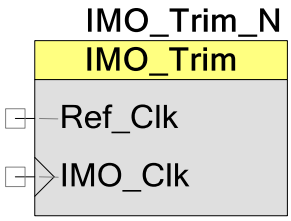
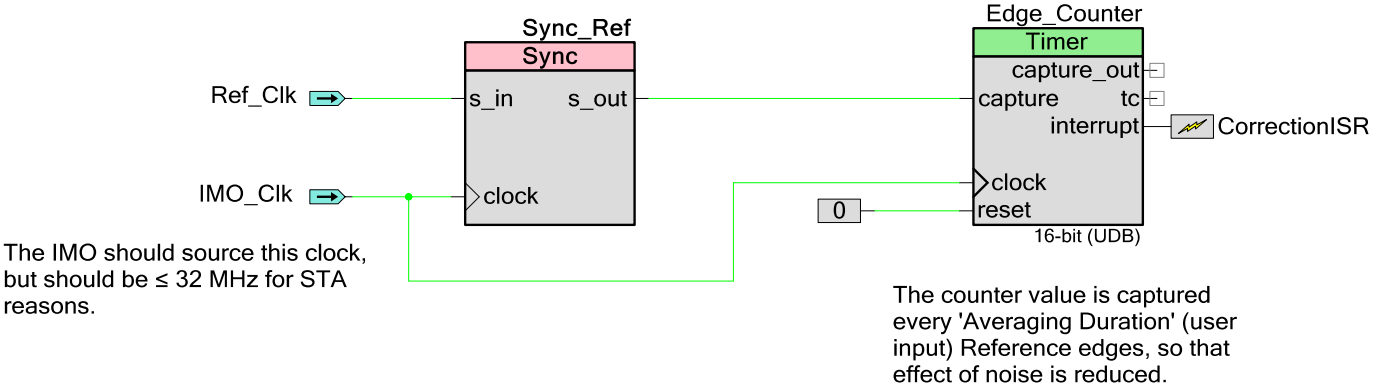
IMO_Trim
1.20

Features



- Measures the frequency error of the Internal Main Oscillator (IMO)
- Trims the IMO using an external higher-accuracy clock

Schematic



General Description

The IMO in PSoC 3 and PSoC 5 is usually the source of the system clock. IMO accuracy decreases as its frequency increases. However, by calibrating the IMO at run time with an accurate clock source such as an external kHz crystal, a high degree of accuracy can be obtained with minimal hardware cost. This further enhances PSoC's best-in-class internal oscillator performance.

The sign and magnitude of the IMO frequency error is calculated by counting the number of edges of the IMO for a known number of reference cycles. This information enables the adjustment of dedicated trim registers to correct the IMO frequency. The trimming process continues until the IMO is within an acceptable accuracy range.

When to Use the IMO_Trim component

Use the IMO_Trim component any time you need an accurate MHz clock for the price of a 32 kHz crystal.

An accurate IMO has a wide range of applications, particularly for high-reliability communication. MHz crystals are generally more expensive than kHz crystals. Thus, calibrating the IMO with

respect to a kHz crystal can reduce bill-of-material costs. For example, a high accuracy IMO eliminates the need for a MHz crystal for UART communication in PSoC 5.

Input/Output Connections

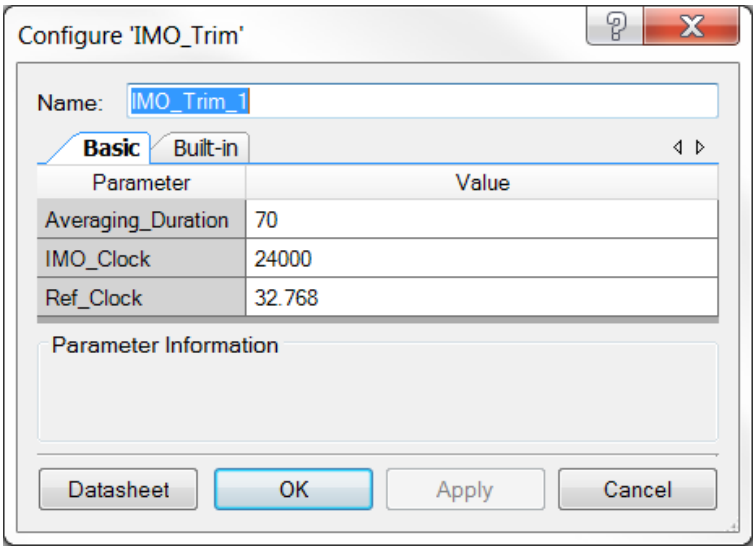
There are two inputs and no outputs to the IMO_Trim component.

Input	May Be Hidden	Description
Ref_Clk	N	Reference clock from which IMO derives higher accuracy
IMO_Clk	N	IMO-derived clock such as BUS_CLK or IMO itself

Component Parameters

Drag the IMO_Trim component on to your design and double-click it to open the Configure dialog box. The Configure dialog box has two tabs. The Basic tab contains three parameters which configure the IMO_Trim component.

Basic Options



Averaging_Duration

Number of reference cycles over which IMO frequency is counted – the larger the value, the more accurate the result. It is of type uint8.

The condition Averaging_Duration × IMO_Clock/Ref < 65535 must be valid with a margin greater than or equal to IMO Error in percent.

IMO_Clock

The frequency in kHz of the IMO_Clk input.

Ref_Clock

The frequency in kHz of the Ref_Clk input.

Placement

The timer used in this component is UDB-based. PSoC Creator automatically places it within the UDB array in the best possible configuration.

Resources

The total available resources may vary across device part numbers. The IMO_Trim component uses the following device resources:

Resources	Count
Datapath	2 of 24
Status cells	1 of 24
Control/count7 cells	2 of 24
PLDs	3 of 48
Macrocells	8 of 192
Interrupts	1 of 32
Digital clock dividers	0 of 8
Sync blocks	1 of 92
Flash (bytes)	725
RAM (bytes)	24
CPU cycles consumed per iteration	PSoC 3: 312 PSoC 5: 147

Application Programming Interface

Application Programming Interface (API) routines allow you to interact with the component using firmware. 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 IMO_Trim_1 to the first instance of the 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 of the component is taken to be IMO_Trim.

Functions

For readability, the instance name used in the following table is IMO_Trim.

Function	Description
IMO_Trim_Start()	Starts the component but does not begin the trim iterations
IMO_Trim_Stop()	Stops the component
IMO_Trim_BeginTrimming()	Enables the ISR to begin trimming the IMO
IMO_Trim_CheckStatus()	Returns current status of IMO Trim
IMO_Trim_CheckError()	Returns IMO frequency error value in PPM or status flags
IMO_Trim_RestoreTrim()	Restores IMO factory trim
IMO_Trim_Sleep()	Prepares the component for sleep
IMO_Trim_Wakeup()	Prepares the component for wakeup

void IMO_Trim_Start(void)

Description: Starts the timer, starts and assigns a vector to CorrectionISR.

Parameters: None

Return Value: None

Side Effects: Sets the isrOff global variable.

void IMO_Trim_Stop(void)

Description: Stops the timer and the CorrectionISR.

Parameters: None

Return Value: None

Side Effects: None

void IMO_Trim_BeginTrimming(void)

Description: Enables the CorrectionISR – this allows the trimming procedure to begin.

Parameters: None

Return Value: None

Side Effects: Clears the isrOff global variable.

uint8 IMO_Trim_CheckStatus (void)

Description: Checks whether the IMO frequency is within the acceptable error range. Requires the Edge_Counter to be running.

Parameters: None

Return Value:

Value	Description
IMO_Trim_IS_INACCURATE	IMO frequency is outside error range - Typically if return is IS_INACCURATE, BeginTrimming() is called.
IMO_Trim_IS_ACCURATE	IMO is within error range
IMO_Trim_NOT_READY	Timer Capture is not Full
IMO_Trim_ISR_ON	CorrectionISR is running

Side Effects: Clears Timer capture FIFO - so should not be used immediately after the CheckError function.

int16 IMO_Trim_CheckError(void)

Description: Calculates and returns IMO frequency error in parts per million (PPM)

Parameters: None

Return Value: Returns an error value in PPM if the CorrectionISR is not running and the timer capture is full. Otherwise, it returns NOT_READY or ISR_ON.

Value	Description
Error in PPM	IMO frequency error
IMO_Trim_NOT_READY	Timer Capture is not Full
IMO_Trim_ISR_ON	CorrectionISR is running

Side Effects: Clears Timer capture FIFO - so should not be used immediately after the CheckStatus function.

void IMO_Trim_RestoreTrim(void)

Description: Restores factory trim value.

Parameters: None

Return Value: None

Side Effects: Stops CorrectionISR, sets the isrOff global variable.

void IMO_Trim_Sleep(void)

Description: Prepares the component for sleep by disabling the CorrectionISR and putting the Timer to sleep.

Parameters: None

Return Value: None

Side Effects: Sets the isrOff global variable.

void IMO_Trim_Wakeup(void)

Description: Wakes up the Timer, sets the IMO Trim registers with last known trim values and disables the CorrectionISR. To re-initiate trim iterations, call the BeginTrimming function.

Parameters: None

Return Value: None

Side Effects: Sets the isrOff global variable, changes IMO trim register values.

DC and AC Electrical Characteristics

AC Specifications

Parameter	Description	Conditions	Min	Typ	Max	Units
	Trimmable IMO frequency*		3		62.6	MHz
	IMO frequency accuracy*			0.05		%

* Not characterized

Concept Component Changes

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

Version	Description of Changes
1.0	Initial design
1.10	Added CheckError, RestoreTrim APIs
1.20	Fixed two bugs: frequency range for PSoC 5, and ensuring `$INSTANCE_NAME`_REF is cast to float.

© Cypress Semiconductor Corporation, 2012. 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.