# 第五章 80C51单片机中断与定时

帅千钧

Email:sqj@cuc.edu.cn

办公室：主楼813

# 本章内容

- 5.1 80C51中断系统
  - 输入/输出(I/O)控制方式
  - 中断技术概述
  - 51单片机的中断系统
  - 中断应用
- 5.2 80C51定时器/计数器
  - 定时器/计数器的结构和功能
  - 与定时/计数器相关的寄存器
  - 定时器/计数器的工作方式
  - 应用举例

# 5.1.1 输入/输出(I/O)寻址方式

- **专用I/O地址方式**

  有专用I/O控制信号和I/O指令。I/O接口独立编址，不占用存储器的地址空间。

- **存储器地址方式**

  I/O接口共用存储器的地址空间，每个I/O端口视为一个存储单元。

  ☞ **MCS-51、96为存储器地址方式!**

  - MCS-51单片机有片内I/O接口和扩展I/O接口：
  - 片内I/O接口寄存器在SFR中，使用片内数据存储器空间
  - 扩展I/O接口使用片外数据存储器地址空间：

| | 输出指令： | 输入指令： |
|---|---|---|
| 片内寻址： | MOV　P1，A | MOV　A，P1 |
| 片外寻址： | MOVX @DPTR，A | MOVX A，@DPTR |
| | MOVX @R0，A | MOVX A，@R0 |

# 5.1.1 输入/输出(I/O)操作控制方式

- CPU与外设之间传送数据的方式有以下几种：
  1. 无条件传送方式
  2. 程序查询方式
  3. 中断控制方式
  4. DMA方式

# 5.1.1 输入/输出(I/O)操作控制方式

1. 无条件传送方式

- 处理器认为外设一定是准备好的，通过程序直接向指定的I/O端口执行输入/输出操作。

优点：硬件电路和软件编程都简单

缺点：可靠性不高，容易出错。

2. 程序查询方式

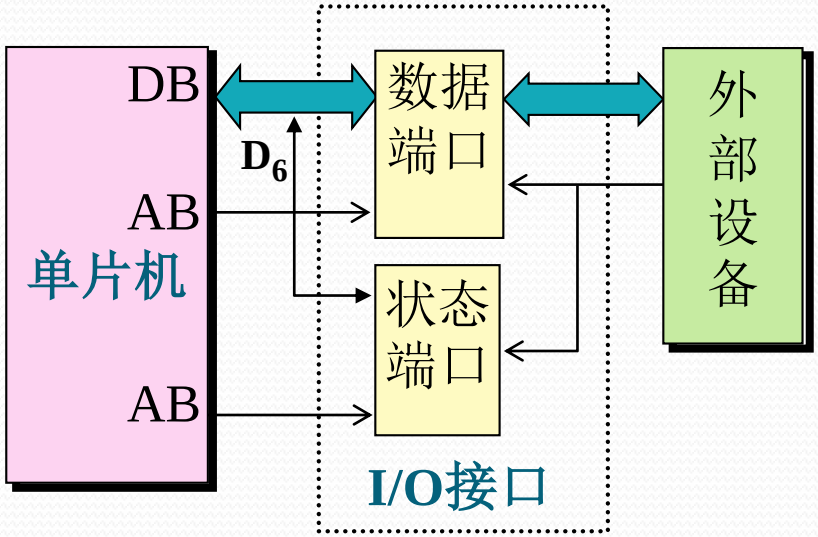
- 先查询I/O设备当前状态，若准备就绪，则交换数据，否则循环查询状态。

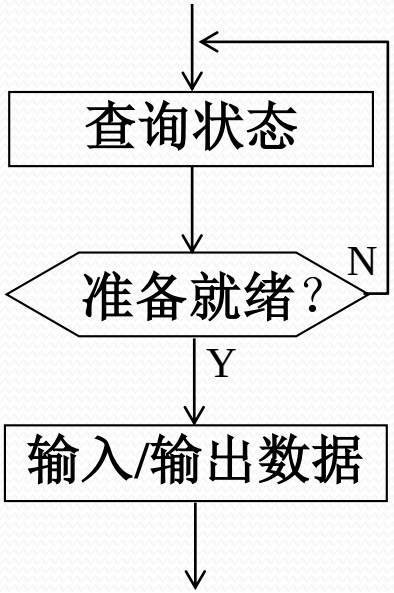
特点：I/O操作(包括启动、控制和输入/输出)都由CPU控制。

优点：硬件电路简单

缺点：浪费CPU时间。

# 程序查询方式：



# 5.1.1 输入/输出(I/O)操作控制方式

3. 中断控制方式

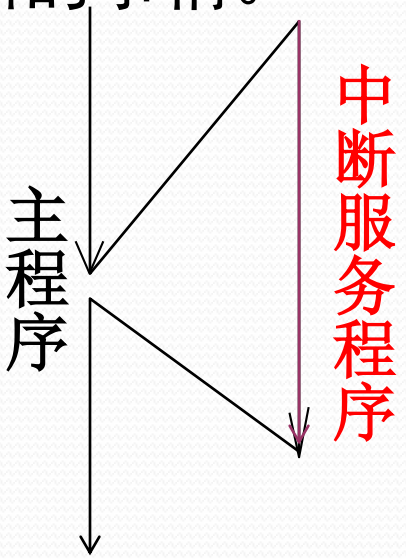
- 处理器与外设并行工作。处理器在正常运行程序时，由于外设准备就绪，向CPU发出中断请求信号，使得处理器中断正在执行的程序，转而处理中断服务；处理完后，仍继续被中断的程序。

4. DMA方式

- (DMA：Direct Memory Access)直接存储器存取，用于计算机与高速外设进行大批量数据交换，由DMA控制器接管总线控制权，RAM与外设之间直接进行数据传输，而不需CPU的介入。

# 5.1.2 中断技术概述

- 中断的概念
  - 中断系统是计算机的重要指标之一！
  - 中断指CPU在正常运行程序时，由于CPU以外某一事件的发生，引起了CPU中断正在执行的程序，而转去为预先安排的该事件服务的程序中去执行。处理完成后，能够返回到断点处继续执行被中断的事情。这一过程就是中断。
  - 这些引起程序中断的事件称为中断源。
  - 中断源要求为其服务的请求称为中断请求。
  - 申请处理的事情，称为中断服务程序。
  - 主程序被中断的地方，称为断点。



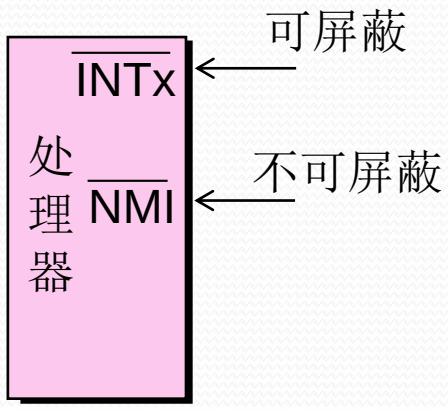
## 两种中断类型

1. 可屏蔽中断

   可程序控制“开中断/关中断”，软件设置允许/禁止CPU响应中断

2. 非屏蔽中断

   不可程序控制“关中断”，若有中断请求信号，CPU必须响应。



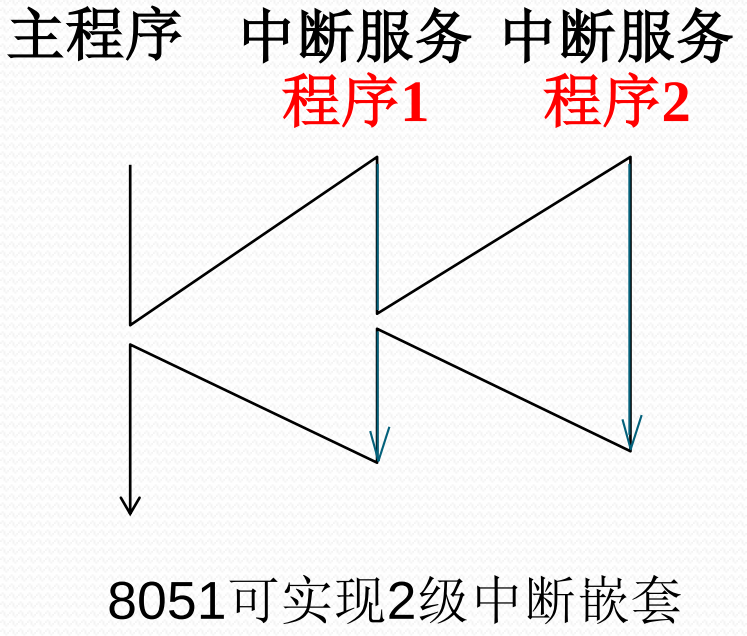

## 中断源

- 能发出中断请求信号的各种事件，如：
  - I/O设备
  - 定时/计数溢出
  - 系统故障
  - 软件设定

每个中断源对应一个中断服务程序。

# 5.1.2 中断技术概述

- 中断优先权
  - 当同时有多个中断请求信号，先响应优先级别高的中断请求。
  - 高优先级中断请求信号，可中断低优先级中断服务。
  - 同级别优先权的中断请求信号，不能中断正在执行的中断服务。




8051可实现2级中断嵌套

# 5.1.2 中断技术概述

## 中断响应条件

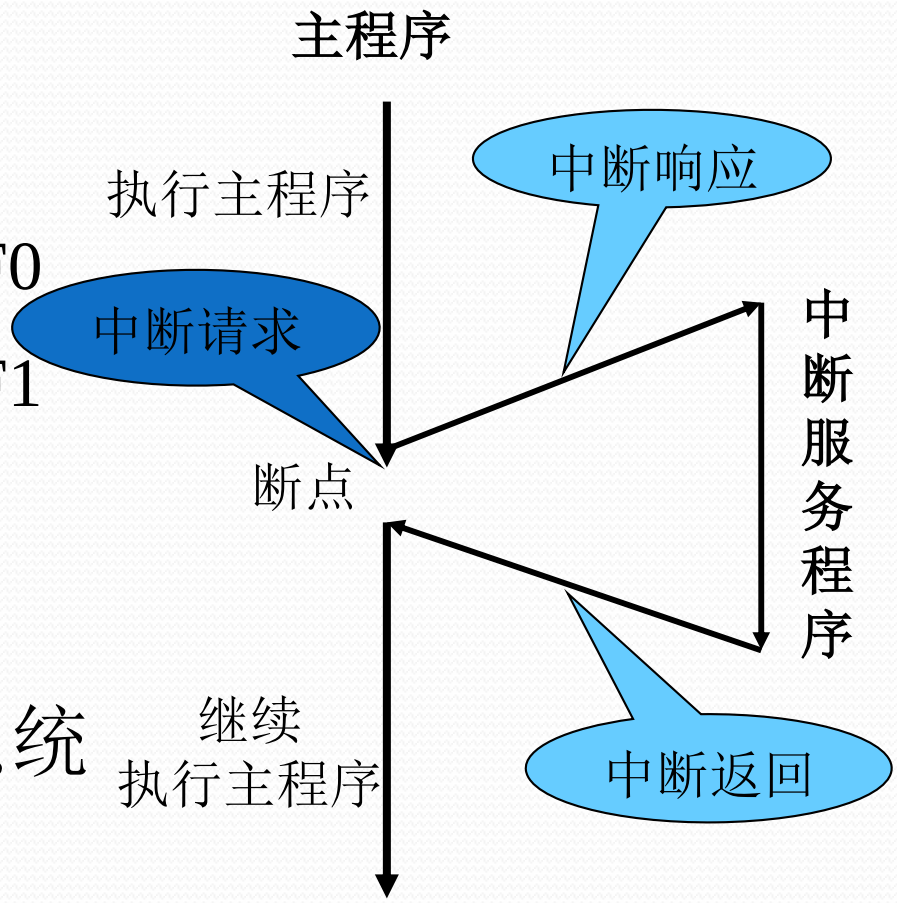
1. 有中断请求信号
2. 系统处于开中断状态
3. 中断标志位置1

## 中断处理过程

1. 中断请求
2. 中断优先权判决
3. 中断响应
4. 中断处理
5. 中断返回

# 5.1.3 8051单片机的中断系统

- 51子系列单片机的中断系统有6个中断源：
  - 外部中断$\overline{\text{INT0}}$
  - 外部中断$\overline{\text{INT1}}$
  - 定时/计数器T0溢出中断TF0
  - 定时/计数器T1溢出中断TF1
  - 串行口中断TI
  - 串行口中断RI
- 51子系列单片机的中断系统有5个中断入口（向量）。

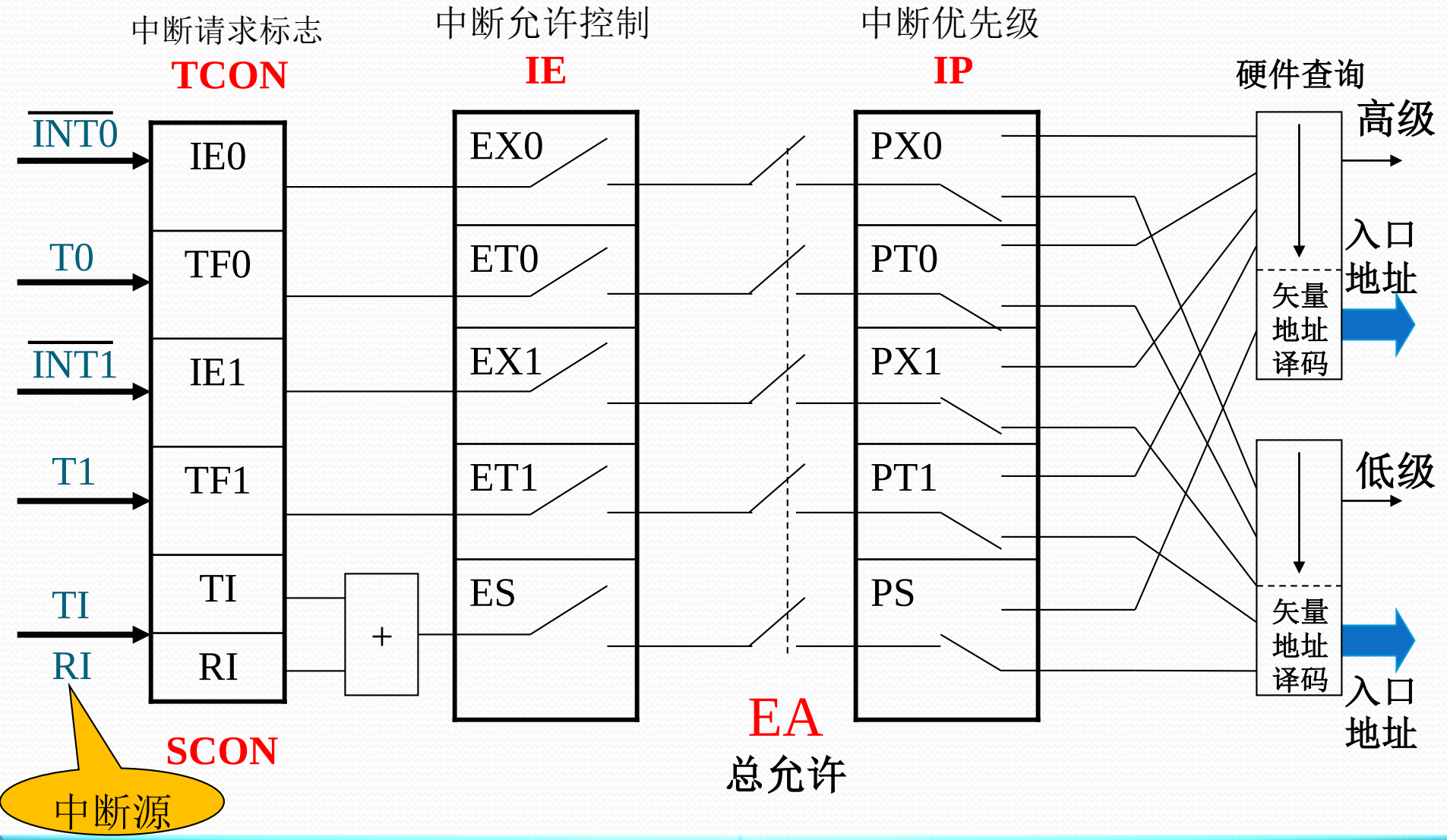


# 5.1.3 8051单片机的中断系统

- 中断源6个：
  1. 外部中断0—$\overline{\text{INT0}}$，由P3.2提供
  2. 外部中断1—$\overline{\text{INT1}}$，由P3.3提供
  3. T0溢出中断—TF0，由片内定时/计数器0提供
  4. T1溢出中断—TF1，由片内定时/计数器1提供
  5. 串行口中断RI，由片内串行口提供
  6. 串行口中断TI，由片内串行口提供

  ✵ 外部中断有两种信号方式：电平方式和脉冲方式。
- 中断控制
  - 中断请求标志
  - 中断允许控制和总允许控制
  - 中断优先级控制

# 5.1.3 8051单片机的中断系统结构(书图5.1)



# 5.1.3 8051单片机中断入口方式

- 单片机的中断为固定入口式中断，即一响应中断就转入固定入口地址执行中断服务程序。入口地址：

| *中断源* | *入口地址* |
|---|---|
| $\overline{\text{INT0}}$ | 0003H |
| TF0 | 000BH |
| $\overline{\text{INT1}}$ | 0013H |
| TF1 | 001BH |
| RI/TI | 0023H |

- 每个中断源入口空间只有8个单元。
- 在这些单元中，往往存放跳转指令，使其跳到真正的中断服务程序。

# 5.1.3 与中断控制有关的寄存器

- 80C51与中断控制有关的控制寄存器有四个：
  - IE，中断允许控制寄存器
  - TCON，定时控制寄存器
  - SCON，串行口控制寄存器
  - IP，中断优先级控制寄存器

# 5.1.3 与中断控制有关的寄存器

1. IE中断允许控制寄存器 (A8H)

| IE A8H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | EA | | | ES | ET1 | EX1 | ET0 | EX0 |
| 位地址 | AFH | | | ACH | ABH | AAH | A9H | A8H |

- EA：CPU中断总允许位

  EA=1，CPU开放中断

  EA=0，CPU禁止中断
- ES：串行口中断允许位

  ES=1，允许串行中断

  ES=0，禁止串行中断

- ET0、ET1：定时器T0、T1溢出中断允许位

  ET0=1，允许定时器T0中断

  ET1=1，允许定时器T1中断

- EX0、EX1：外部中断INT0、INT1允许位

  EX0=1，允许外部中断INT0

  EX1=1，允许外部中断INT1

**结论：**

- **MCS-51通过中断允许控制寄存器对中断的允许实行两级控制：以EA位作为总控制位，以各中断源的中断允许位作为分控制位。**
- **MCS-51单片机复位后(IE)＝00H，因此，复位时中断系统处于禁止状态。**
- 单片机在中断响应后不会自动关闭中断。因此在转中断服务程序后，应使用有关指令禁止中断，即以软件方式关闭中断。

## 2. TCON定时控制寄存器（88H）

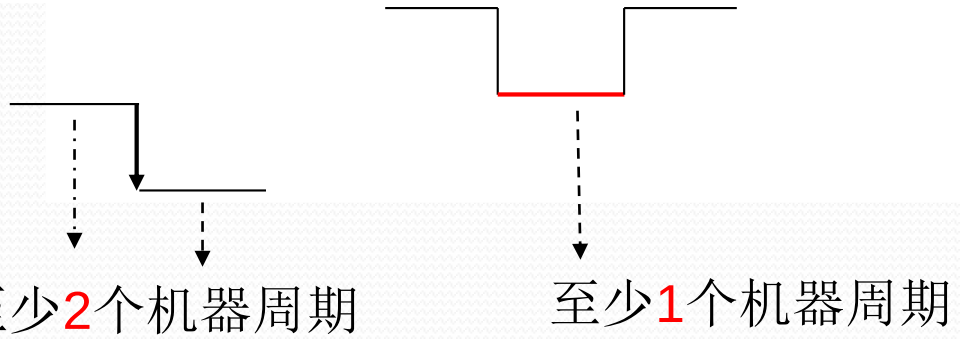
TCON
88H

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

中断请求标志

触发方式选择 0 低电平
1 下降沿

- IT0、IT1：相应外部中断(INT0、INT1)触发方式控制位；
  - IT*x*=0，INTx为电平触发方式，低电平有效；
  - IT*x*=1，INTx为脉冲触发方式，下降沿有效；



- IE0、IE1：相应外部中断(INT0、INT1)请求标志位；
  - IE0=1，外部中断0向CPU申请中断
  - IE1=1，外部中断1向CPU申请中断
- 外部中断由硬件自动置位和清零：
  - CPU采样到INT*x*端有效的中断请求后，IE*x*位由硬件自动置 1。
  - 在中断响应期间（要转向中断服务程序时），由硬件自动清零。

- TF0、TF1：定时器溢出中断请求标志位；
- 相应溢出标志位TF*x*由硬件置1。两种使用方式：

a) 软件查询时，作为溢出标志位使用，查询有效后，用户软件清零☆。

b) 中断方式时，作为中断请求标志位使用，为“1”时，自动转向中断服务程序，再由硬件自动清零。

```
LOOP：JBC   TF0，NEXT
      SJMP  LOOP
```

- TR0、TR1：定时器启动标志位。

## 3. SCON串行口控制寄存器 (98H)

| SCON 98H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | TI | RI |
| 位地址 | | | | | | | 99H | 98H |

- TI：串行口发送中断标志位
  - 当CPU将一个数据写入发送缓冲器时，就启动发送。
  - 当最后一个数据位发送完后，TI由硬件自动置位，向CPU发中断申请。
  - 进行中断服务程序时，TI要由软件清0 ☆ 。
  - 软件查询时，TI可作为状态位使用。

- RI：串行口接收中断标志位
  - CPU接收数据时，当接收到最后一个数据位后，RI由硬件自动置位，并向CPU发中断申请。
  - 进行中断服务程序时，RI要由软件清0 ☆ 。
  - 软件查询时，RI可作为状态位使用。

# 5.1.3 与中断控制有关的寄存器

4. IP中断优先级寄存器 (B8H)

| IP<br>**B8H** | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | | | | PS | PT1 | PX1 | PT0 | PX0 |
| **位地址** | | | | BCH | BBH | BAH | B9H | B8H |

- PS：串行口中断优先级控制位

  PS=1，置1为高优先级

  PS=0，清0为低优先级
- PT0、PT1：片内定时器T0、T1中断优先级控制位
- PX0、PX1：外部中断INT0、INT1中断优先级控制位

# 5.1.3 与中断控制有关的寄存器

- 用户通过TCON、SCON、IE和IP四个控制寄存器来使用中断系统。
- 这四个控制寄存器都是既可进行字节寻址又可进行位寻址的。因此，对位状态的设置既可以使用字节操作指令又可以使用位操作指令。
- 字节操作指令：**MOV IE,#81H** ; EA, /, /, ES, ET1, EX1, ET0, EX0
- 位操作指令：

```
SETB EA
SETB EX0
```

# 5.1.3中断优先级控制原则和控制逻辑

- 8051单片机具有两级优先级，其中断优先级的控制原则是：
  1. 低优先级中断请求不能打断高优先级的中断服务；但高优先级中断可以中断低优先级的中断服务，从而实现中断嵌套。
  2. 如果一个中断请求已被响应，则同级的其他中断服务将被禁止，即同级不能嵌套。
  3. 如果同级的多个中断请求同时出现，则按CPU查询次序确定哪个中断请求被响应。其查询次序为：外部中断0 → 定时中断0 → 外部中断1 → 定时中断1 → 串行中断。

# 同级中断源自然优先级顺序

| 中断源 | 入口地址 | 同级内的优先级 |
| --- | --- | --- |
| 外部中断0(IE0) | 0003H | 高优先级 |
| 定时器T0溢出中断(TF0) | 000BH | ↓ |
| 外部中断1(IE0) | 0013H | |
| 定时器T1溢出中断(TF1) | 001BH | |
| 串行口中断(TI/RI) | 0023H | 低优先级 |

# 5.1.3 中断响应过程

- 中断响应：在满足CPU中断响应条件后，CPU对中断源中断请求的应答。

一. 中断响应基本条件(来中断，开中断）

a. 有中断源发出中断申请（中断采样S5P2、中断查询S6）

- 中断采样：采样主要针对外中断请求信号进行，对芯片引脚/INT0和/INT1进行采样----电平方式或是脉冲方式。
- 中断查询：由CPU检测TCON和SCON中各标志位的状态，以确定有没有中断请求发生以及是哪一个中断请求；
- 中断请求是随机发生，要求CPU不停地重复进行查询。

b. 中断总允许位EA＝1

c. 各个中断源相应的允许位置1，即ES、ET1、EX1、ET0、EX0。

## 二. 中断响应可能会被阻止或被延迟：

中断响应是有条件的，并不是查询到的所有中断请求都能被立即响应，当存在下列情况之一时，中断响应被封锁。

a. **CPU正处在为一个同级或高级的中断服务中。**

b. **查询中断请求的机器周期不是当前指令的最后一个机器周期，即当前指令完成前，不会响应任何中断请求。**

c. **当前指令是返回指令(RET，RETl)或访问IE、IP的指令。MCS-5l中断系统规定，在执行完这些指令之后，还应再继续执行一条指令，然后才能响应中断。**

# 三. 中断响应过程

- 中断响应过程
  - ♦ 优先级状态触发器置1
  - ♦ 硬件清除中断请求标志
  - ♦ 转入中断服务程序

如果满足中断响应条件且不存在
中断响应周期，完成以下工作：

**a.** 将响应的优先级状态触发器置1。

**b.** 由硬件清除相应的中断请求标志位：$TF_X$，$IE_X$。 但串行中断标志位必须手动清除。

**c.** 执行一条由硬件生成的长调用指令LCALL。其格式为LCALL addrl6，这里的addrl6就是程序存储器中相应中断区的入口地址。该指令将自动把断点地址（当前PC值）压入堆栈保护起来，然后将对应的中断入口地址装入PC，使程序转向该中断入口地址，去执行中断服务程序。

注意：中断服务程序入口地址仅间隔8个字节，因此通常在这些入口地址处存放一条无条件转移指令，以控制程序转到用户安排好的中断服务程序地址去执行。

## 四. 中断响应时间：

从中断请求产生到CPU转到相应的中断服务程序的入口地址所需的时间。

1、基本响应时间（3个机器周期）

a. 查询中断标志：1个机器周期；

b. 转去执行中断服务程序：2个机器周期

2、额外的处理时间（0～5 个机器周期）

a. RETI指令：2个机器周期；

b. MUL或DIV指令：4个机器周期。

1. MCS－51单片机所需最短响应时间为3个机器周期。

中断请求标志位查询占1个机器周期（每个机器周期的S5P2期间查询中断标志位）。若此时恰好是正在执行指令的最后一个机器周期，中断立即会在该机器周期结束后被响应，硬件自动产生LCALL指令，耗费2个机器周期。因此总共3个机器周期。

2. 最长响应时间为8个机器周期。

查询中断标志位时，恰好开始执行(RET，RETl)或访问IE、IP的指令，最长需要2个机器周期；单片机要求必须再继续执行一条指令，然后才能响应中断。若再执行的最长是4个机器周期的指令（MUL AB），再加上执行LCALL的2个机器周期，总共是8个机器周期。

即一般的中断响应时间都是3～8个机器周期。

# 5.1.3 中断服务程序

- 中断服务程序：从入口地址开始执行到返回指令**RETI**为止。

1. 现场保护和现场恢复

所谓现场，指中断时刻单片机存储单元中的数据或状态，一定是主程序和中断程序都用到的且不能被修改的存储单元。

- 现场保护：位于中断处理程序首部，**PUSH** 操作数
- 现场恢复：位于中断处理程序尾部，**POP** 操作数

2. 开中断和关中断

- 在中断处理程序中，关中断，禁止响应其他一切中断，无中断嵌套
- 要保持中断嵌套，就要分别在现场保护和现场恢复前后，加上关中断和开中断，避免现场被破坏。

3. 中断返回(TETI)

- 清除优先级状态
- 返回PC

- 中断服务程序一般格式：

```
CLR  EA          ;关中断
PUSH  PSW        ;保护现场
PUSH  A
SETB  EA         ;开中断
…
CLR  EA          ;关中断
POP  PSW         ;恢复现场
POP  A
SETB  EA         ;开中断
RETI
```

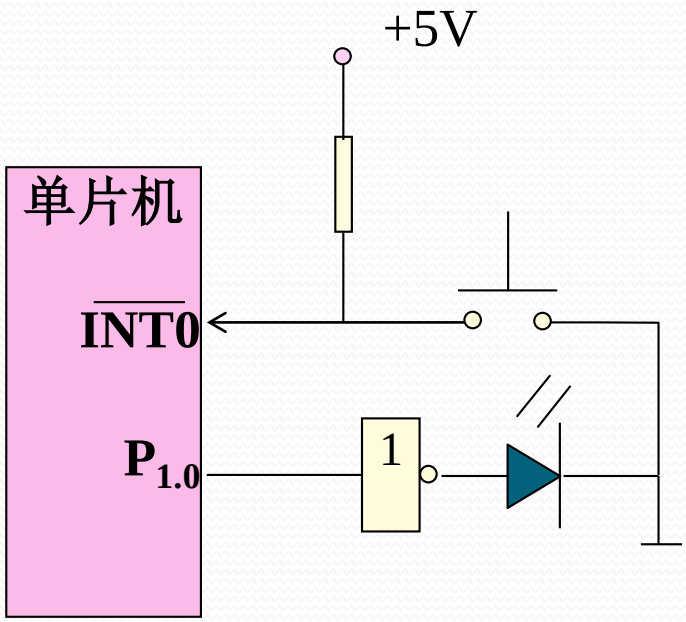
# 外部中断处理举例

例：要求每次按动按键，使外接发光二极管LED改变一次亮灭状态。

解：$\overline{\text{INT0}}$输入按键信号，$P_{1.0}$输出改变LED状态。

1．跳变触发：每次跳变引起一次中断请求。

```
        ORG     0000H；复位入口
        AJMP MAIN
        ORG     0003H ；中断入口
        AJMP    PINT0
        ORG     0100H ；主程序
MAIN：MOV SP，#40H；设栈底
        SETB    EA      ；开总允许开关
        SETB    EX0     ；开INT0中断
        SETB    IT0     ；负跳变触发中断
H：     SJMP    H       ；执行其它任务
        ORG     0200H ；中断服务程序
PINT0：CPL  P1.0        ；改变LED
        RETI            ；返回主程序
```



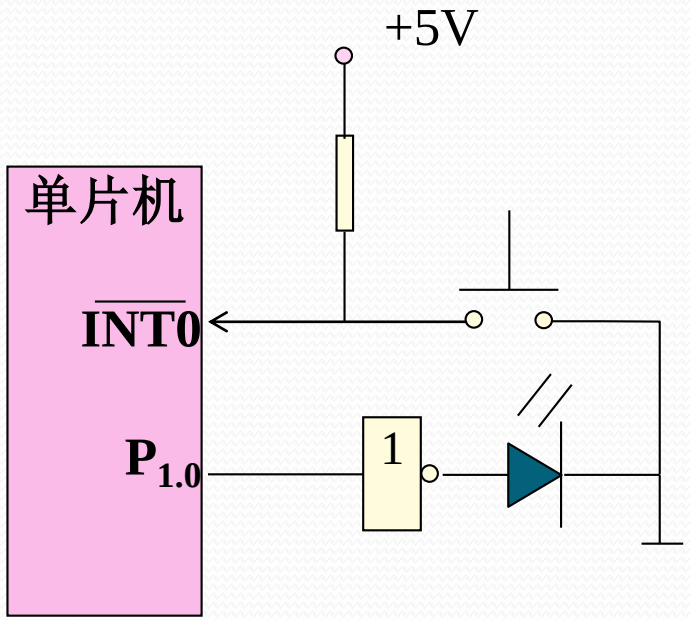
```
ORG     0000H；复位入口
        AJMP MAIN
        ORG     0003H    ；中断入口
        AJMP    PINT0
        ORG     0100H    ；主程序
MAIN：  MOV SP，#40H     ；设栈底
        SETB    EA       ；开总允许开关
        SETB    EX0      ；开INT0中断
        SETB    IT0      ；负跳变触发中断
        NOP
        MOV     A，#00H
        MOV     R0，A
        DEC     A
        MOV     R1，A
        。。。
H:      SJMP    H        ；执行其它任务

        ORG     0200H    ；中断服务程序
PINT0：CPL  P1.0         ；改变LED
        RETI             ；返回主程序
        END
```

2. 电平触发：避免一次按键引起多次中断响应。

1．软件等待按键释放。

2．硬件清除中断信号。

```
        ORG     0000H；复位入口
        AJMP MAIN
        ORG     0003H ；中断入口
        AJMP    PINT0
        ORG     0100H ；主程序
MAIN：MOV SP，#40H；设栈底
        SETB    EA      ；开总允许开关
        SETB    EX0     ；开INT0中断
        CLR     IT0     ；低电平触发中断
H：     SJMP    H       ；执行其它任务
        ORG     0200H ；中断服务程序
PINT0：CPL  P1.0        ；改变LED
WAIT：JNB       P3.2，WAIT；等按键释放
        RETI            ；返回主程序
```



# 5.1.4 中断相关的寄存器

| TCON 88H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| 位地址 | 8FH | 8EH | 8DH | 8CH | 8BH | 8AH | 89H | 88H |

| SCON 98H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | TI | RI |
| 位地址 | | | | | | | 99H | 98H |

| IE A8H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | EA | | | ES | ET1 | EX1 | ET0 | EX0 |
| 位地址 | AFH | | | ACH | ABH | AAH | A9H | A8H |

| IP B8H | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| | | | | PS | PT1 | PX1 | PT0 | PX0 |
| 位地址 | | | | BCH | BBH | BAH | B9H | B8H |

# 5.1.4 中断应用举例

- 中断服务程序要注意的问题
  1. 保护现场。
  2. 中断服务程序的入栈和出栈指令应是成对使用。
  3. 及时清除不能硬件清除的中断请求标志。

# 5.1.4 中断请求的清除

- 在CPU响应中断后，能被硬件自动清除中断标志的有：
  - 边沿触发的外部中断标志IE0、IE1
  - 定时器中断标志TF0、TF1
- 不能硬件清零，在中断返回前，需要通过软件来清除的中断标志有：
  - 电平触发的外部中断标志IE0、IE1
  - 串行口中断标志TI/RI

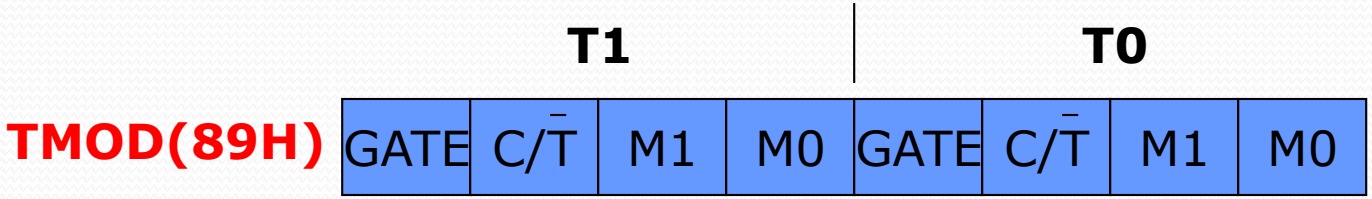
# 5.1.4 有关定时/计数器

- 定时器 / 计数器的两个功能：

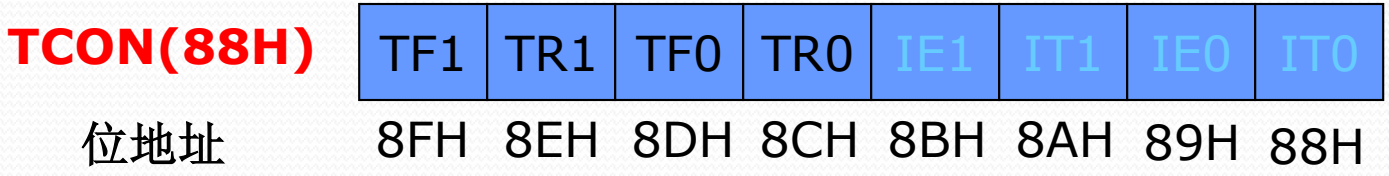
1. 精确地确定某一段时间间隔(作定时器用)；
   - 作定时器时，计数输入信号是内部时钟脉冲，每个机器周期产生一个脉冲使计数器增1。
2. 累计外部输入的脉冲个数(作计数器用)。
   - 作计数器时，计数脉冲来自相应的外部输入引脚T0或T1。当输入信号产生由1至 0的跳变时，计数器的值增1。

- 定时/计数器的4种工作方式：
  - 方式0：13位计数器，由THx的全部8位和TLx的低5位构成。
  - 方式1：16位计数器，由THx和TLx的全部8位构成。
  - 方式2：自动重装8位计数器，其中，TLx作为8位计数器，THx存放计数初值，计满后将THx的值自动装入TLx。
  - 方式3：定时器T0被拆成2个独立的8位计数器(TL0和TH0)来使用。

# 与定时/计数器有关的寄存器




☆不可位寻址



## 中断举例：

例：单片机的5个中断源全部使用，其中INT0和串行口中断为高级中断，其它3个为低级中断，INT0下降沿有效，INT1低电平有效，请对单片机进行中断初始化。

```
ORG 0000H
LJMP MAIN
ORG 0003H
LJMP INT0P
ORG 000BH
LJMP T0P
ORG 0013H
LJMP INT1P
```

```
        ORG 001BH
        LJMP T1P
        ORG 0023H
        LJMP SPPRO
        ORG 0030H
MAIN:MOV SP,#0A0H
        MOV IP,#11H
        SETB IT0
        CLR IT1
        MOV IE,#9FH
HERE:SJMP HERE
```

```
        ORG   0100H
INT0P: …
          RETI
        ORG   0200H
T0P: …
          RETI
        ORG   0300H
INT1P: …
          RETI
        ORG   0400H
T1P: …
          RETI
        ORG   0500H
SPPRO: …
        RETI
```

# 5.2 单片机的定时方法

1. 软件定时

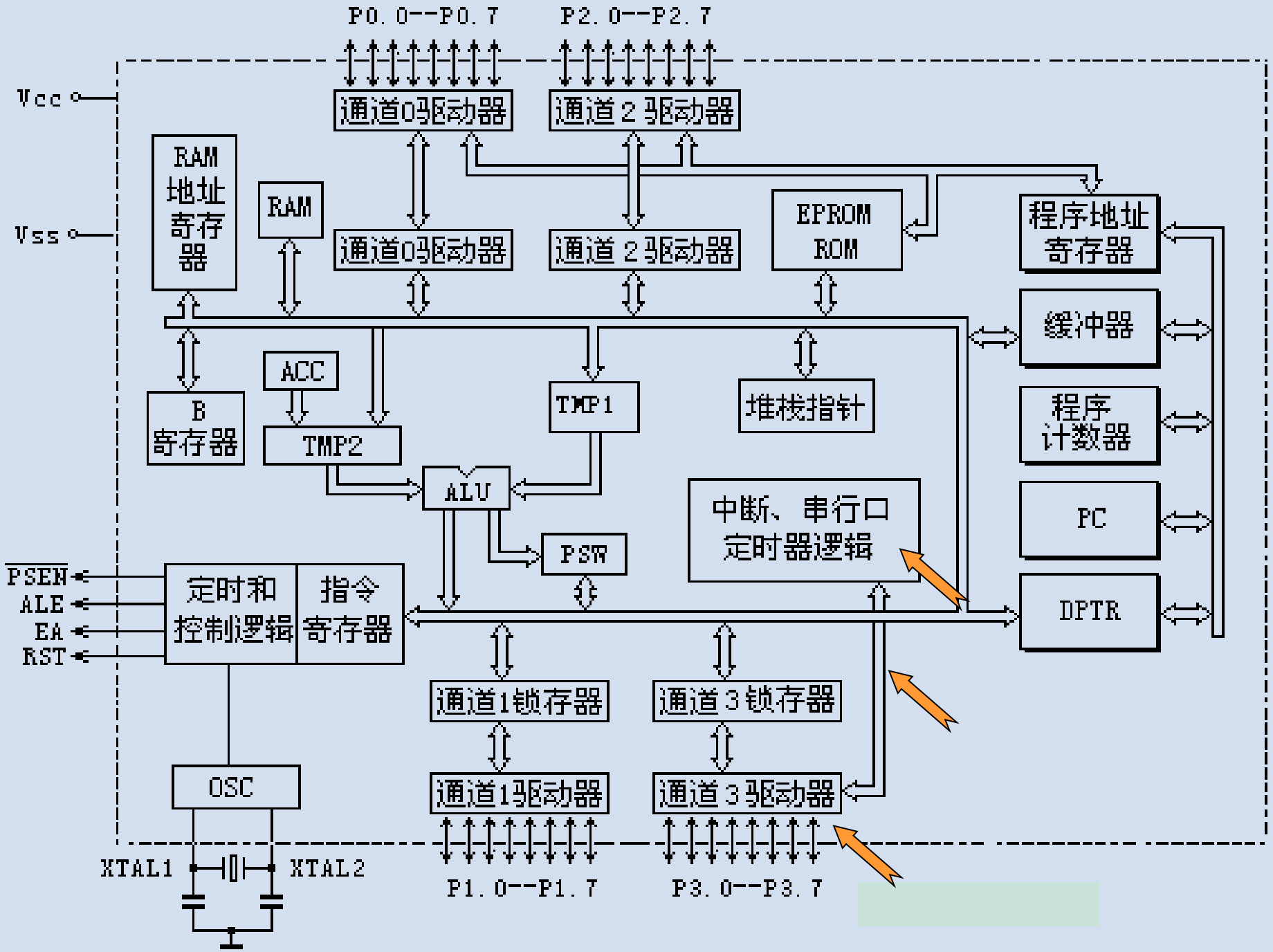
   通过执行一个循环程序来得到一定的时间延时；占用CPU的时间。

2. 硬件定时

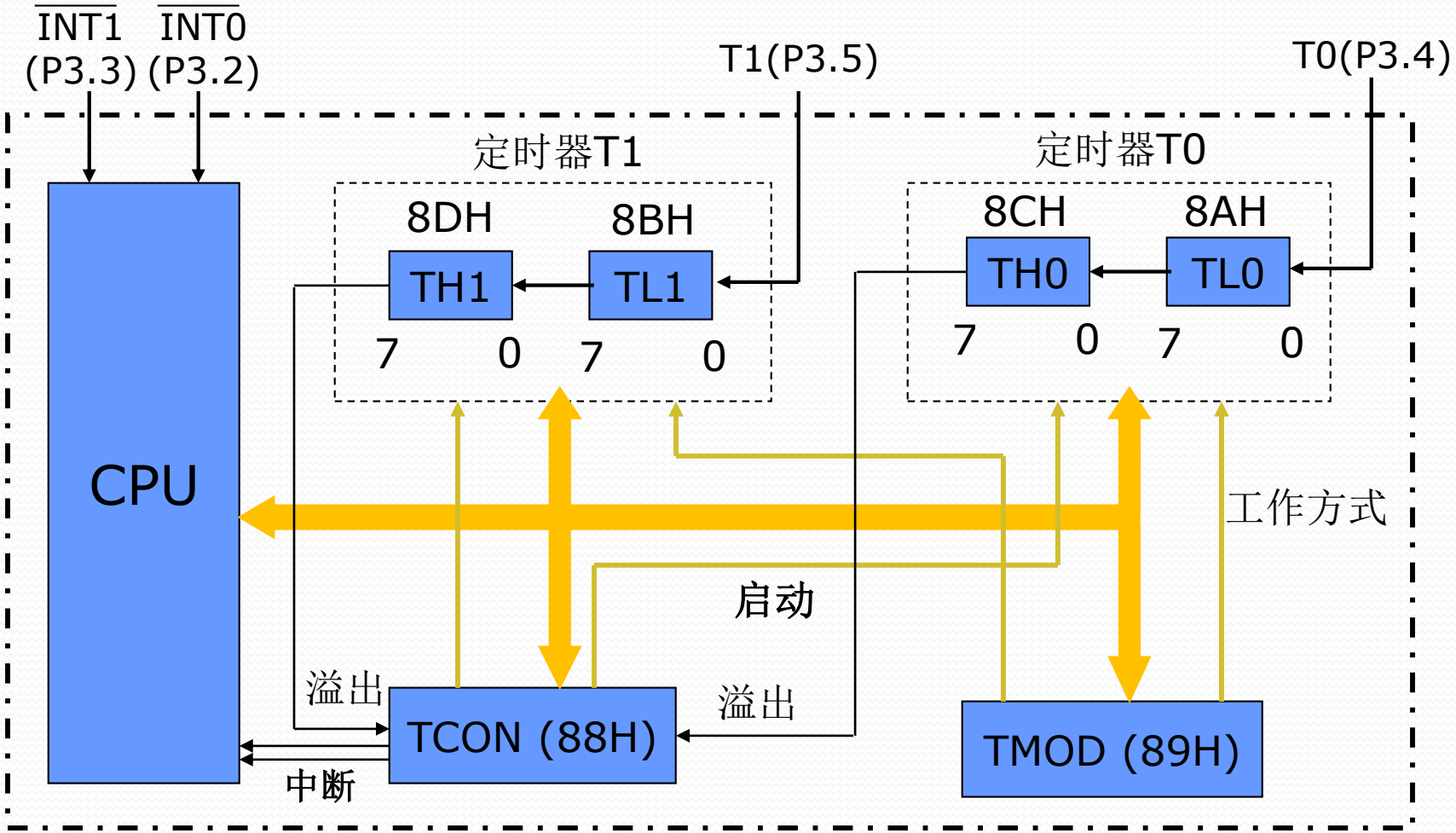
   定时功能完全由硬件电路来完成；不占用CPU的时间。

3. 可编程定时器定时

   通过对系统时钟脉冲的计数来实现定时；所以该定时器兼有定时和计数的功能。


P0.0--P0.7
P2.0--P2.7
Vcc
Vss
通道0驱动器
通道2驱动器
RAM
地址
寄存
器
RAM
通道0驱动器
通道2驱动器
EPROM
ROM
程序地址
寄存器
缓冲器
ACC
B
寄存器
TMP1
堆栈指针
程序
计数器
TMP2
ALU
中断、串行口
定时器逻辑
PC
PSW
PSEN
ALE
EA
RST
定时和
控制逻辑
指令
寄存器
DPTR
通道1锁存器
通道3锁存器
OSC
通道1驱动器
通道3驱动器
XTAL1
XTAL2
P1.0--P1.7
P3.0--P3.7

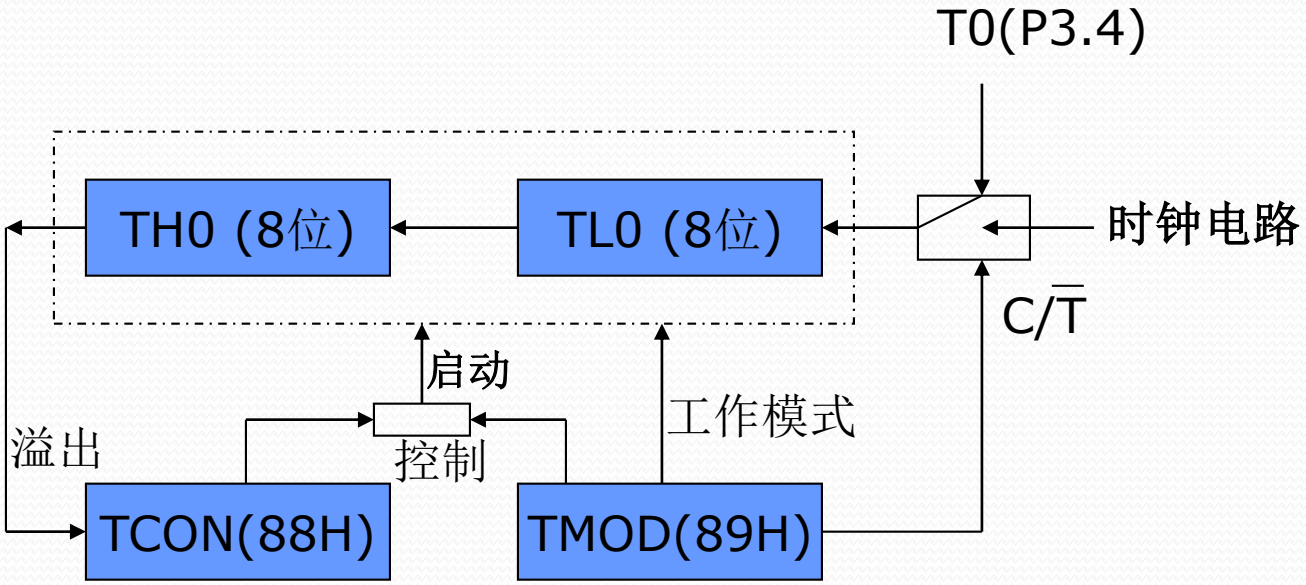
# 5.2.1 8051的定时/计数器结构图



# 5.2.1 定时/计数器的构成

- 定时/计数器的核心是二进制加1计数器(TH0、TL0和TH1、TL1)。
- TL0、TH0和TL1、TH1(定时/计数器)：
  - MCS-51单片机中有两个16位的定时/计数器T0和T1。
  - 它们由四个8位寄存器组成的，分别为TH0、TL0和TH1、TL1。我们可以单独对这四个寄存器进行寻址，但不能把T0和T1当作16位寄存器来使用。

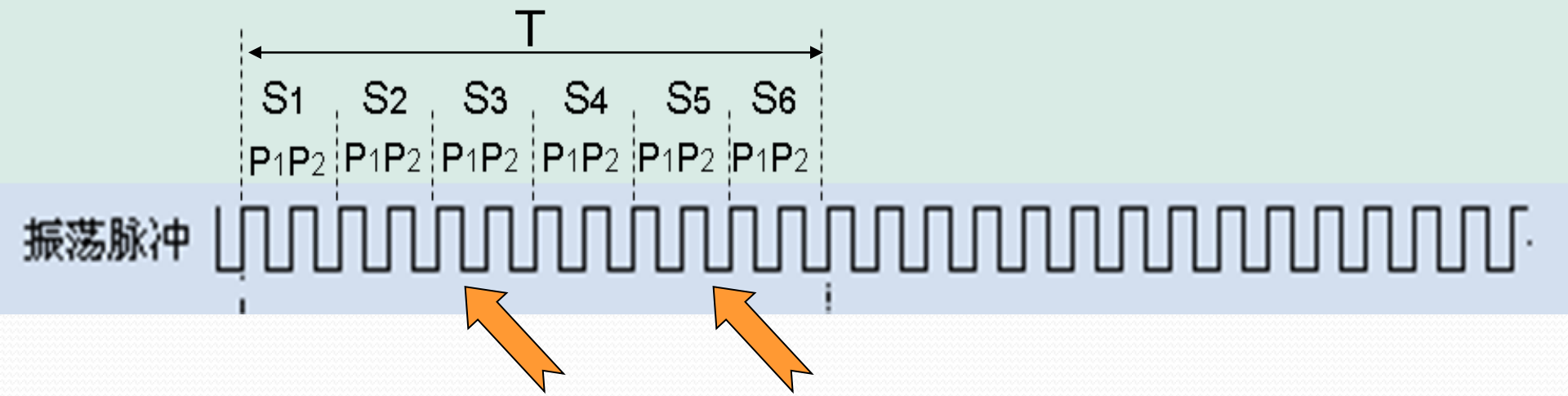
# 5.2.1 定时/计数器T0的结构



# 5.2.1 定时/计数器的功能

- **定时功能**——计数输入信号是内部时钟脉冲，每个机器周期使寄存器的值加1。所以，计数频率是振荡频率的1/12。
- **计数功能**——计数脉冲来自相应的外部输入引脚，即T0(P3.4)或T1(P3.5)；当T0或T1引脚的输入脉冲出现1到0的跳变时，计数器值加1。
- 定时/计数器T0和T1是工作在定时器方式还是计数器方式，通过方式选择寄存器TMOD中的控制位(C/T)来设置。

# 计数功能



# 5.2.2 与定时/计数器相关的寄存器

- 方式选择寄存器(TMOD)

只能字节寻址

| | T1 | | | | T0 | | | |
|---|---|---|---|---|---|---|---|---|
| TMOD(89H) | GATE | $C/\overline{T}$ | M1 | M0 | GATE | $C/\overline{T}$ | M1 | M0 |

| M1 | M0 | 方式 | 功能 |
|---|---|---|---|
| 0 | 0 | 0 | 13位定时/计数器 |
| 0 | 1 | 1 | 16位定时/计数器 |
| 1 | 0 | 2 | 8位自动重装定时/计数器 |
| 1 | 1 | 3 | 定时器0分为2个8位定时/计数器 |

| | T1 | | | | T0 | | | |
|---|---|---|---|---|---|---|---|---|
| TMOD(89H) | GATE | $C/\overline{T}$ | M1 | M0 | GATE | $C/\overline{T}$ | M1 | M0 |

- $C/\overline{T}$：
  - $C/\overline{T}$=0时，选择定时器功能
  - $C/\overline{T}$=1时，选择计数器功能
- GATE：
  - GATE=0时，定时器的启动通过软件将TR0或TR1置1即可。
  - GATE=1时，定时器的启动受外部中断引脚$\overline{INT0}$或$\overline{INT1}$的电平影响。
- TMOD寄存器不可位寻址，复位值为00H。

## 控制寄存器(TCON)

| TCON(88H) | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
|---|---|---|---|---|---|---|---|---|
| 位地址 | 8FH | 8EH | 8DH | 8CH | 8BH | 8AH | 89H | 88H |

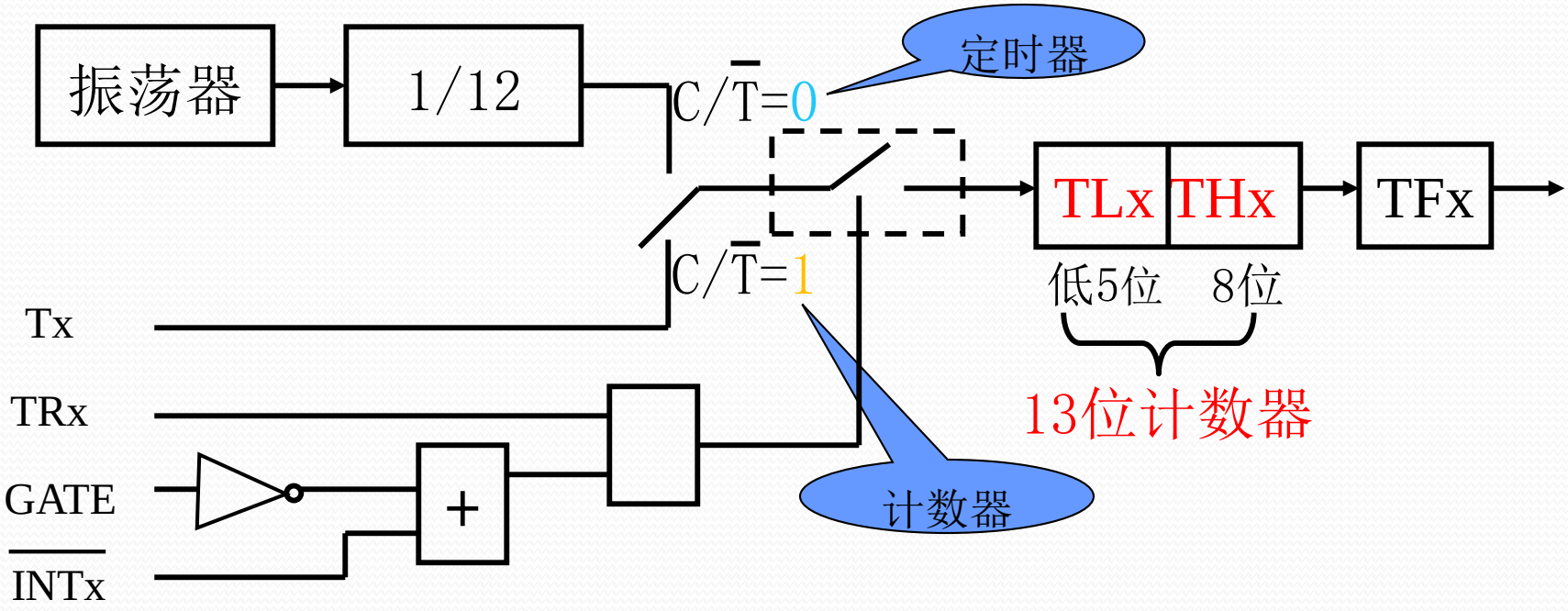
- TF1、TF0：定时器T1、T0的溢出标志位。
  - TF1、TF0=1时，定时/计数器的产生计数溢出。
- TR1、TR0 ：定时/计数器T1、T0启动控制位。
  - TR1、TR0置1启动工作。
  - TR1、TR0清零停止工作。
- IE1、IE0 ：外部中断(INT1、INT0)请求标志位。
  - IE1、IE0=1时，INT1、INT0端出现有效中断请求信号。
- IT1、IT0 ：外部中断1触发方式选择位。
  - IT1、IT0=1时，脉冲触发方式，下降沿有效。
  - IT1、IT0=0时，电平触发方式，低电平有效。

# 5.2.3 定时器/计数器的工作方式

## 一. 工作方式0

### 1. 方式0的电路逻辑结构

方式0是13位的定时/计数结构，其计数器由THx的全部8位和TLx的低5位构成。



# 工作方式0

- 当TLx的低5位计数溢出时，就向THx进位，只有当全部13位计数溢出时，才向计数溢出标志位TFx进位。

2. 方式0定时时间的计算

定时时间=$(2^{13}-$ Tx初值$) \times$ 机器周期

- 计数范围：1~ $2^{13}$(8192)
- 最小定时单位为一个机器周期(T)
- 最大计数值M= $2^{13}$

- 例题：已知单片机系统外接晶振为6MHz，请计算在方式0下的最小定时时间$T_{min}$和最大定时时间$T_{max}$；若使用定时器T0，要求每经过10ms溢出一次，定时器的计数器初值应为多少？

- 最小定时时间：

$$T_{min} = T = 12T_{OSC} = 12 \times 1/(6 \times 10^6) = 2\mu s$$

- 最大定时时间：

$$T_{max} = 2^{13} \times T = 8192 \times 2\mu s = 16.384ms$$

- 计数初值(X)的计算：

$$定时时间 = (2^{13} - X) \times T$$

定时时间为10ms，求得：X = 3192

- 赋初值：3192 = 0C78H = 0110001111000B

  所以，TH0=63H，TL0=18H(高3位填0)

## 二. 工作方式1

### 1. 方式1的电路逻辑结构

方式1为16位的定时/计数器，其计数器由THx和TLx的全部8位构成。

### 2. 方式1定时时间的计算

定时时间=$(2^{16}-$ Tx初值$)\times$ 机器周期

- 计数范围：1~ $2^{16}$ (65536)
- 最小定时单位为一个机器周期(T)
- 最大计数值M= $2^{16}$

## 三. 工作方式2

### 1. 方式2的电路逻辑结构

方式2是把8051单片机16位的计数器设置成可以自动重装初值的8位计数器。

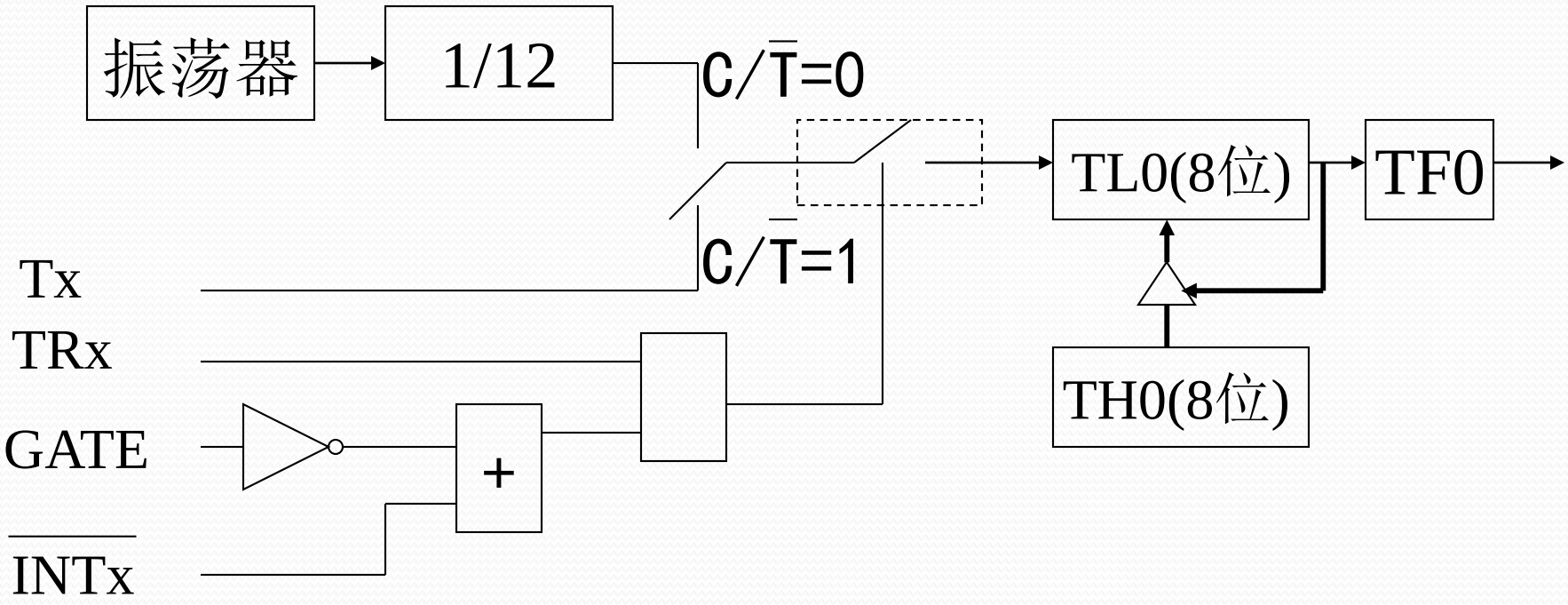
- 方式2和方式0、方式1不同点在于：当计数到满值溢出后，方式0、方式1需要通过软件重新赋计数初值；而方式2将TLx作为8位计数器，THx存放计数初值，计满后将THx的值自动装入TLx。

### 2. 方式2定时时间的计算：

定时时间=$(2^8 -$ Tx初值$) \times$ 机器周期

- 计数范围：1~ $2^8$ (M=256)
- 方式2尤其适合于串行口波特率发生器。

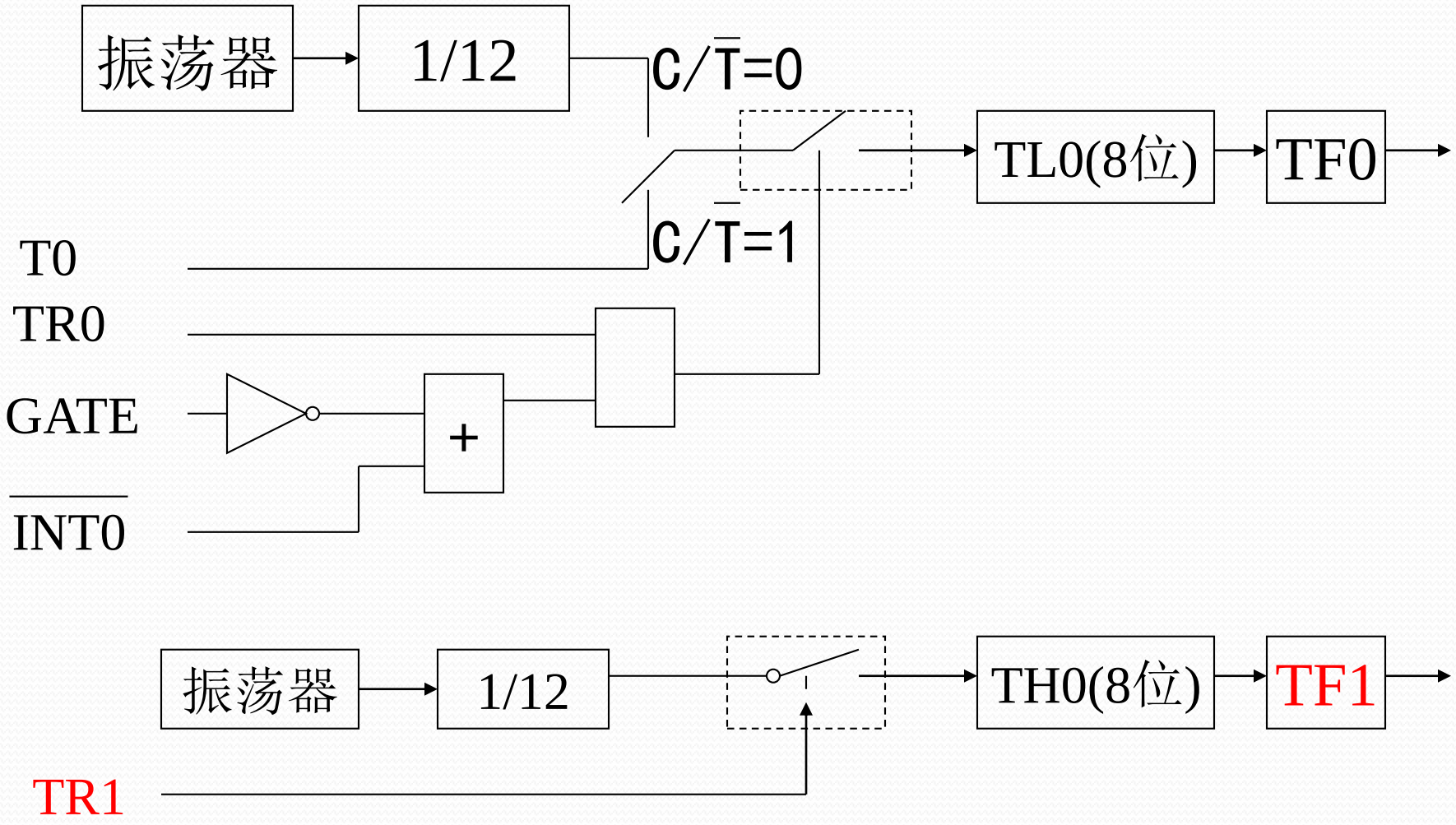
# 方式2逻辑结构图



# 四. 工作方式3

- 方式3只适用于定时器/计数器T0。
- 方式3下的定时器T0被拆成2个独立的8位计数器(TL0和TH0)来使用。
- TL0既可作为定时器使用，又可作为计数器使用，定时/计数器T0的各个控制位、引脚和中断源归它使用。所以其功能和操作类似于方式0、方式1。
- TH0就只能作为一个简单的定时器使用，它使用定时/计数器T1的控制位TF1、TR1和中断源。
- 方式3可以构成2个定时器或一个定时器一个计数器。

# 定时器T0方式3的逻辑结构图



# 5.2.4 定时/计数器的应用举例

- 可编程器件在使用前需要进行初始化：
  - 确定TMOD控制字：编程时将控制字送TMOD
  - 计算计数器的计数初值：

    计算计数初值送THx、 TLx
  - 开中断(如果使用中断方式)：置位EA、ETx
  - 启动定时器：置位TRx

- 初值计算：
  - 计数器初值X

    X = M − 计数值
    - 计数值：为外部脉冲信号下降沿的次数
  - 定时器初值X

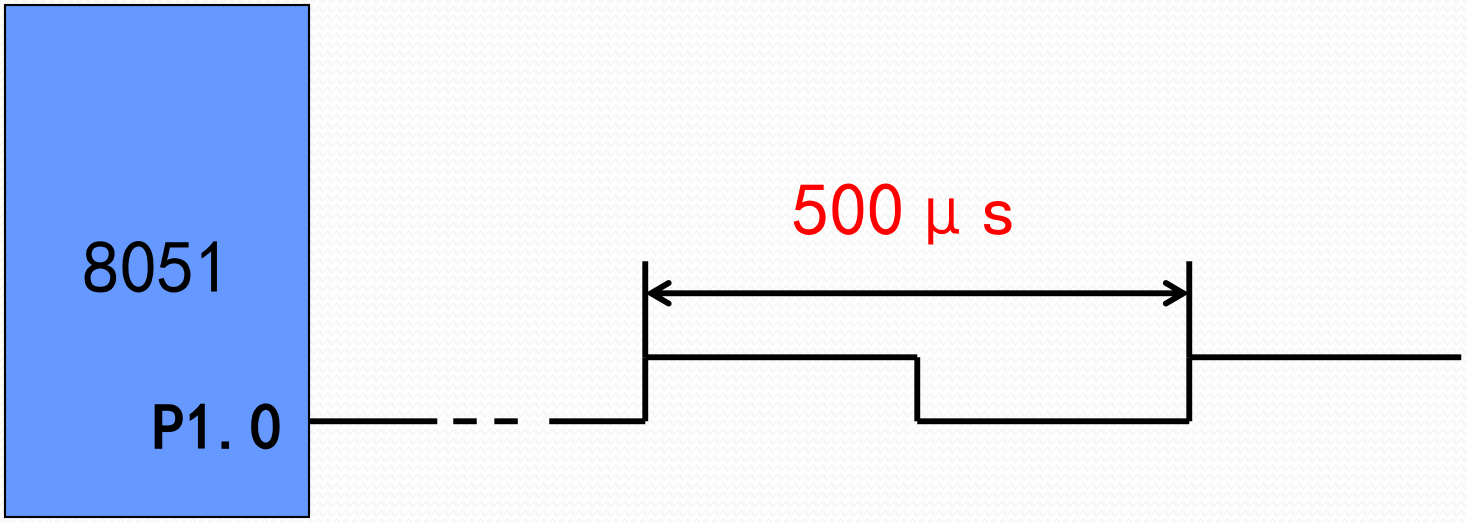
    定时时间=(M − X) × 机器周期

- 例1：设晶振频率fosc=6MHz，分别讨论各种工作方式下最长定时时间$T_{max}$。

  解：

1. 由fosc=6MHz算得：T=2us。
2. 由于是加1计数，所以最长定时应是计数初值最小时(为0时)的定时时间。
   - 方式0：$(2^{13}-0)\times 2us = 16.384ms$
   - 方式1：$(2^{16}-0)\times 2us = 131.072ms$
   - 方式2、3：$(2^{8}-0)\times 2us = 0.512ms$

- 例2：设晶振频率fosc=6MHz，使用定时器1以方式1产生周期为500μs的方波脉冲，并由P1.0 输出。



- 定时器T1工作于方式1：

| T1控制 | | | | T0控制 | | | |
|---|---|---|---|---|---|---|---|
| GATE | C/$\overline{T}$ | M1 | M0 | GATE | C/$\overline{T}$ | M1 | M0 |
| 0 | 0 | 0 | 1 | X | X | X | X |

控制字10H

- 计算计数器初值：

$(2^{16}–X)\times 2\mu s = 250\mu s$

即$2^{16}–X=125$

$X = 2^{16}–125 = 10000H – 7DH$

$= 0FF83H$

所以，初值为：TH1=0FFH，TL1=83H

- 主程序：

```
MAIN：  MOV TMOD，#10H
        MOV TH1，#0FFH
        MOV TL1，#83H
        SETB TR1
LOOP：JBC TF1，NEXT
        SJMP  LOOP
NEXT：  MOV TH1，#0FFH
        MOV TL1，#83H
        CPL  P1.0
        SJMP  LOOP
        END
```

# 5.3 应用举例

例1：方式0应用，时钟频率采用6MHZ，要求在P1.0口输出周期为2ms的方波

- 采用定时器T0，程序查询方式

1. 计算定时初值x：

   $(2^{13}- x) \times 2\mu s=1000\mu s$

   X=7692D=1E0CH

   TH0=1EH,TL0=0CH

2. 定时器初始化和主程序：

   设置定时器相关寄存器

3. 主程序查询定时器状态

```
        MOV  TMOD，#00H     ;设置定时器T0方式0
        SETB  TR0
LOOP:MOV  TH0,#1EH
        MOV  TL0,#0CH
LOOP1:JNB  TF0,LOOP1
        CLR TF0
        CPL P1.0
        SJMP  LOOP
```

- 采用定时器T0，中断方式实现。解：

1. 计算定时初值x

   $(2^{13}- x) \times 2\mu s=1000\mu s$

   X=7692D=1E0CH

   TH0=0F0H,TL0=0CH

2. 程序初始化

   包括定时器初始化和中断系统初始化

3. 主程序和中断服务程序

   - 主程序，可以是完成其它工作的程序
   - 产生方波，并注意定时器初值重装。

- 程序设计：

```
        ORG   0000H
RESET:  AJMP  MAIN
        ORG  000BH          ；定时中断T0入口
        AJMP IT0P
        ORG  1000H
MAIN:   MOV  SP，#40H
        LCALL  INITP
HERE:   AJMP  HERE
INITP:  MOV TL0，#0CH
        MOV  TH0，#0F0H
        SETB  TR0           ；启动T0
        SETB  ET0           ；开T0中断
        SETB  EA            ；开CPU中断
        RET
```

```
IT0P：    MOV  TL0，#0CH        ;重装T0初值
          MOV  TH0，#0F0H
          CPL  P1.0
          RETI
```

例2：设晶振频率为6MHz，使用定时器T1在P1.1口输出周期为500us，占空比为1:5的方波。

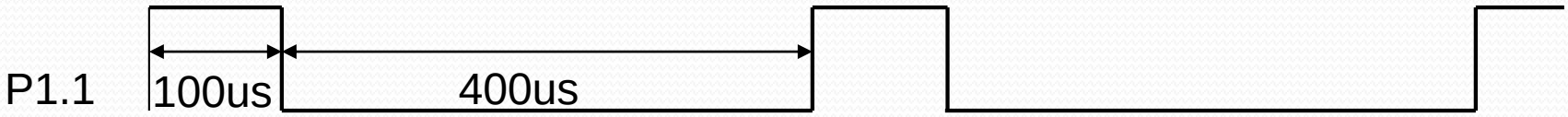
分析：占空比1:5，所以得到一个周期正脉冲为100us，负脉冲为400us，。

- 采用方式2，定时100us让P1.1取反，接下来，4个100us再让P1.1取反。

1. 计算定时初值x

$(2^8 - x) \times 2us = 100us$

$x = 0CEH$

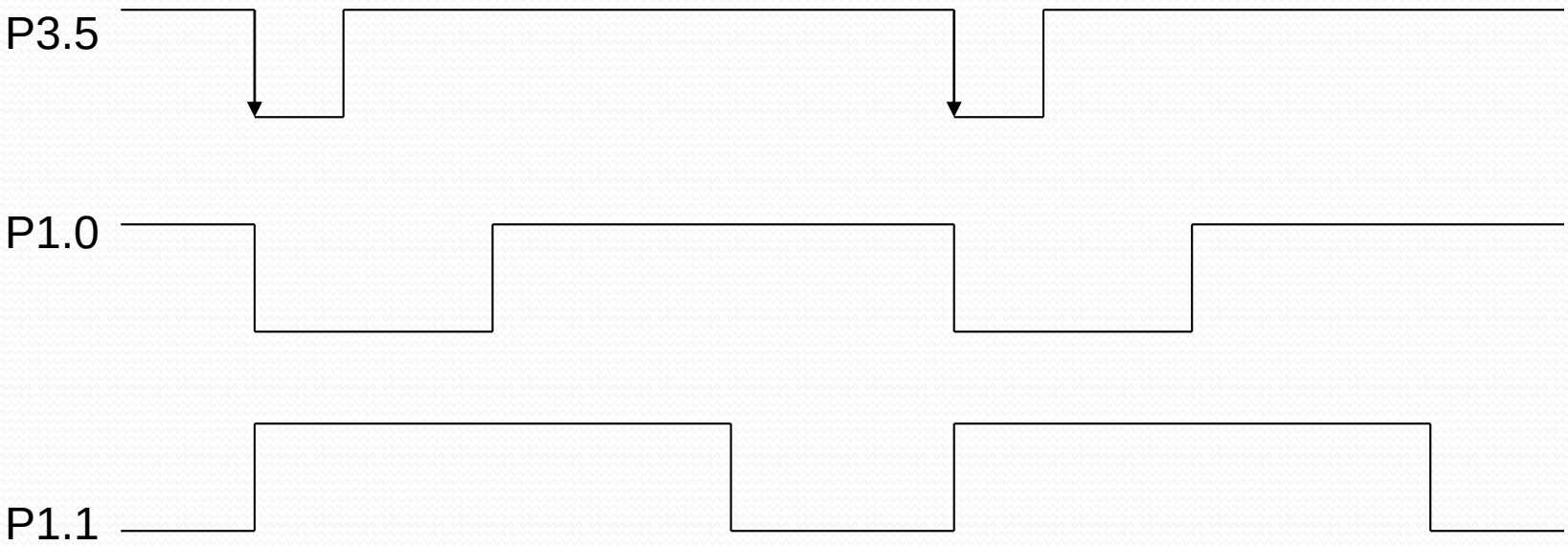


# 程序设计：

```
START:MOV  TMOD，#20H
      MOV  TH1，#0CEH
      MOV  TL1，#0CEH
      SETB  P1.1
      SETB  TR1
LOOP1:JBC TF1，ST1
      MOV  R2，#04H
      SJMP  LOOP1
ST1:  CLR  P1.1
      CLR  TF1
LOOP2:JBC  TF1，ST0
      SJMP  LOOP2
ST0:  DJNZ  R2，LOOP2
      SETB  P1.1
      CLR  TF1
      AJMP  LOOP1
```

## 定时/计数器的计数功能：

例3：设晶振频率为6MHz， T1(P3.5)作为外部低频脉冲输入信号线，假设输入脉冲周期大于1ms。要求P3.5每发生一次负跳变，P1.0输出一个500us的同步负脉冲，并且P1.1输出一个1ms的同步正脉冲。



1. 采用计数器功能，工作方式2

分析：外部脉冲采用计数方式， 初值设为0FFH，每来一次负跳变，计数溢出。定时器定时500us，采用方式2，得到初值为：06H

```
START：MOV  TMOD，#60H      ；定义工作方式
       MOV  TH1，#0FFH      ；设置计数初值
       MOV  TL1，#0FFH
       SETB  P1.0
       CLR   P1.1
       SETB  TR1            ；开计数器
JS：   JBC  TF1，DS         ；下降沿到否
       SJMP  JS
```

```
DS: CLR  TR1                    ;  关计数器
    MOV  TMOD，#20H             ;  设置T1定时方式2
    MOV  TH1，#06H              ;  定时500us初值
    MOV  TL1，#06H
    CLR   P1.0                  ;  设置初始电平
    SETB  P1.1
    SETB  TR1                   ;  开定时器
LP:JBC  TF1,NEX1                ;  判断500us到否
    SJMP  LP
NEX1:SETB  P1.0                 ;  P1.0置高
    CLR  TF1                    ;  软件清标志位
LP1:JBC  TF1,NEX2               ;  判断1ms到否
    SJMP  LP1
NEX2:CLR  P1.1                  ;  P1.1置低
    CLR  TR1                    ;  关定时器
    AJMP  START
```

## 2. 中断方式(外部中断源的扩展)

分析：P3.5为定时中断T1的输入，采用定时中断T1。

从P1.0端输出500us的同步负脉冲，从P1.1端输出1ms的同步正脉冲，采用定时器T0方式2，程序查询方式。

```
        ORG  0000H              ；复位主程序
RESET:AJMP  MAIN
        ORG  000BH              ；定时中断T0入口
        AJMP  IT0P
        ORG  001BH              ；
        AJMP IT1P
        ORG  1000H
MAIN: MOV  SP,#40H
        ACALL  INIT             ；初始化子程序
HERE: AJMP  HERE
```

```
INIT:   MOV  TMOD,#62H        ;T0定时方式2,T1计数方式2
        MOV  TH0，#06H        ;定时500us初值
        MOV  TL0，#06H
        MOV  TH1，#0FFH       ;计数初值
        MOV  TL1，#0FFH
        SETB  P1.0
        CLR   P1.1
        SETB  TR1             ;开T1
        SETB EA
        SETB  ET1             ;开T1中断允许
        RET
```

```
IT1P: CLR   P1.0              ;设置初始电平
      SETB  P1.1
      MOV   R2,#02H
      SETB  TR0               ;开定时器T0
      SETB  ET0               ;开T0中断允许
      RETI
IT0P:SETB  P1.0
      CLR   TF0
      DJNZ  R2,HERE
      CLR   P1.1
      CLR   TR0
      CLR   ET0
      RETI
```

```
IT0P: CLR   ET0               ;禁止T0中断
      SETB  P1.0
      CLR   TF0
LOOP: JBC   TF0,NEX1
      SJMP  LOOP
NEX1: CLR   P1.1
      CLR   TR0               ;关定时器T0
      RETI
```

例4：设晶振频率为6MHz，使用定时器/计数器T0工作方式3，分别产生200微秒和400微秒的定时中断，使P1.0和P1.1产生400us和800us的方波(晶振频率为6MHz)。

方式3：

- TL0和TH0作为两个独立的8位定时器/计数器使用。
- 可以构成2个定时器或一个定时器一个计数器。

1. 计算定时初值：

$(2^8–x1)\times 2us = 200us$ ； $(2^8–x2)\times 2us = 400us$

x1=9CH ； x2=38H

程序设计：

```
        ORG  0000H      ;复位入口
        AJMP  MAIN
        ORG  000BH      ;定时中断0入口(TL0)
        AJMP  IT0P
        ORG  001BH      ;定时中断1入口(TH0)
        AJMP  IT1P
        ORG  0100H
MAIN:   MOV  SP,#40H
        ACALL  INIT
HERE:   SJMP  HERE
INIT:   MOV  TMOD,#03H
        MOV  TL0,#9CH
        MOV  TH0,#38H
        SETB  TR0
        SETB  ET0
        SETB  TR1
        SETB  ET1
        SETB  EA
        RET
```

```
IT0P:MOV  TL0,#9CH
      CPL  P1.0
      RETI

IT1P:MOV  TH0,#38H
      CPL  P1.1
      RETI
```

例5：设晶振频率为6MHz，要求每隔100ms从外部ROM以data开始的数据区传送一个字节数据到P1口输出，共传送100个字节数据。

分析：1、100ms定时采用定时器T0方式2，中断方式

2、中断服务实现数据输出。

1. 100ms超出8位定时器的最大定时长度，所以采用软件计数+定时
2. 定时设为0.5ms，初值计算

$(2^8-x)\times 2us = 500us$

x=06H

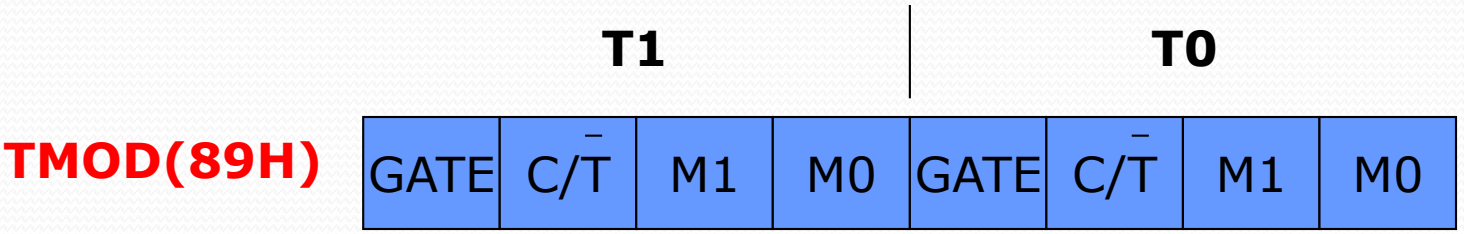
100ms软件计数200次

# 程序设计：

```
        ORG  0000H
        AJMP START
        ORG  000BH
        AJMP  T0INT
START: MOV  TMOD,#02H     ;T0方式2
        MOV  TL0，#06H
        MOV  TH0，#06H
        SETB ET0
        SETB PT0
        SETB  EA
        SETB  TR0
        MOV  DPTR，#data
        MOV  R6，#100
LOOP：  MOV  R7，#200
        MOV  A，R7
HERE：  JNZ  HERE
        DJNZ  R6，LOOP
        CLR  TR0
```

```
T0INT：DJNZ  R7，LP1
        CLR  A
        MOVC  A，@A+DPTR
        MOV  P1,A
        INC  DPTR
LP1:    MOV  A,R7
        RETI
```

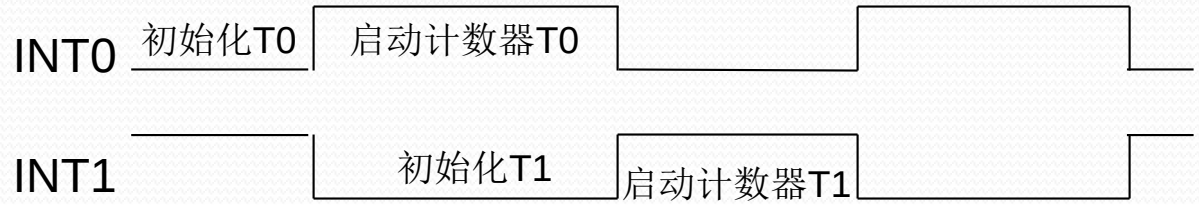
# 门控制位(GATE)的应用




- GATE用于设置定时器的启动是否受外部中断引脚$\overline{\text{INT0}}$或$\overline{\text{INT1}}$的电平控制。
- 所以，可以通过GATE控制位测量$\overline{\text{INTx}}$引脚上的外部脉冲的宽度。

## 例6：

```
START：MOV  TMOD，#99H
       MOV  TL0，#00H
       MOV  TH0，#00H
       MOV  TL1，#00H
       MOV  TH1，#00H
WT1:   JB  P3.2，WT1
       SETB  TR0        ;INT0=0,初始化T0
WT2:   JNB  P3.2,WT2
       SETB  TR1        ;INT0=1,即INT1=0，初始化T1；  T0计数开始
WT3:   JB  P3.2,WT3
       CLR  TR0         ; INT1=1, T1计数开始
WT4:   JB  P3.3,WT4
       CLR  TR1
       MOV  R1,#40H  ;取出计数值
       MOV  @R1,TL0
       INC  R1
       ……
```



# 上机实验

1. 输出方波
2. 流水灯：分别用2种定时方式使P1口输出信号，轮流点亮8个LED，每个LED点亮时间为50ms。
   - 方式1：利用调用延时子程序方式；
   - 方式2：利用定时器定时50ms，设晶振频率为12MHz。

- 流水灯：要求LED点亮时间为2s

# 作业题讲解

- 4-14 编写多个字节十进制转换成二进制
  - 单字节转换：数据在累加器A中

```
BCDH：  MOV  B，#10H
        DIV  AB
        MOV  R7，B
        MOV  B，#10
        MUL  AB
        ADD  A，R2
        RET
```

- 多字节：假设2个字节
  - 入口参数：原数据在R2R3中
  - 出口参数：二进制数据仍存在R2R3中

```
MOV  A，R3
LCALL  BCDH
MOV  R3，A
MOV  A，R2
LCALL  BCDH
MOV  B，100
MUL  AB
ADD  A，R3
MOV  R3，A
CLR  A
ADDC  A，B
MOV  R2，A
RET
```

- BCD码转换成16进制

```
MOV A, #69D
```

- BCDH: MOV B,#10H

```
DIV  A,B
MOV R2,B
MOV  B,#10H
MUL  A,B
ADD A,R2
RET
```

## 4-14 编写多个字节十进制转换成ASCII码

- 单字节则转换为双字节ASCII码：
- 入口参数：原数据在累加器A中
- 入口参数：高四位的ASCII码在A中，低四位的ASCII码在B中

```
HXASC:   MOV B,A
         LCALL HXASC1
         XCH A,B
         SWAP A ；高四位
HXASC1:  ANL A,#0FH
         ADD A,#90H
         DA A
         ADDC A,#40H
         DA A
         RET
```

- 多字节(n)BCD码转换成ACSII码：
  - 入口参数：原数据初始地址在R0中
  - 入口参数：转换后的初始地址在R1中

```
      MOV  R2,#n
LOOP:MOV   A,@R0
      LCALL  HXASC
      MOV @R1,B
      INC  R1
      MOV  @R1,A
      INC  R0
      INC  R1
      DJNZ  R2,LOOP
      SJMP  $
```

# 上机实验

- 按键中断编程：每次只有一个LED亮，首先让第一个点亮，采用外部中断0，每按一次键，所有的LED灯闪烁3次，然后第二个LED等点亮，顺序往下。