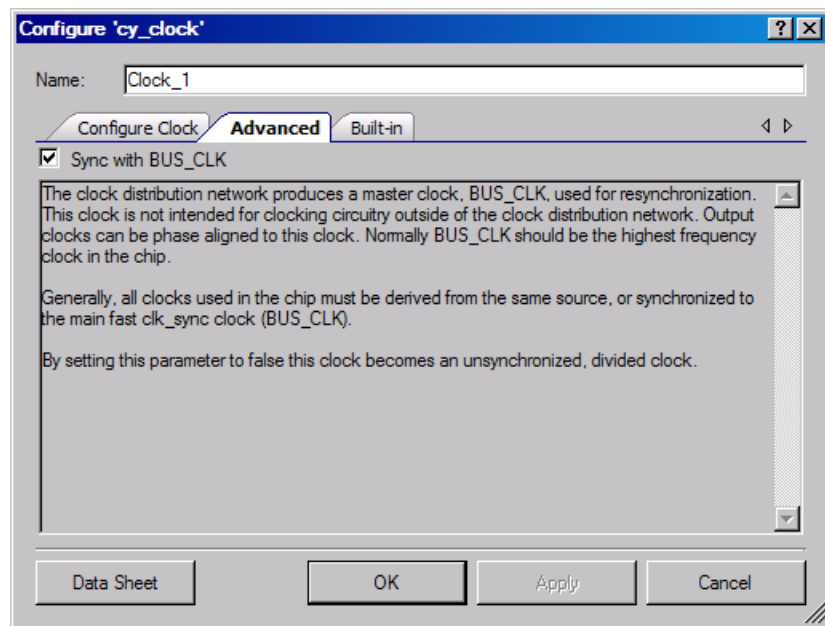
## Divider

If you choose a specific **Source Clock**, you can enter an explicit value for the **Divider**. Otherwise, if you leave the **Source Clock** set to <Auto>, the **Divider** option is not available (default).

If you do select the **Divider** option, then the **Frequency** option is not available.

## Advanced Tab

The Advanced tab contains only one parameter.



## Sync With BUS\_CLK

If checked (default = checked) the clock is synchronized with the BUS clock; otherwise, the clock is unsynchronized.

## Placement

Clock components configured as "Existing" do not consume any resource on the chip.

Clock components configured as "New" consume a single clock block. PSoC Creator automatically discovers whether the clock connects to digital or analog peripherals and consumes a digital clock block or analog clock block as necessary.

## Resources

Resource usage varies based on configuration and connectivity. See the Placement section for details.



**PRELIMINARY**

## Application Programming Interface

Application Programming Interface (API) routines allow you to configure the 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 "Clock\_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 "Clock".

Function	Description
Clock_Start	Enables the clock.
Clock_Stop	Disables the clock.
Clock_StandbyPower	Selects the power for standby operation modes.
Clock_SetDivider	Sets the divider of the clock.
Clock_SetMode	Sets the mode of the clock.
Clock_SetSource	Sets the source of the clock.
Clock_SetPhase	Sets the phase of the analog clock. This only applies to analog clocks.

**Note** Local clocks configured as "Existing" will not have any APIs generated.

### void Clock\_Start(void)

**Description:** Enables the clock by setting the enable bit in the 'Active Power Mode Configuration Register' corresponding to this clock.

**Parameters:** void.

**Return Value:** void.

**Side Effects:** The clock is enabled.

PRELIMINARY



## void Clock\_Stop(void)

- Description:** Disables the clock by clearing the enable bit in the 'Active Power Mode Configuration Register' corresponding to this clock.
- Parameters:** void.
- Return Value:** void.
- Side Effects:** The clock is disabled. If the clock is disabled, the output will be held at logic 0 when the signal transitions to a logic 0.

## void Clock\_StandbyPower(uint8 state)

- Description:** Selects the power state for this clock when in standby power mode.
- Parameters:** uint8 state: State of this clock during standby power mode. 1 is active, 0 inactive.
- Return Value:** void.
- Side Effects:** None

## void Clock\_SetDivider(uint16 clkDivider)

- Description:** Sets the divider of the clock.
- Parameters:** uint16 clkDivider: Value by which to divide the clock. This parameter is the divide value + 1.
- Return Value:** void.
- Side Effects:** Any value other than 0 is acceptable.



**PRELIMINARY**

## void Clock\_SetMode(uint8 clkMode)

**Description:** Sets the operating mode of the clock.

**Parameters:** uint8 clkMode: For PSoC 3 and PSoC 5 devices, clkMode should be a set of the following optional bits or'ed together:

- CYCLK\_PIPE: Reserved, should not be set.
- CYCLK\_SSS: Bypass divider and synchronization. Should be set when output frequency is the same as the synchronizer frequency.
- CYCLK\_EARLY: Enable early phase mode. Rising edge of output clock will occur when the divider counter reaches half of the divide value.
- CYCLK\_DUTY: Enable 50% duty cycle output. By default, the output clock is asserted for one period of the input clock.
- CYCLK\_SYNC: Enable output synchronization to master clock.

See the Technical Reference manual for details about setting the mode of the clock. Specifically about CFG2 register.

**Return Value:** void.

**Side Effects:** None

## void Clock\_SetSource(uint8 clkSource)

**Description:** Sets the input source of the clock. The clock must be disabled before changing the source of the clock.

**Parameters:** uint8 clkSource: For PSoC 3 and PSoC 5 devices, clkSource should be one of the following input sources:

- CYCLK\_SRC\_SEL\_SYNC\_DIG: Phase-delayed master clock.
- CYCLK\_SRC\_SEL\_IMO: Internal main oscillator.
- CYCLK\_SRC\_SEL\_XTALM: 4-33 MHz external crystal oscillator.
- CYCLK\_SRC\_SEL\_ILO: Internal low-speed oscillator.
- CYCLK\_SRC\_SEL\_PLL: Phase locked loop output.
- CYCLK\_SRC\_SEL\_XTALK: 32.768 kHz external crystal oscillator.
- CYCLK\_SRC\_SEL\_DSI\_G: dsi\_global[0].
- CYCLK\_SRC\_SEL\_DSI\_D: dsi\_d.

See the technical Reference Manual about mapping a clock source.

**Return Value:** void.

**Side Effects:** None

**PRELIMINARY**





## void Clock\_SetPhase(uint8 clkPhase)

**Description:** Sets the phase delay of the analog clock. This is an analog clock function only.

**Parameters:** uint8 clkPhase: Amount to delay the phase of the clock, in 1.0 ns increments up to 0x0B (12.5ns).

**Return Value:** void.

**Side Effects:** None

## Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the Clock. This example assumes the component has been placed in a design and is named "Clock\_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>

void main()
{
    /* Clock_1_Start(); */
    /* The DWR Clock Editor contains a "Start on Reset" option that enables */
    /* the clock if you always need it. In some cases, such as to reduce */
    /* power consumption, you may wish to control the clock programmatically.*/
    /* In such cases, you would de-select the DWR Clock Editor option, and */
    /* uncomment the above function. */
}
```



**PRELIMINARY**

## Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0a	Move CYCLK_ constants to cydevice.h/cydevice_trm.h.	The CYCLK_ constants for the mode and source are now generated from the selected device's register map. This allows the clock component to be independent of device-specific register values. The <i>cydevice.h</i> file is already included from the clock header, so no user code changes should be necessary.
	Add description of CYCLK_ constants in the data sheet.	The parameter descriptions for the SetMode and SetSource APIs now contain a description of each value.

© Cypress Semiconductor Corporation, 2009-2010. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

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

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

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

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

**PRELIMINARY**

