本书所附光盘包含书中全部范例程序

8051 单片机

彻底研究实习篇

林伸茂 编著
管继斌 白雁钧 改编

人民邮电出版社
POSTS & TELECOM PRESS

封面设计：胡平利

FLAG

8051 单片机

彻底研究实习篇

本书适合对 8051 单片机有一定基础的读者阅读。

书中主要以单片机控制板为描述主体，再配合其他的电路组合成一个典型的数字控制系统。本书共分为四大部分，分别探讨了 8051 单片机的诸多经典范例。

ISBN 7-115-12203-2

9 787115 122032 >

人民邮电出版社网址 www.ptpress.com.cn

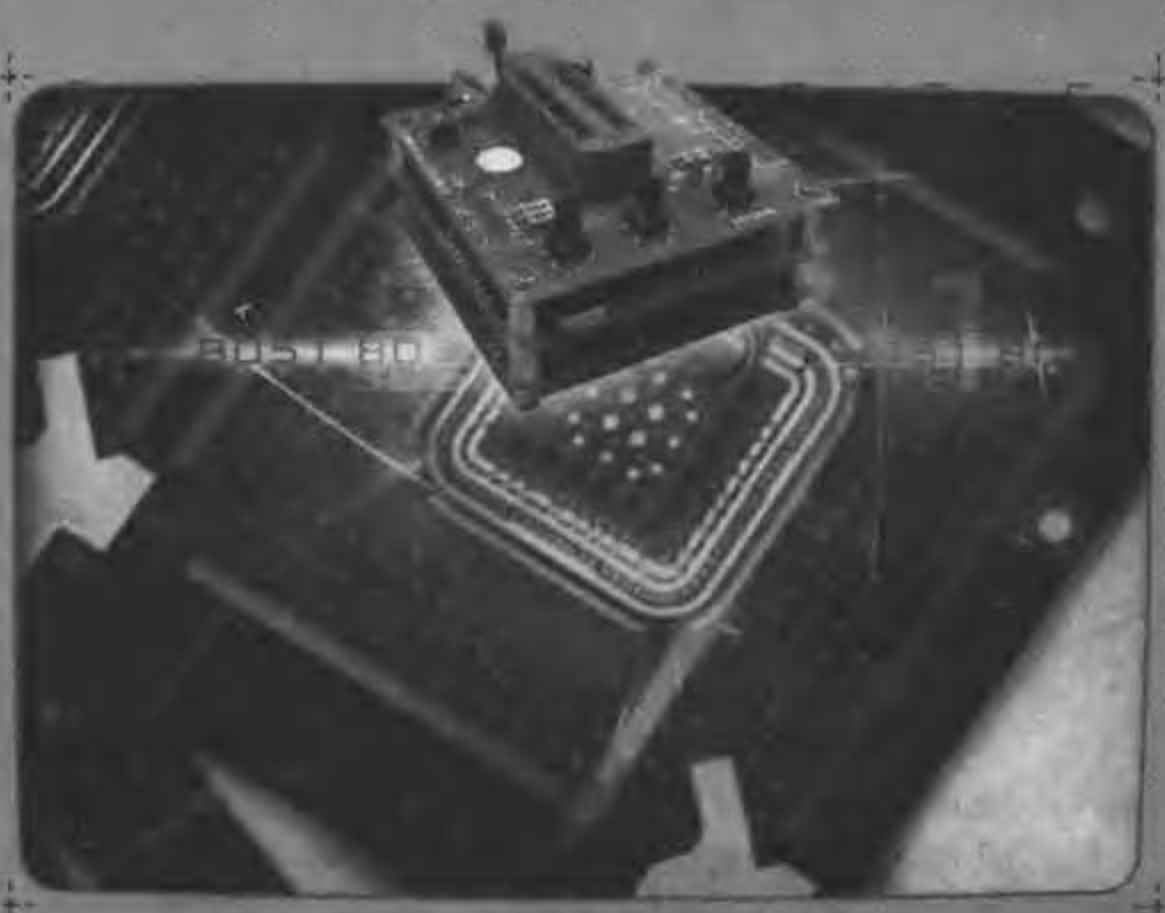
ISBN7-115-12203-2/TP·3927
定价：32.00 元(附光盘)

8051 单片机

彻底研究实习篇

林伸茂 编著

管继斌 白雁钧 改编



人民邮电出版社

图书在版编目（CIP）数据

8051 单片机彻底研究. 实习篇/林伸茂编著；管继斌，白雁钧改编.
—北京：人民邮电出版社，2004.5
ISBN 7-115-12203-2

Ⅰ. 8… Ⅱ. ①林… ②管… ③白… Ⅲ. 单片微型计算机，8051—基本知识

Ⅳ. TP368.1

中国版本图书馆 CIP 数据核字（2004）第 031088 号

8051 单片机彻底研究 实习篇

◆ 编　　著　林伸茂
改　　编　管继斌　白雁钧
责任编辑　俞　彬

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号
邮编　100061　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
读者热线　010-67132705
北京汉魂图文设计有限公司制作
北京隆昌伟业印刷有限公司印刷
新华书店总店北京发行所经销

◆ 开本：787×1092　1/16
印张：16.75
字数：395 千字　　2004 年 5 月第 1 版
印数：1-5 000 册　　2004 年 5 月北京第 1 次印刷

著作权合同登记　图字：01－2002－6542 号

ISBN 7-115-12203-2/TP・3927

定价：32.00 元（附光盘）

本书如有印装质量问题，请与本社联系　电话：（010）67129223

内容提要

本书适合对 8051 单片机有一定基础的读者阅读。书中主要以“旗威科技”生产的FLAG51 单片机控制板为描述主体，再配合其他的电路组合成一个典型的数字控制系统。本书共分为四大部分，分别探讨了 8051 单片机的诸多经典范例。

第一部分 8051 进阶研究，包括第 1 章至第 7 章，主要介绍了 8051 的基本应用范例与数字仪器的使用，以及软硬件的除错技巧，这些都是 8051 进阶者所需具备的专业知识。本部分还介绍了把 8051 单片机改成 8052 后的差异所在和 8051 的时序（Timing）研究。

第二部分 FLAG51 扩充，包括第 8 章至第 15 章，主要介绍了一个以 8051 单片机为基础的控制板 FLAG51 的开发过程。

第三部分温度湿度仪制作应用，包括第 16 章至第 20 章，该部分是温湿度制作的专题报告，主要介绍了如何通过利用 AT89C2051 制作温度计与湿度计。

第四部分为 RS485 串行通信彻底研究，包括第 21 章至第 23 章，主要对 8051 串行通信 RS485 进行了技术与程序上的探讨。

本书选材实用性和可操作性强，范例丰富，文字叙述清楚，对于已经有一定基础的 8051 读者具有重要的参考价值，也适合作为大专院校学生做实验、专题制作、研究和设计单片机产品的专业参考书。

前　言

我们的确不一样

本书虽然不是我亲手写在稿纸上，却是我亲自用拼音输入法一个字一个字地打进电脑，但是接下来的校稿与编排就不是我一个人能够完成的。书可是白纸黑字绝对不能随便敷衍了事的。别人出书的模式与心态我们是不知道的，可是自己的书一定尽力做到自己满意为止。所以，文章的内容我们斟酌再三，所引用的照片几乎都是自己拍摄的，示波器的波形图也是我们直接从仪器上得到的，许多图表与 8051 的指令集也是经过我们细心地整理后才公布的。我们认为这样做才是我们做事的态度，同时这也是“旗威科技”公司写程序的正确态度。如果无法与众不同，又如何脱颖而出？做人处事如此，个人认为出书也是如此。

所以有些书籍的名称如果是《一举学成 8051》或《30 天学会 8051》，我们都会投以怀疑的眼光，如果真那么容易，那么就不会有人写书了。因为我们自己花了整整 3 年的时间透彻地研究过 8051，才有第一本 8051 单片机彻底研究的书（1993 年旗标出版社出版）出来，这本书现在已经绝版了。又过了整整 10 年的时间，我们把 8051 从头到尾确确实实地翻了好几遍，才有这系列新书问世，时间与事实证明 8051 确实是继 Z80CPU 后另一个单片机的常青树。

单片机系列书籍介绍

单片机系列套书：

《8051 单片机彻底研究基础篇》、《8051 单片机彻底研究实习篇》、《8051 单片机彻底研究经验篇》。

本书主要强调在 8051 单片机系统扩充与整合应用两方面的专业知识。如果你想彻底了解 8051 单片机与汇编语言的写法，请务必参考《8051 单片机彻底研究基础篇》，书中对 8051 各个指令的用法与暂存器的运作，做了深入详尽的交待。而《8051 单片机彻底研究经验篇》谈到的方向就更广了，例如：8051C 语言的导入、如何取得 Shareware SDCC C 编译程序，以及 8051 的优缺点分析等等，这些都是相当宝贵且不可多得的 8051 专业知识。

8051 相当具有弹性的串行通信搭起了与 PC 连接的桥梁，8051 单片机系列书籍中仔细地分析了 8051 内部串行通信的架构与写法。另一方面，你若想对 PC 端的串行通信进一步了解与运用视窗程序操控时，建议你参考《VB 与串行通信彻底研究》一书，只要你有微软的 VB 就可以很方便地与自行设计的 8051 控制器沟通，当然也可以与各种外部的仪器连线了。

本书内容收录

本书的内容有部分是取自历年来我们在《RUN!PC》杂志上发表过的文章，主要的内容谈到 FLAG51 控制板的扩充与应用。当然已删除部分硬件设备不合时宜的部分，另外，我们也对 8051 的重要时序与 8052 新增的功能做了相当详尽的分析与探讨。

AT89C2051 是缩小版的 8051，它内含 2KB 的 Flash 且只有 20 根引脚，但是用它来做一般的实验都已足够，本书有一系列的温湿度的控制应用范例，都是以 AT89C2051 为控制主体。你绝对可以在这里看到 8051 的深度应用实例。

8051 最让人津津乐道的正是其串行通信的能力，在本书最后几章中也有提到 RS485 接口的应用解说与实例，“旗威科技”早已把类似的线路应用在石化工业与医疗设备上。

如何阅读本书

这本书适合对 8051 稍有基础的读者阅读。书中主要以“旗威科技”生产的 FLAG51 单片机控制板为描述主体，再配合其他的电路组合成一个典型的数字控制系统。本书共分为四大部分，分别探讨到 8051 单片机的诸多经典范例。

第一部分谈到 8051 的基本应用范例与数字仪器的使用，以及软硬件的排错技巧，这些都是 8051 进阶者所需具备的专业知识。当我们要把 8051 单片机改成 8052 时，你知道其中的差异吗?这些不同点都在本书的第 6 章上提到。8051 的时序（Timing）研究则是硬件工程师另一项考验，相同的线路经过两个硬件工程师的处理与安排后，可能会有完全不同的结果，其中的差异可能就是对时序的了解程度了，第 7 章里我们分别用示波器与逻辑分析仪说明 8051 的重要时序，这可能是除了 Intel 原厂的资料外，对时序探讨最透彻的中文文章了。

第二部分提到一个以 8051 单片机为基础的控制板 FLAG51 是如何被开发出来的，从构想到整合是一连串设计的组合。接下来我们以 FLAG51 控制板为主体，陆续开发了 I/O 监视板、七段显示板、数字隔离输入板与 RELAY 输出板等等，这些控制板的设计与开发的历程都一并记录在书中。

第三部分为温湿度制作的专题报告，我们利用 AT89C2051 去制作温度计与湿度计，这方面的测量虽是属感测器的范畴，但是控制与显示的主体却是 8051 的汇编语言程序，我们认为所有 8051 的进阶者都要经过类似的考验，方能堂堂正正进入单片机的设计主流群体当中。

第四部分为 8051 串行通信 RS485 的彻底研究。许多仪器或设备都有 RS485 通信接口，只通过两条对绞线就可以控制多达 32 台设备。本书的这部分即做这方面技术与程序上的探讨，懂得这方面的知识后，你绝对会对 8051 另眼相看的。

致谢

编写 8051 单片机一系列书绝对不是一个人所能完成的，它绝对是一个团队的工作总整合，3 年前我就开始筹备新书的出版事宜，所有的文章与内容经过整理过滤与调整补充，最后确实的章节与内容才得以在 2002 年元月时固定下来。在这段整合的期间，我要特别感谢以下帮助我的人们：

王圣心小姐与姜莹贞小姐：初步整理已发表过的文章，光是校稿就校了无数次，并拍摄许多照片，让本系列的书籍得以完成初步的架构。李浩蓁先生与曾琼惠小姐：进行本书版面调整与最后的校稿，整本书是在他们的手中完成的。太克科技台湾分公司罗仕林先生与浩网科技公司的庄昱宏与黄芳川先生：提供最高级的示波器与逻辑分析仪，以及技术上的协助，让本书的图表资料与数据更有看头。

最后，我还是要谢谢家人所给予的鼓励，尤其是刚在牙牙学语的小女儿，没有他们几近狂热的激励与支持，就没有这系列书的问世。

林伸茂

chipware@chipware.com.tw

目　录

第 1 章　8051 新手入门

第 2 章　单片机相关仪器设备的认识与使用

第 3 章　试写两个 8051 范例程序

第 4 章　单片机实战应用三例

第 5 章　软硬件排错技巧

第 6 章　8052 与 8051 的差异

第 7 章　8051 的时序彻底研究

第 8 章　FLAG51 开发过程

第 9 章　简易计数器的设计规划

第 10 章　8051 单片机的专长：计数及计时

第 11 章　FLAG51 单片机的问与答

第 12 章　I/O 输出/输入板的开发

第 13 章　8051 应用实例 FLAG-DISP

第 14 章　FLAG-DISP 显示板应用与 DIY

第 15 章 FLAG-DISP 的创新应用

第 16 章 亲手做一台数字式温度计

第 17 章 用 AT89C2051 做一台温湿度显示计

第 18 章　智能型温湿度计 TH2030 的制作

第 19 章　温湿度传感器应用

第 20 章　个人计算机温度监视器的制作

第 21 章　RS485 通信接口彻底研究（一）

第 22 章 RS485 通信接口彻底研究（二）

第 23 章 RS485 通信接口彻底研究（三）

附 录

第 1 章

8051 新手入门

对于 8051 初学者遇到的各种问题，我们将逐一进行分析，并在此基础上深入探讨 8051 的各项特性及技巧，如果你可以轻松且平静地面对这些问题，那么你就可以进入 8051 的设计领域了。

每年学生快要毕业的时候，我们照例会接到许多有关 8051 单片机的求助电话，内容基本上都是课题做不出来，希望能够“外包”给我们来处理，费用则由他们来支付。对于这类问题，我们通常只是指出其问题解决可能的方向，而不是“包办”。我们这样做，绝对不是置求助者于“死地”，因为纵使该专题不幸被否定，求助者自己还有把 8051 学好的机会。如果你糊里胡涂就毕业了，真正进入就业市场时，就会有部分公司因为你的不用心，花更多的心血与精力才能够将产品顺利地开发出来。最近我们帮助好几个开发性质的公司拟定单片机 8051 方面的面试考题，其实只要几个问题就可以检测出你在单片机或 8051 方面的实力，我们把这些相关的资料放在“旗威科技”的网站（http://www.chipware.com.tw）上，供需要的人查询或下载。

相反的情形是每个月我们也会收到许多初学者的传真或信函，希望能指引他们如何入门 8051，如何完成领导交给的任务以及如何收集 8051 参考数据的等等，对于这些问题我们都很乐意回答，因为你是真正有兴趣才会加入或从事这一行业，这时已经不是学校必修或必选的问题了，站在提高国内技术与设计水准的立场上，我们愿意毫不吝惜地提供自己的实际经验与你交流。我们也希望你除了提出问题外，还能表明身份以及目前的工作性质，方便我们更能适切地回答，对于上述回复，基本上我们是纯服务性的，唯一的条件是：将此问题同步公开在“旗威”的网站上，以同时也给其他人参考或学习用，因为这是一种无私的知识交流机会。

1-1 如何步入 8051 设计者的行列

针对 8051 初学者遇到的各种问题，我们打算用 6 个单元逐一来分析并深入探讨 8051 的各项特性及技巧，如果你可以轻松地面对这些问题时，表示你已经进入 8051 的设计之门了。接下来应该找几个实用的专题测试一下，唯有通过不断地学习与实践才能让你更进一步认识 8051 单片机。

我们将要探讨的项目有：

（1）初学者的准备。

（2）基本设备的使用。

（3）进入 8051 的第一步。

（4）单片机 8051 相关族系的认识与介绍。

（5）单片机 8051 的刻录相关技巧。

（6）单片机的基本应用。

（7）单片机 8051 的正确使用时机。

（8）单片机 8051 的通信方式分析。

1-2 初学者的准备

10 年前如果有人询问我要准备好哪些东西，才可以开始学单片机 8051，我一定会请他先去买一本 Intel 的 MCS-51 参考手册，这里面有许多 8051 指令的说明与介绍，而且也有许多标准的硬件参考线路。可是现在如果再次回答这个问题时，我会先请这位 8051 初学者准备好计算机与 Internet 的连接软件，随时由网络上获取最新的单片机相关资料。通常我会建议你连接到一个 8051 单片机专属的网站（8051 主页—— http:// www.ece.orst.edu/serv/8051），然后把有关 8051 的文字说明全部下载到计算机内，如果能够打印出来更好。仔细地看完这些专业人士所整理的资料后，你会很惊讶地发现：全世界竟然有这么多人埋头在使用及学习 8051 单片机！从这些文件说明中也可以知道 8051 相关芯片的发展趋势，以及各种软件全球各地的价位与使用范围，当然你也可以知道哪里可以下载 8051 的汇编语言共享软件等等。5 年前这些第一手资料绝对是无法唾手可得的，不过这些国外的数据清一色是英文版的，所以你必须有相当的英文程度，否则很难得知其中的精髓。图 1-1 是 Intel 公司在 1981 年出版的单片机 8051 用户手册。

图 1-1

1-3　8051 汇编程序何处寻

谈到 8051 的汇编语言翻译程序，国内已经有软件厂商开发完成类似的产品，其名称为 EP51，这是由“艾硕”公司的计算机工程师所写的，我们习惯将此套软件推荐给尚未有 8051 汇编程序的读者，其使用的方式与原版的 X8051 非常类似。另外你也可以在国外的网站里下载免费的 8051 汇编程序，我们经常在计算机屏幕前辛苦地排错，深刻感受到程序员的辛劳，所以我们常常建议别人买原版的软件，因为唯有这样才能使开发者有足够的持续力继续去开发下一套更新的软件。

以“旗威科技”为例，公司也花了许多时间完成 8051 的反汇编程序 DIS51.EXE，其中的许多数据汇整的功能是一般的反汇编程序所无法比拟的，我们把 32KB 以下的版本归为共享软件，你可以自行由网站中下载及使用，而完整的 64KB 版本则是商业版本，真正有如此需要时才花钱购买，这是一个相当变通的方法。我们发现一个很奇怪的现象：许多大公司及研究单位都会购买 64KB 的完整版本，专门用来看或是“研究”别人的程序，也许这也是一种学习方法。

有了汇编语言软件后，你应该有一套简单的 8051 控制板，方便我们将程序放在这块控制板上执行，并且查看执行后的结果。市面上有各式各样的 8051 控制板，较高价位的 8051 控制板上还有键盘及 LCD 的显示电路，你可依实际的试验需要购买。不过要特别考虑的是控制板的扩充及支持性是否足够。以“旗威”的 FLAG51 单片机控制板为例，我们已经发展出一系列可扩充的外加板，如：I/O 监视器、七段显示器、隔离输入板、RELAY 输出板以及 EPROM/FLASH 微控制刻录器等等，你可以随兴进行各种的实际工业控制应用，这是业界一般的 8051 学习板所欠缺的，而这也是我们一直努力的方向。图 1-2 是“旗威科技”

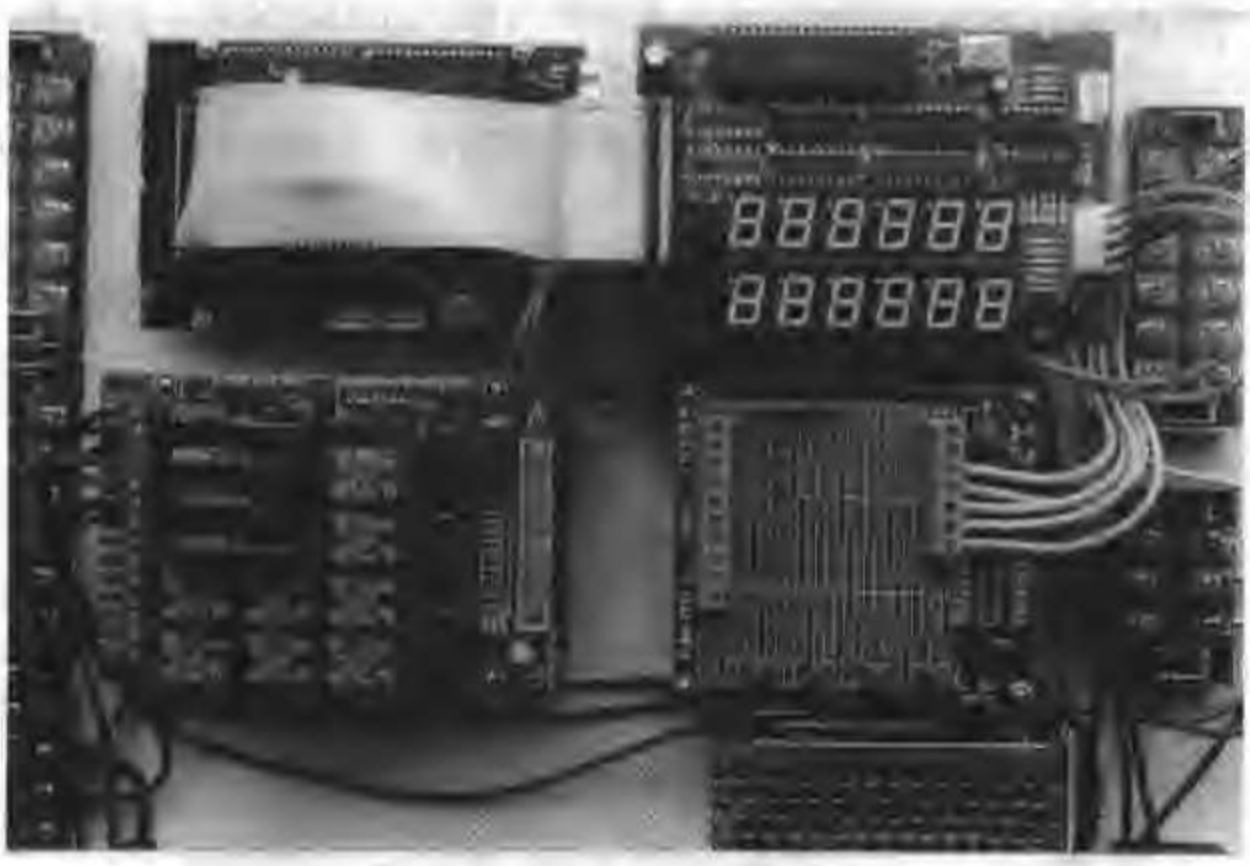


图 1-2

注：上图为以 8051 单片机为控制主体的专用机，左上为 FLAG51 单片机控制板，右上为 FLAGDISP 七段显示器及隔离输入板，左下为两片 RELAY 输出板，右下为额外的控制板。

于 1997 年 7 月完成的设计项目，这台设备是用于零件的 QC 测试的，除了 8051 单片机控制板外，内部有两块 RELAY 控制板，一块隔离输入板及 FLAGDISP 七段显示器等，许多人一直以为 RELAY 输出板及隔离输入板要配合 PLC 程序方能启动。其实只要我们知道其中 8255 I/O 的真正地址，用哪种语言都是可以控制的。当然，该类型的控制器用工业级的计算机也可以做，但是价位上就差上一大截了。

1-4 慎选电源供应器及计算机

硬件与软件汇编语言都定下来后，你还需要一台交换式的电源供应器，为 8051 单片机控制板提供+5V 的直流电，通常这类控制板的耗电量都不会太高，电源供应器只要有 1A 以上的电流量就可以了，但是如果考虑到其他附加板的耗电时，+5V 端若有 3A 的供电能力就绰绰有余了。如果你的试验中还牵涉到 RELAY（继电器）的控制时，还必须有另一组+12V 的电源供应端，专门给这些 RELAY 来用。那么为什么不直接选用也是+5V 规格的 RELAY 呢？这是有原因的，因为 RELAY 在跳开切离时，不可避免地会产生相当大的反电动势，必然会对+5V 的电源造成某种程度的影响，这时就会干扰到 8051 动作的稳定性，而这正是一个好的控制系统应避免的。

市面上交换式的电源供应器的价格应该在千元以内，图 1-3 是我们试制的电源供应器，利用两颗 NS（National Semiconductor）专门的 Switcher IC，可以产生 5V（3A）及 12V（1A）的输出，较特别的是其电压的输入范围为 15V~40V 皆可，并可接受交流或直流电池输入，且其转换效率都在 70%以上，比一般的 7805/7812 好用。如果你是用一般的传统模拟电源供应器时，请先打开该电源至少等待 1 秒钟，这时输出端的电压已经稳定在设置值了，然后才将电源接到 8051 控制板上。这种做法就是要避免刚打开的瞬间突然损坏了控制板上的某些组件。除此之外我们也要改掉对电源瞬间开开关关的习惯，因为许多高价位的数字零件往往无法承受这种电压极度不稳定的考验，有些重要的控制器在打开电源后，至少要停个 5 秒钟等电源供应器的输出全部稳定后，方才接通主电源给控制板，主要的用意就是要防止电源不稳定的情况发生。

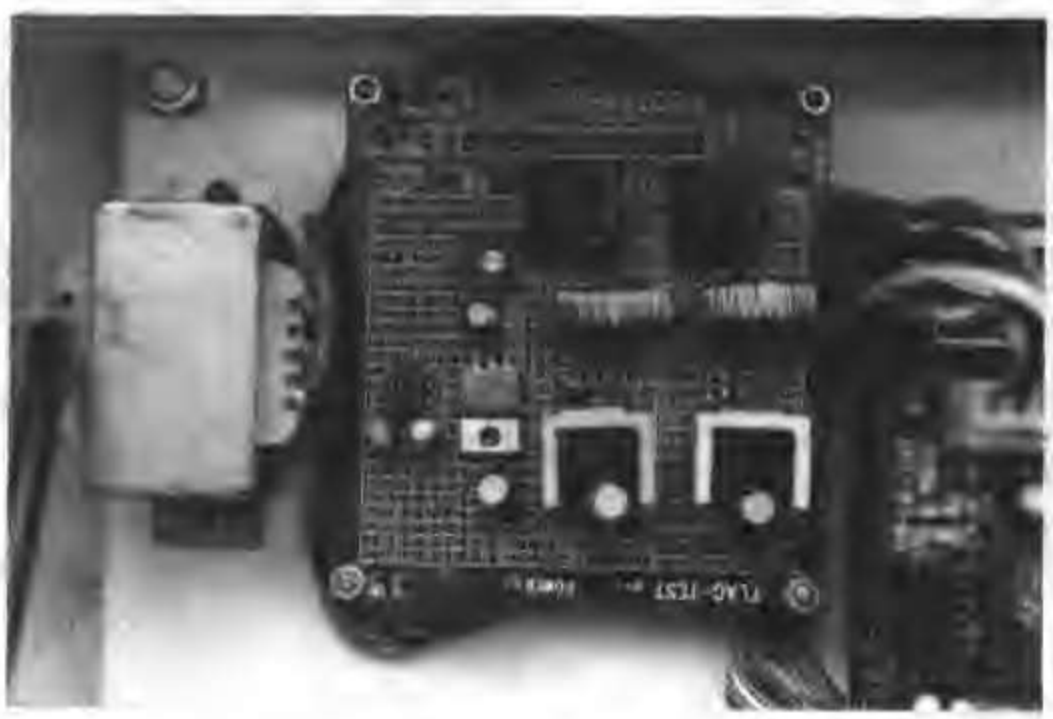


图 1-3

注：该电源供应器有+5V 及+12V 输出，可直接用 24V 或 48V 的电池当电源。

一台稳定但不一定是最顶级的计算机也是学习 8051 的必备工具，一般只要在 DOS 的环境下就可以开始撰写 8051 的汇编程序了。接着使用 8051 的汇编程序将我们所写的程序转成 8051 专用的机器码（machine code），再将这些数据下载到 8051 控制板上，直接执行验证我们所写的程序是否正确。

撰写程序最初一定是拿别人写好的程序下载试试看，然后尝试做细节的修改，改了部分的写法后，一定要立刻进行确认，如果动作不对的话，一定要找出其中的差异来。说实在的，写程序就是不断的尝试错误，到了最后才是整个程序的大修改。这段学习的时间我们认为少则半年多则数年，视你倾注的心血而定。

图 1-4 为市面上现成的交换式电源供应器，应选有+5V 及+12V 输出且通过电磁安全认证的为宜。

图 1-4

图 1-5 为购买现成的交换式电源供应器，应选通过质量安全认证的为宜。

图 1-5

图 1-6 是数字示波器，设计工程师的最爱。

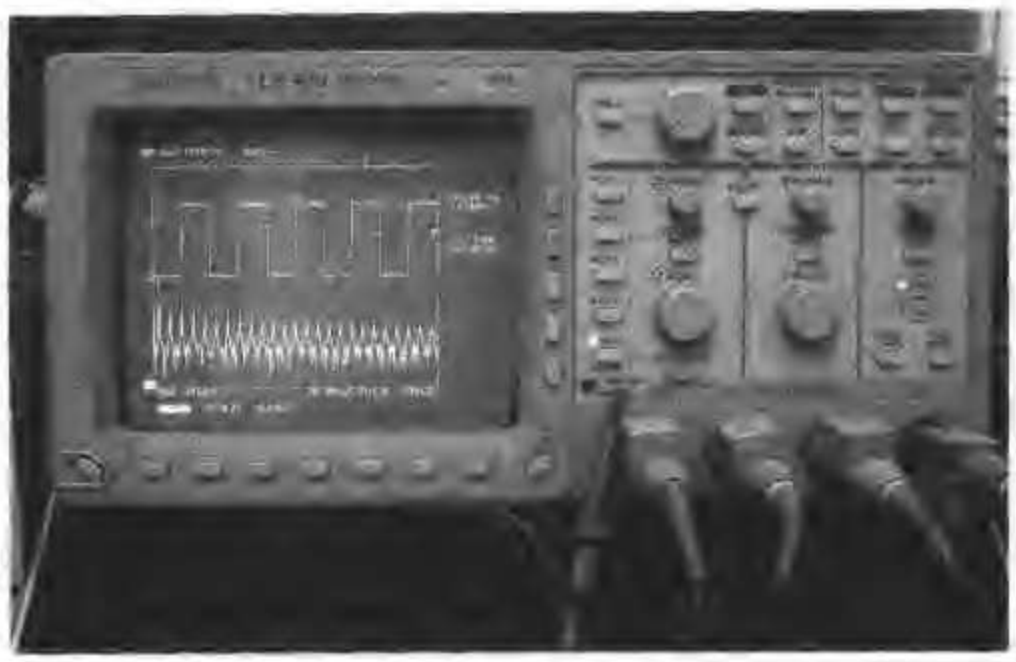


图 1-6

1-5 额外的辅助工具：示波器

要想进入 8051 的控制与设计之门，除了以上所谈的软硬设备之外，我们强烈建议初学者若还有部分研究经费时，可以再添购一台示波器，绝对可以帮上许多硬件上排错的忙，倘若经费真的有限时，那就至少买个逻辑笔（Logic probe）。

在学习 8051 单片机的软硬件设计时，有许多信号是必须查看的，例如：

（1）XTAL 石英晶体的振荡信号，是模拟方波还是正弦波？

（2）ALE 的信号是振荡信号的 1/6，但是有时候会突然失踪，你知道原因吗？

（3）8051 的串行通信有许多模式，这些模式所送出的信号的样子如何？

（4）8051 可支持 RS485 的通信模式，这些信号的转换准位如何区分？

（5）地址译码信号如何分辨？如果你是用可程序化组件来译码时，一定要用示波器来看其中的变化。

一台好用的示波器可以帮你解决许多和硬件有关的问题，而且新款的示波器都可以外接打印机，能打印出所要的波形，如图 1-7 所示。

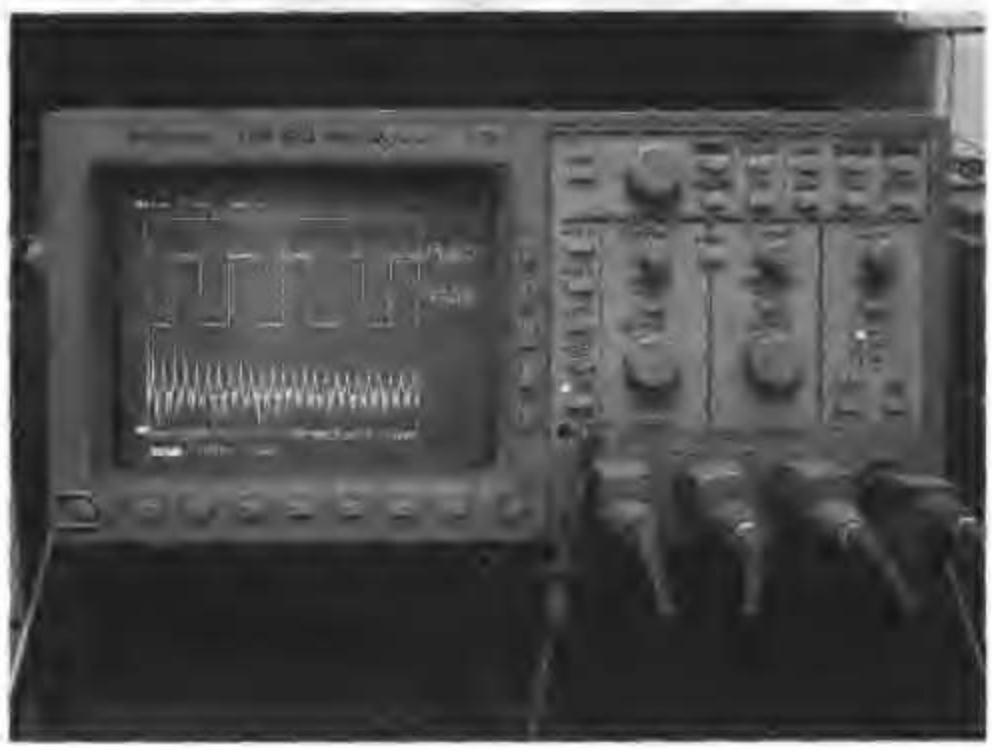


图 1-7

CH1：单片机 8051 的振荡器上的信号波形；CH2：ALE 信号波形，如图 1-8 所示。

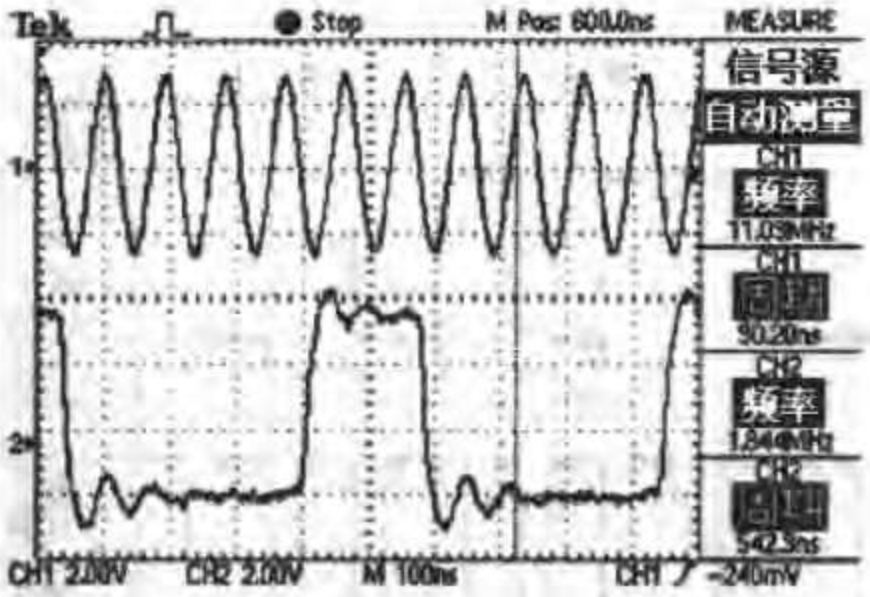


图 1-8

图 1-9 是单片机 8051 的 ALE 信号，我们可由图的右方看到信号频率值。

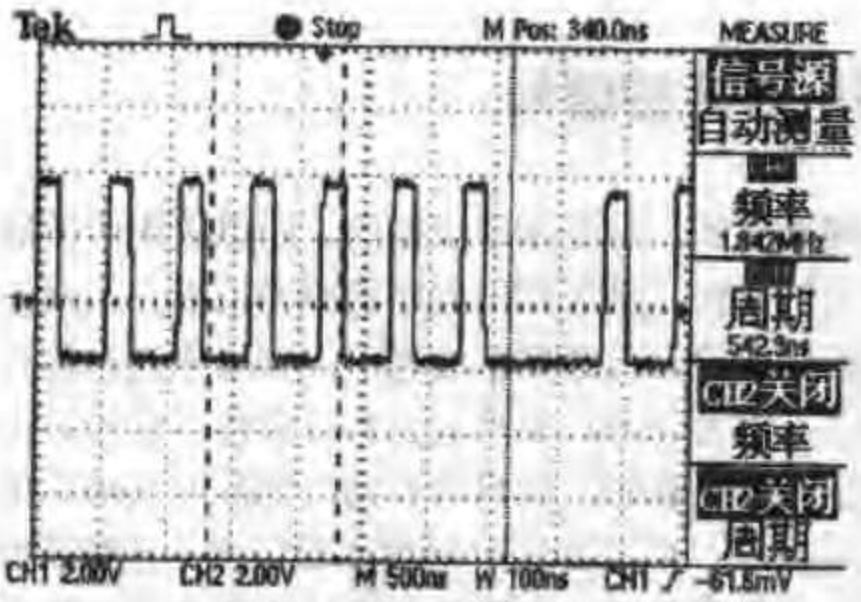


图 1-9

单片机 8051 的串行信号，以前的程序设计师碰到通信方面的问题时，往往要靠不停的循环设计，才能隐约看到这些信号，如图 1-10 所示。

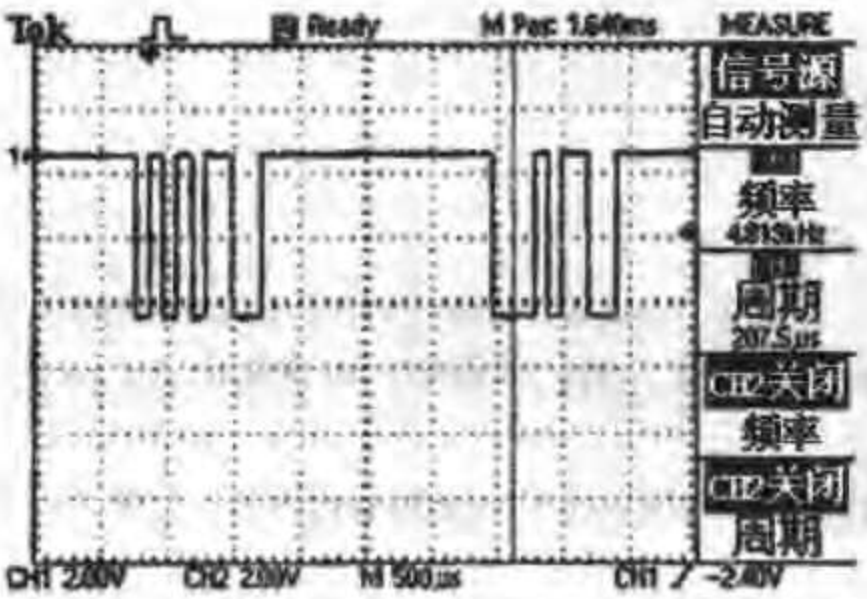


图 1-10

单片机 8051 的地址（A7-A0）与数据线（D7-D0）是共享的，使用数字示波器可以很清楚地看出其中的变化，如图 1-11 所示。

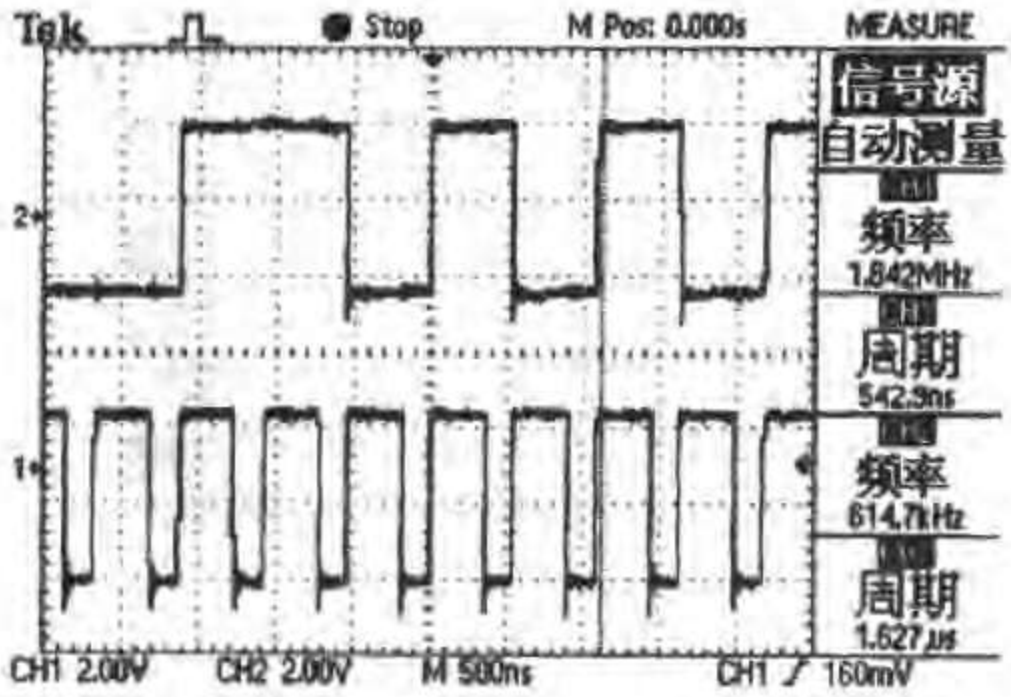


图 1-11

1-6 A/D 转换实验时各种信号

以上的信号有些是纯数字的，有些则是模拟的，所以选用示波器来直接观察其中的波形才能让我们分辨波形信号的真伪，进而查出程序的不当控制点。示波器分成传统的模拟式与数字式两种，这两种示波器的价格已经非常接近了，如果你尚未拥有示波器时，我们建议你直接购买数字式的示波器，因为 8051 单片机上有许多数字信号必需借助储存的功能才能分辨，如果使用模拟式的示波器查看时，就只能看到一团乱七八糟的信号了。不过使用数字式示波器需要一到两周的适应期，而且刚刚开始使用时很难一次就抓到要看的波形信号。

综合以上我们所提的工具及软件，只要你有或是借到部分工具后，就可以着手学习 8051 的各种设计了，我们觉得应该先从指令的学习与小程序的撰写开始，然后再来学习扩充硬件系统，最后才是 8051 软硬件的合并使用。在学习的路上你将有一大段路要走，但是请记住所有的“大内高手”都历经这个艰涩无法突破的阶段，没有人能够不学习走路就直接用飞的，这也包括“旗威科技”的工程师在内。

1-7 本章使用软件

本章使用的软件如下：

（1）8051 数据表：直接从 Intel 或 Atmel 网站上获取完整的说明。

（2）Acrobat Reader：PDF 文件的阅读程序，可直接由 Adobe 的网站上免费获取最新的版本。

（3）8051 FAQ：请自 http://www.ece.orst.edu/serv/8051 上直接下载，所有 8051 的文件及常见问题解答都在这里。

（4）8051 Assembler：请购买商用软件 X8051/EP51 或从网上下载共享软件。

（5）8051 Dis-assembler：专业的 8051 反汇编程序，直接到“旗威科技”的网站上下载该共享软件，只要是 32KB 以内的程序空间都适用。

（6）MS-DOS 操作系统或 Windows 的 DOS 模式。

1-8　本章使用硬件

本章使用的硬件如下：

（1）FLAG51 控制板：内含 32KB EPROM 及 8KB SRAM，最少有一个 8255 的可编程输出/入端口，可以做实际外部的输出/入用。

（2）I/O 监视器：查看 8255 的输出/输入信号。

（3）光隔离输入板：外接信号输入用。

（4）RELAY 输出板：送出控制信号给外部的设备。

（5）数字示波器：Tektronix TDS220/TDS420 观察 8051 的各种信号。

（6）交换式电源供应器：可输出+5V 与+12V 的电压，若要做模拟 OP 的实验时，最好把-12V 的输出也考虑进去。

（7）数字电表：FLUKE 87 或 FLUKE 187/189。

（8）焊接烙铁及焊锡枪。

1-9　相关信息网站

你可经由下列公司、网站获取更进--步的信息：

http://www.chipware.com.tw：获取单片机控制板相关信息。

http://www.rs232.com.tw：查询各种电子零组件信息。

http://www.google.com：用“8051FAQ”关键词去查询有关的信息。

http://www.Intel.com：8051 或 80251 信息。

http://www.atmel.com：AT89C 系列单片机数据。

http://www.adobe.com：下载 Acroread PDF 文件阅读程序。

http://www.iar.com：8051 C 及 Assembler 软件商。

第 2 章

单片机相关仪器设备的认识与使用

写 8051 的程序需要仪器的配合吗？听听 8051 专家的话：那是一定要的！当你碰到 bug 无法克服障碍时，可能一台数字示波器就会让 bug 原形毕露。当你设计的产品无法通过电磁安全检测规定的静电测试时，8051 的专家提醒你光会在计算机前写程序是绝对不够的。

你可能不知道的事

◆ 许多高级车上的喷射引擎控制器广泛使用 8051 系列的单片机控制器。

◆ 全世界有许多工业用的控制器全面使用 8051，作为其众多控制节点（Node）中的一个控制点。

◆ 许多扫描仪、条形码扫描器及刷卡机都是 8051 的应用实例。

◆ “旗威”网站是专业性相当高的技术交流网站。

◆ “旗威”的 8051 温湿度控制器已经成功地用在国内某知名品牌的防潮箱上。

◆ 我们使用 8051 单片机去做工业用的控制器，其性能与方便性一点都不比进口的设备或 PLC 还差。

◆ “旗威”的单片机控制板已经成功用在 GPS 卫星定位系统上。

◆ “旗威”的单片机控制板的使用人口中，在校学生及教师约为三成，其余都是在职的专业工程师。

曾经有人向“旗威科技公司”反映，他用的 8051 单片机控制板经常无法顺利开机，成功的机率只占一半而已，换了好几块单片机控制板后，异常的情况依旧存在，这是否意味着我们的控制板有某些盲点尚未解决，才导致这个问题的产生。我们只提了几个问题，就确定了问题所在。

第 1 个问题是：使用何种电源？

这位仁兄回答是一般的交换式电源，所以没有电源突波的情况发生。

第 2 个问题是：程序在重启（RESET）后，首先做的是那一件事？

他毫不犹豫地说：“当然是设置 8255 的控制模式”，我们马上接着说：“问题就出在这里”。

8051 在 RESET 后绝对不可以立即进行其他外围的控制设置，这一点我们已经在许多场合上提过许多次了。因为该 RESET 信号对 8051 来讲可能已经结束了，可是对其他外围而言，

电源的电压尚未升到 4.5V 以上，可能认定 RESET 信号仍然还存在，所以你此时送出的一连串控制信号，对这些外围 IC 而言，可能都是无效的！因此才造成有些时候可以动作，有些时候又动不了的窘态。这时如果使用者马上关闭电源又随即开启时，这种瞬间的开关开三个动作，对单片机控制板而言，供应电源并未实际中断，仅仅多一个 RESET 信号出现，所以这次 RESET 后的外围设置又变成有效的，机器又可以正常运作了。类似这种问题我们自己都会记得避开，可是如果使用者没有经常阅读相关的书籍或杂志，就会不自觉地掉到这个陷阱中。也许这点您还不知道，不过今后要千万记得，我们打算在网站上另外设一个 8051 的医疗救护专栏，如果您有 8051 方面的困扰时，请直接上网（http://www.chipware.com.tw）挂号吧！

2-1 数字电表的认识与使用

当我们第一次试验任何电路板时，一定先用数字电表检查每个 IC 的电源脚上是否得到正确的电源供应，这是一个非常值得学习的好习惯，因为第一道关卡若没有过关的话，其余都是免谈的。在 8051 单片机的开发阶段，我们也经常用数字电表来监视 RELAY 的动作与否。当 12V 规格的 REALY 不动作时，其线圈两端的电压差几乎是 12V 左右；动作时，则两端电压差就会降低到 1V 以内。数字电表所监视的电压或电流值，其变化速度都不能过快，否则将无法读取正确值。

在线路板的维修方面，我们也经常用数字电表来追查线路是否断线或短路，电路的断线通常是 PC 板在制作时所造成的，一定要加焊或补焊额外线路才行。可是短路的情况就很难去处理了，有些是外加的零件本身短路造成的，也有可能是 PC 板在加工过锡炉时，多余的锡渣所产生的影响。最差的情况是 PC 板在制作压合时，“不小心”造成的短路，这些状况都可以用一般数字电表的欧姆档检查出来。通常我们一发现线路有不正常短路时，首先查看相关的线路及焊点是否有短路，至于零件短路的情况也顺便列入检查，最后才是怀疑 PC 板本身有所短路的情况。

单片机控制板如图 2-1 所示，在程序设计师的手上可以是成功的典范，也有可能是失败的明镜。

图 2-1

数字电表如图 2-2 所示是数字设计工程师的最贴身工具。

图 2-2

数字电表最常用的档位为 DCV 与欧姆档，通常有四或五位数字的显示，如图 2-3 所示，一般应用应该都足够。这类电表都是随身携带的，所以特别要选择耐用耐摔型。

图 2-3

如果要做精密的模拟与数字转换时，数字电表就要选择桌上型的，精度可达五位半或六位半的数字显示，如图 2-4 所示。

图 2-4

2-2 示波器的认识与使用

学习或研究 8051 单片机控制器时，我们身边一定有一台好的示波器，因为这方面的程序一定跟硬件或外部的线路有关。如果不用示波器辅助查看信号时，程序再怎么修正都看不出来其中的差异性来。我们的经验是示波器可以帮忙解决七到八成的硬件问题，剩下的问题就只有靠软件修正或是更精密的电子仪器来协助了。为何要使用示波器来做 8051 的实验呢？我们最常举的例子是研究 8051 的串行通信。

8051 单片机最让人赞赏的就是内含一组全双工的串行通信端口，只要设置好通信的状态值及中断后，就可以从容地使用 SBUF 去通信，可是你看过 8051 的送出的波形吗？如果还要写 PC 端的串行联机程序时，该怎么做会最快呢？我们是借用示波器的波形储存能力，先确定发送端是正确的，进而确认接受端已获得正确的串行值，我们的实际程序撰写经验是这样的：

（1）先确定 8051 单片机会送出一个正确的串行值。比方说是 ASCII 码的'1'，其十六进制值为 31H，选择此码的原因是内有 0 与 1 的变化，可让我们用示波器轻易地辨别其与 Start bit/Stop bit 间的差异。在这个阶段我们要下对 8051 的状态句柄，才能顺利地传出该串行码来。

（2）在 PC 端写一个简单的 RS232 接收程序，接受 8051 送来的串行数据，并且将接收值放在屏幕上。若 8051 送串行数据的速度为一秒一次时，我们应该在画面上看到'1'的字样以每秒一个字的速度出现在屏幕上。如果没办法这样的话，一定是我们在 PC 端的接收程序有问题了，因为我们先前已经确认 8051 能无误地送出串行值来。

图 2-5 为 TDS220 100MHz 储存式示波器。

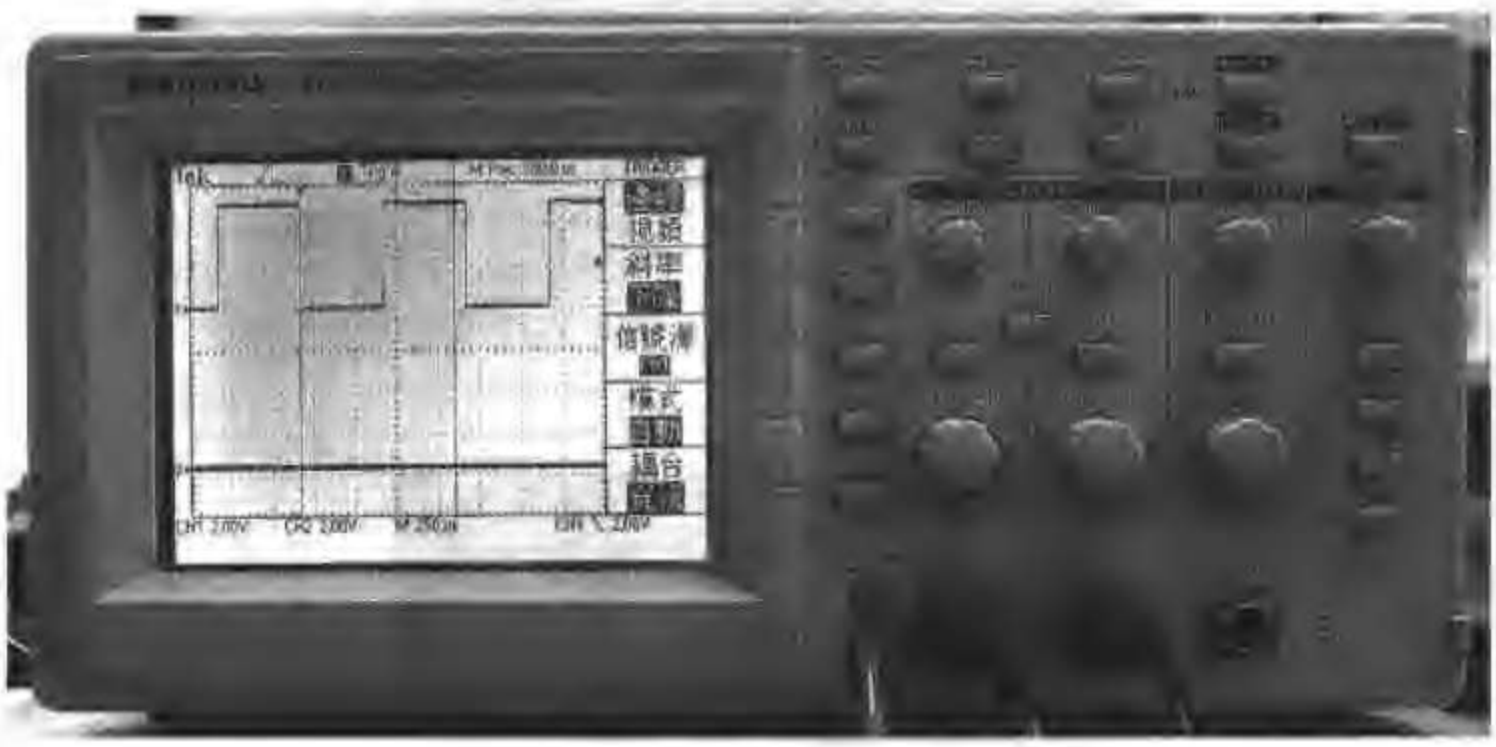


图 2-5

TDS220 示波器有 2 CH 的信号输入，并且波形显示板的右边有功能显示，如图 2-6 所示。

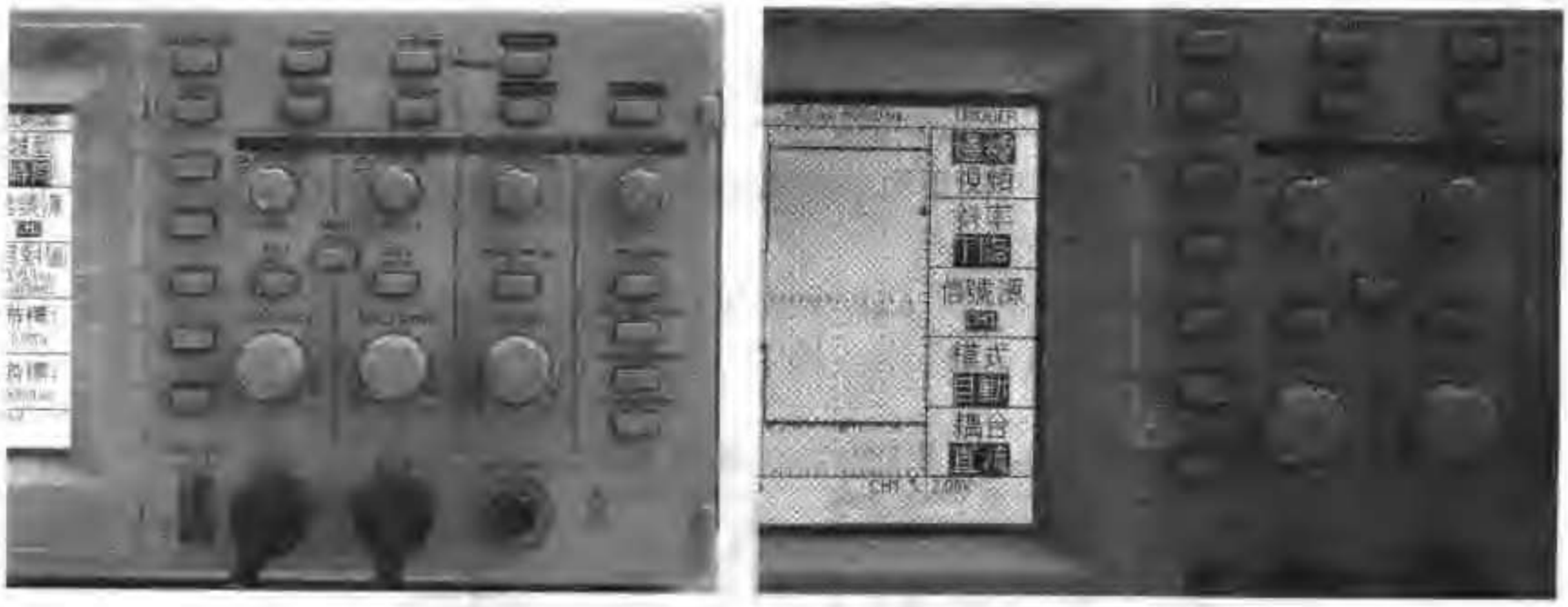


图 2-6

TDS220 背部可选购打印机输出接口，用测试棒抵住待测点即可测量，如图 2-7 所示。

图 2-7

（3）在 PC 端写一个简单的字符送出程序，一次只送出一个字符，传送的速度可以保持一秒钟一次，程序写好后也是用示波器观看 COM 端口送出来的信号。如果您曾经看过类似的信号，应该知道经过 RS232 的准位转换后，串行信号的振幅已经放大到正负 12V 左右，但是实际的传输信号也被反相了。

（4）在 8051 控制板上写一个接受串行数据的小程序，设置完状态值后就进入一个无穷的循环，只要察觉 RI 位被设成 1 时，就执行 MOV A，SBUF 的指令，然后将该值顺便转送到 LED 端口上，只要传输的速度不是很快时，我们一定会从 LED 上看出其中的变化来。

（5）当双方的简易接受发送程序都没有问题时，就可以把串行中断的服务程序加入，让串行的接送转变成背景程序来处理，而主程序还是处理原来的事情。如果程序完成到这个阶段时，就纯粹是程序上的问题了，而需要示波器的机会会越来越少。

（6）程序更深入进行的阶段，有时候我们需要 8051 送出一个状态值，可是此时系统的 LED 端口及画面显示部分都无法挪做除错用时，我们还是可以把 8051 的串行端口当成该状态值的专用输出端口来用。只要我们用数字式的示波器，就可以很清楚地看出 8051 所送回的

串行状态值了。

（7）程序更深入进行的阶段，有时候我们需要 8051 送出一个状态值，可是此时系统的 LED 端口及画面显示部分都无法挪做除错用时，我们还是可以把 8051 的串行端口当成该状态值的专用输出端口来用。只要我们用数字式的示波器，就可以很清楚地看出 8051 所送回的串行状态值了。

CH1 是 8051 送出来的串行信号，CH2 则是硬件产生的 one-shot 波形。使用储存式示波器可以锁住要看的波形后，再做更进一步的分析，如图 2-8 所示。

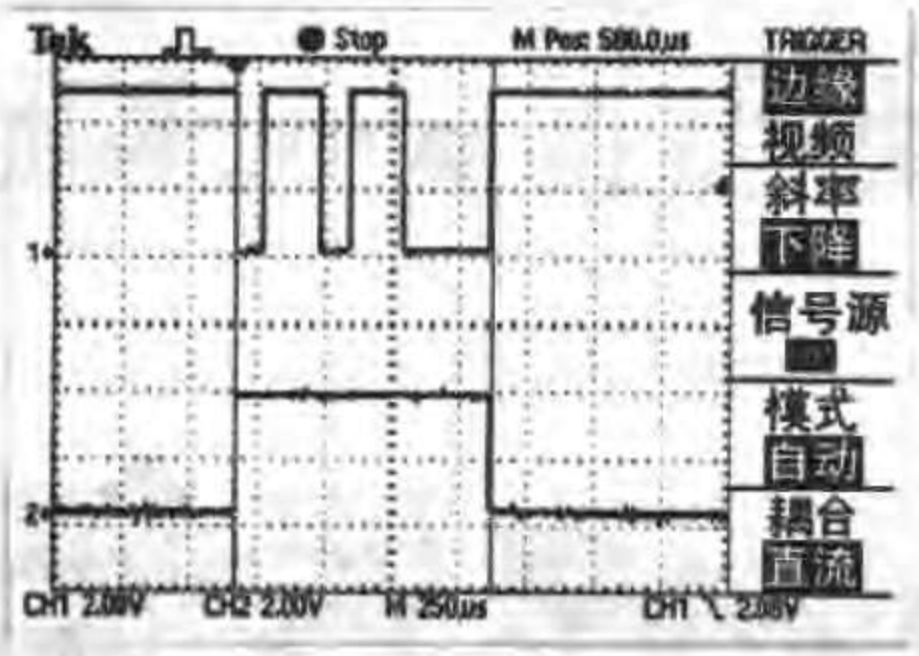


图 2-8

CH1 也是串行信号，请看右边的时间测量值，光标 12 间的时间差为 950µs，由此我们可以断定串行传输的速度应该是 9600bit/s，如图 2-9 所示。

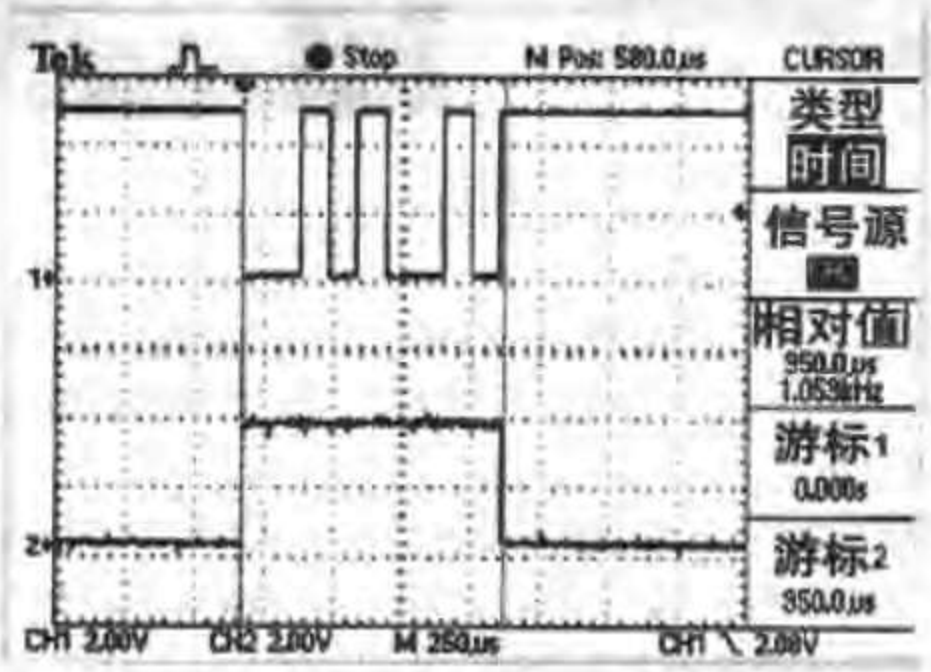


图 2-9

2-3 ATMEL AT89CXX 刻录器的使用

依据我们的统计，现在约有八成以上的大专电子科系都改用 AT89C51 或 AT89C52 来做 8051 单片机的各项实验，因为 ATMEL 的 Flash Microcontroller 可以在线直接擦除，而且价位

比起 Intel 的 8751/8752 都还来得便宜。要使用 AT89CXX 一定要有专用的刻录器将程序数据刻录到 AT89CXX 的微控制器上。现在可在市面上看到的 AT89CXX 刻录器统归有 3 种：一是万用刻录器型的，这类刻录器的接脚有相当的灵活性，所以只要 ATEML 公司一有刻录数据公布时，刻录程序重新定义各脚的刻录电压后，就能直接刻录 AT89CXX 类的 IC。第二类的刻录器是接在 PC 的打印口的专用刻录器，其体积都不会很大，借用 PC 丰富的资源及运算能力，能很轻松就完成刻录的动作。第三类就是属 8051 的应用实例了。以“旗威”的 AT89CXX 刻录器而言，主要的使用对象是喜欢 DIY 的人士，它要结合 8051 单片机控制板，并且要更换 EPROM 程序及 PEEL 组件后，才会成为一台专用的 AT89CXX 刻录器。这块刻录控制板最近几年始终处于热卖阶段，是我们始料未及的。

AT89CXX 刻录板如图 2-10 所示，该刻录板的发展从最早只能刻录 AT89C51，到能够支援 20PIN 的 AT89C2051 以及 AT89C1051，“旗威科技”在此方面是最为显著的。

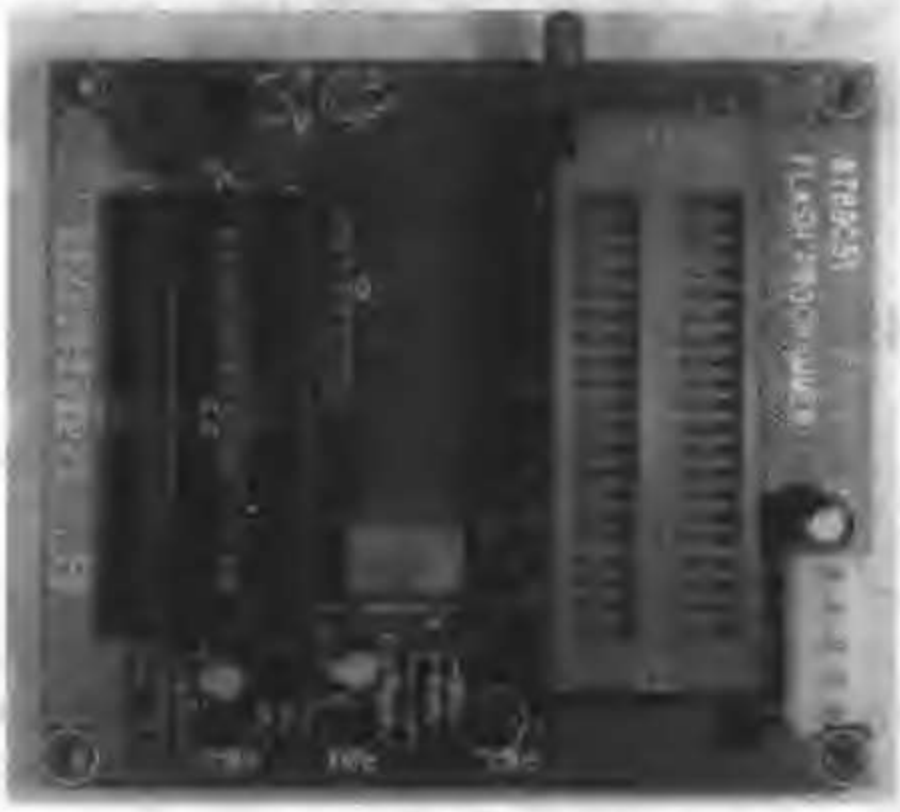


图 2-10

在原有的测试夹上再加上转换板即可刻录 AT89C2051 和 AT89C1051，如图 2-11 所示。

图 2-11

刻录 AT89C2051 转换板上的按键特写，如图 2-12 所示，右键为 PROGRAM 刻录键，左键为 READ 数据读取键。

图 2-12

2-4 EPROM 刻录器的使用

在 PC 机刚盛行时，许多 PC 机硬件杂志都会提到如何制作 EPROM（可擦写可编程只读存储器）的刻录器，因为那时候的 EPROM 刻录器非常贵，除非是工厂要大量刻录 EPROM，要不然一般的玩家根本买不下手。但是随着刻录数据的逐渐流通，以及有心人对大厂刻录器的电路追踪及研究，国内早在 10 年前就有许多相当出众的 EPROM 刻录器出现，此类的刻录器特别强调的是高速对拷的功能，以便使用者能在短时间内就完成数据的复制，但是刻录速度的提高仅有少部分是软件程序的改善，重要的是 EPROM 制造厂商对于内部制程及规格加以改善，才得以有快速刻录的 EPROM 出现，如果这些制程不做修正，有谁会买一个要整整刻录两分钟的 27256 或 27512 呢？

EPROM 刻录器的使用是很简单的，把待拷贝的母 EPROM 放上去，按下读取数据的按键，然后把一颗空白 EPROM 放到刻录座上，再按下刻录键进行数据的刻录，十数秒钟后就完成一颗 EPROM 的刻录。

如果您是要刻录刚刚写好的程序数据时，就需要一台 PC 将程序代码转变成十六进制文件或是纯机器码文件，然后将通过下传程序将刻录数据传到 EPROM 刻录器的内存中，接着才进行刻录。现在市面上有许多 PC 机专用的万用刻录器是在主机板上加挂一块刻录电压控制电路，就能刻录上千种 EPROM 及可程序化组件。

如果您有足够的经费来源时，当然是买万用刻录器较划算。可是若只纯刻录 EPROM 几款 IC 时，还是可以购买基本型的 EPROM 刻录器。“旗威科技”在 8051 的应用上也有推出类似的 EPROM 刻录器，我们最近正在对其中的程序进行更新，以便能够刻录 64KB 的 EEPROM，这样使用者就能够直接运用 Erase（擦除）指令直接清除内部的数据，而不需要再靠紫外灯的清除。如果您需要试一下这个程序时，请由“旗威”的网站上获取该测试软件。

“旗威”的 EPROM 刻录板需配合单片机控制板，就变成 27256 / 27512 的专用刻录器，

如图 2-13 所示。

图 2-13

图 2-14 是 EPROM 刻录器修改部分的电压输出及程序后就可以刻录 512KB 的 EEPROM，详细情形请看本书的稍后章节。

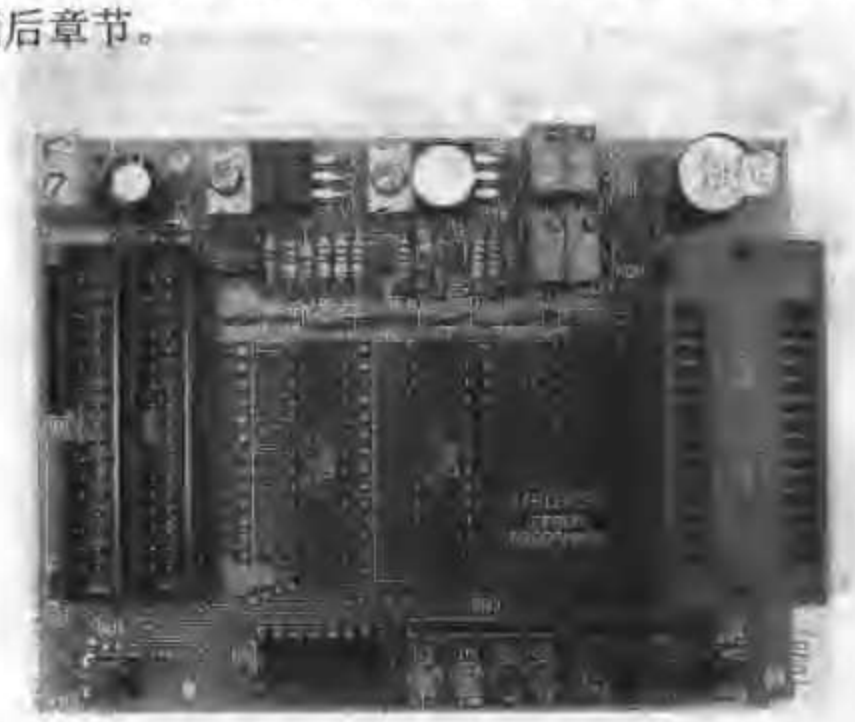


图 2-14

2-5　逻辑分析仪的认识

当你的单片机系统运作起来不是很顺畅的时候，请先检查是否软件在做怪，因为这种机率通常高达九成以上。如果用了各种软件验证方法都无法确认程序的问题点时，就要靠逻辑分析仪（Logic Analyzer，简称 LA）来做进一步的检查了。

逻辑分析仪会在设置触发的条件成立后，一边记录程序代码的变化，也同时记录 I/O 点上的变化情况，直到整个记忆缓冲器满了才停止。接下来就是我们逐一分析逻辑分析仪上记录数据了，如果我们对自己的控制系统愈了解时，就愈能缩小记录的对象，进而找出真正的问题点来。在电路设计上，用到逻辑分析仪的机会实在不多，但是有些时候用逻辑分析仪来看数字的真实动作，尤其是捕捉计数器瞬间的 Glitch（干扰）信号，确实是相当方便的。

图 2-15 是“旗威科技”公司所使用的 LA（逻辑分析仪），现在新款的 LA 都是基于 Windows

的，操作方法更简单，但是工作原理都是相同的。

图 2-15

逻辑分析仪操作上最为重要的是触发点的设置，设置对了才能获取我们想要的波形（Waveform）及时序（Timing），如图 2-16 所示。

图 2-16

图 2-17 是一个异步计数器所送出的波形，由逻辑分析仪观看时，发现最高位上有明显的 Glitch 出现，这类的信号很难在示波器上看到。

图 2-17

2-6　逻辑分析仪使用实例

“旗威”公司曾经刻录了 20 个 PEEL18CV8（速度 25ns）可编程组件给温湿度控制器用，但是有部分组件的刻录数据经过比对后完全正确，可是却导致 TH2030 温湿度显示器不启动。之前我们已经用示波器看出解码输出已不动作，但我们想用 LA 来看一下其中的实际波形，是地址出现的时机不对导致 18CV8 无法输出，还是 IC 故障而使输出不动作。

图 2-18 及图 2-19 分别是 TH2030 在 8000H 与 9000H 所送出的读回与写入信号波形，我们可以看出 AT89C52 的 Read 与 Write 脉波都有 500 ns 以上的宽度，而且地址线早就稳定了，所以可以断定应该是 PEEL 组件不良而导致系统的不能开机，解决的方法是换一个新的 PEEL 就行了。虽然类似的问题用示波器就可以找出故障点来，但是通过 LA 观看波形时，却能同时看到几十条线（BUS）的变化情形，进而确定所有信号的对与错，而一般的示波器则只能观看 2ch 到 4ch，对复杂的数字信号则帮助有限。

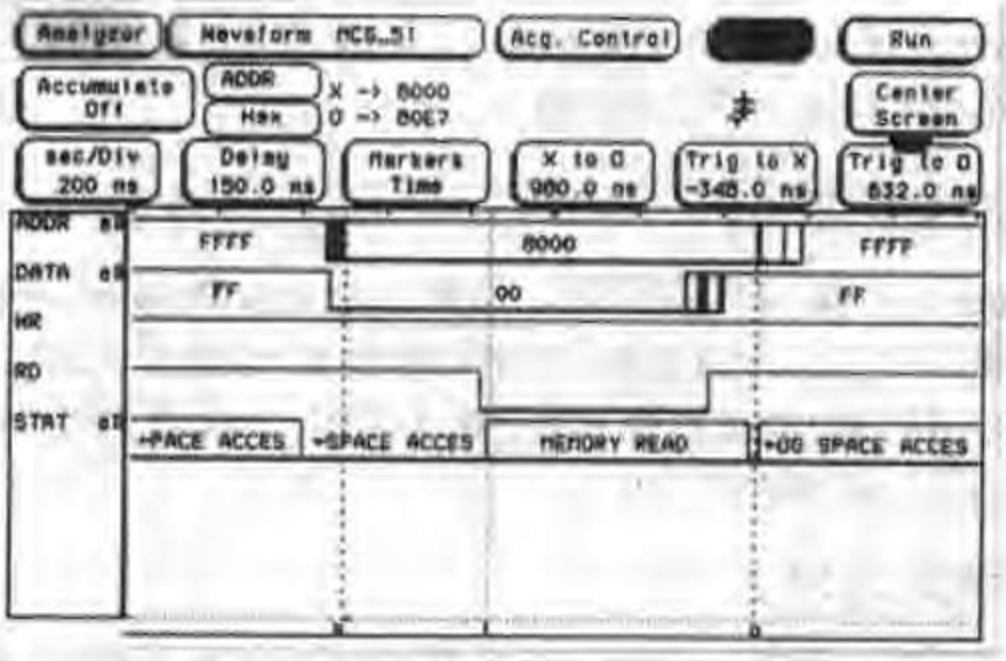


图 2-18

注：8051 对地址 8000H 所送出的读取周期，其 RD 信号宽度绝对是足够的。

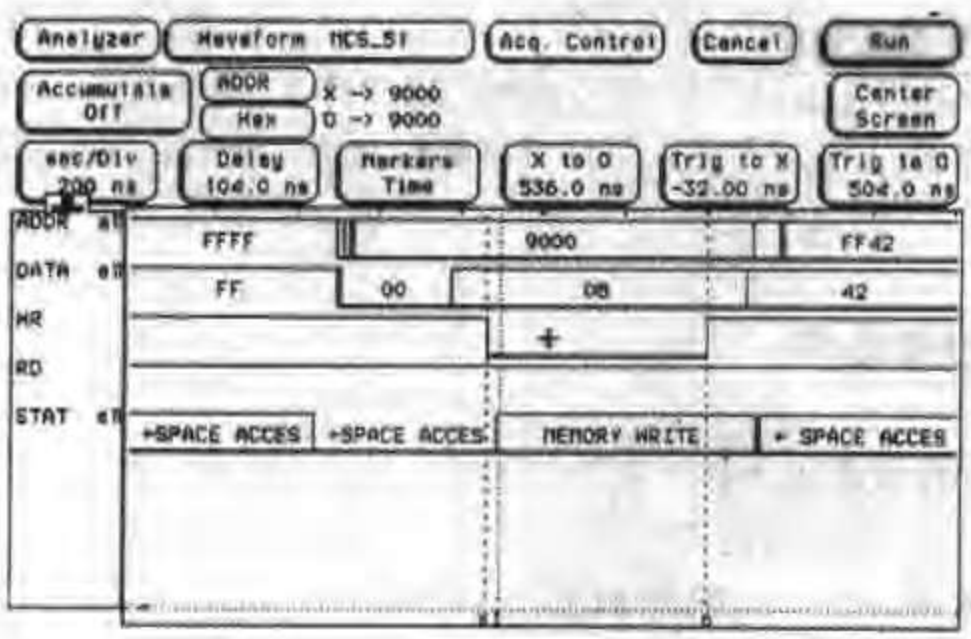


图 2-19

注：8051 对地址 9000H 所送出的写入周期，WR 信号也非常完整。

2-7 必要的相关信息及常识

学习单片机 8051 绝对不是只在实验室或工作桌上埋头苦干地写程序而已，因为这种写法的不可靠性太高了。若是学校的专题时，只要会动作就行了。可是如果是一项正准备上市的产品时，我们强烈建议您多找一些人去做软硬件方面的测试，否则推出后要再修改就要大费周章了。我们认为身为一个单片机的研究或开发者，你必须要再熟悉以下几个课题，这些都会对你的设计与规划图上产生绝对的影响力。

◆ ISO-9000 的质量认证

国内已经有许多工厂陆续获取 9001 或 9002 的质量认证，这对我们的设计有绝对的影响，而且层面是相当广的。

◆ CE 相关的法规与测试步骤

外销欧洲的产品必须有欧盟 CE 认证标志才行，亚洲许多国家也有类似的要求，我们可以从这一部分获得电磁干扰防制的许多技巧。

◆ ISP 方面 IC 的使用

这可不是 Internet 上的服务提供商简写，而是 In System Programming 的缩写，即“在线刻录”或“在线下载”。近几年来有愈来愈多的可编程 IC 强调有“ISP 在线刻录”的功能，这也就是说不需要万用刻录器就能直接下载刻录这些 IC。如果我们猜得没错的话，再过几年后刻录器的市场将会风云大变的。

◆ VHDL 的学习与导入

早期数字电路为了防止别人的恶意拷贝，或是为了简化数字电路，都会在线路中加入一个以上的可编程组件，例如 PAL、GAL 或 PEEL 组件等等。但是随着数字电路的高度复杂化，上述组件逐渐无法胜任了。取而代之的是更高密度的 CPLD（Complex PLD）或是 FPGA（Field Programmable Gate Array，可编程逻辑器件），要进入这方面的门槛是相当高的，除了要有原版的软件包外，还要努力地看其中的英文说明，而 VHDL 则是其中一种规划 FPGA 的方法，它较我们所写的 PLD 程序更为严谨，而且可移植性更高，我们认为这将是未来数字电路设计的新趋势。

2-8 本章使用的硬件

本章所使用的硬件设备如下：

（1）FLAG51 单片机控制板 。

（2）AT89CXX 刻录板：须配合 FLAG51 使用。

（3）EPROM 刻录板：须配合 FLAG51 使用。

（4）EEPROM 刻录板：须配合 FLAG51 使用。

（5）Fluke 87 Multimeter：四位半数字电表。

（6）HP3478 Multimeter：五位半数字电表。

（7）Tektronix TDS220：100MHz 数字示波器。

（8）HP 1662A LOGIC ANALYZER 逻辑分析仪：可做 100MHz 的状态分析与 500MHz 时序分析。

2-9　相关信息网站

您可经由下列公司、网站获取更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.intel.com：查询 8051 相关信息。

http://www.atmel.com：查询 AT89C 系列 Flash 微控制器信息。

http://www.fluke.com：查询高级数字电表信息。

http://www.tek.com：查询示波器及仪器信息。

http://www.agilent.com：查询数字电表及逻辑分析仪信息。

http://www.altera.com：查询 CPLD/FPGA 信息。

http://www.latticesemi.com：查询 CPLD 信息。

http://www.xilinx.com：查询 CPLD/FPGA 信息。

http://www.etc.org.cn：查询各种质量认证申请及 ISO9000 信息。

第 3 章

试写两个 8051 范例程序

> “新手上路请注意安全”。初次写汇编语言时也是如此。写汇编语言首先讲求的是步步为营，先把程序的主要结构写好，然后再进行细节的分解。汇编语言能学得好的话，再学习其他语言都能得心应手。

3-1 写汇编语言需要有条不紊的思考能力

写汇编程序是一项兼具挑战性及创作性的工作，有人在学习阶段初期就宣告放弃了，也有人在交出汇编语言的作业后，对自己讲以后再也不碰这方面的书了。当然也有人屡败屡战，一定要找出程序的 bug 才罢休，如果您正好是最后这种类型的人，绝对有可能成为最佳的程序开发者。不过，想要到达顶尖的地步还是要用点心力才行。写程序首要的条件是细心与步步为营，细心的人写程序的时间一定比粗枝大叶的人要来得短；能够步步为营的人则很容易过滤程序故障的原因。当你养成这些写程序的好习惯后，做起任何事情时都会更为完备，可以尽量避免错误情事的发生。

写 8051 的汇编语言所需要的工具有：Windows 版的 DOS 模式或 DOS 版本的个人计算机、8051 的 Assembler 汇编程序、一台以 8051 CPU 为主的控制板以及相关的刻录器。如果您有 FLAG51 单片机控制板，还可以将待试验的程序下载到线路板的 SRAM 上直接进行测试。这种学习的方法有相当大的好处，因为验证程序在帮大忙，免除了许多软件设置的问题，程序执行停止后顶多重新开启电源面已。可是，这种学习方法还是漏掉有些 8051 的精华，例如开机之后如何完成相关缓存器的设置，如何进行正确的输出与输入，如何开始管理内存等等。所以我们本章的程序示范将不借助 FLAG51 控制板，而是直接将程序刻录到与 8051 程序代码完全兼容的 AT89C2051 上，进行所有软硬件的测试。AT89C2051 只有 20 根脚，内部除了电源，石英振荡输入及 RESET 脚外，其余 15 根脚都可以拿来作为 I/O 用。许多用过 AT89C2051 的程序设计人员，对于 ATMEL 这个缩小版的 8051 都非常有兴趣，除了它的电压范围（2.7V~6V）让人赞叹之外，其耗电量也是相当少的，用来设计一些消费性的产品绝对划得来，比起 16CXX 的 PIC 芯片，它还有一个优点：AT89C2051 内部有 2KB 的闪存，所以程序写错了没关系，可以用电气方法直接清除后重写，对经常要修改的产品而言，选用内含闪存的 AT89C2051 应该是正确的。

本实验所用到的硬件有：

1. FLAG 51 单片机控制板。
2. AT89CXX 刻录板。
3. AT89C2051 学习板。

注：

1. AT89CXX 刻录器，可刻录 40 根脚的 AT89C51/52 及 20 根脚的 AT89C2051/1051。
2. 以上三块板子均由“旗威”公司所开发。

可是 AT89C2051 需要刻录器才能将程序代码存入，一般常见的万用刻录器都可以支持，而且市面上也有一些较低价位的 AT89C2051 专用刻录器可购得。图 3-1 是我们所使用的刻录器，这是由 FLAG51 单片机控制板加上 AT89CXX 刻录板所组合成的，现阶段可以刻录 40 根脚的 AT89C51/AT89C52 以及 20 根脚的 AT89C2051/AT89C1051。AT89C2051 和 8051 的线路非常类似，只要加上 5V 电源，石英晶体以及 RESET 电容 1uF 就能运作了。我们在这里所做的示范，是将 AT89C2051 插入旗威的 AT89C2051 学习板上。图 3-2 是该学习板的电路图。板上是规划 P1 端口可做输出或输入用，P3 端口（少了 P3.6）的引脚准位都有 LED 做指示，比较特殊的是板上还有一个 RS485 的接口，这片 AT89C2051 学习板可以通过串行 RS485 接口与其他的控制板联机，在 PC 个人计算机上也可以通过 RS485 接口与多台 AT89C2051 学习板联机，达成一对多的控制模式。如果你要做到如此深度的应用，不先学会 8051 的汇编语言是行不通的，万丈高楼还不都是从平地盖起的。

图 3-1

注：AT89CXX 刻录器，可以刻录 AT89C2051/1051，也可以刻录 40 根脚的 AT89C51/89C52。

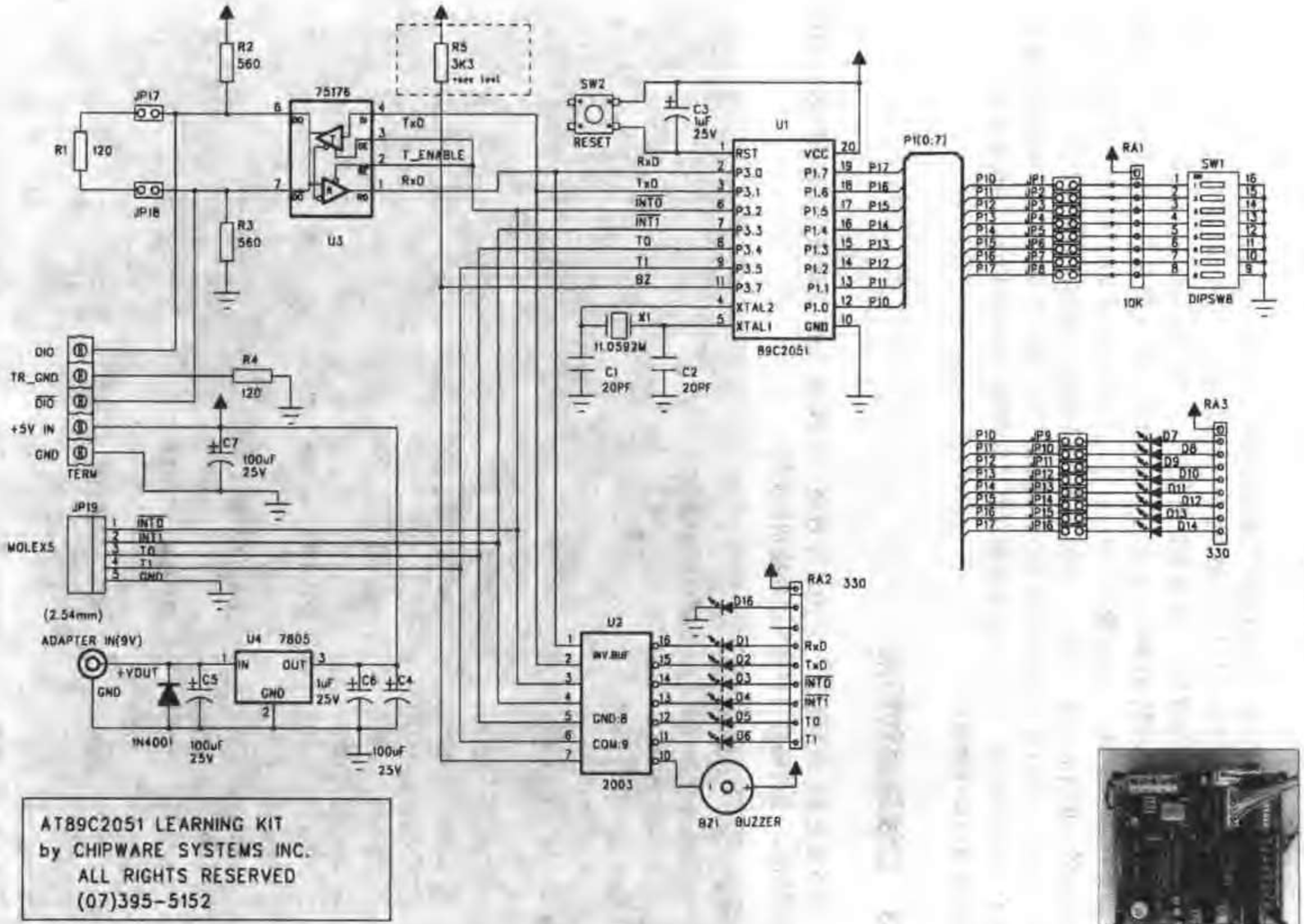


图 3-2

3-2 首先确认电路板是正常的

写任何 8051 的程序之前，我们首先要确定的是电路板本身是正常的，也就是说，该 PC 板上的电路不可以有不正常的短路情况发生，而且通电后必须提供非常稳定的+5V 电源给 AT89C2051 微控制器来使用，也唯有如此前提下，我们所写的 8051 单片机控制程序才能正常运作。所以当我们拿到 AT89C2051 学习 DIY 板时，焊接完所有零件后，必须先用目视的方法检查 PC 板上的线路，确定没有任何短路后，不插上 AT89C2051 直接接上电源，然后用数字电表检查所有的电源引脚端，是否已经得到正确的供应电压，如果都正确的话，就可以开始规划第一个程序了。由于此时没有监督程序的介入，所以许多 8051 重要的参数值都要自己来了，比如堆栈值，是否 128B 的数据存储区该清除，Timer0/Timer1 的设置等等，经过一次学习后就会依样画葫芦了。

3-3 让线路板动起来

以下就是我们所写的第一个 8051 小程序，让 PC 板上的 LED 进行交替的闪烁，程序看起来很简单，不过还是有许多重点要特别说明的。

```
;program name T1_2051.asm
;This program demo how to use ATMEL AT89C2051
;program written by Lin S.M.;Chipware Systems Inc 886-7-393-5152
;ALL RIGHTS RESERVED
;
STACK   EQU     60H
;
        ORG     0000H
        LJMP    RESET           ;RESET ROUTINE
        ORG     0003H
        LJMP    EXT0            ;EXTERNAL 0 INTERRUPT
        ORG     000BH
        LJMP    INT0            ;TIMER0 INTERRUPT
        ORG     0013H
        LJMP    EXT1            ;EXTERNAL 1 INTERRUPT
        ORG     001BH
        LJMP    INT1            ;TIMER1 INTERRUPT
        ORG     0023H
        LJMP    SINT            ;SERIAL I/O INTERRUPT
;
RESET
        CLR     IE.7
        MOV     R0,#7FH;1
```

```
        MOV     @R0,#00H
        DJNZ    R0,$1           ;CLEAR 128 BYTES DATA MEMORY
        MOV     SP,#STACK
        CLR     P3.7            ;CLEAR BEEPER
        CALL    DELAY
MAIN
        MOV     A,#AAH
        MOV     P1,A
        CALL    DELAY
        MOV     A,#55H
        MOV     P1,A
        CALL    DELAY
        LJMP    MAIN
;
DELAY
        MOV     R6,#00H
$1      MOV     R7,#00H
$2      DJNZ    R7,$2
        DJNZ    R6,$1
        RET
;
EXT0    RETI                    ;DISABLE INTERRUPT
INT0    RETI                    ;DISABLE INTERRUPT
EXT1    RETI                    ;DISABLE INTERRUPT
INT1    RETI                    ;DISABLE INTERRUPT
SINT    RETI                    ;DISABLE INTERRUPT
```

程序主要动作是很单纯的闪烁，但是 RESET 之后还是有很多件事要做的，首先是清除数据存储区内部的数据，然后设置系统的堆栈值，而清除 P3.7 的用意是将蜂鸣器的声响关闭，最后才开始做 P1 端口的闪烁动作。分别对 P1 送出 55H（01010101B）与 AAH（10101010B），并加上适当的延迟后，肉眼看起来就好像 8 个 LED 都在走动的感觉。8051 共有 128 个字节的数据暂存区，为了避免某些参数值发生错误，我们特地在程序开始运作后，就执行默认值清除的动作，这段程序是不能放在堆栈设置之后，否则会使得堆栈值也被清除成 0，接下来的程序调用后一定导致死机的。在第一个程序示范当中，我们也把 8051 的中断服务程序进入点标示出来，虽然只有一个 RETI 的指令而已，但是如果程序错误因而进入中断程序时，可以立即返回主程序，而不会造成严重的当机。我们希望所有写 8051 程序的读者都能养成类似的习惯，才能在程序不幸出错时以最短的时间找出 bug 来。

3-4　定时中断程序的重要性

第一个 8051 的程序如果能顺利执行的话，它可代表两种意义：一是线路板是正常工作

的，二是基本的程序与 AT89C2051 的刻录过程都是对的。接下来我们可以放心去撰写更大更复杂的程序。当然此时的挑战还会更多。在上面的程序范例当中，我们程序调用的写法都是 call，好像与 8051 汇编语言的标准写法（ACALL/LCALL）有些不同，这是我们是使用 X8051 进行程序的编译，该编译程序会自行判断被调用程序的远近，而转换成对应的 ACALL 或 LCALL。如果您是使用其他厂牌的 8051 编译程序时，可能还是要用原来的写法才行。接下来的示范程序是一个定时中断的写法，我们打算安排一个 10ms 的定时中断给 AT89C2051，并且每隔一秒钟后，调整 P1 端口上的 LED 显示。如果不用定时中断的话，上述的动作会非常难写，在正式写该程序之前，我们可以把《单片机 8051 彻底研究基础篇》一书中的“8051 的中断彻底研究”翻阅一遍之后，再开始写程序。

第二个 8051 中断示范程序见光盘。

这个示范程序就有一点难度了，不过若每个函数分别去分析的话，也许就不难分析了。程序首先设置 MS10 与 SEC1 等的初始值，然后设置 TIMER0 的计时值，由于 8051 的 TIMER 都是上数式的计数器，所以正确的数值需要用 65536 去减才行，接着让定时器启动，并且允许 TIMER0 做定时中断，最后才正式允许系统中断（即 EA=1），设置到这里后，只要 TIMER0 一数超过 65536 时就产生定时中断，进入程序 INT0 的部分。由于 8051 的 16bit 定时器无法自动加载，所以一进入中断服务程序 ISR 后，首先就是填入 TIMER0 的计时值，然后判断 MS10 的变量值，若 MS10 是 50 时，把 P3.0 上的 LED 点亮，若 MS10 是 100 时代表 1 秒钟已经过去了，接着就调整 SEC1 变数内部值，并且在 P1 端口上做秒数的显示。

特别值得一提的是：我们在一进入定时中断的服务程序时，就把 P3.4 位设成 1，ISR 最后要回原程序时才把 P3.4 清除成 0，当程序执行时只看到代表 P3.4 的 LED 微微地亮而已，并无法看出任何端倪。可是如果用示波器观看时就很有意思了，P3.4 上的信号代表 TIMER0 定时中断的处理时间，以这个示范程序为例，我们测出的时间约是 25μs 左右，这代表定时中断约处理了 20 来个指令后就回 MAIN 程序，约占中断周期的 0.25%，这段时间应该愈短愈好。反之，时间过长时主程序的执行效率会迅速降低，造成整个系统的极度不稳定。如果您对中断的写法理解的话，就能写出可多任务的应用程序，但是其中的困难程度会相对地提高许多。

其实第二个 8051 程序例也是一个多任务的执行范例，它同时做了以下的事情：

（1）每秒改变 P1 的状态一次。

（2）每半秒改变 P3.0 的状态一次。

（3）每 10ms 中断时，在 P3.4 点上产生一个 25μs 的正向脉冲。

（4）主程序 MAIN 部分在 P3.5 上产生约 3.48Hz 的方波信号。

上述的四项动作都是分别独立的，而程序的写法就是如此而已，通常我们会把重要性较低的程序放在主程序上执行，而重要的检查程序则放到定时中断里，以便能够实时判断并处理。这里只有加入定时中断的程序范例而已，如果其他的中断都加入时，我们就必须要去指定各个中断源的优先级了，并且尽量减少执行时间，以免某个中断占用过多的系统资源。

有许多用功的读者曾经来函提到 8051 在 RS485 串行通信的问题，其实我们在本书最后三章中，对串行通信方面的讲解已花了相当大的篇幅。如果要对 RS485 更深入研究的话，可能再写一本书都还讲不完，对于这一方面的通信技巧与写法实例将在另一本新书《VB 与串行通信彻底研究》书中进行介绍，希望这一章的两个范例程序对您有启示的作用，仔细且好

好地研究一番吧！绝对有所收获的。

3-5　本章使用软件

本章使用的软件为：X8051 8051 Assembler 汇编程序。

3-6　本章使用硬件

本章节使用硬件为：

（1）FLAG51 单片机控制板。

（2）AT89C51 刻录板。

（3）AT89C2051 学习板。

（4）AT89C2051：内含 2KB Flash 的 8051 单片机。

（5）AT89C1051：内含 1KB Flash 的 8051 单片机。

3-7　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.rs232.com.tw：查询电子零件信息。

http://www.atmel.com：查询 AT89 系列 Microcontroller 信息。

http://www.keil.com：查询 8051 Assembler 及 C 语言信息。

http://www.iar.com：查询 8051 Assembler 及 C 语言信息。

http://www.ti.com：查询 75176 串行通信相关信息。

第 4 章

单片机实战应用三例

介绍了这么久的单片机，读者一定也想看看实际的应用技巧吧？本章分别介绍电子计时控制器、电子测速器及自助加水机等三个实例的内幕与窍门。

经过了前面几章的开发与探讨，“旗威”的单片机专栏分别完成了几项学习性相当高的产品，分别有 I/O 监视器、七段显示控制板和 AT89C51 的刻录器等等，似乎可实验和试验的项目也逐渐增多了。这一章里我们要通过三个应用实例，来示范并说明 8051 单片机的应用。读者或许可以从这些说明中获得其他的灵感，然后开始您自己的实验应用。

4-1 电子计时控制器

实验工具：FLAG51 控制板、FLAG-DISP 显示器、外部按键开关、FLAG 万用实验板和 SSR 固态继电器。

大多数计算机工程师的实验工作室中都有一盏紫外灯，不过这可不是用来杀菌的，而是拿来清除 EPROM 内部数据的。通常我们是一早开始工作时，就把紫外灯打开，碰到数据需清除时，就随手把 EPROM 放到紫外灯底下，30 分钟后 EPROM 内部的数据就变成空白的了，可是这种作法感觉有一点浪费电力且不环保，因此，我们想利用 FLAG51 单片机控制器，七段显示器以及一块外加的 AC 电力控制板组成一套电子式的计时控制系统。每次欲清洗 EPROM 的数据时，只要按个开始键后，这个系统就点亮紫外灯持续 30 分钟，并实时显示剩余的时间（精确度到秒），当设置的时间到达后，则自行关闭紫外灯的电源。

图 4-1 是本系统的组合示意图，图 4-2 为外加的 AC 电力控制电路，线路中只用到一枚 SSR（Solid State Relay 固态继电器），即可控制交流电力 25A 以内的电子设备。由图中可看到本系统中只有一个机械开关输入，一个输出信号以便控制 SSR，再通过 SSR 控制外部的 AC 电力。

系统程序应该包含以下几个部分，请在确实明了整体架构后，再开始动手写程序，否则到最后阶段会演变成无法好好收尾的“四不像”程序，初次写汇编语言的读者请务必记住以下窍门：

（1）每秒定时显示：显示所剩余的时间，FLAG-DISP 有好多种显示模式可应用，应能判别使用何种模式最为方便，对 FLAG-DISP 不熟悉的读者可以查看本书第 13、14、15 章的

介绍。

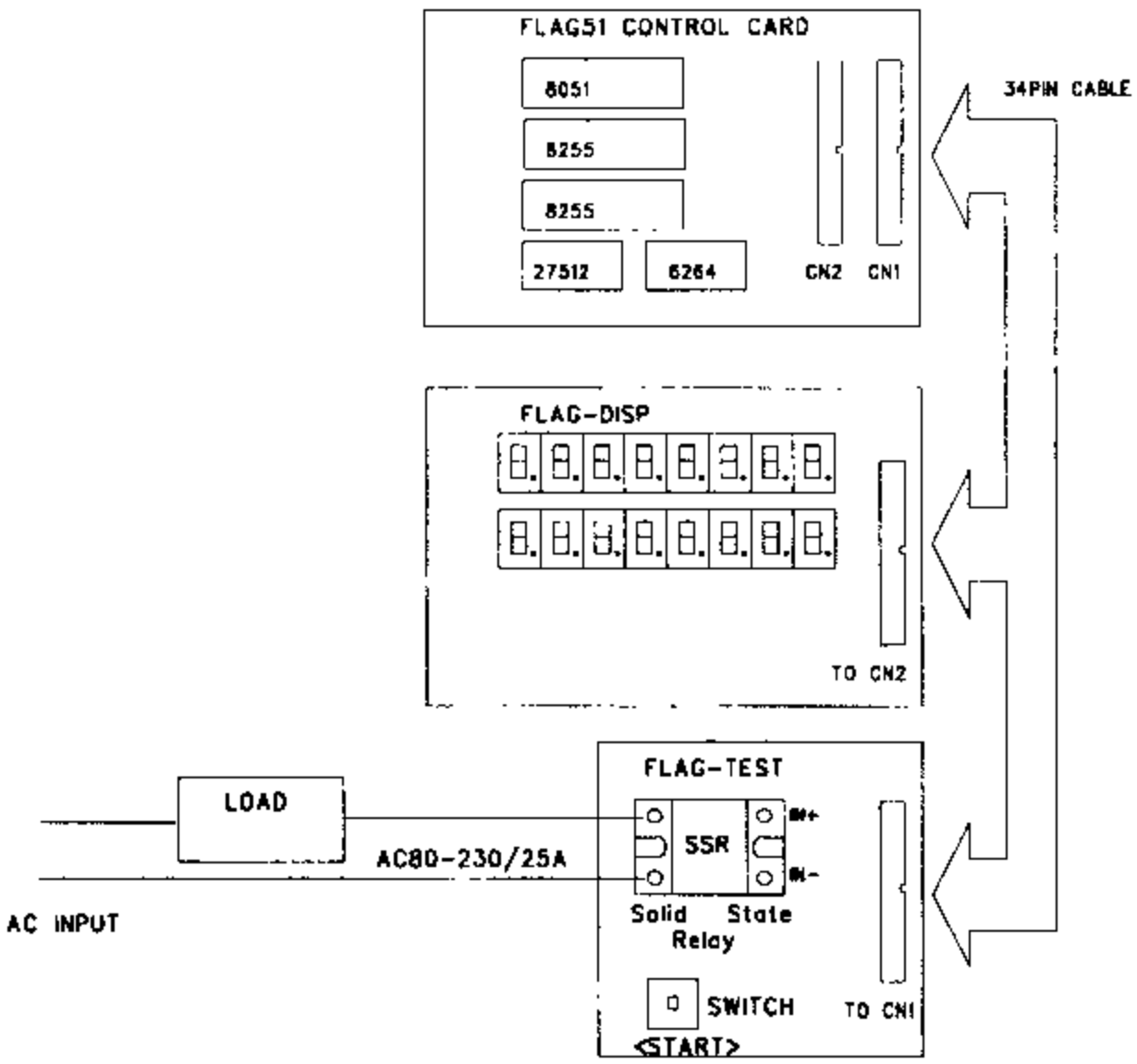


图 4-1

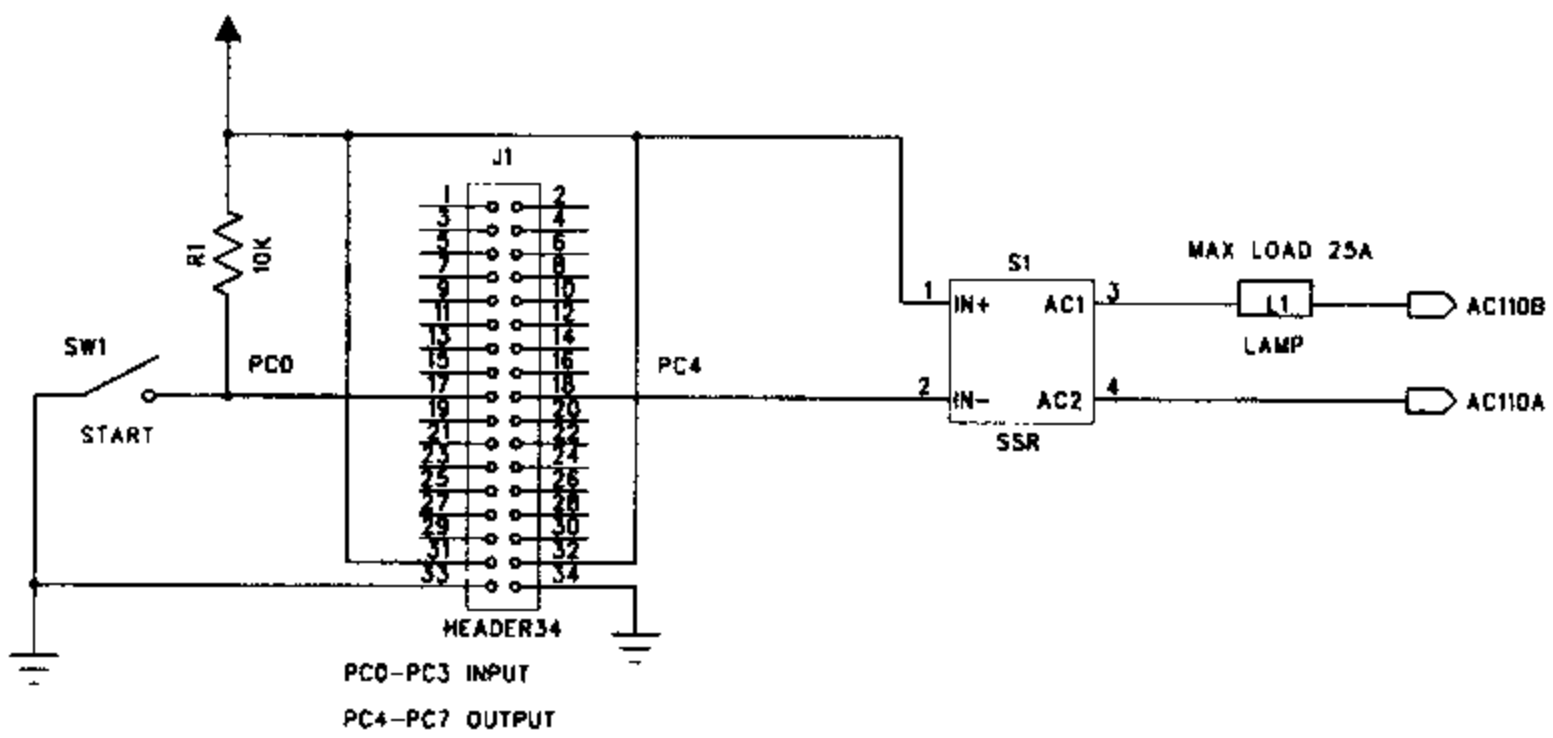


图 4-2

（2）输入开关的状态监视，此信号一经确认后会立即送出启动的信号给 SSR 固态继电器。

（3）设置时间到达监测，此条件成立后即关闭 SSR。

（4）考虑累积时数显示：另一组显示器则指示当天清洗的总时数。

做类似的控制系统通常都要先准备好外围硬件后，并测试正常后才逐一加入程序模块，一边实验一边整合系统，依实际的需要修正硬件外围，最后才完成整个系统的软硬件设备。由于 FLAG51 控制板可与 PC 联机，所以在测试阶段时，我们可以先在 PC 上发展程序，将执行程序的二进制文件由 PC 转送到 FLAG51 控制板的 SRAM 6264 上执行，直到所有的动作都正确后才将程序刻录到 EPROM 上。如果读者身边有 8051 的 ICE（仿真器）时，开发程序的时间可能还会更少一些。

程序 1 的例子是下数计时控制器，我们也可以改成上数定时器，然后把输入开关与 PC 的电源开关并联（当中必须做信号隔离），每次计算机开机后便立即记录使用时间，每天将开机时间累计起来，便可统计该计算机是否有妥善利用。如果把这种观念延伸到制造工厂，应用的范围将更加宽广了。

```
;程式名称:TIMER.ASM
;程式规划者:林伸茂
;设计日期:1994/08/01
;应用范围:紫外灯自动计时器
;使用控制器:FLAG51+FLAG-DISP 七段显示器
PPIPA1      EQU     A000H           ;CN1 接 SSR 输出及 SW 输入
PPIPB1      EQU     A001H
PPIPC1      EQU     A002H
PPICTL1     EQU     A003H           ;PC0-SW 输入,PC4-SSR 输出
;
PPIPA2      EQU     B000H           ;CN2 接 FLAG-DISP 七段显示卡
PPIPB2      EQU     B001H           ;FLAG-DISP 操作说明请参考 RUN!PC
PPIPC2      EQU     B002H           ;第六期和第七期
PPICTL2     EQU     B003H
;
MINUTE      EQU     83F0H
SECOND      EQU     83F1H
            ORG     8000H
START
;           MOV     SP,#60H
            LCALL   DELAY
LCALL       INIT_8255               ;8255 起始设置
            LCALL   INIT_TIME       ;时间起始设置
            CLR     RI
            MOV     P1,#FFH         ;LED ON
$WAIT       LCALL   RD_KEY          ;READ KEY IN PC0
            JZ      $WAIT           ;IF A=0 NO KEY PRESS
            LCALL   DELAY_10MS      ;DELAY 10 ms
            LCALL   RD_KEY          ;RECHECK KEY
```

```
            JZ          $WAIT           ;DEBOUNCE 弹跳消除
AGAIN
            LCALL       INIT_TIME
            MOVX        @DPTR,A
            SJMP        $ADJ_END
$1          MOV         A,#59H
            MOVX        @DPTR,A
            MOV         DPTR,#MINUTE
            MOVX        A,@DPTR
            JZ          $ADJ_END        ;NO MORE
            ADD         A,#99H
            DA          A
            MOVX        @DPTR,A
$ADJ_END    RET
;输入键检查
;A=1 KEY PRESS,A=0 NO KEY PRESS
RD_KEY      MOV         DPTR,#PPIPC1
            MOVX        A,@DPTR         ;READ PORT C
            CPL         A               ;A=/A
            ANL         A,#00000001B    ;GET ONLY BIT0
            RET
;紫外灯点亮
LAMP_ON     MOV         DPTR,#PPICTL1
            MOV         A,#08H
            MOVX        @DPTR,A         ;PC4=0
            MOV         P1,#0FH
            RET
;紫外灯熄灭
LAMP_OFF    MOV         DPTR,#PPICTL1
            MOV         A,#09H
            MOVX        @DPTR,A         ;PC4=1
            MOV         P1,#00H
            RET
;显示剩余的分秒值
DISP_TIME
            MOV         R1,#80H         ;PA,PB & PC ALL OUTPUT
            MOV         DPTR,#PPICTL1
            MOV         A,R0
            MOVX        @DPTR,A
            MOV         A,#FFH
            MOV         DPTR,#PPIPC1
```

程序 1：紫外灯自动计时器

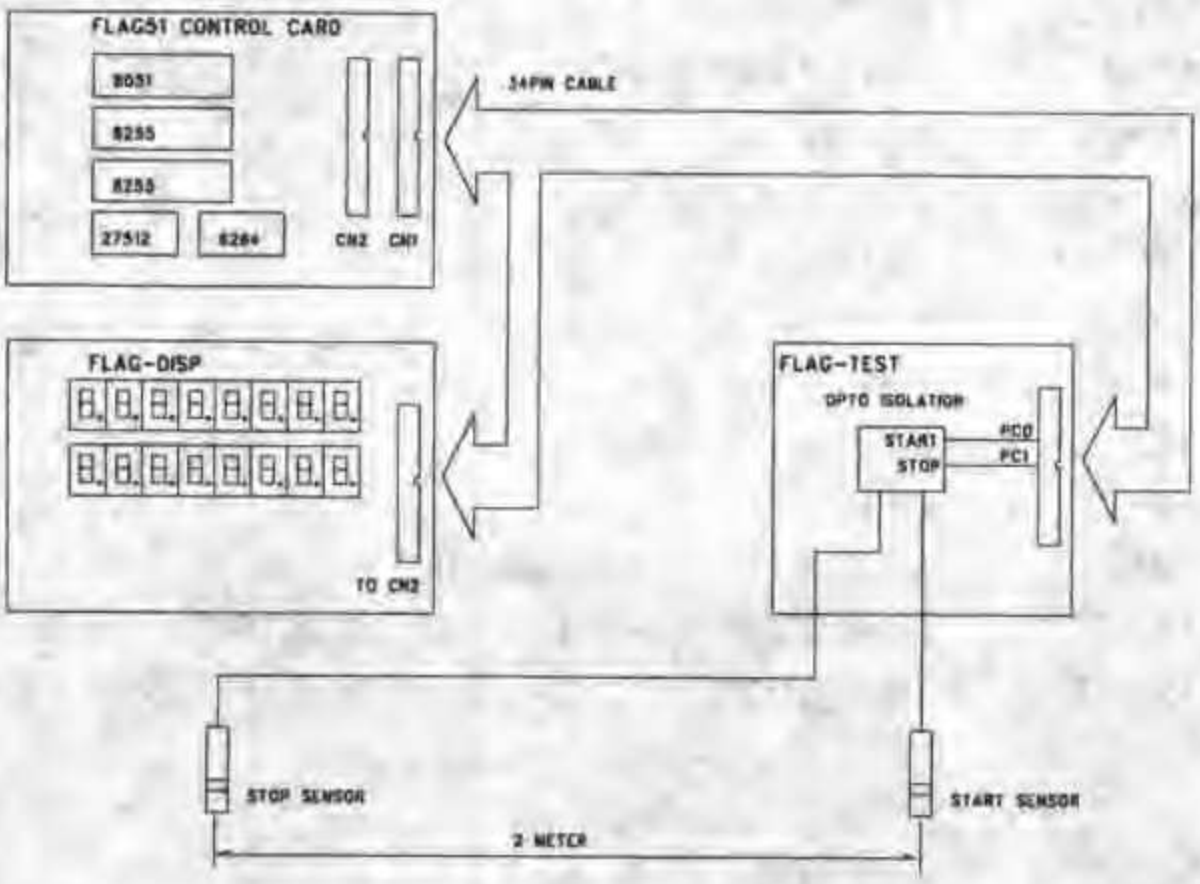
```
            MOVX    @DPTR,A         ;LET PC=FFH
;
            MOV     DPTR,#PPICTL2
            MOV     A,R1
            MOVX    @DPTR,A
            MOV     A,#FFH
            MOV     DPTR,#PPIPA2
            MOVX    @DPTR,A
            INC     DPTR
            MOVX    @DPTR,A
            INC     DPTR
            MOVX    @DPTR,A
            RET
;计时值的初始设置
INIT_TIME
            MOV     DPTR,#MINUTE
            MOV     A,#30H          ;30 MINUTES
            MOVX    @DPTR,A
            MOV     DPTR,#SECOND
            MOV     A,#00H
            MOVX    @DPTR,A
            INC     DPTR
            MOV     R0,#6
            MOV     A,#FFH
$1          MOVX    @DPTR,A
            INC     DPTR
            DJNZ    R0,$1           ;FILL BLANK
            LCALL   DISP_TIME       ;DISPLAY TIME ON UPPER SIDE
            LCALL   DISP_TIME       ;DISPLAY TIME ON UPPER SIDE
            RET
;
DELAY       MOV     R6,#00H
DLY1        MOV     R7,#00H
            DJNZ    R7,$
            DJNZ    R6,DLY1
            RET
;
SDELAY      MOV     R7,#20H
            DJNZ    R7,$
```

```
            RET
;
DELAY_10MS
            MOV     R6,#19
$2          MOV     R7,#00H
$1          DJNZ    R7,$1
            DJNZ    R6,$2
            RET
;
```

4-2 电子测速器

实验工具：FLAG51 控制板+FLAG-DISP 显示器+IO 监视板+外部开关或光电开关两个。

谈到测速器马上有人会联想到高速公路的超速照相机，它是结合高频雷达波与多普勒效应的原理测出汽车的实时速度，不过在此我们要举的例子其原理较为简单。在一段已知直线距离的头尾两端放上两个开关或传感器，只要测得两个开关的时间差，再经过换算即可以得到当时的时速值。图 4-3 是本系统的组合示意图，而本程序的绝窍应该在时间与距离单位的转换上，以下是本程序发展时的几个重点提示：




```
V(velocity)   =D(distance)/T(time)
              =2 METER / T mS
              =(2/T) (METER/mS)
              =(2/T) (METER/mS) * (1KM/1000METR) * (1mS/0.001SEC) * (3600SEC/1HR)
              =(7200/T) (KM/HR)
```

图 4-3

（1）利用定时中断去计时，若测试车速时定时中断的时间应在 ms 等级。

（2）计时单位到时速间的转换安排。

（3）时速值的显示程序。

（4）如何通过 RS-232 联机传回时速值。

SPEED.ASM 程序的安排是这样的（如程序 2）：FLAG-DISP 上排显示时速值，下排则显示内部的确实计数值。当汽车通过第一个光电开关时，内部计数值马上由零往上数，直到第二个光电信号成真时才停止。假设计数值是 63（十进制）时，则经过程序的换算：7200/63=114 公里/小时，这代表若该车花了 63ms 的时间通过 2 米的直线距离时，此时的车速应是每小时 114 公里左右。不过程序为了提高显示时速的精确度，特地将在分子的常数 7200 改成 72000，而 72000/63=1142，这表示更精确的时速值是 114.2 公里/小时，只须在 1142 值送到 FLAG-DISP 七段显示器时，动个手脚在数字 4 处加个小数点时，显示值就成为 114.2 了。SPEED.ASM 能显示的最大时速为 720.0 公里/小时，程序中使用到一个 32 位的除法例程，该例程取自 1993 年“旗标”出版的《单片机 8051 彻底研究》一书，对 SPEED.ASM 程序除错时，发现了其中有一个 bug，会使得 DIV_4BYTE 例程进行除法时出错，可是却没有任一个仔细的读者发觉！其实，书中的这些例程可都是宝藏啊！当您翻遍其他 8051 的书都找不到一个像样的数值加减乘除运算例时，这里都有现成的可供参考，马上挪过来即可享用，而不需对这些表达式的写法伤透脑筋。许多设备的马达转速，还有现在颇流行的保龄球球速显示器，大概都是运用以上的原理开发出来的，您还想到其他的应用吗？

程序 SPEED.ASM 见光盘文件。

程序 2：车速监视器

```
            INC     DPTR
            DJNZ    R0,$1               ;FILL BLANK
            LCALL   DISP_SPD            ;DISPLAY SPD
            LCALL   DISP_SPD            ;DISPLAY SPD
            RET
;
;A<>0 SOME KEY PRESS,A=0 NO KEY PRESS
RD_KEY1
            MOV         DPTR,#PPIPC1
            MOVX    A,@DPTR             ;READ PORT C
            CPL     A                   ;A=/A
            ANL     A,#00000001B        ;GET ONLY BIT0
            RET
;
RD_KEY2
            MOV     DPTR,#PPIPC1
            MOVX    A,@DPTR             ;READ PORT C
            CPL     A                   ;A=/A
            ANL     A,#00000010B        ;GET ONLY BIT1
            RET
```

```
;
DELAY       MOV     R6,#00H
DLY1        MOV     R7,#00H
            DJNZ    R7,$
            DJNZ    R6,DLY1
            RET
;
SDELAY      MOV     R7,#28H
            DJNZ    R7,$
            RET
;
DELAY_10MS
            MOV     R6,#19
$2          MOV     R7,#00H
$1          DJNZ    R7,$1
            DJNZ    R6,$2
            RET
;
;ROUTINES FOR FLAG-51 CONTROL BOARD
;以下程序请参考《单片机彻底研究》一书中的程序说明,在此不再重复。
```

4-3 自助加水机

实验工具：FLAG51 控制板+FLAG-DISP 显示板+外部按键开关+FLAG 万用实验板+SSR 固态继电器+抽水马达

某城市常可看到自助式的加水机，只要投入硬币，再按下开始键后，就会流出 20 公升的水，其动作就好像是加油站的加油机一样，唯一的差别是加油机的安全因子较高，动作时绝对不能有任何火花出现。图 4-4 是自助加水机的控制方块图，用户投入两枚正确的硬币后，显示水量的显示器自动归零，当用户按下开始键时，启动抽水马达开始打出水来。由于事先已知水管的直径，所以可以计算出每 0.1 秒的出水量（公升/0.1 秒），马达一启动后显示器上的出水量一直往上累加，直到 20.0 公升到达后，才将抽水马达关闭。

真正加油机的动作更精确，当快到达设置的加油量时，马达的转速会变慢，以便缓缓到达最后的设置值，而且绝对不能过量，只能够少加而不能多加，否则怎么赚钱呢？在控制上这种控制方法称为弱衰减，因为一流过量后便不可能再抽回来。FLAG-DISP 显示板上有两排七段显示器，上排可显示已投入的金额，抽水马达一启动后，总金额立即减去 20 元，下排则显示每次累计的出水量，显示单位应可到 0.1 公升。

FLAG51 控制板则负责抽水马达的控制以及出水量的计算等等。自助加水机若当成学校的专题来做时，只要动作正常即可交差，但是，如果真的要能够营业，还要做更进一步的改良，才不会“漏气”，以下就是一些须改良待加强的地方：

（1）投币器要改用可防伪币的电子投币器，以免收到假的拾元硬币。

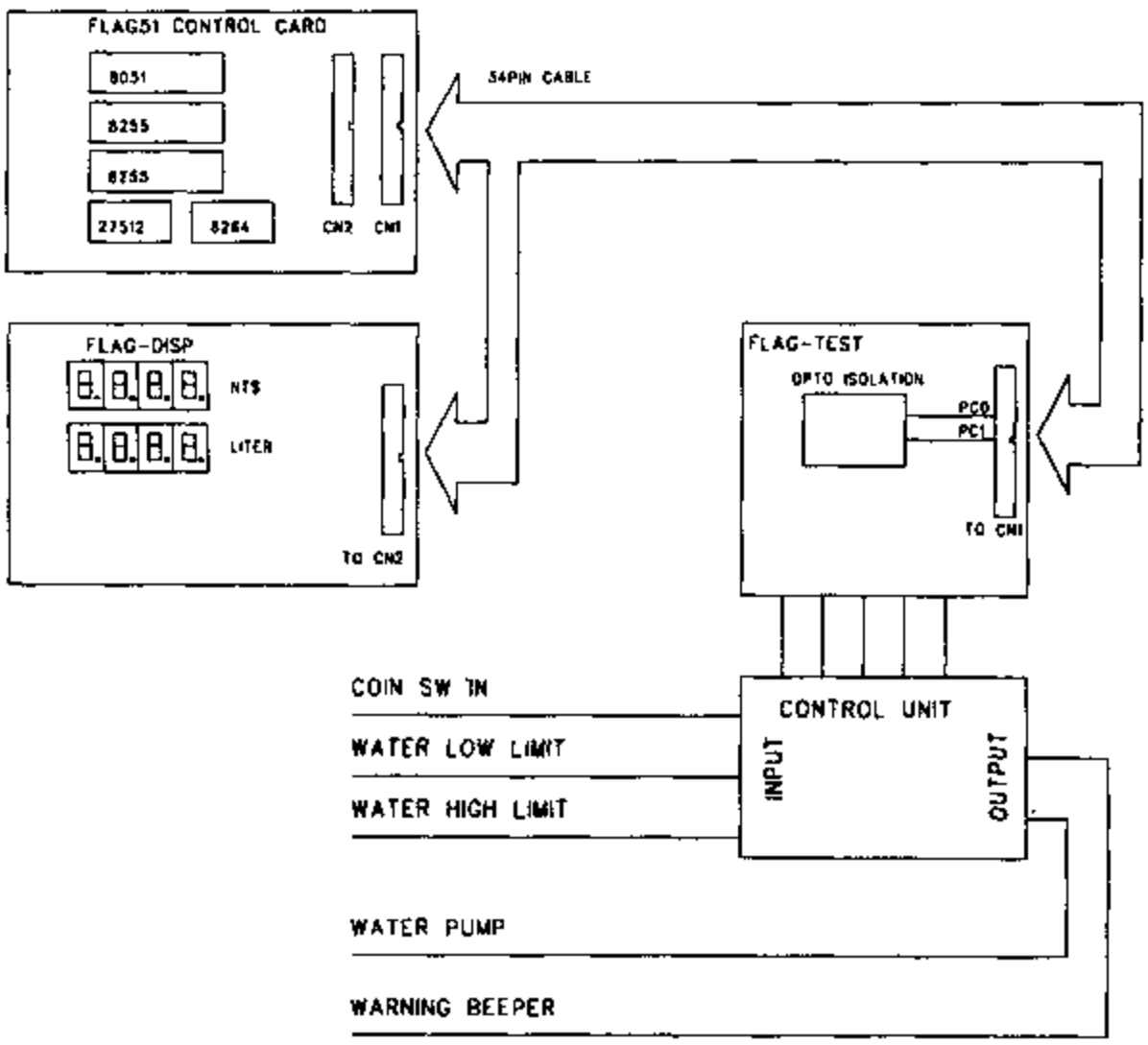


图 4-4

（2）投币信号要用程序检查其有效时间，以免有人用“钓鱼”的方式来免费加水。

（3）马达起动后若没水出来时，要有回传信号通知控制板有故障发生，以免操作者加不到水，拿你设计的机器出气。

（4）系统必须有数据保留的功能，以免停电后重新启动时动作错误。这里所谓的停电可分成正常的停电和操作者的故意断电，不论何种情况下都必须功能正常。

（5）系统必须有防止干扰的能耐，这点特别重要。电信局的公共电话机遭受干扰后即可免费打国际长途电话，就是软件程序设计上的一大败笔。

4-4　本章使用软件

本章使用的硬件如下：

（1）X8051.EXE Assembler。

（2）DIS51.EXE“旗威科技”的 8051 反汇编程序。

4-5　本章使用硬件

本章使用的硬件如下：

（1）紫外灯：可在医疗仪器商店内购得。

（2）FLAG51 单片机控制板。

（3）FLAG-DISP 七段显示板。
（4）SSR 固态继电器。
（5）按键开关。
（6）光电开关或近接开关。
（7）I/O 监视板。
（8）抽水马达。

4-6 相关信息网站

您可经由下列公司、网站取得更进一步的信息：
http://www.chipware.com.tw：查询 8051 单片机相关数据。
http://www.rs232.com.tw：查询电子零件相关数据。
http://www.cpu.com.tw：查询零件相关数据。
http://www.sensor.com.tw：查询传感器相关数据。
http://www.atmel.com：查询 AT89 系列单片机相关数据。
http://www.iar.com：查询 8051 软件数据。

第 5 章

软硬件排错技巧

在单片机的开发过程中，排错往往会占掉 1/3 以上的时间，我们将提出许多亲身的宝贵经验与读者分享，懂了这些窍门后，就不再害怕排错了。

5-1 案例一：外派排错维修

好几年前随着整厂设备到外地出差，由于有许多监控设备都是我们亲自设计的，现场调试时总会碰到一些错误，小修改是难免的，我们出发前准备好的工具箱内只装了一盒磁盘、一迭原始程序报表、一支最便宜的逻辑笔（Logic Probe）和一台 EPROM 刻录器，当时老板是这样交待的：人和备份磁盘回来就好，其他工具就留给当地的工程师使用。这个案例告诉我们：排错可以很容易掌握；许多仪器都可以用来排错。

5-2 案例二：没有 ICE 无法做事

某公司正要应征几位有经验的固件工程师，应征的考题中有一题非常有趣：通常我们都靠 MICE（“全友”的在线仿真器）来做汇编语言的系统排错，可是如果有一天突然 MICE 不再是 MICE 时，你有办法继续工作吗？有的人回答：工作必须暂停，也有人提再买（或借）一台吧！有人则认为买两套设备以免在机器故障时出状况，换成是你的话，你的答案呢？

5-3 案例三：卖得越多麻烦越多

某公司设计了一台微电脑控制的生产设备，大部分行销台湾地区，有一天用户回报说机器无法正常动作，请派人过去“看看”，问题是该公司的出货量愈来愈大，维修问题的麻烦愈来愈大，最好的情况是请客户自行更换故障板，可是当初规划设备的控制线路及程序时，并没有将故障码的显示纳入，所有的维修都必须公司派人维修，现在只好开始品尝苦果了。

5-4 案例四：RESET 键不能随便加

FLAG51 单片机控制板设计完成后，历经近千余位读者的试用，有不少读者反应：FLAG51 怎么没有 RESET 键？有时候程序执行完就死机了，少了 RESET 键似乎很不方便！如果我们站在另一个角度看，用微电脑控制的微波炉或冷气机也没有 RESET 键，平常我们的个人计算机如果死机时，只需按下 RESET 键重新再来一次，系统（主机板以及软硬盘）整个重来一次，好像只有这一个方法了。再假设你正坐在波音 747 飞在几万米的高空上，当机长发现引擎控制有问题时，你能允许机长按下系统的 RESET 键（实际上并未有此键）吗？当然是不行了，难道要等到飞机开始垂直下坠时，你才会同意？RESET 这东西是不能随意加的。

上面四个例子分别在说明微电脑的控制系统初期的规划是相当重要的，我们的经验是至少花上一周到十天以上的时间去构筑整个硬软件的架构，实际写程序或画线路的时间反而较少，规划如果周详时，排错与维修都不是问题，至于是否使用 MICE 去做软硬件排错就不是那么重要了。不论系统是大或小，请记住一个重要的观念：设计要有易测性（Design For Testability）。硬件如此，软件程序更要如此，虽然初期会增加一些成本，但是如果和以后昂贵的维修费用比较时，就显得微不足道了。

5-5 排错方法 1：LED 接口

大约 10 年前开始，我们设计的微处理机控制板上都会留一个 LED 接口，FLAG51 单片机控制板也不例外，只要程序一启动后，先将系统状态值送到该 LED 接口上，而我们只要观看该接口的显示情形即可得知系统的状态如何，如果 LED 接口完全没动作时，那就表示 CPU 并没有取到程序代码，这时要检查 CPU 与 ROM/RAM 之间的总线（ADDR/DATA Bus），看是否有相互短路的情况，如果这一关卡通过了，你大概可以放心地去写接下来的程序，并且参照上述的方法在重要需检查的关卡加上 LED 接口的数值显示，排错的人员只要观察 LED 接口的值即可以判断系统是否正常，可是有些时候，LED 接口上的变化太快了，看不出送出的状态值来，此时可以视情况在 LED 接口显示之后加上一段 0.5～1 秒的时间延迟，那就可以清晰地看到 LED 接口的状态了。保守地估计，有一半以上的软件错误可用这种简易而有效的方法找出问题点来。

5-6 排错方法 2：逻辑笔配合法

原本可以动作的程序，加个小修正或判断式后就失常的事是经常发生的，问题就出在被修改的部分是否被执行，如要做这种验证通常需配合 MICE 才行，而且还要知道该段程序的真正地址方能配合中断。在此我们提供另外一种方法，首先检查系统上是否有空余的译码地址输出点（Decoder Output），在欲修正判断式的最前端加上一个 WRITE OUTPUT 指令，若以 8051 来讲，假设 F000H 这个外部地址是空余的，我们就可以在修正程序前加上两行指令：MOV DPTR，#F000H 及 MOVX @DPTR，A，程序开始执行后，把逻辑笔的探针接到 F000H 的解码输出点上，只要该修正段的程序执行到时，逻辑笔上一定会侦测到 MOVX 这个指令，

并且笔上的 LED 灯也会亮一下，由此我们就可以断定程序是否正常了。由于译码地址输出点送出的信号宽度很窄（<10μs），除非是用储存式的示波器，否则几乎看不出此译码地址输出点上的变化。

5-7　排错方法 3：沿途记录法

如果你是用 ICE 仿真器来排错程序时，可以用这个方法来找 bug。通常，在我们的系统中总会有一小段的 RAM 空间是没用的，我们可以在程序执行前先将该 RAM 区清除掉，然后程序执行时依序将数个检查值摆入 RAM 区中，最后再检查其中的内容，即可以判断程序哪里生病了。假如你写一个 16 位的除法例程，显示输出时一直有错误，你就可以在 RAM 区内分别存入除数与被除数，然后除法例程内每经过一次计算后，就把中途的结果存入这段 RAM 区内，最后再查看这些值就会发觉问题点了，不过，这种写法在使用时要先将程序重要的缓存器 PUSH 起来，免得排错例程本身无法排错，反而变成系统错误的主要来源。

5-8　排错方法 4：善用串行端口通信

单片机 8051 的排错有一个优点：多了串行通信端口，只要系统启始时规划妥当，随时都可以把工作状态传回给 PC 端，传输的信息最好都换成标准的 ASCII 码，以便 PC 收到后可以立即显示到屏幕上。如果你这一方面熟练的话，还可以加上接收 PC 送来的命令，让程序不仅可以排错，还可以让程序暂时停在某个阶段，必须收到 PC 下的某些指令后才继续执行，如能到此境界，你的排错程序已经有一点像基本的 MICE 了，不一定非要 MICE 才能开发微电脑系统。

5-9　我们的固件排错经验

不论做任何项目，我习惯单独用一本记事簿记录有关此项目的数据与信息，当程序做了任何修改或订正时，也顺便把修改的地方记下来，所以，程序一改错时，可以立即回复原先的样子。排错时也尽量以模块的概念排错，一定要这个模块确认正确后，才继续下个模块的排错，绝对不允许多个地方同时施工，然后再统一验收，这样一定会出问题，而且模块间的责任归属将无法认定，徒增系统维护与排错的困难度，这一点请读者一定要留意。

最近“旗威科技”主要的工作在完成一个可程控器（PLC）的控制程序，主架构早已完工了，并且也经由 MICE 验证，基本功能都已正常，剩下的工作是功能验证与新增功能的加入，虽然无法立即提升到欧美大厂的等级，但是看来已有类似的架构了，系统开始能动作后，我都是直接将程序刻录成 ROM，再插到 FLAG51 控制板上，而这段刻录时间刚好让忙碌的大脑做适度的休息，准备做下一个冲刺。

图 5-1 是从 Amazon 网络书店找到的写程序的好书。

图 5-1

可是，电源开启后，系统动作就有问题了，有时候 LED 接口工作不正常，有时候正常但一下子就死机了。首先我是怀疑可能 STACK（堆栈）设得不够大，可是以前写的程序都是用相同的设置环境，也没遇到问题！所以只好乖乖地把新加上去的程序模块拿下，重新又刻录一个 EPROM 插到 FLAG51 上，还是行不通，这下子问题可大了！继续删掉部分程序，即对程序进行 Remark（程序中的注解，对指令、语句等的非执行性解释）以让编译程序不产生该段程序代码的部分程序，继续刻录 EPROM，可是问题好像依然存在，到最后程序缩减到只剩 10 行左右，其中有一行是将某个特定值送到 P1 的 LED 接口上，结果程序竟然还是不能执行！可以动作的程序却变得很脆弱，只要手一靠近振荡线路部分就马上死机！竟然如此！这时我的心中已经有个谱了，但是天已快亮了，只好先小睡片刻后再作奋斗。

系统很容易死机的理由不少，但有可能是 CPU 到 ROM/RAM 间的总线信号强度不足，可是插上以前的 FLAG51 监督程序却不会这样啊？这时我才想到插在 FLAG51 控制板上的 EPROM 编号是 27C512，原先 FLAG51 使用的是 27C256，两者容量相差一倍，唯一的差异在第 1 脚上，27C256 的第 1 脚是 Vpp，正常情况下是接+5V，27C512 的第 1 脚则是 A15 地址线，在 FLAG51 控制板上是将摆 MONITOR ROM 的第 1 脚接+5V，如果以 27512 取代 27256 时，程序正确的摆入地址应该是由 8000H 开始，但是对 FLAG51 控制板而言都是正确的！知道问题的症结后就好办了，赶紧火速地再刻录一个 EPROM（数据由 8000H 开始放起），插上 FLAG51 后动作正常了，再花一个多小时的时间分别把其他新增的程序一一加入，系统动作也都完全正确了，也不会轻易死机，总算完成又一次排错工作了，下次还要再玩吗？当然要了。

图 5-2 是 27512 与 27256 引脚间的差异。

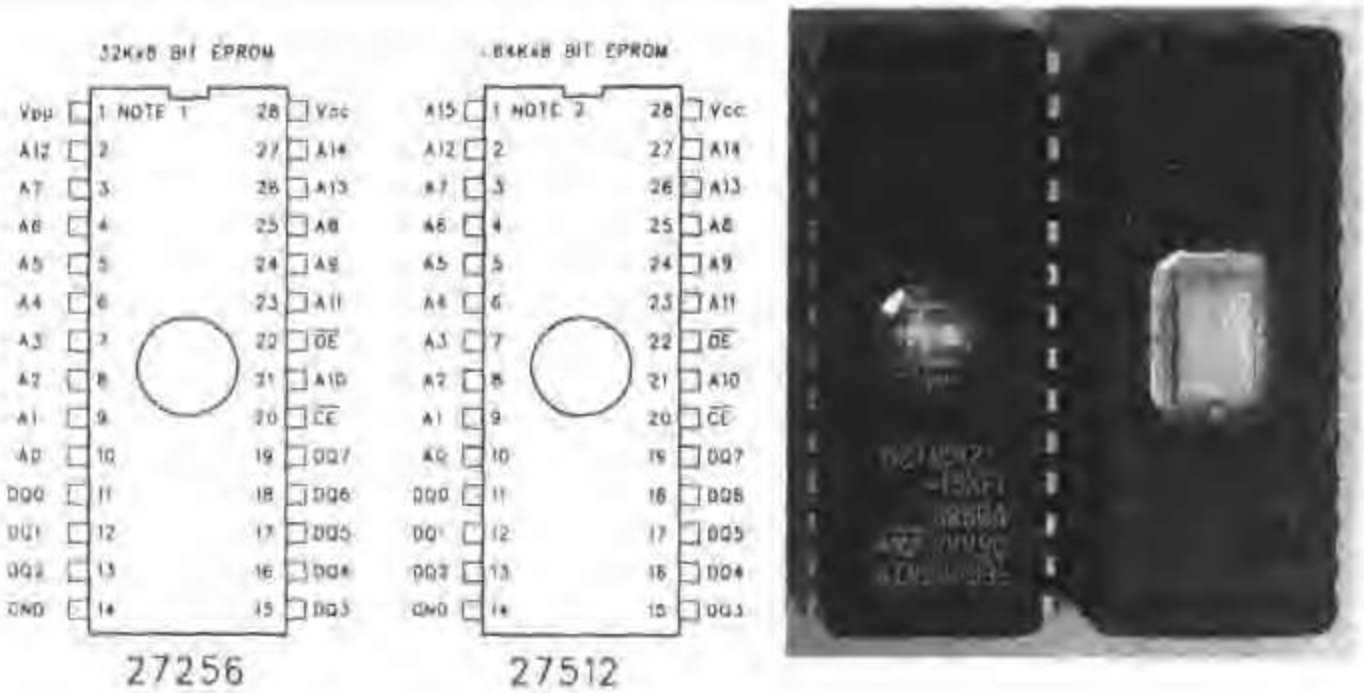


图 5-2

附注：

（1）27256 的第 1 脚 Vpp 仅在刻录时有用，正常使用时，该脚应该接+5V 的电压，可是某些厂商的 27256，Vpp 脚是空接或接 0 低电位时仍能正常动作。

（2）27512 的第 1 脚是 A15 地址线，控制上半段（8000-FFFF）的数据空间，当 A15=1 时代表 CPU 要读取上半段的数据。FLAG51 控制板的 ROM 插座已内定是 27256，所以板上的第 1 脚固定接 5V，造成若以 27512 替代时，刻录数据必须由 8000H 开始放起，否则将无法开机执行。

5-10　本章使用软件

本章使用的软件为：

（1）2500AD 8051 Assembler。

（2）IAR 8051 C Compiler。

5-11　本章使用硬件

本章使用的硬件如下：

（1）“固纬”逻辑笔（Logic Probe）。

（2）Tektronix TDS220 储存式示波器。

（3）“全友”计算机 MICE。

（4）“新华”计算机 Microtime 8051 ICE。

5-12　相关信息网站

可经由下列公司网站取得更进一步的信息：

http://www.chipware.com.tw：取得 8051 单片机控制板相关信息。

http://www.flag.com.tw：查询“旗标”的新书。

http://www.amazon.com：查询各种技术性的新书，不过邮寄费不低。
http://www.tek.com：查询示波器信息。
http://www.goodwill.com.tw：查询电子仪器与设备信息。
http://www.iar.com：查询 8051 C 与 Assembler 相关信息。
http://www.micetek.com.tw：查询 8051 ICE 相关信息。
http://www.microtime.com.tw；查询 8051 ICE 相关信息。
http://www.superlink.com.tw：查询 8051 ICE 相关信息。

第 6 章

8052 与 8051 的差异

如果你认为 8052 与 8051 只是内部程序空间大小不同，那就大错特错了。8052 除了程序空间与数据空间加倍外，又特别增强了 Timer（计时器）的功能，在本章里将对这些差异做详尽的说明。

6-1 脚位功能的差异

图 6-1 是 8052 与 8051 的脚位比较图。从硬件接脚看来，8052 与 8051 多了 T2 与 T2EX 两根脚，占用了 51 原先的第 1 脚（P1.0）与第 2 脚（P1.1），新增的功能脚都是跟增加的计数器 Timer2 有关。T2 是一个外部的脉波信号源（clock source）。8052 可接受的脉波信号源其中之一是系统的振荡信号源，即石英振荡晶体的频率除以 12 或 2 后，再当成脉波信号源。另一个脉波信号源就是外部的 T2 输入。当 8052 使用前者做信号源时，我们称 Timer2 在做计时的动作，因为系统频率已知，Timer2 的计数值就等于时间值。若使用未知频率的 T2 做脉波信号源时，只能知道所计算到的次数，所以我们称这种情况为计数（Event count）。

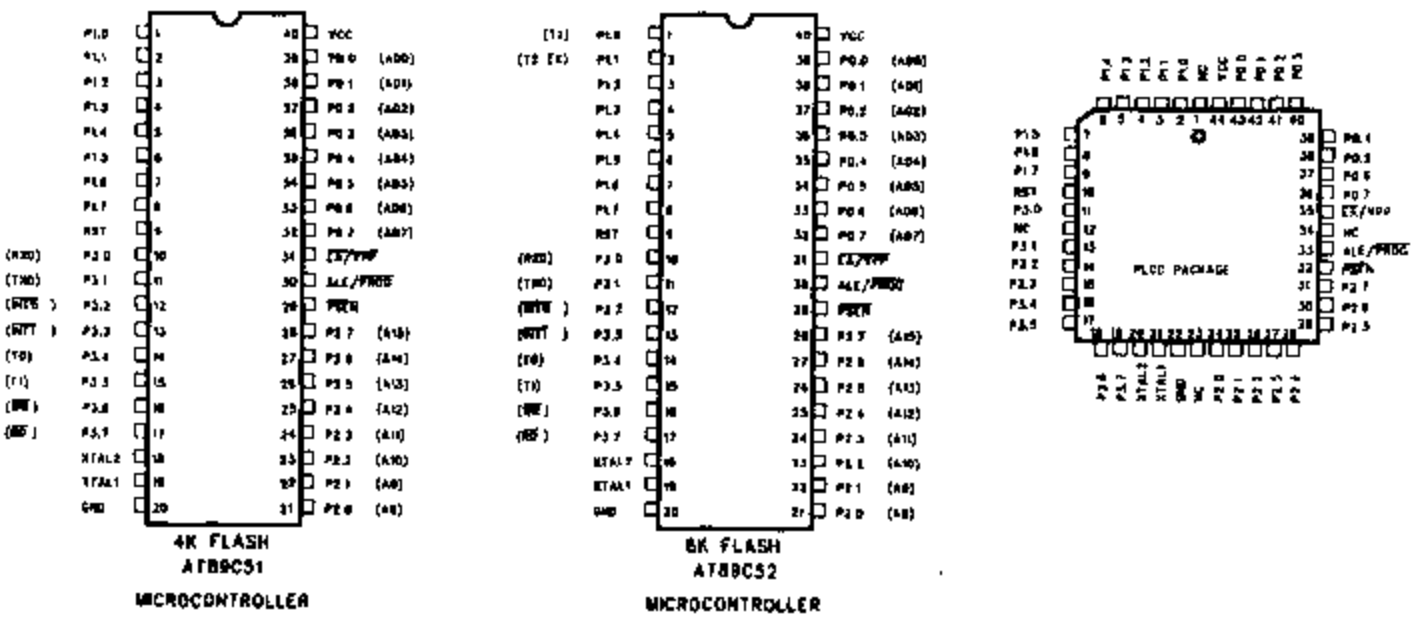


图 6-1

T2EX 脚由 1 转为 0（负缘触发）时可以对系统产生中断信号，当然其前提是我们的程序要先允许 T2EX 位动作才行。8052 的 T2 与 T2EX 若没有作用时，P1 的 8bits 的操作还是可以照常运作。

6-2 程序空间的差异

8052 已经把内容的程序空间加倍了，内部的程序（Program Memory）空间成为 8KB，数据（Data Memory）也扩充到 256 字节，比原先 8051 多出 128 字节，这段 SRAM 区占用 80H-FFH 的位置，这段区域都不是位地址。可是这不就跟 51 的特殊缓存器区冲突了？其实是不会的，8052 对新增的数据区特别加一个限制：一定要用间接寻址的方式存取。这也就是说，你需要用两行指令：

```
MOV     R0,#90H          ; 指定地址
MOV     A,@R0            ; 取得 Data Memory 90H 的内容,A=(90H)
```

才能取得 90H 这个地址的内容。8052 的 P1 port 也是占用 90H 这个地址，不过要存取 P1 的内容时其写法是：

```
MOV     A,P1             ; 该指令转换成机器码，其中第二个机器码>80H
                         ; 时，CPU 会去抓 SFR 的值而非 Data Memory，很明
                         ; 显地写法是有所不同的。
```

新增加的 Data Memory 空间要用间接方法存取是有点不方便，所以我们建议你应该把常用的变量放在 80H 以前的位置，这样的话，只要一个指令就可以做存取。较不常用的变数才放在 80H 以后。另外我们还要提醒你，系统改为 8052 后，你可以把堆栈 SP 放在新增的 Data Memory 区上，这样的话，从 30H-7FH 都可以放系统的变量，一般的应用应该是足够的。

6-3　8052 的 Timer2 彻底研究

当 8051 单片机推出后，在众人激赏之余，8051 的明显弱点也逐渐浮现了，最主要的部分是在 Timer 的部分，当 8051 做串行通信及定时中断后，Timer0 与 Timer1 皆被占用，如果要再做其他硬件方面的计时或计数，都会或多或少产生一些困扰。所以 Intel 在 8052 问世之际，就针对 Timer 部分做了一些修正，并增加 16bit 的 Timer2，以便解决计时与计数的诸多问题。

标准 8052 的 Timer2 有三种工作模式，分别是：

Capture 模式：计数计时值的捕获模式，使用场合为特定的计时或计数。

Auto-reload 模式：自动加载模式，使用场合在定时中断上。

Baud Rate Generator 模式：波特率产生器模式，这主要是用在串行通信上。

Timer2 的工作模式设置如表 6-1 所示。

表 6-1　　Timer2 工作模式

RCLK+TCLK	CP/$\overline{RL2}$	TR2	模式
0	0	1	16bit Auto reload
0	1	1	16bit Capture
1	×	1	Baud Rate Generator
×	×	0	OFF

8052 的 Timer2 主要的控制缓存器为 TCON2，该缓存器内的每个 bit 都是位地址，我们可以根据实际需要去进行设置或清除。要指定 Timer2 是计时或计数就由 C/T2 位来决定。当 Timer2 启动时，我们要注意两个外接 PIN 的状态，分别是 T2（1.0）T2EX（1.1）脚，因为这两根脚的状态会影响到 Timer2 的运作与中断请求。要熟悉 8052 的 Timer2 一定要通过方块图的解说才行，所以接下来的说明与叙述请一定要对照方块图。

TCON2 缓存器各位的定义，如图 6-2 所示。

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
D7	D6	D5	D4	D3	D2	D1	D0

图 6-2

◆ TF2（TCON2.7）8052 Timer2 的溢位标志，每次 16 bit 计数器数超过 FFFFH（65535），又从 0000H 开始时，TF2 被设成 1，必须用程序来清除。当 RCLK=1 或 TCLK=1 时，代表 Timer2 在 Baud Rate Generator 模式，这时的 TF2 就没有反应。

◆ EXF2（TCON2.6）Timer2 的外部输入标志，当外部的 T2EX 由 1 变为 0（负缘触发），而且 EXEN2 位已经设置为 1，则 EXF2 会成为 1，当 TF2=1 或 EXF2=1 时，都会产生 Timer2 的中断要求，EXEN2 必须用程序来清除。

◆ RCLK（TCON2.5）当 RCLK=1 时，串行通信中的接收是采用 Timer2 所产生的波特率。反之采用 Timer1 产生的波特率。

◆ TCLK（TCON2.4）当 TCLK=1 时，串行通信中的发送是采用 Timer2 所产生的波特率。反之采用 Timer1 产生的波特率。

◆ EXEN2（TCON2.3）当 EXEN2=1 时，外部 T2EX 信号的状态才会被处理。

◆ TR2（TCON2.2）Timer2 是否开始运作由此 bit 决定。

◆ C/$\overline{T2}$（TCON2.1）决定 Timer2 的输入 clock 是系统的振荡晶体或是由 T2 输入的外部 clock。

◆ CP/$\overline{RL2}$（TCON2.0）当 CP/$\overline{RL2}$=1 时，Timer2 当成 Capture 模式。当 CP/$\overline{RL2}$=0 时，Timer2 为 Auto reload（自动加载）模式。若 TCLK 或 RCLK=1 时，Timer2 自动变为 Auto Reload 模式。

6-4 Timer2 的 Capture 模式分析

请先看图 6-3 的 8052 的 Timer2 为 Capture 模式时的方块图，Timer2 的 clock 来源有两个，一是系统的 OSC/12，另一个是外部的 T2 输入，这根脚正是 8052 的第 1 脚 P1.0，clock 是否进入就看 TR2 位是否被设成 1，clock 进入 TH2 与 TL2 做上数（Up count）计数的动作，若数过 16bit 的表示极限 65535 时，TF2 就变为 1，TF2 可以对系统产生中断。

当 Timer2 在进行高速计数时，我们的程序若随机读取 TH2 与 TL2 的值时，可能会得到不正确的 16bit 计数值，8052 安排了 T2EX（P1.1）的输入控制信号来做计数值的锁定，当 T2EX 状态由 1 变成 0 的瞬间，将 Timer2 的 16bit 计数值撷取到 RACP2L 与 RCAP2H 缓存器上，请注意是一次锁住 16bit 的计数值，我们就可以用程序从容地读取 Timer2 的 16bit 计数值，T2EX 的信号也可以对系统产生 Timer2 的中断，如果不想要 T2EX 产生中断时，只要把 TCON2 缓存器内的 EXEN2 位设成 0 即可，其实只要你看得懂右边的方块图，8052 的 Capture 模式大概已懂 90%了。

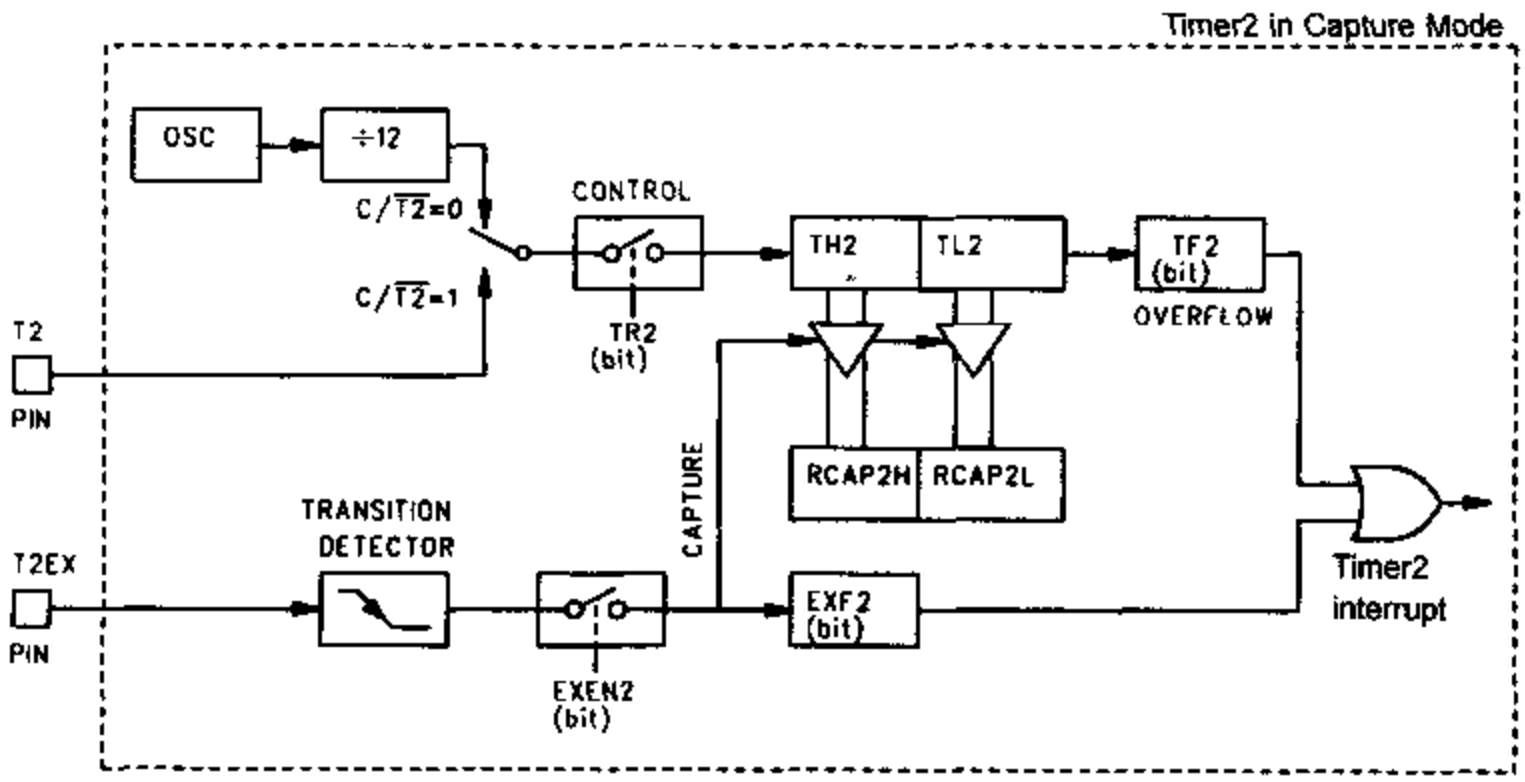


图 6-3

说明 1：虚线框内部是 8052 的 Timer2 硬件方块图，框外则是 IC 外部接过来的信号。

说明 2：Timer2 产生中断的来源有两个：TF2 与 EXF2。

说明 3：Timer2 开始计数前，要先指定 C/$\overline{T2}$ bit，并且让 TR2=1。

说明 4：Capture 要动作的条件是 EXEN2=1 而且 T2EX 由 1 变为 0 瞬间。

说明 5：8052 允许 Timer2 中断后，当 Timer2 的中断发生时，8052 要分别检查 EXF2 与 TF2，确认是外部的 Capture 信号或是 Timer2 的溢位信号，若是后者就不需要去读 RCAP2H 与 RCAP2L 两个值。

6-5　Timer2 的 Auto-reload 模式分析

在此模式下，8052 的 Timer2 仍旧做 16bit 的计数的动作，clock 的来源可以是 OSC/12 或是外部 T2（P1.0）的输入信号。当计数值超过 65535 时，产生一个溢位（overflow）的信号，这个信号会把 RCAP2 内共 16bit 值重新加载（Auto-reload）Timer2 中。这个溢位也同时会传到 TF2 位上，进而对系统产生中断要求。

外部的 T2EX 脚由 1 变成 0 时也可以重新加载（reload）计数值给 Timer2，而这个信号也会影响 EXF2 位，进而产生中断要求。8052 的 Timer2 是上数计数器，所以如果要数 10000 次后中断，我们应该把（65536-10000）=55536 这个值分别加载 RCAP2H 与 RCAP2L 中。如果我们用 Timer2 此模式做成定时中断，其方便性应该较 Timer1 的 Mode1 好，不需要一进入中断后才又重新设置上数值，而且定时中断的准确度也较高。不过，纵使如此 Timer2 最长的定时中断时间也只到 65ms 左右，如果我们要系统 1s 中断一次时，就要靠软件另外去处理了。

如果不需要外部 T2EX 产生 reload 信号时，只要事先把 TCON2 缓存器上的 EXEN2 位设成 0 就行了。

图 6-4 是 Timer2 的 Auto-reload 模式方块图。

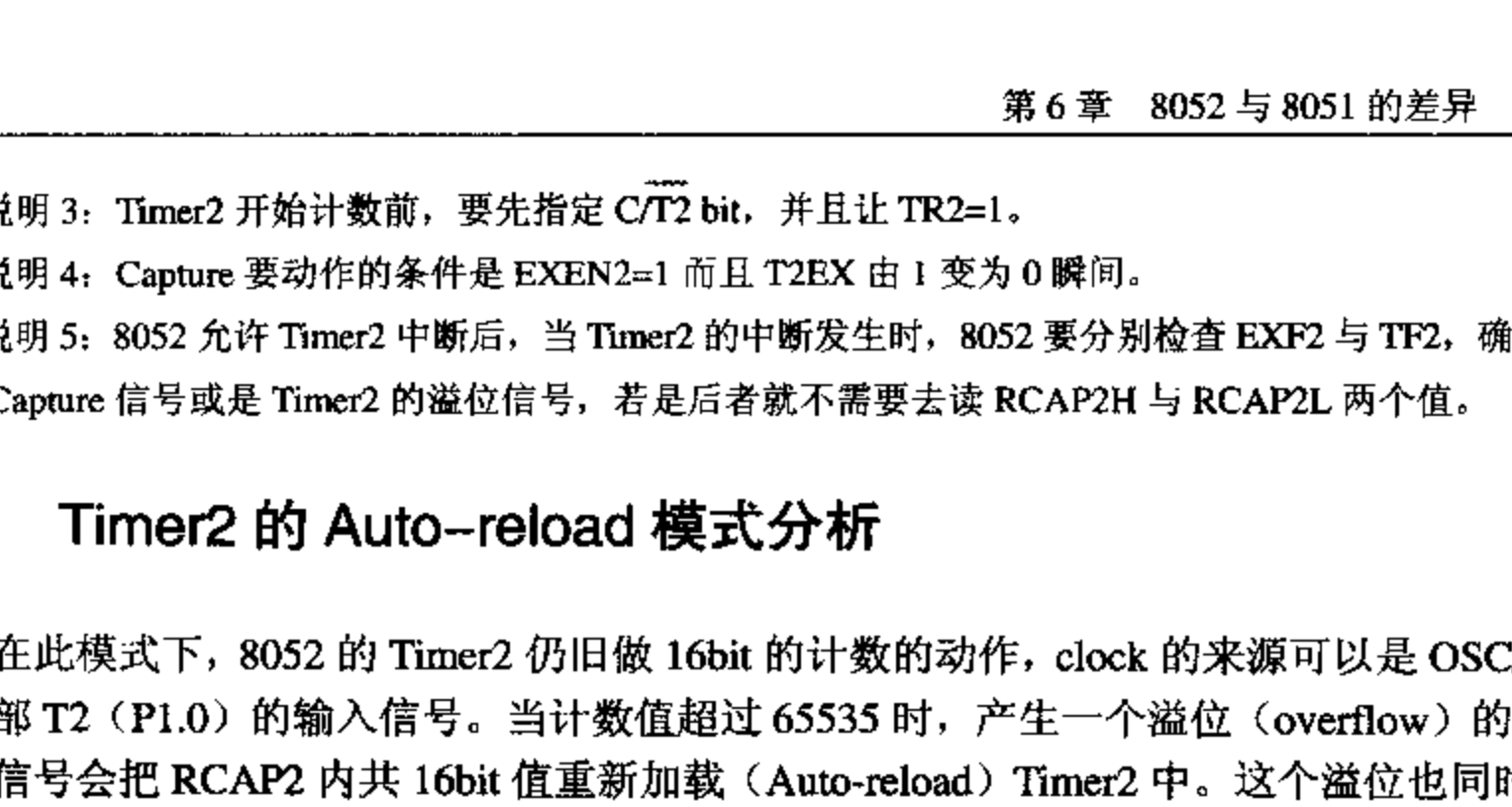


图 6-4

说明 1：虚线框内部是 8052 的 Timer2 硬件方块图，框外则是 IC 外部接过来的信号。

说明 2：Timer2 启动前要先指定 clock 的来源，亦即 C/$\overline{T2}$ 位要指定，然后再让 TR2=1，此时 Timer2 开始往上数。

说明 3：reload 的动作依据有两个 T2EX 与 Timer2 的 overflow 标志。

说明 4：Timer2 在此模式产生中断时，我们的 Timer2 中断服务程序要分别查看 TF2 与 EXF2 两个位，再来决定程序的走向。

说明 5：如果只要定时中断，但不需外部 reload 信号时，只要事先将 EXEN2 位设为 0 即可。

6-6 Timer2 的 BaudRate Generator 模式分析

8052 在此模式下，当 Timer2 计数溢位时，16bit 的 RCAP2H 与 RCAP2L 的值会自动加载到 TH2 与 TL2 上。这个溢位信号（不会产生中断）再提供给串行传输的波特率产生器用。Timer2 的来源为 OSC/2（注意是除 2，而非除 12）与外部的 T2 接脚。

8052 串行模式 1 与 3 的波特率 = 振荡频率/（32×〔65536-RCAP2〕）

上述的计算式是 C/$\overline{T2}$=0 时可以套用的公式。当 8051 做串行通信时，其传与收的波特率都由 Timer1 来决定。但是 8052 就可以选择串行通信波特率产生器的来源。若 T2CON 内的 RCLK 位等于 1 时，表示接收的波特率改由 Timer2 的波特率接手，这表示串行通信 Rx 与 Tx 的速度是可以不同，但是真实世界中这种用法应该不多。当 TCLK=1 时则传送的波特率改由 Timer2 供应。这种发与收不同速度的做法很可能会对应用程序产生困扰。比较好的做法是 Timer2 专做串行传送的波特率产生器，然后把 Timer1 移做其他用途，而非波特率生产器。

在此模式下，外部的 T2EX 信号由 1 变成 0 时，也可以对系统产生 Timer2 的中断，此信号对 Timer2 没有任何影响，严格地说，这个中断应该是所谓的外部硬件中断而不是 Timer2 溢位中断。

图 6-5 是波特率产生器模式的方块图。

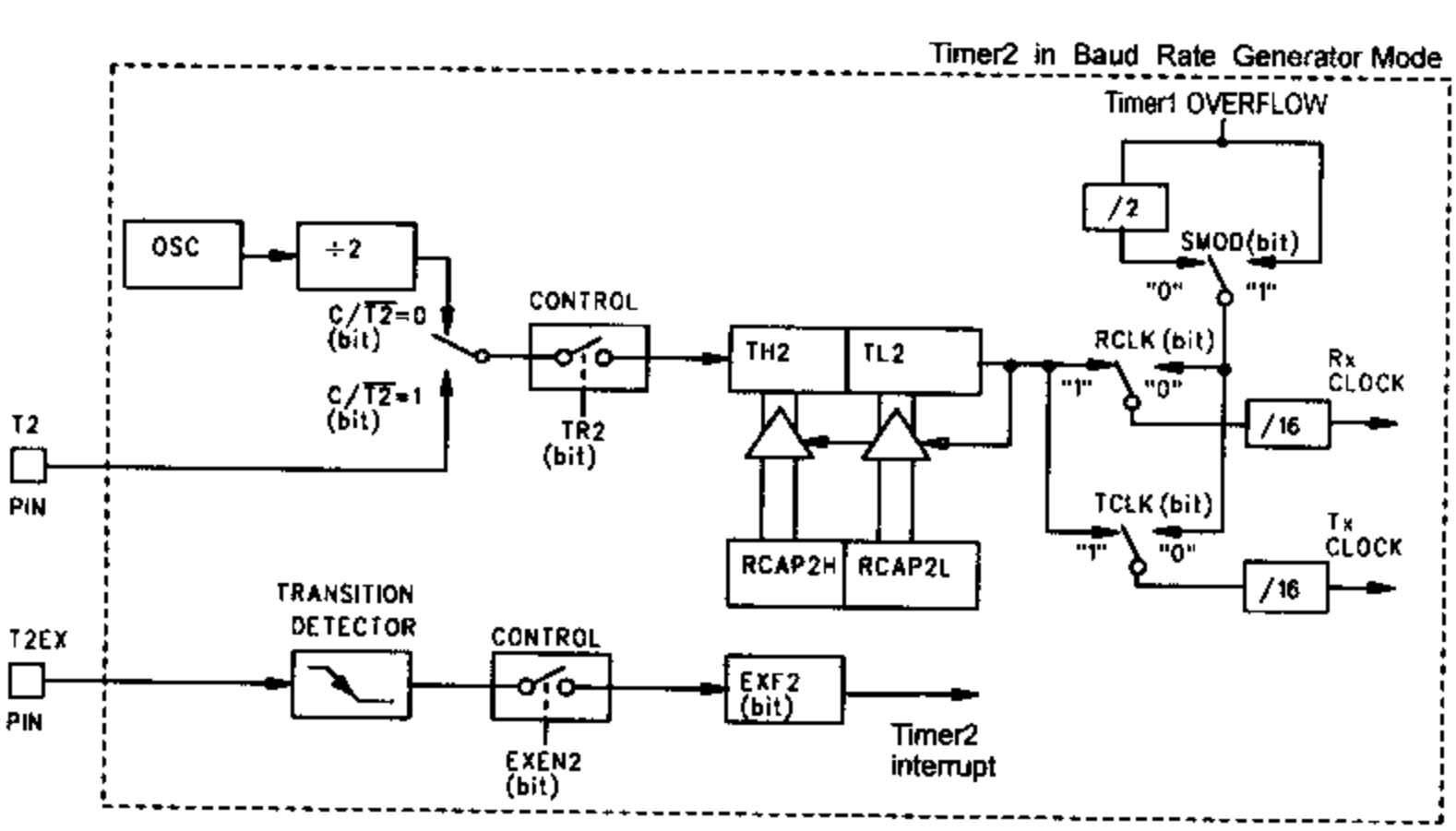


图 6-5

说明 1：虚线框内部是 8052 的 Timer2 硬件方块图，框外则是 IC 外部接过来的信号。

说明 2：Timer2 的溢位信号直接变成 reload 信号，所以在计数前，我们要把上数值放到 RCAP2H 与 RCAP2L 上。

说明 3：Timer2 开始计数前，要把 TR2 设成 1。

说明 4：本模式的波特率产生的频率较 Timer1 更有变化。

说明 5：若 EXEN2=0，则不产生 Timer2 中断。

说明 6：若 RCLK=0 且 TCLK=0，则 Timer2 完全没波特率的功能了。

6-7　AT89C52 新增的 Clock-out 功能

8052 问世也有一段时间了，有人发现 C/$\overline{T2}$ 位的功能实际上反而很少用，于是 Atmel 在其 AT89C52 的 Timer2 上就做了一些修改，在 Auto reload 模式下可以指定往上计数（Up count）或是往下计数（Down count），当溢位或借位时依然会产生中断。

在波特率产生器模式时，AT89C52 可以经由程序设置新增的 T2MOD 缓存器，把原先的 T2 输入改为 Clock 输出，让 P1.0 变成可变频率的输出点。更详细的操作说明请参考 AT89C52 的数据表。

图 6-6 是 AT89C52 Timer2 在波特率产生器模式时的方块图。

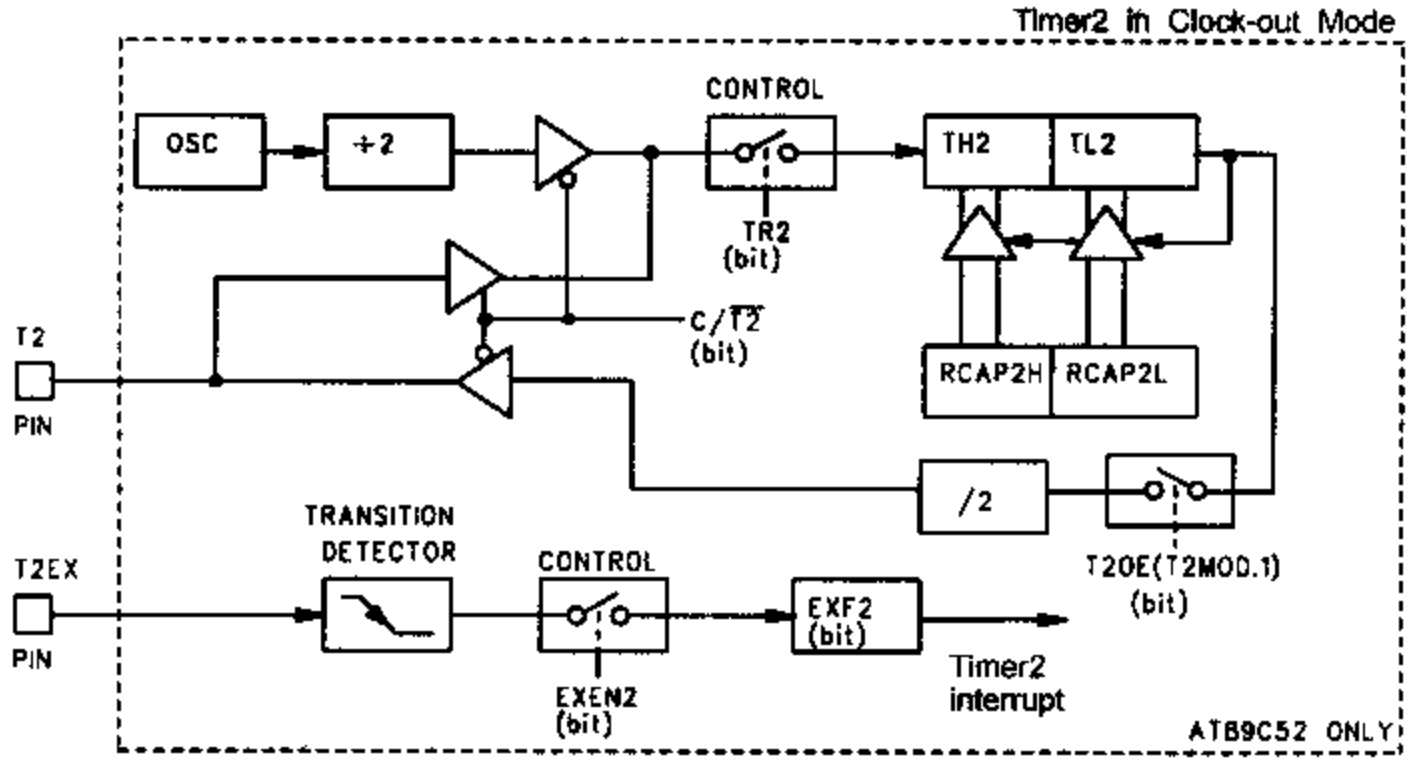


图 6-6

说明 1：虚线框内部是 8052 的 Timer2 硬件方块图，框外则是 IC 外部接过来的信号。

说明 2：当 C/$\overline{T2}$=0 且 T2OE=1 时，T2 变成方波产生器的输出，其 Duty Cycle 为 50%，这是因为 Timer2 的溢位输出经过除 2 处理后才送到 T2 端输出。

说明 3：T2EX 依然可对 Timer2 产生外部中断信号，先决条件当然是 EXEN2 要先为 1。

6-8　8KB 空间若还不够时

写汇编语言很少碰到程序空间不够的困扰，常碰到的反而是被条件转移指令搞得团团转。用 C 写 8051 单片机的控制程序时，就经常会遇到程序空间限制的问题。可能程序当中加一个浮点表达式后，程序空间就超过 8052 的敏感数字 8192，只要编译出来的二进制文件长度超过此数字，一定要对程序做“瘦身”，否则无法将程序放进 AT89C52 内部。我们的经验是：尽量将程序结构化，所有动作都标准化，这样有时候还可以节省 1/3 以上的程序空间。如果已经努力过了，可是程序空间还是没办法减下来，这时，有两个方法解决：一是选择内

含 32KB 或 64KB 程序的 8052 变种（Variant）CPU，当然你还要找到合适的刻录工具将程序刻入该 IC 当中。

另一个方法是改用外部的内存，在 CPU 外用一枚 27512（64KB）或 27256（32KB）当成程序内存，当数据空间不够时，也可以照这个方式扩充数据存储器到 64KB，这样做应该就够用了，可是 P0 与 P2 端口就无法做一般的 IO 使用。

FLAG51 就是使用外部的程序与数据空间，ROM 可达 64KB，RAM 可达 32KB，剩余的地址供 8255 及其他 IO 使用，如图 6-7 所示。

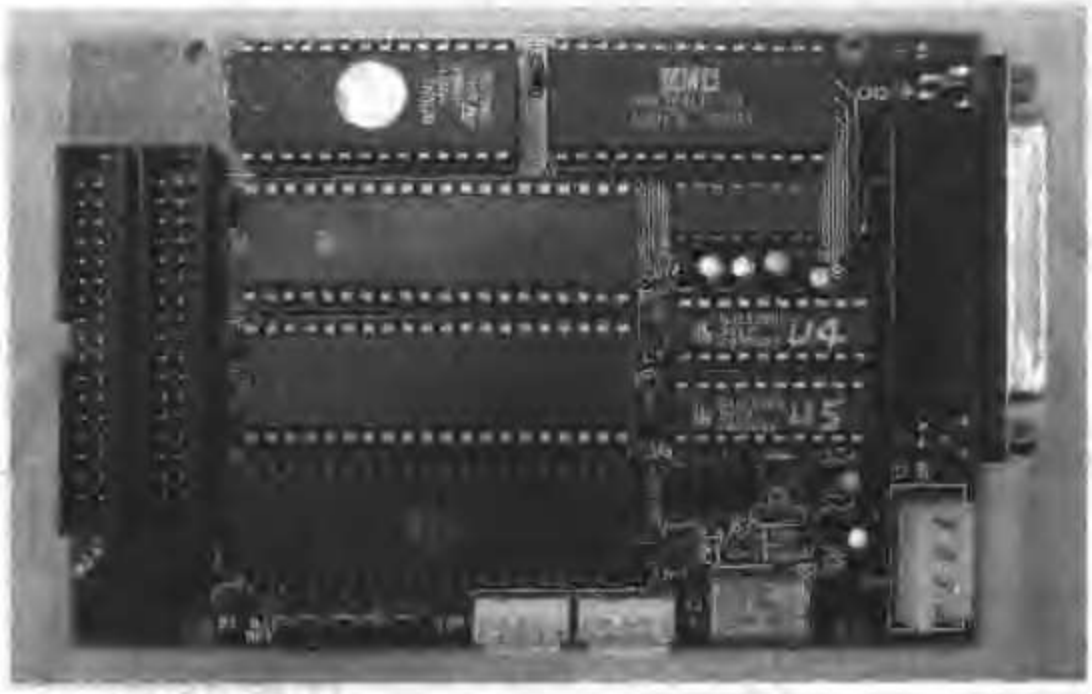


图 6-7

图 6-8 是 FLAG51 控制板上 27512 EPROM 与 6264 SRAM 的特写。

图 6-8

6-9 本章使用软件

本章使用软件如下：

（1）2500AD 8051 C Compiler and Assembler。

（2）IAR C compiler。

（3）Keil C compiler。

6-10　本章使用硬件

本章使用硬件如下：

（1）8051/8052 单片机，内含 4KB/8KB ROM。

（2）8031/8032 单片机，不含程序内存。

（3）AT89C51/AT89C52，内含 4KB/8KB Flash Memory。

（4）EPROM 27512，512K bit EPROM。

（5）SRAM 6264，64K bit SRAM。

（6）SRAM 62256，256K bit SRAM。

6-11　相关信息网站

可经由下列网站取得更进一步的信息：

http://www.chipware.com.tw：查询 8051 单片机控制板相关信息。

http://www.intel.com：查询 8052 后续发展信息。

http://www.atmel.com：查询 AT89CXX 相关单片机信息。

第 7 章

8051 的时序彻底研究

时序对大多数的软件工程师是有点陌生，程序可以运行似乎是最重要的，但是如果你设计的硬件经常会不明原因死机时，除了要怀疑程序的写法外，你可能要确认一下 CPU 时序是否符合设计规范。

7-1 时序分析的工具

对于单片机 8051 而言，时序的观察是相当困难且不易理解的，因为单片机大多数场合都使用内部的 ROM，即 8051 或 8052 之类，其外接的脚位都被用来做输出/入 I/O 用，所以很难看到 CPU 如何获取指令、执行程序甚至如何反应中断等等的时序。所幸 8051 系列的单片机系统还有使用外部程序内存的版本 8031 和 8032，才使得我们能顺利及仔细地看到 51 系列 CPU 的各种时序及信号的先后顺序。

图 7-1 是 HP 1662A 逻辑分析仪。

图 7-1

图 7-2 是 HP 1662A 逻辑分析仪所看到的列表画面。

图 7-2

图 7-3 是逻辑分析仪的 8051 POD 定义。

图 7-3

图 7-4 是逻辑分析仪的触发状态定义。

图 7-4

图 7-5 是逻辑分析仪触发条件的设置画面。

图 7-5

逻辑分析仪最难设置的就是触发条件的设置以及条件成立后储存的项目，这些部分就要麻烦软件工程师去动脑筋了。例如我们想要知道 8000H SRAM 地址被调用后，程序是否动到了 I/O 位置是 A000H 的 8255，这时触发条件就要设成当 8051 的地址值是 8000H 且是外部数据读取（read）或写入（write）时，触发条件成立并且开始记录外部地址为 A000H 的读写状态，直到逻辑分析仪内部存储器用完为止。设置这种触发条件就可以观察 A000H 所有的读写状态，进而判断程序是否有问题。

图 7-6 是 Tektronix 3054 4CH 储存式示波器。

图 7-6

Tek 3054 示波器可增加 FFT 与多种触发条件的设置等选项，如图 7-7 所示。

图 7-7

Tektronix 3054 有多种智能型触发条件的设置，如图 7-8 所示。

图 7-8

Tektronix 3054 示波器上每个 CH 的波形颜色皆不同，如图 7-9 所示。

图 7-9

观察各种硬件线路的时序，最佳的工具就属逻辑分析仪（Logic Analyzer）了，观察的轨道数应该有 40 CH 或更多为宜，因为唯有如此才能一次抓着所有 CPU 地址线与数据线的时序变化情形。第二顺位的工具就是多轨的数字式储存式示波器（Digital Storage Scope），亦可以同时观察数个重要信号的时序，由于有波形储存的功能，所以我们能设置示波器在某些条件（Conditions）下开始记录波形，事后再做详尽的分析与判断。如果手边都缺少上述两种昂贵的仪器设备，也可以利用传统的同步触发式示波器做波形分析，这类的示波器必须有连续且重复性的信号方能做信号显示，所以我们必须用程序产生一个永不终止的循环，才能在示波器上看到欲观察的波形信号了。

在本章中，我们将为各位详尽介绍 8051 的时序和波形，而被测试的线路板为 FLAG51 控制板，我们会谈到 8051 如何获取指令，如何做 I/O 动作，如何反应中断信号，甚至如何启始程序（INIT）等，当然也谈到波形测量的方法和诀窍。

图 7-10 是逻辑分析仪与 FLAG51 的连接方式。

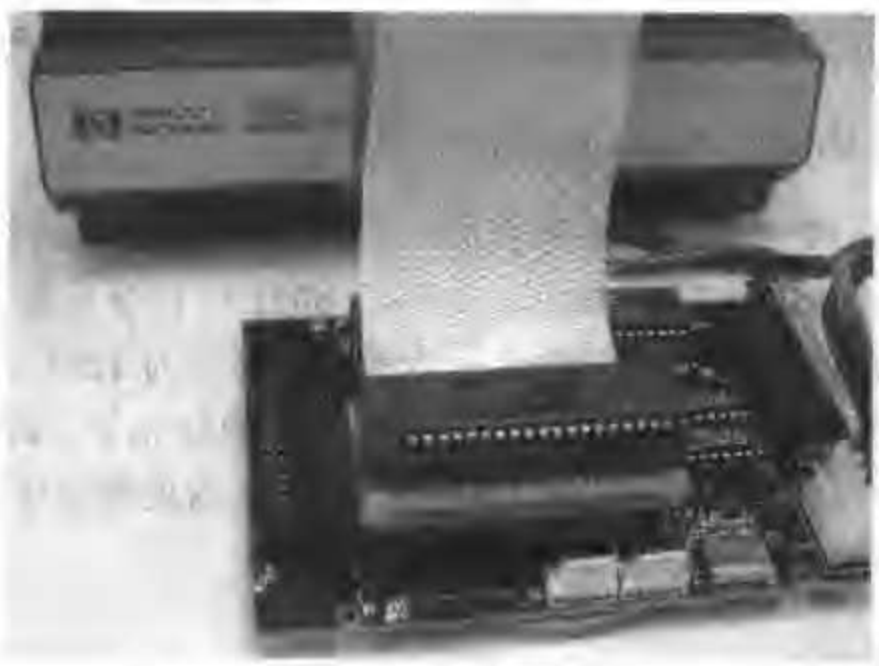


图 7-10

逻辑分析仪做状态分析时，需对各个 CPU 时序做特定的硬件处理，HP 称这个转换电路为 POD，所有 CPU 的信号先经过 POD 做缓冲处理后，再送到逻辑分析仪的输入端。除此之外，在逻辑分析仪上还要执行 8051 的特定软件，才能在屏幕上看到 8051 的所有状态数据。

图 7-11 是另一角度观看逻辑分析仪与 FLAG51 的连接方式。

图 7-11

7-2 有关 CPU 时序的关键字

在正式探讨 8051 单片机的各种硬件时序前，我们必须对几个有关的名词进行详尽的说明，这些名词包括振荡周期（Clock）、状态（State）、相位（Phase）、机器周期（Machine Cycle）和指令获取（Code Fetch）等等，这些名词将随时出现在我们介绍时序的正文中。为了解释方便起见，我们假设 8051 的 18 和 19 脚上所接的石英振荡晶体频率为 12 MHz，而实际上的 FLAG51 控制板则为 11.059 MHz，而采用此频率的主要原因是配合 RS232 的通信规模，否则就无法与 PC 或其他微电脑设备做高速的联机了。

◆ 振荡周期（Clock）

当我们将某个频率的石英晶体加在 8051 系列的第 18 和 19 脚时，就决定了该 CPU 的振荡周期，例如石英晶体的频率为 12 MHz 时，则该石英晶体每振荡一次所花的时间刚好是 83.33 ns（1/12 μs），这正是此系统的振荡周期，它是计算 CPU 执行速度的最基本单位。

◆ 机器周期（Machine Cycle）

在 8051 的系统上由 12 个振荡周期组合成一个机器周期，所谓机器周期是指该 CPU 处理 1 个指令（Instruction）所要花的单位时间，大部分 8051 的指令只要 1 个机器周期就可以完成，这即代表 8051 处理大部分指令时，每个指令所花的时间为 1μs（1/12μs×12 Clock），8051 还有一些指令需要 2 个机器周期或是 4 个机器周期才能执行完毕，此时的处理时间就要 2μs 或是 4μs，不管何种指令，8051 单片机都可在 4μs 的时间内处理完毕，若把 4μs 时间取倒数刚好是 250 KHz，这个频率正是系统最慢的反应速率。

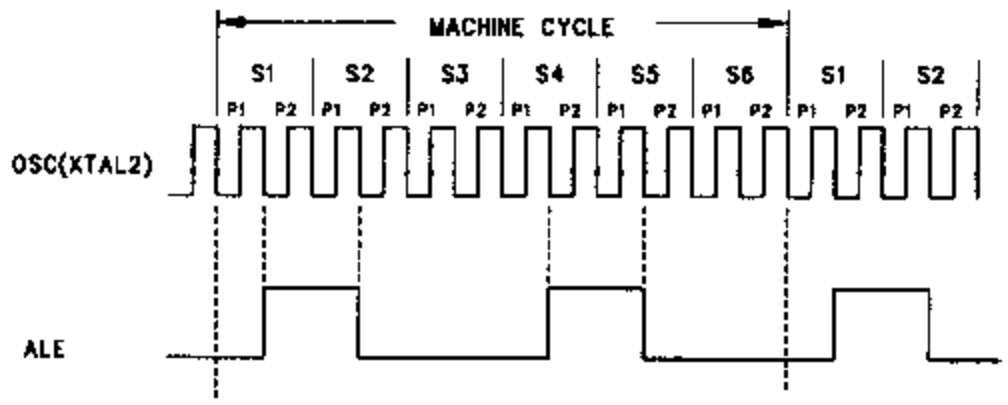
◆ 状态（State）

8051 系统中将一个机器周期再分成 6 个状态，分别是 S1、S2、S3、S4、S5 和 S6，每个状态占 2 个振荡周期（Clock），8051 指令的长度共分成 3 款，有单字节，双字节和 3 字节，在每个机器周期的 S1 和 S4 两个状态中，8051 会分别向程序内存中提取（Fetch）两个数据码，第 1 码正是指令即运算码（Opcode），当运算码进入 CPU 后，会自动判断该指令的长度为何，若是单字节的指令则忽略第 2 个运算码，即此码虽被读入 CPU 内部，但是并未加入指令的表达式中，如果是双字节的指令，第二码也是表达式的一部分，CPU 刚好用得上，免得再花一个机器周期去读程序代码，而 8051 的 3 位指令一定要花 2 个机器周期的时间去执行，而两个机器周期共有 4 次读程序数据码的机会，所以读取 3 个字节是足足有余的。

◆ 相位（Phase）

8051 系统中又将 1 个状态分成 2 个相位，分别是相位 1（Phase 1）和相位 2（Phase 2），刚好一个相位占 1 个振荡周期，在 8051 的数据中提到属于算术和逻辑方面的运算被安排在相位 1 内处理，而 CPU 内部缓存器与缓存器间的数据搬移则安排在相位 2 内处理。由于 CPU 处理这些指令的过程中并未将控制信号放到 IC 的接脚上，所以我们只好认定其动作正如数据表上所描述的了。

图 7-12 是 8051 的 clock 与机器周期，状态与相位等的相对关系图。



8051 CPU TIMING

图 7-12

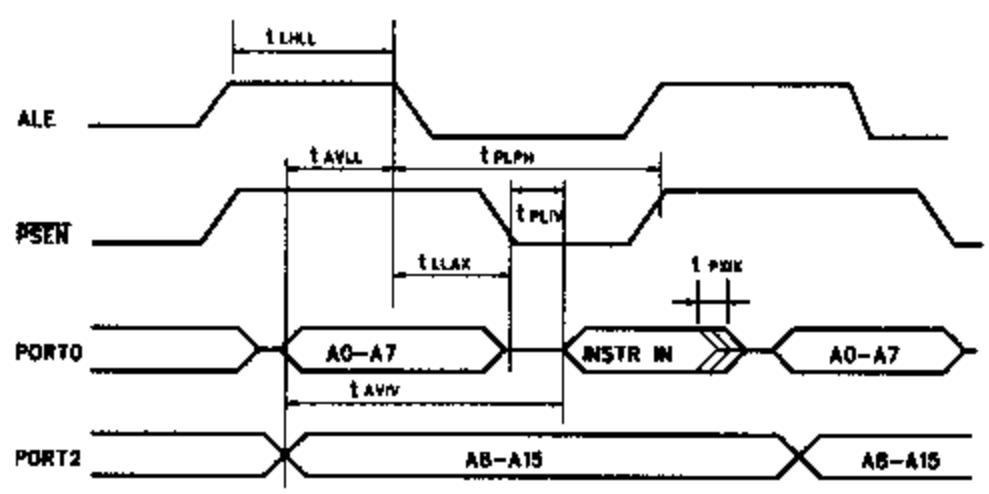
◆ 指令获取（Opcode Fetch）

当 8051 的 $\overline{EA}$ 脚为低电位时，程序重新启始后，它会由 P0 和 P2 端口送出地址值，开始读取程序内容并且执行，当 CPU 处于指令获取的状态时，除了送出 16bit 的地址值外，还伴随着 ALE 和 $\overline{PSEN}$ 信号，如果时序以 8051 的第 18 脚 XTAL 2 为参考源时，ALE 和 $\overline{PSEN}$ 都在 S1P2 时变为高电位，当 S2P1 结束时 ALE 降成低电位，而当 S2P2 结束时 $\overline{PSEN}$ 也降为低电位，在一个机器周期的另一个阶段 S4P2 开始时，ALE 和 $\overline{PSEN}$ 又同时升成高电位，在 S5P1 和 S5P2 两个相位结束时，分别将 ALE 和 $\overline{PSEN}$ 降成低电位。请看图 7-13，当 ALE 由高降为低时，程序内存上的地址值（A15-A0）必须已放妥，而当 PSEN 由低升为高电位时，CPU 由程序内存中读入一个运算码。另外我们称第二个运算码读取动作为 Operand Prefetch（运算码预先读取），采用这种预先读运算码的架构，可以有效地减少指令读取的次数和时间。

7–3　8051 程序代码的读取时序

逻辑分析仪的记录可分为两种，一是状态（State）的显示，另一个是时序（Timing）的显示，当我们在分析指令的抓取顺序时，用状态来监视会比较方便，因为逻辑分析仪可以过滤掉不要的瞬时信号，只记录有效的指令。我们可以看到 8051 是如何抓取程序代码及 IO 的存取顺序。第二种时序分析方式，可以看到所有数据与信号线的实时变化情形。由于逻辑分析仪内部的取样频率都超过 500 MHz，亦即每 2ns 就记录一次，所以可以得到非常真实的数字信号值。

图 7-13 是原厂所公布的外部程序代码读取时序图。



8051 EXTERNAL PROGRAM MEMORY READ CYCLE

图 7-13

图 7-14 是从 FLAG51 上所看到的程序代码读取波形。

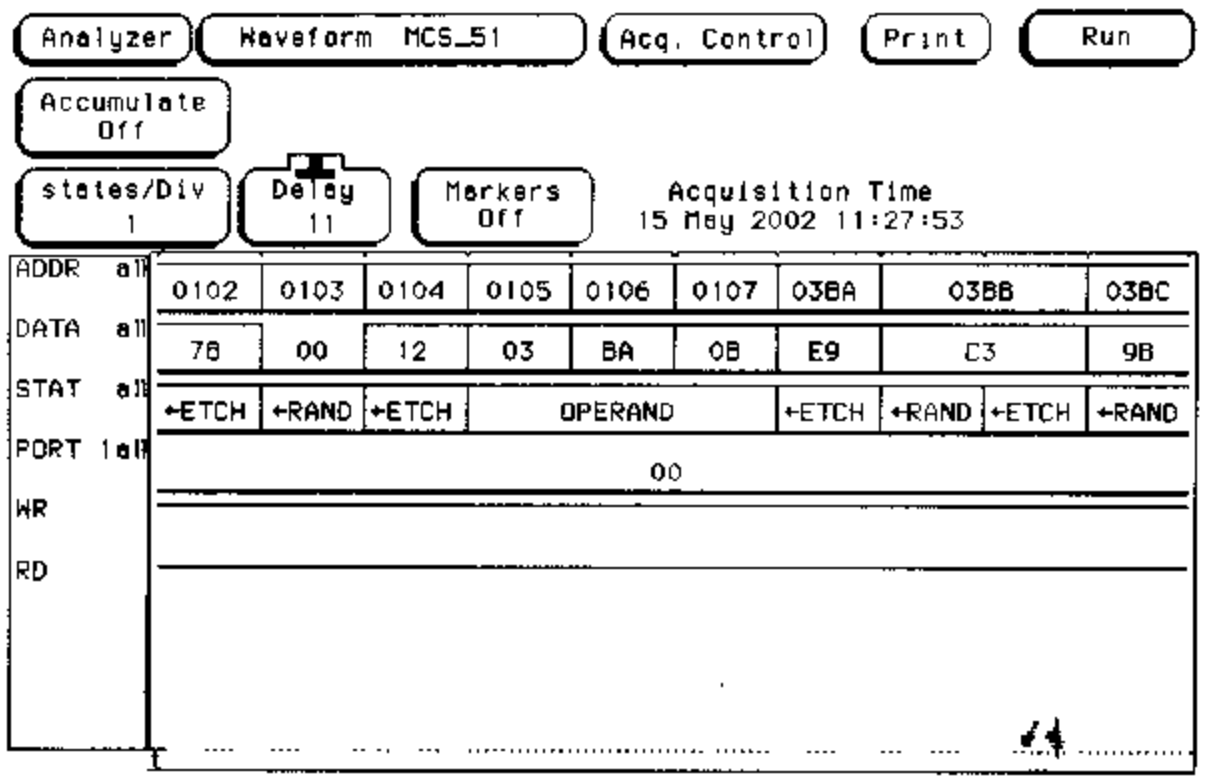


图 7-14

接下来我们举例说明 8051 是如何获取指令并且分析各项时序关系是否与原厂公布的数据相符，测试程序中只有几行指令，分别是：

```
MOV     R0,#00H     ;双字节指令，查书后的附录得知仅须一个。
                    ;机器周期就可执行完毕。
LCALL   03BAH       ;三字节指令，两个机器周期执行完毕。
MOV     A,R1        ;单字节指令，仅需一个机器周期就可执行完毕。
```

图 7-15 是由逻辑分析仪上所看到的状态分析。

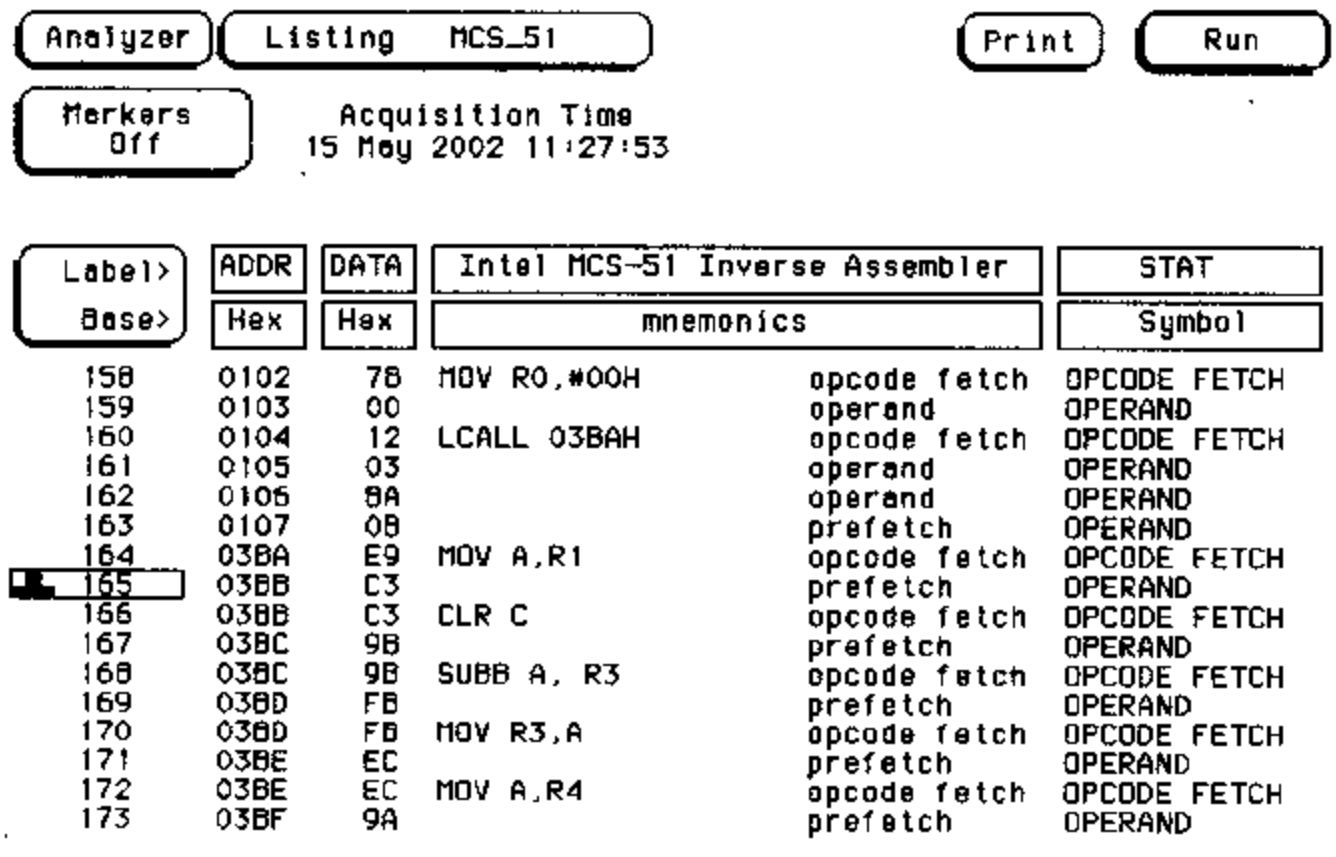


图 7-15

此时，逻辑分析仪每行所显示的是从程序存储器中抓取 1 个字节数据的状态及其指令的意义，我们可以从图中看到地址线的变化情形。同时图中每两行代表一个机器周期，所以 MOV R0，#00H 的确只花了一个机器周期就执行完毕，可是 CALL 指令却花了 2 个机器周期才做完，这是因为 CALL 指令牵涉到 SP 堆栈器的调整，并且还要从数据存储器中存入要返回的地址值，所以才会用比较久的时间来执行，由图中还可以看到 Prefetch（预取）的动作，在做 LCALL 时，CPU 最后还抓到地址 0107H 的内容 0BH，但是这个值并未用上，程序计数值 PC 直接跳到 03BAH 继续执行下去。

逻辑分析仪的另外一种表示方式是详细波形时序的展示，图 7-16 是逻辑分析仪用时序分析时，所看到的数字信号的变化。

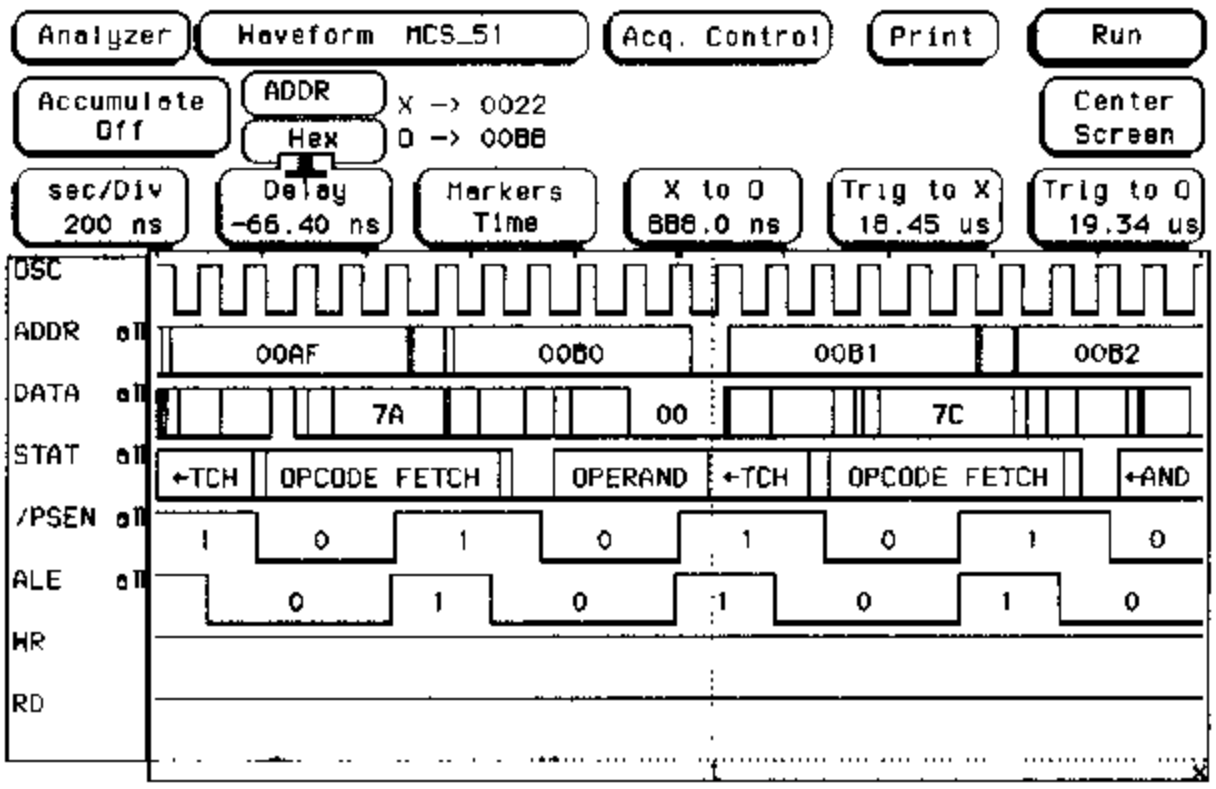


图 7-16

7-4　8051 指令长度和机器周期的关系

8051 指令的长度共分成 3 种：单字节指令、双字节指令和三字节指令，在附录中有完整的 8051 指令总整理，我们可以由其中得知，单字节指令占了大多数，其次才是双字节指令，而三字节指令只占了少部分，在附录中亦列出了各个指令所花的机器周期数，很显然地指令长度和机器周期数并没有直接的关系，例如 NOP 指令占 1 个字节，执行时只花 1μs 时间（系统振荡频率是 12 MHz 时），而 MUL AB 指令也只占 1 个字节，可是执行时却要花上 4μs 的时间，由于 8051 每个指令所执行的机器周期数有 1、2 或 4 个周期共 3 种，所以 3 种指令长度组合起来共有 3×3=9 种情况：

（1）单字节指令、单机器周期。
（2）单字节指令、双机器周期。
（3）单字节指令、四机器周期。
（4）双字节指令、单机器周期。
（5）双字节指令、双机器周期。
（6）双字节指令、四机器周期。
（7）三字节指令、单机器周期。

（8）三字节指令、双机器周期。

（9）三字节指令、四机器周期。

在前面的波形时序分析中，我们知道，8051 在 1 个机器周期内只能抓取 2 字节的程序数据，所以不可能有三字节指令能在单一个机器周期内执行完毕，因此第 7 项组合不存在，另外对照附录的指令集整理亦发现，三字节指令一定花双机器周期的时间就执行完，同时需要 4 个机器周期才执行完毕的指令只有 MUL AB 和 DIV AB 两个单位元指令，所以实际上 8051 指令的组合仅有 6 种而已，分别是：

（1）单字节单机器周期指令，如 NOP，MOV A，Rn。

（2）单字节双机器周期指令，如 INC DPTR，RET 等指令。

（3）单字节四机器周期指令，仅有 MUL AB 和 DIV AB 两指令。

（4）双字节单机器周期指令，如 ADD A，#data 和 CLR bit 指令。

（5）双字节双机器周期指令，如 PUSH direct 和 MOV Rn，direct 指令。

（6）三字节双机器周期指令，如 ORL direct，# data 和 CJNE 比较后转移指令。

接下来的几个节我们将详细地分析 8051 动作的时序关系，所展示的数据和时序图是由 FLAG51 控制板通过 HP 的逻辑分析仪而得到的，并且通过下面两种方式做示范。

◆ 展示 1：8051 状态显示模式（State Analysis），将 8051 每次获取指令（有效值）的状态显示出来。

◆ 展示 2：所有时序的完整波形图，包括系统振荡频率（OSC）、ALE 地址闩锁、地址线、数据线、$\overline{\text{PSEN}}$、$\overline{\text{WR}}$ 和 $\overline{\text{RD}}$ 等控制线。

7-5 MOVX 指令的时序及状态观察

8051 的重要时序除了程序代码的读取之外，就以外部的数据存取（External Data Memory Access）时序是最重要的，因为所有扩充的内存与 IO 都须符合时序才行。8051 做外部数据存取所使用的指令为 MOVX，我们的示范也是以该指令为主体，分别观察 Write 与 Read 的所有时序与状态。

图 7-17 是外部 Read 的状态观察。

Analyzer | Listing MCS_51 | Print | Run

Markers Off | Acquisition Time 15 May 2002 15:42:28

Label> Base>	ADDR Hex	DATA Hex	Intel MCS-51 Inverse Assembler mnemonics		WR Bi	RD He	INTO Hex
234	8000	90	MOV DPTR,#9000H	opcode fetch	1	1	1
235	8001	90		operand	1	1	1
236	8002	00		operand	1	1	1
237	8002	00		prefetch	1	1	1
238	8003	E0	MOVX A,@DPTR	opcode fetch	1	1	1
239	8004	74		prefetch	1	1	1
240	9000	00		status error	1	0	1
241	8004	74	MOV A,#55H	opcode fetch	1	1	1
242	8005	55		operand	1	1	1
243	8006	90	MOV DPTR,#A000H	opcode fetch	1	1	1
244	8007	A0		operand	1	1	1
245	8008	00		operand	1	1	1
246	8008	00		prefetch	1	1	1
247	8009	F0	MOVX @DPTR,A	opcode fetch	1	1	1
248	800A	02		prefetch	1	1	1
249	A000	55		memory write	0	1	1

图 7-17

外部读取波形分析：

（1）MOVX　A，@DPTR 指令由于有执行特别的 Read 动作，所以也要两个机器周期才能执行完毕，可是该指令却只有 1 字节长。

（2）由图中看出 MOVX　A，@DPTR 指令在第一次机器周期中，做了一次指令码 E0H 读取和一次 Prefetch 的动作，第二个机器周期则执行 Memory Read 的读入动作。由于 8051 内部的 DPTR 此时刚好值为 9000H，所以才在 ADDR 那行上看到 9000H 值出现，而 00H 则为读入的值。

（3）由时序图中可以看出 RD 的宽度约为 530ns，而整个 9000H 地址值稳定的时间超过 1μs。

图 7-18 是外部 Read 的时序观察。

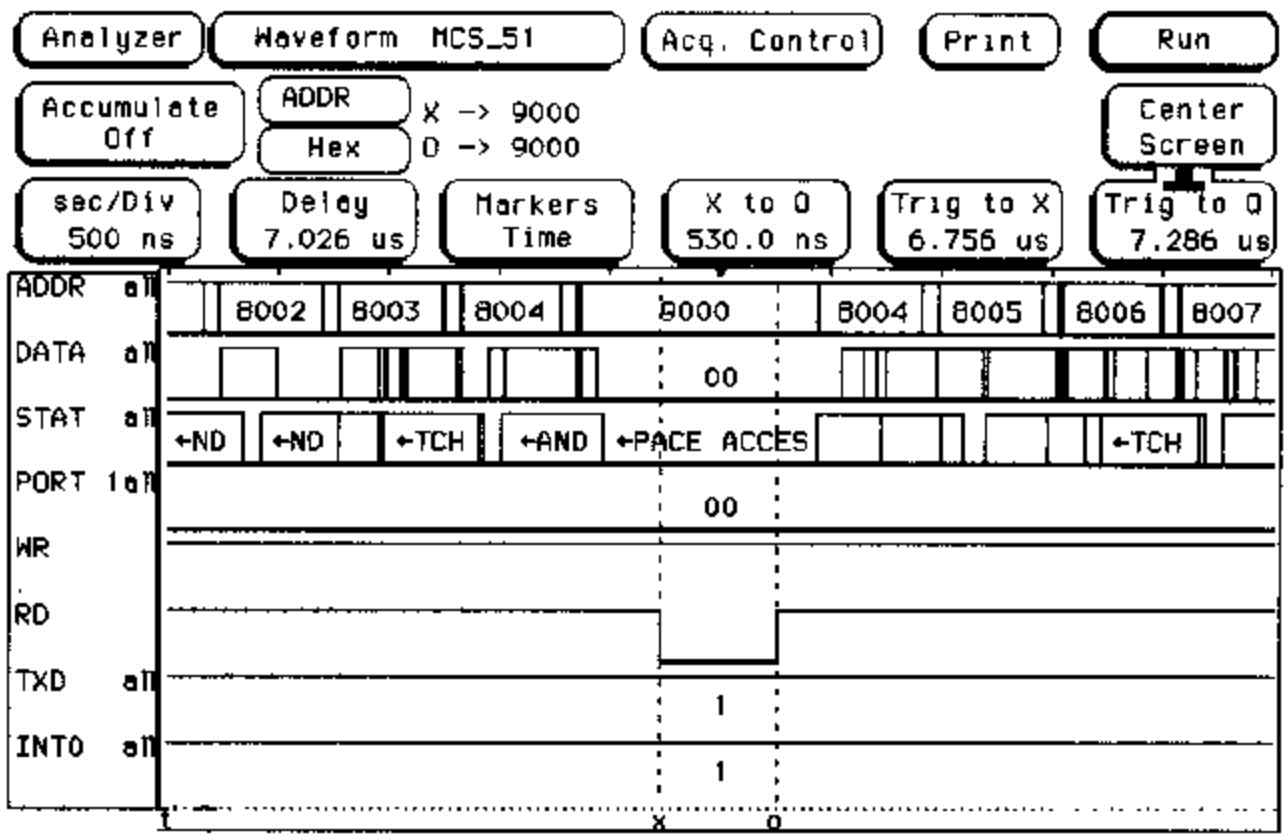
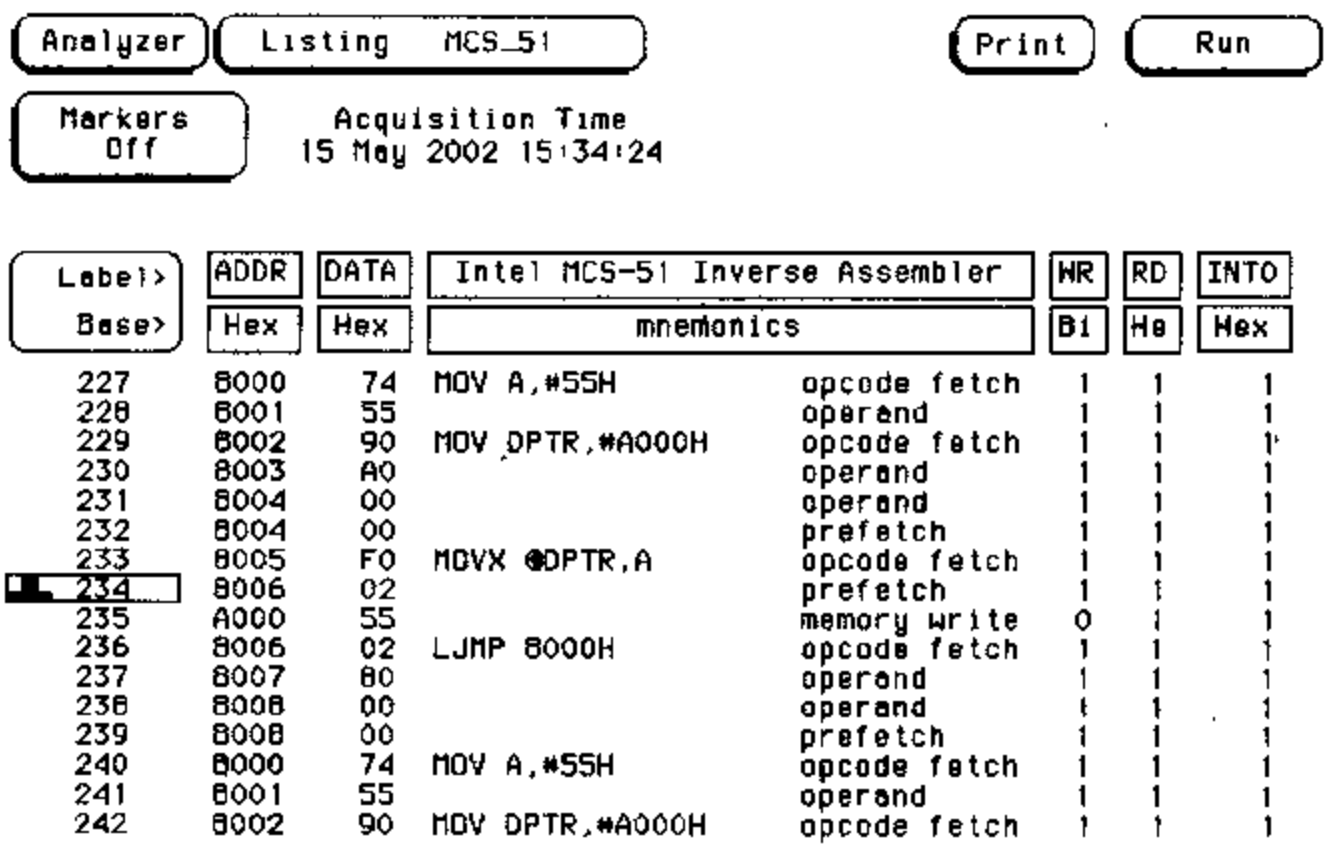


图 7-18

图 7-19 是外部 Write 的状态观察。



Label> / Base>	ADDR Hex	DATA Hex	Intel MCS-51 Inverse Assembler mnemonics		WR Bi	RD Hex	INT0 Hex
227	8000	74	MOV A,#55H	opcode fetch	1	1	1
228	8001	55		operand	1	1	1
229	8002	90	MOV DPTR,#A000H	opcode fetch	1	1	1
230	8003	A0		operand	1	1	1
231	8004	00		operand	1	1	1
232	8004	00		prefetch	1	1	1
233	8005	F0	MOVX @DPTR,A	opcode fetch	1	1	1
234	8006	02		prefetch	1	1	1
235	A000	55		memory write	0	1	1
236	8006	02	LJMP 8000H	opcode fetch	1	1	1
237	8007	80		operand	1	1	1
238	8008	00		operand	1	1	1
239	8008	00		prefetch	1	1	1
240	8000	74	MOV A,#55H	opcode fetch	1	1	1
241	8001	55		operand	1	1	1
242	8002	90	MOV DPTR,#A000H	opcode fetch	1	1	1

图 7-19

图 7-20 外部 Write 的时序观察。

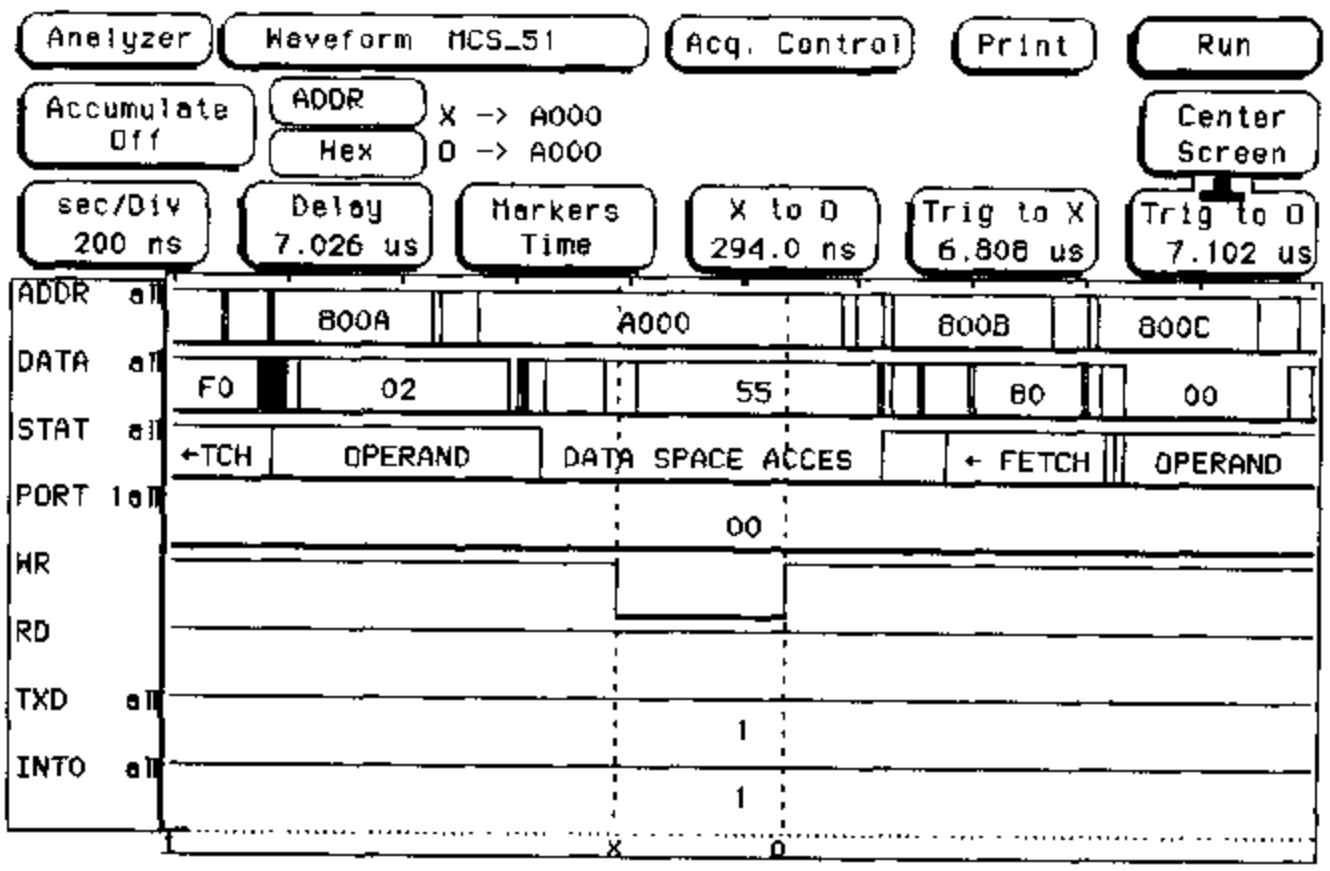


图 7-20

外部写入波形分析：

（1）MOVX @DPTR，A 指令是将 A 值放入由 DPTR 所指定的外部存储器地址中，它是属于一个数据写入（Write）的动作，它也需要两个机器周期才能执行完毕。

（2）在 MOVX @DPTR，A 指令的两个机器周期中，第一个机器周期包括一次指令码提取和 Prefetch（预取）两项动作，第二个机器周期则分配给 CPU 对外部内存做写入的动作。

（3）由时序图中可以看出 WR 的宽度约为 480ns，而整个外送 A000H 地址值稳定的时间接近 1μs。

（4）8051 每个机器周期（以 12 MHz 的系统频率而言）为 1μs，而一个 MOVX 指令约 2μs 时间，把此时间取倒数为 500 KHz，这个频率是 8051 对外部数据做存取动作的最快速率，换句话说，若有人想对外部的外围设备做 500KB 以上的数据传输率，这将是 8051 无法办到的。

（5）当 8051 做 MOVX 动作时，信号时序会和一般程序代码读取的时序不同，请特别留意其间的差异点。

（6）下两页将观察 ALE 与 $\overline{\text{PSEN}}$ 信号在外部 Read 与 Write 时的变化。

图 7-21 是做外部 Read 时，8051 ALE 与 PSEN 信号的变化情形。

请留意 ALE 信号在 RD 有效时是保持低电位的，而 $\overline{\text{PSEN}}$ 一直保持在高电位，代表此时不是程序代码读取的状态，在 RD 由低变成高的上升缘阶段读入外部内存的数据。

图 7-22 是做外部 Write 时，8051 ALE 与 PSEN 信号的变化情形。

请留意 ALE 信号在 WR 有效时是保持低电位的。当 WR 上升缘出现时，外部的内存必须准备妥当，等待 CPU 送出来的累加器值，由图中我们亦可看到在 WR 尚未变成低电位前，资料在线早已排妥 55H 值，准备让外部内存或 IO 外围接收。

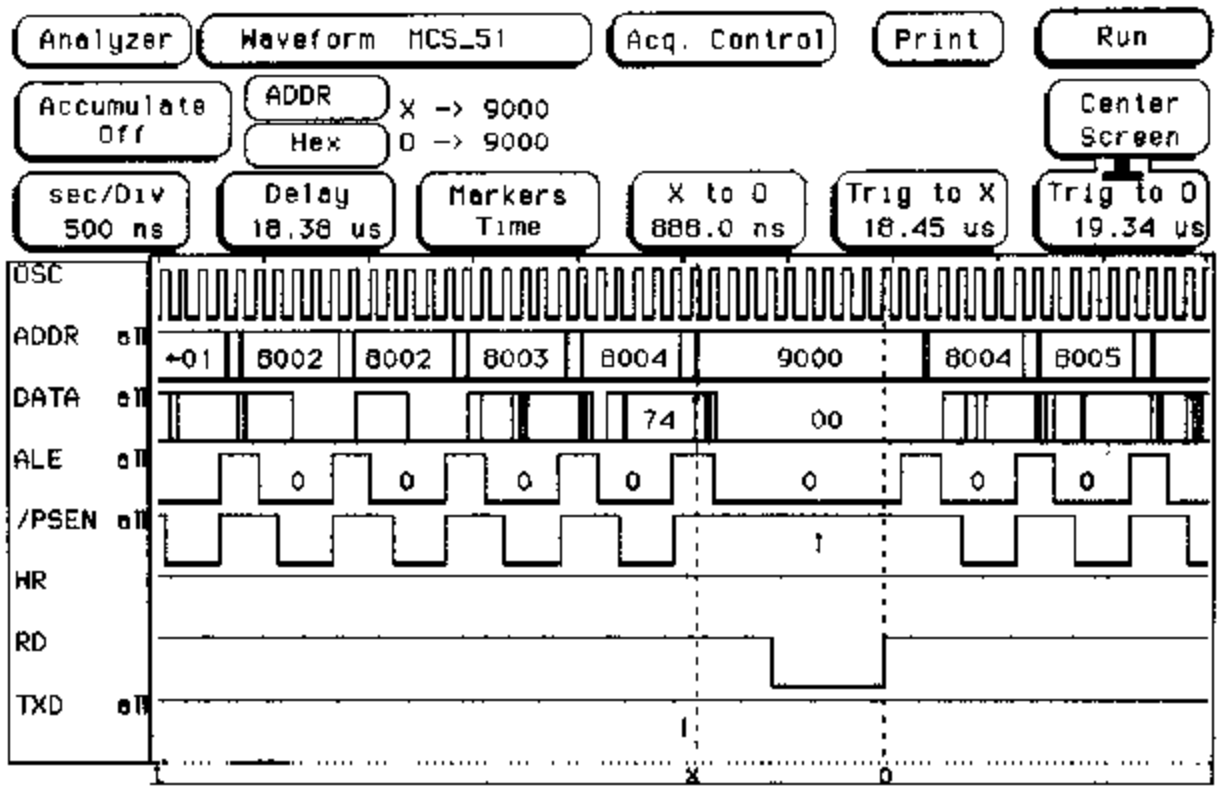


图 7-21

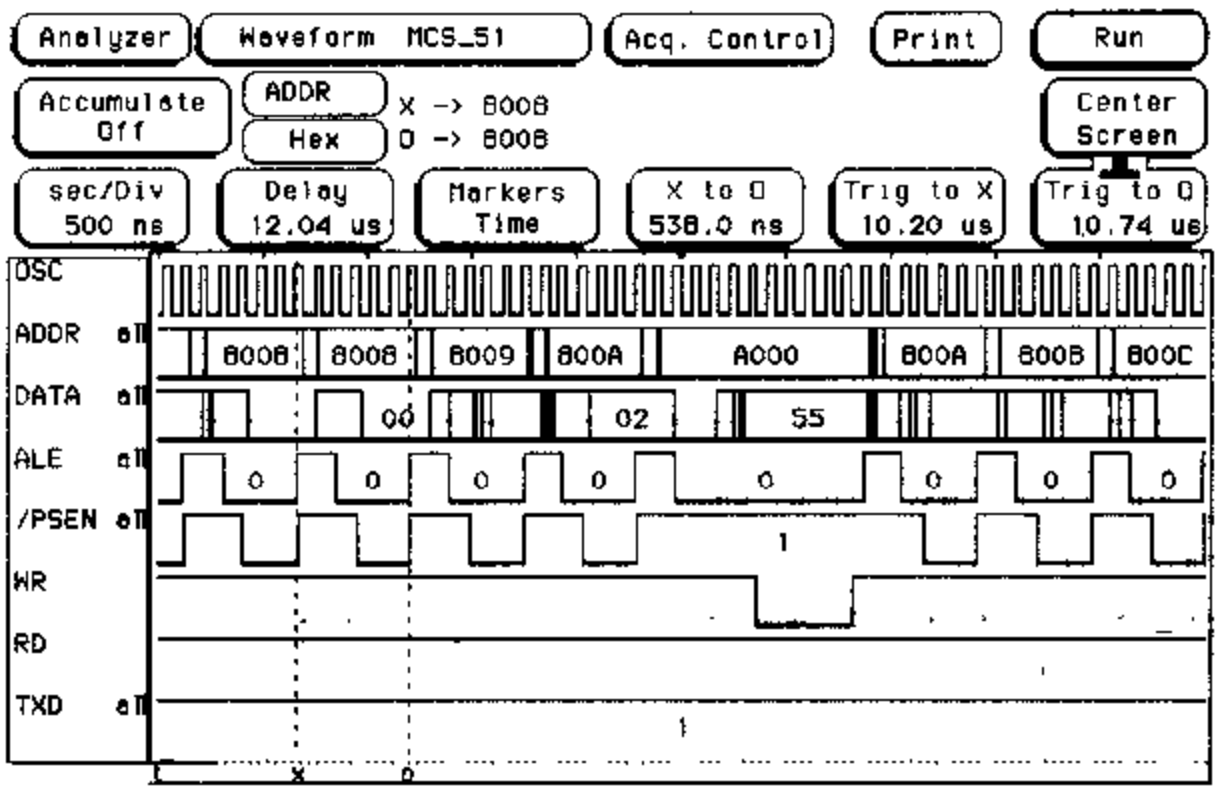


图 7-22

7-6　Dallas 80C320 的波形观察

我们经常看到许多号称是高速的 8051 单片机，只要更换 CPU 后，执行速度就是原先的三倍或是更快，在本节里我们将以 Dallas 80C320 为例，分别观察其程序代码的读取以及外部 I/O 读写的波形。从程序代码的读取波形看来，80C320 确实有比较快，其一个机器周期只要 4 个 clock 就行了，比原先 8051 的 12 个 clock 快三倍。

不过要留意的是：许多号称高速的 CPU，其实是把石英晶体的频率经由 PLL 相锁环路提高到原来的四倍（以现在的半导体制程来讲是可能的），再修改部分硬件架构，就可以把执行速度提高到原来 8051 的五倍以上。

图 7-23 是 80C320 的外部程序读取时序。

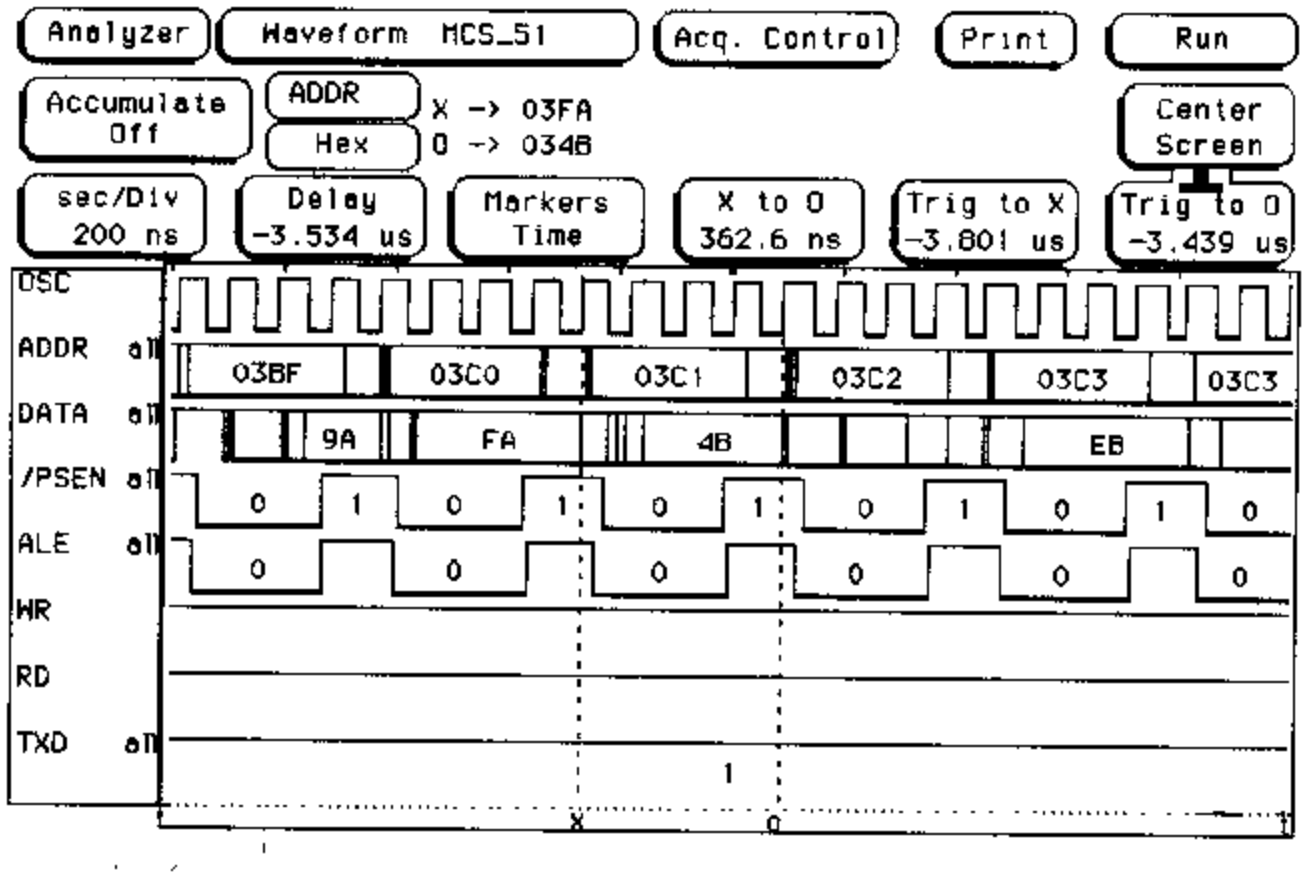


图 7-23

图 7-24 是 80C320 的外部 Read 的时序分析，其 RD 宽度约是 358ns。

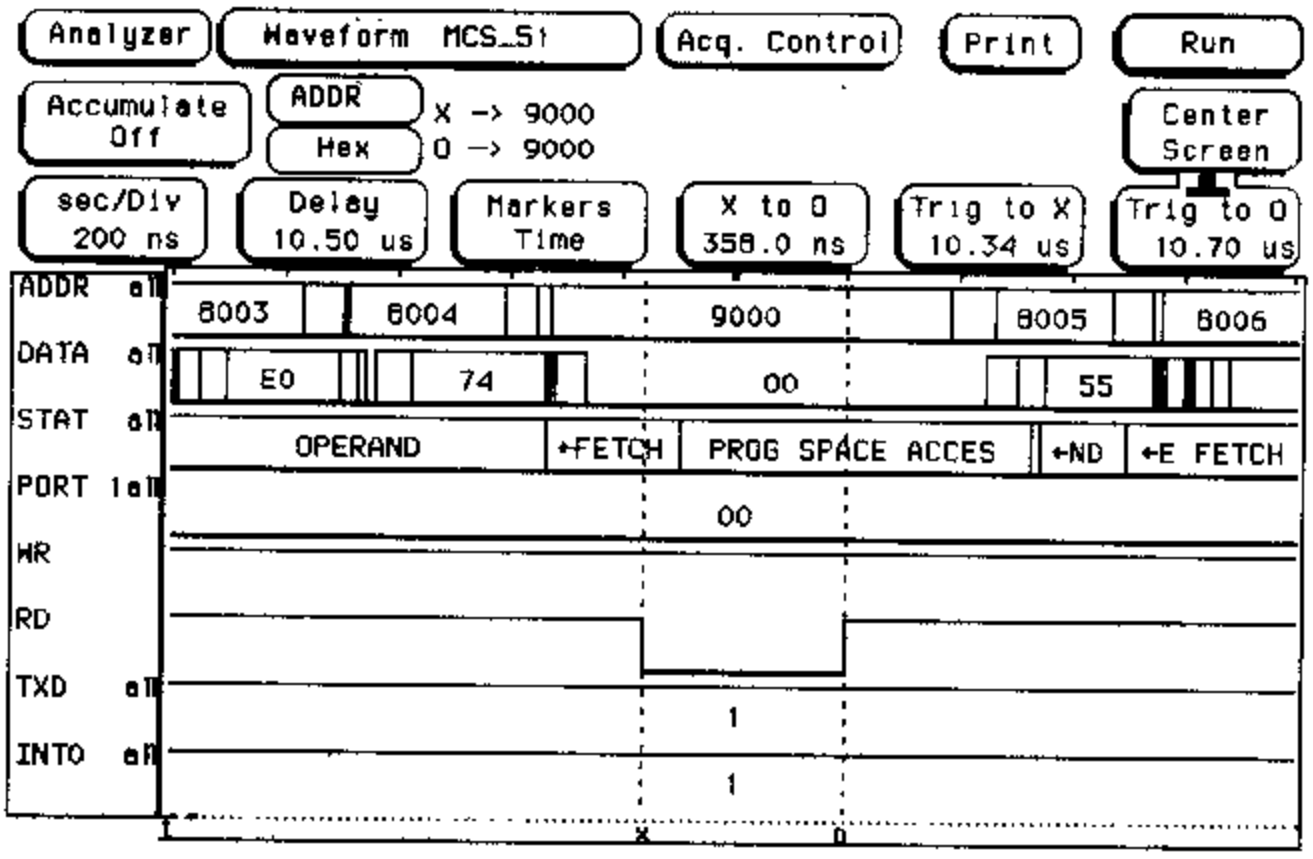


图 7-24

图 7-25 是 80C320 的外部 Write 的时序分析，WR 宽度为 294ns。

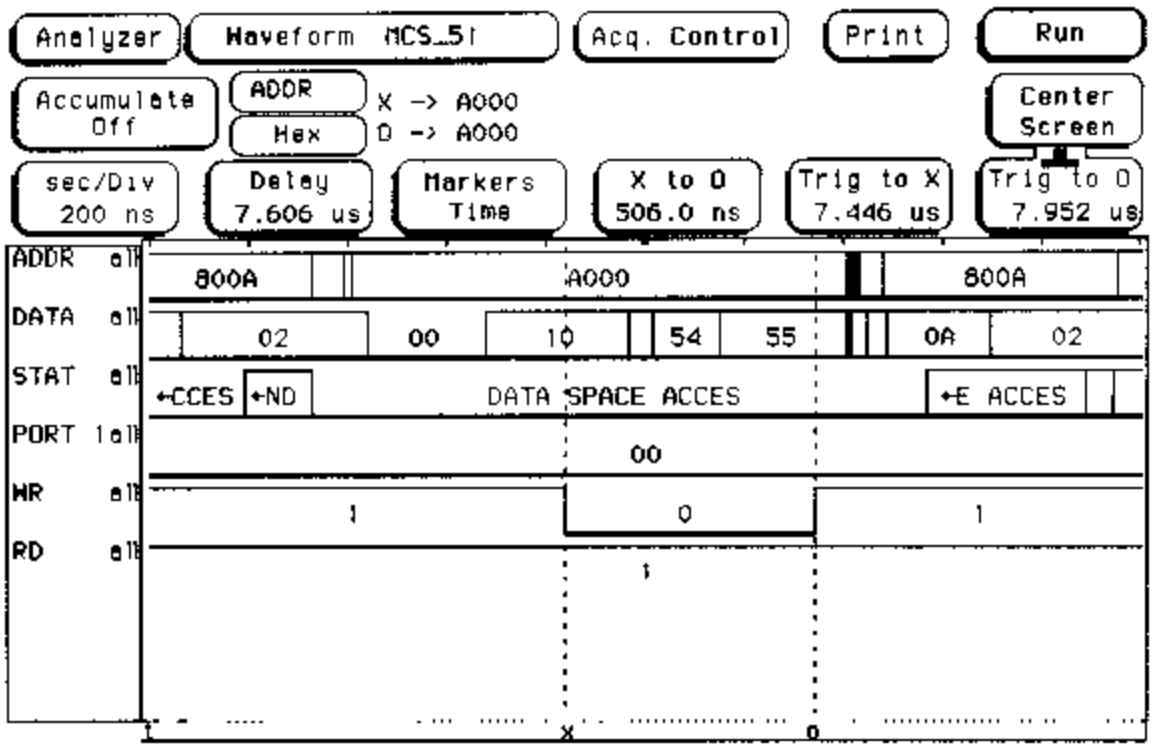


图 7-25

7–7　本章使用软件

本章使用软件如下：

（1）2500AD 8051 C Compiler and Assembler。

（2）IAR C compiler。

（3）Keil C compiler。

7–8　本章使用硬件

本章使用硬件如下：

（1）8051/8052 单片机，内含 4K/8KB ROM。

（2）8031/8032 单片机，不含程序内存。

（3）AT89C51/AT89C52，内含 4K/8KB Flash Memory。

（4）HP1662A 逻辑分析仪。

（5）Tektronix 3054 4CH 高速数字示波器。

（6）FLAG51 单片机控制板。

7–9　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询 8051 单片机控制板相关信息。

http://www.agilent.com：查询逻辑分析仪相关仪器设备信息。

http://www.tek.com：查询高级示波器相关仪器设备信息。

http://www.intel.com：查询 8051 的时序信息。

http://www.atmel.com：查询 AT89CXX 相关单片机信息。

第 8 章

FLAG51 开发过程

学计算机的人如果“吃软不吃硬”，实在很可惜，玩硬件却不碰单片机与微处理机，那也是颇遗憾的事，而要学习单片机，就要由目前最热门的 8051 单片机开始。

最近许多友人一见面就问：我要学 8051 单片机，请问如何速成？通常我都会笑着回答：学问是不可能速成的，经验也不是三两天就可以拥有的。如果真的想学单片机，可以找几本好书及教材，先学习基本的汇编语言及其基本的应用，然后找一个单片机的制作专题，从头到尾亲自走一回，绝对可以学到单片机的精髓。如果你对单片机与微处理机已有了基本的概念时，可以找个较复杂的制作专题来考验一下自己的能力，反之则应限定在某种基础应用上，千万不要目标定得过高，造成一直无法达成目标。

8-1 FLAG51 的系统开发过程

学习 8051 单片机的方式有两方面，第一是将我们写的测试程序下载到 RAM 区，执行后看看结果如何，这类的程序空间都在数千字节以内。另一方面是学习如何写监控程序，前者正是我们通称的应用程序，后者则是系统程序，较正确且稳当的学习过程是先学应用程序，然后弄懂系统程序的各项运作情形，最后才又来写应用程序。市面上讲 8051 单片机的书籍百分之九十都谈应用，很少提到如何规划及整合 8051 单片机系统，本章我们就以 FLAG51 控制板的开发过程为实际案例，来和大家经验分享。

8-2 FLAG51 的构想、设计、布置、整合

由于单片机教学的缘故，发现讲 8051 指令时最令学生感到听而无味，也不太可能叫每个学习者都去买 MICE（“全友”公司的 CPU 在线仿真器）。于是作者利用接项目的工作之余规划一个可以与 PC 联机的基本监控程序架构，前前后后又经过一年左右的修正，才完成整个程序的排错动作，不过此时的系统是架构在其他的硬件线路上，体积过大携带非常不方便，而且线路也过于复杂，不利于教学与学习，于是才有重新规划一片单片机控制板的

念头。

初步的想法是设计一块控制线路板，体积与耗电越少越好，并且有某种程度的技术前瞻性，于是特地将可程序化 PEEL 组件加入电路中，后来由于 PEEL 零件严重缺货，我们改用功能相同的 GAL 组件替代，线路上不需做任何修改。这类组件的用法是一般教科书完全没提到的，但是此类组件却到处可见。FLAG51 的线路经过构想及初步的布局后，我们首先完成了手绘的线路图，请看图 8-1，手绘线路图其实已经是相当完整的单片机电路了。

学习硬件的基本设备如下：

- 数字电表。
- 逻辑笔。
- 电铬铁及焊锡。
- 电源供应器。
- 剪线剥线钳。
- 电线及所需零件。
- 基本零件 IC 的资料手册。

为了证实线路的正确性，也抽空完成了图 8-1 的雏型电路板（prototype）。请看图 8-2 的照片，对线路若相当熟悉有把握时，当然可以省略此步骤，不过个人认为焊接雏型电路板和写程序一样，可以训练头脑的思绪及判断力，并且能增加对各种困难的排除能力，不相信的读者可以试试看。不过，焊接一块类似 FLAG51 控制板的电路可能要花上十小时的时间，若每晚不看连续剧花两小时焊接时，也要一个星期才能完成。但我们相信能顺利完成雏型电路板的读者也能更顺利地学好 8051。

当雏型电路板在制作焊接的同时，我们把图 8-1 的线路交到专门规划 PC 板的计算机公司，请他们将此线路图输入计算机。图 8-4 是用专门做零件布置的软件（PCAD）所绘出来的线路图，通常我们以此图与原线路图做核对。图 8-3 则是初步的零件排列与接线图，我们会特别仔细检查每个零件所占的空间是否足够，尤其是每个接头在装卸的过程中是否会与其他零件相碰，若等到 PC 板制作好才发觉时，可能又要耗掉至少一个月的时间去修正这些错误，若没有任何错误时，就可通知计算机公司将图 3 制成洗 PC 板专用的 1:1 底片，以及产生线路板上所有零件的钻孔磁盘（通称 NC DRILL FILE）。图 8-5 则是其中的一张 1:1 底片图，只要有了底片与钻孔数据，就可以委请 PC 板制造厂完成 FLAG51 的 PC 板了，交货期通常在两周到一个月以内。图 8-6 则是尚未插上任何零件的 PC 板图。图 8-7 则是完成品的实际照片图。从有线路的构想到完成图 8-7 的 FLAG51 控制板，大概要花上四个月的时间，若加上程序撰写和系统排错，则整个开发过程已超过半年的时间了，这和抄袭别人的线路板是大不相同的，至少所有的技术与资料都可以完全掌握，程序也可以随时轻易地修正。

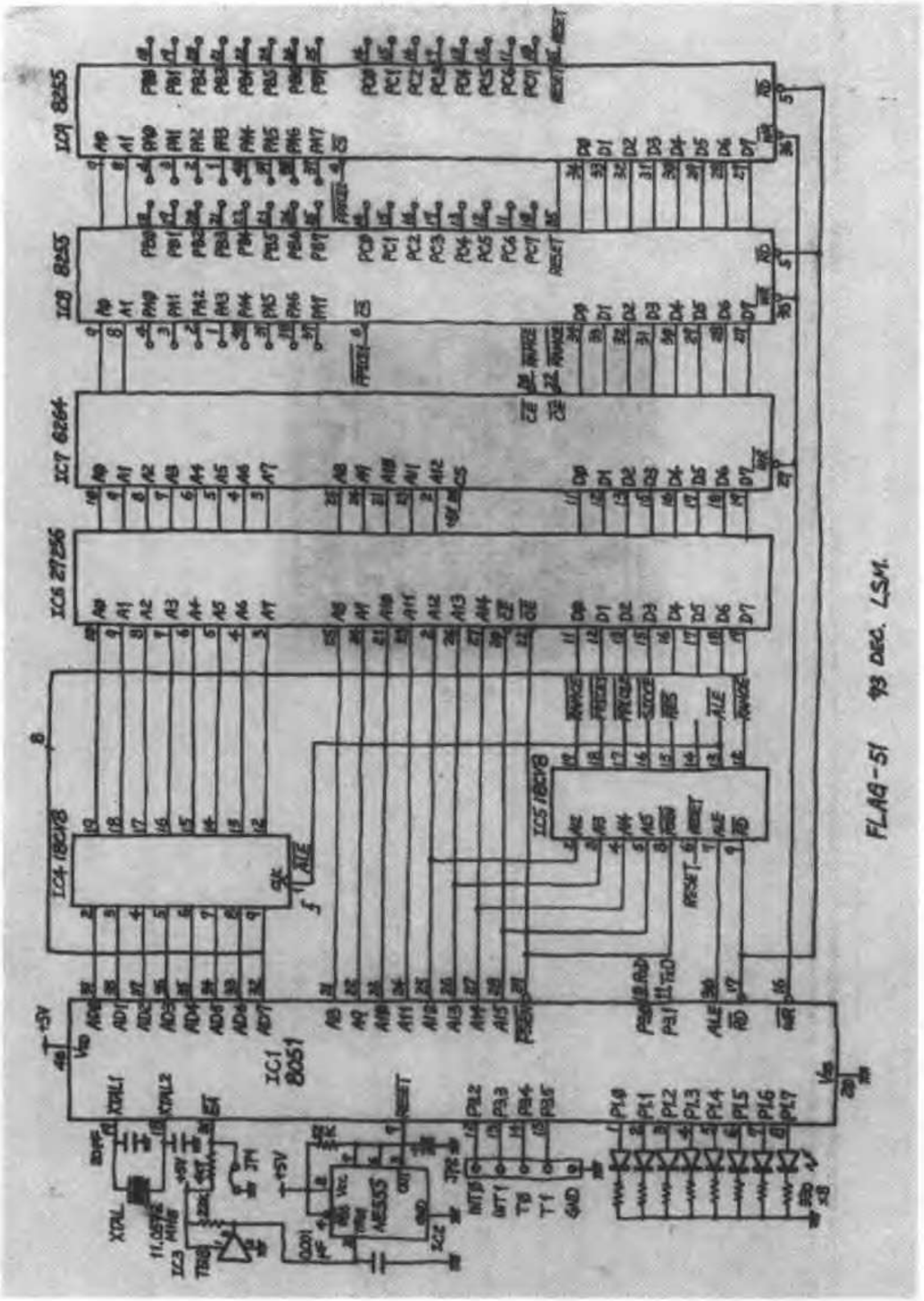


图 8-1

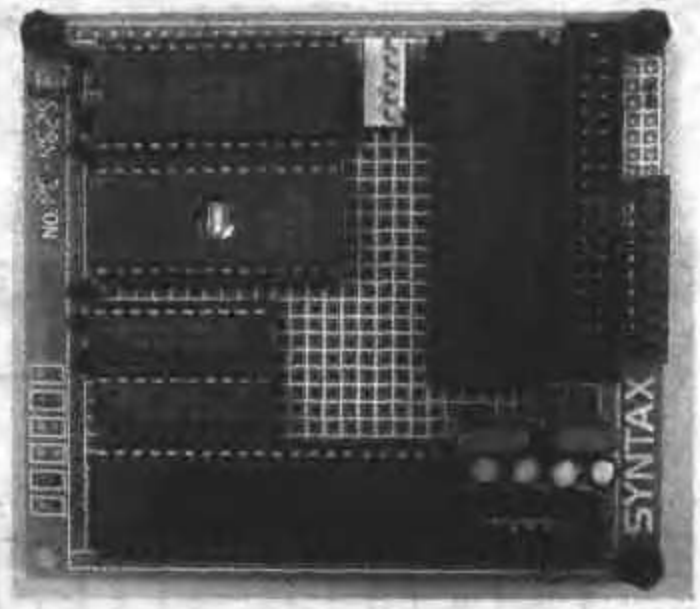


图 8-2

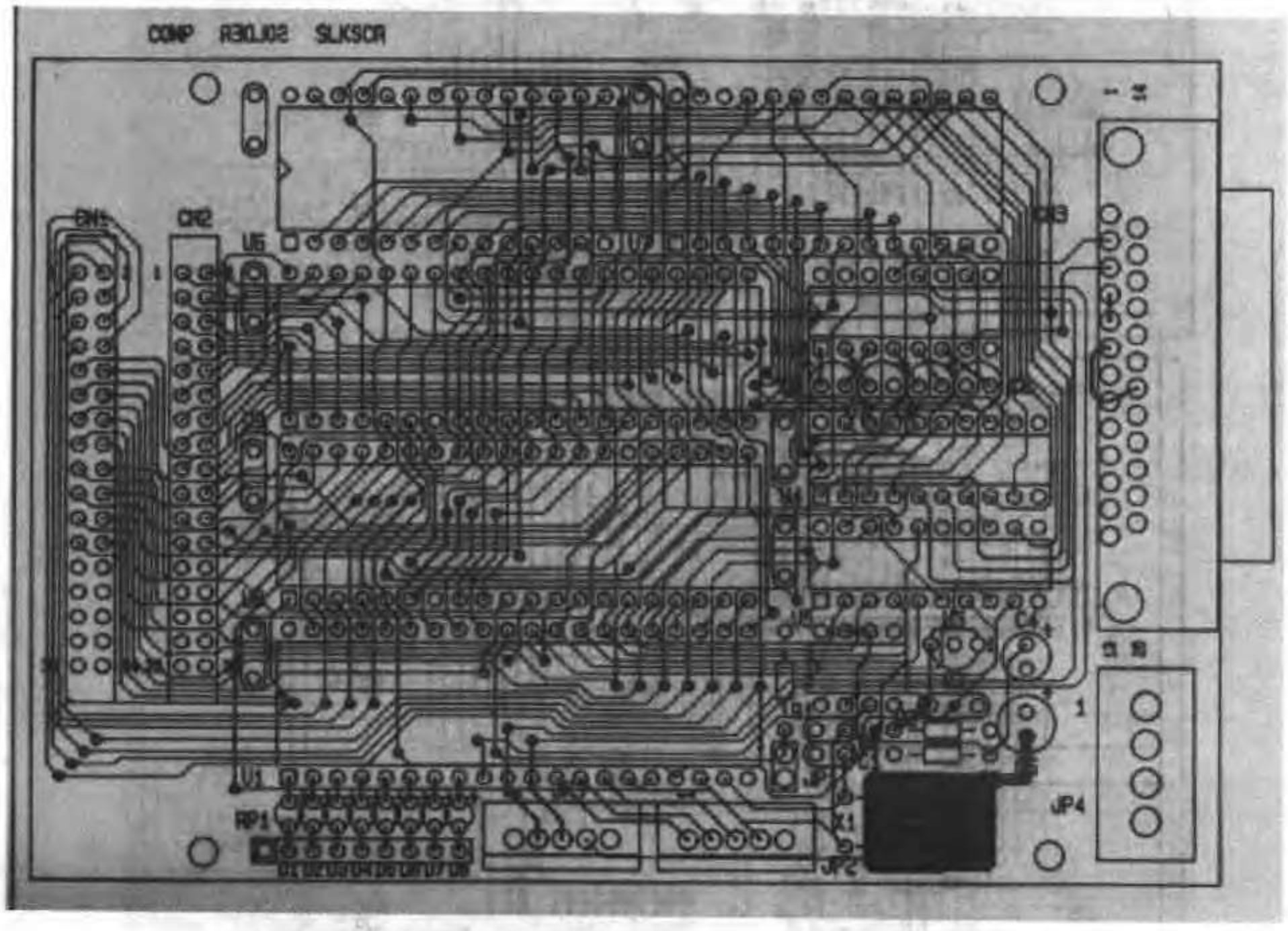


图 8-3

注：原图系以红、蓝、黑三色区分零件面线路、焊接面线路与文字标识。

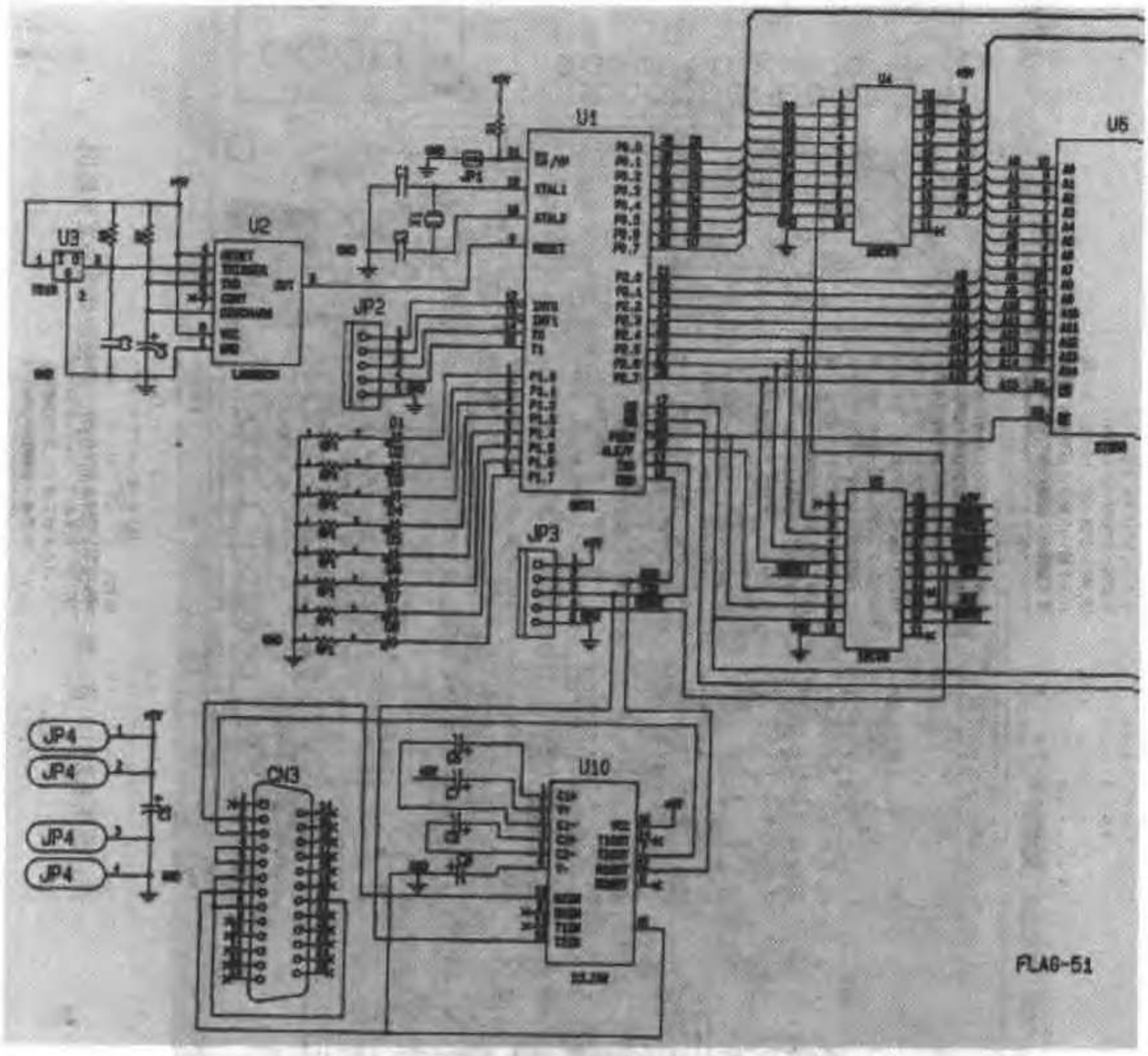
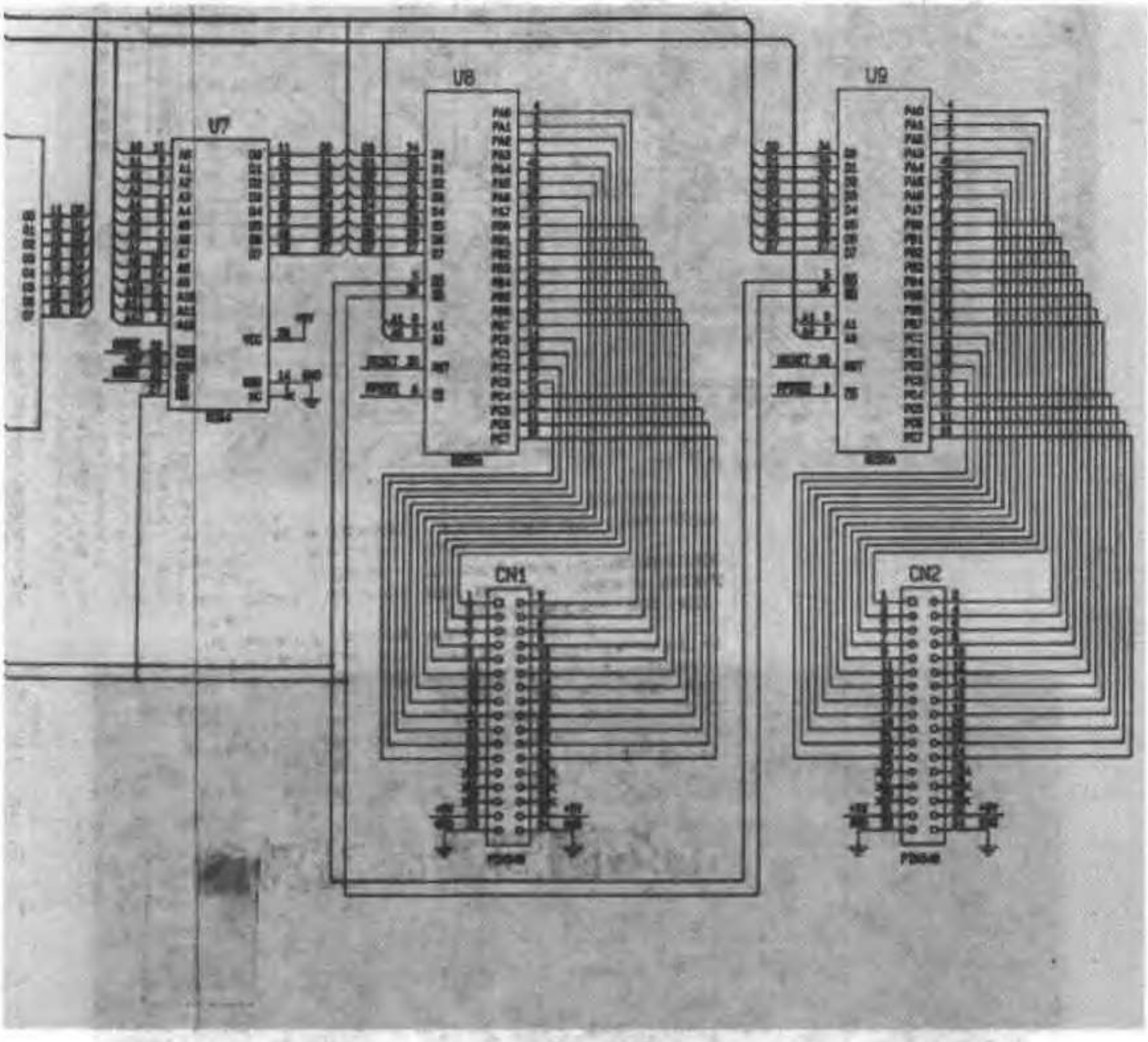


图 8-4



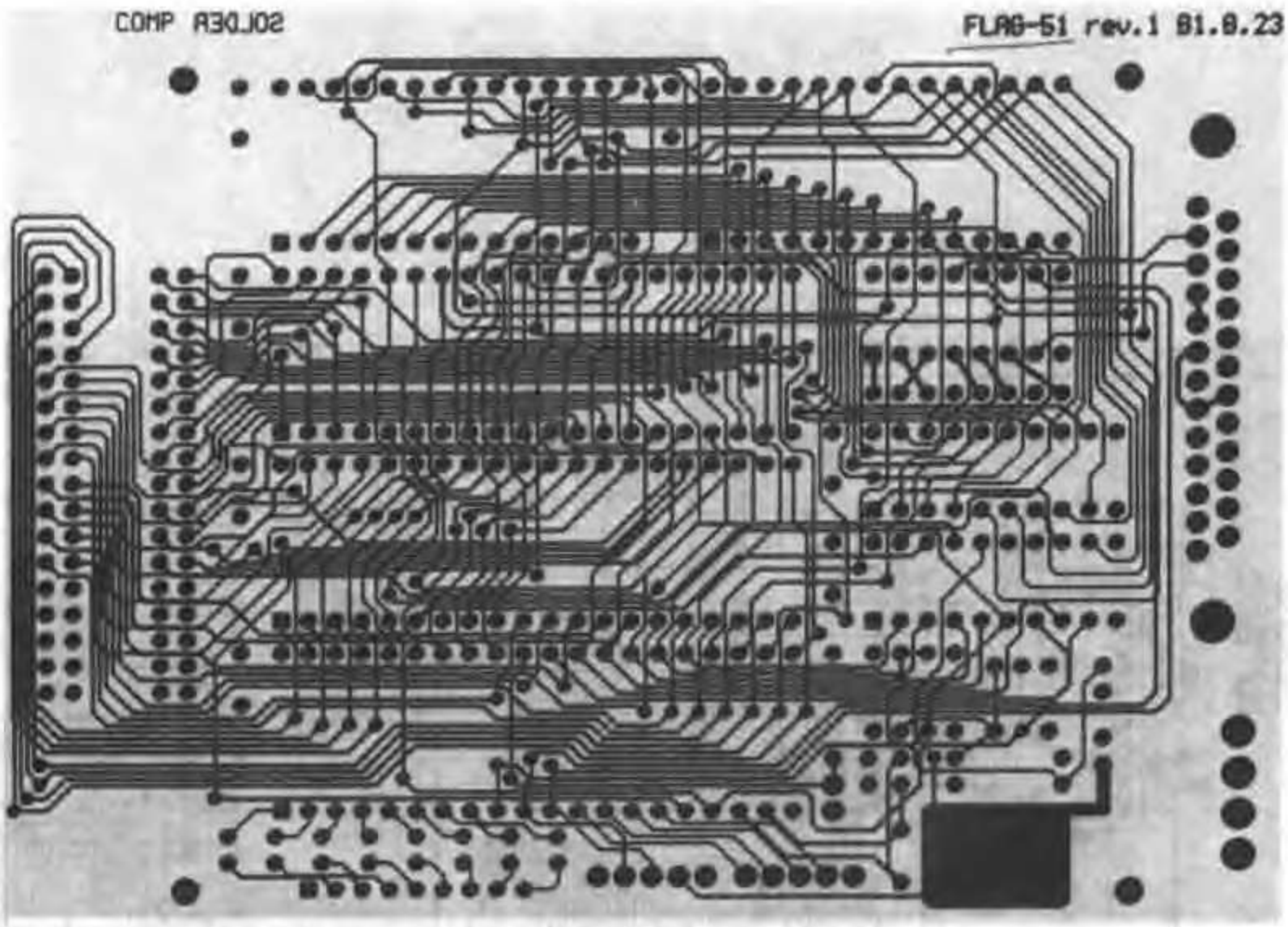

COMP A30J02
FLAG-51 rev.1 01.8.23

图 8-5

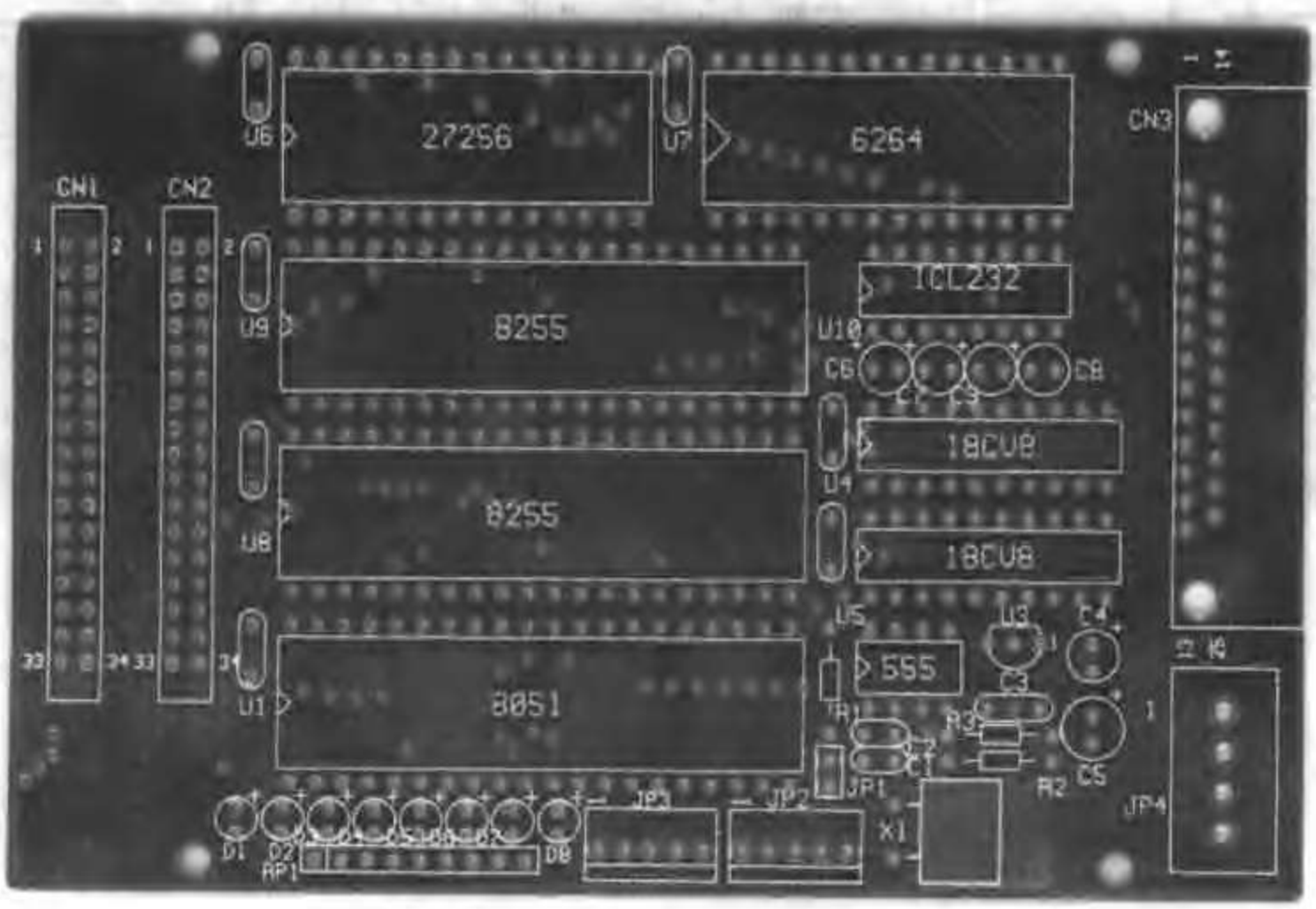

U6
27256
U7
6264
CN3
CN1
CN2
U9
8255
ICL232
U10
C6
C8
C7
C3
U4
MAX
U8
8255
U5
555
U3
C4
C3
R1
R3
C2
C1
C5
R2
JP1
JP2
JP3
JP4
X1
U1
8051
D1
D2
RP1
D8

图 8-6

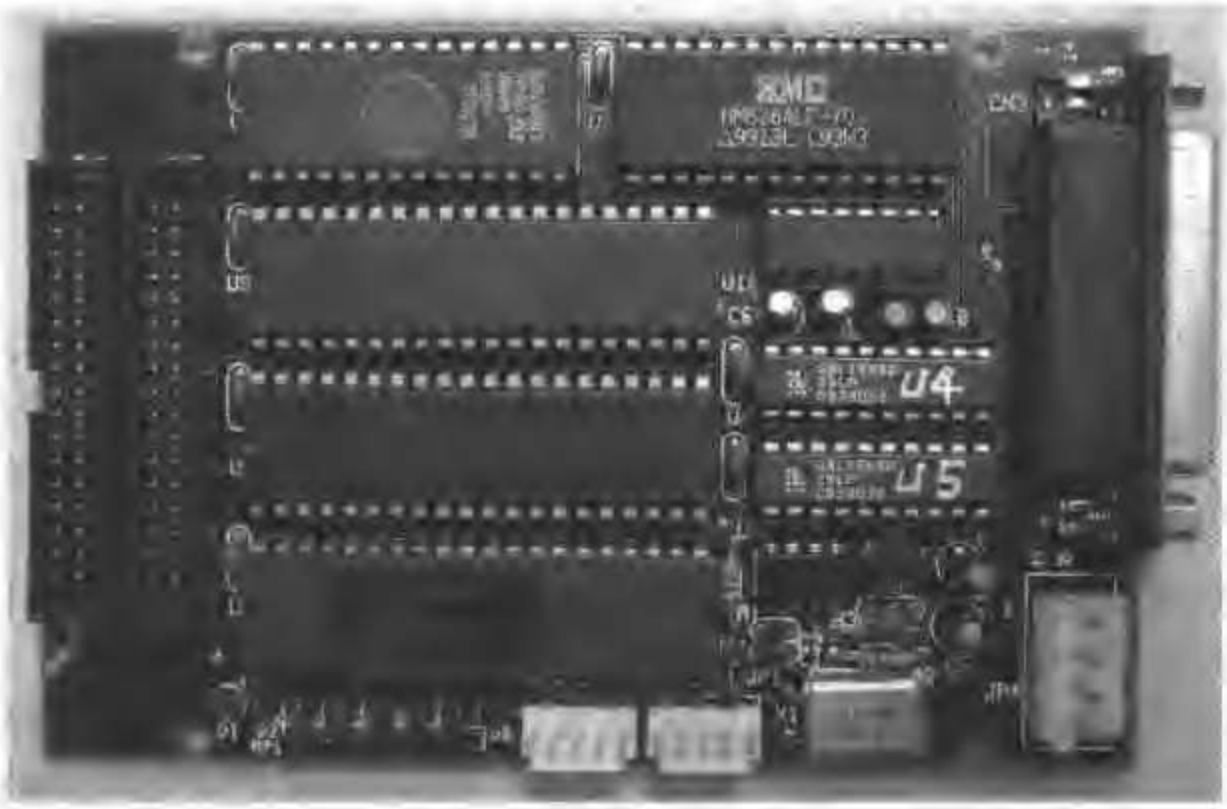


图 8-7

8-3　测试流程的安排

一个好的系统设计背后必定有一套完善的测试流程支持着。当 FLAG51 的 PC 板完成后交到手上时，我们首先完成的正是该 PC 板的测试程序 ROM，因为通过测试程序检验的控制板，才能保证所有的硬件线路是正确无误的，之后才是系统程序的开发与排错，只要在任何情况对系统的电路有怀疑时，可以立即换上测试程序 ROM，马上试验系统是否正常运作。更进一步的设计还可以隔离有错误的电路，并发出系统故障的信息。

国外有许多测试仪器能够做这方面非常完备的线路板测试，以后有机会的话我们会对这方面做更详尽的介绍与报导。若国内的专业工程师能对这方面多了解一点，相信对整个信息产业的发展也是相当有帮助的。

8-4　用C语言也可以测试

其实整个 FLAG51 的监控程序百分之九十五都是用 C 语言写成的，所以一发现程序有错误到修改完成，只须花不到二十分钟的时间，这是用汇编语言写程序所无法比拟的。但是要学会用 C 语言写程序之前，我们认为先学好汇编语言却是必要的，因为唯有完全懂得 8051 单片机，才能妥善应用 8051 的各项优点，进而避开其存在的缺点。为了减少开发时间并增加程序的可读性，我们希望这里所示范的单片机控制程序大部分是以 C 语言为主，对时间要求较严格的场合才以汇编语言来替代，我们打算采用下面的设计开发模式来示范一台以 8051 为 CPU 的计数器是如何被开发出来的。

◆ 步骤一：整个系统先用汇编语言完成各项硬件动作的验证，并且尽量采取模块化的设计。

◆ 步骤二：再用 C 语言改写整个系统，若步骤一的程序真的是模块化的规划时，转换的动作就会非常简单。

◆ 步骤三：整个系统以C语言为主，汇编语言为辅，后者占的份量可能不到整个程序的 10%，顺利完成本开发项目。

8-5 FLAG51 使用的电源

FLAG51 套件正式在此与读者见面，只要额外准备一条 RS232 的连接线和一台+5V 500mA 的电源供应器即可开始学习 8051 的各项实验，前者可以在普通的计算机商店或电子材料店内购得，后者可在电子材料店内买的到，价格视其输出的电压电流值而定。FLAG51 本体不加任何扩充板时只耗 200mA 左右的电流，换算的功率消耗只有 1W，所以，只要能提供超过 5V 200mA 以上的电源供应器就能使 FLAG51 控制卡动作，若考虑到未来的扩充性时，5V 电压端应能提供至少 3A 以上的电流，而且最好还能有±12V 的电压输出，以方便做模拟数字转换的实验。若纯以实验观点而言，PC 个人计算机用的电源供应器（100W 或 150W）是最方便的选择，但是体积过大为其缺点，我们正尝试与的电源供应器制造商联系，看看是否有合适的机种（供笔记本型计算机用的）能供应给“旗威”的读者使用，其 AC 输入电压由 90V 到 250V 皆可，输出端则有+5V（3A），+12V（0.5A）和-12V（0.5A）等输出，体积则是越小越好。

使用硬件需注意的事项：

◆ 电路电源千万不可接反。

◆ 电路电源接着的时候不可以拔除或插上 IC。

◆ 使用电烙铁时避免让 IC 过热，没把握的话可以使用 IC 脚座焊接，再插上 IC。

8-6 FLAG51 控制卡故障排除案例

在我收到的一个小邮包内有一块 FLAG51 控制板，写着无法与 PC 联机，请检查是否有问题？由于 FLAG51 控制板出货时是百分之百测试的，竟然会有故障品出现，难道这整个制造过程有问题？心里已对这个环节开始一连串的思考。等到有空时，将故障的控制卡换上测试 ROM 并接上电源后，发现果然连最基本的 LED 闪烁都没有，这种情况有两个问题存在。

第一是 8051 CPU 不振荡或是内部已损坏，喜好用旧零件的小厂商时常遇到这类毛病，只要换上新的 8051 就行了。可是 FLAG51 控制卡内都是新品，应不至于有这种情形发生。

第二种情况是译码部分电路故障，导致 CPU 无法正确取得程序 ROM 的数据，当然所有的动作都不对了，由于 FLAG51 控制板以两颗 PEEL 组件充当译码以及闩锁电路，有问题时只要重新刻录这两个 PEEL 即可。

首先取下故障板上的 CPU，换上新的 8051，结果仍是不动作，又找到两颗 PEEL 18CV8，内部已刻好译码以及闩锁电路的数据，换上后就能正常动作了。我们除了把故障品修好之外，还要探讨为何 U4 与 U5 的数据会不见了？于是把这两个 PEEL 移到刻录器（PEEL PROGRAMMER）上，先检查是否为空白的（Blank Test），可是一测试之下，屏幕上出现“该 IC 已损坏，原因是内部短路”等字样，而且两颗的故障原因都相同！PEEL 会故障通常是零件插反的缘故，若是这种情况在出厂前就应被拦下来了，不应该到了客户那边才出毛病，或是客户加电源时，输入的接头弄反了才使 PEEL 损坏，可是其他零件检查后都是正常的又做

何解释？

另一种可能是原先线路板是正确的，可是客户受强烈好奇心的驱使，把两个 PEEL 都取下，想要自行读取其内部的逻辑状态，可惜的是，操作刻录器时出错（执行的程序不对或 IC 放颠倒了），而且错了两次都没发觉，结果两个 PEEL 都阵亡了。

图 8-8 是 FLAG51 控制板上的可程序化组件 U4 与 U5。

图 8-8

其实，两个 PEEL 出厂前已在刻录器上执行 SECURITY ON 的设置，禁止任何装置读取其状态，所以此时再用刻录器做 READ 动作时，得到的内容将全是空白的，但此时的 PEEL 也不至于损坏。综合以上的分析，我们认为客户自己弄坏的机率较大，不过，希望仅此一次下不为例。U4 和 U5 的动作在书中已谈得很清楚了，请暂时不要更变其中的内容，另外操作 PEEL 刻录器时也请先看懂说明手册，否则这种事还是会一再发生的。

8-7　FLAG51 常见问题问答

问题 1：（简称 Q1）FLAG51 的 ROM 有多大？

回答（简称 A）： FLAG51 控制板上的程序 ROM 有 32KB 之多，32KB 的空间写一般程序应该足够，若在学习阶段，应该将内含监控程序的 ROM（27256）放入控制板的 U6 插座中，此时可以通过 RS-232 接口和个人计算机（PC 机）联机，在 PC 的键盘上就可以学习和控制 FLAG51 单片机控制板。当进入实际开发应用阶段时，这个 32KB 的 ROM 插座应该放上我们自己所写的应用控制程序，此时的控制信息依然可以传回给 PC，但是通信软件就得亲自来做了。

Q2：FLAG51 的断点有几个？

A：8051 单片机提供 5 个中断方式，其中包括两个外部硬件中断（INT0 和 INT1），两个内部计时数中断（TF0 和 TF1）以及通讯用的串行中断，在 FLAG51 控制板的监控程序中完全没有动用到这 5 个中断，所以使用者可以直接做各种中断的实验和研究。

不过，中断若处理不当时，CPU 死机的机会是很大的。当发觉 FLAG51 控制板因中断程

序导致完全不动作时，只要关闭单片机控制板的电源，然后重新开启电源，让监控程序再度掌握控制权，即可回到原先的控制模式。若想偷懒时，可以拿支“一字型”的螺丝刀直接在控制板555（U5）第二脚和第三脚上短路一次，也能使系统重新启动。

Q3：FLAG51上的缓存器是否可以修改？

A：当初规划FLAG51的监控程序时，设计思想是动作越单纯越好，所以在PC上并不能直接修改8051各个缓存器的命令，可是能通过程序的执行间接地修改缓存器的内容。其实8051内部缓存器的值都有一份备份值存放在RAM地址9F80h到9FFFh内，以ACC累加器为例，它占用单片机SFR地址的E0h位置，所以其备份值就在9FE0h上，而PSW值则存放在9FD0h地址，若直接更改这些RAM地址的内容，执行后的结果和缓存器的修改是相同的。当使用者的程序被执行时，监控程序做了相当的事前准备动作和事后储存动作，前者是将9F80h到9FFFh内有意义的数据值加载8051内部缓存器中，然后才开始执行用户程序，当程序执行完毕后，监控程序要立即将8051内部缓存器完完全全地转存到9F80h-9FFFh中，其中的过程是相当值得研究和探讨的。

Q4：FLAG51控制板是否可做单步执行？

A：所有包含微处理机的电子电路都是期望速度越快越好，可是在学习阶段我们却希望CPU能一次只做一个指令，好让我们能仔细地检查CPU内部各个缓存器的内容。只要外加一个可产生低频方波的硬件线路，接到8051的INT0硬件中断输入脚上，即可做单步操作的实验，程序中我们要指定INT0做负缘触发中断，中断的服务程序可以把现在的程序计数值(PC)及特殊的缓存器值显示出来，并且持续检查INT0（P3.2）的状态，若一直是真时就不跳回使用者程序，这就是所谓的单步执行操作。当输入方波频率越慢，单步执行的速度也跟着放慢，反之则越快。别的学习机只强调有单步操作的功能，可是却只字不提单步操作的原理，你认为那一种较适合学习呢？

Q5：FLAG51控制板的存储内容是否可移动？

A：FLAG51控制板的监控程序提供RAM区内存MOVE的指令，其格式为Mnnnn mmmm xxxx，它是将地址nnnnh-mmmmh块内的值搬到xxxxh开始的地址上，当然xxxxh开始的地址也必须是RAM区，否则搬移的动作是无效的。

Q6：如何进行反汇编，模拟监控？

A：FLAG51控制卡本身就是一个活生生的实验电路，所以不需要做任何软件上的仿真或监控。程序的结果可以过RS2-232接口传回PC端上，在监控程序中提供一个在线反汇编（On Line Disassembler）的功能，其格式为Zaddr，就是把addr地址上的机器码化成汇编语言的格式显示出来。

另一种反汇编的方法是使用书中光盘所附的程序（dis51.exe），可以反汇编多达32KB的程序数据，并且还能产生多个相关数据及程序呼叫文件，这对研究单片机程序很有帮助。另一方面，如何写这类工具软件也是相当值得学习的。原则上，包括FLAG51监控程序和其相关工具程序的源程序都将在本章最后头100%公开，请各位读者自行研究与参考。

8-8　本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler。

（2）IAR 8051 C Compiler。

8-9　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）逻辑笔 Logic Probe。

（3）数字式万用电表。

（4）模拟或数字示波器。

（5）直流电源供应器。

8-10　相关信息网站

你可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.goodwill.com.tw：查询逻辑笔与电源供应器信息。

http://www.fluke.com：查询数字电表信息。

http://www.agilent.com：查询示波器信息。

http://www.tek.com：查询示波器信息。

http://www.idrc.com.tw：查询直流电源供应器信息。

http://www.ictpld.com：查询 PEEL 组件。

http://www.latticesemi.com：查询 GAL 组件。

8-11　FLAG51 的监控程序分析

FLAG51 监控程序是我们在多年前所写的 8051 控制程序，可接受二进制文件的 8051 程序，下载后先存到 SRAM 中，等到使用者按了 G 键时，FLAG51 监控程序会先把 8051 的所有缓存器值复原，然后才开始执行使用者的程序，直到碰到最后一个 RET 指令为止，这时监控程序并不闲着，它要把所有的缓存器分别存入系统保护的 SRAM 区，最重要的是所有的缓存器都不能被更动。这些缓存器的数据也会分门别类地同时显示在屏幕上，供使用者检查及确认。

FLAG51 监控程序的另一个功能是在线的 8051 反汇编，通常我们下载程序到 FLAG51 内部后，会执行一小段反汇编程序，以确定程序是没问题的，但是即使你只执行一小段的反汇编的程序，但是监控程序仍要保证所有的指令与转移地址都是正确的。反汇编的程序都是

采用模块化的设计，有部分是由“旗威科技”所发展的 DIS51 程序移转过来的，光是这方面就有相当高的参考价值了。

FLAG51 监控程序内超过 90%是用 C 程序来处理的，只有程序执行前后的缓存器存入与下载是用汇编语言写成的。这个情况与 UNIX 系统非常类似，监控程序跟一般的应用程序还是有一些差距，例如堆栈的调整与缓存器的保护等等，如果你知道其中的奥妙，一定会对写 8051 程序有所帮助的。下页开始就是 FLAG51 的原始程序数据，如果你在研究时有任何疑问，还请上网询问。

Monitor.h 程序彻底公开

```
#include <ctype.h>
#include <io51.h>
#define BASE   0x9f00
#define START    0x8000

#define rate    0xfd
#define t_ol_delay 15
```

Monitor.xcl 程序彻底公开

```
-e8051
-D_R=0
-Z(CODE)CSTART,RCODE,CODE,CDATA,ZVECT,CONST,CSTR,CCSTR=0
-Z(IDATA)DATA,TEMP,IDATA,UDATA,ECSTR,WCSTR,ISTACK=8
log
monitor1
monitor2
monitor3
monitor4
cl8051s
-x
-l monitor.map
-o monitor.hex
-z
```

monitor.c 程序请查看书附光盘中的 CH8_MONITOR C 文件。

（注：本书中占用篇幅过长的程序，以记事本文档的形式记录在书附光盘中，请按程序名称在书附光盘中查找，下同。）

Monitor1.c 程序请查看书附光盘中的 CH8_MONITOR1 C 文件。

Monitor2.c 程序请查看书附光盘中的 CH8_MONITOR2 C 文件。

Monitor3.c 程序请查看书附光盘中的 CH8_MONITOR3 C 文件。

Monitor4.c 程序请查看书附光盘中的 CH8_MONITOR4 C 文件。

第 9 章

简易计数器的设计规划

简单的计数器，若加入通信联机的功能，那么应用范围就无形中扩大了许多，8051 单片机正好提供了相当优异的通信联机能力。本文将循序渐进地告诉您如何运用 8051 来进行计数器的开发。

9-1 计数器的基本功能

日常生活或工业控制中用到计数器的机会相当多，计数器的基本单位是次，所以显示的数字就代表次数，常见的计数器至少能提供 5~8 位整数值的显示，因此能显示的数字范围最大可到达千万之多，只要再经过适度的单位转换，就可变成其它的单位值。若输入的计数信号受到其它闸的控制，比方说接受计数信号并累积一秒钟，则此时得到的实际单位是 Hz（次/秒），汽车上的电子时速表也是依此原理推算出的，所以计数器可以说是大部分仪器设备的最基本的测量单位，所有的待测信号若能化成次数的单位，就能经由计数器读取其值，只要把这些计数值读入计算机内，就可以做进一步的数据处理。如果能再加入与其它设备联机通信的功能，那应用就更宽广了。接下来我们举几个例子说明计数器的应用时机。

9-2 定时器的应用实例

〔例 1〕某大百货公司开张后为了有效统计进入的人数，所以在大门的各个进入口上安装隐藏式光电感应器，每进入一个人时感应器就送出一个信号，这些信号接着到达电子计数器的输入点，这些计数器又与计算机实时联机，所以当天结束营业后的数分钟内，就可以得到各个进入口的人数统计，以及人数尖峰时间统计。得到这些重要资料数据后即可安排或变动出入口的摆设或调整公司内部的广告看板。

〔例 2〕交通单位为了监测某段高速公路的汽车时速，于是在该路段的某个地点埋入两个金属感应器，其间隔为 2.5 米，只要汽车一经过该感应点就有脉波信号送出，计数器此时计算出两个感测信号的时间间隔，再经过单位转换即可得到现在通过汽车的时速（公里/小时）。

假设所测得的时间间隔为 95ms，由于微处理机的计时单位可以小到 μs 等级，所以上述

的值是非常容易得到的，因此立即得到：

汽车的时速

=（2.5 m/0.095 s）×（1 km/1000 m）×（3600 s/1 h）

=9.000/0.095 公里/小时

=94.7 公里/小时

车速=9.000/（所测得的时间间隔 s）公里/小时

=9000/（所测得的时间间隔 ms）公里/小时

只要把常数 9000 除以所测得的时间间隔即可得到汽车的速度，若此部分用 8051 单片机处理时，整个计算时间花不到 1000μs。若时速值超过 100 公里/小时，就即刻启动照相机的快门，该车后车轮尚未离开传感器前，就已被照相存证了，除非汽车经过传感器的时间间隔小于 1ms，才会使计数器内的程序无法计算出其时速。您猜此时的时速大约是多少呢？答案是将近是每小时九千公里，所以千万别相信“车速超过 200 公里/小时就拍摄不到”的说法。若这些测速计又能够与控制中心联机，我们就可以推算出该路段的平均时速，若平均时速小于 40 公里时，就代表该路段开始塞车了，控制中心开始通过广播请驾驶员改道。所以说，测速器并不是只拍摄违规超速而已。

〔例 3〕某工厂为了随时掌握生产情形，特地在每台生产机器上安装计数器，并且这些计数器都与计算机联机，所以每台机器的加工状况都能在计算机上看到。而且这些计数值与订单生产值比对，随即可以算出正确交货期为何日，若能再与整个工厂的订单计算机联机，就可算出接受此订单是否划得来以及每台机器的生产量损益点。

由以上的例子得知，单纯的计数器只能做值的显示而已，若能够加入通信联机的功能，那么应用范围就无形中扩大了好几倍，而单片机 8051 也刚好提供了相当优异的通信联机能力。所以说，运用 8051 来进行计数器的开发是最合适不过的。

9-3 计数器设计前的功能规划

计数器最主要的是将计数值显示出来，显示的方式当然是我们最习惯的十进制数值指示，这方面可以借助简单的硬件线路完成。计数器显示的位数至少要有五到八位数，输入方面为了单纯起见暂定为标准的 TTL 输入端，其它的准位只要能化成 TTL 的信号准位，也就能做各方面计数值的显示了。综合以上的说法，我们把基本的计数器功能归纳如下：

（1）显示位数：6~8 位的七段 LED 显示器。

（2）输入准位：标准 TTL 接口准位。

（3）可写计数数据内存。

（4）控制器采用 FLAG51 单片机控制板。

（5）计数器可以做清除（CLEAR）和默认值设置（PRESET）功能。

（6）计数值可以上数（COUNT UP）或下数（COUNT DOWN）。

（7）当计数值到达设置值时，可以有信号或声响输出。

（8）可与其他计算机或仪器联机。

（9）内含预除器，可以做各种单位值的转换，此时显示的值可能就有小数点出现，所

以显示数字必须有小数点指示的功能。

9-4　预除器的加入

微处理机最擅长的是二进制值的运算和处理。所以，在单片机 8051 内部存放的计数值也是二进制的，只有在显示时将二进制的计数值化成十进制码，最后才把结果送到十进制显示器上。

如果这个讲法读者能接受的话，我们可以在转换程序当中插入一个单位转换程序，将显示值换算成更有意义的显示单位，这就是所谓的预除器（PRESCALER）功能了。例如工业控制上常把圆周角度的变化量以脉波信号送出，假设 360 个脉波信号等于一圈，若计数器收到 720 个脉波信号时，以 720 数字值表示较佳，还是 2.00 圈的表示值较佳呢？答案应是后者较为妥当。尤其是计数值往上累计时，用圈数来表示比脉波数更好。因此，我们将在计数器的程序开发后段加入预除器的运算功能。用汇编语言来处理这方面的数据是颇富挑战性的。

9-5　I/O 监视器的最初测试

为了使计数器的开发过程更为顺畅，我们重新规划了一个新的 I/O 监视线路板以方便程序的排错。请看图 9-1，基本上这是一个测试 8255 外围 IC 的线路，J1 通过三个 DIPSW 开关共 24bit 作信号输入之用，可测 8255 的输入（INPUT）特性，并且有提升电阻（10K 欧姆）将输入点提到高电位，所以 DIPSW 的开关在 OFF 位置时，读入的逻辑状态值是 1，反之则是状态 0。J2 上共接有 24 个 LED 发光二极管，所以 J2 是用来测试 8255 的输出（OUTPUT）特性，做完基本的 8255 I/O 实验后，还可以做有光隔离的输入实验。图 9-2 左下方的 J3 共享了 4 个光耦合器 K817（PHOTO ISOLATOR）做输入信号的隔离，用 K817 当成输入隔离可以有许多优点：

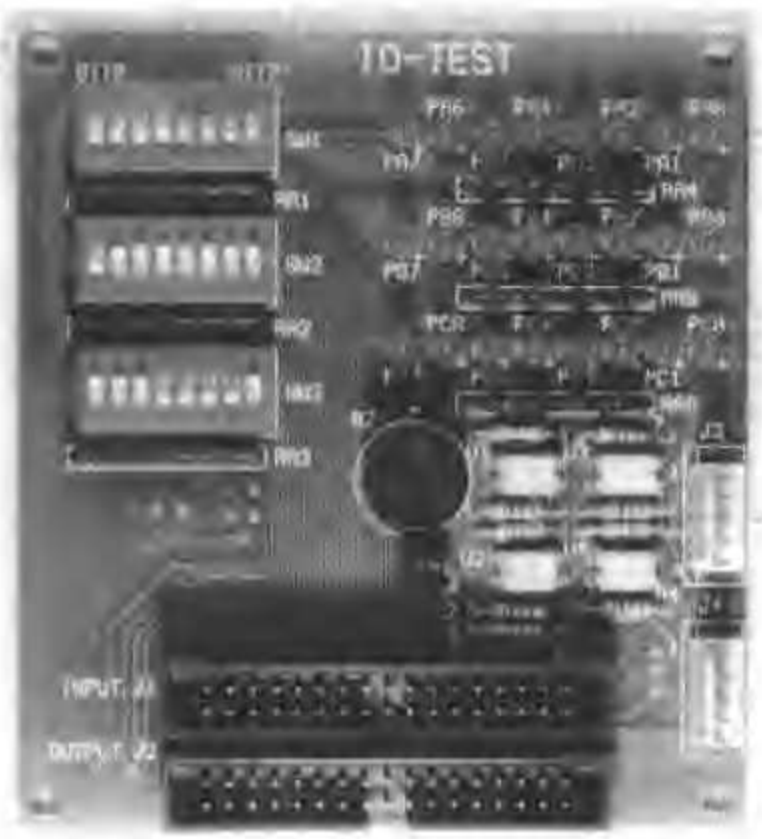


图 9-1

注：I/O 监视器完成品。

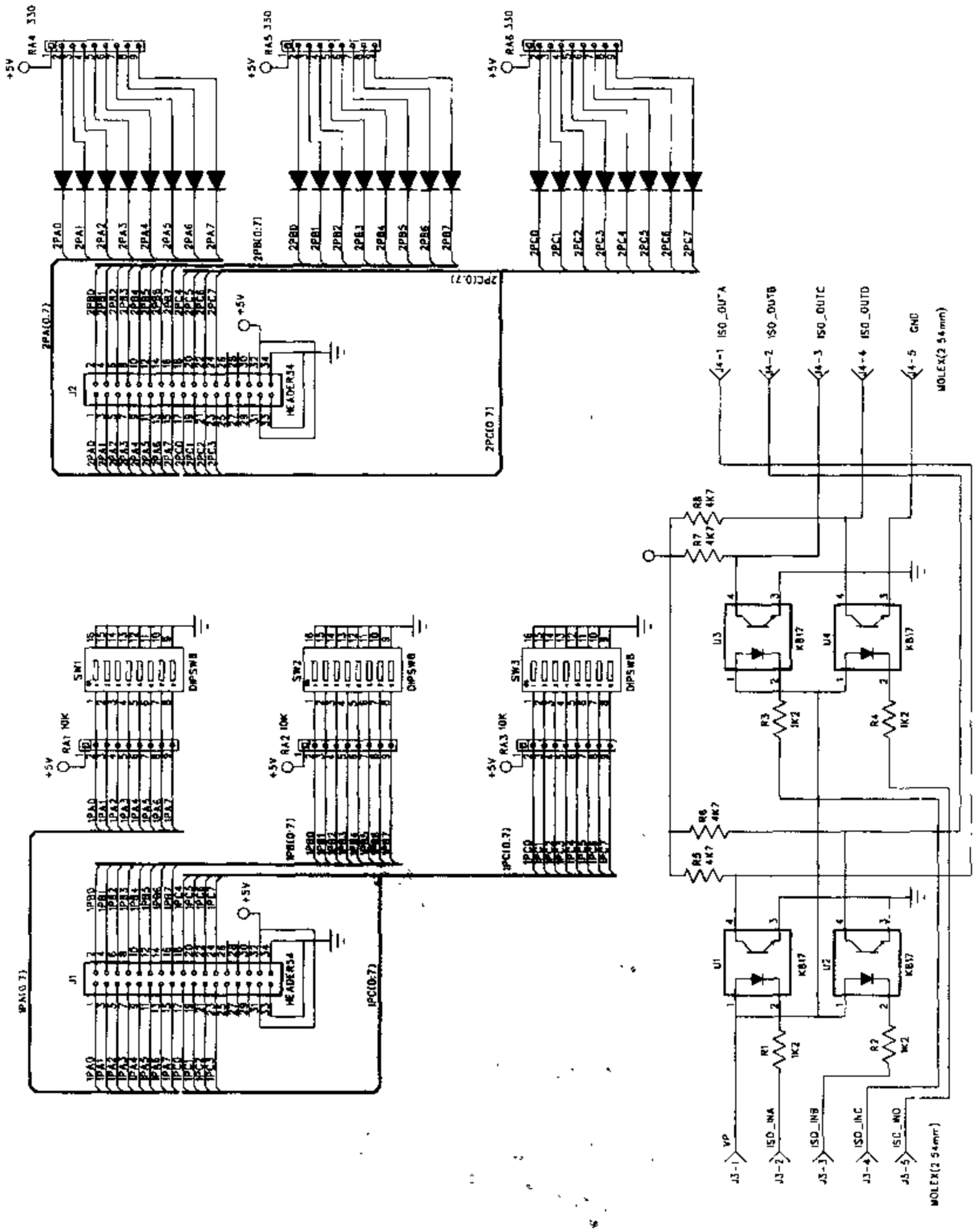

RA4 330
RA5 330
RA6 330
+5V
2PA0
2PA1
2PA2
2PA3
2PA4
2PA5
2PA6
2PA7
2PB0
2PB1
2PB2
2PB3
2PB4
2PB5
2PB6
2PB7
2PC0
2PC1
2PC2
2PC3
2PC4
2PC5
2PC6
2PC7
2PA(0.7)
2PB(0.7)
2PC(0.7)
J2
HEADER34
1PA0
1PA1
1PA2
1PA3
1PA4
1PA5
1PA6
1PA7
1PB0
1PB1
1PB2
1PB3
1PB4
1PB5
1PB6
1PB7
1PC0
1PC1
1PC2
1PC3
1PC4
1PC5
1PC6
1PC7
1PA(0.7)
1PB(0.7)
1PC(0.7)
J1
SW1
SW2
SW3
DIPSW8
RA7 10K
RA2 10K
RA3 10K
J4-1 ISO_OUTA
J4-2 ISO_OUTB
J4-3 ISO_OUTC
J4-4 ISO_OUTD
J4-5 GND
MOLEX(2.54mm)
R8 4K7
R7 4K7
R6 4K7
R5 4K7
U3
U4
U1
U2
K817
R3 1K2
R4 1K2
R1 1K2
R2 1K2
VP
J3-1
J3-2 ISO_INA
J3-3 ISO_INB
J3-4 ISO_INC
J3-5 ISO_IND
MOLEX(2.54mm)

图 9-2

（1）输入信号的准位可以超过+5V，在工业控制上常用直流电压（+12V 或+24V）传送状态或信号，此类的信号可接到本测试板上。

（2）另一好处是信号已完全隔离，所以没有接地的问题，更能有效地减少信号干扰的程度。经验告诉我们：如果把马达电路和微处理机线路摆在一起时，一定会死机不断，恶梦连连，主要原因正是接地及干扰的抑制处理未妥当，如果双方能利用 PHOTO ISOLATOR 做信号的隔离，马达启动时的电源变动以及火花干扰信号就不会通过地端传给微处理机了。

J3 的信号经过 K817 耦合后传到 J4 接头上，J4 通常再接到 FLAG51 单片机控制板的 J3（INT0，INT1，T0，T1 输入点），所以外部的信号可以当成 8051 的中断源或计时基准，经过光耦合的信号亦可通过板上的四个跳线（JUMPER）将信号传给 8255 的 PC7-PC4（4bit），我们可以用程序立即检查外部输入信号的状态。I/O 测试卡上还有一个小型蜂鸣器，只要将 JP6 短路，然后把担任输出功能的 PC7 位设成 1，蜂鸣器就会发出声响，直到 PC7 被清除成 0 为止。所以一片 I/O 测试板实际上包含了基本的输出入、光耦合输入及蜂鸣器声响输出四项功能。在计数器的显示功能尚未完全敲定前，所有的计数值仍能以 LED 做指示，对实验是有相当助益的。

光隔离输入正确的接法是 J3-1 点接到外部信号的电源端（+5V、+12V 或+24V），其余四点则是信号输入点。由于 K817 光耦合器内部的发光 LED 只要 10mA 左右的电流即可全亮，所以线路上的限流电阻（R1-R4）必须依外部电源的高低做适度的调整，其计算公式为 R=V（外部电源）/I（10mA），在图中我们选用的电阻值是 1.2K 欧姆，推算回去得到外部电源电压为 12V 左右。假如您对 1K2 的标示值感到疑问时，不用怀疑，这正是代表 1.2K 欧姆，因为小数点常在书或杂志印刷过程中消失不见了，所以特地把 1.2K 标示成 1K2，省略掉小数点，以便读者在自行购买或装配零件时较不会出错。

9-6　I/O 监视器的程序测试

I/O 测试板的程序并不会过于复杂，其动作顺序依序为：设置 8255 为输入或输出模式，然后读入对应的位（INPUT）或送出对应的位（OUTPUT），以下是几个典型的程序范例。唯一需留意的是 24 个 LED 的接法是安排成负逻辑方式，所以要让 LED 亮，必须把对应的位设成 0 而不是 1。

LED 输出示范

```
;实验前请先将 8255 输出接头与 I/O 测试卡 J2 间的数据线接妥
    MOV     DPTR,#A003H             ;8255 的控制地址
    MOV     A,#80H                  ;PA,PA 和 PC 都是输出
    MOVX    @DPTR,A                 ;送出控制码
;以上设置动作只须在开机时设置一次即可
    MOV     A,#00H
    MOV     DPTR,#A000H
    MOVX    @DPTR,A                 ;PA 端口的八个 LED 全亮
    MOV     A,#FFH
    MOV     DPTR,#A001H
```

```
    MOVX    @DPTR,A                 ;PB 端口的 8 个 LED 全灭
    MOV     A,#0FH
    MOV     DPTR,#A002H
    MOVX    @DPTR,A                 ;PC 端口的 PC7-PC4 共 4 个 LED 亮
                                    ;PC3-PC0 共 4 个 LED 灭
```

DIPSW 输入示范

```
;实验时请先将 8255 输入接头与 I/O 测试卡 J1 间的数据线接妥
    MOV     DPTR,#B003H             ;8255 的控制地址
    MOV     A,#9BH                  ;PA,PA 和 PC 都是输入
    MOVX    @DPTR,A                 ;送出控制码
;以上设置动作只须在开机时设置一次即可
    MOV     DPTR,#B000H
    MOVX    A,@DPTR                 ;读入 PA 端口的状态
    MOV     20H,A                   ;存入(20H)的 DATA MEMORY 中
    MOV     DPTR,#B001H
    MOVX    A,@DPTR                 ;读入 PB 端口的状态
    MOV     21H,A                   ;存入(21H)的 DATA MEMORY 中
    MOV     DPTR,#B002H
    MOVX    A,@DPTR                 ;读入 PA 端口的状态
    MOV     22H,A                   ;存入(22H)的 DATA MEMORY 中
```

DIPSW 输入状态直接送到 LED 的显示端口上

```
;实验时请先将两个 8255 的输出入接头与 I/O 测试卡 J1 与 J2 间的数据线接妥
    MOV     DPTR,#A003H             ;8255 的控制地址
    MOV     A,#80H                  ;PA,PA 和 PC 都是输出
    MOVX    @DPTR,A                 ;送出控制码
    MOV     DPTR,#B003H             ;8255 的控制地址
    MOV     A,#9BH                  ;PA,PA 和 PC 都是输入
    MOVX    @DPTR,A                 ;送出控制码
;以上设定动作只须在开机时设定一次即可
    MOV     DPTR,#B000H
    MOVX    A,@DPTR                 ;读入 DIPSW1 状态
    CPL     A
    MOV     DPTR,#A000H
    MOVX    @DPTR,A                 ;值送到 LED 端口上
    MOV     DPTR,#B001H
    MOVX    A,@DPTR                 ;读入 DIPSW2 状态
    CPL     A
    MOV     DPTR,#A001H
    MOVX    @DPTR,A                 ;值送到 LED 端口上
```

```
    MOV     DPTR,#B002H
    MOVX    A,@DPTR                    ;读入 DIPSW2 状态
    CPL     A
    ANL     A,#0FH                     ;MASK BIT4-7
    MOV     DPTR,#A002H
    MOVX    @DPTR,A                    ;值送到 LED 端口上
```

BUZZER 声响示范

```
;测试前请先将 I/O 测试卡上的 JP6 短路
;BUZZER 请选购加上+5V 电压后会自动发声的规格
BUZZER_ON
    MOV     DPTR,#A003H
    MOV     A,#00001111B               ;PC7=1 的控制码
    MOVX    @DPTR,A                    ;BUZZER 开始叫,请留意旁人的抗议
;
BUZZER_OFF
    MOV     DPTR,#A003H
    MOV     A,#00001110B               ;PC7=0 的控制码
    MOVX    @DPTR,A                    ;BUZZER 停止叫
;
```

光耦合输入示范

```
;开启电源前请先完成外部信号的配线
;I/O 测试板上的 JP1 到 JP4 都要事先短路
    MOV     DPTR,#B003H                ;8255 的控制地址
    MOV     A,#9BH                     ;PA,PA 和 PC 都是输入
    MOVX    @DPTR,A                    ;送出控制码
;以上设定动作只须在开机时设定一次即可
READ_K817
    MOV     DPTR,#B002H
    MOVX    A,@DPTR
    ANL     A,#F0H                     ;只取 BIT7-BIT4 共 4 个位
    SWAP    A                          ;BIT7-BIT4 与 BIT3-BIT0 对调
    MOV     20H,A                      ;状态值的储存
```

9-7　简易计数器的制作

有了 FLAG51 控制板及 I/O 测试板后，就可以凑成一台简易的计数器了。信号由光耦合器端输入，计数值暂时用 16 个 LED 做指示，若到达设置值时，则以蜂鸣器作警告声响，等到正式的数字显示器完成后，再修改显示部分相关的例程，即可完成一部具有特殊意义的简

易计数器了。接下来就是简易计数器的几项程序动作概略解说：

- 步骤 1：完成刚开机的起始动作。
- 步骤 2：清除 16-bit 的计数值。
- 步骤 3：决定往上数或往下数，以及设置值。
- 步骤 4：依据光耦合器的输入信号将计数值作加一或减一的调整。
- 步骤 5：将计数值送到 LED 端口上（以后要先调用十进制转换程序再送到七段数字显示器上）。
- 步骤 6：判断计数值是否已到达设置值，若已到达设置值时，即刻启动蜂鸣器。
- 步骤 7：判断是否有通信需求，并随即通过 RS-232 界面送出计数值。
- 步骤 8：判断是否有清除的信号进入，有的话清除计数器的内容回到步骤 4 继续计数。

以上的步骤说明看似简单，但是真正下手去写程序时也要花掉几天的时间，有兴趣的读者请立即动手试看，因为唯有经过正式的考验，才能得到非常宝贵的经验，并且这些实验线路只要再经过稍微的修正，就可以将这些硬件应用在现实的生活当中，而不是做完实验后就可丢弃的线路。

下一章我们将公布简易计数器的配线以及程序内容分析，当然不可避免地会有一些状况出现，我们的心态是：面对问题，承接问题和解决问题，没有任何问题才真的有问题。我们希望所有的读者抱持的心态也是如此。假使您对计数器这方面或 8051 单片机有任何意见或疑问时，欢迎到“旗威科技”的网站（http://www.chipware.com.tw），我们会很乐意地为您解开这方面问题的。

9-8 8051 汇编语言小锦囊

◆ 请善用$符号减少 LABEL 标志

写汇编语言除了思路要相当清醒外，最怕的就是转移地址的名称重复使用。假如有一个 1000 行的汇编程序，里面有一段例程其进入点的标志（LABEL）为 CHECK，则其余部分的程序就不可以再度采用此 LABEL，所以早期的写法就是改用类似但较长的标志，例如 CHECK_A，CHECK_B 或 CHECK_ERR 等等权宜性的写法，这种写法往往使排错时间更长，因为我们碰到程序有问题时，还要多一道手续去思考这个标志的真正用处。其实，2500AD SOFTWARE INC 在发展一系列汇编语言的汇编器时，就已考虑到这个问题，特地提供一个特殊的标志加“$”符号的功能，只要在标志前加上$记号，就代表此标志是供作为单一例程使用，其它例程仍能使用这个标志，彼此之间不会造成标志的混淆。总而言之，如果一个标志要整个程序内都能共享时，标志前就不要加上$符号，反之属于例程本身内部的转移标志时，则尽量在标志前加上$符号，以免系统有过多的 LABEL 名称而使得程序设计师的排错时间加长。以下是实际应用范例：

（范例一）DELAY 延迟例程有无$符号的写法比较

```
DELAY     MOV    R0,#00H
DELAY_A   MOV    R1,#00H          ;DELAY_A 只用在 DELAY 程序内
```

```
DELAY_B     DJNZ    R1,DELAY_B      ;DELAY_B 只用在 DELAY 程序内
            DJNZ    R0,DELAY_A
            RET
;$ 有的写法
DELAY       MOV     R0,#00H         ;其他程序都可调用 DELAY 标志
$1          MOV     R1,#00H         ;省略了两个 LABEL
$2          DJNZ    R1,$2
            DJNZ    R0,$1
            RET
```

（范例二）两个例程共享某些 LABEL 但汇编时不会有 ERROR 出现

```
ROUTINE1    MOV     R7,#08H
            MOV     A,R7
$WAIT_LOOP
            LCALL   WAIT
            DJNZ    R7,$WAIT_LOOP           ;跳至 LCALL WAIT
            RET
;另一个程序
ROUTINE2    MOV     R2,30H
            MOV     B,A
$WAIT_LOOP
            LCALL   SUBTRACT
            DJNZ    R2,$WAIT_LOOP           ;跳至 LCALL SUBTRACT
            RET
```

（范例三）符号$的用法比较

```
;符号$的错误用法
ROUTINE1    MOV     R7,#08H
            MOV     A,R7
$BCD        LCALL   CONV_BCD        ;跳到 LCALL CONV_BCD
            DJNZ    R7,$BCD
            RET
ROUTINE2    MOV     DPTR,#8000H     ;错误示范
            MOVX    A,@DPTR
            MOV     R7,A
            SJMP    $BCD            ;不可跳出 ROUTINE2 程序外部
;正确的写法
ROUTINE1    MOV     R7,#08H         ;其他程序都可调用 ROUTINE1 标志
            MOV     A,R7
$BCD        LCALL   CONV_BCD        ;跳到 LCALL CONV_BCD
            DJNZ    R7,$BCD
```

```
            RET
ROUTINE2    MOV     DPTR,#8000H     ;其他程序都可调用 ROUTINE2 标志
            MOVX    A,@DPTR
            MOV     R7,A
$BCD        LCALL   CONV_BCD        ;重复一段
            DJNZ    R7,$BCD
            RET
;另一种正确的写法
ROUTINE1    MOV     R7,#08H         ;其他程序都可调用 ROUTINE1 标志
            MOV     A,R7
BCD_XXX                             ;其他程序都可调用 BCD_XXX 标志
$BCD        LCALL   CONV_BCD        ;跳到 LCALL CONV_BCD,$BCD 只在 ROUTINE1
            DJNZ    R7,$BCD         ;程序内使用
            RET
ROUTINE2    MOV     DPTR,#8000H     ;其他程序都可调用 ROUTINE2 标志
            MOVX    A,@DPTR
            MOV     R7,A
            SJMP    BCD_XXX
```

◆ HEX 码前是否要加 0?

如果再度回到 10 年前，写汇编语言若碰到十六进制码时，通常要在该值前加上 0，以免翻译程序指出有错误存在。比方说：数值 5 写成 05 或 05H 都不成问题，但是数值 160 若写成 A0H 却行不通，这是因为当时的汇编语言认为 A0H 是一个 LABEL，而不是一个数值，由于汇编程序遍寻不着 A0H 标志的进入点，所以才有错误出现。但是现今的汇编程序都变聪明了，它会判断疑似 LABEL 的最后一个字符是否为 H，若是的话再对该 LABEL 做数值转换，若转换成功，则该 LABEL 将被认定成数值而非标志。所以以前 0A0H 的写法就变成 A0H，80H 仍是 80H。假如各位读者看到别人的程序有上述的写法时，也许该人的软件程序功力应当在五年以上，千万不要小看了人家！

◆ LINK 时的 HEX 文件和 TSK 文件有何用途

当汇编语言汇编成功后，接下来就是链接程序的处理了，其中最常用的最终输出文件分别是 HEX 文件和 TSK 文件。HEX 文件可载到（DOWNLOAD）大部分的 ICE（实体仿真器）上，用来验证程序是否正确，HEX 文件本身是以 ASCII 码储存，所以占用较大的文件空间，而 TSK 文件则完全以二进制代码的方式储存程序内容，通常我们将此文件载到刻录器（EPROM PROGRAMMER）内，以便正式刻录最终的程序 ROM。TSK 文件和 PC 上的 EXE 文件类似，它是不可以用 DOS 的 TYPE 指令去看文件内部的内容。

9-9 本章使用软件

本章使用的软件为：2500AD 8051 Assembler。

9-10　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）IO 监视器

（3）光电开关/近接开关。

（4）K817/TL521 光耦合器。

（5）直流电源供应器。

（6）EPROM 刻录器。

9-11　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.toshiba.com：查询光耦合器信息。

http://www.sharp.com：查询光耦合器信息。

http://www.omron.co.jp：查询计数器与光电开关信息。

http://www.keyence.com.tw：查询光电开关信息。

第 10 章

8051 单片机的专长：计数及计时

> 您有没有这样的经验：站在某个位置一段时间后，就有服务出现？要做到这样的服务就必须用程序计算物体接近的时间。

在本章我们将实地演练 FLAG51 控制板与 I/O 测试板的控制程序写法，FLAG51 控制板对外部 I/O 的监控都是通过 8255 来完成的。由于控制板上有两颗 8255，所以我们常把其中之一专做输出，另一个则做输入之用。除此之外，我们亦可用程序重新设置这两颗 8255 的输出入功能，以下是 8255 的硬件分配地址，除非有特别声明，否则这里的程序例子都会用到这些地址声明。

```
PPIPA1      EQU     A000H   ;接到 CN1 的 8255，专供输出用
PPIPB1      EQU     A001H
PPIPC1      EQU     A002H
PPICTL1     EQU     A003H

PPIPA2      EQU     B000H   ;接到 CN2 的 8255，专供输入用
PPIPB2      EQU     B001H
PPIPC2      EQU     B002H
PPICTL2     EQU     B003H
```

如果依照上面的设置，我们必须将 FLAG51 控制板的 CN1 用 34P 的数据线连到 I/O 测试板的 J2 OUTPUT 端，CN2 则接到 J1 INPUT 端，I/O 测试板本身消耗的电流不会很大，所以与 FLAG51 控制板共享单一个电源是可以的。由于 I/O 监视器另外提供四组光隔离的输入点，所以我们打算利用这个特别的安排，由外界输入超过 5V 准位的信号，这次举的例子是一个光电近接开关，其实体图请参考图 10-1。假使您找不到类似的光电近接开关时，也可以用机械式的开关取代，两者唯一的差异是光电开关没有机械弹跳（bounce）的情形发生，而且几乎没有损耗，可靠度相对地提高了许多。不过光电近接开关都是用在工业控制上，价格都不会便宜。

图 10-1

10-1 DIP SW 状态的观察与光电开关的使用

I/O 监视器最初的用意是用来检查 8255 传送的数据是否正确，所以我们第一个测试程序，即将板上三组 DIP SW 开关状态读回，并且又立即送到板上的 24 个 LED 显示器上（请看图 10-2 的 I/O 监试板线路图）。当我们把 JP6 处短路时，若担任输出的 2PC7 处被设为 1 时，不仅会使对应的 LED 灭掉（负逻辑接法），同时还会让蜂鸣器产生叫声。以下是程序的内容，至于地址的定义声明请参考前一节的说明。

当程序 1 IOTEST1.ASM 执行时，只要一改变 DIP SW 上的状态，对应的 LED 灯也会做相对的亮或灭。我们使用的光电近接开关是属于 NO（Normal Open）型，即平常输出高电位，有物体接近本开关时，其输出才改为低电位状态，此时会导致图 10-3 中的 U1 内部发光二极管点亮，接着此信号藉由光的传输转给 U1 右边的光接收晶体管，造成该晶体管导通，于是 U1 的第 4 脚出现几乎为 0V 的低电位，所以只要 8255 一察觉到此点降为 0V 左右时，即可立刻认定有确定的物体靠近该近接开关了。在工业控制领域中经常利用类似的信号做各种加工动作的依据。

图 10-3 是 I/O 监视器实物图。

〔程序 1〕IOTEST1.ASM

```
;PROGRAM NAME IOTEST1.ASM
;若使用 MAG51 控制板载入本程序时，请取消 MOV SP,#60H 这行，让监督程序自行调整堆栈
START
        MOV     SP,#60H         ;堆栈设置
        LCALL   DELAY           ;系统刚起始，等待一小段时间后才开始动作
        LCALL   INIT_8255       ;8255 输出入模式设置
$1      LCALL   IO_TEST         ;读入三组 DIP SW 数据，并送到 LED 处
        SJMP    $1
;
IO_TEST
```

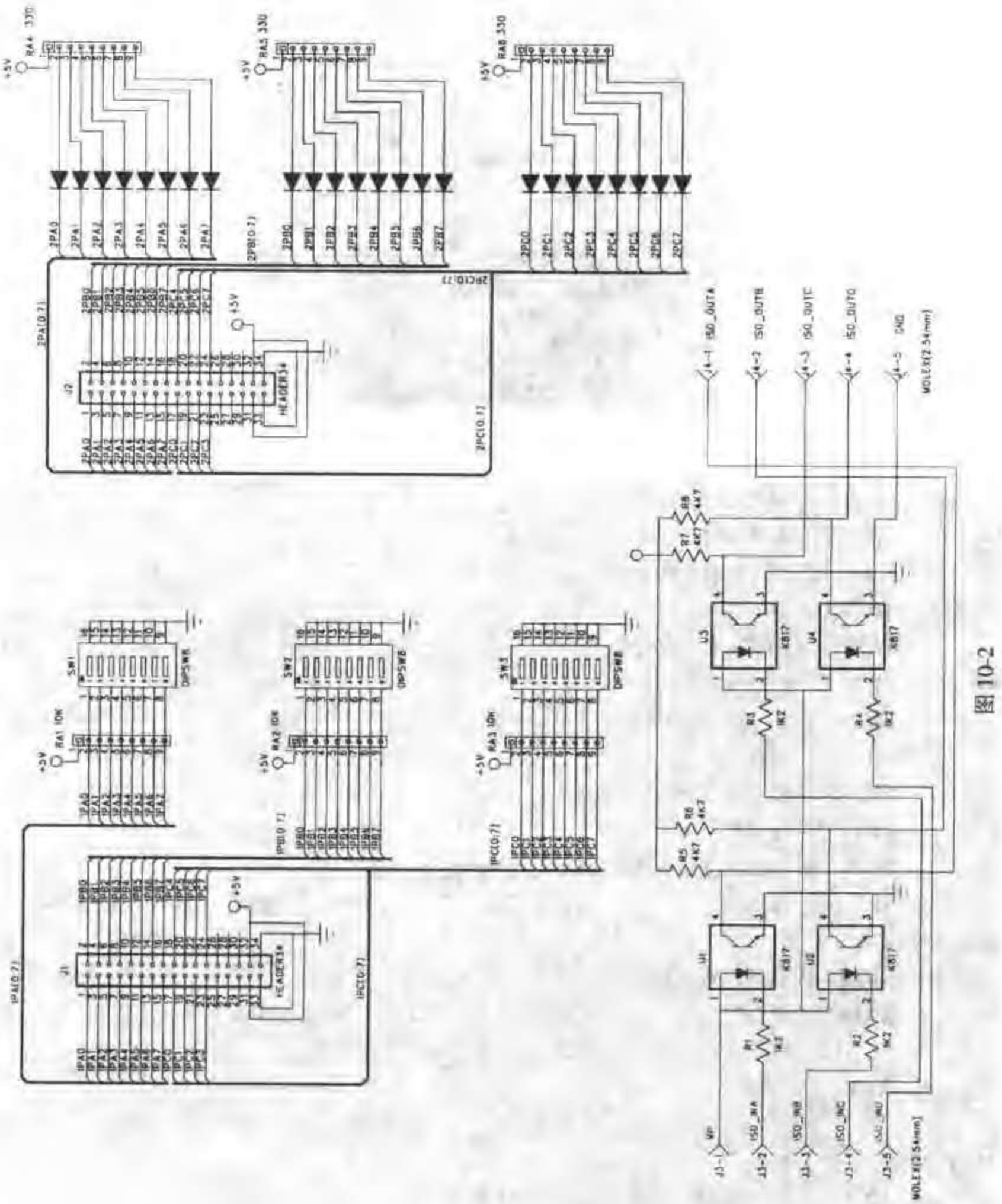


图 10-2

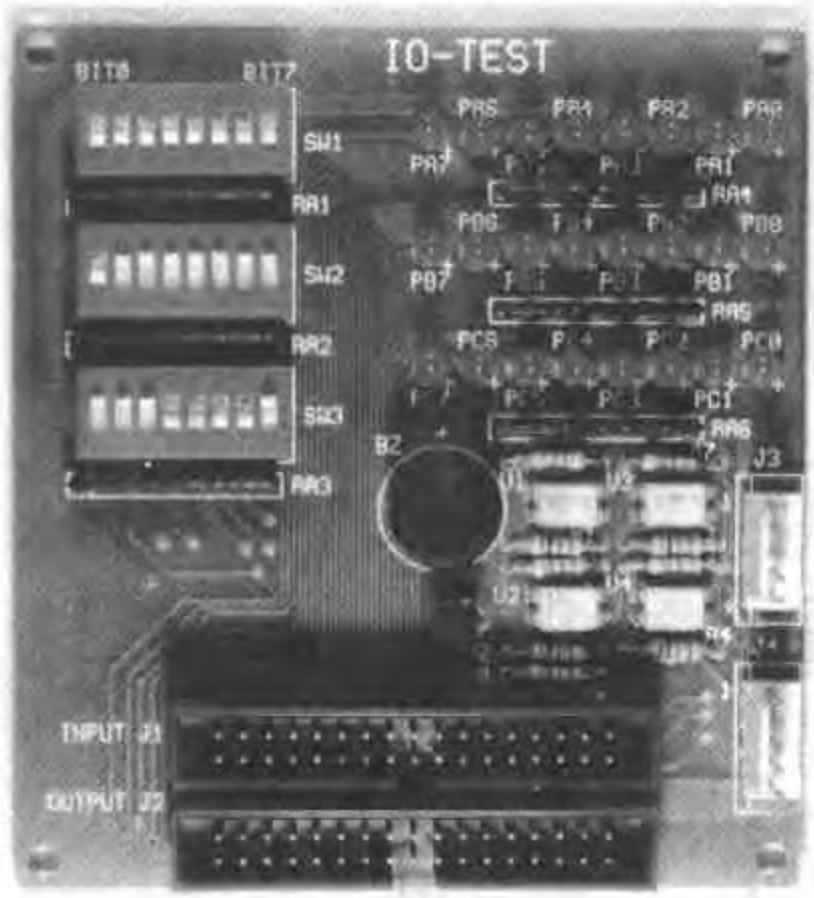


图 10-3

```
        MOV     DPTR,#PPIPA2
        MOVX    A,@DPTR
        MOV     DPTR,#PPIPA1
        MOVX    @DPTR,A
        MOV     DPTR,#PPIPB2
        MOVX    A,@DPTR
        MOV     DPTR,#PPIPB1
        MOVX    @DPTR,A
        MOV     DPTR,#PPIPC2
        MOVX    A,@DPTR
        JB      ACC.4,NO_SENSOR ;BIT4 接光隔离输入点
        SETB    ACC.7           ;若输入的 BIT4=1 则把 BIT7 设成 1，即 BUZZER 动作
NO_SENSOR
        MOV     DPTR,#PPIPC1
        MOVX    @DPTR,A         ;BIT7=1 时，I/O 测试板上的 BUZZER 会叫
        LCALL   DELAY           ;延迟一段时间后才回主循环
        RET
;
INIT_8255
        MOV     DPTR,#PPICTL1   ;接 CN1 的 8255
        MOV     A,#80H          ;PA,PB,PC ALL OUTPUT
        MOVX    @DPTR,A
        MOV     DPTR,#PPICTL2   ;接 CN2 的 8255
        MOV     A,#9BH          ;PA,PB,PC ALL INPUT
```

```
        MOVX    @DPTR,A
        RET
;
DELAY   MOV     R6,#00H         ;时间延迟
DLY1    MOV     R7,#00H
        DJNZ    R7,$
        DJNZ    R6,DLY1
        RET
```

10-2　计算物体接近的时间——基本写法

有一种日常应用不知道读者有没有印象，站在某个位置超过一段时间后，就会有服务出现？在高速公路休息站及各大餐厅的洗手间里尽是这类自动冲水的应用，只要我们靠近时间超过 3 秒就冲第一道水，当我们远离该设备时才又冲第二次水，这些应用中就有计算物体接近的持续时间。接下来的程序计算物体靠近的总时间值，并且将时间值通过 16 个 LED 表示出来。程序 2 修改自上一程序，故地址声明与共享例程部分就不再列出。

程序 2 执行时，只要一有物体接近光电开关，蜂鸣器就会开始鸣叫，并且由 I/O 测试板的上两排 LED 显示物体接近的累积时间。在程序说明中我们曾提到，只要将 DELAY 例程的时间设成一定数，即可准确地求出累积时间来。如果 8051 单片机只做单一项工作时，是可以采用上述的写法，亦即重复不间断地去监视外界输进来的信号。不过，若想让 8051 同时做多件事情时，每件事情处理的时间长短不一，若仍想保持上面的写法是很难的，碰到这类的问题时，就得运用另一种定时中断的程序写法才行得通了。

〔程序 2〕IOTEST2.ASM

```
;PROGRAM NAME IOTEST2.ASM
;若使用 FLAG51 控制板载入本程序时,请取消 MOV SP,#60H 这行,让监督程序自行调整堆栈
;
BUFFER      EQU     8200H           ;计时值储存区,共占两个字节
;
START       MOV         SP,#60H
            LL          DELAY
            CALL        INIT_8255
            LCALL       COUNT_CLEAR
$1          LCALL       IO_TEST
            SJMP        $1
;
IO_TEST     MOV         DPTR,#PPIPC2
            MOVX        A,@DPTR
            JB          ACC.4,NO_SENSOR
            SETB        ACC.7
```

```
            MOV     DPTR,#PPIPC1
            MOVX    @DPTR,A
            LCALL   COUNT_INC  ;计时值加一
            SJMP    NEXT
NO_SENSOR   MOV     DPTR,#PPIPC1
            MOVX    @DPTR,A
            LCALL   COUNT_CLEAR;计时值清除为 0
NEXT        LCALL   DELAY           ;基本计时单位，可定成刚好 0.1 秒或其他值
            RET
;
COUNT_CLEAR                         ;计时值清除成零
            MOV     DPTR,#BUFFER
            MOV     A,#00H
            MOVX    @DPTR,A
            INC     DPTR
            MOVX    @DPTR,A
            LCALL   COUNT_DISPLAY ;计时值送到 16 个 LED 灯上
            RET
COUNT_INC                           ;计时值内容加一
            MOV     DPTR,#BUFFER
            MOVX    A,@DPTR
            INC     A
            MOVX    @DPTR,A
            JNZ     $1
            INC     DPTR            ;进位调整
            MOVX    A,@DPTR
            INC     A
            MOVX    @DPTR,A
$1          LCALL   COUNT_DISPLAY ;计时值送到 16 个 LED 灯上
            RET
COUNT_DISPLAY                       ;16BIT 显示计时值
            MOV     DPTR,#BUFFER
            MOVX    A,@DPTR
            MOV     DPTR,#PPIPA1
            CPL     A               ;反相后输出
            MOVX    @DPTR,A
            MOV     DPTR,#BUFFER+1
            MOVX    A,@DPTR
            MOV     DPTR,#PPIPB1
            CPL     A               ;反相后输出
            MOVX    @DPTR,A
```

RET

10-3　计算物体接近的时间——定时中断写法

中断在单片机控制器上占有相当重要的地位，不熟悉它往不能更有效地控制程序的流程，对这方面仍有畏惧感的读者，请自行参考《8051 单片机彻底研究基础篇》一书中“中断彻底研究”的详尽说明，这里所举的程序例子应用 8051 内部的定时器做 1ms 的定时中断，然后在中断服务程序（ISR）内作近接开关的状态监视与计时，并且把有效的计时值显示在 I/O 控制板的 16 个 LED 灯上。本程序的写法与标准工业控制器内部的做法类似，许多标准化的产品也将循此原则陆陆续续地开发出来。

本程序的主要精神所在是主循环做系统较次要的动作，而重要的检查动作则每间隔 1ms 做一次，系统重要的研究与分析都在 ISR 中断服务例程内完成，所以不论程序执行到何时，系统还是能够“面面俱到”，随时能掌握系统中各项重要且不得忽略的信息。

由于 8051 单片机的 ISR 程序只有两种优先权限，且不能作 RE-ENTRY，所以 ISR 例程一定要保持简捷，做完该做的事后就立即返回原断点，否则将占用过多的时间，造成系统执行效率的降低，尤其是大型的控制程序更要注意此点。换句话说，中断程序是可以使用的，唯一的限制是使用次数不可过于频繁且不得耗掉过久的时间，否则会有反面效应出现。

〔程序 3〕IOTEST3.ASM

```
;PROGRAM NAME IOTEST3.ASM
;本程序请最好刻录成 ROM 后再做测试
;
BUFFER      EQU     8200H
T0COUNT     EQU     922             ;1mS IN XTAL=11.0592MHz(PAGE 345)
;
            ORG     0000H
            LJMP    START           ;主程序进入点
;
            ORG     000BH
            LJMP    T0ISR           ;TIMER0 溢位中断进入点
;
START       MOV     SP,#60H
            LCALL   DELAY
;
            LCALL   INIT_8255       ;8255 输出入启始设置
            LCALL   INIT_TIMER      ;计时器设置
            LCALL   ENABLE_INT      ;系统中断准许设置
            LCALL   COUNT_CLEAR     ;近接开关用计时器归零
;MAIN LOOP 主循环其实还可以做很多其他事情
$1          MOV     DPTR,#PPICTL1   ;主循环让在 PC0 的 LED 一下子亮一下子灭
```

```
        MOV     A,#00H              ;本指令送到 8255 时可使 PC0=0
        MOVX    @DPTR,A
        LCALL   DELAY
        LCALL   DELAY
        LCALL   DELAY
        MOV     DPTR,#PPICTL1
        MOV     A,#01H              ;本指令送到 8255 时可使 PC0=1
        MOVX    @DPTR,A
        LCALL   DELAY
        LCALL   DELAY
        LCALL   DELAY
        SJMP    $1                  等待中断
;
;ISR 中断服务程序由此开始,程序首先重新设置 TH0 与 TL0 计数值
TF0ISR
        MOV     TH0,#(65536-T0COUNT)/256      ;COUNT RELOAD
        MOV     TL0,#(65536-T0COUNT) .MOD. 256    ;COUNT RELOAD
        PUSH    PSW                 ;重要暂存器事先存入堆栈中
        PUSH    A
        PUSH    DPL
        PUSH    DPH
        LCALL   IO_TEST             ;调用近接开关检查程序
        POP     DPH                 ;取回原先存入的暂存器值
        POP     DPL
        POP     A
        POP     PSW
        RETI                        ;ISR 结束回原中断点
;
IO_TEST
        MOV     DPTR,#PPIPC2
        MOVX    A,@DPTR
        JB      ACC.4,NO_SENSOR
        MOV     A,#0FH              ;SET PC7=1
        MOV     DPTR,#PPICTL1
        MOVX    @DPTR,A
        LCALL   COUNT_INC
        SJMP    NEXT
NO_SENSOR
        MOV     A,#0EH              ;RESET PC7=0
        MOV     DPTR,#PPICTL1
        MOVX    @DPTR,A
```

```
        LCALL   COUNT_CLEAR
NEXT    RET
;
```

10-4　物体速度的测量

速度依照物理上的定义是经过的距离除以所需的时间，即 V（速度）=S（距离）/T（时间），如果 S（距离）是固定值，则只要测量待测物体经过的时间即可得到该物体的平均速度，汽车的测速器和职棒投手球速测速器都是运用上述的原理开发出来的。

要测量速度时，要有两个传感器分别摆在待测距离的起点与终点，当物体刚通过起点时，控制器内部的定时器开始启动，而当物体通过终点时立即停止定时器的计时动作。接下来是将距离值除以所得到的计时值即可得到一个正确的速度值，上一章内我们曾提到一个汽车测速的例子，假设两个传感器的距离为 2.5 米，则汽车的速度（千米/小时）公式经过化简最后成为 9000/测得的时间（ms），汽车的时速约是 94.7 公里，测试的时间愈准确则所得的速度也愈精确，假使读者您刚刚接到这个速度测量项目，必须用 FLAG51 控制板与 I/O 测试板完成全部的软硬件设计，您会如何动手规划呢？

10-5　本章使用软件

本章使用的软件为：2500AD 8051 Assembler。

10-6　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）I/O 监视器。

（3）光电开关/近接开关。

（4）K817/TL521 光耦合器。

（5）直流电源供应器。

10-7　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.toshiba.com：查询光耦合器信息。

http://www.sharp.com：查询光耦合器信息。

http://www.omron.co.jp：查询光电开关信息。

http://www.keyence.com.tw：查询光电开关信息。

第 11 章

FLAG51 单片机的问与答

FLAG51 单片机控制板是学习 8051 单片机的好帮手，自推出后即受广大用户关注，我们特地将读者的问题整理解答后，利用本章公开给所有的读者，希望能增进学习的效率。

11-1 问题与解答

从以前到现在，接了不少读者的信件，也客串维修了几块 FLAG51 单片机控制板，我们把这些疑问与心得做个整理，并且以一问一答的方式展现给所有有兴趣的读者，或许其中有部分正是各位心中的疑点，当您对 8051 单片机或硬软件有任何建议或问题时，请记得来函讨论与询问。

问题一：我使用 FLAG51 单片机控制板发生以下问题：

（1）自我测试失败。

（2）LED 灯恒亮。

（3）8051 RESET 脚无信号。

（4）无振荡频率输出。

解答：依照纸上描述看来，这块控制板应该是“阵亡”了，可是打开示波器并接上电源后，却发现动作完全正常，FLAG51 控制板竟然复活了，新鲜事可真多！想了又想还是百思不解，于是把写故障原因的纸条摆在工作桌上，过了两天后终于找到原因了：电源绝对有问题，因为他指出控制板上的 LED 恒亮，代表加入了电源，而且电源的接法正确，但是 RESET 无信号，且 8051 无振荡信号输出，会造成这种现象，除非是板上的 IC 供电不正常，或是低于额定的 4.5V 以下，所以只要把供应电压调高到 5V，即可使 FLAG51 正常动作。不过，有一点感到纳闷的是该读者一定是用示波器检查线路，才有以上的故障叙述，如果当时能顺便量一下电源的电压值，就可立即找出原因了，该单片机控制板也不会一再地碰到用户无法维修的问题。

问题二：我的 PC 上 COM1 接鼠标，COM2 连接 FLAG51 控制板，当键入 FLAG51 之后，屏幕上只见到光标一直往右移，而且屏幕持续上卷，这大概是什么问题？

解答：FLAG51.EXE 程序执行时会自动去在 COM1 或 COM2 检测 FLAG51 控制板，你提的现象是程序判断错误，以为 COM1 上就是 FLAG51 控制板，所以误把鼠标传回的数据送到屏幕上，造成屏幕的持续上卷，解决的方法是键入 FLAG51 2<ENTER>，指定 PC 由 COM2 与 FLAG51 沟通，若你的 COM 口是规划成 COM1 接 FLAG51 而 COM2 连鼠标时，则要键入 FLAG51 1<ENTER>。

问题三：将 FLAG51 控制板与 RS232 连接，执行 FLAG51.EXE 之后，只能显示书上所提示的信息，FLAG51 并没有回送状态，接着计算机死机，可是接其他 8051 学习机却正常？

解答：所有的 FLAG51 控制板在出厂前都经过一连串的目视检验与程序测试后才包装在静电袋（图 11-1）内，你的控制板经过再次的测试，发觉功能完全正常，所以请再次检查：

图 11-1

（1）电源是否正常。

（2）RS232 联机是否有接牢。

（3）试试强迫 PC 由 COM2 与 FLAG51 沟通（键入 FLAG51 2）。

（4）联机中加入 RS232 监视器（图 11-2），确定 FLAG51 是否有送出串行信息。

图 11-2

注：RS-232 监视器（左上方）可用来观察串行通信的状态，其余则为 RS232 的各种转换接头。

（5）用逻辑笔查看 FLAG51 是否仍有“心跳”。

由于 FLAG51 的价格并不贵，实在无法提供上门检修的服务，如果仍无法正常操作时，请将控制板退回“旗威”公司。

问题四：在《8051 单片机彻底研究基础篇》中，发现书中提到有关 8051 反汇编程序那部分并不很完整，很希望能索取这一部分的内容。

解答：在台湾地区可以买到的 8051 实体仿真器（ICE）都附有反汇编的功能，但是对想确实追踪程序的人却帮忙有限。书上提供的反汇编程序是针对用户的需求发展出来的，程序是以 TURBO C 写成的，从最基本的反汇编功能到最终的数据整理共花了三个月的时间，这方面有需要的读者，请自行至旗威科技的网站 http://www.chipware.com.tw 下载。

问题五：我的 FLAG51 控制板使用计算机专用的电源（Switching Mode Power Supply），并且已关闭计算机内部所有的常驻程序，但是程序运行之后就再也不肯动了，不知是操作步骤或硬件设定上有误，敬请不吝指正。

解答：如果 FLAG51 开机后有正确的联机信息传回 PC 时，代表 FLAG51 线路板的硬件动作是正常的。运行之后会死机有两种可能：一是你程序执行前的缓存器储存区内容不对，解决的方法是在程序加载 RAM 区前，先下一个 X 指令给 FLAG51，让 FLAG51 的监督程序调整所有缓存器的内容，之后再执行用户程序就不会死机了。另一种可能是你的程序动到了 8051 的 SP 堆栈值，请将这几行相关的指令取消掉即可，让系统自动检查并调整 SP 的值，也应能避开当机的机会。不过，当你移走 FLAG51 的监督程序自行发展系统程序时，可要自己留意 SP 值，稍有不慎绝对是死机连连的。

可调整电压的电源供应器要小心开机瞬间的脉冲电压，如图 11-3 所示。

图 11-3

问题六：学习单片机时，要看那些书籍以及如何思考？

解答：我们的回答是什么书都要看（从时报周刊、新闻、独家报导到各式专业书籍）才会有足够的思考空间。几年前有个从事控制的老师傅接了一个电磁阀自动冲水设计项目，老师傅于是交代下面的三个年轻设计工程师来处理，并且希望在一周内看到三件初步的规划案，

接下来当然是一连串的资料收集与讨论，一周后三份建议书顺利出炉了，最差的工程师打算用一部 PC 配合附加的 IO 控制卡完成本设计项目，如此设计的依据是 PC 相关的参考书籍最齐全，东抄一些西抄一些即可。另一位工程师则选用 Z80 担当控制中心，再配合其他控制线路完成本项目，设计的依据为节省控制板空间并有许多 Z80 子程序可供参考，所需的开发时间最短。最后一位工程师则选用单片机来做控制，硬件线路最简捷，但参考数据有限（几年前单片机的数据确实不多），老师傅看了这些建议数据后，只提了两个问题：价格如何压低与如果下个月开始交货可以吗？三位小师傅当然猛摇头说：不行。

其实类似的情形每天都不断地在发生，我们的思考路径千万不要被现有的知识给限制住了，当有人问你现在台北到高雄坐什么最快时，千万不要马上回答：搭飞机最快，因为对整个事件（系统）没有完整的理解前，遽然下任何决定都是很危险的。最后，老师傅交出的答案竟然是采用模拟电路，只用了一个 555 振荡器和两个 OP 运算放大器就完成了，价格压低到 100 元以内，三位工程师失望地想回去，将所有的书籍通通销毁，老师傅连忙制止，并且说道：如果本项目多增一项智能型功能时，你们的设计就可以派上用场了，千万不要妄自菲薄因而误了大事，下次接到任何问题时记得作全方位的思考。

学习单片机不可避免地要做一些硬件实验，简单的实验可以选择在面包板上接线（见图 11-4），利用单芯线将多个电子零件组装起来，实验成功后再把所有的零件拆除，回复面包板原来完全空白的面貌。不过，用面包板做实验最大的缺点是线路连接不够牢靠，若是不小心碰了一下，可能要找很久才会发现哪边的连接线松了，而且实验后的零件也不易保存。另一种实验的方法，是用空白线路板直接焊接，零件与零件之间使用镀银连接，而重要的 IC 零件，则使用 IC 座，以方便日后的更换或测试，不仅连接牢固并且保存方便。在相继推 FLAG-51 控制板与 I/O 测试板后，我们又再规划一片专供单片机实验的线路板，外观请看图 11-5。

这是一片双面都作实孔并喷锡的实验线路板，线路板上提供一个 34P 的数据线连接公座，可以利用数据线与 FLAG51 控制板相连，并且有 LED 灯指示电源是否正确，同时保留一组电源输入端子，以方便实验者外加其他的电源。

除此之外，尚保留一组+5V 的电源输出端，专供用户接逻辑笔用。本实验线路板上四个螺丝固定孔距离相同，所以可以直接用铜柱将两片线路板固定在一起。线路非常简单，不过自己焊接时，请特别注意数据线座与 LED 的方向，绝对不能出错，否则会无法接上 FLAG-51 控制板传来的信号。尔后简单的实验我们都尽量利用这片实验卡做示范。

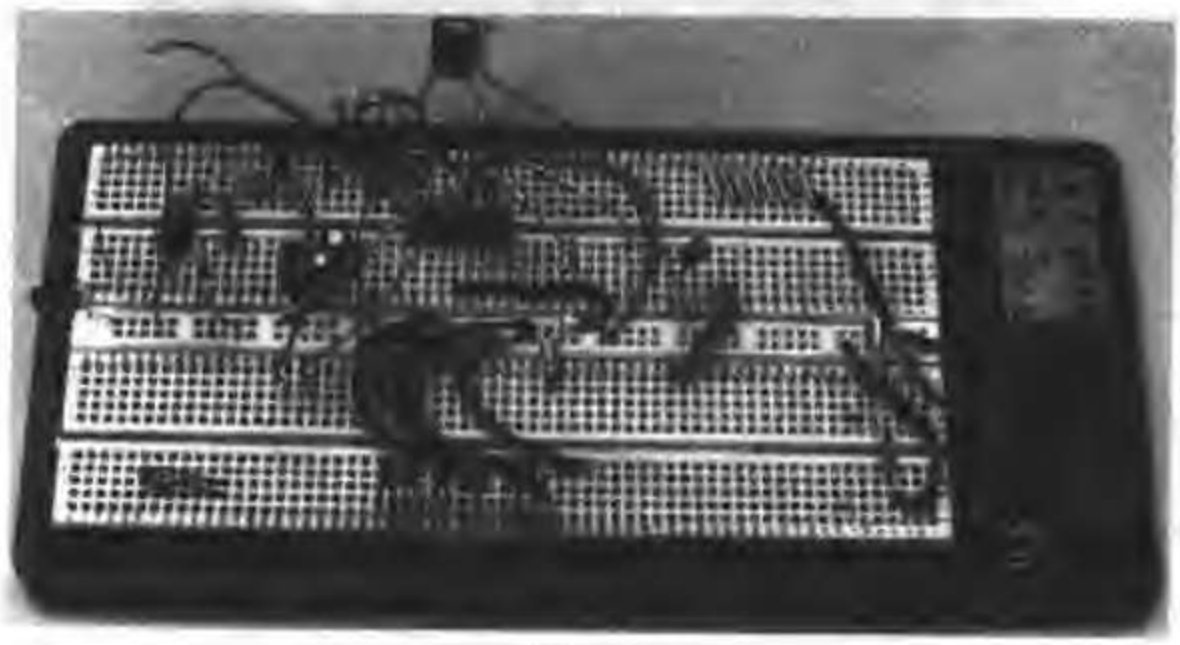


图 11-4

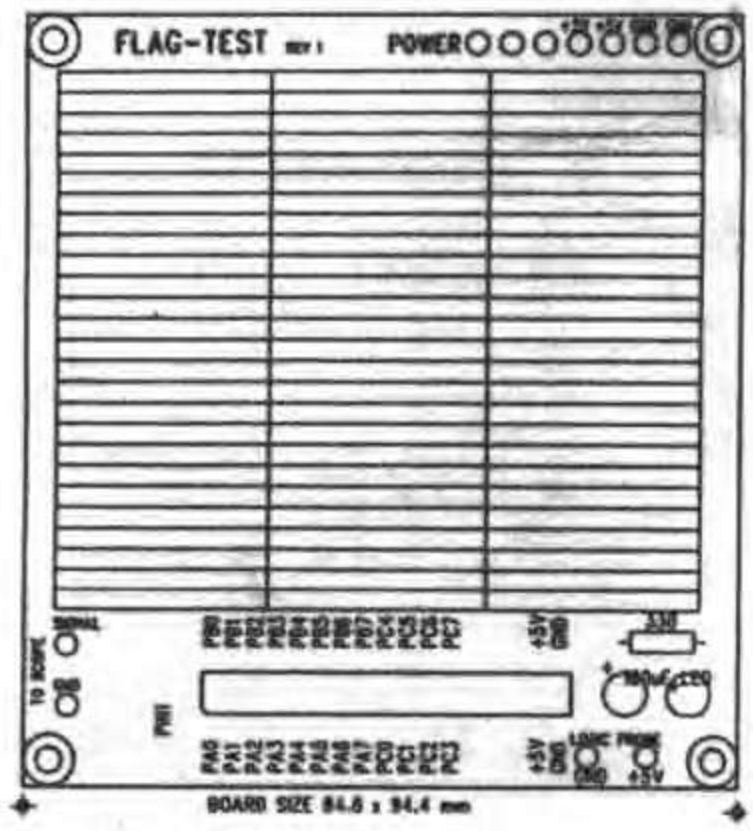


图 11-5

注：FLAG 万用实验板考虑到实验与测试两方面。

用空白万用板可做各式各样的硬件实验，如图 11-6 所示。

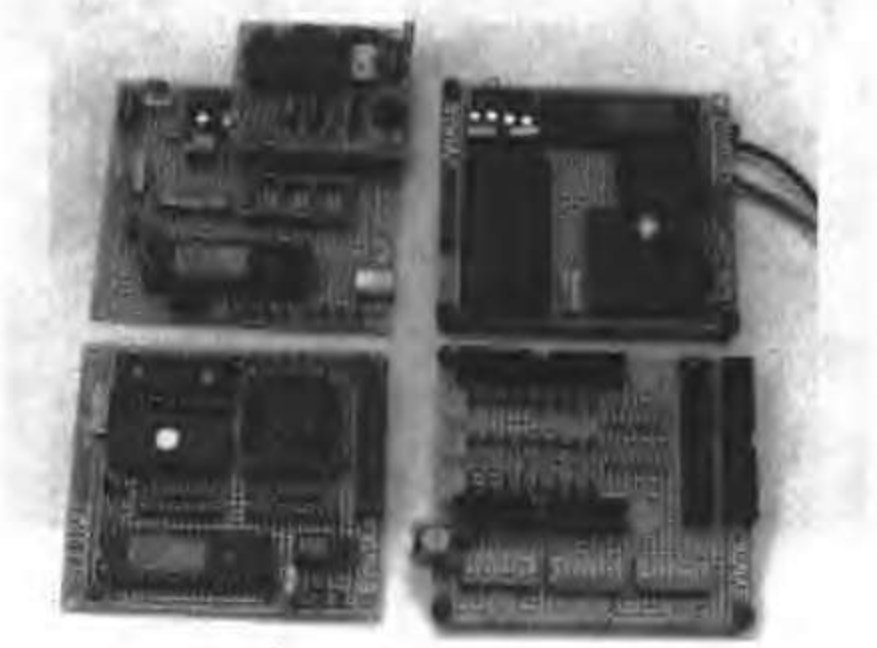


图 11-6

图 11-7 是 FLAG-51 单片机控制板。

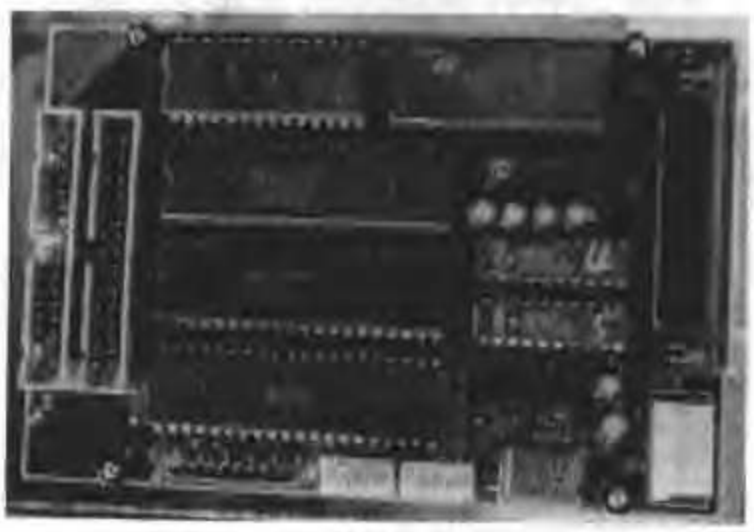


图 11-7

图 11-8 是 FLAG I/O 监视器。

图 11-8

图 11-9 是 FLAGDISP 七段数字显示器。

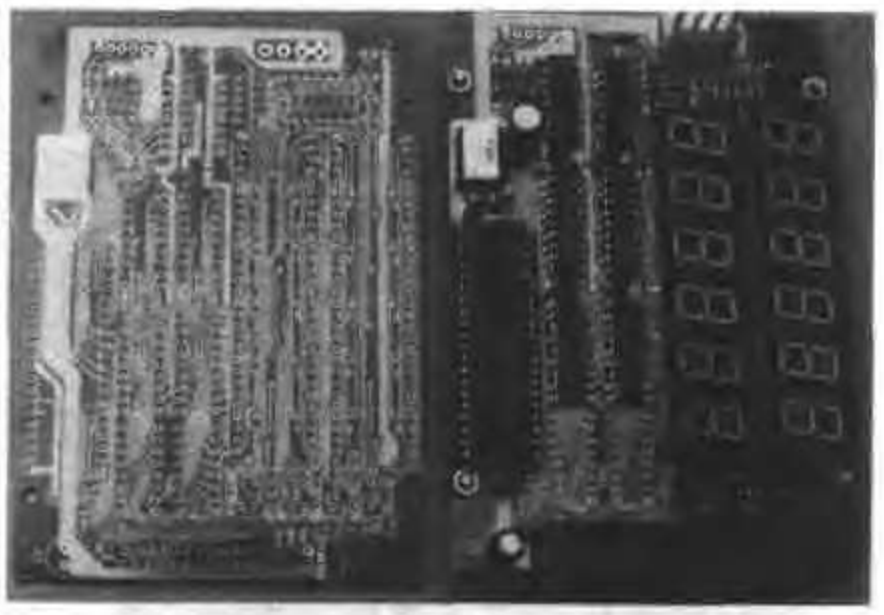


图 11-9

图 11-10 是稍后推出的 AT89C2051 学习板。

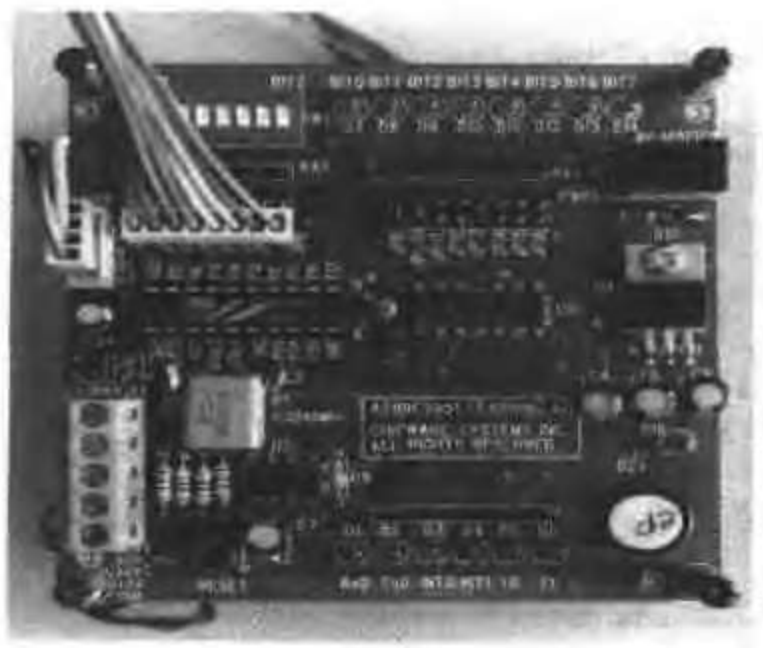


图 11-10

11-2　本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler。

（2）2500AD 8051 C Compiler。

11-3　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）I/O 监视器。

（3）实验用面包板与焊接用方用板。

（4）FLAG-DISP 七段显示器。

（5）直流电源供应器。

（6）RS232 缆线监视器。

11-4　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.angilent.com：查询高级电源供应器信息。

http://www.tek.com：查询示波器信息。

http://www.iar.com：查询 C Compiler 相关信息。

http://www.keil.com：查询 C Compiler 相关信息。

http://www.pads.com：查询 PC 板布线软件相关信息。

http://www.protel.com：查询 PC 板布线软件相关信息。

http://www.keyence.com.tw：查询光电开关信息。

第 12 章

I/O 输出/输入板的开发

一个完整的控制系统除了 FLAG51 控制板外，还应有隔离输入板与继电器输出板等等的配合，这样的组合就能涵盖一半以上的数字控制系统应用了。

到目前为止，我们所提到的线路与排错都是在 FLAG51 上执行，不论输入或输出都是数字 TTL 的准位。那就是说，所有输入的信号都必须转换成 TTL 的信号后，才能进入 FLAG51 控制板内。另一方面，FLAG51 的输出准位也只达到 5V 左右，所以，推动 LED 是可以的。但是若要推动继电器时，就有点吃力了！虽然也有 5V 规格的继电器，不过，比较正确的做法是另外加上缓冲器（Buffer）IC，进一步提升推动能力，这才能让继电器的动作更加确实。这章里我们就会讲到输出/入板的设计依据及做法。

12-1 隔离输入板的线路说明

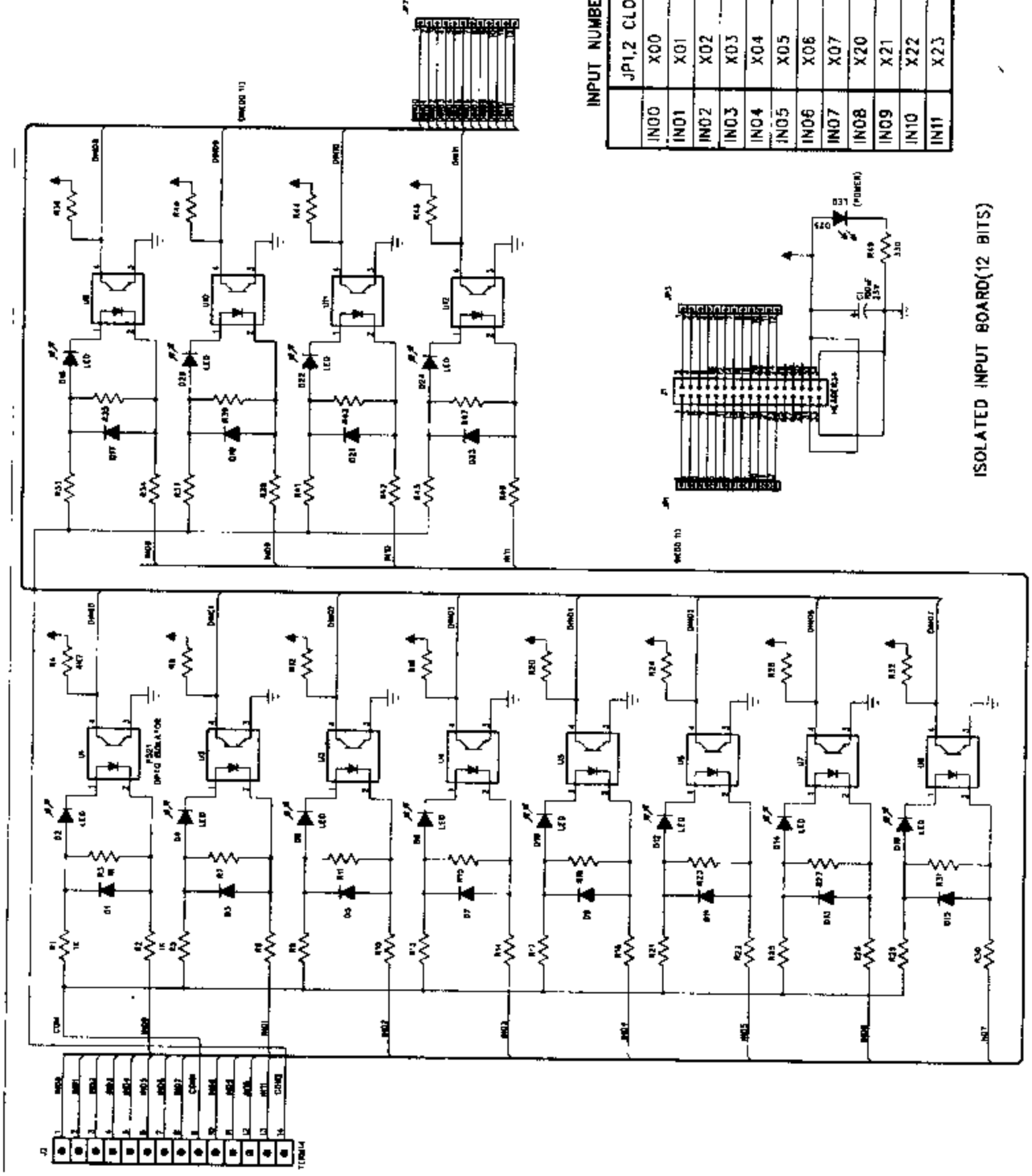
FLAG51 控制板上通过两个 8255 总共提供 48 个 I/O 输出入点，如果做一般的实验是不需做隔离的，可是若用在真实的控制领域时，最好的方法是在输入端加上光耦合隔离界面，除了可以隔开噪声与过大的信号外，还可以保护担任输出入沟通的 8255 与单片机控制板。

隔离输入板的线路请看图 12-1，每块隔离输入板提供 12 个输入点，每一个输入端都采用相同的线路结构；图 12-2 是单一输入端的线路分析，R1 与 R2 担任输入信号分压用，D1 则用来防止输入信号颠倒用，R1 的左端是输入信号的共同点，通常是接+12V 或+24V 直流电源端，平常无信号输入时，R1 左方的电压准位应该在+12V（+24V）上下，U1 右侧的光敏晶体管尚处于截止的状态，所以 DIN00 的输出状态被 R4 提升电阻拉到 5V 左右。

当输入信号降到接近 0V 时，R3 上下两点的电压差约是 4V 左右，这个电压配合足够的电流（12V/（1K+1K+1K）=4mA）就可以点亮 D2 的 LED 与 U1 内部的 LED，所以只要输入信号降到 0V（共同点必须接+12V）时，相对应的 LED 就会点亮。此时 U1 上的光敏晶体管也会导通，将 DIN00 端的输出电压拉低到 0V 左右，DIN00 的状态又通过数据线到 FLAG51 的 8255 上，因此我们的程序只要读回 8255 的状态时，即能得知外部信号的真或假，进而做为控制及处理上的依据。

INPUT NUMBER CONFIGURATION

	JP1,2 CLOSE	JP2,3 CLOSE
IN00	X00	X10
IN01	X01	X11
IN02	X02	X12
IN03	X03	X13
IN04	X04	X14
IN05	X05	X15
IN06	X06	X16
IN07	X07	X17
IN08	X20	X24
IN09	X21	X25
IN10	X22	X26
IN11	X23	X27



ISOLATED INPUT BOARD(12 BITS)

图 12-1

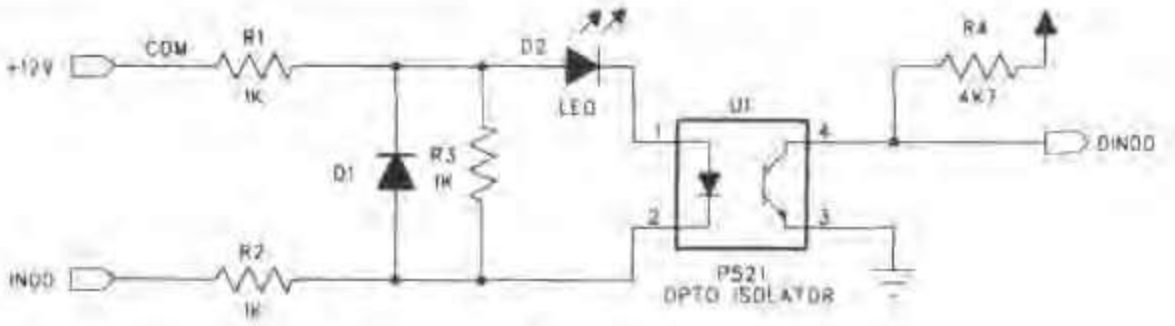


图 12-2

一片隔离输入板可提供 12 个输入点，并且可视需要扩充到 24 个输入点，图 12-3 是隔离输入板的零件布置图。当系统只有一片隔离输入板时，请把 JP1 与 JP2 短路（共有 12 点），刚好占用输入 8255 的 PA7-PA0 与 PC3-PC0；若是扩充的输入板时，就必须把 JP2 与 JP3 短路，把输入信号转到 8255 的 PB7-PB0 与 PC7-PC4，同时连接输入板与 FLAG51 控制板的 34P 数据线就必须有 3 个母接头，分别接到 FLAG51 与两片隔离输入板上，亦即 24 点输入信号可共享一条 34P 的数据线。

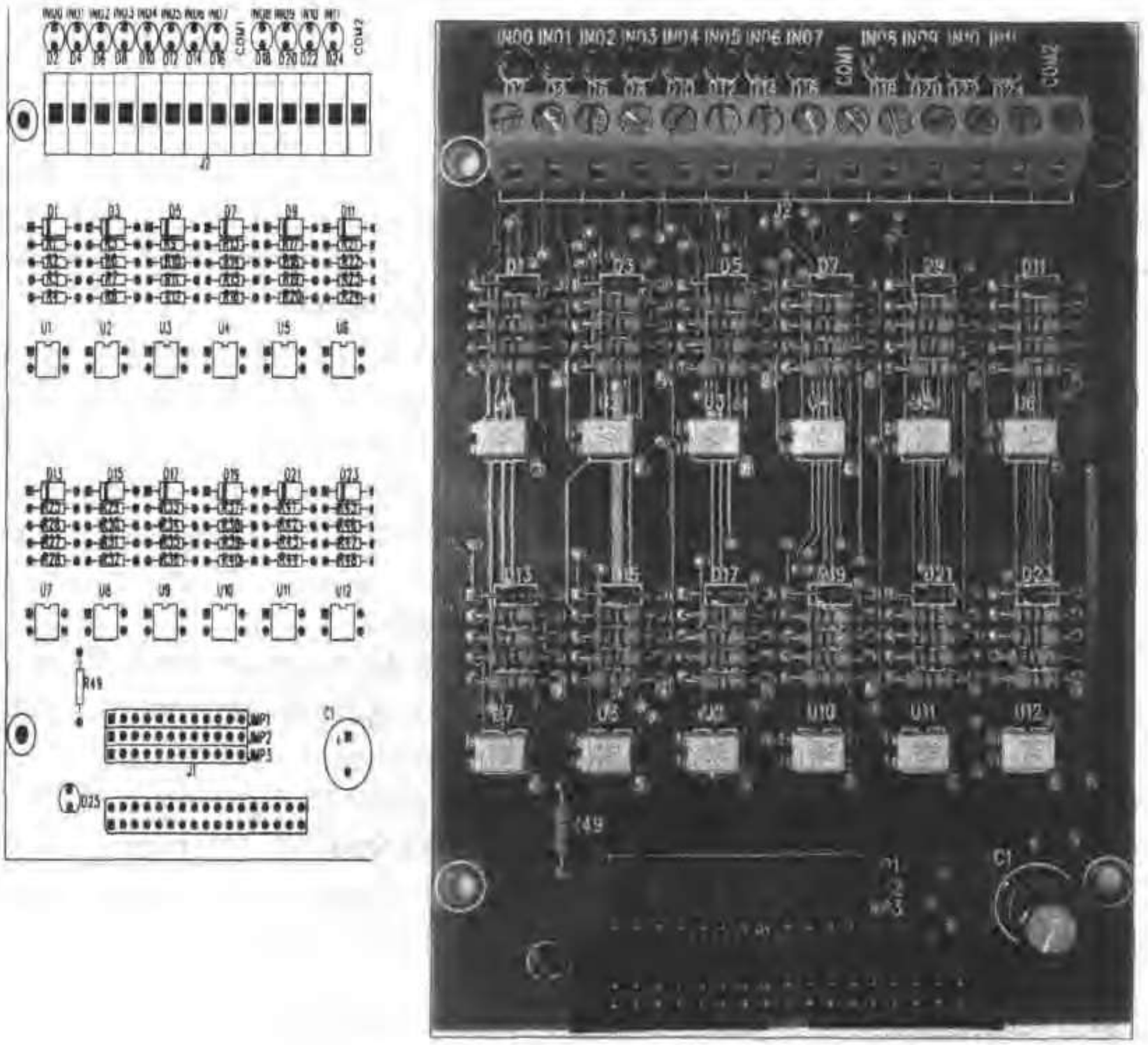


图 12-3

注：隔离输入板的零件位置图与实物图。

隔离输入板也很适合自己动手 DIY 的读者，只要把所有零件按布置图上的位置焊妥，并且再做一次零件值的确认，然后把隔离输入板与 FLAG51 控制板用 34P 数据线相连，先开启 FLAG51 的+5V 电源，该电源也会通过数据线传到隔离输入板上，所以隔离输入板上的 POWER 电源灯会亮着。如果您的电源供应器上还有+12V 的输出时最好了，请由 12V 输出端接一条电源线到隔离输入板的共同输入端（COM1 或 COM2），然后再拉一条 GND 线到输入点 IN00 上，如果标示 D2 的 LED 点亮时，代表该点的隔离输入端正常，请用电表检查光耦合 IC 的 3 和 4 脚的电压差；若是 0V 左右时，表示光耦合 IC 的输出正常，接着把 IN00 上的 GND 线移开，该光耦合 IC 的 3 与 4 脚的电压差应回升到 5V 左右，若不是 5V 时，这代表光耦合 IC 可能损换须更换。依此原则分别检查 12 个输入点，若都正确时，最后请将 12 个 JP 跳线分别接上，这就完成一块自己装的隔离输入板了。

实际的应用场合中，有可能输入信号所使用的电源与 FLAG51 的电源是分开独立的，此时另一组电源只须将其+12V 端接到标示 COM1 或 COM2 共同端上，然后将信号输出接到各个 IN 输入端即可，两组分别独立的电源，其 GND 地端并未相接，仅凭借光耦合 IC 传递状态，可大大地提升系统抗噪声的能耐，而这正是隔离输入板的精神所在。

12-2 RELAY 输出板的线路说明

有了隔离输入板后，也要有正式的输出控制板才行，我们将其命名为 RELAY 输出板，每片 RELAY 输出板提供 8 个机械式的继电器（Mechanical Relay）与 4 个固态继电器（Solid State Relay），前者提供交流或直流信号切换用，每个接点各有 10A 左右的电流，后者则供做 AC 负载（3A 以内）控制用，请参考图 12-4 的继电器输出板线路图。

每片 RELAY 输出板共占用输出用 8255 的 12 个输出点，经过 74LS04 先将输出信号反相，接着到输出缓冲 IC ULN2803 上，2803 本身又做了一次反相（第二次反相）才到继电器或 SSR 上，我们的线路上特别安排当该 RELAY ON 时，对应的 LED 也会亮了起来，这种做法并没有很高深的技巧，但是对硬件上的排错却相当有帮助。S1 到 S4 是 4 个内含光隔离的固态继电器，由于 AC 端在负载 ON/OFF 时会有较大的干扰信号出现，所以采用光隔离后再做输出控制，是 AC 控制最正确的做法。继电器输出板与隔离输入板一样，可以再扩充一倍的输出点，JP 跳线的连接方式（参考图 12-4）也和隔离输入板相同。

也许会有人疑问，为什么要加入 74LS04 做反相呢？拿掉两个 74LS04 对线路丝毫没有影响！为何要多此一举呢？还是有原因的，而且原因不在继电器输出板而是在 FLAG51 控制板上。当 FLAG51 系统开机刚刚启动时，FLAG51 是完全管不到两个 8255 的状态如何，但对 8255 而言，RESET 之后就默认成 ALL INPUT 的模式，也就是 24 个输出端都提升成 HIGH 状态，若我们的继电器输出板缺少了这两个 74LS04，当 ULN2803 输入是 HIGH 时，其输出就反相降成 0V，这会导致所有的 RELAY 及 SSR 都进入 ON 的状态，这个“大大的”缺陷虽然能用软件来弥补，可是如果很不幸地写程序的人忘了这个重要的关键点时，就会由此衍生出许许多多的问题来，设计线路的人不可不谨慎。

继电器输出板上的零件不是很多，也很适合 DIY 的，自行焊接时注意不要把 IC 的方向颠倒，LED 的方向若弄倒反了一定是不会亮的，由于机械式的继电器另外需要+12V 的电源，所以请准备的电源供应器。

OUTPUT BOARD (12 BITS)

8 RELAY N.O. OUTPUT + 4 SSR OUTPUT

OUTPUT NUMBER CONFIGURATION

	JP1,2 CLOSE	JP2,3 CLOSE	TYPE
OUT00	Y30	Y40	RELAY
OUT01	Y31	Y41	RELAY
OUT02	Y32	Y42	RELAY
OUT03	Y33	Y43	RELAY
OUT04	Y34	Y44	RELAY
OUT05	Y35	Y45	RELAY
OUT06	Y36	Y46	RELAY
OUT07	Y37	Y47	RELAY
OUT08	Y50	Y54	AC SSR
OUT09	Y51	Y55	AC SSR
OUT10	Y52	Y56	AC SSR
OUT11	Y53	Y57	AC SSR

图 12-4

除了有+5V 的输出外，还要多一组+12V 的输出，否则继电器就没办法动作了。由于 RELAY 输出板在制作线路板时漏了一个+12V 的电源输入端，所以必须由电源供应器上拉一条电源线到图 12-4 标示+12V 的接点上。

图 12-5 是 RELAY 输出板的零件位置图与实物图。

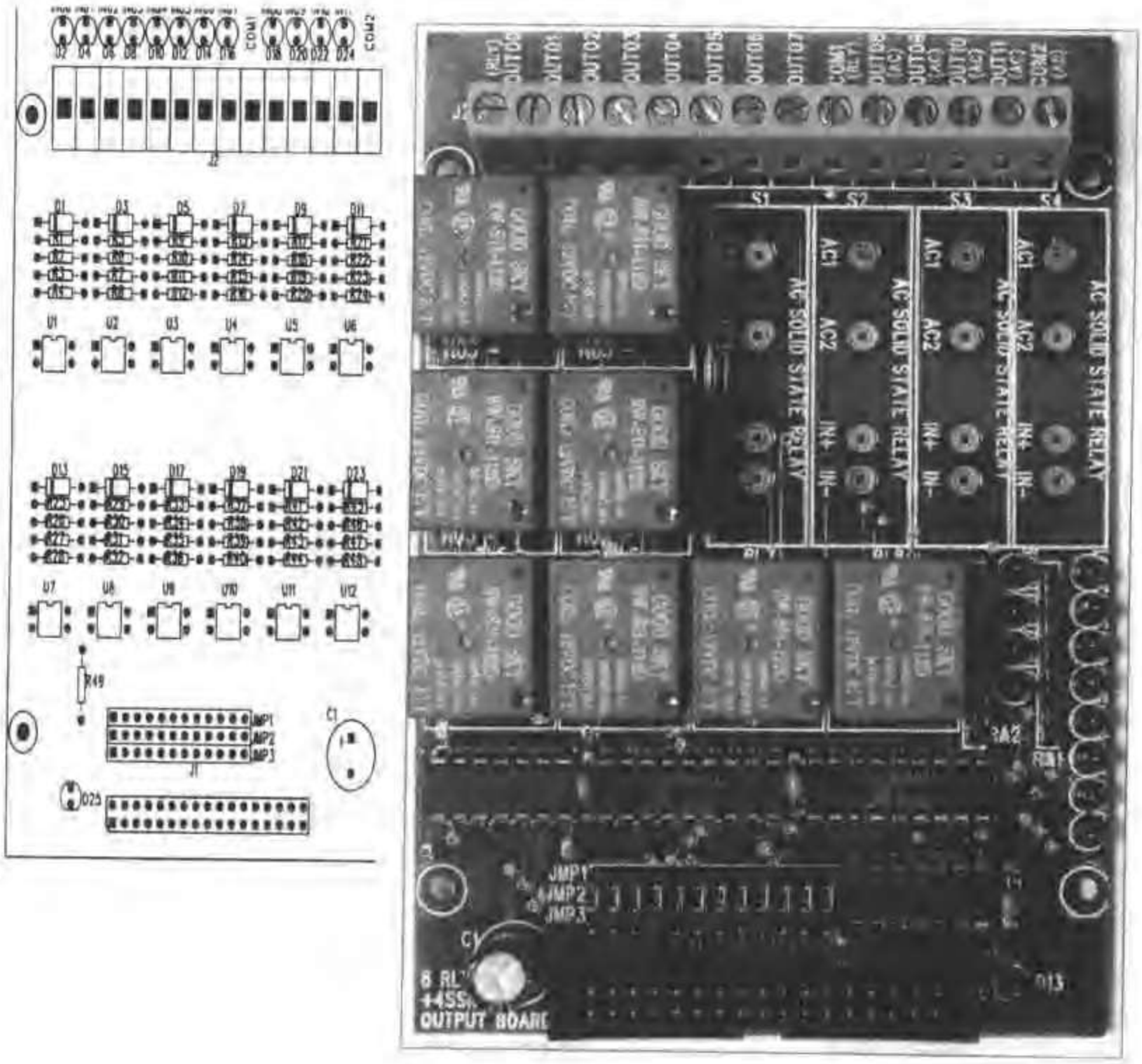


图 12-5

12-3 输出/输入板的动作验证

如何测试输出/输入板是好的？方法很简单，我们只要写一个小小的汇编程序，把隔离输入板的信号读入，不做任何数值的处理，直接把该值转送到输出板上。换句话说，只要我们看到 IN00 LED 灯亮的同时，OUT00 的代表 LED 也会跟着点亮。如果上述动作重复 12 次都正确的话，这就代表这两块板都是正常的。

试验前我们要完成以下必要的接线：

（1）FLAG51 的 CN1 接隔离输入板。

（2）FLAG51 的 CN2 接 RELAY 输出板。

（3）完成电源线的配置及检查。

〔测试程序 1〕TESTIO.ASM

```
IN_PORT EQU A000H
 OUT_PORT  EQU B000H
 ;
 RESET     MOV    DPTR, #IN_PORT+3
           MOV    A, #9BH
           MOVX   @DPTR, A              ;设定为 ALL INPUT
           MOV    DPTR, #OUT_PORT+3
           MOV    A, #80H
           MOV    @DPTR, A              ;设定为 ALL OUTPUT
 ;
 $LOOP     MOV    DPTR, #IN_PORT        ;读入 I_PA
           MOVX   A, @DPTR
           MOV    DPTR, #OUT_PORT
           MOVX   @DPTR, A              ;直接送到 O_PA
           SJMP   $LOOP
```

测试程序会把 INxx 的输入信号直接转送到 OUTxx 上，所以，当我们把 COM 点接+12V 且 IN00 接成 OV 时，RELAY 输出板上的 RL1 也会动作，您会听到 RELAY 切换的清脆声响，而且对应的 LED 也会被点亮，这时我们可以用三用电表的欧姆档去检查 OUT00 与 8 个 RELAY 的共同点 COM1（RLY）是否有接通，重复上述动作共八次即可确认继电器的功能是否正常。

图 12-6 是 SSR 输出的试验配线。

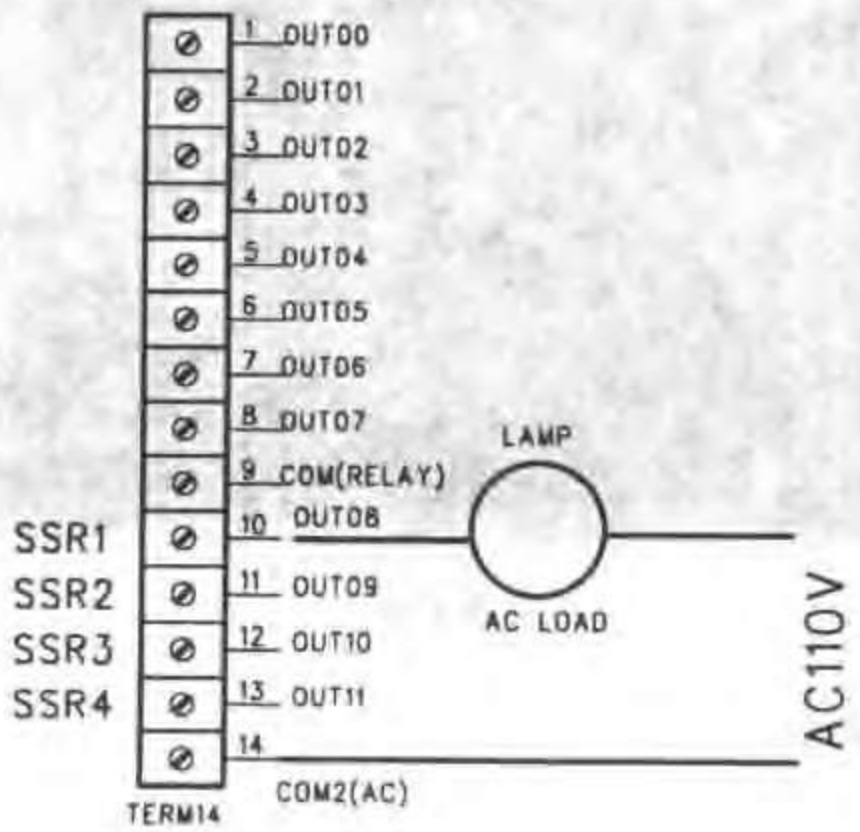


图 12-6

图 12-7 是 FLAG51 以 I/O 监视板的 DIP SW 为输入，输出仍是继电器输出板。

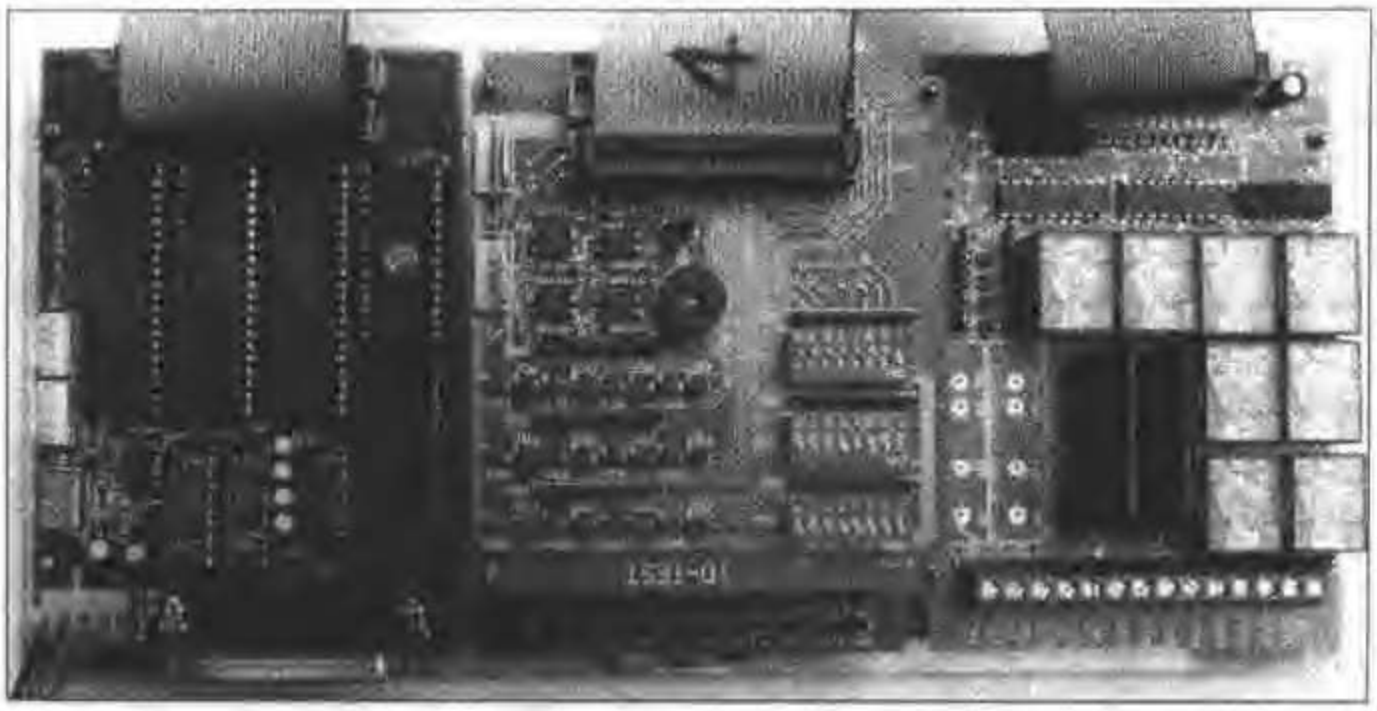


图 12-7

接着是试验 AC SSR 负载控制的部分，此时请找来一个规格是 110V 的小灯泡，参照图 12-6 的接法接妥电线（小心！有触电的危险），当我们把 IN00 输入端接地时，代表 SSR1 的 LED 会亮，同时在负载端的交流小灯泡也会被点亮，依此原则改变跳线 4 次，即可检查出 4 个 SSR 是否都正常动作了。由于 RELAY 与 SSR 本身都有方向性，自己装时出错的机会微乎其微，倒是几个反相 IC 与 34P 数据线接头在焊接前再多看一眼，即可避免出错。我们为了测试及实验用，自己用电烙铁焊了十来片 RELAY 控制板，成功率几乎是百分之百。

图 12-8 是 FLAG51 加 I/O 输出/入板的实物图。

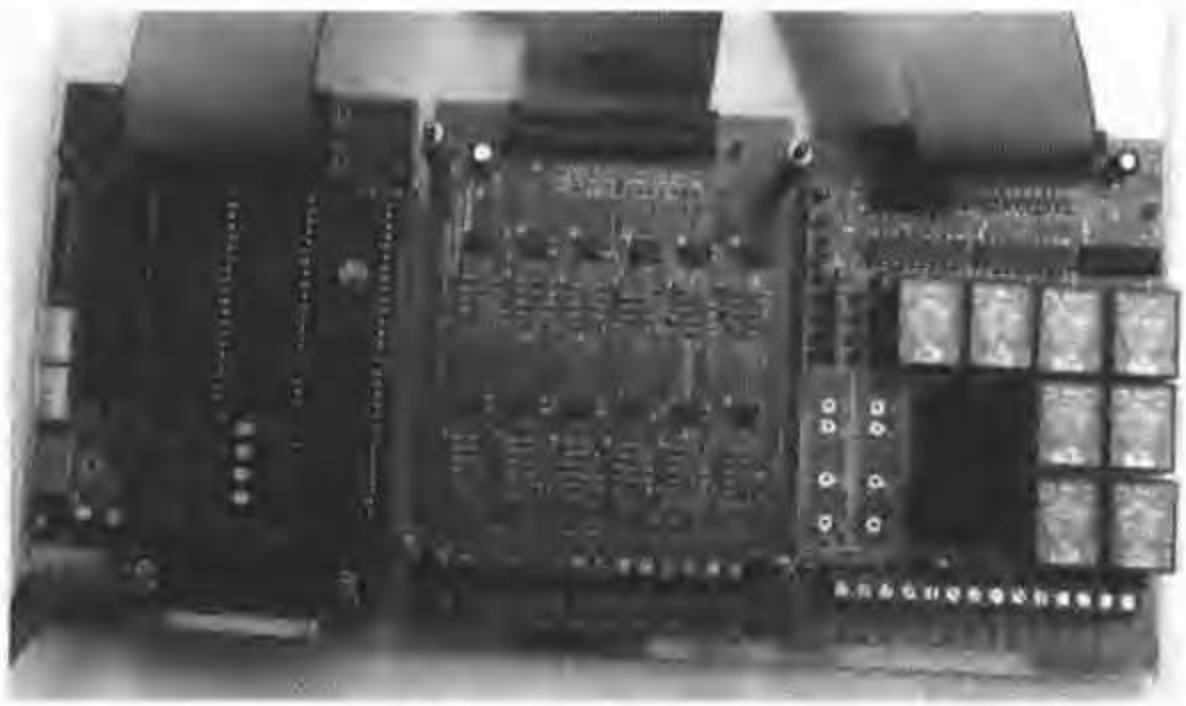


图 12-8

12-4 本章使用软件

本章使用软件为：2500AD 8051 Assembler。

12-5　本章使用硬件

本章使用硬件如下

（1）FLAG51 单片机控制板。

（2）隔离输入板。

（3）RELAY 继电器输出板。

（4）K817/TL521 光耦合器。

（5）继电器。

12-6　相关信息网站

您可经由下列公司、网站获取更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.toshiba.com：查询光耦合器信息。

http://www.sharp.com：查询光耦合器信息。

http://www.omron.co.jp：查询继电器信息。

http://www.takamisawa.co.jp：查询继电器信息。

http://www.sensor.com.tw：查询 SSR 信息。

第 13 章

8051 应用实例 FLAG-DISP

七段显示器的显示程序是许多新手首次碰到的程序，从这个程序中你会学到 8051 开机后要做那些设置及起始动作，另一个重要的课题是中断服务程序的导入，这当中又有通过查表的方式找出显示的数据，这也代表着单单显示一组数字并不如我们想象中那么容易。

许多微电脑控制的设备都有七段显示器，显示系统的状态或是测量的值，通常这类的显示电路为了简化硬件都是用扫描的方式达成，亦即每次只点亮一个数字，然后切换到下一个数字的显示。由于扫描的速度非常快，其每个数字所分配的时间都少于 10ms，再加上人眼的视觉暂留的效应，会使我们看起来好像所有的显示数字都同时出现一样。在本章里我们将谈到一组七段显示器的设计与规划始末。如果你是 8051 的新手，请特别花下心思，仔细研究我们的 FLAGDISP.ASM 程序，因为它内部有许多写法与常规都是新手所迫切需要掌握的。

13-1 AT89C51 应用实例：FLAG-DISP 线路说明

FLAGDISP 是 AT89C51 的应用实例之一。请先看图 13-1，这是一个可以显示 12 个数字的七段显示界面，有关显示的所有动作完全由一枚 AT89C51 控制，12 个数字是安排成每 6 个一行，以符合计数器显示的规格，七段显示器的规格是共阳极的（Common Anode）。FLAG-DISP 显示板的尺寸与螺丝孔与 FLAG51 控制板完全一致，所以两块线路板可以用铜柱固定在一起，然后以一条 34P 的数据线连接在一起。

由于 AT89C51 已内含 4KB 程序空间及 128 字节 RAM，所以板上极为精简只剩两个 74LS138（负责显示位置），两个 74LS273（负责显示内容）和两个 ULN2803（电流放大以便推动七段显示器），为了能与 FLAG51 做串行通信，所以系统使用的石英晶体的频率也是 11.0592MHz。FLAG-DISP 上另外也提供一个 34P 的接头，以便接收由外部送来的并列信号。

FLAG-DISP 显示的动作大略是这样的：AT89C51 先由 P0 端口送出两组显示的数据到 U4 与 U5 74LS273 上，接着由 P3.5~P3.7 共 4 个 bit 送出显示的位置，此时相对应的两个显示数字将被点亮，上述的动作持续做 6 次后就可以完成 12 个数字的所有显示。显示的值由何处

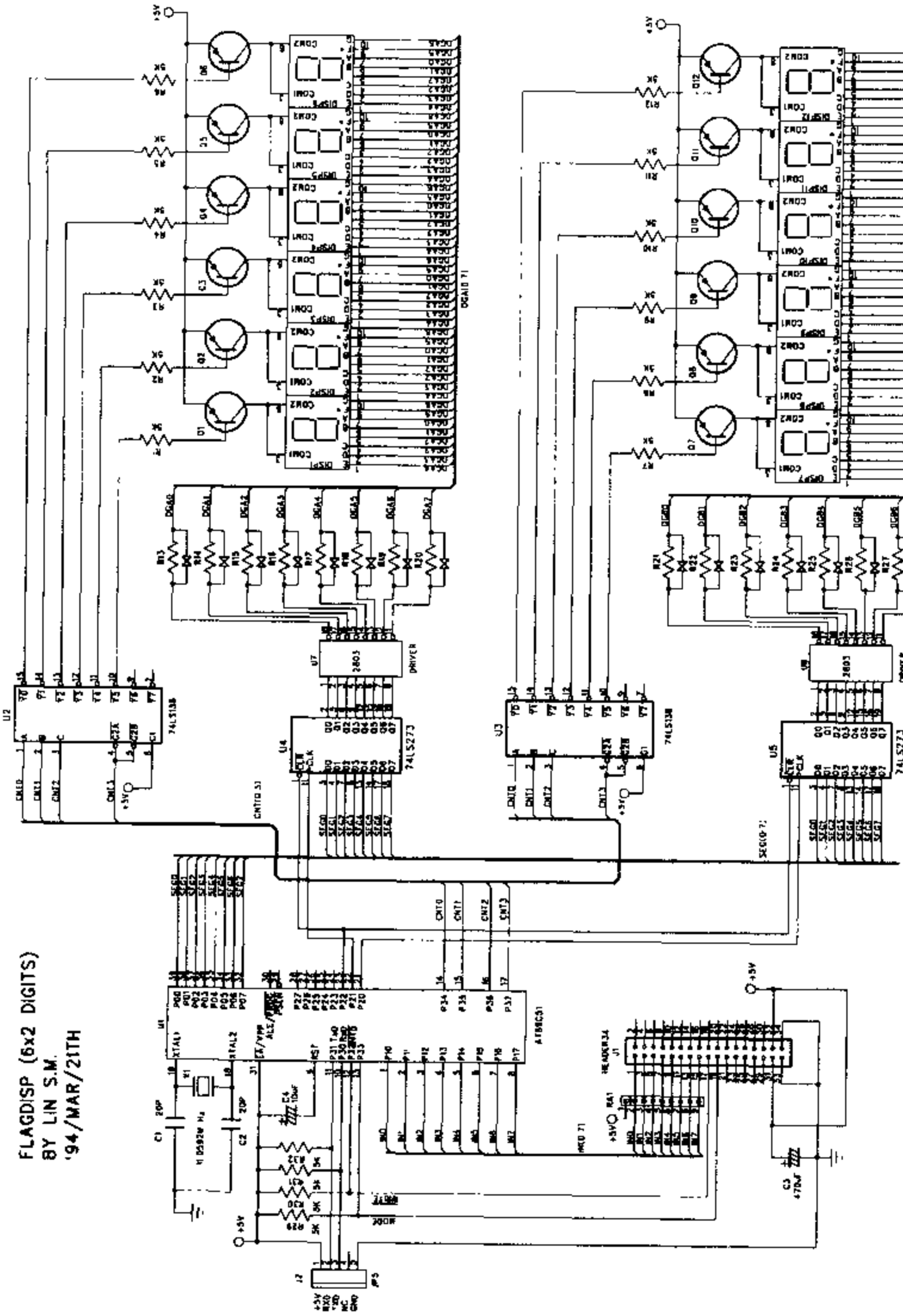


图 13-1 FLAG-DISP 的完整线路图。

注：FLAG-DISP 的完整线路图。

来呢？我们的想法是这样的，可以同时接受串并列的显示数据，并列的显示数据可由图中的 J1 传到 89C51 的 P1 端口上，接着由 J1 上的 WRITE 脚产生 INT0 中断信号给 AT89C51，串行数据则直接传入串行通信端口 SBUF 上。

图 13-2 是 FLAG-DISP 的成品图。

图 13-2

表 13-1 是 FLAG-DISP 零件表

表 13-1　　FLAG-DISP 零件表

名　称	数　量	说　明
FLAG-DISP 线路板	1	尺寸 85mm×125mm，双面贯孔电路板
ULN2803	2	电流放大用
74LS138	2	显示位置选择用
74LS273	2	显示数据 LATCH 用
AT89C51	1	内含 4KB 闪存
2SA1015	12	日系 PNP 小信号晶体管
0.5"七段显示器	12	共阳极型的 0.5 英寸七段显示器
10uF 电解电容	1	RESET 电路用
470uF 电解电容	1	电源滤波用
20PF 陶质电容	2	
0.1uF 积层电容	5	去除电源噪声用
10K（9P）排阻	1	提升电阻
4.7K 电阻	16	1/4W
11.0592Mz 石英晶体	1	系统与串行通信用，请勿改用其他频率晶体
34P 数据线公接头（180 度）	1	双排接头脚距 2.54mm
4P 电源接头	1	脚距 3.96mm

FLAG-DISP 不仅仅可作 12 个数字显示而已，它可以学习的范围包含：

（1）七段显示器的使用与控制。

（2）显示数据的规划与安排。

（3）定时中断的使用。

（4）并列数据的接收与显示。

（5）串行数据的接收与显示。

（6）AT89C51 的使用与数据保护的 LOCK bit 使用。

（7）AT89C51 的刻录程序的理解。

假如您接到类似 FLAG-DISP 软件设计的项目时，该如何着手规划程序呢？又如何做才不会到最后又问题百出呢？请继续看我们的说明。

13-2 AT89C51 应用实例：FLAG-DISP 软件说明

设计单片机的控制程序最忌讳的是马上就着手写汇编程序，最后的结果往往转变成“头痛医头，脚痛医脚”的窘境，比较好的做法是先把整个控制动作流程确定下来，统计一下有几个中断程序要处理，接着逐一列出所有待调用的函数，并详细规划内存的使用情况，最后也是最重要的一环：清清楚楚地列出每个函数如何验证其正确性以及如何排错等等，当所有动作都就绪后，才作整个系统的大整合，如此系统本身的正确性将会提高很多，而且程序也非常容易排错与修正。当然不采取上述的系统规划原则也能完成程序的开发，不过，这些程序内部的变换性与修正性将变得很差，只要程序一发现错误要修改时，一定让程序设计师吃足苦头。

以下就是我们所整理出的函数名称及其说明：

函数 SYSTEM_INIT：系统启始，并清除相关数据区。

函数 ENABLE_INT：系统允许中断的设定程序，分别允许计时、串行以及外部硬件中断。

函数 GET_PATTERN：将 BCD 码转换成七段显示器的专用码。

函数 DISPLAY：七段显示器的输出显示程序。

函数 SERIAL：串行中断的显示数据接收程序。

函数 EXT_INT0：外部中断的显示数据接收程序。

函数 TIMER0：定时器定时中断以便送出显示数据到七段显示器上。

函数 DATA_ADJ：显示数据的处理与调整。

函数 HEX_TO_BCD：将一个 16 进制数值转换成 BCD 码，程序可以改成接计数值而非显示值。

函数 CONVERT：分别转换两个 16 进位值，成为六位数的 BCD 码。

显示数据暂存区（30H-3BH）：共 12 个字节，摆放欲显示的数据。

FLAG-DISP 整个显示程序的主要流程如下：

（1）系统初始化，并清除数据缓存等空间。

（2）系统允许中断：TIMER0、SERIAL 及 INT0 中断。

（3）显示数据暂存区的内容。

（4）进入固定的显示与等待循环（WAIT LOOP），之后就等待以下三种中断信号的到来。

◆ 定时 TIMER0 中断发生时：

（1）取出一组显示数据（该数据早已转换成七段显示码）。

（2）送出七段显示码及显示位置。

（3）结束定时中断服务例程。

◆ 外部 INT0 中断发生时：

（1）由 P1 端口取到一个字节的显示数据。

（2）显示数据经过计算及转换后存入显示数据暂存区中。

（3）结束外部中断服务例程。

◆ 串行 SERIAL 中断发生时：

（1）由串行通信端口 SBUF 处取到一个字节的显示数据。

（2）显示数据经过计算及转换后存入显示数据暂存区中。

（3）结束串行中断服务例程。

13-3　FLAG-DISP 的显示格式定义

由于 FLAG-DISP 显示卡是我们自行设计的产品，所以所有的数据格式都可以自己完全掌握及安排，串行通信时的通信协议为 9600bit/s, NO PARITY, 8 DATA bit,1 STOP bit，以下我们谈的 FLAG-DISP 显示格式说明，不论串并列时都适用本格式，本格式可以显示两组六位数值。每次欲显示或更新数字值时，必须由外连续送入 12 组（字节）显示数据值，此数据值的格式为标准 BCD 码（占 4bit），若要加上小数点指示时，请把该 Byte 的最高位（bit 7）设成 1 即可。若其中有一位数不想显示时，只要把让字节的低四位设成 1111 即可。比方说，我们想要显示的内容是：上排显示 12345.6，下排显示_ _0.234（_代表空白），则我们要送给 FLAG-DISP 显示卡的数据格式为：

上排共六个显示数字

显示 1	不含小数点	所以送出的数据是　01H
显示 2	不含小数点	所以送出的数据是　02H
显示 3	不含小数点	所以送出的数据是　03H
显示 4	不含小数点	所以送出的数据是　04H
显示 5.	含小数点	所以送出的数据是　05H
显示 6 不含小数点所以送出的数据是　06H		

下排共六个显示数字

显示 __	不含小数点	所以送出的数据是　0FH
显示 __	不含小数点	所以送出的数据是　0FH
显示 0.	含小数点	所以送出的数据是　80H
显示 2	不含小数点	所以送出的数据是　02H
显示 3	不含小数点	所以送出的数据是　03H
显示 4	不含小数点	所以送出的数据是　04H

所以要送给 FLAG-DISP 显示卡的数据串依序为 01H，02H，03H，04H，85H，OFH，OFH，80H，02H，03H，04H。只要 FLAG-DISP 收妥 12 组显示数据后，即可作相对应的数

据显示。如果是由 FLAG-51 经由 34P 数据线控制 FLAG-DISP 显示板时，刚好 8255 的 PA 端口负责并列数据的传送，PC0 位当成 WRITE 信号，对 AT89C51 产生外部中断的要求信号，至于 PC1 的 MOVE（以后另有他用），则始终保持 HI 状态，并列传输的动作是相当单纯与直接的，其汇编语言的控制程序可以作如此安排：

（1）设置 8255 为输出模式，并随即把 PA 与 PC 端口都设成 1。

（2）把欲显示值转换成六位数的 BCD 码，然后存入显示暂存区域中。

（3）逐一送出显示数据到 FLAG-DISP 中，总共 12 次，最后把 PC0 与 PC1 都设成 1，以免 FLAG-DISP 收到额外的中断信号。

假设 FLAG51 的显示暂存区在外部存储器的 8200H-820BH 时，则输出到 FLAG-DISP 的驱动程序，应该类似以下的写法：

```
OUT_TO_FLAG_DISP
        MOV     A, #FFH
        LCALL   OUT_1_BYTE          ;先送出显示格式码（详细说明下期再述）
        MOV     DPTR, #8200H        ;DPTR 指到显示暂存区的开头
        MOV     R0, #12             ;共有 12 字节要送出
$1      MOVX    A, @DPTR            ;取得一个显示数据
        LCALL   OUT_1_BYTE
        INC     DPTR
        DJN     R0, $1              ;送出 12 个数据给 FLAG-DISP
        RET
;
OUT_1_BYTE
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR, #PPI_PA
        MOVX    @DPTR, A            ;显示数据由 PA 端口送出
        MOV     DPTR, #PPI_PC
        MOV     A, #11111110B       ;WRITE=0
        MOVX    @DPTR, A            ;WRITE 变成 LOW
        LCALL   DELAY               ;DELAY 至少 100μs 的间，以便 89C51 中断
        MOV     A, #11111111B
        MOVX    @DPTR, A            ;WRITE 升成 HIGH
        POP     DPL                 ;取回原来的 DPTR 值
        POP     DPH
        RET
```

13-4 FLAG-DISP 的学习方向

FLAG-DISP 内含 AT89C51 以及相关的显示控制程序，只要准备妥+5V 的直流电源供应器，接上电源到 FLAG-DISP 显示板后，内部的程序会先做 12 个显示器的自我测试，所有数

字由 0 到 9 变化一次，然后又加入小数点（DECIMAL POINT）由 0 到 9 变化一次，接着 12 个七段显示器全部关闭，开始等待接受外部传来的显示数据，如果到此显示都正常时，代表 FLAG-DISP 一切正确，可以作为一个单纯的 12 个七段显示外围来应用。接下来请关闭电源，用 34P 数据线连妥 FLAG51 与 FLAG-DISP，重新开启电源后，就能做各种数值显示的应用了，此时亦可加入 IO TEST 卡，做光电隔离信号的输入等应用，实验至此我们已完成一个简易具智能型的计数器了。

接下来的学习路径可分成两方面：一是研究 FLAGDISP 显示程序的写法，您可以利用万用刻录器或图 13-3 的线路读回控制程序的所有内容，再利用 DIS51.EXE（本书中所附的反编译工具）将程序还原回汇编语言的样子，再作些修改后又把控制程序重新刻录到 AT89C51 上，这些学习过程与经验是教科书上绝对吸收不到的。

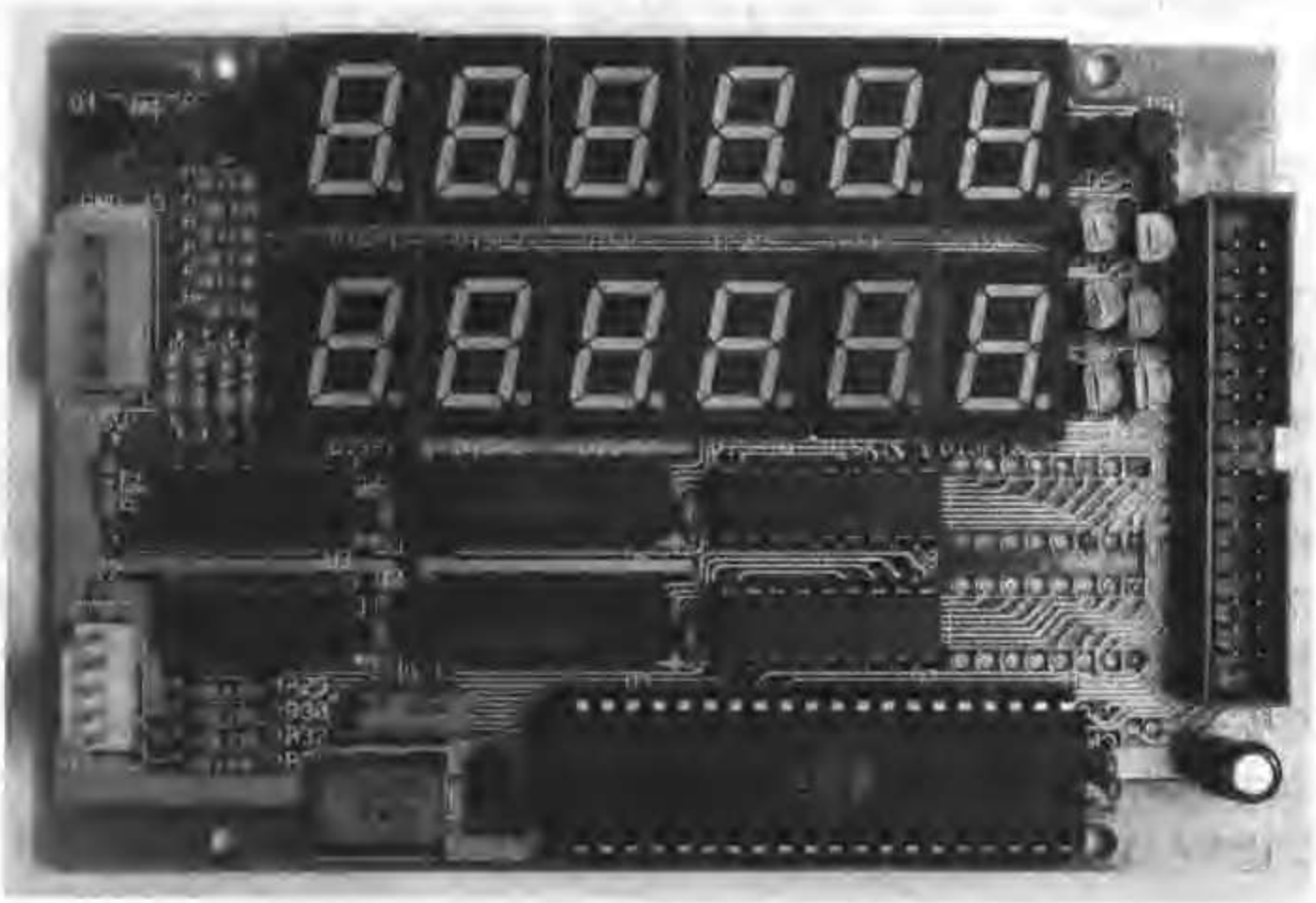


图 13-3

另一个学习方向是将 FLAG-DISP 显示卡当成一个外部组件，多做几个有关数值显示的应用程序，您应能发现 FLAG-DISP 不仅仅是块显示实验卡而已！下一章我们将利用 FLAGDISP 正式完成一个产品，它结合了计数、显示与通信的控制器，我们已经起跑了，有兴趣往单片机发展的读者请赶快加入吧！

13–5　本章使用软件

本章使用的软件为：2500AD 8051 Assembler。

13–6　本章使用硬件

本章使用的硬件为：

（1）FLAG51 单片机控制板。
（2）AT89CXX 刻录板。
（3）FLAG-DISP 七段显示板。
（4）74LS138 TTL 译码器。
（5）74LS273 TTL 8bit Latch。

13-7 相关信息网站

您可经由下列公司、网站取得更进一步的信息：
http://www.chipware.com.tw：查询单片机控制板相关信息。
http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。
http://www.ti.com：查询 74LS138/74LS273 信息。

13-8 FLAGDISP.ASM 原始程序

FLAGDISP.ASM（七段显示控制程序）是完全用 8051 汇编语言写成的，显示的数据可以用并列或串行的方式传给 AT89C51。当它接收这些显示数据后，就会把对应的显示值显示在板上两组六位数字群上。

这个程序会利用定时中断去扫描所有的七段显示数字，剩下的时间再来做数值到七段显示码的转换等动作。虽然 AT89C51 仅做数字显示，但是本程序的架构还是相当完整的，有外部中断、定时 10ms 中断与串行通信中断等等，主程序部分还有 16bit 的二进制对 BCD（二进制编码的十进制）转换程序与系统自我测试程序等等，我们认为它相当适合 8051 的高级人士参考。您也可以参考这几章的线路图，自行 DIY 相同或类似的电路，再将本程序刻录到 AT89C51 当中，即可同时建构出一个完整的七段显示系统。

FLAGDISP.ASM 原始程序，请查看书附光盘中的 CH13_FLAGDISP.ASM 文件。

第 14 章

FLAG-DISP 显示板应用与 DIY

本章将告诉你 FLAG-DISP 显示的几种正确格式，这样一来在进行各种显示值的应用时，FLAG-DISP 就可派上用场了。

“旗威”的 FLAG-DISP7 段显示程序总共规划了 6 种最常用的显式格式，这一章里我们将尽量详细地说明其中的用法，只要您知道 FLAG-DISP 显示的正确格式后，就可以立即做各种显示值的应用，两个占 6 位数 7 段显示器的应用是相当多的！

14-1 FLAG-DISP 显示格式说明

◆ 显示格式 1：BCD 码方式

FFH（导前码）+UUUUUU（上显示值）+DDDDDD（下显示值）总共要送出 13 个字节。

这是最常用的显示格式，UUUUUU 与 DDDDDD 分别是欲显示的六位数 BCD 数值，若该位要加入小数点显示时，请把此位的 BCD 码再加上 80H，即其 bit7 等于 1 就可，若某位数想成为空白时，请把对应的送出值设成 0FH 即可。本格式每次一定要送足 13 个字节，其中的 UUUUUU 代表上排的显示数值，DDDDDD 代表下排的 6 个显示数值。

FLAG-DISP 开机后，内定即为本显示格式，以下是本显示格式的输入范例：

上排显示数字：123456
下排显示数字：789012

则送给 FLAG-DISP 的数据为 FFH,01H,02H,03H,04H,05H,06H,07H,08H,09H,00H,01H,02H。

上排显示数字：12345.6
下排显示数字：___78.9（_代表空白）

则送给 FLAG-DISP 的数据为 FFH,01H,02H,03H,04H,85H,06H,0FH,0FH,0FH,07H,88H,09H

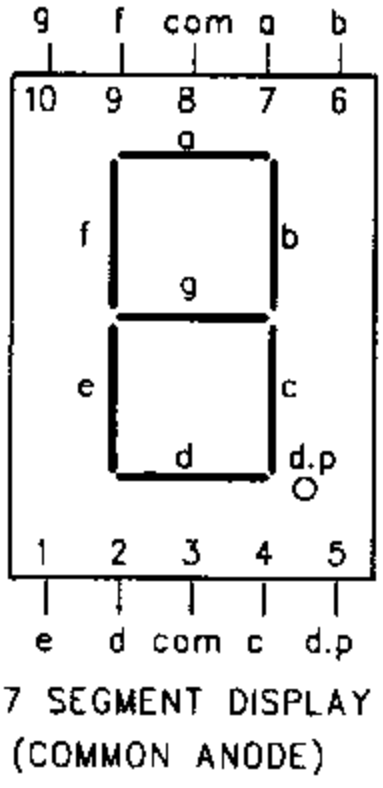
上排显示数字：12_34.5（_代表空白）
下排显示数字：1__78.9（_代表空白）

送给 FLAG-DISP 的数据为 FFH,01H,02H,0FH,03H,84H,05H,01H,0FH,0FH,07H,88H,09H

◆ 显示格式 2：7SEGMENT 码方式

FEH（导前码）+UUUUUU（上显示值的七段码）+DDDDDD（下显示值的七段码）共有 13 个字节要送出。

当显示的数据是数字以外的值时，就可以采用 7SEGMENT 码的格式，本格式是直接传 7 段显示器的笔划（SEGMENT）给 FLAG-DISP，所以就可以显示如 H，L，P 或 A 等非数字值。由于每个 7 段显示器内共有 8 划（含小数点），因此每次传给 FLAG-DISP 显示控制器的数据长度为 13 个字节（2x6+1 导前码）。7 SEGMENT 码的数据安排方式请看图 14-1，对应的 bit 等于 1 时，则该笔划点亮，反之则该笔划熄灭。例如，我们要字划 a 亮时，只要把 bit0 设成 1；若要小数点亮时，就必须把 bit7 定成 1 才可以。FLAG-DISP 一收到数据后，就不做任何转换的动作，立即将 12 个显示器的笔划数据送出。



7 SEGMENT DISPLAY
(COMMON ANODE)

7 SEGMENT DATA FORMAT							
D7	D6	D5	D4	D3	D2	D1	D0
dp	g	f	e	d	c	b	a

图 14-1

注：FLAG-DISP7 段笔划的格式及显示器的引脚图。

上排显示数字：HELP__（_代表空白）
下排显示数字：Err_20（_代表空白）

送给 FLAG-DISP 的数据为 FEH,76H,79H,68H,73H,00H,00H,79H,50H,50H,00H,5BH,3FH

上排显示数字：_____（_代表空白）
下排显示数字：Err_2.0（_代表空白）

送给 FLAG-DISP 的数据为 FE,40H,40H,40H,40H,40H,40H,79H,50H,50H,00H,DBH,3FH

◆ 显示格式 3：上排显示器 BINARY 码

FDH（导前码）+UUU（上排显示值的二进制代码，最低位在最前面）共有 4 个字节要送出。

这个显示命令只更改上排显示器的值，FLAG-DISP 收到该二进制值后，会自行转换成七段显示器的专用码，然后显示在上排的 6 位显示器上。由于传送三个字节的二进制代码值可达 16777216（十进制值），而上排显示器仅有 6 位数字，所以，FLAG-DISP 在收到这三个值后，会自动把最高位的 dbit7~bit1（共 7bit）给遮盖掉，以免数值超过 6 位数。这种显示格式无法加入小数点的指示，若您的显示应用中一定要显示小数点时，请改用显示格式 1 或显示格式 2。这种显示格式可以使主控制器（FLAG51）的负担减到最小，遇到显示值要变化时，只需对 FLAG-DISP 丢出 4 个字节，就可得到一个最大六位数的整数值。

上排显示数字：_____0（_代表空白）

送给 FLAG-DISP 的数据为 FDH,00H,00H,00H

上排显示数字：__4096（_代表空白）

送给 FLAG-DISP 的数据为 FDH,00H,10H,00H

◆ 显示格式 4：下排显示器 BINARY 码

FCH（导前码）+DDD（上排显示值的二进制代码，最低位在最前面）共有 4 个字节要送出值。

这个显示命令只更改下排显示器的值，其余动作与显示格式 3 完全相同。

◆ 显示格式 5：上排显示值加减 1

FBH（导前码）只要 1 个字节，即可将上排显示的整数值加 1。
FAH（导前码）只要 1 个字节，即可将上排显示的整数值减 1。

只要主控制器有下过显示格式 3 的命令后，就可以把上排的显示值加减 1，而且只要送出一个字节的数据，即可通知 FLAG-DISP 将内部储存的整数值加一或减一。

◆ 显示格式 6：下排显示值加减 1

F9H（导前码），只要 1 个字节即可将下排显示的整数值加 1。
F8H（导前码），只要 1 个字节即可将下排显示的整数值减 1。

只要主控制器下过显示格式 4 的命令后，就可以把下排的显示值加减 1，而且只要送出一个字节的数据，即可通知 FLAG-DISP 将内部储存的整数值加一或减一。

上述的显示格式几乎包含了大部份显示器的应用。而且能大大地降低主控制器的程序负荷，所以 FLAG-DISP 将有极大的应用与发展空间。

14-2 FLAG-DISP 显示板的 DIY 步骤

旗威的数字显示器分别以“完成品”和“DIY 自己装”两种形式与读者见面，前者只要加上数据线与电源后，立即可以用程控显示的内容与方式。“DIY 自己装”则多了一道由使用者自行焊接的手续。自己装时请先由高度最低的零件开始焊接，以下是自己装的步骤：

◆ 步骤 1：焊接电阻（只有一种规格），电解电容与 0.1 uF 积层电容和 11.0592 MHz 的石英晶体。

◆ 步骤 2：焊接 40 脚的 IC 座，本 IC 座稍后将插上 AT89C51。

◆ 步骤 3：焊接 9P 的 10K 电阻，电阻的共同点通常有一圆标示点，请依板上的标示方向焊接。

◆ 步骤 4：焊接 12 个 PNP 晶体管（2SA1015），请依板上的标示方向焊接并保持零件的整齐。

◆ 步骤 5：焊接六个 IC（74138，74273 和 2803），方向绝对不能倒反。

◆ 步骤 6：焊接 J1 与 J2 接头，J1 是与 FLAG51 联机的并列接头，如果方向错了就没法实验了。

◆ 步骤 7：焊接 12 个共阳极的七段显示器，小数点的方向应朝 CPU 这方。焊接前请花两分钟先用三用电表确认显示器的八段笔划都没有问题，焊上后要再检查就很费事。

◆ 步骤 8：焊接 4P 电源接头（脚距为 3.96mm），PC 板上有标示+5V 与 GND 的位置，接上电源时请勿弄反。

◆ 步骤 9：再次检查焊接点无误后，装上 4 根铜柱及螺丝，准备开始测试 FLAG-DISP 显示板。

图 14-2 为 FLAG-DISP 的完成图。

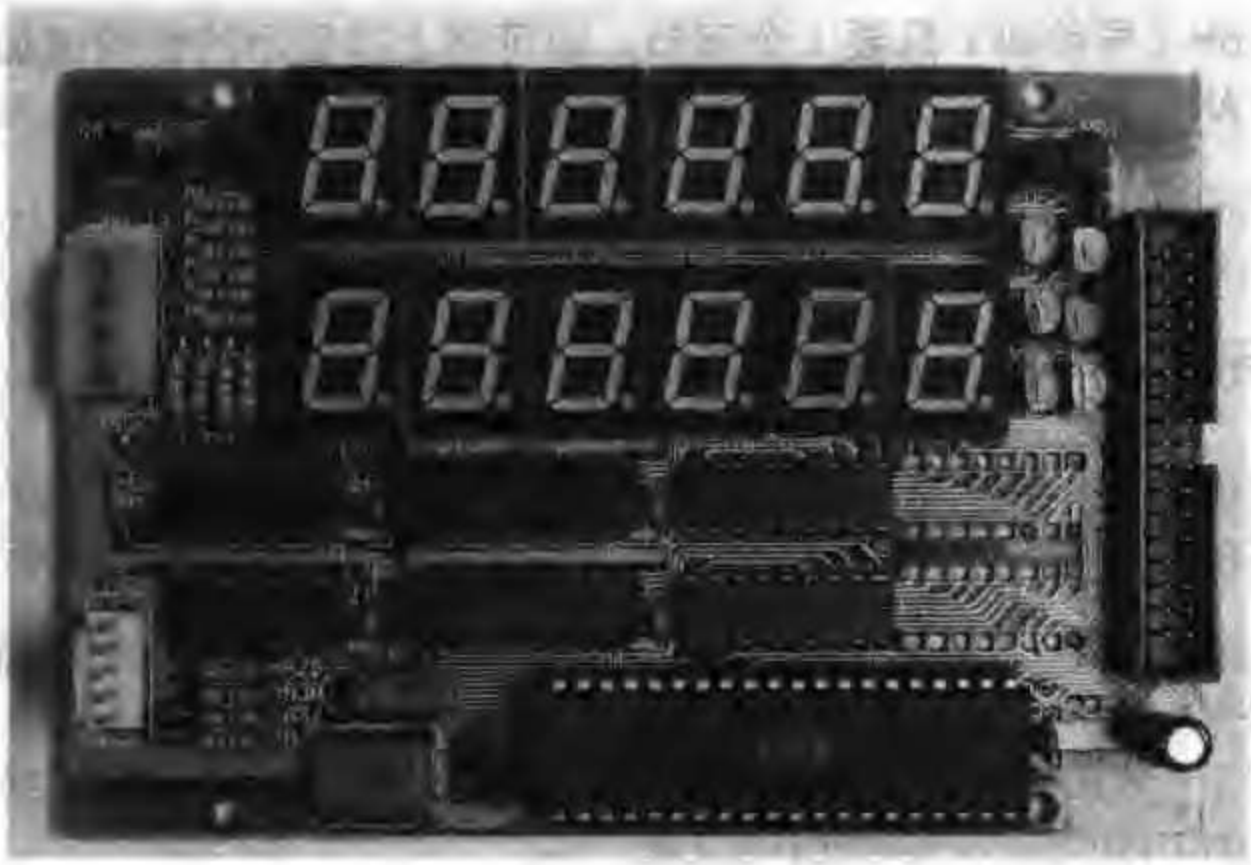


图 14-2

图 14-3a 为零件面图（Component Side）。

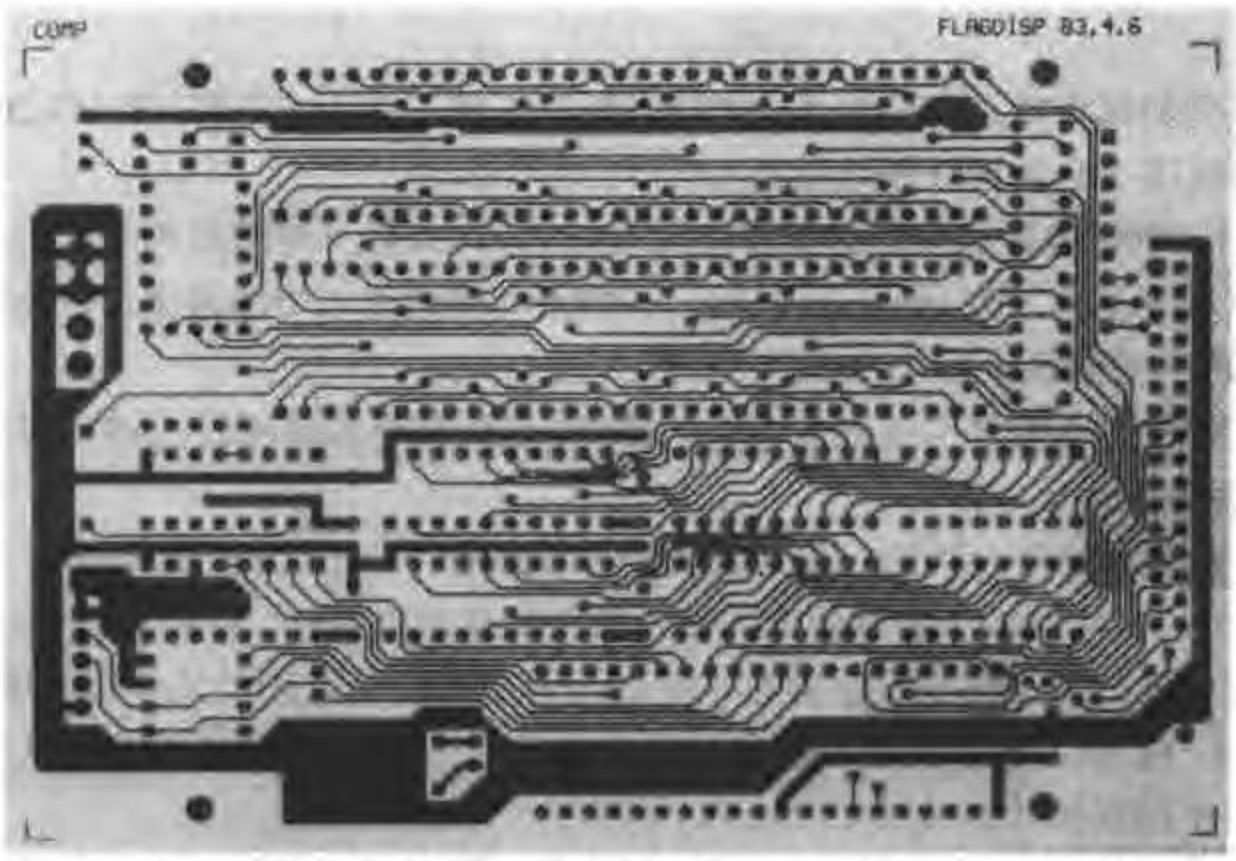


图 14-3a

图 14-3b 为焊接面图（Solder Side）。

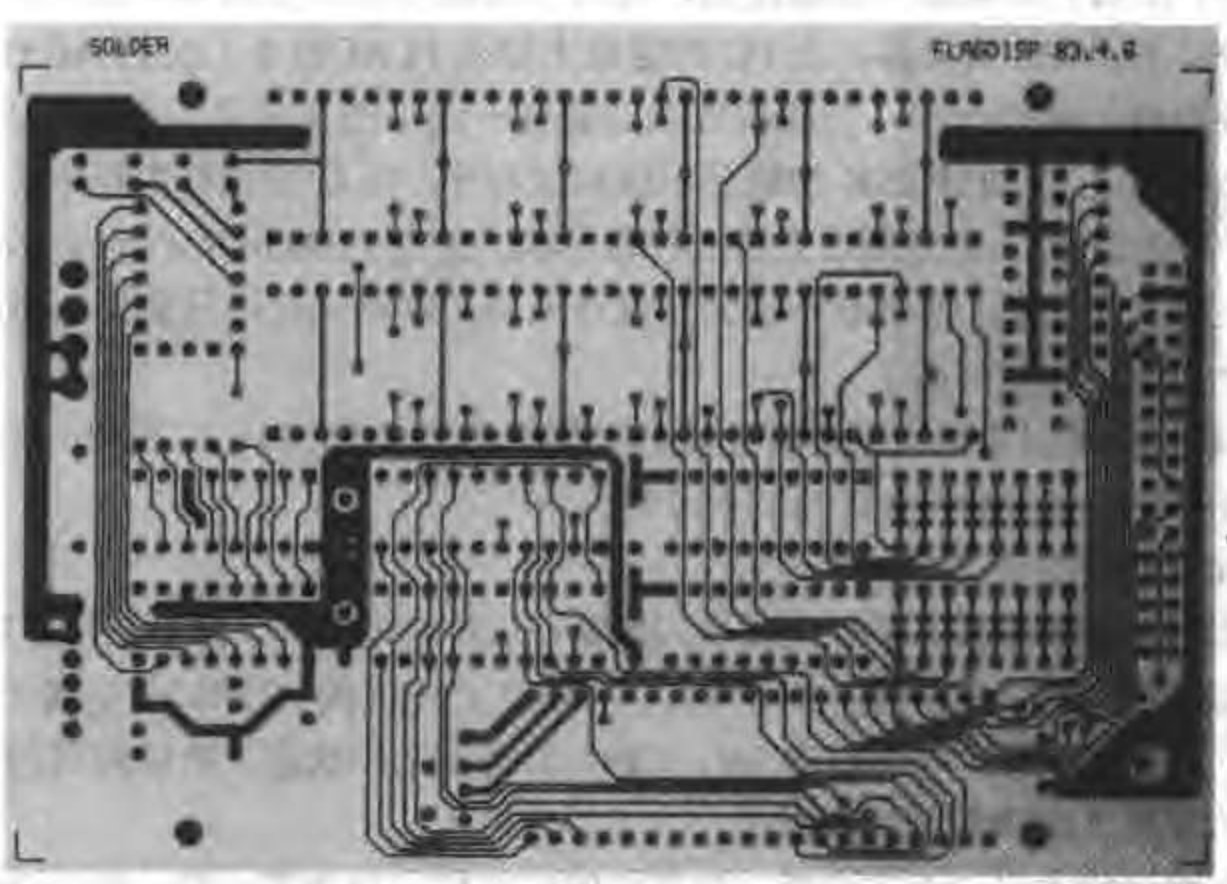


图 14-3b

14-3　FLAG-DISP 显示板的测试步骤

自己装的读者请做完基本的硬件焊点及进行测试前请准备以下仪器及设备：

（1）+5V 直流电源一组，其供应电流量应该在 2A 以上。

（2）FLAG51 控制板，可透过 34P 数据线下达显示命令给 FLAG-DISP 7 段显示器。

（3）PC 个人计算机（286/386/486 均可），至少有一个通信端口（COM）能与 FLAG51 控制板联机。

请依照下列测试步骤：先确定 FLAG-DISP 已正确工作后，再将 FLAG-DISP 控制板与 FLAG51 控制板用 34P 数据线相连接。

◆ 步骤 1：确认电源供应器的输出电压为+5V，请先打开电源后再用三用电表测量其输出端，输出电压应在+4.75V 到+5.25V 间，否则请勿继续做以下的测试，以免因电源错误而损坏 FLAG-DISP 控制板。

◆ 步骤 2：找到本控制卡提供的电源连接线，红线接电源供应器的+5V 端、黑线接 GND 端，接妥后打开电源。

◆ 步骤 3：电源打开一秒后，12 个显示器全部熄灭，FLAG-DISP 开始进入自我测试阶段。

◆ 步骤 4：FLAG-DISP 控制板上的显示器全部显示数字 0，然后是 1，2……最后到 9，显示器变化值时，请留意每个数字的显示字划是否正确，最后数字又全部灭掉。

◆ 步骤 5：FLAG-DISP 控制板接着又重新显示数字 0 到 9，但此时又加入了小数点的显示，最后数字全部灭掉。

◆ 步骤 6：测试到此阶段时，FLAG-DISP 进入随时接受显示命令的模式，只要收到显示值时即可将值显示到上下两组七段显示器上（各占六位数字）。

◆ 步骤 7：关闭电源，请用 34P 数据线将 FLAG51 控制卡（CN1）与 FLAG-DISP（J1）相连，FLAG51 控制卡上也加入电源接头，待所有接线都确定无误后才重新打开电源。

◆ 步骤 8：打开 PC 的电源，在 PC 的键盘上键入 FLAG51 1（或 FLAG51 2），执行 PC 与 FLAG51 控制板的联机程序，若联机成功后会有一连串的文字显示在屏幕上。

◆ 步骤 9：按下 LTDISP.TSK 加载显示值测试程序，按 G 键执行本程序，如果本数字显示板正常时，FLAG51 板上会分别用上述三种显示模式控制 FLAG-DISP 显示卡，最后 FLAG51 把 7 段显示卡当成计数器，两个显示值会一直往上加，本测试程序只要稍加修改就可成为一个精度到百分之一秒的定时器了。

◆ 步骤 10：如果以上显示动作都正确时，代表 FLAG-DISP 一切顺利，可以开始自行写应用程序，自行控制 FLAG-DISP 的显示内容。

FLAG-DISP 使用注意事项：

（1）电源千万不能接反，所以电源供应器第一次打开电源时，请再三检查后才打开电源。

（2）电源请勿经常开开关关，由于 FLAG-DISP 上使用的 AT89C51 是属于 CMOS 架构，电源关闭后，只要电源端的电压超过 3V，CPU 仍处于工作状态，所以瞬间的电源关闭又打开，是不会影响 FLAG-DISP 的动作。

（3）34P 数据线本身已有方向性，排在线的红色标示处为第一脚，所以连接时请依方向插入即可。

14-4 AT89C51 刻录与使用时的考虑

上一节中，曾经提到刻录 AT89C51 最佳的工具是万用刻录器，我们在开发 FLAG-DISP7 段显示器时，也是抱持这种观念。因为采用市面上现成的刻录工具，可以更有效地节省开发

时间，可是，这个理想最后还是没有实现！以我使用的万用刻录器为例，它刻录 EPROM 是绝对没问题的，但是在刻录内含闪存的 AT89C51 时，却是始终怪怪的，一枚新的 AT89C51 第一次刻录没问题，可是数据擦除之后，进行第二次刻录时就过不了关了。换句话说，就是数据刻录不成功，起先我们总是先怀疑是否 AT89C51 正好是瑕疵品，换一个新的 AT89C51 试试看，结果刻录成功！所以还是认定刻录器是好的，问题是刻录不成功的数量持续增加，当数量到达 20 个时，我不得不怀疑刻录器是否提供了正确的刻录时序给 AT89C51？否则怎么会有如此多的故障出现！

若一枚 AT89C51 的零售价格是 90 元时（2003 年 AT89C51 的价格已经降至 20 元以下），在两天的开发测试时间内已经损耗掉近二千元的零件费用了，这个费用已足足可以买到半台万用刻录器！依我们粗略的估算，完成整个 FLAG-DISP 显示程序至少要进行将近百次的刻录手续，那要花掉多少钱呢？我们最后的如意算盘是自行制作一台专门刻录 AT89C51 的刻录器，一来可以降低刻录时的故障率，二来可以完整地学习到闪存刻录的时序与技巧，这些当然是花上千元买万用刻录器所学不到的知识。关于刻录器 DIY 的探讨与制作将在“8051 单片机彻底研究经验篇”书中再做详尽的报告。

14-5　本章使用软件

本章使用软件如下：

（1）2500 AD 8051 Assembler。

（2）FLAGDISP，ASM 请参看前一章的清单。

14-6　本章使用硬件

本章使用的硬件如下：

（1）FLAG51 单片机控制板。

（2）AT89CXX 刻录板。

（3）FLAG-DISP 七段显示器。

（4）74LS138 TTL 译码器。

（5）74LS273 TTL 8bit Latch。

（6）万用刻录器 Universal Programmer。

14-7　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.ti.com：查询 74LS138/74LS273 信息。

http://www.ledtech.com.tw：查询七段显示器的引脚图。

第 15 章

FLAG-DISP 的创新应用

前一章我们谈了 FLAGDISP 的线路设计与程序规划。“旗威科技”用这片显示板做了许多应用，例如：机械状态指示器，多功能的定时器以及工具机的位置指示器，由于大部分数值转换的程序都已在 FLAGDISP 内处理完了，FLAG51 只要送出值后就可继续其他的程序处理，减轻了许多系统的负担。

15-1 七段显示器的再利用

最近几天公司正在忙着为客户安装测试设备，以增加其产品的生产量。在这些产业设备中，我们用了许多独立的定时器来计算待测物通电的时间。这些市售普及型定时器的价位约在几百元上下，其动作是收到计时开始的信号后，就开始计时并且将时间值显示出来，显示的单位是 0.1 秒。更高级的机种还可以设定计时值，当设定值到达时，会送出额外的信号，我们就可以用这个信号来做其他控制的依据。采用现成定时器的好处是立即可用不需要重新设计，缺点则是变化性较低，多功能的机种价位就高了许多，使用数量一多时成本就增加许多。

8051 单片机最擅长的就是计时与计数的功能了。在“旗威科技”现有的产品线中，已经有 7 段显示器 FLAGDISP，它是以一枚 AT89C51 Flash Microcontroller 当成控制主体，可以用来显示两排共 12 个数字。我们打算修改 AT89C51 内部的程序，把原先只接受显示命令的 7 段显示器转变成两组多功能的定时器。图 15-1 是 7 段显示器的线路图，12 组显示数字是以扫描的方式显示。AT89C51 必须产生 10ms 的定时中断，强迫系统逐一去推动七段显示字划，这段时间间隔是不能过长的，否则显示时会看起来有晃动的感觉。

以下就是我们所写的 C 计时程序范例，真正的版本当中，我们又加入了 93C46 EEPROM，以存放所设定的两组计时值，这样我们就可以顺理成章地取代原有的市售定时器了。由于整个程序都是我们自己所发展的，所以我们有绝对的自主权与修改权，加上 93C46 的读取程序后，程序已超过 4096 字节，因此我们的控制器需改变成有 8192 字节程序空间的 AT89C52，可是其总价格还是比两台现有的定时器便宜。

15-2　数字显示程序的宝贵经验

如果要写一个六位数的 HEX 转十进制的数值显示程序，您会怎么做呢？例如把 8704H 的十六进制值换算成十进制值 34564，大部分的人（包括我在内）都会先碰到一个相当困扰数值转换的问题，我们必须先完成该二进制值的转换后，再将 BCD 码送到七段显示器上，这才能把值显示出来。以我们打算做的计时显示器为例，每隔 0.1 秒钟就会把内部的计时值加 1，然后再调用二进制转 BCD 码的函数，得到一组六位 BCD 码，然后再经过 7 段字划（7 segments）的查表转换，最后我们得到所要的字划数据后，并暂时存入 SRAM 中，接着由中断程序取出字划数据并送达显示器上，我们才得以顺利看到所有的数字。

这些做法从推断及研究上看来都是正确的，所以我们的 C 程序最初是这样子：

```
int display(unsigned int t,int u_d)
{
 char d1,d2,d3,d4,d5;

        d1=t/10000;       t=t%10000; /* 取万位数 */
        d2=t/1000;        t=t%1000;  /* 取千位数 */
        d3=t/100;         t=t%100;   /* 取百位数 */
        d4=t/10;            /* 取十位数 */
        d5=t%10;            /* 取个位数 */

        if (u_d==UP) /* 决定上组或下组数字的显示 */
        {     u_digit[4]=hex_table[d1]; u_digit[3]=hex_table[d2];
              u_digit[2]=hex_table[d3]; u_digit[1]=hex_table[d4];
              u_digit[0]=hex_table[d5]; }
        else
        {     d_digit[4]=hex_table[d1]; d_digit[3]=hex_table[d2];
              d_digit[2]=hex_table[d3]; d_digit[1]=hex_table[d4];
              d_digit[0]=hex_table[d5]; }
}
```

这个方法是对的，若在个人计算机上执行也是可以的，在 8051 的七段显示器上执行也行得通，可是速度却稍嫌慢了一些，这是因为 FLAGDISP 系统上，有一半左右的时间在做七段字划的定时扫描，而连续有八个 16 位除法运算，对 8051 而言确是相当大的考验，这不仅耗时而且也增加了系统的程序空间。所以计时值虽然可以顺利显示了，但是用肉眼仔细注视显示器时，可以感觉出其中位数间变化的闪烁，这明显代表着这个函数有执行速度过慢的重大缺点。

图 15-2 为执行 C 程序 display()数值转换程序所花的时间，该函数被调用前已经全面禁止中断，以节省计算时间，但是还是花了 4.52ms，这也难怪七段显示器看起来有点闪烁，这个信号是这样得来的：调用 display()之前将 P3.0 设成 1，display()执行完后立即将 P3.0 清除为 0。

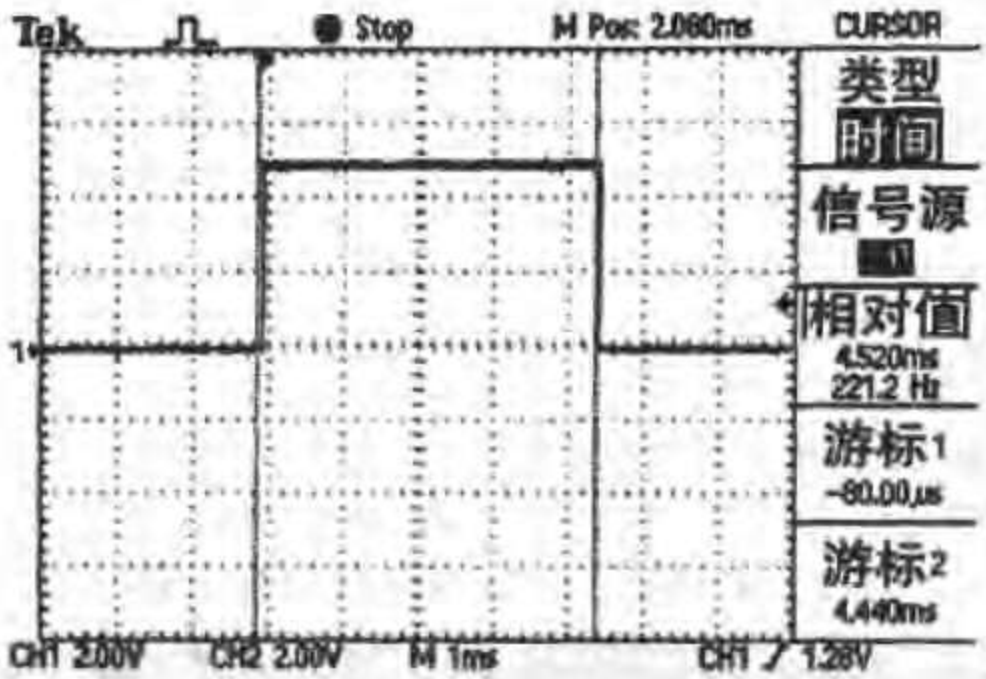


图 15-2

重新检讨整个动作时我们又有重大的发现：系统每次 10ms 中断时，会把中断计次值加 1，当该值到达 10 时代表 0.1 秒的时间过了，所以就要将内部记录的计时值加 1，这个 0.1 秒计时值才是真正的显示值，而整个系统中最重要的环节就是这里。

如果单纯只做十进制值的显示时，并不需要先前提到如此复杂的二进制转 BCD 码的转换公式，我们可以事先规划出一个 5 字节的数据区，程序一开始时全部填入 0 值，每次 0.1 秒钟到达时，就把个位值加 1，当个位值到达 10 时则把个位值归 0，并且将十位数值加 1，更高位数值也是依此原则来调整，这种方式不仅单纯，而且执行速度非常快，以下就是修正后的C程序写法：

```
int adjust_timer()
{
        u_d[0]+=1;  /* 个位数加 1 */
        if (u_d[0]==10) { u_d[0]=0; u_d[1]+=1; }    /*个位数调整*/
        if (u_d[1]==10) { u_d[1]=0; u_d[2]+=1; }    /*十位数调整*/
        if (u_d[2]==10) { u_d[2]=0; u_d[3]+=1; }    /*百位数调整*/
        if (u_d[3]==10) { u_d[3]=0; u_d[4]+=1; }    /*千位数调整*/
        if (u_d[4]==10) { u_d[4]=0; u_d[5]+=1; }    /*万位数调整*/
        if (u_d[5]==10) { u_d[5]=0; }
}
```

这种写法要显示多少位数都没有问题，而且所占用的执行时间都很容易估算出来。毕竟要 8051 做加减运算当然是比乘除运算要快上好几十倍。许多写程序的技巧就是经由不断地排错后得到的，如果您不去动手做，那就永远不会得到这些知识的。我们这里所示范的例子是配合 8051 程序所做的修正，换成其他的 CPU 时，这些问题或许不会出现，但是只要有问题时，我们就要有能力去解决它，这也是我们一直强调的："随时培养解决问题的能力。"

图 15-3 是直接调整数据区 adjust_timer()所花的时间，只剩 840µs 而已，时间节省到原来的 1/6，只要写程序的观念稍做修改，就可节省大量的执行时间。

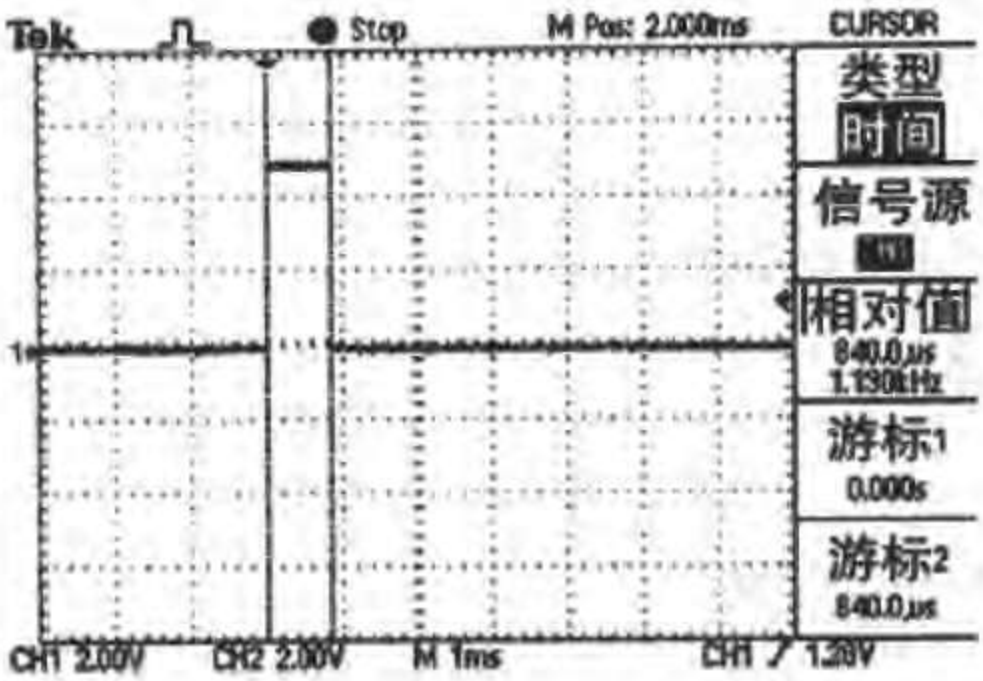


图 15-3

为了进一步证明我们写法的正确性，我们分别在计时值的计算前后加上识别信号，试验的方法非常简单，我们在程序调用的前后分别加上 SETB P3.0 及 CLR P3.0 的指令，然后用示波器观看执行该段程序所花的时间，两者的时间差异真的很大。如果整个计时程序再改用汇编语言来处理，执行速度还会更快一些，不过，用C语言来写的结果已经相当令人满意了。

如果您要下载我们所提的程序及二进制刻录文件，请直接到“旗威科技”的网站（http://www.chipware.com.tw/download.htm）取得上述文件，我们也将该程序稍微修改，以便接受光学编码器（encoder）的信号，此时 FLAGDISP 摇身一变成为一个可同时显示两轴的位置指示器了，有了这个电路及程序，机电整合的 XY 轴加工表上，就可以正确地显示出 X 与 Y 两轴正确的位置了。

大部分的人会以为写数字扫描程序是最简单不过的，只要时间一到就把扫描值依序送出即可。但是如图 15-4 所示，当 8051 的中断全部可以启用（enable）时，情况就不是那么单纯了。如果系统正在测量 AD 值，同一时间内串行中断又进来，刚好，10ms 的定时中断又到了，这时，你认为 8051 应该先处理那个工作呢?

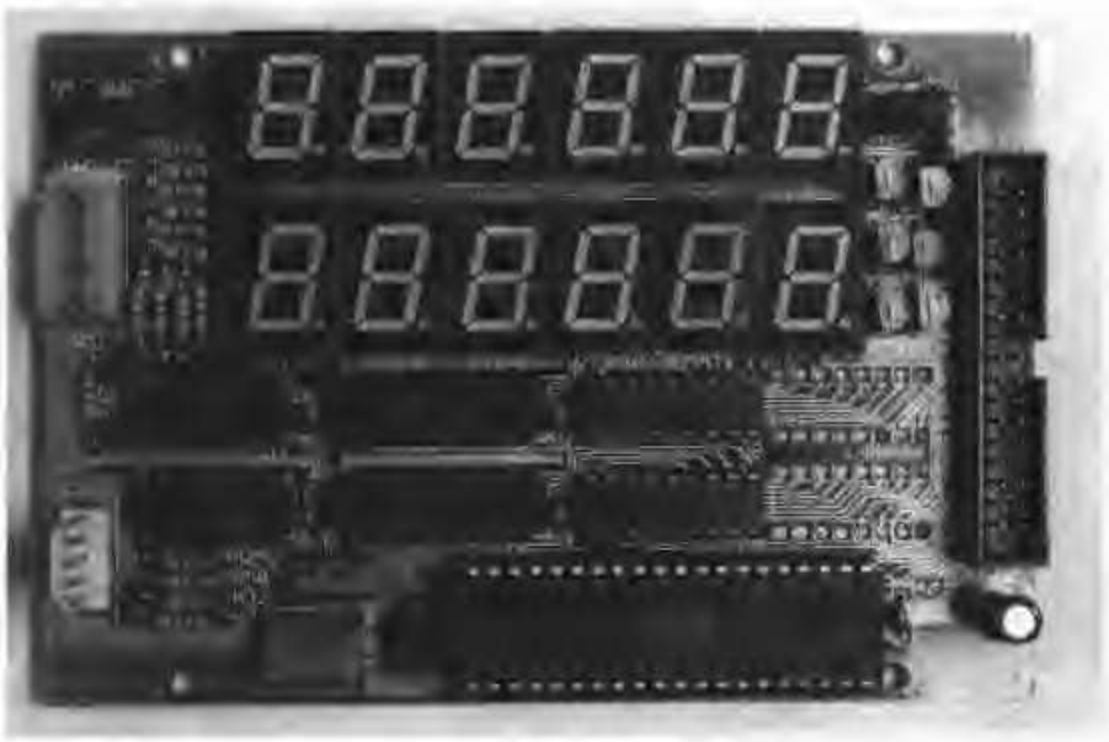


图 15-4

15-3 本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler。

（2）2500AD 8051 C Compiler。

15-4 本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）AT89CXX 刻录板。

（3）FLAG-DISP 七段显示器。

（4）74LS138 TTL 译码器。

（5）74LS273 TTL 8bit Latch。

（6）93C46 串行 EEPROM。

（7）Tektronix TDS220 储存式示波器。

15-5 相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C51 与 AT89C52 信息。

http://www.microchip.com：查询 93C46/93C56/93C66 等信息。

http://www.ti.com：查询 74LS138/74LS273 信息。

http://www.tek.com：查询 TDS220 示波器信息。

第 16 章

亲手做一台数字式温度计

你知道 IC 型感温器温度测量的相关知识吗？只要你懂得其基本测量的方法，再结合 8051 单片机的控制程序，就可以完成一个自己 DIY 的温度计了。

温度测量的应用是随处可见的，如果您在车上放一个温度计，在大热天下可以发现车内的温度竟然升高到摄氏 50 度以上。本章的目的在于给您 IC 型感温器温度测量的相关知识，只要您懂得测量温度的方法，再结合 8051 单片机的程序，就可以完成一个 DIY 式的温度计了。

16-1 无处不在的温度时测量

每天新闻联播最后的气象报告一定会提及明天各地的温度变化范围，在我们室内早已有各式各样的温度计，家中空调的控制器上也都会指出室内现在的温度情况。其实温度对人的影响是相当大的，所以已有许多方法用来测量温度，最近几年来有许多 IC 厂商纷纷推出 IC 型的测温组件，以便取代原先复杂的温度转换电路，不过因为感测组件是做在 IC 上，所以其温度测量范围都在摄氏+150 度以内，温度若再高时就会导致零件的损坏而无法度量。在众多的 IC 式感温器当中，较有代表的产品且可以买到的分别有：

◆ AD590：电流输出型的测温组件，温度每升高一度 K（凯式温度），电流增加 1 uA，温度测量范围在-55°C～+150°C，许多微计算机的实验教材都以此 IC 做温度测量的示范。

◆ DS1620：这是于 1994 年问世的感测控制 IC（由美国的 Dallas 公司开发，2001 年 Dallas 公司已并入美国的 Maxim 公司），除了测量温度外，它还可以把温度值以数字的方式（9bit）送出，温度送出的精度为 0.5°C，温度测量范围在-55°C～+125°C，DS1620 还可独立做智能型的恒温控制。

◆ SMARTEC 感温组件：这是一个只有三根脚的温度感测 IC，它最特殊之处是将测量到的温度以方波的型式输出，亦即方波的 DUTY CYCLE 是和摄氏温度成比例的，温度测量范围在-45°C～+130°C，整个测量范围的误差可以保持在 0.7°C 以内。

如果用 8051 的单片机来读取温度并做控制时，上述三种感温组件都是相当适合的。不过，AD590 所送出的温度值必须再经过 AD 转换后，才能得到实际的数字温度值，这是上述三种感温组件中处理起来最为麻烦的。DS1620 的温度读取就比较直接了，只要保留三支接

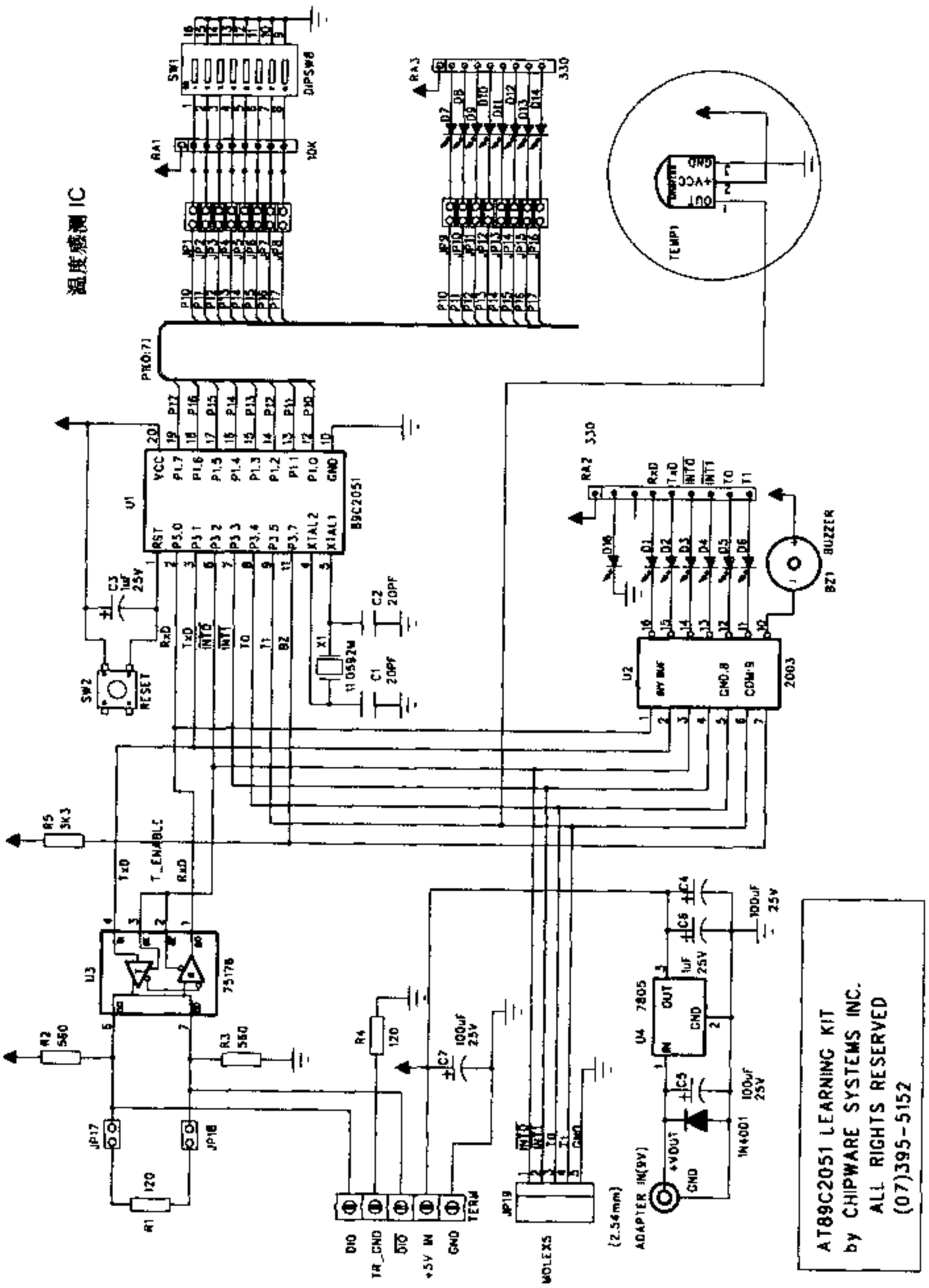


图 16-1

脚来控制 DS1620 即可。如果用 SMARTEC 的感温组件只要用一支数字输入脚即可得到温度值，若想一次测量多点温度时，用这个感测组件也是相当方便的，所以我们在这一章的 8051 应用及制作中，首先就是要用 SMARTEC 感温 IC 完成一个简便但精度高的温度显示器。

SMARTEC 感测组件有三根脚，除了电源与地脚外，就剩下方波的输出脚了，其输出信号与温度的关系为 Duty Cycle=0.320 +0.00470×temp（temp 温度的单位为摄氏温度）。

如果您是初次碰到这类的公式，一定会纳闷：怎么规划及写程序才是对的？由于原厂宣称该传感器的线性误差可以保持在摄氏 0.2 度以内，所以我们可以由上面的关系式直接推导出温度来：

temp = (Duty Cycle-0.320)/0.00470
= (Duty Cycle/0.00470)-(0.320/0.00470)
= 212×(Duty Cycle)-68（小数点以下值忽略）----公式 1

这表示说：如果我们只要计算出 SMARTEC 感温器的 Duty Cycle 值（单位为%），把此值代入公式后即可得到温度值。例如 Duty Cycle=45%时，那就代表温度为摄氏 27.4 度。由 SMARTEC 所提供的数据得知：该感测 IC 的输出频率约在 1～4KHz 间，换算成周期为 1000μs 到 250μs 左右。若用 8051 单片机来计算其时间应该是没有问题的。我们这里所做的示范是用 ATMEL 的 AT89C2051 当成主控制器，其内部有 2048 字节的闪存程序空间，并且可以有 15 个输出入点，我们打算用 AT89C2051 的 P1 端口（共 8 点）去显示温度值，然后剩下的输出入点去读取感温 IC 所送出的方波信号。请看图 16-1 的初步规划线路图。

16-2　DutyCycle 的测量

由于 SMARTEC 所送出的信号是隐含在其 Duty Cycle 里，不论其送出的频率如何，只要计算出 DutyCycle 的比例即可算出摄氏温度值来，可是要怎样度量 SMARTEC 所送出的方波 DutyCycle 呢？以下是我们所采取的方法：

◆ 步骤 1：把两个计数值都清除成 0，第 1 个计数值记录整个方波所要的时间 t1，第 2 个计数器记录方波是高电位状态所花的时间 t2，所以其 DutyCycle=（t2/t1）×100%。

◆ 步骤 2：先判断方波信号的状态，如果处于 HI 时，则一直等到状态由 0 变换成 1 时，才同时启动 t2 与 t1 的计数。

◆ 步骤 3：当方波为 1 的状态消失时，t2 要停止计数。

◆ 步骤 4：当方波刚要再由状态 0 转换成状态 1 时，代表此一输出周期已经结束，所以也应该停止 t1 的软件计数器。

◆ 步骤 5：我们现有的 8051 汇编语言子程序仅能处理整数值的加减乘除运算，所以计算 DutyCycle 时，不能用（t2/t1）×100 这个公式，因为 t2 永远小于等于 t1，这会使得 t2/t1 的结果一直是 0，乘以 100 之后还是 0。所以正确的计算式是先把 t2 乘 100，然后再除以 t1 就可以得到一个介于 0～99 的整数值了。

◆ 步骤 6：前一个步骤所得到的 DutyCycle 值是一个介于 0~99 的整数值，可是公式 1 所要代入的值却是 0.00 到 0.99 的小数值，所以我们再把温度值放大 100 倍，以方便程序的处理，所以我们的温度计算式变成：

temp100=100×temp=212×(DutyCycle×100)-6808----公式 2

◆ 步骤 7：把 DutyCycle 值代入公式 2 中即可以得到 100 倍的温度值 t100，只要取出该值的千位值及百位值，正是我们所要的温度值了，而 temp100 的十位数值正是测量温度的小数点以下第一位数值。

◆ 步骤 8：temp100 值除以 100，取出得到的商值当成最后的温度值，然后把此值转换成标准的 BCD 码，以便显示及串行通信输出用。

图 16-2 是温度显示及推动电路。

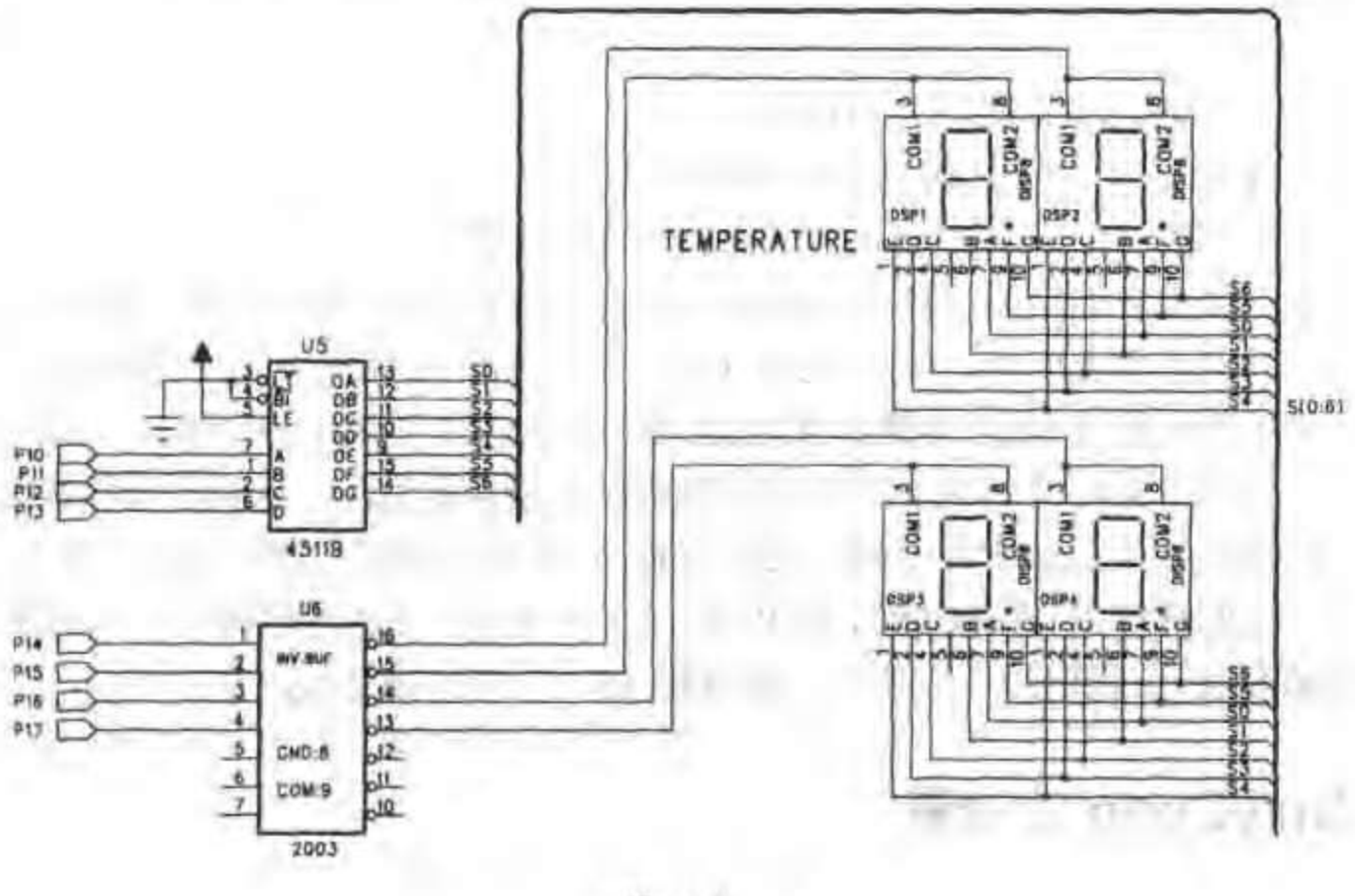


图 16-2

图 16-3 是用示波器所观看到 SMARTEC 感温组件所产生的波形。其 DutyCycle 为 46%，经过换算后约是摄氏 29.5 度左右。

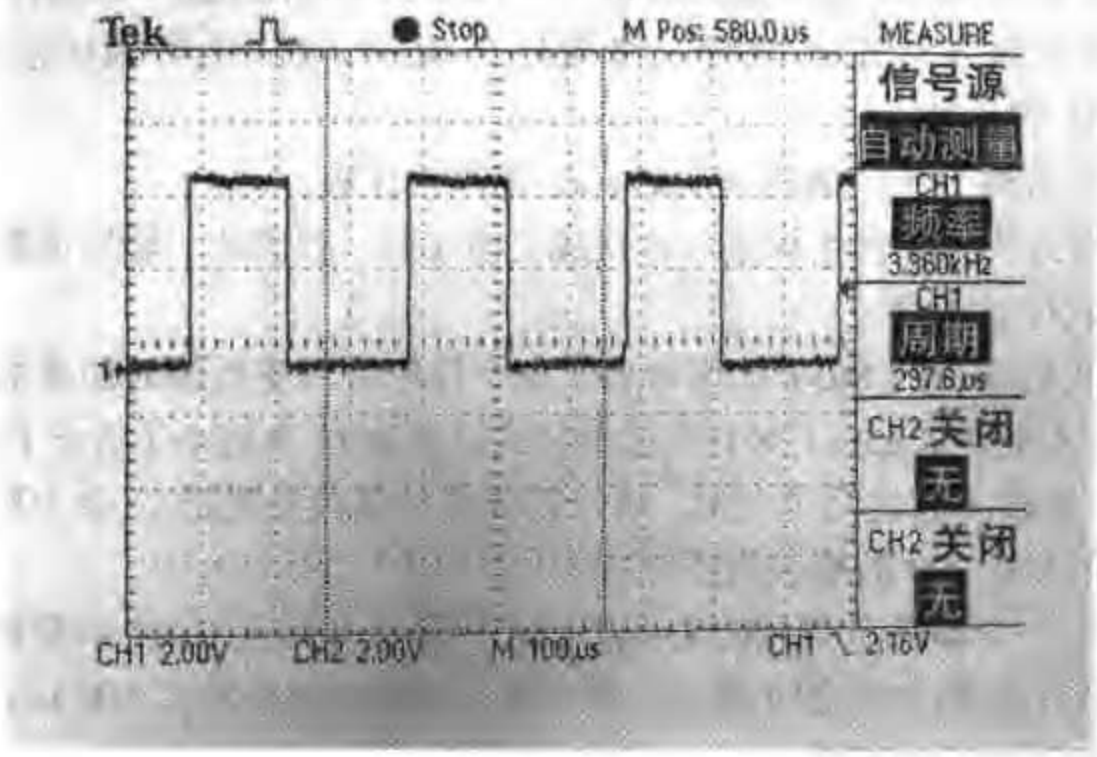


图 16-3

16-3　温度的显示

请看图 16-2，这是温度显示的线路图。线路上共有四组显示数字，其中每两个数字排成一排，上一组数字代表温度值，下一组数字可代表其它的温度值，其实温度值还可以加上小数点以下一位数，不过在这里似乎不必这么画蛇添足了。为了简化线路我们采用显示字形扫描的方式，仅用 P1 端口就完成四个位数的数字显示。在 P1 端口上分成两大部分，分别送出要显示的 BCD 码以及待显示的位数，当程序要做温度显示时，要先将 BCD 码送出，然后指定位数即可，总共送出 4 次即可完成所有数字的显示，采用这种方法显示时，显示程序一定要安排在定时中断服务程序内持续执行，否则会使数字显示突然中断。

温度的显示不就是把计算出来的温度值直接送到显示器上吗？这种说法表面上是对的，实际上照着做却不一定是对的。依照我们的计算，每次测量 SMARTEC 感测 IC 的 Duty Cycle 到计算出温度所需的时间不到二十分之一秒，如果这样立刻将此值送到显示屏上，可能会使显示屏上的数字变化速率过快，这也会使得数字看起来一直闪烁着，反而不是很妥当的做法。

我们较常用的处理方式是连续读入数十个甚至上百个 SMARTEC 所送出的方波，并且累计其 t1 与 t2 的计数值，然后计算出整体的 Duty Cycle 值，最后才计算温度并显示其值。这种做法会使温度显示值的更新率降到一秒两到三次，可是对室内温度的显示或监控，这种速度已经绰绰有余了，同时得到的值也是这数十个温度值的平均值，这也有一点噪声过滤消除的功能，假设在室内测量室温时，如果测到 99 组温度都是 28 度，但是有一个值是 35 度时，经过平均之后得到的温度值为（(28×99）+35）/100=28.07 度，这对我们温度的显示值并没有多大的影响，因为室内大体积的温度并不可能突然温升 7 度，然后又陡降回原来的温度值，所以我们这种平均法是可以被接受的，在正式的温度计中，我们还采用了另一种温度噪声消除的写法，可以使显示的温度值更加稳定及正确。

16-4　联机功能的加入

温度值能够正常稳定地显示，我们所做的温度计只和一般的温度计相同而已。由于在实验的线路中还包含有 RS485 的线路，所以我们再把温度值转换成 ASCII 码的字符串，只要外部的计算机或设备有需要时，可随时向此温度计要求 ASCII 方式的测量值，以达到联机监控的目的。因此，我们的温度控制程序中又加入一段串行中断的服务程序，当外界有此需求且 RS485 界面传来的 ID 码正确时，该台温度计就会送出 TEMP=XXC 的字符串，这些值再经过处理即可当成控制上的依据了。

RS485 接口还有另一个好处是可以多台并联监控，所以我们可以用一台个人计算机 PC 去监视并控制一个系统中的多个温度点。对于 RS485 的相关知识及用法，我们将在稍后的章节中再做说明。如果您是用功的读者，想进一步得到本程序的内容时，请参考文章最后的程序。如果对这些感温组件的使用有疑问时，也可以通过 E-mail：chipware@seed.net.tw 与我们联络。

16-5 本章使用软件

本章使用软件为：8051Assembler 汇编语言编译器。

16-6 本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。
（2）AT89CXX 刻录板。
（3）AT89C20512KB 闪存控制器。
（4）SMT160 温度感测 IC。
（5）AD590 温度感测 IC。
（6）DS1620 温度感测 IC。

16-7 相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw:查询单片机控制板相关信息。
http://www.atmel.com:查询 AT89C2051 相关信息。
http://www.analog.com:AnalogDevice 网站查询 AD590 信息。
http://www.dalsemi.com:查询 DS1620 信息。
http://www.smartec.nl:查询 SMT160 温度 IC 信息。
http://www.national.com:查询 LM35 温度 IC 信息。

16-8 TEMPONLY.ASM 程序说明

用汇编语言写控制程序是最具有挑战性的，接下来的程序就是读取 SMARTEC 温度的范例程序，但不包括温度显示的部分。TEMPONLY.ASM 一直读取温度值，经过公式转换后存在缓冲区内，当串行端口传来读取温度的 ID 时，程序就会把已经换算成 ASCII 码的温度值传出。跟典型的 8051 单片机程序一样，本程序分别启动了定时中断与串行中断，如果你对这方面不是很熟悉的话，请务必回头看看 8051 单片机较基础性的文章与说明。

程序执行时，系统每 10ms 做一次定时中断，中断的同时也顺检查 RI 位,看看是否有串行信息进来，若有的话则立即读入，并且做相对的处理，这种处理方式对串行数据不多的场合是很适用的。

TEMPONLY.ASM 程序请查看书附光盘中的 CH16_TEMPONLY.ASM 文件。

第 17 章

用 AT89C2051 做一台温湿度显示计

温度测量的方法很多种，精确度也可以到达很高。但是，湿度的度量却不是我们想象中那么容易，这是什么原因呢？以下就是我们的说明。

在南方每年的四五月都是阴雨绵绵的天气，这种高湿度的天气会让人觉得处处不对劲。有人说：开一下空调或除湿机不就得了！可是其中湿度的变化程度您可知道？每天的天气除了温度外，就属湿度的影响最大了，可是到底此时的正确相对湿度值为多少却没人知道！

17–1　湿度的定义以及常见的湿度计

老实说：湿度的测量是相当难的，物理上对相对湿度（Relative Humidity 简称 RH）的定义是：此时在空气中水分子的含量与相同温度下饱和（能够容许最多的）水分子含量的比率。所以相对湿度的单位是%。在某个特定气压及温度下，相对湿度可以由绝对的干燥（RH=0%）改变到露点 RH=100%，当露点到达时，几乎所有的水蒸汽将凝结成水，这也就是说在下雨的天气下，室外的相对湿度都几乎是 95%以上。

最早期的湿度计是所谓的干湿球湿度计，它由两支水银温度计组成，一支显示温度，另一支指示相对的湿度值。不过，测量湿度时要用一块砂布包裹住水银温度计的感温部位，再由对照表显示其相对湿度值来，湿度大时两支温度计的温度差较小，反之则增大。干湿球湿度计使用时必须确定砂布是完全潮湿的，否则得到的湿度值就不正确了。

第二种湿度计是毛发式湿度计，它是靠湿度对毛发间的关系所做成的度量仪器。当相对湿度高时毛发就会伸长，反之则缩短。这类湿度计的误差度是相当高的，所以显示值只能供参考用而已。一般买冷气机附赠的温湿度显示表或防潮箱上的湿度指针都属于此类，其指示值准确度不高。

第三种湿度计就比较有参考性了，它是利用高分子多孔性材料当成感测组件，任何时间内都会有或多或少的水分子附着在此组件上，因而使此感测组件上的电阻值改变。我们只要测量感测组件的电阻特性，即可推导出其对应的相对湿度来，其能测量相对湿度的范围约在 10%～90%间。由于湿度与电阻间的变化量并非线性的（这点关系湿度计的总体误差），而且其电阻值又会受到周遭温度的影响，所以这类的传感器都会列出好几个图表来，供软硬件设

计工程师来参考修正。我们先前提到这类的湿度感测组件易受温度的影响，所以电路设计时，应该对温度再做补偿，否则会有较大的湿度误差出现。

图 17-1 所示的 4 个组件皆为湿度传感器，由左到右分别是：电阻式，SMARTEC 的湿度感测组件 RH05（正面），RH05（反面）与最新且面积最小的感湿组件 RH10（附注：SMARTEC 的 RH05 已经停产）。

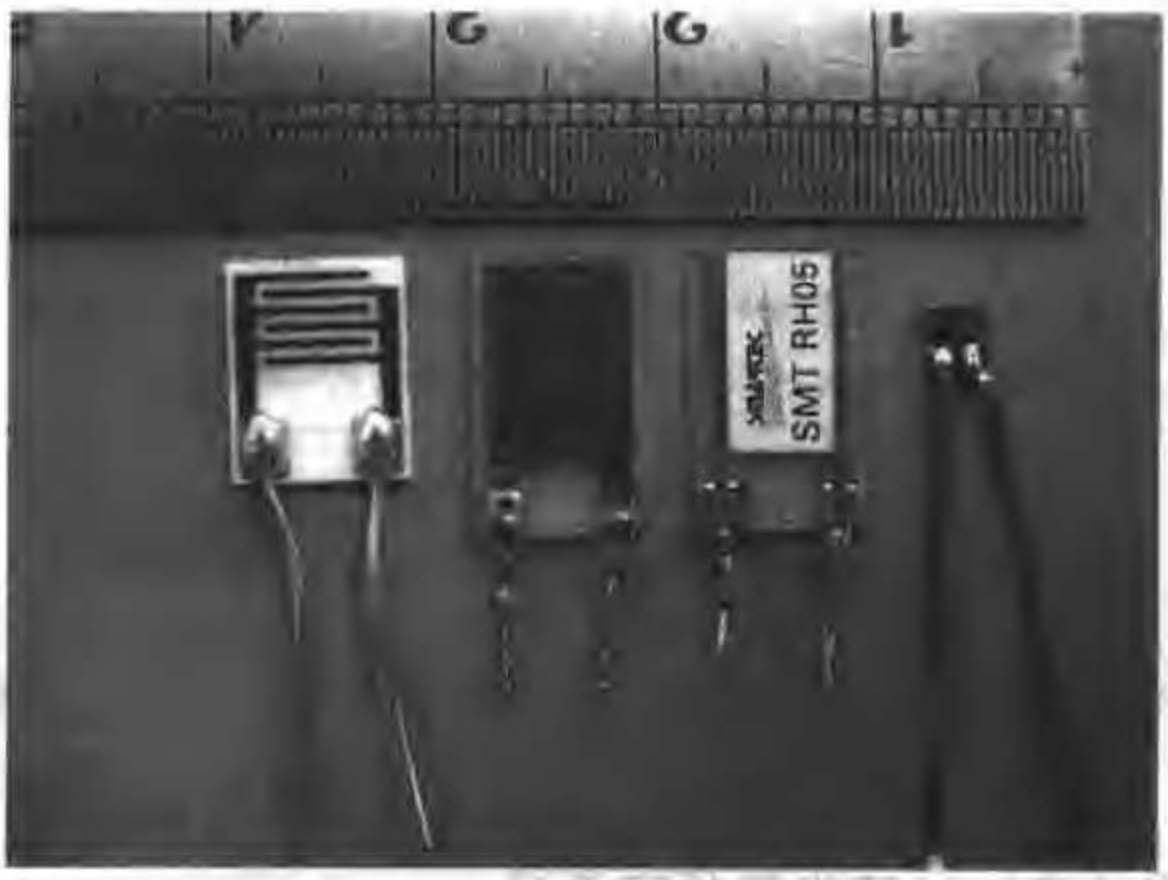


图 17-1

第四种湿度计为第三种感测组件的改良，由空气中的水分子含量来改变感测组件的电容量值，这类的感湿组件许多工业先进的国家都可以制造。我们先后比较了几种进口的感湿组件，最后挑选了由荷兰 SMARTEC 厂所开发的感湿组件，主要考虑因素正是其与湿度的变化是成正比的，这一点可以使我们在做湿度电路时，可以免除许多线性化处理的线路。由于半导体厚膜技术的导入，这个感湿器是蒸镀（coating）在一小片玻璃纤维板上，长宽为 7×13mm，厚度仅 0.2mm，原厂提供的数据显示其可以度量的湿度范围可由 0～100%，而且湿度的变化与其电容变化量是保持线性的，原厂的规格上注明其误差在 2%以内，这个特点是其他感湿组件所无法做到的。资料上说：当湿度为 0%时，感测组件的电容值为 300PF，而湿度到达 100%时其电容值升到 370PF，这也就是说：相对湿度每改变 1%时，电容量增减 0.7PF。原厂的数据中还特别提到该感测组件对温度的影响几乎是微乎其微的（negligible），所以我们就毫不迟疑地向台湾的代理商买了十个样品进行各项特性的测试，虽然原厂供应的数据只有 4 页，但是结果比我们预期的还要满意。

17-2 原厂线路说明

图 17-2 是原厂保证能够动作的湿度线路，图 17-3 是我们简化后的方块图，它是先用第 1 个 555 振荡出一个固定的几百 K 的频率，然后将方波输出送到第 2 个 555 上，第 2 个 555 则是做单击输出（one-shot），其输出时间为 1.1RC（秒），R 是固定的 2K7 欧姆，

而 C 则是我们的 SMARTEC 感湿组件。第 2 个 555 的输出方波信号中就隐含着湿度值的量了，再经过一个简单的滤波成为直流成份后，就转送到运算放大器 LM358 做缓冲输出。另一个 LM358 则是做湿度值的设定，只要一超过此设定湿度值时，LM358 会正饱和输出以便启动继电器，如果您想做单纯除湿机的话，这个电路就已经足够了，湿度定点值由图中的 VR 来调整。可是若是要把湿度值用数字方式显示出来时，这个电路就显得有点不足了。

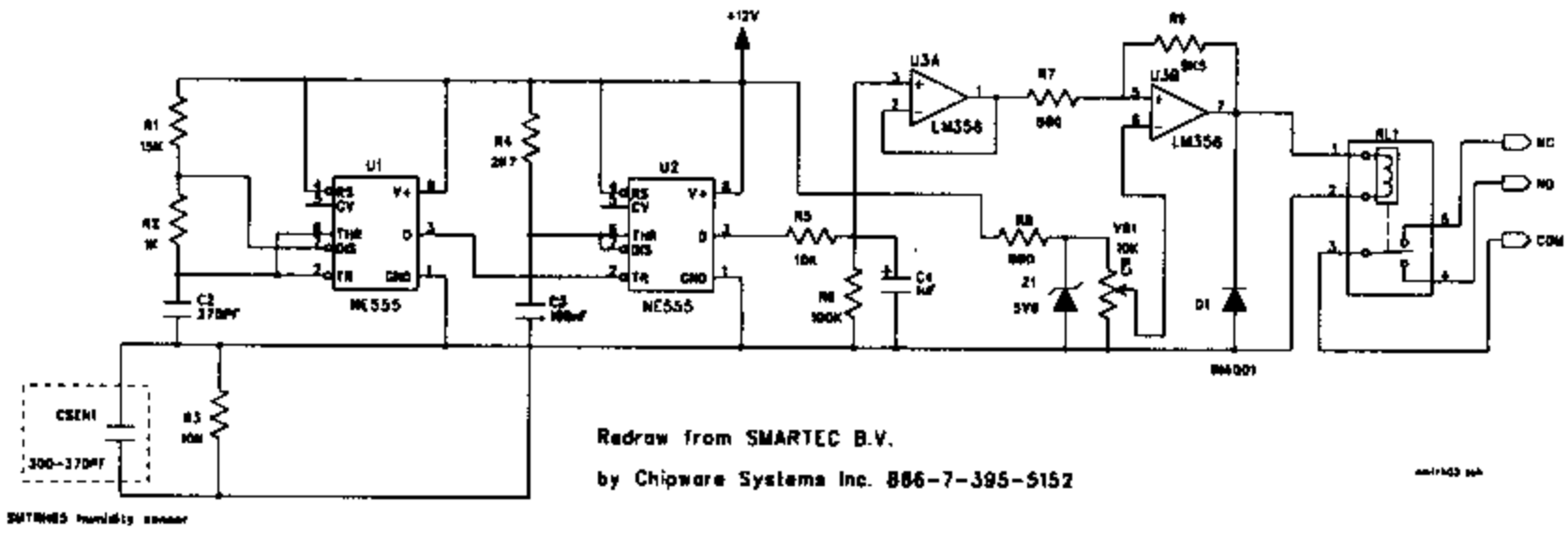


图 17-2

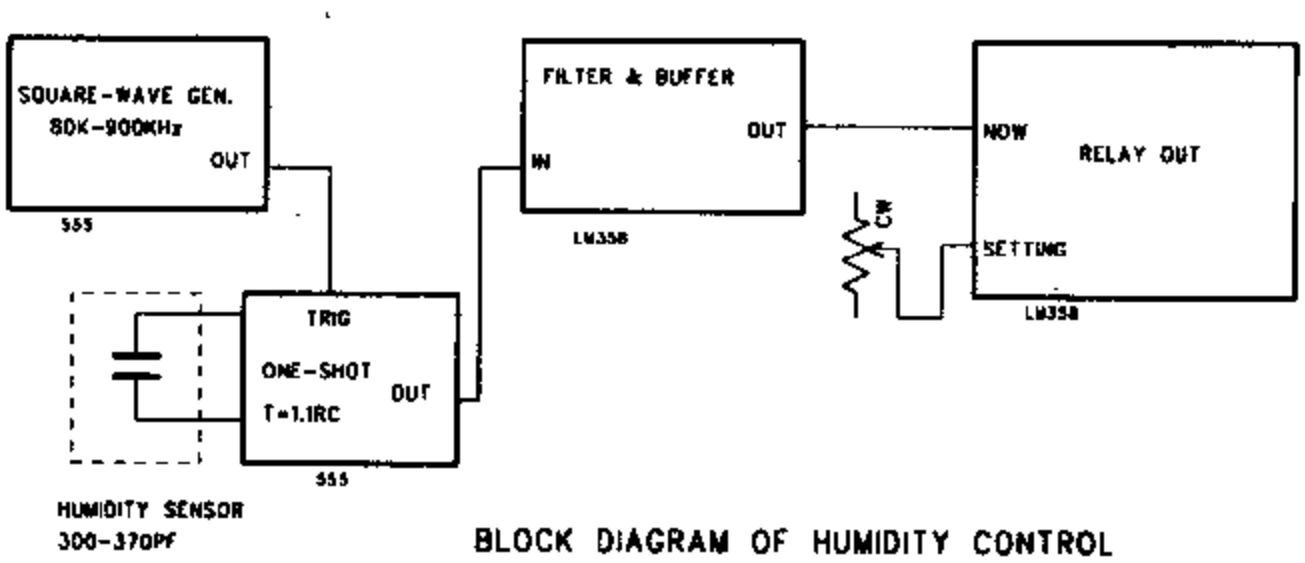


图 17-3

如果把图 17-2 含湿度的直流电压值转送到一个 AD 转换电路上，应该就可以得到数字式的湿度值了，这个值可以做数字的显示，也可以累积储存以方便做统计或是研究用。可是这种做法会使湿度转换的电路变成非常非常复杂，当然最后的价格就相当高了。我们试做的第一个湿度 prototype 就是这样被规划出来，可是对于这些结果我们还是不满意的。

上述线路的重点在于把湿度的方波信号转换成直流信号，这使得我们的线路还要多一道手续将信号再度转回数字的方式，这样一来一往之后，不仅使线路复杂化之外，还会使得测量的湿度值丧失其该有的精度。所以，我们经过审慎的评估及试验后，决定把线路做了大幅度的精简，将湿度侦测线路变成一个相当单纯的 555 单击电路。请看图 17-4，你一定会讶异！这样做行得通吗？经过我们数个月的密集试验，结果是行得通的。

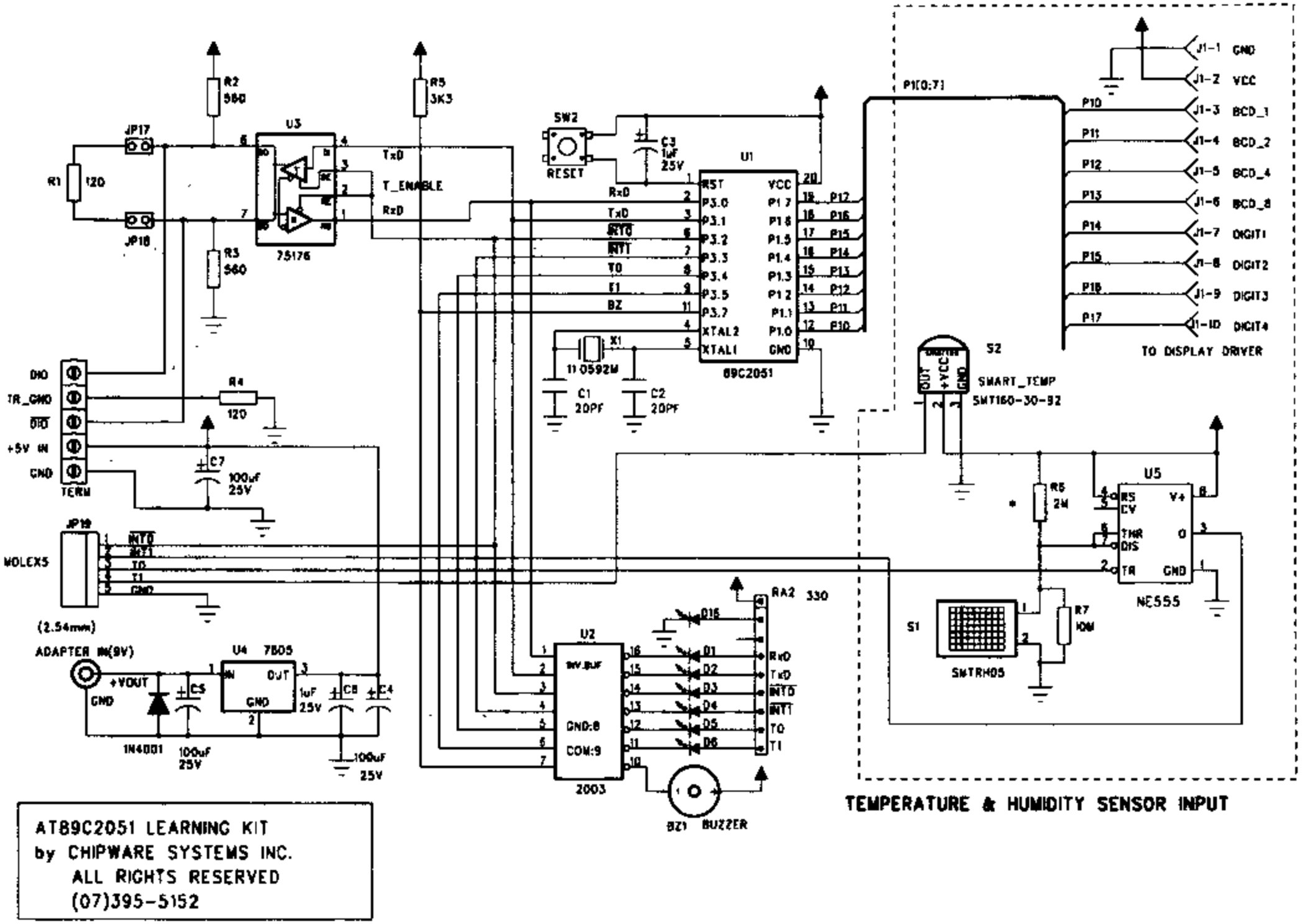


图 17-4

17-3　湿度计脱胎换骨的新设计

温湿度计若想数字化的话，所有的设计理应从头开始就进行数字化的处理，原厂提供的示范电路中是由第 1 个 555 产生方波，而要单片机 8051 产生方波是轻而易举的，所以我们就把这个 555 给省下来了，只保留做 one-shot 的 555 电路了。另外我们也不要忘了：用单片机来计算 one-shot 时间也不是难事，因此也同时把原线路滤波电路以后的线路全部删掉，所以，湿度检测电路精简到只剩下 3 个零件：555 计时 IC、电阻以及电容（即 SMARTEC 的感湿组件）。我们的构想是这样的：由 AT89C2051 产生一个触发信号给 555 的第 2 脚触发输入点，由于 555 的线路是被规划成 one-shot 的动作，所以我们用 AT89C2051 另一个输入点来计算 one-shot 的时间（T=1.1RC 秒），时间的秒值得到了，而电阻值也是已知的，就可以换算出对应的电容值了。所换算出的电容值应当在 300PF 到 370PF 之间，最后我们再用内差法计算出实际的相对湿度值来，并且做两位数的数字显示测量到的湿度值。

可是这个电路还有一个电阻值尚未决定。由于 SMARTEC 感湿器的电容量不是很大，仅有 300～370pF 而已，湿度由 0%升到 100%时，电容的变化量只有 70pF，这个变化量是相当小的，所以我们在选择 R 的阻值时就须尽量大一点，以便让 one-shot 的时间可以到达接近毫秒（ms）的等级，方便让 AT89C2051 计时。如果 one-shot 时间过短时，将无法得到相对湿度 1%左右的精确度。在我们的湿度检测实验线路板中，第一次试制时所选用的电阻值是 2.00MΩ（1/4W，精度 1%以内），所以 555 的 one-shot 电路产生的时间约是 660μs-814μs 左右。图 17-5 是我们试做的温湿度显示器 prototype，由于是试制品，所以我们用了两块线路板，一是 AT89C2051 学习板，上有单片机 CPU 及 15 个 IO 点，我们用 P3 口空余的接脚去读取温湿度的方波信号，P1 端口则推动 4 个七段显示器。另一块 PC 板是万用的线路板，其中包含了两个荷兰 SMARTEC 的高信赖度传感器，还有两组各两位数的数字显示，分别显示温湿度值。

图 17-5

17-4 湿度程序的规划

湿度测量程序如果依照我们先前的说法来规划，说实在的并不是很困难的，通常我们会用纸与笔先行将整个计算过程模拟一次，如果看起来完全没有大问题时，再去规划细部程（不论用汇编语言或是高阶的 C 语言都是如此），因为实际地彻底运算一次后，所有的毛病及问题都会一一浮现出来，接下来的所有程序才会有相当的模块化概念，而且也很容易除错及验证。以下就是我们初步模拟的假定条件以及湿度测量步骤的详尽说明：

内定条件 1：AT89C2051 的工作频率是 11.0592MHz，以方便做 RS485 的联机。

内定条件 2：不使用 8051 内部的定时器，因为串行传输及扫描显示用的 10 ms 定时中断已分别占用了 Timer0 及 Timer1。

内定条件 3：单片机计时循环所要花的最短时间为 3.255μs，相当于执行三个基本指令所花的时间，8051 虽然执行速度不慢，但是碰到这类高速计算时，还是会觉得不够快。

内定条件 4：one-shot 的电阻值为 2MΩ，并且暂不考虑其误差。

内定条件 5：感湿组件的输出是完全线性的，亦即我们拿到的值不须再经过线性化处理，就可以求出湿度值。

◆ 步骤 1：指定一个 16 bit 的计数器，并在定时器前先清除成 0。

◆ 步骤 2：送出一个触发信号给 555，它就会依外加的 RC 值产生一个 one-shot 方波信号，而其持续时间的标准公式为 1.1RC，R 的单位为欧姆，C 的单位为法拉，我们这里是使用 1%误差的 2MΩ 精密电阻。

◆步骤 3：假设现在室内的相对湿度是 50%，则感湿组件所呈现的电容值应该在（370PF-300PF）/2，即 335pF 左右。

◆步骤 4：所以 555 的 one-shot 时间输出为

1.1x（2.0E+06）x（335.0E-12）=0.737E-3（sec）=0.737ms

◆ 步骤 5：单片机 AT89C2051 用一个输入点去检测 one-shot 信号是否已由 1 降成 0，先前我们已假设每次循环检查所花的时间为 3.255μs，所以软件计数器所算到的次数是 0.737ms/3.255 μs=226 次。

◆ 步骤 6：若相对湿度是 0%时，感湿组件的电容值为 300pF，换算出 one-shot 的时间为 0.660ms，若再换算成循环计数值为 0.660 ms/3.255 μs=203 次。

◆ 步骤 7：如果相对湿度是 100%时，感湿组件的电容值为 370pF，换算出 one-shot 的时间为 0.814ms，转换算成循环计数值为 0.814 ms/3.255μs=250 次，湿度从 0%到 100%总共有（250-203）=47 次的变化量，这也就是说每增加一次则湿度增加 100%/47=2.12%。

◆ 步骤 8：由于我们度量到的循环次数是 226 次，所以在推导出 RH 相对湿度时要先把此值减去相对湿度为 0%的 203 次，有效值成为 23 次。

◆ 步骤 9：把 23 次再乘上 212 得 4899，只取千位及百位数 48，若十位数是 5 到 9 时做四舍五入的动作，所以再把 48 加 1 得到最后的湿度值 49%。

◆ 步骤 10：计算的湿度值 49%送到输出缓冲区上，并且化成十进制值准备显示用。

我们这里所做的示范是针对 one-shot 上的电阻为 2MΩ 时，所推导出的相对湿度值。我们也发觉：每增加一个循环次数时，湿度显示的增量约是 2.12%，如果暂时忽略小数点不看时，显示的湿度分辨率为 2%，虽然这对湿度的表示已经足够了。产生这些误差的原因有：

电阻不是刚好我们设定的欧姆值，计算式的误差以及感湿组件的误差等等，如果依照原厂公布的规格来看，其感湿组件的整体线性误差可以保持在 2%以内，我们是否有可能提高显示分辨率到每次 1%吗？答案当然是可行的，但是对线路做适度的修正是免不了的，首先当然是修改 555 旁的电阻值，尽量让电阻值刚好等于 4.21MΩ，数值 4.221MΩ 恰巧是每次循环检查时间 3.255 μs 的 1.297 倍，为何选择此一幸运的阻值，请看接下来进一步的说明。

当 R=4.221M 欧姆时，相对湿度由 0%变化到 100%所产生的 one- shot 时间为 1.39293 ms～1.717947ms，再次换算成循环的次数时，分别是 428 次～528 次。湿度 100%的变化刚好等于 100 次的变化量，如果相对湿度刚好是 50%时，AT89C2051 测量到的循环次数应该在 478 次左右，程序中只须把 478 减去 428 得 50，这个值不须进行任何转换就直接是相对湿度值了。我们规划任何测量的程序一定要做到这里才算是告一段落，否则完全相同的电路却会导致完全不同的精度了，想成为单片机相关线路设计 PRO 级的读者不可不慎！

在我们的线路图中，one-shot 上的电阻只有一个，但是实际电路中我们是用两个精密电阻及一个精密的 20 圈 100KΩ 可调电阻所组成的，以便合成我们所要的 4.221MΩ 的电阻（更精确的值是 4221636Ω）。所有的计算式都模拟并计算过后，就可以开始写细部程序了，这些编码动作（coding）反而变成相当的单纯。如果想把显示的分辨率提高到小数点以下一位，是不是要把电阻值增加 10 倍呢？理论上是可以的，不过在 555 的特性资料中，不建议我们使用过大的电阻值，以免有过大的 one-shot 输出误差。所以，实际上我们是分别触发 555 十次得到十个测量值，把这些值累加起来后，结果应该在 0 与 999 之间，累计值的最后一位数即个位数值正是我们所要的值了，从这些运算方法中，读者可以发现在单片机的程序中，所有的计算都是整数的加减乘除计算，完全不包含浮点的运算，唯有如此才会有最快的数据处理速度。

17-5　温湿度系统程序的发展

温湿度程序的发展可以分成以下几大部分：

（1）温度 sensor 其 Duty Cycle 的读取及温度值转换，详细的转换方法与公式请参考上一章的说明。

（2）湿度的 one-shot 时间计算，并转换成相对的湿度值，单位为%。如果选对电阻值会让程序的写法变成更为单纯。

（3）温湿度值的显示，在 prototype 中，温湿度都只有两位数字而已，在实际的应用中，温度的显示似乎可以到小数点以下一位数，而湿度的表示若太多位数反而觉得“画蛇添足”。由于我们的显示线路是采取扫描的方式，所以这段程序一定要在 10 ms 的定时中断内执行，否则在测量温湿度期间都会造成显示屏的暂时罢工。同时还要注意的是数值显示程序所用到的定时中断，其优先性应该比温湿度的测量程序还要高些。

（4）RS485 联机程序部分，这也是这台温湿度计的重点之一，只要有外部的计算机要求温湿度值时，温湿度计会把最近测量到的值转成 ASCII 码后，经由 RS485 界面送出。由于可能有多台温湿度计并联在 RS485 的界面上，所以每台温湿度计都要有各自的 ID 值，以免造成送出数据的碰撞。这段串行通信的程序也要放在串行通讯的中断服务程序上，而且要有最高的中断优先性。

（5）温湿度计的故障指示程序，所有的电路都是有其寿命的，在这个电路中以两个感温及感湿的组件最为重要，所以程序在显示之前，要先检查两个传感器是否都正常工作着，温度传感器只要确认其有状态高低的变化即可。至于湿度传感器正常与否的测试方法，则是先送出一个触发信号，然后确认 one-shot ON 的时间是否在 3ms 以内，若时间超过时一定是感湿器发生故障或是传感器开路。

以上这些读取与显示的动作摆得进一个 AT89C2051 的 2KB 程序空间吗？答案是可以的，我们用汇编语言完成了所有温湿度的动作，有说明的原始程序（Source Program）约占 25KB，而最后的二进制执行文件则仅有 1.5KB 而已。

如果是用 C 语言来处理这些复杂的动作时，很有可能无法很顺利地摆进 AT89C2051 内，我们并非否定 C 语言无法处理温湿度的测量，而是说：如果系统要同时兼顾温湿度，还有显示以及串行等中断处理等等，用汇编语言来做还是比较恰当的。

最近几年来 SMARTEC 的湿度传感器也做了些许的修正，RH05 已经停产，取而代之的是 RH10，其第一项修正是感测面积缩小了，第二项修正是其电容值也改小了，RH=0%时电容值仅有 210PF，RH=100%时，电容值升到 240PF，仅有 30PF 的变化量，如果你买到的是新的 RH10，先前所提到的计算式统统要重新推导一次才行。只要我们掌握住所有的设计原则，这些都会有解答的。

图 17-6 所示为 RH10 的面积缩小了许多。

图 17-6

17-6 组装及温湿度的校验

我们在这里示范的数字式温湿度计，除了两个温湿度传感器较不易购得外，其余的零件都是很容易购买的。

在使用上我们应该注意：湿度传感器的感测面要尽量避免手的触摸以及其他液体接触，否则会使测量误差加大。另外我们会设计出一套完整的温湿度控制器，供工业控制或温室的温湿度控制用。如果您照着电路自己焊接温湿度计线路时，温度上的校验应该比较简单，只

要找来一台数字式的温度计进行比对，即可得知 SMARTEC 公布的误差值是否正确，当然温度转换程序在运算时也会有一些误差出现，这个部分只要修正软件程序就可以了。

至于湿度的校验就要多费点心力了。首先我们要找到或是自行制造一个透明且可密闭的塑料容器，其体积至少可以摆进我们的控制及显示电路板（上面还包含温湿度传感器），然后在容器底部加入饱和的食盐水，形成一个所谓的盐浴（Salt Bath），先让此密闭容器在室温下摆上一到两天，然后放入温湿度控制板，而且立即封闭容器并再度摆上至少半个小时，以便让容器内的水蒸汽循环达到稳定的状态，此时显示的湿度值应该在 75%才对，否则就是我们整个运算过程中有所偏差，必须重新考虑程序的运算以及 one-shot 电路上电阻电容值的误差等等。这就是定点湿度（Humidity Fixed Point）的标准校验方法，我们还可以用其他化学盐类检测其他的湿度定点。我们常提醒读者要多方面吸收知识，就像这里的湿度校验，如果没有多吸收点化学常识是不行的。

由于这台温湿度计本身已经有串行通信的能力，所以我们可以事先把与 PC 个人计算机通讯的接线接妥，一进入上述的校验容器后，就直接读取湿度的值，读取的速度约在一秒一次就可以了，由此也可以看到感湿组件的反应速率如何，我们也可以用计算机做长时间的观察，监视某个特定区域内湿度的变化情形。在某些精密仪器上是绝对不允许湿度过高或者是水蒸汽凝结成水滴的露点（Dew Point）出现，藉用这台温湿度计的串行通信界面，就可做类似的湿度应用及控制。写到这里我们突然想到家家都有的录像机上磁头旁也有一个湿度传感器，当湿度过高时，录像机会自动对磁鼓的金属部分加热，以便将水气蒸发掉，如果不这样做时，在潮湿的环境下启动磁鼓的话，一定会将录放影的精密磁头打坏。所以如果您家处在湿度甚高的地区时，千万不要随便切断录像机的电源，否则一按下 PLAY 键卷动录像带后，就会不幸地发现录像机真的出问题了。

17-7　本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler 汇编语言编译器。

（2）IAR 8051 C Compiler。

17-8　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）AT89CXX 刻录板。

（3）AT89C2051 学习板配合外加显示电路。

（4）AT89C2051 2KB 闪存微控制器。

（5）SMT160 温度感测 IC。

（6）RH10/RH05 湿度传感器。

（7）干湿球温度计。

（8）毛发湿度计。

17-9 相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C2051 相关信息。

http://www.smartec.nl：查询 SMT160 温度 IC 及 RH05 湿度传感器信息。

http://www.national.com：查询 LM35 温度 IC 信息。

17-10 湿度测量程序说明

用汇编语言写湿度的测量程序理论上是比温度测量程序要难一点，但是都避免不了公式的转换与计算。正确的湿度值还需要套用内插法求出值来，许多在初中与高中学到的数学公式在这里是相当管用的，忘记了没关系再温习一次就会重新烙印在大脑中。

湿度测量程序首先触发 IC 555，让湿度传感器回送一个单击信号回来，此时用程序去计算该脉波的时间，然后与 RH=0%到 100%的时间比较，再用内插法求出现在的湿度值。由于湿度感测组件会更改部分规格，所以我们特地把湿度的某些设定参数存放在 EEPROM93C46 上，以便每次开机时读取。在下面的程序范例中若有 93C46 的字样时，就是在做这方面设定值的存入或读回。

TEMP2052.ASM 程序请查看书附光盘中的 CH17_TEMP2052.ASM 文件。

THM.ASM 程序请查看书附光盘中的 CH17_thm.ASM 文件。

第 18 章

智能型温湿度计 TH2030 的制作

前几章我们分别谈了温度与湿度的测量方法，当时是以 AT89C2051 为主体的，在本章里我们将这些软硬件做个整合，规划一个完整的温湿度控制器 TH2030。如果你想追踪或记录一段时间内的温湿度起伏变化，就请跟我们一起 DIY 吧。

如果您想知道所处环境的温湿度情形，最廉价的方法是买一台 LCD 式的温湿度计，就可以随时得知温湿度值了。可是这类的温湿度计只能显示实时的值，并没有记录及储存的功能，所以只能在很干冷或很燥热的情况下，才看一下温湿度计上的值。曾经在电视上看到一个空调的广告，使用者一直抱怨空调的运转是走走停停的，导致房间内的温度也跟着起伏不定，身体觉得非常不舒服。如果您也有这方面的需求与应用时，势必要购买一台具有记录或联机功能的温湿度计，才能得知温湿度与时间的变化关系了。这里我们就要制作一台兼有记录与判断的智能型温湿度计，让我们可以随时读取温湿度值，并且也可以做实时的监视与控制。

以下就是这台智能型温湿度计 TH2030 的硬件规格及主要特性。

◆ 硬件主要规格：

（1）CPU：ATMEL AT89C52 8KB 闪存微控制器。
（2）外部内存：32K SRAM 并加 3.6V 的 Ni-Cd 充电电池。
（3）SMARTEC 的温湿度感测组件。
（4）温度及湿度显示各占两位数。
（5）三个 LED 充当 ALARM 警告指示。
（6）一组蜂鸣器及四组开放集极（Open Collector）输出。
（7）1K bit EEPROM 校正数据保存。
（8）三个按键开关及 4P DIP 指拨开关输入。
（9）RS485 两线式串行通信接口。
（10）AC/DC 交直流输入皆可。

◆ 主要特性：

（1）使用内含 8KB 闪存空间的 AT89C52。

（2）SRAM 空间为 32KB，且有充电电池做备用电源，可以防止数据的漏失。

（3）使用 SMARTEC 的温湿度感测组件，值的读取非常方便。

（4）所有在线的设置值直接存入板上的 EEPROM 中，校验与调整都非常方便。

（5）RS485 两线式通信，适合在多点的控制与监视用。

（6）内含整流与稳压电路，7-12V 的交直流电压输入皆可。

（7）有 4 组 ALARM 输出：蜂鸣器输出、一秒定时输出、设置温度与湿度到达报警输出。

（8）提供单机校验或 RS485 联机校验的功能。

（9）可定时记录温湿度值各 16000 点。

（10）提供温湿度值记录“黑盒子”的功能，最近 30 分钟内的温湿度值同时存放在 EEPROM 中，机器故障时，可得知最后温湿度的变化情形。

（11）内含所有零件的检验及测试程序，方便自己装或维修。

（12）应用场合：标准实验室及计算机信息室的温湿度监视，温室的温湿度控制及其他须监控温湿度的环境。

18-1 TH2030 温湿度计线路分析

图 18-1 到图 18-3 都是智能型温湿度计的线路图。这是一个 8052 外加扩充 I/O 的控制系统，系统程序有 8KB，外部的扩充 SRAM 则有 32KB。图 18-1 是系统的主要线路，图 18-2 为 RS485 联机电路以及温湿度的测量电路，图 18-3 则是地址译码及电池充电电路。在前几章我们已经对 SMARTEC 温湿度感测组件做了相当完整的介绍，并对温湿度的线路也做了相当仔细的描述，这里的线路是把这些电路做了妥善的综合考虑，并且加入了温湿度两种数字的显示。线路上特别加了几个工业设备上常见的电路，我们必须要对这些电路稍加说明，以便让您有更深入一层的认识。

◆ 32KB SRAM 与镍镉电池充电电路

如果控制系统中需要保存大量的测试数据时，一定会利用低功率消耗的 SRAM 外加电池做断电时的数据保存用，本线路（请看图 18-3）是一个相当典型而且几乎不会出错的线路，U15 PEEL18CV8 主要的功能是译码输出，SRAM 的使能输出脚为第 19 脚，当外部输出的地址在 0000H-7FFFH 时，此点为正向输出，若电源有加入时，74HC00 的第 3 脚输出负向的使能信号给 SRAM 62256；反之，主电源不见时，74HC00 的第 2 脚几乎是 0V，这会使第 3 脚一直保持在 HI 电位，若 SRAM 的 CS 脚保持高电位且电源端仍有电源供应（>2.0V）时，该 SRAM 会进入数据保护的状态，不论其他脚的状态如何，都不会把数据送出，只要在此保护模式下，SRAM 会将 IC 的耗电量减少到 1mA 以下，所以此时就可以用电池来当备用电源。如果重新有+5V 电源加入时，充电电池则通过图中的电路及限流电阻 220 奥姆充电（10mA 以内），以便尽快充满电力，由于断电时，NAND 闸仍要能工作，所以该 IC 应该要选用低电力消耗型的 74HC00。

◆ EEPROM 93C46 的加入

把 DATA 存在 EEPROM 内部，也可以半永久性保存该数据。为了防止数据遭受不正当

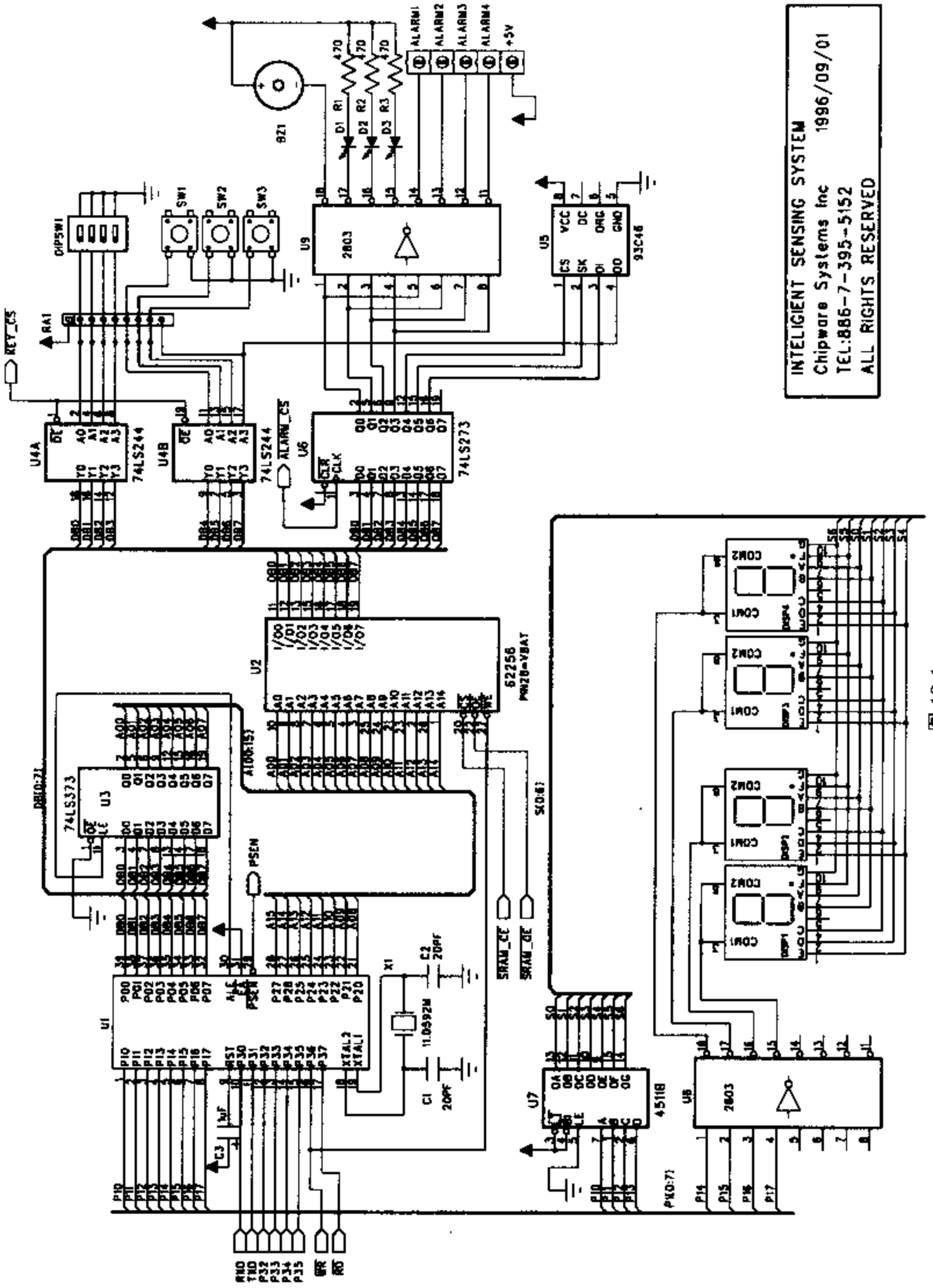
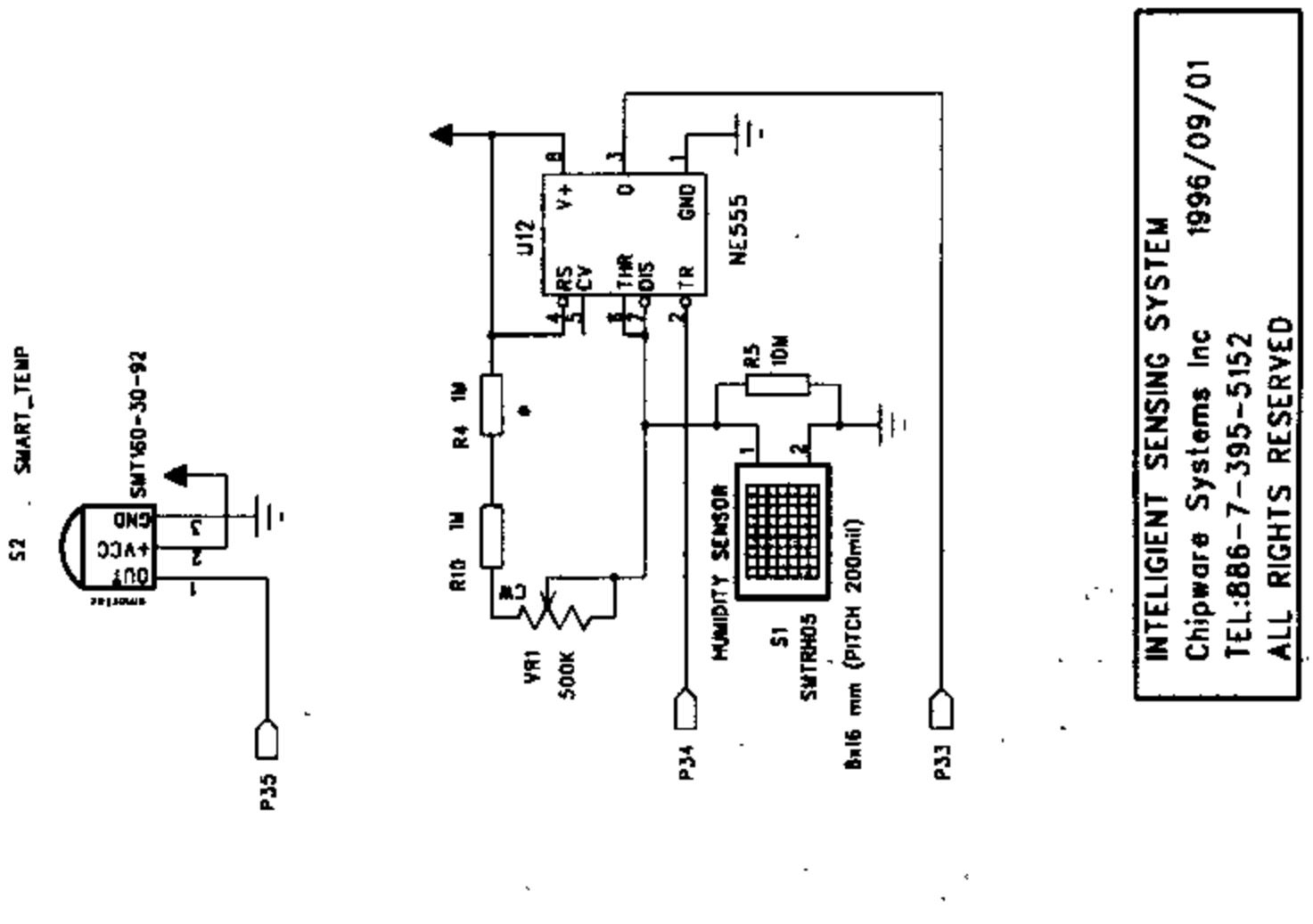

INTELIGIENT SENSING SYSTEM
Chipware Systems Inc 1996/09/01
TEL:886-7-395-5152
ALL RIGHTS RESERVED
ALARM1
ALARM2
ALARM3
ALARM4
+5V
470
470
470
D1 R1
D2 R2
D3 R3
BZ1
DIPSW1
SW1
SW2
SW3
U9
2803
U5
93C46
VCC
DC
ORG
GND
CS
SK
DI
DO
RA1
REV_CS
U4A
U4B
74LS244
ALARM_CS
U6
74LS273
CLR
CLK
U2
62256
PWR28=VBAT
74LS373
U3
PSEN
SRAM_CE
SRAM_OE
U1
X1
11.0592M
C1
20PF
C2
20PF
C3
1uF
U7
4511B
U8
2803
COM2
COM1
RXD
TXD
P32
P33
P34
P35
WR
RD

图 18-1



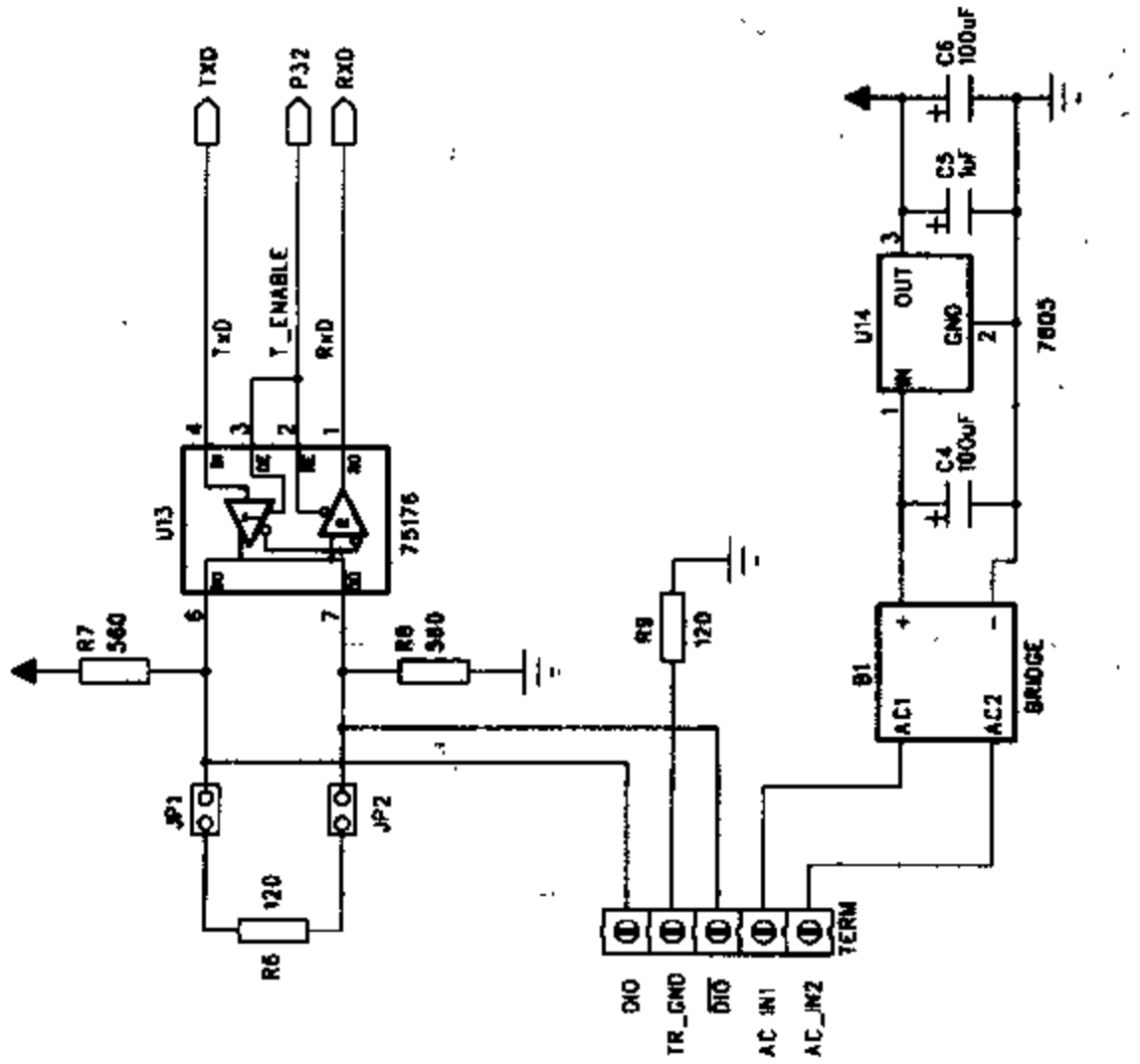


图 18-2

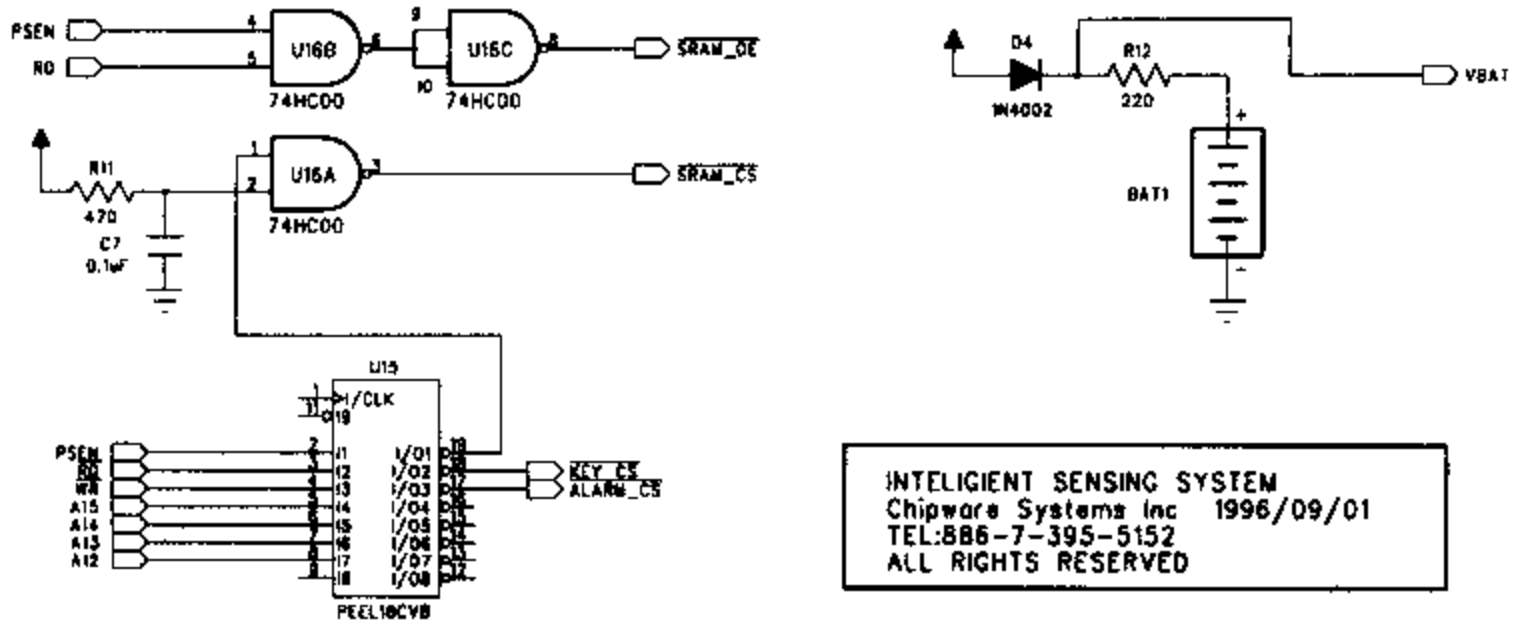


图 18-3

的篡改，我们特地把系统所使用的特别参数值存在 93C46 中，该 EEPROM 为串行型的 EEPROM，内有 1Kbit 的存放空间，这些值只有在温湿度校验阶段才需要调整。除此之外，我们也安排把最近 15 分钟内的温湿度值摆进该 EEPROM 中，如果整个温湿度监测系统遭不明因素而损坏，仍能用其他的万用刻录器读到最后 15 分钟内的温湿度值，这个安排与飞机的飞航记录器（俗称黑盒子）非常类似，由于 93C46 读写过程较为复杂，不可能用几个读写指令或是随便加上电压就更改其中的内容，所以其保密及安全性都较 SRAM 来得高。

◆ 输出 ALARM 接点

智能型温湿度计 TH2030 不纯粹是显示温湿度值而已，所以我们共有 4 点 ALARM 信号的输出点。请看图 18-1 右方的电路，4 个控制点中有一个是供外部的蜂鸣器用（ALARM1），ALARM2 我们打算安排成 1 秒钟的定时输出；ALARM3 及 ALARM4 则是温湿度计到达设置值的输出信号，我们可以用这两组信号去推动外部的继电器，进而启动风扇或空调（或除湿机）以达成实时温湿度监控的目的。这四个输出都是由 2803 送出，这类的蜂鸣器其输出能力都相当大，不过，2803 的输出是开放集极式的，动作时它可以吸入相当大的电流，把输出端保持在几乎是 0V 左右的电位，但是不动作时，它必须外加一个电阻到外部的电源上，方能维持高电位。

◆ 输入按键与 DIP SW 开关

如果要能够单机不借助计算机来设置本控制器的温湿度值，一定要有按键开关的输入才行，我们这里只安排了三个按键，供一般的温湿度设置用。开机时若按下这三个键则是做特定的自我测试用，这些功能我们稍后将会再做详细的说明。至于 4P 的 DIP SW 则是供做 RS485 的 ID 码用。当本控制器与 PC 用 RS485 联机交换信息时，只有收到符合的 ID 码时，才会送出温湿度的 ASCII 值来。

18–2　TH2030 的 DIY 制作步骤

TH2030 温湿度控制板的 DIY 制作也是相当简易的，正式焊接前请先把焊接在正常焊接

面与焊在零件面的零件分隔开。先焊接高度最低的零件，如电阻电容及石英晶体，然后再焊接各个 IC 座，最后才是桥式整流与充电电池。焊完整个零件面后，才焊正面的温湿度传感器与 7 段显示器等零件，然后插上所有的 IC 与 AT89C52 微处理机。

当然在加上 AC 或 DC ADAPTOR 电源前，我们还是要提醒您：确认有方向性的零组件焊接的方向是否正确，所有 IC 的插入方向是否正确，以及零件的焊接点不得有假焊的情形出现。

在图 18-2 的湿度感测电路中，我们在电路中有安排一个 500K 的可变电阻，原本的用意是调整 555 输出的脉波宽度，后来在写程序时发觉用软件的补偿反而较 VR 的调整来得方便，所以便把该 VR 给删掉了，在板上只须将 VR 上的两个有效点短路即可。图 18-5 是我们把温湿度控制板装到订做压克力箱的照片，温湿度传感器前方要做开孔，以测量出外部环境的温湿度值来。

图 18-4 为温湿度计的全貌，PC 板两面都有焊接零件的零件面及焊接面。

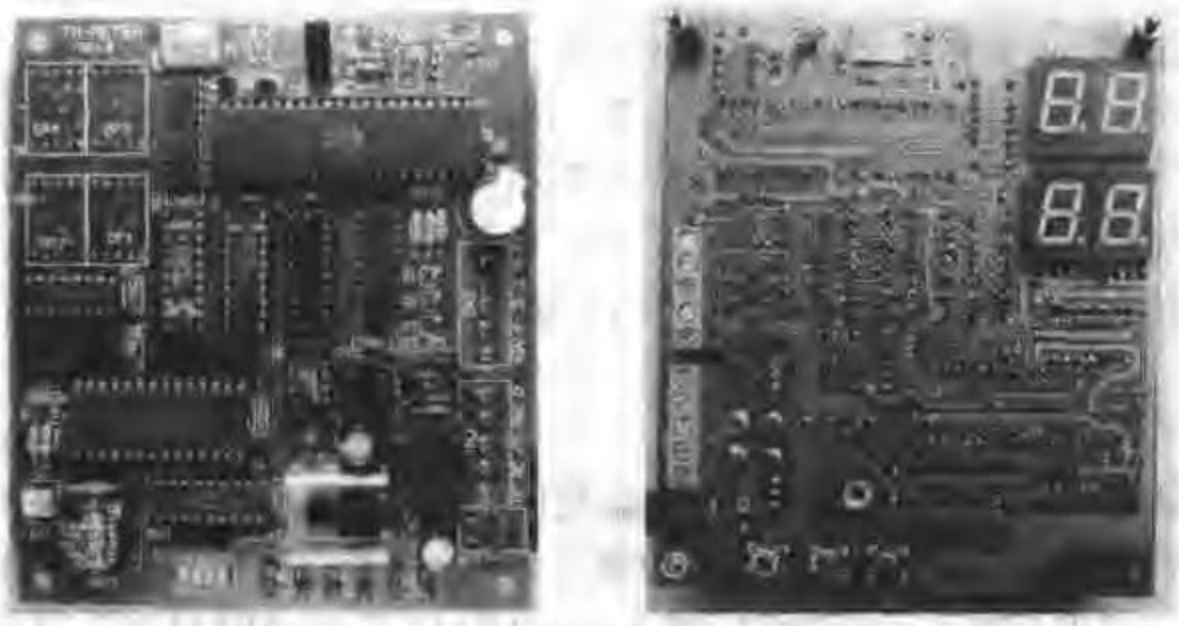


图 18-4

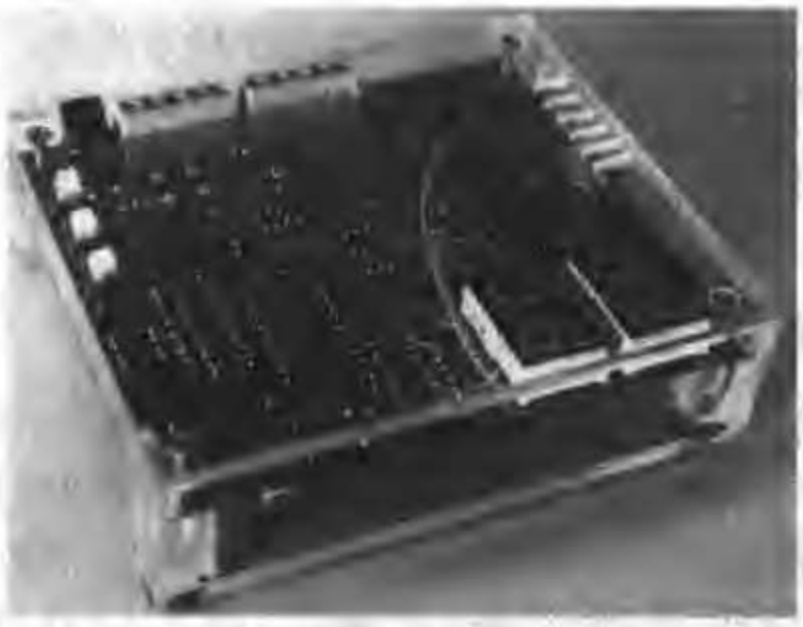


图 18-5

18-3 TH2030 温湿度计的自我测试方法

我们已经在 AT89C52 内部安排了本控制器的所有测试程序，所以只要您接上电源的瞬间，按住控制板上的 TEMP 键，即代表本系统进入自我测试的阶段。可对板上的所有零组件

进行基本的测试，测试的步骤码显示在温度的位置上，当测出硬件有错误时，测试程序会以蜂鸣器的 BEEP 声提醒操作者注意。

以下就是这些测试步骤的说明：

◆ 步骤 1：进行 7 段显示器的所有比划测试，温湿度的显示数字由 11 变化到 99，请操作者自行观看显示的比划是否正确。

◆ 步骤 2：进行 SMARTEC 温度传感器的信号检测，若没有持续的 HI-LOW 变化时，即认定温度传感器有故障，并发出 BEEP 警告声响。

◆ 步骤 3：进行 SMARTEC 湿度传感器的信号检测，若触发 555 产生 ONE- SHOT 后，没有一个有效脉冲出现时，即认定湿度传感器有故障，并发出 BEEP 警告声响。

◆ 步骤 4：进行 AT89C52 内部 8KB 程序代码的 CHECKSUM（校验和）核对，若内容被修改或调整时，则发出 BEEP 警告声响。

◆ 步骤 5：进行外部 32K SRAM 的读写功能测试，SRAM 内部的值将被改成测试值，若读写有问题时，也会发出 BEEP 警告声响。

◆ 步骤 6：进行 EEPROM 93C46 内部 CHECKSUM 值的测试，若值不符合时，也会发出 BEEP 警告声响。

如果上述的测试步骤都通过之后，请将电源关闭，然后再度开启电源，系统会先显示 DIP SW 所设置的 ID 值约一秒钟后，随即可在 7 段显示器上看到现在所测得的温湿度值。由于系统第一次开机时，并未由 EEPROM 上取得正确的校验参数值，所以开启电源后会花较多的时间在 EEPROM 内数据的加载及计算上，并且使用内定的预设参数进行温湿度值的换算。第二次开机时所花的时间就会较短了，亦即约两秒后就开始显示温湿度值了。我们也可以在正面看到 D1 的 LED 灯会一闪一闪地发亮，其闪烁频率刚好是 1 秒钟，LED 灯闪六十次后，即一分钟后会把平均的温湿度值存入 EEPROM 93C46 中，由于 EEPROM 写入的周期约是 10 ms 左右，所以约略可以发觉 7 段显示屏会轻微地闪了一下，就在这一瞬间数据就以串行的方式存入 EEPROM 内部了。

如果我们用手按住温度感测组件，约一秒内就可以发现显示的温度开始上升了，若把手放开后，温度显示也会逐渐降到室温上下，与其他精密温度计比较其误差应该在 1 度以内。如果用嘴对着湿度传感器吹风时，也会发现湿度往上提高了 10～20%，湿度的反应通常较慢，其值要过一阵子约几分钟后才会慢慢降下来。

图 18-6 湿度计表面应随保持清洁，而且最好不要用手去触摸。

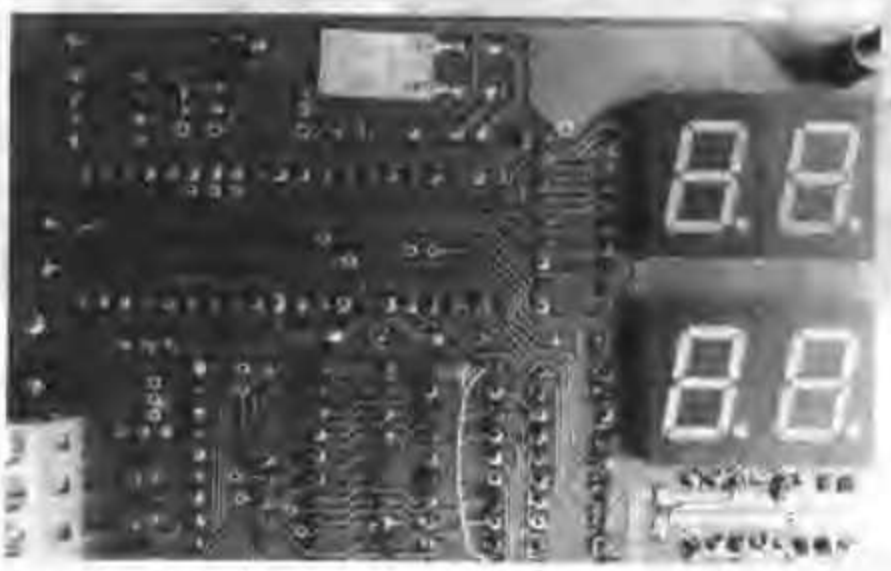


图 18-6

图 18-7 是本控制板已包含整流与稳压，所以 6-12V 的 AC 或 DC ADAPTOR 当电源输入皆可，不过电压愈高的话，散热片旁的温度会愈高。

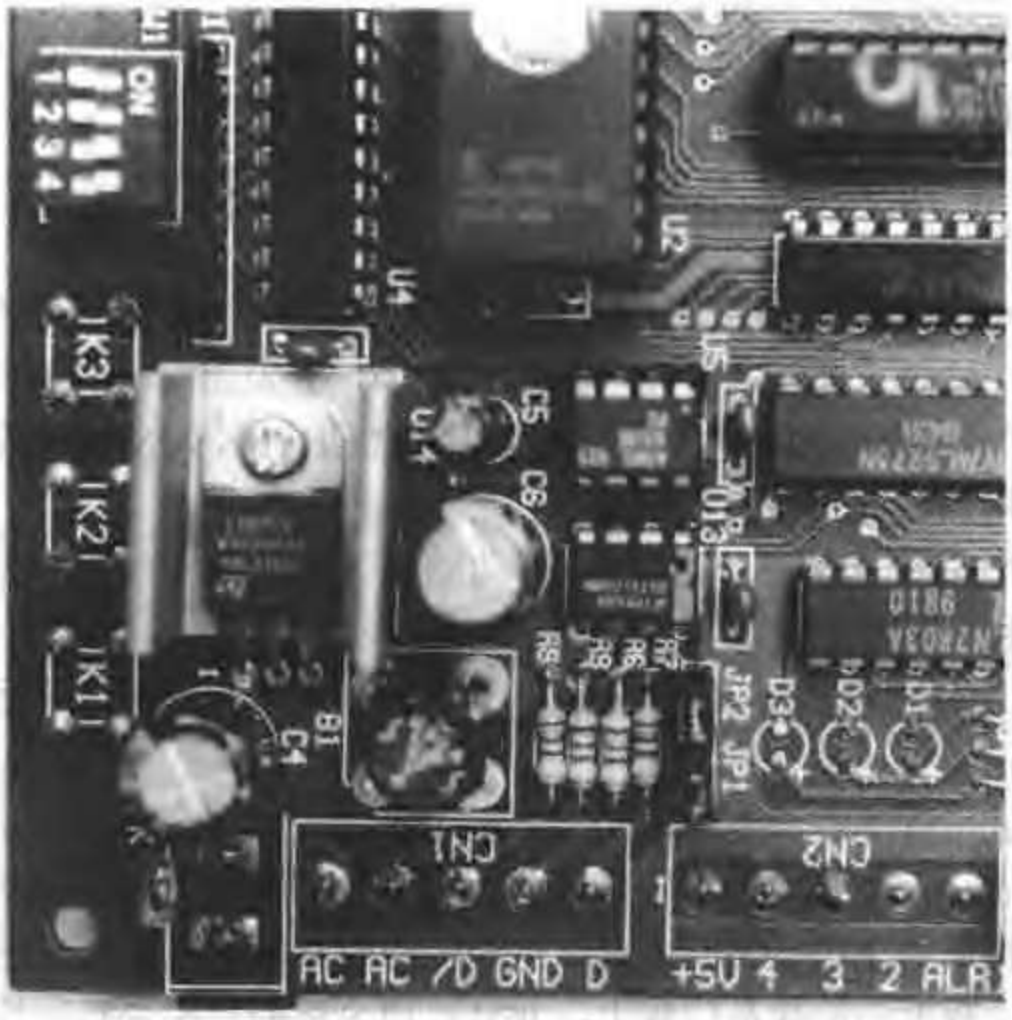


图 18-7

图 18-8 是湿度计控制板上的 PEEL 组件 18CV8 主要在做地址译码。

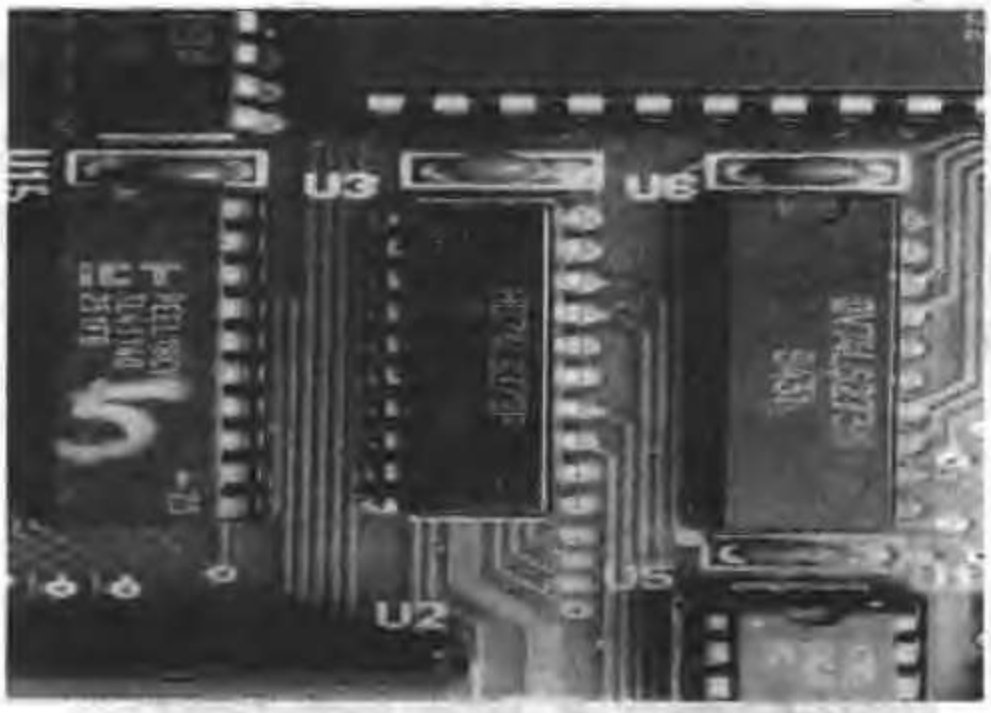


图 18-8

图 18-9 是智能型温湿度控制器暂定的机箱面板图。

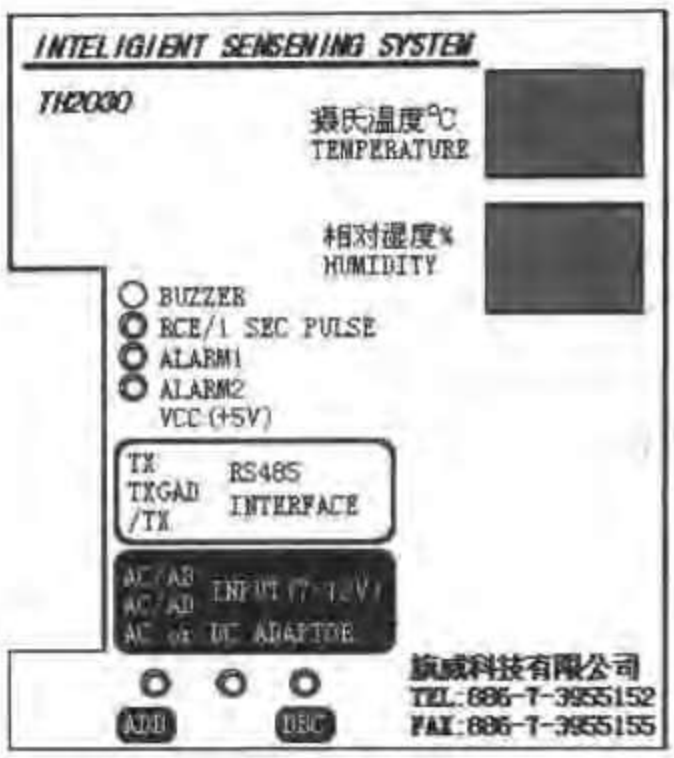


图 18-9

18–4　TH2030 智能型温湿度计的程序介绍

智能型温湿度计本身就是一个相当完整的控制系统，所以 TH2030 的程序除了原本的温湿度读取及换算外，还用了定时中断去做 7 段显示器的笔划扫描，也应用了串行中断去做 RS485 的通信并且判定 ID 码，同时加入了复杂的程序去读写串行的 EEPROM 93C46，这些动作都是相当复杂且具学习性的。

如果对温湿度控制板的制作有兴趣的话，可以参照我们公布的线路自行制作，而线路中的 8KB 主控制程序及 PEEL 数据文件可以另外用磁盘向“旗威科技”公司索取，或者到“旗威科技公司”的网站 http://www.chipware.com.tw 下载这些文件。如果你对这方面的制作及温湿度传感器有疑问的话，请利用电子邮件信箱 E-mail：chipware@seed.net.tw 与我们联络。

图 18-10 是 TH2030 温湿度计上有 SRAM 加电池线路以及串行 EEPROM 93C46 的示范电路。

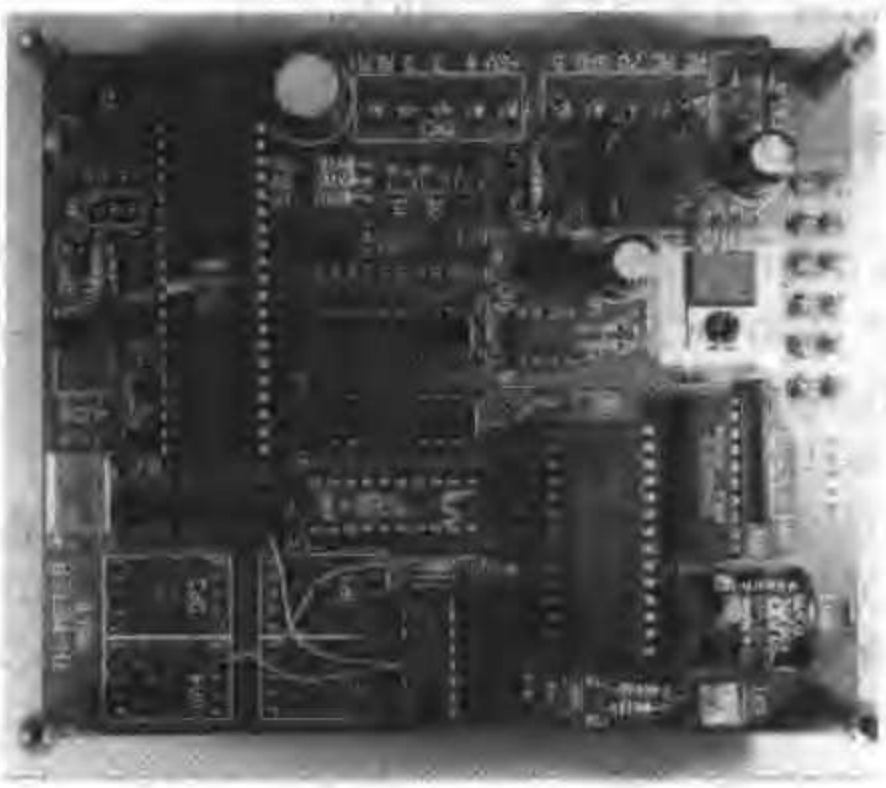


图 18-10

18-5 本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler。

（2）IAR 8051 C Compiler。

（3）ABEL CPLD TRANSLATOR。

（4）PADS PCB 布线软件。

18-6 本章使用硬件

本章使用硬件如下：

（1）CPU AT89C52。

（2）外部内存 62256 SRAM。

（3）SMARTEC 的温湿度感测组件。

（4）1K bits EEPROM 93C46 。

（5）75176 RS485 两线式串行通信 IC。

18-7 相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C52 与 93C46 信息。

http://www.smartec.nl：查询 SMARTEC 温湿度信息。

http://www.ict.com：查询 PEEL18CV8 信息。

http://www.lattice-semi.com：查询 GAL16V8 信息（若买不到 PEEL18CV8）。

http://www.national.com：查询 75176 与 74HC00 信息。

18-8 温湿度控制程序说明

这个程序（TH_METER.C）是旗威科技相当自豪的程序之一，整个程序经过编译后刚刚好是 8192 字节，“凑巧”是 AT89C52 的容量极限。其实，刚完成 70%功能时，程序空间已经约有 6KB 了，当我们把自我测试程序及 93C46 存取程序也加入时，要刻录到 AT89C52 早已超过 11KB，足足多了 3K 以上。

怎么办？换更大容量的 IC AT89C55，它有 20KB 的空间。行不通的!那时 AT89C55 可能价格贵也不好买，我们只好硬着头皮先将程序初步的优化，功能类似的子程序最好共享相同的模块，尽量减少变量的数量，然后把部分 C 程序改用汇编语言来处理，最后再把所有模块做一次优化处理，方才产生刚好等于 8192 字节的程序执行码，而这整个修改与调整的时间就花了近 45 天的时间，光是这点就足够吸引您好好研究这个程序了。写完这个程序我们才体会出什么叫“斤斤计较”，而我们这个程序则是不折不扣的是字字（Byte）计较。

TH2030.ABL 程序彻底公开

```
MODULE th2030
title 'decoder circuit for TH2030'
Declarations
    clk,psen,rd,wr,a15,a14,a13,a12 PIN 1,2,3,4,5,6,7,8;
    ramcs,keycs,alarmcs PIN 19,18,17;

Equations
    ramcs    =!a15 & psen & !rd # !a15 & psen & !wr;
    !keycs   = a15 & !a14 & !a13 & !a12 & psen & !rd;
    !alarmcs= a15 & !a14 & !a13 &   a12 & psen & !wr;
"Test_vectors
END
```

TH_METER.C 程序请查看书附光盘中的 CH18_TH_METER.C 文件。

TH_SUB.C 程序请查看书附光盘中的 CH18_TH_SUB.C 文件。

第 19 章

温湿度传感器应用

本章提供了温湿度传感器制作上的问题解答，并说明了它的应用场合，最后安排了应用串行传输实验，相信必能增加你对温湿度传感器的认识与应用。

上一章我们已经把温湿度计的制作与测试步骤交待清楚了，自行 DIY 的读者应该不会有很大的问题才对。你在自己装时应该尽量运用 AT89C52 上的测试程序，对 PC 板上的零件进行基本的功能确认。以下是一些较常遇见的安装与使用的问题与解答。

19-1 温湿度控制器的问题解答

问题一：我想要完全自己装，是否可以只购买温湿度组件就可以了？

解答：对于想亲自 DIY 的读者我们是百分之百支持的，如果您有此制作计划时，请写一封信（E-mail 亦可）给我们，我们就会把相关的磁盘及手册寄给您参考。请在完成后寄张相片给我们存盘，我们准备把这些经过耕耘的心血照片放置在“旗威科技“公司的网站上供其他 DIY 读者参考与观摩。

问题二：是否可以把温湿度感测组件移到 PC 板外部，以免受 PC 板下方 7805 稳压电路产生热量的影响？

解答：这是一个很好的建议，最初我们所设计的温湿度计是不包含电源整流与稳压的部分，但是一直有读者反应要再花一百元多，去买一个交换式的电源供应器实在划不来。所以我们顺应用户的意思安插一个 7805 的稳压 IC 在控制板上，以便提供稳定的+5V 电源给所有的 IC 使用。同时为了防止用户接错电源线造成电路板的严重损害，所以才在输入端又加入一个桥式整流子，所以不论是交流或直流的电源，只要其电压输入在 7～12V 间即可使用。但是我们忘了提醒用户一个衍生的散热问题，当输入电压与+5V 的输出电压差过大时，会有许多功率会经由 7805 上的散热片消耗掉，这会使 PC 板下半部的温度较高，进而造成温度测量上的误差。假设您是使用 AC9V 的 ADAPTOR 当成电源输入，经过桥式整流后得到的直流电压是 12.6V，如果温湿度控制板需要 0.2A 的电力供应时，则 7805 上的散热片要消耗（12.6V～5V）x 0.2A = 1.52W 的电力，竟然比控制板上实际消耗的还多。所以我们建议用户最好选用 6～8V 左右的 AC 变压器当电源，以节省部分的能源消耗。您也可以修改电源部分的线路，

把+5V 稳压的部分移到外面，这样就不会有热量聚集的问题出现了。在实际使用时，您当然可以把两个感测组件移到 PC 板外面，以便做更适当的感温与量测。不过要留意的是湿度感测组件本身是电容性零件，外接的引线最好不要太长，免得影响湿度的测量。

问题三：温湿度控制器上有一个 1 秒的定时输出，这种安排有何用途呢？

解答：当温湿度计正常工作时，它会不断地送出 1 秒的脉波信号，这可以代表此设备处于正常的状态，所以只要您看到 TH2030 上的第一个 LED 一直在闪烁着，就可以认定其正忙碌于温湿度值的量测。此信号可当成其他设备的计时信号，也可以当成其他控制单元确认温湿度计存在的依据，如果此信号没有出现时，就要启动另一个备用的温湿度控制器了。

问题四：是否考虑过为温湿度控制器找一个象样的“家”？

解答：当然是有的，当温湿度控制器刚完成时，我们就找过做模具的厂商来估价，也有请专门加工有机玻璃的协力厂商做一个样品，但是结果及外观都不是很理想。最后在 10 月的台北电子展上找到一个相当合适的塑料机箱，请看图 1 的安装示范，照片中的风扇是把内部的热量排到外部，以免测到的环境温度不准确，电源线与 RS485 通信线通通由背板上的专用孔引入，并且钻了一排通气孔，以取得湿度值。这个控制盒本身的外观与密封性都相当好，可以很正式地挂或摆在墙壁上，也很适合装在电机用的控制盘中。

图 19-1 是选用标准的塑料箱当成 TH2030 温湿度控制器的“家”。

图 19-1

问题五：为何我制作的温湿度计刚开机时所显示的湿度值都是 99%？

解答：在空气中永远会保有部分的水蒸气，当温湿度计未开启电源时，这些水气有少部分会附着在湿度感测组件上，这才使得刚开机时所测得的湿度值极不稳定，通常要通电至少五分钟后，显示的湿度值才会正式地稳定下来。SMARTEC 的湿度规格中也提到，传感器约要四分钟以上的时间后，才能有一个稳定的电容值出来。因为此时感湿组件表面多余的水蒸气已经完全挥发掉了，所以才能测得更正确的湿度值来。如果可以的话，我们建议用户尽量不要关闭 TH2030 的电源，让它做长时间的温湿度监视，以免影响测量的准确度。

19-2　温湿度计的应用场合

TH2030 温湿度控制器的应用场合是相当宽广的，聪明的读者总是有其特定的使用时机。以下是我们首批用户运用温湿度控制器所做的各种应用实例：

◆ 兰花温室内温湿度的监测

试验可行的话，再做温湿度的实际控制。

◆ 计算机内部 CPU 的温升情况监视

如果 CPU 表面温度始终过高可能就是风扇或散热片不良导致的。

◆ 计算机内部电源供应器温度的监控

温度愈高时，排热用的风扇转速会自动加快。

◆ 硬盘的温度观察

HD 的温度过高可能会影响其使用的寿命。您可以在 HD 外壳上锁一个温度感测组件，当温度超出临界值时就启动备用风扇。

◆ 地下室环境温湿度的监视

如果您是住在湿度较高的北部地区时，一定会有此需求，提醒您何时要开除湿机了。

◆ 银行摆保管箱的环境监控

这种地区一定要做温湿度的监控。

◆ 三次元测量仪器的测试环境监视

金属在不同温度下，其膨胀系数都不相同。所以必须要在几乎恒温恒湿的环境下进行各种尺寸的精密测量。

◆ 办公室的温湿度观察

我们可以看出在哪一个湿度范围下最适合办公。

◆ 马达的温度监视

马达过热且时间一久后，是会引发工安危险的，所以这方面数值的监视是有其必要性。

◆ 气象温湿度值的监视与记录

我们不需要买昂贵的记录仪就可以得到一天内所有温湿度的变化情况，而且还可以通过 RS485 界面传送所有的测试数据。

◆ 小麦草的生长加湿控制

当湿度值过低时，就自行启动洒水设备或加湿机，让小麦草处于最佳的生长环境。

◆ 计算机机房及实验室的温湿度记录

用它来查看冷气设备的效果及冷房速度。

19-3　温湿度计的入门应用——恒温箱的制作

一位好朋友常常对我们“自动控制”这行业感到很怀疑，他总是提出一个问题：“到底你们是在做什么？好像外人都看不懂似的。”不过，最近他总算知道我们所努力的方向了。几星期前他要旗威科技公司帮忙找一台小型的恒温控制箱，针对这类的问题我们总是会询问用户的用途所在，原来他想用来自行制作优酪乳，希望能找到一个恒温箱把温度维持在 42 度上下。由于牛奶加入酵母菌后要 6～8 小时之后才会形成优酪乳，在此温度下酵母菌的生长速度

会最佳，可是如果温度高于 45 度时酵母菌就会开始死亡，所以这段期间温度的监控与保持是很重要的。类似这类的产业用恒温箱价格昂贵，我们的建议是用 TH2030 温湿度控制器来做看看，并且运用板上的温度设定功能，去开启或关闭外部的继电器，进而控制外部的加热设备。这种事总要先试看看，不行的话再买产业用的恒温箱也不迟。

实际上，TH2030 温湿度控制器运用输出信号配合继电器对一台大同电饭锅加热，当温度尚未到达时，电饭锅内的保温电热线持续被加热（真正的煮饭用的电热线，因加热速度过快而无法适用），我们把温度上限值设定在 42 度，当到达摄氏 42 度时，电热线被断电约 60 秒钟（至少保留一分钟的时间），如果温度已降回 42 度以下时则继续加热，反之则仍就切断 AC 电源，让温度缓慢地降到设定值以下，这些监控的动作每秒至少判定三次以上，因此其控制的速度并不快，但是对这一应用却是足够的。

由于控制的对象是大同电饭锅，我们很难把整个温湿度计摆进电饭锅内部，所以便把温度感测组件从 PC 板上移下来，并找来一根测温专用的中空不锈钢棒，如图 19-2 所示，然后把温度感测组件固定在钢棒的最前端，接着引出两条电源线及一条输出信号线到控制板上。测量时，将测棒紧密地接触电饭锅底部的加热线部位，即可得到最正确的电饭锅当时的温度值。

图 19-2

我们把温度感测组件移到感温棒的最前端，如果测量的温度超过 50 度时，最好在电线外部加隔热套管。

一套正式的温度控制系统是相当复杂的。工业应用中最常见的温度控制模式是 PID 控制（比例式的积分微分控制模式），它要导入许多微分与积分的计算式，以期在最短的时间内将温度保持在设定点上。可是这套优酪乳的保温控制系统中，温度一加到设定点后始终保持在此定点上，热量散失的比例不高，我们的温湿度控制器 ON/OFF 反应虽然简易，但是经过实际的模拟试验后，发现乳品的温度可以有效地控制在正负一度以内，所以就放心地使用了，而且所花的费用也是最低的。由于这里的控制条件只有温度而已，所以我们特地把湿度的设定点定在 95%以上，免得板上的蜂鸣器一直叫不停。

图 19-3a 是 Energy Meter 能源监视器，也具备 RS485 通信能力。

图 19-3a

图 19-3b 是 TH2030 温湿度监视板。

图 19-3b

图 19-3c 是 AT89C2051 学习板。

图 19-3c

注：以上是实验桌上有 RS485 界面的仪器或控制板，如果这些设备都可以用 485 串在一起，其应用范围是相当大的。

19-4 温湿度计的 RS485 应用范例

TH2030 温湿度计可以用来对环境做长时间的监视，所以我们在设计阶段就把通信的功能列为必备的。况且我们亦期望能与其他有 RS485 的测量设备用最简便的方式做两线式的通信。我们原先的通信协议很单纯：收到 ID 码后就立即传回该台控制器的温湿度值。这些数据及资料最后都汇整到个人计算机的通信端口或并列端口上，您可以用套装的电子表格程序去分析或画图。以前在 DOS 的环境下所有的驱动程序都需由软件工程师来撰写，现在这种情况已经改观了，只要存成标准的文件格式就可以有各种报表出现。由于考虑各种仪器通信协议的问题，我们打算把 TH2030 温湿度控制器的 RS485 通信再做更周详的规划，欢迎到我们的网站 http://www.chipware.com.tw 下载取得这些资料。

Turbo C 下的 RS485 联机程序范例见光盘文件。

19-5 本章使用软件

本章使用软件如下：

（1）2500AD 8051 Assembler。

（2）IAR 8051 C Compiler。

（3）PC 上的 Turbo C Compiler。

19-6 本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）AT89CXX 刻录板。

（3）SMARTEC 的温湿度感测组件。

（4）7805 稳压 IC。

（5）75176 RS485 两线式串行通信 IC。

（6）恒温槽。

（7）PID 温度控制器。

（8）ENERGY METER 能源监视器。

（9）AT89C2051 学习板。

19-7 相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C52 与 93C46 信息。

http://www.smartec.nl：查询 SMARTEC 温湿度信息。

http://www.ict.com：查询 PEEL18CV8 信息。

http://www.lattice-semi.com：查询 GAL16V8 信息（若买不到 PEEL18CV8）。

http://www.national.com：查询 75176 与 74HC00 信息。

http://www.carlogavazzi.com：查询 Energy Meter 信息。

第 20 章

个人计算机温度监视器的制作

温度的测量是无所不在的。个人计算机 PC 上有几个温度值是应该检测的：CPU 的温度、硬盘的温度以及电源供应器的温度等等。我们示范的是两组温度的监视，其中有硬件的规划以及软件汇编语言的示范。你也可以把这个线路应用在其它温度监控场合上。

20-1 一个逐渐被重视的问题：CPU 的升温

最近几年间，我们看到个人计算机进步的速度是愈来愈快。CPU 本身的体积并未扩大，但是内部容纳的集成电路却一再增加，这使得个人计算机内 CPU 升温，变成一个大家不得不重视的问题。如果 486 等级以上 CPU 背面上不放置一个散热用的风扇，可能在开机后几分钟之内就死机，这也促使设计主板及机箱的厂商更认真地考虑 CPU 的热量散发的问题。如果 CPU 在热量无法散发的环境下继续工作，最后的结果是怎样呢？CPU 内部所聚集的高热会把芯片内部的线路破坏，造成整个 CPU 的永久故障，而这种故障当然只有更换新品了。也许有人会问：这段热量的大量聚集到 CPU，损坏的时间要花上多久？我们没有亲自试过，但是我们相信这段时间应该在几十秒之内。如果您的 CPU 未接风扇而继续通电的话，几分钟之后，很可能就要去购买新的 CPU 了。

图 20-1 是温度感测 IC 在 PC 机箱内所安放的位置，如果能愈靠紧待测物，其温度会愈正确。

即使您的 CPU 加上了原厂供应的风扇，您是否也会考虑到一个严肃的问题：哪天风扇突然不动的时候，接着会有什么灾难发生？由于有这些顾虑，所以某些较高级的风扇电路会在风扇罢工时，立即产生蜂鸣声提醒使用者注意，也有部分的主板在其 BIOS 的设置上，加上了 CPU 温度上限的设置，只要有温度异常时，就可以立即警告使用者。纵使有这些添加的温度检测功能，可是我们还是很想知道几个非常 HOT 的温度值：

（1）到底现在 CPU 的温度是多少？

（2）开机经过了多少时间之后，整个机箱内到达热平衡？

（3）硬盘的温度是热得几乎可烫人或只是温热的？

图 20-1

这些温度的问题看起来似乎与 PC 的使用寿命没有直接的关联，但是您必须要知道，或许将硬盘安装的位置稍微做些修改，就可以将整个机箱的温度降低摄氏好几度。两台相同配备的 PC 会因内部排列的不同，而导致使用寿命的明显差异。如果您的自己装个人计算机并非因超频而经常不正常死机时，请先检查风扇是否已经呈现老态了：风扇的速度时慢时快，导致 CPU 的散热不良。这时若有多组温度计可以实时地显示这些重要温度值的话，绝对可以增加我们判断计算机出问题的原因。

20-2 温度测量的工具

测量温度最方便的当然是体温计了，不过其所能测量的范围在摄氏 42 度以下，要来测量 CPU 的温度行不通。工业上最常用的测温组件是热电耦（Thermal Couple），它是利用两种金属接合后，对温度所产生的电压差，这个电压差只有 mV 等级，必须经过电路放大及线性化后，才能转换成摄氏温度，这种方法可测到 1000℃以上，可是我们预估的 CPU 温度应该在 100℃以内，用热电耦的方式又有一点儿大材小用。

最后我们决定采用 IC 型的温度感测组件来检测 CPU 的升温。由于旗威科技公司之前推出过一款 TH2030 温湿度控制器，板上已经有一个 SMARTEC 的温度感测 IC 及电容感湿组件，我们的构想是将沿用原来的两组数字显示器，然后把感湿组件取下，改成第二个感温 IC，这样就能显示两组温度值了。此时两个感温 IC 都移出 PC 板外，固定在 CPU 散热片及硬盘框架上，分别用三条线（两条为电源线，另一条为代表温度的信号线）传回温度值。在这里我们改用塑料包装的感温 IC，而不是原先的铁壳包装的感温 IC，这是因为其塑料包装的外形有一边是平面的，让我们得以靠紧 CPU 及硬盘都是热热的表面。

图 20-2 是两组温度显示，上排为 CPU 的温度值，下排为硬盘的温度值，看了这些数据后您就知道哪一面可以煎蛋了。

图 20-3 是感温 IC 的三根脚先用热缩套管套住后，再用“快干”粘到 CPU 的热散片及硬盘外壳上。

图 20-2

图 20-3

由于 SMARTEC 的温度 IC 是以其输出数字信号的 Duty Cycle 代表温度值，不需要任何模拟转换电路做输出值的线性化处理就是实际的摄氏温度值了，最后这个值再送到七段显示器上做显示就行了。

20–3　硬件线路的修正

依我们的经验来看，用 SMARTEC 的感温 IC 来测 CPU 的升温是最方便的方法之一。图 20-4 是我们修改后的线路图，由于感温 IC 都在外部，实际上接线时，要留意不要将电源线颠倒，修改的成功率几乎是百分之百的。另一方面要修正的是 AT89C52 内部的程序代码了，此程序代码的二进制文件数据可以到“旗威科技”的技术交流网（http://www.chipware.com.tw/download.htm）上取得。

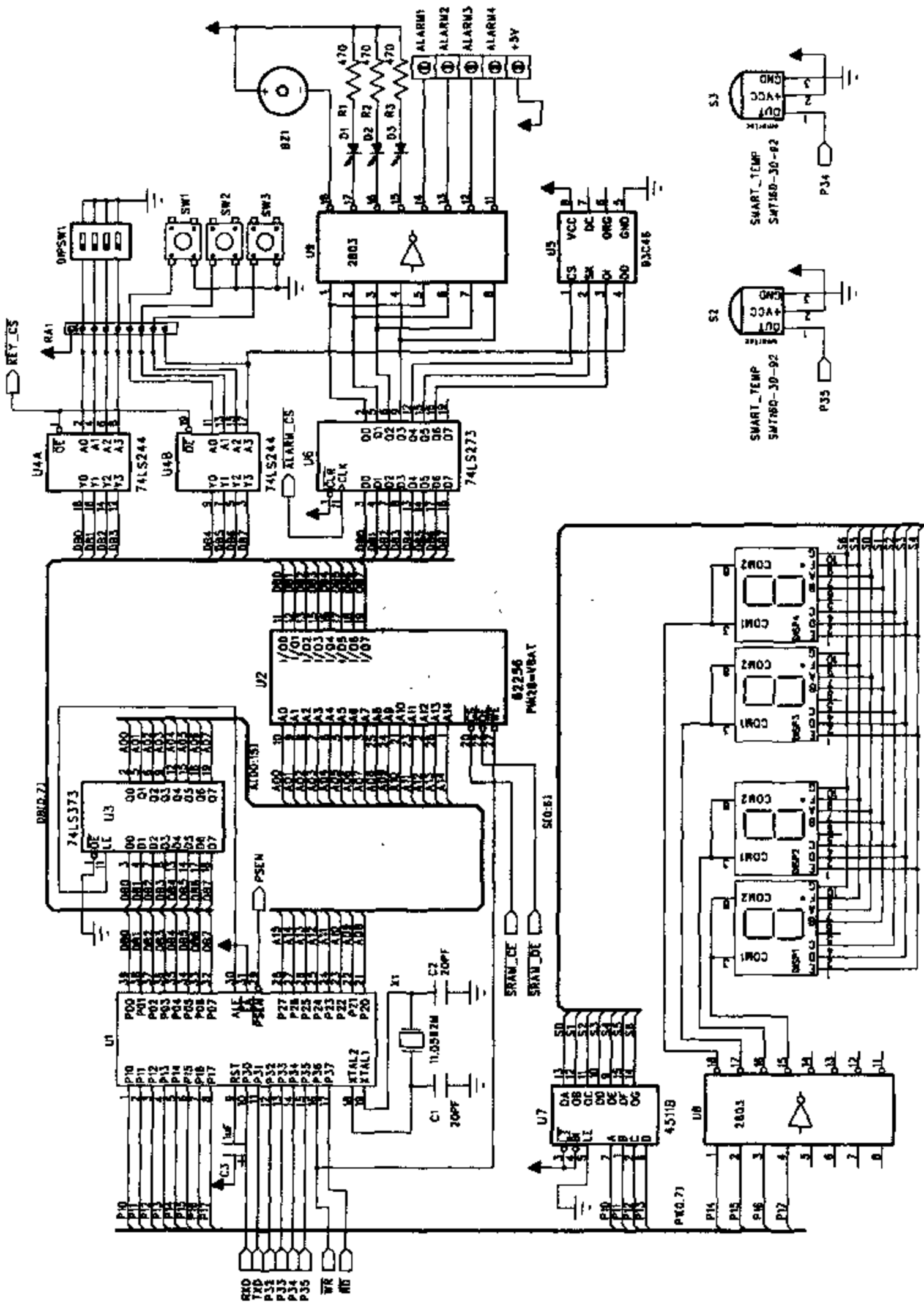


图 20-4

20-4　软件程序的修正

在开发的过程中，我们一直想传递一个重要的观念：所有的程序只要您亲自研究过或做过，一定会留下许多别人无法揣摩的经验。由于先前我们已经用C语言完成整个温湿度计的程序开发，并保留有所有的文件及技术数据，所以要把控制器改为两组温度的显示是轻而易举的，而实际的修改时间只花了不到一周的时间。

如果要从头开发这类型的控制器，一个熟练的程序设计师至少也要花上一个月的时间去完成所有控制程序的开发，其中较重要的子程序有：

- 定时中断程序。
- 七段显示器的显示。
- RS485 串行通信程序。
- 温度感测 IC 的 Duty Cycle 测量。
- 两组温度值的换算。
- 温度上限值的设置及判断。

这个控制程序也是一个C语言与汇编语言深度应用的例子，所有的子程序集合在一起时，许多程序设计的细节都会一一突显出来，例如：线路上的显示器是使用定时扫瞄的方式。若花太多时间在温度值的计算时，扫瞄的动作就会被中断，我们观看时就会感觉显示器好像会晃动似的。另一个例子是当感温 IC 存在时，所有的温度显示都是对的，可是温度 IC 的接线中断或根本未装上时，程序该如何处理呢？这些都是一个完整程序所要考虑到的。由于 SMARTEC 的感温 IC 需要校正的机会非常小，我们顺便将此方面的程序取消，程序代码的空间也跟着减小了许多。这些程序的实战经验是无法在一般的 8051 程序设计的书中看到，如果您真的想对 8051 进一步地了解时，自行尝试规划一套类似的系统，然后亲自地从头到尾做一次，我们保证绝对可以学到许多经验，这些将是我们面对下一个设计项目的许多重要参考程序之一。

在 8051 单片机的开发过程中，我们已将近三年的时间没有碰 ICE 仿真器了，这并不是意谓 ICE 不好用而不用，而是我们已自行发展出一连串的除错步骤，只要有一台 AT89C52 的刻录器（8051 单片机控制板＋AT89CXX 刻录器），将程序直接刻录到芯片后立即执行，就是最严谨的软件基本验证程序，只要经过这个关卡的认证，其它软硬件的问题自然全部有解了。而我们所借助的工具就只有一台 486 计算机，AT89C52 刻录器及一台 TEK 的储存式示波器。以后有机会的话，我们会逐一将这些经验公开出来。前几天刚碰到一个微处理机的老前辈，他至今仍是用机械码来写控制程序，虽然这是最原始的方法，现在看来也许是最没有效率的做法。事实上，在微电脑的领域中再也没有任何障碍可以阻挡这类型的技术人员。

程序 1　读取温度的汇编语言部分程序

```
read_temp:
        MOV     R2,#00H
        MOV     R3,#00H
HI      JB      P3.5,HI
LO      JNB     P3.5,LO
```

```
        SETB    P3.2            ;START
LOGIC1  JNB     P3.5,LOGIC0
        INC     R2              ;R2++ high byte
        SJMP    LOGIC1
LOGIC0  JB      P3.5,TEST_E
        INC     R3              ;R3++ low byte
        SJMP    LOGIC0
TEST_E:
        CLR     P3.2            ;STOP
        RET                     ;return value in R2,R3

int c_read_temp()
{
 char x;
 int m,n,y;
 int offset,gain;
 unsigned int hh,ll;

        offset=eeprom_read(A_T_OFFSET);/* 由 93C46 上取得校验调整值 */
        gain   =eeprom_read(A_T_GAIN);
        if (offset> 88 || offset< 48) offset= 68; /* 校验值的判定 */
        if (gain   >222 || gain   <202) gain   =212;
        for (x=0,y=0;x<TEMP_CNT;x++)
        {
         hh=ll=0;
         for (n=0;n<10;n++) { /* 一次读取 10 组温度值 */
          clear_bit(EA_bit); /* 暂时停止中断 */
          m=read_temp();          /* 调用汇编语                */
          set_bit(EA_bit);        /* 恢复中断服务程序 */
          hh+=m/256;
          ll+=m%256;
        }
}
```

20-5 温度控制器的温度读取核心程序

程序 1 是我们的温度测量中最重要的核心程序。温度感测 IC 从 P3.5 输入，程序开始测量前一定等到上升缘开始时才进行 R2 与 R3 的计数。为了要与C程序结合，我们将传回值摆在 R2 与 R3 缓存器上，R2 代表输出是 Low 的时间值，R3 则是输出是 High 的时间，这个值再经过转换后就可求得温度值，由此我们也可能看出：SMARTEC 的感温 IC 的温度输出，只

和 Duty Cycle 的比例有关，与其它时序因素无关。

20-6　本章使用软件

本章使用软件如下：

（1）8051 Assembler 汇编语言编译器。

（2）IAR 8051 C Compiler 编译程序。

（3）PC 上的 Turbo C 或 Borland C。

20-7　本章使用硬件

本章使用硬件如下：

（1）FLAG51 单片机控制板。

（2）AT89CXX 刻录板。

（3）AT89C52 8KB 闪存微控制器。

（4）SMT160 温度感测 IC。

（5）热电耦（Thermal Couple）。

20-8　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询单片机控制板相关信息。

http://www.atmel.com：查询 AT89C52 相关信息。

http://www.smartec.nl：查询 SMT160 温度 IC 信息。

http://www.national.com：查询 LM35 温度 IC 信息。

http://www.omega.com：查询热电耦的相关技术信息。

另外你也可以到各主板的网站上，比较各种主板是如何检测 CPU 的温度，而当温度过高时，其 BIOS 是做了那一种对应的处置。

TWOSMART.ASM 程序请查看书附光盘中的 CH20_TWOSMART.ASM 文件。

第 21 章

RS485 通信接口彻底研究（一）

听过 RS485 通信吗?它与我们常见的 RS232 串行通信很类似。不过，RS485 兼具距离长与不怕噪声干扰的优点，很早就应用在许多工业控制的场合里，许多五星级大饭店房间管理与 KTV 的点歌系统也是 RS485 的应用实例。我们将针对 RS485 通信接口的 IC 使用、网络连接、通信协议及其学习的工具，作实践上的分析。

个人计算机上所使用的串行通信接口是标准的 RS232C，其中的 RS 是 Recommended Standard 的缩写，这是由美国的电子工业协会 EIA（Electronic Industries Association）所制定出来的工业标准。通常一台 PC 上有两个通信接口：COM1 与 COM2，我们常把 COM1 接鼠标，COM2 则接调制解调器，做一般的应用是足够的。如果你还要学习 8051 单片机控制板的话，就要把原来接到调制解调器的接头改接到 8051 控制板上，以便将机器码下载到控制板的 SRAM 区中。可是如果一次要控制两台以上的单片机板时，就要大费周章了，这是因为 RS232C 的接口当初是规划来接调制解调器的，所以是标准一对一的接法，想要一对多是不可能的，RS232C 所能容许的联机长度约为 15 米，如果联机距离还要更长时，就心有余而力不足了。除非是将通信接口更改成其他的接口标准，才能兼做长距离且一对多的通信，而 RS485 则是众多选择其中的一个。由于 RS485 的接线非常方便，早已用在许多工业控制及商业的应用上，如果你正在研习 8051 单片机的话，除了学好基本程序的写法外，若能顺便得知 RS485 的控制方式，对多台微处理机的设计及应用上绝对是有帮助的。尤其值得一提的是 8051 在串行通信的功能上做得相当成熟，可做一对一通信，也可以一对多通信，如果这部分不熟悉的话，有很多应用就使不上力了。

21-1 RS485 与 RS232C 的比较

RS485 工业标准比起 RS232C 有许多优点，它可以在一个联机中连接多达 32 个接收及发送者，连接长度更长达 1200m，短距离的通信速度可以到达 10M bit/s，同时 RS485 收发 IC 的价格除了不算贵之外，仅要+5V 的电源供应，这会使控制系统的电源供应变得非常单纯。表 21-1 就是 RS232C 与 RS485 的主要特性比较。

表 21-1 RS232C 与 RS485 主要特性比较表

	RS232	RS485
传输模式	非平衡式	平衡式
最大电缆长度	15m	1200m
最快传输速度	20K bit/s	10M bit/s
最小发送端输出	5 V	1.5 V
最大发送端输出	15 V	6 V
接收端灵敏度	3 V	0.2 V
最多发送端数	1	32
最多接收端数	1	32
发送端负载	3K-7kΩ	60Ω（最小值）

21-2 认识 RS485 接口

RS485 接口传输是所谓的平衡式传输，它是指其发送是两线式的，而且这两条在线的信号是互为反相的，接收端就以这两端的电压差来反应接收到的信号，这就是电路测量学上的差动量测（differential measurement）。平衡式传输最大的优点是抵抗噪声，通常的噪声包括电压突波、火花、振荡以及电磁干扰等等，若它们是从传输在线进入，由于 RS485 是平衡式的接法，这些干扰会在接收端被抵消掉，这使得 RS485 的抗干扰能力较 RS232C 高出许多。平衡式传输另一个优点是，对于每个 RS485 节点的对地电压差有相当的免疫性，因为如果每个节点都相距甚远时，传输线对地的电压都有所变化，但是差动电路仅对输入线的电压差反应，所以每个节点的对地电压不同时，RS485 仍然能愉快胜任。

RS485 所使用的电缆线是便宜的双绞线（twisted-paircable），这是指这两条线是相互对绞在一起，这种线材可以把感应到的电磁干扰信号相互抵消掉。虽然 RS485 是使用+5V 的电源供应，但是其在接口上的准位却不是标准 TTL 或 CMOS 的准位。如果我们把传输在线的信号称为 A 与 A 时，对输出端（driver）而言，通常 A 与 A 的电压差都是+5V 左右，可是只要 A 较 A 高 1.5V 就是逻辑上的 1，而当 A 较 A 少 1.5V 时就是逻辑 0。RS485 信号经过长距离的传送后，一定会有所下降，因此在接收端（receiver）的灵敏度就要高一些，只要两线的电压差超过 0.2V 就认定是有效的准位。所以在接收端上，A 较 A 高 0.2V 时就是逻辑 1，A 较 A 低 0.2V 时为逻辑 0。若输出端在最差的情况下，使得 A 与 A 的电压差只有 1.5V 时，只要在传输在线的衰减量小于 1.3V，其他的接收端还是能反应正确的逻辑准位。

图 21-1 是一个 RS485 网络中可以有多达 32 个收发装置。

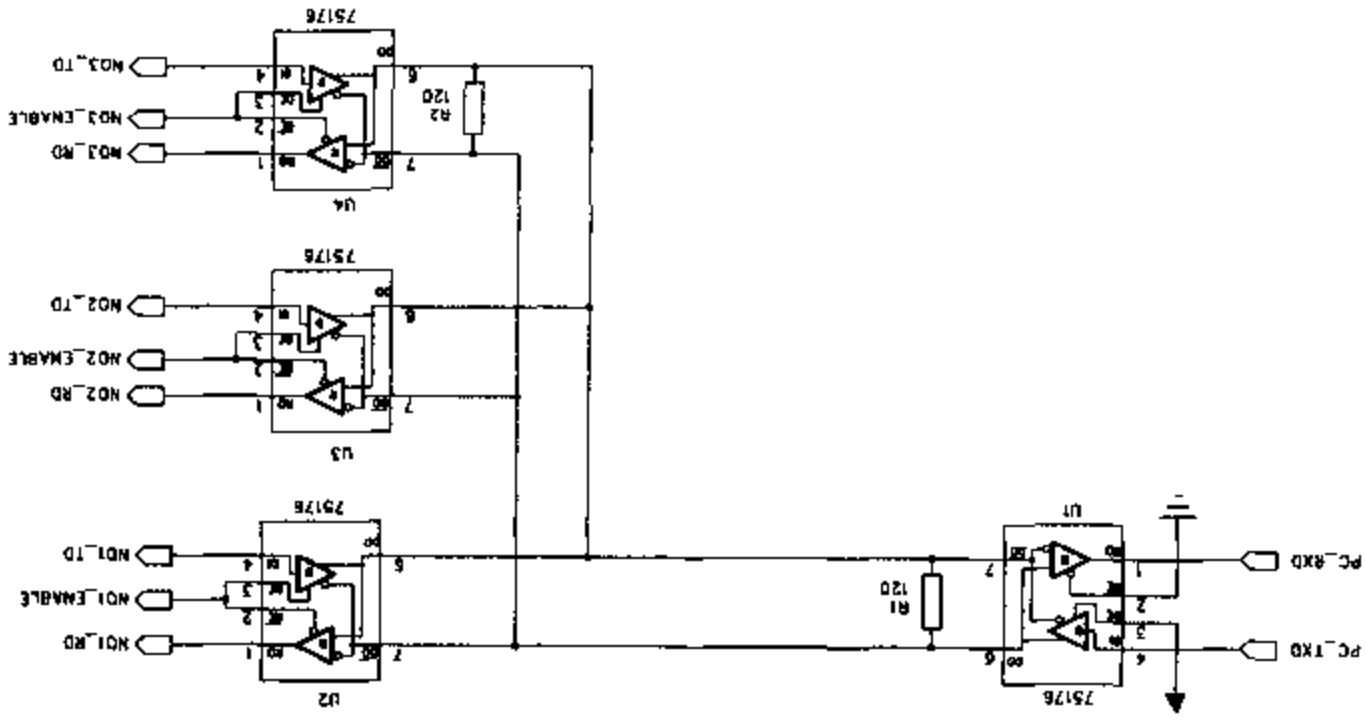
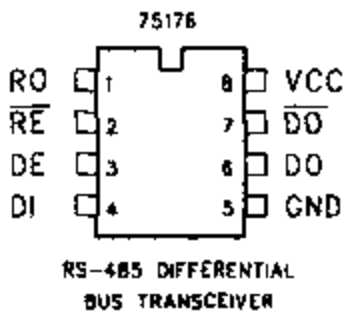
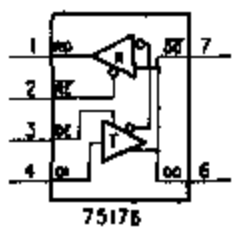


图 21-1

21-3　RS485 接口 IC 的使用说明

RS485 专用的收发 IC 中，最常见的有 TI，National Semiconductor 及 Motolora 的 7517X 系列，我们这里仅举出其中最常用的 75176，图 21-2 是其引脚图及接收与发送端的特性表，75176 内部各有一个发送端及接收端，是否发送是由 DE（drive enable）脚来控制，是否接收则由 RE（receive enable）脚控制，而且 DE 与 RE 的控制状态刚好反相，所以，我们可以用一个控制脚去掌握 75176 是接收还是发送。由图中我们也可以看出 75176 实际的发送与接收点是共享的，所以在传输上是归类成半双工（Half-Duplex）的通信方式，这就是说：每个节点都能接收与发送数据，但是不能同时收发。半双工的通信方式对 RS485 而言是最方便不过的，而且可以使接线数降到只有两线。





RECEIVER		
DIFFERENTIAL INPUTS DO-$\overline{DO}$	ENABLE $\overline{RE}$	OUTPUT RO
V >= 0.2Volt	L	H
V <=-0.2Volt	L	L
X	H	Z
INPUT OPEN	L	H
-0.2V < V < +0.2V	L	?

DRIVER			
INPUT	ENABLE	OUTPUTS	
DI	DE	DO	$\overline{DO}$
H	H	H	L
L	H	L	H
X	L	Z	Z

图 21-2

注：上图中的 Z 代表高阻态输出，?代表状态未知。

21-4 RS485 网络的分析

图 21-1 是一个典型的 RS485 连接方式，图 21-3 左方是个人计算机 PC 的 RS232 通信端口，经过一个 RS232C 到 RS485 准位转换电路，将原来信号改变成 RS485 的准位，由图中还可以看到：我们是用 RS232C 上的 RTS 脚去控制串行数据的流向，当 RTS=1 时就是要发送数据，反之则是接收数据。在 RS485 半双工的通信中，所有的控制器大部分时间都是处于接收的状态，以便能收到其他设备传来的信息。由于 PC 端上是经由串行通信接口送出信号，所以传输速度上有些限制，无法到达 RS485 的传输极限 10Mbit/s，不过，我们最常用的传输速率 9600bit/s 或 19200bit/s，对 RS485 而言都是轻而易举的。传输速率的快慢也要考虑到接收端的 CPU 处理速度，若接收端是 8051 时，传输速度若高过 100K bit/s 且数据量甚大时，可能就无法反应过来了。

RS485 标准中并未规定信号的功能，甚至引脚及联机接头也都不做定义，这些都保留给设计者去发挥。只要每个节点的两个通信接点都并接在一起，就可以进行相互间的通信，由范例图中可以看到所有标示 DO 的地方都接在一起，标示 DO 的地方也接在一起。

讲到 RS485 网络也不可避免地要提到传输线（transmition line）及终端电阻（terminator）的概念，在传输线中最重要的概念是要一定用终端电阻去吸收发送端传来的信号，而其电阻值要刚好等于传输线的阻抗值。终端电阻只需要两个，而且要加在传输线的开头与结尾两端上。如果不加会怎么样呢？这会使通信信号无法正常的传递。为何情况会这样，请看下面的简单描述。

如果我们在游泳池的中央产生一个波浪，这个波浪会逐渐往游泳池两边移动，当波浪到达池边后，由于池边通常是厚厚的水泥结构，会再度产生较先前稍微弱的反射波，然后再往游泳池对方传递，如果我们此时又发出一个波浪时，这个波浪会和前一个波浪的反射波重叠或抵消，造成接收端信号的不正确。如何解决这个问题呢？最简单的方法就是不要有反射波的出现，如果我们在游泳池两边都加入小型的“消波块”，尽量吸收掉发射波的能量，那就可以使得反射波几乎不会出现，从而保证发送信号的正确性了。所以，我们在 RS485 传输线两端所加上的终端电阻就是高频信号的“消波块”，它在终点处吸收掉信号，以免有反射信号又再度灌回传输在线。先前我们提到传输线是有阻抗的，而且理论上这个阻抗值是不会与传输的长度有关，所以不论联机的总长度为何，只要我们选择的终端电阻值刚好等于该传输线的阻抗值时，就可以有效地吸收发送端的信号。如果您想更了解传输线的理论及特性，请在 Internet 上使用 google 的搜索引擎去搜索 transmition line 这个关键词，我想一定能找到数十篇以上这方面的技术文献，在这里我们就不多做理论性的描述了。

以 AWG24 号的标准双绞线电缆为例，其特性阻抗值约在 120 欧姆上下，因此我们选用的终端电阻阻值应该是 120 欧姆。在半双工的传输在线，在线两端的终端电阻并联后的阻抗值为 60 欧姆，所以 RS485 的每个发送端必须有推动总体阻抗 60 欧姆的能力。在实际的测试当中，我们以 bit/s 的速度传送数据时，短距离内的传输是否加上终端电阻都能动作，不过，为了线路的稳定起见，我们还是建议您在传输线的两端加上终端电阻。如果您想用 8051 做更高速的数据传输时，请一定记得要加上 terminator（终止器）。

图 21-4 是 AT89C2051 学习训练板上的内建 RS485 的硬件电路。

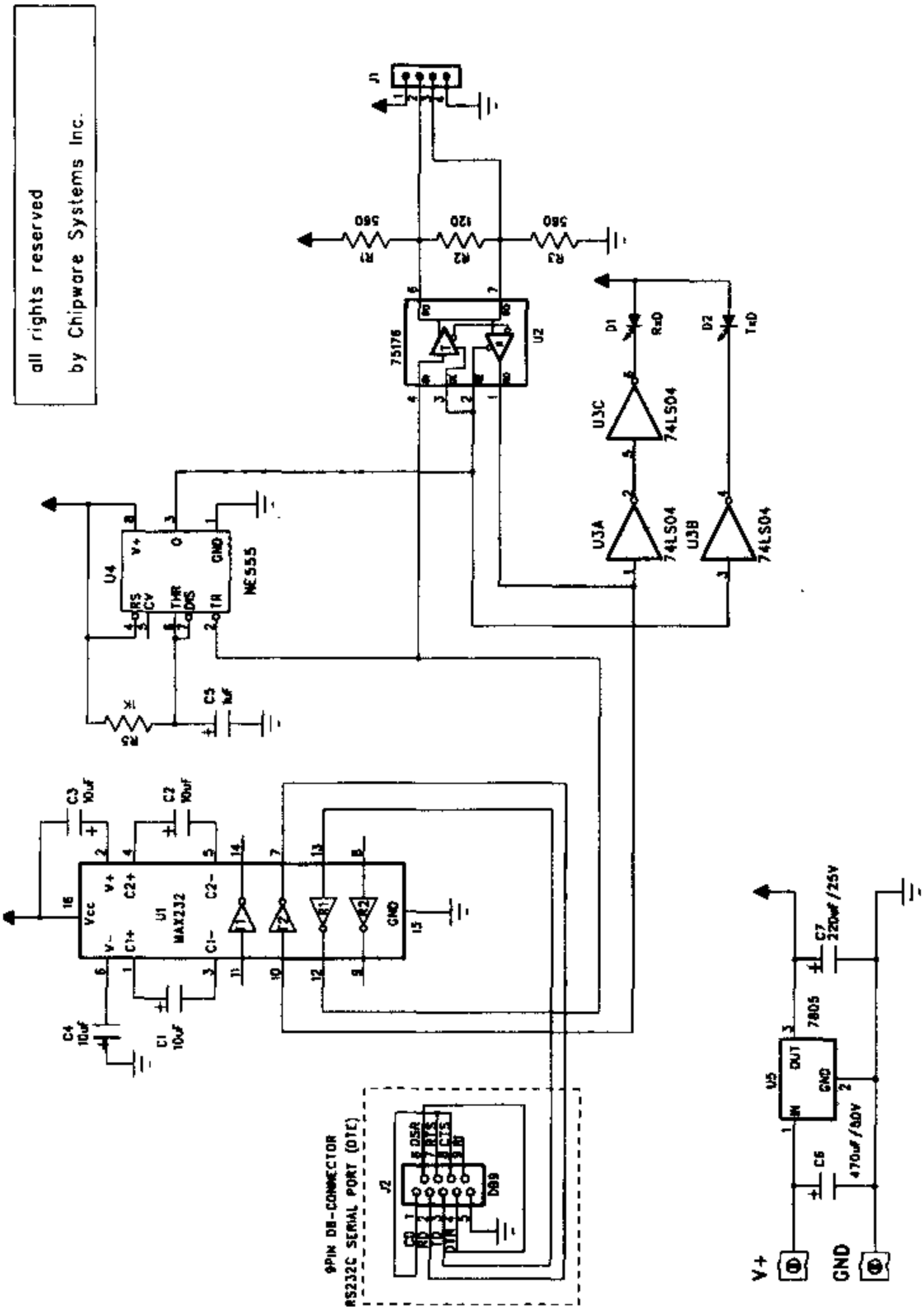

all rights reserved
by Chipware Systems Inc.
RS232 TO RS485 converter
J1
R1 560
R2 120
R3 560
U2
75176
D1 RxD
D2 TxD
U3C 74LS04
U3A 74LS04
U3B 74LS04
U4 NE555
R4 1K
C5 1uF
U1 MAX232
C1 10uF
C2 10uF
C3 10uF
C4 10uF
9PIN DB-CONNECTOR
RS232C SERIAL PORT (DTE)
J2
DB9
U5 7805
C6 470uF/50V
C7 220uF/25V
V+
GND

图 21-3

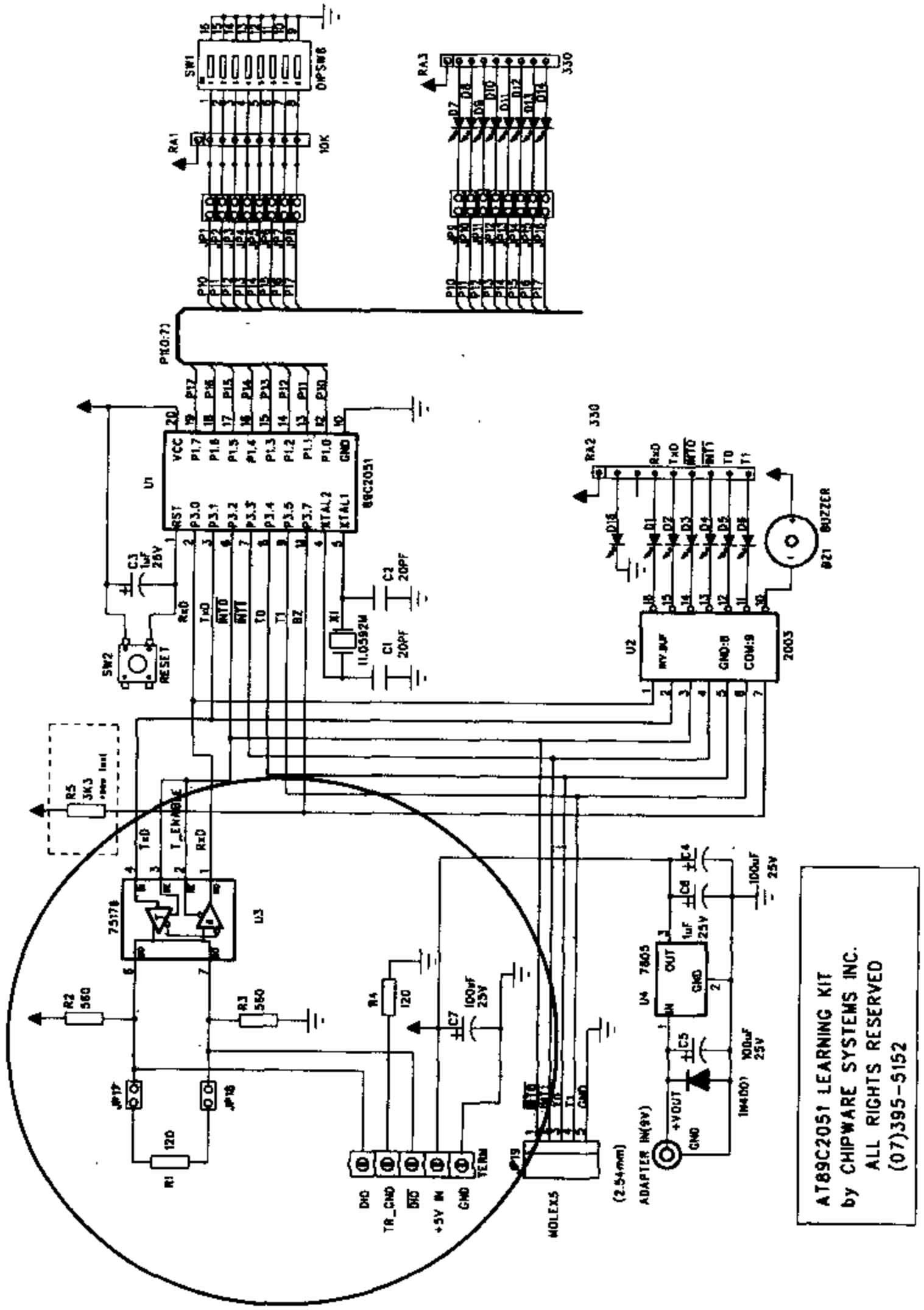


图 21-4

在 75176 内部的信号输入端 DO 及 DO 上各有一个 100K 欧姆左右的电阻，分别提升到 +5V 及地端，这样没有信号时，DO 点的信号准位一定比 DO 还要高。这会使得接收端的输出保持在 1 的状态，不过当我们接上终端电阻（120 欧姆）后，产生分压的效应，这会导致 DO 与 DO 间的电压差小于 0.2V，进而使输入接收端误判信号，所以我们特地在 75176 的外部加上两个 560Ω 的电阻，故意使 DO 点一定比 DO 高出 0.25V 以上。由于 RS485 并没有用到对地的参考电位，所有的节点还是需要一个共同的参考地点，有两种连接的方法：一是每个节点分别接到大地的地点上，另一个方法是多一条导线连接所有的地点。在真正的线路当中我们在共同地点与各个节点的 GND 点间串入一个 100 欧姆的电阻，以保护该节点不会因故障时有过大的电流通过，进而损害该节点上的设备。不过如果 RS485 接口已有光耦合 IC 进行 GND 隔离时，就不需要此条信号了。

图 21-5 是 AT89C2051 学习训练板也是运用 75176 做两线式的串行通信。

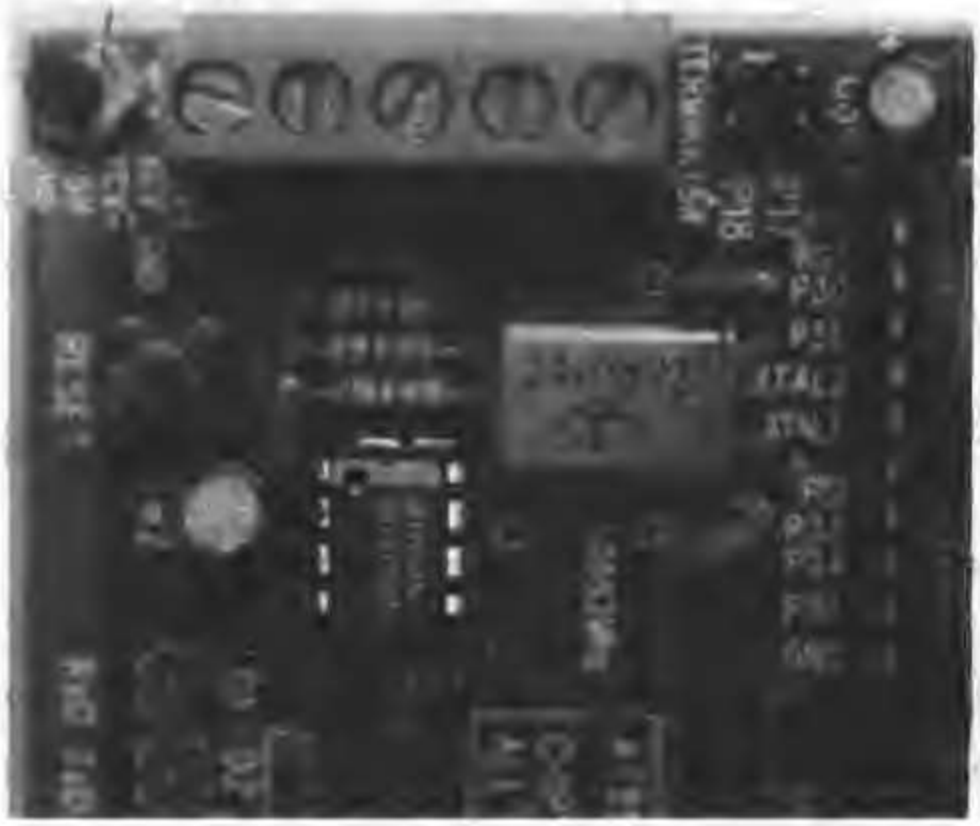


图 21-5

21-5　RS485 的通信协议

在 RS485 的通信上因为收发都是共享相同的线路，就无法像一对一的 RS232C 一样可以全双工通信。所有的仪器设备都共享一条 RS485BUS 线，在通信开始前我们必须先指定某一个设备或计算机有控制权，其他节点各有一个 ID 码，主控者可以指定哪一个节点动作及送回数据，其余被动者只能依照有控制权者的指示来送出数据，而且绝对不允许私自送出数据。这些种种的限制是要保持线路数据的畅通，如果没有这些严谨的规定时，大家都把数据往在线丢，除了会造成大多数数据碰撞在一起外，接收端也无法取得正确的数据。

如果 RS485 在线两台设备要传递数据，最好也要通过主控制者的转接才行，首先由主控制器指定甲设备送出一段数据，先传到主控制器的 RAM 区中，然后再由主控制者将数据传送到乙设备上。由于主控制器有最大的总线主宰权，所以这些数据在传送当中，不会还有其他节点送出数据来，造成整个传输线的停摆。可是任何通信都会碰到一个问题：叫到某个节

点可是一直没有响应，这时该怎样处理呢?主控者要有“逾时不候”的决心，当送出一个命令给某节点希望它回送数据，如果等待时间超过 50ms 后，就自动跳离开，我们称这个动作为（timeout），以免整个控制系统在你我相互等待的状况下停摆。超时的时间过短会使得该节点尚未反应就跳离，时间过长时则导致系统的反应迟钝，使得总体处理效率降低。

我们最后以一个几年前完成的 KTV 点歌系统来说明整个 RS485 的联机是如何工作的。首先主控者是一台在 DJ 控制室的个人计算机，与 PC 连接的是 31 个 KTV 歌唱厢房，PC 开机后就进行每个厢房的联机侦测，如果送出一个 ID 且伴随测试串行信号，若立即有响应时，代表该厢房正在使用中，所以接着送出是否需要点歌的信息，若在此之前已有人按下歌曲的代号时，厢房中的设备就立即把此代号送回给在 DJ 室的个人计算机，并且立即显示在 PC 的屏幕上。接着 PC 发出 ID 与询问信号给下一个厢房，有服务需求时则再拿到点歌代号，反之就转询问下一个厢房，若每一个厢房占用 RS485 总线上 10ms 的时间，则询问完 31 个厢房约要 0.3 秒的时间。PC 就是以这种询问（polling）的方式服务 31 个厢房。这种反应速度对 KTV 的点歌系统以经绰绰有余了，而且操作者也无从察觉系统是如何运作的。

类似的应用也可以用在防盗系统以及生产机器控制上，自从单片机 8051 问世后，在串行通信上也添加了多微处理机通信的功能，做类似的应用绝对不成问题。很可惜的是一些 8051 教科书都蜻蜓点水地提到可以做这方面的应用，可是进一步的说明却含糊带过，这一重要的部分知识教导与实验就由“旗威科技”来做了。

21-6 学习 RS485 通信的工具：AT89C2051 训练器

如果您想做 RS485 通信的实验，以下几个工具是必备的：

（1）个人计算机。编写 8051 的通信程序及测试 PC 端的 RS485 通信程序。

（2）8051 单片机控制板。你可以在旁边的串行通信接头加一块附加板做 RS485 的实验。

（3）AT89C2051 学习训练板。板上已内建 RS485 接口，接上 9V 的 ADAPTOR 即能工作。

（4）AT89C2051 专用刻录器。第 3 项上的 CPU 必须靠此刻录器将 2KB 以内的 8051 程序烧入，如果您有万用刻录器，也应该可以找到刻录 AT89C2051 的程序。

（5）示波器。观察 RS485 的波形，若是储存式的示波器更佳。

（6）数字电表。检修及简测量电路用。

（7）RS232C 到 RS485 的转接板。你可以参考我们公布的电路自己焊接一块，或是买市面上已有的产品。这些外加的转换板都要额外的电源，连接时请确实检查一次才加入电源。

有了以上的工具之后就可以开始着手规划 RS485 程序了，为了解说方便起见，我们所有的 RS485 程序范例都将以 AT89C2051 训练板为主，相同的程序也可以移转到 AT89C51 或 AT89C52 上，或是下载到单片机控制板上。下一章节起我们将把学习的重点摆在 RS485 的串行通信上，逐一地讨论如何用 8051 汇编语言写 RS485 收发程序，如何在 PC 上规划 RS485 程序，最后所有的有 RS485 接口的设备全部连在一起，场面一定相当壮观的。

如果您在 8051 的应用碰到大问题，或是学习上遇到障碍都欢迎来电寻求援助及引导，唯一不受理的是学校专题做不出来的求救。我们会把这些问题整理成“问题与解答”，并且在适当的时候公布给其他的读者参考。当然您也可以通过 Internet 的电子邮箱 E-mail：

chipware@seed.net.tw 与我们取得联络。另外您还可以加入 Internet 以下几个有关电子方面的讨论组，取得世界各地第一手的电子相关信息：

sci.electronics.basics 电子方面的基础问题及解答。

sci.electronics.cad 线路图输入，PC 板布线及电路仿真等等。

sci.electronics.component 电子零组件的认识与采购。

sci.electronics.design 电路设计。

sci.electronics.equipment 相关的电子仪器设备。

sci.electronics.misc 电子相关领域的探讨。

sci.electronics.repair 如何维修的专门讨论群。

21-7　本章使用软件

本章使用软件如下：

（1）Borland Turbo C V2.0。

（2）2500AD 8051 Assembler 和 C compiler。

21-8　本章使用硬件

本章使用硬件如下：

（1）75176 485 收发器。

（2）AT89C2051 学习板。

（3）RS485 专用隔离双绞线。

（4）RS232 状态监视器。

（5）RS232 转 RS485 转换板。

（6）Tektronix 数字示波器。

21-9　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询 8051 控制板相关信息。

http://www.ns.com：查询 75176 IC 相关信息。

http://www.rs232.com.tw：查询各类 IC 的数据表。

http://www.tek.com：查询数字式示波器信息。

第 22 章

RS485 通信接口彻底研究（二）

> 串行通信的程序撰写通常是极具挑战性的，我们的经验是一定要配合示波器的信号观察才行，因为通过仪器你才可以真正看到信号相互传递的画面，以及每个装置的反应速度，学 8051 单片机若没有应用到串行通信，那就太可惜了。

在单片机 8051 的发展过程中，写串行通信程序是相当有挑战性的。尤其是在缺少 8051 在线仿真器（ICE）的情况下，往往很难立即分辨出错误的地方来。但是，众多的读者中拥有高价位 ICE 的应属少数！所以我们接下来的 RS485 串行通信实验，将示范不靠任何厂牌的 ICE，完成所有通信程序的撰写。虽然试验时要多花一些时间，但是彻底地理解开发过程中的诀窍后，除了可以省下实质开销外，还可以摆脱“没有 ICE 就无法开发”的依赖心理。现在很多单片机的教学及书籍中，所有的程序示范及排错上经常过于依赖 ICE，往往给学习者一个错误的观念：少了 ICE 就一切停放了，事实却不是这样的。

串行通信程序由于牵涉到收发双方的协议，如果一开始就埋头苦干写通信程序，可能会无从找出错误的所在，因此我们必须分两头做详细的检查，确定这方传送的方式正常后，再写对方的接收程序，这样才能在短时间内完成双方通信协议的连接。另外 8051 的通信模式中也有支持多微处理机通信的模式，这也是我们急于理解与试验的。

22-1 MASTER 端 RS485 通信的写法

在RS485 串行通信试验的初期，我们首先选用一台个人计算机当成整个网络中的主控者，用 PC 的串行通信端口 COM1 或 COM2，产生 RS485 串行信号给其它的外部设备。图 22-1 是我们上一章中所公布的 RS232 转 RS485 的线路架构。线路中先把 RS232 的准位用 MAX232IC 转成 TTL 准位，然后把此信号经由 75176 转为 RS485 的准位。其中较特别的是我们用 RS232C 上的 RTS 线去控制 75176 的收发状态。而线路上的 LED D1 与 D2 则在指引收送信号是否有发生。在线路中我们照例把 DSR 线与 DTR 线短路，以骗过 PC 上的 BIOS 以便能顺利收送串行信号。图 22-2 为转换板雏型板实物图。

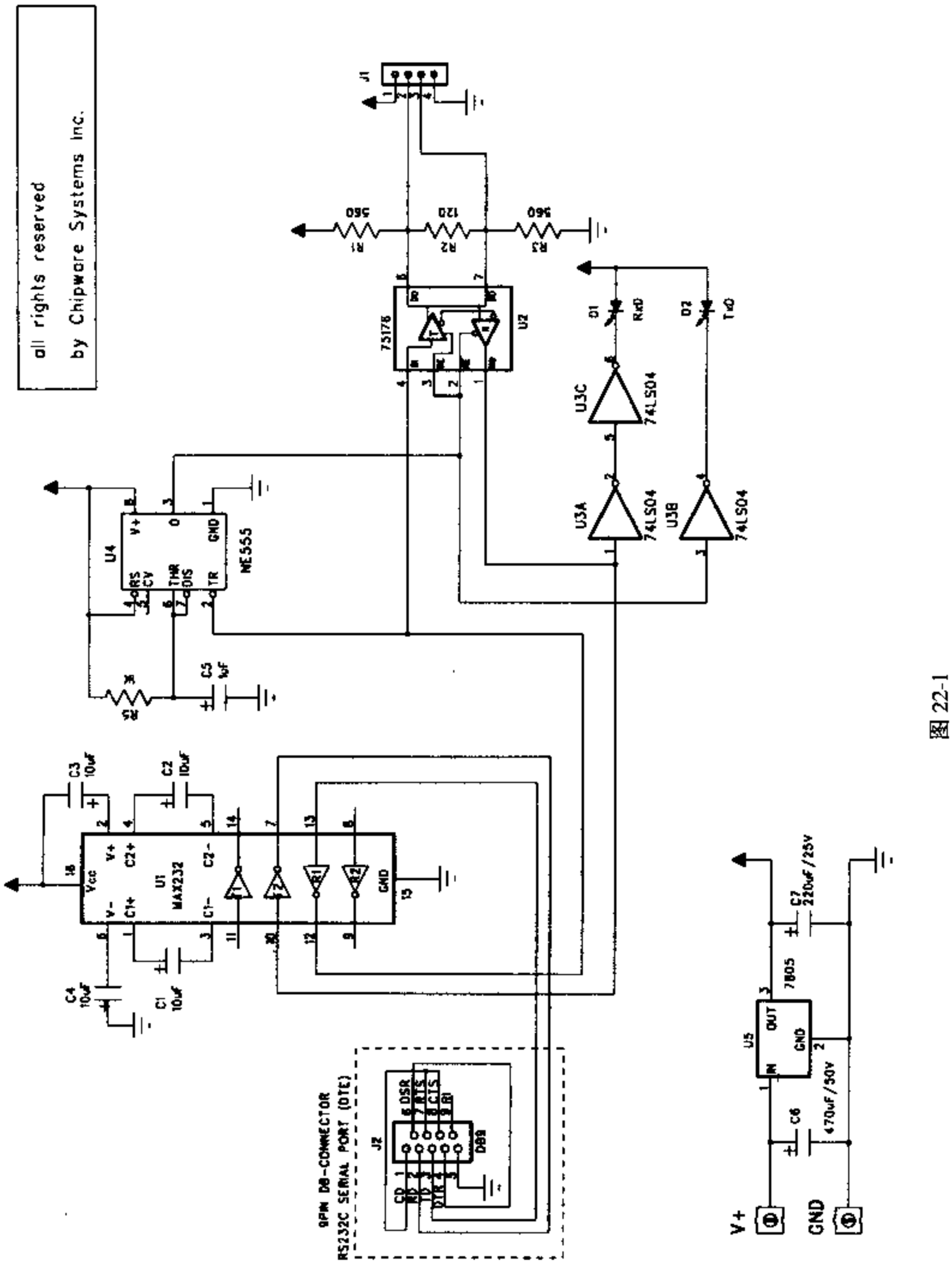


图 22-1

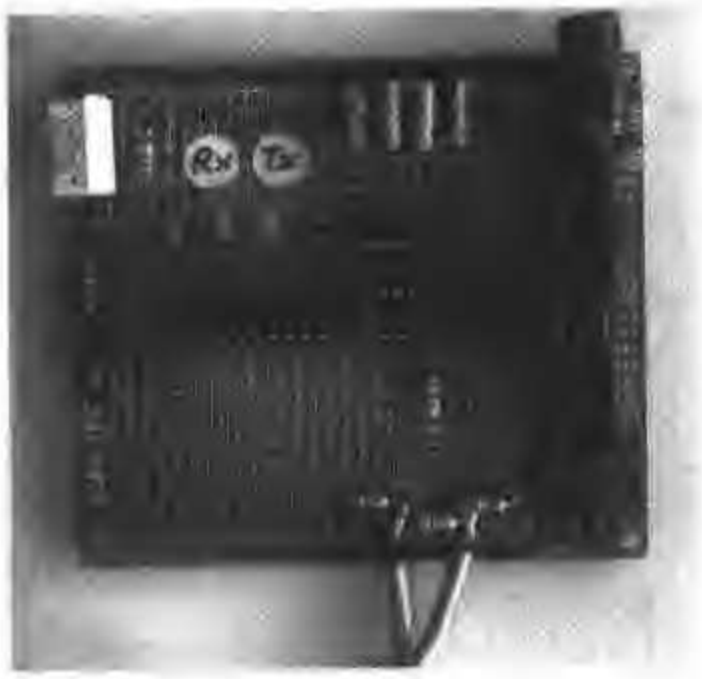


图 22-2

如果您已经完成了这个外接式的准位转换电路，接下来就要用 TURBO C 写试验程序，当然您也可以用其它语言写这个程序，测试一下送出的过程以及准位是否正确，试验时最佳的方法就是持续产生几个测试用的串行信号，然后用示波器观察输出的准位，顺便检查送出的串行数字码是否是对的？如果送出的源头不对，接收端也就不可能期待有所反应了。

程序 1 是我们所写第一个 TURBO C 程序，这个程序平常要让 PC 处于接受数据的模式，所以 75176 的 2 及 3 脚必须保持在逻辑 0 的状态，当要送出串行数据时，会先把 75176 的 DE 改成 1，让发送 75176 的驱动器启动，送完数据后才又将 DE 改变成 0。这些都是由 RS232C 上的 RTS 来掌握，所以我们必须要知道如何改变串行端口上 RTS 的状态。手边的数据都是 PC 上串行通信最原始的 IC：8250 的数据，虽然串行 IC 已经改朝换代数次了，除了加大数据输出入缓冲区的空间外，其它的控制方式几乎没做修改，所以我们就参照 8250 的数据在串行数据送出前，先对 8250 下指令把 RTS 线设成 1，当然这些设置及调整方式是否正确还要用示波器去做双重验证。在这个程序当中，我们安排每隔一秒钟才送出一组串行数据，这样子可以用来确定接收端是否收到正确的信息。

图 22-3 是 RTS 与送出数据串间的关系。

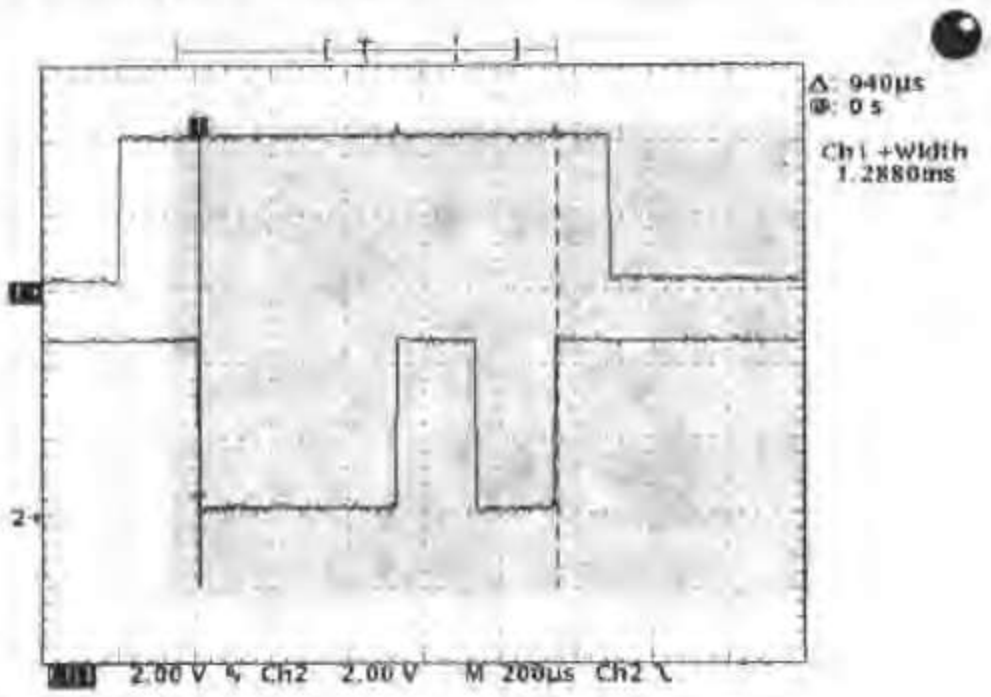


图 22-3

图 22-4 是 75176 送出的串行差动信号。

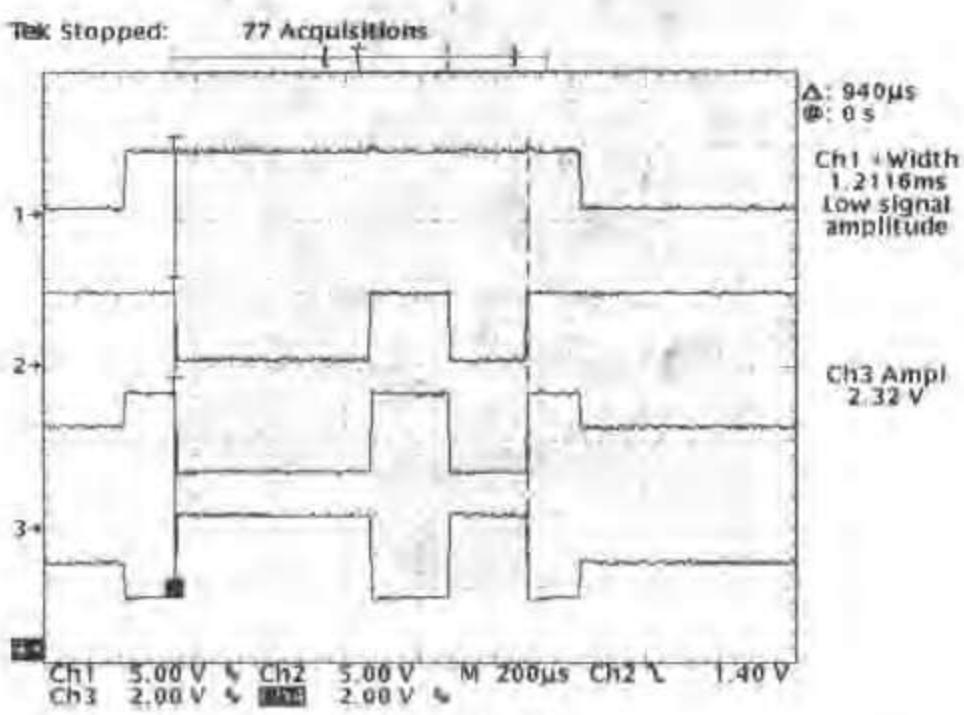


图 22-4

〔程序 1〕在 PC 端每隔一秒送出一个字符

此程序见光盘文件。

22-2 SLAVE 端 RS485 通信的写法

熟悉并确认了 RS485 串行的准位之后，接下来就要在 AT2051 学习板上试写一个 8051 小程序来接收 PC 送来的值，然后把发送端与接收端用 RS485 专用缆线连接起来。假使专用的 RS485 连接缆线一时无法购得时，可以改以一般的信号线或喇叭线替代，试验前我们还是要在发送与接收双方加上终端电阻，以便隔阻反射信号。如果 AT89C2051 收到的串行字符是 '0'（30H）时，就让板上的蜂鸣器叫一声。这个程序讲起来简单，转换成程序代码时要分别启动定时器及串行控制端口，动作也是一大串的，首先我们示范的是标准询问式（POLLING）的写法。唯有这种写法试验正确后，改成中断的写法才有意义。

图 22-5 是 RS422/485 专用的电缆线，内有两对隔离屏蔽的双绞线（Twisted Pair），外围另有隔离地网保护双绞线。

图 22-5

图 22-6 是接收端上所看到的 RS485 信号。

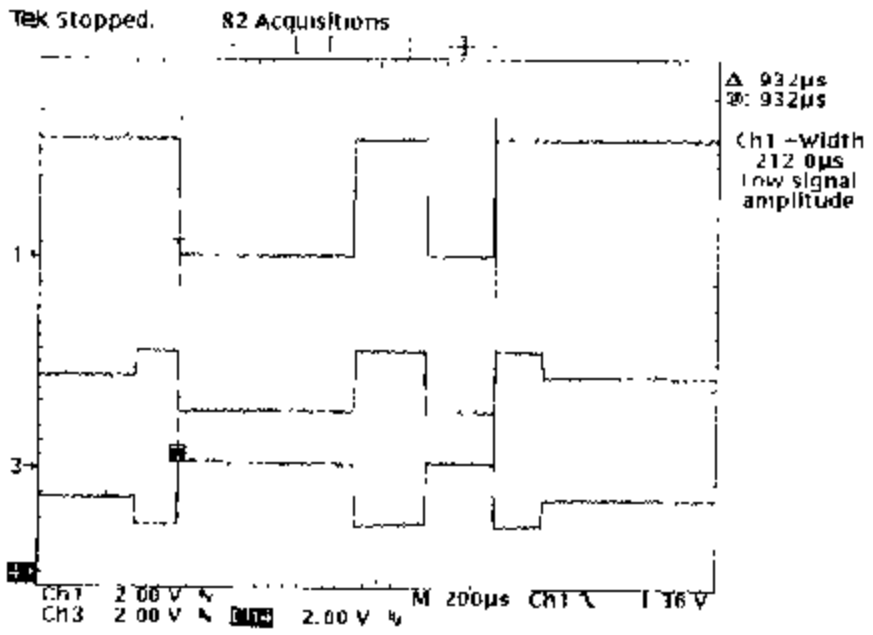


图 22-6

（程序 2）AT89C2051 学习训练板上的接收程序示范

此程序见光盘文件。

接收端的 AT89C2051 开机后就处于数据接收的模式，并随时把收到的数据丢到 P1 的 LED 端口上，并且对收到的数据进行分析，如果是 30H 时才将蜂鸣器启动半秒钟。这个程序主要在验证接收端的硬件及接线是正确的，实际的程序对接收数据分析的步骤应当更为完备才对。如果程序改为中断服务程序（Interrupt Service Routine）时，要设置与启动的条件更多了，以下是我们更进一步的接收端 ISR 写法。如果您对串行方面的设置仍觉得生疏时，请回头看《8051 单片机彻底研究基础篇》一书上对串行通信的详细说明。

（程序 3）AT89C2051 接收端的中断接收写法

此程序见光盘文件。

图 22-7 是 AT89C2051 收到正确的 ID 码后，100µs 后就开始回送数据。

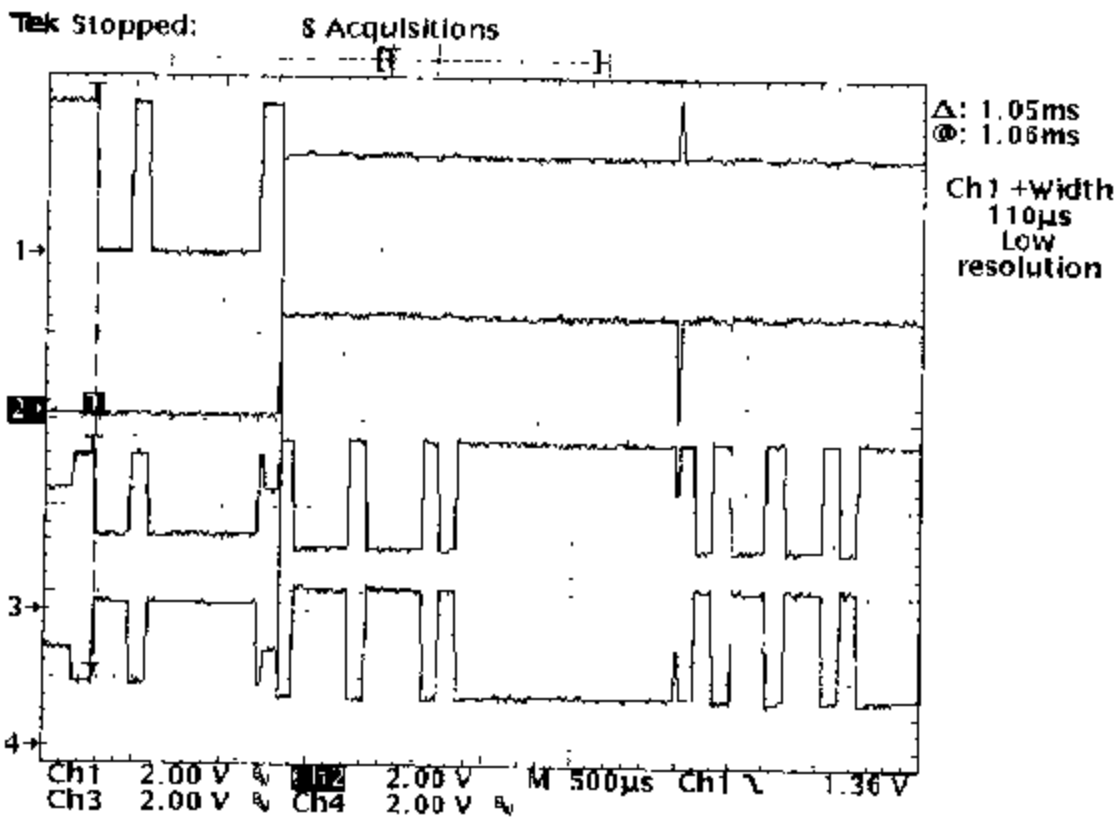


图 22-7

图 22-8 是 AT89C2051 共送回 12 字节的 ASCII 数据，总共花了约 27.5 ms 的时间。

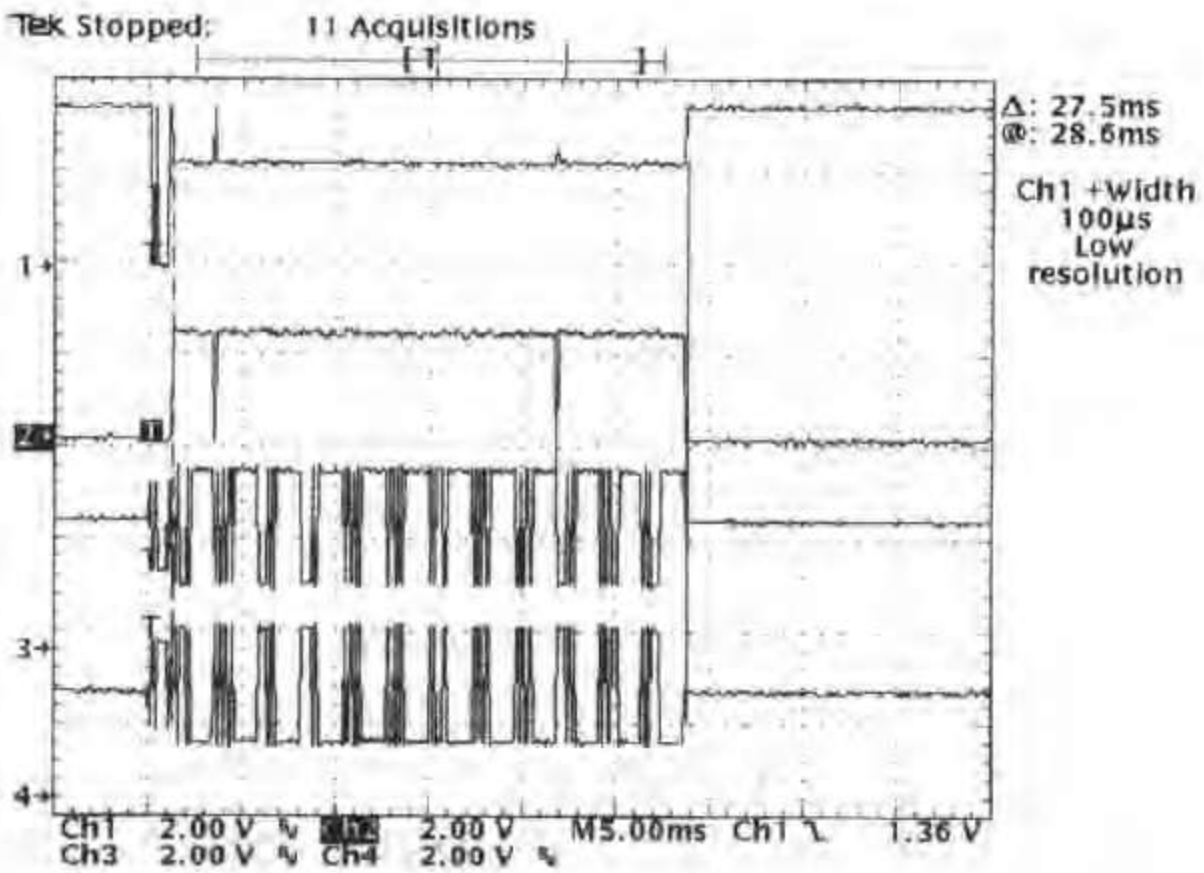


图 22-8

图 22-9 是双方速度提高到 19200bit/s 时，AT89C2051 依然反应正常。

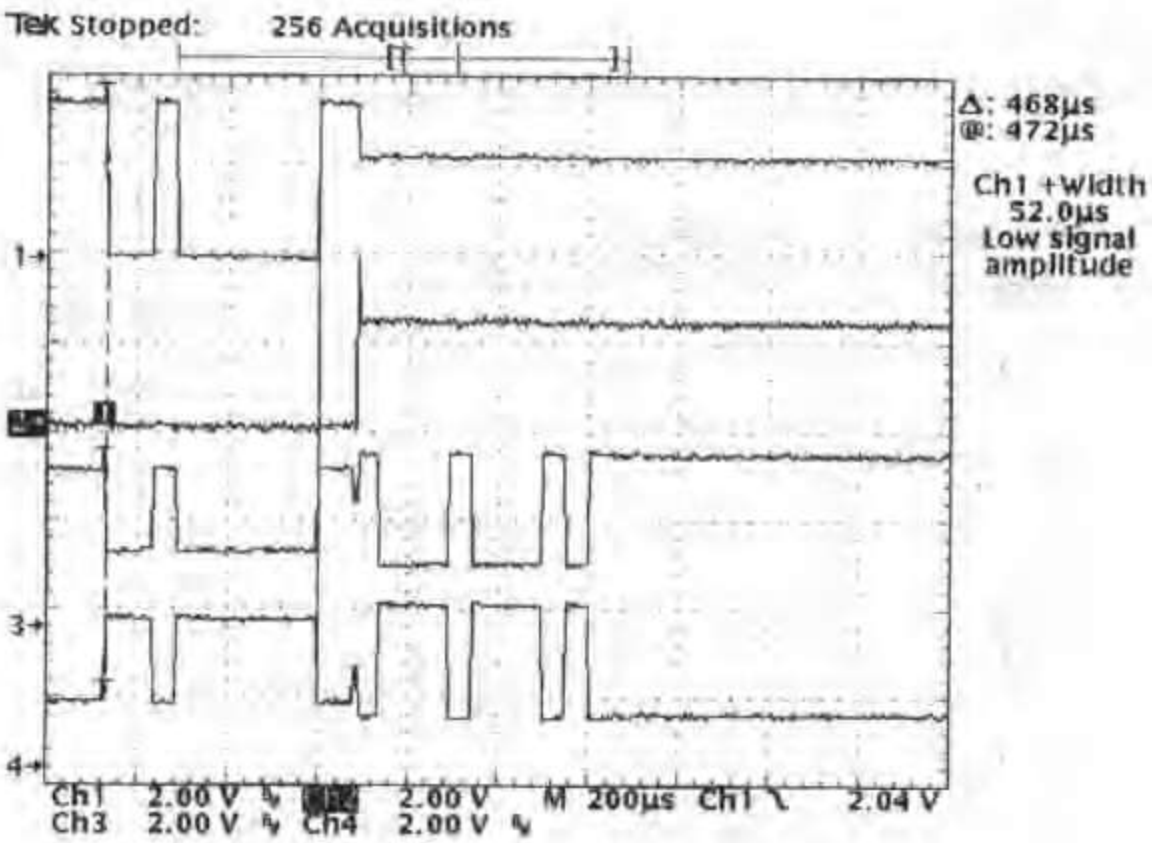


图 22-9

图 22-10 是 AT89C2051 串行通信 MODE1 的极限速度约 58K bit/s。

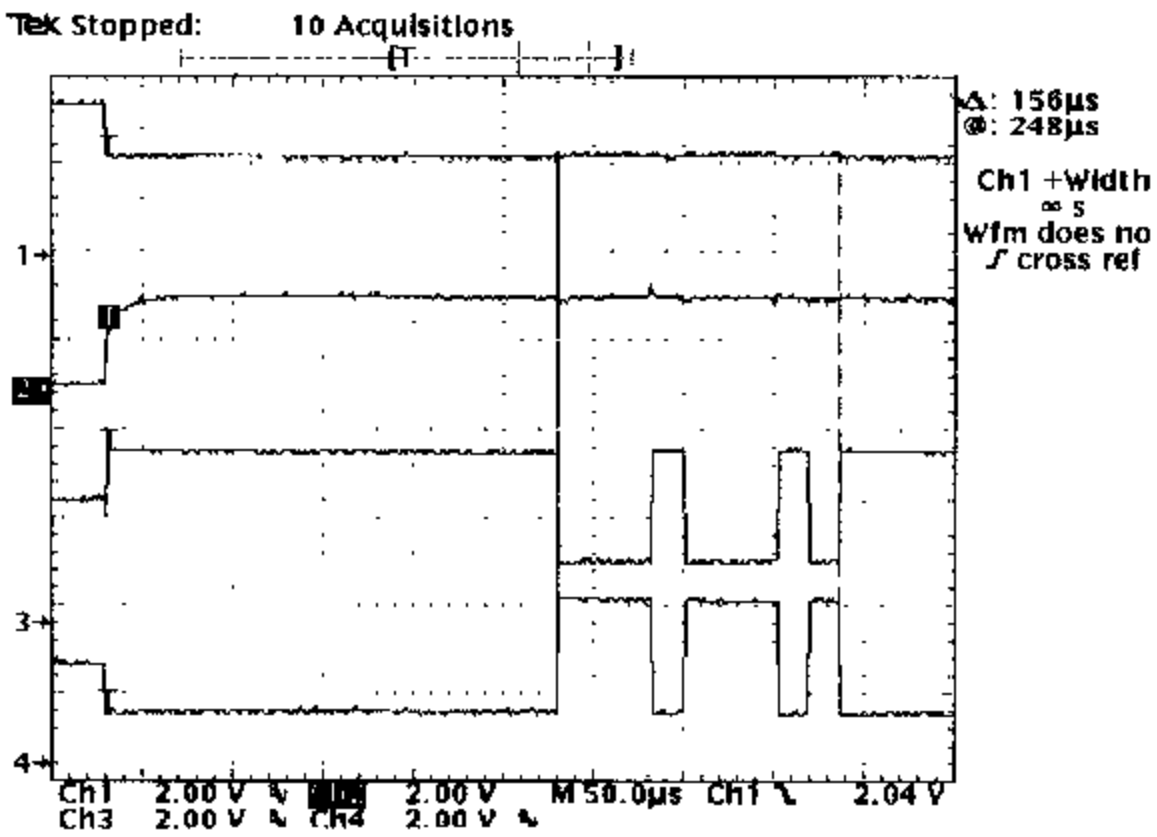


图 22-10

22-3　SLAVE 端的响应程序

如果收发双方联机都验证无误之后，可以开始动手规划整个传输在线的通信协议。我们先来尝试最简单的通信方式：由 PC 端送出 ID 码，然后由 ID 码正确的 AT89C2051 端回应一个 ASCII 的状态值，这个状态值约占 10 字节左右。我们知道 RS485 上传输的信号可分成 ID 码与信息数据，前者首先由主控者送出，当有符合该 ID 码的设备或仪器存在时，会立即回送该设备所量测到的值。由于通信协议不是很复杂，所以我们把 ASCII 码中的 00H-1FH 规划成 ID 码的部分，可显示的 ASCII 码则是 AT89C2051 训练器端回送的资料码，ID 码与数据码各有其存在空间，只要在中断服务程序中稍作判断即可分辨出来。我们将原来在 PC 端的程序再做修改，把原先送出的固定码改成由 00H 变化到 1FH，然后随时读回其它 SLAVE 端送回的数据并显示在屏幕上，如果您有两台以上的 AT89C2051 训练板时，就可以把这些控制板都用 RS485 的方式并连在一起，随时得到各个控制板上 DIPSW 的状态值了。

在做这个串行试验时，AT89C2051 训练板上必须有一个 ID 码的设置，由于该控制板上已经有一个 DIP SW，所以我们特地安排该 DIP SW 中的 1~4 为其 ID 值，而 DIP SW 的 5-8 则当成回送的状态值，回送前当然要转换成 ASCII 码后，再由 75176 的发送端送出。AT89C2051 端要送出数据前要先把 P3.2 设成 1，以便让 75176 的发送端实现。

〔程序 4〕PC 端的状态查询程序

此程序见光盘文件。

〔程序 5〕AT89C2051 的状态回送程序

此程序见光盘文件。

图 22-11 是 PC 端每分钟去取“智能型温湿度计”后所绘出的温湿度变化图。

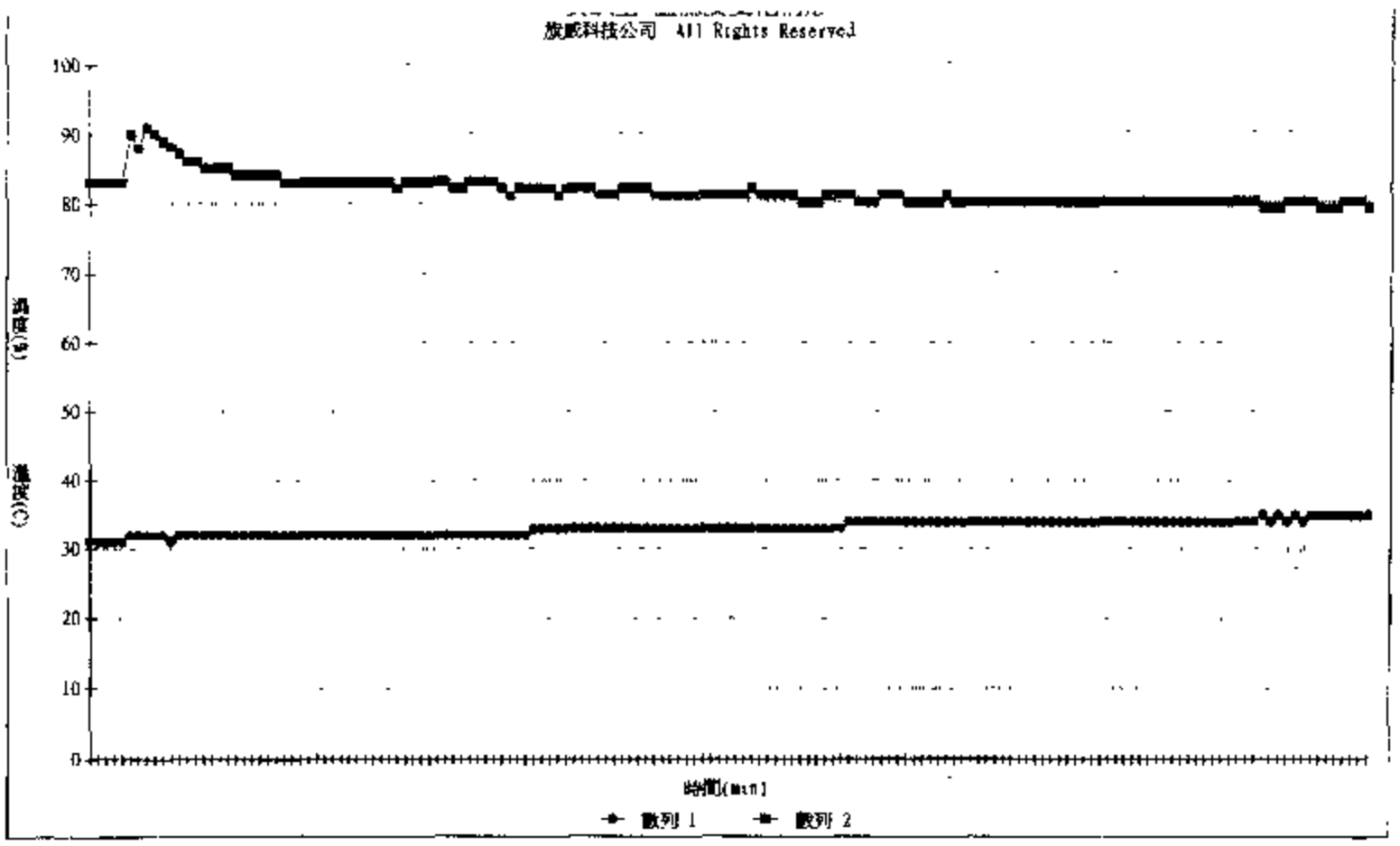


图 22-11

22-4 RS485 信号准位的观察与分析

RS485 联机主要的只有两条通信线而已，如果确定送的信号是正常的，而且接收的电路及程序也是正确的，若双方仍无法联机时，此时就要检查并接的 RS485 线是否颠倒了。更好的方法是直接用示波器观察 PC 送出的信号以及接收端 75176 输出的信号。这些必要的手续可以使我们马上就分离出是联机的问题或是程序的问题了。我们一直提到 Intel 当初在 8051 的串行通信上做了许多努力，可是在一般的 RS232 通信上却无法利用这些优点，或许改用 RS485 就可以充分地使用诸多项优点。以下就是我们几个额外测试的重点。不过受到篇幅的限制，我们这里仅公布测试后的结果。

测试点 1：将串行传输速度提升到 19200bit/s 是否可行。

测试点 2：将传输速度加到 8051 的最高速度，75176 受得了吗？

测试点 3：与我们最近刚完成开发的新产品“智能型温度计”的联机测试。

22-5 本章使用软件

本章使用软件如下：

（1）Borland Turbo C V2.0。

（2）2500AD 8051 Assembler and C compiler。

22-6 本章使用硬件

本章使用硬件如下：

（1）75176 485 收发器。
（2）AT89C2051 学习板。
（3）RS485 专用隔离双绞线。
（4）RS232 状态监视器。
（5）Tektronix 数字示波器。

22-7　相关信息网站

您可经由下列公司、网站取得更进一步的信息：
http://www.chipware.com.tw：查询 8051 控制板相关信息。
http://www.ns.com：查询 75176 IC 相关信息。
http://www.rs232.com.tw：查询各类 IC 的信息。
http://www.tek.com：查询数字式示波器信息。

第 23 章

RS485 通信接口彻底研究（三）

延续前一章的内容，“旗威科技”设计了一个智能型温度计，除了可以用来测量温度以外，还可以用程序来记录一段长时间的温度变化，进一步可用来做温度控制。很多有温差效应的化学实验、温室的温度调整、环境的监控等都可以通过此种方法来管理，看看我们是怎么做到的。

23-1 智能型温度计

智能型温度计是由 AT89C2051 的芯片作为核心，搭配温度感应组件所制成，其中含有一个二位数的显示器可以在独立状态下接上电源进行温度的测量，比较特殊的是该控制板具有 RS485 的通信端口，它可以跟任何有 RS485 接口的装置或计算机联机。由于 PC 计算机并没有 RS485 接口，我们这里所做的联机示范是 PC 通过一个 USB 转 RS485 的转接装置，将其联机到智能型温度计上，便可做温度的实时监控。连接方法见图 23-1。使用 USB 的好处是为了方便安装、加快数据的传输速度，并且可以配合计算机系统即插即用的功能，减少设置上的困扰。您准备好跟我们一起动手做了吗？

图 23-1

◆ 硬件的准备

首先您先要准备的是一个由旗威科技所开发的智能型温度计，接着是一条 USB 公对公的传输线、一个温度计与计算机联机用的 USB 转接盒及一个 6V-12V 的直流电源，见图 23-2。我们强烈建议使用蓄电池或是直流电源供应器，可以确保电源的稳定供应，假如您的电源输出电压不足 6V，很可能会导致此温度计无法正常运作。

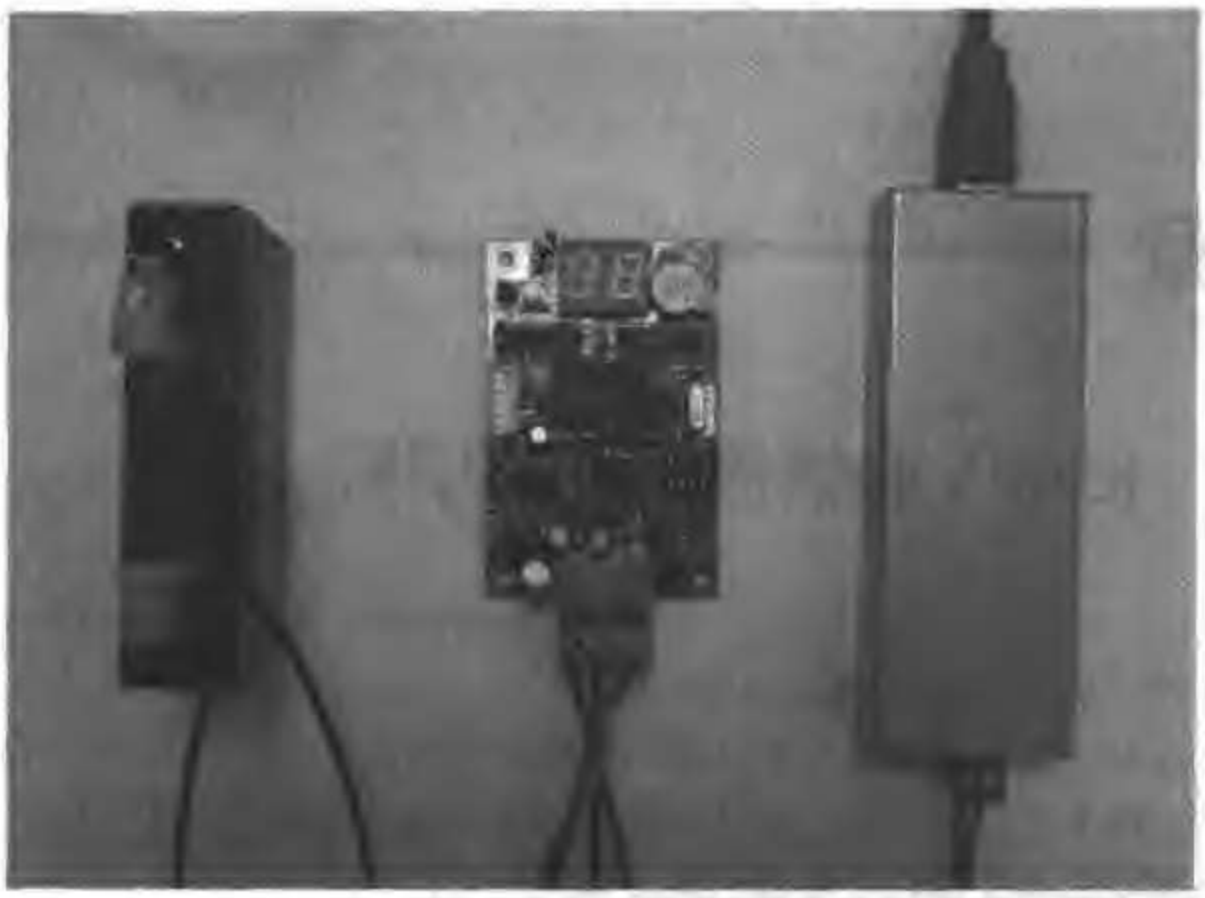


图 23-2

注：图 23-2 是智能型温度计及其相关设备，由左到右分别是电池、温度计与 RS485 到 USB 转换器。

◆ USB 装置驱动程序的安装

首先要安装 USB 的驱动程序，如果您的系统使用的是 Windows 98 系列的版本，那么可以利用“开始”→“设置”→“控制面板”命令，在打开的“控制面板”窗口中双击“添加新硬件”图标，会出现一个窗口，见图 23-3，请选择“否，希望从列表中选择硬件”→“连接端口”来做安装，并“指定位置”到存放驱动程序的路径。在安装过程中会出现“请放入 98 光盘”的字样，只要将光盘置入并继续安装，不一会儿就安装完成了。

图 23-3

如果您所用的系统是 Windows ME/NT/2000/XP 系列之版本，可能会自动检测完成设置，不行时仍然需要用驱动程序来做装置的登录，只不过在较新的版本中已经将 USB 列为即插即

用的装置，所以不需再置入安装磁盘。安装完成后我们到系统的装置管理员当中，查看连接端口的选项时，我们会发现一个新的连接端口（COM3）见图 23-4，本装置会因系统的不同而产生不同的连接端口序号，如您使用的计算机上并没有串行连接端口装置，则产生的连接端口序号就会变成（COM1），这个序列号码会影响到您在控制程序上一些参数的设置，这在稍后的部分会提到。

图 23-4

注：图 23-4 是完装完成后，检查连接端口 COM 装置是否已经启动。

23-2　VB 控制程序的产生

一切就绪后，加上电源后温度计会开始每秒一响滴滴滴地叫着，而显示器上也会显示现在的温度值，而这个温度值的计算方式，在较前面的章节里我们已经描述过，在此不再赘述。然而智能型温度计应用至此，似乎觉得好像少了些什么，没错，我们还未加入温度的监控，所以这个温度计到此只能和一般的温度计功能相似，并没有特别的地方，然而我们要以什么方式来对这些测量值做管理呢？

现在程序语言的百家争鸣似乎已经渐渐趋于整合，各家的语法也纷纷相互支持，然而面对窗口时代的来临，及广大使用者对于操作界面选择的趋势来看，Visual Basic 似乎是最易上手也最容易设计出适合窗口管理界面的软件，当然，Visual C、Java 等也都是不错的选择，但对于一个习惯汇编语言却未曾设计窗口程序的工程师而言，Visual Basic 不论在学习上所花的时间，或是界面的设计上，都相较其他一般软件来得符合经济效益，因此接下来控制的部分我们将以 Visual Basic 为主体来做介绍。

◆ 安装 Microsoft Visual Basic 6.0 专业版

我们所使用的软件是 Microsoft Visual Basic 6.0 专业版（以下简称 VB6），原版的包装里

会有一张 VB6 的主程序及一张 MSDN for VB6，我们的作法是两片都完成安装，一来方便程序设计时遇到困难可以实时查阅，二来可以顺便学习指令的语法和应用限制，加速我们对 VB6 的熟悉与使用。不管您是初学者或是经验老练的程序工程师，我们都建议您这么做！而安装的过程，您可以参考一些 VB6 的相关书籍，在此我们仅做应用上的介绍。

◆ 窗口界面的设计

打开 VB6 之后，会出现一个安装向导，我们选择“标准 exe”并将它打开，这时会出现一个“工程 1（工程 1）”的窗口，图 23-5 便是我们的初始画面。前面提到我们装置是由串行端口联机的，因此我们必须引用一个叫 Microsoft Comm Control（简称 MSCOMM）的控制组件，您可以在“工程”→“部件”→“控件”中找到它，然后将它单击并确定后，在主窗口会出现一个电话的按钮，这样便算引用成功。

图 23-5

注：图 23-5 把 MSCOMM 对象导入到我们的 VB 程序当中。

紧接着我们开始设计窗口，通过工具栏中的 Command Button、Label、Textbox、Timer、及 MSComm 等按钮初步规划出我们的窗口，我们来看看关于它们的一些设置：

Command1：将属性窗口中的 Caption 字符串值改成 EXIT，这个按钮是为了联机中止所做的设计。

Command2：将属性窗口中的 Caption 字符串值改成 START，这个按钮是为了开始联机所做的设计。

Timer1：用来计时的工具，在程序中会做设置。

Text1：用来显示温度测量值的文字框，在程序中会做设置。

Label1：用来显示联机的状态，在程序中会做设置。

MSComm1：用来取得测量值最关键的组件，原则上也是利用程序设置即可。

◆ 取得温度值

想要做温度的监控，首先便是取得温度计上的温度值，我们利用 VB6 中所提供的 MSComm 指令与温度计做联机，以下是取得温度值的范例：

```
Private Sub Command1_Click()                    'EXIT 按钮
  MSComm1.PortOpen = False                      '将序列连接端口关闭
  End                                           '强制结束程序
End Sub

Private Sub Command2_Click()                    'START 按钮
Dim buffer$                                     '定义返回字符串为 buffer$

   Timer1.Interval = 1000                       '将时间设置为 1000ms，也就是一秒
   Timer1.Enabled = True                        '每计时一秒会将计时器打开一次

retry:                                          '设置返回测量值的函数
   buffer$ = ""
   Do
      DoEvents
      buffer$ = buffer$ & MSComm1.Input
      If (Len(buffer$) > 0) Then                '当产生返回字符时
          Label1 = "数据更新中"                 '让 Label 显示"数据更新中"的信息
      Else                                      '如果没有返回字符时
          Label1 = "等待中"                     '让 Label 显示"等待中"的信息
      End If
   Loop Until Len(buffer$) > 10                 '确认返回完整后才停止返回
      Text1 = buffer$                           '将返回值显示在 Text1 的文字框中
GoTo retry
End Sub

Private Sub Form_Load()                         '储存输入字符串的暂存区
    MSComm1.CommPort = 3                        '设置与 COM3 连线
    MSComm1.Settings = "9600,N,8,1"
                    '联机速度 9600 baud、无同位检查、8 数据位、停止位 1
    MSComm1.InputLen = 0   '告诉控制项当使用 Input 时，清除暂存区之值
    If (MSComm1.PortOpen = False) Then MSComm1.PortOpen = True '打开序列连接端口
    Text1 = ""          '等待数据传回到序列连接端口，先清除文字框的内容
End Sub

Private Sub Timer1_Timer()    '计时器的动作设置
```

```
    MSComm1.Output = "1"        '温度计的 ID 默认为 1
                                '启动计时器时，通知 AT89C4051 送回温度值
End Sub
```

到此为止，我们的前置步骤算是告一段落，先执行程序让他运行一下，在窗口上按 START 键便可在屏幕上见到现在的温度值，测量值应该到小数点以下一位数，看看得到的结果是不是我们所想要的？假如答案是肯定的，恭喜您，您已经成功取得温度计上的温度值了。

◆ 温度值变化的监控

学会了取得温度值的方法，我们接下来所要做的工作，便是如何去记录我们的温度值，并将其转化为图表，方便我们观察在不同时间、不同环境下温度的变化值，不必大老远地拿着一本笔记簿，走到每一个温度计旁把其上的温度值一一写下来。我们只要利用联机和简单的程序，便可以边听音乐、边喝咖啡，坐在计算机前把所有的数据全都搜集并记录下来，当然这需要多花一点时间，但也是本章节的关键阶段，耐心地看下去吧！

要画图表，我们首先会联想到的工具是 Microsoft Excel，Excel 提供了很多计算上及绘制图表上的强大功能，假如这些功能要全靠写程序的方式去自行绘制，那会是相当大的工程，因此我们省略了这个烦人的步骤，以 Excel 来取代。不过真正的挑战现在才要开始。

由于 Excel 是属于 Microsoft Office 的应用程序之一，所以要在 VB6 使用它时，也是要经过引用的程序才能使用它，更重要的是：你必须先安装 Excel 才有办法引用它！在“项目”“设置引用项目”的选单中，找到 Microsoft Excel Object Library 的选项，其中的数字为其版本的设置，若是 Excel 97 则出现 8.0，若是 Excel 2000 则出现 9.0，若是 Excel 2002 则出现 10.0。

引用之后，我们就可以开始进行一些图表的控制，不再只是单纯的从文字框看到这些东西，在这阶段是整个控制过程中比较难的部分，我们尝试过很多方法让 VB6 和 Excel 的表格互通，但是都没有很好的结果，最后以最笨但也是最实用的方法，就是先将数据记录到记事本（Notepad）中，再将记事本的数据指定到 Excel 的栏位上，这样的缺点是数据没有办法实时更新，但可以确定所有监控的数据完整地被记录下来，万一不小心因指定字段发生错误，还可以回去检查记事本中的数据做比对。或者您有更好的意见跟作法，也欢迎您和我们讨论交流，我们会将它公布在我们的网页上，让更多需要的人可以欣赏您宝贵的意见。

经过一番折腾，我们的工作就要完工了，接下来把写好的程序打开，我们可以看到折线图上温度和时间的关系，看看是否得到我们所要的结果。

图 23-6 是 Excel 下的温度值与图表。

以下程序为取得温度值并将数值送至 Excel 的表格中的范例：

```
Option Explicit
Dim test_byte As Integer
Dim a(1000) As String

Private Sub Command1_Click()            'EXIT 按钮
 MSComm1.PortOpen = False               '将序列连接端口关闭
 End                                    '强制结束程序
End Sub
```

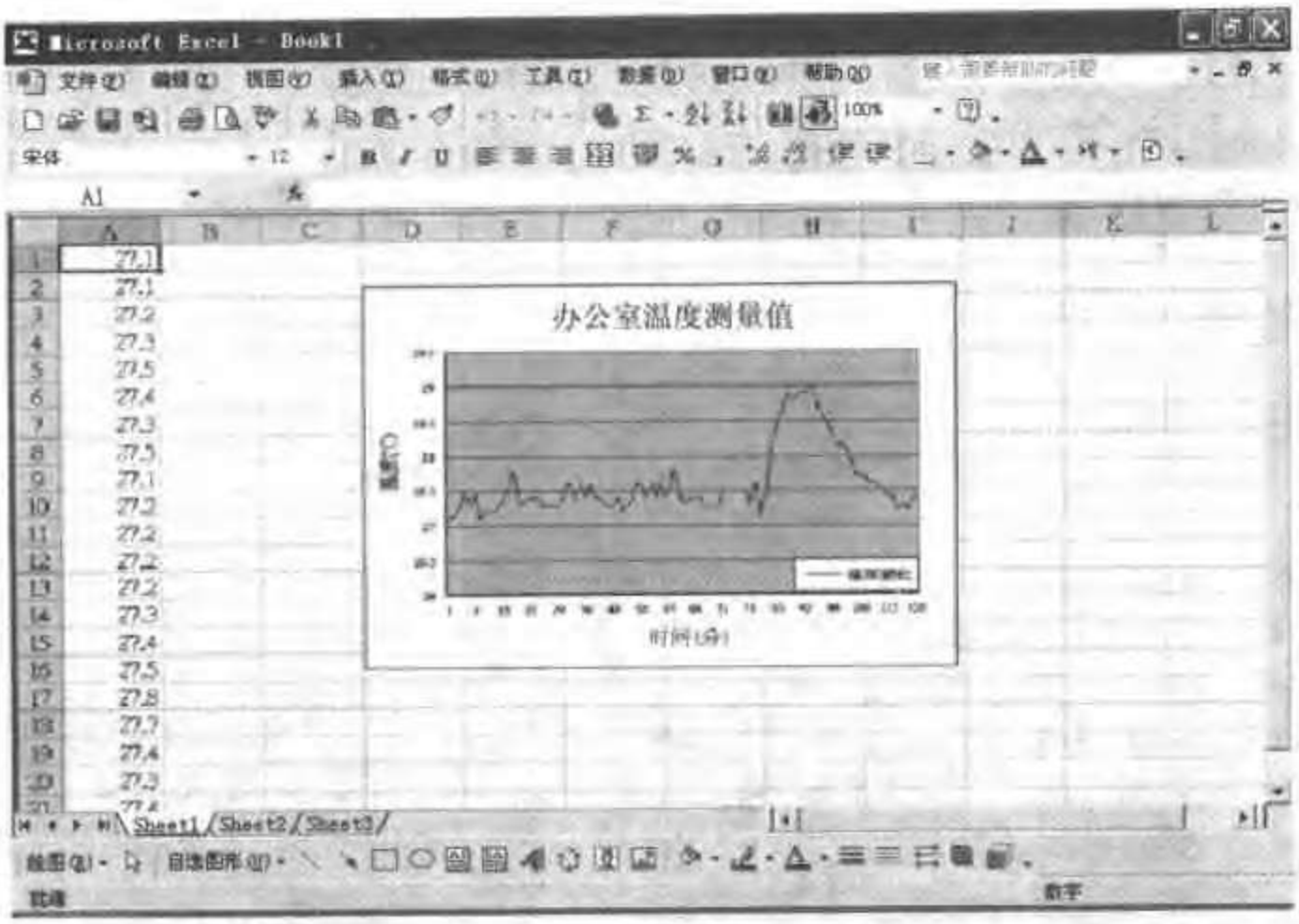


图 23-6

```
Private Sub Command2_Click()                    'START 按钮
Dim buffer$                                     '定义返回字符串为 buffer$
Dim k
Dim i
  Timer1.Interval = 1000                        '将时间设置为 1000ms，也就是一秒
  Timer1.Enabled = True                         '每计时一秒会将计时器打开一次
For i=1 to 100
retry:                                          '设置返回测量值的函数
    buffer$ = ""
  Do
    DoEvents
    buffer$ = buffer$ & MSComm1.Input
    If (Len(buffer$) > 0) Then                  '当产生返回字符时
        Label1 = "数据更新中"                   '让 Label 显示"数据更新中"的信息
    Else                                        '如果没有返回字符时
        Label1 = "等待中"                       '让 Label 显示"等待中"的信息
    End If
    Loop Until Len(buffer$) > 10                '确认返回完整后才停止返回
    Text1 = buffer$                             '将返回值显示在 Text1 的文字框中
    a(i) = Mid(buffer$, 4, 5)                   '取字符串的部份内容,从第 4 个位置开始取 5 位
 Next i
End Sub

Public Sub Command3_Click()
  Dim test As Excel.Application                 '定义 test 的型态为 Excel 试算表
```

```
  Dim i
  Set test = Excel.Application
    test.Visible = True
    test.Workbooks.Add                    '打开一个新的 Excel 工作表
  For i = 1 To 100
      test.Cells(i, 1) = a(i)             '将接收到的温度值送到 Excel 的第一行中
  Next i
  Set test = Nothing
End Sub

Private Sub Form_Load()
  ' 储存输入字符串的暂存区
  ' 使用 COM4.
  MSComm1.CommPort = 4
  ' 连线速度 9600 baud、无同位检查、数据位 8，停止位 1
  'MSComm1.Settings = "9600,N,8,1"
  ' 告诉控制项当使用 Input 时，读取整个暂存区
  MSComm1.InputLen = 0
  ' 打开序列连接端口
  If (MSComm1.PortOpen = False) Then MSComm1.PortOpen = True
  ' 等待数据传回到序列连接端口
  Text1 = ""
End Sub

Private Sub Timer1_Timer()
    MSComm1.Output = "1"      'ID='1'
    '通知 AT2051 送回温度值
End Sub
```

23-3 温度测量实验的问题解答

问题一：我想要同时检测很多不同地方的温度时，要如何让很多的温度计同时联机？最多可以同时联机多少个？又如何同时取得这些数据？

解答：关于这个问题，我们可以通过温度板上的 EEPROM 来做设置或修改，在前面的程序代码中提到温度计的 ID 设为 1，如果我们将 ID 值做修改，最多可以提供 32 个温度计在同一个 COM 端口上，而正确的数目要视联机的硬件而定，并没有一定的上限，如果是想要使用不同的 COM 端口来联机，原则上 COM 端口所做的设置最高可以提供到 16 个，如果超过 16 个端口，系统便无法判断您的装置而导致不能驱动使用。至于取得数据，只需将程序代码的 ID 改成其他的数字便可以同步取得这些数据了。

问题二：测量到的温度值好像有点误差，要如何修正？

解答：这个问题牵扯到感温组件的灵敏度跟误差值，本温度计所使用的感温组件最大误

差值为 0.7 度，修正的方式有两种：第一种是改变 AT89C2051 内的温度设置，直接改变温度的计算公式，并将修正值存入 EEPROM 内，这是高级仪器的调整方法；第二种是利用 Excel 的试算能力，将误差修正的函数置入，使得温度通过计算而得到修正。不过第二种作法无法改变温度计上的显示值，仅能改变计算机上所取得的数值。

问题三：能不能对 VB6 和 Excel 的沟通部分多做一些介绍？如果想要得到温度测量值的实时更新图表，应该从哪部分着手？

解答：关于这方面，以我们的经验，最好的方法就是要让 Excel 能够直接运用 VB6 的 MSComm 指令，而这个过程是需要花很大的篇幅来做说明的，最直接的作法就是写一个 Windows API 的外挂给 Excel 使用，这在 VB 与 Windows API 的相关书籍中会有提到，也可以在 VBA 的相关书籍找到一些来龙去脉。

问题四：为什么我所测量的温度值特别低，而且显示器上的灯号有跳动的情形？我在计算机上用程序所截取的数值和温度计完全一样，这是不是 AT89C2051 芯片的控制程序故障了？

解答：请您再检查一下显示器是否显示正常，除了显示数据外，亮度是否有异样的状况，通常遇到这样的情况，我们的经验是：先检查电源供应是否正常，换一个电压较高的电源，或将电池充电后再试试看。如果依旧没法恢复正常，再检查一下感温组件是否有正常的温度变化，排除了这些因素后还是没办法改善的话，那么芯片可能真的故障了，不过这样的可能性并不高。

问题五：文章中所提到的程序范例，我将测量的时间间隔设置为 100ms，可是文字框所显示的字符串为什么不太正常？情况严重一点还会造成程序死机，这是什么原因？

解答：这跟芯片所设置的数据传送速度有关，因为芯片本身设置每个字符的返回时间是 10ms，而返回的讯号总共会有 11 个字符（例 T1=+27.0C），其中有两个字符是 ASCII 的指令码度（CR 与 LF 码）不会显示出来，所以整个返回所需要的反应时间最少要 110ms，若忽略计算机本身从输出到返回的这一段时间差，最少也都必须给这个程序有 110ms 以上的反应时间。所以当您的时间间隔设置比 110ms 小的时候，那么程序执行后所得到的结果是绝对会出问题的，严重一点会导致系统无法跳出程序的循环，造成程序死机，这是正常会发生的状况。另外，计算机本身的速度也有相对的关系，我们尝试在 CPU 为 AMD Duron 750M Hz 及 AMD Duron 1G Hz 的不同计算机上测试，而其他的外部设备完全相同的情况下，在 Dorun 750M 上所需的时间间隔最少需 122 ms，但在 Dorun 1G 上却只需要 111ms，当然这只是个参考值，不过我们却可以发现在芯片控制的设计上，这一段时间差是不允许被忽略的，否则看似正常无误的程序却往往会导致不正常的结果。关于这方面的深入探讨，因为牵扯到很多其他的专业知识，在我们正着手的新书中会有更详细的介绍，如果您对这方面有兴趣，欢迎您跟我们一起讨论交流，我们会将经验无私地提供您作为参考，相对地我们也有个小小的请求：就是将您的问题公布在我们的网站上，让更多有兴趣的人一起加入这个行列。

图 23-7 是内建 RS485 接口的智能型温度计，所有的制作与程序说明已在《8051 单片机彻底研究基础篇》一书中详述。

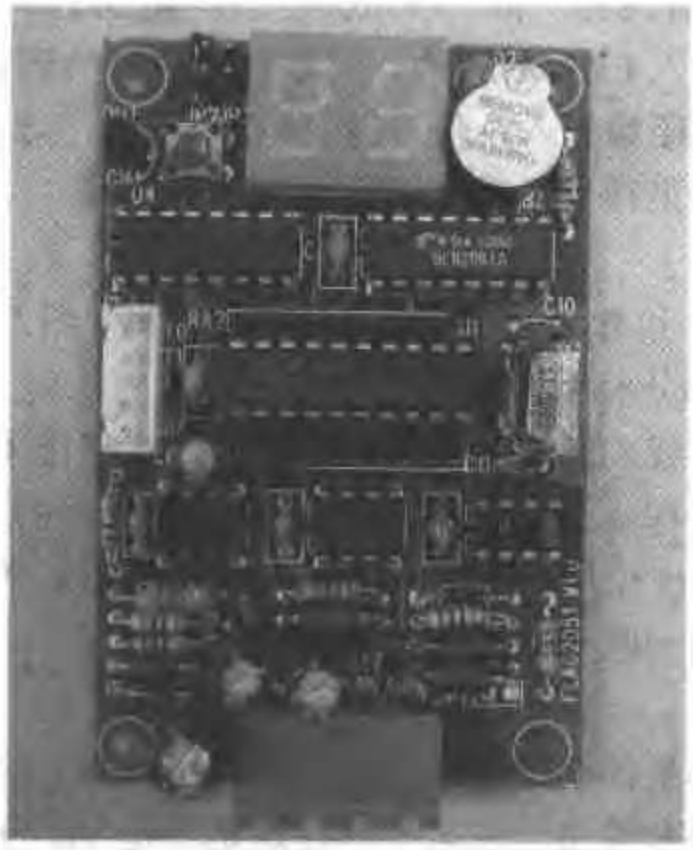


图 23-7

图 23-8 是 USB 转 RS485 接口。

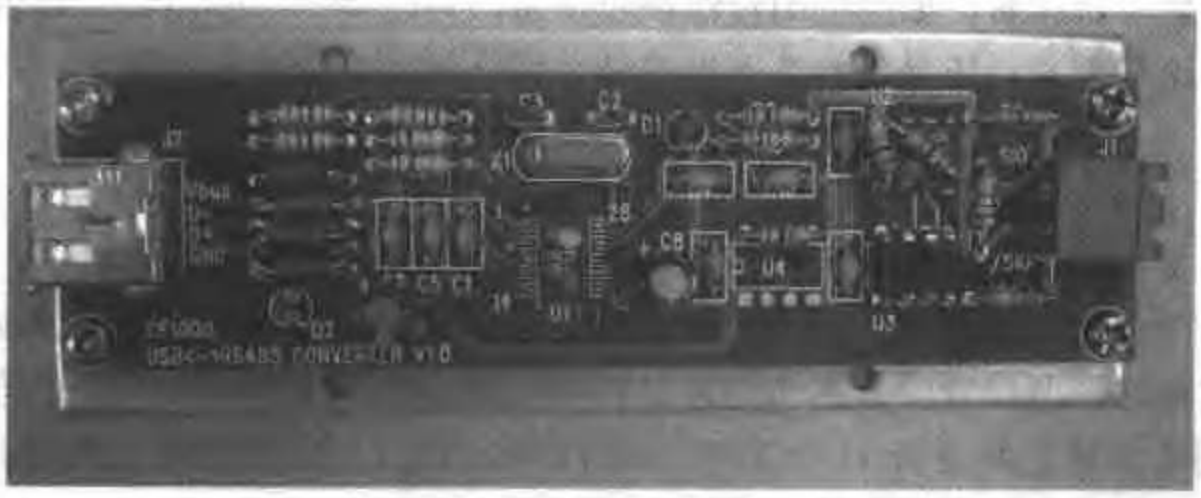


图 23-8

23-4 本章使用软件

本章使用软件如下：

（1）Microsoft Visual Basic 6.0。

（2）Microsoft Excel。

（3）2500AD 8051 Assembler。

（4）USB 驱动程序。

23-5 本章使用硬件

本章使用硬件如下：

（1）“旗威科技”的 AT2051 智能型温度计。

（2）USB 公对公的传输线。

（3）“旗威科技”的 USB 转 RS485 转接盒。

（4）6V-12V 的直流电源或蓄电池。

23-6　相关信息网站

您可经由下列公司、网站取得更进一步的信息：

http://www.chipware.com.tw：查询 AT2051 温度控制板相关信息及示范的 VB 程序。

http://www.idrc.com.tw：查询 AC 功率表与高功率直流电源等信息。

http://www.agilent.com：查询高级测量仪器与设备等信息。

http://www.microsoft.com：查询 VB6 及 VB.Net 相关信息。

附　录

不一样的书给你不一样的感觉，请查阅经过我们妥善整理的 8051 相关资料，本书的附录仍颇有参考的价值。

附录 A　ASCII 表

	0	1	2	3	4	5	6	7
0	NUL 00 0	DEL 10 16	SP 2D 32	0 30 48	@ 40 64	P 50 80	' 80 96	p 70 112
1	SOH 01 1	DC1 11 17	! 21 33	1 31 49	A 41 65	Q 51 81	a 61 97	q 71 113
2	STX 02 2	DC2 12 18	" 22 34	2 32 50	B 42 66	R 52 82	b 62 98	r 72 114
3	ETX 03 3	DC3 13 19	# 23 35	3 33 51	C 43 67	S 53 83	c 63 99	s 73 115
4	EOT 04 4	DC4 14 20	$ 24 36	4 34 52	D 44 68	T 54 84	d 64 100	t 74 116
5	ENQ 05 5	NAK 15 21	% 25 37	5 35 53	E 45 69	U 55 85	e 65 101	u 75 117
6	ACK 06 6	SYN 16 22	& 26 38	6 36 54	F 46 70	V 56 86	f 66 102	v 76 118
7	BEL 07 7	ETB 17 23	' 27 39	7 37 55	G 47 71	W 57 87	g 67 103	w 77 119
8	BS 08 8	CAN 18 24	(28 40	8 38 56	H 48 72	X 58 88	h 68 104	x 78 120
9	HT 09 9	EM 19 25	) 29 41	9 39 57	I 49 73	Y 59 89	i 69 105	y 79 121
A	LF 0A 10	SUB 1A 26	* 2A 42	: 3A 58	J 4A 74	Z 5A 90	j 6A 106	z 80 122
B	VT 0B 11	ESC 1B 27	+ 2B 43	; 3B 59	K 4B 75	[5B 91	k 6B 107	{ 7A 123
C	FF 0C 12	FS 1C 28	, 2C 44	< 3C 60	L 4C 76	\ 5C 92	l 6C 108	| 7B 124
D	CR 0D 13	GS 1D 29	- 2D 45	= 3D 61	M 4D 77	] 5D 93	m 6D 109	} 7C 125
E	SO 0E 14	RS 1E 30	. 2E 46	> 3E 62	N 4E 78	^ 5E 94	n 6E 110	~ 7D 126
F	SI 0F 15	US 1F 31	/ 2F 47	? 3F 63	O 4F 79	_ 5F 95	o 6F 111	DEL 7F 127

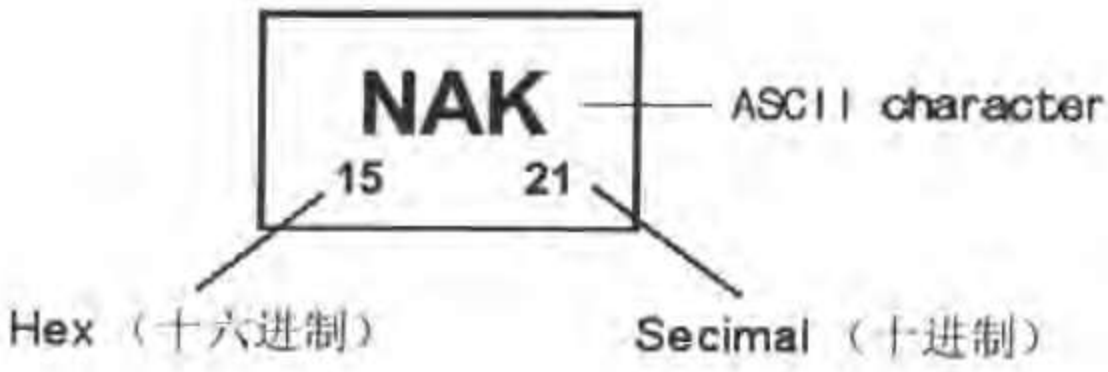
高位元

低位元

Example

NAK

HEX　DECIMAL



如何利用本表格

如果我们收到了 30H 这个十六进制码时，代表何种意义呢？请先查直行的 3，再查横列的 0，两条线交插刚好是 0（数字 0），这表示我们收到的是一个数字。如下表所示。许多可与电脑连线的仪器设备由于只传递数据，所以送回的值都介于 30H 到 39H 间，查 ASCII 表的结果，刚好就是数字 0~9。另外如果我们送一个 45H 给 PC 时，PC 的屏幕上就会显示出 E 的字样。

step 1

step 2	0	1	2	3	4	5	6	7
0	NUL	DEL	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z

附录 B　8051 指令集总整理

参见光盘文件附录 B.doc。

附录 C　8051 指令整理(依功能区分)

参见光盘文件附录 C.doc。

附录 D　8051 指令整理(按十六进制排列)

十六进制代码	字　节　数	助　记　符	操　作　数
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@ R0
07	1	INC	@ R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr，code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@ R0
17	1	DEC	@ R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7

续表

20	3	JB	bit addr，code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A，#data
25	2	ADD	A，data addr
26	1	ADD	A，@ R0
27	1	ADD	A，@ R1
28	1	ADD	A，R0
29	1	ADD	A，R1
2A	1	ADD	A，R2
2B	1	ADD	A，R3
2C	1	ADD	A，R4
2D	1	ADD	A，R5
2E	1	ADD	A，R6
2F	1	ADD	A，R7
30	3	JNB	bit addr，code addr
31	2	ACALL	code addr
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A，#data
35	2	ADDC	A，data addr
36	1	ADDC	A，@ R0
37	1	ADDC	A，@ R1
38	1	ADDC	A，R0
39	1	ADDC	A，R1
3A	1	ADDC	A，R2
3B	1	ADDC	A，R3
3C	1	ADDC	A，R4
3D	1	ADDC	A，R5
3E	1	ADDC	A，R6
3F	1	ADDC	A，R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr，A
43	3	ORL	data addr，#data

续表

44	2	ORL	A，#data
45	2	ORL	A，data addr
46	1	ORL	A，@ R0
47	1	ORL	A，@ R1
48	1	ORL	A，R0
49	1	ORL	A，R1
4A	1	ORL	A，R2
4B	1	ORL	A，R3
4C	1	ORL	A，R4
4D	1	ORL	A，R5
4E	1	ORL	A，R6
4F	1	ORL	A，R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr，A
53	3	ANL	data addr，#data
54	2	ANL	A，#data
55	2	ANL	A，data addr
56	1	ANL	A，@ R0
57	1	ANL	A，@ R1
58	1	ANL	A，R0
59	1	ANL	A，R1
5A	1	ANL	A，R2
5B	1	ANL	A，R3
5C	1	ANL	A，R4
5D	1	ANL	A，R5
5E	1	ANL	A，R6
5F	1	ANL	A，R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	date addr，A
63	3	XRL	date addr，#data
64	2	XRL	A，#data
65	2	XRL	A，data addr
66	1	XRL	A，@ R0
67	1	XRL	A，@ R1

续表

68	1	XRL	A，R0
69	1	XRL	A，R1
6A	1	XRL	A，R2
6B	1	XRL	A，R3
6C	1	XRL	A，R4
6D	1	XRL	A，R5
6E	1	XRL	A，R6
6F	1	XRL	A，R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C，bit addr
73	1	JMP	@ A+DPTR
74	2	MOV	A，#data
75	3	MOV	data addr，#data
76	2	MOV	@ R0，#data
77	2	MOV	@ R1，#data
78	2	MOV	R0，#data
79	2	MOV	R1，#data
7A	2	MOV	R2，#data
7B	2	MOV	R3，#data
7C	2	MOV	R4，#data
7D	2	MOV	R5，#data
7E	2	MOV	R6，#data
7F	2	MOV	R7，#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C，bit addr
83	1	MOVC	A，@ A+PC
84	1	DIV	AB
85	3	MOV	data addr，data addr
86	2	MOV	data addr，@ R0
87	2	MOV	data addr，@ R1
88	2	MOV	data addr，R0
89	2	MOV	data addr，R1
8A	2	MOV	data addr，R2

续表

8B	2	MOV	data addr，R3
8C	2	MOV	data addr，R4
8D	2	MOV	data addr，R5
8E	2	MOV	data addr，R6
8F	2	MOV	data addr，R7
90	3	MOV	DPTR，#data
91	2	ACALL	code addr
92	2	MOV	bit addr，C
93	1	MOVC	A，@ A+DPTR
94	2	SUBB	A，#data
95	2	SUBB	A，data addr
96	1	SUBB	A，@ R0
97	1	SUBB	A，@ R1
98	1	SUBB	A，R0
99	1	SUBB	A，R1
9A	1	SUBB	A，R2
9B	1	SUBB	A，R3
9C	1	SUBB	A，R4
9D	1	SUBB	A，R5
9E	1	SUBB	A，R6
9F	1	SUBB	A，R7
A0	2	ORL	C，/bit addr
A1	2	AJMP	code addr
A2	2	MOV	C，bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		rese rved	
A6	2	MOV	@ R0，data addr
A7	2	MOV	@ R1，data addr
A8	2	MOV	R0，data addr
A9	2	MOV	R1，data addr
AA	2	MOV	R2，data addr
AB	2	MOV	R3，data addr
AC	2	MOV	R4，data addr
AD	2	MOV	R5，data addr
AE	2	MOV	R6，data addr

续表

AF	2	MOV	R7，data addr
B0	3	ANL	C，/bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A，#data，code addr
B5	3	CJNE	A，data addr，code addr
B6	3	CJNE	@ R0，#data，code addr
B7	3	CJNE	@ R1，#data，code addr
B8	3	CJNE	R0，#data，code addr
B9	3	CJNE	R1，#data，code addr
BA	3	CJNE	R2，#data，code addr
BB	3	CJNE	R3，#data，code addr
BC	3	CJNE	R4，#data，code addr
BD	3	CJNE	R5，#data，code addr
BE	3	CJNE	R6，#data，code addr
BF	3	CJNE	R7，#data，code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A，data addr
C6	1	XCH	A，@ R0
C7	1	XCH	A，@ R1
C8	1	XCH	A，R0
C9	1	XCH	A，R1
1CA	1	XCH	A，R2
CB	1	XCH	A，R3
CC	1	XCH	A，R4
CD	1	XCH	A，R5
CE	1	XCH	A，R6
CF	1	XCH	A，R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr

续表

D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	A，data addr，code addr
D6	1	XCHD	A，@ R0
D7	1	XCHD	A，@ R1
D8	2	DJNZ	R0，code addr
D9	2	DJNZ	R1，code addr
DA	2	DJNZ	R2，code addr
DB	2	DJNZ	R3，code addr
DC	2	DJNZ	R4，code addr
DD	2	DJNZ	R5，code addr
DE	2	DJNZ	R6，code addr
DF	2	DJNZ	R7，code addr
E0	1	MOVX	A，@ DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A，@ R0
E3	1	MOVX	A，@ R1
E4	1	CLR	A
E5	2	MOV	A，data addr
E6	1	MOV	A，@ R0
E7	1	MOV	A，@ R1
E8	1	MOV	A，R0
E9	1	MOV	A，R1
EA	1	MOV	A，R2
EB	1	MOV	A，R3
EC	1	MOV	A，R4
ED	1	MOV	A，R5
EE	1	MOV	A，R6
EF	1	MOV	A，R7
F0	1	MOVX	@ DPTR，A
F1	2	ACALL	code addr
F2	1	MOVX	@ R0，A
F3	1	MOVX	@ R1，A
F4	1	CPL	A
F5	2	MOV	data addr A
F6	1	MOV	@ R0，A
F7	1	MOV	@ R1，A

续表

F8	1	MOV	R0，A
F9	1	MOV	R1，A
FA	1	MOV	R2，A
FB	1	MOV	R3，A
FC	1	MOV	R4，A
FD	1	MOV	R5，A
FE	1	MOV	R6，A
FF	1	MOV	R7，A

附录 E　8051 SFR 表与 RESET 后的初始值

F8H									FFH
F0H	B 00000000								F7H
E8H									EFH
E0H	ACC 00000000								E7H
D8H									DFH
D0H	PSW 00000000								D7H
C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			CFH
C0H									C7H
B8H	IP XX000000								BFH
B0H	P3 11111111								B7H
A8H	IE 0X000000								AFH
A0H	P2 11111111								A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111								97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXXS0000	87H

附录 F　SFR 特殊功能缓存器整理表

缓存器名称	地　　址	可位寻址	8051	8052
ACC 累积器	E0H	*	*	*
B 通用缓存器	F0H	*	*	*
PSW 程序状态字符	D0H	*	*	*
SP 堆栈指向器	81H		*	*
DPH 数据指向器(高位)	83H		*	*
DPL 数据指向器(低位)	82H		*	*
P0 端口 0	80H	*	*	*
P1 端口 1	90H	*	*	*
P2 端口 2	A0H	*	*	*
P3 端口 3	B0H	*	*	*
IP 中断优先控制器	B8H	*	*	*
IE 中断致能控制	A8H	*	*	*
TMOD 计时/计数模式控制	89H		*	*
TCON 计时/计数控制	88H	*	*	*
T2CON 第 2 计时/计数控制	C8H	*		*
TH0 TIMER0 高位设置	8CH		*	*
TL0 TIMER0 低位设置	8AH		*	*
TH1 TIMER1 高位设置	8DH		*	*
TL1 TIMER1 低位设置	8BH		*	*
TH2 TIMER2 高位设置	CDH			*
TL2 TIMER2 低位设置	CCH			*
RCAP2H 撷取缓存器(高位)	CBH			*
RCAP2L 撷取缓存器(低位)	CAH			*
SCON 串行通信控制器	98H	*	*	*
SBUF 串行数据缓冲器	99H		*	*
PCON 电源控制缓存器	87H		*	*

PSW 各位定义

CY	AC	F0	RS1	RS0	OV	-	P
D7	D6	D5	D4	D3	D2	D1	D0

TCON 控制缓存器解析

TF1	TR1	TF0	TR0	ID1	IT1	IE0	IT0
D7	D6	D5	D4	D3	D2	D1	D0

TCON2 各位定义

TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
D7	D6	D5	D4	D3	D2	D1	D0

SCON 串行控制端口位说明

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
D7	D6	D5	D4	D3	D2	D1	D0

PCON 电源控制缓存器各位说明

SMOD	----	----	----	GF1	GF0	PD	IDL
D7	D6	D5	D4	D3	D2	D1	D0

IP 中断优先权缓存器各位定义

---	----	PT2	PS	PT1	PX1	PT0	PX0
D7	D6	D5	D4	D3	D2	D1	D0

IE 中断致能缓存器各位定义

EA	---	ET2	ES	ET1	EX1	ET0	EX0
D7	D6	D5	D4	D3	D2	D1	D0

附录 G　DIS51 的进阶使用

我们在数个月以前曾经让读者来函索取 8051 的反汇编程序（DIS51.EXE），用这个程序来反汇编我们的二进制文件范例程序 READTEMP，相信许多用功分析程序的读者一定从中获得宝贵的知识。这个程序总共有三百多个读者索取。不过，有一天清晨两点计算机要收工关机前，我照惯例会再进到 Internet 查询是否有 E-mail 邮件进来，结果在清晨一点半的时候，有一位正在加班的工程师传来一封信，信中的大意是：用 DIS51.EXE 来看一个 4KB 的 8051 程序，会看得很辛苦。我想这个程序是别人写的吧！程序中用了可位寻址的标志，所有的流程及动作都是以这些标志为依据。当然如果不知道这些标志的用意时，会使得程序可读性变成相当低，所以程序看了好几天还是不得其门而入。看来今天又要无法入眠了，这种限期完

工的压力是相当大的。

在我们开发 8051 程序的经验中，一定会用或多或少个可位寻址的标志当成流程控制的依据。如果你不是原始程序的开发者时，就很难在短时间理解程序了，所以一定要费更多的时间去推敲。当 DIS51 反汇编完程序后，除了 DIS 反汇编文件之外，还会产生一个 BAS 文件，上面会记载每个可位寻址（Bit Addressable Segment）的调用次数及调用的地址，我们建议你把整个程序打印出来，报表纸通常会有数百张之多，你可以通过 BAS 文件找出使用次数最多的位地址，然后查询它在那一部分使用了 JB/JNB 做转移的依据，接着找出该位在何处被设置及重设置，从这三方面去推断程序的用意。不过可以理解的是：这类程序不是一两天能搞得清楚的，有些程序要反复多次推敲才能弄懂的。看别人的 8051 程序从头到尾看了二三十次是很正常的。

如果分析的 8051 控制器还有输出入电路的关系时，看起来又更复杂了。首先，我们会把整个电路画成线路图，哪些点对应到输入，哪些点对应到输出，这些全要弄清楚才可以开始分析程序。不过有些程序为了阻止其它人的读取，会加入软硬件的保护，这么一来，若不先解开这些障碍，就无法顺利读取程序了。

假使待分析的控制板上有一个空余的独立 I/O 点时，我们可以拿来做程序的除错或修改开关，并且适时地把状态值送出来，如果你可以做到这个地步时，就可以重新改写程序了。看别人的程序太累了！的确是这样的，但是有些时候又非得如此，今年五月的时候我就碰到一个案例：某大学某系的实验器材：多通道（Multichannel）高速 AD 转换仪器故障，该设备已用超过五年的时间，早已过保修期，原厂也不愿意提供维修的服务，只好求助于旗威科技来处理，因为有好几个研究生要靠这台设备去做论文，我们只好“死马当活马医”了。首先我们仔细地研究了一下机器，发现应该只有其中一组 AD 转换电路损坏，开机后就是因为该 CH 没有响应，而使得机器一直停在等待状态响应上。由于没有线路图可供维修，而且没有备用零件可以更换，所以我们只好走另一条路试着修改内部固件（fireware）程序了，那就可以让机器跳过故障点继续动作了。所以朝着这个方向去反汇编程序并且分析程序，然后重新刻录一个 EPROM，两三天内就把这台机器修好了。可是，如果不这样做时，原单位可能要多花预算去购买新的设备了。

这一章我们尝试用另一种方式来回答读者所遇到的问题，希望对您有所助益，如果您有类似的问题时，也可以通过 E-mail 传给“旗威科技”公司。不过要向读者致歉的是：您先前寄来的若是属于 MIME 格式的信函，在我们这端看来都是乱码，所以无法回复您的问题了。所幸这些信函格式的问题已经解决了，请尽量以 E-mail（chipware@seed.net.tw）来提出问题，以便我们有足够的时间来思考与探讨。

反汇编程序（DIS51.EXE）的操作与使用

我们必须承认学习的第一步就是模仿，学习单片机 8051 也是如此，市面上有许多 8051 的相关产品，若你想对这些东西做更进一步的研究时，该怎么办呢？首先，我们来看看其中的困难点，这类的产品本身都有某些程度的保护，纵使这些保护线路可顺利被解开，若想进一步研究或修改程序也是困难重重，因为程序经过编译、连结后已变成机器码（Machine Code），我们很难由这些机器码中倒推原来的程序写法和动向，而且这些程序机器码中有指令也有数据，必须花相当的心血才能顺利地把指令和数据分开。

所有的 ICE 仿真器都可以将程序机器码反汇编成汇编语言的格式，分别显示在屏幕或打

印在报表上，这种反汇编的方式是逐行翻译的，可以知道程序的写法，但是对程序整体性的分析则略嫌不足，因为我们看程序时，首重各个例程的动作分析，只要知道各个例程的来龙去脉，就能很容易地看出程序的重要关键点。

针对上述的问题，我们特地发展一个 8051 的反汇编程序，只要得到 8051 的二进制文件数据（必须是正确的），并存入软盘或硬盘中，就可以通过这个反汇编程序，DIS51.EXE 把机器码转换成汇编语言的格式，除此之外，它另外产生三个参考数据文件：

（1）翻译后的汇编程序文件其扩展名为 DIS。

（2）DPTR 参考数据文件其扩展名为 DTR。

（3）CALL 参考调用文件其扩展名为 CAL。

（4）JUMP 参考转移文件其扩展名为 JMP。

DIS51.EXE 额外产生的文件把程序中所有关于 CALL、DPTR 和 JUMP 的位置和次数整理出来，并分别存到三个文件中，我们可以通过这些信息，很快地分析出那些例程被调用的次数最多，DPTR 后若接着是 MOVC 指令，则 DPTR 所指的位置一定是数据而非程序，而 DPTR 出现后若接着是 MOVX 指令时，该地址一定是 RAM 或 I/O 地址，我们另外还可轻松地算出有多少个 I/O 端口接在这个系统上，这些信息确实是 ICE 仿真器所无法提供的。

DIS51.EXE 最多可翻译 32KB（32768）的程序数据，其操作步骤如下：

◆ 步骤 1：取得正确的 ROM 程序内存数据，通常由 ICE 仿真器中存入的数据都是正确的，或是由 EPROM 刻录器将程序 ROM 读入后再以二进制格式存入硬盘或软盘中，存入的文件不必指定扩展文件名，假设我们已得到某个程序 ROM，其存入软盘的文件名称为 A:TEST。

◆ 步骤 2：由本书所附的光盘中加载 DIS51.EXE，若该文件已存入 C 硬盘时，我们只要在 C>下按入 DIS51 即开始进入反汇编程序。

◆ 步骤 3：DIS51 加载后，会在屏幕上要求输入待反汇编文件的文件名称，这时我们可以输入先前已存入的文件名称 A:TEST。

步骤 4：DIS51 反汇编程序首先检查程序的长度，并将结果以 16 进位的方式显示该长度。例如：

长度是 2000H 时，代表该程序有 8KB 长。

长度是 4000H 时，代表该程序有 16KB 长。

长度是 8000H 时，代表该程序有 32KB 长。

◆ 步骤 5：反汇编程序接着会显示它将产生四个标准文字文件

（1）反汇编程序文件 A:TEST.DIS。

（2）该程序所使用的 DPTR 文件 A:TEST.DTR。

（3）该程序所使用的 CALL 文件 A:TEST.CAL。

（4）该程序所使用的 JUMP 文件 A:TEST.JMP 。

◆ 步骤 6：程序接着要求输入停止反汇编的地址值，这个值应该小于等于该文件的长度值。

◆ 步骤 7：反汇编程序开始动作，并产生对应的 4 个文字文件来，这些文字文件可用一般的编辑程序（PE2 或 KS2）来查看其中的内容。

◆ 步骤 8：上述的 4 个文字文件中，以 TEST.DIS 反汇编后的程序文件占的空间最大，

通常会超过 100KB 以上，这时可再用另一个执行程序（TEST_DVD.EXE），将 TEST.DAT 切割成数小部分，以免空间过大而造成编辑程序无法处理。

◆ 步骤 9：TEXT_DVD.EXE 执行时，只需分别输入原文字文件的名称（不加扩展名）及欲输出的文件名称（也不必加扩展名）即可。

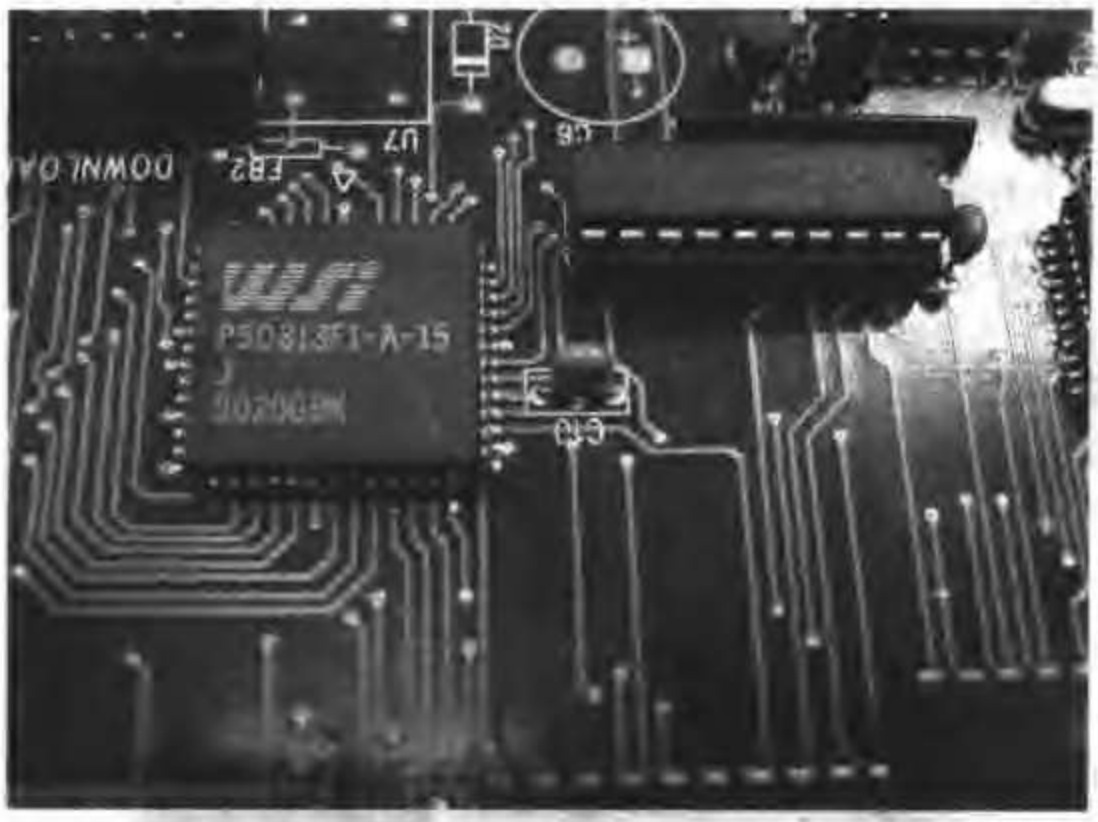
◆ 步骤 10：TEXT_DVD.EXE 将 TEST.DIS 文字文件每隔 4000 行分割成一个另一个文字文件，若 TEST.DIS 有 10000 行，且输出的文件指定为 TESTX 时，则共产生三个文件：TESTX.DS1 占 40000 行、TESTX.DS2 也占 4000 行，TESTX.DS3 则只占 2000 行。

我们也用另一个汇编程序分别对照原程序和反汇编后的程序间有何不同点。对此程序写法有兴趣的读者可直接与作者联络，以便索取该原始程序。

附录 H　一张照片一个故事

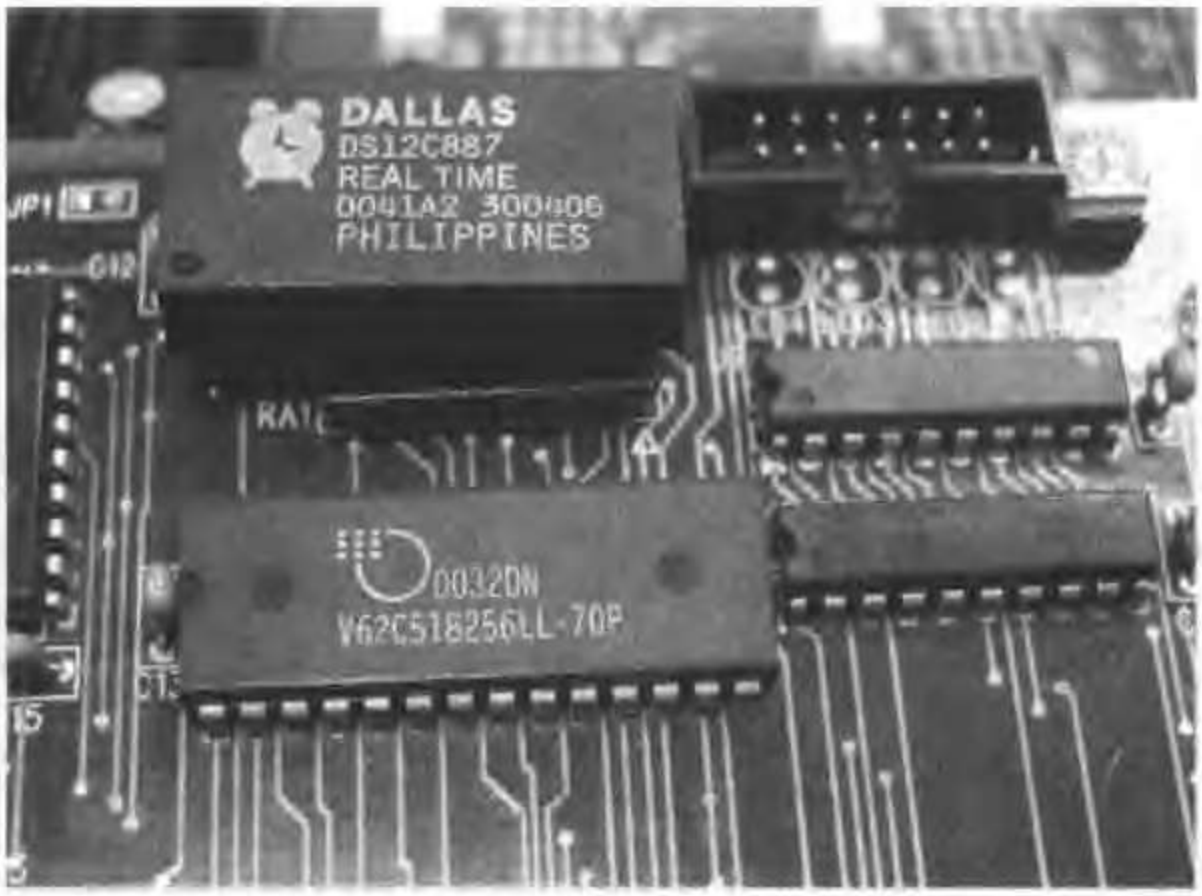
在 8051 的硬件扩充上，我们习惯用 8255 来做缓冲 IC，最多可达 24 点之多。当点数到达数百点甚至千点时，8255 的接法就有点行不通了，可能要改用传统的 TTL IC 来做，硬件成本才不会过高。这是一片 256 点的输入板，我们用 4 片输入板来测试 BGA 基材的导通特性。数字的输入信号经过缓冲与译码后才传回 FLAG51 控制板上。产品用在工业用途上首重的是稳定不故障，要达到这个目的当然是要把硬件保护线路做好，让系统在正常操作的情形下根本没有出错的可能。

一张照片一个故事



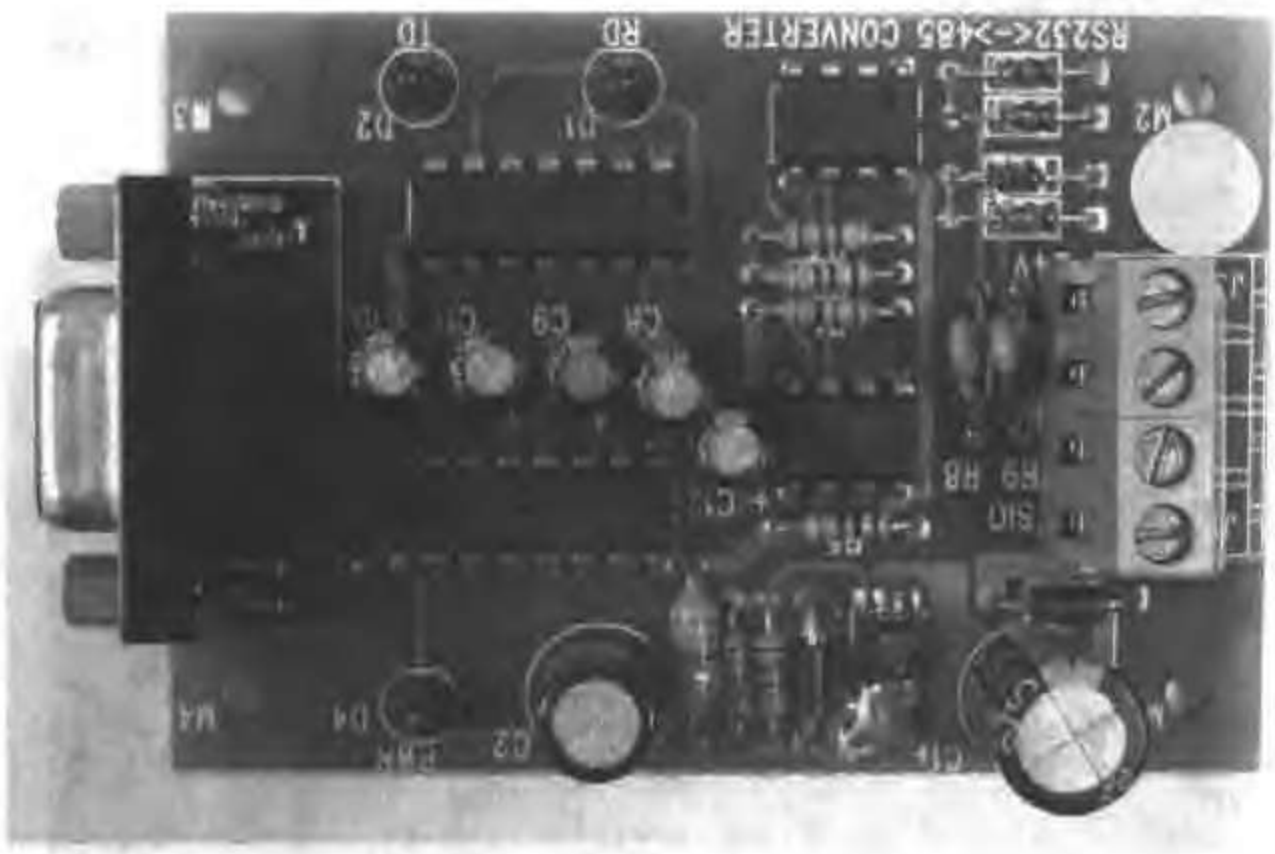
大部分的学习者不是通过购买 8051 的 ICE 来认识该 CPU，就是用 8051 通过 RS232 端口下载程序来学习指令。有没有更便利的方法呢？答案当然是有的，有一些特殊的内存 IC，内部有 Flash 与 EEPROM 两个方块，我们可以在 EEPROM 段放置系统程序的启动程序（BOOT），另外的 Flash 内则是我们自己的应用程序，可以随时下载更新。这两个部分是可以用特定程序来切换的，开机时先从 EEPROM BOOT 区启动，一切没问题时切换到 Flash 程序，系统自动重启直接改由 Flash 重新启动。

一张照片一个故事



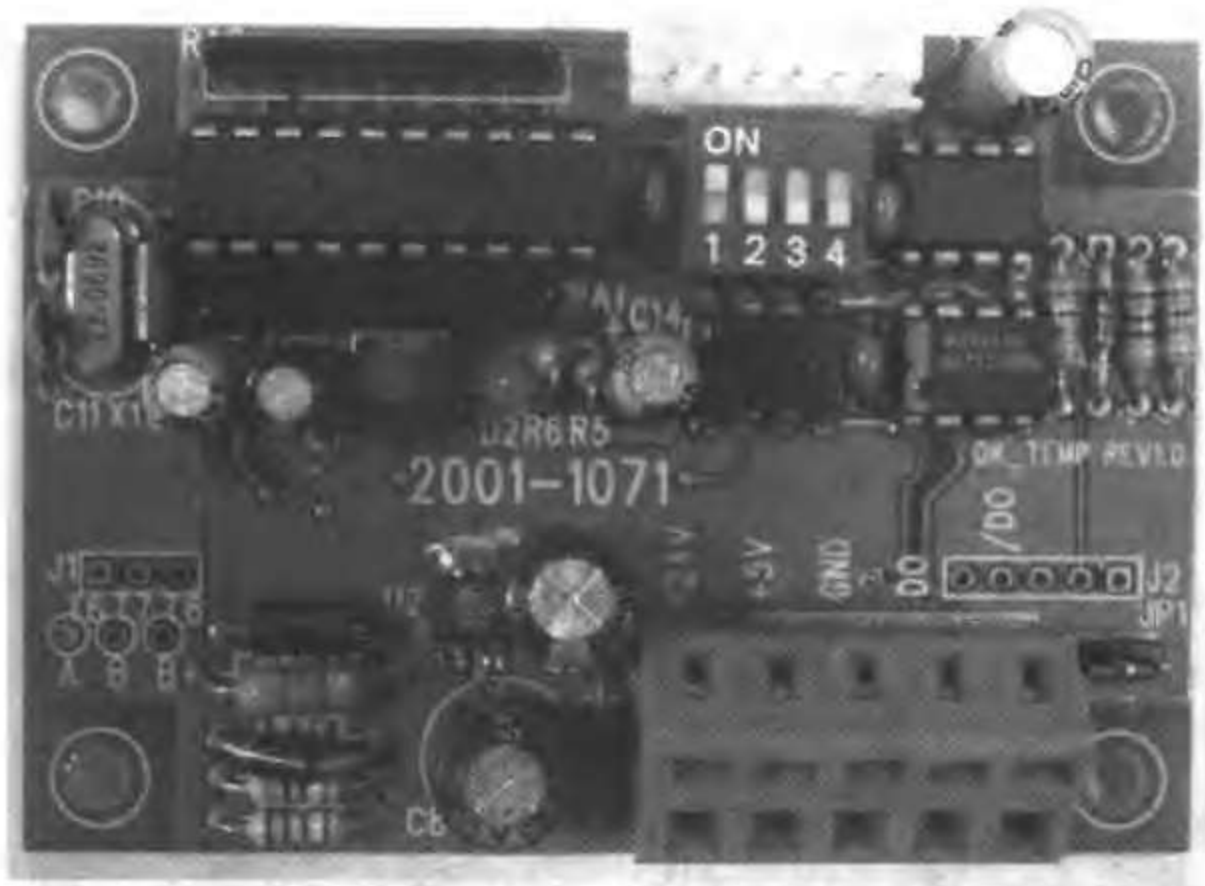
如果你的控制板要记录日期或时期，硬件上一定要有实时时钟的 IC 在上面，这类的 IC 内部会有锂电池包含在内，所以 IC 的厚度会很明显地增加不少。使用上只要只要插上该枚 IC 即可。但是在我们的应用程序上，要先对该 IC 执行启动的命令，方能进行实时时钟的日期与时间更新，在此同时就会开始耗掉锂电池的电力。PC 板长时间不动作时，也可以下达类似的命令给 RTC，再度关闭锂电池的电源，以减少电力损耗。

一张照片一个故事



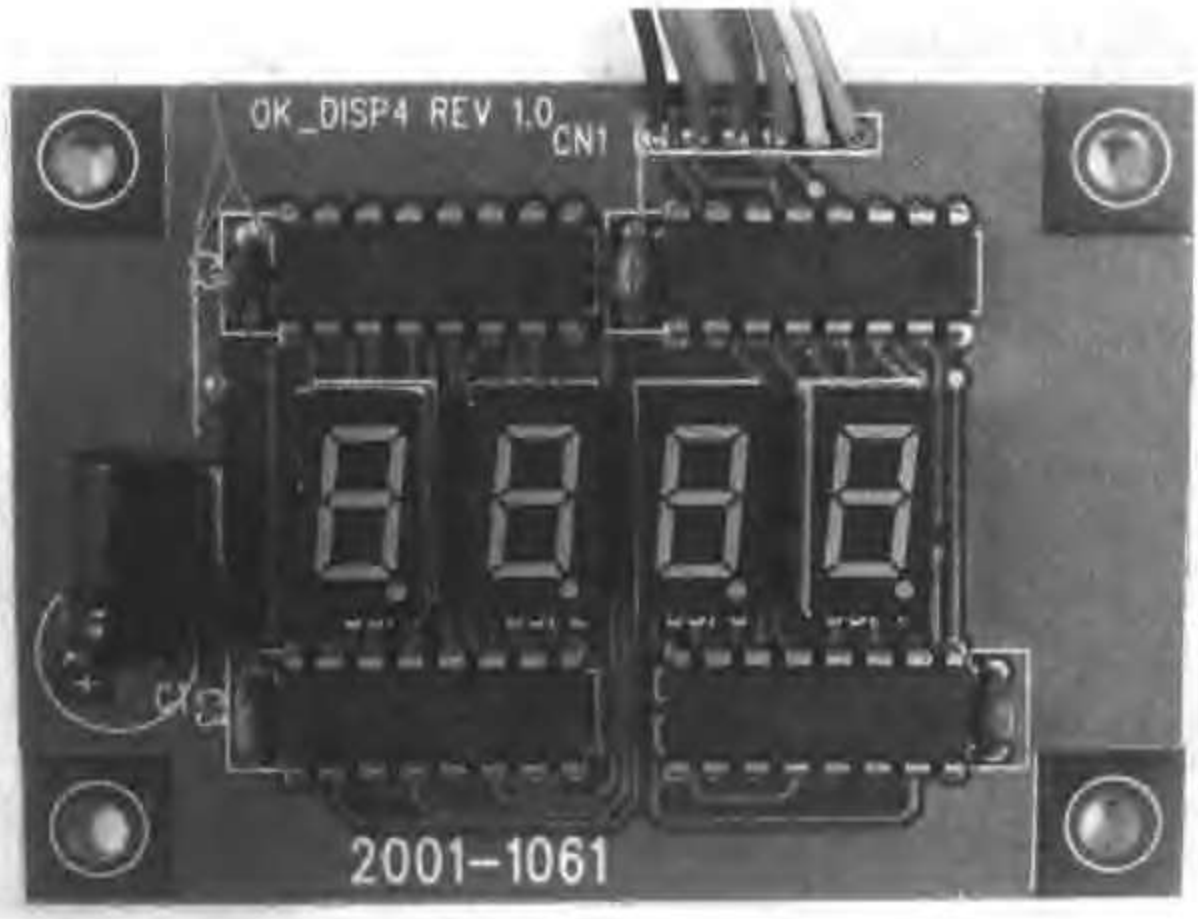
这是一块 RS232 转 RS485 的转换接口，传统的 PC 通过此转换板就可以跟 31 台具备 RS485 的外围通信连接。在化工厂的计算机控制室里，我们用这块接口去监视数百个温度点，并且将收到的数据做成文件供以后分析用。RS485 通信虽然传输距离可以长达数百米，但是信号处理不妥时，可能抓到的数据都是一连串乱码。所以我们通常会制作一个 RS485 的数据串监视器（monitor），来察看传输在线的信号是否有所差错。

一张照片一个故事



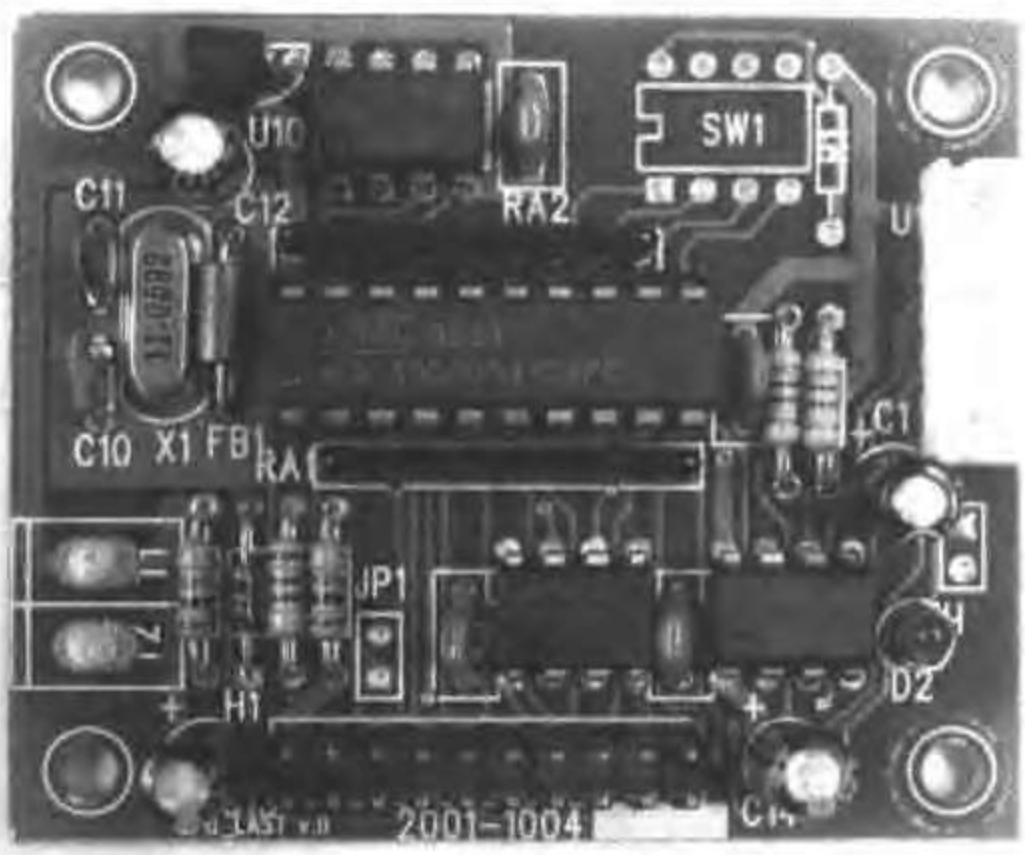
这是“旗威科技”所设计的高精度温度控制板的照片，硬件规格上也有 RS485 的接口，我们用此设备来监视商场的许多温度数据，例如入口的温度、室外的温度、冷却水塔的温度以及空调设备的温度等等。这些数据经过 PC 记录及分析后可以用遥控的方式将重要耗电的设备解连或卸载，以便节省更多的电力。

一张照片一个故事



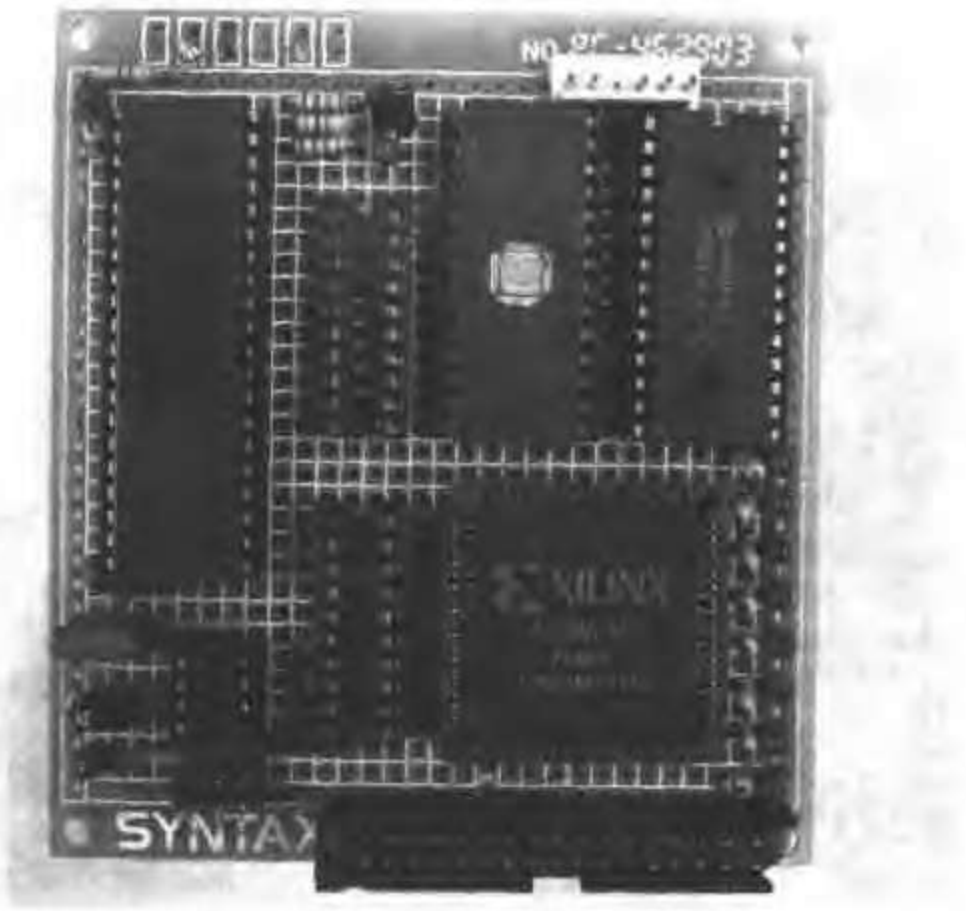
显示 4 个数字需要多少个 IC 来推动？有的人会用软件扫描的方式来做，只要两个 IC 就解决了。在这里我们却用了四个相同编号的 IC 来做，所有的控制都是用 Latch 门锁电路来做，硬件费用可能高一点。第一个做法需要定时中断，随时对显示屏做数据的更新。第二个做法只要更新数据一次即可，完全不会降低系统的效能，而且程序写法也很简单。微电脑设计工程师永远会在程序写法、硬件成本与开发时程上找到一个平衡点。

一张照片一个故事



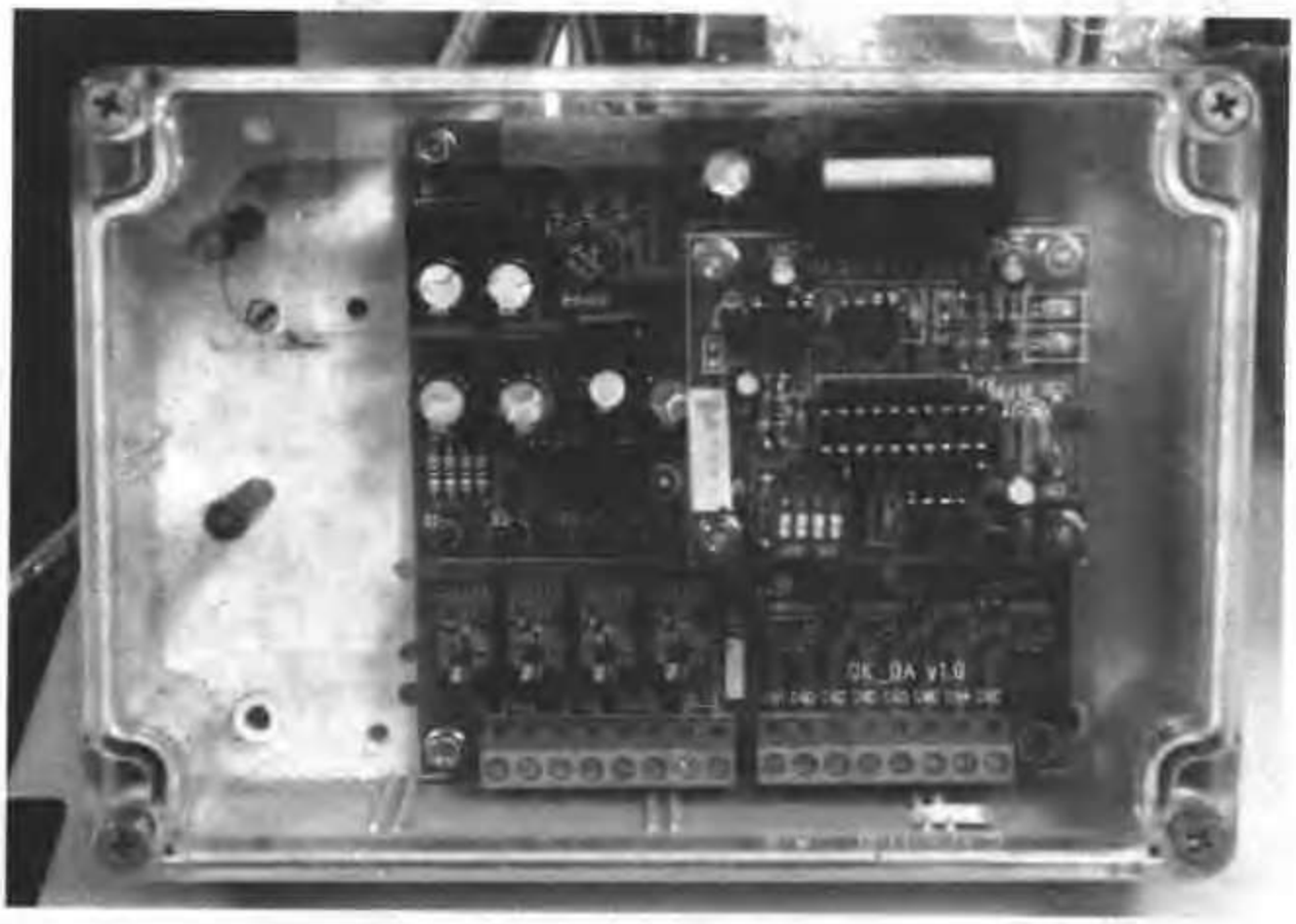
这是 AT89C2051 的硬件设计范例。小小的控制板上有 4P 的 DIP SW 或串行 EEPROM 空间，照片上两者都未焊上。也有一个标准的 RS485 接口，除此之外 AT89C2051 的所有接脚都被拉到 20P 的接头上，用一条 20P 的数据线就可以做外部的控制应用。当该 RS485 正在做传出（Transmit）动作时，板上的 LED 会跟随亮起，我们可借用此灯号来判断是否系统要求数据。

一张照片一个故事



这是 FLAG51 设计时的内幕消息之一，当初（10 年前）我们在规划 FLAG51 时，就曾经考虑到用 Xilinx 的 FPGA 做外部设备。有图为证!这张照片就是当初我们手工焊接的雏型板，那时 FPGA 的价格还是很高，所以，到最后我们还是乖乖地回头采用 8255 做可程序化的外部 I/O，可是 10 年后情况变了，FPGA 随处可见价格漂亮，反到是 8255 不好买而且价格又贵。

一张照片一个故事



具备 RS485 接口的继电器与 DA 数字转模拟输出板，在工业控制的领域上有某些应用不只是 ON/OFF 控制而已。以中央空调用的冷却水塔为例，其内部的排热风扇应该是无段调整的，当热量过多时，风扇应该转得更快，以方便将更多的热气排除。反之，在寒冷天时冷却水塔上的风扇就可以转慢一点。这种应用场合就可以把 DA 转换板派上用场。

一张照片一个故事

这是一台恒温的水槽，内部包括让温度下降的冷冻压缩机及让温度上升的电热线（Heater），温度调整范围可以由 0℃到 100℃，只要设定好温度值，几十分钟后就可以在水槽内得到一个相当稳定的温度参考值。比室温高的温度值比较快达到，低温就比较难，这是因为压缩机使水降温的速度本来就比较慢。当我们设计温度方面的外部设备时，也是用这台恒温水槽来做温度值的比对。