

第一章:XLISP 系列 单片机综合仿真试验仪系统简介	
1. 1 系统简介	2
1. 2 各个模块接口的定义	3
第二章: 快速入门篇- 跟我来用 XLISP 系列作跑马灯实验	
2. 1 软件安装介绍	5
2. 2 软件操作	6
第三章 USB 接口安装指南 (仅限 XL1000)	
3. 1 USB 驱动程序安装	8
3. 2 特殊情况下的usb 安装	10
第四章 ISP 下载部份的应用	
4. 1 ISP 下载部份介绍	11
4. 2 XLISP 系列下载头之插头定义	12
4. 3 常用芯片的 ISP 相关引脚连接方法	13
第五章 XLISP 系列 仿真操作指南	
5. 1 仿真概述	14
5. 2 KEIL UV2 软件操作指南	15
第六章: XLISP 系列单片机系统实验	
MCS-51 单片机引脚说明	17
实验 1 最简单的八路跑马灯	18
实验 2 用 XLISP 系列试验仪做一个 8 路彩灯控制器	20
实验 3 8 路指示灯读出 8 路拨动开关的状态	21
实验 4 数码管静态扫描	22
实验 5 数码管动态扫描显示 01234567	23
实验 6 端口按键判断技术 (按键显示数字)	26
实验 7 矩阵按键识别技术	27
实验 8 74LS14 反向器实验	29
实验 9 74LS138 38 译码器部分实验	30
实验 10 74LS164 串入并出实验	31
实验 11 74LS165 并入串出实验	32
实验 12 DA 转换 dac0832 的原理与应用	34
实验 13 模拟/数字转换器 ADC0804	36
实验 14 小喇叭警报器试验	38
实验 15 红外线遥控试验	39
实验 16 汉字显示屏显示倚天一出宝刀屠龙 (仅限 XL1000)	42
实验 17 1602 液晶显示屏显示 A	44
实验 18 8155 试验 (仅限 XL1000)	46
实验 19 24C02 储存开机次数实验	48
实验 20 步进电机实验	50
实验 21 93c46 演示程序	51
实验 22 串行双向通信实验	53
实验 23 综合实验 18B20 数字温度显示系统	55
第七章 怎样产生 hex 文件? Dais 集成开发环境使用	58
第八章 常见问题解答	60
第九章 系统配置和售后服务指南	61
部分配套的例子程序说明	62
注意! 无论您是高手还是新手, 第一次使用前请用 1 小时时间仔细阅读此文件, 开卷有益	

第一章： XLISP 系列单片机综合仿真试验仪系统简介



1.1 系统简介

XLISP 系列单片机综合仿真试验仪（以下简称 XLISP 系列）是深圳市学林电子有限公司（ 51 测试网 www.51c51.com www.8951.com）综合多年经验开发出的多功能 8051 单片机平台（兼容 AVR/PIC 单片机的部份烧写实验功能）。本系列目前包含 XL600 单片机试验仪和 XL1000 USB 型单片机实验仪，集成常用的单片机外围硬件，ISP 下载线，单片机仿真器， 单片机试验板，编程器功能于一身，特别适合新手学习使用！

特别声明：

本手册为 XL600 和 XL1000 共用， 在以下描述中凡是注明仅 XL1000 适用的 则 XL600 不拥有此功能， XL600 用户可略过。

XL600 有哪些功能？ 可以做哪些试验？ XL600 本身由 20 多个全开放的模块组成，可以随心自由搭配，因此实际上可以做的试验是千变万化的。

MAX232 串行通信	12864 中文液晶（可选）	8 位发光管	8 位数码管显示
89C2051 核心监控	1602 字符液晶屏	74LS165 并入串出	74LS47 数码管译码
ADC 0804 模拟-数字	93C46 E PROM	74LS138 译码器	1 位继电器输出
Dac0832 数字-模拟	24C02 串行 EPROM	74LS164 串入并出	16*16 汉字点阵屏
16*16 矩阵按键开关	小喇叭警报器	74LS14 反向器	8 位拨动开关
单片机仿真	红外线遥控	步进电机	4 位按键开关
单片机在线编程	ISP 下载		

XL1000 与的 XL600 区别？

XL1000 在 XL600 的基础上增加了 FT232BM 对 USB 通信的支持，为此适用全 USB 的笔记本电脑。

增加了 8155 IO 扩展试验； 增加了 16*16 汉字点阵屏试验；

增加了系统晶体选择，可以直接在板上更换其他频率晶体。

增加了 18B20 数字温度采集； 增加了 8 位继电器输出（x1600 为 1 路继电器）；

增加了 4 路端口按键，按键数达到 8 路。

XLISP 系列的特点：

- 1 全开放的模块化设计：所有硬件资源对用户开放，搭配随心所欲，不会出现硬件束缚软件的情况。既可学习软件， 更可深入的了解硬件。
- 2 高品质的工艺： 本机采用高档透明亚克力面板， 铝合金外箱， 全贴片机器生产， 工艺精美绝伦！
- 3 超强的电路资源配置：集成了基本上所有单片机应用中可能遇到的功能模块部份，你再也不必去找其它零件，即可轻松完成您所需要的开发任务。
- 4 完美的例子程序： 集合本公司多年的经验，每个模块都有完整的带中文注释例子程序， 原理图， 接线方法， 很多都可以直接拿来应用。快速提高您的硬件，软件编程水平。
- 5 全 USB 通信接口（仅限 XL1000），完全适应最新的无串口并口的笔记本电脑。全中文操作界面，一线接法，简单至极。更适合野外环境下现场调试工作。
- 6 端口采用防插反设计，均配有连接照片和中文注解，即便您是初学者，也能轻松掌握。
- 7 配有 ISP 下载头。可选 232/USB 通信，CPU 控制编程，不受电脑配置及操作系统影响，稳定性一流。 可以作为一台独立的 ISP 下载线使用，支持芯片包括 51/AVR/PIC/24C 各系列！
- 8 配有 40P 外接仿真头，可以作为一台独立的 51 单片机硬件仿真器使用，通过 KEIL SOFT 软件配合， 即可对外部硬件以及板上资源实现单步调试， 断点， 全速等等全部功能。
- 9 全中文软件操作导航， 独有智能一键通设计，擦除， 写入， 校验，运行自动完成，特别设计的烧写实验仿真 3IN1 的公用卡座， CPU 插上后即不需要插拔，使用极其方便简洁。



系统需求:

WINDOWS98/ME/2000/XP/2003 操作系统。特有中英文操作界面， 可适用于海外地区英文/繁体操作系统！
最低 50MB 硬盘剩余空间。 1 个 USB 口（对于 XL600 需要一个串行口）

循序渐进的学习方法:

当您拿到这台精美的实验仪时, 请不要急于通电。我们建议您

第一： 先用 1 小时左右的时间仔细阅读本手册。

第二： 了解本套件各个模块，跳线的含义和定义。

第三： 从本套件的例子中挑选适合您了解程度的例子做实验， 如果正常了，那么想想这个例子为什么要这样写？这句话不要可不可以？ 想好了再改程序，重新做实验。如果有问题，最好的办法是登陆 <http://www.8951.com/bbs/index.asp> 访问官方论坛， 随时有数十位热心朋友和你在一起，分享学习的喜悦和进步！



1.2 各个模块接口的定义:

所属部份	端口名	简要说明
通信下载部份	JP14	232/USB 通信方式选择（选择实际使用的端口）（仅限 XL1000）
	JP13	232/USB 通信方式选择（选择实际使用的端口）（仅限 XL1000）
	JP17	编程/仿真方式选择，注意编程，实验时为弹起状态！仿真时为按下状态。
	K16	复位键。按此键程序重新运行（仅对实验/仿真部份有效）
	JP2	XL600 此端口只限于供电作用。XL1000 可使用 USB 端口。也可使用 232 端口）
	JP1	232 通信端口。（XL1000 可以使用 232 通信也可使用 USB 通信）
	JP23	ISP 下载头。可对其他芯片进行下载，此项操作要求对芯片有所了解。
电源部份	JP33	电源开关。按下电源开，电源指示灯亮（x11000 此开关只对辅助电源有效）
	J4	辅助电源输入。如果使用 232 通信，也可插入 USB 电缆供电。如果用 USB 通信，则可可不插此电源，除非负载过重，可以插入此电源提供辅助电力。
Cpu 仿真部份	IC16	实验卡座。实验时插入 AT89S51 等兼容 40 脚单片机实验，仿真时插入仿真监控芯片 SST89E564 进行仿真（方向向上，为 IC 缺口（第一脚）对卡座的扳手方向）
	JT1（仅限 XL1000）	晶体选择，当跳线跳到里面时，选择用户晶体，此时板上默认 12M 晶体无效！XL1000 用户须注意此跳线默认向外（板子的边缘，否则无法工作）
	JP44	P1 口输出（P1.0—P1.7）
	JP53	P3 口输出（P3.0—P3.7）
	JP51	P0 口输出（P0.0—P0.7）
	JP52	P2 口输出（P2.0—P2.7）
	JP59	40PIN 外接仿真头，请注意其引脚排列的方向。
AD 转换部份 ADC0804	J3	ADC0804 控制端。本套件例子中，分别接 P36, P37。
	JP35	ADC0804 数据端。本套件例子中，接 CPU 的 P1 口。
	JP30	外部模拟信号输入端。
	JP60	AD 跳线。默认为闭合状态，如果需要外接信号则跳开，从 AD 输入口 JP30 输入。
	ADJ13	Ad 试验。用于 AD 试验时模拟信号，10K 的可调整电阻。
DA 转换部份 DAC0832	JP24	DAC0832 控制端。本套件例子中，分别接 P36, P27。
	JP27	DAC0832 数据端。本套件例子中，接 CPU 的 P0 口。

	J2	DAC0832 输出端。接示波器看输出波形。(另外一端为接地)
液晶显示部份	JP41	液晶实验时闭合此跳线。 (注意, 不使用液晶时, 请将液晶屏取下以避免干扰!)
12864LCD	Jp49	12864 标准中文液晶屏接口。(此液晶为选配件)
1602LCD	J13	1602 标准字符液晶屏接口。
8 位数码管部份	JP5	数码管数据端, 本套件例子中, 接 CPU 的 P0 口。
	JP8	数码管显示位控制, 分别控制 1—8 位数码管亮灭。本例子中, 接 CPU 的 P2 口。
数码管译码部份 (74LS47)	JP19	74LS47 数据输入端, 本套件例子中 A, B, C, D 接 CPU 的 P1. 0, P1. 1, P1. 2, P1. 3, 可以直接实现在 LED2 单位数码管上解码显示数字。
	JP88	74LS47 数据输出端。本套件例子中可以接入 8 位数码管部份的 JP5, 以实现数码管显示的硬件解码, 简化程序的编写。
并入串出部份 (74LS165)	JP28	74LS165 数据输入端。
	JP21	74LS165 数据输出端。
三八译码部份 (74LS138)	JP22	74LS138 数据输入端。
	JP25	74LS138 数据输出端。本套件例子中, 接 JP32 八路指示灯观察输出情况。
串入并出部份 (74LS164)	JP20	74LS164 数据输入端, 本套件例子中, A/B CLK 分别接 CPU 的 P3. 0, P3. 1。
	JP26	74LS164 数据输出端, 本套件例子中, 接 JP32 八路指示灯观察输出情况。
反向器部份 (74LS14)	JP29	74LS14 数据输入端。
	JP34	74LS14 数据输出端, 信号由此处理后, 高电平变低点平, 低电平变高电平。
步进电机部份	JP31	步进电机控制信号输入端, 本套件例子中 A, B, C, D 分别接 CPU 的 P1. 0, P1. 1, P1. 2, P1. 3
	JP36	步进电机输出。
93C46 部分	JP46	93C46 控制。本套件例子中, DO, DI, SK, CS 分别接单片机的 P34, P35, P36, P37
24C02 部分	JP38	24C02 控制。本套件例子中, SLC, SDA 分别接 CPU 的 P36, P37
小喇叭部份	JP16	JP16 是小喇叭信号输入。本套件例子中, 接单片机的 P3. 3 口
遥控部份	JP45	JP45 是红外线接收头的信号输入。本套件例子中, 接单片机的 P3. 7 口
18B20 部分	JP39	数字温度采集数据输出端。本套件例子中, 接单片机的 P3. 0 口 (XL600 的 18B20 为选配件, 需要可自行添加)
8 路继电器输出 (仅限 XL1000)	JP1	JP1 分别对应 JP00—JP07 八个继电器。继电器动作时, JP00—JP07 为闭合状态, 无动作时, JP00—JP07 为开路状态。JP18 (XL600 对应的 1 路继电器)
汉字显示屏部份 (仅限 XL1000)	JP55	汉字显示的行扫描上半部分。本套件例子中, 接 CPU 的 P0 口。
	JP54	汉字显示的行扫描下半部分。本套件例子中, 接 CPU 的 P2 口。
	JP62	汉字显示列扫描, 为 74LS154 的输入口。本套件例子中, A, B, C, D 分别接 CPU 的 P1. 0, P1. 1, P1. 2, P1. 3
拨动开关部份	JP40	JP40 分别对应 J00—J07 八个拨动开关。开关为 ON 状态时, 对应端口则为接地低电平, 可以用来做数据输入实验。开关为 OFF 状态时, 对应端口则悬空。
八路指示灯部份	JP32	JP32 分别对应 D00—D07 八个指示灯。灯亮时, 表示对应端口状态为低电平。可以用来做流水灯实验以及端口电平的指示灯。
矩阵按键部份	JP50	JP50 对应 4*4 键盘的 4 行和 4 列。本套件例子中, 接在 CPU 的 P1 口。
端口按键部分		XL1000 为 8 路按键端口为 JP48, (XL600 为 4 路按键为 JP37), 按下为低电平
8155 扩展部分	仅 XL1000	Jp63: 8155 控制 jp58: 8155 数据 JP61: 8155PB 口引出

第二章：快速入门篇-跟我来使用 XLISP 系列作跑马灯实验



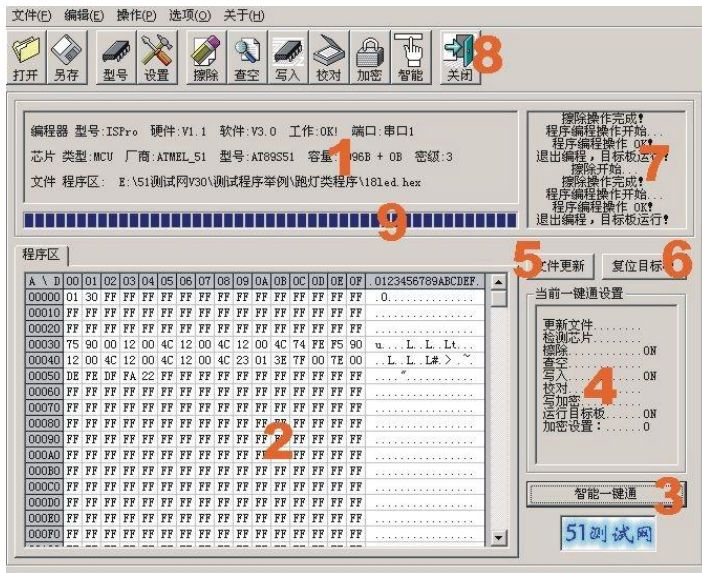
2.1 软件安装：

插入随机的驱动光盘，打开 驱动程序\XLISP 系列试验仪目录。 双击其中的 setup.exe， 按照提示，点下一步直到完成安装：如果提醒您系统文件过旧则需要重新启动计算机，重启后继续安装。

// COPY/XLISP /

备份的免安装的全部文件,如果您确实没有办法安装,可以直接 COPY 该目录到您的硬盘,运行 XLISP. EXE。另外不要忘记把所有文件的只读属性去掉。

xlisp 软件具有先进的串口自动检测功能， 特别方便使用！



①软件状态信息窗口 分别显示：当前的编程器的信息，连接的工作状态，使用的串口端口，当前芯片资料，当前目标文件。	②程序区与数据区 显示被烧录的数据，可以编辑。
③智能一键通 本软件的特色之一，可以实现快捷编程。详细资料请参考有关章节。	④智能一键通设置状态窗口 显示智能一键通当前的设置状态。
⑤文件更新 手工更新目标文件。	⑥复位目标板 按动该按钮可以复位目标板，方便在线调试程序。
⑦操作状态汇报窗口 动态显示当前所有编程操作情况。	⑧菜单与工具条 详细功能看了就知道。
⑨编程进度指示 动态显示当前编程操作进度。	



硬件安装:

打开 XLISP 系列的包装。插入随机的串口线（9 针对 9 针），然后插上随机的 USB 电缆（用于提供 5V 电源）。按下电源开关，此时电源指示灯亮。同时注意 XL600 烧写指示灯亮大约 1 秒后熄灭，此时表示系统自检正常！（XL1000 烧写指示灯为长亮。）

确认 232/usb 选择跳线 JP14, JP13 都跳在 232 一端。（仅限 XL1000）

确认 编程/仿真按钮处于弹起（编程）状态。

确认 CPU 卡座上的 CPU 为 AT89S51 并且已经压下扳手安装好。

XLISP 典型应用接线图1 在系统编程实验

软件用 XLISP，直接对卡座上的 AT89S51 编程，接上下载头即可对目标板编程。



连线如图所示

注意



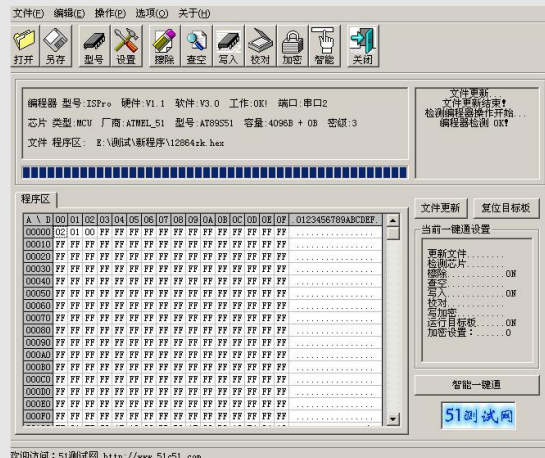
2.2 软件操作



下面介绍基本的软件操作：双击桌面的 xlisp 启动程序，此时弹出选择串口的对话框，此时可以选择“自动检测”即可检测到目前正在使用的端口。一般默认情况下是串口 1

（USB 通信方式，可能是串口 3, 4 等。（仅限 XL1000）具体请看第三章 USB 接口安装指南）在 XLISP 操作过程中遇到重新启动等情况，进入时可选择：选项/串口/自动检测，将非常方便！

■1 此时如果串口的设置正确。那么会出现以下主界面：操作状态汇报窗口显示编程器监测 OK！
否则就是没有正确的设置串口：请检查串口设置的号码是否正确？电源指示灯是否亮？编程仿真模式是否处于编程（弹起）状态？ USB/232 选择跳线是否选择在 232 上？ 仅限 XL1000



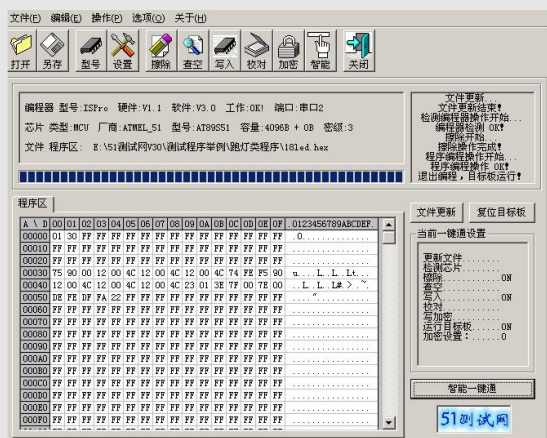
■2 为了简化操作步骤，第一次使用时，应当确认一下软件的设置。首先点击下图的“设置”对话框在自动编程的选项中选择：“擦除” “写入” “运行目标板”（下图打钩的地方）然后点确定。由于一般情况下，查空，校对，写加密等等选项没有必要选择，相反会浪费您宝贵的编程时间，当然，您也可根据您的需要加上这些选项。



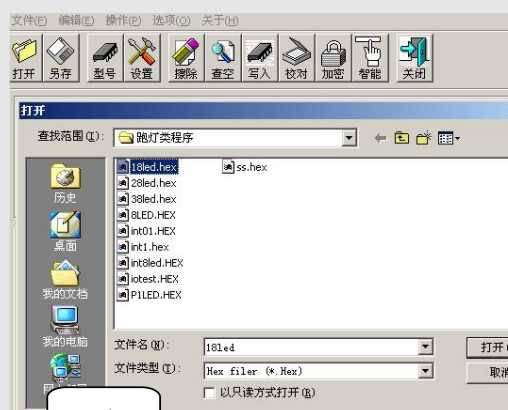
■3 在芯片类型的对话框中选择 MCU\ATMEL_51\AT89S51 然后点确定。



■5 此时点击：“智能一键通”，将出现烧写进度条指示，对话框中出现
擦除开始…。擦除操作完成。
程序编程操作开始…。
程序编程操作 OK！
退出编程，目标板运行！



■4 点击 主界面的 “打开”，找到配套光盘中的：测试程序举例\跑灯类程序\18LED.HEX，然后点确定



注意

注意！在烧写操作的过程中，**烧写指示的发光管会亮，烧写完毕，发光管熄灭**。此灯的作为烧写状态的指示非常重要！
至此，程序已经烧写完成。

下面我们要连接硬件了：由于这个程序是一个 P1 口的跑马灯，取出一条随机配备的 8PIN 排线，一端插入 XLISP 系列的 CPU 部份的 P1 口（JP44）另外一端插入八路指示灯部份的 JP32，我们就看到 P1 口的 8 个发光管轮流闪烁。表示运行成功了。
如果没有运行那么请按一下系统复位开关（K16）即可。

其他程序的硬件连接请参考相关的程序说明！

注意

在接插排线过程中，一般并不需要断电操作。当然，也可根据需要自行决定。

第三章 USB 接口安装指南 (仅限 XL1000) 注意: x1600 无此部分功能

很多新型的笔记本电脑都已经取消了 COM 口, 而只留有 USB 端口供给用户使用。现在我们来讲解一下用 USB 通信方式来使用 XLISP 系列套件。XL1000 的 USB 通信方式的实质是利用 FT232BM 芯片, 实现 USB 通信协议到 232 通信协议的转换。所以, 在以后所需要用到 COM 口的时候, 我们实际是用 FT232BM 芯片虚拟出来的一个串口供用户使用, 它本质还是一个串行 232 口。请务必理解这一点。



3. 1 USB 驱动程序安装

请遵循如下步骤去安装 USB 转通用串口的驱动程序 (以操作系统使用 Windows 2000 为例)

■1 打开电脑并确认 USB 端口可用且工作正常。

找到新的硬件向导

欢迎使用找到新硬件向导

此向导帮助您为硬件设备安装驱动程序。

要继续, 请单击“下一步”。

< 上一步(B)

下一步(N) >

取消

2 请把 XLISP 系列的 USB 线插到 USB 端口上, 串口线不要连接, 也不必连接电源适配器。此时电源指示灯亮, 系统将会检测到它。并自动运行“欢迎使用找到新硬件向导”帮助你设置新设备, 单击“下一步”继续;

找到新的硬件向导

安装硬件设备驱动程序

设备驱动程序是可使硬件设备与操作系统工作的软件程序。

该向导将完成这个设备的安装:

USB <-> Serial

设备驱动程序是一份可以使硬件设备工作的软件程序。Windows 需要您的新设备的驱动程序文件。找到驱动程序文件的位置并完成安装, 请单击“下一步”。

您期望向导做什么?

☒ 搜索适合于我的设备的驱动程序(推荐)(S)。

☐ 显示已知设备驱动程序的列表, 从中选择特定驱动程序(D)。

< 上一步(B)

下一步(N) >

取消

3、把驱动程序光盘插入光驱, 单击“下一步”继续。对于 Windows 2000 操作系统来说, 选择“搜索适合于我的设备的驱动程序(推荐)”选项, 单击“下一步”继续;

找到新的硬件向导

找到驱动程序文件

您想让 Windows 在什么地方寻找驱动程序文件?

为下列硬件设备搜索驱动程序文件:

USB <-> Serial

向导在您的计算机驱动程序数据库和所指定的下列可选搜索地址中搜索合适的驱动程序。

单击“下一步”开始搜索。如果在软盘或 CD-ROM 驱动器搜索, 在单击“下一步”前, 插入软盘或 CD。

可选的搜索位置:

☐ 软盘驱动器(D)

☐ CD-ROM 驱动器(C)

☒ 指定一个位置(S)

☐ Microsoft Windows Update(U)

< 上一步(B)

下一步(N) >

取消

4、选择“指定一个位置”, 并单击“浏览”按钮, 打开光盘中的“F: (当前光驱盘符) \usbdriver2.0”目录, 按“确定”键继续;

找到新的硬件向导

请在选定驱动器中插入厂商的安装盘, 然后单击“确定”。

厂商文件复制来源(C):

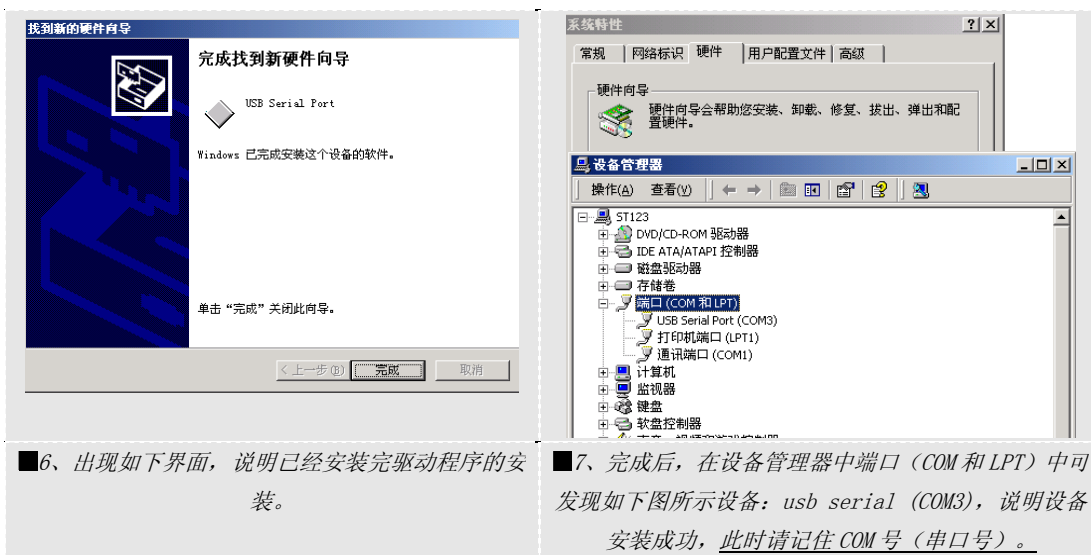
F:\usbdriver2.0

浏览(B)...

确定

取消

5、找到它的驱动程序之后, 单击“下一步”继续安装。Windows 将复制所需的文件到你的计算机硬盘上。



■6、出现如下界面，说明已经安装完驱动程序的安装。

■7、完成后，在设备管理器中端口（COM 和 LPT）中可发现如下图所示设备：usb serial (COM3)，说明设备安装成功，此时请记住 COM 号（串口号）。

■8、若出现“Port already Open”等端口错误，请进入设备管理器，将相应的串口设备先停用，再启用，重新启动计算机即可：



特别注意：

安装完成 USB 驱动程序以后：运行 XLISP 驱动，出现选择串口号码的时候，可以直接点自动检测，这时 XLISP 软件会自动检测当前正在使用的串口，然后点确定就可以了。同样的，作 KEIL UV2 仿真试验，和作计算机串口通信实验的时候都要选择相应的串口号码。



(USB 方式运行 XLISP 软件提示找不到串口)



(点自动检测，提示 COM3 OK!) 点确定即可。

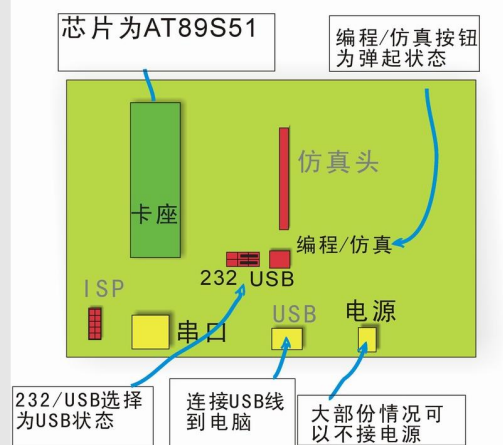
注意：在 XLISP 操作过程中遇到重新启动等情况，进入时可选择：选项/串口/自动检测，将非常方便！



3. 3 使用 USB 方式使用 XLISP 软件进行在系统编程实验的示意图：

XI1000 典型应用接线图3 USB 方式在系统编程实验

软件用 XLISP，直接对卡座上的 AT89S51 编程，接上下载头即可对目标板编程。





3.2 特殊情况下的 usb 安装:

一般情况下的 usb 安装如上所述, 特别情况下, 您的电脑可能已经安装了其他的 usb 打印机, usb 编程器等。在这种情况下, 分配给 XLISP 系列的 usb 虚拟串口可能是串口 5, 甚至是串口 6! 而 keil 仿真软件, 有些串口调试软件可能只有 com1, com2, com3, com4 四个串口, 如何解决呢? 比较笨的办法是干脆重新安装系统, 在新的系统中首先安装 XLISP 系列的 usb 驱动, 这样系统分配的串口自然靠后了。

下面介绍一种比较好的办法来解决这个问题:



图 1 当 usb 设备较多的时候, 系统分配的虚拟串口号为 com5, 导致无法使用)



图 2: 此时我们把不用的系统默认的 com2 停用, 鼠标指向后, 点右键选择: 停用)



图 3: com2 停用后上面打了一个红叉, 指向 usb 虚拟出来的 com5, 点右键, 察看它的属性)

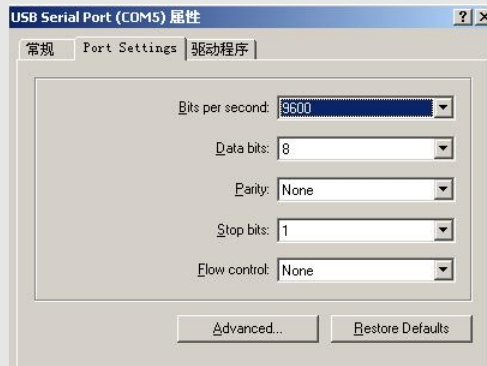


图 4: 进入 usb serial port (com5) 属性后, 点上面的 port settings 端口设置), 然后点 Advanced... (高级属性)

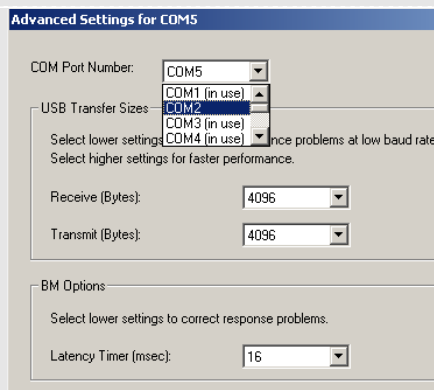


图 5: 然后在 COM port number 选项中选择 com2。)



图 6: 系统已经把 com2 分配给 usb 虚拟串口, 以后您就可以使用此 com 口啦)

第 4 章 ISP 下载部份的应用



4.1 ISP 下载部份介绍

ISP (In-System Programming) 在系统可编程, 指电路板上的空白器件可以编程写入最终用户代码, 而不需从电路板上取下器件, 已经编程的器件也可以用 ISP 方式擦除或再编程, ISP 技术是未来发展方向。XLISP 系列的 ISP 部份是业界首创的具备传统编程器的可靠性与下载线的方便性于一身的前卫产品。通过单片机控制时序, 232 串行通信, 彻底解决并口 ISP 兼容性, 可靠性的问题。将下载头的相关引脚引入目标板, 即可方便快速的对目标板在系统编程。



ISP 下载部份支持的芯片列表:

51 系列单片机:

AT89S51, AT89S52, AT89S53, AT89S8252, AT89LS51, AT89LS52, AT89LS53, AT89LS8252

AVR 系列单片机:

AT90S2323, AT90S2343

AT90S4433, AT90S8535

ATmega16, ATmega16L, ATmega161, ATmega161L, ATmega163, ATmega32, ATmega32L

ATmega323, ATmega323L, ATmega64, ATmega64L

ATmega8, ATmega8L, ATmega8515, ATmega8515L

ATmega8535, ATmega8535L

ATtiny12, ATtiny12L, ATtiny13, ATtiny13L

ATtiny15, ATtiny15L, ATtiny22, ATtiny22L, ATtiny26, ATtiny26L

PIC 系列单片机:

PIC16F870, PIC16F871, PIC16F872, PIC16F873, PIC16F874, PIC16F876, PIC16F877,

PIC16F873A, PIC16F874A, PIC16F876A, PIC16F877A

AT24CXX:

AT24C01, AT24C02, AT24C04, AT24C08, AT24C16, AT24C32, AT24C64, AT24C128, AT24C256, AT24C512



使用 XLISP 系列下载头的几种优点:

- 1、如果您设计的产品是开放的或可升级的系统, 那么您需要使用本下载线或其他下载线对单片机不断在线升级。
- 2、发货前升级产品软件版本。很多产品在交付客户或即将交付客户的时候可能因为种种 BUG 而需要修正, 然而 IC 已经焊接到电路板上无法拆卸。这个时候您可以使用 XLISP 系列下载头方便的为其在线修改。



灵活使用 XLISP 系列下载头的几种方式:

- 1、在目标电路板上预留下载插座, 可以方便的在线下载程序代码。
- 2、如果您现在的目标电路板上没有预留下载插座而您又想使用 XLISP 系列的下载头, 您可以自制一个仅带裸线的下载头, 把线头焊接到您的电路板上即可。



4. 2 XLISP 系列下载头之插头定义(使用前请仔细了解此表格的相关含义!)

1 NC(悬空)	2 NC (悬空)	3 NC(悬空)	4 +5V	5 GND
6 NC (悬空)	7 SCK/SCL	8 MISO/SEN/WP	9 MOSI/SDA	10 RST



下载头排列方向和一般芯片方向相同， 逆时针旋转。

红色线为第一脚

Isp 下载头



插头的 (1) 脚处有一个三角符号，请确认以免接错线。

小提示：如果您还是对各个引脚的顺序不清楚， 连接好后最好用万用表量一下。



特别注意：

下载头的第 4，第 5 脚已经提供了+5V 电源，为此目标板可以不用电源。 在使用下载头的之前，请务必检查目标板电源是否短路？以及各 ISP 相关引脚是否接错？ISP 下载头部份的应用属于单片机应用中较高级的范围，如果您没有过此类应用， 那么请在充分了解 ISP 相关资料后再行动手实验。

另外，由于 XLISP 系列套件本身利用了 XLISP 软件的 S51 系列单片机的下载部份，在使用 ISP 下载头的时候， XLISP 系列的实验卡座上不要有单片机芯片。



4. 3 常用芯片的 ISP 相关引脚连接方法

以下介绍部份常见芯片的 ISP 相关引脚连接方法， 软件操作请参考第二章：跟我来使用 XLISP 系列作跑马灯实验（快速安装指南）

AT89S 系列、AVR 系列的连接方法（参考自制的 89S51 下载目标板）

下载头<----->目标单片机

RST <----->RST

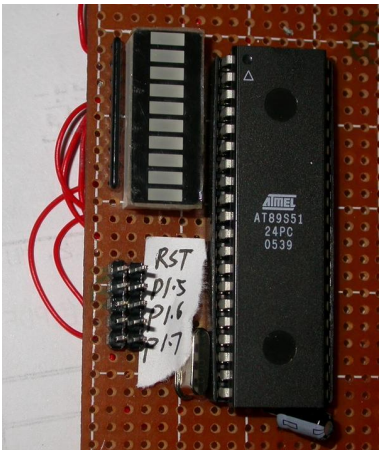
MOSI <----->MOSI

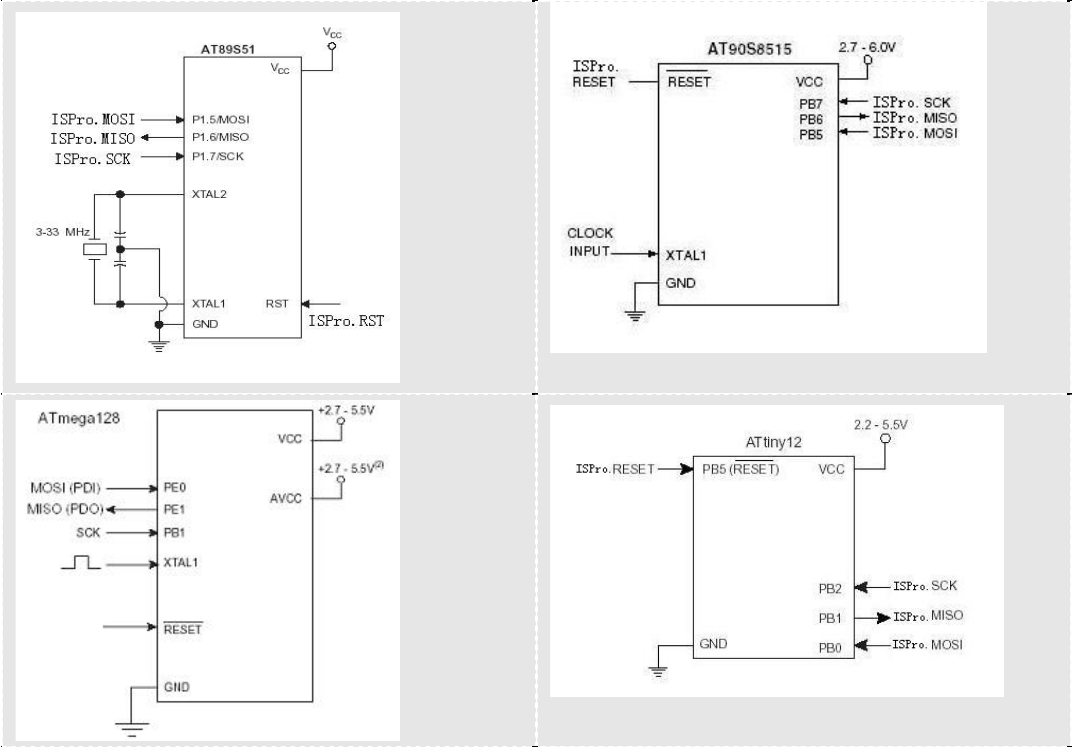
MISO <----->MISO

SCK <----->SCK

VCC <----->VCC

GND <----->GND





- 1、AT89S8252 内部还有数据 EEPROM 可以在线编程。
- 2、目标电路板最简系统需求：晶振（4MHz 以上），目标 MCU，5V 电源（供目标板及下载型编程器用）。
- 3、设计电路板时目标单片机的 ISP 相关管脚最好专门供 ISP 使用，而不要设计其他功能。
- 4、如果复位电路由 RC 电路组成则 RESET 管脚可以直接相连接，同时提醒您为了 MCU 的安全电容不能过大，建议取值 1uF，最好不要超过 10uF。

PIC 系列的连接方法注意：PB7，PB3，PB6 以芯片的数据手册为准，不同的 PIC 可能不同。

PIC16F87X

Pin Name	During Programming	
	Function	Pin Type
RB3	PGM	I
RB6	CLOCK	I
RB7	DATA	I/O
MCLR	RST	P*
VDD	VDD	P
VSS	VSS	P

下载头<----->目标单片机

RST <----->Vpp(/MCLR)

SDA <----->DATA (PB7)

SEN <----->PGM (PB3)

SCK <----->CLOCK (PB6)

VCC <----->VCC

GND <----->GND

AT24CXX 系列的连接方法

AT24CXX 8-Pin PDIP

NC 1 8 VCC

NC 2 7 WP — ISPPro.WP

NC 3 6 SCL — ISPPro.SCL

GND 4 5 SDA — ISPPro.SDA

下载头<----->AT24CXX

RST <----->RST（控制 24 的单片机）

SDA <----->SDA

WP <----->WP

SCK <----->SCK

VCC <----->VCC

GND <----->GND

第五章 XLISP 系列 仿真操作指南

(由于 KEIL 操作较复杂,如果您是新手,请仔细参考以下说明书,一步步的操作,相信您一定能够成功!)



5.1 概述: XLISP 系列的仿真部份通过随机的 SST89E564 仿真监控芯片实现。通过和 KEIL 软件配合使用,

不仅可以实现对板上资源的仿真, 更可通过随机的 40PIN 仿真头对外接的目标板进行仿真。

. 可仿真 89C51、89C52、89S51、89S52、89C58 等等 51 内核的单片机。

. 直接支持 KEIL C51 的 IDE 开发仿真环境。

. USB/RS-232 通讯接口, 波特率 4.8 Kbps- 57.6 Kbps 自适应。特别适合无串口的笔记本电脑用户。

. 监控程序占用用户的资源少,全速运行不占用资源。

. 片内 63K 程序空间可以随时进行在线程序更新。

. 可单步、断点、全速、可参考变量、RAM 变量。

. 支持汇编、C 语言, 混合调试。

. 板载仿真头接口可以和任何的试验板、目标板进行连接, 从而达到硬件仿真的无限扩展。



硬件安装: 首先将 9 针串口插头接到 PC 的串口上, 比如 COM1。然后连接好 usb 电缆供电, 将 仿真芯片 (不

是 at89s51) 插入试验卡座并锁紧。(注意: 此处是串口方式运行, 如果是 XL1000 的 USB 方式运行, 那么插入 USB 线同时注意 JP13, JP14 232/USB 选择到 USB 位置) 如果直接仿真板上的资源, 那么首先用随机的排线根据实验目的把相关 I/O 端口连接到相关的硬件上, 仿真结果直接体现到 XLISP 系列的试验板上。

如果您需要仿真自己的目标板, 把仿真头插入目标板, 仿真结果将体现到您自己的目标板上。

XLISP 典型应用接线图 2 串口方式在系统仿真实验

软件用 KEIL UV2, 对板上资源仿真, 接上仿真头即可对目标板仿真。

芯片为 SST89E654
仿真监控

编程/仿真按钮
为按下状态



连接串口线
到电脑

连接USB电缆供电

红色线为第一脚

仿真头

请注意仿真头插入方向!



软件安装: 直接拷贝光盘\开发工具\keil\到c盘的根目录, 去掉所有文件的只读属性。

(注意必须是c盘根目录, 不要安装在其他目录)

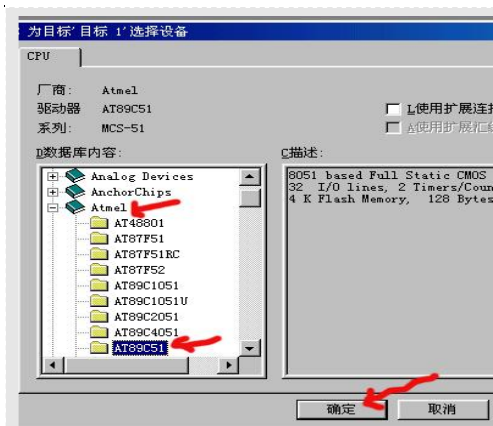
执行 keil\uv2\uv2.exe 即可 (可将文件发送到桌面快捷方式) KEIL 软件就是他的驱动软件。

(uv2 的图标)



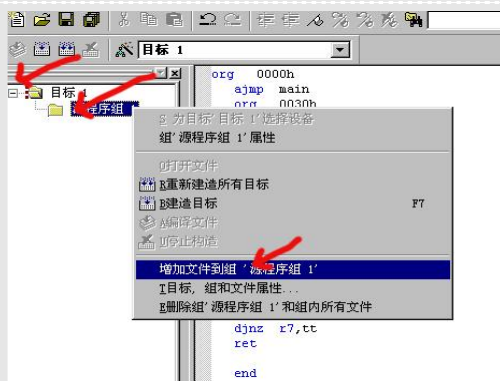


5.2 建立您的第一个项目(软件操作指南):

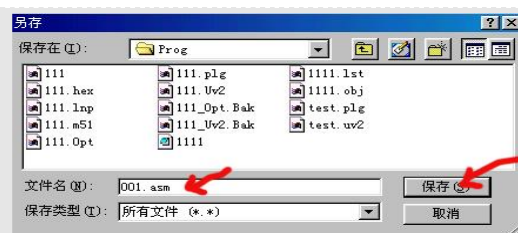


1 点击 Project(工程)菜单, 选择 New Project (新工程), 在“文件名”中输入您的第一个程序项目名称, 假定我们用“test.”.保存后的文件扩展名为 uv2 这是 KEIL uVision2 项目文件扩展名, 以后我们可以直接点击此文件以打开先前做的项目。

这时会弹出让你选择单片机型号的对话框, 我们选择 ATMEL---AT89C51. 注意我们实际使用的是 89s51, 他们的资源基本是一致的, 不必理会。



3. 点击 Target 1 (目标 1) 前面的 “+”, 出现 Source Group 1, (源程序组 1) 选中右键点选 “Add files Group Source Group 1” (增加文件到源程序组 1) 这时选择文件类型为 Asm, 再选中 001.asm 文件, 再按 Add 添加, 在随后的提示框中按 “确定”



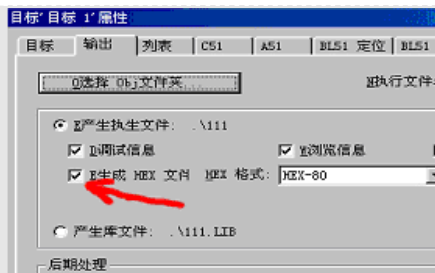
2 汇编源文件的建立: 点击 FILE (文件) 菜单中的 NEW... (新文件) 命令新建一个文本编辑窗口, 在里面输入一个程序范例如流水灯程序, 然后点击 FILE (文件) 菜单中的 SAVE (保存) 命令保存文件, 注意**必须输入文件的后缀名 .asm**, 例如保存为 d:/prog/001.asm

(注意不要有中文目录, 文件名不超过8个字符, 否则编译不通过! 如果您是第一次使用, **那么我建议您直接保存在c盘的根目录, 文件名同样取001.ASM, 不要试图把他保存在桌面/我的文档等等!)**

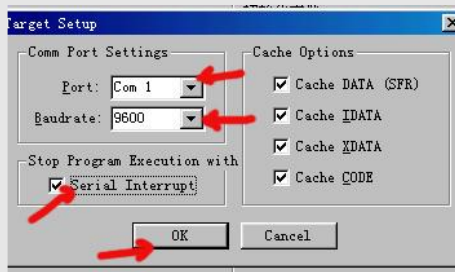


4. 仿真部分采用Mon51协议, 在使用之前必须对软件项目进行如下设置:

单击“Project”(工程)菜单, 再在下拉菜单中单击“options for target target 1”(目标 1 属性) 在下图中, 单击“Target”(目标) 输入仿真器的工作频率 (12MHz)。



5. 选择output(输出)“Create HEX file” (产生hex文件) 的选项, 以便汇编后生成HEX代码, 供编译器使用。



7单击“Settings”(设置)选择要使用串口必须和实际相符合, 你的计算机可能是COM2或者其他, 对此不能确定可以通过察看 控制面板\硬件\端口\通讯端口来解决, 同时注意电脑上不能够同时运行其他可能占用串口资源的软件例如XLISP下载软件, 串口调试软件等等也必须退出, 否则将引起冲突, (如果您使用的是USB通信方式, 那么选择系统分配的虚拟串口号 仅限XL1000)



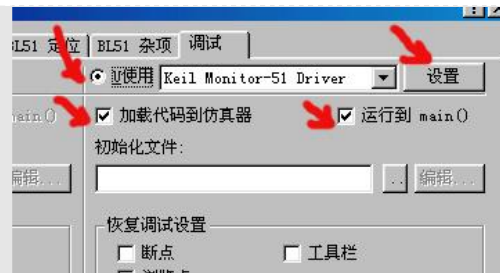
9 现在按图第三个红色的 d (debug)或按 Ctrl +F5 快捷键可以进入仿真



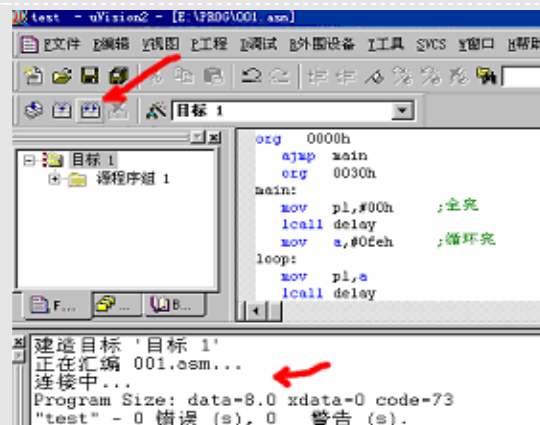
然后按一下图示的”全速运行”按钮。连接上相关的实验资源, 本试验用一条 8PIN 的数据排线把 CPU 部份的 P1 口 (JP44) 连接到八路指示灯部份的 JP32。这时你会看到 XLISP 系列的八个红色 LED, 轮流点亮, 表示运行成功, 也可以查看相关的变量和参数, 非常方便。

退出仿真时, 先按XLISP系列的复位键, 然后再点一下d, 就又重新回到编辑模式。修改程序后重新编译, 再点一下红色的d, 又可进入仿真模式。

到这里也就完成了keil软件的标准操作过程. 你可以通过连续点单步运行, 来观察试验板上的变化和各个寄存器的变化, 从而体会程序运行的过程。注意: 进入仿真和退出仿真时请按一下XLISP系列的复位键。



6. 在“Debug”(调试)中点选“KeilMonitor-51 Driver”, 同时选中加载代码到仿真器, 运行到main0, 然后点击设置



8. 按以下两个向下小箭头的图标进行编译, (或者按 F7 快捷) 编译成功后如会出现下图红箭头所指的文字, 表示编译成功! 此时就生成了 hex 文件, 同时, 有了这个 hex 文件, 我们也可用 isp 方式烧写到单片机试验)。然后可以进行硬件仿真了, 将仿真芯片插入 XLISP 系列的试验卡座, 开始仿真退出仿真时务必按一下 XLISP 系列上的复位按键。

此时界面将出现连接成功的提示! (否则设置有误请仔细检查)



示)

(仿真连接成功的提示)



MCS-51 单片机引脚说明

51 系列单片机 8031、8051 及 89c51/89s51 均采用 40Pin 封装的双列直接 DIP 结构。下图是它们的引脚配置:40 个引脚中,正电源和地线两根,外置石英振荡器的时钟线两根,4 组 8 位共 32 个 I/O 口,中断口线与 P3 口线复用。现在我们对这些引脚的功能加以说明:

• Pin20:接地脚。

P1.0	1	40	V _{CC}
P1.1	2	39	P0.0/AD0
P1.2	3	38	P0.1/AD1
P1.3	4	37	P0.2/AD2
P1.4	5	36	P0.3/AD3
P1.5	6	35	P0.4/AD4
P1.6	7	34	P0.5/AD5
P1.7	8	33	P0.6/AD6
RST	9	32	P0.7/AD7
RXD/P3.0	10	31	EA/V _{PP}
TXD/P3.1	11	30	ALE/PROG
INT0/P3.2	12	29	PSEN
INT1/P3.3	13	28	P2.7/A15
T0/P3.4	14	27	P2.6/A14
T1/P3.5	15	26	P2.5/A13
WR/P3.6	16	25	P2.4/A12
RD/P3.7	17	24	P2.3/A11
XTAL2	18	23	P2.2/A10
XTAL1	19	22	P2.1/A9
GND	20	21	P2.0/A8

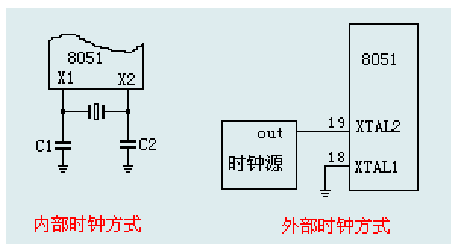
PDIP

• Pin40:正电源脚,工作时,接+5V 电源。

• Pin19:时钟 XTAL1 脚,片内振荡电路的输入端。

• Pin18:时钟 XTAL2 脚,片内振荡电路的输出端。

8051 的时钟有两种方式,一种是片内时钟振荡方式,但需在 18 和 19 脚外接石英晶体(2-12MHz)和振荡电容,振荡电容的值一般取 10p-30p。另外一种方式是外部时钟方式,即将 XTAL1 接地,外部时钟信号从 XTAL2 脚输入。



Pin39-Pin32 为

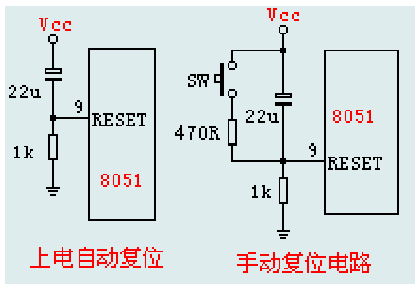
输入输出(I/O)引脚:

P0.0-P0.7 输入输出脚。

Pin1-Pin8 为 P1.0-P1.7 输入输出脚。

Pin21-Pin28 为 P2.0-P2.7 输入输出脚。

Pin10-Pin17 为 P3.0-P3.7 输入输出脚。



• Pin9:RESET/V_{pd}复位信号复用脚,当 8051 通电,时钟电路开始工作,在 RESET 引脚上出现 24 个时钟周期以上的高电平,系统即初始复位。

8051 的复位方式可以是自动复位,也可以是手动复位,见下图。此外,RESET/V_{pd}还是一复用脚,V_{cc}掉电期间,此脚可接上备用电源,以保证单片机内部 RAM 的数据不丢失。

• Pin30:ALE/PROG当访问外部程序器时,ALE(地址锁存)的输出用于锁存地址的低字节。而访问内部程序存储器时,ALE 端将有一个 1/6 时钟频率的正脉冲信号,这个信号可以用于识别单片机是否工作,也可以当作一个时钟向外输出。

如果单片机是 EPROM,在编程期间,PROG 将用于输入编程脉冲。

• Pin29:PSEN当访问外部程序存储器时,此脚输出负脉冲选通信号,PC 的 16 位地址数据将出现在 P0 和 P2 口上,外部程序存储器则把指令数据放到 P0 口上,由 CPU 读入并执行。

• Pin31:EA/V_{pp}程序存储器的内外部选通线,8051 和 8751 单片机,内置有 4kB 的程序存储器,当 EA 为高电平并且程序地址小于 4kB 时,读取内部程序存储器指令数据,而超过 4kB 地址则读取外部指令数据。如 EA 为低电平,则不管地址大小,一律读取外部程序存储器指令。

阅读说明:



表示每一章节的开始



表示该实验的源程序



表示该实验的接线方法

第六章 单片机实验与实例

由于本文篇幅有限，特别截取几个有代表意义的例子，贴出源程序，以及运行照片，接线方法，线路原理图，供初学者参考。XLISP 系列并不是仅仅能够做以下的几个试验，由于本实验仪采用模块化结构，搭配方式千变万化，硬件，程序编程更是随心所欲，读者可举一反三，练习掌握。

更多汇编程序以及 C 语言程序请参考配套光盘的例子程序，以及 www.8951.com 的最新更新！



实验 1：最简单的八路跑马灯

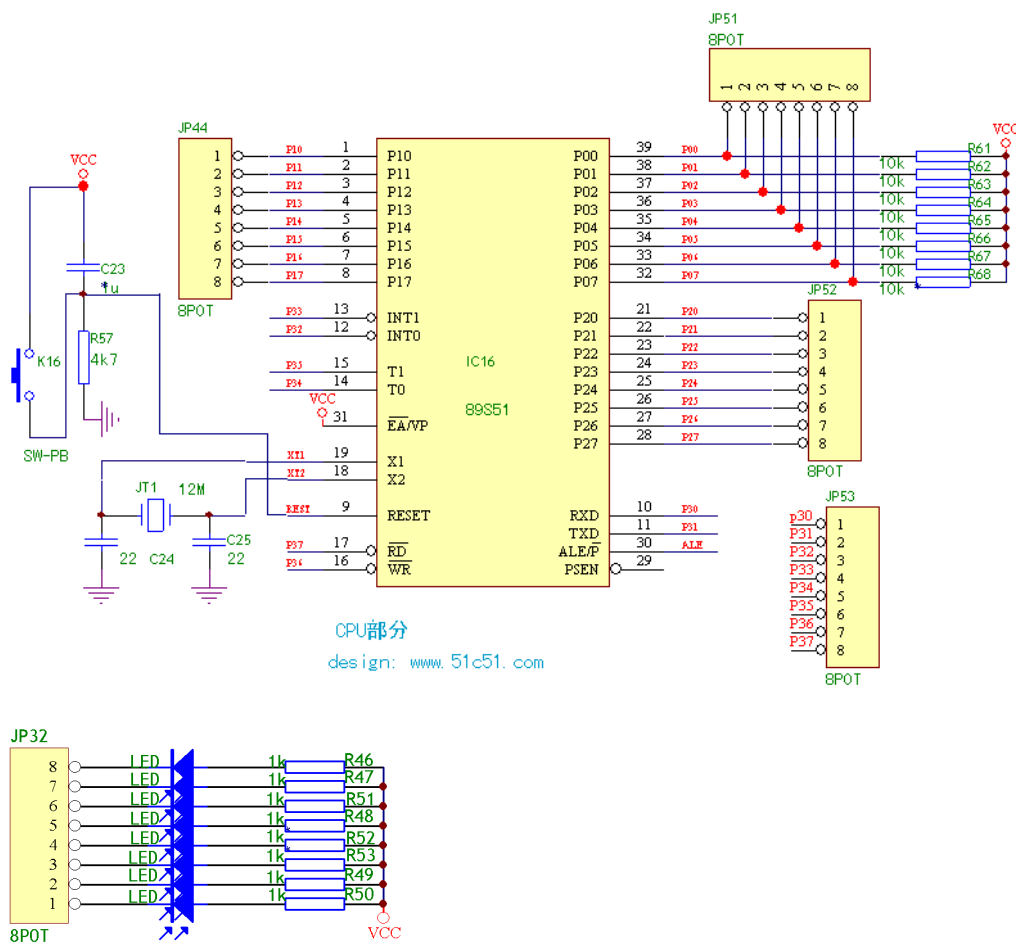
本例实验主要用到了延时子程序，clr, lcall, ajmp 指令，通过轮流点亮 p1.0 ... P1.7 实现效果。用户可以通过此程序的学习，初步掌握单片机的 IO 端口操作。

练习 1：怎样把延时的时间改为更短，或者更长？

练习 2：怎样用 P0, P2, P3 端口来操作？

练习 3：怎样实现不同的跑马花样？

相关原理图：

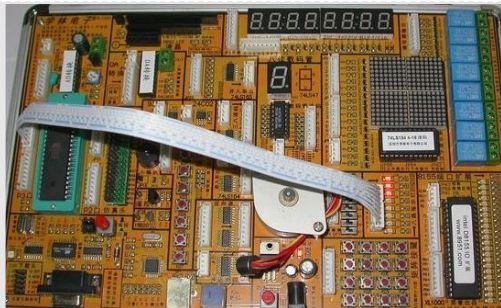




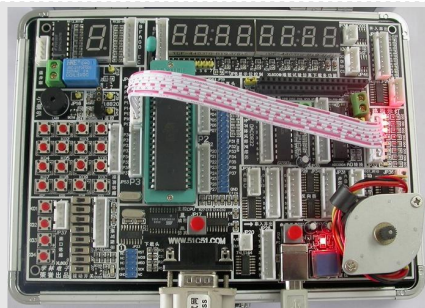
参考程序:

```
org 0000h ;开始
ajmp loop
org 0030h ;到 0030h 处避开 00-30 的敏感地址
loop:
mov p1,#0ffh;关闭所有灯
clr p1.0 ;点亮灯 p1.0
lcall delay ;延时一段时间?
clr p1.1 ;点亮灯 p1.1
lcall delay
clr p1.2 ;点亮灯 p1.2
lcall delay
clr p1.3 ;点亮灯 p1.3
lcall delay
clr p1.4 ;点亮灯 p1.4
lcall delay
clr p1.5 ;点亮灯 p1.5
lcall delay
clr p1.6 ;点亮灯 p1.6
lcall delay
clr p1.7 ;点亮灯 p1.7
lcall delay
AJMP LOOP;到最开始 loop 处重新运行
delay: mov r5,#20 ;延时。
d1: mov r6,#40
d2: mov r7,#248
d1: djnz r7,$
d2: djnz r6,d2
d1: djnz r5,d1
ret
end
```

接线方法: 用一条 8PIN 的数据排线把 CPU 部份的 P1 口 (JP44) 连接到八路指示灯部份的 JP32
接线图以及运行瞬间的照片 (可以看到 8 路指示灯轮流点亮)



XL1000 运行照片



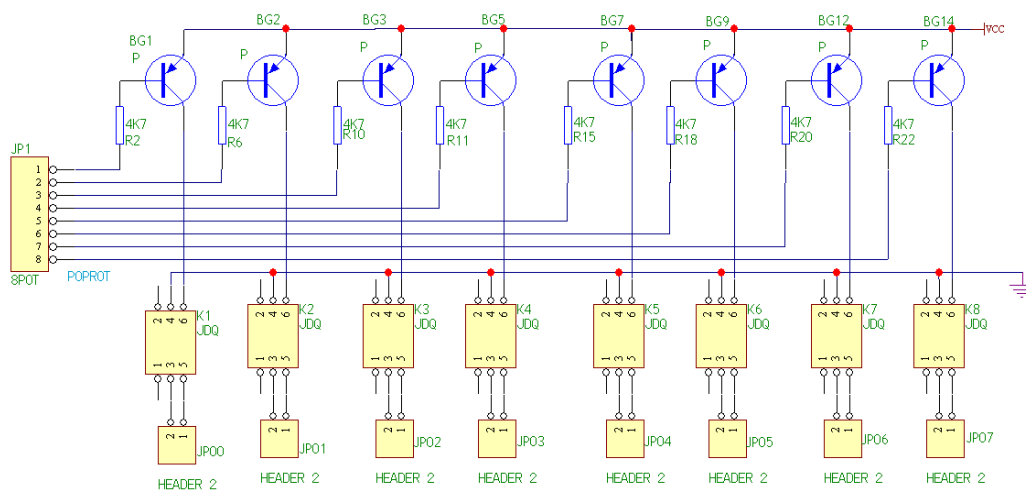
XL600 运行照片



实验 2: 用 XLISP 系列试验仪做一个 8 路彩灯控制器 (仅限 XL1000)

过节了，班上开晚会，想不想用单片机自己做一个彩灯控制器酷一把？我们用 XL1000 的 8 路继电器，编几个小花样的彩灯闪烁程序，把晚会装饰一下。可以去电工商店买 8 组小的彩灯，最好是不同的颜色，把他按照不同的方位分别布线。最后汇总到 XLISP 系列上，就可以用上面的继电器来控制了。当然，这个操作必须在老师的指导下进行。不要用 220v 的彩灯，最好是 12v 的小灯，以保证安全。切记！

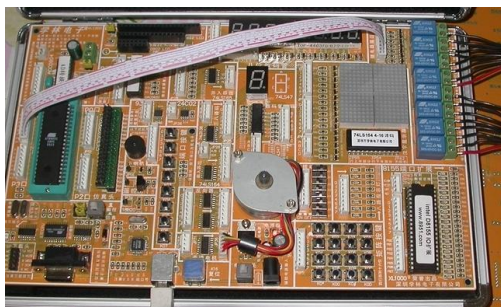
继电器部分由 8 只 PNP 型三级管 2N5401 驱动 8 只 5V 继电器。继电器触点由 JP00-JP07 个端子引出。在继电器没有动作时，端子的触点断开。在继电器动作时，触点闭合。把触点引出，即可控制外部的设备，如彩灯等等。需要注意的是控制电流不大于 1A，电压不大于 120V。触点部分如果控制高压，必须注意安全！



输入上例的 8 路跑马灯程序。



(仅限 x11000, x1600 是一路继电器) 用一条 8PIN 的数据排线把 CPU 部份的 P1 口 (JP44) 连接到八路继电器部份的 JP1, 既可以听到 8 只继电器轮流吸合的啪啪声。把端子全部接线引出，就可以控制外部的彩灯了。接线照片如下所示

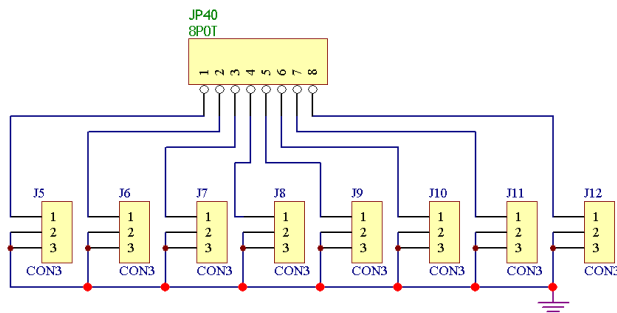


XL1000 运行照片

实验 3 8 路指示灯读出 8 路拨动开关的状态

8 路拨动开关一端接地，另外一端 8 路通过 JP40 引出。

相关原理图：



实验目的：通过设置不同的 8 路拨动开关的状态，使对应的 8 路指示灯亮灭。从而理解单片机中数和位的概念以及数据传递的概念。



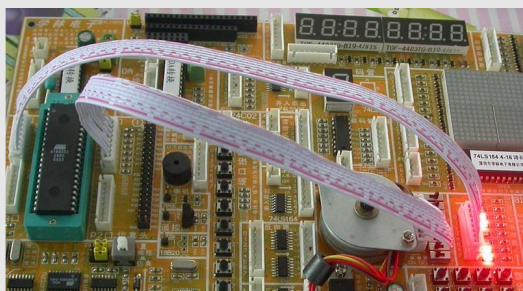
参考程序：

```
org 0000h ;开始
ajmp loop
org 0030h ;到 0030h 处避开 00-30 之间的敏感地址
loop: MOV A, P0; P0 口的状态（开关）读到 A
MOV P1, A ;把 A 的数据送到 P1 口（指示灯）
JMP LOOP ;重新开始
END
```

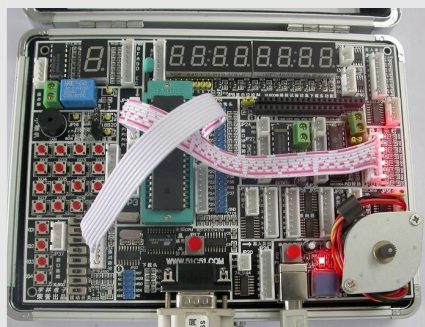


把八路拨动开关的 JP40 用一条 8PIN 的数据排线引入到 CPU 的 P0 口 JP51, 然后用一条 8PIN 的数据排线把 CPU 部份的 P1 口（JP44）连接到八路指示灯部份的 JP32。

运行结果：看下图照片，对应的开关打开，则对应的灯亮。



XL1000 运行照片

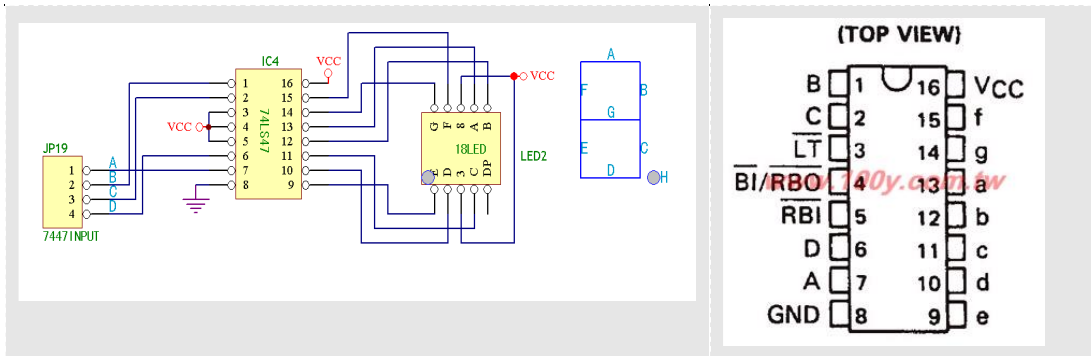


XL600 运行照片

实验 4：数码管静态扫描

74LS47 是一个 7 段数码管的驱动芯片，通过它解码，可以直接把数字转换为数码管的显示数字，从而简化了程序，节约了单片机的 I/O 开销。因此是一个非常好的芯片！但是由于目前从节约成本的角度考虑，此类芯片已较少用，大部份情况下都是用动态扫描数码管的形式来实现数码管显示。作为学习而言，我们还是很有必要了解此类 IC 的应用。

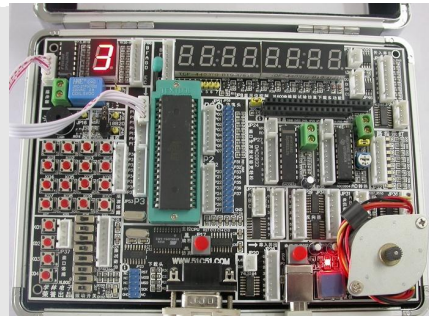
以下是 74ls47 的接线图：



接线方法：用一条 4PIN 的排线，把数码管译码部份的 JP19 接到 CPU 部份 P1 口的 JP44 的 P1.0, P1.1, P1.2, P1.3 四个端口。（即插入 P1 口的上半部分）



XL1000 运行照片



XL600 运行照片



程序说明：此程序轮流往 P1 口送数字 0, 1, 2, 3，数码管也轮流显示 0, 1, 2, 3

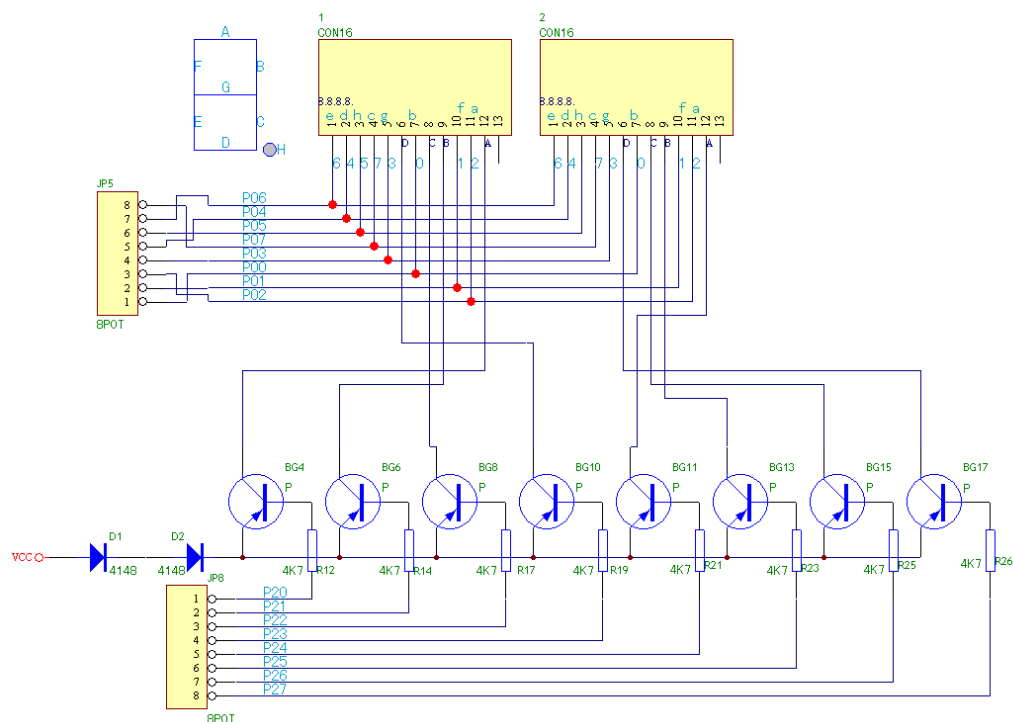
```
org 0000h ;开始
ajmp loop
org 0030h ;到 0030h 处避开 00-30 之间的敏感地址
LOOP:
MOV P1, #1 ;p1 口送数字 1
CALL DELAY ;延时
MOV P1, #2 ;p1 口送数字 2
CALL DELAY ;延时
MOV P1, #3
CALL DELAY
MOV P1, #4
```

```
CALL DELAY
JMP LOOP ;重新开始
DELAY: MOV R5, #50 ;延时子程序
D1: MOV R6, #40
D2: MOV R7, #248
DJNZ R7, $
DJNZ R6, D2
DJNZ R5, D1
RET
END
```

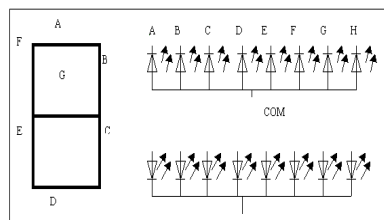


实验 5 数码管动态扫描显示 01234567

原理图：8 个数码管它的数据线并连接到 JP5， 位控制由 8 个 PNP 型三级管驱动后由 JP8 引出。



数码管是怎样来显示 1, 2, 3, 4 呢? 数码管实际上是由 7 个发光管组成 8 字形构成的, 加上小数点就是 8 个。我们分别把他命名为 A, B, C, D, E, F, G, H。



搞懂了这个原理, 我们如果要显示一个数字 2, 那么 A, B, G, E, D 这 5 个段的发光管亮就可以了。也就是把 B, E, H (小数点) 不亮, 其余全亮。根据硬件的接法我们编出以下程序。当然在此之前, 还必须指定哪一个数码管亮, 这里我们就指定最后一个 P2. 7。

LOOP:

CLR P2. 7; 选中最后的数码管

SETB P0. 7; B 段不亮

SETB P0. 5; 小数点不亮

SETB P0. 1; C 段不亮

CLR P0. 2; 其他都亮

CLR P0. 3

CLR P0. 4

CLR P0. 6

CLR P0. 0

JMP LOOP;跳转到开始重新进行

END

把这个程序编译后写入单片机，可以看到数码管的最后一位显示了一个数字 2。

也许你会说：显示 1 个 2 字就要 10 多行程序，太麻烦了。

显示数字 2 则是 C，F，H（小数点）不亮，同时由于接法为共阳接法，那么为 0（低电平）是亮

为 1（高电平）是灭。从高往低排列，（p0.7_p0.0）写成二进制为 01111110，把他转化为 16 进制则为 A2H。

我们可以根据硬件的接线把数码管显示数字编制成一个表格，以后直接调用就行了。

显示	p0.7 C	p0.6 E	p0.5 小 数点	p0.4 D	p0.3 G	p0.2 A	p0.1 F	p0.0 B	HEX 代码
0	0	0	1	0	1	0	0	0	28H
1	0	1	1	1	1	1	1	0	7EH
2	1	0	1	0	0	0	1	0	A2H
3	0	1	1	0	0	0	1	0	62H
4	0	1	1	1	0	1	0	0	74H
5	0	1	1	0	0	0	0	1	61H
6	0	0	1	0	0	0	0	1	21H
7	0	1	1	1	1	0	1	0	7AH
8	0	0	1	0	0	0	0	0	20H
9	0	1	1	0	0	0	0	0	60H

有了这个表格上面显示一个 2 的程序则可简化为

LOOP:

CLR P2.7;选中左边的数码管

MOV P0,#0A2H;送数字 2 的代码到 P0 口

JMP LOOP;跳转到开始重新进行

END

原理图中把所有数码管的 8 个笔划段 a-h 同名端连在一起，而每一个显示器的公共极 COM 是各自独立地受 I/O 线控制。CPU 向字段输出口送出字形码时，所有显示器接收到相同的字形码，由 8 个 PNP 的三极管，来控制这 8 位哪一位工作，例如上面的例子中我们选中的是 P2.7。就是最后的一位亮了。同样的如果要第一位亮，只需要把程序 CLR P2.7 改为 CLR P2.0 即可。

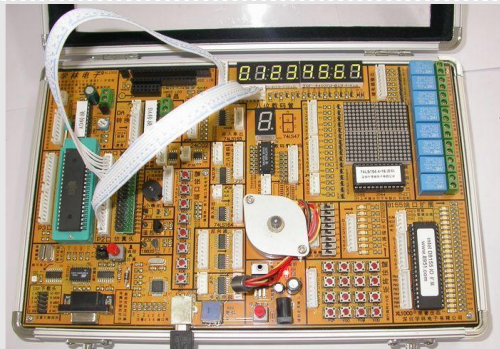
在这里就有了一个矛盾，所有数码管的 8 个笔划段 a-h 同名端连在一起，那么在一个屏幕上如何显示 0,1,2,3,4,5 这样不同的数字呢？的确，在这样的接法中，同一个瞬间所有的数码管显示都是相同的，不能显示不同的数字。在单片机里，首先显示一个数，然后关掉。然后显示第二个数，又关掉，那么将看到连续的数字显示，轮流点亮扫描过程中，每位显示器的点亮时间是极为短暂的（约 1ms），由于人的视觉暂留现象及发光二极管的余辉效应，尽管实际上各位显示器并非同时点亮，但只要扫描的速度足够快，给人的印象就是一组稳定的显示数据，不会有闪烁感。

例如数码管显示 01234567 这么 8 个数，在单片机中实际的工作流程如下：先打开 P2.0，送 0，然后关掉 P2.0，打开 P2.1 送 1，再关掉 P2.1，打开 P2.2 送 2，依次向下，由于速度足够快，那么我们将连续的看到 01234567 这 8 个数。

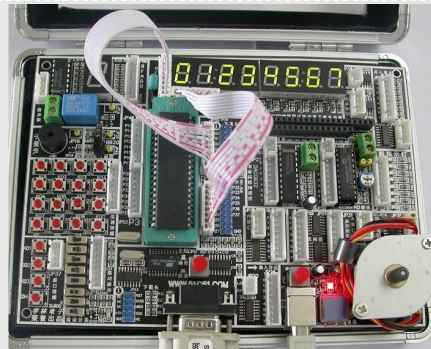


(数码管动态扫描显示 01234567 接线方法以及运行的显示照片)

- 1 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51.
- 2 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52.



XL1000 运行照片



XL600 运行照片



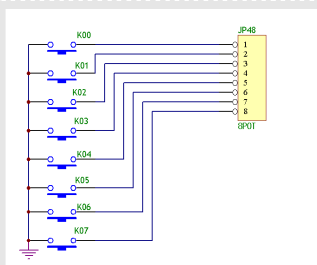
```

ORG 0000H
    AJMP MAIN
    ORG 0030H
MAIN:
    CLR P2.0           ;选中第一个数码管
    MOV P0,#28H        ;显示 0
    LCALL DELAY        ;调用延时
    MOV P0,#0FFH       ;关显示
    SETB P2.0
    CLR P2.1           ;选中第二个数码管
    MOV P0,#7EH        ;显示 1
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.1
    CLR P2.2           ;选中第三个数码管
    MOV P0,#0A2H       ;显示 2
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.2
    CLR P2.3           ;选中第四个数码管
    MOV P0,#62H        ;显示 3
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.3
    CLR P2.4           ;选中第五个数码管
    MOV P0,#74H        ;显示 4
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.4
    CLR P2.5           ;选中第六个数码管
    MOV P0,#61H        ;显示 5
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.5
    CLR P2.6           ;选中第七个数码管
    MOV P0,#21H        ;显示 6
    LCALL DELAY
    MOV P0,#0FFH
    SETB P2.6
    CLR P2.7           ;选中第八个数码管
    MOV P0,#7AH        ;显示 7
    LCALL DELAY
    SETB P2.7
    MOV P0,#0FFH
    AJMP MAIN          ;重新开始
DELAY:
    MOV R7,#2
D1:  MOV R6,#25
D2:  DJNZ R6,D2
    DJNZ R7,D1
    RET
    END
  
```

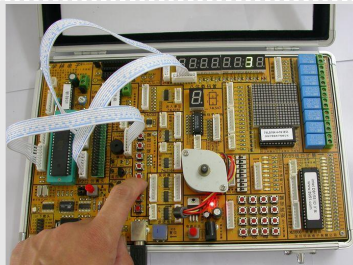


实验 6 端口按键判断技术（按键显示数字）

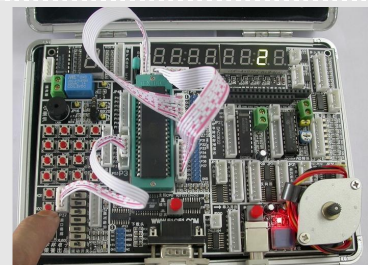
XL1000 此部份由 8 个轻触按键组成，一端接地，一端由 JP48 引出，当按下按键时，相应端口为低电平。（XL600 为 4 个按键，由 JP37 引出）



端口按键原理图



XL1000 有 8 个按键直接插 P3 口



XL600 有 4 个按键插 P3.2-P3.5 口。



接线方法：1 按键接到 P3 口。用一条 8PIN 数据排线，把按键部份的 JP48，接到 CPU 部份的 P3 口 JP53。（XL600 用 4PIN 排线一端插 P3 口 P3.2-P3.5 一端插 JP37）

2 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51。

3 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52。



任务：键盘显示数字，通过按键盘上的 K02, K03, K04, K05 四个按键，实现数码管显示 0, 1, 2, 3

org 0000h; (1)	ljmp 11 ;循环检测; (11)
ljmp start; (2)	12: mov p0, #28H ;显示'0'; (12)
org 0030h; (3)	ljmp 11; (13)
start: mov p2, #0ffh; (4)	13: mov p0, #7EH ;显示'1'; (14)
CLR P2.6 ;选中最右边的七段码。; (5)	ljmp 11; (15)
mov p3, #0ffh ;初始化, P3 口置高; (6)	14: mov p0, #0A2H ;显示'2'; (16)
11: jnb p3.2, 12 ; (7)	ljmp 11; (17)
jnb p3.3, 13 ; ; (8)	15: mov p0, #62H ;显示'3'; (18)
jnb p3.4, 14 ; ; (9)	ljmp 11; (19)
jnb p3.5, 15 ;检测按键; (10)	end; (20)

为了便于程序的讲解，我们把每一句分别编号，第 1, 2, 3 行是单片机的初始化，一般的程序都是这样开头的，作用是跳开 00-30h 的地址单元，这些单元是分配给特殊寄存器使用的。第 6 行把 p3 送 0ffh，作用是把这些端口 p3.0-p3.7 置高电平。

因为本例实验中 XLISP 系列的键盘接在 p3 口，只有当端口为高电平时，我们按下键盘，端口被强行变为 0。根据这个大家自行分析第 4 句的意思。第 5 行 clr p2.6，就是选中最右边第 2 个的数码管。板上一共有 8 个数码管，分别是 p2.0, p2.1, p2.2, p2.3, p2.4, p2.5, p2.6, p2.7

程序运行到这里，如果我们往数码管送数，那么数码管就应当亮了，这里插入了一段简单的键盘检测程序：我们来看第 7 行，jnb p3.2, 12，意思是如果 p3.2 为 0，那么执行 12。本实验的 p3.2 接了一个小开关到地，此时就有两种情况，如果键盘没有按下：程序继续向下跑第 8 行，继续检测 p3.3 有没有按下，直到第 11 行：ljmp 11 又跳转到第 7 行，继续检测。所以如果一直没有键盘按下，那么这个程序将一直在第 7, 8, 9, 10, 11 行反复循环。

如果某键盘已经按下，（假设为 p3.2）此时被强行拉低为 0，此时第 7 行的 jnb p3.2, 12 就会使程序跳转到 12（第 12 行）：mov p0, #28H，这里就把 28h 这个数送到 p0 口，数码管就显示“0”了。然后继续第 13 行 ljmp 11，程序又跳转到第 7 行，继续检测键盘。如果这个键不松开那么程序将反复执行 7, 12, 13，数码管就始终显示一个“0”剩下的几个键的程序流程与这个类似大家可自行分析。

实验 7 矩阵按键识别技术

矩阵按键部分由 16 个轻触按键按照 4 行 4 列排列，连接到 JP50 端口。将行线所接的单片机的 I/O 口作为输出端，而列线所接的 I/O 口则作为输入。这样，当按键没有按下时，所有的输出端都是高电平，代表无键按下。行线输出是低电平，一旦有键按下，则输入线就会被拉低，这样，通过读入输入线的状态就可得知是否有键按下了。

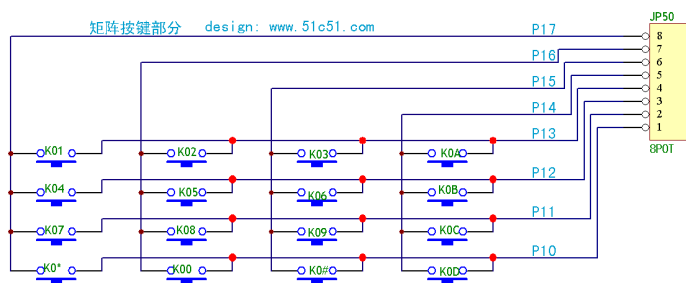
确定矩阵式键盘上何键被按下，介绍一种“行扫描法”。

行扫描法 行扫描法又称为逐行（或列）扫描查询法，是一种最常用的按键识别方法。

判断键盘中是否有键按下：将全部行线置低电平，然后检测列线的状态。只要有一列的电平为低，则表示键盘中有键被按下，而且闭合的键位于低电平线与 4 根行线相交的 4 个按键之中。若所有列线均为高电平，则键盘中无键按下。

判断闭合键所在的位置：在确认有键按下后，即可进入确定具体闭合键的过程。其方法是：依次将行线置为低电平，即在置某根行线为低电平时，其它线为高电平。在确定某根行线位置为低电平后，再逐行检测各列线的电平状态。若某列为低，则该列线与置为低电平的行线交叉处的按键就是闭合的按键。

下面给出一个具体的例子：



8031 单片机的 P1 口用作键盘 I/O 口，键盘的列线接到 P1 口的低 4 位，键盘的行线接到 P1 口的高 4 位。列线 P1.0-P1.3 设置为输入线，行线 P1.4-P1.7 设置为输出线。4 根行线和 4 根列线形成 16 个相交点。

1. 检测当前是否有键被按下。检测的方法是 P1.4-P1.7 输出全“0”，读取 P1.0-P1.3 的状态，若 P1.0-P1.3 为全“1”，则无键闭合，否则有键闭合。
2. 去除键抖动。当检测到有键按下后，延时一段时间再做下一步的检测判断。
3. 若有键被按下，应识别出是哪一个键闭合。方法是对键盘的行线进行扫描。P1.4-P1.7 按下述 4 种组合依次输出：

P1.7 1 1 1 0	P1.6 1 1 0 1	P1.5 1 0 1 1	P1.4 0 1 1 1
--------------	--------------	--------------	--------------

在每组行输出时读取 P1.0-P1.3，若全为“1”，则表示为“0”这一行没有键闭合，否则有键闭合。由此得到闭合键的行值和列值，然后可采用算法或查表法将闭合键的行值和列值转换成所定义的键值。

4. 为了保证键每闭合一次 CPU 仅作一次处理，必须去除键释放时的抖动。

实验目的：通过 XLISP 系列的 16 位矩阵按键，在数码管上分别显示 0—9, A, B, C, D, E, F。



接线方法：1 用一条 8PIN 数据排线，把矩阵按键部份的 JP50，接到 CPU 部份的 P1 口 JP44。

2 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51。

3 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52。

参考程序：

;本程序实现扫描按键显示功能。

;分别按 16 个键盘显示分别显示数字 123A456B789C*0#D

;键盘口 P1, 数码管显示第二位 p21, 数码管段位 p0 口



org 0000h

ajmp main

org 0030h

main:

```

    mov dptr,#tab      ;将表头放入 DPTR
    lcall key          ;调用键盘扫描程序
    movc a,@a+dptr     ;查表后将键值送入 ACC
    mov p0,a           ;将 Acc 值送入 P0 口
    CLR P2.1          ;开显示
    ljmp main          ;返回反复循环显示
KEY: LCALL KS          ;调用检测按键子程序
     JNZ K1            ;有键按下继续
     LCALL DELAY2      ;无键按调用延时去抖
     AJMP KEY          ;返回继续检测按键
K1:  LCALL DELAY2      ;有键按下延时去抖动
     LCALL KS          ;再调用检测按键程序
     JNZ K2            ;确认有按下进行下一步
     AJMP KEY          ;无键按下返回继续检测
K2:  MOV R2,#0EFH      ;将扫描值送入 R2 暂存
     MOV R4,#00H       ;将第一列值送入 R4 暂存
     K3: MOV P1,R2      ;将 R2 的值送入 P1 口
L6:  JB P1.0,L1        ;P1.0 等于 1 跳转到 L1
     MOV A,#00H        ;将第一行值送入 ACC
     AJMP LK           ;跳转到键值处理程序
L1:  JB P1.1,L2        ;P1.1 等于 1 跳转到 L2
     MOV A,#04H        ;将第二行的行值送入 ACC
     AJMP LK           ;跳转到键值理程序进行键值处理
L2:  JB P1.2,L3        ;P1.2 等于 1 跳转到 L3
     MOV A,#08H        ;将第三行的行值送入 ACC
     AJMP LK           ;跳转到键值处理程序

```

```

L3:  JB P1.3,NEXT      ;P1.3 等于 1 跳转到 NEXT 处
     MOV A,#0cH        ;将第四行的行值送入 ACC
LK:  ADD A,R4           ;行值与列值相加后的键值送入 A
     PUSH ACC          ;将 A 中的值送入堆栈暂存
K4:  LCALL DELAY2      ;调用延时去抖动程序
     LCALL KS          ;调用按键检测程序
     JNZ K4            ;按键没有松开继续返回检测
     POP ACC           ;将堆栈的值送入 ACC
     RET

```

NEXT:

```

     INC R4            ;将列值加一
     MOV A,R2          ;将 R2 的值送入 A
     JNB ACC.7,KEY     ;扫描完至 KEY 处进行下一扫描
     RL A              ;扫描未完将 A 中的值右移一位进行下一
                       ;列的扫描
     MOV R2,A          ;将 ACC 的值送入 R2 暂存
     AJMP K3           ;跳转到 K3 继续
KS:  MOV P1,#0FH       ;将 P1 口高四位位置 0 低四位值 1
     MOV A,P1          ;读 P1 口
     XRL A,#0FH        ;将 A 中的值与 A 中的值相异或
     RET              ;子程序返回

```

DELAY2: ;40ms 延时去抖动子程序

MOV R5,#08H

L7: MOV R6,#0FAH

L8: DJNZ R6,L8

DJNZ R5,L7

RET

tab:

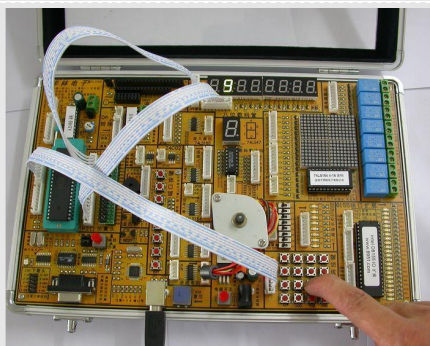
db 28h, 34h, 28h, 34h, 0a9h, 60h, 20h, 7ah,

20h, 21h, 61h, 74h, 30h, 62h, 0a2h, 7eh

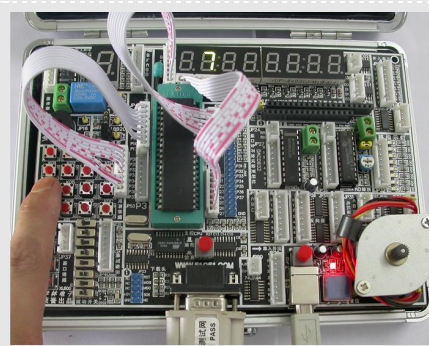
;0h0hc9878654a321 轮流显示键盘因为无法表达

*# 就用 H 表示, B 用 8 表示

end

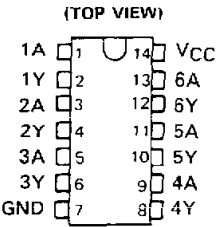


XL1000 运行照片



XL600 运行照片

实验 8 74LS14 反向器实验



74LS14 是一个 6 反向器， 引脚定义如下图： A 端为输入端， Y 端为输出端， 一片芯片一共 6 路， 即

1, 3, 5, 9, 11, 13 为输入端， 2, 4, 6, 8, 10, 12 为输出端， 输出结果与输入结果反向。即如果输入端为高电平， 那么输出为低电平。 如果输入低电平， 输出为高电平。

在本套件中， 一共使用了 2 片 74LS14 芯片， 组成 8 路反向器。 反向器是重要的数字电路之一， 下面我带领大家来学习反向器的应用。

实验目的： 先做一个 8 路加法器的实验， 然后通过反向器后， 观察 LED 的发光管的状态变化。

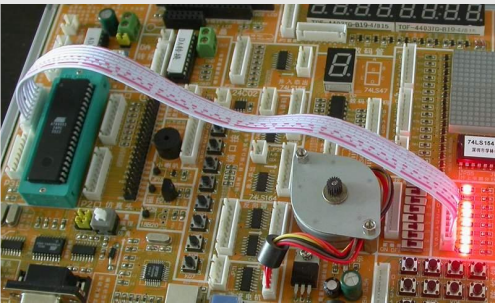


```
org 0000h
ajmp main
org 0030h
main: mov a, #0
loop: mov p1, a ;输出到 P1 口
      call delay ;延时。
      inc a ;每次加 1
      jmp loop
```

```
delay: mov r5, #50 ;延时。
d1: mov r6, #40
d2: mov r7, #248
    djnz r7, $
    djnz r6, d2
    djnz r5, d1
    ret
end
```

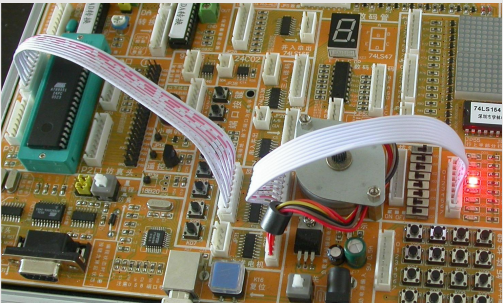


接线方法：



1： 直接连接： 用一条 8PIN 的数据排线把 CPU 部份的 P1 口（JP44）连接到八路指示灯部份的 JP32

此时可以观察到指示灯按照二进制顺序逐次熄灭。



2 ： 通过反向器连接：

用一条 8PIN 的数据排线把 CPU 部份的 P1 口（JP44）接到反向器部分的输入端 JP29

用一条 8PIN 的数据排线把反向器的输出端 JP34 接到八路指示灯部份的 JP32

此时可以观察到指示灯按照二进制顺序逐次点亮。

与原来的结果正好相反。



实验 9：74LS138 部分 38 译码器部分实验

38 译码器是通过 3 条线来达到控制 8 条线的状态，就是通过 3 条控制线不同的高低电平组合，一共可以组合出 8 种状态，即 2 的 3 次方。在电路中，主要起到扩展 I/O 资源的作用。

74LS138 引脚定义

74LS138 逻辑表

Inputs				Outputs								
Enable		Select										
G1	G2 (Note 1)	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

74138 的电路原理图

XL1000 运行照片

XL600 运行照片

以下实验通过 P1.0, P1.1, P1.2 三条数据线通过 74LS138 译码作一个 8 路跑马灯实验。(注意我们用的一条 4PIN 的数据线作他的输入端，在 JP22 的 4 脚为悬空状态。实际上 3 条数据线)

接线方法：

- 1 用 4PIN 数据排线插入 CPU 部分的 JP44 的 P1.0-P1.3 口。另一端插入 74LS138 部分的 JP22 作为数据输入。
- 2 用一条 8PIN 数据排线，连接 78LS138 的输出口 JP25 到 8 路指示灯的 JP32，运行可看到跑马灯。

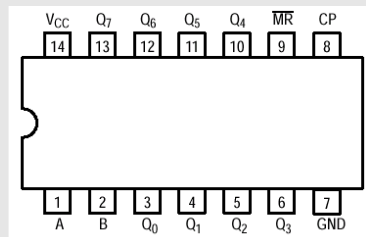
```
org 0000h
ajmp main
org 0030h;
;定义 P1.0, P1.1, P1.2 分别为 AA, BB, CCAA EQU P1.0
BB EQU P1.1
CC EQU P1.2
MAIN:
CLR CC      ;状态 1 A, B, C 为 000
CLR BB
CLR AA
CALL DELAY ;调用延时
CLR CC      ;状态 2 A, B, C 为 100
CLR BB
SETB AA
CALL DELAY ;调用延时
CLR CC      ;状态 2 A, B, C 为 010
SETB BB
CLR AA
CALL DELAY
CLR CC
SETB BB
SETB AA
CALL DELAY
SETB CC
CLR BB
CALL DELAY
JMP MAIN ;重新开始
DELAY: ;延时子程序
MOV R5, #255
D3:MOV R2, #255
D4: DJNZ R2, D4
DJNZ R5, D3
RET
END
```



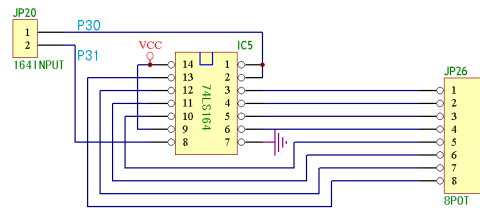
实验 10: 74LS164 串入并出实验

在单片机系统中，如果并行口的 IO 资源不够，而串行口又没有其他的作用，那么我们可以用 74LS164 来扩展并行 IO 口，节约单片机资源。74LS164 是一个串行输入并行输出的移位寄存器。并带有清除端。其中：Q0—Q7 并行输出端。A,B 串行输入端。MR 清除端，为 0 时，输出清零。CP 时钟输入端。

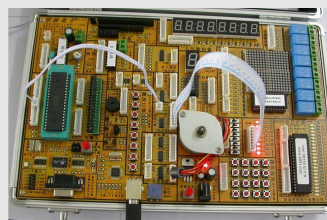
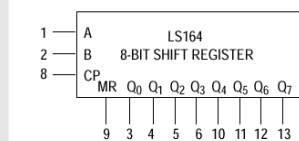
74LS164 引脚定义



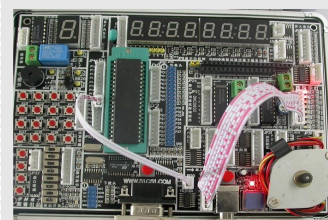
74LS164 电路原理:



74LS164 逻辑表



XL1000 运行照片



XL600 运行照片



接线方法：1 用一个 2PIN 数据排线一端插入 CPU 部分 JP53（P3 口）的 P3.0, P3.1。

另外一端插入 74LS164 部分的输入端 JP20。

2 用一根 8PIN 的数据排线，一端插入 74LS164 部分的输出端 JP26，另一端插入 8 路指示灯的 JP32。



参考程序：我们通过以下实验利用 74LS164 串行口实现 8 路跑马灯程序。

```

org 00h
mov scon, #00000000b ;设定 UART 方式 MODE0
start: mov dptr, #table ;数据指针指到 TABLE
loop: clr a ;清除 ACC
movc a, @a+dptr ;到 TABLE 取数据
cjne a, #03, a1 ;到结束码 03 了么？不是到 A1
jmp start ;是则重新开始
a1: cpl a ;取到的数据反向
mov 30h, a ;A 存入 30H
mov sbuf, 30h ;30H 的值存入 SBUF
loop1: jbc ti, loop2 ;监测 TI=1? 是则跳到 LOOP2
jmp loop1 ;否则继续监测
loop2: call delay ;延时
inc dptr ;数据指针加 1
jmp loop
DELAY: ; 延时程序

MOV R5, #255
D3: MOV R2, #255
D4: DJNZ R2, D4
DJNZ R5, D3
RET
table: db 01h, 02h, 04h, 08h ;左移
db 10h, 20h, 40h, 80h
db 01h, 02h, 04h, 08h
db 10h, 20h, 40h, 80h
db 80h, 40h, 20h, 10h ;右移
db 08h, 04h, 02h, 01h
db 80h, 40h, 20h, 10h
db 08h, 04h, 02h, 01h
db 00h, 0ffh, 00h, 0ffh ;闪烁
db 03h
END

```


实验 11 74LS165 并入串出实验

74LS165 是并行输入，串行输出移位寄存器。80C51 单片机内部的串行口在方式 0 工作状态下,使用移位寄存器芯片可以扩展一个或多个 8 位并行 I/O 口。

其引脚如图所示。

A, B, C, D, E, F, G, H 并行输入端。

QH 串行输出端。

CLOCK: 时钟输入端。

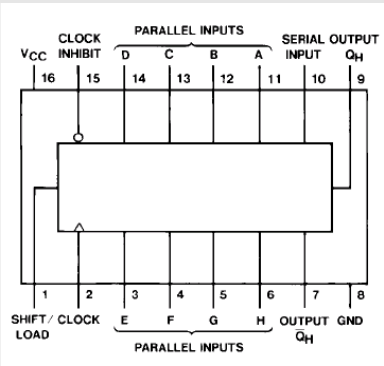
CLOCK INHIBIT: 时钟禁止端。当时钟禁止端 CLK2 为低电平时，充许时钟输入。

SHIFT/LOAD: 移位与置位控制端。

SER: 扩展多个 74LS165 的首尾连接端。

本练习 TXD（P3.1）作为移位脉冲输出端与 74LS165 的移位脉冲输入端 CLOCK 相连；RXD（P3.0）作为串行输入端与 74LS165 的串行输出端 QH 相连；P3.2 用来控制 74165 的移位与置入。

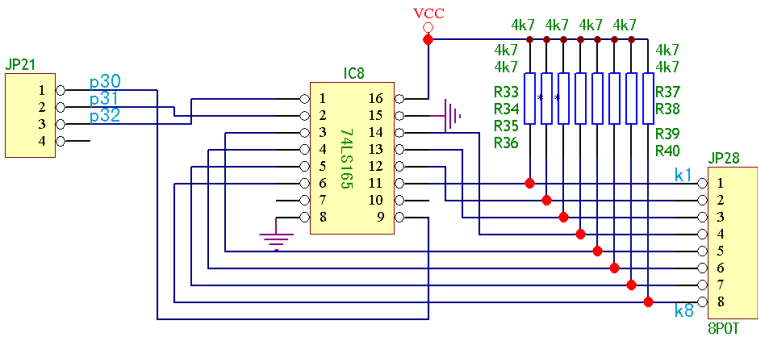
74LS165 引脚定义



74LS165 逻辑表

Inputs					Internal Outputs		Output
Shift/Load	Clock	Clock Inhibit	Serial	Parallel A...H	Q _A	Q _B	Q _H
L	X	X	X	a...h	Q _{A0}	Q _{B0}	Q _{H0}
H	L	L	X	X	H	Q _{An}	Q _{Gn}
H	L	↑	H	X	L	Q _{An}	Q _{Gn}
H	L	↑	L	X	Q _{A0}	Q _{B0}	Q _{H0}
H	H	X	X	X	Q _{A0}	Q _{B0}	Q _{H0}

74LS165 电路原理：

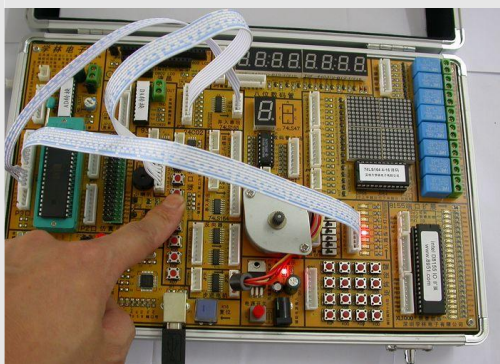


本练习利用串行口扩展了 1 片 74LS165，从而实现了 8 个按键的输入，在单片机的 P1 口连接了 8 个 LED 发光二极管，程序中实现了用按键控制发光二极管的亮与灭。流程为 8 路开关-74165-3PIN 数据线-8 路指示灯。由此可以看出通过 165 传输，只用了 3 条数据线，就实现了 8 个开关控制 8 个灯的目的。从而节约了控制的连线。

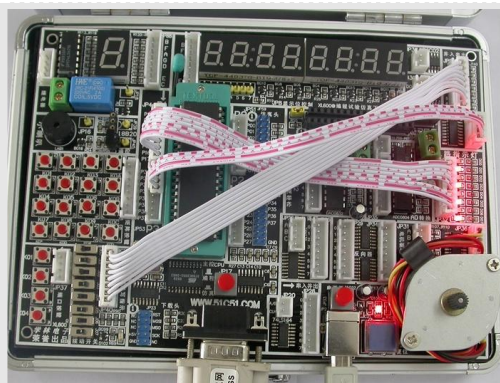


接线方法:

- 1 用一个 4PIN 数据排线一端插入 CPU 部分 JP53 (P3 口) 的 P3.0, P3.1, P3.2, P3.3。另外一端插入 74LS165 部分的输入端 JP21。
- 2 用一根 8PIN 的数据排线, 一端插入 74LS165 部分的输出端 JP28, 另一端插入 8 路按键开关的 JP48。
- 3 用一条 8PIN 的数据排线把 CPU 部份的 P1 口(JP44)连接到八路指示灯部份的 JP32。运行后分别按 k00-k07, 可以看到 d00-d07 八个灯受控制。



XL1000 运行照片



XL600 运行照片



74LS165 并入串出实验参考程序:

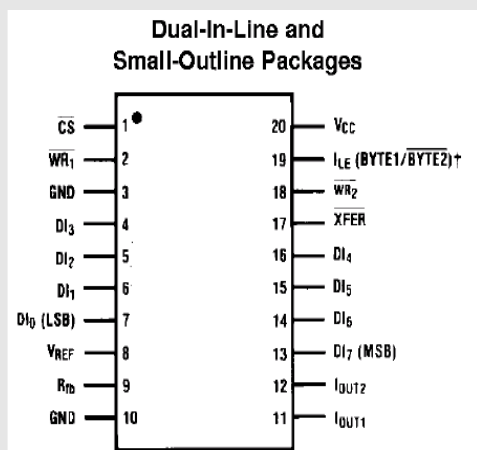
```
org 00h
loop : mov scon, #00010000b ; 设定 MODE=0
      clr p3.2                ; P3.2=0 载入数据
      call delay1             ; 延时
      setb p3.2;P3.2=1        ; 数据输出
      clr ri                  ; RI=0
      loop1:jbc ri, loop2     ; RI=1?是则到 LOOP2
      jmp loop1               ; 否则继续监测
      loop2:mov a, sbuf        ; 将 SUBF 载入 ACC
      mov p1, a                ; 输出到 P1
      jmp loop                 ; 重新开始
```

```
delay:  mov r7, #00h          ; 延时子程序
tt:     mov r6, #00h
      djnz r6, $
      djnz r7, tt
      ret
delay1:                                ; 短延时子程序
      mov r7, #02
      djnz r7, $
      ret
end
```


实验 12 DA 转换 dac0832 的原理与应用

DAC0832 是 8 位分辨率 D/A 转换集成芯片, 与处理器完全兼容, 其价格低廉, 接口简单, 转换控制容易等优点, 在单片机应用系统中得到了广泛的应用。

(1) DAC0832 的引脚及功能



DI0~DI7: 数据输入线, TLL 电平。

ILE: 数据锁存允许控制信号输入线, 高电平有效。

CS: 片选信号输入线, 低电平有效。

WR1: 为输入寄存器的写选通信号。

XFER: 数据传送控制信号输入线, 低电平有效。

WR2: 为 DAC 寄存器写选通输入线。

Iout1: 电流输出线。当输入全为 1 时 Iout1 最大。

Iout2: 电流输出线。其值与 Iout1 之和为一常数。

Rfb:反馈信号输入线,芯片内部有反馈电阻.

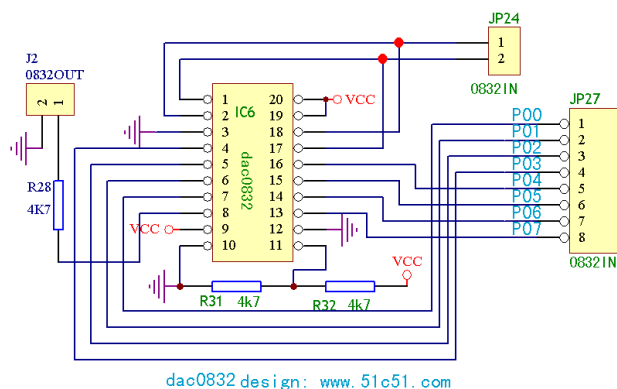
Vcc:电源输入线 (+5v~+15v)

Vref:基准电压输入线 $(-10\text{V}\sim+10\text{V})$

AGND:模拟地,模拟信号和基准电源的参考地.

DGND:数字地,两种地线在基准电源处共地比较好.

DAC0832 的实验原理图:



参考程序：用 DAC0832 产生梯形波。

```

ORG 0000h

MAIN:  mov a,#0
MOV DPTR,#07FFFFH
      MOV R1,#0ah

LP:    MOVX @DPTR,A ;选通地址
ACALL DELAY ;延时
djnz r1,next
sjmp main

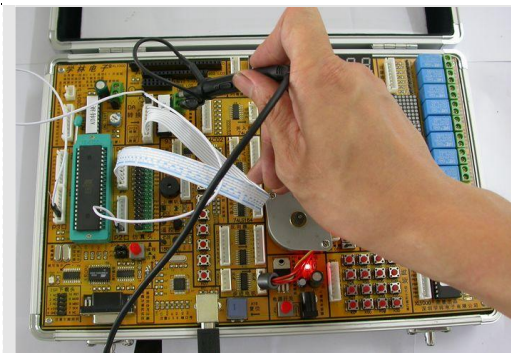
```

```
next :
ADD A, #10 ;每次以 19H 的梯度增加
sjmp lp
DELAY:      ;延时子程序
D2: MOV 31H, #0FFH
D1: DJNZ 31H, D1
RET
END
```

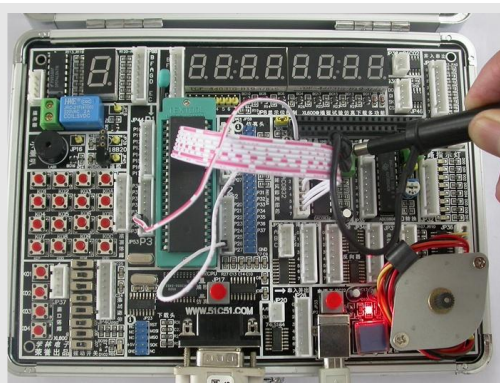


接线方法:

- 1 用一个 1PIN 数据线一端插入 CPU 部分 JP53 (P3 口) 的 P3.6, 另外一端插入 DAC0832 部分的输入端 JP24 的 WR 端。
- 2 用一个 1PIN 数据线一端插入 CPU 部分 JP52 (P2 口) 的 P2.7, 另外一端插入 DAC0832 部分的输入端 JP24 的 CS 端。
- 3 用一根 8PIN 的数据排线, 一端插入 DAC0832 部分的数据输入端 JP27, 另一端插入 CPU 部分 JP51 (P0 口)
- 4 用一台示波器, 在 J2 口观察 DA 转换后的输出信号。 建议调整到 Y 轴 0.2V/格 X 轴 1MS/格.



XL1000 运行照片



XL600 运行照片

示波器输出波形照片:



(输出矩形波正旋波的程序和照片请参考本站论坛:

<http://www.8951.com/bbs/dispbbs.asp?boardID=19&ID=10874&page=1>



实验 13 模拟/数字转换器 ADC0804

所谓 A/D 转换器就是模拟/数字转换器（Analog to Digital Converter 简称 ADC）是将输入的模拟信号转换成为数字信号。

ADC0804 的规格及引脚图

试验原理图:

- /CS: 芯片选择信号。

VREF: 辅助参考电压。

/RD: 外部读取转换结果的控制脚输出信号。

/INTR: 中断请求信号输出，低电平动

CLK IN, CLK R: 时钟输入或接振荡元件（R，C），频率约限制在 100KHz~1460KHz

VCC: 电源供应以及作为电路的参考电压。
- /WR: 用来启动转换的控制当/WR 自 HI 变为 LO 时，转换器被清除；当/WR 回到 HI 时，转换正式启动；

DB0~DB7: 8 位数字输出。

AGND, DGND: 模拟信号以及数字信号接地。

VIN (+), VIN (-): 差动模拟电压输入。输入单端正电压时，VIN (-) 接地；

ADC0804 电压输入与数字输出关系如表所示

十六进制	二进制码	与满刻度的比率		相对电应值 VREF=2.560 伏	
		高四位字节	低四位字节	高四位字节电压	低四位字节电压
F	1111	15/16	15/256	4.800	0.300
E	1110	14/16	14/256	4.480	0.280
D	1101	13/16	13/256	4.160	0.260
C	1100	12/16	12/256	3.840	0.240
B	1011	11/16	11/256	3.520	0.220
A	1010	10/16	10/256	3.200	0.200
9	1001	9/16	9/256	2.880	0.180
8	1000	8/16	8/256	2.560	0.160
7	0111	7/16	7/256	2.240	0.140
6	0110	6/16	6/256	1.920	0.120
5	0101	5/16	5/256	1.600	0.100
4	0100	4/16	4/256	1.280	0.080
3	0011	3/16	3/256	0.960	0.060
2	0010	2/16	2/256	0.640	0.040
1	0001	1/16	1/256	0.320	0.020
0	0000			0	0

; ADC0804 参考程序:

```
//定义 ADC 的连接端口
ad_cs equ P3.6
ad_wr equ P2.0
ad_rd equ P3.7
ad_input_port equ p1

org 0000h
ajmp main
org 0030h

main:
    lcall adc_demo
    ajmp main
;/// 启动 AD 转换
Adc_Start:
    clr ad_cs
    nop
    clr ad_wr
    nop
    setb ad_wr
    nop
    setb ad_cs
    nop
    ret

Adc_Read: ; // 读 AD 转换
    mov ad_input_port, #0ffh
    clr ad_cs
    nop
    clr ad_rd
```

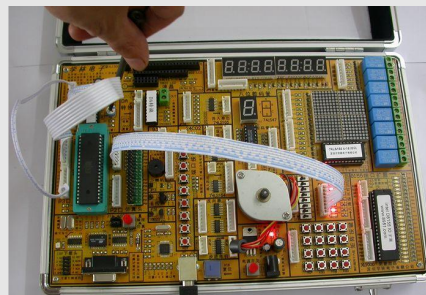
```
    nop
    nop
    mov a, AD_INPUT_PORT
    nop
    setb ad_rd
    nop
    setb ad_cs
    ret
;/// AD 转换读取延时程序，显示读到的数值
Adc_Demo:
    lcall Adc_Start
    lcall delay1ms
    lcall adc_read
    clr p2.0
    clr p2.3
    mov p0, a
    ret
delay1ms:
    mov r7, #10
ttl:
    mov r6, #50
    djnz r6, $ ;2us
    djnz r7, ttl
    ret
end
```



接线方法:

- 1 用一个 2PIN 数据线插入 CPU 部分 JP53 (P3 口) 的 P3.6, P3.7 另一端插入 ADC0804 部分的输入端 J3.
- 2 用一根 8PIN 的数据排线，一端插入 ADC0804 部分的数据输入端 JP35, 另一端插入 CPU 部分 JP44 (P1 口)
- 3 用一根 8PIN 的数据排线，一端插入八路指示灯部分的 JP32, 另一端插入 CPU 部分 JP51 (P0 口)

运行照片，用一个小螺丝刀调整 AD 转换部分的 ADJ3 可以看到 P0 口引出的 8 个 LED 逐次变化。



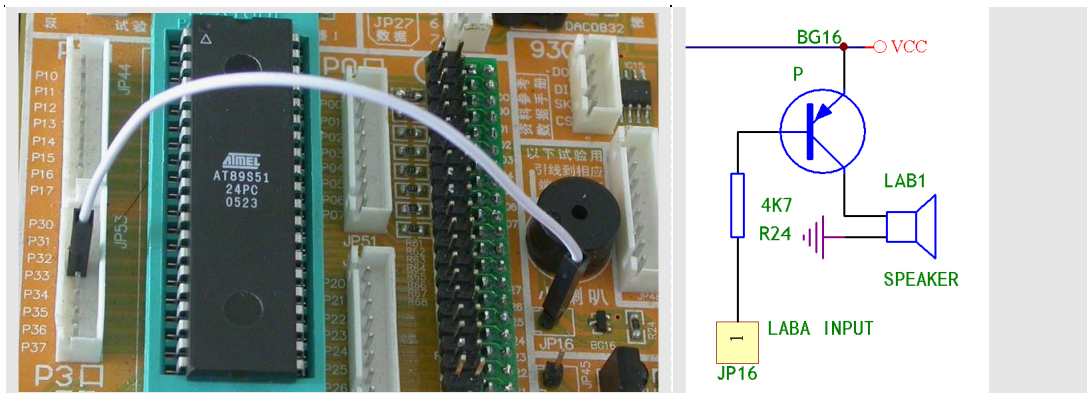


实验 14 小喇叭警报器试验

声音是由震动所产生的，一定频率的震动就产生了一定频率的声音。这个实验是喇叭里发出滴答一长一短的报警声音，送出的端口是 p3.3 输出 1khz, 2khz 变频信号报警，每一秒交换一次。



接线方法：用一根 1PIN 数据线一端插入 CPU 部分 JP53（P3 口）的 P3.3 另外一端插入小喇叭部分的输入端 JP16。



警报器声音的参考程序：

```
dlv: mov r2, #08h; 1khz 持续时间
dlv1: mov r3, #0fah
dlv2: cpl p3.3; 输出 1khz 方波
      lcall d5ms; 调用延时程序 1
      djnz r3, dlv2; 持续 1 秒
      djnz r2, dlv1
      mov r2, #10h; 2khz 持续时间
dlv3: mov r3, #0fah
dlv4: cpl p3.3; 输出 2khz 方波
      lcall d25ms; 调用延时程序 2
      djnz r3, dlv4
      djnz r2, dlv3
      sjmp dlv; 反复循环
d5ms: mov r7, #0ffh; 延时子程序 1
loop: nop
      nop
      djnz r7, loop
      ret
d25ms: mov r6, #0ffh; 延时子程序 2
lin: djnz r6, lin
      ret
end
```

实验 15 红外线遥控试验

XLISP 系列试验仪配有一个标准的 32 位 HT6121 编码红外遥控器。通用红外遥控系统由发射和接收两大部分组成，应用编/解码专用集成电路芯片来进行控制操作，如图 1 所示。发射部分包括键盘矩阵、编码调制、LED 红外发送器；接收部分包括光、电转换放大器、解调、解码电路。

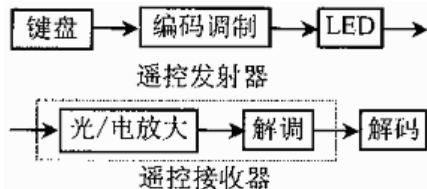


图 1 红外遥控系统框图

当发射器按键按下后，即有遥控码发出，所按的键不同遥控编码也不同。这种遥控码具有以下特征：采用脉宽调制的串行码，以脉宽为 0.565ms、间隔 0.56ms、周期为 1.125ms 的组合表示二进制的“0”；以脉宽为 0.565ms、间隔 1.685ms、周期为 2.25ms 的组合表示二进制的“1”，其波形如图 2 所示

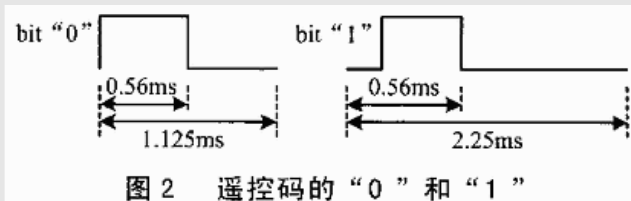
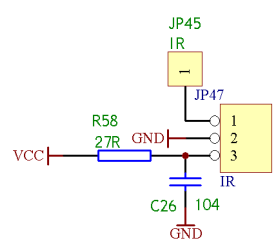
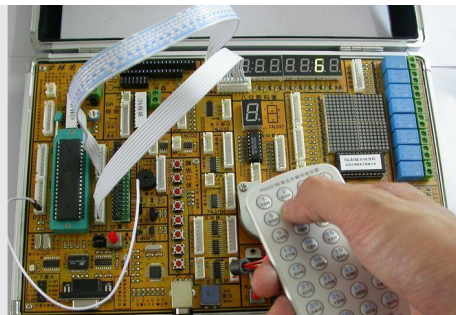


图 2 遥控码的“0”和“1”



遥控原理图

解码的关键是如何识别“0”和“1”，从位的定义我们可以发现“0”、“1”均以 0.56ms 的低电平开始，不同的是高电平的宽度不同，“0”为 0.56ms，“1”为 1.68ms，所以必须根据高电平的宽度区别“0”和“1”。如果从 0.56ms 低电平过后，开始延时，0.56ms 以后，若读到的电平为低，说明该位为“0”，反之则为“1”，为了可靠起见，延时必须比 0.56ms 长些，但又不能超过 1.12ms，否则如果该位为“0”，读到的已是下一位的高电平，因此取 $(1.12\text{ms} + 0.56\text{ms}) / 2 = 0.84\text{ms}$ 最为可靠，一般取 0.84ms 左右均可。



XL1000 运行照片



XL600 运行照片

以下试验是按遥控器的 1—16 键，数码管分别对应显示 1—9，0, A, B, C, D, E, F。



接线方法：

- 1 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51。
- 2 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52。
- 3 用一根 1PIN 数据线一端插入 CPU 部分 JP53（P3 口）的 P3.7 另外一端插入红外遥控部分的输出端 JP45。

org 00h ORG 30H MOV P1, #0FFH MOV P2, #0FFH MOV P3, #0FFH START: JB P3.7, \$;等待遥控信号出现 SB: MOV R4, #8 ;8 毫秒为高电平错误 SBA: MOV R5, #250 SBB: JB P3.7, SXB1 DJNZ R5, SBB DJNZ R4, SBA MOV R4, #2 JMP SBC SXB1: MOV R5, #5 SXB2: ;去掉 20US 的尖峰干扰信号 JNB P3.7, SBB DJNZ R5, SXB2 JMP START SBC: MOV R5, #250 SB1: JB P3.7, SB2 ;2MS 内不为高电平错误 DJNZ R5, SB1 DJNZ R4, SBC JMP START SB2: ;去掉 20US 的尖峰干扰信号 MOV R5, #5 SB2_A: JNB P3.7, SB1 DJNZ R5, SB2_A MOV R4, #3 SB2_1: MOV R5, #250 SB3: ;监测 4.5MS 高电平, 如 3MS 内出现低电平错误 JNB P3.7, SXC DJNZ R5, SB3 DJNZ R4, SB2_1 MOV R4, #2 JMP SB3_1	SXC: ;去掉 20US 的尖峰干扰信号 MOV R5, #5 SXC1: JB P3.7, SB3 DJNZ R5, SXC1 JMP START SB3_1: ;监测 4.5MS 高电平, 如 5MS 内不为低错误 MOV R5, #250 SB3_2: JNB P3.7, SB4 DJNZ R5, SB3_2 DJNZ R4, SB3_1 JMP START SB4: ;去掉 20US 的尖峰干扰信号 MOV R5, #5 SB4_1: JB P3.7, SB3_2 DJNZ R5, SB4_1 MOV R1, #1AH ;设定 1AH 为起始 RAM 区 MOV R2, #4 PP: MOV R3, #8 JJJJ: MOV R5, #250 JJJJ2: ;1MS 内不为低电平错误 JB P3.7, JJJJ3 DJNZ R5, JJJJ2 JMP START JJJJ3: LCALL YS1 ;高电平开始后用 882 微秒的 时间尺去判断信号此时的高低电平状态 MOV C, P3.7 ;将 P3.7 状态 0 或 1 存入 C 中 JNC UUU ;如果为 0 就跳转到 UUU MOV R5, #250 JJJJ4: JNB P3.7, UUU NOP DJNZ R5, JJJJ4 JMP START UUU: MOV A, @R1 ;将 R1 中地址的给 A RRC A ;将 C 中的值移入 A 中的最低位 MOV @R1, A ;将 A 中的数暂存在 R1 中 DJNZ R3, JJJJ ;接收地址码的高 8 位 INC R1 ;对 R1 加 1, 换成下一个 RAM DJNZ R2, PP ;
--	---

;以下对代码是否正确和定义进行识别

MOV A, 1AH;比较高 8 位地址码

XRL A, #00000000B ;判断 1AH 的值是否等于 00000000, 相等的话 A 为 0

JNZ EXIT ;如果不等解码失败退出

MOV A, 1BH;比较低 8 位地址

XRL A, #11111111B ;再判高 8 位地址是否正确

JNZ EXIT ;如果不相等说明解码失败退出

LCALL YS3

MOV A, 1CH;比较数据码和数据反码是否正确?

CPL A

XRL A, 1DH ;将 1CH 的值取反后和 1DH 比较不同则无效丢弃, 核对数据是否准确

JNZ EXIT ;如果不相等说明解码失败退出

LCALL YS3

CLR P2.6 ;选中数码管

CLR P3.3 ;解码成功喇叭响?

AJMP BIJIA0

;判断在 118 毫秒内是否有连发码

AA: MOV R1, #25

XX: ACALL YS2

JNB P3.7, HH ;跳转到 HH

DJNZ R1, XX

EXIT: ;对所有端口清零

AJMP START

;连发码判断程序段-----

HH: MOV R6, #4

S: ACALL YS1;调用 882 微秒延时子程序

JB P3.7, EXIT ;延时 882 微秒后判断 P3.7 脚是否出现高电平如果有就退出解码程序

DJNZ R6, S ;重复 4 次, 目的是确认

JNB P3.7, \$;等待高电?

LCALL YS3

AJMP AA

BIJIA0: MOV A, 1CH;按键数值判断执

mov p1, a

CJNE A, #10h, TT1

MOV P0, #7EH ;数码管显示 1

TT1: CJNE A, #03h, T2

MOV P0, #0A2H ;数码管显示 2

T2: CJNE A, #01h, T3

MOV P0, #62H ;数码管显示 3

T3: CJNE A, #06h, T4

MOV P0, #74H ;数码管显示 4

T4: CJNE A, #09h, T5

MOV P0, #61H ;数码管显示 5

T5: CJNE A, #1dh, T6

MOV P0, #21H ;数码管显示 6

T6: CJNE A, #1fh, T7

MOV P0, #7AH ;数码管显示 7

T7: CJNE A, #0dh, T8

MOV P0, #20H ;数码管显示 8

T8: CJNE A, #19h, t9

MOV P0, #60H ;数码管显示 9

T9: CJNE A, #1bh, t10

MOV P0, #28H ;数码管显示 0

T10: CJNE A, #11h, t11

MOV P0, #30H ;数码管显示 A

T11: CJNE A, #15h, t12

MOV P0, #25H ;数码管显示 b

T12: CJNE A, #17h, t13

MOV P0, #0a9H ;数码管显示 C

T13: CJNE A, #12h, t14

MOV P0, #26H ;数码管显示 d

T14: CJNE A, #16h, t15

MOV P0, #0a1H ;数码管显示 E

T15: CJNE A, #4ch, t16

MOV P0, #0b1H ;数码管显示 f

t16: AJMP AA

YS1: MOV R4, #20 ;延时子程序 1, 882 微秒

D1: MOV R5, #20

DJNZ R5, \$

DJNZ R4, D1

RET

YS2: MOV R4, #10 ;延时子程序 2, 4740 微秒

D2: MOV R5, #235

DJNZ R5, \$

DJNZ R4, D2

RET

YS3: MOV R4, #2;延时程序 3, 1000 微秒

D3: MOV R5, #248

DJNZ R5, \$

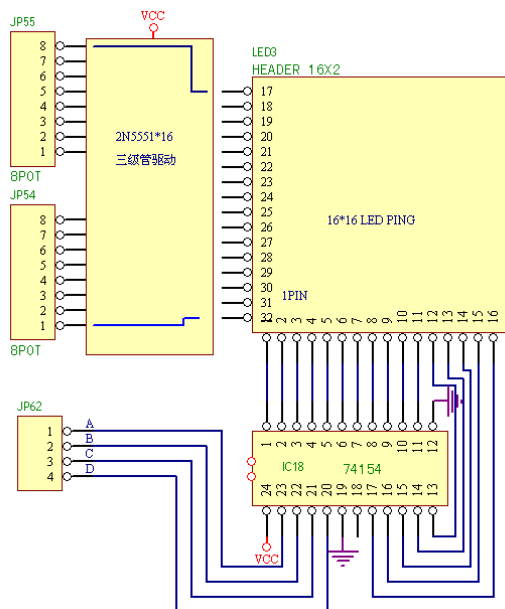
DJNZ R4, D3

RET

END

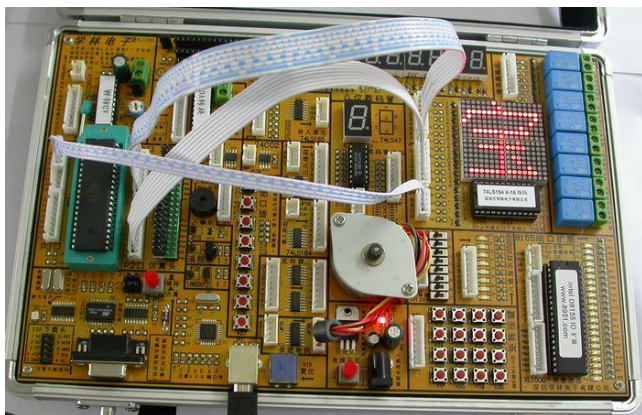
实验 16 汉字显示屏倚天一出宝刀屠龙 (仅限 XL1000)

汉字显示屏广泛应用于汽车报站器，广告屏等。XLISP 系列试验仪有一个标准的 16*16 点阵的汉字显示屏。他的行扫描部分分别由单片机的 P0, P2 口担任，列扫描部分则是由一个 74LS154 的 4-16 译码器组成。其电路原理如下：



接线方法：此实验仅限 XL1000

- 1 用一根 8PIN 的数据排线，一端接到汉字屏部分的 JP55 一端接到 CPU 部份的 P0 口 JP51.
- 2 用一根 8PIN 的数据排线，一端接到汉字屏部分的 JP54 一端接到 CPU 部份的 P2 口 JP52.
- 3 用一根 4PIN 数据线一端插入 CPU 部分 JP44 (P1 口) 的 P1. 0, P1. 1, P1. 2, P1. 3 另外一端插入汉字屏部分的 JP62.





试验参考程序：汉字屏轮流显示倚天一出宝刀屠龙八个汉字。

```

ORG 00H
START: MOV A, #0FFH
      MOV P0, A
      ANL P2, #00
      MOV R2, #200
D1:   MOV R3, #248      ;延时
      DJNZ R3, $
      DJNZ R2, D1
      MOV 20H, #00H
11:   MOV R1, #100
L2:   MOV R6, #16
      MOV R4, #00H
      MOV R0, 20H
L3:   MOV A, R4
      MOV P1, A          ;开三极管
      INC R4
      MOV A, R0
      MOV DPTR, #TABLE
      MOVC A, @A+DPTR
      MOV P0, A          ;查表送 P0 口
      INC R0
      MOV A, R0
      MOV DPTR, #TABLE
      MOVC A, @A+DPTR
      MOV P2, A          ;查表送 P2 口
      INC R0
      MOV R3, #02
D2:   MOV R5, #248
      DJNZ R5, $
      DJNZ R3, D2
      MOV A, #00H
      MOV P0, A
      ANL P2, #00H
      DJNZ R6, L3
      DJNZ R1, L2
      MOV 20H, R0
      CJNE R0, #0FFH, L1  ;检测是否送完
      JMP START

TABLE :
db 01H, 00H, 02H, 00H, 04H, 00H, 1FH, 0FFH; 倚
db 0E2H, 00H, 22H, 00H, 22H, 0FCH, 26H, 88H
db 2AH, 88H, 0F2H, 88H, 2AH, 0FAH, 26H, 01H
db 63H, 0FEH, 26H, 00H, 02H, 00H, 00H, 00H

db 02H, 01H, 42H, 01H, 42H, 02H, 42H, 04H; 天
db 42H, 08H, 42H, 30H, 42H, 0C0H, 7FH, 00H
db 42H, 0C0H, 42H, 30H, 42H, 08H, 42H, 04H
db 0C2H, 02H, 46H, 03H, 02H, 02H, 00H, 00H

db 01H, 00H, 01H, 00H, 01H, 00H, 01H, 00H; 一
db 01H, 00H, 01H, 00H, 01H, 00H, 01H, 00H
db 01H, 00H, 01H, 00H, 01H, 00H, 01H, 00H
db 01H, 00H, 03H, 00H, 01H, 00H, 00H, 00H

db 00H, 00H, 00H, 7FH, 3FH, 02H, 02H, 02H; 出
db 02H, 02H, 02H, 02H, 02H, 02H, 0FFH, 0FEH
db 02H, 02H, 02H, 02H, 02H, 02H, 02H, 02H
db 3FH, 02H, 00H, 7FH, 00H, 00H, 00H, 00H

db 08H, 00H, 30H, 02H, 22H, 02H, 22H, 42H; 宝
db 22H, 42H, 22H, 42H, 0A2H, 42H, 63H, 0FEH
db 22H, 42H, 22H, 52H, 22H, 0DAH, 26H, 42H
db 22H, 06H, 28H, 02H, 30H, 00H, 00H, 00H

db 00H, 00H, 20H, 01H, 20H, 02H, 20H, 04H; 刀
db 20H, 18H, 20H, 60H, 3FH, 80H, 20H, 00H
db 20H, 00H, 20H, 02H, 20H, 01H, 20H, 02H
db 7FH, 0FCH, 20H, 00H, 00H, 00H, 00H, 00H

db 00H, 02H, 00H, 0CH, 7FH, 0F4H, 51H, 08H; 屠
db 55H, 10H, 55H, 3FH, 55H, 6AH, 5FH, 0AAH
db 55H, 2AH, 57H, 2AH, 55H, 2AH, 59H, 2AH
db 0F1H, 7FH, 43H, 20H, 01H, 00H, 00H, 00H

db 08H, 00H, 08H, 01H, 08H, 02H, 08H, 0CH; 龙
db 08H, 30H, 09H, 0C4H, 0FEH, 04H, 08H, 08H
db 0FH, 0FCH, 48H, 22H, 28H, 42H, 08H, 82H
db 0BH, 02H, 18H, 02H, 08H, 1EH, 00H, 00H
end

```


实验 17

1602 液晶显示屏显示 A

XLISP 系列液晶显示屏部分可以分别挂接 12864 中文汉字屏, 1602 字母显示屏, 0802 字母显示屏等, 随机机附带的是 1602 字母显示屏。他可以显示 2 行每行 16 个英文字符。0802 液晶屏和 1602 液晶屏完全兼容。其编程方法, 指令都是完全一样的, 所不同的是他只能显示 2 行每行 8 个字符。12864 中文液晶则可以显示 128*64 点阵, 包括带字库和不带字库两种, 目前市面上较多的都是带字库的。

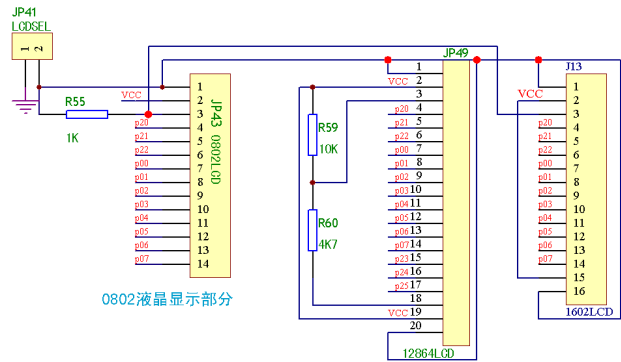
下面我们简要介绍一下 1602 显示屏的操作方法。

XLISP 系列的 1602 液晶的接线原理如下图所示:

接口信号说明:

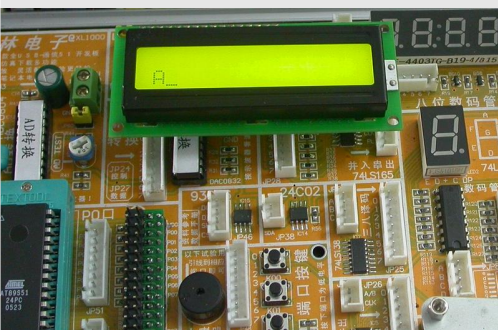
编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	Data I/O
2	VDD	电源正极	10	D3	Data I/O
3	VL	液晶显示偏压信号	11	D4	Data I/O
4	RS	数据/命令选择端 (H/L)	12	D5	Data I/O
5	R/W	读/写选择端 (H/L)	13	D6	Data I/O
6	E	使能信号	14	D7	Data I/O
7	D0	Data I/O	15	BLA	背光源正极
8	D1	Data I/O	16	BLK	背光源负极

0802/1602/12864 的接线原理图:

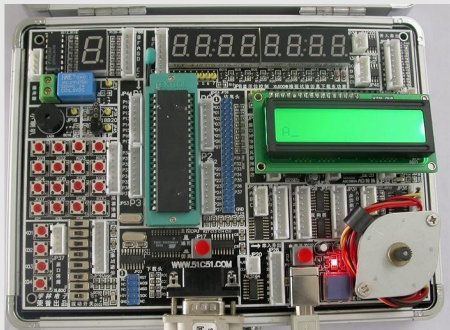


在本套件中, 所有的引脚都已经和 CPU 部分的相关端口连接好。直接按照图所示方向插入液晶即可。

其中 JP41 是液晶使能跳线。如果作液晶试验就必须闭合此跳线, 否则应当断开以避免干扰。其中 R55 是 1602/0802 公用的液晶对比度调整电阻。R59, r60 则是 12864 液晶的对比度调整电阻, 如果用其他厂家的液晶则可能要适当调整这些电阻以达到最佳显示效果。



x11000 运行照片 (实验完成后请取下液晶以免干扰)



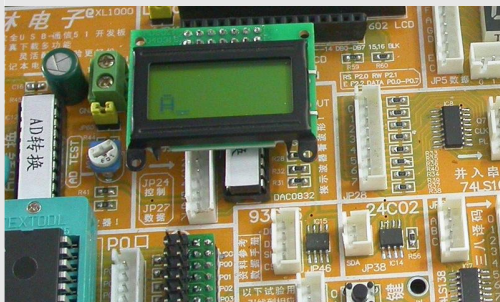
x1600 运行照片 (实验完成后请取下液晶以免干扰)



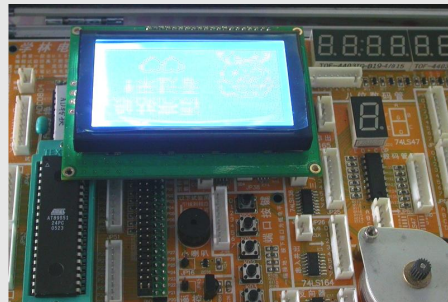
参考程序：用 1602 显示屏显示一个字母 A。

```
RS EQU P2.0
RW EQU P2.1
E EQU P2.2
ORG 0000H
MOV P0,#00000001B ;清屏
ACALL ENABLE
MOV P0,#00000001B ;清屏
ACALL ENABLE
MOV P0,#00111000B ;显示功能
ACALL ENABLE
MOV P0,#00001111B ;显示开关控制
ACALL ENABLE
MOV P0,#00000110B ;输入模式
ACALL ENABLE
MOV P0,#0C0H ;数据存储器地址
ACALL ENABLE
mov p0,#01000001b ;ASCII 码
SETB RS
CLR RW
CLR E

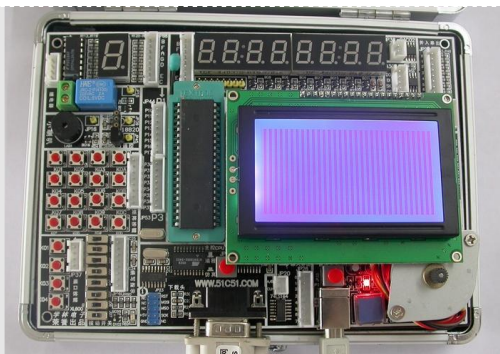
ACALL DELAY
SETB E
AJMP $
ENABLE: CLR RS ;送命令
CLR RW
CLR E
ACALL DELAY
SETB E
RET
DELAY: MOV P0,#0FFH
CLR RS
SETB RW
CLR E
NOP
SETB E
JB P0.7,DELAY ;判断忙标志
RET
END
```



x11000 连接 0802 液晶的照片



x11000 连接 12864 液晶的照片



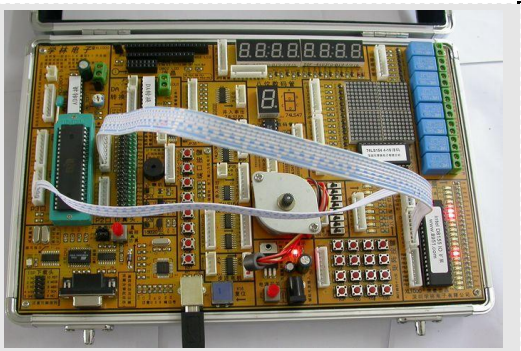
x1600 连接 12864 液晶的照片



运行光盘 12864ZIKU.HEX 的效果
(注意 12864 是选配件)

实验 18 8155 试验(仅限 XL1000)

8155 是作为输入输出以及 RAM 扩充的 IC。
特性如下： 256 字节的 RAM。
一组可编程 6 位 IO 口。
两组可编程 8 位 IO 口。
可编程 14 位二进制计时计数器。
多工地址和数据总线。
内部地址锁存。
8155 采用 40 脚双列直插封装，单一+5v 电源。



管脚说明：

PC3— 1	ALE— 11	21 —PA0	31 —PB2
PC4—2	AD0— 12	22 —PA1	32 —PB3
TIMERIN— 3	AD1— 13	23 —PA2	33 —PB4
RESET—4	AD2— 14	24 —PA3	34 —PB5
PC5—5	AD3— 15	25 —PA4	35 —PB6
TIMEROUT—6	AD4— 16	26 —PA5	36 —PB7
IO/M— 7	AD5— 17	27 —PA6	37 —PC0
CE/CE— 8	AD6— 18	28 —PA7	38 —PC1
RD— 9	AD7— 19	29 —PB0	39 —PC2
WR— 10	VSS— 20	30 —PB1	40 —VCC

RESET:复位信号线，高电平有效。在该输入端加一脉冲宽度为 600ns 的高电平信号，就可使 8155 可靠复位，复位时三个输入/输出口预置为输入方式。

CE:片选端,8155 为低电平有效。8156 为高电平有效，当 8155 上加上一个低电平时，芯片被选中，可以与单片机交换信息。

AD0~AD7:三态地址/数据总线。在 ALE 的下降沿把 8 位地址锁存于内部地址锁存器，地址可代 RAM 或输入/输出用，由 IO/M 信号的极性而定，8 位数据的流向取决于 RD 或 WR 信号的状态。

ALE:地址锁存器启用信号线，高电平有效，其下降沿把 AD0~AD7 上的地址，片选信号、IO/M 信号锁存起来。

IO/M:IO 和 RAM 选择信号线。高电平造反输入/输出，该线低电平选择存储器。

RD：读信号线，低电平有效。当片选信号与 RD 有效时，开启 AD0~AD7 缓冲器，如果 IO/M 为低电平，则 RAM 的内容读至 AD0~AD7，如果 IO/M 为高电平，则选中的输入/输出口的内容读到 AD0 ~AD7。

WR:写信号线，低电平有效。当片选信号和 WR 信号有效时，AD0~AD7 上的数据将根据 IO/M 极性写入 RAM 或 I/O 口。

PA0~PA7:输入/输出口 A 的信号线。通用 8 位输入/输出口，输入/输出的方向通过对命令/状态寄存器的编程来选择。

PB0~PB7:输入/输出口 B 的信号线。通用 8 位输入/输出口，输入/输出的方向通过对命令/状态寄存器的编程来选择。

PC0~PC5:输入/输出口 C 的信号线。6 位可编程输入/输出口，也可用作 A 和 B 口的控制信号线，通过对命令/状态寄存器编程来选择。

VSS:接地线, 接到公用地线上。



(注意在 XL1000 中另外设立了一个 JP61 作为 8155 的 PB 口输出。可用于其他试验)



8155 试验例程：用 8155 扩展出 26 路的跑马灯程序，运行可看到 26 路灯轮流闪烁。

```
A3: MOVX @R1, A
CALL DELAY
RL A
DJNZ R2, A3
MOV R1, #03H ; PC 口地址 03H
MOV R2, #06H
MOV A, #0FEH
A4:MOVX @R1, A
CALL DELAY
RL A
DJNZ R2, A4
JMP A1
    DELAY: ; 延时程序
MOV R7, #255
D1:MOV R6, #255
D2: DJNZ R6, D2
DJNZ R7, D1
RET
End
```

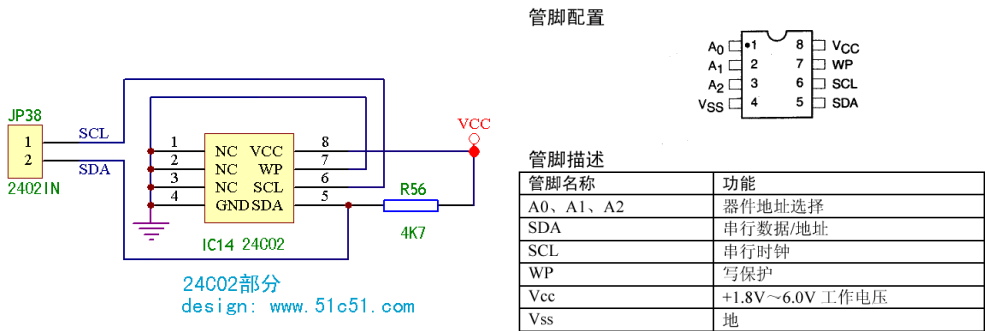

实验 19

24C02 储存开机次数实验

24C02 是 2K 字节的串行 EEPROM，内部含有 256 个 8 位字节，该器件通过总线操作，并有专门的写保护功能。下面给出的是 24C02 的电路原理图和器件引脚描述。

串行 EEPROM 简称 I2C 总线式串行器件。串行器件不仅占用很少的资源和 I/O 线，而且体积大大缩小，同时具有工作电源宽、抗干扰能力强、功耗低、数据不易丢失和支持在线编程等特点。

I2C 总线是一种用于 IC 器件之间连接的二线制总线。它通过 SDA（串行数据线）及 SCL（串行时钟线）两根线在连到总线上的器件之间传送信息，并根据地址识别每个器件：不管是单片机、存储器、LCD 驱动器还是键盘接口。



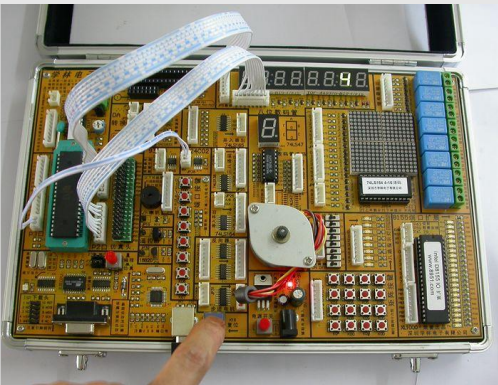
我们通过一个试验来了解 24C02 的读写操作过程：

该试验功能是单片机复位一次，自动从 24C02 中读取数据，然后加 1，最终数码管中的数据就是开机的次数，具有一定的实用意义。

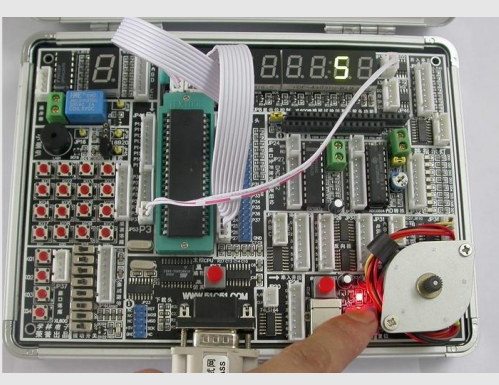
接线方法：

- 1 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51。
- 2 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52。
- 3 用一根 2PIN 数据线一端插入 CPU 部分 JP53（P3 口）的 P3. 6, P3. 7 另外一端插入 24C02 部份的控制端 JP38。

烧写后用手按复位键可以看到数码管每按一下加一。



XL1000 运行照片



XL600 运行照片

```
SDA24 EQU P3.7
SCLK24 EQU P3.6
ORG 0000H
AJMP MAIN
ORG 0030H
MAIN: CLR P3.7 ;打开写保护
MOV DPTR, #TAB
MOV A, #00H ;读地址
LCALL RD24
CJNE A, #10, TT
TT: JNC TT1
AJMP TT2
TT1: MOV A, #00
TT2: MOV 30H, A
MOVC A, @A+DPTR
CLR P2.6 ;开数码管
MOV P0, A ;送显示
MOV A, 30H
INC A
MOV B, A
MOV A, #00H
LCALL WT24
AJMP $
TAB: DB
28H, 7EH, 0A2H, 62H, 74H, 61H, 21H, 7AH, 20H, 60H
RD24: PUSH ACC ;读 24C02 子程序。
SETB SDA24
CLR SCLK24
CALL START24 ;启动
MOV A, #0A0H
CALL SHIFT8 ;移位
CALL ACK ;响应
POP ACC
CALL SHIFT8
CALL ACK
CALL START24
MOV A, #0A1H
CALL SHIFT8
CALL ACK
SETB SDA24
MOV R7, #8
CLR A
SETB SDA24
RD000: RL A ;8 个位
SETB SCLK24
MOV C, SDA24
MOV ACC.0, C
CLR SCLK24
DJNZ R7, RD000
RET
WT24: PUSH ACC;写入 24C02 程序:
SETB SDA24
CLR SCLK24
CALL START24
MOV A, #0A0H
CALL SHIFT8
CALL ACK
POP ACC
CALL SHIFT8
CALL ACK
MOV A, B
CALL SHIFT8
CALL ACK
CALL STOP
CALL DELAY2
RET
START24: CLR SDA24;开始条件
SETB SDA24
SETB SCLK24
CLR SDA24
CLR SCLK24
RET
STOP: CLR SDA24;停止条件
SETB SCLK24
SETB SDA24
RET
ACK: SETB SCLK24;应答信号
CLR SCLK24
RET
SHIFT8: MOV R7, #8;读、写数据
SH01: RLC A
MOV SDA24, C
SETB SCLK24
CLR SCLK24
DJNZ R7, SH01
RET
DELAY2: NOP
END
```


实验 21 93c46 演示程序

93c46 是 1k 位串行 EEPROM 存储器。每一个存储器都可以通过 DI/DO 引脚写入或读出。它的存储容量为 1024 位，内部为 128×8 位或 64×16 位。93C46 为串行三线 SPI 操作芯片，在时钟时序的同步下接收数据口的指令。指令码为 9 位十进制码，具有 7 个指令，读、擦写使能、擦除、写、全擦、全写及擦除禁止。该芯片擦写时间快，有擦写使能保护，可靠性高，擦写次数可达 100 万次，以下给出了 93C46 与单片机的接线图，和引脚说明。

管脚名称	功能
CS	片选信号
SK	时钟输入
DI	串行数据输入
DO	串行数据输出
Vcc	电源 (+1.8 伏到 6 伏)
GND	接地
ORG	存储器结构选择
NC	不用连接
PE*	写入保护

DIP Package (P)

93c46部分

说明：93c46 的第 6 脚 org 为 16/8 位模式选择，XLISP 系列套件第 6 脚接地，仅工作在 8 位模式下。

XLISP 系列的 93C46 部分原理图



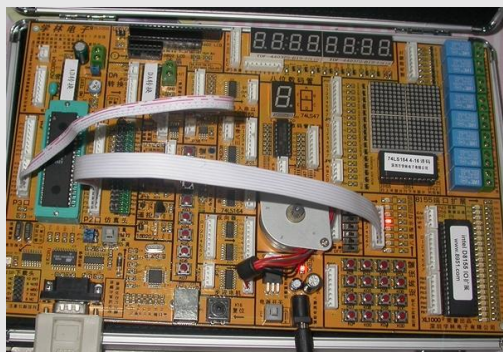
接线方法：

1 用一个 4PIN 数据排线一端插入 CPU 部分 JP53（P3 口）的 P3. 4, P3. 5, P3. 6, P3. 7。另外一端插入 93C46 部分的输入端 JP46。

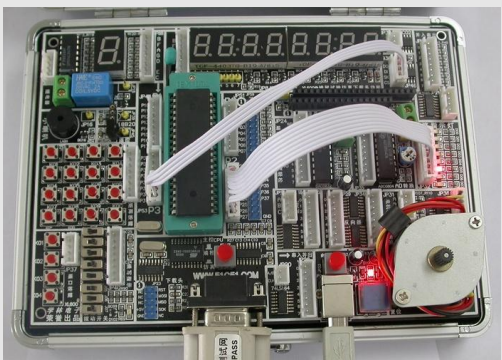
2 用一条 8PIN 的数据排线把 CPU 部份的 P2 口（JP52）接到八路指示灯部份的 JP32。

此程序先把 8 路跑马灯的数据写入 93C46，然后读出送 P2 口显示。运行程序可以观察到 8 路跑马灯。为证明 93C46 的作用，可以拔下 93c46 的数据线，再次运行程序，则没有跑马灯。

93C46 的跑马灯运行照片



XL1000 运行照片



XL600 运行照片

```

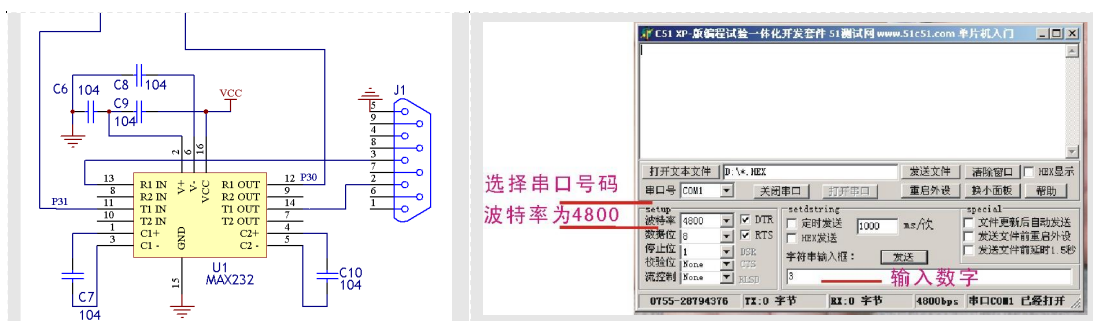
READ EQU 0 ;读 93C46
WRITE EQU 2 ;写入 93C46
EWEN EQU 4 ; 93C46 写入使能
EWDS EQU 6 ; 93C46 写入禁止
CS BIT P3.7 ; 93C46 CS
CLK BIT P3.6 ; 93C46 CLK
DI BIT P3.5 ; 93C46 DI
DO BIT P3.4 ; 93C46 DO
ADR46 EQU 20H
ORG 00H
START0: MOV 21H, #00H ; 取地址码
        MOV 22H, #08H ; 8 组 8 个码
        MOV P3, #10H
        MOV ADR46, #00H ;93C46 地址 00H
START: MOV A, #EWEN ;写入使能
        CALL T09346
        MOV A, 21H ;载入取码指针
        MOV DPTR, #TABLE ;至 TABLE 取码
        MOVC A, @A+DPTR
        MOV R4, A ;存入“写入寄存器”
        MOV A, #WRITE ;写入数据
        CALL T09346
        MOV A, #EWDS ;写禁止
        CALL T09346
        INC 21H ;取下一个码
        INC ADR46 ;下一个地址
        CALL DELAY1
        DJNZ 22H, START
        MOV R0, #30H ;存入 51 的 RAM 地址
        MOV ADR46, #00H ;93C46 :00 地址
        MOV 22H, #08H ; 93C46 8 个码
A1: MOV A, #READ ;读(地址)的数据
     CALL T09346
     MOV A, R4 ;93C46 输出数据载入 ACC
     MOV @R0, A
     INC ADR46 ;读下一个地址
     INC R0 ;下一个 RAM
     DJNZ 22H, A1
A2: MOV 22H, #08H
     MOV R0, #30H
OUTPUT: MOV A, @R0
        MOV P2, A ;输出至 P2
        CALL DELAY1 ;
        INC R0
        DJNZ 22H, OUTPUT
        JMP A2
T09346: SETB CS ;93C46 CS=1
        SETB DI ;DI=1
        SETB CLK ;CLK=1
        CALL DELAY ;启动位
        CLR CLK ;CLK=0
        CALL DELAY
        MOV DPTR, #JPTBL46
        JMP @A+DPT ;依 A 值跳至相关地址
JPTBL46: AJMP SREAD ;读
        AJMP SWRITE ;写
        AJMP SEWEN ;写使能
        AJMP SEWDS ;写禁止
SREAD: MOV A, #80H ;送操作码 10
        CALL SDATA
        MOV A, ADR46 ;载入 93C46 地址
        CALL SADR ;写入 7 位地址 (DI)
        CALL RDT46 ;读 (DO), 存入 3CH
        MOV R4, A
        JMP EX9346
SWRITE: MOV A, #40H ;送操作码 01
        CALL SDATA
        MOV A, ADR46 ;载入 93C46 地址
        CALL SADR ;写入 7 位地址
        MOV A, R4 ;写入 (3CH) 数据
        CALL SDT46
        JMP EX9346
SEWEN: CLR A ;送操作码 00
        CALL SDATA
        MOV A, #60H ;写使能码 11XXXXX
        CALL SADR
        JMP EX9346
SEWDS: CLR A ;送操作码 00
        CALL SDATA
        CLR A ;写 7 位禁止操作码 000XXXX
        CALL SADR
EX9346: CLR CS ;CS=0
        RET
SADR: CLR C
        RLC A
        MOV R3, #7
        JMP SD1
SDATA: MOV R3, #2
        JMP SD1
SDT46: MOV R3, #8 ;写入 8 位数据
SD1: RLC A ;
        MOV DI, C
        SETB CLK
        CALL DELAY
        CLR CLK
        CALL DELAY
        DJNZ R3, SD1
        RET
RDT46: MOV R3, #8 ;读出 8 位数据
RD1: SETB CLK
        CALL DELAY
        CLR CLK
        CALL DELAY
        MOV C, DO
        RLC A
        DJNZ R3, RD1
        RET
DELAY: MOV R7, #0FH
        DJNZ R7, $
        RET
DELAY1: MOV R4, #20
D1: MOV R5, #20
D2: MOV R6, #248
        DJNZ R6, $
        DJNZ R5, D2
        DJNZ R4, D1
        RET
TABLE: DB 0FEH, 0FDH, 0FBH, 0F7H ;广告一个灯左移
        DB 0EFH, 0DFH, 0BFH, 7FH ;
        END

```

实验 22 串行双向通信实验

所谓“串行通信”是指外设和计算机间使用一根数据信号线(另外需要地线,可能还需要控制线),数据在一根数据信号线上一位一位地进行传输,每一位数据都占据一个固定的时间长度。这种通信方式使用的数据线少,在远距离通信中可以节约通信成本。例如,目前在 IBM PC 机上的 COM1、COM2 接口,就是 RS-232C 接口。

在简单的应用中,只需要有 3 条线即可完成通信,分别是 第二脚 RXD, 第 3 脚 TXD, 第 5 脚 GND。串行通信与单片机之间的接口:由于串行通信的电平逻辑定义是+15V(高电平 1) -15V(低电平 0) 而单片机中分别用 5V, 0V 来表示 1, 0 它们之间必须通过电平转换才可以完成通信。最常用的是美信的 MAX232 电路。



以下程序烧写通过后,关闭 xlist 烧写软件,把 XLISP 系列的 JP17(编程/仿真)按下到仿真状态。打开配套光盘的工具软件 sscomv20 串口调试工具。选择通信串口号,波特率 4800:



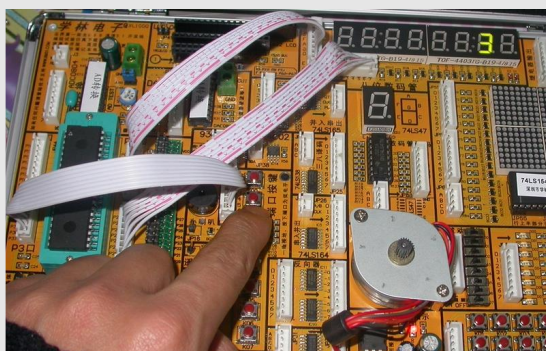
接线方法:

- :1 按键接到 P3 口。用一条 8PIN 数据排线,把按键部份的 JP48,接到 CPU 部份的 P3 口 JP53。
- :2 接 8 位数码管的数据线。将数码管部份的数据口 JP5 接到 CPU 部份的 P0 口 JP51。
- :3 接 8 位数码管的显示位线。将数码管部份的显示位口 JP8 接到 CPU 部份的 P2 口 JP52。

接收过程:在字符串输入框中输入 1,然后点“发送”XLISP 系列的数码管则显示 1。输入 2 则显示 2。

发送过程:分别按下 XLISP 系列端口按键部分的 K02, K03, K04, K05 四个按键,这时可以看到串口调试工具分别收到 4 句话。你按的是 p3.2 键,你按的是 p3.3 键,你按的是 p3.4 键,你按的是 p3.5 键,

窗口发送数字 3 数码管显示 3



按键窗口接收到的 4 句话。

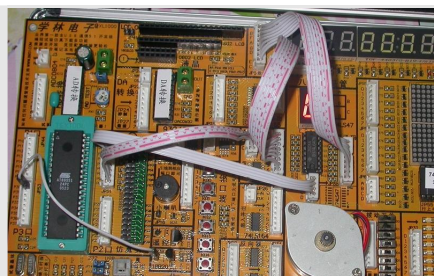
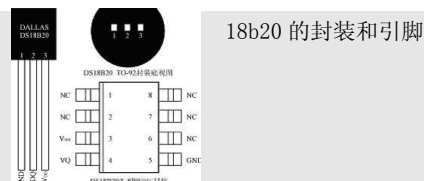
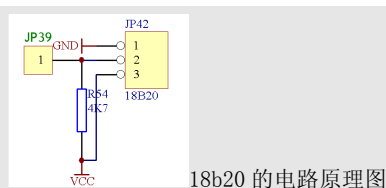
```
ORG 0000H
AJMP main0
ORG 0030H
main0: MOV 20H, #00H      ;设置串口工作方式
MOV TMOD, #22H
MOV SCON, #40H
MOV TH1, #0F3H          ;设置速率 4800, (12M 晶体)
MOV PCON, #80H
SETB TR1
SETB REN      ;允许接收
clr p2.6      ;开数码管
ajmp start
start:
l1: jnb p3.3, l3      ;检测键盘, 如果 P3.3 按下则执行 L3
jnb p3.4, l4
jnb p3.5, l5
JNB P3.2, L6
jbc ri, rec
mov p0, #0ffh
rec: mov a, sbuf      ;缓冲数据送到 a
mov p1, a
CJNE A, #00110001b, T00      ;比较接收到的数据
MOV P0, #07eH      ;数码管显示 1
mov p0, #0ffh
t00: CJNE A, #00110010b, T2
MOV P0, #0A2H      ;数码管显示 2
mov p0, #0ffh
T2: CJNE A, #00110011b, T3
MOV P0, #62H
T3: CJNE A, #00110100b, T4
MOV P0, #74H
T4: CJNE A, #00110101b, T5
MOV P0, #61H
T5: CJNE A, #00110110b, T6
MOV P0, #21H
T6: CJNE A, #00110111b, T7
MOV P0, #7AH
T7: CJNE A, #00111000b, T8
MOV P0, #20H
T8: CJNE A, #00111001b, t9
MOV P0, #60H
T9: CJNE A, #00110000b, t10
MOV P0, #28H
t10: ljmp L1
l3: mov dptr, #ok1 ;置表头 1
ajmp senda3
senda3: clr a ;
movc a, @a+dptr
cjne a, #'$', senda_3
ajmp start ;遇到$则从头开始执行。
senda_3: mov sbuf, a ;发送数据
JNB TI, $;
CLR TI ;
inc dptr
Ljmp senda3
l4: mov dptr, #ok2 ;置表头 2
ajmp senda3
l5: mov dptr, #ok3 ;置表头 3
ajmp senda3
l6: mov dptr, #ok4 ;置表头 4
ajmp senda3
ok1: db 0dh, 0ah, 0dh, 0ah, " 你 按 的 是 p3.2 键", 0dh, 0ah, '$'
ok2: db 0dh, 0ah, 0dh, 0ah, " 你 按 的 是 p3.3 键", 0dh, 0ah, '$'
ok3: db 0dh, 0ah, 0dh, 0ah, " 你 按 的 是 p3.4 键", 0dh, 0ah, '$'
ok4: db 0dh, 0ah, 0dh, 0ah, " 你 按 的 是 p3.5 键", 0dh, 0ah, '$'
end
```



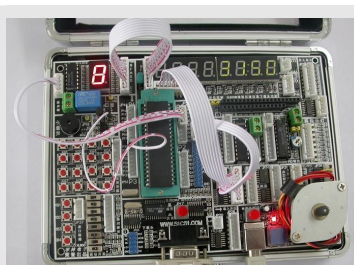
实验 23 综合实验 18B20 数字温度显示系统(仅限 XL1000)XL600 需要自配芯片

综合实验：用 18B20（数字温度采集）74LS47(数码管译码)74LS138(三八译码) DS18B20 是 DALLAS 公司生产的一线式数字温度传感器，具有 3 引脚 TO-92 小体积封装形式；温度测量范围为 $-55^{\circ}\text{C} \sim +125^{\circ}\text{C}$ ，可编程为 9 位~12 位 A/D 转换精度，测温分辨率可达 0.0625°C ，被测温度用符号扩展的 16 位数字量方式串行输出。

主机控制 DS18B20 完成温度转换必须经过三个步骤：初始化、ROM 操作指令、存储器操作指令。必须先启动 DS18B20 开始转换，再读出温度转换值。本程序仅挂接一个芯片，使用默认的 12 位转换精度，外接供电电源，读取的温度值高位字节送 WMSB 单元，低位字节送 WLSB 单元，再按照温度值字节的表示格式及其符号位，经过简单的变换即可得到实际温度值。



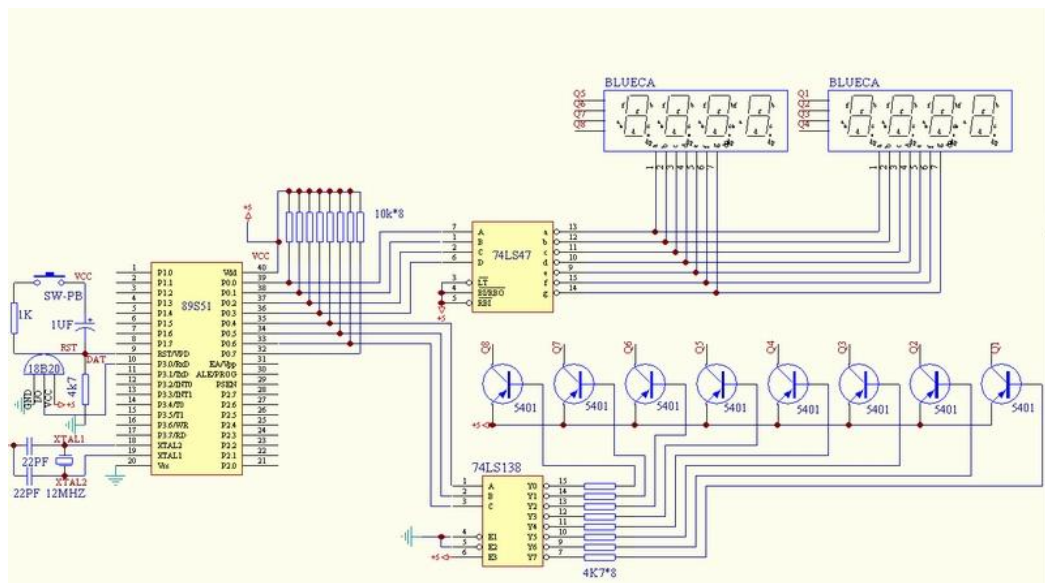
运行程序数码管后 4 位显示当前测试到的温度。



xl600 的运行照片

在本系统中：为了简化程序，采用了 74LS47(数码管译码)74LS138(三八译码)。即 P0 口的 P0.0, P0.1, P0.2, P0.3 四个端口接到 74LS47 进行硬件数码管译码，然后输出到数码管部分的数据口 JP5。P0.4, P0.5, P0.6 三个端口接到 74LS138 进行 38 译码，然后输出到数码管的位控制 JP8。

系统综合原理图：



```

DAT    BIT P3.0          ;数据通信口
WDLSB DATA 30H          ;读出的温度低字节
WDMSB DATA 31H          ;读出的温度高字节
        ORG 0000H
        LJMP MAIN
        ORG 000BH
        SJMP TIMERO      ;定时显示
;定时显示子程序,循环扫描方式;显示缓冲区 40H~47H
        ORG 0040H
TIMERO:
        MOV P0,40H        ;最后一位值送 p0
        ORL P0,#70H; 点亮最后一位
        CALL T1MS
        MOV P0,41H
        ORL P0,#60H
        CALL T1MS
        MOV P0,42H
        ORL P0,#50H
        CALL T1MS
        MOV P0,43H
        ORL P0,#40H
        MOV TH0,#LOW(65536-10000) ;定时器 10MS 中断
        MOV TLO,#HIGH(65536-10000)
        RETI
T1MS:  mov r5,#00h        ;延时子程序
tt:    mov r6,#9
        djnz r6,$
        djnz r5,tt
        ret
;***** *****;主程序:
MAIN:   MOV SP,#60H
        MOV P2,#0FFH
        MOV R2,#8
        MOV RO,#40H ;
OVER:   MOV @RO,#00H
        INC RO
        DJNZ R2,OVER
        MOV TMOD,#01H
        MOV TH0,#LOW(65536-10000)
        MOV TLO,#HIGH(65536-10000)
        SETB EA
        SETB ETO
        SETB TR0
LOOP:   LCALL DSWD ;调用读出 DS18B20 温度程序
        SJMP
LOOP
        ; 读出 DS18B20 温度程序
DSWD:   CLR EA
        LCALL RST
        JNB FO,KEND      ;如果没有应答,返回主程序
        MOV RO,#0CCH
        LCALL SEND_BYTE  ;跳过 ROM 匹配
        MOV RO,#44H      ;发出温度转换命令
        LCALL SEND_BYTE
        SETB EA
        MOV P1,#00001111B
        MOV 48H,#1       ;延时 75ms 以上准备读
SS2:    MOV 49H,#255
SS1:    MOV 4AH,#255
SS0:    DJNZ 4AH,SS0
        DJNZ 49H,SS1
        DJNZ 48H,SS2
        MOV P1,#11111100B
        CLR EA
        LCALL RST
        JNB FO,KEND
        MOV RO,#0CCH      ;跳过 ROM 匹配
        LCALL SEND_BYTE
        MOV RO,#0BEH      ;发出读温度命令
        LCALL SEND_BYTE
        LCALL READ_BYTE
        MOV WDLSB,A
        LCALL READ_BYTE
        MOV WDMSB,A
        LCALL TRANS12
KEND:   mov p1,#01010101b
        SETB EA
        RET
;*****
;温度转换程序: 精确到: 0.06 ;
TRANS12:MOV A,30H
        ANL A,#0F0H
        MOV 3AH,A
        MOV A,31H
        ANL A,#0FH
        ORL A,3AH
        SWAP A
        MOV B,#10
        DIV AB
        MOV 43H,A
        MOV 42H,B ;
        MOV DPTR,#TABB
        MOV A,30H
        ANL A,#0FH
        MOV B,#2
        MUL AB
        MOVC A,@A+DPTR
        MOV 41H,A
        MOV A,30H
        ANL A,#0FH
        MOV B,#2
        MUL AB
        INC A
        MOVC A,@A+DPTR
        MOV 40H,A
        RET
TABB:DB 0,0,0,6,1,2,1,8,2,5,3,1,3,7,4,3,5,0
        DB 5,6,6,2,6,8,7,5,8,1,8,7,9,3
;*****
SEND_BYTE:      ;发送一个字节程序
        MOV A,RO
        MOV R5,#8
SEN3:   CLR C
        RRC A
        JC SEN1

```

<pre> LCALL WRITE_0 SJMP SEN2 SEN1: LCALL WRITE_1 SEN2: DJNZ R5, SEN3 ; 循环 8 次, 写一个字节 RET READ_BYTE: ; 读一个字节程序 MOV R5, #8 READ1: LCALL READ RRC A DJNZ R5, READ1 ; 循环 8 次, 读一个字节 MOV R0, A RET ;复位程序, 如果复位位置位 F0, 没有就复位 F0 RST: SETB DAT NOP NOP CLR DAT MOV R6, #250 ;主机发复位脉冲持续 3 μs×200=600μs DJNZ R6, \$ MOV R6, #50 DJNZ R6, \$ SETB DAT ;主机释放总线, 口线改为输入 MOV R6, #15 DJNZ R6, \$ CALL CHCK ;调用应答检查程序 MOV R6, #60 DJNZ R6, \$ SETB DAT RET CHCK: MOV C, DAT JC RST0 SETB F0 ;检测到信号, 置位 F0 SJMP CHCK0 RST0: CLR F0 ;未准备好 F0 复位 CHCK0: RET </pre>	<pre> WRITE_0: ;写 0 CLR DAT MOV R6, #30 DJNZ R6, \$ SETB DAT RET WRITE_1: CLR DAT ; 写 1 NOP NOP NOP NOP NOP SETB DAT MOV R6, #30 DJNZ R6, \$ RET ;读一位数据程序 READ: SETB DAT ;复位 1μs 产生读起始信号 NOP NOP CLR DAT NOP NOP SETB DAT ;置位 DAT 准备接收数据 NOP NOP NOP NOP NOP NOP NOP MOV C, DAT MOV R6, #23 DJNZ R6, \$ RET END </pre>
---	---



本程序测量温度范围: 0~125 度, 精确到 0.06;

综合接线图: 仅限 XL1000)XL600 需要自配芯片

- 1 用一根单条数据线把 18B20 的 JP39 接到 CPU 部分的 JP53(P3 口)的 P3.0
- 2 用一条 4PIN 的排线, 把数码管译码部份的 JP19 接到 CPU 部份 P0 口的 JP51 的 P0.0, P0.1, P0.2, 0.3 四个端口。(即插入 P0 口的上半部分)。
- 3 用一条 8PIN 的排线。把数码管译码部份的输出端 JP88, 接到数码管部分的数据口 JP5
- 4 用一条 4PIN 的排线, 把 38 译码部份的 JP22 接到 CPU 部份 P0 口的 JP51 的 P0.4, P0.5, P0.6, 07 四个端口。(即插入 P0 口的下半部分)。此处不太好插入, 小心操作。
- 5 用一条 8PIN 的排线。把 38 译码部份的输出端 JP25, 接到数码管部分的显示位口 JP8。

第七章 介绍一个好用的开发软件---dais 集成开发环境

本文是笔者发表在无线电杂志 2005 年 11 期的一篇文章，学习单片机的朋友可以学习使用该软件编译与软仿真功能。特别注意：这里只是介绍了一款软件的使用法，和 XLISP 系列套件并无任何关系！

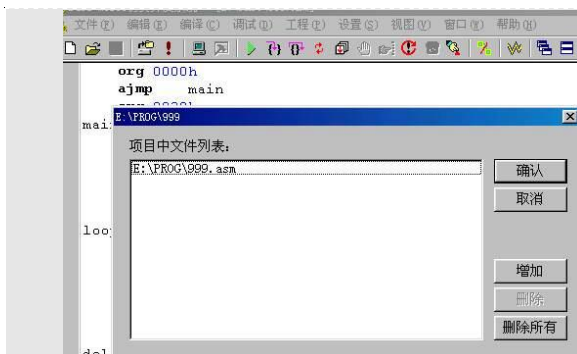
学习单片机之初最先遇到的问题就是不知道怎样把程序生成 hex 文件。常用的 keil demo 版使用起来设置比较复杂，为此向大家介绍一款 dais 开发环境软件。该软件全中文界面，可以实现编译，软件仿真一体化。并且程序支持汇编/C 混合调试，中文注释，全兼容于 WINDOWS98/2000/XP 操作体系。

点击配套驱动光盘/开发工具/DIAS 集成开发环境/目录下的 SETUP，一路回车，（注意不要安装在其他路径，应当安装在默认的 C:\DIAS 中）即可安装成功，在桌面上生成图标：

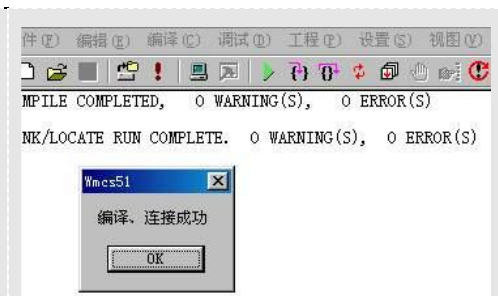


双击运行，出现连接对话框，**在这里需要注意的是事实上我们并没有这个仿真器**，点取消，我们可以利用该软编译生成 HEX 文件，并进行软件仿真。在本篇中，主要介绍它的软件仿真和编译功能。

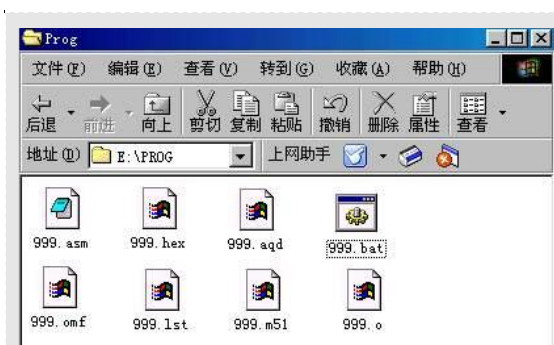
进入后，点新建文件，在窗口中输入源程序，然后把他另存为 ASM，假定我们把他存为 E:\PROG\999.ASM 具体方法为点文件\另存为即可，如果在 E 盘没有 PROG 目录，应当首先建立一个，当然并不是一定要取这个名字，只是举例说明而已，你也可以把他存在 C 盘或者 D 盘。汇编是底层操作，需要注意的是不要使用中文的文件夹，更不能把文件名取为中文名字。例如你把文件存在 D:\程序\流水灯.ASM 就不行。同时文件的名称不要超过 8 个英文字符。（没办法，谁叫咱比老外落后呢）另外也不要吧系统的盘符设置为“系统盘”，“软件盘”，“游戏盘”之类的中文盘符。



点“保存”后，再点菜单的工程“新建”在这里，我们仍然把工程的名称取为 999 这时将弹出工程的对话框，此时它是一个新的工程，我们需要为它增加一个项目文件，即刚刚保存的 999.ASM 点“增加”然后点“确定”



好了，准备工作完成了，点击菜单/编译/文件编译，连接，装载（或者下图图标的红色感叹号）如没有异常将出现成功的对话框。这里需要补充一点，汇编语言对语法要求比较严格，如果编译不成功请仔细的检查程序本身有没有语法错误。当然这种情况下，dais 会提示您的第几行出现错误。



此时在我们的 E:\PROG 目录下即生成了以上的八个文件，其中的 999.ASM 是汇编的源程序，999.HEX 就是我们所需要的目标代码。我们可以用编程器将这个文件写进单片机中，程序就可以运行了。

现在我们介绍利用他的软件仿真功能。对于初学者来说，用他来熟悉 8051 单片机的指令系统非常的直观，如我们用他来学习单片机的加减法。

以下面的一个简单的小例子来说明：25h+36h 等于多少？

我们可以首先输入下面的程序：

```
MOV R0,#25H;将立即数 25 送寄存器 R0
```

```
MOV A,#36H;将立即数 36 送累加器 A
```

```
ADD A,R0;a 与 r0 相加
```

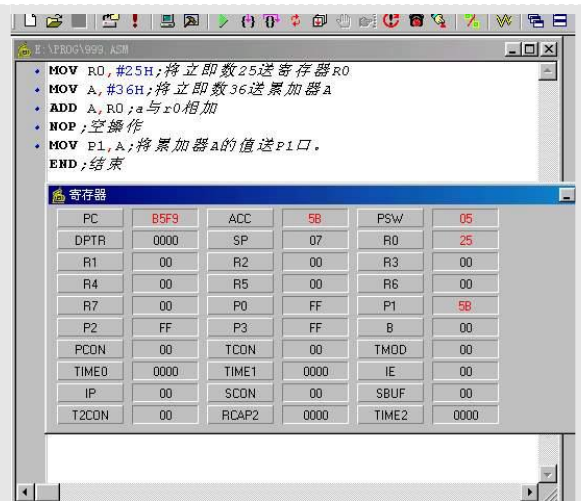
```
NOP;空操作
```

```
MOV P1,A;将累加器 A 的值送 P1 口
```

```
END;结束
```

然后把程序按照上面的方法编译，打开视图\寄存器窗口并单步运行，

可以直观的看到 ACC，P1，R0 三个寄存器在执行每一步之后的变化，最后的结果如下图：



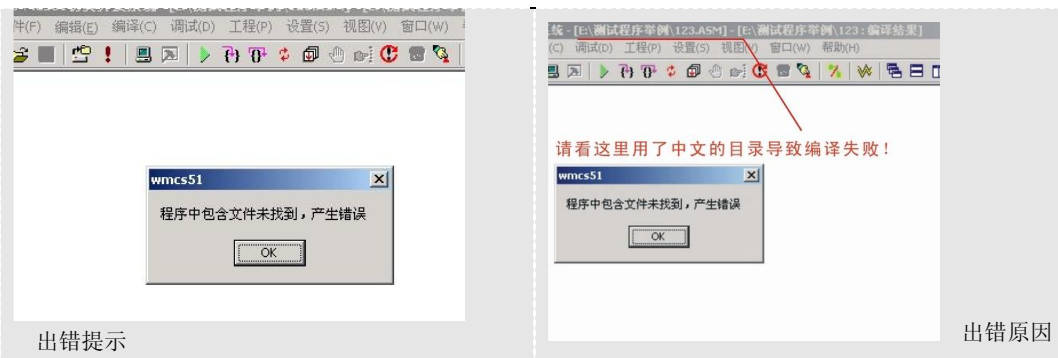
我们可以看到 ACC 的值为 5BH，这个就是 25H+36H 的结果了。P1 口的值也是 5BH，这是 MOV P1，A（把 A 送往 P1 口）的结果。同时我们也可以观察到 R0 的值是 25H，这个是由于程序中我们把 R0 送入 25H 的结果。

同样的我们来实验单片机的减法指令，例子如下：计算 36H-25H 的值，把上面的程序第 3 行改为 SUBB A，R0 其余的部分相同，编译后打开寄存器窗口并单步运行，我们可以观察到 P1 口和 ACC 的值都变成了 11H，这就是 36-25h 的值了。

参照以上方法我们可以利用这个软件来熟悉 8051 单片机的其他指令，体会各个寄存器之间的变化，对于初学单片机的朋友很有帮助。

特别补充：dais 软件为啥会出现编译失败？

dais 软件在我们推广使用来，已经有数千的用户使用这个软件，但是有很多朋友无法正常使用。经过打电话或者发邮件，咨询我们，发现是由于用了非法的文件名或者非法的目录引起的！如下图所示！希望引起初学的朋友重视，虽然我们在 dais 的说明文档中反复强调这个问题，但是还是不断有朋友犯相同的错误.....



出错提示

出错原因

第八章 常见问题解答

如果您是一个初学者，那么我强烈建议您仔细阅读以下内容，并反复校对对自己可能错误的地方。



1 为何我的 XLISP 系列套件在烧写程序的时候总是显示：当前串口不存在请选择其他串口？

电源是否确定打开了？ 电源指示灯是否已经亮？

检查编程/仿真的按钮的状态，必须在弹起的状态下才可以编程！

是否有其他的软件占用了串口？ 例如其他的串口编程器，串口调试器，以及 kiel uv2 仿真软件。

一台机器上只能同时运行一个串口的软件。 否则相互冲突，不能联机。

是否串口选择错误？ 可以在控制面板/系统/硬件属性/端口中察看串口的号码， 设置为一致即可。



2 为何提示烧写成功，目标板运行， 但我的 XLISP 系列却没有反应？

检查卡座上的芯片是否 89s51？ 如果你放入的是仿真监控芯片， 或者根本没有放芯片，当然不能运行。

所需要的端口是否经过信号排线正确的连接？ 用排线引入相应的实验资源才可以看到运行结果。

可能是您的系统没有正确的复位， 可以按一下系统的复位键看看。



3 在仿真状态下，为何总是不能联机？

仿真状态下，系统的卡座上必须放入随机的仿真监控芯片 sst89e564. 并且这个芯片已经烧写有监控文件。

如果这里放入的不是这一个芯片， 当然不行。

仿真状态下，必须使用我们光盘中的 keil uv2 软件， 并且注意这个软件必须安装在 c 盘的根目录， 不能安装在其他的目录！（参考第 5 章，XLISP 系列 仿真操作指南）建立文件的时候， 也不要保存在其他的目录，更不要使用中文的文件名和文件夹。

串口的选择是否正确？ 由于仿真/烧写都需要用到串口， 为此：仿真操作时， 必须关闭掉 xlisp 烧写软件以及其他可能用串口的软件。

编程/仿真按键是否设置正确？（仿真时此按键为按下状态）



4 为何仿真头仿真我自己的电路板不行， 仿真本身的资源又可以？

检查一下自己的电路板的复位， 晶体是否正常， 直接烧片工作正常否？

检查您的仿真头的方向插入是否正确？ 特别注意，仿真头的第一脚（红色）对应 cpu 的第一脚。 可以用万用表检查一下。



6 为何有些程序在烧写方式运行时可以， 而仿真方式却不行？

在仿真状态，会占用部份的系统资源，不能做到 100%的完全真实的仿真， 事实上，仿真的概念也就是“模拟”， 如果您对结果有疑问， 那么请切换到 isp 烧写方式运行。 一般情况下，我建议用烧写-试验的方式。



7 如果我想用 2051 单片机怎么办？

由于 2051 和 s51 相比只是少了 p0 口和 p2 口， 其编程方法，指令都是一样的，所以你完全可以用 s51 代替 2051 来做实验，编程时避开 p0, p2 口即可。如果必须用 2051， 那么我们将用优惠的价格供应其他编程器给您。



8 随机配的 89s51 和书上的 89c51 是一样的么？

89s51 是 89c51 的升级版本， 他们的指令格式完全是兼容的，也完全可以相互替代。并且价格更低，寿命更长。89s51 增加了看门狗功能和 isp 下载功能。

第九章：系统配置和售后服务指南



标准配置: (此配置据不同情况增减)

- XLISP系列 单片机综合仿真实验仪主机一台（已经调试好，板载试验用89s51一片，1602液晶）
- 备用89s51芯片一片
- 仿真芯片SST89E564一片
- 串行通信电缆一根
- USB电缆一根
- 超薄HT6221红外线遥控器一个
- 40PIN 仿真头一个
- 10PIN ISP 下载头一个
- 试验用系统连接数据线15条
- 使用手册一本（本手册）
- 软件光盘一张（驱动程序\开发工具\例子程序\单片机资料\教程。）



选配件:

- (12864字库液晶，可选件， 和本产品一起购买， 只需要90元)



保修条款:

在无自行拆修情况下， 本产品自售出日起三个月免费包换一年保修。(试验芯片，仿真芯片除外)客户可直接与本公司或者到各代理商处更换。



技术支持:

凡本产品用户，请到最新本公司论坛 <http://www.8951.com/bbs/index.asp> 注册用户，以便得到更多的使用帮助和更多的源程序！ 在本论坛上随时都是数十位朋友在和您一起分享学习的喜悦！

更多信息及源程序请点击 www.8951.com www.51c51.com

相信大家在完成以上试验后， 对 8051 单片机的 I/O 口已经有了充分的了解，本实验仪配套光盘中有更多的例子程序，只需要仔细的看看程序中所调用的 I/O 口， 就可以在本实验仪上完成实验。或者， 您在其他的教材，网站看到的源程序， 只要本实验仪上有此硬件资源， 就可根据程序的 I/O 口来安排实验。 从而抛砖引玉， 达到更好的学习效果。

本光盘中赠送的部分实验参考程序：

*****跑灯类程序

8led.asm	八个左右跑马灯。采用了寄存器存中间数
18led.asm	p1 口 8 个跑马灯的另外一个程序。用移位指令
28led.asm	p1 口八个灯作二进制加法。理解二进值的计算
38led.asm	P1 口八个灯做 3 路跑动显示。分别往端口送三个不同的数，理解数字和端口的对应关系
int1.asm	p1.1 中断方式闪烁。
plled.c	c 语言的跑灯程序。
int01.asm	P1 口的 8 个灯循环亮，按 P32 后全部闪烁 5 次后继续循环亮， 中断程序的典型应用。
int8led.asm	开机 8 个灯循环， 按 p32 后暂停，小喇叭响，松开继续循环，中断的典型应用。
iotest.asm	端口测试程序用于仿真器的测试， 每一个端口轮流点亮

*****简易键盘控制类程序

p3212.asm	p3.2 小键盘控制 p1.2 发光管的程序。
-----------	-------------------------

p3305.asm p3.3 小键盘控制 p1.3led, 瞬间亮, 0.5s 熄灭。

p3355.asm p3.3 小键盘控制 p1.3led, 0.5s 亮, 0.5s 熄灭。

keyshuzi.asm 四个小键盘控制的数字显示。

keyled.asm 四个小键盘控制的花样彩灯;

key.asm 四个小键盘控制的数码管小灯程序。

k4.asm 一键多功能按键识别技术

k3.asm 按键识别技术

***** 小喇叭类程序

dida.asm 小喇叭的警报器。

p33didi.asm 小喇叭发出嘀声。

p33dida.asm 小喇叭发出救护车的声音。

HAPPY.asm 小喇叭唱歌祝你生日快乐。

music.asm 小喇叭唱歌祝你平安;

8music.asm 八月桂花香唱歌程序。

p33pa.asm 小喇叭的爬音演奏, 即 do, rui, mi, fa, so, la, xi, do

keymic 小电子琴, 按 P3 口的小键盘喇叭发出不同频率的声音。同时数码管显示不同的数字。

6music.asm 六首唱歌的音乐程序按 p3.4 循环

2002.asm 小喇叭唱 2002 年第一场雪

*****0802 /1602 液晶显示程序

(0802 和 1602 液晶程序电路完全兼容, 0802 只能显示 2 行 8 个字)

0802A.asm 为驱动 0802 液晶显示器显示一个字母 A 的源程序。倚天版直接使用

0802logo.asm 为驱动 0802 液晶显示器 (hd44780 兼容芯片) 的 51 测试网 logo 程序倚天版直接使用

80209.asm 为 0802 液晶显示器 (hd44780 兼容芯片) 的全屏显示 0-9 驱动程序倚天版直接使用

0802key.asm 小键盘控制的 0802 液晶显示器。腾龙版直接使用

0802999.asm 液晶的 6 位计数器程序

0802go.asm 0802 液晶的 16 位按键显示程序

0802pass.asm 0802 液晶的 16 位键盘密码输入器

0802PLAY.ASM 液晶显示的网站广告, 两屏轮流显示

INPASS.ASM 0802 液晶的 16 位键盘密码输入器的另一种形式 (位置不变)

PASSPLAY.ASM 液晶的密码输入器的屏显部分 (无输入部分供大家对比研究)

***** 数码管显示类程序

999999.asm 为 0-999999 的计数器 (适用各个版本显示)

999999B.asm 为 0-999999 的计数器的另一种形式 (适用各个版本显示)

clock.asm 为六位数码管时钟程序, p3.2 控制. (适用各个版本显示)

clockok.asm 为 六位数码管时钟程序, p3.2, p3.3, p3.4, p3.5 四键控制!

led895152.asm 显示 895152 同时 8 个小灯跑动 (适用各个版本显示)

012345.asm 数码管显示 012345 的程序. (适用各个版本显示)

8051.asm (适用各个版本显示) 8051 数字的程序.

test.asm (适用各个版本显示) 测试程序轮流显示 1-7, 流水灯。

0099.asm (适用各个版本显示) 数码管显示 100 位计数器。

ABCDEF.ASM 两路动态扫描显示程序, 开机显示 012345, 如果按下 P32 显示 ABCDEF

shu09.asm 定时中断实现显示的程序, 第一位始终显示 0, 第二位显示在 0-9 之间循环

***** 计数器控制类程序

daojishi.asm 倒计时器 开机默认为 15 秒, p32 启动键, p34 个位调整, p35 十位调整

9999jishu.asm; 6 位数的计数器源程序, 可以从 0 一直到 999999, 按 p3.2 计数, 开机显示 000000

9jishu.asm; 一位计数器, 按 p3.2 数码管加 1, 到 9 后回 0

d9999.asm; 可预制的倒计时钟, 2 位显示 p34 个位调整 p35 十位调整 p32 启动!

9keybb.asm; 一位计数器, 按 p3.2 数码管加 1, 到 9 后回 0, 同时小喇叭响一下报警,

setkey.asm 可预制的倒计时钟, p35 个位, p34 十位, p32 确定

*****遥控器控制程序:

n6221.asm 为配套的红外线遥控 (P3.7) 显示数码管 1234567890 的程序

n6221c.asm 倚天版遥控器控制 30 个 io 的 led 点亮, 可由此控制继电器实现家电的遥控

n6221b.asm 倚天版遥控器控制显示 0123456789abcdef 数据保持

yaokong.asm 遥控器测试程序按任意键试验版数码管指示灯闪烁。任意遥控器均可!

*****通信类程序:

232.asm 为 XP 版 (11.0592m 晶体, 供参考) 试验串口通信程序,

232KEY.asm 串口通信程序。(该程序适用于本站各个版本用户晶体 12M)

232shuzi.asm 为串口通信程序的计算机控制单片机,

s232.asm 双向通信程序, 以上两个程序的结合。

*****AD 转换类程序:

0804.asm.....0804 的基本应用, 转动电位器, P0 口显示取到的数

0804kong.asm....adc0804 制作的温度控制器, 设一个加热器 p2.0 一个制冷器 p1.4

AD09.asm.....转动电位器, 数码管依次显示 1-9, 0, ABCDE

AD255.asm.....控制程序, 转动电位器, 数码管后三位依次显示 000-255,

*****步进电机控制类程序

6221mada.asm....遥控电机程序, 按遥控器 0-9 步进电机正转 0-9 步, 按遥控器 11-19 步进电机反转

dianji.asm.....步进电机正转, 反转各 3 圈

keyd.asm.....步进电机控制程序 p3.2 正转, p3.3 反转, p3.4 停止 步进电机接 p1.0p1.1p1.2p1.3

abdian.asm.....步进电机正反快慢转!

madatest.asb.....步进电机测试, 最简单的转圈程序

*****部分 c 语言编程的范例。

plled.c.....流水灯 c 语言编程实例

0802dis.c.....腾龙 0802 液晶显示 c 语言编程实例

4ledint.c.....4 位不带小数点的频率计 c 语言编程实例

led09.c.....在 6 个 LED 数码管上依次显 0123456789 速度由快到慢

999jishu.c.....在 6 个 LED 数码管上显示 000000, 按 p3.2 开始计数再按停止

cclock.c.....简易电子时钟, LED 数码管显示 p32——时调整 p33——分调整

jinbao.c.....小喇叭警报声源程序 端口 P3.3

kp1.c.....按键扫描程序按 P34 LED 向右移一位 按 P35 LED 向左移一位

led34567.c.....在 6 个 LED 数码管上依次显 234567

*****综合程序:

24c02.asm 为 24c02 储存计数器程序, 断电数据保持, 每次开机数码管显示加 1. 一共记到 9.

ledping.asm 是斑竹新近的小制作 256 个发光管制作的 16*16 汉字 led 屏, 轮流显示“倚天一出

mytest.hex 倚天版测试程序! 该程序由湖北枝江 - 飞扬软件工作室 吴涛 友情提供 11/22/2003

ldmsapp.asm 倚天版液晶遥控程序, 液晶显示遥控器的编码- 飞扬软件工作室 吴涛 友情提供

05hz.asm ;变频输出! 按 p3.2 连续输出 50hz, 1KHZ, 1KHZ 窄脉冲, 停止。输出口 P3.3。
