应用手册 AN2283

频率测量

作者：Dave Van Ess
相关项目：是
相关器件系列：任何
PSoC Designer Version：4.2
相关应用手册：AN2099

摘要
许多应用都要求进行频率测量。目前，测量频率的方法有几种，每种方法都有各自的优势。本文简要讨论了其中的两种方法并介绍了一种混合方法。另外列举了一个误差为0.0016%（16ppm）的典型频率测量示例。

前言

测量信号频率是一种常用的混合信号的应用。此类信号可以是来自马达的测速信号，也可以是用于音频检测的模拟信号。无论何种情况都需要测量信号的振荡频率。对于机械系统而言，称为速率，通常以“每分钟转数”（rpm）表示。而对于电气系统，即为频率，众所周知用“每秒周期数”或赫兹表示。

测量频率的两种传统方法：

- 测量固定时间段内的周期数。
- 测量单个周期所用的时间。

在测量范围和精度方面，每种方法都具有各自的优势和局限性，下面，我们将逐一介绍。

计算固定时间内的周期数

图1显示了计算固定时间内周期数的拓扑结构。

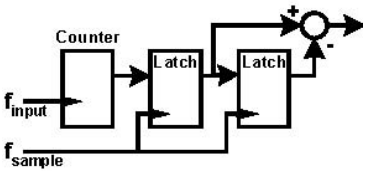


图1．计算固定时间内的周期数

计数器的值锁定在采样频率f_{sample}，并且随每个周期的输入信号f_{input}增加而增加。双重锁定功能可以从新计数值中减去旧计数值，从而获得每个采样周期的新累加值。

这种频率测量方法的优势是可以在固定的采样频率f_{sample}下得出答案。给定输入频率和采样频率，根据公式（1）可计算出累加值。

$$AccumulatedValue = \frac{f_{input}}{f_{sample}} \quad \textbf{(1)}$$

根据定义，累加值必须是整数，而两个频率的比值很有可能不是整数。因此，在这样的情况下，累加值会存在1个计数的差值，因此累加值可以取大量累加值的平均值。假设采样频率为每秒1个采样（sps），输入频率为12.3Hz，那么累加值即为：

- 12，70%的时间。
- 13，30%的时间。

如果采样频率提高到10sps，则累加值为：

- 1，67%的时间。
- 2，23%的时间。

将上述量化误差（eq）考虑进去，公式（1）经重新定义变为公式（2）。

$$AccumulatedValue = \frac{f_{input}}{f_{sample}} + e_q \quad \{|e_q| < 1\} \qquad (2)$$

公式（3）表达如何计算测量频率。

$$f_{measured} = AccumulatedValue \cdot f_{sample} \qquad (3)$$

将公式（2）定义的累加值代入公式（3）可得到公式（4）。

$$f_{measured} = f_{input} + e_q \cdot f_{sample} \qquad (4)$$

如果输入频率为12.3Hz，采样速率为1sps，则测量频率为：

- 12Hz，70%的时间。
- 13Hz，30%的时间。

如果采样频率提高到10sps，则累加值为：

- 1Hz，67%的时间。
- 2Hz，23%的时间。

速率为1sps时，最坏情况误差约为1Hz；速率为10sps时，最坏情况误差约为10Hz。1Hz的误差对测量较高频率时影响不大，比如说141421Hz，计算出的误差为百万分之7（ppm）。但是，对于测量较低频率，比如说14Hz，这种误差就很惊人。

假设采样频率和输入频率已知，根据公式（5）可计算出相对误差。

$$error_{rel} = \frac{|f_{measured} - f_{input}|}{f_{input}} = |e_q| \cdot \frac{f_{sample}}{f_{input}} < \frac{f_{sample}}{f_{input}} \qquad (5)$$

显然，输入信号频率越高，则相对于采样频率而言，误差就越低。为了保证误差小于0.1%，那么输入信号频率必须至少是采样频率的1000倍。对于1Hz的采样频率，可计算出最低输入频率为1kHz。

可测量的输入频率的上限取决于计数器大小。

公式（6）将频率上限定义为采样速率和计数器大小（单位是位）的函数。

$$f_{input} < f_{sample} \cdot 2^n \qquad (6)$$

试图测量高于上限的频率会造成计数器溢出。

表1对这种测量频率的方法做了总结。它最适合测量较高的频率并能在稳定、可预测的速率下得出答案。

表1. 测量较高频率的方法

采样频率	fsample
最低输入频率	0Hz
最高输入频率	fsample * 2^n -1
相对误差	< fsample/ finput

测量单个周期所用的时间

图2显示了测量单个输入周期所用时间的拓扑结构。

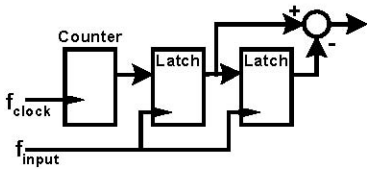


图2. 测量单个周期所用的时间

计数器的值随时钟频率*fclock* 增加而持续递增，同时被输入信号*finpu* 锁定。双重锁定功能可以从新计数值中减去旧计数值，从而获得每个采样周期的新累加值。

此方法的优势是可以按输入频率*finput*的速率进行更新。给定输入频率和采样频率，根据公式（7）可计算出累加值。

$$AccumulatedValue = \frac{f_{clock}}{f_{input}} + e_q \qquad (7)$$

请注意，为了说明量化过程，已经在公式中添加了一个误差项。假设输入信号频率为3kHz，时钟频率为1MHz，在忽略误差的情况下，那么利用公式（7）求出的累加值即为333.33。实际累加计数为：

- 333，三分之二的时间。
- 334，三分之一的时间。

公式（8）表达如何计算测量频率。

$$f_{measured} = \frac{f_{clock}}{AccumulatedValue} \quad (8)$$

给定与上述相同的输入与时钟频率，则测量频率为：

- 3.003kHz，三分之二的时间；
- 2.994kHz，三分之一的时间。

将公式（7）中定义的累加值代入公式（8），则可得到公式（9）。

$$f_{measured} = \frac{f_{clock}}{\left(\frac{f_{clock}}{f_{input}} + e_q\right)} \approx f_{input}\left(1 - e_q \cdot \frac{f_{input}}{f_{clock}}\right) \quad (9)$$

公式（10）定义了给定时钟和输入频率的相对误差。

$$error_{rel} = \frac{|f_{measured} - f_{input}|}{f_{input}} = |e_q| \cdot \frac{f_{input}}{f_{clock}} < \frac{f_{input}}{f_{clock}} \quad (10)$$

为保证误差小于0.1%，时钟频率必须至少是输入频率的1000倍。假设时钟频率为24MHz，则可求出输入频率最高为24kHz。

可测量的输入频率的下限由计数器的大小决定。公式（11）将输入频率下限定义为时钟速率和计数器大小（单位为位）的函数。

$$f_{input} < \frac{f_{clock}}{2^n - 1} \quad (11)$$

试图测量低于下限的频率会造成计数器溢出。

表2对这种测量频率的方法做了总结。它最适合测量较低的频率。

表2. 较低频率的测量方法

采样速率	f_{input}
最低f_{signal}	$f_{clock}/(2^n-1)$
最高f_{signal}	f_{clock}
相对误差	$< f_{input}/f_{clock}$

混合方法

这两种测量频率方法都具有各自的优势。计算周期数最适合测量较高的频率，而测量周期时间更适合测量较低的频率。但当无论精度多么高（即相对误差较小），对于任何特定的时钟或采样频率而言，输入频率范围是有限的，在这种情况下，上述两种频率测量方法都存在局限性。

传统的解决方案是采用不同的范围。也就是说采用多个可选择的时钟或采样频率。这需要很高的推理能力才能确定改变范围或测量方法的最佳时机。

测量多个周期所用的时间可采用混合方法。图3显示此类测量的拓扑结构。

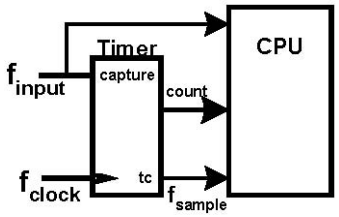


图3. 测量多个周期所用的时间

定时器是一个随每个时钟周期f_{clock}递减的*n*位递减计数器。计数器每次下溢时产生一个终端计数。信号f_{sample} 确定采样周期。它是时钟频率和计数器大小的函数。CPU采用此信号使第一个和最后一个采集的周期同步。输入f_{input}连接到定时器采集输入。采集的上升沿可以锁定计数器的值，同时接入CPU。输入信号也连接到CPU，以便对其计数。

同样，此方法需要测量多个周期所用的时间。公式（12）描述了采样频率、采样速率和计数器大小（单位是位）之间的函数关系。

$$f_{sample} = \frac{f_{clock}}{2^n} \quad \textbf{(12)}$$

对于时钟频率为100kHz的16位定时器，采样速率为1.52588sps。

采样周期内检测到的输入周期数是采样频率和输入频率的函数。公式（13）对此作了定义。

$$CycleCount = \frac{f_{input}}{f_{sample}} + e_{q1} \quad \textbf{(13)}$$

在忽略量化误差的情况下，假如输入信号的频率为15kHz，周期计数为9830.4。那么实际周期计数为：

- 9830，60%的时间。
- 9831，40%的时间。

实际周期计数会发生变化。请注意：如果周期计数为“c”，则测量的周期为“c-1”。定时器累加值是数据时钟、输入频率和周期计数值的函数。公式（14）对此作了定义。

$$AccumulatedValue = \frac{f_{clock}}{f_{input} / CycleCount - 1} + e_{q2} \quad \textbf{(14)}$$

在参数相同并且忽略新量化误差的情况下，累加值为：

- 65526.67，当周期计数为9830。
- 65533.33，当周期计数为9831。

公式（15）定义测量的频率是如何计算的。它是数据时钟、测量的周期数和那些周期累加值的函数。

$$f_{measured} = \frac{f_{clock} \cdot (CycleCount - 1)}{AccumulatedValue} \quad \textbf{(15)}$$

同样，在参数相同时，表3给出了周期计数和累加值的所有4种排列的测量频率。

表3．所有4个数列的频率

$F_{measured}$ (kHz)	Cycle Count	Accumulated Value
15.00015	9830	65526
14.99992	9830	65527
15.00008	9831	65533
14.99985	9831	65534

对于所有四种可能，相对误差为10ppm时，最坏情况误差是0.15Hz。

这种频率测量方法的相对误差很难精确地算出，不过，根据累加值、利用公式（16）的定义可大致求出。

$$error_{rel} = \frac{\left| f_{measured} - f_{input} \right|}{f_{input}} < \frac{1}{AccumulatedValue} \quad \textbf{(16)}$$

最坏情况误差来自输入信号，其造成最小的累加值。对于达不到采样速率3倍的输入信号，则周期计数低于3。它在2和3之间变动，造成1个或2个测量周期。对于1个周期的情况，累加值约为计数器范围总值的三分之一。假设采样频率计数器大小（单位是位）和输入频率与采样频率的比值已知，根据公式（17）大致可以确定最坏情况误差。

$$error_{rel} = \frac{\left(\frac{ratio}{ratio - 2} \right)}{2^n} \quad ;ratio \geq 3 \quad \textbf{(17)}$$

当比值（ratio）为3时误差最坏。对于16位定时器，误差是3/65536或46ppm。随着比值的增加，误差大幅降低到下限1/65536或15ppm。

图4说明对于16位定时器的误差与输入频率和采样频率的比值之间的模拟关系曲线。

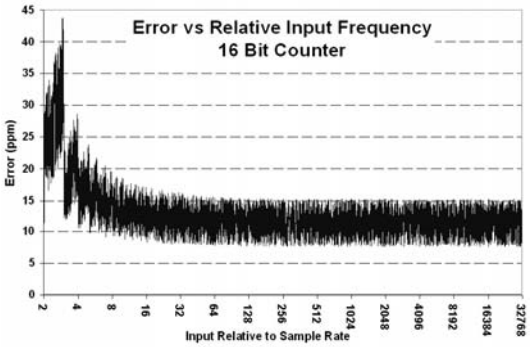


图4．16位定时器的相对误差

请注意：当信号是采样速率的3倍时误差曲线达到峰值，对于达到采样速率16倍以上的值，误差曲线保持相对平坦。

表4给出了三种不同大小的定时器的最坏情况误差和极限误差。

表4．最坏情况和极限相对误差

定时器大小（位）	最坏情况误差	极限误差
8	1.2%	0.4%
16	46 ppm	15 ppm
24	0.18 ppm	0.06 ppm

可以连续测量的最低频率是采样速率的2倍。低于2倍的值无法保证能检测到2个周期。最高频率是时钟频率。

范围是最高值与最低值的比值，公式（18）对此作了定义。

$$range = \frac{\max(f_{input})}{\min(f_{input})} = \frac{f_{clock}}{2 \cdot f_{sample}} = \frac{2^n \cdot f_{sample}}{2 \cdot f_{sample}} = 2^{n-1} \quad \textbf{(18)}$$

表5总结了测量频率的混合方法。这种测量方法适用于较宽的频率范围而且在稳定、可预测的频率下得出答案。

表5．测量频率的混合的方法

采样频率	$f_{clock}/2^n$
最低f_{signal}	$f_{clock}/2^{n-1}$
最高f_{signal}	f_{clock}
大小	2^{n-1}
相对误差	< $3/2^n$下至$1/2^n$

参数选择

表5的5个参数仅来自两个变量。

- 时钟频率
- 定时器大小

每种具体的应用会强调其中的一个或多个参数。因此需要选择这两个变量。

假设某种应用需要根据以下要求控制4极风扇的转速：

- 需要测量1000～8000rpm的转速；
- 精度高于50rpm；
- 采样频率必须准确达到10sps。

第三个条件要求f_{sample} = 10sps。

转速在1000～8000rpm的4极风扇可以产生频率为66.66Hz～533.33Hz的脉冲。如果采样频率为10sps，则最低输入频率为20Hz。这远远低于所要求的66.66Hz的下限。

要求的范围仅为8，因此看来8位定时器就够用。据此，设置所需的时钟频率f_{clock} = 2.56kHz。另外，设置测量的上限为2.56kHz。这远远高于要求的543.33Hz。

唯一不确定的是精度要求。利用公式（17）可以确定相对误差。相对误差乘以输入频率可以求出实际误差。图5显示了误差与输入频率的关系曲线图。

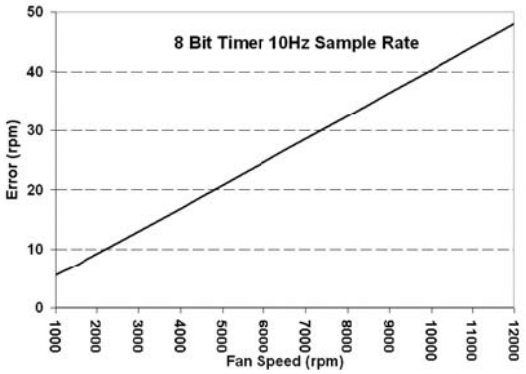


图5．误差与输入频率的关系曲线

对于1000～8000rpm的频率范围而言，误差远远低于50rpm的限值。8位定时器即可满足该特定的应用需求。两个参数为：

- f_{clock} = 2.56kHz
- n = 8（8位定时器）

PSoC实施

本例采用采样频率为1sps的通用16位定时器，在CY8C29466-24XI（28引脚DIP CY8C29xxx系列器件）中实施。之所以选择该PSoC™器件是因为它已经包含在CY3210-PSoCEval1之中。本项目可以容易地克隆到任何PSoC器件系列中。实现图3所示拓扑所需要的模块包括：

- 调节输入信号和提供“采集”中断的比较器；
- 提供终端计数中断的16位定时器；
- LCD；
- 控制器软件。

示例硬件

下面介绍每个硬件模块的布局与参数选择。

比较器用户模块

采用可编程阈值比较器用户模块来实现比较器的功能。图6与图7说明其布局与参数选择。

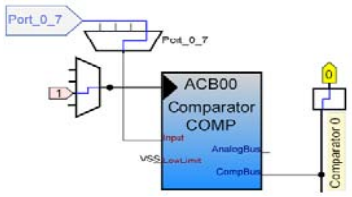


图6．比较器用户模块布局

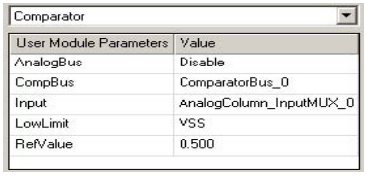


图7．比较器用户模块参数

选择Port 0[7]作为输入。阈值设置为电源电压的50%，不过可以轻松变为其他阈值电压。比较器输出连接到Comparator Bus 0，通过这里提供给数字模块、CPU和中断服务器。

定时器用户模块

采用Timer16用户模块来实现定时器的功能。图8和图9说明其布局与参数选择。

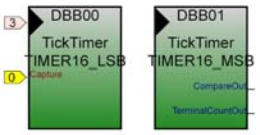


图8．定时器用户模块布局

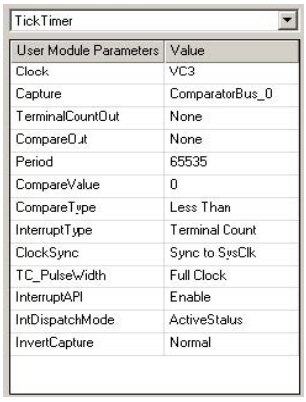


图9．定时器用户模块参数

定时器设置为每隔2^{16}个时钟周期产生一个中断信号的16位定时器。时钟连接到VC3上，针对65.21739kHz的时钟频率设置为24MHz/368。采样频率等于时钟频率除以周期值或0.995sps。采集输入连接到比较器，从而可以锁定定时器值并且供CPU使用。

LCD

设置LCD用户模块使用2号端口。

采样信号源

增加了Counter24用户模块，用于产生采样频率。图10与图11说明其布局与参数选择。

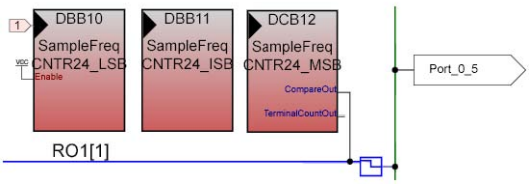


图10．计数器用户模块布局

SampleFreq

User Module Parameters	Value
Clock	VC1
Enable	High
CompareOut	Row_1_Output_1
TerminalCountOut	None
Period	9719
CompareValue	4860
CompareType	Less Than
InterruptType	Terminal Count
ClockSync	Sync to SysClk
InterruptAPI	Disable
IntDispatchMode	ActiveStatus
InvertEnable	Normal

图11．采样频率源（计数器）用户模块参数

VC1是计数器的时钟，设置为12MHz。当前周期设置可以产生1.2346kHz的输出频率。输出连接到Port 0[5]。通过改变周期和对比值可以在6MHz～0.72Hz之间对采样频率进行调节。

其他要考虑的问题

Port 1[7]连接到以采样频率闪动的LED上。图12说明本例的外引脚。

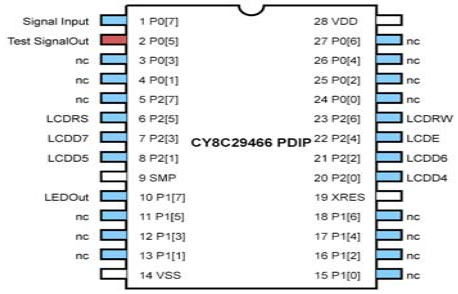


图12．示例项目的外引脚

示例软件

频率的计算需要满足以下要求：

- 保存定时器周期内第一次采集到定时器值；
- 保存定时器周期内最后一次采集到的定时器值；
- 保存定时器周期内检测到的周期总数；
- 根据前3个值处理测量的频率。

保存第一次和最后一次采集到的定时器值需要2个16位变量，即：

- `wFirstValue`
- `wLastValue`

计算检测到的周期数需要一个16位变量，保存此变量的溢出需要一个8位变量，即：

- `wTickCount`
- `cOverFlow`

它们可以视为一个单独的24位变量。

在定时器周期结束时，两个定时器值的差值必须与周期计数值副本一起存储。由于此过程在后台运行，因此必须设置一个信号量，用于宣布收到新数据。这些变量包括：

- `wSaveTickNum`
- `cSaveOverFlow`
- `wSaveCountNum`
- `bDataAvailable`

总共需要8个采用13个字节的RAM的变量。

用于频率测量的软件包含三个部分：

- 初始化
- 周期采集
- 定时器周期结束时的处理

初始化

代码1说明初始化功能。它称为StartFreq，位于本应用手册相关项目的*freqcode.asm*中。

```
export _StartFreq
include "m8c.inc"
include "memory.inc"
include "PSoCAPI.inc"
_StartFreq:
   RAM_PROLOGUE RAM_USE_CLASS_4
   RAM_SETPAGE_CUR >_bDataReady
   mov [_cOverFlow],ffh      ;TickCount =-1
   mov [_wTickCount + 0],ffh
   mov [_wTickCount + 1],ffh
   mov [_bDataAvailable],0   ;clear bDataReady
   and reg[INT_CLR1],~02h    ;clear timer int
   TickTimer_EnableInt_M     ;enable timer
   TickTimer_Start_M        ;start timer
   and reg[INT_CLR0],~02h    ;clear comp int
   or  reg[INT_MSK0],02h     ;enable comp int
   RAM_EPILOGUE RAM_USE_CLASS_4
   ret
```

代码1．初始化功能

此例程初始化周期计数变量。请注意：初始化值为-1。自动从计数值减去1。启动定时器，同时启用两个中断。

周期采集

代码2说明周期采集功能。它称为TickHandler_ISR，位于本应用手册相关项目的*tickhandler.asm*中。

```
include "m8c.inc"
include "memory.inc"
include "TickTimer.inc"

export  _TickHandler_ISR
_TickHandler_ISR:
   push A
   inc  [_wTickCount + 1]
   adc  [_wTickCount + 0],0
   adc  [_cOverFlow],0
   jnc  LastTick
FirstTick:
   mov  A, reg[TickTimer_COMPARE_LSB_REG]
   mov  [_wFirstValue + 1],A
   mov  A, reg[TickTimer_COMPARE_MSB_REG]
   mov  [_wFirstValue + 0],A
   pop  A
   reti
LastTick:
   mov  A, reg[TickTimer_COMPARE_LSB_REG]
   mov  [_wLastValue + 1],A
   mov  A, reg[TickTimer_COMPARE_MSB_REG]
   mov  [_wLastValue + 0],A
   pop  A
   reti
```

代码2．周期采集功能

计数值递增。如果计数器清零，则必须是检测到的第一个周期。采集到的定时器值存储在wFirstTick。其他情况下的定时器值全部加载到wLastTick。

定时器周期结束时的处理

代码3说明此功能。它位于*TickTimerint.asm*的定时器中断处理程序的保留区域。

```
    ;-------------------------------------------
   ; Insert your custom code below this banner
   ;-------------------------------------------
    push A
    mov  [_wSaveCountNum+1],[_wFirstValue+1]
    mov  [_wSaveCountNum+0],[_wFirstValue+0]
    mov  A,[_wLastValue +1]
    sub  [_wSaveCountNum+1],A
    mov  A,[_wLastValue +0]
    sbb  [_wSaveCountNum+0],A

    mov  [_cSaveOverFlow]   ,[_cOverFlow]
    mov  [_wSaveTickNum + 0 ],[_wTickCount + 0]
    mov  [_wSaveTickNum + 1 ],[_wTickCount + 1]

    mov  A,ffh
    mov  [_cOverFlow],A
    mov  [_wTickCount +  0],A
    mov  [_wTickCount +  1],A
    mov  [_bDataAvailable] ,A
    pop  A
   ;-------------------------------------------
   ; Insert your custom code above this banner
   ;-------------------------------------------
```

代码3．周期结束时的数据处理

此例程从第一个定时器中减去第二个计数器，以便确定时间值并且由主程序把它存储到可以存取的变量中。在重新初始化到-1之前，周期计数也被复制到该变量上。最后设置信号量。

这些功能在后台运行，产生计数值和时间值。它们在主程序中转换成频率。

主程序

代码4说明主程序，位于本应用手册相关项目的*main.c*中。

```
#define SYSCLOCK    24000000.0
#define FCLOCK     (SYSCLOCK / 368.0)
#define PERIOD      65536.0
#define FSAMPLE    (FCLOCK / PERIOD)

void main()
{
  Comparator_Start(Comparator_HIGHPOWER);
  SampleFreq_Start();
  PRT0DR = 0x80; //Enable Input Pullup P0.7
  PRT1DR = 0x0c;//Set P1.2 & P1.3 High
  LCD_Start();
  M8C_EnableGInt;
  StartFreq();  //Start TickTimer, Initialize
  while(1){
    bDataAvailable = 0x00;
    while(bDataAvailable == 0x00);
    PRT1DR |= 0x80; //LED On
    If(cSaveOverFlow < 0 )fFreqValue = 0.0;
    else if(cSaveOverFlow > 0){
      fFreqValue  =(float)(cSaveOverFlow);
      fFreqValue *= 65536.0;
      fFreqValue +=(float) wSaveTickNum;
      fFreqValue += 1.0;
      fFreqValue /= FSAMPLE;
    }
    else{//No OverFlow
      if( wSaveTickNum ==0)fFreqValue = 0.0;
      else{
        fFreqValue  =(float) wSaveTickNum;
        fFreqValue *= FCLOCK;
        fFreqValue /=(float)wSaveCountNum;
      }
    }
    DisplayValue();
    PRT1DR &= ~0x80; //LED Off
  }
}
```

代码4．主程序

程序从系统定义开始。然后初始化相关变量，启动用户模块，启用中断，同时启动后台频率测量进程。然后进入执行以下任务的连续循环：

- 清除bDataAvailable标记；
- 等待再次设置该标记；
- 打开LED；
- 计算测量的频率；
- 显示此值以及周期计数与定时器值；
- 关闭LED。

在查看上述代码之后即可明白6个步骤的4个，因此无需赘述。LCD显示是单独的功能。它用于把计算的频率值写入LCD。如果读者感兴趣，可以在与本应用手册相关的项目中查阅此功能的代码。此处不再讨论。然后是代码频率计算。

4种不同的情况下有不同的计算方法：

-
- cSaveOverFlow < 0
- cSaveOverFlow = 0. wSaveTickNum = 0
- cSaveOverFlow = 0. wSaveTickNum >0
- cSaveOverFlow > 0

cSaveOverFLow < 0

如果定时器周期内未做任何采集，则造成此情况的发生。频率必须低于采样频率，因此计算的频率设置为0。

cSaveOverFLow = 0. wSaveTickNum = 0

如果定时器周期内仅采集一次，则造成此情况的发生。频率必须低于最低的容许输入频率采样频率，因此计算的频率再次设置为0。

cSaveOverFLow = 0. wSaveTickNum != 0

如果输入位于要求的范围内，则造成此情况的发生。采用公式（15）来计算频率。

cSaveOverFLow > 0

在输入频率超过最高容许的输入频率时，则造成此情况的发生。此时的问题是测量固定时间的计数。采用公式（3）来计算频率。

不同测量方法的转换可以提高范围。根据采集输入信号所需要的时间确定新的最高输入频率。输入信号的周期必须大于处理中断所需要的时间。

该中断需要91个CPU周期。对于24MHz的CPU时钟，要求最高输入频率不超过260kHz。对于12MHz的CPU时钟，最高输入频率降低到130kHz。对于频率更高的输入，需要采用其他的解决方案。一种方法是重新配置数字块来对固定时间的周期进行计数。另一种方法是在输入前面放置一个8位计数器，用于分割输入频率（预计算器）。

图13显示在PSoC评估板上执行此项目。

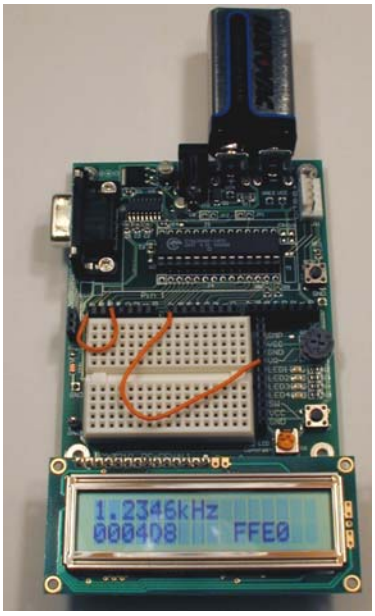


图13. 工作频率计

请注意：下面一行显示24位周期计数值和16位定时器值。为了验算，通过求解测量的频率、计数值和定时器值来计算系统时钟。对于图13显示的值，求出的频率为24000497Hz。

约16ppm的误差（求出或获得25000ppm）

我编写本节的原因是：因为我知道，如果我不这么做，某些社交能力低下（可能低于一般的工程师）、不近人情的工程师有可能发电子邮件或打电话提出置疑，比如：

> *""Dave，你好！*
> *我创建了**你**声称典型误差为16ppm的频率计。*
> *于是，我**准确无误地**输入了10kHz，但是得到*
> *的结果是10.127kHz！"*

显然，这种直截了当的说法意味着这人太聪明，很难受到像我这类人的愚弄！

这种差异是被测的输入频率与内部系统时钟（24MHz）产生的时钟频率相比较的结果。这种系统时钟的最坏情况误差为2.5%。本项目的开发采用了内部系统时钟，因为评估板未配备晶振组件。

以下步骤说明如何采用晶振控制的时钟修改上述示例。

必须按照以下步骤修改评估板：

- 拆除**R8**与**R9**；
- 在位置**Y1**安装一个**32.768kHz时钟晶振**；
- 在**C10**位置安装一个**12pF**电容器，在**C9**位置安装一个**100pF**电容器。

必须修改以下PSoC参数：

- **P1[0]** 驱动器必须设置为**XtalOut**，**P1[1]**驱动器必须设置为**XtalIn**；
- 全局资源**32K_Select**必须设为**External**；
- 全局资源**PLL_Mode**必须设为**Ext Lock**。

现在系统时钟由晶振控制。它是晶振频率的732倍，或23.986MHz，而并非标准的24MHz。需要把*main.c*中的SYSCLOCK定义修改为23986176.0。

对于1.2345kHz的目标测试频率，上述新系统时钟频率需要把采样信号源周期修改为9715。

有关所有上述项目修改，参见与本应用手册相关的项目文件夹：…\FrequencyProjectPLL。

图14显示该晶振控制的频率计。

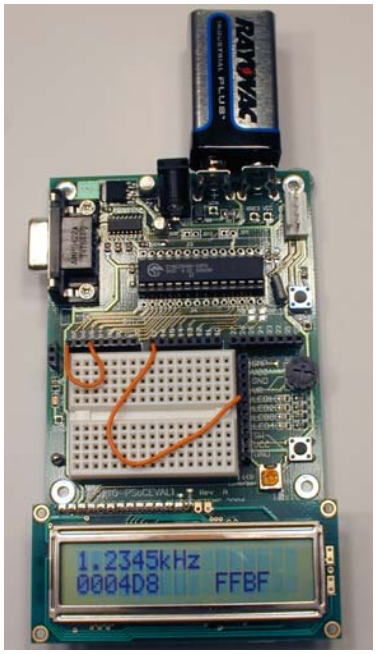


图14. 晶振控制的频率计

您可能注意到已经拆掉了2个电阻器。另外已增加了2个电容器和1个晶振。

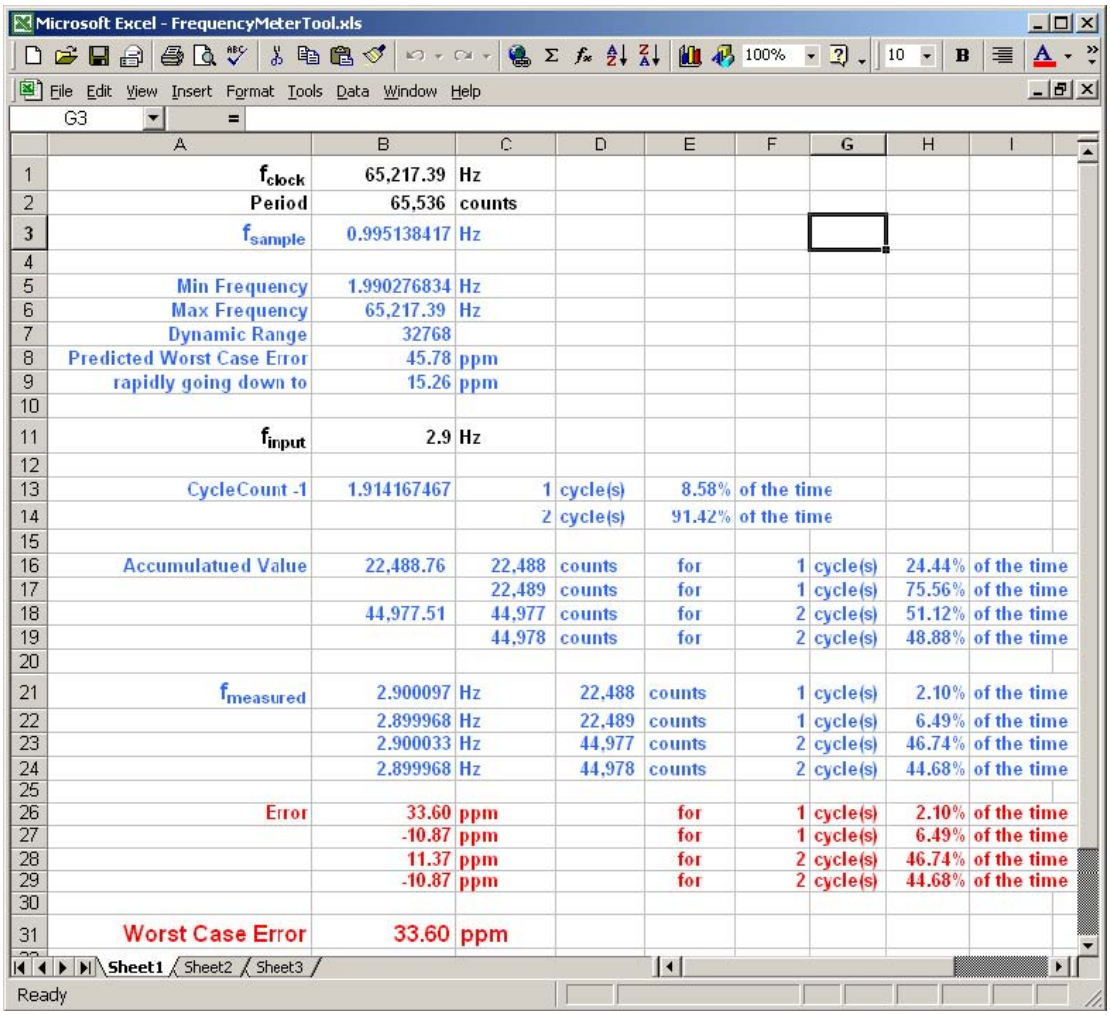
通过求解测量的频率、计数值和定时器值可以计算系统时钟。对于图14显示的值，可求出23986462Hz。

建立属于你的

本示例是针对特定规格而开发的。您的项目可能具有不同的要求。我们提供一份数据手册，以便用户选择时钟频率和定时器大小。结果包括采样频率、最低和最高容许的输入频率。用户还可以输入频率，然后确定测量的相对误差。本应用手册相关的项目文件夹是*FrequencyMeterTool.xls*，如图15所示。

结论

用混合方法测量频率，适用于广泛的频率范围，可以检测到超出上下范围的频率值，并且可以按照预测的频率输出答案。它可以有效地利用PSoC器件资源，根据精度和定时器采集中断产生方法，需要1个、2个或3个数字块。



	A	B	C	D	E	F	G	H	I
1	f_{clock}	65,217.39	Hz						
2	Period	65,536	counts						
3	f_{sample}	0.995138417	Hz						
4									
5	Min Frequency	1.990276834	Hz						
6	Max Frequency	65,217.39	Hz						
7	Dynamic Range	32768							
8	Predicted Worst Case Error	45.78	ppm						
9	rapidly going down to	15.26	ppm						
10									
11	f_{input}	2.9	Hz						
12									
13	CycleCount -1	1.914167467	1	cycle(s)	8.58%	of the time			
14			2	cycle(s)	91.42%	of the time			
15									
16	Accumulatued Value	22,488.76	22,488	counts	for	1	cycle(s)	24.44%	of the time
17			22,489	counts	for	1	cycle(s)	75.56%	of the time
18		44,977.51	44,977	counts	for	2	cycle(s)	51.12%	of the time
19			44,978	counts	for	2	cycle(s)	48.88%	of the time
20									
21	$f_{measured}$	2.900097	Hz	22,488	counts	1	cycle(s)	2.10%	of the time
22		2.899968	Hz	22,489	counts	1	cycle(s)	6.49%	of the time
23		2.900033	Hz	44,977	counts	2	cycle(s)	46.74%	of the time
24		2.899968	Hz	44,978	counts	2	cycle(s)	44.68%	of the time
25									
26	Error	33.60	ppm		for	1	cycle(s)	2.10%	of the time
27		-10.87	ppm		for	1	cycle(s)	6.49%	of the time
28		11.37	ppm		for	2	cycle(s)	46.74%	of the time
29		-10.87	ppm		for	2	cycle(s)	44.68%	of the time
30									
31	Worst Case Error	33.60	ppm						

图15. *FrequencyMeterTool.xls*

作者简介

姓名： Dave Van Ess

职称： 赛普拉斯半导体公司首席应用工程师

背景： 他是内布拉斯加州一名训练有素的工程师、性情诗人和桀骜不逊的男人。Dave的抽象思维能力强，善于分析具体问题和实施严格。他荣获了加州大学伯克利分校电子工程学士学位(BSEE)。在电路、信号处理、数字、软件、模拟和系统设计领域拥有27年以上经验。拥有6项美国医疗系统、信号处理和数字模块专利（另外还有3项专利正在申请）。他编写了许多应用手册、网播和技术文章。

他于2000年就职赛普拉斯微系统公司。

联系方法： dwv@cypress.com

赛普拉斯微系统公司
地址：林恩伍德市D大厦西南街162号2700
邮编：98037
电话：800.669.0557
传真：425.787.4641

http://www.cypress.com/