Internal Temperature Sensor Measurement Datasheet FlashTempV 2.40

Resources	PSoC® Blocks			API Memory (Bytes)		Pins (per External I/O)
	Digital	Analog CT	Analog SC	Flash	RAM	
CY8C29x66, CY8C27x43, CY8C24xx3, CY8C24x94, CY8C23x33, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xx3, CY8C28x52						
	0	0	1	74	3	

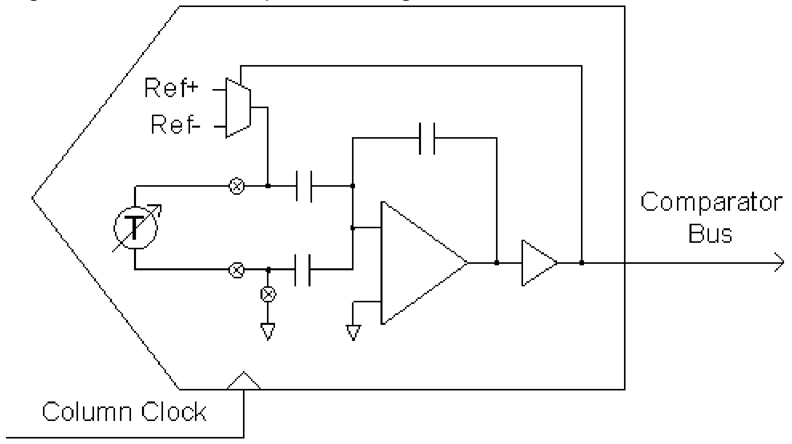
For one or more fully configured, functional example projects that use this user module go to www.cypress.com/psocexampleprojects

Features and Overview

- Range of -40 °C to +85 °C
- Accuracy of ± 20 °C with no calibration
- Single PSoC block implementation
- 8-bit 2's complement output in degrees Celsius

The FlashTemp User Module gives a coarse temperature measurement for the bFlashWriteBlock routine, which varies its programming pulse width with temperature. A single switch capacitor analog block is used and requires no calibration. The output of the FlashTemp User Module is the junction temperature of the PSoC microcontroller in a 2's complement format, with 1 count per °C.

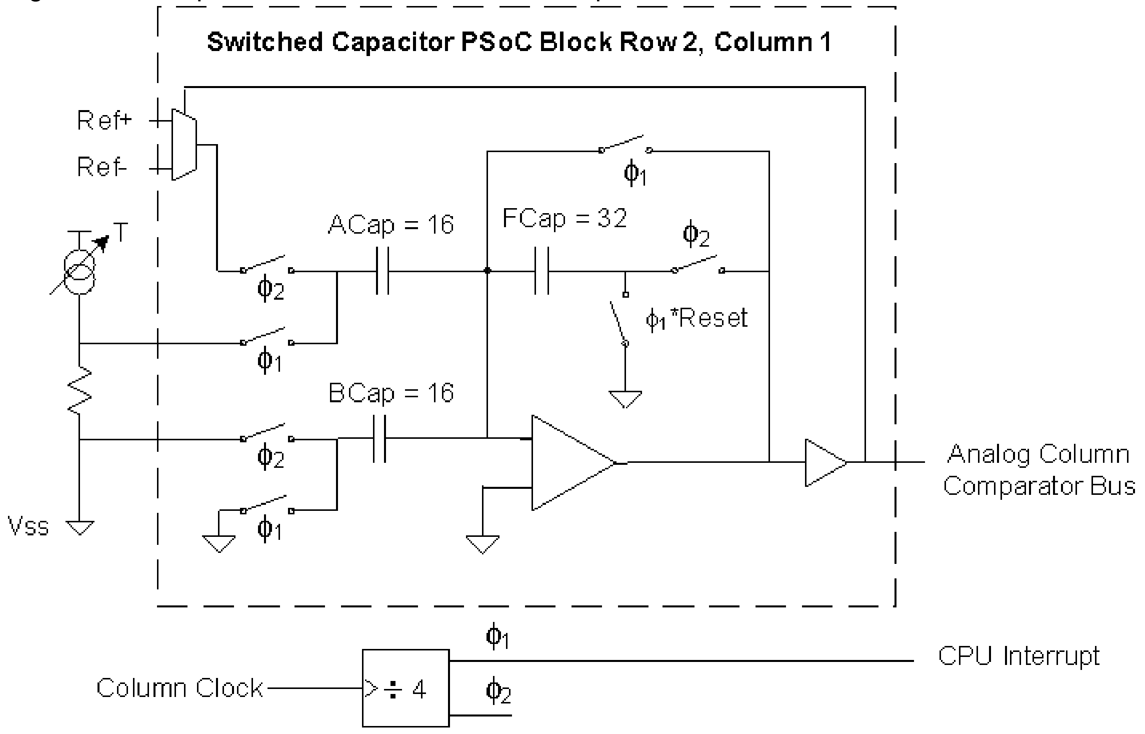
Figure 1. FlashTemp Block Diagram



Functional Description

The bandgap temperature sensor built into the PSoC microcontroller has an output voltage proportional to its absolute temperature. This output is approximately linear, with a nominal slope of 3 mV per °C. The FlashTemp Application Programming Interface (API) translates the temperature sensor to 1 count per °C. Since the output voltage is relative to absolute zero, -273 °C, a large offset voltage at room temperature limits the overall dynamic range and limits the accuracy without calibration. There is a large offset voltage from the temperature sensor, due to the fact that the operational temperature of the PSoC microcontroller is about 300 °C above absolute zero. This offset voltage is the primary limit to accuracy without calibration.

Figure 2. Simplified Schematic of the FlashTemp



The FlashTemp User Module is related to the Incremental ADC User Module (ADCINC12) but is configured to use differential inputs. The functions of the Timer and Counter digital blocks are replaced with software functions in an Interrupt Service Routine (ISR). This enables the FlashTemp to use a single analog switch capacitor PSoC block.

The analog block is configured as an integrator with differential inputs that can be reset. The inputs are the anode and cathode of the bandgap temperature sensor. The anode is connected to the ACap of the analog block. The cathode is connected to V_{ss} and to the BCap of the analog block. Depending on the output polarity of the integrator (as detected by the comparator), the reference voltage is either added to or subtracted from the difference of the inputs. The resulting charge is placed in the integrating cap (FCap with reset disabled). This mechanism attempts to pull the integrator voltage back toward AGND. The differential input voltage can be determined by the number of times the comparator output is positive.

The FlashTemp API includes an ISR that performs data collection. This ISR is linked to an interrupt generated by the falling edge of $ø_1$ which allows the comparator output to be sampled immediately after the $ø_1$ cycle of the analog column clock. The sampling is repeated 255 times and the number of positive outputs from the comparator is used to determine the temperature.

DC and AC Electrical Characteristics

Unless otherwise noted in the table below, V_{DD} = 4.75 V to 5.25 V or 3.0 V to 3.6 V; temperature is -40 °C to 85 °C.

Table 1. FlashTemp DC and AC Electrical Characteristics

Parameter	Conditions and Notes	Typical	Limit	Units
Resolution		3.3		°C
Accuracy		20[1]		°C
Column Clock F_{max}	CPU Clock set to 24 MHz[2]		960	kHz
Column Clock F_{min}	See Note 3		125	kHz
Sample Rate	At F_{max}		900	sps
Conversion Time	At F_{max}		1100	μsec

Electrical Characteristics Notes

1. At 12 MHz CPU and 250 kHz column clock.
2. Required to meet interrupt routine latency timing.
3. Due to voltage decay from capacitor leakage.

Timing

For optimum accuracy, the output of the comparator should be read once per $ø_1$ clock period. This read of the comparator occurs within the FlashTemp ISR. Therefore, the interrupt latency plus the execution time of the FlashTemp ISR should not take longer than one $ø_1$ clock period. If it does take longer than one $ø_1$ clock period, the accuracy will be degraded. The level of degradation is proportional to the number of comparator reads missed.

When the $ø_1$ interrupt is recognized, the worst-case time to complete the ISR is approximately 100 CPU clocks. In a simplified example where there are no other interrupts and interrupts are not disabled at any time while temperature is being collected, the relationship between the CPU clock and the analog column clock should be as given in Equation 1:

Equation 1

$$\frac{CloumnClock}{4} < \frac{CPUClock}{100}$$

When the CPU and analog column clocks are such that both the left and right side of Equation 1 are equal, 100 percent of the CPU bandwidth will be used.

Placement

The bandgap temperature sensor is only connected to one switch capacitor analog block in the PSoC microcontroller. PSoC Designer only allows placement of the FlashTemp User Module in the block with the temperature sensor.

The FlashTemp analog block drives the analog column comparator. Other blocks placed in this column may not drive the comparator.

Parameters and Resources

Note Global Resources shall be used to configure the parameters of this user module.

Column Clock

The user must give a column clock for the FlashTemp User Module. The column clock is divided by four to produce ϕ_1. The sample rate is computed as given in Equation 2:

Equation 2

$$SampleRate = \frac{ColumnClock}{255 \times 4}$$

In Equation 2, 255 is the number of times the integrator cap is sampled per temperature conversion, and 4 is the column clock divisor that generates ϕ_1.

Vcc Power

V_{cc} operating range, in the global resources, can be set to 3.3 V or 5 V.

Analog Power

Analog power, in the global resources, must be set at high.

Reference Mux Setting

The RefMux setting, in the global resources, must be set to either:

$(V_{dd}/2) \pm$ BandGap

or

(BandGap) ± BandGap

Opamp Bias

The Opamp Bias setting, in the global resources, can be set to either LOW or HIGH.

Application Programming Interface

The Application Programming Interface (API) routines are given as part of the user module to allow the designer to deal with the module at a higher level. This section specifies the interface to each function together with related constants provided by the “include" files.

Note

In this, as in all user module APIs, the values of the A and X register may be altered by calling an API function. It is the responsibility of the calling function to preserve the values of A and X before the call if those values are required after the call. This “registers are volatile" policy was selected for efficiency reasons and has been in force since version 1.0 of PSoC Designer. The C compiler automatically takes care of this requirement. Assembly language programmers must ensure their code observes the policy, too. Though some user module API function may leave A and X unchanged, there is no guarantee they will do so in the future.

For Large Memory Model devices, it is also the caller's responsibility to preserve any value in the CUR_PP, IDX_PP, MVR_PP, and MVW_PP registers. Even though some of these registers may not be modified now, there is no guarantee that will remain the case in future releases.

Entry points are given to start the data collection, poll to see if data collection has been completed, retrieve the result, and turn off the analog block. The sequence of events should be:

1. Call FlashTemp_Start to begin the data collection.
2. Use FlashTemp_fIsData to determine when the data collection is complete. After data collection is complete, the API will disable the interrupt and power down the FlashTemp User Module.
3. Call FlashTemp_cGetData to get the result.

FlashTemp_Start

Description:

This API sets the calibration data and temperature trimming values, turns on the Power of the analog block, enables the analog column interrupt, initializes a data accumulation variable. It turns off the auto reset option of the feedback cap (Fcap) so that this capacitor becomes an integrator for ADC; initializes the count variable with the number of interrupts to be processed for one temperature measurement.

This API provides only one sample processing and reading. After the one sample is read no processing is provided next. To read the samples continuously, you have to call FlashTemp_Start API every time before the next sample reading.

C Prototype:

```
void  FlashTemp_Start(void)
```

Assembly:

```
lcall  FlashTemp_Start
```

Parameters:

None

Return Value:

None

Side Effects:

You can modify the A and X registers by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP, MVW_PP, MVR_PP, IDX_PP page pointer registers are modified.

FlashTemp_fIsData

Description:

Checks the status of the temperature ADC process. Returns zero flag if data sample is not complete. Returns a non-zero value if the data sample is complete.

C Prototype:

```
CHAR  FlashTemp_fIsData(void)
```

Assembly:

```
lcall  FlashTemp_fIsData
```

Parameters:

None

Return Value:

The return value is non-zero if data sample is complete and the result has not been read. It is zero if data sample is not complete or the result has been read.

Side Effects:

You can modify the A and X registers by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

FlashTemp_cGetData

Description:

Returns the temperature ADC result (1°C/count). The value is the junction temperature of the PSoC microcontroller as a 2’s complement number. This function clears the internal data ready flag to mark the data as old.

C Prototype:

```
CHAR  FlashTemp_cGetData(void)
```

Assembly:

```
lcall  FlashTemp_cGetData
```

Parameters:

None

Return Value:

The temperature of the PSoC device.

Side Effects:

You can modify the A and X registers by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions. Currently, only the CUR_PP page pointer register is modified.

FlashTemp_Stop

Description:

Powers down the user module PSoC block and disables the interrupt.

C Prototype:

```
void  FlashTemp_Stop(void)
```

Assembly:

```
lcall  FlashTemp_Stop
```

Parameters:

None

Return Value:

None

Side Effects:

You can modify the A and X registers by this or future implementations of this function. The same is true for all RAM page pointer registers in the Large Memory Model (CY8C29xxx). When necessary, it is the calling function's responsibility to preserve the values across calls to fastcall16 functions.

Sample Firmware Source Code

The following sample assembly code starts the FlashTemp User Module, collects data, and saves it in a variable:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Description:
;   This sample code shows an example where the temperature is collected
;   in the background while the main loop continues to run. This example
;   collects the temperature once. Other code (e.g., an ISR) could call
;   FlashTemp_Start to start collecting temperature data again. This
;   could be keyed off of an event, such as the passage of a set amount
;   of time or an external signal.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

include "m8c.inc"       ; part specific constants and macros
include "memory.inc"    ; Constants & macros for SMM/LMM and Compiler
include "PSoCAPI.inc"   ; PSoC API definitions for all User Modules

area bss(RAM)
  cTemperature:    BLK  1
area text(ROM,REL)

export _main
```

```
_main:
   ; initialize the m8c
   mov  reg[INT_VC],0             ; clear any outstanding interrupts
   M8C_EnableGInt                 ; enable interrupts
   ; other initialization code could be placed here

   ; start the collection of temperature data
   call FlashTemp_Start           ; start the analog block

MainLoop:
   ; The processor can do other important work here. When the FlashTemp
   ; ISR has finish collecting the temperature data it will be read once.

   call FlashTemp_fIsData
cmp  A,0                       ; test for zero
   jz   NoTempYet                 ; data not ready

TempReady:
   call FlashTemp_cGetData        ; get the temperature data
   mov  [cTemperature],A          ; save the data for use later

NoTempYet:
   ; More processing could be done here if wanted.
   jmp  MainLoop                  ; keep doing the main loop
```

The same program in C is:

```
//**********************************************************************
//
//  Description:
//  This sample code shows an example where the temperature is collected
//  in the background while the main loop continues to run. This example
//  collects the temperature once. Other code (e.g., an ISR) could call
//  FlashTemp_Start to start collecting temperature data again. This
//  could be keyed off of an event, such as the passage of a set amount
//  of time or an external signal.
//
//**********************************************************************

#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules

char cTemperature;

void main(void)
{
   // initialize the m8c
   M8C_EnableGInt;
   // other initialization code could be placed here

   // start the collection of temperature data
   FlashTemp_Start();

while(1)
{
      // The processor can do other important work here.  When the FlashTemp
      // ISR has finished collecting data it will only be read once.

      if (FlashTemp_fIsData())
      {
         cTemperature = FlashTemp_cGetData();
      }
   }
}
```

Configuration Registers

TEMPERATURE is the switch capacitor block containing the bandgap temperature sensor. The block is set up to have inputs to the ACap and the BCap, and use FCap as an integrator that can be reset.

Table 2. Block TEMPERATURE: Register CR0

Bit	7	6	5	4	3	2	1	0
Value	1	0	0	1	0	0	0	0

Table 3. Block TEMPERATURE: Register CR1

Bit	7	6	5	4	3	2	1	0
Value	0	1	1	1	0	0	0	0

Table 4. Block TEMPERATURE: Register CR2

Bit	7	6	5	4	3	2	1	0
Value	0	1	1	0	0	0	0	0

Table 5. Block TEMPERATURE: Register CR3

Bit	7	6	5	4	3	2	1	0
Value	1	1	1	1	1	1	Power	

Power sets the power level for the analog block.

Table 6. Register INT_MSK0

Bit	7	6	5	4	3	2	1	0
Value						Acolumn1		

Acolumn1 enables the interrupt from Analog Column 1. This is set to occur on the rising edge of $ø_1$.

Table 7. Register CMP_CR

Bit	7	6	5	4	3	2	1	0
Value			COMP1					

COMP1 indicates the state of the analog comparator bus for column 1.

Version History

Version	Originator	Description
2.2	DHA	Added Version History
2.30	DHA	1. Added SSCParamBlk RAM area for internal user module usage. 2. Added define `@INSTANCE_NAME`_TABLE_1.
2.30.b	DHA	Updated the "Parameters and Resources" section to notify the user that Global Resources will be used to configure the parameters of the FlashTemp User Module.
2.40	MYKZ	Corrected Start fuction to include FLS_PR1 register initilization. This eliminates cases where incorrect temperature values were returned after flash write operations.

Note PSoC Designer 5.1 introduces a Version History in all user module datasheets. This section documents high level descriptions of the differences between the current and previous user module versions.