

内部温度传感器测量数据手册 FlashTempV 2.30

Copyright © 2012 Cypress Semiconductor Corporation. All Rights Reserved.

资源	PSoC® 模块			API 存储器 (字节)		引脚 (每个外部 I/O)
	数字	模拟 CT (连续时序)	模拟 SC (开关电容)	闪存	RAM	
CY8C29x66, CY8C27x43, CY8C24xx3, CY8C24x94, CY8C23x33, CY8CLED04/08/16, CY8CLED0xD, CY8CLED0xG, CY8C28x45, CY8CPLC20, CY8CLED16P01, CY8C28xx3, CY8C28x52						
	0	0	1	74	3	

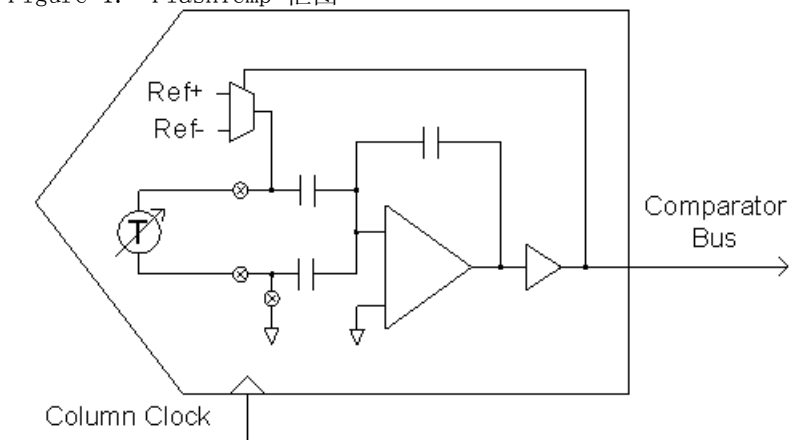
如需一个或多个使用此用户模块且完全配置的功能性示范项目，请转到
www.cypress.com/psoexampleprojects

功能和概述

- 量程: -40°C 至 $+85^{\circ}\text{C}$
- 精度: $\pm 20^{\circ}\text{C}$, 无校准
- 单一 PSoC 模块实现
- 8 位的二进制补码输出, 以摄氏度表示

FlashTemp 用户模块为 bFlashWriteBlock 子程序提供粗略温度测量, 其编程脉冲宽度随温度的变化而变化。它使用单一开关电容模拟模块, 无需校准。FlashTemp 用户模块的输出是 PSoC 微控制器的结温, 以二进制补码格式表示, 每摄氏度一个计数。

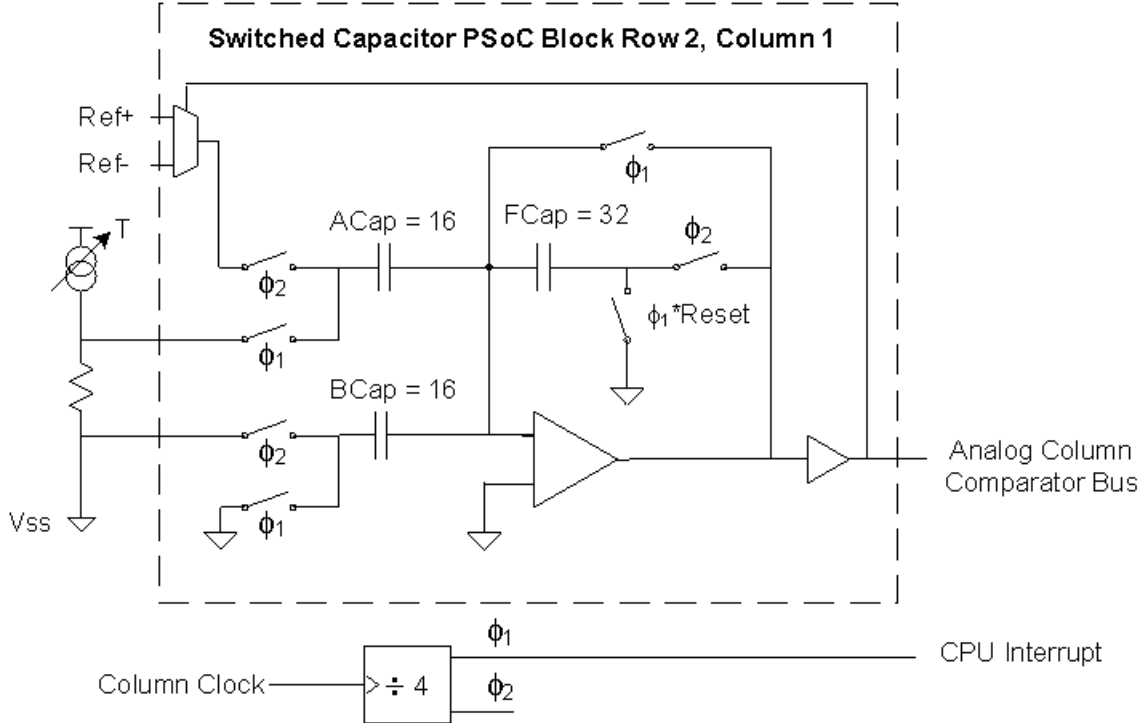
Figure 1. FlashTemp 框图



功能描述

PSoC 微控制器内置的带隙温度传感器的输出电压与其绝对温度成正比。此输出接近线性，额定斜率为每摄氏度 3 mV。FlashTemp 应用程序编程接口 (API) 将温度传感器转换为每摄氏度 1 个计数。由于输出电压和绝对零度 (-273°C) 有关，因此在室温下的大偏移电压会限制总体动态范围和未校准时的精度。因为 PSoC 微控制器的运行温度大概比绝对零度高 300°C ，温度传感器上会有大偏移电压。此偏移电压是未校准精度的主要限制。

Figure 2. FlashTemp 的简化原理图



FlashTemp 用户模块与增量型 ADC 用户模块 (ADCINC12) 相关，但能够配置使用差分输入。定时器和计数器的数字模块功能被中断服务子程序 (ISR) 的软件功能替代。这使 FlashTemp 能够使用单一模拟开关电容 PSoC 模块。

模拟模块配置为可复位的差分输入积分器。这些输入是带隙温度传感器的阳极和阴极。阳极与模拟模块的 ACap 相连。阴极与 V_{SS} 和模拟模块的 BCap 相连。根据积分器的输出极性 (由比较器检测)，对差分输入增加或减少参考电压。生成的电荷置于积分电容中 (禁用复位的 FCap)。此机制意在将积分器电压下拉至 AGND。差分输入电压取决于比较器正向输出的次数。

FlashTemp API 包括执行数据收集的 ISR。此 ISR 与 ϕ_1 下降沿生成的中断相连，以便在模拟列时钟完成 ϕ_1 循环后对迅速对比较器输出进行采样。此采样重复 255 次，且比较器正向输出次数被用来确定温度。

直流和交流电气特性

除非下表另行说明，否则 $V_{DD} = 4.75\text{ V}$ 至 5.25 V 或 3.0 V 至 3.6 V ；温度为 -40°C 至 85°C 。

Table 1. FlashTemp 直流和交流电气特性

参数	条件和注释	典型值	限制	单位
分辨率		3.3		°C
精度		20 ¹		°C
模拟列时钟 F _{max}	CPU 时钟设置为 24 MHz ²		960	kHz
模拟列时钟 F _{min}	见注释 3		125	kHz
采样率	F _{max} 时		900	sps
转换时间	F _{max} 时		1100	µsec

电气特性说明

1. CPU 速度为 12 MHz；模拟列时钟速度为 250 kHz。
2. 需要满足中断子程序延迟时序。
3. 电容泄露导致电压衰减。

时序

为了获得最佳的精度，比较器的输出应每 ϕ_1 时钟周期读取一次。比较器在 FlashTemp ISR 内读取。因此，中断延迟时间加上 FlashTemp ISR 执行时间不应超过一个 ϕ_1 时钟周期。如果超过了一个 ϕ_1 时钟周期，精度将降低。降低程度与比较器遗漏读取的次数成正比。

当系统识别到 ϕ_1 中断，完成 ISR 的最差时间将接近 100 个 CPU 时钟。简单举例：当收集温度时，如果未发生中断且未禁用中断，则 CPU 时钟与模拟列时钟的关系应如公式 1 所示：

Equation 1

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

当公式 1 两边的 CPU 时钟与模拟列时钟相等时，则 CPU 带宽的占用率为 100%。

放置

带隙温度传感器仅与 PSoC 微控制器内部的一个开关电容模拟模块相连。PSoC Designer™ 只能将 FlashTemp 用户模块放置在带有温度传感器的模块中。

FlashTemp 模拟模块将驱动模拟列比较器。此模拟列的其他模块可能不会驱动此比较器。

参数和资源

Note 将使用全局资源配置此用户模块的参数。

模拟列时钟

用户必须为 FlashTemp 用户模块提供模拟列时钟。模拟列时钟除以 4 得出 ϕ_1 。采样率的计算如公式 2 所示：

Equation 2

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

公式 2 中，255 是积分器电容在每次温度转换时的采样次数，4 是模拟列时钟生成 ϕ_1 的除数。

V_{CC} 电源

在全局资源中，V_{CC} 工作范围可设置为 3.3 V 或 5 V。

模拟功率

在全局资源中，必须将模拟功率调高。

参考复用器设置

在全局资源中，RefMux 必须设置成以下两种形式之一：

(V_{DD}/2) ± 带隙

或

(带隙) ± 带隙

运算放大器偏压

在全局资源中，运算放大器偏压应设置成 LOW 或 HIGH。

应用程序编程接口

应用程序编程接口 (API) 例程作为用户模块的一部分提供，从而使设计人员能够采用更高级的方式处理模块。本节指定每个函数的接口，以及“引用”文件所提供的相关常量。

Note

在这里，如同所有用户模块 API 中的一样，A 和 X 寄存器的值可能由于调用 API 函数发生更改。如果在调用后需要 A 和 X 的值，则调用函数负责在调用前保留 A 和 X 的值。选择此“寄存器易失”策略是为了提高效率，自 PSoC Designer 1.0 版起已强制使用此策略。C 编译器自动遵循此要求。汇编语言程序员也必须确保其代码遵守这一策略。虽然一些用户模块 API 函数可以保留 A 和 X 不变，但是无法保证它们将来也会如此。

对于大型存储器模块驱动，保存 CUR_PP、IDX_PP、MVR_PP 以及 MVW_PP 寄存器中的所有值也是调用程序的职责。尽管部分寄存器现在可能不可修改，但是无法保证在将来的版本中也会如此。

所提供的进入点用于启动数据收集、轮询确认是否完成数据收集、检索结果以及关闭模拟模块。这系列事件的顺序应为：

1. 调用 FlashTemp_Start 以启动数据收集。
2. 使用 FlashTemp_fIsData 确定数据收集的完成时间。在完成数据收集后，API 将禁用中断并关闭 FlashTemp 用户模块。
3. 调用 FlashTemp_cGetData 以获得结果。

FlashTemp_Start

说明:

此 API 用于设置校准数据和温度调整值、开启模拟模块的电源、启用模拟列中断和初始化数据积累变量。它关闭了反馈电容 (Fcap) 的自动复位选项，因此该电容成为 ADC 的积分器；使用待处理的中断数量初始化计数变量，用于温度测量。

此 API 仅提供一个采样处理和读取。待采样读取完毕后，不提供后续处理。为了连续读取采样，您在读取下个采样前必须调用 FlashTemp_Start API。

C 语言原型:

```
void FlashTemp_Start(void)
```

汇编:

```
lcall FlashTemp_Start
```

参数:

无

返回值:

无

副作用:

寄存器 A 和寄存器 X 可以由此函数的本次执行或以后执行而被修改。在大内存模式下 (CY8C29xxx)，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，调用函数负责通过调用 fastcall16 函数保留值。当前，仅修改 CUR_PP、MVW_PP、MVR_PP、IDX_PP 页面指针寄存器。

FlashTemp_fIsData

说明:

检查温度 ADC 进程的状态。如果数据采样未完成，则返回 0。如果数据采样完成，则返回非零值。

C 语言原型:

```
CHAR FlashTemp_fIsData(void)
```

汇编:

```
lcall FlashTemp_fIsData
```

参数:

无

返回值:

如果数据采样完成且结果尚未读取，则返回非零值。如果数据采样未完成或结果已读取，则返回值为 0。

副作用:

寄存器 A 和寄存器 X 可以由此函数的本次执行或以后执行而被修改。在大内存模式下 (CY8C29xxx)，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，调用函数负责通过调用 fastcall16 函数保留值。当前，仅修改 CUR_PP 页面指针寄存器。

FlashTemp_cGetData

说明:

返回 ADC 温度结果 (1° C/ 计数)。此值是 PSoC 微控制器的结温，以二进制补码形式显示。该函数清除内部数据就绪标志，将此数据标志为旧数据。

C 语言原型:

```
CHAR FlashTemp_cGetData(void)
```

汇编:

```
lcall FlashTemp_cGetData
```

参数:

无

返回值:

PSoC 器件的温度。

副作用:

寄存器 A 和寄存器 X 可以由此函数的本次执行或以后执行而被修改。在大内存模式下 (CY8C29xxx)，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，调用函数负责通过调用 fastcall16 函数保留值。当前，仅修改 CUR_PP 页面指针寄存器。

FlashTemp_Stop

说明:

关闭 PSoC 用户模块并禁用中断。

C 语言原型:

```
void FlashTemp_Stop(void)
```

汇编:

```
lcall FlashTemp_Stop
```

参数:

无

返回值:

无

副作用:

寄存器 A 和寄存器 X 可以由此函数的本次执行或以后执行而被修改。在大内存模式下 (CY8C29xxx)，所有 RAM 页面指针寄存器也会出现这种状况。如果需要，调用函数负责通过调用 fastcall16 函数保留值。

固件源代码示例

下列汇编代码示例将启动 FlashTemp 用户模块、收集数据并将其存入变量：

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; 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 "PSoC_API.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

```

C 语言编写的相同程序是：

```

//*****
//

```

```
// 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 "PSoC_API.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();
    }
}
}
```

配置寄存器

TEMPERATURE 是含有带隙温度传感器的开关电容模块。此模块将向 ACap 和 BCap 输入数据，并将 FCap 用作可复位的积分器。

Table 2. TEMPERATURE 模块：寄存器 CR0

位	7	6	5	4	3	2	1	0
值	1	0	0	1	0	0	0	0

Table 3. TEMPERATURE 模块：寄存器 CR1

位	7	6	5	4	3	2	1	0
值	0	1	1	1	0	0	0	0

Table 4. TEMPERATURE 模块: 寄存器 CR2

位	7	6	5	4	3	2	1	0
值	0	1	1	0	0	0	0	0

Table 5. TEMPERATURE 模块: 寄存器 CR3

位	7	6	5	4	3	2	1	0
值	1	1	1	1	1	1	功耗	

功耗用于设定模拟模块的功率级别。

Table 6. 寄存器 INT_MSK0

位	7	6	5	4	3	2	1	0
值						Acolumn1		

Acolumn1 支持从模拟列 1 启用中断。这是要设置在₁上升沿触发。

Table 7. 寄存器 CMP_CR

位	7	6	5	4	3	2	1	0
值			COMP1					

COMP1 表示模拟列 1 的模拟比较器总线状态。

版本历史记录

版本	创作者	说明
2. 2	DHA	添加了 “ 版本历史 ”
2. 30	DHA	1. 添加了 SSCParamBlk RAM 区域，供内部用户模块使用。 2. 补充了对 `@INSTANCE_NAME`_TABLE_1. 的定义
2. 30. b	DHA	更新了 “ 参数和资源 ” 章节，告知用户 “ 全局资源将用于配置 FlashTemp 用户模块的参数 ”。

Note PSoC Designer 5.1 在所有用户模块数据手册中都引入了 “ 版本历史 ”。本数据表详细介绍了当前和先前用户模块版本之间的区别。