

# 8051单片机C编程

计算机系902教研室

马忠梅

中 # 920

[bitmzm@sina.com](mailto:bitmzm@sina.com)

# 8051单片机基础知识

## MCS-51的特点

基本组成部件：

中央处理器：CPU 8位

数据存储器：RAM 128B

程序存储器：ROM 4KB

定时器/计数器：2个16位

I/O接口：8位 $\times$ 4

# MCS-51 INTEL 1980年

## 单片机标志:

MCS-48, MCS-51, MCS-96(16位)

8位机: 8051系列      教学首选

8051      掩膜

8031      无ROM, EPROM, FLASH

8751      EPROM

## 低功耗基本型:

80C51, 80C31, 87C51

# 8051衍生产品

Atmel 89C51, 89C52, 89C2051

Philips 80C51, 80C552, 87C752

Dallas 80C390, 80C400

Infineon C517, C509, 80C537

ADI ADuC812, ADuC824

TI MSC1210

Cygnal C8051F

## AT89C51

## AT89C52

闪存	4KB	8KB
内存	128B	256B
工作频率	24MHz	24MHz
输入/输出线	32	32
定时/计数器	2	3
中断源	5	8
串行口	1	1

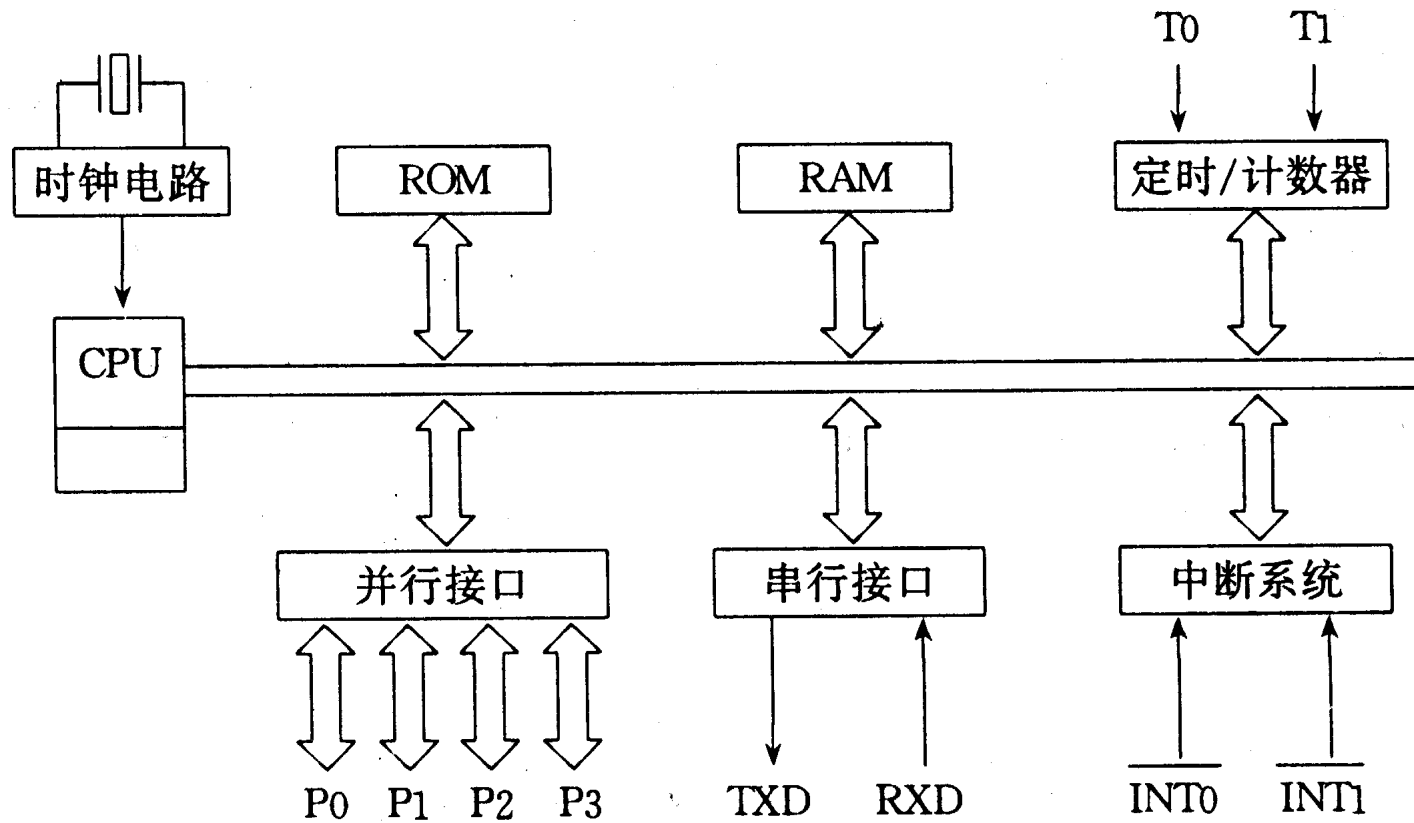


图 1-1 MCS-51 单片机的基本结构

# 内部结构

## CPU——ALU

算术运算：加，减，乘，除

逻辑运算：与，或，异或

位操作（布尔）：与，或，取反

ACC = A:累加器， B:寄存器

程序状态字：PSW 8位寄存器

## 8051时钟

内部方式：石英晶体，晶振

外部方式：外部振荡信号

## 基本时序周期

振荡周期： $1/f_{\text{OSC}}$

时钟周期： $2/f_{\text{OSC}}$

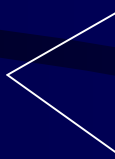
机器周期： $12/f_{\text{OSC}}=T$

指令周期： $1\sim 4T$



# 存储器组织

## 存储器特点

程序存储器		分开，哈佛型
数据存储器		合并，普林斯顿型

物理上的4 个空间：

- 1) 片内程序存储器
- 2) 片外程序存储器
- 3) 片内数据存储器
- 4) 片外数据存储器

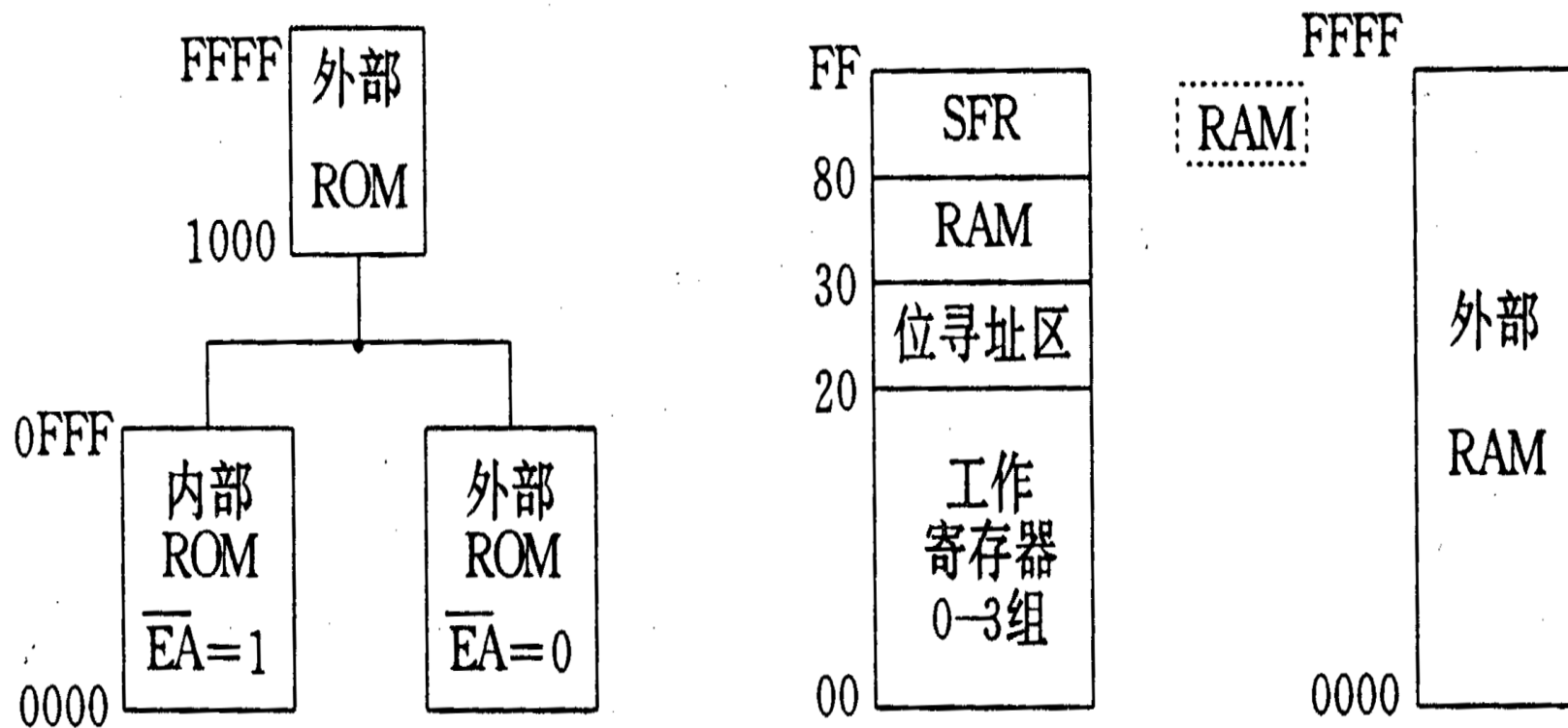


图 1-5 8051 存储器组织结构

## 程序存储器

ROM型（只读）：程序,表格常数  
当PC超过4KB,  
自动转1000H~FFFFH（片外）

## 数据存储器

RAM型（读，写）：数据暂存,  
运算结果,标志位,堆栈

片内： 256B, MOV

片外： 64KB, MOVX

片内部分2块：

00~7FH, 128B, RAM区

80H~FFH, 128B, SFR区

（特殊功能寄存器）

## 低128B

通用寄存器区， 4组 ,R0~R7

可位寻址区， 20H~2FH , 16个

用户RAM

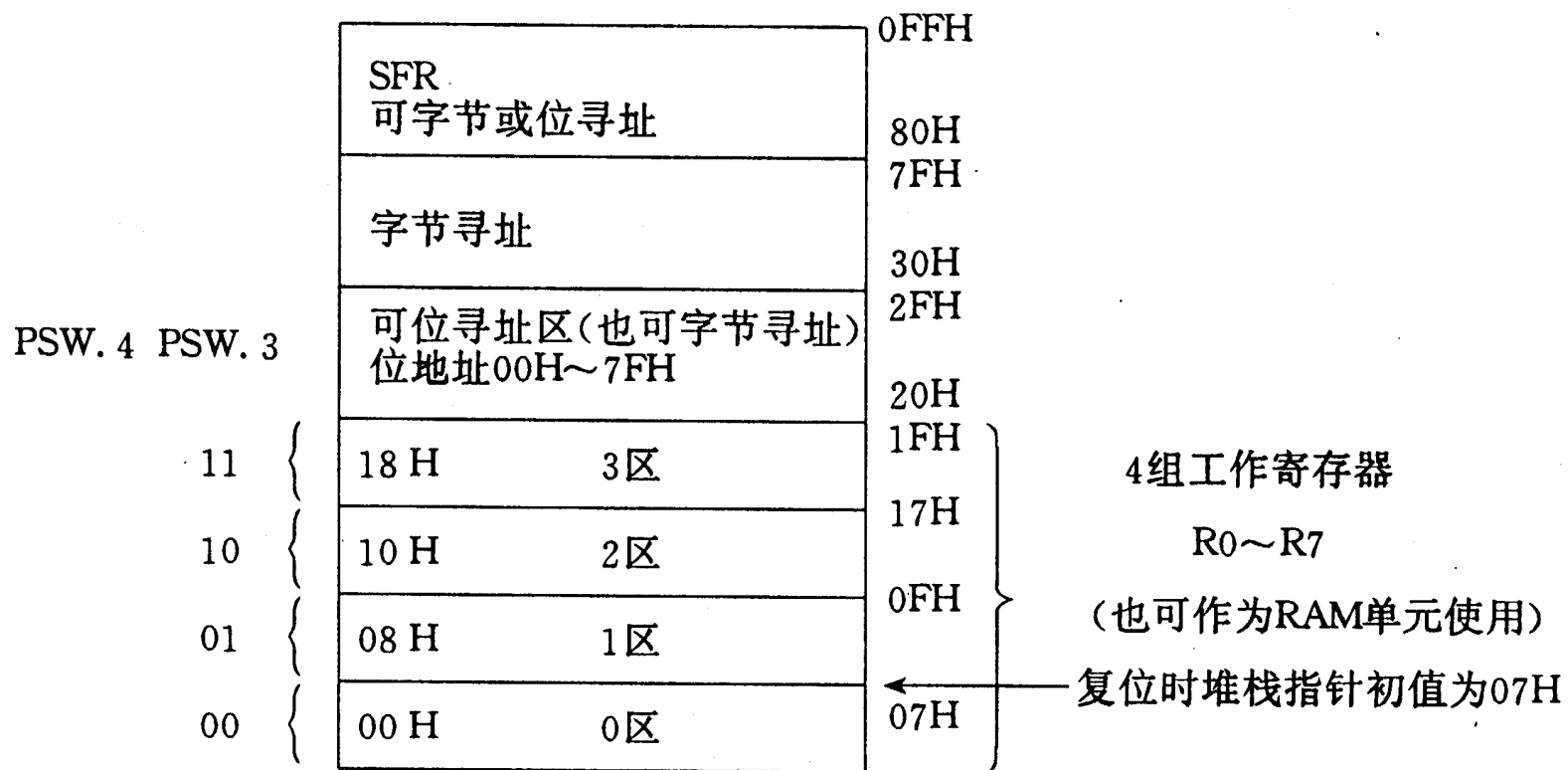


图 3-5 8051 片内 RAM 区结构

# 通用寄存器区

4个组:

0区 00H~07H

1区 08H~0FH

2区 10H~17H

3区 18H~1FH

由PSW中的RS1,RS2来决定用哪个工作区 (00, 01, 10, 11)



## 可位寻址区

20~2FH, 16字节

00~7FH, 128位

## 用户RAM

30H~7FH

堆栈, 60H ( 向上增长 )

复位后设置

# 特殊功能寄存器

21个SFR

(Special Function Register)

80H~FFH, 只能直接寻址

除PC和4组R0~R7外, 其他都是SFR,  
可位寻址的SFR, 其地址可被8整除

不同的特点：

程序存贮器和数据存贮器严格分开

特殊功能寄存器和内部数据存贮器统一编址

## 片内并行接口

4部分：端口锁存器，输入缓冲器，  
输出驱动器，端口引脚

## 准双向口

没有专用地址总线：

P2 高8位， A15~A8;

P0 低8位， A7~A0;

专用数据总线： P0, D7~D0

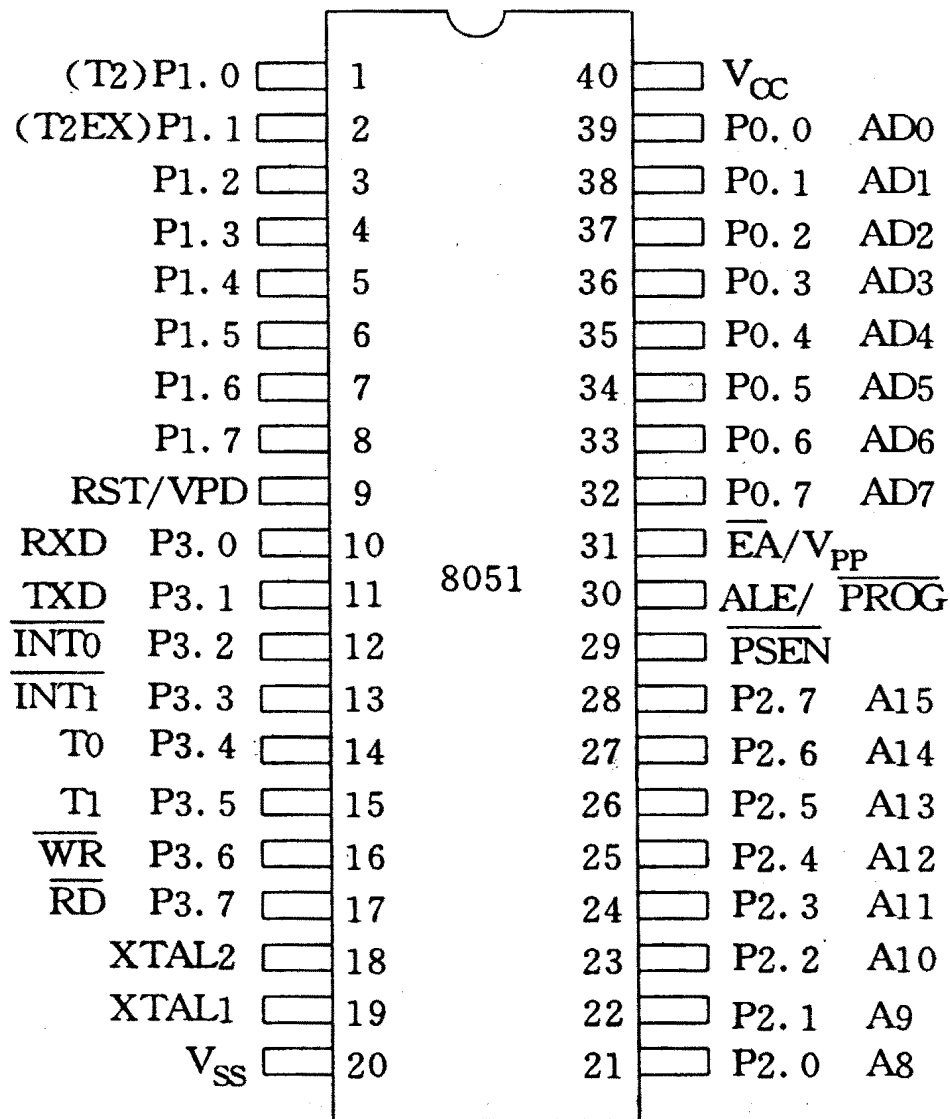


图1-8 MCS-51引脚图

# 8051内部资源

串行口

定时器/计数器

中断端口

# 单片机工作方式

低功耗操作

两种：节电（空闲）方式  
掉电

单步执行方式：

用于单片机开发工具或仿真器

# 指令系统

## 寻址方式

寄存器寻址:

MOV A, R0

寻址空间: R0~R7, A, B, C, DPTR,  
AB (乘法)



直接寻址:

MOV A, 4FH

MOV A, P0

寻址空间: 内部RAM低128字节,  
SFR

## 寄存器间接寻址:

MOV A, @R1 ;(R1)=40H

MOVX A, @R0

MOVX A, @DPTR

## 寻址空间:

内部RAM(@R0, @R1, @SP)

外部RAM(@R0, @R1, @DPTR)

立即寻址:

MOV A, #6FH

MOV DPTR, #1234H

寻址空间: 程序存储器

变址寻址:

MOVC A, @A+DPTR

MOVC A, @A+PC

变址寻址方式

只适用于8051的程序存储器，用于  
读取数据表。

相对寻址:

SJMP REL

寻址空间: 程序存储器

位寻址:

SETB BIT

寻址空间: 内部RAM可位寻址区;  
SFR 可位寻址位

# C与8051

8051的编程语言：

有4种语言支持

汇编，PL/M, C和BASIC

C语言作为一种方便的语言而得到支持，不依赖于机器的硬件系统。

## C51编译器

作为工业标准地位，从1985年开始就有8051单片机的C语言编译器，简称C51。

KEIL和IAR领先，

KEIL以它的紧凑代码和使用方便领先

IAR以它性能完善和资料完善领先

FRANKLIN (KEILV4.0)

ARCHIMEDES (IARV4.0)



《单片机的C语言应用程序设计》(修订版)

北京航空航天大学出版社

Intel Microcontroller Data Sheet

Schltz, Thomas W. C and 8051: Programming  
for multitasking .Prentice Hall

嵌入式C编程技术

《单片机与嵌入式系统应用》 2001(1~6)

# 《单片机C语言Windows环境编程宝典》

KEIL Cx51 uVision2

[www.zlgmcu.com](http://www.zlgmcu.com)

[www.c51bbs.com](http://www.c51bbs.com)

[www.dpj.com.cn](http://www.dpj.com.cn)

# 数据与数据类型

C51编译器具体的数据类型：

位型，无符号字符，有符号字符，  
无符号整型，有符号整型，无符号长型，  
有符号长型， 浮点和指针类型等。

bit, unsigned char, signed char, unsigned int,  
signed int, unsigned long, signed long,  
float, double

## 常量与变量

习惯上，符号常量名用大写，变量用小写，以示区别。

只有bit和unsigned char两种数据类型可以直接支持机器指令，必须慎重选择变量的数据类型。

程序的开头都加上以下三行：

```
#include<reg51.h>
```

```
#define uchar unsigned char
```

```
#define uint unsigned int
```

头文件reg51.h中有所有8051的SFR  
及可位寻址位的定义

```
bit direction_bit;  
uchar i;  
{   i=PSW;  
    C=direction_bit;  
    P1=0x10;  
    direction_bit = C;  
    PSW = i;  
}
```

```
#include<absacc.h>
```

```
#define PORTA XBYTE[0xffc0]
```

```
{
```

```
    i=PORTA;
```

```
    PORTA=i;
```

```
}
```

## 存储类型

## 说明

code	程序，MOVC @A+DPTR访问
xdata	外部数据，由MOVX @DPTR访问
pdata	分页外部数据，由MOVX @Ri访问
data	直接寻址内部数据存储区
bdata	可位寻址内部数据存储区
idata	间接寻址内部数据存储区



## 存储模式

## 说明

SMALL	可直接寻址的内部数据存储区
COMPACT	分页外部数据存储区
LARGE	外部数据存储区

参数和局部变量放入

基于存储器的指针

1~2字节

一般指针

3字节

2字节偏移和1字节存储类型

float \*p

3字节

char data \*dp

1字节

int idata \*ip

1字节

long pdata \*pp

1字节

char xdata \*xp

2字节

int code \*cp

2字节

# 内部资源的C编程

## 中断

8051单片机有5个中断源，有2个中断  
优先级

外中断方式： 电平触发  
边沿触发

## 五个中断源

- 1) 外部中断请求0
- 2) 外部中断请求1
- 3) 片内定时器/计数器0
- 4) 片内定时器/计数器1
- 5) 片内串行口发送/接收中断请求

## 中断允许寄存器IE

EA ET2 ES ET1 EX1 ET0 EX0

## 中断优先级寄存器IP

PS PT1 PX1 PT0 PX0

## 中断源

## 入口地址

0	外中断0	0003H
1	定时器/计数器0	000BH
2	外中断1	0013H
3	定时器/计数器1	001BH
4	串口	0023H

# 中断撤除

可中断撤除

串行口 JBC TI TSEVICE

软件清除

电平触发，返回主程序，又进入中断

## 中断初始化

- 1、开中断
- 2、确定中断优先级
- 3、外中断、中断类型

标志位：必须撤除



# 中断编程

返回值 函数名 `interrupt n using m`

寄存器组切换，现场保护

# 定时器/计数器 (I/C)Timer/Counter

两个T/C 16位 可编程选择

T/C0:TH0 TL0 T/C1:TH1 TL1

定时器---固定软件计数----计数值算时-  
---定时

计数器----外部输入T0/T1 脉冲计数

T/C加1计数 (每个T)

## 工作方式四种:

方式0      13计数器    MCS-48兼容

方式1      16计数器

方式2      可重装8位计数器

方式3      T0分为两个8位计数器

方式2: TH(重装) TL(8位)

$2^8$ 溢出 TH自动装入TL

串行口波特率发生器

方式3: TH0(8) TL0(8)

TH0: 只能T 占T/C1

TL0: 可作T/C 占T/C0

**注:** T/C1 不能工作方式3

T/C1 只能用于串口

# T/C的控制

## 启动和中断

TR0和TR1 启动控制位

Timer Run Control Bit

方式控制 TMOD

GATE C/T ( 1: 计数器 )

M1 M0: 工作方式选择

00 01 10 11

GATE 门控信号

GATE=1即:

T/C0启动=TR0 INT0引脚高电平

T/C1启动=TR1 INT1引脚高电平

INT0/INT1 不是中断请求,附加控制

脉冲宽度的测量

TMOD不能位寻址

# T/C的初始化

## 步骤

- ①确定(T/C工作方式)----TMOD赋值
- ②计算(T/C初值)----装TH TL
- ③T/C中断方式---IE
- ④启动T/C ---TR置位

# 初值的计算

方式0	13位	满计数值	$2^{13}=8192$
方式1	16位	满计数值	$2^{16}=65536$
方式2	8位	满计数值	$2^8=256$

计数模值

初值 $X=M$ -计数模值



# 定时器初值

$$\text{定时值} = (M - X)T$$

向上计数

$$f_{\text{osc}} = 12\text{MHz}$$

$$T = 1\mu\text{s}$$

$$f_{\text{osc}} = 6\text{MHz}$$

$$T = 2\mu\text{s}$$

方式1      16 位

$$2^{16}\mu\text{S} = 65.536\text{mS}$$

方式2      8位

$$2^8\mu\text{S} = 256\mu\text{S}$$

最大定时值

定时1ms, 求初值

方式1

$$(2^{16}-x)2\mu s=1ms$$

$$X=2^{16}-500=FE0CH$$

$$TH1=FEH \quad TL1=0CH$$

## T/C的应用举例

$$(2^{16}-x)1\mu s=1ms$$

$$X=2^{16}-1000$$

$$TH0=(65536-1000)/256;$$

$$TL0=(65536-1000)\%256;$$

```
void timer(void)interrupt 1 using1
{  P1_0=!P1_0;
    TH0=(65536-1000)/256;
    TL0=(65536-1000)%256;
}
```

# 串行口

片内全双工UART

发TXD 收RXD

两个缓冲器 同一个地址99H

收发过程由UART管理

定时器有关

用定时器1作波特率发生器  
(固定波特率除外)

TMOD TH1 TL1 TCON(TR1)  
IE(ET1 EA)

没用IP 重装方式2可不用IE

# 串行口控制寄存器SCON

SM0 SM1: 工作方式选择

SM2: 第九位方式控制

REN: 接收允许(RECEIVE ENABLE)

TB8: 发送的第九位数据(TRANSMIT BIT)

RB8: 接收的第九位数据(RECEIVE BIT)

TI: 发送中断标志位 (硬件置位)

RI: 接收 (软件清除)

电源控制寄存器

SMOD=1 波特率就加倍

与中断有关

四种工作方式



## 方式0

移位寄存器输入/输出方式

串行数据通过RXD输入/输出

TXD用于输出移位时钟

8位数据 波特率固定

并入串出(发送)或串入并出(接收)寄存器

用于并行I/O扩展

## 方式1

10位异步接收/发送

1位起始位 8位数据 1位停止位

波特率可变，定时器1波特率信号经  
16或32分频

波特率 =  $2^{\text{SMOD}}$  定时器1溢出率 / 32

## 注意：

①发送时 $TI=0$  以SBUF为目的的指令启动发送，完成后 $TI=1$

②接收时 $REN=1$ 与 $RI=0$

$SM2=0$  接收到 $RI=1$

$SM2=1$  要求接收到有效停止位才 $RI=1$

有效接收前提： $REN=1$

条件:(1) $RI=0$

(2) $SM0=0$ 或接收停止位为1

方式2、方式3

11位异步接收/发送

第九位 TB8/RB8

方式2 波特率固定

方式3 同方式1可变

**注意：**同方式1，只多一个第9位  
有效接收条件，(1)RI=0 (2)SM2=0  
或第9位为1。**SM2**多机通信

# 串行口编程

**注意：**波特率和通信格式一致，波特率可查表

1) 点对点通信

主方：发一个，收一个

从方：收一个，发一个

设置1200波特，串行口方式1，

$f_{osc}=11.0592\text{Hz}$

```
slave()
```

```
{ uchar a;
```

```
  TMOD= 0x20;
```

```
  TL1=0xe8;TH1=0xe8; /* T/C1方式2 */
```

```
  SCON=0x50;PCON=0x00; /* 方式1 */
```

```
  TR1=1;
```

```
while(1){  
    while(RI==0);  
    RI=0;  
    a=SBUF;  
    SBUF=a;  
    while(TI==0);  
    TI=0;  
}  
}
```

## 2) 多机通信

主机对多个从机

①每个从机都有不同的地址。

②主机发出信息两个

a、地址    b、数据

③采用设置SM2=1，第9位为1时才接收



## 多机通信的过程

- ①所有从机的SM2置1，从而接收主机发来的地址
- ②主机令TB8=1，发送从机地址到各从机
- ③所有从机都收到地址，确认地址(可采用中断)
- ④被寻址的从机用指令置SM2=0，准备收数据并向主机回发地址以便核对
- ⑤主机发送数据给已被寻址的从机

# 8051扩展资源的C编程

单片机系统扩展

P0 复用 P2高址

P3 多功能 (RD, WR)

外部总线的扩展

ALE—地址锁存允许

Address Latch Enable

控制外接锁存器 P0口地址锁存

在ALE无效期间数据

ALE每个T出现两次,  $f=1/6f_{osc}$

- 1、作脉冲信号
- 2、证明单片机工作

## MCS-51外部扩展三总线

1、ALE下降沿 P0 地址有效

高电平触发或下降沿触发锁存器

Intel8282 74LS373

带三态输出锁存器

2、外部RAM——RD WR

外部ROM——PSEN

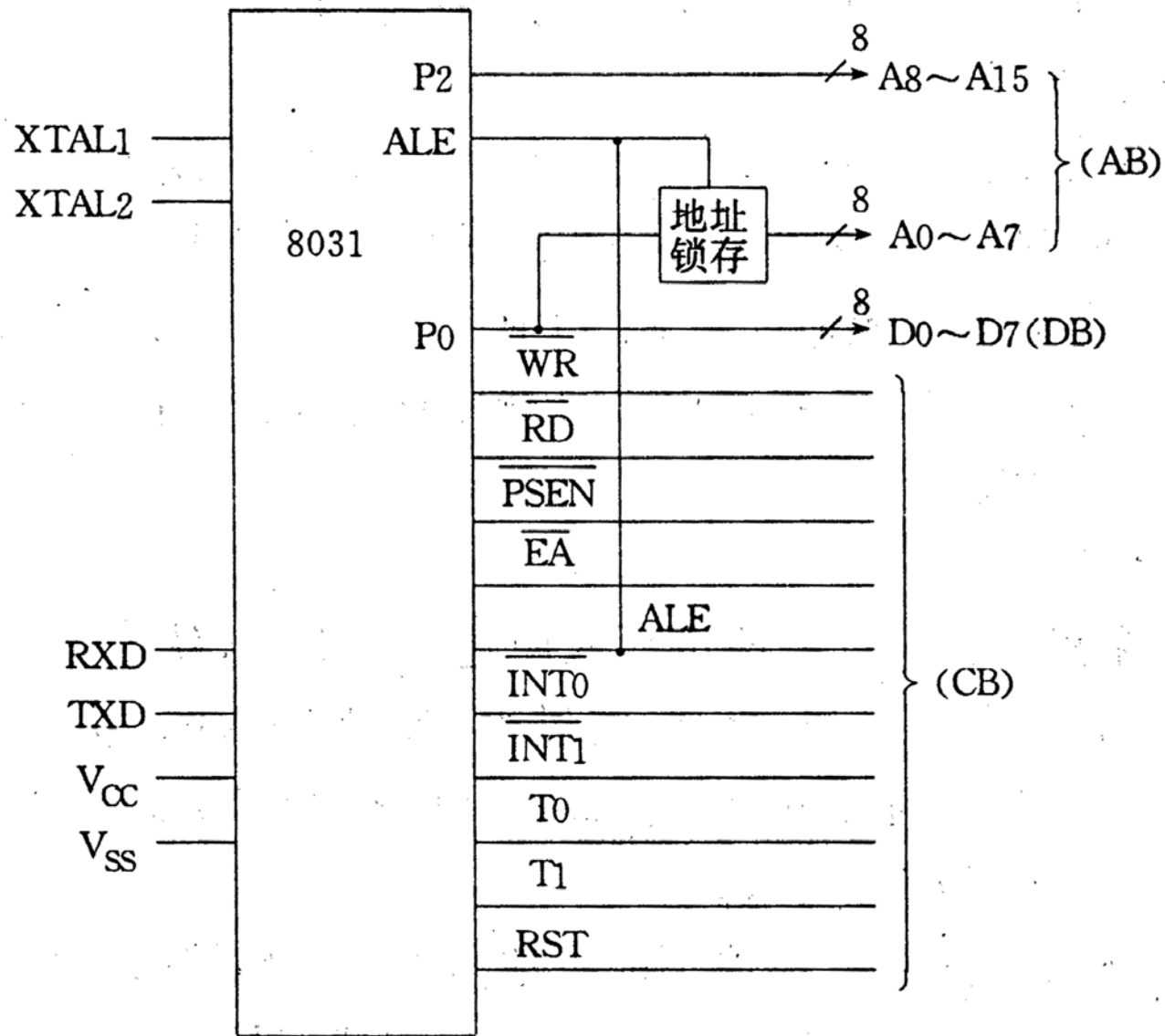


图 1-10 MCS-51 外部三总线示意图

# 多片的扩展

片选信号 线选法 分别作片选信号

译码法 加译码器译码

译码法 全译码 无地址重叠区

部分译码 有重叠

A15	A14	A13	
0	1	1	X
1	0	1	X
1	1	0	X

# 人机交互的C编程

## 键盘和LED显示

### 键盘

有键按下→哪一个键→键的代码

# 行列式键盘工作原理

接上拉电阻——平时——高电平

行输出低 扫描法——键识别

①查询是否有键按下

②按键所在的行列

③行号、列号译码→键值

特征码：0001 0001 / 0010 0001

④键的抖动处理



# 多位LED

1、静态

2、动态:停留 $1\sim 5\text{ms}$ , 视觉惯性  
省I/O线, 但占用CPU时间

# 可编程键盘/显示芯片8279

六部分：

(1) I/O控制及数据缓冲器

(2) 控制及定时寄存器及定时控制

(3) 扫描计数器

编码工作方式 译码工作方式

(4) 回复缓冲器 键盘反复的控制

(5) FIFO/传感器 RAM: 状态寄存器

(6) 显示RAM和显示地址寄存器

# 混合编程

在用C语言开发程序的过程中，有时感到速度达不到要求，如很明显的显示内容的更新或做显示内容的移动。在这种情况下，可以找到速度的瓶颈函数：数据移动和把显示内容从内存移到显示扫描存储器的函数。

**move( )和toscr( )函数**

## 参数传递的寄存器选择

参数类型	char	int	long,float	一般指针
第1个参数	R7	R6,R7	R4~R7	R1,R2,R3
第2个参数	R6	R4,R5	R4~R7	R1,R2,R3
第3个参数	R5	R2,R3	R4~R7	R1,R2,R3

# 函数返回值的寄存器

返回值	寄存器	说明
bit	C	进位标志
char	R7	
int	R6,R7	高字节在R6，低字节在R7
long	R4~R7	高字节在R4，低字节在R7
float	R4~R7	IEEE格式
指针	R1,R2,R3	R3放存储类型，高R2，低R1

# 函数名的转换

说 明	符号名	解 释
<code>void func(void)</code>	<code>FUNC</code>	无参数传递或不含寄存器参数的函数名不作改变转入目标文件中，名字只是简单地转为大写形式
<code>void func(char)</code>	<code>_FUNC</code>	带寄存器参数的函数名加入“_”字符前缀以示区别，它表明这类函数包含寄存器内的参数传递

## 混合编程的设计过程

用汇编语言重新编制上面两个函数。  
应注意的是：首先设计包含哑函数的C模块，即把源程序中的“move( )”和“toscr( )”用空函数来代替。

```
void move(uint src,uint dest,uint Length)
{

}
}
```

使用debug code控制命令编译生成的列表文件相应内容如下：

; FUNCTION \_move (BEGIN)

0000 8E00 R MOV src,R6

0002 8F00 R MOV src+01H,R7

0004 8C00 R MOV dest,R4

0006 8D00 R MOV dest+01H,R5

0008 8A00 R MOV length,R2

000A 8B00 R MOV length+01H,R3

000C 22 RET

; FUNCTION \_move (END)



可源程序中的“move( )”和“toscr( )”函数注释掉，然后用汇编语言重新编制这两个函数

```
PUBLIC      _MOVE
MOVEP      SEGMENT      CODE
RSEG       MOVEP
_MOVE:     NOP
MOVE:      MOV          P1,#1FH
           MOV          DPL,R5
           MOV          DPH,R4
MOVEL:     PUSH         DPL
           PUSH         DPH
           MOV          DPL,R7
           MOV          DPH,R6
           MOVBX        A,@DPTR
```

END