STC12C2052AD 系列单片机器件手册

--- 1 个时钟 / 机器周期 8051
--- 无法解密
--- 低功耗,超低价
--- 高速，高可靠
--- 强抗静电，强抗干扰

STC12C0552, STC12C0552AD
STC12C1052, STC12C1052AD
STC12C2052, STC12C2052AD
STC12C3052, STC12C3052AD
STC12C4052, STC12C4052AD
STC12C5052, STC12C5052AD

STC12LE0552, STC12LE0552AD
STC12LE1052, STC12LE1052AD
STC12LE2052, STC12LE2052AD
STC12LE3052, STC12LE3052AD
STC12LE4052, STC12LE4052AD
STC12LE5052, STC12LE5052AD

技术支援：宏晶科技（深圳）
www.MCU-Memory.com support@MCU-Memory.com
Update date: 2005-11-19 (草案，请随时注意更新)

领导业界革命 覆盖市场需求

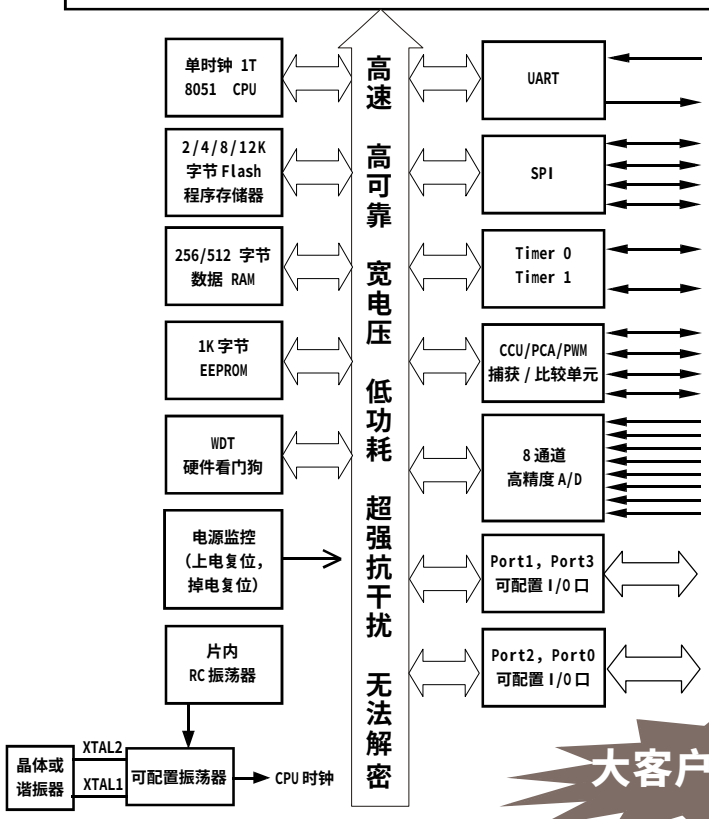
STC 12C2052AD 系列 1T 8051 单片机

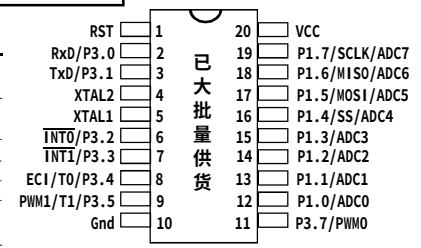
——1 个时钟 / 机器周期，高速、高可靠，相当于普通 8051： 0～420MHz

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,全力推出“1 个时钟 / 机器周期”的单片机,全面提升 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。

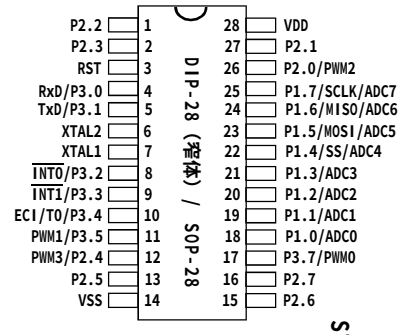
DIP-20，SOP-20 超小封装 8051 单片机

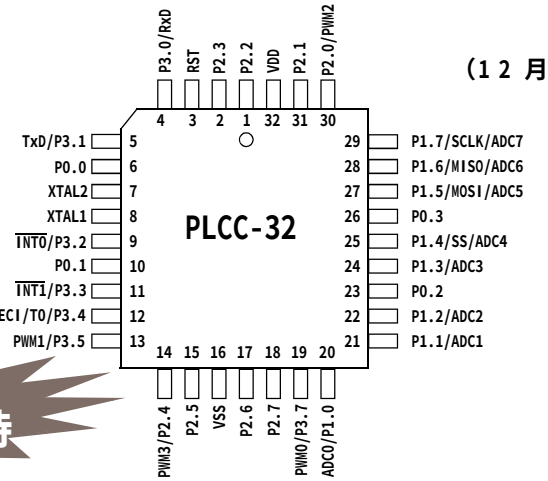
1 个时钟 / 机器周期，超小封装 8051 单片机





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





STC12C5410 系列（12 月供货），28/32Pin （12 月供货）

大客户特价支持

STC12C2052/STC12C5410 系列主要性能：

- 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
- 2K/4K/8K/10K/12K 片内 Flash 程序存储器，擦写次数 10 万次以上
- 256/512 字节片内 RAM 数据存储器
- 芯片内 E²PROM 功能
- ISP / IAP，在系统可编程
- ~~1 个模拟比较器~~（无）
- 8 通道高精度 8 位 ADC，STC12C5410AD 系列为 10 位精度 ADC
- 2 通道捕获 / 比较单元（CCU/PCA/PWM），STC12C5410AD 系列为 4 通道
- 2 个硬件 16 位定时器，兼容普通 8051 的定时器
- 硬件看门狗（WDT）
- 高速 SPI 通信端口
- 全双工异步串行口(UART),兼容普通 8051 的串口
- 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）

STC12C1052，人民币 3.8 元

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 <1mA
  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

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

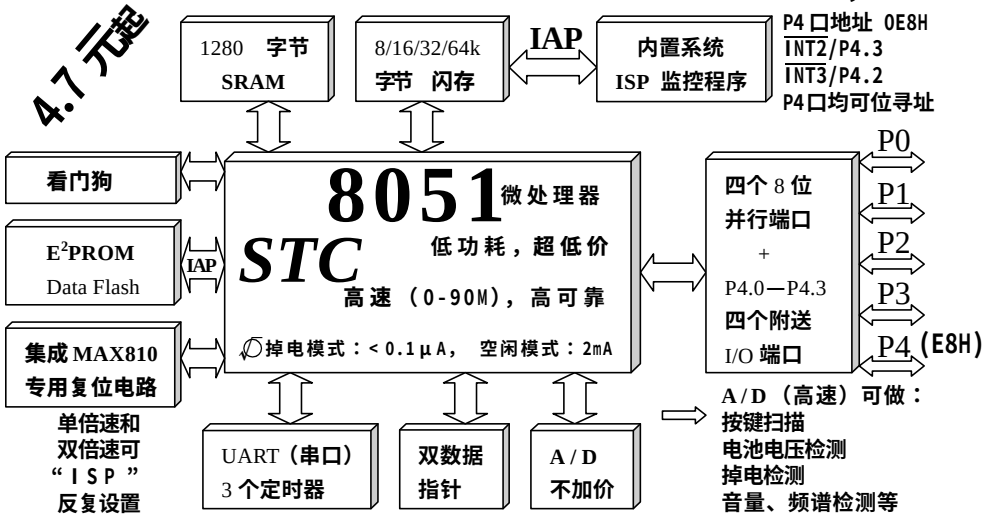
技术支持：13922805190 网址：www.MCU-Memory.com

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

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

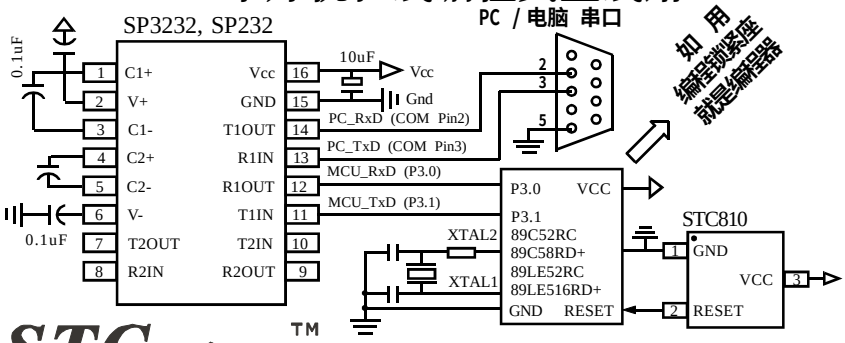
STC 89系列单片机选型一览表 超低价

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 单片机在线编程典型线路



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

目录

STC12C2052AD 系列 1T 单片机简介

STC12C2052 系列单片机是单时钟 / 机器周期(1T)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,内部集成 MAX810 专用复位电路。

特 点 ：

1.增强型 1T 流水线 / 精简指令集结构 8051 CPU

2.工作电压：5.5V - 3.4V（5V 单片机） / 3.8V - 2.4V（3V 单片机）

3.工作频率范围：0 - 35 MHz，相当于普通 8051 的 0～420MHz

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

5.片上集成 256 字节 RAM

6.15 个通用 I/O 口，复位后为： 准双向口 / 弱上拉（普通 8051 传统 I/O 口）

可设置成四种模式：准双向口 / 弱上拉，推挽 / 强上拉，仅为输入 / 高阻，开漏

7.ISP（在系统可编程）/ IAP（在应用可编程），无需专用编程器

可通过串口（P3.0/P3.1）直接下载用户程序，2 秒 ～ 3 秒即可完成一片

8.EEPROM 功能

9.看门狗

10.内部集成 MAX810 专用复位电路

11.时钟源：高精度外部晶体 / 时钟，内部 R/C 振荡器

用户在下载用户程序时，可选择是使用内部 R/C 振荡器还是外部晶体 / 时钟

常温下内部 R/C 振荡器频率为：5.65MHz ～ 5.95MHz

精度要求不高时，可选择使用内部时钟，但因为有温漂，应认为是 5MHz ～ 6.5MHz

12.共 2 个 16 位定时器 / 计数器

13.PWM(2 路）/PCA（可编程计数器阵列）

14.ADC，8 路 8 位精度

15. 通用异步串行口(UART)

16. SPI 同步通信口， 主模式 / 从模式

17.工作温度范围： 0 - 75℃ / -40 - +85℃

18.封装： PDIP-20，SOP-20(宽体)，TSSOP-20(超小封状，定货)

供 货 ：

现已开始批量供货（PDIP-20/SOP-20），欢迎提前定货(2-4 周)

STC12C5410AD 系列是 STC12C2052AD 系列的后续产品

1.内部 RAM 增加到 512 字节

2.内部 I/O 口 增加到 23 个（PDIP-28/SOP-28）， PLCC-32 为 27 个

3.内部 PCA/PWM 模块 增加到 4 个

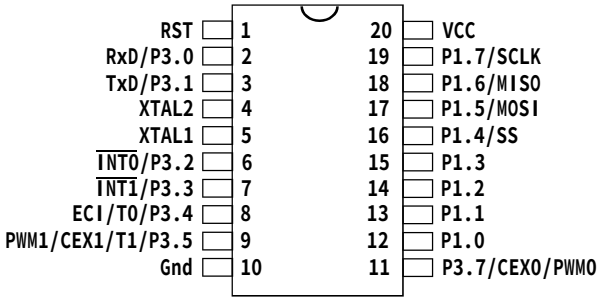
4.内部 A/D 模块 提升到 10 位 精度

5.内部 Flash 程序空间为 2K / 4K / 6K / 8K / 10K / 12K

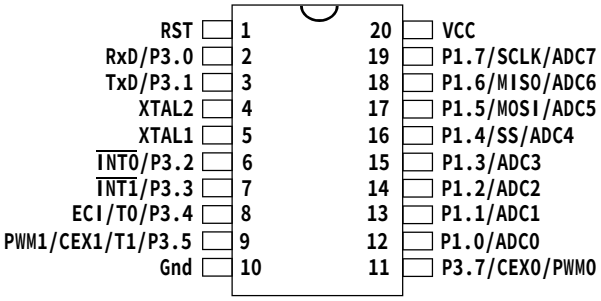
6.封装：PDIP-28(窄体)，SOP-28，PLCC-32，PDIP-20,SOP-20，TSSOP-20

7.供货：2005-12-1 开始提供样品（PDIP-28/SOP-28），2005 年 12 月底批量供货

STC12C2052AD 系列单片机管脚图及封装尺寸



STC12C2052

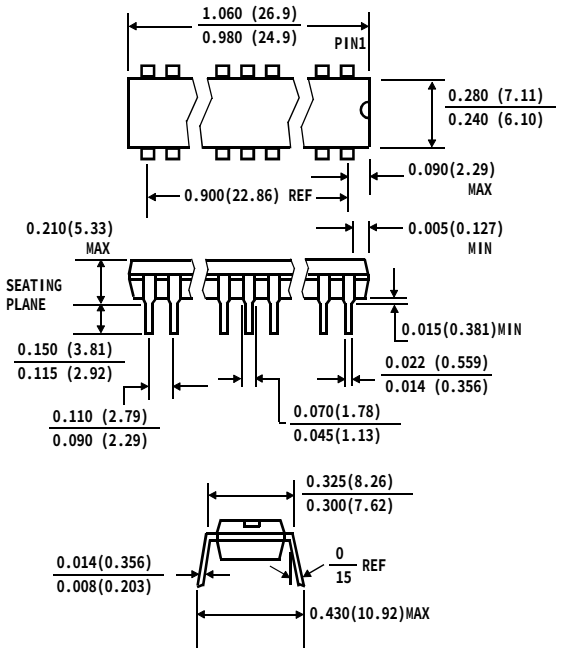


STC12C2052AD

20P3, 20-lead, 0.300" Wide, Plastic Dual Inline Package (PDIP-20)

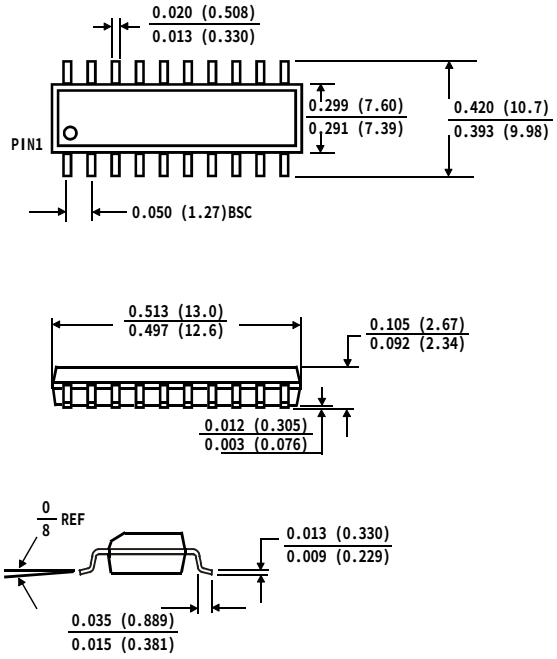
Dimensions in Inches and (Millimeters)

JEDEC STANDARD MS-001 AD



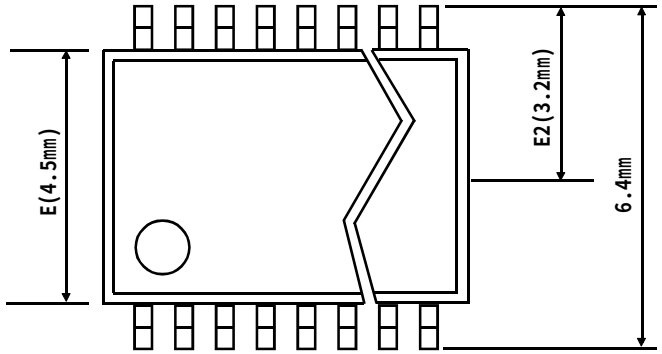
20S, 20-lead, 0.300" Wide, Plastic Gull WIng Small Outline (SOIC-20 / SOP-20)

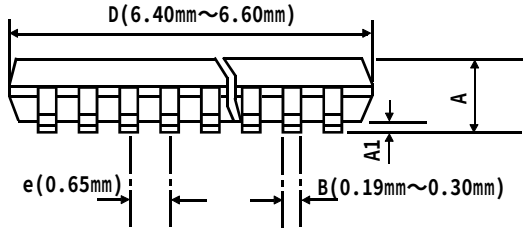
Dimensions in Inches and (Millimeters)

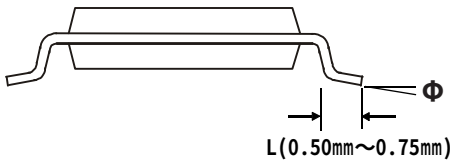


STC12LE4052AD 提供 TSSOP-20 封装
STC12C4052AD 的 TSSOP-20 封装需订货

PACKAGE：PLASTIC SHRINK SMALL OUTLINE (TSSOP-20，6.4mm×6.4mm)



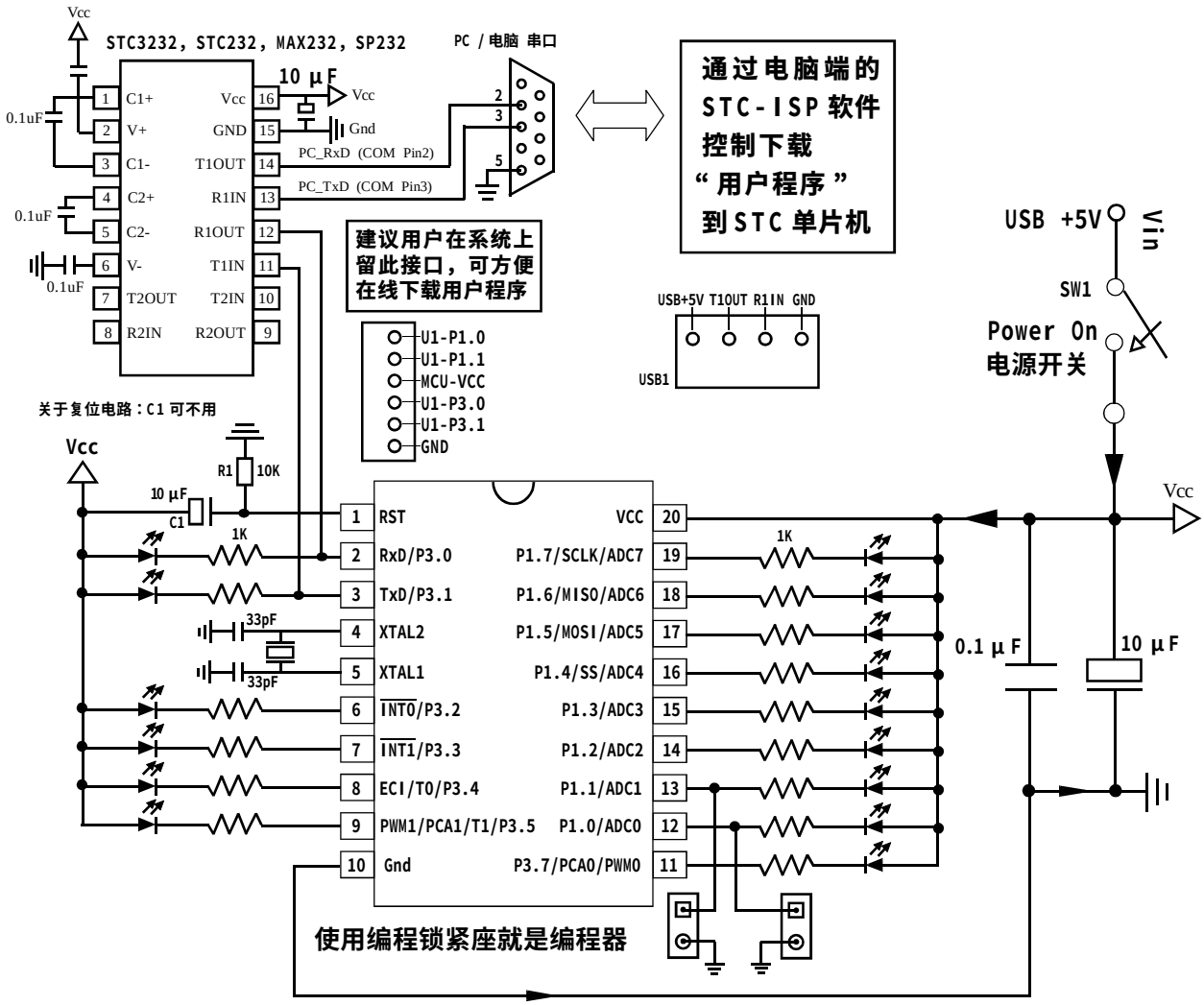




DIMENSIONS in inches (mm) Minimum/Maximum	20-PIN
A	- /0.043 (- /1.10)
A1	0.002/0.006 (0.05/0.15mm)
B	0.007/0.012 (0.19/0.30mm)
D	0.252/0.260 (6.40/6.60mm)
E	0.169/0.177 (4.30/4.50mm)
e	0.026 BSC (0.65mm BSC)
E2	0.126 BSC (3.20mm BSC)
L	0.020/0.030 (0.50/0.75mm)
Φ	0°/8°

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



此线路已做成一个 STC12C2052 系列单片机 ISP 下载编程工具，可直接赠送给客户

用户在自己的目标系统上，如将 P3.0/P3.1 经过 RS-232 电平转换器转换后连接到电脑的普通 RS-232 串口，就可以在系统编程 / 升级用户软件。建议如果用户板上无 RS-232 电平转换器，应引出一个插座，含 Gnd / P3.1 / P3.0 / Vcc 四个信号线，当然如能引出 Gnd / P3.1 / P3.0 / Vcc / P1.1 / P1.0 六个信号线为最好，这样就可以在用户系统上直接编程了。关于 ISP 编程的原理及应用指南详见附录部分“STC12C2052AD 系列单片机 ISP 编程 原理 工具使用说明”部分。另外我们有标准化的编程下载工具，用户可以在上面编程后再插到目标系统上，也可以借用它上面的 RS-232 电平转换器连接到电脑，以做下载编程之用。编程一个芯片大致需 2 秒到 3 秒钟，速度比普通的通用编程器快很多，故无须买通用编程器。

电脑端 STC-ISP 软件从网站 www.MCU-Memory 下载

STC12C2052AD 系列单片机选型一览表（全部有 ISP 功能）：

	工作电压(V)	Flash 程序存储器字节	SRAM 字节	定时器	UART	PCA PWM	A/D	I/O	看门狗	内置复位	EEPROM	SPI	封装 20-Pin
STC12C0552	5.5 - 3.4	512	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C0552AD	5.5 - 3.4	512	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12C1052	5.5 - 3.4	1K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C1052AD	5.5 - 3.4	1K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12C2052	5.5 - 3.4	2K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C2052AD	5.5 - 3.4	2K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12C3052	5.5 - 3.4	3K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C3052AD	5.5 - 3.4	3K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12C4052	5.5 - 3.4	4K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C4052AD	5.5 - 3.4	4K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12C5052	5.5 - 3.4	5K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12C5052AD	5.5 - 3.4	5K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE0552	2.4 - 3.8	512	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE0552AD	2.4 - 3.8	512	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE1052	2.4 - 3.8	1K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE1052AD	2.4 - 3.8	1K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE2052	2.4 - 3.8	2K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE2052AD	2.4 - 3.8	2K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE3052	2.4 - 3.8	3K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE3052AD	2.4 - 3.8	3K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE4052	2.4 - 3.8	4K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE4052AD	2.4 - 3.8	4K	256	2	有	2路	有	15	有	有	有	有	DIP/SOP
STC12LE5052	2.4 - 3.8	5K	256	2	有	2路		15	有	有	有	有	DIP/SOP
STC12LE5052AD	2.4 - 3.8	5K	256	2	有	2路	有	15	有	有	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
LCALL	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
LJMP	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,xx00	ISP_TRIG xxxx,xxxx	ISP_CONTR 0000,1000	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	IDLE_CLK xxxx,x000	C7h
B8h	IP x000,0000	SADEN don't use							BFh
B0h	P3 1x11,1111	P3M0 0000,0000	P3M1 0000,0000					IPH x000,0000	B7h
A8h	IE 0000,0000	SADDR don't use							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	SPSTAT 00xx,xxxx	SPCTL 0000,0100	SPDAT 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	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
IDLE_CLK	C7h	Clock Divder	-	-	-	-	-	IDLCLKS2	IDLCLKS1	IDLCLKS0	xxxx,x000

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 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	ELVDI	-	-	0000,00xx

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	ELVDI	-	-	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 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_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
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
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

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	1	WT2	WT1	WT0	0000,1000

STC12C2052AD 系列 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

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_LVD	EADC_SPI	ES	ET1	EX1	ET0	EX0	0000,0000
IP	B8h	Interrupt Priority Low	-	PPCA_LVD	PADC_SPI	PS	PT1	PX1	PT0	PX0	xx00,0000
IPH	B7h	Interrupt Priority High	-	PPCA_LVDH	PADC_SPIH	PSH	PT1H	PX1H	PT0H	PX0H	0000,0000
AUXR	8Eh	Auxiliary Register	T0x12	T1x12	UART_M0x6	EADCI	ESPI	ELVDI	-	-	0000,00xx
ADC_CONTR	C5h	A/D 转换控制寄存器	ADC_POWER	SPEED1	SPEED0	ADC_FLAG	ADC_START	CHS2	CHS1	CHS0	0x00,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000

STC12C2052AD 系列 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
ADC/SPI	002BH	5	PADC_SPIH,PADC_SPI	0,0	0,1	1,0	1,1	ADC_FLAG + SPIF
PCA/LVD	0033H	6	PPCA_LVDH,PPCA_LVD	0,0	0,1	1,0	1,1	CF + CCF0 + CCF1+ LVDF

PCA/PWM 特殊功能寄存器 ，其中部分位与 PCA 中断有关

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

STC12C2052AD 系列 8051 单片机 SPI 功能模块特殊功能寄存器 SPI Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000

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	ELVDI	-	-	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

EADCI： 0，禁止 A/D 中断； 1，允许 A/D 中断

ESPI： 0，禁止 SPI 中断； 1，允许 SPI 中断

ELVDI： 0， 禁止低压中断； 1，允许低压中断。

5V 单片机，3.7V 以下为低压，3V 单片机，2.4V 以下为低压，

如 ELVDI=1（允许低压中断），则会产生中断

系统工作时钟

STC12C2052AD 系列是 1T 的 8051 单片机，系统时钟兼容传统 8051。

现出厂标准配置是使用芯片内部的 R/C 振荡器，5V 单片机常温下频率是 5.65MHz - 5.95MHz,因为随着温度的变化，内部 R/C 振荡器的频率会有一些温飘，故内部 R/C 振荡器只适用于对时钟频率要求不敏感的场合。

在对 STC12C2052AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择：

“ 下次冷启动时使用内部 R/C 振荡器： No ”

这样下载完用户程序后，停电，再冷启动后单片机的工作时钟使用的就不是内部R/C振荡器，而是外部晶体振荡后产生的高精度时钟了（接在 XTAL1/XTAL2 管脚上），也可以直接从 XTAL1 脚输入外部时钟，XTAL2 脚浮空。

如果还要设置成使用内部 R/C 振荡器，在对 STC12C2052AD 系列单片机进行 ISP 下载用户程序时，可以在选项中选择：

“ 下次冷启动时使用内部 R/C 振荡器： Yes ”

空闲模式时的系统时钟

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
PCON	87h	Power Control	SMOD	SMOD0	LVDF	POF	GF1	GF0	PD	IDL	0011,0000
IDLE_CLK	C7h	Clock Divder	-	-	-	-	-	IDLCLKS2	IDLCLKS1	IDLCLKS0	xxxx,x000

如用户系统希望进入空闲模式后(MOV PCON,#00000001B)大幅降低功耗，还可对系统时钟进行分频，注意在STC12C2052AD系列中此分频只对空闲模式有效，正常工作时是无效的。

IDLCLKS2	IDLCLKS1	IDLCLKS0	空闲模式时，CPU的工作时钟
0	0	0	系统时钟(外部时钟或内部R/C振荡时钟)
0	0	1	系统时钟/2
0	1	0	系统时钟/4
0	1	1	系统时钟/8
1	0	0	系统时钟/16
1	0	1	系统时钟/32
1	1	0	系统时钟/64
1	1	1	系统时钟/128

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

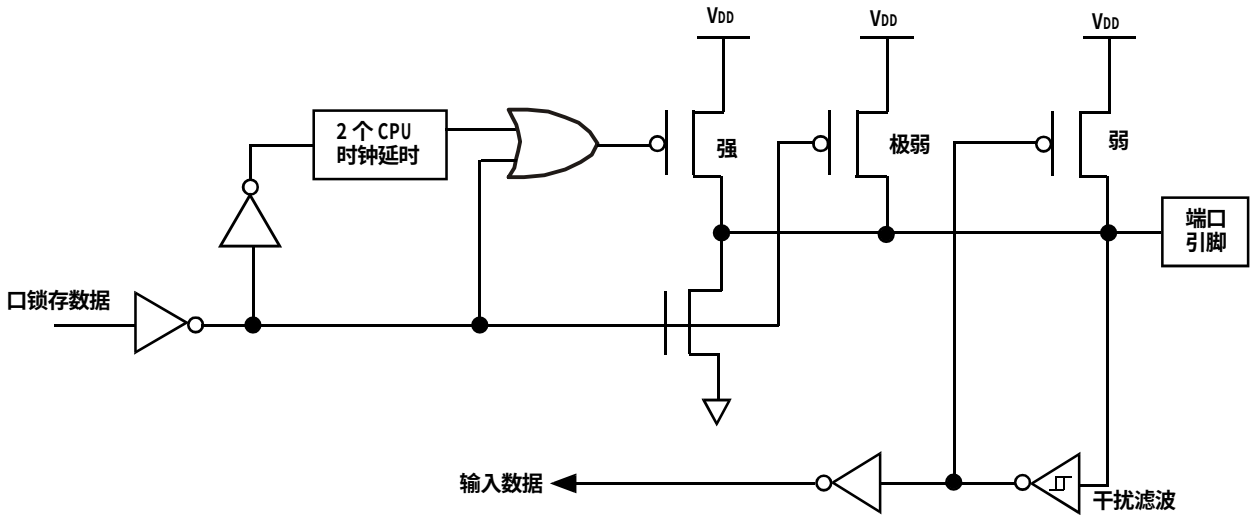
准双向口输出类型可用作输出和输入功能而不需重新配置口线输出状态。这是因为当口线输出为 1 时驱动能力很弱，允许外部装置将其拉低。当引脚输出为低时，它的驱动能力很强，可吸收相当大的电流。准双向口有 3 个上拉晶体管适应不同的需要。

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降到门槛电压以下。

第 2 个上拉晶体管，称为“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 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。

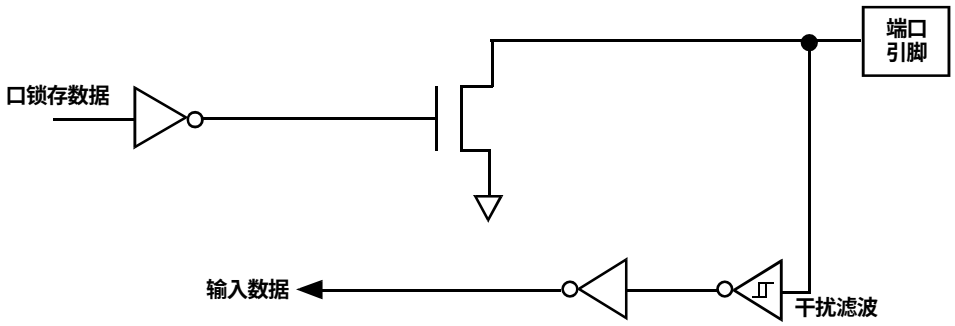


STC12LE2052 系列单片机为 3V 器件，如果用户在引脚加上 5V 电压，将会有电流从引脚流向 VDD，这样导致额外的功率消耗。因此，建议不要在准双向口模式中向3V单片机引脚施加5V电压，如使用的话，要加限流电阻，或用二极管做输入隔离，或用三极管做输出隔离。

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

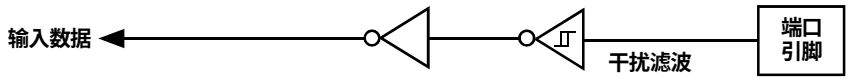
当口线锁存器为 0 时，开漏输出关闭所有上拉晶体管。当作为一个逻辑输出时，这种配置方式必须有外部上拉，一般通过电阻外接到 VDD。这种方式的下拉与准双向口相同。输出口线配置如下图所示。



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。

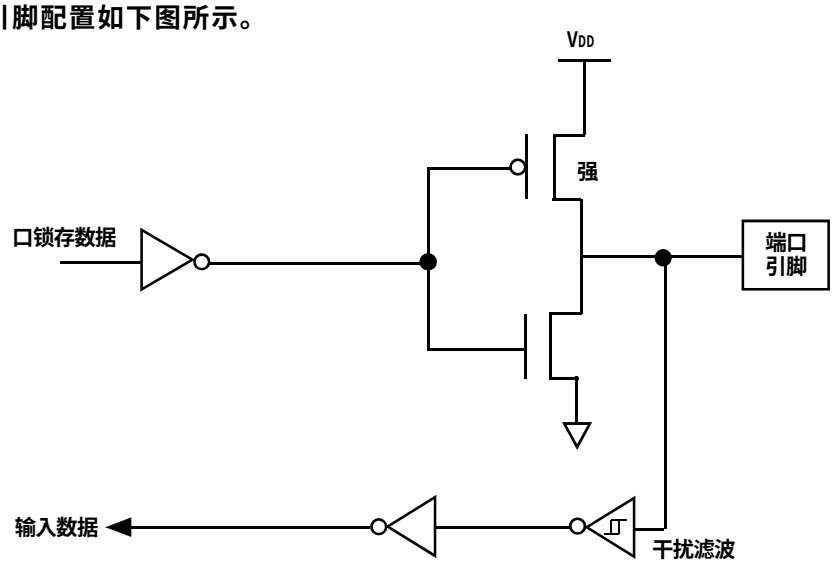


输入口带有一个施密特触发输入以及一个干扰抑制电路。

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 转换控制寄存器 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_POWER： ADC 电源控制位。

0：关闭 ADC 电源；1：给 AD 转换器提供电源

启动 AD 转换时要打开 AD 电源，AD 转换结束后关闭 AD 电源可降低功耗。

初次打开内部 A/D 转换模拟电源，需适当延时，等内部模拟电源稳定后，再启动 A/D 转换并且建议启动 A/D 转换后，在 A/D 转换结束之前，不改变任何 I/O 口的状态，有利于高精度 A/D 转换

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。

A/D 转换功能汇编程序示例

```
; ------------------------ 宏晶科技   2005/9/6 --------------------
; --------------------Mobile:0755-82948409,13922805190----------------
; ---------------------Email: support@mcu-memory.com------------------

;ADC DEMO_2052_ASM.ASM 汇编程序演示 STC12C2052AD 系列 MCU 的 A/D 转换功能。
;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。
;时钟 18.432MHz, 波特率 = 9600。
;各通道转换结果轮流在 P1 口用 LED 显示，通道号同时在 P3.2 -- P3.5 LED 显示。

;-----------------------------------------------------------------
;定义与 ADC 有关的特殊功能寄存器
ADC_CONTR    EQU     0C5H             ;A/D 转换控制寄存器
ADC_DATA     EQU     0C6H             ;A/D 转换结果寄存器
P1M0         EQU      91H             ;P1 口模式寄存器 0
P1M1         EQU      92H             ;P1 口模式寄存器 1

;-----------------------------------------------------------------
;定义变量
counter                  EQU 30H
display_AD_channel_ID    EQU 31H      ;当前通道号
AD_channel_1_result      EQU 32H      ;各通道 A/D 转换结果
AD_channel_2_result      EQU 33H
AD_channel_3_result      EQU 34H
AD_channel_4_result      EQU 35H
AD_channel_temp          EQU 36H

;-----------------------------------------------------------------
main:
    ACALL initiate_RS232

    MOV   display_AD_channel_ID, #0
    MOV   counter, #0
main_loop1:
    ACALL set_P1_ADC_channels        ;将 P1.1 -- P1.4 设置为适宜 A/D 转换的模式

    MOV   A, #01H                    ;P1.1 为 A/D 当前通道，测量电压并发送结果
    ACALL get_AD_result_and_send_it

    MOV   A, #02H                    ;P1.2 为 A/D 当前通道，测量电压并发送结果
    ACALL get_AD_result_and_send_it

    MOV   A, #03H                    ;P1.3 为 A/D 当前通道，测量电压并发送结果
    ACALL get_AD_result_and_send_it

    MOV   A, #04H                    ;P1.4 为 A/D 当前通道，测量电压并发送结果
    ACALL get_AD_result_and_send_it
```

```
    MOV   A, #00H                   ;连续发送 4 个 00H，便于观察输出显示
    ACALL Send_Byte
    ACALL Send_Byte
    ACALL Send_Byte
    ACALL Send_Byte

    INC   counter                   ;循环若干次后换通道
    MOV   A, counter
    CLR   C
    SUBB  A, #06H
    JC    main_1

    MOV   counter, #0
    INC   display_AD_channel_ID
    ANL   display_AD_channel_ID, #03H

main_1:
    ACALL set_P1_IO_port            ;P1 口设置为普通 I/O 模式
    ACALL display

    MOV   R2, #10
main_loop2:
    MOV   A, #0A0H                  ;延时
    ACALL delay
    DJNZ  R2, main_loop2

    SJMP  main_loop1

;-------------------------------------------------------------------
get_AD_result_and_send_it:          ;A = A/D 当前通道，测量电压并发送结果
    ACALL get_AD_result
    ACALL Send_Byte
    MOV   A, #1
    ACALL delay
    RET

;-------------------------------------------------------------------
get_AD_result:
    ANL   ADC_CONTR, #0E0H          ;1110,0000 清 ADC_FLAG, ADC_START 位和低 3 位
    ANL   A, #07H                   ;0000,0111 清 0 高 5 位
    MOV   AD_channel_temp, A        ;暂存当前通道号
    ORL   ADC_CONTR, A              ;设置 A/D 当前通道

    MOV   A, #1                     ;延时，使输入电压达到稳定
    ACALL delay
```

```
    CLR   A
    MOV   R7, A                     ;R7 用于检测 A/D 转换是否结束
    MOV   ADC_DATA, A               ;清 A/D 转换结果寄存器

    ORL   ADC_CONTR, #08H           ;0000,1000 令 ADCS = 1，启动A/D转换，
wait_AD_finishe:
    MOV   A, #10H                   ;0001,0000 测试A/D转换结束否
    ANL   A, ADC_CONTR
    JZ    wait_AD_finishe

    ANL   ADC_CONTR, #0E7H          ;1110,0111 清 ADC_FLAG 位，停止A/D转换

    MOV   A, AD_channel_temp        ;取回当前通道号
    DEC   A
    ADD   A, #AD_channel_1_result
    MOV   R0, A
    MOV   A, ADC_DATA                ;保存、返回 A/D 转换结果
    MOV   @R0, A
    RET

;-------------------------------------------------------------------
display:
    MOV   R2, display_AD_channel_ID  ;用 P3.2 -- P3.5 显示第几通道
    INC   R2
    SETB  C
    CLR   A
display_loop:
    RLC   A
    CLR   C
    DJNZ  R2, display_loop

    RL    A
    RL    A
    CPL   A
    ORL   P3, #3CH                  ;0011,1100 使 P3.2 -- P3.5 LED 熄灭
    ANL   P3, A

    MOV   A, display_AD_channel_ID   ;取相应通道的转换结果
    ADD   A, #AD_channel_1_result
    MOV   R0, A
    MOV   A, @R0
    CPL   A
    MOV   P1, A                     ;显示转换结果
    RET

;-------------------------------------------------------------------
set_P1_ADC_channels:                ;将 P1.2 -- P1.5 设置为适宜 A/D 转换的模式
```

```
    MOV   P1, #0FFH                 ;将 P1 口置高，为 A/D 转换作准备
    ORL   ADC_CONTR, #80H           ;1000,0000 打开 A/D 转换电源
    MOV   P1M0, #1EH                ;0001,1110，用作 A/D 转换的 P1.x 口，先设为开漏
    MOV   P1M0, #1EH                ;0001,1110，P1.2 -- P1.5 先设为开漏
                                    ;断开内部上拉电阻
    MOV   A, #20H
    ACALL delay
    RET

;-------------------------------------------------------------------
set_P1_IO_port:                     ;P1 口设置为普通 I/O 模式
    MOV   P1M0, #00H
    MOV   P1M0, #00H
    ANL   ADC_CONTR, #7FH           ;0111,1111 关闭 A/D 转换电源
    RET

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

initiate_RS232:                     ;串口初始化
    CLR   ES                        ;禁止串口中断
    MOV   TMOD, #20H                ;设置 T1 为波特率发生器
    MOV   SCON, #50H                ;0101,0000 8位数据位，无奇偶校验
    MOV   TH1, #0FBH                ;18.432MHz 晶振，波特率 = 9600
    MOV   TL1, #0FBH

    CLR   RI
    SETB  TR1
    SETB  ES                        ;允许串口中断
    RET

;-------------------------------------------------------------------
Send_Byte:                          ;发送一个字节
    CLR   TI                        ;清零串口发送中断标志
    MOV   SBUF, A
Send_Byte_wait:                     ;等待发送完毕
    JNB   TI, Send_Byte_wait
    CLR   TI                        ;清零串口发送中断标志
    RET

;-------------------------------------------------------------------
delay:                              ;延时
    PUSH  02
    PUSH  03
    PUSH  04
```

```
    MOV   R4, A
delay_loop1:
    MOV   R2, #18H
    MOV   R3, #0
delay_loop2:
    DJNZ  R3, delay_loop2
    DJNZ  R2, delay_loop2
    DJNZ  R4, delay_loop1

    POP   04
    POP   03
    POP   02
    RET
;-------------------------------------------------------------------
    END
```

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 STC12C2052AD 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 = (12 * Pre_scale *32768)/Oscillator frequency
WDTCR           EQU 0E1H ;看门狗地址
WDT_TIME_LED    EQU P1.5 ;用 P1.5 控制看门狗溢出时间指示灯,
                         ;看门狗溢出时间可由该指示灯亮的时间长度或熄灭的时间长度表示
WDT_FLAG_LED    EQU P1.7 ;用 P1.7 控制看门狗溢出复位指示灯，如点亮表示为看门狗溢出复位
Last_WDT_Time_LED_Status  EQU 00H ;位变量，存储看门狗溢出时间指示灯的上一次状态位
;WDT 复位时间(Oscillator frequency = 18.432MHz):
;Pre_scale_Word  EQU 00111100B   ;清0、启动看门狗，预分频数=32   0.68S
Pre_scale_Word  EQU 00111101B   ;清0、启动看门狗，预分频数=64   1.36S
;Pre_scale_Word  EQU 00111110B   ;清0、启动看门狗，预分频数=128  2.72S
;Pre_scale_Word  EQU 00111111B   ;清0、启动看门狗，预分频数=256  5.44S
    ORG   0000H
    AJMP  MAIN
    ORG   0100H
MAIN:
    MOV   A, WDTCR              ;检测是否为看门狗复位
    ANL   A, #10000000B
    JNZ   WDT_Reset             ;WDTCR.7 = 1，看门狗复位，跳转到看门狗复位程序
;上电复位，冷启动，RAM 单元内容为随机值
    SETB  Last_WDT_Time_LED_Status       ;上电复位,
                                         ;初始化看门狗溢出时间指示灯的状态位 = 1
    CLR   WDT_TIME_LED                   ;上电复位，点亮看门狗溢出时间指示灯
    MOV   WDTCR, #Pre_scale_Word ;启动看门狗
WAIT1:
    SJMP  WAIT1                 ;循环执行本语句(停机)，等待看门狗溢出复位

;看门狗复位，热启动，RAM 单元内容不变，为复位前的值
WDT_Reset:                      ;看门狗复位，热启动
    CLR   WDT_FLAG_LED          ;点亮看门狗溢出复位指示灯

    JB    Last_WDT_Time_LED_Status, Power_Off_WDT_TIME_LED
    ;根据看门狗溢出时间指示灯的上一次状态位设置 WDT_TIME_LED 灯,
    ;若上次亮本次就熄灭，若上次熄灭本次就亮
    CLR   WDT_TIME_LED          ;上次熄灭本次点亮看门狗溢出时间指示灯
    CPL   Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT2:
    SJMP  WAIT2                 ;循环执行本语句(停机)，等待看门狗溢出复位
Power_Off_WDT_TIME_LED:
    SETB  WDT_TIME_LED          ;上次亮本次就熄灭看门狗溢出时间指示灯
    CPL   Last_WDT_Time_LED_Status ;将看门狗溢出时间指示灯的上一次状态位取反
WAIT3:
    SJMP  WAIT3                 ;循环执行本语句(停机)，等待看门狗溢出复位
    END
```

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  ET0            ;要由外部中断 1 唤醒,“ ET0=1 ”是必须的,硬件就这样做的
                         ;外部中断 0 就无此必要,建议 Powerdown 用外部中断 0 唤醒

    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,xx00
ISP_TRIG	E6h	ISP/IAP Flash Command Trigger									xxxx,xxxx
ISP_CONTR	E7h	ISP/IAP Control Register	ISPEN	SWBS	SWRST	CMD_FAIL	1	WT2	WT1	WT0	0000,1000

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

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,STC12C5052AD 系列除外。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	1	WT2	WT1	WT0	0000,1000

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
0	1	1	2	660	252144	12MHz
0	1	0	2	1100	420240	20MHz
0	0	1	2	1320	504288	24MHz
0	0	0	2	1760	672384	30MHz

STC12C2052AD系列单片机的底层ISP固件为Ver3.3D(支持EEPROM功能)。老版本Ver3.2D部分不支持EEPROM功能,如需使用IAP/EEPROM可免费更换新的单片机。

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。
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
; 可加入软件陷阱判断，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
; 可加入软件陷阱判断，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
; 可加入软件陷阱判断，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
;可加入软件陷阱判断，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
;可加入软件陷阱判断，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
;可加入软件陷阱判断，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
;可加入软件陷阱判断，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
;可加入软件陷阱判断，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
;可加入软件陷阱判断，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
```

```
;STC12C2052AD 系列单片机 EEPROM/IAP 功能程序演示
;-----------------------------------------------
;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 80H             ;<30MHz
;ENABLE_ISP      EQU 81H             ;<24MHz
 ENABLE_ISP      EQU 82H             ;<20MHz
;ENABLE_ISP      EQU 83H             ;<12MHz
;ENABLE_ISP      EQU 84H             ;<6MHz
;ENABLE_ISP      EQU 85H             ;<3MHz
;ENABLE_ISP      EQU 86H             ;<2MHz
;ENABLE_ISP      EQU 87H             ;<1MHz

DEBUG_DATA      EQU 5AH
;------------------------
;选择 MCU 型号
DATA_FLASH_START_ADDRESS EQU 1000H ;STC12C2052AD
;-----------------------------------------------
    ORG   0000H
    AJMP  main
;-----------------------------------------------
    ORG   0100H
main:
    MOV     P1,#0F0H        ;演示程序开始工作
    LCALL   Delay           ;延时
    MOV     P1,#0FH         ;演示程序开始工作
    LCALL   Delay           ;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
    LCALL  Delay     ;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
;***********************************************
```

STC12C2052AD 系列单片机定时器的使用

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下表所列。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。

寄存器 TMOD 各位的功能描述

TMOD 地址：89H 复位值：00H

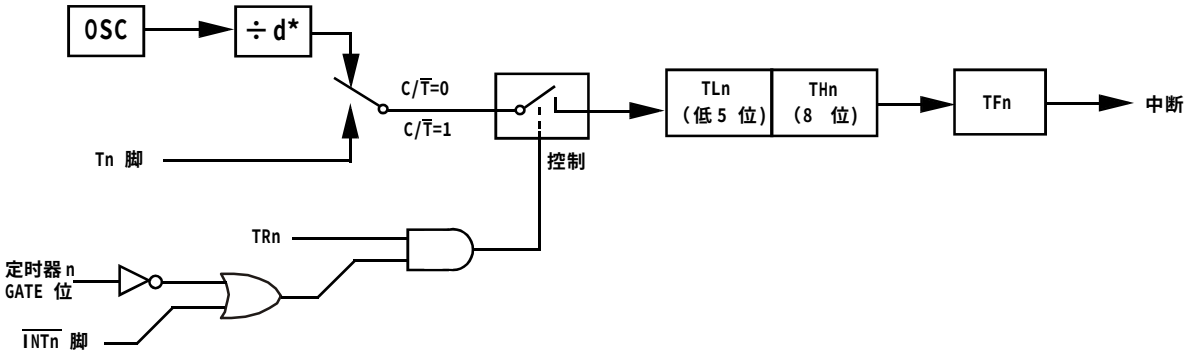
不可位寻址

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.7/	GATE	TMOD.7 控制定时器 1,置 1 时只有在 $\overline{INT1}$ 脚为高及 TR1 控制位置 1 时才可打开定时器 / 计数器 1。
TMOD.3/	GATE	TMOD.3 控制定时器 0,置 1 时只有在 $\overline{INT0}$ 脚为高及 TR0 控制位置 1 时才可打开定时器 / 计数器 0。
TMOD.6/	C/$\overline{T}$	TMOD.6 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 1 用作计数器（从 T1/P3.5 脚输入）
TMOD.2/	C/$\overline{T}$	TMOD.2 控制定时器 0 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 1 用作计数器（从 T0/P3.4 脚输入）
TMOD.5/TMOD.4	M1、M0	定时器 / 计数器 1 模式选择
	0 0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL1 只用低 5 位参与分频，TH1 整个 8 位全用。
	0 1	16 位定时器 / 计数器，TL1、TH1 全用
	1 0	8 位自动重装载定时器，当溢出时将 TH1 存放的值自动重装入 TL1。
	1 1	定时器 / 计数器 1 此时无效（停止计数）。
TMOD.1/TMOD.0	M1、M0	定时器 / 计数器 0 模式选择
	0 0	13 位定时器 / 计数器，兼容 8048 定时器模式，TL0 只用低 5 位参与分频，TH0 整个 8 位全用。
	0 1	16 位定时器 / 计数器，TL0、TH0 全用
	1 0	8 位自动重装载定时器，当溢出时将 TH0 存放的值自动重装入 TL0。
	1 1	定时器 0 此时作为双 8 位定时器 / 计数器。TL0 作为一个 8 位定时器 / 计数器，通过标准定时器 0 的控制位控制。TH0 仅作为一个 8 位定时器，由定时器 1 的控制位控制。

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。下图所示为模式 0 工作方式。此模式下，定时器配置为 13 位的计数器，由 TLn 的低 5 位和 THn 的 8 位所构成。TLn 低 5 位溢出向 THn 进位，THn 计数溢出置位 TCON 中的溢出标志位 TFn（n=0，1）。GATE=0 时，如 TRn=1，则定时器计数。GATE=1 时，允许由外部输入 $\overline{INT1}$ 控制定时器 1，$\overline{INT0}$ 控制定时器 0，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见 TCON 寄存器各位的具体功能描述表。



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

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

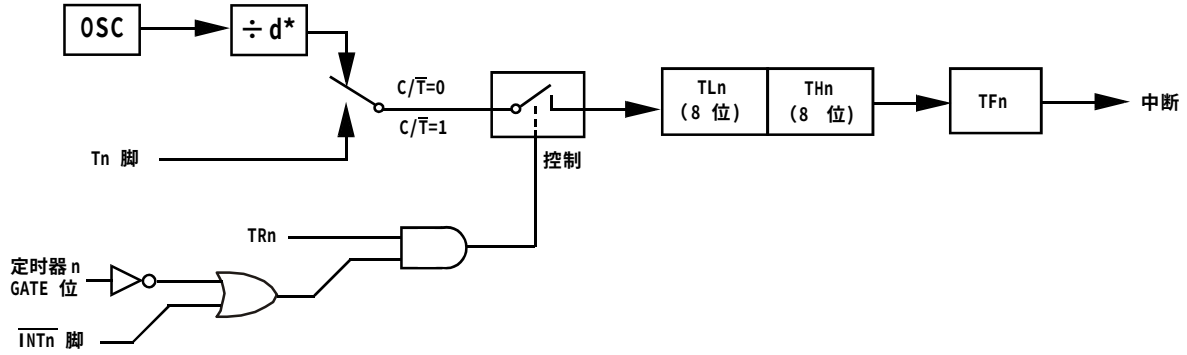
表 寄存器 TCON 各位的功能描述

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 / 计数器 1 溢出标志位。当 T1 被允许计数后，T1 从初值开始加 1 计数，最高位产生溢出时，置“ 1 ”TF1，并向 CPU 请求中断，当 CPU 响应时，由硬件清“ 0 ”TF1，TF1 也可以由程序查询或清“ 0 ”。								
TCON.6	TR1	定时器 T1 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.7）=0，TR1=1 时就允许 T1 开始计数，TR1=0 时禁止 T1 计数。当 GATE（TMOD.7）=1，TR1=1 且 INT1 输入高电平时，才允许 T1 计数。								
TCON.5	TF0	定时器 / 计数器 0 溢出标志位。当 T0 被允许计数后，T0 从初值开始加 1 计数，最高位产生溢出时，置“ 1 ”TF0，并向 CPU 请求中断，当 CPU 响应时，由硬件清“ 0 ”TF0，TF0 也可以由程序查询或清“ 0 ”。								
TCON.4	TR0	定时器 T0 的运行控制位。该位由软件置位和清零。当 GATE（TMOD.3）=0，TR0=1 时就允许 T0 开始计数，TR1=0 时禁止 T0 计数。当 GATE（TMOD.3）=1，TR0=1 且 $\overline{\text{INT0}}$ 输入高电平时，才允许 T0 计数。								
TCON.3	IE1	外部中断 1 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE1 位清 0。								
TCON.2	IT1	外部中断 1 触发方式控制位。IT1=0 时，外部中断 1 为低电平触发方式，当 $\overline{\text{INT1}}$（P3.3）输入低电平时，置位 IE1。采用低电平触发方式时，外部中断源（输入到 INT1）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.3 要变高），否则将产生另一次中断。当 IT1=1 时，则外部中断 1（$\overline{\text{INT1}}$）端口由“ 1 ”→“ 0 ”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。								
TCON.1	IE0	外部中断 0 中断请求标志位。当主机响应中断转向该中断服务程序执行时，由内部硬件自动将 IE0 位清 0。								
TCON.0	IT0	外部中断 1 触发方式控制位。IT0=0 时，外部中断 0 为低电平触发方式，当 $\overline{\text{INT0}}$（P3.2）输入低电平时，置位 IE0。采用低电平触发方式时，外部中断源（输入到 $\overline{\text{INT0}}$）必须保持低电平有效，直到该中断被 CPU 响应，同时在该中断服务程序执行完之前，外部中断源必须被清除（P3.2 要变高），否则将产生另一次中断。当 IT0=1 时，则外部中断 0（$\overline{\text{INT0}}$）端口由“ 1 ”→“ 0 ”下降沿跳变，激活中断请求标志位 IE1，向主机请求中断处理。								

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 1 及定时器 0。

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。

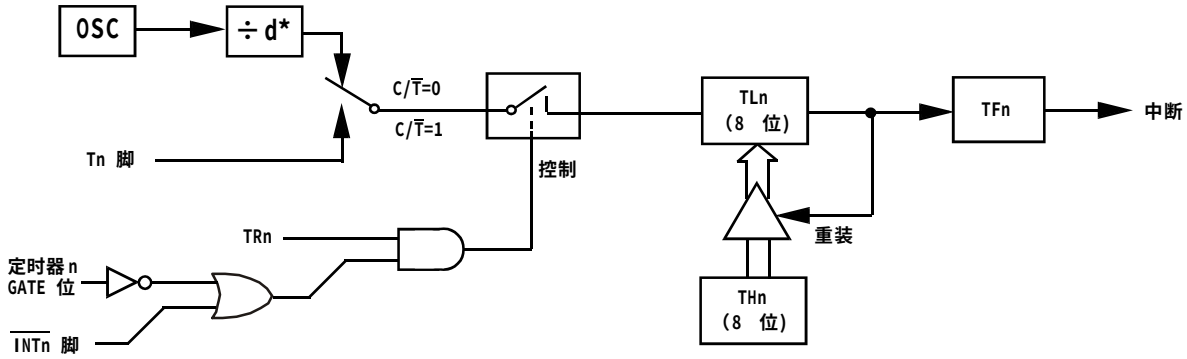


* 在 T0x12 = 0 模式下，d=12(12 时钟模式)； 在 T0x12 = 1 模式下，d=1(1T)。

图 定时器 / 计数器 0 和定时器 / 计数器 1 的模式 1 ： 16 位定时 / 计数器

3 . 模式 2

此模式下定时器 / 计数器 0 和 1 作为可自动重装载的 8 位计数器（TLn），如下图所示。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。



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

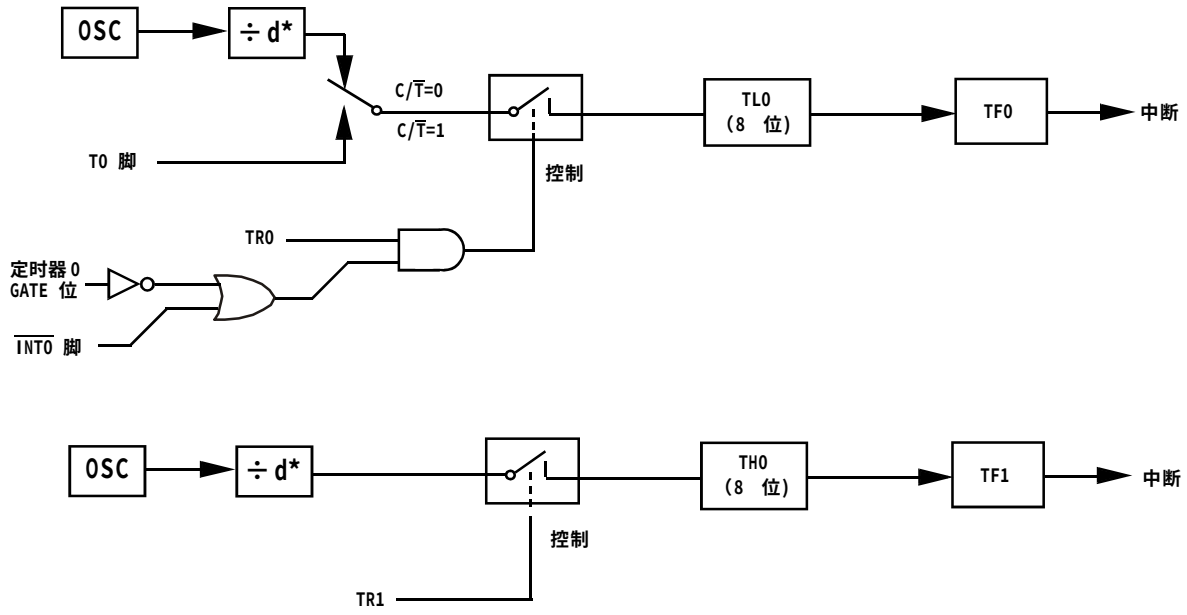
图 定时器 / 计数器 0 和 1 的模式 2 ： 8 位自动重装载

4 . 模式 3

对定时器 1 ，在模式 3 时，定时器 1 停止计数，效果与将 TR1 设置为 0 相同。

对定时器 0，此模式下定时器 0 的 TL0 及 TH0 作为 2 个独立的 8 位计数器。下图为模式 3 时的定时器 0 逻辑图。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 是为了增加一个附加的 8 位定时器 / 计数器而提供的，使单片机具有三个定时器 / 计数器。模式 3 只适用于定时器 / 计数器 0，定时器 T1 处于模式 3 时相当于 TR1=0，停止计数（此时 T1 可用来作串行口波特率发生器），而 T0 可作为两个定时器用。



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

图 定时 / 计数器 0 的模式 3 ： 两个 8 位计数器

定时器应用举例

【例 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：

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

$$\text{实际定时时间 Tc} = x \cdot Tp$$

式中 Tp 为机器周期，Tc 为所需定时时间，x 为所需计数次数。Tp 和 Tp 一般为已知值，在求出 Tp 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

$$(x)\text{补} = 2^n - x$$

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

$$x = \frac{5ms}{2\ \mu s} = 2500\text{ 次}$$

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 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 Tc = 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：

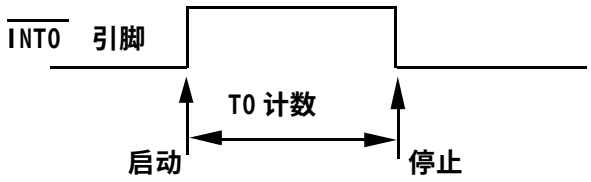
```
START：
        ⋮                       ；主程序段
        MOV  SP，#60H           ；设置堆栈区域
        MOV  TMOD，#10H         ；选择 T 1 、定时模式，工作方式 1
        MOV  TH1，#0D8H         ；设置高字节计数初值
        MOV  TL1，#0F0H         ；设置低字节计数初值
        SETB EA                 } 开中断
        SETB ET1                }
        ⋮                       ；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                  } 现场保护
        PUSH DPL                }
        PUSH DPH                }
        ⋮
        MOV  TL1,#0F0H          } 重新置初值
        MOV  TH1,#0D8H          }
        ⋮                       ；中断处理主体程序
        POP  DPH                } 现场恢复
        POP  DPL                }
        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 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（T0）以定时模式，工作方式 1 对 $\overline{\text{INT0}}$ 引脚上的正脉冲进行脉宽测试。



设置 GATE 为 1，机器周期 TP 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 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              ;
                                } 现场入栈保护
            PUSH DPL            ;
            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5】 某应用系统需通过P1.0和P1.1分别输出周期为200μs和400μs的方波。为此，系统选用定时器/计数器0（T0），定时方式3，主频为6MHz，TP=2μs，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
        MOVR 1 ，A                 ；将 THx 存于 R1 中
         ⋮
```

附录 A： STC12C2052 系列单片机 PWM/PCA 应用

STC12C2052AD 系列 8051 单片机 PCA/PWM 特殊功能寄存器 PCA/PWM SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
CCON	D8h	PCA Control Register	CF	CR	-	-	-	-	CCF1	CCF0	00xx,xx00
CMOD	D9h	PCA Mode Register	CIDL	-	-	-	-	CPS1	CPS0	ECF	0xxx,x000
CCAPM0	DAh	PCA Module 0 Mode Register	-	ECOM0	CAPP0	CAPN0	MAT0	TOG0	PWM0	ECCF0	x000,0000
CCAPM1	DBh	PCA Module 1 Mode Register	-	ECOM1	CAPP1	CAPN1	MAT1	TOG1	PWM1	ECCF1	x000,0000
CL	E9h	PCA Base Timer Low									0000,0000
CH	F9h	PCA Base Timer High									0000,0000
CCAP0L	EAh	PCA Module-0 Capture Register Low									0000,0000
CCAP0H	FAh	PCA Module-0 Capture Register High									0000,0000
CCAP1L	EBh	PCA Module-1 Capture Register Low									0000,0000
CCAP1H	FBh	PCA Module-1 Capture Register High									0000,0000
PCA_PWM0	F2h	PCA PWM Mode Auxiliary Register 0	-	-	-	-	-	-	EPC0H	EPC0L	xxxx,xx00
PCA_PWM1	F3h	PCA PWM Mode Auxiliary Register 1	-	-	-	-	-	-	EPC1H	EPC1L	xxxx,xx00

CMOD - PCA 模式 寄存器的位分配 （地址：D9H）

位	7	6	5	4	3	2	1	0
符 号	CIDL	-	-	-	-	CPS1	CPS0	ECF

CMOD - PCA 模式 寄存器的位描述 （地址：D9H）

位	符号	描述
7	CIDL	计数器阵列空闲控制：CIDL=0时，空闲模式下PCA计数器继续工作。CIDL＝1时，空闲模式下PCA计数器停止工作。
6 - 3	-	保留为将来之用。
2 - 1	CPS1,CPS0	PCA计数脉冲选择（见下表）。
0	ECF	PCA计数溢出中断使能：ECF=1时，使能寄存器CCON CF位的中断。ECF=0时，禁止该功能。

CMOD - PCA 计数器阵列的计数脉冲选择 （地址：D9H）

CPS1	CPS0	选 择 PCA输 入
0	0	0，内 部 时 钟 ， fosc/12
0	1	1，内 部 时 钟 ， fosc/2
1	0	2，定 时 器 0溢 出
1	1	3，ECI/P3.4脚 的 外 部 时 钟 输 入 （最 大 速 率 ＝ fosc/4）

CCON - PCA 控制寄存器的位分配 （地址：D8H）

位	7	6	5	4	3	2	1	0
符 号	CF	CR	-	-	-	-	CCF1	CCF0

CCON - PCA 控制寄存器的位描述 （地址：D8H）

位	符号	描述
7	CF	PCA计数器阵列溢出标志。计数值翻转时该位由硬件置位。如果CMOD寄存器的ECF位置位，CF标志可用来产生中断。CF位可通过硬件或软件置位，但只可通过软件清零。
6	CR	PCA计数器阵列运行控制位。该位通过软件置位，用来起动PCA计数器阵列计数。该位通过软件清零，用来关闭PCA计数器。
5 - 2	-	保留位，保留为将来使用。
1	CCF1	PCA模块1中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。
0	CCF0	PCA模块0中断标志。当出现匹配或捕获时该位由硬件置位。该位必须通过软件清零。

CCAPMn - PCA 比较 / 捕获模块寄存器的位分配（CCAPM0 地址：0DAH；CCAPM1 地址：0DBH）

位	7	6	5	4	3	2	1	0
符 号	-	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn

CCAPMn - PCA 比较 / 捕获模块寄存器的位描述 （ n：0，1 ）

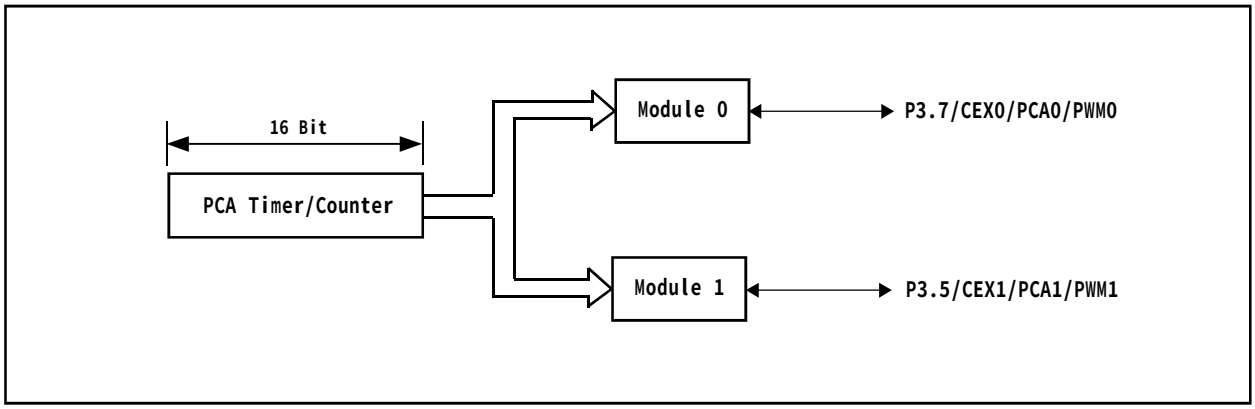
位	符号	描述 n：0,1
7	-	保留为将来之用。
6	ECOMn	使能比较器。ECOMn＝1时使能比较器功能。
5	CAPPn	正捕获。CAPPn＝1时使能上升沿捕获。
4	CAPNn	负捕获。CAPNn＝1时使能下降沿捕获。
3	MATn	匹配。当MATn＝1时，PCA计数值与模块的比较/捕获寄存器的值的匹配将置位CCON寄存器的中断标志位CCFn。
2	TOGn	翻转。当TOGn＝1时，PCA计数值与模块的比较/捕获寄存器的值的匹配将使CEXn脚翻转。(CEX0/P3.7,CEX1/P3.5)
1	PWMn	脉宽调节模式。当PWMn＝1时，使能CEXn脚用作脉宽调节输出。
0	ECCFn	使能CCFn中断。使能寄存器CCON的比较/捕获标志CCFn，用来产生中断。

PCA 模块工作模式（CCAPMn 寄存器，n：0，1）

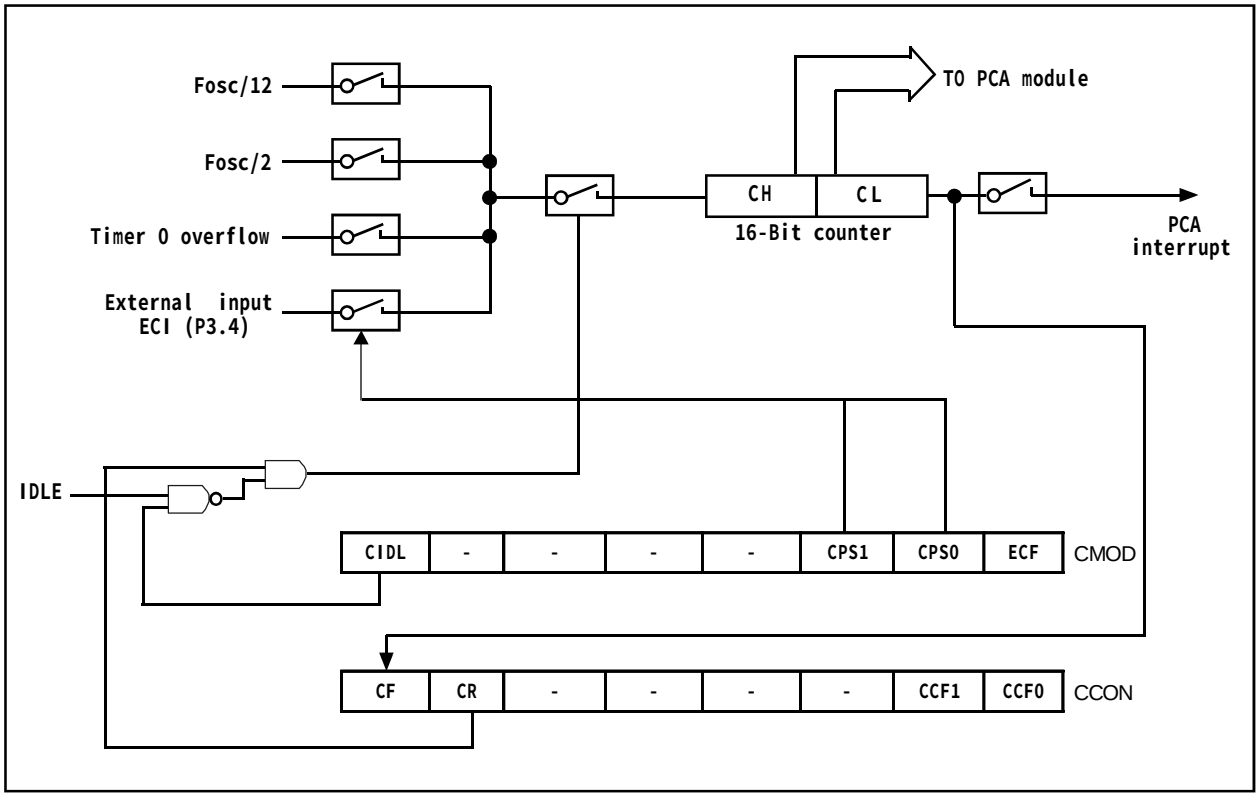
ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	模块功能
0	0	0	0	0	0	0	无此操作
X	1	0	0	0	0	X	16位捕获模式，由CEXn的上升沿触发
X	0	1	0	0	0	X	16位捕获模式，由CEXn的下降沿触发
X	1	1	0	0	0	X	16位捕获模式，由CEXn的跳变触发
1	0	0	1	0	0	X	16位软件定时器
1	0	0	1	1	0	X	16位高速输出
1	0	0	0	0	1	0	8位PWM

STC12C2052AD 系列单片机可编程计数器阵列（PCA）

PCA 含有一个特殊的 16 位定时器，有 2 个 16 位的捕获 / 比较模块与之相连。每个模块可编程工作在 4 种模式下：上升 / 下降沿捕获、软件定时器、高速输出或可调制脉冲输出。每个模块都与 P3 口相连，模块 0 连接到 P3.7（CEX0/PCA0/PWM0），模块 1 连接到 P3.5（CEX1/PCA1/PWM1）。寄存器 CH 和 CL 的内容是正在自由递增计数的 16 位 PCA 定时器的值。PCA 定时器是 2 个模块的公共时间基准，可通过编程工作在：1/12 振荡频率、1/2 振荡频率、定时器 0 溢出或 ECI 脚的输入（P3.4）。定时器的计数源由 CMOD SFR 的 CPS1 和 CPS0 位来确定（见 CMOD 特殊功能寄存器说明）。



Programmable Counter Array



PCA Timer/Counter

CMOD SFR 还有 2 个位与 PCA 相关。它们分别是：CIDL，空闲模式下允许停止 PCA；ECF，置位时，使能 PCA 中断，当 PCA 定时器溢出将 PCA 计数溢出标志 CF（CCON SFR）置位。

CCON SFR 包含 PCA 的运行控制位（CR）和 PCA 定时器标志（CF）以及各个模块的标志（CCF1/CCF0）。通过软件置位 CR 位（CCON.6）来运行 PCA。CR 位被清零时 PCA 关闭。当 PCA 计数器溢出时，CF 位（CCON.7）置位,如果 CMOD 寄存器的 ECF 位置位,就产生中断。CF 位只可通过软件清除。CCON 寄存器的位 0～1 是 PCA 各个模块的标志（位 0 对应模块 0，位 1 对应模块 1），当发生匹配或比较时由硬件置位。这些标志也只能通过软件清除。所有模块共用一个中断向量。PCA 的中断系统如图所示。

PCA 的每个模块都对应一个特殊功能寄存器。它们分别是：模块 0 对应 CCAPM0，模块 1 对应 CCAPM1。特殊功能寄存器包含了相应模块的工作模式控制位。

PCA 的每个模块都对应一个特殊功能寄存器。它们分别是：模块 0 对应 CCAPM0，模块 1 对应 CCAPM1。特殊功能寄存器包含了相应模块的工作模式控制位。

当模块发生匹配或比较时，ECCF 位（CCAPMn.0，n ＝ 0，1，由工作的模块决定）使能 CCON SFR 的 CCFn 标志来产生中断。

PWM（CCAPMn.1）用来使能脉宽调制模式。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果 TOG 位（CCAPMn.2）置位，模块的 CEXn 输出将发生翻转。

当 PCA 计数值与模块的捕获 / 比较寄存器的值相匹配时，如果匹配位 MATn（CCAPMn.3）置位，CCON 寄存器的 CCFn 位将被置位。

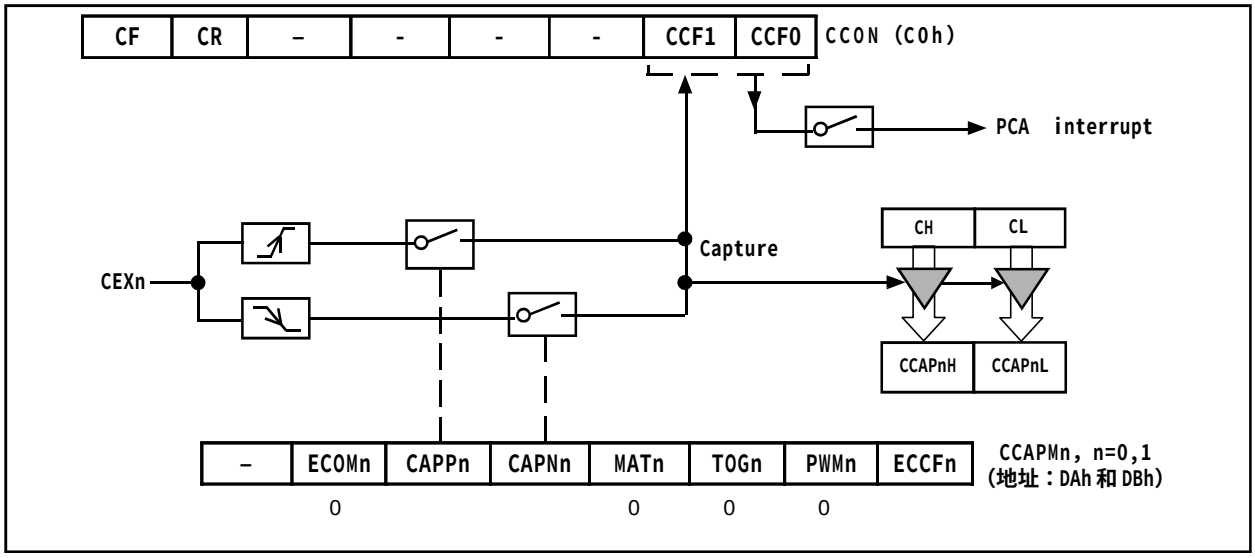
CAPNn（CCAPMn.4）和 CAPPn（CCAPMn.5）用来设置捕获输入的有效沿。CAPNn 位使能下降沿有效，CAPPn 位使能上升沿有效。如果两位都置位，则两种跳变沿都被使能，捕获可在两种跳变沿产生。

通过置位 CCAPMn 寄存器的 ECOMn 位（CCAPMn.6）来使能比较器功能。

每个 PCA 模块还对应另外两个寄存器，CCAPnH 和 CCAPnL。当出现捕获或比较时，它们用来保存 16 位的计数值。当 PCA 模块用在 PWM 模式中时，它们用来控制输出的占空比。

PCA 捕获模式

要使一个 PCA 模块工作在捕获模式（下图），寄存器 CCAPMn 的两位（CAPNn 和 CAPPn）或其中任何一位必须置 1。对模块的外部 CEXn 输入（CEX0/P3.7，CEX1/P3.5 口）的跳变进行采样。当采样到有效跳变时，PCA 硬件就将 PCA 计数器阵列寄存器（CH 和 CL）的值装载到模块的捕获寄存器中（CCAPnL 和 CCAPnH）。

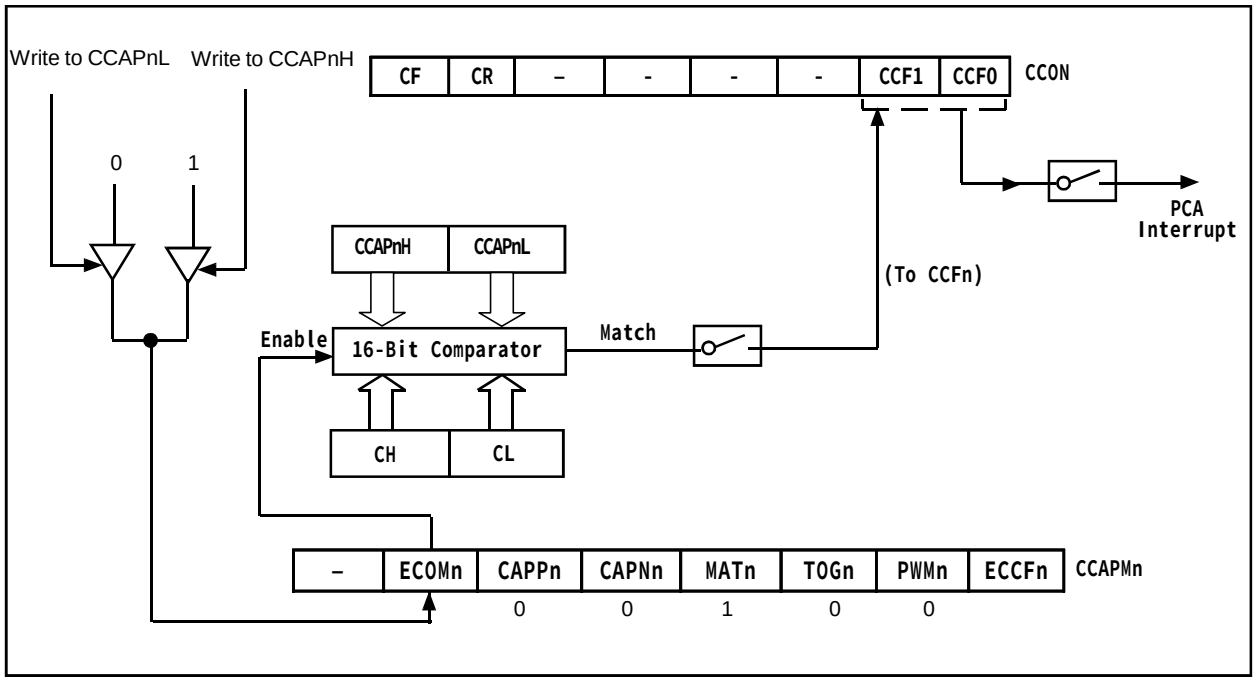


PCA Capture Mode （PCA 捕获模式图）

如果 CCON SFR 的位 CCFn 和 CCAPMn SFR 的位 ECCFn 位被置位，将产生中断。

16 位软件定时器模式

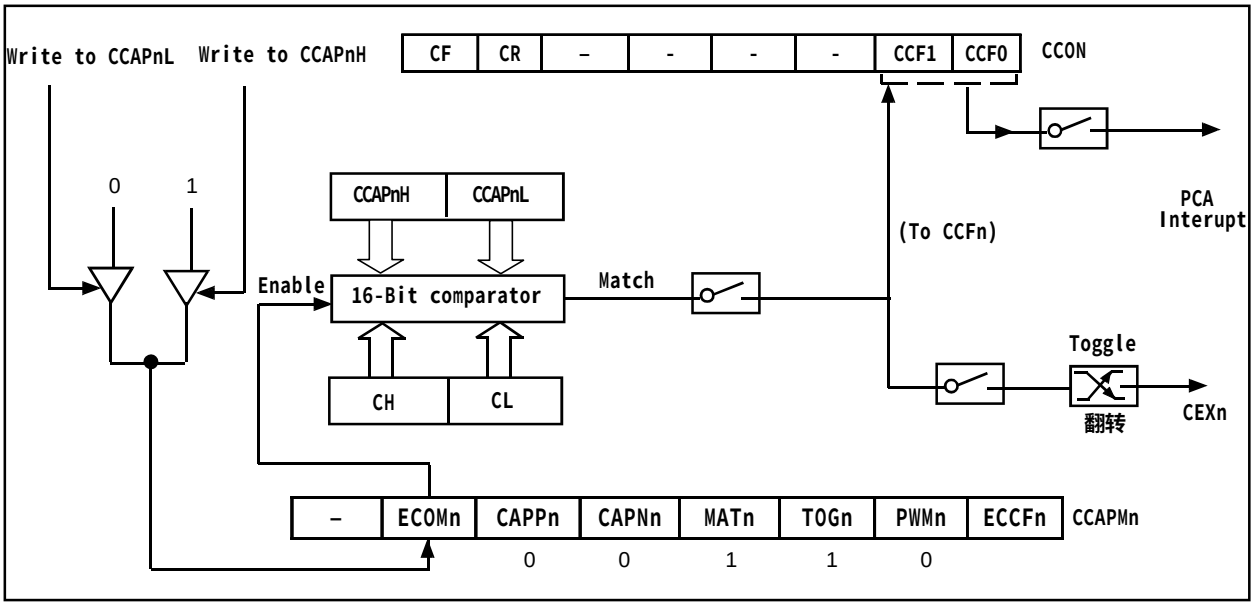
通过置位 CCAPMn 寄存器的 ECOM 和 MAT 位，可使 PCA 模块用作软件定时器（下图）。PCA 定时器的值与模块捕获寄存器的值相比较，当两者相等时，如果位 CCFn（在 CCON SFR 中）和位 ECCFn（在 CCAPMn SFR 中）都置位，将产生中断。



PCA Software Timer Mode/ 软件定时器模式 /PCA 比较模式

高速输出模式

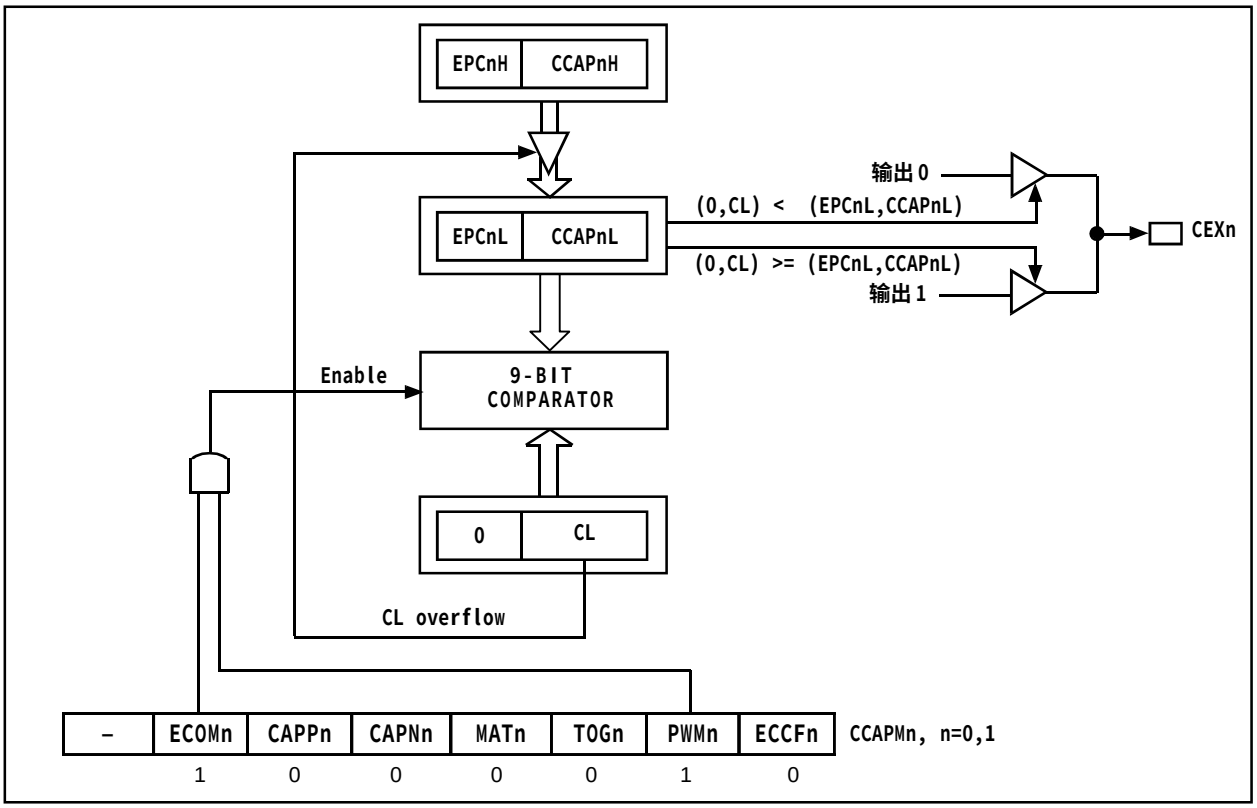
该模式中（下图），当 PCA 计数器的计数值与模块捕获寄存器的值相匹配时，PCA 模块的 CEXn 输出将发生翻转。要激活高速输出模式，模块 CCAPMn SFR 的 TOG,MAT 和 ECOM 位必须都置位。



PCA High-Speed Output Mode / PCA 高速输出模式

脉宽调节模式

所有 PCA 模块都可用作 PWM 输出（下图）。输出频率取决于 PCA 定时器的时钟源。



PCA PWM mode / 可调制脉冲宽度输出模式

由于所有模块共用仅有的 PCA 定时器，所有它们的输出频率相同。各个模块的输出占空比是独立变化的，与使用的捕获寄存器｛EPCnL，CCAPnL｝有关。当 CL SFR 的值小于｛EPCnL，CCAPnL｝时，输出为低，当 PCA CL SFR 的值等于或大于｛EPCnL，CCAPnL｝时，输出为高。当 CL 的值由 FF 变为 00 溢出时，｛EPCnH，CCAPnH｝的内容装载到｛EPCnL，CCAPnL｝中。这样就可实现无干扰地更新 PWM。要使能 PWM 模式，模块 CCAPMn 寄存器的 PWMn 和 ECOMn 位必须置位。

STC12C2052AD 系列单片机 PCA/PWM 新增特殊功能寄存器声明

```
;PCA_2052_SFR.ASM, PCA/PWM 新增特殊功能寄存器声明
;定义 STC12C2052 系列 MCU 特殊功能寄存器
EPCA      EQU    IE.6              ;PCA 中断屏蔽位。
CH        EQU    0F9H              ;PCA 计数器高 8 位。
CL        EQU    0E9H              ;PCA 计数器低 8 位。

;------------------------
CCON      EQU    0D8H              ;PCA 控制寄存器。
CCF0      EQU    CCON.0            ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU    CCON.1            ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CR        EQU    CCON.6            ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU    CCON.7            ;PCA 计数器溢出标志,由硬件或软件置位,必须由软件清 0。

;------------------------
CMOD      EQU    0D9H              ;PCA 工作模式寄存器。
;CMOD.7      CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2      CPS1: PCA 计数器脉冲源选择位 1。
;CMOD.1      CPS0: PCA 计数器脉冲源选择位 0。
;            CPS1   CPS0
;             0       0     内部时钟，fosc/12。
;             0       1     内部时钟，fosc/2。
;             1       0     Timer0 溢出。
;             1       1     由 ECI/P3.4 脚输入的外部时钟。

;CMOD.0      ECF: PCA 计数器溢出中断允许位，1--允许 CF(CCON.7) 产生中断。
;------------------------

CCAP0H    EQU    0FAH              ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU    0FBH              ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。

CCAP0L    EQU    0EAH              ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU    0EBH              ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。

;------------------------
PCA_PWM0    EQU    0F2H             ;PCA 模块 0 PWM 寄存器。
PCA_PWM1    EQU    0F3H             ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:     7       6       5       4       3       2       1       0
;              -       -       -       -       -       -     EPCnH   EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;------------------------
CCAPM0    EQU    0DAH              ;PCA 模块 0 的工作模式寄存器。
CCAPM1    EQU    0DBH              ;PCA 模块 1 的工作模式寄存器。
```

```
;CCAPMn:    7      6      5      4      3      2      1      0
;           -     ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn  =  1:允许比较功能。
;CAPPn  =  1:允许上升沿触发捕捉功能。
;CAPNn  =  1:允许下降沿触发捕捉功能。
;MATn   =  1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn   =  1:当匹配情况发生时，CEXn 将翻转。
;PWMn   =  1:将 CEXn 设置为 PWM 输出。
;ECCFn  =  1:允许 CCON 中的 CCFn 触发中断。

;   -   ECOMn  CAPPn  CAPNn  MATn  TOGn  PWMn  ECCFn
;   -     0      0      0      0     0     0     0    未启用任何功能。      00H
;   -     x      1      0      0     0     0     x    16 位 CEXn 上升沿触发捕捉功能。   20H
;   -     x      0      1      0     0     0     x    16 位 CEXn 下降沿触发捕捉功能。   10H
;   -     x      1      1      0     0     0     x    16 位 CEXn 边沿(上、下沿)触发捕捉功能。30H
;   -     1      0      0      1     0     0     x    16 位软件定时器。          48H
;   -     1      0      0      1     1     0     x    16 位高速脉冲输出。         4CH
;   -     1      0      0      0     0     1     0    8 位 PWM。              42H
```

```
;STC12C2052 系列单片机 PCA 功能  PWM 示例程序，使用 18.432MHz 晶振。
;-------------------------------------------------------------
#include <..\PCA_2052_SFR.ASM>     ;定义 PCA 特殊功能寄存器
;-------------------------------------------------------------

;定义常量
;pulse_width_MAX = pulse_width_MIN 时，输出脉冲宽度不变。
pulse_width_MAX   EQU   0F0H        ;PWM 脉宽最大值，占空比 = 93.75%
pulse_width_MIN   EQU   10H         ;PWM 脉宽最小值，占空比 = 6.25%
step              EQU   38H         ;PWM 脉宽变化步长
;-------------------------------------------------------------

;定义变量
pulse_width     EQU  30H
;-------------------------------------------------------------

    ORG 0000H
    AJMP  main
;-------------------------------------------------------------
    ORG 0050H
main:
    MOV   SP, #0E0H

    ACALL PCA_init
main_loop:
    ACALL PWM
    SJMP  main_loop

;-------------------------------------------------------------
PCA_init:
    MOV   CMOD, #80H;           ;PCA 在空闲模式下停止 PCA 计数器工作
                                ;PCA 时钟模式为 fosc/12
                                ;禁止 PCA 计数器溢出中断

    MOV   CCON, #00H            ;禁止 PCA 计数器工作，清除中断标志、计数器溢出标志
    MOV   CL, #00H              ;清 0 计数器
    MOV   CH, #00H
;-------------------------
;设置模块 0 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.7(第 11 脚)输出
    MOV   CCAPM0, #42H          ;*** 示例程序核心语句，---->0100,0010
    MOV   PCA_PWM0, #00H        ;*** 示例程序核心语句
;   MOV   PCA_PWM0, #03H        ;释放本行注释，PWM 输出就一直是 0，无脉冲。

;-------------------------
;设置模块 1 为 8 位 PWM 输出模式，PWM 无需中断支持。脉冲在 P3.5(第 9 脚)输出
    MOV   CCAPM1, #42H          ;*** 示例程序核心语句，---->0100,0010
    MOV   PCA_PWM1, #00H        ;*** 示例程序核心语句
```

```
;    MOV    PCA_PWM1, #03H          ;释放本行注释，PWM 输出就一直是 0，无脉冲。

     SETB   EPCA                    ;开 PCA 中断
     SETB   EA                      ;开总中断
     SETB   CR                      ;将 PCA 计数器打开
     RET

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

PWM:                                ;用示波器进行观察较为理想。

      ;逐渐变亮。
     MOV    A, #pulse_width_MIN     ;为输出脉冲宽度设置初值。
     MOV    pulse_width, A          ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop1:
     MOV    A, pulse_width          ;判是否到达最大值。
     CLR    C
     SUBB   A, #pulse_width_MAX
     JNC    PWM_a                   ;到达最大值就转到逐渐变暗。
      MOV   A, pulse_width          ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
     MOV    CCAP0H, A               ;*** 示例程序核心语句
     MOV    CCAP1H, A               ;*** 示例程序核心语句

     CPL    A                       ;用 P1 口的 LED 显示占空比，
     MOV    P1, A                   ;占空比 = ( pulse_width/256 ) * 100%  。

     MOV    A, pulse_width          ;计算下一次输出脉冲宽度数值。
     ADD    A, #step
     MOV    pulse_width, A
     ACALL  delay                   ;在一段时间内保持输出脉冲宽度不变。
     SJMP   PWM_loop1

PWM_a:
      ;逐渐变暗。
     MOV    A, #pulse_width_MAX     ;为输出脉冲宽度设置初值。
     MOV    pulse_width, A          ;pulse_width 数字越大脉宽越窄，P3.5 的 LED 越亮。
PWM_loop2:
     MOV    A, pulse_width          ;判是否到达最小值。
     CLR    C
     SUBB   A, #pulse_width_MIN
     JC     PWM_b                   ;到达最小值就返回。
     JZ     PWM_b                   ;到达最小值就返回。
      MOV   A, pulse_width          ;设置脉冲宽度。数字越大、脉宽越窄、LED 越亮。
     MOV    CCAP0H, A               ;*** 示例程序核心语句
     MOV    CCAP1H, A               ;*** 示例程序核心语句

     CPL    A                       ;用 P1 口的 LED 显示占空比，
```

```
    MOV   P1, A                ;占空比 = ( pulse_width/256 ) * 100%  。

    MOV   A, pulse_width       ;计算下一次输出脉冲宽度数值。
    CLR   C
    SUBB  A, #step
    MOV   pulse_width, A
    ACALL delay                ;在一段时间内保持输出脉冲宽度不变。
    SJMP  PWM_loop2
PWM_b:
    RET
;-------------------------------------------------------------

delay:
    CLR   A
    MOV   R1, A
    MOV   R2, A
    MOV   R3, #80H
delay_loop:
    NOP
    NOP
    NOP
    DJNZ  R1, delay_loop
    DJNZ  R2, delay_loop
    DJNZ  R3, delay_loop
    RET
;-------------------------------------------------------------
    END
```

STC12C2052AD 系列单片机 PWM 输出 C 语言示例

PWM 输出 C 语言示例

```
#include<reg52.h>
sfr  CCON = 0xD8;
sfr  CMOD = 0xD9;
sfr  CCAP0L = 0xEA;
sfr  CCAP0H = 0xFA;
sfr  CCAPM0 = 0xDA;
sfr  CCAPM1 = 0xDB;
sbit  CR = 0xDE;
void main(void)
{
    CMOD = 0x02; // Setup PCA timer
    CL = 0x00;
    CH = 0x00;
    CCAP0L = 0xc0;  //Set the initial value same as CCAP0H
    CCAP0H = 0xc0;  //25% Duty Cycle
    CCAPM0 = 0x42;  //0100,0010  Setup PCA module 0 in PWM mode
    CR = 1; //Start PCA Timer.
    while(1){};
 }
```

STC12C2052 系列单片机 PCA 的高速脉冲输出

```
;*******************************************************************************
;                  输出 125.0KHz 的脉冲(晶体频率 = 33.000MHz)
;
;示例程序: 使用 功能，在 P3.5(第 9 脚)输出
;          125.0KHz 的方脉冲。
;------------------------------------------------------------------------------
;    程序中定义的常量 CCAPnL_Value 决定了 PCA 模块 n 输出脉冲的频率 f：
;        f =  Fosc / (4 * CCAPnL_Value )
;           式中 Fosc = 晶体频率
;        CCAPnL_Value  =  Fosc / (4 * f)
;
;    如算出的结果不是整数，则进行取整 CCAPnL_Value =  INT(Fosc / (4 * f) + 0.5)
;         INT() 为取整数运算，直接去掉小数。
;*******************************************************************************
;定义 STC12C2052 系列 MCU 特殊功能寄存器
IPH       EQU   0B7H              ;中断优先级高位寄存器

EPCA_LVD  EQU   IE.6              ;PCA/LVD 中断允许位。
                  ;要打开 PCA 中断还要打开相应的 ECF，ECCF0，ECCF1 位
                  ;要打开 LVD 中断还要打开相应的 ELVDI 位
CH        EQU   0xF9              ;PCA 计数器高 8 位。
CL        EQU   0xE9              ;PCA 计数器低 8 位。

;------------------------
CCON      EQU   0D8H              ;PCA 控制寄存器。
CCF0      EQU   CCON.0            ;PCA 模块 0 中断标志，由硬件置位，必须由软件清 0。
CCF1      EQU   CCON.1            ;PCA 模块 1 中断标志，由硬件置位，必须由软件清 0。
CR        EQU   CCON.6            ;1:允许 PCA 计数器计数，必须由软件清 0。
CF        EQU   CCON.7            ;PCA 计数器溢出标志,由硬件或软件置位,必须由软件清 0。

;------------------------
CMOD      EQU   0D9H              ;PCA 工作模式寄存器。
;CMOD.7      CIDL: idle 状态时 PCA 计数器是否继续计数，0: 继续计数，1: 停止计数。

;CMOD.2      CPS1: PCA 计数器脉冲源选择位 1。
;CMOD.1      CPS0: PCA 计数器脉冲源选择位 0。
;           CPS1   CPS0
;            0      0     内部时钟，fosc/12。
;            0      1     内部时钟，fosc/2。
;            1      0     Timer0 溢出。
;            1      1     由 ECI/P3.4 脚输入的外部时钟。

;CMOD.0      ECF: PCA 计数器溢出中断允许位，1-- 允许 CF(CCON.7) 产生中断。
;------------------------
```

```
CCAP0H    EQU    0FAH                ;PCA 模块 0 的捕捉 / 比较寄存器高 8 位。
CCAP1H    EQU    0FBH                ;PCA 模块 1 的捕捉 / 比较寄存器高 8 位。
CCAP0L    EQU    0EAH                ;PCA 模块 0 的捕捉 / 比较寄存器低 8 位。
CCAP1L    EQU    0EBH                ;PCA 模块 1 的捕捉 / 比较寄存器低 8 位。

;------------------------
PCA_PWM0 EQU    0F2H                ;PCA 模块 0 PWM 寄存器。
PCA_PWM1 EQU    0F3H                ;PCA 模块 1 PWM 寄存器。

;PCA_PWMn:    7      6      5      4      3      2      1      0
;             -      -      -      -      -      -    EPCnH  EPCnL
;B7-B2: 保留
;B1(EPCnH): 在 PWM 模式下，与 CCAPnH 组成 9 位数。
;B0(EPCnL): 在 PWM 模式下，与 CCAPnL 组成 9 位数。

;------------------------
CCAPM0    EQU    0DAH               ;PCA 模块 0 的工作模式寄存器。
CCAPM1    EQU    0DBH               ;PCA 模块 1 的工作模式寄存器。

;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;
;ECOMn = 1:允许比较功能。
;CAPPn = 1:允许上升沿触发捕捉功能。
;CAPNn = 1:允许下降沿触发捕捉功能。
;MATn  = 1:当匹配情况发生时，允许 CCON 中的 CCFn 置位。
;TOGn  = 1:当匹配情况发生时，CEXn 将翻转。
;PWMn  = 1:将 CEXn 设置为 PWM 输出。
;ECCFn = 1:允许 CCON 中的 CCFn 触发中断。

;ECOMn CAPPn CAPNn MATn TOGn PWMn ECCFn
;  0     0     0    0    0    0    0   0x00   未启用任何功能。
;  x     1     0    0    0    0    x   0x21   16 位 CEXn 上升沿触发捕捉功能。
;  x     0     1    0    0    0    x   0x11   16 位 CEXn 下降沿触发捕捉功能。
;  x     1     1    0    0    0    x   0x31   16 位 CEXn 边沿(上、下沿)触发捕捉功能。
;  1     0     0    1    0    0    x   0x49   16 位软件定时器。
;  1     0     0    1    1    0    x   0x4d   16 位高速脉冲输出。
;  1     0     0    0    0    1    0   0x42   8 位 PWM。

;------------------------------------------------------------
;定义常量 CCAPnL_Value
;CCAPnL_Value 决定了模块 1 输出脉冲的频率 f：
;         f =  Fosc / (4 * CCAPnL_Value )
;    式中 Fosc = 晶体频率
;    或  CCAPnL_Value  =  INT(Fosc / (4 * f) + 0.5)
;        INT() 为取整数运算。
;
```

```
;     假定 fosc = 20MHz时, 要求 PCA 高速脉冲输出 125KHz 的方波:
;         CCAPnL_Value = INT( 20000000/4/125000 + 0.5)
;                      = INT( 40 + 0.5)
;                      = INT( 40.5 )
;                      = 40
;                      = 28H
;     输出脉冲的频率 f  = 20000000/4/40
;                      = 125000 (125.0KHz)

;CCAPnL_Value  EQU 25H      ;25H = 37, fosc = 18.432MHz时, 高速脉冲输出 = 124.540KHz
;CCAPnL_Value  EQU 28H      ;28H = 40, fosc = 20MHz时, 高速脉冲输出 = 125KHz
CCAPnL_Value   EQU 42H      ;42H = 66, fosc = 33MHz时, 高速脉冲输出 = 125KHz
;-------------------------------------------------------------
    ORG 0000H
    AJMP  main
;-------------------------------------------------------------
    ORG  0033H                  ;interrupt 6
PCA_interrupt:
    PUSH  ACC                   ;4 Clock
    PUSH  PSW                   ;4 Clock

    CLR   CCF1                  ;1 Clock, 清 PCA 模块 1 中断标志

    MOV   A, #CCAPnL_Value ;2 Clock
    ADD   A, CCAP1L             ;3 Clock
    MOV   CCAP1L, A             ;3 Clock
    CLR   A                     ;1 Clock
    ADDC  A, CCAP1H             ;3 Clock
    MOV   CCAP1H, A             ;3 Clock

    POP   PSW                   ;3 Clock
    POP   ACC                   ;3 Clock
    RETI                        ;4 Clock

;此中断服务程序共用 34 Clock, 进入中断服务程序还要数个 Clock
;-------------------------------------------------------------
    ORG 0060H
main:
    MOV   SP, #0E0H             ;设置堆栈指针
    ACALL PCA_init              ;调用 PCA 初始化程序

main_loop:
    NOP
    NOP
    NOP
    SJMP  main_loop
;-------------------------------------------------------------
```

```
PCA_init:                          ;PCA 初始化程序
    MOV   CMOD, #00000010B         ;02H，PCA 计数器在空闲模式下继续工作，CIDL = 0
                                   ;PCA 计数器计数脉冲来源为系统时钟源 fosc/2，CPS1，CPS0 = (0,1)
                                   ;禁止 PCA 计数器(CH，CL)计数溢出(CH，CL=0000H)中断，ECF = 0
    MOV   CCON, #00H               ;清除 PCA 计数器(CH，CL)计数溢出中断标志，CF = 0
                                   ;停止 PCA 计数器(CH，CL)计数，CR = 0
                                   ;清除 模块 1 中断标志，CCF1 = 0
                                   ;清除 模块 0 中断标志，CCF0 = 0
    MOV   CH, #00H                 ;清 0 PCA 计数器高 8 位
    MOV   CL, #00H                 ;清 0 PCA 计数器低 8 位
;-------------------------
;设置模块 1 为高速脉冲输出模式，脉冲在 P3.5(第 9 脚)输出
    MOV   CCAPM1, #01001101B       ;4DH，设置 PCA 模块 1 为高速脉冲输出模式,允许触发中断
;CCAPMn:    7      6      5      4      3      2      1      0
;           -    ECOMn  CAPPn  CAPNn  MATn   TOGn   PWMn   ECCFn
;           0      1      0      0      1      1      0      1

    MOV   CCAP1L, #CCAPnL_Value ;给模块 1 置初值，此句不可少
    MOV   CCAP1H, #0 ;给模块 1 置初值，此句不可少

     ;其它中断服务可能会使模块 1  高速脉冲输出的某个周期突然变得很大，因此必须将
     ;PCA 中断的优先级设置为唯一的最高级，其它中断的优先级都要比它低。
    MOV   IPH, #01000000B          ;PCA 中断的优先级设置为唯一的最高级
    MOV   IP, #01000000B

    SETB  EPCA_LVD                 ;开 PCA 中断
    SETB  EA                       ;开总中断
    SETB  CR                       ;将 PCA 计数器打开
    RET

;-------------------------------------------------------------
    END
;-------------------------------------------------------------
```

附录 B ：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

3.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：STC12C2052AD 系列 ISP 经济型下载编程工具(人民币 50 元，可申请免费样品)

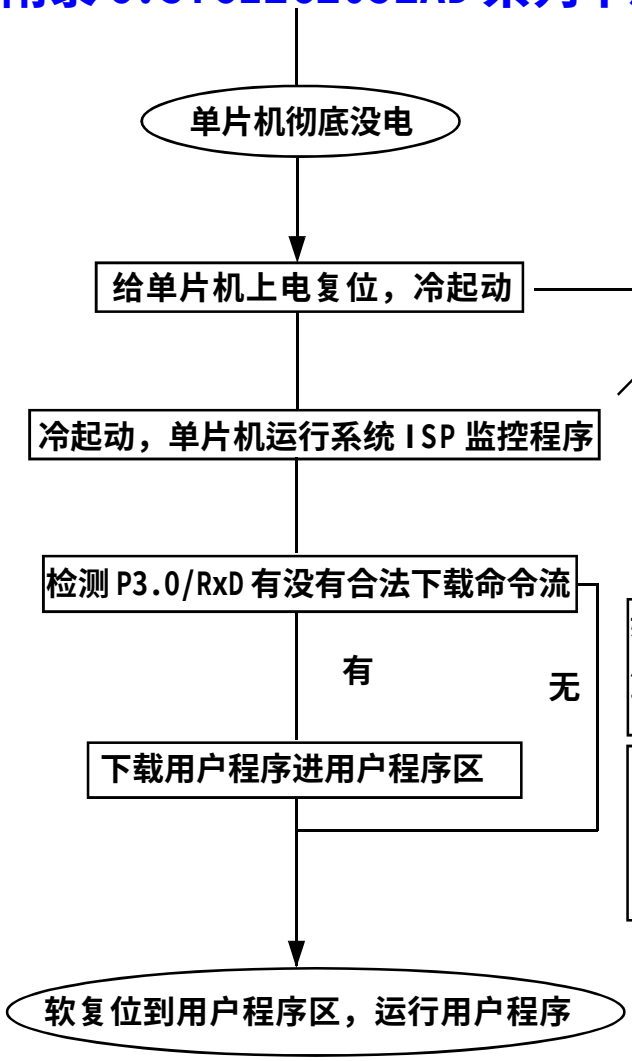
仿真器:如您已有老的仿真器，可仿真普通 8052 的基本功能

STC12C2052AD 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

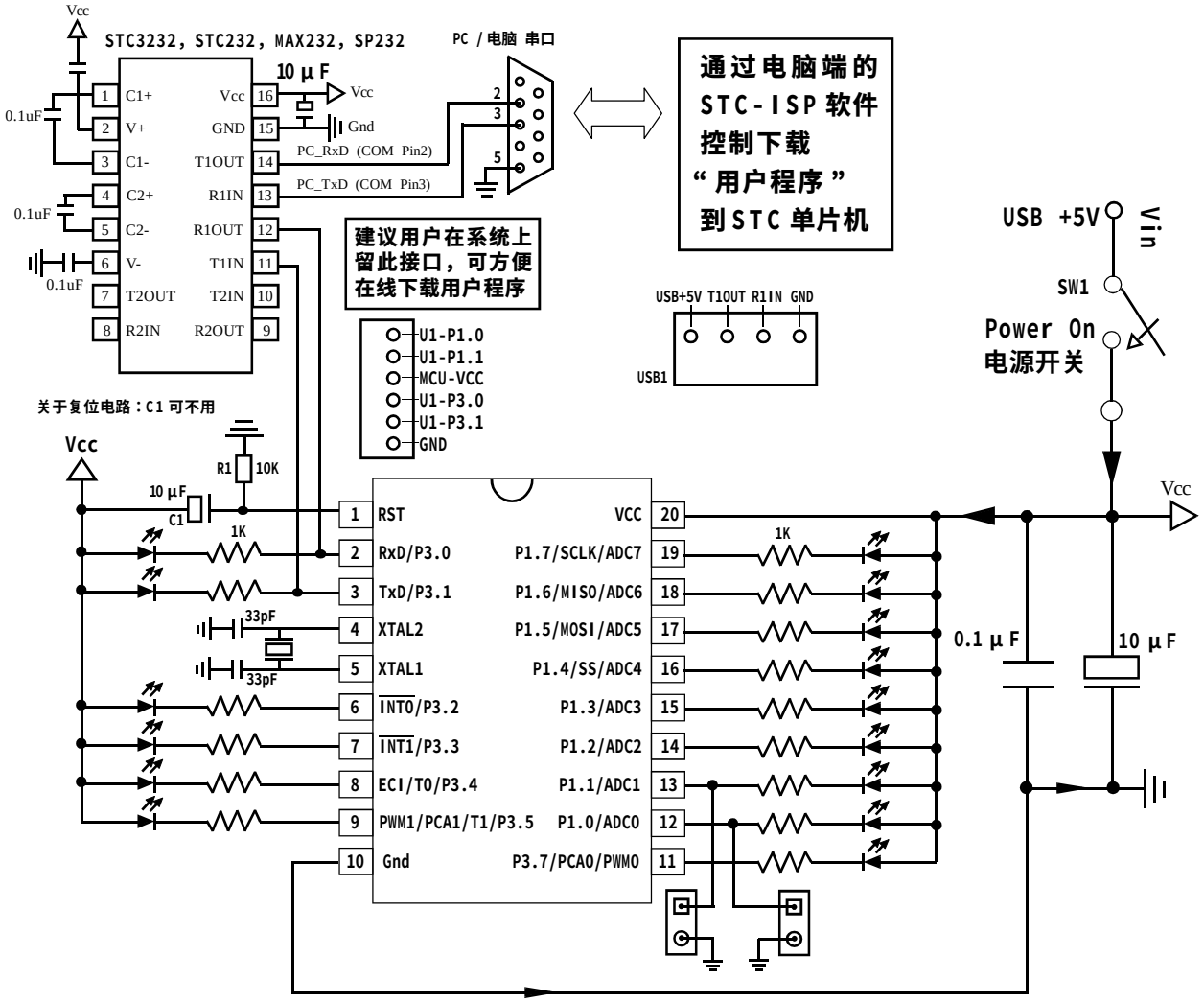
无须添加新的设备

附录 C:STC12C2052AD 系列单片机 ISP 编程原理 使用说明



STC 12C2052AD 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



STC12C2052AD 系列单片机具有在系统可编程（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。由于可以在用户的目标系统上将程序直接下载进单片机看运行结果对错，故无须仿真器。

STC12 系列单片机内部固化有 ISP 系统引导固件，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，2 秒到 3 秒一片)。

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。

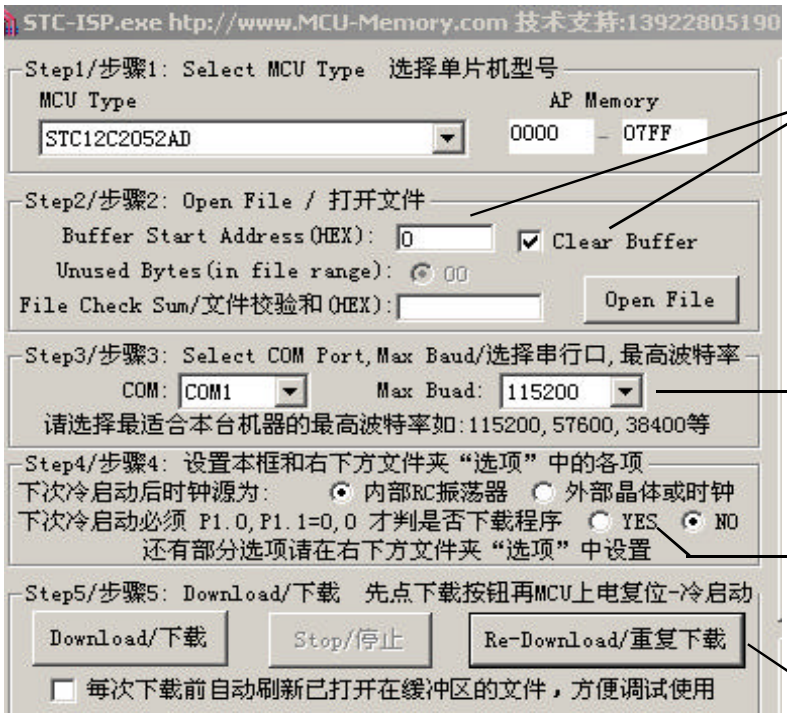
(2). 使用 STC-ISP 下载工具（软件),请随时更新，目前已到 Ver3.0 Beta 版本以上,支持 *.bin, *.hex(Intel 16 进制格式)文件。STC12C2052AD 系列单片机的底层 ISP 固件为 Ver3.3D(支持 EEPROM 功能)。老版本 Ver3.2D 部分不支持 EEPROM 功能。

请随时注意升级 PC（电脑）端的 ISP 程序。

单片机底层 ISP 固件版本为 Ver3.3D 的单片机，电脑端的 ISP 程序应用 Ver3.0 Beta 以上

(3).STC12C2052 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，如从 P3.0/RxD 检测到合法的下载命令流就下载用户程序，如检测不到就系统复位到用户程序区。

(4).如果用户板上 P3.0/RxD，P3.1/Txd 接了 RS-485 等电路，下载时需要将其断开。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。



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

如可将要写入 EEPROM 区的数据调入从缓冲区 1000H 开始的地方，并不清缓冲区，然后和应用程序一起写入

用户根据需要限制最高通信波特率，如 57600，38400，19200

如 P3.0/P3.1 外接 RS-485 总线等，建议选择如 P1.0/P1.1 不同时等于 0.0，则直接运行用户程序，跨过系统 ISP 引导程序

大批量生产时使用

Step1/ 步骤 1：选择你所使用的单片机型号，如 STC12C2052,STC12C4052AD 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：暂时无效 。但您应在选项里选择是用内部 R/C 振荡器做时钟还是不是

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。

一定要先选择“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，先上电，检测不到合法的下载命令流，单片机就直接跑用户程序了。

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). 如果您的系统 P3.0/P3.1 连接到 RS-485 电路，推荐

在选项里选则“下次冷启动需要 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”

这样冷启动后如 P1.0，P1.1 不同时 0,单片机直接运行用户程序，免得由于 RS-485 总线上的乱码造成单片机反复判断乱码是否为合法，浪费几百 mS 的时间

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

STC232/MAX232/SP232 尽量选用 SOP 封装(窄体)，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 3.0A.PCB 演示板作为编程工具

STC-ISP Ver 3.0APCB 板如焊接的是 STC12C2052AD 的线路，则
可完成 STC12C2052AD 系列单片机的 ISP 下载编程 / 烧录用户程序的功能。

在 STC-ISP Ver 3.0A 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.)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 3.0A PCB 板 USB1 插座用于 RS-232 通信和供电,此时 USB +5V Power 灯亮(D43,USB 接口有电)

(3.)其他插座不需连接

(4.)SW1 开关处于非按下状态，此时 MCU-VCC Power 灯不亮(D41)，没有给单片机通电

(5.)SW3 开关

处于非按下状态，P1.0，P1.1 = 1，1,不短接到地。

处于按下状态， P1.0，P1.1 = 0，0,短接到地。

如果单片机已被设成“下次冷启动 P1.0/P1.1 = 0,0 才判 P3.0/RxD 有无合法下载命令流”
就必须将 SW3 开关处于按下状态，让单片机的 P1.0/P1.1 短接到地

(6.)将单片机插进 U1-Socket 锁紧座，锁紧单片机,注意单片机是 20-PIN,而 U1-Socket 锁紧座是 40-PIN,我们的设计是靠下插，靠近晶体的那一端插。

(7.)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)

(8.)按下 SW1 开关，给单片机上电复位，此时 MCU-VCC Power 灯亮(D41)

此时 STC 单片机进入 ISP 模式(STC12C2052AD 系列冷启动进入 ISP)

(9.)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(D41)，给单片机断电，取下单片机。

利用 STC-ISP Ver 3.0A 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(如果需要的话)
4. 如须 P1.0，P1.1 = 0，0,短接到地，可在用户系统上将其短接到地，或将 P1.0/P1.1 也从用户系统上引到 STC-ISP Ver3.0A PCB 板上，将 SW3 开关按下，则 P1.0/P1.1=0,0。
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,如有，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,或其他线路，在下载时应将其断开。

附录 D: 内部数据 RAM 存储器

内部数据 RAM 存储器

STC12C2052AD 系列单片机内部有 256 字节常规的 RAM。

器件的内部常规数据存储器由 3 部分组成：

1.低 128 字节 RAM（00H～7FH），可直接和间接寻址，用“MOV”和“MOV @Ri”

2.高 128 字节 RAM（80H～FFH），间接寻址，用“MOV @Ri”

3.特殊功能寄存器（80H～FFH），只可直接寻址， 用“MOV”

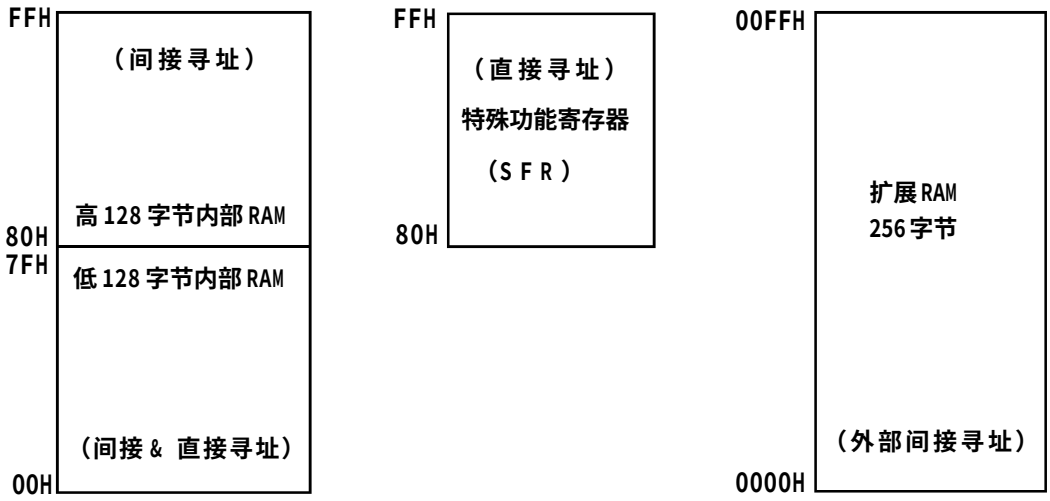
由于高 128 字节 RAM 和 SFR（特殊功能寄存器）占用相同的地址，因此高 128 字节 RAM 空间必须用间接寻址（MOV @Ri）来区分。特殊功能寄存器（80H～FFH），只可直接寻址（用“MOV”）来区分。尽管 RAM 和 SFR 的地址相同，但它们在物理上是独立的。

扩展数据 RAM

STC12C2052AD 系列下一代产品 STC12C5410AD 系列会有 256 字节的扩展 RAM，称其为 XRAM（附加 RAM），用“MOVX”寻址。

扩展的 256 字节 RAM（0000H～00FFH），通过 MOVX 指令间接寻址。

使用“MOVX @DPTR” / “MOVX @Ri”



内部常规 256 字节 RAM 间接寻址测试程序

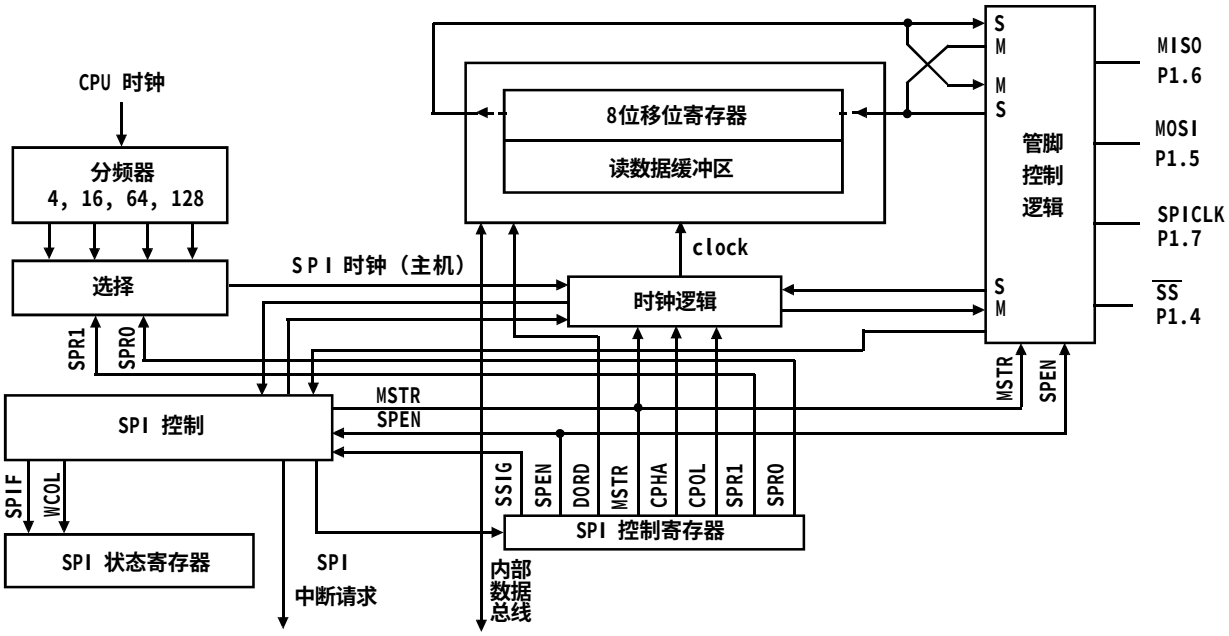
```
TEST_CONST EQU  5AH
;TEST_RAM  EQU  03H

     ORG  0000H
     LJMP INITIAL

     ORG  0050H
INITIAL:


     MOV  R0,  #253

     MOV  R1,  #3H
TEST_ALL_RAM:
     MOV  R2,  #0FFH
TEST_ONE_RAM:
     MOV  A,   R2
     MOV  @R1, A
     CLR  A
     MOV  A,   @R1

     CJNE A,   2H,  ERROR_DISPLAY
     DJNZ R2,  TEST_ONE_RAM
     INC  R1
     DJNZ R0,  TEST_ALL_RAM

OK_DISPLAY:
     MOV  P1,  #11111110B
Wait1:
     SJMP Wait1

ERROR_DISPLAY:
     MOV  A,   R1
     MOV  P1,  A
Wait2:
     SJMP Wait2

     END
```

附录 E： 串行外围接口（SPI）

STC12C2052AD 系列单片机还提供另一种高速串行通信接口——SPI 接口。SPI 是一种全双工、高速、同步的通信总线，有两种操作模式：主模式和从模式。在主模式中支持高达 3Mbit/s 的速率(工作频率为 12MHz 时,如果 CPU 主频采用 20MHz 到 36MHz,则可更高，从模式时速度无法太快，越慢越好),还具有传输完成标志和写冲突标志保护。

STC12C2052AD 系列 8051 单片机 SPI 功能模块特殊功能寄存器 SPI Management SFRs

Mnemonic	Add	Name	7	6	5	4	3	2	1	0	Reset value
SPCTL	85h	SPI Control Register	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	0000,0000
SPSTAT	84h	SPI Status Register	SPIF	WCOL	-	-	-	-	-	-	00xx,xxxx
SPDAT	86h	SPI Data Register									0000,0000



SPI 方框图

SPI 接口有 4 个管脚：SPICLK/P1.7，MOSI/P1.5，MISO/P1.6 和 $\overline{SS}$/P1.4。

- SPICLK，MOSI 和 MISO 通常和两个或更多 SPI 器件连接在一起。数据通过 MOSI 由主机传送到从机，通过 MISO 由从机传送到主机。SPICLK 信号在主模式时为输出，在从模式时为输入。如果 SPI 系统被禁止，即 SPEN (SPCTL.6)=0(复位值)，这些管脚都可作为 I/O 口使用。
- /SS 为从机选择管脚。在典型的配置中，SPI 主机使用 I/O 口选择一个 SPI 器件作为当前的从机。SPI 从器件通过其 /SS 脚确定是否被选择。如果满足下面的条件之一，/SS 就被忽略：
  - －如果 SPI 系统被禁止，即 SPEN(SPCTL.6)=0(复位值)
  - －如果 SPI 配置为主机，即 MSTR(SPCTL.4)=1，并且 P1.4 配置为输出（通过 P1M0.4 和 P1M1.4）
  - －如果 /SS 脚被忽略，即 SSIG(SPCTL.7)位＝1，该脚配置用于 I/O 口功能。

注：即使 SPI 被配置为主机（MSTR ＝ 1），它仍然可以通过拉低 /SS 脚配置为从机（如果 P1.4 配置为输入且 SSIG=0)。要使能该特性，应当置位 SPIF(SPSTAT.7)。

典型连接如 SPI 图 1～3 所示。

SPI 控制寄存器的位分配（SPCTL - 地址：85h）

位	7	6	5	4	3	2	1	0
符 号	SSIG	SPEN	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
复 位	0	0	0	0	0	1	0	0

SPI 控制寄存器的位描述（SPCTL – 地址：85h）

位	符号	描 述
0	SPR0	SPR0/SPR1是SPI 时钟速率选择控制位。
1	SPR1	SPR1，SPR0： 0 0 －CPU_CLK/4 0 1 －CPU_CLK/16 1 0 －CPU_CLK/64 1 1 －CPU_CLK/128
2	CPHA	SPI 时钟相位选择（见SPI图4～图7）： 1：数据在SPICLK 的前时钟沿驱动，并在后时钟沿采样。 0：数据在/SS 为低（SSIG＝00）时被驱动，在SPICLK 的后时钟沿被改变，并在前时钟沿被采样。 （注：SSIG=1 时的操作未定义）
3	CPOL	SPI 时钟极性（见SPI图4～图7）： 1：SPICLK 空闲时为高电平。SPICLK 的前时钟沿为下降沿而后沿为上升沿。 0：SPICLK 空闲时为低电平。SPICLK 的前时钟沿为上升沿而后沿为下降沿。
4	MSTR	主/从模式选择（见SPI 主从选择表）。
5	DORD	SPI 数据顺序： 1：数据字的LSB(最低位) 最先发送；0：数据字的MSB(最高位) 最先发送。
3	SPEN	SPI 使能。 1：SPI 使能。 0：SPI 被禁止，所有SPI 管脚都作为I/O 口使用。
7	SSIG	/SS 忽略。 1：MSTR（位4）确定器件为主机还是从机。 0：/SS 脚用于确定器件为主机还是从机。/SS 脚可作为I/O 口使用（见SPI 主从选择表）。

SPI 状态寄存器的位分配（SPSTAT – 地址：84h）

位	7	6	5	4	3	2	1	0
符号	SPIF	WCOL	-	-	-	-	-	-
复位	0	0	X	X	X	X	X	X

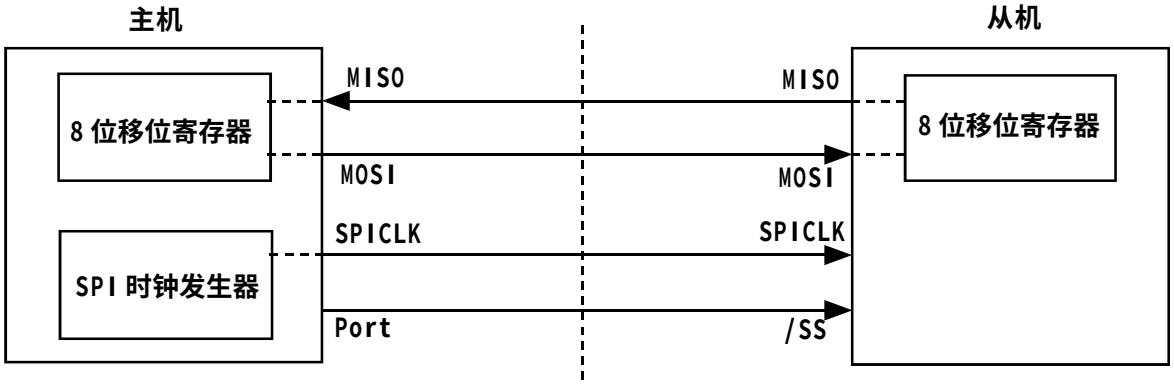
SPI 状态寄存器的位描述（SPSTAT – 地址：84h）

位	符号	符号
7	SPIF	SPI 传输完成标志。当一次串行传输完成时，SPIF 置位，并当ESPI 和EA 都置位时产生中断。当SPI 处于主模式且SSIG=0 时，如果/SS 为输入并被驱动为低电平，SPIF 也将置位。SPIF标志通过软件向其写入“1”清零。
6	WCOL	SPI 写冲突标志。在数据传输的过程中如果对SPI 数据寄存器SPDAT 执行写操作，WCOL 将置位。WCOL 标志通过软件向其写入“1”清零。
5 - 0	-	保留

SPI 数据寄存器的位分配（SPDAT – 地址：86h）

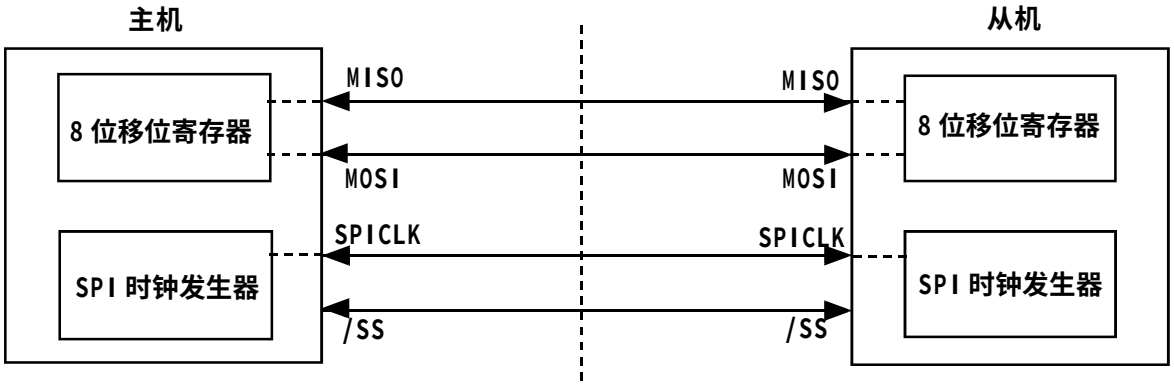
位	7	6	5	4	3	2	1	0
符 号	MSB							LSB
复 位	0	0	0	0	0	0	0	0

SPDAT.7 - SPDAT.0: 传输的数据位 Bit7～Bit0



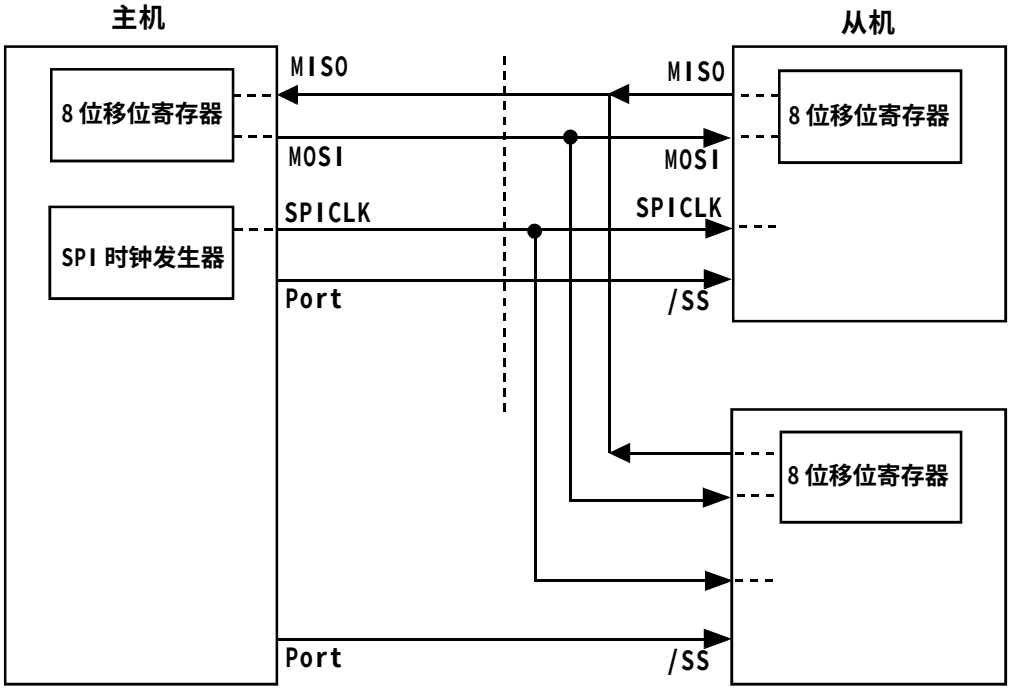
SPI 图 1 SPI 单主机 - 单从机 配置

在上图 SPI 图 1 中，从机的 SSIG(SPCTL.7)为 0，/SS 用于选择从机。SPI 主机可使用任何端口（包括 P1.4/$\overline{SS}$）来驱动 /SS 脚。



SPI 图 2 SPI 双器件配置（可器件可互为主从）

上图 SPI 图 2 所示为两个器件互为主从的情况。当没有发生 SPI 操作时，两个器件都可配置为主机(MSTR=1)，将 SSIG 清零并将 P1.4(/SS)配置为准双向模式。当其中一个器件启动传输时，它可将 P1.4 配置为输出并驱动为低电平，这样就强制另一个器件变为从机。



SPI 图 3 SPI 单主机 - 多从机 配置

在上图 SPI 图 3 中，从机的 SSIG(SPCTL.7)为 0，从机通过对应的 /SS 信号被选中。SPI 主机可使用任何端口(包括 P1.4/$\overline{SS}$）来驱动 /SS 脚。

对 SPI 进行配置

下表 所示为主 / 从模式的配置以及模式的使用和传输方向。

SPI 主从模式选择

SPEN	SSIG	/SS 脚 P1.4	MSTR	主或从模式	MISO P1.6	MOSI P1.5	SPICLK P1.7	备注
0	X	P1.4	X	SPI 功能禁止	P1.6	P1.5	P1.7	SPI 禁止。P1.4/P1.5/P1.6/P1.7作为普通I/O口使用
1	0	0	0	从机模式	输出	输入	输入	选择作为从机
1	0	1	0	从机模式未被选中	高阻	输入	输入	未被选中。MISO 为高阻状态，以避免总线冲突
1	0	0	1—>0	从机模式	输出	输入	输入	P1.4/ SS 配置为输入或准双向口。SSIG 为0。如果择/SS 被驱动为低电平，则被选择作为从机。当SS 变为低电平时，MSTR将清零。 注：当/SS处于输入模式时，如被驱动为低电平且SSIG=0 时，MSTR 位自动清零。
1	0	1	1	主(空闲)	输入	高阻	高阻	当主机空闲时MOSI 和SPICLK 为高阻态以避免总线冲突。用户必须将SPICLK 上拉或下拉（根据CPOL-SPCTL.3 的取值）以避免SPICLK出现悬浮状态。
				主(激活)		输出	输出	作为主机激活时，MOSI 和SPICLK 为推挽输出
1	1	P1.4	0	从	输出	输入	输入	
1	1	P1.4	1	主	输入	输出	输出	

作为从机时的额外注意事项

当 CPHA ＝ 0 时，SSIG 必须为 0，/SS 脚必须取反并且在每个连续的串行字节之间重新设置为高电平。如果 SPDAT 寄存器在 / SS 有效（低电平）时执行写操作，那么将导致一个写冲突错误。CPHA=0 且 SSIG=0 时的操作未定义。

当 CPHA ＝ 1 时，SSIG 可以置位。如果 SSIG ＝ 0，/SS 脚可在连续传输之间保持低有效（即一直固定为低电平）。这种方式有时适用于具有单固定主机和单从机驱动 MISO 数据线的系统。

作为主机时的额外注意事项

在 SPI 中，传输总是由主机启动的。如果 SPI 使能（SPEN=1）并选择作为主机，主机对 SPI 数据寄存器的写操作将启动 SPI 时钟发生器和数据的传输。在数据写入 SPDAT 之后的半个到一个 SPI 位时间后，数据将出现在 MOSI 脚。

需要注意的是，主机可以通过将对应器件的 /SS 脚驱动为低电平实现与之通信。写入主机 SPDAT 寄存器的数据从 MOSI 脚移出发送到从机的 MOSI 脚。同时从机 SPDAT 寄存器的数据从 MISO 脚移出发送到主机的 MISO 脚。

传输完一个字节后，SPI 时钟发生器停止，传输完成标志（SPIF）置位并产生一个中断（如果 SPI 中断使能）。主机和从机 CPU 的两个移位寄存器可以看作是一个 16 循环移位寄存器。当数据从主机移位传送到从机的同时，数据也以相反的方向移入。这意味着在一个移位周期中，主机和从机的数据相互交换。

通过 /SS 改变模式

如果 SPEN=1， SSIG=0 且 MSTR=1，SPI 使能为主机模式。/SS 脚可配置为输入或准双向模式。这种情况下，另外一个主机可将该脚驱动为低电平，从而将该器件选择为 SPI 从机并向其发送数据。

为了避免争夺总线，SPI 系统执行以下动作:

1)MSTR 清零并且 CPU 变成从机。这样 SPI 就变成从机。MOSI 和 SPICLK 强制变为输入模式，而 MISO 则变为输出模式。

2）SPSTAT 的 SPIF 标志位置位。如果 SPI 中断已被使能，则产生 SPI 中断。

用户软件必须一直对 MSTR 位进行检测，如果该位被一个从机选择所清零而用户想继续将 SPI 作为主机，这时就必须重新置位 MSTR，否则就进入从机模式。

写冲突

SPI 在发送时为单缓冲，在接收时为双缓冲。这样在前一次发送尚未完成之前，不能将新的数据写入移位寄存器。当发送过程中对数据寄存器进行写操作时，WCOL 位（SPSTAT.6）将置位以指示数据冲突。在这种情况下，当前发送的数据继续发送，而新写入的数据将丢失。

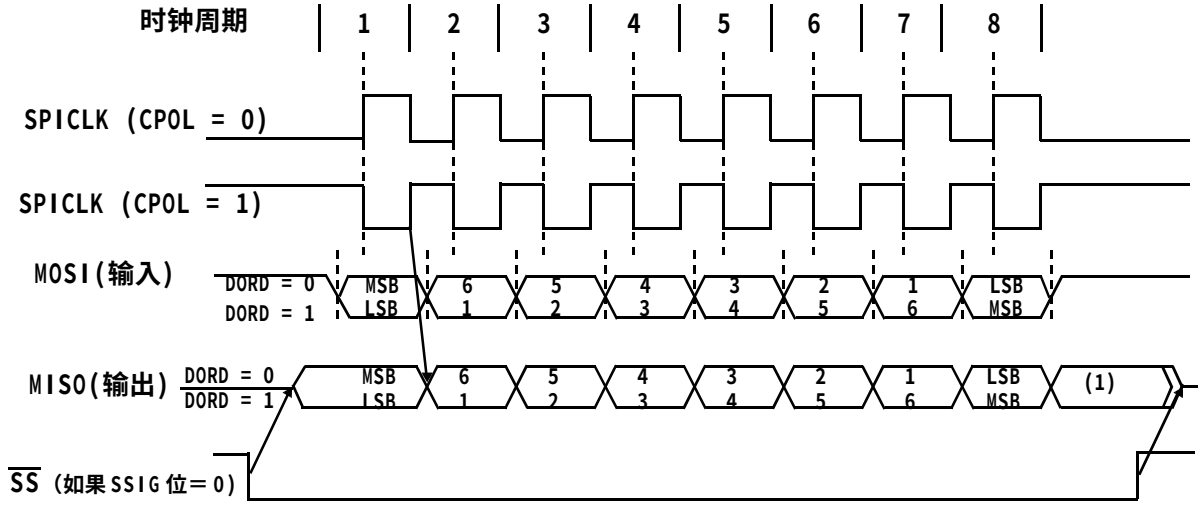
当对主机或从机进行写冲突检测时，主机发生写冲突的情况是很罕见的，因为主机拥有数据传输的完全控制权。但从机有可能发生写冲突，因为当主机启动传输时，从机无法进行控制。

接收数据时，接收到的数据传送到一个并行读数据缓冲区，这样将释放移位寄存器以进行下一个数据的接收。但必须在下个字符完全移入之前从数据寄存器中读出接收到的数据，否则，前一个接收数据将丢失。

WCOL 可通过软件向其写入“1”清零。

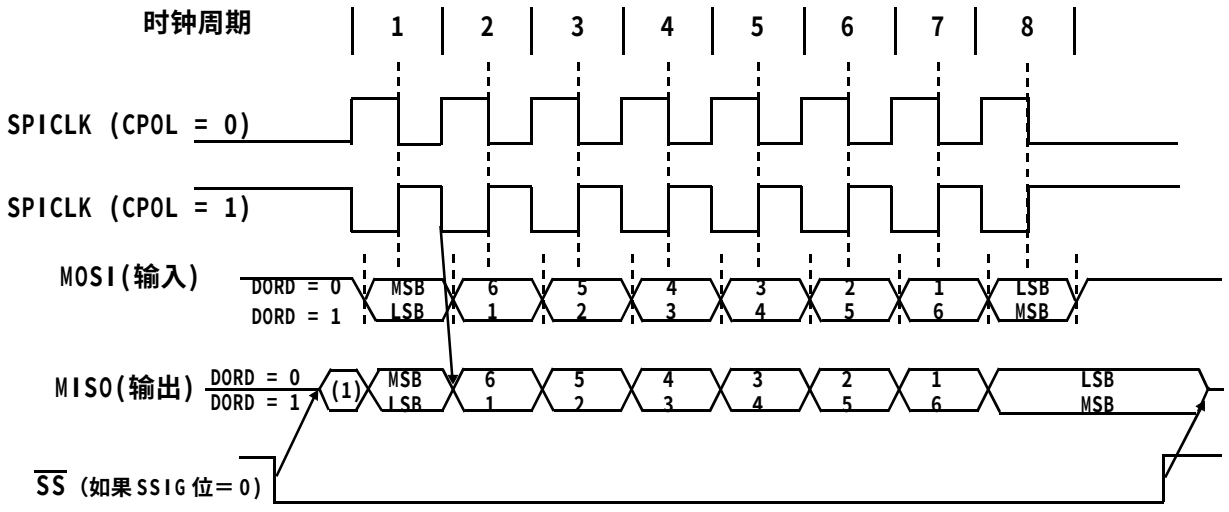
数据模式

时钟相位位(CPHA)允许用户设置采样和改变数据的时钟边沿。时钟极性位 CPOL 允许用户设置时钟极性。 SPI 图 4～图 7 所示为时钟相位位 CPHA 的不同设定。



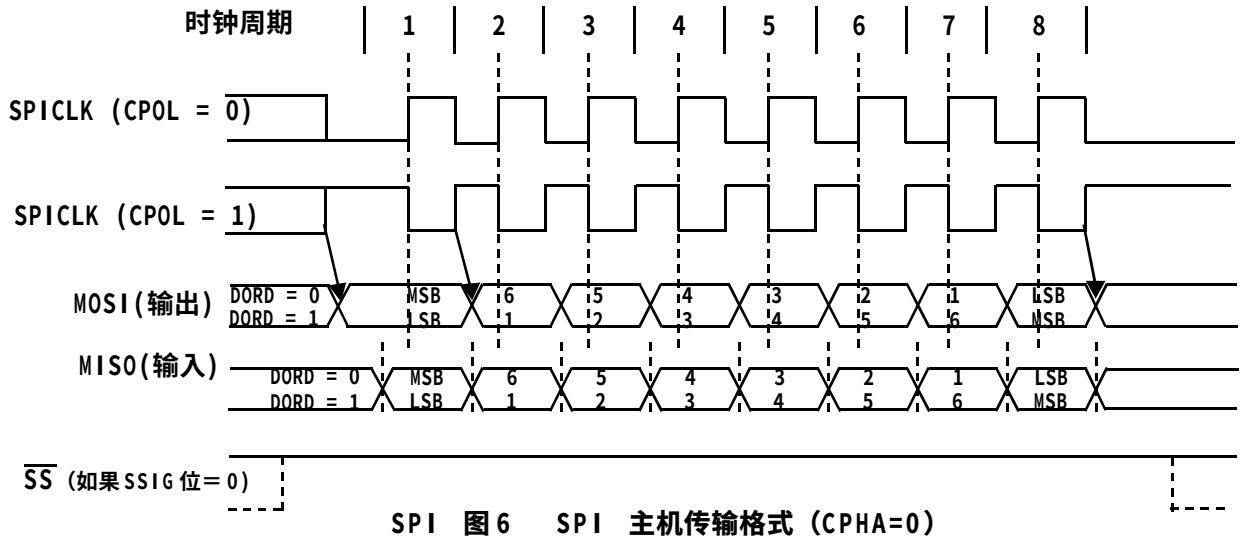
(1) ___ 未定义

SPI 图 4 SPI 从机传输格式（CPHA=0）

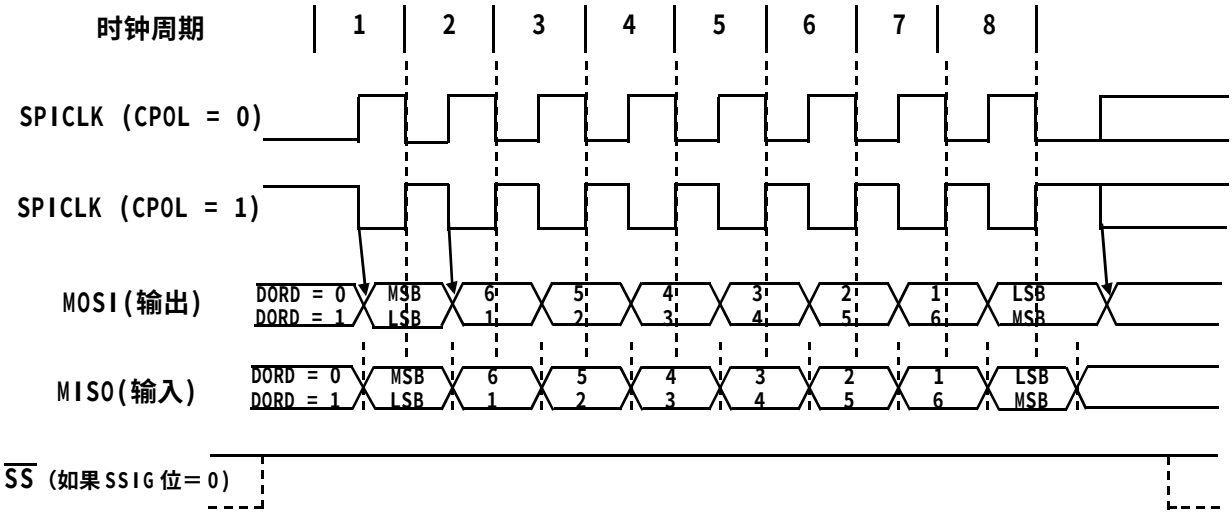


(1) ___ 未定义

SPI 图5 SPI 从机传输格式（CPHA=1）



SPI 图6 SPI 主机传输格式（CPHA=0）



SPI 图7 SPI 主机传输格式（CPHA=1）

SPI 时钟预分频器选择

SPI 时钟预分频器选择是通过 SPCTL 寄存器中的 SPR1-SPR0 位实现的

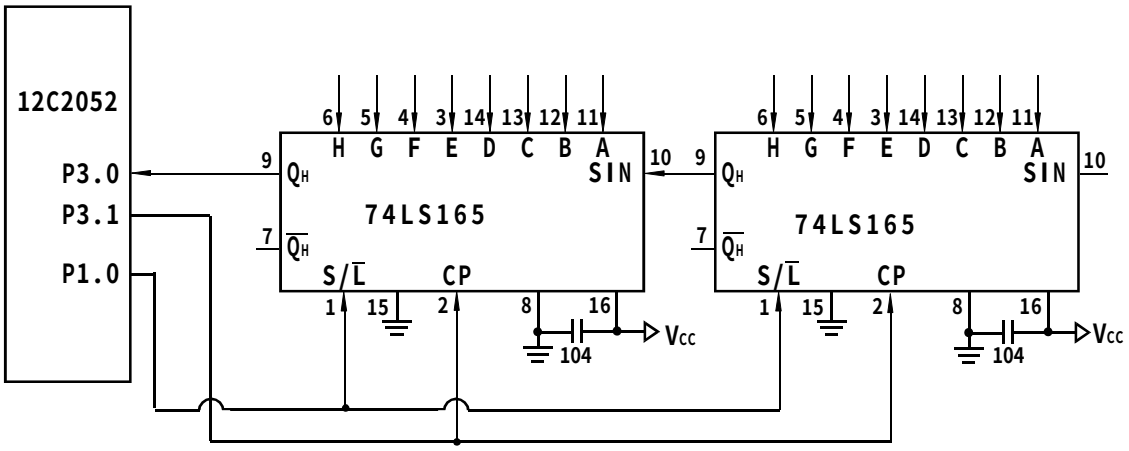
附录 F: 用串行口扩展 I/O 接口

STC12C2052 串行口的方式 0 可用于 I/O 扩展。如果在应用系统中，串行口未被占用，那么将它用来扩展并行 I/O 口是一种经济、实用的方法。

在操作方式 0 时，串行口作同步移位寄存器，其波特率是固定的，为 fosc/12（fosc 为振荡器频率)。数据由 RXD 端（P3.0）出入，同步移位时钟由 TXD 端（P3.1）输出。发送、接收的是 8 位数据，低位在先。

一、用 74LS165 扩展并行输入口

下图是利用两片 74LS165 扩展二个 8 位并行输入口的接口电路图。



74LS165 是 8 位并行置入移位寄存器。当移位 / 置入端(S/$\overline{L}$)由高到低跳变时，并行输入端的数据置入寄存器；当 S/$\overline{L}$=1，且时钟禁止端（第 15 脚）为低电平时，允许时钟输入，这时在时钟脉冲的作用下，数据将由 Q_A 到 Q_H 方向移位。

上图中，TXD(P3.1)作为移位脉冲输出端与所有 74LS165 的移位脉冲输入端 CP 相连；RXD（P3.0）作为串行输入端与 74LS165 的串行输出端 Q_H 相连；P1.0 用来控制 74LS165 的移位与置入而同 S/$\overline{L}$ 相连；74LS165 的时钟禁止端（15 脚）接地，表示允许时钟输入。当扩展多个 8 位输入口时，两芯片的首尾（Q_H 与 S_{IN}）相连。

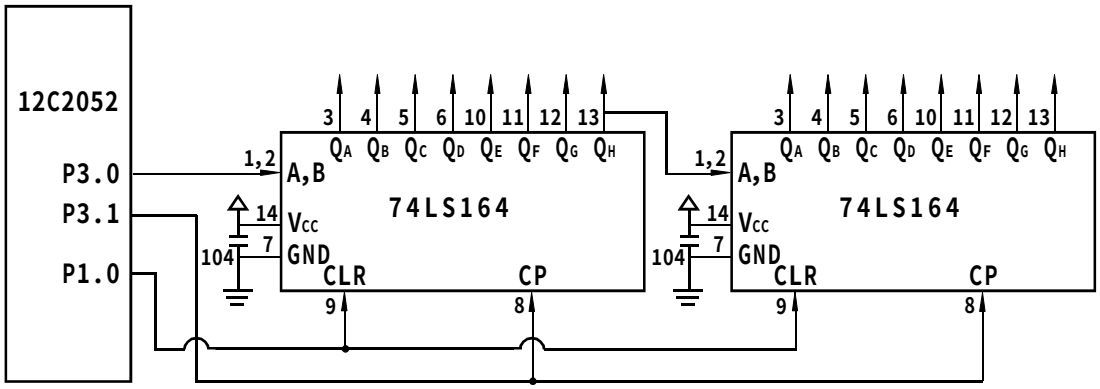
下面的程序是从 16 位扩展口读入 5 组数据（每组二个字节），并把它们转存到内部 RAM 20H 开始的单元中。

```
          MOV  R7, #05H          ; 设置读入组数
          MOV  R0, #20H          ; 设置内部 RAM 数据区首址
START:    CLR  P1.0              ; 并行置入数据，S/L=0
          SETB P1.0              ; 允许串行移位 S/L=1
          MOV  R1, #02H          ; 设置每组字节数，即外扩 74LS165 的个数
RXDATA:   MOV  SCON, #00010000B  ; 设串行方式 0 ，允许接收，启动接收过程
WAIT:     JNB  RI, WAIT          ; 未接收完一帧，循环等待
          CLR  RI                ; 清 RI 标志，准备下次接收
          MOV  A, SBUF           ; 读入数据
          MOV  @R0, A            ; 送至 RAM 缓冲区
          INC  R0                ; 指向下一个地址
          DJNZ R1, RXDATA        ; 为读完一组数据，继续
          DJNZ R7, START         ; 5 组数据未读完重新并行置入
          ……                     ; 对数据进行处理
```

上面的程序对串行接收过程采用的是查询等待的控制方式，如有必要，也可改用中断方式。从理论上讲，按上图方法扩展的输入口几乎是无限的，但扩展的越多，口的操作速度也就越慢。

二、用 74LS164 扩展并行输出口

74LS164 是 8 位串入并出移位寄存器。下图是利用 74LS164 扩展二个 8 位输出口的接口电路。



当单片机串行口工作在方式 0 的发送状态时，串行数据由 P3.0（RXD）送出，移位时钟由 P3.1（TXD）送出。在移位时钟的作用下，串行口发送缓冲器的数据一位一位地移入 74LS164 中。需要指出的是，由于 74LS164 无并行输出控制端，因而在串行输入过程中，其输出端的状态会不断变化，故在某些应用场合，在 74LS164 的输出端应加接输出三态门控制，以便保证串行输入结束后再输出数据。

下面是将 RAM 缓冲区 30H、31H 的内容串行口由 74LS164 并行输出的子程序。

```
START：  MOV     R7，#02H          ；设置要发送的字节个数
         MOV     R0，#30H          ；设置地址指针
         MOV     SCON，#00H        ；设置串行口方式 0
SEND：   MOV     A，@R0
         MOV     SBUF，A           ；启动串行口发送过程
WAIT：   JNB     TI，WAIT          ；一帧数据未发送完，循环等待
         CLR     TI
         INC     R0                ；取下一个数
         DJNZ    R7，SEND
         RET
```

附录 G: STC12C5410AD 系列 1T 单片机简介

STC12C5410AD 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 / 3.4V - 5.5V

3.工作频率范围：0 - 35 MHz，相当于普通 8051 0～420MHz

4.用户应用程序空间 1K / 2K / 4K / 6K / 8K / 10K / 12K 字节

5.片上集成 512 字节 RAM

6.EEPROM 功能

7.共 2 个 16 位定时器 / 计数器

8.PWM(4 路）/PCA（可编程计数器阵列）

9.ADC，8 路 10 位精度

10. 通用异步串行口(UART)

11. SPI 同步通信口， 主模式 / 从模式

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

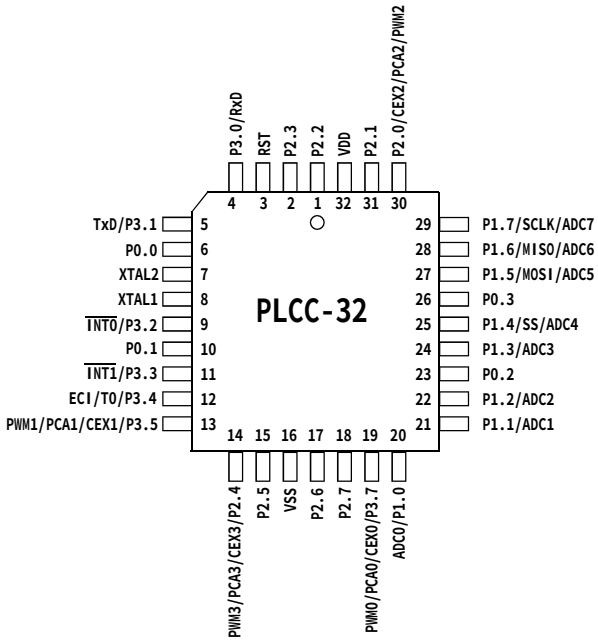
14.ISP/IAP

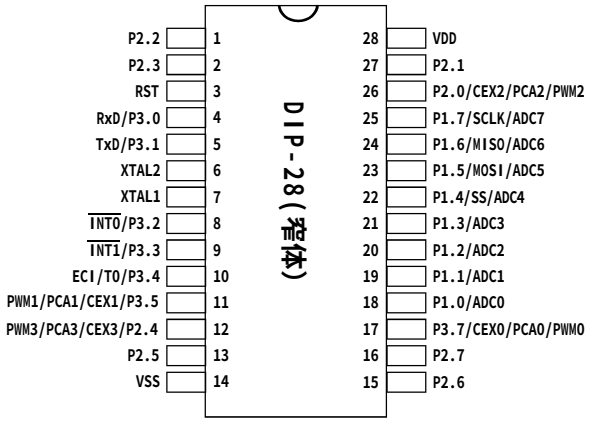
15.工作温度范围 ： 0 - 75℃ / -40 - +85℃

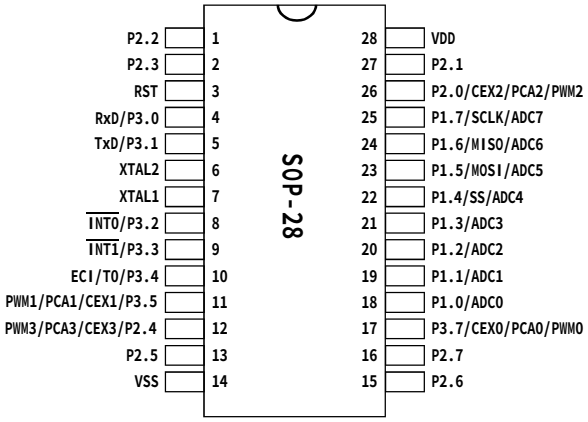
16.封装： PDIP-28(窄体)，SOP-28，TSOP-28，PLCC-32

17.供货： 2005-12-1 开始提供样品（PDIP-28/SOP-28)，2005 年 12 月底批量供货

STC12C5410AD 系列 1T 单片机管脚图







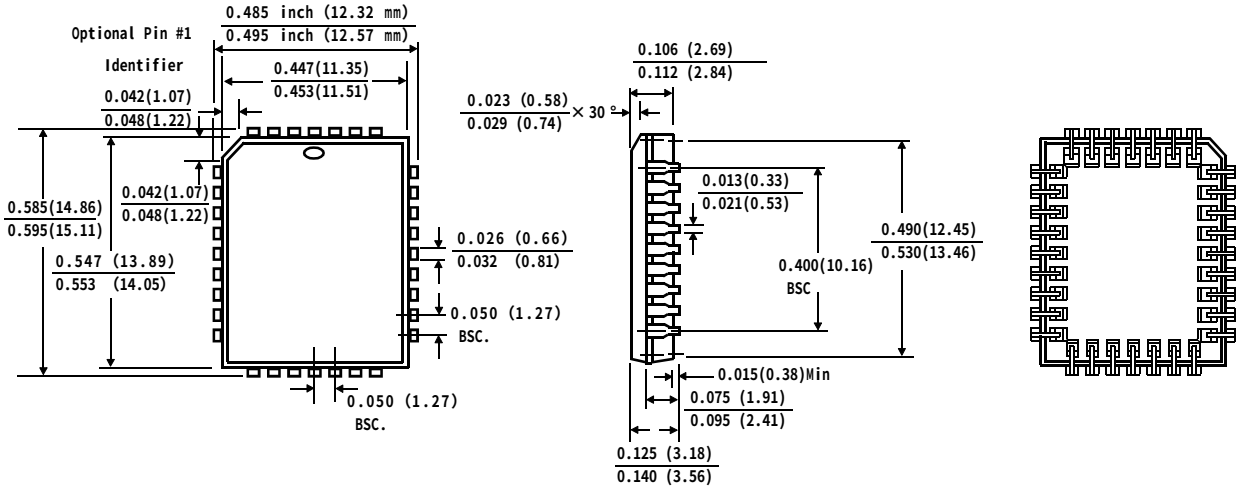
STC12C5410AD 系列单片机选型一览（全部 ISP 功能）

-----2005-12-1 供货

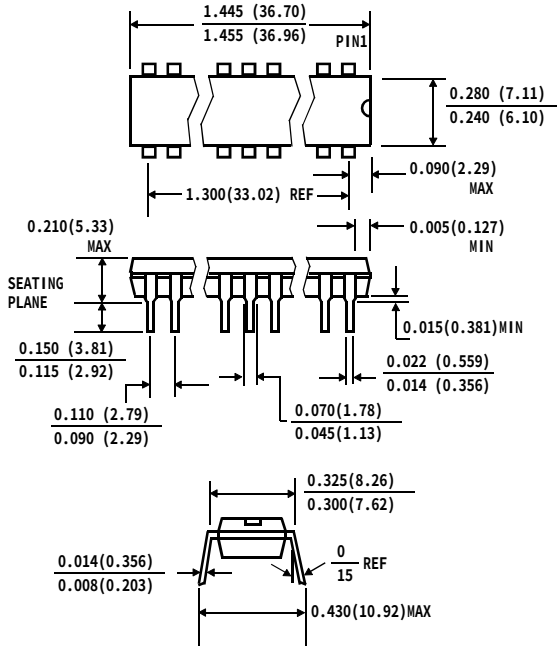
	工作电压(V)	Flash 程序存储器字节	SRAM 字节	定时器	UART	PCA PWM	A/D 8路	I/O	看门狗	内置复位	EEPROM 字节	SPI	封装 28-Pin
STC12C5401	3.4 - 5.5	1K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5401AD	3.4 - 5.5	1K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5402	3.4 - 5.5	2K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5402AD	3.4 - 5.5	2K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5404	3.4 - 5.5	4K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5404AD	3.4 - 5.5	4K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5406	3.4 - 5.5	6K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5406AD	3.4 - 5.5	6K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5408	3.4 - 5.5	8K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5408AD	3.4 - 5.5	8K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5410	3.4 - 5.5	10K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5410AD	3.4 - 5.5	10K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12C5412	3.4 - 5.5	12K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12C5412AD	3.4 - 5.5	12K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5401	2.0 - 3.8	1K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5401AD	2.0 - 3.8	1K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5402	2.0 - 3.8	2K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5402AD	2.0 - 3.8	2K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5404	2.0 - 3.8	4K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5404AD	2.0 - 3.8	4K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5406	2.0 - 3.8	6K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5406AD	2.0 - 3.8	6K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5408	2.0 - 3.8	8K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5408AD	2.0 - 3.8	8K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5410	2.0 - 3.8	10K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5410AD	2.0 - 3.8	10K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP
STC12LE5412	2.0 - 3.8	12K	512	2	有	4路		23	有	有	有	有	DIP/SOP
STC12LE5412AD	2.0 - 3.8	12K	512	2	有	4路	有	23	有	有	有	有	DIP/SOP

STC12C5410AD 系列 1T 单片机封装尺寸图

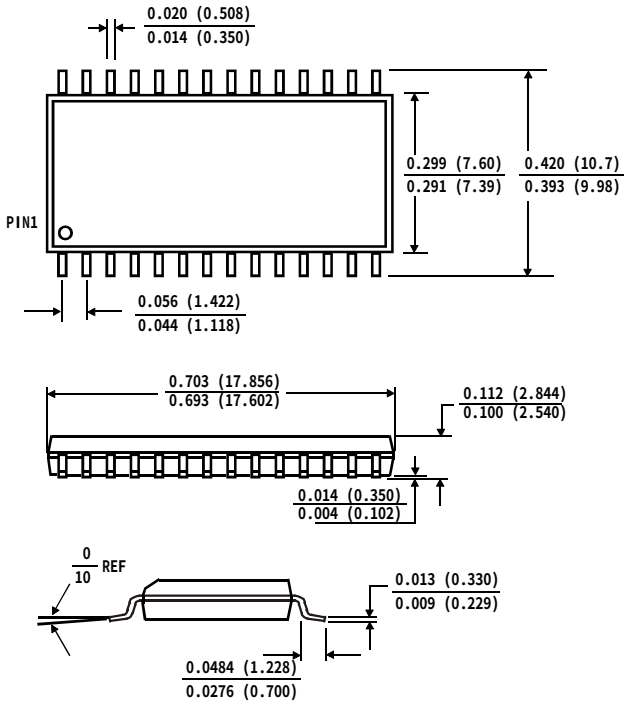
32-PIN PLASTIC LEAD CHIP CARRIER (PLCC)



28-PIN PLASTIC DUAL-IN-LINE PACKAGE (PDIP)



28-PIN SMALL OUTLINE PACKAGE (SOP)



附录H：STC12C2052AD 系列单片机调试助手程序 - 程序一

```
;*********************  STC_Debug_For_STC2052.ASM  ************************
;本程序不做技术支持，请自行消化吸收
;1. 简要说明
;     汇编程序 STC_Debug.asm 内有几个子程序可供调用：
;     1)STC_Debug_send_SFR();        发送特殊功能寄存器数据
;     2)STC_Debug_send_idata();    发送直接寻址 RAM，间接寻址 RAM 数据
;     3)STC_Debug_send_IAP_EEPROM(INT8U sector_address_H);
;                                    发送 IAP EEPROM 数据，每次只发送一个扇区(512 字节)。
;     这些数值通过 RS232 串口传送到上位机 STC-ISP.exe(STC 下载程序，版本 3.0 Beta
;以上) 显示出来，实现简易调试功能。
;    STC_Debug.asm 程序尽量先将 T0、T1、T2、PCA 等计数器关闭，但是关闭这些计数器
;之前必须保存其它数据，因此这些计数器可能会多计了一些。为使这些数据更加精确，
;可以在调用 STC_Debug.asm 之前先关闭计数器。
;
;2. 使用方法
;2.1 在项目中增加 STC_Debug.asm 程序。
;2.2 在主函数(程序)中初始化串行口。 STC12C2052 系列 MCU 使用 T1 作波特率发生器。
;2.3 修改 STC_Debug.asm 程序中相关参数 Baud_Source ，使用 T1 作波特率发生器。
;2.4 在想要观察 MCU 运行状态的地方，调用所需的子程序，调用前可以先关闭计数器。
;2.5 运行上位机 STC-ISP.exe(STC 下载程序，版本 3.0 Beta 以上)，设置调试窗中的串行
;     口、波特率、奇偶检验位、数据位和停止位，应与 MCU 程序中的相同。
;     调试通讯端口与下载应用程序的通讯端口可以完全不相关，因此他们的通讯端口
;     参数可以不同，调试通讯端口参数仅受 MCU 中的应用程序限制。
;2.6 下载应用程序。为观测 STC_Debug_send_IAP_EEPROM() 效果，下载时选 " 下载时一并
;     擦除 flash 区 "。可能需下载 2 次，因为有时第 1 次并未将 flash 区擦除。
;2.7 选中上位机调试窗口，打开调试串口，观察各存储器的变化情况。
;2.8 根据观察结果，修改 MCU 应用程序，重新编译。
;     重复 2.6--2.8。
;
;3. WAIT_ACK, WAIT_BACK 标志位
;3.1 WAIT_ACK 标志位
;    WAIT_ACK = 1 时，STC_Debug.asm 发送数据到 PC 机后等待其回发应答信号。PC 收到
;    数据后检查校验和是否正确。若校验和不正确或通讯有误，PC 请求重发(最多 3 次)，
;    否则 PC 机回发通讯成功应答信号。
;3.2 WAIT_BACK 标志位
;3.2.1 WAIT_BACK=1 时，STC_Debug.asm 发送数据后一定要收到 PC 机回发的返回命令才返回。
;3.2.2 如何令 PC 机回发返回命令：点击 PC 机调试窗口中 " 从暂停处继续运行 " 按钮。若按
;       钮处于无效状态，只要点击串口指示灯打开调试串口即可使它变成有效状态。
;3.3 WAIT_ACK, WAIT_BACK 中只要有一个为 1，就需要打开调试串口，否则 STC_Debug.asm
;    一直等待 PC 机回发应答信号。
;    竖立 WAIT_ACK 标志位可大大提高通讯可靠性，但发生校验和不正或通讯有误的情况，
;    等待双方发送完毕和再次发送完毕平均约需 1.5 秒，这时必须耐心等待一下再点击 PC
;    调试窗口中 " 从暂停处继续运行 " 按钮 "。
;
```

```
;     本示例程序适用于 STC12C2052AD 系列 MCU，晶体频率=18.432MHz，波特率=115200 。
;*******************************************************************
;定义外部函数
EXTRN CODE (STC_Debug_send_SFR)
EXTRN CODE (STC_Debug_send_idata)
EXTRN CODE (_STC_Debug_send_IAP_EEPROM)
;-------------------------------------------------------------------
;定义全局变量
PUBLIC  DEBUG_CHECK_SUM         ;STC_Debug.asm 用作计算校验和

;
PUBLIC  WAIT_ACK                ;=1 时需等待 PC 发送的应答信号才返回
PUBLIC  WAIT_BACK               ;=1 时需等待 PC 发送的返回命令才返回
;-------------------------------------------------------------------
;定义特殊功能寄存器
AUXR                EQU 0x8E;

ISP_DATA            EQU 0xE2;   ;定义与 IAP 功能有关的特殊功能寄存器
ISP_ADDRH           EQU 0xE3;
ISP_ADDRL           EQU 0xE4;
ISP_CMD             EQU 0xE5;
ISP_TRIG            EQU 0xE6;
ISP_CONTR           EQU 0xE7;
;-------------------------------------------------------------------
;定义变量
my_flag             EQU 20H
flag_time0_over     EQU my_flag.0
WAIT_ACK            EQU my_flag.1
WAIT_BACK           EQU my_flag.2

T0_overflow_count   EQU 30H     ;利用 T0 定时器周期性改变 P1 输出
Half_second         EQU 31H
DEBUG_CHECK_SUM     EQU 32H

IAP_point_H         EQU 35H     ;为观察 STC_Debug_send_IAP_EEPROM()
IAP_point_L         EQU 36H     ;运行效果定义的变量
IAP_write_byte      EQU 37H

idata_0x38          EQU 38H     ;为观察 STC_Debug_send_idata()
idata_0x80          EQU 80H     ;运行效果定义的变量
idata_0xFE          EQU 0FEH
;-------------------------------------------------------------------
    ORG   0000H
    AJMP  MAIN
;-------------------------------------------------------------------
```

```
ORG   000BH
timer0:                         ;定时器0，定时改变 P1 口状态
    PUSH  PSW
    PUSH  ACC

    MOV   TH0, #3CH
    MOV   TL0, #0B4H

    INC   T0_overflow_count
    MOV   A, AUXR
    JNB   ACC.6, timer0_12T
    MOV   A, #239               ;1T
    SJMP  timer0_a
timer0_12T:
    MOV   A, #23                ;12T
timer0_a:
    CLR   C
    SUBB  A, T0_overflow_count
    JNC   timer0_ret
    MOV   A, P1
    CPL   A
    MOV   P1, A
    MOV   T0_overflow_count, #0
    SETB  flag_time0_over
timer0_ret:
    POP   ACC
    POP   PSW
    RETI
;-------------------------------------------------------------------
initiate_RS232:                 ;串口初始化，T1 作波特率发生器
    MOV   SCON, #50H            ;0101,0000 8位可变波特率，无奇偶校验
    MOV   TMOD, #21H
    MOV   TH1, #0FBH            ;速度 = 1T, 18.432MHz, baud = 115200
    MOV   TL1, TH1
;   MOV   PCON, #80H            ;波特率加倍
    SETB  TR1
    CLR   RI
    SETB  EA
    RET
;-------------------------------------------------------------------
;IAP 字节编程，入口: A=编程字节，DPTR=编程地址
byte_program:
    MOV   ISP_CONTR, #80H       ;打开 IAP 功能
    MOV   ISP_CMD, #2           ;Select Byte Program Mode
    MOV   ISP_ADDRH, DPH        ;Fill byte address in ISP_ADDRH & ISP_ADDRL
    MOV   ISP_ADDRL, DPL
    MOV   ISP_DATA, A           ;Fill the data to be programmed in ISP_DATA
```

```
CLR   EA
    MOV   ISP_TRIG, #46H          ;Trigger ISP processing
    MOV   ISP_TRIG, #0b9H
    SETB  EA
    MOV   ISP_CONTR, #00          ;关闭 IAP 功能
    RET
;-------------------------------------------------------------------
MAIN:
    MOV   SP, #40H
    MOV   AUXR, #0C0H              ;1100，0000  T0, T1 以 1T 的速度运行，
                                  ;是普通 8051DE 12 倍
    ACALL initiate_RS232
    ORL   TMOD, #01H
    MOV   TH0, #3CH
    SETB  TR0
    SETB  ET0
    MOV   Half_second, #0
    MOV   P1, #0FH
    SETB  EA

    MOV   IAP_point_H, #11H       ;初始指向 IAP EEPROM 区域的指针
    MOV   IAP_point_L, #80H
    MOV   IAP_write_byte, #0
MAIN_loop:
    JNB   flag_time0_over, MAIN_loop
    CLR   flag_time0_over
    INC   Half_second
    MOV   A, Half_second          ;若干个半秒观察一次 MCU 状态
    CLR   C
    SUBB  A, #5
    JC    MAIN_loop

    MOV   Half_second, #0

    INC   B                       ;改变累加器 A 和 B，可以在 sfr
    MOV   A, B                    ;调试窗口中观察到它们不断变化
;   SETB  WAIT_ACK                ;需等待 PC 发送的应答信号才返回
    CLR   WAIT_ACK                ;需等待 PC 发送的应答信号才返回
    CLR   WAIT_BACK               ;无需等待 PC 发送的返回命令
    ACALL STC_Debug_send_SFR

    INC   idata_0x38              ;改变它们的数值，可以在 idata 调试窗口
    MOV   R0, #idata_0x80         ;中观察到它们不断变化
    INC   @R0
    MOV   R0, #idata_0xFE
    INC   @R0
    ACALL STC_Debug_send_idata    ;发送 idata 数据到上位机。
```

```
    MOV   DPH, IAP_point_H       ;改变 IAP EEPROM 数值，地址由 IAP_point 确定，
    MOV   DPL, IAP_point_L      ;数值由 IAP_write_byte 确定。
    MOV   A, IAP_write_byte
    ACALL byte_program
    INC   IAP_write_byte
    MOV   R7, IAP_point_H       ;数值由 IAP_write_byte 确定。R7= 子程序所需参数
;    SETB  WAIT_ACK              ;需等待 PC 发送的应答信号才返回
;    SETB  WAIT_BACK             ;需等待 PC 发送的返回命令
    CLR   WAIT_ACK              ;需等待 PC 发送的应答信号才返回
    CLR   WAIT_BACK             ;需等待 PC 发送的返回命令
    ACALL _STC_Debug_send_IAP_EEPROM ;发送 IAP EEPROM 数据到上位机。

    MOV   A, IAP_point_L        ;IAP_point + 1
    ADD   A, #1
    MOV   IAP_point_L, A
    CLR   A
    ADDC  A, IAP_point_H
    MOV   IAP_point_H, A

    SJMP  MAIN_loop
;------------------------------------------------------------------------
    END
;------------------------------------------------------------------------
```

STC12C2052AD 系列单片机调试助手程序 — 程序二

```
;STC_Debug.ASM
PUBLIC   STC_Debug_send_SFR
_STC_Debug_send_SFR SEGMENT    CODE    ;程序段
RSEG    _STC_Debug_send_SFR

EXTRN ..... BIT     (WAIT_ACK)              ;=1 时需等待 PC 发送的应答信号才返回
EXTRN ..... BIT     (WAIT_BACK)             ;=1 时需等待 PC 发送的返回命令才返回

EXTRN ..... DATA    (DEBUG_CHECK_SUM)       ;用作计算校验和

PUBLIC STC_Debug_send_idata
PUBLIC STC_Debug_send_xdata
PUBLIC _STC_Debug_send_IAP_EEPROM
PUBLIC _Send_Byte
;-------------------------------------------------------------------
;定义特殊功能寄存器
T2CON                   EQU 0C8H
TR2                     EQU T2CON.2
CCON                    EQU 0D8H
CR                      EQU CCON.6

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
WAIT_TIME               EQU 2       ;设置等待时间,
                                    ;30M 以下=0，24M 以下=1，20M 以下=2，12M 以下=3,
                                    ;6M 以下=4，3M 以下=5，2M 以下=6，1M 以下=0
;如果通讯波特率比较低，传输数据可能要较长时间，可以修改以下常数,
;只传输需要关注的数据

addition                EQU 11H     ;附加长度

SFR_Length              EQU 0080H   ;发送 SFR 数据长度 = 128
idata_Length            EQU 0100H   ;发送 idata 数据长度 = 256
xdata_Length            EQU 0200H   ;发送 xdata 数据长度 = 512
;xdata_Length           EQU 0400H   ;发送 xdata 数据长度 = 1024
IAP_EEPROM_Length       EQU 0200H   ;发送 IAP_EEPROM 数据长度 = 512
;-------------------------------------------------------------------
Baud_Source             EQU 1       ;使用 T1 作波特率发生器
;Baud_Source            EQU 2       ;使用 T2 作波特率发生器
;-------------------------------------------------------------------
```

```
;通讯协议
;1. 发送数据
;------------------
;-  A5H
;-  5AH
;0  AAH        从此处开始计算长度
;1  number_H   AAH 及跟随的字节数高位，n + 17
;2  number_L   跟随的字节数低位
;3  区域代码   0:SFR; 1:idata; 2:xdata; 3:IAP flash; 4:EEPROM
;4  地址 2     高地址
;5  地址 1
;6  地址 0     低地址

;7  PSW
;8  ACC
;9  TCON
;10 T2CON
;11 CCON
;12 IE
;13 SP
;14 R1         工作寄存器组 0 的 R0 地址 = 00H
;15 R0         工作寄存器组 0 的 R1 地址 = 01H

;16 数据 0
;17 数据 1
;    ..
;    ..
;   数据 n-1
;   校验和，从第 0 个字节 AA 到数据 n-1
;总长 = n + 19 (包括 A5H，5AH)

;2. 接收返回信息
;------------------
;   AAH
;   返回信息，1 字节
;   校验和，AA + 返回信息
;
; 返回信息:
;   50H: PC 机已收到 MCU 发送的数据
;   52H: PC 机收到的数据有误，请求重发
;   54H: 返回应用程序继续运行

;-------------------------------------------------------------------
IAP_Enable:                     ;打开 IAP 功能，
    MOV   ISP_CONTR, #WAIT_TIME ;设置等待时间
    ORL   ISP_CONTR, #10000000B ;允许 ISP/IAP 操作
    RET
;-------------------------------------------------------------------
```

```
IAP_Disable:                    ;关闭 IAP 功能
    MOV   ISP_CONTR, #0
    MOV   ISP_CMD, #0
    MOV   ISP_TRIG, #0
    MOV   ISP_ADDRH, #0
    MOV   ISP_ADDRL, #0
    RET
;-------------------------------------------------------------
;入口:DPTR = 字节地址
;出口:A = 读出字节
;读一字节 IAP command
byte_verify:
    MOV   ISP_CMD, #1           ;Select Read AP Mode
    MOV   ISP_ADDRH, DPH        ;Fill byte address in ISP_ADDRH & ISP_ADDRL
     MOV   ISP_ADDRL, DPL
    MOV   ISP_DATA, #00H        ;清数据寄存器
trigger_ISP:
    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
;-------------------------------------------------------------
Send_head_1:

    if Baud_Source = 1
        SETB  TR1               ;使用 T1 作波特率发生器
    else
        SETB  TR2               ;使用 T2 作波特率发生器
    endif

    MOV   A, #0A5H
    ACALL _Send_Byte
    MOV   A, #05AH
    ACALL _Send_Byte
    MOV   A, #0AAH              ;发送第 0 个字节
    ACALL _Send_Byte
    MOV   DEBUG_CHECK_SUM, A
    RET
;-------------------------------------------------------------
Send_Stack:
    MOV   A, SP                 ;发送第 7--15 个字节
    CLR   C
    SUBB  A, #10
```

```
    MOV   R0, A
    MOV   R1, #9
LOOP:
    MOV   A, @R0
    ACALL _Send_Byte
    INC   R0
    DJNZ  R1, LOOP
    RET
;-------------------------------------------------------------
Exit_debug:
    CLR   A                     ;延时，给上位机留下处理数据的时间
    MOV   R0, A
    MOV   R1, A
    MOV   A, #100               ;根据晶体频率调整此数，不要延时过长
Exit_debug_loop:
;    DJNZ  R0, Exit_debug_loop  ;延时过长可注释本行指令
    DJNZ  R1, Exit_debug_loop
    DJNZ  ACC, Exit_debug_loop

    CLR   RI
    POP   0
    POP   1
    POP   ACC
    POP   IE
    POP   CCON
    POP   T2CON
    POP   TCON
    POP   ACC
    POP   PSW
    RET
;-------------------------------------------------------------
set_stack_point:
    MOV   R0, SP
    INC   R0
    INC   R0
    INC   R0
    RET
;-------------------------------------------------------------
;入口: R7 = 扇区地址高位
_STC_Debug_send_IAP_EEPROM:     ;发送 IAP EEPROM 数据，每次发送一个扇区(512 字节)。
    PUSH  PSW
    PUSH  ACC
    PUSH  TCON
    PUSH  T2CON
    PUSH  CCON
    PUSH  IE
```

```
CLR   TR0
    CLR   TR1
    CLR   TR2
    CLR   CR
    CLR   EA
    MOV   A, R7                 ;保存原工作寄存器组的 R7-- 扇区首地址高 8 位
    MOV   PSW, #0               ;工作寄存器组 0
    PUSH  SP
    PUSH  1                     ;保存 R1
    PUSH  0                     ;保存 R0
STC_Debug_IAP_EEPROM_Resend:
    ACALL set_stack_point
    ANL   A, #11111110B
    MOV   @R0, A                ;保存原工作寄存器组的 R7

    ACALL Send_head_1

    MOV   A, #HIGH(IAP_EEPROM_Length + addition) ;1 跟随的字节数高 8 位
    ACALL _Send_Byte
    MOV   A, #LOW(IAP_EEPROM_Length + addition) ;2 跟随的字节数低 8 位
    ACALL _Send_Byte
    MOV   A, #03H               ;3  区域代码:
    ACALL _Send_Byte            ;   0:SFR; 1:idata; 2:xdata; 3:IAP flash; 4:EEPROM
    MOV   A, #00H               ;4  地址 2      高地址
    ACALL _Send_Byte
    MOV   A, @R0                ;5  地址 1      扇区首地址高 8 位
    ACALL _Send_Byte
    MOV   A, #0                 ;6  地址 0      低地址 = 0
    ACALL _Send_Byte

    ACALL Send_Stack

    ACALL set_stack_point
     MOV   A, @R0               ;取回扇区首地址高 8 位

    PUSH  DPH
    PUSH  DPL
    MOV   DPH, A                ;扇区地址高位 -->DPH
    MOV   DPL, #0

    MOV   R0, #0
    MOV   R1, #2
    ACALL IAP_Enable            ;打开 IAP 功能,
send_IAP_EEPROM_Loop:
    ACALL byte_verify
ACALL _Send_Byte
    INC   DPTR
```

```
DJNZ  R0, send_IAP_EEPROM_Loop
    DJNZ  R1, send_IAP_EEPROM_Loop

    ACALL IAP_Disable           ;关闭 IAP 功能
    POP   DPL
    POP   DPH
    ACALL send_check_sum
    MOV   R0, #03
    AJMP  wait_PC_command
;----------------------------------------------------------------
STC_Debug_send_xdata:           ;发送内部扩展 RAM 数据
    PUSH  PSW
    PUSH  ACC
    PUSH  TCON
    PUSH  T2CON
    PUSH  CCON
    PUSH  IE

    CLR   TR0
    CLR   TR1
    CLR   TR2
    CLR   CR
    CLR   EA
    MOV   PSW, #0               ;工作寄存器组 0
    PUSH  SP
    PUSH  1                     ;保存 R1
    PUSH  0                     ;保存 R0
STC_Debug_xdata_Resend:
    ACALL Send_head_1

    MOV   A, #HIGH(xdata_Length + addition) ;1 跟随的字节数高 8 位
    ACALL _Send_Byte
    MOV   A, #LOW(xdata_Length + addition) ;2 跟随的字节数低 8 位
    ACALL _Send_Byte
    MOV   A, #02H       ;3  区域代码  0:SFR; 1:idata; 2:xdata; 3:IAP flash; 4:EEPROM
    ACALL _Send_Byte
    MOV   A, #00H       ;4  地址 2     高地址
    ACALL _Send_Byte
    MOV   A, #00H       ;5  地址 1
    ACALL _Send_Byte
    MOV   A, #00H       ;6  地址 0     低地址
    ACALL _Send_Byte

    ACALL Send_Stack

    PUSH  DPH
    PUSH  DPL
    MOV   DPTR, #0
```

```
    MOV   R1, #HIGH(xdata_Length)
    MOV   R0, #LOW(xdata_Length)
send_xdata_Loop:
    MOVX  A, @DPTR
    ACALL _Send_Byte
    INC   DPTR
    DJNZ  R0, send_xdata_Loop
    DJNZ  R1, send_xdata_Loop

    POP   DPL
    POP   DPH
ACALL send_check_sum
    MOV   R0, #2
    AJMP  wait_PC_command
;----------------------------------------------------------------
STC_Debug_send_idata:            ;发送直接寻址 RAM，间接寻址 RAM 数据
    PUSH  PSW
    PUSH  ACC
    PUSH  TCON
    PUSH  T2CON
    PUSH  CCON
    PUSH  IE

    CLR   TR0
    CLR   TR1
    CLR   TR2
    CLR   CR
    CLR   EA
    MOV   PSW, #0                ;工作寄存器组 0
    PUSH  SP
    PUSH  1                      ;保存 R1
    PUSH  0                      ;保存 R0
STC_Debug_idata_Resend:
    ACALL Send_head_1

    MOV   A, #HIGH(idata_Length + addition) ;1 跟随的字节数高 8 位
    ACALL _Send_Byte
    MOV   A, #LOW(idata_Length + addition) ;2 跟随的字节数低 8 位
    ACALL _Send_Byte
    MOV   A, #01H        ;3 区域代码  0:SFR; 1:idata; 2:xdata; 3:IAP flash; 4:EEPROM
    ACALL _Send_Byte
    MOV   A, #00H        ;4  地址 2     高地址
    ACALL _Send_Byte
    MOV   A, #00H        ;5  地址 1
    ACALL _Send_Byte
    MOV   A, #00H        ;6  地址 0     低地址
    ACALL _Send_Byte
```

```
    ACALL Send_Stack
    POP   ACC
    ACALL _Send_Byte
    POP   ACC
    ACALL _Send_Byte
    INC   SP
    INC   SP
    MOV   R0, #2
    MOV   R1, #254
send_idata_Loop:
    MOV   A, @R0
    ACALL _Send_Byte
    INC   R0
    DJNZ  R1, send_idata_Loop
    ACALL send_check_sum
    MOV   R0, #1
    AJMP  wait_PC_command
;----------------------------------------------------------------
STC_Debug_send_SFR:             ;发送特殊功能寄存器数据
    PUSH  PSW
    PUSH  ACC
    PUSH  TCON
    PUSH  T2CON
    PUSH  CCON
    PUSH  IE
    CLR   TR0
    CLR   TR1
    CLR   TR2
    CLR   CR
    CLR   EA
    MOV   PSW, #0               ;工作寄存器组 0
    PUSH  SP
    PUSH  1                     ;保存 R1
    PUSH  0                     ;保存 R0
STC_Debug_SFR_Resend:
    ACALL Send_head_1
    MOV   A, #HIGH(SFR_Length + addition) ;1 跟随的字节数高 8 位
    ACALL _Send_Byte
    MOV   A, #LOW(SFR_Length + addition) ;2 跟随的字节数低 8 位
    ACALL _Send_Byte
    MOV   A, #00H       ;3 区域代码  0:SFR; 1:idata; 2:xdata; 3:IAP flash; 4:EEPROM
    ACALL _Send_Byte
    MOV   A, #00H       ;4  地址 2     高地址
    ACALL _Send_Byte
    MOV   A, #00H       ;5  地址 1
    ACALL _Send_Byte
```

```
    MOV   A, #80H        ;6  地址0     低地址
    ACALL _Send_Byte
    ACALL Send_Stack
    ACALL Send_SFR_01
    ACALL Send_SFR_02
    ACALL Send_SFR_03
    ACALL Send_SFR_04
    ACALL Send_SFR_05
    ACALL Send_SFR_06
    ACALL Send_SFR_07
    ACALL Send_SFR_08
    ACALL Send_SFR_09
    ACALL Send_SFR_10
    ACALL Send_SFR_11
    ACALL Send_SFR_12
    ACALL Send_SFR_13
    ACALL Send_SFR_14
    ACALL Send_SFR_15
    ACALL Send_SFR_16
    ACALL Send_SFR_17
    ACALL Send_SFR_18
    MOV   A, 0FEH
    ACALL _Send_Byte
    MOV   A, 0FFH
     ACALL _Send_Byte
    ACALL send_check_sum
    MOV   R0, #0
    AJMP  wait_PC_command
;-------------------------------------------------------------------
Send_SFR_01:
    PUSH  86H
    PUSH  85H
    PUSH  84H
    PUSH  83H
     PUSH  82H
    PUSH  81H
    MOV   A, 80H
    AJMP  Send7Bytes
;-------------------
Send_SFR_02:
    PUSH  8DH
    PUSH  8CH
    PUSH  8BH
    PUSH  8AH
    PUSH  89H
    PUSH  88H
```

```
MOV   A, 87H
    AJMP  Send7Bytes
;-------------------
Send_SFR_03:
    PUSH  94H
    PUSH  93H
    PUSH  92H
    PUSH  91H
    PUSH  90H
    PUSH  8FH
    MOV   A, 8EH
    AJMP  Send7Bytes
;-------------------
Send_SFR_04:
    PUSH  9BH
    PUSH  9AH
    PUSH  99H
    PUSH  98H
    PUSH  97H
    PUSH  96H
    MOV   A, 95H
    AJMP  Send7Bytes
;-------------------
Send_SFR_05:
    PUSH  0A2H
    PUSH  0A1H
    PUSH  0A0H
    PUSH  9FH
    PUSH  9EH
    PUSH  9DH
    MOV   A, 9CH
    AJMP  Send7Bytes
;-------------------
Send_SFR_06:
    PUSH  0A9H
    PUSH  0A8H
    PUSH  0A7H
    PUSH  0A6H
    PUSH  0A5H
    PUSH  0A4H
    MOV   A, 0A3H
     AJMP  Send7Bytes
;-------------------
Send_SFR_07:
    PUSH  0B0H
    PUSH  0AFH
```

```
    PUSH  0AEH
    PUSH  0ADH
    PUSH  0ACH
    PUSH  0ABH
    MOV   A, 0AAH
    AJMP  Send7Bytes
;------------------
Send_SFR_08:
    PUSH  0B7H
    PUSH  0B6H
    PUSH  0B5H
    PUSH  0B4H
    PUSH  0B3H
    PUSH  0B2H
    MOV   A, 0B1H
    AJMP  Send7Bytes
;------------------
Send_SFR_09:
    PUSH  0BEH
    PUSH  0BDH
    PUSH  0BCH
    PUSH  0BBH
    PUSH  0BAH
    PUSH  0B9H
    MOV   A, 0B8H
    AJMP  Send7Bytes
;------------------
Send_SFR_10:
    PUSH  0C5H
    PUSH  0C4H
    PUSH  0C3H
    PUSH  0C2H
    PUSH  0C1H
    PUSH  0C0H
    MOV   A, 0BFH
    AJMP  Send7Bytes
;------------------
Send_SFR_11:
    PUSH  0CCH
    PUSH  0CBH
    PUSH  0CAH
    PUSH  0C9H
    PUSH  0C8H
    PUSH  0C7H
    MOV   A, 0C6H
    AJMP  Send7Bytes
;------------------
```

```
Send_SFR_12:
    PUSH  0D3H
    PUSH  0D2H
    PUSH  0D1H
    PUSH  0D0H
    PUSH  0CFH
    PUSH  0CEH
    MOV   A, 0CDH
    AJMP  Send7Bytes
;-------------------
Send_SFR_13:
    PUSH  0DAH
    PUSH  0D9H
    PUSH  0D8H
    PUSH  0D7H
    PUSH  0D6H
    PUSH  0D5H
    MOV   A, 0D4H
    AJMP  Send7Bytes
;-------------------
Send_SFR_14:
    PUSH  0E1H
    PUSH  0E0H
    PUSH  0DFH
    PUSH  0DEH
    PUSH  0DDH
    PUSH  0DCH
    MOV   A, 0DBH
    AJMP  Send7Bytes
;-------------------
Send_SFR_15:
    PUSH  0E8H
    PUSH  0E7H
    PUSH  0E6H
    PUSH  0E5H
    PUSH  0E4H
    PUSH  0E3H
    MOV   A, 0E2H
    AJMP  Send7Bytes
;-------------------
Send_SFR_16:
    PUSH  0EFH
    PUSH  0EEH
    PUSH  0EDH
    PUSH  0ECH
    PUSH  0EBH
    PUSH  0EAH
```

```
MOV   A, 0E9H
    AJMP  Send7Bytes
;------------------
Send_SFR_17:
    PUSH  0F6H
    PUSH  0F5H
    PUSH  0F4H
    PUSH  0F3H
    PUSH  0F2H
    PUSH  0F1H
    MOV   A, 0F0H
    AJMP  Send7Bytes
;------------------
Send_SFR_18:
    PUSH  0FDH
    PUSH  0FCH
    PUSH  0FBH
    PUSH  0FAH
    PUSH  0F9H
    PUSH  0F8H
    MOV   A, 0F7H
    AJMP  Send7Bytes
;---------------------------------------------------------------
Send7Bytes:
    ACALL _Send_Byte            ;发送一个字节
     MOV   R0,#6                ;设置计数器
SendStackFrame:
    POP   ACC                   ;将要传送的内容送到ACC中
    ACALL _Send_Byte            ;调用发送子程序,发送一个字节
    DJNZ  R0,SendStackFrame     ;判断是否发送完毕
    RET
;---------------------------------------------------------------
send_check_sum:
    MOV   A, DEBUG_CHECK_SUM
     ACALL _Send_Byte           ;发送一个字节
    RET
;---------------------------------------------------------------
_Send_Byte:
    CLR   TI                    ;清零串口发送中断标志
    MOV   SBUF, A
    XCH   A, DEBUG_CHECK_SUM
    ADD   A, DEBUG_CHECK_SUM
    XCH   A, DEBUG_CHECK_SUM
    JNB   TI, $                 ;若 TI = 1, 跳回当前指令
    CLR   TI
    RET
;---------------------------------------------------------------
```

```
Receive_Byte:
    CLR   RI
Receive_Byte_loop:
    JNB   RI, Receive_Byte_loop
    MOV   A, SBUF
    RET
;-------------------------------------------------------------
wait_PC_command:
;   AAH                      1 字节
;   返回信息,                1 字节
;   校验和, AA + 返回信息,  1 字节
;
; 返回信息:
;   50H: PC 机已收到 MCU 发送的数据
;   52H: PC 机收到的数据有误，请求重发
;   54H: 返回应用程序继续运行

    JB    WAIT_ACK, wait_PC_command_A
    JB    WAIT_BACK, wait_PC_command_A
    AJMP  Exit_debug             ;无需等待 PC 发送的任何命令，返回应用程序继续运行
wait_PC_command_A:
    ACALL receive_byte
    XRL   A, #0AAH
    JNZ   wait_PC_command
    ACALL receive_byte           ;接收返回信息
    MOV   R1, A
    ACALL receive_byte           ;接收校验和, AA + 返回信息
    ADD   A, #56H                ;56H = 0AAH 的补码
    XRL   A, R1
    JNZ   wait_PC_command
;-------------------------------------------------------------
    MOV   A, R1
    CJNE  A, #50H, wait_Resend  ;PC 机已收到 MCU 发送的数据
    JB    WAIT_BACK, wait_PC_command ;需等待 PC 发送的返回命令才返回
    AJMP  Exit_debug             ;NOWAIT=1, 无需等待返回应用程序命令, 立刻返回
;-------------------------------------------------------------
wait_Resend:
    CJNE  A, #52H, wait_Return
     MOV   A, R0                 ;PC 机收到的数据有误，请求重发
;--------------------
    CJNE  A, #0, wait_Resend_idata
    AJMP  STC_Debug_SFR_Resend
;--------------------
wait_Resend_idata:
    CJNE  A, #1, wait_Resend_xdata
    AJMP  STC_Debug_idata_Resend
;--------------------
```

```
wait_Resend_xdata:
     CJNE  A, #2, wait_Resend_IAP_EEPROM
    AJMP  STC_Debug_xdata_Resend
;--------------------
wait_Resend_IAP_EEPROM:
    CJNE  A, #3, wait_PC_command
    AJMP  STC_Debug_IAP_EEPROM_Resend
;------------------------------------------------------------
wait_Return:
    CJNE  A, #54H, wait_PC_command ;命令有误，继续等待
    AJMP  Exit_debug              ;返回应用程序继续运行
;------------------------------------------------------------------
    END
;------------------------------------------------------------------

;80H P0
;81H SP
;82H DPL
;83H DPH
;84H SPI_STATUS
;85H SPI_CONTR
;86H SPI_DATA
;87H PCON
;88H TCON
;89H TMOD
;8AH TL0
;8BH TL1
;8CH TH0
;8DH TH1
;8EH AUXR
;90H P1
;91H P1M0
;92H P1M1
;98H SCON
;99H SBUF
;A8H IE
;A9H SADDR
;B0H P3
;B1H P3M0
;B2H P3M1
;B7H IPH
;B8H IP
;B9H SADEN
;C5H ADC_CONTR
;C6H ADC_DATA
;C7H CLOCK_DIV
```

```
;D0H PSW
;D8H CCON
;D9H CMOD
;DAH CCAPM0
;DBH CCAPM1
;E0H ACC
;E1H WDT_CONTR
;E2H ISP_DATA
;E3H ISP_ADDRH
;E4H ISP_ADDRL
;E5H ISP_CMD
;E6H ISP_TRIG
;E7H ISP_CONTR
;E9H CL
;EAH CCAP0L
;EBH CCAP1L
;F0H B
;F2H PCA_PWM0
;F3H PCA_PWM1
;F9H CH
;FAH CCAP0H
;FBH CCAP1H
```