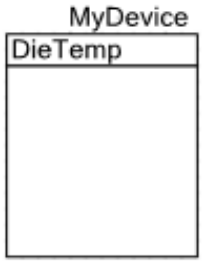
Die Temperature (DieTemp) 1.0

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,

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)

Description: 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.

Parameters: void

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.

Side Effects: None

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 written. The caller would poll this function until completion of the command.

Parameters: int16 * temperature. Address to store the temperature.

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.

Side Effects: None

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. After DieTemp_MAX_WAIT ticks, the function will return even if the SPC has not finished.
Parameters:	int16 * temperature. Address to store the temperature.
Return Value:	CYRET_SUCCESS if the command was completed successfully. CYRET_TIMEOUT if the command times out. Status codes from DieTemp_Start or DieTemp_Query.
Side Effects:	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);
   }

}
```

References

N/A