STC89C51RC / RD+ 系列单片机中文指南

---无法解密

---低功耗,超低价

--- 高速，高可靠

--- 强抗静电，强抗干扰

STC89C51RC, STC89LE51RC
STC89C52RC, STC89LE52RC
STC89C53RC, STC89LE53RC

STC89C54RD+, STC89LE54RD+
STC89C58RD+, STC89LE58RD+
STC89C516RD+, STC89LE516RD+

STC89LE516AD, STC89LE58AD
STC89LE54AD, STC89LE52AD

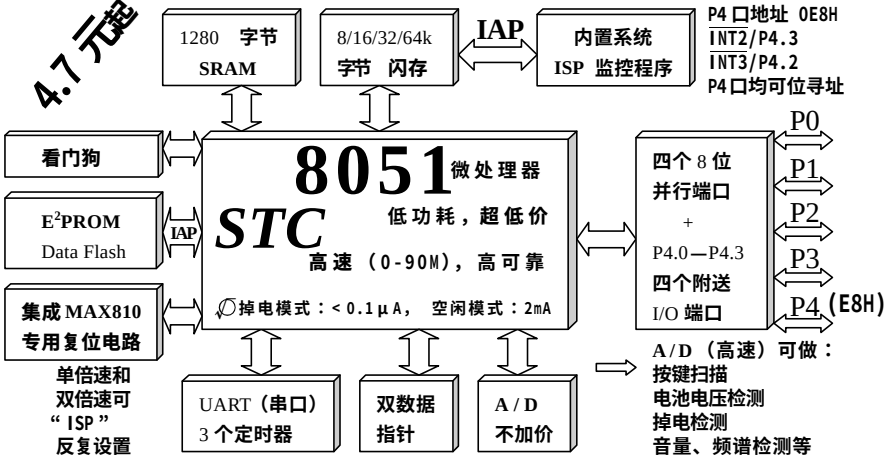
STC12C2052, STC12C4052
STC12C2052AD, STC12C4052AD

国内技术支援：宏晶科技（深圳）
www.MCU-Memory.com support@mcu-memory.com
Update date: 2005-9-20

目 录

STC 89 系列单片机，高速、高可靠、在线编程

—— 提升的是性能，降低的是成本



选择 STC89C52RC 系列 STC89C58RD+ 系列单片机的理由：

★加密性强，无法解密

★超强抗干扰：

1、高抗静电（ESD 保护）
2、轻松过2KV/4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围，-40℃～85℃
5、I/O 口经过特殊处理
6、单片机内部的电源供电系统经过特殊处理
7、单片机内部的时钟电路经过特殊处理
8、单片机内部的复位电路经过特殊处理
9、单片机内部的看门狗电路经过特殊处理

★三大降低单片机时钟对外部电磁辐射的措施：
—— 出口欧美的有力保证

1、禁止ALE 输出；
2、如选6 时钟/机器周期，外部时钟频率可降一半；
3、单片机时钟振荡器增益可设为1/2gain。

★超低功耗：

1、掉电模式：　典型功耗　<0.1μA
2、空闲模式：　典型功耗　2mA
3、正常工作模式：　典型功耗　4mA - 7mA
4、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★在系统可编程，无需编程器，可远程升级

★可送STC-ISP 下载编程器，1万片/人/天

★可供应内部集成MAX810 专用复位电路的单片机，只有D 版本才有内部集成专用复位电路，原复位电路可以保留，也可以不用，不用时RESET 脚直接短到地

STC 89 系列单片机选型一览表　超低价

型　号	最高时钟频率Hz 5V	最高时钟频率Hz 3V	Flash存储器	RAM字节	降低EMI	看门狗	双倍速	P4口	ISP	IAP	E²PROM字节
STC 89C51 RC	0～80M		4K	512	√	✆	√	√	√	√	2K
STC 89C52 RC	0～80M		8K	512	√	✆	√	√	√	√	2K
STC 89C53 RC	0～80M		15K	512	√	✆	√	√	√	√	
STC 89C54 RD+	0～80M		16K	1280	√	✆	√	√	√	√	16K
STC 89C55 RD+	0～80M		20K	1280	√	✆	√	√	√	√	16K
STC 89C58 RD+	0～80M		32K	1280	√	✆	√	√	√	√	16K
STC 89C516 RD+	0～80M		64K	1280	√	✆	√	√	√	√	
STC 89LE51 RC		0～80M	4K	512	√	✆	√	√	√	√	2K
STC 89LE52 RC		0～80M	8K	512	√	✆	√	√	√	√	2K
STC 89LE53 RC		0～80M	15K	512	√	✆	√	√	√	√	
STC 89LE54 RD+		0～80M	16K	1280	√	✆	√	√	√	√	16K
STC 89LE58 RD+		0～80M	32K	1280	√	✆	√	√	√	√	16K
STC 89LE516RD+		0～80M	64K	1280	√	✆	√	√	√	√	

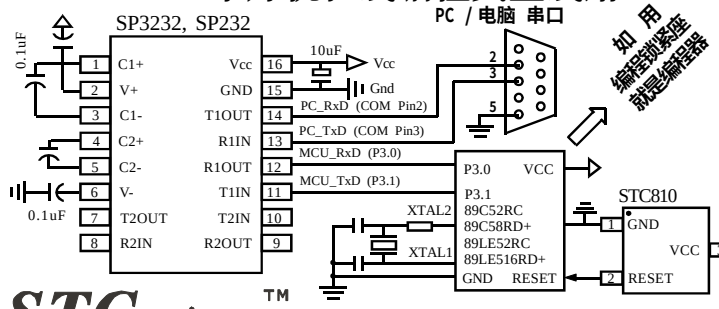
关于单片机说明：<管脚与流行的8051 兼容>　　人民币4.7 元起

DIP-40，PLCC-44，PQFP-44 封装(RC/RD+ 系列PLCC、PQFP 有P4 口地址E8H,AD 系列P4 口为C0H)
RC/RD+ 系列PLCC、PQFP 多两个外部中断P4.2/INT3，P4.3/INT2。 P4 口均可位寻址
5V：5.5V～3.8V 乃至3.4V（24M 以下）；3V：3.6V～2.4V 乃至2.0V，仅针对RC/RD+ 系列
✆ 真正的看门狗，可放心省去外部看门狗，缺省为关闭，打开后无法关闭。单倍速和双倍速可反复设置
✆ “6 时钟/机器周期”和“12 时钟/机器周期”可在ISP 编程时反复设置，新的设置冷启动后才生效
另 STC89LE516AD、58AD、54AD、52AD、51AD 系列单片机，带高速A/D 转换

STC 具有A/D 转换功能的单片机选型指南

型　号	最高时钟频率Hz 1.9-3.6V	程序存储器	RAM字节	降低EMI	双倍速	P4口	ISP	A/D	供货
STC89LE516 AD	0-90M	64K	512	√		√	√	√	现货
STC89LE58 AD	0-90M	32K	512	√		√	√	√	现货
STC89LE54 AD	0-90M	16K	512	√		√	√	√	现货
STC89LE52 AD	0-90M	8K	512	√		√	√	√	现货
STC89LE51 AD	0-90M	4K	512	√		√	√	√	定货
STC89LE516 X2	0-90M	64K	512	√	√	√	√	√	现货

STC 单片机在线编程典型线路



STC 单片机在系统可编程的使用，软件界面

STC micro ™

宏 晶 科 技

专业单片机、存储器供应商

新客户请直接联系深圳以获得更好的技术支持和服务

欢迎海内外厂家前来洽谈合作

技术支持：13922805190　　网址：www.MCU-Memory.com

深　圳：Tel：0755-82948409　82948410　Fax：0755-82944243　82905966
上海办：Tel：021-53560136　53560138　Fax：021-53080587
北京办：Tel：010-62538687　62634001　Fax：010-62538683
南京办：Tel：025-86893767　86893566　Fax：025-86893757
广州办：Tel：020-38851405　38850557　Fax：020-38850581

免费索取

从网上下载样品申请单，传真至深圳申请STC 单片机样片及ISP 下载线/编程工具

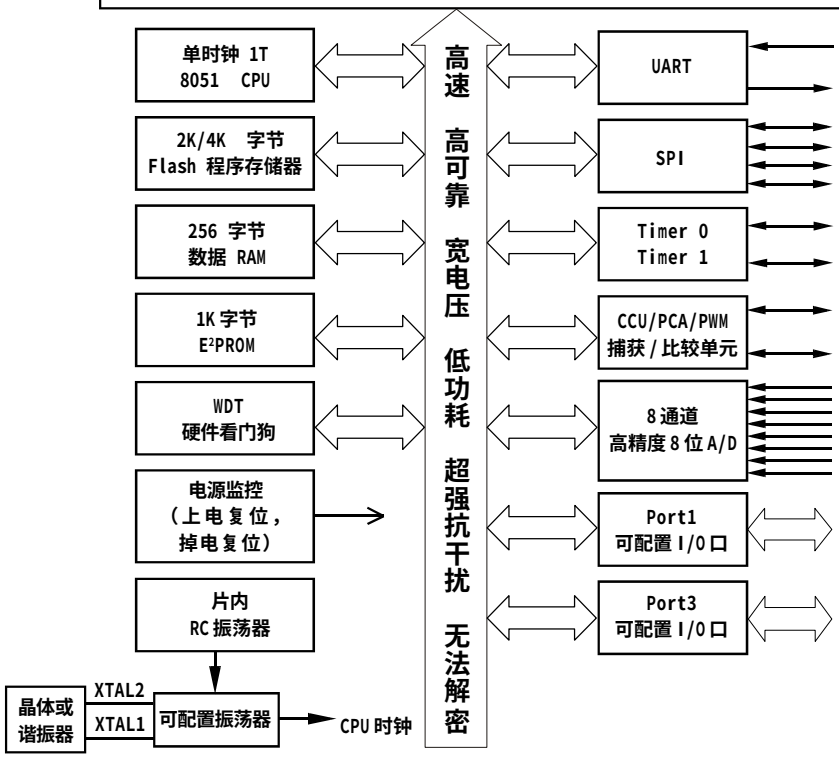
领导业界革命 覆盖市场需求

STC 12C2052AD 系列 1T 8051 单片机

——1 个时钟 / 机器周期，高速、高可靠，相当于普通 8051 0～420MHz

宏晶科技是新一代增强型8051单片机标准的制定者,致力于提供满足中国市场需求的世界级高性能单片机技术,在业内处于领先地位。销售网络覆盖全国,在保证质量的基础上,以极低的价格和完善的服务赢得了客户的长期信赖。目前,即将推出“1 个时钟 / 机器周期”的单片机,提升 8051 单片机性能。欢迎海内外厂家前来洽谈合作!新客户请直接联系深圳,以获得更好的技术支持与服务。

DIP-20，SOP-20 超小封装 8051 单片机



选择 STC12C2052AD 系列单片机的理由：

★加密性强,无法解密

★超强抗干扰:

1、高抗静电（ESD 保护）
2、轻松过 4KV 快速脉冲干扰(EFT 测试)
3、宽电压，不怕电源抖动
4、宽温度范围,-40℃～85℃
5、I/O 口经过特殊处理
6、单片机内部的电源供电系统经过特殊处理
7、单片机内部的时钟电路经过特殊处理
8、单片机内部的复位电路经过特殊处理
9、单片机内部的看门狗电路经过特殊处理

★1 个时钟 / 机器周期，可用低频晶振，大幅降低 EMI

★超低功耗:

1、掉电模式： 典型功耗 <0.1μA
2、空闲模式： 典型功耗 2mA
3、正常工作模式： 典型功耗 4mA - 7mA
4、掉电模式可由外部中断唤醒，适用于电池供电系统，如水表、气表、便携设备等。

★在系统可编程,无需编程器,可远程升级

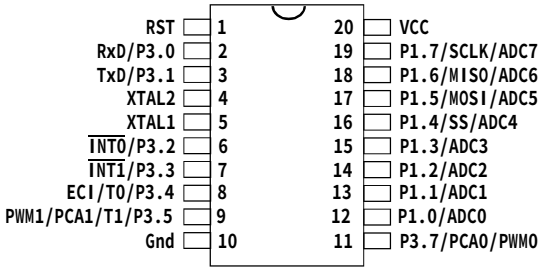
★可送 STC-ISP 下载编程器,1 万片 / 人 / 天

★内部集成 MAX810 专用复位电路，原复位电路可以保留，也可以不用，不用时 RESET 脚直接短到地

STC12C2052/STC12C4052 主要性能：

●高速：1 个时钟 / 机器周期，RISC 型 CPU 内核，速度比普通 8051 快 12 倍
●宽电压：3.4～5.5V，2.0～3.8V（STC12LE2052AD 系列）
●低功耗设计：空闲模式，掉电模式（可由外部中断唤醒）
●工作频率：0～35MHz，相当于普通 8051：0～420MHz
●时钟：外部晶体或内部 RC 振荡器可选
●512/1K/2K/3K/4K/5K 片内 Flash 程序存储器，擦写次数 10 万次以上
●256 字节片内 RAM 数据存储器
●芯片内 E²PROM 功能
●ISP / IAP，在系统可编程
●~~2 个模拟比较器~~ （无）
●8 通道高精度 8 位 ADC
●2 通道捕获 / 比较单元（CCU/PCA/PWM），提供 PWM 功能
●2 个 16 位定时器
●硬件看门狗（WDT）
●高速 SPI 通信端口
●增强型 UART
●先进的 RISC 精简指令集结构，兼容普通 8051 指令集
　111 条功能强大的指令，有 12 条指令只需 1 个时钟就可完成
　片内集成硬件乘法器和硬件除法器（执行速度为 4 个时钟周期）
　4 组 8 个 8 位通用工作寄存器（共 32 个通用寄存器）

1 个时钟 / 机器周期，超小封装 8051 单片机



DIP-20，SOP-20 超小封装
STC12C2052、STC12C4052 不带 A/D 转换
STC12C2052AD、STC12C4052AD 带 A/D 转换

STC micro™
宏 晶 科 技
专业单片机、存储器供应商
新客户请直接联系深圳以获得更好的技术支持和服务
欢迎海内外厂家前来洽谈合作

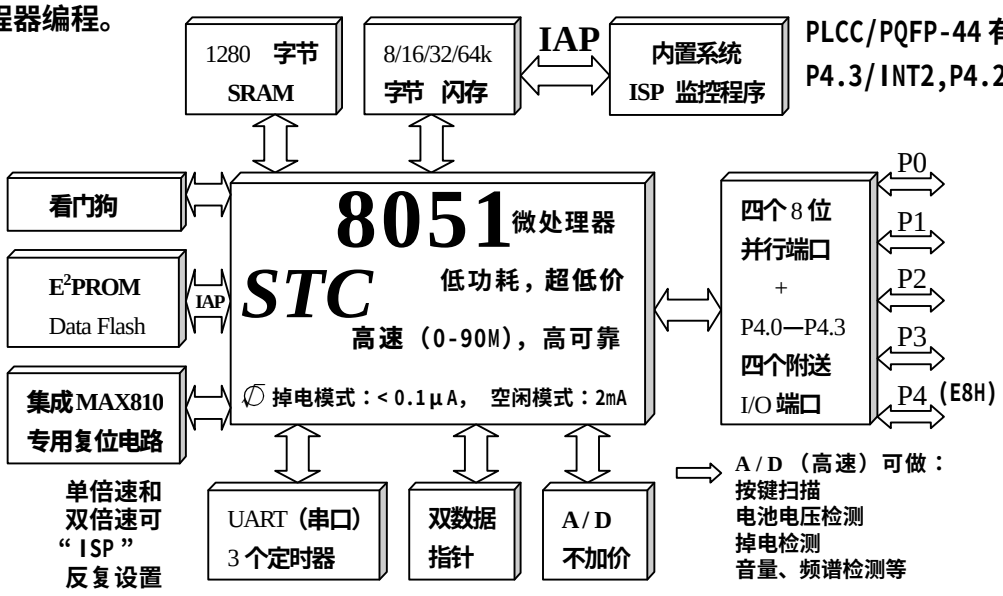
技术支持：13922805190 网址：www.MCU-Memory.com

深 圳:Tel：0755-82948409 82948410 Fax：0755-82944243 82905966
上海办:Tel：021-53560136 53560138 Fax：021-53080587
北京办:Tel：010-62538687 62634001 Fax：010-62538683
南京办:Tel：025-86893767 86893566 Fax：025-86893757
广州办:Tel：020-38851405 38850557 Fax：020-38850581

本应用技术手册是针对有一定 8051 系列（MCS-51）单片机编程基础的用户编写的。

选用 STC 单片机的理由：降低成本，提升性能，原有程序直接使用,硬件无需改动。

STC 公司鼓励您放心大胆选用 PLCC， PQFP 小型封装，3.3V 工作电压单片机，使您的产品更小，更轻，功耗更低。如果相关新增功能没有用到，则不需看相应部分。用 STC 提供的 STC-ISP.exe 工具将您原有的代码下载进 STC 相关的单片机即可,或用通用编程器编程。



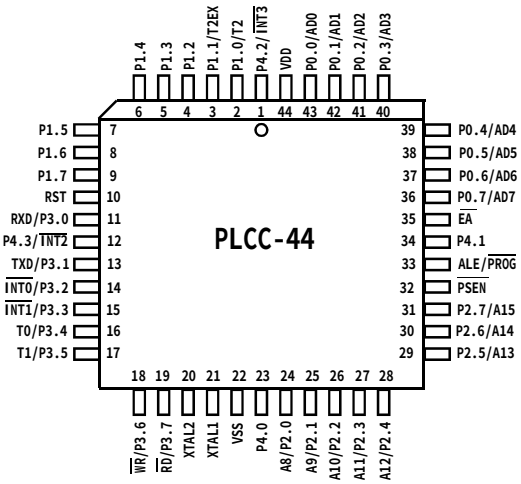
5V: 5.5 - 3.8V,乃至 3.4V； 3V: 3.6 - 2.4V,乃至 1.9V

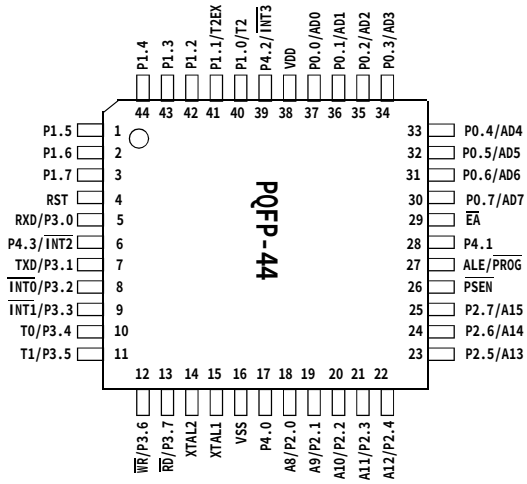
STC89 系列单片机选型一览表 IAP / ISP 美国技术， 超低价格，15 分钟学会

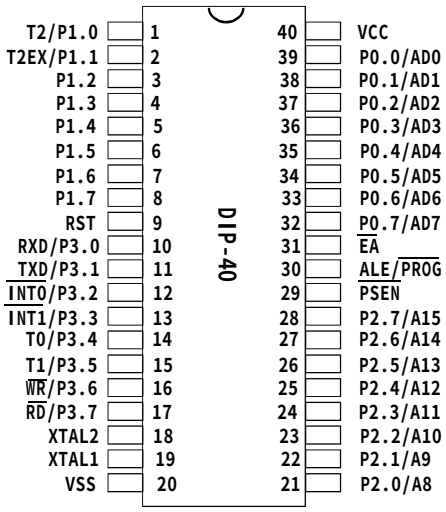
型 号	最高时钟频率Hz 5V	最高时钟频率Hz 3V	Flash 程序存储器	RAM 数据存储器	降低 EMI	看门狗	双倍速	P4 口	ISP	IAP	EEP ROM	数据指针	串口 UART	中断源	优先级	定时器	A/D	向下兼容 Winbond	向下兼容 Philips	向下兼容 Atmel
STC89C51 RC	0-80M		4K	512	√	√	√	√	√	√	2K+	2	1ch+	8	4	3		W78E51	P89C51	AT89C51
STC89C52 RC	0-80M		8K	512	√	√	√	√	√	√	2K+	2	1ch+	8	4	3		W78E52	P89C52	AT89C52
STC89C53 RC	0-80M		15K	512	√	√	√	√	√	√		2	1ch+	8	4	3		W78E54	P89C54	AT89C55
STC89C54 RD+	0-80M		16K	1280	√	√	√	√	√	√	16K+	2	1ch+	8	4	3		W78E54	P89C54	AT89C55
STC89C55 RD+	0-80M		20K	1280	√	√	√	√	√	√	16K+	2	1ch+	8	4	3		W78E58	P89C58	AT89C55
STC89C58 RD+	0-80M		32K	1280	√	√	√	√	√	√	16K+	2	1ch+	8	4	3		W78E58	P89C58	AT89C51RC
STC89C516 RD+	0-80M		63K	1280	√	√	√	√	√	√		2	1ch+	8	4	3		W78E516	P89C51RD2	AT89C51RD2
STC89LE51 RC		0-80M	4K	512	√	√	√	√	√	√	2K+	2	1ch+	8	4	3		W78LE51		AT89LV51
STC89LE52 RC		0-80M	8K	512	√	√	√	√	√	√	2K+	2	1ch+	8	4	3		W78LE52		AT89LV52
STC89LE53 RC		0-80M	14K	512	√	√	√	√	√	√		2	1ch+	8	4	3		W78LE54		AT89LV55
STC89LE54 RD+		0-80M	16K	1280	√	√	√	√	√	√	16K+	2	1ch+	8	4	3		W78LE54		AT89LV55
STC89LE58 RD+		0-80M	32K	1280	√	√	√	√	√	√	16K+	2	1ch+	8	4	3		W78LE58		AT89LV51RC
STC89LE516RD+		0-80M	63K	1280	√	√	√	√	√	√		2	1ch+	8	4	3		W78LE516	P89LV51RD2	AT89LV51RD2
STC89LE516AD		0-90M	64K	512	√			√	√			2	1ch+	6	4	3	√	需要A/D转换时才选用，8路8位精度		
STC89LE516X2		0-90M	64K	512	√		√	√	√			2	1ch+	6	4	3	√	在P1.0 - P1.7口，17 个机器周期一次		

RC/RD+ 系列为真正的看门狗，缺省为关闭（冷启动），启动后无法关闭，可放心省去外部看门狗。内部 Flash 擦写次数为 100,000 次以上，STC89C51RC/RD+ 系列单片机出厂时就已完全加密，无法解密。用户程序是用 ISP/IAP 机制写入，一边校验一边写，无读出命令，彻底无法解密。DIP-40，PLCC-44，PQFP-44 三种封装（PLCC、QFP 有 P4 口），RC/RD+ 系列单片机 P4 口地址为 E8H,并有 2 个附加外部中断，P4.2/INT3,P4.3/INT2。STC89LE516AD/X2 系列单片机 P4 口地址为 C0H,无附加外部中断。

STC89C51RC / RD+ 系列单片机 管脚图







封装尺寸大小详见 61 页

关于编译器 / 汇编器：

1.任何老的编译器 / 汇编器均可使用
Keil C51 中：Device 选择标准的 Intel8052
头文件包含标准的 <reg52.h>
2.新增特殊功能寄存器如要用到，则用
“sfr”及“sbit”声明地址即可
3.汇编中用“data”,或“EQU”声明地址

关于仿真及仿真器：

1.任何老的仿真器均可使用
2.老的仿真器仿真他可仿真的基本功能
3.新增特殊功能用 ISP 下载看结果即可
4.STC8051 专用仿真器也已推出，人民币 1950

关于工作电压 / 时钟频率:RC/RD+ 系列是真正的 6T 单片机，兼容普通的 12 时钟 / 机器周期

内核实际6T	现有老版5V单片机,无版本号					现有新版5V单片机（B版）					
工作电压	外部时钟	单倍速相当于普通8052	实际内核运行时钟	双倍速相当于普通8052	实际内核运行时钟	外部时钟	单倍速相当于普通8052	实际内核运行时钟	双倍速相当于普通8052	实际内核运行时钟	IAP/ISP可以
5.5V - 4.5V	0-24MHz	0-24MHz	0-12MHz	0-48MHz	0-24MHz	0-40MHz	0-40MHz	0-20MHz	0-80MHz	0-40MHz	读，编程，擦除
5.5V - 3.8V	0-20MHz	0-20MHz	0-10MHz	0-40MHz	0-20MHz	0-33MHz	0-33MHz	0-16.5M	0-66MHz	0-33MHz	读，编程，擦除
5.5V - 3.6V	0-18MHz	0-18MHz	0-9MHz	0-36MHz	0-18MHz	0-24MHz	0-24MHz	0-12MHz	0-48MHz	0-24MHz	读，编程，擦除
5.5V - 3.4V	0-12MHz	0-12MHz	0-6MHz	0-24MHz	0-12MHz	0-20MHz	0-20MHz	0-10MHz	0-40MHz	0-20MHz	读（不要编程/擦除）

3V：3.6 - 2.4V(可外部 24MHz,双倍速 48MHz),2.3-1.9V 时不要进行 IAP 擦除 / 编程

关于看门狗： RC/RD+ 系列为真正的看门狗，缺省为关闭（冷启动），启动后无法关闭。

A.看门狗溢出复位无法关看门狗(C 版)；
B.单片机软复位无法关看门狗(C版)
C.带电工作时，外部复位无法关看门狗(C 版)
D.软件无法关看门狗
E.外部干扰无法关看门狗
F.只有给单片机彻底断电，才可以

超低功耗 ---- STC89C51RC / RD+ 系列单片机

1.掉电模式：

典型功耗 < 0.1uA， 可由外部中断唤醒，中断返回后，继续执行原程序

2.空闲模式：

典型功耗 2mA

3.正常工作模式：

典型功耗 4mA - 7mA

3.掉电模式可由外部中断唤醒，适用于水表、气表等电池供电系统及便携设备

超强抗干扰 ---- STC89C51RC / RD+ 系列单片机

1．I/O口

输入/输出口经过特殊处理，很多干扰是从I/O进去的,每个I/O均有对VCC/对GND二级管箝位保护。

2．电源

单片机内部的电源供电系统经过特殊处理，很多干扰是从电源进去的

3．时钟

单片机内部的时钟电路经过特殊处理，很多干扰是从时钟部分进去的

4．看门狗

单片机内部的看门狗电路经过特殊处理，打开后无法关闭，可放心省去外部看门狗

5．复位电路

单片机内部的复位电路经过特殊处理，很多干扰是从复位电路部分进去的，STC89C51RC/RD+系列单片机为高电平复位。推荐外置复位电路为MAX810/STC810,STC6344,STC6345,813L,706P；也可用R/C复位，10uF电容/10k电阻,22uF/8.2k等。

新版本D版本内部含有专用复位电路，外部复位电路可以继续保留，也可以不用，不用时复位脚直接短到地。

6．宽电压，不怕电源抖动 5V：6v - 3.4v 3V：4v - 2.0v

7．高抗静电(高ESD保护)，轻松过4000V快速脉冲干扰(严格的日本及欧洲EFT标准)

降低单片机对外部的电磁辐射（EMI)---三大措施

1．禁止ALE时钟信号输出：

RC/RD+系列8051 单片机 扩展RAM管理及禁止ALE输出 特殊功能寄存器 只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

禁止ALE信号输出(应用示例供参考,汇编语言)：

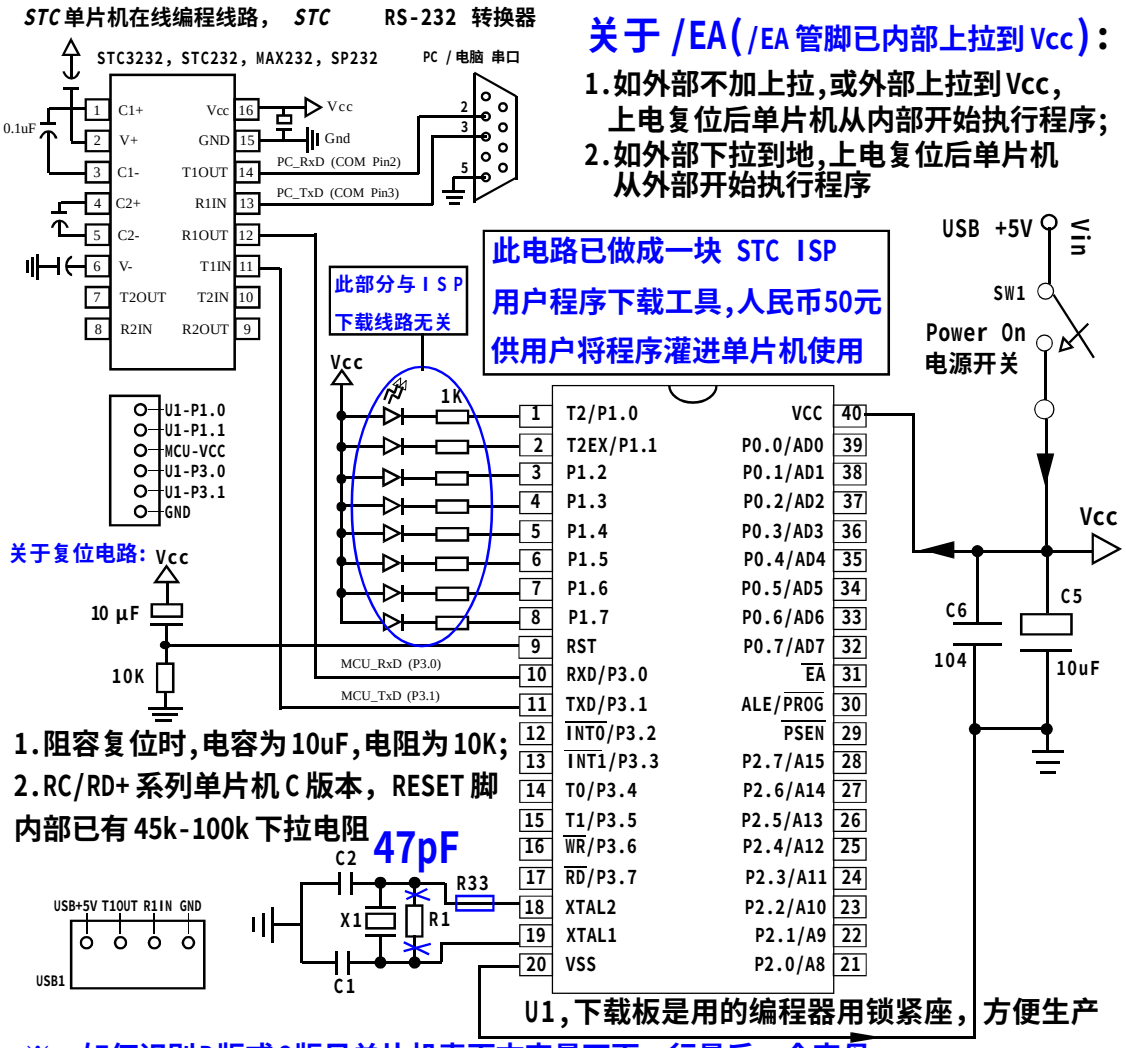
```
MOV AUXR, #00000001B; ALEOFF位置“1”,禁止ALE时钟输出
```

2．外部时钟频率降一半，6T模式：传统的8051为每个机器周期12时钟，如将STC的增强型8051单片机在ISP烧录程序时设为双倍速（即6T模式，每个机器周期6时钟），则可将单片机外部时钟频率降低一半，有效的降低单片机时钟对外界的辐射

3．单片机内部时钟振荡器增益降低一半： 在ISP烧录程序时将OSCDN设为1/2 gain

可以有效的降低单片机时钟高频部分对外界的辐射,单片机外部晶振频率<16MHz时，可将OSCDN设为1/2 gain,有利于降低EMI，16M以上选择 full gain。

STC 单片机 典型应用电路(89C51RC/RD+系列,C版)



关于 /EA(/EA 管脚已内部上拉到 Vcc):

1.如外部不加上拉,或外部上拉到Vcc,
上电复位后单片机从内部开始执行程序;
2.如外部下拉到地,上电复位后单片机
从外部开始执行程序

1.阻容复位时,电容为10uF,电阻为10K;
2.RC/RD+系列单片机C版本，RESET脚
内部已有 45k-100k 下拉电阻

U1,下载板是用的编程器用锁紧座，方便生产

※ 如何识别B版或C版见单片机表面文字最下面一行最后一个字母

关于晶振电路:

OSCDN,晶体振荡器增益控制 = full gain, R33 = 0 - 160欧姆(可以不用)						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<= 47pF	<=10pF	<=10pF	<=10pF	<=10pF	<=5pF
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K
针对C版 STC 89C51RC/RD+,STC89LE51RC/RD+ 系列单片机推荐晶体振荡电路						

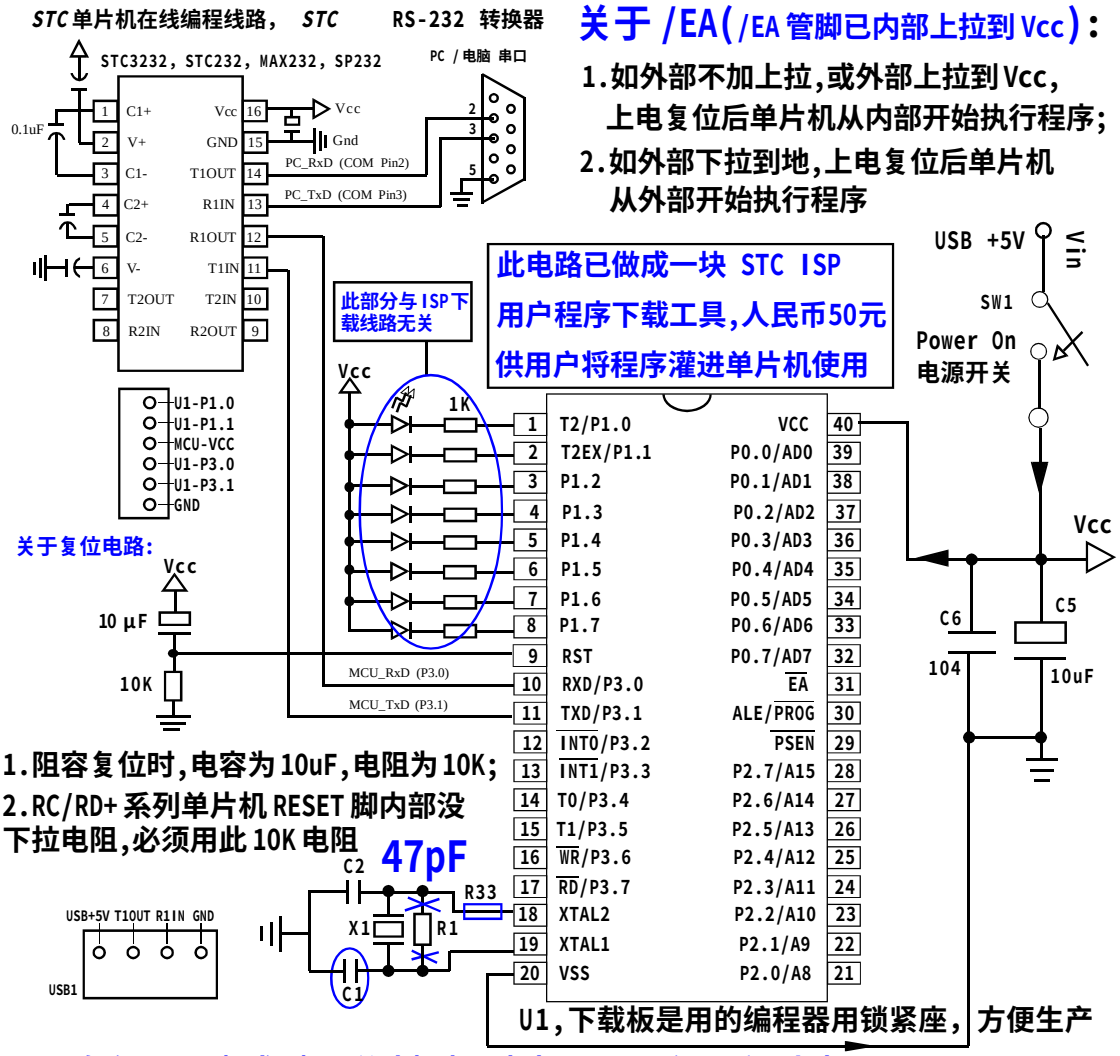
OSCDN(OSC Control),振荡器增益 = 1/2 gain, R33 = 0 - 160欧姆(可不用)						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<=47pF	<=5pF	不用	不用	不用	不用
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K
针对C版 STC 89C51RC/RD+,STC89LE51RC/RD+ 系列单片机推荐晶体振荡电路						

1.STC 单片机C版本的，R33 可以短路，也可以上 160 欧姆以下的电阻，C1 也可以不上。

推荐工作时钟频率(总线) STC单片机RC/RD+系列 (I/O方式可到40M/80M)	内部振荡器产生时钟，外接晶体		外部时钟直接输入，由XTAL1输入	
	12时钟模式	6时钟模式	12时钟模式	6时钟模式
5.0V 单片机	2MHz - 48MHz	2MHz - 36MHz	2MHz - 48MHz	2MHz - 36MHz
3.3V 单片机	2MHz - 48MHz	2MHz - 32MHz	2MHz - 36MHz	2MHz - 18MHz

STC ISP 编程典型应用电路(89C51RC/RD+ 系列,B 版)

--- 出厂就加密(超级加密)，利用 ISP 技术写入程序，无读出命令，无法解密



关于 /EA(/EA 管脚已内部上拉到 Vcc):

1.如外部不加上拉,或外部上拉到 Vcc，
上电复位后单片机从内部开始执行程序;

2.如外部下拉到地,上电复位后单片机
从外部开始执行程序

1.阻容复位时,电容为 10uF,电阻为 10K;

2.RC/RD+ 系列单片机 RESET 脚内部没下拉电阻,必须用此 10K 电阻

※ 如何识别B版或C版见单片机表面文字最下面一行最后一个字母

关于晶振电路:

OSCDN(OSC Control),晶体振荡器增益控制 = full gain, R33 = 160欧姆(附近)						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<=47pF	<=10pF	<=10pF	<=10pF	<=10pF	<=5pF
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K
针对A版/B版 STC 89C51RC/RD+,STC89LE51RC/RD+ 系列单片机推荐晶体振荡电路						

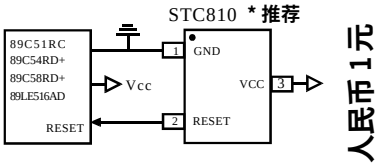
OSCDN(OSC Control),晶体振荡器增益控制 = 1/2 gain, R33 = 160欧姆(附近)						
X1	2-25MHz	26-30MHz	31-35MHz	36-39MHz	40-43MHz	44-48MHz
C1,C2	<=47pF	<=5pF	不用	不用	不用	不用
R1	不用	6.8K	5.1K	4.7K	3.3K	3.3K
针对A版/B版 STC 89C51RC/RD+,STC89LE51RC/RD+ 系列单片机推荐晶体振荡电路						

1.推荐在 XTAL2 管脚串一个 160 - 120 欧姆的电阻再到晶振的管脚,可限制晶振驱动电平,并有利于晶体起振，这对低频晶振尤其重要,如果这样 XTAL1 管脚旁的 C1 推荐上。

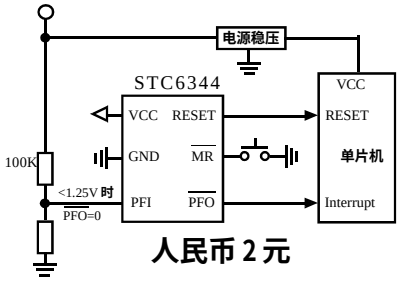
2.如果 XTAL2 管脚不串一个 160 欧姆的电阻的话,C1 不上，C2 上 47pF - 180pF。

典型 MCU/DSP/uC 复位、电源监控、外部看门狗专用电路

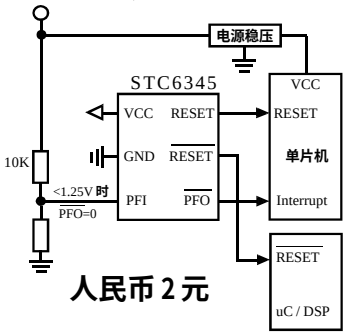
1.高电平复位信号输出



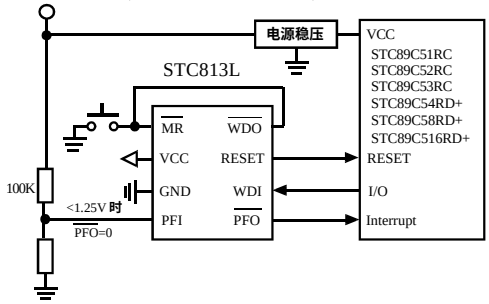
上电复位，掉电复位



上电复位，掉电复位,外部手动复位，稳压块前端掉电检测

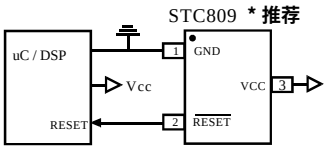


上电复位，掉电复位
电源稳压块前端掉电检测 ，
高 / 低电平 2 路复位信号输出

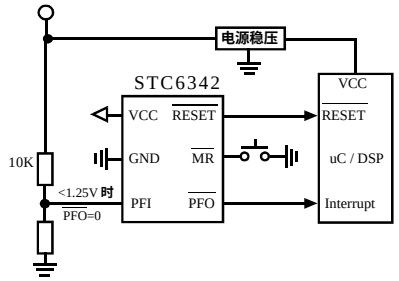


上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测 ，
外部硬件看门狗

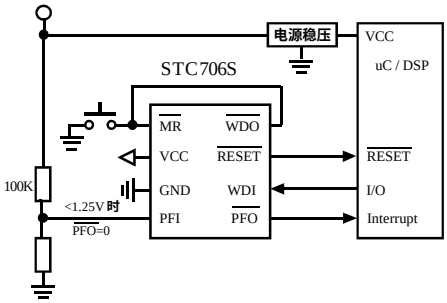
2.低电平复位信号输出



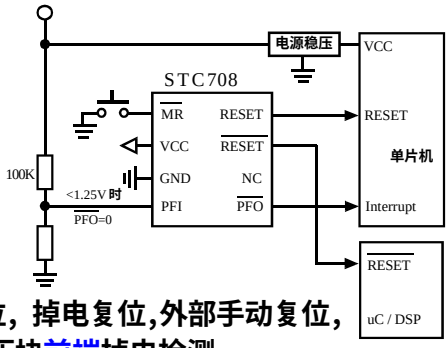
上电复位，掉电复位



上电复位，掉电复位,外部手动复位，稳压块前端掉电检测



上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测 ，
外部硬件看门狗

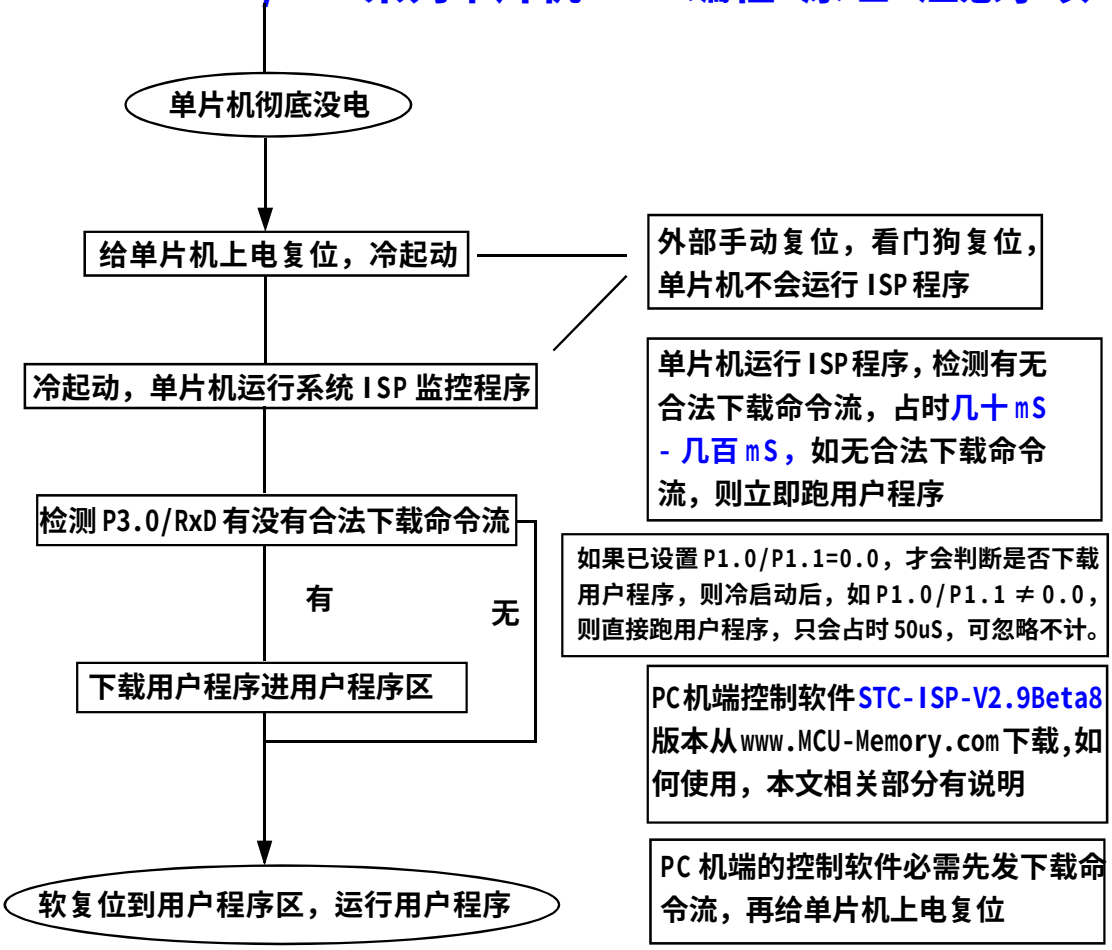


上电复位，掉电复位,外部手动复位，
电源稳压块前端掉电检测 ，
高 / 低电平 2 路复位信号输出

使用外部专用复位电路的好处：(推荐使用 STC6344，STC6345，STC810)

1．确保上电时，在用户设定的电源电压之上，时钟振荡稳定后，单片机才开始工作
2．确保掉电时，在用户设定的电源电压之下，立即让单片机复位，以免单片机误动作
3.具有电源稳压块前端掉电检测的专用复位电路，确保掉电前有充分的时间保存数据
4.复位门槛电压可选： L:4.63V; M:4.38V; J: 4.00V; T:3.08V; S:2.93V; R:2.63V

STC89C51RC/RD+ 系列单片机 ISP 编程 原理 注意事项



为什么有些用户下载程序不成功(在宏晶提供的下载板上)

1.可能电脑端的 STC-ISP 控制软件要升级，现须升级到 STC-ISP-2.9 Beta8
2.现在单片机端(下位机)ISP 软件是 3.2C,解决了少数电脑慢，通信连不上的问题.
3.运行用户程序时，可到 40M/80MHz,但 ISP 下载程序以前的版本软件只能到 33M/66MHz
4.少数客户的 PLCC-44,PQFP-44 转 DIP-40 的转换座走线过长，造成时钟振荡不稳定，下载不成功，可将 XTAL1 脚的电容去掉，XTAL2 脚的电容加大到 47pF 以上。
5.也有电脑 USB 电源供电不足的，可用万用表测一下，看 5V 部分是否在 4.5V 以上。
6.可能单片机内部没有 ISP 引导码，或 PC 串口波特率达不到 115200，选 57600 试一下
7.电脑端的 ISP 控制软件 STC-ISP-V2.9 测试版 8 加了一些功能，欢迎测试。
8.电脑端的 ISP 控制软件 STC-ISP-V2.6 测试版不要用，有误
9.新的单片机端(下位机)的 ISP 软件是 V3.2C,加了一些措施，主要解决冷起动运行 ISP 程序时间过长的问题，以免客户感觉“复位慢”，实为 ISP 程序在检测要不要下载程序。

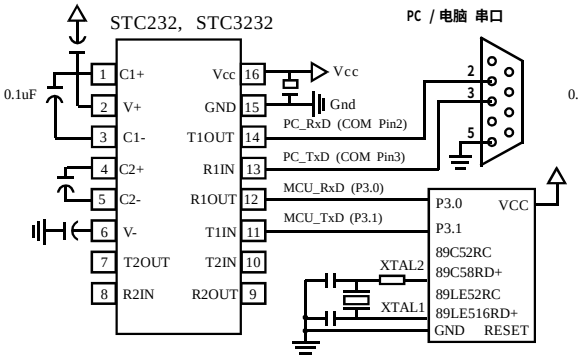
为什么有些用户下载程序不成功(在用户自己的系统上)

1.可能用户板上有外部看门狗，需不让其起作用，另要查时钟、复位是否正常。
2.可能用户板上 P3.0/RxD，P3.1/Txd 除了接 RS-232 转换器外，还接了 RS-485 等电路，需要将其断开。用户系统接了 RS-485 电路的，推荐在选项中选择下次冷启动时需 P1.0/P1.1=0.0 才判是否下载程序。

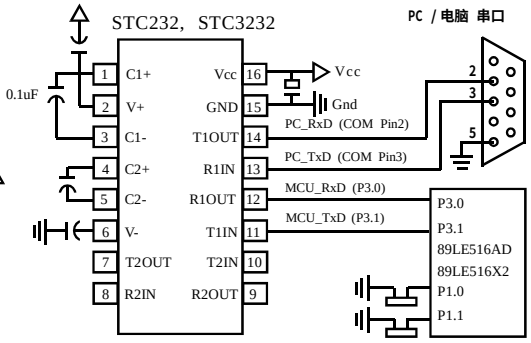
STC 89C51RC/RD+ 系列单片机在系统可编程的使用

--- 将用户代码下载进单片机内部，不用编程器

STC 单片机在线编程线路， *STC* RS-232 转换器



STC 单片机在线编程线路， *STC* RS-232 转换器



上面左图适用如下型号：

STC89C51RC， STC89C52RC， STC89C53RC

STC89LE51RC，STC89LE52RC，STC89LE53RC

STC89C54RD+，STC89C58RD+，STC89C516RD+

STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

STC89C516RD，STC89C58RD （老产品，不要选）

STC89LV516RD,STC89LV58RD （老产品，不要选）

上面右图适用如下型号：

STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD,STC89LE52AD,LE51AD

STC89 系列单片机大部分具有在系统可编程（ISP）特性，ISP 的好处是：省去购买通用编程器，单片机在用户系统上即可下载 / 烧录用户程序，而无须将单片机从已生产好的产品上拆下，再用通用编程器将程序代码烧录进单片机内部。有些程序尚未定型的产品可以一边生产，一边完善，加快了产品进入市场的速度，减小了新产品由于软件缺陷带来的风险。由于可以将程序直接下载进单片机看运行结果故也可以不用仿真器。

大部分 STC89 系列单片机在销售给用户之前已在单片机内部固化有 ISP 系统引导程序，配合 PC 端的控制程序即可将用户的程序代码下载进单片机内部，故无须编程器(速度比通用编程器快)。不要用通用编程器编程，否则有可能将单片机内部已固化的ISP系统引导程序擦除，造成无法使用STC提供的ISP软件下载用户的程序代码。

如何获得及使用 STC 提供的 ISP 下载工具（STC-ISP.exe 软件）：

(1). 获得 STC 提供的 ISP 下载工具（软件）

登陆 www.MCU-Memory.com 网站，从 STC 半导体专栏下载 PC（电脑）端的 ISP 程序，然后将其自解压，再安装即可（执行setup.exe），注意随时更新软件。

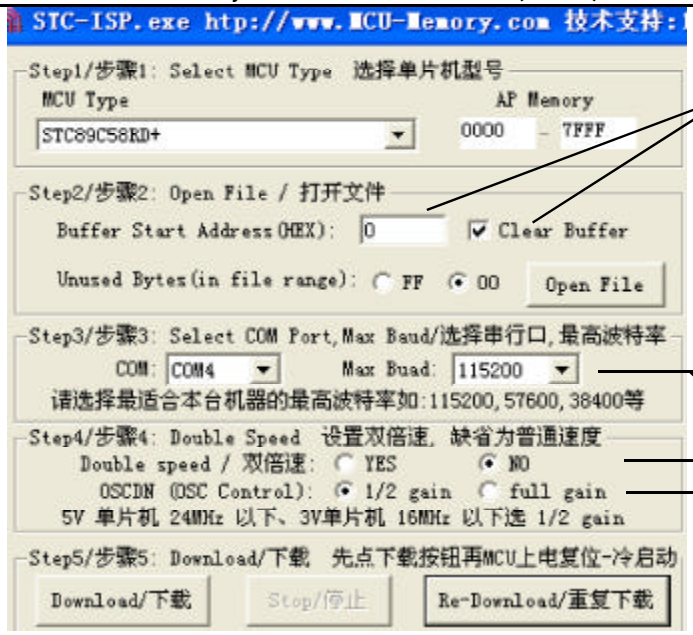
(2). 使用 STC-ISP 下载工具（软件），请随时更新，目前已到 Ver2.9 Beta8 版本(2005/7/15)，支持 *.Hex(Intel 16 进制格式)文件，RC/RD+系列单片机的底层软件版本为Ver3.2C(旧版可更换)。

请随时注意升级 PC（电脑）端的 ISP 程序，现 Ver2.9 Beta8 测试版欢迎测试。

单片机底层软件版本为 Ver3.2C 的单片机，PC（电脑）端的 ISP 程序应用 Ver2.9 Beta8 以上

(3).已经固化有 ISP 引导码，并设置为上电复位进入 ISP 的 STC89C51RC/RD+ 系列单片机出厂时就已完全加密，需要单片机内部的电放光后上电复位(冷起动)才运行系统 ISP 程序。

(4).可能用户板上 P3.0/RxD，P3.1/Txd 除了接 RS-232 转换器外，还接了 RS-485 等电路，需要将其断开。用户系统接了 RS-485 电路的，推荐在选项中选择下次冷启动时需 P1.0/P1.1=0.0 才判是否下载程序。



大批量生产时使用

第一次调文件进缓冲区，要清缓冲区。

要调几个文件进缓冲区，一次同时烧录进单片机，除每次均要指定缓冲区起始地址外，第二次及以后不能清缓冲区

用户根据需要限制最大通信波特率,如38400，9600

双倍速 / 单倍速设置，新的设置，停电后冷启动才生效

晶体振荡器增益：16MHz 以下可设为1/2 gain,有利于降低 EMI。

Step1/ 步骤 1：选择你所使用的单片机型号，如 STC89C58RD+，STC89LE516AD 等

Step2/ 步骤 2：打开文件，要烧录用户程序，必须调入用户的程序代码（*.bin，*.hex）

Step3/ 步骤 3：选择串行口，你所使用的电脑串口，如串行口 1--COM1，串行口 2--COM2,...

有些新式笔记本电脑没有 RS-232 串行口,可买一条 USB-RS232 转接器，人民币 70 元左右。

Step4/ 步骤 4：设置是否双倍速，双倍速选中 Double Speed 即可

STC89C51RC/RD+系列可以反复设置 双倍速 / 单倍速，新的设置停电后重新冷启动后才能生效

STC89LE516AD 为单倍速，STC89LE516X2 为双倍速,用户自己无法指定双倍速 / 单倍速

STC89C516RD 系列出厂时为单倍速，用户可指定设为双倍速，如想从双倍速恢复成单倍速，则需用通用编程器擦除整个晶片方可，这会将单片机内部已烧录的 ISP 引导程序擦除。一般使用缺省设置即可，无须设置。

OSCDN：单片机时钟振荡器增益降一半

选 1/2 gain 为降一半，降低 EMI；选 full gain（全增益）为正常状态。

Step5/ 步骤 5：选择“Download/ 下载”按钮下载用户的程序进单片机内部，可重复执行 Step5/ 步骤 5， 也可选择“Re-Download/ 重复下载”按钮

下载时注意看提示，主要看是否要给单片机上电或复位，下载速度比一般通用编程器快。

一般先选择“Download/ 下载”按钮，然后再给单片机上电复位(先彻底断电)，而不要先上电

关于硬件连接：

(1). MCU/ 单片机 RXD(P3.0) --- RS-232 转换器 --- PC/ 电脑 TXD(COM Port Pin3)

(2). MCU/ 单片机 TXD(P3.1) --- RS-232 转换器 --- PC/ 电脑 RXD(COM Port Pin2)

(3). MCU/ 单片机 GND ---------- PC/ 电脑 GND(COM Port Pin5)

(4). STC89LE516AD/X2 系列单片机冷启动时需要 P1.0，P1.1 = 0，0 ；上电复位才进入 ISP 模式,下载完后释放P1.0,P1.1，STC89LE516AD/X2运行用户程序.

STC89C51RC/RD+ 系列单片机不需要 P1.0,P1.1 = 0，0，但软件可选下次需要。

STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD 必需要 P1.0,P1.1 = 0，0

(5). RS-232 转换器可选用 SP232/MAX232/STC232(4.5-5.5V),SP3232/MAX3232/STC3232(3V-5.5V).

SP232/MAX232/STC232 尽量选用 SOP 封装(窄体,SP232EEN)，SP3232 尽量选用 SSOP 封装(SP3232EEA)

如用户系统没有 RS-232 接口，
可使用 STC-ISP Ver 2.0B.PCB 演示板作为编程工具

STC-ISP Ver 2.0B PCB 板可完成下载 / 烧录用户程序的功能。

在 STC-ISP Ver 2.0B PCB 板完成下载 / 烧录：

关于硬件连接：

(1.) 根据单片机的工作电压选择单片机电源电压

A. 5V 单片机,短接 JP1 的 MCU-VCC，5V 电源管脚

B. 3V 单片机,短接 JP1 的 MCU-VCC，3.3V 电源管脚

(2.) 根据单片机的工作电压选择复位信号

A. 5V 单片机,短接 JP2 的 MCU-RST，5V/MCU-RST 信号管脚

B. 3.3V 单片机,短接 JP2 的 MCU-RST，3.3V/MCU-RST 信号管脚

(3.)连接线(宏晶提供)

A. 将一端有 9 芯连接座的插头插入 PC/ 电脑 RS-232 串行接口插座用于通信

B. 将同一端的 USB 插头插入 PC/ 电脑 USB 接口用于取电

C. 将只有一个 USB 插头的一端插入宏晶的 STC-ISP Ver 2.0B PCB 板 USB1 插座用于 RS-232 通信和供电,此时 +5V Power 灯亮(D10,USB 接口有电)

(4.)其他插座不需连接

(5.)SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D9)，没有给单片机通电

(6.)SW3 开关

处于非按下状态，P1.0，P1.1 = 1，1,不短接到地。

适用于：STC89C51RC / RD+, /RD 系列单片机

处于按下状态， P1.0，P1.1 = 0，0,短接到地。

适用于：STC89LE516AD，STC89LE58AD，STC89LE54AD，STC89LE516X2

(7.)将单片机插进 U1-Socket 锁紧座，锁紧单片机

(8.)关于软件：选择“Download/ 下载”(必须在给单片机上电之前让 PC 先发一串合法下载命令)

(9.)按下 SW1 开关，给单片机上电复位，此时 MCU-VCC Power 灯亮(D9)

此时 STC 单片机进入 ISP 模式(STC89C51RC/RD+ 系列冷启动进入 ISP)

(10.)下载成功后，再按 SW1 开关，此时 SW1 开关处于非按下状态，MCU-VCC Power 灯不亮(D9)，给单片机断电，取下单片机。

利用 STC-ISP Ver 2.0B PCB 板进行 RS-232 转换
单片机在用户自己的板上完成下载 / 烧录：

1. U1-Socket 锁紧座不得插入单片机
2. 将用户系统上的电源(MCU-VCC,GND)及单片机的 P3.0/RXD,P3.1/TXD 接入转换板 CN2 插座
   这样用户系统上的单片机就具备了与 PC/ 电脑进行通信的能力
3. 将用户系统的单片机的 P1.0,P1.1 接入转换板 CN2 插座(仅 STC89LE516AD/X2 系列需要)
4. SW3 开关处于按下状态，P1.0，P1.1 = 0，0,短接到地。仅 STC89LE516AD/X2 系列需要
5. 关于软件：选择“Download/ 下载”
6. 给单片机系统上电复位(注意是从用户系统自供电，不要从电脑 USB 取电,电脑 USB 座不插)
7. 下载程序时，如用户板有外部看门狗电路，不得启动，单片机必须有正确的复位,但不能在 ISP 下载程序时被外部看门狗复位,可将外部看门狗电路 WDI 端 / 或 WDO 端浮空
8. 如有 RS-485 晶片连到 P3.0/Rxd,P3.1/Txd,在下载时应将其断开。

特殊功能寄存器映像 SFR Mapping

STC89C51RC, STC89C52RC, STC89C53RC, STC89C54RD+,STC89C58RD+,STC89C516RD+
STC89LE51RC,STC89LE52RC,STC89LE53RC,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h									FFh
F0h	B 0000,0000								F7h
E8h	P4 xxxx,1111								EFh
E0h	ACC 0000,0000	WDT_CONTR xx00,0000	ISP_DATA 1111,1111	ISP_ADDRH 0000,0000	ISP_ADDRL 0000,0000	ISP_CMD 1111,1000	ISP_TRIG xxxx,xxxx	ISP_CONTR 000x,x000	E7h
D8h									DFh
D0h	PSW 0000,0000								D7h
C8h	T2CON 0000,0000	T2MOD xxxx,xx00	RCAP2L 0000,0000	RCAP2H 0000,0000	TL2 0000,0000	TH2 0000,0000			CFh
C0h	XICON 0000,0000								C7h
B8h	IP xx00,0000	SADEN 0000,0000							BFh
B0h	P3 1111,1111							IPH 0000,0000	B7h
A8h	IE 0000,0000	SADDR 0000,0000							AFh
A0h	P2 1111,1111		AUXR1 xxxx,0xx0						A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111								97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR xxxx,xx00		8Fh
80h	P0 1111,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000				PCON 00x1,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

RC/RD+系列8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

RC/RD+系列8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	-	POF	GF1	GF0	PD	IDL	00x1,0000
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00
AUXR1	A2h	Auxiliary Register 1	-	-	-	-	GF2	-	-	DPS	xxxx,0xx0

不同：STC89LE516AD / 89LE516X2 系列单片机没有 EXTRAM 控制位.

RC/RD+系列8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
XICON	C0h	Auxiliary Interrupt Control	PX3	EX3	IE3	IT3	PX2	EX2	IE2	IT2	0000,0000

不同：STC89LE516AD 系列单片机没有(XICON,PX3H,PX2H),因为 P4.2/P4.3 无中断.

RC/RD+系列8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	1111,1111
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P3	B0h	8-bit Port 3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1111,1111
P4	E8h	4-bit Port 4	-	-	-	-	P4.3	P4.2	P4.1	P4.0	xxxx,1111

不同：STC89LE516AD / 89LE516X2 系列单片机 P4 口地址为 C0h,而不是 E8h.

RC/RD+系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000

RC/RD+系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
T2CON	C8h	Timer / Counter 2 Control	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#	0000,0000
T2MOD	C9h	Timer / Counter 2 Mode	-	-	-	-	-	-	T2OE	DCEN	xxxx,xx00
RCAP2L	CAh	Timer / Counter 2 Reload/Capture Low Byte									0000,0000
RCAP2H	CBh	Timer / Counter 2 Reload/Capture High Byte									0000,0000
TL2	CCh	Timer / Counter 2 Low Byte									0000,0000
TH2	CDh	Timer / Counter 2 High Byte									0000,0000

RC/RD+系列 8051单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	-	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

RC/RD+系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

降低单片机对系统的电磁干扰（EMI)--- 三大措施

1. 禁止 ALE 信号输出，适用型号：

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC

STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

STC89LE516AD / X2 系列 （注：此系列单片机无 EXTRAM 控制位）

RC/RD+系列 8051 单片机 扩展 RAM 管理及禁止 ALE 输出 特殊功能寄存器 只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

禁止 ALE 信号输出(应用示例供参考,C 语言)：

```
sfr     AUXR =   0x8e;     /* 声明AUXR 寄存器的地址 */
AUXR =   0x01;
/* ALEOFF位置1,禁止ALE信号输出,提升系统的EMI性能,复位后为0,ALE信号正常输出 */
```

禁止 ALE 信号输出(应用示例供参考,汇编语言)：

```
AUXR  EQU  8Eh  ;      或      AUXR  DATA  8Eh
MOV   AUXR,#00000001B;  ALEOFF 位置“1”,禁止 ALE 信号输出,提升了系统的 EMI 性能
```

2. 外部时钟频率降一半，6T 模式：传统的 8051 为每个机器周期 12 时钟，如将 STC 的增强型 8051 单片机在 ISP 烧录程序时设为双倍速（及 6T 模式，每个机器周期 6 时钟），则可将单片机外部时钟频率降低一半，有效的降低单片机时钟对外界的干扰

3. 单片机内部时钟振荡器增益降低一半：在 ISP 烧录程序时将 OSCDN 设为 1/2 gain 可以有效的降低单片机时钟高频部分对外界的辐射,但此时外部晶振频率尽量不要高于24MHz。

STC89C51RC/RD+ 系列单片机扩展 RAM 的使用
STC89C51RC/RD+ 系列单片机扩展 RAM 的禁止

适用型号:

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC

STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

普通89C51,89C52系列单片机的内部RAM只有128(89C51)/256(89C52)供用户使用

(1).低128字节的内部RAM（地址:00H-7FH),可直接寻址或间接寻址,(data/idata)

(2).高128字节的内部RAM（地址:80H-FFH),只能间接寻址(普通89C51没有),(idata)

(3).特殊功能寄存器SFR（地址:80H-FFH)，只能直接寻址,(data)

特殊功能寄存器SFR和高128字节的内部RAM是通过寻址方式来区分的,传统的8051系列单片机只有128-256字节RAM供用户使用，在此情况下STC公司响应广大用户的呼声，在一些单片机内部增加了扩展RAM。 STC89C58RD+系列单片机扩展了1024个字节RAM，共1280字节RAM；STC89C52RC系列扩展了256个字节RAM,共512字节RAM。访问内部扩展RAM时，不影响P0口/P2口/P3.6/P3.7。

RC/RD+系列8051 单片机 扩展RAM管理及禁止ALE输出 特殊功能寄存器 只写

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

Symbol符号 Function功能

EXTRAM Internal/External RAM access 内部/外部 RAM 存取

0: 内部扩展的EXT_RAM可以存取.

RD+系列单片机

在00H到3FFH单元(1024字节),使用MOVX @DPTR指令访问,超过400H的地址空间总是访问外部数据存储器（含400H单元)，MOVX @Ri只能访问00H到FFH单元

RC系列单片机

在00H到FFH单元(256字节),使用MOVX @DPTR指令访问,超过100H的地址空间总是访问外部数据存储器（含100H单元)，MOVX @Ri只能访问00H到FFH单元

1: External data memory access.

外部数据存储器存取,禁止访问内部扩展RAM，此时MOVX @DPTR / MOVX @Ri的使用同普通8052单片机

ALEOFF Disable/enable ALE.

0: ALE is emitted at a constant rate of 1/3 the oscillator frequency in 6 clock mode, 1/6 fosc in 12 clock mode

ALE脚输出固定的1/6晶振频率信号在12时钟模式时,在6时钟模式时输出固定的1/3晶振频率信号.

1: ALE is active only during a MOVX or MOVC instruction.

ALE脚仅在执行MOVX or MOVC指令时才输出信号,好处是：降低了系统对外界的EMI.

注解：STC89LE516AD,STC89LE516X2 系列无EXTRAM控制位,仅有ALEOFF控制位。

STC89LE516AD/X2系列用 MOVX A，@Ri / MOVX @Ri，A指令固定访问内部扩展的EXTRAM，用 MOVX A，@DPTR / MOVX @DPTR，A指令固定访问外部RAM.

应用示例供参考（汇编）：

访问内部扩展的 EXTRAM

```
;新增特殊功能寄存器声明(汇编方式)
AUXR    DATA    8EH;        或者用    AUXR  EQU   8EH   定义
MOV     AUXR,   #00000000B;  EXTRAM 位清为 "0", 其实上电复位时此位就为 "0".
    ;MOVX  A,  @DPTR / MOVX  @DPTR, A 指令可访问内部扩展的 EXTRAM
        ;RD+ 系列为(00H - 3FFH,共 1024 字节)
        ;RC 系列为(00H - FFH,共 256 字节)
    ;MOVX  A,  @Ri / MOVX  A, @Ri 指令可直接访问内部扩展的EXTRAM
        ;使用此指令 RD+ 系列 只能访问内部扩展的 EXTRAM(00H - FFH,共 256 字节)
;写芯片内部扩展的 EXTRAM
    MOV     DPTR,   #address
    MOV     A,  #value
    MOVX    @DPTR,  A
;读芯片内部扩展的 EXTRAM
    MOV     DPTR,   #address
    MOVX    A,  @DPTR
RD+ 系列
; 如果 #address < 400H，则在 EXTRAM 位为 "0" 时，访问物理上在内部，逻辑上在外部的
      此 EXTRAM
; 如果 #address >= 400H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（400H--FFFFH）
RC 系列
; 如果 #address < 100H，则在 EXTRAM 位为 "0" 时，访问物理上在内部，逻辑上在外部的
      此 EXTRAM
; 如果 #address >= 100H，则总是访问物理上外部扩展的 RAM 或 I/O 空间（100H--FFFFH）
```

禁止访问内部扩展的 EXTRAM ，以防冲突

```
MOV     AUXR,   #00000010B;  EXTRAM 控制位设置为 "1", 禁止访问 EXTRAM,以防冲突
```

有些用户系统因为外部扩展了 I/O 或者用片选去选多个 RAM 区,有时与此内部扩展的 EXTRAM 逻辑地址上有冲突，将此位设置为 "1", 禁止访问此内部扩展的 EXTRAM 就可以了.

大实话： 其实不用设置 AUXR 寄存器即可直接用 MOVX @DPTR 指令访问此内部扩展的 EXTRAM,超过此 RAM空间,将访问片外单元.如果系统外扩了SRAM,而实际使用的空间小于1024/256字节,则可直接将此 SRAM 省去,比如省去 STC62WV256,IS62C256,UT6264 等. 另外尽量用 MOVX A, @Ri / MOVX @Ri, A 指令访问此内部扩展的 EXTRAM,这样只能访问 256 字节的扩展 EXTRAM,但可与很多单片机兼容。如STC89LE516AD/X2系列MOVX A, @Ri / MOVX @Ri, A指令只能固定访问内部扩展的EXTRAM, MOVX A, @DPTR / MOVX @DPTR, A 指令固定访问外部 RAM.

应用示例供参考（C 语言）：

```
/*  访问内部扩展的 EXTRAM */
/*  RD+ 系列为(00H - 3FFH，共 1024 字节扩展的 EXTRAM) */
/*  RC 系列为(00H -  FFH，共 256 字节扩展的 EXTRAM)  */
    /*  新增特殊功能寄存器声明(C 语言方式)  */
    sfr AUXR    =   0x8e    /*  如果不需设置 AUXR 就不用声明 AUXR     */
    AUXR    =   0x00;   /*  0000,0000  EXTRAM 位清 0，其实上电复位时此位就为 0 */
    unsigned char  xdata  sum,  loop_counter, test_array[128];
    /*  将变量声明成 xdata 即可直接访问此内部扩展的 EXTRAM  */
```

```
/*  写芯片内部扩展的EXTRAM */
        sum =   0;
        loop_counter    =   128;
        test_array[0]   =   5;
/*  读芯片内部扩展的EXTRAM */
        sum =   test_array[0];
    /*  RD+系列:
                如果 #address < 400H，则在EXTRAM位为”0”时，访问物理上在内部，逻辑
                     上在外部的此EXTRAM
                如果#address>=400H，则总是访问物理上外部扩展的RAM或I/O空间（400H-FFFFH）
        RC系列:
                如果 #address < 100H，则在EXTRAM位为”0”时，访问物理上在内部，逻辑
                     上在外部的此EXTRAM
                如果#address>=100H，总是访问物理上外部扩展的RAM或I/O空间（100H--FFFFH）
    */
```

禁止访问内部扩展的EXTRAM，以防冲突

```
        AUXR    =   0x02;   /*  0000,0010，EXTRAM位设为”1”，禁止访问EXTRAM,以防冲突 */
```

有些用户系统因为外部扩展了I/O 或者用片选去选多个RAM 区,有时与此内部扩展的EXTRAM逻辑上有冲突，将此位设置为”1”， 禁止访问此内部扩展的EXTRAM 就可以了.

AUXR是只写寄存器

所谓只写，就是直接用“MOV AUXR， #data”去写,而不要用含读的操作如“或，与，入栈”因为他不让你读，如去读，读出的数值不确定，用含读的操作如“或，与，入栈”，会达不到需要的效果。

单片机C版本和以前版本的区别（关于内部扩展RAM）

传统的8051，内部无扩展RAM，而STC89C51RC/RD+系列单片机内部均已扩展了RAM,少数客户的老产品P0/P2是作为总线用的而不是作为普通I/O口用，有些需要用软件关闭此内部扩展RAM。而客户的源程序早已遗失，或开发工程师早已离职，所以STC89C51RC/RD+系列单片机为了解决此问题，推出C版本以供用户在ISP下载程序时就可选择关闭此内部扩展RAM，以达到完全兼容以前的老产品的目的。

一般不要在ISP下载程序时就选择关闭此内部扩展RAM，因为流行用法是复位后缺省是允许访问扩展RAM,复位后AUXR.1/AUXR.EXTRAM = 0，选择关闭此内部扩展RAM,则本来是：

	在ISP下载程序时选择“允许访问内部扩展RAM”	在ISP下载程序时选择“禁止访问内部扩展RAM”
AUXR.1/AUXR.EXTRAM = 0	是允许访问内部扩展RAM	是禁止访问内部扩展RAM
AUXR.1/AUXR.EXTRAM = 1	是禁止访问内部扩展RAM	是允许访问内部扩展RAM

另STC89C51RC/RD+系列单片机C版本以前的单片机“AUXR寄存器是只写特性”,现C版本及以后的版本将都是既可以读又可以写。

STC89C58RD+系列单片机内部扩展RAM演示程序

```
/* ---  STC International Limited ---------------- */
/* ---  Web : www.mcu-memory.com  ---------------- */
/* ---  xdata test ------------------------------- */
/* ---  宏晶科技 姚永平  设计 2005/6/1   ---------- */
/* ---  STC89C54RD+, STC89C58RD+, STC89C516RD+ --- */
/* ---  STC89LE54RD+,STC89LE58RD+,STC89LE516RD+ -- */
/* ---  Mobile: 13922805190 ---------------------- */
/* ---  Fax: 0755-82944243 ----------------------- */
/* ---  Tel: 0755-82908285 ----------------------- */
/* ---  Web  : www.mcu-memory.com ---------------- */

#include <reg52.h>
#include <intrins.h>        /* use _nop_() function */

sfr AUXR = 0x8e;
sfr AUXR1 = 0xa2;

sfr P4 = 0xe8;
sfr XICON = 0xc0;

sfr IPH = 0xb7;

sfr WDT_CONTR = 0xe1;
sfr ISP_DATA = 0xe2;
sfr ISP_ADDRH = 0xe3;
sfr ISP_ADDRL = 0xe4;
sfr ISP_CMD = 0xe5;
sfr ISP_TRIG = 0xe6;
sfr ISP_CONTR = 0xe7;

sbit ERROR_LED = P1^5;
sbit OK_LED = P1^7;

void main()
{
    unsigned int array_point = 0;

    /* 测试数组 Test_array_one[512],Test_array_two[512]*/
    unsigned char xdata Test_array_one[512]        =
    {
        0x00,     0x01,     0x02,     0x03,     0x04,     0x05,     0x06,     0x07,
        0x08,     0x09,     0x0a,     0x0b,     0x0c,     0x0d,     0x0e,     0x0f,
        0x10,     0x11,     0x12,     0x13,     0x14,     0x15,     0x16,     0x17,
        0x18,     0x19,     0x1a,     0x1b,     0x1c,     0x1d,     0x1e,     0x1f,
        0x20,     0x21,     0x22,     0x23,     0x24,     0x25,     0x26,     0x27,
```

```
0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f,
0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37,
0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f,
0x40,   0x41,   0x42,   0x43,   0x44,   0x45,   0x46,   0x47,
0x48,   0x49,   0x4a,   0x4b,   0x4c,   0x4d,   0x4e,   0x4f,
0x50,   0x51,   0x52,   0x53,   0x54,   0x55,   0x56,   0x57,
0x58,   0x59,   0x5a,   0x5b,   0x5c,   0x5d,   0x5e,   0x5f,
0x60,   0x61,   0x62,   0x63,   0x64,   0x65,   0x66,   0x67,
0x68,   0x69,   0x6a,   0x6b,   0x6c,   0x6d,   0x6e,   0x6f,
0x70,   0x71,   0x72,   0x73,   0x74,   0x75,   0x76,   0x77,
0x78,   0x79,   0x7a,   0x7b,   0x7c,   0x7d,   0x7e,   0x7f,
0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87,
0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f,
0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97,
0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f,
0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7,
0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf,
0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7,
0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf,
0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7,
0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf,
0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7,
0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf,
0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7,
0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef,
0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7,
0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
0xff,   0xfe,   0xfd,   0xfc,   0xfb,   0xfa,   0xf9,   0xf8,
0xf7,   0xf6,   0xf5,   0xf4,   0xf3,   0xf2,   0xf1,   0xf0,
0xef,   0xee,   0xed,   0xec,   0xeb,   0xea,   0xe9,   0xe8,
0xe7,   0xe6,   0xe5,   0xe4,   0xe3,   0xe2,   0xe1,   0xe0,
0xdf,   0xde,   0xdd,   0xdc,   0xdb,   0xda,   0xd9,   0xd8,
0xd7,   0xd6,   0xd5,   0xd4,   0xd3,   0xd2,   0xd1,   0xd0,
0xcf,   0xce,   0xcd,   0xcc,   0xcb,   0xca,   0xc9,   0xc8,
0xc7,   0xc6,   0xc5,   0xc4,   0xc3,   0xc2,   0xc1,   0xc0,
0xbf,   0xbe,   0xbd,   0xbc,   0xbb,   0xba,   0xb9,   0xb8,
0xb7,   0xb6,   0xb5,   0xb4,   0xb3,   0xb2,   0xb1,   0xb0,
0xaf,   0xae,   0xad,   0xac,   0xab,   0xaa,   0xa9,   0xa8,
0xa7,   0xa6,   0xa5,   0xa4,   0xa3,   0xa2,   0xa1,   0xa0,
0x9f,   0x9e,   0x9d,   0x9c,   0x9b,   0x9a,   0x99,   0x98,
0x97,   0x96,   0x95,   0x94,   0x93,   0x92,   0x91,   0x90,
0x8f,   0x8e,   0x8d,   0x8c,   0x8b,   0x8a,   0x89,   0x88,
0x87,   0x86,   0x85,   0x84,   0x83,   0x82,   0x81,   0x80,
0x7f,   0x7e,   0x7d,   0x7c,   0x7b,   0x7a,   0x79,   0x78,
0x77,   0x76,   0x75,   0x74,   0x73,   0x72,   0x71,   0x70,
0x6f,   0x6e,   0x6d,   0x6c,   0x6b,   0x6a,   0x69,   0x68,
0x67,   0x66,   0x65,   0x64,   0x63,   0x62,   0x61,   0x60,
0x5f,   0x5e,   0x5d,   0x5c,   0x5b,   0x5a,   0x59,   0x58,
```

```
    0x57,   0x56,   0x55,   0x54,   0x53,   0x52,   0x51,   0x50,
    0x4f,   0x4e,   0x4d,   0x4c,   0x4b,   0x4a,   0x49,   0x48,
    0x47,   0x46,   0x45,   0x44,   0x43,   0x42,   0x41,   0x40,
    0x3f,   0x3e,   0x3d,   0x3c,   0x3b,   0x3a,   0x39,   0x38,
    0x37,   0x36,   0x35,   0x34,   0x33,   0x32,   0x31,   0x30,
    0x2f,   0x2e,   0x2d,   0x2c,   0x2b,   0x2a,   0x29,   0x28,
    0x27,   0x26,   0x25,   0x24,   0x23,   0x22,   0x21,   0x20,
    0x1f,   0x1e,   0x1d,   0x1c,   0x1b,   0x1a,   0x19,   0x18,
    0x17,   0x16,   0x15,   0x14,   0x13,   0x12,   0x11,   0x10,
    0x0f,   0x0e,   0x0d,   0x0c,   0x0b,   0x0a,   0x09,   0x08,
    0x07,   0x06,   0x05,   0x04,   0x03,   0x02,   0x01,   0x00
};

unsigned char xdata Test_array_two[512]       =
{
    0x00,   0x01,   0x02,   0x03,   0x04,   0x05,   0x06,   0x07,
    0x08,   0x09,   0x0a,   0x0b,   0x0c,   0x0d,   0x0e,   0x0f,
    0x10,   0x11,   0x12,   0x13,   0x14,   0x15,   0x16,   0x17,
    0x18,   0x19,   0x1a,   0x1b,   0x1c,   0x1d,   0x1e,   0x1f,
    0x20,   0x21,   0x22,   0x23,   0x24,   0x25,   0x26,   0x27,
    0x28,   0x29,   0x2a,   0x2b,   0x2c,   0x2d,   0x2e,   0x2f,
    0x30,   0x31,   0x32,   0x33,   0x34,   0x35,   0x36,   0x37,
    0x38,   0x39,   0x3a,   0x3b,   0x3c,   0x3d,   0x3e,   0x3f,
    0x40,   0x41,   0x42,   0x43,   0x44,   0x45,   0x46,   0x47,
    0x48,   0x49,   0x4a,   0x4b,   0x4c,   0x4d,   0x4e,   0x4f,
    0x50,   0x51,   0x52,   0x53,   0x54,   0x55,   0x56,   0x57,
    0x58,   0x59,   0x5a,   0x5b,   0x5c,   0x5d,   0x5e,   0x5f,
    0x60,   0x61,   0x62,   0x63,   0x64,   0x65,   0x66,   0x67,
    0x68,   0x69,   0x6a,   0x6b,   0x6c,   0x6d,   0x6e,   0x6f,
    0x70,   0x71,   0x72,   0x73,   0x74,   0x75,   0x76,   0x77,
    0x78,   0x79,   0x7a,   0x7b,   0x7c,   0x7d,   0x7e,   0x7f,
    0x80,   0x81,   0x82,   0x83,   0x84,   0x85,   0x86,   0x87,
    0x88,   0x89,   0x8a,   0x8b,   0x8c,   0x8d,   0x8e,   0x8f,
    0x90,   0x91,   0x92,   0x93,   0x94,   0x95,   0x96,   0x97,
    0x98,   0x99,   0x9a,   0x9b,   0x9c,   0x9d,   0x9e,   0x9f,
    0xa0,   0xa1,   0xa2,   0xa3,   0xa4,   0xa5,   0xa6,   0xa7,
    0xa8,   0xa9,   0xaa,   0xab,   0xac,   0xad,   0xae,   0xaf,
    0xb0,   0xb1,   0xb2,   0xb3,   0xb4,   0xb5,   0xb6,   0xb7,
    0xb8,   0xb9,   0xba,   0xbb,   0xbc,   0xbd,   0xbe,   0xbf,
    0xc0,   0xc1,   0xc2,   0xc3,   0xc4,   0xc5,   0xc6,   0xc7,
    0xc8,   0xc9,   0xca,   0xcb,   0xcc,   0xcd,   0xce,   0xcf,
    0xd0,   0xd1,   0xd2,   0xd3,   0xd4,   0xd5,   0xd6,   0xd7,
    0xd8,   0xd9,   0xda,   0xdb,   0xdc,   0xdd,   0xde,   0xdf,
    0xe0,   0xe1,   0xe2,   0xe3,   0xe4,   0xe5,   0xe6,   0xe7,
    0xe8,   0xe9,   0xea,   0xeb,   0xec,   0xed,   0xee,   0xef,
    0xf0,   0xf1,   0xf2,   0xf3,   0xf4,   0xf5,   0xf6,   0xf7,
    0xf8,   0xf9,   0xfa,   0xfb,   0xfc,   0xfd,   0xfe,   0xff,
    0xff,   0xfe,   0xfd,   0xfc,   0xfb,   0xfa,   0xf9,   0xf8,
```

```
        0xf7,     0xf6,     0xf5,     0xf4,     0xf3,     0xf2,     0xf1,     0xf0,
        0xef,     0xee,     0xed,     0xec,     0xeb,     0xea,     0xe9,     0xe8,
        0xe7,     0xe6,     0xe5,     0xe4,     0xe3,     0xe2,     0xe1,     0xe0,
        0xdf,     0xde,     0xdd,     0xdc,     0xdb,     0xda,     0xd9,     0xd8,
        0xd7,     0xd6,     0xd5,     0xd4,     0xd3,     0xd2,     0xd1,     0xd0,
        0xcf,     0xce,     0xcd,     0xcc,     0xcb,     0xca,     0xc9,     0xc8,
        0xc7,     0xc6,     0xc5,     0xc4,     0xc3,     0xc2,     0xc1,     0xc0,
        0xbf,     0xbe,     0xbd,     0xbc,     0xbb,     0xba,     0xb9,     0xb8,
        0xb7,     0xb6,     0xb5,     0xb4,     0xb3,     0xb2,     0xb1,     0xb0,
        0xaf,     0xae,     0xad,     0xac,     0xab,     0xaa,     0xa9,     0xa8,
        0xa7,     0xa6,     0xa5,     0xa4,     0xa3,     0xa2,     0xa1,     0xa0,
        0x9f,     0x9e,     0x9d,     0x9c,     0x9b,     0x9a,     0x99,     0x98,
        0x97,     0x96,     0x95,     0x94,     0x93,     0x92,     0x91,     0x90,
        0x8f,     0x8e,     0x8d,     0x8c,     0x8b,     0x8a,     0x89,     0x88,
        0x87,     0x86,     0x85,     0x84,     0x83,     0x82,     0x81,     0x80,
        0x7f,     0x7e,     0x7d,     0x7c,     0x7b,     0x7a,     0x79,     0x78,
        0x77,     0x76,     0x75,     0x74,     0x73,     0x72,     0x71,     0x70,
        0x6f,     0x6e,     0x6d,     0x6c,     0x6b,     0x6a,     0x69,     0x68,
        0x67,     0x66,     0x65,     0x64,     0x63,     0x62,     0x61,     0x60,
        0x5f,     0x5e,     0x5d,     0x5c,     0x5b,     0x5a,     0x59,     0x58,
        0x57,     0x56,     0x55,     0x54,     0x53,     0x52,     0x51,     0x50,
        0x4f,     0x4e,     0x4d,     0x4c,     0x4b,     0x4a,     0x49,     0x48,
        0x47,     0x46,     0x45,     0x44,     0x43,     0x42,     0x41,     0x40,
        0x3f,     0x3e,     0x3d,     0x3c,     0x3b,     0x3a,     0x39,     0x38,
        0x37,     0x36,     0x35,     0x34,     0x33,     0x32,     0x31,     0x30,
        0x2f,     0x2e,     0x2d,     0x2c,     0x2b,     0x2a,     0x29,     0x28,
        0x27,     0x26,     0x25,     0x24,     0x23,     0x22,     0x21,     0x20,
        0x1f,     0x1e,     0x1d,     0x1c,     0x1b,     0x1a,     0x19,     0x18,
        0x17,     0x16,     0x15,     0x14,     0x13,     0x12,     0x11,     0x10,
        0x0f,     0x0e,     0x0d,     0x0c,     0x0b,     0x0a,     0x09,     0x08,
        0x07,     0x06,     0x05,     0x04,     0x03,     0x02,     0x01,     0x00
    };
    ERROR_LED = 1;
    OK_LED = 1;
    for(array_point=0; array_point<512; array_point++)
    {
         if(Test_array_one[array_point]!=Test_array_two [array_point]){
             ERROR_LED = 0;
             OK_LED = 1;
             break;
        }
         else{
             OK_LED = 0;
             ERROR_LED = 1;
         }
    }
    while(1);
}
```

双数据指针 DPTR0,DPTR1 的使用

适用型号:

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC

STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

STC89LE516AD,STC89LE516X2

RC/RD+/AD/X2 系列8051 单片机 双数据指针 特殊功能寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR1	A2h	Auxiliary Register 1	-	-	-	-	GF2	-	-	DPS	xxxx,0xx0

Symbol符号 Function功能

GF2 General purpose user-defined flag. 通用功能用户自定义位

DPS DPTR registers select bit. DPTR 寄存器选择位

0: DPTR0 is selected DPTR0被选择

1: DPTR1 is selected DPTR1被选择

此系列单片机有两个 16-bit 数据指针,DPTR0,DPTR1.当 DPS 选择位为 0 时,选择 DPTR0, 当 DPS 选择位为 1 时,选择 DPTR1.

AUXR1 特殊功能寄存器,位于 A2H 单元,其中的位不可用布尔指令快速访问.但由于 DPS 位位于 bit0,故对AUXR1寄存器用 INC指令,DPS位便会反转,由0变成1或由1变成0,即可实现双数据指针的快速切换.

应用示例供参考:

```
;新增特殊功能寄存器定义
AUXR1   DATA    0A2H
MOV     AUXR1,  #0      ;此时 DPS 为 0,DPTR0 有效

MOV     DPTR,   #1FFH   ;置 DPTR0 为 1FFH
MOV     A,      #55H
MOVX    @DPTR,  A       ;将 1FFH 单元置为 55H

MOV     DPTR,   #2FFH   ; 置 DPTR0 为 2FFH
MOV     A,      #0AAH
MOVX    @DPTR,  A       ;将 2FFH 单元置为 0AAH

INC     AUXR1           ; 此时 DPS 为 1,DPTR1 有效
MOV     DPTR,   #1FFH   ; 置 DPTR1 为 1FFH
MOVX    A,      @DPTR   ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC     AUXR1           ; 此时 DPS 为 0,DPTR0 有效
MOVX    A,      @DPTR   ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.

INC     AUXR1           ; 此时 DPS 为 1,DPTR1 有效
MOVX    A,      @DPTR   ; 读 DPTR1 数据指针指向的 1FFH 单元的内容,累加器 A 变为 55H.

INC     AUXR1           ; 此时 DPS 为 0,DPTR0 有效
MOVX    A,      @DPTR   ; 读 DPTR0 数据指针指向的 2FFH 单元的内容,累加器 A 变为 0AAH.
```

结论: 与Philips使用方式一致

P4 口 （可以位寻址,可像操作 P1/P2/P3 一样操作 P4 口）

RC/RD+系列 8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P0	80h	8-bit Port 0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	1111,1111
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P2	A0h	8-bit Port 2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	1111,1111
P3	B0h	8-bit Port 3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1111,1111
P4	E8h	4-bit Port 4	-	-	-	-	P4.3	P4.2	P4.1	P4.0	xxxx,1111

汇编语言：

```
P4      DATA    0E8H ;   or  P4  EQU  0E8H
MOV     A,      P4   ;   Read P4 status to Accumulator.
MOV     P4,     #0AH ;   Output data “A”through P4.0 - P4.3
SETB    P4.0         ;   P4.0 = 1
CLR     P4.0         ;   P4.0 = 0
MOV     P4,     #0AH ;   Output data “A”through P4.0 - P4.3
```

C 语言:

```
sfr     P4   =   0xe8;    C 语言中声明 P4 口特殊功能寄存器地址
```

注：STC89LE516AD,STC89LE516X2,STC89LE58AD,STC89LE54AD 的 P4 口地址在 C0h。

看门狗应用

适用型号:

STC89C51RC,STC89C52RC,STC89C53RC,STC89LE51RC,STC89LE52RC,STC89LE53RC

STC89C54RD+,STC89C58RD+,STC89C516RD+,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

宏晶技术支持,请直接向宏晶采购晶片,而不要通过中间商,支持我们更好地服务

RC/RD+系列 8051单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFR

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	-	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol符号 Function功能

EN_WDT Enable WDT bit. When set, WDT is started

看门狗允许位， 当设置为“1”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.

看门狗清“0”位,当设为“1”时，看门狗将重新计数。硬件将自动清“0”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE mode

看门狗“IDLE”模式位,当设置为“1”时,看门狗定时器在“空闲模式”计数

当清“0”该位时，看门狗定时器在“空闲模式”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:

看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz and 12 clocks mode
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算

看门狗溢出时间 = (N x Pre-scale x 32768) / Oscillator frequency

N = 12，当在12 clock mode时，N = 6，当在6 clock mode时

设时钟为12MHz，12时钟模式

看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz and 12 clocks mode
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为11.0592MHz，12时钟模式

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz and 12 clocks mode
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR   DATA    0E1H ;    或者   WDT_CONTR  EQU   0E1H
;复位入口
    ORG     0000H
    LJMP    Initial
    …
    ORG     0060H
Initial:
    MOV     WDT_CONTR,  #00110100B; Load initial value 看门狗定时器控制寄存器初始化
               ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 0, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL   Display_Loop
    LCALL   Keyboard_Loop
    …
    MOV     WDT_CONTR,  #00110100B ;  喂狗，  不要用 ORL    WDT_CONTR,  #00010000B
    …
    LJMP    Main_Loop
```

C 语言程序示例

```
#include<reg52.h>
sfr     WDT_CONTR   =   0xe1;
void main()
{   …
    WDT_CONTR   =   0x34;
        /* 0011,0100 EN_WDT = 1,CLR_WDT = 1, IDLE_WDT = 0, PS2 = 1, PS1 = 0, PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR   =   0x34;  /* 喂狗，  不要用  WDT_CONTR  =   WDT_CONTR | 0x10;*/
    }
}
```

```
;本程序用于验证 STC89C/LE51RC/RD+ 系列单片机的看门狗及其溢出时间计算公式
;看门狗及其溢出时间 = (N * Pre_scale *32768)/Oscillator frequency
;  N = 12，当在 12 clock mode 时，N = 6，当在 6 clock mode 时。

WDTCR       EQU       0E1H          ;看门狗地址
LED         EQU       P1.5          ;用 P1.5 控制发光二级管

Pre_scale_Word   EQU    0x35        ;清0、启动看门狗，预分频数=64
;f=18.432MHz、12clock mode 时
;    看门狗溢出时间 = (12 * 64 * 32768)/18432000 = 1.36S

    ORG    0000H
    AJMP   main

    ORG    0100H
main:
    CLR    LED                    ;点亮 LED
    ACALL delay                   ;延时，让 LED 亮大约 1S 的时间

     MOV    WDTCR, #Pre_scale_Word ;启动看门狗，若注释掉本条指令即不启动狗，
                                   ;LED 只会亮一次
    SETB   LED                    ;熄灭 LED
wait:
    SJMP   wait           ;跳转到本语句(停机)，等待看门狗溢出复位，复位后将再次点亮 LED

delay:
    MOV    R0, #0
    MOV    R1, #0
    MOV    R2, #15
delay_loop:
    DJNZ   R0, delay_loop
    DJNZ   R1, delay_loop
    DJNZ   R2, delay_loop
    RET

    END
```

中断

RC/RD+系列8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	PX3H	PX2H	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
XICON	C0h	Auxiliary Interrupt Control	PX3	EX3	IE3	IT3	PX2	EX2	IE2	IT2	0000,0000

中断与普通8052完全兼容，优先级可设为4级，另增加2个外部中断INT2/P4.3，INT3/P4.2。

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	IF1
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
Timer 2	002BH	5	PT2H,PT2	0,0	0,1	1,0	1,1	TF2 + EXF2
/INT2	0033H	6	PX2H,PX2	0,0	0,1	1,0	1,1	IE2
/INT3	003BH	7(最低)	PX3H,PX3	0,0	0,1	1,0	1,1	IE3

XICON（扩展中断控制）寄存器，控制外部中断INT2/INT3

Name	Function
PX3	External interrupt 3 priority high if set
EX3	External interrupt 3 enable if set
IE3	IE3 is set/cleared automatically by hardware when interrupt is detected/serviced
IT3	External interrupt 3 is falling-edge/low-level triggered when this bit is set/cleared by software
PX2	External interrupt 2 priority high if set
EX2	External interrupt 2 enable if set
IE2	IE2 is set/cleared automatically by hardware when interrupt is detected/serviced
IT2	External interrupt 2 is falling-edge/low-level triggered when this bit is set/cleared by software
PXH3	External interrupt 3 priority highest if set
PXH2	External interrupt 2 priority highest if set

STC89C51RC/RD+ 系列单片机如何从掉电模式唤醒

```
;*******************************************************************
;Wake Up Idle and Wake Up Power Down
;*******************************************************************
    ORG   0000H
    AJMP MAIN

    ORG   0003H
int0_interrupt:
    CLR   P1.7          ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0013H
int1_interrupt:
    CLR   P1.6          ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0100H
delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #02
delay_loop:
    DJNZ  R0, delay_loop
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    RET

main:
    MOV   R3, #0        ;P1 LED 递增方式变化，表示程序开始运行
main_loop:
    MOV   A, R3
    CPL   A
    MOV   P1, A
    ACALL delay
```

```
    INC   R3
    MOV   A, R3
    SUBB  A, #18H
    JC    main_loop

    MOV   P1, #0FFH      ;熄灭全部灯表示进入 Power Down 状态

    CLR   IT0            ;设置低电平激活外部中断
;   SETB  IT0            ;下降沿激活不了 Power Down 状态下的外部中断。原因是
                         ;MCU 判断下降沿需要 2 个机器周期，而此时 CLOCK 已停止，
                         ;MCU 无法运行 2 个机器周期。
    SETB  EX0            ;允许外部中断 0

    CLR   IT1            ;设置低电平激活外部中断
;   SETB  IT1            ;下降沿激活不了 Power Down 状态下的外部中断，原因同上
    SETB  EX1            ;允许外部中断 1

    SETB  EA             ;开中断，若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV   A, PCON        ;令 PD=1，进入 Power Down 状态，PD = PCON.2
    ORL   A, #02H
    MOV   PCON, A

    MOV   PCON, #01H     ;删除本语句前的";"，同时将前3条语句前加上注释符号";",
                         ;令 IDL=1，可进入 idle 状态，IDL = PCON.1

    MOV   P1, #0DFH      ;请注意:
                         ; 1.外部中断使MCU退出 Power Down 状态,执行本条指令后
                         ;响应中断，表现为 P1.5 与 P1.7 的 LED 同时亮(INT0 唤醒)
                         ; 2.外部中断使MCU退出 idle 状态,先响应中断然后再执行本
                         ;条指令，表现为 P1.7 的 LED 先亮(INT0 唤醒)P1.5 的 LED 后亮

WAIT1：
    SJMP  WAIT1             ;跳转到本语句，停机

    END
```

STC 单片机定时器 0/1/2 的使用

1.1 定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 C/$\overline{T}$ 进行选择，TMOD 寄存器的各位信息如表 1 所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

表 1 寄存器 TMOD 各位的功能描述

TMOD 地址：89H 复位值：00H

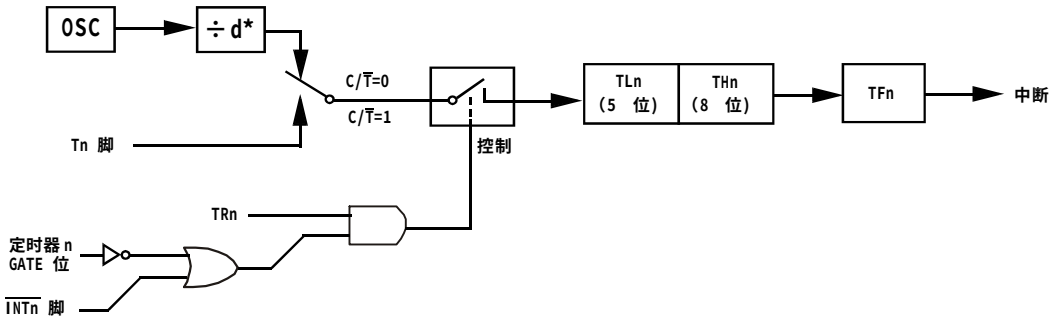
不可位寻址

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.3/ TMOD.7	GATE	用于定时器 1，置位时只有在 $\overline{INT1}$ 脚置高及 TR1 控制置位时才可打开定时器 / 计数器。清零时，置位 TR1 即可打开定时器 / 计数器。
TMOD.2/ TMOD.6	C/$\overline{T}$	控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置位用作计数器（从 T n 脚输入）
	M1、M0	定时器模式选择
	M1、M0	定时器模式
	0 0	8048 定时器 TLn 用做 5 位预分频器
	0 1	16 位定时器 / 计数器，无预分频器
	1 0	8 位自装载定时器，当溢出时将 THn 存放的值装入 TLn。
	1 1	定时器 0 此时作为双 8 位定时 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 控制位控制，在这种模式下定时 / 计数器 1 关闭

1. 模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。图 1 所示为模式 0 工作方式。此模式下，定时器寄存器配置为 13 位寄存器。当计数从全为“1”翻转为全为“0”时，定时器中断标志位 TFn 置位。当 TRn=1、同时 GATE=0 或 $\overline{INTn}$=1 时，定时器计数。置位 GATE 时允许由外部输入 $\overline{INTn}$ 控制定时器，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述如表 2 所列。



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 1　　定时 / 计数器 0 / 1 的模式 0：13 位定时 / 计数器

表 2　寄存器 TCON 各位的功能描述

TCON　地址：88H										
可位寻址		7	6	5	4	3	2	1	0	
复位值：00H		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
位	符 号	功 能								
TCON.7	TF1	定时器 1 溢出标志。定时 / 计数器溢出时由硬件置位；中断处理时由硬件清除，或用软件清除								
TCON.6	TR1	定时器 1 运行控制位。由软件置位 / 清零，将定时 / 计数器打开 / 关闭								
TCON.5	TF0	定时器 0 溢出标志。定时 / 计数器溢出时由硬件置位；中断处理时由硬件清除，或用软件清除								
TCON.4	TR0	定时器 0 运行控制位。由软件置位 / 清零，将定时 / 计数器打开 / 关闭								
TCON.3	IE1	中断 1 边沿触发标志。当检测到外部中断 1 边沿时由硬件置位，中断处理时清零								
TCON.2	IT1	中断 1 触发类型控制位。由软件置位 / 清零，以选择外部中断下降沿 / 低电平方式触发								
TCON.1	IE0	中断 0 边沿触发标志。当检测到外部中断 0 边沿时由硬件置位，中断处理时清零								
TCON.0	IT0	中断 0 触发类型控制位。由软件置位 / 清零，以选择外部中断下降沿 / 低电平方式触发								

该 13 位寄存器包含 THn 全部 8 个位及 TLn 的低 5 位。TLn 的高 3 位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式 0 的操作对于定时器 0 及定时器 1 都是相同的。2 个不同的 GATE 位（TMOD.7 和 TMOD.3）分别分配给定时器 0 及定时器 1。

2. 模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

3. 模式 2

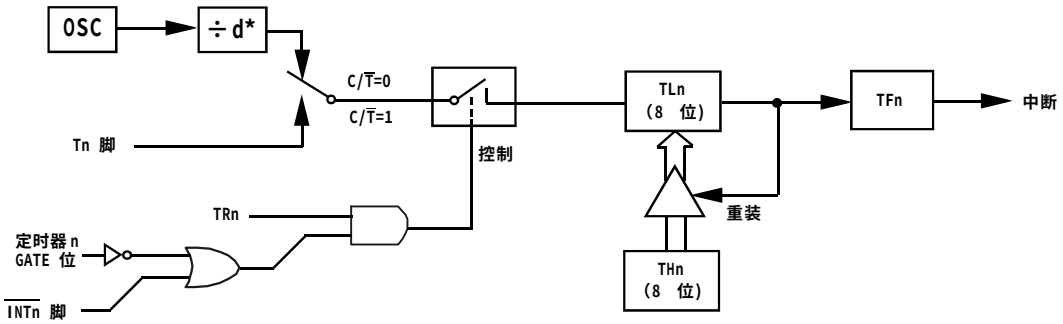
此模式下定时器寄存器作为可自动重装的 8 位计数器（TLn），如图 2 所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。

4. 模式 3

在模式 3 中，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。

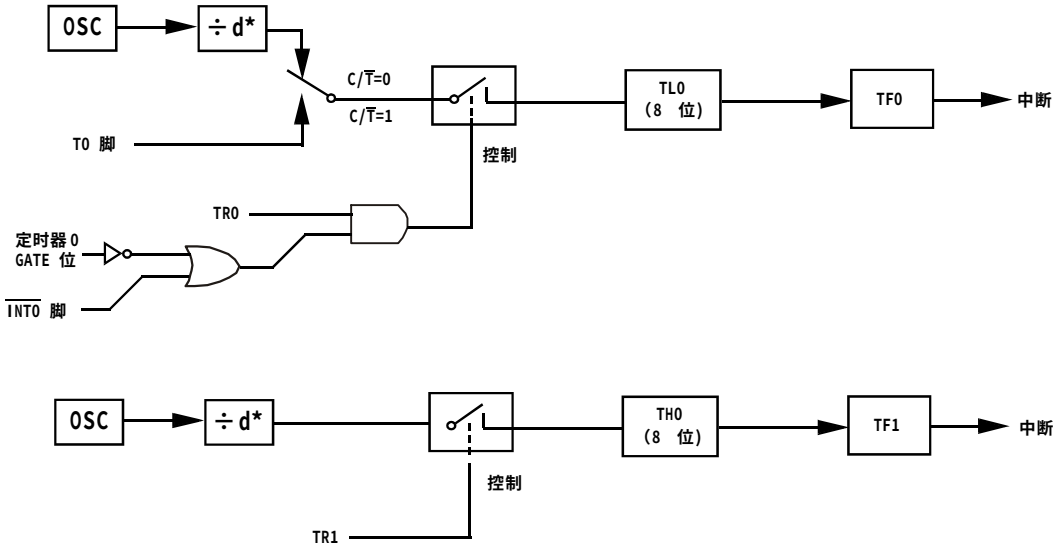
此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。图 3 为模式 3 时的定时器 0 逻辑。TL0 占用定时器 0 的控制位：C/$\overline{T}$、GATE、TR0、$\overline{INT0}$ 及 TF0。TH0 限定为定时器功能（计数器周期），占用定时器 1 的 TR1 及 TF1。此时，TH0 控制定时器 1 中断。

模式 3 可用于需要一个额外的 8 位定时器场合。定时器 0 工作于模式 3 时，80C51 看似有 3 个定时器 / 计数器。当定时器 0 工作于模式 3 时，定时器 1 可通过开关进入 / 退出模式 3，它仍可用作串行端口的波特率发生器，或者应用于任何不要求中断的场合。



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 2 定时 / 计数器 0 / 1 的模式 2：8 位自动重装



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 3 定时 / 计数器 0 的模式 3：双 8 位计数器

1.应用举例

【例 1.1】 定时 / 计数器编程，定时 / 计数器的应用编程主要需考虑：根据应用要求，通过程序初始化，正确设置控制字，正确计算和计算计数初值，编写中断服务程序，适时设置控制位等。通常情况下，设置顺序大致如下：

1) 工作方式控制字（TMOD、T2CON）的设置；

2) 计数初值的计算并装入 THx、TLx、RCAP2H、RCAP2L；

3) 中断允许位 ETx、EA 的设置，使主机开放中断；

4) 启 / 停位 TRx 的设置等。

现以定时 / 计数器 0 或 1 为例作一简要介绍。

STC89C51RC/RD+ 系列单片机的定时器 / 计数器 0 或 1 是以不断加 1 进行计数的，即属加 1 计数器，因此，就不能直接将实际的计数值作为计数初值送入计数寄存器 THx、TLx 中去，而必须将实际计数值以 2^8、2^{13}、2^{16} 为模求补，以其补码作为计数初值设置 THx 和 TLx。

设：实际计数值为 X，计数器长度为 n（n=8、13、16），则应装入计数器 THx、TLx 中的计数初值为 2^n-x，式中 2^n 为取模值。例如，工作方式 0 的计数长度为 13 位，则 n=13，以 2^{13} 为模，工作方式 1 的计数长度为 16，则 n=16，以 2^{16} 为模等等。所以，计数初值为（x）= 2^n-x。

对于定时模式，是对机器周期计数，而机器周期与选定的主频密切相关。因此，需根据应用系统所选定的主频计算出机器周期值。现以主频 6MHz 为例，则机器周期为：

$$一个机器周期 = \frac{12}{主振频率} = \frac{12}{6\times10^6}\ \mu s = 2\ \mu s$$

实际定时时间 $T_c = x \cdot T_p$

式中 T_p 为机器周期，T_c 为所需定时时间，x 为所需计数次数。T_p 和 T_p 一般为已知值，在求出 T_p 后即可求得所需计数值 x，再将 x 求补码，即求得定时计数初值。即

$$(x)_{补} = 2^n - x$$

例如，设定时时间 T_c = 5ms，机器周期 T_P = 2 μs，可求得定时计数次数

$$x = \frac{5ms}{2\ \mu s} = 2500次$$

设选用工作方式 1，则 n=16，则应设置的定时时间计数初值为：$(x)_{补} = 2^{16} - x$=65536-2500=63036，还需将它分解成两个 8 位十六进制数，分别求得低 8 位为 3CH 装入 TLx，高 8 位为 F6H 装入 THx 中。

工作方式 0、1、2 的最大计数次数分别为 8192、65536 和 256。

对外部事件计数模式，只需根据实际计数次数求补后变换成两个十六进制码即可。

【例 1.2】 定时 / 计数器应用编程，设某应用系统，选择定时 / 计数器 1 定时模式，定时时间 T_c = 10ms，主频频率为 12MHz，每 10ms 向主机请求处理。选定工作方式 1。计算得计数初值：低 8 位初值为 F0H，高 8 位初值为 D8H。

（1 ）初始化程序

所谓初始化，一般在主程序中根据应用要求对定时/计数器进行功能选择及参数设定等预置程序，本例初始化程序如下：

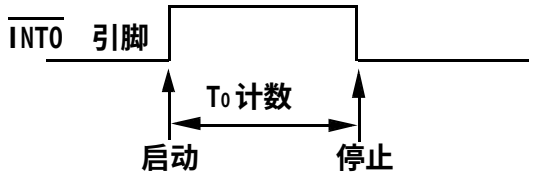
```
START：
        ⋮                       ；主程序段
       MOV  SP，#60H            ；设置堆栈区域
       MOV  TMOD，#10H          ；选择T1、定时模式，工作方式1
       MOV  TH1，#0D8H          ；设置高字节计数初值
       MOV  TL1，#0F0H          ；设置低字节计数初值
       SETB EA                  }
       SETB ET1                 }  开中断
        ⋮                       ；其他初始化程序
       SETB TR1                 ；启动T1开始计时
        ⋮                       ；继续主程序
```

（2）中断服务程序

```
INTT1： PUSH A                  }
        PUSH DPL                }  现场保护
        PUSH DPH                }
         ⋮
        MOV  TL1,#0F0H          }
        MOV  TH1,#0D8H          }  重新置初值
         ⋮                      ；中断处理主体程序
       POP  DPH                 }
       POP  DPL                 }  现场恢复
       POP  A                   }
       RETI                     ；返回
```

这里展示了中断服务子程序的基本格式。STC89C51RC/RD+系列单片机的中断属于矢量中断，每一个矢量中断源只留有8个字节单元，一般是不够用的，常需用转移指令转到真正的中断服务子程序区去执行。

【例1.3】 对外部正脉冲测宽。选择定时/计数器2进行脉宽测试较方便，但也可选用定时/计数器0或定时/计数器1进行测宽操作。本例选用定时/计数器0（T_0）以定时模式，工作方式1对$\overline{INT0}$引脚上的正脉冲进行脉宽测试。



设置GATE为1，机器周期T_P为1 μs。本例程序段编制如下：

```
INTT0： MOV  TMOD，#09H         ；设T0为定时方式1，GATE为1
```

```
            MOV  TL0，#00H          } TH0,TL0 清 0
            MOV  TH0，#00H          }
            CLR  EX0                ；关 INT0 中断
    LOP1：  JB   P3.2，LOP1         ；等待 INT0 引低电平
    LOP2：  JNB  P3.2，LOP2         ；等待 INT0 引脚高电平
            SETB TR0                ；启动 T0 开始计数
    LOP3：  JB   P3.2，LOP3         ；等待 INT0 低电平
            CLR  TR0                ；停止 T0 计数
            MOV  A，TL0             ；低字节计数值送 A
            MOV  B，TH0             ；高字节计数值送 B
             ⋮                      ；计算脉宽和处理
```

【例 1.4】 利用定时 / 计数器 0 或定时 / 计数器 1 的 Tx 端口改造成外部中断源输入端口的应用设计。

在某些应用系统中常会出现原有的两个外部中断源 INT0 和 INT1 不够用，而定时 / 计数器有多余，则可将 Tx 用于增加的外部中断源。现选择定时 / 计数器 1 为对外部事件计数模式工作方式 2（自动再装入），设置计数初值为 FFH，则 T1 端口输入一个负跳变脉冲，计数器即回 0 溢出，置位对应的中断请求标志位 TF1 为 1，向主机请求中断处理，从而达到了增加一个外部中断源的目的。应用定时 / 计数器 1（T1）的中断矢量转入中断服务程序处理。其程序示例如下：

（1 ）主程序段：

```
    ORG  0000H
    AJMP MAIN           ；转主程序
    ORG  001BH
    LJMP INTER          ；转 T1 中断服务程序
     ⋮
    ORG  0100           ；主程序入口
MAIN：…
     ⋮
    MOV  SP，#60H       ；设置堆栈区
    MOV  TMOD，#60H     ；设置定时 / 计数器 1，计数方式 2
    MOV  TL1，#0FFH     ；设置计数常数
    MOV  TH1，#0FFH
    SETB EA             ；开中断
    SETB ET1            ；开定时 / 计数器 1 中断
    SETB TR1            ；启动定时 / 计数器 1 计数
     ⋮
```

（2 ）中断服务程序（具体处理程序略）

```
            ORG  1000H
    INTER： PUSH A              }
            PUSH DPL            } 现场入栈保护
            PUSH DPH            }
```

```
        ⋮                    } 中断处理主体程序
        ⋮
      POP  DPH               }
      POP  DPL               } 现场出栈复原
      POP  A                 }
      RETI                   ;返回
```

这是中断服务程序的基本格式。

【例 1.5】 某应用系统需通过 P1.0 和 P1.1 分别输出周期为 200 μs 和 400 μs 的方波。为此，系统选用定时器 / 计数器 0（T_0），定时方式 3，主频为 6MHz，T_P=2 μs，经计算得定时常数为 9CH 和 38H。

本例程序段编制如下：

（1）初始化程序段

```
        ⋮
PLT0:MOV  TMOD,#03H        ;设置 T0 定时方式 3
     MOV  TL0，#9CH        ;设置 TL0 初值
     MOV  TH0，#38H        ;设置 TH0 初值
     SETB EA               ;
     SETB ET0              ;} 开中断
     SETB ET1              ;
     SETB TR0              ;启动
     SETB TR1              ;启动
      ⋮
```

(2)中断服务程序段

1)

```
INT0P：⋮
       ⋮
    MOV  TL0，#9CH         ;重新设置初值
    CPL  P1.0              ;对 P1.0 输出信号取反
       ⋮
    RETI                   ;返回
```

2)

```
INT1P  ⋮
       ⋮
    MOV  TH0，#38H         ;重新设置初值
    CPL  P1.1              ;对 P1.1 输出信号取反
       ⋮
    RETI                   ;返回
```

在实际应用中应注意的问题如下。

（1）定时 / 计数器的实时性

定时 / 计数器启动计数后，当计满回 0 溢出向主机请求中断处理，由内部硬件自动进行。但从回0溢出请求中断到主机响应中断并作出处理存在时间延迟，且这种延时随中断请求时的现场环境的不同而不同，一般需延时 3 个机器周期以上，这就给实时处理带来误差。大多数应用场合可忽略不计，但对某些要求实时性苛刻的场合，应采用补偿措施。

这种由中断响应引起的时间延时，对定时 / 计数器工作于方式0或1而言有两种含义：一是由于中断响应延时而引起的实时处理的误差；二是如需多次且连续不间断地定时 / 计数，由于中断响应延时，则在中断服务程序中再置计数初值时已延误了若干个计数值而引起误差，特别是用于定时就更明显。

例如选用定时方式 1 设置系统时钟，由于上述原因就会产生实时误差。这种场合应采用动态补偿办法以减少系统始终误差。所谓动态补偿，即在中断服务程序中对 THx、TLx 重新置计数初值时，应将 THx、TLx 从回 0 溢出又重新从 0 开始继续计数的值读出，并补偿到原计数初值中去进行重新设置。可考虑如下补偿方法：

```
    ⋮
CLR  EA            ；禁止中断
MOV  A，TLx        ；读 TLx 中已计数值
ADD  A，#LOW       ；LOW 为原低字节计数初值
MOV  TLx，A        ；设置低字节计数初值
MOV  A，#HIGH      ；原高字节计数初值送 A
ADDC A，THx        ；高字节计数初值补偿
MOV  THx，A        ；置高字节计数初值
SETB EA            ；开中断
    ⋮
```

（2）动态读取运行中的计数值

在动态读取运行中的定时 / 计数器的计数值时，如果不加注意，就可能出错。这是因为不可能在同一时刻同时读取 THx 和 TLx 中的计数值。比如，先读 TLx 后读 THx，因为定时 / 计数器处于运行状态，在读 TLx 时尚未产生向 THx 进位，而在读 THx 前已产生进位，这时读得的 THx 就不对了；同样，先读 THx 后读 TLx 也可能出错。

一种可避免读错的方法是：先读 THx，后读 TLx，将两次读得的 THx 进行比较；若两次读得的值相等，则可确定读的值是正确的，否则重复上述过程，重复读得的值一般不会再错。此法的软件编程如下：

```
RDTM：MOV A，THx              ；读取 THx 存 A 中
      MOV R0，TLx             ；读取 TLx 存 R0 中
      CJNE    A，THx，RDTM    ；比较两次 THx 值，若相等，则读得的值正
                              ；确，程序往下执行，否则重读
      MOV R1，A               ；将 THx 存于 R1 中
        ⋮
```

STC 定时器 2 的操作

定时器 2 是一个 16 位定时 / 计数器。通过设置特殊功能寄存器 T2CON 中的 C/T2 位，可将其作为定时器或计数器（特殊功能寄存器 T2CON 的描述如表 1 所列)。定时器 2 有 3 种操作模式：捕获、自动重新装载（递增或递减计数）和波特率发生器，这 3 种模式由 T2CON 中的位进行选择（如表 1 所列)。

表 1 特殊功能寄存器 T2CON 的描述

T2CON 地址 =0C8H 可位寻址 复位值 =00H

7	6	5	4	3	2	1	0
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$

符 号	位	名称和意义
TF2	T2CON.7	定时器 2 溢出标志。定时器 2 溢出时置位，必须由软件清除。当 RCLK 或 TCLK=1 时，TF2 将不会置位
EXF2	T2CON.6	定时器 2 外部标志。当 EXEN2=1 且 T2EX 的负跳变产生捕获或重装时，EXF2 置位。定时器 2 中断使能时，EXF2=1 将使 CPU 从中断向量处执行定时器 2 中断子程序。EXF2 位必须用软件清零。在递增 / 递减计数器模式（DCEN=1）中，EXF2 不会引起中断
RCLK	T2CON.5	接收时钟标志。RCLK 置位时，定时器 2 的溢出脉冲作为串行口模式 1 和模式 3 的接收时钟。RCLK=0 时，将定时器 1 的溢出脉冲作为接收时钟
TCLK	T2CON.4	发送时钟标志。TCLK 置位时，定时器 2 的溢出脉冲作为串行口模式 1 和模式 3 的发送时钟。TCLK=0 时，将定时器 1 的溢出脉冲作为发送时钟
EXEN2	T2CON.3	定时器 2 外部使能标志。当其置位且定时器 2 未作为串行口时钟时，允许 T2EX 的负跳变产生捕获或重装。EXEN2=0 时，T2EX 的跳变对定时器 2 无效
TR2	T2CON.2	定时器 2 启动 / 停止控制位。置 1 时启动定时器
C/$\overline{T2}$	T2CON.1	定时器 / 计数器选择。（定时器 2 ） 0= 内部定时器（OSC/12 或 OSC/6） 1 = 外部事件计数器（下降沿触发）
CP/$\overline{RL2}$	T2CON.0	捕获 / 重装标志。置位：EXEN2=1 时,T2EX 的负跳变产生捕获。清零：EXEN2=0 时，定时器 2 溢出或 T2EX 的负跳变都可使定时器自动重装。当 RCLK=1 或 TCLK=1 时，该位无效且定时器强制为溢出时自动重装

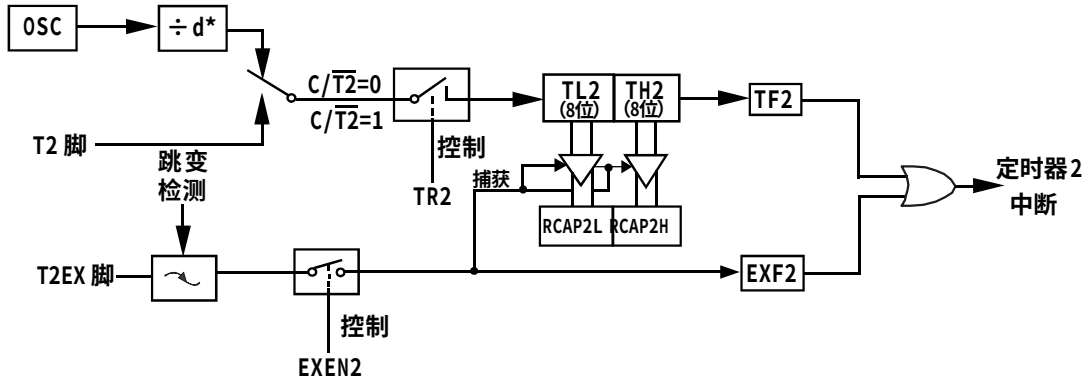
表 2 定时器 2 工作方式

RCLK+TCLK	CP/$\overline{RL2}$	TR2	模 式
0	0	1	16 位自动重装
0	1	1	16 位捕获
1	X	1	波特率发生器
X	X	0	（关闭）

1.捕获模式

在捕获模式中，通过 T2CON 中的 EXEN2 设置 2 个选项。如果 EXEN2=0，定时器 2 作为一个 16 位定时器或计数器（由 T2CON 中 C/$\overline{T2}$ 位选择)，溢出时置位 TF2（定时器 2 溢出标志位)。该位可用于产生中断（通过使能 IE 寄存器中的定时器 2 中断使能位)。如果 EXEN2=1，与以上描述相同，但增加了一个特性，即外部输入 T2EX 由 1 变零时，将定时器 2 中 TL2 和 TH2 的当前值各自捕获到 RCAP2L 和

RCAP2H。另外，T2EX 的负跳变使 T2CON 中的 EXF2 置位，EXF2 也像 TF2 一样能够产生中断（其向量与定时器 2 溢出中断地址相同，定时器 2 中断服务程序通过查询 TF2 和 EXF2 来确定引起中断的事件），捕获模式如图 1 所示。在该模式中，TL2 和 TH2 无重新装载值，甚至当 T2EX 产生捕获事件时，计数器仍以 T2EX 的负跳变或振荡频率的 1/12（12 时钟模式）或 1/6（6 时钟模式）计数。



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 1 定时器 2 捕获模式

2. 自动重装模式（递增 / 递减计数器）

16 位自动重装模式中，定时器 2 可通过 C/T2 配置为定时器 / 计数器，编程控制递增 / 递减计数。计数的方向是由 DCEN（递减计数使能位）确定的，DCEN 位于 T2MOD 寄存器中，T2MOD 寄存器各位的功能描述如表 3 所示。当 DCEN=0 时，定时器 2 默认为向上计数；当 DCEN=1 时，定时器 2 可通过 T2EX 确定递增或递减计数。图 2 显示了当 DCEN=0 时，定时器 2 自动递增计数。在该模式中，通过设置 EXEN2 位进行选择。如果 EXEN2=0，定时器 2 递增计数到 0FFFFH，并在溢出后将 TF2 置位，然后将 RCAP2L 和 RCAP2H 中的 16 位值作为重新装载值装入定时器 2。RCAP2L 和 RCAP2H 的值是通过软件预设的。

表 3 定时器 2 模式（T2MOD）控制寄存器的描述

T2MOD 地址 =0C9H 复位值 =XXXX XX00B
不可位寻址

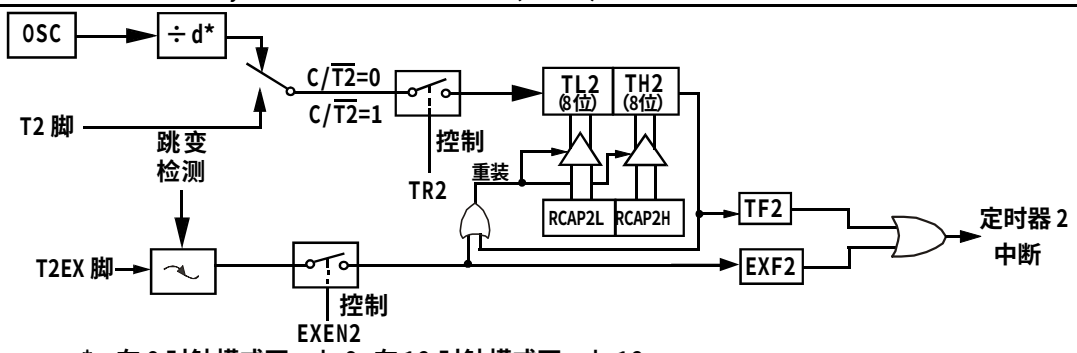
7	6	5	4	3	2	1	0
-	-	-	-	-	-	T2OE	DCEN

符 号	功 能
—	不可用，保留将来之用 *
T2OE	定时器 2 输出使能位
DCEN	向下计数使能位。定时器 2 可配置成向上 / 向下计数器

* 用户勿将其置 1。这些位在将来 80C51 系列产品中用来实现新的特性。在这种情况下，以后用到保留位，复位时或非有效状态时，它的值应为 0；而这些位为有效状态时，它的值为 1。从保留位读到的值是不确定的。

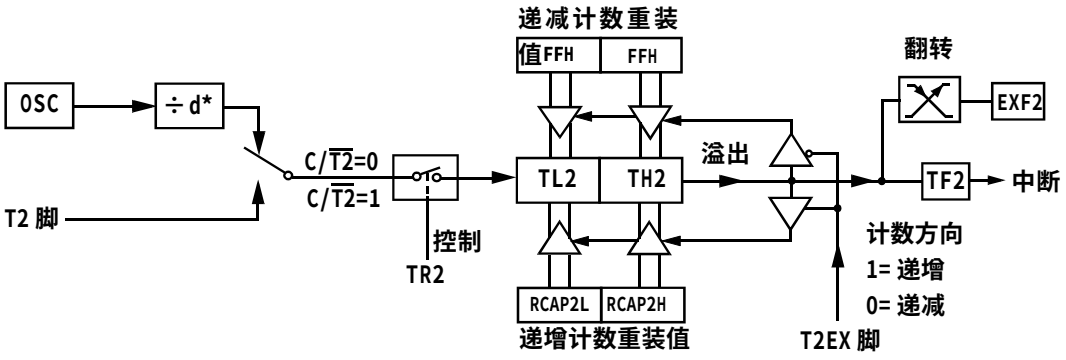
如果 EXEN2=1，16 位重新装载可通过溢出或 T2EX 从 1 0 的负跳变实现。此负跳变同时将 EXF2 置位。如果定时器 2 中断被使能，则当 TF2 或 EXF2 置 1 时产生中断。在图 3 中，DCEN=1 时，定时器 2 可递增或递减计数。此模式允许 T2EX 控制计数的方向。当 T2EX 置 1 时，定时器 2 递增计数，计数到 0FFFFH 后溢出并置位 TF2，还将产生中断（如果中断被使能）。定时器 2 的溢出将使 RCAP2L 和 RCAP2H 中的 16 位值作为重新装载值放入 TL2 和 TH2。

当 T2EX 置零时，将使定时器 2 递减计数。当 TL2 和 TH2 计数到等于 RCAP2L 和 RCAP2H 时，定时器产



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 2 定时器 2 自动重装模式（DCEN=0）



* 在 6 时钟模式下，d=6；在 12 时钟模式下，d=12。

图 3 定时器 2 自动重装模式（DCEN=1）

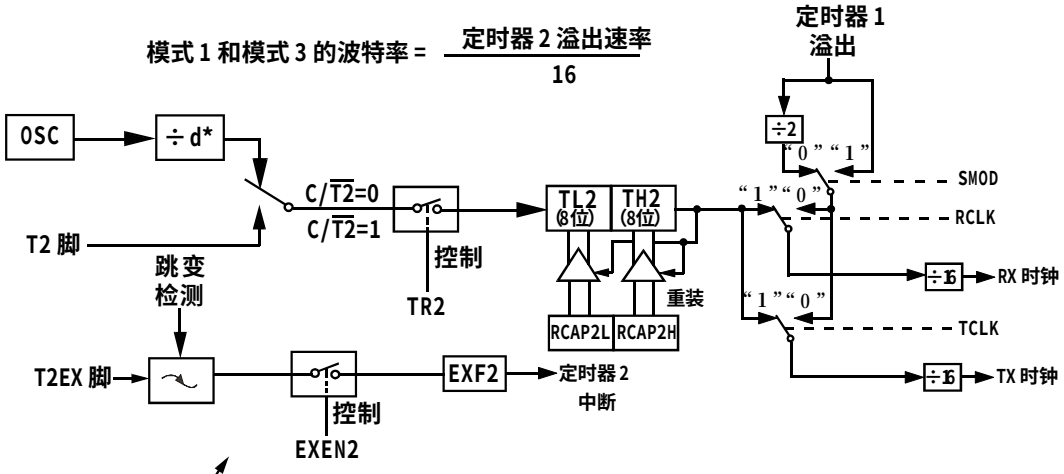
产生中断。

3.波特率发生器模式

寄存器 T2CON 的位 TCLK 和（或）RCLK 允许从定时器 1 或定时器 2 获得串行口发送和接收的波特率。当 TCLK=0 时，定时器 1 作为串行口发送波特率发生器；当 TCLK=1 时，定时器 2 作为串行口发送波特率发生器。RCLK 对串行口接收波特率有同样的作用。通过这 2 位，串行口能得到不同的接收和发送波特率，一个通过定时器 1 产生，另一个通过定时器 2 产生。

如图 4 所示为定时器 2 工作在波特率发生器模式。与自动重装模式相似，当 TH2 溢出时，波特率发生器模式使定时器 2 寄存器重新装载来自寄存器 RCAP2H 和 RCAP2L 的 16 位的值,寄存器 RCAP2H 和 RCAP2L 的值由软件预置。当工作于模式 1 和模式 3 时，波特率由下面给出的公式所决定：

$$\text{模式 1 和模式 3 的波特率} = \frac{\text{定时器 2 溢出速率}}{16}$$



* 在 6 时钟模式下，d=1；在 12 时钟模式下，d=2。

图 4 定时器 2 波特率发生器模式

定时器可配置成“定时”或“计数”方式，在许多应用上，定时器被设置在“定时”方式（C/$\overline{T2}$=0）。当定时器 2 作为定时器时，它的操作不同于波特率发生器。通常定时器 2 作为定时器，它会在每个机器周期递增（1/6 或 1/12 振荡频率）。当定时器 2 作为波特率发生器时，它在 6 时钟模式下，以振荡器频率递增(12 时钟模式时为 1/12 振荡频率)。

这时的波特率公式如下：

$$\text{模式 1 和模式 3 的波特率} = \frac{\text{振荡器频率}}{n \times [65536 - (RCAP2H, RCAP2L)]}$$

式中：n=16（6 时钟模式）或 32（12 时钟模式）；RCAP2H，RCAP2L —— RCAP2H 和 RCAP2L 的内容，为 16 位无符号整数。

如图 4 所示，定时器 2 是作为波特率发生器，仅当寄存器 T2CON 中的 RCLK 和（或）TCLK=1 时，定时器 2 作为波特率发生器才有效。注意：TH2 溢出并不置位 TF2，也不产生中断。这样当定时器 2 作为波特率发生器时，定时器 2 中断不必被禁止。如果 EXEN2（T2 外部使能标志）被置位，在 T2EX 中由 1 到 0 的转换会置位 EXF2（T2 外部标志位），但并不导致（TH2，TL2）重新装载（RCAP2H，RCAP2L）。当定时器 2 用作波特率发生器时，如果需要，T2EX 可用做附加的外部中断。

当计时器工作在波特率发生器模式下，则不要对 TH2 和 TL2 进行读 / 写，每隔一个状态时间（fosc/2)或由 T2 进入的异步信号，定时器 2 将加 1。在此情况下对 TH2 和 TH1 进行读 / 写是不准确的；可对 RCAP2 寄存器进行读，但不要进行写，否则将导致自动重装错误。当对定时器 2 或寄存器 RCAP 进行访问时，应关闭定时器（清零 TR2）。表 4 列出了常用的波特率和如何用定时器 2 得到这些波特率。

4.波特率公式汇总

定时器 2 工作在波特率发生器模式，外部时钟信号由 T2 脚进入，这时的波特率公式如下：

$$\text{波特率} = \frac{\text{定时器 2 溢出率}}{16}$$

如果定时器 2 采用内部时钟信号，则波特率公式如下：

$$\text{波特率} = \frac{fosc}{n \times [65536 - (RCAP2H, RCAP2L)]}$$

表 4 由定时器 2 产生的常用波特率

波特率		振荡器频率 /MHz	定时器 2	
12 时钟模式	6 时钟模式		RCAP2H	RCAP2L
375 000	750 000	12	FF	FF
9 600	19 200	12	FF	D9
2 800	9 600	12	FF	B2
2 400	4 800	12	FF	64
1 200	2 400	12	FE	C8
300	600	12	FB	1E
110	220	12	F2	AF
300	600	6	FD	8F
110	220	6	F9	57

式中:n=32(12 时钟模式)或 16(6 时钟模式),fosc= 振荡器频率。

自动重装值可由下式得到:

$$RCAP2H,RCAP2L = 65536 - [fosc/(n \times \text{波特率})]$$

5.定时器 / 计数器 2 的设置

除了波特率发生器模式,T2CON 不包括 TR2 位的设置，TR2 位需单独设置来启动定时器。如表 5 和表 6 分别列出了 T2 作为定时器和计数器的具体设置方法。

表 5 T2 作为定时器的设置

模 式	T2CON	
	内部控制①	外部控制②
16 位重装	00H	08H
16 位捕获	01H	09H
波特率发生器接收和发送相同波特率	34H	36H
只接收	24H	26H
只发送	14H	16H

① 仅当定时器溢出时进行捕获和重装。

② 当定时 / 计数器溢出并且 T2EX(P1.1)发生电平负跳变时产生捕获和重装(定时器 2 用于波特率发生器模式时除外)。

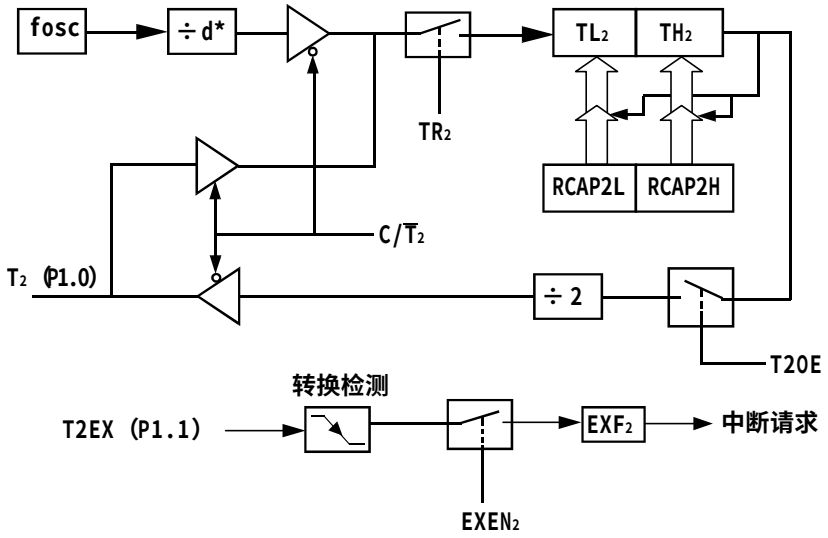
表 6 T2 作为计数器设置

模 式	TMOD	
	内部控制①	外部控制②
16 位	02H	0AH
自动重装	03H	0BH

注①和注②同表 5 的注①和注②。

6.可编程时钟输出

STC89C51RC/RD+ 系列单片机，可设定定时 / 计数器 2，通过 P1.0 输出时钟。P1.0 除作通用 I/O 口外还有两个功能可供选用：用于定时 / 计数器 2 的外部计数输入和定时 / 计数器 2 时钟信号输出。图 5 为时钟输出和外部事件计数方式示意图。



* d=1，6 时钟 / 机器周期；d=2，12 时钟 / 机器周期

图 5 定时器 2 时钟输出和外部事件计数方式示意图

通过软件对T2CON.1位C/$\overline{T}_2$复位为0，对T2MOD.1位T2OE置1就可将定时/计数器2选定为时钟信号发生器，而T2CON.2位TR_2控制时钟信号输出开始或结束（TR_2为启/停控制位).由主振频率（fosc）和定时/计数器2定时、自动再装入方式的计数初值决定时钟信号的输出频率。其设置公式如下：

$$\text{时钟信号输出频率} = \frac{fosc}{n \times [65536 - (RCAP2H \times RCAP2L)]}$$

* n=2，6时钟/机器周期；n=4，12时钟/机器周期

从公式可见，在主振频率（fosc）设定后，时钟信号输出频率就取决于定时计数初值的设定。

在时钟输出模式下，计数器回0溢出不会产生中断请求。这种功能相当于定时/计数器2用作波特率发生器，同时又可以作时钟发生器。但必须注意，无论如何波特率发生器和时钟发生器不能单独确定各自不同的频率。原因是两者都用同一个陷阱寄存器RCAP2H、RCAP2L，不可能出现两个计数初值。

```
;/* ---  STC International Limited ---------------- */
;/* ---  宏晶科技 姚永平  设计 2004/9/11   V1.0 ---- */
;/* ---  RD+/RC 系列   Timer2  Operation ----------- */
;/* ---  STC89C54RD+, STC89C58RD+, STC89C516RD+ --- */
;/* ---  STC89LE54RD+,STC89LE58RD+,STC89LE516RD+ -- */
;/* ---  STC89C51RC,  STC89C52RC,  STC89C53RC ----- */
;/* ---  STC89LE51RC, STC89LE52RC, STC89LE53RC ---- */
;/* ---  Mobile: 13922805190 ---------------------- */
;/* ---  Fax: 0755-82944243 ----------------------- */
;/* ---  Tel: 0755-82908285 ----------------------- */
;/* ---  Web  : www.mcu-memory.com ---------------- */

;-----Timer 2 做波特率发生器 -------------------------

;----- 本程序不提供技术支持，一定要自己测试 --------------

;定义特殊功能寄存器

;与 RS232 口、TIMER2 有关的特殊功能寄存器
T2CON                   EQU 0C8H
TR2                     EQU T2CON.2   ;TR2 是 T2CON 特殊功能寄存器的第 2 位

RCAP2L                  EQU 0CAH
RCAP2H                  EQU 0CBH
TH2                     EQU 0CDH
TL2                     EQU 0CCH

;-----------------------------------------------

;设置波特率自动重装数
RELOAD_COUNT_HIGH       EQU  0FFH

;使用以下参数必须将 RELOAD_COUNT_HIGH 设置为 0FFH
;RELOAD_COUNT_LOW        EQU  0FAH    ;Fosc = 22.1184MHz, Baud = 115200
;RELOAD_COUNT_LOW        EQU  0EEH    ;Fosc = 22.1184MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0F0H    ;Fosc = 20.000MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0F6H    ;Fosc = 12.000MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0F7H    ;Fosc = 11.059MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0F8H    ;Fosc = 10.000MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0FBH    ;Fosc = 6.000MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  0FCH    ;Fosc = 5.000MHz, Baud = 38400
;RELOAD_COUNT_LOW        EQU  070H    ;Fosc = 11.059MHz, Baud = 2400

;--------------
;计算自动重装数:
;--------------
;晶体频率: Fosc
;波特率:   Baud
```

```
;自动重装数:   RELOAD = INT(Fosc/Baud/32 + 0.5), INT 表示取整运算(舍去小数)
;将自动重装数转换成 16 进制, 用 10000H 减自动重装数, 存入 RCAP2H, RCAP2L
;计算实际的波特率: Baud = Fosc/RELOAD/32, 如果误差>3.5 要更改波特率.

;例: Fosc = 22.1184MHz, Baud = 115200
;  RELOAD = INT( 22118400/115200/32 + 0.5)
;         = INT( 6.5 )
;         = 6
;         = 0006H
; 10000H - 0006H = FFFAH
;
;   MOV   RCAP2H, #0FFH
;   MOV   RCAP2L, #0FAH

;例: Fosc = 20.MHz, Baud = 57600 (Baud=115200 时误差太大)
;  RELOAD = INT( 20000000/57600/32 + 0.5)
;         = INT( 10.85 + 0.5 )
;         = INT( 11.35 )
;         = 11
;         = 000BH
; 10000H - 000BH = FFF5H
;
;   MOV   RCAP2H, #0FFH
;   MOV   RCAP2L, #0F5H
;-----------------------------------------
    ORG   0000H
    AJMP  MAIN
;-----------------------------------------
    ORG   0023H                     ;RS232 串口中断
    AJMP  UART
    NOP
    NOP
;-----------------------------------------
MAIN:
    MOV   SP, #0E0H
    ACALL Initial_UART              ;初始化串口
    MOV   R0, #30H                  ;发送 10 个字符 '0123456789'
    MOV   R2, #10
LOOP:
    MOV   A, R0
    ACALL Send_One_Byte             ;发送一个字节
    INC   R0
    DJNZ  R2, LOOP

WAIT1：
    SJMP  WAIT1                     ;跳转到本行, 无限循环
;-----------------------------------------------
UART:                               ;串口中断服务程序
```

```
    JBC   RI, UART_1
    RETI                            ;发送时使用的是查询方式，不使用中断
UART_1:                             ;接收一个字节．此时 RI 已被清 0
    PUSH  ACC
    MOV   A, SBUF                   ;取接收到的字节
    ACALL Send_One_Byte             ;回发收到的字节
    POP   ACC
    RETI
;-----------------------------------------------
Initial_UART:                       ;初始化串口
;       Bit:   7       6     5     4     3     2     1     0
; SCON        SM0/FE   SM1   SM2   REN   TB8   RB8    Tl   Rl
    MOV   SCON, #50H                ; 0101,0000 8 位可变波特率，无奇偶校验
Init_RS232_1:
    MOV   A, #RELOAD_COUNT_HIGH     ;波特率自动重装数
    MOV   RCAP2H, A
    MOV   TH2, A
    MOV   A, #RELOAD_COUNT_LOW
    MOV   RCAP2L, A
    MOV   TL2, A
    MOV   T2CON, #0x34              ;使用 T2 作波特率发生器
    SETB  ES                        ;允许串口中断
    SETB  EA                        ;开总中断
    RET
;-----------------------------------------------
Send_One_Byte:                      ;发送一个字节
    CLR   ES
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF, A

WAIT2：
    JNB   TI, WAIT2                 ;等待发送完毕
    CLR   TI                        ;清零串口发送中断标志
    SETB  ES
    RET
;-----------------------------------------------
    END
;-----------------------------------------------
```

```
;-------------------------------------------------------------------------------
;本程序演示了如何使用定时器 2 的时钟 / 脉冲输出功能，在 P1.0 口输出
;-------------------------------------------------------------------------------
;定义特殊功能寄存器

;与 RS232 口、TIMER2 有关的特殊功能寄存器
T2CON                   EQU 0C8H
T2MOD                   EQU 0C9H

TR2                     EQU T2CON.2   ;TR2 是 T2CON 特殊功能寄存器的第 2 位

RCAP2L                  EQU 0CAH
RCAP2H                  EQU 0CBH
TH2                     EQU 0CDH
TL2                     EQU 0CCH

;定时器 / 计数器 2 控制寄存器 T2CON
;                   D7    D6    D5    D4    D3    D2    D1    D0      Reset Value
;  位地址           CF    CE    CD    CC    CB    CA    C9    C8
; T2CON(C8H)       TF2   EXF2  RCLK  TCLK  EXEN2 TR2   C/T2  CP/RL2      00

;T2MOD 寄存器
;                   D7    D6    D5    D4    D3    D2    D1    D0      Reset Value
; T2CON(C9H)        -     -     -     -     -     -     T2OE  DCEN     xxxxxx00b

;-----------------------------------------------
    ORG   0000H
    AJMP  MAIN
;-----------------------------------------------
    ORG   0100H
MAIN:
    MOV   SP, #0E0H

    MOV   P1, #0FFH                 ;熄灭 P1 口的发光二级管
    ACALL SET_T2_OUT_MODE           ;设置 T2 为高速脉冲输出方式

    MOV   DPTR, #0FFF0H             ;设置 T2 脉冲输出速率
    ACALL SET_T2_OUT_SPEED

WAIT1：
    SJMP  WAIT1                     ;可加入此行，用频率计或其它仪器测量 P1.0 的
                                    ;输出信号，验证脉冲输出频率的计算公式

    ACALL DELAY
    ACALL PAUSE                     ;暂停输出，便于观察

    MOV   DPTR, #0FFE0H             ;设置 T2 脉冲输出速率，比前一次降低一半
    ACALL SET_T2_OUT_SPEED
```

```
    ACALL DELAY
    ACALL PAUSE                     ;暂停输出，便于观察
    MOV   DPTR, #0FFD0H             ;设置 T2 脉冲输出速率，比前一次降低 1/3
    ACALL SET_T2_OUT_SPEED
    ACALL DELAY
    ACALL PAUSE                     ;暂停输出，便于观察

WAIT2：
    SJMP  WAIT2                     ;跳转到本行，无限循环
;-----------------------------------------------

DELAY:
    MOV   R1, #0
    MOV   R2, #0
    MOV   R3, #30
DELAY_LOOP:
    DJNZ  R1, DELAY_LOOP
    DJNZ  R2, DELAY_LOOP
    DJNZ  R3, DELAY_LOOP
    RET
;-----------------------------------------------

SET_T2_OUT_MODE:                    ;设置 T2 为脉冲输出方式
    MOV   T2CON, #0                 ;设置 T2 为定时器方式
    MOV   T2MOD, #02                ;0000, 0010 允许 T2 溢出脉冲由 P1.0 输出
    RET
;-----------------------------------------------
;脉冲输出频率由振荡器频率和 T2 的捕获寄存器 RCAP2H、RCAP2L 的重新装入值确定，
;计算公式：
;      振荡器频率 / (n*(65536 - RCAP2H,RCAP2L)
;公式中 n = 2，在 6 Clock 模式; n = 4，在 12 Clock 模式
;       RCAP2H,RCAP2L 是由 RCAP2H 和 RCAP2L 组成的 16 位无符号整数，
;入口: DPTR = 重装数
SET_T2_OUT_SPEED:                   ;设置 T2 脉冲输出速率
    CLR   TR2                       ;停止 T2 工作
    MOV   RCAP2H, DPH
    MOV   RCAP2L, DPL
    SETB  TR2                       ;启动 T2
    RET
;-----------------------------------------------
PAUSE:                              ;暂停输出
    CLR   TR2                       ;停止 T2 工作
    MOV   P1, #0FFH                 ;熄灭 P1 口的发光二级管
    ACALL DELAY
    RET
;-----------------------------------------------
    END
;-----------------------------------------------
```

STC89C51 RC / RD+ 系列 ISP / IAP 应用
STC89C51 RC / RD+ 系列 内部EEPROM的应用

-- 利用IAP技术可实现EEPROM，内部Flash擦写次数为100,000次以上

RC/RD+系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

ISP_DATA: ISP/IAP操作时的数据寄存器。
ISP/IAP从Flash读出的数据放在此处，向Flash写的数据也需放在此处

ISP_ADDRH: ISP/IAP操作时的地址寄存器高八位。

ISP_ADDRL: ISP/IAP操作时的地址寄存器低八位。

ISP_CMD: ISP/IAP操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令 / 操作 模式选择
保留					命令选择			
-	-	-	-	-	0	0	0	Standby 待机模式，无ISP操作
-	-	-	-	-	0	0	1	AP-Flash / Data-Flash Read 对用户的应用程序Flash区及数据Flash区字节读
-	-	-	-	-	0	1	0	AP-Flash / Data-Flash Program 对用户的应用程序Flash区及数据Flash区字节编程
-	-	-	-	-	0	1	1	AP-Flash / Data-Flash Sector Erase 对用户的应用程序Flash区及数据Flash区扇区擦除

;5V单片机，应在3.6V以下时让其复位，3V单片机，应在2.4V以下时让其复位
;在此电压以下，此时再用ISP/IAP功能，编程/擦除Flash，不能保证能达到要求
;复位电路可选MAX810,STC810,STC6345,STC6344，813L,706P等

程序在系统ISP程序区时可以对用户应用程序区/数据Flash区(EEPROM)进行字节读/字节编程/扇区擦除；程序在用户应用程序区时，仅可以对数据Flash区(EEPROM)进行字节读/字节编程/扇区擦除。已经固化有ISP引导码，并设置为上电复位进入ISP的STC89C51RC/RD+系列单片机出厂时就已完全加密。

ISP_TRIG: ISP/IAP操作时的命令触发寄存器。
在ISPEN(ISP_CONTR.7) = 1时,对ISP_TRIG先写入46h,再写入B9h,
ISP/IAP命令才会生效。

ISP_CONTR: ISP/IAP 控制寄存器。

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

ISPEN: ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash
SWBS: 软件选择从用户主程序区启动（0），还是从 ISP 程序区启动（1）。
SWRST: 0： 不操作； 1： 产生软件系统复位，硬件自动清零。

设置等待时间			CPU 等待时间(机器周期)			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
0	1	1	6	30	5471	5MHz
0	1	0	11	60	10942	10MHz
0	0	1	22	120	21885	20MHz
0	0	0	43	240	43769	40MHz

STC89C51RC/RD+ 系列内部可用 Data Flash(EEPROM)的地址(与程序空间是分开的)：
有网友来电说不能 IAP 写数据，后检查原来他把地址指向了程序区，被单片机忽略
程序在用户应用程序区(AP区)时,仅可以对Data Flash(EEPROM)进行IAP/ISP操作,不可以修改自身，这是系统可靠的基础。

STC89C51RC,STC89LE51RC 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议一次同时修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

STC89C52RC,STC89LE52RC 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议一次同时修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2000h	21FFh	2200h	23FFh	2400h	25FFh	2600h	27FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
2800h	29FFh	2A00h	2BFFh	2C00h	2DFFh	2E00h	2FFFh	

STC89C54RD+,STC89LE54RD+ 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH	
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH	
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH	
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH	
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH	
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH	
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH	
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH	
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH	
第五十七扇区		第五十八扇区						
起始地址	结束地址	起始地址	结束地址					
F000h	F1FFh	F200h	F3FFh					

STC89C55RD+,STC89LE55RD+ 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH	
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH	
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH	
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH	
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH	
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH	
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH	
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH	
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH	
第五十七扇区		第五十八扇区						
起始地址	结束地址	起始地址	结束地址					
F000h	F1FFh	F200h	F3FFh					

STC89C58RD+,STC89LE58RD+ 单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区512字节 建议同一次修改的数据放在同一扇区
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8000h	81FFh	8200h	83FFh	8400h	85FFh	8600h	87FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
8800h	89FFh	8A00h	8BFFh	8C00h	8DFFh	8E00h	8FFFh	
第九扇区		第十扇区		第十一扇区		第十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9000h	91FFh	9200h	93FFh	9400h	95FFh	9600h	97FFh	
第十三扇区		第十四扇区		第十五扇区		第十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
9800h	99FFh	9A00h	9BFFh	9C00h	9DFFh	9E00h	9FFFh	
第十七扇区		第十八扇区		第十九扇区		第二十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A000h	A1FFh	A200h	A3FFh	A400h	A5FFh	A600h	A7FFh	
第二十一扇区		第二十二扇区		第二十三扇区		第二十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
A800h	A9FFh	AA00h	ABFFh	AC00h	ADFFh	AE00h	AFFFH	
第二十五扇区		第二十六扇区		第二十七扇区		第二十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B000h	B1FFh	B200h	B3FFh	B400h	B5FFh	B600h	B7FFH	
第二十九扇区		第三十扇区		第三十一扇区		第三十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
B800h	B9FFh	BA00h	BBFFh	BC00h	BDFFh	BE00h	BFFFH	
第三十三扇区		第三十四扇区		第三十五扇区		第三十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C000h	C1FFh	C200h	C3FFh	C400h	C5FFh	C600h	C7FFH	
第三十七扇区		第三十八扇区		第三十九扇区		第四十扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
C800h	C9FFh	CA00h	CBFFh	CC00h	CDFFh	CE00h	CFFFH	
第四十一扇区		第四十二扇区		第四十三扇区		第四十四扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D000h	D1FFh	D200h	D3FFh	D400h	D5FFh	D600h	D7FFH	
第四十五扇区		第四十六扇区		第四十七扇区		第四十八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
D800h	D9FFh	DA00h	DBFFh	DC00h	DDFFh	DE00h	DFFFH	
第四十九扇区		第五十扇区		第五十一扇区		第五十二扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E000h	E1FFh	E200h	E3FFh	E400h	E5FFh	E600h	E7FFH	
第五十三扇区		第五十四扇区		第五十五扇区		第五十六扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
E800h	E9FFh	EA00h	EBFFh	EC00h	EDFFh	EE00h	EFFFH	
第五十七扇区		第五十八扇区						
起始地址	结束地址	起始地址	结束地址					
F000h	F1FFh	F200h	F3FFh					

STC89C51 RC / RD+ 系列 IAP 应用汇编简介

STC89C51 RC / RD+ 系列 内部 EEPROM 的应用

有网友要求提供汇编示例，请尽量参考已调通的 C 原程序

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    ISP_DATA        DATA    0E2h;    或   ISP_DATA        EQU     0E2h
    ISP_ADDRH       DATA    0E3h;    或   ISP_ADDRH       EQU     0E3h
    ISP_ADDRL       DATA    0E4h;    或   ISP_ADDRL       EQU     0E4h
    ISP_CMD         DATA    0E5h;    或   ISP_CMD         EQU     0E5h
    ISP_TRIG        DATA    0E6h;    或   ISP_TRIG        EQU     0E6h
    ISP_CONTR       DATA    0E7h;    或   ISP_CONTR       EQU     0E7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ       EQU  1    ;字节读
    ISP_IAP_BYTE_PROGRAM    EQU  2    ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE    EQU  3    ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME       EQU  0            ;设置等待时间,40MHz 以下 0,20M 以下 1,
                                      ;          10MHz 以下 2,5M 以下 3
;字节读
    MOV  ISP_ADDRH,    #BYTE_ADDR_HIGH    ;    送地址高字节
    MOV  ISP_ADDRL,    #BYTE_ADDR_LOW     ;    送地址低字节
    CLR  EA        ;    关中断,此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_CONTR,    #WAIT_TIME         ;    设置等待时间
    ORL  ISP_CONTR,    #10000000B         ;    允许 ISP/IAP 操作
    MOV  ISP_CMD,      #ISP_IAP_BYTE_READ ;    送字节读命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #46h      ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #0B9h     ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                           ;数据读出到 ISP_DATA 寄存器后，CPU 继续执行程序
    MOV  ISP_CONTR,    #00000000B ;禁止 ISP/IAP 操作
    MOV  ISP_CMD,      #00000000B ;去除 ISP/IAP 命令
    MOV  ISP_TRIG,     #00000000B ;防止 ISP/IAP 命令误触发
    MOV  ISP_ADDRH,    #0         ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV  ISP_ADDRL,    #0         ;送地址低字节单元为 00,防止误操作
    SETB EA    ;    开中断，CPU 处理完 ISP/IAP 动作即可开中断
    MOV  A,            ISP_DATA   ;将读出的数据送往 Acc
```

```
;字节编程，该字节为 FFh/ 空时，可对其编程，否则不行,要先执行扇区擦除
    MOV  ISP_DATA,     #ONE_DATA          ;    送字节编程数据到 ISP_DATA
    MOV  ISP_ADDRH,    #BYTE_ADDR_HIGH    ;    送地址高字节
    MOV  ISP_ADDRL,    #BYTE_ADDR_LOW     ;    送地址低字节
    CLR  EA        ;    关中断,此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_CONTR,    #WAIT_TIME         ;    设置等待时间
    ORL  ISP_CONTR,    #10000000B         ;    允许 ISP/IAP 操作
    MOV  ISP_CMD,      #ISP_IAP_BYTE_PROGRAM   ;送字节编程命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #46h       ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #0B9h      ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                                   ;字节编程成功后，CPU 继续执行程序
    MOV  ISP_CONTR,    #00000000B         ;禁止 ISP/IAP 操作
    MOV  ISP_CMD,      #00000000B         ;去除 ISP/IAP 命令
    MOV  ISP_TRIG,     #00000000B         ;防止 ISP/IAP 命令误触发
    MOV  ISP_ADDRH,    #0                 ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV  ISP_ADDRL,    #0                 ;送地址低字节单元为 00,防止误操作
    SETB EA    ;    开中断，CPU 处理完 ISP/IAP 动作即可开中断
```

小常识： (STC 单片机的 Data Flash 当 EEPROM 功能使用)

3 个基本命令 - - - - 字节读，字节编程，扇区擦除

字节编程：如果该字节是“1111,1111B”,则可将其中的“1”编程为“0”，如果该字节中有位为“0”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0”变为“1”。

扇区擦除：只有“扇区擦除”才可能将“0”擦除为“1”。

大建议：

1.同一次修改的数据放在同一扇区中，单独修改的数据放在另外的扇区,就不须读出保护。

2.如果一个扇区只用一个字节，那就是真正的 EEPROM,STC 单片机的 Data Flash 比外部 EEPROM 要快很多，读一个字节 / 编程一个字节 / 擦除一个扇区大概是 10uS/60uS/10mS。

3.如果同一个扇区中存放了一个以上的字节，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在 STC 单片机的 RAM 中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

```
;扇区擦除，没有字节擦除，只有扇区擦除，512 字节 / 扇区，
;建议同一次修改的数据放在同一个扇区
;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机
;内部的 RAM 中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，所以每个扇区
;中用的字节数越少越好，操作起来越灵活越快
;强烈建议同一次修改的数据放在同一个扇区

    MOV   ISP_ADDRH,    #SECTOR_FIRST_BYTE_ADDR_HIGH      ;送扇区起始地址高字节
    MOV   ISP_ADDRL,    #SECTOR_FIRST_BYTE_ADDR_LOW       ;送扇区起始地址低字节
    CLR   EA            ;    关中断，此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV   ISP_CONTR,    #WAIT_TIME     ;设置等待时间
    ORL   ISP_CONTR,    #10000000B     ;允许 ISP/IAP
    MOV   ISP_CMD,      #ISP_IAP_SECTOR_ERASE             ;送扇区擦除命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV   ISP_TRIG,     #46h      ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV   ISP_TRIG,     #0B9h     ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                           ;扇区擦除成功后，CPU 继续执行程序
    MOV   ISP_CONTR,    #00000000B     ;禁止 ISP/IAP 操作
    MOV   ISP_CMD,      #00000000B     ;去除 ISP/IAP 命令
    MOV   ISP_TRIG,     #00000000B     ;防止 ISP/IAP 命令误触发
    MOV   ISP_ADDRH,    #0             ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV   ISP_ADDRL,    #0             ;送地址低字节单元为 00,防止误操作
```

```
;从用户应用程序区(AP 区)软件复位并切换到 ISP 程序区开始执行程序
    MOV   ISP_CONTR,    #01100000B     ;SWBS = 1(选择 ISP 区), SWRST = 1(软复位)

;从 ISP 程序区软件复位并切换到用户应用程序区(AP 区)开始执行程序
    MOV   ISP_CONTR,    #00100000B     ;SWBS = 0(选择 AP 区), SWRST = 1(软复位)

;使用 ISP/IAP 功能的朋友尽量给 13922805190（姚工）一个电话交流一下
;建议在打开 ISP 及在 ISP 触发送 46H,B9H 之前三个地方，各加些软件陷阱(欢迎来电讨论)
;5V 单片机，应在 3.6V 以下时让其复位，3V 单片机，应在 2.4V 以下时让其复位
;在此电压以下，此时再用 ISP/IAP 功能，编程 / 擦除 Flash，不能保证能达到要求
;复位电路可选 MAX810,STC810,STC6345,STC6344，813L,706P 等
```

```
;本程序演示 EEPROM/IAP
;-----------------------------------------------
;定义与 IAP 有关的特殊功能寄存器
ISP_DATA         EQU 0E2H
ISP_ADDRH        EQU 0E3H
ISP_ADDRL        EQU 0E4H
ISP_CMD          EQU 0E5H
ISP_TRIG         EQU 0E6H
ISP_CONTR        EQU 0E7H

;-----------------------------------------------
;定义常量
;------------------------
;Flash 操作等待时间
;ENABLE_ISP       EQU 83H             ;<5MHz
;ENABLE_ISP       EQU 82H             ;<10MHz
;ENABLE_ISP       EQU 81H             ;<20MHz
;ENABLE_ISP       EQU 80H             ;>20MHz

 DEBUG_DATA       EQU 5AH

;------------------------
;选择 MCU 型号
DATA_FLASH_START_ADDRESS EQU 2000H ;STC89C/LE52RC

;-----------------------------------------------
    ORG   0000H
    AJMP  main

;-----------------------------------------------
    ORG   0100H
main:
    MOV     P1,#0F0H ;演示程序开始工作
     LCALLDelay      ;延时
    MOV     P1,#0FH  ;演示程序开始工作
     LCALLDelay      ;延时

    MOV     SP,#0E0H ;堆栈指针指向 0E0H 单元
;***********************************************

;读回写入 flash 的第 1 个字节

MAIN1:
    MOV   DPTR, #DATA_FLASH_START_ADDRESS
    LCALL byte_read
    MOV     40H, A                ;值送 40H 单元保存
    CJNE  A, #DEBUG_DATA, DATA_NOT_EQU_DEBUG_DATA
```

```
DATA_IS_DEBUG_DATA:
    MOV     P1,  #01111111B  ; (DATA_FLASH_START_ADDRESS) = #5A, 亮P1.7
    LCALL Delay      ;延时
    MOV     A, 40H   ;值从40H 单元送 ACC
    CPL     A
    MOV     P1,A     ;数据是对的，送 P1显示
WAIT1:
    SJMP    WAIT1    ;数据是对的，送 P1显示，并在此停止

DATA_NOT_EQU_DEBUG_DATA:
    MOV     P1,  #11110111B ;(DATA_FLASH_START_ADDRESS) != #5A, 亮P1.3
     LCALLDelay      ;延时

    MOV     A,   40H ;值从40H 单元送 ACC
    CPL     A
    MOV     P1,  A   ;数据不对，送 P1显示
     LCALLDelay      ;延时

    MOV     DPTR,#DATA_FLASH_START_ADDRESS
    ACALL sector_erase  ;擦除扇区, (DATA_FLASH_START_ADDRESS) != #DEBUG_DATA

    MOV     DPTR, #DATA_FLASH_START_ADDRESS
    MOV     A,    #DEBUG_DATA           ;写入 flash 的数据为 DEBUG_DATA
    ACALL   byte_program                ;字节编程
    MOV     P1,   #11011111B            ;先亮P1.3 ,再亮 P1.5
WAIT2:
    SJMP    WAIT2        ;字节编程后在此停止

;*********************************************
;---------------------------------------------
;读一字节
;调用前需打开 IAP 功能
;入口:DPTR = 字节地址
;返回:A = 读出字节

byte_read:
    MOV     ISP_CONTR,  #ENABLE_ISP     ;打开 IAP 功能, 设置 Flash 操作等待时间
    MOV     ISP_CMD,    #01             ;Select Read AP Mode
    MOV     ISP_ADDRH,  DPH             ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV     ISP_ADDRL,  DPL
    CLR     EA
    MOV     ISP_TRIG,   #46H            ;Trigger ISP processing
    MOV     ISP_TRIG,   #0B9H           ;Trigger ISP processing
    NOP
    MOV     A,   ISP_DATA               ;数据在ISP_DATA
    SETB    EA
;Now in processing.(CPU will halt here before completing)
```

```
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET

;-----------------------------------------------

;字节编程
;调用前需打开 IAP 功能
;入口:DPTR = 字节地址，A= 须编程字节的数据


byte_program:

    MOV   ISP_CONTR, #ENABLE_ISP    ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   ISP_CMD, #02H             ;Select Byte Program Mode
    MOV   ISP_ADDRH, DPH            ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV   ISP_ADDRL, DPL
    MOV   ISP_DATA, A               ;数据进 ISP_DATA
    CLR   EA
    MOV   ISP_TRIG, #46H            ;Trigger ISP processing
    MOV   ISP_TRIG, #0B9H           ;Trigger ISP processing
    NOP
    SETB  EA
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET
;-----------------------------------------------

;擦除扇区，入口:DPTR = 扇区地址
sector_erase:

    MOV   ISP_CONTR, #ENABLE_ISP    ;打开 IAP 功能，设置 Flash 操作等待时间
    MOV   ISP_CMD, #03H             ;Select Page Erase Mode
    MOV   ISP_ADDRH, DPH            ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV   ISP_ADDRL, DPL
    CLR   EA
    MOV   ISP_TRIG, #46H            ;Trigger ISP processing
    MOV   ISP_TRIG, #0B9H           ;Trigger ISP processing
    NOP
    SETB  EA
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET

;-----------------------------------------------

trigger_ISP:
    CLR   EA
    MOV   ISP_TRIG, #46H            ;Trigger ISP processing
    MOV   ISP_TRIG, #0B9H           ;Trigger ISP processing
    NOP
```

```
    SETB  EA
    RET

;-------------------------------------------------
IAP_Disable:                        ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    MOV   ISP_CONTR, #0             ;关闭 IAP 功能
    MOV   ISP_CMD, #0
    MOV   ISP_TRIG, #0
    RET

;-------------------------------------------------

Delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;-------------------------------------------------

    END
;*************************************************
```

STC89C51 RC/RD+ 系列单片机交直流特性

ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings

Parameter	Symbol	MIN	MAX	UNIT
Storage temperature	T_{ST}	-55	+125	℃
Operating Temperature(I)	T_A	-40	+85	℃
Operating Temperature(C)	T_A	0	+70	℃
DC Power Supply(5V MCU)	V_{DD} - V_{SS}	-0.3	+6.0	V
DC Power Supply(3V MCU)	V_{DD} - V_{SS}	-0.3	+4.0	V
Voltage on any Pin		-0.5	+5.5	V

DC Specification(5V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	3.8	5.0	5.5	V	
I_{PWDN}	Power Down Current		0.4		uA	5V
I_{IDLE}	Idle Current		2.0		mA	5V
I_{CC}	Operating Current		4 mA	20	mA	5V
V_{IL1}	Input low voltage (P0,1,2,3,4)			0.8	V	5V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.5	V	5V
V_{IH1}	Input High voltage (P0,1,2,3,4,/EA)	2.0			V	5V
V_{IH2}	Input High voltage (RESET)	3.0			V	5V
I_{OL1}	Sinking Current for Output Low (P1,P2,P3,P4)	4	6		mA	5V
I_{OL2}	Sinking Current for Output Low (P0,ALE,PSEN)	8	12		mA	5V
I_{OH1}	Sourcing Current for Output High (P1,P2,P3,P4)	150	220		uA	5V
I_{OH2}	Sourcing Current for Output High (ALE,PSEN)	14	20		mA	5V
I_{IL}	Logic 0 input current (P1,2,3,4)		18	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P1,2,3,4)		270	600	uA	V_{PIN}=2V

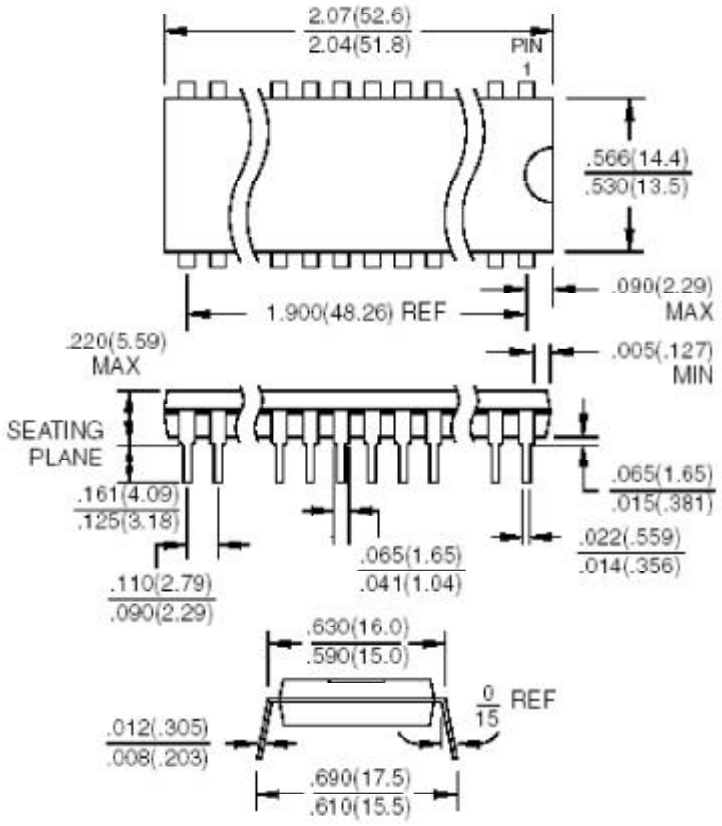
DC Specification(3.3V MCU)

Symbol	Parameter	Specification				Test Condition
		Min.	Typ.	Max.	Unit	
V_{DD}	Operating Voltage	1.8	3.3	3.6	V	
I_{PWDN}	Power Down Current		0.4		uA	3.3V
I_{IDLE}	Idle Current		2.0		mA	3.3V
I_{CC}	Operating Current		4 mA	15	mA	3.3V
V_{IL1}	Input low voltage (P0,1,2,3,4)			0.8	V	3.3V
V_{IL2}	Input low voltage (RESET,XTAL1)			1.5	V	3.3V
V_{IH1}	Input High voltage (P0,1,2,3,4,/EA)	2.0			V	3.3V
V_{IH2}	Input High voltage (RESET)	3.0			V	3.3V
I_{OL1}	Sinking Current for Output Low (P1,P2,P3,P4)	2.5	4		mA	3.3V
I_{OL2}	Sinking Current for Output Low (P0,ALE,PSEN)	5	8		mA	3.3V
I_{OH1}	Sourcing Current for Output High (P1,P2,P3,P4)	40	70		uA	3.3V
I_{OH2}	Sourcing Current for Output High (ALE,PSEN)	8	13		mA	3.3V
I_{IL}	Logic 0 input current (P1,2,3,4)		8	50	uA	V_{PIN}=0V
I_{TL}	Logic 1 to 0 transition current (P1,2,3,4)		110	600	uA	V_{PIN}=2V

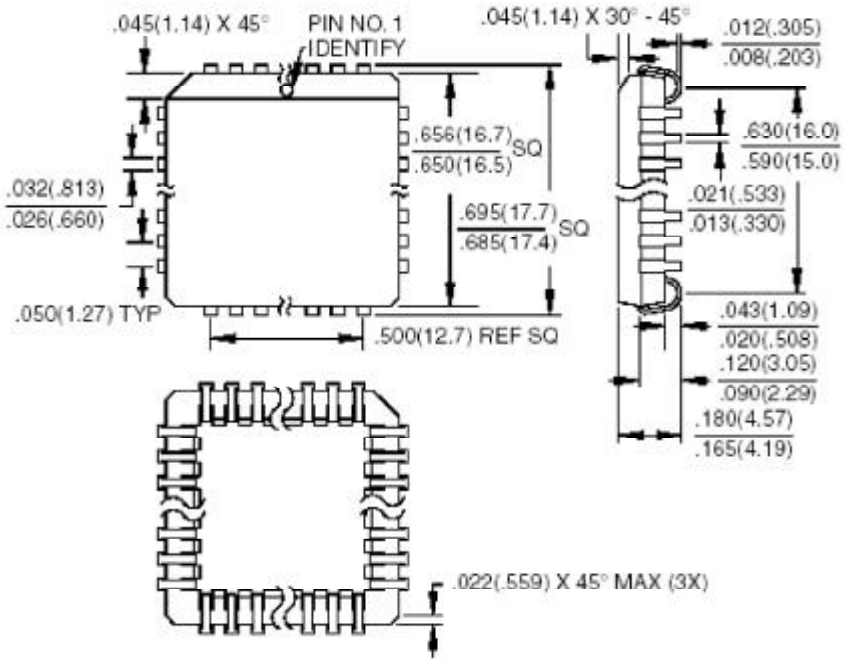
STC 8051 封装尺寸图

Packaging Information

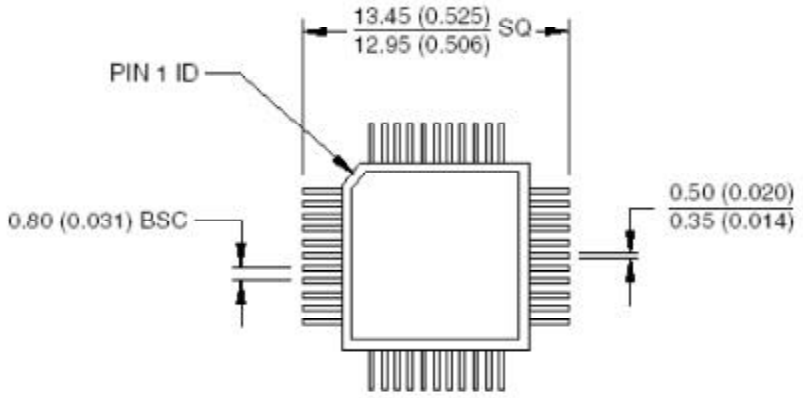
40P6, 40-lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)

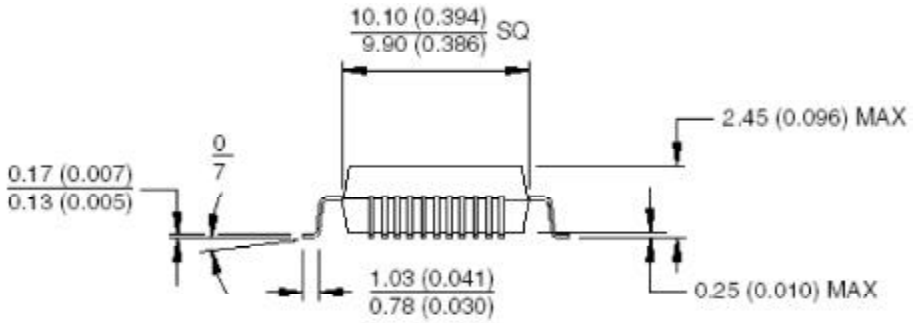


44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-018 AC



44Q, 44-lead, Plastic Quad Flat Package (PQFP)
Dimensions in Millimeters and (Inches)*
JEDEC STANDARD MS-022 AB

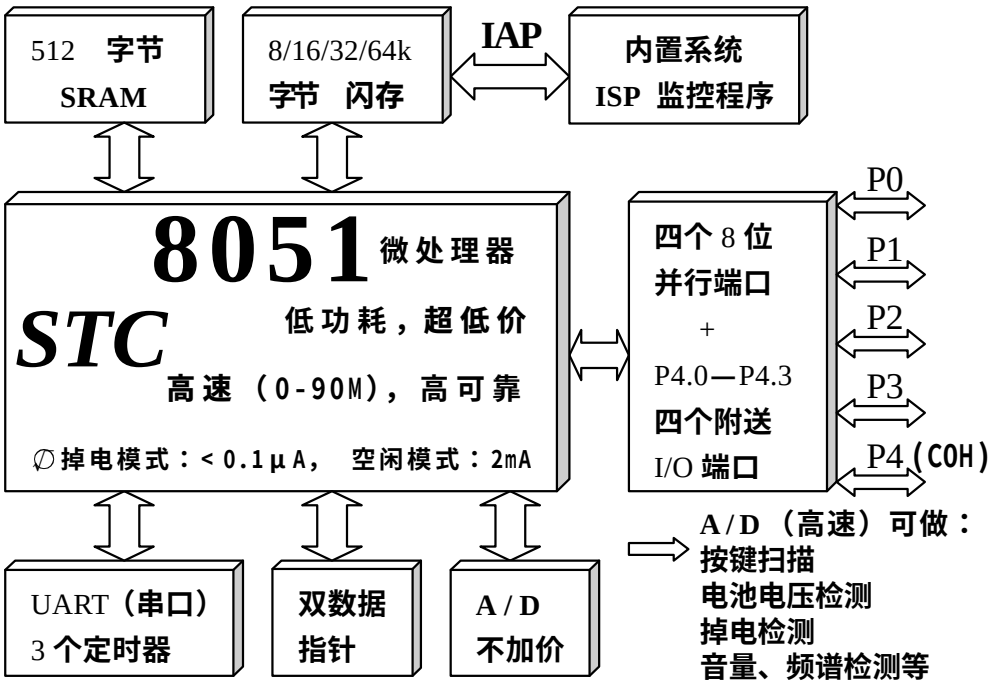




Controlling dimension: millimeters

附录 A:

STC89LE516AD 系列单片机手册



STC 具有 A/D 转换功能的单片机选型指南

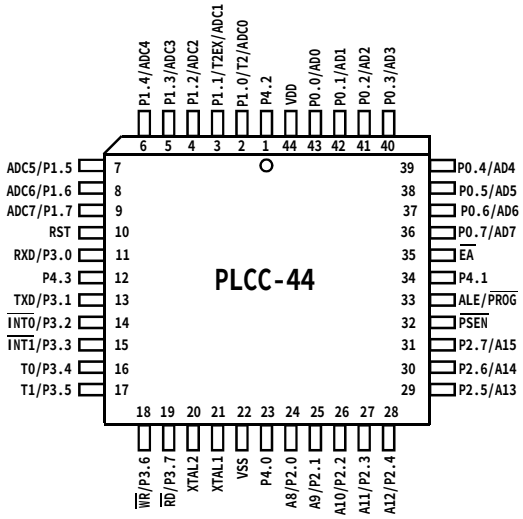
型 号	最高时钟频率Hz 1.9-3.6V	程序存储器	RAM	降低EMI	双倍速	P4口	ISP	A/D	供货
STC89LE516 AD	0-90M	64K	512	√		√	√	√	现货
STC89LE58 AD	0-90M	32K	512	√		√	√	√	现货
STC89LE54 AD	0-90M	16K	512	√		√	√	√	现货
STC89LE52 AD	0-90M	8K	512	√		√	√	√	现货
STC89LE51 AD	0-90M	4K	512	√		√	√	√	定货
STC89LE516 X2	0-90M	64K	512	√	√	√	√	√	现货

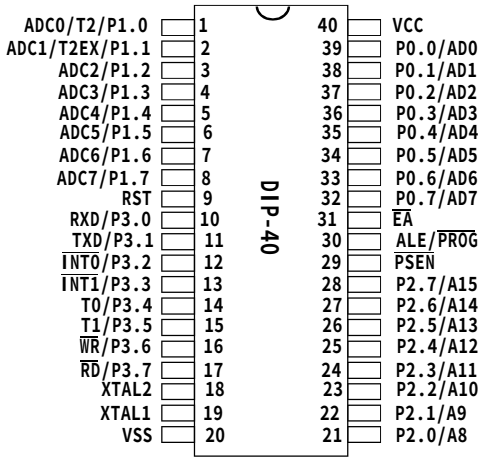
特殊功能寄存器映像 SFR Mapping

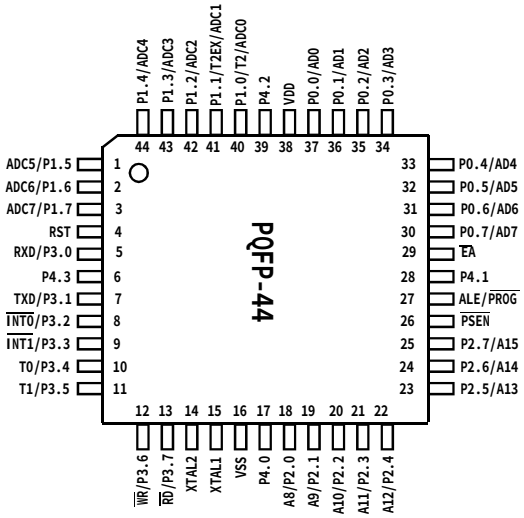
STC89LE516AD,STC89LE58AD,STC89LE54AD,STC89LE52AD,STC89LE51AD为单倍速
因AD系列用户ISP设不了双倍速，故有一STC89LE516AD的双倍速版本(6T)STC89LE516X2是同一芯片，只是出厂时就设为双倍速了。

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h									FFh
F0h	B 0000,0000								F7h
E8h									EFh
E0h	ACC 0000,0000								E7h
D8h									DFh
D0h	PSW 0000,0000								D7h
C8h	T2CON 0000,0000	T2MOD xxxx,xx00	RCAP2L 0000,0000	RCAP2H 0000,0000	TL2 0000,0000	TH2 0000,0000			CFh
C0h	P4 xxxx,1111					ADC_CONTR xxx0,0000	ADC_DATA xxxx,xxxx		C7h
B8h	IP x000,0000	SADEN 0000,0000							BFh
B0h	P3 1111,1111							IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR 0000,0000							AFh
A0h	P2 1111,1111		AUXR1 xxxx,xxx0						A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111							P1_ADC_EN 0000,0000	97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR xxxx,xxx0		8Fh
80h	P0 1111,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000				PCON 0xx1,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

STC89LE516AD / X2 系列单片机 管脚图

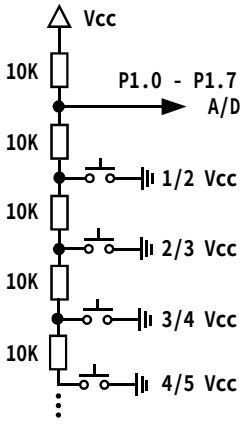






A/D 转换典型应用线路：

按键扫描



工作电压：3.6V - 2.4V， 可低至2.4V - 1.9V

I/O口： P0,P2,P3,P4口可以承受5V输入。

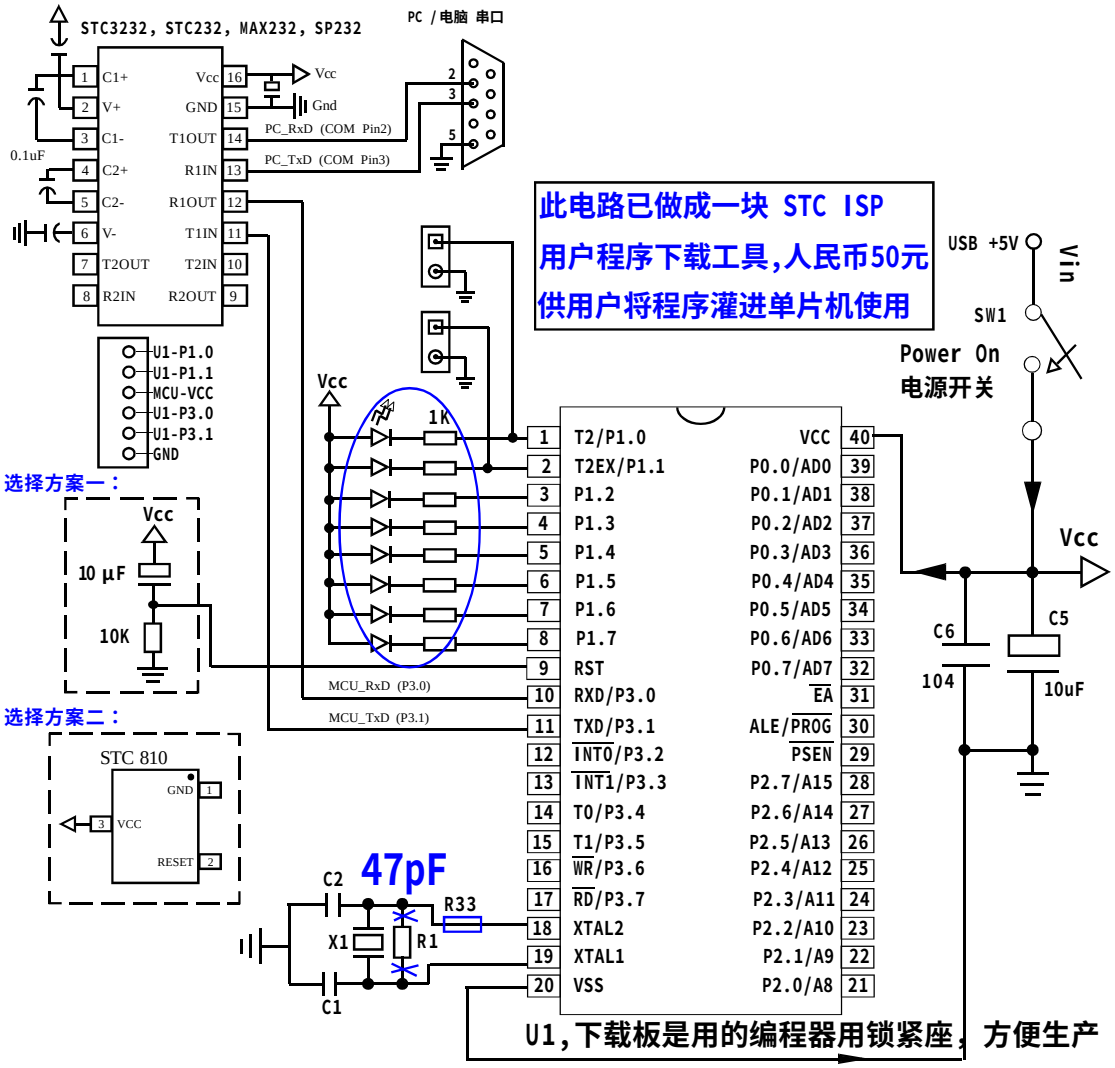
P1口不要接到5V系统,只能接3.3V系统.

P4口： 地址在C0H，P4.3 - P4.0

A/D转换在P1口，P1.0 - P1.7八路。

STC89LE516AD 系列单片机 ISP 下载编程典型应用电路

STC 单片机在线编程线路， *STC* RS-232 转换器



R33 = 0 - 160 欧姆，推荐用此电阻，可以不用								
X1	2-3MHz	4-9MHz	10-18MHz	19-26MHz	27-34MHz	35-39MHz	40-43MHz	44-48MHz
C1,C2	47pF	20pF	10pF	不用	10pF	10pF	10pF	5pF
R1	不用	不用	不用	不用	6.8K	5.1K	3.3K	3.3K

任何一种复位后，如 P1.1/P1.0 = 0,0 则进入系统 ISP 监控程序。
任何一种复位后，如 P1.1,P1.0 不同时为 0，则不进入系统 ISP 监控程序。
注意与 STC89C51RC/RD+ 系列单片机的不同,RC/RD+ 系列是冷启动进入系统 ISP 监控程序。
使用 STC89LE516AD 系列单片机时,尽量将 P1.0/P1.1 中的一个管脚空着，免得不需要进入系统 ISP 监控程序时，由于 P1.1/P1.0 = 0,0 复位后进入系统 ISP 监控程序。

1.STC89LE516AD 系列单片机扩展 AUX-RAM 的使用

STC89LE516AD/X2 系列单片机内部的 RAM 为 512 字节(256+256),即扩展了 256 字节的 AUX-RAM,其访问方式为:

汇编语言: (访问内部扩展的 256 字节 AUX-RAM)

```
MOVX   @Ri,  A   ;   将累加器 A 的值送至 @Ri 指向的单元，i = 0，1
MOVX   A,    @Ri ;   将 @Ri 指向的单元的值读到累加器 A，i = 0，1
```

STC89LE516AD/X2 系列单片机用“MOVX A，@Ri”,“MOVX A，@Ri”指令访问不到外部 64K 数据空间。

汇编语言: (访问外部 64K 数据空间)

```
MOVX   @DPTR,  A     ;   将累加器 A 的值送至 @DPTR 指向的单元
MOVX   A,      @DPTR ;   将 @DPTR 指向的单元的值读到累加器 A
```

STC89LE516AD/X2 系列单片机用“MOVX A，@DPTR”,“MOVX A，@DPTR”指令访问不到内部扩展的 256 字节 AUX-RAM。

C 语言:

用 pdata 声明的变量访问单片机内部扩展的 256 字节 AUX-RAM
用 xdata 声明的变量访问单片机外部 64K 数据空间

2.双数据指针 及 AUXR1 寄存器

AUXR1	A2h	Auxiliary Register 1	-	-	-	-	-	-	-	DPS	xxxx,xxx0

DPS = 0 时选择 DPTR0， DPS = 1 时选择 DPTR1
可以用 “INC AUXR1” 快速切换 DPTR0 / DPTR1

3.禁止 ALE 输出 及 AUXR1 寄存器

AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	-	ALEOFF	xxxx,xxx0

ALEOFF = 0，ALE 信号正常输出。
ALEOFF = 1，禁止 ALE 信号输出。但在访问外部数据空间及外部程序空间时有信号输出。

4.中断 及 中断优先级控制寄存器 IP / IPH

中断与普通 8052 完全兼容，优先级可设为 4 级,通过增加的 IPH 寄存器

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	-	ET2	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	-	PT2	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	-	PT2H	PSH	PT1H	PX1H	PT0H	PX0H	xx00,0000

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	1	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	2	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	3	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	4	PT1H,PT1	0,0	0,1	1,0	1,1	IF1
UART	0023H	5	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
Timer 2	002BH	6	PT2H,PT2	0,0	0,1	1,0	1,1	TF2 + EXF2

4.A/D 及 A/D转换寄存器 ADC_CONTR/ADC_DATA/P1_ADC_EN

STC89LE516AD/X2 在 P1 口，有 8 位精度的高速 A/D 转换器，P1.7 - P1.0 共 8 路电压输入型，可做按键扫描，电池电压检测，频谱检测等。17 个机器周期可完成一次转换，时钟在 40MHz 以下时。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1_ADC_EN	97h	允许P1.x成为A/D口	ADC_P17	ADC_P16	ADC_P15	ADC_P14	ADC_P13	ADC_P12	ADC_P11	ADC_P10	0000,0000
ADC_CONTR	C5h	A/D 转换控制寄存器	-	-	-	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	xxx0,0000
ADC_DATA	C6h	A/D 转换结果寄存器	-	-	-	-	-	-	-	-	0000,0000

P1_ADC_EN 特殊功能寄存器： P1.x 作为 A/D 转换输入通道来用允许特殊功能寄存器

允许P1.x成为A/D口	ADC_P17	ADC_P16	ADC_P15	ADC_P14	ADC_P13	ADC_P12	ADC_P11	ADC_P10	0000,0000

相应位为“1”时，对应的 P1.x 口作为 A/D 转换使用，内部上拉电阻自动断开

ADC_CONTR 特殊功能寄存器： A/D 转换控制特殊功能寄存器

A/D 转换控制寄存器	-	-	-	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	xxx0,0000

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Input Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用
1	1	1	选择 P1.7 作为 A/D 输入来用

ADC_START： 模拟 / 数字转换(ADC)启动控制位，设置为“1”时，开始转换

ADC_FLAG： 模拟 / 数字转换结束标志位，当 A/D 转换完成后，ADC_FLAG = 1。

ADC_DATA 特殊功能寄存器： A/D 转换结果特殊功能寄存器

A/D 转换结果寄存器	-	-	-	-	-	-	-	-	0000,0000

模拟 / 数字转换结果计算公式如下： 结果 = 256 x Vin / Vcc

Vin 为模拟输入通道输入电压，Vcc 为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

```
/* ------------- 宏晶科技   2005/1/5      A/D 转换 C 语言示例 ------------ */
/* ---Mobile: 13922805190, Tel: 0755 - 82908285,  Fax: 0755 - 82944243  */
/* ----Website: www.mcu-memory.com   Email: support@dsp-memory.com  --- */

//   ADC DEMO 程序演示 STC89LE516AD/X2 系列 MCU 的 A/D 转换功能。 时钟   11.0592MHz
//  转换结果以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。

#include <reg52.H>
#include <intrins.H>

//定义与 ADC 有关的特殊功能寄存器
sfr   P1_ADC_EN       =    0x97;              // A/D 转换功能允许寄存器
sfr   ADC_CONTR       =    0xC5;              // A/D 转换控制寄存器
sfr   ADC_DATA        =    0xC6;              // A/D 转换结果寄存器

typedef    unsigned char   INT8U;
typedef    unsigned int    INT16U;

void delay(INT8U delay_time)            // 延时函数
{
    INT8U       n;
    INT16U      m;
    for (n=0; n<delay_time; n++)
    {
        for (m=0; m<10000; m++);
    }
}

void initiate_RS232 (void)          // 串口初始化
{
    ES = 0;                         // 禁止串口中断
    SCON = 0x50;                    // 0101,0000  8 位数据位， 无奇偶校验
    T2CON = 0x34;                    // 0011，0100， 由 T2 作为波特率发生器

    RCAP2H = 0xFF;                  // 时钟 11.0592MHz， 9600 波特率
    RCAP2L = 0xDB;

    ES = 1;                         // 允许串口中断
}

void Send_Byte(INT8U one_byte)      // 发送一个字节
{
    TI = 0;                         // 清零串口发送中断标志
    SBUF = one_byte;
    while (TI == 0);
    TI = 0;                         // 清零串口发送中断标志
}
```

```
INT8U get_AD_result(INT8U channel)
{
    INT8U AD_finished      =    0;            // 存储 A/D 转换标志
    ADC_DATA    =     0;
    ADC_CONTR  = channel;                  //选择 A/D 当前通道
    delay(1);                              //使输入电压达到稳定
    ADC_CONTR |= 0x08;                     //0000,1000 令 ADC_START = 1，启动A/D转换
    AD_finished = 0;
    while ( AD_finished == 0 )             //等待A/D转换结束
    {
        AD_finished = (ADC_CONTR & 0x10);  //0001,0000，ADC_FLAG ==1 测试A/D转换结束否
    }
    ADC_CONTR &= 0xF7;                    //1111,0111 令 ADC_START = 0，关闭A/D转换，
    return (ADC_DATA);                    // 返回 A/D 转换结果
}
void main()
{
    initiate_RS232();
    P1     =     P1    |     0x63;    // 0110,0011,要设置为 A/D 转换的P1.x口,先设为高
    P1_ADC_EN = 0x63;       //0110,0011，P1 的P1.0,P1.1,P1.5,P1.6设置为 A/D 转换输入脚
                            //断开P1.0,P1.1,P1.5,P1.6内部上拉电阻
    while(1)
    {
        Send_Byte(get_AD_result(0));  //P1.0 为 A/D 当前通道，测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(1));  //P1.1 为 A/D 当前通道，测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(5));  //P1.5 为 A/D 当前通道，测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(6));  //P1.6 为 A/D 当前通道，测量并发送结果
        delay(0x200);

        Send_Byte(0);                //连续发送 4 个 00H，便于观察输出显示
        Send_Byte(0);
        Send_Byte(0);
        Send_Byte(0);

        delay(0x200);                //延时
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
    }
}
```

附录 B:

为什么少数用户的普通 8051 程序烧录后，不能运行

1．增强型的 STC89C58RD+，STC89C52RC 系列单片机内部扩展了 AUX-RAM.

型号	内部扩展RAM	占外部64K数据空间	MOVX @DPTR / xdata	MOVX @Ri / pdata
STC89C51RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE51RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C52RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE52RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C53RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89LE53RC	256 字节	0000H - 00FFH	地址>=100H,才访问外部	只访问内部扩展RAM
STC89C54RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE54RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89C58RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE58RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89C516RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE516RD+	1024 字节	0000H - 03FFH	地址>=400H,才访问外部	只访问内部扩展RAM
STC89LE516AD	256 字节		0000-FFFFH,均访问外部	只访问内部扩展RAM
STC89LE516X2	256 字节		0000-FFFFH,均访问外部	只访问内部扩展RAM

STC89C52RC / STC89C58RD+系列

如果用户的单片机 P0 / P2 口是作为普通 I/O 口(输入 / 输出口)用的，无冲突

如果用户的单片机 P0 / P2 口是作为总线扩展用的，外部扩展器件的地址在内部扩展 RAM 之上，无冲突

如果用户的单片机 P0 / P2 口是作为总线扩展用的，访问外部扩展器件的地址在内部扩展 RAM 的范围内，则访问的是内部扩展的 AUX-RAM,所以有些系统用户要禁止内部扩展 RAM

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
AUXR	8Eh	Auxiliary Register 0	-	-	-	-	-	-	EXTRAM	ALEOFF	xxxx,xx00

将 AUXR 寄存器的 EXTRAM 设为 1，即可禁止内部扩展的 AUX-RAM，使之与标准 8052 一样。

STC89LE516AD / STC89LE516X2 系列 AUXR 寄存器无 EXTRAM 控制位，通过寻址方式区分，“ MOVX @Ri ”固定访问内部扩展 RAM,“ MOVX， @DPTR ”固定访问外部扩展 64K 数据空间，详见相应部分说明。

2.晶振 / 时钟电路

STC89C52RC / STC89C58RD+ 现系列时钟电路部分请参照相应部分典型应用线路，用示波器查看时钟是否正常。

3.复位电路

电阻 / 电容复位的值为 电阻 = 10k，电容 = 10uF。如为用外部专用芯片复位，RESET 管脚(复位脚)不要加任何上拉 / 下拉电阻。详见相应部分说明

附录 C:

STC89C51 RC / RD+ 系列 单片机 ISP (DIY)

自己动手写自己的 ISP,写自己的远程升级程序还需了解的信息,只提供给策略性伙伴

用户程序区空间和 ISP 程序区空间:

型号	用户应用程序区	ISP 引导区空间(ISP Space)			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C51RC	0000H - 0FFFH	目前版本禁止ISP/IAP操作	3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89LE51RC	0000H - 0FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89C52RC	0000H - 1FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH
STC89LE52RC	0000H - 1FFFH		3800H - 3BFFH	3400H - 3BFFH	2C00H - 3BFFH

型号	用户应用程序区	ISP 引导区空间(ISP Space)			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C53RC	0000H - 3BFFH	目前版本禁止ISP/IAP操作			
STC89LE53RC	0000H - 3BFFH				
STC89C53RC	0000H - 37FFH		3800H - 3BFFH		
STC89LE53RC	0000H - 37FFH		3800H - 3BFFH		
STC89C53RC	0000H - 33FFH			3400H - 3BFFH	
STC89LE53RC	0000H - 33FFH			3400H - 3BFFH	
STC89C53RC	0000H - 2BFFH				2C00H - 3BFFH
STC89LE53RC	0000H - 2BFFH				2C00H - 3BFFH

型号	用户应用程序区	ISP 引导区空间(ISP Space),从ISP区启动MCU才有IAP功能			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C54RD+	0000H - 3FFFH	目前版本禁止ISP/IAP操作	F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89LE54RD+	0000H - 3FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89C58RD+	0000H - 7FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH
STC89LE58RD+	0000H - 7FFFH		F800H - FBFFH	F400H - FBFFH	EC00H - FBFFH

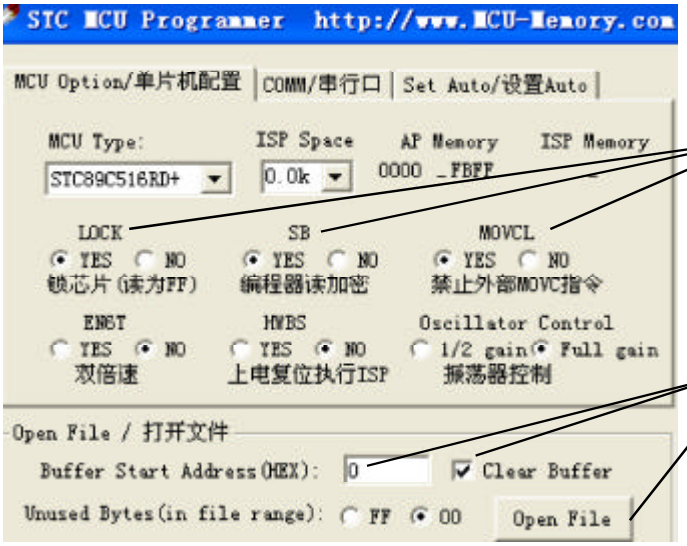
型号	用户应用程序区	ISP 引导区空间(ISP Space),从ISP区启动MCU才有IAP功能			
	AP Space 地址	0 K	1K(ISP/IAP)	2 K(ISP/IAP)	4K(ISP/IAP)
STC89C516RD+	0000H - FBFFH	目前版本禁止ISP/IAP操作			
STC89LE516RD+	0000H - FBFFH				
STC89C516RD+	0000H - F7FFH		F800H - FBFFH		
STC89LE516RD+	0000H - F7FFH		F800H - FBFFH		
STC89C516RD+	0000H - F3FFH			F400H - FBFFH	
STC89LE516RD+	0000H - F3FFH			F400H - FBFFH	
STC89C516RD+	0000H - EBFFH				EC00H - FBFFH
STC89LE516RD+	0000H - EBFFH				EC00H - FBFFH

动手写自己的 ISP 还必需要的工具,只提供给策略性伙伴

-----STC89C51RC / RD+ 系列单片机专用烧录器（200 元）

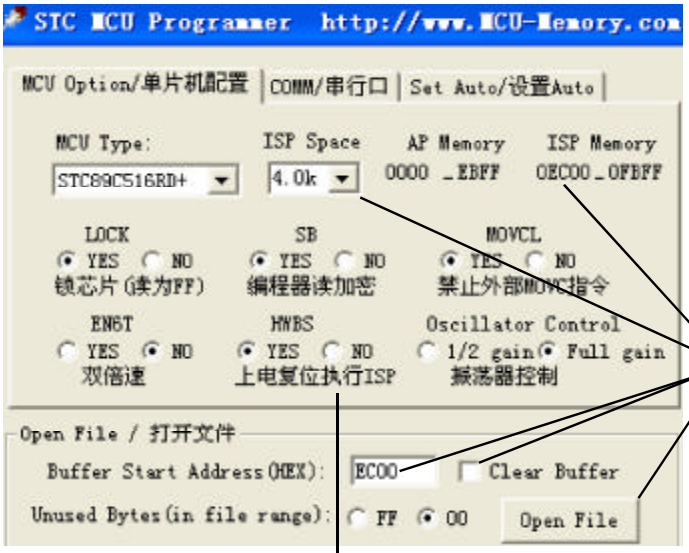
STC89C51RC / RD+ 系列单片机出厂时一般都固化有 ISP 引导码程序，如只烧录普通的用户程序，则只需将P3.0/P3.1经过RS-232转换器连到电脑的RS-232串口，通过STC-ISP下载软件打开用户程序下载就可以了。如果不用 STC 的 ISP 程序而要编写自己的 ISP 程序，则需要 STC89C51RC / RD+ 系列单片机专用烧录器，才可将用户的 ISP 程序烧录进单片机内部,软件使用 STC MCU Programmer 编程工具。

方式一：用户主应用程序和 ISP 程序一起烧录，上电复位后程序从 ISP 区开始运行



一．三个加密项全部加密

二．缓冲区从 0 开始，清缓冲区调入用户主应用程序



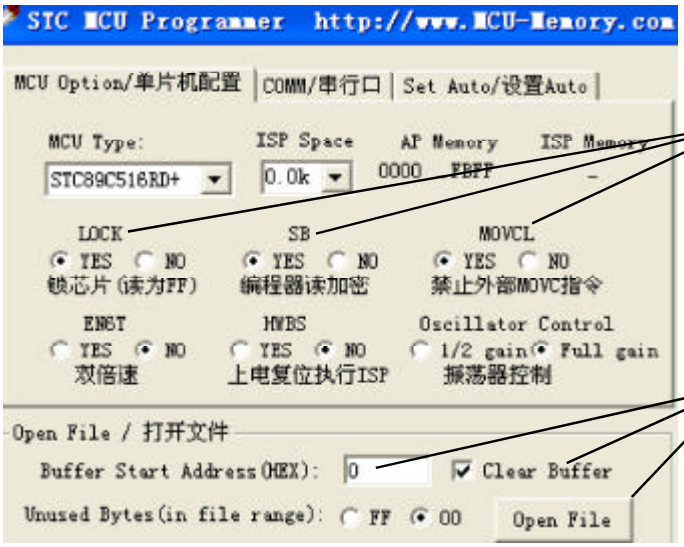
三．选定 ISP 空间从那里开始，不清缓冲区调入 ISP 程序,保留已调入的程序，用户 ISP空间可选0/1K/2K/4K

四．选择从 ISP 区启动

五．烧录程序

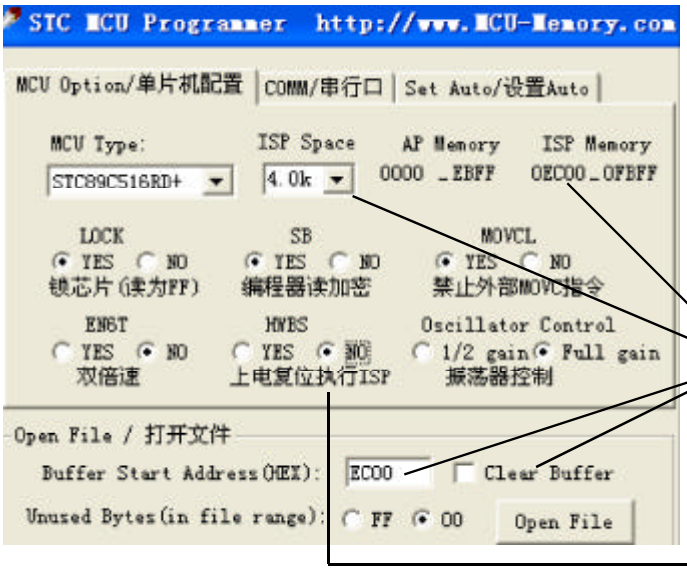
也可直接调入 ISP 程序至 ISP 区，选择从 ISP 程序区启动，再烧录。用户主应用程序由用户自己编的 ISP 程序下载。还可以把 ISP 区当成用户程序区，把余下的空间当成 Data Flash 数据空间。如 STC89C516RD+,4K ISP 区当用户程序使用，余下 59K 当 EEPROM 用。

方式二：用户主应用程序和 ISP 程序一起烧录，上电复位后程序从用户主应用程序区开始运行(但单片机 ISP 空间大小必须要选择 1K/2K/4K，否则只能当成无 ISP/IAP 的单片机使用)



一．三个加密项全部加密

二．缓冲区从 0 开始，清缓冲区调入用户主应用程序



三．选定 ISP 空间从那里开始，不清缓冲区调入用户自己的 ISP 程序，保留已调入的程序

四．选择从用户主应用程序区启动。，

五．烧录程序

已加过密的单片机，或已设置过 ISP 空间为 1k/2k/4k，上电复位从 ISP 空间启动的单片机，重新烧录程序时，要先将整个芯片擦除，再停电(单片机电放光后)，再上电，再重烧程序，才会写对。擦除整个芯片后，重新允许读出芯片内容的状态，是停电后再上电才能生效。

用户写自己的 ISP 程序的格式：

```
    ISP_DATA   EQU  0E2H     ;申明 ISP_DATA 寄存器地址，或 ISP_DATA  DATA 0E2H
    ISP_ADDRH  EQU  0E3H     ;申明 ISP_ADDRH 寄存器地址，或用 DATA 申明
    ISP_ADDRL  EQU  0E4H     ;申明 ISP_ADDRL 寄存器地址，或用 DATA 申明
    ISP_CMD    EQU  0E5H     ;申明 ISP_CMD 寄存器地址，或用 DATA 申明
    ISP_TRIG   EQU  0E6H     ;申明 ISP_TRIG 寄存器地址，或用 DATA 申明
    ISP_CONTR  EQU  0E7H     ;申明 ISP_CONTR 寄存器地址，或用 DATA 申明


    ORG        0000H;    ISP 程序的入口地址，逻辑上也是从“0”开始
    LJMP       INIT_MCU
    ORG        0003H;    ISP 程序的外部中断 0 入口地址，逻辑上是从“0003H”开始
    LJMP       INT_0
    ORG        000BH;    ISP 程序定时器 0 中断入口地址，逻辑上是从“000BH”开始
    LJMP       TIMER_0
    ORG        0023H;    ISP 程序的串口中断入口地址，逻辑上也是从“0023H”开始
    LJMP       Serial_Port_INT

    ORG        0050H;    初始化单片机
INIT_MCU:
    MOV        SP, #0E0H;
    ......
    ......

;收到外部命令要更新“用户主应用程序区”（AP 区）
    ......
UPDATE_USER_AP_CODE:
    ......  ; 更新用户程序区(AP),用 ISP/IAP 对用户 AP 区操作，物理地址从 0 开始

From_ISP_to_AP_Soft_Reset:
    MOV        ISP_CONTR,    #00100000B;更新完用户程序区后,从ISP区软复位到AP区
    END  ASM
;将以上 ISP 程序烧录到单片机的 ISP 区就可已了。
-------------------------------------------------------------
也可以在用户程序区收到命令后，从 AP 区软复位到 ISP 区。
From_AP_to_ISP_Soft_Reset:
    MOV        ISP_CONTR,    #01100000B;在AP区收到命令后，从AP区软复位到ISP区
```

附录D： 如何实现运行中自定义下载,无仿真器时方便调试

自定义下载原理：STC-ISP.exe 软件

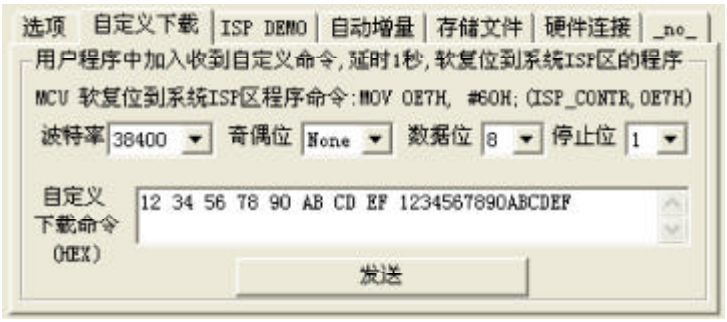
1.STC-ISP.exe 软件,可由用户设置，按 UART/RS-232 的格式向用户程序发送命令

波特率： 38400bps， 19200bps， 9600bps， 4800bps， 2400bps， 1200bps 等

奇偶校验位： 无， 偶校验， 奇校验

数据位几位： 8 位， 7 位，6 位，5 位， 等

停止位几位： 1 位，1.5 位，2 位，等



2.需向用户程序发送的命令用户可在上图自定义下载命令输入窗口中输入(HEX)

命令之间建议加一个空格，也可不加，STC-ISP.exe 会处理，上例为发送自定义命令 12H,34H,56H,78H,0ABH,0CDH,0EFH， 12H,34H,56H,78H,0ABH,0CDH,0EFH

3.用户程序中应加入收到自定义下载命令，延时一秒，软复位到系统 ISP 程序区的程序

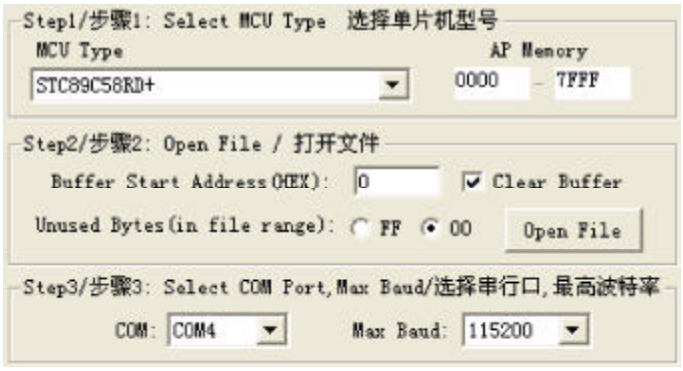
4.将以上含有接收自定义下载命令的用户程序先用老方法下载进 STC 单片机内部：

STC89C51RC,STC89C52RC,STC89C53RC,STC89C54RD+,STC89C58RD+,STC89C516RD+

STC89LE52RC,STC89LE53RC,STC89LE54RD+,STC89LE58RD+,STC89LE516RD+

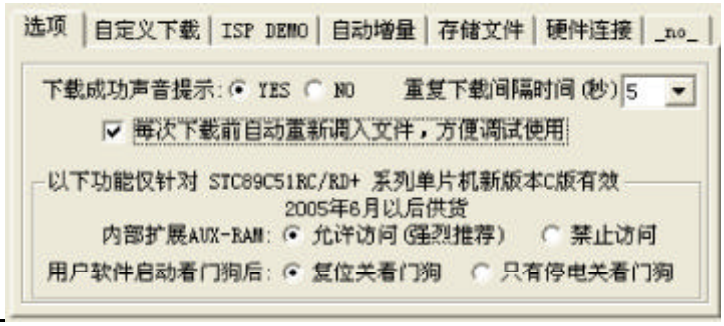
这样以上型号的 STC 单片机就具有了“ 不停电 / 运行中 / 自定义下载功能 ”

6 . 以下选好型号，打开文件. . . ,在自定义下载中设置相关选项，选择“ 发送 ”即可，



STC-ISP.exe 在“ 发送 ”完用户自定义下载命令后,就会转去调用老的那一套下载命令,而不管用户单片机程序收到命令没有。不过这个转换有些电脑有时需要将近1S的时间,所以用户应用程序要延时1S，否则系统 ISP 程序收不到下载命令,又会回到用户应用程序。

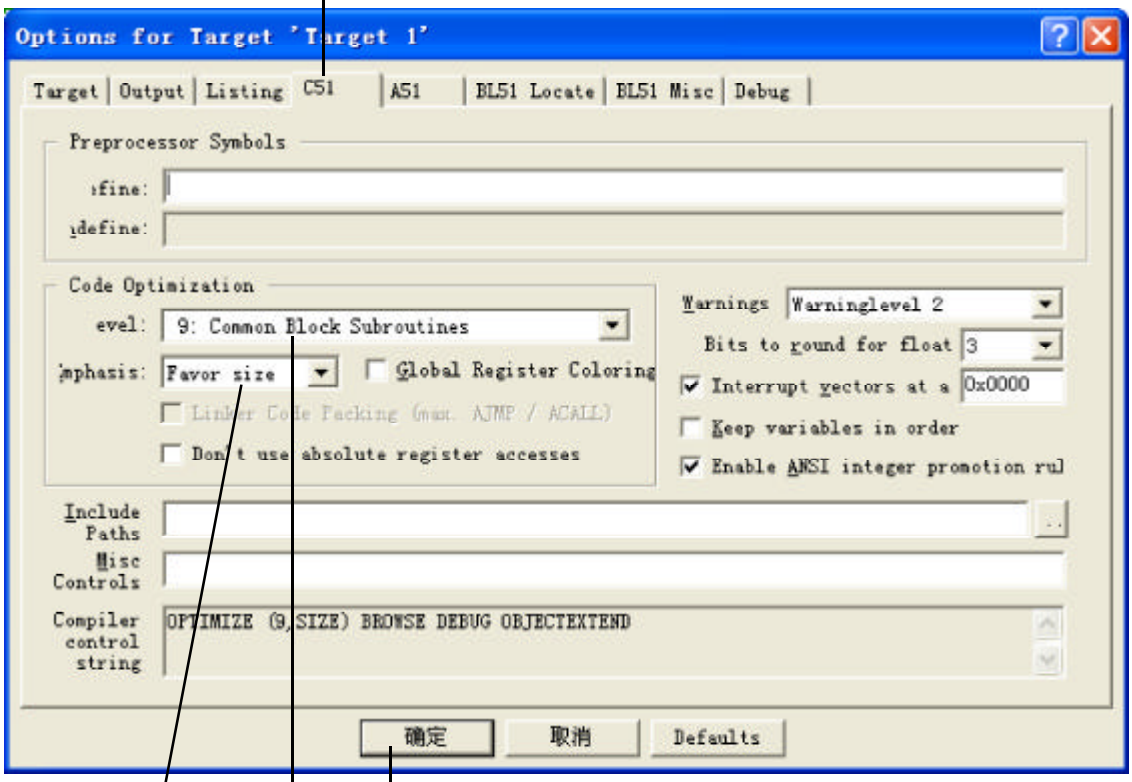
7 . 调试程序时，还可以在选项中选择‘ 每次下载前自动重新调入文件 ”，这样你每次修改原文件并从新编译/汇编后生成的*.hex,*.bin文件就不要再次手工调入了，生产时不要用。



附录 E: Keil C51 高级语言编程的软件如何减少代码长度

1.在“Project”菜单中选择“Options for Target”

2.在“Options for Target”中选择“C51”



3.选择按空间大小， 9级优化程序

4.重新编译程序即可。

附录F:(写给用STC89C51RC/RD+系列单片机做仿真器的朋友)

STC89C51 RC / RD+ 系列 单片机 做仿真器须知

1.对Flash的IAP字节读/字节编程/扇区擦除等待机器周期

设置等待时间			CPU 等待时间(机器周期)			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
0	1	1	6	30	5471	5MHz
0	1	0	11	60	10942	10MHz
0	0	1	22	120	21885	20MHz
0	0	0	43	240	43769	40MHz

512Byte/Sector

2.程序在ISP区可对用户AP区进行IAP读/编程/扇区擦除

3.程序在用户AP区不可对ISP区进行IAP操作。

4.程序在ISP/AP区可对Data Flash区进行IAP操作。

5.可从ISP区软复位到AP区，也可从AP区软复位到ISP区。

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	MS2	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	-	-	WT2	WT1	WT0	000x,x000

```
;从用户应用程序区(AP区)软件复位并切换到ISP程序区开始执行程序
    MOV   ISP_CONTR,    #01100000B     ;SWBS = 1(选择ISP区), SWRST = 1(软复位)

;从ISP程序区软件复位并切换到用户应用程序区(AP区)开始执行程序
    MOV   ISP_CONTR,    #00100000B     ;SWBS = 0(选择AP区), SWRST = 1(软复位)
```

软复位后所有的寄存器复位，可在软复位前将需保存的SFR的值，保存在RAM或DATA Flash中，建议采用STC89C58RD+设计。

STC89C58RD+50-C-PI: 独立32k字节用户程序区，独立可选0/1/2/4k字节系统ISP区。
独立27k字节数据Flash区

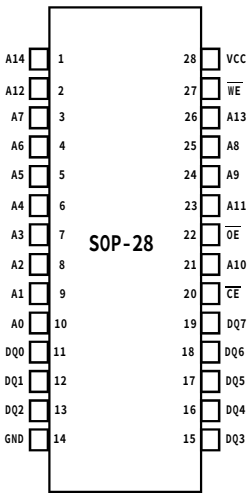
只有选择1/2/4k字节ISP区,并选择上电复位从ISP区启动，晶片IAP功能才起作用

工作电压：5.5v - 3.4v，不要过低，否则工作不了。5V串2个1N4001二极管是3.6V

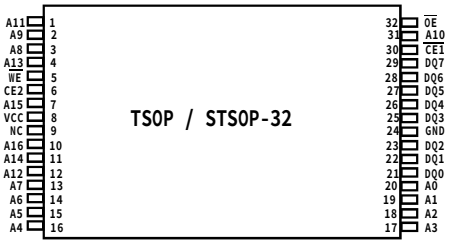
附录 G： STC 高性能 SRAM 选型一览表

型号	容量	工作电压	温度	速度	推荐封状	供货
STC62WV256	32K x 8	2.4 - 5.5V	-40℃～85℃	70nS	SOP/TSOP/DIP	现货
STC62WV1024	128K x 8	2.4 - 5.5V	-40℃～85℃	70nS	SOP/STSOP/TSOP	现货
STC62WV2568	256K x 8	2.4 - 5.5V	-40℃～85℃	70nS	STSOP-32	现货
STC62WV5128	512K x 8	2.4 - 5.5V	-40℃～85℃	70nS	STSOP/SOP-32	现货
STC62WV1M8	1M x 8	2.4 - 5.5V	-40℃～85℃	70nS	TSOP2-44	订货
STC62WV6416	64K x 8	2.4 - 5.5V	-40℃～85℃	70nS	TSOP2-44	现货
STC62WV12816	128K x 16	2.4 - 5.5V	-40℃～85℃	70nS	TSOP2-44	现货
STC62LV12816	128K x 16	2.4 - 3.6V	-40℃～85℃	70nS	TSOP2-44	现货
STC62WV25616	256K x 16	2.4 - 5.5V	-40℃～85℃	70nS	TSOP2-44	现货
STC62WV51216	512K x 16	2.4 - 5.5V	-40℃～85℃	70nS	TSOP2-44	现货

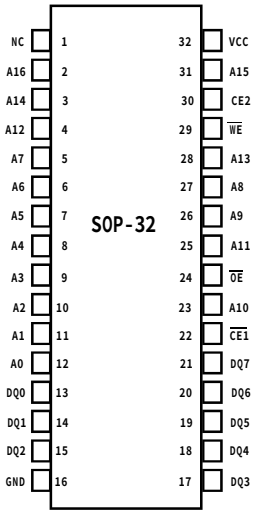
STC62WV256



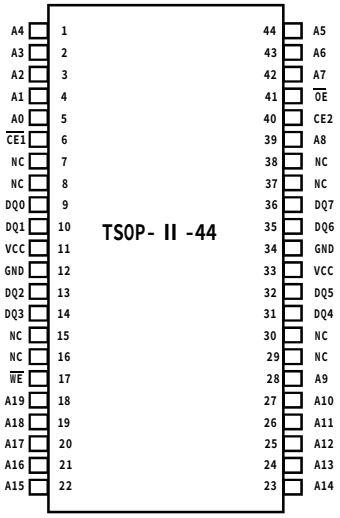
STC62WV1024



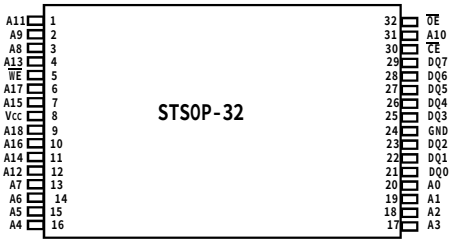
STC62WV1024



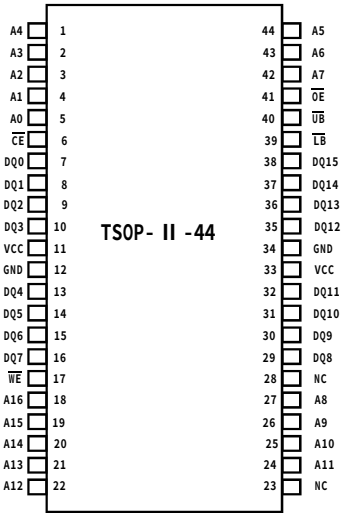
STC62WV1M8



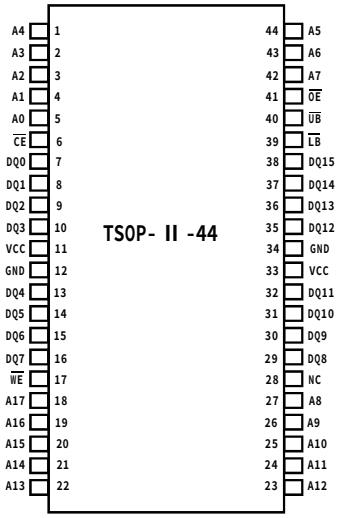
STC62WV5128



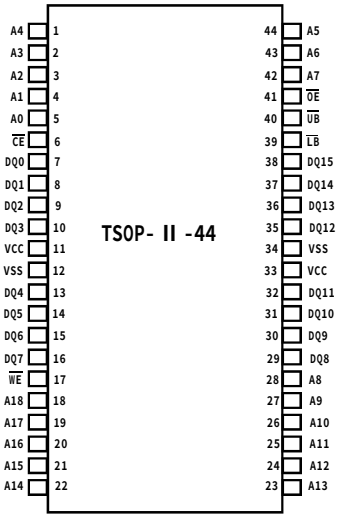
STC62WV12816



STC62WV25616



STC62WV51216



附录 H: STC 单片机配套工具价格

1.STC 单片机 ISP 下载编程工具，人民币 50 元(可申请样品)

2.STC 单片机 ISP 到 ISP 脱机下载工具，人民币 200 元(脱机就是量产时无须电脑)

3.STC 单片机专用烧录器，人民币 200 元(有些用户生产时希望擦除 ISP,或要烧录自己开发的 ISP 程序，用户自己开发的 ISP 程序可与用户应用程序一起烧录)

4.STC- 单片机仿真器，人民币 1950 元,测试中

STC89C51RC/RD+ 系列 5V 单片机批量价格，1K 以下零售加 0.3 元，快递费 25 元

型 号	尾 缀	封 装	商 规 / 工 规	单 价	RAM 字 节	Flash 程 序	P4 口
STC89C51RC	40C-PDIP	DIP-40	商 规	4.7元	512	4K	
STC89C51RC	40C-PLCC	PLCC-44	商 规	5.2元	512	4K	有
STC89C51RC	40C-PQFP	PQFP-44	商 规	5.6元	512	4K	有
STC89C51RC	40I-PDIP	DIP-40	工 规	5.5元	512	4K	
STC89C51RC	40I-PLCC	PLCC-44	工 规	5.7元	512	4K	有
STC89C51RC	40I-PQFP	PQFP-44	工 规	6.0元	512	4K	有
STC89C52RC	40C-PDIP	DIP-40	商 规	5.4元	512	8K	
STC89C52RC	40C-PLCC	PLCC-44	商 规	6.0元	512	8K	有
STC89C52RC	40C-PQFP	PQFP-44	商 规	6.0元	512	8K	有
STC89C52RC	40I-PDIP	DIP-40	工 规	6.7元	512	8K	
STC89C52RC	40I-PLCC	PLCC-44	工 规	6.7元	512	8K	有
STC89C52RC	40I-PQFP	PQFP-44	工 规	7.0元	512	8K	有
STC89C53RC	40C-PQFP	DIP-40	商 规	8.0元	512	15K	
STC89C53RC	40C-PLCC	PLCC-44	商 规	8.0元	512	15K	有
STC89C53RC	40C-PQFP	PQFP-44	商 规	8.0元	512	15K	有
STC89C53RC	40I-PQFP	DIP-40	工 规	9.0元	512	15K	
STC89C53RC	40I-PLCC	PLCC-44	工 规	9.0元	512	15K	有
STC89C53RC	40I-PQFP	PQFP-44	工 规	9.0元	512	15K	有
STC89C54RD+	40C-PQFP	DIP-40	商 规	9.0元	1280	16K	
STC89C54RD+	40C-PLCC	PLCC-44	商 规	9.0元	1280	16K	有
STC89C54RD+	40C-PQFP	PQFP-44	商 规	9.0元	1280	16K	有
STC89C54RD+	40I-PQFP	DIP-40	工 规	11元	1280	16K	
STC89C54RD+	40I-PLCC	PLCC-44	工 规	11元	1280	16K	有
STC89C54RD+	40I-PQFP	PQFP-44	工 规	11元	1280	16K	有
STC89C58RD+	40C-PQFP	DIP-40	商 规	12元	1280	32K	
STC89C58RD+	40C-PLCC	PLCC-44	商 规	12元	1280	32K	有
STC89C58RD+	40C-PQFP	PQFP-44	商 规	12元	1280	32K	有
STC89C58RD+	40I-PDIP	DIP-40	工 规	15元	1280	32K	
STC89C58RD+	40I-PLCC	PLCC-44	工 规	15元	1280	32K	有
STC89C58RD+	40I-PQFP	PQFP-44	工 规	15元	1280	32K	有
STC89C516RD+	40C-PDIP	DIP-40	商 规	13.5元	1280	63K	
STC89C516RD+	40C-PLCC	PLCC-44	商 规	13.5元	1280	63K	有
STC89C516RD+	40C-PQFP	PQFP-44	商 规	13.5元	1280	63K	有
STC89C516RD+	40I-PDIP	DIP-40	工 规	17元	1280	63K	
STC89C516RD+	40I-PLCC	PLCC-44	工 规	17元	1280	63K	有
STC89C516RD+	40I-PQFP	PQFP-44	工 规	17元	1280	63K	有
STC89LE516AD				14元			
STC89LE58AD				12.5元			
STC89LE54AD				11元			
STC89LE52AD				9元			
STC89LE51AD				8元			

附录 I：

STC12C2052AD 系列 1T 单片机简介

STC12C2052 系列单片机是单时钟 / 机器周期的兼容 8051 内核单片机，是高速 / 低功耗的新一代 8051 单片机，全新的流水线 / 精简指令集结构。

特 点 ：

1.增强型 1T 流水线 / 精简指令集结构 8051 CPU

2.工作电压：2.4V - 3.8V / 3.8V - 5.5V

3.工作频率范围：0 - 20 MHz

4.用户应用程序空间512 / 1K / 2K / 3K / 4K / 5K / 6K 字节

5.片上集成 256 字节 RAM

6.EEPROM 功能

7.共 2 个 16 位定时器 / 计数器

8.PWM(2 路）/PCA（可编程计数器阵列）

9.增强型通用异步串行口(UART)

10. SPI 同步通信口， 主模式 / 从模式

11. ADC，8 路 8 位

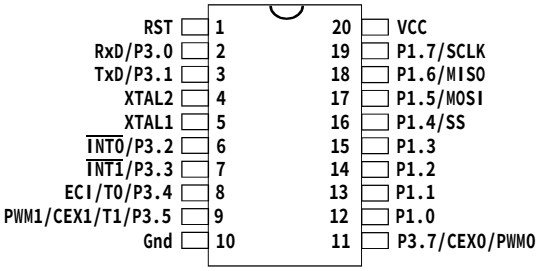
12.看门狗

13.内部集成 R/C 振荡器，精度要求不高时可省外部晶体

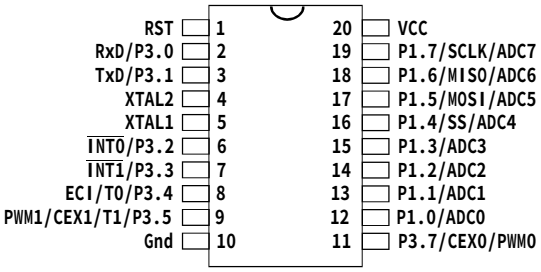
14.ISP/IAP

15.工作温度范围： 0 - 75℃ / -40 - +85℃

STC12C2052AD 系列单片机管脚图及封装尺寸

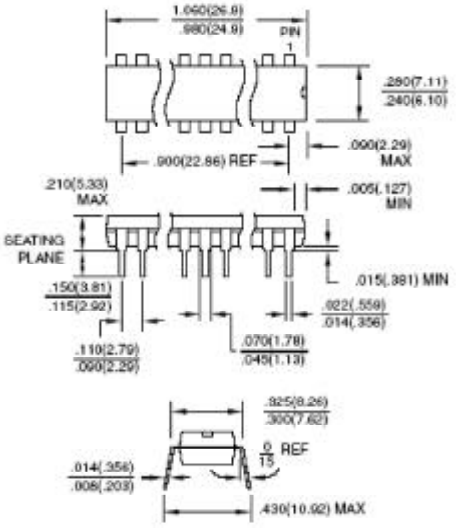


STC12C2052

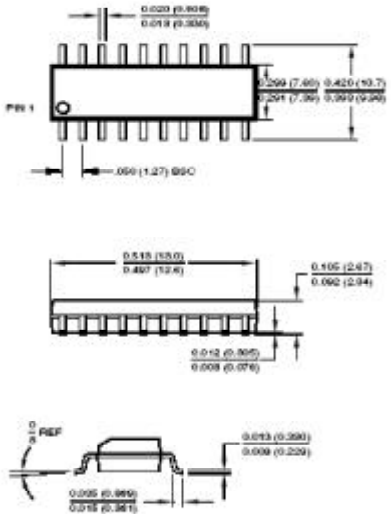


STC12C2052AD

20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP)
Dimensions in Inches and (Millimeters)
JEDEC STANDARD MS-001 AD

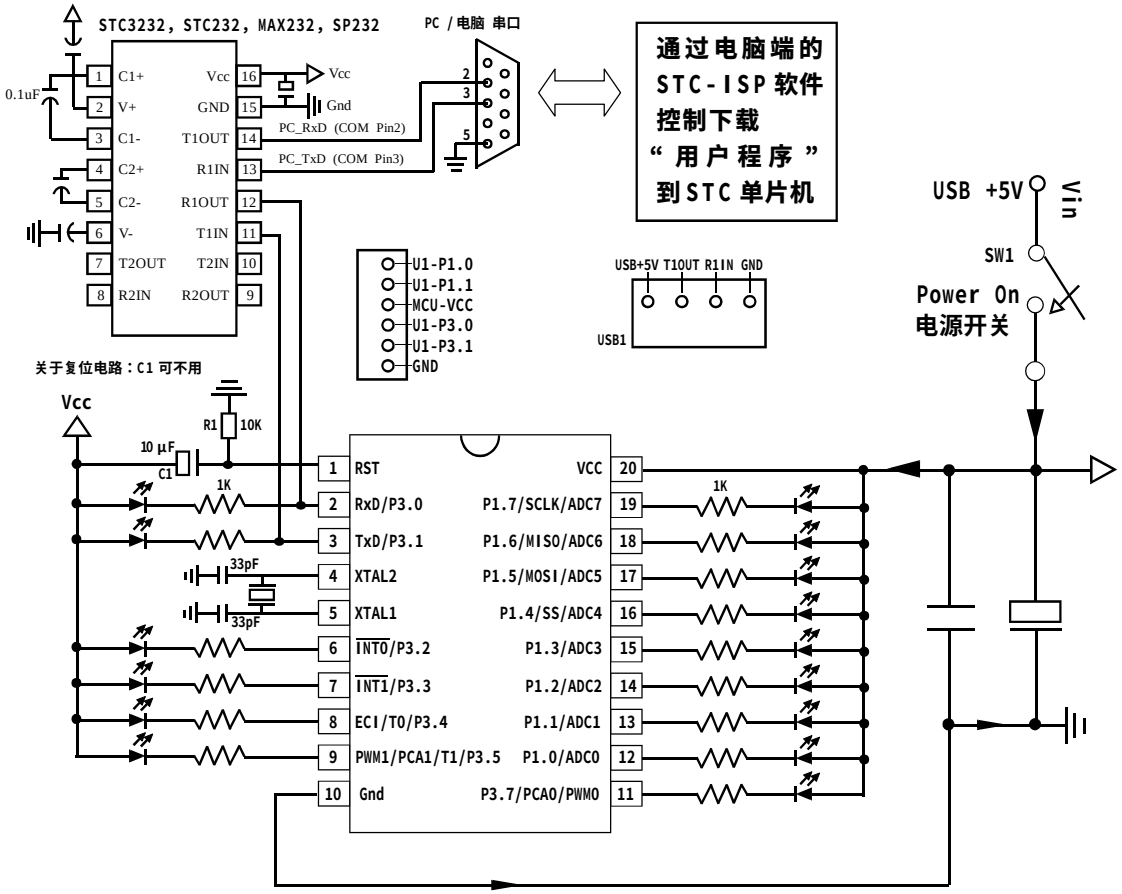


20S, 20-lead, 0.300" Wide, Plastic Gull Wing Small Outline (SOIC)
Dimensions in Inches and (Millimeters)



STC 单片机 典型应用电路(STC12C2052AD 系列)

---- 通过 RS-232 转换器连接电脑就可以下载程序



STC12C2052AD 系列单片机选型一览表（全部有 ISP 功能）：

	工作电压(V)	Flash 程序存储器	SRAM	定时器	UART	SPI	PCA PWM	A/D	I/O	看门狗	内置复位	EEPROM	封装
STC12C0552	3.8 - 5.5	512	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12C0552AD	3.8 - 5.5	512	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12C1052	3.8 - 5.5	1K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12C1052AD	3.8 - 5.5	1K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12C2052	3.8 - 5.5	2K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12C2052AD	3.8 - 5.5	2K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12C3052	3.8 - 5.5	3K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12C3052AD	3.8 - 5.5	3K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12C4052	3.8 - 5.5	4K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12C4052AD	3.8 - 5.5	4K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12C5052	3.8 - 5.5	5K	256	2	有	有	2路		15	有	有		DIP/SOP
STC12C5052AD	3.8 - 5.5	5K	256	2	有	有	2路	有	15	有	有		DIP/SOP
STC12C6052	3.8 - 5.5	6K	256	2	有	有	2路		15	有	有		DIP/SOP
STC12C6052AD	3.8 - 5.5	6K	256	2	有	有	2路	有	15	有	有		DIP/SOP
STC12LE0552	2.4 - 3.8	512	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12LE0552AD	2.4 - 3.8	512	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12LE1052	2.4 - 3.8	1K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12LE1052AD	2.4 - 3.8	1K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12LE2052	2.4 - 3.8	2K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12LE2052AD	2.4 - 3.8	2K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12LE3052	2.4 - 3.8	3K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12LE3052AD	2.4 - 3.8	3K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12LE4052	2.4 - 3.8	4K	256	2	有	有	2路		15	有	有	256	DIP/SOP
STC12LE4052AD	2.4 - 3.8	4K	256	2	有	有	2路	有	15	有	有	256	DIP/SOP
STC12LE5052	2.4 - 3.8	5K	256	2	有	有	2路		15	有	有		DIP/SOP
STC12LE5052AD	2.4 - 3.8	5K	256	2	有	有	2路	有	15	有	有		DIP/SOP
STC12LE6052	2.4 - 3.8	6K	256	2	有	有	2路		15	有	有		DIP/SOP
STC12LE6052AD	2.4 - 3.8	6K	256	2	有	有	2路	有	15	有	有		DIP/SOP

指令系统分类总结

如果按功能分类，STC89/12 系列单片机指令系统可分为：

1 . 数据传送类指令；

2 . 算术操作类指令；

3 . 逻辑操作类指令；

4 . 控制转移类指令；

5 . 布尔变量操作类指令。

按功能分类的指令系统表如下表所示。

STC89/12 系列单片机指令与机器码速查表见...

传统 12T 8051 ←

STC12C2052AD →

数据传送类指令

助记符		功能说明	字节数	12时钟/机器周期所需时钟	1时钟/机器周期所需时钟
MOV	A, Rn	寄存器内容送入累加器	1	12	1
MOV	A, direct	直接地址单元中的数据送入累加器	2	12	2
MOV	A, @Ri	间接RAM中的数据送入累加器	1	12	2
MOV	A, #data	立即送入累加器	2	12	2
MOV	Rn, A	累加器内容送入寄存器	1	12	2
MOV	Rn, direct	直接地址单元中的数据送入寄存器	2	24	4
MOV	Rn, #data	立即数送入寄存器	2	12	2
MOV	direct, A	累加器内容送入直接地址单元	2	12	3
MOV	direct, Rn	寄存器内容送入直接地址单元	2	24	3
MOV	direct, direct	直接地址单元中的数据送入另一个直接地址单元	3	24	4
MOV	direct, @Ri	间接RAM中的数据送入直接地址单元	2	24	4
MOV	direct, #data	立即数送入直接地址单元	3	24	3
MOV	@Ri, A	累加器内容送间接RAM单元	1	12	3
MOV	@Ri, direct	直接地址单元数据送入间接RAM单元	2	24	3
MOV	@Ri, #data	立即数送入间接RAM单元	2	12	3
MOV	DPTR, #data16	16位立即数送入地址寄存器	3	24	3
MOVC	A,@A+DPTR	以DPTR为基地址变址寻址单元中的数据送入累加器	1	24	4
MOVC	A,@A+PC	以PC为基地址变址寻址单元中的数据送入累加器	1	24	4
MOVX	A,@Ri	外部RAM（8位地址）送入累加器	1	24	3
MOVX	A,@DPTR	外部RAM（16位地址）送入累加器	1	24	2
MOVX	@Ri,A	累加器送外部RAM（8位地址）	1	24	3
MOVX	@DPTR ,A	累加器送外部RAM（16位地址）	1	24	2
PUSH	direct	直接地址单元中的数据压入堆栈	2	24	4
POP	direct	出栈送直接地址单元	2	24	3
XCH	A,Rn	寄存器与累加器交换	1	12	3
XCH	A,direct	直接地址单元与累加器交换	2	12	4
XCH	A,@Ri	间接RAM与累加器交换	1	12	4
XCHD	A,@Ri	间接RAM的低半字节与累加器交换	1	12	4

算术操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟
ADD	A，Rn	寄存器内容加到累加器	1	12	2
ADD	A，direct	直接地址单元中的数据加到累加器	2	12	3
ADD	A，@Ri	间接RAM中的数据加到累加器	1	12	3
ADD	A，#data	立即加到累加器	2	12	2
ADDC	A，Rn	寄存器内容带进位加到累加器	1	12	2
ADDC	A，direct	直接地址单元的内容带进位加到累加器	2	12	3
ADDC	A，@Ri	间接RAM内容带进位加到累加器	1	12	3
ADDC	A，#data	立即数带进位加到累加器	2	12	2
SUBB	A，Rn	累加器带借位减寄存器内容	1	12	2
SUBB	A，direct	累加器带借位减直接地址单元的内容	2	12	3
SUBB	A，@Ri	累加器带借位减间接RAM中的内容	1	12	3
SUBB	A，#data	累加器带借位减立即数	2	12	2
INC	A	累加器加1	1	12	2
INC	Rn	寄存器加1	1	12	3
INC	direct	直接地址单元加1	2	12	4
INC	@Ri	间接RAM单元加1	1	12	4
DEC	A	累加器减1	1	12	2
DEC	Rn	寄存器减1	1	12	3
DEC	direct	直接地址单元减1	2	12	4
DEC	@Ri	间接RAM单元减1	1	12	4
INC	DPTR	地址寄存器DPTR加1	1	24	1
MUL	AB	A乘以B	1	48	4
DIV	AB	A除以B	1	48	5
DA	A	累加器十进制调整	1	12	4

逻辑操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟
ANL	A，Rn	累加器与寄存器相“与”	1	12	2
ANL	A，direct	累加器与直接地址单元相“与”	2	12	3
ANL	A，@Ri	累加器与间接RAM单元相“与”	1	12	3
ANL	A，#data	累加器与立即数相“与”	2	12	2
ANL	direct，A	直接地址单元与累加器相“与”	2	12	4
ANL	direct，#data	直接地址单元与立即数相“与”	3	24	4
ORL	A，Rn	累加器与寄存器相“或”	1	12	2
ORL	A，direct	累加器与直接地址单元相“或”	2	12	3
ORL	A，@Ri	累加器与间接RAM单元相“或”	1	12	3
ORL	A，#data	累加器与立即数相“或”	2	12	2
ORL	direct，A	直接地址单元与累加器相“或”	2	12	4
ORL	direct，#data	直接地址单元与立即数相“或”	3	24	4
XRL	A，Rn	累加器与寄存器相“异或”	1	12	2
XRL	A，direct	累加器与直接地址单元相“异或”	2	12	3
XRL	A，@Ri	累加器与间接RAM单元相“异或”	1	12	3
XRL	A，#data	累加器与立即数相“异或”	2	12	2
XRL	direct，A	直接地址单元与累加器相“异或”	2	12	4
XRL	direct，#data	直接地址单元与立即数相“异或”	3	24	4
CLR	A	累加器清“0”	1	12	1
CPL	A	累加器求反	1	12	2
RL	A	累加器循环左移	1	12	1
RLC	A	累加器带进位位循环左移	1	12	1
RR	A	累加器循环右移	1	12	1
RRC	A	累加器带进位位循环右移	1	12	1
SWAP	A	累加器半字节交换	1	12	1

控制转移类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟
ACALL	addr11	绝对（短）调用子程序	2	24	6
ACALL	addr16	长调用子程序	3	24	6
RET		子程序返回	1	24	4
RETI		中断返回	1	24	4
AJMP	addr11	绝对（短）转移	2	24	3
AJMP	addr16	长转移	3	24	4
SJMP	re1	相对转移	2	24	3
JMP	@A+DPTR	相对于DPTR的间接转移	1	24	3
JZ	re1	累加器为零转移	2	24	3
JNZ	re1	累加器非零转移	2	24	3
CJNE	A, direct, re1	累加器与直接地址单元比较，不相等则转移	3	24	5
CJNE	A, #data, re1	累加器与立即数比较，不相等则转移	3	24	4
CJNE	Rn, #data, re1	寄存器与立即数比较，不相等则转移	3	24	4
CJNE	@Ri, #data, re1	间接RAM单元与立即数比较，不相等则转移	3	24	5
DJNZ	@Rn, re1	寄存器减1，非零转移	3	24	4
DJNZ	direct, re1	直接地址单元减1，非零转移	3	24	5
NOP		空操作	1	12	1

布尔变量操作类指令

助记符		功能说明	字节数	12时钟/周期所需时钟	1时钟/周期所需时钟
CLR	C	清进位位	1	12	1
CLR	bit	清直接地址位	2	12	4
SETB	C	置进位位	1	12	1
SETB	bit	置直接地址位	2	12	4
CPL	C	进位位求反	1	12	1
CPL	bit	直接地址位求反	2	12	4
ANL	C, bit	进位位和直接地址位相“与”	2	24	3
ANL	C, /bit	进位位和直接地址位的反码相“与”	2	24	3
ORL	C, bit	进位位和直接地址位相“或”	2	24	3
ORL	C, /bit	进位位和直接地址位的反码相“或”	2	24	3
MOV	C, bit	直接地址位送入进位位	2	12	3
MOV	bit, C	进位位送入直接地址位	2	24	3
JC	re1	进位位为1则转移	2	24	3
JNC	re1	进位位为0则转移	2	24	3
JB	bit, re1	直接地址位为1则转移	3	24	4
JNB	bit, re1	直接地址位为0则转移	3	24	4
JBC	bit, re1	直接地址位为1则转移，该位清0	3	24	5

特殊功能寄存器映像 SFR Mapping

	Bit Addressable	Non Bit Addressable							
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	
F8h		CH 0000,0000	CCAP0H 0000,0000	CCAP1H 0000,0000					FFh
F0h	B 0000,0000		PCA_PWM0 xxxx,xx00	PCA_PWM1 xxxx,xx00					F7h
E8h		CL 0000,0000	CCAP0L 0000,0000	CCAP1L 0000,0000					EFh
E0h	ACC 0000,0000	WDT_CONTR 0x00,0000	ISP_DATA 1111,1111	ISP_ADDRH 0000,0000	ISP_ADDRL 0000,0000	ISP_CMD xxxx,x000	ISP_TRIG xxxx,xxxx	ISP_CONTR 0000,x000	E7h
D8h	CCON 00xx,xx00	CMOD 0xxx,x000	CCAPM0 x000,0000	CCAPM1 x000,0000					DFh
D0h	PSW 0000,0000								D7h
C8h									CFh
C0h						ADC_CONTR 0000,0000	ADC_DATA 0000,0000	CLOCK_DIV xxxx,x000	C7h
B8h	IP x000,0000	SADEN 0000,0000							BFh
B0h	P3 1x11,1111	P3M0 0000,0000	P3M1 0000,0000					IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR 0000,0000							AFh
A0h								TEST_WDT don't use	A7h
98h	SCON 0000,0000	SBUF xxxx,xxxx							9Fh
90h	P1 1111,1111	P1M0 0000,0000	P1M1 0000,0000						97h
88h	TCON 0000,0000	TMOD 0000,0000	TL0 0000,0000	TL1 0000,0000	TH0 0000,0000	TH1 0000,0000	AUXR 0000,00xx		8Fh
80h	P0 1111,1111	SP 0000,0111	DPL 0000,0000	DPH 0000,0000	SPI_STATUS 00xx,xxxx	SPI_CONTR 0000,0100	SPI_DATA 0000,0000	PCON 0011,0000	87h
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F	

STC12C2052AD 系列 8051 单片机内核特殊功能寄存器 C51 Core SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ACC	E0h	Accumulator									0000,0000
B	F0h	B Register									0000,0000
PSW	D0h	Program Status Word	CY	AC	F0	RS1	RS0	OV	F1	P	0000,0000
SP	81h	Stack Pointer									0000,0111
DPL	82h	Data Pointer Low Byte									0000,0000
DPH	83h	Data Pointer High Byte									0000,0000

STC12C2052AD 系列 8051 单片机系统管理特殊功能寄存器 System Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	BODF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ENBODI	-	-	0000,00xx
CLOCK_DIV	C7h	Clock Divder	-	-	-	-	-	CLOCK_S2	CLOCK_S1	CLOCK_S0	xxxx,x000

STC12C2052AD 系列 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA	ESPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA	PSPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCAH	PSPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

STC12C2052AD 系列 8051 单片机 I/O 口 特殊功能寄存器 Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
P1	90h	8-bit Port 1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	1111,1111
P1M0	91h										0000,0000
P1M1	92h										0000,0000
P3	B0h	8-bit Port 3	P3.7	-	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0	1x11,1111
P3M0	B1h										0000,0000
P3M1	B2h										0000,0000

STC12C2052AD系列 8051 单片机 串行口 特殊功能寄存器 Serial I/O Port SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
SCON	98h	Serial Control	SM0/FE	SM1	SM2	REN	TB8	RB8	TI	RI	0000,0000
SBUF	99h	Serial Data Buffer									xxxx,xxxx
SADEN	B9h	Slave Address Mask									0000,0000
SADDR	A9h	Slave Address									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ENLVFI	-	-	0000,00xx

STC12C2052AD系列 8051 单片机 定时器 特殊功能寄存器 Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
TCON	88h	Timer / Counter 0 and 1 Control	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	0000,0000
TMOD	89h	Timer / Counter 0 and 1 Modes	GATE GATE1	C/T# C/T1#	M1 M1_1	M0 M1_0	GATE GATE0	C/T# C/T0#	M1 M0_1	M0 M0_0	0000,0000
TL0	8Ah	Timer / Counter 0 Low Byte									0000,0000
TH0	8Ch	Timer / Counter 0 High Byte									0000,0000
TL1	8Bh	Timer / Counter 1 Low Byte									0000,0000
TH1	8Dh	Timer / Counter 1 High Byte									0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ENLVFI	-	-	0000,00xx

STC12C2052AD系列 8051单片机 看门狗定时器 特殊功能寄存器 Watch Dog Timer SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

STC12C2052AD系列 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	000x,x000

中断

STC12C2052AD 系列 1T 8051 单片机 中断 特殊功能寄存器 Interrupt SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
IE	A8h	Interrupt Enable	EA	EPCA	ESPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA	PSPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCAH	PSPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000

中断与普通 8051 完全兼容，优先级可设为 4 级。

Interrupt Source 中断源	Vector Address 中断向量地址	Polling Sequence 中断查询次序	中断优先级设置	优先级0 最低	优先级1	优先级2	优先级3 最高	Interrupt Request 中断请求
/INT0	0003H	0(最优先)	PX0H,PX0	0,0	0,1	1,0	1,1	IE0
Timer 0	000BH	1	PT0H,PT0	0,0	0,1	1,0	1,1	TF0
/INT1	0013H	2	PX1H,PX1	0,0	0,1	1,0	1,1	IE1
Timer 1	001BH	3	PT1H,PT1	0,0	0,1	1,0	1,1	IF1
UART	0023H	4	PSH, PS	0,0	0,1	1,0	1,1	RI + TI
SPI/ADC	002BH	5	PSPIH,PSPI	0,0	0,1	1,0	1,1	
PCA/BOD	0033H	6	PPCAH,PPCA	0,0	0,1	1,0	1,1	

定时器 0/ 定时器 1，UART 串口的速度

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ENBODI	-	-	0000,00xx

定时器 0 和定时器 1：

STC12C2052AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，定时器 0 和定时器 1 复位后是传统 8051 的速度，即 12 分频，这是为了兼容传统 8051。但也可不进行 12 分频，实现真正的 1T。

T0x12： 0，定时器 0 是传统 8051 速度，12 分频；1，定时器 0 的速度是传统 8051 的 12 倍，不分频

T1x12： 0，定时器 1 是传统 8051 速度，12 分频；1，定时器 1 的速度是传统 8051 的 12 倍，不分频

UART 串口的模式 0：

STC12C2052AD 系列是 1T 的 8051 单片机，为了兼容传统 8051，UART 串口复位后是兼容传统 8051 的

UART_M0x6： 0，UART 串口的模式 0 是传统 12T 的 8051 速度，12 分频；
1，UART 串口的模式 0 的速度是传统 12T 的 8051 的 6 倍，2 分频

I/O口结构

I/O口配置

STC12C2052AD系列单片机其所有I/O口均可由软件配置成4种类型之一，如下表所示。4种类型分别为：准双向口（标准8051输出模式）、推挽输出、仅为输入（高阻）或开漏输出功能。每个口配置2个控制寄存器控制每个引脚输出类型。STC12C2052AD系列单片机上电复位后为准双向口（标准8051输出模式）模式。

口输出方式设定

P3M0【7：0】	P3M1【7：0】	I/O 口模式
0	0	准双向口(传统8051 I/O 口模式)
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻）
1	1	开漏(Open Drain)

P1M0【7：0】	P1M1【7：0】	I/O 口模式（P1.x 如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051 I/O 口模式）
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open Drain)，如果该I/O口需作为A/D使用，可选此模式

1.准双向口输出配置

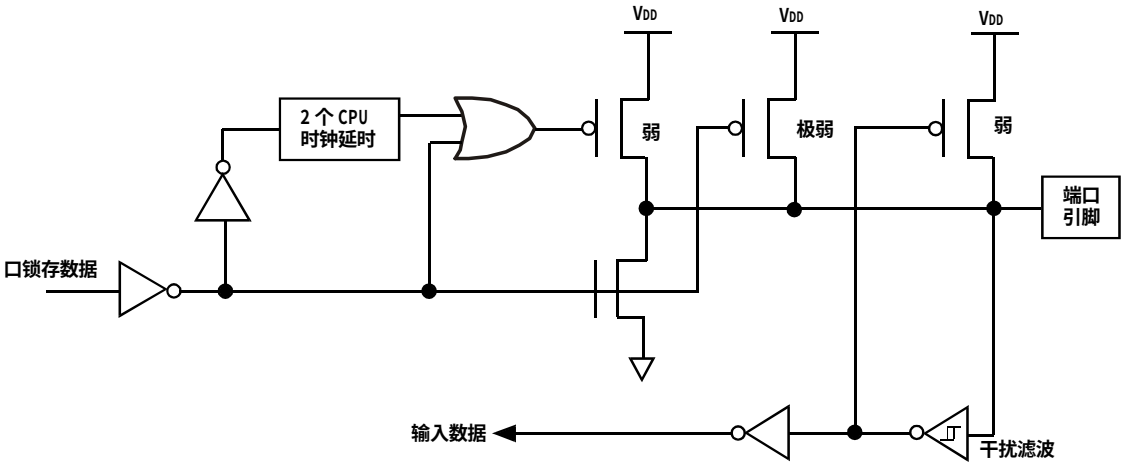
准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为 1 时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有 3 个上拉晶体管适应不同的需要。

在 3 个上拉晶体管中，有 1 个“极弱上拉”，当口线锁存为 1 时打开。当引脚悬空时，这个极弱的上拉源产生很弱的上拉电流将引脚上拉为高电平。

第 2 个上拉晶体管称为“弱上拉”，当口线寄存器为 1 且引脚本身也为 1 时打开。此上拉提供基本驱动电流使准双向口输出为 1。如果一个引脚输出为 1 而由外部装置下拉到低时，弱上拉关闭而“极弱上拉”维持开状态，为了把这个引脚强拉为低，外部装置必须有足够的灌电流能力使引脚上的电压将到门槛电压以下。

第 3 个上拉晶体管称为“强上拉”。当口线锁存器由 0 到 1 跳变时，这个上拉用来加快准双向口由逻辑 0 到逻辑 1 转换。当发生这种情况时，强上拉打开约 2 个机器周期以使引脚能够迅速地上拉到高电平 。

准双向口输出如下图所示。

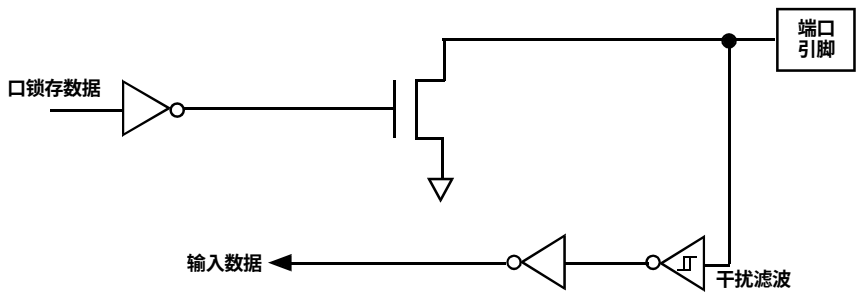


尽管 STC12LE2052 系列单片机为 3V 器件，但引脚可承受 5V 电压。在准双向口模式中，如果用户在引脚加上 5V 电压，将会有电流从引脚流向 VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向 3V 单片机引脚施加 5V 电压。

准双向口带有一个施密特触发输入以及一个干扰抑制电路。

2.开漏输出配置

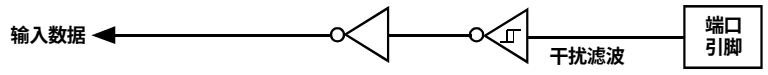
当口线锁存器为 0 时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出时，这种配置方式必须有外部上拉，一般通过电阻外接到 VDD。这种方式的下拉与准双向口相同。输出口线配置如下图所示。



开漏端口带有一个施密特触发输入以及一个干扰抑制电路。

3．仅为输入（高阻）配置

输入口配置如下图所示。

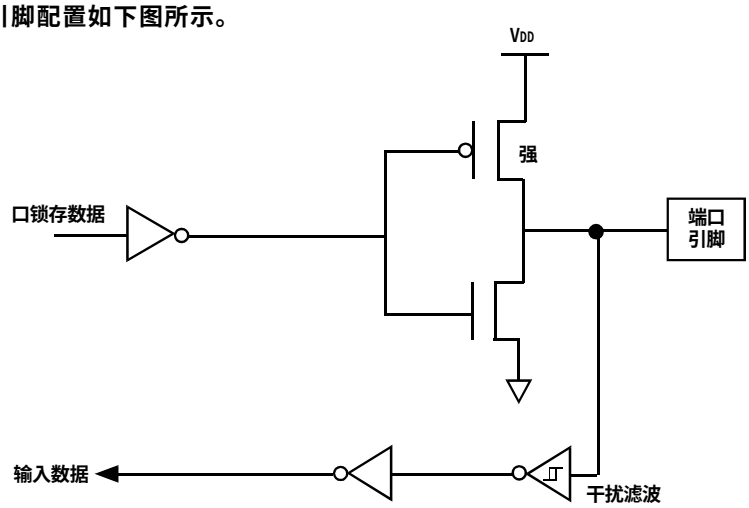


输入口带有一个施密特触发输入以及一个干扰抑制电路。

4.推挽输出配置

推挽输出配置的下拉结构与开漏输出以及准双向口的下拉结构相同，但当锁存器为 1 时提供持续的强上拉。推挽模式一般用于需要更大驱动电流的情况。

推挽引脚配置如下图所示。



A/D 及 A/D转换寄存器 ADC_CONTR/ADC_DATA

STC12C2052AD 系列带 A/D 转换的单片机在 P1 口，有 8 路 8 位高精度的高速 A/D 转换器，速度可达 100KHz。P1.7 - P1.0 共 8 路电压输入型 A/D，可做温度检测、电池电压检测、按键扫描、频谱检测等。上电复位后 P1 口为弱上拉型 I/O 口，用户可以通过软件设置将 8 路中的任何一路设置为 A/D 转换，不需作为 A/D 使用的口可继续作为 I/O 口使用。需作为 A/D 使用的口需先将其设置为高阻输入或开漏模式。在 P1M0、P1M1 中对相应的位进行设 置 。

P1M0【7：0】 地址：91h	P1M1【7：0】 地址：92h	I/O 口模式（P1.x 如做A/D使用，需先将其设置成开漏或高阻输入）
0	0	准双向口（传统8051 I/O 口模式）
0	1	推挽输出（强上拉输出，可达20mA，尽量少用）
1	0	仅为输入（高阻），如果该I/O口需作为A/D使用，可选此模式
1	1	开漏(Open Drain)，如果该I/O口需作为A/D使用，可选此模式

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000
ADC_DATA	C6h	A/D 转换结果寄存器	-	-	-	-	-	-	-	-	xxxx,xxxx

ADC_CONTR 特殊功能寄存器： A/D转换控制特殊功能寄存器

A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0xx0,0000

CHS2 / CHS1 / CHS0：模拟输入通道选择，CHS2 / CHS1 / CHS0

CHS2	CHS1	CHS0	Analog Input Channel Select 模拟输入通道选择
0	0	0	选择 P1.0 作为 A/D 输入来用
0	0	1	选择 P1.1 作为 A/D 输入来用
0	1	0	选择 P1.2 作为 A/D 输入来用
0	1	1	选择 P1.3 作为 A/D 输入来用
1	0	0	选择 P1.4 作为 A/D 输入来用
1	0	1	选择 P1.5 作为 A/D 输入来用
1	1	0	选择 P1.6 作为 A/D 输入来用
1	1	1	选择 P1.7 作为 A/D 输入来用

ADC_START： 模数转换器(ADC)转换启动控制位，设置为“1”时，开始转换

ADC_FLAG： 模数转换器转换结束标志位,当 A/D 转换完成后，ADC_FLAG = 1。

SPEED1，SPEED0：模数转换器转换速度控制位

SPEED1	SPEED0	A/D转换所需时间
1	1	210个时钟周期转换一次， CPU工作频率20MHz时， A/D转换速度约100KHz
1	0	420个时钟周期转换一次
0	1	630个时钟周期转换一次
0	0	840个时钟周期转换一次

ADC_DATA 特殊功能寄存器： A/D转换结果特殊功能寄存器

A/D 转换结果寄存器	-	-	-	-	-	-	-	-	0000,0000

模拟/数字转换结果计算公式如下：结果 = 256 x Vin / Vcc

Vin 为模拟输入通道输入电压，Vcc 为单片机实际工作电压，用单片机工作电压作为模拟参考电压。

```
/* ------------- 宏晶科技   2005/8/29      A/D 转换 C 语言示例 ------------ */
/* ---Mobile: 13922805190, Tel: 0755 - 82908285,  Fax: 0755 - 82944243  */
/* ----Website: www.mcu-memory.com   Email: support@dsp-memory.com  --- */

//   ADC DEMO 程序演示 STC12C2052AD 系列 MCU 的 A/D 转换功能。 时钟   11.0592MHz
//   转换结果以 16 进制形式输出到串行口，可以用串行口调试程序观察输出结果。

#include <reg52.H>
#include <intrins.H>
//定义与 ADC 有关的特殊功能寄存器
sfr  ADC_CONTR       =    0xC5;              // A/D 转换控制寄存器
sfr  ADC_DATA        =    0xC6;              // A/D 转换结果寄存器
sfr  P1M0            =    0x91;              // P1 口模式寄存器 0
sfr  P1M1            =    0x92;              // P1 口模式寄存器 1

typedef    unsigned char   INT8U;
typedef    unsigned int    INT16U;

void delay(INT8U delay_time)            // 延时函数
{
    INT8U       n;
    INT16U      m;
    for (n=0; n<delay_time; n++)
    {
        for (m=0; m<10000; m++);
    }
}

void initiate_RS232 (void)          // 串口初始化
{
    ES = 0;                         // 禁止串口中断
    SCON = 0x50;                    // 0101,0000  8 位数据位， 无奇偶校验
    T2CON = 0x34;                    // 0011，0100， 由 T2 作为波特率发生器

    RCAP2H = 0xFF;                  // 时钟 11.0592MHz， 9600 波特率
    RCAP2L = 0xDB;

    ES = 1;                         // 允许串口中断
}

void Send_Byte(INT8U one_byte)      // 发送一个字节
{
    TI = 0;                         // 清零串口发送中断标志
    SBUF = one_byte;
    while (TI == 0);
    TI = 0;                         // 清零串口发送中断标志
}
```

```
INT8U get_AD_result(INT8U channel)
{
    INT8U AD_finished          =    0;            // 存储 A/D 转换标志
    ADC_DATA    =     0;
    ADC_CONTR  = channel;                 //选择 A/D 当前通道
    delay(1);                             //使输入电压达到稳定
    ADC_CONTR |= 0x08;                    //0000,1000 令 ADC_START = 1, 启动A/D转换
    AD_finished = 0;
    while ( AD_finished == 0 )            //等待A/D转换结束
    {
        AD_finished = (ADC_CONTR & 0x10);  //0001,0000, ADC_FLAG ==1 测试A/D转换结束否
    }
    ADC_CONTR &= 0xF7;                    //1111,0111 令 ADC_START = 0, 关闭A/D转换,
    return (ADC_DATA);                   // 返回 A/D 转换结果
}
void main()
{
    initiate_RS232();
    P1M0   =    P1M0 |    0x63;    // 0110,0011,要设置为 A/D 转换的P1.x口,先设为开漏
    P1M1   =    P1M1 |    0x63;    //0110,0011, P1 的P1.0,P1.1,P1.5,P1.6先设为开漏
                                    //断开P1.0,P1.1,P1.5,P1.6内部上拉电阻
    while(1)
    {
        Send_Byte(get_AD_result(0)); //P1.0 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(1)); //P1.1 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(5)); //P1.5 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(get_AD_result(6)); //P1.6 为 A/D 当前通道, 测量并发送结果
        delay(0x200);

        Send_Byte(0);                 //连续发送 4 个 00H, 便于观察输出显示
        Send_Byte(0);
        Send_Byte(0);
        Send_Byte(0);

        delay(0x200);                 //延时
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
        delay(0x200);
    }
}
```

看门狗应用

适用型号： STC12C2052AD 系列

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
WDT_CONTR	E1h	Watch-Dog-Timer Control register	WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0	xx00,0000

Symbol 符号 Function 功能

WDT_FLAG When WDT overflows, this bit is set. It can be cleared by software.
看门狗溢出标志位,当溢出时，该位由硬件置 1，可用软件将其清 0。

EN_WDT Enable WDT bit. When set, WDT is started
看门狗允许位， 当设置为“ 1 ”时，看门狗启动。

CLR_WDT WDT clear bit. When set, WDT will recount. Hardware will automatically clear this bit.
看门狗清“ 0 ”位，当设为“ 1 ”时，看门狗将重新计数。硬件将自动清“ 0 ”此位。

IDLE_WDT When set, WDT is enabled in IDLE mode. When clear, WDT is disabled in IDLE mode
看门狗“ IDLE ”模式位,当设置为“ 1 ”时,看门狗定时器在“ 空闲模式 ”计数
当清“ 0 ”该位时， 看门狗定时器在“ 空闲模式 ”时不计数

PS2, PS1, PS0 Pre-scale value of Watchdog timer is shown as the bellowed table:
看门狗定时器预分频值，如下表所示

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @20MHz
0	0	0	2	39.3 mS
0	0	1	4	78.6 mS
0	1	0	8	157.3 mS
0	1	1	16	314.6 mS
1	0	0	32	629.1 mS
1	0	1	64	1.25S
1	1	0	128	2.5S
1	1	1	256	5S

The WDT period is determined by the following equation 看门狗溢出时间计算
看门狗溢出时间 = (12 x Pre-scale x 32768) / Oscillator frequency
设时钟为 12MHz：
看门狗溢出时间 = (12 x Pre-scale x 32768) / 12000000 = Pre-scale x 393216 / 12000000

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @12MHz
0	0	0	2	65.5 mS
0	0	1	4	131.0 mS
0	1	0	8	262.1 mS
0	1	1	16	524.2 mS
1	0	0	32	1.0485S
1	0	1	64	2.0971S
1	1	0	128	4.1943S
1	1	1	256	8.3886S

设时钟为 11.0592MHz：

看门狗溢出时间 = (12 x Pre-scale x 32768) / 11059200 = Pre-scale x 393216 / 11059200

PS2	PS1	PS0	Pre-scale 预分频	WDT Period @11.0592MHz
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377S
1	0	1	64	2.2755S
1	1	0	128	4.5511S
1	1	1	256	9.1022S

汇编语言程序示例

```
WDT_CONTR   DATA    0E1H ;    或者   WDT_CONTR  EQU   0E1H
;复位入口
    ORG     0000H
    LJMP    Initial
    …
    ORG     0060H
Initial:
    MOV     WDT_CONTR,  #00111100B; Load initial value 看门狗定时器控制寄存器初始化
                ; EN_WDT = 1, CLR_WDT = 1, IDLE_WDT = 1, PS2 = 1, PS1 = 0, PS0 = 0
    …
Main_Loop:
    LCALL   Display_Loop
    LCALL   Keyboard_Loop
    …
    MOV     WDT_CONTR,  #00111100B ;  喂狗，  不要用 ORL    WDT_CONTR,  #00010000B
    …
    LJMP    Main_Loop
```

C语言程序示例

```
#include<reg52.h>
sfr     WDT_CONTR   =   0xe1;
void main()
{   …
    WDT_CONTR   =   0x3c;
       /* 0011,1100 EN_WDT = 1,CLR_WDT = 1,IDLE_WDT = 1,PS2 = 1,PS1 = 0,PS0 = 0 */
    while(1){
        display();
        keyboard();
        …
        WDT_CONTR   =   0x3c;  /* 喂狗，  不要用  WDT_CONTR  =   WDT_CONTR | 0x10;*/
    }
}
```

```
;本程序用于验证 STC12C2052 系列单片机的看门狗及其溢出时间计算公式
;看门狗及其溢出时间 = (N * Pre_scale *32768)/Oscillator frequency
;  N = 12，当在 12 clock mode 时，N = 6，当在 6 clock mode 时。

WDTCR       EQU      0E1H           ;看门狗地址
LED         EQU      P1.5           ;用 P1.5 控制发光二级管

Pre_scale_Word   EQU    0x35        ;清0、启动看门狗，预分频数=64
;f=18.432MHz、12clock mode 时
;   看门狗溢出时间 = (12 * 64 * 32768)/18432000 = 1.36S

    ORG    0000H
    AJMP   main

    ORG    0100H
main:
    CLR    LED                     ;点亮 LED
    ACALL  delay                   ;延时，让 LED 亮大约 1S 的时间

     MOV    WDTCR, #Pre_scale_Word ;启动看门狗，若注释掉本条指令即不启动狗，
                                    ;LED 只会亮一次
    SETB   LED                     ;熄灭 LED
wait:
    SJMP   wait           ;跳转到本语句(停机)，等待看门狗溢出复位，复位后将再次点亮 LED

delay:
    MOV    R0, #0
    MOV    R1, #0
    MOV    R2, #15
delay_loop:
    DJNZ   R0, delay_loop
    DJNZ   R1, delay_loop
    DJNZ   R2, delay_loop
    RET

    END
```

以上程序是按 STC89C51RC 写的，因 STC12C2052AD 是 1T，客户可作适当调整，否则速度大至 12 倍。

STC12C2052AD 系列 1T 单片机通过外部中断从掉电模式唤醒

```
;*************************************************************************
;Wake Up Idle and Wake Up Power Down
;*************************************************************************
    ORG   0000H
    AJMP MAIN

    ORG   0003H
 int0_interrupt:
    CLR   P1.7          ;点亮 P1.7 LED 表示已响应 int0 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0013H
 int1_interrupt:
    CLR   P1.6          ;点亮 P1.6 LED 表示已响应 int1 中断
    ACALL delay         ;延时是为了便于观察，实际应用不需延时
    CLR   EA            ;关闭中断，简化实验．实际应用不需关闭中断
    RETI

    ORG   0100H
 delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #02
 delay_loop:
    DJNZ  R0, delay_loop
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    RET

 main:
    MOV   R3, #0        ;P1 LED 递增方式变化，表示程序开始运行
 main_loop:
    MOV   A, R3
    CPL   A
    MOV   P1, A
    ACALL delay
```

```
    INC   R3
    MOV   A, R3
    SUBB  A, #18H
    JC    main_loop

    MOV   P1, #0FFH      ;熄灭全部灯表示进入 Power Down 状态

    CLR   IT0            ;设置低电平激活外部中断
;   SETB  IT0            ;下降沿激活不了 Power Down 状态下的外部中断。原因是
                         ;MCU 判断下降沿需要 2 个机器周期，而此时 CLOCK 已停止，
                         ;MCU 无法运行 2 个机器周期。
    SETB  EX0            ;允许外部中断0

    CLR   IT1            ;设置低电平激活外部中断
;   SETB  IT1            ;下降沿激活不了 Power Down 状态下的外部中断，原因同上
    SETB  EX1            ;允许外部中断1

    SETB  EA             ;开中断，若不开中断就不能唤醒 Power Down

;下条语句将使 MCU 进入 idle 状态或 Power Down 状态
;低电平激活外部中断可以将 MCU 从 Power Down 状态中唤醒
;其方法为:将外部中断脚拉低

    MOV   A, PCON        ;令 PD=1，进入 Power Down 状态，PD = PCON.2
    ORL   A, #02H
    MOV   PCON, A

    MOV   PCON, #01H     ;删除本语句前的";"，同时将前3条语句前加上注释符号";",
                         ;令 IDL=1，可进入 idle 状态，IDL = PCON.1

    MOV   P1, #0DFH      ;请注意:
                         ; 1.外部中断使MCU退出 Power Down 状态,执行本条指令后
                         ;响应中断，表现为P1.5与P1.7的LED同时亮(INT0唤醒)
                         ; 2.外部中断使MCU退出 idle 状态,先响应中断然后再执行本
                         ;条指令，表现为P1.7的LED先亮(INT0唤醒)P1.5的LED后亮

WAIT1：
    SJMP  WAIT1          ;跳转到本语句，停机

    END
```

STC12C2052AD 系列 1T 8051 单片机IAP 应用
STC12C2052AD 系列 1T 8051 单片机内部EEPROM的应用

-- 利用 IAP 技术可实现 EEPROM，内部 Flash 擦写次数为 100,000 次以上

STC12C2052AD系列 1T 8051 单片机 ISP/IAP 特殊功能寄存器 ISP/IAP SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset Value
ISP_DATA	E2h	ISP/IAP Flash Data Register									1111,1111
ISP_ADDRH	E3h	ISP/IAP Flash Address High									0000,0000
ISP_ADDRL	E4h	ISP/IAP Flash Address Low									0000,0000
ISP_CMD	E5h	ISP/IAP Flash Command Register	-	-	-	-	-	-	MS1	MS0	xxxx,x000
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	0000,x000

ISP_DATA: ISP/IAP 操作时的数据寄存器。
ISP/IAP 从 Flash 读出的数据放在此处，向 Flash 写的数据也需放在此处

ISP_ADDRH: ISP/IAP 操作时的地址寄存器高八位。
ISP_ADDRL: ISP/IAP 操作时的地址寄存器低八位。
ISP_CMD: ISP/IAP 操作时的命令模式寄存器，须命令触发寄存器触发方可生效。

B7	B6	B5	B4	B3	B2	B1	B0	命令 / 操作 模式选择
保留						命令		
-	-	-	-	-	-	0	0	Standby 待机模式，无ISP操作
-	-	-	-	-	-	0	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节读
-	-	-	-	-	-	1	0	从用户的应用程序区对 "Data Flash/EEPROM区" 进行字节编程
-	-	-	-	-	-	1	1	从用户的应用程序区对 "Data Flash/EEPROM区" 进行扇区擦除

;5V 单片机，应在 3.6V 以下时让其复位，3V 单片机，应在 2.4V 以下时让其复位
;在此电压以下，此时再用 ISP/IAP 功能，编程 / 擦除 Flash，不能保证能达到要求

程序在用户应用程序区时，仅可以对数据 Flash 区(EEPROM)进行字节读 / 字节编程 / 扇区擦除。STC12C2052AD 系列单片机出厂时就已完全加密。

ISP_TRIG: ISP/IAP 操作时的命令触发寄存器。
在 ISPEN(ISP_CONTR.7) = 1 时,对 ISP_TRIG 先写入 46h,再写入 B9h,
ISP/IAP 命令才会生效。

ISP_CONTR: ISP/IAP 控制寄存器。

B7	B6	B5	B4	B3	B2	B1	B0	Reset Value
ISPEN	SWBS	SWRST	CMD_FAIL	-	WT2	WT1	WT0	0000,x000

ISPEN: ISP/IAP 功能允许位。0：禁止 ISP/IAP 编程改变 Flash,1:允许编程改变 Flash
SWBS: 软件选择从用户主程序区启动（0），还是从 ISP 程序区启动（1）。
SWRST: 0： 不操作； 1： 产生软件系统复位，硬件自动清零。
CMD_FAIL: 如果送了 ISP/IAP 命令，并对 ISP_TRIG 送 46h/B9h 触发失败，则为 1，需由软件清零。

设置等待时间			CPU 等待时间（CPU 的工作时钟）			
WT2	WT1	WT0	Read	Program	Sector Erase	Recommended System Clock
1	1	1	2	55	21012	1MHz
1	1	0	2	110	42024	2MHz
1	0	1	2	165	63036	3MHz
1	0	0	2	330	126072	6MHz
1	0	0	2	660	252144	12MHz
1	0	0	2	1100	420240	20MHz
1	0	0	2	1320	504288	24MHz
1	0	0	2	1760	672384	30MHz

STC12C2052AD 系列单片机内部可用 Data Flash(EEPROM)的地址(与程序空间是分开的)：
如果对应用程序区进行 IAP 写数据，则该语句会被单片机忽略，继续执行下一句。
程序在用户应用程序区(AP区)时,仅可以对Data Flash(EEPROM)进行IAP/ISP操作,不可以修改自身，这是系统可靠的基础。但 STC12C5052/STC12C5052AD 可以修改自己(灵活)。

STC12C0552,STC12C0552AD,STC12LE0552,STC12LE0552AD
STC12C1052,STC12C1052AD,STC12LE1052,STC12LE1052AD
STC12C2052,STC12C2052AD,STC12LE2052,STC12LE2052AD
STC12C3052,STC12C3052AD,STC12LE3052,STC12LE3052AD
STC12C4052,STC12C4052AD,STC12LE4052,STC12LE4052AD
系列单片机内部可用 Data Flash(EEPROM)的地址：

第一扇区		第二扇区		每个扇区 512字节 建议同一次修改的数据放在同一个扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	
1000h	11FFh	1200h	13FFh	

STC12C5052,STC12C5052AD,STC12LE5052,STC12LE5052AD单片机可对自身内部应用程序区进行 IAP/ISP 操作，故所有部分均可当 Data Flash(EEPROM)使用，其地址如下：

第一扇区		第二扇区		第三扇区		第四扇区		每个扇区 512字节 建议同一次修改的数据放在同一个扇区，不必用满，当然可全用
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0000h	01FFh	0200h	03FFh	0400h	05FFh	0600h	07FFh	
第五扇区		第六扇区		第七扇区		第八扇区		
起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	起始地址	结束地址	
0800h	09FFh	0A00h	0BFFh	0C00h	0DFFh	0E00h	0FFFh	
第九扇区		第十扇区						
起始地址	结束地址	起始地址	结束地址					
1000h	11FFh	1200h	13FFh					

STC12C2052AD 系列 IAP 应用汇编简介
STC12C2052AD 系列 内部 EEPROM 的应用

```
;用 DATA 还是 EQU 声明新增特殊功能寄存器地址要看你用的汇编器 / 编译器
    ISP_DATA        DATA    0E2h;    或   ISP_DATA        EQU     0E2h
    ISP_ADDRH       DATA    0E3h;    或   ISP_ADDRH       EQU     0E3h
    ISP_ADDRL       DATA    0E4h;    或   ISP_ADDRL       EQU     0E4h
    ISP_CMD         DATA    0E5h;    或   ISP_CMD         EQU     0E5h
    ISP_TRIG        DATA    0E6h;    或   ISP_TRIG        EQU     0E6h
    ISP_CONTR       DATA    0E7h;    或   ISP_CONTR       EQU     0E7h
;定义 ISP/IAP 命令及等待时间
    ISP_IAP_BYTE_READ       EQU  1    ;字节读
    ISP_IAP_BYTE_PROGRAM    EQU  2    ;字节编程,前提是该字节是空，0FFh
    ISP_IAP_SECTOR_ERASE    EQU  3    ;扇区擦除,要某字节为空，要擦一扇区
    WAIT_TIME       EQU  0            ;设置等待时间,30MHz 以下 0,24M 以下 1,
        ;20MHz 以下 2,12M 以下 3,6M 以下 4,3M 以下 5,2M 以下 6,1M 以下 7,
;字节读
    MOV  ISP_ADDRH,    #BYTE_ADDR_HIGH    ;    送地址高字节
    MOV  ISP_ADDRL,    #BYTE_ADDR_LOW     ;    送地址低字节
    CLR  EA        ;    关中断,此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_CONTR,    #WAIT_TIME         ;    设置等待时间
    ORL  ISP_CONTR,    #10000000B         ;    允许 ISP/IAP 操作
    MOV  ISP_CMD,      #ISP_IAP_BYTE_READ ;    送字节读命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #46h      ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #0B9h     ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                           ;数据读出到 ISP_DATA 寄存器后，CPU 继续执行程序
    MOV  ISP_CONTR,    #00000000B ;禁止 ISP/IAP 操作
    MOV  ISP_CMD,      #00000000B ;去除 ISP/IAP 命令
    MOV  ISP_TRIG,     #00000000B ;防止 ISP/IAP 命令误触发
    MOV  ISP_ADDRH,    #0         ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV  ISP_ADDRL,    #0         ;送地址低字节单元为 00,防止误操作
    SETB EA    ;    开中断，CPU 处理完 ISP/IAP 动作即可开中断
    MOV  A,            ISP_DATA   ;将读出的数据送往 Acc
```

```
;字节编程，该字节为 FFh/ 空时，可对其编程，否则不行,要先执行扇区擦除
    MOV  ISP_DATA,    #ONE_DATA          ;    送字节编程数据到 ISP_DATA
    MOV  ISP_ADDRH,   #BYTE_ADDR_HIGH    ;    送地址高字节
    MOV  ISP_ADDRL,   #BYTE_ADDR_LOW     ;    送地址低字节
    CLR  EA       ;    关中断,此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_CONTR,   #WAIT_TIME         ;    设置等待时间
    ORL  ISP_CONTR,   #10000000B         ;    允许 ISP/IAP 操作
    MOV  ISP_CMD,     #ISP_IAP_BYTE_PROGRAM   ;送字节编程命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,    #46h      ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,    #0B9h     ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                                  ;字节编程成功后，CPU 继续执行程序
    MOV  ISP_CONTR,   #00000000B         ;禁止 ISP/IAP 操作
    MOV  ISP_CMD,     #00000000B         ;去除 ISP/IAP 命令
    MOV  ISP_TRIG,    #00000000B         ;防止 ISP/IAP 命令误触发
    MOV  ISP_ADDRH,   #0                 ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV  ISP_ADDRL,   #0                 ;送地址低字节单元为 00,防止误操作
    SETB EA   ;    开中断，CPU 处理完 ISP/IAP 动作即可开中断
```

小常识： (STC 单片机的 Data Flash 当 EEPROM 功能使用)

3 个基本命令 - - - - 字节读，字节编程，扇区擦除

字节编程：如果该字节是“1111,1111B”,则可将其中的“1”编程为“0”，如果该字节中有位为“0”，则须先将整个扇区擦除，因为只有“扇区擦除”才可以将“0”变为“1 ”。

扇区擦除：只有“扇区擦除”才可能将“0 ”擦除为“1 ”。

大建议：

1.同一次修改的数据放在同一扇区中，单独修改的数据放在另外的扇区,就不须读出保护。

2.如果一个扇区只用一个字节，那就是真正的 EEPROM,STC 单片机的 Data Flash 比外部 EEPROM 要快很多，读一个字节 / 编程一个字节 / 擦除一个扇区大概是 10uS/60uS/10mS。

3.如果同一个扇区中存放了一个以上的字节，某次只需要修改其中的一个字节或一部分字节时，则另外的不需要修改的数据须先读出放在 STC 单片机的 RAM 中，然后擦除整个扇区，再将需要保留的数据和需修改的数据一并写回该扇区中。这时每个扇区使用的字节数是使用的越少越方便(不需读出一大堆需保留数据)。

```
;扇区擦除，没有字节擦除，只有扇区擦除，512 字节 / 扇区,每个扇区用得越少越方便
;如果要对某个扇区进行擦除，而其中有些字节的内容需要保留，则需将其先读到单片机
;内部的 RAM 中保存，再将该扇区擦除，然后将须保留的数据写回该扇区，所以每个扇区
;中用的字节数越少越好，操作起来越灵活越快(每个扇区只用 1-128 字节以内较方便)


    MOV  ISP_ADDRH,    #SECTOR_FIRST_BYTE_ADDR_HIGH      ;送扇区起始地址高字节
    MOV  ISP_ADDRL,    #SECTOR_FIRST_BYTE_ADDR_LOW       ;送扇区起始地址低字节
    CLR  EA        ;   关中断,此时各中断请求，会被挂起，一开中断，立即响应
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_CONTR,    #WAIT_TIME     ;设置等待时间
    ORL  ISP_CONTR,    #10000000B     ;允许 ISP/IAP
    MOV  ISP_CMD,      #ISP_IAP_SECTOR_ERASE             ;送扇区擦除命令
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #46h       ;先送 46h,再送 B9h 到 ISP/IAP 触发寄存器
;加入软件陷阱判断，如为非法状态，则让单片机进入掉电模式或软复位
    MOV  ISP_TRIG,     #0B9h      ;送完 B9h 后，ISP/IAP 命令立即被触发起动
;CPU 等待 IAP 动作完成后，才会继续执行程序，要先关中断（EA），
;再送 46h,B9h 到 ISP/IAP 触发寄存器,起动 ISP/IAP 命令,关中断在触发之前即可
    NOP                           ;扇区擦除成功后，CPU 继续执行程序
    MOV  ISP_CONTR,    #00000000B     ;禁止 ISP/IAP 操作
    MOV  ISP_CMD,      #00000000B     ;去除 ISP/IAP 命令
    MOV  ISP_TRIG,     #00000000B     ;防止 ISP/IAP 命令误触发
    MOV  ISP_ADDRH,    #0             ;送地址高字节单元为 00,指向非 EEPROM 区
    MOV  ISP_ADDRL,    #0             ;送地址低字节单元为 00,防止误操作
```

```
;从用户应用程序区(AP 区)软件复位并切换到 ISP 程序区开始执行程序
    MOV  ISP_CONTR,    #01100000B     ;SWBS = 1(选择 ISP 区), SWRST = 1(软复位)

;从 ISP 程序区软件复位并切换到用户应用程序区(AP 区)开始执行程序
    MOV  ISP_CONTR,    #00100000B     ;SWBS = 0(选择 AP 区), SWRST = 1(软复位)

;使用 ISP/IAP 功能的朋友尽量给 13922805190（姚工）一个电话交流一下
;建议在打开 ISP 及在 ISP 触发送 46H,B9H 之前三个地方，各加些软件陷阱(欢迎来电讨论)
;5V 单片机，应在 3.6V 以下时让其复位，3V 单片机，应在 2.4V 以下时让其复位
;在此电压以下，此时再用 ISP/IAP 功能，编程 / 擦除 Flash，不能保证能达到要求
```

```
;本应用程序介绍 STC12C2052AD 系列单片机内部 EEPROM 的使用,及怎样用软件方法加强应用软件抗干扰能力
;-------------------------------------------------
;定义与 IAP 有关的特殊功能寄存器
ISP_DATA         EQU 0E2H
ISP_ADDRH        EQU 0E3H
ISP_ADDRL        EQU 0E4H
ISP_CMD          EQU 0E5H
ISP_TRIG         EQU 0E6H
ISP_CONTR        EQU 0E7H
;-------------------------------------------------
;定义常量
;------------------------
;Flash 操作等待时间
;ENABLE_ISP       EQU 87H             ;<1MHz
;ENABLE_ISP       EQU 86H             ;<2MHz
;ENABLE_ISP       EQU 85H             ;<3MHz
;ENABLE_ISP       EQU 84H             ;<6MHz
;ENABLE_ISP       EQU 83H             ;<12MHz
;ENABLE_ISP       EQU 82H             ;<20MHz
;ENABLE_ISP       EQU 81H             ;<24MHz
ENABLE_ISP       EQU 80H             ;<30MHz

;------------------------
;选择 MCU 型号
;DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C0552AD
;DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C1052AD
DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C2052AD
;DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C3052AD
;DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C4052AD
;-------------------------------------------------
;定义变量
check_RAM_20H    EQU 20H             ;抗干扰监控 RAM
check_RAM_30H    EQU 30H             ;抗干扰监控 RAM
password         EQU 31H
temp             EQU 32H             ;临时变量
;-------------------------------------------------
    ORG   0000H
    AJMP  main
;-------------------------------------------------
    ORG   0100H
main:
    MOV   SP, #0E0H
    ACALL initiate_system            ;系统初始化

    MOV   DPTR, #DATA_FLASH_START_ADDRESS
    MOV   password, #0B1H            ;设置口令
    ACALL sector_erase               ;擦除扇区
```

```
    ACALL IAP_Enable                ;检查 check_RAM，打开 IAP 功能

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 5
    MOV   A, #55H                   ;写入 flash 的第 1 个字节为 55H
    ACALL byte_program              ;字节编程并校验，返回:C=0--成功，C=1--失败
    JC    display_ERROR_LED

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 6
    MOV   A, #0AAH                  ;写入 flash 的第 2 个字节为 AAH
    ACALL byte_program              ;字节编程并校验，返回:C=0--成功，C=1--失败
    JC    display_ERROR_LED

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 7
    MOV   A, #0CCH                  ;写入 flash 的第 3 个字节为 CCH
    ACALL byte_program              ;字节编程并校验，返回:C=0--成功，C=1--失败
    JC    display_ERROR_LED

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 5
    ACALL byte_verify               ;读回写入 flash 的第1个字节
    CPL   A                         ;显示
    MOV   P1, A
    ACALL Delay

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 6
    ACALL byte_verify               ;读回写入 flash 的第2个字节
    CPL   A                         ;显示
    MOV   P1, A
    ACALL Delay

    MOV   DPTR, #DATA_FLASH_START_ADDRESS + 7
    ACALL byte_verify               ;读回写入 flash 的第3个字节
    CPL   A                         ;显示
    MOV   P1, A
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器

WAIT1：
    SJMP  WAIT1                     ;跳转到本语句，在本语句无限循环

;-----------------------------------------------
display_ERROR_LED:
    MOV   P1, #0BFH                 ;点亮第7个发光二极管表示写入 flash 出错
    ACALL Delay
    ACALL Delay
    JMP   halt                      ;停机
;-----------------------------------------------
;读一字节
;调用前需打开 IAP 功能
```

```
;入口:DPTR = 字节地址
;返回:A = 读出字节

byte_verify:
    MOV   ISP_CMD, #01              ;Select Read AP Mode
    ACALL Set_ISP_ADD
read_RD_RC:
    MOV   ISP_DATA, #00H            ;清数据寄存器
    CLR   EA
    MOV   ISP_TRIG, #46H            ;Trigger ISP processing
    MOV   ISP_TRIG, #0B9H
    NOP
    SETB  EA
    ;Now in processing.(CPU will halt here before completing)
    MOV   A, ISP_DATA               ;Data will be in ISP_DATA
    RET

;-----------------------------------------------

;字节编程并校验
;调用前需打开 IAP 功能
;入口:DPTR = 字节地址, A= 字节
;返回:C=0--成功, C=1--失败
byte_program:
;    MOV   A, #0B3H                 ;需要时可去除注释, 检查口令
;    ACALL Check_Password

    PUSH  temp
    MOV   temp, A
    MOV   ISP_CMD, #02H             ;Select Byte Program Mode
    ACALL Set_ISP_ADD
    MOV   ISP_DATA, A               ;Fill the data to be programmed in ISP_DATA
    ACALL trigger_ISP               ;Trigger ISP processing
    ACALL byte_verify               ;读回写入的字节
    CJNE  A, temp, byte_program_ERR
    CLR   C
    POP   temp
    RET
byte_program_ERR:
    SETB  C
    POP   temp
    RET
;-----------------------------------------------
;擦除扇区, 入口:DPTR = 扇区地址
sector_erase:
    MOV   A, #0B1H                  ;检查口令
    ACALL Check_Password
```

```
    ACALL IAP_Enable                ;检查 check_RAM，打开 IAP 功能
    MOV   ISP_CMD, #03              ;Select Page Erase Mode
    ACALL Set_ISP_ADD
    ACALL trigger_ISP               ;Trigger ISP processing
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    RET

;-----------------------------------------------

trigger_ISP:
    CLR  EA
    ACALL Check_RAM_At_20H_30H      ;检查 20H、30H 单元是否变化
    MOV  ISP_TRIG, #46H             ;Trigger ISP processing
    ACALL Check_RAM_At_20H_30H      ;检查 20H、30H 单元是否变化
    MOV  ISP_TRIG, #0B9H            ;Trigger ISP processing
    NOP
    SETB EA
    RET

;-----------------------------------------------

Set_ISP_ADD:
    MOV  ISP_ADDRH, DPH             ;Fill page address in ISP_ADDRH & ISP_ADDRL
    MOV  ISP_ADDRL, DPL
    RET

;-----------------------------------------------
IAP_Enable:                         ;检查 check_RAM，打开 IAP 功能
    ACALL Check_ISP_Sfr             ;检查与 ISP 有关的特殊功能寄存器
    ACALL Check_RAM_At_20H_30H      ;检查 20H、30H 单元是否变化
    MOV   ISP_ADDRH, #0H            ;ISP_ADDRH 不可以指向 ISP 区
    MOV   ISP_ADDRL, #05H           ;ISP_ADDRH，ISP_ADDRL 不可以同时为 0
    MOV   ISP_CONTR, #ENABLE_ISP    ;打开 IAP 功能，设置 Flash 操作等待时间
    RET

;-----------------------------------------------

IAP_Disable:                        ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    CLR   A
    MOV   ISP_CONTR, A              ;关闭 IAP 功能
    MOV   ISP_CMD, A
    MOV   ISP_TRIG, A
    MOV   ISP_ADDRH, #0FFH
    MOV   ISP_ADDRL, #0FFH          ;ISP_ADDRL 可以置为 0FFH
    RET

;-----------------------------------------------
```

```
Check_ISP_Sfr:                          ;检查与 ISP 有关的特殊功能寄存器
    CLR   A
; ~~CJNE  A, ISP_CONTR, Check_ISP_Sfr_Err~~ 不要用此句，STC89C51RC/RD+系列可用
    CJNE  A, ISP_CMD, Check_ISP_Sfr_Err
    DEC   A
    CJNE  A, ISP_ADDRH, Check_ISP_Sfr_Err
    RET
;---------------------
Check_ISP_Sfr_Err:
    JMP   halt   ;停机， 实际应用时建议是软复位到用户程序区  MOV  ISP_CONTR, #00100000B

;-----------------------------------------------
initiate_check_RAM:                     ;初始化 check_RAM
    MOV   check_RAM_20H, #0A5H
    MOV   check_RAM_30H, #05AH
    RET
;-----------------------------------------------

;检查 check_RAM 单元变化否，若有变化，预示着 RAM 区内容因干扰等原因被破坏。
Check_RAM_At_20H_30H:
    MOV   A, #0A5H
    CJNE  A, check_RAM_20H, Check_ISP_Sfr_Err  ;停机
    MOV   A, #05AH
    CJNE  A, check_RAM_30H, Check_ISP_Sfr_Err  ;停机
    RET

;-----------------------------------------------

Check_Password:                         ;检查口令，若口令不对就停机
    CJNE  A, password, Check_ISP_Sfr_Err   ;停机
    MOV   password, #0                  ;清口令
    RET

;-----------------------------------------------

halt:                       ;停机，也可改成软复位、进入 power down 或 idle 状态
    ACALL IAP_Disable       ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
    MOV   Password, #0      ;清口令
    MOV   check_RAM_20H, #0
    MOV   check_RAM_30H, #0
    MOV   P1, #7FH                  ;点亮第 8 个发光二极管表示停机

WAIT2：
    SJMP  WAIT2                     ;跳转到本语句，在本语句无限循环
;-----------------------------------------------
initiate_system:                    ;系统初始化
    ACALL IAP_Disable               ;关闭 IAP 功能，清与 ISP 有关的特殊功能寄存器
```

```
    MOV   Password, #0              ;清口令
    ACALL initiate_check_RAM        ;初始化监控 RAM
    RET

;-----------------------------------------------
Delay:
    CLR   A
    MOV   R0, A
    MOV   R1, A
    MOV   R2, #20H
Delay_Loop:
    DJNZ  R0, Delay_Loop
    DJNZ  R1, Delay_Loop
    DJNZ  R2, Delay_Loop
    RET

;-----------------------------------------------

    END
;***********************************************
```

STC12C2052AD 系列单片机定时器的使用

1.1 定时器 0 和 1

定时和计数功能由特殊功能寄存器 TMOD 的控制位 $\overline{C}$/T 进行选择，TMOD 寄存器的各位信息如表 1 所列。可以看出，2 个定时 / 计数器有 4 种操作模式，通过 TMOD 的 M1 和 M0 选择。2 个定时 / 计数器的模式 0 、1 和 2 都相同，模式 3 不同，各模式下的功能如下所述。

表 1 寄存器 TMOD 各位的功能描述

TMOD 地址：89H 复位值：00H

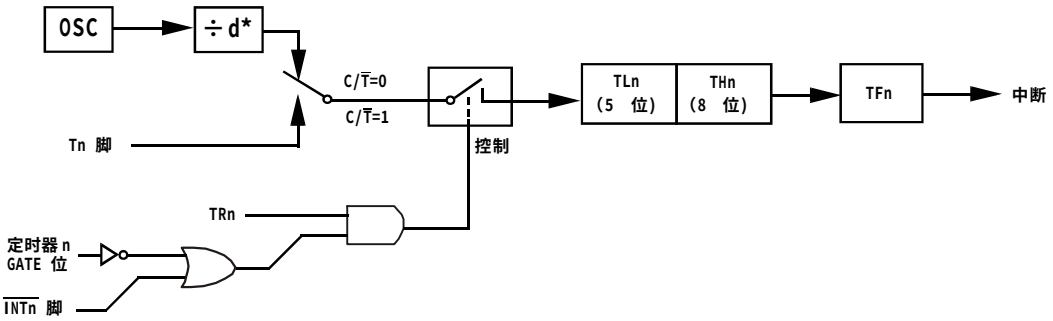
不可位寻址

7	6	5	4	3	2	1	0
GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
定时器 1				定时器 0			

位	符号	功能
TMOD.3/ TMOD.7	GATE	用于定时器 1，置位时只有在 $\overline{INT1}$ 脚置高及 TR1 控制置位时才可打开定时器 / 计数器。清零时，置位 TR1 即可打开定时器 / 计数器。
TMOD.2/ TMOD.6	C/$\overline{T}$	控制定时器 1 用作定时器或计数器，清零则用作定时器（从内部系统时钟输入），置位用作计数器（从 T n 脚输入）
	M1、M0	定时器模式选择
	M1、M0	定时器模式
	0 0	8048 定时器 TLn 用做 5 位预分频器
	0 1	16 位定时器 / 计数器，无预分频器
	1 0	8 位自装载定时器，当溢出时将 THn 存放的值装入 TLn。
	1 1	定时器 0 此时作为双 8 位定时 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 控制位控制，在这种模式下定时 / 计数器 1 关闭

1 . 模式 0

将定时器设置成模式 0 时类似 8048 定时器，即 8 位计数器带 32 分频的预分频器。图 1 所示为模式 0 工作方式。此模式下，定时器寄存器配置为 13 位寄存器。当计数从全为“ 1 ”翻转为全为“ 0 ”时，定时器中断标志位 TFn 置位。当 TRn=1、同时 GATE=0 或 $\overline{INTn}$=1 时，定时器计数。置位 GATE 时允许由外部输入 $\overline{INTn}$ 控制定时器，这样可实现脉宽测量。TRn 为 TCON 寄存器内的控制位，TCON 寄存器各位的具体功能描述如表 2 所列。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 1 定时 / 计数器 0 / 1 的模式 0：13 位定时 / 计数器

表 2 寄存器 TCON 各位的功能描述

TCON 地址：88H									
可位寻址		7	6	5	4	3	2	1	0	
复位值：00H		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
位	符 号	功 能								
TCON.7	TF1	定时器 1 溢出标志。定时 / 计数器溢出时由硬件置位；中断处理时由硬件清除，或用软件清除								
TCON.6	TR1	定时器 1 运行控制位。由软件置位 / 清零，将定时 / 计数器打开 / 关闭								
TCON.5	TF0	定时器 0 溢出标志。定时 / 计数器溢出时由硬件置位；中断处理时由硬件清除，或用软件清除								
TCON.4	TR0	定时器 0 运行控制位。由软件置位 / 清零，将定时 / 计数器打开 / 关闭								
TCON.3	IE1	中断 1 边沿触发标志。当检测到外部中断 1 边沿时由硬件置位，中断处理时清零								
TCON.2	IT1	中断 1 触发类型控制位。由软件置位 / 清零，以选择外部中断下降沿 / 低电平方式触发								
TCON.1	IE0	中断 0 边沿触发标志。当检测到外部中断 0 边沿时由硬件置位，中断处理时清零								
TCON.0	IT0	中断 0 触发类型控制位。由软件置位 / 清零，以选择外部中断下降沿 / 低电平方式触发								

该 13 位寄存器包含 THn 全部 8 个位及 TLn 的低 5 位。TLn 的高 3 位不定，可将其忽略。置位运行标志（TRn）不能清零此寄存器。模式 0 的操作对于定时器 0 及定时器 1 都是相同的。2 个不同的 GATE 位（TMOD.7 和 TMOD.3）分别分配给定时器 0 及定时器 1。

2. 模式 1

模式 1 除了使用了 THn 及 TLn 全部 16 位外，其他与模式 0 完全相同。

3. 模式 2

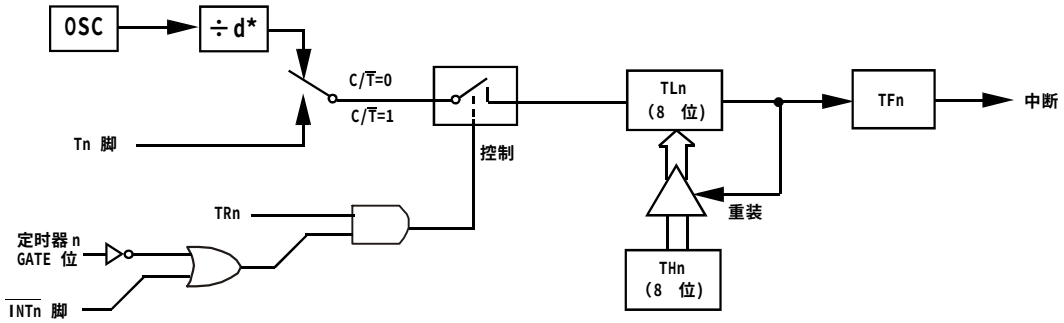
此模式下定时器寄存器作为可自动重装的 8 位计数器（TLn），如图 2 所示。TLn 的溢出不仅置位 TFn，而且将 THn 内容重新装入 TLn，THn 内容由软件预置，重装时 THn 内容不变。模式 2 的操作对于定时器 0 及定时器 1 是相同的。

4. 模式 3

在模式 3 中，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。

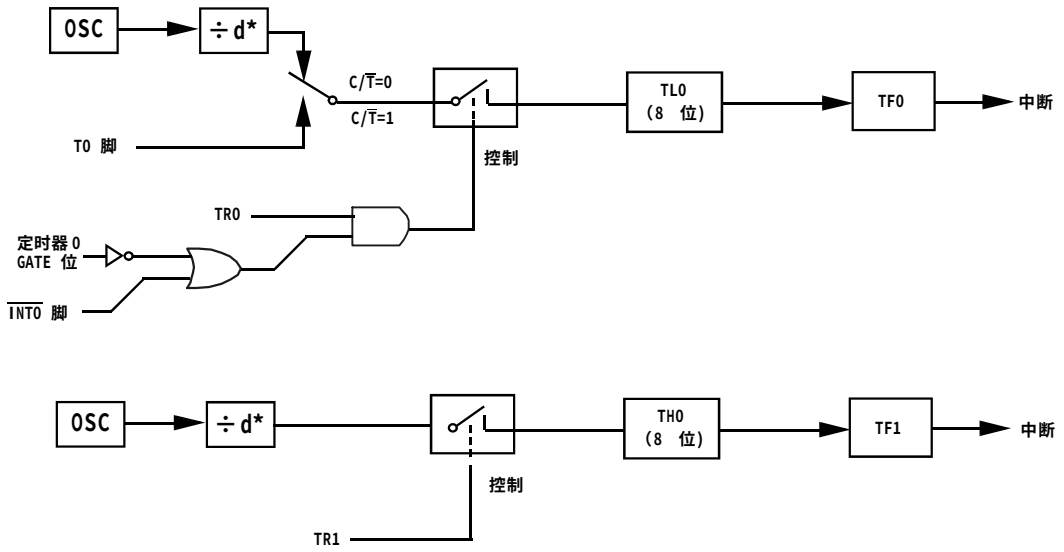
此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。图 3 为模式 3 时的定时器 0 逻辑。TL0 占用定时器 0 的控制位：C/$\overline{T}$、GATE、TR0、$\overline{INT0}$ 及 TF0。TH0 限定为定时器功能（计数器周期），占用定时器 1 的 TR1 及 TF1。此时，TH0 控制定时器 1 中断。

模式 3 可用于需要一个额外的 8 位定时器场合。定时器 0 工作于模式 3 时，80C51 看似有 3 个定时器 / 计数器。当定时器 0 工作于模式 3 时，定时器 1 可通过开关进入 / 退出模式 3，它仍可用作串行端口的波特率发生器，或者应用于任何不要求中断的场合。



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 2 定时 / 计数器 0 / 1 的模式 2：8 位自动重装



* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 3 定时 / 计数器 0 的模式 3：双 8 位计数器

STC12C2052AD 系列 编译器 / 汇编器，编程器，仿真器

STC 单片机应使用何种编译器 / 汇编器

1.任何老的编译器 / 汇编器都可以支持，流行用 Keil C51

2.把 STC 单片机，当成 Intel 的 8052/87C52/87C54/87C58,Philips 的 P87C52/P87C54/P87C58 就可以了. 如果要用到扩展的专用特殊功能寄存器，直接对该地址单元设置就行了，当然先声明特殊功能寄存器的地址较好

编程烧录器:

我们有： STC-ISP 经济型下载编程工具(人民币 50 元，可申请免费样品)

仿真器:如您已有老的仿真器，可仿真普通 8052 的基本功能

STC122052AD 系列单片机扩展功能如它仿不了

可以用 STC-ISP 直接下载用户程序看运行结果就可以了

无须添加新的设备

附录 J:

指令系统与程序设计

执行软件是微型计算机与通用数字集成电路的主要区别，也是微电子技术区别于通用电器和电子技术的根本特征。

软件是由具有一定意义的指令组成的。一台计算机所执行的指令集合就是它的指令系统。指令系统是计算机厂商定义的，它成为应用计算机必须理解和遵循的标准。每种计算机都有自己专用的指令系统。

指令常以英文名称或缩写形式作为助记符。用助记符表示的指令称为汇编语言，用汇编语言编写的程序称为汇编语言程序。

目前单片机主要使用汇编语言，指令系统的学习和应用是使用单片机的重要前提。

STC89 系列单片机与 MCS-51 系列在软件上完全兼容，编制的汇编语言程序可运行于这两种系列单片机。也就是说，STC89 系列单片机采用的也是 MCS-51 指令系统。本章详细介绍该指令系统及其编程方法。

1 指令格式及其符号说明

指令的表示方法称为指令格式。一条指令通常由两部分组成：操作码和操作数。操作码规定指令执行什么操作，而操作数是操作的对象。操作数可以是一个具体的数据，也可以是存储数据的地址或寄存器。指令的基本格式如下：

操作码	操作数（地址码、寄存器或立即数）

汇编语言编写的程序必须翻译成单片机可执行的机器码。根据机器码的长短，可分为单字节、双字节和 3 字节等不同长度的指令。

1．单字节指令

指令系统中有些指令的功能很专一而明确，不需要具体指定操作数，便形成了单字节指令。单字节指令的机器码只有一个字节，操作码和操作数同在其中。例如，指令 INC DPTR，功能为数据指针加 1，指令码为

1010 0011	A3H

有些指令的操作数在工作寄存器 R0～R7 中，寄存器的编码可用 3 位二进制数表示。例如，指令 MOV A，Rn，功能是工作寄存器向累加器传输数据，指令码为

1110 1rrr

用 rrr 表示工作寄存器的二进制编码。对于不同的工作寄存器，单字节的机器码如下表所列。

指令 MOV A，Rn 指令码

指令	指令码（机器码）	
	二进制	十六进制
MOV A，R0	1110 1000	E8H
MOV A，R1	1110 1001	E9H
MOV A，R2	1110 1010	EAH
MOV A，R3	1110 1011	EBH
MOV A，R4	1110 1100	ECH
MOV A，R5	1110 1101	EDH
MOV A，R6	1110 1110	EEH
MOV A，R7	1110 1111	EFH

2．双字节指令

双字节指令的第一字节是操作码，第二字节是操作数。例如，指令 MOV A，#data，功能是将立即数传送到 A，指令码为

0111 0100
立即数

例如，指令 MOV A，#35H 的指令码为 7435H。

3．3 字节指令

3 字节指令中，操作码占一字节，操作数占两字节。其中操作数既可以是数据，也可以是地址。例如，指令 ANL direct，#data，功能是直接地址单元中的内容与立即数进行“与”操作，结果存于直接地址单元，指令码为

0101 0011
直接地址
立即数

例如，指令 ANL 35H，#20H 的机器码为 533520H。

在介绍指令之前，先将指令中使用的一些符号意义作简要说明。

- Rn ——当前工作寄存器 R0～R7，即 n＝0～7，在指令中表示寄存器寻址方式。
- Ri ——间接寻址的寄存器 R0 和 R1，即 i=0, 1，在指令中表示间接寻址方式。
- direct ——8 位直接地址，在指令中表示直接寻址方式，寻址范围为 00H～FFH。
- #data ——8 位立即数，表示立即数寻址方式。
- #data16 ——16 位立即数，表示立即数寻址方式。
- addr16 ——16 位目的地址，只限用于 LCALL 和 LJMP 指令。
- addr11 ——11 位目的地址，只限用于 ACALL 和 AJMP 指令。
- rel ——相对转移指令中的偏移量，为 8 位带符号补码数，在指令中表示相对寻址方式。
- DPTR ——数据指针，16 位。
- bit ——内部数据 RAM 和特殊功能寄存器中的可寻址位。

- A ——表示累加器。
- ACC ——直接寻址方式的累加器。
- B ——寄存器 B。
- C ——进位标志位，可作为位处理器的位累加器，也称为累加位。在指令中代表 CY。
- @ ——间址寄存器的前缀标志。
- / ——加在位地址前面，表示该位状态取反。
- （X）——某个寄存器或某地址单元中的内容。
- ((X)) ——由 X 间接寻址单元中的内容。
- ←——箭头右边的内容传送到箭头左边的存储器单元或寄存器中，即表示数据的传送方向。
- →——箭头左边的内容传送到箭头右边的存储器单元或寄存器中，即表示数据的传送方向。

2 寻址方式

指令执行是都要应用操作数。指令必须指明如何取得操作数，也必须指明程序转移目的地址。所谓寻址，就是如何指定操作数所在的单元，或者如何指定程序转移的目的地址。根据指定的方法不同，形成了不同的寻址方式。MCS-51 指令系统有 7 种不同的寻址方式，下面分别介绍。

1． 寄存器寻址方式

寄存器寻址时，指令中操作数为某一寄存器的内容。指定了寄存器，就指定了操作数。该寻址方式中，用符号名称表示寄存器。

寄存器寻址方式所使用的寄存器包括：

1） 工作寄存器 R0～R7，只能寻址当前寄存器组，即由 PSW 中的 RS1 和 RS0 位的状态对应的当前寄存器组。

2） 部分特殊功能寄存器，例如 A、AB 寄存器对以及数据指针 DPTR 等。

例如：

```
INC  R0 ;(R0) ← (R0) +1
```

其功能是把寄存器 R0 的容量加 1，再送回 R0 中。由于操作数在 R0 中，指定了 R0，也就得到了操作数。

2． 直接寻址方式

直接寻址时，指令中操作数部分直接给出了操作数地址。例如：

```
MOV  A，4AH ;(A) ← (4AH)
```

该指令的功能是把片内 RAM 4AH 单元的内容送入累加器 A。指定了地址 4AH，也就的到了操作数。

直接寻址中的操作数以存储单元形式出现，因此直接寻址方式只能用 8 位二进制数表示的地址，寻址范围只限于内部 RAM，即：

1） 片内 RAM 低 128 单元，在指令中直接以单元地址形式给出。

2）特殊功能寄存器。特殊功能寄存器除了用单元地址形式给出外，还可以用寄存器的名称符号表示。应当指出，直接寻址方式是访问特殊功能寄存器的主要方法。例如：

```
MOV  A，P1    ;(A) ← (P1)
MOV  A，90H   ;(A) ← (90H)
```

由于 SFR P1 的地址为 90H，两条指令本质上是一样的，有相同的机器码，都是直接寻址方式。

3． 寄存器间接寻址方式

寄存器间接寻址时，指令中给出的寄存器内容为操作数地址，而不是操作数本身，即寄存器

为地址指针。

为区别寄存器寻址和寄存器间接寻址，在寄存器间接寻址中，应在寄存器的名称前加前缀@。例如：

```
MOV  R1，#60H
MOV  A，@R1
```

该指令的功能是将60H单元的内容送入累加器A。

2） 外部数据RAM空间的256个单元。例如：

```
MOVX  A，@R1
```

由R1中内容指定的外RAM单元内容送入累加器A。

用DPTR作间址寄存器，其形式为@DPTR，可寻址外部RAM 64 KB（0000H～FFFH)。例如：

```
MOVX  @DPTR，A
```

将累加器的内容传送到由DPTR内容指定的片外RAM16位地址单元。

堆栈操作指令（PUSH和POP）也应算作是寄存器间接寻址，即以堆栈指针（SP）作间址寄存器的间接寻址方式，只不过SP不出现在堆栈操作指令中。

4．立即寻址方式

立即寻址方式是由指令直接给定操作数的方式。例如：

```
MOV  A，#48H；(A) ← #48H
```

其中#作为立即数的标志符。指令的功能是将数据48H送入累加器A。

除8位立即数外，MCS-51指令系统中还有一条16位立即数传送指令，以#data16表示16位立即数。该指令为

```
MOV  DPTR，#data16
```

其功能是将16位立即数送入数据指针DPTR。例如：

```
MOV  DPTR，#1234H
```

其功能是将12H送入DPH，34H送入DPL。

5．变址寻址方式

变址寻址方式是以程序计数器PC或数据指针DPTR作为基址寄存器，以累加器A作为变址寄存器，这两者内容之和为有效地址。例如：假定指令执行前（A）= 54H，(DPTR) = 3F21H，执行指令

```
MOVC  A，@A+DPTR
```

其功能是将程序存储器3F75H单元的内容读入累加器A。

这类寻址方式特别适用于查表。DPTR可指向64KB存储空间；@A+PC指向以PC当前值为起始地址的256个字节单元。

对变址寻址方式说明如下：

1）变址寻址方式只能对程序存储器寻址,或者说它是专门针对程序存储器的寻址方式。

2） 变址寻址指令只有3条，即

```
MOVC  A，@A+DPTR
MOVC  A，@A+PC
JMP      @A+DPTR
```

前两条是程序存储器指令，最后一条是无条件转移指令。

3） 变址寻址方式中的A、DPTR以及PC中的内容为无符号数。

4）尽管变址寻址方式比较复杂,但变址寻址的指令却都是单字节指令。

6．位寻址方式

位寻址时，操作数是二进制数表示的地址，其位地址出现在指令中。例如：

CLR bit

该指令使地址为 bit 的位单元清 0。

位寻址的寻址范围如下：

1） 片内 RAM 中的位寻址区。 其单元地址为 20H～2FH，共 16 个单元 128 位，位地址为 00H～7FH。对这 128 位的寻址可使用直接地址表示。

2） 特殊功能寄存器的可寻址位。对这些寻址位在指令中常用以下几种表示方法：

- 直接使用位地址，例如 PSW 中的位 5 地址为 D5H；
- 位名称表示法，例如 PSW 的位 5 是 F0 标志位，可使用 F0 表示；
- 特殊功能寄存器符号名称加位数的表示方法，例如 PSW 的位 5 可表示成 PSW.5。

7．相对寻址方式

前面介绍的 6 种寻址方式主要解决操作数的给出问题，而相对寻址方式是为解决程序转移而专门设置的，为转移指令所采用。

相对寻址是以 PC 的相对值为基地址，加上指令中所给定的偏移量，形成有效转换地址。偏移量是带符号的 8 位二进制数，以补码的形式出现。因此，程序的转移范围为 +127～-128。转移目的地址可用如下公式表示：

目的地址 = 转移指令所在地址 + 转移指令字节数 +rel

例如：

SJMP rel；(PC) ← (PC) + 2 + rel

执行这条指令时，程序转移到指令 PC 值加 2 再加 rel 的方向地址处。其中，2 为该指令的字节长度，rel 以 8 位带符号的补码形式出现。

3 指令分类介绍

MCS-51 指令系统共有 111 条指令，分为 5 大类：

- 数据传送类指令（29 条）；
- 算术运算类指令（24 条）；
- 逻辑运算及移位类指令（24 条）；
- 控制转移类指令（17 条）；
- 位操作类指令（17 条）。

1 数据传送类指令

数据传送操作属于复制性质,而不是搬家性质。一般传送类指令的助记符号为 MOV,通用格式为

MOV <目的操作数>,<源操作数>

传送指令中有从右向左传送数据的约定,即指令的右边操作数为源操作数,表达的是数据的来源,而左边的操作数为目的操作数,表达的是传送数据的目的地址。

源操作数可以是：累加器 A、工作寄存器 Rn、直接地址 direct、间址寄存器和立即数。目的操作数可以是：累加器 A、工作寄存器 Rn、直接地址 direct 和间址寄存器。两者只差一个立即数。

在数据传送操作中，除了奇偶标志 P 外，一般不影响程序状态字 PSW（指令直接访问 PSW 除外）。

1．一般传送指令

（1） 以累加器 A 为目的操作数的传送指令

```
MOV  A，Rn          ；(A ) ← (R n )
MOV  A，direct      ；(A) ← (direct)
MOV  A，@Ri         ；(A) ←((Ri))
MOV  A，#vdata      ；(A) ←#data
```

(2) 以工作寄存器为目的操作数的传送指令

```
MOV  Rn，A              ；(R n ) ← (A )
MOV  Rn，direct         ；(Rn) ← (direct)
MOV  Rn，#data          ；(A) ←#data
```

(3) 以直接地址为目的操作数的传送指令

```
MOV  direct，A      ；(direct) ← (A)
MOV  direct，Rn     ；(direct) ← (Rn)
MOV  direct，@Ri    ；(direct) ← (Ri)
MOV  direct1，direct2；(direct1) ← (direct2)
MOV  direct，#data；  (direct) ←#data
```

(4) 以寄存器间接地址为目的操作数的传送指令

```
MOV  @Ri，A         ；((R i )) ← (A )
MOV  @Ri，direct    ；((Ri)) ← (direct)
MOV  @Ri，#data     ；((Ri)) ←#data
```

例 1 把 25H 和 10H 数据分别送到片内 RAM20H 和 25H 单元；把 CAH 送 P1 口；将 P1 口内容送 P2 口；将 RAM20H 单元内容送以 R0 间址的存储单元。

```
MOV  20H，#25H      ；(20H) ←#25H
MOV  25H，#10H      ；(25H) ←#10H
MOV  P1，#0CAH      ；(P1) ←#0CAH
MOV  P2，P1         ；(P2) ←P1
MOV  @R0，20H       ；((R0)) ← (20H)
```

操作数的寻址方式如下表所列。

例 1 操作数寻址方式

指令	目的操作数	源操作数
MOV 20H，#25H	直接寻址	立即寻址
MOV 25H，#10H	直接寻址	立即寻址
MOV P1，#0CAH	直接寻址	立即寻址
MOV P2，P1	直接寻址	直接寻址
MOV @R0，20H	间接寻址	直接寻址

2． 16 位地址指针传送指令

```
MOV  DPTR，#data16      ；(DPTR) ←#data16
```

这条指令的功能是将 16 位常数送入数据指针 DPTR。这是 MCS-51 指令系统中惟一一条 16 位数据传送指令。DPTR 由 DPH 和 DPL 组成。该指令将高 8 位立即数送入 DPH，低 8 位立即数送入 DPL。例如：

```
MOV  DPTR，#1992H          ；(DPH) ←#19H，(DPL) ←#92H
```

也可写成两条 8 位传送指令：

```
MOV  DPH，#19H
MOV  DPL，#92H
```

3．栈操作指令

栈操作指令有进栈 PUSH 和出栈 POP 两条指令：

```
PUSH   direct；(SP) ← (SP) +1；((SP)) ← (direct)
POP  direct；(direct) ← ((SP))；(SP) ← (SP) -1
```

栈操作指令的操作数有两种寻址方式：SP 间接寻址（隐含在指令中）和直接寻址方式。例如：

```
PUSH  B          ；B 为直接寻址方式
PUSH  DPH        ；DPH 为直接寻址方式
```

对于工作寄存器的栈操作，只能使用 Rn 的当前直接地址，而不能用 Rn 名称，因为栈操作指令不能区别 Rn 的当前组别。如果 Rn 工作在组 1 时，R1 的直接地址为 09H，对 R1 的直接地址为 09H，对 R1 的栈操作应写成：PUSH 09H 或 POP 09H。

4．累加器 A 数据交换指令

（1） 字节交换指令

```
XCH  A，Rn          ；(A) ⇆ (Rn)
XCH  A，direct      ；(A) ⇆ (direct)
XCH  A，@Ri         ；(A) ⇆ ((@Ri))
```

该指令的功能是将 A 与源操作数内容互相交换。

例 2 设（A）= 92H，(R0) = 20H，(20H) = 12H，执行指令 XCH A，@R0 后，则 (A) =12H，(20H) = 92H。

（2） 半字节交换指令

```
XCHD  A，@Ri        ；(A)0～3 ⇆ ((Ri))0～3
```

这条指令的功能是将 A 中的低 4 位与 Ri 间址单元内容的低 4 位交换，各自的高 4 位不变。

例 3 设 A 中的内容为 58H，(R0) = 20H，片内 RAM 20H 单元的内容为 65H，执行 XCHD A，@R0 后，则 A 的内容为 55H，片内 RAM 20H 单元内容为 68H。

（3） 累加器 A 高 4 位与低 4 位相互交换指令

```
SWAP  A          ；(A)0～3  (A)4～7
```

例如，设 A 中的内容为 ABH，执行上述指令后，A 中的内容就变为 BAH。

5．累加器 A 与外部 RAM 传送指令

```
MOVX  A，  @Ri        ；(A) ← ((Ri))
MOVX  A，  @DPTR      ；(A) ← ((DPTR))
MOVX  @Ri，A          ；((Ri)) ← (A)
MOVX  @DPTR，A        ；((DPTR)) ← (A)
```

单片机与外部 RAM 进行数据交换时，只能通过累加器 A。采用 R0 和 R1 作间址寄存器时，在给定 P2 情况下，可寻址外 RAM 的 256 个单元；采用 DPTR 作间址寄存器时，可寻址外 RAM 的 64KB 空间。

6．累加器 A 与程序存储器传送指令

```
MOVC  A, @A+DPTR      ;(A) ← ((A) + (DPTR))
MOVC  A, @A+PC        ;(A) ← ((A) + (PC))
```

上述两条适龄以DPTR或PC作为基址寄存器，A中的内容为8位无符号数（A称为变址寄存器），将基址寄存器内容与A中的内容相加，得到一个16位地址，将该地址指出的程序存储器单元的内容送入累加器A。

例4 程序存储器中有一字形表的首地址为0198H，若要调用表中第一字符，则可用下列指令：

```
MOV  DPTR, #0198H     ;设置地址指针
MOV  A, #00H          ;设置变形首址
MOVC  A, @A+DPTR      ;寻找字形码
MOVX  @R0, A          ;字形码送外字形口
```

例5 根据累加器A的内容（0～3）找出由伪令DB所定义的4个字符中的一个。

```
START：INC A          ;(A) ← (A) + 1，单字节指令
       MOVC A,@A+PC   ;(PC) ← (PC) + 1，(A) ← ((A) + (PC))，单字节指令
       RET            ;单字节指令
       DB  29H
       DB  0A2H
       DB  92H
       DB  45H
```

DB是伪指令，功能是将右边的单字节数据存入其左边标号地址单元内。如果DB左边没有标号，则DB伪指令的右边字节数据在DB指令的当前地址连续存放。

该子程序在MOVC指令前面有一条INC A指令，其作用是跳过表格中的RET指令。如果指令MOVC所在地址与表格首地址由若干字节隔开，就需要在累加器A中加上相应的数目。本例中A的取值限定在0～3。在调用上述子程序时，若（A）= 02H，则在执行完这段程序后，A中的内容为92H。

PC的当前值是指读取“MOVC A,@A+PC”后的PC值，即该指令下面的指令所对应的地址。由于“MOVC A,@A+PC”为单字节指令，将该指令所在地址加1，即为PC当前值指向指令RET所在地址。

2 算术运算类指令

MCS-51指令系统具有较强的加、减、乘、除四则运算指令，但只有8位数据运算指令，没有16位数据运算指令。

1．加法类指令

（1） 加法指令

```
ADD   A，Rn           ;(A) ← (A) + (Rn)
ADD   A，direct       ;(A) ← (A) + (direct)
ADD   A，@Ri          ;(A) ← (A) + ((Ri))
ADD   A，#data        ;(A) ← (A) +#data
```

上述指令的功能是将累加器A中的内容与源操作数相加，结果存于A中。

当相加结果的第3位和第7位有进位时，分别将AC和CY置1，否则清0。

无符号数相加后，若CY=1，表示溢出；CY=0，表示无溢出。

对于带符号数相加结果的溢出，取决于第7位和第6位。若第7位有进位而第6位没进位，或第7位没进位而第6位有进位，则OV=1；若第7位和第6位都有进位，或都没进位，则OV=0。OV=1表示两个正数相加而和变为负数，或两个负数相加而和变为正数的错误结果。

例如：(A) =0C2H，(R0) =0A9H，执行ADD　A，R0 指令，过程表示为

```
        1100 0010
+)      1010 1001
-------------------
      10110 1011
```

运算结果 (A) =6BH，(AC) =0，(CY) =1，(0V) =1。若0C2H 和0A9H 是两个无符号数，则结果是正确的；若0C2H 和0A9H 是两个带符号的数，由于有溢出，则表明结果是错误的，因为两个负数相加的结果不可能的到正数。

例6　片内RAM 40H 和41H 单元分别放两个加数，相加结果存放在41H 和40H 单元。

编制程序如下：

```
      MOV  R0，#40H   ；设置地址指针
      MOV  A，  @R0   ；取第一个加数
      INC  R0         ；修改地址指针
      ADD  A，  @R0   ；两数相加
      DEC  R0         ；修改地址指针
      MOV  @R0，A     ；存放和的低字节
      INC  R0         ；修改地址指针
      JC   LOOP       ；有进位则转
      MOV  @R0，#00H；存放和的高字节
      RET
LOOP：MOV@R0，#00H；存放和的高字节
      RET
```

(2)　带进位的加法指令

```
ADDC A，Rn          ；(A) ← (A) + (Rn) + (CY)
ADDC A，direct      ；(A) ← (A) + (direct) + (CY)
ADDC A，@Ri         ；(A) ← (A) + ((Ri)) + (CY)
ADDC A，#data       ；(A) ← (A) +#data+ (CY)
```

上述4 条指令的操作数除了需要加上进位CY 外，其余与ADD 的4 条指令的操作相同。

例7 设A 中的内容为C3H，R0 的内容为AAH，CY=1，执行指令ADDC A，R0 的过程为

```
          1100 0011
          1010 1010
+)              1
-------------------
      1 0110 1110
```

结果：A 中的内容为6EH，(AC) =0，(CY) =1，(0V) =1。

(3)　加1 指令

```
INC  A          ；(A) ← (A) + 1
INC  Rn         ；(Rn) ← (Rn) +1
INC  direct     ；(direct) ← (direct) +1
INC  @Ri        ；((Ri)) ← ((Ri)) + 1
INC  DPTR       ；(DPTR) ← (DPTR) +1
```

INC指令是把指定的单元内容加1，结果仍存原单元中。加1指令除影响奇偶标志P外，运算结果不影响其他标志位。

加1指令为，当目的操作数是P0～P3口时，数据来自端口锁存器（即为SFR)，结果仍写回端

口锁存器。这类以端口为目的操作数的指令被称为“读 - 修改 - 写”指令。

例8 设DPTR的内容为12FEH，执行下列指令：

```
INC  DPTR      ;(DPH)←12H，(DPL)←FFH
INC  DPTR      ;(DPH)←13H，(DPL)←00H
INC  DPTR      ;(DPH)←13H，(DPL)←01H
```

(4) 二 - 十进制调整指令

```
DA   A
```

这是一条专用指令，用于对BCD码十进制加法运算的结果进行修正。MCS-51指令系列系统中没有十进制（BCD）的加法指令，只能借助于二进制加法指令。然而，二进制数的加法用于十进制加法运算时，有时会产生错误结果。例如：

1）6+3=9

```
  0110
+ 0011
------
  1001
```

2）8+7=15

```
  1000
+ 0111
------
  1111
```

3）8+9=17

```
  1000
+ 1001
------
 1 0001
```

其中：1）的运算是正确的，因为9的BCD码就是1001；2）的运算结果是不正确的，因为BCD码没有1111；3）的运算结果也是错误的，因为运算结果是11，而不是17。

出错的原因在于，BCD码是4位的二进制编码，而4位二进制编码共有16个编码，但BCD码只用了其中的10个，剩下的6个没有用。通常把这6个没有用的编码（1010，1011，1100，1101，1110，1111）称为无效码。

在BCD码的加法运算中，凡是结果已进入或跳过无效编码区时，其结果都是错误的。相加的结果大于9，说明已进入无效编码区；相加的结果有进位，说明已跳过无效编码区。但不管是哪一种出错情况，相加结果都比正确值小6。出错是由6个无效编码造成的。

为此，对BCD码运算结果进行“加6”调整，才能得到正确的结果。“加6”的条件是：

1） $(A)_{3\sim0}$ >9 或（AC）=1；

2） $(A)_{7\sim4}$ >9 或（CY）=1。

十进制调整指令不影响溢出标志。

例9 设累加器A的内容为1000 1000B（即 BCD码88），工作寄存器R5的内容为1001 1001B（即 BCD码99），(CY)=1。执行下列指令：

```
ADDC A，R5
DA   A
```

第一条加法指令执行后，A中的内容为0010 0010B；(CY)=1，(AC)=1。然后执行十进制调整指令DA A。因为（CY）=1，(AC)=1，所以高4位和低4位均自动加6调整，即

```
      (A)=1000 1000  BCD88
     (R5)=1001 1001  BCD99
+)                 1
-------------------------
      1 0010 0010    BCD122
调整+) 0110 0110     BCD66
-------------------------
      1 1000 1000    BCD188
```

以上所讲的十进制调整的原理和方法，在具体操作时是通过片内硬件逻辑电路实现的。

例10 设一个加数存于40H和41H单元，另一个加数存于42H和43H单元，和存于40H和41H单元。4位BCD码的加法程序如下（假定相加的结果仍为4位BCD码）：

```
MOV  R0，#40H        ;R0指向加数低字节
```

```
MOV  R1，#42H       ；R1 指向另一个加数低字节
MOV  A，@R0
ADD  A，@R1         ；个位、十位数相加
DA   A              ；十进制调整
MOV  @R0，A         ；存低位和于 40H 单元
INC  R0             ；指针指向百位、千位数
INC  R1
MOV  A，@R0
ADDC A，@R1         ；百位、千位数相加
DA   A
MOV  @R0，A         ；存高位和于 41H 单元
RET
```

2．减法类指令

（1） 带借位减法指令

```
SUBB A，Rn          ；(A) ← (A) — (Rn) — (CY)
SUBB A，direct      ；(A) ← (A) — (direct) — (CY)
SUBB A，@Ri         ；(A) ← (A) — ((Ri)) — (CY)
SUBB A，#data       ；(A) ← (A) — #data — (CY)
```

如果第 7 位借位，则（CY）= 1，否则（CY）= 0；若第 3 位有借位，则（AC）= 1，否则（AC）=0；溢出标志 OV 用于带符号的整数减法，若第 7 位和第 6 位中只有一位有借位，而另一位没有借位，则（OV）=1。（OV）=1 表示一个正数减去一个负数结果为负数，或一个负数减去一个正数为正数的错误结果。当无符号数运算时，溢出标志无意义。

例 11 设累加器 A 中的内容为 0ECH，寄存器 R5 中的内容为 75H，（CY）=1，执行指令 SUBB A，R5，其运算操作过程为

```
      1110 1100
-)    0111 0101
-------------------
      0111 0111
-)           1=(CY)
-------------------
      0111 0110
```

结果：（A）= 76H，（CY）= 0，（AC）= 0 （OV）= 1。

（2） 减 1 指令

```
DEC  A          ；(A) ← (A) - 1
DEC  Rn         ；(Rn) ← (Rn) - 1
DEC  direct     ；(direct) ← (direct) -1
DEC  @Ri        ；((Ri)) ← ((Ri)) - 1
```

减 1 指令的功能是指令单元的内容减 1，结果存于原单元中。除了标志 P 外，本指令不影响其他标志位。

当减 1 指令的目的操作数是 P0～P3 端口时，该指令属于“读 - 修改 - 写”指令，即将端口数据读出，减 1，又送回原端口。

4．乘法和除法指令

（1） 乘法指令

```
MUL  AB          ;(B15～8 A7～0) ← (A) × (A)
```

将A和B的无符号数相乘，16位乘积的低8位存于A，高8位存于B。乘法指令影响3个标志位：(CY)=0；若(B)=0，则(OV)=0，若若(B)≠0，则(OV)=1；P标志仍按A中的内容设置。

(1) 除法指令

```
DIV  AB          ;(A) ←商，(B) ←余数
```

将A中的8位无符号数除以B中的8为2无符号数，商存于A，余数存于B。

DIV操作影响3个标志位：(CY)=0；(B)=0（即非法）时（OV)=1，表明除法没有意义，而其他情况下（OV)=0；P标志仍取决于A的内容。

3 逻辑运算及移位类指令

MCS-51指令系统能对位和字节操作数进行基本的逻辑运算。本节介绍字节操作数的逻辑运算，有关位操作将在后面介绍。

1．逻辑“与”运算指令

```
ANL  A，Rn          ;(A) ← (A) ∧ (Rn)
ANL  A，direct      ;(A) ← (A) ∧ (direct)
ANL  A，@Ri         ;(A) ← (A) ∧ ((Ri))
ANL  A，#data       ;(A) ← (A) ∧#data
ANL  direct，A      ;(direct) ← (direct) ∧ (A)
ANL  direct，#data  ;(direct) ← (direct) ∧#data
```

例12 已知(A)=1010 1101B，(R4)=0110 0101B。执行指令ANL A，R4的过程为

```
     (A) =1010 1101
∧)   (R4) =0110 0101
-----------------------
     (A) =0010 0101
```

2．逻辑“或”运算指令

```
ORL  A，Rn          ;(A) ← (A) ∨ (Rn)
ORL  A，direct      ;(A) ← (A) ∨ (direct)
ORL  A，@Ri         ;(A) ← (A) ∨ ((Ri))
ORL  A，#data       ;(A) ← (A) ∨#data
ORL  direct，A      ;(direct) ← (direct) ∨ (A)
ORL  direct，#data  ;(direct) ← (direct) ∨#data
```

例13 将累加器A的高5位送到P1口的高5位，而P1口的低3位保持不变。程序如下：

```
MOV  R2，A          ;暂存A的内容
ANL  A，#0F8H       ;取A的高5位
ANL  P1，#07H       ;取P1的低3位
ORL  P1，A          ;组合P1口内容
MOV  A，R2          ;恢复A的内容
```

3．逻辑“异或”指令

“异或”操作也是按位进行的。当两个操作数相同时，结果为0；不同时，结果为1。运算符号为⊕。

```
XRL  A，Rn          ;(A) ← (A) ⊕ (Rn)
```

```
XRL  A,direct      ;(A) ← (A) ⊕ (direct)
XRL  A,@Ri         ;(A) ← (A) ⊕ ((Ri))
XRL  A,#data       ;(A) ← (A) ⊕ #data
XRL  direct,A      ;(direct) ← (direct) ⊕ (A)
XRL  direct,#data  ;(direct) ← (direct) ⊕ #data
```

使用“异或”指令可判别两个数是否相等。若相等，则结果为全0。利用本指令可对目的操作数的某些位取反或保留：用1去“异或”的位，则取反；用0去“异或”的位，则保留。

在MCS-51指令系统中的逻辑“与”、“或”、“异或”运算时，当目的操作数为P0～P3端口时，指令属于“读-修改-写”指令。

4．累加器清0及取反指令

```
CLR  A    ;(A) ←#00H
CLR  A    ;(A) ← (Ā)
```

MCS-51指令系统没有“求补”指令，若需要进行“求补”运算，可用“取反加1”运算规则实现。

5．移位指令

MCS-51指令系统的移位操作只对累加器A进行，有左、右小循环和左、右大循环4种：

左小循环 RL A
右小循环 RR A
左大循环 RLC A
右大循环 RRC A

以上4条指令的操作过程，如图2.1所示。

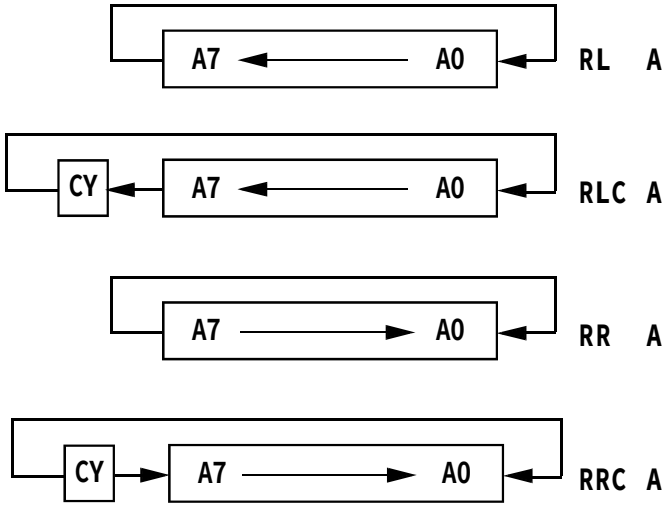


图2.1 移位指令示意图

4 控制转移类类指令

程序的顺序执行是靠PC自动加1实现的。要改变程序的执行顺序，实现分支转向，应通过

强迫改变 PC 值的方法来实现。这就是控制转移类指令的基本功能。

共有两类转移：无条件转移和有条件转移。

1．无条件转移指令

（1） 长转移指令

LJMP addr16 ；(PC) ←addr16

这是一条 3 字节指令，指令执行后把 16 位地址（addr16）送入 PC，从而实现了程序的转移。因为转移范围大，可达 64KB，故称为“长转移”。

（2） 绝对转移指令

LJMP addr11 ；(PC) ← (PC) +2，$(PC)_{10\sim0}$ ←addr11

AJMP 指令提供 11 位地址去替换 PC 的低 11 位地址内容，形成新的 PC 值，即转移目的地址。

AJMP 是一条双字节指令，指令的格式为

第一字节	A10	A9	A8	0	0	0	0	1
第二字节	A7	A6	A5	A4	A3	A2	A1	A0

指令提供的 11 位地址中，A7～A0 在第二字节，A10～A8 则占据第一字节的高 3 位，而指令操作码只占第一字节的低 5 位（00001）。AJMP 指令的功能是构造程序转移目的地址，实现程序的转移。其构造新地址的方法是：以指令提供的 11 位地址（A10～A0）去替换 PC 的低 11 位，形成新的 PC 值，即转移目的地址。但要注意，被替换的 PC 值是 AJMP 指令的地址加 2 的 PC 值，即指向 AJMP 下条指令的 PC 值，称为 PC 当前值。例如，在程序存储器的 2070H 单元存放一条绝对转移指令：

2070H AJMP NEWAD

标号地址 NEWAD 的低 11 位地址为 16AH=001 0110 1010B，构成的指令代码为 216AH，即

0	0	1	0	0	0	0	1
0	1	1	0	1	0	1	0

程序计数器 PC 加 2 的内容为：0010 0000 0111 0010B=2072H，以 11 位绝对地址（16AH）代替 PC 中的低 11 位，形成的转移目的地址为：0010 0001 0110 1010B=216AH。

addr11 是无符号整数，最小值为 000H，最大值为 7FFH，因此绝对转移指令所能转移的最大范围是 2KB。对于“2070H AJMP NEWAD”指令，其转移范围是 2000H～27FFH。

（3） 短转移指令

SJMP rel

SJMP 是相对寻址方式的双字节指令，其中 rel 为相对偏移量。指令的功能是按计算得到转移目的地址，实现程序转移。计算公式为

目的地址 =（PC）+2+rel

其中，PC 称为源地址，即指令“SJMP rel”所在程序单元的地址；偏移量 rel 是一个带符号的 8 位二进制补码数。如果 rel 为正，则向前转移；如果 rel 为负，则向后转移。计算偏移量的公式为

rel= 目的地址 -（源地址 +2）

若相对转移指令是 3 字节指令，则偏移量为

rel= 目的地址 -（源地址 +3）

例 14 在 835AH 处有 SJMP 指令

835AH SJMP 35H

源地址=835AH，rel=35H且为正，则目的地址=835AH+02H+35H=8391H，即程序转移到8391H地址。

例15 在835AH处的SJMP指令为

835AH SJMP 0E7H

rel=0E7H且为负数19H的补码，因此目的地址=835AH+02H-19H=8343H，即程序转移到8343H处 。

若rel=FEH，为负数02H的补码，则目的地址=PC+02-02=PC，即目的地址和指令源地址相同，程序就在该指令上踏步，即

HERE：SJMP HERE或HERE：SJMP $

在MCS-51指令系统中，以$代表指令源地址。

若rel=00H，则目的地址=PC+02H，即目的地址为下一条指令地址。如：

SJMP 00H

NEXT：MOV A，#00H

程序转移到NEXT处。

(4) 变址寻址转移指令

JMP @A+DPTR ；(PC)←(A)+(DPTR)

以DPTR内容为基础（称为基址），A中的内容作为变址。当DPTR固定时，A中赋值不同，可以实现程序的对分支转移。其计算公式为

转移目的地址= (A)+(DPTR)

这种由基址寄存器（DPTR）和变址寄存器（A）共同实现的间址方式，称为变址寻址。

2．条件转移指令

执行条件转移指令时，如指令中规定的条件满足，则进行程序转移；否则，程序顺利执行。

(1) 累加器判零转移指令

JZ rel ；若(A)=0，则(PC)←(PC)+2+rel，即转移

；否则(PC)←(PC)+2，即顺序执行

JNZ rel ；若(A)≠0，则(PC)←(PC)+2+rel，即转移

；否则(PC)←(PC)+2，即顺序执行

上述两条指令均为双字节指令。第一条指令转移条件是(A)=0，第二条指令转移条件是(A)≠0，A中的内容为转移指令前面最后一条指令的执行结果。单片机的程序状态字PSW中没有零标志，只能用累加器的内容为零（非零）作为判断条件。

(2) 比较条件转移指令

比较条件转移指令是把两个操作数进行比较，以是否相等作为条件来控制程序转移。共有4条指令：

```
CJNE A，#data，rel   ；累加器内容与立即数不等则转移，否则顺序执行
CJNE A，direct，rel  ；累加器内容与内RAM中指定单元内容不等则转移，否则顺序执行
CJNE Rn，#data，rel  ；工作寄存器内容与立即数不等则转移，否则顺序执行
CJNE @Ri，#data，rel ；内部RAM中指定单元（间址形式）内容与立即数不等则转移，否则顺序执行
```

上述4条指令是3字节指令，具有数值比较和程序转移两方面功能。

两个操作数比较结果影响CY标志，但不影响操作数中的内容。当左操作数=右操作数时，(CY)=0，程序顺序执行；当左操作数>右操作数时，(CY)=0，程序转移执行；当左操作数<右操作数时，(CY)=1，程序转移执行。

(3) 减1条件转移指令

这是一组把减1与条件转移两种功能结合在一起的指令，共有两条。

● 寄存器减 1 条件转移指令（双字节指令）为

DJNZ Rn，rel ；(Rn) ← (Rn) - 1

若 (Rn) ≠ 0，则 (PC) ← (PC) +2+rel，即程序转移；

若 (Rn) = 0，则 (PC) ← (PC) +2，即程序顺序执行。

● 直接寻址单元减 1 条件转移指令（3 字节指令）为

DJNZ direct，rel；(direct) ← (direct) -1

若 (direct) ≠ 0，则 (PC) ← (PC) + 3 + rel，即程序转移；

若 (direct) = 0，则 (PC) ← (PC) +3，即程序执行。

这两条指令主要用于控制程序循环。如预先把寄存器或内部 RAM 单元赋值循环次数，利用减 1 条件转移指令，以减 1 后是否为 0 作为转移条件，即可实现按次数控制循环。

例 16 将外部 RAM 地址为 1100H～11FFH 的 256 个单元清 0，试编制实现程序。

```
     MOV  R7，#00H  ；置计数初值
     MOV  A， #00H
     MOV  DPTR，#1100H   ；清 0 单元首地址
LOOP：MOV  @DPTR，A      ；清 0
     INC  DPTR
     DJNZ R7，LOOP       ；计数值减 1，不为 0 则循环
     RET                 ；返回
```

3.子程序调用及返回指令

从主程序转向子程序的指令称为子程序调用指令；从子程序返回主程序的指令称为返回指令。

调用指令与钻仪指令的主要区别是转移指令不保存返回地址，而子程序调用指令在转向目的地址的同时，必须保留返回地址（称为断点地址），以便执行返回指令时回到主程序断点的位置。通常采用堆栈技术保存断点地址,这样可以允许多重子程序调用(在子程序中再次调用子程序)。

（1）绝对调用指令（双字节指令）

ACALL addr11 ；(PC) ← (PC) +2，(SP) ← (SP) +1，$(SP) \leftarrow (PC)_{7\sim0}$

；(SP) ← (SP) +1，$(SP) \leftarrow (PC)_{15\sim8}$

；$(PC)_{10\sim0} \leftarrow$ addr11，$(PC)_{15\sim11}$ 保留

该指令格式为

第一字节	A10	A9	A8	1	0	0	0	1
第二字节	A7	A6	A5	A4	A3	A2	A1	A0

指令代码中提供了子程序入口地址的低 11 位。这 11 位地址的 A7～A0 占据指令的第二字节，A10～A8 占据指令的第一字节的高 3 位，低 5 位为操作码。指令的调用范围为 2KB。

为了实现直程序调用，该指令共完成两项操作：

断点保护 断点保护是通过自动方式的堆栈操作实现的。即把加 2 以后的 PC 值（称为 PC 当前值）自动送入栈区保存起来，待子程序返回时再送回 PC。

构造目的地址 目的地址的构造是在 PC 加 2 的基础上，以提供的 11 位地址取代 PC 当前值中的低 11 位，PC 的高 5 位保持不变。

例 17 在程序存储器 8100H 单元处有一条绝对调用指令，确定子程序目的地址。

8100H ACALL 48FH

由于 48FH=0100 1000 1111B，即 addr11 的高 3 位（A10 A9 A8）=100，因此指令第一字节为 91H，第二字节为 8FH，即机器码为 918FH。

PC的当前值PC=8102H=1000 0001 0000 0010指令提供的低11位地址替换PC中的低11位后，形成的目的地址是

1000 0100 1000 1111B=848FH

即被调用的子程序入口地址为848FH。本指令的地址为8100H，不变的高5位是1000B，因此本指令的调用范围是8000H～87FFH(2 KB).

（2）长调用指令（3 字节指令）

```
ACALL addr16    ;(PC) ← (PC) + 3, (SP) ← (SP) + 1, (SP) ← (PC)7～0
                ;(SP) ← (SP) + 1, (SP) ← (PC)15～8
                ;(PC) ← addr16
```

子程序入口地址在指令中直接给出.指令执行后，断点进栈保存,addr16作为子程序入口地址.本指令的调用范围是64KB(0000H～FFFH),使用比较方便,但3字节指令较ACALL指令占有较多的存储空间.

例18 已知下列程序段:

```
    ORG  0100H
    MOV  SP,#60H
    ……
    ORG  0200H
START：LCALL  MIR
    ……
    RET
    MIR  EQU  8100H
    END
```

程序执行结果：(SP)=62H，(61H)=03H，(62H)=02H，(PC)=8100H。

（3）返回指令

●子程序返回指令

RET ;$(PC)_{15\sim8} \leftarrow (SP), (SP) \leftarrow (SP)-1, (PC)_{7\sim0} \leftarrow (SP), (SP) \leftarrow (SP)-1$

●中断返回指令为

RETI ;$(PC)_{15\sim8} \leftarrow (SP), (SP) \leftarrow (SP)-1, (PC)_{7\sim0} \leftarrow (SP), (SP) \leftarrow (SP)-1$

子程序返回和中断返回指令的功能都是从堆栈中取出16位断点地址送PC，使子程序返回主程序。RET指令安排在子程序出口处，RETI指令安排在中断服务程序出口处。

此外，RETI指令还具有清除中断响应时被触发的优先级状态，开放较低级中断和恢复中断逻辑等功能。

例19 已知（SP）=62H，(62H)=07H，(61H)=30H，执行RET指令后，其结果是：(SP)=60H，(PC)=0730H，即CPU从0730H处开始执行程序。

4.空操作指令

```
NOP   ;(PC) ← (PC) + 1
```

空操作指令也是一条控制指令，控制CPU不做任何操作，只消耗一个机器周期的时间。空操作指令是单字节指令，依次执行后PC加1，时间 延续一个机器周期。NOP指令常用于程序的等待或时间的延迟。

5 位操作类指令

位操作（又称位处理）就是以位（bit）为单位进行的运算和操作。位变量也称为布尔变量或开关变量。

MCS-51 指令系统适用位操作的地址空间是片内 RAM 20H～2FH 单元（位地址为 00H～7FH）以及 SFR 区中可寻址的位。

1.位传送指令

```
MOV  C，bit   ；(CY) ← (bit)
MOV  bit，C   ；(bit) ← (CY)
```

bit 表示位地址。位传送就是可寻址的位与 CY 之间的相互传送。由于没有可寻址位之间的直接传送指令，因此位之间无法实现直接传送。如果需要位之间传送，必须以 CY 作中介实现。

例 20 将位地址为 20H 的内容传送到位地址 5AH。编制程序如下：

```
MOV  10H，C   ；暂存 CY 内容
MOV  C，20H   ；20H 位送 CY
MOV  5AH，C   ；CY 送 5AH 位
MOV  C，10H   ；恢复 CY 内容
```

2.位置位和复位指令

```
SETB C    ；(CY) ←1
SETB bit  ；(bit) ←1
CLR  C    ；(CY) ←0
CLR  bit  ；(bit) ←0
```

3.位运算指令

位运算都是逻辑运算，有“与”、“或”、“非”3 种，共 6 条指令

```
ANL  C，bit   ；(CY) ← (CY) ∧ (bit)
ANL  C，/bit  ；(CY) ← (CY) ∧ (/bit)
ORL  C，bit   ；(CY) ← (CY) ∨ (bit)
ORL  C，/bit  ；(CY) ← (CY) ∨ (/bit)
CPL  C    ；(CY) ← (/CY)
CPL  bit  ；(CY) ← (/bit)
```

“/bit”表示位中内容的“非”，运算后 bit 中的内容不取反，保持原内容不变。

在位操作指令中，没有位的“异或”运算，需要时可由上述多条位操作指令实现。此外，通过位逻辑运算，可对各种组合逻辑电路进行模拟，即用软件方法来获得组合电路的逻辑功能。

例 21 用位运算指令实现“异或”操作：

$$D=E \oplus B$$

由于 $D=E \oplus B=\overline{E}B+E\overline{B}$，实现的程序如下：

```
MOV  C，B
ANL  C，/E    ；(CY) ←/E B
MOV  D，C
MOV  C，E
ANL  C，/B    ；(CY) ←E /B
ORL  C，D     ；/EB+E/B
MOV  D，C     ；D=E/B+/EB
```

4.位控制转移指令

位控制转移指令就是以位的状态作为实现程序转移的判断条件。

（1）以C 状态为条件的转移指令（双字节指令）

```
JC    rel   ;若（CY）=1，则（PC）←（PC）+2+rel，即转移
            ;若（CY）=0，则（PC）←（PC）+2，即程序顺序执行
JNC   rel   ;若（CY）=0，则（PC）←（PC）+2+rel，即转移
            ;若（CY）=1，则（PC）←（PC）+2，即程序顺序执行
```

（2）以bit 状态为条件的转移指令（3 字节指令）

```
JB    bit,rel   ;若（bit）=1，则（PC）←（PC）+3+rel，即转移
                ;若（bit）=0，则（PC）←（PC）+3，即程序顺序执行
JNB   bit,rel   ;若（bit）=0，则（PC）←（PC）+3+rel，即转移
                ;若（bit）=1，则（PC）←（PC）+3，即程序顺序执行
JBC   bit,rel   ;若（bit）=1，则（PC）←（PC）+3+rel，即转移，且同时伴随着清bit 位，
即（bit）=0
                ;若（bit）=0，则（PC）←（PC）+3，即程序顺序执行
```

JBC 指令中，若可寻位为1 时，则转移，并同时清该位。当bit 是P0～P3 端口中某一位时，该指令称为：“读-修改-写”指令。

4 汇编语言程序设计

用助记符表示的指令就是计算机的汇编语言，每一条指令就是汇编语言的一条语句。

所谓程序设计就是编写计算机程序。汇编语言程序设计就是使用汇编指令来编写计算机程序。

1 汇编语言的特点及其语句格式

1.汇编语言的特点

汇编语言有以下特点

1） 助记符指令与机器指令一一对应，所以用汇编语言编写的程序占用存储器空间小，运行速度快，可编写出最优化程序。

2）汇编语言是面向计算机的。汇编语言的程序设计人员必须对计算机硬件有相当深入的了解。

3） 汇编语言能直接访问存储器及接口电路，也能处理中断，因此汇编语言程序能直接管理和控制硬件设备。

4）各种计算机都有自己的汇编语言，不同计算机的汇编语言之间不能通用，因此汇编语言缺乏通用性，程序不易移植。

2.汇编语言的语句格式

各种计算机汇编语言的语句格式及语法规则基本相同。MCS-51 汇编语言的语句格式为

[标号]:[操作码][目的操作数],[源操作数];[注释]

其中每部分也称为字段。各部分之间用一个空格或字段分界符分隔。常用的字段分界符有冒号“：”、逗号“,”和分号“；”。

（1） 标号

标号用来说明指令的地址，用于其他语句对该句的访问。标号有以下规定：

1）标号由1～8 个字母和数符组成，字母打头，冒号“：”结束，中间允许数字符号。标号中的字符个数不超过8 个，若超过8 个，则以前面的8 个字符有效，后面的字符不起作用。

2）不能用本汇编语言已经定义的符号作为标号，如指令助记符、伪指令以及寄存器的符号名称符。

3）同一标号在一个程序中只能定义一次，不能重复定义。

4）一条语句可以有标号，也可以没有标号，取决于本程序中有无语句访问这条语句。

（2） 操作码

操作码是汇编语句格式中惟一不能空缺的部分，用于规定语句执行的操作内容。

（3 ）操作数

操作数用于表明指令操作的数据或数据存储地址。操作数可以是空白，也可以是一项、两项，各操作数之间用逗号分开。MCS-51 指令系统的操作数有寄存器、立即数、直接、间接等 7 种寻址方式。

操作数与操作码之间用空格分开。

（4 ）注释

注释不属于语句的功能部分，只是对语句的解释说明，只要用“ ；”号开头，即表明以下为注释的内容。使用注释可使程序文件编制显得更加清楚，帮助程序人员阅读程序。注释可有可无，长度不限，一行不够时可以换行接着写，但换行时要注意在开头使用“ ；”号。

（5 ）分界符

分界符（分隔符）用于把语句格式中的各部分隔开，以便区分，包括空格、冒号、分号或逗号等多种符号。

冒号（：）——用于标号之后。

空格（ ）——用于操作码和操作数之间。

分号（；）——用于注释之前。

逗号（,）——用于操作数之间。

3.汇编语言程序设计的特点

汇编语言程序设计有以下特点：

1） 在程序中要对存取数据的存储器单元地址以及寄存器等作出明确分配。

2） 设计人员对单片机应用系统的硬件结构要有详细了解，以便在程序中熟练使用。

3） 设计程序要尽量采用模块化结构，便于阅读和修改。

4） 在满足工艺要求和便于阅读的基础上，尽量选用字节少，工作进行效率高的指令和结构形式。

2 汇编语言程序的基本结构形式

一般把程序结构分为 3 种形式：顺序结构、分支结构和循环结构。

1.顺序结构

顺序结构是最简单的程序结构，在顺序程序中无分支、循环和调用子程序，程序是逐条顺序执行的。

例 22 被加数存于片内 RAM 32H，31H 和 30H；加数存于片内 RAM 35H，34H 和 H；相加之和存于片内 RAM 32H，31H 和 30H；进位存于 00H 单元，试编制程序。

```
START:  MOV     R0，#30H      ；被加数低字节地址
        MOV     R1，#33H      ；加数低字节地址
        MOV     A，@R0
        ADD     A，@R1        ；低字节相加
        MOV     @R0，A        ；存低字节相加结果
        INC     R0
        INC     R1
        MOV     A，@R0
        ADDC    A，@R1        ；中间字节相加
        MOV     @R0，A        ；存中间字节相加结果
```

```
        INC     R0
        INC     R1
        MOV     A，@R0
        ADDC    A，@R1          ；高字节相加
        MOV     @R0，A          ；存高字节相加结果
        CLR     A
        ADDC    A，#00H
        MOV     00H，A          ；存进位
        RET
```

2.分支结构

分支结构是通过转移指令实现的。根据程序的功能特点，又可分为单分支程序、多分支程序等。

例23 假定在外RAM 2000H，2001H，2002H的3个连续单元中，2000H和2001H单元存放着两个无符号数，要求找出其中较大者并存于2002H单元。其程序如下：

```
        ORG   0100H
START：  CLR   C
        MOV   DPTR，#2000H      ；设置数据指针
        MOVX  A，@DPTR          ；取第一个数
        MOV   R2，A             ；暂存于R2
        INC   DPTR              ；数据指针加1
        MOVX  A，@DPTR          ；取第二个数
        SUBB  A，R2             ；两数比较
        JNC   LOOP1             ；第二个数大则转LOOP1
        XCH   A，R2             ；第一个数大则交换
LOOP0：  INC   DPTR
        MOVX  @DPTR， A         ；存大数
        RET
LOOP1：  MOVX  A，@DPTR
        SJMP  LOOP0
```

3.循环结构

循环是为了重复执行一个程序段。在汇编语言中可以通过条件判断循环是否结束。

例24 将内部RAM 20H为起始地址的数据串（最大长度为32字节）传送到外部RAM 2000H为首地址的区域，直到发现“$”字符的ASCII码为止。其程序如下：

```
        MOV   R0，#20H          ；内RAM数据串首地址
        MOV   DPTR，#2000H      ；外RAM数据串首地址
        MOV   R7，#20H          ；最大数据串长度
LOOP0：  MOV   A，R0
        XRL   A，#24H           ；判断是否为“$”字符
        JZ    LOOP1
        MOV   A，@R0
        MOVX  @DPTR， A
```

```
            INC     R0
            INC     DPTR
            DJNZ    R7，LOOP0
LOOP1：     RET
```

5 汇编语言的伪指令与汇编

用指令系统编写的汇编语言程序称为源程序，必须将其翻译成机器码（称为目标程序），单片机方可执行。源程序转换成目标程序的过程是由通用计算机执行一种特定的翻译程序（称为汇编程序）自动完成的。这个翻译过程称为汇编。

1 汇编语言的伪指令

源程序中应有向汇编程序发出指示信息，告诉汇编程序如何完成汇编工作的控制命令，称之为伪指令。伪指令具有控制汇编程序的输入 / 输出、定义数据和符号、条件汇编和分配存储空间等功能。不同的汇编语言的伪指令也有所不同，但一些基本的东西却是相同的。

伪指令是由程序员发给汇编程序的命令，也称为汇编命令或汇编程序控制指令。只有在汇编前的源程序中才有伪指令，汇编后得到的目标程序（机器码）中没有伪指令相应的机器代码。

下面介绍MCS-51汇编语言程序中常见的伪指令。

1.ORG 汇编起始地址命令

在汇编语言源程序的开始，通常都要用一条ORG（Origin）伪指令规定程序的起始地址。命令格式为

[标号]：ORG [地址]

其中：[标号]是选择项，根据需要选用；[地址]项通常为16绝对地址，但也可以使用标号或表达式。例如：

```
            ORG   8000H
START：     MOV   A，#00H
            ……
```

即规定标号START代表地址8000H，目标程序的第一条指令从8000H开始。

2.END 汇编终止命令

END（END of assembly）是汇编语言源程序的结束标志，在整个源程序中只能有一条END命令，且位于程序的最后。如果END命令出现在中间，则其后面的源程序汇编时将不予处理。命令格式为

[标号]：END

命令中的[标号]是选择项。这个标号应是源程序第一条指令的符号地址。例如：

```
            ORG   8100H
START：     MOV   A，#00H
            MOV   R7，#10H
            MOV   R0，#20H
LOOP：      MOV   @R0，A
            INC   R0
            DJNZ  R7，LOOP
            RET
            END
```

3.EQU 赋值命令

EQU（Equate）命令用于给标号赋值。赋值以后，其符号值在整个程序中有效。命令格式为

[字符名称] EQU [赋值项]

其中,[赋值项]可以是常数、地址、标号或表达式。其值为 8 位或 16 位而进制数。赋值以后的字符名称既可以作立即数使用，也可以作地址使用。例如：

```
          ORG       6000H
START：   MOV       R7，#05H
LOOP：    LCALL     DELAY
          DJNZ      R7，LOOP
          RET
          DELAY     EQU  1880H
          END
```

4.DB 定义字节命令

DB（Define Byte）命令用于从指定的地址开始，在程序存储器的连续单元中定义字节数据。命令格式为

[标号]：DB [8 位数据表]

字节数据可以是一字节常数或字符，或用逗号分开的字符串，或用引号括起来的字符串。例如：

DB“How are you？”

把字符串中的字符按 ASC II 码存于连续的 ROM 单元中。

常使用本命令存放数据表格，例如存放数码管显示的十六进制数的形码，可使用多条 DB 命令定义：

```
DB   3FH，06H，5BH，4FH
DB   66H，6DH，7DH，07H
DB   7FH，6FH，77H，7CH
DB   0C0H，0F9H，0A4H，0B0H
```

5.DW 定义字命令

DW（Define Word）命令用于从指定地址开始，在程序存储器的连续单元中定义 16 位的数据字。命令格式为

[标号]：DW [16位数据表]

存放时，数据的高 8 位在前（低地址），低 8 位在后（高地址）。例如：

```
DW  “A A ”              ；存入 41H，41H
DW  “A  ”               ；存入 00H，41H
DW  “A B C ”            ；不合法，因超过两个字节
DW   100H，1ACH，814    ；按顺序存入 01H，00H，01H，ACH，FCH，DCH
```

DB 和 DW 定义的数据表，数的个数不得超过 80 个。如果数据的数目较多时，可使用多个定义命令。在 MCS-51 程序设计中，常以 DB 定义数据，以 DW 是定义地址。

6.DS 定义存储区命令

DS（Define Storage）命令用于从指定地址开始，保留指定树木的字节单元作为存储区，供程序运行使用。汇编时，这些单元不赋值。命令格式为

[标号]：DS [16位数据表]

例如：

```
ADDTAL ： DS   20
```

从标号 ADDTBL 带表的地址开始，保留 20 个连续的地址单元。又例如：

```
ORG  8100H
DS   08H
```

从 8100H 地址开始，保留 8 个连续的地址单元。

注意 DB, DW 和 DS 命令只能对程序存储器使用，而不能对数据存储器使用。

6.BIT 位定义命令

本命令用于给字符名称赋以位地址。命令格式为

[字符名称] BIT [位地址]

其中[位地址]可以是绝对地址，也可以是符号地址（即位符号名称）。例如：

```
AQ   BIT  P1.0
```

把 P1.0 的位地址赋给变量 AQ。在其后的编程中，AQ 就可以作为位地址（P1.0）使用。

2 汇编语言的汇编

将用助记符编写的源程序转换成机器码的过程称为汇编。汇编分为手工汇编和机器汇编。

对于简单的应用程序，可以通过查表翻译指令的方法将源程序翻译成机器码，称之为手工汇编 。

由于手工汇编是按绝对地址进行定位，所以手工汇编时要根据转移的目标地址计算转移指令的偏移量，而且容易出错。此外，对于汇编后的目标程序，如须增加、删除和修改指令，就会引起以后各指令地址的改变，转移指令的偏移量也要重新计算。因此，手工汇编不是理想的方法，通常只用于小的程序。

编写完单片机的源程序之后，由于单片机本身软硬件资源所限，无法由单片机本身自动汇编（机器汇编），只能借助于通用计算机对源程序进行汇编。

使用一种计算机的汇编程序去汇编另一种计算机源程序，具体说就是运行汇编程序进行汇编的是一种计算机，而运行汇编得到目标程序的则是另一种计算机。这种使用一种计算机的汇编程序去汇编另一种计算机的源程序的汇编过程，被称为交叉汇编。单片机的机器汇编就是交叉汇编。

在交叉汇编之前，一般还要借助于通用计算机进行单片机的程序设计。通常使用编辑软件进行源程序的编辑，以形成一个由汇编指令和伪指令组成的源程序文件。这个过程被称为机器编辑。

交叉汇编之后，再使用串行通信方法，把汇编得到的目标程序传送到单片机，进行程序的调试和运行。

“机器编辑→交叉编辑→串行发送”，这3个过程构成了单片机软件设计的3个基本步骤。

源程序编写如下：

```
          ORG  8000H
START：   MOV  R0，#20H
          MOV  R7，#07H
          CLR  F0
LOOP：    MOV  A ，  @R0
          MOV  2BH，A
          INC  R0
          MOV  2AH,@R0
          CLR  C
          SUBB A ，  @R0
```

```
            JC    NEXT
            MOV   @R0，2BH
            DEC   R0
            MOV   @R0，2AH
            INC   R0
            SETB  F0
NEXT：      DJNZ  R7，LOOP
            JB    F0，START
HERE：      SJMP  $
            END
```

手工汇编结果如下表所列。

手工汇编结果

目标程序部分		源程序部分		
地址	机器码	标号	助记符指令	备注
8000	7820	START：	MOV R0，#20H	
8002	7F07		MOV R7，#07H	
8004	C2D5		CLR F0	
8006	E6	LOOP：	MOV A，@R0	
8007	F52B		MOV 2BH，A	
8009	08		INC R0	
800A	862A		MOV 2AH，@R0	
800C	C3		CLR C	
800D	96		SUBB A，@R0	
800E	4008		JC NEXT	偏移1
8010	A62B		MOV @R0，2BH	
8012	18		DEC R0	
8013	A62A		MOV @R0，2AH	
8015	08		INC R0	
8016	D2D5		SETB F0	
8018	DFEC	NEXT：	DJNZ R7，LOOP	偏移2
801A	20D5E3		JB F0，START	偏移3
801D	80FE	HERE：	SJMP $	偏移4

偏移 1 的计算：

rel1= 目的地址 -（源地址 +2）=8018H-（800EH+2）=08H

偏移 2 的计算：

rel2= 目的地址 -（源地址 +2）=8006H-（8018H+2）=-14H

（-14H）补码 =ECH

偏移 3 的计算：

rel3= 目的地址 -（源地址 +2）=8000H-（801AH+3）=-1DH

(-1DH）补码 =E3H

偏移 4 的计算：

rel4= 目的地址 -（源地址 +2）=801DH-（801DH+2）=-2H

(-2H）补码 =FEH

6 汇编语言程序设计举例

1 算术运算程序

1．加、减运算程序

（1）不带符号的多字节数加法

例 25 设有两个 4 字节的二进制数，分别存放在以 30H 和 50H 为起始地址的单元中（先存放低字节)。求这两个数的和，并将和存放在以 30H 为起始地址的单元中，试编制程序。

程序如下：

```
          ORG   2000H
JAZ：     MOV       R0，#30H        ；指向加数最低位
          MOV       R1，#50H        ；另一加数最低位
          MOV       R2，#04H        ；字节个数存于 R2
          LCALL     JAFA            ；调用加法子程序
          JC        OVER            ；有进位则转出
          MOV       34H,#00H        ；无进位清最低字节单元
          SJMP      HERE
OVER：    MOV       34H,#01H        ；最高字节单元为 01H
HERE：    SJMP      HERE
          ORG       1000H
JAFA：    CLR       C               ；C 清 0
JAADD：   MOV       A ，  @R0       ；取出加数一个字节
          ADDC      A ，  @R1       ；加上另一个数的一个字节
          MOV       @R0,A           ；保存和
          INC       R0              ；修改加数的地址
          INC       R1
          DJNZ      R2，JAADD       ；没加完则继续
          RET
```

（2）不带符号的两个多字节数减法

例 26 设有两个 N 字节无符号数分别存于内 RAM 单元中，低字节在前，高字节在后。由 R0 指定被减数单元地址，由 R1 指定减数单元地址，要求差值存放在原被减数单元中，假定最高字节没有错位。

程序如下：

```
          CLR  C
          MOV  R7，#N          ；设定 N 字节
LOOP：    MOV  A，@R0          ；从低位取被减数字节
```

```
        SUBB    A，@R0          ；两位数减
        MOV     @R0，A          ；保存差
        INC     R0
        INC     R1
        DJNZ    R7，LOOP
        RET
```

（3 ）带符号数加、减运算

对于带符号数的减法运算，只要将减数的符号位取反，就可把减法运算按加法运算处理。

对于带符号数的加法运算，首先要进行两数符号的判定。如果两数符号相同，应进行两数相加，并以被加数符号为结果符号。

如果两数符号不同，应进行两数相减。如果相减的差为正，则差即为最后结果，并以被减数符号为结果符号；如果相减的差为负，则应将其差值取补，并把被减数的符号取反作为结果符号。

例 27 假定 20H 和 21H 以及 22H 和 23H 分别存放两个 16 位的带符号二进制数，其中 20H 和 22H 的最高位为两数的符号位。请编写带符号双字节二进制数的加减法程序，以 BUSB 为减法程序入口，以 BADD 为加法程序入口，以内 RAM 24H 和 25H 保存运算结果。

程序如下：

```
BUSB：  MOV     A，22H          ；取减数高字节
        CPL     ACC.7
        MOV     22H，A          ；减数符号位取反进行加法
BADD：  MOV     A，20H          ；取被加数
        MOV     C，ACC.7
        MOV     F0，C           ；被加数符号位存于 F0
        XRL     A，22H          ；两数高字节“异或”
        MOV     C，ACC.7        ；两数同号（CY）=0，异号（CY）=1
        MOV     A，20H          ；取被加数
        CPL     ACC.7           ；被加数高字节符号位清 0
        MOV     20H，A          ；取其数值部分
        MOV     A，22H          ；取加数
        CLR     ACC.7           ；加数高字节符号位清 0
        MOV     22H，A          ；取其数值部分
        JC      JIAN            ；两数异号转 JIAN
JIA：   MOV     A，21H          ；两数同号进行加法
        ADD     A，23H          ；低字节相加
        MOV     25H，A          ；保存低字节和
        MOV     A，20H
        ADDC    A，22H          ；高字节相加
        MOV     24H，A          ；保存高字节和
        JB      ACC.7，QAZ      ；符号位为 1 转溢出处理
QWE：   MOV     C，F0           ；结果符号处理
        MOV     ACC.7，C
        MOV     24H，A
        RET
```

```
JIAN：  MOV     A，21H          ；两数异号进行减法
        CLR     C
        SUBB    A，23H          ；低字节相减
        MOV     25H，A          ；保存差
        MOV     A，20H
        SUBB    A，22H          ；高字节相减
        MOV     24H，A          ；保存差
        JNB     ACC.7，QWE      ；没借位转QWE
BMP：   MOV     A，25H          ；有借位，差值取补
        CPL     A
        ADD     A，#01H
        MOV     25H，A
        MOV     A，24H
        CPL     A
        ADDC    A，#00H
        MOV     24H，A
        CPL     F0              ；符号位取反
        SJMP    QWE
QAZ：   ……                      ；溢出处理（从省略）
```

2.乘法运算

对于单字节乘法运算，使用一条乘法指令 MUL AB 即可；对于多字节的乘法就必须通过程序实现。

例 28 假设被乘数存放于 R6 和 R7 中，乘数存放于 R4 和 R5 中，乘积存放于 40H，41H，42H 和 43H 中，低字节在前，双字节乘法结果最多为 4 字节。

双字节乘法按一般竖式相乘原理，设 R6 × R4=H64，L64；R7 × R4=H74，L74；R5 × R6=H56，L56；R7 × R5=H75，L75。其中，H 表示高字，L 表示低字节。竖式乘法过程表示为

```
                R7      R6
  ×  )          R5      R4
 ------------------------------------------
                        H64     L64 ◄──  R6 × R4
                H74     L74 ◄─────────── R7 × R4
                H56     L56 ◄─────────── R5 × R6
          H75   L75 ◄─────────────────── R7 × R5
 ------------------------------------------
          (43H)  (42H)  (41H)  (40H)
```

具体程序如下：

```
        ORG     0020H
MUL16:  MOV     R0,#40H         ；积地址指针
        MOV     A，R6
        MOV     B，R4
        MUL     AB              ；R6 × R4=H64，L64
        MOV     @R0，A          ；L64 → (40H)
        MOV     R3，B           ；H64 → R3
        MOV     A，R7
```

```
        MOV     B，R4
        MUL     AB              ；R7 × R4=H74，L74
        ADD     A，R3           ；L74+H64 → R3
        MOV     R3，A
        MOV     A，#00H         ；H74+CY → R2
        MOV     R2，A
        MOV     A，R6
        MOV     B，R5
        MUL     AB              ；R5 × R6=H56，L56
        ADD     A，R3           ；L56+L74+H64 → A
        INC     R0
        MOV     @R0，A          ；A → （41H）
        MOV     R1，#00H
        MOV     A，R2
        ADDC    A，B            ；H56+R2+CY → R2
        MOV     R2，A
        JNC     NEXT
        INC     R1
NEXT：  MOV     A，R7
        MOV     B，R5
        MUL     AB              ；R7 × R5=H75，L75
        ADD     A，R2           ；L75+R2 → A
        INC     R0
        MOV     @R0，A          ；A → （42H）
        MOV     A，B
        ADDC    A，R1           ；H75+R1+CY → A
        INC     R0
        MOV     @R0，A
        RET
```

3.除法运算

对于单字节除法运算使用一条除法指令 DIV AB 即可；但对于多字节的除法就必须通过程序实现。

多字节除法的程序设计常采用“恢复余数法”，其设计思想是做减法。

仿照手工算法进行除法 设被除数为 100011，除数为 101，求 100011B ÷ 101B=？

```
                  000111
除数：101 ) 100011   ……被除数
           -) 101    ……2^-1 × 除数
              111    ……余数
           -)  101   ……2^-2 × 余数
               101   ……余数
           -)  101   ……2^-3 × 除数
                0
```

计算机除法运算采用“左移被除数相除法”。做除法前先将余数单元清0，在CY=0条件下，执行左循环移位，将被除数最高位移入余数单元最低位，被除数最低位变为0，然后用余数减去除数。若够减，则此时被除数移位单元最低位置1，即商为1，同时用差取代余数；若不够减，则此时的被除数移位单元仍为0，即商为0。这样重复移位，做减法，直到被除数全部左移入余数单元。最后被除数移位单元变成了商数单元，余数单元存有余数。

设被除数为1011，除数为0101，余数单元全清0，下面是采用左移位除法过程。

第一次移位：余数单元=0001，被除数移位单元=0110，余数单元减去除数，不够减，继续左移。

第二次移位：余数单元=0010，被除数移位单元=1100，余数单元减去除数，不够减，继续左移。

第三次移位：余数单元=0101，被除数移位单元=1000，余数单元减去除数，够减且差为0000，用此时的差值取代原来余数，并将被除数移位单元最低位置1，即余数单元=0000，被除数移位单元=1001，继续左移。

第四次移位：余数单元=0001，被除数移位单元=0010，移位完成，最后结果是：商为0010，余数为0001。

例29 编写一个16位÷16位除法程序。假设被除数存于40H和41H中，除数存于44H和45H中，商存于40H和41H中，余数存于42H和43H中。低字节在前，48H和49H为暂存单元。

程序如下：

```
            ORG     0059H
DIV16:      MOV     R0,#40H         ；被除数为0则退出
            MOV     A，@R0
            JNZ     LOP0
            INC     R0
            MOV     A，@R0
            JNZ     LOP0
            CLR     A
            MOV     42H，A
            MOV     43H，A
            RET
LOP0：      MOV     R0,#44H         ；除数为0则退出
            MOV     A，@R0
            JNZ     LOP1
            INC     R0
            MOV     A，@R0
            JNZ     LOP1
            RET
LOP1：      CLR     A
            MOV     42H，A          ；清余数单元42H和43H
            MOV     43H，A
            MOV     R2,#10H         ；置移位次数
LOP2：      CLR     C               ；CY=0
            MOV     R3，#04H
```

```
        MOV     R0，#40H        ；被除数地址指针
LOP3：  MOV     A，@R0          ；余数单元，被除数单元左移一次
        RLC     A
        MOV     @R0，A
        INC     R0
        DJNZ    R3，LOP3
        MOV     R0，#42H        ；余数单元减除数
        MOV     R1，#44H
        MOV     A，@R0
        CLR     C
        SUBB    A，@R1
        MOV     48H，A          ；暂存差的低字节
        INC     R0
        INC     R1
        MOV     A，@R0
        SUBB    A，@R1
        MOV     49H，A          ；暂存差的高字节
        JC      LOP4            ；不够减继续左移
        MOV     R0，#42H        ；够减时差值取代原余数
        MOV     R1，#48H
        MOV     A，@R1
        MOV     @R0,A
        INC     R0
        INC     R1
        MOV     A，@R1
        MOV     @R0,A
        MOV     A，40H
        INC     A               ；够减时被除数单元加 1
        MOV     40H，A
LOP4：  DJNZ    R2，LOP2        ；移位次数不到，继续
        RET
        END
```

2 数制转换程序

1.十六进制数转换成 ASC II 码

例 30 在片内 RAM 20H 单元中存有 2 位十六进制数，将其转换成 ASC II 码，并存于 21H 和 22H 两个单元中。

程序如下：

```
        MOV     SP，#3FH
MAIN：  PUSH    20H             ；十六进制数进栈
        LCALL   HASC            ；调用转换子程序
        POP     21H             ；第一位转换结果送 21H 单元
        MOV     A，20H          ；再取原十六进制数
```

```
            SWAP    A               ；高低半字节交换
            PUSH    ACC             ；交换后的十六进制数进栈
            LCALL   HASC            ；调用转换子程序
            POP     22H             ；第二位转换结果送 22H 单元
            RET
HASC：      DEC     SP              ；跨过断点保护对象
            DEC     SP
            POP     ACC             ；弹出转换数据
            ANL     A，#0FH         ；屏蔽高 4 位
            ADD     A，#07H         ；修改变址寄存器内容
            MOVC    A，@A+PC        ；查表
            PUSH    ACC             ；查表结果进栈
            INC     SP              ；修改堆栈指针回到断点保护内容
            INC     SP
            RET
ASCTAB：    D B “0，1，2，3，4，5，6，7”；ASC II码表
            D B “8，9，A，B，C，D，E，F”
```

2.ASC II码转换成十六进制数

例 31 将外部 RAM 30H～3FH 单元中的 ASC II码依次转换为十六进制数，并存入内 RAM 60H～67H单元中。

程序如下：

```
MAIN：      MOV     R0，#30H        ；设置 ASC II码地址指针
            MOV     R1，#60H        ；设置十六进制数地址指针
            MOV     R7，#08H        ；需拼装的十六进制数的字节数
LOOPA：     LCALL   TRAN            ；调用转换子程序
            SWAP    A               ；A 中高低 4 位交换
            MOV     @R1，A          ；存于内部 RAM
            INC     R0
            LCALL   TRAN            ；调用转换子程序
            XCHD    A，@R1          ；十六进制数拼装
            INC     R0
            INC     R1
            DJNZ    R7，LOOPA
            RET
TRAN：      CLR     C
            MOVX    A，@R0          ；取 ASC II码
            SUBB    A，#30H         ；减去 30H
            CJNE    A，#0AH，LOOPB
            SJMP    LOOPC
LOOPB：     JC      DONE
LOOPC：     SUBB    A，#07H
DONE：      RET
```

3 定时程序

在单片机应用系统中，定时功能除可使用定时器 / 计数器实现外，还可使用定时程序完成。定时程序是典型的循环程序,是通过执行一个具有固定延迟时间的循环体来实现延时的。

1.单循环定时程序

```
        MOV     R7，#TIME
LOOP：  NOP
        NOP
        DJNZ    R7，LOOP
        RET
```

NOP 指令的机器周期为 1，DJNZ 指令的机器周期为 2，则一次循环共 4 个机器周期。如果单片机的晶振频率为 6MHz，则一个机器周期是 2 μs，因此一次循环的延迟时间为 8 μs。定时程序的总延迟时间是循环程序段的整数倍，该程序的延迟时间为 8 × TIME（μs)。这个程序的最长延时时间为 256 × 8=2048 μs。

2.较长时间的定时程序

为了加长定时时间，通常采用多重循环的方法。如下面的双重循环的定时程序，最长可延时 262 914 个机器周期，即 525 828 μs 或大约 526ms（晶振频率为 6MHz)。

```
        MOV     R7，#TIME1      ;1 个机器周期
LOOP1： MOV     R6，#TIME2      ;1 个机器周期
        NOP                     ;1 个机器周期
        NOP                     ;1 个机器周期
        DJNZ    R6，LOOP2       ;2 个机器周期
        DJNZ    R7，LOOP1       ;2 个机器周期
        RET                     ;2 个机器周期
```

最长定时时间计算公式为

（256 × 4+2+1）× 256 × 2+4=525 828 μs

3.以一个基本的延时程序满足不同的定时要求

如果系统中有多个定时需要，可以先设计一个基本的延时程序，使其延迟时间为各定时时间的最大公约数，然后以此基本程序作为子程序，通过调用的方法实现所需要的不同定时。例如：要求的定时时间分别为 5s，10s 和 20s，设计一个 1s 延时子程序 DELAY，则不同定时的调用情况表示如下（晶振频率为 6MHz)：

```
        MOV     R5，#05H        ;延时 5s
LOOP1： LCALL   DELAY
        DJNZ    R5，LOOP1
        RET
        MOV     R5，#0AH        ;延时 10s
LOOP2： LCALL   DELAY
        DJNZ    R5，LOOP2
        RET
        MOV     R5，#14H        ;延时 20s
LOOP3： LCALL   DELAY
        DJNZ    R5，LOOP3
```

```
            RET
DELAY：     MOV     R7，#0FAH
LOOPA：     MOV     R6，#0FAH
LOOPB：     NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            DJNZ    R6，LOOPB
            DJNZ    R7，LOOPA
            RET
```

延时时间为

$$(250 \times 8+2+1) \times 250 \times 2+4=1\ 001\ 504\ \mu s \approx 1s$$

4.查表程序

预先把数据形式存放在程序存储器中，然后使用程序读出。这种能读出表格数据的程序被称为查表程序。MCS-51指令系统准备了专用的查表指令：

```
MOVC A，@A+DPTR
MOVC A，@A+PC
```

这两个MOVC指令的功能是完全相同的。它们在不改变DPTR和PC的状态下，只根据A的内容就可以取出表格中的数据。但这两条指令在具体使用上也存在差异。前一条指令的基址寄存器DPTR能提供16位基址，而且还能在使用前给DPTR赋值，查表空间可达64KB。后一条指令是以PC作为基址寄存器，虽然也能提供16位地址，但PC不能被赋值，所以其基址值是固定的。由于A的内容为8位无符号数，因次只能在当前指令下面的256个地址单元内进行查表，即数据只能放在该指令后面的256个地址单元之内，而且表格只能被程序段所使用。

例32 设有一个巡回检测报警装置，需要对16路输入值进行比较，当每一路输入值等于或超过该路的报警值时，实现报警。下面根据这一要求，编制一个查表程序。

设Xi为路数，查表是Xi按0，1，2，…，15（i=15）取数，表中报警值是2字节数，依Xi顺序列成表格放在TAB中。进入查表程序之前，路数Xi放在R2中，其输入值存于R0和R1当中，查表结果若许报警，将P1.0置1，否则清0。

```
            ORG     1000H
TB1:        MOV     A,R2            ；路数Xi → R2 → A
            ADD     A,R2            ；R2+R2 → A
            MOV     R2,A            ；A → R2
            MOV     DPTR,#TAB       ；取数据表首地址
            MOVC    A，@A+DPTR      ；取出高字节
            MOV     R4,A            ；高字节→ R4
            INC     R2              ；地址指向低字节
            MOV     A,R2
            MOVC    A，@A+DPTR      ；取出低字节
            MOV     R3,A            ；低字节→ R3
            CLR     C
```

```
        MOV     A,R0            ；当前输入值与报警值比较
        SUBB    A，R3           ；低字节相减
        MOV     A,R1
        SUBB    A，R4           ；高字节相减
        JNC     LOOP
        CLR     P1.0            ；输入值＜报警值
        RET                     ；返回
LOOP：  SETB    P1.0            ；输入值≥报警值
        RET                     ；返回
        ORG     2000H
TAB：   DW      05F0H，0E89H，0A69H，1EAAH
        DW      0D9BH，7F93H，0373H，26D7H
        DW      2710H，9E3FH，1A66H，22E3H
        DW      1174H，16EFH，33E4H，6CA0H
        END
```

5 数据极值查找程序

极值查找就是在指定的数据区中挑出最大值或最小值。

例 33 片内 RAM 20H 单元开始存放 8 个无符号 8 位二进制数，找出其中的最大值。极值查找操作的主要内容是进行数值大小的比较。假定在比较过程中，以 A 存放大数，与之逐个比较的另一个数放在 3AH 单元中。比较结束后，把查找到的最大数送到 3BH 单元中。

程序如下：

```
        MOV     R0，#20H        ；数据区首地址
        MOV     R7，#08H        ；数据区长度
        MOV     A，@R0          ；读第一个数
LOOP：  INC     R0
        MOV     3AH，@R0        ；读下一个数
        CJNE    A，3AH，CHK     ；数值比较
        SJMP    LOOP1
CHK：   JNC     LOOP1           ；A 值大则转
        MOV     A，@R0          ；大数送 A
LOOP1： DJNZ    R7，LOOP        ；继续比较
        MOV     3BH，A
        RET
```