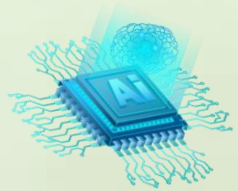
第8章 定时器、PWM与输入捕捉

定时器是微型计算机中的重要模块之一，它也是操作系统的运行基础。同时，现在的微型计算中的常用定时器模块，不仅具备基本计时功能，还还含有脉宽调制及输入捕捉功能。基本定时基于中断方式进行时间记录，脉宽调制是基于程序控制电机转动的重要基础，输入捕捉可以获得比较精确的时刻。



8.1 定时器

在嵌入式应用系统中，有时要求能对外部脉冲信号或开关信号进行计数，这可通过计数器来完成。有些设备要求每间隔一定时间开启并在一段时间后关闭，有些指示灯要求不断地闪烁，这可利用定时信号来完成。另外系统日历时钟、产生不同频率的声源等也需要定时信号。

计数与计时的功能是一致的，统一称为定时器。

8.1.1 定时器的基本含义

中国古代的主要计时方式与现代的对应：

十二地支来表示，以晚上十一点为子时、凌晨一点丑时、凌晨三点寅时、早晨五点卯时、上午七点辰时、上午九点巳时、中午十一点午时、下午一点未时、下午三点申时、傍晚五点酉时、晚上七点戌时、晚上九点亥时。

计时误差很大

沙漏计时等

实现计数与定时的基本方法有三种：完全硬件方式、完全软件方式、可编程计数器/定时器。

完全硬件方式基于逻辑电路实现，现已很少使用。

完全软件方式是利用计算机执行指令的时间实现定时，但这种方式占用CPU，不适用于多任务环境，一般仅用于时间极短的延时且重复次数较少的情况。

可编程定时器，它在设定之后，与CPU并行地工作，不占用CPU的工作时间。这种方法的主要思想是根据需要的定时时间，用指令对定时器设置定时常数，并用指令启动定时器开始计数，当计数到指定值时，便自动产生一个定时输出，或中断信号告知CPU。

8.1.2 CH573内部的内核中系统定时器SysTick

青稞V4F内核中包含了一个简单的定时器SysTick，又称为“滴答”定时器。这个定时器由于是包含在内核中，凡是使用该内核生产的MCU均含有SysTick，因此使用这个定时器的程序方便在MCU间移植。若使用实时操作系统，一般可用该定时器作为操作系统的时间滴答，可简化实时操作系统在以RISC-V为内核的MCU间移植工作。

1. SysTick定时器的寄存器

1）SysTick定时器的寄存器地址

SysTick定时器中有6个32位寄存器，基地址为：0xE000F000

表8-1　Systick定时器的寄存器偏移地址及简明功能

偏移地址	寄存器名	简称	简明功能
0x00	系统计数控制寄存器	CTLR	配置功能及状态标志
0x04	系统计数器低位寄存器	CNTL	当前计数器计数值低 32 位
0x08	系统计数器高位寄存器	CNTH	当前计数器计数值高 32 位
0x0C	计数比较低位寄存器	CMPLR	设置比较计数器值低 32 位
0x10	计数比较高位寄存器	CMPHR	设置比较计数器值高 32 位
0x14	计数器计数标志寄存器	CNTFG	1：开中断，0：关中断

2）控制及状态寄存器

控制及状态寄存器的4个位有实际含义

表8-2 系统计数控制寄存器（CTLR）相关位

位	英文含义	中文含义	R/W	功能说明
8	STRELOAD	重装载控制	W1	写 1 将计数重加载寄存器(64 位)数值更新到当前计数器寄存器中
2	STCLK	计数时钟源选择	R/W	1：HCLK 做时基 0：HCLK/8 做计数时基
1	STIE	计数器中断使能控制位	R/W	1：使能计数器中断 0：无计数器中断
0	STE	系统计数器使能控制位	R/W	1：启动系统计数器 STK 0：关闭系统计数器 STK，计数器停止计数

2）控制及状态寄存器

控制及状态寄存器的4个位有实际含义

表8-2 系统计数控制寄存器（CTLR）相关位

位	英文含义	中文含义	R/W	功能说明
8	STRELOAD	重装载控制	W1	写 1 将计数重加载寄存器(64 位)数值更新到当前计数器寄存器中
2	STCLK	计数时钟源选择	R/W	1：HCLK 做时基 0：HCLK/8 做计数时基
1	STIE	计数器中断使能控制位	R/W	1：使能计数器中断 0：无计数器中断
0	STE	系统计数器使能控制位	R/W	1：启动系统计数器 STK 0：关闭系统计数器 STK，计数器停止计数

3）64位系统计数器寄存器（CNT）、64位计数比较寄存器（CMPR）和计数器计数标志寄存器（CNTFG）

系统主频时钟HCLK默认的频率为60MHZ，系统计数寄存器会从计数 比较寄存器的值开始进行减1计数，打开系统时钟中断后，当计数器的值减为0时，触发Systick中断，中断向量号为12，对计数器计数标志寄存器CNTFG的第1位写0则清除对应的中断标志。

2．Systick构件制作过程

Systick构件是一个最简单的构件，只包含一个初始化函数，一个清中断函数。Systick的清中断需要将计数器计数标志寄存器CNTFG的计数器减为0标志CNTIF位置0即可，要设计Systick初始化函数systick_init，分为三步。

1）梳理初始化流程 （重点）

青稞V4芯片中的SysTick是一个64位减1计数器，采用减1计数的方式工作计数到0，产生SysTick异常（中断），中断号为12，中断优先级可设置。初始化时，选择时钟源（决定了计数频率）、设置计数比较寄存器（决定了计数周期）、打开计数器中断、开启计数器。由此，该定时器开始工作，计数器采用减1计数，计数器的初值为计数比较寄存器中的值，计数到0，计数器计数标志寄存器CNTFG的计数器减为0标志CNTIF会被自动置1，产生中断请求，系统自动将比较寄存器的值更新到计数器寄存器中。

2）确定初始化参数及其范围

需要的参数：首先是确定时钟源，它决定了计数频率，本书使用的CH573芯片，编程时将SysTick的时钟源设置为内核时钟，不做传入参数；其次，由于当计数器（CNT）减到0时或加到比较值会产生SysTick中断，因此应确定SysTick中断时间间隔，单位一般为毫秒（ms）。这样，SysTick初始化函数只有一个参数：中断时间间隔。

范围：设时钟频率为f，计数器有效位数为n，则中断时间间隔的范围为τ=1~（$2^n/f$）*1000(ms)。

3）Systick构件API

Systick构件API在Systick.h文件中。

（参看源码工程SysTick-ASM-CH573）

4）Systick构件源文件

Systick构件的源码在Systick.c文件中。

（参看源码工程SysTick-ASM-CH573）

8.2 基于构件的SysTick定时器中断编程举例

本节CH573的内核定时器SysTick为例，阐述定时器中断编程方法，样例程序参见电子资源【03-Software\CH08\SysTick-ASM】。

1. 确定中断号与中断服务例程名称

查询表6-4（CH573中断源），CH573中断源可知，SysTick的中断向量号为12，中断优先级默认为0，为可设置，缺省的SysTick中断入口函数名称为SystTick_Handler，这个中断服务例程名称，符合实际需要，不需要进行重新宏定义。所以在编写中断服务例程时，直接使用。

2. SysTick构件使用方法

程序功能：设每隔10ms产生一次Systick中断，当时间秒数发生变化时利用printf函数打印输出时分秒。编程时，调用Systick构件中的systick_init 函数进行SysTick定时器初始化，开放SysTick中断，在SysTick定时器中断服务例程SystTick_Handler中进行计时。

```
li a0, 10                    /* 设置 10 毫秒 */
call   systick_init          /* 调用 systick_init 函数 */
```

3. 中断服务例程的编写

```
//=====================================================
//中断服务例程名称：Systick_Handler
//功能概要：SysTick定时器中断服务例程。SysTick定时器每10ms中断
//          一次，触发本程序运行。功能为：当1秒到达时，调用
//          SecAdd1函数进行时分秒计时，对全局变量数组gtime进行
//          更新。地址gtime存放时，地址gtime+1存放分，
//          地址gtime+2存放秒
//中断触发条件：SysTick定时器每10ms触发一次
//=====================================================
```

课堂练习：一步一步练习编写秒加1子函数SecAdd1

当中断服务例程中到达一秒时，为了记录时分秒，可以编制一个汇编子函数SecAdd1，实现时分秒计时。测试程序可以使用类似于高级语言的字节型全局变量数组gtime[0]、gtime[1]、gtime[2]分别存放时、分、秒，在汇编语言中，gtime是首地址，偏移0、1、2分别存放时、分、秒。编写汇编子函数SecAdd1，有以下几点思考。

（1）**不使用内存变量**。既然是汇编子函数，建议不使用全局变量，使用a0作为存放存放时、分、秒的内存地址入口，这样调用时，可以将gtime地址赋给a0。

（2）**不使用除法**。考虑到运行效率及普适性，不使用除法进行基于秒计算机时、分，而是每次调用该子函数进行秒加1，当秒大于等于60，进行分加1计算，后面类似。

（3）**使用秒加1后未到60秒（1分钟）时向后跳转到单出口**。这样既可以避免向前跳转，影响程序阅读，也条件跳转的嵌套。

可以根据这段文字，一步一步的编程调试，完成秒加1子函数SecAdd1的编写，提高底层编程的基本功。

8.3 PWM

脉宽调制（Pulse Width Modulator，PWM）是一种可以通过软件编程方式，从芯片引脚输出高低电平持续时间可调整的周期性信号，常用于电机的变频控制、灯光的细分亮暗控制等。

8.3.1 脉宽调制PWM通用基础知识

1. PWM知识要素

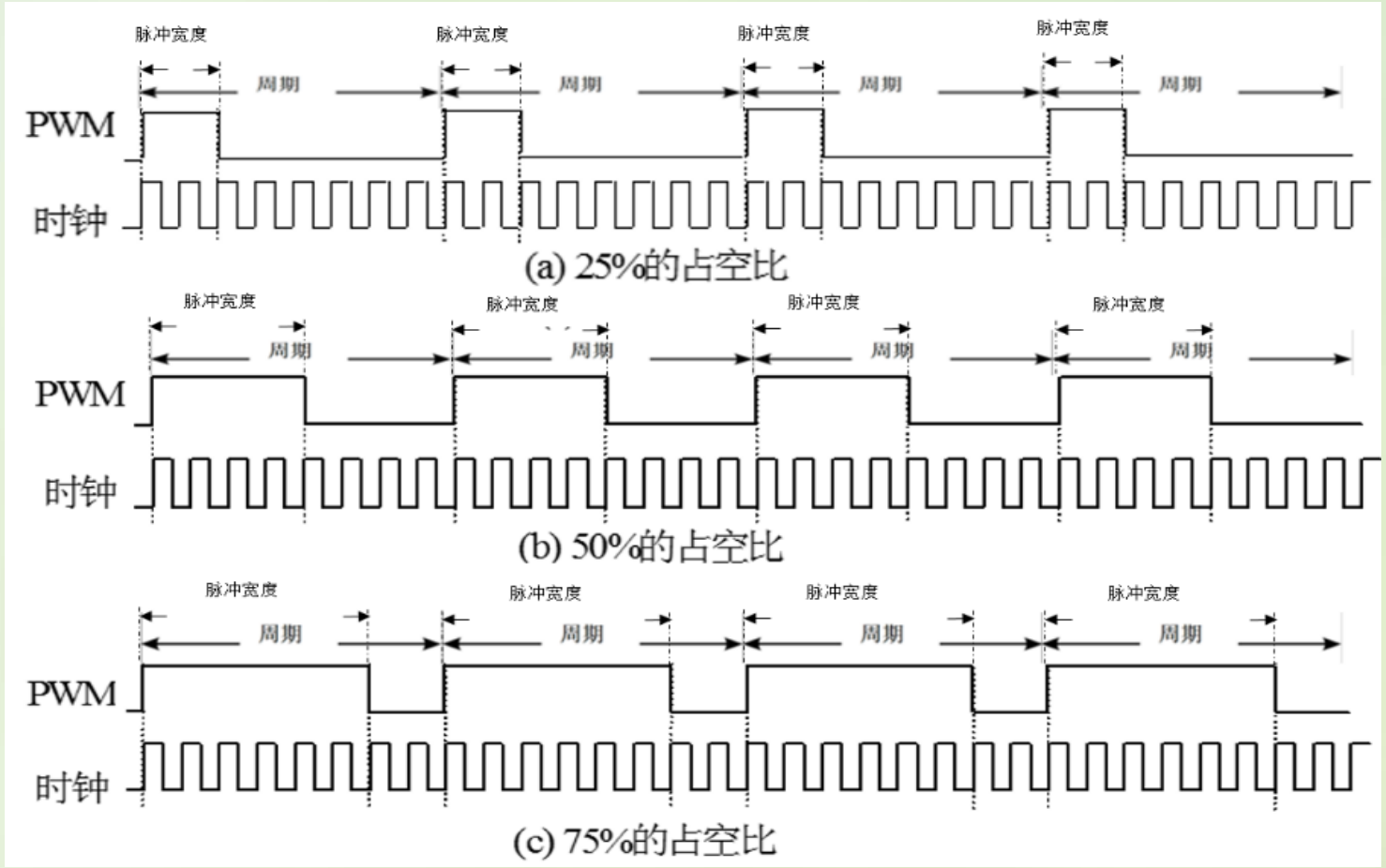
PWM信号的主要技术指标有：PWM时钟源频率、PWM周期、占空比、脉冲宽度与分辨率、极性与对齐方式等。

1）时钟源频率、PWM周期与占空比

利用MCU编程方式产生PWM波的实例，这个方法需要有一个产生**PWM波的时钟源**，其频率记为F_{CLK}，单位为Hz，相应时钟周期为$T_{CLK}=1/F_{CLK}$，单位为秒。

PWM周期用其有效电平持续的时钟周期个数来度量，记为N_{PWM}

PWM占空比被定义为PWM信号处于有效电平的时钟周期数与整个PWM周期内的时钟周期数之比，用百分比表征



2）脉冲宽度与分辨率

脉冲宽度是指一个PWM周期内，PWM波处于有效电平的时间（用持续的时钟周期数表征）。

PWM分辨率△T是指脉冲宽度的最小时间增量，等于时钟源周期，$\triangle T=T_{CLK}$

3）极性

PWM极性决定了PWM波的有效电平。正极性表示PWM有效电平为高电平，负极性表示PWM有效电平为低电平。

4）对齐方式（不作要求）

2．PWM的应用场合

PWM的最常见的应用是电机控制。还有一些其他用途，这里举几例。

（1）利用PWM为其他设备产生类似于时钟的信号。例如，PWM可用来控制灯以一定频率闪烁。

（2）利用PWM控制输入到某个设备的平均电流或电压。在一定程度上可以替代D/A转换。

（3）利用PWM控制命令字编码。例如，通过发送不同宽度的脉冲，代表不同含义。

8.3.2 CH573的PWM

1. TIMER中关于PWM的寄存器（略）

PWM信号的主要技术指标有：PWM时钟源频率、PWM周期、占空比、脉冲宽度与分辨率、极性与对齐方式等。

2. 关于PWM的构件制作过程

1）梳理初始化流程 （了解）
2）确定初始化参数及其范围 （了解）

3）PWM头文件

```
//=========================================================================
// 函数名称：pwm_init
// 功能概要：pwm 初始化函数
// 参数说明：pwmNo：通道号，使用.h 文件中的宏常数
//           clockFre：时钟频率，单位：hz
//                     定时器 PWM 选 2-60M；专用 PWM 选：235294-60M
//           period：周期，单位为个数，即计数器跳动次数，
//                   定时器 PWM：可选不大于时钟频率的任何正整数；
//                   专用 PWM：只能取 31、32、63、64、127、128、255、256
//           duty：占空比：0.0-100.0 对应 0%-100%
//           align：对齐方式 ，本构件无，取 0
//           pol：极性，在头文件宏定义给出，如 PWM_PLUS 为正极性
//函数返回：无
//使用说明：使用时注意，专用 PWM 的时钟频率和周期是所有通道共用的，
//          一个通道不能独立设置，但占空比和极性可以独立设置
//=========================================================================
void pwm_init(uint16_t pwmNo,uint32_t clockFre,uint32_t period,
              uint8_t  duty,uint8_t align, uint8_t pol);
```

```
//=====================================================================
// 函数名称：pwm_update
// 功能概要：PWM 更新
// 参数说明：pwmNo：通道号，使用.h 文件中的宏常数
//           duty：占空比：0.0-100.0 对应 0%-100%
// 函数返回：无
//=====================================================================
void pwm_update(uint16_t pwmNo,uint8_t duty);
```

4）PWM构件源文件（略）

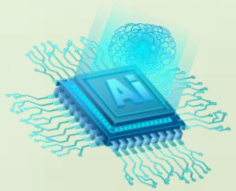
3. PWM构件测试工程（实际运行）

8.4 输入捕捉

讲座性质，不做要求

实验五：理解中断与定时器

在学院网站上传实验报告及实验程序，注意注释规范。



本章作业：1、2、3、6