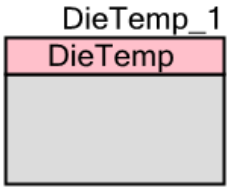
# Die Temperature (DieTemp) 2.10

## Features




- Accuracy of ±5 °C
- Range –40 °C to +140 °C (0xFFD8 to 0x008C)
- Blocking and non-blocking API

## General Description

The Die Temperature (DieTemp) component provides an Application Programming Interface (API) to acquire the temperature of the die. The System Performance Controller (SPC) is used to get the die temperature. The API includes blocking and non-blocking calls.

### When to Use a DieTemp

Use a DieTemp component when you want to measure the die temperature of a device.

## Input/Output Connections

There are no Input/Output Connections on the DieTemp component. It is a software component only.

## Component Parameters

The DieTemp has no configurable parameters other than standard Instance Name and Built-in parameters.

# 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 “DieTemp_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 “DieTemp.”

Note that the device should not enter low power modes during temperature measurements. After calling DieTemp_Start() you should wait for the DieTemp_Query() API to report the status of the request that is different from CYRET_STARTED. For more information see the DieTemp_Query() API description.

## Functions

| Function | Description |
|---|---|
| DieTemp_Start() | Starts the SPC command to get the die temperature |
| DieTemp_Stop() | Stops the temperature reading |
| DieTemp_Query() | Queries the SPC to see if the temperature command is finished |
| DieTemp_GetTemp() | Sets up the command to get the temperature and blocks until finished |

### cystatus DieTemp_Start(void)

**Description:** Sends the command and parameters to the SPC to start a Die Temperature reading. This function returns before the SPC finishes. This function call must always be paired with a call to the DieTemp_Query() API to complete the Die Temperature reading..

**Return Value:** CYRET_STARTED if the SPC command was started successfully.
CYRET_UNKNOWN if the SPC command failed.
CYRET_LOCKED if the SPC was busy.

### void DieTemp_Stop(void)

**Description:** There is no need to stop or disable this component. This component is naturally a slave that sends request to SPC through SPC API of cy_boot and waits for data to be ready or in case of non-blocking operation it sends request to SPC and user can manually poll the result.

### cystatus DieTemp_Query(int16 * temperature)

**Description:** Checks to see if the SPC command started by DieTemp_Start() has finished. If the command has not finished, the temperature value is not read and returned. The caller will need to poll this function while the return status remains CYRET_STARTED.

This can be used only in conjunction with the DieTemp_Start() API to successfully get the correct Die Temperature.

The Die Temperature reading returned on the first sequence of DieTemp_Start() followed by DieTemp_Query() can be unreliable, so you must do this sequence twice and use the value returned from the second sequence.

**Parameters:** int16 * temperature: Address to store the temperature in degrees Celsius

**Return Value:** CYRET_SUCCESS if the temperature command completed successfully.

CYRET_UNKNOWN if the there was an SPC failure.

CYRET_STARTED if the temperature command has not completed.

CYRET_TIMEOUT if waited too long before reading data.

### cystatus DieTemp_GetTemp(int16 * temperature)

**Description:** Sends the command and parameters to the SPC to start a Die Temperature reading and waits until it fails or completes. This is a blocking API.

This function reads the Die Temperature twice and returns the second value to work around an issue in the silicon that causes the first value read to be unreliable.

**Parameters:** int16 * temperature: Address to store the temperature in degree of Celsius

**Return Value:** CYRET_SUCCESS if the command was completed successfully.

Along with additional status codes from DieTemp_Start() or DieTemp_Query().

## Sample Firmware Source Code

PSoC Creator provides many code examples that include schematics and example code in the Find 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.

## 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 DieTemp component does not have any specific deviations.

### 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 5LP (GCC) | |
|---|---|---|---|---|
| | Flash Bytes | SRAM Bytes | Flash Bytes | SRAM Bytes |
| Default | 339 | 0 | 210 | 0 |

## Resources

The DieTemp uses the on-chip Temperature sensor to measure the internal die temperature.

## DC and AC Electrical Characteristics

Specifications are valid for –40 °C ≤ $T_A$ ≤ 85 °C and $T_J$ ≤ 100 °C, except where noted. Specifications are valid for 1.71 V to 5.5 V, except where noted.

| Parameter | Description | Conditions | Min | Typical | Max | Units |
|---|---|---|---|---|---|---|
| | Temp sensor accuracy | Range: –40 °C to +85 °C | – | ±5 | – | °C |

# Component Changes

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

| Version | Description of Changes | Reason for Changes / Impact |
|---|---|---|
| 2.10 | New Component version to remove the long delay (> 6 ms) in the DieTemp_Query() function. | The DieTemp_Query() function had a long delay (> 6 ms) when the result was not ready. |
| 2.0.c | Minor datasheet edits. | |
| 2.0.b | Datasheet update. | Added a note to not enter low power modes during temperature measurements. |
| 2.0.a | Minor datasheets edits and updates.<br>Removed PSoC 5 references. | <br>PSoC 5 replaced by PSoC 5LP. |
| 2.0 | Removed CySpcStop() API call from DieTemp_Stop(). | SPC cannot be stopped from runtime chip operation. |
| | Added MISRA Compliance section. | The component does not have any specific deviations. |
| 1.80 | Added PSoC 5LP support. | |
| | DieTemp APIs were updated. | Due to changes to the cyboot SPC APIs. |
| 1.70 | The DieTemp_GetTemp() API was changed to read the DieTemp twice to fix an issue. | The DieTemp_GetTemp() API was returning the temperature even if the first read was unsuccessful. |
| 1.60 | The *DieTemp.c* file GetTemp API was edited to fix the power-on reset error output. | The DieTemp output on power-on reset was erroneous. |
| 1.50.a | Added characterization data to datasheet | |
| | Added information to the component that advertises its compatibility with silicon revisions. | The tool returns an error/warns if the component is used on incompatible silicon. This component is not compatible with PSoC 3 ES2 or PSoC 5. |
| | Minor datasheet edits and updates | |
| 1.50 | Switch from *cydevice.h* to *cydevice_trm.h*. | The *cydevice.h* file has been made obsolete, so the APIs and generated code provided with PSoC Creator are not included with *cydevice_trm.h*. |

© Cypress Semiconductor Corporation, 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.