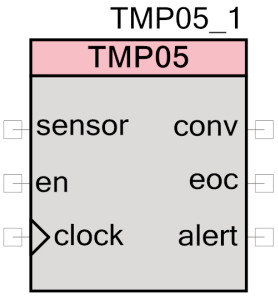
TMP05 Interface Component

1.0



Features

- Supports up to four TMP05 or TMP06 digital temperature sensors connected in daisy chain mode
- Continuous and one-shot modes of operation
- Supports frequencies from 100 to 500 kHz
- Supports temperature range from 0 to 70 Celsius degree

General Description

The TMP05 Temp Sensor Interface component is a simple, easy to use component capable of interfacing with Analog Device's TMP05/06 digital temperature sensors in daisy chain mode. Designers can configure and monitor the temperature readings in one of two ways: 1) continuous monitoring option enables the designer to record temperatures in a continuous fashion, at a sample rate dictated by the temperature sensor(s), while 2) one-shot mode triggers the temperature measurement at a rate controllable by the user. The first mode is intended for use in an environment where temperature variations are abrupt and need to be monitored frequently, while the second option should be used when temperature measurements only need to be sampled once in a while or in applications where minimizing power consumption is important.

When to Use a TMP05 Interface

The TMP05 Temp Sensor Interface component should be used in Thermal Management applications that require monitoring of the temperatures of several remote subsystems. This component is designed to directly interface to TMP05/06 temperature sensors that produce a PWM output whose duty cycle is directly related to ambient temperature.

Modes of Operation

Two modes of operation are supported in the TMP05 Temp Sensor Interface component: continuous and one-shot.

1. In continuous mode, temperature measurements occur at the fastest possible rate, dictated by the conversion time of the sensors.
2. In one-shot mode, temperature measurements occur at a rate dictated by application firmware. In this mode, the designer can set the rate of temperature monitoring

Input/Output Connections

This section describes the various input and output connections for the TMP05Intf.

sensor – Input

For single sensor applications, this input should be connected to the TMP05 temperature sensor output. That output is a pulse-width modulated digital signal whose duty cycle is proportional to temperature.

For multi-sensor applications where the TMP05 temperature sensors are daisy chained together, this input should be connected to the output of the final TMP05 temperature sensor in the daisy chain.

When interfacing to open-drain output TMP06 sensors, this signal requires a pull-up resistor for proper operation. This can be achieved using internal pull-up resistors on the selected PSoC GPIO pin or externally on the circuit board

en – Input

Active high enable. Set this pin to a low level to disable the component.

clock – Input

Clock that operates this block. Must be between 100 kHz and 500 kHz.

conv – Output

Conversion Start output. To start monitoring the temperatures, TMP05 sensors require a trigger input with specific timing constraints on low and high times. This output provides a trigger input to the first TMP05 sensor connected in daisy-chain mode.

eoc – Output

End of conversion output. This signal pulses active high for one clock cycle when a new temperature measurement is available. When sensors are connected in daisy chain mode, this signal indicates a measurement has been made on all sensors. The eoc signal should not assert until data is ready to be obtained by the APIs (after any internal processing).

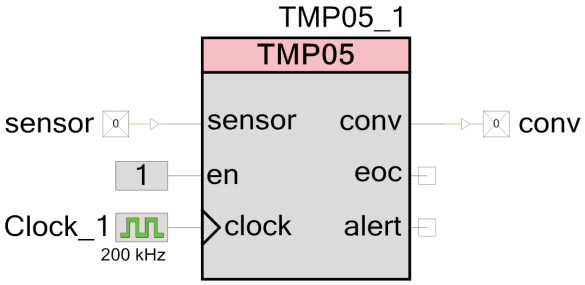
alert – Output

Temperature conversion error output. This signals pulses active high one clock cycle when the temperature sensor measurement times out. This could occur due to a fault in one or more of the attached temperature sensors or in the case of connection failure to the sensor(s).

Schematic Macro Information

This section contains pertinent information for the schematic macro in the TMP05 Interface.

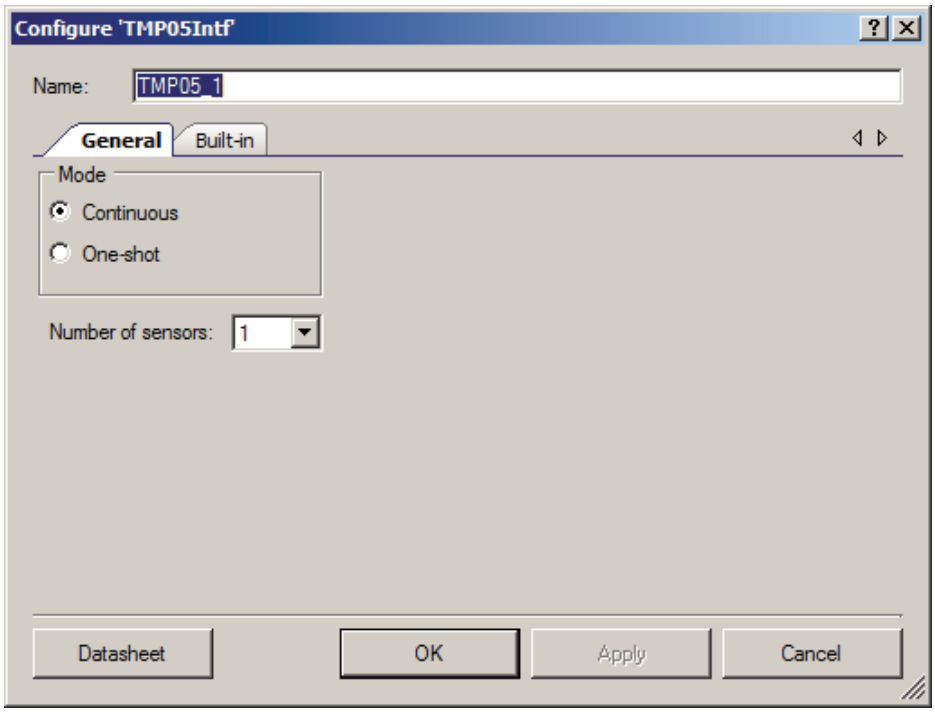
The default TMP05 Interface in the Component Catalog is a schematic macro using a TMP05 Interface component with default settings. It is connected to clock, a Logic High component and Digital Input/Output pins.



Component Parameters

Drag a TMP05 Interface component onto your design and double-click it to open the Configure dialog. This dialog has one tab to guide you through the process of setting up the TMP05 Interface component.

General Tab



The **General** tab provides the following parameters.

Continuous

This parameter is used to specify the operating mode of the component. This parameter is a Boolean value to select between continuous and one-shot modes of operation. Valid values for this parameter are true and false. Default setting is true, enabling a continuous mode of operation.

Number of Sensors

This parameter is used to specify the number of TMP05 temperature sensors that are connected in daisy-chain mode. Currently the component can support up to four TMP05 temperature sensors. Valid range for this parameter is 1-4. Default setting is 1.

Clock Selection

There is no internal clock in this component. You must attach a clock source.

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 "TMP05_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 "TMP05".

Function	Description
TMP05_Start()	Starts the component
TMP05_Stop()	Stops the component
TMP05_Init()	Initializes the component
TMP05_Enable()	Enables the component
TMP05_Trigger()	Triggers the interfaced TMP05 sensors to start temperature measurement depending upon the mode of operation
TMP05_ GetTemperature()	Calculates the temperature(s) in degrees Celsius
TMP05_SetMode()	Sets the operating mode of the component
TMP05_DiscoverSensors()	Automatically detects how many temperature sensors are daisy-chained to the component
TMP05_ConversionStatus()	Returns current state of temperature conversion (busy, completed or error)
TMP05_SaveConfig()	Saves the current state of the component before entering low power mode

Function	Description
TMP05_RestoreConfig()	Restores the previous state of the component after waking from low power mode
TMP05_Sleep()	Puts the component into low power mode
TMP05_Wakeup()	Wakes the component up from low power mode

Global Variables

Variable	Description
TMP05_initVar	The initVar variable is used to indicate initial configuration of this component. This variable is pre-pended with the component name. The variable is initialized to zero and set to 1 the first time TMP05_Start() is called. This allows for component initialization without re-initialization in all subsequent calls to the TMP05_Start() routine. If re-initialization of the component is required the TMP05_Stop() routine should be called followed by the TMP05_Init() and TMP05_Enable().
TMP05_busyFlag	The busyFlag variable is used to indicate component measuring. The variable is initialized to zero after component initialization is used or after measuring is finished and set to 1 after TMP05_Trigger API is called.

void TMP05_Start(void)

Description: Starts the component. Calls the TMP05_Init() API if the component has not been initialized before. Calls the enable API.

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Stop (void)

Description: Disables and stops the component

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Init(void)

Description: Initializes the component

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Enable(void)

Description: Enables the component

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Trigger (void)

Description: Provides a valid strobe/trigger output on the conv terminal.

Parameters: None

Return Value: None

Side Effects: None

int16 TMP05_GetTemperature (uint8 SensorNum)

Description: Calculates the temperature in degrees Celsius

Parameters: uint8 SensorNum - The TMP05 sensor number from 0..3

Return Value: Temperature in 1/100ths degrees C of the requested sensor.

Side Effects: None

void TMP05_SetMode (uint8 mode)

Description: Sets the operating mode of the component

Parameters: uint8 mode - operating mode:
- MODE_CONTINUOUS - Continuous mode
- MODE_ONESHOT - One-shot mode

Return Value: None

Side Effects: None

uint8 TMP05_DiscoverSensors (void)

Description: This API is provided for applications that might have a variable number of temperature sensors connected. It automatically detects how many temperature sensors are daisy-chained to the component. The algorithm starts by checking to see if the number of sensors actually connected matches the NumSensors parameter setting in the Basic Tab of the component customizer. If not, it will retry assuming 1 less sensor is connected. This process will repeat until the actual number of sensors connected is known.

Confirming whether or not a sensor is attached or not takes a few hundred milliseconds per sensor per iteration of the algorithm. To limit the sensing time, reduce the NumSensors parameter setting in the Basic Tab of the component customizer to the maximum number of possible sensors in the system.

Parameters: none

Return Value: uint8 - representing the number of sensors actually connected (0..4)

Side Effects: None

uint8 TMP05_ConversionStatus (void)

Description: Enables firmware to synchronize with the hardware

Parameters: none

Return Value: uint8 status code:
- STATUS_IN_PROGRESS - Conversion in progress
- STATUS_COMPLETE - Conversion complete
- STATUS_ERROR - Sensor Error

Side Effects: None

void TMP05_SaveConfig (void)

Description: Saves the user configuration of the TMP05 non-retention registers. This routine is called by TMP05_Sleep() to save the component configuration before entering sleep.

Parameters: None

Return Value: None

Side Effects: None

void TMP05_RestoreConfig (void)

Description: Restores the user configuration of the TMP05 non-retention registers. This routine is called by TMP05_Wakeup() to restore the component configuration when exiting sleep.

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Sleep (void)

Description: Stops the TMP05 operation and saves the user configuration along with the enable state of the TMP05.

Parameters: None

Return Value: None

Side Effects: None

void TMP05_Wakeup (void)

Description: Restores the user configuration and restores the enable state.

Parameters: None

Return Value: None

Side Effects: None

MISRA 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. Project deviations are described in the MISRA Compliance section of the *System Reference Guide* along with information on the MISRA compliance verification environment.

The TMP05 Temp Sensor Interface component does not have any specific deviations.

This component has the following embedded components: Interrupt, Status Register, Control Register. Refer to the corresponding component datasheet for information on their MISRA compliance and specific deviations.

Sample Firmware Source Code

PSoC Creator provides numerous example projects that include schematics and example code in the Find Example Project 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 Example Project” topic in the PSoC Creator Help for more information.

Interrupt Service Routine

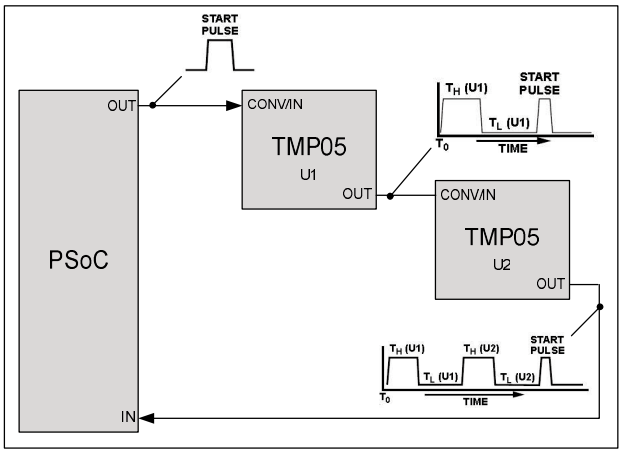
The following is a C language example of the generated Interrupt Service Routine locations where the user should enter custom ISR code:

```
/* `#START TMP05_INT_HEADER` */

/* `#END` */
```

Functional Description

The component uses one input and one output pin for interfacing with a single sensor or multiple TMP05 sensors in daisy chain mode (shown below).



The TMP05 sensor has three modes of operation:

- Continuously converting mode
- Daisy-chained mode
- One-shot mode

A three-state function control input pin (FUNC) sampled at power-up determines the mode in which the device will operate. Setting the FUNC pin to a high state allows multiple TMP05s to be connected in daisy chained mode. In that mode, multiple TMP05 temperature sensors can be connected, which enables TMP05 Interface component to read all the sensors through the same two-wire interface.

In this configuration, TMP05 Interface component generates a “Start” pulse to begin a new temperature-to-PWM conversion cycle. The output of the first TMP05 sensor begins with its own PWM output, followed by a new Start pulse that it generated internally. The output of the second TMP05 sensor begins with the PWM output of the previous sensor, followed by its own PWM output and terminated by a new Start pulse that it generated internally.

When more sensors are daisy-chained in this fashion, the final return signal to PSoC contains the PWM outputs from all sensors starting with the output of the first sensor, followed by the output from the second sensor and so on until the terminating Start pulse is detected. Temperature-to-PWM conversion then stops until TMP05 Interface component generates another Start pulse. The system-level connection scheme and expected waveforms are above.

Temperature calculated using the equation provided in the TMP05/06 datasheet: Temperature = 421 – 751 * (TH/TL).

Resources

The TMP05 component is placed throughout the UDB array. The component utilizes the following resources.

Configuration	Resource Type					
	Datapath Cells	Macrocells	Status Cells	Control Cells	DMA Channels	Interrupts
All	2	11	1	1	–	1

API Memory Usage

The component memory usage varies significantly, depending on the compiler, device, number of APIs used and component configuration. The following table provides the memory usage for all APIs available in the given component configuration.

The measurements have been done with the associated compiler configured in Release mode with optimization set for Size. For a specific design the map file generated by the compiler can be analyzed to determine the memory usage.

Configuration	PSoC 3 (Keil_PK51)		PSoC 5 (GCC)		PSoC 5LP (GCC)	
	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes	Flash Bytes	SRAM Bytes
Continuous	656	8	516	11	516	11
One shot	654	8	512	11	512	11

DC and AC Electrical Characteristics

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

TMP05 Interface AC Specifications

Parameter	Description	Min	Typ	Max	Units
f_{CLOCK}	Component clock frequency	100	–	500	kHz

Performance

The measurements below have been gathered using a CPU speed of 24 MHz, with the associated compiler configured in Release mode. These numbers should be treated as approximations and used to determine necessary trade-offs.

GetTemperature API

No. of computation cycles (8051)	No. of computation cycles (ARM cortex M3)
1160	70

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	Version 1.0 is the first release of the TMP05Intf component	

© 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 HALFICULAR 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.