单片机学习板 89C51-IV

（STC89C51 系列）

西安石油大学电子工程学院

二 00 八年九月

1. 系统概述

89C51-III 单片机学习板是一款基于 8 位单片机处理芯片 STC89C52RC 的系统。其功能强大，可以实现单片机开发的多种要求，学习、开发者可以根据需要选配多种常用模块，达到实验及教学的目的。

89C51-III 单片机学习板功能强大，具有报警，跑马灯、串行通信（max232）、段码液晶（msm0801LCD）和字符液晶显示(LCD1602)、电机控制(L298)、A/D 转换(TLC2543)、D/A 转换(TLC5615)、温度采集(DS1602)、数字信号合成(AD9851)、实时时钟电路(DS1302)、4—20mA 输出、PWM 输出（UC3842)、红外检测（KSM-603LM）控制等十七种功能，供学习者学习开发使用。89C51-III 单片机学习板采用的芯片都是常用芯片，使学习者对常用电子产品进一步学习理解。

2. 系统原理

2.1 系统组成

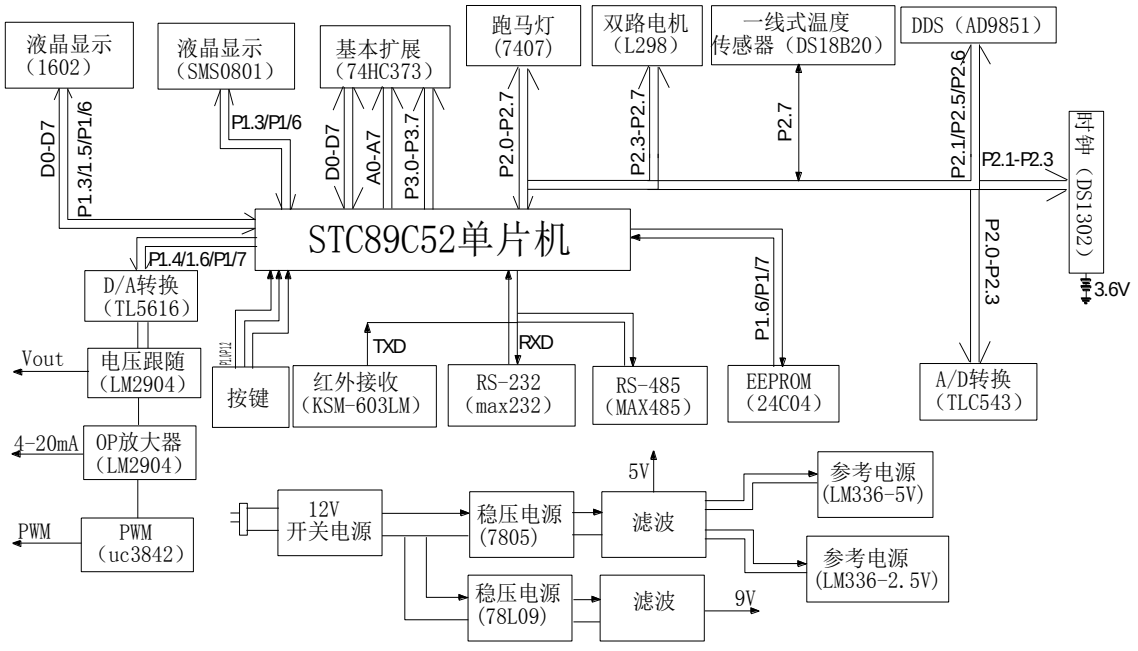


图 1-1 系统原理图

2.2 主 CPU 电路

主 CPU 电路选用 STC89C52RC 系列单片机，STC89C52RC 是采用 8051 核的 ISP（In System Programming）在系统可编程芯片，最高工作时钟频率为 80MHz，片内含 8K Bytes 的可反复擦写 1000 次的 Flash 只读程序存储器，器件兼容标准 MCS-51 指令系统及 80C51 引脚结构，芯片内集成了通用 8 位中央处理器和 ISP Flash 存储单元，具有在系统可编程（ISP）特性，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，省去了购买通用编程器，而且速度更快。

STC89C52RC 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。

STC89C51 系列单片机的特点：

(1) 增强型 1T 流水线/ 精简指令集结构 8051 CPU

(2) 工作电压：3.4V - 5.5V （5V 单片机） / 2.0V - 3.8V （3V 单片机）

(3) 工作频率范围：0 - 35 MHz，相当于普通 8051 的 0～420MHz.实际工作频率可达 48MHz.

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

(5) 片上集成 512 字节 RAM

(6) 通用 I/O 口（27/23 个），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

每个 I/O 口驱动能力均可达到 20mA，但整个芯片最大不得超过 55mA

(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，数秒即可完成一片

(8) EEPROM 功能

(9) 看门狗

(10)内部集成 MAX810 专用复位电路（外部晶体 20M 以下时，可省外部复位电路）

(11)时钟源：外部高精度晶体/ 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：5.2MHz ～ 6.8MHz。精度要求不高时，可选择使用内部时钟，因为有温漂，请选 4MHz ～ 8MHz

(12)有 2 个 16 位定时器/ 计数器

(13)外部中断 2 路,下降沿中断或低电平触发中断,Power Down 模式可由外部中断低电平触发中断方式唤醒

(14)PWM （ 4 路）/ P C A（可编程计数器阵列），也可用来再实现 4 个定时器或 4 个外部中断(上升沿中断/ 下降沿中断均可支持)

(15) STC89Cc516AD 具有 ADC 功能。 10 位精度 ADC，共 8 路

(16) 通用异步串行口(UART)

(17) SPI 同步通信口， 主模式/ 从模式

(18)工作温度范围： 0 - 75℃ / -40 - +85℃

(19)封装： PDIP-28，SOP-28，PDIP-20，SOP-20，PLCC-32,TSSOP-20(超小封状，定货)

STC89C52RC 系列单片机为真正的看门狗，缺省为关闭（冷启动），启动后无法关闭，可省去外部看门狗。此系列单片机 P4 口地址为 E8H，并有 2 个附加外部中断，P4.2/INT3,P4.3/INT2。

晶振电路部分，使用 11.0592M 晶体，和 20PF 的电容。

在复位电路中，采用阻容复位时，电容为 10uF，电阻为 10k；晶振及复位电路如图 2.1。因为 STC89C52RC 系列单片机 RESET 脚内部没有下拉电阻，必须接 10k 电阻。

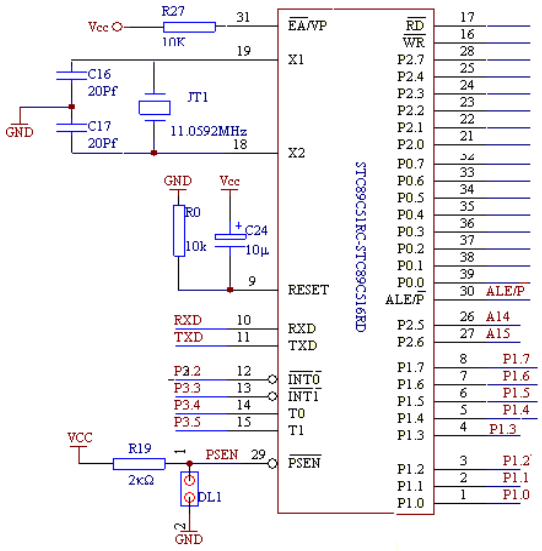


图 2.1 晶振及复位电路

2.3 电源电路：

电源电路采用外部供电的方式，通过变压器将 220V 交流电转变为 12V，再通过接口 J0 向实验板供电，为保护系统的安全性，增加了开关 k0,防止因电源不当引起硬件的烧坏，电源经过 k0 后，经过整流桥，再通过电源芯片 7805 和 7809 得到+5V 和+9V，为系统及周围芯片提供电源。电源供电原理图如图 2.2

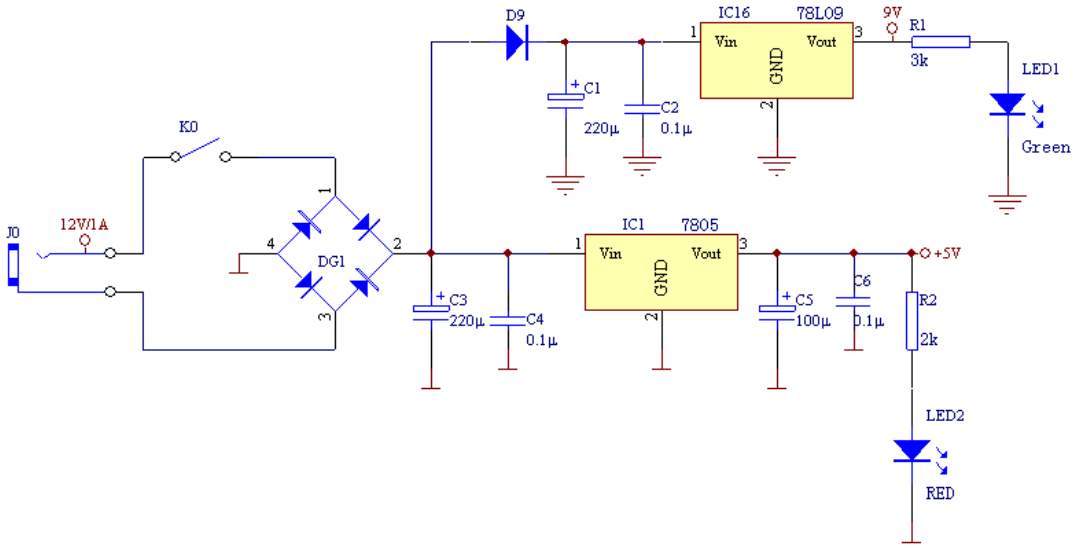


图 2.2 电源供电原理图

2.4 报警器电路

报警器电路如图 2.3 所示，使用三极管 9012 进行驱动控制。用单片机控制引脚 P3.2 控制报警器工作，实现报警功能。

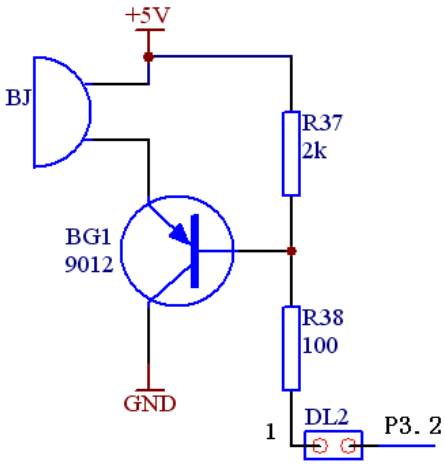


图 2.3 报警器电路与 STC89C52RC 的连接

2.5 按键

系统设计有三路独立的输入按键，按键直接接入到单片机的 P1 口，键盘电路如图 2.4 所示。当按键未按下时，由于上拉电阻的作用，单片机检测到引脚为高电平；当按键被按下时，单片机检测到引脚为低电平。所以只要通过检测相应端口的状态的变化，就可以检测到是否有按键按下。

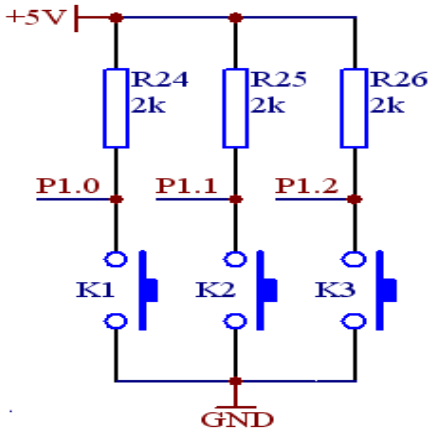


图 2.4 键盘电路与 STC89C52RC 的连接

2.6 跑马灯电路

系统跑马灯模块设计中，发光二极管 LED 作为指示器件，用亮或灭或是颜色的变化来告诉系统的或某个模块的工作状态。 在该系统跑马灯模块设计中，由于考虑到 P2 口使用的外围器件比较多，同时使用时，可能使端口的驱动能力下降。为了使以后 P2 口更易于扩展，在 P2 口加入缓冲驱动器 SN7407，提高 P2 口的驱动能力。跑马灯电路如图 2.5 所示：

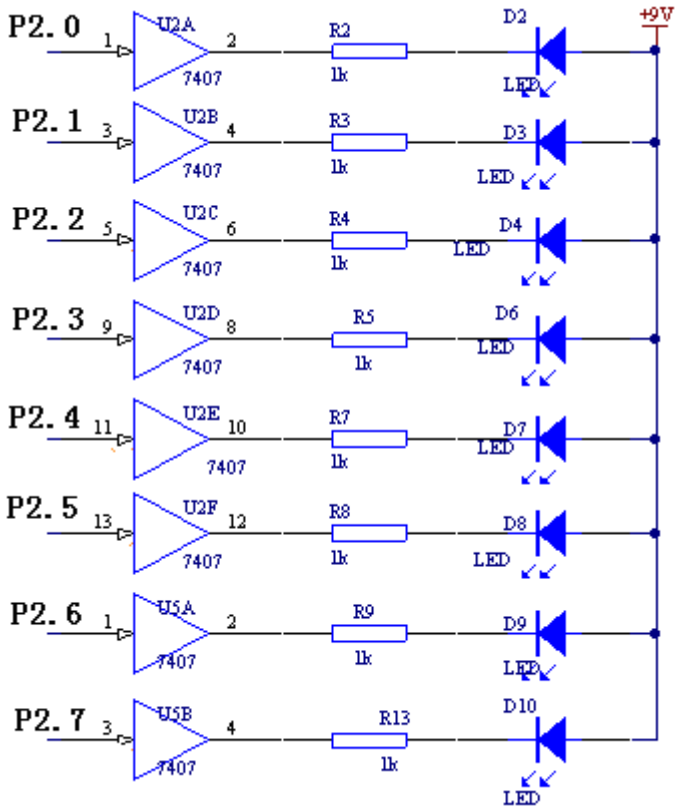


图 2.5 跑马灯电路与 STC89C52RC 的连接

2.7 串行通信模块

2.7.1. RS232 接口电路

系统设计了 RS232 接口电路，来实现系统与 PC 机串口通讯。在此系统中 RS232 接口电路主要用来将用户程序下载进控制器。用户通过 USB 线将程序代码送入 RS232 串口 J9，经 MAX232 将程序下载进单片机。接线方法如图 2.6.1。

用户也可在自己的目标系统上，可将 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“STC12C5410AD 系列单片机 ISP 编程 原理工具使用说明” 部分。

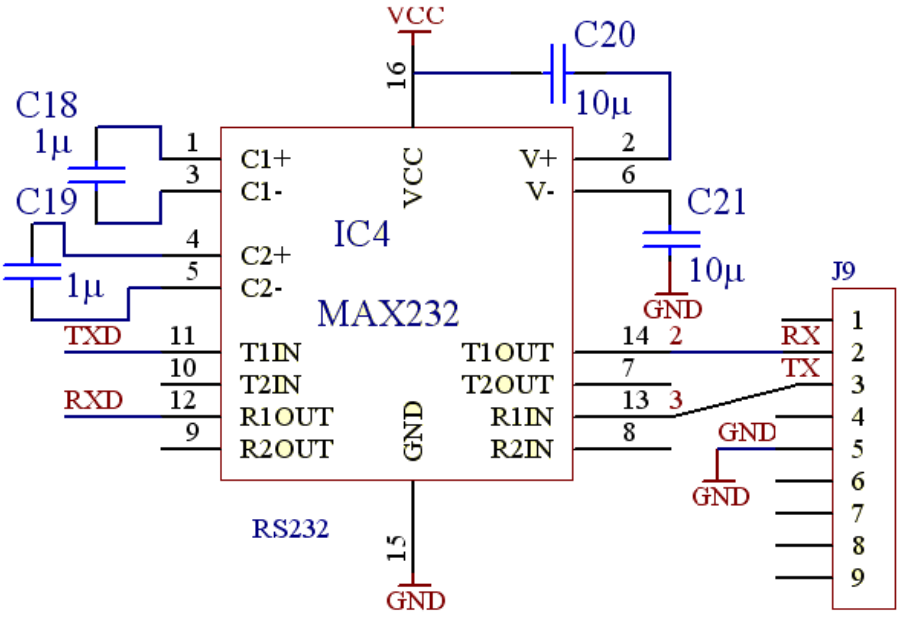


图 2.6.1 RS232、与 TTL 电平转换电路与 STC89C52RC 的连接

2.7.2. RS485 接口电路

RS485 是一个半双工通信的接口电路，其电路采用 MAX485。利用 RS485 接口可方便实现多一机对多机的组网通信。P3.5 为收发控制脚。电路如图 2.6.2

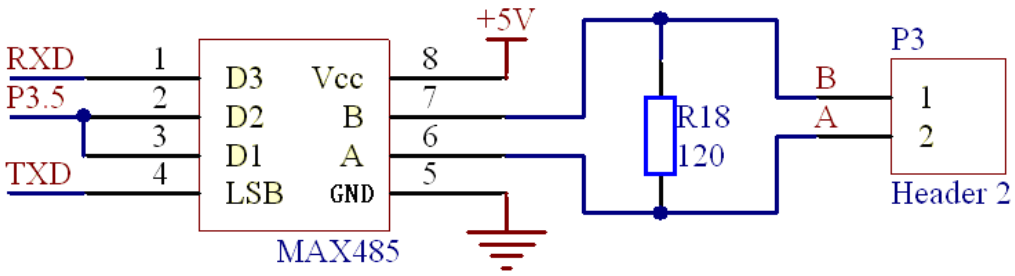


图 2.6.2　RS485 接口电路与 STC89C52RC 的连接

2.8 液晶显示模块

系统设计中，液晶显示采用 SMS0801 LCM 液晶屏。SMS0801 LCM 可以显示 8 位带小数点数字，采用串行接口，使用方便，只需将 1，2 脚接电源地，3 脚接单片机 P1.6，4，5 脚接电源，6 脚接单片机 P1.3 即可实现显示。如图 2.7 所示。

表 1　SMS0801 LCM 使用说明

一．　主要技术参数

显示容量	8 位带小数点数字		
芯片工作电压	2.7v~5.5v		
工作电流	20uA(3.0V)		
字　　高	10.7mm	环境相对湿度	<85%
视　　角	6:00	工作温度	-10～+50℃
显示方式	反射式正显式	存储温度	-20～+60℃

接口方式	串行接口	

二．接口信号说明

1 VSS:	电源地	2 VSS:	电源地
3 CLK:	串行移位脉冲输入	4 VDD:	电源正极
5 VDD:	电源正极输入	6 DI:	串行数据输入

三、SMS0401 地址映射表

LCDB UF	D7	D6	D5	D4	D3	D2	D1	D0
0	A1	B1	C1	D1	E1	F1	G1	H1
1	A2	B2	C2	D2	E2	F2	G2	H2
2	A3	B3	C3	D3	E3	F3	G3	H3
3	A4	B4	C4	D4	E4	F4	G4	H4
4	A5	B5	C5	D5	E5	F5	G5	H5
5	A6	B6	C6	D6	E6	F6	G6	H6
6	A7	B7	C7	D7	E7	F7	G7	H7
7	A8	B8	C8	D8	E8	F8	G8	H8

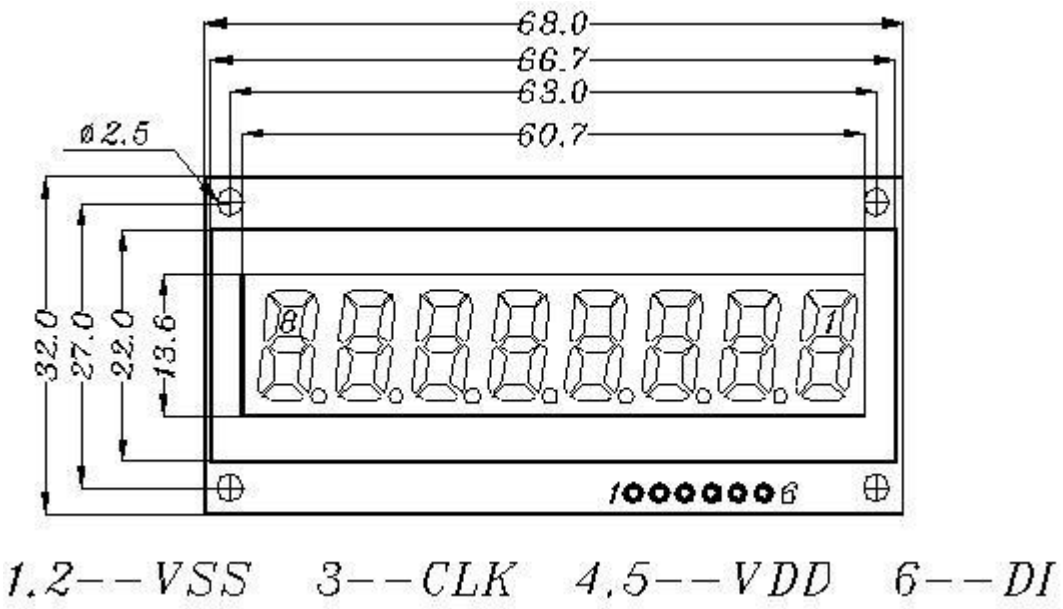


图 2.7 SMS0801 的结构图

2.9 电机控制模块

电机控制模块中采用驱动二相和四相步进电机的专用芯片 L298 来完成，其内部包含 4 通道逻辑驱动电路，具有双电机控制功能，内含二个 H 桥的高电压大电流双全桥式驱动器，接收标准 TTL 逻辑电平信号，可驱动 46V、2A 以下的电机。

这种驱动电路可以很方便实现直流电机的四象限运行，分别对应正转、正转制动、反转、反转、制动等功能。

为了进一步将功能扩展，将L298的输出out1, out2, out3, out4经扩展口J6, J7扩展出来。电机控制电路接线图如图2.8所示。表2-2 为电机控制功能表

表2-2　电机控制功能表

输入			电机状态	备注
Ven(6 脚、11 脚)	C(5 脚、10 脚)	D(7 脚、12 脚)		
H	H	L	正转	电机转动
	L	H	反转	
	0	0	停止	电机被制动
	1	1	停止	
L	X（任意）	X（任意）	停止	电机不受控制

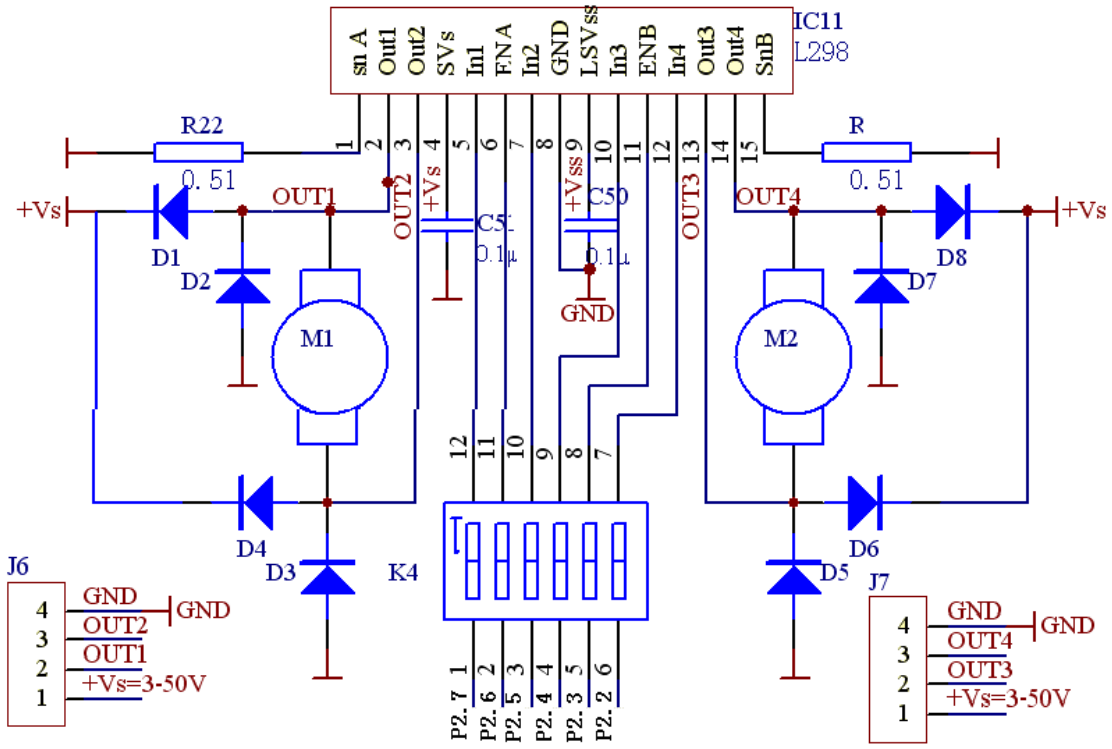


图2.8　电机控制电路与STC89C52RC的连接

2.10 A/D模块

系统使用12位模数转换器TLV2543来实现，TLVC2543采用串行接口，具有11路输入，有三个控制输入端为CS(片选)、输入/输出时钟(I/O CLOCK)以及串行数据输入端(DATA INPUT)。片内的14通道多路器可以选择11个输入中的任何一个或3个内部自测试电压中的一个，采样－保持是自动的，转换结束，EOC输出变高。

主要特性如下：

11个模拟输入通道；66ksps的采样速率；

最大转换时间为10μs；SPI串行接口；

线性度误差最大为±1LSB；

低供电电流(1mA典型值)；

TLC2543与STC89C52的连接如图2.9所示。　TLC2543的I/O时钟、数据输入、片选

信号由 P2.1、P2.2、P2.0 提供,转换结果由 P2.3 口串行读出。另外将 11 路输入端接 J3 扩展接口，以便信号输入。

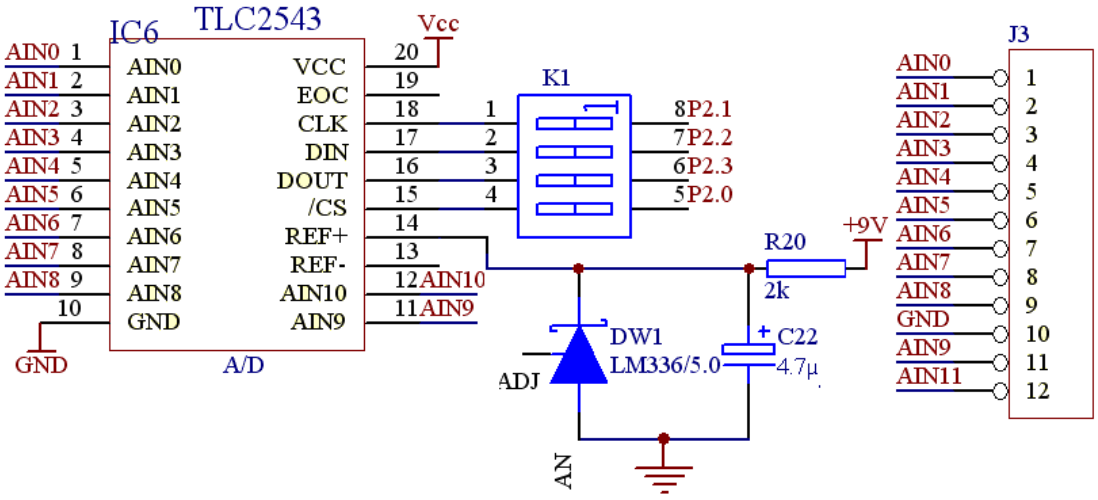


图 2.9　A/D 转换电路与 STC89C52RC 的连接

2.11　D/A 模块

TLV5615 是一个 10 位电压输出数模转换器（DAC），包括 4 个控制位和 12 个数据位的 16 位字符串来编程，可以用于宽范围的电源电压：2.7V 至 5.5V。

引脚说明如下：

DIN：串行数字数据输入；

SCLK:串行数字时钟输入；

/CS：片选。数字输入，用来使能和禁止输入，低电平有效；

FS：帧同步，数字输入用于 4 线串行接口；

AGND：模拟地

REFIN：基准模拟电压输入；

OUT：DAC 模拟输出；

Vdd：电源；

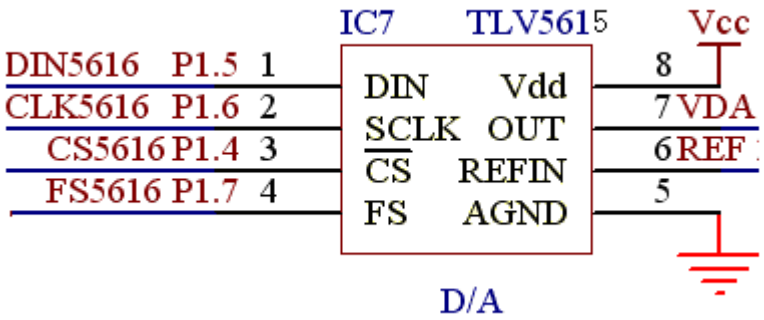


图 2.10　D/A 转换电路与 STC89C52RC 的连接

2.12 温度采集模块

温度采集部分采用单总线温度采集芯片 DS18B20,DS18B20 独特的单线接口仅需要一个端口引脚进行通讯。测量温度范围为 -55$^{\circ}C$~+125$^{\circ}C$，在-10~+85$^{\circ}C$ 范围内,精度为±0.5$^{\circ}C$。。

现场温度直接以“一线总线”的数字方式传输，大大提高了系统的抗干扰性。支持 3V~5.5V 的电压范围，

接线方法如图 2.11 所示：DQ 为数字信号输入/输出端；GND 为电源地；VDD 为外接供电电源输入端。

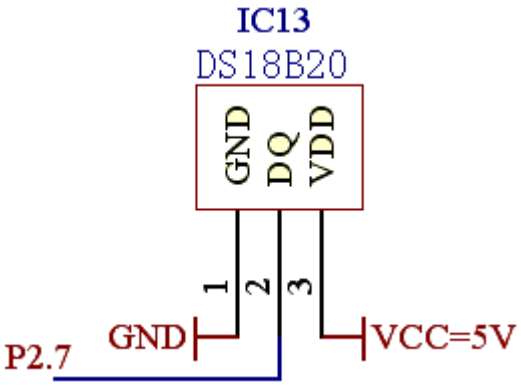


图 2.11　DS18B20 与 STC89C52RC 的连接

2.13　数字频率合成电路

直接数字信号合成部分采用 180MHZ 直接数字频率合成器（ DDS）AD9851 来实现，AD9581 是一个高度集成的器件，它是用先进的 DDS 技术，内部有一个高速的高性能的 D/A 转换器和比较器以构成一个可数字编程的频率发生器和时钟功能。当作为准确的时钟参考时，AD9851 可以通过数字编程产生一个具有稳定的频率和相角的模拟正弦函数波形输出信号。产生的正弦波形可被直接用作一个频率信号，或者可在内部转换为一个方波信号，可作为灵活调节的时钟发生器。AD9851 创新的高速 DDS 内　核，可接受 32 位的可调频率的信息，可以产生一个大约 0.04HZ 的调谐输出信号并伴有 180MHZ 的系统时钟。AD9851 包含一个特殊的 $6\times REFCLK$ 乘法器电路，因此不需要高速的振荡器。这个 $6\times REFCLK$ 乘法器对 SFDR 只有很微弱的冲击和很小的相角噪声特点。AD9851 提供了 5 位可编程的相角调节方案，使相角能随输出信号的增加 11.25° 。接线图如图 2.12。

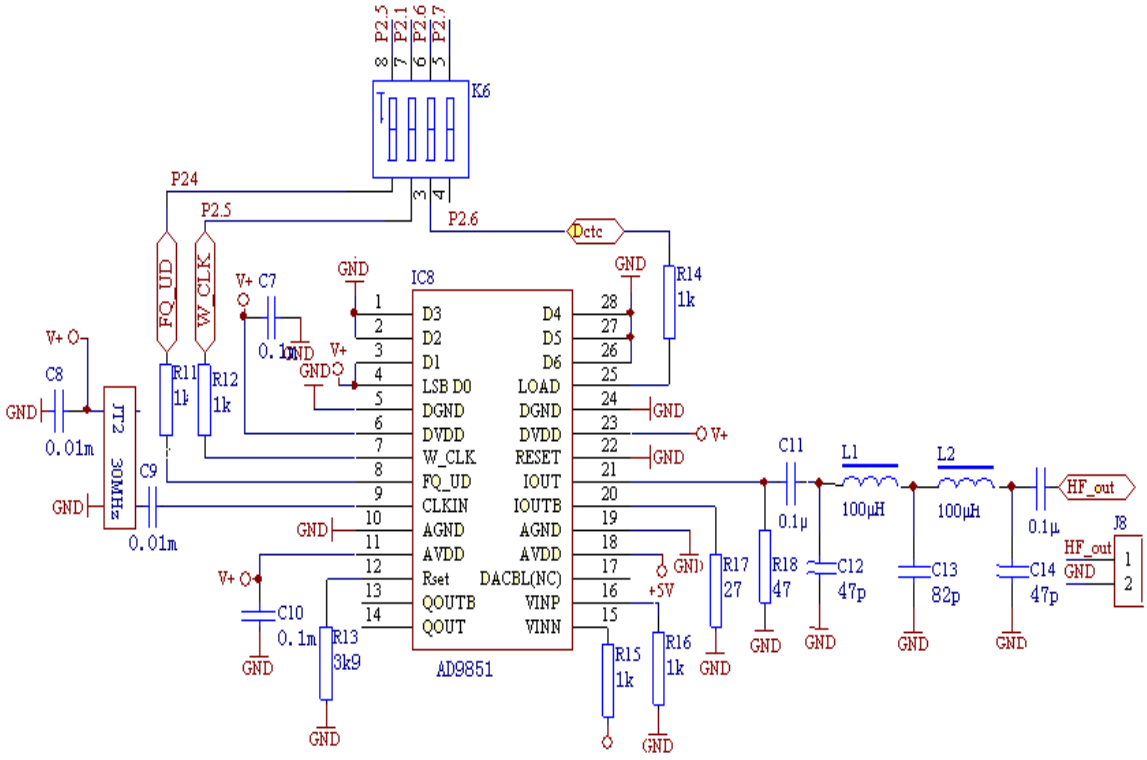
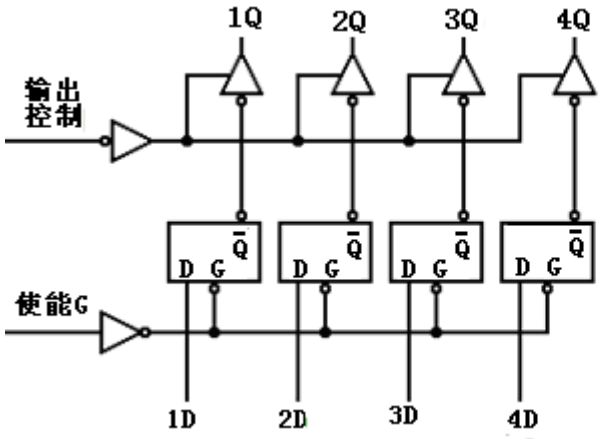


图 2.12　DDS 数字频率合成器连接电路与 STC89C52RC 的连接

2.14　锁存器及其外扩接口

系统通过锁存器 74HCS373 暂存单片机输出信号，74LS373 包含了 8 个具有三态输出的 D 型锁存器。当 LE 为高电平时，数据从 D_n 输入端进入锁存器。8 位锁存器 74LS373 的逻辑图见图所示 2.13.1。其中使能端 G 加入 CP 信号，D 为数据信号。输出控制信号为 0 时，锁存器的数据通过三态门进行输出。输出后通过外扩 接口 J3 将数据引出。电路原理图如图 2.13.2。



功 能 表

输出控制	G	D	输出
0	1	1	1
0	1	0	0
0	0	×	Q^n
1	×	×	高 阻

图 2.13.1 74HC373 原理图及功能

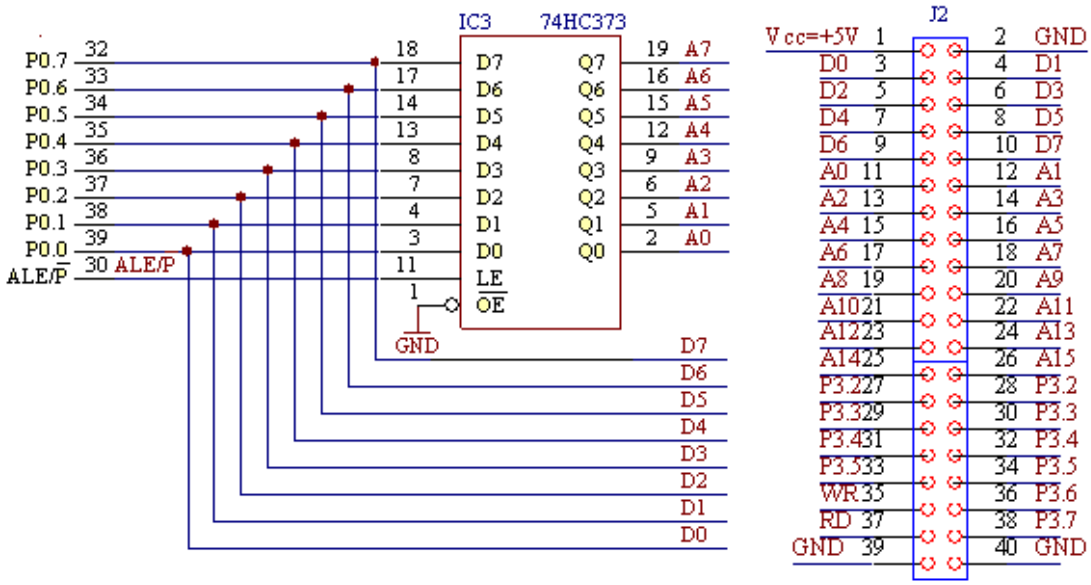


图 2.13.2　74HC373 及对外扩展电路与 STC89C52RC 的连接

2.15　E² PROM 电路

系统设计了一个 E²PROM 电路,用于存储数据,该电路使用 24LC02B 芯片,可以存储 2Kbit 数据，是电可擦除的 PROM。由 256*8 位的记忆单元与两线串口单元相连接。电路如图 2.14 所示。

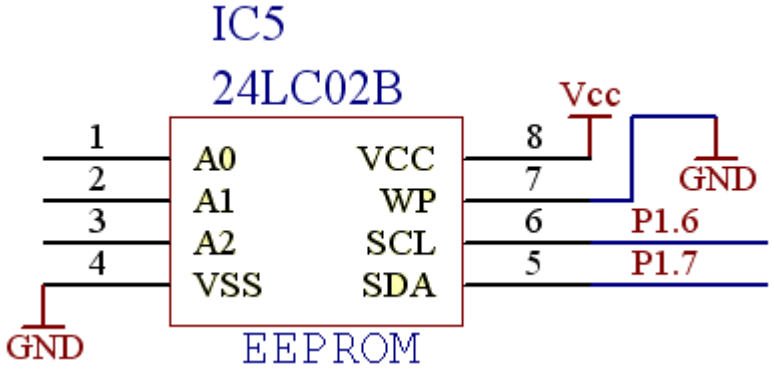


图 2.14 EEOROM 电路与 STC89C52RC 的连接

2.16 实时时钟电路

系统使用 DALLAS 公司的具有涓细电流充电能力的低功耗实时时钟电路 DS1302。它可以对年、月、日、周、时、分、秒进行计时，且具有闰年补偿等多种功能。DS1302 的主要特点是采用串行数据传输，可为掉电保护电源提供可编程的充电功能，并且可以关闭充电功能。采用普通 32.768kHz 晶振。工作电压为 2.5V～5.5V。 DS1302 的引脚排列,其中 Vcc1 为后备电源，Vcc2 为主电源。在主电源关闭的情况下，也能保持时钟的连续运行。DS1302 由 Vcc1 或 Vcc2 两者中的较大者供电。当 Vcc2 大于 Vcc1＋0.2V 时，Vcc2 给 DS1302 供电。当 Vcc2 小于 Vcc1 时，DS1302 由 Vcc1 供电。X1 和 X2 是振荡源，外接 32.768kHz 晶振。RST 是复位/片选线，通过把 RST 输入驱动置高电平来启动所有的数据传送。RST 输入有两种功能：首先，RST 接通控制逻辑，允许地址/命令序列送入移位寄存器；其次，RST 提供终止单字节或多字节数据的传送手段。当 RST 为高电平时，所有的数据传送被初始化，允许对 DS1302 进行操作。

如果在传送过程中 RST 置为低电平，则会终止此次数据传送，I/O 引脚变为高阻态。上电运行时，在 Vcc≥2.5V 之前，RST 必须保持低电平。只有在 SCLK 为低电平时，才能将 RST 置为高电平。I/O 为串行数据输入输出端(双向)， SCLK 始终是输入端。

DS1302 有 12 个寄存器，其中有 7 个寄存器与日历、时钟相关，DS1302 还有年份寄存器、控制寄存器、充电寄存器、时钟突发寄存器及与 RAM 相关的寄存器等。 DS1302 与 CPU 的连接需要三条线，即 SCLK(7)、I/O(6)、RST(5)。其电路连接

如图 2.15 所示。

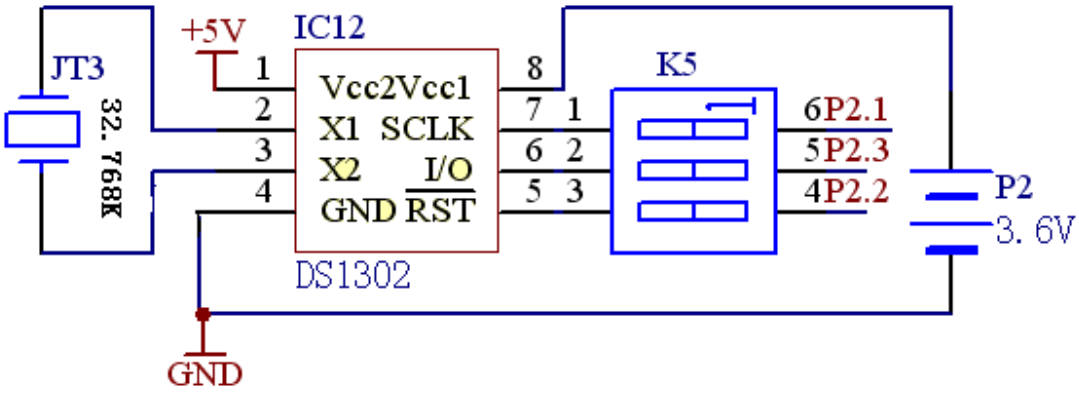


图 2.15　DS1302 实时时钟电路与 STC89C52RC 的连接

2.17 输出 4—20mA

系统为适应工业界的标准信号，还设计了可以产生 4～20mA 的电路。该部分主要采用双运算放大器 LM2904 来完成，并将输出的 4～20mA 电流信号通过扩展口 J5 引出以便备用。电路图如图 2.16 所示

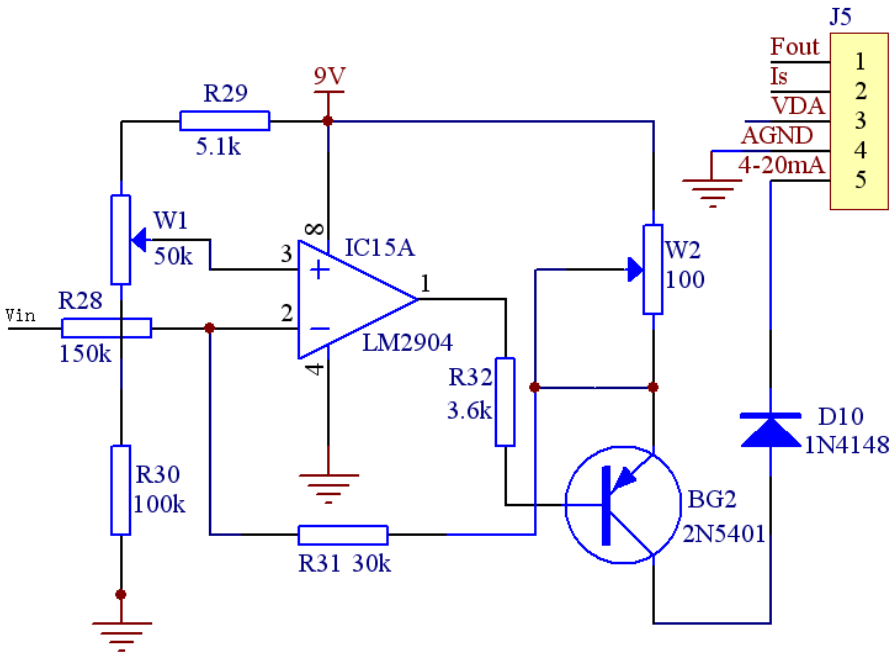


图 2.16　0-5V 到 4-20mA 转换电路

2.18 红外控制电路

系统还增加了红外遥控电路，该电路采用 KSM-6031LM 来实现。KSM-6031LM 是由一个 PIN 高速光电二极管和一个前置放大芯片封装而成的红外遥控接收器。它兼容 TTL 和 CMOS，接线简

单，不易受外界干扰，接线简单，如图 2.17.

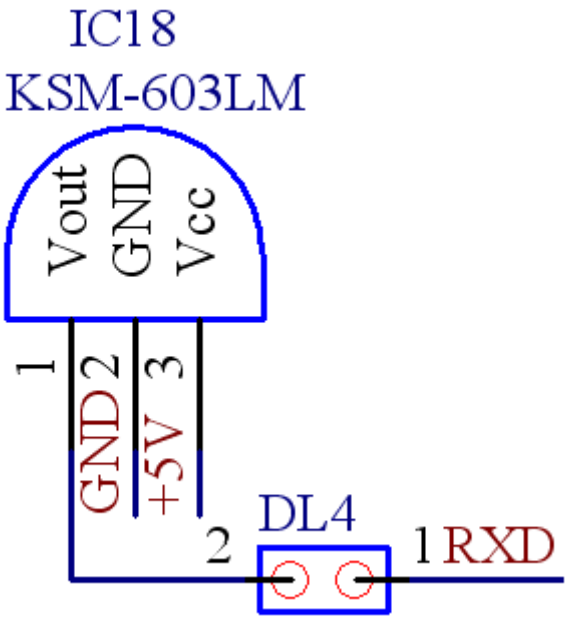
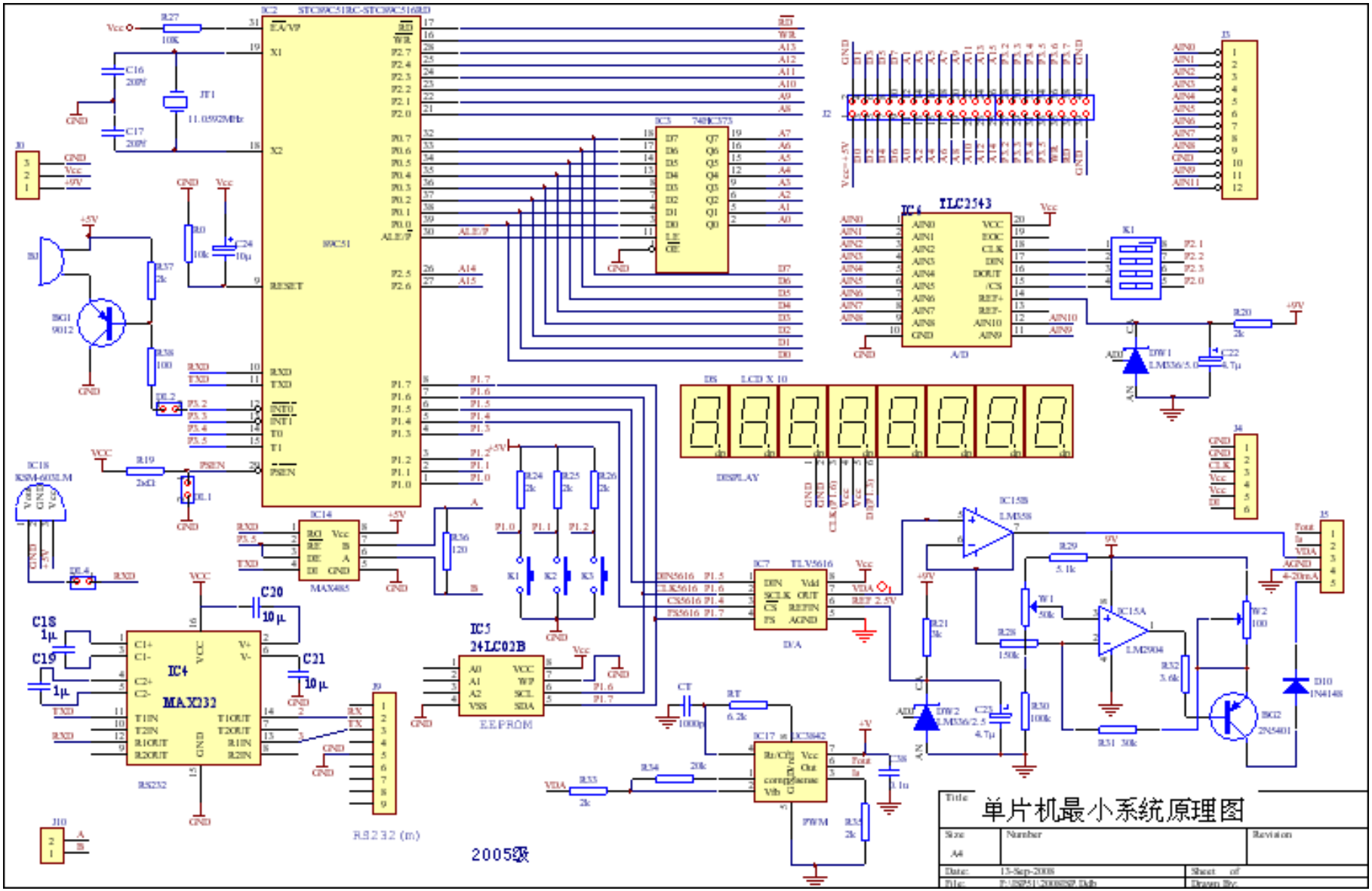
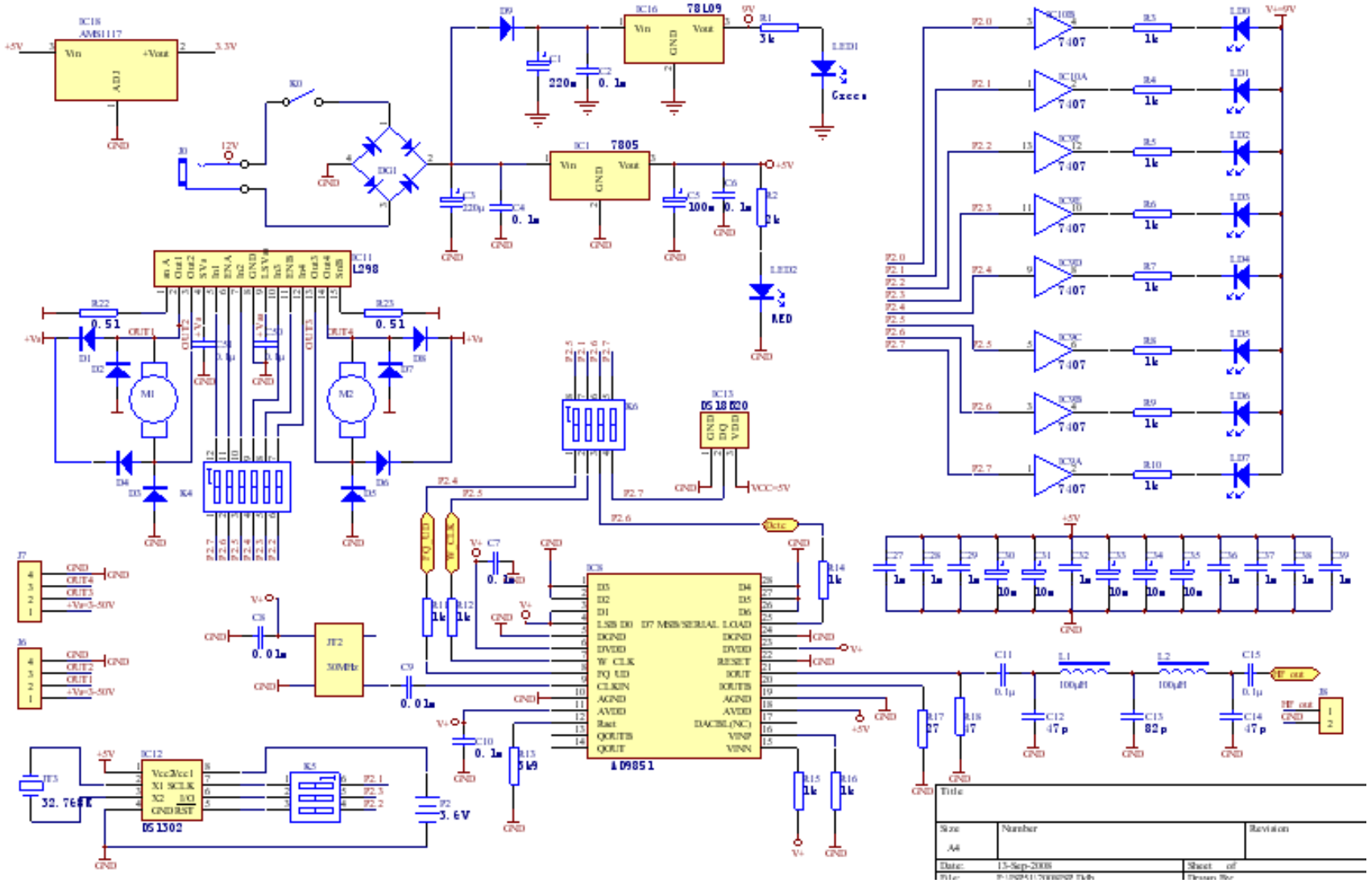


图 2.17 红外接收（KSM-603LM）与 STC89C52RC 的连接

2．19 系统原理图


单片机最小系统原理图
2005级
IC4 TLC2543
IC4 MAX232
IC5 24LC02B
EEPROM
IC7 TLV5616
D/A
IC15B LM358
IC15A LM2904
IC14 MAX485
IC3 74HC373
IC2 STC89C51RC-STC89C516RD
89C51
A/D
DISPLAY
LCD X 10
RS232
RS232 (m)
Title
Size
A4
Number
Revision
Date: 13-Sep-2008
Sheet of
Drawn By:


IC18
AMS1117
Vin
+Vout
ADJ
+5V
3.3V
GND
K0
12V
J0
DG1
D9
IC16
78L09
9V
R1
3k
LED1
C1
220u
C2
0.1u
IC1
7805
+5V
C3
220u
C4
0.1u
C5
100u
C6
0.1u
R2
2k
LED2
RED
IC11
L298
R22
0.51
R23
0.51
OUT1
OUT2
OUT3
OUT4
M1
M2
D1
D2
D3
D4
D5
D6
D7
D8
K4
K3
R6
IC13
DS18B20
GND
DQ
VDD
VCC=5V
P2.4
P2.5
P2.6
P2.7
FQ_UD
W_CLK
Vcc
IC8
AD9851
D0
D1
D2
D3
D4
D5
D6
D7
LSB D0
MSB/SERIAL
DGND
DVDD
W_CLK
FQ_UD
CLKIN
AGND
AVDD
Rset
QOUTB
QOUT
RESET
IOUT
IOUTB
AGND
AVDD
DACBL(NC)
VINP
VINN
C7
0.1u
C8
0.01u
JT2
30MHz
C9
0.01u
C10
0.1u
R13
5k9
R11
1k
R12
1k
R14
1k
R15
1k
R16
1k
R17
27
R18
47
C11
0.1u
L1
100μH
L2
100μH
C15
0.1u
C12
47p
C13
82p
C14
47p
RF out
J8
IC12
Vcc2/Vcc1
X1
SCLK
X2
I/O
GND
RST
DS1302
JT3
32.768K
JP2
3.6V
K5
IC10B
IC10A
IC9F
IC9E
IC9D
IC9C
IC9B
IC9A
7407
R3
R4
R5
R6
R7
R8
R9
R10
1k
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
V+=9V
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C39
1n
10n
J7
J6
ORUT4
ORUT3
ORUT2
ORUT1
+Vs=5-50V
Title
Size
A4
Number
Revision
Date:
13-Sep-2008
Sheet
of

3．KEIL51 及 C 语言入门

学习单片机实在不是件易事，一来要购买高价格的编程器，仿真器，二来要学习编程语言，还有众多种类的单片机选择真是件让人头痛的事。在众多单片机中 51 架构的芯片风行很久，学习资料也相对很多，是初学的较好的选择之一。51 的编程语言常用的有二种，一种是汇编语言，一种是 C 语言。汇编语言的机器代码生成效率很高但可读性却并不强，复杂一点的程序就更是难读懂，而 C 语言在大多数情况下其机器代码生成效率和汇编语言相当，但可读性和可移植性却远远超过汇编语言，而且 C 语言还可以嵌入汇编来解决高时效性的代码编写问题。对于开发周期来说，中大型的软件编写用 C 语言的开发周期通常要小于汇编语言很多。综合以上 C 语言的优点，我在学习时选择了 C 语言。以后的教程也只是我在学习过程中的一些学习笔记和随笔，在这里加以整理和修改，希望和大家一起分享，一起交流，一起学习，一起进步。

3.1 建立您的第一个 C 项目

使用 C 语言肯定要使用到 C 编译器，以便把写好的 C 程序编译为机器码，这样单片机才能执行编写好的程序。KEIL uVISION2 是众多单片机应用开发软件中优秀的软件之一，它支持众多不同公司的 MCS51 架构的芯片，它集编辑，编译，仿真等于一体，同时还支持，PLM，汇编和 C 语言的程序设计，它的界面和常用的微软 VC++的界面相似，界面友好，易学易用，在调试程序，软件仿真方面也有很强大的功能。因此很多开发 51 应用的工程师或普通的单片机爱好者，都对它十分喜欢。

以上简单介绍了 KEIL51 软件，要使用 KEIL51 软件，必需先要安装它。KEIL51 是一个商业的软件，对于我们这些普通爱好者可以到 KEIL 中国代理周立功公司的网站上下载一份能编译 2K 的 DEMO 版软件，基本可以满足一般的个人学习和小型应用的开发。（安装的方法和普通软件相当这里就不做介绍了）安装好后，您是不是迫不及待的想建立自己的第一个 C 程序项目呢？下面就让我们一起来建立一个小程序项目吧。或许您手中还没有一块实验板，甚至没有一块单片机，不过没有关系我们可以通过 KEIL 软件仿真看到程序运行的结果。

首先当然是运行 KEIL51 软件。怎么打开？噢，天！那您要从头学电脑了。呵呵，开个玩笑，这个问题我想读者们也不会提的了：P。运行几秒后，出现如图 1－1 的屏幕。

图 3.1 启动时的屏幕

接着按下面的步骤建立您的第一个项目：

（1）点击 Project 菜单，选择弹出的下拉式菜单中的 New Project，如图 1－2。接着弹出一个标准 Windows 文件对话窗口，如图 1－3，这个东东想必大家是见了 N 次的了，用法技巧也不是这里要说的，以后的章节中出现类似情况将不再说明。在“文件名”中输入您的第一个 C 程序项目名称，这里我们用“test”，这是笔者惯用的名称，大家不必照搬就是了，只要符合 Windows 文件规则的文件名都行。“保存”后的文件扩展名为 uv2，这是 KEIL uVision2 项目文件扩展名，以后我们可以直接点击此文件以打开先前做的项目。

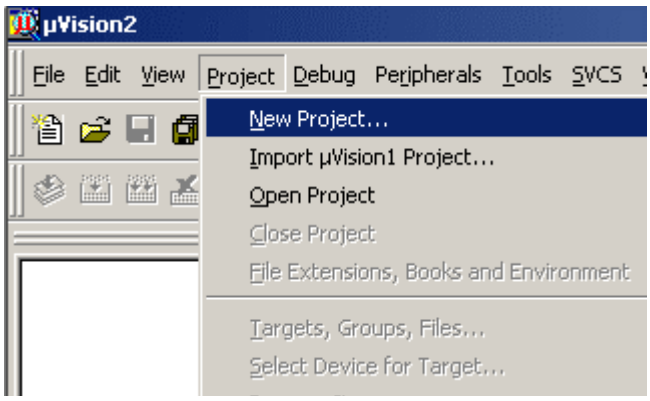


图 2.2 New Project 菜单

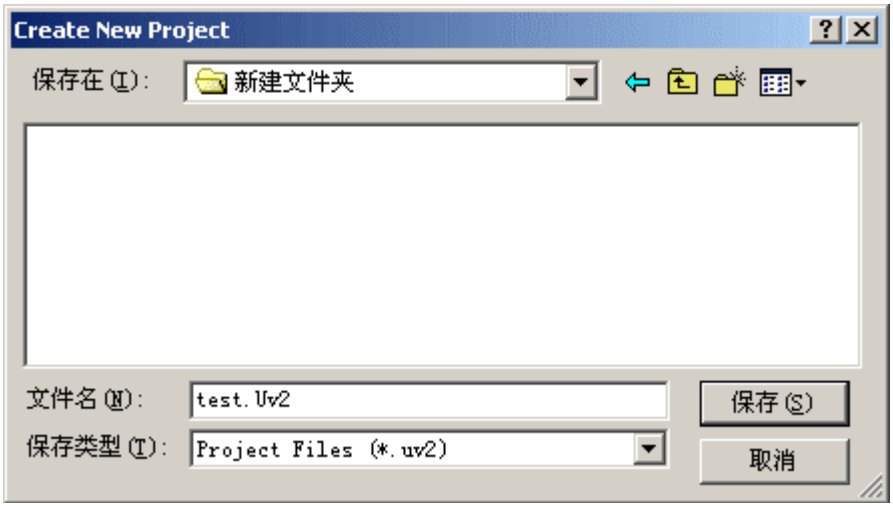


图 2.3 文件窗口

（2）选择所要的单片机，这里我们选择常用的 Ateml 公司的 AT89C51。此时屏幕如图 1－4 所示。AT89C51 有什么功能、特点呢？不用急，看图中右边有简单的介绍，稍后的章节会作较详细的介绍。完成上面步骤后，我们就可以进行程序的编写了。

（3）首先我们要在项目中创建新的程序文件或加入旧程序文件。如果您没有现成的程序，那么就要新建一个程序文件。在 KEIL 中有一些程序的 Demo，在这里我们还是以一个 C 程序为例介绍如何新建一个 C 程序和如何加到您的第一个项目中吧。点击图 1－5 中 1 的新建文件的快捷按钮，在 2 中出现一个新的文字编辑窗口，这个操作也可以通过菜单 File－New 或快捷键 Ctrl+N 来实现。好了，现在可以编写程序了，光标已出现在文本编辑窗口中，等待我们的输入了。第一程序嘛，写个简单明了的吧。下面是经典的一段程序，如果您看过别的程序书也许也有类似的程序：

```
#include <AT89X51.H>
#include <stdio.h>
void main(void)
{
SCON = 0x50; //串口方式 1,允许接收
TMOD = 0x20; //定时器 1 定时方式 2
TCON = 0x40; //设定时器 1 开始计数
TH1 = 0xE8; //11.0592MHz 1200 波特率
TL1 = 0xE8;
```

```
TI = 1;
TR1 = 1; //启动定时器
while(1)
{
printf ("Hello World!\n"); //显示 Hello World
}
}
```

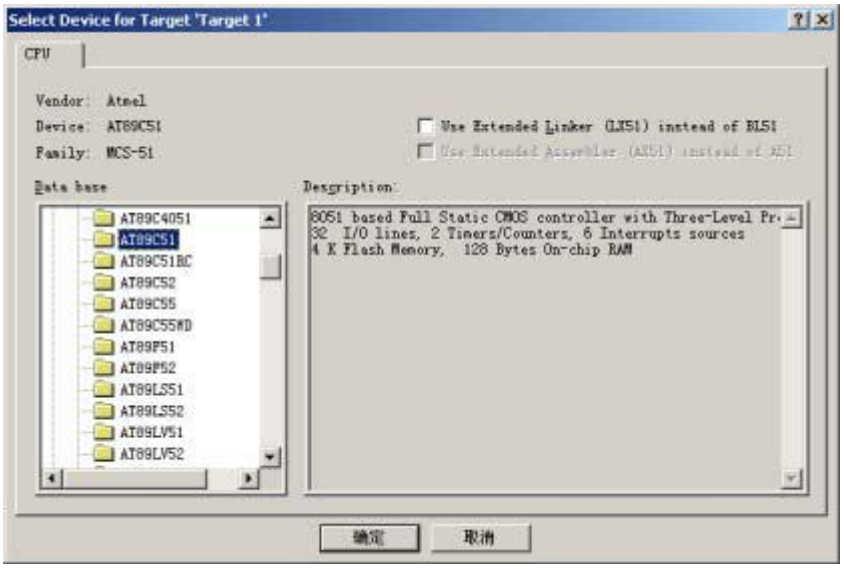


图 2.4 选取芯片

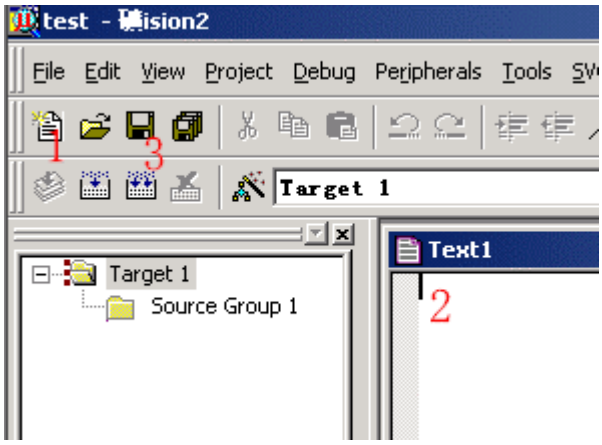


图 2.5 新建程序文件

这段程序的功能是不断从串口输出“Hello World!”字符，我们先不管程序的语法和意思吧，先看看如何把它加入到项目中和如何编译试运行。

（4)点击图 1－5 中的 3 保存新建的程序,也可以用菜单 File－Save 或快捷键 Ctrl+S 进行保存。因是新文件所以保存时会弹出类似图 1－3 的文件操作窗口，我们把第一个程序命名为 test1.c，保存在项目所在的目录中，这时您会发现程序单词有了不同的颜色，说明 KEIL 的 C 语法检查生效了。如图 1－6 鼠标在屏幕左边的 Source Group1 文件夹图标上右击弹出菜单，在这里可以做在项目中增加减少文件等操作。我们选“Add File to Group ‘Source Group 1’ ”弹出文件窗口，选择刚刚保存的文件，按 ADD 按钮，关闭文件窗，程序文件已加到项目中了。这时在 Source Group1 文件夹图标左边出现了一个小+号说明，文件组中有了文件，点击它可以展开查看。

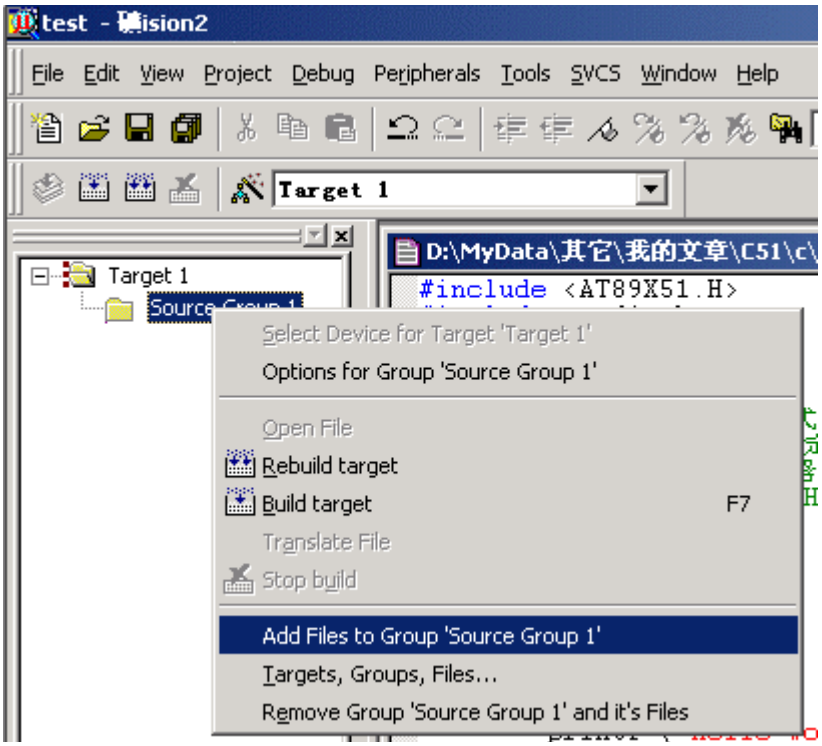


图 3.6 把文件加入到项目文件组中

（5）C 程序文件已被我们加到了项目中了，下面就剩下编译运行了。这个项目我们只是用做学习新建程序项目和编译运行仿真的基本方法，所以使用软件默认的编译设置，它不会生成用于芯片烧写的 HEX 文件，如何设置生成 HEX 文件就请看下面的第三课。我们先来看图 1－7 吧，图中 1、2、3 都是编译按钮，不同是 1 是用于编译单个文件。2 是编译当前项目，如果先前编译过一次之后文件没有做动编辑改动，这时再点击是不会再次重新编译的。3 是重新编译，每点击一次均会再次编译链接一次，不管程序是否有改动。在 3 右边的是停止编译按钮，只有点击了前三个中的任一个，停止按钮才会生效。在 4 中可以看到编译的错误信息和使用的系统资源情况等，以后我们要查错就靠它了。6 是有一个小放大镜的按钮，这就是开启\关闭调试模式的按钮，它也存在于菜单 Debug－Start\Stop Debug Session，快捷键为 Ctrl+F5。

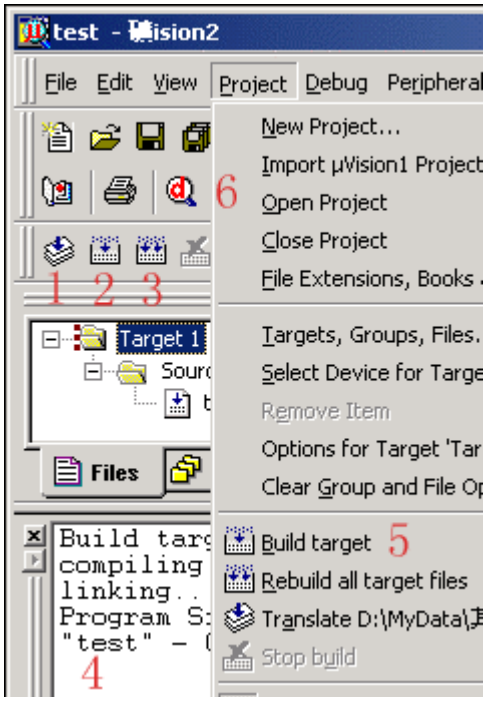


图 3.7 编译程序

(6)进入调试模式，软件窗口样式大致如图 1－8 所示。图中 1 为运行，当程序处于停止状态时才有效，2 为停止，程序处于运行状态时才有效。3 是复位，模拟芯片的复位，程序回到最开头处执行。按 4 我们可以打开 5 中的串行调试窗口，这个窗口我们可以看到从 51 芯片的串行口输入输出的字符，这里的第一个项目也正是在这里看运行结果。这些在菜单中也有，这里不再一一介绍大家不妨找找看，其它的功能也会在后面的课程中慢慢介绍。首先按 4 打开串行调试窗口，再按运行键，这时就可以看到串行调试窗口中不断的打印“HelloWorld!”。呵呵，是不是不难呀？这样就完成了您的第一个 C 项目。最后我们要停止程序运行回到文件编辑模式中，就要先按停止按钮再按开启\关闭调试模式按钮。然后我们就可以进行关闭 KEIL 等相关操作了。

到此为止，初步学习了一些 KEIL uVision2 的项目文件创建、编译、运行和软件仿真的基本操作方法。其中一直有提到一些功能的快捷键的使用，的确在实际的开发应用中快捷键的运用可以大大提高工作的效率，建议大家多多使用，还有就是对这里所讲的操作方法举一反三用于类似的操作中。

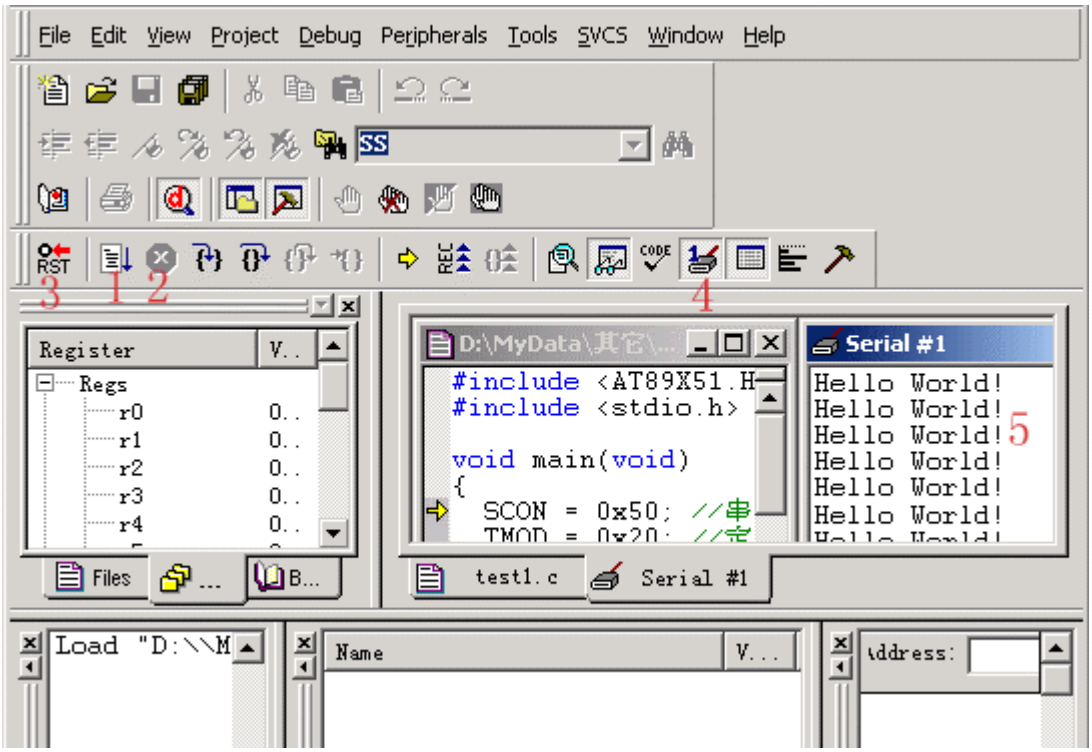


图 3.8 调试运行程序

3.2 初步认识 51 芯片

上一接我们的第一个项目完成了，可能有懂 C 语言的朋友会说，“这和 PC 机上的 C 语言没有多大的区别呀”。的确没有太大的区别，C 语言只是一种程序语言的统称，针对不同的处理器相关的 C 语言都会有一些细节的改变。编写 PC 机的 C 程序时，如要对硬件编程您就必须对硬件要有一定的认识，51 单片机编程就更是如此，因它的开发应用是不可与硬件脱节的，所以我们先要来初步认识一下 51 芯片的结构和引脚功能。MSC51 架构的芯片种类很多，具体特点和功能不尽相同（在以后编写的附录中会加入常用的一些 51 芯片的资料列表），

在此后的教程中就以 Atmel 公司的 AT89C51 和 AT89C2051 为中心对象来进行学习，两者是 AT89 系列的典型代表，在爱好者中使用相当的多，应用资料很多，价格便宜，是初学 51 的首选芯片。嘿嘿，口水多多有点卖广告之嫌了。：P

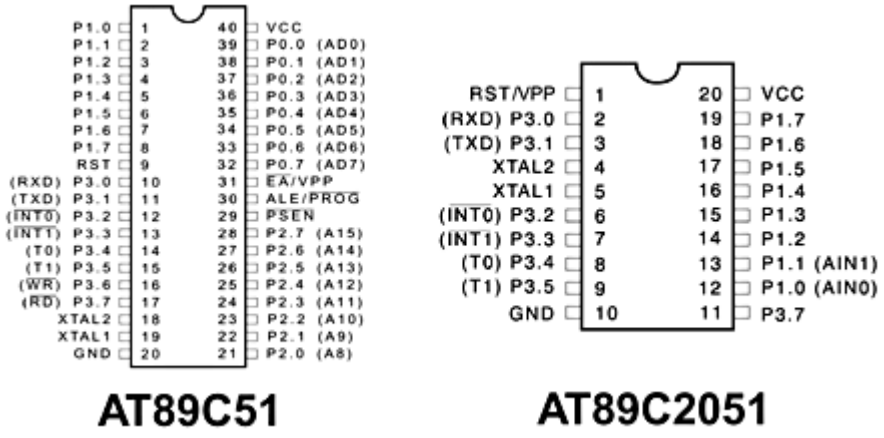


图 3.9 AT89C51 和 AT89C2051 引脚功能图

表 3.1 AT89C51 和 AT89C2051 主要性能表

AT89C51	AT89C2051
4KB 可编程 Flash 存储器（可擦写 1000 次）	2KB 可编程 Flash 存储器（可擦写 1000 次）
三级程序存储器保密	两级程序存储器保密
静态工作频率:0Hz-24MHz	静态工作频率:0Hz-24MHz
128 字节内部 RAM	128 字节内部 RAM
2 个 16 位定时/计数器	2 个 16 位定时/计数器
一个串行通讯口	一个串行通讯口
6 个中断源	6 个中断源
32 条 I/O 引线	15 条 I/O 引线
片内时种振荡器	1 个片内模拟比较器

图 3.9 中是 AT89C51 和 AT89C2051 的引脚功能图。而表 3.1 中则是它们的主要性能表。

以上可以看出它们是大体相同的，由于 AT89C2051 的 IO 线很少，导致它无法外加 RAM 和程序 ROM，片内 Flash 存储器也少，但它的体积比 AT89C51 小很多，以后大家可根据实际需要来选用。它们各有其特点但其核心是一样的，下面就来看看 AT89C51 的引脚具体功能。

1.电源引脚

Vcc 40 电源端, GND 20 接地端。

＊工作电压为 5V，另有 AT89LV51 工作电压则是 2.7-6V, 引脚功能一样。

2.外接晶体引脚

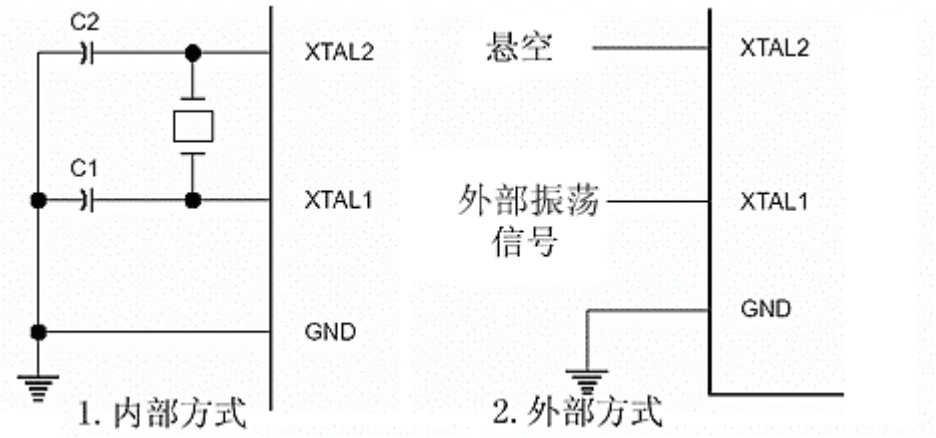


图 3.10 外接晶体引脚

XTAL1 19，XTAL2 18。

XTAL1 是片内振荡器的反相放大器输入端，XTAL2 则是输出端，使用外部振荡器时，外部振荡信号应直接加到 XTAL1，而 XTAL2 悬空。内部方式时，时钟发生器对振荡脉冲二分频，如晶振为 12MHz，时钟频率就为 6MHz。晶振的频率可以在 1MHz-24MHz 内选择。电容取 30PF 左右。

同样为 AT89C51 的芯片，在其后面还有频率编号，有 12,16,20,24MHz 可选。大家在购买和选用时要注意了。如 AT89C51 24PC 就是最高振荡频率为 24MHz,40P6 封装的普通商用芯片。

3.复位 RST 9

在振荡器运行时，有两个机器周期（24 个振荡周期）以上的高电平出现在此引腿时，

将使单片机复位，只要这个脚保持高电平，51 芯片便循环复位。复位后 P0－P3 口均置 1 引脚表现为高电平，程序计数器和特殊功能寄存器 SFR 全部清零。当复位脚由高电平变为低电平时，芯片为 ROM 的 00H 处开始运行程序。常用的复位电路如图 3.11 所示。

＊复位操作不会对内部 RAM 有所影响。

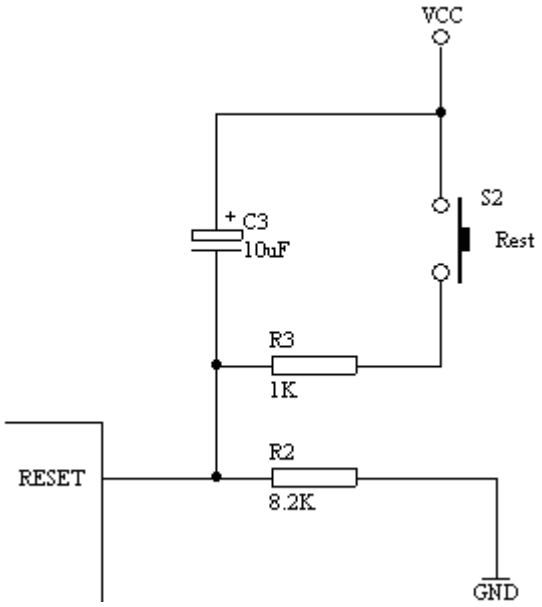


图 3.11 常用复位电路

4.输入输出引脚

(1) P0 端口[P0.0-P0.7] P0 是一个 8 位漏极开路型双向 I/0 端口，端口置 1（对端口写 1）时作高阻抗输入端。作为输出口时能驱动 8 个 TTL。对内部 Flash 程序存储器编程时，接收指令字节;校验程序时输出指令字节，要求外接上拉电阻。在访问外部程序和外部数据存储器时，P0 口是分时转换的地址(低 8 位)/数据总线，访问期间内部的上拉电阻起作用。

(2) P1 端口[P1.0－P1.7] P1 是一个带有内部上拉电阻的 8 位双向 I/0 端口。输出时可驱动 4 个 TTL。端口置 1 时，内部上拉电阻将端口拉到高电平，作输入用。对内部 Flash 程序存储器编程时，接收低 8 位地址信息。

(3) P2 端口[P2.0－P2.7] P2 是一个带有内部上拉电阻的 8 位双向 I/0 端口。输出时可驱动 4 个 TTL。端口置 1 时，内部上拉电阻将端口拉到高电平，作输入用。对内部 Flash 程序存储器编程时，接收高 8 位地址和控制信息。

在访问外部程序和 16 位外部数据存储器时，P2 口送出高 8 位地址。而在访问 8 位地址的外部数据存储器时其引脚上的内容在此期间不会改变。

(4) P3 端口[P3.0－P3.7] P2 是一个带有内部上拉电阻的 8 位双向 I/0 端口。输出时可驱动 4 个 TTL。端口置 1 时，内部上拉电阻将端口拉到高电平，作输入用。对内部 Flash 程序存储器编程时，接控制信息。除此之外 P3 端口还用于一些专门功能，具体请看 表 2－2.。

＊P1－3 端口在做输入使用时，因内部有上接电阻，被外部拉低的引脚会输出一定的电流。

表 3.2 P3 端口引脚兼用功能表

P3 引脚	兼用功能
P3.0	串行通讯输入 (RXD)
P3.1	串行通讯输出 (TXD)
P3.2	外部中断 0 ($\overline{INT0}$)
P3.3	外部中断 1 ($\overline{INT1}$)
P3.4	定时器 0 输入(T0)
P3.5	定时器 1 输入(T1)
P3.6	外部数据存储器写选通 $\overline{WR}$
P3.7	外部数据存储器写选通 $\overline{RD}$

什么叫上拉电阻？上拉电阻简单来说就是把电平拉高，通常用 4.7－10K 的电阻接到 Vcc 电源，下拉电阻则是把电平拉低，电阻接到 GND 地线上。具体说明也不是这里要讨论的，接下来还是接着看其它的引脚功能吧。

5.其它的控制或复用引脚

(1) ALE/PROG 30 访问外部存储器时，ALE（地址锁存允许）的输出用于锁存地址的低位字节。即使不访问外部存储器，ALE 端仍以不变的频率输出脉冲信号(此频率是振荡器频率的 1/6)。在访问外部数据存储器时，出现一个 ALE 脉冲。对 Flash 存储器编程时，这个引脚用于输入编程脉冲 PROG。

(2) PSEN 29 该引是外部程序存储器的选通信号输出端。当 AT89C51 由外部程序存储器取指令或常数时，每个机器周期输出 2 个脉冲即两次有效。但访问外部数据存储器时，将不会有脉冲输出。

(3) EA/Vpp 31 外部访问允许端。当该引脚访问外部程序存储器时，应输入低电平。要使 AT89C51 只访问外部程序存储器（地址为 0000H-FFFFH),这时该引脚必须保持低电平。对 Flash 存储器编程时，用于施加 Vpp 编程电压。Vpp 电压有两种，类似芯片最大频率值要根据附加的编号或芯片内的特征字决定。具体如表 3.3 所列。

表 3.3 Vpp 与芯片型号和片内特征字的关系

	Vpp = 12V		Vpp = 5V	
印刷在芯片面上的型号	AT89C51 xxxx YYWW	AT89LV51 xxxx YYWW	AT89C51 xxxx-5 YYWW	AT89LV51 xxxx-5 YYWW
片内特征字	030H=1EH	030H=1EH	030H=1EH	030H=1EH
	031H=51H	031H=61H	031H=51H	031H=61H
	032H=FFH	032H=FFH	032H=05H	032H=05H

看到这您对 AT89C51 引脚的功能应该有了一定的了解了，引脚在编程和校验时的时序我们在这里就不做详细的探讨，通常情况下我们也没有必要去撑握它，除非您想自己开发编程器。下来的课程我们要开始以一些简单的实例来讲述 C 程序的语法和编写方法技巧，中间穿插相关的硬件知识如串口，中断的用法等等。

3.3 生成 HEX 文件和最小化系统

在开始 C 语言的主要内容时，我们先来看看如何用 KEIL uVISION2 来编译生成用于烧写芯片的 HEX 文件。HEX 文件格式是 Intel 公司提出的按地址排列的数据信息,数据宽度为字节,所有数据使用 16 进制数字表示, 常用来保存单片机或其他处理器的目标程序代码。它

保存物理程序存储区中的目标代码映象。一般的编程器都支持这种格式。我们先来打开第一课做的第一项目，打开它的所在目录，找到 test.Uv2 的文件就可以打开先前的项目了。然后右击图 3.10 中的 1 项目文件夹，弹出项目功能菜单，选 Options for Target’Target1’，弹出项目选项设置窗口，同样先选中项目文件夹图标，这时在 Project 菜单中也有一样的菜单可选。打开项目选项窗口，转到 Output 选项页图 3.13 所示，图中 1 是选择编译输出的路径，2 是设置编译输出生成的文件名，3 则是决定是否要创建 HEX 文件，选中它就可以输出 HEX 文件到指定的路径中。选好了？好，我们再将它重新编译一次，很快在编译信息窗口中就显示 HEX 文件创建到指定的路径中了，如图 3.14。这样我们就可用自己的编程器所附带的软件去读取并烧到芯片了，再用实验板看结果，至于编程器或仿真器品种繁多具体方法就看它的说明书了，这里也不做讨论。

（技巧：一、在图 3.12 中的 1 里的项目文件树形目录中，先选中对象，再单击它就可对它进行重命名操作，双击文件图标便可打开文件。二、在 Project 下拉菜单的最下方有最近编辑过的项目路径保存，这里可以快速打开最近在编辑的项目。）

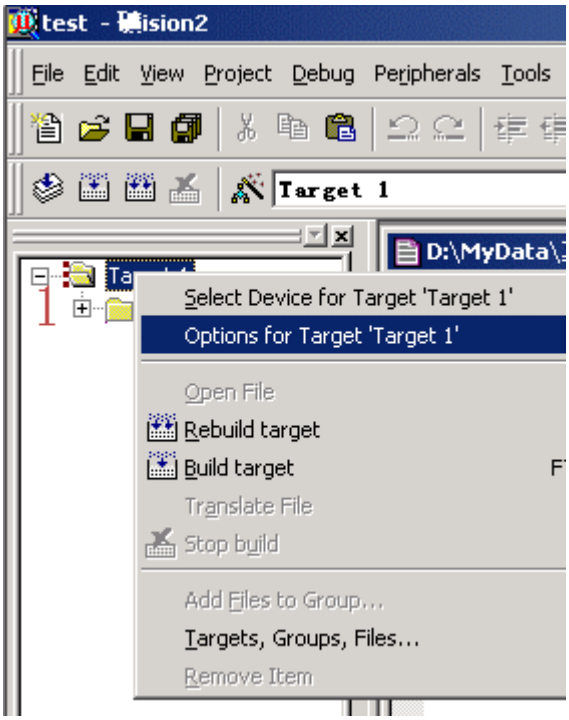


图 3.12 项目功能菜单

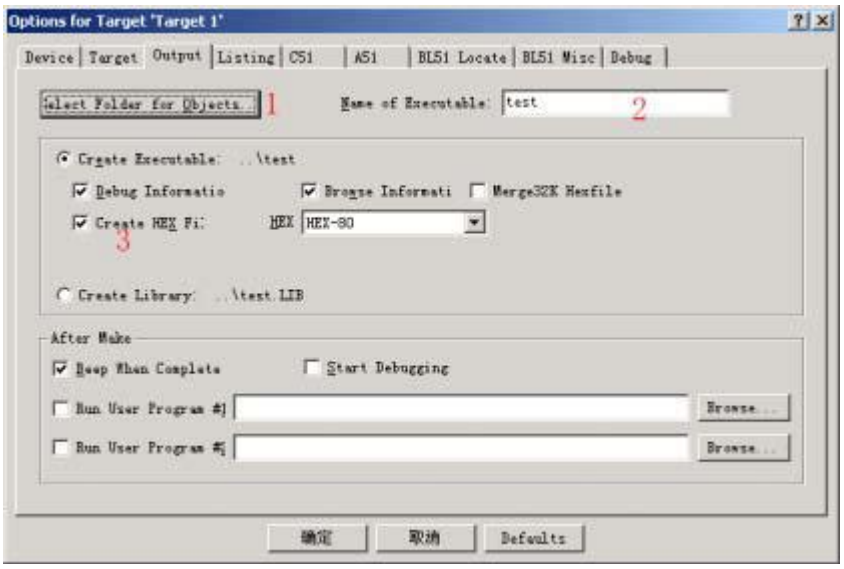


图 3.13 项目选项窗口

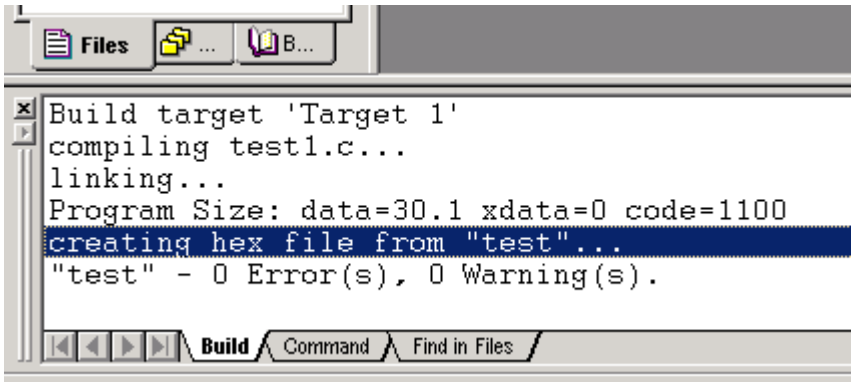


图 3.14 编译信息窗口

或许您已把编译好的文件烧到了芯片上，如果您购买或自制了带串口输出元件的学习实验板，那您就可以把串口和 PC 机串口相联用串口调试软件或 Windows 的超级终端,将其波特率设为 1200，就可以看到不停输出的“Hello World!”字样。也许您还没有实验板，那这里先说说 AT89C51 的最小化系统，再以一实例程序验证最小化系统是否在运行，这个最小化系统也易于自制用于实验。图 3.15 便是 AT89C51 的最小化系统,不过为了让我们可以看出它是在运行的，我加了一个电阻和一个 LED，用以显示它的状态，晶振可以根据自己的情况使用,一般实验板上是用 11.0592MHz 或 12MHz,使用前者的好外是可以产生标准的串口波特率，后者则一个机器周期为 1 微秒，便于做精确定时。在自己做实验里，注意的是 VCC 是+5V 的，不能高于此值，否则将损坏单片机，太低则不能正常工作。在 31 脚要接高电平，这样我们才能执行片内的程序，如接低电平则使用片外的程序存储器。下面，我们建一个新的项目名为 OneLED 来验证最小化系统是否可以工作。程序如下：

```
#include <AT89X51.h> //预处理命令
void main(void) //主函数名
{
//这是第一种注释方式
unsigned int a; //定义变量 a 为 int 类型
/*
```

```
这是第二种注释方式
*/
do{                          //do while 组成循环
for (a=0; a<50000; a++);     //这是一个循环
P1_0 = 0;                    //设 P1.0 口为低电平，点亮 LED
for (a=0; a<50000; a++);     //这是一个循环
P1_0 = 1;                    //设 P1.0 口为高电平，熄灭 LED
}
while(1);
}
```

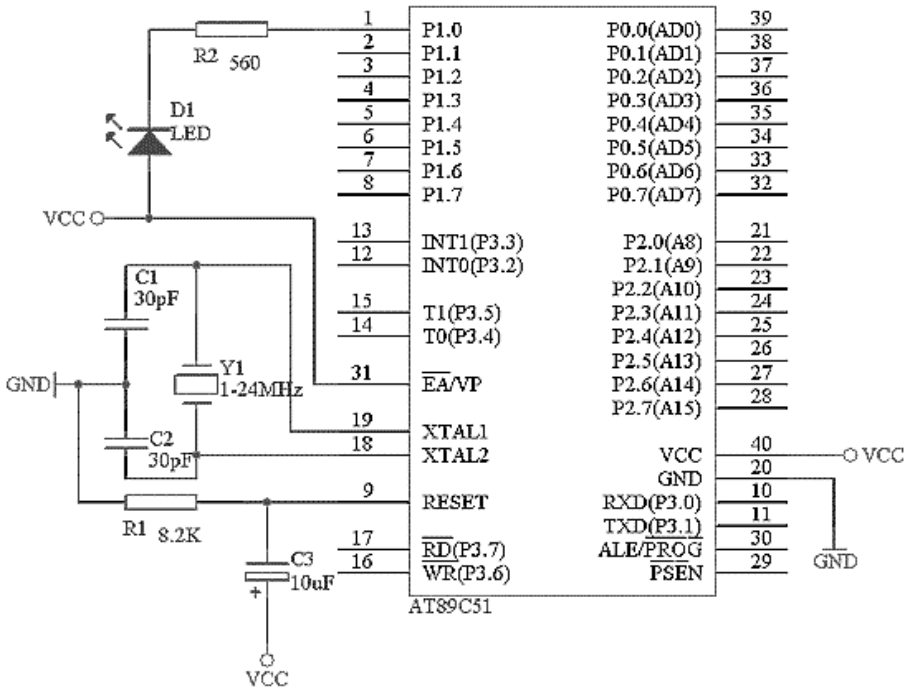


图 3.15　AT89C51 最小化系统

在 KEIL C 编译器所支持的注释语句。一种是以“//”符号开始的语句，符号之后的语句都被视为注释，直到有回车换行。另一种是在“/*”和“*/”符号之内的为注释。注释不会被 C 编译器所编译。一个 C 应用程序中应有一个 main 主函数，main 函数可以调用别的功能函数，但其它功能函数不允许调用 main 函数。不论 main 函数放在程序中的那个位置，总是先被执行。用上面学到的知识编译写好的 OneLED 程序，并把它烧到刚做好的最小化系统中。上电，刚开始时 LED 是不亮的（因为上电复位后所有的 IO 口都置 1 引脚为高电平），然后延时一段时间（for (a=0; a<50000; a++)这句在运行），LED 亮，再延时，LED 熄灭，然后交替亮、灭。第一个真正的小应用就做完，呵呵，先不要管它是否实用哦。如果没有这样的效果那么您就要认真检查一下电路或编译烧写的步骤了。

3.4 数据类型

简单来说是 C 语言的标识符和关键字。标识符是用来标识源程序中某个对象的名字的，这些对象可以是语句、数据类型、函数、变量、数组等等。C 语言是大小字敏感的一种高级语言，如果我们要定义一个定时器 1，可以写做“Timer1”，如果程序中有“TIMER1”，那么这两个是完全不同定义的标识符。标识符由字符串，数字和下划线等组成，注意的是第一个

字符必须是字母或下划线，如“1Timer”是错误的，编译时便会有错误提示。有些编译系统专用的标识符是以下划线开头，所以一般不要以下划线开头命名标识符。标识符在命名时应当简单，含义清晰，这样有助于阅读理解程序。在 C51 编译器中，只支持标识符的前 32 位为有效标识，一般情况下也足够用了，除非你要写天书：P。

关键字则是编程语言保留的特殊标识符，它们具有固定名称和含义，在程序编写中不允许标识符与关键字相同。在 KEIL uVision2 中的关键字除了有 ANSI C 标准的 32 个关键字外还根据 51 单片机的特点扩展了相关的关键字。其实在 KEIL uVision2 的文本编辑器中编写 C 程序，系统可以把保留字以不同颜色显示，缺省颜色为天蓝色。先看表 3.3，表中列出了 KEIL uVision2 C51 编译器所支持的数据类型。在标准 C 语言中基本的数据类型为 char,int,short,long,float 和 double，而在 C51 编译器中 int 和 short 相同，float 和 double 相同，这里就不列出说明了。下面来看看它们的具体定义：

表 3.3 KEIL uVision2 C51 编译器所支持的数据类型

数据类型	长　度	值　域
unsigned char	单字节	0～255
signed char	单字节	-128～+127
unsigned int	双字节	0～65535
signed int	双字节	-32768～+32767
unsigned long	四字节	0～4294967295
signed long	四字节	-2147483648～+2147483647
float	四字节	±1.175494E-38～±3.402823E+38
*	1～3 字节	对象的地址
bit	位	0 或 1
sfr	单字节	0～255
sfr16	双字节	0～65535
sbit	位	0 或 1

1.　char 字符类型

char 类型的长度是一个字节，通常用于定义处理字符数据的变量或常量。分无符号字符类型 unsigned char 和有符号字符类型 signed char，默认值为 signed char 类型。unsigned char 类型用字节中所有的位来表示数值，所可以表达的数值范围是 0～255。signed char 类型用字节中最高位字节表示数据的符号，“0”表示正数，“1”表示负数，负数用补码表示。所能表示的数值范围是-128～+127。unsigned char 常用于处理 ASCII 字符或用于处理小于或等于 255 的整型数。

＊正数的补码与原码相同，负二进制数的补码等于它的绝对值按位取反后加 1。

2.　int 整型

int 整型长度为两个字节,用于存放一个双字节数据。分有符号 int 整型数 signed int 和无符号整型数 unsigned int，默认值为 signed int 类型。signed int 表示的数值范围是-32768～+32767，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。unsigned int 表示的数值范围是 0～65535。

我们来写个小程序看看 unsigned char 和 unsigned int 用于延时的不同效果，说明它们的长度是不同的，呵，尽管它并没有实际的应用意义，这里我们学习它们的用法就行。依旧用我们上一课的最小化系统做实验，不过要加多一个电阻和 LED，如图 4－1。实验中用 D1 的点亮表明正在用 unsigned int 数值延时，用 D2 点亮表明正在用 unsigned char 数值延时。

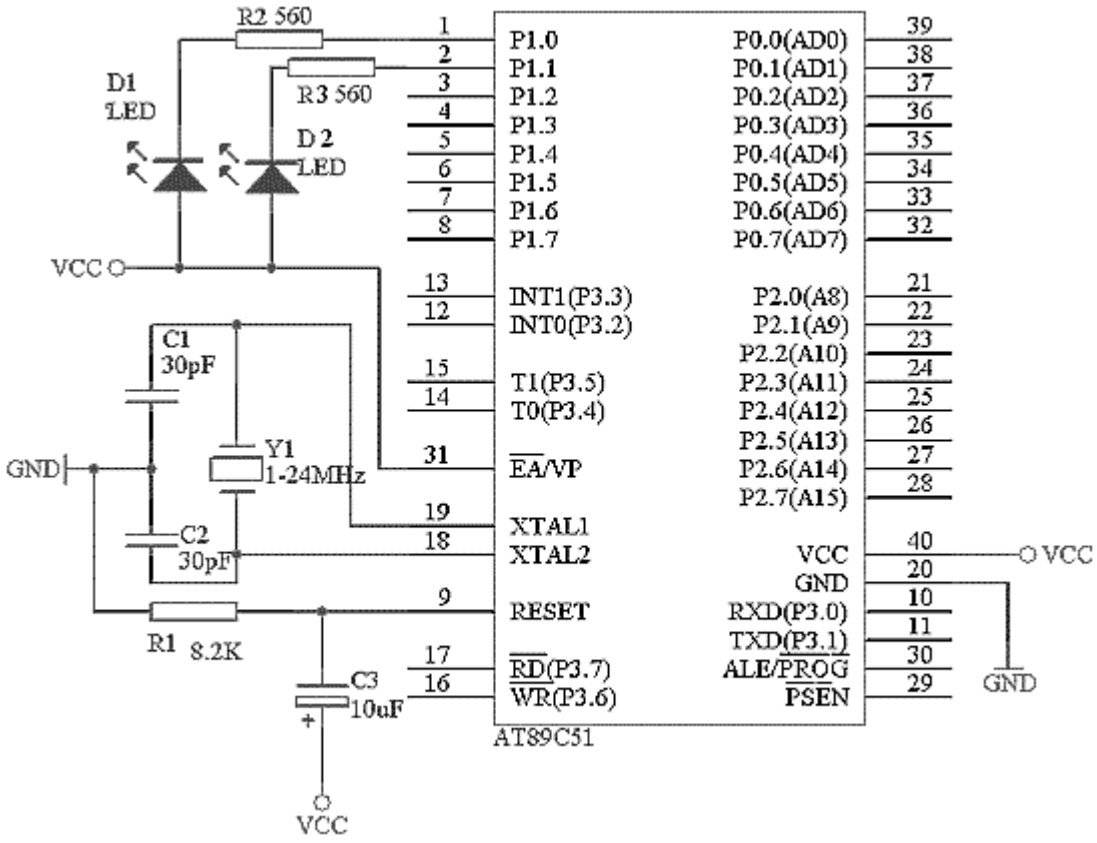


图 3.16　实验用电路

我们把这个项目称为 TwoLED,实验程序如下：

```
#include <AT89X51.h> //预处理命令
void main(void) //主函数名
{
unsigned int a; //定义变量 a 为 unsigned int 类型
unsigned char b; //定义变量 b 为 unsigned char 类型
do{                                                    //do while 组成循环
for (a=0; a<65535; a++)
P1_0 = 0; //65535 次设 P1.0 口为低电平，点亮 LED
P1_0 = 1; //设 P1.0 口为高电平，熄灭 LED
for (a=0; a<30000; a++); //空循环
for (b=0; b<255; b++)
P1_1 = 0; //255 次设 P1.1 口为低电平，点亮 LED
P1_1 = 1; //设 P1.1 口为高电平，熄灭 LED
for (a=0; a<30000; a++); //空循环
}
while(1);
}
```

同样编译烧写，上电运行您就可以看到结果了。很明显 D1 点亮的时间长于 D2 点亮的时间。程序中的循环延时时间并不是很好确定，并不太适合要求精确延时的场合，关于这方面我们以后也会做讨论。这里必须要讲的是，当定义一个变量为特定的数据类型时，在程序使用该变量不应使它的值超过数据类型的值域。如本例中的变量 b 不能赋超出 0～255 的

值，如 for(b=0; b<255; b++)改为 for (b=0; b<256; b++),编译是可以通过的，但运行时就会有问题出现，就是说 b 的值永远都是小于 256 的，所以无法跳出循环执行下一句 P1_1 = 1，从而造成死循环。同理 a 的值不应超出 0～65535。大家可以烧片看看实验的运行结果，同样软件仿真也是可以看到结果的。

3. long 长整型

long 长整型长度为四个字节,用于存放一个四字节数据。分有符号 long 长整型 signed long 和无符号长整型 unsigned long，默认值为 signed long 类型。signed int 表示的数值范围是-2147483648～+2147483647，字节中最高位表示数据的符号，“0”表示正数，“1”表示负数。unsigned long 表示的数值范围是 0～4294967295。

4. float 浮点型

float 浮点型在十进制中具有 7 位有效数字，是符合 IEEE－754 标准的单精度浮点型数据，占用四个字节。因浮点数的结构较复杂在以后的章节中再做详细的讨论。

5. * 指针型

指针型本身就是一个变量，在这个变量中存放的指向另一个数据的地址。这个指针变量要占据一定的内存单元，对不同的处理器长度也不尽相同，在 C51 中它的长度一般为 1～3 个字节。指针变量也具有类型，在以后的课程中有专门一课做探讨，这里就不多说了。

6. bit 位标量

bit 位标量是 C51 编译器的一种扩充数据类型，利用它可定义一个位标量，但不能定义位指针，也不能定义位数组。它的值是一个二进制位，不是 0 就是 1，类似一些高级语言中的 Boolean 类型中的 True 和 False。

7. sfr 特殊功能寄存器

sfr 也是一种扩充数据类型，点用一个内存单元，值域为 0～255。利用它可以访问 51 单片机内部的所有特殊功能寄存器。如用 sfr P1 = 0x90 这一句定 P1 为 P1 端口在片内的寄存器，在后面的语句中我们用以用 P1 = 255（对 P1 端口的所有引脚置高电平）之类的语句来操作特殊功能寄存器。

8. sfr16 16 位特殊功能寄存器

sfr16 占用两个内存单元，值域为 0～65535。sfr16 和 sfr 一样用于操作特殊功能寄存器，所不同的是它用于操作占两个字节的寄存器，好定时器 T0 和 T1。

9. sbit 可录址位

sbit 同位是 C51 中的一种扩充数据类型，利用它可以访问芯片内部的 RAM 中的可寻址位或特殊功能寄存器中的可寻址位。如先前我们定义了 sfr P1 = 0x90; //因 P1 端口的寄存器是可位寻址的，所以我们可以定义 sbit P1_1 = P1^1; //P1_1 为 P1 中的 P1.1 引脚//同样我们可以用 P1.1 的地址去写,如 sbit P1_1 = 0x91;

这样我们在以后的程序语句中就可以用 P1_1 来对 P1.1 引脚进行读写操作了。通常这些可以直接使用系统提供的预处理文件，里面已定义好各特殊功能寄存器的简单名字，直接引用可以省去一点时间，我自己是一直用的。当然您也可以自己写自己的定义文件，用您认为好记的名字。

关于数据类型转换等相关操作在后面的课程或程序实例中将有所提及。大家可以用所讲到的数据类型改写一下这课的实例程序，加深对各类型的认识。

3.5 常量

上一节我们学习了 KEIL C51 编译器所支持的数据类型。而这些数据类型又是怎么用在常量和变量的定义中的呢？又有什么要注意的吗？下面就来看看吧。晕！你还区分不清楚什么是常量，什么是变量。常量是在程序运行过程中不能改变值的量，而变量是可以在程序运行过程中不断变化的量。变量的定义可以使用所有 C51 编译器支持的数据类型，而常量的

数据类型只有整型、浮点型、字符型、字符串型和位标量。这一节我们学习常量定义和用法，而下一节则学习变量。

常量的数据类型说明是这样的

1． 整型常量可以表示为十进制如 123,0，－89 等。十六进制则以 0x 开头如 0x34,-0x3B 等。长整型就在数字后面加字母 L，如 104L，034L，0xF340 等。

2． 浮点型常量可分为十进制和指数表示形式。十进制由数字和小数点组成，如 0.888,3345.345,0.0 等，整数或小数部分为 0，可以省略但必须有小数点。指数表示形式为[±]数字[.数字]e[±]数字，[]中的内容为可选项，其中内容根据具体情况可有可无，但其余部分必须有,如 125e3,7e9,-3.0e-3。

3． 字符型常量是单引号内的字符，如‘a’，‘d’等，不可以显示的控制字符，可以在该字符前面加一个反斜杠“\”组成专用转义字符。常用转义字符表请看表 5－1。

4． 字符串型常量由双引号内的字符组成，如“test”,“OK”等。当引号内的没有字符时，为空字符串。在使用特殊字符时同样要使用转义字符如双引号。在 C 中字符串常量是做为字符类型数组来处理的，在存储字符串时系统会在字符串尾部加上\o 转义字符以作为该字符串的结束符。字符串常量“A”和字符常量‘A’是不同的，前者在存储时多占用一个字节的字间。

5． 位标量，它的值是一个二进制。

表 3.4 常用转义字符表

转义字符	含义	ASCII 码（16/10 进制）
\o	空字符(NULL)	00H/0
\n	换行符(LF)	0AH/10
\r	回车符(CR)	0DH/13
\t	水平制表符(HT)	09H/9
\b	退格符(BS)	08H/8
\f	换页符(FF)	0CH/12
\’	单引号	27H/39
\″	双引号	22H/34
\\	反斜杠	5CH/92

常量可用在不必改变值的场合，如固定的数据表，字库等。常量的定义方式有几种,下面来加以说明。

#difine False 0x0; //用预定义语句可以定义常量#difine True 0x1; //这里定义 False 为 0,True 为 1//在程序中用到 False 编译时自动用 0 替换，同理 True 替换为 1unsigned int code a=100; //这一句用 code 把 a 定义在程序存储器中并赋值 const unsigned int c=100; //用 const 定义 c 为无符号 int 常量并赋值以上两句它们的值都保存在程序存储器中，而程序存储器在运行中是不允许被修改的，所以如果在这两句后面用了类似 a=110，a++这样的赋值语句，编译时将会出错。说了一通还不如写个程序来实验一下吧。写什么程序呢？跑马灯！对，就写这个简单易懂的吧，这个也好说明典型的常量用法。先来看看电路图吧。它是在我们上一课的实验电路的基础上增加 6 个 LED 组成的，也就是用 P1 口的全部引脚分别驱动一个 LED，电路如图 3.16 所示。

新建一个 RunLED 的项目，主程序如下：

```
#include <AT89X51.H> //预处理文件里面定义了特殊寄存器的名称如 P1 口定义为 P1
void main(void)
{
```

```
//定义花样数据
Const unsigned char
design[32]={0xFF,0xFE,0xFD,0xFB,0xF7,0xEF,0xDF,0xBF,0x7F,0x7F,0xBF,0xDF,0x
EF,0xF7,0xFB,0xFD,0xFE,0xFF,0xFF,0xFE,0xFC,0xF8,0xF0,0xE0,0xC0,0x80,0x0,
0xE7,0xDB,0xBD,0x7E,0xFF};
unsigned int a; //定义循环用的变量
unsigned char b; //在 C51 编程中因内存有限尽可能注意变量类型的使用
//尽可能使用少字节的类型，在大型的程序中很
受用
do{
for (b=0; b<32; b++)
{
for(a=0; a<30000; a++); //延时一段时间
P1 = design[b]; //读已定义的花样数据并写花样数据到 P1 口
}
}while(1);
}
```

程序中的花样数据可以自以去定义,因这里我们的 LED 要 AT89C51 的 P1 引脚为低电平才会点亮，所以我们要向 P1 口的各引脚写数据 0 对应连接的 LED 才会被点亮，P1 口的八个引脚刚好对应 P1 口特殊寄存器的八个二进位，如向 P1 口定数据 0xFE,转成二进制就是 11111110,最低位 D0 为 0 这里 P1.0 引脚输出低电平，LED1 被点亮。如此类推，大家不难算出自己想要做的效果了。大家编译烧写看看，效果就出来，显示的速度您可以根据需要调整延时 a 的值，不要超过变量类型的值域就很行了。哦，您还没有实验板？那如何可以知道程序运行的结果呢？呵,不用急,这就来说说用 KEIL uVision2 的软件仿真来调试 IO 口输出输入程序。

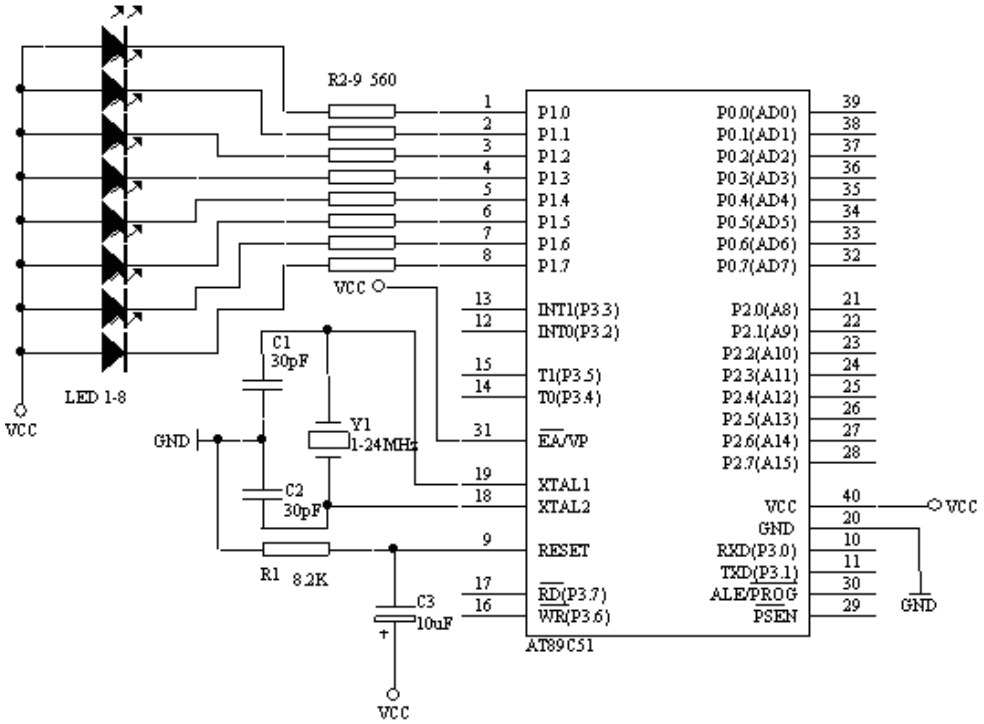


图 3.16 八路跑马灯电路

编译运行上面的程序，然后按外部设备菜单 Peripherals－I/O Ports－Port1 就打开 Port1 的调试窗口了，如图 5－3 中的 2。这时程序运行了，但我们并不能在 Port1 调试窗口上看到有会什么效果，这时我们可以用鼠标左击图 3.17 中 1 旁边绿色的方条，点一下就有一个小红方格在点一下又没有了，哪一句语句前有小方格程序运行到那一句时就停止了，就是设置调试断点，同样图 3.17 中的 1 也是同样功能，分别是增加/移除断点、移除所有断点、允许/禁止断点、禁止所有断点，菜单也有一样的功能，另外菜单中还有 Breakpoints 可打开断点设置窗口它的功能更强大，不过我们这里先不用它。我们在“P1 = design[b];”这一句设置一个断点这时程序运行到这里就停住了，再留意一下 Port1 调试窗口，再按图 5-2 中的 2 的运行键，程序又运行到设置断点的地方停住了，这时 Port1 调试窗口的状态又不同了。也就是说 Port1 调试窗口模拟了 P1 口的电平状态，打勾为高电平，不打勾则为低电平，窗口中 P1 为 P1 寄存器的状态，Pins 为引脚的状态，注意的是如果是读引脚值必须把引脚对应的寄存器置 1 才能正确读取。图 3.18 中 2 旁边的｛｝样的按钮分别为单步入，步越，步出和执行到当前行。图中 3 为显示下一句将要执行的语句。图 3.18 中的 3 是 Watches 窗口可查看各变量的当前值，数组和字串是显示其头一个地址，如本例中的 design 数组是保存在 code 存储区的首地址为 D:0x08,可以在图中 4 Memory 存储器查看窗口中的 Address 地址中打入 D:0x08 就可以查看到 design 各数据和存放地址了。如果你的 uVision2 没有显示这些窗口，可以在 View 菜单中打开在图 5－2 中 3 后面一栏的查看窗口快捷栏中打开。

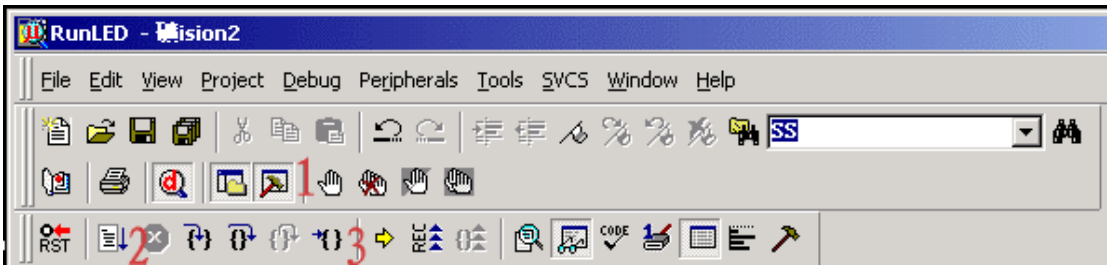


图 3.17 调试用快捷菜单栏

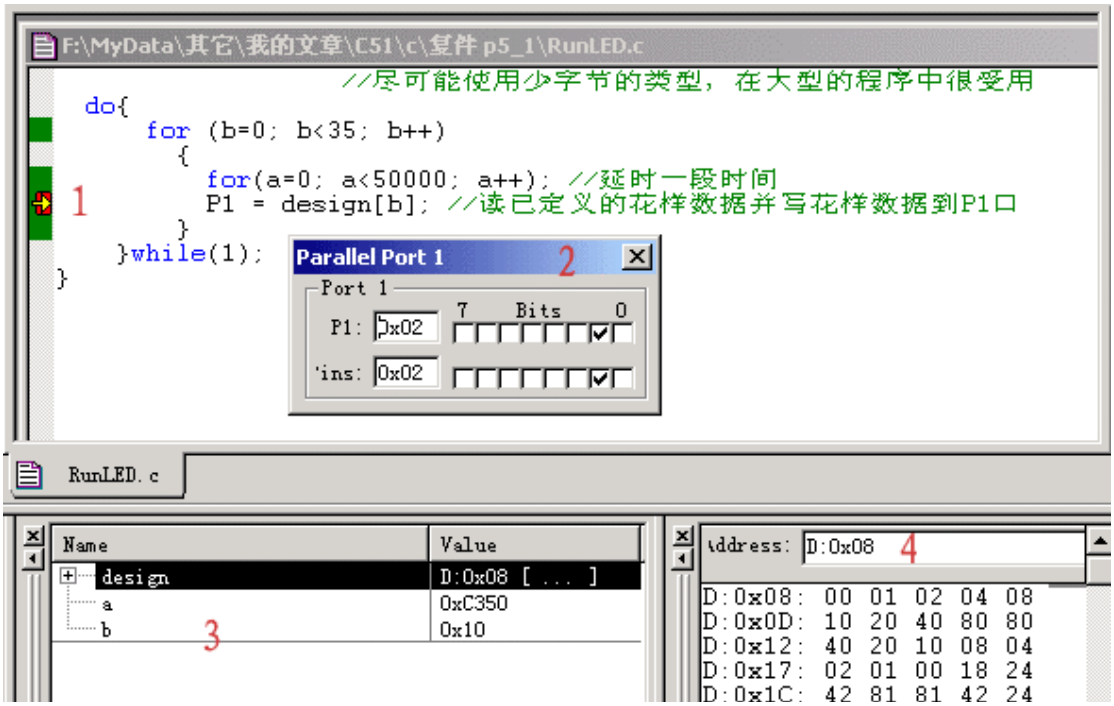


图 3.18 各调试窗口

4. 单片机的 C51 编程技巧

4.1 C51 与 C 语言

1、Main()

```
{
    //通信初始化
   SCON=0x40; // 串行口工作方式 1 ，TI=0 ，RI=0
   TMOD=0x20; //定时器 1 工作方式 2，8 位自动重装式，晶振频率 11.0592MHz
   PCON=0;
   TH1=0xFD;   //9600bps
   TL1=0xFD;
   TR1=1;
   TI=1;
   printf("%f,%f\n",ppf,mf);//通信
}
```

2、C51 的数据类型

bit	0 1	
char	8	-128~127
unsigned char	8	0~256
int	16	-32768~32767
unsigned	16	0~65535
long int	32	-231 ~(231-1)
unsigned	32	0~（232 -1)
float	32	//在汇编中有 24 位

例：int a=721　　BCD

int a=0721　　8

int a=0xFF　　Hex

3、C51 的函数

```
int j 全局变量
void  fun(  )
{
int i;
}
```

4.2 C 语言中常用的语句（也适合于 C51）

①if (条件)

```
{
}
```

② if( )

```
    else if( )
   else
```

③switch(条件式)

```
    { case: 值 1
    break;
```

```
    case: 值 2
    break;
    ……..
    }
④while(条件)
    {   动作
    先条件
    后内容
    }
⑤ do
    {先内容
    后条件
   }while(条件)

⑥for（算式 1; 算式 2; 算式 3）
    // （赋植; 条件; 增量）
{

}
例 1:    for(i=0;i<10;i++)
    {
        动作
    }
例 2：for( ; ; );//C 语言认为合法
例 3：i=0;
 for(;i<10;i++)
{
}
```

4.3 指针与数组

```
char  *a;
char   ab;
a=&ab;
char 名[ ][ ][ ]（三维）
void dalay(unsigned int  count )
 {
    unsigned int I;
    for(i=0;i<count;i++)
    {
        }
}
main( )
{
    dalay(100);
}
```

4.4 语言中按地址传递

```
void demo(char buf[10])
{
    char i;
     for(1=0;i<10;i++)
   {
       buf[i]=0;
    }
}
main( )
{
    char dat[10];
    demo(dat);
}
```

4.5 前置处理器

#define 宏定义 C 语言优先处理

#define 宏包含

条件编辑

#define 宏名 字符串

例：#define PI 3.1415926

#define CS P1.0

#HIBYTE（a） ((unsigned char ((a))>>8))

unsigned char b;

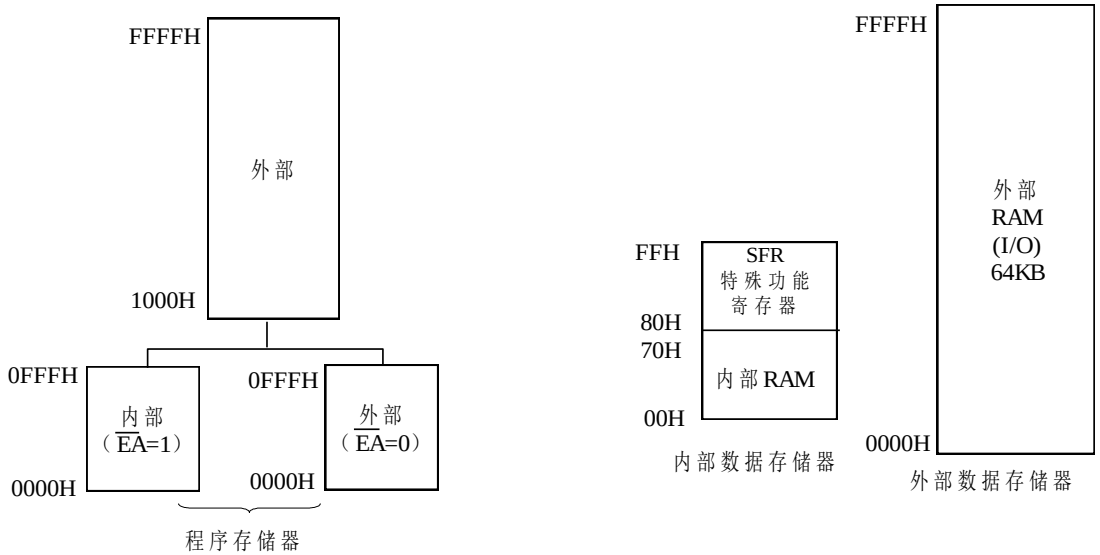
例：b=HIBYTE（0xFF18） 结果：b=0xFF18

宏定义可以定义函数

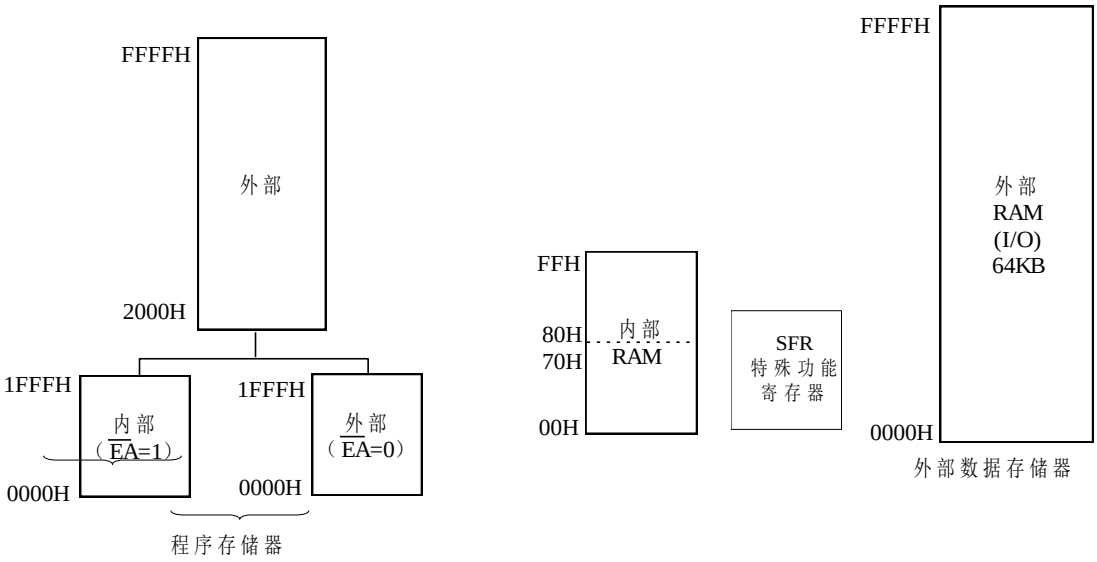
#define F(x) (x)*(x)+2x*(x)+1

4.6 C51 的存储器类型

1、复习



(a) 51系列存储器配置



（b）52系列存储器配置

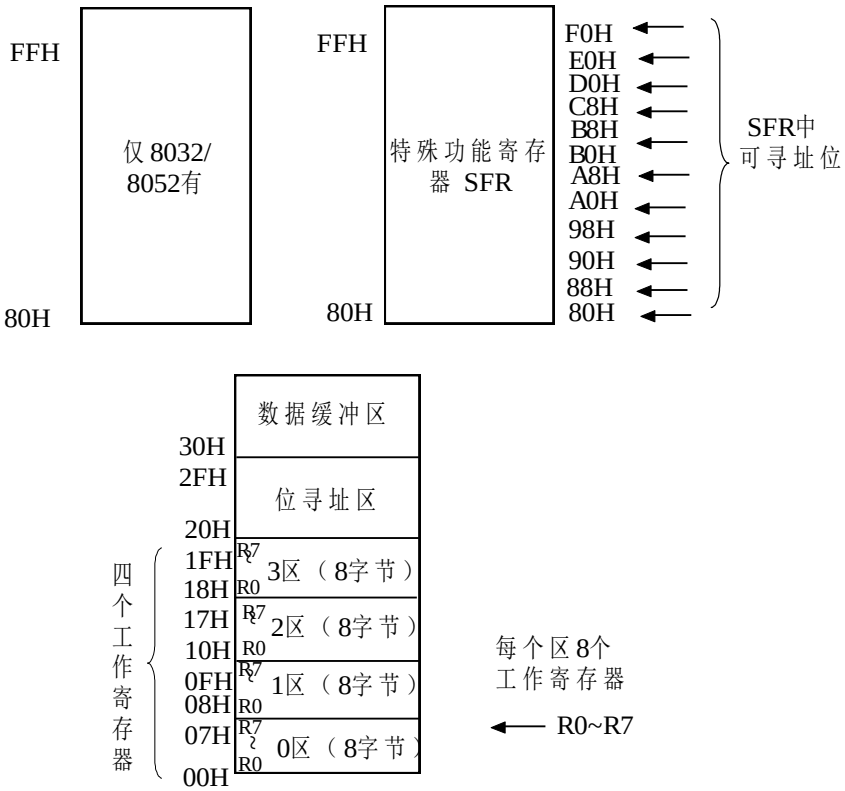


图 4.1　存储体结构

2、code 说的变量在程序空间

data　内　0~7F

idata 内　@R0 ~R　0~FF　256

bdata　可位寻址　　16bit

pdata　R0 R1 间接寻址 0~0xFF

xdata　外 64K 全部空间　0~0xFFFF

例 1：unsigned　char　code　lable[ ]={1,2,3,….,100};

```
unsigned  char  data  i;
unsigned  char  bdata status;
sbit  st_7 =status^7;
sbit  st_6=status^6;
sbit  st_0=status^0;
status=0;
st_7=1;
status=0x80;
```

3、sfr　psw=0xD0;

```
sfr  P0=0x80;
sbit  P0_1=P0^1;
unsigned char address  _at_0x20;
int i;
```

4、C51 中指令的控制存储模式选择

small　0x00~0x7FH

compact　int　pdata　i;

large　int　cdata

4.7　中断函数在 C51 中实现

INT0	0	0x0003
Time/Counter0	1	0x000B
INT1	2	0x0013
Time/Counter1	3	0x001B
Serial port	4	0x0023

Void　函数名(void)interrupt　中断号　缓存器库

例：

```
void TIMER0(void)  interrupt 1 using 2
 {
   }
main( )
{
 ET0=1;
 EA=1;
}
```

4.8　几个编程实例

例 1：串行接口发送程序

```
#include  “AT89x51.h”
main()
  {
   int i;
SCON=0x52;          // 串行口工作方式 1 ， TI=1 ， RI=0
```

```
TMOD=0x21;      //定时器 1 工作方式 2，8 位自动重装式，晶振频率 11.0592MHz
TH1=0xFD;//9600bps
TL1=0xFD;
TR1=1;
While(1)
 {
  while(TI==0);等待
TI=0;
SBUF=0x50;
for(i=0;i<50;i++);
}
}
```

例 2：单片机的 P1 口产生一个方波

```
  #include “AT89x51.h”
  void TIMERTSR(void) interrupt 1 using 2
  {
   TH0=0xFC;
   TL0=0x67;
   P1=~P1;
}
main()
{
 TMOD=0x11;     //定时器方式 1,11.0592MHz, 定时 1ms
TH0=0xFC;
 TL0=ox67;
TR0=1;
ET0=1;
EA=1;
for(;;);
}
```

例 3：并行接口芯片的 C51 编程

```
#include <absacc.h> // 函数原形：#define    CBYTE((unsigned char *)0x50000L)
//#define   DBYTE((unsigned char *)0x40000L)
//#define   PBYTE((unsigned char *)0x30000L)
//#define   XBYTE((unsigned char *)0x20000L)
```

//CBYTE 寻址 CODE 区,DBYTE 寻址 DATA 区, PBYTE 寻址分页的 XDATA 区(采用 MOVX @R0 指令），XBYTE 寻址 XDATA 区（采用 MOVX @DPTR）指令.

```
#include <reg51.h>
#define uchar unsigned char
#define uint unsigned int
#define  CTRL_8254 XBYTE[0X7F00]
#define  T0_8254 XBYTE[0X7C00]
#define  T1_8254 XBYTE[0X7D00]
 #define  T2_8254 XBYTE[0X7E00]
```

```
.......
int idata counter0,counter1,counter3;

/*------------1/60 秒定时 1---------*/
 void time1(void) interrupt 3 using 2
{
        TF1=0;
        ctimer++;
         TL1=0x00;      //11.0592MHz,1/60s
         TH1=0x0c4;
}
/*----------------------------*/
main()
{
uint i,j;
   CTRL_8254=0x30;
   CTRL_8254=0x70;
   CTRL_8254=0xb0;
  ..................
    T0_8254=0;
    T0_8254=0;
    T1_8254=0;
    T1_8254=0;
    T2_8254=0;
    T2_8254=0
.............
    while(1)
{
        while(ctimer<60){
                        }   /*  不等于 1 秒等待 */
        GATE=0;
        i=T0_8254;
        j=T0_8254;
        T0_8254=0;
        T0_8254=0;
        counter1=j*256+i;
        counter1=~counter1+2;
        i=T1_8254;
        j=T1_8254;
        T1_8254=0;
        T1_8254=0;
        counter3=j*256+i;
        counter3=~counter3+2;
        i=T2_8254;
```

```
        j=T2_8254;
        T2_8254=0;
        T2_8254=0;
        counter0=j*256+i;
        counter0=~counter0+2;
}
}
```

例 4：串行接口芯片的 C51 编程

//　　7219LED 显示程序

1、MAX7219 管脚说明

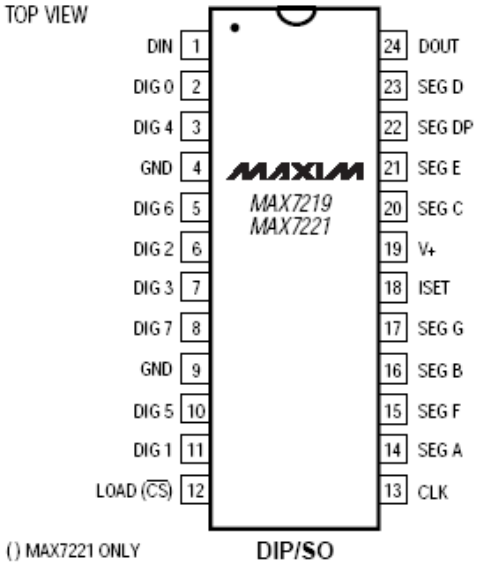
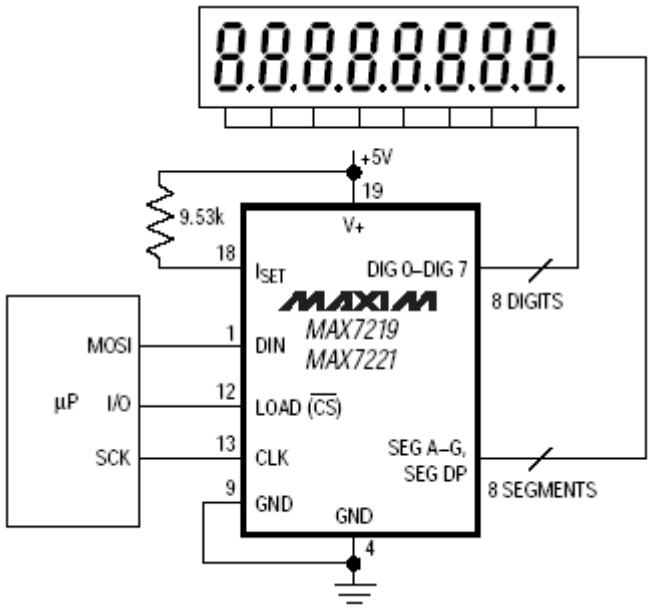


表 1　MAX7219管脚说明

引　脚	符　号	功 能 说 明
19	V+	电源端（4~ 6V,典型值 5V±10%）
4, 9	GND	地
18	I_{SET}	经电阻 R_{SET}与 V+相连,以设定段电流峰值．
1	DIN	串行数据输入端(在 CLK上升沿,数据装入内部 16bit移位寄存器）
13	CLK	串行时钟输入端．在 CLK上升沿,数据移入内部移位寄存器;在 CLK下降沿,数据移出 DOUT端．
24	DOUT	串行数据输出端.移入 DIN端的串行数据在 16.5个时钟周期后,在 DOUT端有效．
2, 3, 5~ 8, 10, 11	DIG 0~ 7	八位位驱动线,从显示器吸收电流．
14~ 17, 20~ 23	SEG A-G, DP	七段和小数点驱动线,为显示器提供段电流．

2、MAX7219 与 8XX51 的连接方法



3、MAX7219 时序图

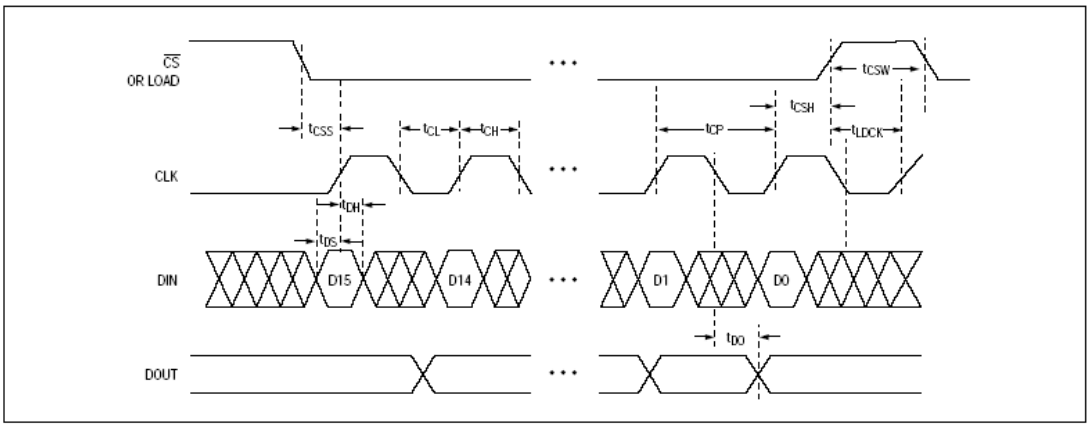


Figure 1. Timing Diagram

4、MAX7219 的串行数据格式

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

D15-D12 ： 任意
D11-D8： 寄存器地址或显示单元地址
D7-D0： 命令或显示的数据

5、寄存器的设置与初始化

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	X0
Digit 0	X	0	0	0	1	X1
Digit 1	X	0	0	1	0	X2
Digit 2	X	0	0	1	1	X3
Digit 3	X	0	1	0	0	X4
Digit 4	X	0	1	0	1	X5
Digit 5	X	0	1	1	0	X6
Digit 6	X	0	1	1	1	X7
Digit 7	X	1	0	0	0	X8
Decode Mode	X	1	0	0	1	X9
Intensity	X	1	0	1	0	XA
Scan Limit	X	1	0	1	1	XB
Shutdown	X	1	1	0	0	XC
Display Test	X	1	1	1	1	XF

Digit0- Digit7 ：1-8LED 显示器件，16 进制代码为：X1-X8
Decode Mode ：译码模式寄存器，16 进制代码为：X9
Intensity： 亮度控制寄存器，16 进制代码为：XA
Scan Limit：扫描位数寄存器，16 进制代码为：XB
Shutdown：显示器关闭控制寄存器，16 进制代码为：XC
Display Test：显示器测试寄存器，16 进制代码为：XF

Table 3. Shutdown Register Format (Address (Hex) = XC)

MODE	ADDRESS CODE (HEX)	REGISTER DATA							
		D7	D6	D5	D4	D3	D2	D1	D0
Shutdown Mode	XC	X	X	X	X	X	X	X	0
Normal Operation	XC	X	X	X	X	X	X	X	1

max7219（0x0C，0x01）；

Table 4. Decode-Mode Register Examples (Address (Hex) = X9)

DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	FF

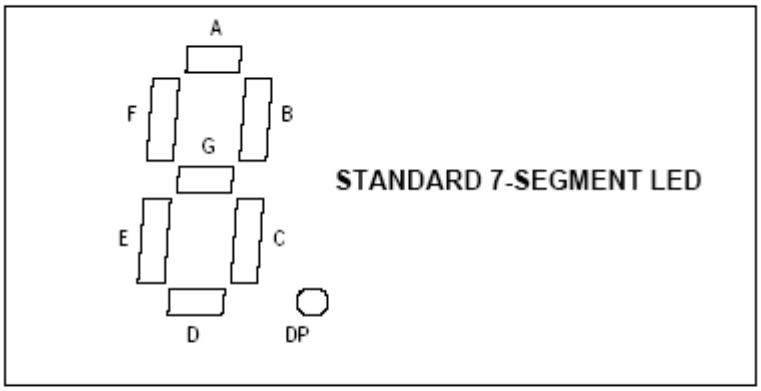
max7219（0x09，0xFF);

Table 5. Code B Font

7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

*The decimal point is set by bit D7 = 1

Table 6. No-Decode Mode Data Bits and Corresponding Segment Lines



	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Corresponding Segment Line	DP	A	B	C	D	E	F	G

Table 7. Intensity Register Format (Address (Hex) = XA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	X0
3/32	2/16	X	X	X	X	0	0	0	1	X1
5/32	3/16	X	X	X	X	0	0	1	0	X2
7/32	4/16	X	X	X	X	0	0	1	1	X3
9/32	5/16	X	X	X	X	0	1	0	0	X4
11/32	6/16	X	X	X	X	0	1	0	1	X5
13/32	7/16	X	X	X	X	0	1	1	0	X6
15/32	8/16	X	X	X	X	0	1	1	1	X7
17/32	9/16	X	X	X	X	1	0	0	0	X8
19/32	10/16	X	X	X	X	1	0	0	1	X9
21/32	11/16	X	X	X	X	1	0	1	0	XA
23/32	12/16	X	X	X	X	1	0	1	1	XB
25/32	13/16	X	X	X	X	1	1	0	0	XC
27/32	14/16	X	X	X	X	1	1	0	1	XD
29/32	15/16	X	X	X	X	1	1	1	0	XE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	XF

max7219（0x0A，0x0B);

Table 8. Scan-Limit Register Format (Address (Hex) = XB)

SCAN LIMIT	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
Display digit 0 only*	X	X	X	X	X	0	0	0	X0
Display digits 0 & 1*	X	X	X	X	X	0	0	1	X1
Display digits 0 1 2*	X	X	X	X	X	0	1	0	X2
Display digits 0 1 2 3	X	X	X	X	X	0	1	1	X3
Display digits 0 1 2 3 4	X	X	X	X	X	1	0	0	X4
Display digits 0 1 2 3 4 5	X	X	X	X	X	1	0	1	X5
Display digits 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	X6
Display digits 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	X7

*See *Scan-Limit Register* section for application.

max7219（0x0B，0x07);

Table 9. Maximum Segment Current for 1-, 2-, or 3-Digit Displays

NUMBER OF DIGITS DISPLAYED	MAXIMUM SEGMENT CURRENT (mA)
1	10
2	20
3	30

Table 10. Display-Test Register Format (Address (Hex) = XF)

MODE	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Normal Operation	X	X	X	X	X	X	X	0
Display Test Mode	X	X	X	X	X	X	X	1

Note: The MAX7219/MAX7221 remain in display-test mode (all LEDs on) until the display-test register is reconfigured for normal operation.

max7219（0x0F，0x00);

Table 11. R_{SET} vs. Segment Current and LED Forward Voltage

I_{SEG} (mA)	V_{LED} (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

6、MAX7219 的编程

```
#include “AT89x51.h”
#include “intrins.h”
#define CLK7219  P2_7;
#define CS7219  P2_5;
#define DIN7219   P2_6;
MAX7219(unsigned int dat)
{
char  i;
CS7219=0;
_nop_();
for(i=0;i<16;i++)
{
DIN7219=data>>(15-i)&0x01;
 CLK7219=1;
_nop_();
CLK7219=0
}
CS7219=1;
```

```
}
main()
{
MAX7219(0x0C01);
MAX7219(0x09FF);
MAX7219(0x0A0F);
MAX7219(0x0B07);
MAX7219(0x0F00);
MAX7219(0x0101);
MAX7219(0x0202);
.
.
.
MAX7219(0x0707);
MAX7219(0x0808);
}
```

存储类型

code	MOV	C @A+DPTR	程序内存空间
Data	MOV	A, @30h	内部内存空间，直接定位法，存取速度最快
Idata	MOV	A,#15H	内部内存空间采用间接定位法
Bdata	MOV	A,@R0	位元定位法
Pdata	MOVX	A,@R0	外部内存空间，以 R0 或 R1 暂存器做指标
Xdata	MOVX	A,@DPTR	外部内存空间，以 DPTR 暂存器做指标,存取速度最慢