# 第 2-15 讲：脉宽调制 PWM

## 1. 学习目的

1. 掌握 PWM 的基本概念。
2. 掌握 PWM 周期和占空比的计算。
3. 掌握 PWM 的基本编程应用，编程实现 PWM 输出不同频率和占空比的波形。
4. 掌握 PWM 实现呼吸灯。
5. 掌握 PWM 驱动震动马达，通过不同的占空比控制震动马达的震动强度。

## 2. PWM 概述

PWM (全称是 Pulse Width Modulation)脉冲宽度调制是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。

所谓的脉冲宽度调制，也就是输出占空比可变的脉冲波形。下图是一个 PWM 控制模拟电路（电压是模拟量）的示例，当开关直接接通时，电灯供电电压为 9V，当使用 PWM 脉冲控制开关的接通和闭合时，PWM 占空比为 50%的情况下，可等效为 4.5V 电池给电灯供电，电灯的亮度会有所下降。同理，占空比是 10%，等效于 0.9V 电池给电灯供电，亮度会很暗。由此，我们可以看到，通过改变 PWM 信号的频率和占空比可以控制电灯的状态和亮度。

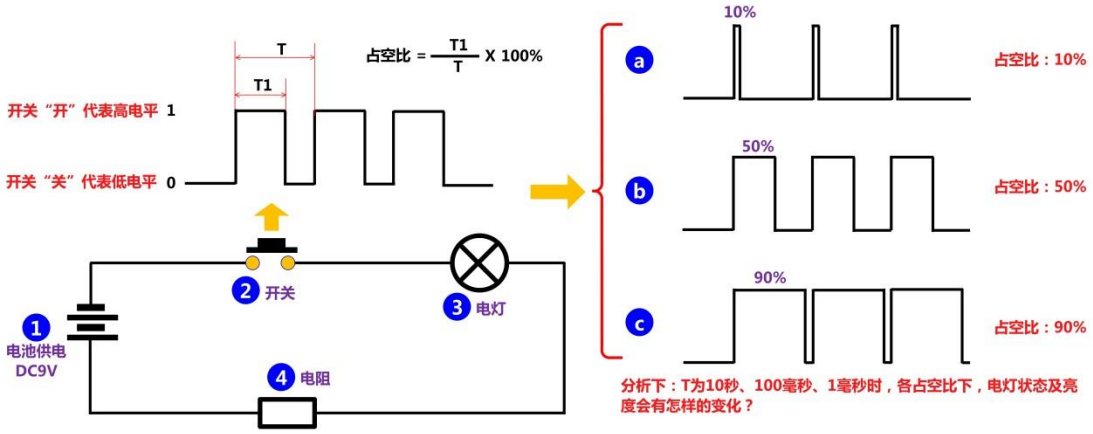



图 1：PWM 控制模拟电路示意图

■ **PWM 的重要概念**

1） 周期和频率

PWM 周期是指 PWM 信号从一个上升沿执行到下一个上升沿所需要的时间，周期的倒数即为 PWM 频率。

2） 占空比

一个 PWM 周期中，高电平保持的时间与该 PWM 的周期的时间之比，如一个 PWM 的

周期是 10ms，高电平的时间是 1ms，那么占空比就是 10%。

3） 边沿对齐、中心对齐、单斜率和双斜率

- 单斜率：PWM 计数器从 0 开始计数到最大值，达到最大值后清零重新从 0 开始计数，如此反复。单斜率和 PWM 比较值实现的是边沿对齐。
- 双斜率：PWM 计数器从 0 开始计数到最大值，再从最大值计数到 0，如此反复。双斜率和 PWM 比较值实现的是中心对齐。

4） 分辨率

占空比最小能达的数值，如 8 位的 PWM，理论的分辨率就是 1:255(单斜率)，16 位的 PWM 理论就是 1:65535(单斜率)。

■ **PWM 信号控制模拟电路的优点**

PWM 信号是数字信号，处理器到被控系统信号都是数字形式的，可将噪声影响降到最低，噪声只有在强到足以将逻辑 1 改变为逻辑 0 或将逻辑 0 改变为逻辑 1 时，也才能对数字信号产生影响。PWM 对噪声抵抗能力的增强，因此，从模拟信号转向 PWM 可以极大地延长通信距离。

■ **PWM 的应用领域**

1） 电机驱动：PWM 被称为“开关驱动装置”，PWM 信号的高低电平可控制电机是否通电，电机通电，就会加速；电机断电，就会减速甚至停止。只要 PWM 信号频率达到一定值，改变 PWM 占空比可实现对电机速度的控制。

2） 开关电源：PWM 开关型稳压电路是在控制电路输出频率不变的情况下，通过电压反馈调整其占空比，从而达到稳定输出电压的目的。

3） 电池充电：在镍氢电池智能充电器中采用的脉宽 PWM 法，它是把每一脉冲宽度均相等的脉冲列作为 PWM 波形，通过改变脉冲列的周期可以调频，改变脉冲的宽度或占空比可以调压，采用适当控制方法即可使电压与频率协调变化。实现通过调整 PWM 的周期、PWM 的占空比而达到控制充电电流的目的。

4） D/A 转换：PWM 高频输出后加 RC 滤波电路，通过改变 PWM 信号的占空比实现输出不同电压值的目的。

5） 逆变电路：目前中小功率的逆变电路几乎都采用 PWM 技术，逆变电路是 PWM 控制技术极为重要的应用场合。

## 3. STC8A8K64D4 的 PWM 应用步骤

STC8A8K64D4 单片机片内集成了 1 组增强型的 PWM 波形发生器，可产生各自独立的 8 路 PWM。

PWM 的时钟源可以通过 PWM 时钟选择寄存器（PWMCKS）配置，但需要注意的是并不是 8 路 PWM 的时钟源可以独立配置，他们是共用一个时钟源的。

PWM 波形发生器内部有一个 15 位的 PWM 计数器供 8 路 PWM 使用，用户可以设置每路 PWM 的初始电平。另外， PWM 波形发生器为每路 PWM 又设计了两个用于控制波

形翻转的计数器 T1/T2，可以非常灵活的控制每路 PWM 的高低电平宽度，从而达到对 PWM 的占空比以及 PWM 的输出延迟进行控制的目的。

由于 8 路 PWM 是各自独立的，且每路 PWM 的初始状态可以进行设定，所以用户可以将其中的任意两路配合起来使用，即可实现互补对称输出以及死区控制等特殊应用。

增强型的 PWM 波形发生器还设计了对外部异常事件（包括外部端口 P3.5 电平异常、比较器比较结果异常）进行监控的功能，可用于紧急关闭 PWM 输出。PWM 波形发生器还可与 ADC 相关联，设置 PWM 周期的任一时间点触发 ADC 转换事件。

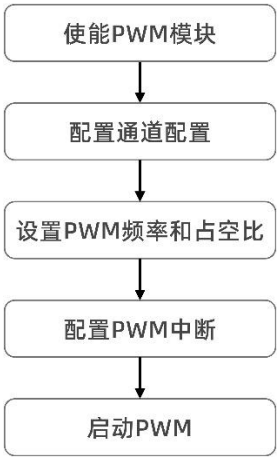



图 2：PWM 应用步骤

### 3.1. 使能 PWM 模块

配置 PWM 模块之前，必须先使能 PWM 模块，否则，配置无效。PWM 模块是通过“增强型 PWM 全局配置寄存器（PWMSET）”中的 ENPWM 位使能的，如下图所示。

**增强型 PWM 全局配置寄存器（PWMSET）：**

PWM模块使能位

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| PWMSET | F1H | - | PWMRST | - | - | - | - | - | ENPWM |

P_SW2 寄存器中的 I2C_S[1:0]为 I2C 功能脚选择位，如下表所示。

- ENPWM：PWM 使能位（使能/关闭 PWM 通道 PWM0~PWM7）。
  - 0：关闭 PWM。
  - 1：使能 PWM。

### 3.2. PWM 通道配置

PWM 通道配置包含 PWM 通道功能引脚配置、PWM 通道初始电平以及 PWM 通道中断配置。PWM 模块的 8 个通道各自具有一个独立的通道配置寄存器（PWMnCR，n=0~7），可以对各个通道单独配置，如下图所示。

每个 PWM 通道对应多个物理引脚，但使用时只能选择其中一个引脚作为该 PWM 通道的输出引脚。选择了 PWM 通道输出引脚后，还需要使能该引脚为 PWM 通道输出，并配置 PWM 通道的初始电平，初始电平会体现在 PWM 通道的输出引脚上。

**PWM 通道控制寄存器（PWMnCR，n=0~7）：**

| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| PWM0CR | FF14H | ENO | INI | - | C0_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM1CR | FF1CH | ENO | INI | - | C1_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM2CR | FF24H | ENO | INI | - | C2_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM3CR | FF2CH | ENO | INI | - | C3_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM4CR | FF34H | ENO | INI | - | C4_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM5CR | FF3CH | ENO | INI | - | C5_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM6CR | FF44H | ENO | INI | - | C6_S[1:0] | | ENI | ENT2I | ENT1I |
| PWM7CR | FF4CH | ENO | INI | - | C7_S[1:0] | | ENI | ENT2I | ENT1I |

1) ENO： PWMn 输出使能位。（ n＝ 0～7）
   - 0： PWM 的 通道 n 相应端口为 GPIO。
   - 1： PWM 的通道 n 相应端口为 PWM 输出口，受 PWM 波形发生器控制。
2) INI： 设置 PWMn 输出端口的初始电平。（ n＝ 0～7）
   - 0： PWM 的 通道 n 初始电平为低电平。
   - 1： PWM 的 通道 n 初始电平为高电平。
3) Cn_S[1:0]（n=0~7）： PWM 通道输出脚选择位。

PWM 模块的每个通道均有多个引脚与之对应，同一时刻，只能选择其中的一个引脚作为 PWM 输出引脚使用， STC8A8K64D4 单片机 PWM 的各个通道的引脚分配如下表所示。

表 1：STC8A8K64D4 单片机 PWM 引脚分配

| PWM 通道 | 信号编号 | 对应的 IO | Cn_S[1:0]位（n=0~7） |
|---|---|---|---|
| PWM 通道 0 | PWM0 | P2.0 | C0_S[1:0] = 00 |
| | PWM0_2 | P1.0 | C0_S[1:0] = 01 |
| | PWM0_3 | P6.0 | C0_S[1:0] = 10 |
| PWM 通道 1 | PWM1 | P2.1 | C1_S[1:0] = 00 |
| | PWM1_2 | P1.1 | C1_S[1:0] = 01 |
| | PWM1_3 | P6.1 | C1_S[1:0] = 10 |
| PWM 通道 2 | PWM2 | P2.2 | C2_S[1:0] = 00 |
| | PWM2_2 | P1.2 | C2_S[1:0] = 01 |
| | PWM2_3 | P6.2 | C2_S[1:0] = 10 |
| PWM 通道 3 | PWM3 | P2.3 | C3_S[1:0] = 00 |
| | PWM3_2 | P1.3 | C3_S[1:0] = 01 |
| | PWM3_3 | P6.3 | C3_S[1:0] = 10 |
| PWM 通道 4 | PWM4 | P2.4 | C4_S[1:0] = 00 |
| | PWM4_2 | P1.4 | C4_S[1:0] = 01 |
| | PWM4_3 | P6.4 | C4_S[1:0] = 10 |

| | | | |
|---|---|---|---|
| PWM 通道 5 | PWM5 | P2.5 | C5_S[1:0] = 00 |
| | PWM5_2 | P1.5 | C5_S[1:0] = 01 |
| | PWM5_3 | P6.5 | C5_S[1:0] = 10 |
| PWM 通道 6 | PWM6 | P2.6 | C6_S[1:0] = 00 |
| | PWM6_2 | P1.6 | C6_S[1:0] = 01 |
| | PWM6_3 | P6.6 | C6_S[1:0] = 10 |
| PWM 通道 7 | PWM7 | P2.7 | C7_S[1:0] = 00 |
| | PWM7_2 | P1.7 | C7_S[1:0] = 01 |
| | PWM7_3 | P6.7 | C7_S[1:0] = 10 |

4) ENI： PWM 的通道 n 中断使能控制位。（n＝ 0～7）
   - 0： 关闭 PWM 通道 n 的 PWM 中断。
   - 1： 使能 PWM 通道 n 的 PWM 中断。
5) ENT2I： PWM 的 i 通道在第 2 个触发点中断使能控制位。（ n＝ 0～7）
   - 0： 关闭 PWM 通道 n 在第 2 个触发点中断。
   - 1： 使能 PWM 通道 n 在第 2 个触发点中断。
6) ENT1I： PWM 的 i 通道在第 1 个触发点中断使能控制位。（ n＝ 0～7）
   - 0： 关闭 PWM 通道 n 在第 1 个触发点中断。
   - 1： 使能 PWM 通道 n 在第 1 个触发点中断。

✧ **说明**：PWM 通道中断在后面的章节中和归零中断一起讲解。

## 3.3. 周期和占空比

对于一个 PWM 输出波形来说，最直观的参数就是波形的频率和占空比，其中频率反映了一个 PWM 周期占用的时间，占空比反映了一个周期内高电平的时间和 PWM 的周期的时间之比。接下来，我们就从这两个方面讲解 PWM 波形的输出。

### 3.3.1. PWM 输出波形的周期、频率

PWM 输出波形的周期和 PWM 计数器的时钟、PWM 计数器寄存器（PWMC）的值以及计数模式有关。PWM 计数器所用的时钟决定了一个计数所用的时间，PWMC 的值决定了计数器的匹配值，这两点很好理解，那么计数模式是如何影响 PWM 周期的？

PWM 计数模式通常有三种：向上计数、向下计数和向上/向下计数，其中向上计数和向下计数又称为边沿对齐模式，向上/向下计数又称为中心对齐模式。**注意，STC8A8K64D4 单片机的 PWM 只有向上计数模式**。为了能更好的理解 PWM 波形的产生，我们有必要了解一下这三种模式。

**1. 向上计数模式**

向上计数模式中，计数器从 0 计数到 PWMC 寄存器的值（该值由用户写入），然后重

新从 0 开始计数，如此反复执行，如下图所示。

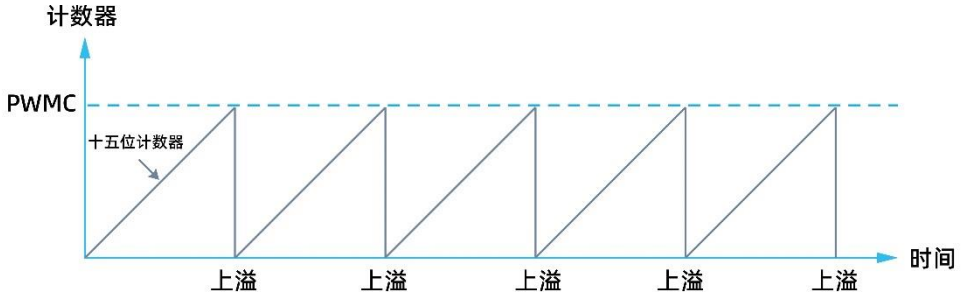



图 3：向上计数

**2. 向下计数模式**

向下计数模式中，计数器从 PWMC 寄存器的值（该值由用户写入）向下计数到 0，然后重新从 PWMC 寄存器的值向下计数，如此反复执行，如下图所示。

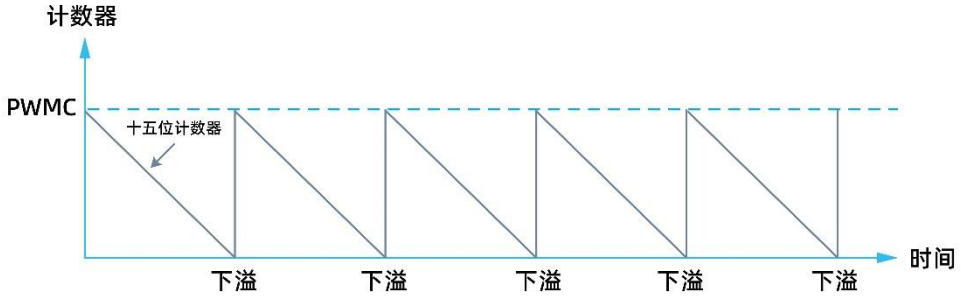



图 4：向上计数

**3. 向上/向下计数模式**

向上/向下计数模式中，从 0 开始计数到 PWMC 寄存器的值（该值由用户写入），接着从 PWMC 寄存器的值向下计数到 0，然后再从 0 开始重新计数，如此反复执行，如下图所示。

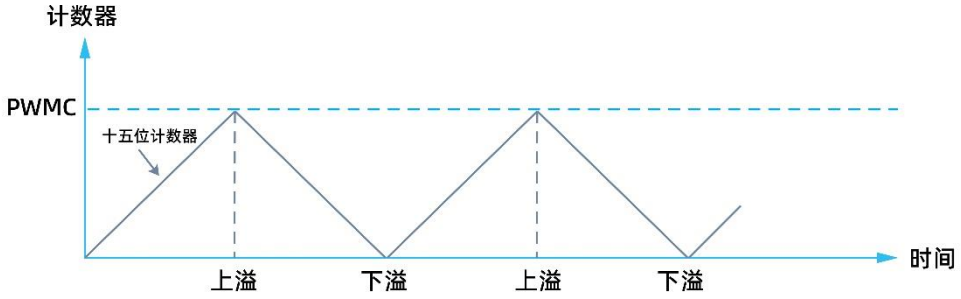



图 5：向上/向下计数

由上面三种计数模式的示意图可以看出，在设置了同样的计数比较值后，向上/向下计数模式的周期是向上计数模式和向下计数模式的 2 倍。

- **PWM 计数器的时钟源**

PWM 计数器的时钟源是通过“PWM 时钟选择寄存器（PWMCKS）”中的“时钟选择位（SELT2）”配置的，有 2 个可选择项：系统时钟和定时器 2 的溢出脉冲。如果选择使用

系统时钟，可通过“PWM_PS[3:0]”设置系统时钟的预分频，如下图所示。

| | | | | | 时钟选择位 | 系统时钟预分频 | | | |
|---|---|---|---|---|---|---|---|---|---|
| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PWMCKS | FF02H | - | - | - | SELT2 | PWM_PS[3:0] | | | |

- SELT2： PWM 时钟源选择。
  - 0： PWM 时钟源为系统时钟经分频器分频之后的时钟
  - 1： PWM 时钟源为定时器 2 的溢出脉冲
- PWM_PS[3:0]：系统时钟预分频参数

| SELT2 | PWM_PS[3:0] | PWM 输入时钟源频率 |
|---|---|---|
| 1 | xxxx | 定时器 2 的溢出脉冲 |
| 0 | 0000 | SYSclk/1 |
| 0 | 0001 | SYSclk/2 |
| 0 | 0010 | SYSclk/3 |
| ... | ... | ... |
| 0 | x | SYSclk/(x+1) |
| ... | ... | ... |
| 0 | 1111 | SYSclk/16 |

- **PWM 计数器寄存器（PWMC）**

由于 PWM 只有向上计数模式，因此，PWMC 设置的值即为 PWM 一个周期所包含的计数个数，PWMC 是一个 15 位的寄存器， 其中：

- PWMCH： PWM 计数器周期值的高 7 位。
- PWMCL： PWM 计数器周期值的低 8 位。

可设定 1~32767 之间的任意值作为 PWM 的周期，PWM 波形发生器内部的计数器从 0 开始计数，每个 PWM 时钟周期递增 1，当内部计数器的计数值达到[PWMCH，PWMCL]所设定的 PWM 周期时，PWM 波形发生器内部的计数器将会从 0 重新开始开始计数。

- **计数模式**

STC8A8K64D4 单片机的 PWM 只有向上计数模式。

综上所述，我们即可得到 PWM 的频率计算公式，如下表所示。

| 时钟源选择（SELT2） | PWM输出频率计算公式 |
|---|---|
| SELT2＝0<br>（系统时钟） | $\text{PWM输出频率} = \frac{\text{系统工作频率SYSclk}}{(\text{PWM\_PS}+1)\times([\text{PWMCH, PWMCL}]+1)}$ |
| SELT2＝1<br>（定时器2的溢出脉冲） | $\text{PWM输出频率} = \frac{\text{定时器2的溢出脉冲频率}}{([\text{PWMCH, PWMCL}]+1)}$ |

### 3.3.2. PWM 输出波形的占空比

上一节中我们知道了如何配置 PWM 输出波形的周期，周期确定后，接下来需要设置的是在一个周期中高电平的占比，即占空比。

PWM 模块的 8 个通道均有两个可配置的触发点，他们都是由两个寄存器组合而成的 15 位寄存器：

- 第 1 触发点：PWMiT1H 的低 7 位和 PWMiT1L 组成 15 位寄存器，i=0~7。
- 第 2 触发点：PWMiT2H 的低 7 位和 PWMiT2L 组成 15 位寄存器，i=0~7。

当 PWM 计数器的计数值到达第 1 触发点设置的数值时，PWM 输出低电平，当 PWM 计数器的计数值到达第 2 触发点设置的数值时，PWM 输出高电平。由此，通过设置第 1 触发点和第 2 触发点的数值，即可得到我们需要的占空比。

为了便于读者理解，我们用图形的方式来说明如何通过两个触发点来配置占空比。下面讨论了两种方式，读者对比一下这两种方式，即可清晰理解两个触发点的作用。

- **示例：**PWM 时钟设置为 2M，PWM 的周期设置为 1ms（即 PWMC 的值设置为 1999），在此周期中配置 PWM 占空比为 40%。

**方法 1：**第 1 触发点的数值设置为 0，第 2 触发点的数值设置为 1200

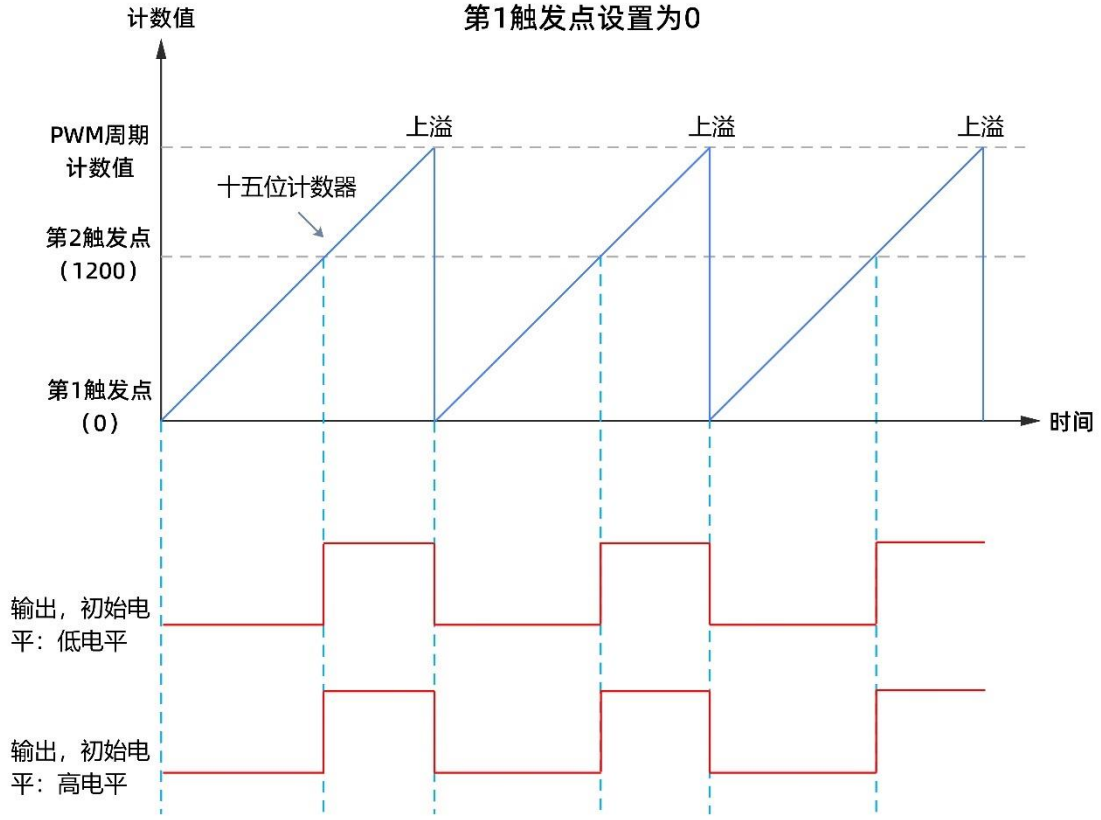



图 6：第 1 触发点的值设置为 0 时 PWM 输出波形

当计数器的值归零后，因为第 1 触发点的值设置的是 0，所以 PWM 输出低电平。计数值递增到 1200，即计数值等于第 2 触发点的值时，PWM 输出高电平。当计数器的值继续递增至 2000（PWMC）后，计数值归零，PWM 输出再次回到低电平，如此反复。即输出

一定占空比的波形。

**方法 2**：第 1 触发点的数值设置为 400，第 2 触发点的数值设置为 1600（当然，用其他的数值也可以的，只要第 2 触发点的数值减去第 1 触发点的数值占周期计数值的 60%即可）。

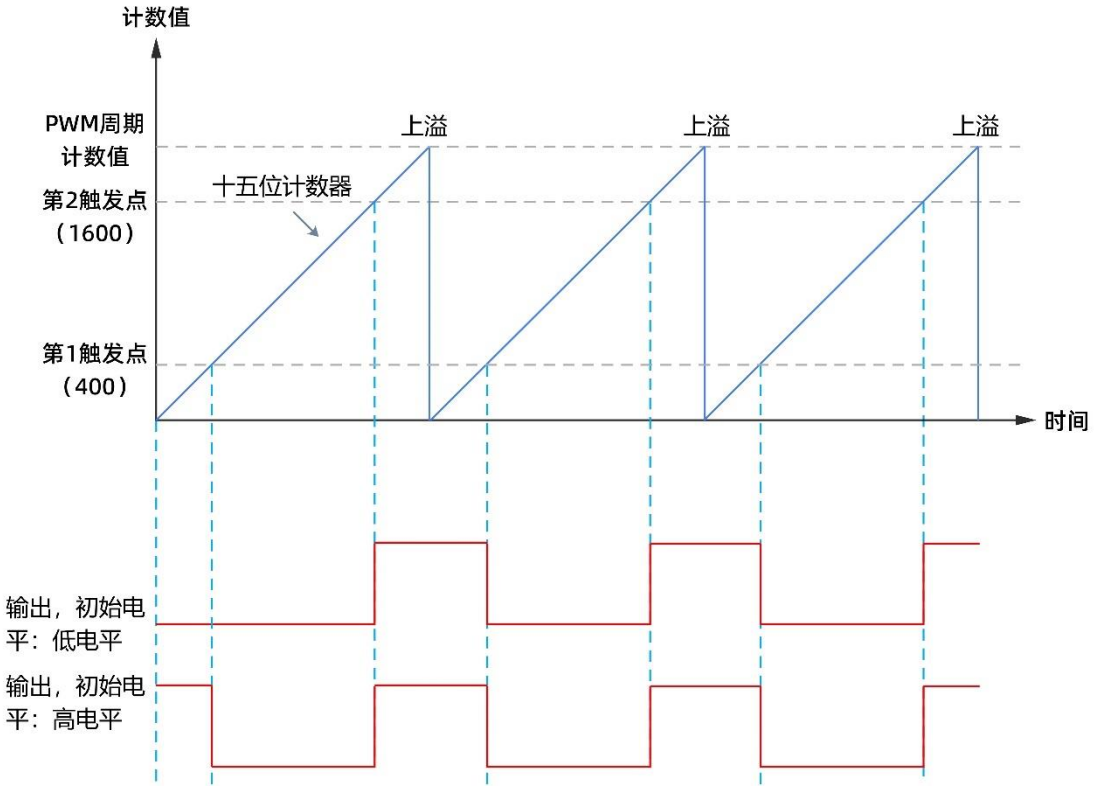



图 7：第 1 触发点的值设置不为 0 时 PWM 输出波形

由上面的示例可以看到，通过第 1 触发点和第 2 触发点可以很灵活的配置 PWM 占空比，读者在开发过程中，可以根据实际应用情况灵活使用。

### 3.4. PWM 中断

PWM 中断包含通道中断、归零中断和异常中断。

1. 通道中断

PWM 通道控制寄存器（PWMnCR）中的 ENT1I 和 ENT2I 位分别用于使能/关闭第 1 触发点中断和第 2 触发点中断，ENI 位用于使能/关闭通道中断。通道中断是针对 PWM 模块的通道的，也就是说 PWM 模块的 8 个通道均可单独配置通道中断。

2. 归零中断

PWM 配置寄存器（PWMCFG）中的 EPWMCBI 位用于使能/关闭计数器的归零中断。归零中断不是针对 PWM 通道，而是针对 PWM 计数器的，归零中断使能后，计数器计数到 PWMC 的数值后，计数值归零并产生中断，中断标志位需要软件清零。

3. 异常检测中断

异常检测中断由 PWM 异常检测控制寄存器（PWMnFDCR）配置，当发生 PWM 异常时，硬件自动产生中断跳转到相应中断入口执行中断服务程序，PWM 异常检测中断标志

位需软件清零。

✧ 注意：开启 PWM 中断的情况下，还需要开启总中断“EA=1”，PWM 中断才能起作用。

✧ 注：PWMnFDCR 的详细解释可阅读《STC8A8K64D4 系列单片机技术参考手册》的 19.2.4。

### 3.5. 启动 PWM

PWM 配置完成后，还需要通过增强型 PWM 配置寄存器（PWMCFG）中的 PWMCEN 位启动计数，PWM 才能输出波形。

| | | | | | | | | | 启动PWM计数 |
|---|---|---|---|---|---|---|---|---|---|
| 符号 | 地址 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| PWMCFG | F6H | - | - | - | - | PWMCBIF | EPWMCBI | ENPWMTA | PWMCEN |

- PWMCEN： PWM 波形发生器开始计数。
  - 0： PWM 停止计数
  - 1： PWM 计数器开始计数

PWMCEN 一旦被使能后，内部的 PWM 计数器会立即开始计数，并与 T1/T2 的值进行比较。所以 PWMCEN 必须在其他所有的 PWM 设置（包括 T1/T2 的设置、初始电平的设置、 PWM 异常检测的设置以及 PWM 中断设置）都完成后，最后才能使能 PWMCEN 位。

在 PWM 计数器计数的过程中， PWMCEN 控制位被关闭时，PWM 计数会立即停止，当再次使能 PWMCEN 控制位时，PWM 的计数会从 0 开始重新计数，而不会记忆 PWM 停止计数前的计数值。

✧ **特别注意**：当 PWMCEN 由 0 变为 1 时，内部的 PWM 计数器是从之前的不确定值归零后重新开始计数，所以此时会产生立即产生一个归零中断，当用户需要使用 PWM 的归零中断时，需特别注意这一点，即第一个归零中断并不是真正的 PWM 周期记满后归零所产生的。

## 4. 软件设计

### 4.1. PWM 输出波形

✧ 注：本节的实验是在“实验 2-1-3：流水灯（自编驱动文件方式）”的基础上修改，本节对应的实验源码是：“实验 2-15-1：PWM 输出波形”。

#### 4.1.1. 实验内容

本例中使用 PWM 通道 6 的 P2.6 引脚输出频率为 1KHz，占空比为 60%的波形。P2.6 在开发板上连接到了指示灯 D1，当我们改变占空比时，可以观察到 D1 亮度的变化，本例

的功能需求如下表所示。

表 2：功能需求表

| | |
|---|---|
| 使用的 PWM 通道 | PWM 通道 6。 |
| 输出引脚 | P2.6。 |
| PWM 时钟 | 系统时钟的 12 分频，系统时钟使用的是 24MHz，所以，PWM 时钟是 2MHz。 |
| PWM 频率 | 1KHz（周期 1ms）。<br>$PWM输出频率 = \frac{系统工作频率SYSclk}{(PWM\_PS + 1) \times ([PWMCH, PWMCL] + 1)}$<br>由上面的 PWM 频率计算公式可计算出 PWMC 的值应设置为 1999，对应的 16 进制为 0x07CF，即 PWMCH=0x07，PWMCL=0xCF。 |
| 占空比 | 60%。第 1 触发点设置为 0，第 2 触发点设置为 800。 |
| 初始电平 | 高电平。 |
| 中断配置 | 不使用中断。 |

✧ **说明**：读者可以通过修改 PWM 时钟和 PWMC 的值得到自己需要的 PWM 周期，通过修改触发点的值得到自己需要的占空比。

### 4.1.2. 代码编写

1. 新建一个名称为“pwm.c”的文件及其头文件“pwm.h”保存到工程的“Source”文件夹，并将“pwm.c”加入到 Keil 工程中的“SOURCE”组。该文件用于存放 I2C 硬件操作相关的函数。
2. 引用头文件

因为在“main.c”文件中使用了“pwm.c”文件中的函数，所以需要引用下面的头文件“pwm.h”。

**代码清单**：引用头文件

```
1.  //引用 PWM 的头文件
2.  #include    "pwm.h"
```

3. 配置 PWM

按照功能需求表配置 PWM，代码清单如下。

**代码清单**：配置 PWM

```
1.  /*************************************************************************************
2.  功能描述：初始化 PWM 模块。
3.            ：使能比较器上升沿和下降沿中断
4.            ：使用通道 6，选择 P2.6 为通道 6 的输出引脚，频率：1KHz，占空比 60%
```

```
5.  参    数：无
6.  返 回 值：无
7.  ******************************************************************************/
8.  void pwm_config(void)
9.  {
10.     PWMSET |= 0x01;              //使能 PWM 模块，必须先使能后面的配置才能生效
11.
12.     /*--------------------PWMCFG-----------------------------------
13.     位 7  位 6  位 5  位 4    位 3      位 2      位 1      位 0
14.     --   --   --   --   PWMCBIF  EPWMCBI  EPWMTA  PWMCEN
15.     x    x    x    x      0        0        0       0
16.     PWMCBIF：PWM 计数器归零中断标志位,需软件清零
17.     PEPWMCBI=0：使能 PWM 计数器归零中断
18.     EPWMTA = 0：PWM 与 ADC 不关联
19.     PWMCEN = 0：PWM 停止计数
20.     ------------------------------------------------------------*/
21.     PWMCFG = 0x00;
22.
23.     P_SW2 |= 0x80;               //将 EAXFR 位置 1，允许访问扩展 RAM 区特殊功能寄存器(XFR)
24.     /*--------------------PWMCKS-----------------------------------
25.     位 7  位 6  位 5   位 4     位 3~0
26.     --    --    --   SELT2   PWM_PS[3:0]
27.      1     0     x     0      1011
28.
29.     SELT2=0：PWM 时钟源为系统时钟经分频器分频之后的时钟
30.     PWM_PS[3:0]=1011：系统时钟的 1/12,(24/12)MHz = 2MHz
31.     ------------------------------------------------------------*/
32.     PWMCKS = 0x0B;
33.
34.     /*--------------------PWM6CR-----------------------------------
35.     位 7  位 6  位 5  位 4~3     位 2   位 1    位 0
36.     ENO  INI  --  C0_S[1:0]  ENI  ENT2I  ENT1I
37.      1    0    x     00       1     0      0
38.
39.     ENO = 1:PWM 的通道 6 相应 PWM 端口为 PWM 输出口
40.     INI = 0：PWM 的通道 6 初始电平为低电平
41.     C0_S[1:0] =00：选择 P2.6 为 PWM 通道 6 输出脚
42.     ENI = 1：使能 PWM 通道 6 的 PWM 中断
43.     ENT2I = 0：关闭 PWM 的通道 6 在第 2 个触发点中断
44.     ENT1I = 0：关闭 PWM 的通道 6 在第 1 个触发点中断
45.     ------------------------------------------------------------*/
46.     PWM6CR = 0xC0;
47.
```

```
48.     PWMC   = 1999;          //设置 PWM 周期为 1ms
49.     PWM6T1 = 0;             //第 1 触发点设置为 0
50.     PWM6T2 = 800;           //第 2 触发点设置为 800
51.     P_SW2 &= 0x7F;          //将 EAXFR 位置 0，禁止访问 XFR
52.
53.     PWMCFG |= 0x01;         //将 PWMCEN 位置 1，使能 PWM 波形发生器，PWM 计数器开始计数
54. }
```

4. 主函数

主函数中将 P2.6 配置为准双向口，之后调用 pwm_config()函数配置 PWM 即可，代码清单如下。

**代码清单：**主函数

```
1.  /*************************************************************************************
2.  功能描述：主函数
3.  参    数：无
4.  返 回 值：int 类型
5.  *************************************************************************************/
6.  int main(void)
7.  {
8.     P2M1 &= 0xBF;   P2M0 &= 0xBF;     //设置 P2.6 为准双向口（指示灯 D1）
9.
10.    pwm_config();                     //初始化 PWM
11.    while(1)
12.    {
13.       ;
14.    }
15. }
```

### 4.1.3. 硬件连接

本实验需要使用 LED 指示灯 D1，按照下图所示短接跳线帽。


图 8：跳线帽短接

### 4.1.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-15-1：PWM 输出波

形”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC8”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。

2) 双击“…\pwm\project”目录下的工程文件“pwm.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“pwm.hex”位于工程的“…\pwm\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：24MHz。
5) 电脑上打开串口调试助手，选择开发板对应的串口号，将波特率设置为 9600bps。
6) 程序运行后，可观察到指示灯 D1 点亮，用示波器测量 D1 驱动引脚 P2.6，可以观察到频率为 1KHz（周期为 1ms），占空比为 60%的 PWM 波行。
7) 读者可以尝试修改 PWM 周期和占空比，并用示波器观察波形，直观地观察修改后波形的变化。

## 4.2. PWM 实现呼吸灯

✧ 注：本节的实验是在“实验 2-15-1：PWM 输出波形”的基础上修改，本节对应的实验源码是：“实验 2-15-2：PWM 实现呼吸灯”。

### 4.2.1. 实验内容

本例中使用 PWM 驱动指示灯 D1、D2 实现呼吸灯。在 PWM 周期中，高电平时指示灯熄灭，低电平时指示灯点亮，因此，改变占空比即可改变指示灯点亮的时间，从而达到改变指示灯亮度的目的。程序中以步进值 2 递减/递增占空比，程序运行后可以连续地观察到指示灯光亮度逐渐增加，之后逐渐降低，即呼吸灯效果，本例的功能需求如下表所示。

表 3：功能需求表

| | |
|---|---|
| 使用的 PWM 通道 | PWM 通道 6 和 PWM 通道 7。 |
| 输出引脚 | P2.6（PWM 通道 6）、P2.7（PWM 通道 7）。 |
| PWM 时钟 | 系统时钟的 12 分频，系统时钟使用的是 24MHz，所以，PWM 时钟是 2MHz。 |
| PWM 频率 | 1KHz（周期 1ms）。<br>$PWM输出频率 = \frac{系统工作频率SYSclk}{(PWM\_PS + 1) \times ([PWMCH, PWMCL] + 1)}$<br>由上面的 PWM 频率计算公式可计算出 PWMC 的值应设置为 1999，对应的 16 进制为 0x07CF，即 PWMCH=0x07，PWMCL=0xCF。 |
| 占空比 | 第 1 触发点设置为 0，第 2 触发点的值从 1 开始以步进 2 递增，递增到周期计数值后以步进 2 递减，递减到到 1 后，再次以步进 2 递增，如此反复。 |
| 初始电平 | 高电平。 |
| 中断配置 | 使用 PWM 归零中断，中断服务函数中修改第 2 触发点的值（以步进 2 递增/递减）。 |

### 4.2.2. 代码编写

1. 初始化配置 PWM

按照功能需求表配置 PWM，代码清单如下。

**代码清单：**配置 PWM

```
/*******************************************************************************
功能描述：初始化 PWM 模块。
        ：使用通道 6（选择 P2.6 为通道 6 的输出引脚）和通道 7（选择 P2.7 为通道 7 的输出引脚）
        ：PWM 频率：1KHz
参    数：无
返 回 值：无
*******************************************************************************/
void pwm_config(void)
{
    PWMSET |= 0x01;                 //使能 PWM 模块，必须先使能后面的配置才能生效

    /*------------------- PWMCFG-------------------------------------
    位 7  位 6  位 5  位 4    位 3      位 2      位 1     位 0
    --   --   --   --   PWMCBIF  EPWMCBI  EPWMTA  PWMCEN
    x    x    x    x     0        1        0       0

    PWMCBIF：PWM 计数器归零中断标志位,需软件清零
    PEPWMCBI=1：使能 PWM 计数器归零中断
    EPWMTA = 0：PWM 与 ADC 不关联
    PWMCEN = 0：PWM 停止计数
    ------------------------------------------------------------*/
    PWMCFG = 0x04;

    P_SW2 |= 0x80;                  //将 EAXFR 位置 1，允许访问扩展 RAM 区特殊功能寄存器(XFR)
    /*--------------------- PWMCKS---------------------------------
    位 7  位 6  位 5   位 4     位 3~0
    --   --   --   SELT2  PWM_PS[3:0]
    1    0    x    0       0

    SELT2=0：PWM 时钟源为系统时钟经分频器分频之后的时钟
    PWM_PS[3:0]=1111：系统时钟的 1/16
    ------------------------------------------------------------*/
    PWMCKS = 0x0B;

    /*-------------------PWM6CR-------------------------------------
    位 7  位 6  位 5  位 4~3     位 2   位 1    位 0
    ENO  INI  --  C0_S[1:0]  ENI  ENT2I  ENT1I
    1    1    x    00       1    0      0
```

```
39.
40.     ENO = 1:PWM 的通道 6 相应 PWM 端口为 PWM 输出口
41.     INI = 1: PWM 的通道 6 初始电平为高电平
42.     C0_S[1:0] =00: 选择 P2.6 为 PWM 通道 6 输出脚
43.     ENI = 0: 关闭 PWM 通道 6 的 PWM 中断
44.     ENT2I = 0: 关闭 PWM 的通道 6 在第 2 个触发点中断
45.     ENT1I = 0: 关闭 PWM 的通道 6 在第 1 个触发点中断
46.     ------------------------------------------------------------*/
47.     PWM6CR = 0xC0;
48.     PWM7CR = 0xC0;              //通道 7 配置和通道 6 一样
49.     PWMC   = CYCLE_VALUE;       //设置 PWM 周期为 1ms
50.     PWM6T1 = 0;                 //通道 6 第 1 触发点设置为 0
51.     PWM6T2 = 0x0001;            //通道 6 第 2 触发点设置为 1
52.     PWM7T1 = 0;                 //通道 7 第 1 触发点设置为 0
53.     PWM7T2 = 0x0001;            //通道 7 第 2 触发点设置为 1
54.
55.     P_SW2 &= 0x7F;              //将 EAXFR 位置 0，禁止访问 XFR
56.     PWMCFG |= 0x01;             //将 PWMCEN 位置 1，使能 PWM 波形发生器，PWM 计数器开始计数
57. }
```

2. PWM 中断服务程序

PWM 中断服务程序中修改第 2 触发点的值，也就是修改占空比。这里用了一个变量“dir”来标志递增和递减。第 2 触发点的值从 1 开始以步进 2 递增，递增到周期计数值后以步进 2 递减，递减到到 1 后，再次以步进 2 递增，代码清单如下。

**代码清单：**I2C 初始化

```
1.  /*********************************************************************
2.   * 描  述 : PWM 中断服务函数
3.   * 入  参 : 无
4.   * 返回值 : 无
5.   *********************************************************************/
6.  void pwm_isr() interrupt 22 using 1
7.  {
8.     static bit dir = 1;
9.     static u16 t2_val = 0;
10.
11.    if(PWMCFG & 0x08)
12.    {
13.       PWMCFG &= ~0x08; //清归零中断标志
14.       if (dir)         //第 2 触发点的值以步进值 2 递增
15.       {
16.          t2_val += DUTY_CYCLE_STEP;
17.          if (t2_val >= CYCLE_VALUE) dir = 0;
18.       }
```

```
19.       else            //第 2 触发点的值以步进值 2 递减
20.       {
21.          t2_val -= DUTY_CYCLE_STEP;
22.          if (t2_val <= DUTY_CYCLE_STEP) dir = 1;
23.       }
24.       P_SW2 |= 0x80;        //将 EAXSFR 位置 1，以访问 PWM 在扩展 RAM 区的特殊功能寄存器
25.       PWM6T2 = t2_val;       //更新 PWM 通道 6 第 2 触发点的值
26.       PWM7T2 = t2_val;       //更新 PWM 通道 7 第 2 触发点的值
27.       P_SW2 &= 0x7F;          //将 EAXSFR 位置 0，恢复访问 XRAM
28.    }
29. }
```

3. 主函数

主函数中将 P2.6、P2.7 配置为准双向口，之后调用 pwm_config()函数配置 PWM 即可，代码清单如下。

**代码清单：**I2C 初始化

```
1.  /*************************************************************************************
2.  功能描述：主函数
3.  参    数：无
4.  返 回 值：int 类型
5.  *************************************************************************************/
6.  int main(void)
7.  {
8.     P2M1 &= 0x3F; P2M0 &= 0x3F;    //设置 P2.6、P2.7 为准双向口（指示灯 D1 和 D2）
9.     pwm_config();                 //初始化 PWM
10.    EA = 1;                       //使能总中断
11.    while(1)
12.    {
13.       ;
14.    }
15. }
```

### 4.2.3. 硬件连接

本实验需要使用 LED 指示灯 D1 和 D2，按照下图所示短接跳线帽。


图 9：跳线帽短接

#### 4.2.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-15-2：PWM 实现呼吸灯”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC8”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\pwm_led\project”目录下的工程文件“pwm_led.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“pwm_led.hex”位于工程的“…\pwm_led\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：24MHz。
5) 电脑上打开串口调试助手，选择开发板对应的串口号，将波特率设置为 9600bps。
6) 程序运行后，可连续观察到指示灯 D1、D2 的亮度逐步增加和降低，即呼吸灯效果。

### 4.3. PWM 驱动震动马达

✧ 注：本节的实验是在“实验 2-15-1：PWM 输出波形”的基础上修改，本节对应的实验源码是：“实验 2-15-3：PWM 驱动震动马达”。

#### 4.3.1. 实验内容

本例中通过 PWM 输出不同占空比的波形控制震动马达的震动强度。例中使用 PWM 模块的通道 6，选择 P1.6 为其输出引脚，用于驱动震动马达。

P1.6 以 3 秒的间隔轮流输出 40%占空比和 90%占空比的 PWM 波形，以便于观察不同占空比时马达的震动强度，本例的功能需求如下表所示。

表 4：功能需求表

| | |
|---|---|
| 使用的 PWM 通道 | PWM 通道 6。 |
| 输出引脚 | P1.6（PWM 通道 6）。 |
| PWM 时钟 | 系统时钟的 12 分频，系统时钟使用的是 24MHz，所以，PWM 时钟是 2MHz。 |
| PWM 频率 | 1KHz（周期 1ms）。<br>$PWM输出频率 = \frac{系统工作频率SYSclk}{(PWM\_PS + 1) \times ([PWMCH, PWMCL] + 1)}$<br>由上面的 PWM 频率计算公式可计算出 PWMC 的值应设置为 1999，对应的 16 进制为 0x07CF，即 PWMCH=0x07，PWMCL=0xCF。 |
| 占空比 | 轮流间隔 3 秒使用 40%占空比和 90%占空比的 PWM 波形分别驱动震动马达，以观察不同占空比的波形对马达震动强度的影响。 |
| 初始电平 | 低电平，因为振动马达是高电平驱动，因此引脚初始电平输出低电平。 |
| 中断配置 | 不使用中断。 |

✧ 注：本节的实验需要使用艾克姆科技的震动马达模块。

### 4.3.2. 震动马达模块

艾克姆震动马达模块使用的是 1027 震动马达，其中 10 指的是马达的直径，27 指的是马达的厚度，单位是 mm。模块上板载了 MOS 管驱动电路，可直接通过单片机 GPIO 驱动。模块规格如下图所示。

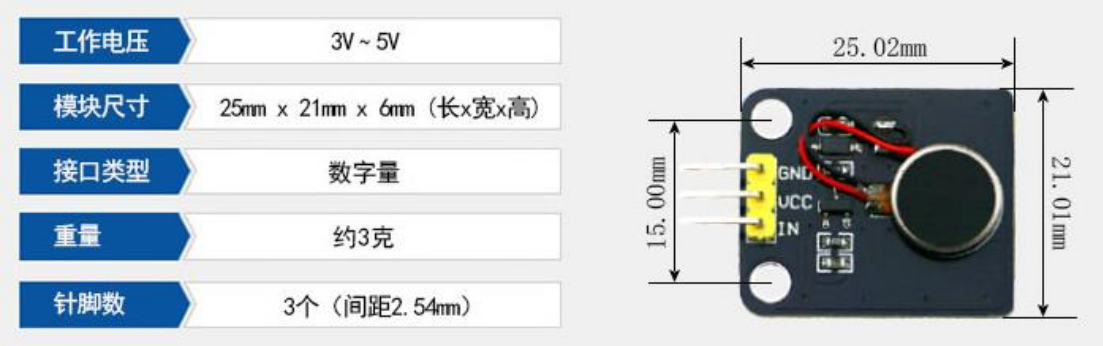



图 10：震动马达模块规格

震动马达模块的引脚如下图所示，IN 脚输入高电平，马达震动，IN 脚输入低电平，马达停止震动。因此，可以通过 PWM 输出不同的占空比的信号给 IN 脚，控制 IN 脚高电平的占比，从而控制马达的震动强度。


图 11：震动马达引脚

### 4.3.3. 代码编写

1. 初始化配置 PWM

按照功能需求表配置 PWM，代码清单如下。

**代码清单：**配置 PWM

```
1.  /*************************************************************************************
2.  功能描述：初始化 PWM 模块。
3.            ：使用通道 6，选择 P1.6 为通道 6 的输出引脚，频率：1KHz，占空比 90%
4.  参      数：无
5.  返 回 值：无
6.  *************************************************************************************/
```

```
void pwm_config(void)
{
    PWMSET |= 0x01;                 //使能 PWM 模块，必须先使能后面的配置才能生效

    /*---------------------PWMCFG-----------------------------------
    位 7  位 6  位 5  位 4    位 3      位 2      位 1     位 0
    --   --   --   --   PWMCBIF  EPWMCBI  EPWMTA  PWMCEN
    x    x    x    x     0        0        0       0
    PWMCBIF：PWM 计数器归零中断标志位,需软件清零
    PEPWMCBI=0：使能 PWM 计数器归零中断
    EPWMTA = 0：PWM 与 ADC 不关联
    PWMCEN = 0：PWM 停止计数
    ----------------------------------------------------------------*/
    PWMCFG = 0x00;

    P_SW2 |= 0x80;                  //将 EAXFR 位置 1，允许访问扩展 RAM 区特殊功能寄存器(XFR)
    /*---------------------PWMCKS-----------------------------------
    位 7  位 6  位 5   位 4     位 3~0
    --   --   --   SELT2  PWM_PS[3:0]
    1    0    x    0        1011

    SELT2=0：PWM 时钟源为系统时钟经分频器分频之后的时钟
    PWM_PS[3:0]=1011：系统时钟的 1/12,(24/12)MHz = 2MHz
    ----------------------------------------------------------------*/
    PWMCKS = 0x0B;

    /*---------------------PWM6CR-----------------------------------
    位 7  位 6  位 5  位 4~3     位 2   位 1    位 0
    ENO  INI  --  C0_S[1:0]  ENI  ENT2I  ENT1I
     1    0    x    01       0    0      0

    ENO = 1:PWM 的通道 6 相应 PWM 端口为 PWM 输出口
    INI = 0：PWM 的通道 6 初始电平为低电平
    C0_S[1:0] =10：选择 P1.6 为 PWM 通道 6 输出脚
    ENI = 0：使能 PWM 通道 6 的 PWM 中断
    ENT2I = 0：关闭 PWM 的通道 6 在第 2 个触发点中断
    ENT1I = 0：关闭 PWM 的通道 6 在第 1 个触发点中断
    ----------------------------------------------------------------*/
    PWM6CR = 0x88;

    PWMC   = 1999;          //设置 PWM 周期为 1ms
    PWM6T1 = 0;             //第 1 触发点设置为 0
    PWM6T2 = 200;           //第 2 触发点设置为 200,此时，占空比为 90%
```

```
    P_SW2 &= 0x7F;          //将 EAXFR 位置 0，禁止访问 XFR
    PWMCFG |= 0x01;         //将 PWMCEN 位置 1，使能 PWM 波形发生器，PWM 计数器开始计数
}
```

2. 主函数

主函数中将 P1.6 配置为准双向口，并输出低电平（震动马达是高电平有效，初始化时关闭其震动），之后调用 pwm_config()函数初始化配置 PWM。主循环中以 3 秒的间隔轮流输出 40%占空比和 90%占空比的 PWM 波形，代码清单如下。

**代码清单：**主函数

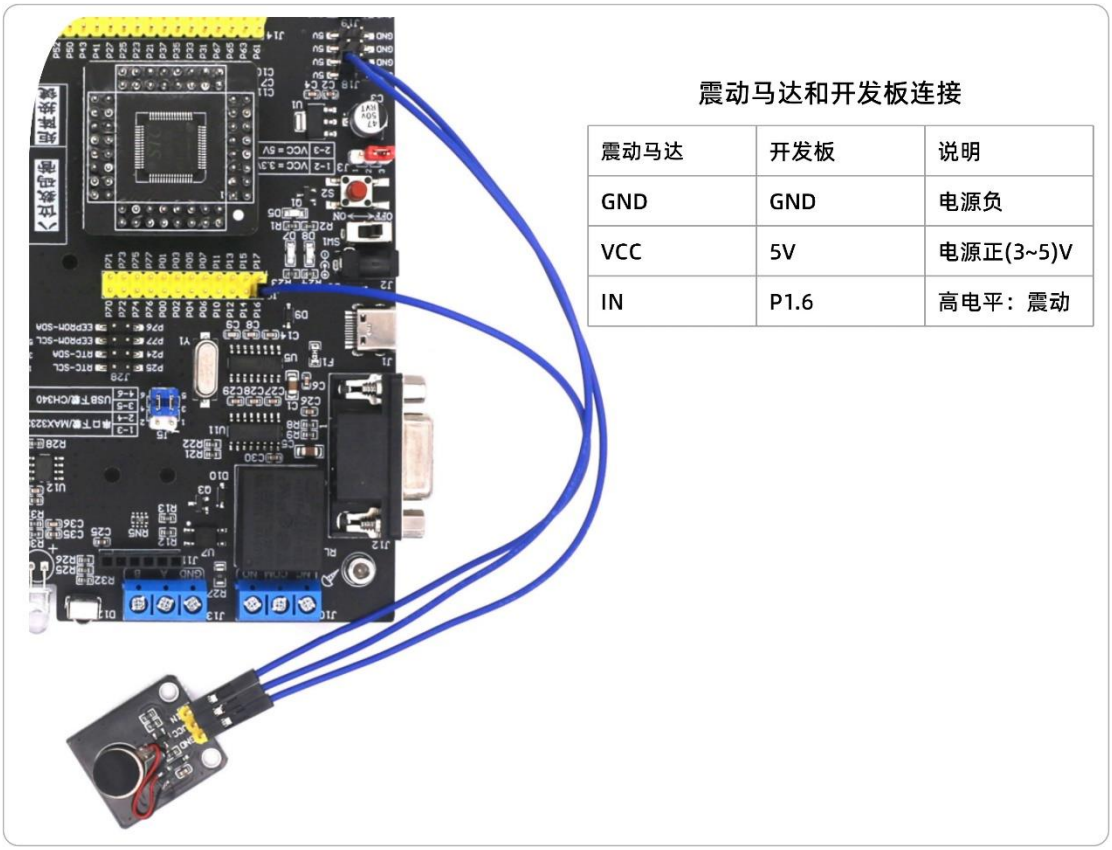
```
/*******************************************************************************
功能描述：主函数
参    数：无
返 回 值：int 类型
*******************************************************************************/
int main(void)
{
    P1M1 &= 0xBF;   P1M0 &= 0xBF;     //设置 P1.6 为准双向口
    P16 = 0;                          //因为震动马达是高电平驱动，P16 输出低电平

    pwm_config();                     //初始化 PWM
    while(1)
    {
        delay_ms(3000);         //延时 3 秒
        P_SW2 |= 0x80;          //将 EAXFR 位置 1，允许访问扩展 RAM 区特殊功能寄存器(XFR)
        PWM6T2 = 1200;          //第 2 触发点设置为 1200,此时，占空比为 40%
        P_SW2 &= 0x7F;          //将 EAXFR 位置 0，禁止访问 XFR
        delay_ms(3000);
        P_SW2 |= 0x80;          //将 EAXFR 位置 1，允许访问扩展 RAM 区特殊功能寄存器(XFR)
        PWM6T2 = 200;           //第 2 触发点设置为 200,此时，占空比为 90%
        P_SW2 &= 0x7F;          //将 EAXFR 位置 0，禁止访问 XFR
    }
}
```

### 4.3.4. 硬件连接

本实验需要用到艾克姆科技的振动马达模块，模块和开发板之间的连接如下图所示。

✧ 注意：P1.6 和矩阵按键共用，因此，震动马达模块的 IN 脚连接到 P1.6 后，不能使用矩阵按键。



震动马达和开发板连接

| 震动马达 | 开发板 | 说明 |
|---|---|---|
| GND | GND | 电源负 |
| VCC | 5V | 电源正(3~5)V |
| IN | P1.6 | 高电平：震动 |

图 12：震动马达模块安装

### 4.3.5. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-15-3：PWM 驱动震动马达”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC8”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\pwm_motor\project”目录下的工程文件“pwm_motor.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“pwm_motor.hex”位于工程的“…\pwm_motor\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：24MHz。
5) 电脑上打开串口调试助手，选择开发板对应的串口号，将波特率设置为 9600bps。
6) 程序运行后，可观察到震动马达连续 3 秒震动强度较大（90%占空比），接着连续 3 秒震动强度较小（40%占空比），如此反复。