PCA 可编程计数器阵列

1. 实验目的

- 掌握 STC15W4K32S4 系列 PCA 可编程计数器阵列的原理。
- 掌握 2 个 PCA 外设相关寄存器配置及程序设计。

2. 实验内容

- 编写程序实现 PCA0 外部脉冲输入捕获的程序设计。
- 编写程序实现 PCA0 定时器的程序设计。
- 编写程序实现 PCA0 和 PCA1 高速脉冲输出的程序设计。
- 编写程序实现 PCA0 和 PCA1 输出 PWM 的程序设计。

3. 硬件设计

3.1. 传感器输出信号介绍

传感器是一种能把物理量或化学量转变成便于处理的电信号的器件。国际电工委员会（IEC:International Electrotechnical ComMIttee）的定义为：“传感器是测量系统中的一种前置部件，它将输入变量转换成可供测量的信号”。简而言之，传感器具有将非电信号转成电信号的功能，传感器原理我们不关注，我们重点关注常用传感器外接信号的类型。

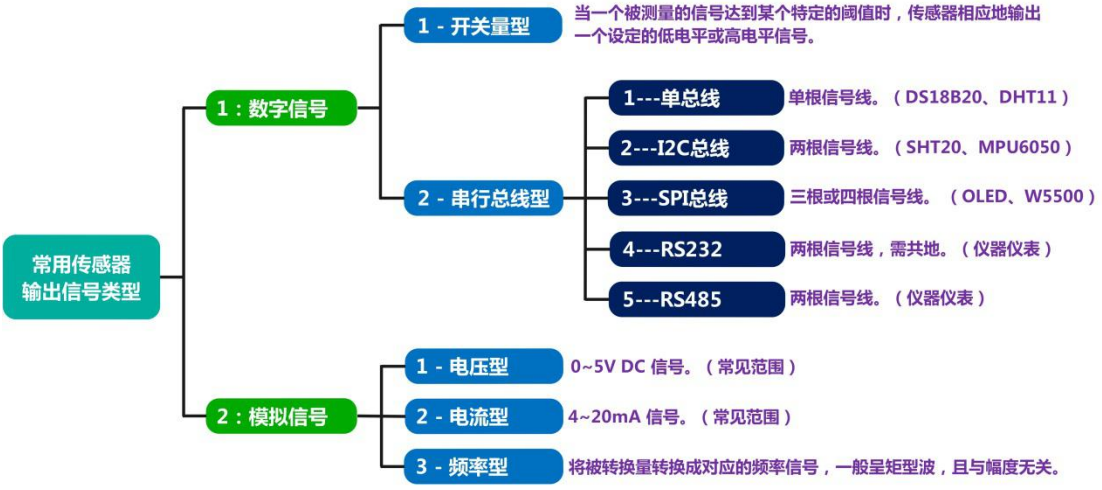


图 1：常用传感器输出信号类型示意图

脉冲信号是一种离散信号，形状多种多样，与普通模拟信号（如正弦波）相比，波形之间在时间轴不连续（波形与波形之间有明显的间隔）但具有一定的周期性是他的特点。最常见的脉冲波是矩形波（也就是方波）。脉冲信号可以用来表示信息，也可以用来作为载波，比如脉冲调制中的脉冲编码调制（PCM），脉冲宽度调制（PWM）等等，还可以作为各种数字电路、高性能芯片的时钟信号。

3.2. PCA 可编程计数器阵列介绍

PCA (全称是 Programmable Counter Array)可编程计数器阵列是 STC 单片机内部集成的外设，很多场合是以 CCP/PCA 放在一起来描述，那么这里的 CCP 又代表什么意思呢？CCP 是 Capture（捕获）、Compare（比较）、PWM（脉宽调制）的简称，从这个简称中我们进一步阐述下每一路 CCP/PCA 都可通过配置 CCP/PCA 相关寄存器使其工作在 4 种工作模式：上升/下降沿捕获、软件定时器、高速脉冲输出和可调脉冲输出。

基于 STC 单片机 CCP/PCA 的 4 种工作模式，其应用及优势可用下表表示。

表 1：单片机 CCP/PCA 应用

序号	工作模式	功能描述	备注
1	上升/下降沿捕获	对输入信号的跳变情况进行采样	
2	软件定时器	扩充了单片机定时器资源	
3	高速脉冲输出	可用于输出频率值比较高的脉冲信号的场合	
4	可调脉冲输出	扩充了单片机 PWM 资源	

✧ 注：STC 不同型号的单片机拥有的 CCP/PCA 资源不同，有的单片机有 3 路 CCP/PCA，有的单片机有 2 路 CCP/PCA，在使用时请注意查看。CCP/PCA 和 PCA 意思是相同的，下文均以 PCA 来简称 STC 单片机这个外设。

3.3. STC15W4K32S4 系列单片机 PCA 介绍

STC15W4K32S4 系列单片机集成了 2 路可编程计数器阵列 PCA0 和 PCA1，该 PCA 模块包含了一个 16 位的定时/计数器，供 2 路相互独立的 PCA 使用。

STC15W4K32S4 系列单片机每一路 PCA 都有 3 个 IO 引脚供选择使用，如下表。

表 2：单片机 PCA 引脚分配

PCAx	对应 IO 口	功能描述	说明	备注
PCA0	P1.1	PCA0 引脚	非独立 GPIO	与 P1.0 同时被选择
PCA0_2	P3.5	PCA0 引脚	非独立 GPIO	与 P3.6 同时被选择
PCA0_3	P2.5	PCA0 引脚	非独立 GPIO	与 P2.6 同时被选择
PCA1	P1.0	PCA1 引脚	非独立 GPIO	与 P1.1 同时被选择
PCA1_2	P3.6	PCA1 引脚	非独立 GPIO	与 P3.5 同时被选择
PCA1_3	P2.6	PCA1 引脚	非独立 GPIO	与 P2.5 同时被选择

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。针对非独立 GPIO 使用时需特别注意。

STC15W4K32S4 系列单片机 PCA 外设的理解首先要对 PCA 计数器的内部结构框图进行解析，下面给出该结构图。

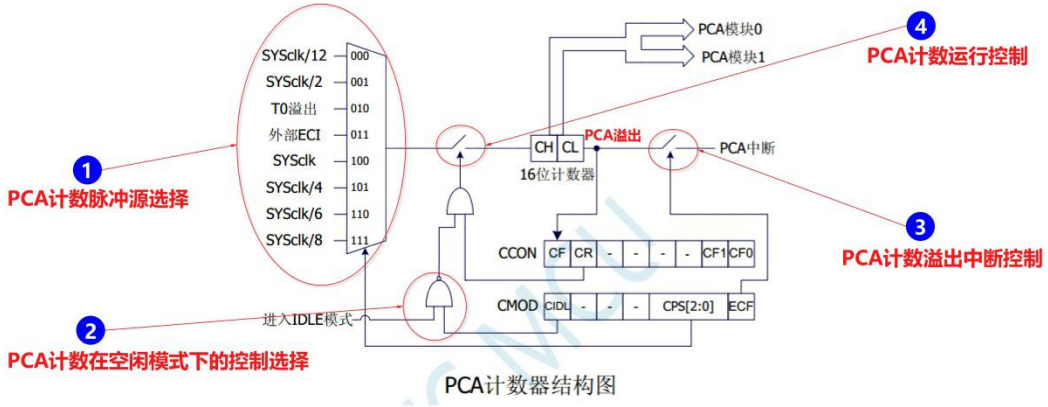


图 2：PCA 计数器内部结构示意图

✧ 注：从该图可知，PCA 计数器是供 2 路 PCA 共用的，虽然 2 路 PCA 相互独立，但却共用同一个计数器，所以同时使用 PCA0 和 PCA1 时需知对 PCA 计数器配置是唯一的。

下面针对 STC15W4K32S4 系列单片机 PCA 外设的 4 种工作模式进行内部结构分析。

■ PCA 工作于上升/下降沿捕获模式：

STC15W4K32S4 系列单片机 PCA 外设可配置为捕获模式，实现对外部信号的跳变采样。原理是当 PCAn 口采样到有效跳变时，PCA 控制器立即将 PCA 计数器 CH 和 CL 中的计数值装载到 PCAn 的比较/捕获寄存器 CCAPnH 和 CCAPnL 中，同时将 CCON 寄存器中的 CCFn 置 1。若 CCAPMn 中的 ECCFn 位被设置为 1，则产生中断，进 PCA 中断服务函数。PCA 捕获模式的内部结构框图如下图所示。

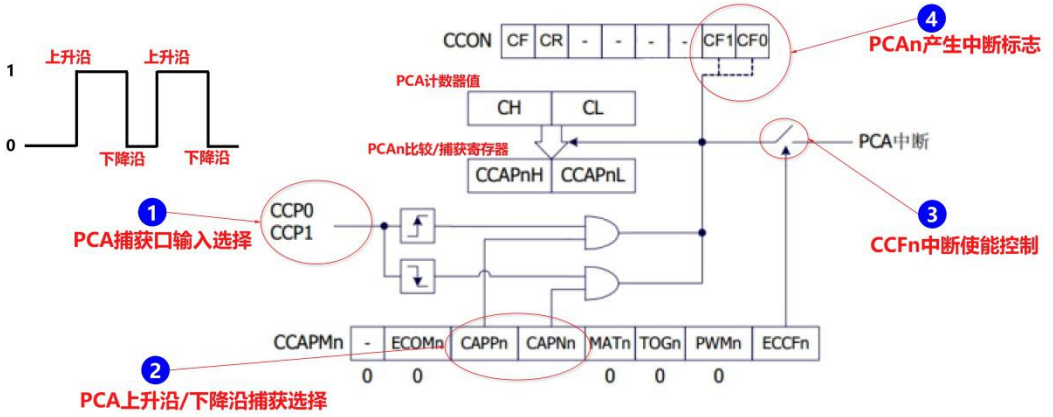


图 3：PCA 捕获模式内部结构示意图

1） 爱思考的初学者肯定会问这里 CCAPnH 和 CCAPnL 为什么叫比较/捕获寄存器，难道比较、捕获还可以共用？首先说一下，PCA 有捕获也有比较功能，肯定可以多定义几个寄存器，比如捕获寄存器和比较寄存器。其次，大家要理解下捕获和比较的区别，这里只说一点，就是捕获是针对 PCA 对外部输入信号处理使用的，而比较则是 PCA 用于输出信号给外部使用。那么 PCA 作为一个外设，其功能不可能既作

为捕获使用，又作为比较使用，那这样就可以节省资源将捕获寄存器和比较寄存器合为比较/捕获寄存器。

2） 大家一定要理解一点，就是 PCA 计数器一旦使能就一刻不停的在工作了，其 CH 和 CL 寄存器里面的值就会按照设置的时钟源在不断变化，这与 PCA 捕获输入引脚有没有产生有效跳变没有任何关系。

3） PCA 捕获输入引脚产生 2 次有效跳变，恰好是一个有规律的周期信号的一个周期，这也是捕获检测频率信号的必要条件。

■ PCA 工作于软件定时器模式：

STC15W4K32S4 系列单片机 PCA 外设可配置为软件定时器模式，实现定时功能。原理是装载了 PCAn 的比较/捕获寄存器 CCAPnH 和 CCAPnL 值后， PCA 计数器 CH 和 CL 中的计数值会和定值 CCAPnH 和 CCAPnL 不断比较，两者相等时，将 CCON 寄存器中的 CCFn 置 1。若 CCAPMn 中的 ECCFn 位被设置为 1，则产生中断，进 PCA 中断服务函数。PCA 软件定时器模式的内部结构框图如下图所示。

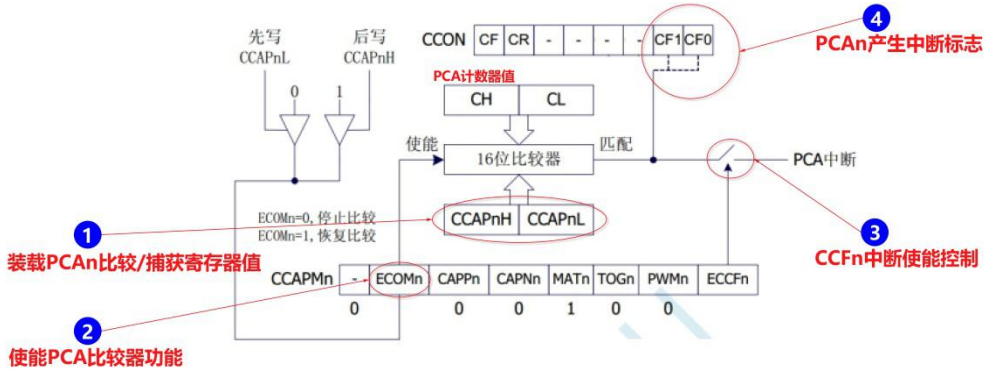


图 4：PCA 软件定时器模式内部结构示意图

1） 单片机 PCA 软件定时器模式作用和普通的定时器一样，STC15W4K32S4 系列单片机有 PCA0 和 PCA1 外设，也就是说可以再扩展 2 个定时器用。

2） 单片机 PCA 计数器 CH 和 CL 值会不断递增，这个递增的时间间隔是由 PCA 计数器选定的时钟源决定。而定时时间长短不仅取决于这个时间间隔，还和装载到 PCAn 的比较/捕获寄存器 CCAPnH 和 CCAPnL 值有关，但如果想持续不断的实现定时功能，则还需要每次进 PCA 中断时对比较/捕获寄存器 CCAPnH 和 CCAPnL 再赋值，再赋的值需在原有值的基础上新增加一个相同数值的值。

■ PCA 工作于高速脉冲输出模式：

STC15W4K32S4 系列单片机 PCA 外设可配置为高速脉冲输出模式，实现高速脉冲输出功能。原理是装载了 PCAn 的比较/捕获寄存器 CCAPnH 和 CCAPnL 值后， PCA 计数器 CH 和 CL 中的计数值会和定值 CCAPnH 和 CCAPnL 不断比较，两者相等时，PCAn 口输出将发生翻转，同时将 CCON 寄存器中的 CCFn 置 1。若 CCAPMn 中的 ECCFn 位被设置为 1，则产生中断，进 PCA 中断服务函数。PCA 高速脉冲输出模式的内部结构框图如下图所示。

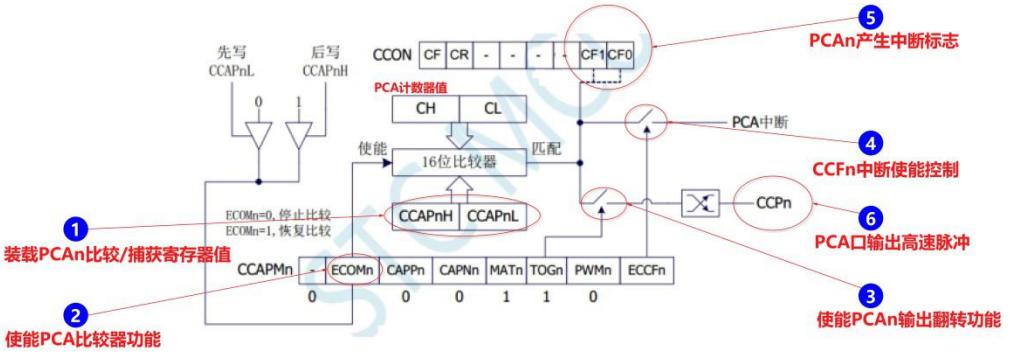


图 5：PCA 高速脉冲输出模式内部结构示意图

1） 单片机 PCA 高速脉冲输出模式下，PCAn 的 CCAPMn 寄存器的 TOGn、MATn 和 ECOMn 位必须都置 1。

2） 单片机 PCA 计数器 CH 和 CL 值会不断递增，这个递增的时间间隔是由 PCA 计数器选定的时钟源决定。而 PCAn 口翻转输出的时间长短不仅取决于这个时间间隔，还和装载到 PCAn 的比较/捕获寄存器 CCAPnH 和 CCAPnL 值有关，但如果想持续不断的实现翻转功能，则还需要每次进 PCA 中断时对比较/捕获寄存器 CCAPnH 和 CCAPnL 再赋值，再赋的值需在原有值的基础上新增加一个值，这个值确定了下一次翻转的时间。

■ PCA 工作于 PWM 模式：

STC15W4K32S4 系列单片机 PCA 外设可配置为 PWM 模式，实现脉宽调制功能。通过对 PCAn 的 PCA_PWMn 寄存器配置来选择单片机 PCA 外设具体工作于 8 位 PWM、7 位 PWM 还是 6 位 PWM。下面以 8 位 PWM 来介绍原理。

PCA_PWMn 寄存器的 ESBn[1:0]设置为 00 时，PCAn 工作于 8 位 PWM 模式，此时将{0,CL[7:0]}与 PCAn 的比较/捕获寄存器{EPCnL,CCAPnL[7:0]}进行比较。当{0,CL[7:0]}的值小于{EPCnL,CCAPnL[7:0]}时，PCAn 口输出低电平；当{0,CL[7:0]}的值等于或大于{EPCnL,CCAPnL[7:0]}时，PCAn 口输出高电平。当 PCA 计数器 CL[7:0] 的值由 0xFF 变为 0x00 溢出时，{EPCnH,CCAPnH[7:0]}的内容重新装载到{EPCnL,CCAPnL[7:0]}中，实现无干扰地更新 PWM。PCA 的 8 位 PWM 模式的内部结构框图如下图所示。

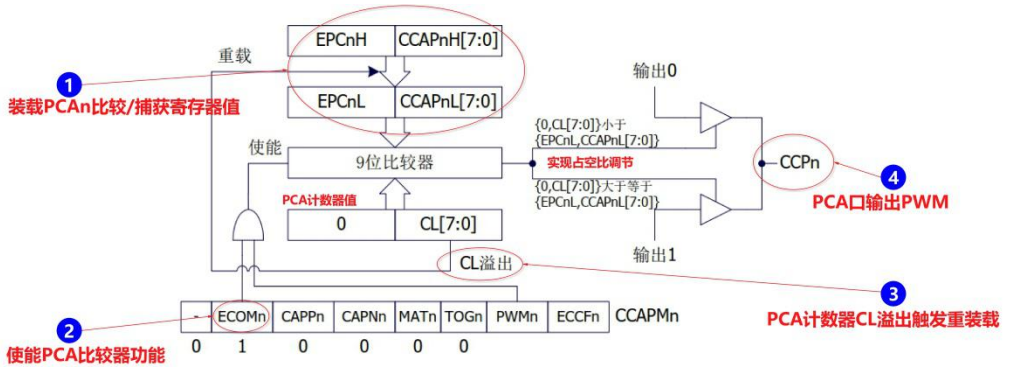


图 6：PCA8 位 PWM 模式内部结构示意图

1） 单片机 PCA 的 PWM 模式可设置为 6 位、7 位和 8 位 PWM 输出，STC15W4K32S4 系列单片机有 PCA0 和 PCA1 外设，也就是说可以再扩展 2 个 PWM 用。

2） 单片机 PCA 计数器 CL 值会不断递增，这个递增的时间间隔是由 PCA 计数器选定的时钟源决定，一旦 CL 溢出，{EPCnH,CCAPnH[7:0]}的内容重新装载到{EPCnL,CCAPnL[7:0]}中，这很关键，这样输出信号的频率和占空比配置非常灵活。

4. 软件设计

4.1. PCA 寄存器汇集

STC15W4K32S4 系列单片机操作 PCA 时会用到 12 个寄存器，如下表所示：

表 3：STC15W4K32S4 系列 PCA 使用寄存器汇总

序号	寄存器名	读/写	功能描述
1	CMOD	读/写	PCA 工作模式寄存器。
2	CCON	读/写	PCA 控制寄存器。
3	CH	读/写	PCA 计数器高 8 位寄存器。
4	CL	读/写	PCA 计数器低 8 位寄存器。
5	CCAPM0	读/写	PCA0 比较/捕获寄存器。
6	CCAPM1	读/写	PCA1 比较/捕获寄存器。
7	CCAP0H	读/写	PCA0 比较/捕获高 8 位寄存器。
8	CCAP0L	读/写	PCA0 比较/捕获低 8 位寄存器。
9	CCAP1H	读/写	PCA1 比较/捕获高 8 位寄存器。
10	CCAP1L	读/写	PCA1 比较/捕获低 8 位寄存器。
11	PCA_PWM0	读/写	PCA0 的 PWM 寄存器。
12	PCA_PWM1	读/写	PCA1 的 PWM 寄存器。

✧ 注:上述寄存器的2路相互独立的PCA都有比较/捕获寄存器和PWM寄存器,只需把PCA0对应的这一组寄存器搞清楚，那么 PCA1 对应的原理完全一致。

4.2. 寄存器解析

4.2.1. PCA 工作模式存器 CMOD

PCA 工作模式寄存器 CMOD 的 CIDL 位为 PCA 计数在空闲模式下的控制位，该位决定单片机在空闲模式下是否停止 PCA 计数功能。关于 CPS0~CPS2 位的操作： PCA 计数脉冲

源选择控制位，根据应用需要进行选择配置。ECF 位为 PCA 计数溢出中断使能位。

CMOD：PCA工作模式寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
CMOD	D9H	name	CIDL	-	-	-	CPS2	CPS1	CPS0	ECF

1 寄存器名

2 PCA计数在空闲模式下的控制位

CIDL：空闲模式下是否停止PCA计数的控制位。
当CIDL=0时，空闲模式下PCA计数器继续工作；
当CIDL=1时，空闲模式下PCA计数器停止工作。

3 PCA计数脉冲源选择控制位

CPS2、CPS1、CPS0：PCA计数脉冲源选择控制位。PCA计数脉冲选择如下表所示。

CPS2	CPS1	CPS0	选择CCP/PCA/PWM时钟源输入
0	0	0	0，系统时钟，SYSclk/12
0	0	1	1，系统时钟，SYSclk/2
0	1	0	2，定时器0的溢出脉冲。由于定时器0可以工作在1T模式，所以可以达到计一个时钟就溢出，从而达到最高频率CPU工作时钟SYSclk。通过改变定时器0的溢出率，可以实现可调频率的PWM输出
0	1	1	3，ECI/P1.2(或P3.4或P2.4)脚输入的外部时钟(最大速率=SYSclk/2)
1	0	0	4，系统时钟，SYSclk
1	0	1	5，系统时钟/4，SYSclk/4
1	1	0	6，系统时钟/6，SYSclk/6
1	1	1	7，系统时钟/8，SYSclk/8

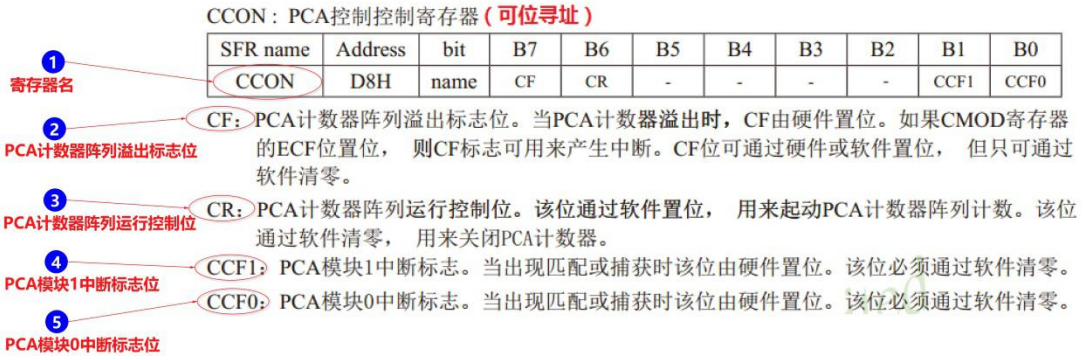
4 PCA计数溢出中断使能位

ECF：PCA计数溢出中断使能位。
当ECF = 0时，禁止寄存器CCON中CF位的中断；
当ECF = 1时，允许寄存器CCON中CF位的中断。

图 7：PCA 工作模式寄存器

4.2.2. PCA 控制寄存器 CCON

PCA 控制寄存器 CCON 的 CF 位和 CR 位操作：分别是 PCA 计数器阵列溢出标志位和运行控制位，对 CF 位和 CR 位的清零操作必须软件实现。



CCON：PCA控制控制寄存器（可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
CCON	D8H	name	CF	CR	-	-	-	-	CCF1	CCF0

1 寄存器名

2 PCA计数器阵列溢出标志位

CF：PCA计数器阵列溢出标志位。当PCA计数器溢出时，CF由硬件置位。如果CMOD寄存器的ECF位置位， 则CF标志可用来产生中断。CF位可通过硬件或软件置位， 但只可通过软件清零。

3 PCA计数器阵列运行控制位

CR：PCA计数器阵列运行控制位。该位通过软件置位， 用来起动PCA计数器阵列计数。该位通过软件清零， 用来关闭PCA计数器。

4 PCA模块1中断标志位

CCF1：PCA模块1中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

5 PCA模块0中断标志位

CCF0：PCA模块0中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

图 8：PCA 控制寄存器

4.2.3. PCA0 比较/捕获寄存器 CCAPM0

PCA0 比较/捕获寄存器 CCAPM0 的 ECOM0 位的操作：用于比较器功能(软件定时器)。CCAPM0 寄存器的 CAPP0、CAPN0 位的操作：用于捕获。CCAPM0 寄存器的 MAT0、ECCF0 位的操作： 用于捕获/比较。CCAPM0 寄存器的 TOG0 位的操作： 用于高速脉冲输出模式。CCAPM0 寄存器的 PWM0 位的操作： 用于脉宽调制模式。

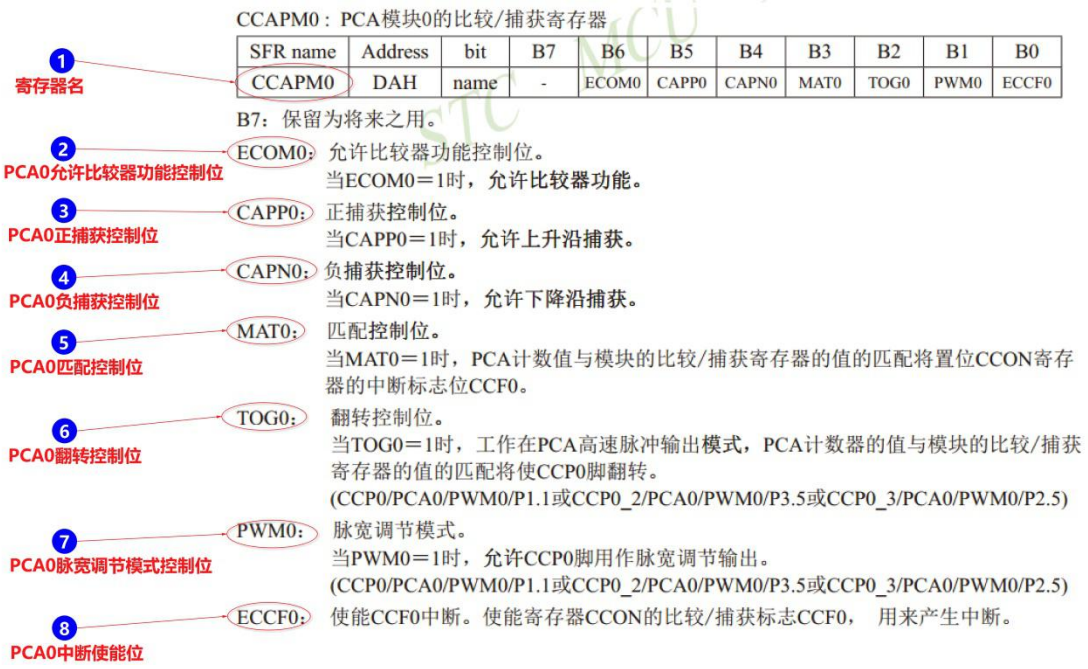


图 9：PCA0 比较/捕获寄存器

4.2.4. PCA0 的 PWM 寄存器 PCA_PWM0

PCA0 的 PWM 寄存器 PCA_PWM0 主要是配置 PCA0 工作于 PWM 模式时的功能选择，又因为 PCA 工作于 PWM 模式时的比较器是 9 位比较器，所以在 PCA_PWM0 寄存器中设置了 EPC0H 位和 EPC0L 位，与 CCAP0H 和 CCAP0L 组成 9 位数以完成比较。（可参考 PCA 工作于 PWM 的内部结构图去理解）

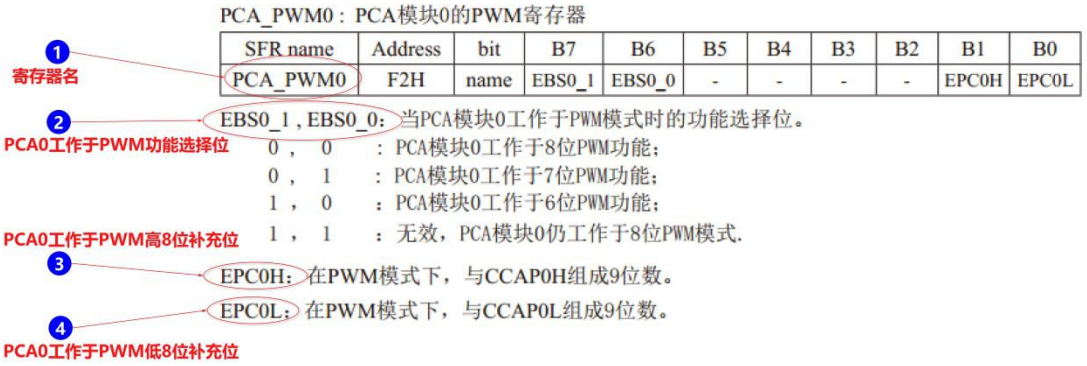


图 10：PCA0 的 PWM 寄存器

✧ 注：只有先使能了 STC15W4K32S4 系列单片机 PCA 模块 0 的 PWM 功能，即模块寄存器 CCAPM0 的 PWM0 和 ECOM0 位都置 1 了，对 PCA_PWM0 寄存器的操作才有意义。

4.3. PCA 捕获实验

✧ 注：本节的实验源码是在“实验 2-9-1：PWM2 和 PWM3 呼吸灯实验”的基础上修改。本节对应的实验源码是：“实验 2-10-1：PCA 捕获实验”。

4.3.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 4：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	uart	.c	外部 UART 有关的用户自定义函数。
1	pca	.c	外部 PCA 捕获有关的用户自定义函数。
2	delay	.c	包含用户自定义延时函数。

4.3.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "uart.h"
3. #include "pca.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 5：头文件包含路径

序号	路径	描述
1	..\ Source	Uart.h、pca.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

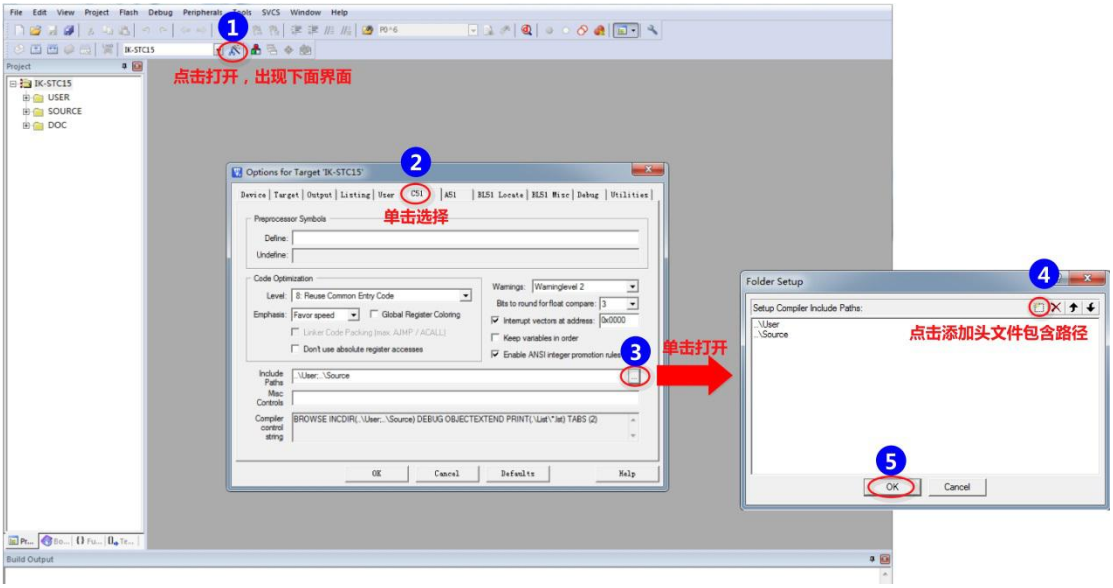


图 11：添加头文件包含路径

4.3.3. 编写代码

首先，在 pca.c 文件中编写 PCA0 的初始化函数 PCAInit，代码如下。

程序清单：PCA0 初始化函数

```
1.  /*************************************************************************
2.  功能描述：PCA 初始化
3.  入口参数：无
4.  返回值：无
5.  *************************************************************************/
6.  void PCAInit(void)
7.  {
8.      CCON = 0x00;                    //CF、CR、CCF1、CCF0 位均清零
9.      CMOD &= 0x7F;                   //CIDL 位置 0，空闲模式下 PCA 计数器仍然工作
10.     CMOD &= 0xF9;                 //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
11.     CMOD |= 0x08;                 //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
12.     CMOD |= 0x01;               //ECF 位置 1,允许寄存器 CCON 中 CF 位中断(使能 PCA 计时中断)
13.     CL = 0x00;                      //PCA 计数器赋初值
14.     CH = 0x00;                      //PCA 计数器赋初值
15.     CCAPM0 &= 0xBF;                 //ECOM0 位置 0，禁止比较器功能
16.     CCAPM0 &= 0xDF;                 //CAPP0 位置 0，禁止上升沿捕获
17.     CCAPM0 |= 0x10;                 //CAPN0 位置 1，允许下降沿捕获
18.     CCAPM0 &= 0xF7;                 //MAT0 位置 0，禁止匹配控制位
19.     CCAPM0 &= 0xFB;                 //TOG0 位置 0，禁止翻转控制位
20.     CCAPM0 &= 0xFD;                 //PWM0 位置 0，禁止 PWM 模式
21.     CCAPM0 |= 0x01;                 //ECCF0 位置 1，允许 CCF0 中断
22.     CCAP0L = 0x00;                  //PCA 捕获值寄存器赋初值
23.     CCAP0H = 0x00;                  //PCA 捕获值寄存器赋初值
24.     CR = 1;                         //启动 PCA 计数器阵列计数
25. }
```

■ 需要特别注意在 pca.c 文件中定义的几个全局变量：

```
1.  volatile uint8 cnt = 0;                         //存储 PCA 计时溢出次数
2.  volatile uint32 count0 = 0;                     //记录上一次的捕获值
3.  volatile uint32 count1 = 0;                     //记录本次的捕获值
4.  volatile uint32 length = 0;                     //存储信号的时间长度
```

然后，编写 PCA 中断服务函数，一旦进入中断除了会软件清除相应中断标志位，还会通过算法计算出脉冲宽度量化值，代码如下。

程序清单：中断服务函数

```
1.  /*************************************************************************
2.   * 描  述 : PCA 中断服务函数
```

```
3.   * 入  参 : 无
4.   * 返回值 : 无
5.   ******************************************************************************/
6.  void PCA_int (void) interrupt PCA_VECTOR using 1
7.  {
8.
9.      if(CF)                                  //PCA 计数器溢出
10.     {
11.         CF = 0;                             //将 PCA 计数器阵列溢出标志位软件清零
12.         cnt++;                              //PCA 计时溢出次数累加
13.     }
14.     if(CCF0)                                //PCA 计数器溢出
15.     {
16.         CCF0 = 0;                           //将 PCA 计数器阵列溢出标志位软件清零
17.         count0 = count1;                    //备份上一次的捕获值
18.         ((uint8 *)&count1)[3] = CCAP0L;
19.         ((uint8 *)&count1)[2] = CCAP0H;
20.         ((uint8 *)&count1)[1] = cnt;
21.         ((uint8 *)&count1)[0] = 0;
22.         length = count1 - count0;          //length 保存的即为捕获的脉冲宽度量化值
23.     }
24. }
```

最后，在主函数中对用到的 PCA 口（P1.1）进行模式配置，调用 PCA 初始化函数，开启总中断，在主循环中获取脉冲宽度量化值 length 并通过串口显示出来。

代码清单：主函数

```
1.  int main()
2.  {
3.      uint32  Temp;
4.      static uint8   strPhoto[9];
5.
6.  ////////////////////////////////////////////////
7.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
8.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
9.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
10. //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
11. ////////////////////////////////////////////////
12.     P1M1 &= 0xFD;   P1M0 &= 0xFD;                   //设置 P1.1 为准双向口
13.     P3M1 &= 0xFC;   P3M0 &= 0xFC;                   //设置 P3.0~P3.1 为准双向口
14.
15.     Uart1_Init();                                   //串口 1 初始化
16.     PCAInit();                                      //PCA 模块 0 初始化
```

```
    EA = 1;                                          //使能总中断
    delay_ms(10);                                    //初始化后延时

  while (1)
    {
        memset(strPhoto, 0, 9);             //将 strPhoto 数组初始化（清零）
        if(length<0xFFFFFF)       //判断采集的脉冲宽度原始值在有效范围内
        {
            Temp=length;
        }
        //将采集的脉冲宽度原始值以十进制形式将各位数值存放于数组 strPhoto 中
        strPhoto[0] = Temp/100000000+48;
        strPhoto[1] = (Temp%100000000)/10000000+48;
        strPhoto[2] = ((Temp%100000000)%10000000)/1000000+48;
        strPhoto[3] = (((Temp%100000000)%10000000)%1000000)/100000+48;
        strPhoto[4] = ((((Temp%100000000)%10000000)%1000000)%100000)/10000+48;

        strPhoto[5] = (((((Temp%100000000)%10000000)%1000000)%100000)%10000)/100
0+48;
        strPhoto[6] = ((((((Temp%100000000)%10000000)%1000000)%100000)%10000)%10
00)/100+48;
        strPhoto[7] = (((((((Temp%100000000)%10000000)%1000000)%100000)%10000)%1
000)%100)/10+48;
        strPhoto[8] = (((((((Temp%100000000)%10000000)%1000000)%100000)%10000)%1
000)%100)%10+48;

        //在串口调试助手上显示脉冲宽度采集原始值
        SendStringByUart1(table1);          //发送固定字符串
        SendStringByUart1(strPhoto);        //发送捕获的脉冲宽度采集原始值
        SendDataByUart1(0x0D);              //发送换行符
        SendDataByUart1(0x0A);              //发送换行符

        delay_ms(1000);                     //1000ms 检测一次，此时间可调，但不宜过小
    }
}
```

4.3.4. 硬件连接

本实验需要使用函数发生器或其他可以提供脉冲信号的信号源给开发板 P1.1 口提供幅值 3.3V~5V 的脉冲信号，实际衡量脉冲宽度的信息量会串口发给 PC 在串口调试助手上显示。

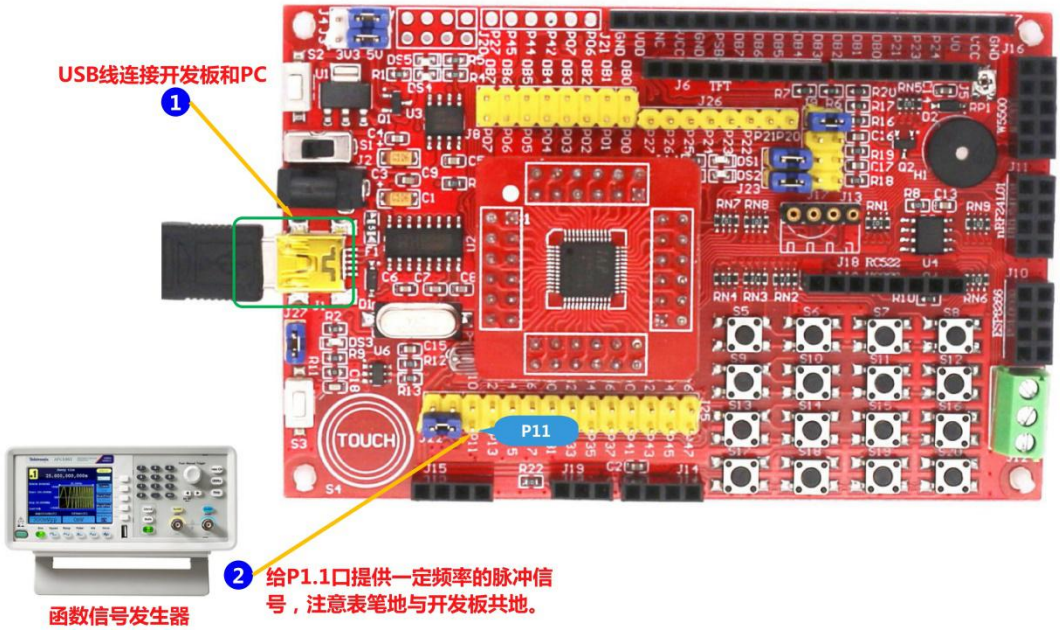


图 12：开发板连接图

4.3.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-10-1：PCA 捕获实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\PCA0_capture\projec”目录下的工程“PCA0_capture.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“PCA0_capture.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，给开发板 P1.1 口提供一定频率的脉冲信号，可在串口调试助手显示捕获的脉冲宽度量化信息，可改变脉冲信号宽度观察串口调试助手显示值情况。

4.4. PCA 定时器实验

✧ 注：本节的实验源码是在“实验 2-10-1：PCA 捕获实验”的基础上修改。本节对应的实验源码是：“实验 2-10-2：PCA 定时器实验”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-10-1：PCA 捕获实验”部分。

4.4.2. 编写代码

首先，在 pca.c 文件中编写 PCA0 的初始化函数 PCAInit，代码如下。

程序清单：PCA0 初始化函数

```
1.  /************************************************************************
2.  功能描述：PCA 初始化
3.  入口参数：无
4.  返回值：无
5.  ************************************************************************/
6.  void PCAInit(void)
7.  {
8.      CCON = 0x00;                    //CF、CR、CCF1、CCF0 位均清零
9.      CMOD &= 0x7F;                   //CIDL 位置 0，空闲模式下 PCA 计数器仍然工作
10.     CMOD &= 0xF1;                 //CP2、CP1、CP0 设置为 110，PCA 时钟源选择为系统时钟/6
11.     CMOD |= 0x0C;                 //CP2、CP1、CP0 设置为 110，PCA 时钟源选择为系统时钟/6
12.     CMOD |= 0x01;               //ECF 位置 1，允许寄存器 CCON 中 CF 位中断（使能 PCA 计时中断）
13.     CL = 0x00;                      //PCA 计数器赋初值
14.     CH = 0x00;                      //PCA 计数器赋初值
15.     CCAPM0 |= 0x40;                 //ECOM0 位置 1，允许比较器功能
16.     CCAPM0 &= 0xDF;                 //CAPP0 位置 0，禁止上升沿捕获
17.     CCAPM0 &= 0xEF;                 //CAPN0 位置 0，禁止下降沿捕获
18.     CCAPM0 |= 0x08;                 //MAT0 位置 1，开启匹配控制位
19.     CCAPM0 &= 0xFB;                 //TOG0 位置 0，禁止翻转控制位
20.     CCAPM0 &= 0xFD;                 //PWM0 位置 0，禁止 PWM 模式
21.     CCAPM0 |= 0x01;                 //ECCF0 位置 1，允许 CCF0 中断
22.     CCAP0L = 0x00;                  //PCA 比较值寄存器赋初值
23.     CCAP0H = 0x24;                  //PCA 比较值寄存器赋初值
24.     CR = 1;                         //启动 PCA 计数器阵列计数
25. }
```

然后，编写 PCA 中断服务函数，一旦进入中断不仅软件清除相应中断标志位，还需不断更新 CCAP0H 和 CCAP0L 值，实现指示灯 1s 闪烁一次，代码如下。

程序清单：中断服务函数

```
1.  /************************************************************************
2.  功能描述：PCA 中断服务程序(每 5ms 进入一次 PCA 中断)
3.  入口参数：无
4.  返回值：无
5.  ************************************************************************/
6.  void PCA_int (void) interrupt PCA_VECTOR using 1
7.  {
8.      uint16   temp;
9.      if(CF)                              //PCA 计数器溢出
10.     {
```

```
11.       CF = 0;                                   //将 PCA 计数器阵列溢出标志位软件清零
12.     }
13.     if(CCF0)                                    //PCA 计数器溢出
14.     {
15.         CCF0 = 0;                                 //将 PCA 计数器阵列溢出标志位软件清零
16.         temp=(CCAP0H<<8)+CCAP0L+0x2400;           //运算符“+”的优先级大于“<<”
17.         CCAP0L=temp;                              //取计算结果的低 8 位
18.         CCAP0H=temp>>8;                           //取计算结果的高 8 位
19.         cnt++;                                    //每 5ms 执行一次 cnt 加一的操作
20.     }
21.     if(cnt == 200)                              //cnt 加到 200 后，正好用到 1000ms
22.     {
23.         led_toggle(LED_1);                        //翻转蓝色指示灯 DS1
24.         cnt = 0;
25.     }
26. }
```

最后，在主函数中调用 PCA0 初始化函数，开启总中断，在主循环中无任务，说明程序执行的是中断服务函数的语句。

代码清单：主函数

```
1.  int main()
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口
10.
11.     PCAInit();          //PCA 模块 0 初始化
12.     EA = 1;             //使能总中断
13.   while (1)
14.     {
15.         ;                 //无任务，说明 LED 亮灭来自于中断
16.     }
17. }
```

4.4.3. 硬件连接

本实验需要用到用户蓝色指示灯 DS1 作为定时器定时指示用，因此需要按下图所示连接短路帽。

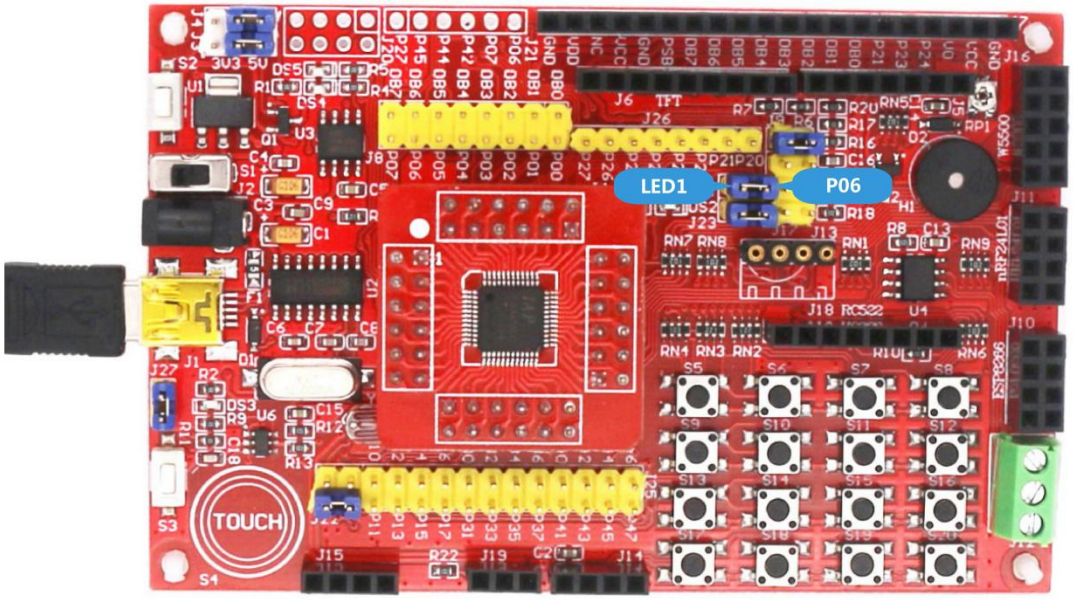


图 13：开发板连接图

4.4.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-10-2：PCA 定时器实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\PCA0_timer \projec”目录下的工程“PCA0_timer.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“PCA0_timer.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色指示灯 DS1 约 1s 闪烁一次。

4.5. PCA 高速脉冲输出实验

✧ 注：本节的实验源码是在“实验 2-10-1：PCA 捕获实验”的基础上修改。本节对应的实验源码是：“实验 2-10-3：PCA 高速脉冲输出实验”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-10-1：PCA 捕获实验”部分。

4.5.2. 编写代码

首先，在 pca.c 文件中编写 PCA0 和 PCA1 的初始化函数 PCAInit，代码如下。

程序清单：PCA0 和 PCA1 初始化函数

```
1.  /*************************************************************************
2.  功能描述：PCA 初始化
3.  入口参数：无
4.  返回值：无
5.  *************************************************************************/
6.  void PCAInit(void)
7.  {
8.      AUXR1 &= 0xEF;              //选择 PCA 模块 0 为引脚 P2.5,选择 PCA 模块 1 为引脚 P2.6
9.      AUXR1 |= 0x20;              //选择 PCA 模块 0 为引脚 P2.5,选择 PCA 模块 1 为引脚 P2.6
10.
11.     CCON = 0x00;                    //CF、CR、CCF1、CCF0 位均清零
12.     CMOD &= 0x7F;                   //CIDL 位置 0，空闲模式下 PCA 计数器仍然工作
13.     CMOD &= 0xF1;               //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
14.     CMOD |= 0x08;               //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
15.     CMOD &= 0xFE;           //ECF 位置 0,禁止寄存器 CCON 中 CF 位中断（禁止 PCA 计时中断）
16.     CL = 0x00;                      //PCA 计数器赋初值
17.     CH = 0x00;                      //PCA 计数器赋初值
18.     //PCA 模块 0 初始化部分
19.     CCAPM0 |= 0x40;                 //ECOM0 位置 1，允许比较器功能
20.     CCAPM0 &= 0xDF;                 //CAPP0 位置 0，禁止上升沿捕获
21.     CCAPM0 &= 0xEF;                 //CAPN0 位置 0，禁止下降沿捕获
22.     CCAPM0 |= 0x08;                 //MAT0 位置 1，开启匹配控制位
23.     CCAPM0 |= 0x04;                 //TOG0 位置 1，开启翻转控制位
24.     CCAPM0 &= 0xFD;                 //PWM0 位置 0，禁止 PWM 模式
25.     CCAPM0 |= 0x01;                 //ECCF0 位置 1，开启 CCF0 中断
26.     value1 =T38K4HZ;                 //value1 赋值
27.     CCAP0L = value1;                  //PCA 比较值寄存器赋初值
28.     CCAP0H = value1>>8;               //PCA 比较值寄存器赋初值
29.     //PCA 模块 1 初始化部分
30.     CCAPM1 |= 0x40;                 //ECOM1 位置 1，允许比较器功能
31.     CCAPM1 &= 0xDF;                 //CAPP1 位置 0，禁止上升沿捕获
32.     CCAPM1 &= 0xEF;                 //CAPN1 位置 0，禁止下降沿捕获
33.     CCAPM1 |= 0x08;                 //MAT1 位置 1，开启匹配控制位
34.     CCAPM1 |= 0x04;                 //TOG1 位置 1，开启翻转控制位
35.     CCAPM1 &= 0xFD;                 //PWM1 位置 0，禁止 PWM 模式
36.     CCAPM1 |= 0x01;                 //ECCF1 位置 1，开启 CCF1 中断
37.     value2 =T76K8HZ;                 //value2 赋值
38.     CCAP1L = value2;                  //PCA 比较值寄存器赋初值
39.     CCAP1H = value2>>8;               //PCA 比较值寄存器赋初值
```

```
40.
41.     value1 +=T38K4HZ;                  //value1 再赋值
42.     value2 +=T76K8HZ;                  //value2 再赋值
43.     CR = 1;                            //启动 PCA 计数器阵列计数
44. }
```

然后，编写 PCA 中断服务函数，一旦进入中断则软件清除相应中断标志位，并对 PCA0 和 PCA1 所用的 CCAPnH 和 CCAPnL 寄存器再赋值，以进行下一次匹配比较，代码如下。

程序清单：中断服务函数

```
1.  /*************************************************************************
2.  功能描述：PCA 中断服务程序
3.  入口参数：无
4.  返回值：无
5.   *************************************************************************/
6.  void PCA_int (void) interrupt PCA_VECTOR using 1
7.  {
8.      if(CF)                                  //PCA 计数器溢出
9.      {
10.         CF = 0;                             //将 PCA 计数器阵列溢出标志位软件清零
11.     }
12.     if(CCF0)                                //PCA 计数器溢出
13.     {
14.         CCF0 = 0;                           //将 PCA 计数器阵列溢出标志位软件清零
15.         CCAP0L=value1;                      //取计算结果的低 8 位
16.         CCAP0H=value1>>8;                   //取计算结果的高 8 位
17.         value1 +=T38K4HZ;                   //value1 再赋值
18.     }
19.     if(CCF1)                                //PCA 计数器溢出
20.     {
21.         CCF1 = 0;                           //将 PCA 计数器阵列溢出标志位软件清零
22.         CCAP1L=value2;                      //取计算结果的低 8 位
23.         CCAP1H=value2>>8;                   //取计算结果的高 8 位
24.         value2 +=T76K8HZ;                   //value2 再赋值
25.     }
26. }
```

最后，在主函数中调用 PCA0 和 PCA1 初始化函数，开启总中断，在主循环中无任务，说明程序执行的是中断服务函数的语句。

代码清单：主函数

```
1.  int main()
2.  {
```

```
//////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////
    P2M1 &= 0x9F;   P2M0 &= 0x9F;                    //设置 P2.5~P2.6 为准双向口

    PCAInit();          //PCA 模块初始化
    EA = 1;             //使能总中断
  while (1)
    {
        ;                 //无任务
    }
}
```

4.5.3. 硬件连接

本实验需要使用示波器或逻辑分析仪测量 P2.6 和 P2.5 信号波形，所以开发板 J22 端子的跳线帽需去除（因为使用了 P2.5 口），如下图所示连接。

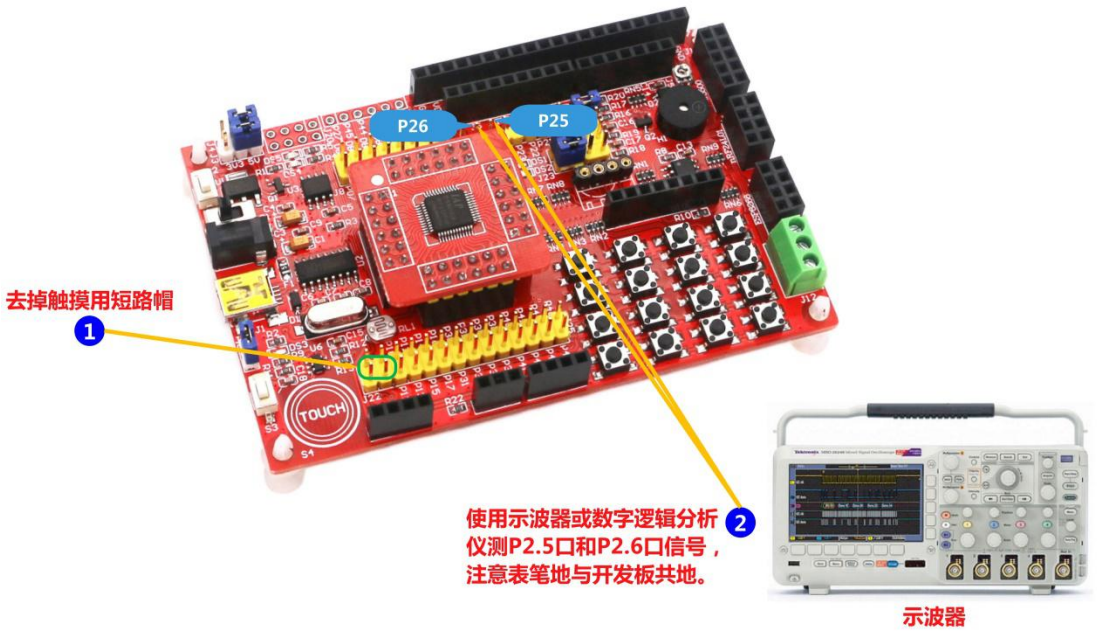


图 14：开发板连接图

4.5.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-10-3：PCA 高速脉冲输出实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。

3. 在 Keil C51 中执行“Project→Open Project”打开“…\PCA_pulse \projec”目录下的工程“PCA_pulse.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“PCA_pulse.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，使用示波器可以观察到 PCA0 选择的 P2.5 输出脉冲信号的频率约是 38.4KHZ，使用示波器可以观察到PCA1 选择的P2.6 输出脉冲信号的频率约是 76.8KHZ。

4.6. PCA 输出 PWM 实验

✧ 注：本节的实验源码是在“实验 2-10-1：PCA 捕获实验”的基础上修改。本节对应的实验源码是：“实验 2-10-4：PCA 输出 PWM 实验”。

4.6.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-10-1：PCA 捕获实验”部分。

4.6.2. 编写代码

首先，在 pca.c 文件中编写 PCA0 和 PCA1 的初始化函数 PCAInit，代码如下。

程序清单：PCA0 和 PCA1 初始化函数

```
1.  /*************************************************************************
2.  功能描述：PCA 初始化
3.  入口参数：无
4.  返回值：无
5.  *************************************************************************/
6.  void PCAInit(void)
7.  {
8.      AUXR1 &= 0xEF;              //选择 PCA 模块 0 为引脚 P2.5,选择 PCA 模块 1 为引脚 P2.6
9.      AUXR1 |= 0x20;              //选择 PCA 模块 0 为引脚 P2.5,选择 PCA 模块 1 为引脚 P2.6
10.
11.     CCON = 0x00;                        //CF、CR、CCF1、CCF0 位均清零
12.     CMOD &= 0x7F;                       //CIDL 位置 0，空闲模式下 PCA 计数器仍然工作
13.     CMOD &= 0xF1;               //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
14.     CMOD |= 0x08;               //CP2、CP1、CP0 设置为 100，PCA 时钟源选择为系统时钟
15.     CMOD &= 0xFE;       //ECF 位置 0，禁止寄存器 CCON 中 CF 位中断（禁止 PCA 计时中断）
16.     CL = 0x00;                          //PCA 计数器赋初值
17.     CH = 0x00;                          //PCA 计数器赋初值
18.     //PCA 模块 0 初始化部分
19.     CCAPM0 |= 0x40;                     //ECOM0 位置 1，允许比较器功能
```

```
    CCAPM0 &= 0xDF;                 //CAPP0 位置 0，禁止上升沿捕获
    CCAPM0 &= 0xEF;                 //CAPN0 位置 0，禁止下降沿捕获
    CCAPM0 &= 0xF7;                 //MAT0 位置 0，禁止匹配控制位
    CCAPM0 &= 0xFB;                 //TOG0 位置 0，禁止翻转控制位
    CCAPM0 |= 0x02;                 //PWM0 位置 1，开启 PWM 模式
    CCAPM0 &= 0xFE;                 //ECCF0 位置 0，禁止 CCF0 中断
    PCA_PWM0 &= 0xBF;               //PCA 模块 0 工作于 6 位 PWM 功能
    PCA_PWM0 |= 0x80;               //PCA 模块 0 工作于 6 位 PWM 功能
    CCAP0L = 0x20;                  //PWM 占空比为 50%
    CCAP0H = 0x20;                  //PCA 比较值寄存器赋初值
    //PCA 模块 1 初始化部分
    CCAPM1 |= 0x40;                 //ECOM1 位置 1，允许比较器功能
    CCAPM1 &= 0xDF;                 //CAPP1 位置 0，禁止上升沿捕获
    CCAPM1 &= 0xEF;                 //CAPN1 位置 0，禁止下降沿捕获
    CCAPM1 &= 0xF7;                 //MAT1 位置 0，禁止匹配控制位
    CCAPM1 &= 0xFB;                 //TOG1 位置 0，禁止翻转控制位
    CCAPM1 |= 0x02;                 //PWM1 位置 1，开启 PWM 模式
    CCAPM1 &= 0xFE;                 //ECCF1 位置 0，禁止 CCF0 中断
    PCA_PWM1 &= 0x3F;               //PCA 模块 0 工作于 8 位 PWM 功能
    CCAP1L = 0x20;                  //PWM 占空比为 87.5%
    CCAP1H = 0x20;                  //PCA 比较值寄存器赋初值

    CR = 1;                         //启动 PCA 计数器阵列计数
}
```

然后，在主函数中调用 PCA0 和 PCA1 初始化函数，开启总中断，在主循环中无任务，说明程序执行的是中断服务函数的语句。

代码清单：主函数

```
int main()
{
//////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////

    PCAInit();          //PCA 模块初始化

  while (1)
    {
        ;                 //无任务
    }
```

```
16. }
```

4.6.3. 硬件连接

本实验需要使用示波器或逻辑分析仪测量 P2.6 和 P2.5 信号波形，所以开发板 J22 端子的跳线帽需去除（因为使用了 P2.5 口），如下图所示连接。

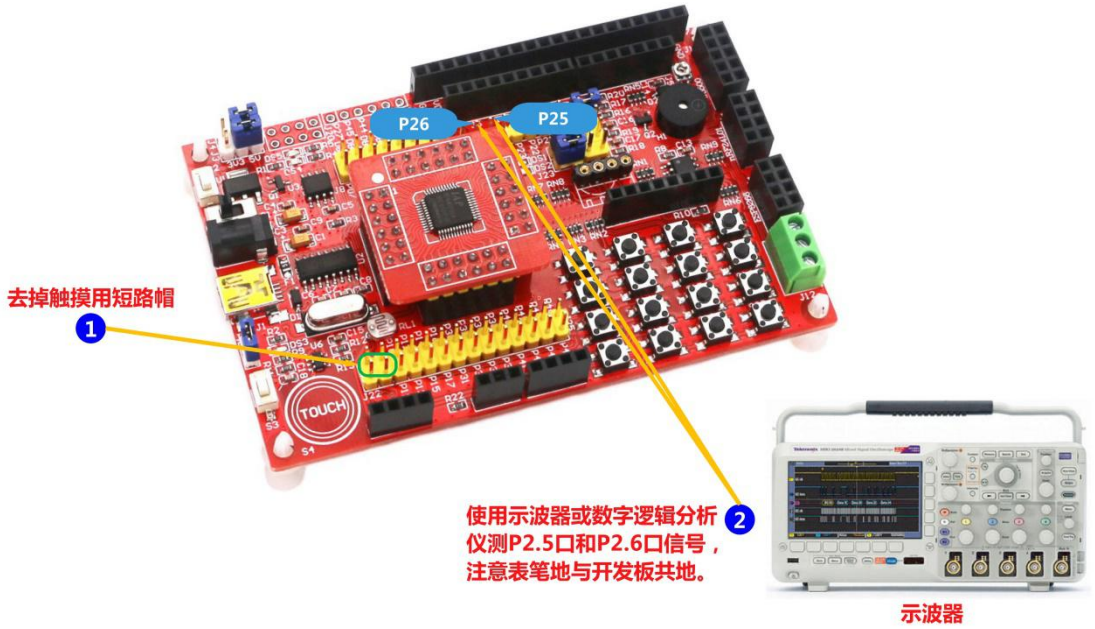


图 15：开发板连接图

4.6.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-10-4：PCA 输出 PWM 实验”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\PCA_pwm \projec”目录下的工程“PCA_pwm.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“PCA_pwm.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，使用示波器可以观察到 PCA0 选择的 P2.5 输出脉冲信号的占空比约是 50%，使用示波器可以观察到 PCA1 选择的 P2.6 输出脉冲信号的占空比约是 87.5%。