AN007 — 用 PCA 实现 16 位 PWM

相关器件

本应用笔记适用于下列器件：

C8051F000、C8051F001、C8051F002、C8051F005、C8051F006、C8051F010、C8051F011和C8051F012。

引言

脉冲宽度调制（PWM）波形常用于闭环反馈和控制应用，例如：控制加热单元的开关状态以调节DWDM（波分复用）系统中激光器的温度。在某些应用中，可编程计数器阵列（PCA）的内建8位PWM方式不能提供任务所需的足够的分辨率。本应用笔记介绍如何用PCA的‘高速输出’方式和最小的软件开销来产生一个16位分辨率的PWM波形。

背景

图1给出了PWM波形的一个例子。用于闭环控制应用的PWM信号的频率并不重要，只要波形‘足够快’就可以了，象控制系统的阶跃响应时间就应远远小于PWM信号的周期。信号所携带的的信息用波形的占空度来编码，占空度是波形为高电平的时间与PWM信号的周期之比。对于一个PWM实现来说，其输入是一个数值，通常为整数，该数值与所要求的输出波形的占空度成正比。

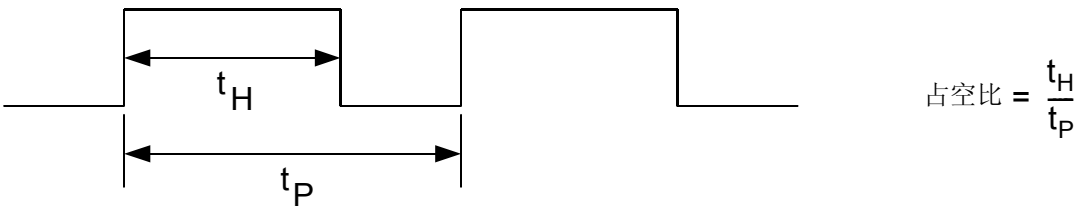

图1. PWM波形示例

实现

在一个基于8051的设计中，有很多方法用于产生PWM波形：软件循环、查询或中断驱动的定时器等等。本应用笔记中的例子使用可编程计数器阵列（PCA）。相对于任何查询机制（基于软件或定时器）而言，使用PCA产生PWM可以大大降低所需要的CPU带宽，并可以消除在中断驱动的基于定时器的设计中因中断延迟不一致而产生的时序抖动。

Silicon Laboratories Inc.
4635 Boston Lane
Austin, TX 78735
Email: mcuinfo@silabs.com
Internet: www.silabs.com

新华龙电子有限公司
深圳市福田区华强北路现代之窗大厦 A 座 13F C 室(518013)
电话：0755-83645240 83645242 83645244 83645251
电邮：shenzhen@xhl.com.cn
网址：www.xhl.com.cn

PCA 简介

PCA包含一个16位的计数器/定时器和5个捕捉/比较模块，如图2所示。计数器/定时器有一个16位的计数器/定时器寄存器（PCA0H:PCA0L）、一个用于选择时间基准的方式寄存器（PCA0MD）和一个包含计数器/定时器运行控制及各模块捕捉/比较标志的控制寄存器（PCA0CN）。每个捕捉/比较模块有一个用于选择模块工作方式（边沿触发捕捉、软件定时器、高速输出或PWM）的配置寄存器（PCA0CPMx）和一个16位的捕捉/比较寄存器（PCA0CPHn:PCA0CPLn）。

由于所有的捕捉/比较模块共享一个时间基准，因此它们同步工作，例如在电机控制应用中可以提供锁相激励波形。另外，由于每个模块有其自己的控制和捕捉/比较寄存器，因此每一个模块工作上又独立于其它模块，只要任何一个模块的服务程序都不影响共享的时间基准（停止或重新设置计数器/定时器、改变计数器/定时器的时钟源）。

本应用笔记中的例子将PCA配置为独立工作；每个模块的服务程序只影响该模块的配置寄存器和捕捉/比较寄存器。PCA方式寄存器只设置一次，不再改变，让计数器/定时器寄存器（PCA0H:PCA0L）自由运行。

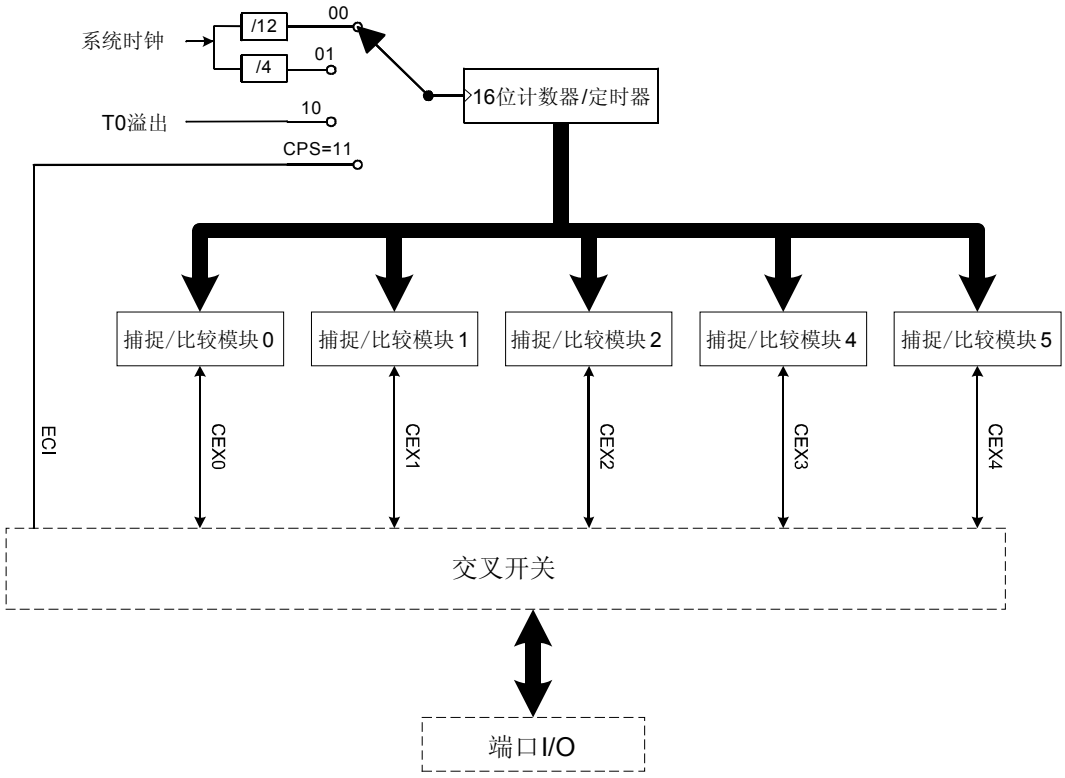

图2. PCA原理框图

选择 PCA 的时基信号

PCA的时基信号可以是下述四个时钟源之一：SYSCLK/12、SYSCLK/4、定时器0溢出、或出现在一个外部引脚ECI上的下降沿。图3给出了PCA计数器/定时器的方框图。

在下面的几节中我们将看到，PCA时基信号的选择决定PWM波形的频率。如前所述，PWM波形的频率通常并不重要，只要‘足够快’就可以了。

一个不很明显的时序选择是：PCA可以按SYSCLK的时钟频率工作。这可以通过选择定时器0溢出作为PCA时钟源，将定时器0设置为8位自动重装载方式并设重载值为‘0xFF’来实现。

本应用笔记中的所有例子都将PCA配置为用SYSCLK/4作为时钟源。

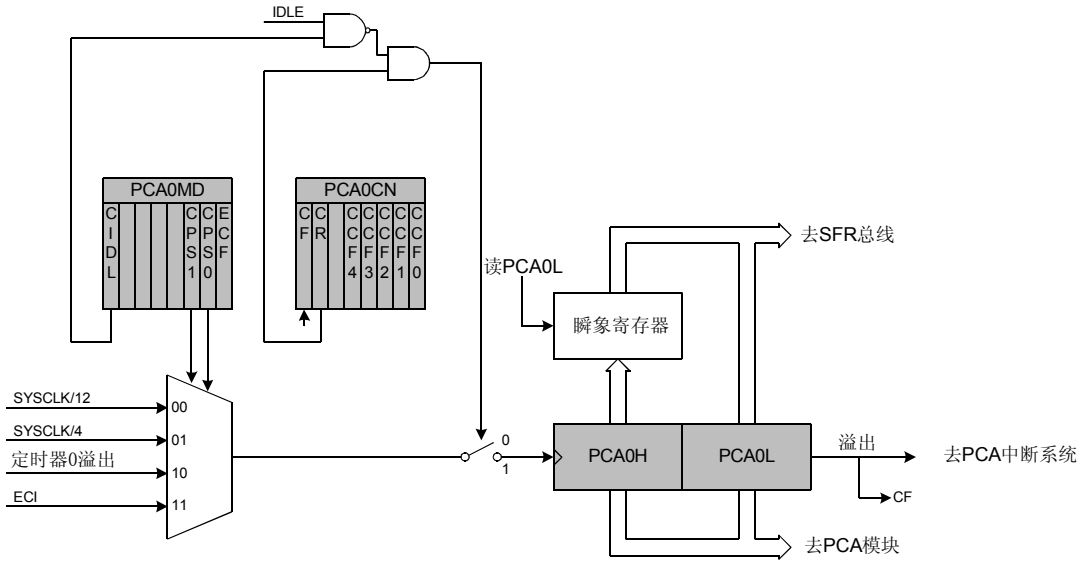

图3. PCA计数器/定时器原理框图

用 PCA 实现 8 位 PWM

我们首先介绍一个能产生8位精度PWM波形的方法。为了叙述的完整性，先介绍PCA的PWM方式。

在该方式中，捕捉/比较模块被配置为图4所示的PWM方式。出现在*CEXn*的波形的周期等于256个PCA时钟周期。该信号的低电平时间等于在模块的捕捉／比较寄存器（PCA0CPLn）的低字节中所存储的8位数值。这种关系如图5所示。

在主PCA计数器（PCA0L）的低字节发生溢出时，模块的比较寄存器的高字节被拷贝到模块的比较寄存器的低字节（PCA0CPLn=PCA0CPHn）。通过更新PCA0CPHn就能改变占空度。拷贝过程保证在输出端不产生毛刺。

输出波形的占空度（用％表示）由下式给出：

$$占空度=\frac{256-PCA0CPHn}{256}\times 100$$

因为PCA0CPHn可以含有一个0~255的数值，所以可编程的最大和最小占空度为0.38%（PCA0CP0H=0xFF）到100% (PCA0CP0H=0x00)。占空度选择的分辨率为：

$$分辨率=\frac{1}{256}\times 100=0.38$$

8位PWM方式的最大优点是不需要CPU的干预就可以输出一个固定占空度的波形。事实上，如果CIDL位（PCA0MD.7）被设置为‘0’（复位状态），即使CPU处于休眠状态，输出波形也将保持。

改变占空度是通过向PCA0CPHn写入一个8位数来完成。

在本应用笔记最后包含的文件‘PWM8_1.C’提供了一个8位PWM方式的例子。

8位PWM方式的其它注意事项：

1. 可以通过将模块配置寄存器中的ECOMn位（PCA0CPMn.6）清0使*CEXn*输出保持为低电平。这样就可以产生一个0%占空度的波形。可以通过向该位写‘1’或向PCA0CPHn写入一个任意值来恢复正常的PWM输出。
2. 将MATn和ECCFn位（分别为PCA0CPMn.3 和 PCA0CPMn.0）设置为‘1’会导致在*CEXn*的下降沿产生个中断。

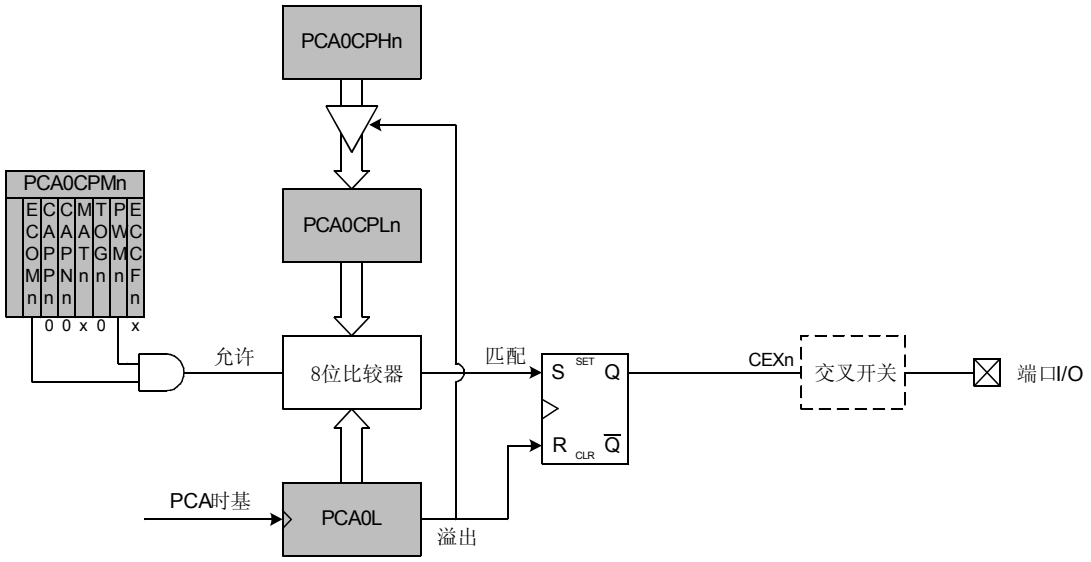

图4. PCA被配置为8位PWM方式

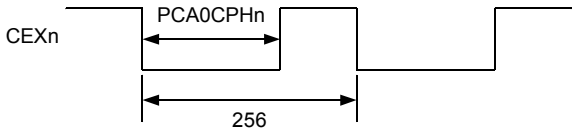

图5. 8位PWM方式的输出波形

用 PCA 产生 16 位 PWM

为了产生一个具有16位精度的PWM波形，我们将PCA模块配置为高速输出方式，如图6所示。在该方式下，每当主定时器 / 计数器的寄存器（PCA0H:PCA0L）与模块的捕捉/比较寄存器（PCA0CPHn:PCA0CPLn）相匹配时，*CEXn*引脚发生电平转换，并可以选择产生中断。

在示例代码中，PCA模块的中断服务程序用两个状态中实现：上升沿状态和下降沿状态，取决于CEXn引脚上哪一个边沿触发中断。注意，现在所说的的*CEXn*引脚被看作状态变量。

在上升沿状态，模块的捕捉/比较寄存器被下一个下降沿的比较值更新（该值在示例代码中被称为*PWM*）。在下降沿状态，下一个上升沿的比较值被装入模块的捕捉/比较寄存器，该值为 0（0x0000）。这一过程如图7所示。注意，PWM波形的周期为65536个PCA时钟周期。

占空度（用%表示）由下式给出：

$$占空度 = \frac{PWM}{65536} \times 100$$

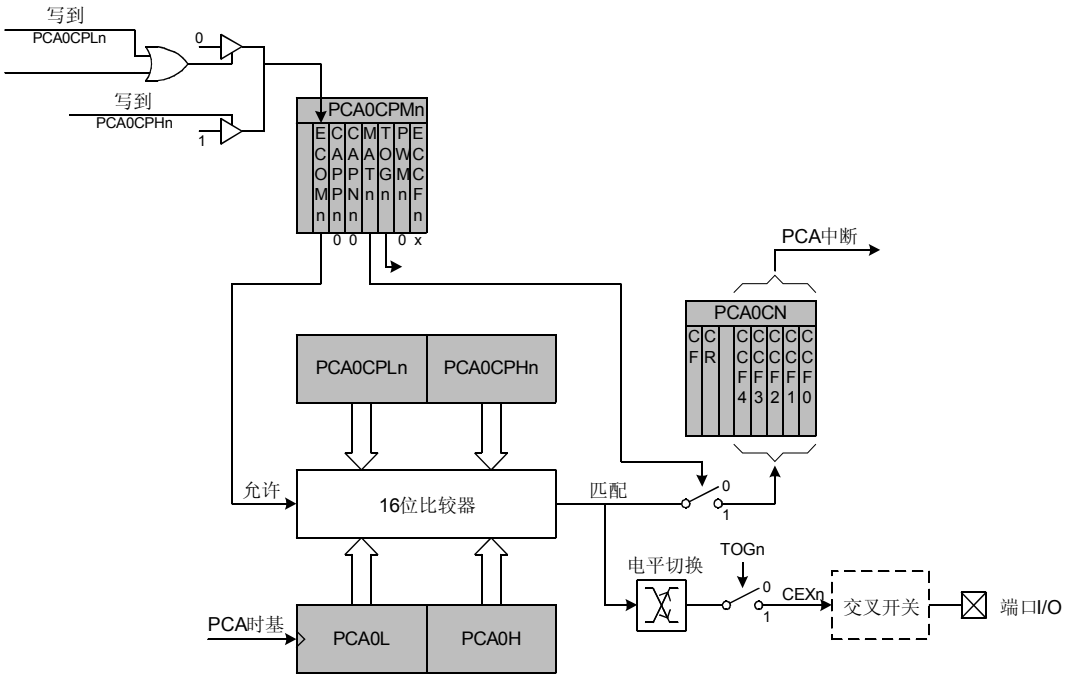

图6. PCA配置为高速输出方式

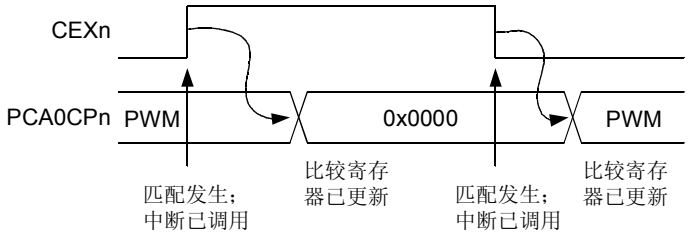

图7. 16位PWM的捕捉/比较寄存器装载过程

所允许的最大和最小占空度由*CEXn*发生变化（它触发过程中断）后更新比较值所需要的最大时间决定。在‘C’示例代码和汇编示例代码中（分别为‘pwm16_1.c’和‘pwm16_1.asm’），PWM的最小值7个PCA时钟（对于本例为28个SYSCLK周期）。在这种情况下最小和最大占空度的值分别为0.01%和99.99%。占空度的分辨率（用%表示）为：

$$\text{分辨率} = \frac{1}{65536} \times 100 = 0.0015$$

或大约15个ppm（1ppm=百万分之一）。

处理该中断的CPU开销是最小的。在汇编示例代码中处理两个边沿共需要41个SYSCLK周期，这不包括中断调用和中断返回所需的时间。两个边沿都必须每隔65536个PCA时钟被处理一次，或65536*4=262144个SYSCLK周期（如果PCA时钟等于SYSCLK / 4）。所消耗的CPU带宽（用%表示）等于（41/262144*100）=0.015%。

还应注意到CPU可以在保留在等待方式，如示例中所做的那样，这是因为PCA模块中断在需要处理时将‘唤醒’CPU内核。

在示例中，占空度是通过向变量*PWM*写入一个16位的值来改变的。

用 PCA 实现 n 位 PWM

作为16位PWM的推广，我们介绍在一些应用中所需要的高于8位精度但低于16位精度的n位PWM。采用n位PWM方案的动机之一是为了获得比16位实现方式更高的PWM输出频率。

在该例中（‘PWMn_1.c’）用到两个16位变量：*PWM__HIGH*（保持用于使输出波形处于高电平所需要的PCA时钟数），*PWM__LOW*（保持用于使输出波形处于低电平所需要的PCA时钟数）。输出波形的周期为这两个变量的和：

$$\textit{周期} = PWM\_HIGH + PWM\_LOW$$

占空度（用%表示）由下式给出：

$$\textit{占空度} = \frac{PWMHIGH}{PWMHIGH + PWMLOW} \times 100$$

占空度的分辨率（用%表示）为：

$$分辨率 == \frac{1}{PWMHIGH + PWMLOW} \times 100$$

与16位PWM的情况类似，中断处理过程在两个状态中实现：一个用于上升沿状态，另一个用于下降沿状态。主要区别是在16位的情况下被装入到PCA模块比较寄存器的数值是一个常数（*PWM*或0）。在n位的情况下，常数（*PWM_HIGH*或*PWM_LOW*）被加到模块的比较寄存器的当前值。这一加法操作比装载一个常数需要多花几个时钟周期，这就导致了输出波形的高电平或低电平的最小时间略大于相应的16位解决方案。

注：通过向*PWM_HIGH*和*PWM_LOW*写入合适的高电平和低电平值，n位PWM方案可以用于产生一个任意频率的波形。

软件示例

PWM8_1.c

```
//-----------------------------------------------------------------------
// PWM8_1.c
//-----------------------------------------------------------------------
//
// AUTH: BW
//
// 目标器件:    C8051F000、F001、F002、F005、F006、F010、F011或F012
// 工具链:      KEIL C51
//
// 说明:
//                  实现8位PWM的示例代码。
//                  PCA被配置为8位PWM方式，使用SYSCLK/4作为时基信号。
//                  <PWM>中保持着每256个计数周期内输出波形为低电平的PCA周期数。
//                  波形为高电平的时间占（256 - PWM）个周期。
//                  输出波形的占空度=(256 - PWM)/256。
//
//                  由于该8位PWM完全受硬件控制，不需要额外的CPU周期来维持固定的
//                  占空度。在本例中，改变占空度只需要向模块的比较寄存器PCA0CP0H
//                  的高字节写一个8位数。
//
//                  可以达到的占空度范围是0.38%(PCA0CP0H = 0xff)
//                  到100%(PCA0CP0H = 0x00)。
//

//-----------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------

#include <c8051f000.h>                  // SFR 声明

//-----------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------

#define PWM             0x80            // 波形为低电平的PCA周期数
                                        // 占空度 = (256 - PWM) / 256
                                        // 注：这是一个8位数
```

```
//-----------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------

void main (void);

//-----------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------

void main (void) {

    WDTCN = 0xde;                       // 禁止看门狗定时器
    WDTCN = 0xad;

    OSCICN = 0x07;                      // 设置SYSCLK到16MHz，内部振荡器。

    XBR0 = 0x08;                        // 使CEX0输出到P0.0
    XBR2 = 0x40;                        // 允许交叉开关和弱上拉

    PRT0CF = 0x01;                      // 设置P0.0输出为推挽方式
    PRT1CF = 0x20;                      // 设置P1.6输出为推挽方式(LED)


    // 配置PCA
    PCA0MD = 0x02;                      // 禁止CF中断
                                        // PCA时基= SYSCLK / 4
    PCA0CPL0 = PWM;                     // 初始化PCA的PWM值
    PCA0CPH0 = PWM;
    PCA0CPM0 = 0x42;                    // CCM0为8位PWM方式
    PCA0CN = 0x40;                      // 允许PCA计数器

    while (1) {
        PCON |= 0x01;                   // 设置等待方式
    }
}
```

PWM16_1.c

```
//-----------------------------------------------------------------------
// PWM16_1.c
//-----------------------------------------------------------------------
//
// 作者:       BW
//
// 目标器件:   C8051F000、F001、F002、F005、F006、F010、F011或F012
// 工具链:     KEIL C51
//
// 说明:
//             实现16位PWM的示例代码。
//             PCA被配置为高速输出方式，使用SYSCLK/4为时基信号。
//             <PWM>保持输出波形为高电平的PCA周期数。
//             波形的低电平时间为(65536-PWM)个周期。输出的占空度=PWM/65536。
//
//             由于有中断服务时间，所以PWM有最小和最大值，
//             占空度也是如此，取决于中断服务时间
//             在使用Keil C51编译器（评估版）时，最小的PWM值是7个PCA时钟。
//             最大值是65530。这相当于最小的占空度0.01%最大的占空度99.99%。
//             这是基于如下假设：PCA时基信号为SYSCLK/4，没有其它中断服务。
//

//-----------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------

#include <c8051f000.h>                  // SFR声明

//-----------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------

#define PWM_START       0x4000          // 对应于PWM高电平时间的起始值
sbit      PWM_OUT = P0^0;               // 定义PWM输出端口引脚

//-----------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------

void main (void);
void PCA_ISR (void);                    // PCA中断服务程序

//-----------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------
unsigned PWM = PWM_START;               // 使波形为高电平的PCA周期数
                                        // 占空度 = PWM/65536
                                        // 注：这是一个16位数
```

```
//-----------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------

void main (void) {

    WDTCN = 0xde;                       // 禁止看门狗定时器
    WDTCN = 0xad;

    OSCICN = 0x07;                      // 设置SYSCLK为16MHz，内部振荡器

    XBR0 = 0x08;                        // 使CEX0输出到P0.0
    XBR2 = 0x40;                        // 允许交叉开关和弱上拉

    PRT0CF = 0x01;                      // 设置P0.0输出为推挽方式
    PRT1CF = 0x20;                      // 设置P1.6输出为推挽方式(LED)


    // 配置PCA
    PCA0MD = 0x02;                      // 禁止CF中断
                                        // PCA时基 = SYSCLK/4
    PCA0CPL0 = (0xff & PWM);            // 初始化PCA比较值
    PCA0CPH0 = (0xff & (PWM >> 8));
    PCA0CPM0 = 0x4d;                    // CCM0为高速输出方式

    EIE1 |= 0x08;                       // 允许PCA中断

    EA = 1;                             // 允许全局中断

    PCA0CN = 0x40;                      // 允许PCA计数器

    while (1) {
        PCON |= 0x01;                   // 设置等待方式
    }
}

//-----------------------------------------------------------------------
// PCA_ISR
//-----------------------------------------------------------------------
//
// 该ISR在PCA CCM0得到一次匹配时被调用。
// PWM_OUT是CEX0端口引脚，它保持当前边沿的状态：1 = 上升沿；0 = 下降沿
// 在上升沿，PWM_HIGH 被装入比较寄存器。在下降沿，比较寄存器被装入0。
//
void PCA_ISR (void) interrupt 9
{
    if (CCF0) {
        CCF0 = 0;                           // 清除比较标志
        if (PWM_OUT) {                      // 处理上升沿

            PCA0CPL0 = (0xff & PWM);        // 设置PWM的下一个匹配值
            PCA0CPH0 = (0xff & (PWM >> 8));
```

```
      } else {                              // 处理下降沿

         PCA0CPL0 = 0;                      // 设置下一个匹配值为0
         PCA0CPH0 = 0;

      }
   } else if (CCF1) {                       // 处理其它PCA中断
      CCF1 = 0;
   } else if (CCF2) {
      CCF2 = 0;
   } else if (CCF3) {
      CCF3 = 0;
   } else if (CCF4) {
      CCF4 = 0;
   } else if (CF) {
      CF = 0;
   }
```

PWM16_1.asm

```
;------------------------------------------------------------------------
;   CYGNAL INTEGRATED PRODUCTS, INC.
;
;
;   文件名:     pwm16_1.ASM
;   目标MCU:    C8051F000、F001、F002、F005、F006、F010、F011或F012
;   说明:       实现16位PWM的示例代码。
;               PCA被配置为高速输出方式，使用SYSCLK/4作为时基信号。
;               <PWM>保持输出波形为高电平的PCA周期数。
;               波形的低电平时间为(65536-PWM)个周期。输出的占空度=PWM/65536。
;
;               由于有中断服务时间，最小的PWM值是7个PCA时钟，最大值是65529。
;               这相当于最小的占空度0.01068%，最大的占空度99.9893%。
;               这是基于如下假设：PCA时基信号为SYSCLK/4，没有其它中断服务。
;
;               如果PCA时基信号变为 SYSCLK/12，则最小和最大PWM值分别为3和65533，
;               对应的最小和最大占空度分别为0.0046%和99.9954%。
;
;               处理上升沿中断需要18个周期。处理下降沿中断需要19个周期。
;
;               中断处理程序在每个边沿被调用，每65536个PCA时钟有两个边沿。
;               用SYSCLK/4作为PCA时基信号时，每(65536*4)=262,144个SYSCLK中
;               有37个周期用于边沿处理，这里未计中断调用和返回时间。
;               CPU占用率为(37/262,144)*100% = 0.0141%。
;
;               用SYSCLK/12作为PCA时基信号时，每(65536*12)= 786,432个SYSCLK中
;               有37个周期用于边沿处理。CPU占用率为(37/786,432)*100% = 0.0047%。
;
;               波形的周期是65536个PCA时钟。用SYSCLK/4作为PCA时基信号时，
;               该周期为262,144个SYSCLK周期。用缺省的内部振荡器工作于2MHz，
```

```
;               该周期为131ms(7.6Hz)。使用16MHz的内部振荡器时（如本例），
;               该周期为16.4ms(61Hz)。
;
;               用SYSCLK/12作为PCA时基信号时，该周期为65536*12 =786,432
;               个SYSCLK周期。用缺省的内部振荡器工作于2MHz，该周期为393ms(2.5Hz)。
;               使用16MHz的内部振荡器时（如本例），该周期为49.2ms(20Hz)。
;
;               在本例中，输出被连到P0.0，也被标识为‘PWM_OUT’。
;
;-----------------------------------------------------------------------

;-----------------------------------------------------------------------
; 等价定义
;-----------------------------------------------------------------------

$MOD8F000

PWM         EQU     32768               ; 波形为高电平的PCA周期数
                                        ; 占空度 = PWM / 65536
                                        ; 最大值 = 65529 (99.9893% 占空度)
                                        ; 最小值 = 7 (0.01068% 占空度)
                                        ; 注：这是一个16位的常数

PWM_OUT     EQU     P0.0                ; 定义PWM输出端口引脚

;-----------------------------------------------------------------------
; 复位和中断向量表
;-----------------------------------------------------------------------

CSEG
        org     00h
        ljmp    Main

        org     04bh
        ljmp    PCA_ISR                 ; PCA中断服务程序

;-----------------------------------------------------------------------
; 主程序
;-----------------------------------------------------------------------
        org     0b3h                            ; 在中断处理空间之后开始
Main:
        ; 禁止看门狗定时器
        mov     WDTCN, #0deh
        mov     WDTCN, #0adh

        ;   允许内部振荡器工作于 16MHz
        mov     OSCICN, #07h

        ; 允许交叉开关和弱上拉
        mov     XBR0, #08h                      ; CEX0连到P0.0
        mov     XBR2, #40h

        orl     PRT0CF, #01h                ; 配置 P0.0
```

```
        ; 配置 PCA
        mov     PCA0MD, #02h                ; 禁止 cf 中断，PCA时基 = SYSCLK/4
        mov     PCA0CPL0, #LOW(PWM)         ; 初始化PCA比较值
        mov     PCA0CPH0, #HIGH(PWM)
        mov     PCA0CPM0, #4dh              ; CCM0 为高速输出方式

        ; 允许中断
        orl     EIE1,  #08h                 ; 允许PCA中断
        setb    EA                        ; 允许全局中断

        mov     PCA0CN, #40h                ; 允许PCA计数器

        jmp     $

;------------------------------------------------------------------------
;   CCF0 中断向量
;
;
// 该ISR在PCA CCM0得到一次匹配时被调用。
// PWM_OUT是CEX0端口引脚，它保持当前边沿的状态：1 = 上升沿；0 = 下降沿
// 在上升沿，PWM_HIGH 被装入比较寄存器。在下降沿，比较寄存器被装入0。

PCA_ISR:
        jbc     CCF0, CCF0_HNDL             ; 处理CCF0比较
        jbc     CCF1, PCA_STUB              ; 分支程序
        jbc     CCF2, PCA_STUB
        jbc     CCF3, PCA_STUB
        jbc     CCF4, PCA_STUB
        jbc     CF,   PCA_STUB

PCA_STUB:
PCA_ISR_END:
        reti

CCF0_HNDL:
        jnb     PWM_OUT, CCF0_FALL
                                            ; 处理上升沿
CCF0_RISE:
        mov     PCA0CPL0, #LOW(PWM)
        mov     PCA0CPH0, #HIGH(PWM)

        reti

CCF0_FALL:                                  ; 处理下降沿
        mov     PCA0CPL0, #00
        mov     PCA0CPH0, #00

        reti

; 上升沿需要 4+3+11 = 18 个周期
; 下降沿需要 4+4+11 = 19 个周期

;------------------------------------------------------------------------
```

```
; END
;------------------------------------------------------------------------

END
```

PWMn_1.c

```
//-----------------------------------------------------------------------
// PWMn_1.c
//-----------------------------------------------------------------------
//
// 作者:        BW
//
// 目标器件:    C8051F000、F001、F002、F005、F006、F010、F011或F012
// 工具链:      KEIL C51
//
// 说明:
//              实现n位PWM的示例源代码。
//              PCA被配置为高速输出方式，使用SYSCLK/4为时基信号。
//              <PWM_HIGH>保持输出波形为高电平的PCA周期数。
//              <PWM_LOW>保持输出波形为低电平的PCA周期数。
//              输出的占空度= PWM_HIGH /( PWM_HIGH+ PWM_LOW)。
//
//              由于有中断服务时间，因此PWM_HIGH和PWM_LOW有最小和最大值，
//              占空度也是如此，取决于中断服务时间。如果不考虑编译器的效率，
//              1%和99%之间的占空率很容易达到。
//              在使用评估版的Keil C51编译器时，最小的高电平和低电平时间各为
//              20个PCA时钟（最大频率约为100kHz w/16MHz内部SYSCLK)。
//              这是基于如下假设：PCA时基信号为SYSCLK/4，没有其它中断服务。
//

//-----------------------------------------------------------------------
// 包含文件
//-----------------------------------------------------------------------

#include <c8051f000.h>                  // SFR声明

//-----------------------------------------------------------------------
// 全局常量
//-----------------------------------------------------------------------

#define PWM_START       0x8000          // PWM_HIGH时间和PWM_LOW时间的起始值
sbit      PWM_OUT = P0^0;               // 定义PWM输出端口引脚

//-----------------------------------------------------------------------
// 函数原型
//-----------------------------------------------------------------------

void main (void);
void PCA_ISR (void);                    // PCA中断服务程序

//-----------------------------------------------------------------------
// 全局变量
//-----------------------------------------------------------------------
unsigned PWM_HIGH = PWM_START;          // 波形为高电平的PCA周期数
```

```
unsigned PWM_LOW = ~PWM_START;          // 波形为低电平的PCA周期数
                                       //占空度=PWM_HIGH/(PWM_HIGH+PWM_LOW)

//-----------------------------------------------------------------------
// 主程序
//-----------------------------------------------------------------------

void main (void) {

    WDTCN = 0xde;                      // 禁止看门狗定时器
    WDTCN = 0xad;

    OSCICN = 0x07;                     // 设置SYSCLK为16MHz，内部振荡器

    XBR0 = 0x08;                       // CEX0输出到P0.0
    XBR2 = 0x40;                       // 允许交叉开关和弱上拉

    PRT0CF = 0x01;                     // 设置P0.0为推挽输出方式
    PRT1CF = 0x20;                     // 设置P1.6为推挽输出方式(LED)


    // 配置PCA
    PCA0MD = 0x02;                     // 禁止CF中断
                                       // PCA时基=SYSCLK/4
    PCA0CPL0 = (0xff & PWM_HIGH);      // 初始化PCA比较值
    PCA0CPH0 = (0xff & (PWM_HIGH >> 8));
    PCA0CPM0 = 0x4d;                   // CCM0为高速输出方式

    EIE1 |= 0x08;                      // 允许PCA中断

    EA = 1;                            // 允许全局中断

    PCA0CN = 0x40;                     // 允许PCA计数器

    while (1) {
        PCON |= 0x01;                  // 设置等待方式
    }
}

//-----------------------------------------------------------------------
// PCA_ISR
//-----------------------------------------------------------------------
//
// 该ISR在PCA CCM0得到一次匹配时被调用。
// PWM_OUT是CEX0端口引脚，它保持当前边沿的状态：1 = 上升沿；0 = 下降沿
// 在上升沿，比较寄存器被更新，以触发下一个下降沿。
// 在下降沿，比较寄存器被更新，以触发下一个上升沿。
//

void PCA_ISR (void) interrupt 9
{
    unsigned temp;                         // 保存16位匹配值
```

```
    if (CCF0) {
        CCF0 = 0;                           // 清除比较标志
        if (PWM_OUT) {                      // 处理上升沿

            // 为下一个下降沿更新比较匹配值
            temp = (PCA0CPH0 << 8) | PCA0CPL0;   // 取当前比较值
            temp += PWM_HIGH;                    // 加上合适的偏移量

            PCA0CPL0 = (0xff & temp);            // 更换比较值
            PCA0CPH0 = (0xff & (temp >> 8));

        } else {                                 // 处理下降沿

            //为下一个上升沿更新比较匹配值
            temp = (PCA0CPH0 << 8) | PCA0CPL0;   // 取当前比较值
            temp += PWM_LOW;                     // 加上合适的偏移量

            PCA0CPL0 = (0xff & temp);            // 更换比较值
            PCA0CPH0 = (0xff & (temp >> 8));

        }
    } else if (CCF1) {                           // 处理其它PCA中断
        CCF1 = 0;
    } else if (CCF2) {
        CCF2 = 0;
    } else if (CCF3) {
        CCF3 = 0;
    } else if (CCF4) {
        CCF4 = 0;
    } else if (CF) {
        CF = 0;
    }
}
```