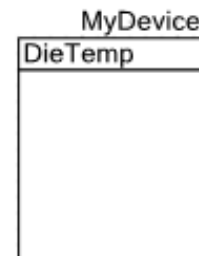


# Die Temperature (DieTemp)

1.0

## Features

- Accuracy of +/-5 °C
- Range -40°C to +140°C
- Blocking and non blocking API



## General Description

The Die Temperature (DieTemp) component provides an API to acquire the temperature of the die. The System Performance Controller (SPC) is used to acquire 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 the device.

## Input/Output Connections

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

## Parameters and Setup

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,

**PRELIMINARY**

variable, and constant symbol. For readability, the instance name used in the following table is "DieTemp".

Function	Description
DieTemp_Start	Starts the SPC command to get the die temperature
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)

<b>Description:</b>	Sends the command and parameters to the SPC to start a Die Temperature reading. This function returns before the SPC finishes. If this function is called successfully, the SPC will be locked and DieTemp_Query will have to be successfully called to unlock it. CySpcUnlock() can also be called if the caller decides not to finish the temperature reading.
<b>Parameters:</b>	void
<b>Return Value:</b>	CYRET_STARTED if the SPC command was started successfully. CYRET_UNKNOWN if the SPC command failed. CYRET_LOCKED if the SPC was busy.
<b>Side Effects:</b>	None

### cystatus DieTemp\_Query (int16 \* temperature)

<b>Description:</b>	Checks to see if the SPC command started by DieTemp_Start has finished. If the command has not finished, the temperature value is not written. The caller would poll this function until completion of the command.
<b>Parameters:</b>	int16 * temperature. Address to store the temperature.
<b>Return Value:</b>	CYRET_SUCCESS if the temperature command completed successfully. CYRET_UNKNOWN if there was an SPC failure. CYRET_STARTED if the temperature command has not completed.
<b>Side Effects:</b>	None

**PRELIMINARY**



## cystatus DieTemp\_GetTemp(int16 \* temperature)

<b>Description:</b>	Sends the command and parameters to the SPC to start a Die Temperature reading and waits until it fails or completes. After DieTemp_MAX_WAIT ticks, the function will return even if the SPC has not finished.
<b>Parameters:</b>	int16 * temperature. Address to store the temperature.
<b>Return Value:</b>	CYRET_SUCCESS if the command was completed successfully. CYRET_TIMEOUT if the command times out. Status codes from DieTemp_Start or DieTemp_Query.
<b>Side Effects:</b>	None

## Sample Firmware Source Code

The following is a C language example demonstrating the basic functionality of the DieTemp component. This example assumes the component has been placed in the design and named "DieTemp\_1."

```
#include <device.h>
#include <DieTemp_1.H>

void main()
{
    cystatus status;
    uint16 temperature;

    /* Blocking call. */
    DieTemp_1_GetTemp(&temperature);

    /* Non blocking method. */

    /* Start the command to get the temperature. */
    status = DieTemp_1_Start();
    if(status != CYRET_STARTED)
    {
        /* Handle error! */
    }

    while(status == CYRET_STARTED)
    {
        /* Do something useful. */

        /* Check to see if the command was completed. */
        status = DieTemp_1_Query (&temperature);
    }
}
```



**PRELIMINARY**

## References

N/A

© Cypress Semiconductor Corporation, 2009. 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® Creator™, Programmable System-on-Chip™, and PSoC Express™ are trademarks and PSoC® is a registered trademark 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**

