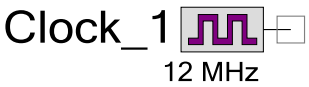
Please note that Cypress is an Infineon Technologies Company.
The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the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

Clock (SysClk_PDL)
1.0

Features



- Generates programmable clock dividers for use with other Components that require clocks
- Supports 8 bit, 16 bit, 16.5 bit, and 24.5 bit dividers
- Configure a clock by specifying its frequency or divider value
- Phase align with another programmable clock divider

General Description

The Clock (SysClk_PDL) Component provides an interface to the programmable peripheral clock dividers. A divider can be configured by specifying its frequency, or by specifying its divider value.

Most peripheral dividers provided by the Clock Component are sourced from clk_peri. Thus, a divider frequency is dependent on the frequency of clk_peri. For details, see the Functional Description section. To configure the frequency of clk_peri, use the Design-Wide Resources Clock Editor in your PSoC Creator project.

The Component also supports selection of clock sources not derived from clk_peri. These can be selected by specifying the clock source in the Configure dialog. These select clock sources and their connections are specific to the peripherals in specific devices.

When to Use a Clock

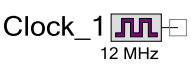
Use the Component when a clock input is needed for another Component such as a TCPWM, an SCB, or a UDB-based Component.

Input/Output Connections

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

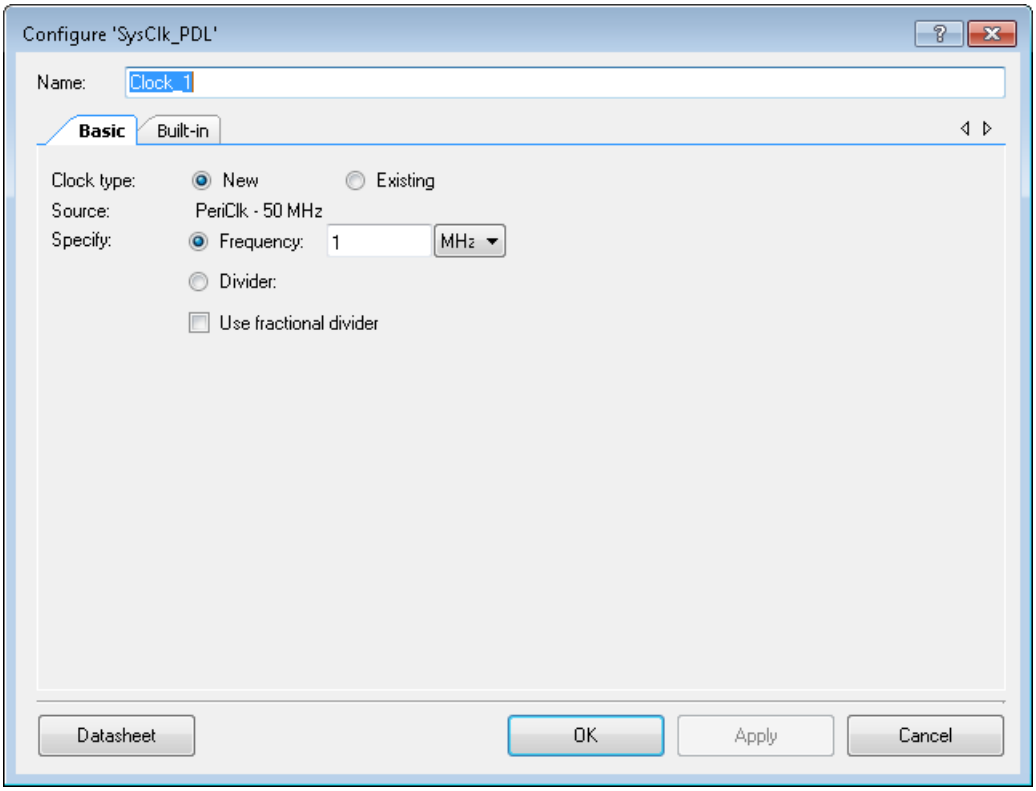
clock – output

The Clock Component has an output terminal that provides access to the clock signal.



Component Parameters

Drag a Clock Component onto your design and double click it to open the Configure dialog. This dialog has the following tabs with different parameters.



Clock type

There are two clock types: **New** and **Existing**. A new clock uses a programmable divider. An existing clock uses a clock that already exists in the system. Note that the option to select **Existing** may not be available, depending on the Component being clocked. When **Existing** is chosen, the **Source** drop-down shows all existing clocks. The list of available clocks is device dependent.

Source

The source is the peripheral clock, clk_peri. Its frequency setting is also displayed.

Specify:

- **Frequency:** You can specify your clock by frequency or by divider value. For frequency, select the **Frequency** option and enter the desired frequency value and units (Hz, kHz, MHz).

- **Divider:** To specify your clock by divider value, select the **Divider** option and enter the desired divider value.
  - **Use fractional divider:** If **Divider** is selected, you can optionally select the **Use fractional divider** box and specify a fractional divider value, in units of 1/32, from 0 to 31. For more information about fractional division, refer to the device *Technical Reference Manual*.

Application Programming Interface

Application Programming Interface (API) is provided by the sysclk driver module from the Peripheral Driver Library (PDL). The driver is copied into the "pdl\drivers\peripheral\sysclk\" directory of the application project after a successful build.

Refer to the PDL documentation for a detailed description of the complete API. To access this document, right-click on the Component symbol on the schematic and choose the "**Open PDL Documentation…**" option in the drop-down menu.

The Component generates the divider number and the divider type as documented in the Preprocessor Macros section. Pass the generated macros to the Peripheral Clock Dividers API in the sysclk driver to configure the divider.

The Clock Component also provides an instance-based reference functions that provide wrappers around the PDL API. These are provided as reference only.

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.

Preprocessor Macros

The Clock Component generates the following preprocessor macro(s). Note that each macro is prefixed with the instance name of the Component (e.g., "Clock_1").

#define Clock_1_DIV_NUM ((uint32_t)Clock_1__DIV_NUM)

The peripheral clock divider number

#define Clock_1_DIV_TYPE ((cy_en_divider_types_t)Clock_1__DIV_TYPE)

The peripheral clock divider type

Code Examples and Application Notes

Code Examples

PSoC Creator provides access to code examples in the Code Example 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 Code Example" topic in the PSoC Creator Help for more information.

Functional Description

The Clock Component is used to express a small subset of the device Clock system. Namely, it is primarily used for assigning and setting a peripheral clock divider to a peripheral. In special cases, it can also be used to provide a connection to an “existing” clock for specific peripherals.

Placement

When a Clock Component is placed on the schematic and connected to a peripheral clock input, PSoC Creator selects an appropriate clock divider based on the Component configuration and the Component to which it is connected.

Start on Reset

For all instances of the Clock Component in the design, the Clock Editor contains a "Start on Reset" option. This option is enabled by default, and it will allow PSoC Creator to automatically configure and enable the clock at startup. If this is not the desired behavior, unselect this option, and manually enable the clock divider in the application code. For more details, see the Clock Editor section of the PSoC Creator Help.

MISRA-C 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. Refer to **PSoC Creator Help > Building a PSoC Creator Project > Generated Files (PSoC 6)** for information on MISRA compliance and deviations for files generated by PSoC Creator.

The Clock Component does not have any specific deviations.

Registers

Refer to the device *Technical Reference Manual* for more information about the system clock control registers.

Resources

A Clock Component consumes a single peripheral clock divider.

DC and AC Electrical Characteristics

Specifications are valid for -40 °C ≤ T_A ≤ 85 °C and 1.71 V to 3.6 V except where noted.

Note Final characterization data for PSoC 6 devices is not available at this time. Once the data is available, the Component datasheet will be updated on the Cypress web site.

DC Specifications

Spec ID	Parameter	Description	Min	Typ	Max	Units	Details/Conditions
SID218	I_{IMO1}	IMO operating current	-	-	100	µA	8 MHz
SID316	IDD_MHz	MHz ECO operating current	-	1	-	mA	-
SID307P	PLL_IDD	PLL operating current	-	-	1.3	mA	-
SID231	I_{ILO2}	ILO operating current	-	0.3	1.05	µA	32 kHz
SID318	WCO_kHz	WCO operating current	-	0.25	1	µA	With 32-kHz crystal. After trimming, including aging and temperature drift.
SID321E	ESR32K	WCO crystal equivalent series resistance	-	50	-	kΩ	-
SID322E	PD32K	WCO drive level	-	-	1	µW	-

AC Specifications

Spec ID	Parameter	Description	Min	Typ	Max	Units	Details/Conditions
SID223	$F_{IMOTOL1}$	IMO frequency variation	-	-	±1	%	Centered on 8 MHz
SID226	$T_{STARTIMO}$	IMO startup time	-	-	12	µs	-
SID227	$T_{JITRMSIMO1}$	IMO RMS jitter at 8 MHz	-	156	-	ns	-
SID317	F_MHz	ECO crystal frequency range	4	-	33	MHz	-
SID305	EXTCLKFREQ	External clock input frequency	0	-	100	MHz	-

Spec ID	Parameter	Description	Min	Typ	Max	Units	Details/Conditions
SID306	EXTCLKDUTY	External clock duty cycle	45	-	55	%	Measured at VDD/2
SID305P	PLL_LOCK	Time to achieve PLL lock	-	40	-	µs	-
SID306P	PLL_OUT	Output frequency from PLL	-	-	150	MHz	-
SID234	$T_{STARTILO1}$	ILO startup time	-	-	2	ms	-
SID236	$T_{ILODUTY}$	ILO duty cycle	40	50	60	%	-
SID237	$F_{ILOTRIM1}$	ILO 32-kHz trimmed frequency	15	32	50	kHz	-
SID319	F_kHz	WCO 32-kHz trimmed frequency	-	32.768	-	kHz	-
SID320	Ton_kHz	WCO startup time	-	-	500	ms	-
SID320E	FTOL32K	WCO frequency tolerance	-	50	250	ppm	-

References

Detailed information regarding the device clock structure can be found in the device *Technical Reference Manual* and the sysclk API reference documentation.

Component Changes

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

Version	Description of Changes	Reason for Changes / Impact
1.0.e	Minor datasheet edits.	
1.0.d	Minor datasheet edits.	
1.0.c	Removed Component API documentation.	These are for reference only. PDL API should be used.
	Removed MISRA deviations.	All deviations are captured in sysclk driver documentation.
	General datasheet updates.	Renamed SysClk_PDL to Clock and general clean up.
1.0.b	Updated datasheet.	Replaced Configure dialog screen capture. Updated the MISRA table.
1.0.a	Updated datasheet.	Fixed a typo. Regenerated API documentation.
1.0	New Component	

© Cypress Semiconductor Corporation, 2016-2017. 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.

