



# 微机原理和接口技术

## 第十一讲 定时器/计数器1

---



# 提 纲

**1. 定时器计数器概述**

**2. 定时器计数器的结构与控制**

**3. 定时器计数器的工作方式**

**4. 定时器计数器的初始化**

**5. 计数器的飞读**

**6. 定时的实现方法**

**7. 定时方式的应用**

**8. 计数方式的应用**

**9. 脉冲宽度的测量**

**10. 扩展外部中断**

**11. 实时时钟的设计**

# 提 纲

## 1. 定时器计数器概述



# 定时器计数器概述

定时器/计数器的核心是计数器（Counter）。计数器是能够对输入脉冲信号的跳变沿进行检测并能进行加法或减法计数的电路模块。

## 1. 加法计数器

**功能：**对输入脉冲进行加法计数，在每个脉冲的上升沿或下降沿计数器加1。

当计数器累加到全1时，再输入1个脉冲，计数器内容变为全0并产生溢出。

通常微控制器中的定时器/计数器模块，大多是采用加法计数器。

## 2. 减法计数器

**功能：**对输入脉冲进行减法计数，在脉冲的上升沿或下降沿计数器减1。

当计数器减到全0时，再输入1个脉冲，计数器内容变为全1并产生溢出。

独立的定时器/计数器芯片，大多采用减法计数方式，如Z80CTC、8253、8254等。



# 定时器计数器概述

MCU中的定时器/计数器模块是可编程的，通过编程可以使其工作在定时模式（用作定时器）或计数模式（用作计数器）。

## 1. 定时功能

- **软件定时：**通过执行一段程序（如延时程序）实现定时。其特点是无需额外的硬件，但需要消耗CPU的时间资源。
- **硬件定时：**用定时器/计数器进行定时。若计数器的计数脉冲是已知频率的脉冲信号，则通过对定时器/计数器的编程，可以实现准确的定时，其特点是无需消耗CPU的时间资源。

## 2. 计数功能

利用定时器/计数器对外部脉冲计数和统计。如记录一定时间内的外部脉冲的个数，实现该脉冲频率的测量；又如统计生产流水线上工件的数量等。



# 定时器计数器概述

在实际的微机测控系统中，通常需要定时采集输入信号或定时输出控制信号，也需要对外部脉冲信号和外界事件进行计数。

## 1. 定时器的应用

实现硬件的准确定时；不占用CPU时间，应用广泛。

- 如10ms进行一次小车轨迹测量，定时输出舵机控制信号，定时进行通信或显示等等。
- 可产生时间基准，如20ms、50ms、1s等，利用1s时基可设计实时时钟。

## 2. 计数器的应用

实现对外部脉冲或事件的计数。

- 利用定时器/计数器，可以实现外部脉冲频率、周期的测量；
- 通过对外部事件的记录，使系统做出处理和控制的决策。如外部事件数达到一定量时，发出相应的控制信号等。

# 提 纲

## 2. 定时器计数器的结构与控制



## 定时器计数器的结构与控制

8051微控制器有2个可编程的16位定时器/计数器T0和T1，采用的是加法计数器。

所谓可编程是指可以通过程序，对其进行相关内容的设置：

- 选择定时或计数模式、以及具体工作方式；
- 预置定时或计数的初值；
- 设置定时器/计数器溢出时，是否允许中断；
- 控制T0、T1启动或停止工作。



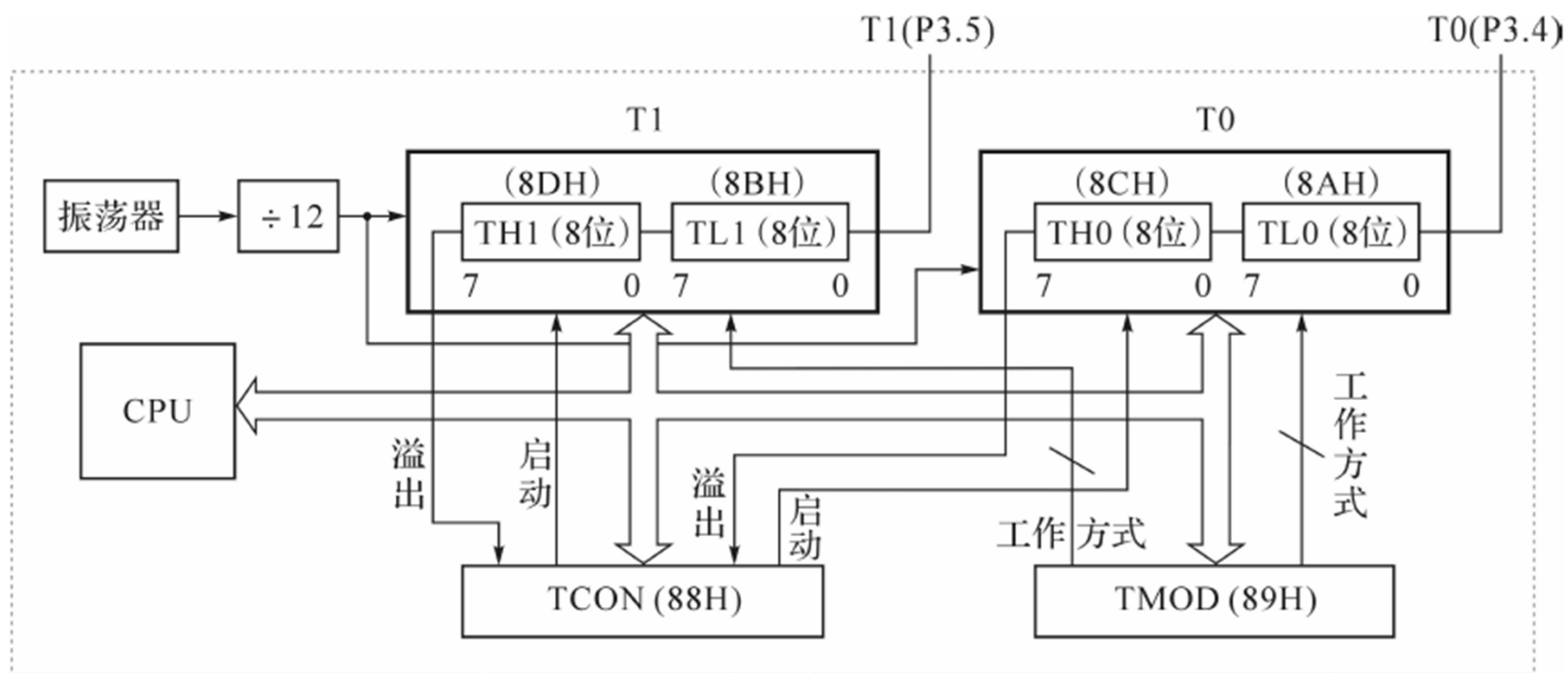
## 定时器计数器的结构与控制

组成结构：6个特殊功能寄存器、内部总线、2个引脚。

2个级联的8位加法计数器组成的T0，对应2个8位寄存器TH0、TL0；

2个级联的8位加法计数器组成的T1，对应2个8位寄存器TH1、TL1；


方式寄存器TMOD和控制寄存器TCON：对T0、T1进行设置和控制等。




# 定时器计数器的结构与控制

1. 方式寄存器TMOD：字节地址为89H，不可位寻址。

	7	6	5	4	3	2	1	0
位符号	GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
英文注释	Gate	Counter/ Timer	Mode bit 1	Mode bit 0	Gate	Counter/ Timer	Mode bit 1	Mode bit 0



定时/计数器 T1



定时/计数器 T0

- **$C/\overline{T}$** ：功能选择位。当 $C/\overline{T}=0$ 时，选择定时模式；当 $C/\overline{T}=1$ 时，选择计数模式。
- **GATE**：门控位。与 $INTi$ 信号相结合，可以实现外部脉冲高电平宽度的测量。将在“脉冲宽度测量”中进行介绍。
- **M1、M0**：工作方式（0~3）选择位。将结合工作方式予以介绍。

## 定时器计数器的结构与控制

2. 控制寄存器TCON：字节地址为88H，可位寻址。

	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
英文注释	Timer 1 Overflow	Timer 1 Run	Timer 0 Overflow	Timer 0 Run	Interrupt External 1 Flag	Interrupt 1 Type Control bit	Interrupt External 0 Flag	Interrupt 0 Type Control bit

➤ **TF0、TF1**：T0、T1溢出标志位也称为中断标志位。

T0、T1溢出时，相应位置“1”。在中断方式时，MCU响应中断后由硬件自动清“0”。在查询方式时，需要由程序清“0”。

➤ **TR0、TR1**：T0、T1的运行(启停)控制位。

GATE=0时：TR0/TR1=1，启动T0/T1计数；TR0/TR1=0，停止T0/T1计数。

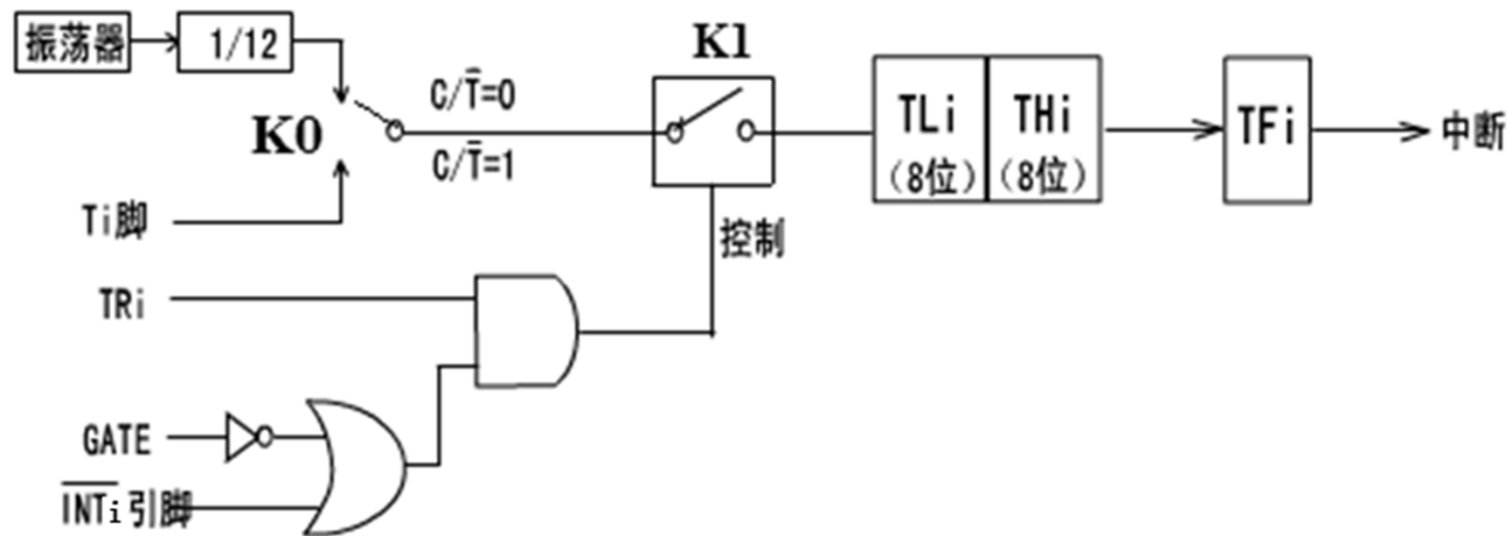
GATE=1时，由TR1和INT1引脚的电平同时控制T1的启停。

系统复位时，寄存器TMOD和TCON的内容为0。

# 定时器计数器的结构与控制

## 1. 定时模式

T0、T1的内部具体电路如图。当 $C/\overline{T}=0$ 时，选择定时模式。此时 $T_i$  ( $i=0$ 或 $1$ )的输入脉冲是内部振荡频率的12分频（即机器周期），即每个机器周期计数器加1。

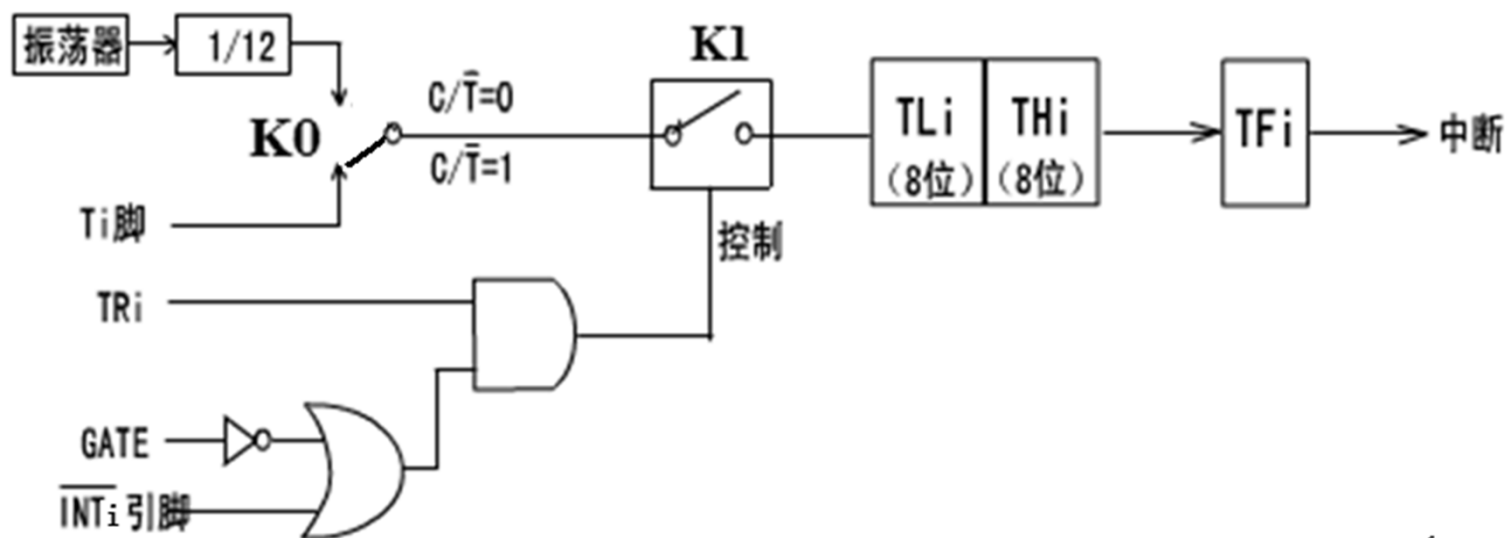


定时模式时，首先设置T0、T1寄存器的计数初值，再启动定时器工作（使K1闭合），当计数器不断加1到溢出时，定时器中断标志TFi ( $i=0$ 或 $1$ ) 置位，表示定时器溢出（定时时间到）。

# 定时器计数器的结构与控制

## 2. 计数模式

当 $C/\overline{T}=1$ 时，选择计数模式。计数器累计连接到T0(P3.4)、T1(P3.5)引脚上的外部脉冲，每输入一个脉冲计数器加1。



对于计数模式，通常设置T0、T1寄存器的计数初值为0，再启动定时器工作（使K1闭合），计数器累计外部脉冲，可以实现外部脉冲频率的测量。

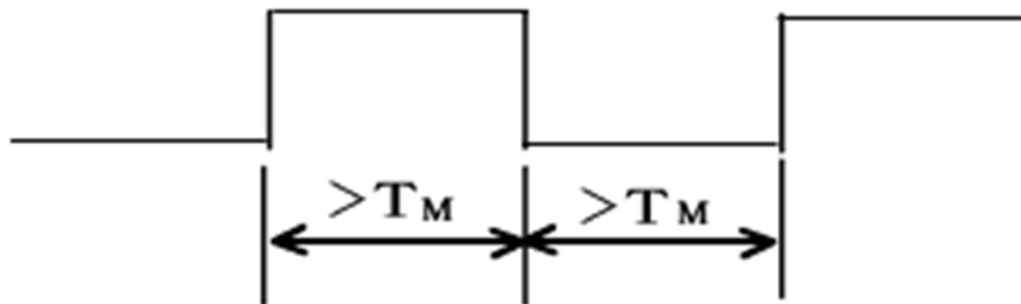
# 定时器计数器的结构与控制

## 2. 计数模式

对外部脉冲的要求（能够测量任何频率的脉冲吗？）

外部脉冲的采样过程：计数模式时，计数器每个机器周期采样一次T0（或T1）引脚状态，当检测到一个下降沿时计数器加1。（下降沿：前一个机器周期的采样值为高电平，当前机器周期采样值为低电平。）

能够测量的最高频率？脉冲的采样需要高电平和低电平宽度均应大于一个机器周期 $T_M$ ，因此计数器能够记录的外部脉冲的最高频率是 $1/(2 T_M)$  = 系统晶振频率的1/24，并要求外部脉冲的电平应与微控制器的电平相匹配。




# 提 纲

## 3. 定时器计数器的工作方式


# 定时器计数器的工作方式

工作方式的选择：有4种工作方式，用TMOD的M1、M0进行选择。

	7	6	5	4	3	2	1	0
位符号	GATE	C/ $\overline{T}$	M1	M0	GATE	C/ $\overline{T}$	M1	M0
英文注释	Gate	Counter/ Timer	Mode bit 1	Mode bit 0	Gate	Counter/ Timer	Mode bit 1	Mode bit 0



定时/计数器 T1



定时/计数器 T0

M1 M0	工作方式	功 能
0 0	方式0	13位计数器
0 1	方式1	16位计数器
1 0	方式2	计数初值自动重载8位计数器
1 1	方式3	定时器0分成两个8位计数器， 定时器1停止计数。

对于普通的定时和计数方式，要令**GATE=0**；

用于测量外部脉冲高电平宽度时，要令**GATE=1**。

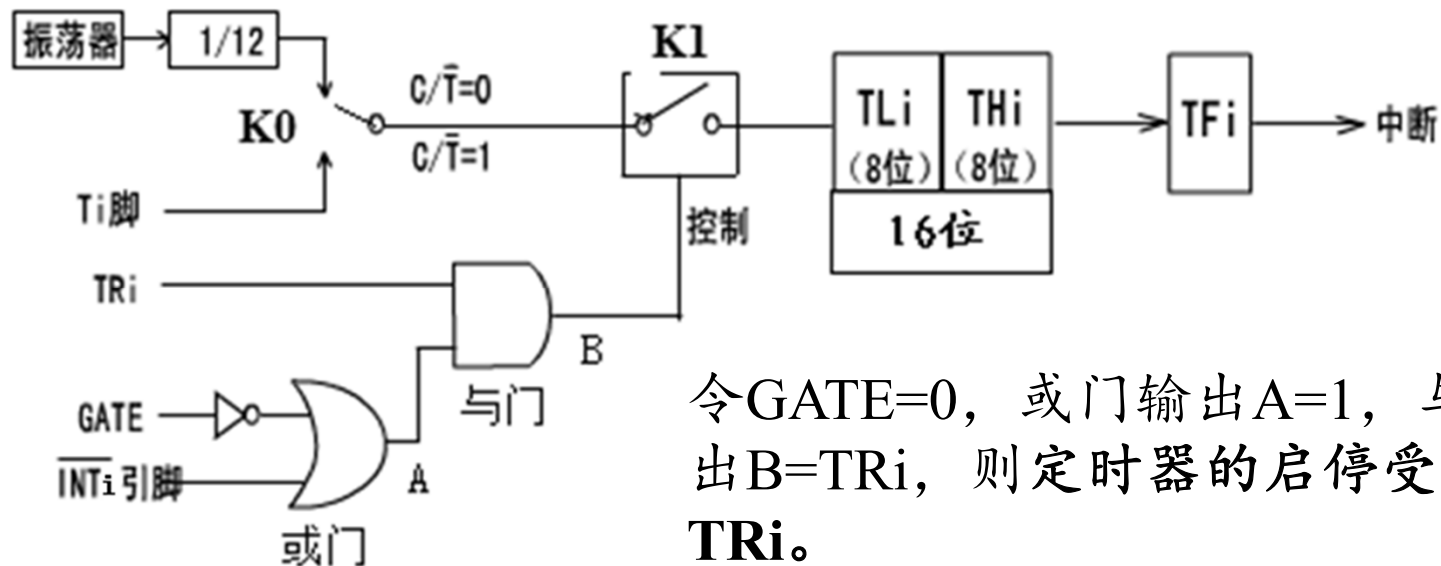
由于方式0和方式3很少应用，因此主要介绍方式1和方式2。



# 定时器计数器的工作方式

## 1. 方式1

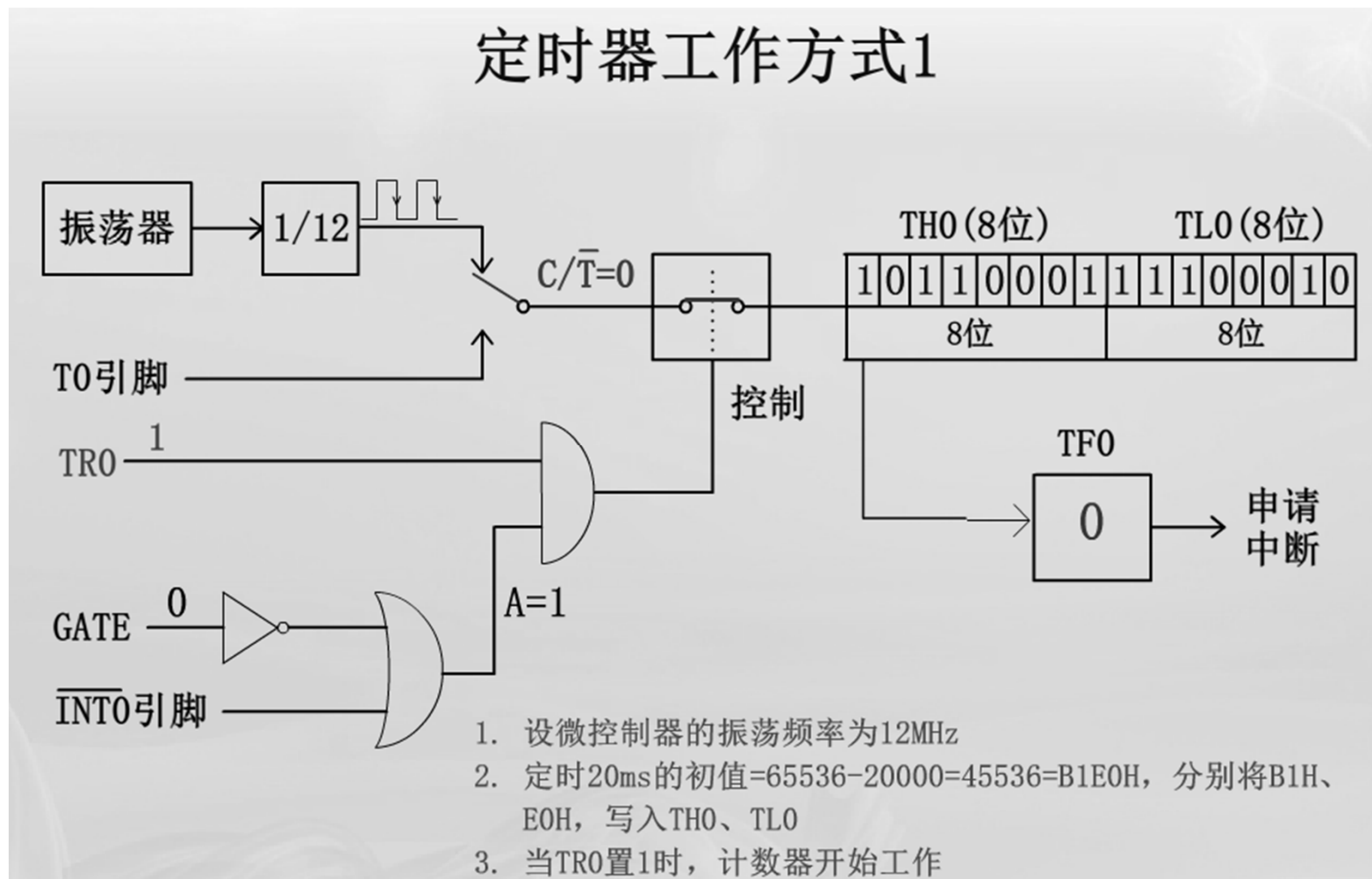
当相应的M1M0设置为0 1时，Ti设定为工作方式1。THi和TLi构成16位的加1计数器，因此称为**16位计数方式**，其最大计数值为 $2^{16}=65536$ 。



TRi=1，K1闭合，开始定时或计数；  
TRi=0，K1断开，结束定时或计数。

# 定时器计数器的工作方式

## 1. 方式1





# 定时器计数器的工作方式

## 1. 方式1

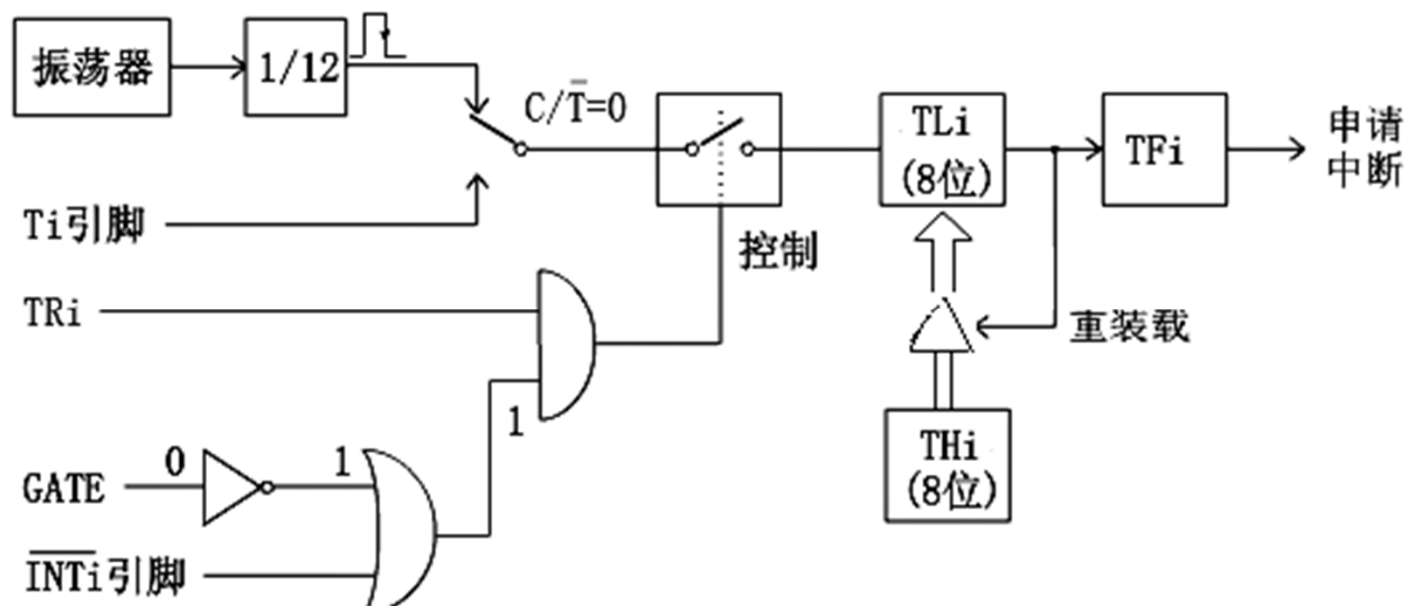
- 当16位计数器从设置初值不断加1到溢出（变为全0）时，表示定时时间到或计数脉冲达到设定个数，溢出标志位TFi变为“1”。如果Ti中断允许，则会向CPU发出中断请求。
- 为使Ti重新从设置初值开始计数，在Ti的中断服务程序或溢出后的处理程序中，必须首先要给THi、TLi重装载初值。
- 由于中断响应需要一定时间并存在中断响应时间的随机性，因此定时时间有一定误差（误差大小与系统程序的设计、中断的优先级别等有关）。

# 定时器计数器的的工作方式

## 2. 方式2

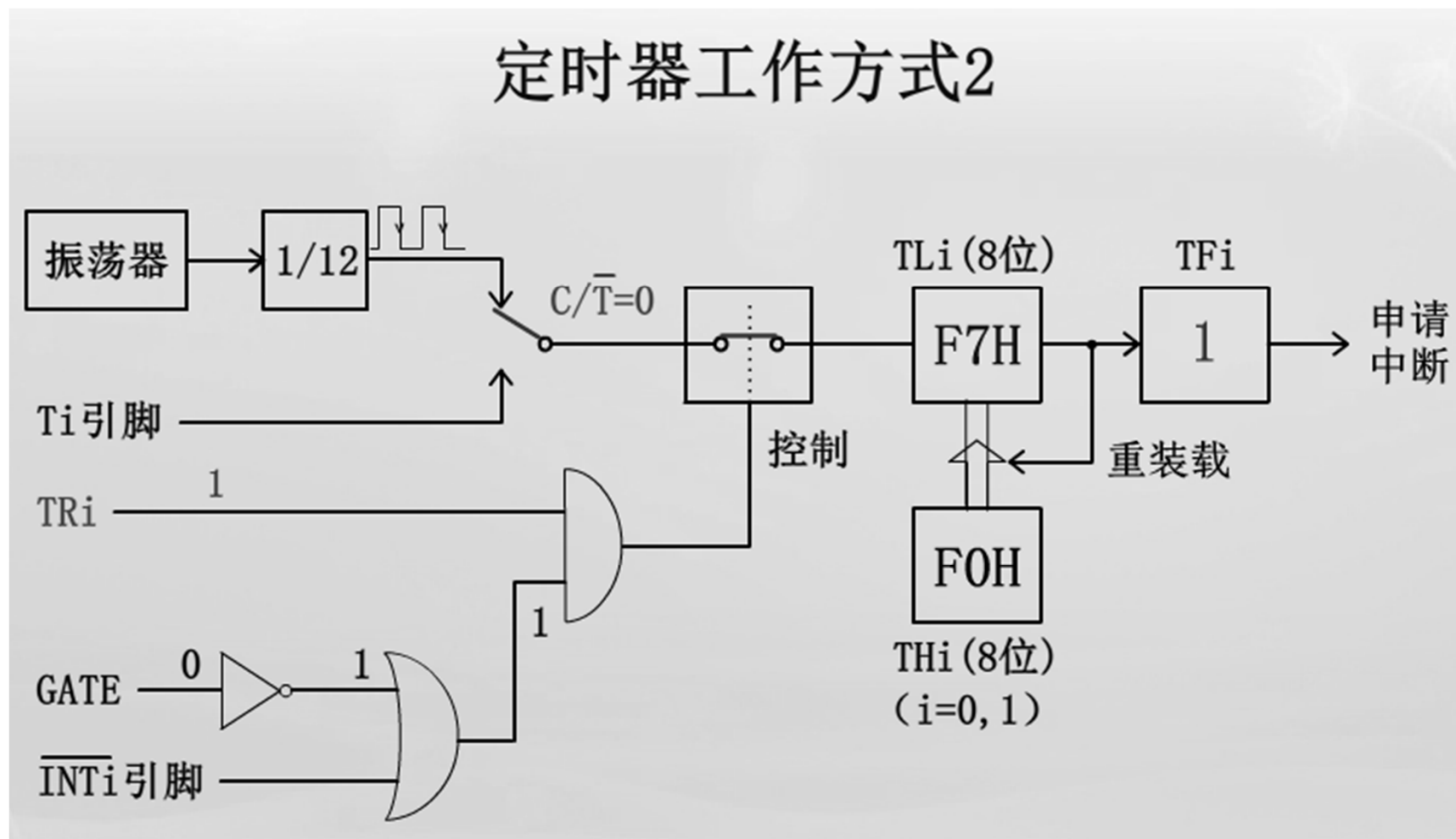
当M1M0设置为1 0时，Ti设定为工作方式2。该方式把TLi配置成一个独立工作的8位加1计数器，THi为重装载初值寄存器，因此称为**8位初值重装载方式**。其最大计数值为 $2^8=256$ 。

当TLi产生溢出时，一方面使溢出标志TFi置1，同时把THi中的8位数据重新装入TLi中，方式2不存在定时误差。



# 定时器计数器的的工作方式

## 2. 方式2



# 提 纲

## 4. 定时器计数器的初始化



# 定时器计数器的工初始化

## 1. 初始化步骤

- 1) 确定工作方式，即给方式寄存器TMOD赋值。
- 2) 预置定时初值或计数初值，将初值写入TH0、TL0或TH1、TL1中。
- 3) 中断设置（给IE赋值），允许或禁止定时器/计数器的中断。
- 4) 启动定时器/计数器，令TCON中的TR0或TR1为“1”。

## 2. 定时/计数初值的确定

对于加1计数器，设置的计数初值应是需要定时（计数）值相对于定时器/计数器最大计数值M的补码。

最大计数值M取决于计数器的位数L， $M=2^L$ 。方式1时L=16，最大计数值 $M=2^{16}=65536$ ；方式2时L=8，最大计数值 $M=2^8=256$ 。



# 定时器计数器的工初始化

## 2. 定时/计数初值的确定

设晶振频率为12MHz，则机器周期为1 $\mu$ s。

### (1) 对于方式1: $M=65536$

若定时初值 $X=0$ ，则定时时间为65536 $\mu$ s。若需要定时 $t$ 为20ms=20000 $\mu$ s，则定时计数初值 $X$ 应为：最长定时时间 - 需要定时时间=65536-20000=45536。加1计数器在该初值基础上，计数20000个机器周期脉冲，就发生溢出，表示20ms定时时间到。

### (2) 对于方式2: $M=256$

若定时初值 $X=0$ ，则定时时间为256 $\mu$ s。若需要定时 $t$ 为200 $\mu$ s，则定时初值 $X$ 应为：最长定时时间 - 需要定时时间=256-200=56。加1计数器在该初值基础上，计数200个机器周期脉冲，就发生溢出，表示200 $\mu$ s定时时间到。



## 定时器计数器的工初始化

### 2. 定时/计数初值的确定

设定时初值为X，则定时时间 $t=(M-X) \times 1\mu s$ ;

对于晶振频率为f的MCU系统，定时时间t为：

$$t = (M - X) \times \frac{12}{f} = (M - X) \times \text{机器周期} (\mu s)$$

如果要求定时的时间为t，则根据上式可以得到定时初值X：

$$X = M - \frac{tf}{12}$$



## 定时器计数器的工初始化

---

问题1：某8051微控制器采用的晶振频率为6MHz，请用T1产生1ms的定时，试计算定时初值？



## 定时器计数器的工初始化

问题1：某8051微控制器采用的晶振频率为6MHz，请用T1产生1ms的定时，试计算定时初值？

解：因为 $f=6\text{MHz}$ ，则机器周期为 $2\mu\text{s}$ 。

计数器需要累计的脉冲个数为 $1000\mu\text{s}/2\mu\text{s}=500$ ；

由于方式2的最大计数值是256，因此无法实现。

采用方式1，定时初值 $X=65536-500=65036=\text{FE0CH}$ ，即向TH1、TL1设置FEH、0CH。



## 定时器计数器的工初始化

**问题2：** 设晶振频率为12MHz，若要T1产生250 $\mu$ s的定时信号，试选择工作方式、确定定时初值？



## 定时器计数器的工初始化

**问题2：**设晶振频率为12MHz，若要T1产生250 $\mu$ s的定时信号，试选择工作方式、确定定时初值。

**解：**计数脉冲即机器周期是1 $\mu$ s；

则计数器需要累计的脉冲个数为250个。

选择工作方式2，定时初值 $X=256-250=6$ ，即向TL1、TH1设置初值为6。

对于计数方式，通常将计数初值设置为0，则经过一定时间（如1s）后，TH0、TL0或TH1、TL1的内容即为这段时间内记录的脉冲数。

# Thank you!

