实验 7 - 定时器实验

1. 实验目的

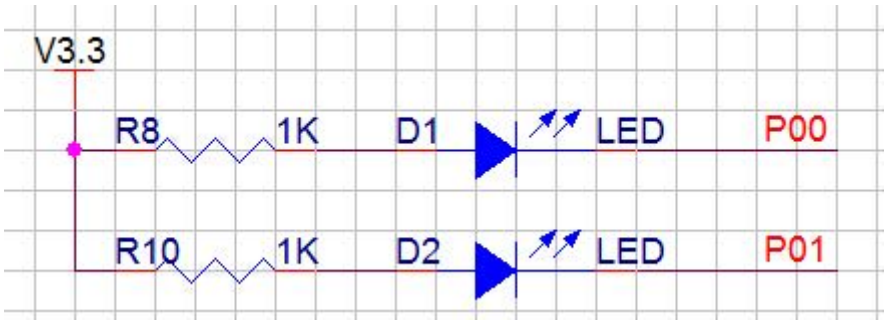
掌握 NRF24LE1 的定时器的配置和使用。

2. 实验内容

配置 NRF24LE1 的 GPIO P0.0 输出控制指示灯 D1 的亮灭。
配置定时器 0 每 20ms 产生一次中断。
在定时器 0 中断中每隔 500ms 将 D1 状态取反。
观察到 D1 以 500ms 的间隔闪烁。

3. 实验原理

3.1. 电路原理



开发板上配置的两个用户指示灯 D1、D2，分别有 GPIO P0.0 和 P0.1 控制，当 GPIO 输出高电平时，LED 两端电压相等，LED 上没有电流流过，LED 处于灭状态，当 GPIO 输出低电平时，LED 两端存在压差，电流流过 LED，LED 被点亮。

3.2. 寄存器配置

1. GPIO 寄存器配置

NRF24LE1 的 GPIO 通过 2 个寄存器来配置：PxDIR 和 PxCON(更详细的内容请查阅 NRF24LE1 数据手册)。

- PxDIR：设置 IO 的方向。
- PxCON：设置 IO 的功能。

表 1：P0DIR 寄存器（地址：0x93，复位值：0xFF）

位	名称	R/W	功能
7~0	方向	R/W	P0.0~P0.7 方向位。输出：dir=0，输入：dir=1. P0DIR 0 – P0.0 P0DIR 1 – P0.1

			P0DIR 2 – P0.2 P0DIR 3 – P0.3 P0DIR 4 – P0.4 P0DIR 5 – P0.5 P0DIR 6 – P0.6 P0DIR 7 – P0.7

按照上述内容，对 P0.0 进行配置如下：

```
P0DIR &= ~0x01;     //配置 P0.0 为输出
D1 = 1;     //设置 D1 初始状态为熄灭
```

P0CON：采用默认值即可。

2. Timer0 寄存器配置

Timer0 的配置涉及到如下寄存器：

- TCON：定时器/计数器控制寄存器；
- TMOD：定时器模式寄存器；
- Timer 0：TH0, TL0。

表 2：TCON 寄存器

地址	复位值	位	名称	自动清除	说明
0x88	0x00	7	tf1	是	Timer1 溢出标志，当 Timer1 溢出时由硬件置位。
		6	tr1	否	Timer1 运行控制位，清零时，Timer1 停止工作。
		5	tf0	是	Timer0 溢出标志，当 Timer0 溢出时由硬件置位。
		4	tr0	否	Timer0 运行控制位，清零时，Timer0 停止工作。
		3	ie1	是	外部中断 1 标志，由硬件置位。
		2	it1	否	外部中断 1 类型控制。1：下降沿触发，0：低电平触发。
		1	ie0	是	外部中断 0 标志，由硬件置位。
		0	it0	否	外部中断 0 类型控制。1：下降沿触发，0：低电平触发。

表 2：TMOD 寄存器

地址	复位值	位	名称	说明
0x89	0x00	7	gate1	Timer1 门控控制
		6	ct1	Timer1 计数器/定时器选择。1：计数器，0：定时器。
		5~4	mode1	Timer1 模式： 00—模式 0：13 位计数器/定时器。 01—模式 1：16 位计数器/定时器。 10—模式 2：8 位自动重装定时器。

				11—模式 3：两个 8 位定时器/计数器。
		3	gate0	Timer0 门控控制
		2	ct10	Timer0 计数器/定时器选择。1：计数器，0：定时器。
		1~0	mode0	Timer0 模式： 00—模式 0：13 位计数器/定时器。 01—模式 1：16 位计数器/定时器。 10—模式 2：8 位自动重装定时器。 11—模式 3：两个 8 位定时器/计数器。

表 4：Timer 0 – TH0, TL0 寄存器

地址	名称
0x8A	TL0
0x8C	TH0

按照上述内容，配置 Timer0 为 16 位定时器，定时时间为 50ms，产生溢出中断的代码如下：

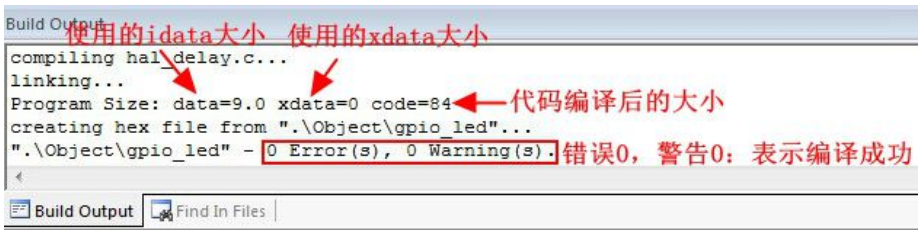
```
TMOD = 0x01;                          //设置 Timer0 工作在模式 1，即 16 位定时器
TH0  = (65536-TIMER0_VALUE)/256;      //写入定时器初值
TL0  = (65536-TIMER0_VALUE)%256;
ET0  = 1;           //使能 Timer0 溢出中断
EA   = 1;           //开启全局中断
TR0  = 1;           //启动 Timer0
```

定时器初值计算

- **计算方式：**先计算出定时时间，再用 65536（16 位定时器）-定时时间对应的数值，即得到定时器初值。
- **计算过程：**
  - 系统时钟：16MHz
  - 定时器使用的是系统时钟的 12 分频
  - 假设预定时的时间为 T，那么对应的数值为：$T \times \frac{16 \times 10^6}{12}$，其中 T 的单位为秒。

4. 实验步骤

- 在 Keil uVision4 中打开工程“Timer0.uvproj”工程；
- 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；

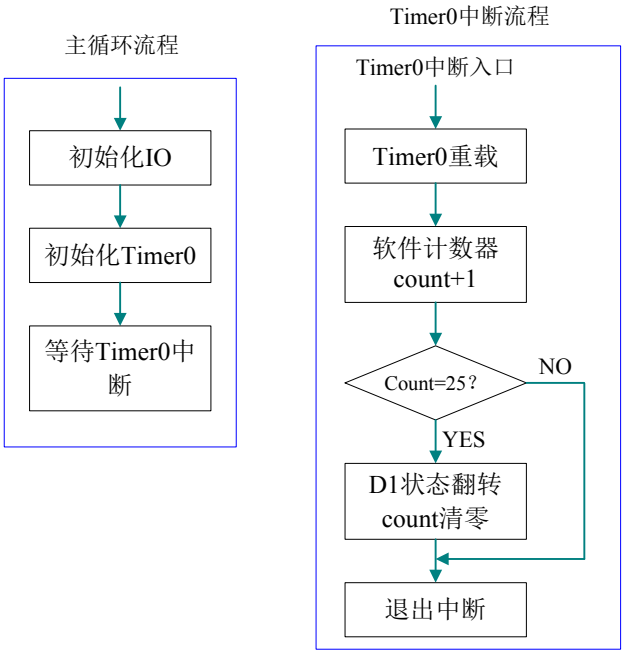


- 将编译生成的 HEX 文件“Timer0.hex” (该文件位于工程目录下的“Object”文件夹中)通过编程器下载到开发板中运行。
- 观察指示灯 D1，应以 500ms 的间隔闪烁。

5. 实验程序

5.1. 程序流程

定时器实验程序执行流程如下图所示：



5.2. 程序清单

```
#define   D1      P00   //开发板上的指示灯 D1
#define TIMER0_VALUE   26666   //Timer0 定时器定时 20ms 对应的数值

uint8_t count=0;    //软件计数变量

/*******************************************************************************
 *描   述 : 配置 IO P0.0
 *入   参 : 无
```

```
 *返回值 ：无
*******************************************************************************/
void IO_Init(void)
{
    P0DIR &= ~0x01;    //配置 P0.0 为输出
    D1 = 1;    //设置 D1 初始状态为熄灭
}
/*******************************************************************************
 *描   述 ：配置 Timer0 为 16 位定时器，定时时间 20ms，开启中断
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void Timer0Init(void)
{
  TMOD = 0x01;                          //16 位定时器
  TH0  = (65536-TIMER0_VALUE)/256;  //写入初值
  TL0  = (65536-TIMER0_VALUE)%256;
  ET0  = 1;          //使能 Timer0 溢出中断
  EA   = 1;          //使能全局中断
  TR0  = 1;          //启动 Timer0
}

/*******************************************************************************
 *描   述 ：主函数
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void main(void)
{
  IO_Init();        //配置 IO
  Timer0Init();    //Timer0 初始化

  while(1);         //死循环，等待 Timer0 溢出中断
}
/*******************************************************************************
 *描   述 :Timer0 中断服务函数
 *入   参 ：无
 *返回值 ：无
*******************************************************************************/
void Timer0_irq() interrupt INTERRUPT_T0
{

  TH0=(65536-TIMER0_VALUE)/256;   //写入初值
```

```
    TL0=(65536-TIMER0_VALUE)%256;
    count++;              //软件计数器加 1
    if(count==25)     //500ms 时间到
    {
        count=0;          //软件计数器清零
        D1 = ~D1;      //D1 指示灯状态取反
    }
}
```