内 容 简 介

本书与《MCS－51 单片机原理与应用》教材配套使用。全书共分 6 章，第 1 章为单片机实验实训概述，第 2 章为单片机基本应用编程和实验，第 3 章为 MCS－51 单片机功能系统实验，第 4 章为 MCS－51 单片机接口扩展实用实验，第 5 章为单片机应用系统设计与实习实训，第 6 章为 Keil 集成开发软件平台介绍。本书以介绍 MCS－51 单片机原理与应用的实践训练为主线，内容丰富，特色鲜明，不仅介绍了 MCS－51 单片机开发方面的知识，而且还编写了大量的课程实验和综合实训；不仅对培养学生提高单片机的工程实践能力有重要的指导作用，而且对该课程的教学方法改革和建设也有重要的指导意义。

本书适合作高职、高专电类专业教材，也可作机电、仪表等专业的教学用书。

图书在版编目(CIP)数据

MCS－51 单片机原理与应用实验实训教程/石从刚主编.
北京：北京航空航天大学出版社，2007.2
ISBN 978－7－81077－624－0

Ⅰ.M… Ⅱ.石… Ⅲ.单片微型计算机—高等学校：
技术学校—教学参考资料 Ⅳ.TP368.1－44

中国版本图书馆 CIP 数据核字(2006)第 157140 号

MCS－51 单片机原理与应用实验实训教程

主　编　石从刚

副主编　宋剑英　胡希勇

责任编辑　韩文礼

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(100083)　发行部电话：010－82317024　传真：010－82328026

http://www.buaapress.com.cn　E-mail:bhpress@263.net

北京市松源印刷有限公司印装　各地书店经销

*

开本：787×960　1/16　印张：10.5　字数：235 千字

2007 年 2 月第 1 版　2007 年 2 月第 1 次印刷　印数：4 000 册

ISBN 978－7－81077－624－0　定价：16.00 元

目　　录

第 1 章　MCS－51 单片机实验实训概述

1.1　MCS－51 单片机实验实训的内容与要求

MCS－51 单片机应用技术是一门实践性很强的课程，具有极其广泛的工程应用价值。工程技术人员若要较深入地掌握该门技术，就必须在加强理论知识学习的同时，还要注重加强对该技术实际操作技能的系统训练学习。不仅要通过传统的实验实训掌握该技术的基本实际操作技能，而且还要通过该课程专门的设计与实习实训，掌握该技术在实际工程应用系统中的设计与开发方面的综合实践技能。最后结合专业知识，在毕业设计时进一步加强单片机在工程控制应用综合设计方面的实践训练，这样才能真正地达到对该技术的教学目的。

1. MCS－51 单片机实验实训的教学安排

MCS－51 单片机应用技术的实践技能应主要体现在对实际工程应用系统的设计与开发方面的综合实践能力的培养上。与一般微型计算机（微机）的工程应用一样，对于一个 MCS－51 单片机应用系统或产品的研制来说，从工程任务的提出到定型生产或投入使用，都要经过方案的总体论证、系统设计、软件及硬件的开发、联机调试、系统装配和产品定型等若干步骤。因此，MCS－51 单片机应用技术的实际技能应包括利用单片机开发实验系统进行基本的实验仿真技能、应用系统综合设计技能和应用系统综合开发技能三大部分。这三大部分的技能都可通过采取“开放式”的实践教学方法加以针对性的系统训练。“开放式”的实践教学方法包括“验证式”、“模仿式”、“探索式”和“开发式”四种基本方式。其中，前三种方式主要用于配合该门课程理论教学中的实验教学，属于进行基本的实验仿真技能范畴的实践训练，其目的是使学生掌握所必需的实验开发系统的使用、基本应用程序、硬件系统扩展的设计与调试等方面的实践操作技能，同时使学生巩固和深化对 MCS－51 单片机应用技术理论知识的掌握和理解，为后续的课程设计与课程实习的综合实践训练打下扎实的理论与实践基础。第四种“开发式”实践教学方法主要用于在本课程结束后，另外专门设置的课程设计与课程实习中进行综合性实践技能训练。

2. 实验实训的教学要求

实验实训的教学安排应贯穿于理论教学的始终，要合理地安排在各理论知识点的教学后及时进行。其教学时数宜占理论教学时数的 25%，应采用“验证式”、“模仿式”和“探索式”三

种开放式的实践教学方法加以循序渐进式的提高训练。

①“验证式”实践教学方法主要用于理论教学的初始阶段。学生应根据教师给定的实验内容和操作步骤进行实验，以便尽快对所需的实验设备和本课程的实验特点有充分的感性认识和实践体验。

②“模仿式”实践教学方法主要用于理论教学的中期阶段。学生应对给定的参考性应用程序和实验电路作适当地改造或完善后再进行实验，并通过观察、分析和研究实验过程和结果，以提高分析问题和解决问题的实践操作能力。

③“探索式”实践教学方法主要用于理论教学的后期阶段。学生应根据教师给定的实验项目要求，自行设计和搭试实验电路和编写程序进行实验，以充分发挥学生的主观创造性并进一步提高分析问题和解决问题的综合性基本实践操作能力。

本课程的实验项目分为三大类：一是 MCS－51 单片机基本应用程序设计与调试实验，二是 MCS－51 单片机功能系统应用编程与调试实验，三是 MCS－51 单片机应用系统扩展编程与调试实验。对上述每一大类实验项目都安排有若干个具体的实验项目，并且按“验证式”、“模仿式”和“探索式”的实践教学要求又可进一步地分为三小类。

注意：在实验实训的教学安排方面，还要安排一定的实验室开放时间，以便使学有余力的学生有更多地发挥自己才能的实践训练机会，学得吃力的学生也有充分的再学习与提高的机会。

3. 综合设计与实习实训的教学要求

综合设计与实习实训的教学应安排在本课程教学结束后进行，教学时数宜为两周，其中课程设计与课程实习各一周，两者的教学内容应相辅相成，紧密结合。课程实习的内容就是要在实验室里具体实现课程设计的内容，以便实现灵活地使用 MCS－51 单片机应用技术进行实际应用系统的开发研制方面的实践训练。

综合设计与实习实训的教学应采用“开发式”的实践教学方法，要求学生独立或分小组完成一个中、小型实训课题的方案论证、系统设计、软件和硬件调试以及书写研制报告等任务，以训练学生综合运用已学的各方面知识进行科技开发和创新方面的能力，并使学生初步掌握有关单片机应用系统开发研制方面的过程和特点，为毕业设计和实际工作需要培养更扎实的动手能力，体现“动手能力强”的优良工程素质。

本课程的综合实训项目主要是 MCS－51 单片机小型应用系统的设计与实习课题，主要用于课程设计与实习实训。

课程设计是课程实习的前期阶段，根据课题的具体性能要求，首先进行系统的、正确的总体方案设计。这是成功地进行产品研制的必要基础，是至关重要的工作，其中要用到多种知识，是对知识合理的、优化的组合，是体现设计者科研开发能力的重要标准。至于单片机应用课题总体方案的设计，包括硬件系统设计和软件系统设计两大部分，前者是后者的基础，后者

是前者的补充，两者互为相关，同时要注意到能用软件系统代替硬件系统的设计部分尽量用软件系统来代替，以最大限度地减少硬件系统的成本，提高产品的可靠性。关于课程设计的具体要求如下。

① 根据给定课题的具体要求，仔细分析，提出粗略的软件系统、硬件系统分割的宏观方案。

② 根据已定的宏观设计方案，选择合适的单片机产品（一般均选用 MCS－51 单片机），考虑最节省的、必要的有关扩展，设计出整机硬件系统原理图，以满足用户的使用要求。

③ 根据已设计出的整机硬件系统原理图，选用合适的、经济的有关电子器件及通用的单片机接口芯片设计出具体的硬件系统布置图，最好再在 PC 机上用 Protel 软件设计出包括各类器件尺寸大小的、能直接用来制作产品的印刷布线图。

④ 根据整机硬件系统原理图及具体产品的功能要求，设计出所需的软件系统，画出已设计出的软件系统原理框图，并尽量将功能模块化，以实现模块化程序设计的基本思路。

⑤ 根据已设计出的软件系统框图，用 MCS－51 汇编语言编制出各功能模块的子程序及整机软件系统的主程序。程序设计时，要充分考虑与所设计硬件系统的连接及有关定量的要求。

至此，单片机应用系统的设计开发工作中的有关课题总体方案的设计告一段落。若是产品开发，其第二阶段的主要工作就是利用合适的单片机开发系统对所设计的软件系统和硬件系统进行在线仿真调试，以进一步完善第一阶段的系统设计。第三阶段就是制作硬件系统的印刷线路板，安装硬件系统，固化软件系统，设计并制作整机机箱以及安装整机。最后，对整机进行现场运行试验，直到满足实际应用要求为止。由此可见，系统设计阶段是产品开发中最基础的也是最重要的一步。

课程实习是课程设计的后续阶段，即对有关课题的系统设计方案全部或主要部分在实验室加以具体实现，这是提高学生工程应用素质的又一重要的实践训练。但由于时间关系，进行系统全面的开发研制是不切合实际的，因此，最好将实习内容分解成基本部分和提高部分。基本部分要求必须完成；而对于提高部分，在综合设计与实习的教学结束后，教师还应安排一定的“第二课堂”指导时间，以使学生能利用课外时间进行科技小产品具体研制方面的实践锻炼，可包括这样一些研制工作：软件系统、硬件系统的联机调试；印刷线路板的制作；设计机箱；现场运行调试。

4. MCS－51 单片机实验实训的教学考核

合理的考核办法是保证教学质量的有力措施。考核的内容应包括对实训过程的考核和对实训成果的考核两大部分：前者主要是针对学生的学习态度和独立完成任务的能力等方面加以考核，后者主要是针对学生所完成任务的成果质量（含实训总结报告）等方面加以考核。考核的方式可以由指导教师在指导过程中的考查评价、对最终成果的验收评价和对若干问题的

口试评价三方面的成绩加以综合评定。对 MCS－51 单片机实验实训的考核应分为实验考核、课程设计考核和课程实习考核三方面，对这些考核应相互独立地进行。

⑴ 实训考核的具体措施

对 MCS－51 单片机实训的实验考核、课程设计考核和课程实习考核应制订相应的考核细则。下面提出的有关考核办法供各学校参考。

① 实验考核细则。对实验考核的成绩记录办法一般有两种方式：一是将实验考核成绩作为本课程理论课成绩的一部分，所占比例通常为 20%，而且宜用“百分制”；二是将实验考核成绩单独记录，以突出对实验实践技能的培养，宜用“五分制”。

对于第一种成绩记录方式的实验考核，主要通过指导教师对学生的实验态度（占 20% 为宜）、实验质量（占 40% 为宜）和报告质量（占 40% 为宜）的综合评价来评定成绩，每次实验都要有详细的记录，最后再根据平时每次实验的记录成绩加以总的评定。

对于第二种成绩记录方式的实验考核，最好在课程全部实验结束后，安排一次综合实验考试，以该成绩为主（占 80% 为宜）并综合考虑平时实验的成绩（占 20% 为宜）。

② 课程设计与课程实习考核细则。对课程设计与课程实习考核的成绩记录办法一般有两种方式：一是将课程设计与课程实习分别单独考核并分别记录成绩；二是将课程设计与课程实习分别单独考核后统一记录成绩，各占 50%，而且宜用“五分制”。

对于课程设计的考核，主要以三项指标：设计态度（占 20% 为宜）、设计报告质量（占 40% 为宜）和口试成绩（占 40% 为宜）的综合评价来评定成绩。

对于课程实习的考核，主要以四项指标：实习态度（占 20% 为宜）、实习成果质量（占 40% 为宜）、实习报告质量（占 20% 为宜）和口试成绩（占 20% 为宜）的综合评价来评定成绩。

值得注意的是，考核中必须强调对学生创新能力的评价，凡是创新能力不够的，不可评定为优秀等级。

⑵ 实训总结的内容要求

书写高质量的工作总结也是反映工程实践素质高低的一个重要方面，工程技术人员应能用书面形式系统、完整、清晰地表达自己的研究工作成果，其目的是让人很容易地看懂所研究项目的内容、方案、原理和实现方法等。因此，也必须训练学生书写高质量的工作总结。这里要求的工作总结分为实验报告、课程设计报告和课程实习报告三种，并分别提出了书写的基本规范要求，为学生工作后能书写高质量的产品使用说明书和科研论文打下坚实的基础。

对于实验报告，应包括如下基本内容：

① 实验项目的内容要求。

② 实验的目的意义。

③ 实验的实现方案和需要的实验条件。

④ 实验的操作过程和实验结果。

⑤ 对关键性的实验步骤及实验现象的分析。

⑥ 对实验操作方法与内容作进一步改革与扩充的设想。

对于课程设计报告，应包括如下基本内容：

① 设计论文的摘要，包括中文和英文的，各 100 字左右。

② 设计论文的目录，列出章、节、页码等。

③ 阐明课题的内容要求及性能指标要求。

④ 阐明课题总体方案设计的思路（含软件系统和硬件系统）。

⑤ 详细阐述总体方案中各功能模块设计的工作原理（含软件系统和硬件系统）。

⑥ 用 Protel 软件绘制整机硬件系统的设计原理和印制布线图。

⑦ 绘制按结构化程序设计思想设计的整机软件系统的原理框图。

⑧ 列出经初步调试并打印输出的整机系统工作程序。

⑨ 列出所选用的元器件明细表。

⑩ 对关键性设计部分的补充分析和进一步拓展性设计的设想。

对于课程实习报告，应包括如下基本内容：

① 本次实习的基本内容和要求。

② 经仿真调试后需对前面所设计的内容加以完善部分的详细更正。

③ 对关键性的实验操作及实验现象的分析。

④ 对课题内容全面加以仿真调试的打算。

⑤ 对课题作二期设计开发的设想。

1.2　MCS－51 单片机应用系统开发的一般过程与特点

MCS－51 单片机应用技术是一门实践性很强的应用技术，除了需要学习其必要的理论知识外，还必须掌握其应用开发技术。同一般微机应用系统的开发一样，由于 MCS－51 单片机自身没有开发能力，因此在其应用系统的开发时也必须要借助于相应的开发系统。单片机应用系统的开发是指从提出任务、定型生产到投入使用的整个过程，单片机应用系统开发周期的长短主要取决于硬件、软件和样机调试所花的时间，特别是软件的开发时间。

1．MCS－51 单片机应用系统开发的一般过程

MCS－51 单片机应用系统的开发过程一般包括总体方案论证、系统总体设计、软件、硬件开发、联机调试和产品定型等几个步骤。

⑴ 总体方案论证

在开始对一个 MCS－51 单片机应用系统或产品设计之前，一般都需要对所研制的产品进行总体设计方案的论证，通常要做包括产品的技术指标和系统组成两个方面的论证工作。

① 技术指标根据产品研制的任务，需要在充分调研的基础上，对产品的先进性、可靠性、

可维护性和性能价格比等进行综合考虑，由此再制定出待开发产品的功能、性能要求、工作环境、外形尺寸和质量（旧称重量）等技术指标。

② 系统组成在确定了一个单片机应用系统或产品技术指标的基础上，就要充分考虑系统的组成。首先要考虑待组成系统的单片机机型和元器件的选择。

对于单片机机型的选择，应根据单片机应用系统或产品的技术指标、单片机的性能和价格、市场货源、相应的开发系统、研制周期等因素来选定。一般，在 MCS－51 单片机能满足控制要求的情况下，尽量选用 MCS－51 系列单片机。

对于元器件的选择，主要有传感器、模拟电路、可编程扩展器件、RAM 和 EPROM 等，应根据系统的技术指标要求，对这些元器件加以适当的选择。

⑵ 系统的总体设计

系统的总体设计包括硬件系统和软件系统的综合设计。一个单片机应用系统或产品的硬件和软件之间有着密切的相互制约的关系，有可能会从硬件的角度对软件提出一些特殊的要求；也有可能会从软件的角度对硬件提出一些特殊的要求。在某些情况下，硬件和软件又具有一定的互换性，有些由硬件实现的功能可以由软件来完成；反之亦然。当然，较多的使用硬件，可以提高系统的工作速度，而且可以减少软件的工作量；较多的使用软件，可以降低硬件成本，简化电路，其缺点是要增加软件的工作量。

鉴于上述情况，必须对单片机应用系统或产品的硬件和软件功能进行合理的划分，划分的原则主要是根据单片机应用系统或产品的实时性要求、研制周期和生产批量来考虑。例如，若对系统或产品的实时性要求高，就应该较多地使用硬件来完成一些功能；若系统或产品的批量大，就应该着重考虑降低硬件的成本；若系统或产品的批量小，就应该着眼于减少研制工作量。此外，如果还需要保护系统或产品研制者的合法权利，就得对系统或产品进行加密，加密可以在硬件上进行，也可以在软件上进行，当然也可以将两者结合起来。

⑶ 硬件、软件的开发与联机调试

在系统总体设计工作大体完成以后，硬件和软件任务的划分也已明确，这时就可以在一定程度上独立地进行硬件和软件的研制，在硬件和软件研制分别完成的基础上便可将它们进行联机调试，用仿真开发器进行仿真。

在联机调试中，应对硬件和软件各个部分进行全面调试，仔细检查样机是否实现系统预期功能和达到规定的性能指标，以便在调试阶段就把可能存在的问题和隐患充分地暴露出来，然后，再进行多次调试，直到符合设计要求为止。

⑷ 产品定型

在对样机进行了全面测试，确信没有错误之后，才能进行系统或产品的最后设计，即绘制最后正式的硬件逻辑图及印制板、固化软件、装配系统或产品，经现场运行无误后，才能结束整个开发研制工作，最后写出系统或产品的技术报告等。

2．MCS－51 单片机应用系统硬件电路的设计与调试

在掌握了 MCS－51 单片机原理、基本结构、系统配置和接口技术以后，就可根据 MCS－51 单片机应用系统所需要的功能来设计硬件电路。硬件电路的设计需要掌握一定的设计原则以及必要的调试方法。

⑴ 硬件电路的设计原则

硬件电路设计一般包括两个部分的内容：一是扩展单片机芯片内部单元的功能，如在 ROM、RAM、定时/计数器和 I/O 接口线等的容量不满足应用系统的要求时，就需要设计相应的功能扩展电路；二是根据应用系统的功能要求配置相应的外部设备，如键盘、显示器、打印机、A/D 转换器、D/A 转换器等。

对应用系统进行扩展与配置时，应充分考虑如下的一些设计原则：

① 尽可能选择典型电路，并符合单片机的常规使用要求；

② 扩展与配置的数量应充分满足应用系统功能的要求并留有余地，以便能方便地进行功能扩展和作进一步的开发；

③ 应充分结合软件方案来考虑硬件结构；

④ 整个系统中的相关器件应尽可能做到性能匹配，如选用的晶振频率较高时，就应选择存取速度较高的芯片；

⑤ 应重视整个系统的可靠性及抗干扰设计，如充分筛选芯片和元器件，增加去祸电路，采取隔离和屏蔽等措施。

⑵ 硬件电路的调试

当把单片机应用系统的试验样机装配完毕以后，便可进入硬件调试阶段。硬件调试一般按脱机检查和联机调试两步进行。

脱机检查就是根据硬件逻辑电路图，用万用表等工具检查试验样机连线的正确性与可靠性，其中对电源系统的极性、短路故障等问题要特别注意。要仔细地检查硬件电路的地址总线、数据总线和控制总线是否有短路、开路或错位情况；在电路检查完成以后，可在不插入集成电路芯片的情况下加电检查，确定一些点的电位是否正常；断电后，再把集成电路芯片正确地插入各插座，然后通电，迅速地检查各芯片是否有温升异常，以及单片机和有关接口电路的通电初始化状态是否正确。在上述情况都正常后，便可进入硬件的联机调试。值得注意的是，在通电情况下，不可拔插任何集成电路芯片。

联机调试就是把试验样机上的单片机和 EPROM 拔下，并将单片机开发装置的仿真头插入试验样机上的单片机插座上，这样就将仿真器与单片机应用系统两者连接起来，构成了联机调试状态。联机调试时，首先分别接通开发装置和试验样机的电源，在通电后，若开发装置能正常工作，就说明试验样机的地址总线、数据总线和控制总线无短路故障；否则应再断电检查试验样机的线路，直到排除了故障为止；联机调试状态下，还可使用开发装置对试验样机进行

全面的检查，例如检查读/写结果、观察试验样机 I/O 设备的状态等，利用这些手段还可进一步检查、排除试验样机的硬件故障（包含设计和工艺错误）。在试验样机中，常见的故障有元器件品质低劣，开发装置或试验样机接地不良，电压波动大，单片机负载过重，线路短路或断路以及设计工艺错误。

3. MCS－51 单片机应用系统软件的开发与调试

对 MCS－51 单片机的应用开发，除了必须保证硬件电路的正确连接以外，更重要的工作是进行软件的开发。单片机同其他微机一样，如果没有软件的支持，所设计出的试验样机就没有任何用途。而同一台试验样机之所以也能应用于不同的场合，不仅是因为它所连接的外围设备不同，更重要的是因为支持它工作的软件不同。在开发应用软件时，应掌握一定的程序设计原则和开发方法。

⑴ 软件的设计原则

设计一个好的应用软件，必须充分考虑如下的一些设计原则：

① 软件在结构上应清晰、简洁及流程合理。

② 各功能子程序应实现模块化、子程序化，以便于调试、连接、移植和修改。

③ 程序存储区、数据存储区应合理规划，做到既节约内存容量，又方便操作。

④ 运行状态应实现标志化管理，对各功能程序的运行状态、运行结果以及运行要求都要设置状态标志以便查询。

⑤ 对需要特殊抗干扰的应用系统应采用软件抗干扰措施，以提高系统的可靠性。

⑥ 如有必要可增加加密措施，以保护自身的合法的知识产权权利。

⑵ 应用软件的开发

应用软件开发的最终要求是在试验样机的程序存储区中存入能满足系统功能要求的应用程序机器码，应用软件的开发包括编写应用程序、将应用程序翻译成机器码、对应用程序进行排错调试、用仿真开发器进行仿真、把应用程序机器码固化到程序存储器中等工作。

① 编写应用程序根据开发系统的性能，一般有机器语言、汇编语言和高级语言三种方法。机器语言编程方法，即是直接用十六进制数表示的机器码编程并输入到单片机的存储器中去。该方法既麻烦又容易出错，现已很少有人使用。汇编语言编程方法，即是用符号指令编写出源程序，并通过 PC 上常采用的 Edlin（行编辑）、Wordstar（全屏幕编辑）或 PE（全屏幕编辑）等软件把源程序输入到 PC 上，并在磁盘上产生源文件。该方法目前最为常用。高级语言编程方法，即在 PC 上配备 PL/M－51 等交叉编译程序，用 PL/M 高级语言来对 MCS－51 单片机进行编程。该方法目前正在推广之中。

② 将应用程序翻译成机器码就是将汇编语言编写的应用程序（源程序）翻译成单片机能直接运行的机器码（目标程序），这个过程称为汇编。汇编一般有手工汇编和汇编程序汇编两种方法。手工汇编是根据一种 CPU 指令与机器码有着一一对应的关系进行手工查表对照翻

译的编译方法 汇编程序汇编通常是用一种称为汇编/反汇编的程序由 PC 自动执行翻译对照工作。汇编是将源程序翻译成机器码，反汇编是将机器码翻译成源程序。由于汇编/反汇编仅是做查找翻译工作，因此可以不受 CPU 型号的限制。也就是说，可以用 A 型 CPU 的指令系统编制一个汇编/反汇编程序，在 A 型微机上对 B 型 CPU 的应用程序进行汇编/反汇编，通常也将这种在不同型号微机上的汇编称为交叉汇编。例如，用 Intel 公司的 ASM－51 汇编/反汇编软件在 IBM-PC 上对 MCS－51 单片机的应用程序进行交叉汇编。

③ 对应用程序进行排错调试就是对已经进行过硬件检查的试验样机和翻译成机器码的应用程序，还要进行联合排错和调试检查。目前常用的排错、调试方法有两种。一是用单片机仿真开发装置与试验样机联机提供排错、调试手段，具体的方法有单步运行、断点运行、跟踪运行和全速运行等。其中单步运行方法是使所编制的程序指令仅执行一条就停止下来，检查试验样机和应用程序中的错误，然后再单步执行下一条指令…… 断点运行方法是在程序中设置断点，使得当程序执行到断点处时停止，供设计者检查试验样机和应用程序中的错误 跟踪运行方法是应用程序指令一条一条地执行，开发装置摄取运行每条指令的地址、单片机各部分数据、I/O 接口等处信息，供调试者随时停止程序，对各种信息进行检查和修改 ;全速运行方法是实时地运行用户程序，可以检查用户程序最终执行结果。二是在 IBM－PC 上创建一个模拟目标单片机的模拟环境，把编好的程序在这个模拟环境下运行，进而进行排错和调试。该方法对单片机软件的开发简单易行，它不需要任何在线仿真器，也不需要试验样机。

1.3　TDN－51 开放式单片机教学实验系统

市场上，MCS－51 单片机教学实验系统种类很多，用户只要掌握了一种实验系统的使用方法，就可以方便地理解和使用其他种类的实验系统，因它们的功能都差不多，只是操作方面有所差别而已。这里主要介绍 TDN－51 开放式单片机教学实验系统及其使用方法。

1．TDN－51 开放式单片机教学实验系统组成

该系统由 CPU 单元、8155 单元、键盘显示单元、电动机单元和 6264 扩展单元等组成，如图 1－1 所示。

2．TDN－51 开放式单片机教学实验系统的操作

⑴ 使用串行通信电缆将实验系统与 PC 连接。

⑵ 将系统状态选择开关拨至 51（此开关位于实验箱右下角），开启实验系统电源开关。

⑶ 进入系统：启动 PC，进入 TD51 子文件夹，双击 MD51 图标，屏幕显示如图 1－2 所示。

根据所使用的通信接口选择输入“1”或“2”，按“Enter”键，便可进入 TDN－51 系统集成软件环境。屏幕显示如图 1－3 所示。

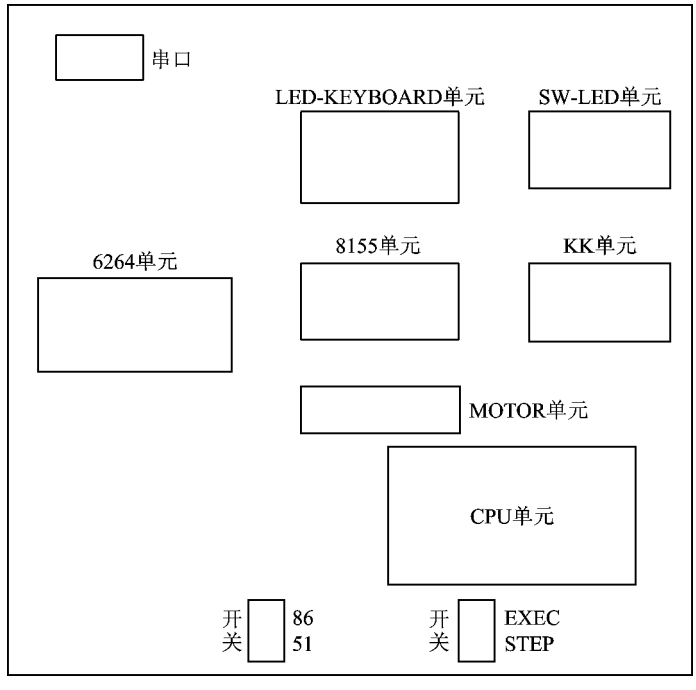


图 1－1　TDN－51 实验系统组成图

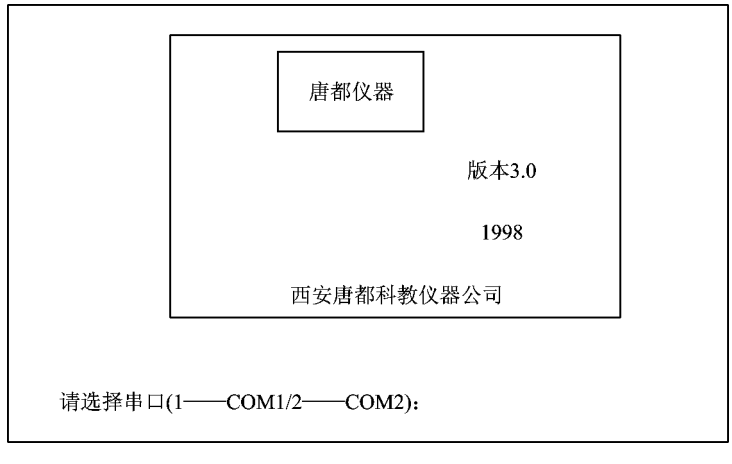


图 1－2　进入系统

屏幕上端为菜单栏，用户可通过键盘各功能键选择使用编辑、汇编和调试等功能；屏幕中部为调试窗口 Debug 和寄存器、标志位显示窗口 REGS；屏幕下端为信息栏，显示当前提示信息。

(4) 全屏幕编辑 (edit)

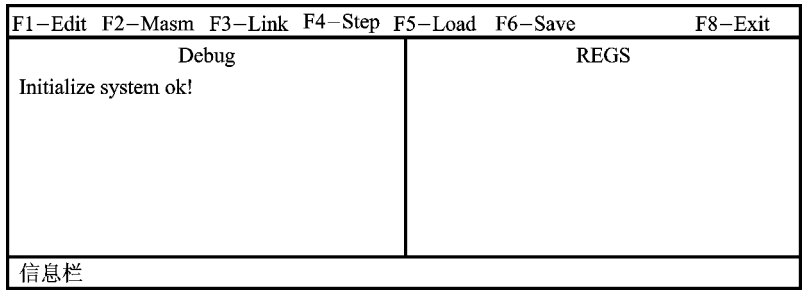


图 1－3　进入软件环境

在图 1－3 所示的操作界面中，按“F1”键，进入全屏幕编辑操作，屏幕显示如图 1－4 所示。

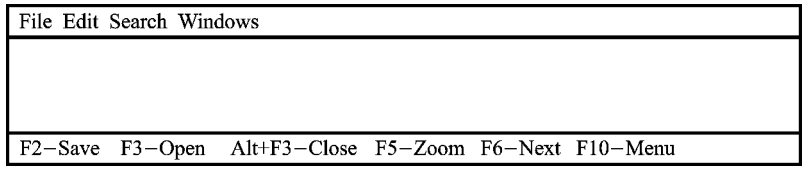


图 1－4　全屏幕编辑操作

通过菜单或快捷键选择需要的功能，可对源程序进行输入、修改和保存等操作。在对源程序进行修改的过程中，用户应先定义需要修改的程序块，然后可使用“Edit”菜单中的复制、剪切、粘贴和恢复等功能。另外，此编辑系统提供了搜索功能，只需在对话框中输入所要查询的程序内容，系统便可自动进行搜索，查找成功后，由光标进行定位并在编辑窗口显示；同时也可以选择此功能下拉菜单中的替换操作，只需在对话框中输入当前语句和替换后的语句内容，系统就会自动完成程序内容的替换。

⑸ 汇编（Masm）

在图 1－3 所示的界面中，按“F2”键，进入汇编状态，信息栏提示：Filename：…，输入待汇编的文件名（省略扩展名时系统默认为“.asm”）后按“Enter”键，系统自动完成汇编并生成“＊.r03”文件和“.M”文本文件，并在屏幕上显示版权及如图 1－5 所示汇编信息。

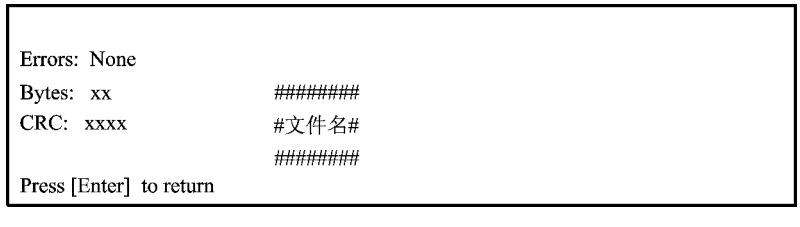


图 1－5　进入汇编状态

若在汇编过程中出现错误，则错误信息会保存在“.M”文件中，用户可在编辑功能中同时打开这个文件和源程序文件，并可用“F6”键在两者间切换，对照出错信息，修改程序。

⑹ 连接（Link）

在图1－3所示的界面中，按“F3”键，进入连接状态，信息栏提示：Filename：…，输入待连接的文件名后按“Enter”键，自动连接生成一个扩展名为“.L”的文本文件，包含了连接信息供用户查询。用户也可在编辑功能中同时打开这个文件及远程文件，并按“F6”键在两者间切换。

⑺ 装入（Load）

在图1－3所示的界面中，按“F5”键，进入装载状态，信息栏提示：Filename：…，输入“文件名.HEX”后按“Enter”键，PC即开始将程序从磁盘装入到教学系统内存，装载完后屏幕显示“LOAD OK”。

⑻ 调试命令

8051Debug界面命令提示符为“-”，进行Debug调试的主要命令如下：

① B 断点设置。在系统提示符下，输入B ↙，系统提示［i］：，等待输入断点地址。输入后按“Enter”键，系统继续提示［i＋1］：。若直接用“Enter”键来响应，则结束该命令。系统允许设至多10个断点，断点的清除只能是通过系统复位或重新通电来实现。需要说明的事，断点不允许设在以下位置：

- MOVC指令所涉及的存储地址；
- 距下一语句标号的距离小于3。

② D 显示一段地址单元中的数据。

格式为：D［起始地址，［尾地址.］］↙

D命令执行后显示一段地址单元的数据，在显示过程中，可用“Ctrl＋S”来暂停显示，用任意键继续 ;也可用“Ctrl＋C”中止数据显示，返回监控状态。

③ E 编辑指定地址单元中的数据。

格式为：E［地址］ ↙

该命令执行后，则按字节显示或修改数据，可通过“空格”键使单元地址向高地址方向移动，也可用“-”键使单元地址向低地址方向移动，并可直接填入新数据来修改地址单元中的内容。若按“Enter”键，则退出E命令。

④ G 运行程序。

格式为：G＝地址↙

GB［＝地址］ ↙

其中，G格式表示无断点连续运行程序，GB格式表示带断点运行程序。连续运行过程中，当遇到断点或按“Reset”键时，终止程序运行。另外，系统设置了STEP（单步）和EXEC（连续）运行方式选择开关，开关拨至STEP位置时，INT0引脚被占用，仅当开关拨至EXEC位置时，全

部引脚均开放。

⑤ R 寄存器或片内 RAM 区显示与修改。

格式为：R ↙或 R07 ↙

R ↙则按下面格式显示当前单片机的状态：

A = XX　B = XX　PSW = XX　SP = XX　DPTR = XXXX　PC = XXXX

R07XX ↙（XX = 00 ~ FF），则可显示/修改单片机片内 RAM 单元的内容，每次只能一字节一字节的显示/修改。进入此状态后，就可通过“空格”键使地址向高地址方向移动；而“-”键使地址向低地址方向移动，也可直接填入新数据来修改当前地址单元中的内容；若直接按“Enter”键响应，就退出 R 命令。单片机专用寄存器的显示/修改亦如此进行。

⑥ T 单步运行指定的程序。

格式为：T［= 地址］↙

每次按照指定的地址或 IP/PC 指示的地址，单步执行一条指令后则显示运行后的 CPU 寄存器情况。系统规定，STEP/EXEC 运行方式选择开关拨至 STEP 状态，否则单步等同于 G 命令。

⑦ U 反汇编。

格式为：U［起始地址［,尾地址］］↙

⑧ X 读/写外部数据区的内容。

格式为：X ↙或 X（地址）↙

⑼ 退出系统（Exit）

在图 1－3 所示的界面中，按“F8”键，退出并返回操作系统。

1.4　TKS－52B 仿真器的使用

单片机应用系统建立后，应用系统的电路是否正确、应用系统的用户软件是否正确，需要通过一定的手段进行调试，以修正软、硬件错误。最直接有效的手段就是采用仿真器进行仿真调试，即把应用系统的单片机芯片拔掉，用仿真系统代替单片机对应用系统进行调试。下面以广州致远电子有限公司的仿真器 TKS－52B 为例，简单说明仿真器的使用方法。

1. TKS－52B 仿真器简介

TKS-52B 仿真器硬件上采用 Philips 公司 MCU 设计部门的经验，采用超高稳定 I/O 总线设计，可靠仿真标准 P0/P2 口特性；另外仿真绝对不占用任何用户资源，包括堆栈/内部 RAM/SFR 等。具备高度运行稳定性/芯片兼容性，运行频率达 32 MHz，低电压仿真方面性能卓越，可以稳定运行在 2.0 V 以下。内部的部件经过全面优化后，能以较低的价格支持多项高级仿真功能。支持多种 80C51 单片机的仿真。

软件上支持 Keil 开发/调试平台（简称 Keil 平台），真正无缝嵌接 Keil 平台，实现多种高级调试功能。

2．仿真器硬件系统组成

在 Keil 平台的支持下，TKS-52B 仿真器和计算机经 RS－232C 接口相连后，便构成了 TKS－52B 在线仿真系统，通过仿真头插接到 MCS－51 应用系统，便可实现高级开发功能，从而提高样机研制工作的效率。在线仿真系统的组成如图 1－6 所示。

图 1－6　TKS－52B 在线仿真系统的组成

⑴ TKS－52B 硬件构成和作用

仿真器主机：完成仿真功能的最主要部分。监控指示 LED 灯点亮表示处于监控状态；运行指示 LED 灯点亮表示进入运行状态；电源指示 LED 灯点亮表示系统电源正常；仿真电缆插头插入仿真电缆连接到仿真头；串口插座插入串口通信电缆连接到计算机；电源插座输入仿真器主机工作需要的电源。

仿真器电源：为仿真器主机提供工作电源，它的输入电源为交流 220 V。

仿真头：仿真头带有针状插座，可以插入应用系统目标板的插座中，它的另外一端通过仿真电缆连接到仿真主机，这样把仿真主机和应用系统目标板连接起来。仿真头结构示意图如图 1－7 所示。

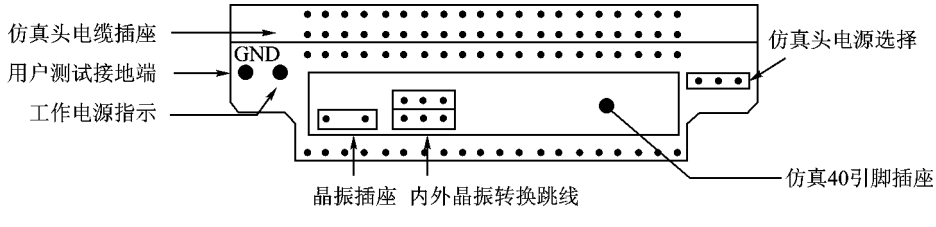


图 1－7　仿真头结构示意图

① 仿真头电缆插座：使用时用仿真电缆连接仿真器和仿真头。

② 用户测试接地端：进行测试时，可以使用该端作为地参考点。

③ 仿真 40 引脚插座：用于连接仿真头和应用系统目标板。

④ 工作电源指示：如果仿真头有电源接入（仿真器内部或外部应用系统目标板）则该 LED 点亮。

⑤ 内外晶振转换跳线：用于转换当前使用的晶振。当该跳线位于右选择时，选择使用应用系统目标板上的晶振；当该跳线位于左选择时，选择仿真头晶振插座的晶振。应用系统目标板上的晶振和仿真头上的晶振可以同时存在。应用系统目标板上必须有 10～50 pF 的补偿电容，并根据使用振荡频率而变化。

⑥ 晶振插座：用于插入晶振。

⑦ 仿真头电源选择：用于选择仿真头的工作电源。当向左选择时，使用仿真器提供的 +5 V电源；当向右选择时，使用外部应用系统目标板上输入的电源。当左右都不选择时，仿真头上的电路不工作。具体使用方法如下：若使用仿真器内部时钟，可以左右都不选择。由于仿真头上无电源，所以时钟电路不工作，这样可以减少干扰提高稳定性；若使用仿真头上产生的时钟且仿真器使用内部电源，一定向左选择使用仿真器内部电源；若使用仿真头上产生的时钟且仿真器使用应用系统目标板上提供的电源，在外部电源在 4～5 V 的稳定电压下可以考虑向右选择使用应用系统提供的电源，如果应用系统的电压小于 4 V 建议仍然向左选择使用仿真器内部电源。

注意：尽量不要使用应用系统上的电源对仿真头振荡电路供电。

通信电缆：用于连接仿真器主机和计算机，通信电缆为 RS－232 串行电缆。

⑵ 仿真头的使用方法和注意事项

在使用仿真器内部时钟时，仿真头上的时钟电路没有必要使用，因此可将仿真头电源选择的短路跳线取下，这样仿真头上的电路将失去工作电压而不工作。这在仿真一些对干扰敏感的应用系统中是有好处的。

如果不使用仿真器内部提供的时钟，可以选择外部时钟。外部时钟可以有两种选择方案：

① 使用仿真头上的时钟振荡电路，晶振需要外接。晶振接入的位置有两种选择：在仿真头上的晶振插座插入（内外晶振转换跳线需要向左选择）；在应用系统目标板上提供接入（XTAL1 和 XTAL2 间，内外晶振转换跳线需向右选择）。

② 使用应用系统目标板上提供的时钟（不是晶振），必须从 XTAL1 接入时钟信号。内外晶振转换跳线需要向右选择。

注意：在使用仿真器外部时钟（包括从用户板上直接输入时钟）时都要使用仿真头上的时钟振荡电路，因此仿真头电源选择必须向左或向右选择，不能悬空。

如果选择了使用外部电源，则仿真芯片工作电源将由应用系统的目标板提供（电压范围为 2.0～5.5 V），从仿真 40 引脚插座的第 40 引脚输入。

⑶ TKS－52B 仿真器对计算机的要求

计算机是使用 TKS－52 仿真器所必需的工具，计算机最低的配置要求：

① Pentium、Pentium－II 或兼容处理器的 PC；

② Windows 95 以上的操作系统；

③ 至少 16 MB 的内存；

④ 20 MB 以上的硬盘空间；

⑤ 至少带有一个串行接口。

3. 仿真调试软件

TKS－52B 仿真器无缝嵌接 Keil 平台，利用 Keil IDE（μVision2/3）集成开发环境可以对基于 80C51 内核的微处理器进行软、硬件仿真，具体到从工程建立和管理，编译、连接、目标代码的生成、软件仿真和硬件仿真等整个完整的开发流程。

（1）进入 μVision2 环境

当得到该软件的安装软件后，可以按照提示进行安装。安装完毕后就可以在这套功能强大的开发平台上完成所有的设计工作，包括源程序的编写、程序的编译和连接、软件模拟仿真的验证和排错，最后可以使用仿真器在 Keil 平台中进行硬件的仿真和调试，直到最终完成设计。安装完成后，可以点击运行图标进入 IDE 环境（μVision2），如图 1－8 所示。

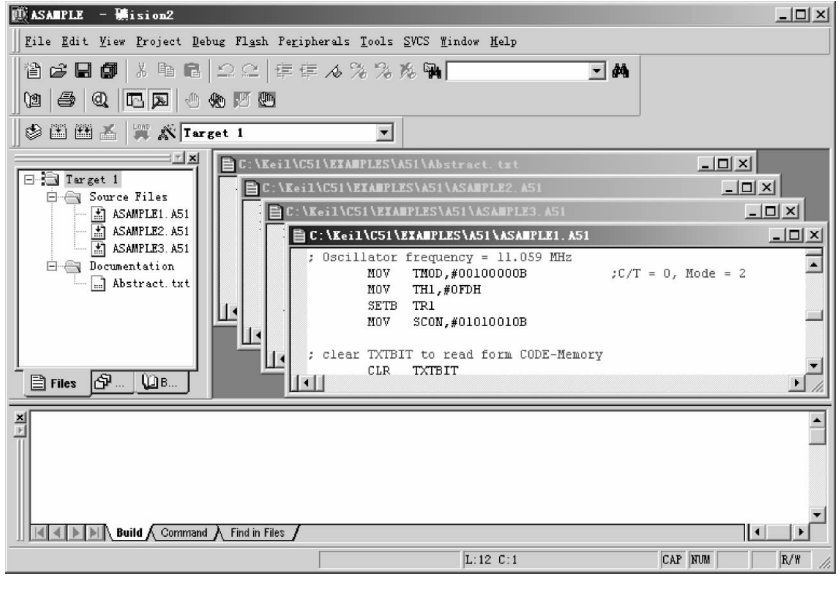


图 1－8 μVision2 操作界面

μVision2 软件有菜单栏、可以快速选择命令按钮的工具栏、一些源代码文件窗口、对话框窗口、信息显示窗口。软件允许同时打开几个源程序文件。菜单栏提供了各种操作菜单，比如

编辑器操作、工程维护、开发工具选项设置、程序调试、窗体选择和操作、在线帮助。工具栏按钮可以快速执行 μVision2 命令。快捷键也可以执行 μVision2 命令。

(2) Keil 软件开发的流程

① 建立工程；

② 为工程选择目标器件，例如选择 ATMEL 的 AT89C52；

③ 设置工程的配置参数；

④ 打开/建立程序文件；

⑤ 编译和连接工程；

⑥ 纠正程序中的书写和语法错误，并重新编译连接；

⑦ 对程序中某些纯软件的部分使用软件仿真验证；

⑧ 使用 TKS－52B 硬件仿真器对应用程序进行硬件仿真；

⑨ 生成 Hex 文件。

上面的流程只是一个标准的开发流程，实际中可能反复重复一个或几个步骤。

通过 Keil (μVision2) 软件配合由应用系统的软、硬件、仿真器、计算机组成的在线仿真系统，按照以上 Keil 软件开发的流程多次调试修正软、硬件，最后生成应用系统正确、完善的程序目标代码。然后再用编程器将目标代码写入到应用系统的程序存储器，即可完成应用系统的整个设计过程。

对于 Keil 软件的具体使用以及结合仿真器的硬件仿真，可参看第 6 章的介绍。

1.5　EasyPRO 800 编程器的使用

应用系统设计的用户软件，在经过软件或硬件仿真调试和全部修改完成后，需要将用户软件固化到程序存储器 (EPROM、EEPROM、Flash 等) 中，固化完成的程序存储器或 MCU (内部含程序存储器) 插入到应用系统的目标板上，应用系统才能脱离仿真开发环境独立工作。用户软件的固化需要通过编程器才能实现，下面以广州致远电子有限公司推出的通用编程器 EasyPRO 800 说明编程器的使用方法。

1. EasyPRO 800 编程器简介

EasyPRO 800 编程器具有 USB 通信、48PIN 引脚万能全驱、1.8 V 超低压芯片编程、引脚检测功能、可靠检测芯片的反插，错插及坏片、全面提升 OTP 芯片烧录的合格率、插片自动启动功能支持自动烧写等特点；支持编程的芯片也比较全面：支持 PHILIPS、ATMEL、WINBOND、HY、MICROCHIP、HOLTEK、SST、ICSI、SYNCOMS、DALLAS 等各厂商大多数 MCU 芯片；支持 26、27、28、29、37、39、49 等系列 EPROM、EEPROM、Flash 芯片；同时支持 24、25、34、45、93 系列芯片。

2. EasyPRO 800 编程器的控制软件

EasyPRO 800 编程器的控制软件安装非常简单，只要双击安装文件即可根据提示向导自动安装 需要注意的是由于采用 USB 接口和计算机通信，所以最后需要根据提示安装该公司提供的 USB 驱动程序。安装完成后可单击控制软件启动界面，如图 1－9 所示。

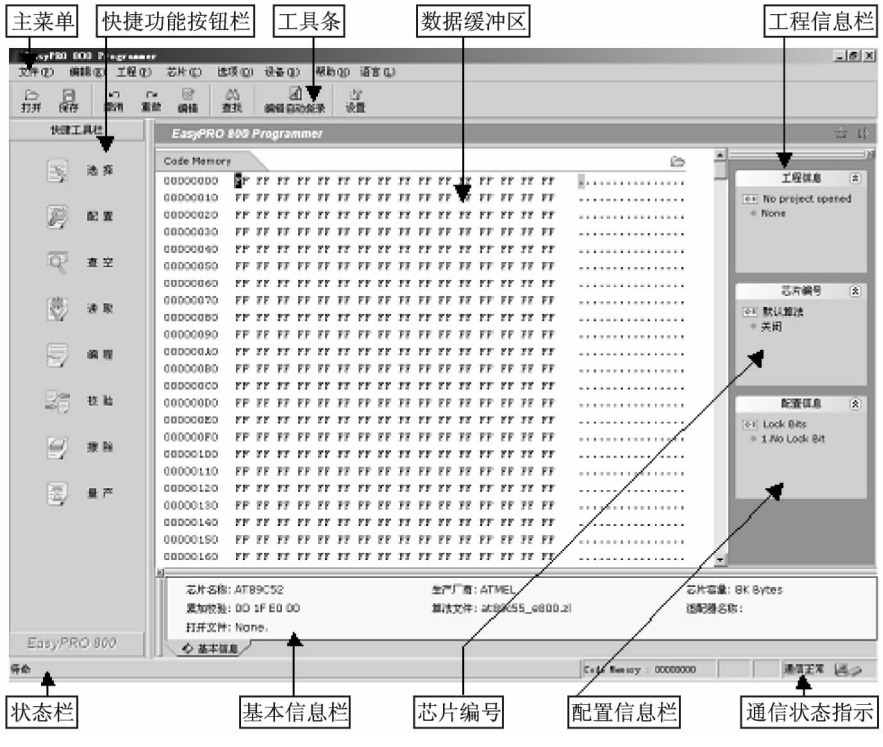


图 1－9　编程器控制软件界面

主菜单：EasyPRO 800 编程器应用软件的所有操作命令都可以在主菜单中找到并执行。

工具条：工具条上提供了打开，保存等常用命令。

快捷功能按钮栏：在该功能按钮栏中可方便地选择编程，校验等操作命令。

信息栏：该栏中 Code Memory 指示当前光标位置，NUM 指示键盘 NumLock 是否按下，如果键盘处于大写锁定状态则会出现 CAPS 字样。

基本信息栏：提示当前所编程芯片基本信息。

通信状态指示：用于指示当前 USB 通信状态，当图标上计算机与芯片之间连接出现红色叉号则表示通信出错。

配置信息栏：显示当前被编程芯片所设置的配置信息。

芯片编号：显示批量编程时芯片编号信息。

工程信息栏：显示工程信息。

数据缓冲区：显示需要烧入芯片中的数据。

3．编程器编程步骤

(1) 硬件准备

在正确安装了 USB 驱动程序，以及 EasyPRO 800 编程器应用软件之后，连接好电源适配器及 USB 通信线。此时编程器的红色指示灯亮指示电源连接正常，绿色编程结果指示灯处于熄灭状态，编程器应用软件下方通信状态指示通信正常。

按编程器锁紧座旁所标示方向正确放置芯片。如果出现芯片反放或错放情况，编程器的引脚检测功能将加以提示。

(2) 选择器件

单击快捷功能按钮栏中“选择”按钮或选择主菜单“芯片”下的“选择芯片”，弹出器件选择窗口，如图 1－10 所示。

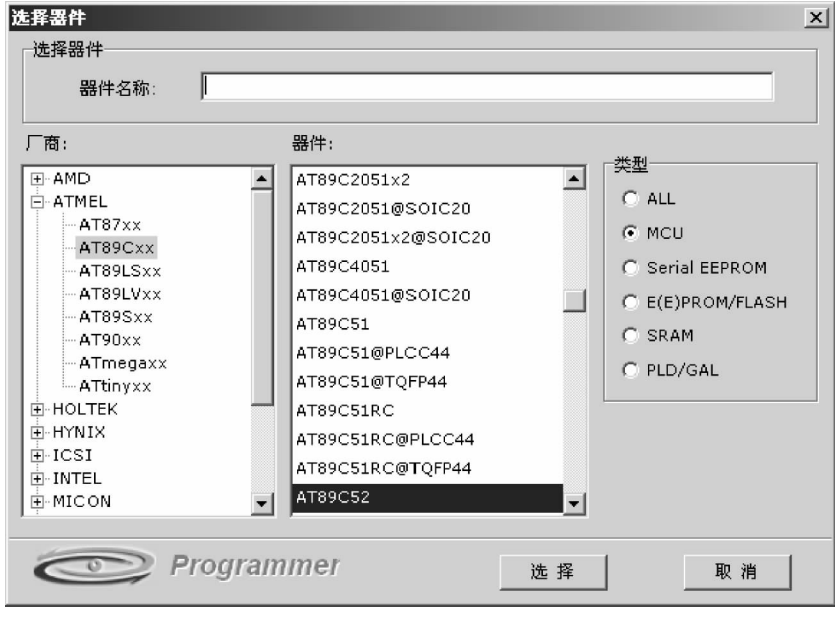


图 1－10　选择器件

首先应在类型栏中选择器件类型，如 MCU、E (E) PROM/FLASH、PLD/GAL 等，然后在厂商选项栏中选择厂家，并在右侧出现的器件选项栏中选择器件，或者在查找栏输入器件名称，快速选择器件。选择器件后，按“选择”按钮，或者双击器件名选中器件，此时“选择器件”对话

框自动关闭，回到用户操作界面，在用户界面下方信息栏中将提示用户所选器件的基本信息。

提示：该窗口支持模糊查找。如查找 AT89C52，在器件名称后的文本框内输入 89C52 或 AT89C52 等均可查到，但注意选对“芯片类型”。

注意：

① 请正确选择芯片型号，否则极易导致编程器和芯片损坏；

② 在选择芯片后，“程序代码缓冲区”中的内容将被清空，所以一定要先选择芯片，再打开所要编程的文件。

⑶ 将数据装入缓冲区

只有将所需要烧写到芯片的数据装载到数据缓冲区，才能执行编程操作。单击菜单工具栏中的“打开”按钮，弹出“调入选择”对话框，如图 1－11 所示。

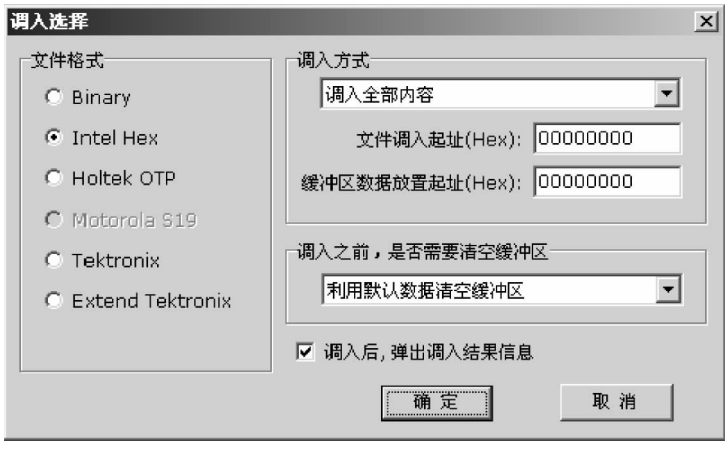


图 1－11 调入选择

从中选择文件格式、调入方式、调入地址，并可选择是否在调入前清空缓冲区、调入后弹出结果信息。设定后单击“确定”按钮，此时将出现提示对话框对所进行选择作出提示。根据提示确定无误后，单击“确定”按钮，则需编程数据将被装入缓冲区，并出现在用户界面的数据缓冲区内。

提示：如果所调入的数据量大于芯片容量，编程器将会提示缓冲区溢出。

⑷ 设置芯片配置信息及加密选项

如果需要对所编程芯片进行加密或其他配置操作，单击“配置”按钮，弹出配置栏对话框。配置信息因芯片不同而不同，如果当所选芯片为 AT89C52 时，将弹出如图 1－12 所示的对话框。

根据实际需要，对芯片配置字进行具体设置，然后单击“设定”按钮，完成设置配置字操作。此时所设定的配置信息将会出现在用户操作界面右侧的配置信息栏中。

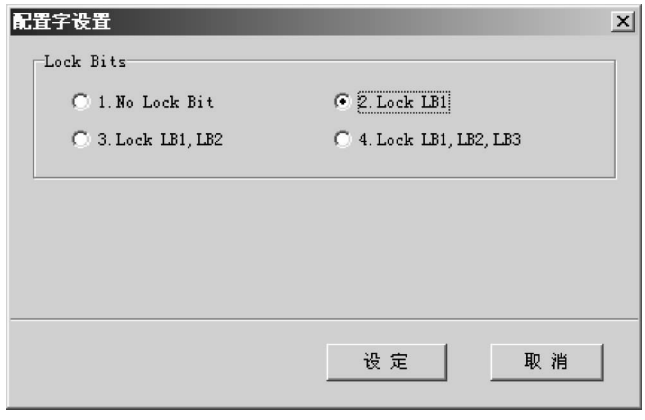


图 1－12　设定配置字

提示：配置字的设置对于每个芯片来说相当重要，请在编程芯片前慎重设置。某些类型的芯片可能不含配置信息。

⑸ 设置编程操作选项

单击“编程”按钮，弹出“编程”对话框，如图 1－13 所示。

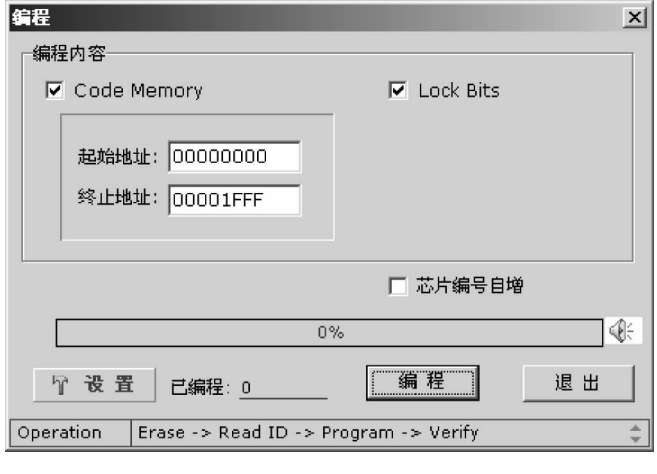


图 1－13　编程操作

在该编程对话框中进行编程设置。

Code Memoey：编程程序代码区，可输入起始以及终止地址来限定数据缓冲区中被编程数据的范围。

配置选项：编程已经设定的配置信息。由于不同芯片的配置信息不同，使得配置选项中编程内容也不同，如图 1－13 中配置选项内容为“Lock Bits”。

芯片编号自增：在指定的位置以默认算法写入数据，使烧录后的每片芯片都有不同的标号。

蜂鸣器提示：设定在操作成功或失败后，是否需要蜂鸣器发声提示。如果不希望声音提示，则单击对话框中黄色喇叭，随之会有红色叉号标示禁止声音提示。

已编程：提示已编程芯片数目，该值可由用户自行设置。

如果还需进行编程高级设置，则可单击“设置”按钮打开设置对话框，如图 1－14 所示。

图 1－14　编程高级设置

从该对话框中选择所需操作选择后，单击“设定”按钮。如果设置错误则单击“取消”按钮重新进行设置。

⑹ 编程芯片

正确完成上述步骤后，便可以单击编程对话框中“编程”按钮对芯片进行编程。

上述步骤为编程芯片的必须过程，根据不同芯片的情况还需增加不同的操作步骤，在应用软件的快捷功能按钮栏中也提供了如查空、校验和擦除等功能按钮。也可以在编程高级设置中选定上述操作，如对 AT89C52 芯片可以自动编程，即擦除、查空、编程、校验四个步骤都选中在编程时一次操作完成。

以上是 EasyPRO 800 编程器的一般使用介绍，详细的操作请参照该产品说明书或者 EasyPRO 800 应用软件的帮助说明。

第 2 章　MCS－51 单片机基本应用编程与调试实验

实验 1　内部数据传送指令实验

一、实验目的

① 掌握数据传送指令的功能，识别各种传送指令助记符；
② 明确各种传送指令的寻址方式。

二、实验要求

① 给出操作数的寻址方式，并根据指令功能写出结果；
② 在单步运行指令之前，先检查源操作数和目的操作数单元中的内容；
③ 单步运行程序，观察是否传送成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

序号	存储地址	机器码	源程序	
1	2000		ORG	2000H
2	2000	7860	MOV	R0,#60H
3	2002	7961	MOV	R1,#61H
4	2004	E8	MOV	A,R0
5	2005	E570	MOV	70H,A
6	2007	857071	MOV	71H,70H
7	200A	7470	MOV	A,#70H
8	200C	F6	MOV	@R0,A
9	200D	E570	MOV	A,70H
10	200F	F9	MOV	R1,A

序号	存储地址	机器码	源程序	
11	2010	E6	MOV	A,@R0
12	2011	FD	MOV	R5,A
13	2012	905000	MOV	DPTR,#5000H
14	2015	8D40	MOV	40H,R5
15	2017	8650	MOV	50H,@R0
16	2019	755588	MOV	55H,#88H
17	201C	A640	MOV	@R0,40H
18	201E	7738	MOV	@R1,#38H
19	2020	80FE	L1：SJMP	L1
20	2022		END	

五、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；

② 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并记录下每条指令的运行结果（即目的地址单元的内容）。例如：

运行第 1 条指令　2000　7860　MOV　R0,#60H

只须输入 T＝2000 ↙，则第 1 条指令运行

输入 R0700 ↙，检查 R0 的内容，为（R0）＝60H。

运行第 2 条指令　2002　7961　MOV　R1,#61H

只须输入 T＝2002 ↙，则第 2 条指令运行

输入 R0701 ↙，检查 R1 的内容，为（R1）＝61H。

以此类推，可单步运行每条指令并查看其结果。

六、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② 每条指令的执行结果与分析的结果一致吗？为什么？

③ 怎样判断程序已装载成功？

实验 2　堆栈及交换指令实验

一、实验目的

① 掌握堆栈的概念和堆栈的结构；

② 掌握堆栈的操作过程和堆栈指令的功能；

③ 掌握交换指令的功能；

④ 进一步熟悉单步运行调试程序和检查指令执行的结果的方法。

二、实验要求

① 给出操作数的寻址方式，并根据指令功能写出结果；

② 检查堆栈的栈底所在单元，观察进栈、出栈后堆栈指针 SP 的内容和目的地址单元的数据。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

序号	存储地址	机器码	源程序	
1	1000		ORG	1000H
2	1000	758130	MOV	SP,#30H
3	1003	74AA	MOV	A,#0AAH
4	1005	7820	MOV	R0,#20H
5	1007	753002	MOV	30H,#02H
6	100A	904000	MOV	DPTR,#4000H
7	100D	F0	MOVX	@DPTR,A
8	100E	F6	MOV	@R0,A
9	100F	C530	XCH	A,30H
10	1011	C083	PUSH	DPH
11	1013	C082	PUSH	DPL
12	1015	905000	MOV	DPTR,#5000H
13	1018	93	MOVC	A,@A+DPTR
14	1019	D082	POP	DPL
15	101B	D083	POP	DPH
16	101D	F530	MOV	30H,A
17	101F	D6	XCHD	A, @R0
18	1020	E0	MOVX	A,@DPTR
19	1021	C6	XCH	A,@R0
20	1022	80FE	L1：SJMP	L1

序号	存储地址	机器码	源程序	
			ORG	5000H
21	5000	33445566	DB	33H,44H,55H,66H
22	5003		END	

五、实验步骤

① 输入程序检查无误,经汇编、连接后装入系统；

② 用 T 命令单步方式依次运行程序中的每条指令 (将运行状态开关拨至 STEP),查看并记录下每条指令的运行结果 (即目的地址单元的内容)。例如：

运行第 1 条指令　MOV　SP,#30H

只须输入 T＝1000 ↙,则第 1 条指令运行

运行第 2 条指令　MOV　A,#0AAH

只须输入 T＝1003 ↙,则第 2 条指令运行

以此类推,可单步运行每条指令并查看其结果。

六、思考问题 (写出实验报告)

① 写出实验过程中所遇到的问题与解决的办法。

② 每条指令的执行结果与分析的结果一致吗？为什么？

③ 指出指令 MOV、MOVC 和 MOVX 的区别。

④ 在检查交换指令执行的结果时应注意什么？

实验 3　加法及十进制调整指令实验

一、实验目的

① 掌握加法指令的功能及对标志位的影响；

② 熟悉十进制调整指令的功能及应用场合；

③ 进一步熟悉单步运行调试程序和检查指令执行的结果的方法。

二、实验要求

① 给出操作数的寻址方式,并根据指令功能写出结果；

② 单步运行程序,并与分析结果相比较；

③ 如有错误,分析原因。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

(1) 程序 1：加法指令

序号	存储地址	机器码	源程序	
1	2000		ORG	2000H
2	2000	752011	MOV	20H,#11H
3	2003	7830	MOV	R0,#30H
4	2005	F8	MOV	R0,A
5	2006	28	ADD	A,R0
6	2007	2520	ADD	A,20H
7	2009	24CD	ADD	A,#0CDH
8	200B	38	ADDC	A,R0
9	200C	3520	ADDC	A,20H
10	200E	08	INC	R0
11	200F	08	INC	R0
12	2010	04	INC	A
13	2011	36	ADDC	A,@R0
14	2012	0520	INC	20H
15	2014	06	INC	@R0
16	2015	34B2	ADDC	A,#0B2H
17	2017	26	ADD	A,@R0
18	2018	904000	MOV	DPTR,#4000H
19	201B	A3	INC	DPTR
20	201C	F0	MOVX	@DPTR,A
21	201D	80FE	SJMP	$
	201F		END	

(2) 程序 2：十进制调整指令

序号	存储地址	机器码	源程序	
1	2000		ORG	2000H
2	2000	7475	MOV	A,#75H
3	2002	2499	ADD	A,#99H

序号	存储地址	机器码	源程序	
4	2004	D4	DA	A
5	2005	7870	MOV	R0,#70H
6	2007	38	ADDC	A,R0
7	2008	D4	DA	A
8	2009	2499	ADD	A,#99H
9	200B	D4	DA	A
10	200C	80FE	SJMP	$
11	200E		END	

五、实验步骤

① 输入程序检查无误,经汇编、连接后装入系统;

② 用 T 命令单步方式依次运行程序中的每条指令(将运行状态开关拨至 STEP),查看并记录下每条指令的运行结果(即目的地址单元的内容)。例如:

运行程序 1 的第 1 条指令　MOV　20H,#11H

只须输入 T=2000 ↙,则第 1 条指令运行

输入 R0720 ↙,检查 20 H 单元的内容,为(20 H)=11H。

运行第 2 条指令　MOV　R0,#30H

只须输入 T=2003 ↙,则第 2 条指令运行

R0 的内容为 30H。

以此类推,可单步运行每条指令并查看其结果。

六、思考问题(写出实验报告)

① 写出实验过程中所遇到的问题与解决的办法。

② 每条指令的执行结果与分析的结果一致吗?为什么?

③ 程序 2 中 DA　A 指令执行后标志位如何变化?与分析的结果是否一样?

④ 用 DA　A 指令的目的是什么?在什么情况下要用 DA　A 指令?

实验 4　减法指令实验

一、实验目的

① 掌握减法指令的功能及对标志位的影响;

② 进一步熟悉单步运行调试程序和检查指令执行的结果的方法。

二、实验要求

① 给出操作数的寻址方式，并根据指令功能写出结果；
② 分析减法运算指令对标志位的影响。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

序号	存储地址	机器码	源程序	
1	2000		ORG	2000H
2	2000	755058	MOV	50H,#58H
3	2003	797A	MOV	R1,#7AH
4	2005	E550	MOV	A,50H
5	2007	99	SUBB	A,R1
6	2008	9550	SUBB	A,50H
7	200A	19	DEC	R1
8	200B	1550	DEC	50H
9	200D	94F4	SUBB	A,#0F4H
10	200F	1550	DEC	50H
11	2011	94F4	SUBB	A,#0F4H
12	2013	19	DEC	R1
13	2014	14	DEC	A
14	2015	757824	MOV	78H,#24H
15	2018	97	SUBB	A,@R1
16	2019	F8	MOV	R0,A
17	201A	7648	MOV	@R0,#48H
18	201C	16	DEC	@R0
19	201D	80FE	SJMP	$
20	201F		END	

五、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；
② 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并

记录下每条指令的运行结果（即目的地址单元的内容）。例如：

运行第 1 条指令　MOV　50H,#58H

只须输入 T＝2000 ↙,则第 1 条指令运行

输入 R0750 ↙检查 50H 单元的内容,为（50H）＝58H。

运行第 2 条指令　MOV　R1,#7AH

只须输入 T＝2003 ↙,则第 2 条指令运行

检查 R1 的内容为（R1）＝7AH。

以此类推,可单步运行每条指令并查看其结果。

六、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② 每条指令的执行结果与分析的结果一致吗？为什么？

③ 减 1 指令影响标志位吗？利用减 1 指令对 DPTR 实现减 1 时应注意什么？

④ 如何用减法指令实现不带借位的减法？

实验 5　逻辑操作指令实验

一、实验目的

① 掌握逻辑操作指令的基本功能；

② 掌握逻辑运算指令对标志位的影响；

③ 进一步熟悉单步运行调试程序和检查指令执行结果的方法。

二、实验要求

① 给出操作数的寻址方式,并根据指令功能写出结果；

② 分析逻辑运算指令对标志位的影响。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

序号	存储地址	机器码	源程序	
1	2000	ORG	2000H	
2	2000	7498	MOV	A,#98H

序号	存储地址	机器码	源程序	
3	2002	7860	MOV	R0,#60H
4	2004	7920	MOV	R1,#20H
5	2006	7520AA	MOV	20H,#0AAH
6	2009	755019	MOV	50H,#19H
7	200C	59	ANL	A,R1
8	200D	49	ORL	A,R1
9	200E	F4	CPL	A
10	200F	5466	ANL	A,#66H
11	2011	4550	ORL	A,50H
12	2013	6250	XRL	50H,A
13	2015	23	RL	A
14	2016	535099	ANL	50H,#99H
15	2019	57	ANL	A,@R1
16	201A	F6	MOV	@R0,A
17	201B	08	INC	R0
18	201C	47	ORL	A,@R1
19	201D	F6	MOV	@R0,A
20	201E	08	INC	R0
21	201F	67	XRL	A,@R1
22	2020	F6	MOV	@R0,A
23	2021	80FE	SJMP	$
24	2023		END	

五、实验步骤

① 输入程序检查无误,经汇编、连接后装入系统;

② 用 T 命令单步方式依次运行程序中的每条指令(将运行状态开关拨至 STEP),查看并记录下每条指令的运行结果(即目的地址单元的内容)。例如:

运行第 1 条指令　MOV　A,#98H

只须输入 T=2000 ↙,则第 1 条指令运行

运行第 2 条指令　MOV　R0,#60H

只须输入 T=2002 ↙,则第 2 条指令运行

以此类推,可单步运行每条指令并查看其结果。

六、思考问题(写出实验报告)

① 写出实验过程中所遇到的问题与解决的办法。
② 每条指令的执行结果与分析的结果一致吗？为什么？
③ 二进制加法指令与逻辑或指令有何区别？

实验 6　控制转移指令实验

一、实验目的

① 掌握无条件转移指令和条件转移指令的基本功能和用法；
② 了解条件转移指令转移条件产生的方法；
③ 进一步熟悉单步运行调试程序和检查指令执行结果的方法；
④ 弄清转移指令所转向的目的地址。

二、实验要求

编写分支程序，实现下列功能。

(1) 二分支程序

$$y=\begin{cases}1 & x\geqslant 0\\ -1 & x<0\end{cases}$$

(2) 三分支程序

$$y=\begin{cases}2x & x>0\\ 2 & x=0\\ x/2 & x<0\end{cases}$$

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序及流程图

(1) 程序 1：二分支程序流程图及实验程序

编写程序时，为 x 和 y 分配存储单元，x 为片内数据存储器 30H 单元，y 为 31H 单元。程序流程图如图 2－1 所示。

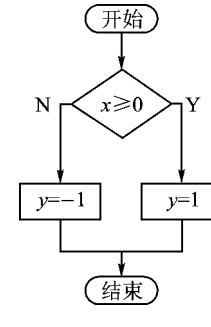


图 2－1　二分支流程图

实验程序清单：

序号	存储地址	机器码		源程序	
1	2000			ORG	2000H
2	2000	E530		MOV	A,30H
3	2002	30E704		JNB	ACC. 7,S1
4	2005	74FF		MOV	A,#0FFH
5	2007	8002		SJMP	S2
6	2009	7401	S1：	MOV	A,#01H
7	200B	F531	S2：	MOV	31H,A
8	200D	80FE		SJMP	$
9	200F			END	

(2) 程序 2：三分支程序流程图及程序清单

编写程序时，x 为 R1 工作单元，y 为片内 RAM 50H 单元。流程图如图 2－2 所示。

程序清单：

序号	存储地址	机器码		源程序	
1	2000			ORG	2000H
2	2000	E9		MOV	A,R1
3	2001	49		ORL	A,R1
4	2002	6006		JZ	S1
5	2004	5480		ANL	A,#80H
6	2006	6006		JZ	S2
7	2008	0112		AJMP	S3
8	200A	7402	S1：	MOV	A,#02H
9	200C	8008		SJMP	AD
10	200E	E9	S2：	MOV	A,R1
11	200F	23		RL	A
12	2010	8004		SJMP	AD
13	2012	E9	S3：	MOV	A,R1
14	2013	C3		CLR	C
15	2014	B3		CPL	C
16	2015	13		RRC	A
17	2016	F550	AD：	MOV	50H,A
18	2018	80FE		SJMP	$
19	201A			END	

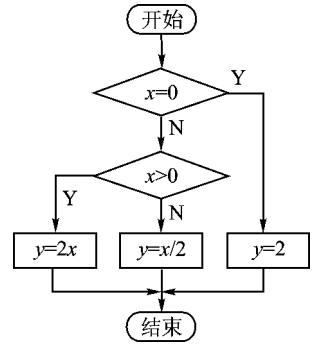


图 2－2　三分支程序流程图

五、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；

② 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并记录下每条指令的运行结果（即目的地址单元的内容）；

③ 对于二分支程序，分三个步骤进行：

- 给片内 RAM 30H 单元设置初始值为正数，如（30H）＝24H，方法为：在监控状态下输入 R0730 ↙，然后输入 34H ↙即可；单步运行程序，观察程序的执行过程，看分支走向是否正确；
- 重新开始，给片内 RAM 30H 单元设置初始值为 00H，执行过程应该与正数相同；
- 给片内 RAM 30H 单元设置初始值为负数，如（30H）＝84H，观察执行过程。

④ 对于三分支程序，分三个步骤进行：

- 给工作寄存器 R1 送入一个正数，如（R1）＝56H，方法为：在监控状态下输入 R0701 ↙，然后输入 56H ↙即可；单步运行程序，观察程序的执行过程，看分支走向是否正确；
- 给工作寄存器 R1 送入一个 00H，（R1）＝00H，观察是否将 2 送入 50H 单元；
- 给工作寄存器 R1 送入一个负数，如（R1）＝98H，单步运行程序，观察程序的执行过程，判断程序是否正确。

六、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② 在分支条件不同时，程序转向不同的分支，怎样完整调试一个分支程序？

③ SJMP 或 AJMP 指令在使用时要注意什么？试举例说明。

实验 7　位操作指令实验

一、实验目的

① 掌握位寻址区的地址空间分配；

② 熟悉位操作指令的功能及使用方法；

③ 进一步熟悉单步运行调试程序和检查指令执行结果的方法。

二、实验要求

① 分析指令功能并写出结果；

② 单步执行指令，与分析结果比较。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序

序号	存储地址	机器码	源程序	
1	2000		ORG	2000H
2	2000	759038	MOV	P1,#38H
3	2003	752035	MOV	20H,#35H
4	2006	753131	MOV	31H,#31H
5	2009	B205	CPL	05H
6	200B	A202	MOV	C,02H
7	200D	8207	ANL	C,07H
8	200F	74AB	MOV	A,#0ABH
9	2011	2520	S2：ADD	A,20H
10	2013	4002	JC	S1
11	2015	0531	S4：INC	31H
12	2017	D290	S1：SETB	P1.0
13	2019	2590	S3：ADD	A,P1
14	201B	50F4	JNC	S2
15	201D	B0E3	ANL	C,/ACC.3
16	201F	20E2F7	JB	ACC.2,S3
17	2022	72E7	ORL	C,ACC.7
18	2024	40EF	JC	S4
19	2026	80FE	SJMP	$
20	2028		END	

五、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；

② 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并记录下每条指令的运行结果（即目的地址单元的内容）；

③ 判断转移指令转移的条件是否满足，是否转到相应位置。

六、思考问题（写出实验报告）

① 位数据传送指令的传送途径是什么？能操作的地址空间有哪些？

② 判位转移指令有哪些？每条指令的功能是什么？

③ 使用位操作指令实现 P1.1 =（ACC.0 ∧ /B.2）∨ B.3 逻辑表达式的计算，编写程序并调试结果是否正确。

实验 8　数码转换程序实验

一、实验目的

① 掌握不同进制数及其编码相互转换——数码转换的程序设计方法，加深对数码转换的理解；

② 掌握连续运行和断点运行调试程序的方法；

③ 进一步掌握检查程序执行结果的方法。

二、实验要求

① 分析指令功能并写出结果；

② 单步执行指令，与分析结果比较；

③ 连续运行程序，检查结果；

④ 设置断点运行程序，检查程序结果。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验程序及流程图

① 程序 1：将累加器 A 中的 8 位二进制转换成 3 位 BCD 码格式的十进制数。其中百位数的 BCD 码放在 21H 单元中，十位和个位数放在 22H 单元中，流程图如图 2－3 所示。程序如下：

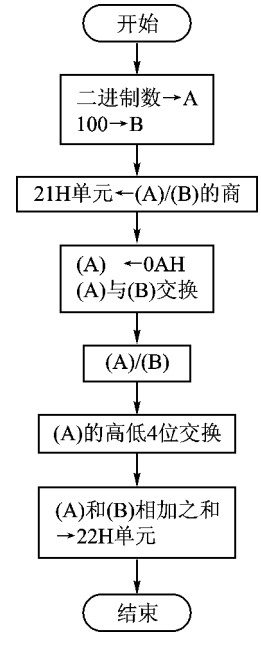


图 2－3　流程图

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	021000		LJMP	MAIN
3	1000			ORG	1000H
4	1000	74FF	MAIN：	MOV	A,#0FFH

序号	存储地址	机器码		源程序	
5	1002	75F064		MOV	B,#64H
6	1005	84		DIV	AB
7	1006	F521		MOV	21H,A
8	1008	740A		MOV	A,#0AH
9	100A	C5F0		XCH	A,B
10	100C	84		DIV	AB
11	100D	C4		SWAP	A
12	100E	25F0		ADD	A,B
13	1010	F522		MOV	22H,A
14	1012	80FE	L1：	SJMP	L1
15	1014			END	

② 程序 2：将累加器 A 中存放的两个 BCD 码拆开，求它们的乘积，并把乘积以压缩的 BCD 码形式送回累加器 A 中，流程图如图 2－4 所示。程序如下：

序号	存储地址	机器码		源程序	
1	0000	ORG		0000H	
2	0000	021000		LJMP	MAIN
3	1000			ORG	1000H
4	1000	7423	MAIN：	MOV	A,#23H
5	1002	75F010		MOV	B,#10H
6	1005	84		DIV	AB
7	1006	A4		MUL	AB
8	1007	75F00A		MOV	B,#0AH
9	100A	84		DIV	AB
10	100B	C4		SWAP	A
11	100C	45F0		ORL	A,B
12	100E	80FE		SJMP	$
13	1010			END	

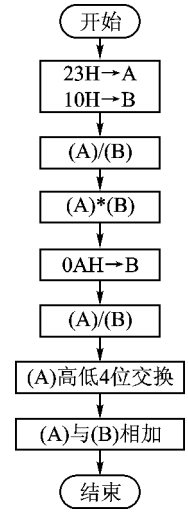


图 2－4　流程图

五、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统。

② 对程序 1 而言，分六步进行：

- 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并记录下每条指令的运行结果（即目的地址单元的内容）；

- 利用连续运行方式运行程序（将运行状态开关拨至 EXEC），在监控状态输入 G = 0000H ↙即可，按复位键返回监控状态，输入 R0721 ↙查看 21H 单元，输入 R0722 ↙查看 22H 单元；
- 在 1012H 处设断点；
- 利用断点运行方式运行程序，在监控状态输入 GB = 0000H ↙即可；
- 在监控状态输入 R0721 ↙查看 21H 单元，输入 R0722 ↙查看 22H 单元；
- 改变累加器 A 中的数，重新运行程序，考查程序的正确性。

③ 对程序 2 而言，分五步进行。

- 用 T 命令单步方式依次运行程序中的每条指令（将运行状态开关拨至 STEP），查看并记录下每条指令的运行结果（即目的地址单元的内容）；
- 在 100EH 处设断点；
- 利用断点运行方式运行程序，在监控状态输入 GB = 0000H ↙即可；
- 在监控状态输入 R ↙查看累加器 A 中的数；
- 改变累加器 A 中的数，重新运行程序，考查程序的正确性。

六、思考问题（写出实验报告）

① 如果想从键盘任意设置累加器 A 的内容，运行程序看不同的结果，程序怎样修改？

② 在程序 1 中，用 ADD 指令实现将十位和个位数放在累加器 A 中，在程序 2 中，用 ORL 指令实现将乘积以压缩的 BCD 码形式送回累加器 A 中。试问在什么情况下可这样做？此时需注意什么？

实验 9　多字节加法程序实验

一、实验目的

① 掌握带进位加法运算指令的使用；

② 学习 BCD 码运算十进制调整的方法；

③ 掌握循环结构程序的设计方法；

④ 掌握单步、连续运行和断点运行调试程序的方法。

二、实验要求

① 学习数据加法指令、十进制调整指令的用法；

② 按照实验内容编制实验程序；

③ 单步执行指令，与分析结果比较；

④ 连续运行程序，检查结果；

⑤ 设置断点运行程序，检查程序结果。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

设计并调试一个 n 字节的无符号十进制数加法程序，其功能为将（R0）和（R1）所指向的片内 RAM 中两个 n 字节压缩 BCD 码无符号十进制整数相加，其结果存放在被加数单元中，字节数 n 存放在 R2 中。

五、参考程序及流程图

流程图如图 2－5 所示。

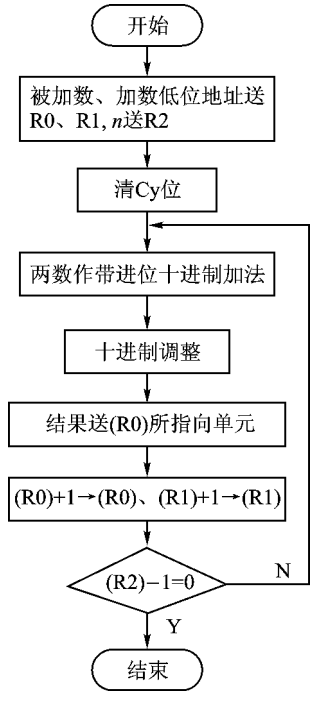


图 2－5　流程图

参考程序：设被加数存放在片内 RAM 60H 开始的存储单元，加数存放在片内 RAM 70H 开始的存储单元，数据长度是 6 字节。

序号	存储地址	机器码		源程序	
1	2000			ORG	2000H
2	2000	7860		MOV	R0,#60H
3	2002	7970		MOV	R1,#70H
4	2004	7A06		MOV	R2,#06H
5	2006	C3		CLR	C
6	2007	E6	SHI：	MOV	A,@R0
7	2008	37		ADDC	A,@R1
8	2009	D4		DA	A
9	200A	F6		MOV	@R0,A
10	200B	08		INC	R0
11	200C	09		INC	R1
12	200D	DAF8		DJNZ	R2,SHI
13	200F	80FE		SJMP	$
14	2011			END	

六、实验步骤

① 编写程序，检查无误，经汇编、连接后装入系统；
② 用 R07 命令给被加数、加数单元设置初始值；
③ 分别用单步运行、连续运行调试程序，验证程序的正确性。

七、思考问题（写出实验报告）

① 改变加数和被加数，测试程序执行结果。
② 把程序改成其他长度字节数相加，调试并验证结果。

实验 10　查表程序设计实验

一、实验目的

① 掌握查表指令的使用；
② 掌握查表程序的设计方法；
③ 进一步掌握单步、连续运行和断点运行调试程序的方法。

二、实验要求

① 学习 MOVC　A，@A＋DPTR 指令、MOVC　A，@A＋PC 指令的用法；
② 按照实验内容编制实验程序；
③ 单步执行指令，与分析结果比较；
④ 连续运行程序，检查结果；
⑤ 设置断点运行程序，检查程序结果。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

在程序中定义一个 0～9 的平方表，利用查表指令找出与累加器 A 的内容相对应的平方值，送到 R0。

五、参考程序

方法 1：利用 MOVC　A，@A＋DPTR 指令完成程序。程序如下：

序号	存储地址	机器码		源程序	
1	2000			ORG	2000H
2	2000	902007		MOV	DPTR,#TABLE
3	2003	93		MOVC	A,@A+DPTR
4	2004	F8		MOV	R0,A
5	2005	80FE		SJMP	$
6	2007	00010409	TABLE：	DB	0,1,4,9,16,25,36,49,64,81
	200B	10192431			
	200F	4051			
7	2011			END	

方法 2：利用 MOVC　A,@A+PC 指令完成程序。程序如下：

序号	存储地址	机器码		源程序	
1	2000			ORG	2000H
2	2000	2403		ADD	A,#03H
3	2002	83		MOVC	A,@A+PC
4	2003	F8		MOV	R0,A
5	2004	80FE		SJMP	$
6	2006	00010409	TABLE：	DB	0,1,4,9,16,25,36,49,64,81
	200A	10192431			
	200E	4051			
7	2010			END	

六、实验步骤

① 编写程序，检查无误，经汇编、连接后装入系统；

② 用 R07 命令给累加器 A 设置初始值；

③ 分别用单步运行、断点运行、连续运行方式调试程序，验证程序的正确性。

七、思考问题（写出实验报告）

① 通过上面两个程序，能否看出 MOVC　A,@A+DPTR 指令、MOVC　A,@A+PC 指令的区别？

② 把方法 1 中的程序修改一下，将常数表格放在程序存储器的任何地方，调试程序并验证结果。

实验 11　子程序设计实验

一、实验目的

① 了解子程序的结构；
② 掌握子程序设计的编程方法；
③ 掌握子程序结构程序的调试；
④ 熟悉主程序调用子程序及子程序返回主程序的过程。

二、实验要求

① 学习子程序设计的编程方法；
② 按照实验内容编制实验程序；
③ 单步执行指令，与分析结果比较；
④ 连续运行程序，检查结果；
⑤ 设置断点运行程序，检查程序结果。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

将片内 RAM 30H 存放的压缩 BCD 码转换成 ASCII 码，存放到 50H 开始的内存单元（50H 为高位），试编程实现。

五、参考程序

序号	存储地址	机器码		源程序	
1	0030			BCD	EQU 30H
2	0050			ASCII	EQU 50H
3	2000			ORG	2000H
4	2000	E530	START：	MOV	A，BCD
5	2002	C4		SWAP	A
6	2003	C0E0		PUSH	ACC
7	2005	122012		LCALL	S1
8	2008	D050		POP	ASCII

序号	存储地址	机器码		源程序	
9	200A	C030		PUSH	BCD
10	200C	1112		ACALL	S1
11	200E	D051		POP	ASCII + 1
12	2010	80FE		SJMP	$
13	2012	1581	S1：	DEC	SP
14	2014	1581		DEC	SP
15	2016	D0E0		POP	ACC
16	2018	540F		ANL	A,#0FH
17	201A	902025		MOV	DPTR,#ASCTAB
18	201D	93		MOVC	A,@A + DPTR
19	201E	C0E0		PUSH	ACC
20	2020	0581		INC	SP
21	2022	0581		INC	SP
22	2024	22		RET	
23	2025	30313233	ASCTAB：	DB	30H,31H,32H,33H,34H
	2029	34			
24	202A	35362538		DB	35H,36H,37H,38H,39H
	202E	39			
25	202F			END	

六、实验步骤

① 编写程序,检查无误,经汇编、连接后装入系统；
② 给每条指令加注释；
③ 根据参考程序画出流程图；
④ 检查长调用指令 LCALL 执行之前后堆栈的使用情况；
⑤ 检查返回指令 RET 执行之前后堆栈的使用情况；
⑥ 分别用单步运行、断点运行、连续运行方式调试程序,验证程序的正确性。

七、思考问题 (写出实验报告)

① 用指令 ACALL 调用子程序时,需注意什么？
② 子程序的一般结构及调用方法是什么？

第 3 章　MCS－51 单片机功能系统应用实验

实验 12　单片机外部中断的应用实验

一、实验目的

① 掌握外部中断功能系统的应用；
② 了解发光二极管驱动电路的设计与应用；
③ 了解按键电路设计。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照接线图连接好电路；
③ 用断点和连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

编写一个程序，要求每中断一次读取 P1.0 ~ P1.3 处开关状态，送 P1.4 ~ P1.7 时刻对应的发光二极管亮、灭。

五、参考程序及电路连线图

⑴ 参考程序

序号	存储地址	机器码	源程序	
1	0000		ORG	0000H
2	0000	020100	LJMP	MAIN
3	0003	ORG	0003H	
4	0003	4100	AJMP	S2
5	0100	ORG	0100H	

序号	存储地址	机器码		源程序	
6	0100	758160	MAIN：	MOV	SP,#60H
7	0103	D2B2		SETB	P3.2
8	0105	D288		SETB	IT0
9	0107	D2A8		SETB	EX0
10	0109	D2AF		SETB	EA
11	010B	210B	S1：	AJMP	S1
12	0200	ORG		0200H	
13	0200	74FF	S2：	MOV	A,#0FFH
14	0202	F590		MOV	P1,A
15	0204	E590		MOV	A,P1
16	0206	C4		SWAP	A
17	0207	F590		MOV	P1,A
18	0209	32		RETI	
19	020A			END	

(2) 电路连线图

电路连线图如图 3－1 所示。

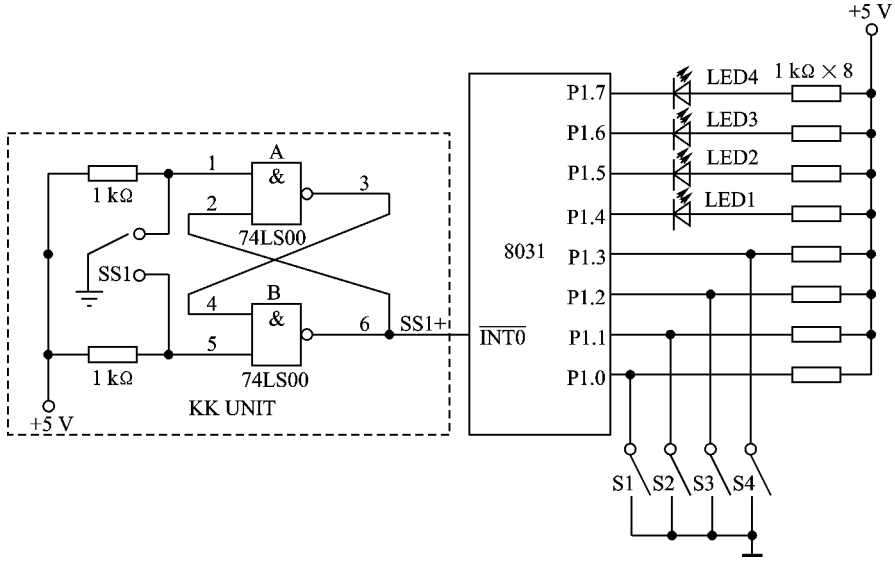


图 3－1　电路连线图

六、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；
② 按照接线图连接好电路；
③ 用断点和连续运行方式运行程序，按一次 SS1 开关，响应一次中断，观察 S1 ~ S4 开关状态与 LED1 ~ LED4 灯指示状态是否相符；
④ 改变 S1 ~ S4 开关状态，再按一次 SS1 开关，观察 S1 ~ S4 开关状态与 LED1 ~ LED4 灯指示状态是否相符。

七、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。
② 画出程序流程图。
③ 在程序中，改变哪一条指令就可改变为电平触发方式？

实验 13　单片机定时/计数器中断实验

一、实验目的

① 利用单片机的定时与中断方式，实现对信号灯的复杂控制；
② 通过定时器程序调试，学会定时器方式 1 的使用；
③ 进一步熟悉中断的基本概念。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照接线图连接好电路；
③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

利用定时器方式 1 编制 1 s 的延时程序，实现信号灯的循环显示控制。

五、实验参考程序及电路连线图

⑴ 利用定时器查询方式编制程序

系统采用 6 MHz 晶振，采用定时器 1，方式 1 定时 50 ms，用 R3 做 50 ms 计数单元。参考程序如下：

序号	存储地址	机器码		源程序	
1	1000			ORG	1000H
2	1000	7A07	CONT：	MOV	R2,#07H
3	1002	74FE		MOV	A,#0FEH
4	1004	F590	NEXT：	MOV	P1,A
5	1006	1116		ACALL	DELAY
6	1008	23		RL	A
7	1009	DAF9		DJNZ	R2,NEXT
8	100B	7A07		MOV	R2,#07H
9	100D	F590	NEXT1：	MOV	P1,A
10	100F	03		RR	A
11	1010	1116		ACALL	DELAY
12	1012	DAF9		DJNZ	R2,NEXT1
13	1014	80EA		SJMP	CONT
14	1016	7B14	DELAY：	MOV	R3,#14H
15	1018	758910		MOV	TMOD,#10H
16	101B	758D9E		MOV	TH1,#9EH
17	101E	758B58		MOV	TL1,#58H
18	1021	D28E		SETB	TR1
19	1023	108F02	LP1：	JBC	TF1,LP2
20	1026	80FB		SJMP	LP1
21	1028	758D9E	LP2：	MOV	TH1,#9EH
22	102B	758B58		MOV	TL1,#58H
23	102E	DBF3		DJNZ	R3,LP1
24	1030	22		RET	
25	1031			END	

⑵ 利用定时器中断方式编制程序

采用定时器 1 中断定时 50 ms，用 R3 做 50 ms 计数单元，在此基础上再用 08H 位做 1 s 计数溢出标志。参考程序如下：

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	2100		AJMP	0100H
3	001B			ORG	001BH
4	001B	212F		AJMP	CONT
5	0100			ORG	0100H
6	0100	758910	MAIN：	MOV	TMOD,#10H
7	0103	758D9E		MOV	TH1,#9EH
8	0106	758B58		MOV	TL1,#58H
9	0109	D2AF		SETB	EA
10	010B	D2AB		SETB	ET1
11	010D	D28E		SETB	TR1
12	010F	C208		CLR	08H
13	0111	7B14		MOV	R3,#14H
14	0113	7A07	DISP：	MOV	R2,#07H
15	0115	74FE		MOV	A,#0FEH
16	0117	F590	NEXT：	MOV	P1,A
17	0119	3008FD		JNB	08H,$
18	011C	C208		CLR	08H
19	011E	23		RL	A
20	011F	DAF6		DJNZ	R2,NEXT
21	0121	7A07		MOV	R2,#07H
22	0123	F590	NEXT1：	MOV	P1,A
23	0125	3008FD		JNB	08H,$
24	0128	C208		CLR	08H
25	012A	03		RR	A
26	012B	DAF6		DJNZ	R2,NEXT1
27	012D	80E4		SJMP	DISP
28	012F	758D9E	CONT：	MOV	TH1,#9EH
29	0132	758B58		MOV	TL1,#58H
30	0135	DB04		DJNZ	R3,EXIT
31	0137	7B14		MOV	R3,#14H
32	0139	D208		SETB	08H
33	013B	32	EXIT：	RETI	
34	013C			END	

⑶ 电路连线图

电路连线图如图 3－2 所示。

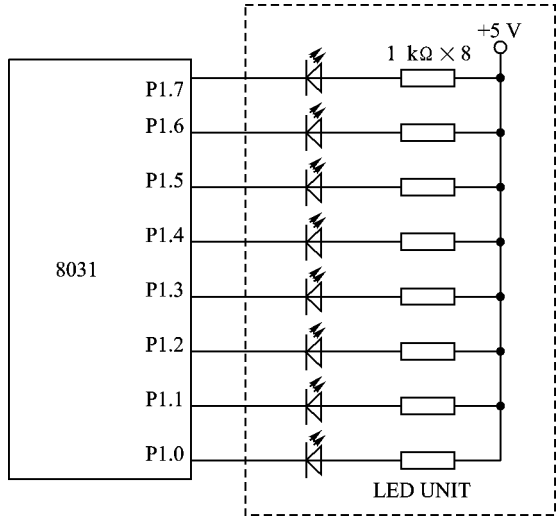


图 3－2　电路连线图

六、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；

② 按照接线图连接好电路；

③ 用连续运行方式运行程序，观察信号灯显示效果。

七、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② 若利用定时器 0 的方式 1 来完成上述程序，试修改程序并调试运行结果。

实验 14　单片机串行接口通信接口实验

一、实验目的

① 利用串行接口发送端发送数据和接收端接收数据，掌握串行异步通信的实现方法；

② 通过串行接口程序调试，学会串行接口方式 1 的使用；

③ 进一步熟悉串行接口中断的基本概念。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照接线图连接好电路；
③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

利用串行接口方式 1 实现异步通信，将片内 RAM 30H 开始的 16 个地址单元的内容依次传送到从片内 RAM 40H 开始的单元。已知系统 f_{osc} = 6 MHz，要求比特率为 110 bit/s。

五、实验参考程序及电路连线图

(1) 实验参考程序

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	021000		LJMP	MAIN
3	0023			ORG	0023H
4	0023	02102E		LJMP	L4
5	1000			ORG	1000H
6	1000	758920	MAIN：	MOV	TMOD,#20H
7	1003	758D72		MOV	TH1,#72H
8	1006	758B72		MOV	TL1,#72H
9	1009	758700		MOV	87H,#00H
10	100C	D28E		SETB	TR1
11	100E	D2AF		SETB	EA
12	1010	759850		MOV	SCON,#50H
13	1013	D2AC		SETB	ES
14	1015	7930		MOV	R1,#30H
15	1017	8799		MOV	SBUF,@R1
16	1019	09		INC	R1
17	101A	7840		MOV	R0,#40H
18	101C	80FE	L1：	SJMP	L1

序号	存储地址	机器码		源程序	
19	101E	E599	L2：	MOV	A,SBUF
20	1020	C298		CLR	RI
21	1022	F6		MOV	@R0,A
22	1023	08		INC	R0
23	1024	B85003		CJNE	R0,#50H,L3
24	1027	C2AC		CLR	ES
25	1029	32		RETI	
26	102A	8799	L3：	MOV	SBUF,@R1
27	102C	09		INC	R1
28	102D	32		RETI	
29	102E	3099ED	L4：	JNB	TI,L2
30	1031	C299		CLR	TI
31	1033	32		RETI	
32	1034			END	

(2) 电路连线图

电路连线图如图 3－3 **所示。**

六、实验步骤

① **输入程序检查无误,经汇编、连接后装入系统；**

② **用** R07 ╳╳↙**操作将待传送数据放入片内** RAM 30H ~ 3FH **中；**

③ **用** G＝0000H ↙**连续运行程序,稍后按“**Reset**”键终止程序运行；**

④ **用** R0740 ↙**检查接收到的数据是否正确。**

七、思考问题(写出实验报告)

① **写出实验过程中所遇到的问题与解决的办法。**

② **若将本实验中串行通信波特率增倍,程序怎样修改？**

③ **根据程序计算串行接口波特率为多少？**

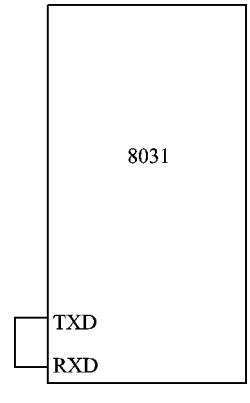


图 3－3　**电路连线图**

第 4 章　MCS－51 单片机接口扩展应用实验

实验 15　数据存储器扩展应用实验

一、实验目的

① 掌握单片机系统存储器扩展方法；
② 进一步掌握断点和连续运行调试程序的方法。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照接线图连接好电路；
③ 用断点和连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

编写程序，把片内 RAM 40H～4FH 单元中共 16 个数据传送到外部数据存储器 7000H～700FH 单元中，然后翻读到片内 RAM 的 50H～5FH 单元中。

五、参考程序及电路连线图

⑴ 参考程序

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	021000		LJMP	MAIN
3	1000			ORG	1000H
4	1000	7840	MAIN：	MOV	R0,#40H
5	1002	7A10		MOV	R2,#10H
6	1004	907000		MOV	DPTR,#7000H

序号	存储地址	机器码		源程序	
7	1007	E6	L1：	MOV	A,@R0
8	1008	F0		MOVX	@DPTR,A
9	1009	08		INC	R0
10	100A	A3		INC	DPTR
11	100B	DAFA		DJNZ	R2,L1
12	100D	7850		MOV	R0,#50H
13	100F	907000		MOV	DPTR,#7000H
14	1012	7A10		MOV	R2,#10H
15	1014	E0	L2：	MOVX	A,@DPTR
16	1015	F6		MOV	@R0,A
17	1016	08		INC	R0
18	1017	A3		INC	DPTR
19	1018	DAFA		DJNZ	R2,L2
20	101A	80FE	L3：	SJMP	L3
21	101C			END	

(2) 电路连线图

电路连线图如图 4－1 所示。

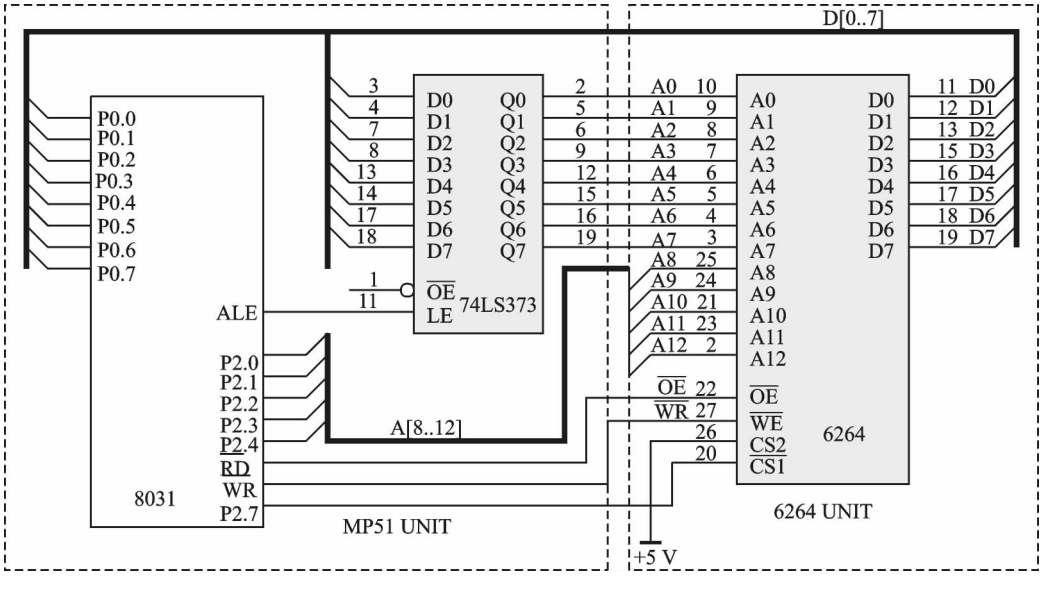


图 4－1　电路连线图

六、实验步骤

① 输入程序检查无误，经汇编、连接后装入系统；
② 按照连线图连接好电路；
③ 在 101A 处设断点；
④ 用 R07 ╳╳↙在 40H～4FH 单元中送入 16 个数据；
⑤ GB＝0000H ↙运行程序；
⑥ 用 R07 ╳╳↙检查 50H～5FH 单元的内容是否与 40H～4FH 单元一致。

七、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。
② 电路中 P2.7 作为 6264 的片选信号；若用 P2.6 作为片选信号，则 6264 的地址范围为多少？程序怎样修改？
③ 通过本次实验，了解怎样进行存储器的扩展，扩展时需注意什么？
④ 本实验电路中采用线选法产生片选信号；若采用译码器法产生片选信号，电路怎样修改？

实验 16　8155 键盘及显示接口实验

一、实验目的

① 掌握利用 8155 扩展并行 I/O 接口的方法；
② 掌握键盘及显示接口的设计方法；
③ 掌握键盘扫描程序和动态显示程序的设计方法。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照连线图连接好电路；
③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

设计 2 行 4 列键盘，4 位动态显示，按键盘上的按键则在相应数码管上显示数字。

五、实验参考程序及电路连线图

(1) 实验参考程序

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	021000		LJMP	MAIN
3	1000			ORG	1000H
4	1000	755200	MAIN：	MOV	52H,#00H
5	1003	755300		MOV	53H,#00H
6	1006	755100		MOV	51H,#00H
7	1009	755000		MOV	50H,#00H
8	100C	7D53		MOV	R5,#53H
9	100E	7403	KEYSUB：	MOV	A,#03H
10	1010	907F00		MOV	DPTR,#7F00H
11	1013	F0		MOVX	@DPTR,A
12	1014	1172	BEGIN：	ACALL	DIS
13	1016	116B		ACALL	CLEAR
14	1018	115E		ACALL	CCSCAN
15	101A	7002		JNZ	INK1
16	101C	0114		AJMP	BEGIN
17	101E	1172	INK1：	ACALL	DIS
18	1020	11AA		ACALL	DL1MS
19	1022	11AA		ACALL	DL1MS
20	1024	116B		ACALL	CLEAR
21	1026	115E		ACALL	CCSCAN
22	1028	7002		JNZ	INK2
23	102A	0114		AJMP	BEGIN
24	102C	7AFE	INK2：	MOV	R2,#0FEH
25	102E	7C00		MOV	R4,#00H
26	1030	907F01	COLUM：	MOV	DPTR,#7F01H
27	1033	EA		MOV	A,R2

序号	存储地址	机器码		源程序	
28	1034	F0		MOVX	@DPTR,A
29	1035	A3		INC	DPTR
30	1036	A3		INC	DPTR
31	1037	E0		MOVX	A,@DPTR
32	1038	20E004		JB	ACC.0,LONE
33	103B	7400		MOV	A,#00H
34	103D	0144		AJMP	KCODE
35	103F	20E111	LONE：	JB	ACC.1,NEXT
36	1042	7404		MOV	A,#04H
37	1044	2C	KCODE：	ADD	A,R4
38	1045	11B3		ACALL	PUTBUF
39	1047	C0E0		PUSH	ACC
40	1049	1172	KON：	ACALL	DIS
41	104B	116B		ACALL	CLEAR
42	104D	115E		ACALL	CCSCAN
43	104F	70F8		JNZ	KON
44	1051	D0E0		POP	ACC
45	1053	0C	NEXT：	INC	R4
46	1054	EA		MOV	A,R2
47	1055	30E304		JNB	ACC.3,KERR
48	1058	23		RL	A
49	1059	FA		MOV	R2,A
50	105A	0130		AJMP	COLUM
51	105C	0114	KERR：	AJMP	BEGIN
52	105E	907F01	CCSCAN：	MOV	DPTR,#7F01H
53	1061	7400		MOV	A,#00H
54	1063	F0		MOVX	@DPTR,A
55	1064	A3		INC	DPTR
56	1065	A3		INC	DPTR
57	1066	E0		MOVX	A,@DPTR
58	1067	F4		CPL	A
59	1068	5403		ANL	A,#03H
60	106A	22		RET	

序号	存储地址	机器码		源程序	
61	106B	907F02	CLEAR：	MOV	DPTR,#7F02H
62	106E	7400		MOV	A,#00H
63	1070	F0		MOV	X@DPTR,A
64	1071	22		RET	
65	1072	C0E0	DIS：	PUSH	ACC
66	1074	C000		PUSH	00H
67	1076	C003		PUSH	03H
68	1078	7403		MOV	A,#03H
69	107A	907F00		MOV	DPTR,#7F00H
70	107D	F0		MOVX	@DPTR,A
71	107E	7850		MOV	R0,#50H
72	1080	7BF7		MOV	R3,#0F7H
73	1082	EB		MOV	A,R3
74	1083	907F01	AGAIN：	MOV	DPTR,#7F01H
75	1086	F0		MOVX	@DPTR,A
76	1087	E6		MOV	A,@R0
77	1088	9010A2		MOV	DPTR,#DSEG
78	108B	93		MOVC	A,@A＋DPTR
79	108C	907F02		MOV	DPTR,#7F02H
80	108F	F0		MOVX	@DPTR,A
81	1090	11AA		ACALL	DL1MS
82	1092	08		INC	R0
83	1093	EB		MOV	A,R3
84	1094	30E004		JNB	ACC.0,OUT
85	1097	03		RR	A
86	1098	FB		MOV	R3,A
87	1099	0183		AJMP	AGAIN
88	109B	D003	OUT：	POP	03H
89	109D	D000		POP	00H
90	109F	D0E0		POP	ACC
91	10A1	22		RET	
92	10A2	3F065B	DSEG：	DB	03FH,06H,05BH
93	10A5	4F666D		DB	04FH,066H,06DH

序号	存储地址	机器码		源程序	
94	10A8	7D07		DB	07DH,07H
95	10AA	7F01	DL1MS：	MOV	R7,#01H
96	10AC	7EFF	DL0：	MOV	R6,#0FFH
97	10AE	DEFE	DL1：	DJNZ	R6,DL1
98	10B0	DFFA		DJNZ	R7,DL0
99	10B2	22		RET	
100	10B3	C000	PUTBUF：	PUSH	00H
101	10B5	C0E0		PUSH	ACC
102	10B7	ED		MOV	A,R5
103	10B8	F8		MOV	R0,A
104	10B9	D0E0		POP	ACC
105	10BB	F6		MOV	@R0,A
106	10BC	1D		DEC	R5
107	10BD	BD4F02		CJNE	R5,#04FH,GOBACK
108	10C0	7D53		MOV	R5,#53H
109	10C2	D000	GOBACK：	POP	00H
110	10C4	22		RET	
111	10C5			END	

(2) 电路连线图

电路连线图如图 4－2 所示。

六、实验步骤

① 输入程序检查无误,经汇编、连接后装入系统；
② 按照连线图连接好电路；
③ 用连续运行方式运行程序；
④ 按动键盘,数码管会显示相应的数字。

七、思考问题 (写出实验报告)

① 写出实验过程中所遇到的问题与解决的办法；
② 画出程序流程图；
③ 每位数码管的显示时间为多少？

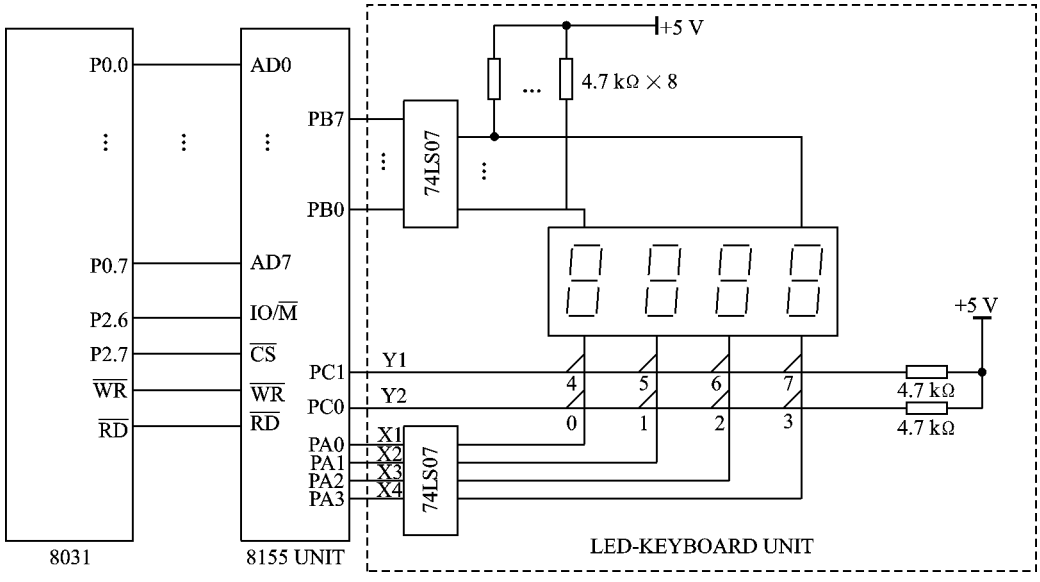


图 4－2　电路连线图

实验 17　ADC 0809 扩展应用设计与调试实验

一、实验目的

① 掌握 ADC 0809 与单片机的连接方法；
② 掌握利用 ADC 0809 实现 A/D 转换程序的设计方法。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照接线图连接好电路；
③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

实验电路如图 4－3 (a) 所示，模拟量输入通道，选择通道 0，其通道地址为 7FF8H，将转换

结果存入片内 RAM30H～3FH 单元。

五、实验参考程序及电路连线图

(1) 实验参考程序

序号	存储地址	机器码		源程序	
1	0000			ORG	0000H
2	0000	021000		LJMP	MAIN
3	1000			ORG	1000H
4	1000	7830	MAIN：	MOV	R0,#30H
5	1002	75A07F		MOV	P2,#7FH
6	1005	7978		MOV	R1,#78H
7	1007	F3	L1：	MOVX	@R1,A
8	1008	121019		LCALL	DALLY
9	100B	30B3FD	L2：	JNB	P3.3,L2
10	100E	E3		MOVX	A,@R1
11	100F	F6		MOV	@R0,A
12	1010	08		INC	R0
13	1011	B840F3		CJNE	R0,#40H,L1
14	1014	80FE	L3：	SJMP	L3
15	1016	00		NOP	
16	1017	00		NOP	
17	1018	00		NOP	
18	1019	7A10	DALLY：	MOV	R2,#10H
19	101B	DAFE	L4：	DJNZ	R2,L4
20	101D	22		RET	
21	101E			END	

(2) 电路连线图

电路连线图如图 4－3 (b) 所示。

六、实验步骤

① 按图 4－3 (b) 连接好线路；

② 输入程序检查无误，经汇编、连接后装入系统；

③ 在 1014 处设断点；

④ 调整 RP，用万用表测试 IN0 线上电压，GB＝0000H ↙运行程序，用 R07 ××↙操作检

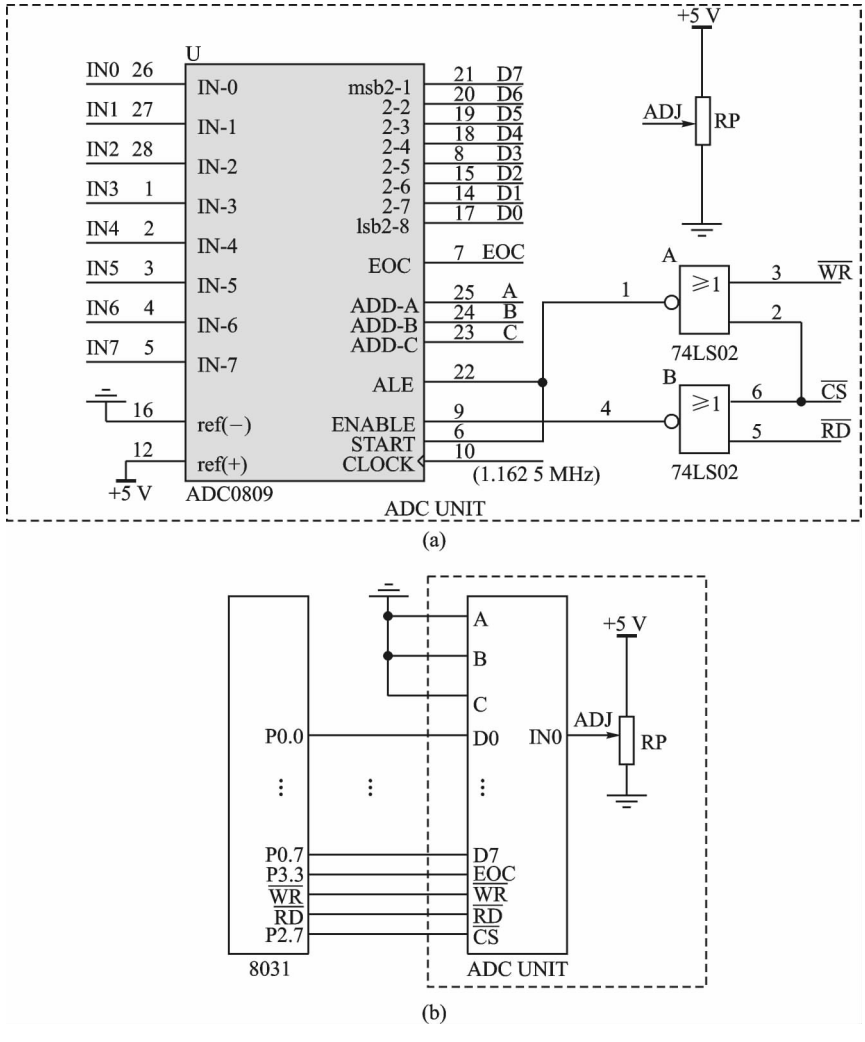


图 4－3　电路连线图

查片内 RAM 30H～3FH 的内容；

⑤ 更改 RP，重新运行程序，检查结果。

七、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② A/D 转换结果与计算的结果是否一致？

③ 如果是对外部 8 路模拟量进行转换，实验电路怎样修改？程序怎样修改？

实验 18　DAC 0832 扩展应用设计与调试实验

一、实验目的

① 掌握 DAC 0832 与单片机的连接方法；
② 掌握利用 DAC 0832 实现 D/A 转换程序的设计方法。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照连线图连接好电路；
③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

实验电路如图 4－4 (a) 所示，当$\overline{WR}$和 P2.7 有效时选中该片，可取地址为 7FFFH，用示波器观察输出波形。

五、实验参考程序及电路连线图

(1) 实验参考程序

① 产生锯齿波程序，周期为 2 ms。

```
        ORG     0000H
        LJMP    MAIN
        ORG     1000H
MAIN：  MOV     DPTR,#7FFFH
        CLR     A
A1：    MOVX    @DPTR,A
        INC     A
        SJMP    A1
        END
```

② 产生三角波程序，周期约为 4 ms。

```
        ORG     0000H
```

```
        LJMP    MAIN
        ORG     1000H
MAIN：  MOV     DPTR,#7FFFH
        CLR     A
L1：    MOVX    @DPTR,A
        INC     A
        CJNE    A,#0FFH,L1
        DEC     A
L2：    MOVX    @DPTR,A
        DEC     A
        CJNE    A,#00H,L2
        SJMP    L1
        END
```

③ **产生阶梯波程序，周期约为** 30 ms。

```
        ORG     0000H
        LJMP    MAIN
        ORG     1000H
MAIN：  MOV     DPTR,#7FFFH
        CLR     A
L2：    MOVX    @DPTR,A
        ADD     A,#10H
        ACALL   L1
        SJMP    L2
L1：    PUSH    0E0H
        PUSH    0F0H
        MOV     A,#02H
L4：    MOV     0F0H,#0FAH
L3：    DJNZ    0F0H,L3
        DEC     A
        JNZ     L4
        POP     0F0H
        POP     0E0H
        RET
        END
```

⑵ 电路连线图

电路连线图如图 4－4 (b) 所示。

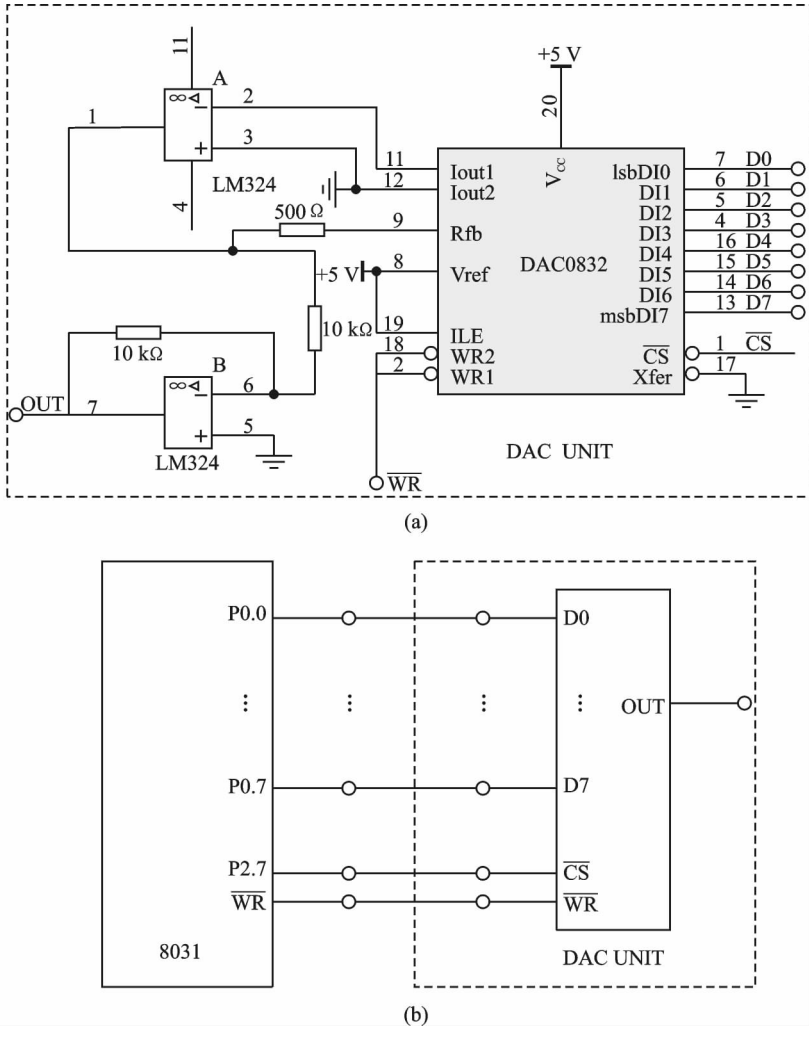


图 4－4 电路连线图

六、实验步骤

① 按图连接好电路；

② 输入程序检查无误，经汇编、连接后装入系统；

③ 分别连续运行上述程序，用示波器观察波形。

七、思考问题(写出实验报告)

① 写出实验过程中所遇到的问题与解决的办法。
② 电路中 DAC 0832 扩展地址是否唯一？
③ 用示波器观察到的电压波形为什么是阶梯状的？应如何消除？

实验 19　步进电动机控制实验

一、实验目的

① 掌握步进电动机的工作原理；
② 掌握步进电动机的单片机控制方法。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；
② 按照连线图连接好电路；
③ 用连续运行方式运行程序,观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

本实验采用的步进电动机为 35BYJ46 型四相八拍电动机,电压为 12 V,其励磁线圈和励磁顺序如图 4－5 所示和表 4－1 所列,实验电路连线图如图 4－6 所示。

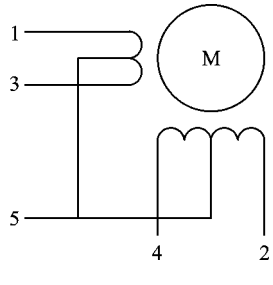


图 4－5　励磁线圈

表 4－1　励磁顺序

序号	1	2	3	4	5	6	7	8
5	+	+	+	+	+	+	+	+
4	-	-	×	×	×	×	×	×
3	×	-	-	-	×	×	×	×
2	×	×	×	-	-	-	×	×
1	×	×	×	×	×	-	-	-

五、实验参考程序及电路连线图

⑴ 实验参考程序

```
        ORG     0000H
        LJMP    MAIN
        ORG     001BH
        LJMP    L3
        ORG     1000H
MAIN：  MOV     R0,#08H
        MOV     DPTR,#L2
        MOV     P1,#00
        MOV     TMOD,#10H
        MOV     TL1,#0B0H
        MOV     TH1,#0AH
        MOV     IE,#88H
        SETB    TR1
L1：    SJMP    L1
L2：    DB      01H,03H,02H,06H,04H,0CH,08H,09H
L3：    MOV     TL1,#0B0H
        MOV     TH1,#0AH
        MOV     A,#00
        MOVC    A,@A+DPTR
        MOV     P1,A
        INC     DPTR
        DJNZ    R0,L4
        MOV     R0,#08
        MOV     DPTR,#L2
L4：    RETI
        END
```

⑵ 电路连线图

电路连线图如图 4－6 所示。

六、实验步骤

① 按图连接好电路；

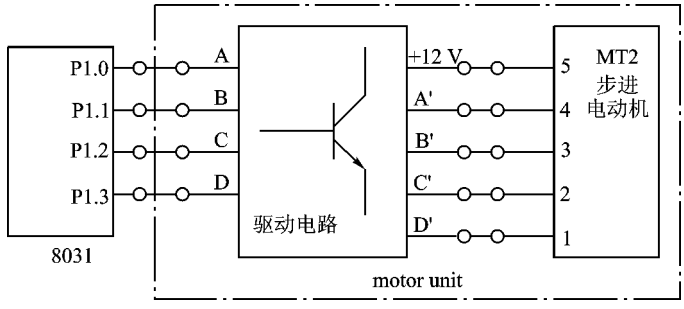


图 4－6　电路连线图

② 输入程序检查无误，经汇编、连接后装入系统；

③ 连续运行程序；

④ 按“Reset”键终止程序运行。

七、思考问题（写出实验报告）

① 写出实验过程中所遇到的问题与解决的办法。

② 如果使步进电动机反转，程序怎样修改？

实验 20　直流电动机调速控制实验

一、实验目的

① 掌握脉宽调制直流调速方法；

② 掌握直流电动机的工作原理。

二、实验要求

① 按照实验内容要求编写应用程序和流程图；

② 按照连线图连接好电路；

③ 用连续运行方式运行程序，观察程序是否运行成功。

三、实验设备

TDN86/51 教学实验系统一台。

四、实验内容

实验电路如图 4－7 所示，利用 P1.0 模拟 PWM 输出经达林顿管输出驱动直流电动机，实

现脉宽调制。

五、实验参考程序及电路连线图

⑴ 实验参考程序

```
        ORG     0000H
        LJMP    MAIN
        ORG     000BH
        LJMP    TT0
        ORG     1000H
MAIN：  SETB    P1.0
        MOV     R0,21H
        MOV     TMOD,#01H
        MOV     TL0,22H
        MOV     TH0,23H
        SETB    TR0
        SETB    ET0
        SETB    EA
L1：    CJNE    R0,#00H,L2
        CPL     P1.0
        MOV     A,20H
        SUBB    A,21H
        MOV     21H,A
        MOV     R0,A
L2：    AJMP    L1
TT0：   MOV     TL0,22H
        MOV     TH0,23H
        DEC     R0
        RETI
        END
```

⑵ 电路连线图

电路连线图如图 4－7 所示。

六、实验步骤

① 按图连接好电路；

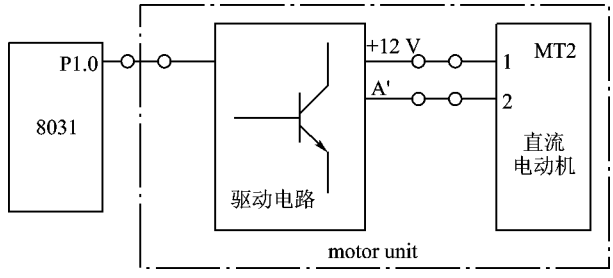


图 4－7　电路连线图

② 输入程序检查无误，经汇编、连接后装入系统；

③ 用 R07XX ↙命令在 20H、21H、22H、23H 单元中分别置入 PWM 参数（如：80、40、00、FF）；

④ 连续运行程序，观察电动机转速 改变参数，观察电动机转速变化；

⑤ 按“Reset”键终止程序运行。

注意：22H、23H 单元中内容分别对应为 T_0 中断，TL0、TH0 时间常数，20H 对应 T 周期，21H 对应 T_1 高电平周期，如图 4－8 所示。

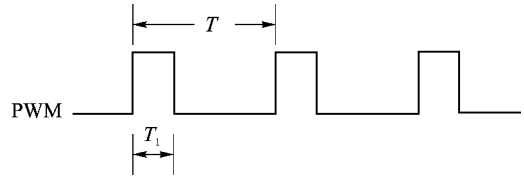


图 4－8　PWM 示意图

$T_{OSC} = 12 \times [10000H - (23H22H)] / (6 \times 10^6)$ s

$T = (20H) \times T_{OSC}$

$T_1 = (21H) \times T_{OSC}$

七、思考问题（写出实验报告）

① 写出实验过程中遇到的问题与解决的办法。

② 试问程序中利用定时器 T_1 完成多长时间定时？是否可以通过改变定时时间常数来改变电动机的转速？

第 5 章　单片机应用系统设计与实习实训课题

实训 1　简易秒表的制作

一、实训目的

① 利用单片机定时器中断和定时器计数方式实现秒定时；
② 通过 LED 显示程序的调整，熟悉 LED 动态显示的控制过程；
③ 熟悉键盘接口、动态显示接口、复位等电路的设计；
④ 熟悉最小应用系统的设计方法；
⑤ 掌握仿真机的应用和软硬件调试方法。

二、设计要求

① 设计 2 位数码显示器，实现秒表计时显示；设计按键 S1、S2、S3，分别实现启动、停止、清零；
② 利用 89S51 为 CPU，设计硬件电路；
③ 利用定时器 0 中断方式实现 1 s 定时，利用定时器 1 方式 2 计数，实现 60 s 计数；
④ 绘制定时中断子程序、动态显示子程序、主程序流程图及编写软件。

三、实训设备

单片机开发系统、微机、实训电路板一套。

四、参考硬件电路图

硬件电路图如图 5－1 所示。

五、参考程序流程图及软件清单

⑴ 参考程序流程图

程序流程如图 5－2 所示。

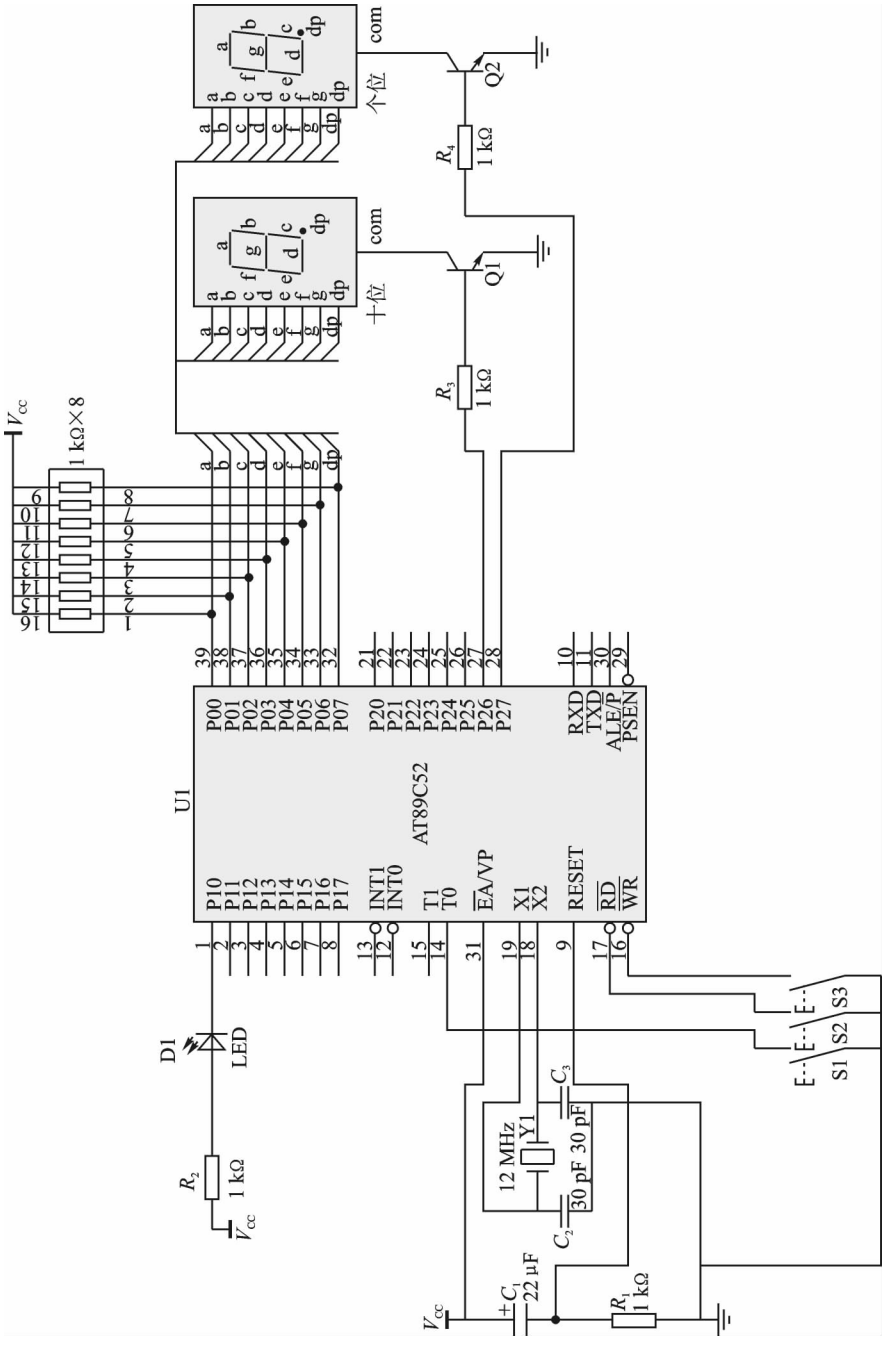
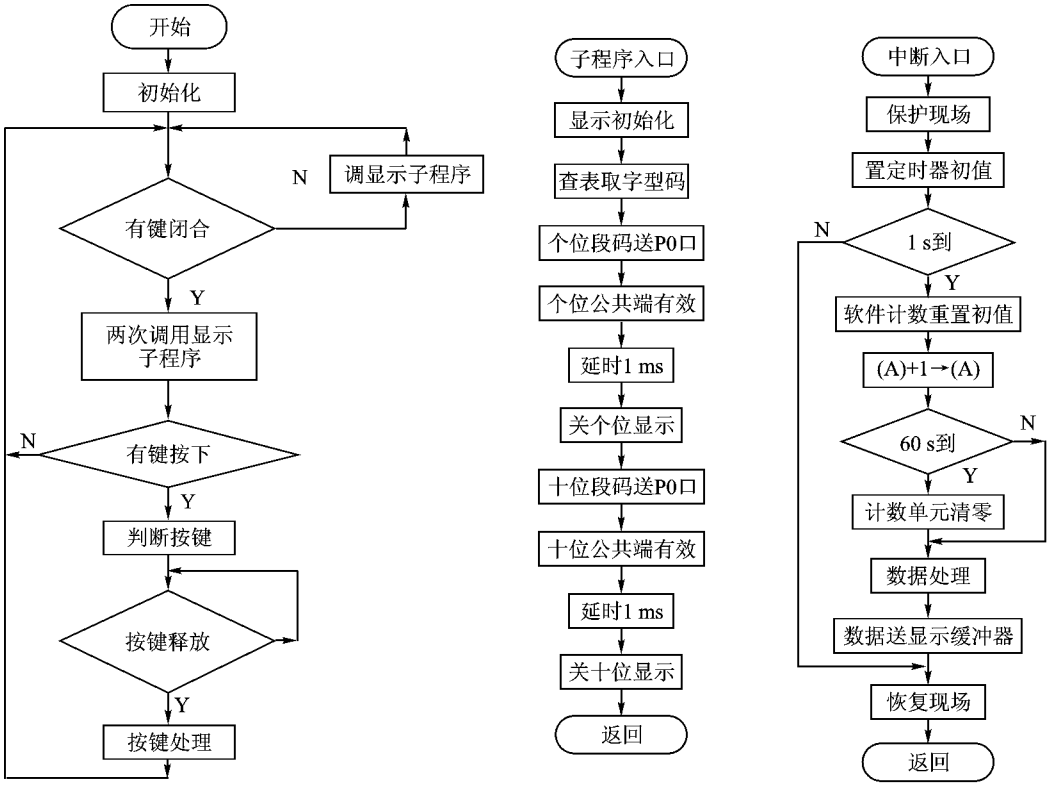

U1
AT89C52
P00 P01 P02 P03 P04 P05 P06 P07
P10 P11 P12 P13 P14 P15 P16 P17
P20 P21 P22 P23 P24 P25 P26 P27
RXD TXD ALE/P PSEN
INT1 INT0 T1 T0 EA/VP X1 X2 RESET RD WR
1 kΩ×8
D1 LED
R_2 1 kΩ
V_{cc}
12 MHz Y1
C_3 30 pF
C_2 30 pF
C_1 22 μF
R_1 1 kΩ
S1 S2 S3
R_3 1 kΩ
R_4 1 kΩ
Q1 Q2
com
十位 个位

图 5－1　硬件电路图



(a) 主程序流程图　　(b) 动态显示子程序　　(c) 定时器中断服务子程序

图 5－2　参考流程图

```
        ORG     0000H
        AJMP    MAIN
        ORG     000BH
        AJMP    CONT
        ORG     0100H
MAIN：  MOV     TMOD,#61H
        MOV     TH0,#3CH
        MOV     TL0,#0B0H
        MOV     TL1,#0C4H
        MOV     TH1,#0C4H
        MOV     20H,#00H
        MOV     21H,#14H
```

```
        MOV     SP,#3FH
        MOV     R2,#02H
        MOV     R0,#30H
STAR：  MOV     @R0,#00H
        INC     R0
        DJNZ    R2,STAR
        CLR     P1.0
        CLR     A
KEY：   ACALL   KS
        JNZ     K1
        ACALL   DISP
        AJMP    KEY
K1：    ACALL   DISP
        ACALL   DISP
        ACALL   KS
        JNZ     K2
        AJMP    KEY
K2：    JB      ACC.7,L1
K4：    ACALL   DISP
        ACALL   KS
        JNZ     K4
        AJMP    KE0
L1：    JB      ACC.6,L2
K5：    ACALL   DISP
        ACALL   KS
        JNZ     K5
        AJMP    KE1
L2：    JB      ACC.6,KEY
K6：    ACALL   DISP
        ACALL   KS
        JNZ     K6
        AJMP    KE2
KE0：   SETB    TR0
        SETB    TR1
```

```
        SETB    ET0
        SETB    EA
        AJMP    KEY
KE1：   CLR     EA
        CLR     ET0
        CLR     TR1
        CLR     TR0
        SETB    P1.0
        AJMP    KEY
KE2：   CLR     EA
        AJMP    MAIN
KS：    MOV     P3,#0FFH
        MOV     A,P3
        CPL     A
        ANL     A,#0D0H
        RET
DISP：  MOV     R1,#00H
        MOV     R0,#30H
        MOV     A,@R0
DISP1： MOV     DPTR,#TAB
        MOVC    A,@A+DPTR
        MOV     P0,A
        SETB    P2.7
        DJNZ    R1,$
        DJNZ    R1,$
        CLR     P2.7
        INC     R0
        MOV     A,@R0
        MOVC    A,@A+DPTR
        MOV     P0,A
        SETB    P2.6
        DJNZ    R1,$
        DJNZ    R1,$
        CLR     P2.6
```

```
        RET
TAB：   DB      3FH,06H,5BH,4FH,66H
        DB      6DH,7DH,07H,7FH,6FH

CONT：  PUSH    ACC
        MOV     TH0,#3CH
        MOV     TL0,#0B0H
        MOV     A,20H
        AJMP    CONT1
REN2：  AJMP    REN1
CONT1：DJNZ     21H,REN2
        CPL     P1.0
        MOV     21H,#14H
        CLR     P3.5
        NOP
        NOP
        SETB    P3.5
        INC     A
        DA      A
        JBC     TF1,CONT2
CONT3：MOV      20H,A
        ANL     A,#0FH
        MOV     30H,A
        MOV     A,20H
        SWAP    A
        ANL     A,#0FH
        MOV     31H,A
        AJMP    REN1
CONT2：MOV      A,#00H
        AJMP    CONT3
REN1：  POP     ACC
        RETI
        END
```

实训 2　智能数字钟的设计与制作

一、实训目的

① 锻炼和提高独立设计、制作和调试应用系统的能力；
② 深入领会单片机应用系统的软、硬件调试方法；
③ 掌握系统研制开发的过程。

二、设计要求

设计并制作具有如下功能的数字钟：
① 自动计时，由 6 位 LED 显示器显示时、分、秒；
② 具备校准功能，可以直接由 0～9 数字键设置当前时间；
③ 具备定时启闹功能；
④ 一天时差不超过 1 s。

三、实训设备

TDN86/51 教学实验系统一台。

四、硬件设计

⑴ 电路原理图

数字钟电路的核心是 89C51 单片机，系统配备 6 位 LED 显示器和 4×3 键盘，采用 8155 作为键盘/显示接口电路。利用 8155 的 A 口作为 6 位 LED 显示器的位选口，其中，PA0～PA5 分别对应位 LED0～LED5；B 口则作为段选口；C 口的低 3 位为键盘输入口，对应 0～2 行；A 口同时用作键盘的列扫描口。由于采用共阴极数码管，因此 A 口输出低电平选中相应的位，而 B 口输出高电平点亮相应的段。P1.0 接蜂鸣器，低电平驱动蜂鸣器鸣叫启闹。电路如图 5－3 所示。8155 的地址分配如下：

控制寄存器：8000H，定义为 PORT
A 口：8001H，定义为 PORTA
B 口：8002H，定义为 PORTB
C 口：8003H，定义为 PORTC

⑵ 系统工作流程

本数字钟具备功能如下：
① 时钟显示：6 位 LED 显示器从左到右依次显示时、分、秒，采用 24 h 计时；

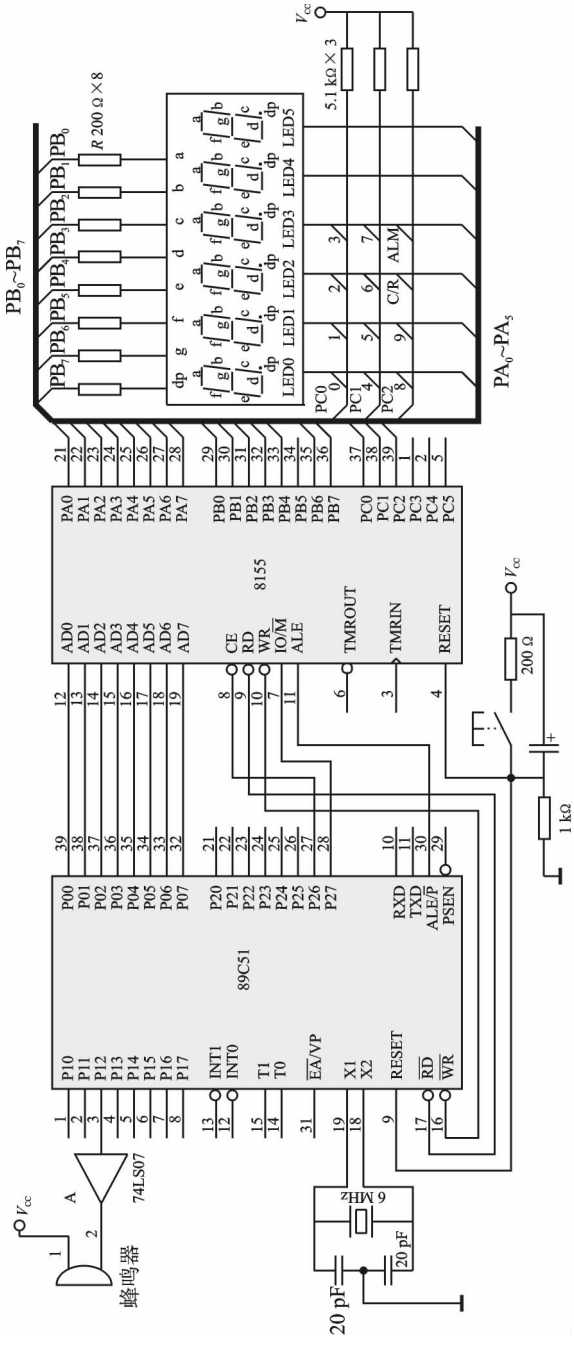


图 5－3　智能数字钟硬件原理图

② 键盘功能：采用 4×3 键盘，包括以下键：

● “0～9” 数字键，键号为 00H～09H；

● “C/R”键 时间设定/启动计时键，键号为 0AH；

● “ALM”键 闹钟设置/启闹/停闹键，键号为 0BH。

其工作流程如下：

① 时间显示：上电后，系统自动进入时钟显示，从 00：00：00 开始计时，此时可以设定当前时间。

② 时间调整：按“C/R”键，系统停止计时，进入时间设定状态，系统保持原有显示，等待键入当前时间。按“0～9”数字键可以顺序设置时、分、秒，并在相应 LED 显示器上显示设置值，直至 6 位设置完毕。系统将自动由设定后的时间开始计时显示。

③ 闹钟设置/启闹/停闹：按“ALM”键，系统继续计时，显示 00：00：00，进入闹钟设置状态，等待键入启闹时间。按“0～9”数字键可以顺序进行相应的时间设置，并在相应 LED 显示器上显示设置值，直至 6 位设置完毕。启动定时启闹功能，并恢复时间显示。定时时间到，蜂鸣器鸣叫，直至重新按“ALM”键停闹，并取消闹钟设置。

五、软件设计

⑴ 系统资源分配

为方便阅读程序，先对系统的资源分配加以说明。

① 定时器：定时器 0 用作时钟定时，按方式 1 工作，每隔 100 ms 溢出中断一次；

② 片内 RAM 及标志位的分配与定义如表 5－1 所列。

表 5－1 片内 RAM 及标志位的分配与定义

地 址	功 能	名 称	初始化值
30H～35H	显示缓冲区，小时、分、秒（高位在前）	DISP0～DISP5	00H
3CH～3FH	计时缓冲区，时、分、秒、100 ms	HOUR，MIN，SEC，MSEC	00H
40H～42H	闹钟值寄存区，时、分、秒	AHOUR，AMIN，ASEC	FFH
50H～7FH	堆栈区		
PSW.5	计时显示允许位（1：禁止，0：允许）	F0	0
PSW.1	闹钟标志位（1：正在闹响，0：未闹响）	F1	0

⑵ 软件流程

根据上述工作流程，软件设计可分为以下几个功能模块。

① 主程序：初始化与键盘监控。

② 计时：为定时器 0 中断服务子程序，完成刷新计时缓冲区的功能。

③ 时间设置与闹钟设置：由键盘输入设置当前时间与定时启闹时间。

④ 显示：完成 6 位动态显示。

⑤ 键盘扫描：判断是否有键按下，并求取键号。

⑥ 定时比较：判断启闹时间到否？如时间到，则启动蜂鸣器鸣叫。

⑦ 其他辅助功能子程序，如键盘设置、拆字、合字、时间合法性检测等。

下面分模块进行软件设计：

① 主程序模块 MAIN：主程序流程图如图 5－4 所示；

② 计时程序模块 CLOCK：计时程序流程图如图 5－5 所示。

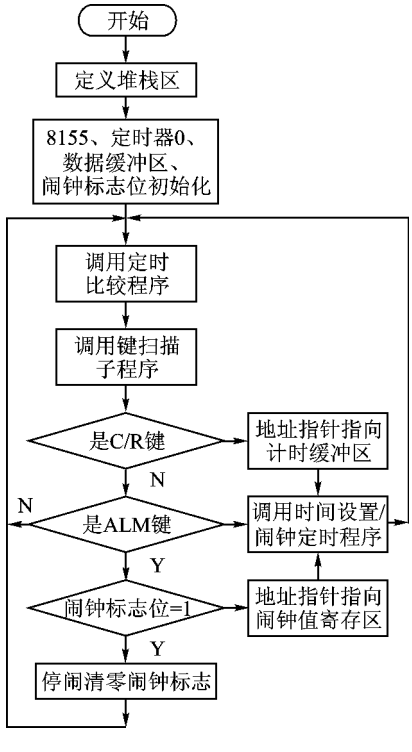


图 5－4　主程序流程图

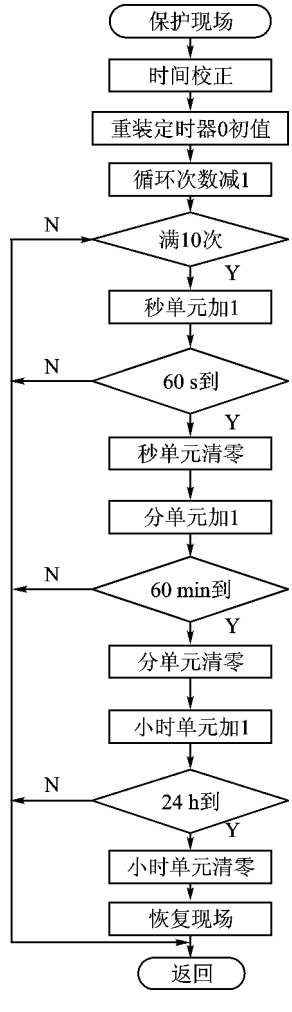


图 5－5　计时程序流程图

如前所述，系统定时采用定时器与软件循环相结合的方法。定时器 0 每隔 100 ms 溢出中断一次，则循环中断 10 次延时时间为 1 s，上述过程重复 60 次为 1 min，分计时 60 次为 1 h，小时计时 24 次则时间重新回到 00：00：00。

设系统使用 6 MHz 的晶振，定时器 0 工作在方式 1，则 100 ms 定时对应的定时器初值可由下式计算得到，即

$$\text{定时时间} = (2^{16} - \text{定时器 0 初值}) \times (12/f_{osc})$$

因此，定时器 0 初值 = 3CB0H，即 TH0 = 3CH，TL0 = 0B0H。

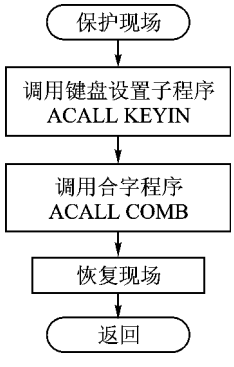


图 5－6 时间设置/闹钟定时流程图

③ 时间设置程序和闹钟定时程序模块 MODIFY：时间设置/闹钟定时流程图如图 5－6 所示。

将键盘输入的 6 位时间值合并为 3 位压缩 BCD 码（时、分、秒）送入计时缓冲区和闹钟值寄存区，作为当前计时起始时间或闹钟定时时间。该模块的入口为计时缓冲区或闹钟值寄存区的首地址，置入 R1 中。程序调用一个键盘设置子程序 KEYIN（如图 5－7 所示）将输入的 6 位时间值送入键盘设置缓冲区，然后用合字子程序 COMB 将键盘设置缓冲区中的 6 位 BCD 码合并为 3 位压缩 BCD 码，送入计时缓冲区或闹钟值寄存区。该程序同时作为时间值合法性检测程序。若键盘输入的小时值大于 23，分和秒值大于 59，则不合法，将取消本次设置，清零重新开始计时。

④ 键盘扫描程序模块 KEYSCAN：键盘扫描流程图如图 5－8 所示。

⑤ 显示程序模块 DISPLAY：显示流程图如图 5－9 所示。

将显示缓冲区中的 6 位 BCD 码用动态扫描方式显示。为此，必须首先将 3 字节计时缓冲区中的时、分、秒压缩 BCD 码拆分为 6 字节（百位、十位分别占有 1 字节）BCD 码，这一功能由拆字子程序 SEPA 来实现。

需要注意的是，当按下时间或闹钟设置键后，在 6 位设置完成之前，应显示键入的数据，而不显示当前时间。为此，通常设置了一个计时显示允许标志位 F0，在闹钟时间/设置期间 F0 = 1，不调用 SEPA，即调用 SEPA 刷新显示缓冲区的前提条件是 F0 = 0。

⑥ 定时比较程序模块 ALARM：定时比较流程图如图 5－10 所示。

将当前时间（计时缓冲区的值）与预设的启闹时间（闹钟设置寄存区的值）比较，二者完全相同时，启动蜂鸣器鸣叫，并置位闹钟标志位。返回后，待重新按下 ALM 键停闹，并清零闹钟标志。

⑦ 拆字程序 SEPA 与合字程序 COMB：如前所述，拆字程序的功能是将 3B 计时缓冲区中的时、分、秒压缩 BCD 码拆分为 6B（百位、十位、个位分别占有 2B）BCD 码并刷新显示缓冲区；合字程序的功能是将键盘设置缓冲区中的 6 位 BCD 码合并为 3 位压缩 BCD 码，送入计时缓冲区或闹钟值寄存区，同时检测时间值的合法性。

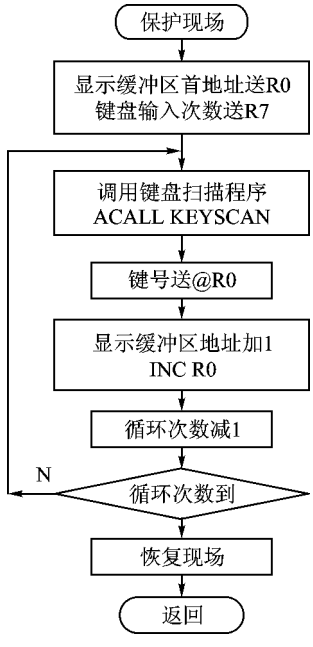


图 5－7　键盘设置子程序流程图

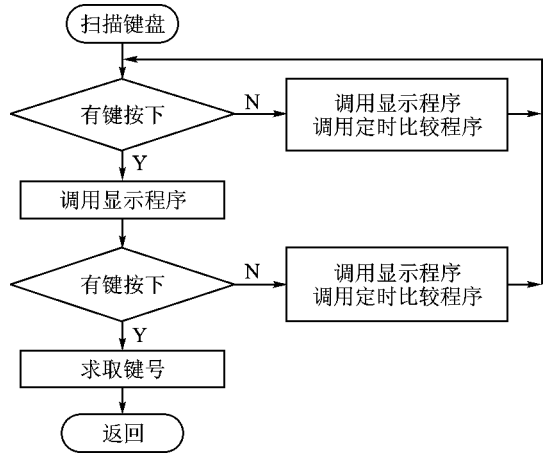


图 5－8　键盘扫描流程图

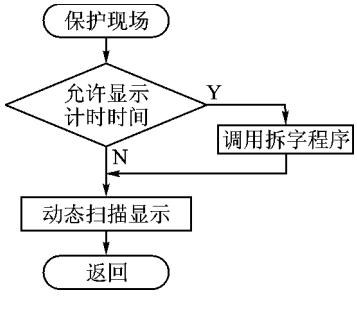


图 5－9　显示流程图

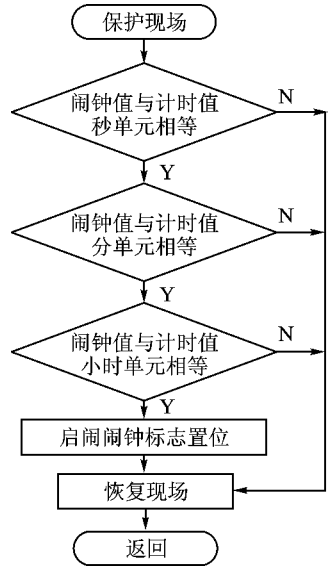


图 5－10　定时比较流程图

(3) 参考程序

下面给出各模块的源程序。

```
;＊＊＊＊＊＊＊＊主程序 MAIN＊＊＊＊＊＊＊＊＊＊
ORG     0000H
AJMP    MAIN
ORG     000BH
AJMP    CLOCK
ORG     0030H
PORT    EQU     8000H
PORTA   EQU     8001H
PORTB   EQU     8002H
PORTC   EQU     8003H
DISP0   EQU     30H
DISP1   EQU     31H
DISP2   EQU     32H
DISP3   EQU     33H
DISP4   EQU     34H
DISP5   EQU     35H
HOUR    EQU     3CH
MIN     EQU     3DH
SEC     EQU     3EH
MSEC    EQU     3FH
AHOUR   EQU     40H
AMIN    EQU     41H
ASEC    EQU     42H
 F1     BIT     PSW.1
MAIN：  MOV     SP,#50H          ;设置堆栈区
        MOVX    DPTR,#PORT
        MOV     A,#03H
        MOVX    @DPTR,A          ;8155 初始化
        CLR     F1               ;清零闹钟标志位
        CLR     F0               ;允许计时显示
        MOV     AHOUR,#0FFH
        MOV     AMIN,#0FFH
```

```
        MOV     ASEC,#0FFH
        MOV     R7,#10H
        MOV     R0,#DISP0
        CLR     A
LOOP:   MOV     @R0,A
        INC     R0
        DJNZ    R7,LOOP          设置初值
        MOV     TMOD,#01H
        MOV     TL0,#0B0H
        MOV     TH0,#3CH         定时器0初始化,定时时间100 ms
        SETB    TR0              启动定时器
        SETB    EA
        SETB    ET0              开中断
BEGIN:  ACALL   ALARM            调用定时比较
        ACALL   KEYSCAN          调用键盘扫描
        CJNE    A,#0AH,NEXT1     按"C/R"键否
        CLR     TR0              是则暂时停止计时
        MOV     R1,#HOUR         地址指针指向计时缓冲区首地址
        AJMP    MOD
NEXT1:  CJNE    A,#0BH,BEGIN     按"ALARM"键否
        JB      F1,NEXT2         闹钟正在闹响否
        MOV     R1,#AHOUR        地址指针指向闹钟值寄存区首地址
MOD:    SETB    F0               置位时间设置/闹钟定时标志,禁止显示
                                 计时时间
        ACALL   MODIFY           调用时间设置/闹钟定时程序
        SETB    TR0              重新开始计时
        CLR     F0               清零时间设置/闹钟定时标志,恢复显示
                                 计时时间
        AJMP    BEGIN
NEXT2:  SETB    P1.0             闹钟正在闹响,停闹
        CLR     F1               清零闹钟标志
        AJMP    BEGIN
;************ 时间设置/闹钟定时模块 MODIFY ***********
MODIFY: ACALL   KEYIN            调用键盘设置子程序
```

```
        ACALL   COMB            ;调用合字子程序
        RET
;************ 键盘设置子程序 KEYIN ***********
KEYIN:  PUSH    PSW
        PUSH    ACC
        SETB    RS1             ;保护现场
        MOV     R0,#DISP0       ;R0 指向显示缓冲区首地址
        MOV     R7,#06H         ;设置键盘输入次数
L1:     CLR     RS1
        ACALL   KEYSCAN         ;调用键盘扫描程序取按下键的键号
        SETB    RS1
        CJNE    A,#0AH,L2       ;输入数合法性检测 (是否大于 9)
L2:     JNC     L1              ;大于 9,重新输入
        MOV     @R0,A           ;键号送显示缓冲区
        INC     R0
        DJNZ    R7,L1           ;6 位时间输入完否？未完继续,否则返回
        POP     ACC
        POP     PSW
        CLR     RS1             ;恢复现场
        RET
;********** 键盘扫描子程序 KEYSCAN **********
KEYSCAN:ACALL   TEST            ;调判按键是否按下子程序 TEST
        JNZ     REMOV           ;有键按下调延时消抖
        ACALL   DISPLAY
        ACALL   ALARM
        AJMP    KEYSCAN         ;无键按下继续判是否按键
REMOV:  ACALL   DISPLAY         ;调用显示子程序延时消抖
        ACALL   ALARM
        ACALL   TEST            ;再判是否有键按下
        JNZ     LIST            ;有键按下转逐列扫描
        ACALL   DISPLAY
        ACALL   ALARM
        AJMP    KEYSCAN         ;无键按下继续判是否按键
LIST:   MOV     R2,#0FEH        ;首列扫描字送 R2
```

```
        MOV     R3,#00H         ;首列键号送 R3
LINE0：  MOV     DPTR,#PORTA     ;DPTR 指针指向 8155 的 A 口
        MOV     A,R2            ;首列扫描字送 R2
        MOVX    @DPTR,A         ;首列扫描字送 8155 的 A 口
        MOV     DPTR,#PORTC     ;DPTR 指针指向 8155 的 C 口
        MOVX    A,@DPTR         ;读入 C 口的行状态
        JB      ACC.0,LINE1     ;第 0 行键无键按下转第 1 行
        MOV     A,#00H          ;第 0 行有键按下,行首键号送 A
        AJMP    TRYK            ;求键号
LINE1：  JB      ACC.1,LINE2     ;第 1 行键无键按下转第 2 行
        MOV     A,#04H          ;第 1 行有键按下,行首键号送 A
        AJMP    TRYK            ;求键号
LINE2：  JB      ACC.2,NEXT      ;第 2 行键无键按下转第 3 行
        MOV     A,#08H          ;第 2 行有键按下,行首键号送 A
        AJMP    TRYK            ;求键号
NEXT：   INC     R3              ;扫描下一列
        MOV     A,R2            ;列扫描字送 A
        JNB     ACC.3,EXIT      ;4 列扫描完,重新进行下一轮扫描
        RL      A               ;4 列未扫描完,扫描字左移扫描下一列
        MOV     R2,A            ;扫描字送 A
        AJMP    LINE0           ;转向扫描下一列
EXIT：   AJMP    KEYSCAN         ;等待下一次按键
TRYK：   ADD     A,R3            ;按公式计算键码,求得键号
        PUSH    ACC             ;键号入栈保护
LETK：   ACALL   TEST            ;等待按键释放
        JNZ     LETK            ;按键未释放,继续等待
        POP     ACC             ;按键释放,键号出栈
        RET                     ;键盘扫描结束,返回
TEST：   MOV     DPTR,#PORTA     ;DPTR 指针指向 8155 的 A 口
        MOV     A,#00H
        MOVX    @DPTR, A        ;全扫描字 00H 送 8155 的 A 口
        MOV     DPTR,#PORTC     ;DPTR 指针指向 8155 的 C 口
        MOVX    A,@DPTR         ;读入 C 口行状态
        CPL     A               ;A 取反,以高电平表示有键按下
```

```
        ANL     A,#07H          ;屏蔽高 5 位
        RET
;************ 显示子程序 DISPLAY ************
DISPLAY: JB     F0,DISP         ;允许时间显示标志 F0 = 1 转 DISP
        ACALL   SEPA            ;否则调用 SEPA 刷新显示缓冲区
DISP:   PUSH    PSW             ;动态扫描显示子程序
        PUSH    ACC
        SETB    RS0
        MOV     DPTR,#PORTA
        MOV     A,#0FFH
        MOVX    @DPTR,A         ;送显示
        MOV     R0,#DISP0
        MOV     R7,#00H
        MOV     R6,#06H
        MOV     R5,#0FEH
DIS1:   MOV     DPTR,#TAB
        MOV     A,@R0
        MOVC    A,@A + DPTR
        MOV     DPTR,#PORTB
        MOVX    @DPTR,A
        MOV     DPTR,#PORTA
        MOV     A,R5
        MOVX    @DPTR,A
HERE:   DJNZ    R7,HERE
        INC     R0
        MOV     A,R5
        RL      A
        MOV     R5,A
        DJNZ    R6,DIS1
        CLR     RS0
        POP     ACC
        POP     PSW
        RET
TAB:    DB      3Fh,06H,5BH,4FH,66H,6DH,7DH,07H
```

```
        DB      7FH,6FH,77H,7CH,39H,5EH,79H,71H    共阴极字型码表
;**************** 合字子程序 COMB ****************
COMB：  MOV     R0,#DISP1           ;R0 指向显示缓冲区小时低位
        ACALL   COMB1               合字
        CJNE    A,#24H,CHK          小时大于 24 否
CHK：   JNC     EXIT1               大于 24 则取消本次设置,退出
        MOV     @R1,A               否则小时送计时缓冲区/闹钟值寄存区小时
                                    单元
        INC     R1
        MOV     R0,#DISP3           ;R0 指向显示缓冲区分低位
        ACALL   COMB1
        CJNE    A,#60H,CHK1
CHK1：  JNC     EXIT1
        MOV     @R1,A
        INC     R1
        MOV     R0,#DISP5           ;R0 指向显示缓冲区秒低位
        ACALL   COMB1
        CJNE    A,#60H,CHK2
CHK2：  JNC     EXIT1
        MOV     @R1,A
        RET
EXIT1：  AJMP    MAIN               输入不合法退出,重新清零计时
COMB1：  MOV     A,@R0
        ANL     A,#0FH              取出低位
        MOV     43H,A               暂存于 43H 单元
        DEC     R0                  指向高位
        MOV     A,@R0
        ANL     A,#0FH
        SWAP    A                   高位送高 4 位
        ORL     A,43H               高低位合并
        RET
;**************** 拆字子程序 SEPA ****************
SEPA：  PUSH    PSW
        PUSH    ACC
```

```
        SETB    RS0
        MOV     R0,#DISP5       指向显示缓冲区秒低位
        MOV     A,SEC
        ACALL   SEPA1
        MOV     A,MIN
        ACALL   SEPA1
        MOV     A,HOUR
        ACALL   SEPA1
        POP     ACC
        POP     PSW
        RET
SEPA1：  MOV     44H,A          暂存 44H
        ANL     A, #0FH         取出低位
        MOV     @R0,A           送显示缓冲区低位
        DEC     R0              指向显示缓冲区高位
        MOV     A,44H
        ANL     A,#0F0H         取出高位
        SWAP    A               高位送往低 4 位形成高位数据
        MOV     @R0,A           高位数据送显示缓冲区高位
        RET
;＊＊＊＊＊＊＊＊＊＊＊＊＊ 定时比较模块 ALARM ＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
ALARM：  MOV     A,ASEC
        CJNE    A,SEC,BACK      秒单元相同则继续比较,否则返回
        MOV     A,AMIN
        CJNE    A,MIN,BACK      分单元相同则继续比较,否则返回
        MOV     A,AHOUR
        CJNE    A,HOUR,BACK     小时单元相同定时时间到
        CLR     P1.0            启动闹钟鸣叫
        SETB    F1              置位闹钟标志
BACK：   RET
;＊＊＊＊＊＊＊＊＊＊＊＊＊＊ 定时器 0 中断服务子程序 CLOCK ＊＊＊＊＊＊＊＊＊＊＊＊＊＊
CLOCK：  MOV     TL0,#0B7H
        MOV     TH0,#3CH        重装初值,时间校正
        PUSH    PSW
```

```
        PUSH    ACC             ;保护现场
        INC     MSEC
        MOV     A,MSEC
        CJNE    A,#0AH,DONE
        MOV     MSEC,#00H
        MOV     A,SEC
        ADD     A,#01H
        DA      A               ;二—十进制转换
        MOV     SEC,A
        CJNE    A,#60H,DONE
        MOV     SEC,#00H
        MOV     A,MIN
        ADD     A,#01H
        DA      A
        MOV     MIN,A
        CJNE    A,#60H,DONE
        MOV     MIN,#00H
        MOV     A,HOUR
        ADD     A,#01H
        DA      A
        MOV     HOUR,A
        CJNE    A,#24H,DONE
        MOV     HOUR,#00H
DONE：  POP     ACC
        POP     PSW             ;恢复现场
        RETI
```

实训 3　单片机作息时间控制钟设计

一、实训目的

① 锻炼和提高独立设计、制作和调试应用系统的能力；
② 深入领会单片机应用系统的软、硬件调试方法；
③ 掌握系统研制开发的过程。

二、设计要求

① 模拟电子钟自动显示，由 6 位 LED 显示器显示时、分、秒；
② 按照学校作息时间按时打铃、播放音乐和广播操节目；
③ 设置两个按钮，分别用来调时和调分。

三、工作原理

本设计可通过 MCS－51 内部定时器 T0 产生中断来实现计时。T0 可工作在定时器工作方式 1，每 100 ms 产生 1 次中断，利用软件将基准 100 ms (1/10 s) 单元进行累加计数。当定时器产生 10 次中断后就产生了 1 s 信号，这时秒单元加 1；同理，可对分单元和时单元计数，从而产生秒、分、时等时间值，并通过连接在 8155A 口、B 口上的 6 位 LED 显示器进行显示。

把学校的作息时间预先制成表格存入 EPROM 数据区中，利用软件每过 1s 将当前时间与数据区时间相比较，相等说明要进行某一项控制，从而可完成打铃、息铃、播放广播操作等控制。

数据区中每一项时间控制字需要占八个存储单元：

控制码 1	时	分	秒

启动装置时间

控制码 2	时	分	秒

关闭装置时间

8031 单片机的 P1.0 用作电铃的开启和关闭，P1.4 用作广播的开启和关闭，控制码的定义如表 5－2 所列。

表 5－2　控制码定义

控制码	功　能	对应输出码
FEH	启动电铃	P1.0 控制电铃
EFH	启动广播	P1.4 控制广播
FFH	关闭装置	P1.0、P1.4 均输出关闭信号
00H	数据区结束	

作息时间与根据作息时间编制的数据区如表 5－3 所列。

表 5－3　作息时间与数据区

作息时间

时间	作息
6:40	起床
6:50	早操
7:15	早饭
8:00	预备铃
8:10～8:55	第一节课
9:05～9:50	第二节课
9:55～9:59	课间操
10:20～11:05	第三节课
11:15～12:00	第四节课
12:15～14:30	午饭及午休
14:40～15:25	第五节课
15:35～16:20	第六节课
16:30	开始课外活动
17:00	晚饭
17:10～17:40	每日广播
19:30～21:30	晚自习

数据区

地址	数据（时间控制字）
1010H～1017H	FE064000 FF064015
1018H～101FH	EF06500 FF065015
1020H～1027H	FE080000 FF080015
1028H～102FH	FE081000 FF081015
1030H～103FH	FE085500 FF085515
1038H～103FH	FE090500 FF090515
1040H～1047H	FE095000 FF095015
1048H～104FH	EF095500 FF095900
1050H～1057H	FE102000 FF102015
1058H～105FH	FE110500 FF110515
1060H～1067H	FE111500 FF111515
1068H～106FH	FE120000 FF120015
1070H～1077H	FE143000 FF143015
1078H～107FH	FE152500 FF152515
1080H～1087H	FE153500 FF153515
1088H～108FH	FE162000 FF162015
1090H～1097H	EF163000 FF173000
1098H～109FH	FE193000 FF193015
10A0H～10A8H	FE213000 FF213015

设程序所用到的片内 RAM 数据存储单元安排如下：

26H　0.1 s 计数单元　　27H　秒计数单元

28H　分计数单元　　29H　时计数单元

2AH　计时单元加 1 暂存器　　2BH　存放秒计数基制

2CH　存放分计数基制　　2DH　存放时计数基制

2EH　保护数据区地址暂存器　　3AH　操作码存储单元

38H、3BH　数据暂存单元　　4AH～4FH　为显示缓冲区

四、参考硬件电路图

硬件电路如图 5－11 所示。

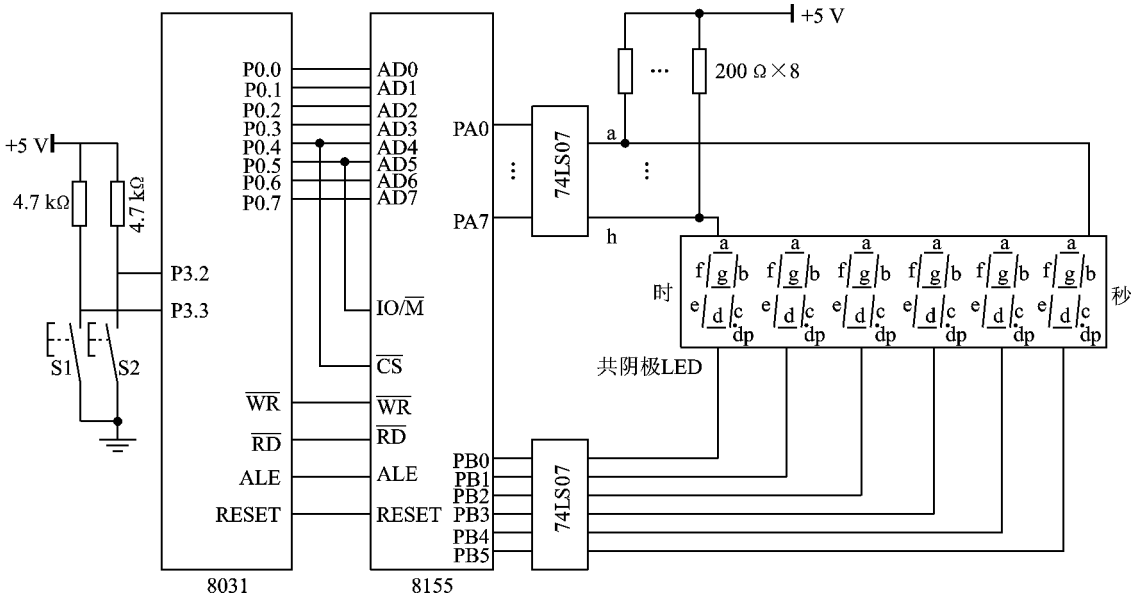


图 5－11　硬件电路图

说明：

① 8155A 口地址为 21H，B 口地址为 22H，显示器显示采用动态扫描法 调试硬件的方法采用向 A 口分别送 1、2、3、5、6、7、8、9 所对应的段码，然后分别选中各个数码管，使它们分别显示 1、2、3、4、5、6、7、8、9。

② 控制装置参考接线方法：

● P1.0 接电铃，P1.4 接广播，电铃和广播可用二极管代替；

● P1.0 二极管亮表示打铃，灭则不打铃；

● P1.4 二极管亮表示广播播放音乐，灭则停止播放音乐。

五、参考程序框图

参考程序框图如图 5－12、图 5－13、图 5－14、图 5－15、图 5－16所示。

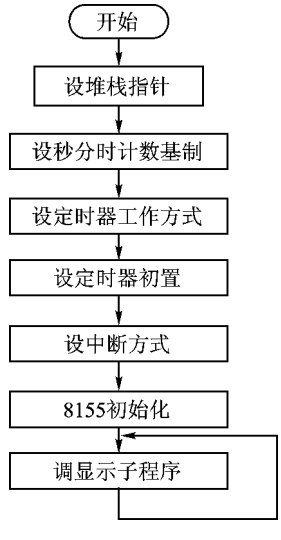


图 5－12　主程序框图

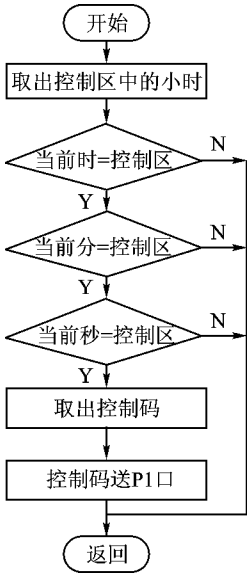


图 5－13　控制程序框图

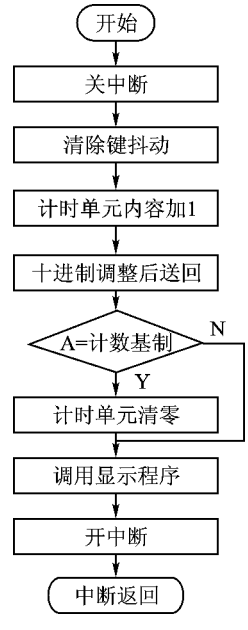


图 5－14　INT0，INT1 中断服务程序框图

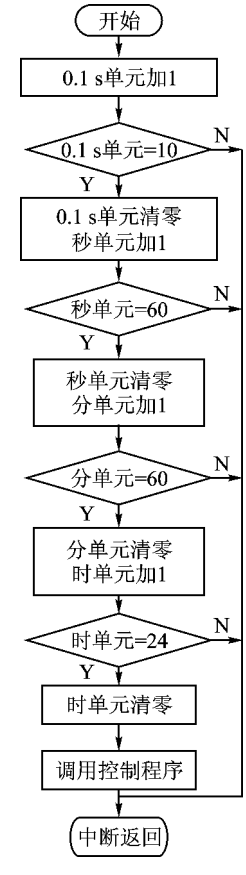


图 5－15　T0 中断服务程序框图

六、参考程序

```
ORG     0000H
LJMP    MAIN        转主程序
ORG     0003H
LJMP    INT0        转 INT0 中断
ORG     000BH
LJMP    CLOCK       转定时器 T0 中断
ORG     0013H
LJMP    INT1        转 INT1 中断
```

(1) INT0 中断服务程序

```
        ORG     0050H
INT0：  CLR     EX0          ;INT0中断关闭
        JNB     P3.2,$       ;消除键抖动
        INC     28H          ;分单元加1
        MOV     A,28H        ;十进制调整
        ADD     A,#00H
        DA      A
        MOV     28H,A
        SUBB    A,#60H       ;不等于计数基制转DSUP2
        JC      DSUP2
        MOV     28H,#00H     ;相等分单元清零
DSUP2：LCALL    DSUP         ;调显示子程序
        SETB    EX0          ;开INT0中断
        RETI                 ;中断返回
```

(2) INT1 中断服务程序

```
INT1：  CLR     EX1          ;开INT1中断
        JNB     P3.3,$       ;消除键抖动
        INC     29H          ;时单元加1
        MOV     A,29H        ;十进制调整
        ADD     A,#00H
        DA      A
        MOV     29H,A
        SUBB    A,#24H       ;不等于计数基制转DSUP3
        JC      DSUP3
        MOV     29H,#00H     ;相等则分单元清零
DSUP3：LCALL    DSUP         ;调显示子程序
        SETB    EX1          ;INT1中断开放
        RETI                 ;中断返回
```

(3) 主程序

```
MAIN：  MOV     A,#03H       ;8155初始化
        MOV     R0,#20H
        MOV     @R0,A
        MOV     SP,#5AH      ;堆栈指针
```

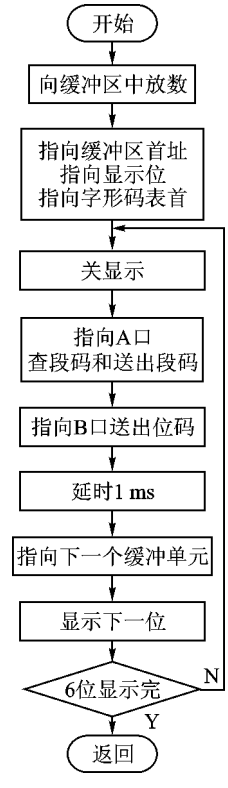


图 5－16　显示程序框图

```
        MOV     2BH,#60H        秒计数基制
        MOV     2CH,#60H        分计数基制
        MOV     2DH,#24H        时计数基制
        MOV     TMOD,#01H       定时器工作方式 1
        MOV     TL0,#0B0H       置 T0 初值
        MOV     TH0,#3CH
        MOV     IE,#87H         允许中断
        SETB    TR0             启动定时器 T0
LOOP:   LCALL   DSUP            调显示子程序
        LJMP    LOOP            循环
```

(4) 显示程序

```
DSUP:   MOV     R0,#4FH         准备向缓冲区放数
        MOV     A,27H
        ACALL   PTDS            放秒值
        MOV     A,28H
        ACALL   PTDS            放分值
        MOV     A,29H
        ACALL   PTDS            放小时值
        MOV     R0,#4AH         指向缓冲区首址
        MOV     R2,#0DFH        左边第一位开始显示
        MOV     DPTR,#SEGPT     指向字形码表首
DSUP1:  MOV     A,#00H          熄灭码
        MOV     R1,#21H         字形口地址
        MOVX    @R1,A           送显示
        MOV     A,@R0           取显示缓冲区中的数
        MOVC    A,@A+DPTR       查表,找字形码
        MOVX    @R1,A           送出字形码
        MOV     A,R2            取字位码
        MOV     R1,#22H         字位口地址
        MOVX    @R1,A           显示一位数
        MOV     R3,#00H
DSUP4:  DJNZ    R3,DSUP4        延时一段时间
        INC     R0              修改显示缓冲区指针
        CLR     C               为移位作准备
```

```
        MOV     A,R2        ;取字位码
        RR      A           ;右移一位,为显示下一位作准备
        MOV     R2,A        ;存位码
        JB      ACC.7,DSUP1 ;不到最后一位,则继续
        RET
PTDS:   MOV     R1,A        ;暂存
        ACALL   PTDS1       ;低4位先放入缓冲区
        MOV     A,R1        ;取出原数
        SWAP    A           ;高4位放入低4位中
PTDS1:  ANL     A,#0FH      ;放进显示缓冲区
        MOV     @R0,A
        DEC     R0          ;缓冲区地址指针减1
        RET
```

(5) T0 中断服务程序

```
CLOCK:  PUSH    PSW         ;保护现场
        PUSH    ACC
        SETB    RS0         ;选择工作寄存器组1
        MOV     TL0,#0B7H   ;重装定时器T0初值
        MOV     TH0,#3CH
        INC     26H         ;0.1 s单元加1
        MOV     A,26H       ;取0.1 s单元内容
        CJNE    A,#0AH,DONE ;不等于10转DONE
        MOV     26H,#00H    ;等于10,则清0.1 s单元
        MOV     R0,#27H     ;指向秒计数单元
        MOV     R1,#2BH     ;指向秒计数单元基制
        MOV     R3,#03H     ;循环三次(秒、分、时)
LOOP0:  MOV     A,@R0       ;取计时单元的值
        ADD     A,#01H
        DA      A           ;十进制调整
        MOV     @R0,A       ;送回计时单元
        MOV     38H,@R1
        CJNE    A,38H,DONE0 ;不等于计时基制则转出
        MOV     @R0,#00H
        INC     R0
```

```
        INC     R1
        DJNZ    R3,LOOP0        ;秒、分、时共三次循环
DONE0： ACALL   LOOP1           ;调用查看数据区子程序
DONE：  POP     ACC             ;恢复现场
        POP     PSW
        RETI                    ;中断返回
```

(6) 控制程序

```
LOOP1： MOV     DPTR,#100CH
        MOV     2EH,DPL
LOOP4： MOV     DPL,2EH
        MOV     R3,#04H
        MOV     R1,#2AH
LOOP2： INC     DPTR
        DJNZ    R3,LOOP2
        MOV     2EH,DPL
        MOV     R3,#03H         ;时、分、秒共三次
        CLR     A
        MOVC    A ,@A + DPTR    ;取控制码
        JZ      LOOP3           ;若 A = 0,则数据区结束
        MOV     3AH ,A          ;保护控制码
LOOP5： INC     DPTR            ;修改数据区时间单元指针
        DEC     R1              ;修改计时单元指针
        CLR     A
        MOVC    A,@A + DPTR     ;读取数据区时间
        MOV     3BH,A           ;暂存
        MOV     A,@R1           ;读取计时单元时间
        CJNE    A,3BH,LOOP3     ;比较时间是否相等
        DJNZ    R3 ,LOOP5       ;3 次循环
        MOV     A ,3AH          ;恢复控制码
        CPL     A               ;控制码变反
        MOV     P1， A          ;由并行口 P1 输出,执行控制
LOOP3： RET
```

段码：

```
SEGPT： DB 3FH,06H,5BH,4FH,66H,6DH,7DH
```

```
        DB 07H,7FH,6FH,77H,7CH,39H,5EH
        DB 79H,71H
```

控制码：

```
        ORG 1010H
CODE：  DB 0FEH,06H,00H,00H,0FFH,06H,00H,15H
        DB 0EFH,06H,30H,00H,0FFH,07H,10H,00H
        DB 0FEH,07H,15H,00H,0FFH,07H,15H,10H
        DB 0FEH,08H,00H,00H,0FFH,08H,00H,10H
        DB 0FEH,08H,50H,00H,0FFH,08H,50H,10H
        DB 0FEH,09H,00H,00H,0FFH,09H,00H,10H
        DB 0FEH,09H,50H,00H,0FEH,09H,50H,10H
        DB 0EFH,09H,55H,00H,0FFH,09H,59H,10H
        DB 0FEH,10H,10H,00H,0FFH,10H,10H,10H
        DB 0FEH,11H,00H,00H,0FFH,11H,00H,10H
        DB 0FEH,11H,10H,00H,0FFH,11H,10H,10H
        DB 0FEH,12H,00H,00H,0FFH,12H,00H,10H
        DB 0FEH,12H,15H,00H,0FFH,12H,15H,10H
        DB 0FEH,13H,30H,00H,0FFH,13H,30H,10H
        DB 0FEH,14H,20H,00H,0FFH,14H,20H,10H
        DB 0FEH,14H,30H,00H,0FFH,14H,30H,10H
        DB 0FEH,15H,20H,00H,0FFH,15H,20H,10H
        DB 0FEH,15H,30H,00H,0FFH,15H,30H,10H
        DB 0FEH,17H,30H,00H,0FFH,17H,30H,10H
        DB 0EFH,18H,00H,00H,0FFH,18H,20H,00H
        DB 0FEH,18H,30H,00H,0FFH,18H,30H,10H
        DB 0FEH,22H,00H,00H,0FFH,22H,00H,10H
        DB 0FEH,22H,30H,00H,0FFH,22H,30H,20H
        DB 00H
```

七、课题实习基本任务及步骤

① 根据图 5－3 扩展显示器接口；

② 分段调试程序采用单步运行和断点运行方式；

③ 调试 T0 中断服务程序，首先在计数单元 26H、27H、28H、29H 中预置数，实现 0.1 s 单元是否向秒单元进位的调试；其次，用同样的方法，调试通过秒单元是否向分单元进位，分单元

是否向时单元进位的程序；最后将 T0 中断服务程序全部调试通过；

④ 在 26H、27H、28H、29H 单元中预置数，调试显示程序；

⑤ 在 26H、27H、28H、29H 单元中预置数，调试控制以及控制硬件接线，如 6：40 起床，则 6：40：00 开始打铃，6：40：15 停止打铃，则 27H，28H，29H 中的内容应设置在 6：40：00 调试控制程序，使接在 P1.0 的发光二极管点亮，27H，28H，29H 设置在 6：40：15，通过调试使发光二极管熄灭。用同样的方法调试程序使接在 P1.4 的发光二极管亮灭符合要求；

⑥ 调试主程序，使报时打铃系统正常工作。

⑦ 调试校对程序及硬件，即 INT0，INT1 中断服务程序，完成题目所要求功能。

实训 4　单片机交通灯控制器设计

一、实训目的

① 锻炼和提高独立设计、制作和调试应用系统的能力；

② 深入领会单片机应用系统的软、硬件调试方法；

③ 掌握系统研制开发的过程。

二、实训要求及任务

模拟十字路口信号灯控制系统的硬软件系统设计。

① 正常情况下 60 s 各信号灯由“红”变“绿”，中间是 2 s 的“黄”灯过渡；

② 设东西方向、南北方向紧急切换按钮各一个，按下时，相应方向紧急切换为“绿”灯，以便利特种车辆通过；

③ 用两个数码管显示点亮的灯还能持续的时间。

三、设计原理

本设计的延时可通过软件延时和定时器延时两种方法来实现。软件延时可先编写一段延时 1 s 的子程序，然后在主程序中反复调用以实现 60 s 和 2 s 的延时，并送出信号去控制相应的交通灯；定时器延时可通过单片机内部定时器 T0 产生中断来实现计时，T0 可工作在定时器工作方式 1，每 100 ms 产生一次中断，然后在中断服务程序实现 60 s 和 2 s 的延时，并送出信号去控制相应的交通灯。

紧急切换可采用中断或查询按钮两种方法实现，数码管的扩展可参照单片机作息时间钟的显示器的扩展方法，或由其他的方法实现显示器的扩展。

四、参考硬件电路图

硬件电路如图 5－17 **所示。**

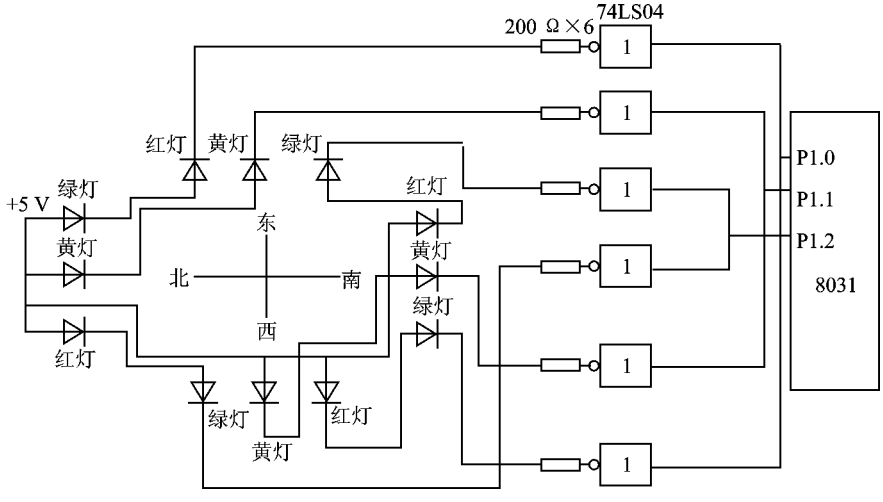


图 5－17　**硬件电路图**

五、参考程序框图

参考程序框图如图 5－18、**图** 5－19 **所示。**

六、参考程序

```
        ORG     0000H
        LJMP    MAIN
        ORG     000BH
        LJMP    CLOCK
        ORG     1000H
MAIN：  MOV     SP ,#5AH
        MOV     2BH ,#60H
        MOV     TMOD,#01H
        MOV     TL0 ,#0B7H
        MOV     TH0 ,#3CH
        MOV     IE ,#82H
        SETB    TR0
```

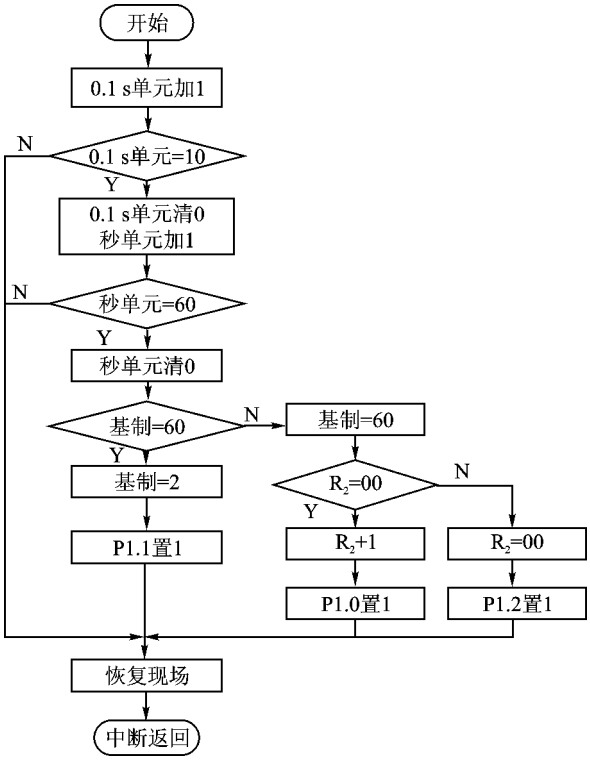


图 5－18　T0 中断服务程序框图

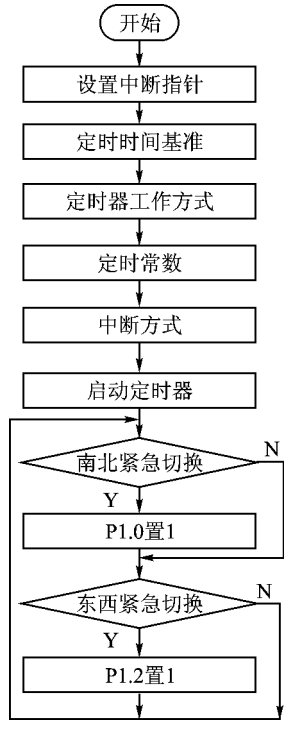


图 5－19　利用查询方式实现紧急切换的参考主程序框图

```
        MOV     R2 ,#00H
QIEH：  JB      P3.3,QIEH1
        MOV     P1,#04H
QIEH1： JB      P3.2, QIEH2
        MOV     P1,#01H
        ORG     1400H
CLOCK：PUSH     PSW
        PUSH    ACC
        SETB    RS0
        MOV     TL0 ,#0B7H
        MOV     TH0 ,#3CH
        INC     26H
        MOV     A ,26H
        CJNE    A,#0AH,DONE
```

```
        MOV     26H ,#00H
        MOV     R0,#27H
        MOV     R1, #2BH
        MOV     A,@R0
        ADD     A,#01H
        DA      A
        MOV     @R0,A
        MOV     38H ,@R1
        CJNE    A ,38H,DONE
        MOV     @R0 ,#00H
        MOV     A ,2BH
        CJNE    A,#60,LOOP0
        MOV     2BH,#02H
        MOV     P1,#02H
        LJMP    DONE
LOOM：  MOV     2BH ,#60
        MOV     A ,R2
        CJNE    A ,#00H,LOOP1
        INC     R2
        MOV     P1 ,#01H
        LJMP    DONE
LOOPl： MOV     R2 ,#00H
        MOV     P1,#04H
DONE：  POP     ACC
        POP     PSW
        RETI
        END
```

此外，显示器的扩展参考单片机作息时间钟，编写显示子程序，完善主程序。

七、课题实习基本任务及步骤

① 调试通过 T0 中断服务程序；
② 与硬件相配合，调试通过原主程序，使二极管的亮灭符合题目的要求；
③ 完成显示器的接线；
④ 调试通过显示程序；
⑤ 调试最后完善的程序，实现设计任务所要求的功能。

实训 5　音乐演奏控制器设计

一、设计要求

① 能在几首歌曲之间任意选曲；

② 设计显示器显示曲目。

二、基础知识

⑴ 声音的产生

声音是由物体振动产生的，振动的频率不同，发出的声音也就不同，有规律的振动发出的声音称为“音乐”。乐谱中的每一个音符都与某一特定频率相对应。例如 C 调中的“1”所对应的振动频率是 524 Hz。

⑵ 利用单片机进行乐曲演奏的原理

音乐演奏控制器是通过控制单片机内部的定时器 T0 来产生不同频率的方波，驱动扬声器（俗称喇叭）发出不同音节的声音。再利用延时来控制发音时间的长短，即可控制音调中的节拍。把乐谱中的音符对应的频率转换为定时常数，把相应的节拍变换为定时常数，然后做成表格存放在存储器中，由程序查表得到定时常数和延时常数，分别用以控制定时器产生方波的频率和该频率方波的持续时间。当延时常数到时，再查下一个音符的定时常数和延时常数。依次进行下去，就可演奏悦耳动听的音乐。将某一首音乐的简谱变换成为常数表，计算机顺序调入时间常数并以中断方式执行，从 P1.7 输出方波驱动扬声器，发出不同音节的声音，节拍的控制可通过调用 200 ms 延时子程序的次数来实现。单片机的晶振频率为 6 MHz，乐谱中的音符、频率及定时常数的关系为

$$T_0 = 2^{16} - \frac{1}{2ft_i}$$

式中，f 为音符对应的频率；t_i 为内部计时一次所用时间。

例如：C 调 1 对应的频率 524 Hz，其半周期 $T/2 = \left(\frac{1}{524} \times \frac{1}{2}\right)$ ms = 0.95 ms，用定时器 T0 方式 1 定时器常数计算公式得到定时常数为十六进制的 FE25。

下面是歌曲《新年好》的一段简谱：

1 = C　$\underline{1\ 1}$ 1 5 |$\underline{3\ 3}$ 3 1 |$\underline{1\ 3}$ 5 5 |$\underline{4\ 3}$ 2 — |

用定时器 T0 方式 1 来产生歌谱中每个音符对应频率的方波，由 P1.7 输出驱动扬声器。节拍的控制可通过调用延时子程序 D200（延时 200 ms）次数来实现，以每拍 800 ms 的节拍时间为例，那么一拍需要循环调用 D200 延时子程序四次。同理，半拍就需要调用 D200 延时子

程序两次。设单片机晶振频率为 6 MHz，乐曲中的音符、频率及定时常数三者之间的关系如表 5－4 所列。

表 5－4　音符、频率及定时常数三者之间关系

C 调音符	1	2	3	4	5	6	7
频率/Hz	263	294	330	349	392	440	494
半周期/ms	1.90	1.7	1.52	1.43	1.28	1.14	1.095
定时值	FC4A	FCAE	FD08	FD35	FD80	FDC6	FE07
C 调音符	1	2	3	4	5	6	7
频率/Hz	524	588	660	698	784	880	988
半周期/ms	0.95	0.85	0.76	0.72	0.64	0.57	0.51
定时值	FE25	FE57	FE84	FE98	FEC0	FEE3	FF01
C 调音符	1	2	3	4	5	6	7
频率/Hz	1046	1175	1318	1397	1568	1760	1976
半周期/ms	0.478	0.426	0.379	0.358	0.319	0.284	0.253
定时值	FF11	FF2B	FF43	FF5D	FF61	FF72	FF82

三、参考硬件图

硬件电路图如图 5－20 所示。

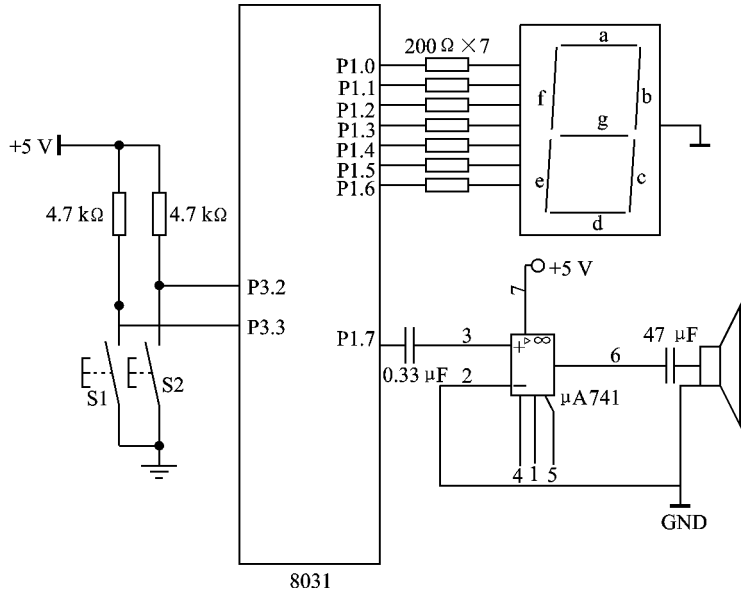


图 5－20　硬件电路图

四、参考程序框图

参考程序图如图 5－21，**图** 5－22，**图** 5－23 **所示。**

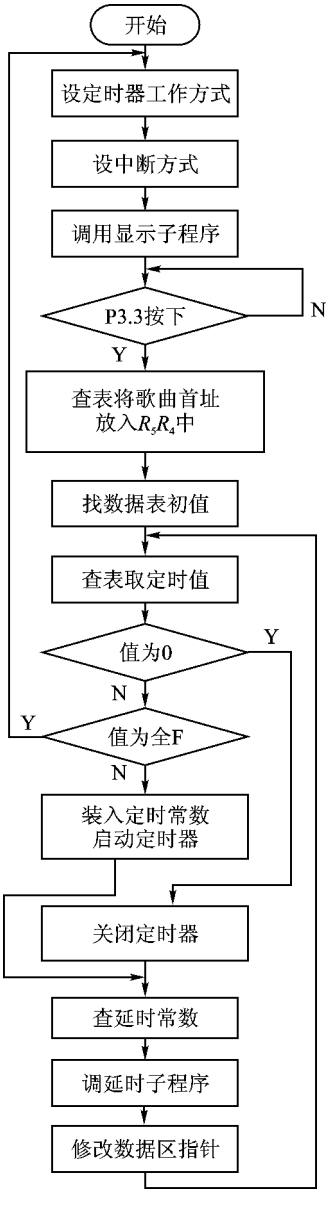


图 5－21　**主程序流程图**

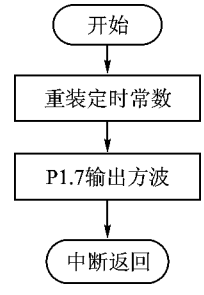


图 5－22　T0 **中断服务程序框图**

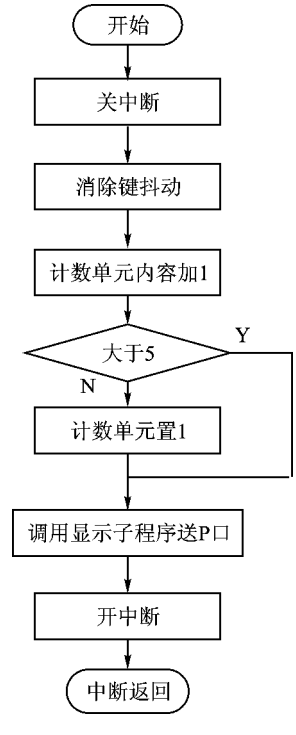


图 5－23　INT0 **中断服务程序**

五、参考程序

```
        ORG     0003H
        LJMP    1060H
        ORG     1060H
        JNB     P3.2,$
        INC     R6
        MOV     A, R6
        CJNE    A ,#05H ,DONE
        MOV     R6 ,#01H
DONE：  LCALL   DSUP
        SETB    EX0
        RETI
        ORG     000BH
        MOV     TH0 ,R1
        MOV     TL0,R0
        CPL     P1.7
        RETI
        ORG     1000H
        MOV     R6 ,#01H
MAIN：  MOV     IE ,#83H
        MOV     TMOD ,#01H
        LCALL   DSUP
        JB      P3.3,$
LOOP1：MOV     DPTR,#TAB
        MOV     A,R6
        DEC     A
        RL      A
        MOV     R7 ,A
        MOVC    A ,@A + DPTR
        MOV     R5 ,A
        INC     DPTR
        MOV     A ,R7
        MOVC    A ,@A + DPTR
```

```
        MOV     R4,A
        MOV     DPH ,R5
        MOV     DPL ,R4
LOOP：  CLR     A
        MOVC    A,@A + DPTR
        MOV     R1 ,A
        INC     DPTR
        CLR     A
        MOVC    A ,@A + DPTR
        MOV     R0 ,A
        ORL     A ,R1
        JZ      NEXT0
        MOV     A ,R0
        ANL     A,R1
        CJNE    A ,#0FFH ,NEXT
        SJMP    MAIN
NEXT：  MOV     TH0,Rl
        MOV     TL0,R0
        SETB    TR0
        SJMP    NEXT1
NEXT0：CLR      TR0
NEXT1：CLR      A
        INC     DPTR
        MOVC    A ,@A + DPTR
        MOV     R2,A
LOOP2：ACALL    D200
        DJNZ    R2,LOOP2
        INC     DPTR
        AJMP    LOOP
D200：  MOV     R3 ,#81H
D200A：MOV      A ,#0FFH
D200B：DEC      A
        JNZ     D200B
        DEC     R3
```

```
        CJNE    R3 ,#00H ,D200A
        RET
        ORG     1080H
DSUP：  MOV     DPTR ,#3000H
        MOV     A ,R6
        MOVC    A ,@A + DPTR
        MOV     P1 ,A
        RET
        ORG     2000H
TAB：   DB      0FEH,25H,02H,0FEH,25H,02H
        DB      0FEH,25H,04H,0FDH,80H,04H
        DB      0FEH,84H,02H,0FEH,84H,02H
        DB      0FEH,84H,04H,0FEH,25H,04H
        DB      0FEH,25H,02H,0FEH,84H,02H
        DB      0FEH,0C0H,04H,0FEH,0C0H,04H
        DB      0FEH,98H,02H,0FEH,84H,02H
        DB      0FEH,57H,08H,00H,00H,04H
        DB      0FFH,0FFH
        END
```

六、课题实习任务及步骤

① 按图接好电路，将已知程序送入单片机，使其演奏一首动听的音乐，并分别调试显示程序和选曲程序。

② 调试编写的主程序，使选曲和显示功能正常实现。

第 6 章　Keil 集成 IDE 软件开发平台介绍

Keil IDE (μVision2/3) 集成开发环境是 Keil Software Inc/Keil Elektronik GmbH 开发的基于 80C51 内核的微处理器软件开发平台，内嵌多种符合当前工业标准的开发工具，可以完成从工程建立和管理，编译、连接、目标代码的生成，软件仿真和硬件仿真等完整的开发流程。由于 Keil 本身是一个纯软件的东西，还不能直接进行硬件仿真，必须挂接仿真器的硬件才可以进行仿真，所以本章首先介绍 Keil 的基本操作，然后结合第 1.4 节“TKS－52B 仿真器的使用”，详细介绍一个应用系统如何完成在 Keil 环境中的软件调试与硬件仿真。

6.1　Keil 软件的基本操作

1. 文件菜单和文件命令

在菜单栏中单击 File 菜单，如图 6－1 所示。File 菜单命令描述如表 6－1 所列。

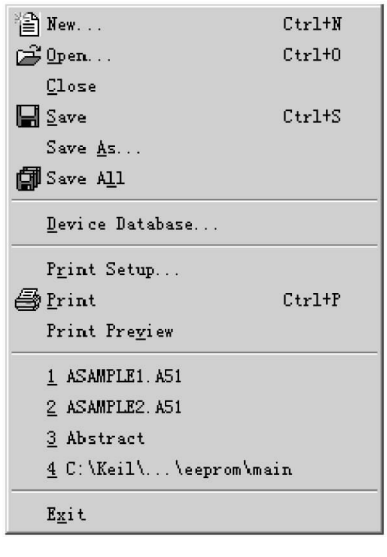


图 6－1　File 菜单

表 6－1　File 菜单命令描述

File 菜单命令	快捷键	描　述
New	Ctrl + N	创建一个新的源文件或文本文件
Open	Ctrl + O	打开已有的文件
Close	—	关闭当前的文件
Save	Ctrl + S	保存当前的文件
Save As...	—	保存并重新命名当前的文件
Save All	—	保存所有打开的源文件和文本文件
Device Database...	—	维护 μVision2 器件数据库
Print Setup...	—	设置打印机
Print	Ctrl + P	打印当前的文件
Print Preview	—	打印预览
1 ~ 9	—	打开最近使用的源文件或文本文件
Exit	—	退出 μVision2，并提示保存文件

2．编辑菜单和编辑器命令

⑴ 菜单及快捷键

在菜单栏中单击 Edit 菜单，如图 6－2 所示。Edit 菜单命令描述如表 6－2、表 6－3所列。

表 6－2　Edit 菜单命令描述

Edit 菜单命令	快捷键	描　述
Undo	Ctrl + Z	撤销上一次操作
Redo	Ctrl + Shift + Z	重做上一次撤销的命令
Cut	Ctrl + X	将选中的文字剪切到剪贴板
	Ctrl + Y	将当前行的文字剪切到剪贴板
Copy	Ctrl + C	将选中的文字复制到剪贴板
Paste	Ctrl + V	粘贴剪贴板的文字
Indent Selected Text	—	将选中的文字向右缩进一个制表符位
Unindent Selected Text	—	将选中的文字向左缩进一个制表符位
Toggle Bookmark	Ctrl + F2	在当前行放置书签
Goto Next Bookmark	F2	将光标移到下一个书签
Goto Previous Bookmark	Shift + F2	将光标移到上一个书签
Clear All Bookmarks	—	清除当前文件中的所有书签
Find	Ctrl + F	在当前文件中查找文字
Replace	Ctrl + H	替换特定的文字
Find in Files...	—	在几个文件中查找文字

图 6－2　Edit 菜单

表 6－3　Edit 其他快捷键描述

快捷键	描　述
Home	将光标移到行的开始处
End	将光标移到行的结尾处
Crtl + Home	将光标移到文件的开始处
Ctrl + End	将光标移到文件的结尾处
Ctrl + ←	将光标移到上一个单词
Ctrl + →	将光标移到下一个单词
Ctrl + A	选中当前文件中的所有文字
F3	继续向前查找文字
Shift + F3	继续向后查找文字
Ctrl + F3	查找光标处（选中）的单词
Ctrl +]	查找匹配的花括号、圆括号、方括号（使用这个命令时将光标移到一个花括号、圆括号或方括号的前面）

⑵ 选择文本命令

在 μVision 中，可以按 Shift 键和相应的光标键来选择文字，例如 Ctrl + →是将光标移到下一个单词，而 Ctrl + Shift + →是选中从光标的位置到下一个单词开始前的文字，也可以用鼠标选择文字，如表 6 - 4 所列。

表 6 - 4　选择文本命令

选择……	用鼠标……
任何数量的文字	在文字上拖动
一个单词	双击这个单词
一行文字	将光标移到行的左边，直到它变成一个指向右的箭头，然后单击
多行文字	将光标移到行的左边，直到它变成一个指向右的箭头，然后向上或向下拖动光标
垂直的一块文字	按着 Alt 键，然后拖动

3. 视图菜单

在菜单栏中单击 View 菜单，如图 6 - 3 所示。View 菜单命令描述如表 6 - 5 所列。

表 6 - 5　View 菜单命令描述

View 菜单命令	快捷键	描　述
Status Bar	—	显示或隐藏状态栏
File Toolbar	—	显示或隐藏文件工具栏
Build Toolbar	—	显示或隐藏编译工具栏
Debug Toolbar	—	显示或隐藏调试工具栏
Project Window	—	显示或隐藏工程窗口
Output Window	—	显示或隐藏输出窗口
Source Browser	—	打开源（文件）浏览器窗口
Disassembly Window	—	显示或隐藏反汇编窗口
Watch & Call Stack Window	—	显示或隐藏观察和堆栈窗口
Memory Window	—	显示或隐藏存储器窗口
Code Coverage Window	—	显示或隐藏代码覆盖窗口
PerformanceAnalyzer Window	—	显示或隐藏性能分析窗口

续表 6－5

View 菜单命令	快捷键	描　述
Symbol Window	—	显示或隐藏符号变量窗口
Serial Window #1	—	显示或隐藏串行窗口 1
Serial Window #2	—	显示或隐藏串行窗口 2
Toolbox	—	显示或隐藏工具箱
Periodic Window Update	—	在运行程序时，周期刷新调试窗口
Workbook Mode	—	显示或隐藏工作簿窗口的标签
Options. . .	—	设置颜色、字体、快捷键和编辑器选项

4. 工程菜单和工程命令

在菜单栏中单击 Project 菜单，如图 6－4 所示。Project 菜单和命令描述如表 6－6 所列。

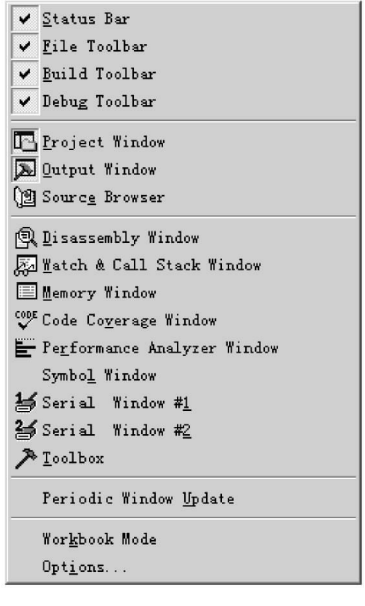


图 6－3　View 视图菜单

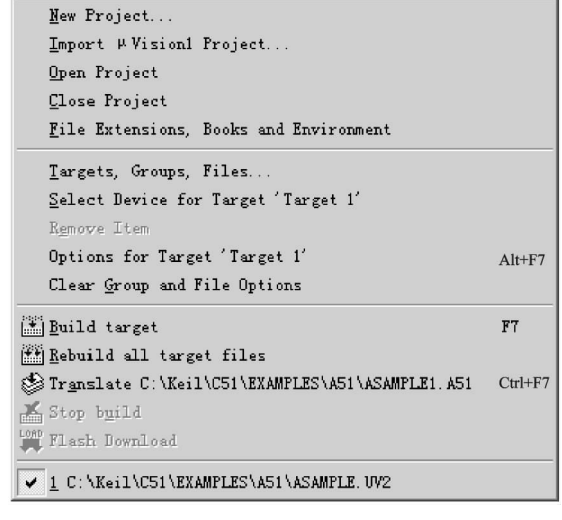


图 6－4　Project 工程菜单

表 6－6　Project 菜单和命令描述

Project 菜单命令	快捷键	描　述
New Project...	—	创建一个新的工程
Import μVision1 Project...	—	输入一个 μVision1 工程文件
Open Project...	—	打开一个已有的工程
Close Project...	—	关闭当前的工程
File Extensions,Books and Environment	—	选择文件的扩展名以区别不同的文件类型，定义工具系列,包含文件、库文件的路径
Targets ,Groups, Files...	—	维护工程的对象、文件组和文件
Select Device for Target...	—	从器件数据库选择一个 CPU
Remove Item	—	从工程中删去一个组或文件
Options for Targets...	Alt + F7	设置对象,组或文件的工具选项
Build target	F7	转换修改过的文件并编译
Rebuild all target files	—	重新转换所有的源文件并编译
Translate...	Ctrl + F7	转换当前的文件
Stop Build	—	停止当前的编译进程
Flash Download	—	
1 ~ 9	—	打开最近使用的工程文件

5. 调试菜单和调试命令

在菜单栏中单击 Debug 菜单,如图 6－5 所示。Debug 菜单和调试命令描述如表 6－7 所列。

表 6－7　Debug 菜单和调试命令描述

Debug 菜单命令	快捷键	描　述
Start/Stop Debug Session	Ctrl + F5	启动或停止 μVision2 调试模式
Go	F5	运行、执行,直到下一个有效的断点
Step	F11	跟踪运行程序
Step Over	F10	单步运行程序
Step Out of current Function	Ctrl + F11	执行到当前函数的程序
Run to Cursor line	Ctrl + F10	执行到光标所在行
Stop Running	Esc	停止程序运行

续表 6－7

Debug 菜单命令	快捷键	描　述
Breakpoints...	—	打开断点对话框
Insert/Remove Breakpoint	—	在当前行设置，清除断点
Enable/Disable Breakpoint	—	使能/禁能当前行的断点
Disable All Breakpoints	—	禁能程序中所有断点
Kill All Breakpoints	—	清除程序中所有断点
Show Next Statement	—	显示下一条执行的语句、指令
Enable/Disable Trace Recording	—	使能跟踪记录，可以显示程序运行轨迹
View Trace Records	—	显示以前执行的指令
Memory Map...	—	打开存储器空间配置对话框
Performance Analyzer...	—	打开性能分析器的设置对话框
Inline Assembly...	—	对某一行重新汇编，可以修改汇编代码
Function Editor	—	编辑调试函数和调试配置文件

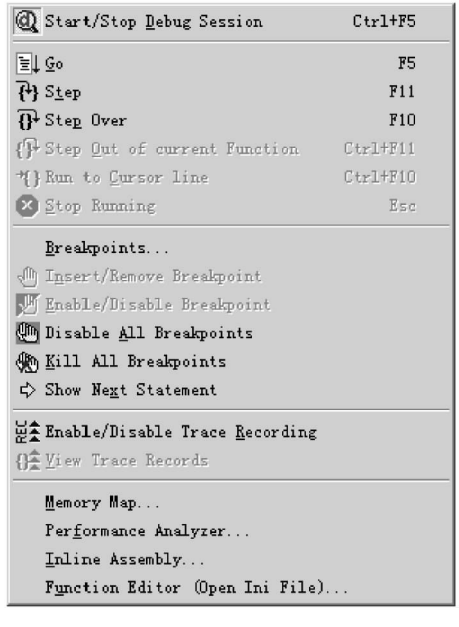


图 6－5　Debug 菜单

6. 外围器件菜单

在菜单栏中单击 Peripherals 外围器件菜单，如图 6－6 所示。Peripherals 菜单命令描述如表 6－8 所列。

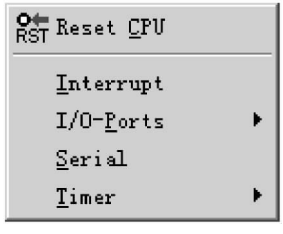


图 6－6　Peripherals 外围器件菜单

表 6－8　Peripherals 菜单命令描述

Peripheral 菜单命令	快捷键	描　述
Reset CPU		复位 CPU
Interrupt I/O-Ports Serial Timer A/D,Converter D/A,Converter I2C,Controller CAN,Controller Watchdog	—	打开在片外围器件的对话框，对话框的列表和内容由在器件数据库中选择的 CPU 决定。不同的 CPU 会有所不同

7. 视窗菜单

在菜单栏中单击 Window 菜单，如图 6－7 所示。Window 菜单命令描述如表 6－9所列。

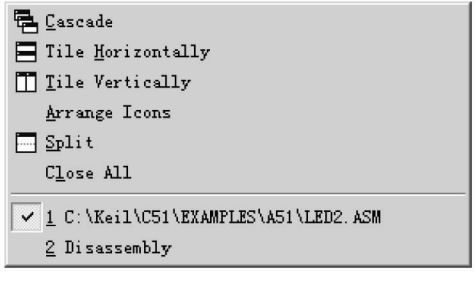


图 6－7　Windows 视窗菜单

表 6－9　Windows 菜单命令描述

Window 菜单命令	快捷键	描　述
Cascade	—	层叠所有窗口
Tile Horizontally	—	横向排列窗口，不层叠
Tile Vertically	—	纵向排列窗口，不层叠
Arrange Icons	—	在窗口的下方排列图标（最小化时）
Split	—	将激活的窗口拆分成几个窗格
Close All	—	关闭所有窗口
1～9	—	激活选中的窗口对象

6.2 Keil 环境中应用系统的调试与仿真

1. 建立工程

在 Keil 中文件的管理使用的是工程的方法，而不是单一文件的模式，所有的文件包括源程序（包括 C 程序和汇编程序）、头文件，甚至说明性的技术文档都可以放在工程里统一管理。

操作：启动 μVision2 后，μVision2 总是打开前一次处理的工程。可以单击菜单 Project→Close Project 关闭。要建立一个新工程，可以单击 Project→New 命令，出现如图 6－8 所示窗口。

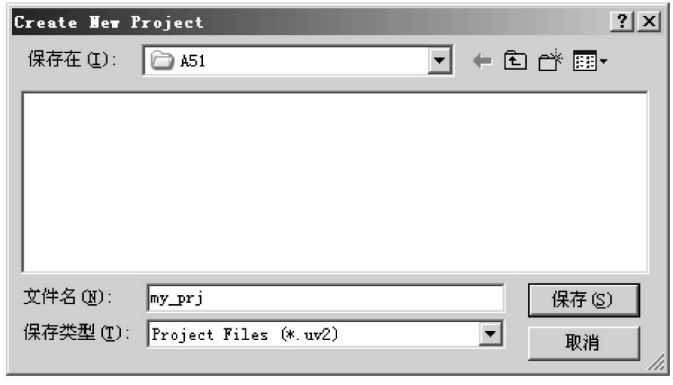


图 6－8 新建工程窗口

填写新建工程的名称：

① 为工程选一个名称，如 my_prj；

② 选择工程存放的路径，建议为每个工程单独建立一个目录，并且工程中需要的所有文件都放在这个目录下；

③ 在选择了工程目录和名称后，单击“保存”按钮，返回。

2. 为工程选择目标器件

在工程建立完毕以后，μVision2 会立即弹出器件选择窗口，如图 6－9 所示。器件选择的目的是告诉 μVision2 使用的 80C51 芯片是哪一个公司的哪一个型号，因为不同型号的 51 芯片内部的资源是不同的，μVision2 可以根据选择进行 SFR 的预定义，在软、硬件仿真中提供易于操作的外设浮动窗口等。

图 6－9 中，μVision2 支持的所有型号根据生产厂家形成器件组，可以点开相应的器件组

并选择相应的器件型号，如选择 ATMEL 器件组里的 AT89C52。另外，如果在选择完目标器件后想重新改变目标器件，可单击菜单 Project→Select Device for…命令，出现图 6－9 所示的对话窗口后重新加以选择。由于不同厂家的许多型号性能相同或相近，因此如果目标器件型号在 μVision2 中找不到，可以选择其他公司的相近型号。

选择正确的仿真器件非常重要，因为仿真时系统将根据所选定的器件提供一些特定的功能菜单，例如选择了 89C52，进入仿真时在 Peripherals 菜单中将会增加很多外设观察选择。

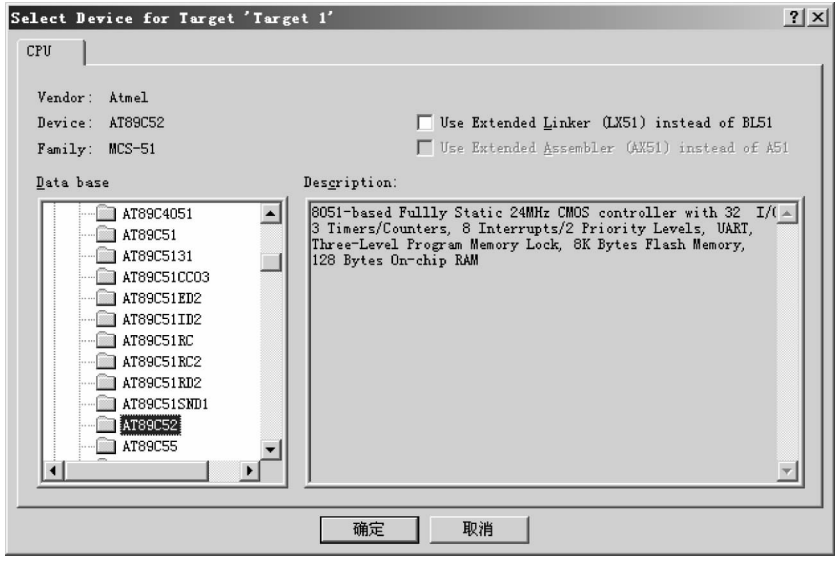


图 6－9　目标器件选择窗口

3. 建立/编辑程序文件

到现在已经建立了一个空白的工程文件，并为工程选择好了目标器件，但是这个工程里没有任何程序文件，程序文件的添加必须人工进行，如果程序文件在添加前还没有创立，用户还必须建立程序文件。

(1) 建立程序文件

单击菜单 File→New 命令后在文件窗口会出现 Text1 新文件窗口，如果多次单击 File→New 命令则会出现 Text2、Text3 等多个新文件窗口。对于新文件框架，还需要把它保存起来，并为它起一个正式的名称。单击菜单 File→Save As 命令，出现图 6－10 所示对话窗口，在文件名栏输入文件的正式名称，如：led. asm。

保存时要注意文件的扩展名，因为 μVision2 要根据扩展名判断文件的类型，从而自动进行处理。如果建立的是一个汇编程序，则输入文件名称 *. asm。唯一需要注意的是文件要保

存在统一的工程目录 my_prj 中，而不要放置在其他的目录中，否则容易造成工程管理混乱。

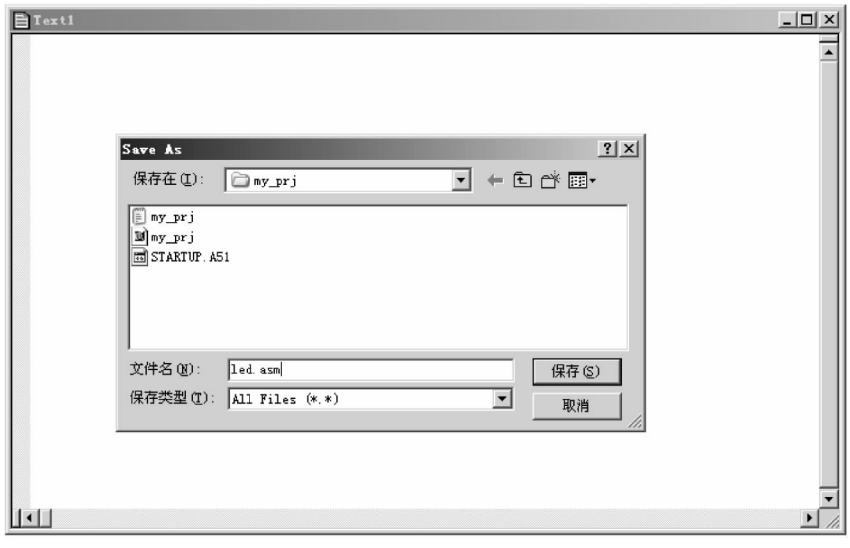


图 6－10　保存新建文件

⑵ 编辑程序文件

上面建立的 led. asm 汇编语言程序文件只是一个空文件，还必须编辑它，并写入程序代码。在 μVision2 中，文件的编辑方法同其他文本编辑器是一样的，用户可以执行输入、删除、选择、复制和粘贴等基本功能。μVision2 不完全支持汉字的输入和编辑，因此如果需要编辑汉字最好使用外部的文本编辑器来编辑，编辑完毕后保存到磁盘中。μVision2 中有文件变化感知功能，提示在外部编辑器改变了该文件，是否需要把 μVision2 中的该文件刷新，选择“是”后 μVision2 中文件刷新。

为了以后说明方便，在这里给出通过 89C52 的并行接口 P1 控制八个发光二极管的程序范例，并给出原理图及详细的程序说明。原理图如图 6－11 所示。

```
/*************************************************/
/* 程序说明                                      */
/* 依次点亮在并行接口 P1 外部的 LED，并无限循环  */
/*************************************************/
        ORG     0000H
        LJMP    START

        ORG     0100H
START:  MOV     R2,#08H         ;设置循环次数
```

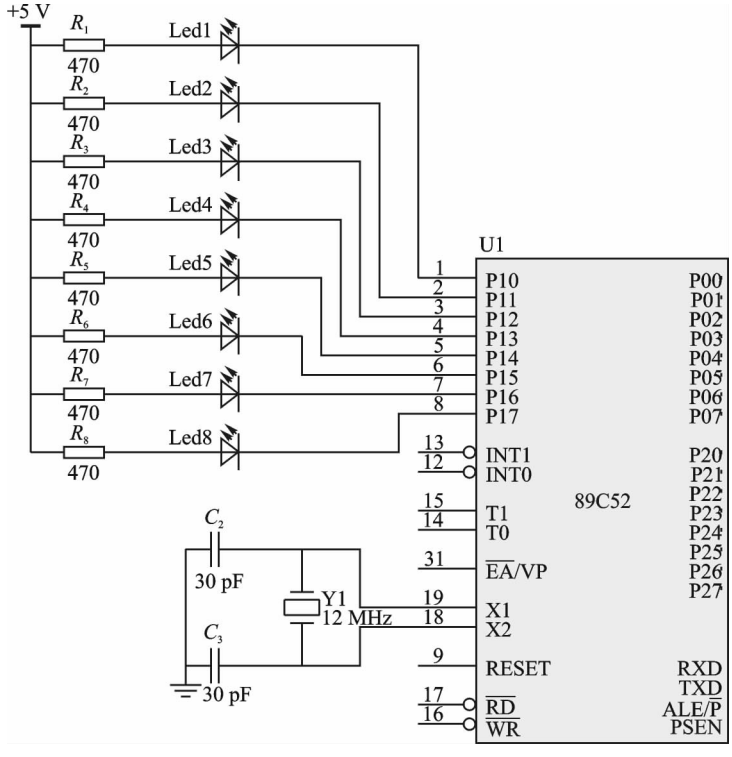


图 6－11　原理图

```
        MOV     A,#0FEH         ;送显示模式字
NEXT：  MOV     P1,A            ;点亮连接 P1.0 的发光二极管
        ACALL   DELAY
        RL      A               ;左移一位,改变显示模式字
        DJNZ    R2,NEXT         ;循环次数减 1,不为零,继续点亮下一个发光二极管
        AJMP    START
/* 延时子程序                    */
DELAY： MOV     R3,#0FFH
DLY1：  MOV     R4,#0FFH
DLY2：  NOP
        DJNZ    R4,DLY2
        DJNZ    R3,DLY1
        RET
```

```
END
/**********************************************/
```

可以把上面的程序复制到 led. asm 文件中，并单击菜单 File→Save 命令加以保存。利用这种建立编辑程序文件的方法，可以同样建立起其他的程序文件。

4. 为工程添加文件

上面建立的程序文件“led. asm”与工程还没有建立起任何关系，接下来要把“led. asm”添加到“my_prj”工程中。

① 右击“Source Group1”弹出如图 6－12 所示菜单，添加工程文件。

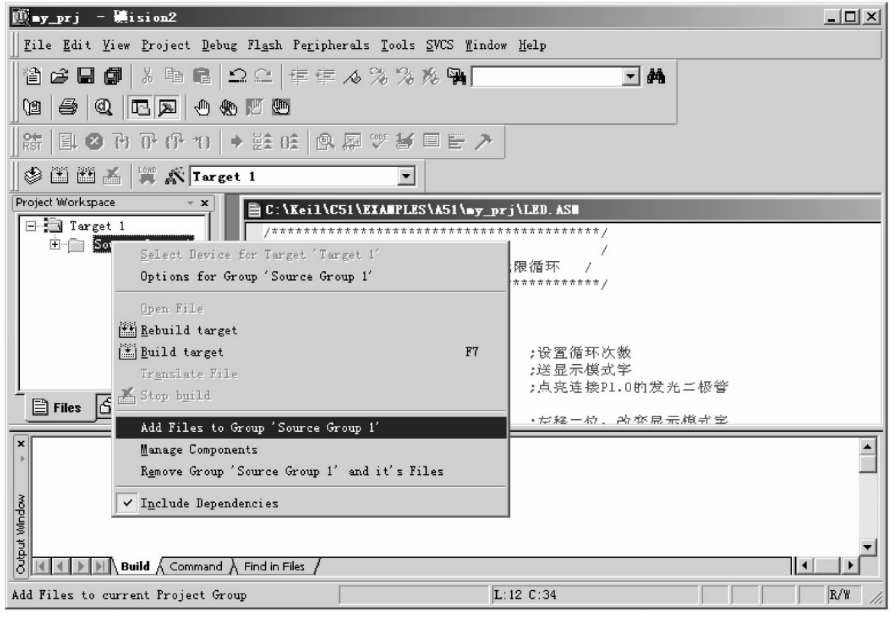


图 6－12　添加工程文件

② 在菜单中选择“Add Files to Group‘Source Group1’(向工程中添加程序文件)”选项后，弹出文件选择窗口，如图 6－13 所示选择要添加的程序文件。可以根据要加入的文件类型显示出所有符合的文件列表，单击“led. asm”，然后单击“Add”按钮，就将“led. asm”加入到工程中。

在图 6－14 中可以看到在工程“Target1”下的组“Source Group1”中已经加入了文件“led. asm”。如果想删除已经加入的程序文件，可以右击该文件，弹出选择菜单，选择“Remove File. . . ”选项，可以将文件从工程中删除。

注意：该文件并没有从磁盘上删除，仍旧保留在原目录下。

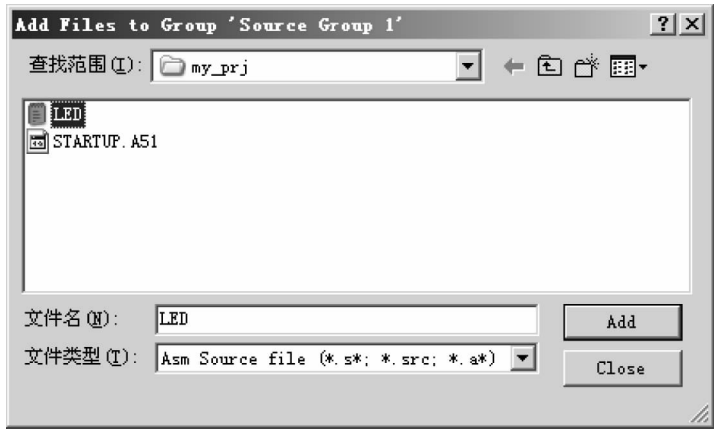


图 6－13　选择要添加的工程文件

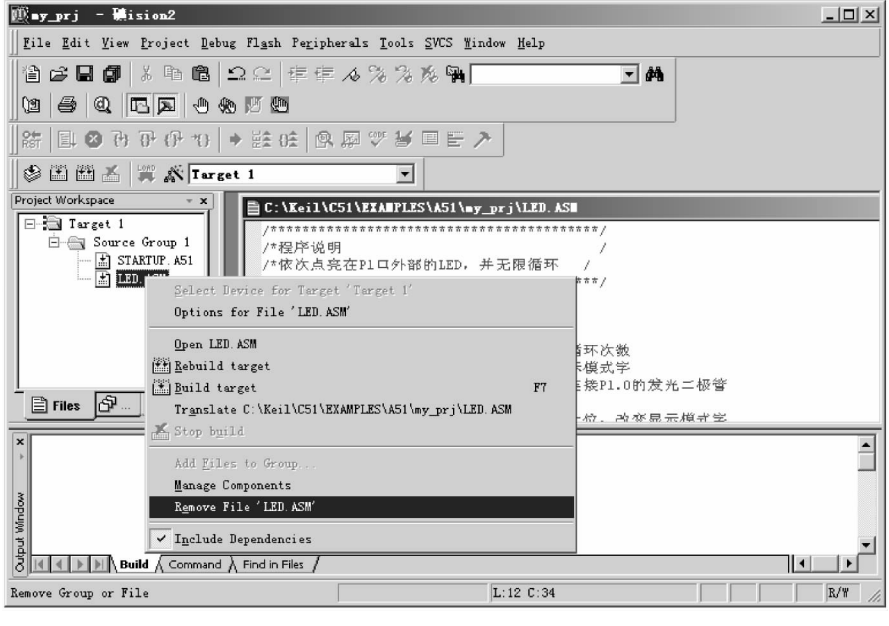


图 6－14　删除工程中的文件

5．对工程进行设置

在工程建立以后，还需要对工程进行设置。工程的设置分软件设置和硬件设置。硬件设置主要针对仿真器，用于硬件仿真时使用 软件设置主要用于程序的编译和连接，也有一些参数用于软件仿真。对于软件和硬件的设置，都应该仔细选择，不恰当的设置会使一些操作无法完成。

右击工程“Target1”，出现选择菜单，如图 6－15 所示。

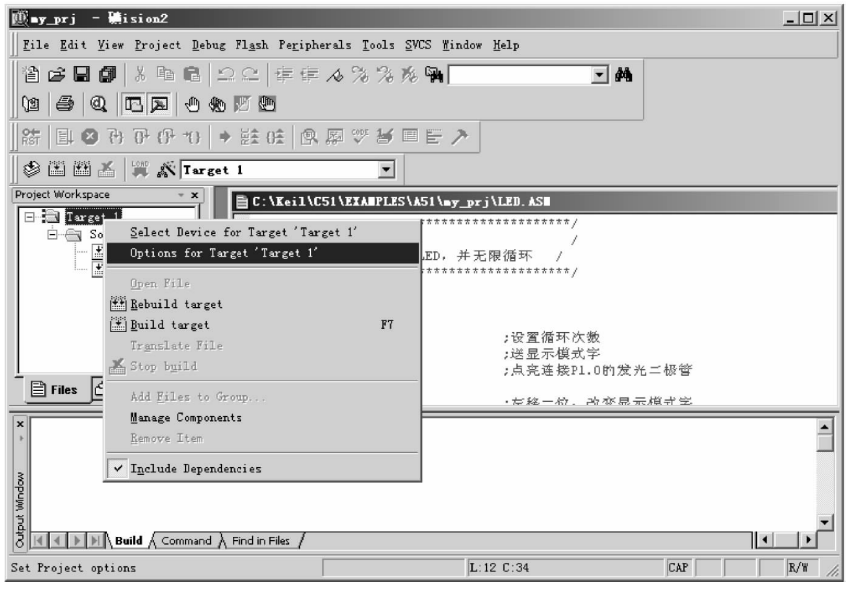


图 6－15　选择工程设置

选择“Option for Target‘Target 1’”选项后，出现工程的配置窗口，如图 6－16 所示。

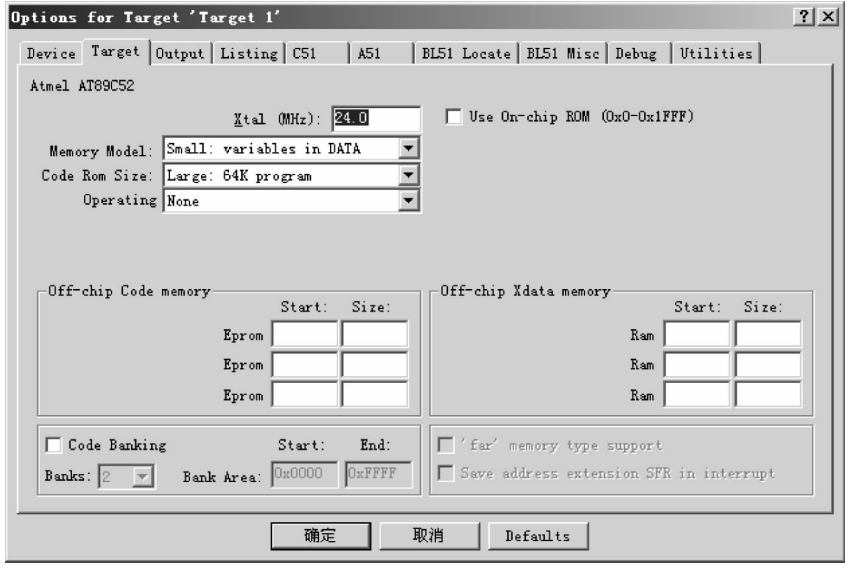


图 6－16　工程配置

一个工程的配置分成以下 10 部分。

Device：选择目标器件。

Target：用户最终系统的工作模式的设定，它决定用户系统的最终框架。

Output：工程输出文件的设定，例如是否输出最终的“Hex”文件以及格式设定。

List：列表文件的输出格式设定。

C51：使用“C51”处理的一些设定。

A51：使用“A51”处理的一些设定。

BL51 Locate：连接时用户资源的物理定位。

BL51 MISC：“BL51”的一些附加设定。

Debug：硬件和软件仿真的设定。

Utilities：Flash 编程的一些设定。

在工程的 10 种设定中，“Target”、“C51”和“Debug”最为重要，其余的设定在一般的工程设计中不需要特别改动，使用 μVision2 的默认设定就可以。

(1) “Target”设置

“Target”设置窗口如图 6－17 所示。

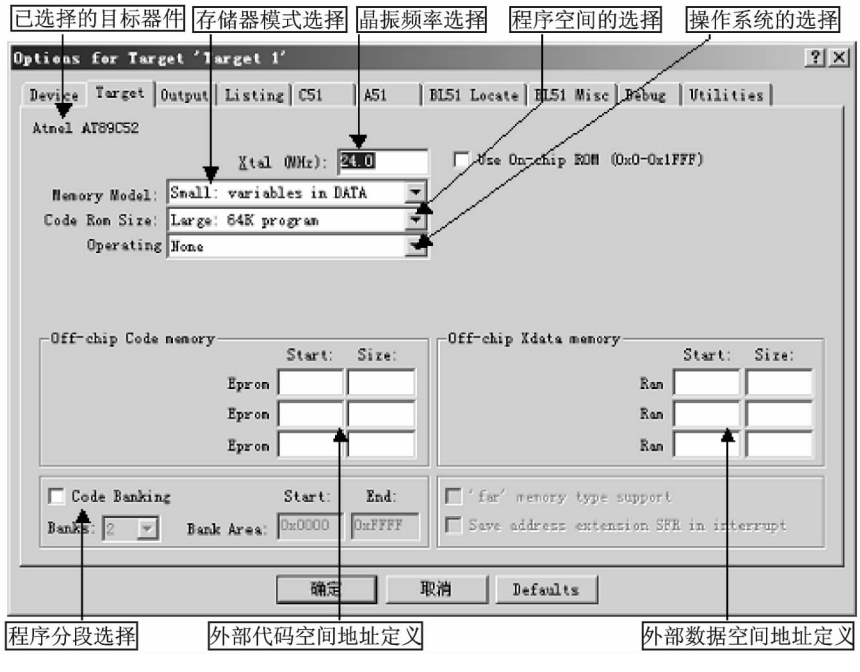


图 6－17　“Target”设置窗口

① 已选择的目标器件：是在建立工程时选择的目标器件型号，这里不能更改，如需要更改，可关闭当前窗体，在工程窗口中单击工程名“Target1”，出现浮动菜单后选择“Select Device for Target ‘Target1’”选项。

② 存储器模式选择：存储器模式有三种可以选择，主要对“C51”中的变量定义起作用。可参看其他资料。

③ 晶振频率选择：晶振的选择主要是在软件仿真时起作用，μVision2 将根据输入频率来决定软件仿真时系统运行的时间和时序。

注意：这个设置在硬件仿真时完全没有作用。

④ 程序空间的选择：选择用户程序空间的大小。

⑤ 操作系统的选择：是否选用操作系统。

⑥ 程序分段选择：是否选用程序分段。

⑦ 外部代码空间地址定义：这个选项主要是用在使用了外部程序空间，但在物理空间上又并不是连续的情况。通过这个选项最多有三个起始地址和结束地址的输入，μVision2 在连接定位时将把程序安排在有效的程序空间内。这个选项一般只用于外部扩展的程序。

⑧ 外部数据空间地址定义：这个选项类似于⑦，但是应用于外部数据空间的定义。

(2)“Debug”设置

“Debug”设置窗口如图 6－18 所示。

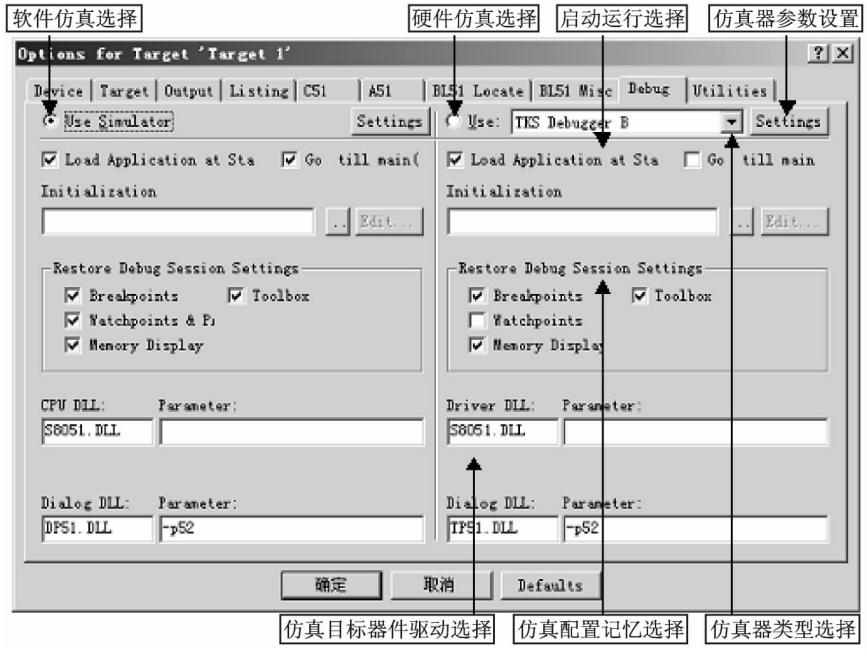


图 6－18 “Debug”设置窗口

“Debug”设置窗口分成两部分：软件仿真设置（左边）和硬件仿真设置（右边）。软件仿真设置和硬件仿真设置基本一样，下面只介绍硬件仿真设置。

在设置硬件仿真前，需要将 TKS－52B 的驱动程序添加到 Keil 的配置文件“Tools. ini”中：首先将 TKS－52B 的驱动程序“TKS_DEB_B. dll”复制到 Keil 的安装目录中。例如，Keil 的安装目录为“C：hKeil”，则将“TKS_DEB_B. dll”复制到“C：hKeil hC51 hbin”目录下，然后打开“C：hKeil”目录下的“Tools. ini”文件，在几个分类中找到“C51”类别，加入下列描述：

TDRV3 = C：hKeil hC51 hbin hTKS_DEB_B. dll（"TKS Debugger B"）

其中，“TDRV3”是驱动“DLL”的序号，如果前面已经安装了多个驱动“DLL”以致占用了 TDRV3，则将 TKS－52B 的驱动程序序号向后顺延，例如“TDRV5”。

① 软件仿真选择/硬件仿真选择：选择当前仿真的模式。软件仿真，使用计算机来模拟程序的运行，不需要建立硬件平台，就可以快速的得到某些运行结果，但是在仿真某些依赖于硬件的程序时，软件仿真则无法实现。硬件仿真是最准确的仿真方法，因为它必须建立起硬件平台，通过 PC↔硬件仿真器↔用户目标系统进行系统调试。适当的结合两种仿真方法，可以快速的对程序进行验证。

② 启动运行选择：选择在进入仿真环境中的启动操作。

Load Application at Start：进入仿真后将用户程序代码下载到仿真器。

Go till main：在使用 C 语言设计时，下载完代码则直接运行到“main”函数位置。

③ 仿真配置记忆选择：对用户仿真时的操作进行记忆。

Breakpoints：选中后记忆当前设置的断点，下次进入仿真后该断点设置存在并有效。

Watchpoints：选中后记忆当前设置的观察项目，下次进入仿真仍有效。

Memory Display：选中后记忆当前存储器区域的显示，下次进入仿真仍有效。

Toolbar：选中后记忆当前的工具条设置，下次进入后仍有效。

④ 仿真目标器件驱动程序选择。

如果在目标器件选择中选择了相应的器件，Keil 将自动选择相应的仿真目标器件驱动程序，由于采用 ATMEL 的 AT89C52，在图中 Keil 选择了标准“S8051. DLL”驱动文件，“Dialog. DLL”选择“TP51. DLL”，根据不同的器件选用不同的仿真驱动“DLL”，这样在仿真时就会有该器件相应的外设菜单，进入仿真（硬件仿真和软件仿真）后，在“Peripherals”菜单中会添加该器件的外设观察菜单，单击后会出现浮动的观察窗口，方便用户观察和修改。

⑤ 仿真器类型选择：用于选择当前 Keil 可以使用的硬件仿真设备，对于 TKS－52B 选择“TKS Debugger B”。

6. 对仿真器进行设置

在工程“Debug”设置窗口中，单击“Settings”命令后出现配置窗体，进入仿真器参数设置，如图 6－19 所示。

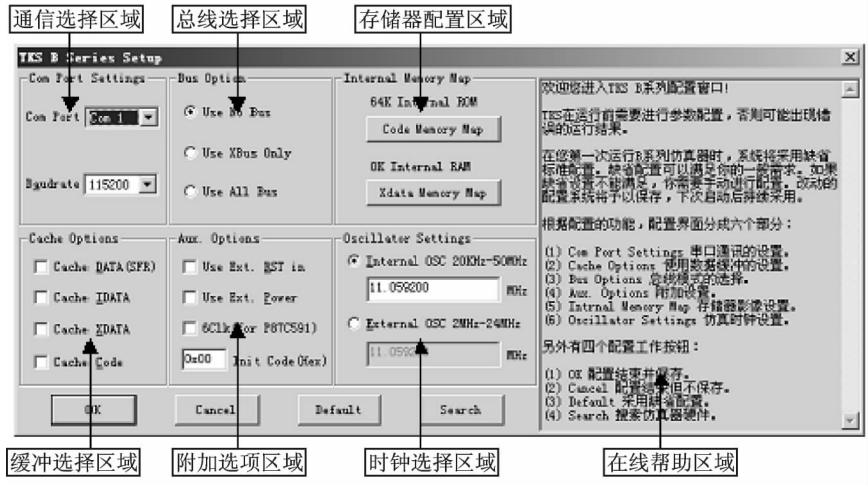


图 6－19　仿真器参数设置

(1) 通信选择区域 (Com Port Settings)

Port：**选择硬件仿真器使用的串口。**TKS－52B **使用串口与** Keil **进行通信。**

Baundrate：**串口使用的比特率。**TKS－52B **使用自动适应比特率，比特率范围在** 1 200 ~ 115 200 bit/s **之间。选择高的比特率能加快运行速度，但是低的比特率在环境恶劣的条件下更可靠。**

注意：选定比特率后，必须先接通 TKS-52B **仿真器电源，仿真器主机面板上的指示灯闪烁结束后，才可以运行** Keil **的硬件仿真程序。在硬件仿真环境中，可以在“**Peripherals**”菜单中打开“**Target Setings**”窗口来更改比特率，更改后整个系统重新建立连接，也可以更改串口，但是必须在更改后的串口上连接仿真器。**

(2) 缓冲选择区域 (Cache Options)

使用存储器缓冲选择区域后，在一般的操作中仿真软件不必频繁的读取仿真器中的内容，而使用缓冲选择区域。使用缓冲可以加快仿真速度，其缺点是屏幕的数据刷新慢，在单步或运行后所有在屏幕上显示的信息将全部刷新一次。

(3) 时钟选择区域 (Oscillator settings)

① Internal OSC 20 kHz ~ 50 MHz：**选择仿真器内部提供的精密线性时钟。**TKS－52B **内部设有一个精密线性时钟发生器，可以产生连续可调产生** 20 kHz ~ 50 MHz **的稳定时钟，精度为** 0.001**。只需要在频率输入框中输入想要的频率数值 (以** MHz **为单位) 即可。输入频率数值后，可以单击窗口其他部分，频率输入框的频率会自动计算并显示出最终产生的频率数值。**

② External OSC 2 ~ 24 MHz：选择仿真器外部时钟。使用应用系统目标板上或仿真头上的晶振，来产生时钟频率。需要在外部时钟频率输入框输入实际的频率，作为精密时间显示的基准。

⑷ 附加选项区域 (Aux. Options)

① Use Ext. RST in：使用外部复位输入选择。如果选择该项，则仿真运行动作 (单步、全速或全速断点) 会根据外部复位信号的电平来动作。如果在开始运行前发现有效的外部复位电平，仿真器将提示并停止运行，等待解除外部复位 如果在运行当中发现有外部复位信号，系统也将提示并复位系统，复位信号解除后重新运行。

② Use Ext. Power：使用外部电源选择。在不使用外部电源时，使用的是内部的 +5 V 电源，要求应用系统目标板的工作电源也必须是 +5 V，但是允许有 0.7 V 的上下偏差。如果需要仿真非 +5 V 供电的系统，则可以选择该项。使用外部电源作为仿真电源，使用的外部电源电压必须在 2.0 ~ 5.5 V 之间。外部输入电源从应用系统目标板得到，经过仿真头电源脚进入仿真器内部。

③ 6Clk for P87C591：为 P87C591 选择时钟模式。对于 TKS - 52B 仿真器该选择项无任何作用。

④ InitCode (Hex)：设置仿真器初始化程序代码。仿真器复位后 (包括上电)，将内部程序空间按照该区域设定的数值进行初始化。在调入应用系统程序代码后，没有使用的区域仍保持初始化的数值。可以直接在该栏文本区域中填入数值，注意必须为十六进制 Hex 数值。主要用于对程序中未用部分设置软件陷阱时使用。

⑸ 总线选择区域 (Bus Option)

为了取得更好的仿真效果，TKS - 52B 仿真器将总线分成三种模式，可以根据实际需要加以选择 选择不同的总线模式只影响 P0/P2 口的状态。

① Use No Bus：不使用任何总线。在仿真时，对于操作总线的情况都加以制止，主要针对 P0/P2 口。P0/P2 口此时是标准的 I/O 模式，遇到 MOVX/MOVC 指令时 P0/P2 口仍然表现出 I/O 特性而不输出总线信号。如果没有使用总线则选择这种模式较好。另外如果仅使用了 MOVX 指令来操作一些单片机的内部 xdata 空间，则也可以选择这种模式。

② Use Xbus Only：仅使用数据总线。仿真时，P0/P2 口将根据指令的运行情况来决定工作模式。遇到 MOVX 指令时 P0/P2 口将输出总线地址/数据，MOVX 指令结束后将恢复 I/O 模式。一个特殊的情况是，如果使用 MOVX @Ri 指令，P2 口仍将表现出 I/O 特性。

③ Use All Bus：使用数据总线和程序总线。仿真时，P0/P2 口完全作为总线使用，可以通过 MOVX 指令操作仿真器外部数据空间，也可以通过 MOVC 指令读取外部代码数据，甚至可以运行外部的程序。在这种模式下，P0/P2 不能再作为 I/O 使用。

注意：总线模式的选择需要同 Internal Memory Map 配合使用。下列的情况必须注意，当需要运行仿真器外部程序时，一定要选择 Use All Bus。

(6) 存储器配置区域(Internal Memory Map)

TKS－52B 内部有 64 KB 的代码空间/64 KB 的数据空间,也可以使用全部/部分外部的程序空间/数据空间。可以把任一地址的程序/数据分配到仿真器内部或外部。

① Code Memory Map:设置内部代码空间。Code Memory Map 窗口中已经设置了几个标准配置分别是仿真器内部 64KB/32KB/16KB/8KB/4KB/0KB,则仿真器外部代码空间为 0KB/32KB/48KB/56KB/60KB/64KB。若标准设置不能满足要求,可以添加内部代码范围。需要注意的是添加的是内部代码空间范围,没有涉及到的空间地址影像到仿真器外部。

按钮上方的文字表明当前代码分配的情况,例如 8 KB Internal ROM 表明 ROM 状态为仿真器内部 8 KB,外部 ROM 空间为 64 KB - 8 KB = 56 KB。如果需要改变当前的 ROM 分配情况,可以单击"Code Memory Map"按钮,进入图 6－20 所示 ROM 空间分配界面,进行修改。

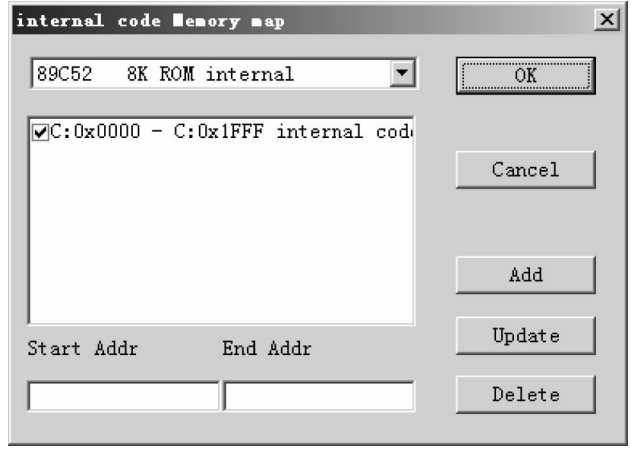


图 6－20 ROM 空间分配界面

如果选择的模式属于以下的标准模式,可以直接使用标准 ROM 模式选择下拉列表选择,可以选择的 ROM 模式有以下几种。

89C516:仿真器内部 ROM 大小为 64 KB 仿真器外部 ROM 大小为 0 KB;

89C58:仿真器内部 ROM 大小为 32 KB 仿真器外部 ROM 大小为 32 KB;

89C54:仿真器内部 ROM 大小为 16 KB 仿真器外部 ROM 大小为 48 KB;

89C52:仿真器内部 ROM 大小为 8 KB 仿真器外部 ROM 大小为 56 KB;

89C51:仿真器内部 ROM 大小为 4 KB 仿真器外部 ROM 大小为 60 KB;

80C31:仿真器内部 ROM 大小为 0 KB 仿真器外部 ROM 大小为 64 KB。

一般情况下,标准模式基本涵盖了要仿真的 CPU 类型,特殊情况可通过 user define 来设定 ROM 空间。

② Xdata Memory Map:设置内部数据空间。Xdata Memory Map 窗口中已经为用户设置了

几个标准配置，分别是仿真器内部 64KB/32KB/16KB/8KB/4KB/0KB，则仿真器外部数据空间为 0KB/32KB/48KB/56KB/60KB/64KB。如果标准设置不能满足要求，可以添加内部数据空间范围。需要注意的是添加的是仿真器内部数据空间范围，没有涉及到的空间地址影像到仿真器外部。

按钮上方的文字表明当前 Xdata 分配的情况。例如 0 KB Internal RAM 表明 RAM 状态为仿真器内部 0 KB，外部 RAM 空间为 64 KB - 0 KB = 64 KB。如果需要改变当前的 RAM 分配情况，可以单击“Xdata Memory Map”按钮进入图 6－21 RAM 空间分配界面，进行修改。

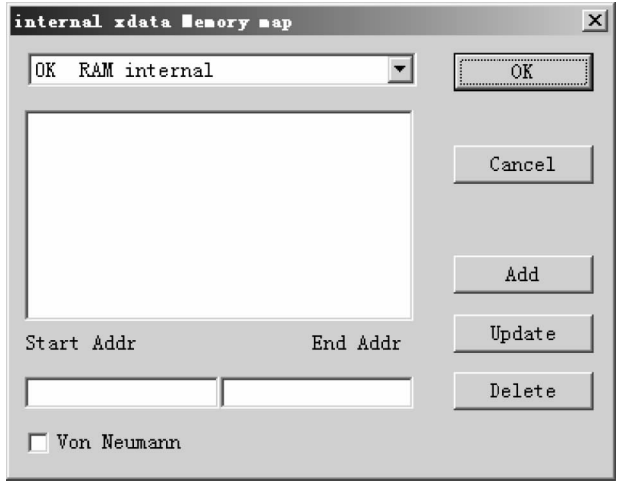


图 6－21　仿真器内部数据空间分配界面

如果选择的模式属于以下的标准模式，可以直接使用标准 RAM 模式选择下拉列表选择，可以选择的 RAM 模式有以下几种。

64 KB 模式：仿真器内部 RAM 大小为 64 KB 仿真器外部 RAM 大小为 0 KB；

32 KB 模式：仿真器内部 RAM 大小为 32 KB 仿真器外部 RAM 大小为 32 KB；

16 KB 模式：仿真器内部 RAM 大小为 16 KB 仿真器外部 RAM 大小为 48 KB；

8 KB 模式：仿真器内部 RAM 大小为 8 KB 仿真器外部 RAM 大小为 56 KB；

4 KB 模式：仿真器内部 RAM 大小为 4 KB 仿真器外部 RAM 大小为 60 KB；

0 KB 模式：仿真器内部 RAM 大小为 0 KB 仿真器外部 RAM 大小为 64 KB。

7. 对工程进行编译/连接

在 μVision2 环境中，程序编写完毕后还需要编译和连接才能够进行软件和硬件仿真。在程序的编译/连接中，如果程序出现错误，则需要修正错误后重新编译/连接。

⑴ 程序的编译/连接

单击 Project 菜单，选择“Rebuild all target files”选项，如图 6－22 所示。

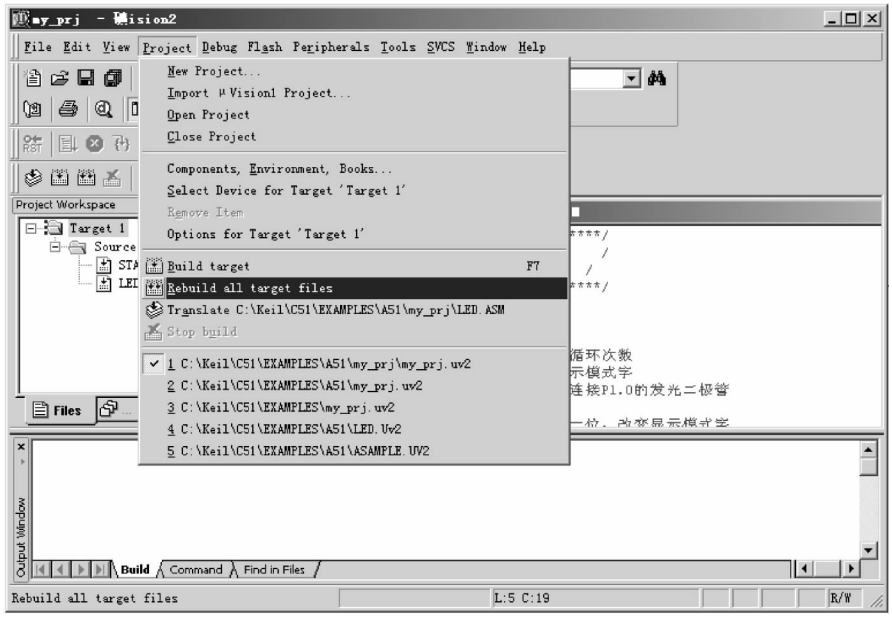


图 6－22 请求编译/连接工程

如果用户程序和工程设置没有错误，编译和连接将能顺利完成，操作信息在图 6－23 信息输出窗口提示。

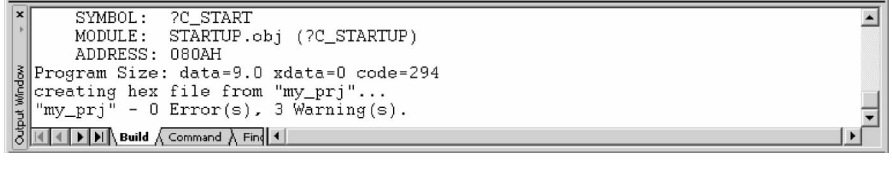


图 6－23 信息输出窗口

⑵ 程序的排错

若程序中有错误，如“MOV P1，A”误写成“MOV @P1，A”，进行编译和连接后，结果如图 6－24 所示。在结果输出中有详细的错误提示，这次错误原因是 LED. ASM (10)：error A9：SYNTAX ERROR，在文件“led. asm”的 10 行，出现 SYNTAX ERROR 类型的错误。μVision2 中有错误定位功能，在输出窗口双击错误提示行，“led. asm”文件中的错误所在行被明显的作了标记，并以当前文件的方式显示出来。经过排错后，要重新进行编译和连接，直到编译和连接成功。

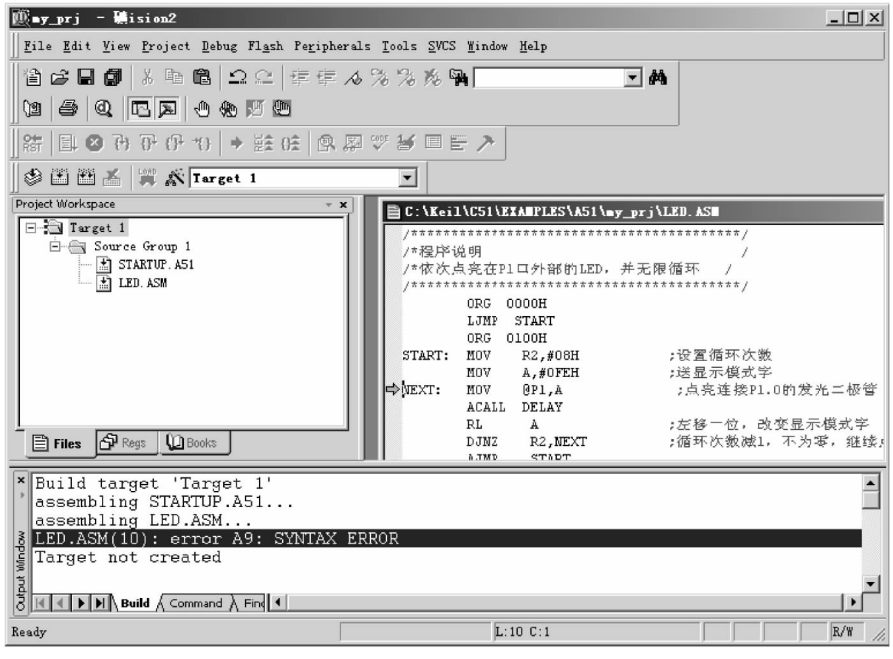


图 6－24　编译后的错误定位

8. 进入硬件仿真前的准备工作

在进入硬件仿真前，还需要将 TKS－52B 仿真器准备好，并按以下步骤操作：

① 将 TKS－52B 仿真器安放在稳定、安全、平整、操作方便的桌面上；

② 用仿真电缆把仿真头组件连接到仿真器上；

③ 用通信电缆将仿真器同计算机连接起来，并拧紧固定螺钉；

④ 将随机提供的电源适配器插到 220 V 交流插座中，输出插头接入到 TKS－52B 仿真器；

⑤ 如果需要接入应用系统目标板，在 TKS－52B 自检完成后，将仿真头插入应用系统目标板插座，然后开启应用系统目标板电源。

9. 进入硬件仿真

在设置中选择硬件仿真，单击菜单“Debug”→“Start/Stop Debug Session”，进入 µVision2 的硬件仿真环境。

如果出现通信失败将无法进入硬件仿真，出现这种现象的原因一般是：

① 仿真器同 PC 的通信电缆接触不良；

② 仿真器没有电源；

③ 错误的选择了硬件驱动程序；

④ 串口选择错误，或者该串口已经被其他设备占用；

⑤ 用户串口已经损坏。

排除故障后，重新进入，图 6－25 所示为硬件仿真环境。

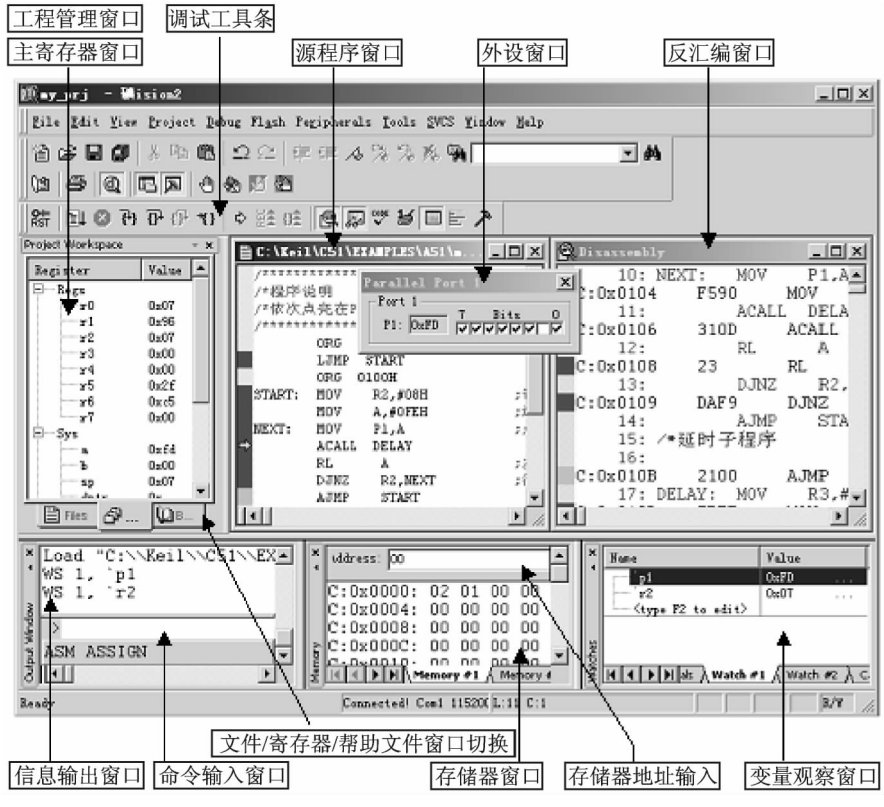


图 6－25 硬件仿真环境

在 μVision2 中，任何窗口都可以显示或关闭，对于已经显示的窗口，可以单击该窗口上的 Windows 标准关闭图标来关闭该窗口。对于没有显示的窗口，可以单击“View”菜单里面的不同选项，用于开启或关闭不同的显示内容。

Project Windows：开启或关闭工程管理窗口、主寄存器窗口、帮助文件窗口。

Output Windows：开启或关闭信息输出窗口。

Disassembly Windows：开启或关闭反汇编窗口。

Memory Windows：开启或关闭存储器窗口。

Watch and Call Stack Windows：开启或关闭变量观察窗口。

在 μVision2 中，还可以开启一些外设窗口，外设窗口的内容与选择的目标器件有关系，例

如，如果选择 89C51，外设中就没有 T2 这个选项，只有选择了 89C52 才可能出现 T2 外设选项。外设窗口只有在进入仿真环境中才有可能打开或关闭。在仿真环境中，单击菜单“Peripherals”就可以看到可以使用的外设或外设组，单击对应的外设就可以打开或关闭该外设窗口。外设窗口以简单的形式告诉用户现在外设的状态，例如图 6－26 所示的外设窗口并行接口 P1，在仿真运行时可以很容易地看出并行接口 P1 各个 I/O 的状态。

⑴ 工程管理窗口

用于工程的管理，包含文件窗口、寄存器窗口、帮助文件窗口三部分。可以使用文件窗口、寄存器、帮助文件窗口切换标签进行切换。

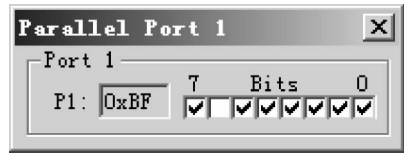


图 6－26　外设窗口

① 文件窗口：显示当前工程的组织结构，用户可以添加/删除文件和文件夹。

② 帮助文件窗口：显示当前 uV2 所有的帮助文件，可以单击某文件进行阅读。

③ 主寄存器窗口：显示/修改主寄存器内容，该窗口只在进入仿真后有效。TKS－52B 的主寄存器窗口中有精密时间显示。

● 查看和修改主寄存器的内容

仿真时主寄存器的内容在主寄存器窗口显示，除了可以观察以外，还可以修改。单击选中一个单元，例如 SP，然后再单击 SP 的数值位置，出现文字框后输入希望数值按 Enter 键即可。另外的输入方法是使用命令行窗口。例如，输入 A＝0X34，将把 A 的数值设置为 0X34。

● 运行时间的使用

总运行时间 Tsum：仿真系统从上电或复位后到当前状态经历的有效运行总时间。在监控状态并不会使该时间增大，只有有效的运行操作，例如单步/全速运行，才能使之增加。

当前运行时间 Tcur：记录了前一次有效运行操作经历的时间。例如运行一个单步经历了 1 μs，则该时间显示为 1 μs；再运行一个单步经历了 2 μs，则该时间显示为 2 μs。与总运行时间的不断累加不同，该时间是一个时间差，便于观察本次操作经历的时间。

总运行时钟数 Nsum：仿真系统从上电或复位后到当前状态经历的振荡周期数。在监控状态并不会使该时间增大，只有有效的运行操作，例如单步/全速运行，才能使之增加。

当前运行时钟数 Ncur：记录了前一次有效运行操作经历的振荡周期数。例如，运行一个单步经历了 12Clk，则该时间显示为 12Clk；再运行一个单步经历了 12Clk，则该时间显示为 12Clk。与 Nsum 的不断累加不同，该时间是一个差值，便于观察本次操作经历的时钟数。

注意：

① Tsum/Tcur 显示的数值跟用户选择的时钟有关系。如果选择仿真器内部提供的时钟，仿真器能完全准确地显示 Tsum/Tcur。如果选择外部时钟，仿真器需要知道从外部输入时钟的频率。如果填写的时钟频率不准确，Tsum/Tcur 显示的数值将是错误的。

② Nsum/Ncur 显示的是运行中经过的时钟周期数，显示的数值是独立的，即使在选择外

部时钟时也不依赖于用户设置的时钟频率。

③ 复位后 Tsum/Tcur/Nsum/Ncur 的数值全部为 0 且不能改动。

图 6－27 源程序窗口的操作菜单

⑵ 源程序窗口

用户编写并加入工程的 c/asm 程序代码。

① 源程序窗口的开启。源程序必须是编写的 C 语言或汇编语言程序，并已经加入了当前的工程中。如果某一个源文件只是在 uV2 中使用主菜单“File”→“Open”命令打开，但是没有加入工程中，则文件在 uV2 中只能进行编辑，不能进行调试，因此不是通常所说的源文件。

如果需要的源文件没有显示，可以在主菜单“Window”中查看该文件是否存在，如果存在则单击该文件，使之成为当前窗口；如果该文件没有存在，可以在“Project WorkSpace”中选择“File”查看文件结构并单击打开需要的源文件。

② 源程序窗口的操作。在源程序窗口中右击，可以看到多种针对源程序的操作，如图 6－27 所示。

Undo：取消操作

Redo：重复操作

Cut：剪切

Copy：复制

Paste：粘贴

Toggle Bookmark：在当前行插入/取消书签

Show Dissembly at：在汇编窗口显示汇编内容

Set Program Counter：设置当前 PC 指针

Run to Cursor line：运行到光标所在行

Goto to line：跳转到指定行

Insert/Remove BreakPoint：插入/取消断点

Enable/Disable Breakpoint：启动/关闭断点

Clear complete Code Coverage Info：清除全部覆盖信息

⑶ 反汇编窗口

以反汇编的形式显示当前的用户程序。

① 反汇编窗口的开启。如果没有出现反汇编窗口，可以从“View”→“Disassembly Window”打开反汇编窗口。在反汇编窗口中右击，可以看到多种针对反汇编的操作，包括反汇编

显示模式、在线汇编和程序代码的调入。

② 反汇编窗口的操作。在反汇编窗口中右击，将弹出操作菜单，如图 6－28 所示，可以选择需要的操作。

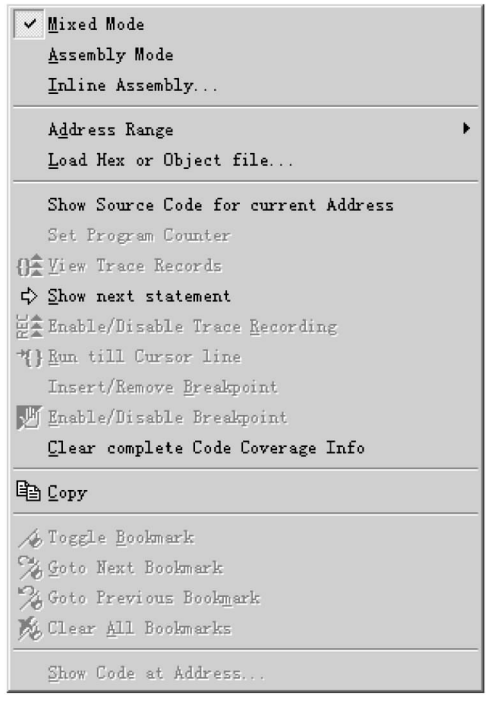


图 6－28　反汇编窗口的操作菜单

Mixed Mode：混合反汇编模式插入源程序语句。

Assembly Mode：反汇编模式下源程序语句。

Inline Assembly：对当前行的反汇编语句进行在线修改。

Address Range：调整当前窗口的地址范围。

Load Hex or Object file...：下载用户程序代码 Hex 或 Object。

Show source Code for current Addrss：显示该汇编行在源程序窗口对应的代码。

Set Program Counter：将当前的 PC 指针设置到当前的汇编行。

View Trace Records：显示跟踪汇编行。

Show next statement：显示当前 PC 指针位置。

Enable/Disable Trace Recording：开启/关闭跟踪记录。

Run till Cursor line：全速运行到当前汇编行

Insert/Remove Breakpoint：插入/取消断点

Enable/Disable Breakpoint：启动/关闭一个断点

Clear complete Code Coverage Info：清除全部的代码覆盖信息。

Copy：复制当前行的反汇编内容。

Toggle Bookmark：在当前行插入/取消书签

Goto Next Bookmark：跳转到下一个书签位置

Goto Previous Booksmark：跳转到上一个书签位置

Clear All Bookmark：清除全部书签

Show Code at Address...：显示汇编行代码

10. 运行程序

在 μVision2 中，有以下的几种程序运行方式：

(1) 单步跟踪 (Step info)

单步跟踪是尽最大的可能跟踪当前程序的最小运行单位。单步跟踪可以使用“F11”键来启动，也可以使用菜单“Debug”→“Step info”命令，另外单击“Step info”图标也可以启动单步跟踪。为了描述方便，下面示例中使用“F11”键。

运行范例程序“my_prj”，把“led. asm”汇编窗口作为当前窗口（也可以把反汇编窗口作为当前窗口）。

在汇编窗口中，首次运行时程序指针指在（<...>内为反汇编窗口指针位置）：

LJMP START <C:0x0000 020100 LJMP START (C:0100)>

按“F11”键后，程序指针应该指在：

START：MOV R2,#08H <C:0x0100 7A08 MOV R2,#0x080100>

到现在为止，按“F11”键，每次都只运行一句汇编指令，连续按“F11”键，使程序指针指在：

ACALL DELAY <C:0x0106 310D ACALL DELAY (C:010D)>

现在程序准备调用子程序 DELAY，按“F11”键，执行单步跟踪操作，可以看到程序指针指在：

DELAY：MOV R3,#0FFH <C:0x010D 7BFF MOV R3,#0xFF>

跟踪进入了 DELAY 子程序，如图 6－29 所示。也就是说程序单步跟踪每一个汇编指令的执行。与 Step info 不同，在下面的 Step over 的操作中，可以看到，在汇编窗口下执行单步运行后将整个 ACALL 指令执行完毕，并不跟踪到子程序内部。

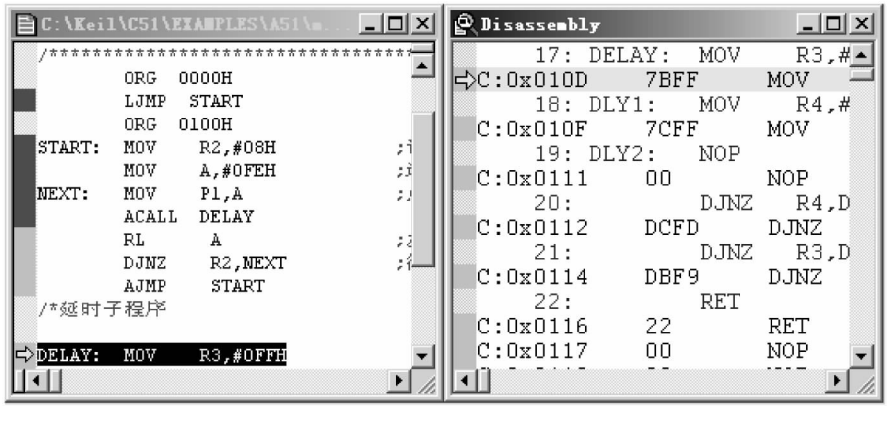


图 6－29 运行程序的汇编和反汇编窗口

(2) 单步运行 (Step over)

单步运行是尽最大的可能执行完当前的程序行。单步运行可以使用“F10”键来启动，也可以使用菜单“Debug”→“Step over”命令，另外单击“Step over”图标也可以启动单步跟踪。为了描述方便，下面示例中使用“F10”键。

运行范例程序“my_prj”，把“led. asm”汇编窗口作为当前窗口（也可以把反汇编窗口作为当前窗口）。

在汇编窗口中，首次运行时程序指针指在（< … > 内为反汇编窗口指针位置）：

LJMP START　< C:0x0000　020100　LJMP START (C:0100) >

按“F10”键后，程序指针应该指在：

START：MOV R2,#08H　< C:0x0100　7A08　MOV R2,#0x080100 >

到现在为止，按“F10”键，每次都只运行一句汇编指令，跟“F11”键的执行完全一致。连续按“F10”键，使程序指针指在：

ACALL　DELAY　< C:0x0106　310D　ACALL　DELAY (C:010D) >

现在程序准备调用子程序 DELAY，按“F10”键，执行单步运行操作，可以看到程序指针指在：

RL　A　< C:0x0108　23　RL　A >

也就是说程序执行完毕了 ACALL　DELAY，而不是跟踪进入 DELAY 子程序，跟“F11”键的执行不一致。

（3）连续单步运行

TKS－52B 仿真器有连续单步运行的功能，每一步运行完后，将刷新屏幕，然后执行下一步运行。将反汇编窗口置为当前窗口，在命令窗口输入，例如 T 1000 的命令（1000 表示运行 1000 个汇编单步），然后就可以看到程序快速的执行单步操作，PC 指针不断的移动，屏幕上的数据也会发生变化。

（4）运行到光标处（Run till cursor line）

在汇编窗口和反汇编窗口中均可打开窗口浮动菜单，选择“Run till cursor line”选项将程序运行到光标所在行。

（5）全速运行

TKS－52B 仿真器在全速运行期间将实时仿真 MCU 的特性，全速运行的启动有三种方法：按“F5”键、单击图标、单击菜单“Debug”→“Go”。

当 μVision2 处于全速运行期间，μVision2 不允许任何资源的查看，也不接受其他的命令。如果想终止程序的运行，可以单击菜单“Debug”→“Stop Running”命令或单击图标。

在范例程序“my_prj”中，如果仿真器接到应用系统目标板，且目标板上的并行接口 P1 接有发光二极管（LED），进入全速运行后会看到 LED 交替连续的闪亮。

11. 程序复位

如果想重新开始运行应用程序，可以对仿真器的应用程序进行复位。由于 80C51 芯片复位后程序计数器将从 0000H 重新开始，因此 TKS－52B 仿真器复位后程序计数器（PC 指针）将复位成 0000H。另外，一些内部特殊功能寄存器在复位期间也将重新赋值，例如 A 将变为

00H，DPTR 变为 0000H，SP 变为 07H，I/O 接口变为 0FFH。

程序的复位有以下几种办法实现：

① 单击图标；

② 单击菜单 Peripherals→Reset CPU 命令；

③ 在命令输入窗口输入 Reset。

12. 断点操作

当需要程序全速运行到某个程序位置停止时，通常可以使用断点。TKS－52B 仿真器有 4×64 KB 断点，分别是程序运行断点、MOVC 读取断点、MOVX 读取断点、MOVX 写入断点等四种。这四种断点可以单独分别使用，也可以混合使用。

(1) 断点种类

程序运行断点：程序运行到该指令位置时停止，进入仿真监控状态。

MOVC 读取断点：当 MOVC 指令读取该地址时，程序在完成操作后停止进入仿真监控状态。

MOVX 读取断点：当 MOVX 指令读取该地址时，程序在完成操作后停止进入仿真监控状态。

MOVX 写入断点：当 MOVX 指令写入该地址时，程序在完成操作后停止进入仿真监控状态。

(2) 程序运行断点的定义

程序运行断点有非常简单的定义方法。可以在源程序窗口（包括 C 语言和汇编）和反汇编窗口中，双击想要设置断点的程序行，则在窗口左边的状态条中出现红色的断点标志块。再次双击该程序行，则为取消断点，取消断点后窗口左边的红色的断点标志块消失。

在源程序窗口中，按照这种方法有的时候无法在某些程序行中设置断点。出现这种情况的原因一般是该源程序行无对应的程序代码。

也可以使用断点菜单和断点工具条。在源程序窗口（包括 C 语言和汇编）和反汇编窗口中，用鼠标或键盘上的上移键或下移键移动光标到想要的程序行点，单击菜单“Debug”→“Insert/Remove Breakpoint”来插入/删除一个断点。

(3) 程序运行断点的关闭/启动（Enable/Disable）

断点的操作有插入断点、关闭断点、取消断点三种，其中关闭断点和取消断点是不同的。取消断点是把该断点取消，当需要时应再次设置；关闭断点是对已经插入的断点暂时关闭，可以在以后再次简单的启动。对于已经定义的断点，可以单击主菜单“Debug”→“Enable/Disable Breakpoint”来启动/关闭一个断点。

(4) 用断点管理器管理断点

由于程序运行断点直接和程序相对应，可以使用简单的方法来设置。对于其他三种断点，

需要启动断点管理器来进行断点的处理。单击主菜单“Debug”→“Breakpoints”打开断点管理器，如图 6－30 所示。

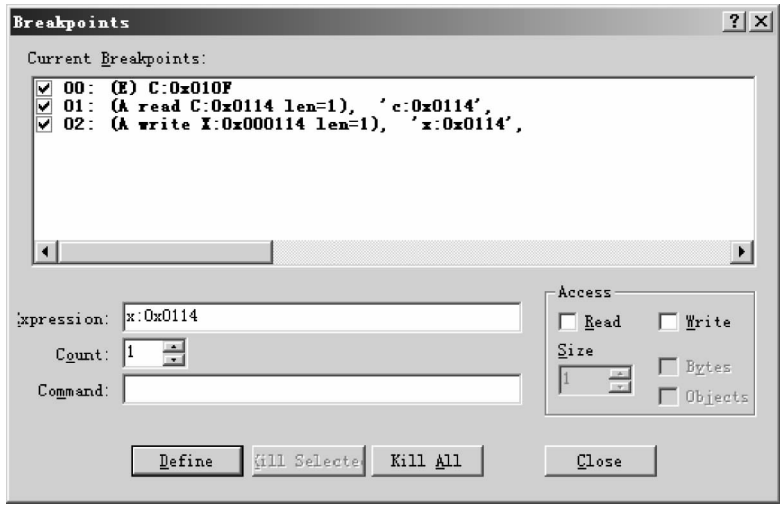


图 6－30　断点管理窗口

断点管理器主要功能部件如下：

Current Breakpoints：当前所有的断点显示窗口。

Expression：断点的表达式。

Count：触发断点的次数（TKS－52B 不支持）。

Command：断点命令输入（TKS－52B 不支持）。

Read：断点操作属性为读。

Write：断点操作属性为写

Size：断点的尺寸，以 Byte 为单位；TKS－52B 只支持 Size＝1 Byte。

Byte：TKS－52B 不支持。

Objects：TKS－52B 不支持。

Define：定义一个断点。

Kill Selected：删除选择的断点。

Kill All：删除全部的断点

① 定义程序运行断点。在 Expression 中写入例如 C：0x010F 的表达式，C 表示程序属性；0x010F 表示程序代码地址。因为是程序运行断点，Access 中的 Read 和 Write 属性均不选择。然后单击“Define”按钮，在 Current Breakpoints 窗口中将出现该断点的表达式：

00：(E) C：0x010F，‘C：0x010F’。

② 定义 MOVC 读取断点。在 Expression 中写入例如 C：0x0114 的表达式，C 表示程序属

性 D:0x0114 表示程序代码地址。因为是 MOVC 读取断点，Access 中的 Read 需要选中，Write 属性不选中。然后单击“Define”按钮在 Current Breakpoints 窗口中将出现该断点的表达式：

01：(A Read C:0x0114 Len = 1),'C D:0x0114'

注意：在定义 MOVC 读取断点时，Read 属性一定要选中，否则将成为程序运行断点。Write 不要选中，因为程序代码无法进行写操作。

③ 定义 MOVX 读取/写入断点。在 Expression 中写入例如 X D:0x0114 的表达式，X 表示是数据空间属性 D:0x0114 表示为数据地址。因为是 MOVX 操作断点，Access 中的 Read/Write 至少必须有一项选中。然后单击“Define”按钮在“Current Breakpoints”窗口中将出现该断点的表达式：

02：(A Read X:0x000114 Len = 1),'X D:0x0114'

注意：在定义 MOVX 读取断点时，Read/Write 至少必须有一项选中，否则将不能形成有效断点。

④ 关闭已经存在的断点。对于“Current Breakpoints”窗口中存在的每个断点，都可以暂时关闭，使之断点作用失效。由于该失效的断点仍然存在于“Current Breakpoints”窗口中，因此户可以在以后很方便重新起用它，这比删除这个断点然后重新增加一个断点要方便得多。

13. 空间资源的查看和修改

在 μVision2 的仿真环境中，标准 80C51 的所有有效空间资源都可以查看和修改。μVision2 把空间资源分成几种类型加以管理。

(1) 片内可直接寻址 RAM(或 SFR)(类型 data，简称 d)

在标准 80C51 中，可直接寻址空间为 0 ~ 0x7F 范围内的 RAM 和 0x80 ~ 0xFF 的 SFR(SFR，特殊功能寄存器)。在 μVision2 中把它们组合成空间连续的可直接寻址的 data 空间。data 空间有下列几种查看和修改方法。

① 使用存储器窗口(Memory Windows)

● 存储器窗口的打开：单击“View”→“Memory Windows”命令可以打开存储器窗口，如图 6－31 所示。

存储器地址输入栏：用于输入空间类型和起始地址。

存储器地址栏：显示每一行的起始地址，便于观察和修改。

存储器数据区域：数据显示区域，显示格式可以改变。

存储器窗口组：分成独立的四个组，每个组可以单独定义空间类型和起始地址，单击组图标可以切换。

● 存储器窗口中数据的修改：右击需要修改的数据，出现浮动菜单，如图 6－32 所示。

选择 Modify Memory at D:0x4A，表示要改动 data 区域 0x4A 地址的数据内容。选择后系统会出现输入栏，输入新的数值后单击 OK 按钮返回。

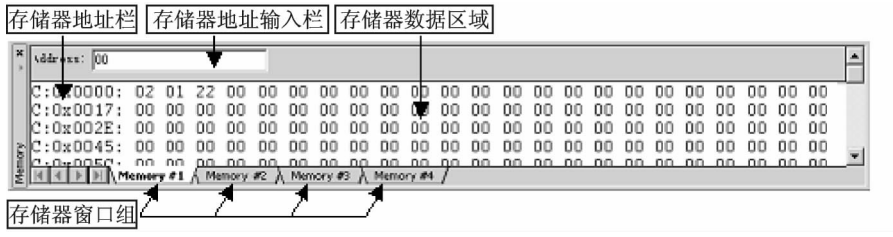


图 6－31　存储器窗口的显示

② 使用命令行：在命令输入窗口通过输入符合格式的命令，可以查看和修改 data 空间的数据。

● 通过命令行查看数据。例如，想查看 data 空间从 0x23～0x34 区间的内容，可以在命令输入窗口输入：d d: 0x23，0x34，按“Enter”键即可完成。第一个 d 表示执行查看数据命令，第二个 d 表示查看的数据类型是 data 空间，0x23 表示起始地址，0x34 表示结束地址，μVision2 把输出结果在信息输出窗口中显示出来，如图 6－33所示。

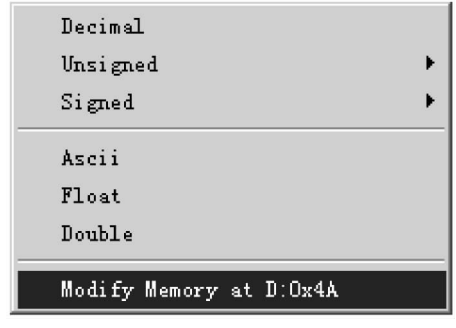


图 6－32　存储器窗口的数据修改

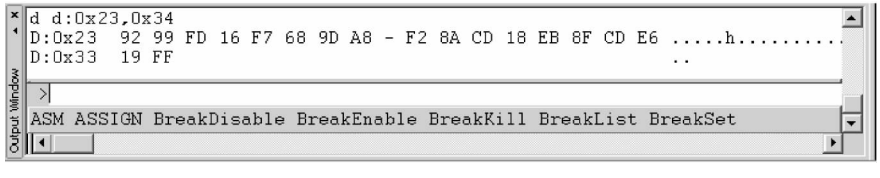


图 6－33　命令窗口的数据显示

● 通过命令行修改数据。例如，想把 d: 0x23 开始的三个数据修改成 0x01，0x02，0x03，可以在命令输入窗口输入：E char d: 0x23 = 0x01，0x02，0x03，按“Enter”键即可完成。E 表示执行的是修改数据命令，char 表示修改数据类型（数据类型在信息窗口中提示，主要有 char，int，long，float，double），d 表示 data 空间，“ = ”后面是输入的修改数据。修改完成后通过命令行查看数据已经进行了修改。

(2) 内部可间接寻址 RAM（类型 idata，简称 i）

在标准 80C51 中，可间接寻址空间为 0～0xFF 范围内的 RAM。其中，地址范围 0x00～0x7F 的 RAM 既可间接寻址，也可直接寻址 地址范围 0x80～0xFF 的 RAM 只能间接寻址。在 μVision2 中把它们组合成空间连续的可间接寻址的 idata 空间。idata 空间有下列几种查看和修改方法。

① 存储器窗口。在存储器窗口中，内部可间接寻址 RAM（idata）的操作方法同内部可直接寻址 RAM（或 SFR）完全相同，只是需要在存储器地址输入栏输入的信息要变为 i。例如，要在存储器窗口显示起始地址为 0x65 的 idata 数据，只要在存储器地址输入栏输入“i：0x65”即可。

② 使用命令行。操作同直接寻址，只是把表示可直接寻址空间的 d 换成表示可间接寻址空间的 i。例如，想查看 idata 空间从 0x23～0x34 区间的内容，可以在命令输入窗口输入：d i：0x23，0x34 按“Enter”键即可完成。

（3）片外数据空间 XRAM（类型 xdata，简称 x）

在标准 80C51 中，片外可间接寻址 64 KB 地址范围的数据存储器，其读取的指令是 MOVX，间接寻址寄存器是 DPTR 或 Ri。在 μVision2 中把它们组合成空间连续的可间接寻址的 xdata 空间。xdata 空间数据的查看和修改也是通过存储器窗口或使用命令行。操作方法同片内可直接寻址 RAM，只是把表示可直接寻址空间的 d，换成表示片外间接寻址数据空间的 x。

（4）程序空间 code（类型 code，简称 c）

在标准 80C51 中，程序空间有 64 KB 的地址范围。程序存储器的数据按用途可分为程序代码（用于程序执行），程序数据（程序使用的固定参数）。在 μVision2 中把它们组合成空间连续的可间接寻址的 code 空间。code 空间数据的查看和修改也是通过存储器窗口或使用命令行。操作方法同片内可直接寻址 RAM，只是把表示可直接寻址空间的 d，换成表示可间接寻址的 code 空间的 c。

例如在示例程序“led. asm”中用 ORG 0100H 开始执行语句 START：MOV　R2，#08H，可在存储器窗口中输入 c 0x0100 显示该行程序编译后的机器码。如果是参数表，还可以直接修改数据表调整参数，当然修改后的数据只会在反汇编窗口体现修改，而不会自动修改源程序。

14. 变量的查看和修改

在一些大型工程中，还可以在汇编源程序中定义一些变量，增加汇编源程序的可读性。例如在示例程序中，可以定义“LEDBUFF”变量代替寄存器 R2，来实现八个 LED 的循环点亮。程序修改如下：

```
/*************************************************/
/* 程序说明 */
/* 依次点亮在并行接口 P1 外部的 LED，并无限循环 */
/*************************************************/
        LEDBUFF     DATA    30H
        ORG         0000H
        LJMP        START
```

```
        ORG     0100H
START:  MOV     LEDBUFF,#08H    设置循环次数
        MOV     A,#0FEH         送显示模式字
NEXT:   MOV     P1,A            点亮连接 P1.0 的 LED
        ACALL   DELAY
        RL      A               左移一位,改变显示模式字
        DJNZ    LEDBUFF,NEXT    循环次数减 1,不为零,继续点亮
                                下一个 LED
        AJMP    START
/* 延时子程序                                          */
DELAY:  MOV     R3,#0FFH
DLY1:   MOV     R4,#0FFH
DLY2:   NOP
        DJNZ    R4,DLY2
        DJNZ    R3,DLY1
        RET
        END
```

在 μVision2 中可以直接观察和修改变量,方法为以下几种:

(1) 使用观察窗口 (Watch Windows)

单击"View"→"Watch & Call Stack Windows"命令可以打开观察窗口,如图 6-34 所示,如果窗口已经打开,该窗口则会关闭。

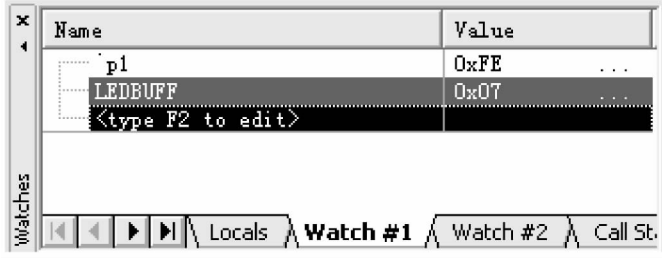


图 6-34　观察 Watchs 窗口的数据显示

Watch1/2:用户可以根据分类把变量添加到#1,或#2 中。

Name:用于输入变量的名称。输入方法是:单击准备添加行 (选择该行) 的"Name"栏,然后按"F2"键,出现文本输入栏后输入变量的名称,确认正确后按"Enter"键。输入的变量名称

必须是文件中已经定义的，图 6－34 中“LEDBUFF”是自己定义的，而“P1”是系统头文件中定义的。

Value：用于显示变量的数值。可以修改变量的数值，方法是：单击该行的“Value”栏，然后按“F2”键，出现文本输入栏后输入修改的数据，确认正确后按“Enter”键。

⑵ 使用命令行

使用命令行可以对变量进行读写。在进入硬件仿真的任何时刻，在命令输入窗口输入“P1”，然后按“Enter”键，在信息输出窗口则会输出从 TKS－52B 仿真器中读回的“P1”数值，例如 0XFE。如果需要把“P1”的数值从 0xFE 修改成 0xFF，则可在命令输入窗口中输入 P1 = 0xFF，按“Enter”键即可。

15. 在 Keil 调试中的使用技巧

⑴ 使用外设菜单“Peripherals”增加调试效率

在进入硬件仿真环境后，根据为工程选择的器件型号，在菜单“Peripherals”中会有一些该型号内部的外设名，单击一个外设名后将会弹出一个浮动窗口，外设的大部分资源在它的外设窗口中都可以直观显示并能进行修改。

⑵ 使用“Show Next Statement”找到当前光标

在调试过程中，可能在查看中发现当前的程序运行指针不在我们的视野中，也可能在另外的一个窗口中，当然可以一点一点的寻找，但有时候这样做非常的麻烦，尤其在工程比较庞大、文件比较多的时候，可以使用“Show Next Statement”快速找到当前的程序运行指针。

① 单击菜单“Debug”→“Show Next Statement”命令；

② 单击工具条上的图标 ⇨。

在完成上述的任何一条操作后，可以看到程序指针所在的汇编和源程序窗口变为最上面的两个窗口，并且程序指针指在了窗口内的位置。

⑶ 将变量快速加到观察窗口中

前面已经介绍了在观察窗口中通过输入变量名可以在“Watch”窗口观察或修改该变量，但是如果变量名比较长或要添加的变量比较多，这样操作也比较烦琐。通过下面提供的操作，可以直观快速地将一个变量添加到“Watch”窗口中。

在硬件仿真环境中，右击变量名，出现浮动菜单，如果变量是有效的，在菜单中将会有“Add ‘变量名’ to Watch Windows”的选项，子层还会有#1，或#2 的选项，单击后该变量就会加到对应的 Watch 窗口去，如图 6－35 所示。

⑷ 在调试中改变程序的汇编代码

在调试已经编译好的程序时，可以在硬件仿真环境中直接修改程序代码，这样就不用返回到编译环境中重新编译。μVision2 支持在调试中的在线汇编。

在汇编窗口中，如果想改动一条汇编指令，单击该行，然后右击，出现浮动菜单后选择“Inline

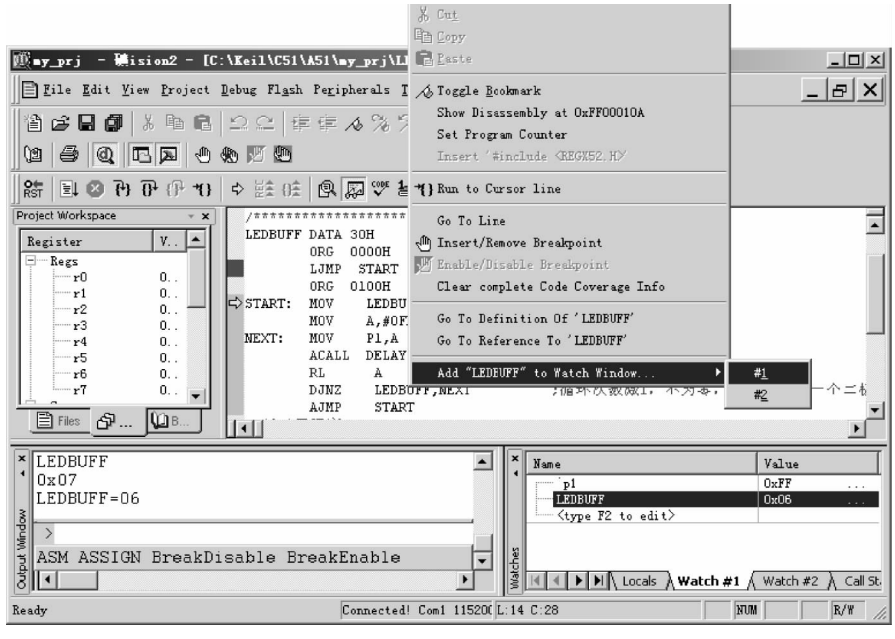


图 6－35　添加变量到 watch 窗口

Assembly”选项，如图 6－36 所示。在弹出输入窗口中输入汇编指令，然后按“Enter”键即可。

16. 退出仿真

第一步退出 μVision2 仿真环境。

如果想退出 μVision2 的硬件仿真环境，可以使用下列方法：

① **单击图标**；

② **单击菜单**“Debug”→“Start/Stop Debug Session”**选项；**

③ **在命令输入窗口输入** Exit。

第二步关闭 TKS－52B **仿真器。**

如果想停止使用 TKS **仿真器，按照下列顺序操作：**

① **关闭用户目标板的电源；**

② **拔下** TKS **仿真器的电源；**

③ **从用户目标板上拔下仿真头；**

④ **如果暂时不再进行硬件仿真，卸下** TKS **同计算机的通信电缆；**

⑤ **将仿真器主机、仿真电缆、仿真头和电源放到适当的地方保存。**

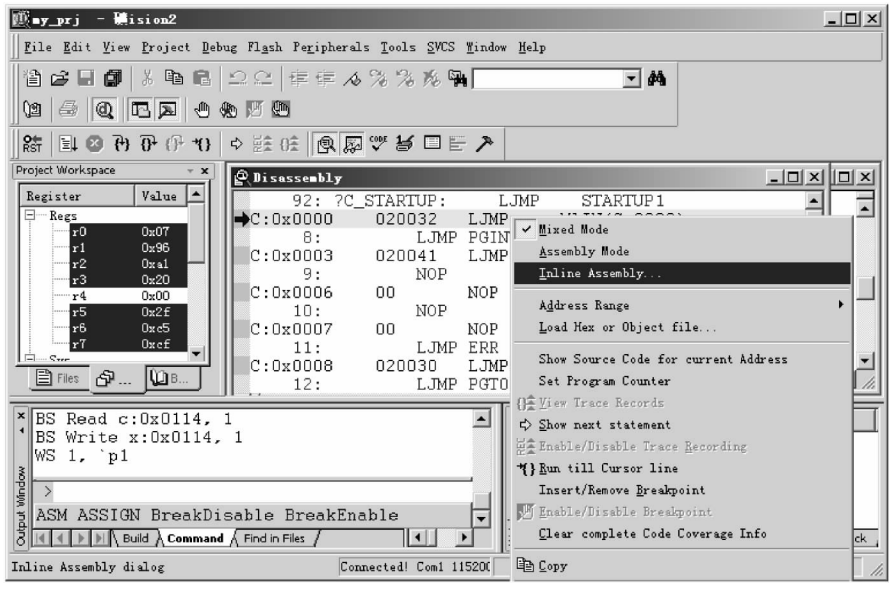
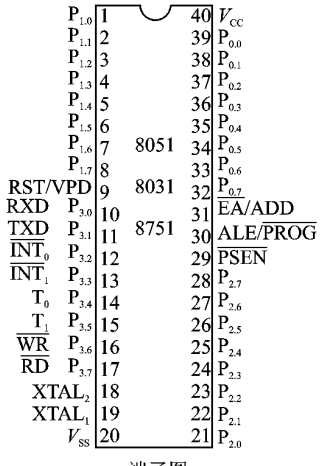

File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Register
Value
Regs
r0 0x07
r1 0x96
r2 0xa1
r3 0x20
r4 0x00
r5 0x2f
r6 0xc5
r7 0xcf
Files
Disassembly
92: ?C_STARTUP: LJMP STARTUP1
C:0x0000 020032 LJMP
8: LJMP PGIN
C:0x0003 020041 LJMP
9: NOP
C:0x0006 00 NOP
10: NOP
C:0x0007 00 NOP
11: LJMP ERR
C:0x0008 020030 LJMP
12: LJMP PGT0
Mixed Mode
Assembly Mode
Inline Assembly...
Address Range
Load Hex or Object file...
Show Source Code for current Address
Set Program Counter
View Trace Records
Show next statement
Enable/Disable Trace Recording
Run till Cursor line
Insert/Remove Breakpoint
Enable/Disable Breakpoint
Clear complete Code Coverage Info
Copy
BS Read c:0x0114, 1
BS Write x:0x0114, 1
WS 1, `p1
ASM ASSIGN BreakDisable BreakEnable
Output Window
Build
Command
Find in Files
Inline Assembly dialog
Connected! Com1 11520(

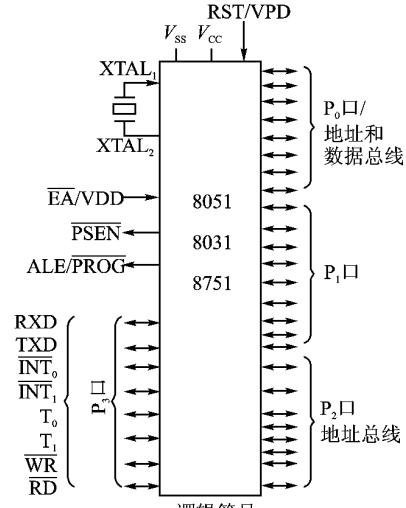
图 6－36　在线汇编界面

附录　常用 IC 电路端子图

1．8031 单片机

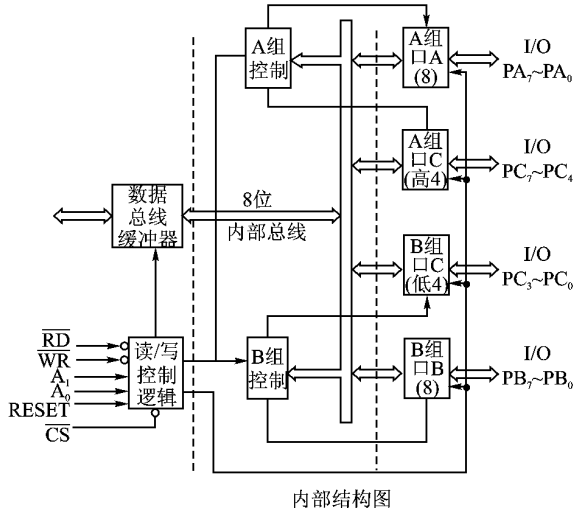


端子图

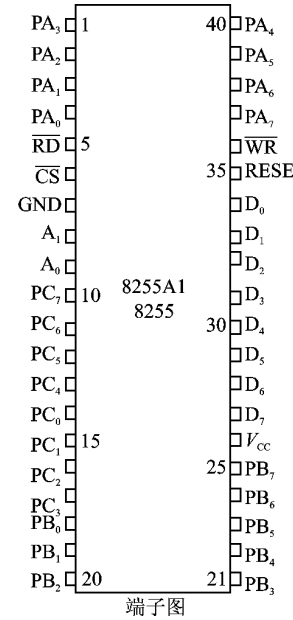


逻辑符号

2．8255 八位并行 I/O 口

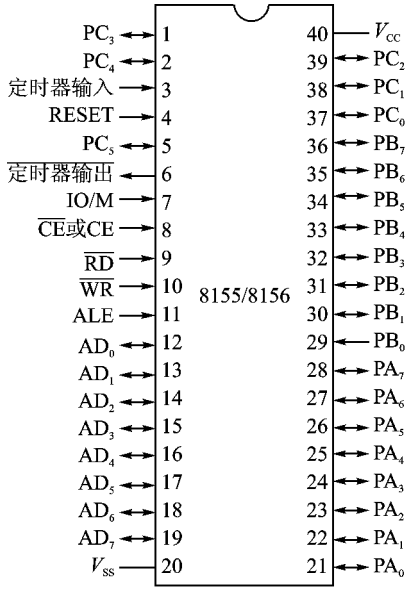


内部结构图

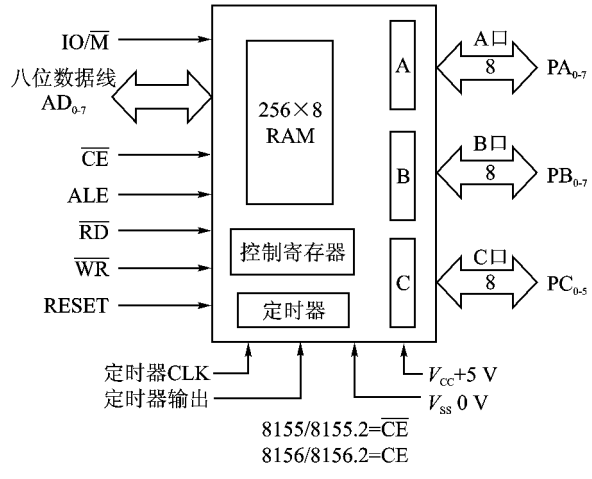


端子图

3. 8155 RAM/IO/CTC 口

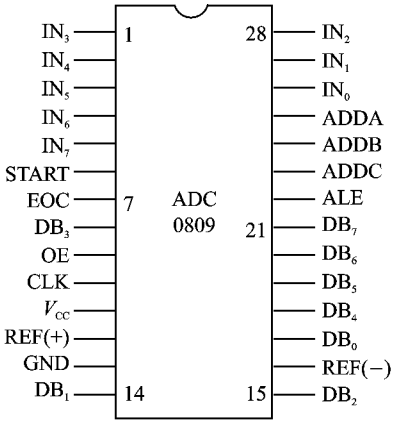


端子排列

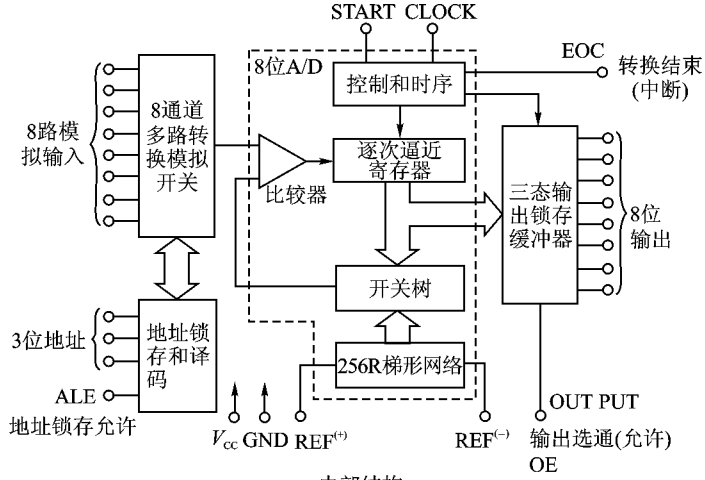


结构框图

4. ADC0809 八位模数转换器

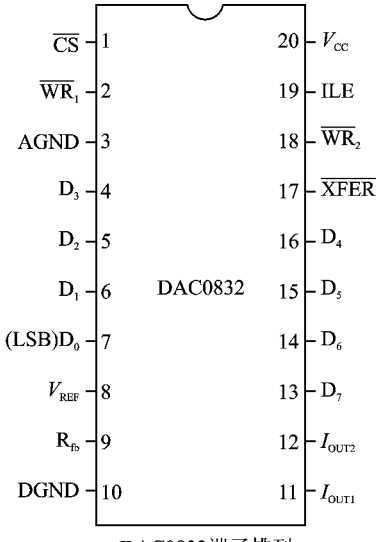


端子排列

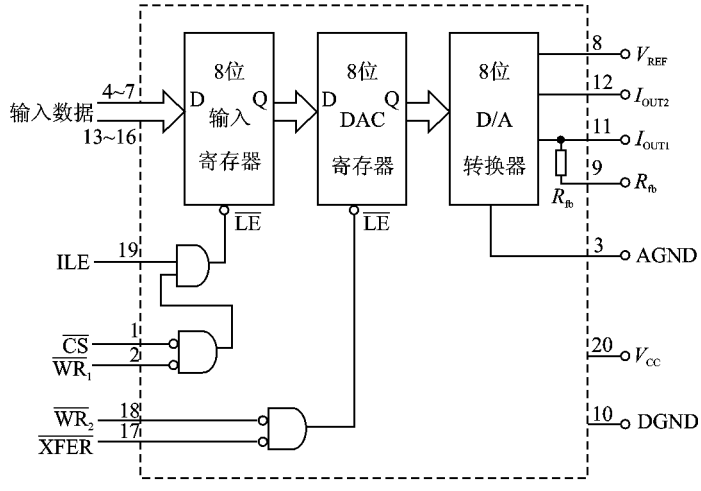


内部结构

5．DAC0832 八位数模转换器

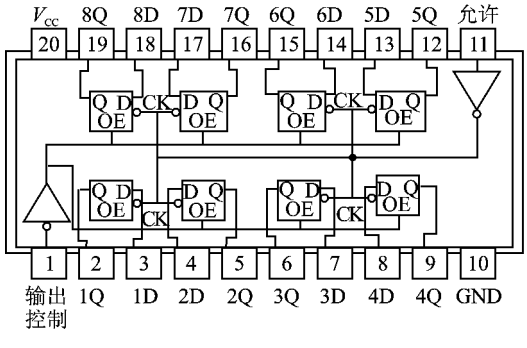


DAC0832端子排列



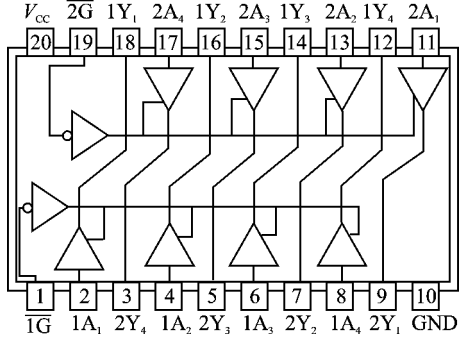
DAC0832框图

6．74LS373 八 D 锁存器（三态输出）

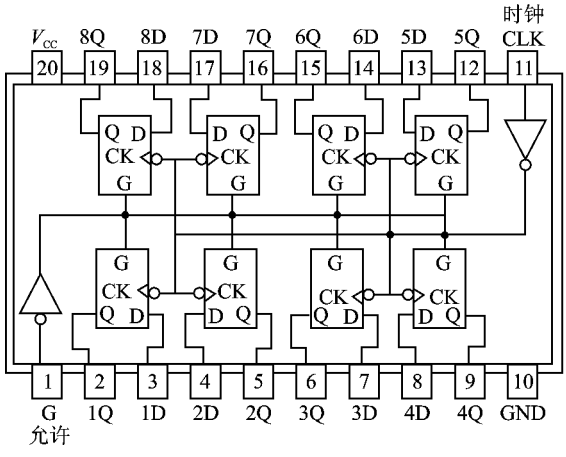


输出控制	允许G	D	输出
L	H	H	H
L	H	L	L
L	L	×	Q_0
H	×	×	Z

7．74LS244 八缓冲器/线驱动器/线接收器（不反相三态输出）



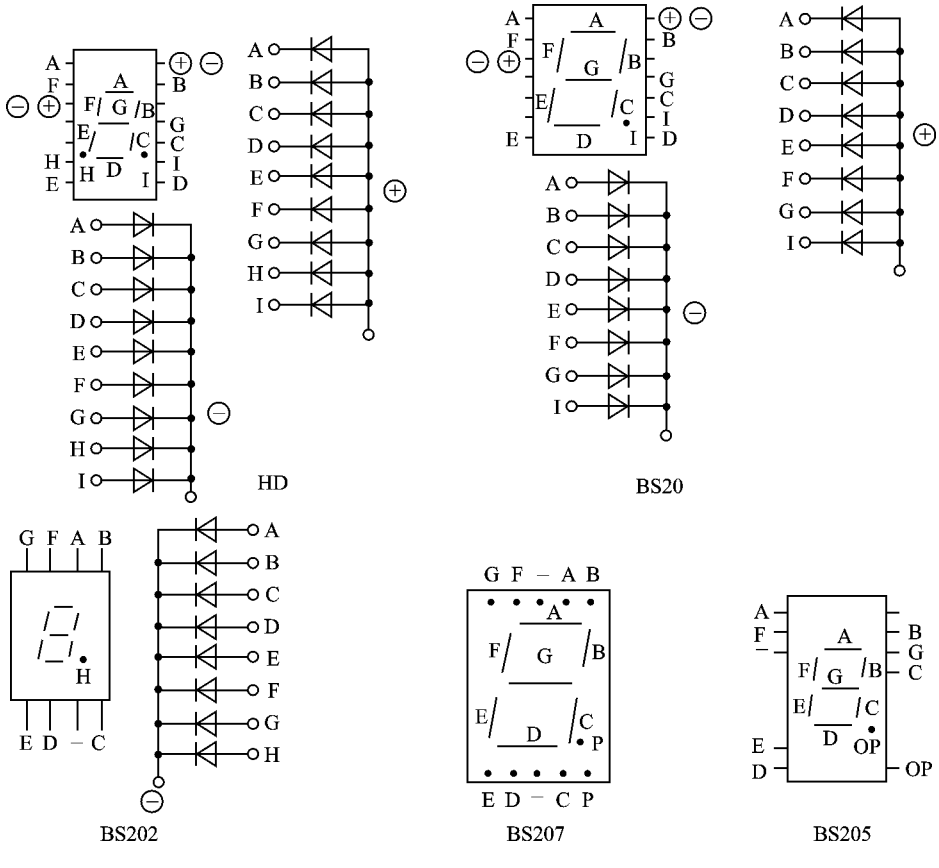
8．74LS377 八 D 触发器



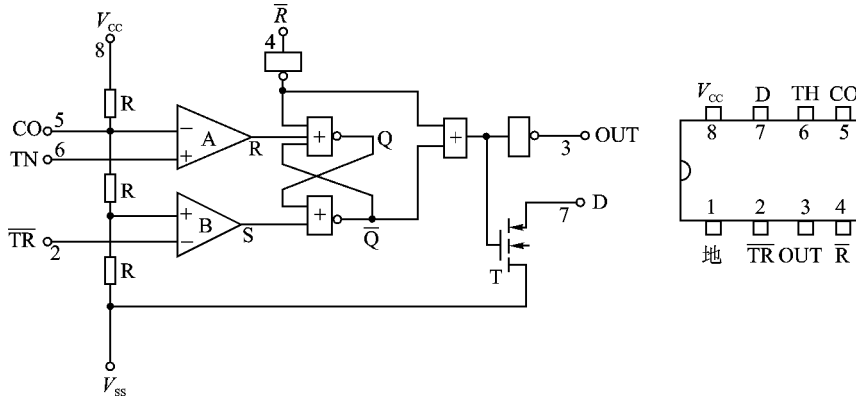
功能表(每个触发器)

输入			输出	
$\overline{G}$	时钟	数据	Q	$\overline{Q}$
H	×	×	Q_0	$\overline{Q}_0$
L	I	H	H	L
L	I	L	L	H
×	L	×	Q_0	$\overline{Q}_0$

9．HD、BS20、BS202、BS207、BS205 端子图



10．5G555

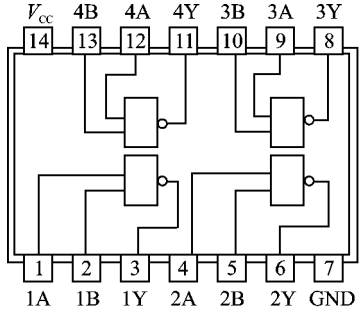


功能表

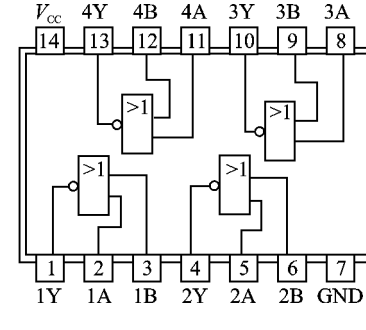
符号	功能	符号	功能
$\overline{TR}$	低触发端	TH	高触发端
OUT	输　出	D	放电端
$\overline{R}$	复　位	CO	控制电压

TH	$\overline{TR}$	$\overline{R}$	OUT	D
×	×	L	L	导通
$>2E_{C/3}$	$>E_{C/3}$	H	L	导通
$<2E_{C/3}$	$>E_{C/3}$	H	不变	不变
×	$<E_{C/3}$	H	H	截止

11．74LS00 和 74LS02

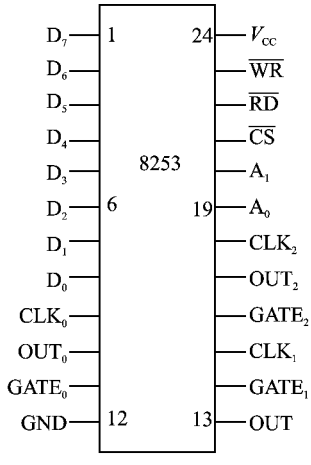


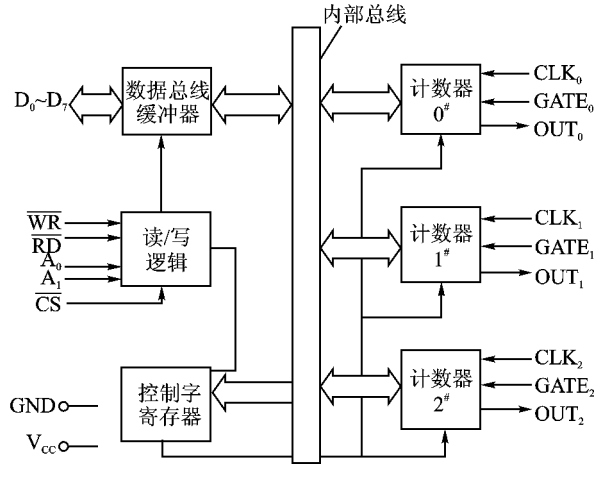
74LS00四2输入正与非门



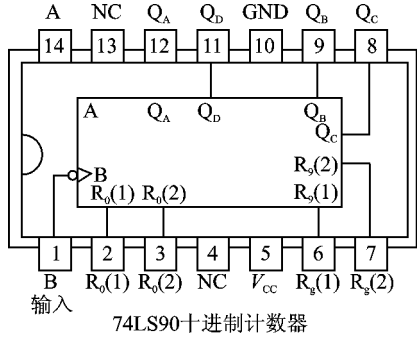
74LS02四2输入正或非门

12．8253 定时器





13．74LS90 二-五-十进制计数器



74LS90十进制计数器

复位　计数功能表

复位输入				输出			
R_0 (1)	R_0 (2)	R_g (1)	R_g (2)	Q_A	Q_B	Q_C	Q_D
1	1	0	×	0	0	0	0
1	1	×	0	0	0	0	0
×	×	1	1	1	0	0	1
×	0	×	0	计数			
0	×	0	×	计数			
0	×	×	0	计数			
×	0	0	×	计数			

BCD 计数时序 (注 A)

计数	输出			
	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

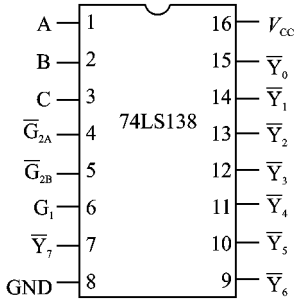
注：A 输出 Q_A 与输入 B 相接作 BCD 计数

二～五混合进制 (注 B)

计数	输出			
	Q_A	Q_B	Q_C	Q_D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

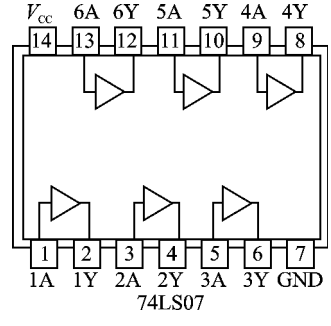
注：B 输出 Q_D 与输入 A 相接作 2～5 混合进制计数。

14．74LS138 三-八线译码器



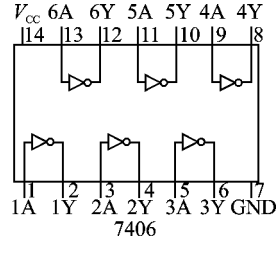
使　能		输　入			输　出							
G_1	$G_{2A}+G_{2B}$	C	B	A	$\overline{Y}_0$	$\overline{Y}_1$	$\overline{Y}_2$	$\overline{Y}_3$	$\overline{Y}_4$	$\overline{Y}_5$	$\overline{Y}_6$	$\overline{Y}_7$
×	H	×	×	×	H	H	H	H	H	H	H	H
L	×	×	×	×	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

15．74LS07 六反相缓冲器/驱动器（OC、高压输出）



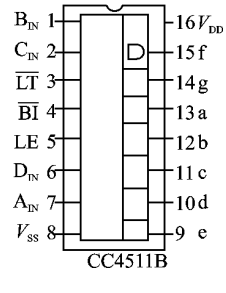
74LS07

16．7406 集电极开路高压输出的六反相缓冲器/驱动器



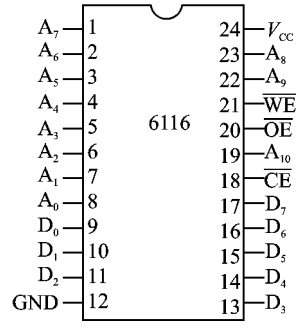
7406

17．CC74HC/HCT4511、CC4511B（BCD-7 段锁存/译码/驱动器）

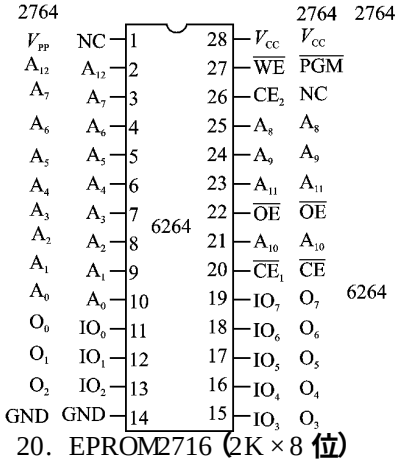


CC4511B

18．静态 RAM6116（2K×8 位）



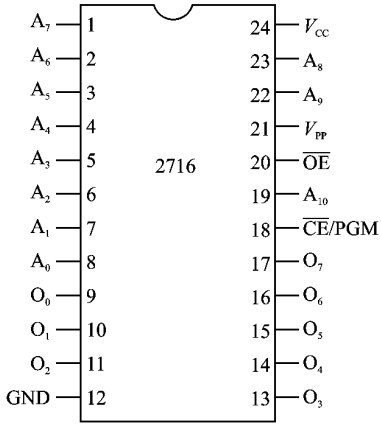
19．EPROM2764（8K×8 位）与 RAM6264（8K×8 位）



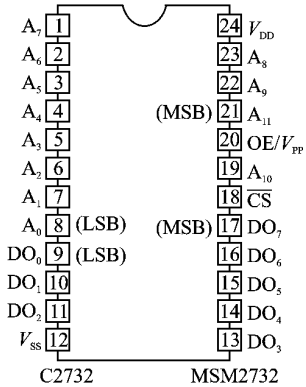
工作方式	$\overline{CE}$ (片选)	$\overline{OE}$ (输出允许)	V_{PP}	$\overline{PGM}$ (编程控制信号)	输出
读	L	L	V_{CC}	H	数据输出
维　持	H	×	V_{CC}	×	高　阻
编　程	L	H	V_{PP}	L	数据输入
编程校验	L	L	V_{PP}	H	数据输出
编程禁止	H	×	V_{PP}	×	高　阻

工作方式	$\overline{CE_1}$ (片选一)	CE_2 (片选二)	$\overline{OE}$ (输出允许)	$\overline{WE}$ (写允许)	IO_0~IO_7 (输入/输出)
未选中	H	×	×	×	高　阻
未选中	×	L	×	×	高　阻
输出禁止	L	H	H	H	高　阻
读	L	H	L	H	数据输出
写	L	H	H	L	数据输入
写	L	H	L	L	数据输入

20．EPROM2716（2K×8 位）

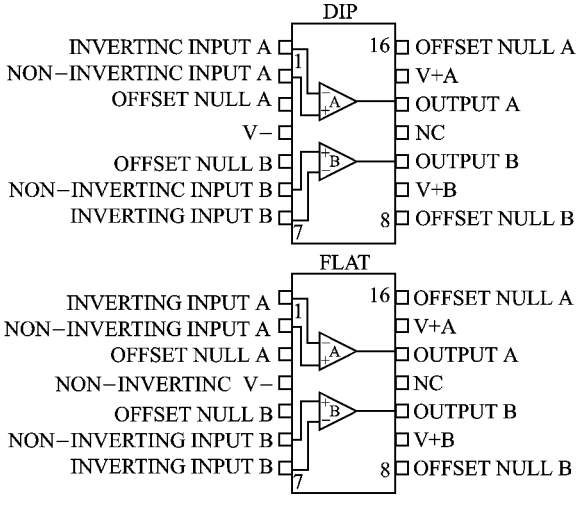


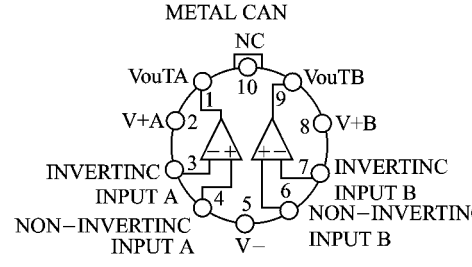
2716端子图



C2732　　MSM2732
M5L2732　　MBM27C32
MB8532　　μPD2732
TMM2732　　MBM2732A
HN462732　　Am2732
NMC2732　　F2732
NMC27C32A　　RD27C32

21. 双运放 μA747 等

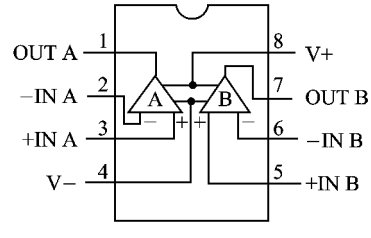


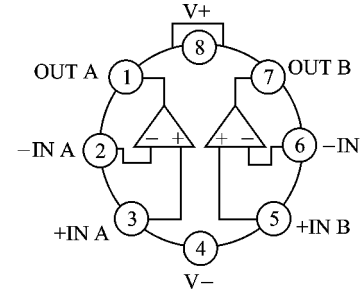


LM747　MC1747　μA747　μA747C
LM747C　MC1747C　μA747A　μA747E

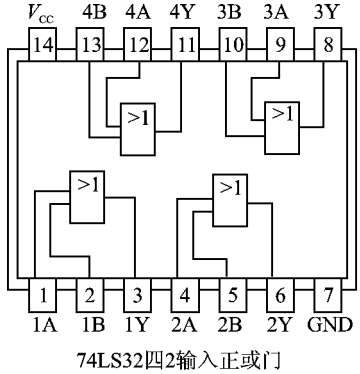
22. 双运放 RC4558 等

RC4558
TL072
LM358
NE5532
5G253　双高阻抗运算放大器
5G022　低功耗双运放

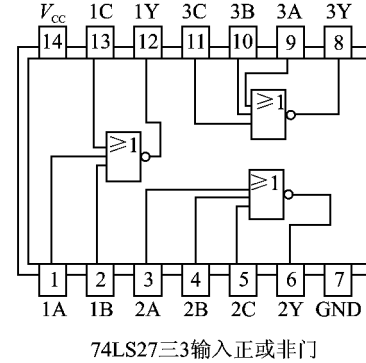




23. 74LS32 和 74LS27

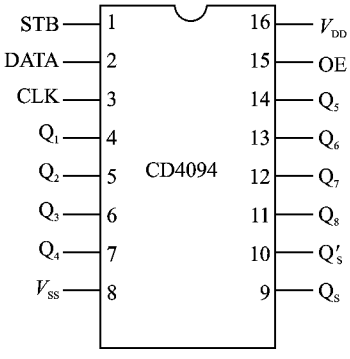


74LS32四2输入正或门



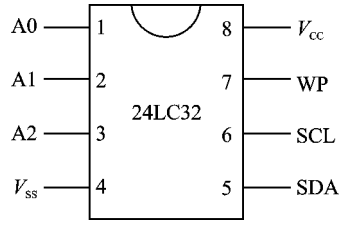
74LS27三3输入正或非门

24. CD4094 八位串行输入/并行输出移位寄存器

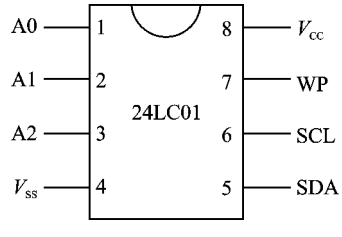


CLK	OE	STB	D	并行输入 Q_1	并行输入 Q_N	串行输出 Q_S	串行输出 Q'_S
↑	0	×	×	高阻	高阻	Q_7	不变
↓	0	×	×	高阻	高阻	不变	Q_1
↑	1	0	×	不变	不变	Q_7	不变
↑	1	1	0	0	Q_{N-1}	Q_7	不变
↑	1	1	1	1	Q_{N-1}	Q_7	不变
↓	1	1	1	不变	不变	不变	Q_7

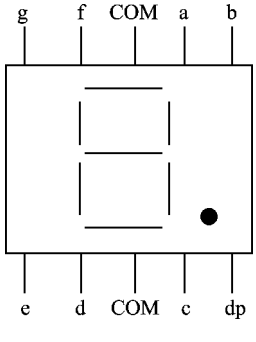
25. 1～16 KB 位串行 EEPROM



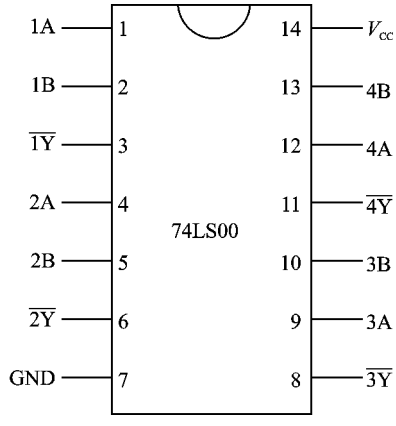
26. 32～64 KB 位串行 EEPROM



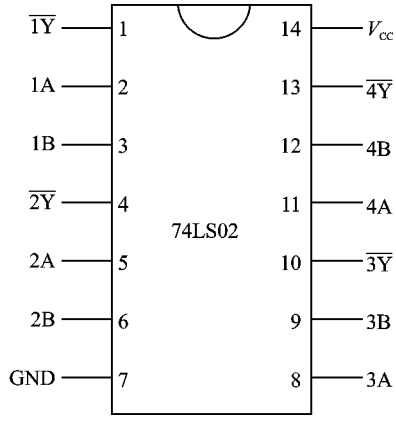
27. FND500LED



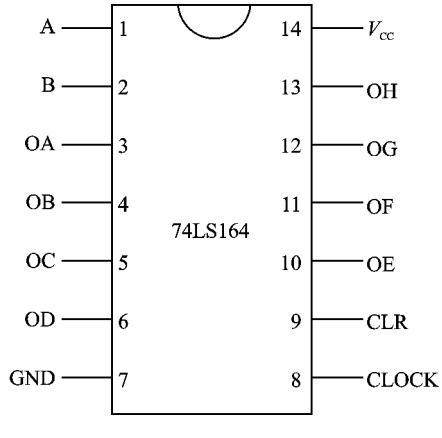
28．2 输入-4 与非门



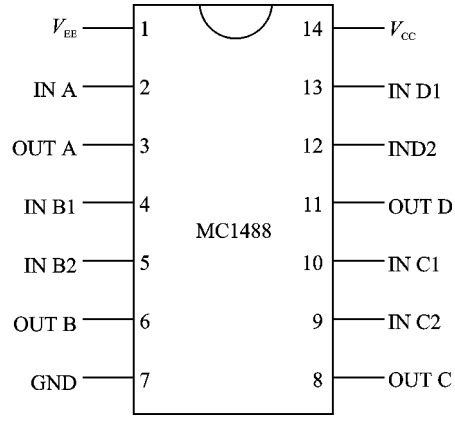
29．2 输入-4 或非门



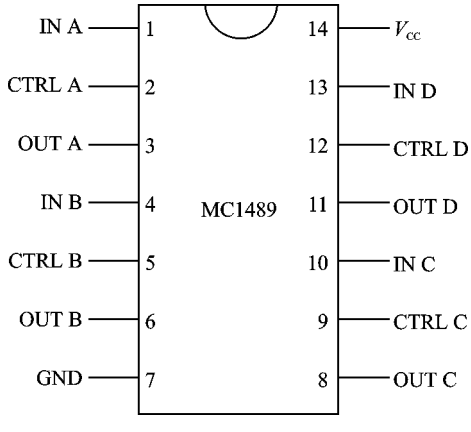
30．8 位串入并出移位寄存器



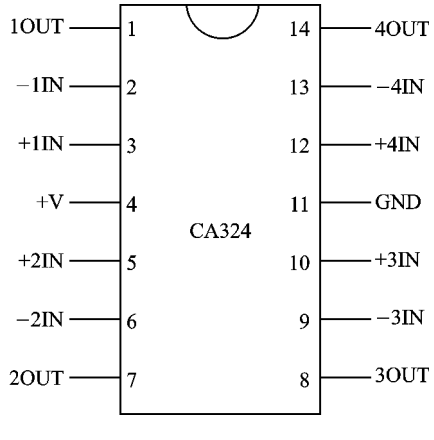
31．TTL-RS232 **电平转换器**



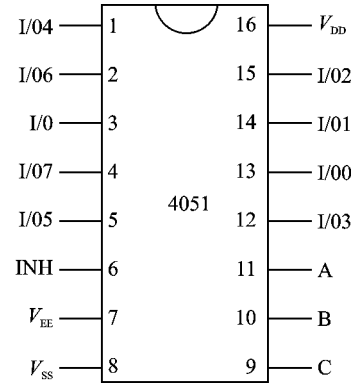
32．RS232-TTL **电平转换器**



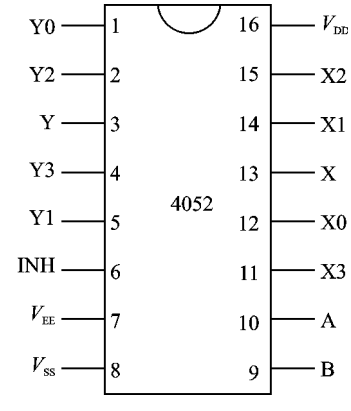
33．**通用** 4 **运放**



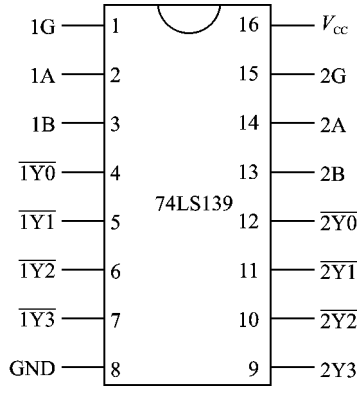
34. 8 路模拟开关



35. 双 4 选 1 模拟开关



36. 双 2-4 译码器



参考文献

[1] 金龙国. 单片机原理与应用 [M]. 北京 :中国水利水电出版社,2005.

[2] 刘守义. 单片机应用技术 [M]. 西安 :西安电子科技大学,2002.

[3] 王恩荣. MCS-51 单片应用技术实训指导 [M]. 北京 :化学工业出版社,2001.

[4] 张国勋. 单片机原理及应用 [M]. 北京 :中国电力出版社,2004.

[5] 徐淑华. 单片微型计算机原理及应用 [M]. 哈尔滨 :哈尔滨工业大学出版社,1991.

[6] 吴国经. 单片机应用技术 [M]. 北京 :中国电力出版社,2004.

[7] 李广第. 单片机基础 [M]. 北京 :北京航空航天大学出版社,1995.

[8] 张迎辉. 单片机实训教材 [M]. 北京 :北京大学出版社,2005.