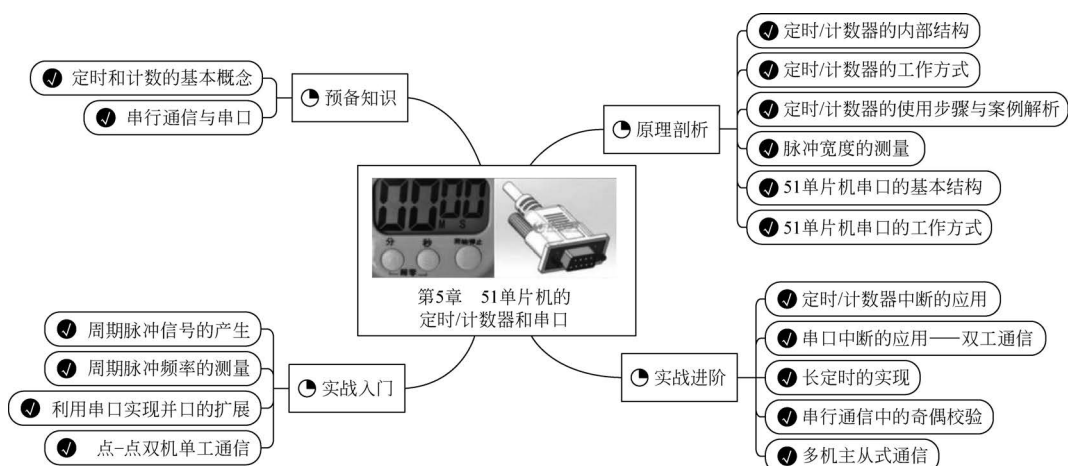


第 5 章



51 单片机的定时/计数器和串口



在前面各章的案例中都多次用到了定时。通过执行延时子程序实现期望的定时功能，称为软件定时。在执行软件定时的过程中，CPU 无法进行其他操作，效率比较低。考虑到定时功能在各种计算机系统中大量用到，目前几乎所有的单片机内部都集成了定时/计数器，利用这一内部资源可以实现可编程的定时和计数功能。

在现代社会生产和生活中，很多时候需要将多个单片机系统相互连接起来，实现信息的交换、传递和共享。当传输距离比较近时，可以采用速度比较快的并行传输。如果传输距离比较远，为节省系统构建和信息传输的成本，通常采用串行通信。

本章将对 51 单片机中的定时/计数器和串行通信接口这两大重要内部资源进行详细介绍。

5.1 磨刀霍霍——预备知识

作为本章内容的入门，这里首先对定时、计数、串行通信及接口的基本概念做个简要了解。

5.1.1 定时和计数的基本概念

从基本工作原理的角度看,51 单片机内部集成的定时/计数器实质上都是一个数字计数器,因此这里首先对数字电子技术中的计数器及相关概念做简要复习。

在数字电子技术中,利用 D 触发器或 JK 触发器的串联,可以实现多位二进制计数。以集成 4 位二进制计数器 74161 为例,其外部引脚如图 5-1 所示,功能表如表 5-1 所示。

74161 内部由 4 个 JK 触发器构成计数器。在引脚 $\overline{\text{CLR}}$ 为高电平期间,当引脚 $\overline{\text{LOAD}}$ 端送入负脉冲时,在时钟脉冲 CLK 的上升沿,由 A~D 端子送入的 4 位二进制数打入芯片内部,并立即出现在 $Q_A \sim Q_D$ 输出端。这 4 位二进制数称为预置数,又称为**计数初值**。

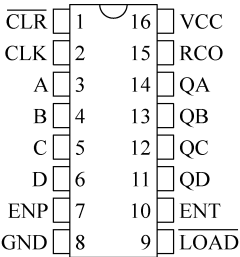


图 5-1 74161 的引脚

在 $\overline{\text{CLR}}$ 、ENP 和 ENT 都为高电平期间,由 CLK 端每送入一个时钟脉冲,计数器的**计数值**递增 1。CLK 端送入的周期脉冲称为**计数脉冲**。当 ENP 或 ENT 端为低电平时,内部计数器暂停计数,计数值将保持不变,称为**门控信号**。

当计数值达到 15,即 4 位二进制数为 1111B 时,RCO 端输出一个正脉冲,称为**计数溢出**。之后,又从 0000B 开始重新计数,也可以通过 LOAD 输入一个正脉冲**重装计数初值**并重新计数。

表 5-1 74161 的引脚功能表

清零	预置	使能		时钟	预置数据输入				计数值输出			
$\overline{\text{CLR}}$	$\overline{\text{LOAD}}$	ENP	ENT	CLK	A	B	C	D	QA	QB	QC	QD
L									L	L	L	L
H	L			上升沿	A	B	C	D	A	B	C	D
H	H	L							保持不变			
H	H		L						保持不变			
H	H	H	H	上升沿					计数			

1. 定时和计数

利用 74161 这样的数字计数器,最基本的功能当然是实现**计数**(Count),也就是统计在指定时间内计数脉冲的个数。例如,假设计数开始前设置计数初值为 0,由外部电路产生的脉冲作为计数脉冲从 CLK 端送入,则在某段给定的时间结束后,根据当前计数值即可确定该段时间内外部电路送入脉冲的个数。在实际系统中,利用计数功能可以实现外部事件出现次数的统计、脉冲宽度或频率的测量等功能。

如果计数脉冲来自一个标准的时钟脉冲,其频率或周期是已知的、恒定不变的,则计数器从给定的计数初值开始计数,直到计数溢出时,其间经过的时间可以由设定的计数初值确定,这样的功能称为**定时**(Time)。利用计数溢出信号可以实现很多定时控制功能,例如实现期望的延时,当期望的延时时间到后,切换 LED 的亮灭状态,或者使并口某个引脚的高/

低电平状态翻转以输出指定频率的周期脉冲等。

例如,假设计数脉冲周期为 1 ms,计数初值设为 0111B,则当计数溢出时,一共经过了 $15-7=8$ 个计数脉冲周期,因此从开始计数到计数溢出,一共经过了 8 ms 的时间。反之,如果希望从启动计数开始,要求经过 10 ms 的时间计数器出现溢出,则可以在计数开始前设置计数初值为 $15-10=5=0101\text{B}$ 。

由此可见,任何一个计数器都可以实现计数和定时功能,二者基本的区别在于计数脉冲是已知频率和周期的,还是频率和周期未知或者不确定的。如果计数脉冲是频率恒定的时钟脉冲,则一般用于实现定时;如果计数脉冲的频率或周期是未知的,或者不断变化的,则一般用于实现计数。实现定时功能的电路称为**定时器**(Timer);实现计数功能的电路称为**计数器**(Counter)。二者统称为**定时/计数器**。

2. 可编程定时/计数器

所谓**可编程定时/计数器**,指的是计数器的所有或部分功能(包括计数初值的装入,计数过程的启停、计数溢出信号的检测等)由 CPU 通过执行程序来实现,从而能够将硬件和软件实现的优点综合起来,实现灵活的设置和控制,又不占用 CPU 过多的时间。

一个典型的可编程定时/计数器与 CPU 的连接可以用图 5-2 表示。工作过程中,CPU 通过执行指令向定时/计数器送出计数初值,并发送启停控制信号。一旦启动,定时/计数器就在计数脉冲的作用下,不断地自动递增或递减计数。在计数过程中,CPU 还能随时通过执行相应的指令暂停或终止计数。

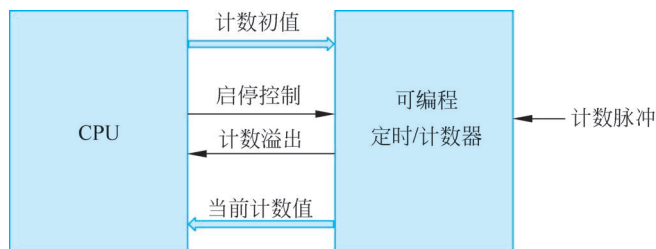


图 5-2 可编程定时/计数器与 CPU 的连接

在计数过程中,CPU 能随时通过指令读取计数值。当计数器发生计数溢出时,CPU 可以及时检测并作出反应,也可以在计数溢出时由定时/计数器主动向 CPU 发出信号标志。

上述所有操作都是由 CPU 内部通过执行指令实现的,也就是能够对定时/计数器的工作过程进行程序控制。例如,在程序中可以通过指令灵活地修改定时/计数器的计数初值以实现不同长度的定时,检测到计数溢出后执行预定的操作,根据需要暂停计数或者重新开始计数等。

MCS-51 的 51 子系列单片机内部集成了 2 个可编程定时/计数器 T0 和 T1,52 子系列还增加了一个 T2。这些定时/计数器作为 51 单片机集成的内部资源,可以实现 16 位、13 位或 8 位计数。每个定时/计数器都有多种工作方式可供选择,其中 T0 有 4 种,T1 和 T2 分别有 3 种。定时/计数器产生的进位溢出使相应的位单元置位,程序中通过查询该位

或采用中断方式进行处理。

3. 脉冲频率和宽度测量的基本原理

定时/计数器一个典型的应用是实现脉冲宽度、频率或周期的测量。这里说的脉冲可以是一个单脉冲或周期脉冲。对于周期脉冲,其宽度也就代表了脉冲的周期,周期的倒数等于脉冲的频率。

(1) 脉冲宽度和周期的测量。

为了测量一个脉冲的宽度,可以将待测脉冲作为计数器的启停控制信号(例如 74161 的门控信号 ENP 或 ENT),将另一个标准的时钟脉冲作为计数脉冲。在待测脉冲为高电平期间,计数器不断对计数脉冲进行计数,每来一个标准时钟脉冲,计数值递增 1。当待测脉冲高电平结束,脉冲变为低电平时,计数器立即停止计数。此时根据当前计数值和设置的计数初值以及标准脉冲的周期,即可计算得到待测脉冲高电平持续的时间,即脉冲的宽度。

假设标准脉冲的周期为 T_c ,计数初值为 N_0 ,待测脉冲高电平结束时刻的当前计数值为 N ,则待测脉冲的宽度为 $T = NT_c$ 。如果待测脉冲为占空比等于 50% 的周期方波,则将上述结果再乘以 2,即可得到脉冲的周期。上述测量原理可以用图 5-3 表示。

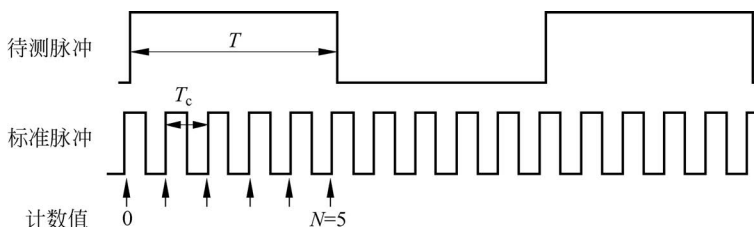


图 5-3 脉冲宽度的测量

显然,根据上述测量的基本原理,由于计数值只能是整数,因此待测脉冲的宽度必须大于标准脉冲的周期。另外,在计数的开始和结束时刻,待测脉冲和标准脉冲的正负跳变不一定同步,这将导致计数结果 N 出现 ± 1 个字的波动。因此,为保证测量精度,一般要求待测脉冲的宽度或周期远大于标准脉冲的周期。同时,一般的计数器都有计数范围的限制(例如 74161 的最大计数值为 15),这又要求待测脉冲相对于标准脉冲的周期不能太大,否则在计数过程中将出现溢出,导致测量结果错误。

(2) 脉冲频率的测量。

利用上述方法测得脉冲的周期后,取倒数即可得到周期脉冲的频率。在 51 单片机系统中,还广泛采用另一种方法实现周期脉冲频率的测量。这种测量方法是基于频率最基本的概念提出来的。

所谓周期脉冲的频率,指的是单位时间内脉冲的个数。因此,如果用一个计数器工作在定时方式,定时设为单位时间(例如 1 s),同时让另一个计数器工作在计数方式,在定时 1 s 时间内,统计待测脉冲的个数,则该计数值也就是脉冲的频率。实际系统中考虑到定时/计数器的位数、计数值的范围以及测量的实时性等,定时单位时间不一定是 1 s。例如,假设定时时间长度为 1 ms,则只需要将计数结果乘以 1000,即可得到待测脉冲的频率。也可以直

接用计数值表示频率,但单位为 kHz。

上述测量原理可以用图 5-4 表示。显然,采用这种方法测量频率,要求待测脉冲的周期不能超过定时的时间长度,或者说频率不能过低,否则将带来比较大的相对测量误差。另外,实际的计数器都有计数范围(最大计数值)的限制,这又决定了待测脉冲的频率不能太高,否则在定时的单位时间范围内计数器将出现溢出,导致计数和测量结果错误。

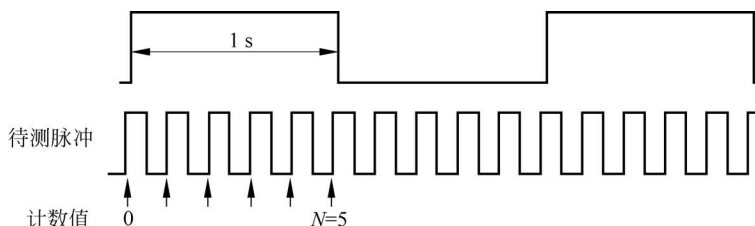


图 5-4 脉冲频率的测量

5.1.2 串行通信与串口

计算机系统中各部件之间的数据传输或者通信有两种基本的形式,即并行通信和串行通信。**并行通信**通常使用多条数据线将数据字节的各个位同时传输,每一位二进制数据分别通过一条传输线进行传输,另外还需要一条或几条控制信号线。并行通信的示意图如图 5-5(a)所示。

串行通信是将数据字节分成一位一位的形式在一条传输线上逐个传输。一次只能传输一位,对于 1 字节的数据,至少要分 8 位才能传输完毕,如图 5-5(b)所示。

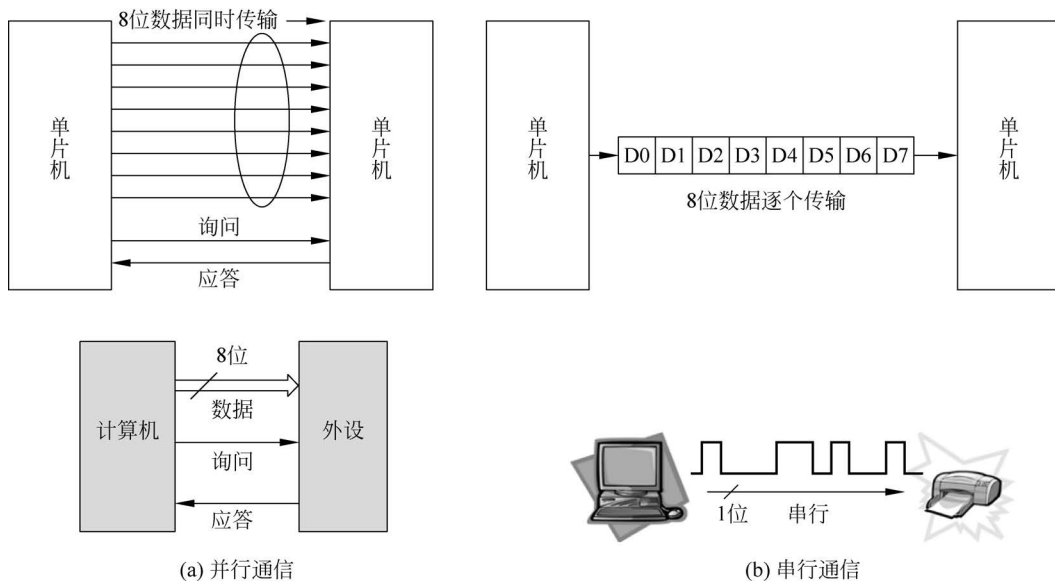


图 5-5 并行通信与串行通信

并行通信相对传输速度快。但由于传输线较多,长距离传输时成本高,因此这种方式适合于短距离的数据传输。串行通信所需的传输线少,连线简单,但传输速度慢,适用于远距离或数据量少的通信,其传输距离可以从几米直到几千千米。

1. 串行通信的基本概念

串行通信又可以分为同步和异步通信、单工和双工通信等。这里对相关的几个基本概念做些简要介绍。

(1) 同步通信和异步通信。

串行通信可以采用同步和异步两种通信方式。在**同步串行通信**中,通过一条同步时钟线将同步时钟同时加到收发双方,双方数据的发送和接收都由同一个时钟控制完全同步地进行。

同步传输用来对数据块进行传输,一个数据块中可以包含若干连续的字符数据,并严格按照事先约定的通信协议确定每个数据帧的传输格式(例如,数据的位数、数据中各位的传输顺序等),通信过程中以特定的位组合作为帧的开始和结束标志(称为同步字符)。同步通信中数据帧的格式如图 5-6 所示。

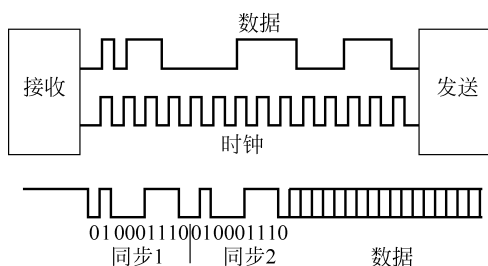


图 5-6 同步通信及数据帧格式

异步串行通信是指收、发双方分别使用各自的时钟控制数据的发送和接收,这样可省去连接收、发双方的一条同步时钟信号线,使得异步串行通信连接更加简单且容易实现。

异步传输以字符为单位进行数据传输,每个字符的传输都从起始位开始、到停止位结束,字符与字符之间的间隙(时间间隔)不固定,但字符中的各位以固定的时间传输,即字符之间是异步的,但同一字符内的各位之间是同步的。

在单片机中集成的串口采用的是异步串行通信方式,其传输的数据格式如图 5-7 所示。当没有数据发送时,数据线保持为高电平状态。在发送数据时,先发送起始位,即将数据线复位为低电平,然后再逐位发送数据。数据发送完毕后再使数据线变为高电平,即向接收方发送一位停止位。

异步串行通信不要求收、发双方时钟严格一致,实现容易,成本低,但是每个数据帧的传送都要附加起始位、停止位,有时还要再加上校验位,因此传输效率不高。

(2) 单工和双工通信。

按照信号传输的方向,一般又将串行通信分为单工方式、半双工方式和全双工方式三种。

所谓**单工方式**,指的是信号(不包括联络信号)在信道中只能沿一个方向传输,而不能沿相反方向传输,如图 5-8(a)所示。显然这种情况下只需要一根数据线即可实现数据的传输。

所谓**半双工方式**,指的是通信双方均具有发送和接收信息的能力,信道也具有双向传输性能,但是通信的任何一方不能同时既发送信息又接收信息,即在指定时刻数据只能沿某一

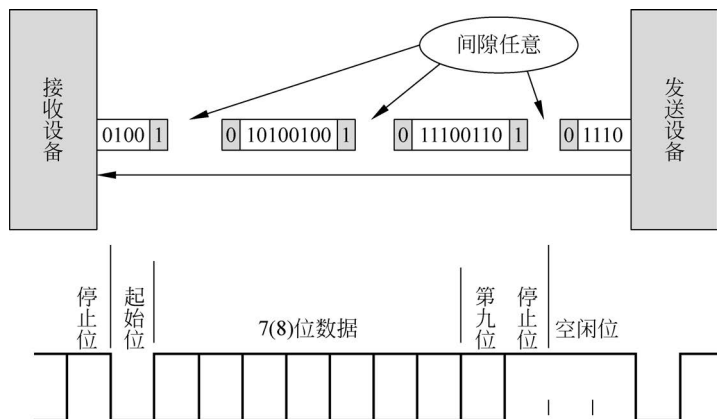


图 5-7 异步通信及数据格式

个方向传输,如图 5-8(b)所示。半双工方式大多采用双线制,在收发两端由内部电路进行发送和接收的切换。在外部发送和接收的数据都通过同一根线传输。

若信号在通信双方之间沿两个方向同时传输,任何一方在同一时刻既能发送又能接收信息,这样的方式称为**全双工方式**,如图 5-8(c)所示。显然,这种传输方式需要在发送端和接收端之间连接两根数据线。

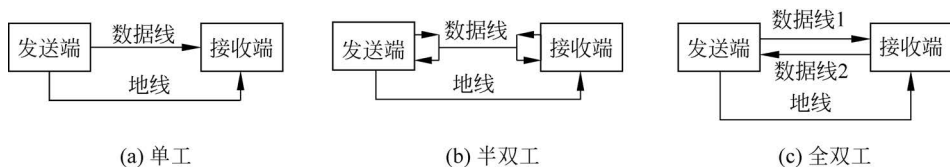


图 5-8 单工、半双工和全双工串行通信方式

(3) 数据传输速率。

串行通信的传输速率表示数据传输的快慢,一般可以用两种指标进行描述,即波特率和字符速率。

波特率(Baud Rate)指的是串行通信时每秒传送的码元数。在计算机通信中,一个码元一般指的是需要传输的一位二进制数据,单位为 bit/s,简写为 b/s,有时也写为 baud(波特)。串行通信常用的标准波特率在 RS-232C 标准中已有规定,典型的有 600、1200、2400、4800、9600、19200 和 115200 等。

在异步串行通信中,传输一个字符不仅需要传输字符本身(例如字符的 ASCII 码),还要在前后附加起始位和停止位,合起来称为一帧数据,代表一个字符或符号。因此串行通信的传输速率还可以用**字符速率**进行描述,又称为**符号速率**(Symbol Rate),单位为字符/秒。

例如:假设在异步串行通信中每个字符帧规定为 10 个数据位(1 位起始位、7 位数据位、1 位偶校验位和 1 位停止位)。如果已知传输的速率为 120 字符/秒,则波特率为 $120 \times 10 = 1200$ b/s,传输一位数据所需的时间为波特率的倒数,即 $1/1200 \approx 0.833$ ms。反之,如果已知波特率为 9600 b/s,则符号速率为 960 字符/秒,表示每秒传输 960 帧,传输一帧所需

的传输时间为 $1/960 \approx 1.04 \text{ ms}$ 。

2. 串口与串行通信接口标准

实现串行通信的接口称为串行通信接口,简称**串口**。实现同步和异步通信的接口分别称为通用同步通信接口和通用异步通信接口,又称为**通用同步收发器**(Universal Synchronous Receiver/Transmitter, **USRT**)和**通用异步收发器**(Universal Asynchronous Receiver/Transmitter, **UART**)。在 51 单片机中只有一个 UART,而在高档的嵌入式微处理器(例如 STM32 系列微处理器)中,还有 **USART**(通用同步异步收发器),既能实现同步通信,也能实现异步通信。

串口的两个最基本的功能是数据传输和数据格式转换。**数据传输**主要定义传输过程中的标准、数据帧格式及工作方式,控制具体的数据传输过程等。**数据格式转换**是将传输的数据进行串行和并行之间的相互转换,在发送端需要将并行数据转换为串行数据,而在接收端需要将串行数据转换为并行数据。

在两台单片机或者两台设备进行串行通信时,除了要求单片机和设备必须有串口以外,还需要用连接导线将收发双方的串口连接起来,并按照事先规定的协议或格式标准进行通信。

51 单片机串行通信接口的输入和输出信号均为 TTL 电平,即二进制 0 码和 1 码分别用低电平 0V 和高电平 +5V 表示。一般认为,当传输距离在 1.5 m 内时,可以将两台 51 单片机的串口直接相连,如图 5-9 所示。其中 RxD 和 TxD 分别为数据接收端和数据发送端,两台 51 单片机的 RxD 和 TxD 端交叉连接在一起。一台 51 单片机发送数据,则另一台 51 单片机接收数据。

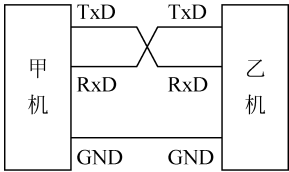


图 5-9 TTL 电平直接连接

这种直接以 TTL 电平实现串行数据传输的方法,抗干扰性差,传输距离短,传输速率低。为提高串行通信可靠性,增大串行通信距离和提高传输速率,在实际系统中可以采用格式**串行通信接口标准**,如 RS-232C、RS-422A、RS-485 等。

(1) RS-232C 接口。

RS-232C 是美国电子工业协会(Electronic Industry Association, EIA)推荐的串行通信总线标准,其全称是“使用二进制进行交换的数据终端设备(Data Terminal Equipment, DTE)和数据通信设备(Data Communication Equipment, DCE)之间的接口标准”。RS-232C 标准的传输距离在 15 m 之内,数据传输速率局限在 20 kb/s 以下,其传输速率主要有: 50、75、110、150、300、600、1200、2400、4800、9600、19200 b/s。

RS-232C 标准规定采用一种具有 25 根引脚的 25 针 D 型连接器(插针和插座)来连接通信双方的串口是一种标准的,而目前绝大多数计算机采用 9 针 D 型连接器,简称 DB9 连接器,如图 5-10 所示。

表 5-2 是 RS-232C 接口 DB9 连接器的引脚功能分配,其中最基本的两根引脚是 TxD 和 RxD。TxD 是发送数据引脚,数据传输时,数据位由该引脚发;RXD 是接收数据引脚,发送器发出的数据位由该引脚进入接收器。

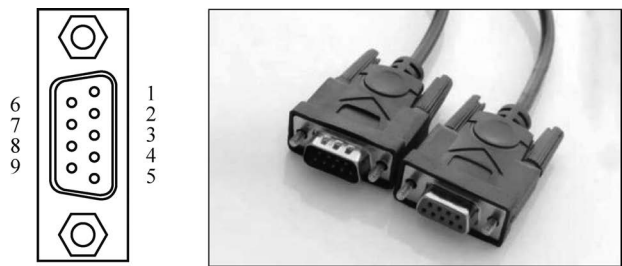


图 5-10 RS-232C 的 9 针 D 型连接器

表 5-2 RS-232C 接口 DB9 连接器的引脚功能

针 脚	信 号	功能(传输信号)
1	DCD	载波检测, Received Line Signal Detector(Data Carrier Detect)
2	RXD	接收数据, Received Data
3	TXD	发送数据, Transmit Data
4	DTR	数据终端准备好, Data Terminal Ready
5	SGND	信号地, Signal Ground
6	DSR	数据准备好, Data Set Ready
7	RTS	请求发送, Request To Send
8	CTS	清除发送, Clear To Send
9	RI	振铃提示, Ring Indicator

当传输的距离比较近时,利用 RS-232C 标准可以实现具有该标准接口的各台设备之间的直接连接。如果传输的距离比较远,需要通过调制解调器(MODEM)进行连接和数据传输,如图 5-11 所示。此时就需要用到表 5-2 中的其他引脚。限于篇幅,这里就不详细介绍了。

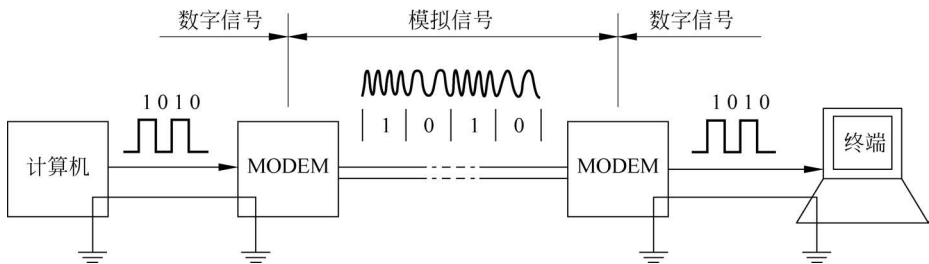


图 5-11 使用 MODEM 时 RS-232C 接口的连接

RS-232C 标准采用负逻辑电平,即逻辑 1 和 0 分贝分别用 $-15\sim-3\text{ V}$ 和 $+3\sim+15\text{ V}$ 表示。由此可见,RS-232C 的逻辑电平与 TTL 逻辑电平不兼容,因此必须进行电平转换。常用的电平转换集成芯片有 Motorola 公司制造的 MC1488 和 MC1489,MAXIM 公司生产的 MAX 系列 RS-232C 收发器 MAX232、MAX213E、MAX241E 芯片等。图 5-12 所示是采用芯片 MAX232A 实现电平转换时的电路连接。

(2) RS-422A 接口。

RS-232C 虽应用广泛,但推出较早,传输速率低、通信距离短、接口处信号易产生串扰,于是国际上又推出了 RS-422A 标准。

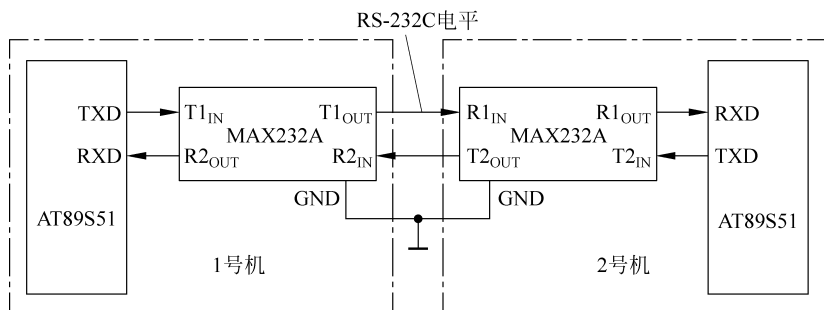


图 5-12 RS-232C 电平转换芯片的连接

RS-422A 与 RS-232C 主要区别是,收发双方信号地不再共地,RS-422A 采用了平衡驱动和差分接收的方法。每个方向用于数据传输的是两条平衡导线,这相当于两个单端驱动器。输入同一个信号时,其中一个驱动器输出永远是另一个驱动器的反相信号,从而使得两条线上传输的信号高/低电平时钟相反。若传输过程中混入了干扰和噪声(以共模形式出现),由于差分接收器的作用,就能识别有用信号并正确接收传输信息,使干扰和噪声相互抵消。

RS-422A 能在长距离、高速率下传输数据。最大传输率为 10 Mb/s,此速率下,电缆允许长度为 12 m,如采用较低速率,最大传输距离可达 1219 m。

(3) RS-485 接口。

RS-422A 接口需要四芯传输线,长距离通信不经济。在工业现场,常采用双绞线传输的 RS-485 串行通信接口。RS-485 是 RS-422A 的改进,二者的主要区别是:RS-422A 为全双工,采用两对平衡差分信号线;RS-485 为半双工,采用一对平衡差分信号线。

采用 RS-485 很容易实现 1 对 N 的多机通信。RS-485 标准允许最多并联 32 台发送器和 32 台接收器。RS-485 与 RS-422A 一样,最大传输距离约 1219 m,最大传输速率为 10 Mb/s。通信线路采用平衡双绞线,双绞线长度与传输速率成反比,只有在很短距离下才能获得最大传输速率,一般 100 m 长双绞线最大传输速率仅为 1 Mb/s。

5.2 小试牛刀——实战入门

在对 51 单片机中的定时/计数器和串口有了初步了解后,这里通过几个案例介绍其应用及程序设计方法。

动手实践 5-1: 周期脉冲信号的产生

要求由 51 单片机的 P1.0 引脚输出一个周期为 0.4 ms 的脉冲信号,脉冲的周期用 51 单片机内部的定时/计数器 T0 实现控制。假设 51 单片机的时钟频率为 6 MHz。

本案例的电路原理图如图 5-13 所示(参见文件 [ex5_1.pdsprj](#)),其中 51 单片机的 P1.0 引脚接入示波器的 A 通道,以便观察输出信号的波形。



微课视频

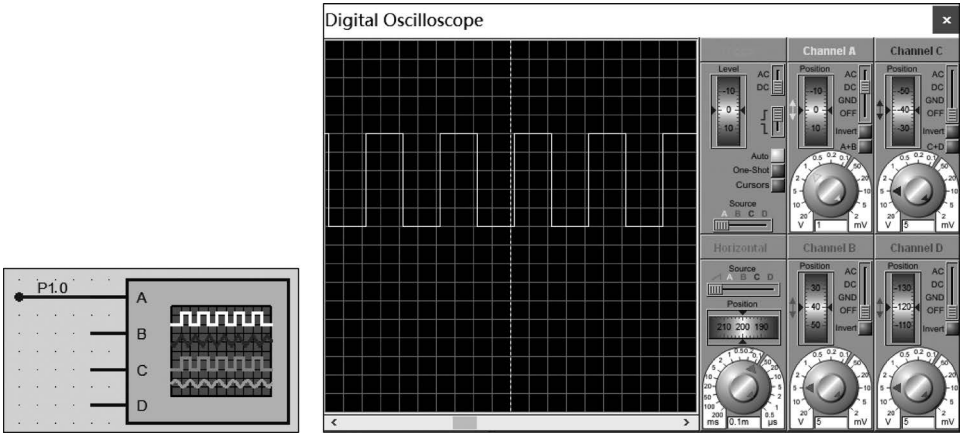


图 5-13 示波器电路符号及面板

示波器 Digital Oscilloscope 是 Proteus 中提供的一个虚拟仪器 (Virtual Instrument)，有 4 个输入端 (通道)，可以同时观察 4 个输入信号的时间波形。向原理图中调入示波器的方法有如下两种：

- (1) 在 Proteus 窗口左侧的工具栏中选择 Virtual Instruments Mode (虚拟仪器模式)，在该工具栏右侧列表框中单击选中 OSCILLOSCOPE 并将其放置到原理图中合适的位置；
- (2) 在原理图中右击，在弹出的快捷菜单中选择 Place/Virtual Instrument/OSCILLOSCOPE 命令。

本案例的主程序如下 (参见文件 [p5_1.asm](#))：

```

; 定时/计数器方式 2 的应用——周期脉冲信号的产生
SIG BIT P1.0 ; 定义输出信号引脚
ORG 0000H
AJMP MAIN
ORG 0100H ; 主程序
MAIN: MOV TMOD, #00000010B ; 设置 T0 工作方式 2
MOV TH0, #156 ; 设置计数初值
MOV TL0, #156
SETB TR0 ; 启动定时
LOOP: JNB TF0, $ ; 定时到?否,则等待
CLR TF0 ; 清除 TF0
CPL SIG ; 输出信号波形取反
SJMP LOOP ; 循环
END
```

运行程序前，注意在原理图中双击 51 单片机芯片，在弹出的对话框中设置 51 单片机的时钟频率为 6 MHz。启动运行后，将自动打开示波器面板。由示波器面板可以观察到 P1.0 输出脉冲的周期为 0.4 ms，脉冲幅度为 5 V。修改程序中的计数初值 156，重新编译，之后在 Proteus 中重新启动运行，可以观察到示波器上脉冲周期的变化。

动手实践 5-2：周期脉冲频率的测量

本案例的原理图如图 5-14 所示(参见文件 `ex5_2.pdsprj`)。图中用信号发生器 SIGNAL GENERATOR 产生频率范围为 1~250 kHz 的待测脉冲,并由 P3.5(第二功能为 T1 引脚)送入 51 单片机内部的定时/计数器 T1,作为其计数脉冲,同时送入示波器的 A 通道,以便观察其波形。

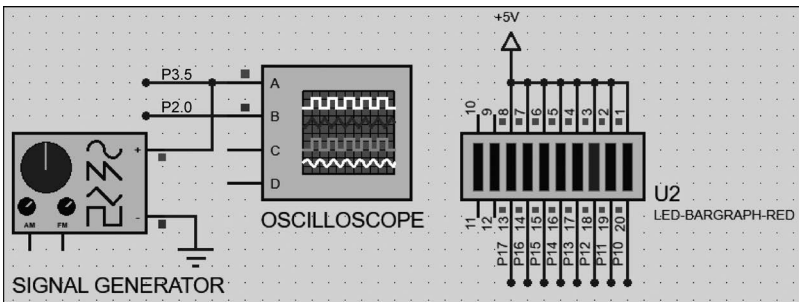


图 5-14 周期脉冲频率的测量电路原理图

根据前述频率测量的基本原理,本案例用 T0 工作在方式 0 重复实现 1 ms 定时,定时脉冲通过 P2.0 引脚送入示波器的 B 通道,以便观察其波形。在每次定时 1 ms 时间内,用定时/计数器 T1 统计待测脉冲的个数,计数结果即为待测脉冲的频率(以 kHz 为单位),以二进制形式用 8 个 LED 显示。

假设 51 单片机的时钟脉冲频率为 12 MHz。为了用 T0 工作在方式 0 实现 1 ms 的重复定时,所需的计数初值为 1C18H(具体计算方法将在后面介绍)。主程序如下(参见文件 `p5_2.asm`):

```

; 定时和计数功能的配合使用——周期脉冲频率的测量
SIG      BIT      P2.0
RESULT   EQU      P1
ORG      0000H
AJMP     MAIN
ORG      0100H
MAIN:    MOV      TMOD, #01100000B ; 设置 T0 工作方式 0 定时方式
                                ; T1 工作在方式 2 计数方式
LP:      MOV      TH0, #0E0H        ; 设置 T0 计数初值已定时 1 ms
MOV      TL0, #18H
MOV      TH1, #0                  ; 初始化 T1 计数初值为 0
MOV      TL1, #0
SETB     TR0                      ; 启动 T0 定时
SETB     TR1                      ; 启动 T1 计数
JNB      TF0, $                  ; 等待 T0 定时到
CLR      TF0
CPL      SIG                      ; 输出定时 1 ms 脉冲
CLR      TR1                      ; 停止 T1 计数,以便读取结果
MOV      A, TL1
CPL      A
MOV      RESULT, A                ; 输出显示测量结果
SJMP     LP
END
```



图 5-14



微课视频

在 Proteus 原理图中加载并启动上述程序的运行,将自动弹出示波器和信号发生器面板,如图 5-15 所示。通过信号发生器面板可以调节待测脉冲的频率,图中设置为 4 kHz。在示波器面板中,可以清楚地看到待测脉冲和 1 ms 定时脉冲。此时,根据 8 个 LED 的显示可知测量结果为 00000100B=4 kHz。调节待测脉冲的频率,可以看到 8 个 LED 亮灭状态的变化。

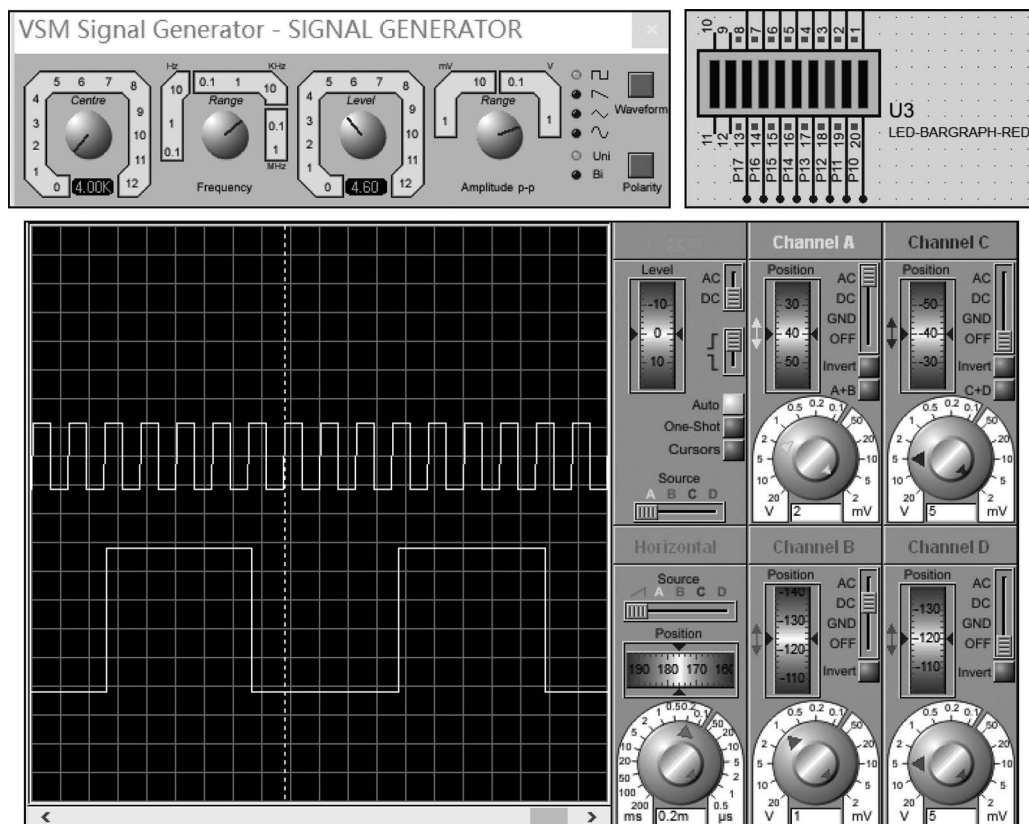


图 5-15 信号发生器和示波器面板

注意运行过程中调节待测脉冲的频率不能超过 255 kHz。例如,当待测脉冲频率为 300 kHz 时,根据上述测量原理,LED 应该显示 300 的二进制代码,已经超过 8 个 LED 能够显示的 8 位结果。待测脉冲频率更不能超过 500 kHz,否则无法得到正确的测量结果,具体原因将在后面解释。

动手实践 5-3: 利用串口实现并口的扩展

在 51 单片机中,串口一个重要的应用是实现并口和各种外部资源的扩展。如果一台 51 单片机系统在工作过程中不需要与外接进行串行通信,则可以利用串口扩展并口,也就是通过串口相关的引脚实现并行数据的输入/输出,相当于为 51 单片机系统增加了并口。

利用串口为 51 单片机扩展并口,在外部需要用到一些特殊的集成电路芯片,典型的有

4094、74164 和 74165 等。其中 4094 和 74164 是串入/并出芯片,74165 是并入/串出芯片。

本案例以扩展并口为例,介绍利用串口为 51 单片机扩展并口的基本方法。电路原理图如图 5-16 所示(参见文件 [ex5_3.pdsprj](#))。

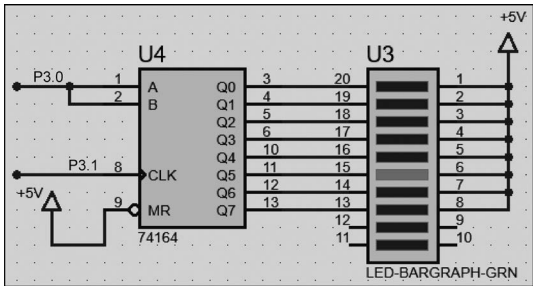


图 5-16 利用串口实现并口的扩展

在图 5-16 中,8 位串入/并出芯片 74164 的 A 和 B 直接短接,用作串行数据输入端,因此与 51 单片机的 RxD(P3.0 引脚的第二功能)相连接;CLK 端为串行时钟信号输入端,直接与 51 单片机的 TxD(P3.1 引脚的第二功能)相连接;Q0~Q7 为并行输出端,输出 8 位数据用于控制 LED 的亮灭。

引脚 MR 为清零端,当输入低电平时 8 个输出端清 0。本案例无需将输出端清零,因此 MR 端恒定地接电源。当 MR=1 时,如果没有 CLK 端的脉冲,74164 输出数据就一直保持不变。

本案例的主程序代码如下(参见文件 [p5_3.asm](#)):

```
ORG 0000H
AJMP MAIN
ORG 0100H
MAIN: MOV SCON, #00H      ; 串口初始化,方式 0
      MOV A, #0FEH
LP:   MOV SBUF, A          ; 串口发送一个数据
      JNB TI, $            ; 等待发送完毕
      ACALL DELAY          ; 延时
      CLR TI               ; 清除 TI 标志位
      RL A                 ; 数据循环移动一位
      SJMP LP              ; 循环
DELAY: MOV R7, #0FFH      ; 延时子程序
LP1:   MOV R6, #0FFH
      DJNZ R6, $
      DJNZ R7, LP1
      RET
END
```

在 Proteus 中加载程序,启动运行后,可以看到 8 个 LED 从下往上进行跑马灯显示。

动手实践 5-4: 点-点双机单工通信

在系统中,一台 51 单片机发送数据,并由另一台 51 单片机接收,两台 51 单片机的串口(或者通过 RS-232C、RS-485 接口实现电平转换后)直接连接起来,相当于在两台 51 单片机



图 5-16



微课视频

之间存在一条专用线路,这种串行通信称为**点-点双机单工通信**。

在图 5-17 所示原理图(参见文件 **ex5_4.pdsprj**)中,51 单片机 U2 从并口 P1 读入 8 个开关数据,并通过其串口引脚 RxD 传送给 51 单片机 U1。51 单片机 U1 从 TxD 引脚串行接收到 8 位数据后,立即通过并口 P2 输出,以控制 8 个 LED 的亮灭。注意设置两台 51 单片机的时钟频率为 12 MHz。



图 5-17

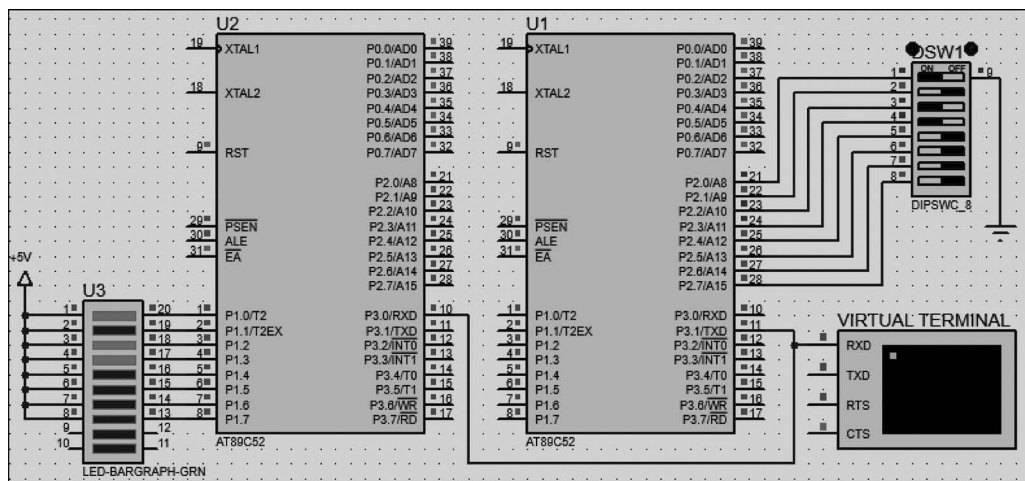


图 5-17 点-点双机单工通信连接原理图

本案例需要在 Keil C51 中创建两个工程,在第一个工程中创建如下源程序实现数据的串行发送(参见文件 **p5_4t.asm**):

```

; 串口点-点双机单工通信(发送)
SWT EQU P2
ORG 0000H
AJMP MAIN
ORG 0100H
MAIN: MOV SP, #60H
      MOV SCON, #01000000B ; 设置 51 单片机 U2 的串口方式 1, 禁止接收
      MOV TMOD, #00100000B ; 设置定时/计数器 T1
      MOV TL1, #0E6H ; 设置 T1 计数初值以确定波特率
      MOV TH1, #0E6H
      SETB TR1 ; 启动 T1 定时
      MOV SWT, #0FFH
LP0:  MOV A, SWT ; 读入开关数据
      CPL A
      MOV SBUF, A ; 通过串口发送
LP1:  JNB TI, LP1 ; 等待发送完毕
      CLR TI ; 清除 TI 标志位
      LJMP LP0 ; 循环
      SJMP $
      END

```

在第二个工程中创建如下源程序实现数据的串行接收(参见文件 **p5_4r.asm**):

```
; 串口点-点双机单工通信(接收)
LED EQU P1
ORG 0000H
AJMP MAIN
ORG 0100H
MAIN: MOV SP, # 60H
      MOV SCON, # 01010000B ; 51 单片机 U1 工作方式 1, 允许接收
      MOV TMOD, # 00100000B ; 设置定时/计数器 T1 工作方式 2 定时功能
      MOV TL1, # 0E6H ; 设置 T1 的计数初值以确定波特率
      MOV TH1, # 0E6H
      SETB TR1 ; 启动 T1 定时
LP: JNB RI, $ ; 清除接收中断请求标志位
    CLR RI ; 清除接收中断请求标志位
    MOV A, SBUF ; 从串口读接收到的数据
    CPL A ; 由 P1 口输出控制 LED
    MOV LED, A
    SJMP LP ; 等待串口接收数据发出中断
END
```

将上述两个工程生成的 HEX 文件分别加载到 U1 和 U2 单片机,之后启动程序运行。在运行过程中,单击 8 个开关以改变其通断状态,可以观察到 8 个 LED 的亮灭状态变化。

此外,在原理图的右下角还添加了一个虚拟终端(VIRTUAL TERMINAL),以便观察两台 51 单片机之间串行发送的 8 位数据。为调入该虚拟终端,在原理图中右击,在弹出的快捷菜单中选择 Place/Virtual Instrument/VIRTUAL TERMINAL 命令即可。

利用虚拟终端可以模拟 PC 的串口。将虚拟终端的 RxD 端子接到原理图中 51 单片机 U1 的 TxD 端(也就是 51 单片机 U2 的 RxD),则 U1 发送的数据不仅送到 U2,同时通过 PC 的串口送入虚拟终端,即可在虚拟终端上观察到 51 单片机串口传送的数据。注意双击虚拟终端,在弹出的对话框中设置 Bode Rate(波特率)为 1200 b/s。

在运行过程中,将自动弹出虚拟终端窗口(如图 5-18 所示),其中不断显示两台 51 单片机之间传输的 8 位开关数据。注意,在虚拟终端窗口右击,在弹出的快捷菜单中勾选 Hex Display Mode(十六进制显示模式)。

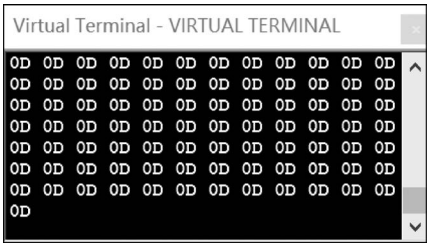


图 5-18 虚拟终端窗口

5.3 庖丁解牛——原理剖析

下面结合上述案例对 51 单片机内部集成的定时/计数器和串口进行详细介绍。

5.3.1 定时/计数器的内部结构

51 单片机内部集成的定时/计数器分别表示为 T0、T1,对 52 子系列还有一个 T2。这些定时/计数器在内部电路上相对独立,可以在应用系统中单独使用,也可以配合使用。在

51 单片机内部,3 个定时/计数器具有相同的结构,其使用方法也都类似。下面的介绍都以 T0 为例,对定时/计数器 T1 和 T2 来说,只需要将相应描述中的 0 换为 1 和 2 即可。

定时/计数器 T0 的内部结构如图 5-19 所示,其中最核心的是一个递增计数器,另外有很多控制电路。

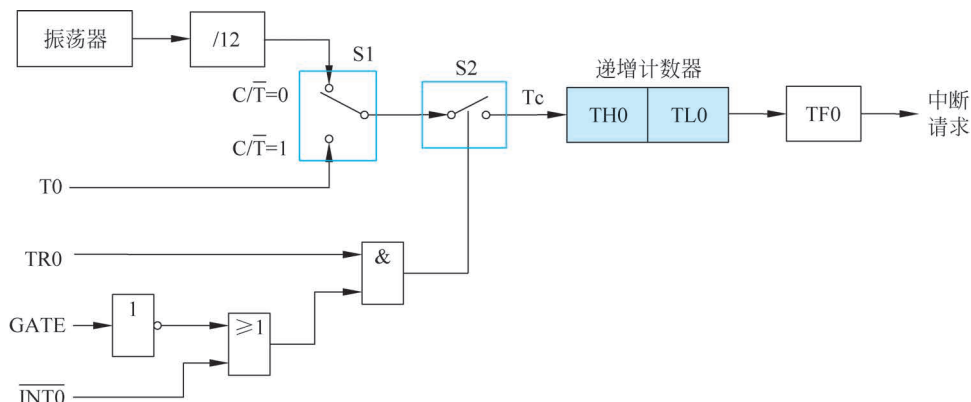


图 5-19 定时/计数器 T0 的内部结构

1. 递增计数器

51 单片机中的定时/计数器是一个递增计数器。启动计数前,可以通过程序指令将计数器的**计数初值**写入特殊功能寄存器 TH0 和 TL0。计数过程中,在每个计数脉冲 T_c 作用下,这两个特殊功能寄存器中保存的计数值会从计数初值开始不断递增变化,称为**当前计数值**。当计数值的最高位出现进位时,称为**计数溢出**,此时 TF0 标志位被置位,可用于向 CPU 发出中断请求,或者供 CPU 查询。

在 51 单片机中,通过程序配置,计数初值和当前计数值都可以是 8 位、13 位或 16 位二进制数据,从而实现 8 位、13 位或 16 位计数。

2. 计数脉冲控制电路

在图 5-19 所示中,除递增计数器和计数溢出标志位 TF0 以外,剩下的电路都是为了对计数器所需的计数脉冲进行控制而设置的。这部分电路的输出控制两个开关 S1 和 S2 的通断和切换,从而将不同的计数脉冲送入计数器,并实现计数的启停控制。

这些电路的输入信号有 $C/\bar{T}$ 、TR0、T0、GATE 和 $\overline{INT0}$,其中 T0 和 $\overline{INT0}$ 信号由外部电路产生,并分别通过 51 单片机的引脚 P3.4 和 P3.2 送入,作为定时/计数器的外部计数脉冲和门控信号。此时这两个引脚工作在第二功能,而不是用作普通的并口引脚。

上述 5 个信号中的另外 3 个信号 $C/\bar{T}$ 、TR0 和 GATE 不是由外部电路送来,而是由 51 单片机内部电路产生的。具体来说,这 3 个信号分别是特殊功能寄存器 TMOD 和 TCON 中的 3 位。当用指令向这两个特殊功能寄存器写入指定的 8 位二进制数据时,各位 1 码和 0 码就决定了对应的信号分别是高电平还是低电平,再送入计数器控制电路。这 3 个信号与两个特殊功能寄存器各位之间的对应关系将在后面介绍,这里先介绍这些信号的作用。

(1) $C/\bar{T}$: 计数和定时功能选择信号。当该信号为低电平时,开关 S1 打在上面,计数器

的计数脉冲由 51 单片机内部时钟电路产生的时钟脉冲经过 12 分频而得到的,因此其频率已知且恒定不变,此时定时/计数器实现定时功能。如果将该信号设置为高电平,则计数脉冲由外部电路通过 T0 引脚送入,此时定时/计数器实现计数功能。

(2) GATE: 门控允许信号。当 $GATE=0$ 时,不管 $\overline{INT0}$ 引脚送入高电平还是低电平,或门输出都为高电平,开关 S2 的通断与 $\overline{INT0}$ 信号无关。当 $GATE=1$ 时,在 $\overline{INT0}$ 引脚送入信号的高电平期间,S2 闭合,计数脉冲能够送到计数器,计数器进行正常计数;在 $\overline{INT0}$ 引脚送入信号的 low 电平期间,开关 S2 断开,计数脉冲不能够送到计数器,计数器停止计数 $\overline{INT0}$ 引入的信号称为门控信号。

(3) TR0: 计数启动信号。在或门输出高电平的前提下,当 $TR0=1$ 时,与门输出高电平,控制开关 S2 接通,允许计数脉冲送入计数器开始计数;否则,如果 $TR0=0$,则不能启动计数。

3. 相关的特殊功能寄存器

51 单片机的所有内部资源都是利用程序指令通过对特殊功能寄存器进行访问和控制的。定时/计数器作为一个重要的内部资源,在其内部电路中,计数初值和计数值保存在 TH0 和 TL0 中,可以采用与普通的内部 RAM 单元一样的程序指令对其进行读写访问。此外,与计数脉冲控制相关的 3 个信号分别对应 TMOD 和 TCON 这两个特殊功能寄存器中的一些位,下面着重介绍这两个特殊功能寄存器。

(1) TMOD。

TMOD 被称为定时方式 (Timer Mode) 寄存器。该寄存器在内部 RAM 中的地址为 89H,只能实现字节访问,不能进行位寻址。

TMOD 寄存器各位的含义如图 5-20 所示。其中高 4 位和低 4 位分别用于设置定时/计数器 T1 和 T0 的工作方式等。以低 4 位为例,其中 D3 位为 GATE,D2 位为 $C/\overline{T}$,分别用于设置门控允许信号和计数、定时功能。最低 2 位 M1M0 可以有 4 种组合,分别用于设置定时/计数器 T0 工作在方式 0、1、2 和 3 的这 4 种工作方式。

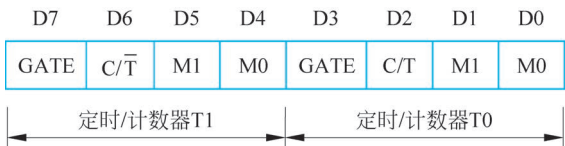


图 5-20 TMOD 寄存器各位的含义

例如,如下指令设置定时/计数器 T0 工作在方式 2 的定时方式,禁用门控;设置定时/计数器 T1 工作在方式 0 的计数方式,允许门控。

```
MOV    TMOD, #11000010B
```

(2) TCON。

在 3.3.6 节已经介绍过该寄存器,其低 4 位用于外部中断的设置,高 4 位用于对定时/计数器的控制,如图 5-21 所示。在高 4 位中,各名称后缀为 0 的位用于控制定时/计数器 T0;后缀为 1 的位用于控制定时/计数器 T1。

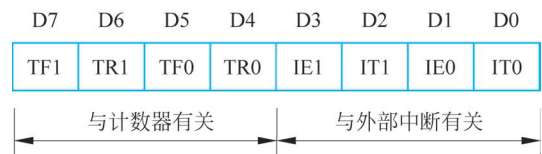


图 5-21 TCON 寄存器各位的含义

以 T0 为例,与之相关的是 TR0 和 TF0 位。其中,TR0 是启动位,用位操作指令设置该位为 1 时启动计数,将该位清零时停止计数;TF0 是计数溢出标志位,当计数溢出时,由硬件将该位设置为 1,可供程序检测。例如,如下指令执行后,将启动定时/计数器 T0 的计数。

```
SETB TR0
```

而如下指令:

```
JB TF0,TOUT
```

将检测 T0 计数是否溢出。当计数溢出时,TF0=1,则转到标号为 TOUT 的指令执行。

5.3.2 定时/计数器的工作方式

在开始计数前,通过向 TMOD 寄存器写入 8 位二进制数据,可以设置定时/计数器的工作方式。其中 T0 可以有 4 种工作方式,分别称为方式 0、方式 1、方式 2 和方式 3;而 T1 只有前面 3 种工作方式。

1. 方式 0 和 1

这两种工作方式的主要区别在于计数位数不同,其中方式 0 是 13 位计数,方式 1 是 16 位计数。

方式 1 是 16 位计数。计数开始前将 16 位计数初值的高、低 8 位直接存放到 TH0 和 TL0 中。在计数过程中,TH0 和 TL0 中存放的 16 位计数值将不断递增变化。当计数到 0FFFFH 时,如果再来一个计数脉冲,则计数值加 1 时,最高位出现进位,当前计数值回到 0000H,同时 TF0 被置位,发生计数溢出。之后,将从 0000H 开始重新计数。

方式 0 是 13 位计数。13 位计数初值的高 8 位存入 TH0,低 5 位存入 TL0 的低 5 位,TL0 的高 3 位始终为 0。例如,假设计数初值为 4660,先将其用 13 位二进制代码表示为 1001000110100B,再将其高 8 位 10010001B=91H 存入 TH0,低 5 位 10100B 前面添加 3 位 0 得到 TL0=00010100=14H。

2. 方式 2

方式 2 是 8 位计数。在计数开始前将 8 位计数初值同时存入 TH0 和 TL0。在计数过程中,TL0 不断做递增计数,而 TH0 中的计数初值保持不变。当计数到 0FFH 时,再来一个计数脉冲,则发生计数溢出。之后,内部电路控制将 TH0 中保存的计数初值重新装入 TL0,并从原来的计数初值重新开始计数。

由此可见,方式 2 与方式 0 和 1 的主要区别在于:方式 2 可以自动重装初值;方式 0 和 1 在计数溢出后,无法自动重新装入计数初值,只能从全 0 码开始重新计数。如果需要,则必

须在计数溢出后用指令将计数初值重新装入 TH0 和 TL0。

3. 方式 3

在方式 3 下,定时/计数器 T0 分为两个相互独立的 8 位计数器 TH0 和 TL0,如图 5-22 所示。其中 TH0 只能实现简单的定时功能,计数脉冲只能是 51 单片机内部时钟的 12 分频,并借用定时/计数器 T1 的 TR1 和 TF1,以控制定时的启停、作为计数溢出标志位。此外,计数溢出时不能自动重装初值。

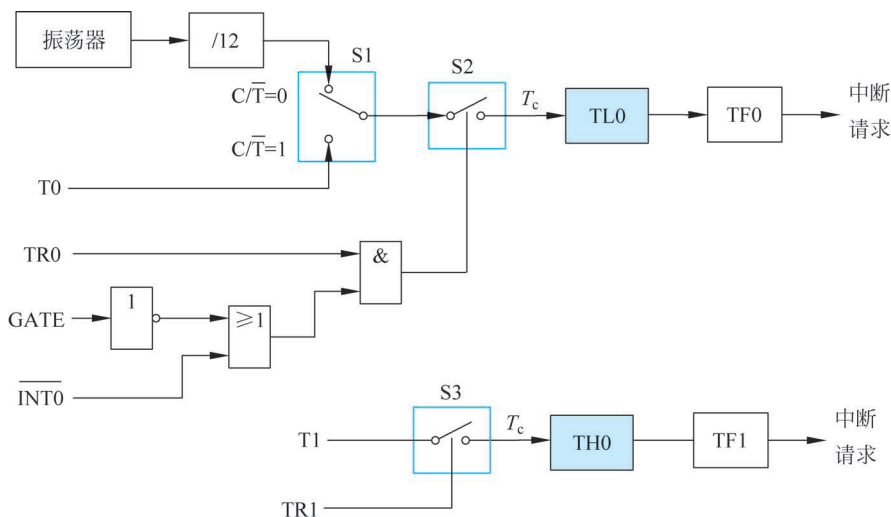


图 5-22 定时/计数器方式 3 的结构

在方式 3 下,TL0 占用原定时/计数器 T0 的控制位和引脚,实现与方式 0 和 1 类似的功能,只是计数位数不同。这种方式下,由于 TH0 单独用作另一个 8 位计数器,因此 TL0 也不能自动重装初值。

当设置定时/计数器 T0 工作在方式 3 时,由于 TH0 需要借用定时/计数器 T1 的 TR1 和 TF1,因此 T1 无法用 TR1 控制其计数的启停,也无法设置溢出标志位。在这种情况下,T1 一般用作串口通信所需的波特率发生器,在启动后就不断重复计数,产生的溢出脉冲经分频后作为串口工作所需的时钟脉冲。

5.3.3 定时/计数器的使用步骤与案例解析

根据前面的介绍,定时/计数器各种工作方式的原理是类似的,主要是计数位数不同。因此一般根据所需的计数初值或者在计数过程中计数器可能达到的最大计数值来选择和设置计数器的工作方式。之后,按照标准的步骤即可很方便地实现定时和计数功能。

1. 定时功能

如果是利用定时/计数器实现定时功能,则一般要求定时到后应能出现计数溢出,以便 CPU 能够通过检测 TF0 或 TF1 判断定时是否到。此时,可以根据所需定时时间的长短计算得到计数初值,再根据各种工作方式下计数初值的范围选择工作方式。

当计数初值 $N_0 = 0$ 时,方式 0~2 的计数过程从启动计数到计数溢出所需的计数脉冲

个数分别为 $2^{13}=8192$ 、 $2^{16}=65536$ 和 $2^8=256$ ，称为**最大计数值**或**计数满值**，记为 N_m 。如果计数初值 $N_0 \neq 0$ ，则计数器从 N_0 开始计数，必须在计数值达到 N_m 时才发生溢出。假设计数脉冲周期为 T_c ，则计数溢出时的定时时间为 $T=(N_m - N_0)T_c$ 。因此为实现期望的定时时间 T ，所需的计数初值应为 $N_0 = N_m - T/T_c$ 。

例如，假设 T0 工作在方式 1，已知 51 单片机的时钟频率为 6 MHz，则计数器的计数脉冲频率为 $6/12=0.5$ MHz， $T_c=2\ \mu\text{s}$ 。因此，为实现 1 ms 定时，所需的计数初值 $N_0 = N_m - T/T_c = 65536 - 1\ \text{ms}/2\ \mu\text{s} = 65036 = 0\text{FE0CH}$ 。

显然，对于不同的工作方式，为实现相同长度的定时，所需的计数初值是各不相同的。在上例中，如果设置 T0 工作在方式 0，则计数初值应为 $8192 - 500 = 7692$ 。如果设置工作在方式 2，则计数初值应为 $256 - 500 < 0$ ，这说明为实现 1 ms 定时，不能工作在方式 2。

在根据所需的定时时间计算计数初值，并根据计数初值确定定时/计数器的工作方式后，实现定时功能的程序只需要做如下设置和操作。

(1) 根据工作方式确定方式控制字，并用 MOV 指令存入 TMOD 寄存器。

(2) 用 MOV 指令将计数初值存入 TH0 和 TL0。对于方式 0 和方式 1，必须根据计数初值的高、低 8 位分别设置 TH0 和 TL0；对方式 2，必须用两条 MOV 指令将 8 位的计数初值同时存入 TH0 和 TL0。

(3) 用 SETB TR0 指令启动计数。

(4) 用 JB、JNB 或 JBC 指令检测 TF0 标志位，并等待计数溢出（定时到）。

(5) 定时到后，继续做相应的操作。

在**动手实践 5-1** 程序 **p5_1.asm** 的主程序中，第一条指令设置 TMOD 寄存器的方式控制字为 $02\text{H}=00000010\text{B}$ ，由其低 4 位指定定时/计数器 T0 的工作方式为方式 2、实现定时功能、禁用门控信号。由于本案例只用到了 T0，因此方式控制字的高 4 位没用，可任意设置为 0 码或 1 码，这里设置为 0 码。

程序接下来设置计数初值为 156（注意没有后缀，所以是十进制数）。由于 T0 工作在方式 2，因此用两条 MOV 指令将该计数初值同时送到 TH0 和 TL0。之后执行 SETB 指令将 TCON 寄存器中的 TR0 置位，从而启动定时。

由于在原理图中设置 51 单片机的时钟脉冲为 6 MHz，则定时时间为 $(256 - 156)T_c = 0.2\ \text{ms}$ 。在定时/计数器的定时计数过程中，CPU 循环执行程序中的 JNB 指令，不断检测计数溢出标志位 TF0。当计数溢出时，定时/计数器电路自动将 TF0 设置为 1，CPU 执行 JNB 指令检测到 $\text{TF0}=1$ 后退出循环。

显然，从启动定时到退出循环，定时/计数器 T0 定时 0.2 ms 时间到。因此，接下来将 TF0 溢出标志清除，以便等到下次定时到后重新将其置位。此外，定时到后通过执行 CPL 指令将 P1.0 引脚取反，使 P1.0 引脚上出现高/低电平的翻转。

由于定时/计数器 T0 工作在方式 2，因此在当前定时到后，计数初值将从 TH0 自动重新装入 TL0，又开始下一轮定时。在程序中，利用 SJMP 返回标号 LOOP 所在指令，又重复上述过程，等待下一次 0.2 ms 定时到后，又使 P1.0 引脚输出高/低电平翻转一次。

通过上述过程，在 P1.0 引脚输出高/低电平不断切换的周期脉冲。显然，每次高/低电

平切换的时间间隔等于利用定时/计数器 T0 每次循环定时的时间 0.2 ms,因此最后输出脉冲的周期为 0.4 ms。

2. 计数功能

如果是利用定时/计数器实现计数功能,则在初始化时一般将计数初值设为 0。启动计数后,每来一个计数脉冲,计数值递增 1。在期望的条件满足时,停止计数,并根据当前计数值求得希望的参数。例如在给定的一段时间内外部送来计数脉冲的个数,并进一步求得脉冲的周期或频率。

显然,如果计数器的最大计数值过小,在计数过程中将出现溢出,导致最后根据当前计数值计算所需参数的过程较复杂。因此为了实现计数功能,一般需要保证计数不溢出。此时,应根据计数脉冲的频率高低对可能达到的最大计数值 N_m 进行估计,并据此选择定时/计数器合适的工作方式。

- 如果 $N_m = 1 \sim 2^8$ (1~256),可以选择任何一种工作方式;
- 如果 $N_m = 2^8 \sim 2^{13}$ (256~8192),则不能设置为方式 2,可以选择方式 0 或 1;
- 如果 $N_m = 2^{13} \sim 2^{16}$ (8192~65536),则只能选择方式 1。

根据由 T0 引脚送入的外部脉冲频率或周期,确定为保证计数不溢出可能达到的计数值,并根据计数初值确定定时/计数器的工作方式。之后,在程序中只需要进行如下设置和操作。

(1) 根据工作方式确定方式控制字,并用 MOV 指令存入 TMOD 寄存器。

(2) 用 MOV 指令向 TH0 和 TL0 存入计数初值 00H。

(3) 用 SETB TR0 指令启动计数。

(4) 当规定的条件满足(例如定时 1 ms)时,停止计数并读取计数值。

在**动手实践 5-2**中,同时利用了 51 单片机内部两个定时/计数器实现待测脉冲频率的测量,其中 T0 实现定时 1 ms 的功能。本案例中同时将 T0 定时产生的 1 ms 定时脉冲由 P2.0 引脚输出送到示波器,以便观察。

在每次定时 1 ms 的过程中,用 T1 对通过 P3.5 引脚送入的待测脉冲进行计数。实现该案例的程序 p5_2.asm 的流程图如图 5-23 所示。

在程序中,首先设置方式控制字为 01100000B,其中,低 4 位确定 T0 工作在方式 0,实现定时功能,禁用门控信号;高 4 位确定 T1 工作在方式 2,实现计数功能,禁用门控信号。

做了上述初始化设置后,在之后的死循环中,利用 T0 不断定时 1 ms,并在每次定时时间范围内对待测脉冲进行

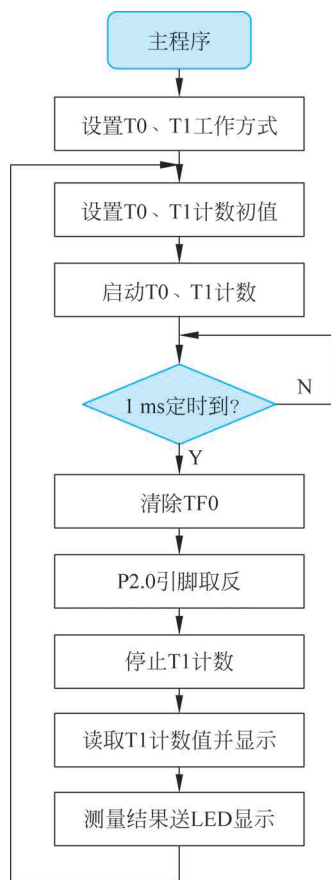


图 5-23 程序 p5_2.asm 流程图

计数,从而得到待测脉冲的频率送入 LED 显示。下面再对案例程序做一些相关说明。

(1) 计数初值的计算和设置。

在本案例中,定时计数器 T1 用于实现对待测脉冲的计数,因此设置其计数初值为 0,同时存入 TH1 和 TL1。程序中每定时 1 ms 测量一次频率后,再跳转到 LP 处,将 T1 的计数初值清零,以便重新开始下一次频率测量。

定时/计数器 T0 用于实现 1 ms 定时,其计数初值的计算方法如下:已知时钟频率为 12 MHz,则计数器计数脉冲频率为 $12/12=1\text{ MHz}$, $T_c=1\text{ }\mu\text{s}$ 。由于 T0 工作在方式 0,因此为实现 1 ms 定时,所需的计数初值 $N_0=N_m-T/T_c=8192-1\text{ ms}/1\text{ }\mu\text{s}=7192$ 。将该计数初值用 13 位二进制表示为 1110000011000B,其高 8 位为 1110000=0E0H,在低 5 位 11000B 之前添加 3 个 0,得到 00011000B=18H。程序中将 0E0H 和 18H 作为计数初值,分别存入 TH0 和 TL0。

特别需要注意的是,由于 T0 工作在方式 0,因此每次定时 1 ms 到后,不会自动重装初值。在主程序中,每次循环都必须跳转到标号 LP 处,用指令重装初值。

(2) T1 计数的启动和停止。

程序中启动 T0 和 T1 开始计数之后,利用 JNB 指令不断检测 TF0,等待 T0 定时 1 ms 到。在此过程中,T1 同时不断对由 P3.5 引脚送入的待测脉冲进行计数。

当检测到 TF0=1 时,T0 定时到,则清除 TF0,以便等到 T0 下一次定时到后重新将其置位。同时清除 TR1,以便停止 T1 计数,并从 TL1 读取当前计数值。

(3) 测量结果的显示。

每次定时 1 ms 到,停止 TR1 计数后,从 TL1 读取计数值,取反后由 P1 口输出控制 LED 的亮灭,以便观察测量结果。在上述过程中,之所以要将读取的计数值取反,是考虑到人眼观察的正常习惯。在 LED 上一般点亮表示 1 码,熄灭表示 0 码。如果将读取的计数值直接输出控制 LED,因为 LED 采用共阳极接法,则显示结果将刚好相反。

(4) 频率测量范围分析。

本案例中,在每次定时 1 ms 时间范围内由 T1 统计待测脉冲的个数。由于计数值只能为整数,因此在 1 ms 范围内至少需要送入一个待测脉冲。这就意味着待测脉冲的频率至少应为 1 kHz。

由于 T1 工作在方式 2,实现的是 8 位二进制数的计数。在每次循环测量时,将计数初值设为 0,因此当计数脉冲在 1 ms 内的个数超过 255 时,将出现计数溢出,计数过程又从 0 重新开始。在定时到时,根据当前计数值得到的测量结果将出现错误。因此在 1 ms 时间范围内待测脉冲的个数不能超过 255 个,也就意味着频率不能超过 255 kHz。

在调试过程中,读者试着调节信号发生器产生的脉冲频率,使其超过 255 kHz,观察 LED 显示的测量结果是否正确。

显然,如果需要扩大频率测量,可以增加计数位数。例如,如果设置 T1 为方式 1 以实现 16 位二进制数的计数,则待测脉冲的最高频率可以增大到 65535 kHz。

但是需要注意,在定时/计数器工作过程中,51 单片机需要两个机器周期的时间才能检测到 T0 或 T1 引脚上输入的一个待测脉冲,因此待测脉冲的频率不能超过 51 单片机时钟

脉冲频率的 1/24。假设时钟频率为 12 MHz,则待测脉冲的最高频率不能超过 500 kHz。

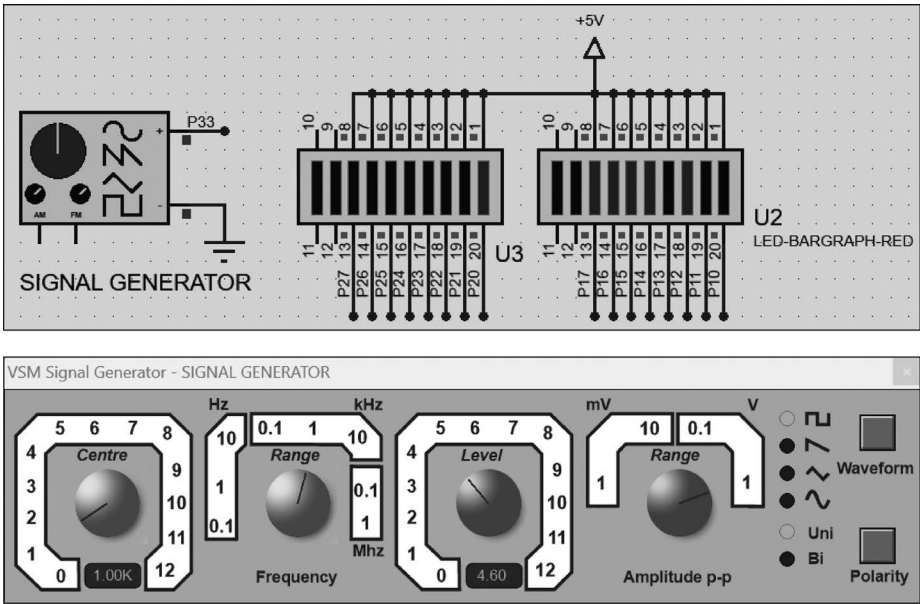
5.3.4 定时/计数器门控信号的作用

在定时/计数器中,当设置 GATE=0 时,计数的启停只需要用 TR0 或 TR1 进行控制。当设置 GATE=1 时,计数的启停不仅要受到 TR0 或 TR1 信号的控制,还要受到由 INT0 或 INT1 引脚送入门控信号的控制。利用这一特性,可以实现一些特殊的应用功能,例如脉冲宽度的测量。

下面通过一个具体案例体会门控信号的作用。

动手实践 5-5: 脉冲宽度的测量

本案例的电路原理图如图 5-24 所示(参见文件 [ex5_5.pdsprj](#))。与图 5-14 不同的是,这里将待测脉冲从 INT1 引脚(P3.3 的第二功能)引入,作为定时/计数器 T1 的门控信号。在该脉冲信号为高电平期间,T1 工作在定时方式对内部标准时钟进行计数。当高电平期间结束后,根据 T1 计数值即可求得脉冲的宽度。



微课视频



图 5-24

图 5-24 脉冲宽度的测量电路原理图

本案例的汇编语言程序如下(参见文件 [p5_5.asm](#)):

```
ORG 0000H
AJMP MAIN
ORG 0100H           ; 主程序
MAIN: MOV  TMOD, #10010000B ; 设置 T1 工作方式 1 定时门控方式
      SETB  TR1           ; 允许 T1 计数
```

```
LP:  MOV    TH1, # 0           ; T1 计数初值初始化为 0
      MOV    TL1, # 0
      JNB    P3.3, $          ; 等待脉冲低电平结束,高电平开始
      JB     P3.3, $          ; 等待脉冲高电平结束
      MOV    A, TL1           ; 读取计数值
      CPL    A
      MOV    P1, A            ; 显示测量结果
      MOV    A, TH1
      CPL    A
      MOV    P2, A
      SJMP   LP
      END
```

根据测量的基本原理,在本案例中用待测脉冲作为门控信号,控制定时/计数器 T1 对 51 单片机内部的标准脉冲进行计数。程序中设置定时/计数器 T1 工作在方式 1 的定时方式,并设置 GATE1=1。需要注意的是,由于是对标准脉冲进行计数,因此 T1 实现的是定时功能而不是计数功能,C/T 位必须设为 0。

设置工作方式以后,程序中用 SETB 指令将 TR1 置位,该操作的作用是使 TR1=1,但计数过程不一定已经启动。在设置的方式控制字中 GATE=1,因此必须等到门控信号(即待测脉冲)高电平到来时才开始计数。

程序之后用了一个大循环,不断循环测量并显示待测脉冲的周期。下面对每次循环实现的主要操作做一些解释说明。

- (1) 循环一开始(即每次计数测量开始),将计数器的初值设为 0,存入 TH1 和 TL1。
- (2) 设置计数初值后,程序中用 JNB 指令检测门控信号的低电平是否结束,等待高电平的到来。

(3) 一旦门控信号高电平到来,计数器开始对内部标准计数脉冲进行计数。而 CPU 通过执行程序中的 JB 指令等待门控信号的高电平结束。当门控信号高电平结束时,计数器自动停止计数,此时 CPU 读取计数值即可得到测量结果。

上述测量过程可以用图 5-25 表示。在 Proteus 原理图中设置 51 单片机的时钟频率为 12 MHz,加载上述程序并启动运行,设置信号发生器输出脉冲的频率为 1 kHz,16 个 LED 上显示 16 位二进制数为 01F4H=500,表示脉冲的高电平宽度为 500 μs。

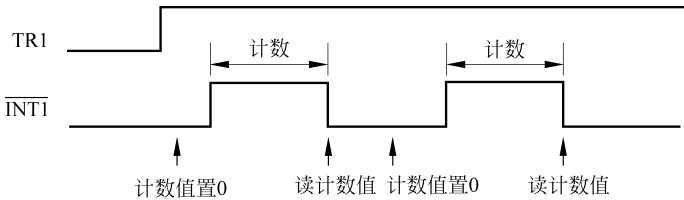


图 5-25 脉冲宽度的测量过程示意图

5.3.5 51 单片机串口的基本结构

51 单片机具有一个全双工的串行异步通信接口 UART,可以同时发送、接收数据。该

串口有四种工作方式,发送和接收数据可通过查询或中断方式处理,使用十分灵活。

51 单片机集成的 UART 也主要实现数据传送和数据格式转换功能,其内部结构如图 5-26 所示,其中主要包括数据缓冲器、发送控制器、接收控制器、输出控制门和输入移位寄存器等。从应用的角度看,可以归纳为 3 个特殊功能寄存器,即 **串口数据缓冲器** (Serial Buffer, SBUF)、**串口控制寄存器** (Serial Controller, SCON) 和 **电源控制寄存器** (Power Controller, PCON)。此外,在串口工作过程中还要用到定时/计数器 T1。

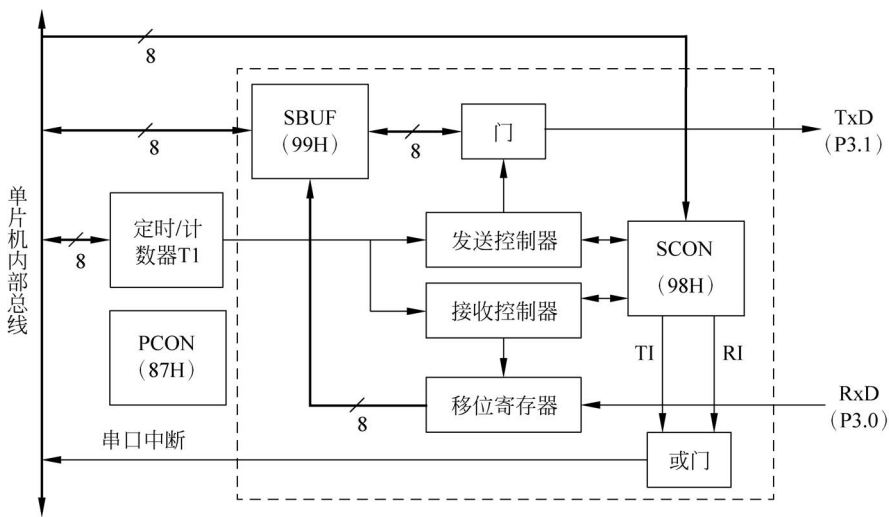


图 5-26 串口的内部结构

1. SBUF

串口内部有一个 8 位的数据缓冲器 SBUF,用于存放待发送和接收到的数据。该特殊功能寄存器的地址为 99H,可以通过名称进行字节操作,不能进行位寻址。

例如,如下指令:

```
MOV SBUF, A
```

将累加器 A 中的数据送入 SBUF,然后由串口内部电路控制发送出去。而如下指令:

```
MOV A, SBUF
```

将 SBUF 中存放的数据读取出来存入累加器 A。

2. SCON

51 单片机内部通过串口控制寄存器 SCON 进行访问,可以对串口内部的发送和接收控制器、输入移位寄存器以及门电路进行控制,以便在 **波特率发生器** (定时/计数器 T1) 送来的时钟信号控制下,通过输入移位寄存器将外部的串行数据转换为并行数据存入 SBUF,或者由门电路将 SBUF 中的并行数据转换为串行数据输出。同时,控制或门电路发出串口中断请求 **RI** (Receive Interruption, **接收中断**) 和 **TI** (Transmission Interruption, **发送中断**)

信号。

特殊功能寄存器 SCON 的地址为 98H, 可以进行字节访问或位操作, 各位的格式如图 5-27 所示。这里首先对最常用的 SM0、SM1、RI 和 TI 位的含义做些解释和说明, 其余各位将在后面再做介绍。

D7	D6	D5	D4	D3	D2	D1	D0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

图 5-27 SCON 寄存器各位的含义

- (1) SM0 和 SM1: 这两位二进制数的组合 00~11 分别指定串口的工作方式为方式 0~3。
- (2) REN: 控制是否允许串口接收数据。通过指令将该位置位, 则允许串口接收数据; 否则禁止串口接收数据, 只能进行数据发送。
- (3) RI: 接收中断标志位。当串口接收到一个数据时, 自动将该位置位。因此通过检测 RI 位的状态, CPU 可以判断一个字符帧是否接收完毕, 能否从接收 SBUF 读取接收到的数据。当 RI=1 时, 串口不会接收数据。因此, 在接收完一个数据并被 CPU 读取后, 必须将 RI 清零, 以便等待接收下一个数据。
- (4) TI: 发送中断标志位。当一个数据发送完毕后, 该位自动置位, 因此通过检测 TI 状态, CPU 可以确定一个字符帧是否发送完毕, 是否可以向串口写入下一个需要发送的数据。当 TI=1 时, 串口不会发送数据, 因此一个数据发送完毕后必须将 TI 清零, 以便准备发送下一个数据。

3. PCON 与波特率发生器

在 51 单片机中, 串口进行数据传送所需的波特率由 PCON 和波特率发生器共同决定, 涉及特殊功能寄存器有 TMOD、TCON、PCON、TL1、TH1 等。具体来说, 在不同的工作模式下, 串口的波特率确定方法不同。

- (1) 方式 0 的波特率固定不变, 始终等于 51 单片机时钟频率的 1/12。
- (2) 方式 2 的波特率只有两种, 等于时钟频率的 1/32 或 1/64, 具体取决对 PCON 寄存器中最高位 SMOD 位的设置。当设置该位为 0 时, 方式 2 的波特率为时钟频率的 1/64; 当设置该位为 1 时, 方式 2 的波特率为时钟频率的 1/32。
- (3) 方式 1 和方式 3 的波特率由 SMOD 位以及定时/计数器 T1 的计数初值和定时时间共同决定。此时, T1 必须工作在方式 2, 定时时间为 (256-计数初值)×12/时钟频率, 相应的波特率为

$$\text{波特率} = 2^{\text{SMOD}} / \text{T1 的定时时间} / 32$$

根据上式, 如果给定串口的波特率, T1 的计数初值应设为

$$\text{计数初值} = 256 - \text{时钟频率} \times 2^{\text{SMOD}} / (12 \times \text{波特率} \times 32)$$

例如, 假设 51 单片机的时钟频率为 12 MHz, SMOD=0, 要求串口传输的波特率为 1200 b/s, 则

$$\text{计数初值} = 256 - 12 \times 10^6 \times 2^0 / (12 \times 1200 \times 32) \approx 230 = 0\text{E6H}$$

5.3.6 51 单片机串口的工作方式

通过设置 SCON 中的最高 2 位,可以确定 51 单片机的 UART 工作在 4 种不同的方式,其中方式 0 一般用于外接移位寄存器芯片扩展 I/O 接口,方式 1 通常用于双机通信,方式 2 和方式 3 通常用于多机通信。这里首先介绍简单的方式 0 和方式 1。

1. 方式 0

方式 0 称为**同步移位寄存器方式**。在这种工作模式下,数据传输的波特率固定为 51 单片机时钟频率的 1/12。在传输数据的过程中,RXD(P3.0 引脚的第二功能)用作数据线,传输输入或输出的数据;TXD(P3.1 引脚的第二功能)用作时钟线,输出同步移位时钟。每次传输,发送和接收 8 位数据,数据由低位到高位逐位传输。

对发送过程,当 TI=0 时,将累加器 A 中实现存入的需要发送的数据由串口内部电路控制逐位移出,并由 TxD 引脚送出 51 单片机。发送完毕时,TI 自动置位。方式 0 数据的发送时序如图 5-28(a)所示。

对接收过程,首先必须设置 SCON 中的 D4 位 REN=1,并且在 RI=0 时,通过串口内部电路将从 RxD 引脚送入的数据逐位存入 SBUF。当接收完 8 位数据后,串口将 RI 置位。CPU 在此过程中不断检测 RI,当检测到 RI=1 时,从 SBUF 寄存器中读取接收到的数据存入累加器 A。数据的接收时序如图 5-28(b)所示。

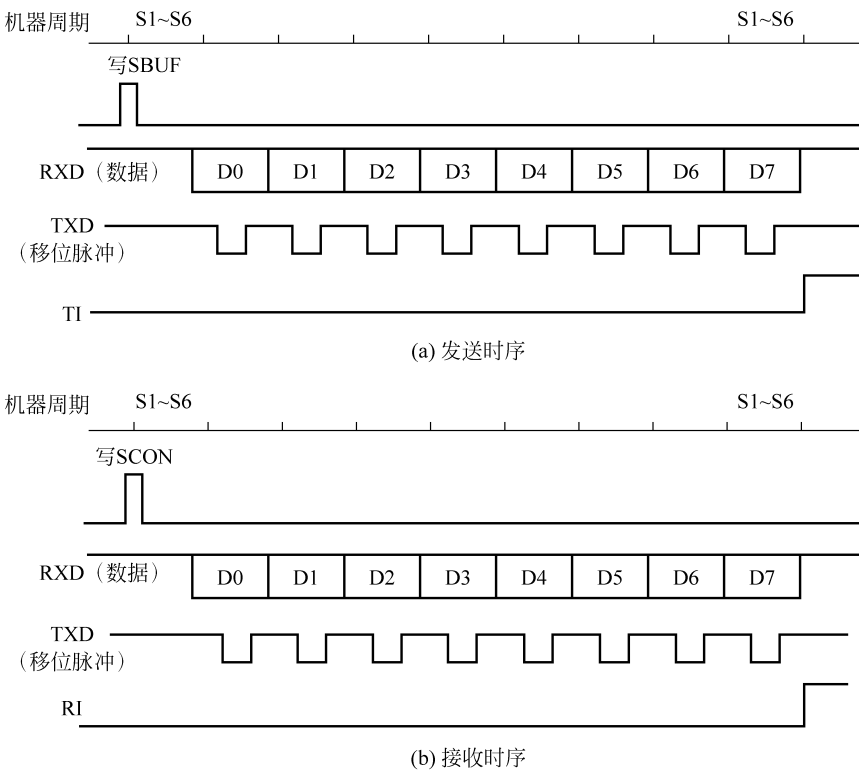


图 5-28 方式 0 数据的收发时序

在**动手实践 5-3**中,将 74164 芯片的 A、B 端和 CLK 端分别连接到 51 单片机的 RxD 和 TxD 引脚,由 TxD 送来的时钟脉冲正好作为 74164 的移位时钟,在其作用下,将 RxD 送来的各位数据依次存入 74164。

在程序 **p5_3.asm** 中,当 CPU 执行完串口的初始化后,通过执行指令

```
MOV SBUF, A
```

将数据存入 SBUF,并启动数据的发送过程。8 位数据从 RxD 逐位串行送出,TxD 端同步输出时钟脉冲。在该时钟脉冲作用下,74161 将从 RxD 线上接收到的数据逐位移入,并立即输出控制各 LED 的亮灭。

之后,CPU 执行 JNB 指令不断检测串口的 TI 位。当检测到 TI=1 时,表示 8 位数据已经发送完毕。经过适当延时以等待 LED 可靠显示当前 8 位数据后,清除 TI 标志位。再将数据循环左移一位,并返回循环的开始,将左移一位后的数据通过串口重新发送出去。

当 TxD 端送来 8 个时钟脉冲后,8 位数据完整地存入 74164,此时 TxD 不再送来时钟脉冲(保持恒定的高电平),74164 输出 8 位恒定的二进制数据,控制 LED 稳定地点亮或熄灭。如果将 RxD 和 TxD 信号接入示波器,运行过程中由示波器窗口观察到这两个信号的工作波形如图 5-29 所示。

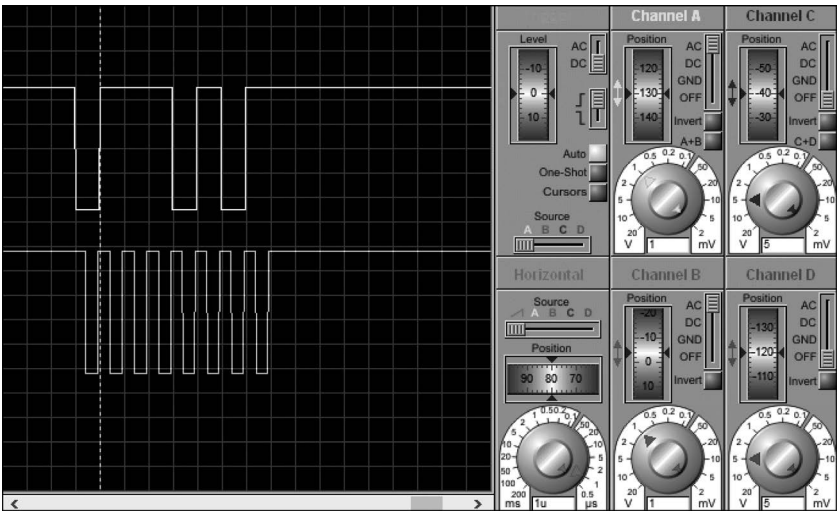


图 5-29 串口 RxD 和 TxD 信号的波形

2. 方式 1

方式 1 称为**8 位异步通信方式**,一个数据帧包括 1 位起始位、8 位数据位和 1 位停止位。传送数据的过程中,发送的数据通过 TxD 引脚送出 51 单片机,接收的数据逐位从 RxD 引脚送入。

显然,方式 1 与方式 0 的一个重要区别是,RxD 和 TxD 都用作数据线,没有专门的时钟线,所以属于异步通信。此外,传输的数据帧格式不同,方式 1 每次数据传输,实际上一共传

输了 10 位二进制数,所以有些地方又称为 10 位异步通信方式,方式 1 的数据收发时序如图 5-30 所示。

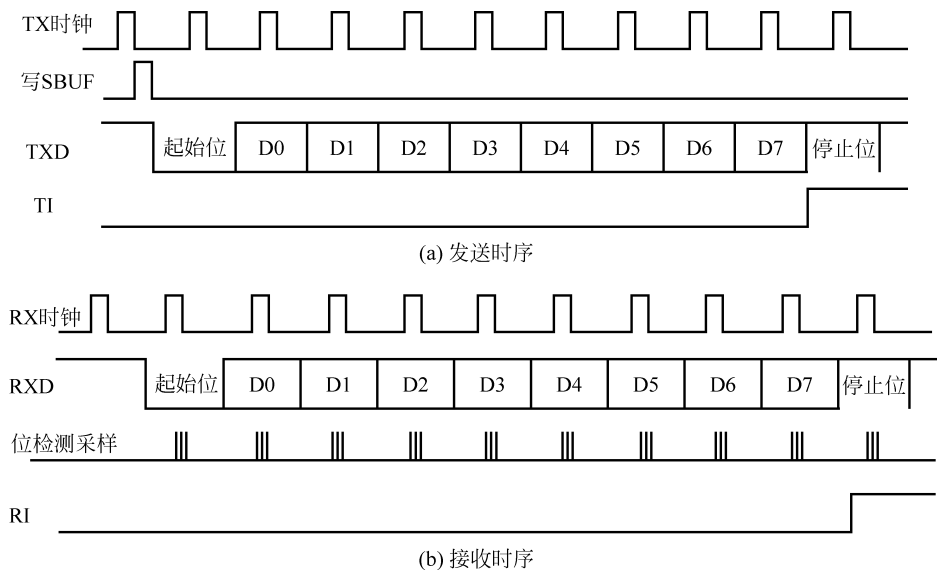


图 5-30 方式 1 数据的收发时序

需要强调的是,对串口的方式 1 在程序中进行初始化时,必须同时对定时/计数器 T1 进行初始化,以便确定传输的波特率和传输速率。

在方式 1 下,51 单片机串口可以实现单工或全双工通信。从程序上看,如果某个系统中两台 51 单片机之间数据是单向传输(例如一台 51 单片机是主机,另一台 51 单片机为从机,数据只由主机发送从机),则可以认为是单工方式。

基于上述思路,在[动手实践 5-4](#)中,假设 51 单片机 U1 只能发送,U2 只是接收,因此将 51 单片机 U1 的 TxD 与 U2 的 RxD 引脚相连接,并且两台 51 单片机分别加载执行不同的程序。

在发送程序 [p5_4t.asm](#) 中,首先对串口和定时/计数器 T1 进行初始化。假设 51 单片机时钟频率为 12 MHz,T1 工作在方式 2,则程序中设置计数初值为 0E6H,对应的波特率为 1200 b/s。注意 51 单片机复位后 PCON 中的最高位 SMOD=0。之后启动 T1 定时,产生周期脉冲作为串口数据传送所需的时钟脉冲。在后面的循环程序中,每次循环从 P2 口读取开关数据,存入 SBUF 后立即由串口发送出去。之后,CPU 通过执行 JNB 指令检测 TI 并等待数据发送完毕,再重复上述过程。

在接收程序 [p5_4r.asm](#) 中,首先设置串口和定时/计数器 T1 的工作方式并启动 T1 定时,为串口提供波特率时钟。注意在方式控制字中,必须设置 REN=1,以便允许 51 单片机 U2 的串口接收数据。之后,不断检测 RI 标志位。当检测到 RI=1 时,表示串口接收到一个有效数据,则将 RI 复位,同时 51 单片机中的 CPU 从 SBUF 中读取接收到的数据,并由 P1 口输出控制 LED。

5.3.7 51 单片机的内部中断

定时/计数器在计数过程中,一旦计数溢出,TCON 寄存器中的 TF0 位将被置位。如果允许,则其可以立即向 CPU 发出中断请求,称为**定时/计数器中断**。51 子系列内部有两个定时/计数器 T0 和 T1,52 子系列还有一个定时/计数器 T2。每个定时/计数器都可以发出中断请求。

同样地,对串口来说,发送端每执行一次数据发送,发送完毕时将自动使 SCON 中的 TI 位置位;接收端每接收到一个数据帧,并将其中的 8 位数据存入 SBUF 后,将自动使 RI 位置位。在上述两种情况下,如果允许,则串口都将向 CPU 发出中断请求,以便通知 CPU 开始发送下一个数据,或者从串口读取接收到的数据。串口发出的中断称为**串口中断**。

与通过 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 引脚送入的外部中断不同,定时/计数器中断和串口中断是由 51 单片机内部的定时/计数器和串口产生的,称为**内部中断**。

至此,已经介绍了 51 单片机中的所有中断源,其中包括 2 个外部中断、2 个或 3 个定时/计数器中断、1 个串口中断。在 51 单片机中,对这些中断源的管理和控制由专门的硬件和软件配合实现,这些硬件和软件构成 51 单片机的**中断系统**。图 5-31 所示为 51 子系列的中断系统。图中,T0 和 T1 为定时/计数器中断,TX 和 RX 为串口的发送中断和接收中断,2 个串口中断 TX 和 RX 利用或门组合为一个串口中断。

在第 3 章已经对 2 个外部中断做了介绍,这里再对定时/计数器中断和串口中断以及所有中断的一些共同问题做进一步概括介绍。

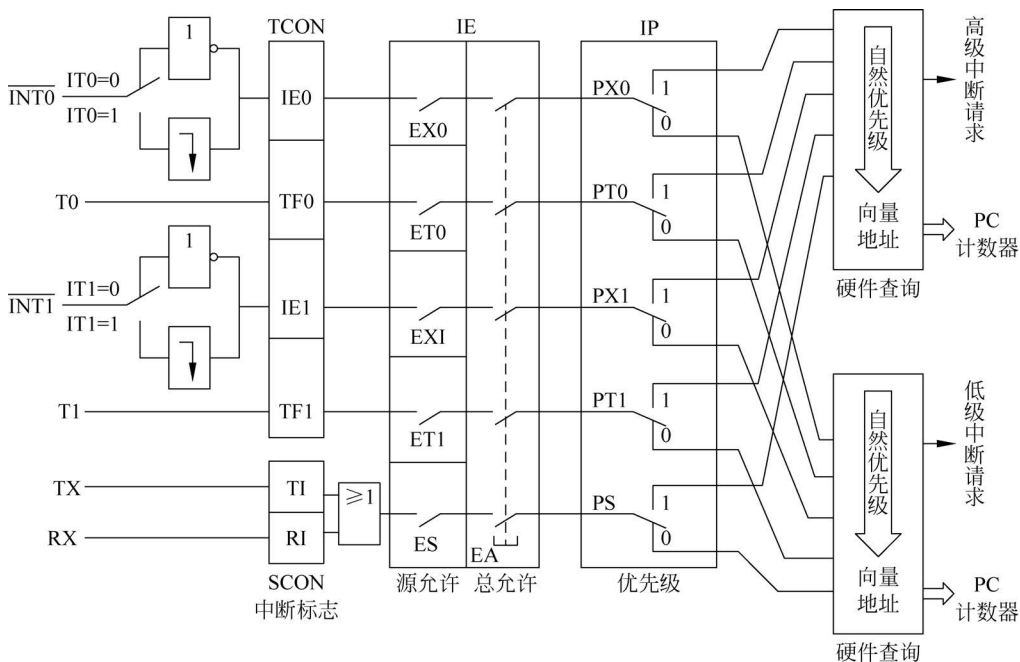


图 5-31 51 单片机的中断系统

1. 中断请求与中断响应

在图 3-23 所示 **IE 寄存器** 中,除了 EX0 和 EX1 这两位用于对 2 个外部中断进行开放和关闭以外,另外的 ET0 和 ET1 位用于开放和屏蔽 2 个定时/计数器中断(对 52 子系列的单片机,还有一个定时/计数器 T2 中断,对应 IE 寄存器中的 ET2 位),ES 位用于开放和屏蔽串口中断。

与外部中断一样,在程序中,如果希望采用中断方式处理定时/计数器的计数溢出和串口数据的收发,必须在中断请求到来前用指令 SETB 使 IE 中的相应位置位,从而开放定时/计数器和串口中断。

一旦开中断,当相应的中断请求到来时,51 单片机即可响应中断,转到相应的中断服务程序。对于定时/计数器 T0、T1 和 T2,其中断服务程序的入口地址分别在 ROM 的 000BH、001BH 和 002BH 单元;对于串口中断,其中断服务程序的入口地址在 ROM 的 0023H 单元(参见表 3-1)。

需要注意的是,51 单片机响应边沿触发的外部中断和定时/计数器中断后,相应的中断请求标志位 IE0、IE1 或 TF0、TF1 将会自动复位。但是,对于电平触发的外部中断和串口中断标志位 RI 和 TI,响应后不能自动复位,必须在中断服务程序中用专门的 CLR 指令将其清零。

2. 中断优先级

不管是外部中断还是内部中断,中断请求到来的时刻是随机的。这就意味着,CPU 正在处理某个中断请求时,可能又有一个新的中断请求到来;还可能在某个时刻,同时有多个中断请求到来。但是,CPU 在任何一个时刻只能响应并处理一个中断请求。这就涉及 CPU 对各中断请求的响应顺序问题,这就是 **中断优先级**。

在 51 单片机中,可以将各中断源的中断优先级规则归纳为下面 3 条:

- (1) CPU 同时接收到几个中断请求时,首先响应优先级最高的中断请求;
- (2) 正在进行的中断过程不能被新的同级或低优先级的中断请求所中断,一直到该中断服务程序结束,返回了主程序且执行了主程序中的一条指令后,CPU 才能响应新的中断请求;
- (3) 新来的高优先级中断请求能够打断正在进行的低优先级中断服务,从而实现两级中断嵌套。

所谓 **中断嵌套**,就是 51 单片机正在执行低优先级的中断服务程序时,可被高优先级中断请求所中断,待高优先级中断处理完毕后,再返回低优先级中断服务程序。中断嵌套的过程如图 5-32 所示。

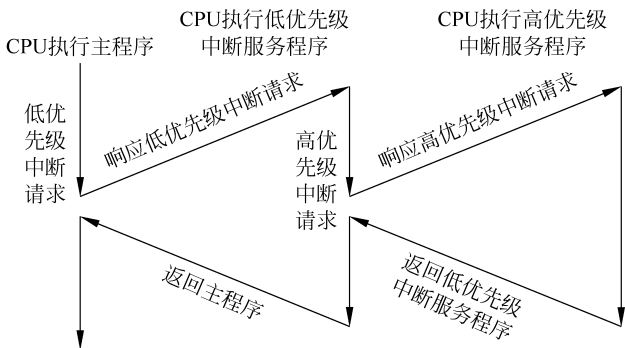


图 5-32 中断嵌套

51 单片机的中断系统中有两个中断优先级组，每个中断请求源所属的优先级组可由软件设置为高优先级或低优先级组，具体是通过图 5-33 所示中的中断优先级 (Interrupt Priority) IP 寄存器实现的。

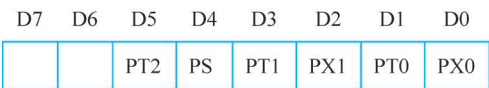


图 5-33 IP 寄存器各位的含义

IP 寄存器的字节地址为 0B8H，可以位寻址。IP 中各位的含义如图 5-33 所示。其中，PX0 和 PX1 分别为外部中断 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$ 的优先级控制位；PT0 和 PT1 分别为定时/计数器 T0 和 T1 的中断优先级控制位；PS 为串口中断的优先级控制位；PT2 为定时/计数器 T2 的中断优先级控制位，只用于 52 子系列。当某位为 1 时，设置相应的中断源为高优先级；否则为低优先级。

除了利用上述 IP 进行优先级设置外，位于同一级的各中断源还有一个默认的优先级顺序，该顺序正好与各中断源中断服务程序入口地址在 ROM 中的存放顺序一致，从高级到低级依次为外部中断 0、定时/计数器 T0 中断、外部中断 1、定时/计数器 T1 中断、串口中断和定时/计数器 T2 中断。

51 单片机复位时，IP 的初始值为 00H，因此所有的中断源都属于低优先级组。在程序中，可以根据实际系统的需要利用指令随意设置各中断源的优先级。

例如，如下指令：

```
MOV IP, #00010100B
```

将 PS 和 PX1 设为 1，其余位设为 0，则外部中断 1、串口中断属于高优先级组，外部中断 0、定时/计数器 T0 中断、定时/计数器 T1 中断和定时/计数器 T2 中断属于低优先级组，再根据默认的优先级顺序，各中断源的优先级顺序从高到低依次为外部中断 1、串行口中断、外部中断 0、定时/计数器 T0 中断、定时/计数器 T1 中断和定时/计数器 T2 中断。

下面通过两个案例介绍定时/计数器中断和串口中断的用法。

动手实践 5-6：定时/计数器中断的应用

将动手实践 5-1 改为用中断方式实现，可以编写如下程序（参见文件 [p5_6.asm](#)）：

```
; 定时/计数器中断的应用
ORG 0000H
AJMP MAIN
ORG 000BH
AJMP T0D
; =====
; 主程序
```



微课视频

```

; =====
ORG    0100H
MAIN:  MOV    TMOD, #00000010B    ; 设置 T0 工作方式
      MOV    TH0, #156            ; 设置计数初值
      MOV    TL0, #156
      SETB   ET0                  ; 开定时/计数器 T0 中断
      SETB   EA
      SETB   TR0                  ; 启动定时
      SJMP   $                    ; 等待定时到
; =====
; 定时/计数器 T0 中断服务程序
; =====
T0D:   CPL    P1.0                ; P1.0 口取反
      RETI                      ; 中断返回
      END

```

与程序 p5_1.asm 采用查询方式相比,在上述采用中断方式的程序中,主要的改动有如下几点:

- (1) 在主程序中启动定时之前需要将 ET0 和 EA 置位,开中断。
- (2) 在 ROM 的 000BH 单元中存放一条 AJMP T0D 指令,当定时到时,51 单片机执行该条指令,转到 T0 的中断服务程序 T0D。
- (3) 在中断服务程序中,原来检测和清除 TF0 的操作都不再需要。当定时到时,定时计数器内部电路将 TF0 置位。由于主程序中已经开了 T0 中断,因此将立即向 51 单片机发出中断请求。这就意味着,一旦程序执行进入中断服务程序,TF0 肯定已经被置位,所以不再需要检测。另外,51 单片机响应此中断后,内部电路会自动将 TF0 重新复位,因此也不需要再安排专门的 CLR 指令将其清零。



微课视频

动手实践 5-7: 串口中断的应用——双工通信

51 单片机的串口本身可以实现双工通信。在工作过程中,两台 51 单片机都可以通过执行程序同时实现数据的发送和接收。为了体会点-点双工通信的控制程序及中断方式下的数据传输过程,重新绘制原理图如图 5-34 所示(参见文件 [ex5_7.pdsprj](#))。

在图 5-34 中,为了实现双工通信,将两台 51 单片机的 RxD 和 TxD 引脚相互交叉连接。两台 51 单片机的 P1 口和 P2 口都分别连接了 8 个 LED 和 8 个开关。要求实现的功能是:每台 51 单片机将开关的状态送到对方单片机连接的 LED 显示。例如,用与 U1 单片机连接的 DSW1 的状态控制与 U2 单片机连接的 LED U3 的显示。注意设置两台 51 单片机的时钟频率都为 12 MHz。

由于两台 51 单片机实现的功能相同,并且都是通过相同的并口与开关和 LED 相连接,因此两台 51 单片机的控制程序完全一样,完整的代码如下(参见文件 [p5_7.asm](#)):



图 5-34

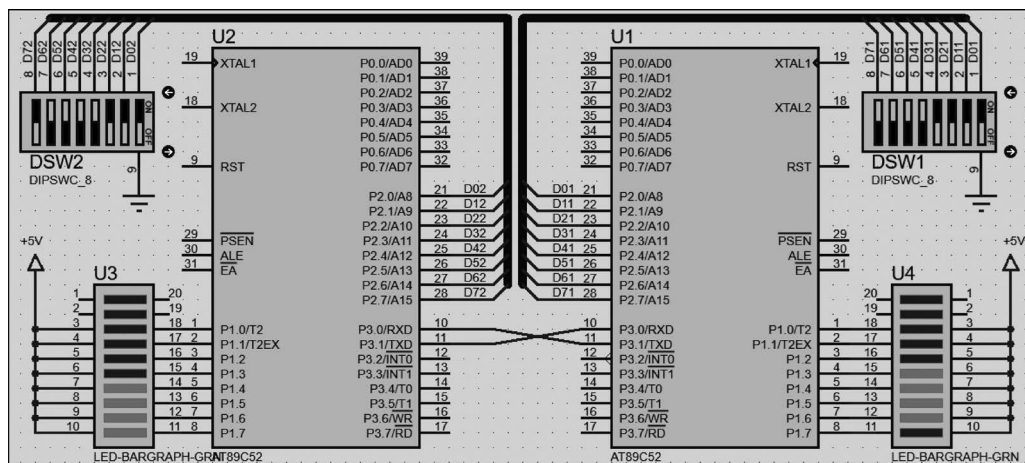


图 5-34 两台 51 单片机之间的点-点双工通信连接原理图

；点-点双机双工通信(中断方式)

LED DATA P1

SWT DATA P2

ORG 0000H

AJMP MAIN

ORG 0023H

AJMP INS

；串口中断服务程序入口

；=====

；主程序

ORG 0100H

MAIN: MOV SP, #60H

；主程序

MOV SCON, #01010000B

；单片机工作方式 1, 允许接收

MOV TMOD, #00100000B

；设置定时/计数器 T1 工作方式 2 定时功能

MOV TL1, #0E6H

；设置 T1 的计数初值以确定波特率

MOV TH1, #0E6H

SETB TR1

；启动 T1 定时

SETB EA

SETB ES

；开串口中断

MOV SWT, #0FFH

MOV A, SWT

MOV SBUF, A

；发送第一个数据

SJMP \$

；循环等待串口中断

；=====

；串口中断服务程序

INS: CLR EA

；关串口中断

JB RI, REC

；是接收串口中断, 则跳转

CLR TI

；是发送串口中断, 则清除 TI

MOV A, SWT

MOV SBUF, A

；发送数据

SJMP EXT

```
REC:  CLR    RI                ; 清除接收串口中断请求标志位
      MOV    A, SBUF           ; 从串口读接收到的数据
      CPL    A                ; 由 P1 口输出控制 LED
      MOV    LED, A
EXT:  SETB   EA                ; 开串口中断
      RETI                    ; 串口中断返回
      END
```

在上述程序中,两台 51 单片机执行完全相同的程序流程,不断读取并发送所连接的关键数据,并等待串口中断。一旦有中断请求到来,则在中断服务程序中首先判断是发送还是接收中断。如果是发送中断,则先清除 TI,然后读取开关数据送到串口的 SBUF 并通过串口发送出去;如果是接收中断,则先清除 RI,然后从串口 SBUF 中读取接收到的数据并通过 P1 口输出控制 LED。

在原理图中单击两个开关,可以看到另一台 51 单片机上连接的 LED 的亮灭状态变化。这就意味着,两台 51 单片机在任何时刻都既可以发送数据,也可以接收数据,从而实现全双工通信。

为了观察数据传送过程,可以在原理图中添加一个示波器,并将单片机 U1 的 RxD 和 TxD 引脚分别接到示波器的 A 和 B 通道。单击两个开关设置需要传输的数据,在运行过程中或暂停运行状态观察示波器的波形如图 5-35 所示。

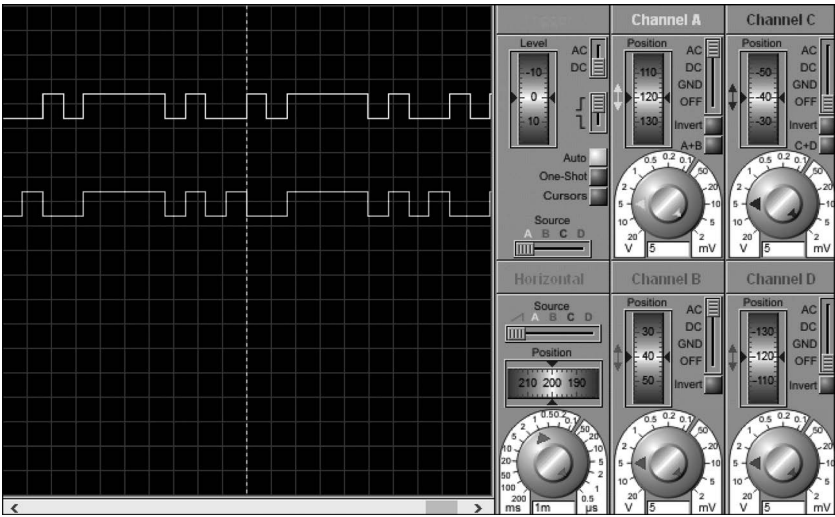


图 5-35 串口通信中 RxD 和 TxD 信号的波形

图 5-35 中的两路脉冲分别是单片机 U1 和 U2 发送的开关数据,读取顺序是从右往左对应数据的高位到低位。此外,在波形上容易读取传输每位数据的时间近似为 0.8 ms,其倒数近似为 1200 b/s,这就是程序中设置的波特率。

注意根据各开关的状态读出波形上传输一帧的最高位和最低位。在波形上可以清楚地看到,在每帧数据的开始(即 8 位开关数据最低位的左边)有一位低电平,这就是起始位。每

帧数据传输的最后一位(即 8 位开关数据的最高位)右边,有一位高电平,这就是停止位。由于程序不断循环收发,所以每次传输一帧共 10 位数据,就这样不断地重复进行数据的发送和接收。

5.4 牛气冲天——实战进阶

在熟悉了 51 单片机定时/计数器和串口的基本概念及其典型应用的基础上,本节继续介绍应用过程中的一些特殊问题和高级技术。

5.4.1 定时/计数器的级联

不管设置定时/计数器工作在哪一种工作方式,由于计数位数有限,计数脉冲的周期或频率固定,因此都只能实现有限长度的定时。表 5-3 给出了在 51 单片机两种典型的时钟脉冲频率下,3 种工作方式能够获得的最长定时时间。

表 5-3 定时/计数器的最长定时时间

时 钟 频 率	方 式 0	方 式 1	方 式 2
6 MHz	16.384 ms	131.072 ms	0.512 ms
12 MHz	8.912 ms	65.536 ms	0.256 ms

在实际系统中,为了获得更长的定时,可以有两种基本的方法,即定时/计数器和程序控制配合、多个定时/计数器级联使用。下面通过具体案例体会后一种方式实现的基本原理。

动手实践 5-8：长定时的实现

在计数位数有限的情况下,定时时间太短的一个主要原因在于定时/计数器进行定时所需的标准计数脉冲频率太高。为此可以考虑将一个定时/计数器用作分频器,将标准计数脉冲进行分频,以增大其周期。然后将分频后得到频率较低的信号再作为另一个定时/计数器的计数脉冲,即可获得更长的定时。

在本案例中,利用定时/计数器 T0 和 T1 级联实现 0.5 s 的定时。其中 T0 工作在定时方式,用于在 P1.1 引脚产生周期为 20 ms 的方波脉冲;P1.1 引脚输出的方波脉冲再作为 T1 计数器的计数脉冲,控制 P1.0 引脚输出 1 Hz 的周期方波脉冲,电路连接如图 5-36 所示(参见文件 [ex5_8.pdsprj](#))。为便于观察,将 P1.0 引脚输出周期为 20 ms 的脉冲和 P1.1 引脚输出周期为 1 s 的脉冲同时送入示波器的 A 和 B 通道。

1. 定时/计数器级联——查询方式

这里设置定时/计数器 T0 工作在方式 1 定时方式,每定时 10 ms 将 P1.0 引脚的输出信号取反,从而得到周期为 20 ms 的计数脉冲。设置定时/计数器 T1 工作在方式 2,则可设置其计数初值为 $256 - 0.5\text{ s} / 20\text{ ms} = 231$ 。每定时 0.5 s,计数溢出,将 P1.1 引脚取反,即可输出周期为 1 s 的方波脉冲。程序代码如下(参见文件 [p5_8_1.asm](#)):


```

        ORG 000BH                ; T0 中断服务程序入口
        AJMP TODEL
; =====
; 主程序
; =====
        ORG 0100H
MAIN:   MOV  TMOD, #01100001B    ; 设置 T0 方式 1 定时方式, T1 方式 2 计数方式
        MOV  TH0, #0D8H          ; 设置 T0 计数初值
        MOV  TL0, #0F0H
        MOV  TH1, #231           ; 设置 T1 计数初值
        MOV  TL1, #231
        SETB ETO                 ; 开 T0 中断
        SETB EA
        SETB TR0                 ; 启动 T0 定时 10 ms
        SETB TR1                 ; 启动 T1 计数
LP:     JNB  TF1, $               ; 等待 T1 计数溢出 (0.5 s 定时到)
        CLR  TF1
        CPL  SIG2                ; P1.1 引脚取反
        SJMP LP
; =====
; T0 中断服务程序
; =====
TODEL:  CPL  SIG1                ; P1.0 (T1 计数脉冲) 取反
        MOV  TH0, #0D8H          ; 重装 T0 计数初值
        MOV  TL0, #0F0H
        RETI
        END

```

在上述程序的主程序中,仍然用同样的方法设置两个定时/计数器的工作方式和计数初值,并启动两个定时/计数器的计数。之后,主程序通过查询 TF1 的状态,等待定时/计数器 T1 计数溢出。一旦计数溢出,则将 P1.1 引脚输出取反,再返回不断循环。

在本案例中,只用了定时/计数器 T0 中断,因此在程序一开始,将 AJMP TODEL 指令存放到 000BH 单元。一旦 T0 定时到,则 CPU 响应中断后执行该指令,从而跳转到标号为 TODEL 的指令。从这条指令开始就是定时/计数器 T0 的中断服务程序。在中断服务程序中,将 P1.0 引脚输出取反,同时重装定时/计数器 T0 的计数初值,再执行 RETI 指令返回。

在 CPU 执行主程序循环的过程中,定时/计数器 T0 在不断计数。一旦计数溢出,立即暂停主程序中循环的执行,跳转到定时/计数器 T0 的中断服务程序。因此,执行 RETI 指令后,将返回到主程序的循环中,继续查询 TF1 的状态并等待定时/计数器 T1 计数溢出。

需要注意的是,如果是采用中断方式检测计数是否溢出,CPU 在响应中断后将由内部硬件电路自动将 TF0 清零,以便下一次计数溢出时重新将其置位。因此,在 T0 的中断服务程序中,不需要另外安排清除 TF0 的操作。

上述两个程序运行后,在示波器上观察 P1.0 和 P1.1 引脚输出的两路脉冲如图 5-37 所示。

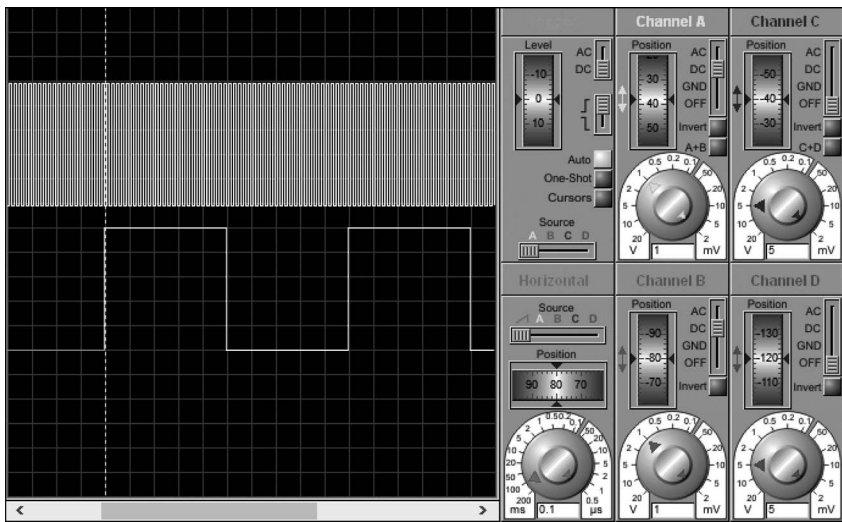


图 5-37 定时/计数器产生的 50 Hz 和 1 Hz 周期方波脉冲

5.4.2 串口方式 2 和方式 3 及其应用

在 SCON 中,当设置 SM1 和 SM0 为 10 和 11 时,分别指定串口的工作方式为方式 2 和方式 3。在这两种方式下,串口仍然用 TXD 和 RxD 引脚发送和接收数据。方式 2 和方式 3 与方式 1 的区别在于发送数据帧的格式不同;而方式 2 和 3 之间的主要区别在于传输的波特率不同。方式 2 的波特率固定为振荡器频率的 1/32 或 1/64;而方式 3 的波特率由定时/计数器 T1 的计数初值决定。

方式 2 和方式 3 数据帧的格式如图 5-38 所示。由此可见,这两种方式除了发送 8 位数据位以外,另外还需要发送一位 TB8,合起来共 9 位构成数据帧,因此称为 **9 位异步通信方式**。此外,与方式 1 一样需要前后分别附加一位起始位和一位停止位,合起来共需传送 11 位数据。因此有些地方又称为 **11 位异步通信方式**。

起始位	D0	...	D7	RB8/TB8	停止位
0	8 位数据位			校验位	1

图 5-38 串口方式 2 和方式 3 数据帧的格式

对发送方,数据帧的最后一位为 TB8。在发送前,首先用位操作指令设置 TB8 为 0 或 1,然后将待发送的 8 位数据存入 SBUF 即可启动串行数据的发送。发送数据结束后将 TI 置位。

对接收方,接收到的 8 位数据存入 SBUF,等待 CPU 读取,而第 9 位数据自动存入 SCON 寄存器的 RB8 位。这 9 位数据接收结束后,串口内部电路将 RI 置位。

1. 串行通信中的错误校验

在串行通信过程(特别是长距离通信)中,数据传输的过程会受到各种干扰(传输线路引

入的噪声)的影响,造成数据传输错误。因此,在接收端一般都需要对数据传输的正确与否进行校验,以保证传输数据准确无误。常用的数据校验方法有奇偶校验、循环冗余码校验等。

(1) 奇偶校验。

串行数据发送时,在数据位后面尾随 1 位奇偶校验位(1 或 0)。当约定为**奇校验**时,数据位和校验位中 1 码的个数应为奇数;当约定为**偶校验**时,1 码的个数应为偶数。

在通信过程中,收发双方需要事先约定采用奇校验还是偶校验。在发送时,发送方根据待发送数据位中 1 码个数的奇偶正确设置相应的校验位,以保证发送的数据位和校验位中 1 码总的个数为奇数或偶数。

按照此约定,如果在传输过程中由于干扰使得数据位中的某一位发生错误,即 1 码变为 0 码或 0 码变为 1 码,则接收到的数据位和校验位中 1 码个数的奇偶将不满足约定。此时,由于接收的数据是错误的,CPU 不应该也不需要读取,接收方可以要求发送方重新发送。

(2) 代码和校验与循环冗余码校验。

代码和校验是发送方将所发数据块求和或各字节异或,产生 1 字节的校验字符(校验和)附加到数据块末尾。接收方接收数据时同时对数据块(除校验字节)求和或各字节异或,将所得结果与发送方的“校验和”进行比较,如果相符,则无差错;否则即认为在传输过程中出现了差错。

循环冗余码校验纠错能力强,容易实现。该校验是通过某种数学运算实现有效信息与校验位之间的循环校验,常用于对磁盘信息的传输、存储区的完整性校验等。是目前应用最广的检错编码方式之一,广泛用于同步通信中。

2. 多机串行通信

在单片机集散式分布系统中,往往采用一台**主机**(Master)和多台**从机**(Slave)。主机可以与系统中任何一台从机进行串行通信(发送或接收数据),一般各从机之间不直接进行数据传输。对这样的**主从式多机通信**系统,由于所有的从机都通过同样的 RxD 和 TxD 两根线与主机进行数据传输,因此工作过程中一个关键的问题是主机如何找到当前需要进行通信的从机,并且禁止其他从机接收主机发出的数据。

一种最简单的做法是,事先为每个从机指定一个唯一的、固定不变的编码代号,称为**从机地址**。传输数据之前,主机先向系统中所有从机发送当前需要通信的从机地址。所有从机都接收到该地址,但只有事先规定的地址与该发送地址相同的从机,才向主机发送一个**应答信号**。主机接收到该应答信号以后,才开始与该从机进行数据传输。

由此可见,相对于方式 1 的点-点通信,多机通信系统中的每次数据传送过程都必须包括如下几个操作步骤:主机向所有从机发送地址;被寻址的从机向主机发送应答信号;传输数据。其中前两个阶段称为**联络过程**,在此根据需要还可能执行其他的操作;第三个阶段的传输数据可能是主机发往从机或者从机发往主机。

为了实现上述多机主从式通信,51 单片机的串口必须工作在方式 2 或方式 3,并且要用到 SCON 寄存器中的 SM2 和 TB8/RB8 位。

(1) SM2：多机通信控制位。对串口工作方式 0 和方式 1，一般将该位设为 0。当串口以方式 2 或方式 3 接收时，如果设置 SM2=0，则与方式 1 一样实现普通的点-点通信，接收到的 8 位数据都将存入 SBUF。如果设置 SM2=1，则只有当接收到的第 9 位数据 RB8 为 1 时，才将接收到的前 8 位数据送入 SBUF。如果接收到 RB8 位为 0，则将接收到的前 8 位数据丢弃，相当于不接收数据。

(2) TB8/RB8：在方式 2、方式 3 的点-点通信中，一般用作奇偶校验位。在多机通信中，一般用该位区分是数据帧(0 码)还是地址帧(1 码)。在方式 1 下，该位无用，可随意设置为 1 或 0。

由此可知，在从机的 SM2 位都设为 1 时，利用 RB8/TB8 位可以控制从机是否接收数据，从而实现从机的选择应答。在应答过程中，主机向所有从机发送一个地址帧，其中 TB8=1，因此所有从机都将接收到该地址。

在所有从机接收到地址帧后，通过执行自身的程序判断接收到的地址是否与自身事先设置的地址是否相同。如果相同，表示主机正在寻址该从机，因此从机将其 SM2 位复位，而没有被寻址的从机其 SM2 位保持为 1。

由于本寻址的从机 SM2 位已经复位，该从机能够与主机之间进行普通的点-点通信，主机发送的数据能够被该从机收到。由于没有被寻址的从机其 SM2 位保持为 1，只要主机后续发送的数据帧中 TB8=0，则主机发送的数据这些从机将不会接收到。

至此，可以将多机主从式通信过程所需的程序操作步骤总结如下。

- (1) 如果系统中只有 1 个主机，则将主机的 SM2 位复位，所有从机的 SM2 位置位，以便所有从机都能接收到主机发来的从机地址。
- (2) 主机发送从机地址，其中设置 TB8=1。
- (3) 所有从机接收到地址帧后，将其与本从机地址比较，并根据比较结果设置 SM2 位。
- (4) 主机发送 TB8=0 的数据帧，送往被寻址到的从机。
- (5) 通信结束后，从机将 SM2 重新置位。

动手实践 5-9：串行通信中的奇偶校验

本案例的原理图与[动手实践 5-7](#) 中的图 5-36 所示相同，为方便，这里将其重新保存为 [ex5_9.pdsprj](#)，相应的程序如下(参见文件 [p5_9.asm](#))：

```

; 点-点通信中的偶校验
LED    DATA    P1
SWT     DATA    P2
        ORG      0000H
        LJMP     MAIN
        ORG      0023H                ; 串口中断服务程序入口
        LJMP     INS
        ORG      0100H
MAIN:   MOV      SP, #60H
        MOV      SCON, #11010000B    ; 串口初始化, 方式 3, 允许接收 (REN = 1)

```

```

MOV    TMOD, # 20H
MOV    TL1, # 0E6H
MOV    TH1, # 0E6H
SETB   TR1
SETB   EA
SETB   ES
MOV    SWT, # 0FFH
MOV    A, SWT                      ; A 中 1 码个数为偶数,则 P = 0;1 码个数为奇数,则 P = 1
MOV    C, P
MOV    TB8, C                      ; 设置 TB8 = P,使总的 1 码个数为偶数(偶校验)
MOV    SBUF, A                     ; 发送第一个数据
SJMP   $                          ; 循环等待中断
; =====
; 串口中断服务程序
; =====
INS:   CLR    EA
       JB     RI, REC
       CLR    TI                      ; 是发送中断,则清除 TI
       MOV    A, SWT
       MOV    C, P                    ; 偶校验
       MOV    TB8, C
       MOV    SBUF, A                 ; 发送数据
       SJMP   EXT
REC:   CLR    RI
       MOV    A, SBUF                 ; 接收数据,并根据 A 中 1 码的个数奇偶设置 P 标志位
                                   ; 接收的第 9 位数据(即发送的第 9 位 TB8)自动存入 RB8
       ;XRL A, # 01H                 ; 模拟接收到的 8 位数据最低位错误
       JB     P, GO                   ; 检测 P = RB8?
       JNB    RB8, DONE
       SJMP   EXT                     ; 否,偶校验错误,返回
GO:    JB     RB8, DONE
       SJMP   EXT
DONE:  MOV    LED, A                  ; 是,偶校验正确,输出点亮 LED
EXT:   SETB   EA
       RETI
END

```

启动运行后,单击各开关,开关数据都能正常发送到对方 51 单片机,并在 LED 上正确显示。为了模拟奇偶校验,在中断服务程序中添加 XRL A, # 01H(去掉上述程序中该条指令前面的分号即可)。现在重新运行程序可以发现: 不管如何单击开关,两个 51 单片机上连接的 LED 都不会点亮,意味着数据没有被正确接收。

上述程序中采用的是偶校验。发送方根据发送开关数据中 1 码个数的奇偶设置 SCON 中的 TB8 位,对应的代码如下:

```

MOV    A, P2
MOV    C, P
MOV    TB8, C

```

在上述 3 条指令中,首先从 P2 口读取开关数据并存入 A,该操作将影响 PSW 中的 P 标志位。当读取的开关数据(即累加器 A)中有偶数个 1 时,P=0; 否则 P=1。后面 2 条

指令将 P 标志位中的 0 或 1 存入 TB8。因此,如果有偶数个开关打在 OFF 位置,表示 A 中发送的 8 位数据中有偶数个 1,则 TB8=0;当 A 中有奇数个 1 时,TB8=1。注意在原理图中,开关打在 ON 位置为 0 码;开关打在 OFF 位置为 1 码。

在接收端,将接收到的前 8 位数据存入 SBUF,第 9 位存入 SCON 中的 RB8 位。在中断服务程序中,如果检测到是接收中断,则首先清除 RI 标志,并将接收到的前 8 位数据读入累加器 A。该操作将影响 P 标志位。

程序中接下来利用 JB 指令检测 P 标志位。如果 P=1,则跳转到标号为 GO 的指令,继续检测 RB8 是否也为 1。如果 P=RB8=1,则将读取得到的 8 位数据由 P1 口输出以点亮 LED;否则 A 中的数据不输出,而是直接中断返回。同理,如果检测到 P=RB8=0,接收到的数据也将由 P1 口输出。

如果传输过程中没有错误,则接收到的数据位和校验位 RB8 一定满足上述两种情况,也就是数据位和校验位中 1 码的个数一定为偶数。如果传输过程中有一位数据位发生错误,则 P 和 RB8 一定不相等,因此接收到的数据都不会由 P1 口输出到 LED,此时 LED 将保持原来的状态,也就相当于 CPU 没有接收错误的数据。

在上述程序中用 XRL 指令将接收到的数据最低位取反,用于模拟传输过程中数据的最低位发生错误。此时,通过奇偶校验发现该错误,则不将数据由 P1 口控制 LED。

动手实践 5-10: 多机主从式通信

本案例利用如图 5-39 所示原理图(参见文件 [ex5_10.pdsprj](#))演示一个简单的多机主从式通信系统。其中 U1 是主机,U2~U4 是 3 个从机,假设其地址分别为 0~2。系统要实现的功能是:将主机内部 RAM 中存放的若干字节数据送往指定的某个从机,并保存到从机内部 RAM 中指定的单元。

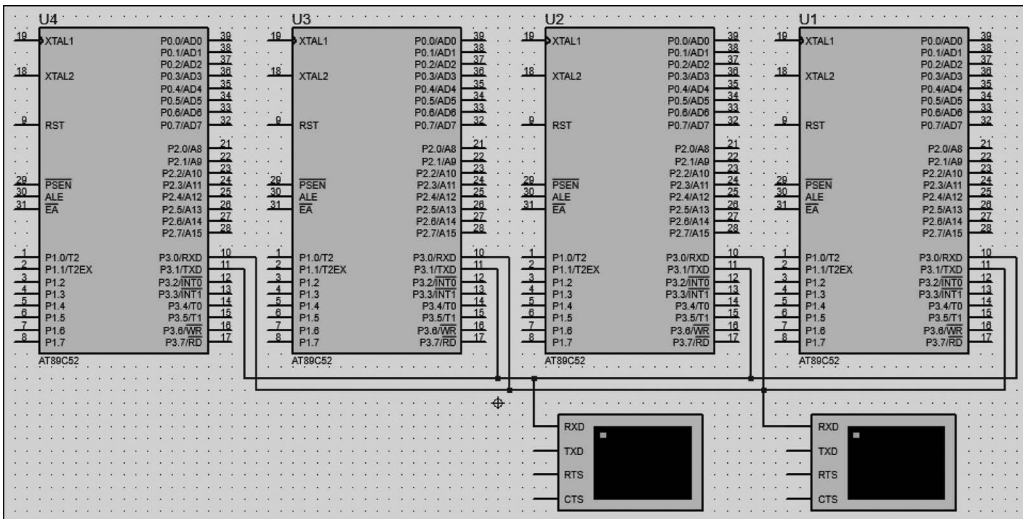


图 5-39 简单的多机主从式通信系统

注意本案例需要在 Keil C51 中创建两个工程,后面将这两个工程代码文件分别加载到主机 U1 和从机 U4 中。在两个工程中分别添加如下主机和从机的源程序(参见文件 [p5_10m.asm](#) 和 [p5_10s.asm](#))。

```

; 多机通信——主机
SADD EQU 1 ; 从机地址
RT EQU 1 ; 命令,1:主机发送数据;0:主机接收数据
BUFT EQU 40H ; 主机发送数据缓冲区
BUFR EQU 50H ; 主机接收数据缓冲区
ORG 0000H
AJMP MAIN
ORG 0100H
MAIN: MOV 40H, #10 ; 初始化设置待发送数据
      MOV 41H, #20
      MOV 42H, #30
      MOV 43H, #40
      MOV SCON, #11010000B ; 串口初始化:方式3,允许接收,SM2=0
      MOV TMOD, #20H ; T1 初始化以确定波特率
      MOV TH1, #0E6H
      MOV TL1, #0E6H
      SETB TR1
LP: SETB TB8 ; TB8 置位
    MOV A, #SADD ; 发送地址帧
    MOV SBUF, A
    JNB RI, $ ; 等待从机应答
    CLR RI
    MOV A, SBUF
    XRL A, #SADD ; 判断应答地址是否相等
    JZ MIO1 ; 相等,则继续
    SJMP LP ; 不相等,则重新联络
MIO1: CLR TB8 ; 地址相等,向从机发送命令(TB8=0表示命令)
      MOV A, #RT
      CLR TI
      MOV SBUF, A ; 发送命令
      JNB TI, $
      JZ RX ; 命令 RT=0?
      ACALL SEND_DATA ; 否,主机发送数据
      SJMP LP
RX: CLR RI
    ACALL REC_DATA ; 是,主机接收数据
    SJMP LP
SEND_DATA: ; 主机发送子程序
          MOV R7, #4 ; 否则,发送数据
          MOV R0, #BUFT
LPT: CLR TI
      MOV SBUF, @R0
      JNB TI, $
      INC R0
      DJNZ R7, LPT
      RET

```

```

REC_DATA:                                ; 主机接收子程序
MOV     R7, #4                            ; 接收数据
MOV     R1, #BUFR
LPR:    JNB     RI, $
        CLR     RI
        MOV     A, SBUF
        MOV     @R1, A
        INC     R1
        DJNZ    R7, LPR
        RET
        END

; =====
; 多机通信——从机
SADD    EQU     1                        ; 本机地址
BUFT    EQU     50H                      ; 从机发送数据缓冲区
BUFR    EQU     40H                      ; 从机接收数据缓冲区
ORG     0000H
AJMP    MAIN
ORG     0023H
AJMP    SIO                                ; 串口中断入口
ORG     0100H
MAIN:    MOV     SP, #20H                  ; 设置堆栈指针
        MOV     50H, #50                  ; 从机发送数据初始化
        MOV     51H, #60
        MOV     52H, #70
        MOV     53H, #80
        MOV     SCON, #11110000B          ; 置串口方式 3, 允许接收, SM2 = 1
        MOV     TMOD, #20H                ; 初始化 T1, 以确定波特率
        MOV     TH1, #0E6H
        MOV     TL1, #0E6H
        SETB    TR1
        SETB    EA
        SETB    ES                        ; 开串口中断
        SJMP    $
SIO:     CLR     RI                        ; 串口中断服务程序
        MOV     A, SBUF                    ; 接收地址
        XRL     A, #SADD
        JNZ     RETN                       ; 不是本机地址, 则跳转
        MOV     SBUF, #SADD                ; 否则, 向主机回送本机地址
        CLR     SM2                       ; 清除 SM2, 准备传输命令和数据
        JNB     RI, $                      ; 接收主机命令
        CLR     RI
        JNB     RB8, SIO1                  ; 是命令帧, 转 SIO1
        SJMP    RETN                       ; 否则, 将 SM2 置位并返回
SIO1:    MOV     A, SBUF                    ; 接收并分析命令
        JZ      TX                         ; 命令 0, 则转发送
RX:      MOV     R0, #BUFR                 ; 否则, 接收数据
        MOV     R7, #4                     ; 数据字节数送 R7
LPRX:    JNB     RI, $                      ; 接收数据存入缓冲区
        CLR     RI
        MOV     @R0, SBUF

```

```
INC    R0
DJNZ   R7,LPRX
SJMP    RETN
Tx:    MOV    R1, #BUFT
      MOV    R7, #4          ; 数据字节数送 R7
LPTX:  CLR    TI
      MOV    SBUF, @R1      ; 发送数据
      JNB    TI, $
      INC    R1
      DJNZ   R7, LPTX
RETN:  SETB    SM2          ; 发送完, 置 SM2 = 1 后返回
      RETI
      END
```

将上述两个工程编译成功后分别加载到主机和从机 U4, 启动运行后, 自动弹出两个虚拟终端窗口, 暂停程序运行后观察主机和从机内部 RAM 分配情况如图 5-40 所示。

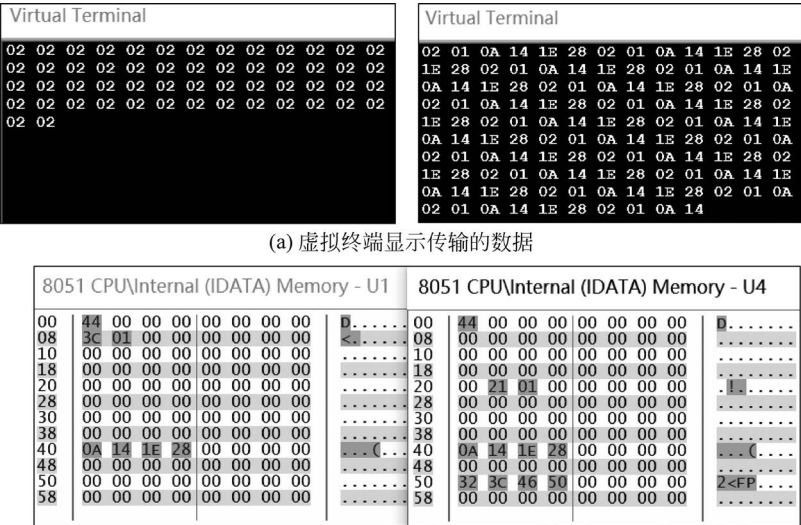


图 5-40 多机通信运行结果

在图 5-40(a)中, 两个虚拟终端分别显示的是主机和从机发送的数据。主机连续不断地循环发送从机地址 02H、数据收发命令 01H 以及内部 RAM 中的 4 字节数据 0AH、14H、1EH 和 28H。对应主机的每次循环, 从机向主机发送 1 字节的应答地址 02H。在图 5-40(b)中, 主机 RAM 中 4 字节数据发送给从机 U4, 并存入内部 RAM 地址从 40H 开始的单元。

本案例演示了一个基本的多机通信过程, 整个过程包括主机和从机 U4 的主程序以及中断服务程序, 其中的主要操作可以用图 5-41 所示流程图进行描述。

需要说明的是, 上述流程图对主机和从机分开绘制, 但在工作过程中, 主机和从机的发送和接收过程相互联系的, 因此接下来按照完整的传输过程, 将主机和从机在每个阶段的操作合并在一起介绍。

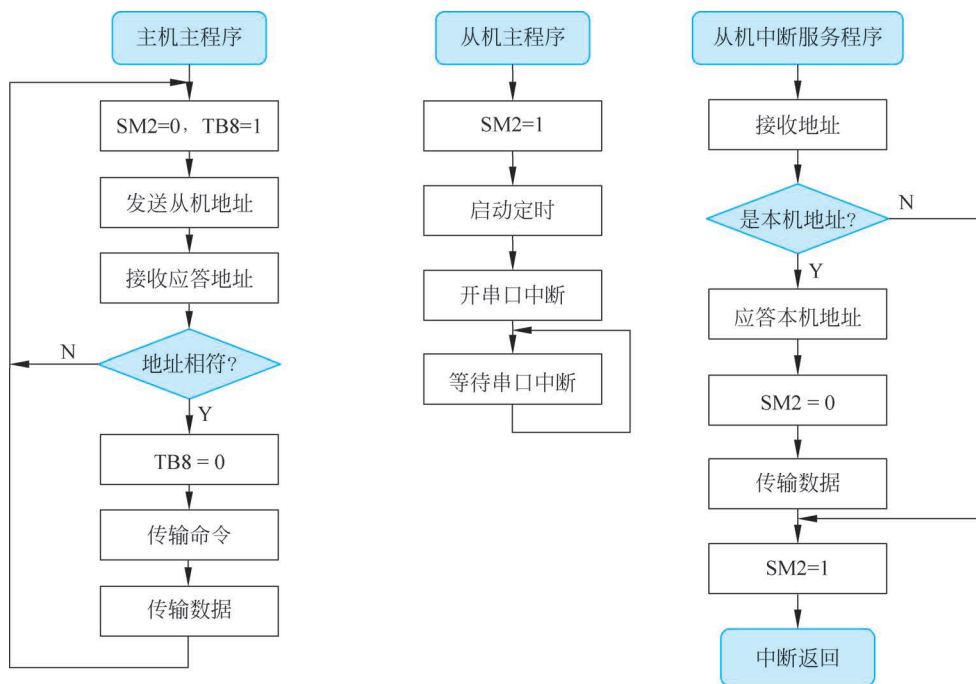


图 5-41 多机通信的基本过程

(1) 串口初始化。

在主机的主程序中，首先设置串口的方式控制字及定时/计数器 T1 的计数初值并启动定时，其中特别注意设置 $SM2=0$ 。

在从机的主程序中，初始化操作也包括上述两项，但其中设置 $SM2=1$ 。此外，定时/计数器 T1 的工作方式和计数初值必须与主机完全相同，以保证主、从机具有相同的波特率。

在本案例中，从机收发数据采用中断方式，因此在从机的初始化程序中还需要开串口中断。

(2) 地址联络过程。

做了上述初始化设置后，主机向所有从机发送当前需要通信的从机地址，并等待被寻址的从机回送应答地址（从机的本机地址）。

由于设置所有从机的 $SM2=1$ ，并且主机发送地址时 $TB8=1$ ，因此所有从机都将接收到主机送来的从机地址。各从机在接收到地址后，立即响应串口中断并进入各自的中断服务程序。

在中断服务程序中，各从机的 CPU 从 SBUF 中读出地址，并与事先规定的本机地址进行比较，比较过程用 XRL 指令实现。如果接收到的地址与从机程序中定义的本机地址 SADD 相等，则该从机将本机地址通过串口发送回主机，并将该从机的 $SM2$ 清零，准备传输

命令和数据。如果两个地址不相等,说明该从机不是当前主机需要寻址的从机,则不向主机回送本机地址,也不再参与后续的命令和数据传输,而直接中断返回。

(3) 命令的发送和接收。

主机接收到正确的应答地址后,意味着找到被寻址的从机,开始继续传输命令和数据。此时将 TB8 清零。

对从机来说,只有被寻址的从机才将其 SM2 清零,之后即可像方式 1 一样参与后续命令和数据的传输。也就是不管接收到 RB8 位是 1 码还是 0 码,接收到的所有命令和数据都将存入 SBUF,等待 CPU 读取。对未被寻址的从机,由于 SM2 保持为 1,因此在 TB8=0 时,接收到的 RB8 也为 0,则将接收到的前 8 位数据丢弃不存入 SBUF,也就是不接收后续的命令和数据。

在将 TB8 清零后,主机向从机首先发送命令,该命令设为一个 8 位二进制数据,并在主程序的一开始用 EQU 伪指令定义为一个常量 RT。RT 的值可以设为 0 或 1,分别表示主机接收数据和发送数据。主机将该命令发送到被寻址的从机,之后根据该命令转入数据的收发。

对被寻址的从机来说,主机发送的上述命令用于指示从机发送还是接收数据。显然与主机相反,当接收到命令为 0 或 1 时,表示从机发送或接收数据。

(4) 数据的收发。

在本案例中,主机和从机通过串口发送和接收数据实现的功能是相同的,因此主、从机程序中实现数据的发送与接收操作是类似的。只是在主机程序中,将数据的发送和接收分别定义为一个子程序;而在从机程序中,将数据的发送和接收都放在中断服务程序中。

下面以主机程序中发送数据的代码为例,介绍数据收发的基本操作。

在主机发送子程序 SEND_DATA 中,首先将常数 4(需要发送的数据字节数)存入 R7,将主机发送数据缓冲区的起始地址存入 R0。之后通过循环注意将缓冲区中原来存放的 4 字节数据通过串口发送出去。注意每次发送将 1 字节数据送入 SBUF 之前,必须清除 TI。发送过程中利用 JNB 指令检测 TI 的状态。当检测到 TI=1 时,表示当前字节数据发送完毕,再修改缓冲区地址指针并继续循环发送下 1 字节数据。

(5) 调试方法。

调试时,必须将上述主机和从机程序分别加载到原理图中的主机 U1 和从机 U4。启动运行后即可观察到在主机主程序一开始定义到缓冲区中的 4 字节数据被正确发送到从机 U4,并保存到其内部 RAM 地址为 50H 开始的单元。

如果将主机程序中的 RT 常量值修改为 0,则表示将从机 U4 内部 RAM 中地址从 50H 开始单元的 4 字节数据发送到主机,并保存到主机内部 RAM 中相同地址的单元。重新运行后,主机和从机内部 RAM 单元分配情况如图 5-42 所示。

8051 CPU\Internal (IDATA) Memory - U1										8051 CPU\Internal (IDATA) Memory - U4									
00	00	52	00	00	00	00	00	02	.R....	00	00	52	00	00	00	00	00	02	.R....
08	42	01	00	00	00	00	00	00	B.....	08	00	00	00	00	00	00	00	00	
10	00	00	00	00	00	00	00	00		10	00	00	00	00	00	00	00	00	
18	00	00	00	00	00	00	00	00		18	00	00	00	00	00	00	00	00	
20	00	00	00	00	00	00	00	00		20	00	51	00	00	00	00	00	00	
28	00	00	00	00	00	00	00	00		28	00	00	00	00	00	00	00	00	
30	00	00	00	00	00	00	00	00		30	00	00	00	00	00	00	00	00	
38	00	00	00	00	00	00	00	00		38	00	00	00	00	00	00	00	00	
40	0A	14	1E	28	00	00	00	00		40	00	00	00	00	00	00	00	00	
48	00	00	00	00	00	00	00	00		48	00	00	00	00	00	00	00	00	
50	32	3C	46	50	00	00	00	00	2<FP...	50	32	3C	46	50	00	00	00	00	2<FP...
58	00	00	00	00	00	00	00	00		58	00	00	00	00	00	00	00	00	

图 5-42 从机向主机发送数据

本章小结

51 单片机内部不仅有 CPU 和并口,还有定时/计数器和串口这两个重要的资源。定时/计数器和串口借用 P3 口某些引脚的第二功能,以便与外部电路相连接,以提供所需的信号(例如输入定时/计数器的计数脉冲和门控信号、通过串口传送数据等)。CPU 通过特殊功能寄存器实现对这些内部资源的访问和工作过程的控制。

1. 定时/计数器

(1) 51 子系列单片机内部分别有 2 个可编程定时/计数器,分别表示为 T0 和 T1。注意在 51 单片机的引脚上,P3.4 引脚和 P3.5 引脚的第二功能名称也为 T0 和 T1。

(2) 定时/计数器可以实现定时和计数功能。T0 有 4 种工作方式(即方式 0~3),T1 有 3 种工作方式(即方式 0~2)。当需要用 T1 工作在方式 2 为串口提供波特率发生器时,可以设置 T0 工作在方式 3,以实现一个 8 位的定时/计数器。

(3) 与定时/计数器相关的特殊功能寄存器有 TMOD 和 TCON。TMOD 用于设置定时/计数器的工作方式、定时和计数功能选择、门控信号的禁用或启用;TCON 中的 TR0 或 TR1 位用于配合门控信号等实现计数的启停控制,并且用 TF0 或 TF1 位标识计数是否溢出。特别注意,TMOD 不能进行位操作。

(4) 不管是计数还是定时功能,定时/计数器的初始化都需要设置工作方式、计数初值,而工作方式的选择也主要考虑所需计数初值的大小范围。

(5) 初始化之后,通过将 TR0 或 T1 置位以启动计数。对定时功能,通过循环不断地检测 TF0 或 TF1 是否为 1 以确定计数溢出是否溢出;对计数功能,可以在达到一定条件(例如指定的计数时间范围结束)后将 TR0 或 TR1 复位以暂停计数,以便读取计数值。

2. 串口

(1) 51 单片机中只集成了一个串口,可实现点-点双工通信或多机主从式串行通信,也可以利用串口实现并口或其他资源的扩展。在串行数据传输过程中,借用了 P3.0 引脚和 P3.1 引脚的第二功能实现串行数据和时钟信号的传输。

(2) 在初始化时,通过对 SCON 的访问,可以设置串口的工作方式、是否允许接收数据等;通过 PCON 和定时/计数器 T1 的计数初值,可以确定串行通信的波特率。

(3) 在串行通信过程中,CPU 对串行数据的收发操作都是通过 SBUF 寄存器实现的。

为确保传输的准确可靠,从 SBUF 收发数据之前必须检测 SCON 中的 TI 或 RI 标志位,以确定串口的数据收发操作已经完成,即 SBUF 中已经存放有接收到的数据,或者已经准备好 CPU 向其写入下一个待发送的数据。

(4) 如果设置串口工作在方式 2 和方式 3,可以实现多机主从式通信。借助于 SCON 寄存器中的 TB8 和 RB8 位,还可以实现串行通信中的奇偶校验等高级功能。

3. 中断系统

(1) 在 51 单片机中,根据中断源所处的位置,将所有中断分为外部中断和内部中断。第 3 章介绍了外部中断,而内部中断主要指的是本章介绍的定时/计数器中断和串口中断。

(2) 内部中断和外部中断的处理过程是类似的。如果需要采用中断方式,必须在主程序中初始化时开中断。之后,一般通过死循环等待中断请求的到来,转到相应的中断服务程序。

(3) 对定时/计数器,如果采用中断方式处理计数溢出和定时到,需要编写相应的中断服务程序,并在 ROM 的 000BH 或 001BH 单元安排一条转移指令,跳转到中断服务程序。

(4) 对串口,如果希望采用中断方式进行数据收发。此时,在初始化中需要通过 IE 寄存器打开串口中断。在工作过程中,一旦数据收发完成,通过 SCON 中的 RI 或 TI 向 CPU 发出中断请求。CPU 响应后,通过执行 ROM 中 0023H 单元中的转移指令跳转到串口中断服务程序。特别注意,每次响应串口中断后,必须用指令清除 RI 或 TI 标志位。

(5) 51 单片机将所有中断源分为两组优先级,并为每个中断规定了一个默认的优先级顺序。优先级的作用是规定了当同时有多个中断请求到来,CPU 响应的先后顺序。当 CPU 正在处理某个中断时,如果有新来的中断请求,CPU 是否予以响应,从而实现中断嵌套。

4. 定时/计数器和串口的高级应用

(1) 由于计数位数和计数脉冲频率有限,任何一个定时/计数器能够实现定时的时间长度也是有限的。在实际系统中,可以采用硬件和软件配合或者采用多个定时/计数器级联的方法实现长定时。

(2) 在工业现场利用串口进行长距离传输过程中,经常会受到各种噪声和干扰的影响,造成数据传送的错误。利用 51 单片机的串口工作在方式 2 或方式 3,可以实现奇偶校验,自动检测是否有传输错误。

(3) 在方式 2 和方式 3 下,51 单片机的串口可以实现多机主从式通信,从而构成由很多套 51 单片机系统构成的多机系统和测控网络。

思考练习

5-1 填空题

(1) 51 单片机的 P3.5 引脚工作在第二功能时称为_____信号引脚,用于输入定时/计数器所需的_____。

(2) 51 单片机的 P3.2 引脚工作在第二功能时称为_____信号引脚,通过该引脚输

入的信号可以作为定时/计数器的_____信号。

(3) 要设置定时/计数器 T0 工作在方式 1 的计数方式,允许门控;设置定时/计数器 T1 工作在方式 2 的定时方式,禁止门控,相应的指令为 MOV TMOD,_____ B。

(4) 在程序中,指令 JNB TF1, \$ 实现的功能是等待定时/计数器 T1 出现_____。

(5) 已知 51 单片机的时钟频率为 6 MHz,设置 TMOD=01000000B,用定时/计数器 T0 实现 2 ms 定时,则应设置 TH0=_____ H,TL0=_____。

(6) 已知串口每秒传送 1000 个字符,每个字符由 1 位起始位、8 位数据位和 1 位停止位构成,则波特率为_____,传输一位二进制所需的时间为_____。

(7) 利用串口进行数据的收发时,发送和接收的数据都在_____中。

(8) 通过串口发送数据之前,必须用指令将_____位复位,否则无法发送数据。当数据发送完毕,该位会自动置位。

(9) 51 单片机从串口读取接收到的数据后,必须用指令将_____位复位。当串口接收到下一个数据时,该位会自动被置位。

(10) 当 51 单片机需要通过串口接收数据时,必须在初始化时将_____位置位。

(11) RS-232C 串行通信接口标准采用负逻辑电平,即逻辑 1 用_____电平表示,逻辑 0 用_____表示。

(12) 51 子系列单片机有_____个外部中断和_____个内部中断。

(13) 在 51 单片机中,不管要使用哪个中断,都必须用指令_____开中断。

(14) 51 单片机将所有中断的优先级分为_____组,即_____优先级和_____优先级。

5-2 选择题

(1) 要使定时/计数器 T0 停止计数,可以选用的指令是()。

- A. CLR IT0 B. CLR TF0 C. CLR IE0 D. CLR TR0

(2) 定时/计数器 T1 用作计数功能时,计数脉冲由()提供。

- A. P3.5 引脚 B. P3.4 引脚 C. P0.0 引脚 D. 内部时钟电路

(3) 已知 51 单片机时钟频率为 12 MHz,要利用定时/计数器工作在方式 1 实现 1 ms 定时,所需的计数初值为()。

- A. 03E8H B. 0FC18H C. 0FCH D. 18H

(4) 要允许用 INT0 启动定时/计数器 T0 的计数,必须将 TMOD 中的()。

- A. C/ $\bar{T}$ 位置位 B. C/ $\bar{T}$ 位复位
C. GATE 位置位 D. GATE 位复位

(5) 要利用 SETB TR0 启动定时/计数器 T0 的计数,必须将 TMOD 中的()。

- A. C/ $\bar{T}$ 位置位 B. C/ $\bar{T}$ 位复位
C. GATE 位置位 D. GATE 位复位

(6) 采用查询方式检测到定时/计数器 T0 计数溢出时,计数溢出标志位 TF0 应()。

- A. 由硬件自动复位 B. 由硬件自动置位

- C. 用程序指令置位 D. 用程序指令复位
- (7) 当定时/计数器 T1 计数溢出时, TF1=()。
- A. 0 B. 1
C. 0FFH D. 当前计数值
- (8) 要从串口读入接收到的数据, 可以选用的指令是()。
- A. MOV A, DPTR B. MOV A, SCON
C. MOV A, SBUF D. MOV A, TCON
- (9) 51 单片机串口发送数据的顺序是()。
- ① 数据送 SBUF ② 硬件自动将 TI 置位
③ 通过 P3.1 引脚串行发送一帧数据完毕 ④ 用软件将 TI 复位
- A. ①③②④ B. ①②③④ C. ④③①② D. ③④①②
- (10) 51 单片机串口接收数据的顺序是()。
- ① 从 P3.0 接收一帧数据后, 硬件将 RI 置位 ② 用软件将 RI 复位
③ 从 SBUF 读取接收到的数据 ④ REN 位置位
- A. ①②③④ B. ④①②③ C. ④③①② D. ③④①②
- (11) 串口工作在方式 0 时, P3.1 引脚传输的是()。
- A. 发送的数据 B. 接收的数据 C. 同步时钟 D. 空闲
- (12) 51 单片机串口适合用于点-点双机通信的工作方式是()。
- A. 方式 0 B. 方式 1 C. 方式 2 D. 方式 3
- (13) 下面有关串口功能的描述, 错误的是()。
- A. 将发送的并行数据转换为串行数据
B. 在发送的串行数据中自动添加起始位和停止位
C. 一帧数据接收完毕将 RI 置位
D. 接收到的数据由 CPU 读走后, 自动将 TI 复位
- (14) 51 单片机复位后, 下列中断源中默认优先级最低的是()。
- A. 外部中断 $\overline{INT1}$ B. 串口中断
C. 定时/计数器 T0 中断 D. 定时/计数器 T1 中断
- (15) 当 CPU 响应定时/计数器 T0 中断时, PC 中自动装入的值是()。
- A. 0003H B. 000BH C. 0013H D. 001BH
- (16) 下面有关中断优先级的描述, 错误的是()。
- A. 每个中断源都有两个优先级
B. 低优先级的中断处理可以被高优先级打断
C. 相同优先级的中断, 可以按照默认优先级实现嵌套处理
D. 复位后, 所有中断都属于低优先级

5-3 简述定时和计数功能的主要区别。

5-4 简述 51 单片机定时/计数器查询方式实现定时功能时程序的基本流程。

- 5-5 简述 51 单片机中定时/计数器方式 2 与方式 0 和方式 1 的主要区别。
- 5-6 简述 51 单片机中用串口扩展并口的基本原理和方法。
- 5-7 总结 51 单片机的串口方式 1、2、3 之间的主要区别,完成表 5-4。

表 5-4 串口各种方式的主要区别

工 作 方 式	数 据 帧	波 特 率	典 型 应 用
方式 1			
方式 2			
方式 3			

- 5-8 简述串口工作在方式 1 时的初始化流程。
- 5-9 简述 51 单片机的中断优先级的管理规则。
- 5-10 执行如下指令 MOV IP, #1CH 后,分析说明各中断的优先级顺序。

综合设计

- 5-1 已知 51 单片机时钟频率为 12 MHz,要求通过 P1.7 端口输出高电平和低电平持续时间分别为 10 ms 和 20 ms 的周期矩形波,编写相应的汇编语言程序。
- 5-2 将动手实践 5-2 改为用中断方式实现,定时每隔 1 ms,响应一次中断。在中断服务程序中测量和显示待测脉冲的频率。
- 5-3 利用串口和并入/串出芯片 74165 为单片机扩展一个输入并口,实现 8 个开关状态的检测。画出连接线路图,并编写相应的汇编语言程序。
- 5-4 将动手实践 5-4 中的接收机改为用中断方式接收数据,编写接收机的控制程序。