第4章

定时器/计数器

项目过程

通过定时器/计数器实现流水灯控制。

建议学时

4 学时。

知识要点

(1)定时器的结构。

(2)TMOD 和 TCON。

(3)定时器/计数器的工作方式。

(4)定时器/计数器的编程步骤。

技能掌握

定时器/计数器的设置以及采用查询方式使用定时器。

4.1 项目导引

前面的流水灯的时间控制通过空循环语句来实现,定时不是很精确。本章通过用定时器来控制流水灯任务,可以实现精确的时间控制,这就需要了解定时器的使用。定时器和计数器实质功能相同,本章利用 LED 灯二进制计数任务来掌握计数器的使用。

4.2 技术准备

4.2.1 定时器的结构

计数器就是对外部输入脉冲的计数。定时器也是对脉冲进行计数完成的,计数的是 MCS-51 内部产生的标准脉冲,通过计数脉冲个数实现定时。单片机定时器引脚位置如图 4-1 所示,单片机引脚的 P3.4 和 P3.5 的第二功能就是对应的定时器 T0 和 T1。MCS-51 系列 8031、8051 单片机有两个 16 位定时器/计数器(即 T0 和 T1);8032、8052 单片机有 3 个 16 位定时器/计数器(即 T0、T1 和 T2)。

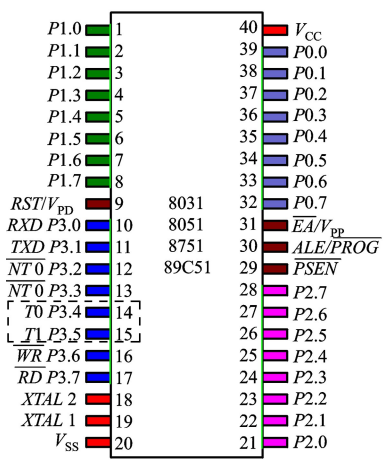

图 4-1 单片机定时器引脚位置

定时器的内部结构如图 4-2 所示。

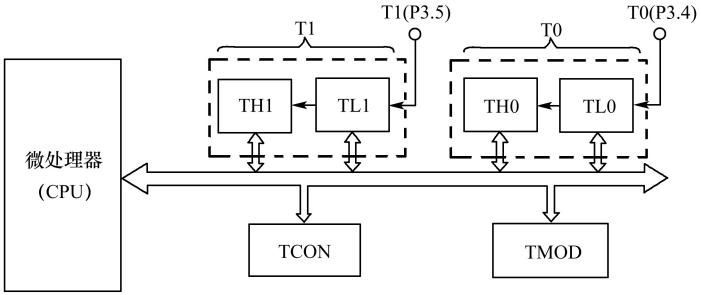

图 4-2 定时器内部结构

定时器功能由 T0 和 T1 以及它们的工作方式寄存器(TMOD)和控制寄存器(TCON)等组成。内部通过总线与 CPU 相连。定时器 T0 和 T1 各由两个 8 位特殊功能寄存器 TH0、TL0、TH1、TL1 构成。

工作方式寄存器:用于设置定时器的工作模式和工作方式。

控制寄存器:用于启动和停止定时器的计数,并控制定时器的状态。

定时器的工作方式、启动、停止、溢出标志、计数器等都是可编程的,通过设置寄存器 TMOD、TCON、TH0、TL0、TH1 和 TL1 实现。TH0 和 TL0 存放定时器 T0 的初值或计数结果。TH0 存放高 8 位,TL0 存放低 8 位;TH1 和 TL1 存放定时器 T1 的初值或计数结果。TH1 存放高 8 位,TL1 存放低 8 位。

4.2.2 定时器的 TMOD 和 TCON

1. TMOD

TMOD 格式如图 4-3 所示。

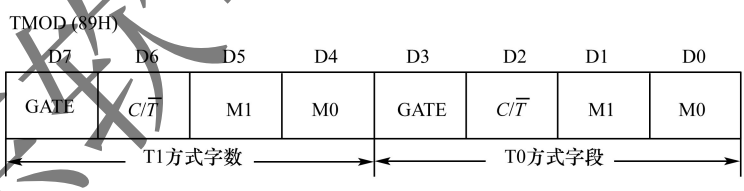

图 4-3 TMOD 格式

(1) GATE——门控位。

GATE=0:定时器的启动不受到外部中断请求信号的影响,一般情况下 GATE=0。

GATE=1:T0 的启动受到 /INT0 (P3.2)控制,T1 的启动还受到/INT1(P3.3)控制,只有当外部中断信号/INT0 和/INT1 为高电平时,才能启动定时器。

(2)M1、M0 —— 工作方式选择位。

(3)C/T* ——计数器模式和定时器模式选择位。

0:定时器模式。

1:计数器模式。

2. TCON

TCON 格式如图 4-4 所示。

D7	D6	D5	D4	D3	D2	D1	D0
TF1	TR1	TF0	TR0				

图 4-4　TCON 格式

低 4 位与外部中断有关。高 4 位的功能如下：

(1)TF1、TF0——计数溢出标志位。

定时器 T0 或 T1 计数溢出时，由硬件自动将此位置“1”；

TFx 可以由程序查询，也是定时中断的请求源。

(2)TR1、TR0——计数运行控制位。

TRx＝1：启动定时器/计数器工作。

Rx＝0：停止定时器/计数器工作。

4.2.3　定时器工作方式

MCS-51 的定时器 T0 有 4 种工作方式，即方式 0、方式 1、方式 2、方式 3。

MCS-51 的定时器 T1 有 3 种工作方式，即方式 0、方式 1、方式 2。

1. 方式 0

在这种方式下，16 位寄存器 TH1 和 TL1 只用 13 位，由 TH1 的 8 位和 TL1 的低 5 位组成。TL1 的高 3 位不定。

其定时时间为：

$$(2^{13}-\text{初值})\times\text{振荡周期}\times 12$$

例如：若晶振频率为 12MHz，则最长的定时时间为

$$(2^{13}-0)\times(1/12)\times 12\ \mu s=8.191\ ms$$

2. 方式 1

在这种方式下，16 位寄存器 TH1 和 TL1 为 16 位的计数器，除位数外，其他与方式 0 相同。

其定时时间为：

$$(2^{16}-\text{初值})\times\text{振荡周期}\times 12$$

例如：若晶振频率为 12MHz，则最长的定时时间为

$$(2^{16}-0)\times(1/12)\times 12\ \mu s=65.536\ ms$$

3. 方式 2

THx 作为常数缓冲器，当 TLx 计数溢出时，在置“1”溢出标志 TFx 的同时，还自动地将 THx 中的初值送至 TLx，使 TLx 从初值开始重新计数。

其定时时间为：

$$(2^{8}-\text{初值})\times\text{振荡周期}\times 12$$

若晶振频率为 12 MHz,则最长的定时时间为

$$(2^8-0)\times(1/12)\times 12\ \mu s=0.256\ ms$$

4. 方式 3

T0 在方式 3 时被拆成两个独立的 8 位计数器:TH0 和 TL0。

当 T0 处于方式 3 时,T1 仍可设置为方式 0、方式 1 和方式 2。此时由于 TR1、TF1 和 T1 的中断源都已被定时器 T0 占用,所以定时器 T1 仅由控制位 C/T 来决定其工作在定时方式或计数方式。当计数器计满溢出时,不能置位“TF1”,而只能将输出送往串口。所以,此时定时器 T1 一般用作串口的波特率发生器或不需要中断的场合。

4.2.4 定时器编程步骤

MCS-51 单片机的定时器/计数器是可编程的,具体步骤如下:

(1)对 TMOD 赋值,以确定定时器的工作模式。

设计数器的最大值为 M,则置入的初值 X 为:

①计数方式:X=M-计数值

②定时方式:由(M-X)T=定时值,得 X=M-定时值/T。其中 T 为计数周期,是单片机的机器周期。

(模式 0:M 为 $2^{13}=8192$,模式 1:M 为 $2^{16}=65536$,模式 2 和 3:M 为 $2^8=256$)

例如:机器时钟频率为 12MHz,机器周期为 1 μs 时,

若工作在模式 0,则最大定时值为:

$$2^{13}\times 1\ \mu s=8.192\ ms$$

若工作在模式 1,则最大定时值为:

$$2^{16}\times 1\ \mu s=65.536\ ms$$

(2)置定时器/计数器初值,直接将初值写入寄存器的 TH0、TL0 或 TH1、TL1。

(3)对 TCON 中的 TR0 或 TR1 置位,启动定时器/计数器。置位以后,计数器即按规定的工作模式和初值进行计数或开始定时。

(4)查询溢出标志 TFx 的状态,决定是否停止定时器/计数器。

4.3 项目实施

为了更好理解定时器和计数器的使用,本项目分成两个任务。任务 1 依旧采用前面的 8 路流水灯电路图 3-2,只是延时程序现在由定时器完成。任务 2 用 LED 灯实现二进制计数来掌握计数器的编程,程序有所变化,但是电路基本一样,只是增加了一个计数器的按钮。

4.4.1 任务 4-1 用定时器 T0 控制流水灯

我们依然采用图 3-2 的流水灯电路图进行仿真。要求 T0 工作方式 1,LED 灯闪烁周期 100 ms,即亮 50 ms,灭 50 ms。

1. 实现方法

(1)对 TMOD 赋值,以确定定时器的工作模式,如表 4-1 所示。

表 4-1

GATE	C/T	M1	M0	GATE	C/T	M1	M0
0	0	0	0	0	0	0	1

由此 TMOD=0x01;

(2)初值计算。

在方式 1 下,计数器的最大值为 $2^{16}=65536$,例如:机器时钟频率为 12 MHz,机器周期为 1 μs 时,则置入的初值为:

$$X=M-\text{定时值}/T=65536-50\ \text{ms}/1\ \mu\text{s}=65536-50000=15536$$

置定时器/计数器初值,直接将初值写入寄存器的 TH0、TL0 。利用移位的方式可以实现:

TH0=15536/256; TL0=15536%256;

(3)对 TCON 中的 TR0 置位,启动定时器/计数器,置位以后,计数器即按规定的工作模式和初值进行计数或开始定时。

(4)查询溢出标志 TF0 的状态。

2. 程序设计

```
//Ex4-1 用定时器 T0 查询方式控制 LED 灯闪烁
#include<reg51.h>
/*************************************
函数功能:主函数
*************************************/
void main(void)
{
    TMOD = 0x01;                    //使用定时器 T0 的模式 1
    TH0 = (65536 - 50000)/256;      //定时器 T0 的高 8 位赋初值
    TL0 = (65536 - 50000) % 256;    //定时器 T0 的低 8 位赋初值
    TR0 = 1;                        //启动定时器 T0
    TF0 = 0;
    P1 = 0xff;
    while(1)
    {
      while(TF0 = = 0) ;            //无限循环等待查询
        TF0 = 0;
        P1 = ~P1;
```

```
        TH0 = (65536 - 50000)/256;    //定时器 T0 的高 8 位赋初值
        TL0 = (65536 - 50000) % 256;  //定时器 T0 的低 8 位赋初值
    }
}
```

3. Proteus 仿真

将任务 4-1 对应的 HEX 装入单片机，可以看到 LED 灯开始闪烁，如图 4-4 所示。

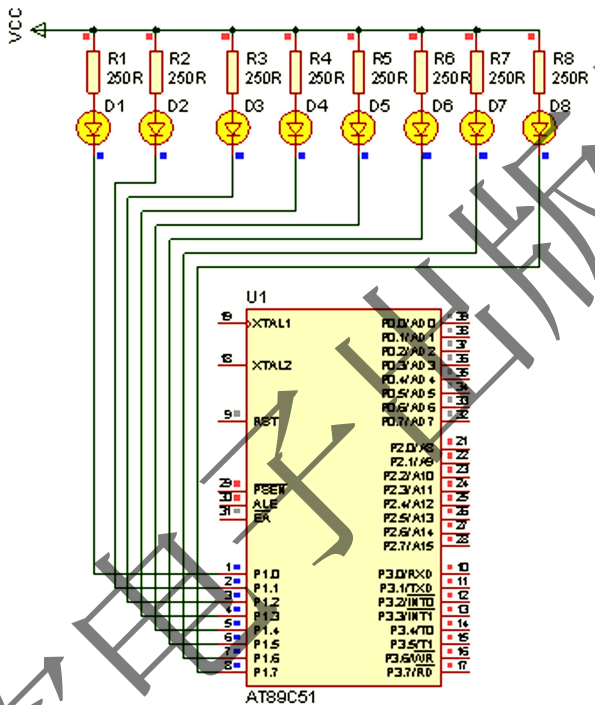

图 4-4　定时器 T0 控制 LED 灯闪烁

为了对比，我们将单片机的时钟频率由 12MHz 更改为 6MHz，可以看到闪烁频率明显减慢。

4.4.2　任务 4-2　T0 控制 LED 灯实现二进制计数

上面任务也可以实现第三章的流水灯功能，只是延时用定时器功能实现。计数器和定时器实质是一样的，任务 4-2 则是演示计数。所用电路图和图 3-2 类似，就是增加了一个计数按钮，如图 4-5 所示。

```
//Ex4-2 计数器 T0 实现二进制显示
#include<reg51.h>
/*********************************************
函数功能:主函数
**********************************************/
```

```
void main(void)
{
    TMOD = 0x06;  //使用计数器 T0 的模式 2
    TH0 = 0;  //定时器 T0 的高 8 位赋初值
    TL0 = 0;  //定时器 T0 的低 8 位赋初值
    TR0 = 1;  //启动定时器/计数器 T0
    while(1)  //无限循环等待查询
    {
        P1 = TL0; //计数器 TL0 加 1 后送 P1 口显示
    }
}
```

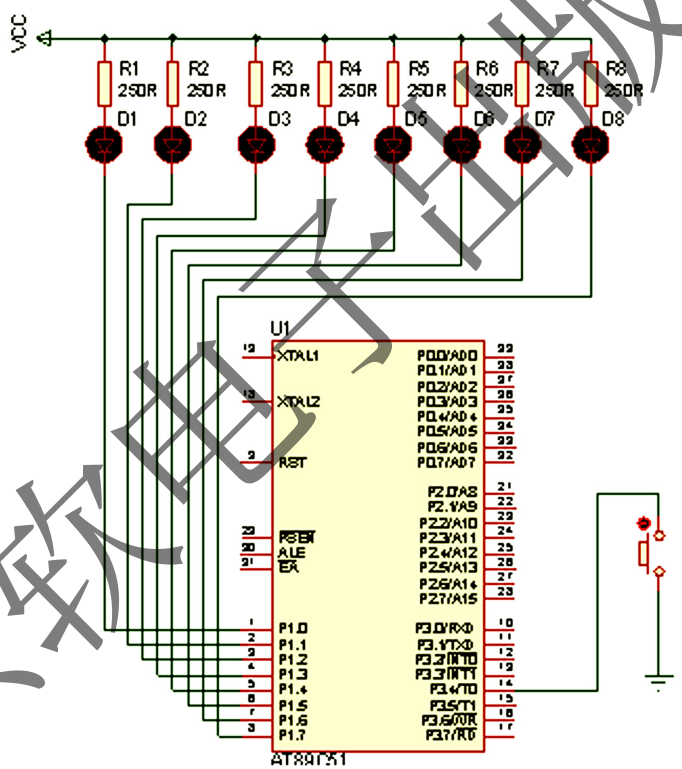

图 4-5　T0 控制 LED 灯实现二进制计数

本例对按键的计数没有使用查询法，没有使用外部中断函数，没有使用定时或计数中断函数，而是启用了计数器。当在工作方式 2 时，C/T＝1，M1/M0＝1/0，所以 TMOD＝0x06。计数器最大计数为 256，当计数溢出后，不用重新赋初值，TL0 自动取 TH0 的值作为初值。

启动仿真，如图 4-6 所示。所有灯都亮，表示对应 2 进制为 00000000。

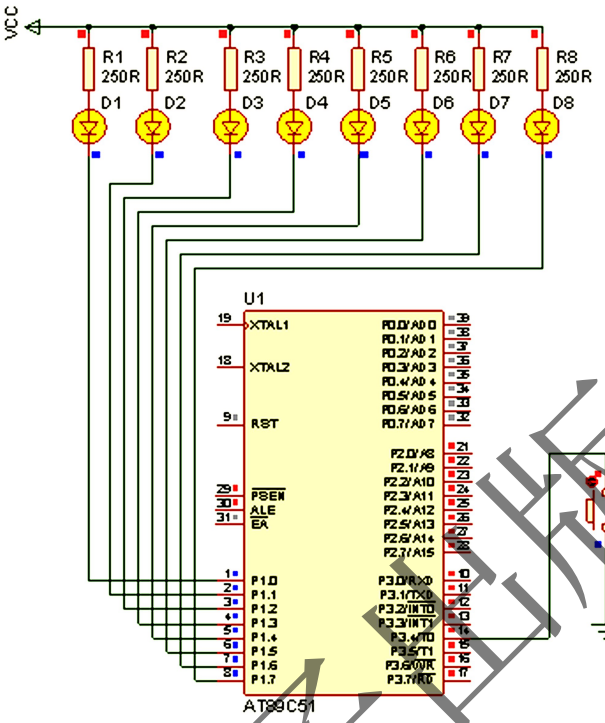

图 4-6　启动仿真初始计数

点击一下按钮，如图 4-7 所示。表示 2 进制 0000 0001。

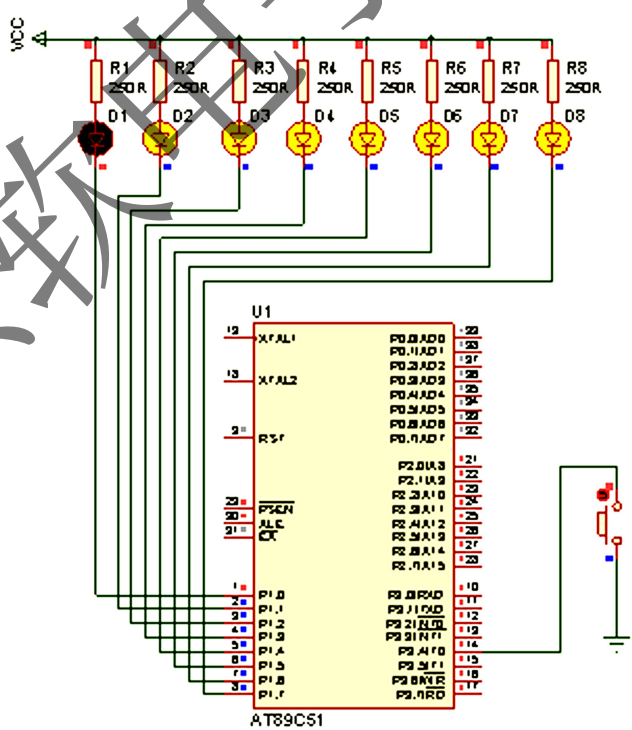

图 4-7　按下按钮进位一次

连接在 T0 引脚的按键每次按下时，会使计数寄存器的值递增，其值通过 LED 以二进制形式显示。其最大值为 2 进制 1111 1111，也就是 255。

4.4　技术拓展

采用查询的方法简单，但是在定时器整个计数的过程中，CPU 要不断地查询溢出标志 TFx 的状态，很难执行其他操作，占用了 CPU 的工作时间，使得 CPU 的工作效率不高。若采用中断的方式来实现，可大大提高 CPU 的工作效率。

我们下一章学习中断，学习完之后再返回来分析和比较两者的区别与特点。

4.5　本章小结

本章主要讲述了定时器/计数器的非中断使用方法。定时器/计数器应用场合主要在定时或延时控制，对外部事件的检测、计数等。主要通过对 TMOD 和 TCON 的设置，在不同工作方式下实现定时或计数。

4.6　强化练习

1. 设计程序实现对一路 LED 灯进行 500 ms 的定时亮灭控制。

2. 修改任务 4-2 在方式 1 情况下，16 位的二进制计数。

3. 按照图 3-2 的电路图，实现以下功能：

(1) P1 口高四位以 0.2 s 周期闪烁(亮 0.1 s 后灭 0.1 s)，低四位以 0.5 s 周期闪烁。

(2) P1 口从左到右的流水灯功能。