



普通高等教育“十五”国家级规划教材

单片机原理及应用

DANPIANJI YUANLI JI YINGYONG

张毅刚 主编

彭喜元 董继成 副主编



高等教育出版社

HIGHER EDUCATION PRESS

高等学校 自动化 专业系列教材 电气工程及其自动化

- | | |
|------------------------|-----------|
| * 《单片机原理及应用》 | 张毅刚主编 |
| * 《现代检测技术》 | 周吉鹏等编著 |
| 《电气工程概论》 | 范瑜主编 |
| 《自动控制原理》 | 程鹏主编 |
| 《自动控制原理》(非自动化类) | 孟庆明主编 |
| 《自动控制原理》(上、下册) | 黄家英 |
| 《现代控制理论》 | 钟秋海主编 |
| 《控制理论与机械系统控制》 | 竺长安、张屹主编 |
| * 《电力电子学——电力电子变换和控制技术》 | 陈坚主编 |
| 《现代电力电子技术基础》 | 张立主编 |
| 《电机与拖动》 | 唐介主编 |
| * 《电机与拖动基础》 | 邱阿瑞主编 |
| 《传感器与检测技术》 | 陈杰、黄鸿编著 |
| 《传感器与检测技术》CAI | 陈杰、黄鸿主编 |
| 《现代检测技术与系统》 | 蔡萍、赵辉主编 |
| 《机电工程智能检测技术与系统》 | 朱名铨等编 |
| 《单片机原理及应用》(英文版) | 韩建国、廖俊必主编 |
| 《交流调速》 | 王君艳编著 |
| 《计算机控制系统》 | 高金源主编 |
| 《智能信息处理技术》 | 王耀南主编 |
| 《信号处理与软计算》 | 史习智主编 |
| 《信号与系统分析》 | 赵录怀等编 |
| 《微型计算机原理与接口技术》(配光盘) | 尹建华等编 |

注：*号为普通高等教育“十五”国家级规划教材

ISBN 7-04-013036-X



9 787040 130362 >

定价：34.10元

普通高等教育“十五”国家级规划教材

单片机原理及应用

张毅刚 主编

彭喜元 董继成 副主编

高等教育出版社

内容简介

本书详细地介绍了 MCS-51 单片机的硬件结构、指令系统,从应用的角度介绍了汇编语言程序设计与各种硬件接口设计、各种常用的数据运算和处理程序、接口驱动程序以及 MCS-51 单片机应用系统的设计,并对 MCS-51 单片机应用系统设计中的抗干扰技术以及各种新器件也作了详细的介绍。本书突出了选取内容的实用性、典型性,书中的应用实例,大多来自科研工作及教学实践,且经过检验,内容丰富、详实。

本书可作为工科院校的专科生、本科生、研究生单片机课程的教材以及毕业设计工作的参考书,也可供从事自动控制、智能仪器仪表、电力电子、机电一体化以及各类 MCS-51 单片机应用的工程技术人员参考。

图书在版编目(CIP)数据

单片机原理及应用/张毅刚主编. —北京:

高等教育出版社,2004 重印

ISBN 7-04-013036-X

I. 单... II. 张... III. 单片微型计算机
-高等学校-教材 IV. TP368.1

中国版本图书馆 CIP 数据核字(2003)第 106732 号

出版发行	高等教育出版社	购书热线	010-64054588
社 址	北京市西城区德外大街 4 号	免费咨询	800-810-0598
邮政编码	100011	网 址	http://www.hep.edu.cn
总 机	010-82028899		http://www.hep.com.cn
经 销	新华书店北京发行所		
印 刷	北京铭成印刷有限公司		
开 本	787×960 1/16	版 次	2004 年 1 月第 1 版
印 张	27.5	印 次	2004 年 7 月第 2 次印刷
字 数	510 000	定 价	34.10 元

本书如有缺页、倒页、脱页等质量问题,请到所购图书销售部门联系调换。

版权所有 侵权必究

前 言

本书为工科电气工程信息类本科全国统编教材。

单片微计算机自 20 世纪 70 年代问世以来,已对人类社会产生了巨大的影响。尤其是美国 Intel 公司生产的 MCS-51 系列单片机,由于其具有集成度高、处理功能强、可靠性高、系统结构简单、价格低廉、易于使用等优点,在世界范围内已经得到广泛的普及和应用。此外,世界各大公司以 MCS-51 单片机基本内核为核心的各种扩展型、增强型的新型单片机不断推出,所以在今后若干年内, MCS-51 系列以及世界其它各大公司生产的与其兼容的各种增强型、扩展型的单片机,仍是我国单片机应用领域的主流机型。目前在工业控制、智能仪器仪表、办公室自动化、家用电器等诸多领域,到处都可看见单片机的踪影,单片机技术开发和应用水平已成为一个国家工业化发展水平的标志之一。

单片机课程是一门应用设计类课程,所以本书在编写时,重点考虑了如下问题:

1. 避免仅仅从原理上去对 MCS-51 单片机进行分析和介绍,而是在对 MCS-51 的硬件结构和指令系统详细介绍的基础上,详细的介绍了应用系统的设计,使得本教材具有较强的应用性。本书不仅详细介绍了各种硬件接口的设计,而且对如何组成硬件系统也加以详细的介绍,并给出实例,使得学生能很快地掌握常用的应用系统设计。

2. 突出了选取内容的实用性、典型性。书中的应用实例,大多来自科研工作及教学实践,内容丰富、详实。所介绍的各种设计方案,均为常用、典型的方案。本书提供了大量的接口设计实例及程序实例,非常有利于学生提高设计工作的效率。

3. 对系统设计用到的新器件也做了详细的介绍。例如各种新型的与 MCS-51 兼容的单片机芯片、存储器芯片、时钟日历芯片、新型功率器件以及其它新型芯片等。

4. 本书是多年教学、科研工作的结晶,文字精练,通俗易懂,深入浅出,便于自学。书中各章后均附有思考题与习题,供学生巩固、消化、理解课堂所学内容之用。

本书首先详细的介绍了 MCS-51 单片机的硬件结构和指令系统,在此基础上详细介绍了 MCS-51 单片机的各类接口技术及应用系统设计。

全书共分为 15 章,第 1 章至第 7 章详细的介绍了 MCS-51 单片机的硬件结构、指令系统及片内各功能部件。第 8 章至第 13 章介绍各种类型的硬件接口及软件设计,如存储器,I/O 接口,键盘、显示器、微型打印机,A/D、D/A 转换,大功率(高压、大电流)芯片以及各种在单片机应用设计中用到的其它接口和电路等,并对各种接口的驱动程序也加以介绍。鉴于抗干扰技术和可靠性在单片机应用系统设计中的重要性,第 14 章详细地介绍了目前常用的各种抗干扰技术和抗干扰设计。第 15 章介绍了如何根据应用需求,来进行应用系统的设计,如何使用仿真开发系统来进行单片机应用系统的开发和调试。

全书的参考学时为 40~60 学时。教师可根据实际情况,对各章所讲授的内容进行取舍。

本书由哈尔滨工业大学电气工程及自动化学院张毅刚担任主编,并完成了第 1 至第 10 章的主要编写工作和全书的统稿工作,彭喜元、董继成担任副主编,彭喜元完成了第 11、12 章的编写工作,董继成完成了第 13 章至第 15 章的编写工作。此外,参加本书编写工作的还有黄灿杰、马云彤、刘兆庆、孟升卫、刘旺。哈尔滨工业大学自动化测试与控制研究所硕士生卢艳东、杨海涛、贺建林在本书的插图工作中付出了辛勤的劳动。哈尔滨理工大学李全利教授审阅了全书并提出了宝贵的修改意见。在此,对他们一并表示衷心感谢。

由于时间紧迫,书中错误及疏漏之处在所难免,敬请读者批评指正。

作者 2003 年 8 月于哈尔滨工业大学

目 录

第 1 章 单片机概述	1
1.1 什么是单片机	1
1.2 单片机的历史及发展概况	2
1.3 8 位单片机的主要生产厂商和机型	3
1.4 单片机的发展趋势	4
1.5 单片机的应用	5
1.6 MCS-51 系列单片机	7
思考题及习题	9
第 2 章 MCS-51 单片机的硬件结构	10
2.1 MCS-51 单片机的硬件结构	10
2.2 MCS-51 的引脚	12
2.2.1 电源及时钟引脚	13
2.2.2 控制引脚	14
2.2.3 I/O 口引脚	15
2.3 MCS-51 的 CPU	15
2.3.1 运算器	15
2.3.2 控制器	17
2.4 MCS-51 存储器的结构	18
2.4.1 程序存储器	19
2.4.2 内部数据存储器	20
2.4.3 特殊功能寄存器(SFR)	21
2.4.4 位地址空间	24
2.4.5 外部数据存储器	25
2.5 并行 I/O 端口	26
2.5.1 P0 端口	26
2.5.2 P1 端口	28
2.5.3 P2 端口	28
2.5.4 P3 端口	29

2.5.5 P0~P3 端口功能总结	30
2.6 时钟电路与时序	31
2.6.1 时钟电路	31
2.6.2 机器周期、指令周期与指令时序	32
2.7 复位操作和复位电路	34
2.7.1 复位操作	34
2.7.2 复位电路	35
思考题及习题	36
第3章 MCS-51 的指令系统	39
3.1 指令系统概述	39
3.2 指令格式	39
3.3 指令系统的寻址方式	40
3.4 MCS-51 指令系统分类介绍	44
3.4.1 数据传送类指令	44
3.4.2 算术操作类指令	48
3.4.3 逻辑运算指令	53
3.4.4 控制转移类指令	55
3.4.5 位操作指令	59
思考题及习题	64
第4章 MCS-51 汇编语言程序设计	67
4.1 汇编语言程序设计概述	67
4.1.1 机器语言、汇编语言和高级语言	67
4.1.2 汇编语言语句的种类和格式	69
4.1.3 伪指令	71
4.1.4 汇编语言程序设计步骤	73
4.2 汇编语言源程序的汇编	74
4.2.1 手工汇编	74
4.2.2 机器汇编	74
4.3 汇编语言实用程序设计	75
4.3.1 汇编语言程序的基本结构形式	75
4.3.2 子程序的设计	76
4.3.3 查表程序设计	79
4.3.4 关键字查找程序设计	82
4.3.5 数据极值查找程序设计	83
4.3.6 数据排序程序设计	84
4.3.7 分支转移程序设计	87
4.3.8 循环程序设计	91

4.3.9 码制转换程序设计	94
思考题及习题	98
第5章 MCS-51 的中断系统	100
5.1 中断的概念	100
5.2 MCS-51 中断系统的结构	101
5.3 中断请求源	102
5.4 中断控制	103
5.4.1 中断允许寄存器 IE	103
5.4.2 中断优先级寄存器 IP	105
5.5 响应中断请求的条件	107
5.6 外部中断的响应时间	108
5.7 外部中断的触发方式选择	109
5.7.1 电平触发方式	109
5.7.2 跳沿触发方式	109
5.8 中断请求的撤消	110
5.9 中断服务程序的设计	111
5.10 多外部中断源系统设计	114
5.10.1 定时器/计数器作为外部中断源的使用方法	114
5.10.2 中断和查询结合的方法	115
思考题及习题	116
第6章 MCS-51 的定时器/计数器	118
6.1 定时器/计数器的结构	118
6.1.1 工作方式控制寄存器 TMOD	119
6.1.2 定时器/计数器控制寄存器 TCON	120
6.2 定时器/计数器的 4 种工作方式	120
6.2.1 方式 0	120
6.2.2 方式 1	121
6.2.3 方式 2	122
6.2.4 方式 3	122
6.3 定时器/计数器对输入信号的要求	124
6.4 定时器/计数器的编程和应用	125
6.4.1 方式 1 的应用	125
6.4.2 方式 2 的应用	128
6.4.3 方式 3 的应用	130
6.4.4 门控制位 GATE 的应用 —— 测量脉冲宽度	132
6.4.5 实时时钟的设计	133

6.4.6 运行中读定时器/计数器	135
思考题及习题	136
第7章 MCS-51 的串行口	137
7.1 串行口的结构	137
7.1.1 串行口控制寄存器 SCON	138
7.1.2 特殊功能寄存器 PCON	139
7.2 串行口的4种工作方式	140
7.2.1 方式0	140
7.2.2 方式1	141
7.2.3 方式2	143
7.2.4 方式3	145
7.3 多机通信	145
7.4 波特率的制定方法	147
7.4.1 波特率的定义	147
7.4.2 定时器 T1 产生波特率的计算	147
7.5 串行口的编程和应用	149
7.5.1 串行口方式1应用编程(双机通信)	149
7.5.2 串行口方式2应用编程	154
7.5.3 串行口方式3应用编程(双机通信)	154
思考题及习题	157
第8章 MCS-51 单片机扩展存储器的设计	159
8.1 概述	159
8.2 系统总线及总线构造	160
8.2.1 系统总线	160
8.2.2 构造系统总线	161
8.2.3 单片机系统的串行扩展技术	162
8.3 读写控制、地址空间分配和外部地址锁存器	163
8.3.1 存储器扩展的读写控制	163
8.3.2 存储器地址空间分配	163
8.3.3 外部地址锁存器	170
8.4 程序存储器 EPROM 的扩展	172
8.4.1 常用 EPROM 芯片介绍	173
8.4.2 程序存储器的操作时序	176
8.4.3 典型的 EPROM 接口电路	178
8.5 静态数据存储器的扩展	180
8.5.1 常用的静态 RAM(SRAM) 芯片	180
8.5.2 外扩数据存储器的读写操作时序	182

8.5.3 典型的外扩数据存储器的接口电路	182
8.6 EPROM 和 RAM 的综合扩展	186
8.6.1 综合扩展的硬件接口电路	186
8.6.2 外扩存储器电路的工作原理及软件设计	188
8.7 E ² PROM 的扩展	190
8.7.1 常用的 E ² PROM 芯片	190
8.7.2 E ² PROM 的工作方式	192
8.7.3 MCS-51 扩展 E ² PROM 的方法	194
8.8 ATME1 89C51/89C55 单片机的片内闪烁存储器	196
8.8.1 89C51 的性能及片内闪烁存储器	197
8.8.2 片内闪烁存储器的编程	198
思考题及习题	198
第 9 章 MCS-51 扩展 I/O 接口的设计	200
9.1 I/O 接口扩展概述	200
9.1.1 I/O 接口的功能	200
9.1.2 I/O 端口的编址	201
9.1.3 I/O 数据的几种传送方式	201
9.1.4 I/O 接口电路	202
9.2 MCS-51 与可编程并行 I/O 芯片 8255A 的接口设计	202
9.2.1 8255A 芯片介绍	202
9.2.2 工作方式选择控制字及 C 口置位/复位控制字	205
9.2.3 8255A 的 3 种工作方式	206
9.2.4 MCS-51 单片机和 8255A 的接口	210
9.3 MCS-51 与可编程 RAM/IO 芯片 8155H 的接口	212
9.3.1 8155H 芯片介绍	212
9.3.2 8155H 的工作方式	216
9.3.3 MCS-51 与 8155H 接口及软件编程	218
9.4 用 74LS TTL 电路扩展并行 I/O 口	220
9.5 用 MCS-51 的串行口扩展并行口	222
9.5.1 用 74LS165 扩展并行输入口	222
9.5.2 用 74LS164 扩展并行输出口	223
思考题及习题	224
第 10 章 MCS-51 与键盘、显示器、拨盘、打印机的接口设计	226
10.1 LED 显示器接口原理	226
10.1.1 LED 显示器的结构	226
10.1.2 LED 显示器工作原理	228
10.2 键盘接口原理	230

10.2.1 键盘接口的工作原理	231
10.2.2 键盘的工作方式	236
10.3 键盘/显示器接口设计实例	237
10.3.1 利用并行 I/O 芯片 8155H 实现键盘/显示器接口	238
10.3.2 利用 8031 的串行口实现键盘/显示器接口	243
10.3.3 利用通用键盘/显示器接口芯片 8279 实现键盘/显示器接口	246
10.4 MCS-51 与液晶显示器(LCD)的接口	256
10.4.1 LCD 显示器的分类	256
10.4.2 点阵字符型液晶显示模块介绍	256
10.4.3 8031 与 LCD 的接口及软件编程	263
10.5 MCS-51 与微型打印机的接口	266
10.5.1 MCS-51 与 TP μ P-40A/16A 微型打印机的接口	266
10.5.2 MCS-51 与 GP16 微型打印机的接口	271
10.6 MCS-51 单片机与 BCD 码拨盘的接口设计	274
10.6.1 BCD 码拨盘	274
10.6.2 BCD 码拨盘与单片机的接口	275
思考题及习题	277
第 11 章 MCS-51 单片机与 D/A 转换器、A/D 转换器的接口 ...	279
11.1 MCS-51 与 DAC 的接口	279
11.1.1 D/A 转换器概述	279
11.1.2 MCS-51 与 8 位 DAC0832 的接口	281
11.1.3 MCS-51 与 12 位 DAC1208 的接口	287
11.1.4 MCS-51 与 12 位 DAC1230 系列的接口	291
11.2 MCS-51 与 ADC 的接口	291
11.2.1 A/D 转换器概述	291
11.2.2 MCS-51 与 ADC0809(逐次比较型)的接口	295
11.2.3 MCS-51 与 AD574(逐次比较型)的接口	298
11.2.4 MCS-51 与 A/D 转换器 MC14433(双积分型)的接口	302
11.3 MCS-51 与 V/F 转换器的接口	306
11.3.1 用 V/F 转换器实现 A/D 转换的原理	307
11.3.2 常用 V/F 转换器 LM331 简介	307
11.3.3 V/F 转换器与 MCS-51 单片机接口	308
11.3.4 LM331 应用举例	310
思考题及习题	311
第 12 章 MCS-51 的功率接口设计	313
12.1 MCS-51 的输出驱动能力及其外围集成数字驱动电路	313
12.1.1 MCS-51 片内 I/O 口的驱动能力	313

12.1.2 外围集成数字驱动电路	314
12.2 MCS-51 的开关型功率接口	316
12.2.1 MCS-51 与光电耦合器的接口	316
12.2.2 MCS-51 与继电器的接口	321
12.2.3 MCS-51 与晶闸管的接口	323
12.2.4 MCS-51 与集成功率电子开关输出接口	326
12.2.5 MCS-51 与固态继电器的接口	329
12.2.6 低压开关量信号输出技术	334
思考题及习题	334
第 13 章 MCS-51 的串行通信技术及其他扩展接口	335
13.1 MCS-51 单片机的串行通信接口技术	335
13.1.1 各种标准串行通信接口	335
13.1.2 MCS-51 单片机双机串行通信接口	344
13.1.3 MCS-51 单片机多机串行通信接口	346
13.1.4 PC 机与 MCS-51 的点对点的串行通信接口	348
13.1.5 PC 机与多个 MCS-51 单片机的串行通信接口	350
13.2 MCS-51 单片机与日历时钟芯片的接口	351
13.2.1 DS12887 日历时钟芯片的性能及引脚说明	352
13.2.2 DS12887 的内部 RAM 和寄存器	353
13.2.3 MCS-51 与 DS12887 的接口设计	357
13.3 MCS-51 单片机的报警接口	360
13.3.1 闪光报警接口	360
13.3.2 蜂鸣音报警接口	361
13.3.3 音乐报警接口	362
思考题及习题	363
第 14 章 MCS-51 应用系统的可靠性及抗干扰设计	364
14.1 干扰的来源	364
14.2 供电系统干扰及抗干扰措施	365
14.2.1 电源噪声来源、种类及危害	365
14.2.2 供电系统的抗干扰设计	366
14.3 过程通道干扰的抑制措施——隔离	367
14.3.1 光电隔离的基本配置	367
14.3.2 光电隔离的实现	368
14.4 空间干扰及抗干扰措施	370
14.4.1 接地技术	370
14.4.2 屏蔽技术	373
14.5 反电动势干扰的抑制	374

14.6 印制电路板的抗干扰设计	375
14.6.1 地线及电源线设计	375
14.6.2 去耦电容的配置	376
14.6.3 印制板布线的抗干扰设计	376
14.7 软件抗干扰措施	377
14.7.1 软件抗干扰的一般方法	378
14.7.2 软件滤波	378
14.7.3 开关量输入/输出软件抗干扰设计	382
14.7.4 指令冗余及软件陷阱	383
14.8 “看门狗”技术和掉电保护	386
14.8.1 “看门狗”和掉电保护的实现	386
14.8.2 微处理器监控器 MAX690A 简介	387
14.8.3 MCS-51 与微处理器监控器 MAX690A/MAX692A 的接口	389
思考题及习题	390
第 15 章 MCS-51 单片机应用系统的设计、开发与调试	392
15.1 MCS-51 单片机应用系统的设计步骤	392
15.2 应用系统的硬件设计	393
15.3 应用系统的软件设计	394
15.4 MCS-51 单片机系统举例	395
15.4.1 8031 的最小系统	395
15.4.2 89C51 的最小系统	396
15.4.3 以单片机为核心的数据采集系统	396
15.4.4 应用设计例 1——水温控制系统的设计	403
15.4.5 应用设计例 2——智能涡街流量计的设计	407
15.5 单片机应用系统的开发和调试	414
15.5.1 仿真开发系统简介	415
15.5.2 用户样机开发调试过程	417
15.5.3 用户样机硬件调试	418
思考题及习题	422
参考书目	423

第 1 章 单片机概述

单片机自 20 世纪 70 年代问世以来,以其极高的性能价格比,受到人们的重视和关注,应用很广,发展很快。单片机体积小,重量轻,抗干扰能力强,环境要求不高,价格低廉,可靠性高,灵活性好,开发较为容易。由于具有上述优点,在我国,单片机已广泛地应用在工业自动化控制、自动检测、智能仪器仪表、家用电器、电力电子、机电一体化设备等各个方面。

1.1 什么是单片机

什么是单片机?单片机就是在—块半导体硅片上集成了微处理器(CPU),存储器(RAM,ROM,EPROM)和各种输入、输出接口(定时器/计数器,并行 I/O 口,串行口,A/D 转换器以及脉宽调制器 PWM 等),这样—块集成电路芯片具有一台计算机的属性,因而被称为单片微型计算机,简称单片机。

单片机主要应用于测控领域,用以实现各种测试和控制功能。为了强调其控制属性,在国际上,多把单片机称为微控制器 MCU(MicroController Unit)。由于单片机在使用时,通常是处于测控系统的核心地位并嵌入其中,所以,通常也把单片机称为嵌入式控制器 EMCU(Embedded MicroController Unit)。而在我国,大部分工程技术人员则比较习惯于使用“单片机”这一名称。

单片机按照其用途可分为通用型和专用型两大类。

通用型单片机具有比较丰富的内部资源,性能全而且适应性强,可满足多种应用需求。通用型单片机是把可开发的内部资源,如 RAM、ROM、I/O 等功能部件等全部提供给用户。用户可以根据实际需要,充分利用单片机的内部资源,设计一个以通用单片机芯片为核心,再配以外围接口电路及其它外围设备,来满足各种不同需要的测控系统。通常所说的和本书所介绍的单片机是指通用型单片机。

然而,有许多应用是使用专门针对某些产品的特定用途而制作的单片机。例如,打印机、家用电器以及各种通信设备中的专用单片机等。这种应用的最大的

特点是针对性强且数量巨大。为此,单片机芯片制造商常与产品厂家合作,设计和生产专用的单片机芯片。这种专用的单片机芯片是为特定产品或某种测控应用而专门进行设计的。在设计中,已经对系统结构的最简化、可靠性和成本的最佳化等方面都作了全面的考虑,所以专用单片机具有十分明显的综合优势,也是今后单片机发展的一个重要方向。但是,无论专用单片机在用途上有多么“专”,其基本结构和工作原理都是以通用单片机为基础的。

1.2 单片机的历史及发展概况

单片机根据其基本操作处理的位数可分为:1位单片机、4位单片机、8位单片机、16位单片机和32位单片机。

继1971年微处理器的研制成功不久,就出现了单片机,但最早的单片机是1位的。

单片机的发展历史可分为四个阶段:

第一阶段(1974年—1976年):单片机初级阶段。因工艺限制,单片机采用双片的形式而且功能比较简单。例如,仙童公司生产的F8单片机,实际上只包括了8位CPU、64B RAM和2个并行口。因此,还需加1块3851(由1KB ROM、定时器/计数器和2个并行I/O构成)才能组成1台完整的计算机。

第二阶段(1976年—1978年):低性能单片机阶段。以Intel公司制造的MCS-48单片机为代表,这种单片机片内集成有8位CPU、并行I/O口、8位定时器/计数器、RAM和ROM等,但是不足之处是无串行口,中断处理比较简单,片内RAM和ROM容量较小且寻址范围不大于4KB。

第三阶段(1978年—现在):高性能单片机阶段。这个阶段推出的单片机普遍带有串行I/O口,多级中断系统,16位定时器/计数器,片内ROM、RAM容量加大,且寻址范围可达64KB,有的片内还带有A/D转换器。这类单片机的典型代表是:Intel公司的MCS-51系列、Motorola公司的6801和Zilog公司的Z8等。由于这类单片机的性能价格比高,所以仍被广泛应用,是目前应用数量较多的单片机。

第四阶段(1982年—现在):8位单片机巩固发展及16位单片机、32位单片机推出阶段。此阶段的主要特征是一方面发展16位单片机、32位单片机及专用型单片机;另一方面不断完善高档8位单片机,改善其结构,以满足不同的用户需要。16位单片机的典型产品如Intel公司生产的MCS-96系列单片机,其集成度已达120000管子/片,主振为12MHz,片内RAM为232B,ROM为8KB,中断处理为8级,而且片内带有多通道10位A/D转换器 and 高速输入/输

出部件(HSI/HSO),实时处理的能力很强。而 32 位单片机除了具有更高的集成度外,其主振已达 20 MHz,这使 32 位单片机的数据处理速度比 16 位单片机提高许多,性能比 8 位、16 位单片机更加优越。

1.3 8 位单片机的主要生产厂家和机型

20 世纪 80 年代以来,单片机的发展非常迅速。就通用单片机而言,世界上一些著名的计算机厂家已投放市场的产品就有几十个系列,数百个品种。其中有 Motorola 公司的 6801、6802。Zilog 公司的 Z8 系列, Rockwell 公司的 6501、6502 等。此外,荷兰的 PHILIPS 公司、日本的 NEC 公司、日立公司等也不甘落后,相继推出了各自的单片机品种,许多国外的公司以 MCS-51 的内核为基础,推出了各种与 MCS-51 系列单片机兼容的衍生品种。

目前世界上较为著名的 8 位单片机的生产厂家和主要机型如表 1-1 所示:

表 1-1 8 位单片机的生产厂家和型号

生产厂家	单片机型号
美国 Intel 公司	MCS-51 系列及其增强型、扩展型系列
美国 ATMEL 公司	89C51、89C52、89C55 等
荷兰 PHILIPS(飞利浦)公司	8XC552 系列
美国 Motorola 公司	6801 系列和 6805 系列
美国 Zilog 公司	Z8 系列及 SUPER8
美国 Fairchild 公司	F8 系列和 3870 系列
美国 Rockwell 公司	6500/1 系列
美国 TI 公司	TMS7000 系列
美国 NS(国家半导体)公司	NS8070 系列
日本松下(National)公司	MN6800 系列
美国 RCA(无线电)公司	CDP1800 系列
日本 HITACHI(日立)公司	HD6301, HD63L05, HD6305
日本 NEC(电气)公司	μ COM87(μ PD7800)系列

尽管单片机的品种很多,但是在我国使用最多的是 Intel 公司的 MCS-51 系列单片机及其增强型、扩展型的衍生机型。MCS-51 系列是在 MCS-48 系列的基础上于 20 世纪 80 年代初发展起来的,是最早进入国内的单片机主流品种之一。虽然它是 8 位的单片机,但它具有品种全、兼容性强、性能价格比高等特点,且软硬件应用设计资料丰富齐全,已为我国广大工程技术人员所熟悉。因此, MCS-51 系列单片机在我国得到了广泛的应用。直至现在, MCS-51 系列

的单片机及其衍生机型仍不失为单片机的主流系列,在最近的若干年内仍是工业检测、控制应用的主角。

1.4 单片机的发展趋势

单片机的发展趋势将是向大容量、高性能化,外围电路内装化等方面发展。为满足不同的用户要求,各公司竞相推出能满足不同需要的产品。

1. CPU 的改进

(1) 采用双 CPU 结构,以提高处理能力。

(2) 增加数据总线宽度,单片机内部采用 16 位数据总线,其数据处理能力明显优于一般 8 位单片机。

(3) 串行总线结构。飞利浦公司开发了一种新型总线: I²C 总线(Inter-IC bus)。该总线是用 2 根信号线代替现行的 8 位数据总线,从而大大地减少了单片机外部引线,使得单片机与外部接口电路连接简单。目前许多公司都在积极的开发此类产品。

2. 存储器的发展

(1) 加大存储容量。新型单片机片内 ROM 一般可达 4 KB 至 8 KB, RAM 为 256 B。有的单片机片内 ROM 容量可达 128 KB。

(2) 片内 EPROM 采用 E²PROM 或闪烁(Flash)存储器。片内 EPROM 由于需要高压(+21 V 或 +12 V)编程写入,紫外线擦抹给用户带来不便。采用 E²PROM 或闪烁存储器后,能在 +5 V 下读写,不需紫外线擦抹,既有静态 RAM 读写操作简便,又有在掉电时数据不会丢失的优点。片内 E²PROM 或闪烁存储器的使用,大大简化了应用系统结构。

(3) 程序保密化。一般 EPROM 中的程序很容易被复制。为防止复制,生产厂家对片内 E²PROM 或闪烁存储器采用加锁方式。加锁后,无法读取其中的程序,这就达到了程序保密的目的。

3. 片内 I/O 的改进

一般单片机都有较多的并行口,以满足外围设备、芯片扩展的需要,并配有串行口,以满足多机通信功能的要求。

(1) 增加并行口的驱动能力。这样可减少外部驱动芯片。有的单片机能直接输出大电流和高电压,以便能直接驱动 LED 和 VFD(荧光显示器)。

(2) 增加 I/O 口的逻辑控制功能。大部分单片机的 I/O 都能进行逻辑操作。中、高档单片机的位处理系统能够对 I/O 口进行位寻址及位操作,大大地加强了 I/O 口线控制的灵活性。

(3) 有些单片机设置了一些特殊的串行接口功能,为构成分布式、网络化系统提供了方便条件。

4. 外围电路内装化

随着集成度的不断提高,有可能把众多的外围功能器件集成在片内。这也是单片机发展的重要趋势。除了一般必须具有的 ROM、RAM、定时器/计数器、中断系统外,随着单片机档次的提高,以适应检测、控制功能更高的要求,片内集成的部件还有 A/D 转换器、D/A 转换器、DMA 控制器、中断控制器、锁相环、频率合成器、字符发生器、声音发生器、CRT 控制器、译码驱动器等。

随着集成电路技术及工艺的不断发展,能装入片内的外围电路也可以是大规模的,把所需的外围电路全部装入单片机内,即系统的单片化是目前单片机发展趋势之一。

5. 低功耗化

8 位单片机中有二分之一的产品已 CMOS 化,CMOS 芯片的单片机具有功耗小的优点,而且为了充分发挥低功耗的特点,这类单片机普遍配置有 Wait 和 Stop 两种工作方式。例如采用 CHMOS 工艺的 MCS-51 系列单片机 80C31/80C51/87C51 在正常运行(5 V,12 MHz)时,工作电流为 16 mA,同样条件下 Wait 方式工作时,工作电流则为 3.7 mA,而在 Stop 方式(2 V)时,工作电流仅为 50 nA。

综观单片机几十年的发展历程,单片机今后将向多功能、高性能、高速度、低电压、低功耗、低价格、外围电路内装化以及片内存储器容量增加和 Flash 存储器化方向发展。但其位数不一定会继续增加,尽管现在已经有了 32 位单片机,但使用的并不多。可以预言,今后的单片机将是功能更强、集成度和可靠性更高而功耗更低,以及使用更方便。

此外,专用化也是单片机的一个发展方向,针对单一用途的专用单片机将会越来越多。

1.5 单片机的应用

单片机以其卓越的性能,得到了广泛的应用,已深入到各个领域。单片机应用在检测、控制领域中,具有如下特点:

(1) 小巧灵活、成本低、易于产品化。它能方便地组装成各种智能测控设备及各种智能仪器仪表。

(2) 可靠性好,适应温度范围宽。单片机芯片本身是按工业测控环境要求设计的,分为民品、工业品、军品、其中工业品、军品具有较强的适应恶劣环境的能力。

MCS-51 系列单片机的民品、工业品、军品的使用温度范围大致为:

民品	0~+70℃
工业品	-40~+85℃
军品	-65~+125℃

设计者可根据不同的应用环境温度需要来选择不同的品种。

(3) 易扩展,很容易构成各种规模的应用系统,控制功能强。单片机的逻辑控制功能很强,指令系统有各种控制功能的指令。

(4) 可以很方便地实现多机和分布式控制系统。

单片机的应用范围很广,在下述的各个领域中得到了广泛的应用。

1. 工业自动化

在自动化技术中,无论是过程控制技术、数据采集还是测控技术,都离不开单片机。在工业自动化的领域中,机电一体化技术将发挥愈来愈重要的作用,在这种集机械、微电子和计算机技术为一体的综合技术(例如机器人技术)中,单片机将发挥非常重要的作用。

2. 智能仪器仪表

目前对仪器仪表的自动化和智能化要求越来越高。在智能仪器仪表中,单片机应用十分普及。单片机的使用有助于提高仪器仪表的精度和准确度,简化结构,减小体积而易于携带和使用,加速仪器仪表向数字化、智能化、多功能化方向发展。

3. 消费类电子产品

该应用主要反映在家电领域。目前家电产品的一个重要发展趋势是不断提高其智能化程度。例如,洗衣机、电冰箱、空调机、电视机、微波炉、手机、IC卡、汽车电子设备等。在这些设备中使用了单片机后,其功能和性能大大提高,并实现了智能化、最优化控制。

4. 通信方面

在调制解调器、程控交换技术以及各种通信设备,单片机得到了广泛的应用。

5. 武器装备

在现代化的武器装备中,如飞机、军舰、坦克、导弹、鱼雷制导、智能武器装备、航天飞机导航系统,都有单片机深入其中。

6. 终端及外部设备控制

计算机网络终端设备如银行终端以及计算机外部设备,如打印机、硬盘驱动器、绘图机、传真机、复印机等,在这些设备中都使用了单片机。

7. 多机分布式系统

可用多片单片机构成分布式测控系统,它使单片机的应用进入了一个新的水平。

综上所述,从工业自动化、智能仪器仪表、家用电器方面等,直到国防尖端技术领域,单片机都发挥着十分重要的作用。

1.6 MCS-51 系列单片机

MCS 是 Intel 公司生产的单片机的系列符号,例如 Intel 公司的 MCS-48、MCS-51、MCS-96 系列单片机。MCS-51 系列单片机既包括三个基本型 8031、8051、8751,也包括对应的低功耗型 80C31、80C51、87C51,因而 MCS-51 系列特指 Intel 公司的这几种型号的单片机。

20 世纪 80 年代中期以后,Intel 公司以专利转让的形式把 8051 内核技术转让给了许多半导体芯片生产厂家,如 ATMEL、PHILIPS、ANALOG DEVICES、DALLAS 公司等。这些厂家生产的芯片是 MCS-51 系列的兼容产品,准确地说与 MCS-51 指令系统兼容的单片机。这些兼容机与 8051 的系统结构(主要是指指令系统)相同,采用 CMOS 工艺,因而常用 80C51 系列来称呼所有具有 8051 指令系统的单片机。它们对 8051 一般都作了一些扩充,更有特点,其功能和市场竞争力更强,不应该把它们直接称为 MCS-51 系列单片机,因为 MCS 只是 Intel 公司专用的单片机系列符号。

MCS-51 系列及 80C51 系列单片机有多种品种。它们的指令系统相互兼容,主要在内部硬件结构上有些区别。目前使用的 MCS-51 系列单片机及其兼容产品通常分成以下几类:

1. 基本型

典型产品:8031/8051/8751。

8031 内部包括 1 个 8 位 CPU、128 B RAM,21 个特殊功能寄存器(SFR)、4 个 8 位并行 I/O 口、1 个全双工串行口,2 个 16 位定时器/计数器,但片内无程序存储器,需外扩 EPROM 芯片。

8051 是在 8031 的基础上,片内又集成有 4 KB ROM,作为程序存储器,是 1 个程序不超过 4 KB 的小系统。ROM 内的程序是公司制作芯片时,代为用户烧制的,出厂的 8051 都是含有特殊用途的单片机。所以 8051 应用在程序已定,且批量大的单片机产品中。

8751 是在 8031 基础上,增加了 4 KB 的 EPROM,它构成了 1 个程序小于 4 KB 的小系统。用户可以将程序固化在 EPROM 中,可以反复修改程序。但其价格相对于 8031 较贵。8031 外扩 1 片 4 KB 的 EPROM 就相当于 8751。

2. 增强型

Intel 公司在 MCS-51 系列三种基本型产品基础上,又推出增强型系列产品,即 52 子系列,典型产品:8032/8052/8752。它们的内部 RAM 增到 256 B,8052、8752 的内部程序存储器扩展到 8 KB,16 位定时器/计数器增至 3 个,6 个中断源,串行口通信速率提高 5 倍。

3. 低功耗型

代表性产品为:80C31/87C51/80C51。均采用 CMOS 工艺,功耗很低。例如,8051 的功耗为 630 mW,而 80C51 的功耗只有 120 mW,它们用于低功耗的便携式产品或航天技术中。此类单片机有 2 种掉电工作方式:一种掉电工作方式是 CPU 停止工作,其它部分仍继续工作;另一种掉电工作方式是,除片内 RAM 继续保持数据外,其它部分都停止工作。此类单片机的功耗低,非常适于电池供电或其它要求低功耗的场合。

4. 专用型

如 Intel 公司的 8044/8744,它们在 8051 的基础上,又增加一个串行接口部件,主要用于利用串行口进行通信的总线分布式多机测控系统。

再如美国 Cypress 公司最近推出的 EZU SR-2100 单片机,它是在 8051 单片机内核的基础上,又增加了 USB 接口电路,可专门用于 USB 串行接口通信。

5. 超 8 位型

在 8052 的基础上,采用 CHMOS 工艺,并将 MCS-96 系列(16 位单片机)中的一些 I/O 部件如:高速输入/输出(HSI/HSO)、A/D 转换器、脉冲宽度调制(PWM)、看门狗定时器(Watch Dog Timer, WDT)等移植进来构成新一代 MCS-51 产品,功能介于 MCS-51 和 MCS-96 之间。PHILIPS(飞利浦)公司生产的 80C552/87C552/83C552 系列单片机即为此类产品。目前此类单片机在我国已得到了较为广泛的使用。

6. 片内闪烁存储器型

随着半导体存储器制造技术和大规模集成电路制造技术的发展,片内带有闪烁(Flash)存储器的单片机在我国已得到广泛的应用。例如,美国 ATMEL 公司推出的 AT89C51 单片机。

在众多的 MCS-51 单片机及各种增强型、扩展型等衍生品种的兼容机中,PHILIPS(飞利浦)公司生产的 80C552/87C552/83C552 系列单片机和美国 ATMEL 公司的 AT89C51 单片机在我国使用较多。尤其是美国 ATMEL 公司推出的 AT89C51 单片机。它是 1 个低功耗、高性能的含有 4 KB 闪烁存储器的 8 位 CMOS 单片机,时钟频率高达 20 MHz,与 MCS-51 的指令系统和引脚完全兼容。闪烁存储器允许在线(+5 V)电擦除、电写入或使用编程器对其重复编程。此外,89C51 还支持由软件选择的 2 种掉电工作方式,非常适于电池供电

或其它要求低功耗的场合。由于片内带 EPROM 的 87C51 价格偏高,而 89C51 芯片内的 4 KB 闪烁存储器可在线编程或使用编程器重复编程,且价格较低,因此 89C51 受到了应用设计者的欢迎。

尽管 MCS-51 系列以及 80C51 系列单片机有多种类型,但是掌握好 MCS-51 的基本型(8031、8051、8751 或 80C31、80C51、87C51)是十分重要的,因为它们是具有 MCS-51 内核的各种型号单片机的基础,也是各种增强型、扩展型等衍生品种的核心。

本书常用 MCS-51 或 8031 这两个名称,MCS-51 是包括了 8031、8051 和 8751 3 个基本产品的总称。后者,仅指特定的 8031。

思考题及习题

1. 微处理器、微计算机、微处理机、CPU、单片机它们之间有何区别?
2. 除了单片机这一名称之外,单片机还可称为()和()。
3. 单片机与普通计算机的不同之处在于其将()()和()三部分集成于一块芯片上。
4. 单片机的发展大致分为哪几个阶段?
5. 单片机根据其基本操作处理的位数可分为哪几种类型?
6. MCS-51 系列单片机的基本型芯片分别为哪几种? 它们的差别是什么?
7. MCS-51 系列单片机与 80C51 系列单片机的异同点是什么?
8. 8051 与 8751 的区别是:
(A) 内部数据存储单元数目的不同 (B) 内部数据存储器的类型不同
(C) 内部程序存储器的类型不同 (D) 内部的寄存器的数目不同
9. 在家用电器中使用单片机应属于微计算机的
(A) 辅助设计应用 (B) 测量、控制应用 (C) 数值计算应用 (D) 数据处理应用
10. 说明单片机主要应用在哪些领域?

第2章 MCS-51 单片机的硬件结构

本章介绍 MCS-51 单片机的硬件结构。熟悉并掌握硬件结构是十分重要的,因为它是单片机应用系统设计的基础。单片机是微计算机的一个分支,在原理和结构上,单片机与微计算机之间不但没有根本性的差别,而且微计算机的许多技术与特点都被单片机继承下来。所以,可以用学习微计算机的思路来学习单片机。

通过对本章的学习,可以使读者对 MCS-51 单片机的硬件结构有较为全面的了解,从应用系统设计的角度,牢记 MCS-51 单片机为用户提供了哪些硬件资源,如何去应用它们。

2.1 MCS-51 单片机的硬件结构

MCS-51 单片机的片内结构如图 2-1 所示。MCS-51 单片机是把这些作为控制应用所必需的基本内容都集成在一个尺寸有限的集成电路芯片上。如果按功能划分,它由如下功能部件组成:

- (1) 微处理器(CPU)。
- (2) 数据存储器(RAM)。
- (3) 程序存储器(ROM/EPROM),8031 没有此部件。
- (4) 4 个 8 位并行 I/O 口(P0 口、P1 口、P2 口、P3 口)。
- (5) 1 个串行口。
- (6) 2 个 16 位定时器/计数器。
- (7) 中断系统。
- (8) 特殊功能寄存器(SFR)。

上述各功能部件都是通过片内单一总线连接而成(见图 2-1),其基本结构依旧是 CPU 加上外围芯片的传统结构模式。但 CPU 对各种功能部件的控制是采用特殊功能寄存器(Special Function Register, SFR)的集中控制方式。

下面对图 2-1 中的各功能部件作以介绍:

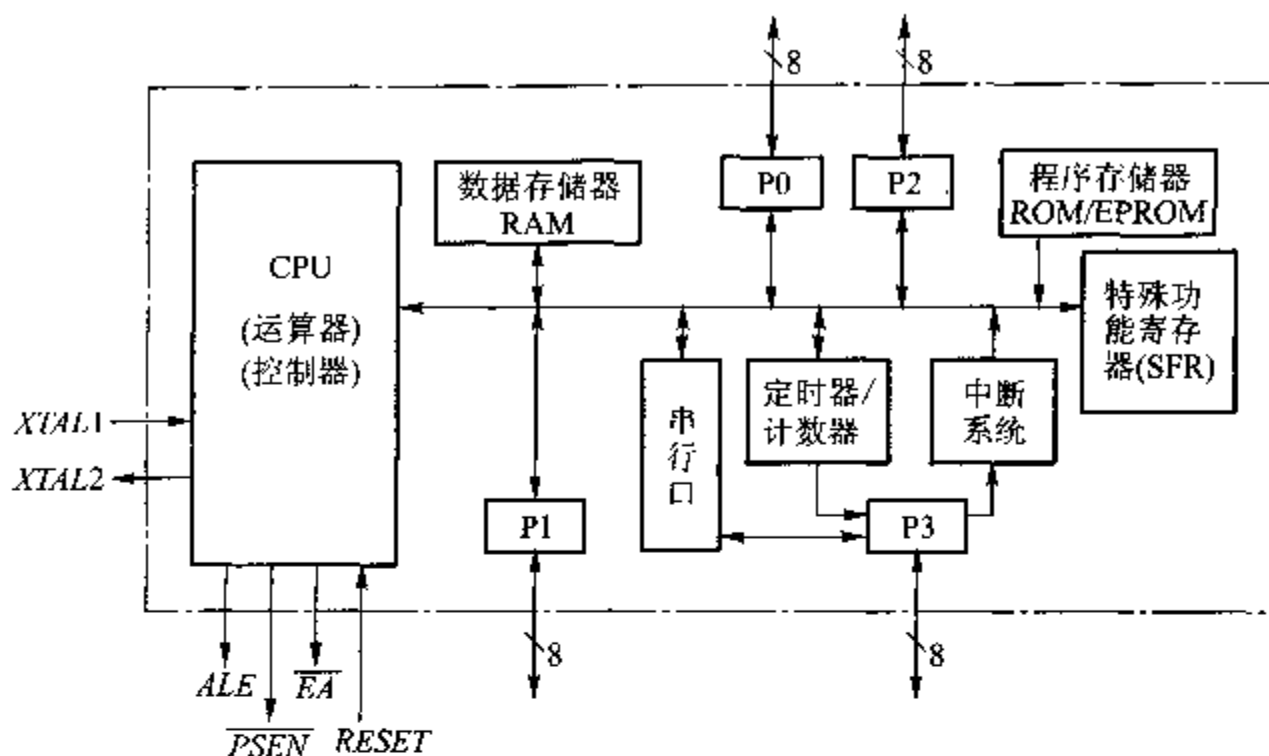


图 2-1 MCS-51 单片机片内结构

1. CPU(微处理器)

MCS-51 单片机中有 1 个 8 位的 CPU,与通用的 CPU 基本相同,同样包括了运算器和控制器两大部分,只是增加了面向控制的处理功能,不仅可处理字节数据,还可以进行位变量的处理。例如:位处理、查表、状态检测、中断处理等。

2. 数据存储器(RAM)

片内为 128 B(52 子系列的为 256 B),片外最多可外扩 64 KB。数据存储器来存储单片机运行期间的工作变量、运算的中间结果、数据暂存和缓冲、标志位等。片内的 128 B 的 RAM,以高速 RAM 的形式集成在单片机内,可以加快单片机运行的速度,而且这种结构的 RAM 还可以降低功耗。

3. 程序存储器(ROM/EPROM)

用来存储程序,8031 无此部件;8051 为 4 KB ROM;8751 则为 4 KB EPROM。如果片内只读存储器的容量不够,则需用扩展片外只读存储器,片外最多可外扩至 64 KB。

4. 中断系统

具有 5 个中断源,2 级中断优先权。

5. 定时器/计数器

片内有 2 个 16 位的定时器/计数器(52 子系列有 3 个 16 位的定时器/计数器),具有 4 种工作方式。在单片机的应用中,往往需要精确的定时,或对外部事件进行计数,因而需在单片机内部设置定时器/计数器部件。

6. 串行口

1 个全双工的串行口,具有 4 种工作方式。可用来进行串行通信,扩展并行

I/O 口,甚至与多个单片机相连构成多机系统,从而使单片机的功能更强且应用更广。

7. P1 口、P2 口、P3 口、P0 口

为 4 个并行 8 位 I/O 口。

8. 特殊功能寄存器(SFR)

特殊功能寄存器共有 21 个,用于 CPU 对片内各功能部件进行管理、控制、监视。实际上是一些控制寄存器和状态寄存器,是一个具有特殊功能的 RAM 区。

由上可见,MCS-51 单片机的硬件结构具有功能部件种类全,功能强等特点。特别值得一提的是 MCS-51 CPU 中的位处理器,它实际上是 1 个完整的 1 位微计算机,这个 1 位微计算机有自己的 CPU、位寄存器、I/O 口和指令集。1 位机在开关决策、逻辑电路仿真、工业控制方面非常有效;而 8 位机在数据采集,运算处理方面有明显的长处。MCS-51 单片机中 8 位机和 1 位机的硬件资源复合在一起,二者相辅相成,它是单片机技术上的 1 个突破,这也是 MCS-51 在设计上的精美之处。

2.2 MCS-51 的引脚

掌握 MCS-51 单片机,应首先了解 MCS-51 的引脚,熟悉并牢记各引脚的功能。MCS-51 系列中各种型号芯片的引脚是互相兼容的。制造工艺为 HMOS 的 MCS-51 的单片机都采用 40 只引脚的双列直插封装(DIP)方式,如图 2-2 所示。目前大多数为此类封装方式。制造工艺为 CHMOS 的 80C31/80C51/87C51 除采用 DIP 封装方式外,还采用方形封装方式,为 44 只引脚(其中 4 只是无用的引脚),如图 2-3 所示。

40 只引脚按其功能来分,可分为如下 3 类:

(1) 电源及时钟引脚: V_{CC} 、 V_{SS} ; XTAL1、XTAL2。

(2) 控制引脚: \overline{PSEN} 、 ALE 、 \overline{EA} 、RESET(即 RST)。

(3) I/O 口引脚: P0、P1、P2、P3,

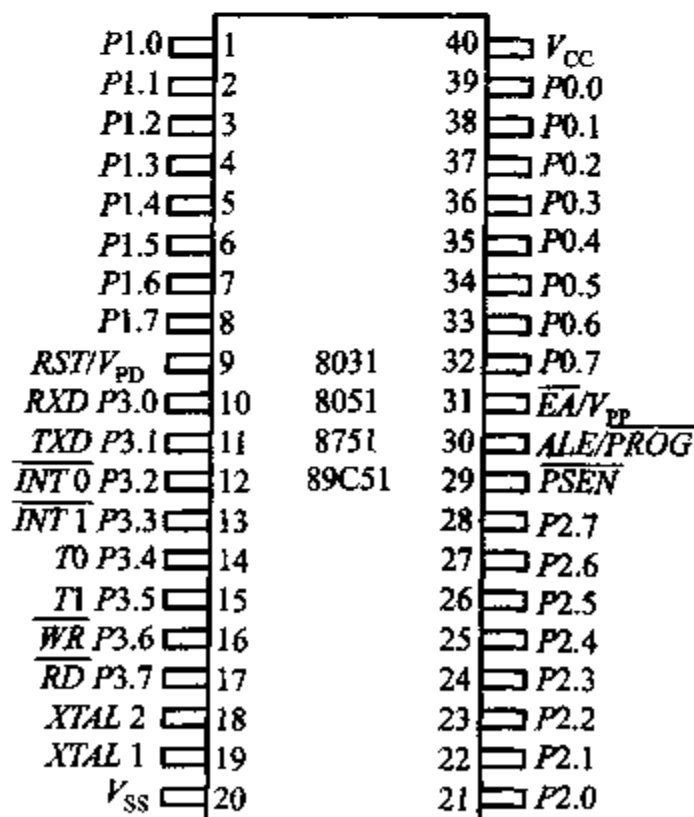


图 2-2 MCS-51 双列直插封装方式的引脚

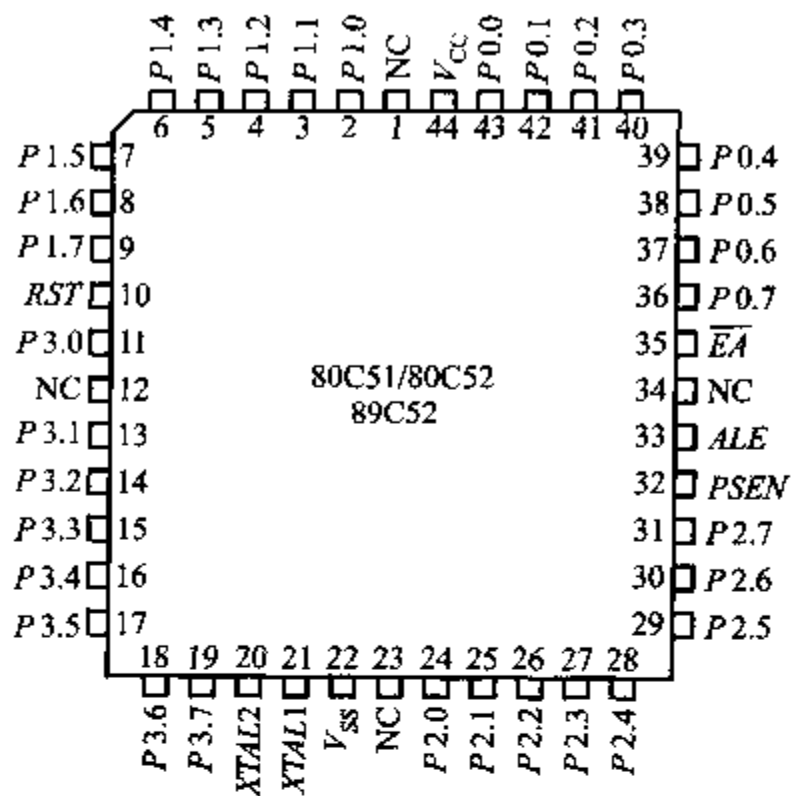


图 2-3 MCS-51 的方形封装方式的引脚

为 4 个 8 位 I/O 口的外部引脚。

下面结合图 2-2 来介绍各引脚的功能。

2.2.1 电源及时钟引脚

1. 电源引脚

电源引脚接入单片机的工作电源。

(1) V_{CC} (40 引脚): 接 +5 V 电源。

(2) V_{SS} (20 引脚): 接地。

2. 时钟引脚

2 个时钟引脚 XTAL1、XTAL2 外接晶体与片内的反相放大器构成了 1 个振荡器, 它为单片机提供了时钟控制信号。2 个时钟引脚也可外接独立的晶体振荡器。

(1) XTAL1 (19 引脚): 接外部晶体的 1 个引脚。该引脚内部是 1 个反相放大器的输入端。这个反相放大器构成了片内振荡器。如果采用外接晶体振荡器时, 此引脚应接地。

(2) XTAL2 (18 引脚): 接外部晶体的另一端, 在该引脚内部接至内部反相放大器的输出端。若采用外部时钟振荡器时, 该引脚接收时钟振荡器的信号, 即把此信号直接接到内部时钟发生器的输入端。

2.2.2 控制引脚

此类引脚提供控制信号,有的引脚还具有复用功能。

(1) RST/ V_{PD} (9 引脚):RST(RESET)是复位信号输入端,高电平有效。当单片机运行时,在此引脚加上持续时间大于 2 个机器周期(24 个时钟振荡周期)的高电平时,就可以完成复位操作。在单片机正常工作时,此引脚应为 $\leq 0.5\text{ V}$ 低电平。

V_{PD} 为本引脚的第二功能,即备用电源的输入端。当主电源 V_{CC} 发生故障,降低到某一规定值的低电平时,将 +5 V 电源自动接入 RST 端,为内部 RAM 提供备用电源,以保证片内 RAM 中的信息不丢失,从而使单片机在复位后能继续正常运行。

(2) ALE/ $\overline{\text{PROG}}$ (Address Latch Enable/PROGramming, 30 引脚):ALE 引脚输出为地址锁存允许信号,当单片机上电正常工作后,ALE 引脚不断输出正脉冲信号。当单片机访问外部存储器时,ALE 输出信号的负跳沿用于单片机发出的低 8 位地址经外部锁存器锁存的锁存控制信号。即使不访问外部锁存器,ALE 端仍有正脉冲信号输出,此频率为时钟振荡器频率 f_{osc} 的 1/6。如果想初步判断单片机芯片的好坏,可用示波器查看 ALE 端是否有正脉冲信号输出。如果有脉冲信号输出,则单片机基本上是好的。

应当注意的是,每当 MCS-51 访问外部数据存储器时(即执行的是 MOVX 类指令),在 2 个机器周期中 ALE 只出现 1 次,即丢失 1 个 ALE 脉冲。因此,严格来说,用户不宜用 ALE 作精确的时钟源或定时信号。ALE 端可以驱动 8 个 LS 型 TTL 负载。

$\overline{\text{PROG}}$ 为本引脚的第二功能。在对片内 EPROM 型单片机(例如 8751)编程写入时,此引脚作为编程脉冲输入端。

(3) $\overline{\text{PSEN}}$ (Program Strobe ENable, 29 引脚):程序存储器允许输出控制端。在单片机访问外部程序存储器时,此引脚输出脉冲负跳沿作为读外部程序存储器的选通信号。此引脚接外部程序存储器的 $\overline{\text{OE}}$ (输出允许)端。 $\overline{\text{PSEN}}$ 端可以驱动 8 个 LS 型 TTL 负载。

如要检查一个 MCS-51 单片机应用系统上电后,CPU 能否正常到外部程序存储器读取指令码,可用示波器查 $\overline{\text{PSEN}}$ 端有无脉冲输出。

(4) $\overline{\text{EA}}$ / V_{PP} (Enable Address/Voltage Pulse of Programing, 31 引脚): $\overline{\text{EA}}$ 功能为内外程序存储器选择控制端。

当 $\overline{\text{EA}}$ 引脚为高电平时,单片机访问片内程序存储器,但在 PC(程序计数器)值超过 0FFFFH(对于 8051、8751)时,即超出片内程序存储器的 4 KB 地址范围时,将自动转向执行外部程序存储器内的程序。

当 \overline{EA} 引脚为低电平时,单片机则只访问外部程序存储器,不论是否有内部程序存储器。对于 8031 来说,因其无内部程序存储器,所以该引脚必须接地,这样只能选择外部程序存储器。

V_{PP} 为本引脚的第二功能。在对 EPROM 型单片机 8751 片内 EPROM 固化编程时,用于施加较高的编程电压(例如 +21 V 或 +12 V)。对于 89C51,则加在 V_{PP} 引脚的编程电压为 +12 V 或 +5 V。

2.2.3 I/O 口引脚

(1) P0 口:双向 8 位三态 I/O 口,此口为地址总线(低 8 位)及数据总线分时复用口,可驱动 8 个 LS 型 TTL 负载。

(2) P1 口:8 位准双向 I/O 口,可驱动 4 个 LS 型 TTL 负载。

(3) P2 口:8 位准双向 I/O 口,与地址总线(高 8 位)复用,可驱动 4 个 LS 型 TTL 负载。

(4) P3 口:8 位准双向 I/O 口,双功能复用口,可驱动 4 个 LS 型 TTL 负载。

这里要特别注意准双向口与双向三态口的差别。

P1 口、P2 口、P3 口是 3 个 8 位准双向的 I/O 口,各口线在片内均有固定的上拉电阻。当这 3 个准双向 I/O 口作输入口使用时,要向该口先写 1,另外准双向 I/O 口无高阻的“浮空”状态。

而双向口 P0 口线内无固定上拉电阻,由两个 MOS 管串接,既可开漏输出,又可处于高阻的“浮空”状态,故称为双向三态 I/O 口。以上的解释,读者在阅读 2.5 节后,将会有深刻的理解。

至此,MCS-51 单片机的 40 只引脚已介绍完毕,读者应熟记每一个引脚的功能,这对于掌握 MCS-51 单片机以及应用系统的硬件设计是十分重要的。

2.3 MCS-51 的 CPU

由图 2-1 可见, MCS-51 的 CPU 是由运算器和控制器所构成的。

2.3.1 运算器

运算器主要用来对操作数进行算术、逻辑运算和位操作的。主要包括算术逻辑运算单元 ALU、累加器 A、位处理器、程序状态字寄存器 PSW 以及 BCD 码修正电路等。

1. 算术逻辑运算单元 ALU

ALU 的功能十分强,它不仅可对 8 位变量进行逻辑与、或、异或、循环、求补和清零等基本操作,还可以进行加、减、乘、除等基本算术运算。MCS-51 的 ALU 还具有位处理操作功能,它可对位(bit)变量进行位处理,如置位、清零、求补、测试转移及逻辑与、或等操作。

2. 累加器 A

累加器 A 是 1 个 8 位的累加器,是 CPU 中使用最频繁的 1 个寄存器,也可写为 Acc。

累加器的作用是:

(1) 累加器 A 是 ALU 单元的输入之一,因而是数据处理源之一。但它又是 ALU 运算结果的存放单元。

(2) CPU 中的数据传送大多都通过累加器 A,故累加器 A 又相当于数据的中转站。由于数据传送大多都通过累加器 A,故累加器容易产生“堵塞”现象,即累加器结构所具有的“瓶颈”现象。为此,MCS-51 单片机增加了一部分可以不经过累加器的传送指令,这样,即可加快数据的传送速度,又减少了累加器的“瓶颈堵塞”现象。

累加器 A 的进位标志 Cy 是特殊的,因为它同时又是位处理机的位累加器。

3. 程序状态字寄存器 PSW

MCS-51 的程序状态字寄存器 PSW(Program Status Word),是 1 个 8 位可读写的寄存器,位于单片机片内的特殊功能寄存区,字节地址为 D0H。PSW 的不同位包含了程序运行状态的不同信息,掌握并牢记 PSW 各位的含义是十分重要的,因为在程序设计中,经常会与 PSW 的各个位打交道。PSW 的格式如图 2-4 所示。

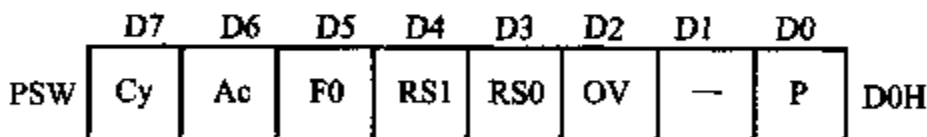


图 2-4 PSW 的格式

PSW 中的各个位的功能如下:

(1) Cy(PSW. 7)进位标志位: Cy 也可写为 C。在执行算术和逻辑指令时, Cy 可以被硬件或软件置位或清除,在位处理器中,它是位累加器。

(2) Ac(PSW. 6)辅助进位标志位: 当进行 BCD 码加法或减法操作而产生的由低 4 位数(代表 1 个 BCD 码)向高 4 位进位或借位时, Ac 将被硬件置 1, 否则被清 0。Ac 用于在进行 BCD 码运算时, 用作十进位调整, 同 DA 指令结合起来用。有关十进位调整的问题, 将在后面有关章节介绍。

(3) F0(PSW. 5)标志位: 它是由用户使用的一个状态标志位, 可用软件来使它

置 1 或清 0,也可由指令来测试标志位 F0,用以控制程序的流向。编程时,用户应充分地利用该标志位。

(4) RS1、RS0(PSW.4、PSW.3)——4 组工作寄存器区选择控制位 1 和位 0:这 2 位用来选择 4 组工作寄存器区中的哪一组为当前工作寄存器区(4 组寄存器在单片机内的 RAM 区中,将在本章稍后介绍),它们与 4 组工作寄存器区的对应关系如下:

RS1	RS0	所选的 4 组寄存器
0	0	0 区(内部 RAM 地址 00H~07H)
0	1	1 区(内部 RAM 地址 08H~0FH)
1	0	2 区(内部 RAM 地址 10H~17H)
1	1	3 区(内部 RAM 地址 18H~1FH)

(5) OV(PSW.2)溢出标志位:当执行算术指令时,由硬件置 1 或清 0,以指示运算是否产生溢出。各种算术运算指令对该位的影响情况较为复杂,将在第 3 章介绍。

(6) PSW.1 位:保留位,未用。

(7) P(PSW.0)奇偶标志位:该标志位用来表示累加器 A 中为 1 的位数的奇偶数。

$P=1$,则 A 中 1 的个数为奇数

$P=0$,则 A 中 1 的个数为偶数

此标志位对串行口通信中的串行数据传输有重要的意义,在串行通信中,常用奇偶检验的方法来检验数据传输的可靠性。

2.3.2 控制器

控制器是单片机的指挥控制部件,控制器的主要任务是识别指令,并根据指令的性质控制单片机各功能部件,从而保证单片机各部分能自动而协调地工作。

单片机执行指令是在控制器的控制下进行的。首先从程序存储器中读出指令,送入指令寄存器保存,然后送到指令译码器对指令进行译码,译码结果送定时控制逻辑电路,由定时控制逻辑电路产生各种定时信号和控制信号,再送到单片机的各个部件去进行相应的操作。这就是执行一条指令的全过程。

控制器主要包括程序计数器、程序地址寄存器、指令寄存器 IR、指令译码器、条件转移逻辑电路及时序控制逻辑电路。

1. 程序计数器 PC(Program Counter)

程序计数器 PC 是控制器中最基本的寄存器,是一个独立的计数器,存放着

下一条要执行的指令在程序存储器中的地址。

程序计数器 PC 基本的工作过程是:读指令时,程序计数器将其中的数作为所取指令的地址输出给程序存储器,然后程序存储器按此地址输出指令字节,同时程序计数器本身内容自动增加,指向下一条指令在程序存储器中的地址。

PC 中内容的变化决定程序的流向。PC 的位数决定了单片机对程序存储器可以直接寻址的范围。在 MCS-51 中,程序计数器 PC 是 1 个 16 位的计数器,故可对 64 KB($64K=2^{16}=65536$)的程序存储器进行寻址。

程序计数器的基本工作方式有以下几种:

(1) 程序计数器自动加 1,这是最基本的工作方式,这也是为何该寄存器被称为计数器的原因。

(2) 执行有条件或无条件转移指令时,程序计数器将被置入新的数值,从而使程序的流向发生变化。

(3) 在执行调用子程序调用或响应中断时,单片机自动完成如下的操作:

① PC 的现行值,即下一条将要执行的指令的地址,也就是断点值,自动压入堆栈,保护起来。

② 将子程序的入口地址或中断向量的地址送入 PC,程序流向发生变化,去执行子程序或中断服务子程序。子程序或中断服务子程序执行完毕,遇到返回指令 RET 或 RETI 时,将栈顶的断点值弹到程序计数器 PC 中,程序的流向又返回到断点处,从断点处继续执行程序。

2. 指令寄存器 IR、指令译码器及控制逻辑电路

指令寄存器 IR 是用来存放指令操作码。IR 的输出送指令译码器;然后对该指令进行译码,译码结果送定时控制逻辑电路。

定时控制逻辑电路根据对指令的译码结果,发出一系列的定时控制信号,控制单片机的各组成部件进行相应的工作,执行指令。

综上所述,单片机整个程序的执行过程就是在 CPU 控制部件的控制下,以主振频率为基准(每个主振周期称为振荡周期),将指令从程序存储器中逐条取出,进行译码,然后由定时控制逻辑电路发出各种定时控制信号,将各个硬件环节的运行组织在一起。对于运算指令,还要将运算的结果特征送入程序状态字寄存器 PSW。

有关 CPU 的时序,在 2.6 节中结合时钟电路进行介绍。

2.4 MCS-51 存储器的结构

MCS-51 单片机存储器采用的是哈佛(Harvard)结构,即程序存储器空间

和数据存储器空间是各自独立的,两种存储器各有自己的寻址方式和寻址空间。

这种结构对于单片机“面向控制”的实际应用极为方便、有利。MCS 51 单片机程序存储器和数据存储器的扩展能力分别可达 64 KB,寻址和操作简单方便。

MCS-51 的存储器空间可划分为如下 5 类:

1. 程序存储器

单片机系统之所以能够按照一定的次序进行工作,主要是程序存储器中存放了经调试正确的应用程序和表格之类的固定常数。程序实际上是一串二进制码,程序存储器可以分为片内和片外两部分。8031 由于无内部程序存储器,所以只能通过外部扩展程序存储器来存放程序。

2. 内部数据存储器

MCS-51 单片机内部有 128 B 的随机存取存储器 RAM,作为用户的数据寄存器,它能满足大多数控制型应用场合的需要,用作处理问题的数据缓冲区。

3. 特殊功能寄存器(Special Function Register, SFR)

特殊功能寄存器实际上是 MCS-51 各功能部件的状态及控制寄存器。例如,前面提到的 PSW 程序状态字寄存器,就是一个特殊功能寄存器。掌握理解好 SFR,对于掌握 MCS-51 是十分重要的。SFR 综合的、实际的反映了整个单片机基本系统内部的工作状态及工作方式。在单片机中设置 SFR,为程序设计提供了不少方便。

4. 位地址空间

MCS-51 的一个很大优点在于它具有 1 个功能很强的位处理机。在 MCS-51 单片机内共有 211 个可寻址位,它们存在于内部 RAM(共有 128 个)和特殊功能寄存器区(共有 83 个)中。在 MCS-51 的指令系统中,有 1 个位处理指令的子集,使用这些指令,所处理的数据仅为 1 位二进制数(0 或 1)。

5. 外部数据寄存器

当 MCS-51 的片内 RAM 不够用时,又给用户提供了在片外可扩展 64 KB RAM 的能力,至于扩多少 RAM,则根据用户实际需要来定。

2.4.1 程序存储器

MCS-51 单片机的程序存储器用于存放经调试正确的应用程序和表格之类的固定常数。由于采用 16 位的程序计数器 PC 和 16 位地址总线,因而可扩充的程序存储器空间最大为 64 KB。有关程序存储器的使用应注意以下两点:

(1) 整个程序存储器空间可以分为片内和片外两部分,CPU 访问片内和片外程序存储器,可由 \overline{EA} 引脚上所接的电平来确定。

\overline{EA} 引脚接高电平时,程序将从片内程序存储器开始执行,即访问片内程序

存储器；当 PC 值超出片内 ROM 的容量时，会自动转向片外程序存储器空间执行程序。

\overline{EA} 引脚接低电平时，迫使单片机只能执行片外程序存储器中的程序。

对于片内有 ROM·EPROM 的 8051、8751 单片机，应将 \overline{EA} 引脚固定接高电平。若把 \overline{EA} 引脚接低电平，可用于程序调试，即 CPU 执行片外程序存储器中的程序来进行程序调试。

8031 无内部程序存储器，应将 \overline{EA} 引脚固定接低电平。

无论从片内或片外程序存储器读取指令，其操作运行速度都是相同的。

(2) 程序存储器的某些单元被固定用于各中断源的中断服务程序的入口地址。

MCS-51 复位后，程序存储器 PC 的内容为 0000H，故系统必须从 0000H 单元开始取指令，执行程序。程序存储器中的 0000H 地址是系统程序的启动地址，这一点初学者要牢记。一般在该单元存放一条绝对跳转指令，跳向主程序的入口地址。

64 KB 程序存储器中有 5 个单元具有特殊用途。5 个特殊单元分别对应于 5 种中断源的中断服务程序的入口地址，见表 2-1。

通常在这些中断入口地址处都放一条绝对跳转指令跳向中断服务子程序，而不是存放中断服务子程序。加跳转指令的目的是，由于 2 个中断入口间隔仅有 8 个单元，如果这 8 个单元存放中断服务子程序，往往是不够用的，所以要利用跳转指令跳向中断服务子程序。

表 2-1 5 种中断源的中断入口地址

中 断 源	入 口 地 址
外部中断 0 ($\overline{INT0}$)	0003H
定时器 0 (T0)	000BH
外部中断 1 ($\overline{INT1}$)	0013H
定时器 1 (T1)	001BH
串行口	0023H

2.4.2 内部数据存储器

MCS-51 的片内数据存储器 (RAM) 单元共有 128 个，字节地址为 00H~7FH。MCS-51 对其内部 RAM 有很丰富的操作指令，从而使得用户在设计程序时非常方便。图 2-5 为 MCS-51 片内数据存储器的结构。

地址为 00H~1FH 的 32 个单元是 4 组通用工作寄存器区,每个区包含 8 个 8 位工作寄存器,编号为 R7~R0。用户可以通过指令改变 PSW 中的 RS1、RS0 这 2 位来切换当前的工作寄存器区,这种功能给软件设计带来极大的方便,特别是在中断嵌套时,为实现工作寄存器现场内容保护提供了极大的方便。

地址为 20H~2FH 的 16 个单元可进行共 128 位的位寻址,这些单元构成了 1 位处理机的存储器空间。单元中的每一位都有自己的位地址,这 16 个单元也可以进行字节寻址。

地址为 30H~7FH 的单元为用户 RAM 区,只能进行字节寻址。用于作为数据缓冲区以及堆栈区。

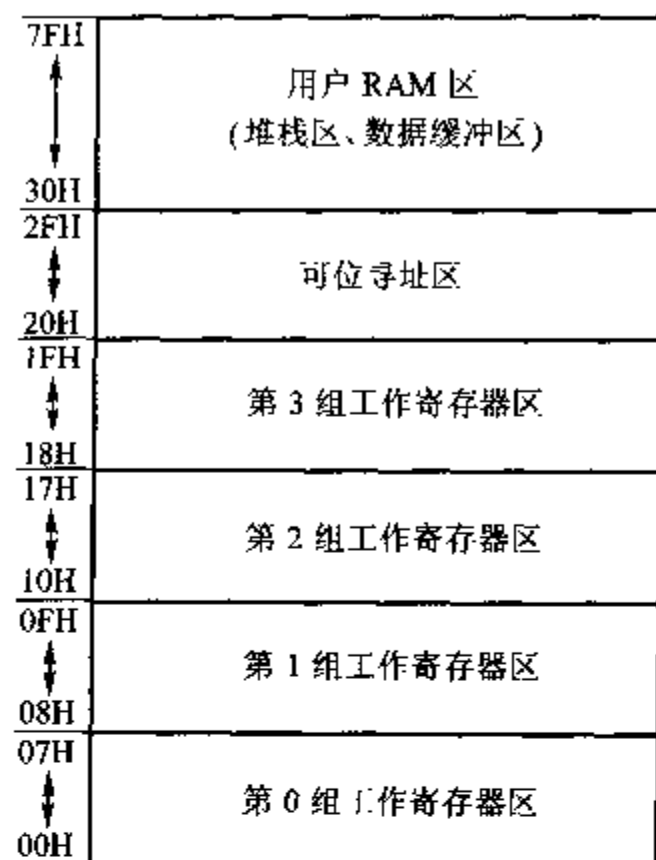


图 2-5 MCS-51 片内 RAM 的结构

2.4.3 特殊功能寄存器(SFR)

MCS-51 中的 CPU 对各种功能部件的控制是采用特殊功能寄存器(Special Function Register, SFR)的集中控制方式。SFR 实质上是一些具有特殊功能的片内 RAM 单元,字节地址范围为 80H~FFH。特殊功能寄存器的总数为 21 个,离散地分布在该区域中,其中有些 SFR 还可以进行位寻址。表 2-2 是 SFR 的名称及其分布。

表 2-2 SFR 的名称及其分布

特殊功能 寄存器符号	名 称	字节地址	位 地 址
B	B 寄存器	F0H	F7H~F0H
A (或 Acc)	累加器	E0H	E7H~E0H
PSW	程序状态字	D0H	D7H~D0H
IP	中断优先级控制	B8H	BFH~B8H
P3	P3 口	B0H	B7H~B0H
IE	中断允许控制	A8H	AFH~A8H
P2	P2 口	A0H	A7H~A0H
SBUF	串行数据缓冲器	99H	

续表

特殊功能 寄存器符号	名 称	字 节 地 址	位 地 址
SCON	串行控制	98H	9FH~98H
P1	P1 口	90H	97H~90H
TH1	定时器/计数器 1(高字节)	8DH	
TH0	定时器/计数器 0(高字节)	8CH	
TL1	定时器/计数器 1(低字节)	8BH	
TL0	定时器/计数器 0(低字节)	8AH	
TMOD	定时器/计数器方式控制	89H	
TCN	定时器/计数器控制	88H	8FH~88H
PCON	电源控制	87H	
DPH	数据指针高字节	83H	
DPL	数据指针低字节	82H	
SP	堆栈指针	81H	
P0	P0 口	80H	87H~80H

从表 2-2 中可发现一个规律,凡是可进行位寻址,即具有位地址的 SFR,其字节地址的末位,只能是 0H 或 8H。另外,要注意的是,128 B 的 SFR 块中仅有 21 B 是有定义的。对于尚未定义的字节地址单元,用户不能作普通寄存器使用,若访问没有定义的单元,则将得到一个不确定的随机数。

下面简单介绍 SFR 块中的某些寄存器,累加器 Acc 和程序状态字寄存器 PSW 已在前面作了介绍。其它没有介绍的特殊功能寄存器将在后续的有关章节进行介绍。

1. 堆栈指针 SP

堆栈指针 SP 是 1 个 8 位的特殊功能寄存器,堆栈是在片内 RAM 中开辟出来的 1 个区域。SP 的内容指示出堆栈顶部在内部 RAM 块中的位置。它可指向内部 RAM 00H~7FH 的任何单元。MCS-51 的这种堆栈结构是属于向上生长型的堆栈(另一种是属于向下生长型的堆栈)。单片机复位后,SP 中的内容为 07H,使得堆栈实际上是从 08H 单元开始,考虑到 08H~1FH 单元分别属于 1~3 组的工作寄存器区,若在程序设计中要用到这些工作寄存器区,则最好把 SP 值改置为 1FH 或更大的值。

堆栈的主要功能是为子程序调用和中断操作面设立的。堆栈的具体功能有两个:保护断点和现场保护。

(1) 保护断点:因为无论是子程序调用操作还是执行中断操作,最终都要返回主程序。因此,应预先把主程序的断点在堆栈中保护起来,为程序的正确返回做准备。

(2) 现场保护:在单片机去执行子程序或中断服务程序之后,很可能要用到单片机中的一些寄存器单元,这样就会破坏这些寄存器单元中的原有内容。所以在执行子程序或中断服务程序之前要把单片机中有关寄存器单元的内容保存起来,送入堆栈,这就是所谓的现场保护。

此外,堆栈也可用于数据的临时存放,在程序的设计中时常用到。

堆栈的操作有两种:一种是数据压入(PUSH)堆栈,另一种是数据弹出(POP)堆栈。堆栈的栈顶由 SP 自动管理。每次进行压入或弹出操作以后,SP 自动加 1;1 B 数据弹出堆栈后,SP 自动减 1。例如 $SP=60H$,CPU 执行 1 条子程序调用指令或响应中断后,PC 内容(断点)进栈,PC 的低 8 位 PCL 压入到 61H 单元,PC 的高 8 位 PCH 压入到 62H,此时,SP 中的内容为 62H。

2. 数据指针 DPTR

数据指针 DPTR 是 1 个 16 位的 SFR,其高位字节寄存器用 DPH 表示,低位字节寄存器用 DPL 表示。DPTR 既可以作为 1 个 16 位寄存器 DPTR 来用,也可以作为 2 个独立的 8 位寄存器 DPH 和 DPL 来用。

3. I/O 端口 P0~P3

特殊功能寄存器 P0~P3 分别为 I/O 端口 P0~P3 的锁存器。

在 MCS-51 中,I/O 端口和 RAM 是统一编址的,使用起来较为方便。所有访问 RAM 单元的指令,都可用来访问 I/O 端口。

4. 寄存器 B

寄存器 B 是为执行乘法和除法操作设置的。

乘法中,ALU 的 2 个输入分别为 A、B,乘积存放在 BA 寄存器对中。B 中放乘积的高 8 位,A 中放乘积的低 8 位。

除法中,被除数取自 A,除数取自 B,商存放在 A 中,余数存放于 B。

在不执行乘、除法操作的情况下,可把它当作一个普通寄存器来使用。

5. 串行数据缓冲器 SBUF

串行数据缓冲器 SBUF 用于存放欲发送或已接收的数据,它在 SFR 块中只有一个字节地址,但物理上是由 2 个独立的寄存器组成,1 个是发送缓冲器,另 1 个是接收缓冲器,当要发送的数据传送到 SBUF 时,进的是发送缓冲器;接收外部来的数据时,接收到的数据存入接收缓冲器。

6. 定时器/计数器

MCS-51 单片机有 2 个 16 位定时器/计数器 T1 和 T0,它们各由 2 个独立的 8 位寄存器组成,共有 4 个独立的寄存器:TH1、TL1、TH0、TL0,可以分别对这 4 个寄存器进行字节寻址,但不能把 T1 或 T0 当作 1 个 16 位寄存器来寻址访问。

2.4.4 位地址空间

MCS-51 有一个功能很强的位处理器,指令系统中有着丰富的位操作指令(将在第3章中详细介绍),这些指令构成了位处理机的指令集。在 RAM 和 SFR 中共有 211 个寻址位的位地址,位地址范围为 00H~FFH,其中 00H~7FH 这 128 个位处于内部 RAM 字节地址 20H~2FH 单元中,如表 2-3 所示。其余的 83 个可寻址位分布在特殊功能寄存器 SFR 中,如表 2-4 所示。可位寻址的寄存器有 11 个,共有位地址 88 个,其中 5 个未用,其余 83 个位的位地址离散地分布于片内字节地址为 80H~FFH 的范围内,其最低的位地址等于其字节地址,并且其字节地址的末位都为 0H 或 8H。

表 2-3 MCS-51 内部 RAM 的可寻址位及其位地址

字节地址	位 地 址							
	D7	D6	D5	D4	D3	D2	D1	D0
2FH	7FH	7EH	7DH	7CH	7BH	7AH	79H	78H
2EH	77H	76H	75H	74H	73H	72H	71H	70H
2DH	6FH	6EH	6DH	6CH	6BH	6AH	69H	68H
2CH	67H	66H	65H	64H	63H	62H	61H	60H
2BH	5FH	5EH	5DH	5CH	5BH	5AH	59H	58H
2AH	57H	56H	55H	54H	53H	52H	51H	50H
29H	4FH	4EH	4DH	4CH	4BH	4AH	49H	48H
28H	47H	46H	45H	44H	43H	42H	41H	40H
27H	3FH	3EH	3DH	3CH	3BH	3AH	39H	38H
26H	37H	36H	35H	34H	33H	32H	31H	30H
25H	2FH	2EH	2DH	2CH	2BH	2AH	29H	28H
24H	27H	26H	25H	24H	23H	22H	21H	20H
23H	1FH	1EH	1DH	1CH	1BH	1AH	19H	18H
22H	17H	16H	15H	14H	13H	12H	11H	10H
21H	0FH	0EH	0DH	0CH	0BH	0AH	09H	08H
20H	07H	06H	05H	04H	03H	02H	01H	00H

表 2-4 SFR 中的位地址分布

特殊功能 寄存器符号	位 地 址								字节地址
	D7	D6	D5	D4	D3	D2	D1	D0	
B	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H	F0H
Acc	E7H	E6H	E5H	E4H	E3H	E2H	E1H	E0H	E0H

续表

特殊功能 寄存器符号	位 地 址								字节地址
	D7	D6	D5	D4	D3	D2	D1	D0	
PSW	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H	D0H
IP	—	—	—	BCH	BBH	BAH	B9H	B8H	B8H
P3	F7H	F6H	F5H	F4H	F3H	F2H	F1H	F0H	B0H
IE	AFH	—	—	ACH	ABH	AAH	A9H	A8H	A8H
P2	A7H	A6H	A5H	A4H	A3H	A2H	A1H	A0H	A0H
SCON	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	98H
P1	97H	96H	95H	94H	93H	92H	91H	90H	90H
TCON	8FH	8EH	8DH	8CH	8BH	8AH	89H	88H	88H
P0	87H	86H	85H	84H	83H	82H	81H	80H	80H

2.4.5 外部数据存储器

MCS-51 单片机内部有 128 B 的 RAM 作为数据存储器,当这 128 B 的 RAM 不够用时,则需要外扩数据存储器,MCS-51 最多可外扩 64 KB 的 RAM 或 I/O,这对很多应用场合已足够用。

至此,前面已详细地介绍了 MCS-51 的存储器结构。使用各类存储器,一定要注意以下几点:

(1) 地址的重叠性。数据存储器与程序存储器全部 64 KB 地址空间重叠;程序存储器中片内外低 4 KB 地址重叠;数据存储器中片内外最低的 128 B 地址重叠。虽然有这些重叠,但不会产生操作混乱。这是因为 MCS-51 单片机采用了不同的操作指令及 \overline{EA} 信号的控制选择来自动区分这些重叠的空间,用户不用担心重叠空间的地址冲突问题。

(2) 程序存储器(ROM)与数据存储器(RAM)在使用上是严格区分的,不同的操作指令不得混用。程序存储器只能放置程序指令及常数表格。除了程序的运行控制外,其操作指令不分片内和片外。而数据存储器则只存放数据,片内和片外的操作指令不同。

(3) 位地址空间共有 2 个区域,即片内 RAM 中的 20H~2FH 的 128 位,以及 SFR 中的位地址(其中有些位无定义)。这些位寻址单元与位指令集以及 PSW 中的 Cy 位构成了位处理器系统。

(4) 片外数据存储器区中,RAM 存储单元与 MCS-51 外部扩展的 I/O 端口统一编址。因此,应用系统中所有外围 I/O 端口的地址均占用 RAM 地址单元。MCS-51 在与外部扩展的 I/O 端口进行数据传送时,使用与访问外部数据存储

器相同的传送指令。

作为对 MCS-51 存储器结构的总结,图 2-6 为 MCS-51 中各类存储器的结构图。从图 2-6 中可以清楚地看出前面介绍的各类存储器在存储器空间的位置。

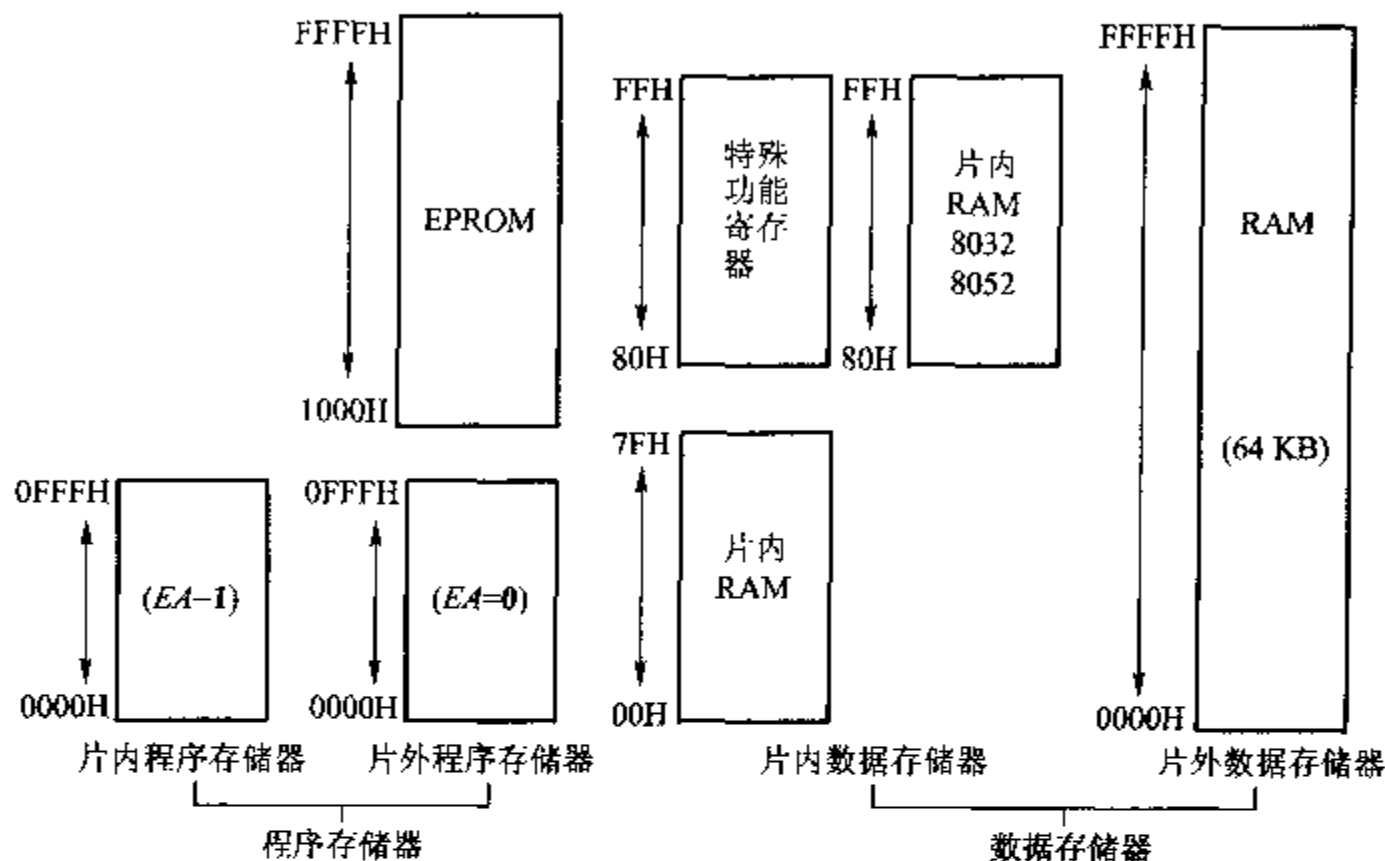


图 2-6 MCS-51 单片机的存储器结构

2.5 并行 I/O 端口

MCS-51 单片机共有 4 个双向的 8 位并行 I/O 端口 (Port), 分别记作 P0~P3, 共有 32 根口线, 端口的每一位均由锁存器、输出驱动器和输入缓冲器所组成。P0~P3 的端口寄存器属于特殊功能寄存器之列。这 4 个端口除了按字节寻址以外, 还可以按位寻址。由于它们在结构上有一些差异, 故各端口的性质和功能有一些差异。下面对这 4 个端口内部的结构和功能进行分析。

2.5.1 P0 端口

P0 口的字节地址为 80H, 位地址为 80H~87H。口的各位口线具有完全相同但又相互独立的逻辑电路, P0 口某一位的内部位结构的电路原理图如图 2-7 所示。

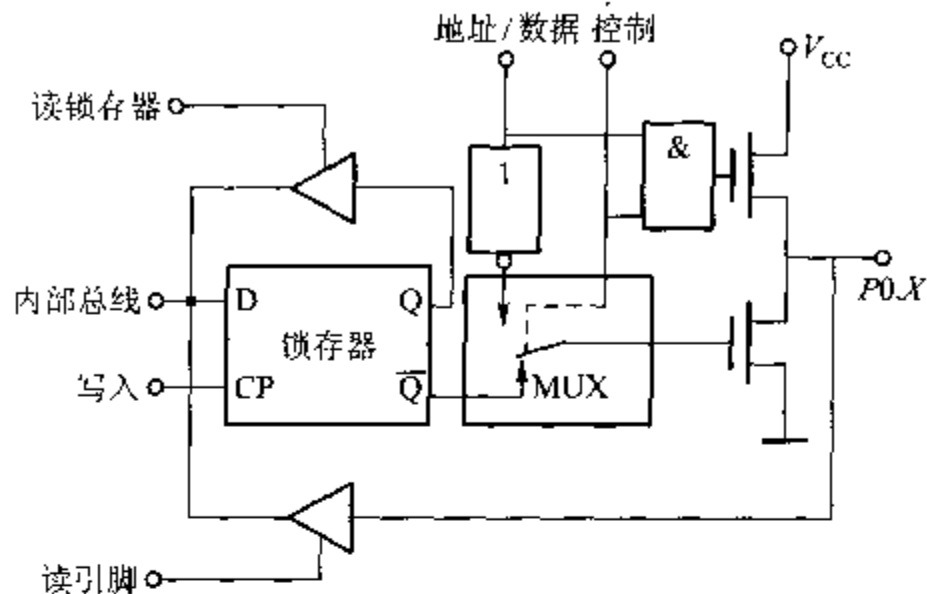


图 2-7 P0 口的位结构的电路原理图

P0 口某一位的电路包括:

- (1) 1 个数据输出锁存器,用于进行数据位的锁存。
- (2) 2 个三态的数据输入缓冲器,分别用于锁存器数据和引脚数据的输入缓冲。
- (3) 1 个多路的转接开关 MUX,开关的 1 个输入来自锁存器,另 1 个输入为“地址/数据”。输入转接由“控制”信号控制。设置多路转接开关的目的,是因为 P0 口既作为通用的 I/O 口,又可作为单片机系统的地址/数据线使用。即在控制信号的作用下,由 MUX 实现锁存器输出和地址/数据线之间的接通转接。
- (4) 数据输出的驱动和控制电路,由 2 只场效晶体管(FET)组成,上面的那只场效晶体管构成上拉电路。

P0 口在实际使用中,绝大多数情况下都是作为单片机系统的地址/数据线使用,当传送地址或数据时,CPU 发出控制信号,打开上面的与门,使多路转接开关 MUX 打向上边,使内部地址/数据线与下面的场效晶体管处于反相接通状态。这时的输出驱动电路由于上下两个 FET 处于反相,形成推拉式电路结构,大大的提高了负载能力。而当输入数据时,数据信号则直接从引脚通过输入缓冲器进入内部总线。

另外,P0 口也可作为通用的 I/O 口使用。这时,CPU 发来的“控制”信号为低电平,封锁了与门,并将输出驱动电路的上拉场效晶体管截止,而多路转接开关 MUX 打向下边,与 D 锁存器的 \bar{Q} 端接通。

当 P0 口作为输出口使用时,来自 CPU 的“写入”脉冲加在 D 锁存器的 CP 端,内部总线上的数据写入 D 锁存器,并向端口引脚 P0.X 输出。但要注意,由于输出电路是漏极开路(因为这时上拉场效晶体管截止),必须外接上拉电阻才能有高电平输出。

当 P0 口作为输入口使用时,应区分“读引脚”和“读端口”(或称“读锁存

器”)两种情况。为此,在口电路中有 2 个用于读入的三态缓冲器。所谓“读引脚”就是直接读取引脚 $P0.X$ 上的状态,这时由“读引脚”信号把下方缓冲器打开,引脚上的状态经缓冲器读入内部总线;而“读端口”则是“读锁存器”信号打开上面的缓冲器把锁存器 Q 端的状态读入内部总线。

2.5.2 P1 端口

P1 口的字节地址为 $90H$,位地址为 $90H \sim 97H$ 。P1 口某一位的内部位结构的电路原理图如图 2-8 所示。

P1 口只能作为通用的 I/O 口使用,所以在电路结构上与 P0 口有一些不同,主要有两点区别:

(1) 因为 P1 口只传送数据,所以不再需要多路转接开关 MUX。

(2) 由于 P1 口用来传送数据,因此输出电路中有上拉电阻,这样电路的输出不是三态的,所以 P1 口是准双向口。

因此:

(1) P1 口作为输出口使用时,与 P0 口不同的是,外电路无需再接上拉电阻。

(2) P1 口作为输入口使用时,应先向其锁存器先写入 1,使输出驱动电路的 FET 截止。

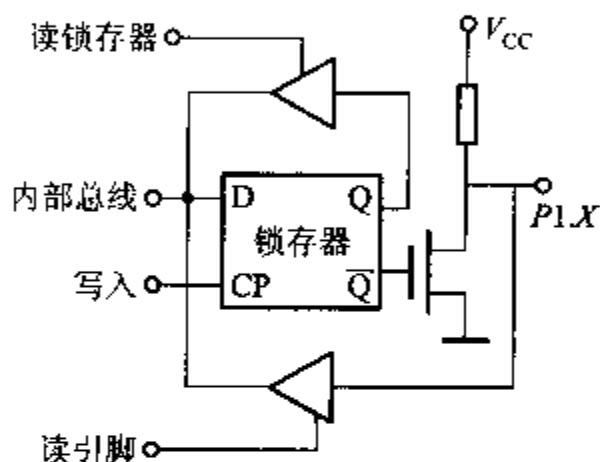


图 2-8 P1 口的位结构的电路原理图

2.5.3 P2 端口

P2 口的字节地址为 $A0H$,位地址为 $A0H \sim A7H$ 。P2 口某一位的位结构的电路原理图如图 2-9 所示。

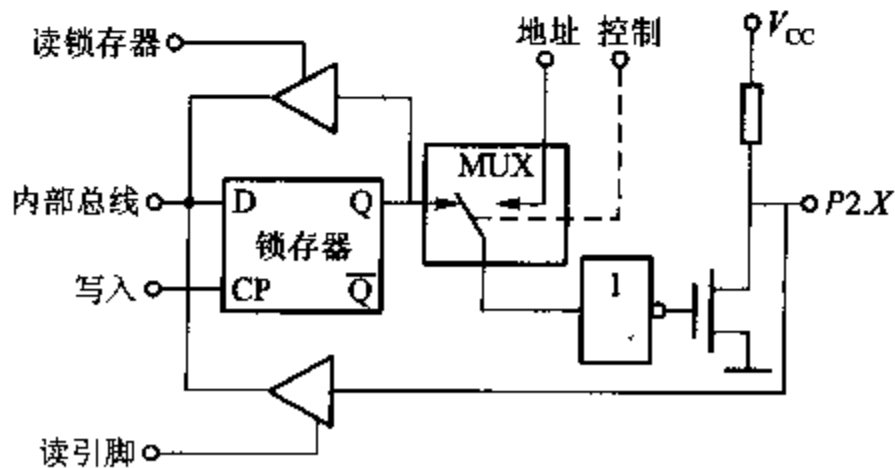


图 2-9 P2 口的位结构的电路原理图

在实际应用中,因为 P2 口用于为系统提供高位地址,因此同 P0 口一样,在口电路中有一个多路转接开关 MUX。但 MUX 的 1 个输入端不再是“地址/数据”,而是单一的“地址”,这是因为 P2 口只作为地址线使用。当 P2 口用作为高位地址线使用时,多路转接开关应接向“地址”端。正因为只作为地址线使用,口的输出用不着是三态的,所以,P2 口也是一个准双向口。

此外,P2 口也可以作为通用 I/O 口使用,这时,多路转接开关接向锁存器 Q 端。

2.5.4 P3 端口

P3 口的字节地址为 B0H,位地址为 B0H~B7H。P3 口某一位的位结构的电路原理图如图 2-10 所示。

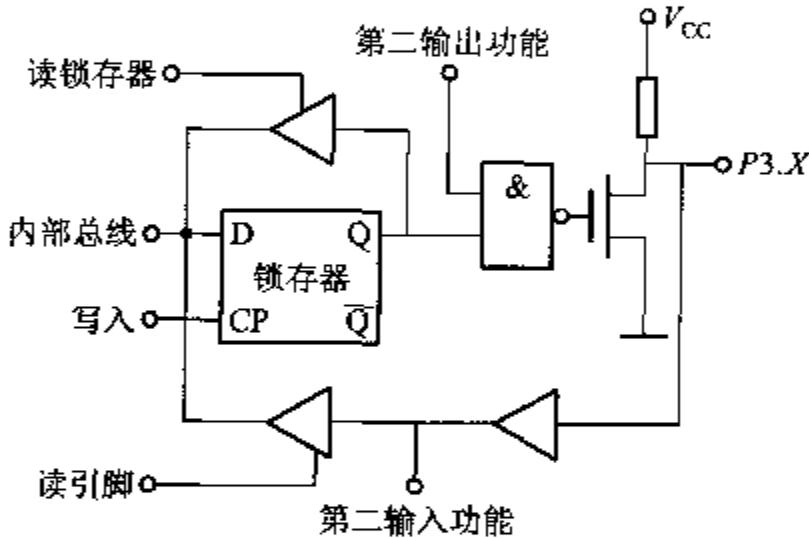


图 2-10 P3 口的位结构的电路原理图

由于 MCS-51 的引脚数目有限,因此在 P3 口电路中增加了引脚的第二功能。P3 口的某一引脚既可以作为通用 I/O 使用,又可根据需要,使用它的第二功能。表 2-5 列出了 P3 口的第二功能定义,读者应熟记。

表 2-5 P3 口的第二功能定义

口 引 脚	第 二 功 能
P3.0	RXD(串行输入口)
P3.1	TXD(串行输出口)
P3.2	$\overline{\text{INT0}}$ (外部中断 0)
P3.3	$\overline{\text{INT1}}$ (外部中断 1)
P3.4	T0(定时器 0 外部计数输入)
P3.5	T1(定时器 1 外部计数输入)
P3.6	$\overline{\text{WR}}$ (外部数据存储器写选通)
P3.7	$\overline{\text{RD}}$ (外部数据存储器读选通)

由于第二功能信号有输出和输入两类,因此,分两种情况进行说明。

(1) 对于作为第二功能输出的引脚,当作为通用的 I/O 口使用时,电路中的“第二输出功能”线应保持高电平,与非门开通,以使锁存器的 Q 端输出通路保持畅通。当输出第二功能信号,该锁存器应预先置 1,使与非门对“第二输出功能”信号的输出是畅通的,从而实现第二功能信号的输出。

(2) 对于作为第二功能输入的引脚,在口线引脚的内部增加了 1 个缓冲器,输入的信号就从这个缓冲器的输出端取得。而作为通用的 I/O 口线使用的数据输入,仍取自三态缓冲器的输出端。总的来说,P3 口无论是作为输入口使用还是第二功能信号的输入,锁存器输出和“第二输出功能”线都应保持高电平。

2.5.5 P0~P3 端口功能总结

前面介绍了 MCS-51 的 P0~P3 口的内部电路和功能,下面把这些口在使用中一些应注意的问题归纳如下。

(1) P0~P3 口都是并行 I/O 口,都可用于数据的输入和输出,但 P0 口和 P2 口除了可进行数据的输入/输出外,通常用来构建系统的数据总线和地址总线,所以在电路中有 1 个多路转接开关 MUX,以便进行 2 种用途的转换。而 P1 口和 P3 口没有构建系统的数据总线和地址总线的功能,因此,在电路中没有多路转接开关 MUX。由于 P0 口可作为地址/数据复用线使用,需传送系统的低 8 位地址和 8 位数据,因此 MUX 的 1 个输入端为“地址/数据”信号。而 P2 口仅作为高位地址线使用,不涉及数据,所以 MUX 的 1 个输入信号为“地址”。

(2) 在 4 个口中只有 P0 口是 1 个真正的双向口,P1~P3 这 3 个口都是准双向口。原因是在应用系统中,P0 口作为系统的数据总线使用时,为保证数据的正确传送,需要解决芯片内外的隔离问题,即只有在数据传送时芯片内外才接通;不进行数据传送时,芯片内外应处于隔离状态。为此,要求 P0 口的输出缓冲器是 1 个三态门。

在 P0 口中输出三态门是由 2 只场效晶体管(FET)组成,所以说它是 1 个真正的双向口。而其它的 3 个口 P1~P3 中,上拉电阻代替 P0 口中的场效晶体管,输出缓冲器不是三态的,因此不是真正的双向口,只能称其为准双向口。

(3) P3 口的口线具有第二功能,为系统提供一些控制信号。因此在 P3 口电路增加了第二功能控制逻辑。这是 P3 口与其它各口的不同之处。

2.6 时钟电路与时序

时钟电路用于产生 MCS-51 单片机工作时所必需的时钟控制信号。MCS-51 单片机的内部电路在时钟信号控制下,严格地按时序执行指令进行工作。而时序所研究的是指令执行中各个信号在时间上的关系。

在执行指令时,CPU 首先要到程序存储器中取出需要执行的指令操作码,然后译码,并由时序电路产生一系列控制信号去完成指令所规定的操作。CPU 发出的时序信号有两类,一类用于片内对各个功能部件的控制,这类信号很多,但用户无需了解。另一类用于对片外存储器或 I/O 端口的控制,这部分时序对于分析、设计硬件接口电路至关重要。这也是单片机应用系统设计者普遍关心和重视的问题。

2.6.1 时钟电路

MCS-51 单片机各功能部件的运行都是以时钟控制信号为基准,有条不紊地一拍一拍地工作。因此,时钟频率直接影响单片机的速度,时钟电路的质量也直接影响单片机系统的稳定性。常用的时钟电路设计有两种方式,一种是内部时钟方式,另一种方式为外部时钟方式。

1. 内部时钟方式

MCS-51 内部有一个用于构成振荡器的高增益反相放大器,该高增益反相放大器的输入端为芯片引脚 XTAL1,输出端为引脚 XTAL2。这两个引脚跨接石英晶体振荡器(简称晶振)和微调电容,就构成一个稳定的自激振荡器,图 2-11 是 MCS-51 内部时钟方式的振荡器电路。

电路中的电容 C_1 和 C_2 典型值通常选择为 30 pF 左右。对外接电容的值虽然没有严格的要求,但电容的大小会影响振荡器频率的高低、振荡器的稳定性和起振的快速性。晶振的振荡频率的范围通常是在 1.2 MHz~12 MHz 之间。晶振的频率越高,则系统的时钟频率也就越高,单片机的运行速度也就越快。但反过来运行速度快对存储器的速度要求就高,对印制电路板(也称印刷电路板)的工艺要求也高,即要求线间的寄生电容要小;晶振和电容应尽可能安装得与单片机芯片靠

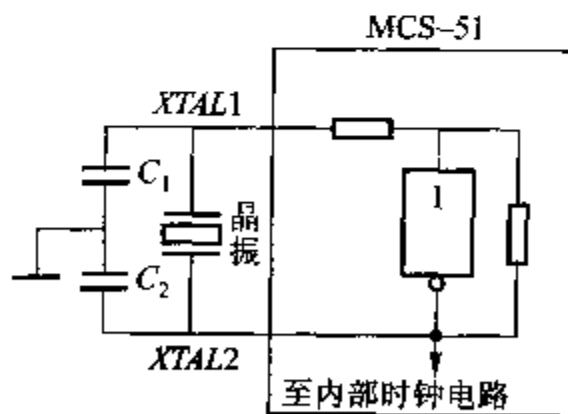


图 2-11 MCS-51 内部时钟方式的电路

近,以减少寄生电容,更好地保证振荡器稳定、可靠地工作。为了提高温度稳定性,应采用温度稳定性能好的电容。

MCS-51 常选择振荡频率 6 MHz 或 12 MHz 的石英晶体。随着集成电路制造工艺技术的发展,单片机的时钟频率也在逐步提高,现在的某些高速单片机芯片的时钟频率已达 40 MHz。

2. 外部时钟方式

外部时钟方式是使用外部振荡脉冲信号,常用于多片 MCS-51 单片机同时工作,以便于多片 MCS-51 单片机之间的同步,一般为低于 12 MHz 的方波。

外部的时钟源直接接到 XTAL2 端,通过 XTAL2 端输入到片内的时钟发生器上。电路如图 2-12 所示。由于 XTAL2 的逻辑电平不是 TTL 的,故建议外接 1 个 4.7~10 kΩ 的上拉电阻。

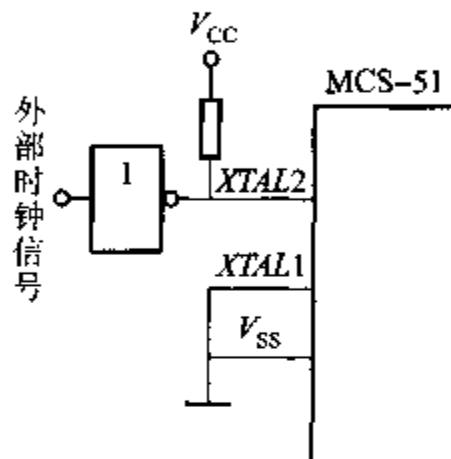


图 2-12 MCS-51 的外部时钟方式电路

3. 时钟信号的输出

当使用片内振荡器时,XTAL1、XTAL2 引脚还能为应用系统中的其它芯片提供时钟,但需增加驱动能力。其引出的方式有两种,如图 2-13 所示。

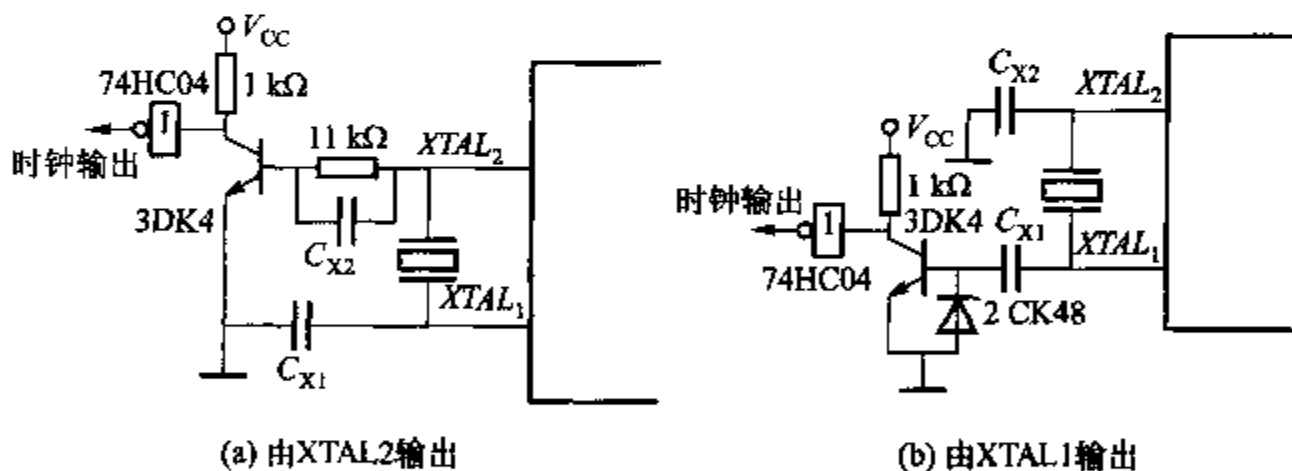


图 2-13 时钟信号的输出

2.6.2 机器周期、指令周期与指令时序

单片机执行的指令均是在 CPU 控制器的时序控制电路的控制下进行的,各种时序均与时钟周期有关。

1. 时钟周期

时钟周期是单片机的基本时间单位。若时钟晶振的振荡频率为 f_{osc} ,则时钟周期 $T_{osc} = 1/f_{osc}$ 。如 $f_{osc} = 6 \text{ MHz}$, $T_{osc} = 166.7 \text{ ns}$ 。

2. 机器周期

CPU 完成一个基本操作所需要的时间称为机器周期。单片机中常把执行一条指令的过程分为几个机器周期。每个机器周期完成一个基本操作,如取指令、读或写数据等等。MCS-51 单片机每 12 个时钟周期为 1 个机器周期,即 $T_{cy} = 12 / f_{osc}$ 。若 $f_{osc} = 6 \text{ MHz}$, $T_{cy} = 2 \mu\text{s}$; $f_{osc} = 12 \text{ MHz}$, $T_{cy} = 1 \mu\text{s}$ 。

MCS-51 的 1 个机器周期包括 12 个时钟周期,分为 6 个状态: S1~S6。每个状态又分为 2 拍: P1 和 P2。因此,1 个机器周期中的 12 个时钟周期表示为: S1P1、S1P2、S2P1、S2P2、...、S6P2,如图 2-14 所示。

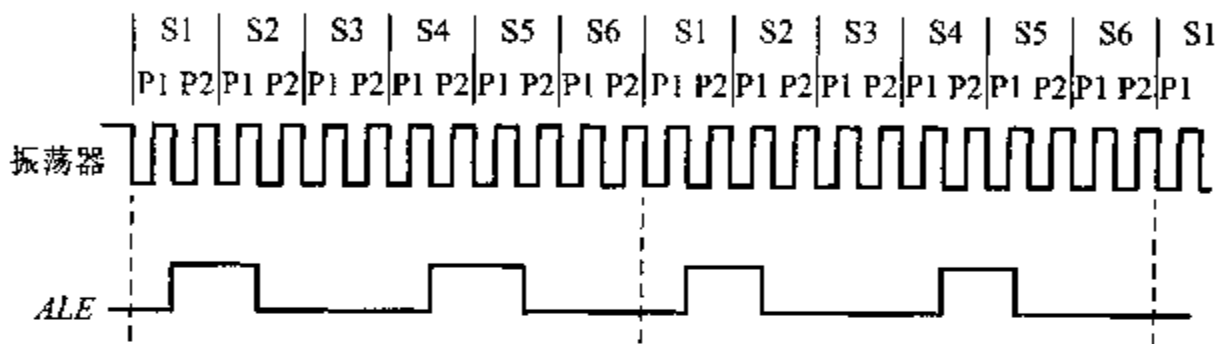


图 2-14 MCS-51 的机器周期

3. 指令周期

指令周期是执行一条指令所需的时间。MCS-51 单片机中按字节可分为单字节、双字节、三字节指令。因此执行一条指令的时间也不同。对于简单的单字节指令,取出指令立即执行,只需 1 个机器周期的时间。而有些复杂的指令,如转移、乘、除指令则需 2 个或多个机器周期。

从指令的执行速度看,单字节和双字节指令一般为单机器周期和双机器周期,三字节指令都是双机器周期,只有乘、除指令占用 4 个机器周期。

4. 指令时序

MCS-51 单片机执行任何一条指令时,都可以分为取指令阶段和指令执行阶段。单片机在取指令阶段,可以把程序计数器 PC 中地址送到程序存储器,并从中取出需要执行指令的操作码和操作数。指令执行阶段可对指令操作码进行译码,以产生一系列控制信号完成指令的执行。

图 2-14 中的 ALE 信号是为地址锁存而定义的,该信号每有效 1 次,则对应 MCS-51 的 1 次读指令的操作。ALE 信号以时钟脉冲 1/6 的频率出现,因此在 1 个机器周期中,ALE 信号 2 次有效(但要注意,在执行访问外部数据存储器的指令 MOVX 时,将会丢失 1 个 ALE 脉冲,将在第 8 章介绍),第 1 次在 S1P2 和 S2P1 期间,第 2 次在 S4P2 和 S5P1 期间,有效宽度为 1 个状态周期。

2.7 复位操作和复位电路

2.7.1 复位操作

复位是单片机的初始化操作,只需给 MCS-51 的复位引脚 RST 加上大于 2 个机器周期(即 24 个时钟振荡周期)的高电平就可使 MCS-51 复位。复位时,PC 初始化为 0000H,使 MCS-51 单片机从 0000H 单元开始执行程序。除了进入系统的正常初始化之外,当由于程序运行出错或操作错误使系统处于死锁状态,为摆脱死锁状态,也需按复位键使 RST 引脚为高电平使 MCS-51 重新启动。

除 PC 之外,复位操作还对其它一些寄存器有影响,这些寄存器复位时的状态如表 2-6 所示。由表中可以看出,复位时,SP=07H,而 4 个 I/O 端口 P0~P3 的引脚均为高电平,这在某些控制应用中,要考虑 P0~P3 引脚的高电平对接在这些引脚上的外部电路的影响。

表 2-6 复位时片内各寄存器的状态

寄 存 器	复 位 状 态	寄 存 器	复 位 状 态
PC	0000H	TMOD	00H
Acc	00H	TCON	00H
PSW	00H	TH0	00H
B	00H	TL0	00H
SP	07H	TH1	00H
DPTR	0000H	TL1	00H
P0~P3	FFH	SCON	00H
IP	×××00000B	SBUF	××××××××B
IE	0××00000B	PCON	0×××0000B

由于单片机内部的各个功能部件均受特殊功能寄存器控制,程序运行直接受程序计数器 PC 的控制。表 2-6 中各寄存器复位时的状态决定了单片机内有关功能部件的初始状态。

另外,在复位有效期间(即复位引脚为高电平期间),MCS-51 的 ALE 引脚和 $\overline{\text{PSEN}}$ 引脚均为高电平,且内部 RAM 的状态不受复位的影响。

2.7.2 复位电路

MCS-51 的复位是由外部的复位电路来实现的。MCS-51 片内复位结构如图 2-15 所示。

复位引脚 RST 通过一个施密特触发器与复位电路相连,施密特触发器用来抑制噪声,在每个机器周期的 S5P2,施密特触发器的输出电平由复位电路采样一次,然后才能得到内部复位操作所需要的信号。

复位电路通常采用上电自动复位和按钮复位两种方式。

最简单的上电自动复位电路如图 2-16 所示。上电自动复位是通过外部复位电路的电容充电来实现的。当电源接通时只要 V_{CC} 的上升时间不超过 1 ms,就可以实现自动上电复位。当时钟频率选用 6 MHz 时, C 取 $22\ \mu\text{F}$, R 取 $1\ \text{k}\Omega$ 。

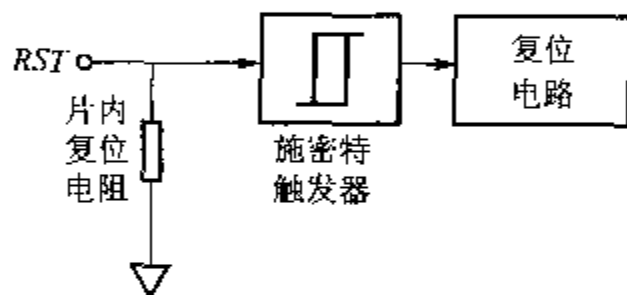


图 2-15 MCS-51 的片内复位结构

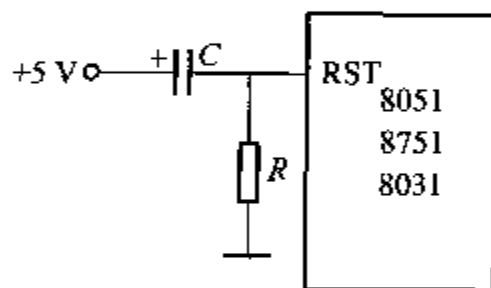


图 2-16 上电复位电路

除了上电复位外,有时还需要按键手动复位。按键手动复位有电平方式和脉冲方式两种。其中电平复位是通过 RST 端经电阻与电源 V_{CC} 接通而实现的,按键手动电平复位电路如图 2-17 所示。当时钟频率选用 6 MHz 时, C 取 $22\ \mu\text{F}$, R_S 取 $200\ \Omega$, R_K 取 $1\ \text{k}\Omega$ 。按键脉冲复位则是利用 RC 微分电路产生的正脉冲来实现的,脉冲复位电路如图 2-18 所示。图中的阻容参数适用于 6 MHz 时钟。

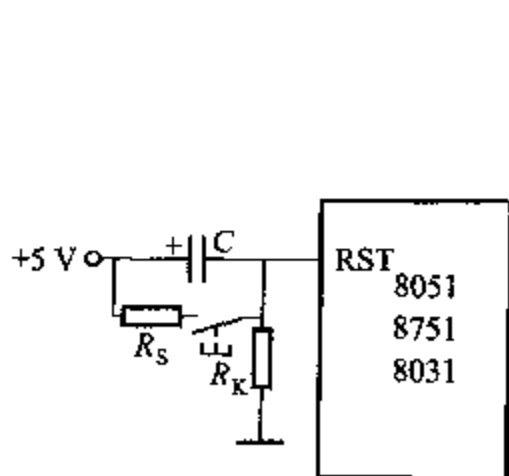


图 2-17 按键电平复位电路

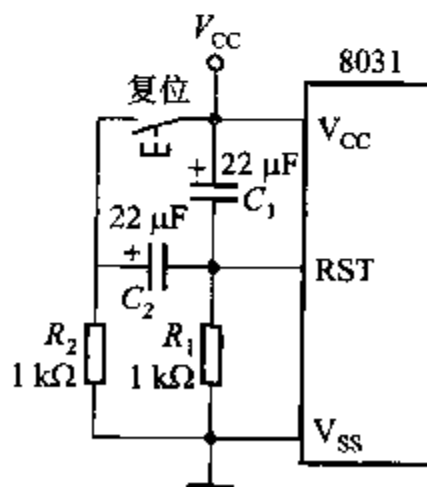


图 2-18 按键脉冲复位电路

图 2-19 为两种实用的兼有上电复位与按钮复位的电路。

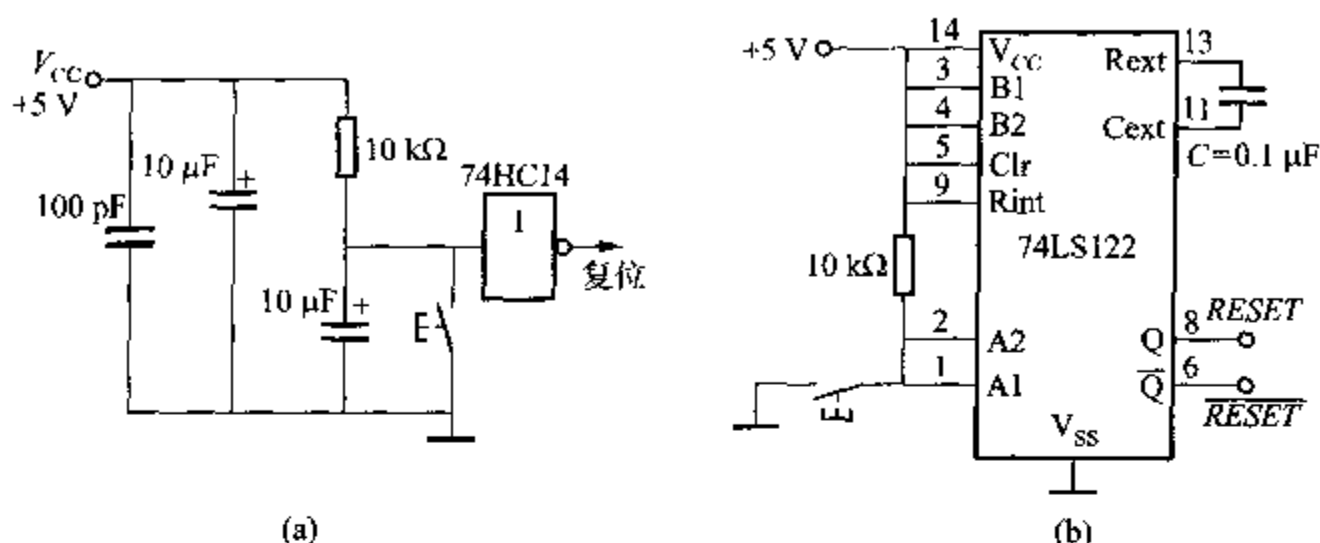


图 2-19 两种实用的兼有上电复位与按钮复位的电路

图 2-19 中(b)的电路能输出高、低两种电平的复位控制信号,以适应外围 I/O 接口芯片所要求的不同复位电平信号。图 2-19(b)中 74LS122 为单稳电路,实验表明,电容 C 的选择约为 $0.1 \mu\text{F}$ 较好。

在实际的应用系统设计中,若有外部扩展的 I/O 接口电路也需初始复位,如果它们的复位端和 MCS-51 的复位端相连,复位电路中的 R 、 C 参数要受到影响,这时复位电路中的 R 、 C 参数要统一考虑,以保证可靠的复位。如果单片机 MCS-51 与外围 I/O 接口电路的复位电路和复位时间不完全一致,使单片机初始化程序不能正常运行,外围 I/O 接口电路的复位也可以不和 MCS-51 复位端相连,仅采用独立的上电复位电路。若 RC 上电复位电路接施密特电路输入端,施密特电路输出接 MCS-51 和外围电路复位端,则能使系统可靠地同步复位。一般来说,单片机的复位速度比外围 I/O 接口电路快些。为保证系统可靠复位,在初始化程序中应安排一定的复位延迟时间。

思考题及习题

1. MCS-51 单片机的片内都集成了哪些功能部件? 各个功能部件的最主要的功能是什么?
2. 说明 MCS-51 单片机的引脚 $\overline{\text{EA}}$ 的作用,该引脚接高电平和接低电平时各有何种功能?
3. MCS-51 的时钟振荡周期和机器周期之间有何关系?
4. 在 MCS-51 单片机中,如果采用 6 MHz 晶振,1 个机器周期为()。
5. 程序存储器的空间里,有 5 个单元是特殊的,这 5 个单元对应 MCS-51 单片机 5 个中断源的中断入口地址,请写出这些单元的地址以及对应的中断源。
6. 内部 RAM 中,位地址为 30H 的位,该位所在字节的字节地址为()。

7. 若 A 中的内容为 63H,那么,P 标志位的值为()。
8. 判断下列说法是否正确:
 - (A) 8031 的 CPU 是由 RAM 和 EPROM 所组成。
 - (B) 区分片外程序存储器和片外数据存储器的最可靠的方法是看其位于地址范围的低端还是高端。
 - (C) 在 MCS-51 中,为使准双向的 I/O 口工作在输入方式,必须保证它被事先预置为 1。
 - (D) PC 可以看成是程序存储器的地址指针。
9. 8031 单片机复位后,R4 所对应的存储单元的地址为(),因上电时 PSW=()。这时当前的工作寄存器区是()组工作寄存器区。
10. 什么是机器周期? 1 个机器周期的时序是如何来划分的? 如果采用 12 MHz 晶振,1 个机器周期为多长时间?
11. 判断以下有关 PC 和 DPTR 的结论是否正确?
 - (A) DPTR 是可以访问的,而 PC 不能访问。
 - (B) 它们都是 16 位的寄存器。
 - (C) 它们都具有加 1 的功能。
 - (D) DPTR 可以分为 2 个 8 位的寄存器使用,但 PC 不能。
12. 内部 RAM 中,哪些单元可作为工作寄存器区,哪些单元可以进行位寻址? 写出它们的字节地址。
13. 使用 8031 单片机时,需将 \overline{EA} 引脚接()电平,因为其片内无()存储器。
14. 片内 RAM 低 128 个单元划分为哪 3 个主要部分? 各部分的主要功能是什么?
15. 判断下列说法是否正确
 - (A) 程序计数器 PC 不能为用户编程时直接使用,因为它没有地址。
 - (B) 内部 RAM 的位寻址区,只能供位寻址使用,而不能供字节寻址使用。
 - (C) 8031 共有 21 个特殊功能寄存器,它们的位都是可用软件设置的,因此,是可以进行位寻址的。
16. PC 的值是:
 - (A) 当前正在执行指令的前一条指令的地址
 - (B) 当前正在执行指令的地址
 - (C) 当前正在执行指令的下一条指令的地址
 - (D) 控制器中指令寄存器的地址
17. 通过堆栈操作实现子程序调用,首先就要把()的内容入栈,以进行断点保护。调用返回时,再进行出栈保护,把保护的断点送回到()。
18. 写出 P3 口各引脚的第二功能。
19. MCS-51 单片机程序存储器的寻址范围是由程序计数器 PC 的位数所决定的,因为 MCS-51 的 PC 是 16 位的,因此其寻址的范围为()KB。
20. 当 MCS-51 单片机运行出错或程序陷入死循环时,如何来摆脱困境?
21. 判断下列说法是否正确?
 - (A) PC 是 1 个不可寻址的特殊功能寄存器。
 - (B) 单片机的主频越高,其运算速度越快。

(C) 在 MCS-51 单片机中,1 个机器周期等于 $1\mu\text{s}$ 。

(D) 特殊功能寄存器 SP 内装的是栈顶首地址单元的内容。

22. 如果手中仅有一台示波器,可通过观察哪个引脚的状态,来大致判断 MCS-51 单片机正在工作?

第 3 章 MCS-51 的指令系统

MCS-51 单片机所能执行的命令(指令)的集合就是它的指令系统。指令常以其英文名称或缩写形式来作为助记符,以助记符、符号地址、标号等书写程序的语言称为汇编语言。本章所介绍的是 MCS-51 汇编语言的指令系统。

3.1 指令系统概述

MCS-51 指令系统是一种简明易掌握、效率较高的指令系统。

MCS-51 的基本指令共 111 条,按指令在程序存储器所占的字节来分,其中:

- (1) 单字节指令 49 条。
- (2) 双字节指令 45 条。
- (3) 三字节指令 17 条。

按指令的执行时间来分,其中:

- (1) 1 个机器周期(12 个时钟振荡周期)的指令 64 条。
- (2) 2 个机器周期(24 个时钟振荡周期)的指令 45 条。
- (3) 只有乘、除两条指令的执行时间为 4 个机器周期(48 个时钟振荡周期)。

在 12 MHz 晶振的条件下,每个机器周期为 $1\mu\text{s}$,由此可见,MCS-51 指令系统对存储空间和时间的利用率较高。

MCS-51 单片机的一大特点是在硬件结构中有一个位处理机,对应这个位处理机,指令系统中相应地设计了一个处理位变量的指令子集,这个子集在进行位变量处理的程序设计时十分有效、方便。

3.2 指令格式

指令的表示方法称为指令格式,一条指令通常由两部分组成,即操作码和操

作数。操作码用来规定指令进行什么操作,而操作数则是指令操作的对象。操作数可能是一个具体的数据,也可能是指出到哪里取得数据的地址或符号。在 MCS-51 指令系统中,有单字节、双字节、三字节这些不同长度的指令,指令长度不同,指令的格式也就不同。

(1) 单字节指令:指令只有 1 B,操作码和操作数同在一个字节中。

(2) 双字节指令:双字节指令包括 2 B。其中一个字节为操作码,另一个字节是操作数。

(3) 三字节指令:在三字节指令中,操作码占 1 B,操作数占 2 B。其中操作数既可能是数据,也可能是地址。

3.3 指令系统的寻址方式

大多数指令执行时,都需要使用操作数。寻址方式就是在指令中说明操作数所在地址的方法。一般说来,寻址方式越多,单片机的功能就越强,灵活性越大,指令系统也就越复杂。MCS-51 单片机的指令系统有以下 7 种寻址方式。下面分别予以介绍。

1. 寄存器寻址方式

寄存器寻址方式就是操作数在寄存器中,因此指定了寄存器就能得到操作数。在寄存器寻址方式的指令中以符号名称来表示寄存器。例如指令:

MOV A,Rn ;(Rn)→A,n=0~7

表示把寄存器 Rn 的内容传送给累加器 A,由于操作数在 Rn 中,因此在指令中指定了从寄存器 Rn 中取得源操作数,所以就称为寄存器寻址方式。

寄存器寻址方式的寻址范围包括:

(1) 4 组通用工作寄存器区共 32 个工作寄存器。但只能寻址当前的工作寄存器区的 8 个工作寄存器,因此指令中的寄存器的名称只能是 R0~R7。

(2) 部分特殊功能寄存器,例如累加器 A、寄存器 B 以及数据指针寄存器 DPTR 等。

2. 直接寻址方式

在这种寻址方式中,指令中操作数直接以单元地址的形式给出。该单元地址中的内容就是操作数。例如指令:

MOV A, 40H

表示把内部 RAM 40H 单元的内容传送给 A。源操作数采用的是直接寻址方式。

直接寻址的操作数在指令中以存储单元的形式出现,因为直接寻址方式只

能使用 8 位二进制数表示的地址,因此,直接寻址方式的寻址范围只限于:

(1) 内部 RAM 的 128 个单元。

(2) 特殊功能寄存器。特殊功能寄存器除了以单元地址的形式外,还可以用寄存器符号的形式给出。例如:MOV A,80H 表示把 P0 口(地址为 80H)的内容传送给 A。也可写为:MOV A,P0 这也表示把 P0 口(地址为 80H)的内容传送给 A,两条指令是等价的。

应当说明的是,直接寻址方式是访问特殊功能寄存器的惟一寻址方式。

3. 寄存器间接寻址方式

寄存器寻址方式,就是寄存器中存放的是操作数,而寄存器间接寻址方式,就是寄存器中存放的是操作数的地址,即先从寄存器中找到操作数的地址,再按该地址找到操作数。由于操作数是通过寄存器间接得到的,因此称为寄存器间接寻址。

寄存器间接寻址也需要以寄存器符号的形式表示,为了区别寄存器寻址和寄存器间接寻址,在寄存器间接寻址方式中,应在寄存器的名称前面加前缀标志“@”。

访问内部 RAM 或外部数据存储器的低 256 B 时,只能采用 R0 或 R1 作为间址寄存器。例如指令:

MOV A,@R_i ;i=0 或 1

其中 R_i 中的内容为 40H,即从 R_i 中找到源操作数所在单元的地址 40H,把该地址中的内容传送给 A,即把内部 RAM 中 40H 单元的内容送到 A。这类指令为单字节指令,其指令代码中最低位是表示采用 R0 还是 R1 作为间接寻址寄存器。

寄存器间接寻址方式的寻址范围:

(1) 访问内部 RAM 低 128 个单元,其通用形式为 @R_i。

(2) 对片外数据存储器的 64 KB 的间接寻址,只能使用 DPTR 作间接寻址寄存器,其形式为 @DPTR。例如:MOVX A,@DPTR,其功能是把 DPTR 指定的外部 RAM 单元的内容送累加器 A。

(3) 片外数据存储器的低 256 B,除可使用 DPTR 作为间址寄存器外,也可使用 R0 或 R1 作间址寄存器。例如:MOVX A,@R_i,其功能是把 @R_i 指定的外部 RAM 单元的内容送累加器 A。

(4) 堆栈区:堆栈操作指令 PUSH(压栈)和 POP(出栈),使用堆栈指针(SP)作间址寄存器来进行对堆栈区的间接寻址。

4. 立即寻址方式

立即寻址方式就是操作数在指令中直接给出。出现在指令中的操作数即为立即数。为了与直接寻址指令中的直接地址相区别,需在操作数前面加前缀标

志“=”。例如指令:

MOV A, #40H

表示把立即数 40H 送给 A。40H 这个常数是指令代码的一部分。采用立即寻址方式的指令是双字节的。第一个字节是操作码,第二字节是立即操作数。因此,操作数就是放在程序存储器内的常数。

5. 基址寄存器加变址寄存器间接寻址方式

这种寻址方式用于读出程序存储器中的数据到累加器中。本寻址方式是以 DPTR 或 PC 作基址寄存器,以累加器 A 作为变址寄存器,并以两者内容相加形成的 16 位地址作为操作数的地址,以达到访问数据表格的目的。例如:指令 MOVC A, @A+DPTR 其中 A 的原有内容为 05H, DPTR 的内容为 0400H,该指令执行的结果是把程序存储器 0405H 单元的内容传送给 A。

下面对本寻址方式作如下说明:

(1) 本寻址方式只能对程序存储器进行寻址,或者说它是专门针对程序存储器的寻址方式,寻址范围可达到 64 KB。

(2) 本寻址方式的指令只有 3 条:

MOVC A, @A+DPTR

MOVC A, @A+PC

JMP @A+DPTR

其中前两条指令是读程序存储器指令,最后一条指令是无条件转移指令。这 3 条指令都是单字节指令。

6. 位寻址方式

MCS-51 有位处理功能,可以对数据位进行操作,因此就有相应的位寻址方式。位寻址指令中可以直接使用位地址,例如:

MOV C, 40H

指令的功能是把位 40H 的值送到进位位 C。

位寻址的寻址范围包括:

(1) 内部 RAM 中的位寻址区

单元地址为 20H~2FH,共 16 个单元,128 个位,位地址是 00H~7FH,对这 128 个位的寻址使用直接地址表示。位寻址区中的位有两种表示方法,一种是位地址,例如,40H;另一种是单元地址加上位,例如,(28H).0,指的是 28H 单元中的最低位。位 40H 与位 (28H).0 是同一个位,它们是等价的。

(2) 特殊功能寄存器中的可寻址位

可供位寻址的特殊功能寄存器有 11 个,共有 88 个位,其中有 5 个位没有定义,所以有可寻址位 83 个。这些寻址位在指令中有如下 4 种的表示方法:

① 直接使用位地址。例如 PSW 寄存器位 5 的位地址为 0D5H。

② 位名称的表示方法。例如 PSW 寄存器位 5 是 F0 标志位,则可使用 F0 表示该位。

③ 单元地址加位数的表示方法。例如 0D0H 单元(即 PSW 寄存器)位 5,表示为(0D0H).5。

④ 特殊功能寄存器符号加位数的表示方法。例如 PSW 寄存器的位 5 表示为 PSW.5。

7. 相对寻址方式

相对寻址方式是为解决程序转移而专门设置的,为转移指令所采用。在 MCS-51 的指令系统中,有多条相对转移指令,这些指令多为双字节指令,但也有个别为三字节的。

在相对寻址的转移指令中,给出了地址偏移量,以“rel”表示,即把 PC 的当前值加上偏移量就构成了程序转移的目的地址。但这里的 PC 的当前值是指执行完该指令后的 PC 值,即转移指令的 PC 值加上它的字节数。因此,转移的目的地址可用如下公式表示:

目的地址 = 转移指令所在的地址 + 转移指令的字节数 + rel

偏移量 rel 是 1 个带符号的 8 位二进制数补码数,所能表示的数的范围是: -128~+127。因此,相对转移是以转移指令所在地址为基点,向地址增加方向最大可转移(127+转移指令字节)个单元地址,向地址减少方向最大可转移(128-转移指令字节)个单元地址。

以上介绍了 MCS-51 指令系统的 7 种寻址方式,概括起来如表 3-1 所示。

表 3-1 寻址方式及寻址空间

序号	寻址方式	使用的变量	寻址空间
1	寄存器寻址	R0~R7、A、B、C(位)、DPTR、AB	4 组通用工作寄存器部分特殊功能寄存器
2	直接寻址		内部 RAM 128 B 特殊功能寄存器
3	寄存器间接寻址	@R1, @R0, SP	片内 RAM
		@R0, @R1, @DPTR	片外数据存储器
4	立即寻址	#data	
5	基址寄存器加变址寄存器间接寻址	@A+DPTR, @A+PC	程序存储器
6	位寻址		内部 RAM20H~2FH 单元的 128 个可寻址位、SFR 中的可寻址位
7	相对寻址	PC+偏移量	程序存储器

3.4 MCS-51 指令系统分类介绍

MCS-51 指令系统共有 111 条指令,按功能分类,可分为下面 5 大类:

- (1) 数据传送类(28 条)。
- (2) 算术操作类(24 条)。
- (3) 逻辑运算类(25 条)。
- (4) 控制转移类(17 条)。
- (5) 位操作类(17 条)。

位操作类指令由位处理机执行。

在分类介绍指令之前,先把描述指令的一些符号的意义,加以简单的介绍:

R_n	当前选中的寄存器区的 8 个工作寄存器 $R_0 \sim R_7 (n=0 \sim 7)$ 。
R_i	当前选中的寄存器区中可作间接寻址寄存器的 2 个寄存器 R_0 、 $R_1 (i=0,1)$ 。
<i>direct</i>	直接地址,即 8 位的内部数据存储器单元或特殊功能寄存器的地址。
<i>#data</i>	包含在指令中的 8 位立即数。
<i>#data16</i>	包含在指令中的 16 位立即数。
<i>rel</i>	相对转移指令中的偏移量,为 8 位的带符号补码数。
<i>DPTR</i>	数据指针,可用作 16 位的地址寄存器。
<i>bit</i>	内部 RAM 或特殊功能寄存器中的直接寻址位。
<i>C</i> 或 <i>Cy</i>	进位标志位或位处理机中的累加器。
<i>addr11</i>	11 位目的地址。
<i>addr16</i>	16 位目的地址。
@	间接寻址寄存器前缀,如 $@R_i$, $@A + DPTR$ 。
(<i>X</i>)	<i>X</i> 中的内容。
((<i>X</i>))	由 <i>X</i> 寻址的单元中的内容。
→	箭头右边的内容被箭头左边的内容所取代。

3.4.1 数据传送类指令

数据传送类指令是编程时使用最频繁的一类指令。

一般数据传送类指令的助记符为“MOV”,通用的格式为:

MOV <目的操作数>, <源操作数>

数据传送类指令是把源操作数传送到目的操作数。指令执行后,源操作数不改变,目的操作数修改为源操作数。所以数据传送类操作属“复制”性质,而不是“搬家”。若要求在数据传送时,不丢失目的操作数,则可以用交换型的传送类指令。

数据传送类指令不影响标志位,这里所说的标志位是指 Cy、Ac 和 OV,但不包括检验累加器奇偶标志位 P。

1. 以累加器为目的操作数的指令

```
MOV  A, Rn      ; (Rn) → A, n=0~7
MOV  A, @Ri     ; ((Ri)) → A, i=0,1
MOV  A, direct  ; (direct) → A
MOV  A, #data   ; #data → A
```

这组指令的功能是把源操作数的内容送入累加器 A,源操作数有寄存器寻址,直接寻址,间接寻址和立即寻址等方式,例如:

```
MOV  A, R6      ; (R6) → A, 寄存器寻址
MOV  A, 70H     ; (70H) → A, 直接寻址
MOV  A, @R0     ; ((R0)) → A, 间接寻址
MOV  A, #78H    ; 78H → A, 立即寻址
```

2. 以 Rn 为目的操作数的指令

```
MOV  Rn, A      ; (A) → Rn, n=0~7
MOV  Rn, direct ; (direct) → Rn, n=0~7
MOV  Rn, #data  ; #data → Rn, n=0~7
```

这组指令的功能是把源操作数的内容送入当前一组工作寄存器区的 R0~R7 中的某一个寄存器。

3. 以直接地址 direct 为目的操作数的指令

```
MOV  direct, A   ; (A) → direct
MOV  direct, Rn  ; (Rn) → direct, n=0~7
MOV  direct1, direct2 ; (direct2) → direct1
MOV  direct, @Ri ; ((Ri)) → direct, i=0,1
MOV  direct, #data ; #data → direct
```

这组指令的功能是把源操作数送入直接地址指出的存储单元。*direct* 指的是内部 RAM 或 SFR 的地址。

4. 以寄存器间接地址为目的操作数的指令

```
MOV  @Ri, A      ; (A) → ((Ri)), i=0,1
MOV  @Ri, direct ; (direct) → ((Ri)), i=0,1
MOV  @Ri, #data  ; #data → ((Ri)), i=0,1
```

这组指令的功能是把源操作数内容送入 R0 或 R1 指出的存储单元中。

5. 16 位数传送指令

MOV DPTR, #data16 ; #data16→DPTR

这条指令的功能是把 16 位常数送入 DPTR,这是整个指令系统中惟一的 1 条 16 位数据的传送指令,用来设置地址指针 DPTR。地址指针 DPTR 由 DPH 和 DPL 组成。这条指令执行的结果把立即数的高 8 位送入 DPH,立即数的低 8 位送入 DPL。

对于所有 MOV 类指令,累加器 A 是 1 个特别重要的 8 位寄存器,CPU 对它具有其它寄存器所没有的操作指令。后面将要介绍的加、减、乘、除指令都是以 A 作为目的操作数的。 R_n 为 CPU 当前选择的寄存器组中的 $R_0 \sim R_7$,直接地址指出的存储单元为内部 RAM 的 $00H \sim 7FH$ 和特殊功能寄存器(地址范围为 $80H \sim FFH$)。在间接地址中,用 R0 或 R1 作地址指针,访问内部 RAM 的 $00H \sim 7FH$ 128 个单元。

6. 堆栈操作指令

在 MCS-51 内部 RAM 中可以设定一个后进先出(Last In First Out, LIFO)的区域称作堆栈。在特殊功能寄存器中有一个堆栈指针 SP,它指出堆栈的栈顶位置。堆栈操作有进栈和出栈两种,因此在指令系统中相应有两条堆栈操作指令。

(1) 进栈指令

PUSH direct

这条指令的功能是首先将栈指针 SP 加 1,然后把 *direct* 中的内容送到栈指针 SP 指示的内部 RAM 单元中。

例如:当 $(SP)=60H$, $(A)=30H$, $(B)=70H$ 时,执行下列指令

PUSH A ; $(SP)+1=61H \rightarrow SP, (A) \rightarrow 61H$

PUSH B ; $(SP)+1=62H \rightarrow SP, (B) \rightarrow 62H$

结果: $(61H)=30H$, $(62H)=70H$, $(SP)=62H$

(2) 出栈指令

POP direct

这条指令的功能是栈指针 SP 指示的栈顶(内部 RAM 单元)内容送入 *direct* 字节单元中,栈指针 SP 减 1。

例如:当 $(SP)=62H$, $(62H)=70H$, $(61H)=30H$,执行下列指令

POP DPH ; $((SP)) \rightarrow DPH, (SP)-1 \rightarrow SP$

POP DPL ; $((SP)) \rightarrow DPL, (SP)-1 \rightarrow SP$

结果: $(DPTR)=7030H$, $(SP)=60H$

7. 累加器 A 与外部数据存储器传送指令

MOVX A,@DPTR	;((DPTR))→A,读外部 RAM/IO
MOVX A,@Ri	;((Ri))→A,读外部 RAM/IO
MOVX @DPTR,A	;(A)→((DPTR)),写外部 RAM/IO
MOVX @Ri,A	;(A)→((Ri)),写外部 RAM/IO

这组指令的功能是读外部 RAM 存储器或 I/O 中的 1 B 的数据到累加器 A 中,或从累加器 A 中的 1 B 的数据写到外部 RAM 存储器或 I/O 中。

采用 16 位的 DPTR 作间接寻址,则可寻址整个 64 KB 片外数据存储器空间,高 8 位地址(DPH)由 P2 口输出,低 8 位地址(DPL)由 P0 口输出。

采用 Ri(i=0,1)作间接寻址,可寻址片外 256 个单元的数据存储器。8 位地址和数据均由 P0 口输出,可选用其它任何输出口线来输出高于 8 位的地址(一般选用 P2 口输出高 8 位的地址)。

上述 4 条指令的助记符是在 MOV 的后面加“X”,“X”表示 MCS-51 单片机访问的是片外 RAM 存储器或 I/O。

8. 查表指令

这类指令共两条,均为单字节指令,这是 MCS-51 指令系统中仅有的两条用于读程序存储器中的数据表格的指令。这里所说的程序存储器既包括内部程序存储器,也包括外部程序存储器。由于对程序存储器只能读不能写,因此其数据的传送都是单向的,即从程序存储器中读出数据到累加器中。两条查表指令均采用基址寄存器加变址寄存器间接寻址方式。

MOVC A,@A+PC:这条指令以 PC 作基址寄存器,A 的内容作为无符号整数和 PC 中的内容(下一条指令的起始地址)相加后得到 1 个 16 位的地址,把该地址指出的程序存储单元的内容送到累加器 A。

例如:(A)=30H,执行地址 1000H 处的指令

1000H: MOVC A,@A+PC

本指令占用 1 B,下一条指令的地址为 1001 H,(PC)=1001 H 再加上 A 中的 30 H,得 1031 H,结果将程序存储器中 1031 H 的内容送入 A。

这条指令的优点是不改变特殊功能寄存器及 PC 的状态,根据 A 的内容就可以取出表格中的常数。缺点是表格只能存放在该条查表指令后面的 256 个单元之内,表格的大小受到限制,而且表格只能被一段程序所利用。

MOVC A,@A+DPTR:这条指令以 DPTR 作为基址寄存器,A 的内容作为无符号数和 DPTR 的内容相加得到 1 个 16 位的地址,把由该地址指出的程序存储器单元的内容送到累加器 A。

例如 (DPTR)=8100H (A)=40H 执行指令

MOVC A,@A+DPTR

结果将程序存储器中 8140H 单元内容送入累加器 A 中。

这条查表指令的执行结果只和指针 DPTR 及累加器 A 的内容有关,与该指令存放的地址及常数表格存放的地址无关,因此表格的大小和位置可以在 64 KB 程序存储器中任意安排,1 个表格可以为各个程序块公用。

上述 2 条指令的助记符是在 MOV 的后面加 C,“C”是 CODE 的第 1 个字母,即代码的意思。

9. 字节交换指令

XCH A,Rn ; (A) ↔ (Rn), n = 0~7

XCH A,direct ; (A) ↔ (direct)

XCH A,@Ri ; (A) ↔ ((Ri)), i = 0,1

这组指令的功能是将累加器 A 的内容和源操作数的内容相互交换。源操作数有寄存器寻址、直接寻址和寄存器间接寻址等方式。例如:

(A) = 80H, (R7) = 08H, (40H) = F0H

(R0) = 30H, (30H) = 0FH

执行下列指令:

XCH A,R7 ; (A) ↔ (R7)

XCH A,40H ; (A) ↔ (40H)

XCH A,@R0 ; (A) ↔ ((R0))

结果:(A) = 0FH, (R7) = 80H, (40H) = 08H, (30H) = F0H

10. 半字节交换指令

XCHD A,@Ri

累加器的低 4 位与内部 RAM 低 4 位交换。例如:

(R0) = 60H, (60H) = 3EH, (A) = 59H

执行完 XCHD A,@R0 指令,则(A) = 5EH, (60H) = 39H。

3.4.2 算术操作类指令

在 MCS-51 指令系统中,有单字节的加、减、乘、除法指令,算术运算功能比较强。算术运算指令都是针对 8 位二进制数无符号数的,如要进行带符号或多字节二进制数运算,需编写程序,通过执行程序实现。

算术执行的结果将使 PSW 中的进位(Cy),辅助进位(Ac),溢出(OV)3 种标志位置 1 或清 0,但是增 1 和减 1 指令不影响这些标志。

1. 加法指令

共有 4 条加法运算指令:

ADD A,Rn ; (A) + (Rn) → A, n = 0~7

ADD A,direct ; (A) + (direct) → A

ADD A,@R1 ; (A) + ((R1)) → A, i = 0, 1

ADD A,#data ; (A) + #data → A

这 4 条 8 位二进制数加法指令的一个加数总是来自累加器 A, 而另一个加数可由寄存器寻址、直接寻址、寄存器间接寻址和立即寻址等不同的寻址方式得到。其相加的结果总是放在累加器 A 中。

使用加法指令时, 要注意累加器 A 中的运算结果对各个标志位的影响:

(1) 如果位 7 有进位, 则置 1 进位标志 Cy, 否则清 0 Cy

(2) 如果位 3 有进位, 置 1 辅助进位标志 Ac, 否则清 0 Ac (Ac 为 PSW 寄存器中的 1 位)。

(3) 如果位 6 有进位, 而位 7 没有进位, 或者位 7 有进位, 而位 6 没有, 则溢出标志位 OV 置 1, 否则清 0 OV。

溢出标志位 OV 的状态, 只有在带符号数加法运算时才有意义。当 2 个带符号数相加时, $OV=1$, 表示加法运算超出了累加器 A 所能表示的带符号数的有效范围 ($-128 \sim +127$), 即产生了溢出, 因此运算结果是错误的, 否则运算是正确的, 即无溢出产生。

例 3-1 (A) = 53H, (R0) = FCH, 执行指令

ADD A,R0

运算式为:

$$\begin{array}{r} 0101\ 0011 \\ +) \quad 1111\ 1100 \\ \hline 1 \leftarrow 0100\ 1111 \end{array}$$

结果为:

(A) = 4FH, Cy = 1, Ac = 0, OV = 0, P = 1 (A 中 1 的位数为奇数)

注意: 上面的运算中, 由于位 6 和位 7 同时有进位, 所以标志位 $OV=0$ 。

例 3-2 (A) = 85H, (R0) = 20H, (20H) = AFH, 执行指令:

ADD A,@R0

运算式为:

$$\begin{array}{r} 1000\ 0101 \\ +) \quad 1010\ 1111 \\ \hline 1 \leftarrow 0011 \leftarrow 0100 \end{array}$$

结果为:

(A) = 34H, Cy = 1, Ac = 1, OV = 1, P = 1

注意: 由于位 7 有进位, 而位 6 无进位, 所以标志位 $OV=1$ 。

2. 带进位加法指令

带进位的加法运算的特点是进位标志位 Cy 参加运算, 因此带进位的加法

运算是 3 个数相加。带进位的加法指令共 4 条：

ADDC A,Rn ; (A)+(Rn)+C→A,n=0~7

ADDC A,direct ; (A)+(direct)+C→A

ADDC A,@Ri ; (A)+(Ri)+C→A,i=0,1

ADDC A,#data ; (A)+data+C→A

这组带进位加法指令的功能是指令中不同寻址方式所指出的加数、进位标志与累加器 A 内容相加,结果存在累加器 A 中。如果位 7 有进位,则置 1 进位标志 Cy,否则清 0 Cy;如果位 3 有进位输出,则置 1 辅助进位标志 Ac,否则清 0 Ac;如果位 6 有进位而位 7 没有进位,或者位 7 有进位而位 6 没有,则置 1 溢出标志 OV,否则清 0 标志 OV。

例 3-3 (A)=85H,(20H)=FFH,Cy=1,执行指令

ADDC A,20H

运算式为:

$$\begin{array}{r} 1000\ 0101 \\ 1111\ 1111 \\ +) \qquad \qquad 1 \\ \hline 1 \leftarrow 1000\ 0101 \end{array}$$

结果为:

(A)=85H,Cy=1,Ac=1,OV=0,P=1(A 中 1 的位数为奇数)

3. 增 1 指令

共有 5 条增 1 指令:

INC A

INC Rn ;n=0~7

INC direct

INC @Ri ;i=0,1

INC DPTR

这组增 1 指令的功能是把指令中所指出的变量增 1,且不影响程序状态字 PSW 中的任何标志。若变量原来为 FFH,加 1 后将溢出为 00H(指前 4 条指令),标志也不会受到影响。第 5 条指令 INC DPTR,是 16 位数增 1 指令。指令首先对低 8 位指针 DPL 的内容执行加 1 的操作,当产生溢出时,就对 DPH 的内容进行加 1 操作,并不影响标志 Cy 的状态。

4. 十进制调整指令

十进制调整指令用于对 BCD 码十进制数加法运算结果的内容修正。其指令格式为:

DA A

这条指令的功能是对压缩的 BCD 码的加法结果进行十进制调整。若两个

BCD 码按二进制相加之后,必须经本指令的调整才能得到正确的压缩 BCD 码的和数。

(1) 十进制调整问题:前面介绍的 ADD 和 ADDC 加法指令,对二进制数的加法运算,都能得到正确的结果。但对于十进制数(BCD 码)的加法运算,只能借助于二进制加法指令。然而,二进制数的加法运算原则并不能适用于十进制数的加法运算,有时会产生错误结果。例如:

① $3+6=9$	② $7+8=15$	③ $9+8=17$
$\begin{array}{r} 0011 \\ +) 0110 \\ \hline 1001 \end{array}$	$\begin{array}{r} 0111 \\ +) 1000 \\ \hline 1111 \end{array}$	$\begin{array}{r} 1001 \\ +) 1000 \\ \hline 1 \leftarrow 0001 \end{array}$

- ① 运算结果正确。
- ② 运算结果不正确,因为十进制数的 BCD 码中没有 1111 这个编码。
- ③ 运行结果也是不正确的,正确结果应为 17,而运算结果却是 11。

这种情况表明,二进制数加法指令不能完全适用于 BCD 码十进制数的加法运算,因此要对结果作有条件的修正。这就是所谓的十进制调整问题。

(2) 出错原因和调整方法:出错的原因在于 BCD 码是 4 位二进制编码,共有 16 个编码,但 BCD 码只用了其中的 10 个,剩下 6 个没用到。这 6 个没用到的编码(1010,1011,1100,1101,1110,1111)为无效码。

在 BCD 码的加法运算中,凡结果进入或者跳过无效码编码区时,其结果就是错误的。因此 1 位 BCD 码加法运算出错情况有以下 2 种:

- ① 相加结果大于 9,说明已经进入无效编码区。
- ② 相加结果有进位,说明已经跳过无效编码区。

无论哪一种出错情况,都是因为 6 个无效编码造成的。因此,只要出现上述 2 种情况之一,就必须进行调整。调整的方法是把结果加 6 调整,即所谓十进制调整修正。

十进制调整的修正方法应是:

- ① 累加器低 4 位大于 9 或辅助进位位 $A_c=1$,则进行低 4 位加 6 修正。
- ② 累加器高 4 位大于 9 或进位位 $C_y=1$,则进行高 4 位加 6 修正。
- ③ 累加器高 4 位为 9,低 4 位大于 9,则高 4 位和低 4 位分别加 6 修正。

上述的十进制调整的修正方法,具体是通过执行指令:DA A 来自动实现的。

例 3-4 $(A)=56H, (R5)=67H$,把它们看作为 2 个压缩的 BCD 数,进行 BCD 数的加法。执行指令:

```
ADD A,R5
DA A
```

由于高、低 4 位分别大于 9, 所以要分别加 6 进行十进制调整对结果进行修正。

$$\begin{array}{r}
 0101 \ 0110 \\
 +) 0110 \ 0111 \\
 \hline
 1011 \ 1101 \\
 +) 0110 \ 0110 \quad \leftarrow \text{十进制调整, 高低 4 位分别加 6} \\
 \hline
 1 \leftarrow 0010 \ 0011
 \end{array}$$

结果为: $(A) = 23H, Cy = 1$

由上可见, $56 \div 67 = 123$, 结果是正确的。

5. 带借位的减法指令

共有 4 条指令:

SUBB A, Rn ; $(A) - (Rn) - Cy \rightarrow A, n = 0 \sim 7$
 SUBB A, direct ; $(A) - (\text{direct}) - Cy \rightarrow A$
 SUBB A, @Ri ; $(A) - ((Ri)) - Cy \rightarrow A, i = 0, 1$
 SUBB A, #data ; $(A) - \#data - Cy \rightarrow A,$

这组带借位减法指令是从累加器 A 中的内容减去指定的变量和进位标志 Cy 的值, 结果存在累加器 A 中。如果位 7 需借位则置 1 Cy, 否则清 0 Cy; 如果位 3 需借位则置 1 Ac, 否则清 0 Ac; 如果位 6 需借位而位 7 不需要借位, 或者位 7 需借位, 位 6 不需借位, 则置 1 溢出标志位 OV, 否则清 0 OV。源操作数允许有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址方式。

例 3-5 $(A) = C9H, (R2) = 54H, Cy = 1$, 执行指令

SUBB A, R2

运算式为:

$$\begin{array}{r}
 1100 \ 1001 \\
 0101 \ 0100 \\
 \hline
 -) 1 \\
 \hline
 0111 \ 0100
 \end{array}$$

结果为:

$(A) = 74H, Cy = 0, Ac = 0, OV = 1$ (位 6 向位 7 借位)

6. 减 1 指令

共有 4 条指令:

DEC A ; $(A) - 1 \rightarrow A$
 DEC Rn ; $(Rn) - 1 \rightarrow Rn, n = 0 \sim 7$
 DEC direct ; $(\text{direct}) - 1 \rightarrow \text{direct}$
 DEC @Ri ; $((Ri)) - 1 \rightarrow (Ri), i = 0, 1$

这组指令的功能是指定的变量减 1。若原来为 00H。减 1 后下溢为 FFH, 不影响标志位(除 A 减 1 影响 P 标志外)。例如:

(A)=0FH, (R7)=19H, (30H)=00H, (R1)=40H, (40H)=0FFH 执行指令:

```
DEC  A      ;(A)-1→A
DEC  R7     ;(R7)-1→R7
DEC  30H    ;(30H)-1→30H
DEC  @R1    ;((R1))-1→(R1)
```

结果:(A)=0EH, (R7)=18H, (30H)=0FFH, (40H)=0FEH, P=1, 不影响其它标志。

7. 乘法指令

```
MUL  AB     ;A×B→BA
```

这条指令的功能是把累加器 A 和寄存器 B 中的无符号 8 位整数相乘, 其 16 位积的低位字节在累加器 A 中, 高位字节在 B 中。如果积大于 255, 则置 1 溢出标志位 OV, 否则清 0 OV。进位标志位 Cy 总是清 0。

8. 除法指令

```
DIV  AB     ;A/B→A(商), 余数→B
```

该指令的功能是把累加器 A 中 8 位无符号整数(被除数)除以 B 中的 8 位无符号整数(除数), 所得的商(为整数)存放在累加器 A 中, 余数在寄存器 B 中, 清 0 Cy 和溢出标志位 OV。如果 B 的内容为 0(即除数为 0), 则存放结果的 A、B 中的内容不定, 并置 1 溢出标志位 OV。

例 3-6 (A)=FBH, (B)=12H, 执行指令

```
DIV  AB
```

结果为:

(A)=0DH, (B)=11H, Cy=0, OV=0

3.4.3 逻辑运算指令

1. 简单逻辑操作指令

```
CLR  A
```

该条指令的功能是累加器 A 清 0。不影响 Cy、Ac、OV 等标志。

```
CPL  A
```

该条指令的功能是将累加器 A 的内容按位逻辑取反, 不影响标志。

2. 左环移指令

```
RL   A
```

这条指令的功能是累加器 A 的 8 位向左循环移位, 位 7 循环移入位 0, 不影响标志。

3. 带进位左环移指令

RLC A

这条指令的功能是将累加器 A 的内容和进位标志位 Cy 一起向左环移一位, Acc. 7 移入进位位 Cy, Cy 移入 Acc. 0, 不影响其它标志。

4. 右环移指令

RR A

这条指令的功能是累加器 A 的内容向右环移一位, Acc. 0 移入 Acc. 7, 不影响其它标志。

5. 带进位环移指令

RRC A

这条指令的功能是累加器 A 的内容和进位标志 Cy 一起向右环移一位, Acc. 0 进入 Cy, Cy 移入 Acc. 7。

6. 累加器半字节交换指令

SWAP A

这条指令的功能是将累加器 A 的高半字节 (Acc. 7 ~ Acc. 4) 和低半字节 (Acc. 3 ~ Acc. 0) 互换。

例 3-7 (A)=0C5H, 执行指令

SWAP A

结果: (A)=5CH

7. 逻辑与指令

ANL A, Rn ; (A) ∧ (Rn) → A, n=0~7

ANL A, direct ; (A) ∧ (direct) → A

ANL A, #data ; (A) ∧ #data → A

ANL A, @Ri ; (A) ∧ ((Ri)) → A, i=0~1

ANL direct, A ; (direct) ∧ (A) → direct

ANL direct, #data ; (direct) ∧ #data → direct

这组指令的功能是在指出的变量之间以位为基础进行逻辑与操作, 结果存放在目的变量所在的寄存器或存储器中去。操作数有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址方式。

例 3-8 (A)=07H, (R0)=0FDH, 执行指令:

ANL A, R0

运算式为:

$$\begin{array}{r} \text{(A)} \quad 00000111 \\ \wedge \quad 11111101 \\ \hline 00000101 \end{array}$$

结果: (A)=05H

8. 逻辑或指令

ORL A, Rn ; (A) \vee (Rn) \rightarrow A, n = 0 ~ 7
 ORL A, direct ; (A) \vee (direct) \rightarrow A
 ORL A, # data ; (A) \vee data \rightarrow A
 ORL A, @Ri ; (A) \vee ((Ri)) \rightarrow A, i = 0, 1
 ORL direct, A ; (direct) \vee (A) \rightarrow direct
 ORL direct, # data ; (direct) \vee # data \rightarrow direct

这组指令的功能是在所指出的变量之间执行以位为基础的逻辑或操作, 结果存到目的变量寄存器或存储器中去。操作数有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址方式。

例 3-9 (P1) = 05H, (A) = 33H, 执行指令

ORL P1, A

运算式为:

$$\begin{array}{r}
 \\
 \vee) \quad \underline{00000101} \\
 \quad \underline{00110011} \\
 \hline
 \quad 00110111
 \end{array}$$

结果: (P1) = 37H

9. 逻辑异或指令

XRL A, Rn ; (A) \oplus (Rn) \rightarrow A
 XRL A, direct ; (A) \oplus (direct) \rightarrow A
 XRL A, @Ri ; (A) \oplus ((Ri)) \rightarrow A, i = 0, 1
 XRL A, # data ; (A) \oplus # data \rightarrow A
 XRL direct, A ; (direct) \oplus (A) \rightarrow direct
 XRL direct, # data ; (direct) \oplus # data \rightarrow direct

这组指令的功能是在所指出的变量之间执行以位为基础的逻辑异或操作, 结果存到目的变量寄存器或存储器中去。

操作数有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址等方式。

例 3-10 (A) = 90H, (R3) = 73H 执行指令:

XRL A, R3

运算式为:

$$\begin{array}{r}
 \\
 \oplus) \quad \underline{10010000} \\
 \quad \underline{01110011} \\
 \hline
 \quad 11100011
 \end{array}$$

结果: (A) = E3H

3.4.4 控制转移类指令

1. 无条件转移指令

AJMP *addr11*

这是 2 KB 范围内的无条件跳转指令。AJMP 把 MCS-51 的 64 KB 程序存储器空间划分为 32 个区,每个区为 2 KB,转移目标地址必须与 AJMP 下一条指令的第一个字节在同 2 KB 区范围内(即转移的目标地址必须与 AJMP 下一条指令的地址的高 5 位地址码 A15~A11 相同),否则,将引起混乱。如果 AJMP 指令正好落在 2 KB 区底的 2 个单元内,程序就转移到下一个区中去了,这时不会出现问题。执行该指令时,先将 PC 加 2,然后把 *addr11* 送入 PC.10~PC.0,PC.15~PC.11 保持不变,程序转移到目标地址指定的地方。

本指令是为了能与 MCS-48 的 JMP 指令兼容而设的。

2. 相对转移指令

SJMP *rel*

这是无条件转移指令,其中 *rel* 为相对偏移量。前面已介绍过,*rel* 是 1 个单字节的带符号的 8 位二进制数的补码数,因此所能实现的程序转移是双向的。*rel* 如为正,则向后(即地址增大的方向)转移,*rel* 如为负,则向前(即地址减小的方向)转移。执行本指令时,在 PC 加 2(本指令为 2 B)之后,把指令的有符号的偏移量 *rel* 加到 PC 上,并计算出目标地址,因此跳转的目标地址可以在这条指令前 127 B 到后 128 B 之间。

用户在编写程序时,只需在相对转移指令中,直接写上要转向的目标地址标号就可以了。例如:

```
LOOP:MOV  A,R6
      :
      SJMP LOOP
      :
```

程序在汇编时,由汇编程序自动计算和填入偏移量。

但在手工汇编时,偏移量 *rel* 的值则需程序设计人员自己计算。这可从如下两个方面来讨论:

(1)根据偏移量 *rel* 计算转移的目标地址

例如,在 1230H 地址上有 SJMP 指令:

1230H: SJMP 46H

假设 SJMP 指令所在地址为 1234H, *rel*=46H 是正数,因此程序是向后转移。目标地址=1230H+02H+46H=1278H,则执行完本条指令后,程序转移到 1278H 地址去执行程序。

又例如在 1234H 地址上的 SJMP 指令是:

1230H: SJMP 0E7H

rel=0E7H 是正数,是负数 19H 的补码,因此程序向前转移,目标地址=

$1230H + 02H - 19H = 1219H$, 则执行完本条指令后, 程序转移到 $1219H$ 地址去执行程序。

(2) 根据目标地址计算偏移量

这种情况下, rel 的计算公式是:

向前转移: $rel = FFH - \text{源地址} - \text{目标地址} - 1$

向后转移: $rel = \text{目标地址} - \text{源地址} - 2$

3. 长跳转指令

`LJMP addr16`

这条指令执行时把指令的第二和第三字节分别装入 PC 的高位和低位字节中, 无条件地转向 $addr16$ 指出的目标地址。转移的目标地址可以在 64 KB 程序存储器地址空间的任何位置。

4. 间接跳转指令

`JMP @A · DPTR`

这是一条单字节的转移指令, 转移的目标地址由 A 中 8 位无符号数与 DPTR 的 16 位数内容之和来确定。本指令以 DPTR 内容作为基址, A 的内容作变址。因此, 只要把 DPTR 的值固定, 而给 A 赋予不同的值, 即可实现程序的多分支转移。

本指令不改变累加器 A 和数据指针 DPTR 内容, 也不影响标志。

5. 条件转移指令

条件转移指令就是程序的转移是有条件的。执行条件转移指令时, 如指令中规定的条件满足, 则进行转移, 条件不满足则顺序执行下一条指令。转移的目标地址在以下 ~ 条指令地址为中心的 256B 范围内 ($+127 \sim -128$)。当条件满足时, PC 装入下 ~ 条指令的第一个字节地址, 再把带符号的相对偏移量 rel 加到 PC 上, 计算出要转向的目标地址。

`JZ rel` ; 如果累加器为 0, 则执行转移

`JNZ rel` ; 如果累加器非 0, 则执行转移

6. 比较不相等转移指令

`CJNE A, direct, rel`

`CJNE A, #data, rel`

`CJNE Rn, #data, rel`

`CJNE @Ri, #data, rel`

这组指令的功能是比较前面两个操作数的大小, 如果它们的值不相等则转移, 在 PC 加到下一条指令的起始地址后, 通过把指令最后一个字节的有符号的相对偏移量加到 PC 上, 计算出转向的目标地址。如果第一操作数(无符号整数)小于第二操作数(无符号整数), 则置进位标志位 Cy, 否则清 0 Cy。该指令的

执行不影响任何一个操作数的内容。

操作数有寄存器寻址、直接寻址、寄存器间接寻址和立即寻址等方式。

7. 减1不为0转移指令

这是一组把减1与条件转移两种功能结合在一起的指令。共2条指令：

DJNZ Rn,rel ;n 0~7

DJNZ direct,rel

这组指令将源操作数(Rn 或 $direct$)减1,结果回送到 Rn 寄存器或 $direct$ 中去。如果结果不为0则转移。本指令允许程序员把寄存器 Rn 或内部RAM的 $direct$ 单元用作程序循环计数器。

这2条指令主要用于控制程序循环。如预先把寄存器 Rn 或内部RAM的 $direct$ 单元装入循环次数,则利用本指令,以减1后是否为0作为转移条件,即可实现按次数控制循环。

8. 调用子程序指令

(1) 短调用指令

ACALL addr11

这是2KB范围内的调用子程序的指令。执行时先把PC加2(本指令为2B),获得下一条指令地址,把该地址压入堆栈中保护,即栈指针SP加1,PCH进栈,SP再加1,PCH进栈。最后把PC的高5位和指令代码中的 $addr11$ 连接获得16位的子程序入口地址,并送入PC,转向执行子程序。所调用的子程序地址必须与ACALL指令下一条指令的第一个字节在同一个2KB区内(即16位地址中的高5位地址相同),否则将引起程序转移混乱。如果ACALL指令正好落在区底的2个单元内,程序就转移到下一个区中去了。因为在执行调用操作之前PC先加了2。

这条指令与AJMP指令相类似,是为了与MCS-48中的CALL指令兼容而设的。指令的执行不影响标志。

(2) 长调用指令

LCALL addr16

LCALL指令可以调用64KB范围内程序存储器中的任何一个子程序。指令执行时,先把程序计数器加3获得下条指令的地址(也就是断点地址),并把它压入堆栈(先低位字节后高位字节),同时把堆栈指针加2。接着把指令的第二和第三字节($A15\sim A8$, $A7\sim A0$)分别装入PC的高位和低位字节中,然后从PC中指出的地址开始执行程序。

本指令执行后不影响任何标志。

9. 子程序的返回指令

RET

执行本指令时:

$(SP) \rightarrow PCH$, 然后 $(SP) - 1 \rightarrow SP$

$(SP) \rightarrow PCL$, 然后 $(SP) - 1 \rightarrow SP$

功能是从堆栈中退出 PC 的高 8 位和低 8 位字节, 把栈指针减 2, 从 PC 值开始继续执行程序。不影响任何标志。

10. 中断返回指令

RETl

这条指令的功能和 RET 指令相似, 2 条指令的不同之处, 是本指令清除了中断响应时, 被置 1 的 MCS-51 内部中断优先级寄存器的优先级状态。

11. 空操作指令

NOP

CPU 不进行任何实际操作, 只消耗 1 个机器周期的时间。只执行 $(PC) + 1 \rightarrow PC$ 操作。NOP 指令常用于程序中的等待或时间的延迟。

3.4.5 位操作指令

MCS-51 单片机内部有一个位处理机, 对位地址空间具有丰富的位操作指令。

1. 数据位传送指令

MOV C, bit

MOV bit, C

这组指令的功能是把由源操作数指出的位变量送到目的操作数指定的单元中去。其中一个操作数必须为进位标志, 另一个可以是任何直接寻址位。不影响其它寄存器或标志。

例 3-11 指令如下:

MOV C, 06H ; (20H). 6 \rightarrow Cy

注意, 这里的 06H 是位地址, 20H 是内部 RAM 的字节地址。06H 是内部 RAM 20H 字节位 6 的位地址。

MOV P1.0, C ; Cy \rightarrow P1.0

2. 位变量修改指令

CLR C ; 清 0 Cy

CLR bit ; 清 0 bit 位

CPL C ; Cy 求反

CPL bit ; bit 位求反

SETB C ; 置 1 Cy

SETB bit ; 置 1 bit 位

这组指令将操作数指出的位清0、求反、置1,不影响其它标志。

例3-12 指令如下:

```
CLR    C           ;0→Cy
CLR    27H         ;0→(27H).7位
CPL    08H         ;(21H).0→(21H).0位
SETB   P1.7        ;1→P1.7位
```

3. 位变量逻辑与指令

```
ANL    C,bit       ;bit ∧ Cy→Cy
ANL    C,/bit;      ; $\overline{\text{bit}} \wedge \text{Cy} \rightarrow \text{Cy}$ 
```

第1条指令的功能是:直接寻址位与进位标志位(位累加器)进行逻辑与,结果送回到进位标志位中。如果直接寻址位的布尔值是逻辑0,则进位标志位C清0,否则进位标志保持不变。

第2条指令的功能是:先对直接寻址位求反,然后与位累加器(进位标志位)进行逻辑与,结果送回到位累加器中。本指令不影响直接寻址位求反前原来的状态,也不影响别的标志。直接寻址位的源操作数只有直接位寻址方式。

4. 位变量逻辑或指令

```
ORL    C,bit
ORL    C,/bit
```

第1条指令的功能是:直接寻址位与进位标志位Cy(位累加器)进行逻辑或,结果送回到进位标志位中。如果直接寻址位的位值为1,则置1进位标志位,否则进位标志位仍保持原来状态。

第2条指令的功能是:先对直接寻址位求反,然后与进位标志位(位累加器)进行逻辑或,结果送回到进位标志位中。本指令不影响直接寻址位求反前原来的状态。

5. 条件转移类指令

```
JC    rel          ;如果进位位 Cy=1,则转移
JNC   rel          ;如果进位位 Cy=0,则转移
JB    bit,rel       ;如果直接寻址位=1,则转移
JNB   bit,rel       ;如果直接寻址位=0,则转移
JBC   bit,rel       ;如果直接寻址位=1,则转移,并清0直接寻址位
```

表3-2~表3-6列出了按指令功能排列的全部指令及功能的简要说明,以及指令长度、执行的时间以及指令代码(机器代码)。读者可根据指令助记符,迅速查到对应的指令代码(手工汇编)。也可根据指令代码迅速查到对应的指令助记符(手工反汇编)。读者应熟练地掌握表3-2~表3-6的使用,因为这是使用MCS-51汇编语言进行程序设计的基础。

表 3-2 数据传送类指令表

助 记 符	说 明	字节数	执行时 间(机器 周期)	指令代码 (机器代码)
MOV A,Rn	寄存器内容传送到累加器 A	1	1	E8H~EFH
MOV A,direct	直接寻址字节传送到累加器	2	1	E5H,direct
MOV A,@Ri	间接寻址 RAM 传送到累加器	1	1	E6H~E7H
MOV A,#data	立即数传送到累加器	2	1	74H,data
MOV Rn,A	累加器内容传送到寄存器	1	1	F8H~FFH
MOV Rn,direct	直接寻址字节传送到寄存器	2	2	A8H~AFH,direct
MOV Rn,#data	立即数传送到寄存器	2	1	78H~7FH,data
MOV direct,A	累加器内容传送到直接寻址 字节	2	1	F5H,direct
MOV direct,Rn	寄存器内容传送到直接寻址 字节	2	2	88H~8FH,direct
MOV direct1,direct2	直接寻址字节 2 传送到直接 寻址字节 1	3	2	85H,direct2,direct1
MOV direct,@Ri	间接寻址 RAM 传送到直接寻 址字节	2	2	86H~87H
MOV direct,#data	立即数传送到直接寻址字节	3	2	75H,direct,data
MOV @Ri,A	累加器传送到间接寻址 RAM	1	1	F6H~F7H
MOV @Ri,direct	直接寻址字节传送到间接寻 址 RAM	2	2	A6H~A7H,direct
MOV @Ri,#data	立即数传送到间接寻址 RAM	2	1	76H~77H,data
MOV DPTR,#data16	16 位常数装入到数据指针	3	2	90H,dataH,dataL
MOVC A,@A+DPTR	代码字节传送到累加器	1	2	93H
MOVC A,@A+PC	代码字节传送到累加器	1	2	83H
MOVX A,@Ri	外部 RAM(8 位地址)传送 到 A	1	2	E2H~E3H
MOVX A,@DPTR	外部 RAM(16 位地址)传送 到 A	1	2	E0H
MOVX @Ri,A	累加器传送到外部 RAM(8 位 地址)	1	2	F2H~F3H
MOVX @DPTR,A	累加器传送到外部 RAM(16 位地址)	1	2	F0H
PUSH direct	直接寻址字节压入栈顶	2	2	C0H,direct
POP direct	栈顶字节弹出到直接寻址字节	2	2	D0H,direct
XCH A,Rn	寄存器和累加器交换	1	1	C8H~CFH
XCH A,direct	直接寻址字节和累加器交换	2	1	C5H,direct
XCH A,@Ri	间接寻址 RAM 和累加器交换	1	1	C6H~C7H
XCHD A,@Ri	间接寻址 RAM 和累加器交换 低半字节	1	1	D6H~D7H
SWAP A	累加器内高低半字节交换	1	1	C4H

表 3-3 算术操作类指令表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
ADD A, Rn	寄存器内容加到累加器	1	1	28H~2FH
ADD A, direct	直接寻址字节内容加到累加器	2	1	25H, direct
ADD A, @Ri	间接寻址 RAM 内容加到累加器	1	1	26H~27H
ADD A, #data	立即数加到累加器	2	1	24H, data
ADDC A, Rn	寄存器加到累加器(带进位)	1	1	38H~3FH
ADDC A, direct	直接寻址字节加到累加器(带进位)	2	1	35H, direct
ADDC A, @Ri	间接寻址 RAM 加到累加器(带进位)	1	1	36H~37H
ADDC A, #data	立即数加到累加器(带进位)	2	1	34H, data
SUBB A, Rn	累加器内容减去寄存器内容(带借位)	1	1	98H~9FH
SUBB A, direct	累加器内容减去直接寻址字节(带借位)	2	1	95H, direct
SUBB A, @Ri	累加器内容减去间接寻址 RAM(带借位)	1	1	96H~97H
SUBB A, #data	累加器减去立即数(带借位)	2	1	94H, data
INC A	累加器增 1	1	1	04H
INC Rn	寄存器增 1	1	1	08H~0FH
INC direct	直接寻址字节增 1	2	1	05H, direct
INC @Ri	间接寻址 RAM 增 1	1	1	06H~07H
DEC A	累加器减 1	1	1	14H
DEC Rn	寄存器减 1	1	1	18H~1FH
DEC direct	直接寻址字节减 1	2	1	15H, direct
DEC @Ri	间接寻址 RAM 减 1	1	1	16H~17H
INC DPTR	数据指针增 1	1	2	A3H
MUL AB	累加器和寄存器 B 相乘	1	4	A4H
DIV AB	累加器除以寄存器 B	1	4	84H
DA A	累加器十进制调整	1	1	D4H

表 3-4 逻辑操作类指令表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
ANL A, Rn	寄存器与到累加器	1	1	58H~5FH
ANL A, direct	直接寻址字节与到累加器	2	1	55H, direct
ANL A, @Ri	间接寻址 RAM 与到累加器	1	1	56H~57H
ANL A, #data	立即数与到累加器	2	1	54H, data
ANL direct, A	累加器与到直接寻址字节	2	1	52H, direct
ANL direct, #data	立即数与到直接寻址字节	3	1	53H, direct, data

续表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
ORL A,Rn	寄存器或到累加器	1	1	48H~4FH
ORL A,direct	直接寻址字节或到累加器	2	1	45H,direct
ORL A,@Ri	间接寻址 RAM 或到累加器	1	1	46H~47H
ORL A,#data	立即数或到累加器	2	1	44H,data
ORL direct,A	累加器或到直接寻址字节	2	2	42H,direct
ORL direct,#data	立即数或到直接寻址字节	3	2	43H,direct,data
XRL A,Rn	寄存器异或到累加器	1	1	68H~6FH
XRL A,direct	直接寻址字节异或到累加器	2	1	65H,direct
XRL A,@Ri	间接寻址 RAM 字节异或到累加器	1	1	66H~67H
XRL A,#data	立即数异或到累加器	2	1	64H,dataH
XRL direct,A	累加器异或到直接寻址字节	2	1	62H,direct
XRL direct,#data	立即数异或到直接寻址字节	3	2	63H,direct,data
CLR A	累加器清零	1	1	E4H
CPL A	累加器求反	1	1	F4H
RL A	累加器循环左移	1	1	23H
RLC A	经过进位位的累加器循环左移	1	1	33H
RR A	累加器循环右移	1	1	03H
RRC A	经过进位位的累加器循环右移	1	1	13H

表 3-5 控制转移类指令表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
ACALL addr11	绝对调用子程序	2	2	a10a9a8 10001,addr(7~0)
LCALL addr16	长调用子程序	3	2	12H,addr(15~8),addr(7~0)
RET	从子程序返回	1	2	22H
RETI	从中断返回	1	2	32H
AJMP addr11	绝对转移	2	2	a10a9a8 00001,addr(7~0)
LJMP addr16	长转移	3	2	02H,addr(15~8),addr(7~0)
SJMP rel	短转移(相对偏移)	2	2	80H,rel
JMP @A+DPTR	相对 DPTR 的间接转移	1	2	73H
JZ rel	累加器为零则转移	2	2	60H,rel
JNZ rel	累加器为非零则转移	2	2	70H,rel
CJNE A,direct,rel	比较直接寻址字节和 A, 不相等则转移	3	2	B5H,direct,rel
CJNE A,#data,rel	比较立即数和 A,不相等则转移	3	2	B4H,data,rel

续表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
CJNE Rn, #data, rel	比较立即数和寄存器, 不相等则转移	3	2	B8H~BFH, data, rel
CJNE @ Ri, #data, rel	比较立即数和间接寻址 RAM, 不相等则转移	3	2	B6H~B7H, data, rel
DJNZ Rn, rel	寄存器减 1, 不为零则转移	3	2	D8H~DFH, rel
DJNZ direct, rel	地址字节减 1, 不为零则转移	3	2	D5H, direct, rel
NOP	空操作	1	2	00H

表 3-6 位变量操作类指令表

助 记 符	说 明	字节数	执行时间 (机器周期)	指令代码 (机器代码)
CLR C	清进位位	1	1	C3H
CLR bit	清直接寻址位	2	1	C2H
SETB C	进位位置 1	1	1	D3H
SETB bit	直接寻址位置 1	2	1	D2H
CPL C	进位位取反	1	1	B3H
CPL bit	直接寻址位取反	2	1	B2H
ANL C, bit	直接寻址位与到进位位	2	2	82H, bit
ANL C, / bit	直接寻址位的反码与到进位位	2	2	B0H, bit
ORL C, bit	直接寻址位或到进位位	2	2	72H, bit
ORL C, / bit	直接寻址位的反码或到进位位	2	2	A0H, bit
MOV C, bit	直接寻址位传送到进位位	2	2	A2H, bit
MOV bit, C	进位位传送到直接寻址位	2	2	92H, bit
JC rel	如果进位位为 1 则转移	2	2	40H, rel
JNC rel	如果进位位为 0 则转移	2	2	50H, rel
JB bit, rel	如果直接寻址位为 1 则转移	3	2	20H, bit, rel
JNB bit, rel	如果直接寻址位为 0 则转移	3	2	30H, bit, rel
JBC bit, rel	如果直接寻址位为 1 则转移, 并清除该位	3	2	10H, bit, rel

思考题及习题

1. 判断以下指令的正误:

(1) MOV 28H, @R2 (2) DEC DPTR (3) INC DPTR (4) CLR R0

- (5) CPL R5 (6) MOV R0, R1 (7) PUSH DPTR (8) MOV F0, C
(9) MOV F0, Acc. 3 (10) MOVX A, @R1 (11) MOV C, 30H (12) RLC R0

2. 判断下列说法是否正确。

(A) 立即寻址方式是被操作的数据本身在指令中, 而不是它的地址在指令中。

(B) 指令周期是执行一条指令的时间。

(C) 指令中直接给出的操作数称为直接寻址。

3. 在基址加变址寻址方式中, 以()作变址寄存器, 以()或()作基址寄存器。

4. MCS-51 共有哪几种寻址方式? 各有什么特点?

5. MCS-51 指令按功能可以分为哪几类? 每类指令的作用是什么?

6. 访问 SFR, 可使用哪些寻址方式?

7. 指令格式是由()和()所组成, 也可能仅由()组成。

8. 假定累加器 A 中的内容为 30H, 执行指令:

1000H: MOVC A, @A+PC

后, 把程序存储器()单元的内容送入累加器 A 中。

9. 在 MCS-51 中, PC 和 DPTR 都用于提供地址, 但 PC 是为访问()存储器提供地址, 而 DPTR 是为访问()存储器提供地址。

10. 在寄存器间接寻址方式中, 其“间接”体现在指令中寄存器的内容不是操作数, 而是操作数的()。

11. 下列程序段的功能是什么?

PUSH A

PUSH B

POP A

POP B

12. 已知程序执行前有 $A=02H$, $SP=52H$, $(51H)=FFH$, $(52H)=FFH$ 。下述程序执行后:

POP DPH

POP DPL

MOV DPTR, #4000H

RL A

MOV B, A

MOVC A, @A+DPTR

PUSH A

MOV A, B

INC A

MOVC A, @A-DPTR

PUSH A

RET

```
ORG 4000H
```

```
DB 10H,80H,30H,50H,30H,50H
```

请问: $A=(\quad)$, $SP=(\quad)$, $(51H)=(\quad)$, $(52H)=(\quad)$, $PC=(\quad)$ 。

13. 写出完成如下要求的指令,但是不能改变未涉及位的内容。

(A) 把 Acc. 3, Acc. 4, Acc. 5 和 Acc. 6 清 0。

(B) 把累加器 A 的中间 4 位清 0。

(C) 使 Acc. 2 和 Acc. 3 置 1。

14. 假定 $A=83H$, $(R0)=17H$, $(17H)=34H$, 执行以下指令:

```
ANL A, #17H
```

```
ORL 17H, A
```

```
XRL A, @R0
```

```
CPL A
```

后, A 的内容为 (\quad) 。

15. 假设 $A=55H$, $R3=0AAH$, 在执行指令 $ANL A, R5$ 后, $A=(\quad)$, $R3=(\quad)$ 。

16. 如果 $DPTR=507BH$, $SP=32H$, $(30H)=50H$, $(31H)=5FH$, $(32H)=3CH$, 则执行下列指令后:

```
POP DPH
```

```
POP DPL
```

```
POP SP
```

则: $DPH=(\quad)$, $DPL=(\quad)$, $SP=(\quad)$

17. 假定, $SP=60H$, $A=30H$, $B=70H$, 执行下列指令:

```
PUSH A
```

```
PUSH B
```

后, SP 的内容为 (\quad) , $61H$ 单元的内容为 (\quad) , $62H$ 单元的内容为 (\quad) 。

18. 借助本书中的指令表: 表 3-3~表 3-7 对如下的指令代码(16 进制)进行手工反汇编。

```
FF C0 E0 E5 F0 F0
```

第4章 MCS-51 汇编语言程序设计

第3章介绍了 MCS-51 单片机的汇编语言指令系统,每一条指令就是汇编语言的一条命令语句。汇编语言是单片机提供给用户的最快、最有效的语言,也是能利用单片机所有硬件特性并能直接控制硬件的编程语言。

由于汇编语言是面向机器硬件的语言,因此使用汇编语言进行程序设计,必须熟悉 MCS-51 单片机的硬件结构、指令系统、寻址方式等,这样才能编写出符合要求的程序。因此,要求程序设计者对 MCS-51 单片机具有很好的软硬结合的功底。

本章首先介绍使用汇编语言进行程序设计的一些基本知识以及如何使用汇编语言来进行基本的程序设计。

4.1 汇编语言程序设计概述

程序是若干指令的有序集合,单片机的工作运行就是执行这一指令序列的过程,编写这一指令序列的过程称为程序设计。

4.1.1 机器语言、汇编语言和高级语言

目前,用于程序设计的语言基本上分为三种:机器语言、汇编语言和高级语言。下面对这三种语言作以简单介绍。

1. 机器语言

在单片机中,用二进制代码表示的指令、数字和符号简称为机器语言,直接用机器语言编写的程序称为机器语言程序。用机器语言编写的程序,不易看懂,不便于记忆,且容易出错。

2. 汇编语言

为了克服机器语言的缺点,用英文字符来代替机器语言,这些英文字符被称为助记符,用助记符表示的指令称为符号语言或汇编语言,用汇编语言编写的程

序称为汇编语言程序。

单片机不能直接执行汇编语言程序,需将汇编语言程序转换为二进制代码表示的机器语言程序,单片机才能识别和执行,通常把这一转换(翻译)工作称为“汇编”。汇编可由专门的程序来完成,这种程序称为汇编程序。经汇编程序“汇编(翻译)”得到的机器语言程序称为目标程序,原来的汇编语言程序称为源程序。

汇编语言具有如下特点:

(1) 汇编语言是面向机器的语言,程序设计人员必须对 MCS-51 单片机的硬件有相当深入的了解。

(2) 助记符指令和机器指令一一对应,所以用汇编语言编写的程序效率高,占用的存储空间小,运行速度快,因此用汇编语言能编写出最优化的程序。

(3) 汇编语言程序能直接管理和控制硬件设备(功能部件),它能处理中断,也能直接访问存储器及 I/O 接口电路。

但是,汇编语言和机器语言一样,都脱离不开具体机器的硬件,因此,这两种语言均是面向机器的语言,缺乏通用性。

3. 高级语言

高级语言不受具体机器的限制,都是参照一些数学语言面设计的,使用了许多数学公式和数学计算上的习惯用语,非常擅长于科学计算。常用的高级语言,诸如 BASIC、FORTRAN 以及 C 语言等。高级语言通用性强,直观、易懂、易学,可读性好。

计算机不能直接识别和执行高级语言,需要将其“翻译”成机器语言才能识别和执行,进行“翻译”的专用程序称为编译程序。

近年来,面向自动控制、工程设计等方面的高级语言发展很快,诸如 LISP、PROLOG 等。尤其是在单片机的程序设计上,采用高级语言,有了较快的发展。例如,使用 C 语言(C51)、PL/M 语言来进行 MCS-51 的应用程序设计。尽管目前已有部分设计人员使用 C 语言(C-51)等高级语言来完成 MCS-51 单片机的应用程序设计,但是对于程序的空间和时间要求很高的场合,汇编语言仍是必不可缺的。在这种场合下,可使用 C 语言和汇编语言混合编程。在很多需要直接控制硬件的应用场合,则更是非用汇编语言不可。从某种意义上来说,掌握汇编语言并能使用汇编语言来进行程序设计,是学习和掌握单片机程序设计的基本功之一。

目前已经有许多专门介绍使用 C51 进行 MCS-51 单片机程序设计的书籍或参考资料,由于篇幅所限,本书不介绍 C51 的编程,请读者查阅有关书籍或参考资料。

4.1.2 汇编语言语句的种类和格式

汇编语言语句有两种基本类型:指令语句和伪指令语句。

1. 指令语句

指令语句我们已在第3章介绍。

每一条指令语句在汇编时都产生一个指令代码,也叫机器代码,该机器代码对应着机器的一种操作。

2. 伪指令语句

伪指令语句是为汇编服务的。在汇编时没有机器代码与之对应。

下面首先介绍汇编语言指令语句的格式。伪指令语句将在下一小节中介绍。

大多数的汇编语言的语法规则基本相同,且具有相同的语句格式。为使所编写的汇编语言源程序可读性好、清晰、明了,汇编时不出错,应严格地遵循汇编语言的语句格式。下面对汇编语言的语句格式作具体说明。

MCS-51 的汇编语言语句是符合典型的汇编语言的4分段格式:

标号字段 (LABEL)	操作码字段 (OPCODE)	操作数字段 (OPRAND)	注释字段 (COMMENT)
-----------------	-------------------	-------------------	-------------------

上述格式中,标号字段和操作字码段之间要有冒号“:”相隔;操作码字段和操作数字段间的分界符是空格;双操作数之间用逗号相隔;操作数字段和注释字段之间的分界符用分号“;”相隔。操作码字段为必选项,其余各段为任选项,也就是说任何语句都必须有操作码字段。

例 4-1 下面是一段汇编语言程序的4分段书写格式

标号字段	操作码字段	操作数字段	注释字段
START:	MOV	A, #00H	;0→A
	MOV	R1, #10	;10→R1
	MOV	R2, #00000011B	;3→R2
LOOP:	ADD	A, R2	;(A)+(R2)→A
	DJNZ	R1, LOOP	;R1 内容减1 不为零,则循环
	NOP		
HERE:	SJMP	HERE	

有关上述4个字段在汇编语言程序中的作用以及应该遵守的基本语法规则说明如下:

1. 标号字段

标号是语句所在地址的标志符号,有了标号,程序中的其它标号才能访问该语句。例如,上例4-1中的标号“START”和“LOOP”等。有关标号的规定

如下:

- (1) 标号后边必须跟以冒号“:”。
- (2) 标号由 1~8 个 ASCII 字符组成,但第一个字符必须是字母。
- (3) 同一标号在一个程序中只能定义一次,不能重复定义。
- (4) 不能使用汇编语言已经定义的符号作为标号,例如指令助记符、伪指令以及寄存器的符号名称等。
- (5) 一条语句可以没有标号,标号的有无,取决于本程序中的其它语句是否访问该条语句。

2. 操作码字段

操作码规定了语句执行的操作,操作码是汇编语言指令中惟一不能空缺的部分。汇编程序就是根据这一字段来生成机器代码的。

3. 操作数字段

操作数用于存放指令的操作数或操作数地址,可以采用字母和数字的多种表示形式。在操作数段中,操作数的个数因指令的不同而不同,通常有单操作数、双操作数和无操作数三种情况。如果是双操作数,则操作数之间,要以逗号隔开。

在操作数的表示中,有以下几种情况需要注意:

(1) 十六进制、二进制和十进制形式的操作数表示:在多数情况下,操作数或操作数地址总是采用十六进制形式来表示的,只有在某些特殊场合才采用二进制或十进制的表示形式。若操作数采用十六进制形式,则需加后缀“H”;若操作数采用二进制形式,则需加后缀“B”;若操作数采用十进制形式,则需加后缀“D”,也可以省略后缀“D”。若十六进制的操作数以字符 A~F 中的某个开头时,则还需在它前面加一个“0”,以便在汇编时把它和字符 A~F 区别开来。

(2) 工作寄存器和特殊功能寄存器的表示:当操作数在某个工作寄存器或特殊功能寄存器中时,操作数字段允许采用工作寄存器和特殊功能寄存器的代号来表示。例如,工作寄存器用 R7~R0,累加器用 A(或 Acc)。另外工作寄存器和特殊功能寄存器也可用其地址来表示。例如,累加器 A 可用 0E0H 来表示,0E0H 为累加器 A 的地址。

(3) 美元符号 \$ 的使用:美元符号 \$ 常在转移类指令的操作数字段中使用,用于表示该转移指令操作码所在的地址。例如,如下指令:

```
JNB F0, $
```

表示若 PSW 寄存器中的 $F0=0$,则机器总是执行该指令,当 $F0=1$ 时,才往下执行下一条指令。它与如下指令是等价的:

```
HERE: JNB F0, HERE
```

例 4-1 中的最后一条指令语句“HERE: SJMP HERE”可写为:

```
SJMP $
```

该指令语句表示在原地跳转。

4. 注释字段

注释字段用于解释指令或程序的含义,对编写程序和提高程序的可读性非常有用。注释字段是任选项,使用时必须以分号“;”开头,注释的长度不限,一行写不下可以换行书写,但必须注意也要以分号“;”开头。在汇编时,注释字段不会产生机器代码。

4.1.3 伪指令

程序设计者使用 MCS-51 汇编语言编写的汇编语言源程序必须汇编(翻译)成机器代码,才能运行。在 MCS-51 汇编语言源程序中应有向汇编程序发出的指示信息,告诉它如何完成汇编工作,这一任务是通过使用伪指令来实现的。

伪指令不属于 MCS-51 指令系统中的汇编语言指令,它是程序员发给汇编程序的命令,也称为汇编程序控制命令。只有在汇编前的源程序中才有伪指令。经过汇编得到目标程序(机器代码)后,伪指令已无存在的必要,所以“伪”体现在汇编时,伪指令没有相应的机器代码产生。

伪指令具有控制汇编程序的输入输出、定义数据和符号、条件汇编、分配存储空间等功能。不同汇编语言的伪指令也有所不同,但一些基本的内容却是相同的。

下面介绍在 MCS-51 汇编语言程序中常用的伪指令。

1. ORG(ORiGin)汇编起始地址命令

在汇编语言源程序的开始,通常都用一条 ORG 伪指令来实现规定程序的起始地址。如果不用 ORG 规定,则汇编得到的目标程序将从 0000H 开始。例如:

```
ORG 2000H
START:MOV A, #00H
:
```

即规定标号 START 代表地址为 2000H 开始。

在一个源程序中,可以多次使用 ORG 指令,来规定不同的程序段的起始地址。但是,地址必须由小到大排列,地址不能交叉、重叠。例如:

```
ORG 2000H
:
ORG 2500H
:
ORG 3000H
:
```

这种顺序是正确的。若按下面顺序的排列则是错误的,因为地址出现了交叉。

```
ORG 2500H
:
ORG 2000H
:
ORG 3000H
:
```

2. END(END of assembly)汇编终止命令

本命令是汇编语言源程序的结束标志,用于终止源程序的汇编工作,它的作用告诉汇编程序,将某一段源程序翻译成指令代码的工作到此为止。因此,在整个源程序中只能有一条 END 命令,且位于程序的最后。如果 END 命令出现在程序中间,则其后面的源程序,将不予以进行汇编处理。

3. DB(Define Byte)定义字节命令

本命令用于从指定的地址开始,在程序存储器的连续单元中定义字节数据。

例如:

```
ORG 2000H
DB 30H,40H,24,"C","B"
```

汇编后:

```
(2000H)=30H
(2001H)=40H
(2002H)=18H(十进制数 24)
(2003H)=43H(字符"C"的 ASCII 码)
(2004H)=42H(字符"B"的 ASCII 码)
```

显然,DB 功能是从指定单元开始定义(存储)若干个字节,十进制数自然转换成十六进制数,字母按 ASCII 码存储。

4. DW(Define Word)定义数据字命令

本命令用于从指定的地址开始,在程序存储器的连续单元中定义 16 位的数据字。例如:

```
ORG 2000H
DW 1246H,7BH,10
```

汇编后:

```
(2000H)=12H      ;第 1 个字
(2001H)=46H
(2002H)=00H      ;第 2 个字
(2003H)=7BH
(2004H)=00H      ;第 3 个字
(2005H)=0AH
```

5. EQU(EQUate)赋值命令

本命令用于给标号赋值。赋值以后,其标号值在整个程序有效。例如:

```
TEST EQU 2000H
```

表示标号 TEST 等同于 2000H,在汇编时,凡是遇到标号 TEST 时,均以 2000H 来代替。

4.1.4 汇编语言程序设计步骤

程序设计有时可能是一件很复杂的工作,为了把复杂的工作条理化,就要有相应的编写程序的步骤和方法。汇编语言程序设计的步骤主要分为以下几步。

1. 分析问题,确定算法

首先对需要解决的问题进行具体的分析。例如,解决问题的任务是什么?工作过程是什么?已知的数据,对运算的精度和速度方面的要求是什么?找出合理的计算方法及适当的数据结构。有时,可能有几种不同的算法,在编写程序之前,先要对不同的算法进行分析、比较,找出最适宜的算法。

2. 根据算法,画出程序框图

画程序框图可以把算法和解决问题的步骤逐步具体化。通过程序框图,把程序中具有一定功能的各部分有机地联系起来。从而使人们能够抓住程序的基本线索,对全局有完整的了解。这样,设计人员容易发现设计上的错误和矛盾,减少出错的可能性。

3. 分配内存工作区及有关端口地址

分配内存工作区,尤其是片内 RAM 的分配,把内存区、堆栈区、各种缓冲区要合理地分配,并确定每个区域的首地址,便于编程使用。

要确定外部扩展的各个 I/O 端口的地址、分配 I/O 接口线。

4. 编写程序

根据程序框图所表示的算法和步骤,选择适当的指令排列起来,构成一个有机的整体,即程序。在这一步,设计者应在掌握程序设计的基本方法和技巧的基础上,注意所编程序的可读性和正确性,养成在程序的适当位置上加上注释的好习惯。

5. 上机调试

上机调试可以验证程序的正确性。任何程序编写完成后总难免有缺点和错误的,只有上机调试和试运行才能比较容易发现和纠正它们。

编写完毕的程序在上机调试前必须汇编成机器代码,才能调试和运行,调试与硬件有关程序还要借助于仿真开发工具并与硬件连接,这部分内容将在本书

的第15章中介绍。

4.2 汇编语言源程序的汇编

程序设计者编写的汇编语言源程序翻译成机器代码(指令代码)的过程称为“汇编”。汇编可分为手工汇编和机器汇编两类。本节来讨论如何实现汇编。

4.2.1 手工汇编

在汇编语言源程序设计中,对于简单的程序可用手工编程,键盘输入的编写方式。既先把程序用助记符指令写出,然后通过查指令的机器代码表,逐个把助记符指令翻译成机器代码,再进行调试和运行。通常把这种人工查表翻译指令的方法称为“手工汇编”。

手工编程时都是按绝对地址对指令定位,但在遇到的相对转移指令的偏移量的计算时,要根据转移的目标地址计算偏移量,不但麻烦,且容易出错,通常只有小程序或受条件限制时才使用。在实际的程序设计中,都是采用机器汇编来自动完成的。

4.2.2 机器汇编

机器汇编是借助于微计算机来代替手工汇编。首先是在微计算机上用编辑软件进行源程序的编辑。编辑完成后,生成一个ASCII码文件,文件的扩展名为“.ASM”。然后在微计算机上运行汇编程序,把汇编语言源程序翻译成机器代码。因此,机器汇编实际上是通过执行汇编程序来对源程序进行汇编的。由于使用微型计算机完成了汇编,而汇编后得到的机器代码却是在另一台计算机(这里是单片机)上运行,称这种机器汇编为交叉汇编。

交叉汇编后,通过微计算机的串行口(或并行口)把汇编得到的机器代码传送到用户样机(或在线仿真器)再进行程序的调试和运行。

因此,一个MCS-51单片机的应用程序的完成,应经过三个步骤:

- (1) 在微计算机上,运行编辑程序进行源程序的输入和编辑。
- (2) 对源程序进行交叉汇编得到机器代码。
- (3) 通过微计算机的串行口(或并行口)把机器代码传送到用户样机(或在线仿真器)进行程序的调试和运行。

上述的第(1)步,只需在微计算机上使用通用的编辑软件即可完成。第(2)

步的交叉汇编所用的汇编程序可在购买单片机的仿真开发工具时,由厂商提供。第(3)步骤的实现要借助于单片机仿真开发工具进行。有关单片机仿真开发工具的知识将在第15章中专门介绍。

有时,在分析现成产品的ROM/EPROM中的程序时,要将二进制数的机器代码语言程序翻译成汇编语言源程序,该过程称为反汇编。

例4-2 下面是一段源程序的汇编结果,读者可通过查第3章中的表3-3~表3-7,进行手工汇编,来验证表4-1的汇编结果是否正确。

表4-1 一段源程序的汇编结果

汇编语言源程序		汇编后的机器代码	
标号	助记符指令	地址(16进制)	机器代码(16进制)
START:	MOV A, #08H	2000	74 08
	MOV B, #76H	2002	75 F0 76
	ADD A, A	2005	75 F0
	ADD A, B	2005	75 F0
	LJMP START	2005	75 20 00

4.3 汇编语言实用程序设计

本节介绍在MCS-51单片机应用设计中常用的汇编语言程序设计

4.3.1 汇编语言程序的基本结构形式

目前,在单片机的应用程序的设计中,广泛使用的一种方法是结构化的程序设计方法。采用结构化方法设计的程序,具有程序结构清晰、可读性好、调试方便、可靠性高等优点。

根据结构化程序设计的观点,功能复杂的程序结构一般常采用以下几种基本结构:顺序结构、分支结构和循环结构,再加上广泛使用的子程序和中断服务子程序,共有5种基本的程序结构。

下面简要介绍前4种基本的程序结构。中断服务子程序将在第5章中介绍。

1. 顺序结构

顺序结构程序是最简单的程序结构,在顺序结构的程序中既无分支,也无循环,也不调用子程序,程序执行时,程序流向不变,按顺序一条一条地执行指令。

2. 分支结构

分支程序的特点是程序中含有转移指令,转移指令又分为无条件转移和有条

件转移,因此分支程序也可分为无条件分支程序和有条件分支程序。无条件分支程序中含有无条件转移指令,由于此类程序简单,这里不再讨论。有条件分支程序按结构类型来分,又分为单分支选择结构和多分支选择结构。

3. 循环结构

循环程序结构的特点是程序中含有可以反复执行的程序段,该程序段通常称为循环体。例如求 100 个数的累加和,则没有必要连续安排 100 条加法指令,可以只用 1 条加法指令并使其循环执行 100 次。因此循环程序不仅可以大大缩短程序长度和使程序所占的内存单元数量少,也能使程序结构紧凑和可读性变好。

4. 子程序

在实际的程序设计中,将那些需多次应用的、完成相同的某种基本运算操作的程序段从整个程序中独立出来,单独编成一个程序段,需要时通过指令进行调用。这样的程序段称为子程序。子程序的最后必须以子程序返回指令 RET 指令结束。

采用子程序能使整个程序的结构简单,缩短程序的设计时间,减少占用的程序存储空间。调用的程序称为主程序或调用程序。

5. 中断服务子程序

中断服务子程序是为响应请求某个中断源的中断请求服务的独立程序段,与子程序类似,不同的是中断服务子程序必须以中断子程序返回指令 RETI 指令结束。有关中断服务子程序的设计,由于与中断系统有关,将在中断系统一章中详细介绍。

4.3.2 子程序的设计

子程序在程序设计中非常重要。读者应熟练地掌握子程序的设计方法。

1. 子程序设计原则和应注意的问题

子程序是一种能完成某一特定任务的程序段。其资源要为所有调用程序共享。因此,子程序在结构上应具有独立性和通用性,在编写子程序时应注意以下问题:

(1) 子程序的第一条指令的地址称为子程序的入口地址。该指令前必须有标号。

(2) 主程序调用子程序,是通过主程序或调用程序中的调用指令来实现的。在 MCS-51 的指令集中,有如下的两条子程序调用指令:

① 绝对调用指令:ACALL addr11。这是一条双字节指令,addr11 指出了调用的目标地址,PC 指针中 16 位地址中的高 5 位不变,这意味着被调用的子程序的首地址与调用指令的下一条指令的高 5 位地址相同,只能在同一个 2 KB($1\text{ K}=1024=2^5$)区内。

② 长调用指令:LCALL addr16。这是一条三字节指令,addr16 为直接调用的目标地址,也就是说子程序可放置在 64 KB 程序存储器区的任意位置。

(3) 注意设置堆栈指针和现场保护,因调用子程序时,要把断点压入堆栈,子程序返回执行 RET 指令时再把断点弹出堆栈送入 PC 指针,因此子程序结构中必须用堆栈。在子程序运行时,首先要保护现场。现场保护通常由堆栈来完成。在需要现场保护时,在子程序的开始安排压入堆栈指令 PUSH,将要保护的内容压入堆栈,在子程序的最后的 RET 指令前,则设置出栈指令 POP,将这些保护的内容弹出堆栈,送回原来的单元,即恢复现场。

(4) 子程序返回主程序时,最后一条指令必须是 RET 指令,它的功能是在执行调用指令时,把自动压入堆栈中的断点地址弹出送入 PC 指针中,从而实现子程序返回主程序断点处继续执行主程序。

(5) 子程序可以嵌套,即主程序可以调用子程序,子程序又可以调用另外的子程序,通常的情况下可允许嵌套 8 层。

(6) 在子程序调用时,还要注意参数传递的问题。调用子程序时,主程序应先把有关参数放到某些约定的位置,子程序运行时,可以从约定位置得到这些参数。同样,子程序结束前也应把运算结果送到约定位置。返回主程序后,主程序从约定位置获得这些结果。子程序参数分为入口参数和出口参数两类。入口参数是指子程序运行需要的原始参数。出口参数是子程序执行后获得的结果参数。

2. 子程序的基本结构

综上所述,典型的子程序的基本结构如下:

主程序或调用程序:

```
MAIN:      ;          ;MAIN 为主程序或调用程序标号
           LCALL SUB   ;调用子程序 SUB
           ;
```

```
SUB:  PUSH PSW      ;现场保护
      PUSH A        ;
```

子程序处理程序段

```
      POP A          ;现场恢复,注意要先进后出
      POP PSW        ;
      RET            ;最后一条指令必须为 RET
```

}子程序

注意,上述子程序结构中,现场保护与现场恢复不是必需的,要根据实际情况来定。另外,所需保护的寄存器是哪些,也要根据实际情况来定。

3. 子程序设计举例

下面通过几个例子说明如何进行子程序的设计。例 4-3 和例 4-4 主要说明子程序的调用过程和参数传递方法。

例 4-3 单字节有符号数的加减法子程序。

该程序的功能为 $(R2) \pm (R3) \rightarrow R7$, $R2$ 和 $R3$ 中为有符号的原码, $R7$ 中存放计算结果的原码。运算结果溢出时 OV 置位。程序如下:

```
MAIN:      ;主程序入口
           LCALL SUB1 ;调用减法子程序 SUB1
           ;
           LCALL ADD1 ;调用加法子程序 ADD1
           ;
SUB1:      MOV A,R3
           CPL ACC.7 ;符号位取反
           MOV R3,A
ADD1:      MOV A,R3
           ACALL CMPT ;调用单字节求补子程序 CMPT,加数或减数求补码
           MOV R3,A
           MOV A,R2
           ACALL CMPT ;调用 CMPT
           ADD A,R3
           JB OV,OVER
           ACALL CMPT ;调用单字节求补子程序 CMPT,将结果转换成原码
           MOV R7,A
OVER:      RET
           ;
CMPT:      ... ;单字节求补子程序入口
           ;
           RET
```

程序中 $SUB1$ 为减法子程序入口, $ADD1$ 为加法子程序入口。 $CMPT$ 为单字节有符号求补码子程序。本例中参数传递是通过累加器 A 完成的,主程序将被转换的数送到 A 中,子程序将 A 中的有符号数求补后存于 A 中,主程序再将结果放回原来的单元。

另外,本子程序没有现场保护的程序段。可根据需要加上。

此外,本子程序中又调用了单字节求补子程序 $CMPT$,进行了子程序嵌套。

例 4-4 4 位 BCD 码的减法程序。

BCD 码调整指令“ $DA A$ ”不能用于减法指令后面进行十进制调整,即不能直接用该指令来完成 BCD 码的减法。本例是处理 BCD 码的减法运算的一种方法。

设被减数放在 $50H$ 和 $51H$ 单元中,减数放在 $60H$ 和 $61H$ 中,差放入 $40H$ 和 $41H$ 中。

主程序为:

```
ORG 0100H
MOV R1,50H
MOV R0,60H
CLR C ;符号位取反
LCALL BSUB ;调用子程序 BSUB
MOV 40H,A
LCALL BSUB ;调用子程序 BSUB
MOV 41H,A
;
BSUB:MOV A,#9AH ;子程序 BSUB
SUBB A,@R0
ADD A,@R1
DA A
INC R0
INC R1
CPL C
RET
```

在这个例子中,主程序通过地址寄存器 R0 和 R1 将参加运算的 BCD 码的地址传递给子程序,子程序则通过累加器将差传递给主程序。

4.3.3 查表程序设计

在单片机应用系统中,查表程序是一种常用的程序。利用它能避免进行复杂的运算或转换过程,可完成数据补偿、修正、计算、转换等各种功能,具有程序简单、执行速度快等优点。

查表就是根据自变量 x ,在表格中寻找 y ,使 $y=f(x)$ 。对于 MCS-51 单片机,数据表格一般存放于程序存储器内,MCS-51 单片机在执行查表指令时,发出读程序存储器选通脉冲信号 \overline{PSEN} 。在 MCS-51 的指令系统中,给用户提供了两条极为有用的查表指令:

```
MOVC A,@A+DPTR
MOVC A,@A+PC
```

两条指令的功能完全相同,但在具体使用上有一些差别。

指令“MOVC A,@A+DPTR”完成把 A 中的内容作为一个无符号数与 DPTR 中的内容相加,所得结果为某一程序存储单元的地址,然后把该地址单元中的内容送到累加器 A 中。DPTR 作为一个基址寄存器,执行完这条指令后,DPTR 的内容不变,仍为执行加法以前的内容。

指令“`MOVC A, @A+PC`”以 PC 作为基址寄存器, PC 的内容和 A 的内容作为无符号数, 相加后所得的数作为某一程序存储器单元的地址, 根据地址取出程序存储器相应单元中的内容送到累加器 A, 这条指令执行完以后 PC 的内容不发生变化, 仍指向查表指令的下一条指令。这条指令的优点在于预处理较少且不影响其它特殊功能寄存器的值, 所以不必保护其它特殊功能寄存器的原先值。这条指令的缺点在于该表格只能存放在这条指令的地址 $X_3X_2X_1X_0$ 以下 $00\sim FFH$ 之中, 即只能存放在地址范围 $X_3X_2X_1X_0+1\sim X_3X_2X_1X_0+100H$ 中, 这就使得表格所在的程序空间受到了限制。

下面举例说明查表指令的用法以及计算偏移量时应该注意的问题。

例 4-5 子程序的功能为: 根据累加器 A 中的数 $x(0\sim 9$ 之间) 查 x 的平方表 y , 根据 x 的值查出相应的平方 y 。本例中的 x 和 y 均为单字节数。

地 址	子 程 序
$Y_3Y_2Y_1Y_0$	<code>ADD A, #01H</code>
$Y_3Y_2Y_1Y_0+2$	<code>MOVC A, @A+PC</code>
$Y_3Y_2Y_1Y_0+3$	<code>RET</code>
$Y_3Y_2Y_1Y_0+4$	<code>DB 00H, 01H, 04H, 09H, 10H ; 数 0~9 的平方表</code> <code>DB 19H, 24H, 31H, 40H, 51H</code>

第一条指令 `ADD A, #01H` 的作用是加上偏移量, 累加器 A 中反映的仅是从表首开始向下查找多少个单元, 基址寄存器 PC 的内容并非表首, 执行查表指令时, PC 中的内容为 $Y_3Y_2Y_1Y_0+3$, 即指向 `RET` 指令, 所以必须使得 A 加上从基址寄存器 PC 到表首的距离, 这就是偏移量, 偏移量的计算公式为:

$$\text{偏移量} = \text{表首地址} - (\text{查表指令所在的地址} + 1)$$

$$\text{上面的例子表首地址} = Y_3Y_2Y_1Y_0 + 4$$

$$\text{所以偏移量} = (Y_3Y_2Y_1Y_0 + 4) - ((Y_3Y_2Y_1Y_0 + 2) + 1) = 1$$

本例中查表的运算为从 X 求 Y , 即 $Y=f(X)$, 而 X 为满足 $0\leq X\leq 9$ 的整数。

上面的例子中, 在进入程序前, A 的内容在 $00\sim 09H$ 之间, 例如 A 中的内容为 $02H$, 它的平方为 $04H$, 以此类推, 可以根据 A 的内容查出 X 对应的平方。

`MOVC A, @+DPTR` 这条指令的应用范围较为广泛, 一般情况下, 大多使用该指令, 使用该指令时不必计算偏移量, 使用该指令的优点是表格可以设在 64 KB 程序存储器空间内的任何地方, 而不必像 `MOVC A, @A+PC` 那样只设在 PC 下面的 256 个单元中, 所以使用较方便。该指令的缺点在于如果 DPTR 已被使用, 则在进入查表以前必须保护 DPTR, 并且结束以后恢复 DPTR, 例 4-5 的子程序可改成如下形式:

```

PUSH DPH                ;保存 DPH
PUSH DPL                ;保存 DPL
MOV DPTR, #TAB1

```

```

MOV C, A, @A + DPTR
POP    DPL                ;恢复 DPL
POP    DPH                ;恢复 DPH
RET
TAB1:  DB  00H,01H,04H,09H,10H    ;0~9 的平方表
        DB  19H,24H,31H,40H,51H

```

例 4-6 在一个以 MCS-51 为核心的温度控制器中,温度传感器输出的电压与温度为非线性关系,传感器输出的电压已由 A/D 转换为 10 位二进制数。根据测得的不同温度下的电压值数据构成一个表,表中放温度值 y , x 为电压值数据。设测得的电压值 x 放入 R2R3 中,根据电压值 x ,查找对应的温度值 y ,仍放入 R2R3 中。本例的 x 和 y 均为双字节无符号数。程序如下:

```

LTB2:  MOV    DPTR, # TAB2
        MOV    A, R3
        CLR    C
        RLC    A
        MOV    R3, A
        XCH    A, R2
        RLC    A
        XCH    R2, A
        ADD    A, DPL                ;(R2R3)+(DPTR)→(DPTR)
        MOV    DPL, A
        MOV    A, DPH
        ADDC   A, R2
        MOV    DPH, A
        CLR    A
        MOVC   A, @A + DPTR        ;查第一字节
        MOV    R2, A                ;第一字节存入 R2 中
        CLR    A
        INC    DPTR
        MOVC   A, @A + DPTR        ;查第二字节
        MOV    R3, A                ;第二字节存入 R3 中
        RET
TAB2:  DW ..., ...                ;温度值表

```

以上程序中,由于使用指令 `MOVC A, @A + DPTR`,表 TAB2 可放入 64 KB 程序存储器空间的任何位置,此外表格的长度可大于 256 B。

例 4-7 设有 1 个巡回检测报警装置,需对 16 路输入进行检测,每路有 1 个最大允许值,为双字节数。装置运行时,需根据测量的路数,找出每路的最大

允许值。看输入值是否大于最大允许值,如大于就报警。下面根据上述要求,编制 1 个查表程序。

取路数为 $x(0 \leq x \leq 15)$, y 为最大允许值,放在表格中。设进入查表程序前,路数 x 已放于 R2 中,查表后最大值 y 放于 R3R4 中。本例中的 x 为单字节数, y 为双字节数。查表程序如下:

```
TB3:  MOV  A,R2
      ADD  A,R2                ;(R2)*2→(A)
      MOV  R3,A                ;保存指针
      ADD  A,#6                ;加偏移量
      MOVC A,@A+PC             ;查第一字节
      XCH  A,R3
      ADD  A,#3
      MOVC A,@A+PC             ;查第二字节
      MOV  R4,A
      RET

TAB3: DW   1520,3721,42615,7580    ;最大值表
      DW   3483,32657,883,9943
      DW   10000,40511,6758,8931
      DW   4468,5871,13284,27808
```

上述查表程序是有限制的,表格长度不能超过 256 B,且表格只能存放于 MOVC A,@A+PC 指令以下的 256 个单元中,如果表格的长度超过 256 B,且需要把表格放在 64 KB 程序存储器空间的任何地方,则应使用指令“MOVC A,@A+DPTR”且对 DPH、DPL 进行运算,求出表目的地址。

4.3.4 关键字查找程序设计

关键字查找实际就是在表中查找关键字的操作,也称为数据检索。数据检索有两种方法,即顺序检索和对分检索。

1. 顺序检索

如果要检索的表是无序的,检索时只能从第 1 项开始逐项顺序查找,判断所取数据是否与关键字相等。

例 4-8 从 50 B 的无序表中查找 1 个关键字 $\times \times H$ 。

```
ORG 1000H
MOV 30H,# $\times \times H$     ;关键字  $\times \times H$  送 30H 单元
MOV R1,#50            ;查找次数送 R1
MOV A,#14             ;修正值送 A
MOV DPTR,#TAB4        ;表首地址送 DPTR
```

```

LOOP:  PUSH A
        MOVCA,@ A+PC      ;查表结果送 A
        CJNE A,40H,LOOP1   ;(40H)不等于关键字则转 LOOP1
        MOV  R2,DPH        ;已查到关键字,把该字的地址送 R2,R3
        MOV  R3,DPL
DONE:   RET
LOOP1:  POP  A              ;修正值弹出
        INC  A              ;A+1 → A
        INC  DPTR           ;修改数据指针 DPTR
        DJNZ R1,LOOP        ;R1≠0,未查完,继续查找
        MOV  R2,#00H        ;R1=0,清 0 R2 和 R3
        MOV  R3,#00H        ;表中 50 个数已查完
        AJMP DONE          ;从子程序返回
TAB4:   DB ..., ..., ...   ;50 个无序数据表

```

2. 对分检索

对分检索的前提是检索的数据表已经排好序,以便于按照对分原则取数进行关键字比较。如何进行数据的排序,将在本节稍后介绍。

对分检索的方法:取数据表中间位置的数与关键字进行比较,如相等,则查找到;如果所取的数大于关键字,则下次对分检索的范围是从数据区起点到本次取数。如果取数小于关键字,则下次对分检索的范围是从本次取数数据区起点到数据区终点。依此类推,逐渐缩小检索范围,减少次数,大大提高了查找速度。

4.3.5 数据极值查找程序设计

数据极值查找就是在指定的数据区中找出最大值(或最小值)。

极值查找操作的主要内容是进行数值大小的比较,从这批数据中找出最大值(或最小值)并存于某一单元中。

例 4-9 片内 RAM 中存放一批数据,查找出最大值并存放于首地址中。设 R0 中存首地址,R2 中存放字节数,程序框图如图 4-1 所示。

程序如下:

```

        MOV  R2,n          ;n 为要比较的数据字节数
        MOV  A,R0           ;存首地址指针
        MOV  R1,A
        DEC  R2
        MOV  A,@R1
LOOP:   MOV  R3,A

```

```

DEC    R1
CLR    C
SUBB   A,@R1      ;两个数比较
JNC    LOOP1      ;C=0,A 中的数大,跳 LOOP1
MOV    A,@R1      ;C=1,则大数送 A
SJMP   LOOP2
LOOP1: MOV    A,R3
LOOP2: DJNZ   R2, LOOP1      ;是否比较结束?
      MOV    @R0, A          ;存最大数
      RET
    
```

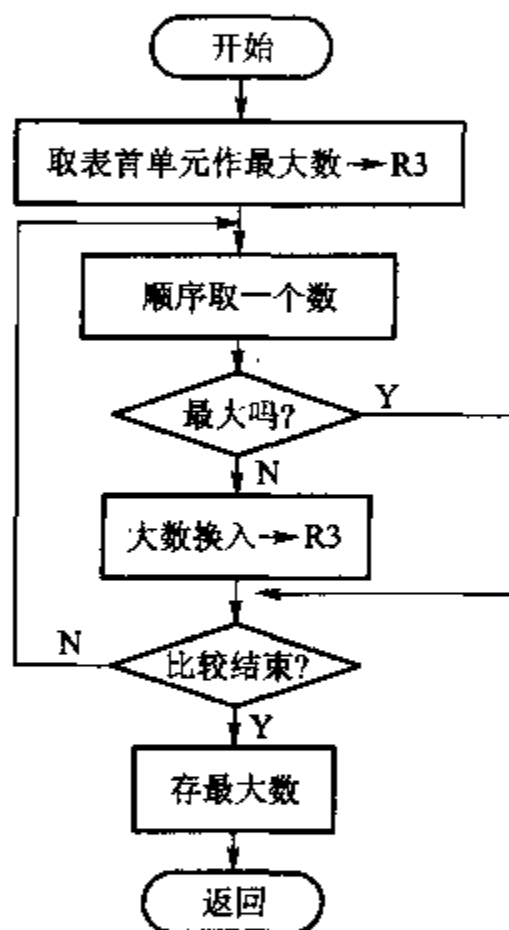


图 4-1 查找单字节无符号最大数程序框图

4.3.6 数据排序程序设计

数据排序就是将一批数由小到大(升序)排列,或由大到小(降序)排列。下面介绍无符号数据升序的排序程序设计。

最常用的数据排序算法是冒泡法。冒泡法是相邻数互换的排序方法,因其过程类似水中气泡上浮,故称冒泡法。排序时从前向后进行相邻两个数的比较,如果数据的大小次序与要求的顺序不符时,就将两个数互换,否则顺序符合要求不互换。如果进行升序排序,应通过这种相邻数互换方法,使小数向前移,大数向后移。如此从前向后进行一次次相邻数互换(冒泡),就会把这一批数据的最

大数排到最后,次大数排在倒数第二的位置,从而实现了这一批数据由小到大的排列。

假设有 7 个原始数据的排列顺序为:6、4、1、2、5、7、3。第 1 次冒泡的过程是:

6、4、1、2、5、7、3	;原始数据的排列
4、6、1、2、5、7、3	;逆序,互换
4、1、6、2、5、7、3	;逆序,互换
4、1、2、6、5、7、3	;逆序,互换
4、1、2、5、6、7、3	;逆序,互换
4、1、2、5、6、7、3	;正序,不互换
4、1、2、5、6、3、7	;逆序,互换,第 1 次冒泡结束

如此进行,各次冒泡的结果如下:

第 1 次冒泡结果:4、1、2、5、6、3、7

第 2 次冒泡结果:1、2、4、5、3、6、7

第 3 次冒泡结果:1、2、4、3、5、6、7

第 4 次冒泡结果:1、2、3、4、5、6、7 ;已完成排序

第 5 次冒泡结果:1、2、3、4、5、6、7

第 6 次冒泡结果:1、2、3、4、5、6、7

由上面的冒泡法可以看出,对于 n 个数,理论上应进行 $(n-1)$ 次冒泡才能完成排序,但实际上有时不到 $(n-1)$ 次就已完成排序。例如,上面的 7 个数,应进行 6 次冒泡,但实际上第 4 次冒泡时就已经完成了排序。如何来判定排序是否已经完成,就是看各次冒泡中是否有互换发生,如果有数据互换,则排序还没完成;否则就表示已经排好序。在程序设计中,常使用设置互换标志的方法,该标志的状态表示在 1 次冒泡中是否有互换进行。下面介绍具体的采用冒泡法的排序程序设计。

例 4-10 一批单字节无符号数,以 R0 为首地址指针,R2 中为字节数,将这批数进行升序排列。程序框图如图 4-2 所示。

程序如下:

```

SORT:  MOV  A,  R0
        MOV  R1,  A
        MOV  A,  R2          ;字节数送入 R5
        MOV  R5,  A
        CLR  F0              ;互换标志位 F0 清零
        DEC  R5
        MOV  A,  @R1
LOOP:  MOV  R3,  A
        INC  R1

```

```

CLR    C
MOV    A, @R1      ;比较大小
SUBB   A, R3
JNC    LOOP1
SETB   F0          ;交换标志位 F0 置 1
MOV    A, R3;
XCH    A, @R1      ;两个数互换
DEC    R1
XCH    A, @R1
INC    R1
LOOP1: MOV    A, @R1
DJNZ   R5, LOOP
JB     F0, SORT
RET
    
```

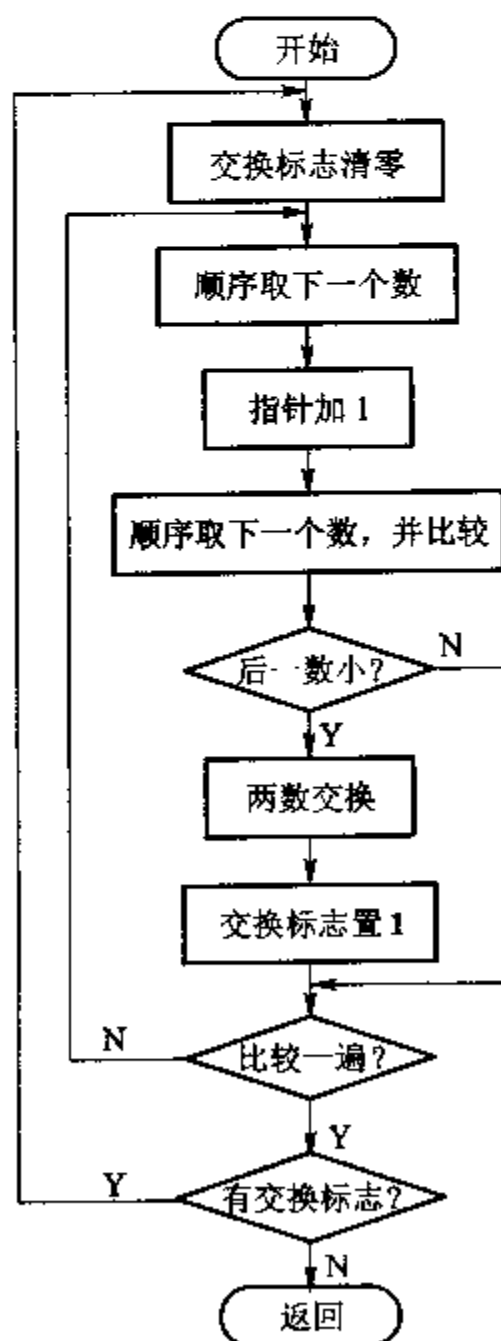


图 4-2 单字节无符号数排序程序框图

4.3.7 分支转移程序设计

分支转移程序的特点是程序中含有转移指令,转移指令又分为无条件转移和有条件转移,因此分支程序也可分为无条件分支转移程序和有条件分支转移程序。无条件分支转移程序简单,这里不再讨论。有条件分支转移程序按结构类型来分,又分为单分支转移结构和多分支转移结构。

(1) 单分支转移结构:程序的判别仅有两个出口,两者选一,称为单分支选择结构,单分支转移在程序设计中应用极为普遍,单分支转移的程序设计一般根据运算结果的状态标志,用条件判断指令来选择并转移。

例 4-11 求单字节有符号数的二进制补码
正数补码是其本身,负数补码是其反码加 1。因此,程序应首先判断被转换数的符号,负数进行转换,正数即为补码。

设二进制数放在累加器 A 中,其补码放回 A 中,程序框图如图 4-3 所示。

参考程序如下:

CMPT:	JNB	Acc. 7, RETURN	; (A) > 0, 不需转换
	MOV	C, Acc. 7	; 符号位保存
	CPL	A	; (A) 求反, 加 1
	ADD	A, #1	
	MOV	Acc. 7, C	; 符号位存在 A 的最高位
RETURN, RET			

此外,单分支选择结构还有如图 4-4、图 4-5 等所示的几种形式:

(2) 多分支转移结构:当程序的判别部分有 2 个以上的出口流向时,为多分支转移结构。多分支转移结构常见的两种形式,如图 4-6 和图 4-7 所示。

MCS-51 的指令系统提供了非常有用的两种多分支选择指令:

间接转移指令: JMP @A+DPTR;

比较转移指令: CJNE A, direct, rel;

CJNE A, #data, rel;

CJNE Rn, #data, rel;

CJNE @Ri, #data, rel;

为分支转移结构程序的编写提供了方便。

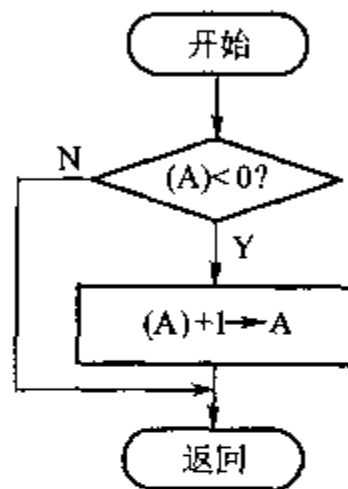


图 4-3 求单字节有符号二进制数补码的框图

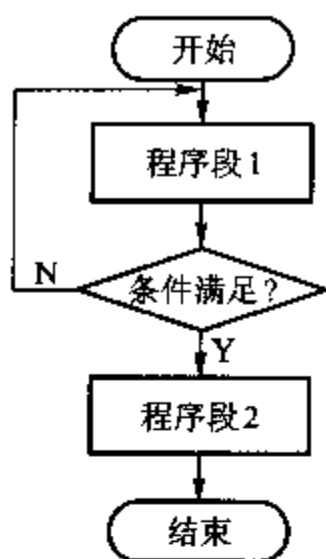


图 4-4 单分支转移结构 2

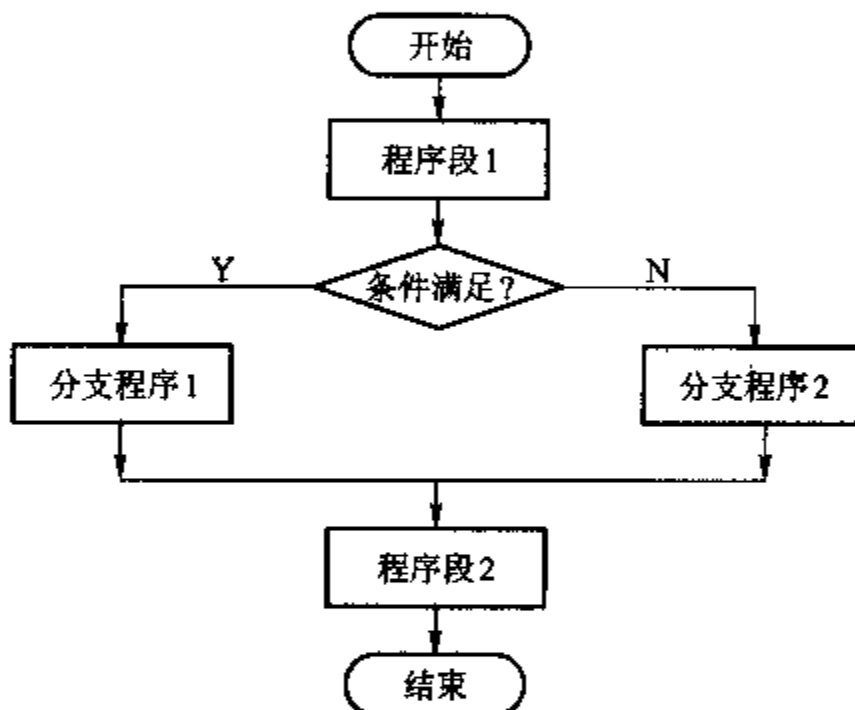


图 4-5 单分支转移结构 3

间接转移指令“`JMP @A+DPTR`”由数据指针 `DPTR` 决定多分支转移程序的首地址,由累加器 `A` 的内容动态地选择对应的分支程序。

4 条比较转移指令 `CJNE` 能对 2 个欲比较的单元内容进行比较,当不相等时,程序作相对转移,并能指出其大小,以备作第二次判断;若两者相等,则程序按顺序往下执行。

最简单的分支转移程序的设计,一般常采用逐次比较法,就是把所有不同的情况一个一个的进行比较,发现符合就转向对应的处理程序。这种方法的主要缺点是程序太长,有 n 种可能的情况,就需有 n 个判断和转移。

例 4-12 求符号函数的值。

符号函数定义如下:

$$Y = \begin{cases} 1 & \text{当 } X > 0 \\ 0 & \text{当 } X = 0 \\ -1 & \text{当 } X < 0 \end{cases}$$

X 存放在 40H 单元, Y 存放在 41H 单元, 程序框图如图 4-6 所示。

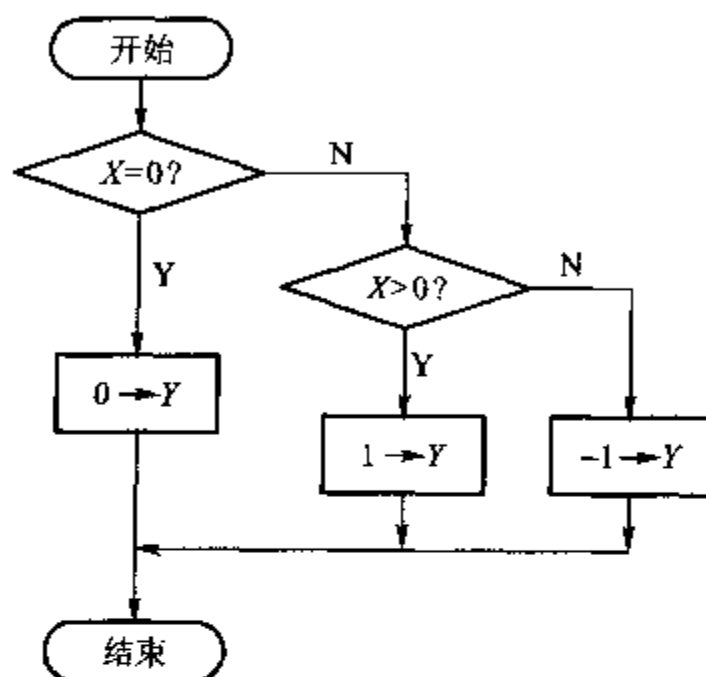


图 4-6 多分支选择结构 1

程序如下:

```

SIGNFUC:  MOV  A,40H
          CJNE A,#00H,NZEAR
          AJMP  NEG
T:
NZEAR:    JB   Acc.7, POSI
          MOV  A,#01H
          AJMP  NEG
T:
POSI:     MOV  A,#81H
NEG:      MOV  41H, A
          END
    
```

在实际的应用中,经常遇到的图 4-7 结构形式的分支转移程序的设计,即在不少应用场合,需根据某一单元的内容是 0,1,...,n,来分别转向处理程序 0,处理程序 1,...,处理程序 n。一个典型的例子就是当单片机系统中的键盘按下时,就会得到一个键值,根据不同的键值,跳向不同的键处理程序入口。对于这种情况,可用直接转移指令(LJMP 或 AJMP 指令)组成一个转移表,然后把该单元的内容读入累加器 A,转移

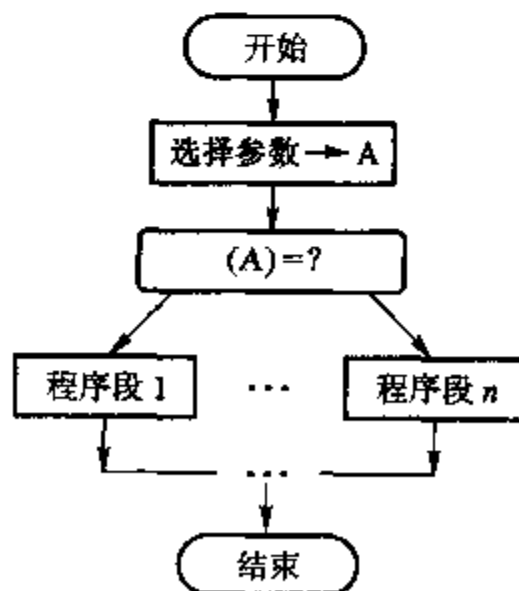


图 4-7 多分支选择结构 2

表首地址放入 DPTR 中,再利用间接转移指令实现分支转移。

例 4-13 根据寄存器 R2 的内容,转向各个处理程序 PRGX($X=0\sim n$)。

(R2)=0,转 PRG0

(R2)=1,转 PRG1

⋮

(R2)= n , 转 PRG n

程序如下:

```
JMP6:  MOV  DPTR, #TAB5      ;转移表首地址送 DPTR
        MOV  A, R2            ;分支转移参量送 A
        MOV  B, #03H          ;乘数 3 送 B
        MUL  AB               ;分支转移参量乘 3
        MOV  R6, A             ;乘积的低 8 位暂存 R6 中
        MOV  A, B              ;乘积的高 8 位送 A
        ADD  A, DPH            ;乘积的高 8 位加到 DPH 中
        MOV  DPH, A
        MOV  A, R6
        JMP  @A+DPTR           ;多分支转移选择
        ⋮
TAB5:   LJMP PRG0              ;多分支转移表
        LJMP PRG1
        ⋮
        LJMP PRGn
```

R2 中的分支转移参量乘 3 是由于长跳转指令 LJMP 要占 3 个单元。本例程序可位于 64 KB 程序存储器空间的任何区域。

对分支转移数目超过 256 个时,必须用 2 B 来存放分支转移参数,并需对分支转移参数的高位字节进行处理,即把它加到 DPH 中去。

例 4-14 根据 31H(高位字节)、30H(低位字节)的内容(分支转移参量)转向不同处理程序。程序如下:

```
JMP7:   MOV  DPTR, #TAB6      ;分支转移表首地址送 DPTR
        MOV  A, 30H           ;分支转移参量低字节送 A
        MOV  B, #3             ;参量低字节乘 3
        MUL  AB                ;乘积低字节送 R3
        MOV  R3, A             ;乘积低字节送 R3
        MOV  A, B              ;乘积高字节送 A
        ADD  A, DPH            ;DPH 1 高字节内容
        MOV  DPH, A
        MOV  A, 31H           ;参量高字节送 A 并乘 3
```

```
MOV B, #3
MUL AB
ADD A, DPH          ; DPH+乘积低字节内容
MOV DPH, A          ;和送 DPH
MOV A, R3           ;参量低字节送 A
JMP @A+DPTR         ;跳相应的分支
:
TAB6: LJMP PRG0      ;分支转移表
      LJMP PRG1
      :
      LJMP PRGn
```

4.3.8 循环程序设计

循环程序结构的特点是程序中含有可以反复执行的程序段,该程序段通常称为循环体。例如求 100 个数的累加和,则没有必要连续安排 100 条加法指令,可以只用 1 条加法指令并使其循环执行 100 次。因此循环程序不仅可以大大缩短程序长度和使程序所占的内存单元数量少,也能使程序结构紧凑和可读性变好。

1. 循环程序的结构

循环结构程序主要由以下 4 部分组成。

(1) 循环初始化:循环初始化程序段用于完成循环前的准备工作。例如,循环控制计数初值的设置、地址指针的起始地址的设置、为变量预置初值等。

(2) 循环处理:这是循环程序结构的核心部分,完成实际的处理工作,是需反复循环执行的部分,故又称循环体。这部分程序的内容,取决于实际处理问题的本身。

(3) 循环控制:在重复执行循环体的过程中,不断修改循环控制变量,直到符合结束条件,就结束循环程序的执行。循环结束控制方法分为循环计数控制法和条件控制法。

(4) 循环结束:这部分是对循环程序执行的结果进行分析、处理和存放。

上面介绍的 4 部分有时能较明显地划分,有时则相互包含,不一定能明显区分。

2. 循环结构的控制

根据循环控制部分不同,循环程序结构可分为循环计数控制结构和条件控制结构。图 4-8 是计数循环控制结构,图 4-9 是条件控制结构。

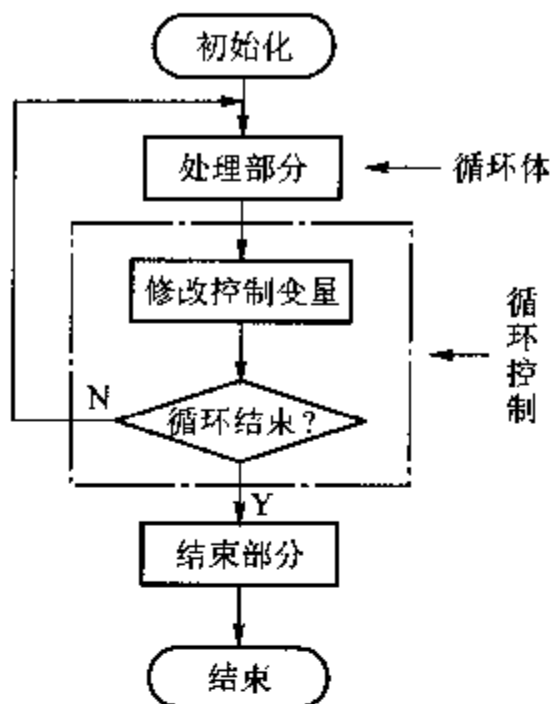


图 4-8 计数循环控制结构

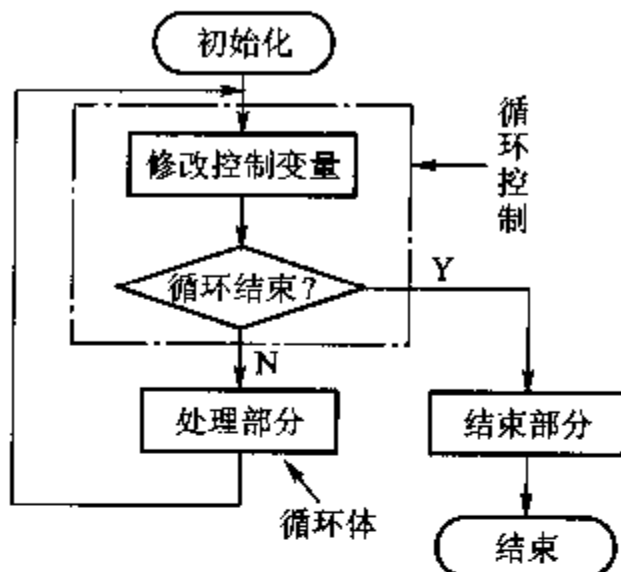


图 4-9 条件控制结构

(1) 计数循环结构:计数循环控制结构是依据计数器的值来决定循环次数,一般为减“1”计数器,计数器减到“0”时,结束循环。计数器的初值是在初始化时设定。

MCS-51 的指令系统提供了功能极强的循环控制指令:

DJNZ Rn,rel ;以工作寄存器作控制计数器

DJNZ direct,rel ;以直接寻址单元作控制计数器

例如,计算 n 个数据的和,计算公式为

$$y = \sum_{i=1}^n X_i$$

如果直接按这个公式编制程序,则 $n = 100$ 时,需编写连续的 100 次加法。这样程序将太长,并且对于 n 可变时,将无法编制出顺序程序。可见,这个公式要改写为易于实现的形式:

$$\begin{cases} y_i = 0; i = 1 \\ y_{i+1} = y_i + x; i \leq n \end{cases}$$

当 $i = n$ 时, y_{n+1} 即为所求的 n 个数据之和 y 。在用计算机程序来实现时, y_i 是 1 个变量,这可用下式表示:

$$\begin{cases} 0 \rightarrow y; 1 \rightarrow i \\ y + x_i \rightarrow y; i + 1 \rightarrow i; i \leq n \end{cases}$$

按这个公式,可以很容易地画出相应的程序框图,如图 4-10 所示。

从这个框图中,也可以看出循环程序的基本结构。

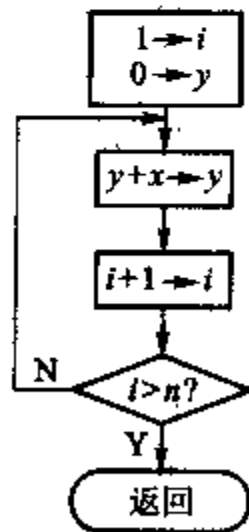


图 4-10 求数据和程序框图

例 4-15 如果 x_i 均为单字节数, 并按 i 顺序存放在 MCS-51 的内部 RAM 从 50H 开始的单元中, n 放在 R2 中, 现将要求的和(双字节)放在 R3R4 中, 程序如下。

```

ADD1:  MOV  R2, #n           ;加法次数 n 送 R2
        MOV  R3, #0
        MOV  R4, #0
        MOV  R0, #50H
LOOP:  MOV  A, R4
        ADD  A, @R0
        MOV  R4, A
        INC  R0
        CLR  A
        ADDC A, R3
        MOV  R3, A
        DJNZ R2, LOOP       ;判断加法循环次数是否已到?
        END

```

在这里, 用寄存器 R2 作为计数控制变量, R0 作为变址单元, 用它来寻址 x_i 。一般来说, 循环工作部分中的数据应该用间接方式来寻址, 如这里用:

```
ADD  A, @R0
```

计数控制方法只有在循环次数已知的情况下才适用。对循环次数未知的问题, 不能用循环次数来控制。往往需要根据某种条件来判断是否应该终止循环。

(2) 条件控制结构

例 4-16 设有一串字符, 依次存放在内部 RAM 从 30H 单元开始的连续单元中, 该字符串以 0AH 为结束标志, 编写测试字符串长度的程序。

本例采用逐个字符依次与“0AH”比较的方法。为此设置一个长度计数器和一个字符串指针。长度计数器用来累计字符串的长度, 字符串指针用于指定字符。如果指定字符与“0AH”不相等, 则长度计数器和字符串指针都加 1, 以便继续往下比较; 如果比较相等, 则表示该字符为“0AH”, 字符串结束, 长度计数器的值就是字符串的长度。程序如下:

```

        MOV  R4, #0FFH       ;长度计数器初值送 R4
        MOV  R1, #2FH        ;字符串指针初值送 R1
NEXT:  INC  R4
        INC  R1
        CJNE @R1, #0AH, NEXT ;比较, 不等则进行下一个字符比较
        END

```

前面介绍的两个例子都是在一个循环程序中不再包含其它循环程序, 则称该循环程序为单循环程序。如果一个循环程序中包含了其它循环程序, 则称为

多重循环程序。这也是经常遇到的实际问题。

最常见的多重循环是由 DJNZ 指令构成的软件延时程序,它是常用的程序之一。

例 4-17 50 ms 延时程序。

延时程序与 MCS-51 指令执行时间有很大的关系。在使用 12 MHz 晶振时,1 个机器周期为 $1\mu\text{s}$,执行 1 条 DJNZ 指令的时间为 $2\mu\text{s}$ 。这时,可用双重循环方法写出如下的延时 50 ms 的程序:

```
DEL:  MOV R7, #200
      DEL1: MOV R6, #125
      DEL2: DJNZ R6, DEL2      ;125 * 2 = 250  $\mu\text{s}$ 
            DJNZ R7, DEL1      ;0.25 ms * 200 = 50 ms
      RET
```

以上延时程序不太精确,它没有考虑到除“DJNZ R6, DEL2”指令外的其它指令的执行时间,如把其它指令的执行时间计算在内,它的延时时间为:

$$(250 + 1 + 2) * 200 + 1 = 50.301\text{ ms}$$

如果要求比较精确的延时,可按如下修改:

```
DEL:  MOV R7, #200
      DEL1: MOV R6, #123
            NOP
      DEL2: DJNZ R6, DEL2 ;2 * 123 + 2 = 248  $\mu\text{s}$ 
            DJNZ R7, DEL1 ;(248 + 2) * 200 + 1 = 50.001 ms
      RET
```

它的实际延时时间为 50.001 ms,但要注意。用软件实现延时程序,不允许有中断,否则将严重影响定时的准确性。

对于需延时更长的时间,可采用更多重的循环,如 1s 延时,可用三重循环。

4.3.9 码制转换程序设计

在单片机应用程序的设计中,经常涉及各种码制的转换问题。在单片机系统内部进行数据计算和存储时,经常采用二进制码,二进制码具有运算方便、存储量小的特点。在输入/输出中,按照人的习惯均采用代表十进制数的 BCD 码(用 4 位二进制数表示的十进制数)表示。此外,打印机要打印某数字字符,则需要将该数字的二进制码转换为该字符的 ASCII 码,才能送到打印机去打印。

1. 二进制码到 BCD 码的转换

十进制数常采用 BCD 码表示,BCD 码是 4 位有效编码。而 BCD 码又有两种形式:一种是 1 B 放 1 位 BCD 码,它适用于显示或输出,一种是压缩的 BCD

码,即 1 B 放 2 位 BCD 码,可以节省存储单元。

例 4-18 单字节二进制数转换为 BCD 码。

二进制数转换为 BCD 码的一般方法是把二进制数除以 1 000、100、10 等 10 的各次幂,所得的商即为千、百、十位数,余数为个位数。

单字节二进制数在 0~255 之间,设单字节数在累加器 A 中,转换结果的百位数放在 R3 中,十位和个位则放入 A 中。本例的程序,因为要多次用到除法指令,所以首先复习除法指令“DIV AB”的功能。除法指令完成的操作为:A 除以 B 的商放入 A 中,余数放入 B 中。

参考程序如下:

```

BINBCD1: MOV  B, #100           ;100 作为除数送入 B 中
          DIV  AB                ;十六进制数除以 100
          MOV  R3, A             ;百位数送 R3,余数在 B 中
          MOV  A, #10           ;分离十位和个位数
          XCH  A, B              ;余数送入 A 中,除数 10 在 B 中
          DIV  AB                ;分离出十位在 A 中,个位在 B 中
          SWAP A                 ;十位数交换到 A 的高 4 位
          ADD  A, B              ;十位数与个位数相加送入 A 中
          END

```

通过两次执行 DIV 除法指令,从而分离出百位数和十位数,这样的转换方法十分简单。

例 4-19 双字节二进制码转换为压缩的 BCD 码。

双字节 16 位二进制数存于 (R2R3) 中, (R4R5R6) 为转换完毕的压缩 BCD 码。

```

BINBCD2: CLR  A
          MOV  R4, A
          MOV  R5, A
          MOV  R6, A
          MOV  R7, #16
LOOP:    CLR  C
          MOV  A, R3
          RLC  A
          MOV  R3, A
          MOV  A, R2
          RLC  A
          MOV  R2, A
          MOV  A, R6
          ADDC A, R6

```

```
DA    A
MOV  R6,A
MOV  A,R5
ADDC A,R5
DA    A
MOV  R5,A
MOV  A,R4
ADDC A,R4
DA    A
MOV  R4,A
DJNZ R7,LOOP
RET
```

2. BCD 码到二进制数的转换

例 4-20 4 位 BCD 码转换成二进制数。

设 BCD 码 a_3 、 a_2 、 a_1 、 a_0 分别放在 50H~53H 单元中,转换完毕的二进制数放在 R3R4 中。

```
IDTB: MOV  R0,#50H
      MOV  R2,#3
      MOV  R3,#0
      MOV  A,@R0
      MOV  R4,A
LOOP: MOV  A,R4
      MOV  B,#10
      MUL  AB
      MOV  R4,A
      MOV  A,#10
      XCH  A,B
      XCH  A,R3
      MUL  AB
      ADD  A,R3
      XCH  A,R4
      INC  R0
      ADD  A,@R0
      XCH  A,R4
      ADDC A,#0
      MOV  R3,A
      DJNZ R2,LOOP
      RET
```

3. 二进制数与 ASCII 码之间的转换

例 4-21 4 位二进制数转换为 ASCII 代码

4 位二进制数(0~F)存放于 R2 中,转换完毕的 ASCII 代码存放于 R2 中。

分析二进制数和 ASCII 码之间的对应关系,得到:对于小于等于 9 的 4 位二进制数加 30H 得到相应的 ASCII 代码,对于大于 9 的 4 位二进制数(十六进制数 A~F)加 37H 即得到相应的 ASCII 代码。

```
BTOASC: MOV  A,R2
          ANL  A,#0FH
          ADD  A,#0F6H ;加 F6H 保证大于 10 的数把进位位置 1
          JNC  LOOP0   ;C=0,数小于 10 转 LOOP0
          ADDC A,#07H   ;C≠0 数大于 10,先加 07H,下面再加 30H
LOOP0:    ADDC A,#30H   ;小于 10 的数加 30H
          MOV  R2,A     ;转换完毕的 ASCII 码存入 R2
          RET
```

例 4-22 单字节的 2 位十六进制数转换为 ASCII 码。

(R0)为待转换的十六进制数的地址指针,转换结果存于(R0)指向的地址单元中。

```
HTA1: MOV  R0,SP
          DEC  R0
          DEC  R0
          PUSH A        ;保护累加器内容
          MOV  A,@R0     ;取出参数
          ANL  A,#0FH
          ADD  A,#14     ;PC 表偏移值
          MOVC A,@A+PC
          XCH  A,@R0     ;低位十六进制数的 ASCII 码放入堆栈中
          SWAP A
          ANL  A,#0FH
          ADD  A,#7      ;PC 到表的偏移量
          MOVC A,@A+PC
          INC  R0
          XCH  A,@R0     ;高位十六进制数的 ASCII 码放入堆栈中
          INC  R0
          XCH  A,@R0     ;低位返回地址放入堆栈,并恢复累加器内容
          INC  R0
          XCH  A,@R0
          RET
```

```
ASCTAB: DB 30H,31H,32H,33H,34H,35H,36H,37H;ASCII 码表
        DB 38H,39H,40H,41H,42H,43H,44H,45H,46H
```

说明:该子程序采用堆栈来传递参数,但这里传到子程序的参数为 1B,传回到主程序的参数为 2 B,这样堆栈的大小在调用前后是不一样的。在子程序中,必须对堆栈内的返回地址和栈指针进行修改。

例 4-23 十六进制数(0~F)的 ASCII 代码转换成 4 位二进制数。

ASCII 代码存放于 R2,转换完毕的 4 位二进制数结果存放于 R2 中。

对于小于等于 9 的数的 ASCII 码减去 30H 得 4 位二进制数,对于大于 9 的十六进制数的 ASCII 码减去 37H 即得二进制数。

```
ASCTB: PUSH PSW
        PUSH A
        MOV A,R2
        CLR C
        SUBB A,#30H
        MOV R2,A
        SUBB A,#0AH
        JC SB10
        XCH A,R2
        SUBB A,#07H
        MOV R2,A
SB10: POP A
        POP PSW
        RET
```

思考题及习题

1. 用于程序设计的语言分为哪几种?它们各有什么特点?
2. 说明伪指令的作用。“伪”的含义是什么?
3. 解释下列术语:“手工汇编”、“机器汇编”、“交叉汇编”以及“反汇编”。
4. 下列程序段经汇编后,从 1 000H 开始的各有关存储单元的内容将是什么?

```
ORG 1 000H
TAB1 EQU 1 234H
TAB2 EQU 3 000H
DB "MAIN"
DW TAB1,TAB2,70H
```

5. 设计子程序时注意哪些问题?
6. 试编写 1 个程序,将内部 RAM 中 45H 单元的高 4 位清 0,低 4 位置 1。

7. 已知程序执行前有 $A = 02H$, $SP = 42H$, $(41H) = FFH$, $(42H) = FFH$ 。下述程序执行后, 请问 $A = (\quad)$; $SP = (\quad)$; $(41H) = (\quad)$; $(42H) = (\quad)$; $PC = (\quad)$ 。

```
POP DPH
POP DPL
MOV DPTR, #3000H
RL A
MOV B, A
MOVC A, @A + DPTR
PUSH A
MOV A, B
INC A
MOVC A, @A + DPTR
PUSH A
RET
ORG 3000H
DB 10H, 80H, 30H, 80H, 50H, 80H
```

8. 计算下面子程序中指令的偏移量和程序执行的时间(晶振频率为12 MHz)。

7B0F	MOV R3, #15	:1 个机器周期
7CFF	DL1: MOV R4, #255	:1 个机器周期
8B90	DL2: MOV P1, R3	:2 个机器周期
DC	DJNZ R4, DL2	:2 个机器周期
DB	DJNZ R3, DL1	:2 个机器周期
22	RET	:2 个机器周期

9. 假定 $A = 83H$, $(R0) = 17H$, $(17H) = 34H$, 执行以下指令:

```
ANL A, #17H
ORL 17H, A
XPL A, @R0
CPL A
```

后, A 的内容为()。

10. 试编写程序, 查找在内部 RAM 的 $30H \sim 50H$ 单元中是否有 $0AAH$ 这一数据。若有, 则将 $51H$ 单元置为“ $01H$ ”; 若未找到, 则将 $51H$ 单元置为“ $00H$ ”。

11. 试编写程序, 查找在内部 RAM 的 $20H \sim 40H$ 单元中出现“ $00H$ ”这一数据的次数。并将查找到的结果存入 $41H$ 单元。

12. 若 $SP = 60H$, 标号 LABEL 所在的地址为 $3456H$ 。LCALL 指令的地址为 $2000H$, 执行如下指令:

```
2000H LCALL LABEL
```

后, 堆栈指针 SP 和堆栈内容发生了什么变化? PC 的值等于什么? 如果将指令 LCALL 直接换成 ACALL 是否可以? 如果换成 ACALL 指令, 可调用的地址范围是什么?

第 5 章 MCS-51 的中断系统

MCS-51 单片机片内的中断系统主要用于实时测控,即要求单片机能及时地响应和处理单片机外部或内部事件所提出的中断请求。由于这些中断请求都是随机发出的,如果采用定时查询方式来处理这些中断请求,则单片机的工作效率低,且得不到实时处理。因此,MCS-51 单片机要实时处理这些中断请求,就必须采用具有中断处理功能的部件——中断系统来完成。

本章介绍 MCS-51 单片机片内中断系统的工作原理及应用,首先介绍有关中断的一些基本概念。

5.1 中断的概念

当 MCS-51 单片机的 CPU 正在处理某件事情(例如,正在执行主程序)的时候,单片机外部或内部发生的某一事件(如外部设备产生的一个电平的变化,一个脉冲沿的发生或内部计数器的计数溢出等)请求 CPU 迅速去处理,于是,CPU 暂时中止当前的工作,转到中断服务处理程序处理所发生的事件。中断服务处理程序处理完该事件后,再回到原来被中止的地方,继续原来的工作(例如,继续执行被中断的主程序),这称为中断。CPU 处理事件的过程,称为 CPU 的中断响应过程,如图 5-1 所示。对事件的整个处理过程,称为中断处理(或中断服务)。

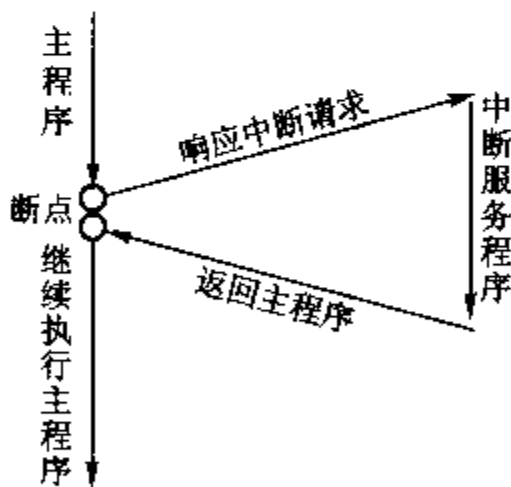


图 5-1 中断响应过程

能够实现中断处理功能的部件称为中断系统;产生中断的请求源称为中断请求源(或中断源);中断源向 CPU 提出的处理请求,称为中断请求(或中断申请)。当 CPU 暂时中止正在执行的程序,转去执行中断服务程序时,除了硬件自动把断点地址(16 位程序计数器 PC 的值)压入堆栈之外,用户应注意保护有关的工作寄存器、累加器、标志位等信息,这称为保护现场。在完成中断服务程

序后,恢复有关的工作寄存器、累加器、标志位内容,这称为恢复现场。最后执行中断返回指令 RETI,从堆栈中自动弹出断点地址到 PC,继续执行被中断的程序,这称为中断返回。

如果没有中断技术,CPU 的大量时间可能会浪费在原地踏步的查询操作上,或者采用定时查询,即不论有无中断请求,都要定时去查询。采用中断技术完全消除了 CPU 在查询方式中的等待现象,大大地提高了 CPU 的工作效率。由于中断工作方式的优点极为明显,因此在单片机的硬件结构中都带有中断系统。

5.2 MCS-51 中断系统的结构

MCS-51 单片机的中断系统有 5 个中断请求源,具有 2 个中断优先级,可实现 2 级中断服务程序嵌套。用户可以用关中断指令“CLR EA”来屏蔽所有的中断请求,也可以用开中断指令“SET EA”来允许 CPU 接收中断请求;每一个中断源可以用软件独立地控制为允许中断或关中断状态;每一个中断源的中断级别均可用软件来设置。

MCS-51 的中断系统结构示意图如图 5-2 所示。下面将从应用的角度来说明 MCS-51 的中断系统的工作原理和编程方法。

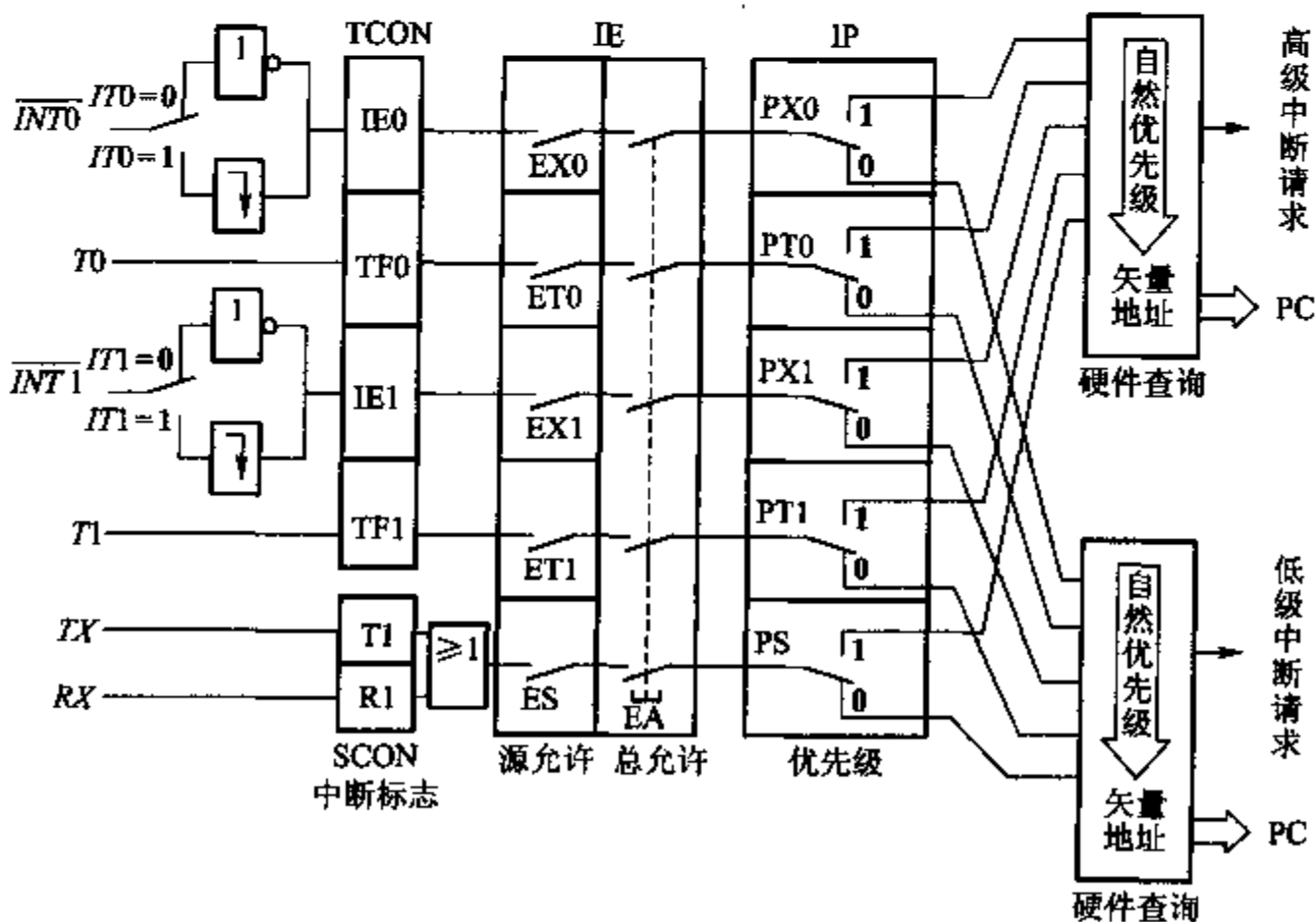


图 5-2 MCS-51 的中断系统结构

5.3 中断请求源

MCS-51 中断系统共有 5 个中断请求源(见图 5-2),它们是:

- (1) $\overline{\text{INT0}}$ ——外部中断请求 0,由 $\overline{\text{INT0}}$ 引脚输入,中断请求标志为 IE0。
- (2) $\overline{\text{INT1}}$ ——外部中断请求 1,由 $\overline{\text{INT1}}$ 引脚输入,中断请求标志为 IE1。
- (3) 定时器/计数器 T0 溢出中断请求,中断请求标志为 TF0。
- (4) 定时器/计数器 T1 溢出中断请求,中断请求标志为 TF1。
- (5) 串行口中断请求,中断请求标志为 TI 或 RI。

这些中断请求源的中断请求标志位分别由特殊功能寄存器 TCON 和 SCON 的相应位锁存。

TCON 为定时器/计数器的控制寄存器,字节地址为 88H,可位寻址。该寄存器中既有定时器/计数器 T0 和 T1 的溢出中断请求标志位 TF1 和 TF0,也包括了有关外部中断请求标志位 IE1 与 IE0。其格式如图 5-3 所示:

	D7	D6	D5	D4	D3	D2	D1	D0	
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88H
位地址	8FH	-	8DH	-	8BH	8AH	89H	88H	

图 5-3 TCON 中的中断请求标志位

TCON 寄存器中与中断系统有关的各标志位的功能如下:

- (1) IT0——选择外部中断请求 0 为跳沿触发方式还是电平触发方式:

$IT0=0$,为电平触发方式,加到引脚 $\overline{\text{INT0}}$ 上的外部中断请求输入信号为低电平有效。

$IT0=1$,为跳沿触发方式,加到引脚 $\overline{\text{INT0}}$ 上的外部中断请求输入信号电平从高到低的负跳变有效。

IT0 位可由软件置 1 或清 0。

- (2) IE0——外部中断请求 0 的中断请求标志位。

当 $IT0=0$,为电平触发方式,CPU 在每个机器周期的 S5P2 采样 $\overline{\text{INT0}}$ 引脚,若 $\overline{\text{INT0}}$ 引脚为低电平,则置 1IE0,说明有中断请求,否则清 0IE0。

当 $IT0=1$,即外部中断请求 0 设置为跳沿触发方式时,当第一个机器周期采样到 $\overline{\text{INT0}}$ 为低电平时,则置 1IE0。 $IE0=1$,表示外部中断 0 正在向 CPU 请求中断。当 CPU 响应该中断,转向中断服务程序时,由硬件清 0IE0。

- (3) IT1——选择外部中断请求 1 为跳沿触发方式还是电平触发方式,其意义与 IT0 类似。

(4) IE1 外部中断请求 1 的中断请求标志位,其意义与 IE0 类似。

(5) TF0—MCS-51 片内定时器/计数器 T0 溢出中断请求标志位。

当启动 T0 计数后,定时器/计数器 T0 从初值开始加 1 计数,当最高位产生溢出时,由硬件置 1TF0,向 CPU 申请中断,CPU 响应 TF0 中断时,清 0TF0,TF0 也可由软件清 0。

(6) TF1—MCS-51 片内的定时器/计数器 T1 的溢出中断请求标志位,功能和 TF0 类似。

TR1(D6 位)、TR0(D4 位)这 2 个位与中断无关,仅与定时器/计数器 T1 和 T0 有关,它们的功能将在定时器/计数器一章中介绍。

当 MCS-51 复位后,TCN 被清 0,则 CPU 关中断,所有中断请求被禁止。

SCON 为串行口控制寄存器,字节地址为 98H,可位寻址。SCON 的低 2 位锁存串行口的发送中断和接收中断的中断请求标志 TI 和 RI,其格式如图 5-4 所示:

	D7	D6	D5	D4	D3	D2	D1	D0	
SCON	-	-	-	-	-	-	TI	RI	98H
位地址	-	-	-	-	-	-	99H	98H	

图 5-4 SCON 中的中断请求标志位

SCON 中各标志位的功能如下:

(1) TI—串行口的发送中断请求标志位。CPU 将 1 B 的数据写入发送缓冲器 SBUF 时,就启动 1 帧串行数据的发送,每发送完 1 帧串行数据后,硬件自动置 1TI。CPU 响应串行口发送中断时,CPU 并不清除 TI 中断请求标志,必须在中断服务程序中用软件对 TI 标志清 0。

(2) RI—串行口接收中断请求标志位。在串行口接收完 1 个串行数据帧,硬件自动置 1RI 中断请求标志。CPU 在响应串行口接收中断时,并不清 0RI 标志,必须在中断服务程序中用软件对 RI 清 0。

5.4 中断控制

5.4.1 中断允许寄存器 IE

MCS-51 的 CPU 对中断源的开放或屏蔽,是由片内的中断允许寄存器 IE 控制的。IE 的字节地址为 A8H,可进行位寻址。其格式如图 5-5 所示:

中断允许寄存器 IE 对中断的开放和关闭实现 2 级控制。所谓 2 级控制,就是有 1 个总的开关中断控制位 EA(IE.7 位),当 EA=0 时,所有的中断请求被屏蔽,

	D7	D6	D5	D4	D3	D2	D1	D0	
IE	EA	-	-	ES	ET1	EX1	ET0	EX0	A8H
位地址	AFH	-	-	ACH	ABH	AAH	A9H	A8H	

图 5-5 中断允许寄存器 IE 的格式

CPU 对任何中断请求都不接受,称 CPU 关中断;当 $EA=1$ 时,CPU 开放中断,但 5 个中断源的中断请求是否允许,还要由 IE 中的低 5 位所对应的 5 个中断请求允许控制位的状态来决定(见图 5-5)。

IE 中各位的功能如下:

(1) EA——中断允许总控制位

$EA=0$,CPU 屏蔽所有的中断请求(CPU 关中断);

$EA=1$,CPU 开放所有中断(CPU 开中断)。

(2) ES——串行口中断允许位

$ES=0$,禁止串行口中断;

$ES=1$,允许串行口中断。

(3) ET1——定时器/计数器 T1 的溢出中断允许位

$ET1=0$,禁止 T1 溢出中断;

$ET1=1$,允许 T1 溢出中断。

(4) EX1——外部中断 1 中断允许位

$EX1=0$,禁止外部中断 1 中断;

$EX1=1$,允许外部中断 1 中断。

(5) ET0——定时器/计数器 T0 的溢出中断允许位

$ET0=0$,禁止 T0 溢出中断;

$ET0=1$,允许 T0 溢出中断。

(6) EX0——外部中断 0 中断允许位。

$EX0=0$,禁止外部中断 0 中断;

$EX0=1$,允许外部中断 0 中断。

MCS-51 复位以后,IE 被清 0,所有的中断请求被禁止。由用户程序置 1 或清 0 IE 相应的位,即可允许或禁止各中断源的中断申请。若使某一个中断源被允许中断,除了 IE 相应的位被置 1 外,还必须使 $EA=1$,即 CPU 开放中断。改变 IE 的内容,可由位操作指令来实现(即 SETB bit;CLR bit),也可用字节操作指令实现(即 MOV IE, #data;ANL IE, #data;ORL IE, #data;MOV IE, A 等)。

例 5-1 若允许片内 2 个定时器/计数器中断,禁止其它中断源的中断请求。请编写出设置 IE 的相应程序段。

(1) 用位操作指令来编写如下程序段:

CLR ES	;禁止串行口中断
CLR EX1	;禁止外部中断 1 中断
CLR EX0	;禁止外部中断 0 中断
SETB ET0	;允许定时器/计数器 T0 中断
SETB ET1	;允许定时器/计数器 T1 中断
SETB EA	;CPU 开中断

(2) 用字节操作指令来编写:

```
MOV IE, #8AH
```

或者用:

```
MOV 0A8H, #8AH ;A8H 为 IE 寄存器的地址字节地址
```

5.4.2 中断优先级寄存器 IP

MCS-51 的中断请求源有 2 个中断优先级,每一个中断请求源可由软件定为高优先级中断或低优先级中断,可实现 2 级中断嵌套,所谓 2 级中断嵌套,就是 CPU 正在执行低优先级中断的服务程序时,可被高优先级中断请求所中断,去执行高优先级中断服务程序,待高优先级中断处理完毕后,再返回低优先级中断服务程序。

2 级中断嵌套的过程如图 5-6 所示。

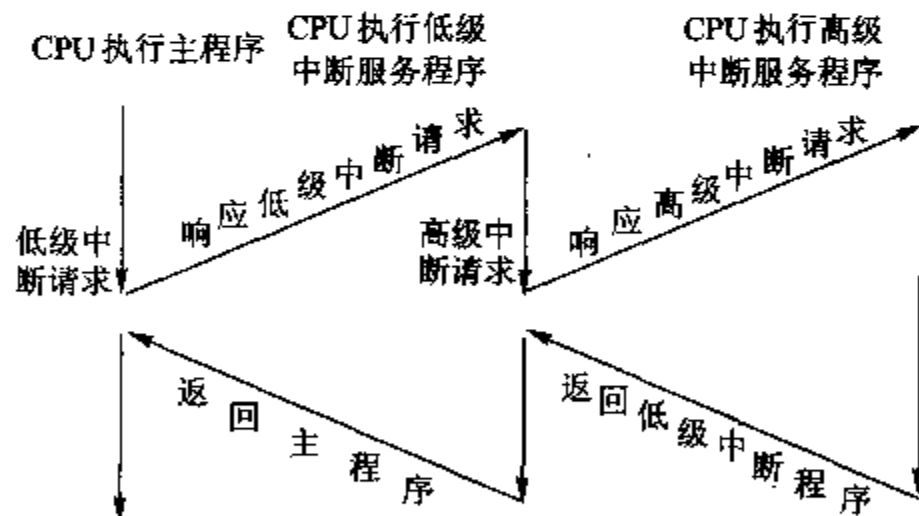


图 5-6 2 级中断嵌套

由图 5-6 可见,一个正在执行的低优先级中断程序能被高优先级的中断源所中断,但不能被另一个低优先级的中断源所中断。若 CPU 正在执行高优先级的中断,则不能被任何中断源所中断,一直执行到中断服务程序结束,遇到中断返回指令 RETI,返回主程序再执行一条指令后才能响应新的中断请求。以上所述可以归纳为下面两条基本规则:

(1) 低优先级可被高优先级中断,反之则不能。

(2) 任何一种中断(不管是高级还是低级),一旦得到响应,不会再被它的同级中断源所中断。如果某一中断源被设置为高优先级中断,在执行该中断源的中断服务程序时,则不能被任何其它中断源的中断请求所中断。

MCS-51 的片内有一个中断优先级寄存器 IP,其字节地址为 B8H,可位寻址。只要用程序改变其内容,即可进行各中断源中断级别的设置,IP 寄存器的格式如图 5-7 所示:

	D7	D6	D5	D4	D3	D2	D1	D0	
IP	-	-	-	PS	PT1	PX1	PT0	PX0	B8H
位地址	-	-	-	BCH	BBH	BAH	B9H	B8H	

图 5-7 中断优先级寄存器 IP 的格式

中断优先级寄存器 IP 各个位的含义如下:

(1) PS——串行口中断优先级控制位

PS=1,串行口中断定义为高优先级中断;

PS=0,串行口中断定义为低优先级中断。

(2) PT1——定时器 T1 中断优先级控制位

PT1=1,定时器 T1 定义为高优先级中断;

PT1=0,定时器 T1 定义为低优先级中断。

(3) PX1——外部中断 1 中断优先级控制位

PX1=1,外部中断 1 定义为高优先级中断;

PX1=0,外部中断 1 定义为低优先级中断。

(4) PT0——定时器 T0 中断优先级控制位

PT0=1,定时器 T0 定义为高优先级中断;

PT0=0,定时器 T0 定义为低优先级中断。

(5) PX0——外部中断 0 中断优先级控制位

PX0=1,外部中断 0 定义为高优先级中断;

PX0=0,外部中断 0 定义为低优先级中断。

中断优先级控制寄存器 IP 的各位都由用户程序置 1 和清 0,可用位操作指令或字节操作指令更新 IP 的内容,以改变各中断源的中断优先级。

MCS-51 复位以后,IP 的内容为 0,各个中断源均为低优先级中断。

为进一步了解 MCS-51 中断系统的优先级,简单介绍一下 MCS-51 的中断优先级结构。MCS-51 的中断系统有两个不可寻址的优先级激活触发器。其中一个指示某高优先级的中断正在执行,所有后来的中断均被阻止。另一个触发器指示某低优先级的中断正在执行,所有同级的中断都被阻止,但不阻断高优先级的中断请求。

在同时收到几个同一优先级的中断请求时,哪一个中断请求能优先得到响应,取决于内部的查询顺序。这相当于在同一个优先级内,还同时存在另一个辅助优先级结构,其查询顺序如下:

中 断 源	中 断 级 别
外部中断 0	最高 ↓ 最低
T0 溢出中断	
外部中断 1	
T1 溢出中断	
串行口中断	

由上可见,各中断源在同一个优先级的条件下,外部中断 0 的中断优先权最高,串行口中断的优先权最低。

例 5-2 设置 IP 寄存器的初始值,使得 MCS-51 的 2 个外中断请求为高优先级,其它中断请求为低优先级。

(1) 用位操作指令

```
SETB PX0      ;2 个外中断为高优先级
SETB PX1
CLR PS        ;串行口、2 个定时器/计数器为低优先级中断
CLR PT0
CLR PT1
```

(2) 用字节操作指令

```
MOV IP, #05H
```

或者用:

```
MOV 0B8H, #05H    ;B8H 为 IP 寄存器的字节地址
```

5.5 响应中断请求的条件

一个中断源的中断请求被响应,需满足以下必要条件:

- (1) CPU 开中断,即 IE 寄存器中的中断总允许位 $EA=1$ 。
- (2) 该中断源发出中断请求,即该中断源对应的中断请求标志为 1。
- (3) 该中断源的中断允许位=1,即该中断没有被屏蔽。
- (4) 无同级或更高级中断正在被服务。

中断响应就是 CPU 对中断源提出的中断请求的接受。当 CPU 查询到有效的中断请求,并满足上述条件时,紧接着就进行中断响应。

中断响应的主要过程是首先由硬件自动生成一条长调用指令 LCALL addr16。这里的 addr16 就是程序存储区中的相应的中断入口地址。例如,对于外部中断 1 的响应,产生的长调用指令为:

LCALL 0013H

生成 LCALL 指令后,紧接着就由 CPU 执行该指令。首先是将程序计数器 PC 的内容压入堆栈以保护断点,再将中断入口地址装入 PC,使程序转向响应中断请求的中断入口地址。各中断源服务程序的入口地址是固定的,如下所示:

中 断 源	入 口 地 址
外部中断 0	0003H
定时器/计数器 T0	000BH
外部中断 1	0013H
定时器/计数器 T1	001BH
串行口中断	0023H

2 个中断入口间只相隔 8B,一般情况下难以安排下一个完整的中断服务程序。因此,通常总是在中断入口地址处放置 1 条无条件转移指令,使程序执行转向在其它地址存放的中断服务程序。

中断响应是有条件的,并不是查询到的所有中断请求都能被立即响应,当遇到下列三种情况之一时,中断响应被封锁:

(1) CPU 正在处理同级的或更高优先级的中断。因为当一个中断被响应时,要把对应的中断优先级状态触发器置 1(该触发器指出 CPU 所处理的中断优先级别),从而封锁了低级中断和同级中断请求。

(2) 所查询的机器周期不是当前所正在执行指令的最后一个机器周期。作这个限制的目的在于只有在当前指令执行完毕后,才能进行中断响应,以确保当前指令完整的执行。

(3) 正在执行的指令是 RETI 或是访问 IE 或 IP 的指令。因为按 MCS-51 中断系统特性的规定,在执行完这些指令后,需要再去执行完一条指令,才能响应新的中断请求。

如果存在上述三种情况之一,CPU 将丢弃中断查询结果,不能对中断进行响应。

5.6 外部中断的响应时间

在设计者使用外部中断时,有时需考虑从外部中断请求有效(外部中断请求

标志置 1) 到转向中断入口地址所需要的响应时间。下面来讨论这个问题。

外部中断的最短响应时间为 3 个机器周期。其中中断请求标志位查询占 1 个机器周期, 而这个机器周期恰好是处于指令的最后一个机器周期, 在这个机器周期结束后, 中断即被响应, CPU 接着执行 1 条硬件子程序调用指令 LCALL 以转到相应的中断服务程序入口, 则需要 2 个机器周期。

外部中断响应的最长时间为 8 个机器周期。这种情况发生在 CPU 进行中断标志查询时, 刚好是开始执行 RETI 或是访问 IE 或 IP 的指令, 则需把当前指令执行完再继续执行 1 条指令后, 才能响应中断。执行上述的 RETI 或是访问 IE 或 IP 的指令, 最长需要 2 个机器周期。而接着再执行的 1 条指令, 按最长的指令(乘法指令 MUL 和除法指令 DIV)来算, 也只有 4 个机器周期。再加上硬件子程序调用指令 LCALL 的执行, 需要 2 个机器周期, 所以, 外部中断响应最长时间为 8 个机器周期。

如果已经在处理同级或更高级中断, 外部中断请求的响应时间取决于正在执行的中断服务程序的处理时间, 这种情况下, 响应时间就无法计算了。

这样, 在 1 个单一中断的系统里, MCS-51 单片机对外部中断请求的响应时间总是在 3~8 个机器周期之间。

5.7 外部中断的触发方式选择

外部中断的触发有 2 种触发方式: 电平触发方式和跳沿触发方式。

5.7.1 电平触发方式

若外部中断定义为电平触发方式, 外部中断申请触发器的状态随着 CPU 在每个机器周期采样到的外部中断输入线的电平变化而变化, 这能提高 CPU 对外部中断请求的响应速度。当外部中断源被设定为电平触发方式时, 在中断服务程序返回之前, 外部中断请求输入必须无效(即变为高电平), 否则 CPU 返回主程序后会再次响应中断。所以电平触发方式适合于外部中断以低电平输入而且中断服务程序能清除外部中断请求源(即外部中断输入电平又变为高电平)的情况。如何清除电平触发方式的外部中断请求源的电平信号, 将在本章的后面介绍。

5.7.2 跳沿触发方式

外部中断若定义为跳沿触发方式, 外部中断申请触发器能锁存外部中断输入

线上的负跳变。即使是 CPU 暂时不能响应,中断请求标志也不会丢失。在这种方式里,如果相继连续两次采样,一个机器周期采样到外部中断输入为高,下一个机器周期采样为低,则置 1 中断申请触发器,直到 CPU 响应此中断时,该标志才清 0。这样不会丢失中断,但输入的负脉冲宽度至少保持 12 个时钟周期(若晶振频率为 6 MHz,则为 $2\mu\text{s}$),才能被 CPU 采样到。外部中断的跳沿触发方式适合于以负脉冲形式输入的外部中断请求。

5.8 中断请求的撤消

某个中断请求被响应后,就存在着一个中断请求的撤消问题。下面按中断请求源的类型分别说明中断请求的撤消方法。

1. 定时器/计数器中断请求的撤消

定时器/计数器中断的中断请求被响应后。硬件会自动把中断请求标志位 (TF0 或 TF1) 清 0,因此定时器/计数器中断请求是自动撤消的。

2. 外部中断请求的撤消

(1) 跳沿方式外部中断请求的撤消:跳沿方式的外部中断请求的撤消,包括 2 项内容:中断标志位的清 0 和外中断信号的撤消。其中,中断标志位 (IE0 或 IE1) 的清 0 是在中断响应后由硬件自动完成的。而外中断请求信号的撤消,由于跳沿信号过后也就消失了,所以跳沿方式的外部中断请求也是自动撤消的。

(2) 电平方式外部中断请求的撤消:对于电平方式外部中断请求的撤消,中断请求标志的撤消是自动的,但中断请求信号的低电平可能继续存在,在以后的机器周期采样时,又会把已清 0 的 IE0 或 IE1 标志位重新置 1。为此,要彻底解决电平方式外部中断请求的撤消,除了标志位清 0 之外,必要时还需在中断响应后把中断请求信号引脚从低电平强制改变为高电平。为此,可在系统中增加如图 5-8 所示的电路。

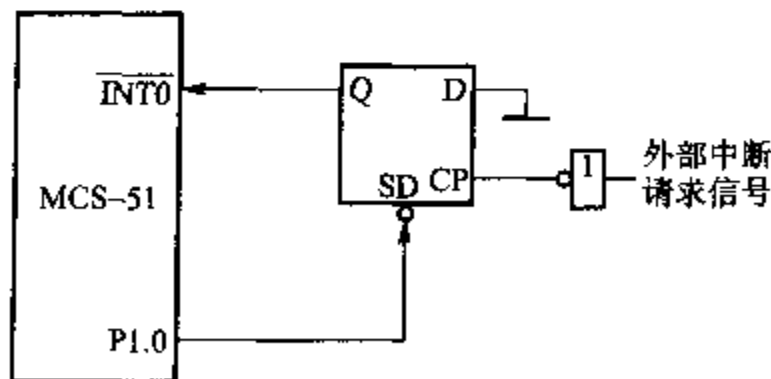


图 5-8 电平方式外部中断请求的撤消电路

由图 5-8 可见,用 D 触发器锁存外来的中断请求低电平,并通过 D 触发器

的输出端 Q 接到 $\overline{\text{INT0}}$ (或 $\overline{\text{INT1}}$)。所以,增加的 D 触发器不影响中断请求。中断响应后,为了撤消中断请求,可利用 D 触发器的直接置位端 SD 实现,把 SD 端接 MCS-51 的 P1.0 端。因此,只要 P1.0 端输出一个负脉冲就可以使 D 触发器置 1,从而撤消了低电平的中断请求信号。所需的负脉冲可通过在中断服务程序中增加如下 2 条指令得到:

```
ORI   P1, #01H      ;P1.0 为 1
ANL   P1, #0FEH      ;P1.0 为 0
```

可见,电平方式的外部中断请求信号的完全撤消,是通过软硬件相结合的方法来实现的。

3. 串行口中断请求的撤消

串行口中断请求的撤消只有标志位清 0 的问题。串行口中断的标志位是 TI 和 RI,但对这 2 个中断标志 CPU 不进行自动清 0。因为在响应串行口的中断后,CPU 无法知道是接收中断还是发送中断,还需测试这 2 个中断标志位的状态,以判定是接收操作还是发送操作,然后才能清除。所以串行口中断请求的撤消只能使用软件的方法,在中断服务程序中进行,即用如下的指令来进行串行口中断标志位的清除:

```
CLR   TI   ;清 TI 标志位
CLR   RI   ;清 RI 标志位
```

5.9 中断服务程序的设计

中断系统虽是硬件系统,但必须由相应软件配合才能正确使用。设计中断服务程序需要弄清楚以下几个问题。

1. 中断服务程序设计的任务

中断服务程序设计的基本任务有下列几条:

- (1) 设置中断允许控制寄存器 IE,允许相应的中断请求源中断。
- (2) 设置中断优先级寄存器 IP,确定并分配所使用的中断源的优先级。
- (3) 若是外部中断源,还要设置中断请求的触发方式 IT1 或 IT0,以决定采用电平触发方式还是跳沿触发方式。
- (4) 编写中断服务程序,处理中断请求。

前 3 条一般放在主程序的初始化程序段中。

例 5-3 假设允许外部中断 0 中断,并设定它为高级中断,其它中断源为低级中断,采用跳沿触发方式。在主程序中可编写如下程序段:

```
SETB  EA      ;EA 位置 1,CPU 开中断
```

```
SETB  ET0      ;ET0 位置 1, 允许外部中断 0 产生中断
SETB  PX0      ;PX0 位置 1, 外部中断 0 为高级中断
SETB  IT0      ;IT0 位置 1, 外部中断 0 为跳沿触发方式
```

2. 采用中断时的主程序结构

由于各中断入口地址是固定的, 而程序又必须先从主程序起始地址 0000H 执行。所以, 在 0000H 起始地址的几个字节中, 要用无条件转移指令, 跳转到主程序。另外, 各中断入口地址之间依次相差 8 B。中断服务程序稍长就超过 8 B, 这样中断服务程序就占用了其它的中断入口地址, 影响其它中断源的中断。为此, 一般在进入中断后, 利用 1 条无条件转移指令, 跳转到远离其它中断入口的中断服务主程序入口地址。

常用的主程序结构如下:

```
ORG  0000H
LJMP  MAIN
ORG  中断入口地址
LJMP  INT
ORG  XXXXH

MAIN: [主程序]

INT:  [中断服务程序]
```

注意: 在以上的主程序结构中, 如果有多个中断源, 就对应应有多个“ORG 中断入口地址”, 多个“ORG 中断入口地址”必须依次由小到大排列。主程序 MAIN 的起始地址 XXXXH, 根据具体情况来安排。

3. 中断服务程序的流程

MCS-51 响应中断后, 就进入中断服务程序。中断服务程序的基本流程如图 5-9 所示。

下面对有关中断服务程序执行过程中的一些问题进行说明。

(1) 现场保护和现场恢复

所谓现场是指中断时刻单片机中某些寄存器和存储器单元中的数据或状态。为了使中断服务程序的执行不破坏这些数据或状态, 以免在中断返回后影响主程序的运行, 因此要把它们送入堆栈中保存起来, 这就是现场保护。现场保护一定要位于中断处理程序的前面。中断处理结束后, 在返回主程序前, 则需要把保存

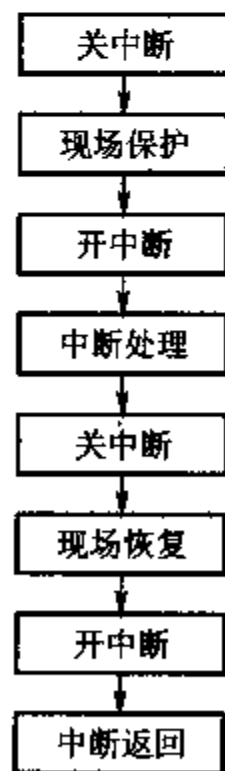


图 5-9 中断服务程序的基本流程

的现场内容从堆栈中弹出,以恢复那些寄存器和存储器单元中的原有内容,这就是现场恢复。现场恢复一定要位于中断处理程序的后面。MCS-51 的堆栈操作指令 PUSH direct 和 POP direct,主要是供现场保护和现场恢复使用的。至于要保护哪些内容,应该由用户根据中断处理程序的具体情况来决定。

(2) 关中断和开中断

图 5-9 中保护现场和恢复现场前关中断,是为了防止此时有高级的中断进入,避免现场被破坏;在保护现场和恢复现场之后的开中断是为下一次的中断作好准备,也为了允许有更高级的中断进入。这样做的结果是,中断处理可以被打断,但原来的现场保护和现场恢复不允许更改,除了现场保护和现场恢复的片刻外,仍然保持着中断嵌套的功能。

但有的时候,对于一个重要的中断,必须执行完毕,不允许被其它的中断所嵌套。对此可在现场保护之前先关闭中断系统,彻底屏蔽其它中断请求,待中断处理完毕后再开中断。这样,就需要将图 5-9 中的“中断处理”步骤前后的“开中断”和“关中断”两个过程去掉。

至于具体中断请求源的关与开,可通过 CLR 或 SETB 指令清 0 或置 1 中断允许寄存器 IE 中的有关位来实现。

(3) 中断处理

中断处理是中断源请求中断的具体目的。应用设计者应根据任务的具体要求,来编写中断处理部分的程序。

(4) 中断返回

中断服务程序的最后 1 条指令必须是返回指令 RETI,RETI 指令是中断服务程序结束的标志。CPU 执行完这条指令后,把响应中断时所置 1 的优先级状态触发器清 0,然后从堆栈中弹出栈顶上的 2B 的断点地址送到程序计数器 PC,弹出的第一个字节送入 PCH,弹出的第二个字节送入 PCL,CPU 从断点处重新执行被中断的主程序。

例 5-4 根据图 5-9 的中断服务程序流程,编写出中断服务程序。假设,现场保护只需要将 PSW 寄存器和累加器 A 的内容压入堆栈中保护起来。

一个典型的中断服务程序如下:

```
INT:  CLR  EA          ;CPU 关中断
      PUSH PSW         ;现场保护
      PUSH A
      SETB EA          ;CPU 开中断
      [中断处理程序段]
      CLR  EA          ;CPU 关中断
      POP  A           ;现场恢复
```

```
POP    PSW
SETB   EA        ;CPU 开中断
RETI                      ;中断返回,恢复断点
```

上述程序有几点需要说明的是:

① 本例的现场保护假设仅仅涉及到 PSW 和 A 的内容,如果还有其它的需要保护的内容,只需要在相应的位置再加几条 PUSH 和 POP 指令即可。注意,对堆栈的操作是先进后出,次序不可颠倒。

② 中断服务程序中的“中断处理程序段”,应用设计者应根据中断任务的具体要求,来编写这部分中断处理程序。

③ 如果本中断服务程序不允许被其它的中断所中断。可将“中断处理程序段”前后的“SETB EA”和“CLR EA”两条指令去掉。

④ 中断服务程序的最后一条指令必须是返回指令 RETI,千万不可缺少。它是中断服务程序结束的标志。CPU 执行完这条指令后,返回断点处,从断点处重新执行被中断的主程序。

5.10 多外部中断源系统设计

MCS-51 为用户提供两个外部中断请求输入端 $\overline{\text{INT0}}$ 和 $\overline{\text{INT1}}$,实际的应用系统中,两个外部中断请求源往往不够用,需对外部中断源进行扩充。本节介绍如何来扩充外部中断源的方法。

5.10.1 定时器/计数器作为外部中断源的使用方法

MCS-51 有两个定时器/计数器(有关定时器/计数器的工作原理将在下一章介绍),当它们选择为计数器工作模式,T0 引脚上发生负跳变时,T0(或 T1)计数器加 1,利用这个特性,可以把 T0(或 T1)引脚作为外部中断请求输入引脚,而定时器/计数器的溢出中断 TF0(或 TF1)作为外部中断请求标志。例如:定时器/计数器 T0 设置为方式 2(自动恢复常数方式)外部计数工作模式,计数器 TH0、TL0 初始值均为 0FFH,并允许 T0 中断,CPU 开放中断,初始化程序如下:

```
ORG    0000H
AJMP   IINI                ;跳到初始化程序
:
INI:   MOV    TMOD,#06H    ;设置 T0 的工作方式寄存器
```

```
MOV  TL0, #0FFH      ;给计数器设置初值
MOV  TH0, #0FFH
SETB TR0              ;启动 T0,开始计数
SETB ET0              ;允许 T0 中断
SETB EA               ;CPU 开中断
```

当连接在 P3.4(T0 引脚)的外部中断请求输入线上的电平发生负跳变时, TL0 加 1,产生溢出,置 1TF0,向 CPU 发出中断请求,同时 TH0 的内容 0FFH 送 TL0,即 TL0 恢复初值 0FFH,这样,P3.4 相当于跳沿触发的外部中断请求源输入端。对 P3.5 也可做类似的处理。

5.10.2 中断和查询结合的方法

若系统中有多多个外部中断请求源,可以按它们的轻重缓急进行排队,把其中最高级别的中断源直接接到 MCS-51 的一个外部中断请求源 IR0 输入端 $\overline{\text{INT0}}$,其余的外部中断请求源 IR1~IR4 用线或的办法连到 MCS-51 的另一个外中断源输入端 $\overline{\text{INT1}}$,同时还连到 P1 口,外部中断源的中断请求由外设的硬件电路产生,这种方法原则上可处理任意多个外部中断。例如,5 个外部中断源的排队顺序依次为:IR0、IR1、…、IR4,对于这样的中断源系统,可以采用如图 5-10 所示的中断电路。

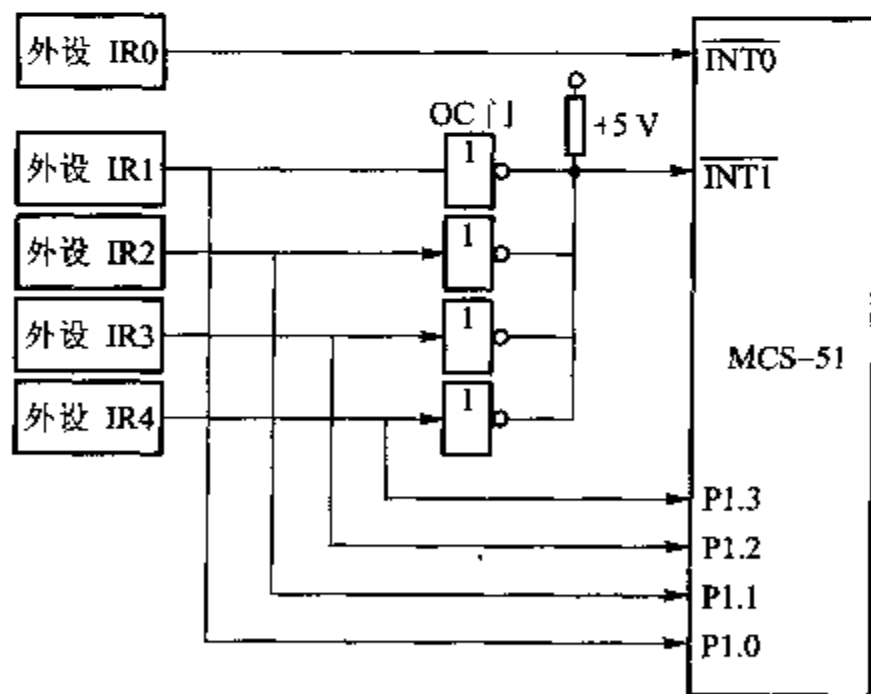


图 5-10 中断和查询相结合的多外部中断请求源系统

图 5-10 中的 4 个外设 IR1~IR4 的中断请求通过集电极开路的 OC 门构成线或的关系,它们的中断请求输入均通过 $\overline{\text{INT1}}$ 传给 CPU。无论哪一个外设提出高电平有效的中断请求信号,都会使 $\overline{\text{INT1}}$ 引脚的电平变低。究竟是哪个外

设提出的中断请求,可通过程序查询 P1.0~P1.3 引脚上的逻辑电平即可知道。设 IR1~IR4 这 4 个中断请求源的高电平可由相应的中断服务程序清 0。

$\overline{\text{INT1}}$ 的中断服务程序如下:

```

ORG 0013H      ; $\overline{\text{INT1}}$  的中断入口
LJMP INT1
:
INT1:  PUSH  PSW      ;保护现场
      PUSH  A
      JB   P1.0,IR1    ;如 P1.0 引脚为高,则 IR1 有中断请求,跳标号 IR1 处理
      JB   P1.1,IR2    ;如 P1.1 引脚为高,则 IR2 有中断请求,跳标号 IR2 处理
      JB   P1.2,IR3    ;如 P1.2 引脚为高,则 IR3 有中断请求,跳标号 IR3 处理
      JB   P1.3,IR4    ;如 P1.4 引脚为高,则 IR4 有中断请求,跳标号 IR4 处理
INTIR:  POP   A        ;恢复现场
      POP  PSW
      RETI             ;中断返回
IR1:    IR1 的中断处理程序
      AJMP INTIR      ;IR1 中断处理完毕,跳标号 INTIR 处执行
IR2:    IR2 的中断处理程序
      AJMP INTIR      ;IR2 中断处理完毕,跳标号 INTIR 处执行
IR3:    IR3 的中断处理程序
      AJMP INTIR      ;IR3 中断处理完毕,跳标号 INTIR 处执行
IR4:    IR4 的中断处理程序
      AJMP INTIR      ;IR4 中断处理完毕,跳标号 INTIR 处执行

```

查询法扩展外部中断源比较简单,但是扩展的外部中断源个数较多时,查询时间稍长。

思考题及习题

1. 什么是中断系统? 中断系统的功能是什么?
2. 什么是中断嵌套?
3. 什么是中断源? MCS-51 有哪些中断源? 各有什么特点?
4. 外部中断 1 所对应的中断入口地址为()H。
5. 下列说法错误的是:
 - (A) 各中断源发出的中断请求信号,都会标记在 MCS-51 系统的 IE 寄存器中。
 - (B) 各中断源发出的中断请求信号,都会标记在 MCS-51 系统的 TMOD 寄存器中。
 - (C) 各中断源发出的中断请求信号,都会标记在 MCS-51 系统的 IP 寄存器中。

(D) 各中断源发出的中断请求信号,都会标记在 MCS-51 系统的 TCON 与 SCON 寄存器中。

6. MCS-51 单片机响应外部中断的典型时间是多少? 在哪些情况下, CPU 将推迟对外部中断请求的响应?

7. 中断查询确认后,在下列各种 8031 单片机运行情况中,能立即进行响应的是:

- (A) 当前正在进行高优先级中断处理
- (B) 当前正在执行 RETI 指令
- (C) 当前指令是 DIV 指令,且正处于取指令的机器周期
- (D) 当前指令是 MOV A, R3

8. 8031 单片机响应中断后,产生长调用指令 LCALL,执行该指令的过程包括:首先把()的内容压入堆栈,以进行断点保护,然后把长调用指令的 16 位地址送(),使程序执行转向()中的中断地址区。

9. 编写出外部中断 1 为跳沿触发的中断初始化程序。

10. 在 MCS-51 中,需要外加电路实现中断撤除的是:

- (A) 定时中断
- (B) 脉冲方式的外部中断
- (C) 外部串行中断
- (D) 电平方式的外部中断

11. MCS-51 有哪几种扩展外部中断源的方法? 各有什么特点?

12. 下列说法正确的是:

- (A) 同一级别的中断请求按时间的先后顺序顺序响应。
- (B) 同一时间同一级别的多中断请求,将形成阻塞,系统无法响应。
- (C) 低优先级中断请求不能中断高优先级中断请求,但是高优先级中断请求能中断低优先级中断请求。
- (D) 同级中断不能嵌套。

13. 中断服务子程序返回指令 RETI 和普通子程序返回指令 RET 有什么区别?

14. 某系统有 3 个外部中断源 1、2、3,当某一中断源变为低电平时,便要求 CPU 进行处理,它们的优先处理次序由高到低依次为 3、2、1,中断处理程序的入口地址分别为 1000H, 1100H, 1200H。试编写主程序及中断服务程序(转至相应的中断处理程序的入口即可)。

第 6 章 MCS-51 的 定时器/计数器

在工业检测、控制中,许多场合都要用到计数或定时功能。例如,对外部脉冲进行计数、产生精确的定时时间等。MCS-51 单片机内有两个可编程的定时器/计数器 T1、T0,以满足这方面的需要。两个定时器/计数器都具有定时器和计数器两种工作模式。

(1) 计数器工作模式

计数功能是对外来脉冲进行计数。MCS-51 芯片有 T0(P3.4)和 T1(P3.5)两个输入引脚,分别是这两个计数器的计数输入端。每当计数器的计数输入引脚的脉冲发生负跳变时,计数器加 1。

(2) 定时器工作模式

定时功能也是通过计数器的计数来实现的,不过此时的计数脉冲来自单片机的内部,即每个机器周期产生 1 个计数脉冲,也就是每经过 1 个机器周期的时间,计数器加 1。如果 MCS-51 采用 12 MHz 晶体,则计数频率为 1 MHz,即每过 1 μ s 的时间计数器加 1。这样可以根据计数值计算出定时时间,也可根据定时时间的要求计算出计数器的初值。

MCS-51 单片机的定时器/计数器具有 4 种工作方式(方式 0、方式 1、方式 2 和方式 3),其控制字均在相应的特殊功能寄存器中,通过对它的特殊功能寄存器的编程,用户可方便地选择定时器/计数器 2 种工作模式和 4 种工作方式。

在了解了 MCS-51 片内的定时器/计数器的上述基本功能后,下面介绍 MCS-51 单片机片内定时器/计数器的结构,功能,有关的特殊功能寄存器,状态字、控制字的含义、工作模式和工作方式的选择以及定时器/计数器的应用举例。

6.1 定时器/计数器的结构

MCS-51 单片机的定时器/计数器结构如图 6-1 所示,定时器/计数器 T0 由特殊功能寄存器 TH0、TL0 构成,定时器/计数器 T1 由特殊功能寄存器 TH1、TL1 构成。

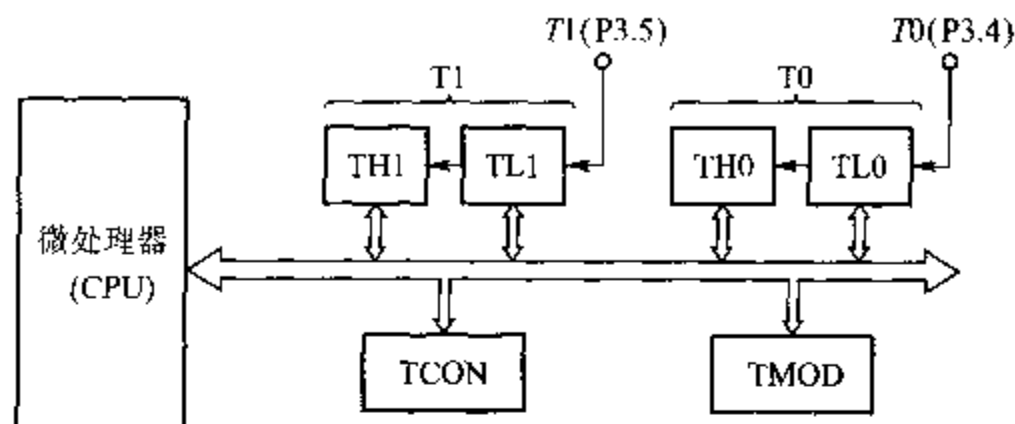
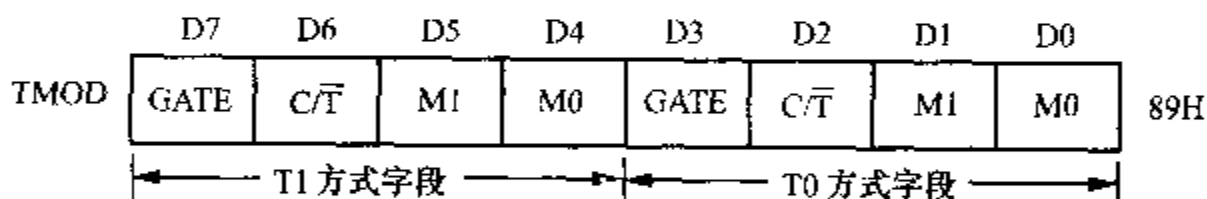


图 6-1 MCS-51 定时器/计数器结构框图

特殊功能寄存器 TMOD 用于选择定时器/计数器 T0、T1 的工作模式和工作方式。特殊功能寄存器 TCON 用于控制 T0、T1 的启动和停止计数,同时包含了 T0、T1 的状态。TMOD、TCON 这两个寄存器的内容由软件设置。单片机复位时,两个寄存器的所有位都被清 0。

6.1.1 工作方式控制寄存器 TMOD

工作方式寄存器 TMOD 用于选择定时器/计数器的工作模式和工作方式,它的字节地址为 89H,不能进行位寻址,其格式如下所示:



8 位分为 2 组,高 4 位控制 T1,低 4 位控制 T0。

下面对 TMOD 的各个位作以说明。

(1) GATE——门控位

$GATE=0$ 时,以运行控制位 $TRX(X=0,1)$ 来启动定时器/计数器运行。

$GATE=1$ 时,用外中断引脚($\overline{INT0}$ 或 $\overline{INT1}$)上的高电平来启动定时器/计数器运行。

(2) M1、M0 ——工作方式选择位

M1、M0 共有 4 种编码,对应于 4 种工作方式的选择,如表 6-1 所示。

表 6-1 工作方式选择

M1	M0	工作方式
0	0	方式 0,为 13 位定时器/计数器。
0	1	方式 1,为 16 位定时器/计数器。
1	0	方式 2,8 位的常数自动重新装载的定时器/计数器。
1	1	方式 3,仅适用于 T0,T0 分成 2 个 8 位计数器,T1 停止计数

(3) C/\bar{T} ——计数器模式和定时器模式选择位

$C/\bar{T}=0$, 为定时器模式。

$C/\bar{T}=1$, 为计数器模式, 计数器对外部输入引脚 $T0(P3.4)$ 或 $T1(P3.5)$ 的外部脉冲(负跳变)计数。

6.1.2 定时器/计数器控制寄存器 TCON

TCON 的字节地址为 88H, 可进行位寻址, 位地址为 88H~8FH。TCON 的格式如下:

	D7	D6	D5	D4	D3	D2	D1	D0	
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	88H

低 4 位与外部中断有关, 已在第 5 章中介绍。高 4 位的功能如下:

(1) TF1、TF0——计数溢出标志位

当计数器计数溢出时, 该位置 1。使用查询方式时, 此位作为状态位供 CPU 查询, 但应注意查询有效后, 应以软件方法及时将该位清 0。使用中断方式时, 此位作为中断请求标志位, 进入中断服务程序后由硬件自动清 0。

(2) TR1、TR0——计数运行控制位

$TR1(TR0)=1$, 启动定时器/计数器工作

$TR1(TR0)=0$, 停止定时器/计数器工作

该位可由软件置 1 或清 0。

6.2 定时器/计数器的 4 种工作方式

6.2.1 方式 0

当 $M1、M0$ 为 00 时, 定时器/计数器被设置为工作方式 0, 这时定时器/计数器的等效框图如图 6-2 所示(以定时器/计数器 $T1$ 为例, $TMOD.5、TMOD.4=00$)。

定时器/计数器工作在方式 0 时, 为 13 位的计数器, 由 $TLX(X=0,1)$ 的低 5 位和 THX 的高 8 位所构成。 TLX 低 5 位溢出则向 THX 进位, THX 计数溢出则置位 TCON 中的溢出标志位 TFX 。

图 6-2 中, C/\bar{T} 位控制的电子开关决定了定时器/计数器的工作模式:

(1) $C/\bar{T}=0$, 电子开关打在上面位置, $T1$ 为定时器工作模式, 以系统时钟振荡器 12 分频后的信号作为计数信号。

(2) $C/\bar{T}=1$, 电子开关打在下面位置, $T1$ 为计数器工作模式, 计数脉冲为

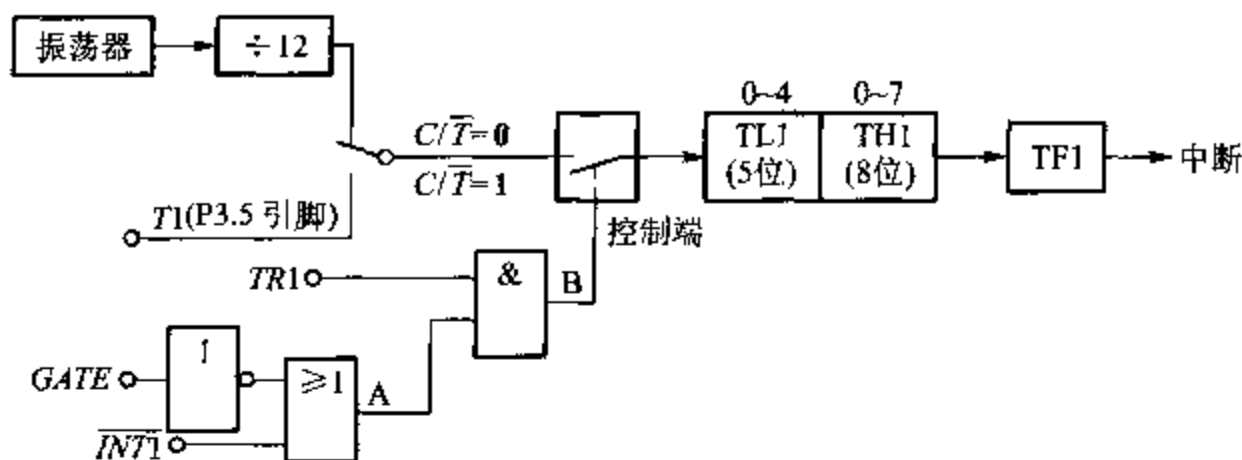


图 6-2 定时器/计数器方式 0 逻辑结构框图

P3.5 引脚上的外部输入脉冲,当引脚上发生负跳变时,计数器加 1。

GATE 位的状态决定定时器/计数器运行控制取决于 TRX 一个条件还是 TRX 和 \overline{INTX} 引脚这两个条件。

(1) $GATE=0$ 时, A 点(见图 6-2)电位恒为 1, B 点的电位仅取决于 TRX 状态。 $TRX=1$, B 点为高电平, 控制端控制电子开关闭合。计数脉冲加到 T1 (或 T0) 引脚, 允许 T1 (或 T0) 计数。 $TRX=0$, B 点为低电平, 电子开关断开, 禁止 T1 (或 T0) 计数。

(2) $GATE=1$ 时, B 点电位由 \overline{INTX} 的输入电平和 TRX 的状态这两个条件来确定。当 $TRX=1$, 且 $\overline{INTX}=1$ 时 ($X=0$ 或 1), B 点才为 1, 控制端控制电子开关闭合, 允许定时器/计数器计数, 故这种情况下计数器是否计数是由 TRX 和 \overline{INTX} 2 个条件来控制的。

6.2.2 方式 1

当 M1、M0 为 01 时, 定时器/计数器工作于方式 1, 这时定时器/计数器的等效电路如图 6-3 所示(以定时器/计数器 T1 为例)。

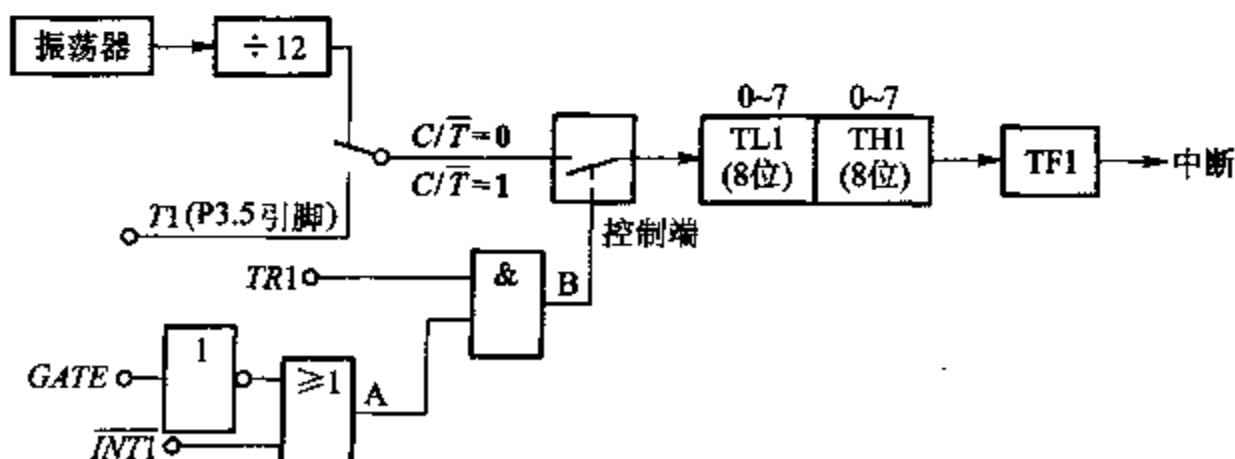


图 6-3 定时器/计数器方式 1 逻辑结构框图

方式1和方式0的差别仅仅在于计数器的位数不同,方式1为16位的计数器,由THX作为高8位和TLX作为低8位构成($X=0,1$),方式0则为13位计数器,有关控制状态位的含义(GATE、C/T、TFX、TRX)与方式0相同。

6.2.3 方式2

方式0和方式1的最大特点是计数溢出后,计数器为全0。因此在循环定时或循环计数应用时就存在反复装入计数初值的问题。这不仅影响定时精度,而且也给程序设计带来麻烦。方式2就是针对此问题而设置的。

当M1、M0为10时,定时器/计数器处于工作方式2,这时定时器/计数器的等效框图如图6-4所示(以定时器T1为例, $X=1$)。

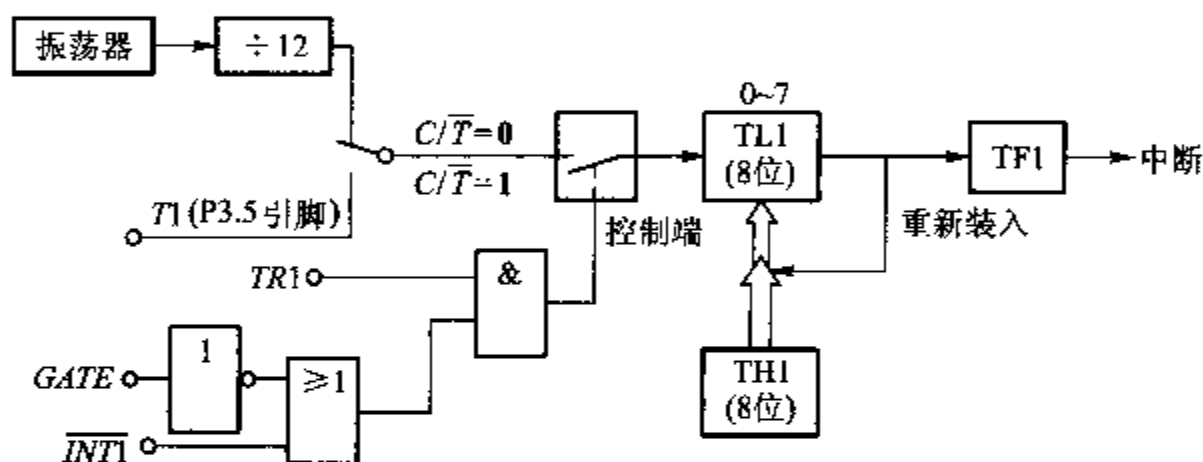


图 6-4 定时器/计数器方式2逻辑结构框图

定时器/计数器的方式2为自动恢复初值的(初值自动装入)8位定时器/计数器,TLX作为常数缓冲器,当TLX计数溢出时,在置1溢出标志TFX的同时,还自动的将THX中的初值送至TLX,使TLX从初值开始重新计数。定时器/计数器的方式2工作过程如图6-5所示($X=0,1$)。

这种工作方式可以省去用户软件中重装初值的程序,简化定时初值的计算方法,可以相当精确的确定定时时间。

6.2.4 方式3

方式3是为了增加1个附加的8位定时器/计数器而提供的,从而使MCS-51具有3个定时器/计数器。方式3只适用于定时器/计数器T0,定时器/计数器T1不能工作在方式3。T1处于方式3时相当于 $TR1=0$,停止计数(此时T1可

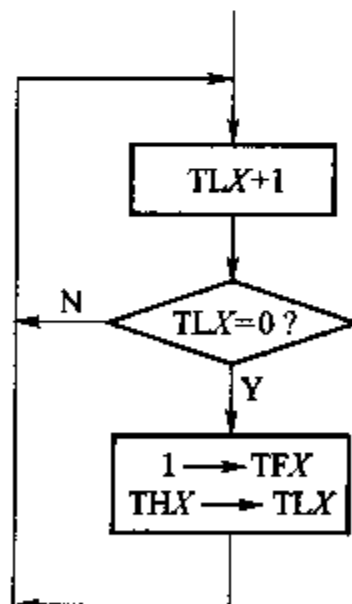
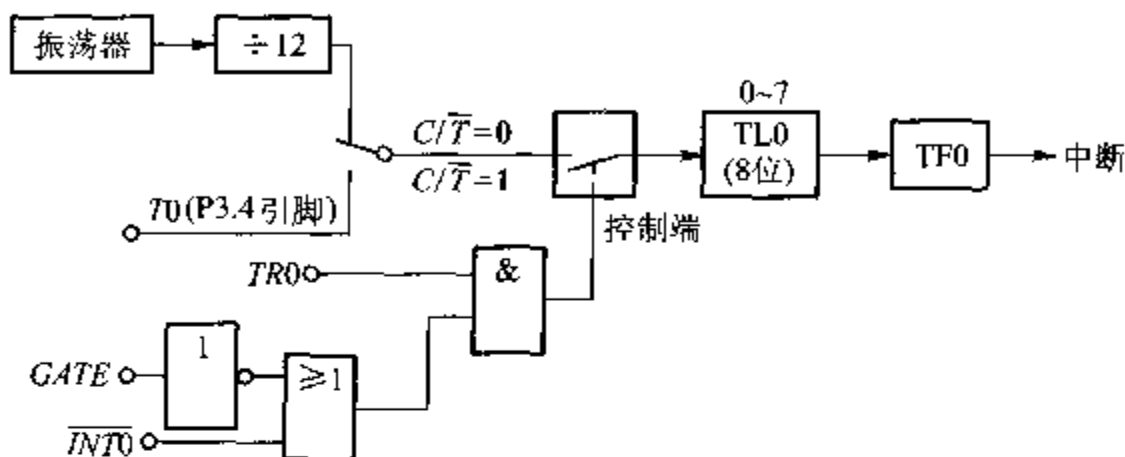


图 6-5 方式2工作过程

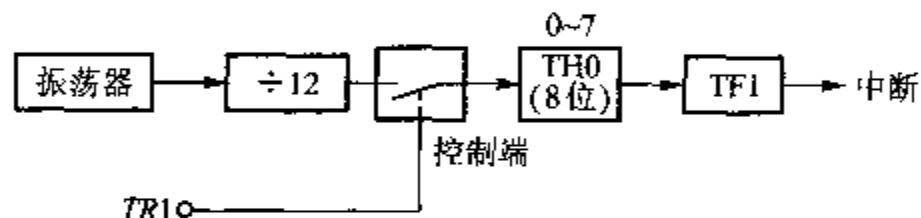
用来作串行口波特率产生器)。

1. 工作方式 3 下的 T0

当 TMOD 的低 2 位为 11 时, T0 的工作方式被选为方式 3, 各引脚与 T0 的逻辑关系框图如图 6-6 所示。



(a) TL0 做 8 位定时器/计数器



(b) TH0 做 8 位定时器

图 6-6 各引脚与 T0 的逻辑关系

定时器/计数器 T0 分为 2 个独立的 8 位计数器: TL0 和 TH0, TL0 使用 T0 的状态控制位 C/\overline{T} 、GATE、TR0、 $\overline{INT0}$, 而 TH0 被固定为 1 个 8 位定时器(不能为外部计数模式), 并使用定时器 T1 的状态控制位 TR1 和 TF1, 同时占用定时器 T1 的中断请求源 TF1。

2. T0 工作在方式 3 下 T1 的各种工作方式

一般情况下, 当 T1 用作串行口的波特率发生器时, T0 才工作在方式 3。T0 处于工作方式 3 时, T1 可定为方式 0、方式 1 和方式 2, 用来作为串行口的波特率发生器, 或不需要中断的场合。

(1) T1 工作在方式 0

T1 的控制字中 $M1、M0 = 00$ 时, T1 工作在方式 0, 工作示意图如图 6-7 所示。

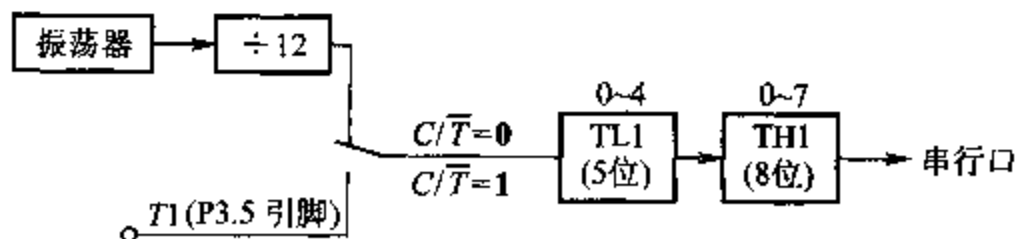


图 6-7 T0 工作在方式 3 时 T1 为方式 0 的工作示意图

(2) T1 工作在方式 1

T1 的控制字中 $M1, M0 = 01$ 时, T1 工作在方式 1, 工作示意图如图 6-8 所示。

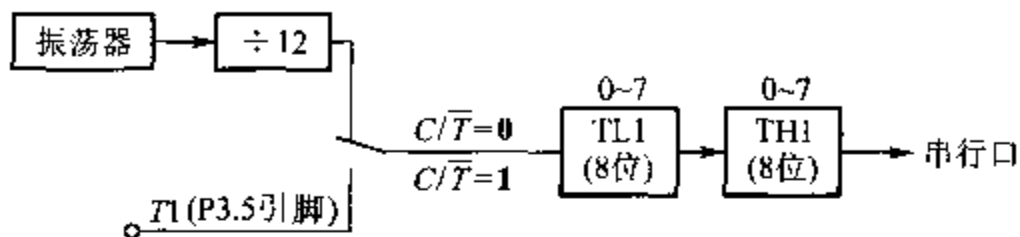


图 6-8 T0 工作在方式 3 时 T1 为方式 1 的工作示意图

(3) T1 工作在方式 2

T1 的控制字中 $M1, M0 = 10$ 时, T1 的工作方式为方式 2, 工作示意图如图 6-9 所示。

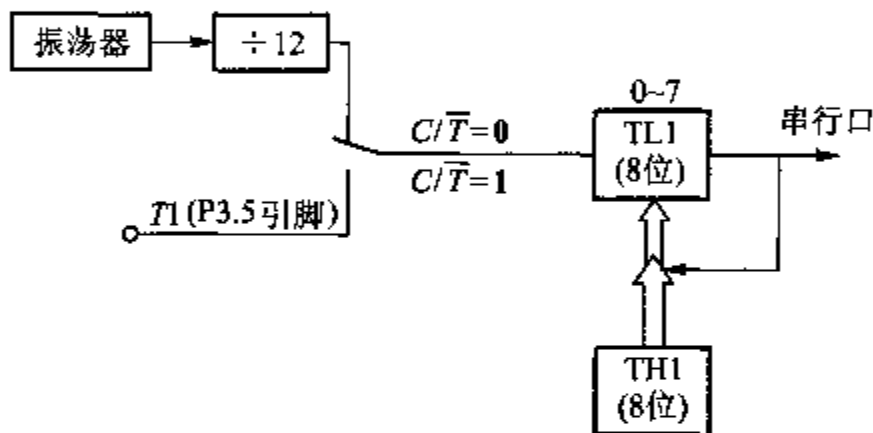


图 6-9 T0 工作在方式 3 时 T1 为方式 2 的工作示意图

(4) T1 工作在方式 3

T1 的控制字中 $M1, M0 = 11$ 时, T1 停止计数。

在 T0 为方式 3 时, T1 运行的控制条件只有 2 个, 即 C/\bar{T} 和 $M1, M0$ 。 C/\bar{T} 选择是定时器模式还是计数器模式, $M1, M0$ 选择 T1 运行的工作方式。

6.3 定时器/计数器对输入信号的要求

当 MCS-51 内部的定时器/计数器被选定为定时器工作模式时, 计数输入信号是内部时钟脉冲, 每个机器周期产生 1 个脉冲使计数器增 1, 因此, 定时器/计数器的输入脉冲的周期与机器周期一样, 为时钟振荡频率的 1/12。当采用 12 MHz 频率的晶体时, 计数速率为 1 MHz, 输入脉冲的周期间隔为 1 μ s。由于定时的精度决定于输入脉冲的周期, 因此当需要高分辨率的定时时, 应尽量选用频率较高的晶体。

当定时器/计数器用作计数器时, 计数脉冲来自相应的外部输入引脚 T0 或

T1。当输入信号产生由 1 至 0 的跳变(即负跳变)时,计数器的值增 1。每个机器周期的 S5P2 期间,对外部输入引脚进行采样。如在第一个机器周期中采得的值为 1,而在下一个周期中采得的值为 0,则在紧跟着的再下一个机器周期 S3P1 的期间,计数器加 1。由于确认 1 次负跳变要花 2 个机器周期,即 24 个振荡周期,因此外部输入的计数脉冲的最高频率为系统振荡器频率的 $1/24$,例如选用 6 MHz 频率的晶体,允许输入的脉冲频率为 250 kHz,如果选用 12 MHz 频率的晶体,则可输入 500 kHz 的外部脉冲。对于外部输入信号的占空比并没有什么限制,但为了确保某一给定的电平在变化之前能被采样 1 次,则这一电平至少要保持 1 个机器周期。故对外部输入信号的基本要求如图 6-10 所示,图中 T_{cy} 为机器周期。

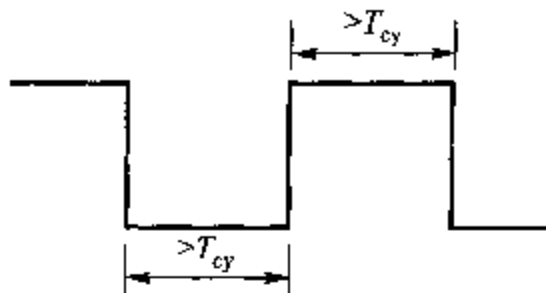


图 6-10 对外部输入信号的基本要求

6.4 定时器/计数器的编程和应用

定时器/计数器的 4 种工作方式中,方式 0 与方式 1 基本相同,只是计数器位数不同。方式 0 为 13 位计数器,方式 1 为 16 位计数器。由于方式 0 是为兼容 MCS-48 而设,且其计数初值计算复杂,所以在实际应用中,一般不用方式 0,而采用方式 1。

6.4.1 方式 1 的应用

例 6-1 假设系统时钟频率采用 6 MHz,要在 P1.0 上输出 1 个周期为 2 ms 的方波,如图 6-11 所示。

方波的周期用定时器 T0 来确定,即在 T0 中设置 1 个初值,在初值的基础上进行计数,每隔 1 ms 计数溢出 1 次,即 T0 每隔 1 ms 产生 1 次中断,CPU 响应中断后,在中断服务程序中对 P1.0 引脚信号取反。T0 中断入口地址为 000BH。为此要做如下几步工作:

(1) 计算初值

$$\text{机器周期} = 2 \mu\text{s} = 2 \times 10^{-6} \text{ s}$$

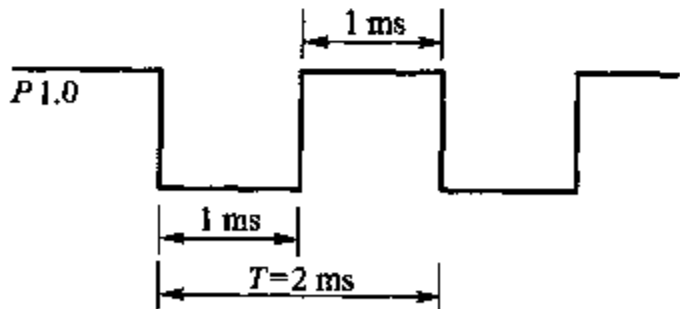


图 6-11 在 P1.0 引脚上输出波形

设:需要装入 T0 的初值为 X,则有: $(2^{16} - X) \times 2 \times 10^{-6} = 1 \times 10^{-3}$

$$2^{16} - X = 500 \quad X = 65\,036$$

X 化为十六进制,即 $X = \text{FE0CH} = \mathbf{1111111000001100B}$ 。

所以,T0 的初值为:

$$\text{TH0} = \text{0FEH} \quad \text{TL0} = \text{0CH}$$

(2) 初始化程序设计

本例采用定时器中断方式工作。初始化程序包括定时器初始化和中断系统初始化,主要是对寄存器 IP、IE、TCON、TMOD 的相应位进行正确的设置,并将计数初值送入定时器中。

(3) 程序设计

中断服务程序除了完成产生要求的方波这一工作之外,还要注意将计数初值重新装入定时器中,为下一次产生中断作准备。主程序可以完成任何其它工作,一般情况下常常是键盘程序和显示程序。在本例中,由于没有这方面的要求,用一条转至自身的短跳转指令来代替主程序。

按上述要求设计的参考程序如下:

```
ORG 0000H
RESET: AJMP MAIN           ;转主程序
ORG 000BH                  ;T0 的中断入口
AJMP IT0P                  ;转 T0 中断处理程序 IT0P
ORG 0100H
MAIN: MOV SP, #60H         ;设堆栈指针
MOV TMOD, #01H            ;设置 T0 为方式 1
ACALL PT0M0               ;调用子程序 PT0M0
HERE: AJMP HERE           ;自身跳转
PT0M0: MOV TL0, #0CH       ;T0 中断服务程序, T0 重新置初值
MOV TH0, #0FEH
SETB TR0                  ;启动 T0
SETB ET0                  ;允许 T0 中断
SETB EA                   ;CPU 开中断
RET
IT0P: MOV TL0, #0CH        ;T0 中断服务子程序, T0 置初值
MOV TH0, #0FEH
CPL P1.0                  ;P1.0 的状态取反
RETI
```

如果 CPU 不做其它工作,也可以采用查询的方式进行控制,程序要简单的多。

查询方式的参考程序如下:

```
MOV    TMOD, #01H    ;设置 T0 为方式 1
SETB   TR0           ;接通 T0
LOOP:  MOV    TH0, #0FEH ;T0 置初值
        MOV    TL0, #0CH
LOOP1: JNB    TF0, LOOP1 ;查询 TF0 标志是否为 1, 如为 1, 说明 T0 溢
                        ;出, 则往下执行
        CLR    TR0      ;T0 溢出, 关闭 T0
        CPL    P1.0      ;P1.0 的状态求反
        SJMP   LOOP
```

由上可见, 程序虽然简单, 但 CPU 必须得不断查询 TF0 标志, 不能再做其它工作。

例 6-2 假设系统时钟为 6 MHz, 编写定时器 T0 产生 1 s 定时的程序。

(1) 定时器 T0 工作方式的确定

因定时时间较长, 采用哪一种工作方式合适呢? 由前面介绍的定时器的各种工作方式的特性, 可以计算出:

{ 方式 0 最长可定时 16.384 ms;
 方式 1 最长可定时 131.072 ms;
 方式 2 最长可定时 512 μ s。

由上可见, 可选方式 1, 每隔 100 ms 中断 1 次, 中断 10 次为 1 s。

(2) 计算计数初值

因为: $(2^{16} - X) \times 2 \times 10^{-6} = 10^{-1}$

所以: $X = 15\,536 = 3CB0H$

因此: $TH0 = 3CH, TL0 = B0H$

(3) 10 次计数的实现

对于中断 10 次计数, 可使 T0 工作在计数方式, 也可用循环程序的方法实现。本例采用循环程序法。

(4) 程序设计

参考程序如下:

```
ORG    0000H
RESET: LJMP  MAIN    ;上电, 转主程序入口 MAIN
        ORG    000BH    ;T0 的中断入口
        LJMP  IT0P      ;转 T0 中断处理程序 IT0P
        ORG    1000H
MAIN:   MOV    SP, #60H ;设堆栈指针
        MOV    B, #0AH  ;设循环次数 10 次
```

```

MOV    TMOD, #01H    ;设 T0 工作在方式 1
MOV    TL0, #0B0H    ;给 T0 设初值
MOV    TH0, #3CH
SETB   TR0            ;启动 T0
SETB   ET0            ;允许 T0 中断
SETB   EA             ;CPU 开放中断
HERE:  SJMP  HERE     ;等待中断
ITOP:  MOV    TL0, #0B0H ;T0 中断服务子程序,重新给 T0 装入初值
      MOV    TH0, #3CH
      DJNZ   B, LOOP
      CLR    TR0       ;1 s 定时时间到,停止 T0 工作
      LOOP: RETI

```

6.4.2 方式 2 的应用

方式 2 是 1 个可以自动重新装载初值的 8 位计数器/定时器。这种工作方式可以省去用户程序中重新装入初值的指令,并可产生相当精确的定时时间。

例 6-3 当 T0(P3.4)引脚上发生负跳变时,从 P1.0 引脚上输出 1 个周期为 1 ms 的方波。如图 6-12 所示。(假设系统时钟为 6 MHz)

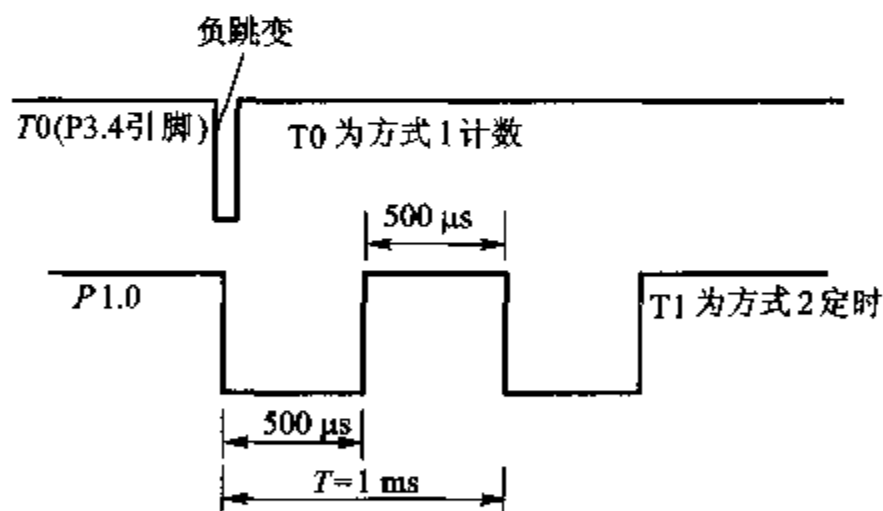


图 6-12

(1) 工作方式选择

T0 定义为方式 1 计数器模式, T0 初值为 0FFFFH, 即外部计数输入端 T0 (P3.4) 发生 1 次负跳变时, 计数器 T0 加 1 且溢出, 溢出标志 TF0 置 1, 向 CPU 发出中断请求。在进入 T0 中断程序后, 把 F0 标志置 1, 说明 T0 引脚上已接收了负跳变信号。T1 定义为方式 2 定时器模式。在 T0 引脚产生 1 次负跳变后, 启动 T1 每 500 μs 产生 1 次中断, 在中断服务程序中对 P1.0 引脚信号求反, 使 P1.0 产生周期为 1 ms 的方波。

(2) 计算 T1 的初值

设 T1 的初值为 X:

则 $(2^8 - X) \times 2 \times 10^{-6} = 5 \times 10^{-4}$

$$X = 2^8 - 250 = 6 = 06H$$

(3) 程序设计

```

ORG    0000H
RESET: LJMP  MAIN           ;复位入口转主程序
ORG    000BH
LJMP    IT0P                ;转 T0 中断服务程序
ORG    001BH
LJMP    IT1P                ;转 T1 中断服务程序
ORG    0100H
MAIN:   MOV    SP, #60H
        ACALL  PT0M2         ;调用对 T0, T1 初始化子程序
LOOP:   MOV    C, F0         ;T0 产生过中断了吗? 产生过中断, 则 F0=1
        JNC    LOOP         ;T0 没有产生过中断, 则跳到 LOOP, 等待 T0
                                ;中断
        SETB   TR1          ;启动 T1
        SETB   ET1          ;允许 T1 中断
HERE:   AJMP   HERE
PT0M2:  MOV    TMOD, #26H    ;对 T1, T0 初始化, T1 为方式 2 定时器, T0
                                ;为方式 1 计数器
        MOV    TL0, #0FFH   ;T0 置初值
        MOV    TH0, #0FFH
        SETB   TR0          ;启动 T0
        SETB   ET0          ;允许 T0 中断
        MOV    TL1, #06H    ;T1 置初值
        MOV    TH1, #06H
        CLR    F0           ;把 T0 已发生中断标志 F0 清 0
        SETB   EA           ;CPU 开放中断
        RET
IT0P:   CLR    TR0          ;T0 中断服务程序, 停止 T0 计数
        SETB   F0           ;建立产生中断标志
        RETI
IT1P:   CPL    P1.0         ;T1 中断服务程序, P1.0 位取反
        RETI

```

在 T1 定时中断服务程序 IT1P 中, 由于方式 2 是初值可以自动重新装载

的,省去了 T1 中断服务程序中重新装入初值 06H 的指令。

例 6-4 利用定时器 T1 的方式 2 对外部信号计数,要求每计满 100 个数,将 P1.0 引脚信号取反。

本例是方式 2 计数模式的应用举例。

(1) 选择工作方式

外部信号由 T1(P3.5)引脚输入,每发生 1 次负跳变计数器加 1,每输入 100 个脉冲,计数器产生溢出中断,在中断服务程序中将 P1.0 引脚信号取反 1 次。

T1 工作在方式 2 的方式控制字为 $TMOD=60H$ 。不使用 T0 时, TMOD 的低 4 位可任取,但不能使 T0 进入方式 3,这里取全 0。

(2) 计算 T1 的初值

$$X=2^8-100=156=9CH$$

因此, TL1 的初值为 9CH,重装初值寄存器 $TH1=9CH$

(3) 程序设计

```
ORG    0000H
LJMP   MAIN
ORG    001BH           ;T1 中断服务程序入口
CPL    P1.0            ;P1.0 位取反
RETI
ORG    0100H
MAIN:  MOV    TMOD, #60H    ;设置 T1 为方式 2 计数
        MOV    TL0, #9CH    ;T0 置初值
        MOV    TH0, #9CH
        SETB   TR1          ;启动 T1
HERE:  AJMP   HERE
```

6.4.3 方式 3 的应用

方式 3 对 T0 和 T1 大不相同。T0 工作在方式 3 时, T1 只能工作在方式 0、1、2。T0 工作在方式 3 时, TL0 和 TH0 被分成 2 个独立的 8 位定时器/计数器。其中, TL0 可作为 8 位的定时器/计数器;而 TH0 只能作为 8 位的定时器。

一般情况下,当定时器 T1 用作串行口波特率发生器时, T0 才设置为方式 3。此时,常把定时器 T1 设置为方式 2,用作波特率发生器。

例 6-5 假设某 MCS-51 应用系统的 2 个外部中断源已被占用,设置定时器 T1 工作在方式 2,作波特率发生器用。现要求增加 1 个外部中断源,并控制 P1.0 引脚输出 1 个 5 kHz 的方波。假设系统时钟为 6 MHz。

(1) 选择工作方式

由第5章介绍的利用定时器作为外部中断源的思想,设置 TL0 工作在方式3 计数模式,把 T0 引脚(P3.4)作附加的外部中断输入端,TL0 的初值设为 0FFH,当检测到 T0 引脚电平出现负跳变时,TL0 溢出,申请中断,这相当于跳沿触发的外部中断源。TH0 为 8 位方式 3 定时模式,定时控制 P1.0 输出 5 kHz 的方波信号。如图 6-13 所示。

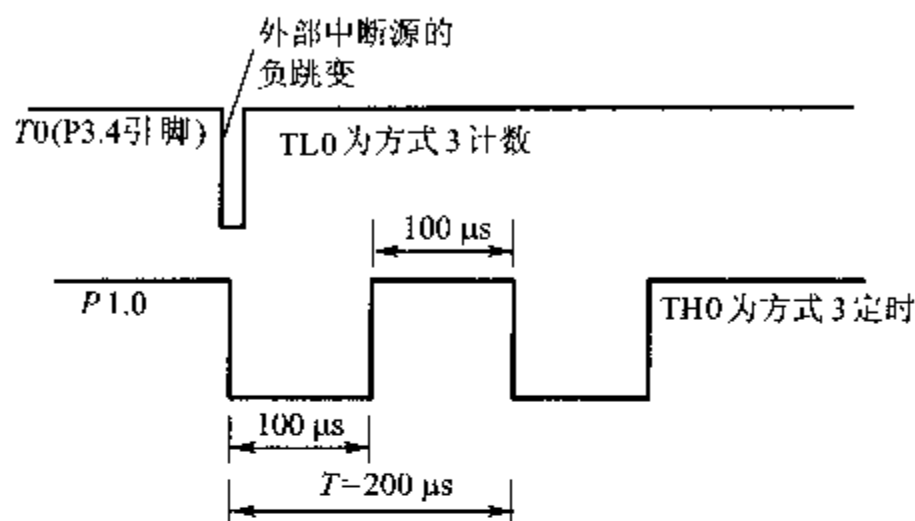


图 6-13

(2) 初值计算

TL0 的初值设为 0FFH。

5 kHz 方波的周期为 $200\ \mu\text{s}$, 因此 TH0 的定时时间为 $100\ \mu\text{s}$ 。TH0 初值 X 计算如下:

$$(2^8 - X) \times 2 \times 10^{-6} = 1 \times 10^{-4}$$

$$X = 2^8 - 100 = 156 = 9\text{CH}$$

(3) 程序设计

源程序如下:

```

ORG    0000H
LJMP   MAIN
ORG    000BH           ;T0 中断入口
LJMP   TL0INT          ;跳 T0 中断服务程序
ORG    001BH           ;注意,在 T1 为方式 3 时,TH0 占用了 T1
                        ;的中断
LJMP   TH0INT          ;跳 TH0 中断服务程序
ORG    0100H
MAIN:  MOV    TMOD, #27H ;T0 为方式 3 计数,T1 为方式 2 定时
      MOV    TL0, #0FFH ;置 TL0 初值
      MOV    TH0, #9CH  ;置 TH0 初值
      MOV    TL1, #data1 ;data 是根据波特率常数要求来定,见第
                        ;7 章

```

```

MOV    TH1, #datah
MOV    TCON, #55H    ;允许 T0 中断
MOV    IE, #9FH      ;启动 T1
      :
TLOINT: MOV    TLO, #0FFH    ;TLO 中断服务程序, TLO 重新装入初值
      [中断处理]
      RETI
TH0INT: MOV    TH0, #9CH    ;TH0 中断服务程序, TH0 重新装入初值
      CPL    P1.0          ;P1.0 位取反输出
      RETI

```

6.4.4 门控制位 GATE 的应用——测量脉冲宽度

下面以 T1 为例,来介绍门控制位 GATE1 的应用。门控制位 GATE1 可使定时器/计数器 T1 的启动计数受 $\overline{INT1}$ 的控制,当 $GATE1=1$, $TR1$ 为 1 时,只有 $\overline{INT1}$ 引脚输入高电平时, T1 才被允许计数,利用 GATE1 的这个功能,(对于 GATE0 也是一样,可使 T0 的启动计数受 $\overline{INT0}$ 的控制),可测量 $\overline{INT1}$ 引脚 (P3.3) 上正脉冲的宽度(机器周期数),其方法如图 6-14 所示。

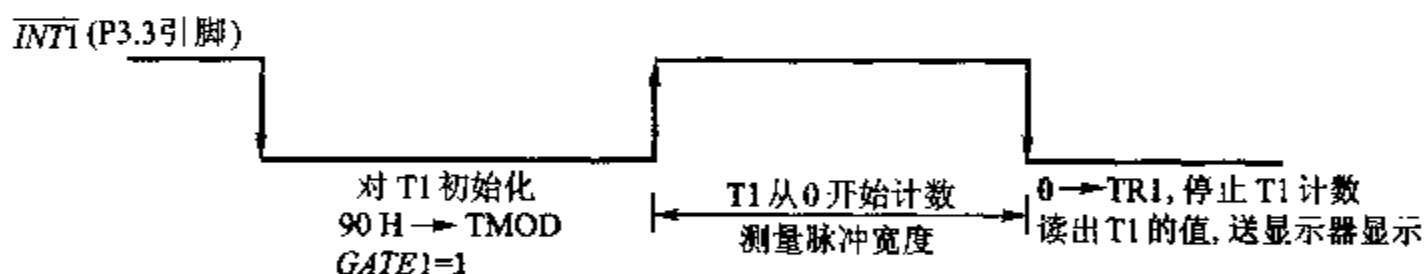


图 6-14 利用 GATE 位测量正脉冲的宽

参考程序如下:

```

ORG    0000H
RESET: AJMP  MAIN          ;复位入口转主程序
ORG    0100H
MAIN:   MOV    SP, #60H
        MOV    TMOD, #90H    ;设控制字, T1 为方式 1 定时
        MOV    TL1, #00H
        MOV    TH1, #00H
LOOP:   JB     P3.3, LOOP    ;等待  $\overline{INT1}$  低
        SETB   TR1          ;如果  $\overline{INT1}$  为低, 启动 T1
LOOP1:  JNB    P3.3, LOOP1    ;等待  $\overline{INT1}$  升高
LOOP2:  JB     P3.3, LOOP2    ;等待  $\overline{INT1}$  降低

```

```
CLR    TR1          ;停止 T1 计数
MOV    A,TL1        ;T1 计数值送 A
```

```
将 A 中的 T1 计数值
送显示缓冲区并转
换成可显示的代码
```

```
LOOP3: LCALL DIR          ;调用显示子程序 DIR(略)显示 T1 计数值
        AJMP  LOOP3
```

执行以上程序,使 $\overline{\text{INT1}}$ 引脚上出现的正脉冲宽度以机器周期数的形式显示在显示器上。

6.4.5 实时时钟的设计

本节介绍如何使用定时器/计数器来实现实时时钟,实时时钟就是以秒、分、时为单位进行计时。

1. 实时时钟实现的基本思想

时钟的最小计时单位是秒,如何获得 1 s 的定时时间,从前面的例 6-2 的介绍可知,使用定时器的方式 1,最大的定时时间也只能达到 131 ms。我们可把定时器的定时时间定为 100 ms,采用中断方式进行溢出次数的累计,计满 10 次,即得到秒计时。而计数 10 次可用循环程序的方法实现。初值的计算请见例 6-2。

时钟运行时,在片内 RAM 中规定 3 个单元作为秒、分、时单元,具体安排如下:

42H:“秒”单元;41H:“分”单元;40H:“时”单元

从秒到分,从分到时是通过软件累加并进行比较的方法来实现的。要求每满 1 s,则“秒”单元 42H 中的内容加 1;“秒”单元满 60,则“分”单元 41H 中的内容加 1;“分”单元满 60,则“时”单元 40H 中的内容加 1;“时”单元满 24,则将 42H、41H、40H 的内容全部清 0。

2. 程序设计

(1) 主程序的设计

主程序的主要功能是进行定时器 T0 的初始化,并启动 T0,然后通过反复调用显示子程序,等待 100 ms 定时中断的到来。主程序的流程如图 6-15 所示。

(2) 中断服务程序的设计

中断服务程序(IT0P)的主要功能是实现秒、分、时的计时处理。实现计时操作的基本思想,已在上

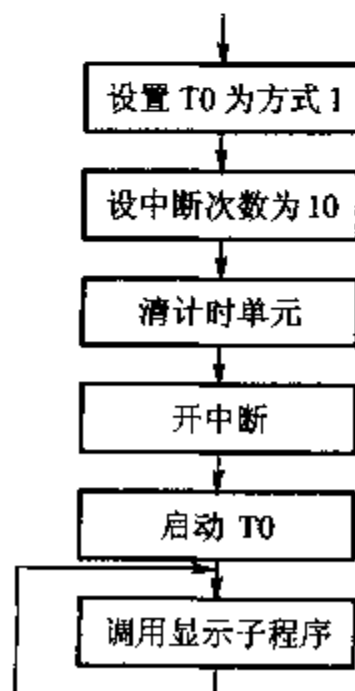


图 6-15 时钟主程序流程

面介绍过。中断服务程序的流程如图 6-16 所示。

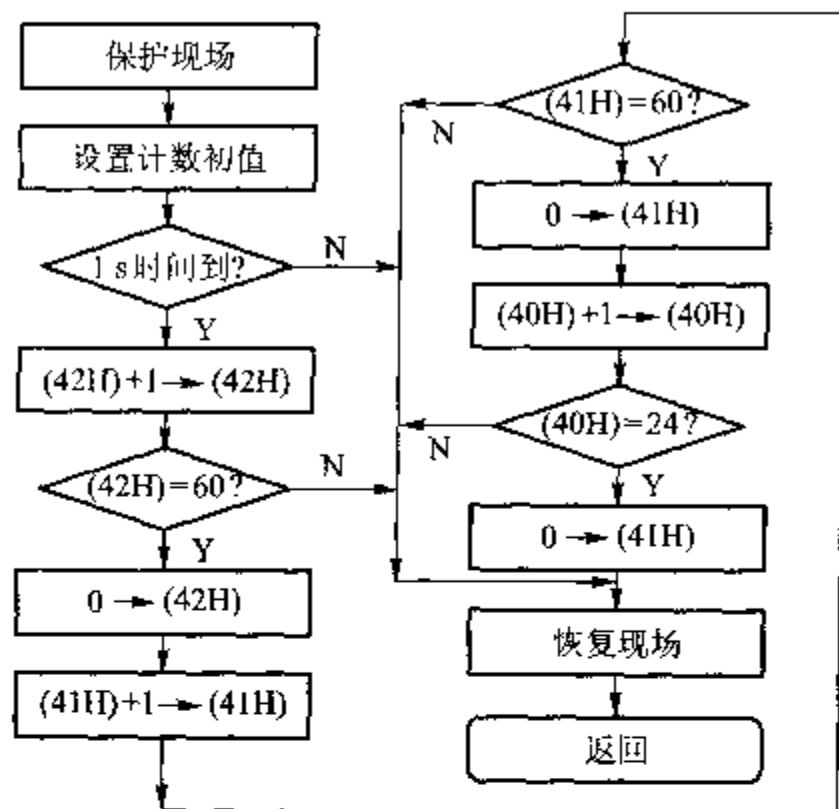


图 6-16 中断服务程序流程

参考程序如下：

```

ORG 1000H
AJMP MAIN                ; 上电, 跳向主程序
ORG 000BH                ; T0 的中断入口
AJMP IT0P

MAIN: MOV TMOD, #01H      ; 设 T0 为方式 1
      MOV 20H, #0AH       ; 装入中断次数
      CLR A
      MOV 40H, A          ; “时”单元清 0
      MOV 41H, A          ; “分”单元清 0
      MOV 42H, A          ; “秒”单元清 0
      SETB ET0            ; 允许 T0 申请中断
      SETB EA             ; CPU 开中断
      MOV TH0, #3CH       ; 给 T0 装入计数初值
      MOV TL0, #0B0H
      SETB TR0            ; 启动 T0
      HERE: SJMP HERE     ; 等待中断(也可调用显示子程序)

IT0P: PUSH PSW            ; 保护现场
      PUSH A
      MOV TH0, #3CH       ; 重新装入初值
      MOV TL0, #0B0H
  
```

```

DJNZ 20H, RETURN      ;1s 未到, 返回
MOV 20H, #0AH          ;重置中断次数
MOV A, #01H            ;“秒”单元增 1
ADD A, 42H
DA A                    ;“秒”单元十进制调整
MOV 42H, A              ;“秒”的 BCD 码存回“秒”单元
CJNE A, #60, RETURN    ;是否到 60 s, 未到则返回
MOV 42H, #00H          ;计满 60 s, “秒”单元清 0
MOV A, #01H            ;“分”单元增 1
ADD A, 41H
DA A                    ;“分”单元十进制调整
MOV 41H, A              ;“分”的 BCD 码存回“分”单元
CJNE A, #60, RETURN    ;是否到 60 分, 未到则返回
MOV 41H, #00H          ;计满 60 分, “分”单元清 0
MOV A, #01H            ;“时”单元增 1
ADD A, 40H
DA A                    ;“时”单元十进制调整
MOV 40H, A              ;
CJNE A, #24, RETURN    ;是否到 24 小时, 未到则返回
MOV 40H, #00H          ;“时”单元清 0
RETURN; POP A           ;恢复现场
POP PSW
RETI                    ;中断返回
END

```

6.4.6 运行中读定时器/计数器

在读取运行中的定时器/计数器时, 需要特别加以注意, 否则读取的计数值有可能出错。原因是 CPU 不可能在同一时刻同时读取 THX 和 TLX 的内容。比如, 先读(TLX), 后读(THX), 由于定时器在不断运行, 读(THX)前, 若恰好出现 TLX 溢出向 THX 进位的情况, 则读得的(TLX)值就完全不对了。同样, 先读(THX)再读(TLX)也可能出错。

解决读错问题的方法是: 先读(THX), 后读(TLX), 再读(THX)。若 2 次读得(THX)相同, 则可确定读得的内容是正确的。若前后 2 次读得的(THX)有变化, 则再重复上述过程, 这次重复读得的内容就应该是正确的。下面是有关的程序, 读得的(TH0)和(TL0)放置在 R1 和 R0 内。

```

RDTIME: MOV A, TH0      ;读(TH0)

```

```
MOV    R0, TL0          ;读(TL0)
CJNE   A, TH0, RDTIME   ;比较2次读得的(TH0), 不相等则重复读
MOV    R1, A             ;(TH0)送入 R1 中
RET
```

思考题及习题

1. 如果采用的晶振的频率为 3 MHz, 定时器/计数器工作在方式 0、1、2 下, 其最大的定时时间各为多少?
2. 定时器/计数器用作定时器时, 其计数脉冲由谁提供? 定时时间与哪些因素有关?
3. 定时器/计数器作计数器使用时, 对外界计数频率有何限制?
4. 采用定时器/计数器 T0 对外部脉冲进行计数, 每计数 100 个脉冲后, T0 转为定时工作方式。定时 1 ms 后, 又转为计数方式, 如此循环不止。假定 MCS-51 单片机的晶体振荡器的频率为 6 MHz, 请使用方式 1 实现, 要求编写出程序
5. 定时器/计数器的工作方式 2 有什么特点? 适用于哪些应用场合?
6. 编写程序, 要求使用 T0, 采用方式 2 定时, 在 P1.0 输出周期为 400 μ s, 占空比为 10:1 的矩形脉冲。
7. 一个定时器的定时时间有限, 如何实现两个定时器的串行定时, 来实现较长时间的定时?
8. 当定时器 T0 用于方式 3 时, 应该如何控制定时器 T1 的启动和关闭?
9. 定时器/计数器测量某正单脉冲的宽度, 采用何种方式可得到最大量程? 若时钟频率为 6 MHz, 求允许测量的最大脉冲宽度是多少?
10. 编写一段程序, 功能要求为: 当 P1.0 引脚的电平正跳变时, 对 P1.1 的输入脉冲进行计数; 当 P1.2 引脚的电平负跳变时, 停止计数, 并将计数值写入 R0、R1 (高位存 R1, 低位存 R0)。
11. THX 与 TLX (X=0,1) 是普通寄存器还是计数器? 其内容可以随时用指令更改吗? 更改后的新值是立即刷新还是等当前计数器计满之后才能刷新?
12. 判断下列说法是否正确?
 - (1) 特殊功能寄存器 SCON, 与定时器/计数器的控制无关。
 - (2) 特殊功能寄存器 TCON, 与定时器/计数器的控制无关。
 - (3) 特殊功能寄存器 IE, 与定时器/计数器的控制无关。
 - (4) 特殊功能寄存器 TMOD, 与定时器/计数器的控制无关。

第7章 MCS-51 的串行口

MCS-51 单片机内部有一个功能强大的全双工的异步通信串行口。所谓全双工就是双机之间串行接收、发送数据可同时进行。所谓异步通信,就是收、发双方没有同步时钟来控制收、发双方的同步传送,而是靠双方各自的时钟来控制数据的异步传送。要传送的串行数据在发方是以数据帧形式一帧一帧地发送,通过传输线由收方一帧一帧地接收。

MCS-51 的串行口有 4 种工作方式,波特率可由软件设置片内的定时器/计数器来控制。每当串行口接收或发送 1 B 完毕,均可发出中断请求。MCS-51 的串行口除了可以用于串行数据通信之外,还可以非常方便地用来扩展并行 I/O 口。

7.1 串行口的结构

MCS-51 串行口的内部结构如图 7-1 所示。它有两个物理上独立的接收、发送缓冲器 SBUF,可同时发送、接收数据,发送缓冲器只能写入不能读出,接收缓冲器只能读出不能写入,两个缓冲器共用一个特殊功能寄存器字节地址(99H)。

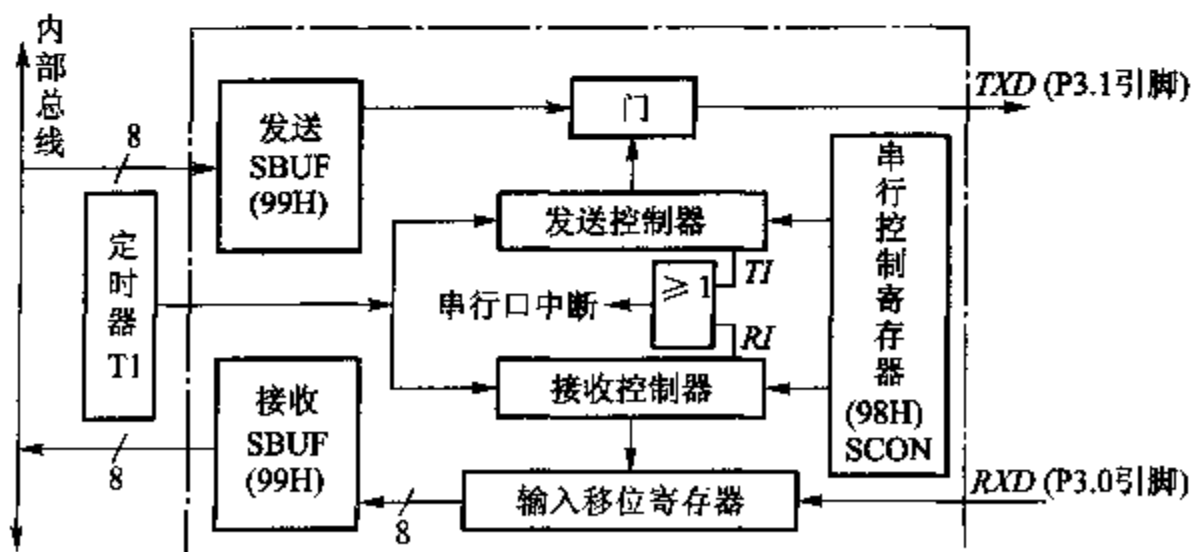


图 7-1 串行口的内部结构

控制 MCS-51 单片机串行口的控制寄存器共有两个：特殊功能寄存器 SCON 和 PCON。下面对这两个特殊功能寄存器各个位的功能予以详细介绍。

7.1.1 串行口控制寄存器 SCON

串行口控制寄存器 SCON, 字节地址 98H, 可位寻址, 位地址为 98H~9FH。SCON 的格式如图 7-2 所示。

	D7	D6	D5	D4	D3	D2	D1	D0	
SCON		SM0	SM1	SM2	REN	TB8	RB8	TI	RI
									98 H
位地址	9FH	9EH	9DH	9CH	9BH	9AH	99H	98H	

图 7-2 串行口控制寄存器 SCON 的格式

下面介绍 SCON 中各个位的功能。

(1) SM0、SM1——串行口 4 种工作方式的选择位

SM0、SM1 两位的编码所对应的工作方式如表 7-1 所示。

表 7-1 串行口的 4 种工作方式

SM0	SM1	方式	功能说明
0	0	0	同步移位寄存器方式(用于扩展 I/O 口)
0	1	1	8 位异步收发, 波特率可变(由定时器控制)
1	0	2	9 位异步收发, 波特率为 $f_{osc}/64$ 或 $f_{osc}/32$
1	1	3	9 位异步收发, 波特率可变(由定时器控制)

(2) SM2——多机通信控制位

因为多机通信是在方式 2 和方式 3 下进行的, 因此, SM2 位主要用于方式 2 或方式 3 中。当串行口以方式 2 或方式 3 接收时, 如果 SM2=1, 则只有当接收到的第 9 位数据(RB8)为 1 时, 才将接收到的前 8 位数据送入 SBUF, 并置 1 RI, 产生中断请求; 当接收到的第 9 位数据(RB8)为 0 时, 则将接收到的前 8 位数据丢弃。而当 SM2=0 时, 则不论第 9 位数据是 1 还是 0, 都将前 8 位数据送入 SBUF 中, 并置 1 RI, 产生中断请求。

在方式 1 时, 如果 SM2=1, 则只有收到有效的停止位时才会激活 RI。

在方式 0 时, SM2 必须为 0。

(3) REN——允许串行接收位

由软件置 1 或清 0。

REN=1 允许串行口接收数据。

$REN=0$ 禁止串行口接收数据。

(4) TB8 — 发送的第 9 位数据

在方式 2 和 3 时, TB8 是要发送的第 9 位数据。其值由软件置 1 或清 0。在双机通信时, TB8 一般作为奇偶校验位使用; 在多机通信中用来表示主机发送的是地址帧还是数据帧, $TB8=1$ 为地址帧, $TB8=0$ 为数据帧。

(5) RB8 — 接收到的第 9 位数据

工作在方式 2 和 3 时, RB8 存放接收到的第 9 位数据。在方式 1, 如果 $SM2=0$, RB8 是接收到的停止位。在方式 0, 不使用 RB8。

(6) TI — 发送中断标志位

串行口工作在方式 0 时, 串行发送第 8 位数据结束时由硬件置 1, 在其它工作方式, 串行口发送停止位的开始时置 1。 $TI=1$, 表示一帧数据发送结束, 可供软件查询, 也可申请中断。CPU 响应中断后, 在中断服务程序中向 SBUF 写入要发送的下一帧数据。TI 必须由软件清 0。

(7) RI — 接收中断标志位

串行口工作在方式 0 时, 接收完第 8 位数据时, RI 由硬件置 1。在其它工作方式中, 串行口接收到停止位时, 该位置 1。 $RI=1$, 表示一帧数据接收完毕, 并申请中断, 要求 CPU 从接收 SBUF 取走数据。该位的状态也可供软件查询。RI 必须由软件清 0。

SCON 的所有位都可进行位操作清 0 或置 1。

7.1.2 特殊功能寄存器 PCON

特殊功能寄存器 PCON 字节地址为 87H, 没有位寻址功能。PCON 的格式如图 7-3 所示:

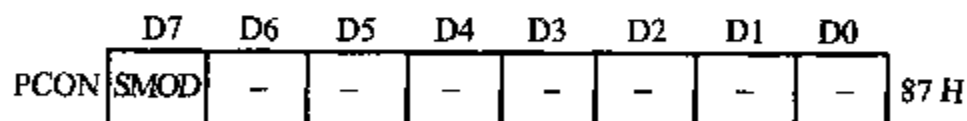


图 7-3 寄存器 PCON 的格式

SMOD: 波特率选择位。

例如: 方式 1 的波特率的计算公式为:

$$\text{方式 1 波特率} = \frac{2^{SMOD}}{32} \times \text{定时器 T1 的溢出率}$$

当 $SMOD=1$ 时, 要比 $SMOD=0$ 时的波特率加倍, 所以也称 SMOD 位为波特率倍增位。

7.2 串行口的4种工作方式

串行口的4种工作方式由特殊功能寄存器 SCON 中 SM0、SM1 位定义,编码见表 7-1。

7.2.1 方式0

串行口的工作方式 0 为同步移位寄存器输入输出方式,常用于外接移位寄存器,以扩展并行 I/O 口。这种方式不适用于两个 MCS-51 之间的串行通信。

方式 0 以 8 位数据为 1 帧,不设起始位和停止位,先发送或接收最低位。波特率是固定的,为 $f_{osc}/12$ 。方式 0 的帧格式如图 7-4 所示:

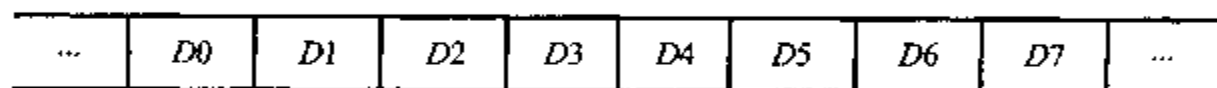


图 7-4 方式 0 的帧格式

1. 方式 0 发送

发送过程中,当 CPU 执行 1 条将数据写入发送缓冲器 SBUF 的指令时,产生 1 个正脉冲,串行口开始即把 SBUF 中的 8 位数据以 $f_{osc}/12$ 的固定波特率从 RXD 引脚串行输出,低位在先,TXD 引脚输出同步移位脉冲,发送完 8 位数据置 1 中断标志位 TI。时序如图 7-5 所示。

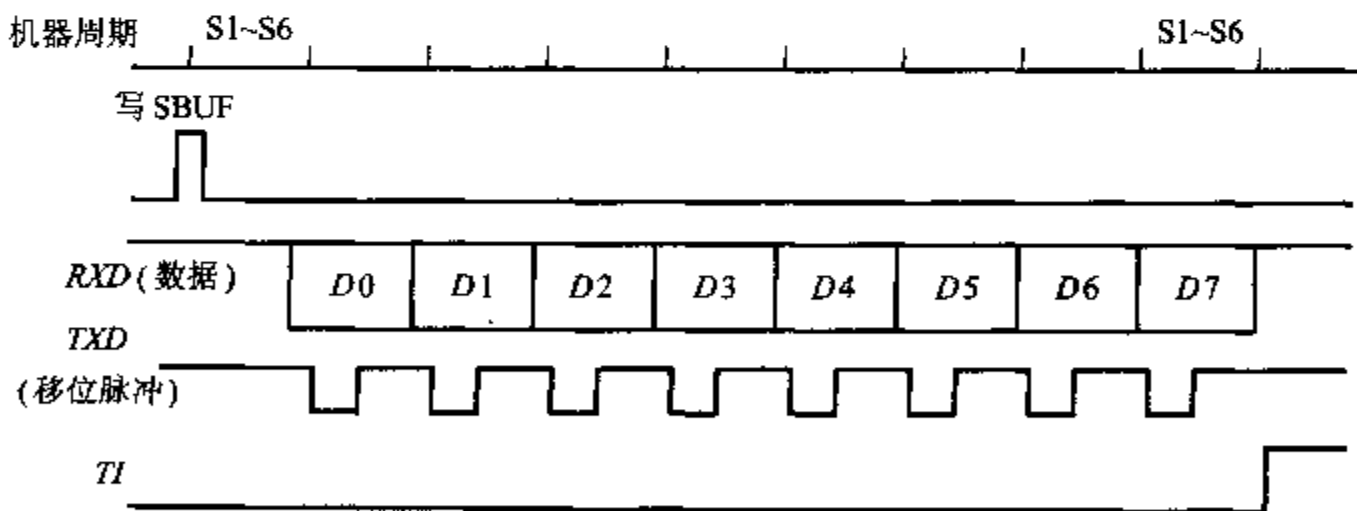


图 7-5 方式 0 发送时序

2. 方式 0 接收

方式 0 接收时,REN 为串行口允许接收控制位,REN=0,禁止接收;REN=1,允许接收。当 CPU 向串行口的 SCON 寄存器写入控制字(置为方式 0,并置 1 REN 位,同时 RI=0)时,产生 1 个正脉冲,串行口即开始接收数据。引脚 RXD 为

数据输入端, TXD 为移位脉冲信号输出端, 接收器也以 $f_{osc}/12$ 的固定波特率采样 RXD 引脚的数据信息, 当接收器接收到 8 位数据时置 1 中断标志 RI, 表示 1 帧数据接收完毕, 可进行下一帧数据的接收。时序如图 7-6 所示。

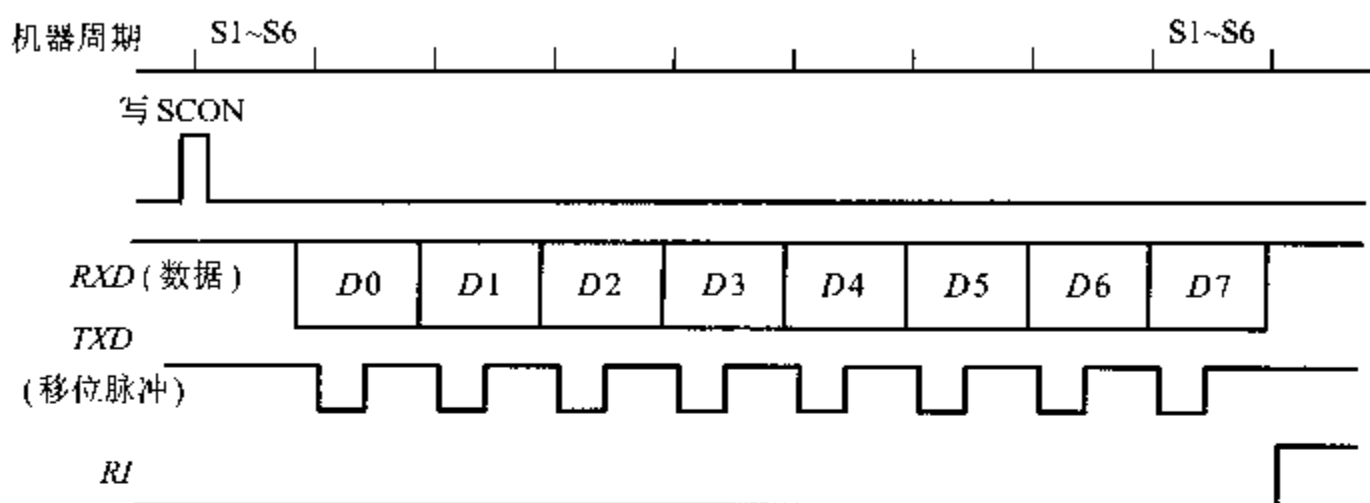


图 7-6 方式 0 接收时序

上面介绍了方式 0 的发送和接收。在方式 0 下, SCON 中的 TB8、RB8 位没有用到, 发送或接收完 8 位数据由硬件置 1 TI 或 RI 中断标志位, CPU 响应 TI 或 RI 中断。TI 或 RI 标志位必须由用户软件清 0, 可采用如下指令:

CLR TI ; TI 位清 0

CLR RI ; RI 位清 0

清 0 TI 或 RI。方式 0 时, SM2 位(多机通信控制位)必须为 0。

7.2.2 方式 1

SM0、SM1 两位为 01 时, 串行口以方式 1 工作。方式 1 真正用于数据的串行发送和接收。TXD 引脚和 RXD 引脚分别用于发送和接收数据。方式 1 收发 1 帧的数据为 10 位, 1 个起始位(0), 8 个数据位, 1 个停止位(1), 先发送或接收最低位。方式 1 的帧格式如图 7-7 所示。

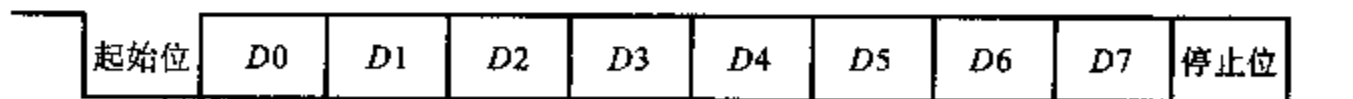


图 7-7 方式 1 的帧格式

方式 1 时, 串行口为波特率可变的 8 位异步通信接口。方式 1 的波特率由下式确定:

$$\text{方式 1 波特率} = \frac{2^{\text{SMOD}}}{32} \times \text{定时器 T1 的溢出率}$$

式中 SMOD 为 PCON 寄存器的最高位的值(0 或 1)。

1. 方式 1 发送

串行口以方式1输出时,数据位由TXD端输出,发送1帧信息为10位,1位起始位0,8位数据位(先低位)和1位停止位1,当CPU执行1条数据写发送缓冲器SBUF的指令,就启动发送。图7-8中TX时钟的频率就是发送的波特率。发送开始时,内部发送控制信号 \overline{SEND} 变为有效,将起始位向TXD输出,此后,每经过1个TX时钟周期,便产生1个移位脉冲,并由TXD输出1个数据位。8位数据位全部发送完毕后,置1中断标志位TI,然后 \overline{SEND} 信号失效。方式1发送数据的时序,如图7-8所示。

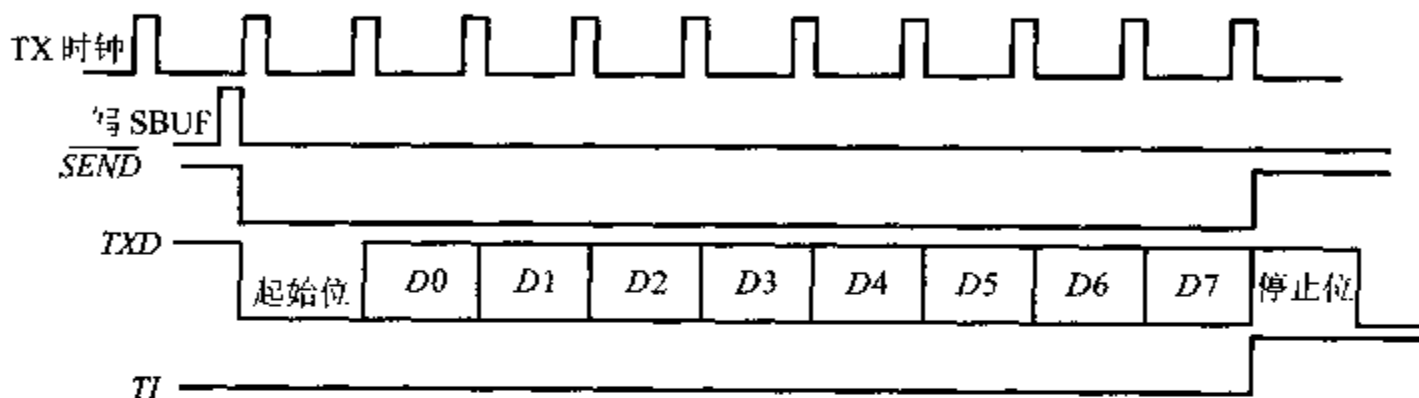


图7-8 方式1发送数据时序

2. 方式1接收

串行口以方式1接收时($REN=1, SM0, SM1=01$),数据从RXD(P3.0)引脚输入。当检测到起始位的负跳变时,则开始接收。接收时,定时控制信号有2种(如图7-9所示),一种是接收移位时钟(RX时钟),它的频率和传送的波特率相同。另一种是位检测器采样脉冲,它的频率是RX时钟的16倍。也就是在1位数据期间,有16个采样脉冲,以波特率的16倍的速率采样RXD引脚状态,当采样到RXD端从1到0的跳变时就启动检测器,接收的值是3次连续采样(第7、8、9个脉冲时采样)取其中2次相同的值,以确认是否是真正的起始位(负跳变)的开始,这样能较好地消除干扰引起的影响,以保证可靠无误的开始接收数据。当确认起始位有效时,开始接收1帧信息。接收每一位数据时,也都进行3次连续采样(第7、8、9个脉冲时采样),接收的值是3次采样中至少2次相同的值,以保证接收到的数据位的准确性。

当一帧数据接收完毕以后,必须同时满足以下两个条件,这次接收才真正有效。

(1) $RI=0$,即上一帧数据接收完成时, $RI=1$ 发出的中断请求已被响应,SBUF中的数据已被取走,说明“接收SBUF”已空。

(2) $SM2=0$ 或收到的停止位=1(方式1时,停止位已进入RB8),则将接收到的数据装入SBUF和RB8(RB8装入停止位),且置1中断标志RI。

若这两个条件不同时满足,收到的数据不能装入SBUF,这意味着该帧数据将丢失。

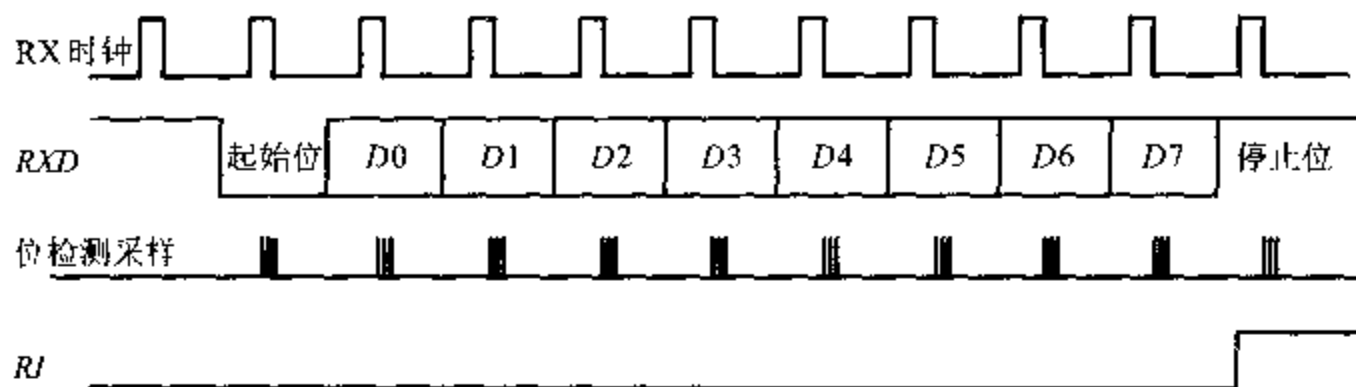


图 7-9 方式 1 接收数据时的时序

7.2.3 方式 2

串行口工作于方式 2 和方式 3 时,被定义为 9 位异步通信接口。每帧数据均为 11 位,1 位起始位 0,8 位数据位(先低位),1 位可编程为 1 或 0 的第 9 位数据和 1 位停止位 1。方式 2 的帧格式见图 7-10。

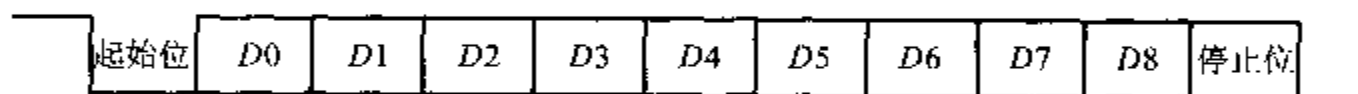


图 7-10 方式 2、方式 3 的帧格式

方式 2 的波特率由下式确定:

$$\text{方式 2 波特率} = \frac{2^{\text{SMOD}}}{64} \times f_{\text{osc}}$$

1. 方式 2 发送

发送前,先根据通信协议由软件设置 TB8(例如,双机通信时的奇偶校验位或多机通信时的地址/数据的标志位)。然后将要发送的数据写入 SBUF,即可启动发送过程。串行口能自动把 TB8 取出,并装入到第 9 位数据位的位置,再逐一发送出去。发送完毕,则把 TI 位置 1。

串行口方式 2 发送数据的时序波形如图 7-11 所示。

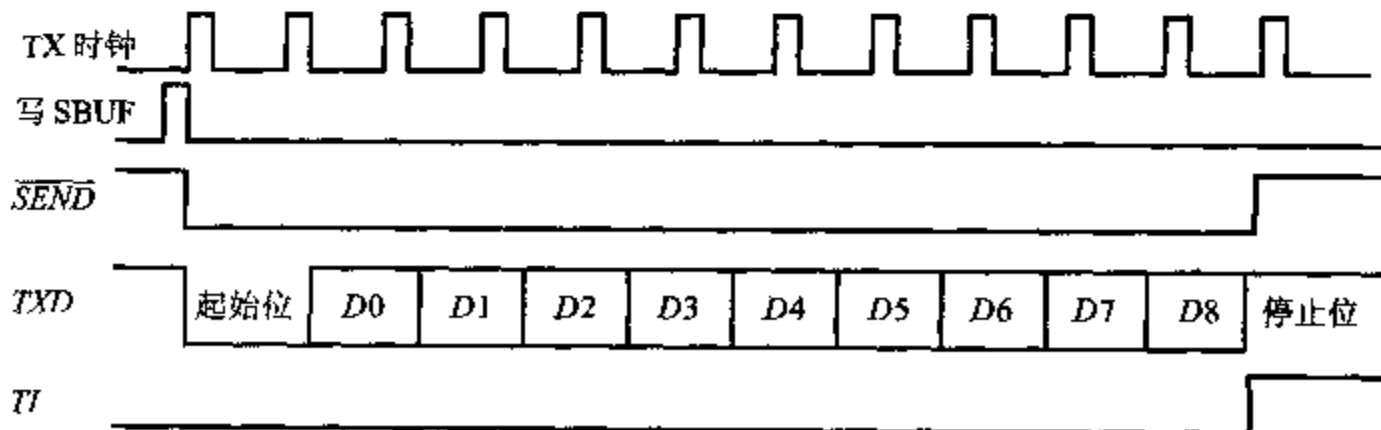


图 7-11 方式 2 和方式 3 发送数据的时序波形

例 7-1 方式 2 发送在双机通信中的应用。

下面的发送中断服务程序,是在双机通信中,以 TB8 作为奇偶校验位,处理方法为数据写入 SBUF 之前,先将数据的奇偶校验位写入 TB8(设第 2 组的工作寄存器区的 R0 作为发送数据区地址指针)。本程序采用偶校验发送。

```

PIPTI:  PUSH PSW           ;现场保护
        PUSH A
        SETB RS1           ;选择第 2 组工作寄存器区
        CLR RS0
        CLR TI             ;发送中断标志清 0
        MOV A,@R0          ;取数据
        MOV C,P            ;校验位送 TB8,采用偶校验
        MOV TB8,C
        MOV SBUF,A         ;数据写入发送缓冲器,启动发送
        INC R0              ;数据指针加 1
        POP A              ;恢复现场
        POP PSW
        RETI                ;中断返回
    
```

2. 方式 2 接收

当串行口的 SCON 寄存器的 SM0、SM1 两位为 10,且 REN=1 时,允许串行口以方式 2 接收数据。接收时,数据由 RXD 端输入,接收 11 位信息。当位检测逻辑采样到 RXD 引脚从 1 到 0 的负跳变,并判断起始位有效后,便开始接收 1 帧信息。在接收完第 9 位数据后,需满足以下 2 个条件,才能将接收到的数据送入 SBUF(接收缓冲器)

(1) RI=0,意味着接收缓冲器为空。

(2) SM2=0 或接收到的第 9 位数据位 RB8=1 时。

当上述 2 个条件满足时,接收到的数据送入 SBUF(接收缓冲器),第 9 位数据送入 RB8,并置 1RI。若不满足这 2 个条件,接收的信息将被丢弃。

串行口方式 2 接收数据的时序波形如图 7-12 所示。

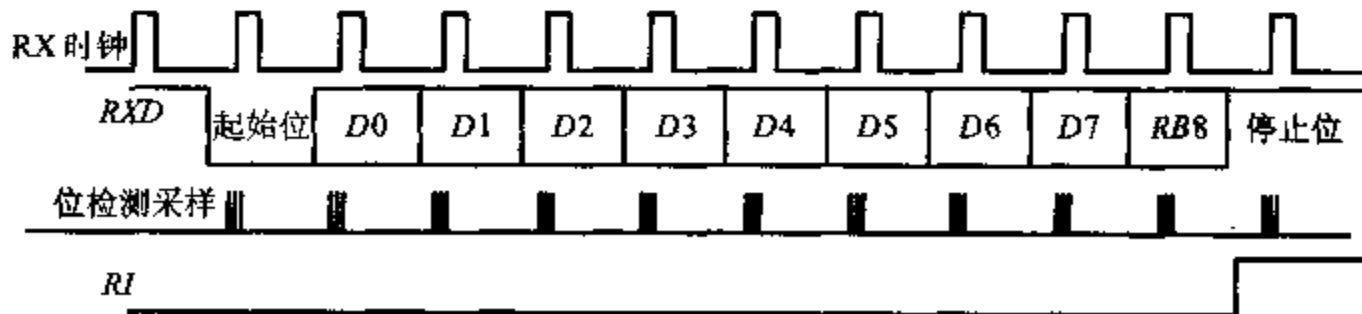


图 7-12 方式 2 和方式 3 接收数据的时序波形

例 7-2 方式 2 接收在双机通信中的应用。

本例与例 7-1 是相对应的。若附加的第 9 位数据为校验位,在接收程序中应作偶校验处理,可采用如下程序(设 1 组寄存器区的 R0 为数据缓冲器指针)。

```

PIR1:  PUSH  PSW
        PUSH  A
        SETB  RS0           ;选择 1 组寄存器区
        CLR   RS1
        CLR   RI
        MOV   A,SBUF        ;将接收到数据送到累加器 A
        MOV   C,P
        JNC   L1
        JNB   RB8,ERP       ;ERP 为出错处理程序标号
        AJMP  L2
L1:     JB     RB8,ERP
L2:     MOV   @R0,A
        INC   R0
        POP   A
        POP   PSW
ERP:    ...                 ;出错处理程序段入口
        :
        RETI

```

7.2.4 方式 3

当 SM0、SM1 两位为 11 时,串行口被定义工作在方式 3。方式 3 为波特率可变的 9 位异步通信方式,除了波特率外,方式 3 和方式 2 相同。方式 3 发送和接收数据的时序波形见图 7-11 和图 7-12。

方式 3 的波特率由下式确定:

$$\text{方式 3 波特率} = \frac{2^{SMOD}}{32} \times \text{定时器 T1 的溢出率}$$

7.3 多机通信

多个 MCS-51 单片机可利用串行口进行多机通信。在多机通信中,要保证主机与所选择的从机实现可靠地通信,必须保证串行口具有识别功能。串行口

控制寄存器 SCON 中的 SM2 位就是为满足这一条而设置的多机通信控制位。其多机控制原理是在串行口以方式 2(或方式 3)接收时,若 $SM2=1$,表示置多机通信功能位,这时出现两种可能情况:

(1) 接收到的第 9 位数据为 1 时,数据才装入 SBUF,并置中断标志 $RI=1$ 向 CPU 发出中断请求;

(2) 接收到的第 9 位数据为 0 时,则不产生中断标志,信息将抛弃。

若 $SM2=0$,则接收的第 9 位数据不论是 0 还是 1,都产生 $RI=1$ 中断标志,接收到的数据装入 SBUF 中。

应用 MCS-51 串行口的这个功能,便可实现 MCS-51 的多机通信。

设在 1 个多机系统中有 1 个主机(MCS-51 或其它具有串行接口的微计算机)和 3 个由 8031 组成的从机系统,如图 7-13 所示。主机的 RXD 与所有从机的 TXD 端相连,TXD 与所有从机的 RXD 端相连。从机的地址分别为 00H、01H 和 02H。

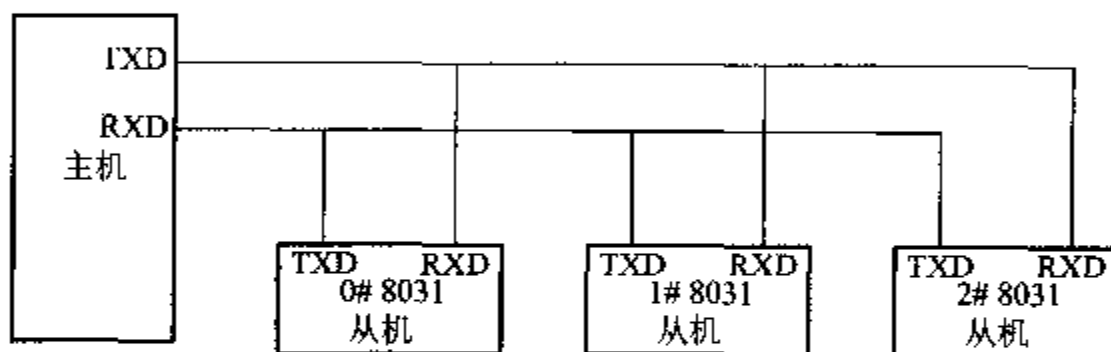


图 7-13 多机通信系统示意图

多机通信的工作过程如下:

(1) 从机初始化程序允许串行口中断,将串行口编程为方式 2 或方式 3 接收,即 9 位异步通讯方式,且置 1SM2 和 REN 位,使从机只处于多机通信且接收地址帧的状态。

(2) 在主机和某一个从机通信之前,先将从机地址(即准备接收数据的从机)发送给各个从机系统。接着才传送数据或命令,主机发出的地址信息的第 9 位为 1,数据(包括命令)信息的第 9 位为 0。当主机向各从机发送地址时,各从机的串行口接收到的第 9 位信息 RB8 为 1,且由于 $SM2=1$,则置 1 中断标志位 RI,各从机 8031 响应中断,执行中断服务程序。在中断服务子程序中,判断主机送来的地址是否和本机地址相符合,若为本机的地址,则该从机清 0SM2 位,准备接收主机的数据或命令;若地址不相符,则保持 $SM2=1$ 状态。

(3) 接着主机发送数据帧,此时各从机串行口接收到的 $RB8=0$,只有与前面地址相符合的从机系统(即已清 0SM2 位的从机)才能激活中断标志位 RI,从而进入中断服务程序,在中断服务程序中接收主机的数据(或命令);其它的从机因

SM2 保持为 1, 又 RB8=0 不激活中断标志 RI, 所以不能进入中断, 把所接收的数据丢失不作处理, 从而保证了主机和从机间通信的正确性。

图 7-13 所示的多机系统是主从式, 由主机控制多机之间的通信, 从机和从机之间的通信只能经主机才能实现。

7.4 波特率的制定方法

在串行通信中, 收发双方发送或接收的波特率必须一致。通过软件对 MCS-51 串行口可设定 4 种工作方式。其中方式 0 和方式 2 的波特率是固定的; 方式 1 和方式 3 的波特率是可变的, 由定时器 T1 的溢出率来确定(定时器 T1 的溢出率就是 T1 每秒溢出的次数)。

7.4.1 波特率的定义

波特率的定义: 串行口每秒钟发送(或接收)的位数称为波特率。设发送 1 位所需要的时间为 T, 则波特率为 $1/T$ 。

MCS-51 的串行口以方式 1 或方式 3 工作, 波特率和定时器 T1 的溢出率有关。

对于定时器的不同工作方式, 得到的波特率的范围是不一样的, 这是由定时器/计数器 T1 在不同工作模式下, 计数位数的不同所决定的。

7.4.2 定时器 T1 产生波特率的计算

波特率和串行口的工作方式有关。

(1) 串行口工作在方式 0 时, 波特率固定为时钟频率 f_{osc} 的 $1/12$, 且不受 SMOD 位的值的影响。若 $f_{osc}=12\text{ MHz}$, 波特率为 $f_{osc}/12$ 即 1 Mb/s 。

(2) 串行口工作在方式 2 时, 波特率与 SMOD 位的值有关。

$$\text{方式 2 波特率} = \frac{2^{SMOD}}{64} \times f_{osc}$$

若 $f_{osc}=12\text{ MHz}$; $SMOD=0$ 波特率 $=187.5\text{ kb/s}$; $SMOD=1$ 波特率 $=375\text{ kb/s}$ 。

(3) 串行口工作在方式 1 或方式 3 时, 常用定时器 T1 作为波特率发生器, 其关系式为:

$$\text{波特率} = \frac{2^{SMOD}}{32} \times \text{定时器 T1 的溢出率} \quad (7-1)$$

由式(7-1)可见, T1 的溢出率和 SMOD 的值共同决定波特率。

在实际设定波特率时, T1 常设置为方式 2 定时(自动装初值), 即 TL1 作 8 位计数器, TH1 存放备用初值。这种方式不仅可使操作方便, 也可避免因软件重装初值而带来的定时误差。

设定时器 T1 方式 2 的初值为 X, 则有:

$$\text{定时器 T1 的溢出率} = \frac{\text{计数速率}}{256 - X} = \frac{f_{\text{osc}}/12}{256 - X} \quad (7-2)$$

将式(7-2)代入式(7-1), 则有:

$$\text{波特率} = \frac{2^{\text{SMOD}}}{32} \times \frac{f_{\text{osc}}}{12(256 - X)} \quad (7-3)$$

由式(7-3)可见, 这种方式波特率随 f_{osc} 、SMOD 以及初值 X 而变化。

在实际使用时, 经常根据已知波特率和时钟频率来计算定时器 T1 的初值 X。为避免烦杂的初值计算, 常用的波特率和初值 X 间的关系常可列成表 7-2, 以供查用。

表 7-2 用定时器 T1 产生的常用波特率

波特率	f_{osc}	SMOD 位	定时器 T1		
			C/ \bar{T}	工作方式	初值 X
串行口方式 0: 1 Mb/s	12 MHz	×	×	×	×
串行口方式 0: 0.5 Mb/s	6 MHz	×	×	×	×
串行口方式 2: 375 kb/s	12 MHz	1	×	×	×
串行口方式 2: 187.5 kb/s	6 MHz	1	×	×	×
串行口方式 1 或 3: 62.5 kb/s	12 MHz	1	0	2	FFH
19.2 kb/s	11.059 2 MHz	1	0	2	FDH
9.6 kb/s	11.059 2 MHz	0	0	2	FDH
4.8 kb/s	11.059 2 MHz	0	0	2	FAH
2.4 kb/s	11.059 2 MHz	0	0	2	F4H
1.2 kb/s	11.059 2 MHz	0	0	2	E8H
137.5 b/s	11.059 2 MHz	0	0	2	1DH
110 b/s	12 MHz	0	0	1	FEEBH
19.2 kb/s	6 MHz	1	0	2	FEH
9.6 kb/s	6 MHz	1	0	2	FDH
4.8 kb/s	6 MHz	0	0	2	FDH
2.4 kb/s	6 MHz	0	0	2	FAH
1.2 kb/s	6 MHz	0	0	2	F4H
0.6 kb/s	6 MHz	0	0	2	E8H
110 b/s	6 MHz	0	0	2	72H
55 b/s	6 MHz	0	0	1	FEEBH

表 7-2 有两点需要注意:

(1) 在使用的时钟振荡频率为 12 MHz 或 6 MHz 时,表中初值 X 和相应的波特率之间有一定误差。例如,FDH 的对应的理论值是 10 416 b/s(时钟振荡频率为 6 MHz 时),与 9 600 b/s 相差 816 b/s,消除误差可以通过调整时钟振荡频率 f_{osc} 实现。例如采用时钟振荡频率为 11.059 2 MHz。

(2) 如果串行通信选用很低的波特率,例如,波特率选为 55 b/s,可将定时器 T1 设置为方式 1 定时。但在这种情况下,T1 溢出时,需在中断服务程序中重新装入初值。中断响应时间和执行指令时间会使波特率产生一定的误差,可用改变初值的方法加以调整。

例 7-3 若 8031 单片机的时钟振荡频率为 11.059 2 MHz,选用 T1 为方式 2 定时作为波特率发生器,波特率为 2 400 b/s,求初值。

设 T1 为方式 2 定时,选 $SMOD=0$ 。

将已知条件带入式(7-3)中:

$$\text{波特率} = \frac{2^{SMOD}}{32} \times \frac{f_{osc}}{12 \times (256 - X)} = 2\,400 \text{ b/s}$$

从中解得: $X=244=F4H$

只要把 F4H 装入 TH1 和 TL1,则 T1 发出的波特率为 2 400 b/s。

上述结果也可直接从表 7-2 中查到。

这里时钟振荡频率选为 11.059 2 MHz,就可使初值为整数,从而产生精确的波特率。

7.5 串行口的编程和应用

串行口的 4 种工作方式中的方式 0 是移位寄存器工作方式,主要用于扩展 I/O 用,并不用于串行通信,有关方式 0 的应用将在后续的有关章节中介绍,下面介绍串行口在串行通信中的应用及软件编程。

7.5.1 串行口方式 1 应用编程(双机通信)

例 7-4 串行口发送/接收共 10 位信息(方式 1),第 0 位是起始位 0,1~8 位是数据位,最后是停止位 1。

甲机把以 78H、77H 单元的内容为首地址,以 76H、75H 单元中内容减 1 为末地址的外部数据存储器中的数据块内容通过串行口向乙机发送,定时器 T1 作为波特率发生器。

设 8031 内部 RAM 中:

(78H) = 20H ;首地址为 2000H

(77H) = 00H ;

(76H) = 20H ;末地址为 2020H

(75H) = 20H ;

下面介绍甲乙两台 8031 单片机之间采用方式 1 的串行通信程序。

1. 甲机发送程序

下面的程序把外部数据存储器 2000 H~201 FH 中的数据块内容通过串行口 TXD 引脚输出。甲机的晶振频率为 6 MHz, 波特率设置为 2 400 b/s。甲机发送数据的参考程序如下:

```
                ORG    0000H
RESET:  LJMP    MAIN
                ORG    001BH
                LJMP    TIINT
                ORG    0023H
                LJMP    SINT
                ORG    0100H
MAIN:      MOV    SP, #60H
            MOV    78H, #20H
            MOV    77H, #00H
            MOV    76H, #20H
            MOV    75H, #20H
            ACALL  TRANS
HERE:      AJMP   HERE
            RET
TRANS:     ANL    TMOD, #0FH      ;设置定时器 T1 为方式 1
            ORL    TMOD, #10H
            MOV    TL1, #0FAH     ;给定时器 T1 置初值
            MOV    TH1, #0FFH
            MOV    PCON, #80H     ;SMOD 位=1
            SETB   EA             ;CPU 开中断
            CLR    ES             ;关串行口中断
            SETB   ET1            ;允许 T1 中断
            SETB   PT1            ;定时器 T1 为高优先级中断
            CLR    PS             ;串行口为低优先级中断
            SETB   TR1            ;定时器 T1 开始运行
            CLR    TI             ;发送中断标志位 TI 清 0
```

```

MOV     SCON, #40H      ;设置串行口为方式 1
MOV     SBUF, 78H        ;输出数据块首地址
WAIT1:  JNB     TI, WAIT1
        CLR     TI
        MOV     SBUF, 77H
WAIT2:  JNB     TI, WAIT2
        CLR     TI
        MOV     SBUF, 76H      ;输出数据块末地址
WAIT3:  JNB     TI, WAIT3
        CLR     TI
        MOV     SBUF, 75H
        SETB    ES          ;允许串行口中断
WAIT4:  JNB     FO, WAIT4
        CLR     TI
        RET
        ...                ;其他处理程序段
TIINT:  CLR     TR1         ;T1 中断服务程序入口,关定时器 T1
        MOV     TL1, #0FAH   ;重新置定时器 T1 初值
        MOV     TH1, #0FFH
        SETB    TR1         ;重开定时器 T1
        RETI              ;T1 中断返回
SINT:   PUSH    DPL         ;串行口中断服务程序入口,首先保护现场
        PUSH    DPH
        PUSH    A
ESCOM:  MOV     DPH, 78H
        MOV     DPL, 77H
        MOVX    A, @DPTR     ;从外部 RAM 读入数据块中的数据
        CLR     TI
        MOV     SBUF, A      ;从串行口输出数据块中的数据
        MOV     A, DPH
        CJNE    A, 76H, END1
        MOV     A, DPL
        CJNE    A, 75H, END1 ;数据块输出结束否? 未结束,跳 END1 处
        CLR     ES          ;数据块输出结束,关串行口中断
        CLR     ET1         ;关定时器中断
        CLR     TR1         ;关定时器
        POP     A
        POP     DPH

```

```

        POP    DPL
        RETI          ;串行口中断返回
END1:   INC     77H   ;数据块地址加 1
        MOV    A,77H
        JNZ    END2
        INC     78H
END2:   SJMP    ESCOM
    
```

2. 乙机接收程序

乙机通过 RXD 引脚接收甲机发送的串行字节,这时必须使得接收方的波特率和发送方波特率相同。设置定时器 T1 工作在方式 1,产生的波特率为 2 400 b/s,晶振频率为 6 MHz。

接收程序功能:接收的第一、二字节是数据块首地址,第三、四字节是数据块末地址减 1,第五字节开始是数据,把数据依次放入数据块首地址开始的外部数据存储器中。

乙机接收的参考程序如下:

```

        ORG    0000H
RESET:  LJMP    MAIN
        ORG    001BH
        LJMP    T1INT
        ORG    0023H
        LJMP    RSINT
        ORG    0100H
MAIN:   MOV     SP, #60H
        ACALL  RECEIVE
        AJMP    $
        RET
RECEIVE:ANL    TMOD, #0FH   ;置定时器 T1 为方式 1
        ORL    TMOD, #10H
        MOV    TL1, #0FAH   ;定时器 T1 置初值
        MOV    TH1, #0FFH
        MOV    PCON, #80H   ;SMOD 位=1
        SETB   EA           ;CPU 开中断
        SETB   ET1          ;允许定时器 T1 申请中断
        SETB   ES           ;允许串行口申请中断
        SETB   PT1          ;定时器 T1 为高优先级中断
        CLR    PS           ;串行口为低优先级中断
        SETB   TR1          ;定时器 T1 开始运行
    
```

```

MOV     SCON, #50H      ;置串行口方式 1 接收
CLR     B.0              ;地址/数据标志 B.0 清 0,为地址
MOV     70H, #78H
CLR     F0                ;设置标志位, F0=0 时接收数据
WAIT:   JNB     F0, WAIT  ;标志位 F0=1 时,接收数据结束
RET
...
; 其他程序段
TIINT:  CLR     TR1       ;定时器 T1 中断程序入口,关定时器 T1
MOV     TL1, #0FAH       ;重新装入初值
MOV     TH1, #0FFH
SETB    TR1              ;开定时器 T1
RETI

RSINT:  PUSH    DPL       ;串行口接收中断服务程序,首先保护现场
PUSH    DPH
PUSH    A
MOV     A, R0
PUSH    A
JB      B.0, DATA1      ;判接收的是地址(B.0=0)还是数据
; (B.0=1)
MOV     R0, 70H          ;是地址,装入内部 RAM 78H-75H 单元中
MOV     A, SBUF
MOV     @R0, A
DEC     70H
CLR     RI
MOV     A, #74H
CJNE    A, 70H, RETURN   ;判前 4 个字节地址是否已完全装入内
; 部 RAM 78H-75H 中
SETB    B.0              ; B.0=1,表示地址已接收完
RETURN: POP     A
MOV     R0, A
POP     A
POP     DPH
POP     DPL
RETI

DATA1:  MOV     DPH, 78H   ;接收的是数据,则转入此处
MOV     DPL, 77H
MOV     A, SBUF
MOVX    @DPTR, A          ;将接收到的数据送入片外 RAM 中

```

```

        CLR    RI
        INC    77H          ;地址加 1
        MOV    A,77H
        JNZ    END2
        INC    78H
END2:   MOV    A,76H
        CJNE   A,78H,RETURN
        MOV    A,75H
        CJNE   A,77H,RETURN
        CLR    ES
        CLR    ET1
        CLR    TI
        SETB   F0          ;PSW. 5=1,设接收完数据块标志
        AJMP   RETURN

```

7.5.2 串行口方式2应用编程

方式2和方式1有2点不同之处,方式2接收/发送11位信息,第0位为起始位0,第1~8位为数据位,第9位是程控位,该位可由用户置TB8决定,第10位是停止位1,这是方式1和方式2的一个不同点,另一个不同点是方式2的波特率变化范围比方式1小,方式2的波特率=振荡器频率/ n :

当 $SMOD=0$ 时, $n=64$;

当 $SMOD=1$ 时, $n=32$ 。

鉴于方式2的使用和方式3基本一样(方式3的波特率要由用户决定),所以具体使用方法可以参照方式3。

7.5.3 串行口方式3应用编程(双机通信)

方式3和方式1的区别就在于发送或接收的位数不同,方式3发送接收的位数以及每位代表的含义和方式2相同。

例7-5 下面程序中,T1作为波特率发生器,选择为方式2。串行口选择方式3进行发送和接收。甲机用查询方式发地址,中断方式发数据。乙机接收采用中断方式。甲机使用晶振频率为6MHz,波特率为2400b/s。

1. 甲机发送程序

甲机首先发送地址,该地址高位存放在78H中,低位存放在77H中,然后发送00H,01H,02H,...,FEH,共255个数据,最后结束。

```

                ORG    0000H
                LJMP   TRANS
                ORG    0023H
                LJMP   SINT
                ORG    0100H
TRANS:  MOV     A, #20H           ;置定时器 T1 为方式 2
        MOV     TMOD, A
        MOV     TL1, #0F3H       ;定时器 T1 置初值
        MOV     TH1, #0F3H
        MOV     PCON, #80H
        SETB    EA               ;CPU 开中断
        CLR     ES
        CLR     ET1
        SETB    TR1              ;定时器 T1 开始工作
        MOV     SCON, #0E0H      ;置串行口为方式 3
        SETB    TB8
        MOV     SBUF, 78H        ;输出高位地址
WAIT1:  JNB     TI, WAIT1
        CLR     TI
        MOV     SBUF, 77H
        MOV     IE, #90H         ;允许串行口中断
        CLR     TB8              ;表示以后发送的是数据
        MOV     A, #00H
WAIT2:  CJNE    A, #0FFH, WAIT2   ;判别结束否
        CLR     ES
WAIT:   AJMP    WAIT             ;其他程序段
SINT:   CLR     TI               ;串行口中断服务程序
        MOV     SBUF, A
        INC     A
        RETI

```

2. 乙机接收程序

乙机把接收的头 2 B 作为首地址,再把接收的 256 B 的数放入以首地址开始的单元中。

乙机晶振频率为 11.059 2 MHz,波特率为 2 400 b/s。

下面是乙机接收程序清单:

```

                ORG    0000H
                LJMP   MAIN
                ORG    0023H

```

```

        LJMP    RE
        ORG     0100H
MAIN:    MOV     SP, #60H
        ACALL  RECEIVE
        SJMP    $
RECEIVE: MOV     A, #20H           ;置定时器 T1 为方式 2
        MOV     TMOD, A
        MOV     TL1, #0E8H        ;定时器 T1 置初值
        MOV     TH1, #0E8H
        MOV     PCON, #80H        ;SMOD=1
        SETB    EA
        CLR     ET1
        SETB    ES
        SETB    TR1               ;定时器 T1 开始工作
        MOV     SCON, #0F0H       ;置串行口方式 3
        MOV     R0, #0FEH
        SETB    F0                ;设置标志位, F0=1 是在接收数据,
                                   ;F0=0 接收完毕
WAIT:    MOV     C, F0
        JC      WAIT              ;其他程序段
        RET
RE:      MOV     C, RB8            ;串行口中断服务程序
        JNC     PD2               ;输入是否地址
        INC     R0
        MOV     A, R0
        JZ      PD
        MOV     DPH, SBUF         ;输入的是高位地址
        AJMP    PD1
PD:      MOV     DPL, SBUF         ;输入的是低位地址
        CLR     SM2               ;下一次输入是数据标志
PD1:     CLR     RI
        RETI
PD2:     MOV     A, SBUF           ;是数据
        MOVX    @DPTR, A
        INC     DPTR
        CLR     RI
        CJNE    A, #0FEH, RETURN  ;判别是否结束
        SETB    SM2              ;恢复数据标志位

```

CLR F0

RETURN; RETI

一般来说,定时器方式 2 用来确定波特率是比较理想的,它不需要中断服务程序设置初值,且算出的波特率比较准确。在用户使用的波特率不是很低的情况下,建议使用定时器 T1 的方式 2 来确定波特率。

思考题及习题

1. 串行数据传送的主要优点和用途是什么?
2. 简述串行口接收和发送数据的过程。
3. 帧格式为 1 个起始位,8 个数据位和 1 个停止位的异步串行通信方式是方式()。
4. 串行口有几种工作方式?有几种帧格式?各种工作方式的波特率如何确定?
5. 假定串行口串行发送的字符格式为 1 个起始位,8 个数据位,1 个奇校验位,1 个停止位,请画出传送字符“A”的帧格式。
6. 判断下列说法是否正确:
 - (A) 串行口通信的第 9 数据位的功能可由用户定义。
 - (B) 发送数据的第 9 数据位的内容在 SCON 寄存器的 TB8 位中预先准备好的。
 - (C) 串行通讯帧发送时,指令把 TB8 位的状态送入发送 SBUF 中。
 - (D) 串行通讯接收到的第 9 位数据送 SCON 寄存器的 RB8 中保存。
 - (E) 串行口方式 1 的波特率是可变的,通过定时器/计数器 T1 的溢出率设定。
7. 通过串行口发送或接收数据时,在程序中应使用:
 - (A) MOVC 指令 (B) MOVX 指令 (C) MOV 指令 (D) XCHD 指令
8. 为什么定时器/计数器 T1 用做串行口波特率发生器时,常采用方式 2?若已知时钟频率、通讯波特率,如何计算其初值?
9. 串行口工作方式 1 的波特率是:
 - (A) 固定的,为 $f_{osc}/32$ 。
 - (B) 固定的,为 $f_{osc}/16$ 。
 - (C) 可变的,通过定时器/计数器 T1 的溢出率设定。
 - (D) 固定的,为 $f_{osc}/64$ 。
10. 在串行通讯中,收发双方对波特率的设定应该是()的。
11. 若晶体振荡器为 11.059 2 MHz,串行口工作于方式 1,波特率为 4 800 b/s,写出用 T1 作为波特率发生器的方式控制字和计数初值。
12. 简述利用串行口进行多机通讯的原理。
13. 使用 8031 的串行口按工作方式 1 进行串行数据通讯,假定波特率为 2 400 b/s,以中断方式传送数据,请编写全双工通讯程序。
14. 使用 8031 的串行口按工作方式 3 进行串行数据通讯,假定波特率为 1 200 b/s,第 9 数据位作奇偶校验位,以中断方式传送数据,请编写通讯程序。

15. 某 8031 串行口, 传送数据的帧格式为 1 个起始位(0), 7 个数据位, 1 个偶校验和 1 个停止位(1)组成。当该串行口每分钟传送 1 800 个字符时, 试计算出波特率。
16. 为什么 MCS-51 串行口的方式 0 帧格式没有起始位(0)和停止位(1)?

第 8 章 MCS-51 单片机扩展 存储器的设计

8.1 概述

MCS-51 单片机片内集成了各种存储器和 I/O 功能部件,但有时根据应用系统的功能需求,片内的资源还不能满足需要,还需要外扩存储器和 I/O 功能部件(也称 I/O 接口部件),这就是通常所说的 MCS-51 单片机的系统扩展问题。

MCS-51 系统扩展的内容主要有外部存储器的扩展(外部存储器又分为外部程序存储器和外部数据存储器)和 I/O 接口部件的扩展。本章介绍 MCS-51 单片机如何扩展外部存储器,有关 I/O 接口部件的扩展将在下一章介绍。

MCS-51 系统扩展结构如图 8-1 所示。

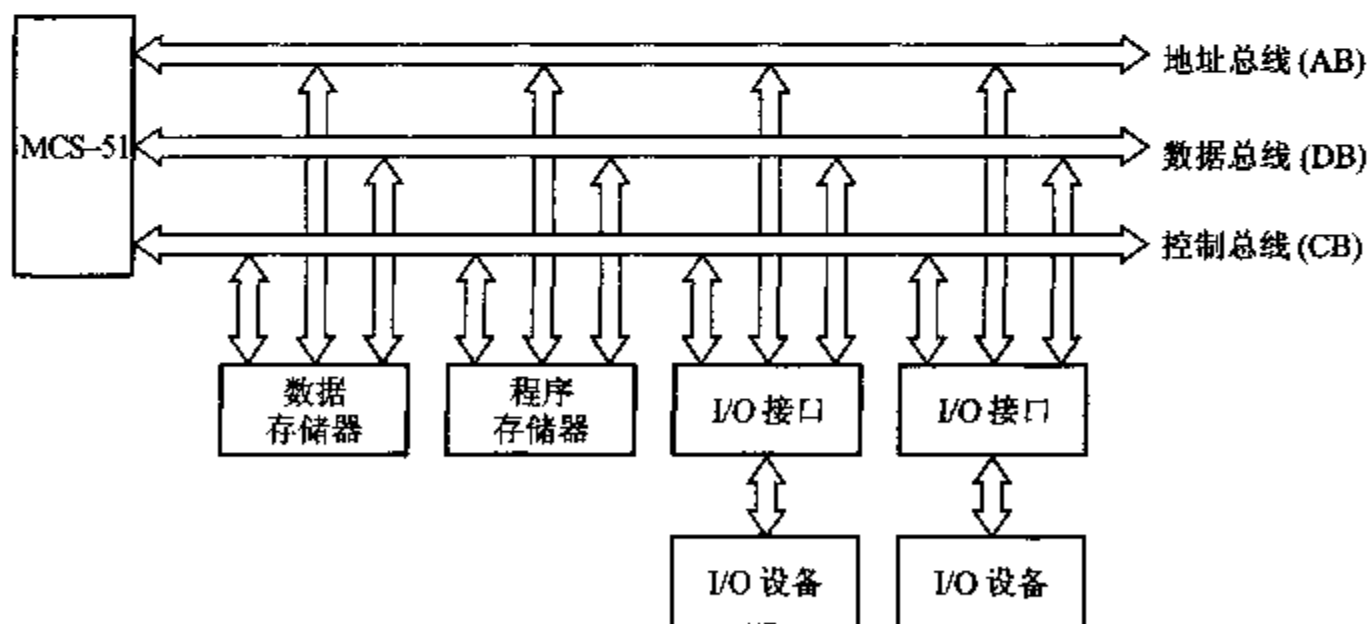


图 8-1 MCS-51 的系统扩展结构

由图 8-1 可以看出:系统扩展是以 MCS-51 单片机为核心进行的。扩展内容包括扩展程序存储器(ROM)、数据存储器(RAM)、I/O 接口部件及 I/O 设备等。

MCS-51 单片机外部存储器结构,采用的是哈佛结构,即程序存储器的空间和数据存储器的空间是截然分开的。还有一种外部存储器的结构,它是程序存储器和数据存储器合用一个空间的结构:普林斯顿结构。例如,MCS-96 单片机的存储器结构就是采用普林斯顿结构。MCS-51 单片机数据存储器和程序存储器的最大扩展空间各为 64 KB,扩展后,系统形成了 2 个并行的 64 KB 外部存储器空间。

由图 8-1 可以看出,扩展是通过系统总线进行的,通过总线把 MCS-51 单片机与各扩展部分连接起来,并进行数据、地址和控制信号的传送。因此,要进行系统扩展首先要构造系统总线。

8.2 系统总线及总线构造

8.2.1 系统总线

所谓总线,就是连接计算机各部件的一组公共信号线。MCS-51 使用的是并行总线结构,按其功能通常把系统总线分为三组,即:

1. 地址总线(Address Bus,简写 AB)

地址总线用于传送单片机发出的地址信号,以便进行存储单元和 I/O 端口的选择。地址总线是单向的,只能由单片机向外送出。地址总线的数目决定着可直接访问的存储单元的数目。例如, n 位地址,可以产生地址的数目为 2^n 个连续地址编码,因此可以访问 2^n 个存储单元,即通常所说的寻址范围为 2^n 个地址单元。MCS-51 单片机最多可以扩展 64 KB,即 65 536 个地址单元,因此,地址总线为 16 条。

2. 数据总线(Data Bus,简写 DB)

数据总线用于在单片机与存储器之间或单片机与 I/O 之间传送数据。单片机系统数据总线的位数与单片机处理数据的字长一致。MCS-51 单片机是 8 位字长,所以,数据总线的位数也是 8 位的。数据总线是双向的,可以进行 2 个方向的传送。

3. 控制总线(Control Bus,简写 CB)

控制总线实际上就是一组控制信号线,包括单片机发出的,以及从其它部件传送给单片机的。对于一条具体的控制信号线来说,其传送方向是单向的,但是由不同方向的控制信号线组合的控制总线则表示为双向。

由于单片机系统采用总线结构形式,可以大大减少单片机系统中传输线的数目,提高了系统的可靠性,增加了系统的灵活性。此外,总线结构也使扩展易

于实现,各功能部件只要符合总线规范就可以很方便地接入系统,实现单片机的系统扩展。

8.2.2 构造系统总线

既然单片机的扩展系统是并行总线结构,因此单片机系统扩展的首要问题是构造系统总线,然后再往系统总线上“挂”存储器芯片或 I/O 接口芯片,“挂”存储器芯片就是存储器扩展,“挂”I/O 接口芯片就是 I/O 扩展。

MCS-51 单片机受引脚数目的限制,数据线和低 8 位地址线是复用的,由 P0 口线兼用。为了将它们分离出来,需要在单片机外部增加地址锁存器,从而构成与一般 CPU 相类似的片外三总线,如图 8-2 所示。

地址锁存器一般采用 74LS373,采用 74LS373 的地址总线的扩展电路如图 8-3 所示。

由 MCS-51 的 P0 口送出的低 8 位有效地址信号是在 ALE(地址锁存允许)信号变高的同时出现的,并在 ALE 由高变低时,将出现在 P0 口的地址信号锁存到外部地址锁存器 74LS373 中,随后,P0 口又作为数据总线口。下面说明总线的具体构造方法。

1. 以 P0 口作为低 8 位地址/数据总线

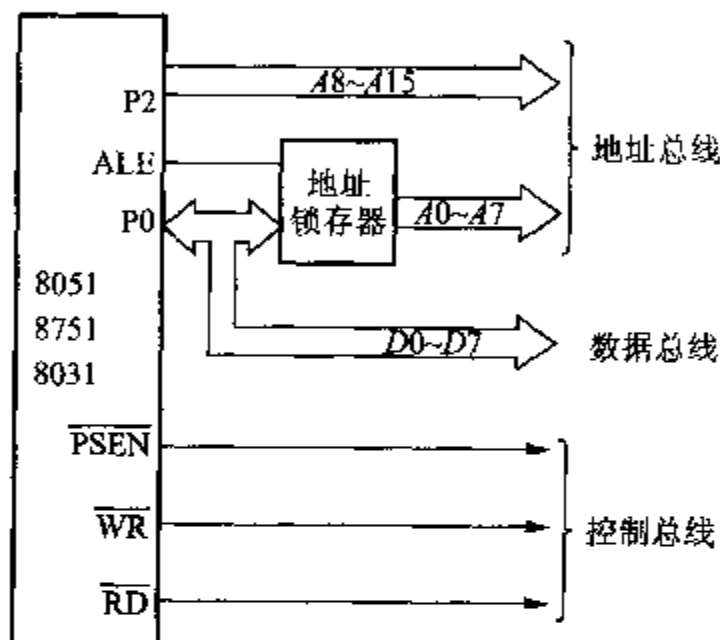


图 8-2 MCS-51 扩展的三总线

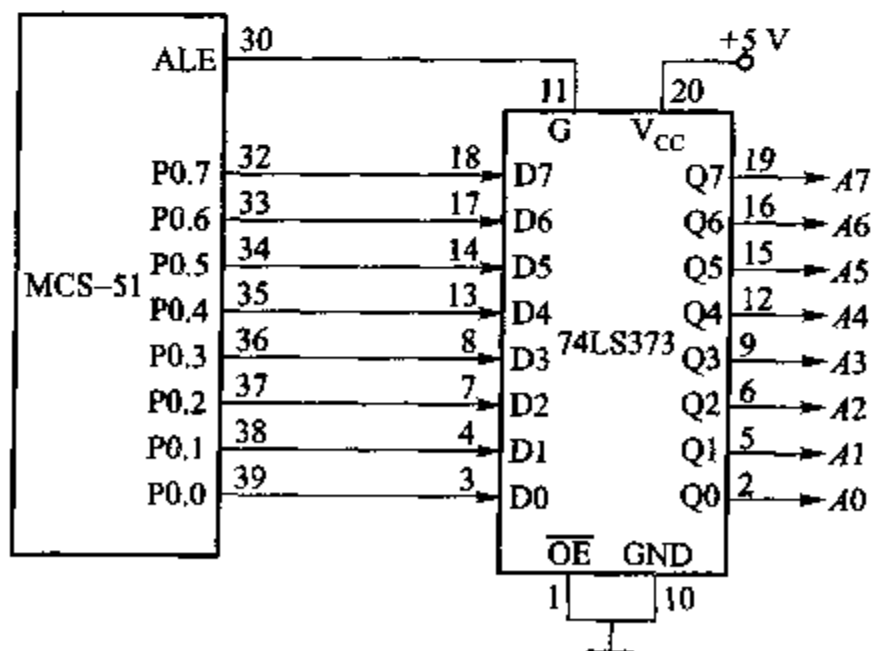


图 8-3 MCS-51 地址总线扩展电路

因为 P0 口既作低 8 位地址线,又作数据总线(分时复用),因此,需要增加 1 个 8 位锁存器。在实际应用时,先把低 8 位地址送锁存器暂存,地址锁存器的输出给系统提供低 8 位地址,而把 P0 口线作为数据线使用。实际上,MCS-51 单片机的 P0 口的电路设计已经考虑了这种应用要求,P0 口线内部电路中的多路转接电路 MUX 以及地址/数据控制(见第 2 章的图 2-10)就是为此目的而设计的。

2. 以 P2 口的口线作高位地址线

如果使用 P2 口的全部 8 位口线,再加上 P0 口提供的低 8 位地址,便形成了完整的 16 位地址总线,使单片机系统的寻址范围达到 64 KB。

但在实际应用系统中,高位地址线并不固定为 8 位,需要用几位就从 P2 口中引出几条口线。

3. 控制信号线

除了地址线和数据线之外,在扩展系统中还需要一些控制信号线,以构成扩展系统的控制总线。这些信号有的就是单片机引脚的第一功能信号,有的则是 P3 口第二功能信号。其中包括:

- (1) 使用 ALE 信号作为低 8 位地址的锁存控制信号。
- (2) 以 \overline{PSEN} 信号作为扩展程序存储器的读选通信号。
- (3) 以 \overline{EA} 信号作为内外程序存储器的选择控制信号。
- (4) 由 \overline{RD} 和 \overline{WR} 信号作为扩展数据存储器 and I/O 口的读选通、写选通信号。

可以看出,尽管 MCS-51 单片机有 4 个并行的 I/O 口,共 32 条口线,但由于系统扩展的需要,真正作为数据 I/O 使用的,就剩下 P1 口和 P3 口的部分口线了。

8.2.3 单片机系统的串行扩展技术

随着单片机技术的发展,并行总线扩展(利用 3 组总线 AB、DB、CB 进行的系统扩展)已不再是单片机惟一的系统扩展结构了,随着集成电路芯片的集成度和结构的发展,近年来除并行总线扩展技术之外,又出现了串行总线扩展技术。

串行扩展技术具有显著的优点,一般地说,串行接口器件体积小,因而,所占电路板的空间,仅为并行接口器件的 10%,明显地减少了电路板空间和成本。串行接口器件与单片机接口时需要的 I/O 口线很少(仅需 3~4 根),极大地简化了器件之间的连接,进而提高了可靠性。

串行扩展是通过串行接口实现的,这样可以减少芯片的封装引脚,降低成本,简化系统结构,增加系统扩展的灵活性。为了实现串行扩展,一些公司(例如 PHILIPS 和 ATMEL 公司等)已经推出了非总线型单片机芯片,并且具有 SPI(Serial Peripheral Interface)三线总线和 I²C 公用双总线的两种串行总线形式。与此

相配套,也推出了相应的串行外围接口芯片。

但是,一般串行接口器件速度较慢,在大多数应用场合,还是并行扩展占主导地位。在进行系统扩展时,应对单片机的系统扩展能力、扩展总线及扩展应用的特点有所了解,这样才能顺利完成系统扩展任务。本书仅介绍并行扩展法,有关串行扩展法,读者也要引起重视,并请读者查阅有关资料和参考文献。

8.3 读写控制、地址空间分配和外部地址锁存器

8.3.1 存储器扩展的读写控制

外扩的 RAM 芯片既能读出又能写入,所以通常都有读写控制引脚,记为 \overline{OE} 和 \overline{WE} 。外扩 RAM 的读、写控制引脚分别与 MCS-51 的 \overline{RD} 和 \overline{WR} 引脚相连。

外扩的 EPROM 在正常使用中只能读出,不能写入,故 EPROM 芯片没有写入控制引脚,只有读出引脚,记为 \overline{OE} ,该引脚与 MCS-51 单片机的 \overline{PSEN} 相连。

8.3.2 存储器地址空间分配

在实际的单片机应用系统设计中,既需要扩展程序存储器,往往又需要扩展数据存储器。在 MCS-51 扩展多片的程序存储器、数据存储器芯片的情况下,如何把外部各自的 64 KB 空间分配给各个芯片,并且使程序存储器的各个芯片之间,数据存储器(I/O 接口芯片也作为数据存储器一部分)各芯片之间,地址不发生重叠,从而避免发生数据冲突。这就是存储器的地址空间的分配问题。存储器的地址空间分配,实际上就是通过地址线,与存储器芯片的地址引脚适当连接,最终达到一个存储器单元对应一个地址的要求。

MCS-51 发出的地址是用来选择某个存储器单元,在外扩的多片存储器芯片中,MCS-51 要完成这种功能,必须进行两种选择:一是必须选中该存储器芯片(或 I/O 接口芯片),这称为片选,只有被选中的存储器芯片才能被 MCS-51 读出或写入数据。二是必须选择该芯片的某一单元,称为单元选择。为了芯片选择(片选)的需要,每个存储器芯片都有片选信号引脚,因此芯片选择的实质就是如何通过 MCS-51 的地址线来产生芯片的片选信号。

通常把单片机系统的地址笼统地分为低位和高位地址,存储器芯片的某一存储单元选择使用低位地址,剩下的高位地址才作为芯片选择使用,因此芯片的选择都是使用高位地址线。实际上,在 16 根地址线中,高、低位地址线的数目并

不是固定的,我们只是把用于存储单元选择所使用的地址线,都称为低位地址线,其余的就为高位地址线。

存储器地址空间分配除了考虑地址线的连接外,还需考虑各存储器芯片在整个存储空间中所占据的地址范围,以便在程序设计时正确地应用它们。

常用的存储器地址分配的方法有两种:线性选择法(简称线选法)和地址译码法(简称译码法),下面分别予以介绍。

1. 线选法

线选法就是直接利用系统的高位地址线作为存储器芯片(或 I/O 接口芯片)的片选信号。为此,只需要把用到的地址线与存储器芯片的片选端直接连接即可。线选法的优点是电路简单,不需要地址译码器硬件,体积小,成本低。缺点是可寻址的器件数目受到限制,故只用于不太复杂的系统中,另外,地址空间不连续,每个存储单元的地址不惟一,这会给程序设计带来一些不方便。

下面通过一个具体例子,来说明线选法的具体应用。

假如某一单片机系统,需要外扩 8 KB 的 EPROM(2 片 2732),4 KB 的 RAM(2 片 6116),这些芯片与 MCS-51 单片机地址分配有关的地址线连线电路如图 8-4 所示。

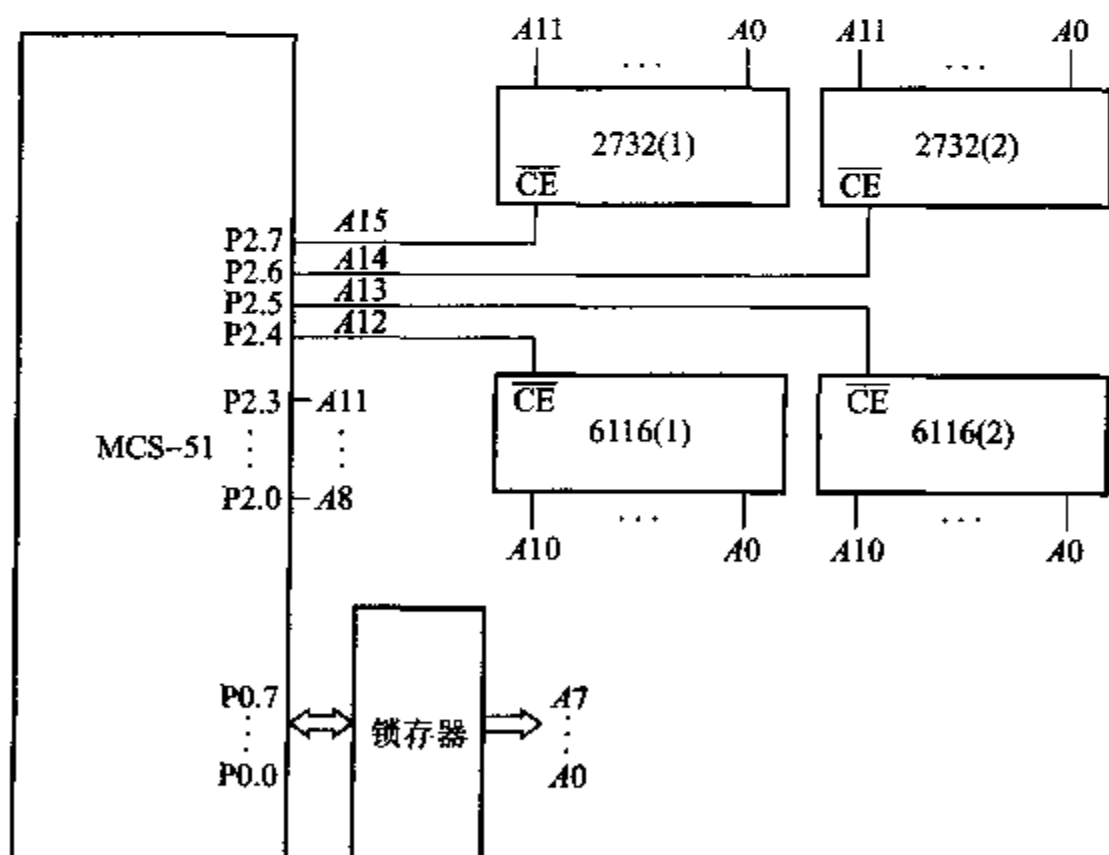


图 8-4 线选法举例

先看程序存储器 2732 与 MCS-51 的连接。由于 2732 是 4 KB 的程序存储器,有 12 根地址线 A0~A11,它们分别与单片机的 P0 口及 P2.0~P2.3 口相连,从而实现 4 KB 范围内的单元选择。由于系统中有 2 片程序存储器,存在 2

片程序存储器芯片之间相区别的问题。2732(1)片选端 \overline{CE} 接 A15(P2.7), 2732(2)片选端 \overline{CE} 接 A14(P2.6), 当要选中某个芯片时, 单片机 P2 口对应的片选信号引脚应为低电平, 其它引脚一定要为高电平。这样才能保证 1 次只选中 1 片, 而在同时不再选中其它同类存储器芯片, 这就是所谓的线性选择地址法, 简称线选法。

再来看数据存储器与单片机的接口。数据存储器也有 2 片芯片需要区别。这里用 P2.5 和 P2.4 输出信号分别作为这 2 片芯片的片选信号。当要选中某个芯片时, 单片机 P2 口对应的片选信号引脚应为低电平, 其它引脚一定要为高电平。由于 6116 是 2 KB 的, 需要 11 根地址线作为存储单元的选择, 而剩下的 P2 口线 (P2.4~P2.7) 正好作为片选线。

从图 8-4 中可以看出, 程序存储器 2732 的低 2 KB 和数据存储器 6116 的地址是重叠的。那么会不会 MCS-51 发出访问 2732 某个单元的地址时, 同时也会选中 6116 的某个单元, 这样 MCS-51 就会同时选中 2 个单元, 从而发生数据冲突, 产生错误呢? 这种情况, 完全不用担心, 虽然 2 个单元的地址是一样的, 但是 MCS-51 发给 2 类存储器的控制信号是不一样的。访问程序存储器, 是 \overline{PSEN} 信号有效; 访问数据存储器, 则是 \overline{RD} 或 \overline{WR} 信号有效。以上控制信号是由 MCS-51 执行访问外部程序存储器或访问外部数据存储器的指令产生, 任何时刻只能执行 1 种指令, 只产生 1 种控制信号, 所以不会产生数据冲突的问题。通过上面的讨论, 可以得出 1 个重要的结论: MCS-51 单片机外扩程序存储器和数据存储器的地址空间可以重叠, 只是注意程序存储器和程序存储器之间, 数据存储器 and 数据存储器之间, 千万不要发生地址重叠。

现在再来看 2 个程序存储器的地址范围。

2732(1)的地址范围:

选中 2732(1)时, P2 口(高 8 位的地址)各引脚的状态为:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
0	1	1	1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见高 8 位的地址变化范围: 70H~7FH。

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见低 8 位的地址变化范围: 00H~FFH。

所以 2732(1)的地址变化范围为 7000H~7FFFH。

2732(2)的地址范围:

选中 2732(2)时, P2 口(高 8 位的地址)各引脚的状态为:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
1	0	1	1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见高 8 位的地址变化范围: B0H~BFH。

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见低 8 位的地址变化范围: 00H~FFH。

所以 2732(2)的地址变化范围为: B000H~BFFFH。

现在再来看 2 个数据存储器的地址范围。

6116(1)的地址范围:

选中 6116(1)时, P2 口(高 8 位的地址)各引脚的状态为:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
1	1	1	0	1	0 或 1	0 或 1	0 或 1

由上可见高 8 位的地址变化范围: E8H~EFH。

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见低 8 位的地址变化范围: 00H~FFH。

所以 6116(1)的地址变化范围为: E800H~EFFFH。

6116(2)的地址范围:

选中 6116(2)时, P2 口(高 8 位的地址)各引脚的状态为:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
1	1	0	1	1	0 或 1	0 或 1	0 或 1

由上可见高 8 位的地址变化范围: D8H~DFH。

P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1	0 或 1

由上可见低 8 位的地址变化范围: 00H~FFH。

所以 6116(2)的地址变化范围为: D800H~DFFFH。

由上可见,线选法的特点是简单明了,不需要另外增加硬件电路。但是,这种方法对存储器空间的利用是断续的,不能充分有效地利用存储空间,扩展的存储器容量有限,只适用于外扩的芯片数目不多,规模不大的单片机系统的存储器扩展。

2. 译码法

译码法就是使用译码器对 MCS-51 的高位地址进行译码,译码器的译码输出作为存储器芯片的片选信号。这是一种最常用的存储器地址分配的方法,它能有效的利用存储器空间,适用于大容量多芯片的存储器扩展。译码电路可以使用现有的译码器芯片。最常用的译码器芯片有:74LS138(3 线-8 线译码器),74LS139(双 2 线-4 线译码器),74LS154(4 线-16 线译码器)。它们使用灵活,完全可以根据设计者的要求来组合译码,产生片选信号。若全部高位地址线都参加译码,称为全译码;若仅仅部分高位地址线参加译码,称为部分译码,部分译码存在着部分存储器地址空间相重叠的情况。

下面介绍几种常用的译码器芯片。

(1) 74LS138

74LS138 是 1 种 3 线-8 线译码器,有 3 个数据输入端,经译码产生 8 种状态。其引脚如图 8-5 所示,译码功能如表 8-1 所示。由表 8-1 可见,当译码器的输入为某一个编码时,就有 1 个固定的引脚输出为低电平,其余的为高电平。

表 8-1 74LS138 真值表

输 入						输 出							
$G1$	$\overline{G2A}$	$\overline{G2B}$	C	B	A	$\overline{Y7}$	$\overline{Y6}$	$\overline{Y5}$	$\overline{Y4}$	$\overline{Y3}$	$\overline{Y2}$	$\overline{Y1}$	$\overline{Y0}$
1	0	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	0	1	1	1	1	1	1	1	0	1
1	0	0	0	1	0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	0	1	1	1
1	0	0	1	0	0	1	1	1	0	1	1	1	1
1	0	0	1	0	1	1	1	0	1	1	1	1	1
1	0	0	1	1	0	1	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	1	1	1	1	1
其它状态			×	×	×	1	1	1	1	1	1	1	1

注:1 表示高电平,0 表示低电平,×表示任意

(2) 74LS139

74LS139 是一种双 2 线-4 线译码器。这两个译码器完全独立,分别有各自

的数据输入端、译码状态输出端以及数据输入允许端。其引脚如图 8-6 所示, 真值表如表 8-2 所示(只给出其中的一组)。

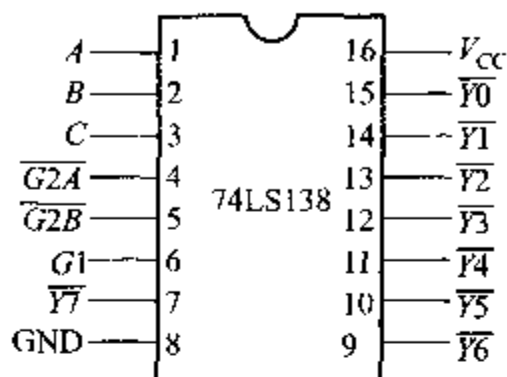


图 8-5 74LS138 引脚图

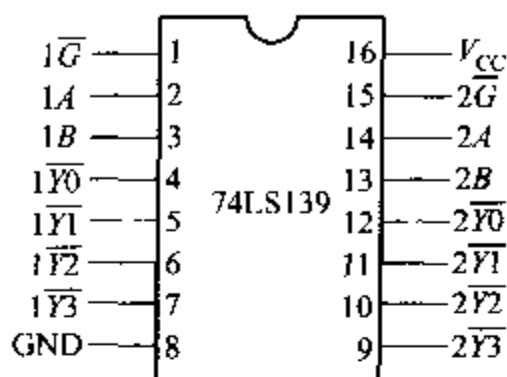


图 8-6 74LS139 引脚图

表 8-2 74LS139 真值表

输 入 端			输 出 端			
允 许	选 择					
\bar{G}	B	A	\bar{Y}_0	\bar{Y}_1	\bar{Y}_2	\bar{Y}_3
1	×	×	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

下面我们以 74LS138 为例,来介绍如何进行地址分配。例如要扩 8 片 8 KB 的 RAM 6264,如何通过 74LS138 把 64 KB 空间分配给各个芯片? 由 74LS138 真值表可知,把 G1 接到 +5 V, \bar{G}_2A 、 \bar{G}_2B 接地, P2.7、P2.6、P2.5 分别接到 74LS138 的 C、B、A 端, P2.4~P2.0, P0.7~P0.0 这 13 根地址线接到 8 片 6264 的 A12~A0 引脚。

由于对高 3 位地址译码,这样译码器有 8 个输出 $\bar{Y}_0 \sim \bar{Y}_7$, 分别接到 8 片 6264 的片选端,而低 13 根地址线(P2.4~P2.0, P0.7~P0.0)完成对 6264 存储单元的选择。这样就把 64 KB 存储器空间分成 8 个 8 KB 空间了。64 KB 地址空间的分配如图 8-7 所示。

这种除了单元选择的地址线外,剩余的高位地址线全部参加译码的方式称为全地址译码方式。由于采用的是全地址译码方式, MCS-51 单片机发地址码时,每次只能选中一个存储单元。这样,同类存储器之间根本不会产生地址重叠的问题。

如果用 74LS138 把 64 KB 空间全部划分为每块 4 KB,如何划分呢? 由于 4 KB 空间需要 12 根地址线进行单元选择,而译码器的输入有 3 根地址线

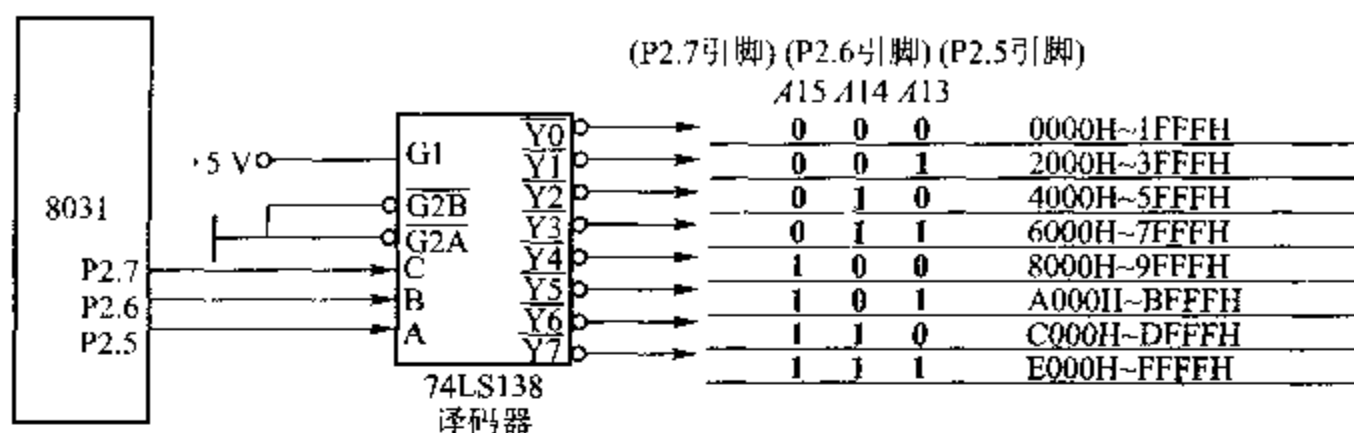


图 8-7 64 KB 地址空间的分配

(P2.6~P2.1), P2.7 没有参加译码, P2.7 发出的 0 或 1 决定了选择 64 KB 存储器空间的前 32 KB 还是后 32 KB. 由于 P2.7 没有参加译码(高位地址线没有全部参加译码, 就不是全译码方式), 这样, 前后 2 个 32 KB 空间就重叠了。但是在实际的应用设计时, 32 KB 存储器空间在大部分情况下是够用的。那么, 这 32 KB 空间利用 74LS138 译码器可划分为 8 个 4 KB 空间。如果把 P2.7 通过 1 个非门与 74LS138 译码器的 G1 端连接起来, 如图 8-8 所示, 这样就不会发生 2 个 32 KB 空间重叠的问题了。这时, 选中的是 64 KB 空间的前 32 KB 空间, 地址范围为 0000H~7FFFH。如果去掉图 8-8 中的非门, 地址范围为 8000H~FFFFH。把译码器的输出连到各个 4 KB 存储器的片选端, 这样就把 32 KB 的空间划分为 8 个 4 KB 空间。P2.3~P2.0, P0.7~P0.0 实现对单元的选择, P2.6~P2.4 通过 74LS138 译码器的译码实现对存储器的片选。

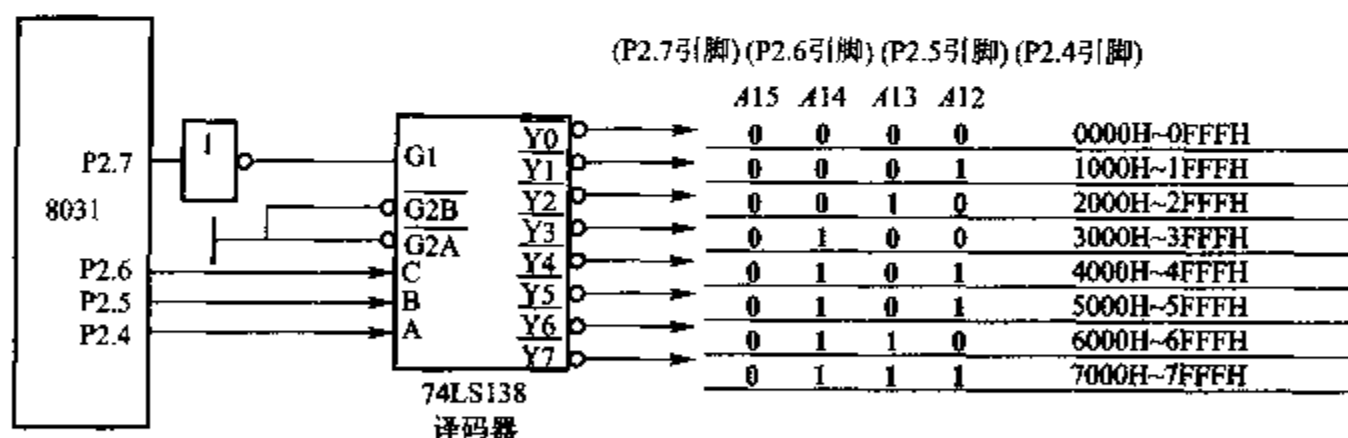


图 8-8 存储器空间被划分成每块为 4 KB

如果利用 74LS138 译码器实现每块为 2 KB 的划分, 这样会产生 4 个 16 KB 存储器空间的划分。如果把 P2.7 同 74LS138 译码器的 G1 端相连, P2.6 同 G2A 端相连, 这样一来就把 64 KB 空间固定为 4 个 16 KB 空间中的 1 个。改变 P2.7、P2.6 同译码器 G1 端、G2A 端连接的逻辑, 即可改变选中 4 个 16 KB 空间中的哪一个。译码器的 8 个输出, 即把 16 KB 空间划分为 2 KB 1 个的存储空间了。读者可自己画出这部分电路以及译码器输出的对应地址范围。

译码器的译码方案是多种多样的,设计者可根据系统的设计要求来选择合适的方案。

8.3.3 外部地址锁存器

MCS-51 单片机受引脚数的限制,数据线和地址线是复用的,由 P0 口线兼用。为了将它们分离出来,以便同单片机片外的扩展芯片正确的连接,需要在单片机外部增加地址锁存器。目前,常用的地址锁存器芯片有:74LS373、8282、74LS573 等。下面就这几种地址锁存器进行介绍,供读者在设计时参考。

1. 锁存器 74LS373

74LS373 是一种带有三态门的 8D 锁存器,其引脚如图 8-9 所示。

其内部结构如图 8-10 所示。

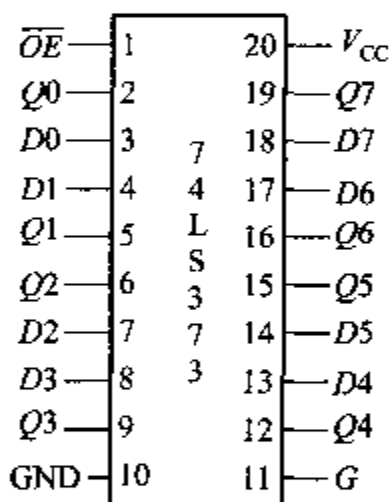


图 8-9 锁存器 74LS373 的引脚

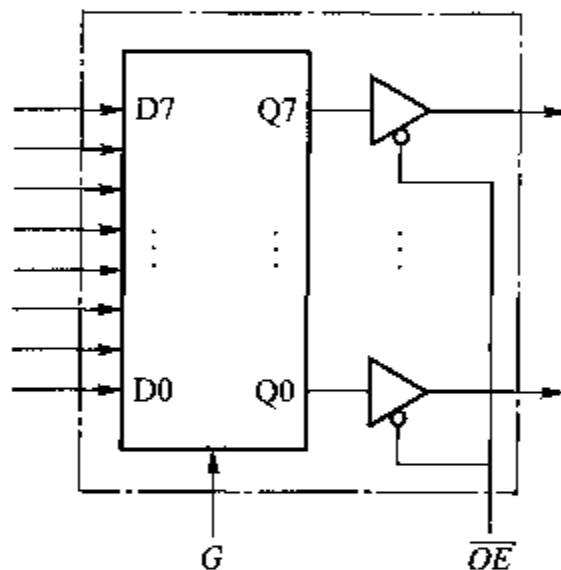


图 8-10 74LS373 的内部结构

其引脚说明如下:

D7~D0:8 位数据输入线。

Q7~Q0:8 位数据输出线。

G:数据输入锁存选通引脚,高电平有效。当该信号为高电平时,外部数据选通到内部锁存器,负跳变时,数据锁存到锁存器中。

\overline{OE} :数据输出允许引脚,低电平有效。当该信号为低电平时,三态门打开,锁存器中数据输出到数据输出线。当该信号为高电平时,输出线为高阻态。

74LS373 功能表如表 8-3 所示。

表 8-3 74LS373 功能表

\overline{OE}	G	D	Q
0	1	1	1

续表

\overline{OE}	G	D	Q
0	1	0	0
0	0	×	不变
1	×		高阻态

2. 锁存器 8282

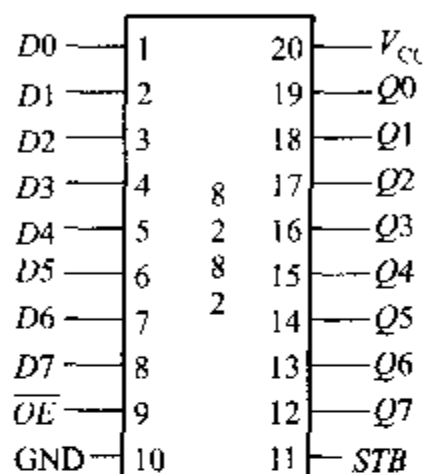


图 8-11 8282 的引脚

Intel 8282 也是一种带有三态输出缓冲的 8D 锁存器, 功能及内部结构与 74LS373 完全一样, 只是其引脚的排列与 74LS373 不同。图 8-11 为 8282 的引脚。

由图 8-11 可以看出, 与 74LS373 相比, 8282 的输入的 D 端和输出的 Q 端各依次排在两侧, 这为绘制印制电路板时的布线提供了方便。

8282 各引脚的功能如下:

D7~D0: 8 位数据输入线。

Q7~Q0: 8 位数据输出线。

STB: 数据输入锁存选通引脚, 高电平有效。当该信号为高电平时, 外部数据选通到内部锁存器, 负跳变时, 数据锁存。该引脚相当于 74LS373 的 G 端。

\overline{OE} : 数据输出允许引脚, 低电平有效。当该信号为低电平时, 锁存器中数据输出到数据输出线。当该信号为高电平时, 输出线为高阻态。

图 8-12 分别给出了 74LS373、8282 芯片作为地址锁存器与 MCS-51 单片机 P0 口的连线方法。

3. 锁存器 74LS573

锁存器 74LS573 引脚的排列与 8282 类似, 输入的 D 端和输出的 Q 端也是依次排在芯片的两侧, 与锁存器 8282 一样, 为绘制印制电路板时的布线提供了方便。74LS573 的功能与 74LS373 相同, 参见表 8-3, 可用来替代 74LS373。74LS573 的引脚见图 8-13。

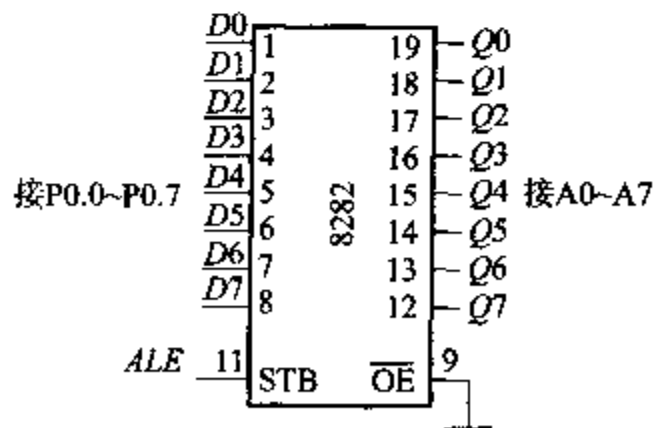
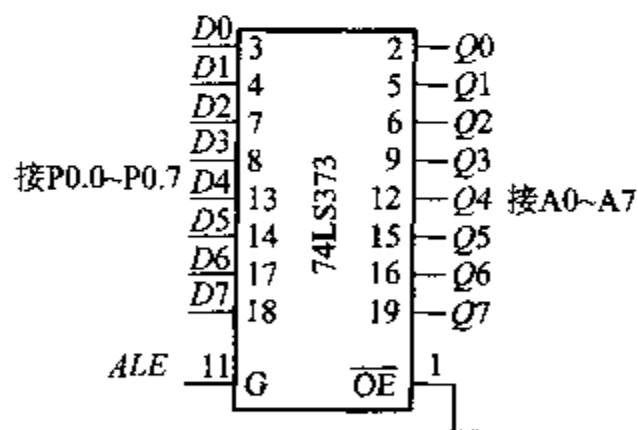


图 8-12 MCS-51 单片机 P0 口与地址锁存器的连接方法

74LS573 各引脚的功能如下:

D7~D0:8 位数据输入线。

Q7~Q0:8 位数据输出线。

G: 数据输入锁存选通引脚。该引脚与 74LS373 G 端的功能相同。

\overline{OE} : 数据输出允许引脚, 低电平有效。当该信号为低电平时, 锁存器中数据输出到数据输出线。当该信号为高电平时, 输出线为高阻态。

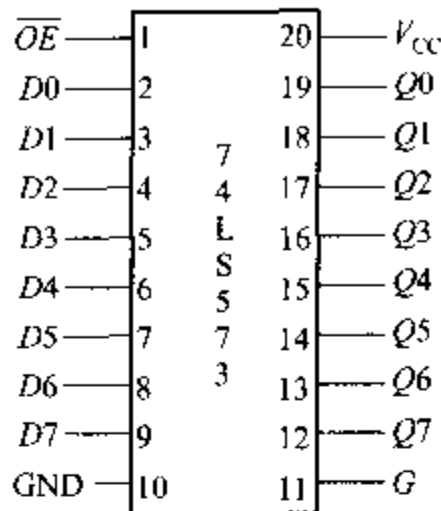


图 8-13 74LS573 的引脚

8.4 程序存储器 EPROM 的扩展

程序存储器一般采用只读存储器, 因为这种存储器在电源关断后, 仍能保存程序(此特性称为非易失性的), 在系统上电后, CPU 可取出这些指令予以重新执行。

只读存储器简称为 ROM(Read Only Memory)。ROM 中的信息一旦写入之后, 就不能随意更改, 特别是不能在程序运行的过程中写入新的内容, 故称之为只读存储器。

向 ROM 中写入信息叫做 ROM 编程。根据编程的方式不同, ROM 分为以下几种:

(1) 掩模 ROM

掩模 ROM 是在制造过程中编程。因编程是以掩模工艺实现的, 因此称为掩模 ROM。这种芯片存储结构简单, 集成度高, 但由于掩模工艺成本较高, 因此只适合于大批量生产。在批量大的生产中, 一次性掩模生产成本才是很低的。

(2) 可编程 ROM(PROM)

PROM(可编程只读存储器)芯片出厂时并没有任何程序信息, 是由用户用独立的编程器写入的。但 PROM 只能写入一次, 写入内容后, 就不能再进行修改。

(3) EPROM

EPROM 是用电信号编程, 用紫外线擦除的只读存储器芯片。在芯片外壳上的中间位置有一个圆形窗口, 通过这个窗口照射紫外线就可擦除原有的信息。

(4) E²PROM(EEPROM)

这是一种用电信号编程, 也用电信号擦除的 ROM 芯片, 对 E²PROM 的读写操作与 RAM 存储器几乎没有什么差别, 只是写入的速度慢一些。但断电后能够保存信息。

(5) Flash ROM

Flash ROM 又称闪烁存储器,简称闪存。Flash ROM 是在 EPROM、E²PROM 的基础上发展起来的一种只读存储器。是一种非易失性、电擦除型存储器。其特点是可快速在线修改其存储单元中的数据,标准改写次数可达 1 万次,而成本却比普通 E²PROM 低得多,因而可大量替代 E²PROM。与 E²PROM 相比,E²PROM 的写入速度较慢。而 Flash ROM 的读写速度都很快,存取时间可达 70 ns。由于其性能比 E²PROM 要好,所以目前大有取代 E²PROM 的趋势。

目前许多公司生产的以 MCS-51 为内核的单片机,在芯片内部集成了数量不等的 Flash ROM。例如,美国 ATMEL 公司生产的兼容 MCS-51 的单片机 89C51/89C52/89C55,片内分别有 4 KB/8 KB/20 KB 的 Flash ROM。对于这类单片机,扩展外部程序存储器的工作即可省去。

8.4.1 常用 EPROM 芯片介绍

程序存储器的扩展可根据需要来使用上述的各种只读存储器的芯片,但使用比较多的是 EPROM、E²PROM,下面首先对常用的 EPROM 芯片进行介绍。

EPROM 的典型芯片是 27 系列产品,例如,2716(2 KB×8)、2732(4 KB×8)、2764(8 KB×8)、27128(16 KB×8)、27256(32 KB×8)、27512(64 KB×8)。型号名称“27”后面的数字表示其位存储容量。如果换算成字节容量,只需将该数字除以 8 即可。例如,“27128”中的“27”后面的数字为“128”, $128 \div 8 = 16$ KB。

随着大规模集成电路技术的发展,大容量存储器芯片的产量剧增,售价不断下降。大容量存储器芯片的性价比明显增高,而且由于有些厂家已停止生产小容量的芯片,使市场上某些小容量芯片的价格反面比大容量芯片还贵(例如,目前 2716、2732 在市场上已经很难买到)。所以,在扩展程序存储器设计时,应尽量采用大容量的芯片。这样,不仅可以使电路板的体积缩小,成本降低,还可以降低整机功耗和减少控制逻辑电路,从而提高系统的稳定性和可靠性。

1. 常用的 EPROM 芯片

27 系列 EPROM 的芯片引脚如图 8-14 所示,参数见表 8-4。

表 8-4 常用 EPROM 芯片参数表

参 数 型 号	V_{CC}/V	V_{PP}/V	I_m/mA	I_1/mA	T_{RM}/ns	容 量
TMS2732A	5	21	132	32	200~450	4 K×8 位
TMS2764	5	21	100	35	200~450	8 K×8 位
INTEL2764A	5	12.5	60	20	200	8 K×8 位
INTEL27C64	5	12.5	10	0.1	200	8 K×8 位

续表

参 数 型 号	V_{CC}/V	V_{PP}/V	I_{in}/mA	I_o/mA	T_{RM}/ns	容 量
INTEL27128A	5	12.5	100	40	150~200	16 K×8 位
SCM27C128	5	12.5	30	0.1	200	16 K×8 位
INTEL27256	5	12.5	100	10	220	32 K×8 位
MBM27C256	5	12.5	8	0.1	250~300	32 K×8 位
INTEL27512	5	12.5	125	10	250	64 K×8 位

在表 8-1 中, V_{CC} 是芯片供电电压, V_{PP} 是编程电压, I_{in} 为最大静态电流, I_o 为维持电流, T_{RM} 为最大读出时间。

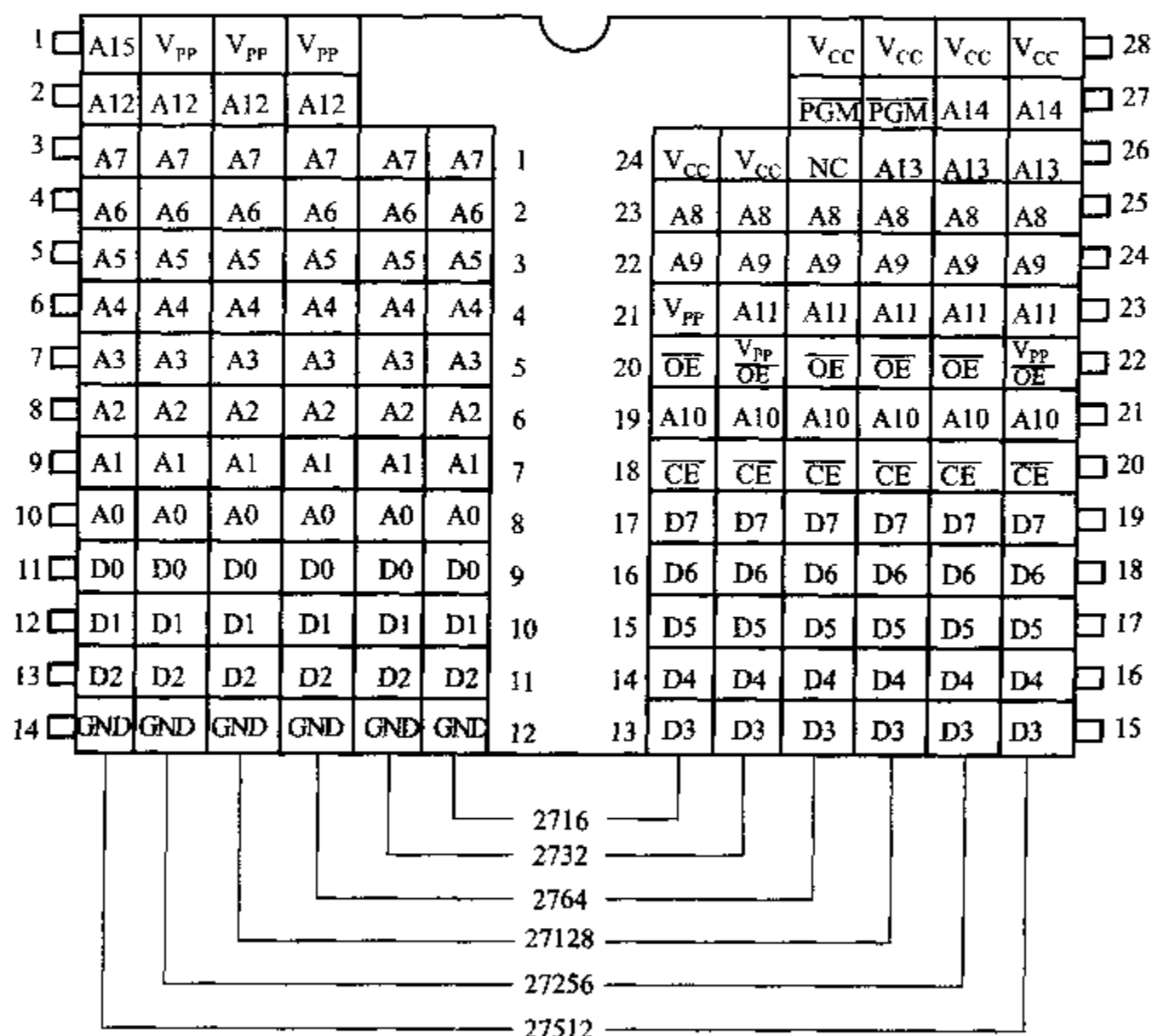


图 8-14 常用 EPROM 芯片引脚图

在图 8-14 中的芯片引脚功能如下:

A0~A15:地址线引脚。地址线引脚的数目由芯片的存储容量来定,用来进行单元选择。

D7~D0:数据线引脚。

\overline{CE} :片选输入端。

\overline{OE} :输出允许控制端。

\overline{PGM} :编程时,加编程脉冲的输入端。

V_{PP} :编程时,编程电压(+12 V 或 +25 V)输入端。

V_{CC} :+5 V,芯片的工作电压输入端。

GND:数字地。

NC:无用端。

2. EPROM 芯片的工作方式

EPROM 一般都有 5 种工作方式,由 \overline{CE} 、 \overline{OE} 、 \overline{PGM} 各信号的状态组合来确定。5 种工作方式如表 8-5 所示。

表 8-5 EPROM 的 5 种工作方式

引脚信号 方式	\overline{CE}/PGM	\overline{OE}	V_{PP}	D7~D0
读出	低	低	+5 V	程序读出
未选中	高	×	+5 V	高阻
编程	正脉冲	高	-25 V(或+12 V)	程序写入
程序校验	低	低	+25 V(或+12 V)	程序读出
编程禁止	低	高	+25 V(或+12 V)	高阻

(1) 读出方式

一般情况下,EPROM 工作在这种方式。工作在此种方式的条件是使片选控制线 \overline{CE} 为低,同时让输出允许控制线 \overline{OE} 为低, V_{PP} 端为 +5 V,就可将 EPROM 中的指定地址单元的内容从数据引脚 D7~D0 上读出。

(2) 未选中方式

当片选控制线 \overline{CE} 为高电平时,芯片进入未选中方式,这时数据输出为高阻抗悬浮状态,不占用数据总线。EPROM 处于低功耗的维持状态。

(3) 编程方式

在 V_{PP} 端加上规定好的高压, \overline{CE} 和 \overline{OE} 端加上合适的电平(不同的芯片要求不同),就能将数据线上的数据写入到指定的地址单元。此时,编程地址和编程数据分别由系统的 A15~A0 和 D7~D0 提供。

(4) 编程校验方式

在 V_{PP} 端保持相应的编程电压(高压),再按读出方式操作,读出编程固化好的内容,以校验写入的内容是否正确。

(5) 编程禁止方式

本工作方式输出呈高阻状态,不写入程序。

8.4.2 程序存储器的操作时序

1. 访问程序存储器的控制信号

MCS-51 单片机访问片外程序存储器时,所用的控制信号有:

- ① ALE ——用于低 8 位地址锁存控制。
- ② \overline{PSEN} ——片外程序存储器读选通控制信号。 \overline{PSEN} 端接外扩 EPROM 的 \overline{OE} 引脚。
- ③ \overline{EA} ——片内、片外程序存储器访问的控制信号。 $\overline{EA}=1$ 时,访问片内程序存储器;当 $\overline{EA}=0$ 时,访问片外程序存储器。

如果指令是从片外 EPROM 中读取的,除了 ALE 为低 8 位地址锁存信号之外,控制信号还有 \overline{PSEN} , \overline{PSEN} 端接外扩 EPROM 的 \overline{OE} 引脚。此外,还要用到 P0 口和 P2 口,P0 口分时用作低 8 位地址总线 and 数据总线,P2 口用作高 8 位地址线。相应的时序如图 8-15 所示。

2. 操作时序

由于 MCS-51 单片机中 ROM 和 RAM 是严格分开的,因此,对片外 ROM 的操作时序分为两种情况:执行非 MOVX 指令的时序,如图 8-15(a)所示;执行 MOVX 指令的时序,如图 8-15(b)所示。

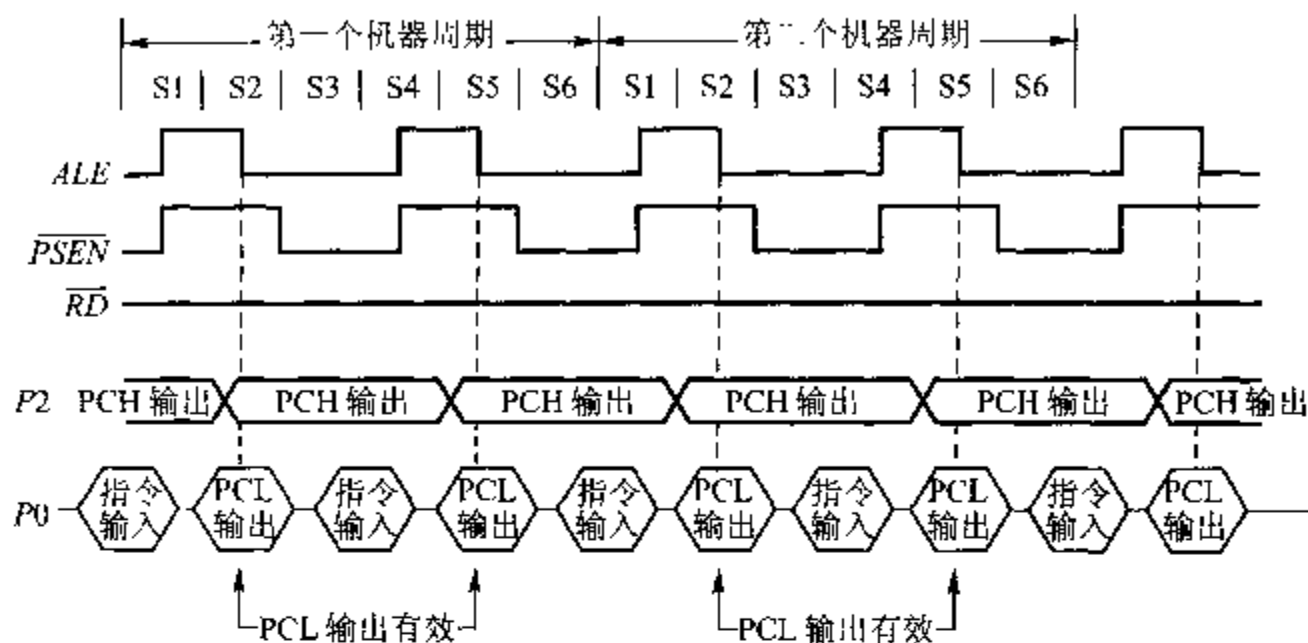
(1) 应用系统中无片外 RAM

无片外 RAM,则不用执行 MOVX 指令。在执行非 MOVX 指令时,操作时序如图 8-15(a)所示,P0 口作为地址/数据复用的双向总线,用于输入指令或输出程序存储器的低 8 位地址 PCL。P2 口专门用于输出程序存储器的高 8 位地址 PCH。P2 口具有输出锁存功能;而 P0 口除输出地址外,还要输入指令,故要用 ALE 来锁存 P0 口输出的地址 PCL。在每个机器周期中,允许地址锁存器 2 次有效, ALE 在下降沿时,锁存出现在 P0 口上的低 8 位 PCL。同时, \overline{PSEN} 也是每个机器周期中 2 次有效,用于选通片外程序存储器,将指令读入片内。

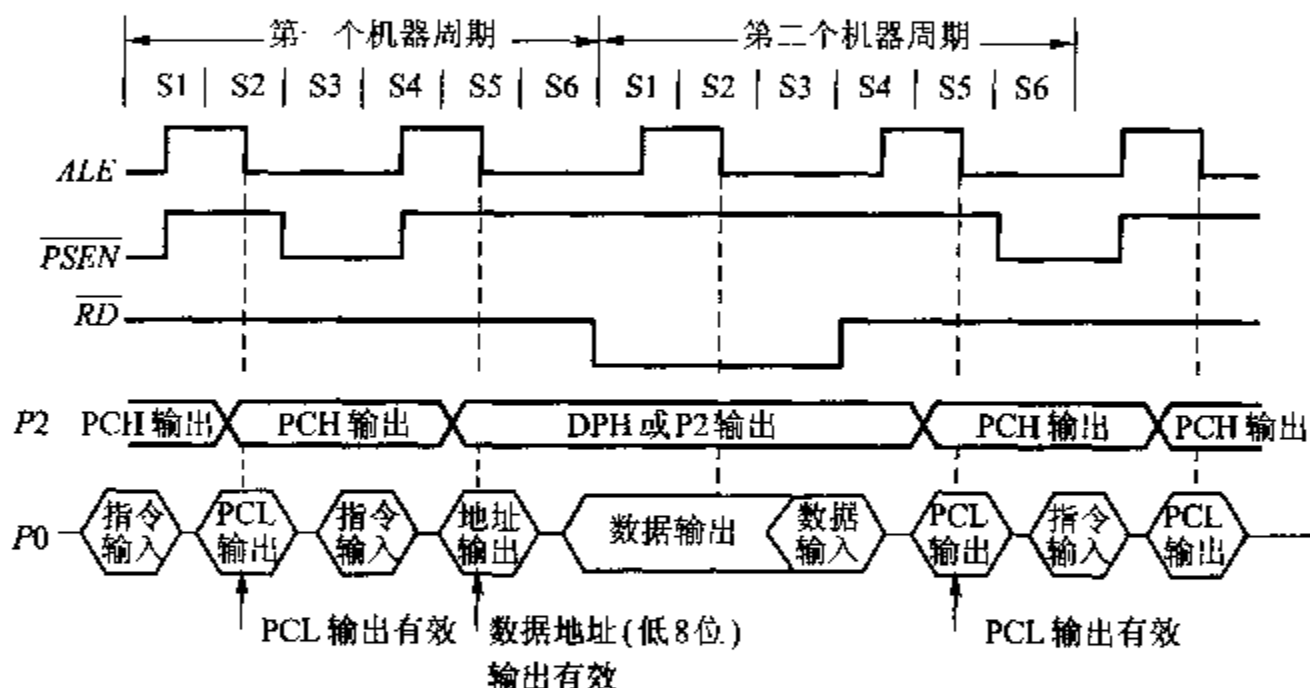
系统无片外 RAM 时,此 ALE 有效信号以振荡器频率的 1/6 出现在引脚上,它可以用作外部时钟或是定时脉冲信号。

(2) 应用系统中接有片外 RAM

在执行访问片外 RAM 的 MOVX 指令时,程序存储器的操作时序有所变化。其主要原因在于,执行 MOVX 指令时,16 位地址应转而指向数据存储器,操作时序如图 8-15(b)所示。在指令输入以前,P2 口输出的地址 PCH、PCL 指向程序存储器;在指令输入并判定是 MOVX 指令后, ALE 在该机器周期 S5 状态锁存的 P0 口的地址不是程序存储器的低 8 位,而是数据存储器的地址。若执行的是“MOVX A,@DPTR”或是“MOVX @DPTR,A”指令,则此地址就是 DPL(数据指



(a) 执行非 MOVX 指令的时序



(b) 执行 MOVX 指令的时序

图 8-15 外部程序存储器的操作时序

针低 8 位);同时,在 P2 口上出现的是 DPH(数据指针的高 8 位)。若执行的是“MOVX A,@Ri”或“MOVX @Ri,A”指令,则 Ri 的内容为低 8 位地址,而 P2 口线上将是 P2 口锁存器的内容。在同一机器周期中将不再出现 PSEN 有效取指信号,下一个机器周期中 ALE 有效锁存信号也不再出现;而当 RD/WR 信号有效时, P0 口将读/写数据存储器中的数据。由图 8-15(b)可以看出:

(1) 将 ALE 用作定时脉冲输出时,执行一次 MOVX 指令就会丢失一个脉冲。

(2) 只有在执行 MOVX 指令时的第二个机器周期期间,地址总线才由数据存储器使用。

8.4.3 典型的 EPROM 接口电路

1. 使用单片 EPROM 的扩展电路

2716、2732 EPROM 价格贵,容量小,且难以买到。在电路设计中一般不选用这两种芯片。因此,这里仅介绍 2764、27128、27256、27512 芯片与 8031 单片机的接口电路。

由于 2764 与 27128 引脚的差别仅在引脚 26 上,2764 的引脚 26 是空引脚,27128 的 26 引脚是地址线 A13,因此在设计外扩存储器电路时,应选用 27128 芯片设计电路。在实际应用时,可将 27128 换成 2764,系统仍能正常运行。反之,则不然。图 8-16 给出了 MCS-51 外扩 16 KB 的 EPROM 27128 的电路图。图中与 MCS-51 无关的电路部分均未画出。图 8-17 给出了 MCS-51 外扩 32 KB 的 EPROM 27256 的线路图。

注意在图 8-16、图 8-17 中的两种地址锁存器的用法。对于图 8-16、图 8-17 中程序存储器所占的地址空间,读者可以自己分析。

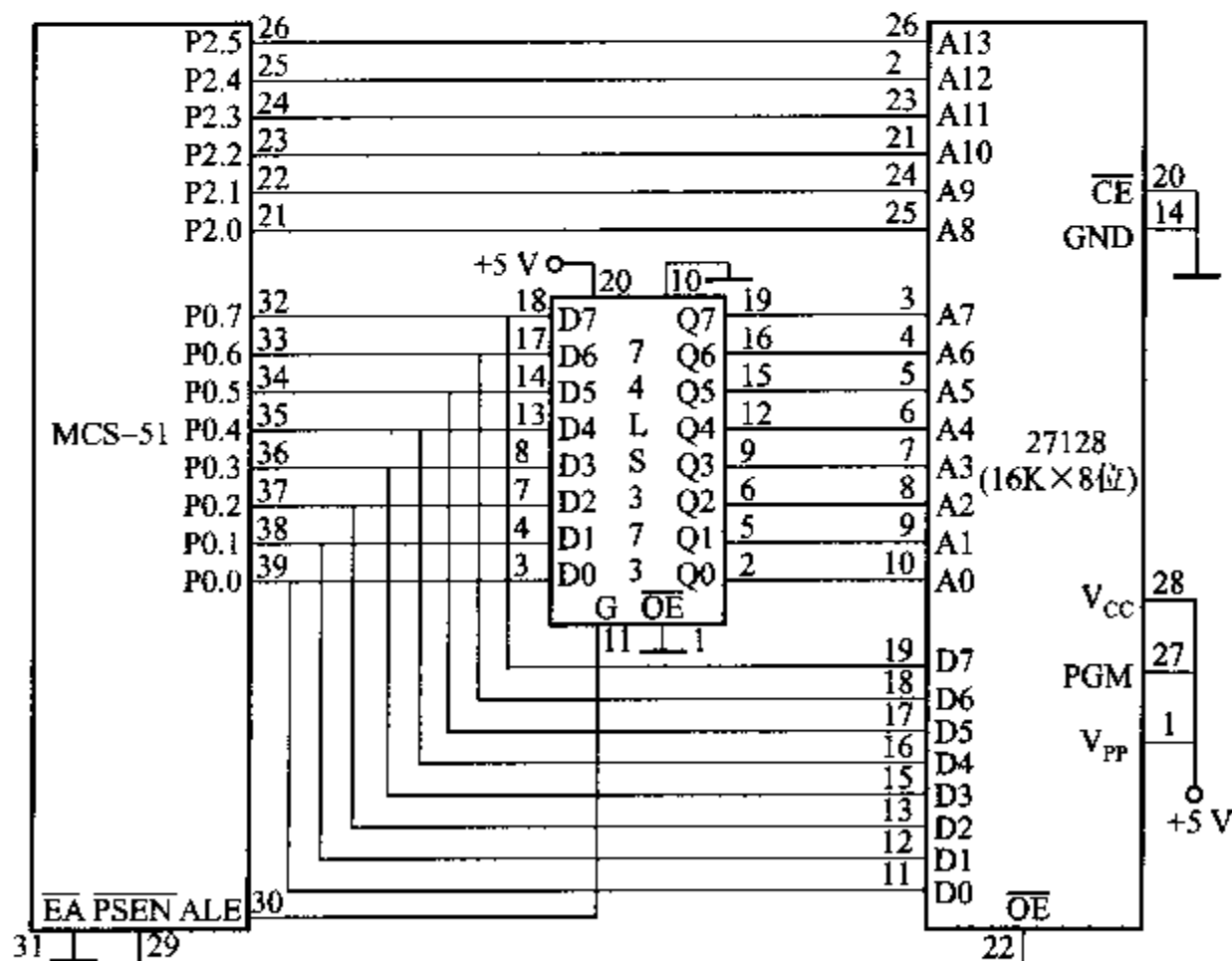


图 8-16 MCS-51 与 27128 的接口电路

2. 使用多片 EPROM 的扩展电路

与单片 EPROM 扩展电路相比,多片 EPROM 的扩展除片选线 \overline{CE} 外,其它均

与单片扩展电路相同。图 8-18 给出了利用 4 片 27128 EPROM 扩展成 61 KB 程序存储器的方法。片选信号由译码选通产生。4 片 27128 各自所占的地址空间,请读者自己分析。

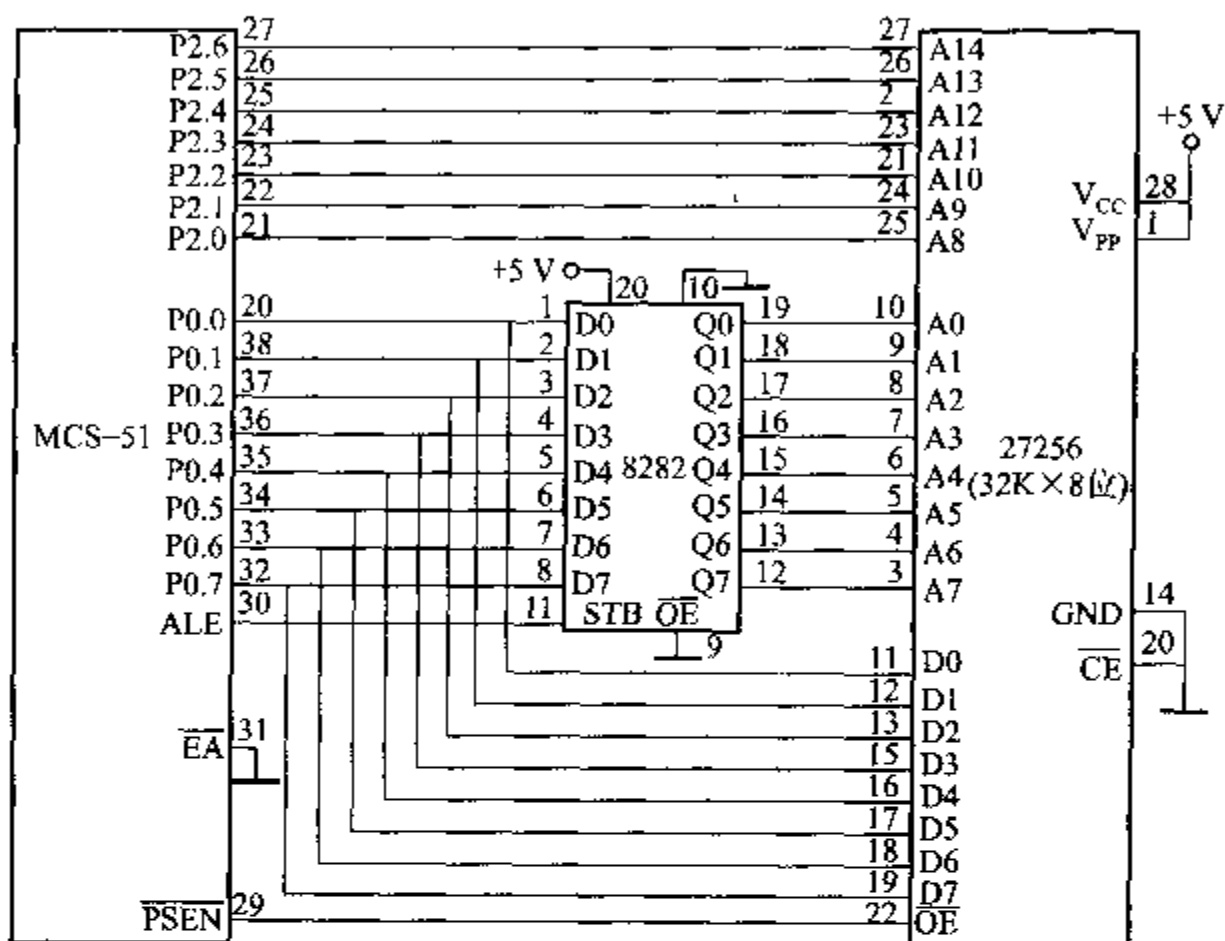


图 8-17 MCS-51 与 27256 的接口电路

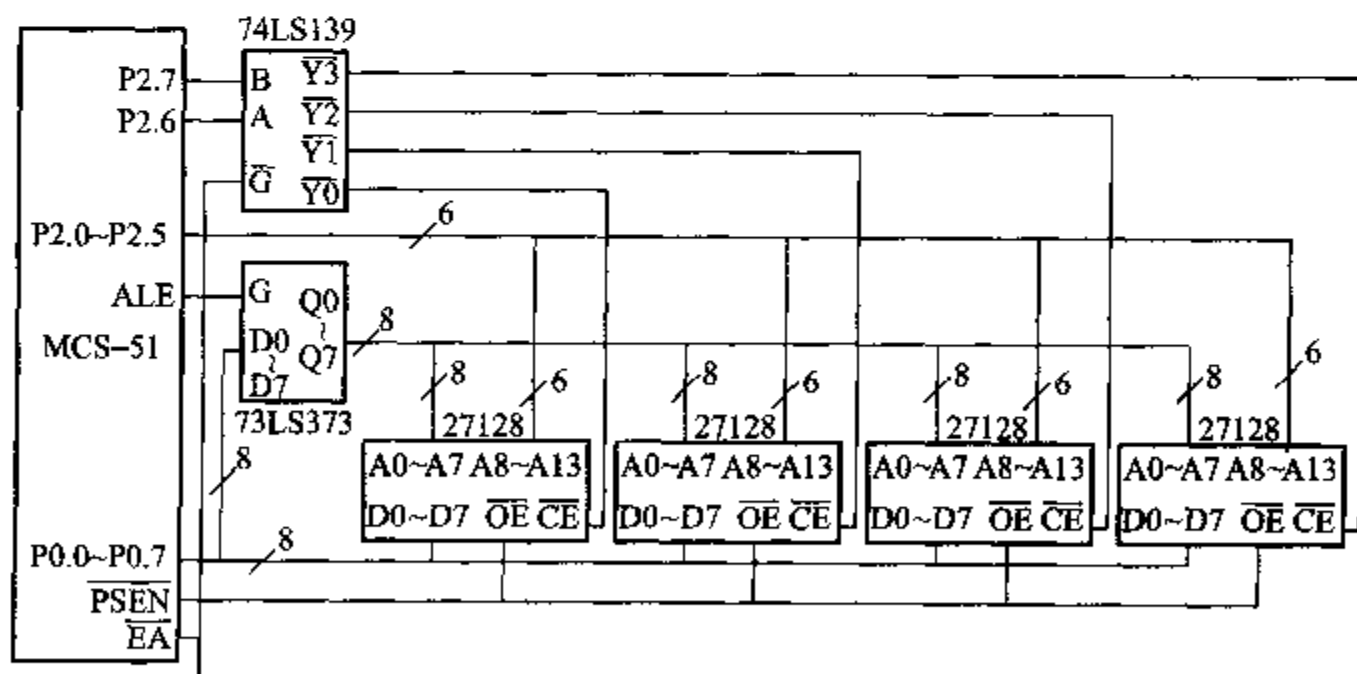


图 8-18 4 片 27128 与 MCS-51 的接口电路

8.5 静态数据存储器的扩展

MCS-51 单片机内部有 128 B RAM。在实际应用中,仅靠片内 RAM 往往不够用,必须扩展外部数据存储器。常用的数据存储器 RAM 器件有两类,即静态 RAM(SRAM)和动态 RAM(DRAM)。在单片机应用系统中,外扩的数据存储器都采用静态数据存储器,所以这里仅讨论静态数据存储器 SRAM 与 MCS-51 的接口。

所扩展的数据存储器空间地址由 P2 口提供高 8 位地址,P0 口分时提供低 8 位地址和 8 位双向数据总线。片外数据存储器 RAM 的读和写由 8031 的 \overline{RD} (P3.7)和 \overline{WR} (P3.6)信号控制,而片外程序存储器 EPROM 的输出允许端(\overline{OE})由读选通 \overline{PSEN} 信号控制。尽管与 EPROM 共处同一地址空间,但由于控制信号不同,故不会发生总线冲突。

8.5.1 常用的静态 RAM(SRAM)芯片

单片机系统中常用的 SRAM 芯片的典型型号有:6116(2K×8),6264(8K×8),62128(16K×8),62256(32K×8)。它们都用单一+5 V 电源供电,双列直插封装,6116 为 24 引脚封装,6264、62128、62256 为 28 引脚封装。这些 SRAM 的引脚图如图 8-19 所示。

SRAM 的各引脚功能如下:

A0~A14:地址输入线。

D0~D7:双向三态数据线。

\overline{CE} :片选信号输入线,低电平有效。对于 6264 芯片,当引脚 26(CS)为高电平时,且 \overline{CE} 为低电平时才选中该片。

\overline{OE} :读选通信号输入线,低电平有效。

\overline{WE} :写允许信号输入线,低电平有效。

V_{CC}:工作电源+5 V。

GND:地线。

静态 SRAM 存储器有读出、写入、维持三种工作方式,这些工作方式的操作控制如表 8-6 所示。

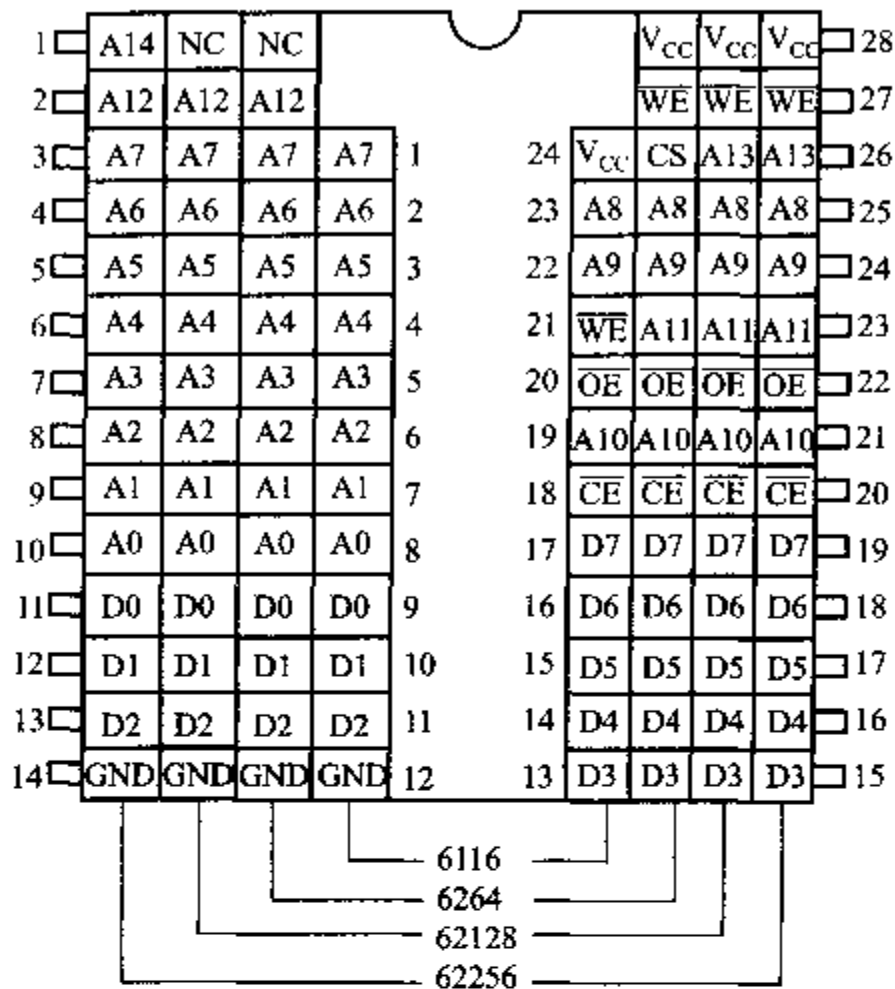


图 8-19 常用 SRAM 的引脚图

表 8-5 6116、6264、62256 的操作方式

方式 \ 信号	\overline{CE}	\overline{OE}	\overline{WE}	D0~D7
读	V_{IL}	V_{IL}	V_{IH}	数据输出
写	V_{IL}	V_{IH}	V_{IL}	数据输入
维持 *	V_{IH}	任意	任意	高阻态

* 对于 CMOS 的静态 RAM 电路, \overline{CE} 端为高电平时, 电路处于降耗状态。此时, V_{CC} 电压可降至 3 V 左右, 内部所存储的数据也不会丢失。

几种 RAM 芯片的主要技术特性见表 8-7。

表 8-7 常用静态 RAM 主要技术特性

规格特性 \ 型号	6116	6264	62256
容量/B	2K	8K	32K
引脚数/只	24	28	28
工作电压/V	5	5	5
典型工作电流/mA	35	40	8
典型维持电流/mA	5	2	0.5
存取时间	由产品型号而定, 例如, 6264-10 为 100 ns, 6264-12 为 120 ns, 6264-15 为 150 ns		

8.5.2 外扩数据存储器的读写操作时序

MCS-51 对外扩 RAM 读和写两种操作时序的基本过程是相同的。所用的控制信号有 ALE 、 \overline{RD} (读) 和 \overline{WR} (写)。

1. 读片外 RAM 操作时序

8031 单片机若外扩一片 RAM, 应将其 \overline{WR} 引脚与 RAM 芯片的 \overline{WE} 引脚连接, \overline{RD} 引脚与芯片 \overline{OE} 引脚连接。 ALE 信号的作用与 8031 外扩 EPROM 作用相同, 即锁存低 8 位地址。

读片外 RAM 周期时序如图 8-20(a) 所示。

在第一个机器周期的 S1 状态, ALE 信号由低变高①, 读 RAM 周期开始。在 S2 状态, CPU 把低 8 位地址送到 P0 口总线上, 把高 8 位地址送上 P2 口 (在执行 “MOVX A, @DPTR” 指令阶段时才送高 8 位; 若是 “MOVX A, @Ri” 则不送高 8 位)。

ALE 的下降沿②用来把低 8 位地址信息锁存到外部锁存器 74LS373 内③。而高 8 位地址信息一直锁存在 P2 口锁存器中。

在 S3 状态, P0 口总线变成高阻悬浮状态④。在 S4 状态, \overline{RD} 信号变为有效⑤ (是在执行 “MOVX A, @DPTR” 后使 \overline{RD} 信号有效), \overline{RD} 信号使得被寻址的片外 RAM 略过片刻后把数据送上 P0 口总线⑥, 当 \overline{RD} 回到高电平后⑦, P0 总线变为悬浮状态。至此, 读片外 RAM 周期结束。

2. 写片外 RAM 操作时序

向片外 RAM 写(存)数据, 是 8031 执行 “MOVX @DPTR, A” 指令后产生的动作。这条指令执行后, 8031 的 \overline{WR} 信号有效, 此信号使 RAM 的 \overline{WE} 端被选通。

写片外 RAM 的时序如图 8-20(b) 所示。开始的过程与读过程类似, 但写的过程是 CPU 主动把数据送上 P0 口总线, 故在时序上, CPU 先向 P0 口总线上送完 8 位地址, 然后在 S3 状态就将数据送到 P0 口总线③。此间, P0 口总线上不会出现高阻悬浮现象。

在 S4 状态, 写控制信号 \overline{WR} 有效, 选通片外 RAM, 稍过片刻, P0 口上的数据就写到 RAM 内了。

8.5.3 典型的外扩数据存储器的接口电路

扩展数据存储器空间地址同外扩程序存储器一样, 由 P2 口提供高 8 位地址, P0 口分时提供低 8 位地址和 8 位双向数据总线。片外 SRAM 的读和写由 8031 的 \overline{RD} (P3.7) 和 \overline{WR} (P3.6) 信号控制, 片选端 \overline{CE} 由地址译码器的译码输出控制。因此, SRAM 在与单片机连接时, 主要解决地址分配、数据线和控制信号线的连

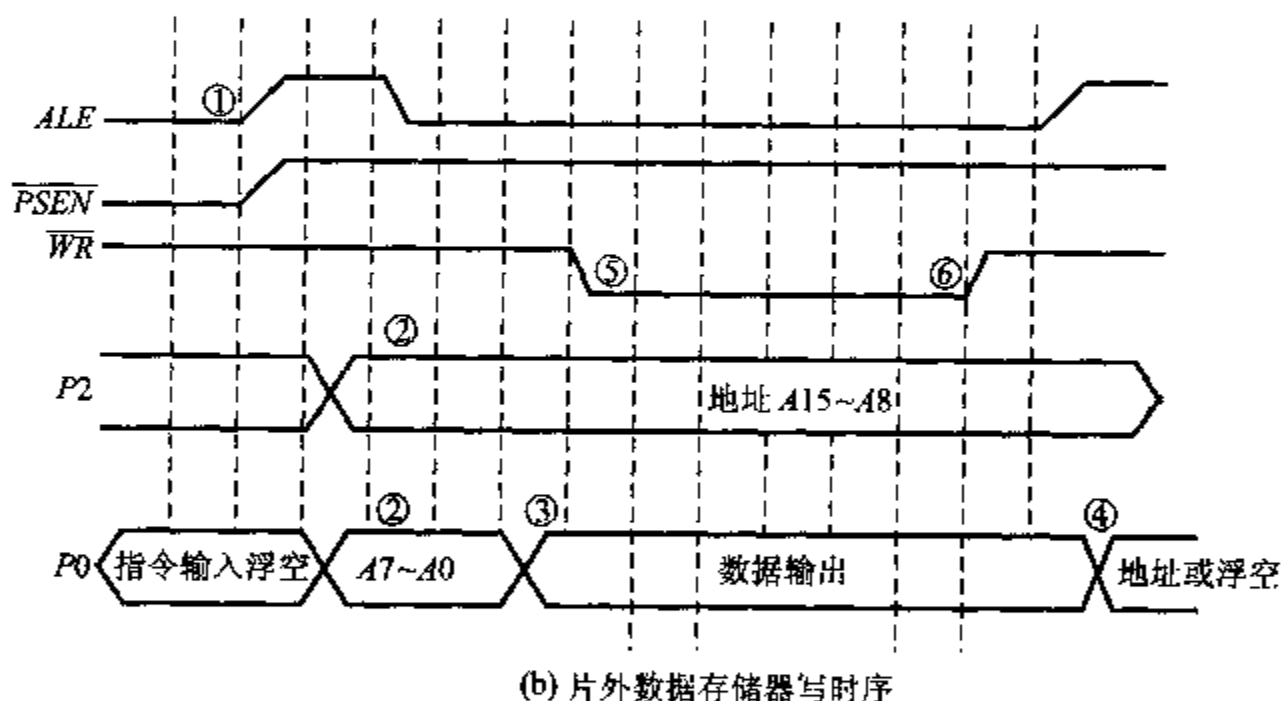
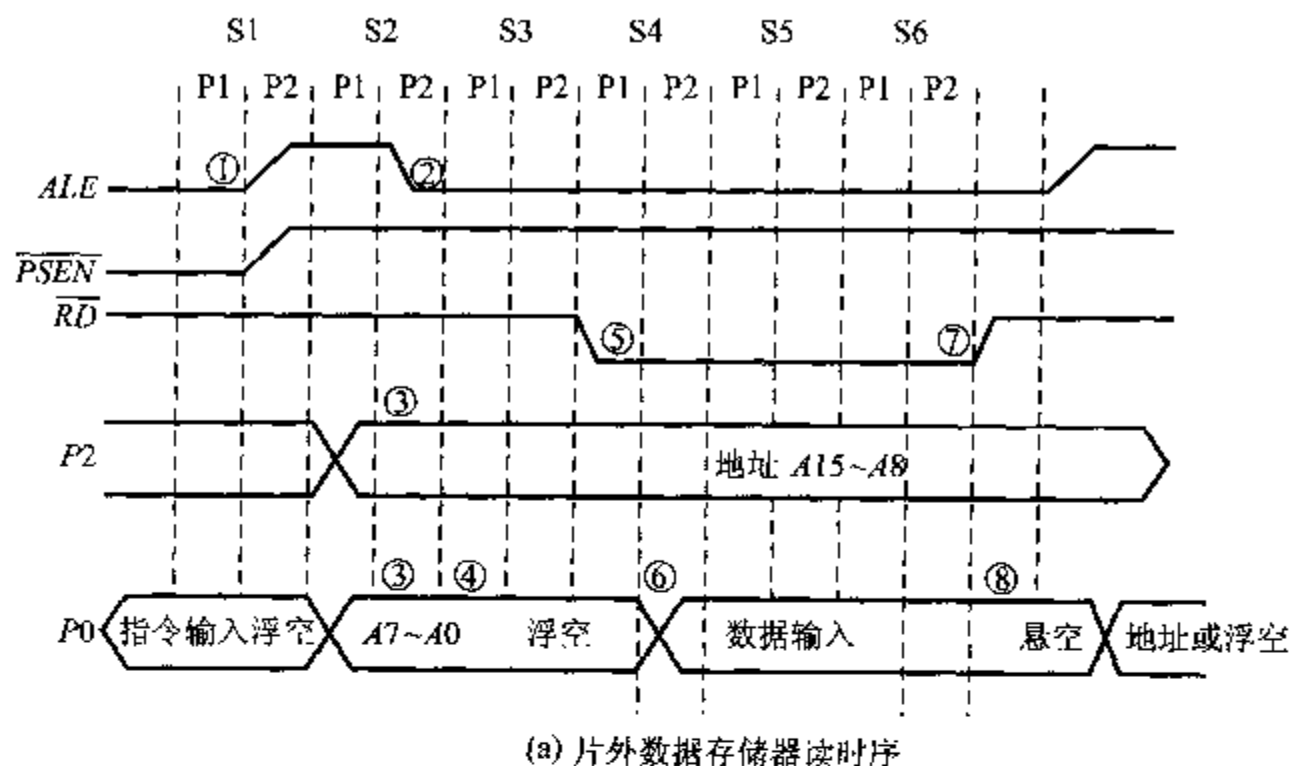


图 8-20 8031 访问片外 RAM 操作时序图

接。在与高速单片机连接时,还要根据时序解决速度匹配问题。

图 8-21 给出了用线选法扩展 8031 外部数据存储器的电路。图中数据存储器选用 6264, 该片地址线为 A0~A12, 故 8031 剩余地址线为 3 根。用线选法可扩展 3 片 6264。3 片 6264 对应的存储器空间见表 8-8 所示。

表 8-8 图 8-21 中 3 片 6264 对应的存储空间表

P2.7	P2.6	P2.5	选中芯片	地址范围	存储容量
1	1	0	IC1	C000H~DFFFH	8 KB
1	0	1	IC2	A000H~BFFFH	8 KB
0	1	1	IC3	6000H~7FFFH	8 KB

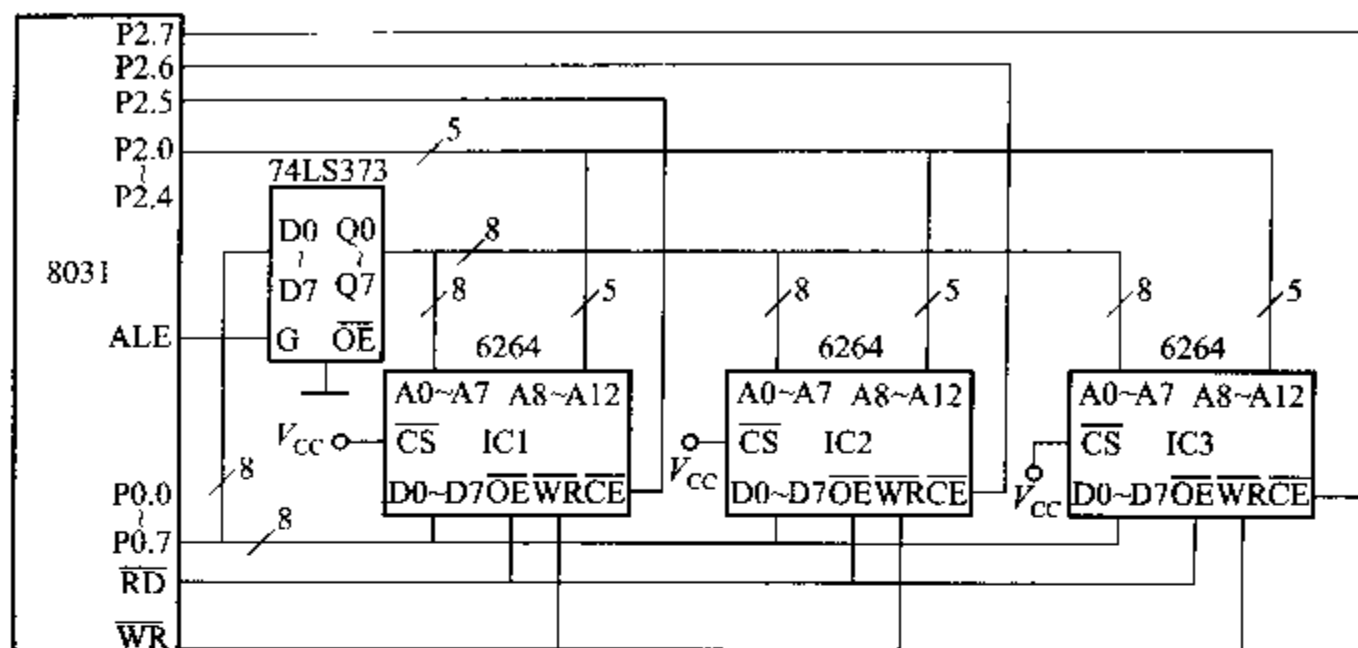


图 8-21 线选法扩展 3 片 6264 电路图

用译码选通法扩展 8031 的外部数据存储器电路如图 8-22 所示。图中数据存储器选用 62128, 该芯片地址线为 A0~A13, 这样, 8031 剩余地址线为 2 根, 若采用 2 线-4 线译码器可扩展 4 片 62128。各片 62128 地址分配见表 8-9。

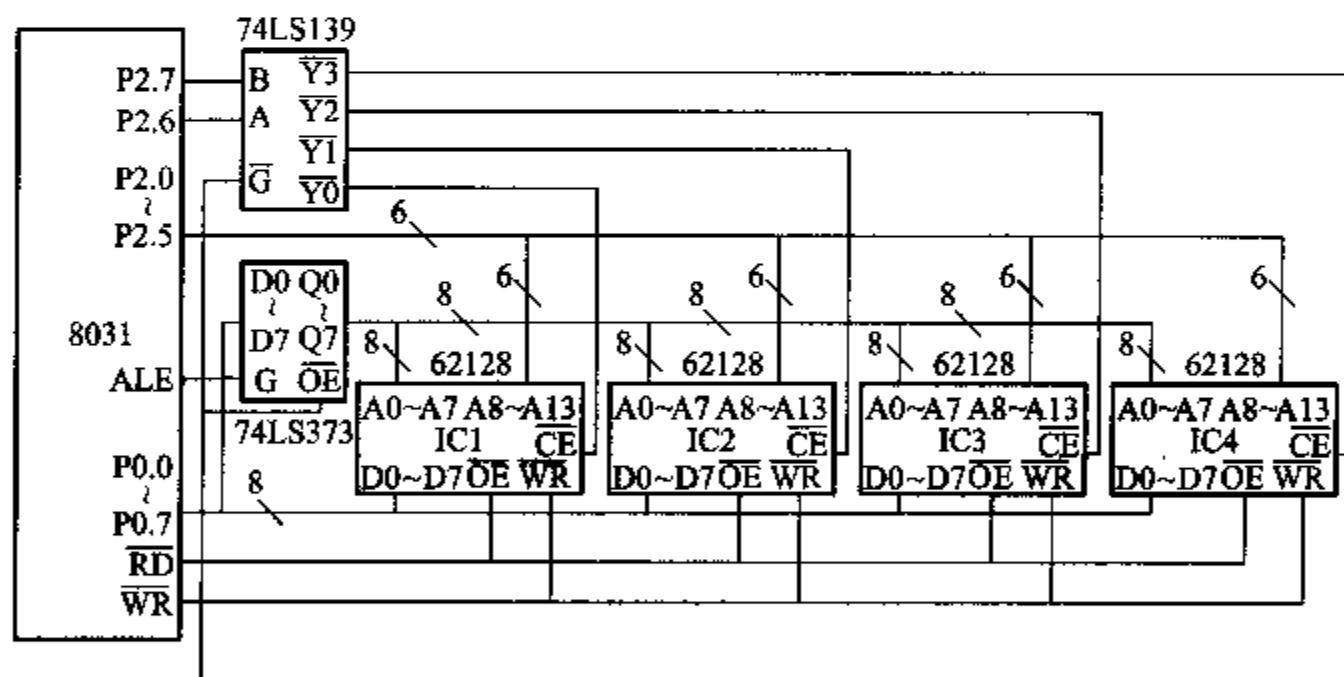


图 8-22 译码选通法扩展 8031 外部数据存储器电路图

表 8-9 各 62128 的地址空间分配

2 线-4 线译码器输入 P2.7 P2.6	2 线-4 线译码器有效输出	选中芯片	地址范围	存储容量
0 0	$\overline{Y_0}$	IC1	0000H~3FFFH	16 KB
0 1	$\overline{Y_1}$	IC2	4000H~7FFFH	16 KB
1 0	$\overline{Y_2}$	IC3	8000H~BFFFH	16 KB
1 1	$\overline{Y_3}$	IC4	C000H~FFFFH	16 KB

8.6 EPROM 和 RAM 的综合扩展

在单片机应用系统设计中,经常是既要扩展程序存储器(EPROM)又要扩展数据存储器(RAM),即存储器的综合扩展。下面通过实例来介绍如何进行综合扩展。

8.6.1 综合扩展的硬件接口电路

例 8-2 采用线选法扩展 2 片 8 KB 的 RAM 和 2 片 8 KB 的 EPROM。RAM 芯片选用 2 片 6264, EPROM 芯片选用 2 片 2764, 共扩展 4 片存储器芯片。扩展接口电路见图 8-24。

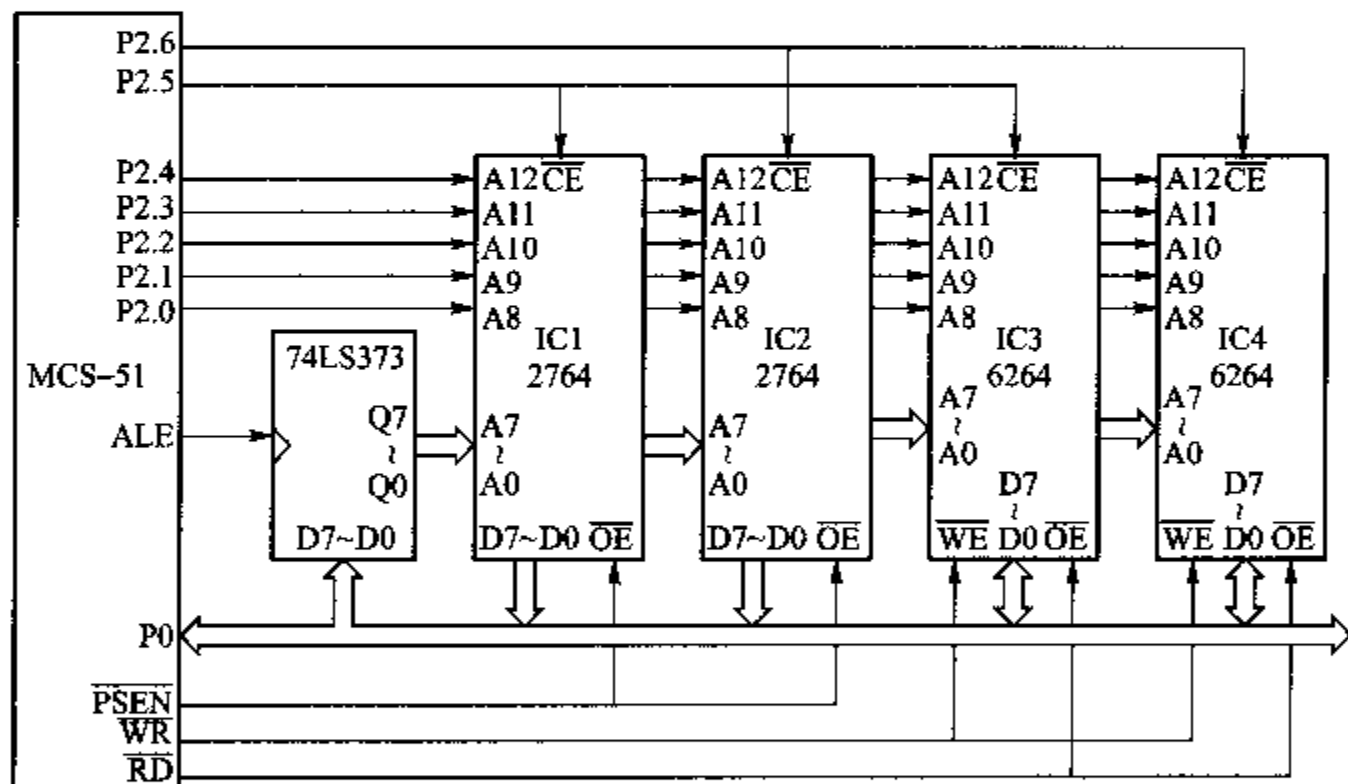


图 8-24 EPROM 和 RAM 的综合扩展(线选法)

(1) 控制信号及片选信号

地址线 P2.5 直接接到 IC1(2764)和 IC3(6264)的片选 \overline{CE} 端, P2.6 直接接到 IC2(2764)和 IC4(6264)的片选 \overline{CE} 端。当 P2.6=0, P2.5=1, IC2 和 IC4 的片选端 \overline{CE} 为低电平, IC1 和 IC3 的 \overline{CE} 端全为高电平。当 P2.6=1, P2.5=0 时, IC1 和 IC3 的 \overline{CE} 端都是低电平, 每次同时选中 2 个芯片, 具体哪个芯片工作还要通过 \overline{PSEN} 、 \overline{WR} 、 \overline{RD} 控制信号控制。当片外程序存储区读选信号 \overline{PSEN} 为低电平, 肯定到 EPROM 中读程序; 当读、写选信号 \overline{RD} 或 \overline{WR} 为低电平则到 RAM 中读数据或向 RAM 写入数据。 \overline{PSEN} 、 \overline{WR} 、 \overline{RD} 3 个信号是在执行指令时产生的, 任一

时刻,只能执行 1 条指令,所以只能 1 个信号有效,不可能同时有效。

(2) 各芯片地址空间分配

硬件电路一旦确定,各芯片的地址范围实际就已经确定,编程时只要给出要选择芯片的地址,就能准确的选中该芯片。结合图 8-24,介绍 IC1、IC2、IC3、IC4 地址范围的确定方法。

程序和数据存储器地址均用 16 位,P0 口确定低 8 位,P2 口确定高 8 位。

如 $P2.6=0$ 、 $P2.5=1$,选中 IC2、IC1,地址线 A15~A0 与 P0、P2 对应关系如下:

P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
空	0	×	×	×	×	×	×	×	×	×	×	×	×	×	×

显然除 P2.6、P2.5 位固定外,其他“×”位均可变。设无用位 $P2.7=0$,”×”各位全为 0 则为最小地址 2000H;若“×”各位均变为 1 则为最大地址 3FFFH,所以 IC2 和 IC4 占用地址空间为 2000H~3FFFH 共 8 KB。同理 IC1、IC3 地址范围 4000H~5FFFH($P2.6=1$ 、 $P2.5=0$ 、 $P2.7=0$),IC2 与 IC4 占用相同的地址空间,由于二者 1 个为程序存储器,1 个为数据存储器,3 条控制线 \overline{PSEN} 、 \overline{WR} 、 \overline{RD} 只能有 1 个有效。因此,地址空间重叠也无关系,IC1 与 IC3 也同样。

从此例看出,线选法地址不连续,地址空间利用不充分。

例 8-3 采用译码器法扩展 2 片 8 KB EPROM,2 片 8 KB RAM。EPROM 选用 2764,RAM 选用 6264。共扩展 4 片芯片。扩展接口电路见图 8-25。图中 74LS139 的 4 个输出端 $\overline{Y0} \sim \overline{Y3}$ 分别连接 4 个芯片 IC1、IC2、IC3、IC4 的片选端。 $\overline{Y0} \sim \overline{Y3}$ 输出信号每次只能有 1 位是 0,其他 3 位全为 1,输出为 0 的一端所连接的芯片被选中。

译码法地址分配,首先要根据译码芯片真值表确定译码芯片的输入状态,由此再判断其输出端选中芯片的地址。

如图 8-25 所示,74LS139 的输入端 A、B、 \overline{C} 分别接 P2 口的 P2.5、P2.6、P2.7 3 端, \overline{C} 为使能端,低电平有效。根据表 8-2 中 74LS139 的真值表可见,当 $\overline{C}=0$ 、 $A=0$ 、 $B=0$ 时,输出端只有 $\overline{Y0}$ 为 0, $\overline{Y1} \sim \overline{Y3}$ 全为 1,选中 IC1。这样,引脚 P2.7、P2.6、P2.5 输出全为 0,其他地址线任意状态都能选中 IC1。当其他位全为 0 最小地址为 0000H,其他位全为 1 最大地址为 1FFFH。所以 IC1 地址范围 0000H~1FFFH。同理可确定电路中各个存储器的地址范围如下:

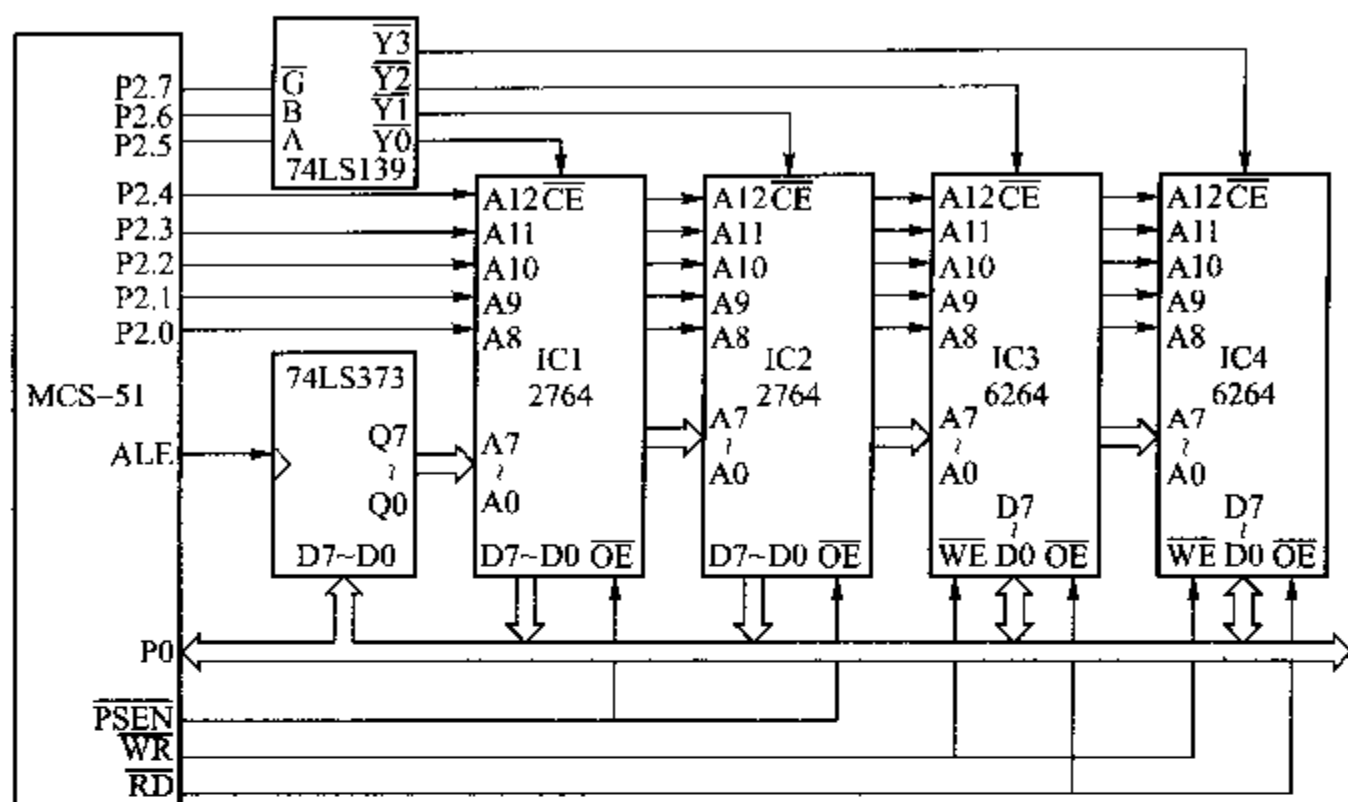


图 8-25 采用译码法的综合扩展电路

芯 片	地 址 范 围
IC4	6000H~7FFFH
IC3	4000H~5FFFH
IC2	2000H~3FFFH
IC1	0000H~1FFFH

由上可见，译码法进行地址分配，各芯片的地址空间是连续的。

8.6.2 外扩存储器电路的工作原理及软件设计

为了使读者弄清楚单片机与扩展的存储器软、硬件之间的关系，结合图 8-25 所示译码电路，说明片外读指令和从片外读、写数据的过程。

1. 单片机片外程序区读指令过程

当一接通电源，单片机上电复位。复位后程序计数器 $PC=0000H$ ， PC 是程序指针，它总是指向将要执行的程序地址。CPU 就从 $0000H$ 地址开始取指令，执行程序。在取指令期间， PC 地址低 8 位送往 $P0$ 口，经锁存器锁存到 $A0\sim A7$ 地址线上。 PC 高 8 位地址送往 $P2$ 口，直接由 $P2.0\sim P2.4$ 锁存到 $A8\sim A12$ 地址线上， $P2.5\sim P2.7$ 输入给 74LS139 进行片选。这样，根据 $P2$ 、 $P0$ 口状态则选中了第一个程序存储器芯片 IC1(2764)的第一个地址 $0000H$ 。然后当 \overline{PSEN} 端变为低电平时，把 $0000H$ 中的指令代码经 $P0$ 口读入内部 RAM 中，进行译码从而决定进行何种操作。取出一个指令字节后 PC 自动加 1，然后取第二个字节，依次类推。当

PC=1FFFH 时,从 IC1 最后一个单元取指令,然后 PC=2000H,CPU 向 P0、P2 送出 2000H 地址时则选中第二个程序存储器 IC2,IC2 的地址范围 2000H~3FFFH,读指令过程同 IC1,不再赘述。

2. 单片机片外数据区读写数据过程

当执行程序中,遇到“MOV”类指令时,表示与片内 RAM 交换数据;当遇到“MOVX”类指令时,表示对片外数据区寻址。片外数据区只能间接寻址。

例如,把片外 1000H 单元的数据送到片内 RAM 50H 单元中,程序如下:

```
MOV      DPTR, #1000H
MOVX     A, @DPTR
MOV      50H, A
```

先把寻址地址 1000H 送到数据指针寄存器 DPTR 中,当执行 MOVX A, @DPTR 时,DPTR 的低 8 位(00H)经 P0 口输出并锁存,高 8 位(50H)经 P2 直接输出,根据 P0、P2 状态选中 IC3(6264)的 1000H 单元。当读选通信号 \overline{RD} 为低电平时,片外 1000H 单元的数据经 P0 口送往 A 累加器。当执行指令 MOV 50H, A 时则把该数据存入片内 50H 单元。

向片外数据区写数据的过程与读数据的过程类似。

例如,把片内 50H 单元的数据送到片外 1000H 单元中,程序如下:

```
MOV      A, 50H
MOV      DPTR, #1000H
MOVX     @DPTR, A
```

先把片内 RAM 50H 单元的数据送到 A 中,再把寻址地址 1000H 送到数据指针寄存器 DPTR 中,当执行 MOVX @DPTR, A 时,DPTR 的低 8 位(00H)由 P0 口输出并锁存,高 8 位(50H)由 P2 口直接输出,根据 P0、P2 状态选中 IC3(6264)的 1000H 单元。当写选通信号 \overline{WR} 为低电平时, A 中的内容送往片外 1000H 单元中。

MCS-51 单片机读写片外数据存储器中的内容,除了使用 MOVX A, @DPTR 和 MOVX @DPTR, A 外,还可以使用 MOVX A, @Ri 和 MOVX @Ri, A。这时通过 P0 口接收 Ri 中的内容(低 8 位地址),而把 P2 口原有的内容作为高 8 位地址输出。下面介绍的例 8-4 即是采用 MOVX @Ri, A 指令的例子。

例 8-4 编写程序,将程序存储器中以 TAB 为首址的 32 个单元的内容依次传送到外部 RAM 以 7000H 为首地址的区域去。

数据指针 DPTR 指向标号 TAB 的首地址。R0 既指示外部 RAM 的地址,又表示数据标号 TAB 的位移量。此程序为一循环程序,循环次数为 32, R0 的值从 0~31, R0 的值达到 32 就结束循环。程序如下:

```
MOV      P2, #10
```

```

MOV     DPTR, # TAB
MOV     R0, # 0
AGIN:   MOV     A, R0
        MOVC    A, (@ A + DPTR)
        MOVX    @ R0, A
        INC     R0
        CJNE    R0, # 32, AGIN
HERE:   SJMP    HERE
TAB:     DB      ..., ...
    
```

8.7 E²PROM 的扩展

E²PROM 是电可擦除可编程只读存储器,其突出优点是能够在线擦除和改写,无须像 EPROM 那样必须用紫外线照射才能擦除。较新的 E²PROM 产品在写入时能自动完成擦除,且不再需要专用的编程电源,可以直接使用单片机系统的 +5 V 电源。

E²PROM 用于单片机系统中,既可以扩展为片外 EPROM,也可以扩展为片外 RAM。它使单片机系统的设计,特别是调试实验更为方便、灵活。在调试程序时,用 E²PROM 代替仿真 EPROM,既可方便地修改程序,又能保存调试好的程序。当然,与 RAM 芯片相比,E²PROM 写操作速度是很慢的。另外,它的擦除/写入是有寿命限制的,虽然有 1 万次之多,但也不宜用在数据频繁更新的场合。因此,应注意均衡使用各单元,不然有些单元可能会提前结束寿命。

E²PROM 既具有 ROM 的非易失性的优点,又能像 RAM 一样随机地读/写,每个单元保留信息的时间长达 20 年,不存在 EPROM 在日光下信息缓慢丢失的问题。

8.7.1 常用的 E²PROM 芯片

常用的 E²PROM 芯片有 2816/2816A,2817/2817A,2864A 等。这些芯片的引脚如图 8-26 所示,其主要性能如表 8-10 所示(表中芯片均为 Intel 公司产品)。

在芯片的引脚设计上,2 KB 的 E²PROM 2816 与相同容量的 EPROM 2716 和静态 RAM 6116 是兼容的,8 KB 的 E²PROM 2864A 与同容量的 EPROM 2764 和静态 RAM 6264 也是兼容的。2816、2817 和 2864A 的读出数据时间均为 250 ns,写入时间 10 ms。

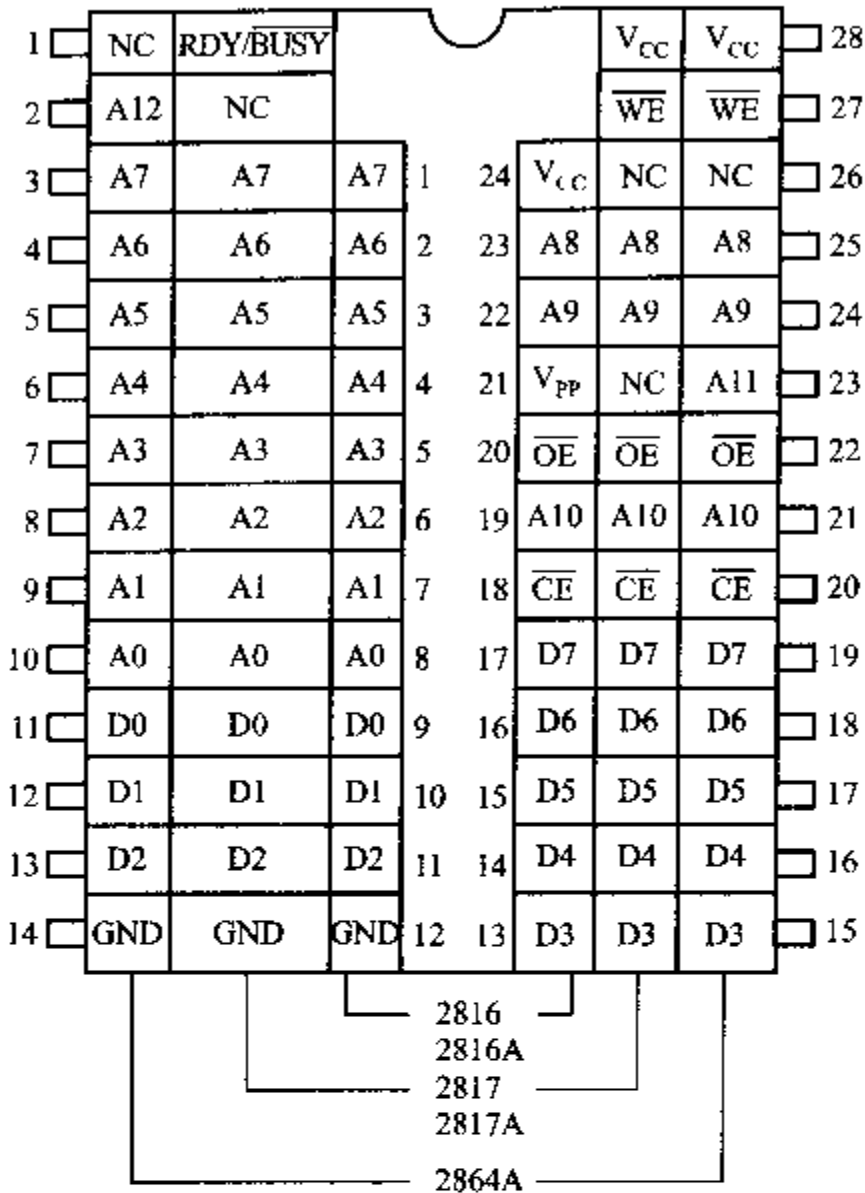


图 8-26 常用 E²PROM 引脚图

表 8-10 E²PROM 的主要性能

参 数 值 参 数	器 件 型 号	2816	2816A	2817	2817A	2864A
取数时间/ns		250	200/250	250	200/250	250
读操作电压/V		5	5	5	5	5
写/擦操作电压/V		21	5	21	5	5
字节擦除时间/ms		10	9~15	10	10	10
写入时间/ms		10	9~15	10	10	10
容量/B		2K×8	2K×8	2K×8	2K×8	8K×8
封装		DIP24	DIP24	DIP28	DIP28	DIP28
兼容		2716	2716			6264A

8.7.2 E²PROM的工作方式

表 8-11 中给出了 Intel 公司生产的常见 E²PROM 的工作方式。E²PROM 芯片中 RDY/ $\overline{\text{BUSY}}$ 为开路输出,应接上拉电阻至 +5 V。

表 8-11 E²PROM 的工作方式

选 型 号	引 脚 号	$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{WE}}$	RDY/ $\overline{\text{BUSY}}$	输入/输出
2816A	引脚号	(18)	(20)	(21)		(9~11, 13~17)
	读	V_{IL}	V_{IL}	V_{IH}		DOUT
	维持	V_{IH}	任意	任意		高阻
	字节擦除	V_{IL}	V_{IH}	V_{IL}		DIN = V_{IH}
	字节写入	V_{IL}	V_{IH}	V_{IL}		DIN
	全片擦除	V_{IL}	10~15 V	V_{IL}		DIN = V_{IH}
	不操作	V_{IL}	V_{IH}	V_{IH}		高阻
	E/W 禁止	V_{IH}	V_{IH}	V_{IL}		高阻
2817A	引脚号	(20)	(22)	(27)	(1)	(11~13, 15~19)
	读	V_{IL}	V_{IL}	V_{IH}	高阻	DOUT
	维持	V_{IH}	任意	任意	高阻	高阻
	字节写入	V_{IL}	V_{IH}	V_{IL}	V_{IL}	DIN
	字节擦除	字节写入之前自动擦除				
2864A	引脚号	(20)	(22)	(27)		(11~13, 15~19)
	待机	V_{IH}	任意	任意		高阻
	读	V_{IL}	V_{IL}	V_{IH}		DOUT
	写	V_{IL}	V_{IH}	负脉冲		DIN
	DATA 查询	V_{IL}	V_{IL}	V_{IH}		DOUT

下面对 Intel 公司产品 2817A 和 2864A 的工作方式作详细说明。

2817A 在写入 1 B 信息之前,自动地对所要写入的单元进行擦除,因而无需进行专门的字节/芯片擦除操作。当向 2817A 发出字节写入命令后,2817A 将锁存地址、数据及控制信号,从而启动一次操作。在写入期间,2817A 的 RDY/ $\overline{\text{BUSY}}$ 引脚呈低电平,此时,它的数据总线呈高阻状态,因而允许处理器在此期间执行其它的任务。一次写操作一旦结束,2817A 便将 RDY/ $\overline{\text{BUSY}}$ 线置高,此时,处理器可以对 2817A 进行新的字节读/写操作。

由表 8-11 可知,2864A 有 4 种工作方式:

1. 维持方式

当信号 \overline{CE} 为高电平时,2864A 进入低耗维持方式。此时,输出线呈高阻态,芯片的电流从 110 mA 降至维持电流 60 mA。

2. 读方式

当信号 \overline{CE} 和 \overline{OE} 均为低电平而 \overline{WE} 为高电平时,内部的数据缓冲器被打开,数据送上总线,此时,可进行读操作。

3. 写方式

2864A 提供了 2 种数据写入方式:字节写入和页写入。

(1) 页写入:为了提高写入速度,2864A 片内设置了 16 B 的页缓冲器,并将整个存储器阵列划分成 512 页,每页 16 B。页的区分可由地址的高 9 位($A_4 \sim A_{12}$)来确定,地址的低 4 位($A_0 \sim A_3$)用以选择页缓冲器中的 16 个地址单元之一。对 2864A 的写操作可分成两步来实现:第一步,在软件控制下把数据写入页缓冲器,这步称为页装载,与一般的静态 RAM 写操作是一样的。第二步,在最后一个字节(即第 16 个字节)写入到页缓冲器 20 ns 后自动开始,把页缓冲器的内容写到 E²PROM 阵列对应地址的单元中,这一步称为页存储。

写方式时, \overline{CE} 为低电平,在 \overline{WE} 下降沿,地址码 $A_0 \sim A_{12}$ 被片内锁存器锁存,在 \overline{WE} 上升沿时数据被锁存。片内还有 1 个字节装载限时定时器,只要时间未到,数据可以随机地写入页缓冲器。在连续向页缓冲器写入数据的过程中,不用担心限时定时器会溢出,因为每当 \overline{WE} 下降沿时,限时定时器自动被复位并重新启动。限时定时器要求写入 1 B 数据的操作时间 T_{BLW} 须满足: $3 \mu s < T_{BLW} < 20 \mu s$, 这样是正确完成对 2864A 页面写入操作的关键。当 1 页装载完毕,不再有 \overline{WE} 信号时,限时定时器将溢出,于是页存储操作随即自动开始。首先把选中页的内容擦除,然后写入的数据由页缓冲器传递到 E²PROM 阵列中。

(2) 字节写入:字节写入的过程与页写入的过程类似;不同之处是仅写入 1 B,限时定时器就溢出。

4. 数据查询方式

数据查询是指用软件来检测写操作中的页存储周期是否完成。

在页存储期间,如对 2864A 执行读操作,那么读出的是最后写入的字节,若芯片的存储工作未完成,则读出数据的最高位是原来写入字节最高位的反码。据此,CPU 可判断芯片的编程是否结束。如果读出的数据与写入的数据相同,表示芯片已完成编程,CPU 可继续向 2864A 装载下一页数据。

上面介绍的 E²PROM 都是针对 Intel 公司的产品,其他公司的产品不一定相同,如市场上常见的 SEEQ 公司的 2864 只有字节写入方式,没有页写入方式,也没有数据查询功能。这时,可采取延时的方法解决可靠写入 E²PROM 数据的问题。

8.7.3 MCS-51 扩展 E²PROM 的方法

1. MCS-51 外扩 2817A

2817A 与 8031 单片机的硬件连接图如图 8-27 所示。在图 8-27 中, 2817A 既可作为外部的数据存储器, 又可作为程序存储器。8031 通过 P1.0 查询 2817A 的 RDY $\overline{\text{BUSY}}$ 引脚状态, 来完成对 2817A 的写操作。2817A 的片选信号由 P2.7 提供, 在系统中有其他 ROM 和 RAM 存储器时, 需统一考虑编址问题。

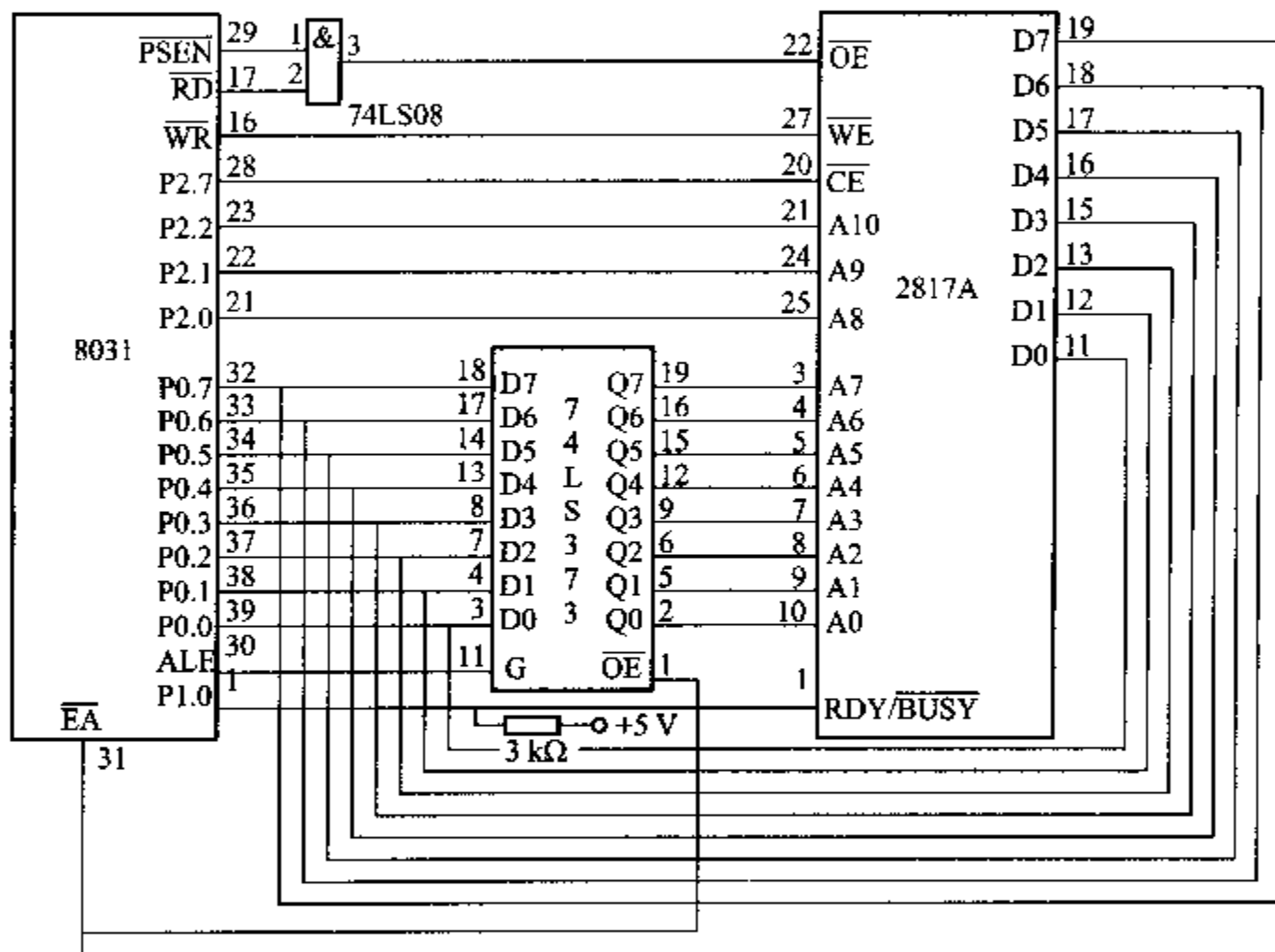


图 8-27 2817A 与 8031 接口电路图

下面给出 8031 对 2817A 进行写操作的子程序 WR1。被写入的数据取自源数据区。子程序的入口参数为：

R0=写入的字节数

R1=2817A 的低 8 位地址

R2=2817A 的高 8 位地址

R3=源数据区低 8 位首地址

R4=源数据区高 8 位首地址

WR1: MOV DPL, R3

```

MOV    DPH, R4          ;将源数据区 16 位地址传输到 DPTR 中
MOVX   A, @DPTR         ;取数据
INC    DPTR             ;源数据地址指针加 1
MOV    R3, DPL
MOV    R4, DPH          ;将新的源数据区地址保存在 R3, R4 中
MOV    DPL, R1
MOV    DPH, R2          ;将 2817A 地址传输到 DPTR 中
MOVX   @DPTR, A         ;将 A 的内容写入 2817A 中
WAIT:  JNB    P1.0, WAIT ;1 B 未写完等待
INC    DPTR             ;2817A 地址增 1
MOV    R1, DPL
MOV    R2, DPH          ;将 2817A 的地址保存在 R1, R2 中
DJNZ   R0, WR1          ;未写完,继续
RET

```

2. MCS-51 外扩 2864A

2864A 与 8031 单片机的接口电路如图 8-28 所示。2864A 的片选端 \overline{CE} 与高地址线 P2.7 连接, P2.7=0 才能选中 2864A, 这种线选法决定了 2864A 对应多组地址空间, 即: 0000H~1FFFH, 2000H~3FFFH, 4000H~5FFFH, 6000H~7FFFH。这 8 KB 存储器可作为数据存储器使用, 但掉电后数据不丢失。

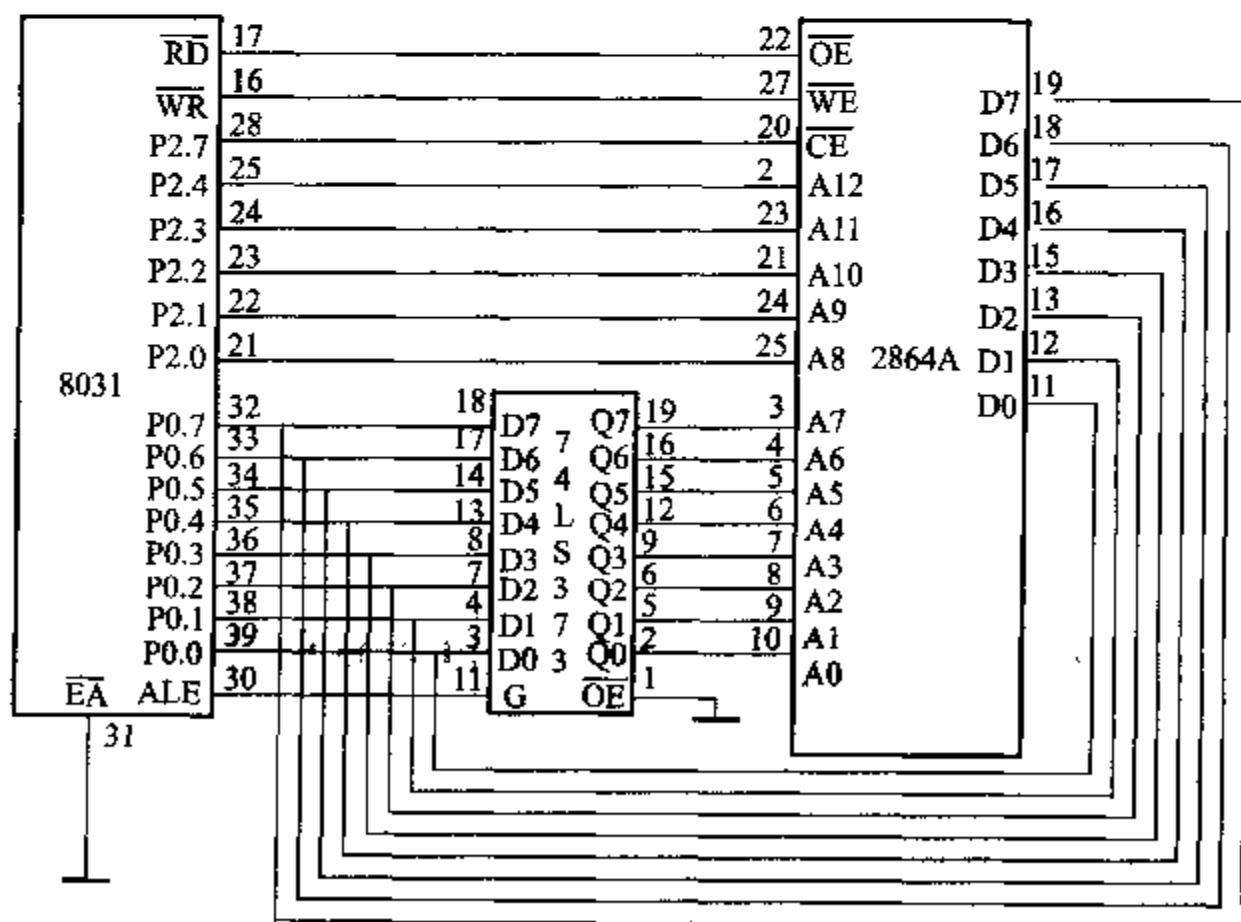


图 8-28 2864A 与 8031 接口电路图

8031 对 2864A 进行写操作时所用指令包括:

MOVX @DPTR, A

MOVX @Ri, A

8031 对 2864A 进行读操作时所用指令包括:

MOVX A, @DPTR

MOVX A, @Ri

对 2864A 装载 1 个页面数据(16B)的子程序 WR2 如下:

被写入的数据取自源数据区,子程序入口参数为:

R1—写入 2864A 的字节数(16B)

R2—2864A 的低位地址

P2—2864A 的高位地址

DPTR = 源数据区首地址

```
WR2: MOVX A, @DPTR      ;取数据
      MOV R2, A          ;数据暂存 R2, 备查询
      MOVX @R0, A        ;写入 2864A
      INC DPTR           ;源地址指针加 1
      INC R0             ;目的地址指针加 1
      CJNE R0, #00H, NEXT ;低位地址指针未满转移
      INC R2             ;否则高位指针加 1
NEXT: DJNZ R1, WR2       ;页面未装载完转移
      DEC R0             ;页面装载完后, 恢复最后写入数据的地址
LOOP: MOVX A, @R0        ;读 2864A
      XRL A, R2          ;与写入的最后一个数据相异或
      JB Acc.7, Loop     ;最高位不等, 再查
      RET               ;最高位相同, 1 页写完
```

上述写入程序中, 完成页面装载的循环部分共 8 条指令, 当采用 12 MHz 晶振时, 进行时间约 $13\mu s$, 完全符合 2864A 的 T_{BLW} 宽度要求。

8.8 AT89C51/89C55 单片机的片内闪烁存储器

AT89C51/89C52/89C55 是 1 种低功耗、高性能的片内含有 4 KB/8 KB/20 KB 闪烁可编程/擦除只读存储器 (Flash Programmable and Erasable Read Only Memory) 的 8 位 CMOS 单片机, 并且与 MCS-51 引脚和指令系统完全兼容。芯片上的 FPEROM 允许在线编程或采用通用的编程器对其重复编程。

由于片内带 EPROM 的 87C51 价格偏高, 而片内带 FPEROM 的 89C51 价

格低且与 Intel 87C51 兼容,所以,89C51 的性能价格比远高于 87C51。

8.8.1 89C51 的性能及片内闪烁存储器

1. 89C51 的主要性能包括

- (1) 与 MCS-51 微控制器系列产品兼容。
- (2) 片内有 4 KB 可在线重复编程的闪烁存储器(Flash Memory)。
- (3) 存储器可循环写入/擦除 1 万次。
- (4) 存储器数据保存时间为 10 年。
- (5) 宽工作电压范围: V_{CC} 可为 $+2.7\text{ V} \sim +6\text{ V}$ 。
- (6) 全静态工作:可从 $0\text{ Hz} \sim 16\text{ MHz}$ 。
- (7) 程序存储器具有 3 级加密保护。
- (8) 空闲状态维持低功耗和掉电状态保存存储器内容。

2. 片内闪烁存储器(Flash Memory)

由于 $E^2\text{PROM}$ 具有在线改写,并在掉电后仍能保存数据的特点,可为用户的特殊应用提供便利。但是,擦除和写入对于要有数据高速吞吐的应用还显得时间过长,这是 $E^2\text{PROM}$ 的主要缺陷。表 8-12 列出了几种典型 $E^2\text{PROM}$ 芯片的主要性能数据。

表 8-12 几种典型芯片主要性能

型 号	2816	2816A	2817	2817A	2864A
性能参数					
取数时间/ms	250	200/250	250	200/250	250
擦/写电压/V	21	5	21	5	5
字节擦除时间/ms	10	9~15	10	10	10
写入时间/ms	10	9~15	10	10	10

由表 8-12 可见,所列各种芯片的字节擦除时间和写入时间基本上均为 10 ms,这样长的时间对于许多实际应用是不能接受的。因此,将存储器集成到单片机芯片内,设法缩短此类存储器的擦除和写入时间是 1 个首要的问题。片内闪烁存储器(Flash Memory)的概念就是在这种背景下提出来的。目前,美国 ATMEL 公司生产的带有片内闪烁存储器的 AT89C51/89C52/89C55 单片机,由于价格便宜,且与 MCS-51 系列兼容,受到了我国广大工程技术人员欢迎,使用该系列单片机,省去了外扩程序存储器的工作,只需了解片内闪烁存储器的特性以及如何对其编程即可。

下面介绍 89C51 片内闪烁存储器的主要性能及其编程使用。

8.8.2 片内闪烁存储器的编程

89C51 的片内程序存储器由 FPEROM 取代了 87C51 的 EPROM 外,其余部分完全相同。89C51 的引脚与 87C51 的引脚也是完全兼容的。

89C51 的 I/O 口 P0、P1、P2 和 P3 除具有与 MCS-51 相同的一些性能和用途外,在 FPEROM 编程时,P0 口还可接收代码字节,但在程序校验时需要外加上拉负载电阻。在 FPEROM 编程和程序校验期间,P1 口接收低地址字节,P2 口接收高位地址位和一些控制信号,P3 口也接收 FPEROM 编程和校验用的控制信号。此时,ALE/PROG 引脚是编程脉冲输入($\overline{\text{PROG}}$)端。

该芯片内有 3 个加密位,其状态可以为编程(P)或不编程(U),各状态提供的功能见表 8-13。如果加密位 LB1 被编程,则 $\overline{\text{EA}}$ 引脚的电平在复位时被采样并锁存。若器件在加电时不进行复位,那么该锁存器初始化为一个随机值,并在复位有效前始终保持该值。为使器件工作正常, $\overline{\text{EA}}$ 的锁存值必须与引脚的当前逻辑电平一致。

89C51 的 3 个加密位可以不被编程(U)或被编程(P),以获得表 8-13 所示的特性。

表 8-13 加密位保护模式

类型	程序加密位			保护功能
	LB1	LB2	LB3	
1	U	U	U	无程序加密特性
2	P	U	U	可对外部程序存储器执行 MOVC 指令,不允许从内部存储器取代码字节。在复位脉冲期间, $\overline{\text{EA}}$ 被采样并锁存。禁止 FPEROM 的进一步编程
3	P	P	U	与类型 2 相同,同时进行校验
4	P	P	P	与类型 3 相同,同时外部执行被禁止。

对 89C51 片内的闪烁存储器编程,只需在市场上购买相应的编程器,按照编程器的说明进行操作。如想对写入的内容加密,只需按照编程器的菜单,选择加密功能选项即可。

思考题及习题

1. 单片机存储器的主要功能是存储()和()。
2. 试编写一个程序(例如将 05H 和 06H 拼为 56H),设原始数据放在片外数据区 2001H 单元和 2002H 单元中,按顺序拼装后的单字节数放入 2002H。

3. 假设外部数据存储器 2000H 单元的内容为 80H, 执行下列指令后:

MOV P2, #20H

MOV R0, #00H

MOVX A, @R0

累加器 A 中的内容为()。

4. 编写程序, 将外部数据存储器中的 4000H~40FFH 单元全部清零。

5. 在 MCS-51 单片机系统中, 外接程序存储器和数据存储器共 16 位地址线和 8 位数据线, 为何不会发生冲突?

6. 区分 MCS-51 单片机片外程序存储器和片外数据存储器的最可靠的方法是:

(1) 看其位于地址范围的低端还是高端

(2) 看其离 MCS-51 芯片的远近

(3) 看其芯片的型号是 ROM 还是 RAM

(4) 看其是与 \overline{RD} 信号连接还是与 \overline{PSEN} 信号连接

7. 在存储器扩展中, 无论是线选法还是译码法, 最终都是为扩展芯片的()端提供信号。

8. 请写出图 8-18 中 4 片程序存储器 27128 各自所占的地址空间。

9. 起止范围为 0000H~3FFFH 的存储器的容量是()KB。

10. 在 MCS-51 中, PC 和 DPTR 都用于提供地址, 但 PC 是为访问()存储器提供地址, 而 DPTR 是为访问()存储器提供地址。

11. 11 根地址线可选()个存储单元, 16 KB 存储单元需要()根地址线。

12. 32 KB RAM 存储器的首地址若为 2000H, 则末地址为()H。

13. 现有 8031 单片机、74LS373 锁存器、1 片 2764 EPROM 和 2 片 6116 RAM, 请使用它们组成 1 个单片机应用系统, 要求:

(1) 画出硬件电路连线图, 并标注主要引脚;

(2) 指出该应用系统程序存储器空间和数据存储器空间各自的地址范围。

14. 使用 89C51 芯片外扩 1 片 E²PROM 2864, 要求 2864 兼作程序存储器和数据存储器, 且首地址为 8000H。要求:

(1) 确定 2864 芯片的末地址;

(2) 画出 2864 片选端的地址译码电路;

(3) 画出该应用系统的硬件连线图。

第9章 MCS-51 扩展 I/O 接口的设计

9.1 I/O 接口扩展概述

MCS-51 的 I/O (输入/输出)接口是 MCS-51 单片机与外部设备(简称外设)交换信息的桥梁。由第 8 章的介绍可知,I/O 扩展也属于系统扩展的一部分。虽然 MCS-51 本身已有 4 个 I/O 口,但是 P0 口和 P2 口用作 16 位地址总线和 8 位数据总线,真正用作 I/O 口线的只有 P1 口的 8 位 I/O 线和 P3 口的某些位线。由于 MCS-51 的 I/O 资源有限,因此,在多数应用系统中,MCS-51 单片机都需要外扩 I/O 接口电路。

9.1.1 I/O 接口的功能

MCS-51 扩展的 I/O 接口电路主要应满足以下功能要求:

1. 实现和不同外设的速度匹配

不同外设的工作速度差别很大,但大多数外设的速度很慢,无法和 μs 量级的单片机速度相比。MCS-51 和外设间的数据传送方式有同步、异步、中断三种。无论采用哪种数据传送方式来设计 I/O 接口电路,单片机只能在确认外设已为数据传送做好准备的前提下才能进行 I/O 操作。而要知道外设是否准备好,就需要 I/O 接口电路与外设之间传送状态信息,以实现单片机与外设之间的速度匹配。

2. 输出数据锁存

由于单片机的工作速度快,数据在数据总线上保留的时间十分短暂,无法满足慢速外设的数据接收。所以,在扩展的 I/O 接口电路中应具有数据锁存器,以保证输出数据能为接收设备所接收。可见数据输出锁存应成为 I/O 接口电路的一项重要功能。

3. 输入数据三态缓冲

输入设备向单片机输入数据时,要经过数据总线,但数据总线上可能“挂”有多个数据源,为了传送数据时不发生冲突,只允许当前时刻正在进行数据传送的数据源使用数据总线,其余的数据源应处于隔离状态,为此要求接口电路能为数据输入提供三态缓冲功能。

9.1.2 I/O 端口的编址

在介绍 I/O 端口编址之前,首先要弄清楚 I/O 接口(Interface)和 I/O 端口(Port)的概念。I/O 端口简称 I/O 口,常指 I/O 接口电路中具有端口地址的寄存器或缓冲器。I/O 接口是指单片机与外设间的 I/O 接口芯片。一个 I/O 接口芯片可以有多个 I/O 端口,传送数据的称为数据口,传送命令的称为命令口,传送状态的称为状态口。当然,并不是所有的外设都需要三种接口齐全的 I/O 接口。

因此,I/O 端口的编址实际上是给所有 I/O 接口中的端口编址,以便 CPU 通过端口地址和外设交换信息。常用的 I/O 端口编址有两种方式,一种是独立编址方式,另一种是统一编址方式。

1. 独立编址方式

独立编址方式就是 I/O 地址空间和存储器地址空间分开编址。独立编址的优点是 I/O 地址空间和存储器地址空间相互独立,界限分明。但是,却需要设置一套专门的读写 I/O 的指令和控制信号。

2. 统一编址方式

这种编址方式是把 I/O 端口的寄存器与数据存储器单元同等对待,统一进行编址。统一编址方式的优点是不需要专门的 I/O 指令,直接使用访问数据存储器的指令进行 I/O 操作,简单、方便且功能强。

MCS-51 单片机使用的是 I/O 和外部数据存储器 RAM 统一编址的方式,用户可以把外部 64 KB 的数据存储器 RAM 空间的一部分作为 I/O 接口的地址空间,每一接口芯片中的 1 个功能寄存器(端口)的地址就相当于 1 个 RAM 存储单元,CPU 可以像访问外部存储器 RAM 那样访问 I/O 接口芯片,对其功能寄存器进行读、写操作。

9.1.3 I/O 数据的几种传送方式

为了实现和不同外设的速度匹配,I/O 接口必须根据不同外设选择恰当的 I/O 数据传送方式。I/O 数据传送的几种方式是:同步传送、异步传送和中断传送。

1. 同步传送方式

同步传送又称为无条件传送。当外设速度可与单片机速度相比拟时,常常采用同步传送方式,最典型的同步传送就是单片机和外部数据存储器之间的数据传送。

2. 查询传送方式

查询传送又称为有条件传送,也称异步传送。单片机通过查询得知外设准备好后,再进行数据传送。异步传送的优点是通用性好,硬件连线和查询程序十分简单,但是效率不高。为了提高单片机的工作效率,通常采用中断传送方式。

3. 中断传送方式

中断传送方式是利用 MCS-51 本身的中断功能和 I/O 接口的中断功能来实现 I/O 数据的传送。单片机只有在外设准备好后,发出数据传送请求,才中断主程序,而进入与外设进行数据传送的中断服务程序,进行数据的传送。中断服务完成后又返回主程序继续执行。因此,采用中断方式可以大大提高单片机的工作效率。

9.1.4 I/O 接口电路

下面来讨论如何实现 I/O 接口的扩展。MCS-51 单片机是 Intel 公司的产品,而 Intel 公司的配套可编程 I/O 接口芯片的种类齐全,这就为 MCS-51 单片机扩展 I/O 接口提供了很大的方便。

Intel 公司常用的外围 I/O 接口芯片有:

(1) 8255A:可编程的通用并行接口电路(3 个 8 位 I/O 口)。

(2) 8155H:可编程的 IO/RAM 扩展接口电路(2 个 8 位 I/O 口,1 个 6 位 I/O 口,256 个 RAM 字节单元,1 个 14 位的减法定时器/计数器)。

它们都可以和 MCS-51 单片机直接连接,且接口逻辑十分简单。此外,74LS 系列的 TTL 电路也可以作为 MCS-51 的扩展 I/O 口,如 74LS244、74LS273 等。本章除了介绍上述各种 I/O 接口电路与 MCS-51 单片机的接口设计,最后还介绍如何利用 MCS-51 的串行口来扩展并行 I/O 口。

9.2 MCS-51 与可编程并行 I/O 芯片 8255A 的接口设计

9.2.1 8255A 芯片介绍

8255A 是 Intel 公司生产的可编程并行 I/O 接口芯片,它具有 3 个 8 位的并行

I/O 口, 3 种工作方式, 可通过编程改变其功能, 因而使用灵活方便, 通用性强, 可作为单片机与多种外围设备连接时的中间接口电路。8255A 的引脚及内部的结构如图 9-1 和图 9-2 所示。

1. 引脚说明

由图 9-1, 8255A 共有 40 只引脚, 采用双列直插式封装, 各引脚功能如下:

D7~D0: 三态双向数据线, 与单片机数据总线连接, 用来传送数据信息。

\overline{CS} : 片选信号线, 低电平有效, 表示本芯片被选中。

\overline{RD} : 读出信号线, 低电平有效, 控制 8255A 中数据的读出。

\overline{WR} : 写入信号线, 低电平有效, 控制向 8255A 数据的写入。

V_{CC} : +5 V 电源。

PA7~PA0: A 口输入/输出线。

PB7~PB0: B 口输入/输出线。

PC7~PC0: C 口输入/输出线。

A1~A0: 地址线, 用来选择 8255A 内部的 4 个端口。

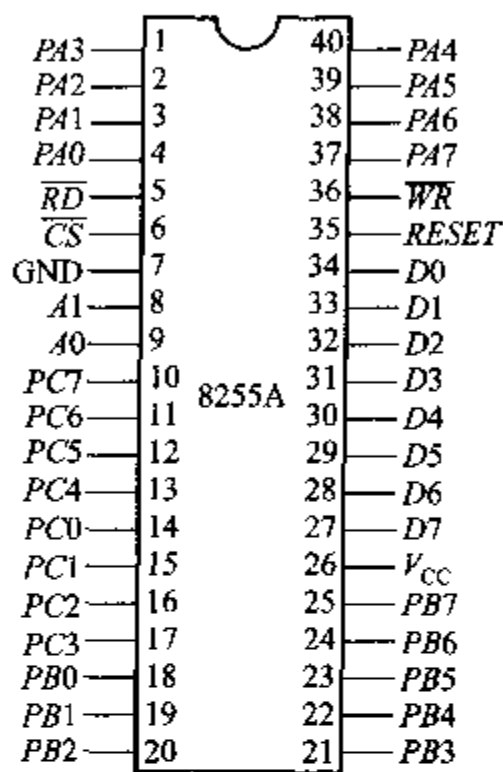


图 9-1 8255A 的引脚

2. 内部结构

8255A 内部结构见图 9-2, 其中包括 3 个并行数据输入/输出端口, 2 个工作方式的控制电路, 1 个读/写控制逻辑电路和 8 位数据总线缓冲器。各部件的功能如下:

(1) 端口 A、B、C

8255A 有 3 个 8 位并行口, PA、PB 和 PC。都可以选择作为输入/输出工作模式, 但在功能和结构上有些差异。

PA 口: 1 个 8 位数据输出锁存器和缓冲器; 1 个 8 位数据输入锁存器。

PB 口: 1 个 8 位数据输出锁存器和缓冲器; 1 个 8 位数据输入缓冲器。

PC 口: 1 个 8 位的输出锁存器; 1 个 8 位数据输入缓冲器。

通常 PA 口、PB 口作为输入/输出口, PC 口可作为输入/输出口, 也可在软件的控制下, 分为 2 个 4 位的端口, 作为端口 A、B 选通方式操作时的状态控制信号。

(2) A 组和 B 组控制电路

这是 2 组根据 CPU 写入的命令字控制 8255A 工作方式的控制电路。A 组控制 PA 口和 PC 口的上半部 (PC7~PC4); B 组控制 PB 口和 PC 口的下半部 (PC3~PC0), 并可根据命令字对端口的每一位实现按位置位或复位。

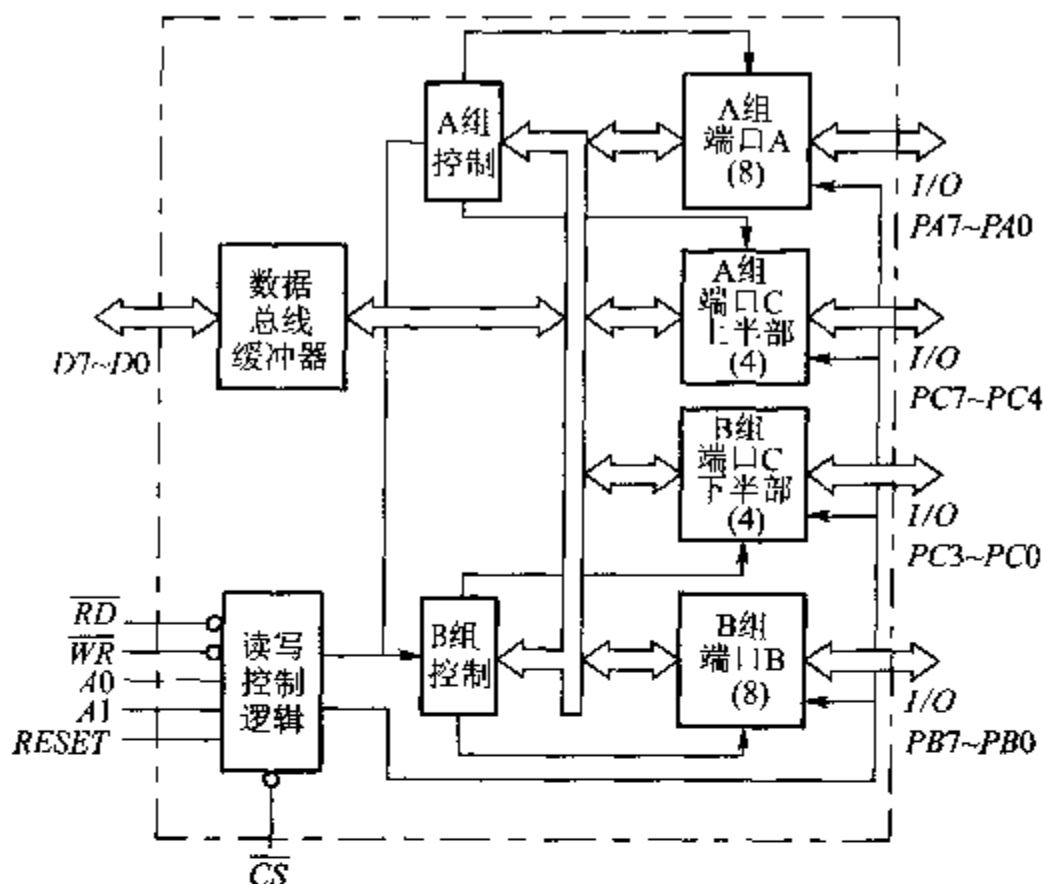


图 9-2 8255A 的内部结构

(3) 数据总线缓冲器

数据总线缓冲器是 1 个三态双向 8 位缓冲器,作为 8255A 与系统总线之间的接口,用来传送数据、指令、控制命令以及外部状态信息。

(4) 读/写控制逻辑电路

读/写控制逻辑电路接收 CPU 发来的控制信号 \overline{RD} 、 \overline{WR} 、RESET、地址信号 A1~A0 等,然后根据控制信号的要求,将端口数据读出,送往 CPU 或者将 CPU 送来的数据写入端口。

各端口的工作状态与控制信号的关系如表 9-1 所示。

表 9-1 8255A 端口工作状态选择表

A1	A2	\overline{RD}	\overline{WR}	\overline{CS}	工作状态
0	0	0	1	0	A 口数据→数据总线(读端口 A)
0	1	0	1	0	B 口数据→数据总线(读端口 B)
1	0	0	1	0	C 口数据→数据总线(读端口 C)
0	0	1	0	0	总线数据→A 口(写端口 A)
0	1	1	0	0	总线数据→B 口(写端口 B)
1	0	1	0	0	总线数据→C 口(写端口 C)
1	1	1	0	0	总线数据→控制字寄存器(写控制字)
×	×	×	×	1	数据总线为三态
1	1	0	1	0	非法状态
×	×	1	1	0	数据总线为三态

9.2.2 工作方式选择控制字及 C 口置位/复位控制字

8255A 有 3 种基本工作方式:

- (1) 方式 0: 基本输入输出;
- (2) 方式 1: 选通输入输出;
- (3) 方式 2: 双向传送(仅 A 口有此工作方式)。

1. 工作方式选择控制字

3 种工作方式由写入控制字寄存器的方式控制字来决定。方式控制字的格式如图 9-3 所示。

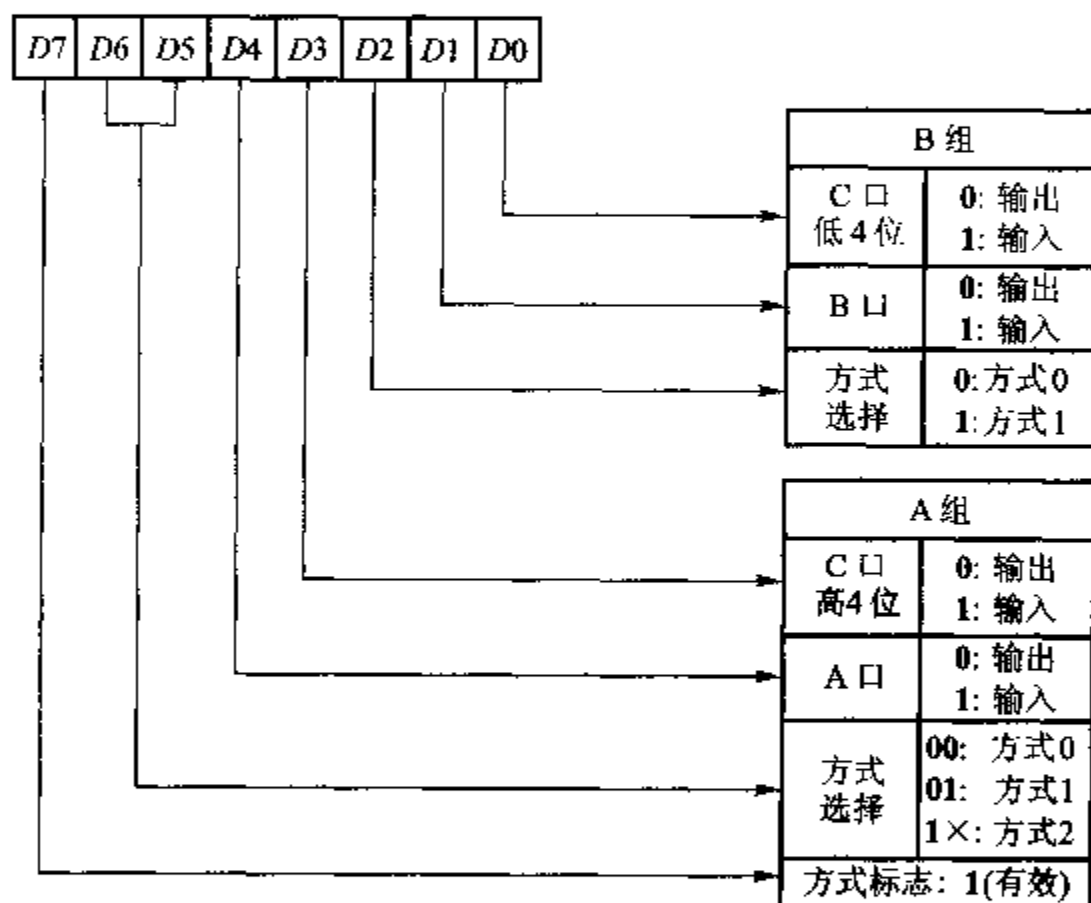


图 9-3 8255A 的方式控制字

3 个端口中 C 口被分为 2 个部分, 上半部分随 A 口称为 A 组, 下半部分随 B 口称为 B 组。其中 A 口可工作于方式 0、1 和 2, 而 B 口只能工作在方式 0 和 1。例如, 写入工作方式控制字 95H, 可将 8255A 编程为: A 口方式 0 输入, B 口方式 1 输出, C 口的上半部分(PC7~PC4)输出, C 口的下半部分(PC3~PC0)输入。

2. C 口按位置位/复位控制字

C 口 8 位中的任一位, 可用 1 个写入控制口的置位/复位控制字来对 C 口按位来置 1 或是清 0。这个功能主要用于位控。C 口按位置位/复位控制字的格式如图 9-4 所示。

例如, 07H 写入控制口, 置 1PC3; 08H 写入控制口, PC4 清 0。

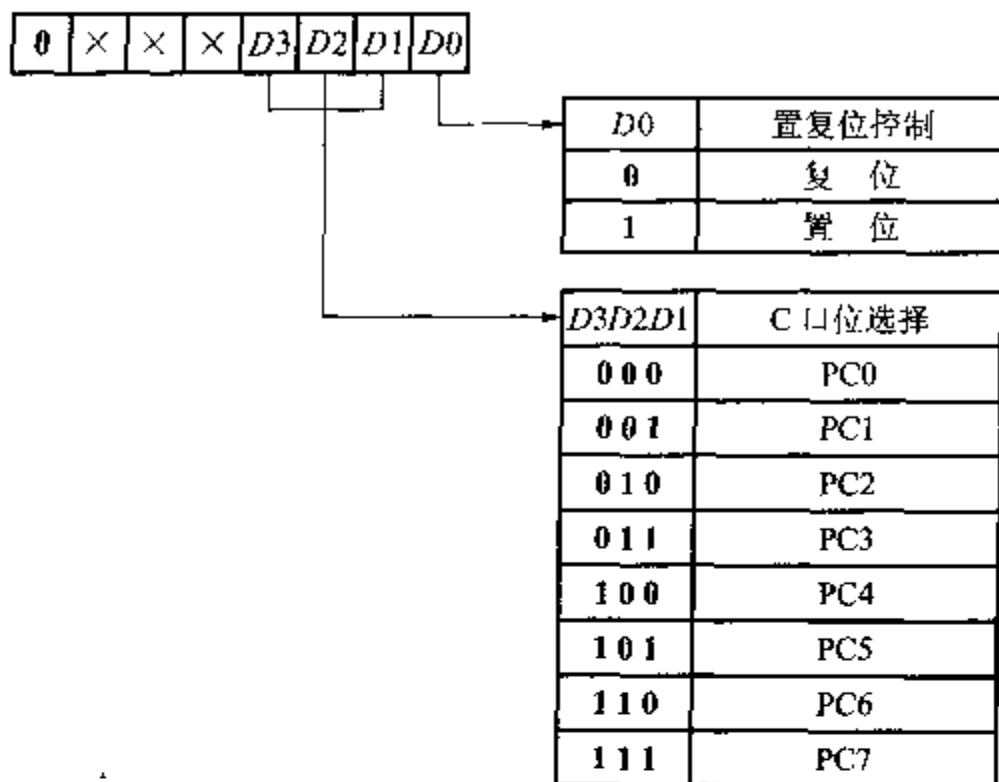


图 9-4 C 口按位置位/复位控制字格式

9.2.3 8255A 的 3 种工作方式

1. 方式 0

方式 0 是一种基本的输入/输出方式。在方式 0 下, MCS-51 可对 8255A 进行 I/O 数据的无条件传送, 例如, 读入一组开关状态, 控制一组指示灯的亮、灭。实现这些操作, 并不需要联络信号, 外设的 I/O 数据可在 8255A 的各端口得到锁存和缓冲。因此, 8255A 的方式 0 称为基本输入/输出方式。

方式 0 下, 3 个端口都可以由程序设置为输入或输出, 不需要应答联络信号。方式 0 的基本功能为:

- (1) 具有 2 个 8 位端口(A、B)和 2 个 4 位端口(C 的上半部分和下半部分)。
- (2) 任一个端口都可以设定为输入或输出, 各端口的输入、输出可构成 16 种组合。
- (3) 数据输出时锁存, 输入时不锁存。

8255A 的 A 口、B 口和 C 口均可设定为方式 0, 并可根据需要规定各端口为输入方式或输出方式。例如: 假设 8255A 的控制字寄存器地址为 FF7FH, 则令 A 口和 C 口的高 4 位工作在方式 0 输出, B 口和 C 口的低 4 位工作在方式 0 输入, 这时, 初始化程序为:

```
MOV DPTR, #0FF7FH      ;控制字寄存器地址送 DPTR
MOV A, #83H             ;方式控制字 83H 送 A
MOVX @DPTR, A           ;83H 送控制字寄存器
```

2. 方式 1

方式 1 是一种选通输入/输出工作方式。A 口和 B 口皆可独立地设置成这种工作方式。在方式 1 下,8255A 的 A 口和 B 口通常用于 I/O 数据的传送,C 口用作 A 口和 B 口的联络线,以实现中断方式传送 I/O 数据。C 口的 PC7~PC0 联络线是在设计 8255A 时规定的,其各位分配见图 9-5 和图 9-7,图中,标有 I/O 各位仍可用作基本输入/输出,不作联络线用。

下面简单介绍方式 1 输入/输出时的控制联络信号和工作原理。

(1) 方式 1 输入

当任一端口工作于方式 1 输入时,控制联络信号如图 9-5 所示, \overline{STB} 与 \overline{IBF} 构成了一对应答联络信号,各个控制联络信号的功能如下:

\overline{STB} :选通输入,低电平有效。是由输入外设送来的输入信号。

\overline{IBF} :输入缓冲器满,高电平有效。表示数据已送入 8255A 的输入锁存器,它由 \overline{STB} 信号的下降沿置位,由 \overline{RD} 信号的上升沿使其复位。

$INTR$:中断请求信号,高电平有效。由 8255A 输出,向 CPU 发中断请求。

$INTE A$:A 口中断允许信号,由 PC4 的置位/复位来控制,

$INTE B$:B 口中断允许信号,由 PC2 的置位/复位来控制。

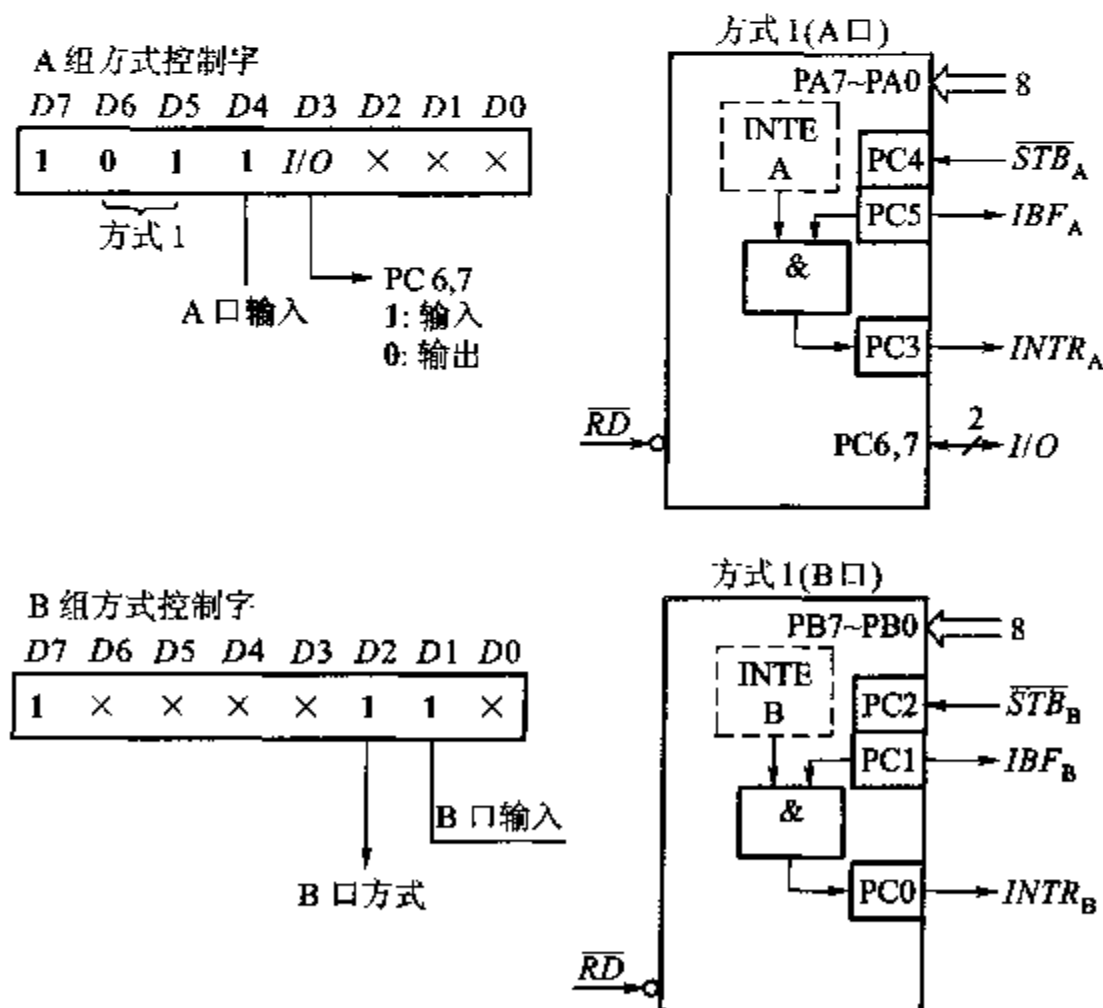


图 9-5 方式 1 输入联络信号

下面以 A 口的方式 1 输入为例,工作示意图见图 9-6。下面介绍方式 1 输入

的工作过程以及各控制联络信号的功能。

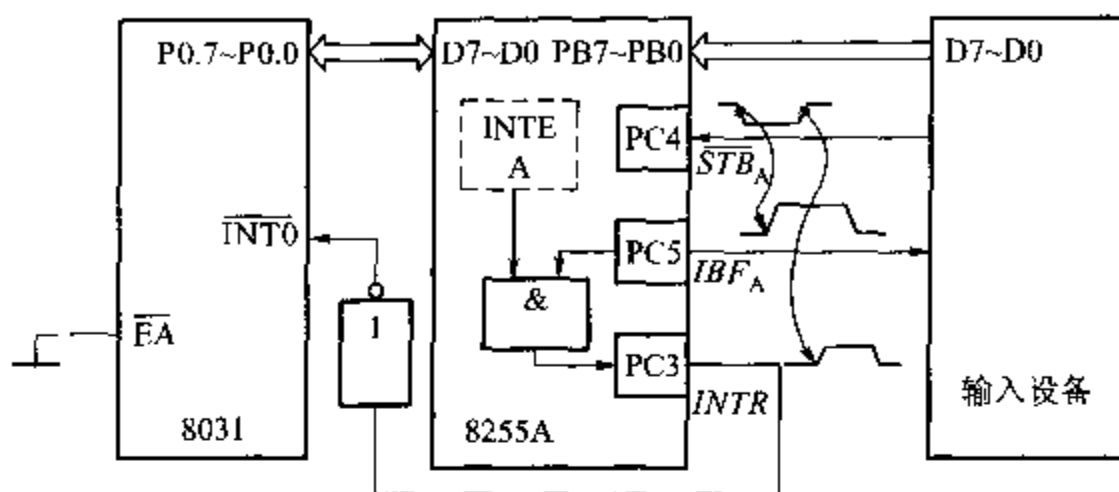


图 9-6 A 口方式 1 输入的工作示意图

① 当外设输入一个数据并送到 PA7~PA0 上时,输入设备自动在选通输入线 \overline{STB}_A 上向 8255A 发送一个低电平选通信号。

② 8255A 收到选通信号后:首先把 PA7~PA0 上输入的数据存入 A 口的输入数据缓冲/锁存器;然后使输入缓冲器输出线 IBF_A 变为高电平,以通知输入设备,8255A 的 A 口已收到它送来的输入数据。

③ 8255A 检测到联络线 \overline{STB}_A 由低电平变为高电平、 IBF_A 为 1 状态和中断允许触发器 $INTE_A$ 为 1 时,使输出线 $INTR_A$ (PC3) 变为高电平,向 8031 发出中断请求。 $INTE_A$ 的状态可由用户通过对 PC4 的置位/复位来控制。

④ 8031 响应中断后,可以通过中断服务程序从 A 口的输入数据缓冲/锁存器读取外设发来的输入数据。当输入数据被 CPU 读走后,8255A 撤消 $INTR_A$ 上的中断请求,并使 IBF_A 变为低电平,以通知输入外设可以送下一个输入数据。

(2) 方式 1 输出

当任何一个端口按照工作方式 1 输出时,控制联络信号如图 9-7 所示。 \overline{OBF} 与 \overline{ACK} 构成了一对应答联络信号,各控制联络信号的功能如下:

\overline{OBF} :输出缓冲器满信号,低电平有效,是 8255A 给外设的联络信号,表示 CPU 已经把数据输出给指定的端口,外设可以将数据取走。它由 \overline{WR} 信号的上升沿置 0(有效),由 \overline{ACK} 信号的下降沿置 1(无效)。

\overline{ACK} :外设的响应信号,低电平有效。指示 CPU 输出给 8255A 的数据已经由外设取走。

$INTR$:中断请求信号,高电平有效。表示该数据已被外设取走,请求 CPU 继续输出下一个数据。中断请求的条件是信号 \overline{ACK} 、 \overline{OBF} 和 $INTE$ (中断允许) 为高电平,中断请求信号由 \overline{WR} 的下降沿复位。

$INTE A$:由 PC6 的置位/复位来控制。

$INTE B$:由 PC2 的置位/复位来控制。

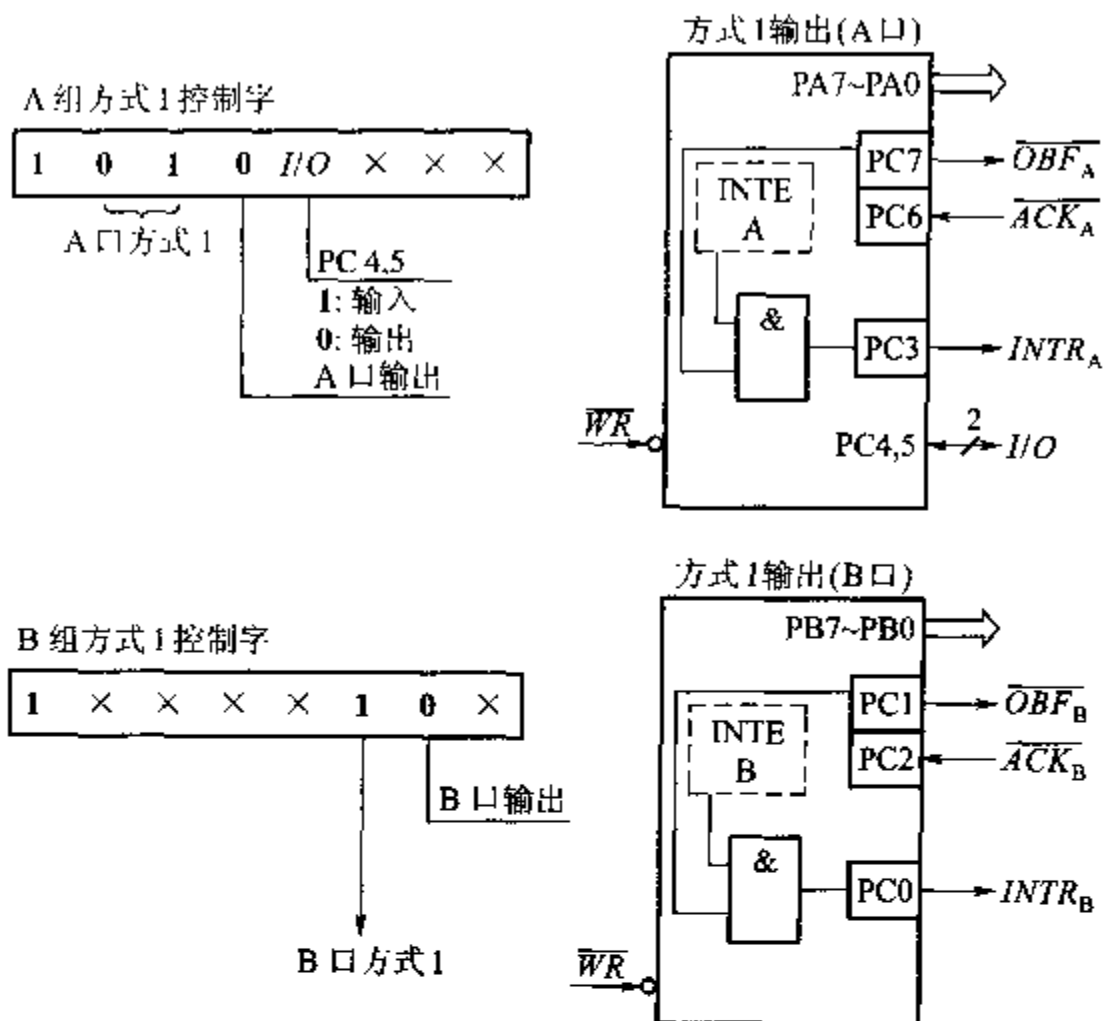


图 9-7 方式 1 输出联络信号

图 9-8 为 B 口工作于方式 1 输出下的工作示意图。

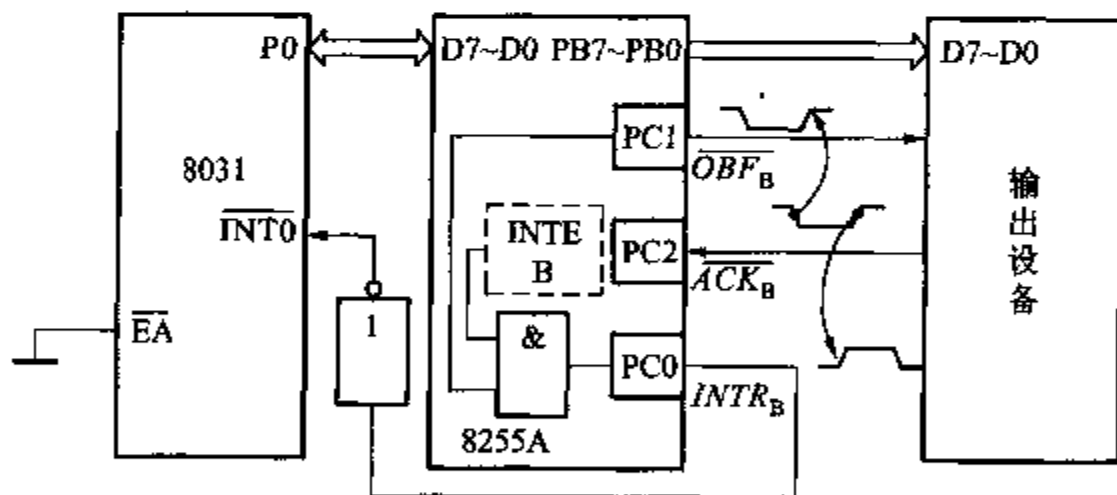


图 9-8 B 口方式 1 选通输出的工作示意图

B 口在方式 1 输出的工作过程如下：

(1) 8031 可以通过 $MOVX @Ri, A$ 指令把输出数据送到 B 口的输出数据锁存器, 8255A 收到后便令输出缓冲器满引脚 \overline{OBF}_B (PC7) 变为低电平, 以通知输出设备输出的数据已在 B 口的 PB7~PB0 上。

(2) 输出设备收到 \overline{OBF}_B 上低电平后, 先从 PB7~PB0 上取走输出数据; 然后使 \overline{ACK}_B 线变为低电平, 以通知 8255A 输出设备已收到输出数据。

(3) 8255A 从回答输入线 \overline{ACK}_B 收到低电平后就对 \overline{OBF}_B 和中断允许控制位

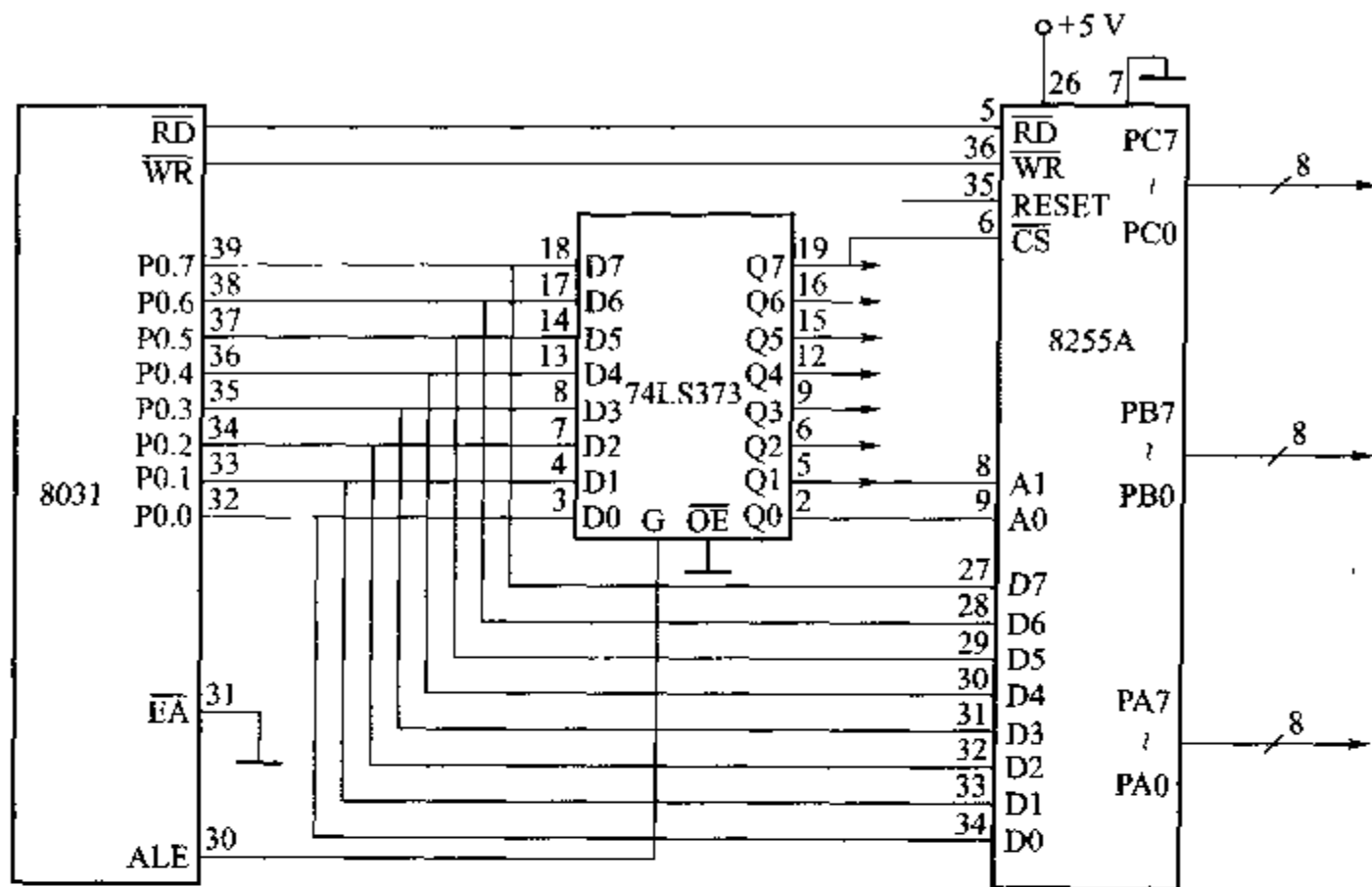


图 9-10 8031 扩展 1 片 8255A 的电路

A0,分别接于 P0.7、P0.1、P0.0,其它地址线全悬空。显然只要保证 P0.7 为低电平时,选中该 8255A,若 P0.1、P0.0 再为 00 则选中 8255A 的 A 口,同理 P0.1、P0.0 为 01、10、11 分别选中 B 口、C 口及控制口。若地址用 16 位表示,其它无用端全设为 1,则 8255A 的 A、B、C 及控制口地址分别为:

FF7CH, FF7DH, FF7EH, FF7FH

如果没有用到的位取 0, 则 4 个地址为 0000H、0001H、0002H、0003H, 只要保证 $\overline{\text{CS}}$ 、A1、A0 的状态, 无用位设为 0 或 1 均可。掌握了确定地址的方法, 使用者可灵活选择地址。

3. 软件编程

在实际的应用系统中,必须根据外部设备的类型选择 8255A 的操作方式,并在初始化程序中把相应控制字写入控制口。下面根据图 9-10,来说明 8255A 的编程方法。

例 9-1 要求 8255A 工作在方式 0,且 A 口作为输入,B 口、C 口作为输出,则程序如下:

MOV	A, #90H	; A 口方式 0 输入, B 口、C 口输出的方式控制字 ; 送 A
MOV	DPTR, #0FF7FH	; 控制寄存器地址 → DPTR
MOVX	@DPTR, A	; 方式控制字 → 控制寄存器
MOV	DPTR, #0FF7CH	; A 口地址 → DPTR
MOVX	A, @DPTR	; 从 A 口读数据

```
MOV    DPTR, #0FF7DH    ;B 口地址 →DPTR
MOV    A, #DATA1        ;要输出的数据 DATA1→A
MOVX   @DPTR, A         ;将 DATA1 送 B 口输出
MOV    DPTR, #0FF7EH    ;C 口地址 →DPTR
MOV    A, #DATA2        ;DATA2→A
MOVX   @DPTR, A         ;将 DATA2 送 C 口输出
```

例 9-2 对端口 C 的置位/复位。

8255A C 口 8 位中的任一位,均可用指令来置位或复位。例如,如果想把 C 口的 PC5 置 1,相应的控制字为:00001011B=0BH(关于 8255A 的 C 口置位/复位的控制字说明参见图 9-4),程序如下:

```
MOV    R1, #7FH        ;控制口地址→R1
MOV    A, #0BH         ;控制字→A
MOVX   @R1, A          ;控制字 → 控制口, PC5=1
```

如果想把 C 口的第 6 位 PC5 复位,相应的控制字:00001010B=0AH,程序如下:

```
MOV    R1, #7FH        ;控制口地址→R1
MOV    A, #0AH         ;控制字→A
MOVX   @R1, A          ;控制字→控制口, PC5=0
```

8255A 接口芯片在 MCS-51 单片机应用系统中广泛用于外部设备,如打印机、键盘、显示器以及作为控制信息的输入、输出口。关于 8255A 在这些方面的具体应用请参见本书有关章节。

9.3 MCS-51 与可编程 RAM/IO 芯片 8155H 的接口

Intel 8155H 芯片内包含有 256 B 的 RAM 存储器(静态),RAM 的存取时间为 400 ns。2 个可编程的 8 位并行口 PA 和 PB,1 个可编程的 6 位并行口 PC,以及 1 个 14 位减法定时器/计数器。PA 口和 PB 口可工作于基本输入/输出方式(同 8255A 的方式 0)或选通输入/输出方式(同 8255A 的方式 1)。8155H 可直接与 MCS-51 单片机相连,不需要增加任何硬件逻辑。由于 8155H 既有 I/O 口又具有 RAM 和定时器/计数器,因而是 MCS-51 单片机系统中常选用的外围接口芯片之一。

9.3.1 8155H 芯片介绍

1. 8155H 的结构

8155H 的逻辑结构如图 9-11 所示。

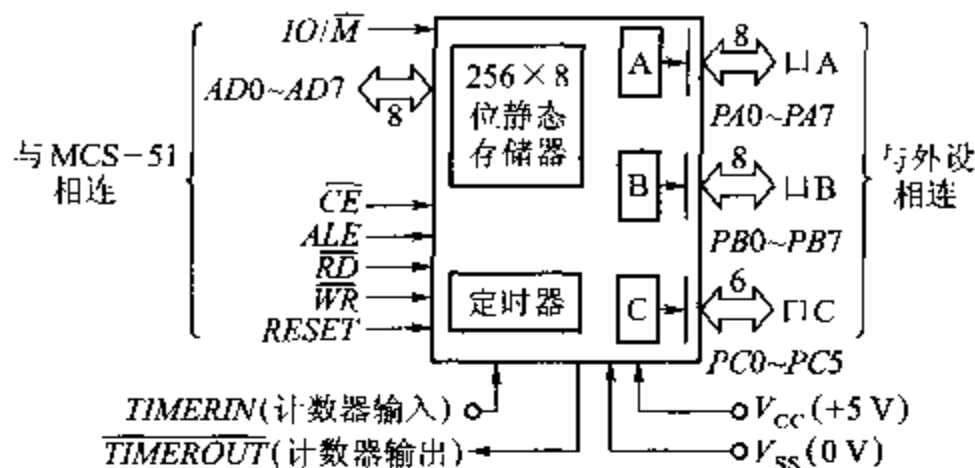


图 9-11 8155H 的逻辑结构

2. 8155H 的引脚功能

如图 9-12 所示,8155H 共有 40 条引脚线,采用双列直插式封装。

8155H 的各引脚功能如下:

(1) AD7~AD0(8 条)

AD7~AD0 为地址/数据线,与 MCS-51 的 P0 口相连,用于分时传送地址/数据信息。

(2) I/O 总线(22 条)

PA7~PA0 为通用 I/O 线,用于传送 A 口上的外设数据,数据传送方向由写入 8155H 的命令字决定(见图 9-13);PB7~PB0 为通用 I/O 线,用于传送 B 口上的外设数据,数据传送方向也由 8155H 命令字决定。PC5~PC0 为数据/控制线,共有 6 条,在通用 I/O 方式下,用作传送 I/O 数据;在选通 I/O 方式下,用作传送命令/状态信息(见表 9-3)。

(3) 控制总线(8 条)

RESET:复位输入线,在 RESET 线上输入 1 个宽大于 600 ns 的正脉冲时,8155H 立即处于复位状态,A、B、C 3 口也定义为输入方式。

CE 和 IO/M:为 8155H 片选输入线,若 CE=0,则 CPU 选中本 8155H 工作;否则,本 8155H 不工作。IO/M 为 I/O 端口或 RAM 存储器的选通输入线;若 IO/M=0,则 CPU 选中 8155H 的 RAM 存储器工作;若 IO/M=1,则 CPU 选中 8155H 片内某一存储器。

RD 和 WR:RD 是 8155H 的读/写命令输入线,WR 为写命令线,当 RD=0 和 WR=1 时,8155H 处于读出数据状态;当 RD=1 和 WR=0 时,8155H 处于写入数据状态。

ALE:为允许地址输入线,高电平有效。若 ALE=1,则 8155H 允许

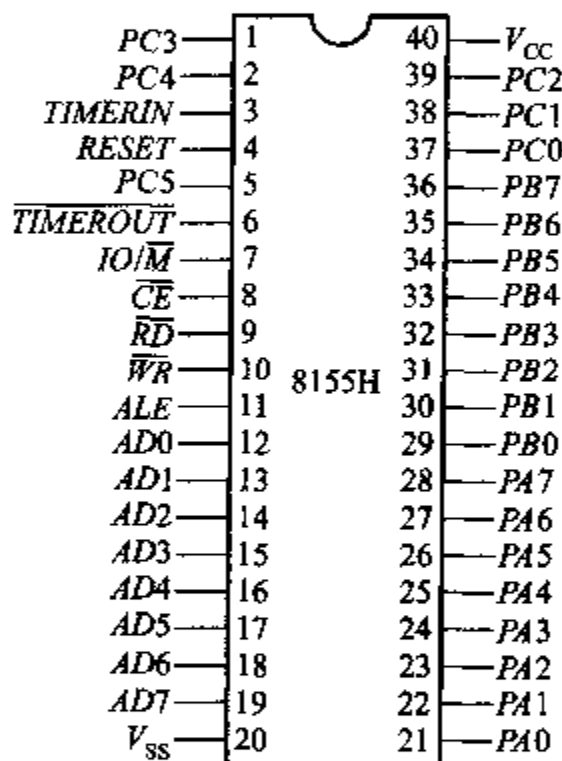


图 9-12 8155H 的引脚

AD7~AD0 上地址锁存到地址锁存器;否则,8155H 的地址锁存器处于封锁状态。8155H 的 ALE 常和 MCS-51 的 ALE 端相连。

TIMERIN 和 $\overline{\text{TIMEROUT}}$:TIMERIN 是计数器输入线,输入的脉冲上跳沿用于对 8155H 片内的 14 位计数器减 1。 $\overline{\text{TIMEROUT}}$ 为计数器输出线,当 14 位计数器减为 0 时就可以在该引线上输出脉冲或方波,输出信号的形状与所选的计数器工作方式有关。

(1) 电源线(2 条)

V_{CC} 为:5 V 电源输入线, V_{SS} 接地。

3. CPU 对 8155H I/O 端口的控制

8155H A、B、C 3 个端口的数据传送是由命令字和状态字控制的。

(1) 8155H 各端口地址分配

8155H 内部有 7 个寄存器,需要 3 位地址 A2~A0 上的不同组合来加以区分。表 9-2 列出了端口地址分配。

表 9-2 8155H 端口地址分配

\overline{CE}	IO/\overline{M}	A7	A6	A5	A4	A3	A2	A1	A0	所选端口
0	1	×	×	×	×	×	0	0	0	命令/状态寄存器
0	1	×	×	×	×	×	0	0	1	A 口
0	1	×	×	×	×	×	0	1	0	B 口
0	1	×	×	×	×	×	0	1	1	C 口
0	1	×	×	×	×	×	1	0	0	计数器低 8 位
0	1	×	×	×	×	×	1	0	1	计数器高 6 位
0	0	×	×	×	×	×	×	×	×	RAM 单元

注:×表示 0 或 1

(2) 8155H 的命令字

8155H 有一个控制命令寄存器和一个状态标志寄存器。8155H 的工作方式由 CPU 写入命令寄存器的命令字来确定。命令寄存器只能写入不能读出,命令寄存器中的 4 位用来设置 A 口、B 口和 C 口的工作方式。D4、D5 位用来确定 A 口、B 口以选通输入/输出方式工作时是否允许中断请求。D6、D7 位用来设置定时器/计数器的操作。命令字的格式如图 9-13 所示。

(3) 8155H 的状态字

在 8155H 中还设置有一个状态标志寄存器,用来存入 A 口和 B 口的状态标志。状态标志寄存器的地址与命令寄存器的地址相同,CPU 只能对其读出,不能写入。状态寄存器的格式如图 9-14 所示,CPU 可以直接查询。

下面仅对状态字中的 D6 位作以说明:

D6 为定时器中断状态标志位 TIMER。若定时器正在计数或开始计数前,

则 $D6=0$; 若定时器的计数长度已计满, 则 $D6=1$, 可作为定时器中断标志。在硬件复位或对它读出后又恢复为 0。

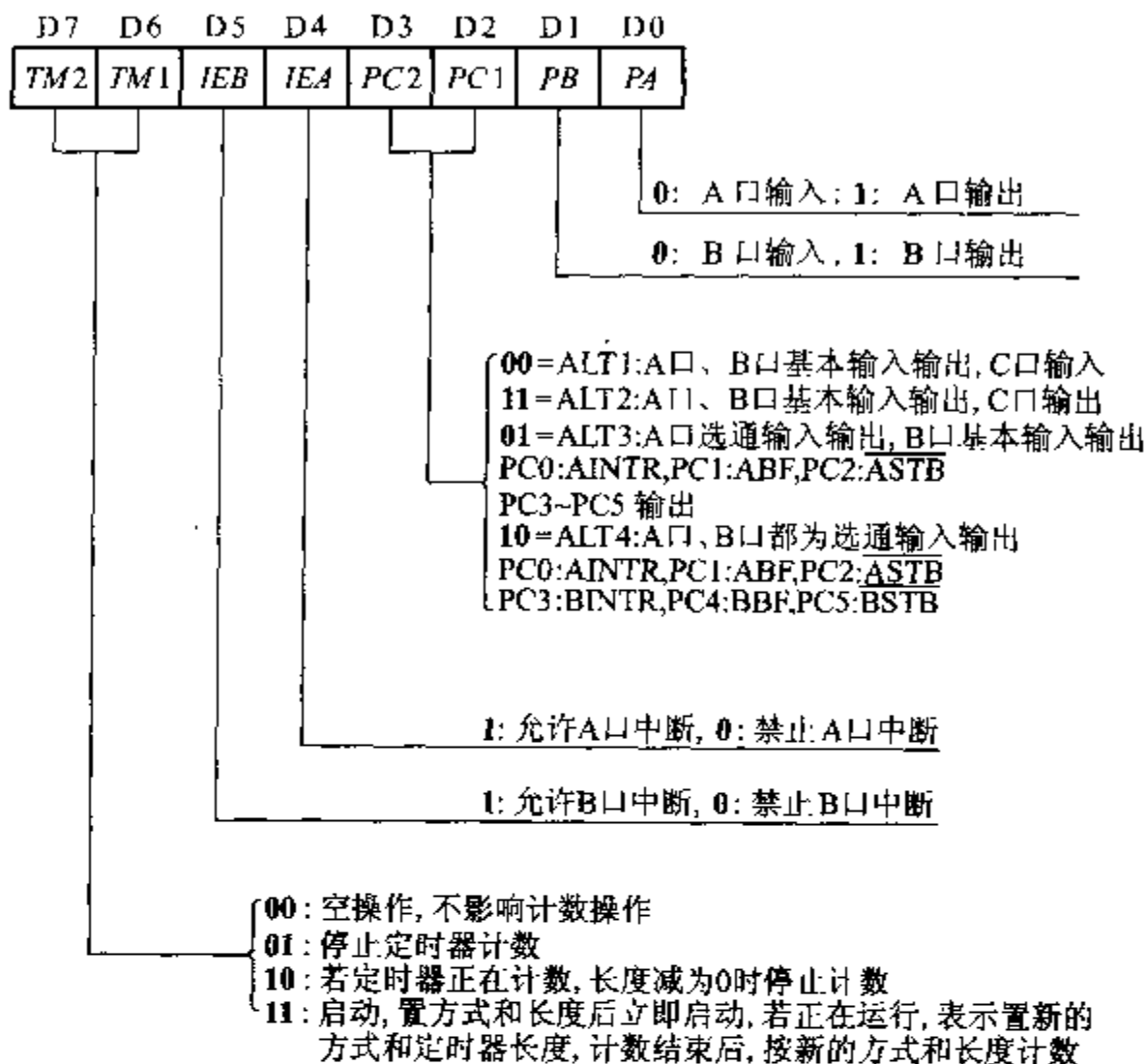


图 9-13 8155H 的命令字

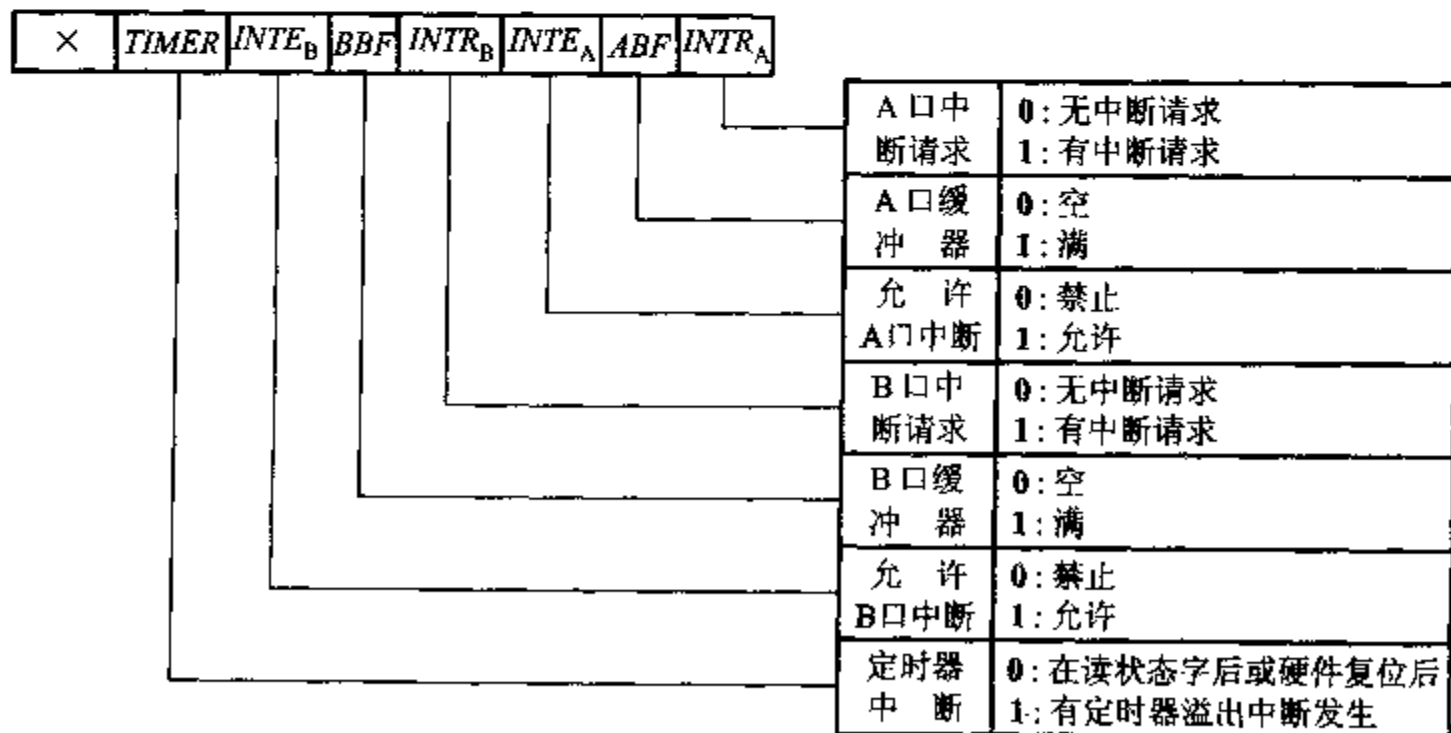


图 9-14 8155H 状态字格式

9.3.2 8155H的工作方式

下面介绍8155H的两种工作方式。

1. 存储器方式

8155H的存储器方式用于对片内256B RAM单元进行读写,若 $IO/\overline{M}=0$ 和 $\overline{CE}=0$,则CPU可以通过AD7~AD0上的地址选择RAM存储器中任一单元读写。

2. I/O方式

8155H的I/O方式分为基本I/O和选通I/O 2种工作方式,如表9-3所示。在I/O方式下,8155H可选择片内任一寄存器读写,端口地址由A2、A1、A0 3位决定(见表9-2)。

表9-3 C口在两种I/O工作方式下各位定义

C口	基本I/O方式		选通I/O方式	
	ALT1	ALT2	ALT3	ALT4
PC0	输入	输出	AINTR(A口中断)	AINTR(A口中断)
PC1	输入	输出	ABF(A口缓冲器满)	ABF(A口缓冲器满)
PC2	输入	输出	\overline{ASTB} (A口选通)	\overline{ASTB} (A口选通)
PC3	输入	输出	输出	BINTR(B口中断)
PC4	输入	输出	输出	BBF(B口缓冲器满)
PC5	输入	输出	输出	\overline{BSTB} (B口选通)

(1) 基本I/O方式

在基本I/O方式下,A、B、C 3口用作输入/输出,由图9-13所示的命令字决定。其中,A、B 2口的输入/输出由D1、D0决定,C口各位由D3、D2状态决定。例如:若把02H的命令字送到8155H命令寄存器,则8155H A口和C口各位设定为输入方式,B口设定为输出方式。

(2) 选通I/O方式

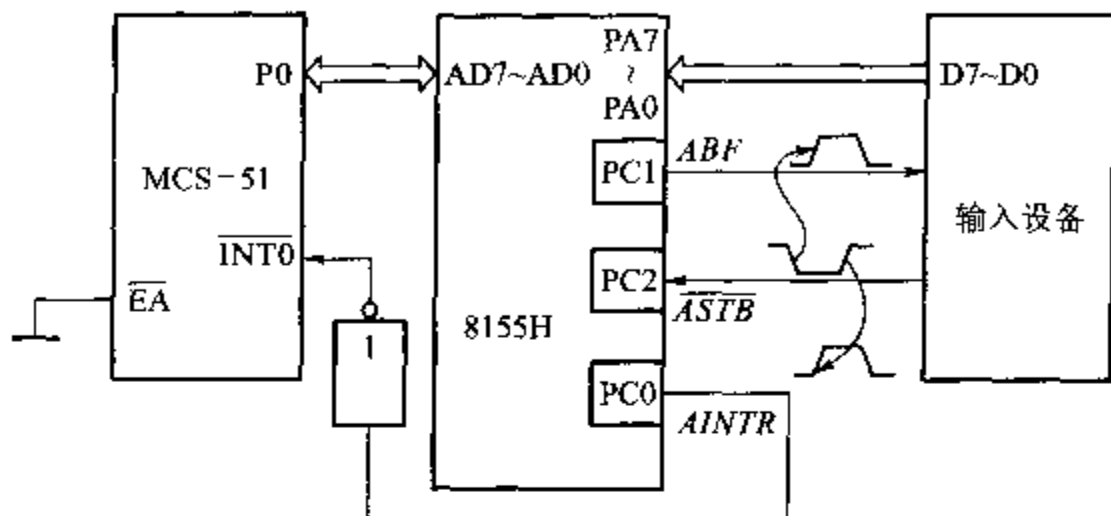
由命令字中D3、D2状态设定,A口和B口都可独立工作于这种方式。此时,A口和B口用作数据口,C口用作A口和B口的联络控制。C口各位联络线的定义是在设计8155H时规定的,其分配和命名如表9-3所示。

选通I/O方式又可分为选通I/O数据输入和选通I/O数据输出两种方式:

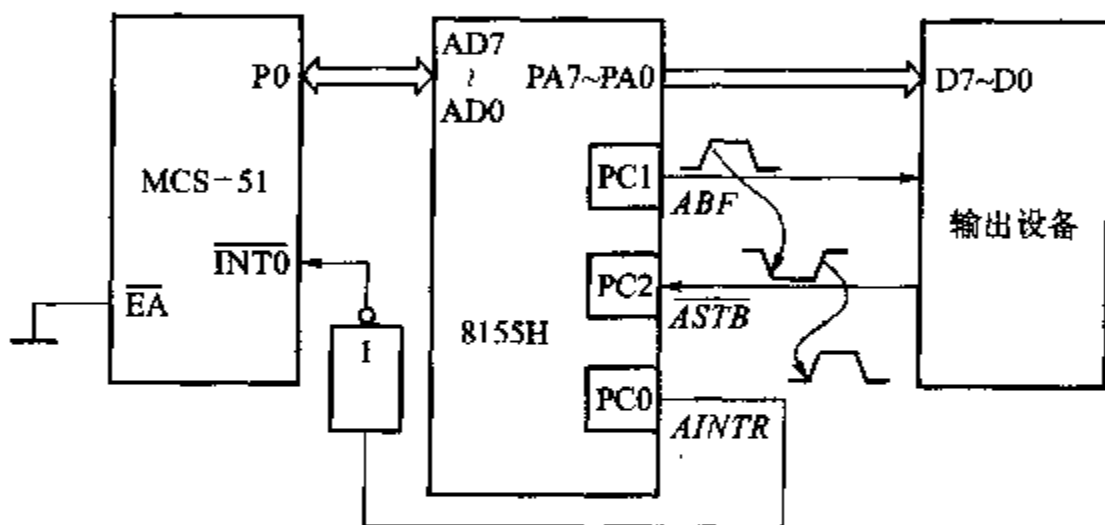
① 选通I/O数据输入

A口和B口都可设定为本工作方式;若命令字中 $D0=0$ 和 $D3、D2=10$ (或11),则A口设定为本工作方式;若命令字中 $D1=0$ 和 $D3、D2=11$,则B口设定为

本工作方式。工作过程和 8255A 的选通 I/O 输入的情况类似,如图 9-15(a)所示。



(a) 选通 I/O 数据输入示意图



(b) 选通 I/O 数据输出示意图

图 9-15 选通 I/O 工作方式示意图

② 选通 I/O 数据输出

A 口和 B 口都可设定为本工作方式;若命令字 $D_0=1$ 和 $D_3、D_2=10$ (或 11),则 A 口设定为本工作方式;若命令字中 $D_1=1$ 和 $D_3、D_2=11$,则 B 口设定为本工作方式。选通 I/O 数据的输出过程也和 8255A 的选通 I/O 输出情况类似,图 9-15(b)给出了它的示意图。

3. 内部定时器/计数器及使用

8155H 中有 1 个 14 位的定时器/计数器,可用来定时或对外部事件计数,CPU 可通过程序选择计数长度和计数方式。计数长度和计数方式由写入计数寄存器的控制字来确定,计数寄存器的格式如图 9-16 所示。

其中 $T_{13} \sim T_0$ 为计数器长度。 $M_2、M_1$ 用来设置定时器的输出方式。8155H 定时器 4 种工作方式及相应的 $\overline{\text{TIMEROUT}}$ 引脚输出波形如图 9-17 所示。

任何时候都可以设置定时器的长度和工作方式,但是必须将启动命令字写入

	D7	D6	D5	D4	D3	D2	D1	D0
$T_L(04H)$	T7	T6	T5	T4	T3	T2	T1	T0
	D7	D6	D5	D4	D3	D2	D1	D0
$T_H(05H)$	M2	M1	T13	T12	T11	T10	T9	T8

图 9-16 8155H 计数寄存器的格式

M2 M1	方式	定时器输出波形
0 0	单方波	
0 1	连续方波	
1 0	单脉冲	
1 1	连续脉冲	

图 9-17 8155H 定时器方式及 $\overline{\text{TIMEROUT}}$ 引脚输出波形

命令寄存器。如果定时器正在计数,那么,只有在写入启动命令字之后,定时器才接收新的计数器长度并按新的工作方式计数。

若写入定时器的初值为奇数, $\overline{\text{TIMEROUT}}$ 的方波输出是不对称的,例如初值为 9 时,定时器输出在 5 个脉冲周期内为高电平,4 个脉冲周期内为低电平,如图 9-18 所示。

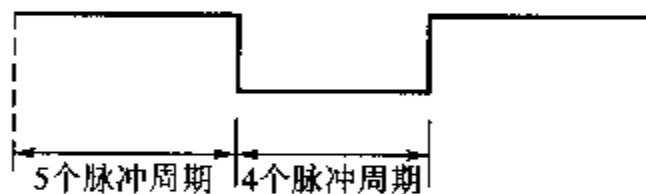


图 9-18 不对称方波输出(长度为 9 个脉冲周期)

值得注意的是,8155H 的定时器初值不是从 0 开始,而是从 2 开始。这是因为如果选择定时器的输出为方波形式(无论是单方波还是连续方波),则规定是从启动计数开始,前半计数输出为高电平,后半计数输出为低电平。显然,如果计数初值是 0 或 1,就无法产生这种方波。因此 8155H 计数器的初值范围是:3FFFH~2H。

如果硬要将 0 或 1 作为初值写入,其效果将与送入初值 2 的情况一样。

8155H 复位后并不预置定时器的方式和长度,但是停止计数器计数。

9.3.3 MCS-51 与 8155H 接口及软件编程

1. MCS-51 与 8155H 的硬件接口电路

MCS-51 单片机可以和 8155H 直接连接而不需要任何外加逻辑器件。

8031 和 8155H 的接口电路如图 9-19 所示。

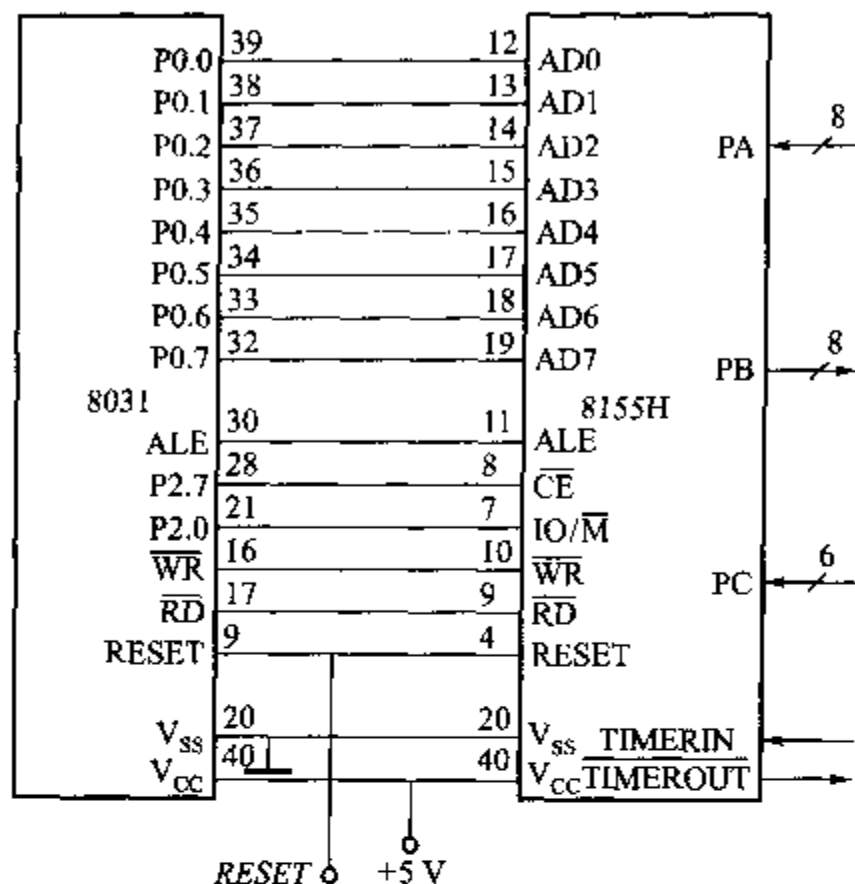


图 9-19 8155H 和 8031 的接口电路

在图 9-19 中,8031 单片机 P0 口输出的低 8 位地址不需要另外加锁存器而直接与 8155H 的 AD0~AD7 相连,既作低 8 位地址总线又作数据总线,地址锁存直接用 ALE 引脚在 8155H 锁存。8155H 的 \overline{CE} 端接 P2.7, $\overline{IO/M}$ 端与 P2.0 相连。当 P2.7 位低电平时,若 P2.0=0,访问 8155H 的 RAM 单元。由此我们得到图 9-19 中 8155H 的地址编码如下:

RAM 单元地址: 7E00H~7EFFH

I/O 口地址:	{	命令/状态口:	7F00H
		PA 口:	7F01H
		PB 口:	7F02H
		PC 口:	7F03H
		定时器低 8 位:	7F04H
		定时器高 6 位:	7F05H

2. 8155H 的编程举例

根据图 9-19 所示的接口电路,说明对 8155H 的具体操作。

(1) 初始化程序设计

例 9-3 若 A 口定义为基本输入方式,B 口定义为基本输出方式,对输入脉冲进行 24 分频(8155H 计数器的最高计数频率为 4 MHz),则 8155H 的 I/O 初始化程序如下:

```

START:  MOV    DPTR, #7F04H    ;指针指向定时器低 8 位
        MOV    A, #18H        ;计数初值 16H 送 A
        MOVX   @DPTR, A       ;计数初值低 8 位装入定时器
        INC    DPTR           ;指向定时器高 8 位
        MOV    A, #40H        ;设定时器连续方波输出
        MOVX   @DPTR, A       ;计数初值高 6 位装入定时器
        MOV    DPTR, #7F00H   ;指向命令/状态口
        MOV    A, #0C2H       ;设定命令控制字
        MOVX   @DPTR, A       ;A 口基本输入方式、B 口基本输出方
                                ;式,开启定时器
    
```

例 9-4 读 8155H 的 F1H 单元。

程序如下:

```

        MOV    DPTR, #7EF1H    ;指向 8155H 的 F1H 单元
        MOVX   A, @DPTR        ;F1H 单元内容→A
    
```

例 9-5 将立即数 41H 写入 8155H RAM 的 20H 单元。

程序如下:

```

        MOV    A, #41H        ;立即数→A
        MOV    DPTR, #7E20H    ;指向 8155H 的 20H 单元
        MOVX   @DPTR, A       ;立即数 41H 送到 8155H RAM 的 20H 单元
    
```

在同时需要扩展 RAM 和 I/O 的 MCS-51 应用系统中,选用 8155H 特别经济。8155H 既有 RAM,又有 I/O 口,此外,还有定时器。因此,8155H 芯片是单片机应用系统中常用的外围接口芯片之一。

本节介绍了 8155H 芯片及其工作方式、接口电路和软件编程。还有与之类似芯片如 8156H,该芯片除片选端 \overline{CE} 高电平有效外,其他功能及引脚与 8155H 完全相同。

9.4 用 74LS TTL 电路扩展并行 I/O 口

在 MCS-51 单片机应用系统中,采用 TTL 电路、CMOS 电路锁存器或三态门电路也可构成各种类型的简单输入/输出口,在有些场合能降低成本、缩小体积。通常这种 I/O 都是通过 P0 口扩展。由于 P0 口只能分时复用,故构成输出口时,接口芯片应具有锁存功能;构成输入口时,要求接口芯片应能三态缓冲或锁存选通,数据的输入、输出由单片机的读/写信号控制。

如图 9-20 所示是 1 个利用 74LS273 和 74LS244,将 P0 口扩展成简单的输

入/输出电路。74LS273 是 8D 锁存器,扩展输出口,输出端接 8 个 LED 发光二极管,以显示 8 个按钮开关状态,某位低电平时二极管发光。74LS244 是缓冲驱动器,扩展输入口,它的 8 个输入端分别接 8 个按钮开关。74LS273 和 74LS244 的工作受 8031 的 $P2.0$ 、 \overline{RD} 、 \overline{WR} 3 条控制线控制。

电路的工作原理如下:

当 $P2.0=0$, $\overline{WR}=0$ ($\overline{RD}=1$) 选中 74LS273 芯片, CPU 通过 P0 口输出数据锁存到 74LS273, 74LS273 的输出端低电平位对应的 LED 发光二极管点亮; 当 $P2.0=0$, $\overline{RD}=0$ ($\overline{WR}=1$) 时选中 74LS244, 此时若无按钮开关按下, 输入全为高电平, 但某开关按下时则对应位输入为 0, 74LS244 的输入端不全为 1, 其输入状态通过 P0 口数据线被读入 8031 片内。

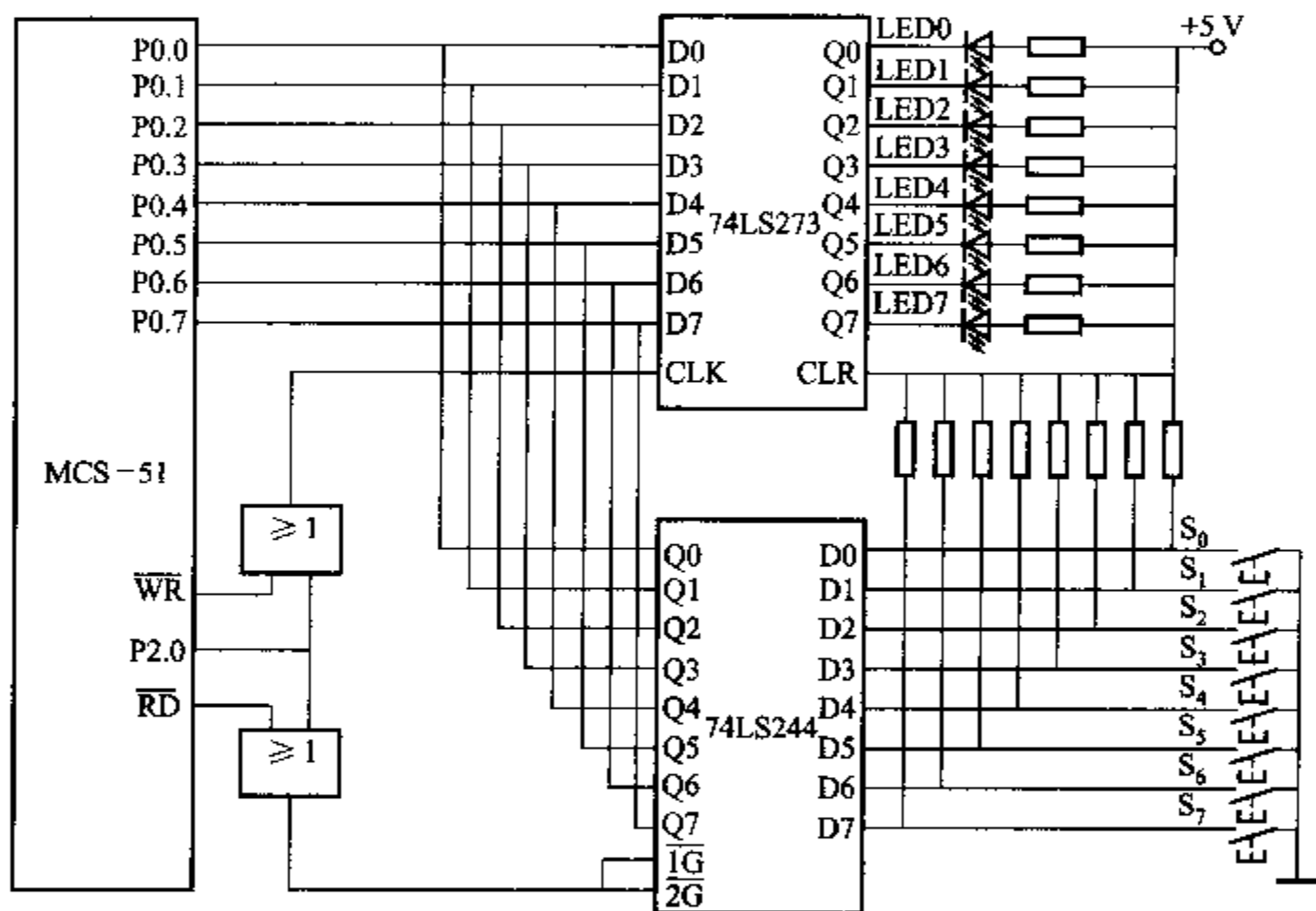


图 9-20 74LS TTL I/O 扩展举例

总之,在图 9-20 中只要保证 $P2.0$ 为 0, 其它地址位或 0 或 1 即可。如地址用 FEFH (无效位全为 1), 或用 0000H (无效位全为 0) 都可。

输出程序段:

```
MOV    A, #data           ;数据→A
MOV    DPTR, #0FEFFH      ;I/O 地址→DPTR
MOVX   @DPTR, A           ; $\overline{WR}$ 为低电平,数据经 74LS273 口输出
```

输入程序段:

```
MOV    DPTR, #0FEFFH      ;I/O 地址→DPTR
```

MOVX A,@DPTR

;RD为低电平,74LS244 接口数据读入内部 RAM

例 9-6 编写程序把按钮开关状态通过图 9-20 中的发光二极管显示出来。

程序:

```
DDIS: MOV DPTR,#0FEFFH      ;输入口地址→DPTR
LP:    MOVX A,@DPTR          ;按钮开关状态读入 A 中
        MOVX @DPTR,A         ;A 数据送显示输出口
        SJMP LP              ;(输入、输出共用同一地址)反复连续
                                ;执行
```

由程序可以看出,对于所扩展接口的输入/输出就像从外部 RAM 读/写数据一样方便。图 9-20 仅仅扩展了两片,如果仍不够用,还可扩展多片 74LS244、74LS273 之类的芯片。但作为输入口时,一定要求有三态功能,否则将影响总线的正常工作。

9.5 用 MCS-51 的串行口扩展并行口

MCS-51 串行口的方式 0 可以用于 I/O 扩展。如果在应用系统中,串行口未被占用,那么将它用来扩展并行 I/O 口既不占用片外的 RAM 地址,又节省硬件开销,是一种经济、实用的方法。

在方式 0 时,串行口为同步移位寄存器工作方式,其波特率是固定的,为 $f_{osc}/12$ (f_{osc} 为系统的振荡器频率)。数据由 RXD 端(P3.0)输入,同步移位时钟由 TXD 端(P3.1)输出。发送、接收的数据是 8 位,低位在先。

9.5.1 用 74LS165 扩展并行输入口

图 9-21 是利用 2 片 74LS165 扩展 2 个 8 位并行输入口的接口电路。

74LS165 是 8 位并行输入串行输出的寄存器。当 74LS165 的 S/\bar{L} 端由高到低跳变时,并行输入端的数据被置入寄存器;当 $S/\bar{L}=1$,且时钟禁止端(15 引脚)为低电平时,允许 TXD(P3.1)移位时钟输入,这时在时钟脉冲的作用下,数据将沿 Q_A 到 Q_B 方向移动。

图 9-21 中,TXD(P3.1)作为移位脉冲输出与所有 74LS165 的移位脉冲输入端 CP 相连;RXD(P3.0)作为串行数据输入端与 74LS165 的串行输出端 Q_H 相连;P1.0 用来控制 74LS165 的移位与并入,同 S/\bar{L} 相连;74LS165 的时钟禁止端(15 引脚)接地,表示允许时钟输入。当扩展多个 8 位数输入口时,相邻两芯片的首

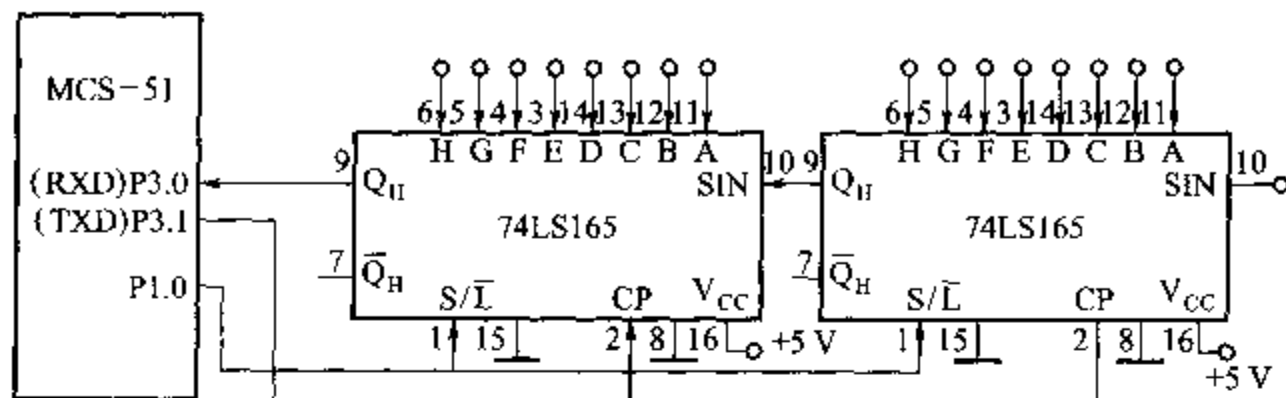


图 9-21 利用 74LS165 扩展并行输入口

尾(Q_{11} 与 SIN)相连。

例 9-7 下面的程序是从 16 位扩展口读入 5 组数据(每组 2 B),并把它们转存到内部 RAM 20H 开始的单元。

```

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

上面的程序对串行接收过程采用的是查询等待的控制方式, 如有必要, 也可改用中断方式。从理论上讲, 按图 9-21 方法扩展的输入口几乎是无限的, 但扩展的越多, 口的操作速度也就越慢。

9.5.2 用 74LS164 扩展并行输出口

74LS164 是 8 位串入并出移位寄存器。图 9-22 是利用 74LS164 扩展 2 个 8 位并行输出口的接口电路。

当 MCS-51 单片机串行口工作在方式 0 的发送状态时, 串行数据由 P3.0 (RXD)送出, 移位时钟由 P3.1(TXD)送出。在移位时钟的作用下, 串行口发送缓冲器的数据一位一位地从 P3.0 移入 74LS164 中。需要指出的是, 由于

74LS164 无并行输出控制端,因而在串行输入过程中,其输出端的状态会不断变化,故在某些应用场合,在 74LS164 的输出端应加接输出三态门控制,以便保证串行输入结束后再输出数据。

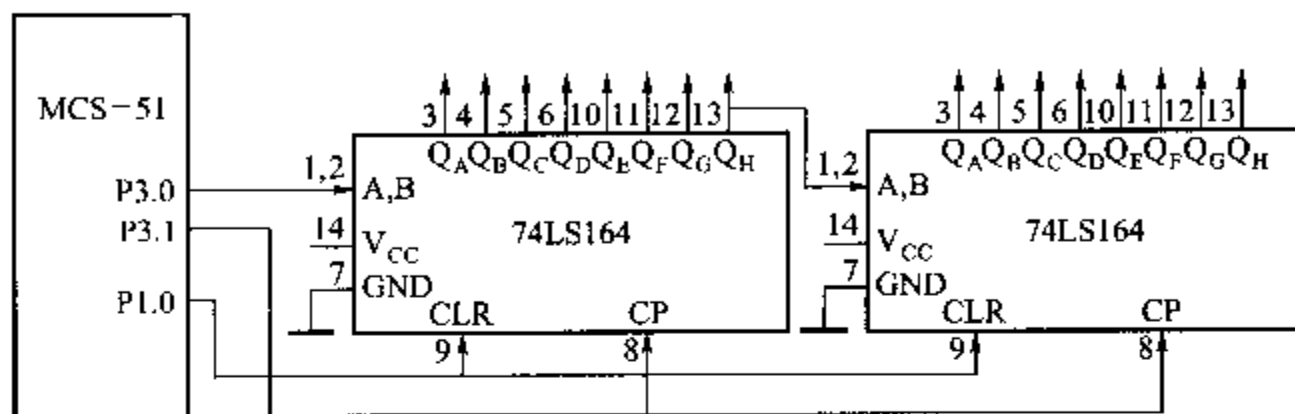


图 9-22 利用 74LS164 扩展并行输出口

例 9-8 下面是将内部 RAM 单元 30H、31H 的内容经串行口由 74LS164 并行输出子程序。

```
START: MOV     R7, #02H           ;设置要发送的字节个数
        MOV     R0, #30H          ;设置地址指针
        MOV     SCON, #00H        ;设置串行口为方式 0
SEND:   MOV     A, @R0
        MOV     SBUF, A           ;启动串行口发送过程
WAIT:   JNB     TI, WAIT          ;1 帧数据未发送完,循环等待
        CLR     TI
        INC     R0                ;取下一个数
        DJNZ    R7, SEND          ;未发送完,继续,发送完从子程序返回
RET
```

思考题及习题

1. I/O 接口和 I/O 端口有什么区别? I/O 接口的功能是什么?
2. 常用的 I/O 端口编址有哪两种方式? 它们各有什么特点? MCS-51 的 I/O 端口编址采用的是哪种方式?
3. I/O 数据传送有哪几种传送方式? 分别在哪些场合下使用?
4. 编写程序,采用 8255A 的 C 口按位置位/复位控制字,将 PC7 置 0,PC4 置 1,(已知 8255A 各端口的地址为 7FFCH~7FFFH)。
5. 8255A 的方式控制字和 C 口按位置位/复位控制字都可以写入 8255A 的同一控制寄存器,8255A 是如何来区分这两个控制字的?
6. 由图 9-6 来说明 8255A 的 A 口在方式 1 的选通输入方式下的工作过程。
7. 8155H 的端口都有哪些? 哪些引脚决定端口的地址? 引脚 TIMERIN 和 TIMEROUT

的作用是什么?

8. 判断下列说法是否正确,为什么?

(1) 由于 8155H 不具有地址锁存功能,因此在与 8031 的接口电路中必须加地址锁存器。

(2) 在 8155H 芯片中,决定端口和 RAM 单元编址的信号线是 AD7~AD0 和 \overline{WR} 。

(3) 8255A 具有三态缓冲器,因此可以直接挂在系统的数据总线上。

(4) 8255A 的 B 口可以设置成方式 2。

9. 现有一片 8031,扩展了一片 8255A,若把 8255A 的 B 口用作输入,B 口的每一位接一个开关,A 口用作输出,每一位接一个发光二极管,请画出电路原理图,并编写出 B 口某一位开关接高电平时,A 口相应位发光二极管被点亮的程序。

10. 假设 8155H 的 TIMERIN 引脚输入的频率为 1 MHz? 问 8155H 的最大定时时间是多少?

11. MCS-51 的并行接口的扩展有多种方法,在什么情况下,采用扩展 8155H 比较适合? 什么情况下,采用扩展 8255A 比较适合?

12. 假设 8155H 的 TIMERIN 引脚输入的脉冲频率为 1 MHz,请编写出在 8155H 的 $\overline{TIMEROUT}$ 引脚上输出周期为 10 ms 的方波的程序。

第 10 章 MCS-51 与键盘、显示器、 拨盘、打印机的接口设计

大多数的 MCS-51 应用系统,都要配置输入外设和输出外设。常用的输入外设有:键盘、BCD 码拨盘等;常用的输出外设有:LED 显示器、LCD 显示器、打印机等。本章介绍 MCS-51 与输入外设、输出外设的接口电路设计以及软件编程。

10.1 LED 显示器接口原理

LED(Light Emitting Diode)是发光二极管英文名称的缩写。LED 显示器是由发光二极管构成的,所以在显示器前面冠以“LED”。LED 显示器在单片机系统中的应用非常普遍。

10.1.1 LED 显示器的结构

常用的 LED 显示器为 8 段(或 7 段,8 段比 7 段多了 1 个小数点“dp”段)。每一个段对应 1 个发光二极管。这种显示器有共阳极和共阴极 2 种。如图 10-1 所示。共阴极 LED 显示器的发光二极管的阴极连接在一起,通常此公共阴极接地。当某个发光二极管的阳极为高电平时,发光二极管点亮,相应的段被显示。同样,共阳极 LED 显示器的发光二极管的阳极连接在一起,通常此公共阳极接正电压,当某个发光二极管的阴极接低电平时,发光二极管被点亮,相应的段被显示。

为了使 LED 显示器显示不同的符号或数字,就要把不同段的发光二极管点亮,这样就要为 LED 显示器提供代码,因为这些代码可使 LED 相应的段发光,从而显示不同字型,因此该代码称之为段码(或称为字型码)。

7 段发光二极管,再加上 1 个小数点位,共计 8 段。因此提供给 LED 显示器的段码(或字型码)正好是 1 B。各段与字节中各位对应关系如下:

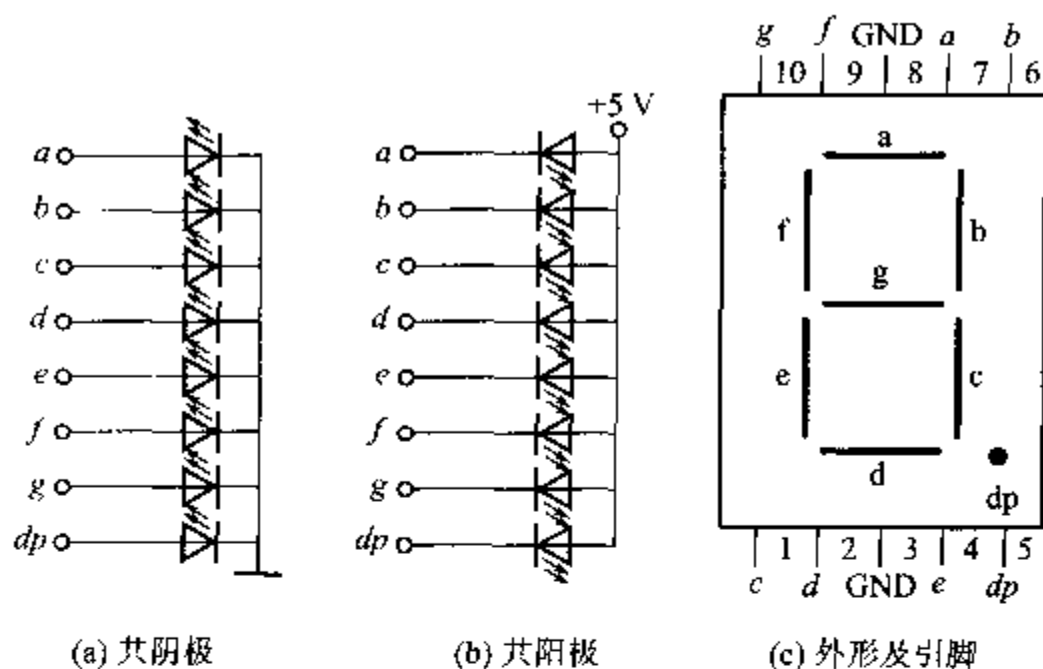


图 10-1 8 段 LED 结构及外形

代码位	D7	D6	D5	D4	D3	D2	D1	D0
显示段	dp	g	f	e	d	c	b	a

按照上述格式,8 段 LED 的段码如表 10-1 所示。

表 10-1 8 段 LED 段码

显示字符	共阴极段码	共阳极段码	显示字符	共阴极段码	共阳极段码
0	3FH	C0H	C	39H	C6H
1	06H	F9H	D	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	8EH
4	66H	99H	P	73H	8CH
5	6DH	92H	U	3EH	C1H
6	7DH	82H	T	31H	CEH
7	07H	F8H	y	6EH	91H
8	7FH	80H	H	76H	89H
9	6FH	90H	L	38H	C7H
A	77FH	88H	“灭”	00H	FFH
B	7CH	83H

表 10-1 只列出了部分段码,读者可以根据实际情况选用。另外,段码是相对的,它由各字段在字节中所处的位决定。例如表 10-1 中 8 段 LED 段码是按格式:

dp	g	f	e	d	c	b	a
----	---	---	---	---	---	---	---

而形成的,对于“0”的段码为 3FH(共阴)。如果将段码改为下列格式:

dp	a	b	c	d	e	f	g
----	---	---	---	---	---	---	---

则字符“0”的段码变为:7EH(共阴)。总之,字型及段码可由设计者自行设定,不必拘于表 10-1 的形式。但一般习惯上还是以“a”段对应段码的最低位。

10.1.2 LED 显示器工作原理

由 N 个 LED 显示块可拼接成 N 位的 LED 显示器。图 10-2 是 4 位的 LED 显示器的结构原理图。

N 个 LED 显示块有 N 位位选线和 $8 \times N$ 根段码线。段码线控制显示字符的字型,而位选线为各个 LED 显示块中各段的公共端,它控制该 LED 显示位的亮或暗。

LED 显示器有静态显示和动态显示 2 种显示方式。

1: LED 静态显示方式

LED 显示器工作于静态显示方式时,各位的共阴极(或共阳极)连接在一起并接地(或+5V);每位的段码线(a~dp)分别与 1 个 8 位的锁存器输出相连。之所以称为静态显示,是因为各个 LED 的显示字符一经确定,相应锁存器锁存的段码输出将维持不变,直到送入另一个字符的段码为止。正因为如此,静态显示器的亮度都较高。

图 10-3 所示为 1 个 4 位静态 LED 显示器电路。该电路各位可独立显示,只要在该位的段码线上保持段码电平,该位就能保持相应的显示字符。由于各

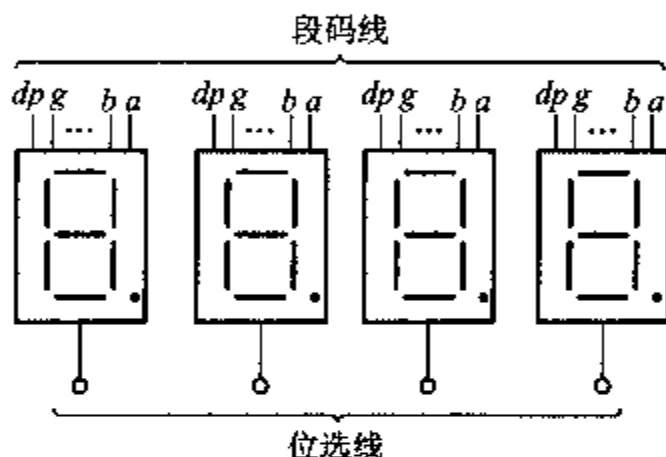


图 10-2 4 位 LED 显示器的构成

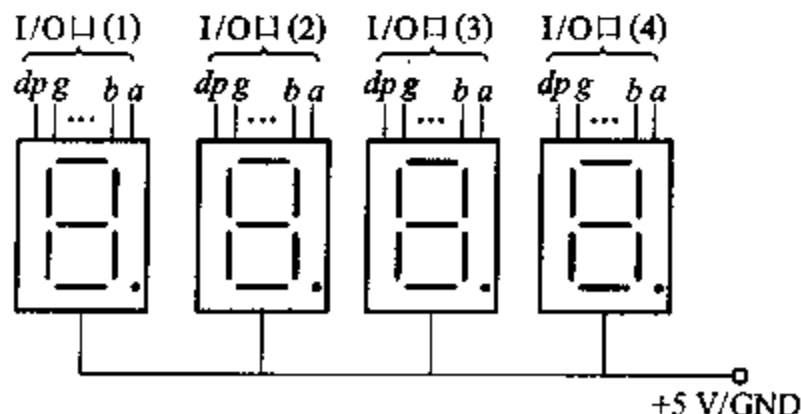


图 10-3 4 位静态 LED 显示器电路

位分别由 1 个 8 位的数据输出口(例如 8255A 的 A、B、C 口)控制段码线,故在同一时间里,每一位显示的字符可以各不相同。这种显示方式接口编程容易,付出的代价是占用口线较多。如图 10-3 电路所示,若用 I/O 口线接口,则要占用 4 个 8 位 I/O 口,若用锁存器(如 74LS373)接口,则要用 4 片 74LS373 芯片。如果显示器的位数增多,则需要增加锁存器。因此在显示位数较多的情况下,一般都采用动态显示方式。

2. LED 动态显示方式

在多位 LED 显示时,为简化硬件电路,通常将所有位的段码线相应段并联在一起,由 1 个 8 位 I/O 口控制,形成段码线的多路复用,而各位的共阳极或共阴极分别由相应的 I/O 线控制,形成各位的分时选通。图 10-4 所示为 1 个 4 位 8 段 LED 动态显示器电路。其中段码线占用 1 个 8 位 I/O 口,而位选线占用 1 个 4 位 I/O 口。由于各位的段码线并联,8 位 I/O 口输出的段码对各个显示位来说都是相同的。因此,在同一时刻,如果各位位选线都处于选通状态的话,4 位 LED 将显示相同的字符。若要各位 LED 能够同时显示出与本位相应的显示字符,就必须采用动态显示方式,即在某一时刻,只让某一位的位选线处于选通状态,而其他各位的位选线处于关闭状态,同时,段码线上输出相应位要显示的字符的段码。这样,在同一时刻,4 位 LED 中只有选通的那 1 位显示出字符,而其他 3 位则是熄灭的。同样,在下一时刻,只让下一位的位选线处于选通状态,而其他各位的位选线处于关闭状态,在段码线上输出将要显示字符的段码,则同一时刻,只有选通位显示出相应的字符,而其他各位则是熄灭的。如此循环下去,就可以使各位显示出将要显示的字符。虽然这些字符是在不同时刻出现的,而在同一时刻,只有一位显示,其他各位熄灭,但由于 LED 显示器的余辉和人眼的视觉暂留作用,只要每位显示间隔足够短,则可以造成多位同时亮的假象,达到同时显示的效果。

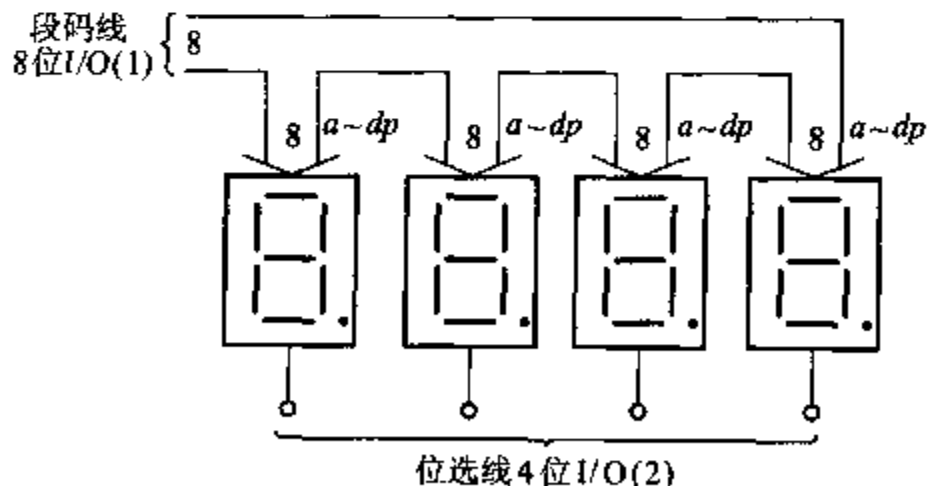


图 10-4 4 位 8 段 LED 动态显示电路

LED 不同位显示的时间间隔应根据实际情况而定。发光二极管从导通到发

光有一定的延时,导通时间太短,则发光太弱,人眼无法看清;但也不能太长,因为要受限于临界闪烁频率,而且此时间越长,占用 CPU 时间也越多。另外,显示位数增多,也将占用大量的 CPU 时间,因此动态显示的实质是以牺牲 CPU 时间来换取器件的减少。

图 10-5 给出了 8 位 LED 动态显示 2003.10 过程。图(a)是显示过程,某一时刻,只有 1 位 LED 被选通显示,其余位则是熄灭的;图(b)是实际的显示结果,人眼看到的是 8 位稳定的同时显示的字符。

显示字符	段 码	位显码	显示器显示状态(微观)	位选通时序
0	3FH	FEH		
1	06H	FDH		
0	BFH	FBH		
1	06H	F7H		
3	CFH	EFH		
0	3FH	DFH		
0	3FH	BFH		
2	5BH	7FH		

2	0	0	3	1	0	1	0
---	---	---	---	---	---	---	---

(a) 8 位 LED 动态显示过程

(b) 人眼看到的结果

图 10-5 8 位 LED 动态显示过程和结果

10.2 键盘接口原理

键盘在单片机应用系统中能实现向单片机输入数据、传送命令等功能,是人工干预单片机的主要手段。下面介绍键盘的工作原理,键盘按键的识别过程及识别方法,键盘与单片机的接口技术和编程。

1. 键盘输入的特点

键盘实质上是一组按键开关的集合。通常,键盘开关利用了机械触点的合、断作用。一个电压信号通过键盘开关机械触点的断开、闭合,其行线电压输出波形如图 10-6 所示。

图 10-6 中 t_1 和 t_3 分别为键的闭合和断开过程中的抖动期(呈现一串负脉冲),抖动时间长短和开关的机械特性有关,一般为 5~10 ms, t_2 为稳定的闭合期,其时间由按键动作所确定,一般为十分之几秒到几秒, t_0 、 t_4 为断开期。

2. 按键的确认

键的闭合与否,反映在行线输出电压上就是呈现高电平或低电平,如果高电平表示键断开,低电平则表示键闭合,通过对行线电平高低状态的检测,便可确认

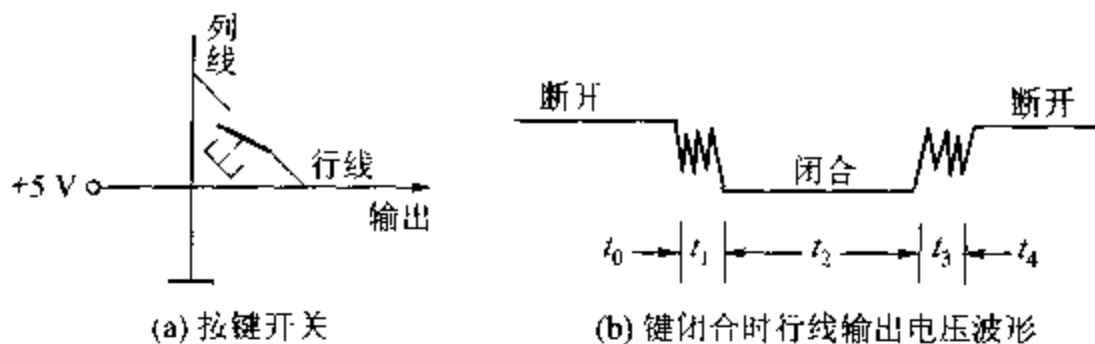


图 10-6 键盘开关及其波形

按键按下与否。为了确保 CPU 对一次按键动作只确认一次按键有效,必须消除抖动期 t_1 和 t_3 的影响。下面将介绍消除抖动的措施。

3. 如何消除按键的抖动

常采用软件来消除按键抖动。

采用软件来消除按键抖动的基本思想是:在第一次检测到有键按下时,该键所对应的行线为低电平,执行一段延时 10 ms 的子程序后,确认该行线电平是否仍为低电平,如果仍为电平,则确认为该行确实有键按下。当按键松开时,行线的低电平变为高电平,执行一段延时 10 ms 的子程序后,检测该行线为高电平,说明按键确实已经松开。采取以上措施,躲开了两个抖动期 t_1 和 t_3 ,从而消除了按键抖动的影响。

10.2.1 键盘接口的工作原理

常用键盘接口分为独立式键盘接口和行列式键盘接口。

1. 独立式键盘接口

独立式键盘就是各键相互独立,每个按键各接一根输入线,通过检测输入线的电平状态可以很容易的判断哪个按键被按下。

在按键数目较多时,独立式键盘电路需要较多的输入口线且电路结构繁杂,故此种键盘适用于按键较少或操作速度较高的场合。下面介绍几种独立式键盘的接口。

图 10-7 中(a)为中断方式的独立式键盘工作电路,只要有一个键按下,与门的输出即为低电平,向 8031 发出中断请求,在中断服务程序中,对按下的键进行识别。图(b)为查询方式的独立式键盘工作电路,按键直接与 8031 的 I/O 口线相接,通过读 I/O 口,判断各 I/O 口线的电平状态,即可以识别出按下的键。

此外,也可以用扩展的 I/O 口作为独立式键盘接口电路,图 10-8 为采用 8255A 扩展 I/O 口,图 10-9 为用三态缓冲器扩展的 I/O 口。这两种接口电路,都是把按键当作外部 RAM 某一工作单元的位来对待,通过读片外 RAM 的方法,识别按键的状态。

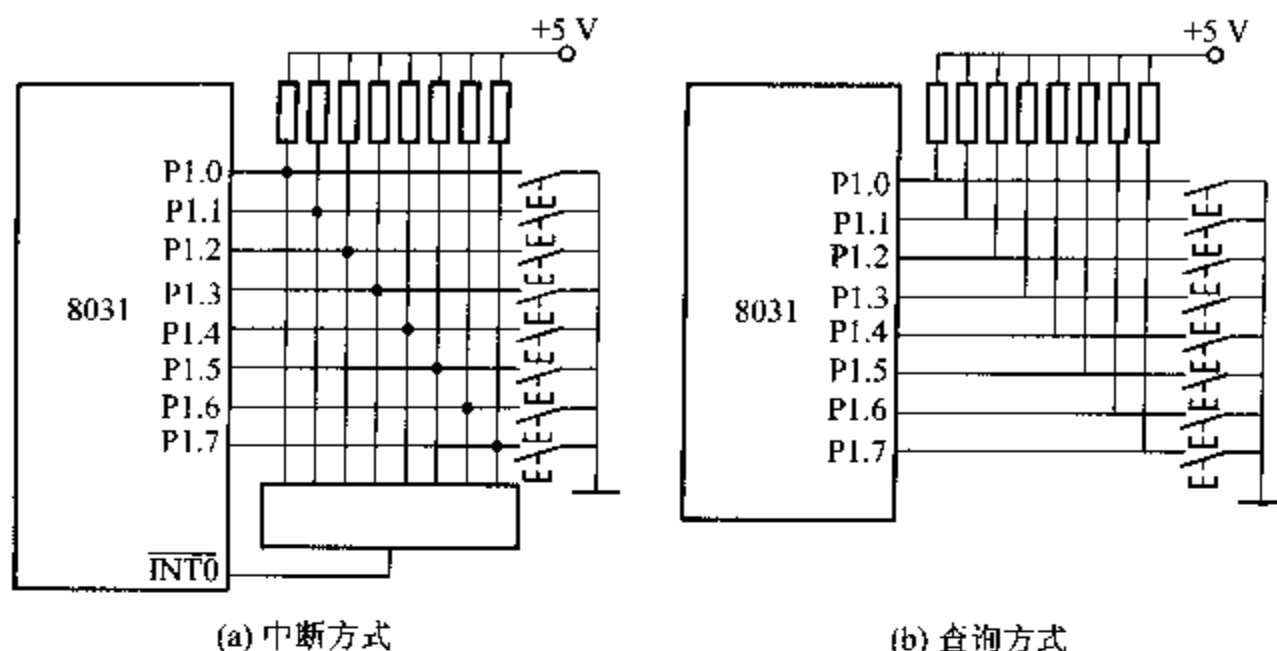


图 10-7 独立式键盘接口电路

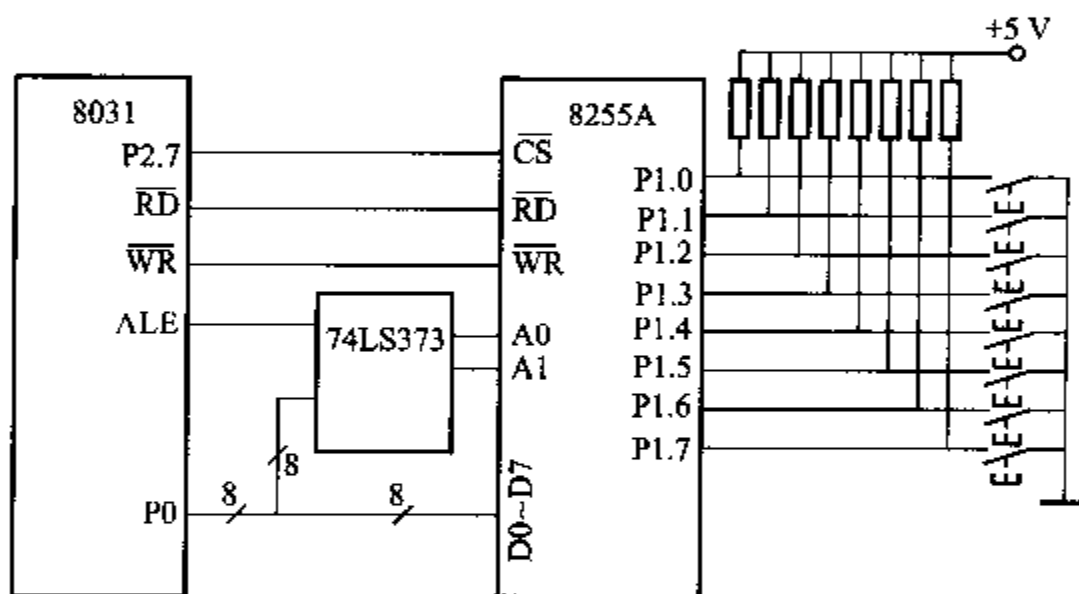


图 10-8 通过 8255A 扩展的独立式键盘接口

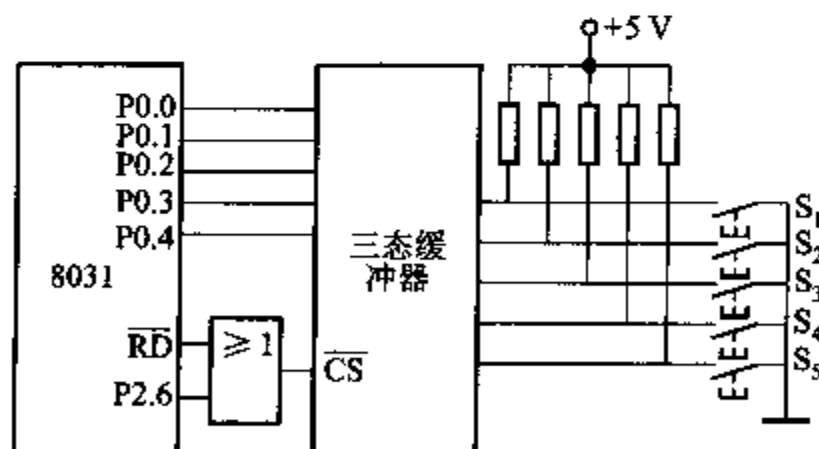


图 10-9 采用三态缓冲器的独立式键盘接口

上述各种独立式键盘电路中,各按键均采用了上拉电阻,这是为了保证在按键断开时,各 I/O 口有确定的高电平,当然如果输入口线内部已有上拉电阻,则外电路的上拉电阻可省去。

现在对图 10-9 所示的独立式键盘进行软件编程,采用软件消抖的方法,以查询工作方式检测各按键的状态。当有且仅有一键按下时才予以识别,如有两个或多个键同时按下将不予以处理。

程序如下:

```

KEYIN:  MOV    DPTR, # BFFFH      ;键盘地址 BFFFH
        MOVX   A, @DPTR          ;读键盘状态
        ANL    A, # 1FH          ;屏蔽高 3 位
        MOV    R3, A             ;保存键盘状态值
        LCALL  DELAY10           ;调用延时 10ms 子程序,软件去键
                                   ;盘抖动
        MOVX   A, @DPTR          ;再读键盘状态
        ANL    A, # 1FH          ;屏蔽高 3 位
        CJNE   A, R3, RETURN      ;2 次结果不一样,说明是抖动引起
                                   ;的,转 RETURN
        CJNE   A, # 1FH, KEY2     ;2 次结果一样,有键按下, S1 键未
                                   ;按下,转 KEY2
        LJMP   PKEY1             ;是 S1 键按下,转 S1 键处理子程序
                                   ;PKEY1
KEY2:   CJNE   A, # 1DH, KEY3     ;S2 键未按下,转 KEY3
        LJMP   PKEY2             ;S2 键按下,转 PKEY2 处理
KEY3:   CJNE   A, # 1BH, KEY4     ;S3 未按下,转 KEY4
        LJMP   PKEY3             ;S3 按下,转 PKEY3 处理
KEY4:   CJNE   A, # 17H, KEY5     ;S4 键未按下,转 KEY5
        LJMP   PKEY4             ;S4 按下,转 PKEY4 处理
KEY5:   CJNE   A, # 0FH, PASS     ;S5 未按下,转 RETURN
        LJMP   PKEY5             ;S5 按下,转 PKEY5 处理
        RETURN;RET               ;重键或无键按下,从子程序返回

```

软件延时 10 ms 子程序 DELAY10 的编写,可参见第 4 章,此处略。

由上可见,独立式键盘的识别和编程非常简单,常用在按键数目较少的场合。

2. 行列式键盘接口

行列式(也称矩阵式)键盘用于按键数目较多的场合,它由行线和列线组成,按键位于行、列的交叉点上。如图 10-10 所示,1 个 3×3 的行、列结构可以构成 1 个具有 9 个按键的键盘。同理 1 个 4×4 的行、列结构可以构成 1 个 16 个按键的键盘等等。很明显,在按键数目较多的场合,行列式键盘与独立式键盘相比,要节省很多的 I/O 口线。

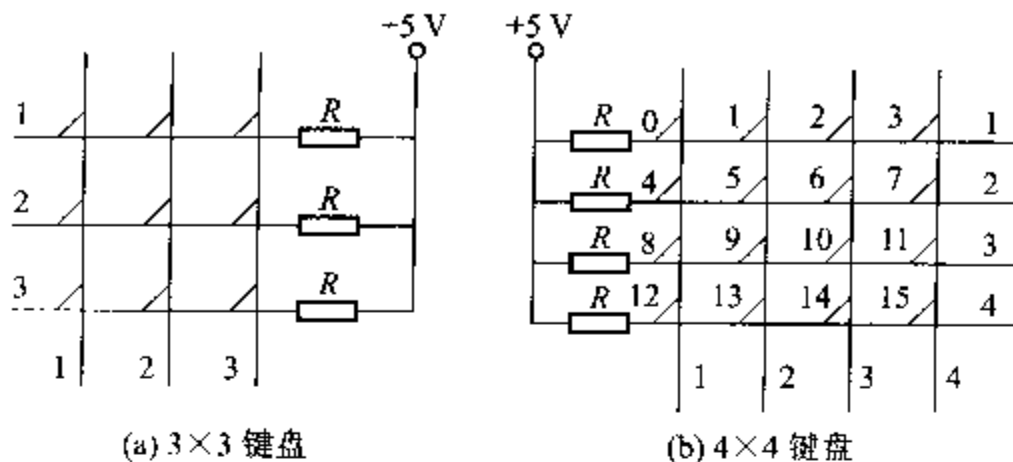


图 10-10 行列式键盘结构

(1) 行列式键盘工作原理

按键设置在行、列线交点上,行、列线分别连接到按键开关的两端。行线通过上拉电阻接到+5V上。无按键按下时,行线处于高电平状态,而当有按键按下时,行线电平状态将由与此行线相连的列线的电平决定。列线的电平如果为低,则行线电平为低;列线的电平如果为高,则行线的电平亦为高。这一点是识别行列式键盘按键是否按下的关键所在。由于行列式键盘中行、列线为多键共用,各按键均影响该键所在行和列的电平。因此各按键彼此将相互发生影响,所以必须将行、列线信号配合起来并作适当的处理,才能确定闭合键的位置。

(2) 按键的识别方法

① 扫描法

下面以图 10-10(b)中 3 号键被按下为例,来说明此键是如何被识别出来的。

当 3 号键被按下时,与 3 号键相连的行线电平将由与此键相连的列线电平决定,而行线电平在无按键按下时处于高电平状态。如果让所有的列线处于低电平,很明显,按键所在行电平将被接成低电平,根据此行电平的变化,便能判定此行一定有键被按下。但还不能确定是键 3 被按下,因为如果键 3 不被按下,而同一行的键 2、1 或 0 之一被按下,均会产生同样的效果。所以,行线处于低电平只能得出某行有键被按下的结论。为进一步判定到底是哪一列的键被按下,可采用扫描法来识别。即在某一时刻只让 1 条列线处于低电平,其余所有列线处于高电平。当第 1 列为低电平,其余各列为高电平时,因为是键 3 被按下,所以第 1 行仍处于高电平状态;而当第 2 列为低电平,而其余各列为高电平时,同样我们会发现第 1 行仍处于高电平状态;直到让第 4 列为低电平,其余各列为高电平时,因为此时 3 号键被按下,所以第 1 行的电平将由高电平转换到第 4 列所处的低电平,据此可判断第 1 行第 4 列交叉点处的按键,即 3 号键被按下。

根据上面的分析,很容易想到识别键盘有无键被按下的方法,此方法分2步进行:第1步,识别键盘有无键被按下;第2步,如有键被按下,识别出具体的按键。分别介绍如下:

首先把所有的列线均置为低电平,检查各行线电平是否有变化,如果有变化,则说明有键被按下,如果没有变化,则说明无键被按下。

上述识别具体按键的方法也称为扫描法,即先把某一列置低电平,其余各列置为高电平,检查各行线电平的变化,如果某行线电平为低电平,则可确定此行此列交叉点处的按键被按下。

② 线反转法

扫描法要逐列扫描查询,当被按下的键处于最后1列时,则要经过多次扫描才能最后获得此按键所处的行列值。而线反转法则显得很简练,无论被按键是处于第1列或最后1列,均只需经过2步便能获得此按键所在的行列值,线反转法的原理如图10-11所示。

图中用1个8位I/O口构成1个4×4的矩阵键盘,采用查询方式进行工作,下面介绍线反转法的2个具体操作步骤:

第1步,让行线编程为输入线,列线编程为输出线,并使输出线输出为全低电平,则行线中电平由高变低的所在行为按键所在行。

第2步,再把行线编程为输出线,列线编程为输入线,并使输出线输出为全低电平,则列线中电平由高变低所在列为按键所在列。

结合上述2步的结果,可确定按键所在行和列,从而识别出所按的键。

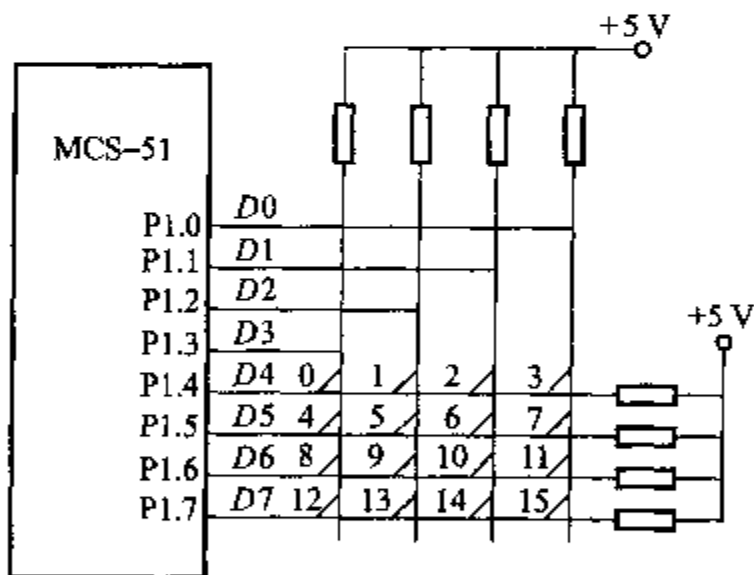


图 10-11 线反转法原理图

假设3号键被按下,那么第1步即在D0~D3输出全为0,然后读入D4~D7位,结果D4=0,而D5、D6和D7均为1,因此,第1行出现电平的变化,说明第1行有键按下;第2步让D4~D7输出全为0,然后读入D0~D3位,结果D0=0,而D1、D2和D3均为1,因此第4列出现电平的变化,说明第4列有键按下。综合上述分析,即第1行第4列按键被按下,此按键即是3号键。因此线反转法非常简单适用。当然实际编程中要考虑采用软件延时进行消抖处理。

(3) 键盘的编码

对于独立式键盘,由于按键的数目比较少,可根据实际需要灵活编码。对于行列式键盘,按键的位置由行号和列号惟一确定,所以常常采用依次排列键号的

方式对键盘进行编码。以 4×4 键盘为例,键号可以编码为 01H,02H,03H,...,0EH,0FH,10H 共 16 个。

10.2.2 键盘的工作方式

单片机应用系统中,键盘扫描只是单片机的工作内容之一。单片机在忙于各项工作任务时,如何兼顾键盘的输入,取决于键盘的工作方式。键盘工作方式的选取应根据实际应用系统中 CPU 工作的忙、闲情况而定。其原则是既要保证能及时响应按键操作,又不要过多占用 CPU 的工作时间。通常,键盘工作方式有 3 种,即编程扫描、定时扫描和中断扫描。

1. 编程扫描方式

这种方式就是只有当单片机空闲时,才调用键盘扫描子程序,反复的扫描键盘,等待用户从键盘上输入命令或数据,来响应键盘的输入请求。图 10-12 为 1 个 4×8 矩阵键盘通过 8255A 扩展 I/O 口与 8031 的接口电路原理图,键盘采用编程扫描方式工作,8255A 的 PC 口低 4 位输出逐行扫描信号,PA 口输入 8 位列信号,均为低电平有效。8255A 的 A0,A1 端分别接于地址线 A0,A1 上, \overline{CS} 与 P2.7 相接, \overline{WR} 、 \overline{RD} 分别与 8031 的 \overline{WR} 和 \overline{RD} 相连。

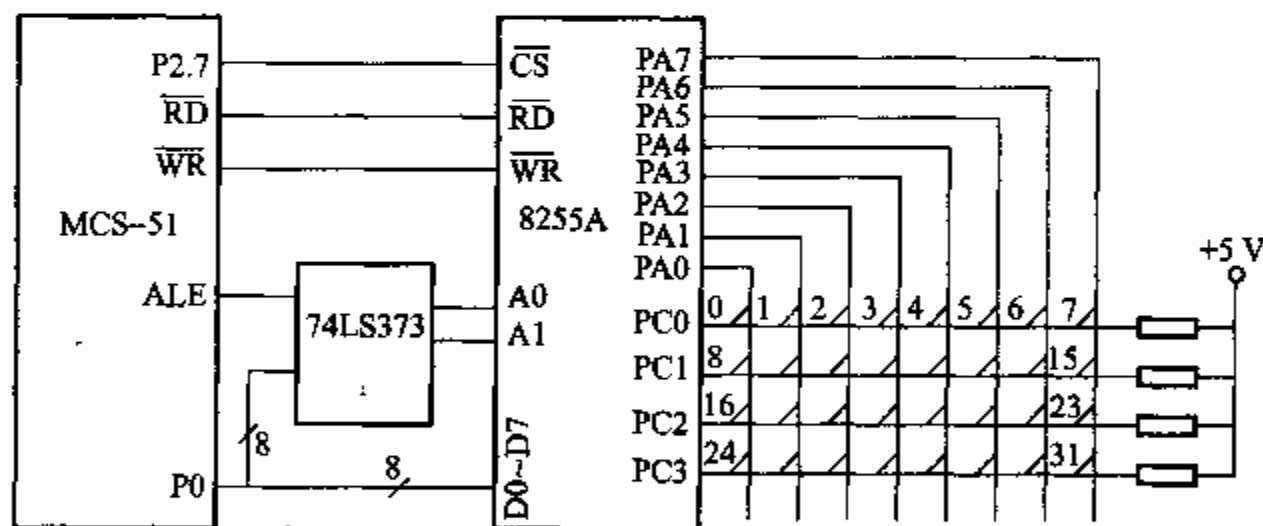


图 10-12 采用扩展 8255A I/O 组成的 4×8 矩阵式键盘

根据图 10-12 可确定 8255A 各端口地址为:

端 口	地 址
PA 口	7F00H
PC 口	7F02H
控制寄存器	7F03H

用写入控制寄存器的方式命令控制字来设置 PA 口工作于方式 0 输出,PC 口低 4 位工作于方式 0 输入。编程扫描工作方式的工作过程如下:

(1) 在键盘扫描子程序中,首先判断键盘上是否有键按下。其方法为 PA 口 8 位输出全 0,读 PC 口低 4 位状态,若 PC0~PC3 为全 1,则说明键盘无键按下;若不全为 1,则说明键盘可能有键按下。

(2) 用软件延时 10 ms 来消除按键抖动的影响。确实有键按下时,进行下一步。

(3) 求按下键的键号。根据前面介绍的扫描法,逐列置 0 扫描,读入行线的状态,最后确定按键位置。

(1) 等待按键释放后,再进行按键功能的处理操作。

2. 定时扫描工作方式

单片机对键盘的扫描也可采用定时扫描方式,即每隔一定的时间对键盘扫描一次。

在这种扫描方式中,通常利用单片机内的定时器,产生 10 ms 的定时中断,CPU 响应定时器溢出中断请求,对键盘进行扫描,在有键按下时识别出该键,并执行相应键的处理功能程序。

3. 中断工作方式

为进一步提高单片机扫描键盘的工作效率,可采用中断扫描方式,即只有在键盘有键按下时,才执行键盘扫描程序并执行该按键功能程序,如果无键按下,单片机将不理睬键盘。

至此,我们可把键盘所做的工作分为 3 个层次,如图 10-13 所示。

第 1 层:单片机如何来监视键盘的输入。体现在键盘的工作方式上就是:①编程扫描工作方式;②定时扫描工作方式;③中断扫描工作方式。

第 2 层:确定具体按键的键号。体现在按键的识别方法上就是:①扫描法;②线反转法。

第 3 层:实现按键的功能,执行键处理程序。

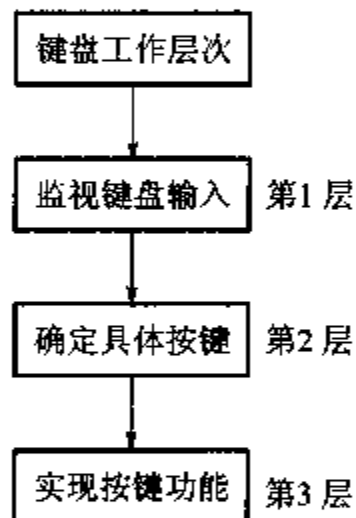


图 10-13 键盘的工作层次

10.3 键盘/显示器接口设计实例

在单片机应用系统设计中,一般都是把键盘和显示器放在一起考虑。下面介绍几种实用的键盘/显示器接口的设计方案。

10.3.1 利用并行 I/O 芯片 8155H 实现键盘/显示器接口

图 10-14 是 8031 单片机用扩展 I/O 接口芯片 8155H 实现的 6 位 LED 显示和 32 键的键盘 显示器接口电路。图中的 8155H 也可用 8255A 来替代。

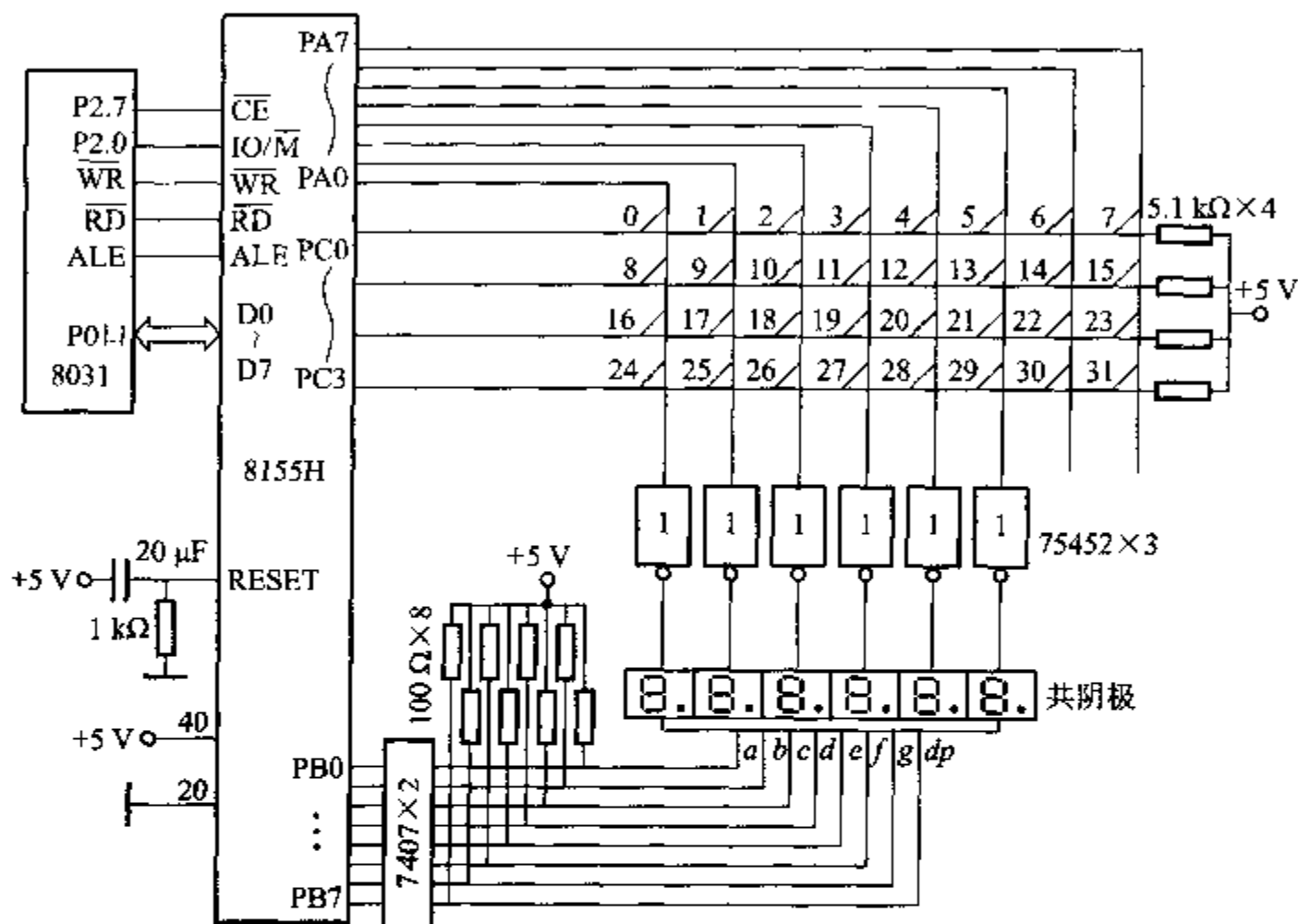


图 10-14 键盘/显示器接口电路

8031 外扩 1 片 8155H，8155H 的 RAM 地址为 7E00H~7EFFH，I/O 口地址为 7F00H~7F05H。8155H 的 PA 口为输出口，控制键盘列线的扫描，PA 口同时又是 6 位共阴极显示器的位扫描口。PB 口作为显示器的段码（字型码）口，8155H 的 PC 口作为键盘的行线状态的输入口，故称为键输入口。图中 75452 为反相驱动器，7407 为同相驱动器。

1. 动态显示程序设计

图 10-14 中的 6 位显示器采用动态显示的方式。在 8031 内部 RAM 中设置 6 个显示缓冲单元 79H~7EH，分别存放显示器要显示的 6 位数据。8155H 的 PA 口扫描输出总是只有 1 位高电平，经 75452 反相后，即显示器的 6 位中仅有 1 位公共阴极为低电平，其他位为高电平。8155H 的 PB 口输出相应位的显示数据的段码，使某一位显示某一字符，其他位为暗。依次改变 PA 口输出为高的位，PB 口输出对应的段码，显示器的 6 位就动态地显示出由缓冲区中显示数据所确定的字符。显示程序的流程如图 10-15 所示。参考程序如下：

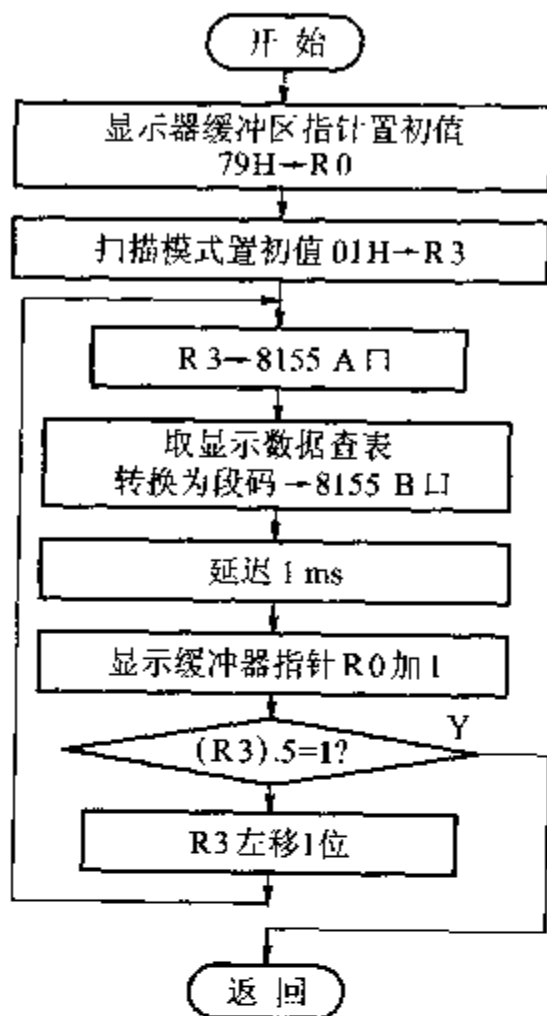


图 10-15 显示子程序流程图

DIR:	MOV	R0, #79H	:置缓冲器指针初值
	MOV	R3, #01H	:位选码的初值送 R3
	MOV	A, R3	
LD0:	MOV	DPTR, #7F01H	:位选码→8155H PA口(PA0位)最 :左边 LED 亮
	MOV	@DPTR, A	
	INC	DPTR	:数据指针指向 PB口
	MOV	A, @R0	:显示数据→A
	ADD	A, #0DH	:加偏移量(下条指令到表首间所有 :指令占的单元数)
	MOVC	A, @A+PC	:根据显示数据来查表取段码
DIR1:	MOVX	@DPTR, A	:段码→8155H 的 PB口
	ACALL	DL1 ms	:延时 1 ms, 即该位显示 1 ms
	INC	R0	:显示数据缓冲区指针指向下一个 :数据单元
	MOV	A, R3	:位选码送入 A 中
	JB	Acc.5, LD1	:判断是否扫描到最右边的 LED, 如 :到最右边, 则返回
	RL	A	:位选码向左移一位, 准备让右边的

```

                                ; 下一位 LED 亮
                                ; 位选码送 R3 中保存
MOV    R3,A
AJMP   LD0
LD0:   RET
DSEG:  DB    3FH,06H,5BH,4FH,66H,6DH ; 共阴极 LED 段码表
        DB    7DH,07H,7FH,6FH,77H,7CH
        DB    39H,5EH,79H,71H,73H,3EH
        DB    31H,6EH,1CH,23H,40H,03H
        DB    18H,00
DL1ms: MOV    R7,#02H           ; 延时 1 ms 子程序
DL:     MOV    R6,#0FFH
DL6:    DJNZ   R6,DL6
        DJNZ   R7,DL
        RET

```

程序中的 ADD A, #0DH 指令中的“0DH”为偏移量(即为查表指令下一条指令到表首地址标号 DESG 之间所有指令所占单元之和),在显示数据的基础上加上偏移量,可查到该显示数据所对应的段码。

2. 键盘程序设计

键盘采用编程扫描工作方式。

键盘程序的功能有以下 4 个方面:

(1) 判别键盘上有无键闭合,其方法为扫描口 PA0~PA7,输出全 0,读 PC 口的状态,若 PC0~PC3 为全 1(键盘上行线全为高电平),则键盘上没有闭合键,若 PC0~PC3 不全为 1,则有键处于闭合状态。

(2) 去除键的机械抖动,其方法为判别出键盘上有键闭合后,延迟一段时间再判别键盘的状态,若仍有键闭合,则认为键盘上有 1 个键处于稳定的闭合期,否则认为是键的抖动。

(3) 判别闭合键的键号,方法为对键盘的列线进行逐列扫描,扫描口 PA0~PA7 依次输出下列编码,即只有 1 列为低电平,其余各列为高电平:

PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	1
1	1	1	1	1	0	1	1
...							
...							
1	0	1	1	1	1	1	1
0	1	1	1	1	1	1	1

相应地依次读 PC 口的状态,若 PC0~PC3 全为 1,则列线为 0 的这 1 列上没有键闭合。闭合键的键号等于为低电平的列号加上行线为低电平的行的首键号。例如:PA 口输出为 11111101 时,读出 PC0~PC3 为 1101,则第 1 行第 1 列相交的键处于闭合状态,第 1 列的首键号为 8,列号为 1。因此,闭合键的键号 N 为:

$$N - \text{行首键号} + \text{列号} = 8 + 1 = 9$$

(4) 使 CPU 对键的 1 次闭合仅作 1 次处理,采用的方法为等待闭合键释放以后再作处理。

键盘程序的流程如图 10-16 所示。采用前述的显示子程序作为延迟子程序,其优点是在进入键盘子程序后,显示器始终是亮的。

键盘子程序如下:

KEYI:	ACALL	KS1	;调用判有无键闭合子程序
	JNZ	LK1	;有键闭合,跳 LK1
NI:	ACALL	DIR	;无键闭合,调用显示子程序,延迟 6 ms
			;后,跳 KEYI
	AJMP	KEYI	
LK1:	ACALL	DIR	;可能有键闭合,延迟 12 ms,软件去抖动
	ACALL	DIR	
	ACALL	KS1	;调用判有无键闭合子程序
	JNZ	LK2	;经去抖动,判键确实闭合,跳 LK2 去处理
	ACALL	DIR	;调用显示子程序延迟 6 ms
	AJMP	KEYI	;抖动引起,跳 KEYI
LK2:	MOV	R2, #0FEH	;列选码→R2
	MOV	R4, #00H	;R4 为列号计数器
LK4:	MOV	DPTR, #7F01H	;列选码→8155H 的 PA 口
	MOV	A, R2	
	MOVX	@DPTR, A	
	INC	DPTR	;数据指针增 2,指向 PC 口
	INC	DPTR	
	MOVX	A, @DPTR	;读 8155H PC 口
	JB	Acc. 0, LONE	;第 0 行线为高,无键闭合,跳 LONE,转判

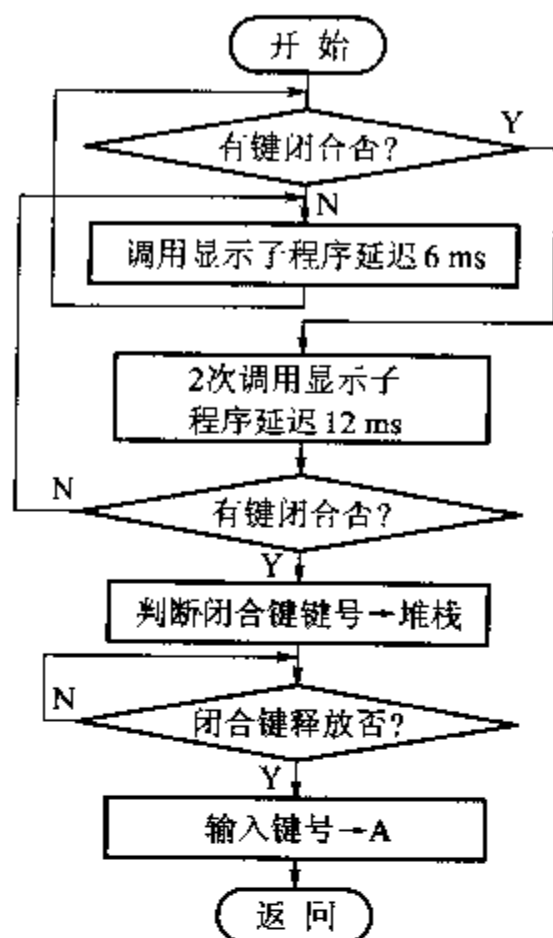


图 10-16 键盘子程序流程图

```

;第 1 行
MOV A, #00H ;第 0 行有键闭合,首键号 0→A
AJMP LKP ;跳 LKP,计算键号
LONE: JB ACC.1,LTW0 ;1 行线为高,无键闭合,跳 LTW0,转判
;2 行
MOV A, #08H ;1 行有键闭合,首键号 8→A
AJMP LKP ;跳 LKP,计算键号
LTW0: JB ACC.2,LTHR ;2 行线为高,无键闭合,跳 LTHR,转判
;3 行
MOV A, #10H ;2 行有键闭合,首键号 10H→A
AJMP LKP ;跳 LKP,计算键号
LTHR: JB ACC.3,NEXT ;3 行线为高,无键闭合,跳 NEXT,准备下
;一列扫描
MOV A, #18H ;3 行有键闭合,首键号 18H→A
LKP: ADD A,R4 ;计算键号,即:行首键号+列号=键号
PUSH A ;键号进栈保护
LK3: ACALL DIR ;调用显示子程序,延时 6 ms
ACALL KS1 ;调用判有无键闭合子程序,延时 6 ms
JNZ LK3 ;判键释放否,未释放,则循环
POP A ;键已释放,键号出栈→A
RET
NEXT: INC R4 ;列计数器加 1,为下一列扫描作准备
MOV A,R2 ;判是否已扫到最后 一列(最右--列)
JNB ACC.7,KND ;键扫描已扫到最后一列,跳 KND,重新进
;行整个键盘扫描
RL A ;键扫描未扫到最后一列,位选码左移一位
MOV R2,A ;位选码→R2
AJMP LK4
KND: AJMP KEY1
KS1: MOV DPTR, #7F01H ;判有无键闭合子程序,全 0→扫描口(PA
;口)
MOV A, #00H ;即列线全为低电平
MOVX @DPTR,A
INC DPTR ;DPTR 增 2,指向 PC 口
INC DPTR ;
MOVX A,@DPTR ;从 PC 口读行线的状态
CPL A ;行线状态取反,如无键按下,则 A 中内容
;为零

```

ANL A, #0FH :屏蔽无用的高4位
RET

10.3.2 利用 8031 的串行口实现键盘/显示器接口

当 8031 的串行口未作它用时,可使用 8031 的串行口来外扩键盘/显示器。应用 8031 的串行口方式 0 的输出方式,在串行口外接移位寄存器 74LS164,构成键盘/显示器接口,其硬件接口电路如图 10-17 所示。

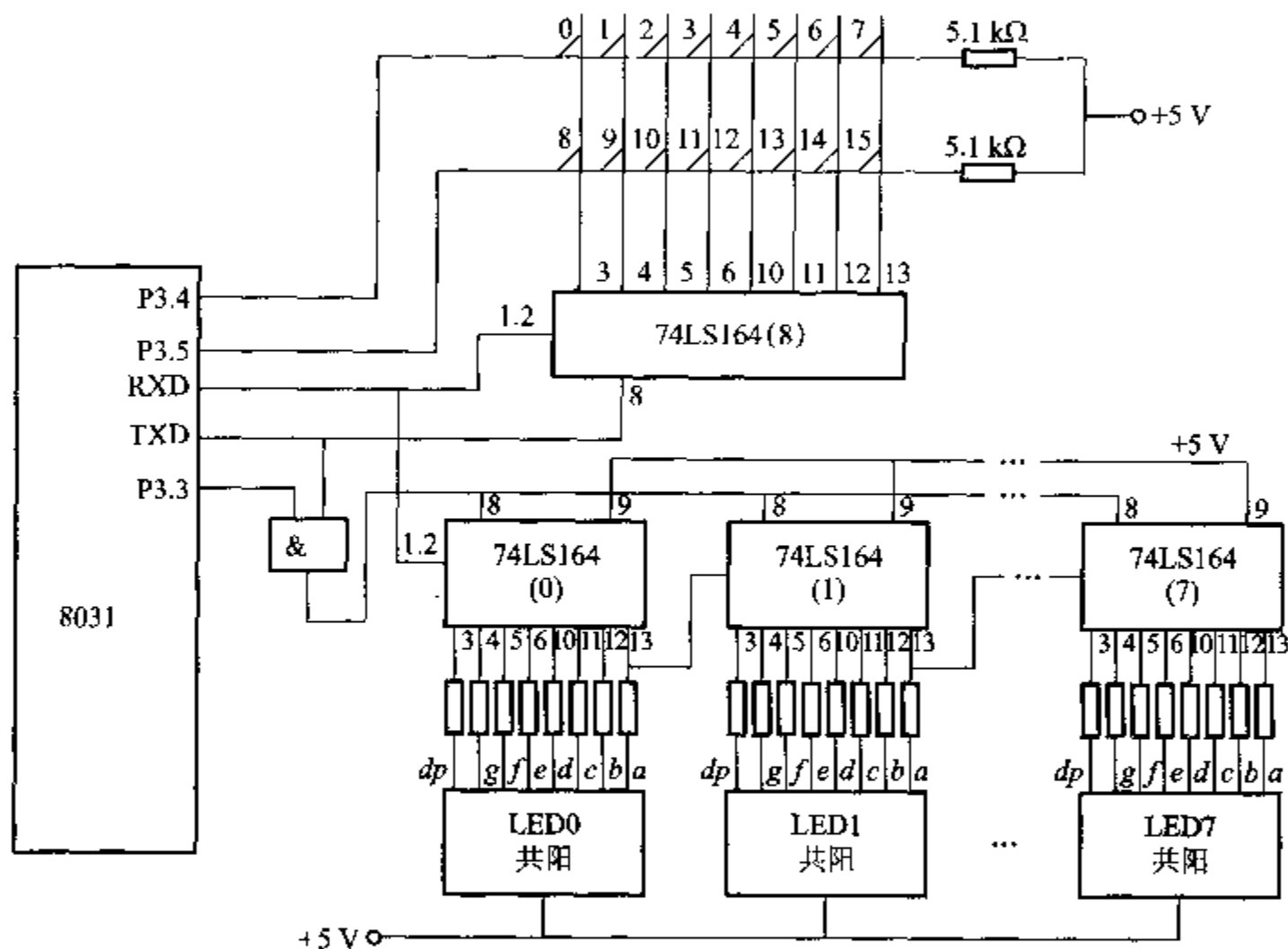


图 10-17 用 8031 串行口扩展键盘/显示器

图 10-17 中的 8 个 74LS164; 74LS164(0)~74LS164(7) 作为 8 位 LED 的段码输出口, 8031 的 P3.4、P3.5 作为 2 行键的行状态输入线, P3.3 作为 TXD 引脚同步移位脉冲输出控制线, P3.3=0 时, 与门输出为 0, 禁止同步移位脉冲输出。这种静态显示方式的优点是亮度大, 很容易做到显示不闪烁, 且 CPU 不必频繁的为显示服务, 因而主程序可不必扫描显示器, 软件设计比较简单, 从而使单片机有更多的时间处理其他事务。下面分别列出显示子程序和键盘扫描子程序的清单。

显示子程序:

```

DIR:  SETB    P3.3          ;P3.3=1,允许 TXD 引脚同步移位脉冲
                                ;输出
                                MOV    R7,#08H          ;送出的段码个数,R7 为段码个数计数器
                                MOV    R0,#7FH          ;7FH~78H 为显示数据缓冲区
DL0:  MOV     A,@R0          ;取出要显示的数送 A
      ADD    A,#0DH          ;加上偏移量
      MOVC   A,@A+PC         ;查段码表 SEGTABLE,取出段码
      MOV    SBUF,A          ;将段码送 SBUF
DL1:  JNB     TI,DL1          ;输出段码,查询 TI 状态,1 B 的段码输出
                                ;完否?
      CLR    TI              ;1 B 的段码输出完,清 TI 标志
      DEC    R0              ;指向下一个显示数据单元
      DJNZ   R7,DL0          ;段码个数计数器 R7 是否为 0,如不为 0,
                                ;继续送段码
      CLR    P3.3            ;8 个段码输出完毕,关闭显示器输出
      RET                    ;返回
SEGTABLE: DB 0C0H,0F9H,0A4H,          ;共阳极段码表,0,1,2,3,4
           0B0H,99H
           DB  92H,82H,0F8H,          ;5,6,7,8,9
           80H,90H
           DB  88H,83H,0C6H,0A1H,    ;A,b,c,d,E
           86H
           DB  8EH,0BFH,8CH,0FFH,    ;F,-,P,暗
           0FFH

```

键盘扫描子程序:

```

KEY1: MOV     A,#00H          ;判有无键按下,使所有列线为 0 的编码
                                ;送 A
      MOV     SBUF,A          ;扫描键盘的(8)号 74LS164 输出为 00H,
                                ;使所有列线为 0
KL0:  JNB     TI,KL0          ;串行输出完否?
      CLR    TI              ;串行输出完毕,清 TI
KL1:  JNB     P3.4,PK1        ;第 1 行有闭合键吗? 如有,跳 PK1 进行处
                                ;理
      JB      P3.5,KL1        ;在第 2 行键中有闭合键吗? 无闭合键
                                ;跳 KL1
PK1:  ACALL   DL10            ;调用延时 10 ms 子程序 DL10,软件消除
                                ;抖动
      JNB     P3.4,PK2        ;判是否抖动引起的?

```

	JB	P3.5, KL1	
PK2:	MOV	R7, #08H	;不是抖动引起的
	MOV	R6, #0FEH	;判别是哪一个键按下, FEH 为最左一列
			;为低
	MOV	R3, #00H	;R3 为列号寄存器
	MOV	A, R6	;
KL5:	MOV	SBUF, A	;列扫描码从串行口输出
KL2:	JNB	TI, KL2	;等待串行口发送完
	CLR	TI	;串行口发送完毕, 清 TI 标志
	JNB	P3.4, PKONE	;读第 1 行线状态, 第 1 行有键闭合, 跳
			;PKONE 处理
	JB	P3.5, NEXT	;读第 2 行线状态, 是第 2 行某键否?
	MOV	R4, #08H	;第 2 行键中有键被按下, 行首键号 08H 送
			;R4
	AJMP	PK3	
PKONE:	MOV	R4, #00H	;第 1 行键中有键按下, 行首键号 00H
			;送 R4
PK3:	MOV	SBUF, #00H	;等待键释放, 发送 00H 使所有列线为低
KL3:	JNB	TI, KL3	
	CLR	TI	;发送完毕, 清标志
KL4:	JNB	P3.4, KL4	;判行线状态
	JNB	P3.5, KL4	
	MOV	A, R4	;2 行线均为高, 说明键已释放
	ADD	A, R3	;计算得键码→A
	RET		
NEXT:	MOV	A, R6	;列扫描码左移 1 位, 判下一列键是否按下
	RL	A	;列扫描向右一列
	MOV	R6, A	;记住列扫描码于 R6 中
	INC	R3	;列号增 1
	DJNZ	R7, KL5	;列计数器 R7 减 1, 8 列键都检查完否?
	AJMP	KEYI	;8 列键扫描完毕, 开始下一个键盘扫描
			;周期
DL10:	MOV	R7, #0AH	;延时 10 ms 子程序
DL:	MOV	R6, #0FFH	
DL6:	DJNZ	R6, DL6	
	DJNZ	R7, DL	
	RET		

10.3.3 利用通用键盘/显示器接口芯片 8279 实现键盘/显示器接口

Intel 公司的 8279 芯片是 1 种通用可编程键盘/显示器接口电路芯片,它能完成监视键盘输入和显示控制 2 种功能。

8279 对键盘部分提供 1 种扫描工作方式,能对 64 个按键键盘阵列不断扫描,自动消抖,自动识别出闭合的键并得到键号,能对双键或 N 键同时按下进行处理。

显示部分为 LED 或其它显示器提供了按扫描方式工作的显示接口,可显示多达 16 位的字符或数字。

1. 8279 的引脚及内部结构

8279 的引脚如图 10-18 所示。图 10-19 为 8279 的引脚功能。

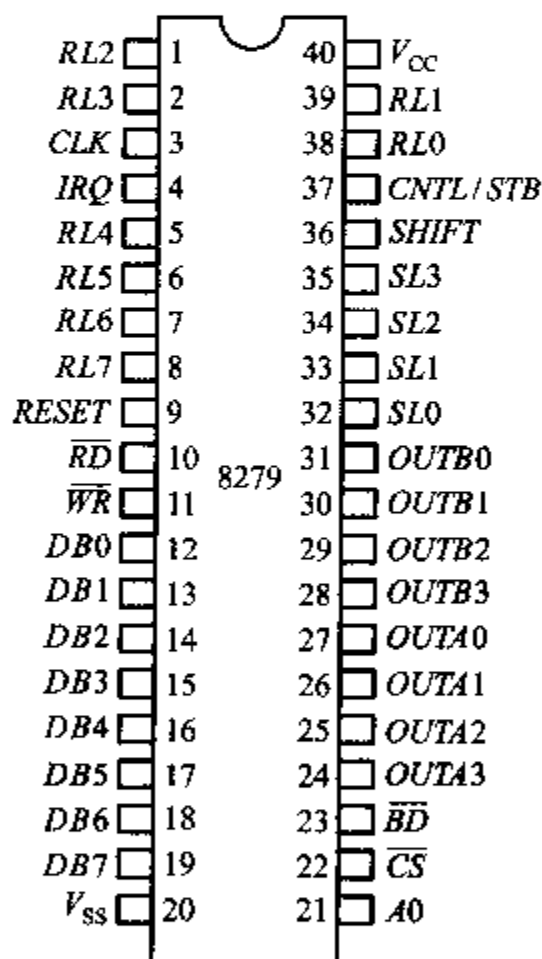


图 10-18 8279 的引脚

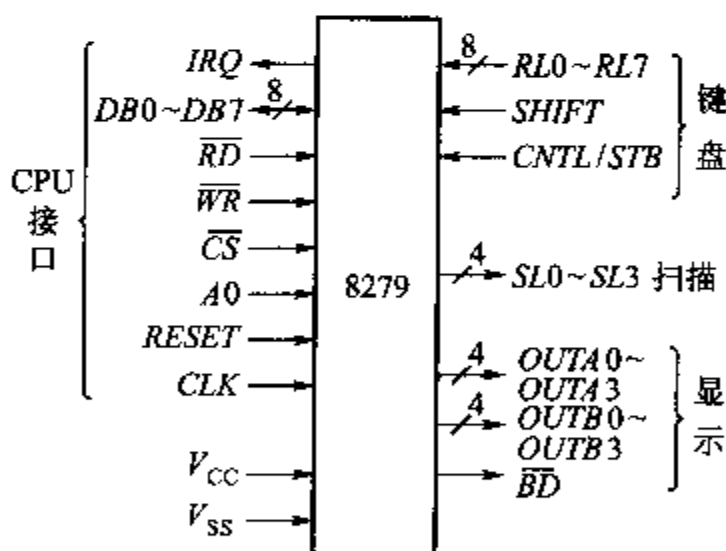


图 10-19 8279 的引脚功能

2. 引脚功能介绍

下面结合图 10-19 来介绍各引脚的功能。

(1) 与 CPU 的接口引脚

① DB0~DB7

数据总线、双向、三态,与单片机数据总线相连,在 CPU 和 8279 之间传送命令或数据。

② CLK

系统时钟输入线。用于 8279 内部定时,以产生其工作所需的时序。

③ RESET

复位输入线。高电平有效。该引脚为高电平时,8279 被复位,复位后的状态如下:

- 16 个字符左边输入显示方式
- 编码扫描键盘、双键锁定方式

④ \overline{CS}

片选线,输入、低电平有效。

$\overline{CS}=0$,8279 被选中,允许单片机对其进行读、写操作;

$\overline{CS}=1$,禁止对 8279 读、写。

⑤ A0

A0=1 时,CPU 写入 8279 的字节是命令字。从 8279 读出的字节是状态字。
A0=0 时,写入或读出的字节均为数据。

⑥ \overline{RD} 、 \overline{WR}

读、写控制引脚。输入线,低电平有效。这两个来自单片机的控制信号,控制单片机对 8279 的读出、写入操作。

⑦ IRQ

中断请求线,高电平有效。

在键盘工作方式中,当键盘 RAM(为先进先出方式)中存有按下键的数据时,IRQ 为高电平,向 CPU 提出中断申请。CPU 每次从键盘 RAM 中读出 1 B 数据时,IRQ 就变为低电平。如果键盘 RAM 中还有未读完的数据,IRQ 将再次变为高电平,再次提出中断请求。

(2) 扫描信号输出引脚

⑧ SL0~SL3

扫描输出线。这 4 条输出线用来扫描键盘和显示器。它们可以编程设定为编码输出,即 SL0~SL3 外接 4 线-16 线译码器,译码器输出 16 中取 1 的扫描信号,也可编程设定为译码输出,即由 SL0~SL3 直接输出 4 中取 1 的扫描信号。

(3) 与键盘连接的引脚

⑨ RL0~RL7

输入线。它们是键盘矩阵的行信号输入线。

⑩ SHIFT

输入线,高电平有效,通常用作键盘上、下挡功能的控制键。

⑪ CNTL/STB

输入线,高电平有效。在键盘方式时,通常用来作为键盘控制功能键

使用。

(4) 与显示器连接的引脚

⑫ OUTA0~OUTA3(A 组显示数据)、OUTB0~OUTB3(B 组显示数据)

这 2 组引脚均是显示信息输出线(例如,向 LED 显示器输出的段码),它们与扫描信号线 SL0~SL3 同步。2 组可以独立使用,也可以合并使用。

⑬ \overline{BD}

消隐显示控制,低电平有效。该输出信号用于显示位切换时的显示消隐或将显示器的显示消隐。

3. 8279 的基本功能部件

8279 中与键盘/显示器扫描有关的基本功能部件如下:

(1) 扫描计数器

扫描计数器有 2 种输出方式。按编码方式工作时,计数器作二进制计数。4 位计数状态从扫描线 SL0~SL3 输出,经外部译码器 4 线-16 线译码后,为键盘和显示器提供 16 中取 1 的扫描线。按译码方式工作时,扫描计数器的最低 2 位在 8279 内部被译码后,从 SL0~SL3 输出,为键盘和显示器直接提供了 4 中取 1 的扫描线。

(2) 键盘去抖动及回复缓冲器

8 根引脚 RL0~RL7 被接到键盘矩阵的行线。在逐列扫描时,当某一键闭合,消抖电路延时等待 10 ms 之后,再检验该键是否仍闭合。若闭合,则该键的行、列地址和附加的移位、控制状态一起形成键盘数据,送入 8279 内部的键盘 RAM 存储器。格式为:

D7	D6	D5	D4	D3	D2	D1	D0
CNTL	SHIFT	扫 描			回 复		

控制(CNTL)和移位(SHIFT)的状态由 2 个独立的附加开关决定,而扫描(D5、D4、D3)和回复(D2、D1、D0)则是被按键的行、列位置数据,D5、D4、D3 3 位是被按键的行编码,而 D2、D1、D0 3 位是被按键的列编码。

(3) 键盘 RAM 及其状态寄存器

键盘 RAM 是 1 个双重功能的 8×8 位 RAM。它是先进先出(FIFO)存储器。内部的 FIFO 状态寄存器用来存放 FIFO 的工作状态,如 FIFO 是空还是满,其中存有多少字符,是否操作出错等等。当 FIFO 存储器空间不足时,状态逻辑将产生 $IRQ=1$ 信号,向 CPU 发出中断申请。

在键盘阵列的行列交叉位置是开关传感器时,键盘 RAM 存放着传感器矩阵中的每一个传感器的开关状态。在此方式中,若检查出传感器的状态变化,

IRQ 信号变为高电平,向 CPU 发出中断申请。

(4) 显示 RAM 和显示地址寄存器

显示 RAM 用来存放显示数据。共 16 B,最多可以存放 16 位的显示信息。在显示过程中,这些信息被轮流从显示寄存器输出。而显示寄存器则分成 A、B 2 组,即 OUTA0~OUTA3 和 OUTB0~OUTB3,它们可以单独送数,也可以共同组成 1 个 8 位的字节。显示寄存器的输出与显示扫描配合,不断从显示 RAM 中读出显示数据,同时轮流驱动被选中的显示位,使显示器呈现出稳定的显示(动态扫描)。

4. 8279 的命令字和状态字

8279 是可编程接口芯片。编程就是 CPU 向 8279 写入命令控制字,共有 8 条。命令字的高 3 位 D7、D6 和 D5 为命令特征位,用来区分 8 条不同的命令。各条命令介绍如下:

(1) 键盘/显示方式设置命令字

D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	D	D	K	K	K

高 3 位 D7、D6、D5 位为特征位 000。

D4、D3 2 位用来设定显示器的显示方式,其定义如下:

D4	D3	显示方式
0	0	左边输入的 8 位字符显示
0	1	左边输入的 16 位字符显示
1	0	右边输入的 8 位字符显示
1	1	右边输入的 16 位字符显示

8279 最多可用来控制 16 位 LED 显示器,当显示位数超过 8 位时,均需设定为 16 位字符显示。显示器的每一位对应 8279 内部的 1 个 8 位的显示 RAM 单元。CPU 将显示数据写入显示 RAM 单元时,有左边输入和右边输入 2 种方式。左边输入是地址为 0~15 的显示缓冲 RAM 单元分别对应于显示器的 0(左)位~15(右)位。显示位置从最左一位开始,显示字符逐个向右顺序排列。右边输入就是显示位置从最右一位开始,以后逐次输入显示字符时,已有的显示字符依次向左移动。

当 16 个显示 RAM 都已写满时(从 0 地址开始写,写了 16 次),第 17 次写,再从 0 地址开始写入。

D2、D1、D0 为键盘工作方式选择位,如下表:

D2	D1	D0	键盘工作方式
0	0	0	编码扫描键盘, 双键锁定
0	0	1	译码扫描键盘, 双键锁定
0	1	0	编码扫描键盘, N 键依次读出
0	1	1	译码扫描键盘, N 键依次读出
1	0	0	编码扫描传感器矩阵
1	0	1	译码扫描传感器矩阵
1	1	0	选通输入, 编码扫描显示器方式
1	1	1	选通输入, 译码扫描显示器方式

当设定为编码工作方式时, 4 位二进制计数器的状态从扫描线 SL0~SL3 输出, 经外部 4 线-16 线译码器译码后, 最多可为键盘/显示器提供 16 根扫描信号线(16 选 1)。

当设定为内部译码工作方式时, 内部扫描计数器的低 2 位在内部被译码后, 再由 SL0~SL3 输出, 此时 SL0~SL3 已经是 4 选 1 的扫描信号线了。

双键锁定, 就是当键盘中同时有两个或两个以上的键被按下时, 任何一个键的编码信息均不能进入键盘 RAM 中, 直至仅剩下一键保持闭合时, 该键的编码信息方能进入键盘 RAM, 这种工作方式可以避免误操作信号进入计算机。

N 键依次读出, 就是各个键的处理都与其它键无关。按下一个键时, 片内去抖动电路等待两个键盘扫描周期, 然后检查该键是否仍按着。如果仍按着, 则该键编码就送入键盘 RAM 中。一次可以按下任意个键, 其它的键也可被识别出来并送入键盘 RAM 中。如果同时按下多个键, 则按键盘扫描过程发现它们的顺序识别, 并送入键盘 RAM 中。

扫描传感器矩阵的工作方式, 是指片内的去抖动逻辑被禁止掉, 传感器的开关状态直接输入键盘 RAM 中, 虽然这种方式不能提供去抖动的功能, 但有下列优点: CPU 知道传感器闭合多久, 何时释放。每当检测到传感器内部状态(开或闭)改变时, 中断线上的 IRQ 就变为高电平, 提出中断请求。

(2) 程控时钟命令

D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	P	P	P	P	P

D7、D6、D5=001 为命令的特征位。

D4、D3、D2、D1、D0=PPPPP 决定了对外部输入时钟 CLK 进行分频的分频系数 N。通过对 N 的设定以获得 8279 内部所需的 100 kHz 的时钟。例如:

外部时钟频率为 2 MHz, 取 N 为 20 即可获得 100 kHz 的内部时钟频率。内部时钟频率的高低控制着扫描时间和键盘去抖动时间的长短。在内部时钟为 100 kHz 时, 扫描时间为 5.1 ms, 去抖动时间为 10.3 ms。注意: 外部时钟信号的周期应不小于 500 ns。

(3) 读键盘 RAM 命令字

D7	D6	D5	D4	D3	D2	D1	D0
0	1	0	AI	×	A	A	A

$D7, D6, D5 = 010$ 为该命令特征位。该命令字只在传感器方式时使用。

$D2, D1, D0 = AAA$ 为传感器 RAM 中的 8 个字节地址。

$D4 = AI$ 为自动增量特征位。当 $AI = 1$ 时, 则每次读出传感器 RAM 之后, RAM 地址将自动加 1, 使地址指针指向顺序的下一个存储单元。这样下一次读数便从下一个地址读出, 而不必重新设置读键盘 RAM 命令。

(4) 读显示 RAM 命令

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	AI	A	A	A	A

$D7, D6, D5 = 100$ 为该命令特征字。该命令字用来设定将要读出的显示 RAM 地址。 $D3, D2, D1, D0 = AAAA$ 用来对显示 RAM 的 16 个存储单元寻址。

$D4 = AI$ 为自动增量特征位。当 $AI = 1$ 时, 每次读出之后, 地址自动加 1, 指向下一个地址, 所以下一次顺序读出数据时, 不必重新设置读显示 RAM 命令字。

(5) 写显示 RAM 命令

D7	D6	D5	D4	D3	D2	D1	D0
0	1	1	AI	A	A	A	A

$D7, D6, D5 = 100$ 为该命令特征字。该命令字用来设定将要写入的显示 RAM 地址。 $D3, D2, D1, D0 = AAAA$ 用来对显示 RAM 的 16 个存储单元寻址。

$D4 = AI$ 为自动增量特征位。当 $AI = 1$ 时, 每次写入之后, 地址自动加 1, 指向下一个地址, 所以下一次顺序写入数据时, 不必重新设置写显示 RAM 命令字。

(6) 显示禁止写入/消隐命令

D7	D6	D5	D4	D3	D2	D1	D0
1	0	1	×	IWA	IWB	BLA	BLB

D7、D6、D5=101 为该命令特征位。

D3、D2=IWA, IWB 位, 此 2 位分别用来屏蔽 A、B 2 组显示。例如当 A 组的屏蔽位 D3=1 时, A 组的显示 RAM 禁止写入。因此, 从 CPU 写入显示器 RAM 的数据不会影响 A 的显示。这种情况通常在采用双 4 位显示器时使用。因为 2 个 4 位显示器是独立的, 为了给其中一个 4 位显示器输入数据而又不影响另一个 4 位显示器, 因此必须对另一组的输入实行屏蔽。

D1、D0=BLA, BLB 位是 2 个消隐特征位。分别对 2 组显示输出进行消隐, 当 BL=1 时, 对应显示组被消隐, 而当 BL=0 时, 则恢复正常显示。

(7) 清除命令

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	CD	CD	CD	CF	CA

该命令字用来对键盘 RAM 和显示 RAM 清 0。

D7、D6、D5=110 为该特征位。

D3、D2、D1=CD CD CD, 用来设定清除显示 RAM 的方式。共有 4 种清 0 方式, 定义如下:

D4	D3	D2	清除显示 RAM 的方式
1	0	×	将显示 RAM 全部清 0
1	1	0	将显示 RAM 全部清成 20H
1	1	1	将显示 RAM 全部置 1
0	×	×	不清除(CA=0 时); 若 CA=1, 则 D3、D2 仍有效

D1=CF 用来清空显示 RAM。当 CF=1 时, 执行清除命令后, 显示 RAM 被清空, 使中断输出线 IRQ 复位, 同时, 传感器 RAM 的读出地址也被清 0。

D0=CA 是总清的特征位。它兼有 CD 和 CF 两位的功效。当 CA=1 时, 对显示的清除方式由 D3、D2 两位编码决定。

清除显示 RAM 大约需要 160 μ s 的时间, 在此期间, CPU 不能向显示 RAM 写入数据。

(8) 结束中断/错误方式设置命令

D7	D6	D5	D4	D3	D2	D1	D0
1	1	1	E	×	×	×	×

$D7, D6, D5 = 111$ 为该命令的特征位。

这个命令有两种不同的应用。

作为结束中断命令,在传感器工作方式中用来结束传感器 RAM 的中断请求。

作为特定错误方式设置命令,在 8279 已被设定为键盘扫描 N 键轮回方式以后,如果 CPU 给 8279 有写入结束中断/错误方式设置命令($E=1$),则 8279 将以一种特定的错误方式工作。即 8279 在消抖周期内,如果发现有多键被同时按下,则 FIFO 状态字中的错误特征位 S/E 将置 1,并将产生中断请求信号和阻止写入 FIFO RAM。

至此,8279 的 8 个命令字已介绍完毕。8 个命令字均由 $D7, D6, D5$ 特征位确定,当写入 8279 之后能自动寻址到相应的命令寄存器。只是在写入命令时,命令字一定要写入命令寄存器端口中,即应让 $A0=1$ 。

(9) 8279 的状态字

8279 的状态字,主要用于键盘工作方式,以指示键盘 RAM 的字符数和有无错误发生。

$D7$	$D6$	$D5$	$D4$	$D3$	$D2$	$D1$	$D0$
DU	S/E	O	U	F	N	N	N

$D7=DU$ 为显示无效特征位。当 $DU=1$ 表示显示无效。当显示 RAM 由于清除显示或全清命令尚未完成时, $DU=1$ 。

$D6=S/E$ 为传感器信号结束/错误特征位。8279 工作在传感器工作方式或特殊错误方式使用。

$D5, D4=O, U$ 为超出、不足错误特征位。对于键盘 RAM 的操作可能出现两种错误:超出或不足。键盘 RAM 已经充满时,若其他的键盘数据还企图写入键盘 RAM 中,则出现超出错误,状态字的 U 位被置 1;当键盘 RAM 为空时,若 CPU 还企图读出,则出现不足错误,状态字的 U 位置 1。

$D3=F$ 表示键盘 RAM 是否已满。当 $F=1$ 时,表示键盘 RAM 已满。

$D2, D1, D0=NNN$ 表示键盘 RAM 中的字符数,最多 8 个。

5. 8279 与键盘/显示器的接口

图 10-20 为 MCS-51 通过 8279 与 8 位显示器, 4×8 键盘的接口电路。图中键盘的行线接 8279 的 $RL0 \sim RL3$, 8279 选用外部译码方式, $SL0 \sim SL2$ 经 74LS138(1)译码输出,接键盘的列线,来实现逐列扫描。

$SL0 \sim SL2$ 又由 74LS138(2)译码输出,经驱动后到显示器各位的公共阴极,进行逐位扫描显示。输出线 $OUTB0 \sim OUTB3, OUTA0 \sim OUTA3$ 作为 8 位段数据输出口,输出段码。当位切换时, \overline{BD} 输出为低电平,使 74LS138(2)输出全为高

与 8279 有关的初始化程序:

```
INITI:  SETB    EX1                ; 允许外部中断 1 中断
        MOV     DPTR, #7FFFH      ; 命令/状态口地址写入 DPTR
        MOV     A, #0D1H          ; 控制字 D1H 送 A
        MOVX    @DPTR, A          ; 向命令/状态口写入控制字
LP:     MOVX    A, @DPTR           ; 读 8279 的状态
        JB      A.7, LP
        MOV     A, #00H
        MOVX    @DPTR, A
        MOV     A, #2AH
        MOVX    @DPTR, A
        SETB    EA
        ...
        ...
```

键输入中断服务程序:

```
PINT1:  PUSH    PSW
        PUSH    DPH
        PUSH    DPL
        PUSH    A
        MOV     DPTR, #7FFFH      ; 向命令口写入读键盘 RAM 命令
        MOV     A, #40H
        MOVX    @DPTR, A
        MOV     DPTR, #7FFEh      ; 读键输入值
        MOVX    A, @DPTR
        CJNE    A, #37H, PRI1      ; 判输入停机命令否
        SETB    20H
PRI1:   POP     A
        POP     DPL
        POP     DPH
        POP     PSW
        RETI
```

显示子程序:

```
DIR:   MOV     DPTR, #7FFFH      ; 输出写显示 RAM 命令
        MOV     A, #90H
        MOVX    @DPTR, A
        MOV     R0, #70H
        MOV     R7, #08H          ; 送显示 RAM 数据的个数
```

```
MOV    DPTR, #7FFEH
DL0:   MOV    A, @R0
ADD    A, #05H      ;05H 为查表偏移量
MOVC   A, @A+PC     ;查表得到段码
MOVX   @DPTR, A      ;写入显示 RAM
INC    R0            ;显示数据单元地址增 1
DJNZ   R7, DL0       ;8 个显示数据是否输出完毕
RET
ADSEG: DB  3FH, 06H, 5BH, 4FH, 66H, 6DH      ;段码表(共阴极)
        DB  7DH, 07H, 7FH, 6FH, 77H, 7CH
        DB  39H, 5EH, 79H, 71H, 73H, 3EH
        DB  31H, 6EH, 1CH, 23H, 40H, 03H
        DB  18H, 38H, 00H
```

10.4 MCS-51 与液晶显示器(LCD)的接口

LCD(Liquid Crystal Display)是液晶显示器英文名称的缩写,液晶显示器是一种被动式的显示器,即液晶本身并不发光,而是利用液晶经过处理后能改变光线通过方向的特性,达到白底黑字或黑底白字显示的目的。液晶显示器具有功耗低、抗干扰能力强等优点,因此被广泛地应用在仪器仪表和控制系统中。

10.4.1 LCD显示器的分类

当前市场上液晶显示器种类繁多,按排列形状可分为字段型、点阵字符型和点阵图形型。

(1) 字段型。字段型是以长条状组成的字符显示。该类显示器主要用于数字显示,也可用于显示西文字母或某些字符,已广泛用于电子表、数字仪表、计算器中。

(2) 点阵字符型。点阵字符型液晶显示模块是专门用来显示字母、数字、符号等点阵型液晶显示模块。它是由若干个 5×7 或 5×10 点阵组成,每一个点阵显示 1 个字符。此类显示模块广泛应用在各类单片机应用系统中。

(3) 点阵图形型。点阵图形型是在平板上排列多行或多列,形成矩阵式的晶格点,点的大小可根据显示的清晰度来设计。这类液晶显示器可广泛应用于图形显示如游戏机、笔记本电脑和彩色电视等设备中。

10.4.2 点阵字符型液晶显示模块介绍

在单片机应用系统中,常使用点阵字符型 LCD 显示器。要使用点阵字符型

LCD 显示器,必须有相应的 LCD 控制器、驱动器,来对 LCD 显示器进行扫描、驱动,以及一定空间的 RAM 和 ROM 来存储写入的命令和显示字符的点阵。现在人们已将 LCD 控制器、驱动器、RAM、ROM 和 LCD 显示器用 PCB 连接到一起,称为液晶显示模块 LCM(LCd Module)。使用者只要向 LCM 送入相应的命令和数据就可实现所需要的显示内容,这种模块与单片机接口简单,使用灵活方便。产品分为字符和图形两种。下面仅就使用较为广泛的国内天马公司制作的点阵字符液晶显示模块的基本结构、指令功能和特点加以介绍。

1. 基本结构

(1) 液晶板

在液晶板上排列着若干 5×7 或 5×10 点阵的字符显示位,从规格上分为每行 8、16、20、24、32、40 位,有 1 行、2 行及 4 行 3 类,用户可根据需要来选择购买。

(2) 模块电路框图

图 10-22 是字符型模块的电路框图,它由控制器 HD44780、驱动器 HD44100 及几个电阻电容组成。HD44100 是扩展显示字符位用的(例如:16 字符 \times 1 行模块就可不用 HD44100,16 字符 \times 2 行模块就要用 1 片 HD44100)。

模块上有 14 个引脚(见图中左侧),其中有 8 条数据线,3 条控制线,3 条电源线,见表 10-2。通过单片机写入模块的数据和指令,就可对显示方式和显示的内容作出选择。

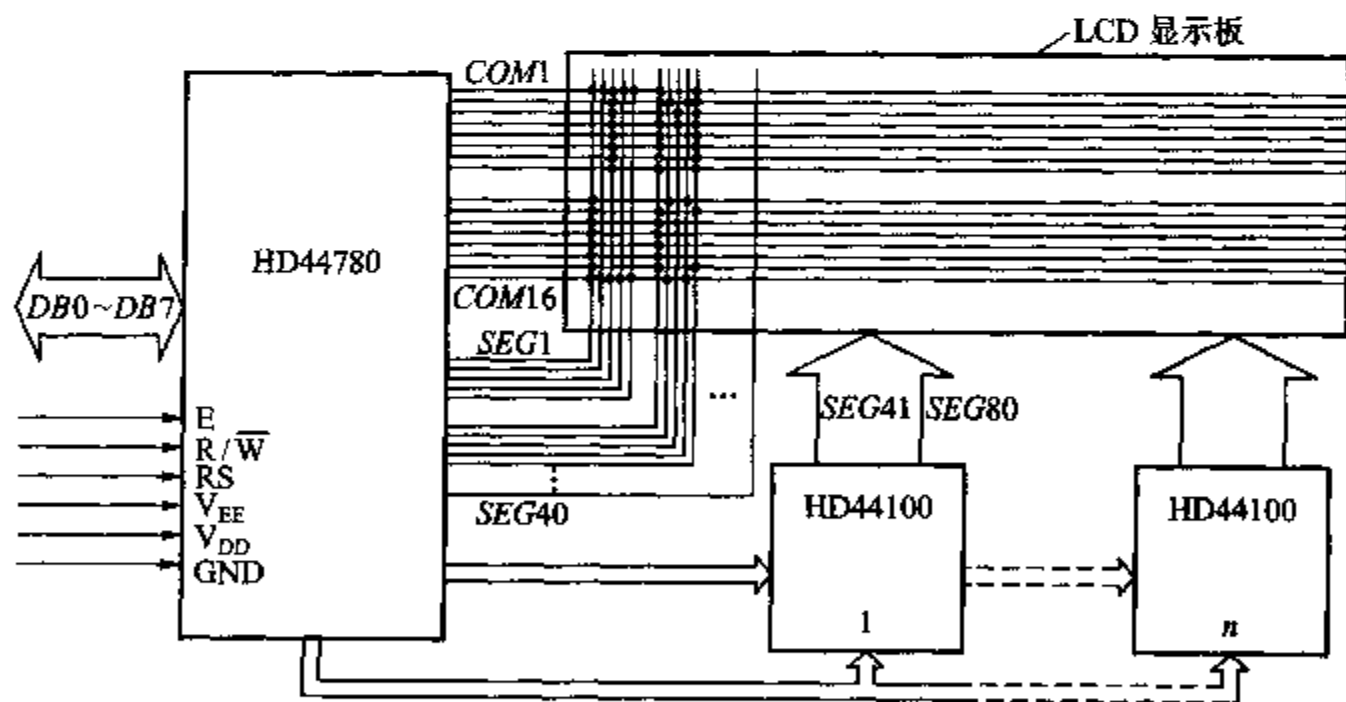


图 10-22 字符型 LCD 模块的电路框图

表 10-2 液晶显示模块的引脚

引线号	符号	名称	功能
1	V_{SS}	地	0 V
2	V_{DD}	电源	5 V \pm 5%

续表

引线号	符号	名 称	功 能
3	V_{EE}	液晶驱动电压	
4	RS	寄存器选择	1:数据寄存器,0:命令寄存器
5	R/\overline{W}	读/写	1:读,0:写
6	E	使能	下降沿触发
7	DB0	8 位数据线	数据传输
⋮	⋮		
14	DB7		

(3) LCD 驱动器和控制器

下面对图 10-22 中驱动器 HD44100 和控制器 HD44780 加以简单介绍。

① LCD 驱动器 HD44100

HD44100 是用低功耗 CMOS 技术制造的大规模 LCD 驱动 IC。它既可当行驱动用,也可当列驱动用,由 20×2 位二进制移位寄存器, 20×2 位数据锁存器和 20×2 位驱动器组成。

② LCD 控制器 HD44780

HD44780(KS0062)是用低功耗 CMOS 技术制造的大规模点阵 LCD 控制器(兼带驱动器),与 4 位/8 位微处理器相连,它能使点阵 LCD 显示大、小写英文字母、数字和符号。应用 HD44780,用户能用少量元件就可组成一个完整的点阵 LCD 系统。

功能和特性如下:

- 可选择 5×7 或 5×10 点字符;
- 显示数据 RAM 容量: 80×8 位(80 字符);
- 字符发生器 ROM 能提供用户所需字符库或标准库;

字库容量:192 个字符(5×7 点字型);32 个字符(5×10 点字型);可自编 8 (5×7 点)或 4(5×10)种字符。

- 输出信号:16 个行扫描信号,40 个列扫描信号;
- 命令:11 种。

2. 命令格式及命令功能说明

(1) 命令格式

LCD 控制器 HD44780 内有多寄存器,如表 10-3 所示。

表 10-3 寄存器的选择

RS	R/\overline{W}	操 作
0	0	命令寄存器写入
0	1	忙标志和地址计数器读出

续表

RS	R/ \overline{W}	操 作
1	0	数据寄存器写入
1	1	数据寄存器读出

RS 位和 R/ \overline{W} 引脚上的电平决定寄存器的选择,而 DB7~DB0 则决定命令功能。命令共 11 种,它们是:清除,返回,输入方式设置,显示开关控制,移位控制,功能设置,CGRAM(字符生成 RAM)地址设置,DDRAM(显示数据 RAM)地址设置,读忙标志和地址,写数据到 CGRAM 或 DDRAM,从 CGRAM 或 DDRAM 读数据。这些命令功能强,可组合成各种输入、显示、移位方式以满足不同的要求。

(2) 命令功能说明

① 清屏

命令格式:

RS	R/ \overline{W}	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	0	1

功能:清除屏幕显示,并置地址计数器 AC 为 0。

② 返回

命令格式:

RS	R/ \overline{W}	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	0	1	×

功能:置 DDRAM 及显示 RAM 的地址为 0,显示返回到原始位置。

③ 输入方式设置

命令格式:

RS	R/ \overline{W}	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	0	1	I/D	S

功能:设置光标的移动方向,并指定整体显示是否移动。其中 I/D 位如为 1,则是增量方式,如为 0,则是减量方式;S 位如为 1,则移位,如为 0,则不移位。

④ 显示开关控制

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	0	1	D	C	B

功能:

D 位控制整体显示的开与关, $D=1$,开显示; $D=0$,则关显示。

C 位控制光标的开与关, $C=1$,光标开; $C=0$,则光标关。

B 位控制光标处字符的闪烁, $B=1$,字符闪烁; $B=0$,字符不闪烁。

⑤ 光标移位

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	0	1	S/C	R/L	×	×

功能:移动光标或整体显示,DDRAM 中内容不变。

其中:

$S/C=1$ 时,显示移位; $S/C=0$ 时光标移位。

$R/L=1$ 时,向右移位, $R/L=0$ 时向左移位。

⑥ 功能设置

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	0	1	DL	N	F	×	×

功能:

DL 设置接口数据位数, $DL=1$ 为 8 位数据接口, $DL=0$ 为 4 位数据接口。

N 设置显示行数, $N=0$,单行显示, $N=1$,双行显示。

F 设置字型大小, $F=1$,为 5×10 点阵, $F=0$ 时为 5×7 点阵。

⑦ CGRAM 地址设置

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

功能:设置 CGRAM 的地址,地址范围为 $0 \sim 63$ 。

⑧ DDRAM 地址设置

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A

功能:设置 DDRAM 的地址,地址范围为 0~127。

⑨ 读忙标志 BF 及地址计数器

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	BF	AC						

功能:

BF 为忙标志, $BF=1$ 表示忙,此时 LCM 不能接收命令和数据, $BF=0$,则表示 LCM 不忙,可以接收命令和数据。

AC 为地址计数器的值,范围是 0~127。

⑩ 向 CGRAM/DDRAM 写数据

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	DATA							

功能:将数据写入 CGRAM 或 DDRAM 中,应与 CGRAM 或 DDRAM 地址设置命令相结合。

⑪ 从 CGRAM/DDRAM 中读数据

命令格式:

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	DATA							

功能:本指令从 CGRAM 或 DDRAM 中读出数据,应与 CGRAM 或 DDRAM 地址设置命令相结合。

(3) 有关说明

① 显示位与 DDRAM 地址的对应关系,见表 10-4

表 10-4 显示位与 DDRAM 地址的对应关系

显示位		1	2	3	4	5	6	7	8	9	...	39	40
DDRAM	第一行	0 0	0 1	0 2	0 3	0 4	0 5	0 6	0 7	0 8	...	2 6	2 7
地址(H)	第二行	4 0	4 1	4 2	4 3	4 4	4 5	4 6	4 7	4 8	...	6 6	6 7

② 标准字符库

图 10-23 所示的是字符库的内容、字符码和字型的对应关系。例如“A”的字符码为 41(HEX),“B”的字符码为 42(HEX)。

高4位 低4位	MSB 0000	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
LSB XXXX0000	CG RAM (1)		0	1	2	3	4	5	6	7	8	9	A	B	C
XXXX0001	(2)	!	1	A	Q	a	q			0	P	チ	ム	ヨ	ウ
XXXX0010	(3)	"	2	B	R	b	r			1	イ	ツ	※	フ	田
XXXX0011	(4)	#	3	C	S	c	s			2	ウ	テ	モ	ミ	ウ
XXXX0100	(5)	\$	4	D	T	d	t			3	エ	ト	カ	ム	ロ
XXXX0101	(6)	%	5	E	U	e	u			4	オ	ナ	1	セ	ウ
XXXX0110	(7)	&	6	F	V	f	v			5	カ	ニ	ヨ	ル	マ
XXXX0111	(8)	'	7	G	W	g	w			6	キ	※	ラ	グ	ハ
XXXX1000	(1)	(8	H	X	h	x			7	ク	ネ	リ	ル	※
XXXX1001	(2))	9	I	Y	i	y			8	ケ	ル	ル	リ	ユ
XXXX1010	(3)	*	:	J	Z	j	z			9	エ	コ	ハ	レ	リ
XXXX1011	(4)	+	:	K	[k	[0	オ	サ	ヒ	ロ	※
XXXX1100	(5)	,	<	L	¥	l	¥			1	カ	シ	フ	ワ	ホ
XXXX1101	(6)	-	=	M]	m]			2	ユ	ズ	ノ	コ	モ
XXXX1110	(7)	.	>	N	^	n	^			3	ヨ	セ	ホ	ノ	ホ
XXXX1111	(8)	/	?	O	_	o	_			4	ツ	リ	マ	ロ	※

图 10-23 字符库的内容

③ 字符码(DDRAM DATA),CGRAM 地址与自编字型(CGRAM DATA)之间的关系,如表 10-5 所示。

字符码的高 4 位 DB7~DB4 为 0 时,即为自编字型码,其低 3 位 DB0~DB2 即 *aaa* 共寻址 8 个自编字符,并与 CGRAM 地址的 DB5~DB3 3 位相对应,而 CGRAM 地址的低 3 位 DB2~DB0 则用来寻址自编字型点阵数据,即 CGRAM DATA。点阵数据每字符 8 B,每字节低 5 位有效。表中为字符“Y”的点阵数据。

表 10-5

DDRAM 数据								CGRAM 地址						CGRAM 数据(字符“Y”的点阵数据)							
7	6	5	4	3	2	1	0	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0 0 0 0 × <i>aaa</i>								<i>aaa</i>						×	×	×	1	0	0	0	1
														×	×	×	0	1	0	1	0
														×	×	×	1	1	1	1	1
														×	×	×	0	0	1	0	0
														×	×	×	1	1	1	1	1
														×	×	×	0	0	1	0	0
														×	×	×	0	0	1	0	0
														×	×	×	0	0	0	0	0

10.4.3 8031 与 LCD 的接口及软件编程

1. 8031 与 LCD 模块的接口

8031 与 LCD 模块(LCM)的接口电路见图 10-24 所示。也可以将 LCM 挂在 8031 的总线上,通过对数据总线的读写实现对 LCM 的控制,如图 10-24 所示。

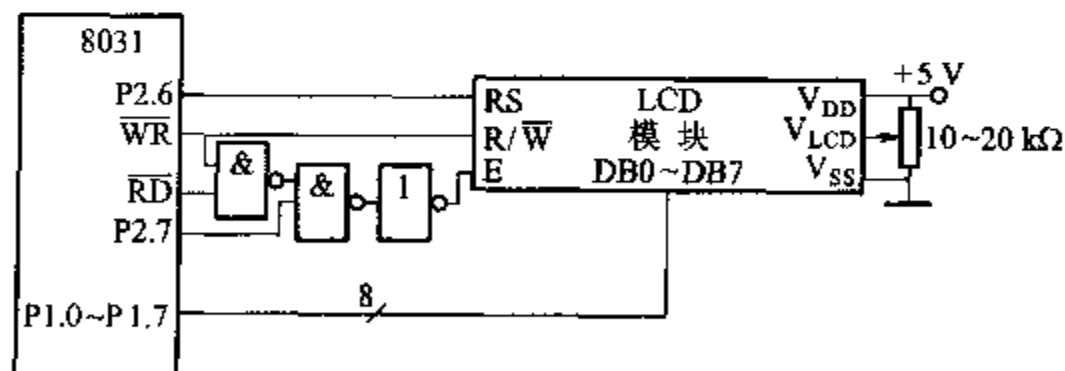


图 10-24 8031 与 LCD 模块的接口电路

2. 软件编程

(1) 初始化

用户所编的显示程序,开始必须进行初始化,否则模块无法正常显示。下面介绍两种初始化方法:

① 利用模块内部的复位电路进行初始化

LCM 有内部复位电路,能进行上电复位。复位期间 BF 为 1,在电源电压 V_{DD} 达 4.5 V 以后,此状态可维持 10 ms,复位时依次执行下列命令:

- 清除显示。
- 功能设置, $DL=1$, 为 8 位数据长度接口; $N=0$, 单行显示; $F=0$, 为 5×7 点阵字符。
- 开/关设置, $D=0$, 关显示; $C=0$, 关光标; $B=0$, 关闪烁功能。
- 进入方式设置, $I/D=1$, 地址采用递增方式; $S=0$, 关显示移位功能。

② 软件初始化

软件初始化流程如图 10-25 所示。



图 10-25 软件初始化流程

(2) 显示程序编写

例 10-1 编写程序在 LCD 第 1 行显示出“CS&S”,第 2 行显示“92”。

假定对 LCM 已完成初始化。程序如下:

```

START:  MOV  DPTR, #8000H      ;命令口地址 8000H 送 DPTR
        MOV  A, #01H          ;清屏并置 AC 为 0
    
```

```

MOVX  @DPTR,A           ;输出命令
ACALL F_BUSY             ;等待直至 LCM 不忙
MOV   A,#30H             ;功能设置,8位接口,2行显
                           ;示,5×7 点阵

MOVX  @DPTR,A
ACALL F_BUSY
MOV   A,#0EH             ;开显示及光标,不闪烁
MOVX  @DPTR,A
ACALL F_BUSY
MOV   A,#06H             ;显示??,AC 为增量
MOVX  @DPTR,A
ACALL F_BUSY
MOV   DPTR,#C000H        ;数据口地址 C000H 送 DPTR
MOV   A,#43H             ;C 的 ASCII 码为 43H
MOVX  @DPTR,A            ;第一行第一位显示 C
ACALL F_BUSY
MOV   A,#53H             ;S 的 ASCII 码为 53H
MOVX  @DPTR,A            ;显示 CS
ACALL F_BUSY
MOV   A,#26H             ;& 的 ASCII 码为 26H
MOVX  @DPTR,A            ;显示 CS&
ACALL F_BUSY
MOV   A,#53H
MOVX  @DPTR,A            ;显示 CS&S
ACALL F_BUSY
MOV   DPTR,#8000H        ;指向命令口
MOV   A,#0C0H            ;置 DDRAM 地址为 40H
MOVX  @DPTR,A            ;光标于第二行首显示
ACALL F_BUSY
MOV   DPTR,#C000H        ;指向数据口
MOV   A,#39H             ;9 的 ASCII 码为 39H
MOVX  @DPTR,A            ;显示 9
ACALL F_BUSY
MOV   A,#32H             ;2 的 ASCII 码为 32H
MOVX  @DPTR,A            ;显示 92
...

```

由于 LCD 是一慢速显示器件,所以在执行每条指令之前一定要确认 LCM 的忙标志为 0,即非忙状态,否则此指令将失效。上面程序中判定忙标志的子程

序 F_BUSY 如下:

```
F_BUSY:  PUSH    DPH      ;保护现场
          PUSH    DPL
          PUSH    PSW
          PUSH    A
LOOP:    MOV     DPTR, #8000H
          MOVX    A, @DPTR
          JB      A.7, LOOP ;忙,继续等待
          POP     A      ;不忙,恢复现场返回
          POP     PSW
          POP     DPL
          POP     DPH
          RET
```

10.5 MCS-51 与微型打印机的接口

在单片机应用系统中多使用微型点阵式打印机,在微型打印机的内部有一个控制用单片机,固化有控打程序,智能化程度高。

打印机通电后,由打印机内部的单片机执行固化程序,就可以接收和分析主控单片机送来的数据和命令,然后通过控制电路,实现对打印头机械动作的控制,进行打印。此外,微型打印机还能接受人工干预,完成自检、停机和走纸等操作。

在单片机应用系统中,常用的微型打印机有 TP μ P-40A/16A、GP16 以及 XLF 嵌入仪器面板上的汉字微型打印机。本节介绍 MCS-51 与常见的 TP μ P-40A/16A、GP16 微型打印机的接口设计。

10.5.1 MCS-51 与 TP μ P-40A/16A 微型打印机的接口

1. TP μ P-40A/16A 微型打印机

TP μ P-40A/16A 是一种单片机控制的微型智能打印机。TP μ P-40A 与 TP μ P-16A 的接口信号与时序完全相同,操作方式相近,硬件电路及插脚完全兼容,只是指令代码不完全相同。TP μ P-40A 每行打印 40 个字符,TP μ P-16A 则每行打印 16 个字符。

2. 主要性能、接口要求及时序

(1) TP μ P-40A 主要技术性能

① 采用单片机控制,具有 2 KB 控打程序以及标准的 Centronics 并行接口。

② 可打印全部标准的 ASCII 代码字符,以及 128 个非标准字符和图符。有 16 个代码字符(6×7 点阵)可由用户通过程序自行定义。并可通过命令用此 16 个代码字符去更换任何驻留代码字型,以便用于多种文字的打印。

③ 可打印出 8×240 点阵的图样(汉字或图案点阵)。代码字符和点阵图样可在一行中混合打印。

④ 字符、图符和点阵图可以在宽和高的方向放大为×2、×3、×4 倍。

⑤ 每行字符的点行数(包括字符的行间距)可用命令更换。即字符行间距空点行在 0~256 间任选。

⑥ 带有水平和垂直制表命令,便于打印表格。

⑦ 具有重复打印同一字符命令,以减少输送代码的数量。

⑧ 带有命令格式的检错功能。当输入错误命令时,打印机立即打印出错误信息代码。

(2) 接口信号

TP_μP-40A 采用国际上流行的 Centronics 打印机并行接口,与单片机间是通过 1 条 20 芯扁平电缆及接插件相连。打印机有 1 个 20 线扁平插座,信号引脚排列如图 10-26 所示。

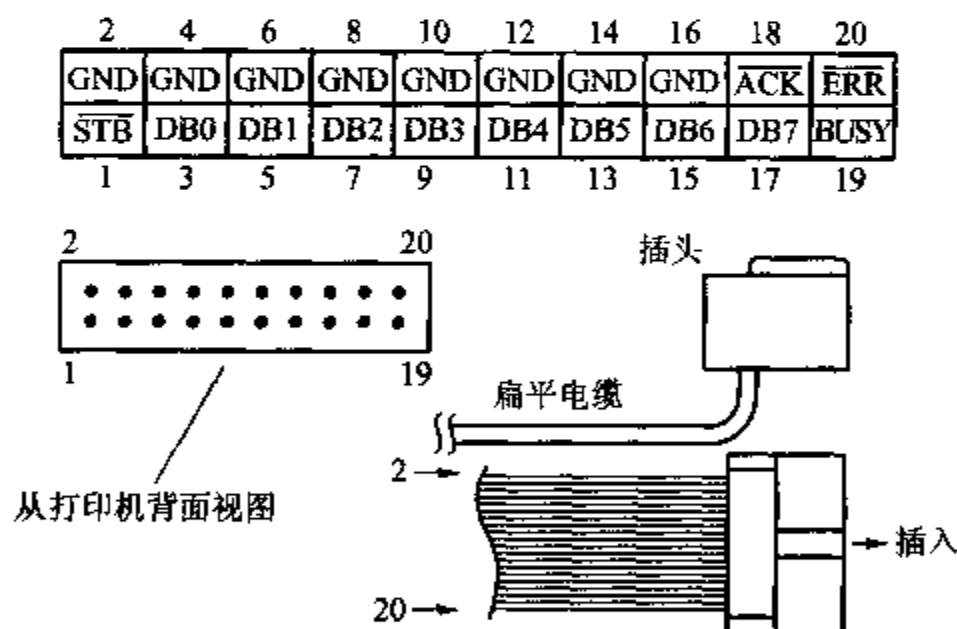


图 10-26 TP_μP-40A 插脚安排(从打印机背视)

其中:

- DB0~DB7: 数据线,单向传输,由单片机输入给打印机。
- \overline{STB} (STROBE): 数据选通信号。在该信号的上升沿,数据线上的 8 位并行数据被打印机读入机内锁存。
- BUSY: 打印机忙状态信号。当该信号有效(高电平)时,表示打印机正忙于处理数据。此时,单片机不得使 \overline{STB} 信号有效,向打印机送入新的数据。
- \overline{ACK} : 打印机的应答信号。低电平有效,表明打印机已取走数据线上的

数据。

- \overline{ERR} : 出错信号。当送入打印机的命令格式出错时, 打印机立即打印 1 行出错信息, 提示出错。在打印出错信息之前, 该信号线出现 1 个负脉冲, 脉冲宽度为 $30\ \mu\text{s}$ 。

(3) 接口信号时序

接口信号时序如图 10-27 所示。

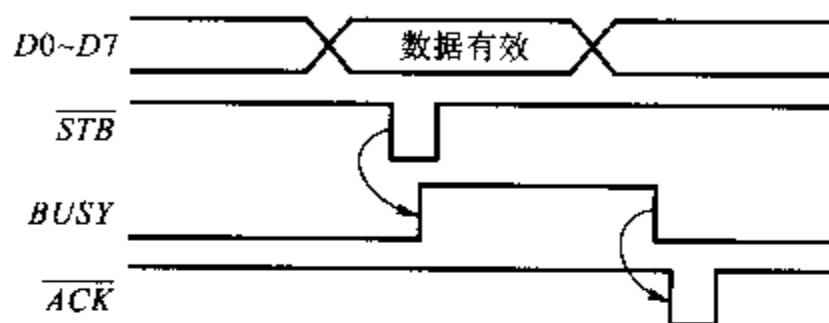


图 10-27 TP μ P-40A/16A 接口信号时序

选通信号 \overline{STB} 宽度须大于 $0.5\ \mu\text{s}$ 。 \overline{ACK} 应答信号可与 \overline{STB} 信号作为一对应答联络信号, 也可使用 \overline{STB} 和 $BUSY$ 作为一对应答联络信号。

3. 字符代码及打印命令

写入 TP μ P-40A/16A 的全部代码共 256 个, 其中 00H 无效。代码 01H~0FH 为打印命令; 代码 10H~1FH 为用户自定义代码; 代码 20H~7FH 为标准 ASCII 代码; TP μ P-40A 可打印的非 ASCII 代码如图 10-28 所示, 代码 80H~FFH 为非 ASCII 代码, 其中包括少量汉字、希腊字母、块图图符和一些特殊字符。

TP μ P-16A 的有效代码表与 TP μ P-40A 的不同之处仅在于 01H~0FH 中的指令代码, 前者为 16 个, 后者为 12 个, 功能也不同。

(1) 字符代码

TP μ P-40A/16A 中全部字符代码为 10H~FFH, 回车换行代码 0DH 为字符串的结束符。但当输入代码满 40/16 个时, 打印机自动回车。几个例子如下:

① 打印 "\$2356.73"

输送代码串为: 24, 32, 33, 35, 36, 2E, 37, 33, 0D。

② 打印 "23.7 cm³"

输送代码为: 32, 33, 2E, 37, 63, 6D, 9D, 0D。

(2) 打印命令

打印命令由一个命令字和若干个参数字节组成, 表 10-6 为 TP μ P-40A 命令代码及功能。命令结束符为 0DH, 除表 10-6 中代码为 06H 的命令必须用它外, 其余均可省略。更详细的说明, 参见技术说明书。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8																
9																
A																
B																
C																
D																
E																
F																

图 10-28 TP_μP-40A 的非 ASCII 代码

表 10-6 命令代码表

命令代码	命令功能	命令代码	命令功能
01H	打印字符、图等,增宽(×1,×2,×3,×4)	08H	垂直(制表)跳行
02H	打印字符、图等,增宽(×1,×2,×3,×4)	09H	恢复 ASCII 代码和清除输入缓冲区命令
03H	打印字符、图等,宽和高同时增加(×1,×2,×3,×4)	0AH	一个空位后回车换行
04H	字符行间距更换/定义	0BH~0CH	无效
05H	用户自定义字符点阵	0DH	回车换行/命令结束
06H	驻留代码字符点阵式样更换	0EH	重复打印同一字符命令
07H	水平(制表)跳区	0FH	打印位点阵图命令

(3) 命令非法时的出错提示

当向 TP_μP-40A 输入非法命令时, TP_μP-40A 打印出出错代码。其意义为:

ERROR 0:放大系数出界,即放大倍数是 1,2,3 和 4 以外的数字。此错误出现在 01H,02H,03H 命令时。

ERROR 1:定义代码非法。用户自定义代码不是 10H~1FH。

ERROR 2:非法换码命令。换码命令只能用 10H~1FH 去代换驻留字符代码,否则为非法。

ERROR 3:绘图命令错误。指定图形字节数为 0 或大于 240。

ERROR 4:垂直制表命令错误。指定空行数为 0。

4. TP_μP-40A/16A 与 MCS-51 单片机接口设计

TP_μP-40A/16A 是智能打印机,内部控制电路由单片机构成,在输入电路中有锁存器,在输出电路中有三态门控制。因此可以直接与单片机相接。

TP_μP-40A/16A 没有读、写信号,只有握手线 \overline{STB} 、BUSY(或 \overline{ACK}),接口电路如图 10-29 所示。

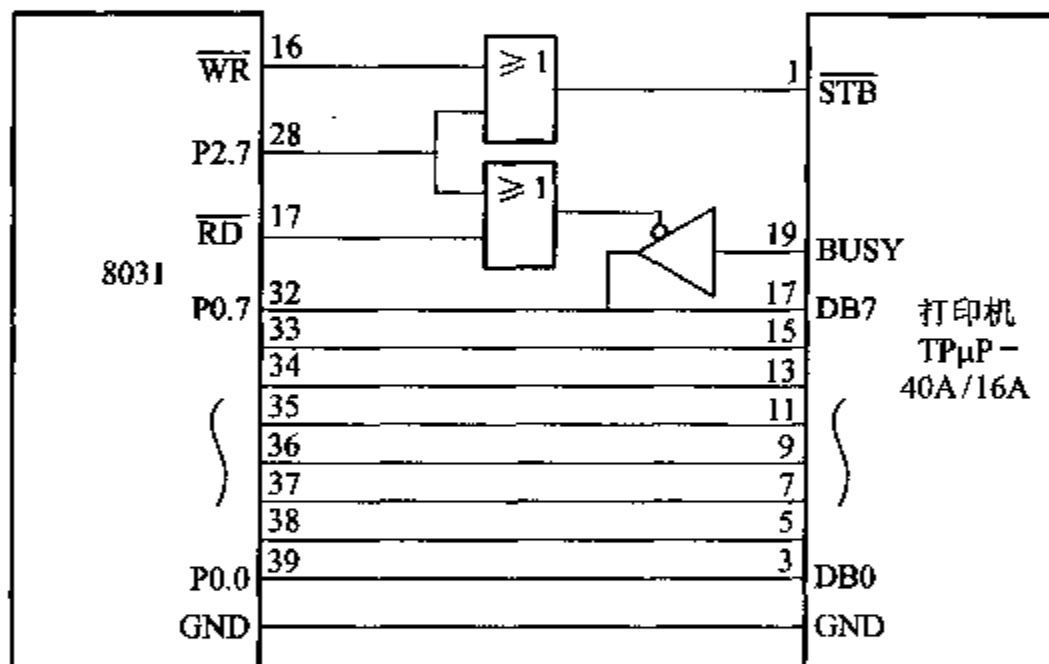


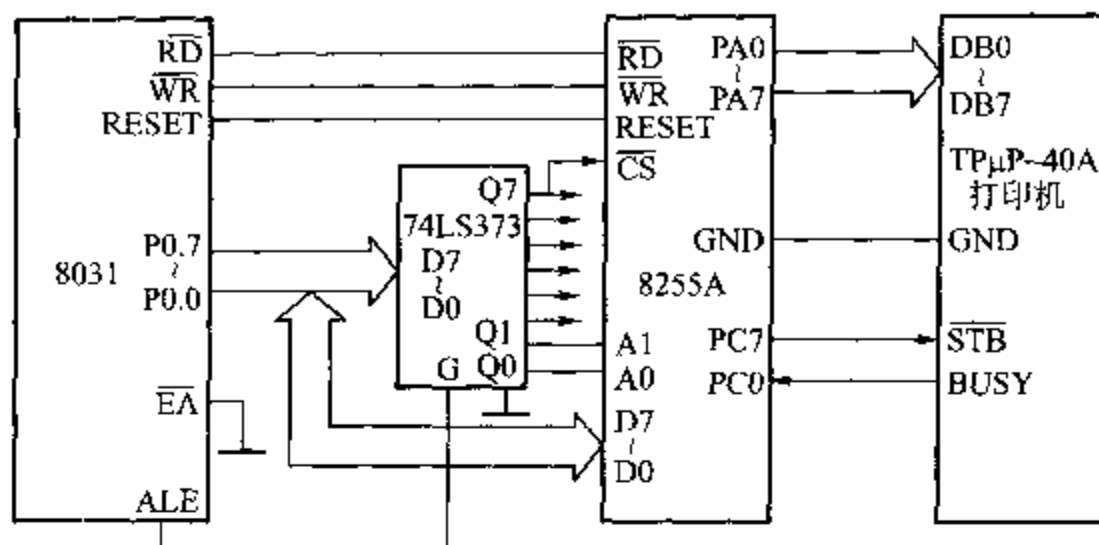
图 10-29 TP_μP-40A/16A 与 8031 的接口

用 1 根地址线(图中使用 P2.7 即 A15)来控制写选通信号 \overline{STB} 和读取 BUSY 引脚状态。

图 10-30 是通过扩展的并行 I/O 口连接的打印机接口电路。图中的扩展 I/O 口为 8255A 的 PA 口,采用了查询法,即通过读 8255A 的 PC0 引脚的状态来判断送给打印机的 1 B 数据是否处理完毕。也可用中断法(BUSY 直接与单片机的 P3.3 引脚相连)。

例 10-2 把 MCS-51 单片机内部 RAM 3FH~4FH 单元中的 ASCII 码数据送到打印机。8255A 设置为方式 0,即端口 A 与端口 C 的上半部为输出方式。端口 C 的下半部为输入方式。

打印程序 PRINT 如下:

图 10-30 TP μ P 40A/16A 与 8031 扩展的 I/O 连接

```

PRINT:      MOV     R0, #7FH      ;控制口地址→R0
            MOV     A, #81H      ;8255A 控制字→A
            MOVX    @R0, A       ;控制字→控制口
            MOV     R1, #3FH      ;数据区首地址→R1
            MOV     R2, #0FH      ;R2 作打印数据个数的计数器用
LOOP:       MOV     A, @R1       ;打印数据单元中内容→A
            INC     R1           ;指向下一个数据单元
            MOV     R0, #7CH      ;8255A 的端口 A 地址→R0
            MOVX    @R0, A       ;打印数据送 8255A 的端口 A 并
                                ;锁存
            MOV     R0, #7FH      ;8255A 的控制口地址→R0
            MOV     A, #0EH      ;PC7 的复位控制字→A
            MOVX    @R0, A       ;PC7 = 0
            MOV     A, #0FH      ;PC7 的置位控制字→A
            MOVX    @R0, A       ;PC7 由 0 变 1
LOOP1:      MOV     R0, #7EH      ;C 口地址→R0
            MOVX    A, @R0       ;读入 C 口的值
            ANL     A, #01H      ;屏蔽掉 C 口的高 7 位, 只留 PC0 位
            JNZ     LOOP1        ;查询 PC0 即 BUSY 的状态, 如为 1
                                ;跳 LOOP1
            DJNZ    R2, LOOP      ;未打完, 循环

```

10.5.2 MCS-51 与 GP16 微型打印机的接口

1. GP16 微型打印机的接口信号

GP16 为智能微型打印机, 机芯为 Model-150-II 16 行针打, 控制器由

8039 单片机(MCS-48 系列)系统构成。GP16-II 为 GP16 的改进型,控制器为 8031 单片机。

GP16 打印机的接口信号如图 10-31 所示。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
+5 V	+5 V	I/O0	I/O1	I/O2	I/O3	I/O4	I/O5	I/O6	I/O7	\overline{CS}	\overline{WR}	\overline{RD}	BUSY	地	地

图 10-31 GP16 打印机的接口信号

各信号的功能如下:

I/O0~I/O7:双向三态数据总线,是 CPU 与 GP16 打印机之间命令、状态和数据信息传输线。

\overline{CS} :设备选择线。

\overline{WR} 、 \overline{RD} :读、写信号线。

BUSY:打印机状态输出线,BUSY 输出高电平表示 GP16 处于忙状态,不能接收 CPU 发出的命令或数据。BUSY 状态输出线可供 CPU 查询或作中断请求线。

GP16 控制器具有数据锁存器,与单片机接口十分方便。

2. GP16 的打印命令和工作方式

(1) 打印命令及打印方式

GP16 的打印命令占 2 B,其格式如下:

第一个字节	D7...D4	D3...D0
	操作码	点行数 n
第二个字节	D7...D0	
	打印行数 NN	

GP16 为微型针打,字符本身占据 7 个点行。命令字中的点行数 n 是选择字符行之间行距的参数,若 $n=10$,则行距为 3 个点行数;打印行数是执行本条命令时打印(或空走纸)的字符行数。打印点行数应大于或等于 8。

表 10-7 是 GP16 的命令编码表。

表 10-7 GP16 的命令编码表

D7	D6	D5	D4	命令功能
1	0	0	0	空走纸
1	0	0	1	打印字符串
1	0	1	0	十六进制数据打印
1	0	1	1	图形打印

① 空走纸命令($8nNNH$)。执行空走纸命令时,打印机自动空走纸 $N \times n$ 点行,其间忙状态(BUSY)置位,执行完后清零。

② 打印字符串($9nNNH$)。执行打印字符串命令后,打印机等待 CPU 写入

字符数据,当接收完 16 个字符(1 行)后,转入打印。打印 1 行需时约 1 s。若收到非法字符作空格处理。若收到换行(0AH),作停机处理,打完本行即停止打印。当打印完规定的 NNH 行数后,忙状态(BUSY)清零。

GP16 打印机可打印的字符请见 GP16 的技术说明书。

③ 十六进制数据打印(A_nNNH)。本命令通常用来直接打印内存数据。当 GP16 接收到数据打印命令后,把 CPU 写入的数据字节分 2 次打印,先打印高 4 位,后打印低 4 位。1 行打印 1 B 数据。行首为相对地址,其格式如下:

00H: ×× ×× ×× ××

04H: ×× ×× ×× ××

08H: ×× ×× ×× ××

0CH: ×× ×× ×× ××

10H: ×× ×× ×× ××

...

④ 图形打印(B_nNNH):GP16 接收到 CPU 的图形打印命令和规定的行数以后,等待主机送来 1 行 96 B 的数据进行打印,把这些数据所确定的图形打印出来,然后再接受 CPU 的图形数据,直到规定的行数打印完为止。

(2) 状态字与工作状态

GP16 有 1 个状态字可供 CPU 查询。状态字格式如下:

D7	D6	D5	D4	D3	D2	D1	D0
错	—	—	—	—	—	—	忙

D7 为错误位。当接收到非法命令时置 1,接到正确命令后清 0。

D0 为忙(BUSY)位。当 CPU 输入的数据、命令没处理完时或处于自检状态时均置 1。空闲时置 0。

3. MCS-51 单片机和 GP16 的接口

由于 GP16 内部的控制电路中有三态锁存器,在 \overline{CS} 和 \overline{WR} 引脚信号控制下能锁存数据总线上的数据,三态门又能与 CPU 实现隔离。故 GP16 可以直接与 MCS-51 数据总线相连而不须外加锁存器。图 10-32 即为 GP16 与 8031 数据总线相连的接口电路。

图中 BUSY 接 $\overline{INT1}$ (P3.3),因此,接口电路直接可用于中断方式($\overline{INT1}$)。如要以查询方式工作时,BUSY 可以不连接,通过查询状态字来获取 BUSY 的状态。

如果使用其他 I/O 或扩展 I/O 口,只须将 P0 口线换成其他 I/O 或扩展 I/O 口即可。

按照图 10-32 的连接,GP16 的打印机地址为 7FFFH,读取 GP16 状态字

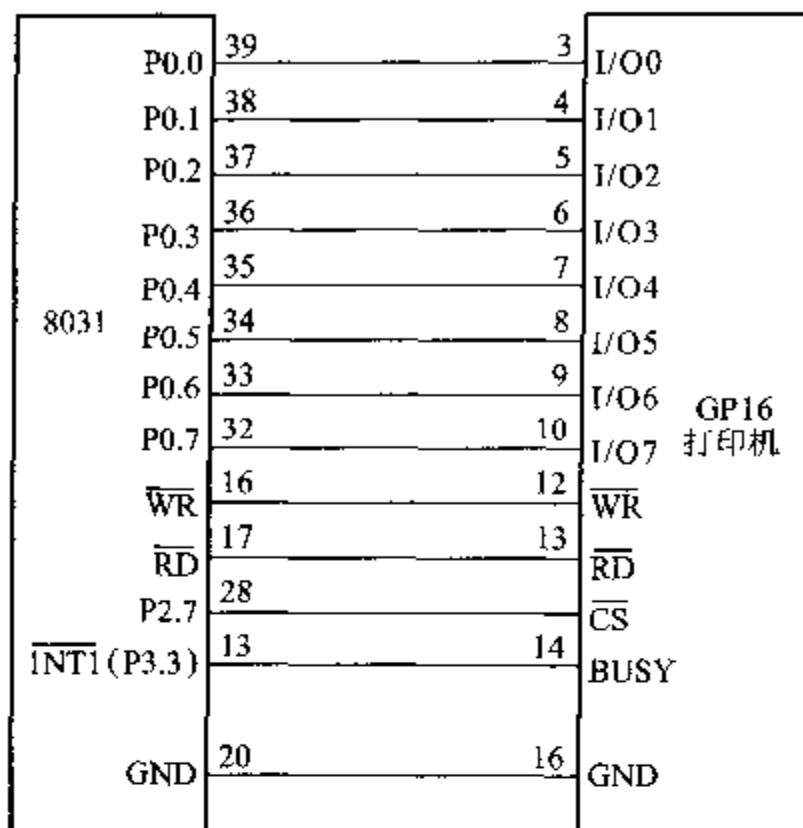


图 10-32 GP16 与 8031 数据总线的接口

时,8031 执行下列程序段:

```
MOV    DPTR, #7FFFH
MOVX   A, @DPTR
```

将命令或数据写入 GP16 时,8031 执行下列程序段:

```
MOV    DPTR, #7FFFH
MOV    A, #DATA/COMMAND
MOVX   @DPTR, A
```

10.6 MCS-51 单片机与 BCD 码拨盘的接口设计

10.6.1 BCD 码拨盘

在某些单片机系统中,有时需要输入一些控制参数,这些参数一经设定将维持不变,除非给系统断电后重新设定。这时使用数字拨盘既简单直观,又方便可靠。

拨盘种类很多,但使用最方便的拨盘是十进制输入,BCD 码输出的 BCD 码拨盘。这种拨盘如图 10-33 所示,图中为 4 片 BCD 码拨盘拼接的 4 位十进制输入拨盘组。每片拨盘具有 0~9 10 个位置,每个位置都有相应的数字显示,代

表拨盘输入的十进制数。因此,每片拨盘可代表 1 位十进制数。需要几位十进制数即可选择几片 BCD 码拨盘拼接。

BCD 码拨盘后面有 5 个接点,其中 A 为输入控制线,另外 4 根是 BCD 码输出线。拨盘拨到不同位置时,输入控制线 A 分别与 4 根 BCD 码输出线中的某根或某几根接通,其接通的 BCD 码输出线状态正好与拨盘指示的十进制数相一致。

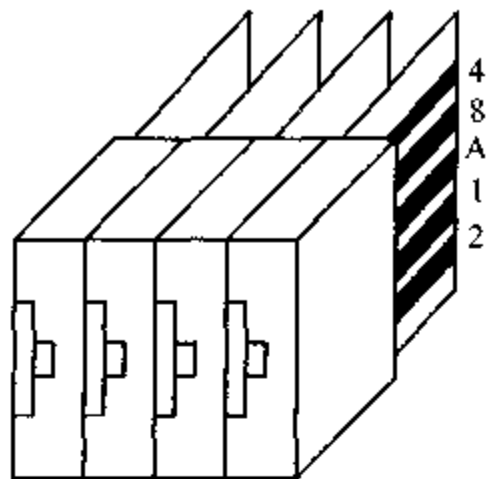


图 10-33 1 位 BCD 码拨盘组

表 10-8 为 BCD 码拨盘的输入/输出状态表。

表 10-8 BCD 码拨盘的输入/输出状态

拨盘输入	控制端 A	输出状态			
		8	4	2	1
0	1	0	0	0	0
1	1	0	0	0	1
2	1	0	0	1	0
3	1	0	0	1	1
4	1	0	1	0	0
5	1	0	1	0	1
6	1	0	1	1	0
7	1	0	1	1	1
8	1	1	0	0	0
9	1	1	0	0	1

*:输出状态为 1 时,表示该输出线与 A 接通。

10.6.2 BCD 码拨盘与单片机的接口

1. 单片机与单片 BCD 码拨盘的接口

单片 BCD 码拨盘可以与任何 1 个 4 位的 I/O 口或扩展的 I/O 口相连,以输入 BCD 码,A 端接 +5 V。为了使输出端在不与控制端 A 相连时有确定的电平,常将 8,4,2,1 输出端通过电阻拉低。图 10-34 是 8031 通过 P1.0~P1.3 与单片 BCD 码拨盘的接口电路。

控制端 A 接 +5 V,当拨盘拨至某输入十进制数时,相应的 8,4,2,1 有效端输出高电平(如拨至“6”时,4,2 端为有效端),无效端为低电平。这时拨盘输出的 BCD 码为正逻辑(原码)。如果控制端 A 接地,8,4,2,1 输出端通过电阻上拉至高电平时,拨盘输出的 BCD 码为负逻辑(反码)。

2. 多片 BCD 码拨盘与单片机的接口

实际应用系统中,有时可能输入不止 1 位十进制数,这时应将多片 BCD 码拨盘拼接在一起,形成 BCD 码拨盘组,以实现多位十进制数的输入。如果还是按图 10-34 的接法, N 位十进制拨盘需占用 $4 \times N$ 根 I/O 口线,为了减少 I/O 口线占用数量,可将拨盘的输出线分别通过 4 个与非门与单片机的 I/O 口相连,而每片拨盘的控制端 A 不

再接 +5 V 或地,而是分别与 I/O 口线相连,用来控制选择多片拨盘中的任意一片。这时, N 位十进制拨盘,用 N 片 BCD 码拨盘拼成时只需占用 $4 + N$ 根 I/O 口线。图 10-35 所示即为通过 P1 口与 4 片 BCD 码拨盘相连的 4 位 BCD 码输入电路。

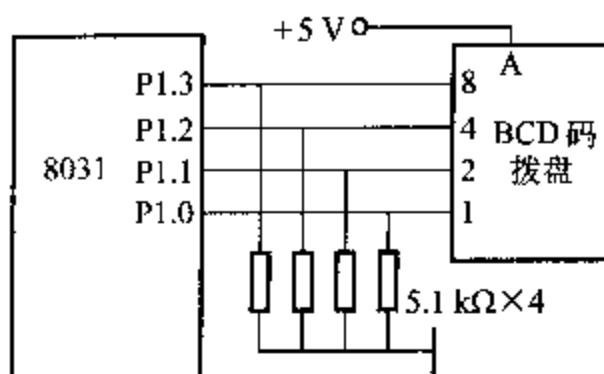


图 10-34 单片 BCD 码拨盘与 8031 的接口

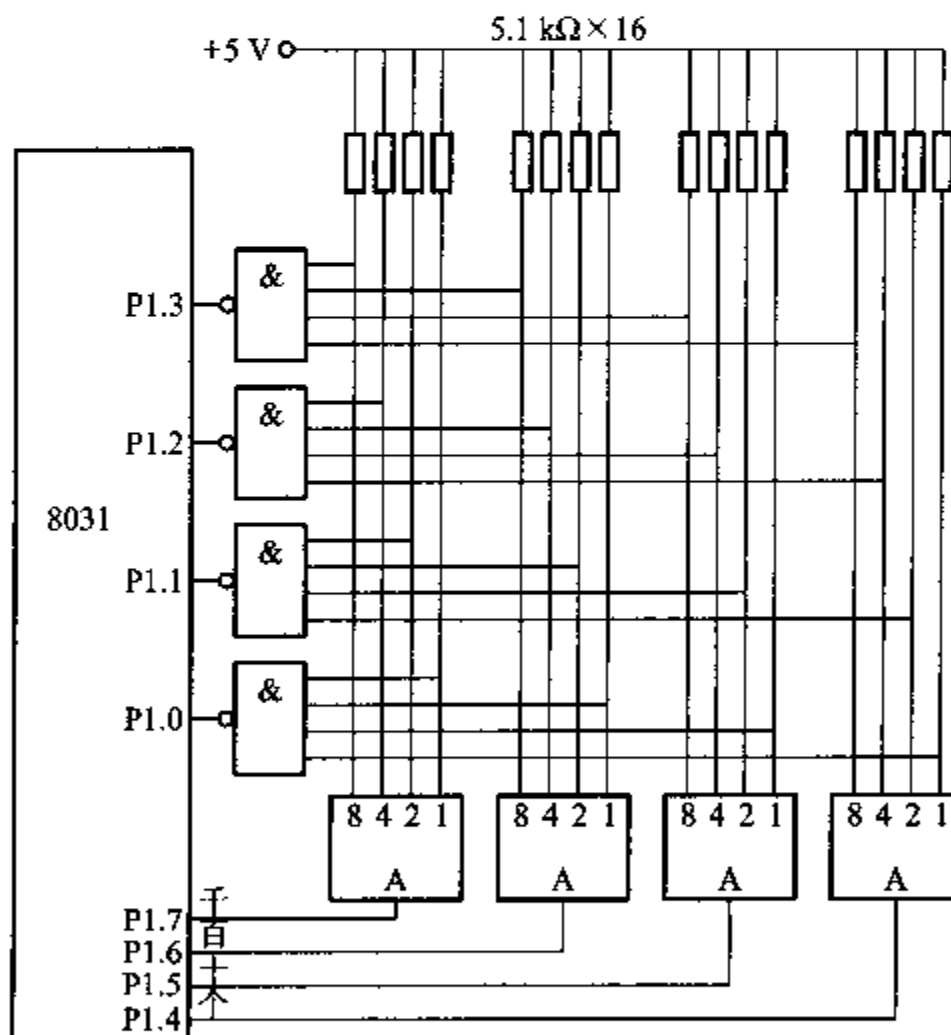


图 10-35 4 片 BCD 码拨盘与 8031 接口

4 片拨盘的 BCD 码输出相同端分别接入同一个与非门。4 个与非门输出 8, 4, 2, 1 端分别接 P1.3, P1.2, P1.1, P1.0。其余的 P1.7, P1.6, P1.5, P1.4 分别与千、百、十、个位 BCD 码拨盘的控制端相连。当选中某位时,该位的控制端置 0,其他 3 个控制端置 1。例如选中千位时,P1.7 置 0,P1.4~P1.6 置 1,此时 4

个与非门所有其他位连接的输入端均为 1 状态,因此 4 个与非门输出的状态完全取决于千位 BCD 拨盘输出状态。由于该位的控制端置 0,因此,拨盘所置之数输出为 BCD 反码,通过与非门输出为该千位数的 BCD 码。

下面以图 10-35 为例,介绍 BCD 码拨盘输入子程序。在执行拨盘输入程序之前,各位的 BCD 码拨盘已拨好数码,例如为 9 345,这时,每位 BCD 码输出端上有相应的数字与 A 接通。

本程序将读入的 4 位 BCD 码按千、百、十、个位依次存放在 8031 片内 RAM 的 30H~33H 单元中,每个地址单元的高 4 位为 0,低 4 位为 BCD 码。

程序清单如下:

```
RDS:  MOV R0, #30H    ;初始化,存放单元首址
      MOV R2, #7FH    ;P1 口高 4 位置控制字及低 4 位置输入方式
      MOV R3, #04H    ;读入 4 个 BCD 码
LOOP:  MOV A, R2
      MOV P1, A        ;P1 口送控制字及低 4 位置输入方式
      MOV A, P1        ;读入 BCD 码
      ANL A, #0FH      ;屏蔽高 4 位
      MOV @R0, A       ;送入存储单元
      INC R0           ;指向下一个存储单元
      MOV A, R2        ;准备下一片拨盘的控制端置 0
      RR A
      MOV R2, A
      DJNZ R3, LOOP    ;未读完返回
      RET              ;读完结束
```

思考题及习题

1. 为什么要消除按键的机械抖动? 消除按键的机械抖动的方法有哪几种? 原理是什么?
2. 判断下列说法是否正确?
 - (1) 8279 是 1 个用于键盘和 LED(LCD)显示器的专用接口芯片。
 - (2) 在单片机与微型打印机的接口中,打印机的 BUSY 信号可作为查询信号或中断请求信号使用。
 - (3) 为给以扫描法方式工作的 8×8 键盘提供接口电路,在接口电路中只需要提供 2 个输入口和 1 个输出口。
 - (4) LED 的字型码是固定不变的。
3. LED 的静态显示方式与动态显示方式有何区别? 各有什么优缺点?
4. 写出表 10-1 中仅显示小数点“.”的段码。

5. 说明矩阵式键盘按键按下的识别原理。
6. 对于图 10-11 的键盘,采用线反转法原理来编写识别某一按键被按下并得到其键号的程序。
7. 键盘有哪 3 种工作方式,它们各自的工作原理及特点是什么?
8. 根据图 10-11 的电路,编写在 6 个 LED 显示器上轮流显示“1,2,3,4,5,6”的显示程序。
9. 根据图 10-17 的接口电路编写在 8 个 LED 上轮流显示“1,2,3,4,5,6,7,8”的显示程序,比较一下与上一题显示程序的区别。
10. 8279 中的扫描计数器有 2 种工作方式,这 2 种工作方式各应用在什么场合?
11. 简述 TP μ P-40A/16A 微型打印机的 Centronics 接口的主要信号线的功能,与 MCS-51 单片机相连接时,如何连接这几条控制线?
12. 如果把图 10-30 中打印机的 BUSY 线断开,然后与 8031 的 $\overline{\text{INT0}}$ 线相接,请简述电路的工作原理并编写把以 20H 为起始地址的连续 20 个内存单元中的内容输出打印的程序。
13. 根据图 10-14,8155H 与 32 键的键盘相连接,编写程序实现如下功能:用 8155H 的定时器定时,每隔 1s 读 1 次键盘,并将其读入的键值存入 8155H 片内 RAM 中 30H 开始的单元中。
14. 采用 8279 芯片的键盘/显示器接口方案,与本章介绍的其他键盘/显示器的接口方案相比,有什么特点?

第 11 章 MCS-51 单片机与 D/A 转换器、A/D 转换器的接口

在单片机的应用系统中,被测量对象的有关变量,如温度、压力、流量、速度等非电物理量,须经传感器转换成连续变化的模拟电信号(电压或电流),这些模拟电信号必须转换成数字量后才能在单片机中用软件进行处理。单片机处理完毕的数字量,也常常需要转换为模拟信号。实现模拟量转换成数字量的器件称为 A/D 转换器(ADC),数字量转换成模拟量的器件称为 D/A 转换器(DAC)。

现在已经有很多介绍 A/D、D/A 转换技术与原理的专著,本章不再赘述。在大规模集成电路技术飞速发展的今天,对于单片机应用系统的设计者来说,只需要合理的选用商品化的大规模 ADC、DAC 集成电路芯片,了解它们的引脚、功能以及与单片机的接口设计方法。本章将着重从应用的角度,介绍几种典型的 ADC、DAC 集成电路芯片,以及它们同 MCS-51 的硬件接口设计及软件设计。

11.1 MCS-51 与 DAC 的接口

11.1.1 D/A 转换器概述

1. 概述

D/A(数/模)转换器输入的是数字量,经转换后输出的是模拟量。转换过程是先将 MCS-51 送到 D/A 转换器的各位二进制数按其权的大小转换为相应的模拟分量,然后再以叠加方法把各模拟分量相加,其和就是 D/A 转换的结果。

使用 D/A 转换器时,要注意区分 D/A 转换器的输出形式和内部是否带有锁存器。

(1) 电压与电流输出形式

D/A 转换器有两种输出形式,一种是电压输出形式,即给 D/A 转换器输入的是数字量,而输出为电压。另一种是电流输出形式,即输出为电流。在实际应用中,对于电流输出的 D/A 转换器,如需要模拟电压输出,可在其输出端加一个由运算放大器构成的电流/电压转换电路,将电流输出转换为电压输出。

(2) D/A转换器内部是否带有锁存器

由于D/A转换是需要一定时间的,在这段时间内D/A转换器输入端的数字量应保持稳定,为此应当在D/A转换器数字量输入端的前面设置锁存器,以提供数据锁存功能。根据转换器芯片内是否带有锁存器,可以把DAC分为内部无锁存器的和内部有锁存器的两类。

① 内部无锁存器的D/A转换器

这种D/A转换器由于不带锁存器而内部结构简单,它们可与MCS-51的P1、P2口直接相接,因为P1口和P2口的输出有锁存功能。但是当与P0口相接时,由于P0口的特殊性,需要在转换器芯片的前面增加锁存器。

② 内部带有锁存器的D/A转换器

这种D/A转换器的芯片内部不但有锁存器,而且还包括地址译码电路,有的还具有双重或多重的数据缓冲电路,可与MCS-51的P0口直接相接。

2. 主要技术指标

D/A转换器的指标很多,使用者最关心的几个指标如下。

(1) 分辨率

分辨率指输入给D/A转换器的单位数字量变化引起的模拟量输出的变化,是输出对输入量变化敏感程度的描述。通常定义为输出满刻度值与 2^n 之比(n 为D/A转换器的二进制位数)。显然,二进制位数越多,分辨率越高,即D/A转换器对输入量变化的敏感程度越高。例如,若满量程为10V,根据分辨率定义则分辨率为 $10\text{V}/2^n$ 。设8位D/A转换,即 $n=8$,分辨率为 $10\text{V}/2^8=39.1\text{mV}$,即二进制数最低位的变化可引起输出的模拟电压变化39.1mV,该值占满量程的0.391%,常用符号1LSB表示。

同理:	10位D/A转换	1LSB=9.77mV=0.1%满量程
	12位D/A转换	1LSB=2.44mV=0.024%满量程
	14位D/A转换	1LSB=0.61mV=0.006%满量程
	16位D/A转换	1LSB=0.076mV=0.00076%满量程

使用时,应根据对D/A转换器分辨率的需要来选定D/A转换器的位数。

(2) 建立时间

建立时间是描述D/A转换器转换快慢的一个参数,用于表明转换速度。其值为从输入数字量到输出达到终值误差 $\pm(1/2)$ LSB(最低有效位)时所需的时间。输出形式为电流的转换时间较短,而输出形式为电压的转换器,由于要加上完成I/V转换的运算放大器的延迟时间,因此建立时间要长一些。快速的D/A转换器的建立时间可达 $1\mu\text{s}$ 以下。

(3) 精度

理想情况下,精度与分辨率基本一致,位数越多精度越高。但由于电源电

压、参考电压、电阻等各种因素存在着误差。严格讲精度与分辨率并不完全一致。只要位数相同,分辨率则相同,但相同位数的不同转换器精度会有所不同。例如,某种型号的 8 位 DAC 精度为 $\pm 0.19\%$,而另一种型号的 8 位 DAC 精度为 $\pm 0.05\%$ 。

11.1.2 MCS-51 与 8 位 DAC0832 的接口

1. DAC0832 芯片介绍

(1) DAC0832 的特性

美国国家半导体公司的 DAC0832 芯片是具有 2 个输入数据寄存器的 8 位 DAC,它能直接与 MCS-51 单片机相连接,其主要特性如下:

- ① 分辨率为 8 位;
- ② 电流输出,稳定时间为 $1\mu\text{s}$;
- ③ 可双缓冲输入、单缓冲输入或直接数字输入;
- ④ 单一电源供电($+5\sim+15\text{V}$);
- ⑤ 低功耗, 20mW 。

(2) DAC0832 的引脚及逻辑结构

DAC0832 的引脚如图 11-1 所示。

DAC0832 的逻辑结构如图 11-2 所示。

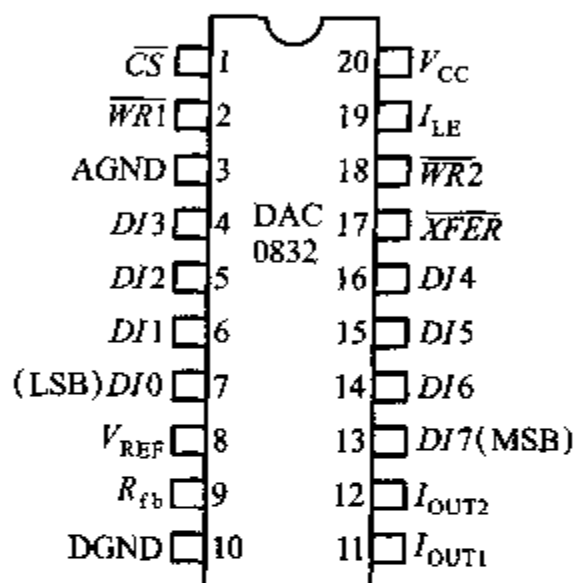


图 11-1 DAC0832 的引脚

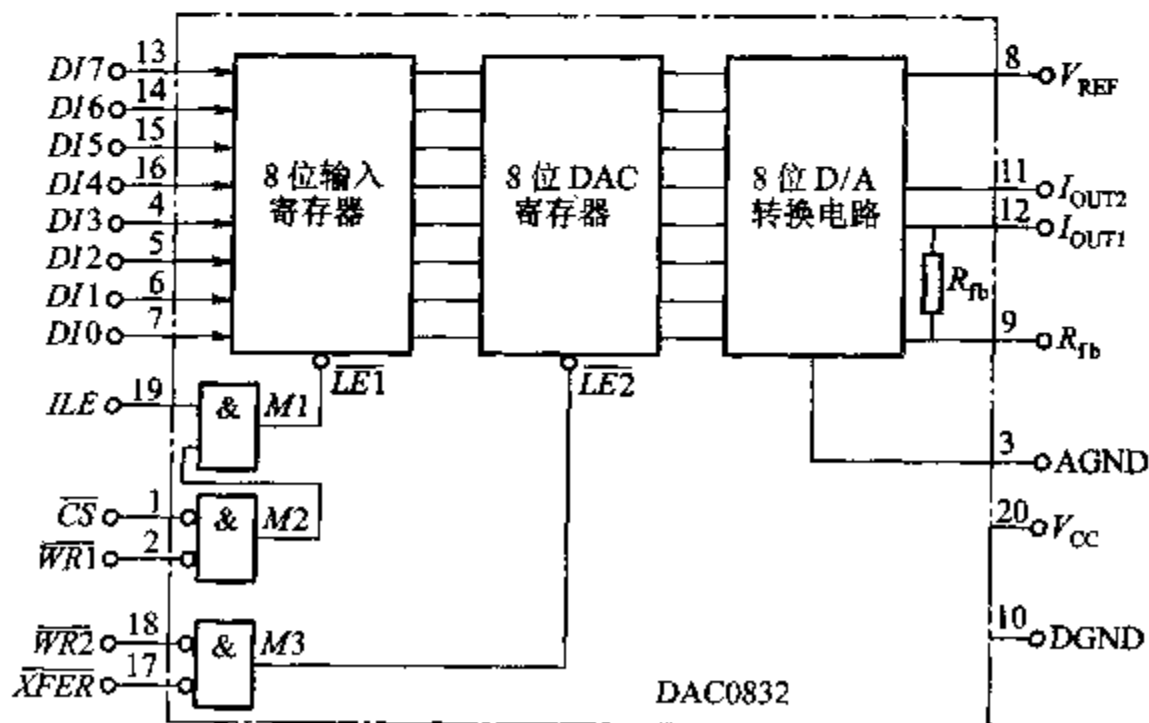


图 11-2 DAC0832 的逻辑结构

各引脚的功能如下:

DI0~DI7: 8 位数字信号输入端, 与单片机的数据总线相连, 用于接收单片机送来的待转换的数字量, DI7 为最高位。

\overline{CS} : 片选端, 当 \overline{CS} 为低电平时, 本芯片被选中。

ILE: 数据锁存允许控制端, 高电平有效。

$\overline{WR1}$: 第一级输入寄存器写选通控制, 低电平有效。当 $\overline{CS}=0$ 、 $ILE=1$ 、 $\overline{WR1}=0$ 时, 数据信号被锁存到第一级 8 位输入寄存器中。

\overline{XFER} : 数据传送控制, 低电平有效。

$\overline{WR2}$: DAC 寄存器写选通控制端, 低电平有效。当 $\overline{XFER}=0$ 、 $\overline{WR2}=0$ 时, 输入寄存器状态传入 8 位 DAC 寄存器中。

I_{OUT1} : D/A 转换器电流输出 1 端, 输入数字量全 1 时, I_{OUT1} 输出最大, 输入数字量全为 0 时, I_{OUT1} 输出最小。

I_{OUT2} : D/A 转换器电流输出 2 端, $I_{OUT2} + I_{OUT1} = \text{常数}$ 。

R_{fb} : 外部反馈信号输入端, 内部已有反馈电阻 R_{fb} , 根据需要也可外接反馈电阻。

Vcc: 电源输入端, 可在 $+5 \sim +15 \text{ V}$ 范围内。

DGND: 数字信号地。

AGND: 模拟信号地, 最好与基准电压共地。

DAC0832 内部的 3 部分电路如图 11-2 所示。8 位输入寄存器用于存放 CPU 送来的数字量, 使输入数字量得到缓冲和锁存, 由 $\overline{LE1}$ 端加以控制; 8 位 DAC 寄存器用于存放待转换的数字量, 由 $\overline{LE2}$ 端控制; 8 位 D/A 转换电路由 8 位 T 型电阻网络和电子开关组成, 电子开关受 8 位 DAC 寄存器输出的数字量控制, T 型电阻网络能输出和数字量成正比的模拟电流。因此, DAC0832 通常需要外接运算放大器, 进行电流/电压转换, 才能得到模拟输出电压。

2. DAC 的应用

MCS-51 与 DAC0832 的接口常和 DAC 的具体应用有关, 因此, 我们先以 DAC0832 为例, 讨论有关 DAC 的应用问题, 然后介绍它与 MCS-51 的接口。

(1) 用作单极性电压输出

在需要单极性模拟电压环境下, 我们可以采用图 11-5 或图 11-9 所示接线。由于 DAC0832 是 8 位的 D/A 转换器, 故可得输出电压 v_o 与输入数字量 B 的关系为:

$$v_o = -B \frac{V_{REF}}{256}$$

式中, $B = b_7 \cdot 2^7 + b_6 \cdot 2^6 + \dots + b_1 \cdot 2^1 + b_0 \cdot 2^0$; $V_{REF}/256$ 为一常数。

显然, v_o 和输入数字量 B 成正比。B 为 0 时, v_o 也为 0, 输入数字量为 255

时, v_O 为最大值, 输出电压为单极性。

(2) DAC 用作双极性电压输出

在需要用到双极性电压输出的场合下, 可以采用图 11-3 所示接线。图中, DAC0832 的数字量由 CPU 送来, A_1 和 A_2 均为运算放大器, v_O 通过 $2R$ 电阻反馈到运算放大器 A_2 输入端, 其他如图所示。G 点为虚拟地, 故由基尔霍夫定律列出方程组, 并解得:

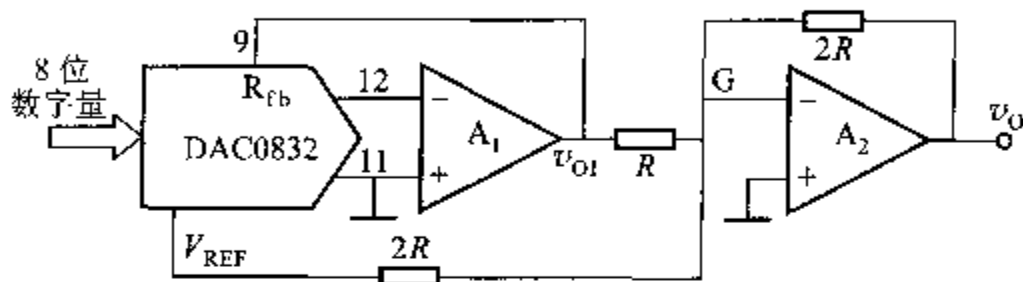


图 11-3 双极性 DAC 的接法

$$v_O = (B - 128) \frac{V_{REF}}{128}$$

由上式可知, 在选用 $+V_{REF}$ 时, 若输入数字量最高位 b_7 为 1, 则输出模拟电压 v_O 为正; 若输入数字量最高位为 0, 则输出模拟电压 v_O 为负。在选用 $-V_{REF}$ 时, v_O 输出值正好和选用 $+V_{REF}$ 时极性相反。

(3) DAC 用作程控放大器

DAC 还可以用作程控放大器。其电压放大倍数可由 CPU 通过程序设定。图 11-4 为用作程控电压放大器的 DAC 接线。由图可见, 需要放大的电压 v_i 和反馈输入端 R_{fb} 相接, 运算放大器输出 v_O 还作为 DAC 的基准电压 V_{REF} , 数字量由 CPU 送来, 其余如图所示。DAC0832 内部 I_O 一边和 T 型电阻网络相连, 另一边又通过内部反馈电阻 R_{fb} 和 v_i 相通, 故可得到 DAC 的输出和输入之间的关系:

$$v_O = -\frac{v_i}{B} \cdot \frac{R}{R_{fb}} \cdot 256$$

选 $R = R_{fb}$, 则上式变为

$$v_O = -\frac{v_i}{B} \cdot 256$$

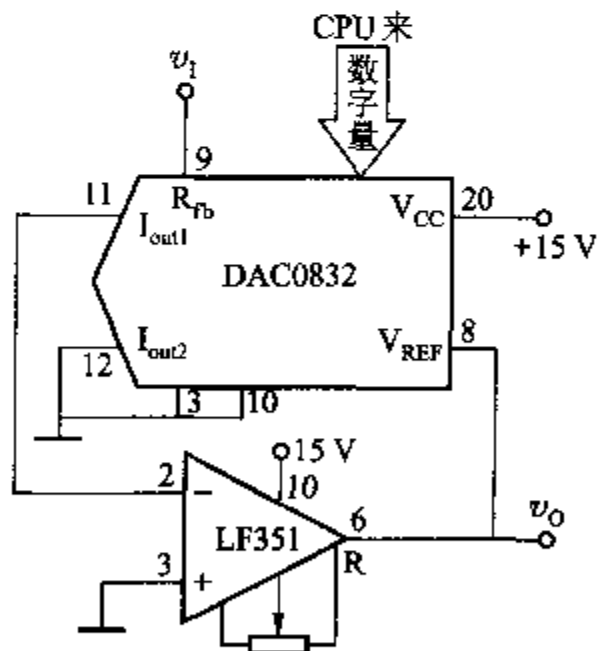


图 11-4 DAC0832 用作程控放大器

上式中的 $256/B$ 看作放大倍数。但输入的数字量 B 不得为 0, 否则放大倍数为无限大, 此时放大器处于饱和状态。

3. MCS-51 与 DAC0832 的接口电路

设计 MCS-51 与 DAC0832 的接口电路时,常用的是单缓冲方式或双缓冲方式的单极性输出。

(1) 单缓冲方式

单缓冲方式是指 DAC0832 内部的两个数据缓冲器有一个处于直通方式,另一个处于受 MCS-51 控制的锁存方式。在实际应用中,如果只有一路模拟量输出,或虽是多路模拟量输出但并不要求多路输出同步的情况下,就可采用单缓冲方式。

单缓冲方式的接口电路如图 11-5 所示。

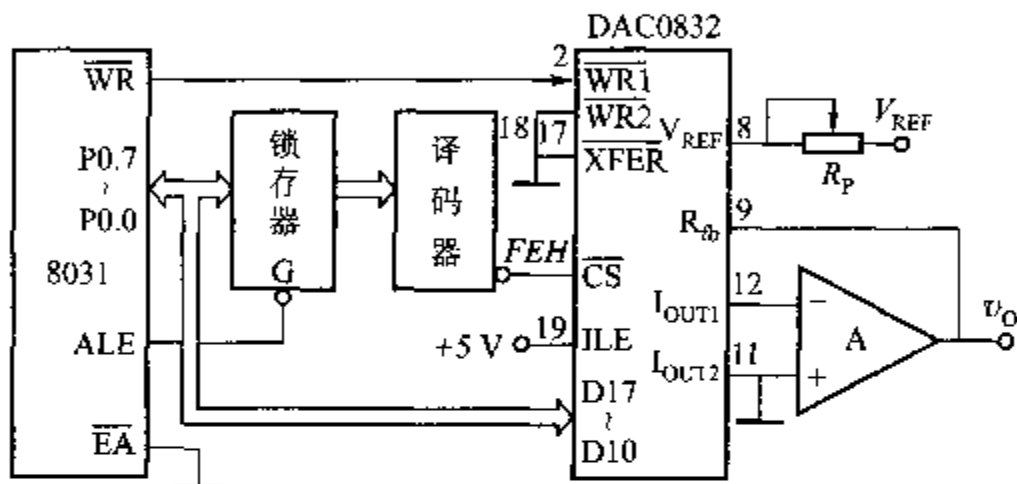


图 11-5 单缓冲方式下的 DAC0832

图中可见,WR2和XFER接地,故 DAC0832 的 8 位 DAC 寄存器(见图 11-2)工作于直通方式。8 位输入寄存器受CS和WR1端控制,而且CS由译码器输出端 FEH 送来(也可由 P2 口的某一根口线来控制)。因此,8031 执行如下两条指令就可在WR1和CS上产生低电平信号,使 DAC0832 接收 8031 送来的数字量。

```
MOV    R0, #0FEH    ;DAC 地址 FEH→ R0
```

```
MOVX   @R0, A        ;8031 的 WR 和译码器 FEH 输出端有效
```

现举例说明单缓冲方式下 DAC0832 的应用。

例 11-1 DAC0832 用作波形发生器。试根据图 11-5,分别写出产生锯齿波、三角波和矩形波的程序。

在图 11-5 中,运算放大器 A 输出 v_o 直接反馈到 R_b ,故这种接线产生的模拟输出电压是单极性的。产生上述三种波形的参考程序如下:

① 锯齿波的产生

```

ORG    2000H
START:  MOV    R0, #0FEH    ;DAC 地址 FEH→ R0
        MOV    A, #00H      ;数字量→ A
LOOP:   MOVX   @R0, A        ;数字量→ D/A 转换器
        INC    A             ;数字量逐次加 1
        SJMP   LOOP
    
```

当输入数字量从 0 开始, 逐次加 1 进行 D/A 变换, 模拟量与其成正比输出。

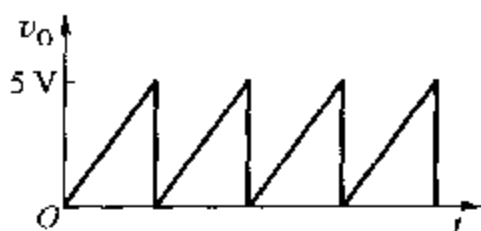


图 11-6 DAC 的锯齿波输出

当 $A = FFH$ 时, 再加 1 则溢出清 0, 模拟输出又为 0, 然后又重新重复上述过程, 如此循环, 输出波形就是 1 个锯齿波, 如图 11-6 所示。但实际上每一个上升斜边要分成 256 个小台阶, 每个小台阶暂留时间为执行程序中的 3 条指令所需要的时间。因此在上述程序“INC A”指令后插入 NOP 指令

或延时程序, 则可以改变锯齿波的频率。

② 三角波的产生

	ORG	2000H	
START:	MOV	R0, #0FEH	
	MOV	A, #00H	
UP:	MOVX	@R0, A	; 三角波上升边
	INC	A	
	JNZ	UP	
DOWN:	DEC	A	
	MOVX	@R0, A	; A=0 时再减 1 又为 FFH
	JNZ	DOWN	; 三角波下降边
	SJMP	UP	

输出的三角波如图 11-7 所示。

③ 矩形波的产生

	ORG	2000H	
START:	MOV	R0, #0FEH	
LOOP:	MOV	A, #data1	
	MOVX	@R0, A	; 置矩形波上限电平
	LCALL	DELAY1	; 调用高电平延时程序
	MOV	A, #data2	
	MOVX	@R0, A	; 置矩形波下限电平
	LCALL	DELAY2	; 调用低电平延时程序
	SJMP	LOOP	; 重复进行下一个周期

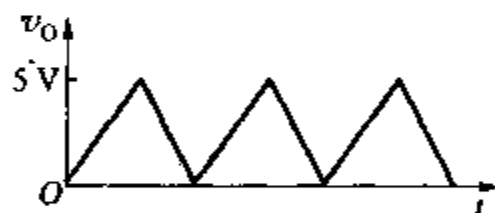


图 11-7 DAC 的三角波输出

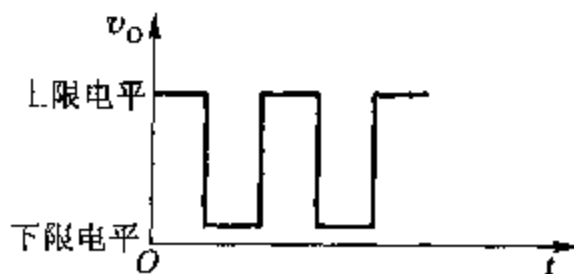


图 11-8 DAC 的矩形波输出

DELAY1、DELAY2 为 2 个延时程序, 分别决定输出的矩形波高、低电平的持续宽度。矩形波如图 11-8 所示。矩形波的频率也可采用调节延时时间长短的方法来改变。

(2) 双缓冲方式

对于多路的 D/A 转换, 要求同步输出

时,必须采用双缓冲同步方式。以此种方式工作时,数字量的输入锁存和 D/A 转换输出是分两步完成的。单片机必须通过 \overline{LE} 端来锁存待转换数字量,通过 $\overline{LE2}$ 端来启动 D/A 转换(见图 11-2)。因此,双缓冲方式下,DAC0832 应该为单片机提供两个 I/O 端口。8031 和 DAC0832 在双缓冲方式的连接如图 11-9 所示。由图可见,1[#] DAC0832 因 \overline{CS} 和译码器 FDH 相连而占有 FDH 和 FEH 两个 I/O 端口地址,而 2[#] DAC0832 的两个端口地址为 FEH 和 FFH。其中,FDH 和 FEH 分别为 1[#] 和 2[#] DAC0832 的数字量输入控制端口地址,而 FFH 为启动 D/A 转换的端口地址。其余连接如图 11-9 所示。

若用图 11-9 中 DAC 输出的模拟电压 V_X 和 V_Y 来控制 X-Y 绘图仪,则应把 V_X 和 V_Y 分别加到 X-Y 绘图仪的 X 通道和 Y 通道,而 X-Y 绘图仪由 X、Y 两个方向的步进电机驱动,其中一个电机控制绘笔沿 X 方向运动;另一个电机控制绘笔沿 Y 方向运动。因此对 X-Y 绘图仪的控制有两点基本要求,一是需要两个 D/A 转换器分别给 X 通道和 Y 通道提供模拟电压信号,使绘图笔能沿 X-Y 轴做平面运动;二是两路模拟信号要同步输出,使绘制的曲线光滑,否则绘制的曲线就是阶梯状的。通过执行下例的程序就可达到控制绘图仪的目的。

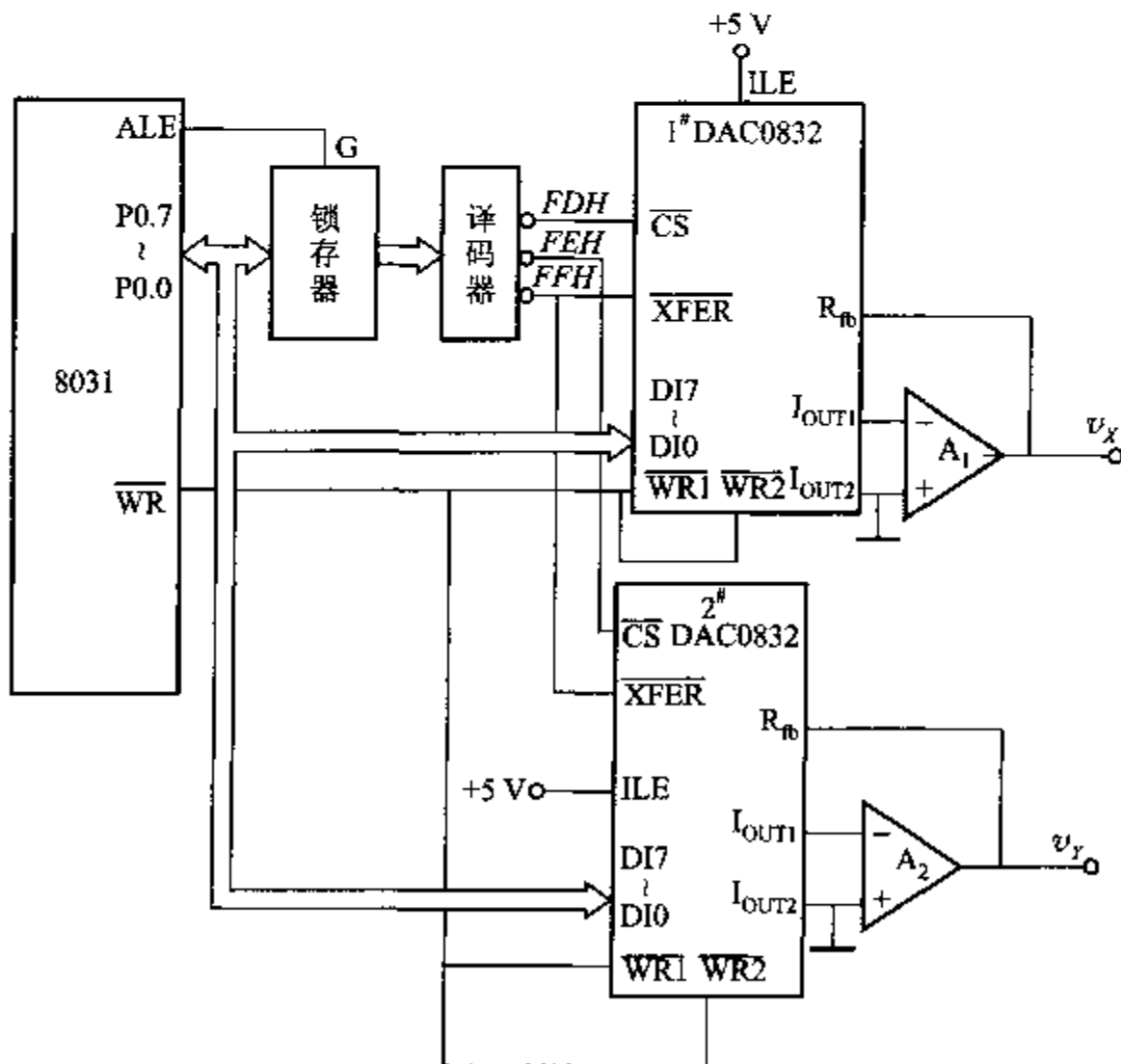


图 11-9 8031 和两片 DAC0832 的接口(双缓冲方式)

例 11-2 设 8031 内部 RAM 中有 2 个长度为 20 的数据块,其起始地址分别为 *addr1* 和 *addr2*,根据图 11-9,编写能把 *addr1* 和 *addr2* 中数据从 1# 和 2# DAC0832 同步输出的程序。程序中的 *addr1* 和 *addr2* 中的数据,即为绘图仪所绘制曲线的 X,Y 坐标点。

由图 11-9,DAC0832 各端口地址为:

FDH 1# DAC0832 数字量输入控制端口

FEH 2# DAC0832 数字量输入控制端口

FFH 1# 和 2# DAC0832 启动 D/A 转换端口

我们使工作寄存器 0 区的 R1 指向 *addr1*;1 区的 R1 指向 *addr2*;工作寄存器 0 区的 R2 存放数据块长度;工作寄存器 0 区和 1 区的 R0 指向 DAC 端口地址。相应程序为:

```

                ORG 2000H
addr1 DATA 20H      ; 定义存储单元
addr2 DATA 40H      ; 定义存储单元
DTOUT: MOV R1,#addr1 ; 0 区 R1 指向 addr1
        MOV R2,#20    ; 数据块长度送 0 区 R2
        SETB RS0      ; 切换到工作寄存器 1 区
        MOV R1,#addr2 ; 1 区 R1 指向 addr2
        CLR RS0       ; 返回工作寄存器 0 区
NEXT:  MOV R0,#0FDH    ; 0 区 R0 指向 1# DAC0832 数字量控制端口
        MOV A,@R1      ; addr1 中数据送 A
        MOVX @R0,A     ; addr1 中数据送 1# DAC0832
        INC R1         ; 修改 addr1 指针 0 区 R1
        SETB RS0      ; 转入 1 区
        MOV R0,#0FEH   ; 1 区 R0 指向 2# DAC0832 数字量控制端口
        MOV A,@R1      ; addr2 中数据送 A
        MOVX @R0,A     ; addr2 中数据送 2# DAC0832
        INC R1         ; 修改 addr2 指针 1 区 R1
        INC R0         ; 1 区 R0 指向 DAC 的启动 D/A 转换端口
        MOVX @R0,A     ; 启动 DAC 进行转换
        CLR RS0       ; 返回 0 区
        DJNZ R2,NEXT   ; 若未完,则跳 NEXT
        LJMP DTOUT     ; 若送完,则循环
END

```

11.1.3 MCS-51 与 12 位 DAC1208 的接口

当 8 位的 DAC 分辨率不够时,可以采用 12 位的 DAC。目前较为常用的 12

位 DAC 有 2 个系列,一种是 DAC1208 系列;另一种是 DAC1230 系列。

DAC1208 系列 D/A 转换器有 DAC1208、DAC1209、DAC1210 三种芯片类型。与 DAC1208 系列结构相类似的产品是 DAC1230 系列,这个系列也有 DAC1230、DAC1231、DAC1232 三种芯片,这一系列将在下一小节介绍。

1. DAC1208 系列的结构引脚及特性

图 11-10 是 12 位 D/A 转换器 DAC1208 系列的内部结构及引脚分布。其结构和 DAC0832 很相似,也是双缓冲的结构,只是把 8 位的部件换成了 12 位的。但对于输入锁存器来说,不是用 1 个 12 位锁存器,而是用 1 个 8 位锁存器和 1 个 4 位锁存器,以便和 8 位的数据总线相连接。

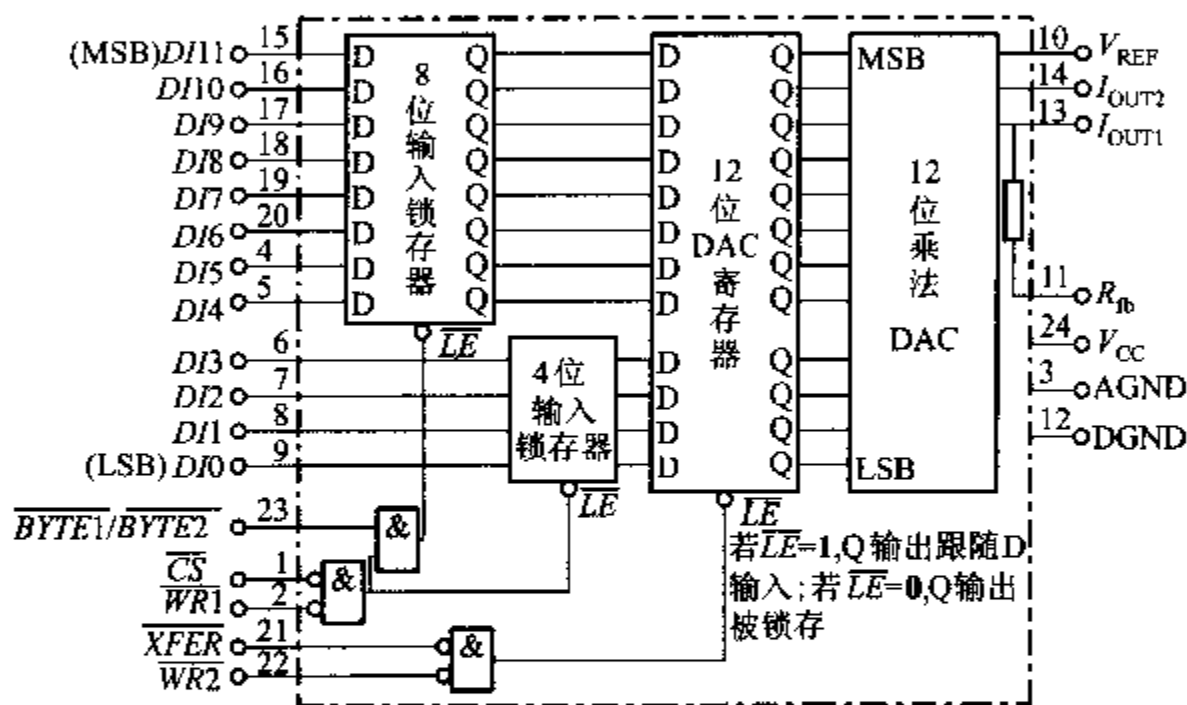


图 11-10 DAC1208 的内部结构与引脚分布

DAC1208 引脚功能如下所示:

引脚信号	功 能
\overline{CS}	片选信号,低电平有效。
$\overline{WR1}$	写信号,低电平有效。
$\overline{BYTE1}/\overline{BYTE2}$	字节顺序控制信号,该信号为高电平时,开启 8 位和 4 位 2 个锁存器,将 12 位全部打入锁存器。当该信号为低电平时,则开启 4 位输入锁存器。
$\overline{WR2}$	辅助写,低电平有效。该信号与 \overline{XFER} 相结合,当 \overline{XFER} 与 $\overline{WR2}$ 同时为低电平时,把锁存器中数据打入 DAC 寄存器。当 $\overline{WR2}$ 为高电平时,DAC 寄存器中的数据被锁存起来。
\overline{XFER}	传送控制信号,低电平有效。该信号与 $\overline{WR2}$ 相结合,用于将输入锁存器中的 12 位数据送至 DAC 寄存器。

续表

引脚信号	功 能
$DI0 \sim DI11$	12 位数据输入信号
I_{OUT1}	D/A 转换电流输出信号 1。当 DAC 寄存器全 1 时, 输出电流最大, 全 0 时输出为 0。
I_{OUT2}	D/A 转换电流输出信号 2。 $I_{OUT1} + I_{OUT2} = \text{常数}$
R_{fb}	反馈电阻输入信号
V_{REF}	参考电压输入信号
V_{CC}	电源电压信号
DGND、AGND	数字地和模拟地信号

DAC1208 的主要应用特性为:

- (1) 输出电流稳定时间: $1 \mu s$;
- (2) 基准电压: $V_{REF} = -10 \sim +10 V$;
- (3) 单工作电源: $+5 \sim +15 V$;
- (4) 低功耗: $20 mW$ 。

2. 接口电路设计及软件编程

(1) 接口电路设计

硬件接口设计最重要的就是 DAC1208 的输入控制线, DAC1208 的输入控制线基本上和 DAC0832 相同。 \overline{CS} 和 $\overline{WR1}$ 用来控制输入寄存器, \overline{XFER} 和 $\overline{WR2}$ 用来控制 DAC 寄存器。但是为了区分 8 位输入寄存器和 4 位输入寄存器, 增加了 1 条控制线 $BYTE1/\overline{BYTE2}$ 。当该线信号为 1 时, 选中 8 位输入寄存器, 为 0 时则选中 4 位输入寄存器。有了这条控制线, 2 个输入寄存器可以接同一条译码器输出端(接至 \overline{CS} 端)。实际上, 在 $BYTE1/\overline{BYTE2} = 1$ 时, 2 个输入寄存器都被选中, 而在 $BYTE1/\overline{BYTE2} = 0$ 时, 只选中 4 位输入寄存器。这样可以用 1 条地址线 A0 来控制 $BYTE1/\overline{BYTE2}$, 用 2 条译码器输出线控制 \overline{CS} 和 \overline{XFER} 。1 片 DAC1208 芯片只占用 3 个 I/O 端口地址。

8031 单片机和 DAC1208 转换器的硬件连接如图 11-11 所示。DAC1208 的高 8 位输入寄存器地址为 4001H, 低 4 位寄存器地址为 4000H, 而 DAC 寄存器的地址为 6000H。考虑到 8031 单片机 P0.0 地址/数据线分时复用, 所以用 P0.0 与 DAC1208 的 $BYTE1/\overline{BYTE2}$ 相连时要有锁存器 74LS377。因 DAC1208 系列内部没有基准源, 故外接 AD581 做 10 V 电压基准源。模拟电压输出接为双极性。

DAC1208 系列 D/A 转换器的工作采用双缓冲方式。在送入数据时要先送

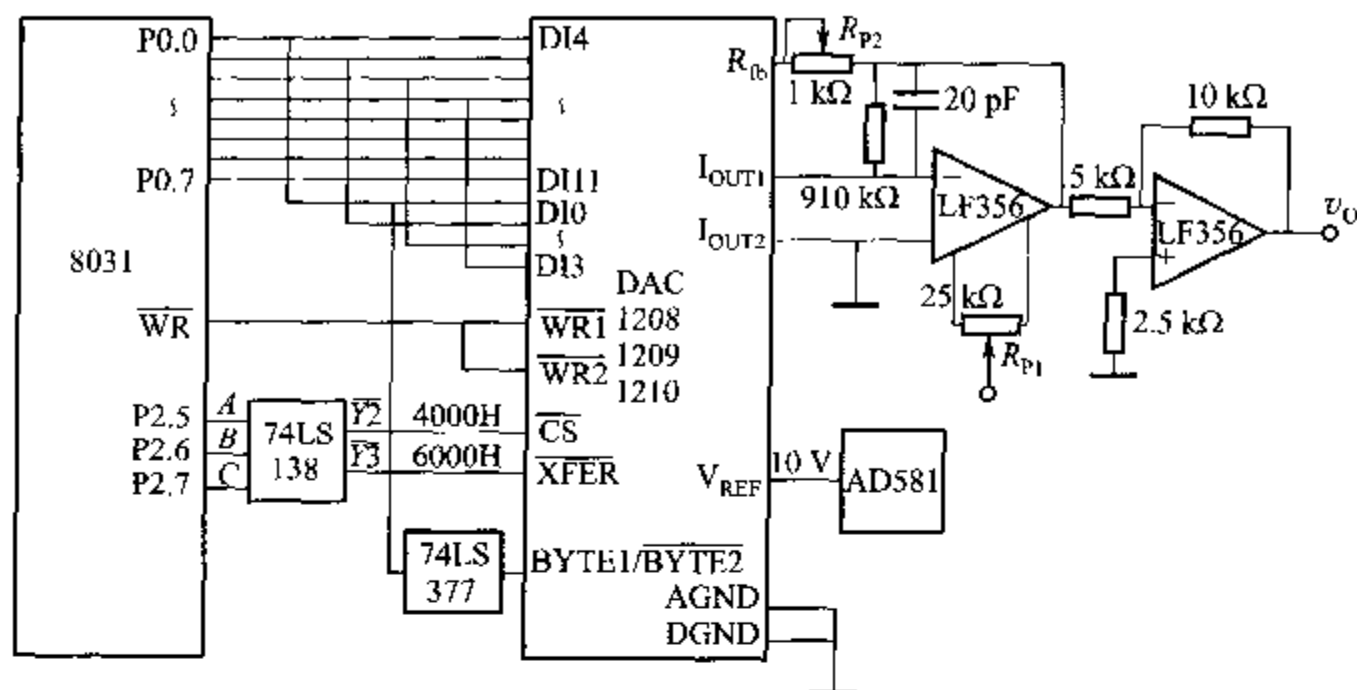


图 11-11 8031 单片机与 DAC1208 系列的接口

入 12 位数据中的高 8 位 DI11~DI4 然后再送入低 4 位 DI3~DI0,而不能按相反的顺序传送。这是因为在输入 8 位寄存器时,4 位输入寄存器也是打开的,如果先送低 4 位后送高 8 位,结果就会不正确。在 12 位数据分别正确地进入 2 个输入寄存器后,再打开 DAC 寄存器,就可以把 12 位数据送到 12 位 D/A 转换器去转换。单缓冲方式在这里是不合适的,在 12 位数据不是 1 次送入的情况下,边传送边转换会使输出产生错误的瞬间毛刺。

图中 DAC1208 的电流输出端外接 2 个运放 LF356,其中运放 1 用作电流/电压转换器,运放 2 实现双极性电压输出($-10\sim+10\text{V}$)。调电位器 R_{P1} 定零点,调电位器 R_{P2} 定满度。

(2) 软件编程

设 12 位数字量存放在内部 RAM 的 2 个单元, $DIGIT$ 和 $DIGIT+1$ 。12 位数的高 8 位在 $DIGIT$ 单元,低 4 位在 $DIGIT+1$ 单元的低 4 位。现在按图 11-11 的硬件接口电路将 12 位数据送到 DAC1208 去转换,接口电路的 D/A 转换程序如下:

MOV	DPTR, #4001H	; 8 位输入寄存器地址
MOV	R1, #DIGIT	; 高 8 位数据地址
MOV	A, @R1	; 取出高 8 位数据
MOVX	@DPTR, A	; 高 8 位数据送 DAC1208
DEC	DPL	; DPTR 修改为 4 位输入寄存器地址
INC	R1	; 低 4 位数据地址
MOV	A, @R1	; 取出低 4 位数据
MOVX	@DPTR, A	; 低 4 位数据送 DAC1208
MOV	DPTR, #6000H	; DAC 寄存器地址

MOVX @DPTR,A

; 12 位同步输出完成 12 位 D/A 转换

11.1.4 MCS-51 与 12 位 DAC1230 系列的接口

DAC1230 系列芯片有 DAC1230、DAC1231、DAC1232 3 种类型。DAC1230 的内部结构和应用特性与 DAC1208 完全相似,只不过 DAC1230 系列的低 4 位数据线在片内与高 4 位数据线相连,在片外表现为 8 位数据线,故 DAC1230 比 DAC1208 少 4 个引脚,是 20 引脚的 DIP 封装。

DAC1230 的内部结构及引脚分布如图 11-12 所示。

DAC1230 芯片的引脚功能和与 8031 单片机的接口电路请参阅前面 DAC1208 的介绍,这里不再重复。显然 DAC1230 系列 D/A 转换器与 8 位单片机的接口比 DAC1208 还要简单;但 DAC1208 系列与 16 位单片机连接更方便。建议 MCS-51 单片机进行 DAC 扩展时,选用 DAC1230 系列接口电路更为简单。

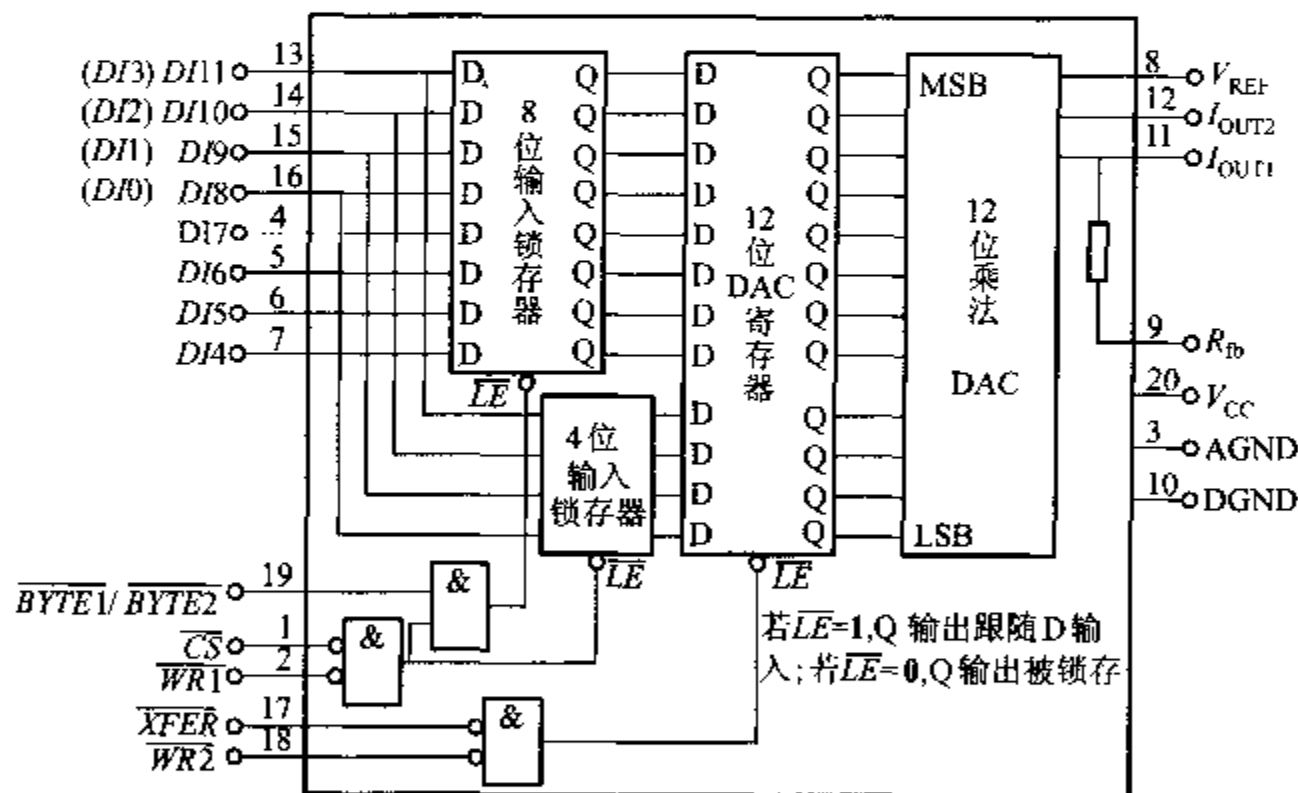


图 11-12 DAC1230 系列的内部结构与引脚分布

11.2 MCS-51 与 ADC 的接口

11.2.1 A/D 转换器概述

A/D 转换器(ADC)的作用就是把模拟量转换成数字量,以便于计算机进行

处理。

随着超大规模集成电路技术的飞速发展, A/D 转换器的新设计思想和制造技术层出不穷。为满足各种不同的检测及控制任务的需要, 大量结构不同、性能各异的 A/D 转换芯片应运而生。

1. A/D 转换器的分类

根据 A/D 转换器的原理可将 A/D 转换器分成两大类。一类是直接型 A/D 转换器, 另一类是间接型 A/D 转换器。在直接型 A/D 转换器中, 输入的模拟电压被直接转换成数字代码, 不经任何中间变量; 在间接型 A/D 转换器中, 首先把输入的模拟电压转换成某种中间变量(如时间、频率、脉冲宽度等等), 然后再把这个中间变量转换为数字代码输出。

A/D 转换器的分类如图 11-13 所示。

尽管 A/D 转换器的种类很多, 但目前应用较广泛的主要有以下几种类型: 逐次比较式转换器、双积分式转换器、 $\Sigma-\Delta$ 式 A/D 转换器和 V/F 转换器。

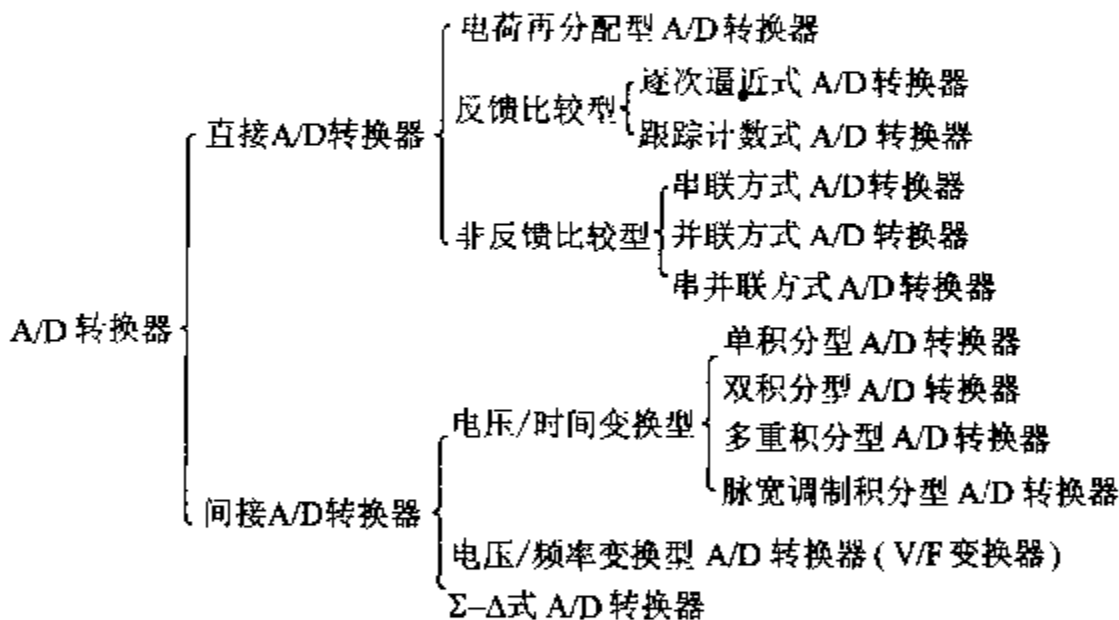


图 11-13 A/D 转换器的分类

逐次比较型 A/D 转换器, 在精度、速度和价格上都适中, 是最常用的 A/D 转换器件。双积分 A/D 转换器, 具有精度高、抗干扰性好、价格低廉等优点, 但转换速度慢, 近年来在单片机应用领域中也得到广泛应用。 $\Sigma-\Delta$ 式 ADC 具有积分式与逐次比较式 ADC 的双重优点。它对工业现场的串模干扰具有较强的抑制能力, 不亚于双积分 ADC, 它比双积分 ADC 有更高的转换速度, 与逐次比较式 ADC 相比, 有较高的信噪比, 分辨率高, 线性度好, 不需要采样保持电路。由于上述优点, $\Sigma-\Delta$ 式 ADC 得到了重视, 目前已有多种 $\Sigma-\Delta$ 式 A/D 芯片投向市场。而 V/F 转换器适用于转换速度要求不太高, 须进行远距离信号传输的 A/D 转换过程。

2. A/D 转换器的主要技术指标

(1) 转换时间和转换速率

转换时间: A/D 完成一次转换所需要的时间。转换时间的倒数为转换速率。

并行式 A/D 转换器, 转换时间最短约为 $20 \sim 50 \text{ ns}$, 速率为 $5 \times 10^7 \sim 2 \times 10^7$ 次/s; 双极性逐次比较式转换时间约为 $0.4 \mu\text{s}$, 速率为 2.5×10^6 次/s。

(2) 分辨率

A/D 转换器的分辨率习惯上用输出二进制位数或 BCD 码位数表示。例如 AD574 A/D 转换器, 可输出二进制 12 位, 即用 2^{12} 个数进行量化, 其分辨率为 1 LSB, 用百分数表示为 $\frac{1}{2^{12}} \times 100\% = 0.0244\%$ 。又如双积分式输出 BCD 码的 A/D 转换器 MC14433, 其分辨率为 $3\frac{1}{2}$ 位, 即三位半。若满字位为 1999, 用百分数表示其分辨率为 $1/1999 \times 100\% = 0.05\%$ 。

量化过程引起的误差为量化误差。量化误差是由于有限位数字量对模拟量进行量化而引起的误差。量化误差理论上规定为 1 个单位分辨率的 $\pm \frac{1}{2}$ LSB, 提高分辨率可减少量化误差。

(3) 转换精度

A/D 转换器的转换精度定义为一个实际 A/D 转换器与一个理想 A/D 转换器在量化值上的差值。可用绝对误差或相对误差表示。

3. A/D 转换器的选择

A/D 转换器按照输出代码的有效位数分为 4 位、8 位、10 位、12 位、14 位、16 位和 BCD 码输出的 $3\frac{1}{2}$ 位、 $4\frac{1}{2}$ 位、 $5\frac{1}{2}$ 位等多种; 按照转换速度可分为超高速(转换时间 $\leq 1 \text{ ns}$)、高速(转换时间 $\leq 1 \mu\text{s}$)、中速(转换时间 $\leq 1 \text{ ms}$)、低速(转换时间 $\leq 1 \text{ s}$) 等几种不同转换速度的芯片。为适应系统集成的需要, 有些转换器还将多路转换开关、时钟电路、基准电压源、二/十进制译码器和转换电路集成在 1 个芯片内, 为用户提供了很多方便。在设计数据采集系统、测控系统和智能仪器仪表时, 要问题就是如何选择合适的 A/D 转换器以满足应用系统设计的要求。下面从不同角度介绍选择 A/D 转换器的要点。

(1) A/D 转换器位数的确定

A/D 转换器位数的确定与整个测量控制系统所要测量控制的范围和精度有关, 但又不能惟一确定系统的精度。因为系统精度涉及的环节较多, 包括传感器变换精度、信号预处理电路精度和 A/D 转换器及输出电路、控制机构精度, 甚至还包括软件控制算法。然而估算时, A/D 转换器的位数至少要比总精度要求的最低分辨率高一位(虽然分辨率与转换精度是不同的概念, 但没有基本的分辨率就谈不上转换精度, 精度是在分辨率的基础上反映的)。实际选取的 A/D 转换器的位

数应与其他环节所能达到的精度相适应。只要不低于它们就行,选得太高既没有意义,而且价格还要高得多。

一般把8位以下的A/D转换器归为低分辨率A/D转换器,9~12位的称为中分辨率,13位以上的为高分辨率。

(2) A/D转换器转换速率的确定

A/D转换器从启动转换到转换结束,输出稳定的数字量,需要一定的时间,这就是A/D转换器的转换时间;转换时间的倒数就是每秒钟能完成的转换次数,称为转换速率。用不同原理实现的A/D转换器其转换时间是大不相同的。总的来说,积分型、电荷平衡型和跟踪比较型A/D转换器转换速度慢,转换时间从几毫秒到几十毫秒不等,只能构成低速A/D转换器。一般适用于对温度、压力、流量等缓变参量的检测和控制。逐次比较型的A/D转换器的转换时间可从1~100 μs ,属于中速A/D转换器,常用于工业多通道单片机控制系统和声频数字转换系统等。转换时间最短的高速A/D转换器是那些用双极型或CMOS工艺制成的全并行型、串并行型和电压转移函数型的A/D转换器。转换时间仅20~100 ns。高速A/D转换器适用于雷达、数字通信、实时光谱分析、实时瞬态记录、视频数字转换系统等。

如用转换时间为100 μs 的集成A/D转换器,其转换速率为 1×10^4 次/s。根据采样定理和实际需要,1个周期的波形需采10个点,那么这样的A/D转换器最高也只能处理1 kHz的信号。把转换时间减小到10 μs ,信号频率可提高到100 kHz。对一般微处理机而言,要在10 μs 内完成A/D转换器转换以外的工作,如读数据、再启动、存数据、循环计数等已经比较困难。要继续提高采集数据的速度就不能用CPU来控制,必须采用直接存储器访问(DMA)技术来实现。

(3) 是否要加采样保持器

原则上直流和变化非常缓慢的信号可不用采样保持器。其他情况都要加采样保持器。根据分辨率、转换时间、信号带宽,可得到如下数据作为是否要加采样保持器的参考:如果A/D转换器的转换时间是100 ms、ADC是8位、没有采样保持器时,信号的允许频率是0.12 Hz;如果ADC是12位,该频率为0.0077 Hz。如果转换时间是100 μs ,ADC是8位时,该频率为12 Hz,12位时是0.77 Hz。

(4) 工作电压和基准电压

有些A/D转换器需要 $\pm 15\text{ V}$ 的工作电压,也有一些可在+12~+15 V范围内工作,这就需要多种电源。如果选择使用单一+5 V工作电压的芯片,与单片机系统可共用1个电源就比较方便。

基准电压源是提供给A/D转换器在转换时所需要的参考电压,这是保证转换精度的基本条件。在要求较高精度时,基准电压要单独用高精度稳压电源供给。

11.2.2 MCS-51 与 ADC0809(逐次比较型)的接口

1. ADC0809 引脚及功能

ADC0809 是一种逐次比较式 8 路模拟输入、8 位数字量输出的 A/D 转换器。其引脚如图 11-14 所示。

由引脚图可见,ADC0809 共有 28 脚,采用双列直插式封装。其主要引脚功能如下:

(1) $IN_0 \sim IN_7$ 是 8 路模拟信号输入端。

(2) $D_0 \sim D_7$ 是 8 位数字量输出端。

(3) A、B、C 与 ALE 控制 8 路模拟通道的切换,A、B、C 分别与 3 根地址线或数据线相连,3 位编码对应 8 个通道地址端口。 $C、B、A = 000 \sim 111$ 分别对应 $IN_0 \sim IN_7$ 通道的地址。

这里要强调的是:ADC0809 虽然有 8 路模拟通道可以同时输入 8 路模拟信号,但每个瞬间只能转换 1 路,各路之间的切换由软件改变 C、B、A 引脚上的代码来实现。

(4) OE、START、CLK 为控制信号端,OE 为输出允许端,START 为启动信号输入端,CLK 为时钟信号输入端。

(5) $V_R(+)$ 和 $V_R(-)$ 为参考电压输入端。

2. ADC0809 结构及转换原理

ADC0809 的结构框图如图 11-15 所示。ADC0809 是采用逐次比较的方法完成 A/D 转换的,由单一的 +5 V 电源供电。片内带有锁存功能的 8 路选 1 的模拟开关,由 C、B、A 引脚的编码来决定所选的通道。0809 完成 1 次转换需 100 μs 左右,输出具有 TTL 三态锁存缓冲器,可直接连到 MCS-51 的数据总线上。通过适当的外接电路,0809 可对 0~5 V 的模拟信号进行转换。

3. MCS-51 与 ADC0809 的接口

在讨论 MCS-51 与 0809 的接口设计之前,先来讨论单片机如何控制 ADC 的问题。

单片机控制 ADC0809 的工作过程如下:

首先用指令选择 0809 的一个模拟输入通道,当执行 `MOVX @DPTR, A` 时,单片机的 \overline{WR} 信号有效,从而产生一个启动信号,给 0809 的 START 引脚送入脉冲,开始对选中通道进行转换。当转换结束后,0809 发出转换结束 EOC(高电平)信号,该信号可供单片机查询,也可反相后作为向单片机发出的中断请求信

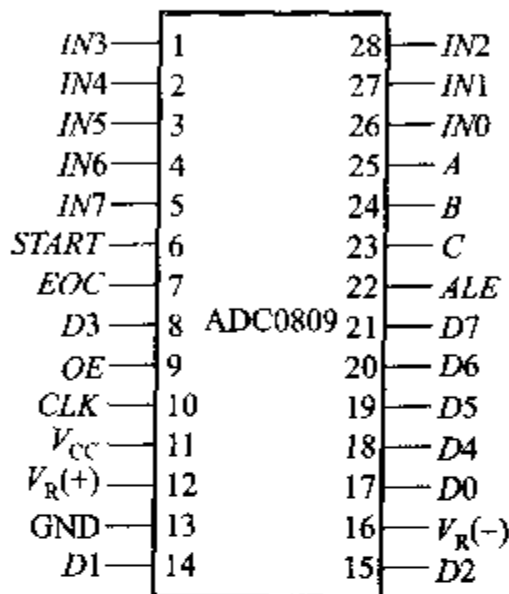


图 11-14 ADC0809 的引脚

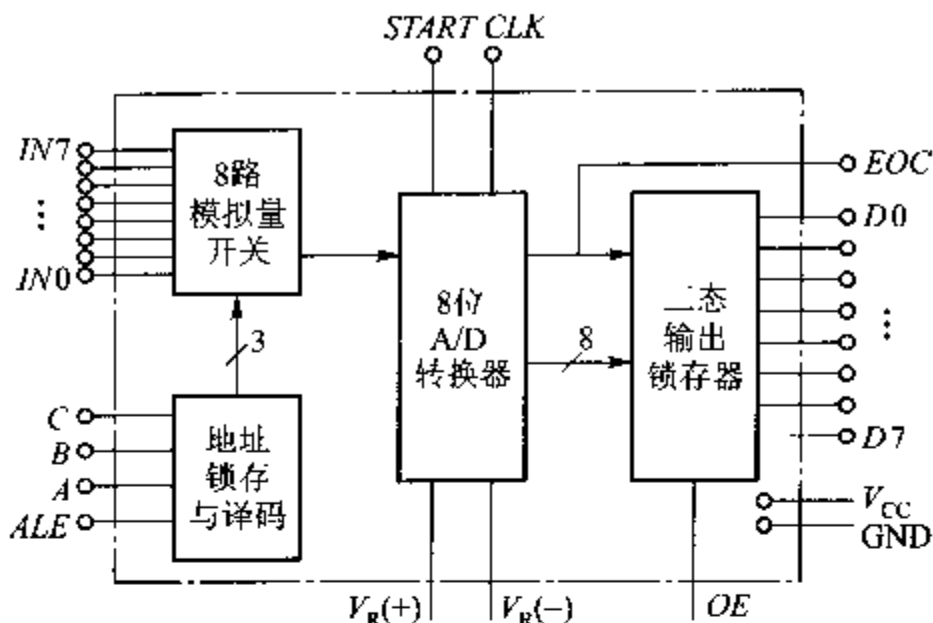


图 11-15 ADC0809 结构图

号;当执行指令:MOVX A, @DPTR,单片机发出读控制 \overline{RD} 信号,OE 端有高电平,且把经过 0809 转换完毕的数字量读到 A 累加器中。

由上述可见,用单片机控制 ADC 时,可采用查询和中断控制两种方式。查询方式是在单片机把启动信号送到 ADC 之后,执行别的程序,同时对 0809 的 EOC 引脚的状态进行查询,以检查 ADC 变换是否已经结束,如查询到变换已经结束,则读入转换完毕的数据。

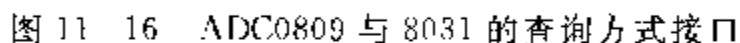
中断控制方式是在启动信号送到 ADC 之后,单片机执行别的程序。0809 转换结束并向单片机发出中断请求信号时,单片机响应此中断请求,进入中断服务程序,读入转换数据。中断控制方式效率高,所以特别适合于变换时间较长的 ADC。

如果对转换速度要求高,采用上述两种 ADC 控制方式往往不能满足要求,可采用 DMA(直接存储器存取)的方法,这时可在 ADC 与单片机之间插入一个 DMA 接口(例如 Intel 公司的 8237 DMA 控制器)。传输一开始,A/D 转换的数据就可以从输出寄存器经过 DMA 中的数据寄存器直接传输到主存储器,因而不受程序的限制。

(1) 查询方式

ADC0809 与 8031 单片机的接口如图 11-16 所示。

由于 ADC0809 片内无时钟,可利用 8031 提供的地址锁存允许信号 ALE 经 D 触发器 2 分频后获得,ALE 引脚的频率是 8031 单片机时钟频率的 1/6(但要注意的,每当访问外部数据存储器时,将少 1 个 ALE 脉冲)。如果单片机时钟频率采用 6 MHz,则 ALE 引脚的输出频率为 1 MHz,再 2 分频后为 500kHz,恰好符合 ADC0809 对时钟频率的要求。由于 ADC0809 具有输出三态锁存器,其 8 位数据输出引脚可直接与数据总线相连。地址译码引脚 C、B、A 分别与地



下面的程序是采用软件延时的方式,分别对 8 路模拟信号轮流采样 1 次,并依次把结果转存到数据存储区的转换程序。

MAIN:	MOV R1, #data	;置数据区首地址
	MOV DPTR, #7FF8H	;端口地址送 DPTR, P2.7=0, 且指向通道 IN0
	MOV R7, #08H	;置通道个数
LOOP:	MOVX @DPTR, A	;启动 A/D 转换
	MOV R6, #0AH	;软件延时, 等待转换结束
DELAY:	NOP	
	NOP	
	NOP	
	DJNZ R6, DELAY	
	MOVX A, @DPTR	;读取转换结果
	MOV @R1, A	;存储转换结果
	INC DPTR	;指向下一个通道
	INC R1	;修改数据区指针
	DJNZ R7, LOOP	;8 个通道全采样完否? 未完则继续

• • •

(2) 中断方式

ADC0809 与 8031 的中断方式接口电路只需要将图 11-16 中的 EOC 引脚经过一非门连接到 8031 的 $\overline{\text{INT1}}$ 引脚即可。采用中断方式可大大节省 CPU 的时间,当转换结束时,EOC 发出一个脉冲向单片机提出中断申请,单片机响应中断请求,由外部中断 1 的中断服务程序读 A/D 结果,并启动 ADC0809 的下一次转换,外部中断 1 采用跳沿触发方式。

程序如下:

```
INT1:  SETB  IT1           ;外部中断 1 初始化编程
        SETB  EA           ;CPU 开中断
        SETB  EX1          ;选择外中断为跳沿触发方式
        MOV   DPTR, #7FF8H ;端口地址送 DPTR
        MOV   A, #00H
        MOVX  @DPTR, A      ;启动 0809 对 IN0 通道转换
        ...                 ;完成其他的工作
```

中断服务程序:

```
PINT1:  MOV   DPTR, #7FF8H ;读取 A/D 结果送内部 RAM 单元 30H
        MOVX  A, @DPTR
        MOV   30H, A
        MOV   A, #00H      ;启动 0809 对 IN0 的转换
        MOVX  @DPTR, A
        RETI
```

11.2.3 MCS-51 与 AD574(逐次比较型)的接口

在某些单片机应用系统中,8 位 ADC 常常不够,必须选择分辨率大于 8 位的芯片,如 10 位、12 位、16 位 A/D 转换器,由于 10 位、16 位接口与 12 位类似,因此仅以常用的 12 位 A/D 转换器 AD574 为例介绍。

1. AD574 简介

AD574 是 12 位逐次比较型 A/D 转换器。转换时间为 $25\mu\text{s}$,转换精度为 0.05%,由于芯片内有三态输出缓冲电路,因而可直接与各种典型的 8 位或 16 位的微处理器相连,而无须附加逻辑接口电路,且能与 CMOS 及 TTL 兼容。

AD574 为 28 引脚双列直插式封装,其引脚如图 11-17 所示。

AD574 共有 6 个控制引脚,功能如下:

$\overline{\text{CS}}$:片选信号端。

CE:片启动信号引脚。

$\text{R}/\overline{\text{C}}$:读出/转换控制信号引脚。

$12/\overline{8}$:数据输出格式选择信号引脚。

当 $12/\bar{8}=1$, 12 条数据线同时输出转换结果;

当 $12/\bar{8}=0$, 转换结果为单字节输出, 即只有高 8 位或低 4 位有效。

A0: 字节选择控制线。

在转换期间:

当 $A0=0$, AD574 进行全 12 位转换, 转换时间为 $25\mu\text{s}$;

当 $A0=1$, 进行 8 位转换, 转换时间为 $16\mu\text{s}$ 。

在读出期间:

当 $A0=0$, 高 8 位数据有效;

当 $A0=1$, 低 4 位数据有效, 中间 4 位为 0, 高 4 位为三态。

因此当采用 2 次读出 12 位数据时, 12 位数据遵循左对齐原则, 如下所示:

结果的高 8 位	结果的低 4 位 + 4 位尾 0
----------	-------------------

AD574 的上述 5 个控制信号组合的真值表如表 11-1 所示。

表 11-1 AD574 控制信号真值表

CE	$\overline{\text{CS}}$	R/ $\overline{\text{C}}$	12/ $\bar{8}$	A0	操 作
0	×	×	×	×	无操作
×	1	×	×	×	无操作
1	0	0	×	0	初始化为 12 位转换器
1	0	0	×	1	初始化为 8 位转换器
1	0	1	+5 V	×	允许 12 位并行输出
1	0	1	接地	0	允许高 8 位输出
1	0	1	接地	1	允许低 4 位 + 4 位尾 0 输出

STS: 输出状态信号引脚。转换开始时, STS 为高电平, 转换过程中保持高电平。转换完成时为低电平。STS 引脚信号可以作为状态信息被 CPU 查询, 也可以用它的下跳沿向 CPU 发出中断申请, 通知 CPU A/D 转换已完成, 可以读取转换结果。

2. AD574 的工作特性

AD574 的工作状态由 CE、 $\overline{\text{CS}}$ 、R/ $\overline{\text{C}}$ 、12/ $\bar{8}$ 、A0 五个控制信号决定, 见表 11-1。

由表 11-1 可见, 当 $\text{CE}=1$, $\overline{\text{CS}}=0$ 同时满足时, AD574 才能处于工作状态。当 AD574 处于工作状态时, R/ $\overline{\text{C}}=0$ 时启动 A/D 转换; R/ $\overline{\text{C}}=1$ 时进行数据读

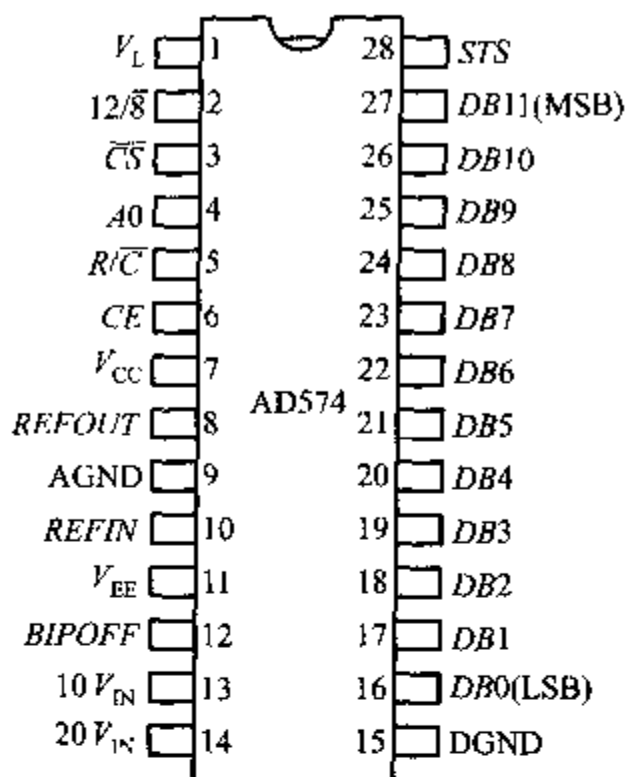


图 11-17 AD574 引脚

出。12/8 和 A0 端用来控制转换字长和数据格式。A0=0 时启动转换,则按完整的 12 位 A/D 转换方式工作,如果 A0=1 时启动转换,则按 8 位 A/D 转换方式工作。当 AD574 处于数据读出工作状态($R/\overline{C}=1$)时,A0 和 12/8 成为数据输出格式控制端。12/8=1,对应 12 位并行输出;12/8=0 则对应 8 位双字节输出。其中 A0=0 时输出高 8 位。A0=1 时输出低 4 位,并以 4 个 0 补足尾随的 4 位。必须指出 12/8 端与 TTL 电平不兼容,故只能直接接至 +5 V 或 0 V 上。另外 A0 在数据输出期间不能变化。

如果要求 AD574 以独立方式工作,只需将 CE、12/8 端接入 +5 V, \overline{CS} 和 A0 接至 0 V,将 R/\overline{C} 作为数据读出和数据转换启动的控制。当 $R/\overline{C}=1$ 时,数据输出端出现被转换后的数据, $R/\overline{C}=0$ 时,即启动 1 次 A/D 转换。在延时 $0.5 \mu s$ 后 STS=1 表示转换正在进行。经过 1 个转换周期(典型值为 $25 \mu s$)后 STS 端跳回低电平表示 A/D 转换完毕,可以从数据输出端读取新的数据。

注意,只有在 $CE=1$ 和 $\overline{CS}=0$ 时才启动转换,在启动信号有效前, R/\overline{C} 端必须为低电平,否则将产生读取数据的操作。

3. AD574 的单极性和双极性输入特性

通过改变 AD574 引脚 8、10、12 的外接电路,可使 AD574 进行单极性和双极性模拟信号的转换,图 11-18(a)所示为单极性转换电路,可实现输入信号 0~10 V 或 0~20 V 的转换。其系统模拟信号的地线应与 9 引脚相连,使其地线的接触电阻尽可能小。图 11-18(b)为双极性转换电路,可实现输入信号 -5~+5 V 或 -10~+10 V 的转换。

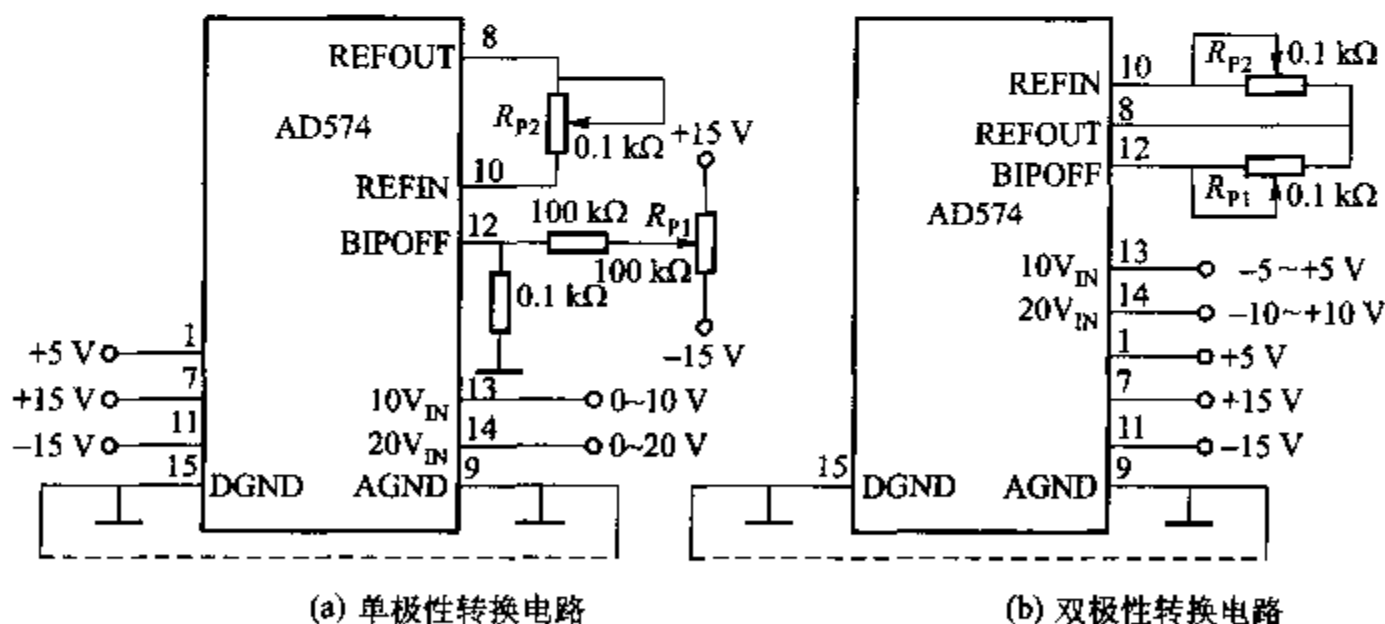


图 11-18 AD574 模拟输入电路的外部接法

4. MCS-51 与 AD574 的接口

图 11-19 是 AD574 与 8031 单片机的接口电路。由于 AD574 片内有时钟,故无须外加时钟信号。该电路采用单极性输入方式,可对 0~10 V 或 0~20 V

模拟信号进行转换。转换结果的高 8 位从 DB11~DB4 输出,低 4 位从 DB3~DB0 输出,并直接和单片机的数据总线相连。如果遵循左对齐原则,D3~D0 应接单片机的数据总线的高半字节。为了实现启动 A/D 转换和转换结果的读出,AD574 的片选信号 \overline{CS} 由地址总线的次低位 A1 提供,在读写时,A1 设置为低电平;AD574 的 \overline{CE} 信号由单片机的 \overline{WR} 和 A7 经 1 级或非门提供,R/ \overline{C} 信号则由 \overline{RD} 和 A7 经 1 级或非门产生,可见在读写时,A7 亦应为低电平。输出状态信号 STS 接 P3.2 端供单片机查询,以判断 A/D 转换是否结束。12/ $\overline{8}$ 端接地,AD574 的 A0 由地址总线最低位 A0 控制,以实现 A/D 的全 12 位转换,并将 12 位数据分 2 次送入数据总线上。

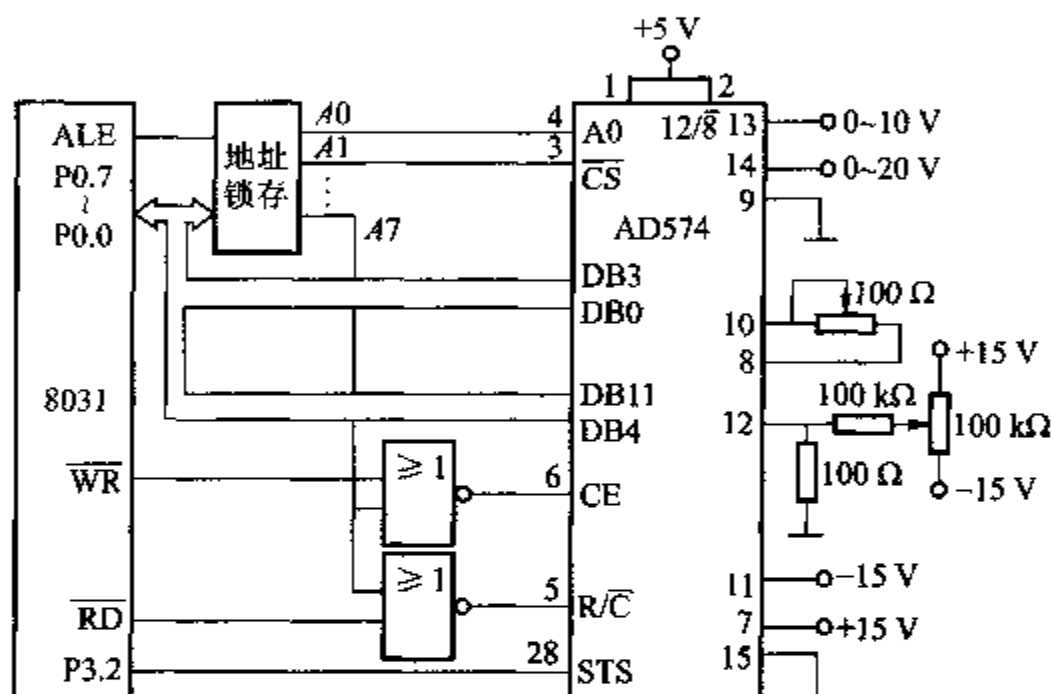


图 11-19 AD574 与 8031 的接口

利用该接口电路完成 1 次 A/D 转换的工作程序如下(假定转换结果高 8 位在 R2 中,低 4 位在 R3 中,按左对齐原则):

```

MAIN:  MOV  R0, #7CH      ; 选择 AD574, 并令 A0=0
        MOVX @R0, A       ; 启动 A/D 转换
LOOP:  NOP
        JB   P3.2, LOOP   ; 查询转换是否结束
        MOVX A, @R0       ; 读取高 8 位
        MOV  R2, A        ; 存入 R2 中
        MOV  R0, #7DH     ; 令 A0=1
        MOVX A, @R0       ; 读取低 4 位
        MOV  R3, A        ; 存入 R3 中
    
```

对于图 11-19 所示的接口电路,在设计印制电路板时,要注意电源去耦、布线以及地线的布置。这些问题对于位数较多的 ADC 与单片机接口时,要给予重视。

AD574 接口电路全部连接完毕后,在模拟输入端输入一稳定的标准电压,启动 A/D 转换,12 位数据亦应稳定。如果变化较大,说明电路稳定性差,则要从电源及接地布线等方面查找原因。AD574 的电源电压要有较好的稳定性和较小的噪声,噪声大的电源会产生不稳定的输出代码。在设计时,AD574 电源要很好地进行滤波调整,还要避开高频噪声源,这对 AD574 来讲是非常重要的。为了取得 12 位精度,除非进行很好的滤波,否则最好不要用开关电源。因为 mV 级的噪声能在 12 位 ADC 中引起好几位的误差。

所有的电源引脚都要用去耦电容。对 +5 V 电源,去耦电容直接接在引脚 1 和 15 之间;并且引脚 V_{CC} 和 V_{EE} 要通过电容耦合到引脚 9,合适的去耦电容是 1 个 $4.7 \mu\text{F}$ 的钽电容再并联 1 个 $0.1 \mu\text{F}$ 的陶瓷电容。

布线时应注意,把 AD574 连同模拟输入电路,尽可能远地离开数字电路部分。

引脚 9 模拟地线是内部参考电压的地线参考点。因此,它应该是一个高质量的地线,应直接接在系统的模拟地参考点上。为了在较大的数字信号干扰的情况下仍能最大限度地利用 AD574 取得高精度的输出,数字地线和模拟地线应接在一起。在一般情况下,在印制电路板布线时可考虑以下三点:

- (1) 数字地与模拟地要在芯片上就近连接在一起。
- (2) $\pm 15 \text{ V}$ 电源经过电容去耦以后,其地线连接到数字地上。
- (3) 外部模拟电路的接地端要分别连接到 AD574 的模拟地。

11.2.4 MCS-51 与 A/D 转换器 MC14433(双积分型)的接口

双积分型的 ADC 由于两次积分时间比较长,所以 A/D 转换速度慢,但精度可以做得比较高;对周期变化的干扰信号积分为零,抗干扰性能也较好。

目前,国外双积分 A/D 转换器集成电路芯片很多,大部分是应用于数字测量仪器上。常用的有 $3\frac{1}{2}$ 位双积分 A/D 转换器 MC14433(精度相当于 11 位二进制数)和 $4\frac{1}{2}$ 位双积分 A/D 转换器 ICL7135(精度相当于 14 位二进制数)。

1. MC14433 A/D 转换器简介

MC14433 是 $3\frac{1}{2}$ 位双积分型的 A/D 转换器,具有精度高、抗干扰性能好等优点,其缺点为转换速度慢,约 $1\sim 10$ 次/s。在不要求高速转换的数据采集系统中,被广泛应用。MC14433 A/D 转换器与国内产品 5G14433 完全相同,可以互换。

MC14433 A/D 转换器的被转换电压量程为 199.9 mV 或 1.999 V 。转换完的数据以 BCD 码的形式分 4 次送出(最高位输出内容特殊,详见表 11-2)。

MC14433 A/D 转换器引脚如图 11-20 所示。

下面分类介绍各引脚的功能。

① 电源及其地端

V_{DD} : 主工作电源 +5 V。

V_{EE} : 模拟部分的负电源端, 接 -5 V。

V_{AG} : 模拟地端。

V_{SS} : 数字地端。

V_R : 基准电压输入端。

② 外接电阻及电容端

R_1 : 积分电阻输入端, 转换电压 $V_A = 2\text{ V}$ 时, $R_1 = 470\ \Omega$; $V_A = 200\text{ mV}$ 时, $R_1 = 27\text{ k}\Omega$ 。

C_1 : 积分电容输入端, C_1 一般取 $0.1\ \mu\text{F}$ 。

R_1/C_1 : R_1 与 C_1 的公共端。

$CLKI$ 、 $CLKO$: 外接振荡器时钟调节电阻 R_C 。

R_C 一般取 $470\ \Omega$ 左右。

③ 转换启动/结束信号端

EOC : 转换结束信号输出端, 正脉冲有效。

DU : 启动新的转换, 若 DU 与 EOC 相连, 每当 A/D 转换结束后, 自动启动新的转换。

④ 过量程信号输出端

\overline{OR} : 当 $|V_X| < V_R$, 过量程 \overline{OR} 输出低电平。

⑤ 位选通控制端

$DS4 \sim DS1$: 分别为个、十、百、千位输出的选通脉冲, 正脉冲有效。DS1 对应千位, $DS4$ 对应个位。每个选通脉冲宽度为 18 个时钟周期, 2 个相应脉冲之间间隔为 2 个时钟周期。如图 11-21 所示。

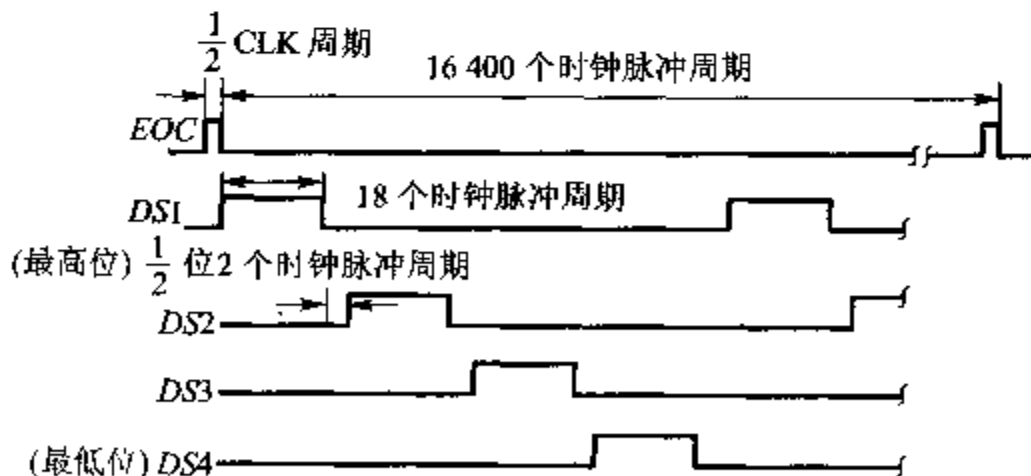


图 11-21 MC14433 选通脉冲时序图

⑥ BCD 码输出端

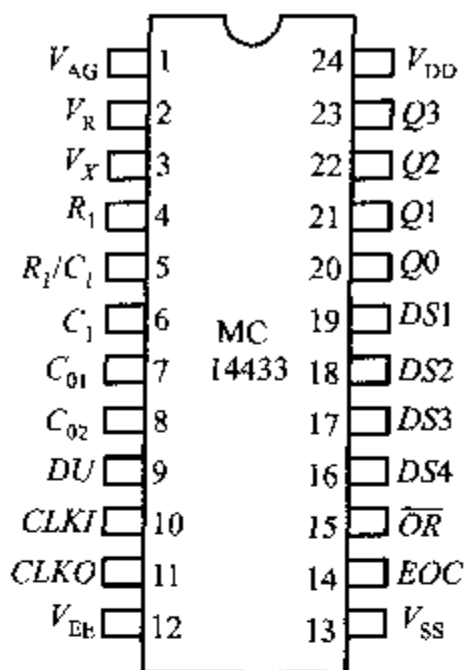


图 11-20 MC14433 引脚图

Q0~Q3:BCD 码数据输出线。其中 Q0 为最低位,Q3 为最高位。当 DS2、DS3 和 DS4 选通期间,输出 3 位完整的 BCD 码数,但在 DS1(千位)选通期间,输出端 Q0~Q3 除了表示千位的 0 或 1 外,还表示被转换电压的正负极性(Q2=1 为正)以及是欠量程还是过量程,其具体含义如表 11-2 所示。

表 11-2 DS1 选通时 Q3~Q0 表示的结果

Q3	Q2	Q1	Q0	表示结果
1	×	×	0	千位数为 0
0	×	×	0	千位数为 1
×	1	×	0	结果为正
×	0	×	0	结果为负
0	×	×	1	输入过量程
1	×	×	1	输入欠量程

由表 11-2 可知:

- Q3 表示最高位千位(1/2 位),Q3=0 对应 1,反之对应 0。
- Q2 表示极性,Q2=1 为正极性,Q2=0 为负极性。
- Q0=1 表示过量程或欠量程:当 Q3=0 时,表示过量程;当 Q3=1 时,表示欠量程。

2. MC14433 与 8031 单片机的接口

MC14433 的 A/D 转换结果是动态分时输出的 BCD 码,Q3~Q0 为千、百、十、个位的 BCD 码,而 DS1~DS4 引脚输出分别为千、百、十、个位的选通信号,由于转换结果输出不是总线式的,因此 MCS-51 单片机只能通过并行 I/O 接口或扩展 I/O 接口与其相连。下面介绍 MC14433 与 8031 单片机 P1 口直接连接的接口电路,电路如图 11-22 所示。

图中 MC1403(与 5G1403 相同)为+2.5 V 精密集成电压基准源,经电位器分压后作为 A/D 转换用基准电压。DU 端与 EOC 端相连,即选择连续转换方式,每次转换结果都送至输出寄存器。EOC 是 A/D 转换结束的输出标志信号。8031 读取 A/D 转换结果可以采用中断方式或查询方式。采用中断方式时,EOC 端与 8031 外部中断输入端 $\overline{\text{INT0}}$ 或 $\overline{\text{INT1}}$ 相连。采用查询方式时 EOC 端可与 8031 的任一 I/O 口线相连。

若选用中断方式读取 MC14433 的结果,应选用跳沿触发方式。如果将 A/D 转换的结果存放到 8031 内部 RAM 的 20H、21H 单元中,则存放的格式如图 11-23 所示。

下面介绍读取 A/D 转换结果的程序编写。

初始化程序开放 CPU 中断,允许外部中断 1 中断请求,置外部中断 1 为跳沿触发方式。每次 A/D 转换结束,都向 CPU 请求中断,CPU 响应中断,执行中

The diagram shows the following connections and components:

- 8031 Microcontroller:**
 - \overline{EA} is grounded.
 - $\overline{INT1}$ is connected to pin 9 of the MC14433.
 - Port 1 (P1.0-P1.7) is connected to pins 20-23, 19, 18, 17, 16, 15, 14, and 13 of the MC14433.
- MC14433 A/D Converter:**
 - Pin 6: $0.1\ \mu\text{F}$ capacitor to ground.
 - Pin 5: $470\ \text{k}\Omega$ resistor to pin 4.
 - Pin 4: $+5\ \text{V}$ supply.
 - Pin 12: $0.047\ \mu\text{F}$ capacitor to $+5\ \text{V}$ and $0.02\ \mu\text{F}$ capacitor to $-5\ \text{V}$.
 - Pin 15: Connected to pin 9 of the 8031.
 - Pin 16: Connected to pin 16 of the 8031.
 - Pin 17: Connected to pin 17 of the 8031.
 - Pin 18: Connected to pin 18 of the 8031.
 - Pin 19: Connected to pin 19 of the 8031.
 - Pin 20: Connected to pin 20 of the 8031.
 - Pin 21: Connected to pin 21 of the 8031.
 - Pin 22: Connected to pin 22 of the 8031.
 - Pin 23: Connected to pin 23 of the 8031.
 - Pin 24: Connected to pin 24 of the 8031.
 - Pin 25: $-5\ \text{V}$ supply.
 - Pin 26: $0.1\ \mu\text{F}$ capacitor to ground.
 - Pin 27: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 28: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 29: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 30: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 31: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 32: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 33: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 34: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 35: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 36: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 37: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 38: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 39: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 40: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 41: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 42: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 43: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 44: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 45: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 46: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 47: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 48: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 49: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 50: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 51: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 52: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 53: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 54: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 55: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 56: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 57: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 58: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 59: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 60: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 61: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 62: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 63: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 64: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 65: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 66: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 67: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 68: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 69: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 70: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 71: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 72: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 73: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 74: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 75: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 76: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 77: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 78: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 79: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 80: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 81: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 82: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 83: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 84: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 85: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 86: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 87: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 88: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 89: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 90: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 91: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 92: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 93: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 94: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 95: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 96: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 97: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 98: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 99: $300\ \text{k}\Omega$ resistor to pin 10.
 - Pin 100: $300\ \text{k}\Omega$ resistor to pin 10.
- MC1403 D/A Converter:**
 - Pin 1: $+5\ \text{V}$ supply.
 - Pin 2: $1\ \text{k}\Omega$ resistor to pin 2 of the MC14433.
 - Pin 3: Ground.
 - Pin 4: $0\ \text{V}$ output (v_x).

20 H	D7	D4	D3	D0
	符号	×	×	千 百

21 H	D7	D4	D3	D0
	+			个

```

ORG    001BH
LJMP   PINT1           ;跳外部中断 1 的中断服务程序

      ORG    0100H

INIT1: SETB   IT1       ;初始化程序,选择外中断 1 为跳沿触发方式
      MOV    IE, # 84H   ;CPU 开中断,允许外部中断 1 中断
      ;
PINT1: MOV    A, P1       ;外部中断 1 服务程序
      JNB    Acc. 4, PINT1;等待 DS1 选通信号的到来
      JB     Acc. 0, PEr  ;是否过、欠量程,是则转向 PEr 处理
      JB     Acc. 2, PL1  ;转换结果是正还是负,为正,跳 PL1
      SETB   07H         ;结果为负,符号位 07H 置 1
      AJMP   PL2

```

```

PL1:  CLR    07H        ;结果为正,符号位清 0
PL2:  JB     Acc. 3,PL3  ;千位的结果,千位为 0,跳 PL3
      SETB   04H        ;千位为 1,把 04H 位(即 20H 单元的 D4 位)置 1
      AJMP   PL4
PL3:  CLR    04H        ;千位为 0,把 04H 位清 0
PL4:  MOV     A,P1
      JNB    Acc. 5,PL4  ;等待百位的选通信号 DS2
      MOV    R0,#20H     ;指针指向 20H 单元
      XCHD   A,@R0       ;百位→20H 单元低 4 位
PL5:  MOV     A,P1
      JNB    Acc. 6,PL5  ;等待十位数的选通信号 DS3 的到来
      SWAP   A           ;读入十位,高低 4 位交换
      INC    R0          ;指针指向 21H 单元
      MOV    @R0,A       ;十位数的 BCD 码送入 21H 高 4 位
PL6:  MOV     A,P1
      JNB    Acc. 7,PL6  ;等待个位数选通信号 DS4 的到来
      XCHD   A,@R0       ;个位数送入 21H 单元的低 4 位
      RETI
PEr:  SETB   10H        ;置过量程、欠量程标志
      RETI               ;中断返回
    
```

MC14433 外接的积分元件 R_1 、 C_1 (图中的 4、5、6 引脚)大小和时钟有关,在实际应用中应加以调整,以得到正确的量程和线性度。积分电容也应选择聚丙烯电容器。

11.3 MCS-51 与 V/F 转换器的接口

目前,A/D 转换技术得到了广泛应用,利用 A/D 转换技术制成的各种测试仪器因其测量结果准确而受到欢迎。但在某些要求数据长距离传输,精确度要求较高的场合,采用一般的 A/D 转换技术就有许多不便,这时可使用 V/F 转换器代替 A/D 器件。

V/F 转换器是把电压信号转变为频率信号的器件,有良好的精度、线性度和积分输入特点,此外,它的应用电路简单,外围元件性能要求不高,适应环境能力强,转换速度不低于一般的双积分型 A/D 器件,且价格低,因此 V/F 转换技术广泛用于非快速 A/D 过程中。

V/F 转换器与单片机接口有以下特点:

(1) 接口简单、占用单片机硬件资源少。产生的频率信号可输入单片机的

一根 I/O 口线或作为中断源及计数输入等。

(2) 抗干扰性能好。用 V/F 转换器实现 A/D 转换, 就是频率计数的过程, 相当于在计数时间内对频率信号进行积分, 因而有较强的抗干扰能力。另外可采用光电耦合器连接 V/F 转换器与单片机之间的通道, 实现光电隔离。

(3) 便于远距离传输。可通过调制进行无线传输或光传输。

由于以上这些特点, V/F 转换器适用于一些非快速而需进行远距离信号传输的 A/D 转换过程。另外, 还可以简化电路、降低成本、提高性价比。

11.3.1 用 V/F 转换器实现 A/D 转换的原理

V/F 转换的工作原理如下: 同时启动频率计数器和定时器, 频率计数器把 V/F 转换器输出的频率信号作为计数脉冲, 进行定时计数, 当定时结束时, 定时器产生输出信号使频率计数器停止计数, 这样计数器的计数值与 V/F 转换器输出的脉冲频率信号之间的关系为:

$$f = \frac{D}{T}$$

上式中, D 是计数的值, T 是计数时间, 是已知的。可见, 只要知道了 D 值, 再除以计数的时间 T , 就可求出 V/F 转换器的输出频率, 从而知道输入电压 V , 这样就实现了 A/D 转换。定时器/计数器可用单片机内部的定时器/计数器, 也可使用外部扩展的定时器/计数器, 用单片机把计数值取入内存即可进行数据处理。

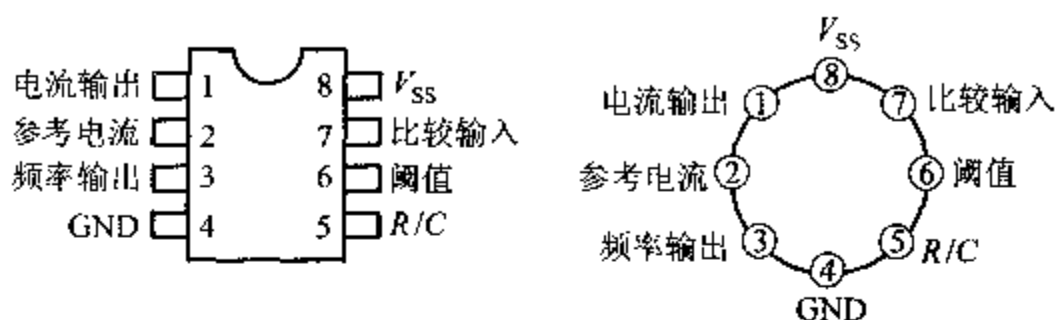
11.3.2 常用 V/F 转换器 LMX31 简介

常用的通用型 V/F 转换器为 LMX31 系列。LMX31 系列包括 LM131/LM231/LM331, 适用于 A/D 转换器、高精度 F/V 变换器、长时间积分器、线性频率调制或解调器等电路。

1. 主要特性

- (1) 频率范围: 1 Hz~100 kHz
- (2) 低非线性: $\pm 0.01\%$
- (3) 单电源或双电源供电
- (4) 单电源供电电压为 +5 V 时, 可保证转换精度
- (5) 温度特性: 最大 +50 ppm/°C
- (6) 低功耗: $V_{SS}=5$ V 时为 15 mW

有两种封装形式, 如图 11-24 所示。



2. 电特性参数:

- (1) 电源电压: +15 V
- (2) 输入电压范围: 0~10 V
- (3) 输出频率: 10 Hz~11 kHz
- (4) 非线性失真: $\pm 0.03\%$

3. LMX31 的 V/F 转换外部接线

LMX31 的 V/F 转换外部接线如图 11-25 所示。

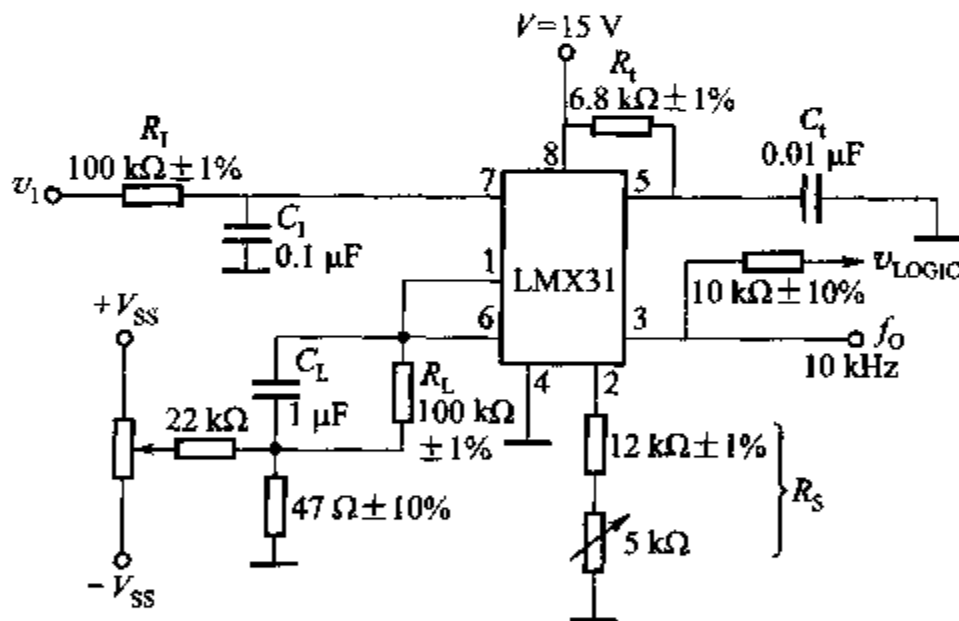


图 11-25 LMX31 外部接线图

11.3.3 V/F 转换器与 MCS-51 单片机接口

被测电压转换为与其成比例的频率信号后送入计算机进行处理。

(1) V/F 转换器可以直接与 MCS-51 单片机连接。这种接口方式比较简单,把频率信号接入单片机的定时器/计数器输入端即可。如图 11-26 所示(以 LM331 为例)。

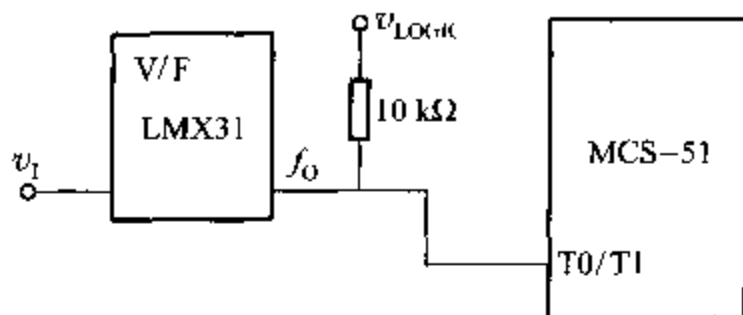


图 11-26 V/F 转换器与单片机接口

(2) 在一些电源干扰大、模拟电路部分容易对单片机产生电气干扰等恶劣环境中,可采用光电隔离的方法使 V/F 转换器与单片机无电信号联系,如图 11-27 所示。

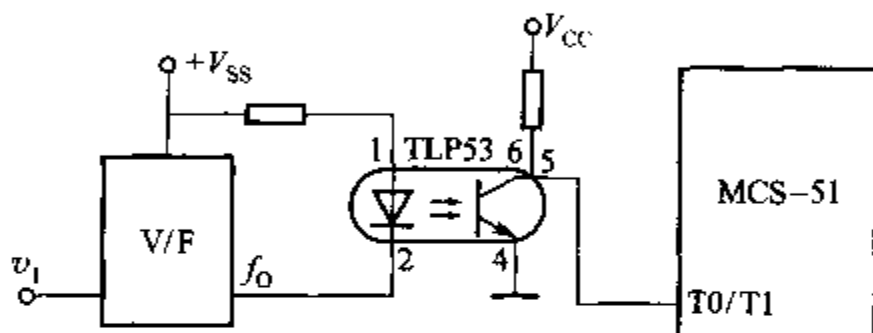


图 11-27 使用光电隔离器的接口

(3) 当 V/F 转换器与单片机之间距离较远时需要采用驱动电路以提高传输能力。一般可采用串行通信的驱动器和接收器来实现。例如使用 RS-422 的驱动器和接收器时,允许最大传输距离为 120 m,如图 11-28 所示。其中 SN75174/75175 是 RS-422 标准的四差分线路驱动/接收器。

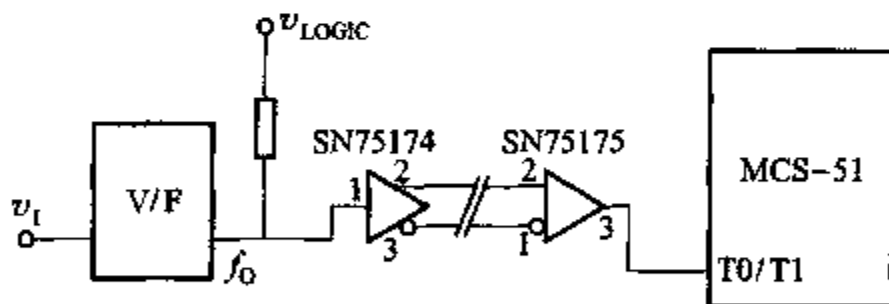


图 11-28 利用串行通信器件的接口

(4) 采用光纤或无线传输时,需配以发送、接收装置。如图 11-29、图 11-30 所示。

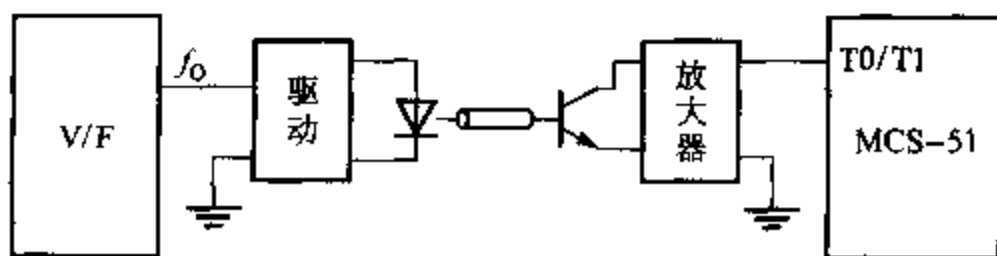


图 11-29 利用光纤进行传输的接口

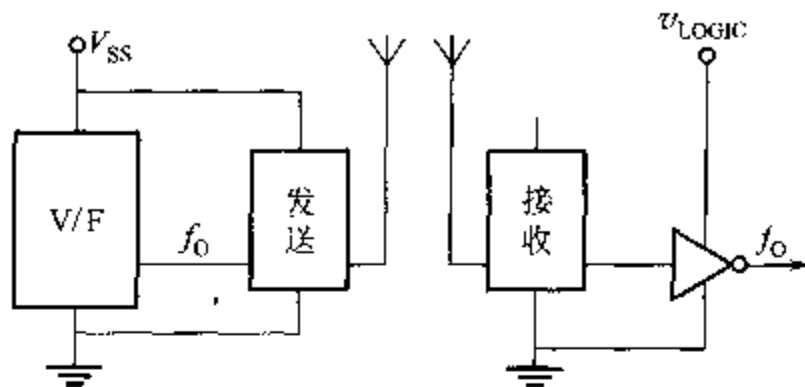


图 11-30 利用无线传输设备作输入通道

11.3.4 LM331 应用举例

本例使用 LM331 和 8031 的内部定时器构成 A/D 转换电路，具有使用元件少、成本低、精度高的特点。

1. 接口电路

MCS-51 与 LM331 的接口电路如图 11-31 所示，

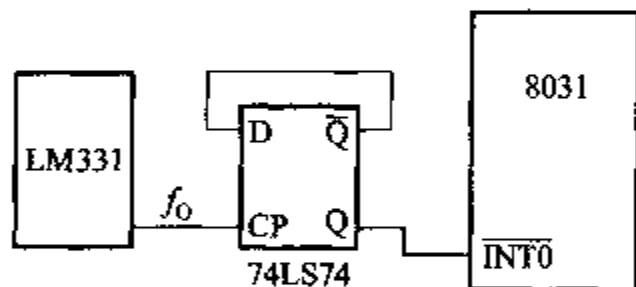


图 11-31 LM331 应用电路

V/F 转换器最大输出频率为 10 kHz，输入电压范围为 0~10 V。由于本电路 V/F 输出频率较低，如用其作为计数脉冲则会降低精度，因此采用测周期的方法。V/F 输出的频率经 D 触发器二分频后接至 $\overline{\text{INT0}}$ ，作为 T0 计数器的控制信号。T0 计数器置定时器状态，取方式 1，将 TMOD.3 (T0 的 GATE 位) 置 1，这样就由 $\overline{\text{INT0}}$ 和 TR0 来共同决定计数器是否工作。这种方法只能测量信号周期小于 65 535 个机器周期的信号。

2. 软件设计

程序包括初始化和计数两部分。初始化程序要对定时器/计数器 0 进行状态设置，使其工作在定时器工作模式，方式 1，并将 GATE 位置 1。计数程序首先需判断 $\overline{\text{INT0}}$ 的电平，当其为低时，打开 TR0 位准备计数；当其变为高时，启动计数，再为低时停止计数并清 TR0，取出数据，将 T0 的时间常数寄存器 TH0、TL0 清 0，准备下一次计数。程序如下：

```
BEGIN:  NOP
        MOV  TMOD, #09H      ;定时器 T0 初始化
        MOV  TL0, #00H
        MOV  TH0, #00H
LOOP1:  NOP
        JB   P3.2, LOOP1
        SETB TR0
LOOP2:  NOP
        JNB  P3.2, LOOP2
LOOP3:  NOP
        JB   P3.2, LOOP3
        CLR  TR0
        MOV  B, TH0          ;高位进 B 寄存器
        MOV  A, TL0          ;低位进 A 寄存器
        MOV  TL0, #00H
        MOV  TH0, #00H
        AJMP LOOP1
```

本程序将计数结果高位存入 B, 低位存入 A, 以便后期处理。

思考题及习题

1. 对于电流输出的 D/A 转换器, 为了得到电压的转换结果, 应使用()。
2. D/A 转换器的主要性能指标都有哪些? 设某 DAC 为二进制 12 位, 满量程输出电压为 5 V, 试问它的分辨率是多少?
3. 说明 DAC 用作程控放大器的工作原理。
4. 使用双缓冲方式的 D/A 转换器, 可实现多路模拟信号的()输出。
5. MCS-51 与 DAC0832 接口时, 有哪三种连接方式? 各有什么特点? 各适合在什么场合使用?
6. A/D 转换器两个最重要的指标是什么?
7. 分析 A/D 转换器产生量化误差的原因, 1 个 8 位的 A/D 转换器, 当输入电压为 0~5 V 时, 其最大的量化误差是多少?
8. 目前应用较广泛的 A/D 转换器主要有哪几种类型? 它们各有什么特点?
9. DAC 和 ADC 的主要技术指标中, 量化误差, 分辨率和精度有何区别?
10. 在 1 个由 8031 单片机与 1 片 ADC0809 组成的数据采集系统中, ADC0809 的 8 个输入通道的地址为 7FF8H~7FFFH, 试画出有关接口电路图, 并编写出每隔 1 分钟轮流采集 1 次 8 个通道数据的程序, 共采样 50 次, 其采样值存入片外 RAM 2000H 单元开始的存储区中。

11. 根据图 11-19, 8031 控制 12 位 A/D 转换器 AD574 采集 10 个数据, 并将这 10 个数据送到内部 RAM 起始地址为 40H 的单元中, 偶地址单元存高 4 位, 奇地址存低 8 位, 编写相应的程序。

12. 判断下列说法是否正确?

- (A) 转换速率这一指标仅适用于 A/D 转换器, D/A 转换器不用考虑转换速率这一问题。
- (B) ADC0809 可以利用转换结束信号 EOC 向 8031 发出中断请求。
- (C) 输出模拟量的最小变化量称为 A/D 转换器的分辨率。
- (D) 对于周期性的干扰电压, 可使用双积分的 A/D 转换器, 并选择合适的积分元件, 可以将该周期性的干扰电压带来的转换误差消除。

第 12 章 MCS-51 的功率接口设计

在 MCS-51 单片机应用系统中,有时需要用单片机控制各种各样的高压、大电流负载,这些大功率负载如电动机、电磁铁、继电器、灯泡等,显然不能用单片机的 I/O 线来直接驱动,而必须通过各种驱动电路和开关电路来驱动。此外,为了使 MCS-51 与强电隔离和抗干扰,有时需加接光电耦合器。本章将介绍这些外围驱动电路、光电耦合器与 MCS-51 单片机的接口电路。我们称此类接口为 MCS-51 的功率接口。

12.1 MCS-51 的输出驱动能力及其外围集成 数字驱动电路

12.1.1 MCS-51 片内 I/O 口的驱动能力

在工业生产现场,有不少控制对象是电磁继电器、电磁开关或晶闸管、固态继电器和功率电子开关,其控制信号都是开关电平量。能否用 MCS-51 片内的 I/O 口直接驱动它们呢?这首先就要了解 MCS-51 片内 I/O 口的驱动能力。

MCS-51 有 4 个并行双向口,每个口由 1 个锁存器、1 个输出驱动器和 1 个输入缓冲器组成。如不用外部存储器,P0、P1、P2、P3 4 个口都可做输出口,但其驱动能力不同,P0 口的驱动能力较大,每位可驱动 8 个 LSTTL 输入,即当其输出高电平时,可提供 400 μ A 的电流;当其输出低电平(0.45 V)时,则可提供 3.2 mA 的灌电流,如低电平允许提高,灌电流可相应加大。P1、P2、P3 口的每一位只能驱动 4 个 LSTTL,即可提供的电流只有 P0 口的一半。所以,任何一个口要想获得较大的驱动能力,只能用低电平输出。8031 通常要用 P0、P2 口作访问外部存储器用,所以只能用 P1、P3 口作输出口。P1、P3 口的驱动能力有限,在低电平输出时,一般也只能提供不到 2 mA 的灌电流,所以通常要加总线驱动器或其他驱动电路。

12.1.2 外围集成数字驱动电路

表 12-1 给出了常用的外围集成数字驱动电路的参数。

这些驱动电路只要加接合适的限流电阻和偏置电阻,即可直接由 TTL、MOS 以及 CMOS 电路来驱动。当它们用于驱动感性负载时,必须加接限流电阻或箝位二极管。此外,有些驱动器内部还设有逻辑门电路,可以完成与、与非、或以及或非的逻辑功能。

下面举例说明外围集成数字驱动电路的应用。

例 12-1 慢开启的白炽灯驱动电路

图 12-1 为慢开启白炽灯驱动电路,白炽灯的延时开启时间长短取决于时间常数 RC 。此电路能直接驱动工作电压小于 30 V、额定电流小于 500 mA 的任何灯泡。注意:在设计此电路的印制电路板时,驱动器要加装散热板,以便散热。

例 12-2 大功率音频振荡器

图 12-2 给出的电路能直接驱动一个大功率的扬声器,可用于报警系统,改变电路中的电阻或电容的值便能改变电路的振荡频率。电路中的两个齐纳二极管 IN751A 用于输入端的保护。

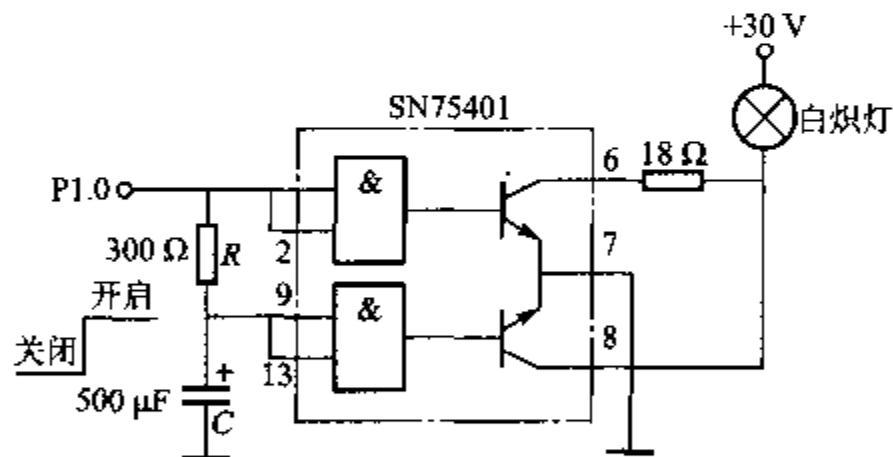


图 12-1 慢开启白炽灯驱动电路

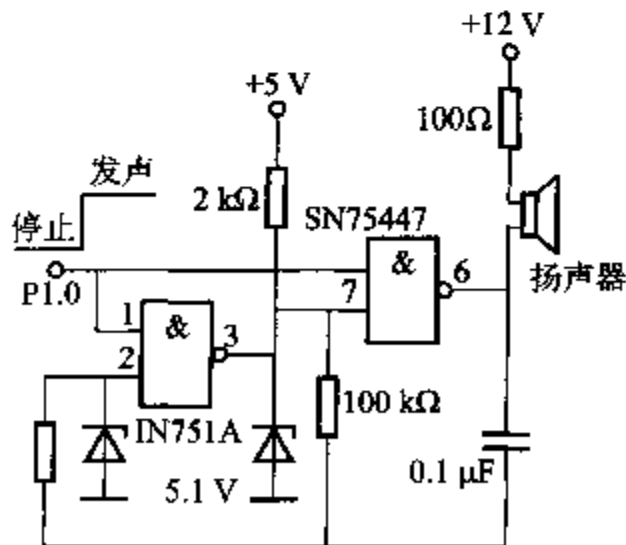


图 12-2 大功率音频振荡器

表 12-1 常用集成驱动器芯片性能参数表

	与	与 非	或	或 非	最大 工作 电压 /V	最大 输出 电流 /mA	驱动 器 的 数 目	典型 延 迟 时 间 /ns
具有 逻辑 门的 驱动 器	SN75431	SN75432	SN75433	SN75434	15	300	2	15
	SN75451B	SN75452B	SN75453B	SN75454B	20	300	2	21
	SN75461	SN75462	SN75463	SN75464	30	300	2	33
	SN75401	SN75402	SN75403	SN75404	30	500	2	33
		SN75437			35	700	1	300
	SN75446	SN75447	SN75448	SN75449	50	350	2	300
	SN75471	SN75472	SN75473	SN75474	55	300	2	30~100
	SN75476	SN75477	SN75478	SN75479	55	300	2	30~100
	SN75411	SN75412	SN75413	SN75414	55	500	2	30~100
	SN75416	SN75417	SN75418	SN75419	55	500	2	30~100
无 逻辑 门 驱 动 器	SN75064	SN75066	SN75068		35	1 500	4	500
	ULN2064	SN75074	ULN2068		35	1 500	4	500
	ULN2074	ULN2066	ULN2841	ULN2845	35	1 500	4	500
	ULN2001A	ULN2002A	ULN2003A	ULN2004A	40	350	7	1 000
	MC1411	MC1412	MC1413	MC1416	50	500	7	1 000
	SN75065	SN75067	SN75069		50	1 500	4	500
	ULN2065	SN75075	ULN2069		50	1 500	4	500
	ULN2075	ULN2067			50	1 500	4	500
	SN75466	SN75467	SN75468	SN75469	60	500	7	130

例 12-3 驱动大电流负载

其电路如图 12-3 所示。ULN2068 芯片具有 4 个大电流达林顿开关,能驱动电流高达 1.5 A 的负载。由于 ULN2068 在 25℃ 时功耗达 2 075 mW,因而使用时一定要加散热板。

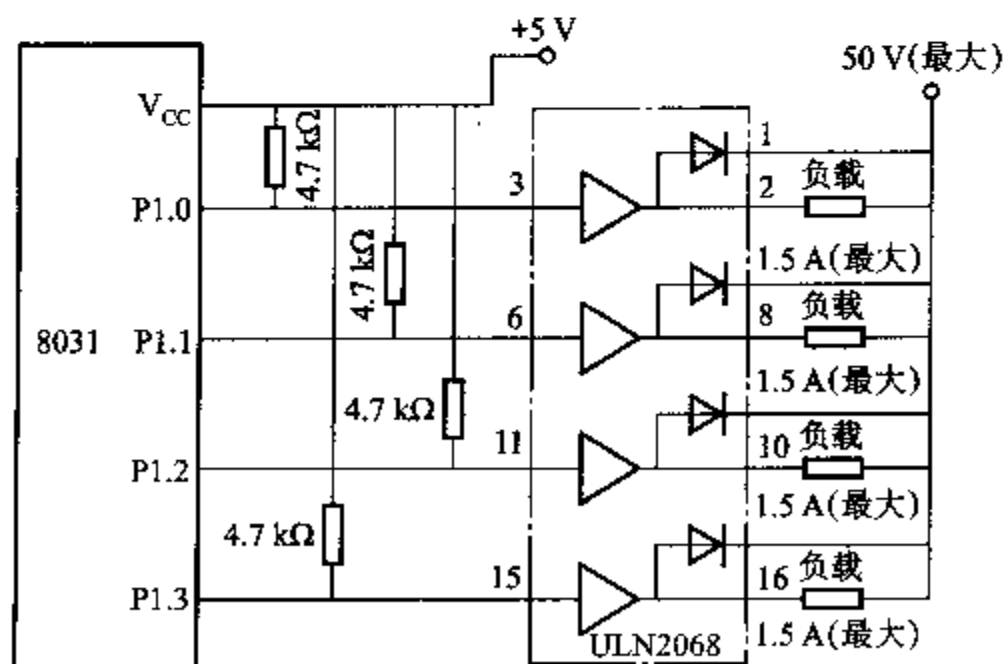


图 12-3 使用 ULN2068 的大电流驱动电路

12.2 MCS-51 的开关型功率接口

开关型功率驱动接口中单片机控制输出的信号是开关量,常用的开关型驱动器件有:光电耦合器、继电器、晶闸管、功率 MOS 管、集成功率电子开关、固态继电器等。下面介绍这些器件与单片机的接口。

12.2.1 MCS-51 与光电耦合器的接口

1. 晶体管输出型光电耦合器驱动接口

晶体管输出型光电耦合器的受光器是光电晶体管。光电晶体管除了没有使用基极外,跟普通晶体管一样。取代基极电流的是以光作为晶体管的输入。当光电耦合器的发光二极管发光时,光电晶体管受光的影响在 cb 间和 ce 间有电流流过,这两个电流基本上受光的照度控制,常用 ce 极间的电流作为输出电流,输出电流受 V_{ce} 的电压影响很小,在 V_{ce} 增加时,稍有增加。光电晶体管的集电极电流 I_c 与发光二极管的电流 I_F 之比称为光电耦合器的电流传输比。不同结构的光电耦合器的电流传输比相差很大。如输出端是单个晶体管的光电耦合器 4N25 的电流传输比是 $\geq 20\%$ 。输出端使用达林顿管的光电耦合器 4N33 的电流传输比是 $\geq 500\%$ 。电流传输比受发光二极管的工作电流大小影响,电流为 $10\sim 20\text{ mA}$ 时,电流传输比最大,电流小于 10 mA 或大于 20 mA ,传输比都下降。温度升高,传输比也会下降,因此在使用时要留一些余量。

光电耦合器在传输脉冲信号时,对不同结构的光电耦合器的输入输出延迟时间相差很大。4N25 的导通延迟 t_{on} 是 $2.8\text{ }\mu\text{s}$,关断延迟 t_{off} 是 $4.5\text{ }\mu\text{s}$,4N33 的导通延迟 t_{on} 是 $0.6\text{ }\mu\text{s}$,关断延迟 t_{off} 是 $45\text{ }\mu\text{s}$ 。

晶体管输出型光电耦合器可作为开关使用,这时发光二极管和光电晶体管平常都处于关断状态。在发光二极管通过电流脉冲时,发光晶体管在电流脉冲持续的时间内导通。光电耦合器也可作线性耦合器使用,在发光二极管上提供一个偏置电流,再把信号电压通过电阻耦合到发光二极管上,引起其亮度的变化,从而输出电流也就将随输入的信号电压线性变化。

图 12-4 是使用 4N25 的光电耦合器接口电路图。4N25 起到耦合脉冲信号和隔离单片机系统与输出部分的作用,使两部分的电流信号独立。输出部分的地线接机壳或接大地,而 8031 系统的电源地线浮空,不与交流电源的地线相接。这样可以避免输出部分电源变化对单片机电源的影响,减少系统所受的干扰,提高系统的可靠性。4N25 输入输出端的最大隔离电

压 $> 2500\text{ V}$ 。

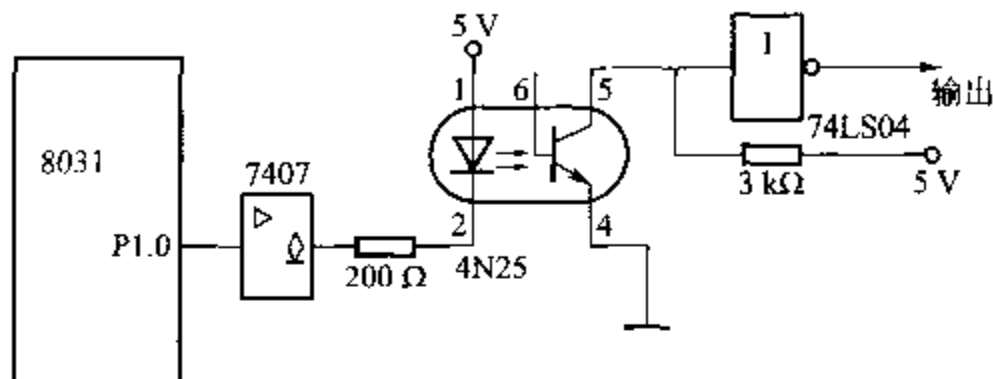


图 12-4 光电耦合器 4N25 的接口电路

图 12-4 接口电路中使用同相驱动器 7407 作为光电耦合器 4N25 输入端的驱动。光电耦合器输入端的电流一般为 $10 \sim 15\text{ mA}$ ，发光二极管的压降约为 $1.2 \sim 1.5\text{ V}$ 。限流电阻由下式计算：

$$R = \frac{V_{CC} - (V_F + V_{CS})}{I_F}$$

式中， V_{CC} 为电源电压； V_F 为输入端发光二极管的压降，取 1.5 V ； V_{CS} 为驱动器的压降； I_F 为发光二极管的工作电流。

如图 12-4 电路要求 I_F 为 15 mA ，则限流电阻计算如下：

$$R = \frac{V_{CC} - V_F - V_{CS}}{I_F} = \frac{5\text{ V} - 1.5\text{ V} - 0.5\text{ V}}{0.015\text{ A}} = 200\ \Omega$$

当 8031 的 P1.0 端输出高电平时，4N25 输入端电流为 0，输出相当于开路，74LS04 的输入端为高电平，输出为低电平。8031 的 P1.0 端输出低电平时，7407 输出端为低电压输出，4N25 的输入电流为 15 mA ，输出端可以流过 $\geq 3\text{ mA}$ 的电流。如果输出端负载电流小于 3 mA ，则输出端相当于 1 个接通的开关。74LS04 输出高电平。4N25 的 6 引脚是光电晶体管的基极，在一般的使用中可以不用接，该引脚悬空。

由于光电耦合器是电流型输出，不受输出端工作电压的影响。因此可以用于不同电平的转换。若图 12-4 的电路中，输出部分不是使用 74LS04，而是要求使用 CMOS 的反相器 MC14069，工作电压用 15 V ，这时只需把 $3\text{ k}\Omega$ 的电阻改为 $10\text{ k}\Omega$ ，工作电压由 5 V 改为 15 V ，74LS04 改用 MC14069 即可。当 P1.0 端输出高电平时，光电耦合器的输出端相当于开路，MC14069 的输入端电压为 15 V 。当 P1.0 端输出低电平时，光电耦合器的输出晶体管导通，MC14069 的输入端电压接近 0 V 。4N25 输出端晶体管的 ce 极间的耐压大于 30 V ，所以 4N25 最大的电平转换可到 30 V 。

光电耦合器也常用于较远距离的信号隔离传送。一方面光电耦合器

可以起到隔离两个系统地线的作用,使两个系统的电源相互独立,消除地电位不同所产生的影响。另一方面,光电耦合器的发光二极管是电流驱动器件,可以形成电流环路的传送形式。由于电流环电路是低阻抗电路,它对噪音的敏感度低,因此提高了通信系统的抗干扰能力。常用于在有噪音干扰的环境下传输信号。图 12-5 是用光电耦合器组成的电流环发送和接收电路。

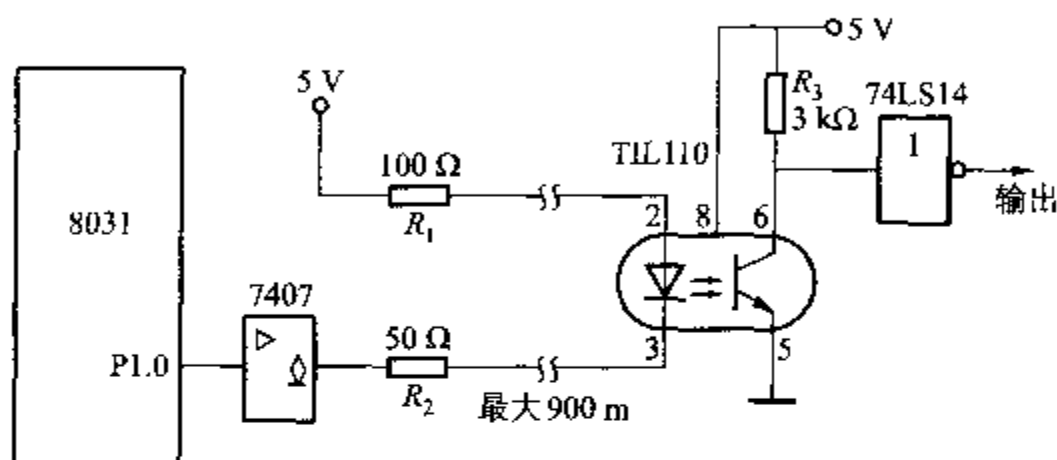


图 12-5 电流环电路

图 12-5 电路可以用来传输数据,最大速率为 50 Kb/s,最大传输距离为 900 m。环路连线的电阻对传输距离影响很大,此电路中环路连线电阻不能大于 30 Ω,当连线电阻较大时,100 Ω 的限流电阻要相应减小。光电耦合管使用 TIL110,TIL110 的功能与 4N25 相同,但开关速度比 4N25 快,当传输速度要求不高时,也可以用 4N25 代替。电路中光电耦合器放在接收端,输入端由同相驱动器 7407 驱动,限流电阻分为 2 个,1 个是 50 Ω,1 个是 100 Ω。50 Ω 电阻的作用除了限流外,最主要的还是起阻尼的作用,防止传送的信号发生畸变和产生突发的尖峰。电流环的电流计算如下:

$$I_F = \frac{V_{CC} - V_F - V_{CS}}{R_1 + R_2} = \frac{5 - 1.5 - 0.5}{50 + 100} \text{ A} = 0.02 \text{ A} = 20 \text{ mA}$$

TIL110 的输出端接一个带施密特整形电路的反相器 74LS14,作用是提高抗干扰能力。施密特触发电路的输入特性有一个回差。输入电压大于 2 V 才认为是高电平输入,小于 0.8 V 才认为是低电平输入。电平在 0.8~2 V 之间变化时,则不改变输出状态。因此信号经过 74LS14 之后便更接近理想波形。

表 12-2 为常用的晶体管输出型光电耦合器,供读者在选用光电耦合器时参考。

表 12-2 晶体管输出型光电耦合器

类 型	器 件 型 号	输 出 结 构	发射极正向电压(最大)	最小输出电压 (V_{ce})	典型 h_{FE}	最小 DC 冲击隔离电压	典型工作速度或带宽	应 用
晶体管型	MCT2	晶体管	1.5 V@20 mA	30 V	250	3 550 V	150 kHz	AC 线/数字逻辑之间的隔离,用于线性接收、继电器监控、电源监控、开关网络、传感系统、开关电源、通信系统等
	MCT271 MCT274	晶体管	1.5 V@20 mA	30 V	120 360	3 550 V	7 μ s 25 μ s	
	4N25 4N27 4N35 4N38	晶体管	1.5 V@10 mA 1.5 V@10 mA	30 V 80 V	250 325 100 250	2 550 V 1 550 V 3 500 V 2 500 V	300 kHz 300 kHz 150 kHz 0.8/7 μ s	
	TIL111 TIL112 TIL116 TIL117 TIL124	晶体管	1.4 V@16 mA 1.5 V@10 mA 1.5 V@60 mA 1.4 V@16 mA 1.4 V@10 mA	30 V 20 V 30 V 30 V 30 V	300 200 300 550 100	1 500 V 1 500 V 2 500 V 2 500 V 5 000 V	5 μ s 2 μ s 5 μ s 5 μ s 2 μ s	
	MCT275 MOC8204 MOC8025 MOC8026	晶体管	1.5 V@20 mA 1.5 V@10 mA	80 V 400 V	170 — — —	3 550 V — — —	4.5/3.5 μ s 5 μ s	
	TIL113 TIL119 TIL156	达林顿晶体管	1.5 V@10 mA 1.5 V@10 mA 1.5 V@10 mA	30 V 30 V 30 V	15 000 — 15 000	1 500 V 1 500 V 3535 V	300 μ s 300 μ s	大电流、低容抗、快速关断等器件的控制。用于通信、遥控逻辑隔离、报警监控电路等
	4N29 4N32		1.5 V@10 mA 1.5 V@10 mA	30 V 30 V	15 000	2 500 V 2 500 V	2/25 μ s 2/60 μ s	
	MOC8020 MOC8030		2 V@10 mA	50 V 80 V	— —	— —	13/60 μ s	
达林顿输出型								

续表

类型	器件型号	输出结构	发射极正向电压(最大)	最小输出电压(V_{ce})	典型 h_{FE}	最小DC冲击隔离电压	典型工作速度或带宽	应用
AC输入型	H11AA1	晶体管输出	1.5 V@10 mA	30 V	400	2550 V	—	用于监控AC掉电的情况
	MHD400	集电极开路逻辑门	1.5 V@30 mA	—		3550 V	1 ms	

2. 晶闸管输出型光电耦合器驱动接口

晶闸管输出型光电耦合器的输出端是光敏晶闸管或光敏双向晶闸管。当光电耦合器的输入端有一定的电流流入时,晶闸管即导通。有的光电耦合器的输出端还配有过零检测电路,用于控制晶闸管过零触发,以减少用电器在接通电源时对电网的影响。

4N40是常用的单向晶闸管输出型光电耦合器。当输入端有15~30 mA电流时,输出端的晶闸管导通。输出端的额定电压为400 V,额定电流有效值为300 mA。输入输出端隔离电压为1 500~7 500 V。4N40的6引脚是输出晶闸管的控制端,不使用此端时,此端可对阴极接1个电阻。

MOC3041是常用的双向晶闸管输出的光电耦合器,带过零触发电路,输入端的控制电流为15 mA,输出端额定电压为400 V,最大重复浪涌电流为1 A,输入输出端隔离电压为7 500 V。MOC3041的5引脚是器件的衬底引出端,使用时不需要接线。图12-6是4N40和MOC3041的接口驱动电路。

4N40输入端限流电阻的计算:

$$R = \frac{V_{CC} - V_F - V_{CS}}{I_F} = \frac{5 - 1.5 - 0.5}{0.03} \Omega = 100 \Omega$$

实际应用中可以留一些余量,限流电阻取91 Ω 。

MOC3041输入端限流电阻的计算:

$$R = \frac{V_{CC} - V_F - V_{CS}}{I_F} = \frac{5 - 1.5 - 0.5}{0.015} \Omega = 200 \Omega$$

为留一定的余量,限流电阻选180 Ω 。

4N40常用于小电流用电器的控制,如指示灯等,也可以用于触发大功率的晶闸管。MOC3041一般不直接用于控制负载,而用于中间控制电路或用于触发大功率的晶闸管。

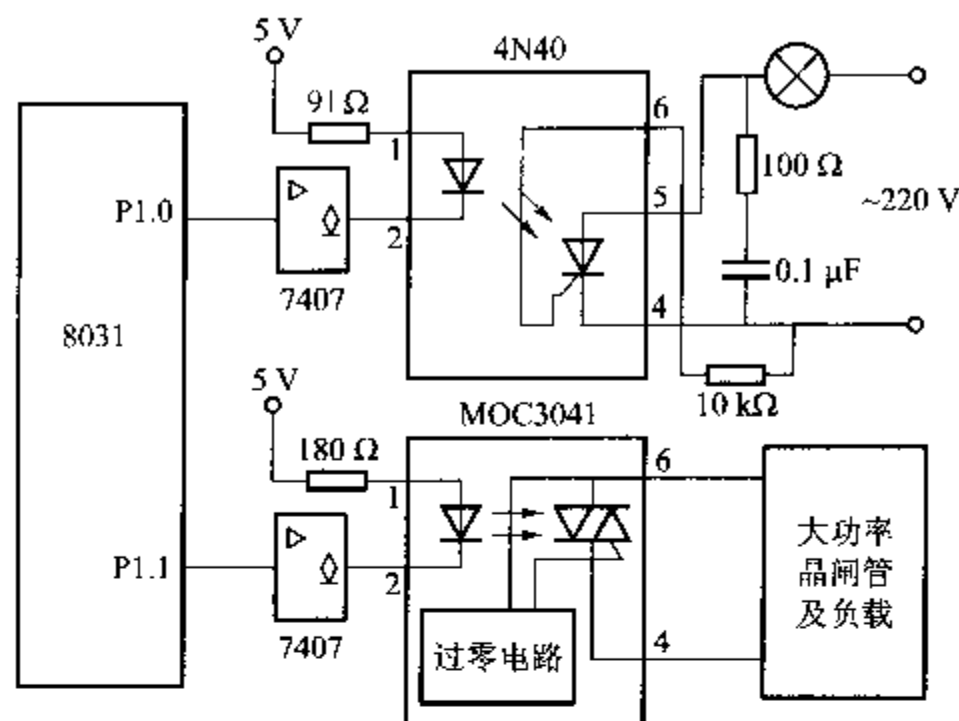


图 12-6 晶闸管输出型光电耦合器驱动接口

12.2.2 MCS-51 与继电器的接口

1. 直流电磁式继电器功率接口

直流电磁式继电器，一般用功率接口集成电路或晶体管驱动。在使用较多继电器的系统中，可用功率接口集成电路驱动，例如 SN75468 等。1 片 SN75468 可以驱动 7 个继电器，驱动电流可达 500 mA，输出端最大工作电压为 100 V。

常用的继电器大部分属于直流电磁式继电器，也称为直流继电器。图 12-7 是直流继电器的接口电路图。

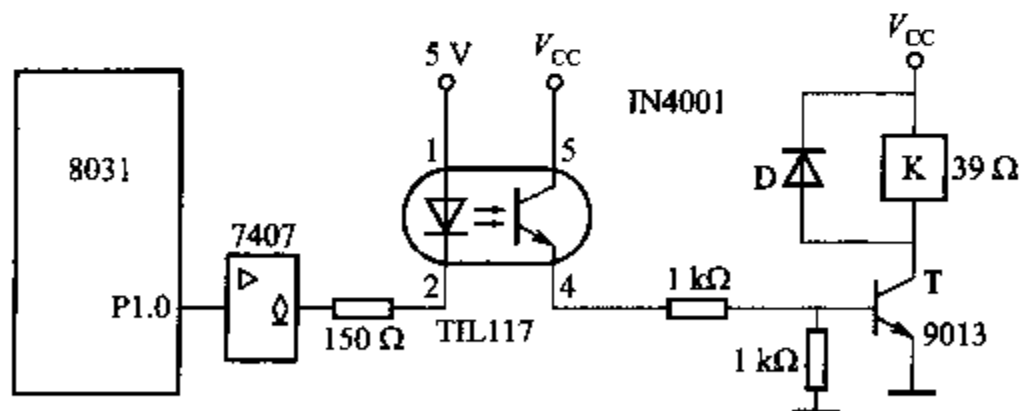


图 12-7 直流继电器接口

继电器的动作由单片机 8031 的 P1.0 端控制。P1.0 端输出低电平时，继电器 K 吸合；P1.0 端输出高电平时，继电器 K 释放。采用这种控制逻辑可以使继电器在上电复位或单片机受控复位时不吸合。

继电器 K 由晶体管 9013 驱动, 9013 可以提供 300 mA 的驱动电流, 适用于继电器线圈工作电流小于 300 mA 的场合。V_{CC} 的电压范围是 6~30 V。光电耦合器使用 TIL117。TIL117 有较高的电流传输比, 最小值为 50%。晶体管 9013 的电流放大倍数大于 50。当继电器线圈工作电流为 300 mA 时, 光电耦合器需要输出大于 6.8 mA 的电流, 其中 9013 基极对地的电阻分流约 0.8 mA。输入光电耦合器的电流必须大于 13.6 mA, 才能保证向继电器提供 300 mA 的电流。光电耦合器的输入电流由 7407 提供, 电流约为 20 mA。

二极管 D 的作用是保护晶体管 T。当继电器 K 吸合时, 二极管 D 截止, 不影响电路工作。继电器释放时, 由于继电器线圈存在电感, 这时晶体管 T 已经截止, 所以会在线圈的两端产生较高的感应电压。这个感应电压的极性是上负下正, 正端接在 T 的集电极上。当感应电压与 V_{CC} 之和大于晶体管 T 的集电结反向耐压时, 晶体管 T 就有可能损坏。加入二极管 D 后, 继电器线圈产生的感应电流由二极管 D 流过, 因此不会产生很高的感应电压, 晶体管 T 得到了保护。

2. 交流电磁式接触器的功率接口

继电器中切换电路能力较强的电磁式继电器称为接触器。接触器的触点数一般较多。交流电磁式接触器由于线圈的工作电压要求是交流电, 所以通常使用双向晶闸管驱动或使用一个直流继电器作为中间继电器控制。图 12-8 是交流接触器的接口电路图。

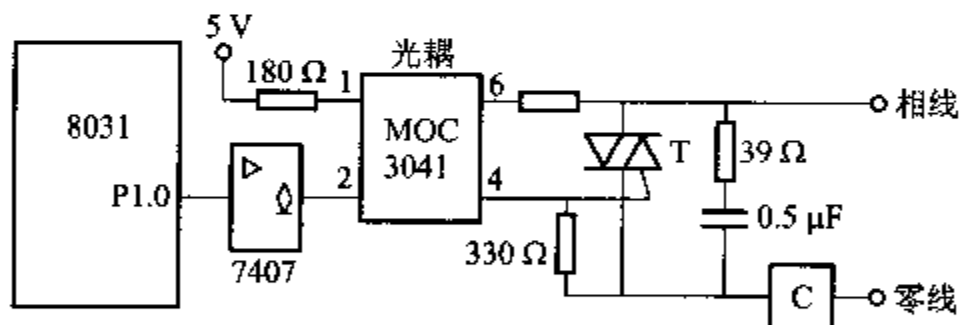


图 12-8 交流接触器接口

交流接触器 C 由双向晶闸管 T 驱动。双向晶闸管的选择要满足: 额定工作电流为交流接触器线圈工作电流的 2~3 倍; 额定工作电压为交流接触器线圈工作电压的 2~3 倍。对于工作电压 220 V 的中、小型交流接触器, 可以选择 3 A、600 V 的双向晶闸管。

光电耦合器 MOC3041 的作用是触发双向晶闸管 T 以及隔离单片机系统和接触器系统。光电耦合器 MOC3041 的输入端接 7407, 由单片机 8031 的 P1.0 端控制。P1.0 输出低电平时, 双向晶闸管 T 导通, 接触器 C 吸合。P1.0 输出高电平时, 双向晶闸管 T 关断, 接触器 C 释放。MOC3041 内部带有过零控制电路, 因此双向晶闸管 T 工作在过零触发方式。接触器动作时, 电源电压较低, 这时接通用电器, 对电源的影响较小。

12.2.3 MCS-51 与晶闸管的接口

1. 单向晶闸管

晶闸管(整流元件)英文名为 thyristor,这是一种大功率半导体器件,它既有单向导电的整流作用,又有可以控制的开关作用。利用它可用较小的功率控制较大的功率。在交、直流电动机调速系统、调功系统、随动系统和无触点开关等方面均获得广泛的应用,如图 12-9 所示,它外部有三个电极:阳极 A、阴极 C、控制极(门极)G。

与二极管不同的是当其两端加上正向电压而控制极不加电压时,晶闸管并不导通,其正向电流很小,处于正向阻断状态;当加上正向电压,且控制极上(与阴极间)也加上一正向电压时,晶闸管便进入导通状态,这时管压降很小(1 V 左右)。这时即使控制电压消失,仍能保持导通状态,所以控制电压没有必要一直存在,通常采用脉冲形式,以降低触发功耗。它不具有自关断能力,要切断负载电流,只有使阳极电流减小到维持电流以下,或加上反向电压实现关断。若在交流回路中应用,当电流过零和进入负半周时,自动关断,为了使其再次导通,必须重加控制信号。

2. 双向晶闸管

晶闸管应用于交流电路控制时,如图 12-10 所示,采用两个器件反并联,以保证电流能沿正反两个方向流通。

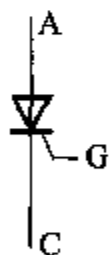


图 12-9 单向晶闸管结构符号

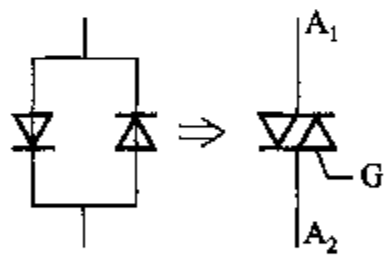


图 12-10 双向晶闸管结构

如把两只反并联的 SCR 制作在同一片硅片上,便构成双向晶闸管,控制极共用一个,使电路大大简化,其特性如下:

① 控制极 G 上无信号时, A_1 、 A_2 之间呈高阻抗,管子截止。

② $V_{A1A2} > 1.5 \text{ V}$ 时,不论极性如何,均可利用 G 触发电流控制其导通。

③ 工作于交流时,当每一半周交替时,纯阻负载一般能恢复截止;但在感性负载情况下,电流相位滞后于电压,电流过零,可能反向电压超过转折电压,使管子反向导通。所以,要求管子能承受这种反向电压,而且一般要加 RC 吸收回路。

④ A_1 、 A_2 可调换使用,触发极性可正可负,但触发电流有差异。

双向晶闸管经常用作交流调压、调功、调温和无触点开关,过去其触发脉冲

一般都用硬件产生,故检测和控制都不够灵活,而在单片机控制应用系统中则经常可利用软件产生触发脉冲。

3. 光耦合双向晶闸管驱动器

这种器件是一种单片机输出与双向晶闸管之间较理想的接口器件,它由输入和输出两部分组成,输入部分是一砷化镓发光二极管,该二极管在 $5\sim 15\text{ mA}$ 正向电流作用下发出足够强度的红外光,触发输出部分。输出部分是一硅光敏双向晶闸管,在红外线的作用下可双向导通。该器件为 6 引脚双列直插式封装,其引脚配置和内部结构见图 12-11。

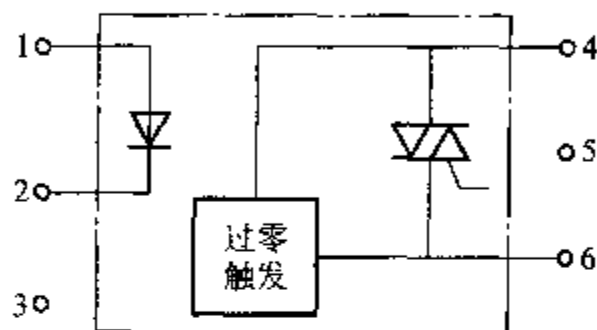
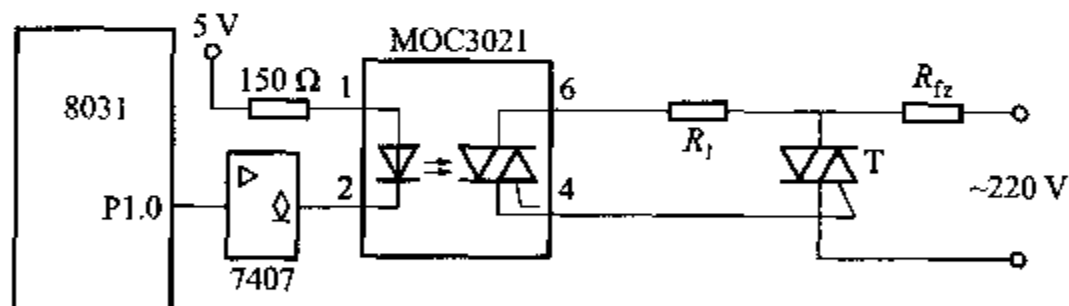
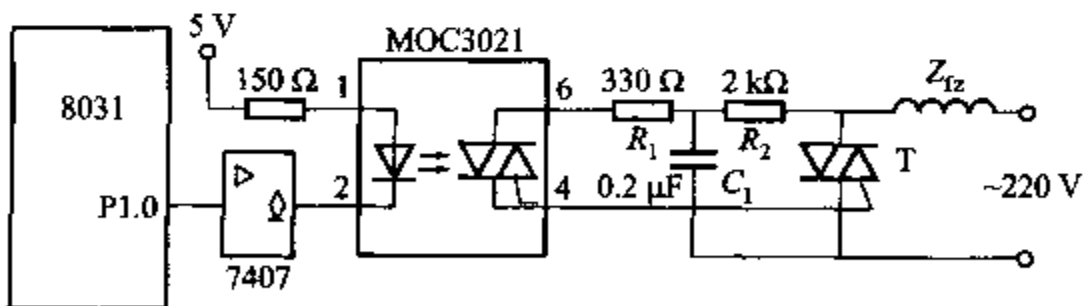


图 12-11 光耦合双向晶闸管驱动器引脚与结构

有的型号的光耦合双向晶闸管驱动器还带有过零检测器,以保证在电压为零(接近于零)时才触发晶闸管导通,如 MOC3030/31/32(用于 115 V 交流),MOC3040/41(用于 220 V 交流)。图 12-12 为这类光耦驱动器与双向晶闸管的典型电路。



(a) 电阻性负载



(b) 电感性负载

图 12-12 双向晶闸管型触发电路

输出端的双向晶闸管导通,触发外部的双向晶闸管 T 导通。当 P1.0 输出高电平时,MOC3021 输出端的双向晶闸管关断,外部双向晶闸管 T 也关断。电

阻 R_1 的作用是限制流过 MOC3021 输出端的电流不要超过 1 A 。 R_1 的大小由下式计算:

$$R_1 = \frac{V_F}{I_F} = \frac{220 \cdot \sqrt{2}}{1} \Omega = 311 \Omega$$

R_1 取 300Ω 。由于串入电阻 R_1 ,使得触发电路有一个最小触发电压,低于这个电压时,T 才导通。最小触发电压 V_T 由下式计算:

$$V_T = R_1 \cdot I_{GT} + V_{GC} + V_{TM} = 300 \times 0.05 \text{ V} + 2 \text{ V} + 3 \text{ V} = 21.5 \text{ V}$$

对应的最小控制角为:

$$\alpha = \arcsin \frac{V_T}{V_P} = \arcsin \frac{21.5}{311} = 3.96^\circ$$

即控制角不能小于 3.96° ,也只有在 3.96° 时,内部双向晶闸管才导通。当外接的双向晶闸管功率较大时,需要较大 I_{GT} ,这时最小控制角比较大,可能会超出使用的要求。解决的方法是在大功率晶闸管和 MOC3021 之间再加入一个触发用的晶闸管,这个触发用的晶闸管的限流电阻可以用得比较小,所以最小控制角也可以做得比较小。当负载为感性负载时,由于电压上升率 dv/dt 较大,有可能超过 MOC3021 允许的范围。在阻断状态下,晶闸管的 PN 结相当于一个电容,如果突然受到正向电压,充电电流流过门极 PN 结时,起了触发电流的作用。当电压上升率 dv/dt 较大时,就会造成 MOC3021 的输出晶闸管误导通。因此,在 MOC3021 的输出回路中加入 R_2 和 C_1 组成的 RC 回路,降低电压上升率 dv/dt ,使 dv/dt 在允许的范围。经计算 R_2 取 $2 \text{ k}\Omega$ 。

$$C_1 = \frac{389 \times 10^{-6}}{2 \times 10^3} \text{ F} = 0.19 \times 10^{-6} \text{ F} = 0.19 \mu\text{F}$$

在使用晶闸管的控制电路中,常要求晶闸管在电源电压为零或刚过零时触发晶闸管,来减少晶闸管在导通时对电源的影响。这种触发方式称为过零触发。过零触发需要过零检测电路,有些光电耦合器内部含有过零检测电路,如 MOC3061 双向晶闸管触发电路。图 12-13 是使用 MOC3061 双向晶闸管的过零触发电路。

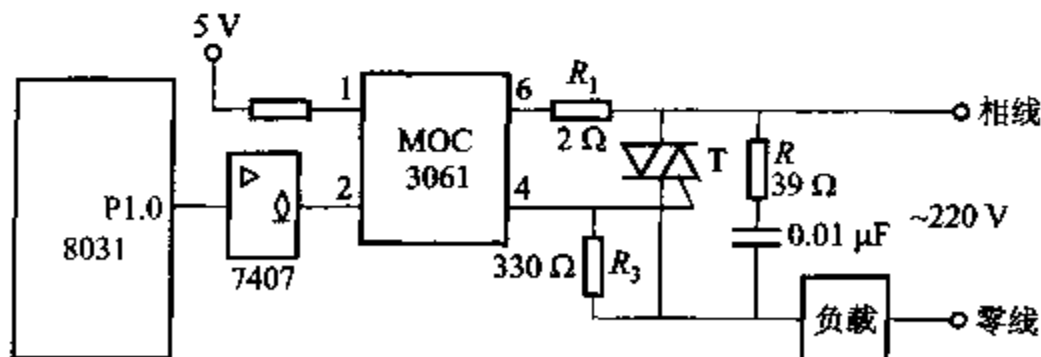


图 12-13 带过零触发的双向晶闸管触发电路

表 12-3 列出了 Motorola 公司 MOC3000 系列光耦合双向晶闸管驱动器的参数。

表 12-3 Motorola MOC3000 系列光耦合双向晶闸管驱动电路性能表

型 号	峰值夹断电压 最小值/V	LED 触发电流 ($V_{-1}=3\text{ V}$)		正向电压 典型值/V
		典型值/mA	最大值/mA	
MOC3009	250	15	30	1.2
MOC3010	250	8	15	1.2
MOC3011	250	5	10	1.2
MOC3012	250	—	5	1.2
MOC3020	400	15	30	1.2
MOC3021	400	8	15	1.2
MOC3022	400	—	10	1.2
MOC3023	400	—	5	1.2
MOC3030	250	—	30	1.3
MOC3031	250	—	15	1.3
MOC3032	250	—	10	1.3
MOC3040	400	—	30	1.3
MOC3041	400	—	15	1.3

12.2.4 MCS-51 与集成功率电子开关输出接口

集成功率电子开关是一种专为逻辑电路输出作接口而设计的直流功率电子开关器件。它可由 TTL、HTL、DTL、CMOS 等数字电路直接驱动,该器件开关速度快、工作频率高、无噪声、无触点,工作可靠、寿命长,目前在控制系统中常用来取代机械触点继电器,已越来越多地在单片机控制应用系统中作微电机控制、电磁阀驱动等。它特别适用于在那些需要抗潮湿、抗腐蚀和防爆场合中作大电流开关。如在那些机械触点继电器无法胜任的高频和高速系统中工作,更能体现其优越性。TWH8751 和 TWH8778 是应用最广泛的两种集成功率电子开关。它们都为标准的 TO-220 塑料封装,自带散热片,具有 5 条外引脚。下面以 TWH8751 为例介绍其性能和基本应用电路。

1. TWH8751 的引脚及其功能

图 12-14 是 TWH8751 的引脚图。

其中 2 引脚 V_{IN} 是输入引脚,1 引脚 S_T 为选通控制引脚,3 引脚为 V_- ,通常接地,4 引脚 V_O 为输出引脚,5 引脚 V_+ 为正电源引脚。

2. TWH8751 的性能特点

该器件设计有滞回特性, 抗干扰性能好, 而且其控制灵敏度高、工作频率高(可达 1.5 MHz)、开关特性好、边沿延迟仅毫微秒级, 控制功率较大, 内部开关功率管反向击穿电压为 100 V, 加上散热器, 可通过的灌电流可达 3 A。由于其输出管采用集电极开路方式, 所以可根据负载的要求选择合适的电源电压, 推荐的工作电压范围是 12~24 V。由于片内设有自我热保护减流电路, 当输出电流超过 2 A 时, 可自动使电流减至 1 A 左右。当断电或在输入端施加控制信号使输出级截止后, 开关电路可恢复 2 A 的输出负荷能力。

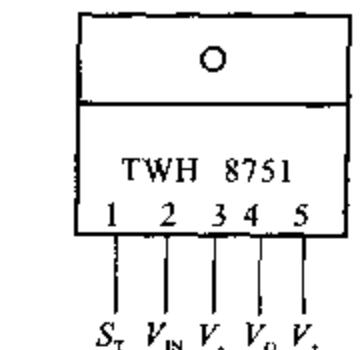


图 12-14 TWH8751 的引脚图

TWH8751 的开关动作延时为 $1\mu\text{s}$ 左右, 图 12-15 和图 12-16 分别给出输入-输出信号($V_{\text{IN}}-V_{\text{O}}$)和选通-输出信号($S_{\text{T}}-V_{\text{O}}$)的开关特性。它们都可在 200 kHz 频率下可靠地工作。

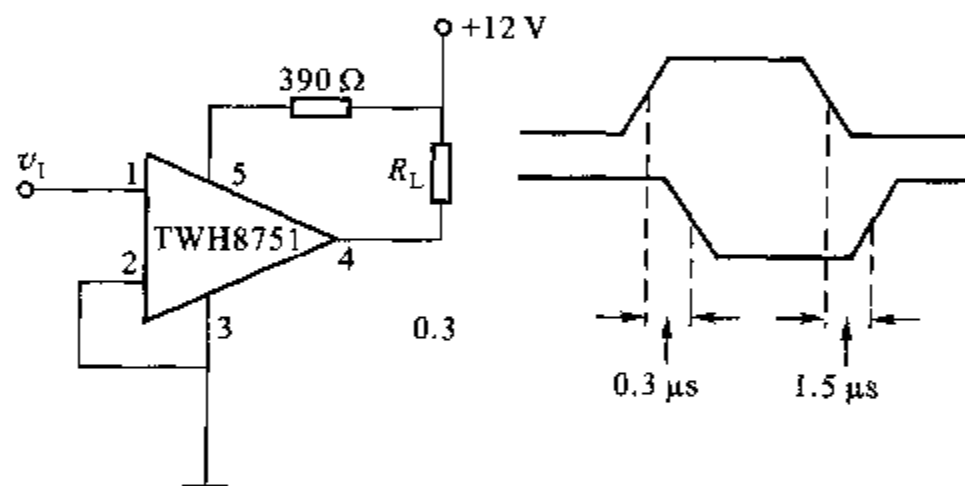


图 12-15 $V_{\text{IN}}-V_{\text{O}}$ 开关特性

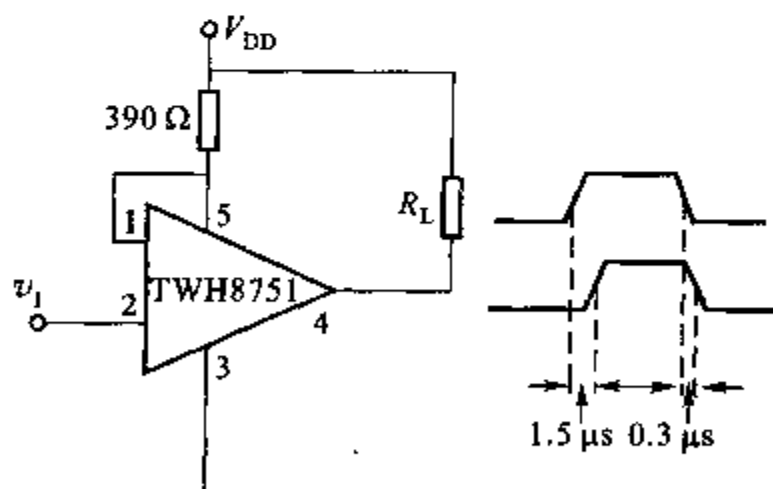


图 12-16 $S_{\text{T}}-V_{\text{O}}$ 开关特性

3. TWH8751 的使用和注意事项

该器件是逻辑开关, 而不是模拟开关, 输出不仅受输入的控制, 还受选通端的控制。当 S_{T} 选通引脚为高电平时, 不论 V_{IN} 引脚的电平是什么, 这时输出级的

达林顿输出管截止,输出引脚与地(3引脚)断开;当 S_1 为低电平时, $V_{IN}=1(>1.6\text{ V})$,输出级导通,输出引脚与地相接。

由于在片内电源与地之间设有1个 6.8 V 的稳压管,当工作电源电压超过 6.8 V 时,应加限流电阻 R_s , R_s 的值可按 $R_s=(V_{CC}-6.8\text{ V})/10\text{ mA}$ 来估算。由于输出开关功率管的反向击穿电压可达 100 V ,所以输出级可以不与 V_+ 端共电源,而根据实际需要加 $80\sim 100\text{ V}$ 的高压于负载上,但注意不能超过 100 V ,如满负荷运用一定要加散热器。

4. 典型应用

(1) 直流开关

TWH8751作直流开关用时,其接法见图12-17。

(2) 交流开关

TWH8751作交流开关用时,其接法见图12-18。

(3) 高压开关

TWH8751作高压开关用时,其接法见图12-19。

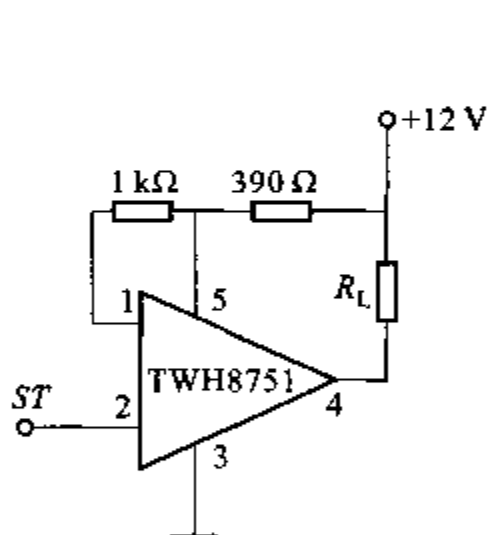


图 12-17 TWH8751 作直流开关

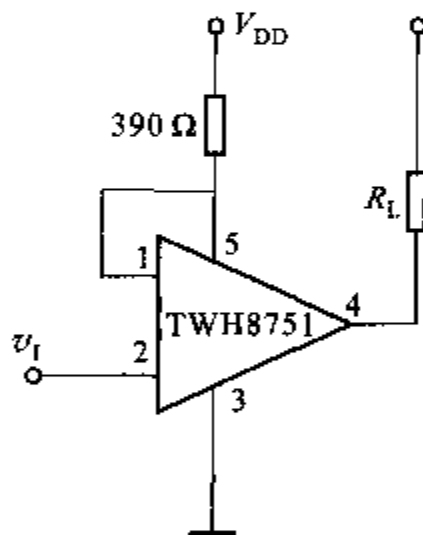


图 12-18 TWH8751 作交流开关

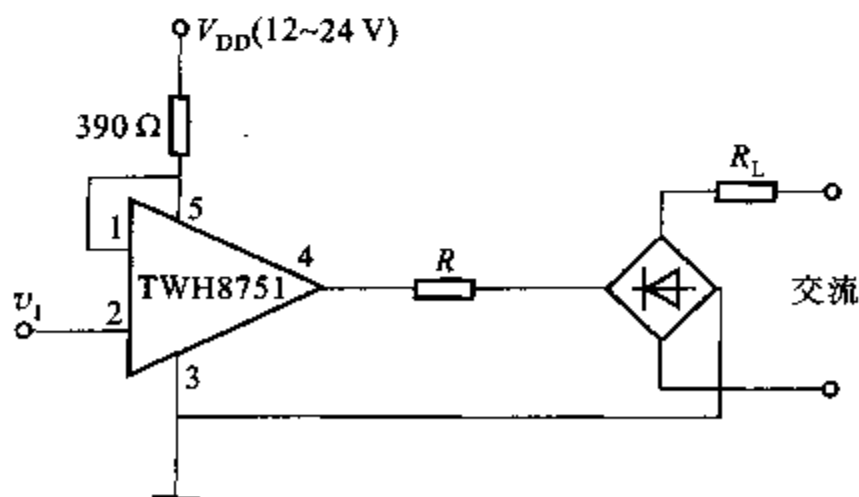


图 12-19 TWH8751 作高压开关

12.2.5 MCS-51 与固态继电器的接口

固态继电器(Solid State Relay, SSR)是近年发展起来的一种新型电子继电器,其输入控制电流小,用 TTL、HTL、CMOS 等集成电路或加简单的辅助电路就可直接驱动,因此适宜于在单片机测控系统中作为输出通道的控制元件;其输出利用晶体管或晶闸管驱动,无触点。与普通的电磁式继电器和磁力开关相比,具有无机械噪声、无抖动和回跳、开关速度快、体积小、重量轻、寿命长、工作可靠等特点,并且耐冲击、抗潮湿、抗腐蚀,因此在单片机测控等领域中,已逐渐取代传统的电磁式继电器和磁力开关作为开关量输出控制元件。

1. 固态继电器的主要特性

(1) 功率小:由于其输入端是采用的光电耦合器,其驱动电流仅需儿毫安便能可靠地控制,所以可以直接用 TTL、HTL、CMOS 等集成驱动电路控制。

(2) 高可靠性:由于其结构上无可动接触部件,且采用全塑密闭式封装,所以 SSR 开关时无抖动和回跳现象,无机械噪声,同时能耐潮、耐振、耐腐蚀;由于无触点火花,可用在有易燃易爆介质的场合。

(3) 低电磁噪声:交流型 SSR 在采用了过零触发技术后,电路具有零电压开启、零电流关断的特性,可使对外界和本系统的射频干扰减低到最低程度。

(4) 能承受的浪涌电流大:其数值可为 SSR 额定值的 6~10 倍。

(5) 对电源电压适应能力强:交流型 SSR 的负载电源电压可以在 30~220 V 范围内任选。

(6) 抗干扰能力强:由于输入与输出之间采用了光电隔离,割断了两者的电气联系,避免了输出功率负载电路对输入电路的影响。另外又在输出端附加了干扰抑制网络,有效地抑制了线路中 dv/di 和 di/dt 的影响。

2. 固态继电器的分类

固态继电器是一种四端器件,两端输入,两端输出。它们之间用光电耦合器隔离。

(1) 以负载电源类型分类:可分为直流型(DC-SSR)和交流型(AC-SSR)两种。直流型是用功率晶体管做开关器件;交流型则用双向晶闸管做开关器件,分别用来接通和断开直流或交流负载电源。

(2) 以开关触点形式分类:可分为常开式和常闭式。目前市场上以常开式为多。

(3) 以控制触发信号的形式分类:可分为过零型和非过零型。它们的区别在于负载交流电流导通的条件。非过零型在输入信号时,不管负载电源电压相位如何,负载端立即导通。而过零型必须在负载电源电压接近零且控制信号有

效时,输出端负载电源才导通。其关断条件是在输入端的控制电压撤销后,流过双向晶闸管的负载电流为零时,SSR 关断。

3. 固态继电器的典型应用

(1) 输入端的驱动

① 触点控制

最基本的驱动 触点控制,见图 12-20。

② TTL 驱动 SSR,见图 12-21。

③ CMOS 驱动 SSR,见图 12-22。

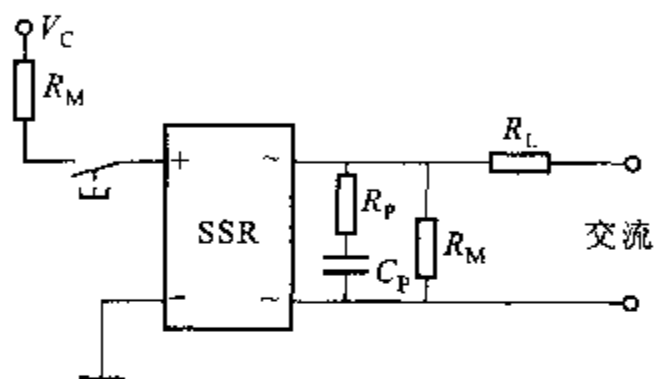


图 12-20 触点控制

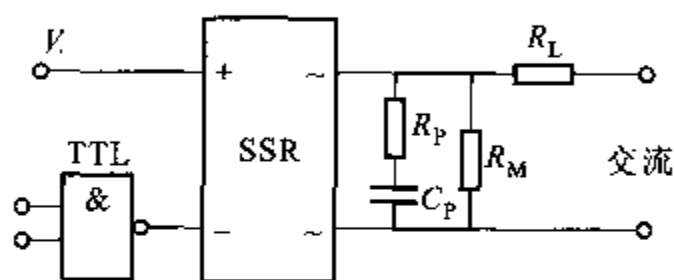


图 12-21 TTL 驱动 SSR

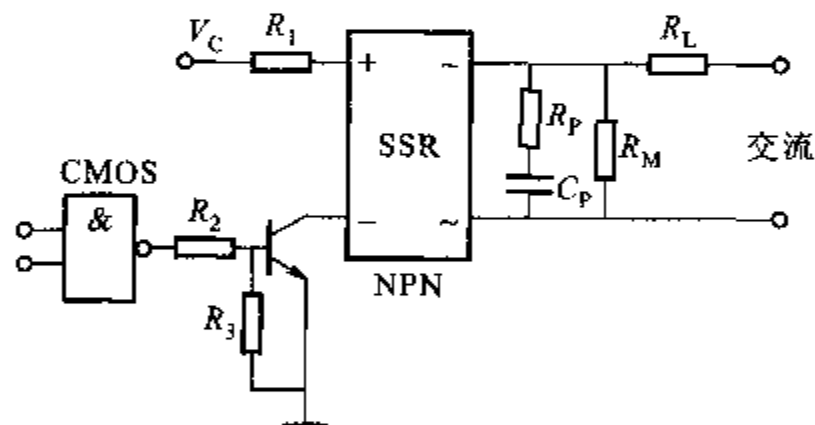


图 12-22 CMOS 驱动 SSR

(2) 输出端驱动负载

① DC-SSR 驱动大功率负载,见图 12-23。

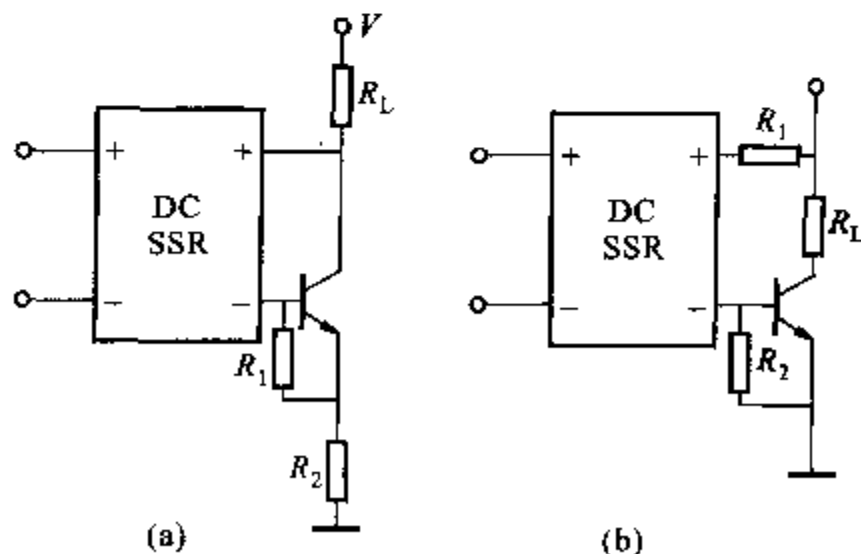


图 12-23 DC-SSR 驱动大功率负载

- ② DC-SSR 驱动大功率高压负载, 见图 12-24。
- ③ 用 SSR 控制单相交流电动机正反转电路, 见图 12-25。
- ④ 用 SSR 控制三相系统负载, 见图 12-26。
- ⑤ 用 SSR 控制大功率交流电动机, 见图 12-27。

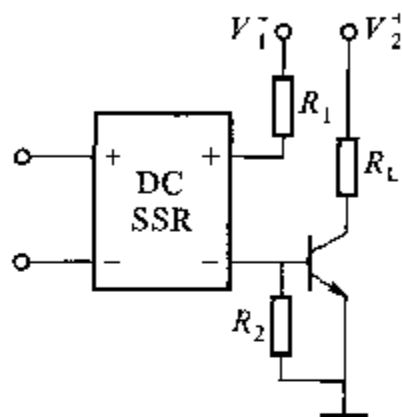


图 12-24 DC-SSR 驱动大功率高压负载

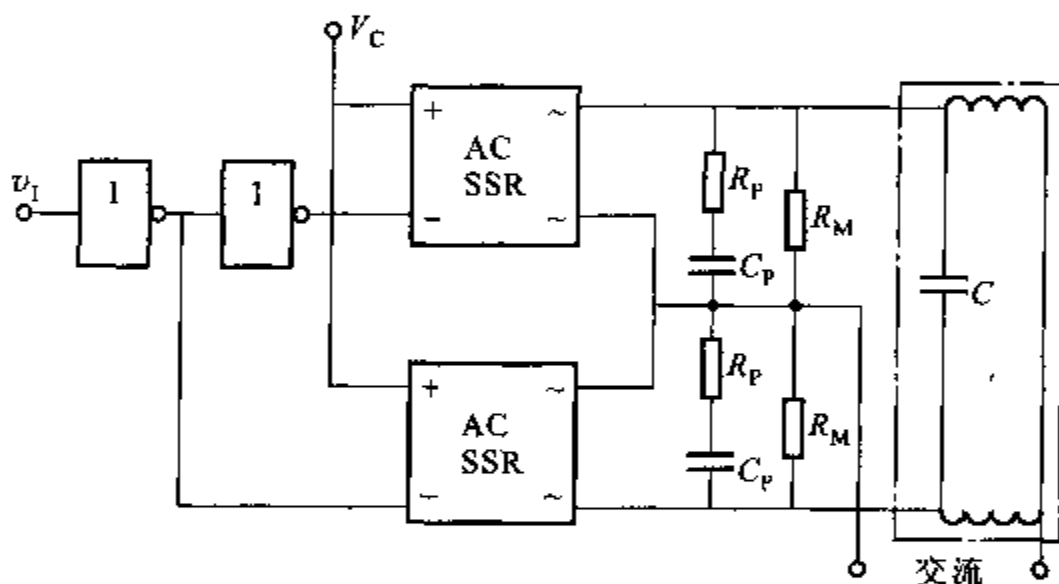


图 12-25 SSR 控制单相交流电动机正反转电路

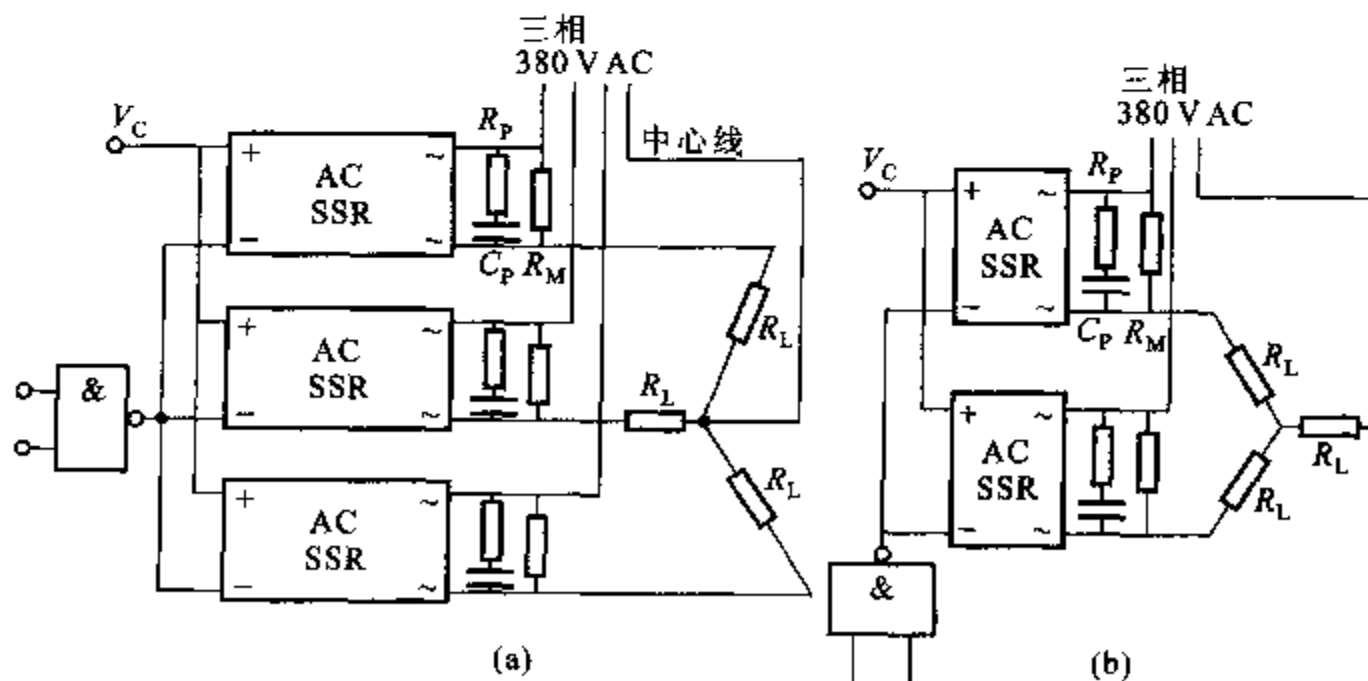


图 12-26 用 SSR 控制三相系统负载

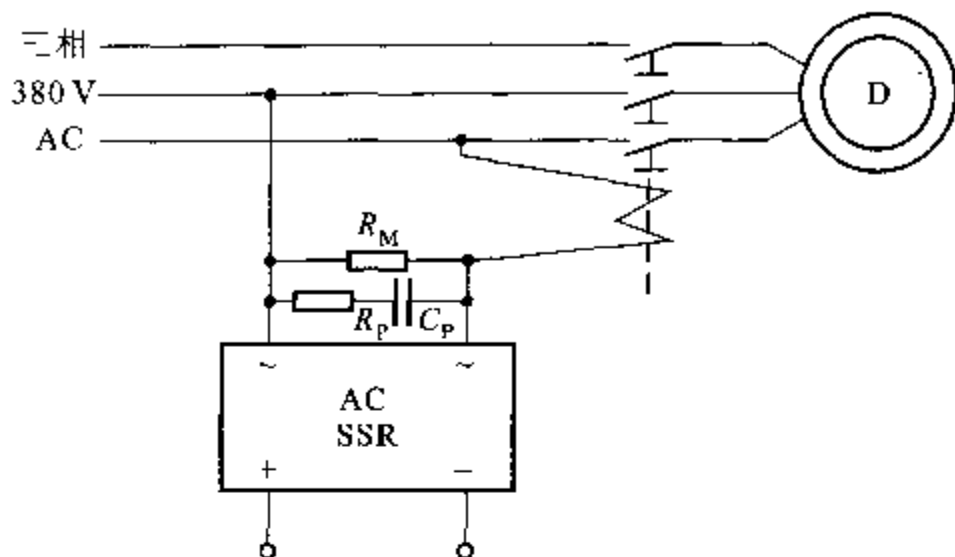


图 12-27 用 SSR 控制大功率交流电动机

4. 固态继电器使用注意事项

(1) 电子开关器件的通病是存在通态压降和断态漏电流。SSR 的通态压降一般小于 2 V, 断态漏电流通常为 5~10 mA。因此使用中要考虑这两项参数, 否则在控制小功率执行器时容易产生误动作。

(2) 固态继电器的电流容量负载能力随温度升高而下降, 其使用的温度范围不太宽($-40\sim+80\text{ }^{\circ}\text{C}$), 所以当使用温度较高时, 选用的 SSR 必须留有一定的余量。

(3) 固态继电器电压过载能力差, 当负载为感性时, 在 SSR 的输出端必须加接 R_M 压敏电阻, 其电压的选择可以取电源电压有效值的 1.6~1.9 倍。

(4) 输出端负载短路会造成 SSR 损坏, 应特别注意避免。对白炽灯、电炉等电阻类负载, 要考虑其“冷阻”特性会造成接通瞬间的浪涌电流, 有可能超过额定工作值, 所以要对电流容量的选择留有余地。为防止故障引起过流, 最简单的方法是采用快速熔断器, 要求熔断器的电压不低于线路工作电压, 其标称电流值(有效值)与固态继电器的额定电流值一致。

5. 常用的固态继电器

为便于读者选择固态继电器, 表 12-4 和表 12-5 分别列出部分常用直流固态继电器和交流固态继电器参数。

表 12-4 常用直流固态继电器参数表

型 号	输入电压/V	输入电流/A	输出电压/V	输出电流/A	厂 家
GZ1	4~28	4	10~50	1	苏州集成科技 实业有限公司
GZ3	4~28	4	10~50	3	
GZ5	4~28	4	10~50	5	
GZ10	4~28	4	10~50	10	

续表

型 号	输入电压/V	输入电流/A	输出电压/V	输出电流/A	厂 家
C603-01	3~14	3~15	30~180	1	北京半导体 器件十一厂
C603-02	3~14	3~15	30~180	2	
C603-03	3~14	3~15	30~180	3	
C603-04	3~14	3~15	30~180	4	
C603-05	3~14	3~15	30~180	5	
C603-10	3~14	3~15	30~180	10	
J83-03-2	4~7	6~18	50	0.3×2	
GTJ-0.5DP	6~30	3~30	24	0.5	上海电器电子 元件厂
GTJ-1DP	6~30	3~30	24	1	

表 12-5 常用交流固态继电器参数表

型 号	输入电压/V	输入电流/A	输出电压/V	输出电流/A	厂 家
GJN-1	3~28	4	30~250	1	
GJN-3	3~28	4	30~250	3	
GJN-5	3~28	4	30~250	5	
GJN-10	3~28	4	30~250	10	
GJ1	3~28	4	30~250	1	
GJ2	3~28	4	30~250	2	
GJ3	3~28	4	30~250	3	
GJ5	3~28	4	30~250	5	
GJ10	3~28	4	30~250	10	
GJ20	3~28	4	30~250	20	
GJ40	3~28	4	30~250	40	
GJH-1	3~28	4	30~250	1	
GJH-3	3~28	4	30~250	3	
CG3C-01	3~14	3~50	140/250 /400	1	北京半导体器 件十一厂
CG3C-02				2	
CG3C-03				3	
CG3C-04				4	
CG3C-05				5	
CG3A-2				20	
GTJ-1AP	3~30	30	30~220	1	
GTJ-2.5AP	3~30	30	30~220	2.5	
SP110	2~6	3~10	350	1	

12.2.6 低压开关量信号输出技术

对于低压情况下开关量控制输出,可采用晶体管、OC 门或运放等方式,如驱动低压电磁闸、指示灯、直流电机等,如图 12-28 所示。需注意的是,在使用 OC 门时,由于其为集电极开路输出,在其输出为“高”电平状态时,实质只是一种高阻状态,必须外接上拉电阻,此时的输出驱动电流主要由 V_C 提供,只能直流驱动并且 OC 门的驱动电流一般不大,在几十毫安量级。如果被驱动设备所需驱动电流较大,则可采用三极管输出方式,如图 12-29 所示。

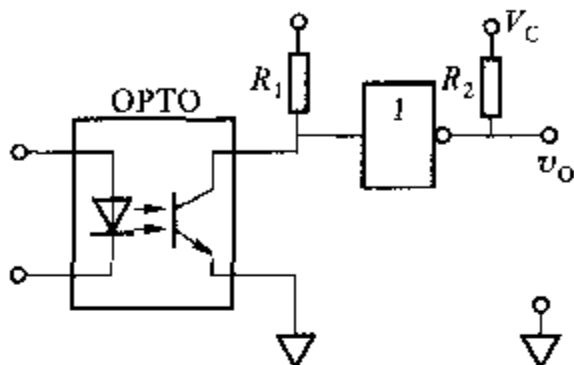


图 12-28 低压开关量输出

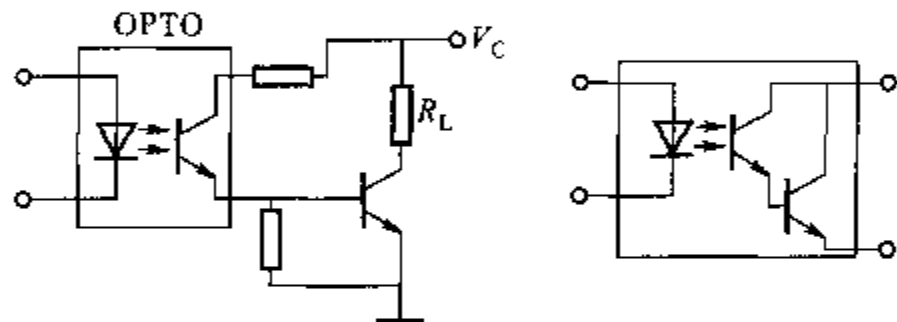


图 12-29 三极管输出驱动

思考题及习题

1. MCS-51 的 4 个并行双向口 P0~P3 的驱动能力各为多少? 要想获得较大的驱动能力最好采用低电平输出还是使用高电平输出?
2. 讨论 MCS-51 的功率接口的意义是什么?
3. 常用的开关型驱动器件都有哪些? 请列举。
4. 列举在单片机应用系统中常用的电子开关的名称, 电子开关的通病是什么?
5. 集成功率电子开关与机械触点继电器相比具有哪些优越性?
6. 固态继电器分为哪几类? 具有哪些优点?
7. 使用固态继电器的注意事项都有哪些?

第 13 章 MCS-51 的串行通信技术及其他扩展接口

前面已分类介绍了各种接口设计,本章主要介绍 MCS-51 单片机的串行通信接口技术以及在单片机应用系统设计中,经常用到的其他一些实用扩展接口。

13.1 MCS-51 单片机的串行通信接口技术

MCS-51 内部的串行口,大大扩展了 MCS-51 的应用范围。利用串行口可以实现 MCS-51 之间的点对点的串行通信、多机通信以及 MCS-51 与 PC 机间的单机或多机通信。

MCS-51 串行口的输入、输出均为 TTL 电平。这种以 TTL 电平串行传输数据的方式,抗干扰性能差,传输距离短。为了提高串行通信的可靠性,增大串行通信的距离,一般都采用标准串行接口,如 RS-232C、RS-422A、RS-485 等标准来实现串行通信。

RS-232C 是由美国电子工业协会(EIA)于 1962 年制定的标准,是在异步串行通信中应用最广的标准串行接口(RS-232C 是 1969 年修订的版本,C 表示此标准已修改至第 3 版)。RS 是“Recommended Standard”的缩写,意为推荐标准。至 1997 年已修订至 RS-232F,由于改动不大,人们还是习惯地称此类接口为“RS-232C”。RS-232C 适用于短距离或带调制解调器的串行通信场合。为了提高串行数据传输率和通信距离以及抗干扰能力,EIA 又公布了 RS-422, RS-423 和 RS-485 串行总线接口标准。

本节首先介绍上述的几种常见的标准串行通信接口,然后讨论单片机的双机、多机以及单片机与 PC 机之间的串行通信的硬件接口技术,至于软件编程,限于篇幅,仅介绍编程的基本思想,具体程序请参阅参考书目[5]。

13.1.1 各种标准串行通信接口

1. RS-232C 接口

RS-232C 是异步串行通信中应用最广的标准串行接口,它定义了数据终端设备(DTE)和数据通信设备(DCE)之间的串行接口标准,主要包括了有关串行数据传输的电气和机械方面的规定。为了更好地理解 RS-232C 标准以及 DTE 和 DCE,首先介绍一个实际的串行通信的例子。图 13-1 是两台计算机利用调制解调器(MODEM)、电话线进行远距离串行通信的示意图,其中 DTE 为计算机,而 DCE 的典型代表是调制解调器(MODEM)。

两台计算机远距离通信时,由于以 TTL 电平传输数据的方式,抗干扰性能差,传输距离短,通常要借用现成的公用电话网,但是电话网是为 300~3400 Hz 的音频模拟信号设计的,其频带有限,不能进行二进制数字量的传输。因此,在发送时需要利用调制器(Modulator)把数字信号转换成模拟信号,然后送到通信线路上去,再由解调器(Demodulator)把从通信线路收到的模拟信号转换成数字信号。由于通信是双向的,调制器和解调器合并在一个装置中,称为 MODEM。由图 13-1 可以清楚地看出,RS-232C 定义的是 DTE 和 DCE 之间的串行通信的接口标准。

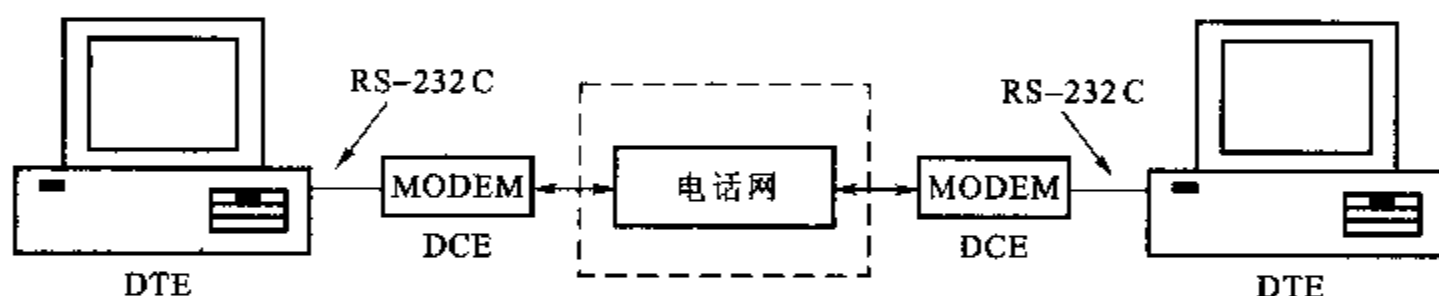


图 13-1 利用调制解调器通信的示意图

目前,PC 机都配有标准的 RS-232C 接口,RS-232C 标准规定了 25 针连接器,但在实际应用中并不一定用到 RS-232C 的全部信号线,所以,PC 机配置的都是 9 针“D”型连接器(插座)。图 13-2 为 RS-232C 的“D”型 9 针插头的引脚定义。

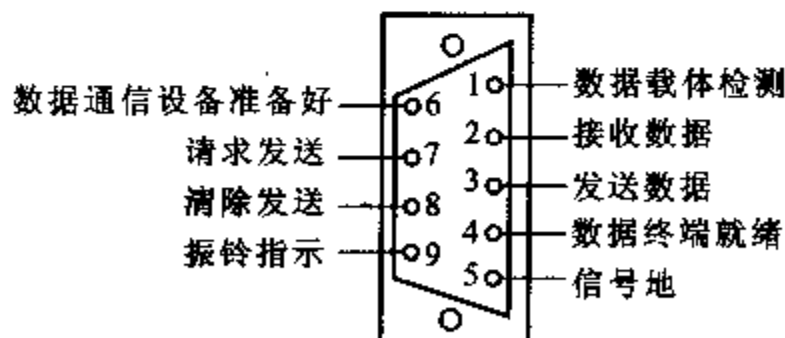


图 13-2 “D”型 9 针插头引脚定义

9 个引脚的功能见表 13-1。

表 13-1 PC 机的 RS-232C 接口信号

引脚号	符号	方向	功能
1	DCD	输入	数据载体检测
2	TXD	输出	发送数据
3	RXD	输入	接收数据
4	DTR	输出	数据终端准备好
5	GND		信号地
6	DSR	输入	数据通信设备准备好
7	RTS	输出	请求发送
8	CTS	输入	清除发送
9	RI	输入	振铃指示

(1) 电气特性

RS-232C 上传送的数字量采用负逻辑,且与地对称。

逻辑 1: $-3 \sim -15$ V;

逻辑 0: $+3 \sim +15$ V。

RS-232C 标准的信号传输的最大电缆长度为 30 m,最高数据传输速率为 20 Kb/s。

(2) 电平转换

由于 TTL 电平和 RS-232C 电平互不兼容,所以两者对接时,必须进行电平转换。RS-232C 与 TTL 电平转换最常用的芯片是 MC1488、MC1489 和 MAX232 等,各厂商生产的此类芯片虽然不同,但原理相似。以美国 MAXIM 公司的产品 MAX232 为例,它是 RS-232C 双工发送器/接收器接口电路芯片,其外部引脚如图 13-3,内部结构及外部元件如图 13-4 所示。由于芯片内部有自升压的电平倍增电路,将 $+5$ V 转换成 $-10 \sim +10$ V,满足 RS-232C 标准对逻辑 1 和逻辑 0 的电平要求。工作时仅需单一的 $+5$ V 电源。其片内有 2 个发送器,2 个接收器,有 TTL 信号输入/RS-232C 输出的功能,也有 RS-232C 输入/TTL 输出的功能。该芯片与 TTL/CMOS 电平兼容,使用比较方便。使用 MAX232 实现 TTL/RS-232C 之间的电平转换电路如图 13-5 所示。

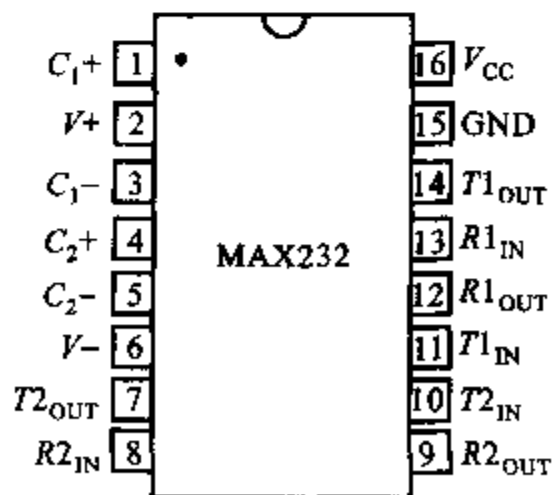


图 13-3 MAX232 的引脚

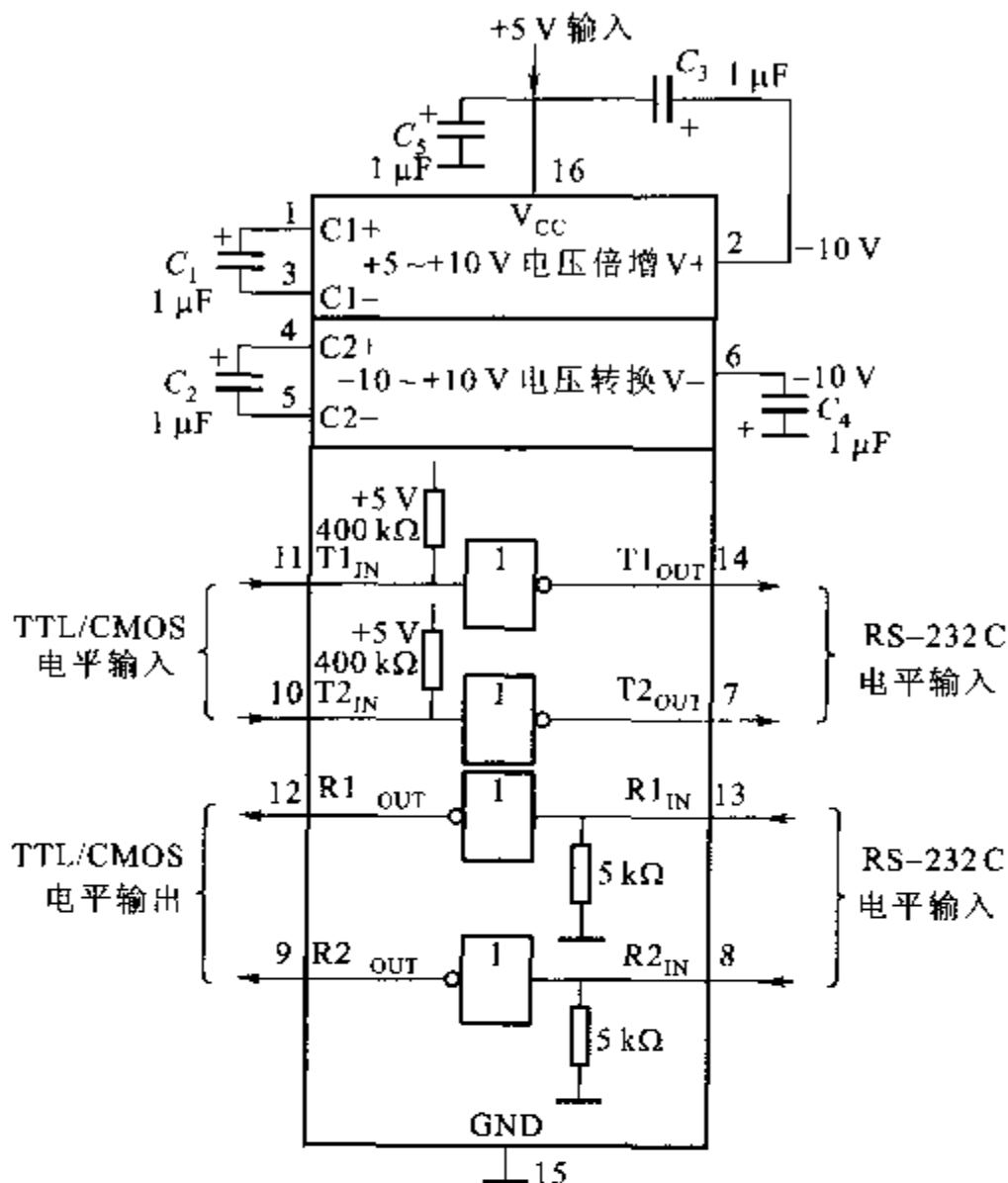


图 13-4 MAX232 的内部结构及外部元件

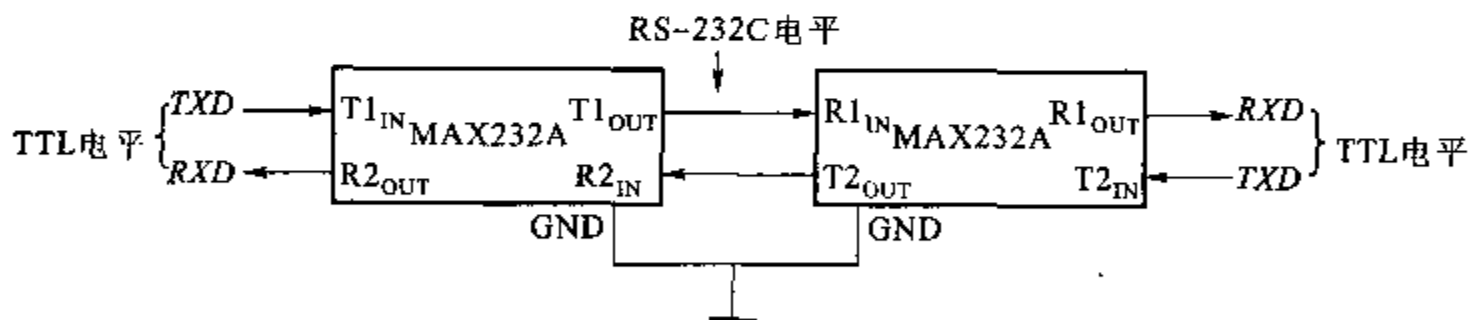


图 13-5 RS-232C 接口的电平转换电路

2. RS-422A 接口

RS-232C 虽然应用很广泛,但其推出较早,在现代网络通信中已暴露出明显的缺点:传输速率低、通信距离短、接口处信号容易产生串扰等。鉴于此,EIA 又制定了 RS-422A 标准。RS-232C 既是一种电气标准,又是一种物理接口功能标准,而 RS-422A 仅仅是一种电气标准。PC 机不带 RS-422A 接口,因此要使用 RS-232C/RS-422A 转换器,把 RS-232C 接口转换成 RS-422A 接口。

(1) 电气特性

RS-422A 与 RS-232C 的主要区别是,收发双方的信号地不再共地,RS-422A 标准规定平衡驱动和差分接收的方法。每个方向用于数据传输的是两条平衡导线,这相当于两个单端驱动器。输入同一个信号时,其中一个驱动器的输出永远是另一个驱动器的反相信号。于是两条线上传输的信号电平,当一个表示逻辑 1 时,另一条一定为逻辑 0。若传输过程中,信号中混入了干扰和噪声(以共模形式出现),由于差分接收器的作用,就能识别有用信号并正确接收传输的信息,并使干扰和噪声相互抵消。

因此,RS-422A 能在长距离、高速率下传输数据。它的最大传输速率为 10 Mb/s,在此速率下,电缆允许长度为 12 m,如果采用较低传输速率,最大传输距离可达 1 200 m。

RS-422A 电路由发送器、平衡连接电缆、电缆终端负载、接收器 4 部分组成。在电路规定只许有 1 个发送器,可以有多个接收器,因此,通常采用点对点通信方式。该标准允许驱动器输出为 $\pm 2 \sim \pm 6$ V,差分接收器可以检测的输入信号电平可低到 200 mV。

(2) 电平转换

TTL 电平转换成 RS-422A 电平的常用芯片有 SN75174、MC3487 等。器件特性为:无负载输出电压 ≤ 6 V,加负载输出电压 ≥ 2 V,断电下输出阻抗 $\geq 4 \Omega$,短路输出电流 ≤ 150 mA。

RS-422A 电平转换成 TTL 电平的常用芯片为:SN75175、MC3486 等。器件特性为:输入阻抗 $\geq 4 \Omega$,阈值为 $-0.2 \sim +0.2$ V,最大输入电压为 ± 12 V。

SN75174、SN75175 分别是具有三态输出的单片四差分驱动器和接收器,其设计符合 EIA 标准 RS-422A,采用 ± 5 V 电源供电。图 13-6,图 13-7 分别给出了电平转换芯片 SN75174、SN75175 内部结构及引脚图。

表 13-2,表 13-3 为对应的 SN75174、SN75175 芯片的功能表。

表 13-2 SN75174 功能表(每个驱动器)

输入 A	使能	输出	
		Y	Z
H	H	H	L
L	H	L	H
X	L	三态	三态

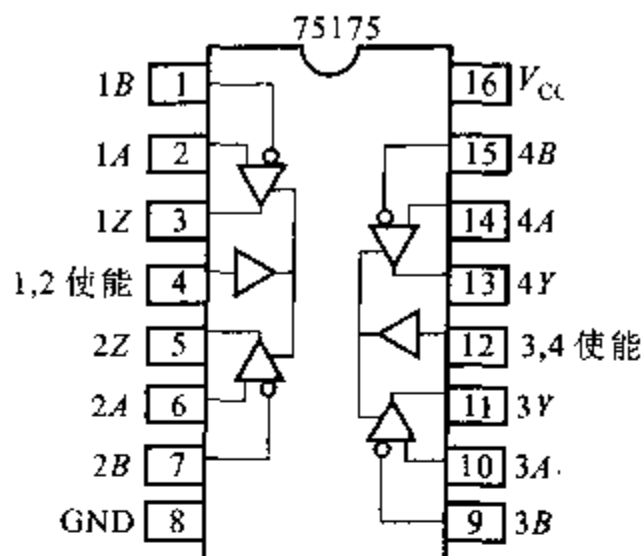
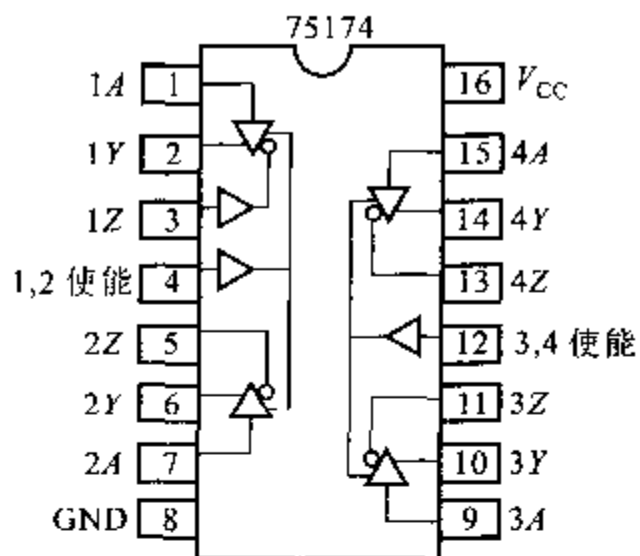


图 13-6 RS-422A 电平转换芯片 SN75174 图 13-7 RS-422A 电平转换芯片 SN75175

表 13-3 SN75175 功能表 (每个接收器)

差分输入 V_{in}	使 能	输 出 Y
$V_{in} > 0.2 \text{ V}$	H	H
$-0.2 \text{ V} < V_{in} < 0.2 \text{ V}$	H	×
$V_{in} < -0.2 \text{ V}$	H	L
×	L	三态

TTL 电平与 RS-422A 电平转换电路如图 13-8 所示。

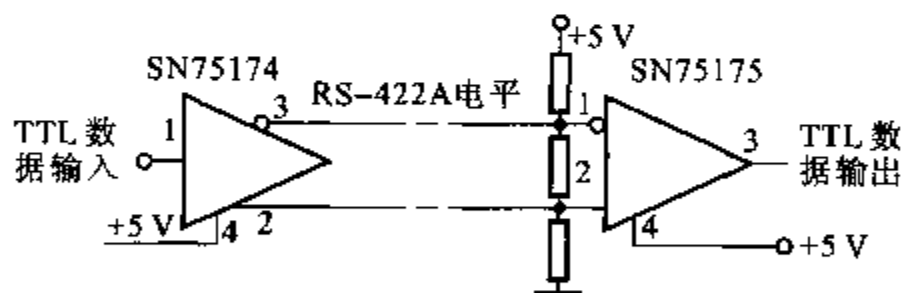


图 13-8 RS-422A 接口的电平转换电路

3. RS-485 接口

RS-485 是 RS-422A 的变型,它与 RS-422A 的区别在于:RS-422A 为全双工,采用 2 对平衡差分信号线;而 RS-485 为半双工,采用 1 对平衡差分信号线。RS-485 对于多站互连是十分方便的。RS-485 标准允许最多并联 32 台驱动器和 32 台接收器。

(1) 电气特性

RS-485 的信号传输采用两线间的电压来表示逻辑 1 和逻辑 0,由于发方需要 2 根传输线,收方也需要 2 根传输线。数据采用差分传输,所以干扰抑制性好,又因为阻抗低,无接地问题,所以传输距离可达 1 200 m,传输速率可

达10 Mb/s。

总线两端接匹配电阻(1000 Ω 左右),驱动器负载为54 Ω 。驱动器输出电平在-1.5 V 以下时为逻辑1,在+1.5 V 以上时为逻辑0。接收器输入电平在-0.2 V 以下时为逻辑1,在+0.2 V 以上为逻辑0。

普通的PC机一般不带RS-485接口,因此要使用RS-232C/RS-485转换器。对于单片机可以通过芯片MAX485来完成TTL/RS-485的电平转换。

(2) 电平转换

在RS-422A标准中所用的驱动器和接收器芯片,在RS-485中均可以使用。除了RS-422A电平转换中所列举的驱动器和接收器外,还有收发器SN75176芯片,该芯片集成了一差分驱动器和一差分接收器,如图13-9所示。SN75176的功能见表13-4。RS-485点对点远程通信电路如图13-10所示。在图13-10中,某一时刻两个站只有一个站可以发送数据,而另一个站只能接收数据,因此,其发送电路必须由使能端加以控制。

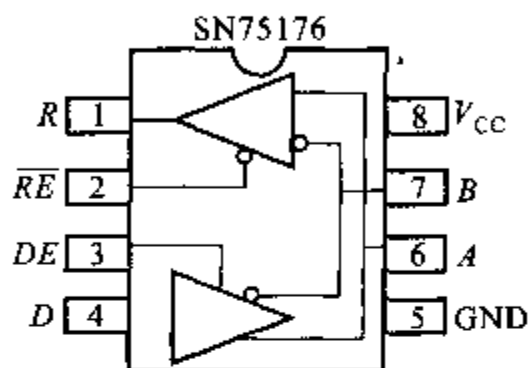


图 13-9 RS-485 电平转换
芯片 SN75176

表 13-4 SN75176 功能表

驱 动 器			
输 入 D	使 能 DE	输 出	
		A	B
H	H	H	L
L	H	L	H
×	L	三态	三态
接 收 器			
差分输入 V_{in}		使能 \overline{RE}	输出 R
$V_{in} \geq 0.2 \text{ V}$		L	H
$-0.2 \text{ V} < V_{in} < +0.2 \text{ V}$		L	×
$V_{in} \leq -0.2 \text{ V}$		L	L
×		H	三态

4. 20 mA 电流环串行接口

20 mA 电流环串行接口也是目前串行通信中广泛使用的一种接口电路,但

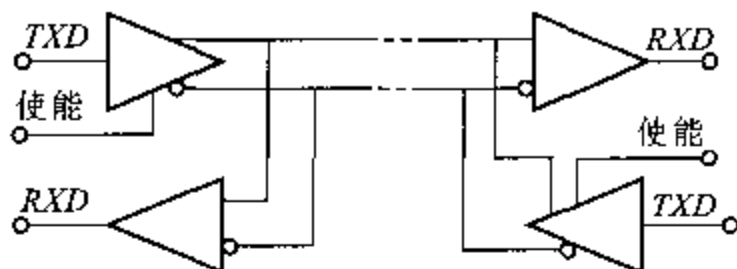


图 13-10 RS-485 点对点远程通信电路

未形成正式标准。这种接口要比 RS-232C 接口简单的多,它只有 4 根线:发送正、发送负、接收正和接收负 4 根线组成 1 个输入电流回路、1 个输出电流回路。当发送数据时,根据数据的逻辑 1、0,有规律的使回路形成通、断状态,即环路中无电流表示逻辑 0,有 20 mA 电流表示逻辑 1。20 mA 电流环工作原理如图 13-11 所示。

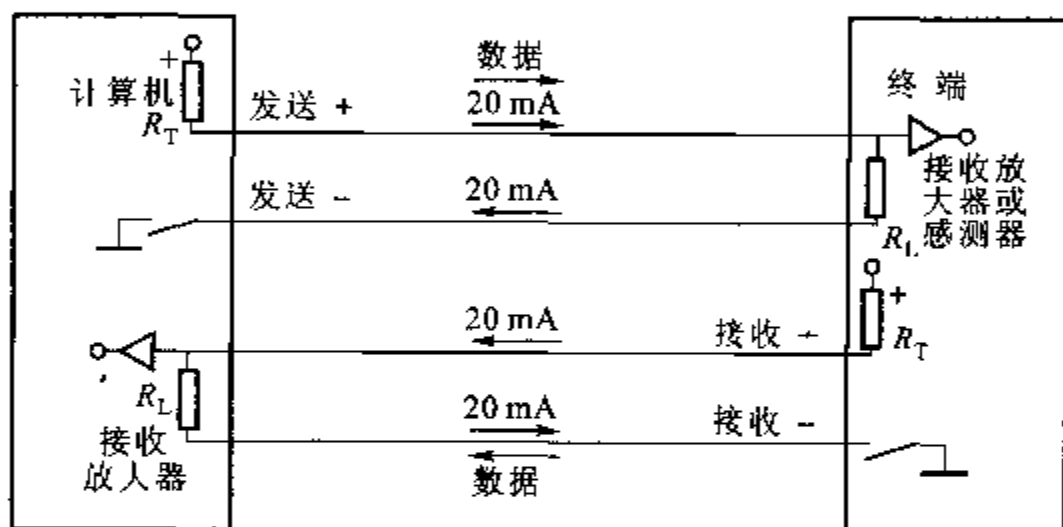


图 13-11 20 mA 电流环原理图

电流环路串行通信接口的最大优点是低阻传输线对电气噪声不敏感,且易实现光电隔离。因此,在长距离传送时,要比 RS-232C 优越的多。电流环在低速率数据传输时,传输距离可达 1 000 m。

由于 20 mA 电流环是 1 种异步串行接口标准,所以在每次发送数据时必须以无电流的起始作为每一个字符的起始位,接收端检测到起始位时便开始接收字符数据。

图 13-12 是 1 个由集成芯片构成的 20 mA 电流环接口线路图。

5. 各种串行接口性能比较

现将 RS-232C、RS-422A、RS-485、20 mA 电流环各串行接口性能列在表 13-5 中,以便比较。

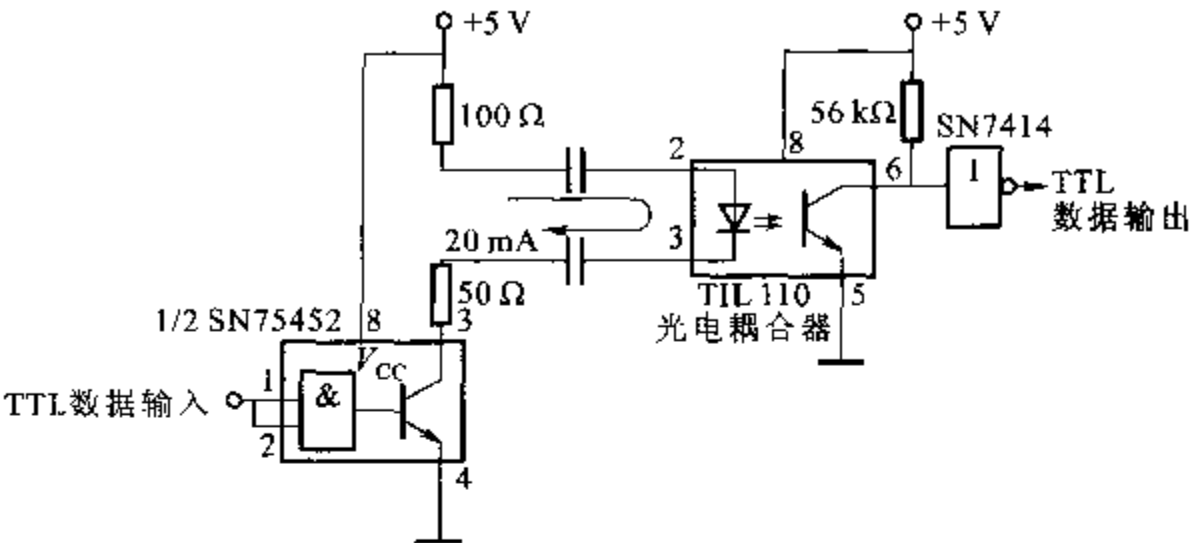


图 13-12 集成芯片构成的 20 mA 电流环接口电路

表 13-5 各种串行接口性能比较

接口 性能	RS-232C	RS-422A	RS-485	20 mA 电流环
功 能	双向、全双工	双向、全双工	双向、半双工	双向、全双工
传输方式	单端	差分	差分	20 mA 电流通断
逻辑 0 电平	3~15 V	2~6 V	1.5~6 V	0 mA
逻辑 1 电平	-3~-15 V	-2~-6 V	-1.5~-6 V	20 mA
最大速率	20 Kb/s	10 Mb/s	10 Mb/s	/
最大距离	30 m	1 200 m	1 200 m	1 000 m
驱动器加载输出电压	±5~±15 V	±2 V	±1.5 V	/
接收器输入灵敏度	±3 V	±0.2 V	±0.2 V	/
接收器输入阻抗	3~7 kΩ	>4 kΩ	>7 kΩ	/
组态方式	点对点	1 台驱动器 10 台接收器	32 台驱动器 32 台接收器	点对点
抗干扰能力	弱	强	强	强
传输介质	扁平或多芯 电缆	2 对双绞线	1 对双绞线	扁平或多芯 电缆
常用驱动器芯片	MAX232 MC1488	SN75174 MC3487	SN75174, MC3487, SN75176, MAX485	/
常用接收芯片	MAX232 MC1489	SN75175 MC3486	SN75174, MC3486, SN75176, MAX485	/

13.1.2 MCS-51 单片机双机串行通信接口

1. 双机通信硬件接口设计

根据 MCS-51 的双机通信距离、抗干扰性能的要求,可选择 TTL 电平传输,或选择 RS-232C、RS-422A、RS-485 串行接口进行串行数据传输。

(1) TTL 电平通信接口

如果两个 MCS-51 单片机相距在几米之内,它们的串行口可直接相连,从而直接用 TTL 电平传输方法来实现双机通信。如图 13-13 所示。

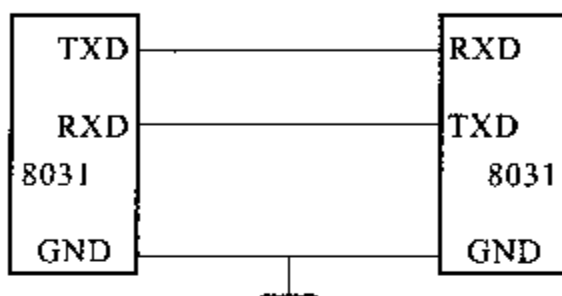


图 13-13 用 TTL 电平传输方法实现双机通信的电路

(2) RS-232C 双机通信接口

如果双机通信距离在 30 m 之内,可利用 RS-232C 标准接口实现点对点的双机通信,接口电路如图 13-14 所示。

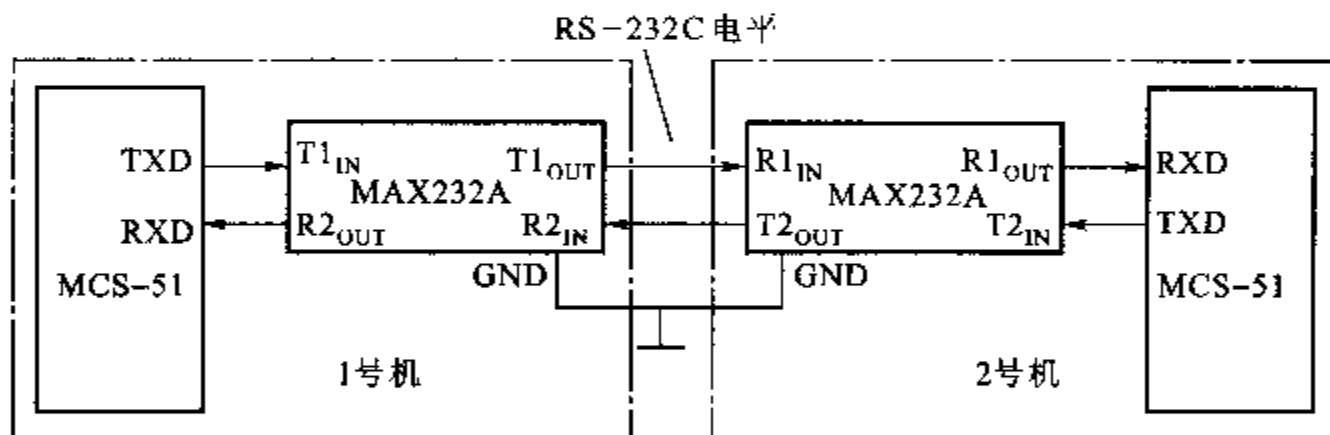


图 13-14 RS-232C 双机通信接口电路

(3) RS-422A 双机通信接口

为了增加通信距离,减小通道及电源干扰,可以在通信线路上采用光电隔离的方法,利用 RS-422A 标准进行双机通信,接口电路如图 13-15 所示。

在图 13-15 中,每个通道的接收端都接有 3 个电阻 R_1 、 R_2 、 R_3 ,其中 R_1 为传输线的匹配电阻,取值范围在 $50\Omega \sim 1\text{k}\Omega$ 之间,其他 2 个电阻是为了解决第一个数据的误码而设置的匹配电阻。为了起到隔离、抗干扰的作用,图 13-15 中必须使用 2 组独立的电源。

(4) RS-485 双机通信接口

RS-422A 双机通信需四芯传输线,这对工业现场的长距离通信是很不经济的,故在工业现场,通常采用双绞线传输的 RS-485 串行通信接口,这种接口很容易实现多机通信。图 13-16 给出了其 RS-485 双机通信接口电路。

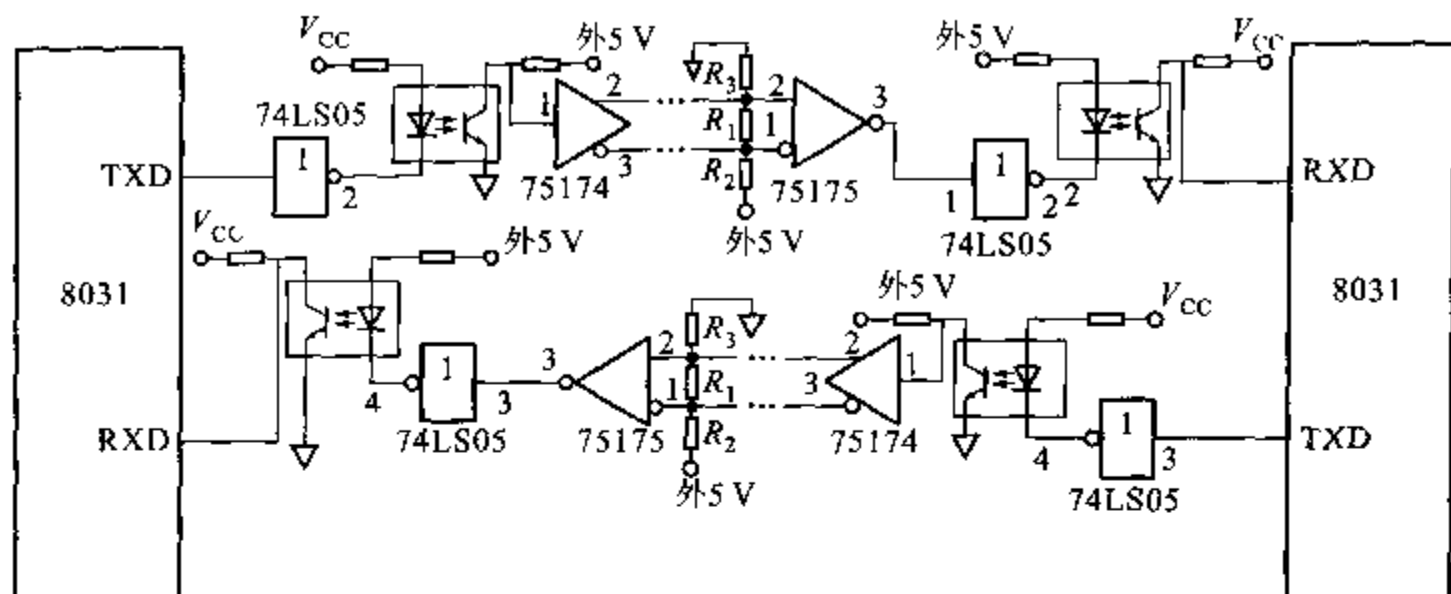


图 13-15 RS-422A 双机通信接口电路

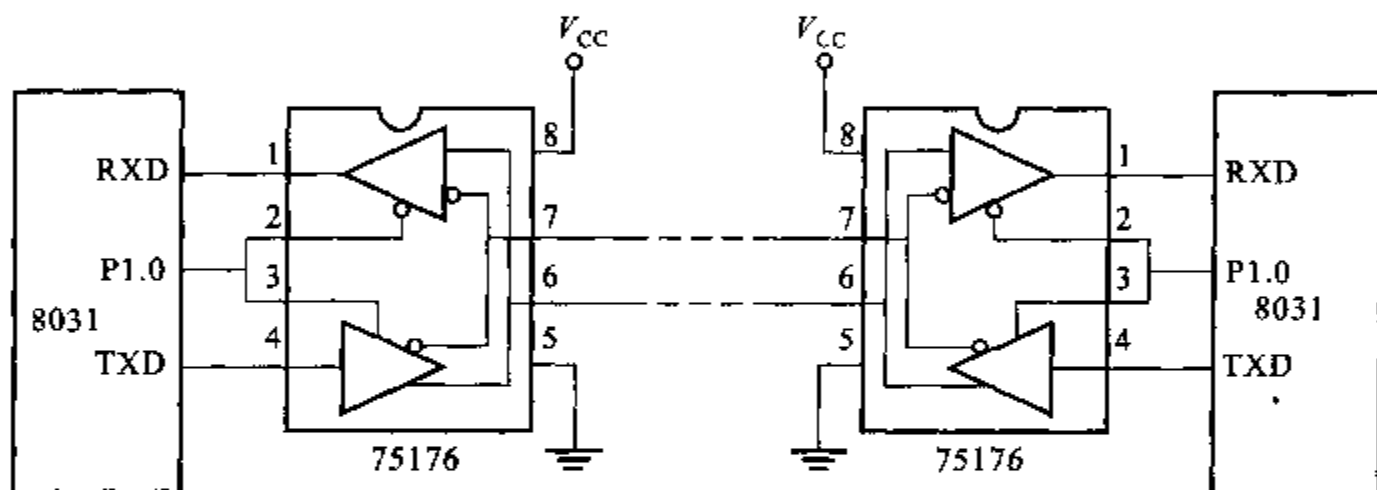


图 13-16 RS-485 双机通信接口电路

由图 13-16 可知：RS-485 以双向、半双工的方式实现了双机通信。在 8031 系统发送或接收数据前，应先将 75176 的发送门或接收门打开，当 $P1.0=1$ 时，发送门打开，接收门关闭；当 $P1.0=0$ 时接收门打开，发送门关闭。

2. 双机通信软件编程

除 RS-485 串行通信外，TTL、RS-232C、RS-422A 双机通信的软件设计方法是一样的，下面仅对设计思想加以说明。

(1) 通信协议

为确保通信成功，通信双方必须在软件上有一系列的约定，通常称为通信协议或软件协议。例如，某应用系统规定双机异步通信的协议如下：

- ① 通信的甲、乙双方均可发送和接收。
- ② 通信波特率为 2400 b/s，定时器 T1 工作在模式 2，对于 6 MHz 时钟频率，计数常数为 F3H，SMOD=1。
- ③ 双方均采用串行口方式 3。
- ④ 欲发送或接收的数据块首地址存放在 64H、63H，其中 64H 为首地址高

字节暂存单元,63H 为首地址低字节暂存单元;数据块长度存放在 62H、61H 中,其中 62H 为数据长度高字节暂存单元,61H 为数据长度低字节暂存单元。

⑤ 发送或接收的数据帧格式为:

双字节地址	双字节数据个数 n	数据 1	...	数据 n	累加校验和
-------	-------------	------	-----	--------	-------

双字节地址:低地址字节在前,高地址字节在后;

双字节数据个数:数据个数的低字节在前,高字节在后;

数据 1~数据 n :所通信的 n B 数据;

累加校验和:为双字节地址,双字节数据个数 n ,数据 1,...,数据 n 这 $(n+4)$ B 的算术累加和,用作校验。

⑥ 接收方接收到校验和后,判断接收到的数据是否正确。若接收正确,向发送方回发 OFH 信号,否则,回发 FOH 信号。

⑦ 甲、乙双方均采用串行口中断方式接收和发送数据。

(2) 程序编写

程序的编写工作要在串行通信的软件协议制订之后,才能进行。本书的第 7 章的例 7-4、例 7-5,就是 1 个双机串行通信的参考程序,读者一定要在弄清通信协议之后,再去阅读程序。

13.1.3 MCS-51 单片机多机串行通信接口

利用串行口实现多机通信的工作原理,已在本书的第 7 章中作过介绍。下面首先介绍多机通信的接口设计。

1. TTL 电平多机串行通信

当一台主机与多台从机之间距离较近时,可直接采用 TTL 电平进行多机通信,多机通信的连接方式如图 13-17 所示。

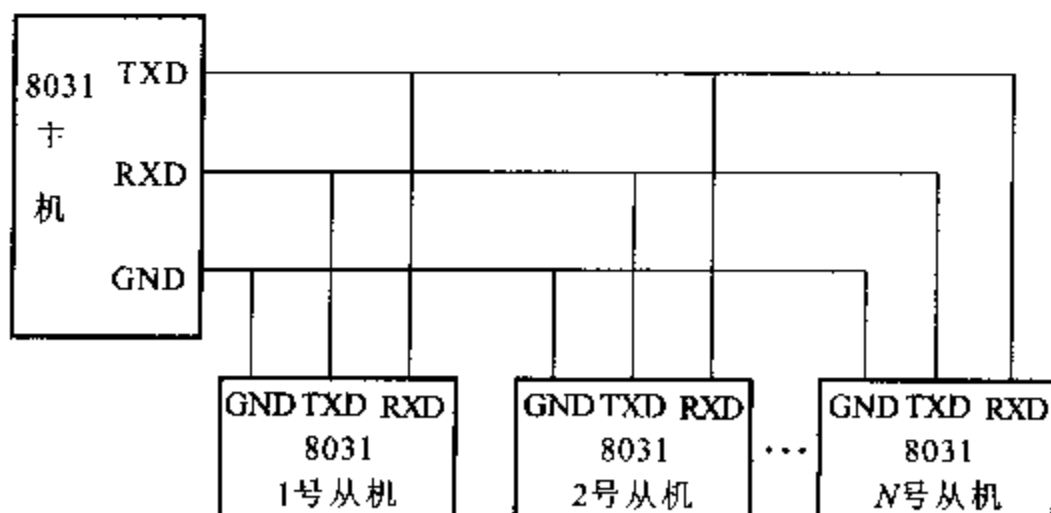


图 13-17 多机全双工通信连接方式

由于 8031 单片机 P3 口可带 4 个 LSTTL, 故在图 13-17 中, N 的取值范围应为 $N \leq 4$ 。如果 $N > 4$, 则 P3 口应加驱动电路。

2. 20 mA 电流环多机串行通信接口

用 TTL 电平进行多机通信时, 有效通信距离约几米左右, 这在实际中往往不能满足要求。采用 20 mA 电流环进行多机通信, 不仅提高了抗干扰能力, 而且可实现远距离通信。

20 mA 电流环串行多机通信原理电路如图 13-18 所示。

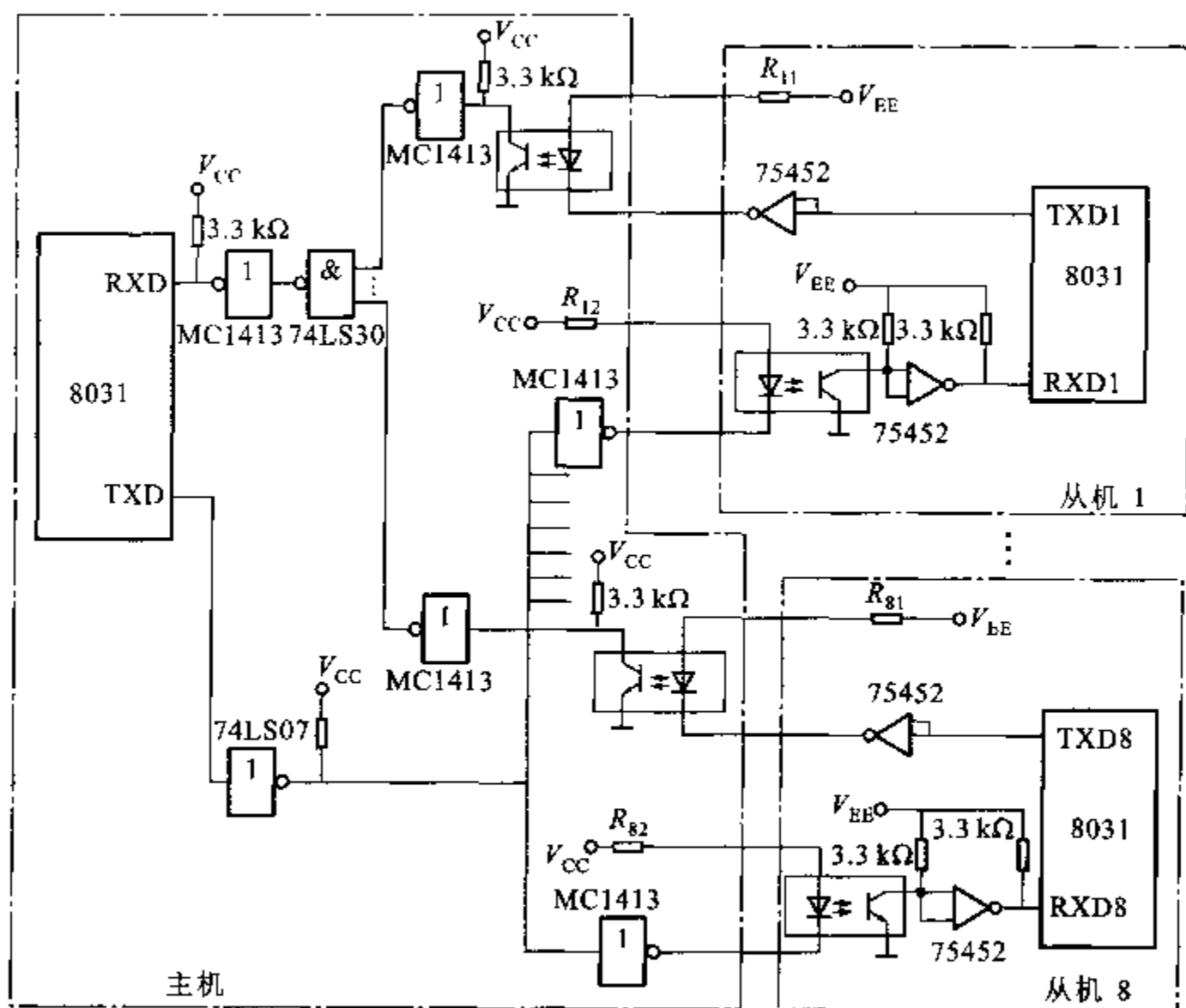


图 13-18 20 mA 电流环串行多机通信接口电路图

在图 13-18 中, 光电耦合器采用 TIL113。所有从机采用的与非门均为 DS75452N, 主机所用非门为 MC1413。图 13-18 的接口电路实现了 1 台主机与 8 台从机之间的通信。由图 13-18 可见, 主机发送信息时, 8 台从机均可接收, 从机发送信息时, 仅主机接收, 从而实现了主从式多机通信。

图 13-18 中, $R_{11}, R_{12}, \dots, R_{81}, R_{82}$ 的选取原则是保证环路中电流为 20 mA。以从机 1 与主机之间的通信为例, 说明 R_{11}, R_{12} 的选取方法。设主机与从机之间的通信距离为 600 m, 每 200 m 通信线电阻为 1Ω , 则通信线总电阻为 6Ω , TIL113

中发光二极管两端电压约 2 V, MC1413 输出低电平约 0.3 V, 则 R_{11}, R_{12} 为:

$$R_{11} = R_{12} = \frac{5 - 2 - 0.3}{0.02} \Omega = 6 \Omega \approx 130 \Omega$$

3. 软件编程思想

我们已在第 7 章中介绍了串行口多机通信的原理。在单片机多机通信中, 要保证主机与从机间可靠的通信, 必须保证通信接口具有识别功能, 而串行口控制寄存器 SCON 中的控制位 SM2 就是为满足这一要求而设置的。当串行口以方式 2(或方式 3)工作时, 发送和接收的每一帧信息都是 11 位, 其中第 9 数据位是可编程位, 通过对 SCON 的 TB8 赋予 1 或 0, 以区别发送的是地址帧还是数据帧(规定地址帧的第 9 位为 1, 数据帧的第 9 位为 0)。若从机的控制位 $SM2=1$, 则当接收的是地址帧时, 数据装入 SBUF, 并置 $RI=1$ 向 CPU 发出中断请求; 若接收的是数据帧, 则不产生中断标志, 信息将被抛弃。若 $SM2=0$, 则无论是地址帧还是数据帧都产生 $RI=1$ 中断标志, 数据装入 SBUF。鉴于此, 我们可规定软件编程的具体要求如下:

- (1) 使所有从机的 SM2 位置 1, 处于只接收地址帧的状态。
- (2) 主机发送 1 帧地址信息, 其中包含 8 位地址, 第 9 位为 1, 以表示发送的是地址。
- (3) 从机接收到地址帧后, 各自将接收到的地址与其本身地址相比较。
- (4) 被寻址的从机清除其 SM2, 未被寻址的其它从机仍维持 $SM2=1$ 不变。
- (5) 主机发送数据或控制信息(第 9 位为 0)。对于已被寻址的从机, 因 $SM2=0$, 故可以接收主机发来的信息。而对于其它从机, 因 SM2 仍为 1, 对主机发来的数据帧将不予理睬, 不予接收。
- (6) 当主机改为与另外从机联系时, 可再发出地址帧寻址其从机。而先前被寻址过的从机在分析出主机是对其它从机寻址时, 恢复其 $SM2=1$, 对随后主机发来的数据帧不加理睬。

串行通信可采用主机查询、从机中断方式的多机通信软件设计。即主机程序部分以子程序方式给出, 要进行串行通信时, 可直接调用; 从机部分以串行口中断服务方式给出, 其中断入口地址为 0023H。若从机未作好接收或发送准备, 就从中断程序返回。在主程序中做好准备。主机应重新和从机联络, 使从机再次执行串行口中断服务程序。

遵循上述多机通信的通信协议, 即可着手编写程序。

13.1.4 PC 机与 MCS-51 的点对点的串行通信接口

1. 硬件接口设计

如前所述,通常 PC 机都配有 RS-232C 串行标准接口,有效通信距离较短,为实现长距离通信应将 RS-232C 接口转换成 RS-422A/RS-485 接口。图 13-19 给出了这种转换的电路原理图。

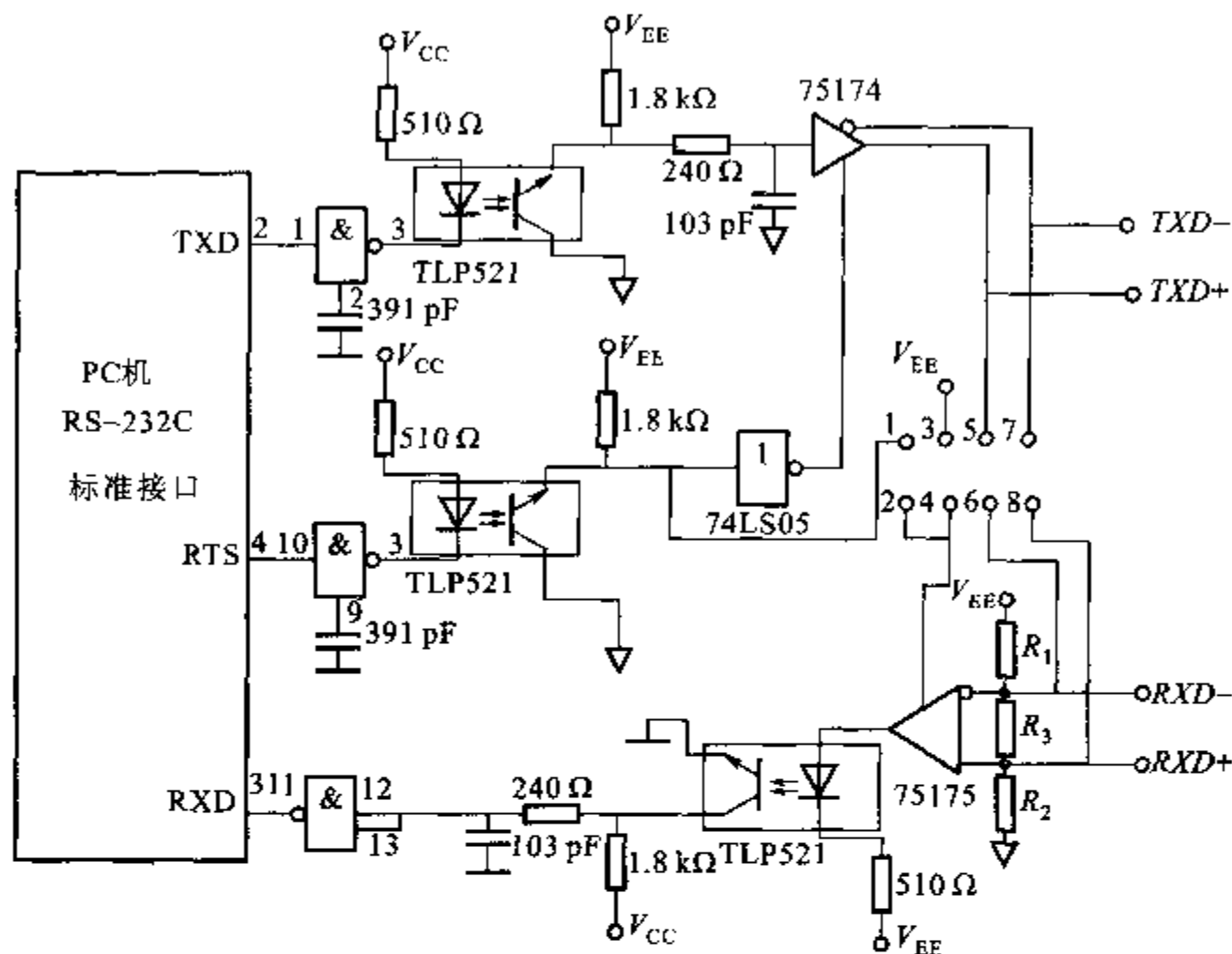


图 13-19 PC 机 RS-232C 至 RS-422A/RS-485 转换电路

图 13-19 可完成 RS-232C 至 RS-422A 的转换,也可完成 RS-232C 至 RS-485 的转换。当选择 RS-422A 输出方式时,3、4 短接;当选择 RS-485 输出方式时,1、2 短接,5、6 短接,7、8 短接。图 13-19 中, R_1 、 R_2 是为排除第一个数据传输误码而设置的匹配电阻。设计者可根据实际情况选择该电阻的大小。电源 V_{CC} 和 V_{EE} 均为 +5 V,但不是 1 个电源, V_{CC} 和 V_{EE} 应为隔离电源信号,只有这样才能实现电隔离。

通过图 13-19 的转换电路可使 PC 机具有 RS-422A/RS-485 串行接口。PC 机与 8031 单片机通信时,由于 8031 输入、输出电平均为 TTL 电平,二者的电气规范不一致,因此要完成 PC 机与单片机的数据通信,必须进行电平转换。只要 MCS-51 单片机配置相应的接口电路,就可实现符合 RS-232C、RS-422A、RS-485 标准的串行通信,其接口电路已在 13.1.2 节中作过介绍,这里不再复述。

2. 软件设计思想

PC 机的通信软件采用 8086/8088 汇编语言编写或采用 VB、C 等高级语言

来编写,单片机的程序编写直接用 MCS-51 的语言编写。具体的程序可参阅有关的参考书目。

13.1.5 PC 机与多个 MCS-51 单片机的串行通信接口

1. 硬件接口电路

将一台 IBM-PC 机和若干台 MCS-51 单片机构成小型分布式测控系统,如图 13-20 所示,是目前单片机应用的一大趋势。这种分布式测控系统在许多实时的工业控制和数据采集系统中,充分发挥了单片机功能强、抗干扰性能好、面向控制等优点,同时又可利用 PC 机弥补单片机在数据处理和交互性等方面的不足。在应用系统中,一般是以 PC 机作为主机,定时扫描以 MCS-51 为核心的前沿单片机,以便采集数据或发送控制信息。在这样的系统中,以 8031 芯片为核心的智能式测量和控制仪表(从机)既能独立地完成数据处理和控制任务,又可将数据传送给 PC 机(主机)。PC 机将这些数据进行处理,或显示,或打印,同时将各种控制命令传送给各个子机,以实现集中管理和最优控制。显然,要组成一个这样的分散控制或测量系统,首先要解决的是 PC 机与单片机之间的串行数据通信接口问题。

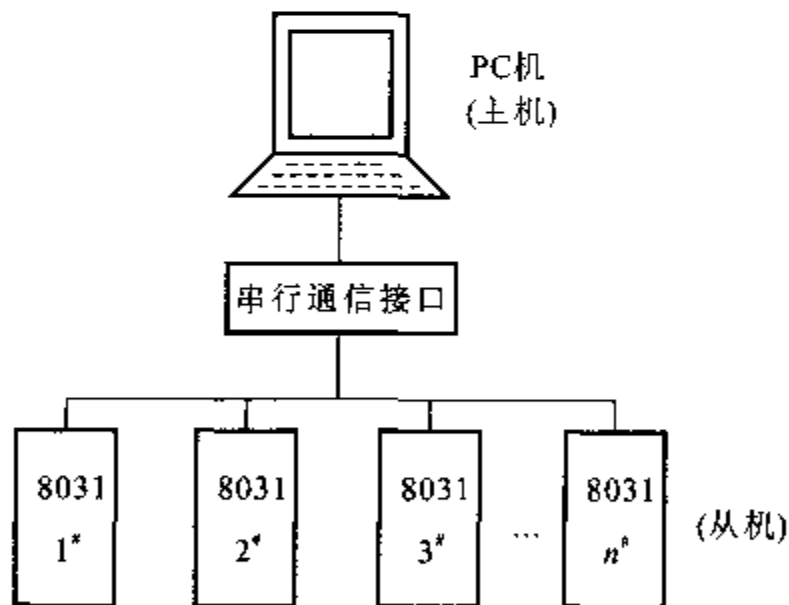


图 13-20 PC 机与多台 MCS-51 单片机构成小型的分布式测控系统

以 RS-485 串行多机通信为例,说明 PC 机与数台 8031 单片机进行多机通信的接口电路的设计方案。PC 机内一般都配有 RS-232C 串行标准接口,通过如图 13-19 的电路可转换成 RS-485 串行接口,8031 单片机本身具有一个全双工的串行口,该串行口加上驱动电路后就可实现 RS-485 串行通信。PC 机与数台 8031 单片机进行多机通信的 RS-485 串行通信接口电路如图 13-21 所示。

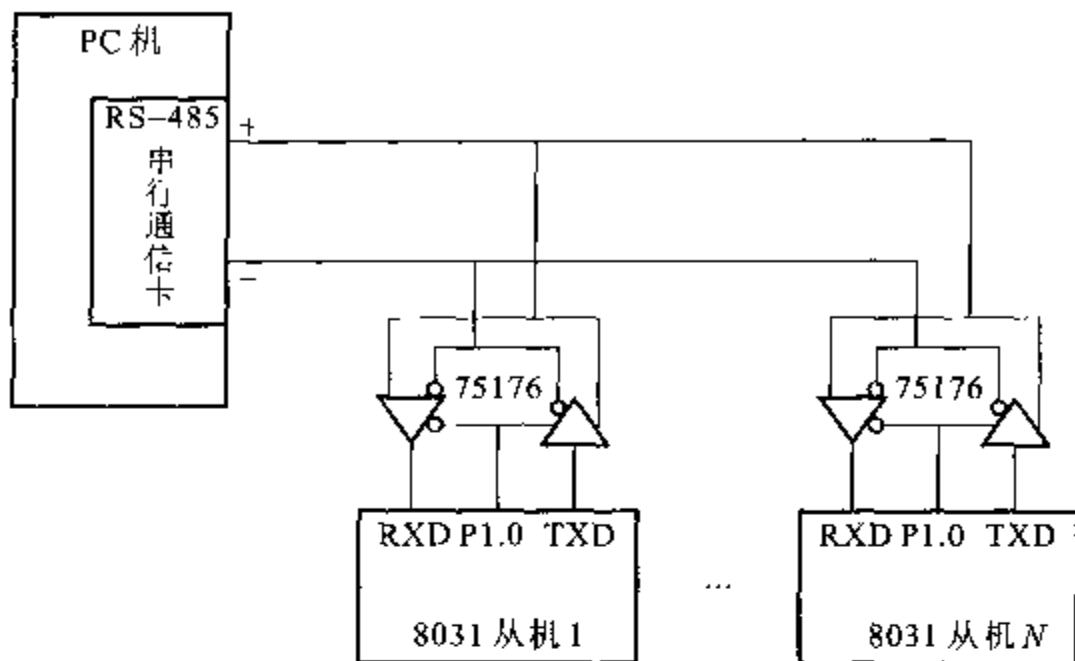


图 13-21 PC 机与 8031 单片机串行通信接口电路

在图 13-21 中,8031 单片机的串行口通过 75176 芯片驱动后就可转换成 RS-485 标准接口,根据 RS-485 标准接口的电气特性,从机数量不多于 32 个。有关 75176 芯片的说明在 13.1.1 节有详述。PC 机与 8031 单片机之间的通信采用主从方式,PC 机为主机,8031 单片机为从机,由 PC 机确定与哪个单片机进行通信。

2. 软件设计思想

为了充分发挥高级语言(如 C, BASIC)编程简单,调试容易,制图作表能力强的优点和汇编语言执行速度快的特点,PC 机软件可采用 C、BASIC 等语言编写的主程序调用汇编子程序的方法,即 PC 机的主程序由 C 语言编写,通信子程序由 PC 机汇编语言编制。这涉及 C 语言与汇编语言混合编程技术。高、低级语言混合编程技术的详细内容请参阅有关参考书目。

13.2 MCS-51 单片机与日历时钟芯片的接口

在单片机应用系统中,有时需要一个实时时钟供定时、测控之用。由于实时日历时钟的集成电路有多种,设计者只需选择合适的芯片即可。本节介绍日历时钟芯片 DS12887 的功能及其与 MCS-51 单片机的接口设计。

DS12887S 是跨越 2000 年的日历时钟芯片,在该芯片中用 4 位数来表示年度,采用 24 引脚双列直插式封装。该芯片的晶体振荡器、振荡电路、充电电路和可充电锂电池等一起封装在芯片的上方,组成 1 个加厚的集成电路模块。电路通电时,充电电路自动对电池充电。充足 1 次电可供芯片时钟运行半年之久,正

常工作时可保证时钟数据 10 年内不会丢失。此外,片内带有 114 B 的 RAM。

13.2.1 DS12887 日历时钟芯片的性能及引脚说明

1. 性能

DS12887 的主要性能如下:

(1) 具有时钟、闹钟功能及到 2100 年的日历功能,可选择 12 小时制或 24 小时制计时,有 AM 和 PM、星期、夏令时制时间操作及闰年自动补偿等功能。

(2) DS12887 内部有 14 个寄存器,包括 10 个时标寄存器、4 个状态寄存器。还有 114 B 作掉电保护用的低功耗 RAM。

(3) 具有用软件编程选择的周期性中断方式和多频率输出的方波发生器功能。

(4) 该芯片可以满足各种不同的待机要求,最长可达 24 小时。

(5) 可选择二进制或 BCD 码来表示时间。

(6) 工作电压: +4.5 ~ +5.5 V。

(7) 工作电流: 7 ~ 15 mA。

(8) 工作温度范围: 0 ~ +70 °C

2. 引脚说明

DS12887/12C887 的引脚如图 13-22 所示。

各引脚功能如下:

(1) MOT: 计算机总线选择端(接低电平为总线方式);

(2) SQW: 方波输出,是否输出以及速率由专用寄存器 A、B 的预置参数决定;

(3) AD7 ~ AD0: 地址/数据(双向)总线,由 ALE 信号的下降沿锁存 8 位地址;

(4) \overline{WR} : 写数据控制信号端,低电平有效;

(5) ALE: 地址锁存信号端;

(6) \overline{RD} : 数据读控制信号端,低电平有效;

(7) \overline{CS} : 片选信号端,低电平有效;

(8) \overline{IRQ} : 中断申请端,低电平有效,由专用寄存器决定;

(9) \overline{RESET} : 复位端,低电平复位;

(10) NC: 空闲端。

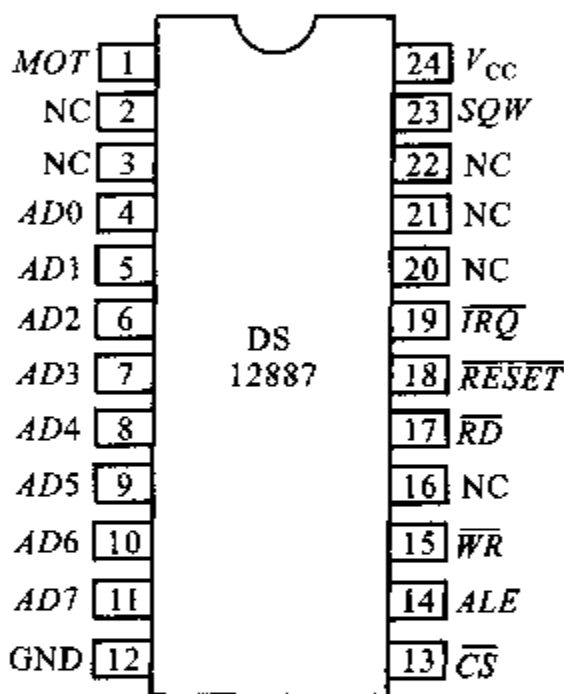


图 13-22 DS12887 的引脚

13.2.2 DS12887 的内部 RAM 和寄存器

CPU 通过读 DS12887 的内部时标寄存器得到当前的时间和日历,也可通过选择二进制码或 BCD 码初始化芯片的 10 个时标寄存器。其中 114 B 的静态 RAM 可供用户使用,也可在系统掉电时用来保存一些重要的数据。DS12887 的 4 个控制与状态寄存器用来控制或反映 DS12887 模块当前的工作状态,除数据更新周期外,软件可随时读写这 4 个寄存器。下面介绍各寄存器的功能和作用。

1. 内部 RAM 与各专用寄存器简介

引脚 AD7~AD0 的 8 位编码决定了对各专用寄存器与内部 RAM 的寻址。其中:

00H~09H:10 个时标寄存器;

0AH~0DII:1 个控制与状态寄存器;

0EH~7FII:114 B 的静态 RAM;

表 13-6 是 DS12887 内部 10 个时标寄存器和 4 个控制/状态寄存器的地址分配。

表 13-6 DS12887 内部 RAM 和专用寄存器地址

地 址 单 元	用 途	地 址 单 元	用 途
00H	秒	01H	秒闹
02H	分	03H	分闹
04H	时	05H	时闹
06H	星期	07H	日(两位数)
08H	月(两位数)	09H	年(两位数)
0AH	寄存器 A	0BH	寄存器 B
0CH	寄存器 C	0DH	寄存器 D
0EH~7FH	掉电保护 RAM 区,共 114 B		

(1) 时标寄存器(10 个)

00H~03H 单元为“秒”、“分”单元,取值范围是 00H~3BH(十进制 0~59);

04H~05H 单元是“时”单元,按 12 小时制的取值范围是上午(AM)01H~0CH(1~12),下午(PM)81H~8CH(1~12);按 24 小时制的取值范围是 00H~17H(0~23);

06H 单元是“星期”单元,取值范围是 01H~07H(1~7);

07H 单元是“日”单元,取值范围是 01H~1FH(1~31);

08H 单元是“月”单元,取值范围是 01H~0CH(1~12);

09H 单元是“年的低 2 位”单元,取值范围是 00H~63H(0~99)。

MCS-51 对 DS12887 内部 RAM 和各专用寄存器的地址分配可采用译码器法。假设 DS12887 的片选地址从 $\overline{CS} = D000H$ 开始有效,则芯片内部 RAM 和寄存器的地址为 $\#D000H \sim \#D07FH$ 。应当注意的是,尽管 DS12887 的专用时标年寄存器只有 1 个,只能显示年度的低 2 位,但通过软件编程可利用其内部的不掉电 RAM 区的 1B 实现年度的高 2 位显示。

(2) 控制与状态寄存器(4 个)

① 寄存器 A

寄存器 A 各位不受复位的影响,格式见表 13-7。

表 13-7 控制寄存器 A 的格式及输出速率选择

D7	D6	D5	D4	D3	D2	D1	D0	以 32 768 Hz 为时基速率输出	
UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0	中断周期	SQW 输出频率
—	—	—	—	0	0	0	0	无	无
—	—	—	—	0	0	0	1	3.906 25 ms	256 Hz
—	—	—	—	0	0	1	0	7.812 5 ms	128 Hz
—	—	—	—	0	0	1	1	0.122 07 ms	8.192 kHz
—	—	—	—	0	1	0	0	0.244 141 ms	4.096 kHz
—	—	—	—	0	1	0	1	0.488 281 ms	7.048 kHz
—	—	—	—	0	1	1	0	0.976 563 ms	1.024 kHz
—	—	—	—	0	1	1	1	1.953 ms	512 Hz
—	—	—	—	1	0	0	0	3.906 25 ms	256 Hz
—	—	—	—	1	0	0	1	7.812 125 ms	128 Hz
—	—	—	—	1	0	1	0	15.625 ms	64 Hz
—	—	—	—	1	0	1	1	31.25 ms	32 Hz
—	—	—	—	1	1	0	0	62.5 ms	16 Hz
—	—	—	—	1	1	0	1	125 ms	8 Hz
—	—	—	—	1	1	1	0	250 ms	4 Hz
—	—	—	—	1	1	1	1	500 ms	2 Hz

其中:

- 位 UIP:更新周期标志位

该位是只读位。UIP=1 时,表示芯片正处于或即将开始更新周期,此时不准读/写时标寄存器;UIP=0 时,表示至少在 241 μ s 后才开始更新周期,此时程序可以读芯片内时标寄存器。

- 位 DV0、DV1、DV2:芯片内部振荡器 RTC 控制位

当芯片解除复位状态,并将 010 写入 DV0、DV1、DV2 后,另一个更新周期将在 500 ms 后开始。因此,在程序初始化时可用这 3 位精确地使芯片在设定的时间开始工作。DS12887 固定使用 32 768 Hz 的内部晶体。所以,DV0、DV1、DV2=010,即只有 010 这一种组合选择才可启动 RTC。

- 位 RS3、RS2、RS1、RS0:周期中断可编程方波输出速率选择位

对这些位进行不同的组合可以产生不同的输出。程序可以通过设置寄存器 B 的 SQWE 和 PIE 位来控制是否允许方波输出和周期中断。寄存器 A 输出方波的速率选择位见表 13-7。

② 寄存器 B

寄存器 B 允许读/写,主要用于控制芯片的工作状态。其控制字的格式见表 13-8。

表 13-8 寄存器 B、C、D 的控制字/状态字的格式

寄 存 器	D7	D6	D5	D4	D3	D2	D1	D0
寄存器 B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE
寄存器 C	IRQF	PF	AF	UF	0	0	0	0
寄存器 D	VRT	0	0	0	0	0	0	0

其中:

- 位 SET:SET=0 时,芯片处于正常工作状态,每秒产生一个更新周期来更新时标寄存器;SET=1 时,芯片停止工作,程序在此期间可初始化芯片的各个时标寄存器。

- 位 PIE、AIE、UIE:分别为周期中断、报警中断、更新周期结束中断允许位。各位为 1 时,允许芯片发出相应的中断。

- 位 SQWE:方波输出允许位。SQWE=1,输出按寄存器 A 输出速率选择位所确定的频率方波;SQWE=0,引脚 SQW 保持低电平。

- 位 DM:时标寄存器用十进制 BCD 码表示或用二进制表示格式选择位。DM=0 时,为十进制 BCD 码;DM=1 时,为二进制码。

- 位 24/12:24/12 小时制模式设置位。24/12=1 时,为 24 小时制工作模

式: $24/12=0$ 时, 为 12 小时制工作模式。

- 位 DSE: 夏时制设置。DSE=1, 夏时制设置有效, 夏时制结束时可自动刷新恢复时间; DSE=0, 无效。

③ 寄存器 C

该寄存器的特点是, 程序访问该寄存器后, 该寄存器的内容将自动清 0, 从而使 IRQF 标志位变为高电平, 否则, 芯片将无法向 CPU 申请下一次中断。寄存器 C 的控制字格式见表 13-8。

其中:

- 位 IRQF: 中断申请标志位。该位有关的逻辑变量的逻辑关系表达式如下:

$$IRQF = PF \cdot PIE + AF \cdot AIE + UF \cdot UIE$$

当 IRQF 位变为 1 时, 引脚变为低电平引起中断申请。

- 位 PF、AF、UF: 这 3 位分别为周期中断、报警中断、更新周期结束中断标志位。只要满足各中断的条件, 相应的中断标志位将置 1。

- 位 D3~位 D0: 保留位, 读出值始终为 0。

④ 寄存器 D

寄存器 D 为只读的状态寄存器, 状态字的格式见表 13-8。其中:

- 位 VRT: 芯片内部 RAM 与寄存器内容有效标志位。该位为 1 时, 指芯片内部 RAM 和寄存器内容有效。读该寄存器后, 该位将自动置 1。

- 位 D6~D0: 保留位, 读出值始终为 0。

2. DS12887 的中断和更新周期

DS12887 处于正常工作状态时, 每秒钟将产生一个更新周期。芯片处于更新周期的标志是, 寄存器 A 中的 UIP 位为 1。在更新周期内, 芯片内部时标寄存器中的数据处于更新阶段。故在该周期内, 微处理器不能读时标寄存器的内容, 否则将得到不确定数据。更新周期的基本功能主要是刷新各个时标寄存器中的内容, 同时秒时标寄存器内容加 1, 并检查其他时标寄存器内容是否有溢出, 如有溢出则相应进位日、月、年。更新周期的另外一个功能是, 检查 3 个时、分、秒报警时标寄存器的内容是否与对应时标寄存器的内容相符。如果相符, 则寄存器 C 中的 AF 位置 1; 如果报警时标寄存器的内容为 C0H~FFH 之间的数据, 则为不关心状态。

为了采样时标寄存器中的数据, 器件提供了两种避开在更新周期内访问时标寄存器的方法。第一种方法是利用更新周期结束发出的中断。可以编程且允许在每次更新周期结束后发生中断申请, 提醒 CPU 将有 998 ms 左右的时间去获取有效的数据, 在中断之后的 998 ms 时间内, 程序可先将时标数据读到芯片内部的不掉电静态 RAM 中。因为芯片内部的静态 RAM 和状态寄存器是可随

时读写的,在离开中断服务子程序前应清除寄存器 C 中的 IRQF 位。另一种方法是,利用寄存器 A 中的 UIP 位来指示芯片是否处于更新周期。在 UIP 位由低变高 $244\mu\text{s}$ 后,芯片将开始其更新周期。所以,若检测到 UIP 位为低电平时,则利用 $244\mu\text{s}$ 的间隔时间去读取时标信息;如检测到 UIP 位为 1,则可暂缓读数据,等到 UIP 位变成低电平后再去读数据。

13.2.3 MCS-51 与 DS12887 的接口设计

1. DS12887 的初始化

DS12887 采用连续工作制,一般无需每次都初始化,即使是系统复位时也如此。但初始化时,首先应禁止芯片内部的更新周期操作。所以,应先将 DS12887 状态寄存器 B 中的 SET 位置 1,然后初始化 00H~09H 时标参数寄存器和状态寄存器 A。此后,再通过读状态寄存器 C,清除寄存器 C 中的周期中断标志位 PF、报警中断标志位 AF、更新周期结束中断标志位 UF。寄存器 D 中的 VRT 位在读操作后将自动置 1。最后,将状态寄存器 B 中的 SET 位置 0,芯片开始计时工作。

2. 闹钟单元的使用

DS12887 共有 3 个闹钟单元,分别为时、分、秒闹钟单元。在其中写入闹钟时间值并且在时钟中断允许的情况下,每天到该时刻就会产生中断申请信号。但这种方式每天只提供 1 次中断信号。另一种方式是在闹钟单元中写入“不关心码”:在时闹钟单元写入 C0H~FFH 之间的数据,可每小时产生 1 次中断;在时、分闹钟单元写入 C0H~FFH 之间的数据,可每分钟产生 1 次中断;而时、分、秒闹钟单元全部写入 FFH,则每秒钟产生 1 次中断。但这种方式也只是在整点、整分、或每秒产生 1 次中断。若控制系统要求的定时间隔不是整数时,应该通过软件来调整实现。

3. 接口电路及软件编程

图 13-23 是 AT89C51 单片机与 DS12887 的接口电路(其中片选地址 $\overline{CS}=0D00H$)。DS12887 状态寄存器的参数设置如下:状态寄存器 A 置为 20H,它表示采用的时钟频率为 32 678 Hz,禁止引脚 SQW 输出;状态寄存器 B 置为 22H,它表示允许报警中断,禁止其他中断,为 24 小时制模式,时标寄存器内容用 BCD 码表示,禁止方波输出和夏时制服务。如果要求定时间隔为 1~59 s 的中断申请,那么时报警寄存器置 FFH,这就表示该报警时标处于不关心状态。

下面为 DS12887 的有关程序。

(1) 初始化程序

```
TIME: MOV     DPTR, #0D00BH      ;寄存器 B 中的 SET 位置 1,禁止芯片
```

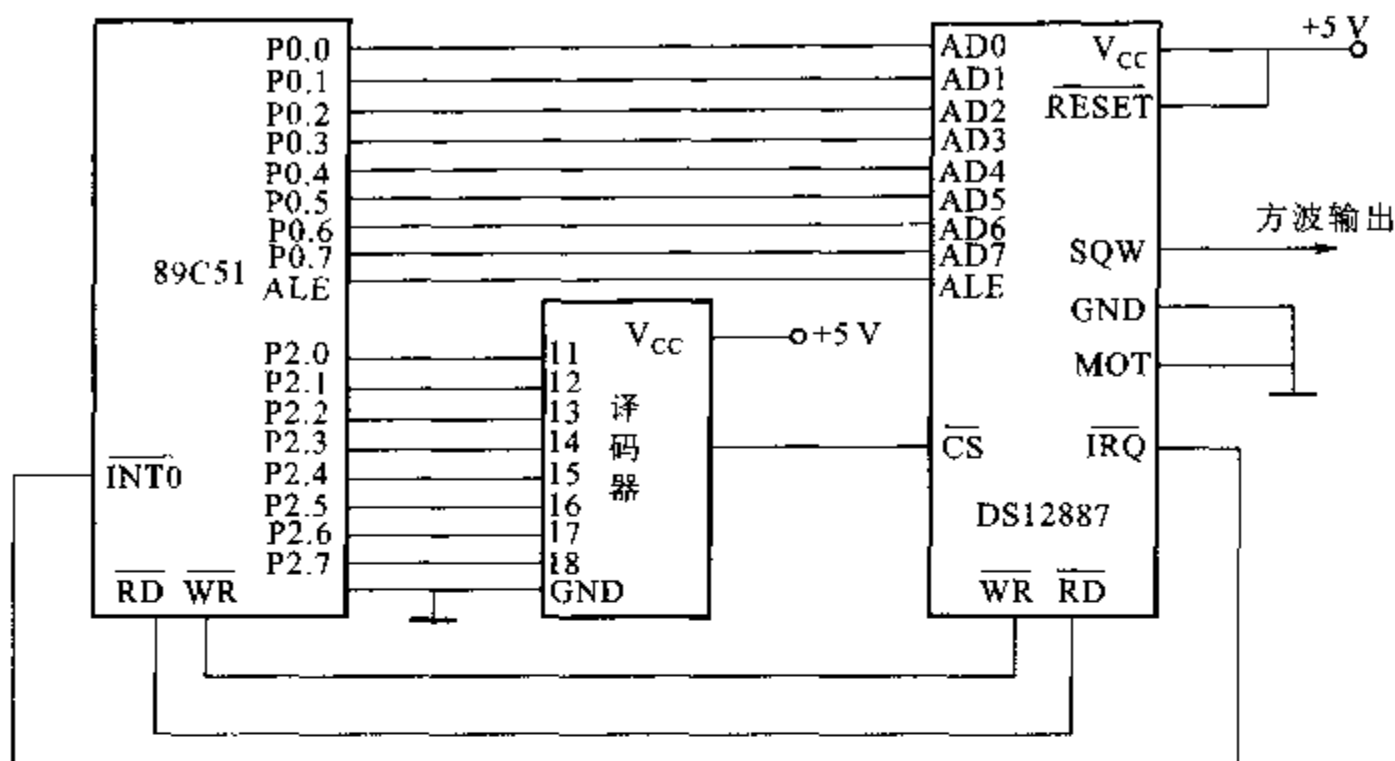


图 13-23 DS12887/12C887 与 MCS-51 单片机的接口电路

MOV	A, #0A2H	;内部的更新周期
MOVX	@DPTR, A	;初始化时标寄存器,输入当前时间:
MOV	DPTR, #0D000H	;2003 年 8 月 9 日,星期六,12:00:00
MOV	A, #00H	;秒时标单元地址送 DPTR
MOVX	@DPTR, A	;秒时标单元送 0
INC	DPTR	
MOV	A, #0FFH	;秒时标报警单元送不关心码
MOVX	@DPTR, A	
INC	DPTR	;DPTR 指向分时标单元
MOV	A, #00H	;分时标单元送 0
MOVX	@DPTR, A	
INC	DPTR	
MOV	A, #0FFH	;分时标单元送不关心码
MOVX	@DPTR, A	
INC	DPTR	;DPTR 指向小时时标单元
MOV	A, #0CH	;小时时标单元送 12
MOVX	@DPTR, A	
INC	DPTR	
MOV	A, #0FFH	;小时时标报警单元送不关心码
MOVX	@DPTR, A	
INC	DPTR	;DPTR 指向星期时标单元

```

MOV    A, #06H           ;星期时标单元送 6
MOVX   @DPTR, A
INC    DPTR              ;DPTR 指向日期时标单元
MOV    A, #09H           ;日期时标单元送 9
MOVX   @DPTR, A
INC    DPTR              ;DPTR 指向月时标单元
MOV    A, #08H           ;月时标单元送 8
MOVX   @DPTR, A
INC    DPTR              ;DPTR 指向年时标单元
MOV    A, #03H           ;年时标单元送 03
MOVX   @DPTR, A
MOV    DPTR, #0D00EH     ;DPTR 指向年度高 2 位 RAM 单元
MOV    A, #14H           ;年度高 2 位送 20
MOVX   @DPTR, A
MOV    DPTR, #0DD0AH
MOV    A, #20H           ;初始化状态寄存器 A
MOVX   @DPTR, A
MOV    DPTR, #0D00CH
MOVX   A, @DPTR          ;清状态寄存器 C
INC    DPTR
MOVX   @DPTR             ;状态寄存器 D 的 VRT 位置 1
MOV    DPTR, #0D00BH
MOV    A, #22H           ;初始化状态寄存器 B
MOVX   @DPTR, A
MOV    IE, #81H         ;89C51 开中断
RET

```

(2) 查询法判别芯片是否处于更新周期子程序

```

XIN:   MOV    DPTR, #0D00AH ;DPTR 指向状态寄存器 A
        MOV    A, @DPTR
        JB     ACC. 7, XIN   ;查询 UIP 位
        SETB   20H          ;设可读时标寄存器标志位
        RET

```

中断服务子程序

```

INT1:   LCALL  XIN
        JB 20H, INTG
        AJMP  INT1
INTG:   ...                ;读当前时标寄存器

```

```

...                ;检查是否溢出
...                ;溢出处理
MOV    DPTR, #0D00CH ;清中断标志寄存器
MOVX   A, @DPTR
RETI

```

13.3 MCS-51 单片机的报警接口

在单片机测控系统发生故障或处于某种紧急状态时,单片机系统应能发出提醒人们警觉的报警信号或提示信号,常见的报警信号可分为闪光报警、鸣音报警和音乐报警。下面对这三种报警接口作以介绍。

13.3.1 闪光报警接口

闪光报警可用 MCS-51 的某一 I/O 口线驱动 LED 闪烁,只要该 I/O 口线发出具有一定频率的高低电平信号,即可使 LED 闪烁。接口电路如图 13-24 所示。

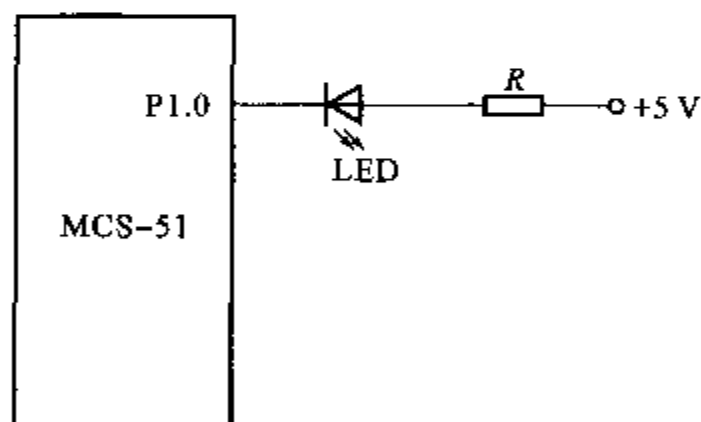


图 13-24 MCS-51 的某一 I/O 口线驱动 LED 闪烁

对图 13-24 闪光报警的程序编写,只需向 P1.0 先写入 1,然后延迟一段时间,再向 P1.0 写入 0,延迟的时间可根据要求的 LED 的闪烁频率而定,可采用软件延时,也可使用定时器定时中断,在中断服务程序中,改变 P1.0 的电平。采用软件延时的参考程序如下:

```

:
FLASH1:  SETB    P1.0
          LCALL   DELAY    ;调用软件延时子程序 DELAY
          CLR     P1.0
          LCALL   DELAY    ;调用软件延时子程序 DELAY

```

```
LJMP    FLASH1
```

```
:
```

如果 MCS-51 本身的 I/O 口线资源紧张,或要控制闪光报警的 LED 数目较多,也可采用扩展 1 片 74LS377 8D 锁存器芯片,锁存器的输出接 8 个 LED。接口电路如图 13-25 所示。

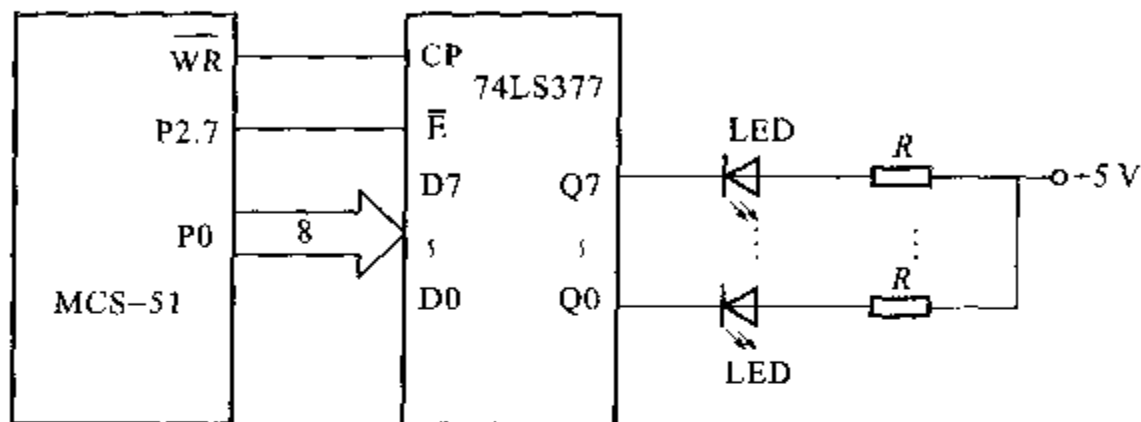


图 13-25 通过扩展 74LS377 8D 锁存器的闪光报警电路

图 13-25 中的 74LS377 的端口地址为 7FFFH,要使某一位的 LED 闪烁,只需向 74LS377 的某一位交替的写入 1 或 0。写入 1 或 0 的时间间隔可采用软件延时,也可使用定时器定时中断。例如,要使 74LS377 Q0 位上的 LED 闪烁,采用软件延时的参考程序如下:

```
FLASH2:  MOV    DPTR,#7FFFH
          MOV    A ,#0FEH
          MOVX   @DPTR,A      ;让最低位 LED 亮
          LCALL  DELAY        ;调用软件延时子程序 DELAY
          MOV    A ,#0FFH     ;让最低位 LED 灭
          LCALL  DELAY        ;调用软件延时子程序 DELAY
          RET
```

13.3.2 蜂鸣音报警接口

蜂鸣音报警接口电路的设计只需购买市售的压电式蜂鸣器,然后通过 MCS-51 的 1 根口线经驱动器驱动蜂鸣器发声。压电式蜂鸣器约需 10 mA 的驱动电流,可以使用 TTL 系列集成电路 7406 或 7407 低电平驱动,如图 13-26 所示,也可以用 1 个晶体三极管驱动,如图 13-27 所示。

在图 13-26 中,MCS-51 的口线 P1.7 接驱动器的输入端。当 P1.7 输出高电平时,7406 的输出为低电平,在压电蜂鸣器 2 条引线上加上近 5 V 的直流电压,由压电效应而发出蜂鸣音。当 P1.7 端输出低电平时,7406 的输出端高约

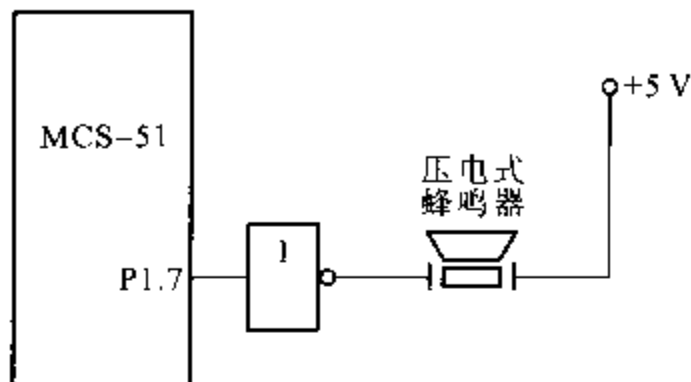


图 13-26 通过 74LS06 来驱动蜂鸣器的报警电路

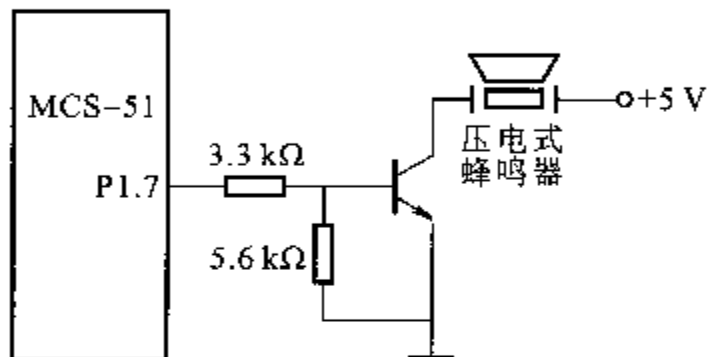


图 13-27 使用三极管驱动的蜂鸣器报警电路

+5 V, 压电蜂鸣器的 2 条引线间的直流电压降至接近于 0 V, 发音停止。

在图 13-27 中, P1.7 接晶体管基极输入端。当 P1.7 输出高电平时, 晶体管导通, 压电蜂鸣器两端获得约 +5 V 电压而鸣叫; 当 P1.7 输出低电平, 三极管截止, 蜂鸣器停止发音。

下面是连续蜂鸣 100 ms 参考程序, 该程序对上述 2 个接口电路都适用。

```
SOUND: SETB    P1.7           ;P1.7 输出 WEI 高电平, 蜂鸣器开始鸣叫
        MOV     R4, #64H       ;延时 100 ms
LOOP:   MOV     R3, #0F9H
LOOP1:  DJNZ    R3, LOOP1       ;延时 1 ms 的循环
        DJNZ    R4, LOOP
        CLR     P1.7           ;P1.7 输出低电平, 蜂鸣器停止鸣叫
        RET
```

如果想要发出更大的声音, 可采用功率大的扬声器作为发声器件, 这时要采用相应的功率驱动器。具体的接口电路读者可参考本书第 12 章的例 12-2。

13.3.3 音乐报警接口

音乐报警电路可使报警声优美悦耳, 克服了蜂鸣音报警音调比较单调的不足。发声电路可购买市售的乐曲发生器, 发出的乐曲声可用来作为某种提示信号

或报警信号。设计者可根据自己对乐曲的喜好来购买相应的集成电路。

音乐报警接口由两部分组成：

- ① 乐曲发生器，即集成电子音乐芯片；
- ② 放大电路，也可采用集成放大器。

音乐报警接口电路如图 13-28 所示，图中采用华尔兹乐曲的电子音乐芯片 7920A。当 8031 从 P1.7 输出高电平时，电子音乐芯片 7920A 的输入控制端 MT 变为 1.5 V 高电平，输出端 V_{OUT} 便发出乐曲信号，经 M51182L 放大而驱动扬声器发出乐曲报警声。音量大小由 10 k Ω 电位器调整。相反，若 P1.7 输出低电平，则 7920A 因 MT 输入电位变低而关闭，故扬声器停止奏曲。音乐报警接口的参考程序为：

```

START:   SETB    P1.7           ;P1.7 为高电平，发出音乐报警乐曲
          RET
STOP:    CLR     P1.7           ;P1.7 为低电平，音乐报警乐曲停止
          RET
    
```

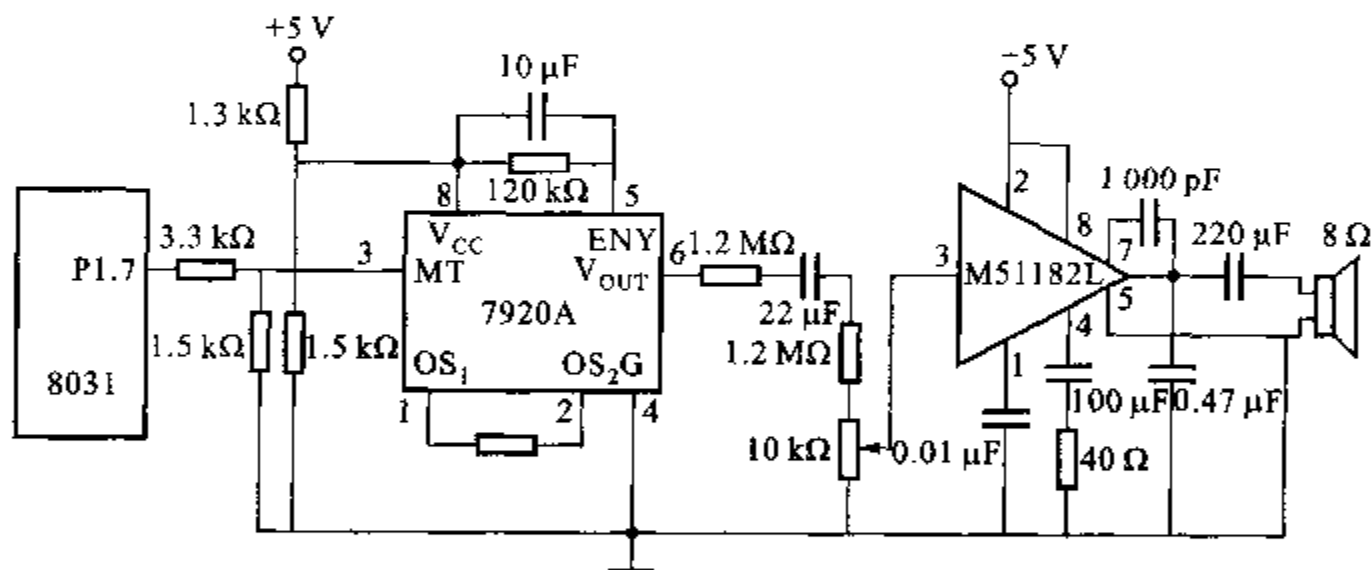


图 13-28 音乐报警接口电路

思考题及习题

1. 直接以 TTL 电平串行传输数据的方式有什么缺点？
2. 两台计算机通过公用电话网远距离串行通信时，为什么要使用调制解调器？
3. 图 13-15 的接口电路中，为什么使用 2 组独立的电源 V_{CC} 和外 5 V？
4. RS-422A 与 RS-232C 相比，都在哪些方面作了改进？哪些性能作了提高？
5. 比较 RS-232C、RS-422A、RS-485、20 mA 电流环这 4 种串行接口各有什么特点？
6. RS-485 串行接口能否实现全双工的数据传输？

第 14 章 MCS-51 应用系统的可靠性及抗干扰设计

目前,单片机应用系统已在工业测控领域中得到广泛应用,单片机系统的可靠性越来越受到人们的关注。单片机系统的可靠性是由多种因素决定的,其中系统的抗干扰性能的好坏是影响系统可靠性的重要因素。因此,研究抗干扰技术,提高单片机系统的抗干扰性能,是本章要研究的内容。本章将从干扰的来源、硬件、软件以及电源系统、接地系统等各个方面研究分析并给出有效可行的解决措施。

14.1 干扰的来源

一般把影响单片机测控系统正常工作的信号称为噪声,又称干扰。在单片机系统中,出现了干扰,就会影响指令的正常执行,造成控制事故或控制失灵,在测量通道中产生了干扰,就会使测量产生误差,电压的冲击有可能使系统遭到致命的破坏。

环境对单片机控制系统的干扰一般都是以脉冲的形式进入系统的,干扰窜入单片机系统的渠道主要有三条,如图 14-1 所示。

(1) 空间干扰

空间干扰来源于周围的电气设备如发射机、中频炉、晶闸管逆变电源等发出的电干扰和磁干扰;广播电台或通信发射台发出的电磁波;空中雷电,甚至地磁场的变化也会引起干扰。这些空间辐射干扰会使单片机系统不能正常工作。

(2) 供电系统干扰

由于工业现场运行的大功率设备众多,特别是大感性负载设备的启停会使得电网电压大幅度涨落(浪涌),工业电网电压的欠压或过压常常达到额定电压

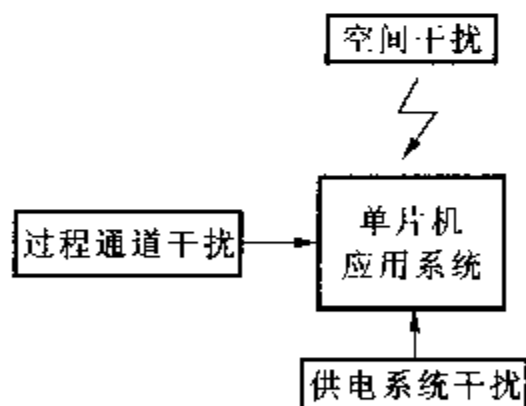


图 14-1 单片机测控系统
主要干扰渠道

的 $\pm 15\%$ 以上。这种状况有时长达几分钟、几小时、甚至几天。由于大功率开关的通断,电机的启停,电焊等原因,电网上常常出现几百伏,甚至几千伏的尖脉冲干扰。

(3) 过程通道干扰

为了达到数据采集或实时控制的目的,开关量输入输出,模拟量输入输出是必不可少的。在工业现场,这些输入输出的信号线和控制线多至几百条甚至几千条,其长度往往达几百米或几千米,因此不可避免地将干扰引入单片机系统。当有大的电气设备漏电,接地系统不完善,或者测量部件绝缘不好,都会使通道中直接串入干扰信号;各通道的线路如果同出自1根电缆中或绑扎在一起,各路间会通过电磁感应而产生瞬间的干扰,尤其是 $0\sim 15\text{ V}$ 的信号与交流 220 V 的电源线同套在1根长达几百米的管中,其干扰更为严重。这种彼此感应产生的干扰其表现形式仍然是通道中形成干扰电压。这样,轻者会使测量的信号发生误差,重者会使有用的信号被完全淹没。有时这种通过感应产生的干扰电压会达到几十伏以上,使单片机系统无法工作。

以上三种干扰以来自供电系统的干扰最甚,其次为来自过程通道的干扰。对于来自空间的辐射干扰,需加适当的屏蔽及接地来解决。

14.2 供电系统干扰及抗干扰措施

任何电源及输电线路都存在内阻,正是这些内阻才引起了电源的噪声干扰。如果没有内阻,无论何种噪声都会被电源短路吸收,在线路中不会建立起任何干扰电压。

单片机系统中最重要、危害最严重的干扰源来自于电源。在某些大功率耗电设备的电网中,经对电源检测发现,在 50 Hz 正弦波上叠加有很多 $1\,000$ 多伏的尖峰电压。

14.2.1 电源噪声来源、种类及危害

如果把电源电压变化持续时间定义为 Δt ,那么,根据 Δt 的大小可以把电源干扰分为:

- (1) 过压、欠压、停电: $\Delta t > 1\text{ s}$;
- (2) 浪涌、下陷: $1\text{ s} > \Delta t > 10\text{ ms}$;
- (3) 尖峰电压: Δt 为 μs 量级;
- (4) 射频干扰: Δt 为 ns 量级;

(5) 其它:半个周期内的停电或者过欠压。

过压、欠压、停电的危害是显而易见的,解决的办法是使用各种稳压器、电源调节器,对付短暂时间的停电则配置不间断电源(UPS)。

浪涌与下陷是电压的快变化,如果幅度过大也会毁坏系统。即使变化不大($\pm 10\% \sim \pm 15\%$),直接使用不一定会毁坏系统,但由于电源系统中接有反应迟缓的磁饱和或电子交流稳压器,往往会在这些变化点附近产生振荡,使得电压忽高忽低。如果有连续几个 $\pm 10\% \sim \pm 15\%$ 的浪涌或下陷,由此造成的振荡能产生 $\pm 30\% \sim \pm 40\%$ 的电源变化,而使系统无法工作,解决的办法是使用快速响应的交流电源稳压器。

尖峰电压持续时间很短,一般不会毁坏系统,但对单片机系统正常运行危害很大,会造成逻辑功能紊乱,甚至冲坏源程序。解决办法是使用具有噪声抑制能力的交流电源调节器、参数稳压器或超隔离变压器。

射频干扰对单片机系统影响不大,一般加接2~3级低通滤波器即可解决。

14.2.2 供电系统的抗干扰设计

单片机测控系统的供电,常常是一个棘手问题,单单一台高质量的电源不足以解决干扰和电压波动问题,必须完整地设计整个电源供电系统。

逻辑电路是在低电压、大电流下工作,电源的分配就必须引起注意,比如1条 0.1Ω 的电源线回路,对于5A的供电系统,就会把电源电压从5V降到4.5V,以至不能正常工作。另一方面工作在极高频率下的数字电路,对电源线有高频要求,所以一般电源线上的干扰是数字系统最常出现的问题之一。

电源分配系统首要的就是良好的接地,系统的地线必须能够吸收来自所有电源系统的全部电流。应该采用粗导线作为电源连接线,地线应尽量短且直接走线;对于插件式线路板,应多给电源线、地线分配几个沿插头方向均匀分布的插针。

在单片机系统中,为了提高供电系统的质量,防止窜入干扰,建议采用如图14-2所示的供电配置和如下措施:

(1) 交流近线端加交流滤波器,可滤掉高频干扰,如电网上大功率设备启停造成的瞬间干扰。滤波器市场上的产品有一级、二级滤波器之分,安装时外壳要加屏蔽并良好接地,进出线要分开,防止感应和辐射耦合。低通滤波器仅允许50Hz交流电通过,对高频和中频干扰有良好的衰减作用。

(2) 要求高的系统加交流稳压器。

(3) 采用具有静电屏蔽和抗电磁干扰的隔离电源变压器。

(4) 采用集成稳压块两级稳压。目前市场上集成稳压块有许多种,如提供正电源的7805、7812、7820、7824以及提供负电压的79系列稳压块,它们内部是多级

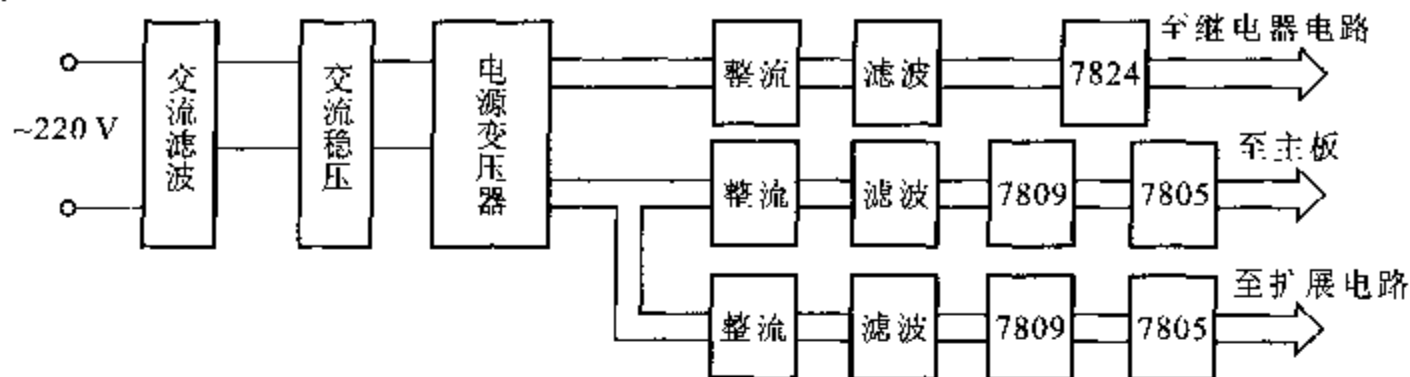


图 14-2 供电配置原理框图

稳压电路,采用两级稳压,效果好。例如主机电源先用 7809 稳到 9 V,再用 7805 稳到 5 V。

(5) 直流输出部分采用大容量电解电容进行平滑滤波。

(6) 交流电源线与其他线尽量分开,减少再度耦合干扰。如滤波器的输出线上干扰已减少,应使其与电源进线级滤波器外壳保持一定距离,交流电源线与直流电源线及信号线分开走线。

(7) 电源线与信号线一般都通过地板下面走线,而且不可把两线靠得太近或互相平行,以减少电源与信号线之间的相互影响。

(8) 在每块印制板的电源与地之间并接去耦电容。即 $5\sim 10\ \mu\text{F}$ 的电解电容和 1 个 $0.01\sim 1.0\ \mu\text{F}$ 的电容,以消除直流电源与地线中的脉冲电流所造成的干扰。

14.3 过程通道干扰的抑制措施——隔离

过程通道是系统输入、输出以及单片机之间进行信息传输的路径。抑制过程通道的干扰主要采用光电隔离技术。

14.3.1 光电隔离的基本配置

采用光电耦合器可以将单片机与前向、后向以及其他部分切断电路的联系,能有效地防止干扰从过程通道进入单片机。其原理如图 14-3 所示。

光电耦合的主要优点是能有效抑制尖峰脉冲以及各种噪声干扰,从而使过程通道上的信噪比大大提高。

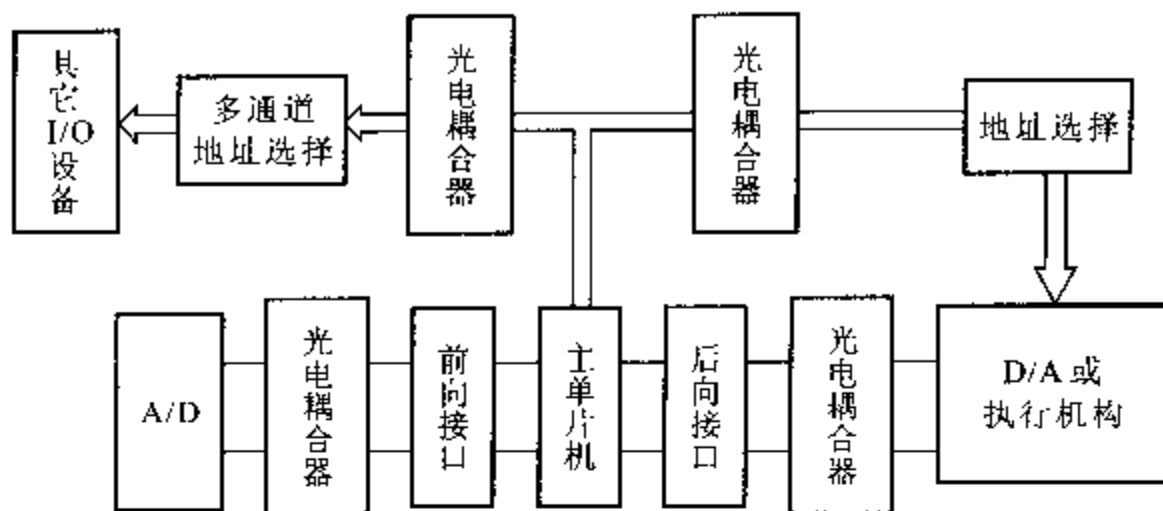


图 14-3 光电耦合隔离的基本配置

14.3.2 光电隔离的实现

1. ADC、DAC 与单片机之间的隔离

对 CPU 数据总线进行隔离是一种十分理想的方法,全部 I/O 端口均被隔离。但是,由于在 CPU 数据总线上是高速(μs 级)双向传输,这就要求频率响应为 MHz 级的隔离器件,而这种器件目前较难买到,价格较高。因此,这种方法采用的不多。

通常采用下列方法将 ADC、DAC 与单片机之间的电气联系切断。

(1) 对 A/D、D/A 进行模拟隔离

对 A/D、D/A 变换前后的模拟信号进行隔离,是常用的一种方法。通常采用隔离放大器对模拟量进行隔离。但所用的隔离型放大器必须满足 A/D、D/A 变换的精度和线性度要求。例如,如果对 12 位 A/D、D/A 变换器进行隔离,其隔离放大器要达到 13 位,甚至 14 位精度,如此高精度的隔离放大器,价格昂贵。

(2) 在 I/O 与 A/D、D/A 之间进行数字隔离

这种方案最经济,也称数字隔离。A/D 变换时,先将模拟量变为数字量,对数字量进行隔离,然后再送入单片机。D/A 变换时,先将数字量进行隔离,然后进行 D/A 变换。这种方法的优点是方便、可靠、廉价,不影响 A/D、D/A 的精度和线性度。缺点是速度不高。如果用廉价的光电隔离器件,最大转换速度约为每秒 3 000~5 000 点,这对于一般工业测控对象(如温度、湿度、压力等)已能满足要求。

图 14-4 所示是实现数字隔离的一个例子。该例将输出的数字量经锁存器锁存后,驱动光电隔离器,经光电隔离之后的数字量被送到 D/A 变换器。但要注意的,现场电源 $F+5\text{V}$,现场地 FGND 和系统电源 $S+5\text{V}$ 及系统地 SGND,必须分别由 2 个隔离电源供电。还应指出的是,光电隔离器件的数量不能太多,由于光电隔离器件的发光二极管与受光三极管之间存在分布电容。当

数量较多时,必须考虑将并联输出改为串联输出的方式,这样可使光电器件大大减少,且保持很高的抗干扰能力,但传送速度下降了。

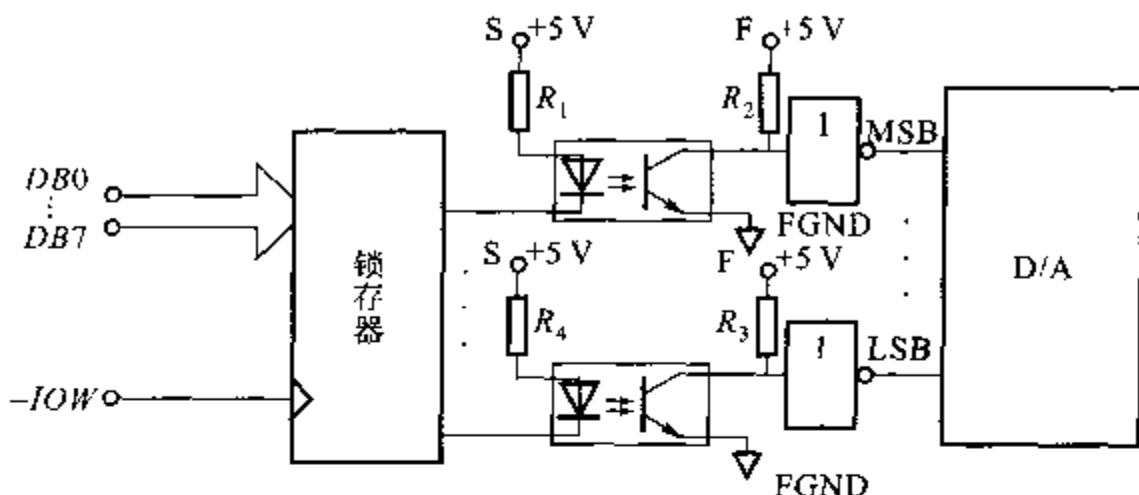


图 14-4 数字隔离原理图

2. 开关量隔离

常用的开关量隔离器件有继电器、光电隔离器、光电隔离固态继电器(SSR)。用继电器对开关量进行隔离时,要考虑到继电器线包的反电动势的影响,驱动电路的器件必须能耐高压。为了吸收继电器线包的反电动势,通常在线包两端并联 1 个二极管。其触点并联 1 个消火花电容器,容量可在 $0.1 \sim 0.047 \mu\text{F}$ 之间选择,耐压视负荷电压而定。

对于开关量的输入,一般用电流传输的方法。此方法抗干扰能力强,如图 14-5 所示。 R_1 为限流电阻, D_1 、 R_2 为保护二极管和保护电阻。当外部开关闭合时,由电源 E 产生电流,使光电二极管导通,采用不同的 R_1 、 R_2 值以保证良好的抗干扰能力。

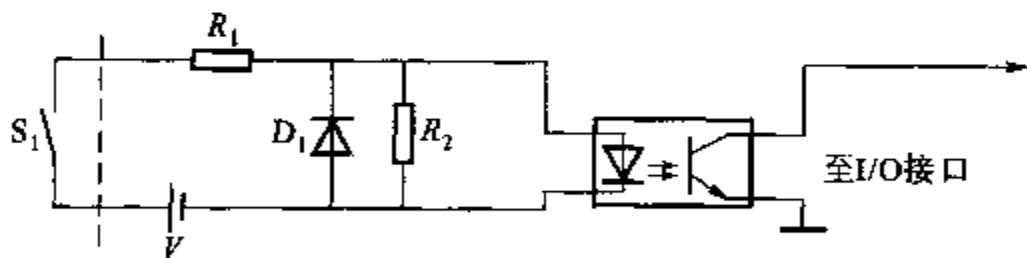


图 14-5 开关量的电流传输原理图

固态继电器代替机械触点的继电器是十分优越的。固态继电器是将发光二极管与晶闸管封装在一起的一种新型器件。当发光二极管导通时,晶闸管被触发而接通电路。固态继电器视触发方式不同,可分为过零触发与非过零触发两大类。过零触发的固态继电器,本身几乎不产生干扰,这对单片机控制是十分有利的,但造价是一般继电器的 5~10 倍。

14.4 空间干扰及抗干扰措施

空间干扰主要指电磁场在线路、导线、壳体上的辐射、吸收和解调。干扰来自应用系统的内部和外部,市电电源线是无线电波的媒介,而在电网中有脉冲源工作时,它又是辐射天线,因而任一线路、导线、壳体等在空间均同时存在辐射、接收、调制。

在现场解决空间干扰时,首先要正确判断是否是空间干扰,可在系统供电电源入口处接入 WRY 型微机干扰抑制器,观察干扰现象是否继续存在,如干扰现象继续存在则可认为是空间干扰。空间干扰不一定是来自系统外部,空间干扰的抗干扰设计主要是地线系统设计,系统的屏蔽与布局设计。

14.4.1 接地技术

1. 接地种类

有两大类接地。一类是为人身或设备安全目的,而把设备的外壳接地,这称之为外壳接地或安全接地;另外一类接地是为电路工作提供一个公共的电位参考点,这种地称为工作接地。

(1) 外壳接地

外壳接地是真正的与大地连接,以使漏到机壳的电荷能及时泄放到地球上,这样才能确保人身和设备的安全。外壳接地的接地电阻应当尽可能低,因此在材料及施工方面均有一定的要求。外壳接地是十分重要的,但实际上往往又为人们所忽视。

(2) 工作接地

工作接地是为满足电路工作需要而进行的。在许多情况下,工作地不与设备外壳相连,因此工作地的零电位参考点(及工作地)相对地球的大地是浮空的。所以也把工作地称为“浮地”。

2. 接地系统

正确、合理地接地,是单片机应用系统抑制干扰的主要方法。

在单片机应用系统中,前述两大类地按单元电路的性质又可分为以下几种接地:

- ① 数字地(又称逻辑地),为逻辑电路的零电位。
- ② 模拟地,为 A/D 转换、前置放大器或比较器的零电位。
- ③ 功率地,为大的电流网络部件的零电位。

- ④ 信号地,通常为传感器的地。
- ⑤ 小信号前置放大器的地。
- ⑥ 交流地,交流 50 Hz 地线,这种地线是噪声地。
- ⑦ 屏蔽地,为防止静电感应和磁场感应而设置的地。

以上这些地线如何处理,是浮地还是接地? 是一点接地还是多点接地? 这些是单片机测控系统设计、安装、调试中的一个大问题。下面就来讨论它们。

(1) 机壳接地与浮地的比较

全机浮空,即机器各个部分全部与大地浮置起来。这种方法有一定的抗干扰能力,但要求机器与大地的绝缘电阻不能小于 $50\text{ M}\Omega$,且一旦绝缘下降便会带来干扰;另外,浮空容易产生静电,导致干扰。

另一种,就是测控系统的机壳接地,其余部分浮空,如图 14-6 所示。浮空部分应设置必要的屏蔽,例如双层屏蔽浮地或多层屏蔽。这种方法干扰能力强,而且安全可靠,但工艺较复杂。两种方法比较,后者较好,并为越来越多的人所采用。

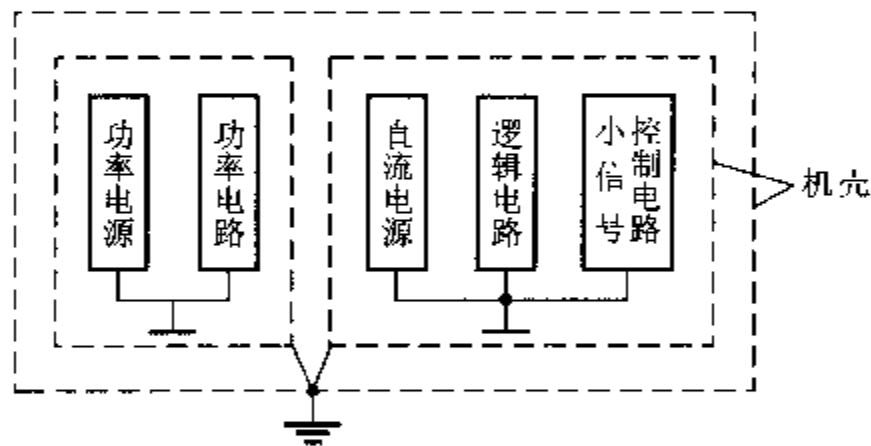


图 14-6 机壳接地

(2) 一点接地与多点接地的应用原则

一般,低频(1 MHz 以下)电路应一点接地,如图 14-7 所示。高频(10 MHz 以上)电路应多点就近接地。因为,在低频电路中,布线和元件间的电感较小,而接地电路形成的环路,受干扰的影响却很大,因此应一点接地;对于高频电路,地线上具有电感,增加了地线阻抗,同时各地线之间还会产生电感耦合,因此应多点就近接地。当频率甚高时,特别是当地线长度等于 $1/4$ 波长的奇数倍时,地线阻抗就会变得很高,这时地线变成了天线,可以向外辐射噪声信号。

单片机测控系统的工作频率大多较低,对它起作用的干扰频率也大都在 1 MHz 以下,故宜采用一点接地。在 1 MHz~100 MHz 之间,如用一点接地,其地线长度不得超过波长的 $1/20$ 。否则应该多点接地。

(3) 交流地与信号地不能共用

因为在一段电源地线的两点间会有数毫伏,甚至几伏电压,对低电平信号电

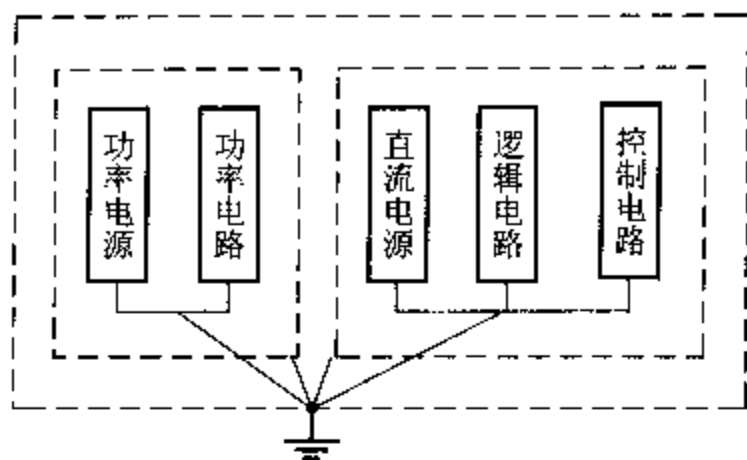


图 14-7 一点接地

路来说,这是一个非常严重的干扰。因此,交流地和信号地不能共用,图 14-8 为一种不正确的接法。

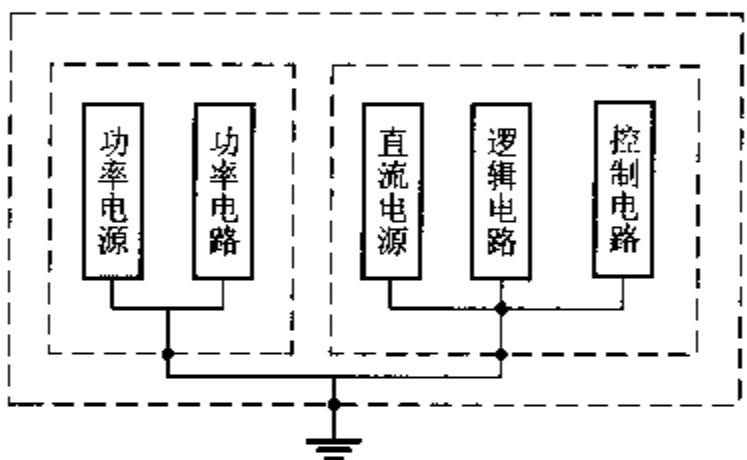


图 14-8 不正确的接地

(4) 数字地和模拟地

数字地通常有很大的噪声而且电平的跳跃会造成很大的电流尖峰。所有的模拟公共导线(地)应该与数字公共导线(地)分开走线,然后只有一点汇在一起。特别是在 ADC 和 DAC 电路中,尤其要注意地线的正确连接,否则转换将不准确。因此 ADC、DAC 和采样保持芯片都提供了独立的模拟地和数字地,他们分别有相应的引脚,必须将所有的模拟地和数字地分别相连,然后模拟(公共)地与数字(公共)地仅在一一点上相连接,在此连接点外,在芯片和其他电路中切不可再有公共点,如图 14-9 所示。

(5) 微弱信号模拟地的接法

A/D 转换器在采集 0~50 mV 微小信号时,模拟地的接法极为重要。为提高抗共模干扰的能力,可用三线采样双层屏蔽浮地技术。这种三线采样双层屏蔽技术是抗共模干扰最有效的方法。

(6) 功率地

这种地线电流大,地线应粗些,且应与小信号分开走线。

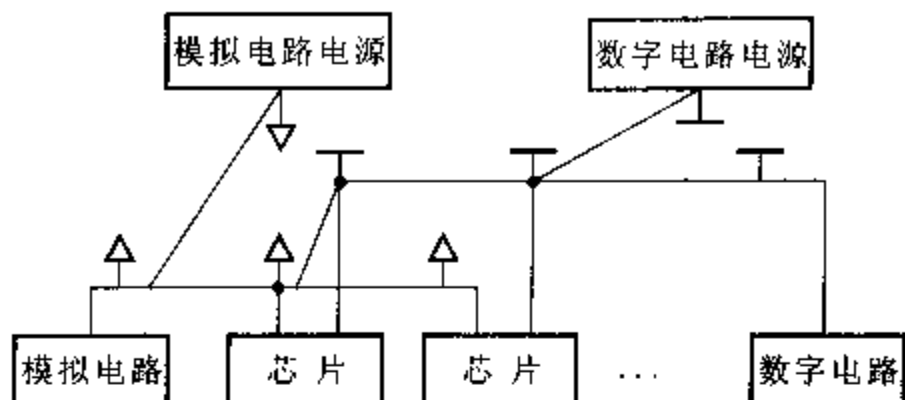


图 14-9 数字地和模拟地正确的地线连接

14.4.2 屏蔽技术

高频电源、交流电源、强电设备产生的电火花甚至雷电,都能产生电磁波,从而成为电磁干扰的噪声源。当距离较近时,电磁波会通过分布电容和电感耦合到信号回路而形成电磁干扰;当距离较远时,电磁波则以辐射形式构成干扰。

单片机使用的振荡器,本身就是一个电磁干扰源,同时它又极易受其他电磁干扰的影响,破坏单片机的正常工作。

屏蔽可分为以下三类:

- (1) 电磁屏蔽,防止电磁场的干扰;
- (2) 磁屏蔽,防止磁场的干扰;
- (3) 电场屏蔽,防止电场的耦合干扰。

电磁屏蔽主要是防止高频电磁波辐射的干扰,以金属板、金属网或金属盒构成的屏蔽体能有效地对付电磁波的干扰。屏蔽体以反射方式和吸收方式来削弱电磁波,从而形成对电磁波的屏蔽作用。

磁场屏蔽是防止电极、变压器、磁铁、线圈等的磁感应和磁耦合,使用高导磁材料做成屏蔽层,使磁路闭合,一般接大地。当屏蔽低频磁场时,选择磁钢、坡莫合金、铁等导磁率高的材料;

而屏蔽高频磁场则应选择铜、铝等导电率高的材料。

电场屏蔽是为了解决分布电容问题,一般是接大地,这主要是指单层屏蔽。对于双层屏蔽,例如双变压器,原边屏蔽接机壳(即接大地),副边屏蔽接到浮地的屏蔽盒。

当一个接地的放大器与一个不接地的信号源相连时,连接电缆的屏蔽层应接到放大器公共端。反之,应接信号源的公共端。高增益放大器的屏蔽层应接到放大器的公共端。

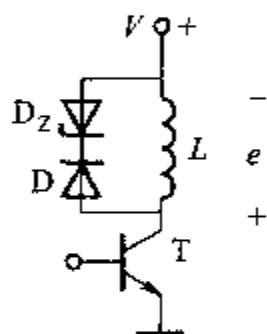
为了有效发挥屏蔽体的屏蔽作用,还应注意屏蔽体的接地问题。为了消除屏

蔽体与内部电路的寄生电容,屏蔽体应按“一点接地”的原则接地。

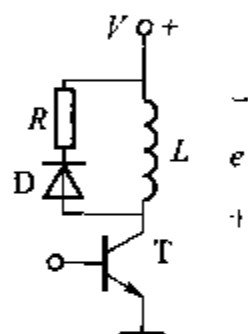
14.5 反电动势干扰的抑制

在单片机的应用系统中,常使用具有较大电感量的元件或设备,诸如继电器、电动机、电磁阀等。当电感回路的电流被切断时,会产生很大的反电动势而形成噪声干扰。这种反电动势甚至可能击穿电路中晶体管之类的器件,反电动势形成的噪声干扰能产生电磁场,对单片机应用系统中的其它电路产生干扰。对于反电动势干扰,可采用如下措施加以抑制:

(1) 如果通过电感线圈的是直流电流,可在线圈两端并联二极管和稳压管,如图14-10(a)所示。



(a) 由二极管和稳压管构成的反电动势抑制电路



(b) 由电阻和二极管制成的反电动势抑制电路

图14-10 反电动势的抑制电路

在稳定工作时,并联支路被二极管D阻断而不起作用;当三极管T由导通变为截止时,在电感线圈两端产生反电动势 e 。此电动势可在并联支路中流通,因此 e 的幅值被限制在稳压管 D_z 的工作电压范围之内,并被很快消耗掉,从而抑制了反电动势的干扰。使用时 D_z 的工作电压应选择得比外加电源高些。

如果把稳压管换为电阻,同样可以达到抑制反电动势的目的,如图14-10(b)所示,因此也适用于直流驱动线圈的电路。在这个电路中,电阻的阻值范围可以从几欧姆到几十欧姆。阻值太小,反电动势衰减得慢;而阻值太大又会增大反电动势的幅值。

(2) 反电动势抑制电路也可由电阻和电容组成,如图14-11所示。适当选择 R 、 C 参数,也能获得较好的耗能效果。这种电路不仅适用于交流驱动的线圈,也适用于直流驱动的线圈。

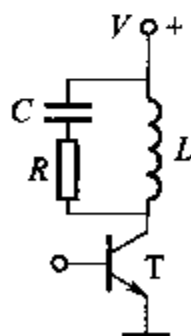


图14-11 由电阻和电容组成的抑制电路

(3) 反电动势抑制电路不但可以接在线圈的两端,也可以接在开关的两端,例如继电器、接触器等部件在操作时,触点开关会产生较大的火花,必须利用 RC 电路加以吸收,如图 14-12 所示。对于图 14-12(b),一般 R 取 $1 \sim 2 \text{ k}\Omega$, C 取 $2.2 \sim 4.7 \mu\text{F}$ 。

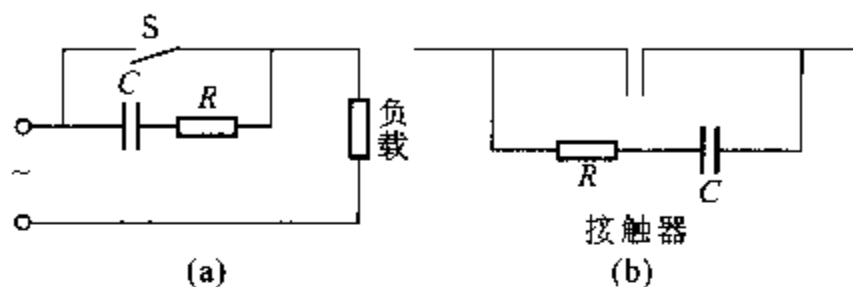


图 14-12 开关两端的反电动势抑制电路

14.6 印制电路板的抗干扰设计

印制电路板(简称印制板)是单片机系统中器件、信号线、电源线的高密度集合体,印制电路板设计的好坏对抗干扰能力影响很大,故印制电路板设计决不单是器件、线路的简单布局安排,还必须符合抗干扰的设计原则。

14.6.1 地线及电源线设计

1. 地线宽度

加粗地线能降低导线电阻,使它能通过 3 倍于印制板上的允许电流。如有可能,地线宽度应在 $2 \sim 3 \text{ mm}$ 以上。

2. 接地线构成闭环路

接地线构成闭环路能明显地提高抗噪声能力。闭环形状能显著地缩短线路的环路,降低线路阻抗,从而减少干扰。但要注意环路所包围的面积越小越好。

3. 印制电路板分区集中并联一点接地

当同一印制板上有多个不同功能的电路时,可将同一功能单元的元器件集中于一点接地,自成独立回路。这就可使地线电流不会流到其它功能单元的回路中去,避免了对其它单元的干扰。与此同时,还应将各功能单元的接地块与主机的电源地相连接,如图 14-13 所示。这种接法称为“分区集中并联一点接地”。为了减小线路阻抗,地线和电源线要采用大面积汇流排。

数字地和模拟地分开设计,在电源处两种地线相连,且地线应尽量加粗。

4. 电源线的布置

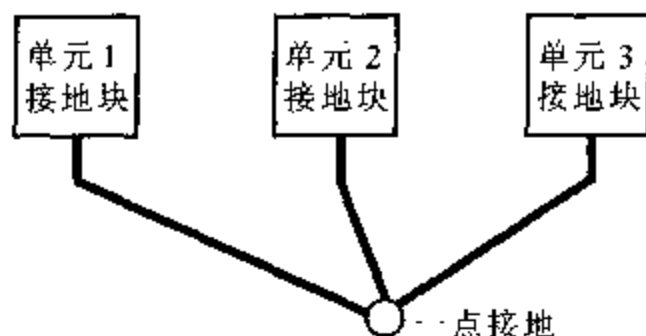


图 14-13 分区集中并联一点接地

电源线除了要根据电流的大小,尽量加粗导体宽度外,还应使电源线、地线的走向与数据传递的方向一致,这将有助于增强抗噪声能力。

14.6.2 去耦电容的配置

印制板上装有多个集成电路,而当其中有些元件耗电很多时,地线上会出现很大的电位差。抑制电位差的方法是在各个集成器件的电源线和地线间分别接入去耦电容,以缩短开关电流的流通过程,降低电阻压降。这是印制板设计的一项常规做法。

1. 电源去耦

电源去耦就是在每个印制板入口外的电源线与地线之间并接去耦电容。并接的电容应为 1 个大容量的电解电容($10\sim 100\ \mu\text{F}$)和 1 个 $0.01\sim 0.1\ \mu\text{F}$ 的非电解电容。我们可以把干扰分解成高频干扰和低频干扰两部分,并接大电容是为了去掉低频干扰成分,并接小电容则是为了去掉高频干扰部分。低频去耦电容用铝或钽电解电容,高频去耦电容采用自身电感小的云母或陶瓷电容。

2. 集成芯片去耦

每个集成芯片都应安置 1 个 $0.1\ \mu\text{F}$ 的陶瓷电容器,安装每个芯片的去耦电容时,必须将去耦电容安装在该集成芯片的 V_{CC} 引脚和 GND 线之间,否则便失去了抗干扰作用。如遇到印制电路板空隙小装不下时,可每 $4\sim 10$ 个芯片安置 1 个 $1\sim 10\ \mu\text{F}$ 的限噪声用的钽电容器。这种电容器的高频阻抗特别小,在 $500\ \text{Hz}\sim 200\ \text{MHz}$ 范围内阻抗小于 $1\ \Omega$,而且漏电流很小($0.5\ \mu\text{A}$ 以下)。

对于抗噪声能力弱,关断电流大的器件和 ROM、RAM 存储器,应在芯片的电源线 V_{CC} 和地线 GND 间直接接入去耦电容。

14.6.3 印制板布线的抗干扰设计

印制板的布线方法对抗干扰性能有直接影响。前面已经间接地介绍了一些布线原则,对于没有介绍到的一些布线原则,下面予以补充说明。

(1) 如果印制板上逻辑电路的工作速度低于 TTL 的速度,导线条的形状无什么特别要求;若工作速度较高,如使用高速逻辑器件时,用作导线的铜箔在 90° 转弯处的导线阻抗不连续,可能导致反射干扰的发生,所以宜采用图 14-14 中右方的形状,把弯成 90° 的导线改成 45° ,这将有助于减少反射干扰的发生。

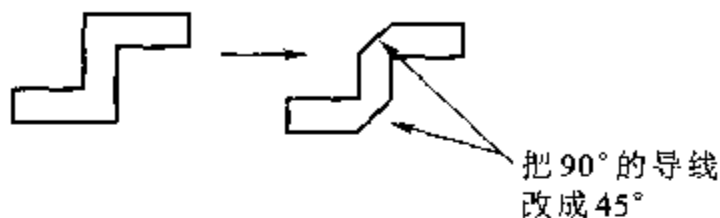


图 14-14 90° 转弯处的导线改成 45°

(2) 不要在印制板中留下无用的空白铜箔层,因为它们可以充当发射天线或接收天线,可把它们就近接地。

(3) 双面布线的印制板,应使双面的线条垂直交叉,以减少磁场耦合,有利于抑制干扰。

(4) 导线间距离要尽量加大。对于信号回路,印制铜箔条的间隔距离要有足够的尺寸,而且这个距离要随信号频率的升高而加大,尤其是频率极高或脉冲前沿十分陡峭的情况更要注意,只有这样才能减小导线之间分布电容的影响。

(5) 高电压或大电流线路对其它线路更容易形成干扰,低电平或小电流信号线路容易受到感应干扰,布线时应使两者尽量相互远离,避免平行铺设,采用屏蔽等措施。

(6) 所有线路尽量沿直流地铺设,尽量避免沿交流地铺设。

(7) 电源线的布线除了要尽量加粗导体宽度外,采取使电源线、地线的走向与数据传递的方向一致,将有助于增强抗噪声能力。

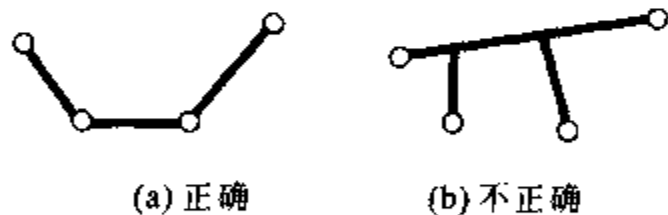


图 14-15 走线不要有分支

(8) 走线不要有分支,这可避免在传输高频信号时发生反射干扰或谐波干扰,如图 14-15 所示。

14.7 软件抗干扰措施

单片机系统在噪声环境下运行,除了前面介绍的各种抗干扰的措施外,还可采用软件来增强系统的抗干扰能力。本节介绍几种常用软件抗干扰的方法。

14.7.1 软件抗干扰的一般方法

软件抗干扰技术是当系统受干扰后使系统恢复正常运行或输入信号受干扰后去伪求真的一种辅助方法。因此软件抗干扰是被动措施,而硬件抗干扰是主动措施。但由于软件设计灵活,节省硬件资源,所以软件抗干扰技术已得到较为广泛的应用。软件抗干扰技术所研究的主要内容如下:

(1) 软件滤波、采用软件的方法抑制叠加在输入信号上的噪声的影响,可以通过软件滤波剔除虚假信号,求取真值。

(2) 开关量的输入/输出抗干扰设计。可采用对开关量输入信号重复检测,对开关量输出 I/O 数据刷新的方法。

(3) 由于 CPU 受到干扰,程序计数器 PC 的状态被破坏,导致程序从一个区域跳转到另一个区域,或者程序在地址空间内“乱飞”,或者进入死循环。因此必须尽可能早的发现并采取措施,把程序纳入正轨。为使“乱飞”的程序被拦截或程序摆脱死循环可采用指令冗余、软件陷阱或“看门狗”等技术。

下面介绍上述的各种软件抗干扰技术。

14.7.2 软件滤波

对于实时数据采集系统,为了消除传感器通道中的干扰信号,硬件上常采用模拟滤波器对信号实现频率滤波。同样,采用软件也可以实现滤波,完成模拟滤波器类似的功能,这就是数字滤波。

1. 算术平均滤波法

算术平均滤波法就是对 1 点数据连续取 n 个值进行采样,然后算术平均。这种方法适用于对一般具有随机干扰的信号进行滤波。这样信号的特点是有 1 个平均值,信号在某一数值范围附近上下波动。这种滤波法当 n 值较大时,信号的平滑度高,但是灵敏度低;当 n 值较小时,平滑度低,但灵敏度高。应视具体情况选取 n ,以便既节约时间,又滤波效果好。对于一般流量测量,通常取 $n=12$;若为压力,则取 $n=4$ 。一般情况下 n 取 3~5 次平均即可。

读者可根据上述设计思想,设计出算术平均滤波法的子程序 AVGFI。

2. 滑动平均滤波法

上面介绍的算术平均滤波法,每计算 1 次数据需要测量 n 次。对于测量速度较慢或要求数据计算速度较快的实时控制系统,上述方法无法使用。下面介绍 1 种只需测量 1 次,就能得到当前算术平均值的方法——滑动平均滤波法。

滑动平均滤波法是把 n 个采样值看成 1 个队列,队列的长度为 n ,每进行 1 次

采样,就把采样值放入队尾,而扔掉原来队首的 1 个采样值.这样在队列中始终有 n 个“最新”采样值.对队列中的 n 个采样值进行平均,就可以得到新的滤波值。

滑动平均滤波法对周期性干扰有良好的抑制作用,平滑度高,灵敏度低;但对偶然出现的脉冲性干扰的抑制作用差,不易消除由于脉冲干扰引起的采样值的偏差。因此它不适用于脉冲干扰比较严重的场合,而适用于高频振荡系统。通常观察不同 n 值下滑动平均的输出响应来选取 n 值,以便既少占有时间,又能达到最好滤波效果,其工程经验值为:

参 数	温 度	压 力	流 量	液 面
n 值	1~4	4	12	1~12

下例为滑动平均滤波法的参考程序。

例 14-1 假定 n 个双字节型采样值,30H 单元为采样队列内存单元首地址, n 个采样值之和不大于 16 位。新的采样值存于 2EH、2FH 单元,滤波值存于 50H、51H 单元。AVGFIL 为本程序调用的算术平均滤波子程序。程序如下:

```
SAVGFIL:  MOV    R2,#n-1          ;采样个数
           MOV    R0,#32H         ;队列单元首地址
           MOV    R1,# 33H
LOOP:     MOV    A,@R0            ;移动低字节
           DEC    R0
           DEC    R0
           MOV    @R0,A
           MOV    A,R0            ;修改低字节地址
           ADD    A,#04H
           MOV    R0,A
           MOV    A,@R1          ;移动高字节
           DEC    R1
           DEC    R1
           MOV    @R1,A
           MOV    A,R1           ;修改高字节地址
           ADD    A,#04H
           MOV    R1,A
           DJNZ   R2,LOOP
           MOV    @R0,2EH        ;存新的采样值
           MOV    @R1,2FH
           ACALL  AVGFIL         ;调用求算术平均值子程序 AVGFIL,
```

;假设已存在

RET

3. 中位值滤波法

中位值滤波法就是对某一被测参数接连采样 n 次(一般 n 取奇数),然后把 n 次采样值按大小排列,取中间值为本次采样值。中位值滤波能有效地克服因偶然因素引起的波动干扰。对温度、液位等变化缓慢的被测参数采用此法能收到良好的滤波效果。但对于流量、速度等快速变化的参数一般不宜采用中位值滤波法。

中位值滤波程序设计的实质是,首先把 n 个采样值从小到大或从大到小进行排序,然后再取中间值。 n 个数据按大小顺序排队的具体做法是采用“冒泡法”(排序程序设计见第4章)进行比较,直到最大数沉底为止。然后再重新进行比较,把次大值放到 $n-1$ 位,依此类推,则可将 n 个数从小到大顺序排列。

例 14-2 设采样值从 8 位 A/D 转换器输入 5 次,存放在 SAMP 为首地址的内存单元中。采用中位值滤波。参考程序如下:

```
SAM EQU 30H
ORG 1000H
INTER: MOV R2, #04H      ;置最大循环次数
SORT:  MOV A, R2          ;小循环次数 → (R3)
        MOV R3, A
        MOV R0, #SAMP     ;采样数据首地址 → (R0)
LOOP:  MOV A, @R0
        INC R0
        MOV R1, A
        CLR C
        SUBB A, @R0
        MOV A, R1
        JC DONE
        MOV A, @R0        ;((R0)) → ((R0)+1)
        DEC R0
        XCH A, @R0
        INC R0
        MOV @R0, A
DONE:  DJNZ R3, LOOP      ;R3≠0,小循环继续进行
        DJNZ R2, SORT    ;R2≠0,大循环继续进行
        INC R0
        MOV A, @R0
        RET
```

4. 去极值平均滤波法

前面介绍的算术平均与滑动平均滤波法,在脉冲干扰比较严重的场合,则干扰将会“平均”到结果中去,故上述两种平均值法不易消除由于脉冲干扰而引起的误差。这时可采用去极值平均滤波法。

去极值平均滤波法的思想是:连续采样 n 次后累加求和,同时找出其中的最大值与最小值,再从累加和中减去最大值和最小值,按 $n-2$ 个采样值求平均,即可得到有效采样值。这类似于体育比赛中的去掉最高分和最低分,再求平均分的评分办法。

为使平均滤波算法简单, $n-2$ 应为 2,4,6,8 或 16,故 n 常取 4,6,8,10 或 18。具体做法有 2 种:对于快变参数,先连续采样 n 次,然后再处理,但要在 RAM 中开辟出 n 个数据的暂存区;对于慢变参数,可一边采样,一边处理,而不必在 RAM 中开辟数据暂存区。实践中,为了加快测量速度,一般 n 取 4。

例 14-3 以 $n=4$ 为例,即连续进行 4 次数据采样,去掉其中最大值和最小值,然后求剩下 2 个数据的平均值。R2R3 存最大值,R4R5 存最小值,R6R7 存放累加和及最后结果。当然,连续采样不只限 4 次,可以进行任意次,这时,只需改变 R0 中的数值。

参考程序如下:

```

DEMAXFL: CLR      A
           MOV      R2,A          ;0 → 最大值寄存器 R2R3
           MOV      R3,A
           MOV      R6,A          ;0 → 累加和寄存器 R6R7
           MOV      R7,A
           MOV      R4,#3FH       ;3FFFH → 最小值寄存器 R4R5
           MOV      R5,#0FFH
           MOV      R0,#4H

DAV1:     LACALL   RDXP           ;调采样子程序 RDXP,数字量从 A/D
           ;读入 B,A 中
           MOV      R1,A          ;采样值低位暂存 R1,高位在 B
           ADD      A,R7
           MOV      R7,A          ;低位加到 R7
           MOV      A,B
           ADDC     A,R6
           MOV      R6,A          ;高位加到 R6,(R6R7)+(BA)
           ;→ R6R7
           CLR      C
           MOV      A,R3
           SUBB     A,R1

```

```

MOV    A,R2
SUBB   A,B
JNC    DAV2      ;输入值 > (R2R3)?
MOV    A,R1
MOV    R3,A
MOV    R2,B      ;输入值 → R2R3
DAV2:  CLR    C
MOV    A,R1
SUBB   A,R5
MOV    A,B
SUBB   A,R4
JNC    DAV3      ;输入值 < (R4R5)?
MOV    A,R1
MOV    R5,A      ;输入值 → R4R5
MOV    R4,B
DAV3:  DJNZ   R0,DAV1 ;n-1-0?
CLR    C
MOV    A,R7      ;
SUBB   A,R3      ;
XCH    A,R6      ;
SUBB   A,R2      ;
XCH    A,R7      ;
SUBB   A,R5      ;
XCH    A,R6      ;
SUBB   A,R1      ;
CLR    C          ;
RRC    A          ;
XCH    A,R6      ;
RRC    A          ;
MOV    R7,A      ;
RETI

```

n 个采样值的累加和减去最大值和最小值, n=4

剩下的采样值求平均(除以 2)

14.7.3 开关量输入/输出软件抗干扰设计

如果干扰只作用在系统的 I/O 通道上,则可用如下方法减小或消除其干扰。

1. 开关量输入软件抗干扰措施

干扰信号多呈毛刺状,作用时间短。利用这一特点,我们采集某一状态信号

时,可多次重复采集,直到连续两次或多次采集结果完全一致时才视为有效。若相邻的检测内容不一致,或多次检测结果不一致,则是伪输入信号。可停止采集,给出报警信号。由于状态信号主要来自各类开关型状态传感器,对这些信号采集不能用多次平均方法,必须绝对一致才行。

在满足实时性要求的前提下,如果在各次采集状态信号之间增加一段延时,效果就会更好,就能对抗较宽时间范围的干扰。延时时间在 $10 \sim 100 \mu\text{s}$ 左右。对于每次采集的最高次数限制和连续相同次数均可按实际情况适当调整。

2. 开关量输出软件抗干扰措施

在单片机系统的输出信号中,有很多是驱动各种警报装置,各种电磁装置等的状态驱动信号。对这类信号的抗干扰有效输出方法是重复输出同一个数据,只要有可能,重复周期应尽量短。外部设备接收到一个被干扰的错误信息后,还来不及做出有效的反应,一个正确的输出信息又到来,就可以及时地防止错误动作的产生。

在执行输出功能时,应该将有关输出芯片的状态也一并重复设置。例如 8155 芯片和 8255 芯片常用来扩展输入输出功能,很多外设通过它们来获得单片机的控制信息。这类芯片均应进行初始化编程,以明确各端口的功能。由于干扰的作用,有可能在无意中将芯片的编程方式改变。为了确保输出功能正确实现,输出功能模块在执行具体的数据输出之前,应该先执行对芯片的初始化编程指令,再输出有关数据。

14.7.4 指令冗余及软件陷阱

由于单片机系统受到干扰而使运行程序发生混乱、导致程序乱飞或陷入死循环时,应及时采取将程序纳入正轨的措施,如指令冗余、软件陷阱等。

1. 指令冗余

CPU 取指令是先取操作码,再取操作码数。当单片机系统受干扰出现错误时,程序便脱离正常轨道乱飞。当乱飞到某双字节指令,若取指令时刻落在操作数上,误将操作数当作操作码,程序就有可能出错。若乱飞到三字节指令,出错几率更大。在关键的地方人为地插入一些单字节指令或将有效单字节指令重写称为指令冗余。

指令冗余无疑会降低系统的效率,通常是在双字节指令和三字节指令后插入 2 B 以上“NOP”指令,可保护其后的指令不被拆散。因此,常在一些对程序流向起决定作用的指令之前插入 2 条 NOP 指令,此类指令有: RET、RETI、ACALL、LCALL、SJMP、AJMP、LJMP、JZ、JNZ、JC、JNC、JB、JNB、JBC、CJNE、DJNZ 等,以保证乱飞的程序被迅速纳入正轨。在某些对系统工作状态至关重

要的指令(如 SETB EA 之类)前也可插入 2 条 NOP 指令。

指令冗余措施可以减少程序乱飞的次数,使其被很快纳入程序轨道,但这并不能保证在失控期间不干坏事,更不能保证程序被纳入正常轨道后就太平无事了。程序的运行事实上已经偏离了正常顺序,有可能做着它现在不该做的事情。解决这个问题还必须采用软件容错技术(限于篇幅,本书不作介绍),使系统的误动作减少,并消灭重大误动作。

2. 软件陷阱

所谓软件陷阱,就是一条引导指令,强行将乱飞的程序引向一个指定的地址,在那里有一段专门对程序出错进行处理的程序。如果我们把这段程序的入口标号称为 ERR 的话,软件陷阱即为一条 LJMP ERR 指令。为加强其捕捉效果,一般还在它前面加 2 条 NOP 指令。

NOP

NOP

LJMP ERR

软件陷阱一般安排在下列 4 种地方:

(1) 未使用的中断向量区:0003H~002FH

当干扰使未使用的中断开放,并激活这些中断时,就会进一步引起混乱。如果我们在这些地方布上陷阱,就能及时捕捉到错误中断。例如,系统共使用三个中断: $\overline{\text{INT0}}$ 、T0、T1,它们的中断子程序分别为 PGINT0、PGT0、PGT1,建议按如下方式来设置中断向量区:

```
                                ORG 000H
0000  START:  LJMP  MAIN      ;引向主程序入口
0003                LJMP  PGINT0 ; $\overline{\text{INT0}}$ 中断正常入口
0006                NOP                      ;
0007                NOP                      ; } 冗余和陷阱
0008                LJMP  ERR                ; }
000B                LJMP  PGT0              ;T0 中断正常入口
0016                NOP                      ;
0017                NOP                      ; } 冗余和陷阱
0018                LJMP  ERR                ; }
001B                LJMP  PGT1              ;T1 中断正常入口
001E                NOP                      ;
001F                NOP                      ; } 冗余和陷阱
0020                LJMP  ERR                ; }
0023                LJMP  ERR                ;未使用串行口中断,设陷阱
```

0026	NOP	;	} 冗余和陷阱
0027	NOP	;	
0028	LJMP ERR	;	
	:		
0030	MAIN:	:	;主程序
	:		

从 0030H 开始再编写正式程序。

(2) 未使用的 EPROM 空间

所编写的程序,很少有将 EPROM 全部用完的情况。对于剩余 EPROM 空间,一般均维持原状态(FFH),对于 MCS-51 指令系统来讲,FFH 是一条单字节指令(MOV R7,A)程序乱飞到这一区域后将顺流而下,不再跳跃(除非受到新的干扰)。这时只要每隔一段设置一个陷阱,就一定能捕捉到乱飞的程序。

软件陷阱一定要指向处理过程 ERR。我们可以将 ERR 安排在 0030H 开始的地方,这样我们就可以用 00 00 02 00 30 5B 作为陷阱来填充 EPROM 中的未使用空间,或者每隔 1 段设置 1 个陷阱(02 00 30),其它单元保持 FFH 不变。

(3) 表格

有两类表格,一类是数据表格,供 MOVC A, @A+PC 指令或 MOVC A, @A+DPTR 指令使用,其内容完全不是指令。另一类是跳转表格,供 JMP @A+DPTR 指令使用,其内容为一系列的三字节指令 LJMP 或两字节指令 AJMP。由于表格内容和检索值有一一对应关系,在表格中间安排陷阱将会破毁其连续性和对应关系,我们只能在表格的最后安排五字节陷阱(NOP, NOP, LJMP ERR)。由于表格区一般较长,安排在最后的陷阱不能保证一定捕捉住乱飞的程序,有可能在中途再次飞走。这时只好指望别处的陷阱或冗余指令来制服它了。

(4) 程序区

程序区是由一串执行指令构成的,我们不能在这些指令串中间任意安排陷阱,否则影响正常执行程序。但是,在这些指令串之间常有一些断裂点,正常执行的程序到此便不会继续往下执行了,这类指令有 LJMP、SJMP、AJMP、RET、RETI。这时 PC 的值应发生正常跳变。如果还要顺次往下执行,必然就出错了。我们在这种地方安排陷阱之后,就能有效地捕捉住它,而又不影响正常执行的程序流程。例如,在一个根据累加器的正、负、零情况进行三支的程序中,软件陷阱的安置方式如下:

JNZ L1	;A 中内容非零,跳 L1 程序段
...	;A 中内容为零的处理程序段
AJMP L3	;断裂点

```

NOP
NOP
LJMP ERR
L1: JB ACC.7,L2
...
LJMP L3
NOP
NOP
LJMP ERR
L2: ...
L3: MOV A,R2
RET
NOP
NOP
LJMP ERR

```

; 冗余指令
 ; 软件陷阱
 : 断裂点
 ; 冗余指令
 ; 软件陷阱
 ; 取结果
 ; 冗余指令
 ; 软件陷阱

由于软件陷阱都安排在正常程序执行不到的地方,故不影响程序执行效率。在当前 EPROM 容量不成问题的条件下,还是多多设置陷阱有益。

14.8 “看门狗”技术和掉电保护

本节讨论如何解决程序陷入死循环问题和单片机系统电源掉电数据的保护问题。

14.8.1 “看门狗”和掉电保护的实现

MCS-51 的 PC 受到干扰而失控,引起程序乱飞,可能会使程序陷入死循环。指令冗余和软件陷阱技术不能使失控的程序摆脱死循环的困境,这时系统将完全瘫痪。如果操作人员在场,可按下人工复位按钮,强制系统复位。但操作人员不可能一直监视着系统,即使监视着系统,也往往是在引起不良后果之后才进行人工复位。能不能不要人来监视,就能使系统摆脱死循环,重新执行正常的程序呢?这可采用“看门狗”(Watchdog)技术来解决这一问题。

“看门狗”技术就是使用一个计数器来不断计数,监视程序循环运行。若发现时间超过已知的循环设定时间,则认为系统陷入了死循环,这时计数器溢出,然后强迫系统复位,在复位入口 0000H 处安排一段出错处理程序,使系统运行进入正轨。

另外,在单片机系统运行时,有可能会发生电源掉电的意外情况,一些重要的数据可能丢失。这时要求系统应首先检测到电源的变化,然后通过切换电路把备用电池接入系统,以保护 RAM 中的数据不丢失。

目前看门狗电路和掉电保护电路,都已经集成在一片微处理器监控器芯片中。因此, MCS-51 只需扩展一片微处理器监控器芯片即可。这类芯片集成化程度高,功能齐全,具有广阔的应用前景。在单片机应用系统中使用微处理器监控器芯片,可以大大提高单片机应用系统的抗干扰能力和可靠性。本节介绍美国 MAXIM 公司的微处理器监控器芯片 MAX690A/MAX692A 以及与 MCS-51 的接口设计。与 MAX690A/MAX692A 功能相类似的芯片,还有 MAX703~MAX709、MAX813L、MAX791 等。

14.8.2 微处理器监控器 MAX690A 简介

1. MAX690A/MAX692A 简介

MAX690A/MAX692A 是美国 MAXIM 公司的产品,具有以下性能:

(1) 具有看门狗电路,进入死循环的时间间隔超过 1.6 s 时,将产生 1 个复位输出。

(2) 具有备用电池切换电路,备用电池可供电给 RAM(CMOS)芯片、 μ P(CMOS)或其他低功耗逻辑电路。

(3) 在微处理器上电、掉电及低压供电时,产生 1 个复位输出信号。

(4) 可用于低电平检测。

MAX690A 与 MAX692A 的不同点在于复位门限电平不同。当电源电压低于 4.65 V 时,MAX690A 产生 1 个复位脉冲;MAX692A 是在电源电压低于 4.4 V 时,才产生 1 个复位脉冲。

2. 主要电气参数

MAX690A 的主要电气参数为:

- (1) 工作电压 V_{CC} : 1.2~5.5 V;
- (2) 静态电流: 200 μ A(典型值);
- (3) 备用电池方式静态电流: 5 μ A;
- (4) 输出电压 V_{OUT} ($I_{OUT}=50$ mA): $V_{CC}-0.25$ V;
- (5) 复位脉冲宽度 T_{RS} : 200 ms(典型值);
- (6) 看门狗定时时间: 1.6 s(典型值);
- (7) 复位门限电平: MAX690A 4.65 V; MAX692A 4.40 V。

MAX690A/MAX692A 封装及引脚如图 14-16 所示。

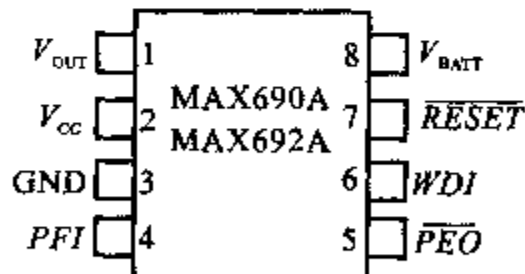


图 14-16 MAX690A/
MAX692A 封装图

3. 电路工作原理

MAX690A/MAX692A 内部原理框图如图 14-17 所示,包括复位电路、看门狗电路、掉电比较和备用电池切换电路四部分。

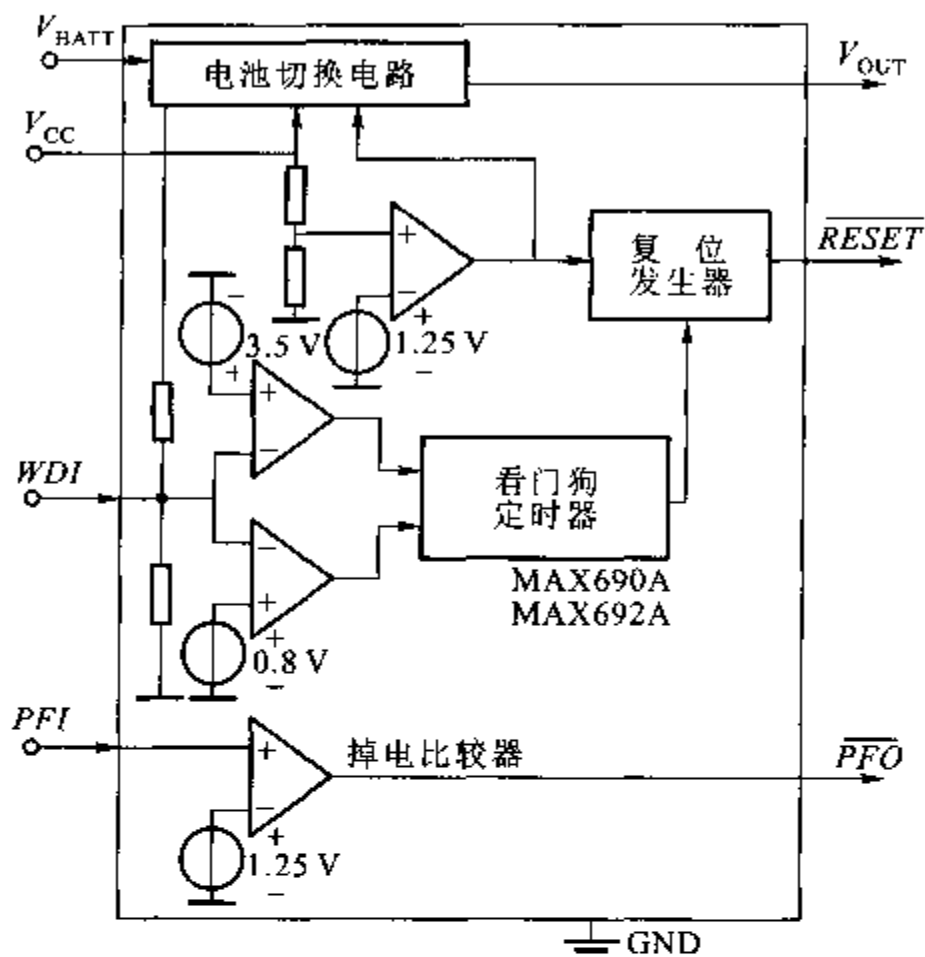


图 14-17 MAX690A/MAX692A 原理框图

(1) 复位电路

微处理器在上电、掉电及低压供电时,监控器产生复位脉冲信号,这可保证微处理器实现上电自动复位;当供电电压过低时,防止 CPU 失控。

电源电压 V_{CC} 上升到 1 V 时, \overline{RESET} 引脚变为低电平。随着 V_{CC} 的继续提高, \overline{RESET} 一直保持低电平。当 V_{CC} 高于复位门限电平时, \overline{RESET} 并不马上变为高电平,而是要滞后 1 个复位脉冲宽度(约 200 ms)后,再变为高电平。

当 V_{CC} 低于复位门限电平时, \overline{RESET} 引脚马上变为低电平;即使以后 V_{CC} 恢复且高于复位门限电平, \overline{RESET} 也不马上变为高电平,而是要延时 1 个复位脉冲宽度。

掉电时, V_{CC} 只要低于复位门限电平, \overline{RESET} 立即变为低电平。

(2) 看门狗电路

看门狗电路是计数器式定时电路。在 WDI 端输入 1 个脉冲(TTL 电平,宽度可小至 50 ns),定时器开始计数。若 WDI 引脚悬空或接至高阻态输出的缓冲器上,定时器则停止计数,并且清零。当定时器启动后,若在 1.6 s 内没有向 WDI 端输入脉冲,监控器将输出 1 个复位信号,引脚 \overline{RESET} 变低电平,同时定时器被清

零。只要 $\overline{\text{RESET}}$ 为低电平,定时器将一直停止工作。

看门狗电路用于使 CPU 摆脱死循环状态。

(3) 掉电比较器

掉电比较器可用于低电平的检测,若电压过低,则发出 1 个低电平信号($\overline{\text{PFO}}$ 端)。掉电比较器是 1 个完全独立的电路,也可以用来完成其他功能。PFI 输入端的电压与内部 1 个 1.25 V 的基准电压相比较,当 PFI 端电压低于 1.25 V 时, $\overline{\text{PFO}}$ 端变为低电平。

(4) 备用电池切换电路

当系统掉电或供电电压过低时,有时需要保存 RAM 中的内容,在 V_{BATT} 端接上备用电池,MAX690A 就会在掉电时自动为 RAM 提供电源。切换电路原理如图 14-18 所示。图中开关 $S_1 \sim S_4$ (为 $5\ \Omega$ 的 PMOS 功率开关)的状态受引脚 V_{CC} 和 V_{BATT} 控制,如表 14-1 所示。

由表 14-1 和图 14-18 可知,当 V_{CC} 高于复位门限电平,或低于复位门限电平但高于 V_{BATT} 时, V_{OUT} 端由 V_{CC} 供电;当 V_{CC} 低于复位门限电平,且又低于 V_{BATT} 时, V_{OUT} 端由 V_{BATT} 供电。

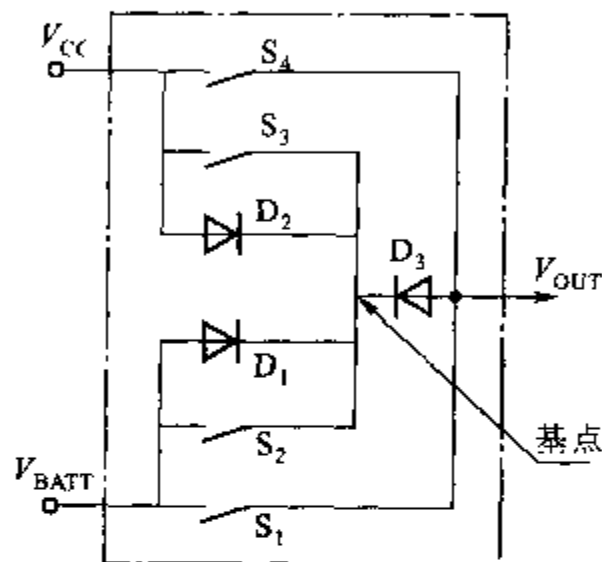


图 14-18 备用电池切换原理图

表 14-1 开关状态表

条 件	S_1/S_2	S_3/S_4
$V_{\text{CC}} > \text{复位门限电平}$	断开	闭合
$V_{\text{CC}} < \text{复位门限电平}$ 且 $V_{\text{CC}} > V_{\text{BATT}}$	断开	闭合
$V_{\text{CC}} < \text{复位门限电平}$ 且 $V_{\text{CC}} < V_{\text{BATT}}$	闭合	断开

当 V_{OUT} 端由 V_{BATT} 供电时,芯片进入备用电池工作方式。当 V_{CC} 稍低于 V_{BATT} 时, V_{BATT} 处流出电流典型值为 $30\ \mu\text{A}$;当 V_{CC} 低于 V_{BATT} 电压 1 V 时,内部电池转换比较器停止工作,电源电流降至 $1\ \mu\text{A}$ 。

在备用电池工作方式下,各输入、输出引脚状态为:掉电比较器不工作, $\overline{\text{PFO}}$ 为低电平, $\overline{\text{RESET}}$ 为低电平,看门狗定时器不工作。

14.8.3 MCS-51 与微处理器监控器 MAX690A/MAX692A 的接口

MCS-51 与 MAX690A 自动监控的接口电路如图 14-19 所示。合理设计

R_1 、 R_2 的值,使得 +5 V 电压跌落到某电压值(如 4.5 V),PFI 端的输入电压低于 1.25 V 时, \overline{PFO} 输出低电平,作为 CPU 的中断输入信号通知单片机,使之进行一些必要的处理(如保存某些重要数据,关掉 LED 显示器等)。 R_1 、 R_2 选取说明如下:

$$\frac{R_1}{R_1 + R_2} = \frac{1.25 \text{ V}}{4.5 \text{ V}} = \frac{1}{3.6}$$

可取 $R_1 = 1 \text{ k}\Omega$, $R_2 = 3.6 \text{ k}\Omega$ 。当 +5 V 电压跌落到 4.5 V 时, $V_R = 1.25 \text{ V}$,再继续跌落, \overline{PFO} 引脚便为低电平。

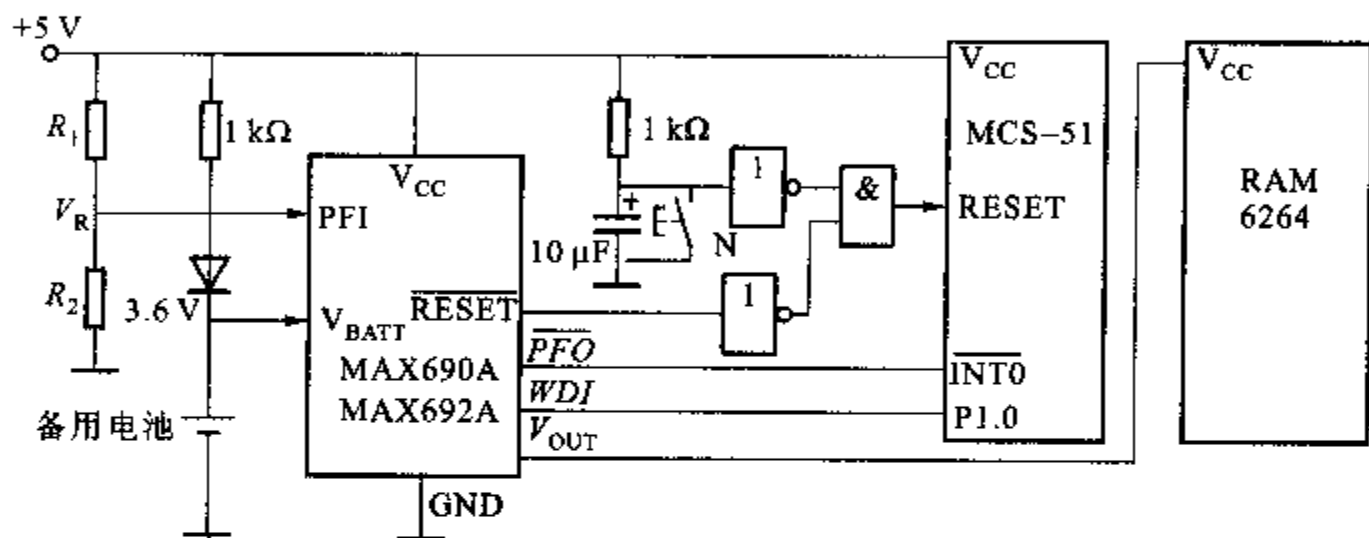


图 14-19 MCS-51 与 MAX690A 的接口电路

单片机正常工作时,P1.0 口定时(小于 1.6 s)改变 WDI 输入端的电平,使看门狗电路不发出复位信号。当由于某种严重干扰而出现死循环时,单片机将不能定期改变 WDI 端电平,看门狗电路便会在 1.6 s 后产生 1 个复位信号,使单片机复位。待经过 200 ms 复位脉冲后,单片机复位结束,程序从 0000H 开始重新执行,摆脱死循环,保证了系统的正常运转。

图中的 N 为手动复位按钮,由于 MAX690A/MAX692A 在系统上电时能自动发生复位信号,可使手动复位按钮的复位时间小于 200 ms。

思考题及习题

1. 单片机系统的干扰来源都有哪些?
2. 为什么要在每块的电源与地之间并接去耦电容? 加几个去耦电容? 电容量选多大为适宜?
3. 光电隔离的主要优点是什么? 在单片机应用系统中,应在什么位置进行光电隔离?
4. 具有较大电感量的元件或设备,诸如继电器、电动机、电磁阀等,在其断电时,应采用什么措施来抑制其反电动势?
5. 为什么要将所有的单片机应用系统中的模拟地和数字地分别相连,然后仅在一点上

相连接?

6. 如何在单片机应用系统中实现电源去耦和集成芯片去耦?
7. 为什么在印制板的设计中, 不要在印制板中留下无用的空白铜箔层, 走线不要有分支?
8. 常见的软件滤波方法都有哪些? 它们各自对哪种干扰信号有效?
9. 什么是指令冗余、软件陷阱?
10. 采用看门狗技术能使单片机应用系统摆脱死循环, 请说明摆脱死循环的工作原理。
11. 如何来实现对单片机应用系统中, 重要数据的掉电保护?

第 15 章 MCS-51 单片机应用系统的设计、开发与调试

本章我们将对单片机应用系统的设计、开发和调试等各个方面加以介绍,以便读者通过对本章的学习,能掌握单片机应用系统的设计、开发和调试的思路和方法。

15.1 MCS-51 单片机应用系统的设计步骤

单片机应用系统是指以单片机为核心,配以一定的外围电路和软件、能实现某种或几种功能的应用系统。它由硬件部分和软件部分组成。单片机应用系统的设计主要包括硬件设计和软件设计两大部分。为保证系统能可靠工作,在软、硬件的设计中,还要考虑其抗干扰能力,有关系统的抗干扰设计,已在第 14 章进行了讨论。

应该指出:在应用系统的设计中,软件、硬件和抗干扰设计是紧密相关、不可分离的。在有些情况下硬件的任务可由软件来完成(如某些软件滤波、校准功能等);而在另一些要求系统实时性强、响应速度快的场合,则往往用硬件代替软件来完成某些功能。设计者应根据实际情况,合理地安排软、硬件的比例,选取最佳的设计方案,使系统具有最佳的性能价格比。

设计一个单片机测控系统,一般可分为四个步骤:

(1) 需求分析,方案论证和总体设计阶段

需求分析,方案论证是单片机测控系统设计工作的开始,也是工作的基础。只有经过深入细致的需求分析,周密而科学的方案论证才能使系统设计工作顺利完成。

需求分析的内容主要包括:被测控参数的形式(电量、非电量、模拟量、数字量等)、被测控参数的范围、性能指标、系统功能、工作环境、显示、报警、打印要求等。

方案论证是根据用户要求,设计出符合现场条件的软硬件方案,在选择测量

结果输出方式上,既要满足用户要求,又要使系统简单、经济、可靠,这是进行方案论证与总体设计一贯坚持的原则。

(2) 器件选择,电路设计制作,数据处理,软件的编制阶段

(3) 系统调试与性能测定

编制好的程序或焊接好的线路,不能按预计的那样正确工作是常见的事,这就需要查错和调试。查错和调试有时是很费时间的。

调试时,应将硬件和软件分成几部分,逐个部分调试,各部分都调试通过后再进行联调。调试完成后,应在实验室模拟现场条件,对所设计的硬件、软件进行性能和功能测定。

(4) 文件编制

文件不仅是设计工作的结果,而且是以后使用、维修以及进一步再设计的依据。因此,一定要精心编写,描述清楚,使数据及资料齐全。

文件应包括:任务描述;设计的指导思想及设计方案论证;性能测定及现场试用报告与说明;使用指南;软件资料(流程图,子程序使用说明,地址分配,程序清单);硬件资料(电路原理图,元件布置图及接线图,接插件引脚图,线路板图,注意事项)。

15.2 应用系统的硬件设计

一个单片机应用系统的硬件设计包括两大部分内容:

- 单片机系统的扩展部分设计。它包括存储器扩展和 I/O 接口扩展。存储器的扩展指 EPROM、EEPROM 和 RAM 的扩展,I/O 接口扩展是指 8255、8155、8279 以及其它 I/O 功能器件的扩展。它们都属于单片机系统扩展的内容。在本书的第 8、9、10 章中已作了介绍。

- 各功能模块的设计。如信号测量功能模块、信号控制功能模块、人机对话功能模块、通信功能模块等,根据系统功能要求配置相应的 A/D、D/A、键盘、显示器、打印机等外围设备。

为使硬件设计尽可能合理,应重点考虑以下几点:

1. 尽可能采用功能强的芯片

(1) 单片机可考虑优先选用片内带有闪烁存储器的产品。例如 ATME1 公司的 89C51/89C52/89C55。使用此类芯片,可省去扩展单片机程序存储器的工作,从而减少芯片的数目,缩小体积。

(2) EPROM 空间和 RAM 空间。目前 EPROM 容量越来越大,一般尽量选用容量大的 EPROM。MCS-51 内部的 RAM 单元有限,当要增强软件数据

处理功能时,往往觉得不足,这就要求系统配置外部 RAM,如 6264,62256 等。

(3) I/O 端口。在样机研制出来后进行现场试用时,往往会发现一些被忽视的问题,而这些问题是不能单靠软件措施来解决的。如有些新的信号需要采集,就必须增加输入检测端,有些物理量需要控制,就必须增加输出端。如果在硬件设计之初就多设计出一些 I/O 端口,这些问题就会迎刃而解。

(4) A/D 和 D/A 通道。和 I/O 端口设计类似,留出一些 A/D 和 D/A 通道将来可能会解决大问题。

2. 以软代硬

原则上,只要软件能做到的,且能满足性能要求的,就不用硬件。硬件多了不但增加成本,而且系统故障率也增大了。以软代硬的实质是以时间代空间,软件执行过程需要消耗时间,因此,这种代替带来的不足就是实时性下降。在实时性要求不高的场合,以软代硬是很合算的。

3. 工艺设计

包括机箱、面板、配线、接插件等。必须考虑到安装、调试、维修的方便。另外,硬件抗干扰措施也必须在硬件设计时一并考虑进去。

15.3 应用系统的软件设计

在进行应用系统的总体设计时,软件设计和硬件设计应统一考虑,相结合进行。当系统的电路设计定型后,软件的任务也就明确了。

系统中的应用软件是根据系统功能要求设计的。一般地讲,软件的功能可分为两大类。一类是执行软件,它能完成各种实质性的功能,如测量、计算、显示、打印、输出控制等;另一类是监控软件,它是专门用来协调各执行模块和操作者的关系,在系统软件中充当组织调度角色。设计人员在进行程序设计时应从以下几个方面加以考虑:

(1) 根据软件功能要求,将系统软件分成若干个相对独立的部分。设计出合理的软件总体结构,使其清晰、简洁、流程合理。

(2) 各功能程序实行模块化、子程序化。既便于调试、链接,又便于移植、修改。

(3) 在编写应用软件之前,应绘制出程序流程图。这不仅是程序设计的一个重要组成部分,而且是决定成败的关键部分。从某种意义上讲,多花一份时间来设计程序流程图,就可以节约几倍源程序的编辑调试时间。

(4) 要合理分配系统资源,包括 ROM、RAM、定时器/计数器、中断源等。其中最关键的是片内 RAM 分配。对 8031 来讲,片内 RAM 指 00H~7FH 单元,这

128 B 的功能不完全相同,分配时应充分发挥其特长,做到物尽其用。例如在工作寄存器的 8 个单元中, R0 和 R1 具有指针功能,是编程的重要角色,避免作为它用; 20H~2FH 这 16 B 具有位寻址功能,用来存放各种标志位、逻辑变量、状态变量等;设置堆栈区时应事先估算出子程序和中断嵌套的级数及程序中栈操作指令使用情况,其大小应留有余量。若系统中扩展了 RAM 存储器,应把使用频率最高的数据缓冲器安排在片内 RAM 中,以提高处理速度。当 RAM 资源规划好后,应列出 1 张 RAM 资源详细分配表,以备编程查用方便。

15.4 MCS-51 单片机系统举例

上节介绍了硬件设计时要考虑的一些问题,下面我们介绍一些基本的单片机应用系统,供读者在设计时参考。

15.4.1 8031 的最小系统

8031 无片内程序存储器,因此,其最小应用系统必须在片外扩展 EPROM,必须有复位及时钟电路。图 15-1 为 8031 外扩程序存储器的最小应用系统。该系统仅能完成数字量的输入和输出控制。

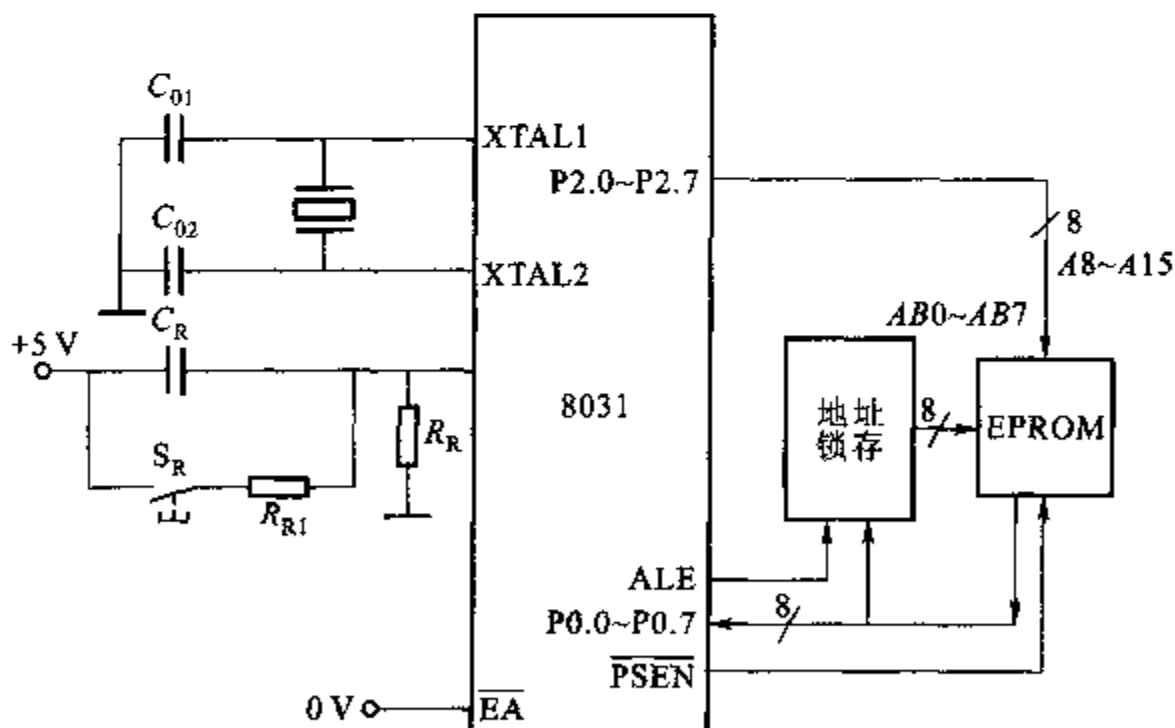


图 15-1 8031 最小应用系统

片外 EPROM 或 RAM 的地址线由 P0 口(低 8 位地址线)和 P2 口(高 8 位地址线)组成。地址锁存器的锁存信号为 ALE。

程序存储器的控制取指信号为 \overline{PSEN} 。当程序存储器只有一片时,可将其片选端直接接地。

数据存储器的读·写控制信号为 \overline{RD} 、 \overline{WR} ,其片选线与译码器输出端相连。

8031 的 \overline{EA} 引脚必须直接接地。

15.4.2 89C51 的最小系统

89C51 内部有 4 KB 闪烁存储器,芯片本身就是 1 个最小系统。在能满足系统的性能要求的情况下,可优先考虑采用此种方案。用这种芯片构成的最小系统简单、可靠。用 89C51 单片机构成最小应用系统时,只要将单片机接上时钟电路和复位电路即可,如图 15-2 所示。与 8031 外扩程序存储器的最小应用系统相比,该系统省去了外扩程序存储器的电路。该最小应用系统只能用作一些小型的数字量的测控单元。

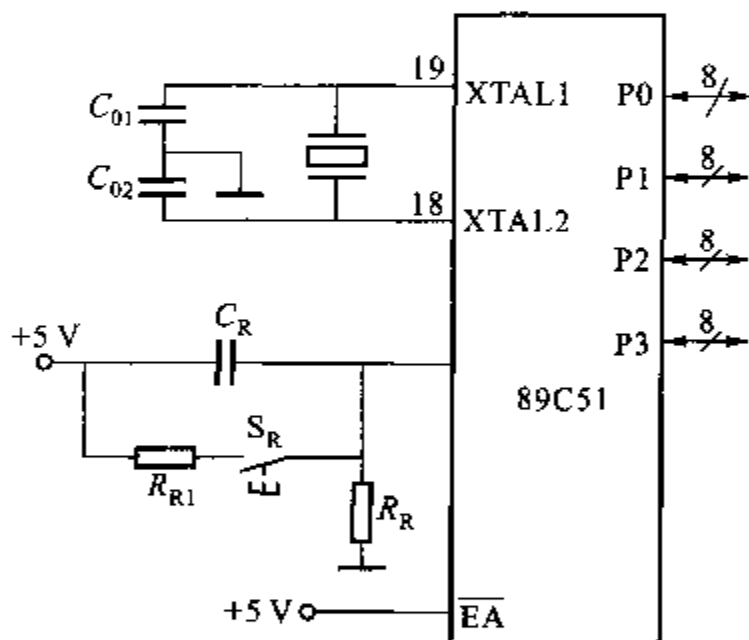


图 15-2 89C51 最小应用系统

15.4.3 以单片机为核心的数据采集系统

上面两个最小系统仅能处理数字量的输入与输出,而一个单片机应用系统往往要对工业现场的各种非电量经传感器转换得到的模拟电信号进行采集。因此,一个单片机应用系统,首先要进行数据采集,然后对数据进行处理,再加上数字量输出、D/A 转换器以及各种功率驱动部件,就构成了一个典型的单片机的测控系统。下面我们首先了解以单片机为核心的数据采集系统。

1. 数据采集系统的组成

数据采集系统一般由信号调理电路、多路切换电路、采样保持电路、A/D、CPU、RAM、EPROM 组成。其原理框图如图 15-3 所示。

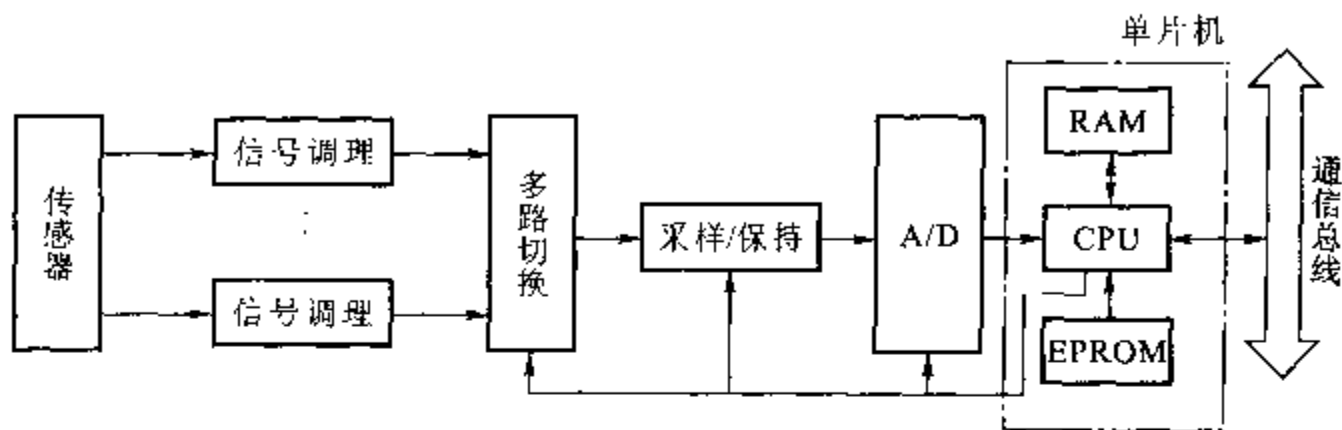


图 15-3 数据采集系统

(1) 信号调理电路

信号调理电路是传感器与 A/D 之间的桥梁,也是测控系统中重要组成部分。信号调理的主要功能是:

① 目前标准化工业仪表通常采用 $0\sim 10\text{ mA}$, $4\sim 20\text{ mA}$ 信号,为了和 A/D 的输入形式相适应,经 I/V 转换器转换成电压信号。

② 某些测量信号可能是非电量,这些非电压量信号必须变为电压信号,还有些信号即使是电压信号,也必须经过放大、滤波,这些处理包括信号形式的变换、量程调整、环境补偿、线性化等。

③ 某些恶劣条件下,共模电压干扰很强,如共模电平高达 220 V ,不采用隔离的办法无法完成数据采集任务,因此,必须根据现场环境,考虑共模干扰的抑制,甚至采用隔离措施,包括地线隔离、路间隔离等。

综上所述,非电量的转换、信号形式的变换、放大、滤波、共模抑制及隔离等,都是信号调理的主要功能。

信号调理电路包括电桥、放大、滤波、隔离等电路。根据不同的调理对象,采用不同的电路。电桥电路的典型应用之一就是热阻测温。用热电阻测温时,工业设备距离计算机较远,引线将很长,这就容易引进干扰,并在热电阻的电桥中产生长引线误差。解决办法有:采用热电阻温度变换器;智能传感器加通信方式连接;采用三线制连接方法。

信号放大电路通常由运放组成,运放的选择主要考虑精度要求(失调及失调温漂),速度要求(带宽、上升率),幅度要求(工作电压范围及增益)及共模抑制要求。常用作前置放大器的有 $\mu\text{A}741$, $\text{LF}347$ (低精度); $\text{OP}-07$, $\text{OP}-27$ (中等精度); $\text{ICL}7650$ (高精度)等。

滤波和限幅电路通常采用二极管、稳压管、电容等器件。用二极管和稳压管的限幅方法会产生一定的非线性且灵敏度下降,这可以通过后级增益调整和非

线性校正补偿。此外,由于限幅值比最大输入值高,当使用多路开关时,某一路超限可能影响其他路,需选用优质的多路模拟开关,如 AD7501。

共模电压的存在对模拟信号的处理有影响,高的共模电压会击穿器件,即使没有损坏器件,也会影响测量精度。隔离是克服共模干扰影响的有效措施。常用的隔离方法有:光电隔离、采用隔离放大器等。

(2) 多路切换电路

通常被检测的物理量有多个,如果每一通道都设有放大、采样保持(S/H)和 ADC 几个环节就很不经济,而且电路也复杂。采用模拟多路开关就可以使多个通道共用一个放大器、S/H 和 ADC,采用时间分割法使几个模拟通道轮流接通,这样既经济,又使电路简单。模拟多路开关的选择主要考虑导通电阻的要求,截止电阻的要求和速度要求,常用的模拟多路开关有 CD4051、CD4066、AD7501、AD7507 等。为了降低截止通道的负载影响,提高开关速度,降低通道串扰,采用多级模拟多路开关来完成通道切换。

(3) 采样保持电路(S/H)

采样保持电路是为了保证模拟信号高精度转换为数字信号的电路。在模拟数字变换电路中,如果变换期间输入电压是变化的,那么就可能产生错误的数字信号输出。采样保持电路就是将快速变化的模拟信号进行采样与保持,以保证在 ADC 转换过程中模拟信号保持不变。采样保持器的选择要综合考虑捕获时间、孔径时间、保持时间、下降率等参数。常用的采样保持器有:AD582、AD583、LF398 等。

(4) 模/数转换(ADC)

ADC 是计算机同外界交换信息所必需的接口器件,因为它能将描述自然现象和生产过程的模拟量转换成便于计算机存储和处理的数字量。因此,从某种意义上说,没有 ADC 的广泛应用,就没有计算机应用技术的发展。ADC 的种类很多,性能各不相同,请见本书的第 10 章。在实际工作中,选择 ADC 时需主要考虑的指标有:分辨率、转换时间、精度、电源、输入电压范围、工作环境、数字输出特性、价格等。

飞利浦公司的 8XC552 系列单片机已将上述有关数据采集的各功能部件集成在单片机内,如果想设计一个上述的数据采集系统,采用 8XC552 系列单片机来设计该系统,则是一个最佳的方案选择。

2. 数据采集系统设计中的地址空间分配与总线驱动

在数据采集系统的设计中,有时候要外接多个芯片,这时要解决 2 个问题,一是如何把 64 KB 程序存储器和 64 KB 数据存储器的空间分配给各个芯片,另一个问题是如何解决对多个芯片的驱动问题。

(1) 地址空间的分配

对于要扩展多片各种芯片的应用系统,首先应考虑如何把 64 KB 程序存储器和 64 KB 数据存储器的空间分配给各个芯片。我们已在第 8 章中介绍了如何进行地址空间分配,有 2 种方法:线选法和译码法。我们通过 1 个例子来说明如何解决这个问题。

图 15-4 是 1 个全地址译码的系统实例。图中 MCS-51 扩展的各器件芯片所对应的地址如表 15-1 所示。

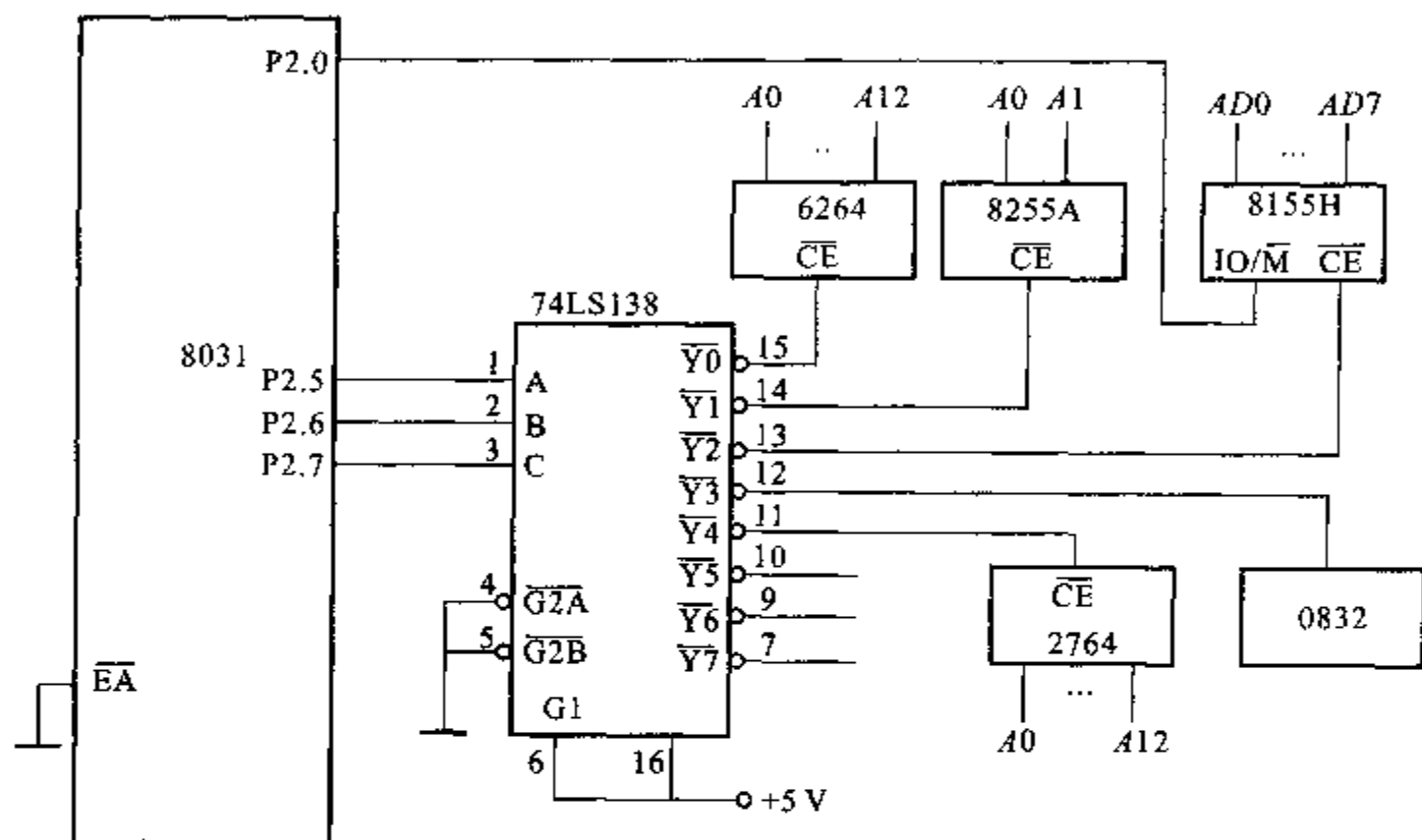


图 15-4 全地址译码实例

表 15-1 各扩展芯片的地址

器 件	地址线(A15~A0)	片内地址单元数	地 址 编 码
6264	000XXXXXXX	8 K	0000H~1FFFH
8255(1)	001111111111XX	4	3FFCH~3FFFH
8155	01011110XXXXXX	256	5E00H~5EFFH
	010111111111XX	6	5FF8H~5FFDH
0832	01111111111111	1	7FFFH
2764	100XXXXXXX	8 K	8000H~9FFFH

因 6264、2764 都是 8 KB,故需要 13 根低位地址线(A12~A0)进行片内寻址,其他 3 根高位地址线 A15~A13 经 3 线-8 线译码器译码后作为外围芯片的片选线。图中尚剩余 3 根地址选择线 $\overline{Y7} \sim \overline{Y5}$,可供扩展 3 片存储器芯片或外围

I/O 接口电路芯片。

(2) 总线的驱动

在应用系统中,所有系统扩展的外围芯片都通过总线驱动,外围芯片工作时有一定输入电流,不工作时也有漏电流存在,因此总线只能带动一定数量的电路。MCS-51 系列单片机的 P0 口可驱动 8 个 LSTTL 电路,而其他口只能驱动 4 个 LSTTL 电路。若应用系统规模过大,就可能造成负载过重,致使驱动能力不够,系统不能可靠地工作。

多芯片应用系统,首先要估计总线的负载情况,以确定是否需要对总线的驱动能力进行扩展。图 15-5 为 MCS-51 单片机总线驱动扩展原理图。

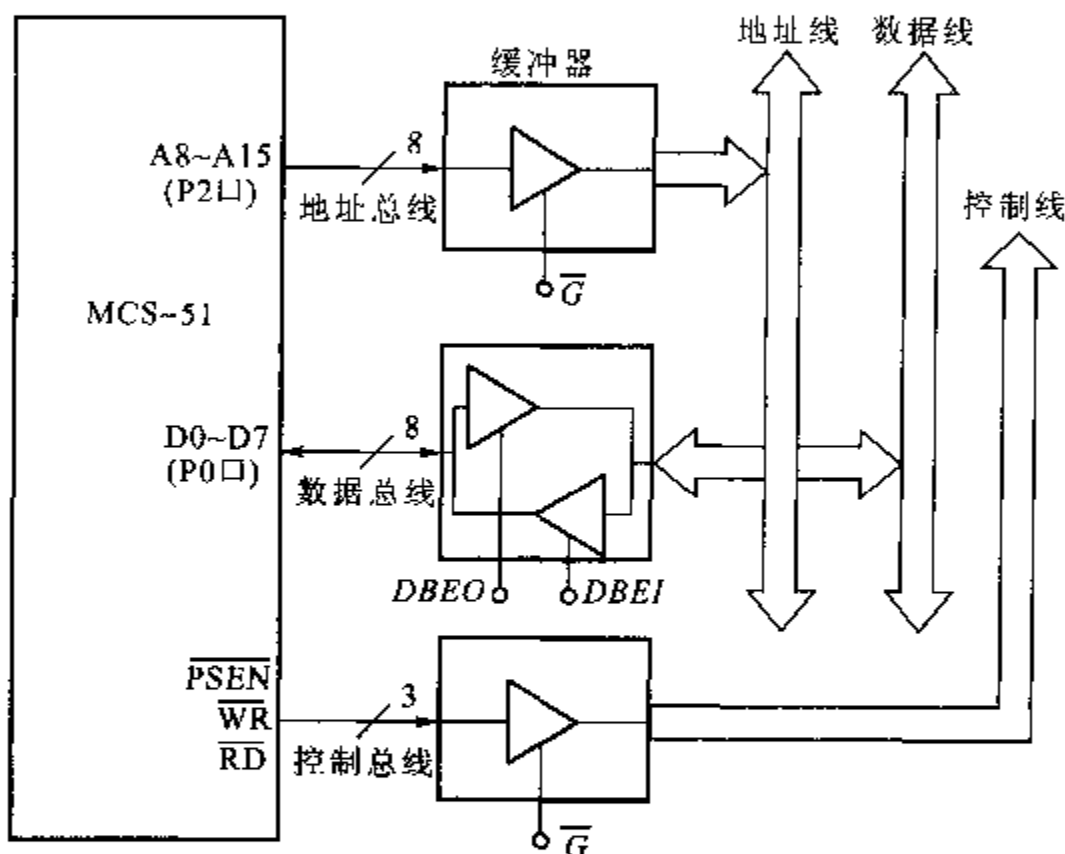


图 15-5 MCS-51 单片机总线驱动扩展原理图

地址总线和控制总线的驱动器为单向驱动器,并具有三态输出功能。驱动器有一个控制端 \overline{G} ,以控制驱动器开通或处于高阻状态。通常,在单片机应用系统中不采用 DMA 功能时,地址总线及控制总线可一直处于开通状态,这时控制端 \overline{G} 接地即可。

常用的单向总线驱动器为 74LS244。图 15-6 为 74LS244 引脚和逻辑图。8 个三态线驱动器分成 2 组,分别由引脚 1 \overline{G} 和 2 \overline{G} 控制。

数据总线的驱动器应为双向驱动、三态输出,并有两个控制端,控制数据传送方向。如图 15-5 所示,数据输出允许控制端 DBEO 有效时,数据总线输入为高阻状态,输出为开通状态;数据输入允许控制端 DBEI 有效时则状态与其相反。

常用的双向驱动器为 74LS245,图 15-7 为其引脚和逻辑图,16 个三态门每

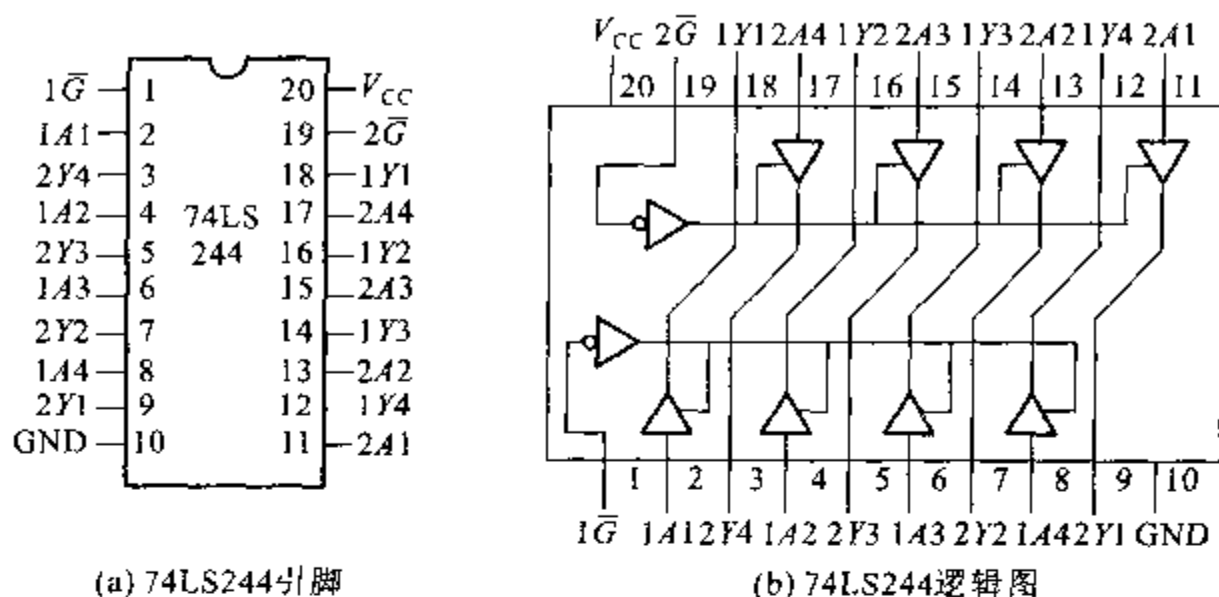


图 15-6 74LS244 引脚和逻辑图

2 个三态门组成 1 路双向驱动。驱动方向由 \overline{G} 、DIR 2 个控制端控制, \overline{G} 控制端控制驱动器有效或高阻态, 在 \overline{G} 控制端有效 ($\overline{G}=0$) 时, DIR 控制端控制驱动器的驱动方向, DIR=0 时驱动方向为从 B 至 A, DIR=1 则相反。

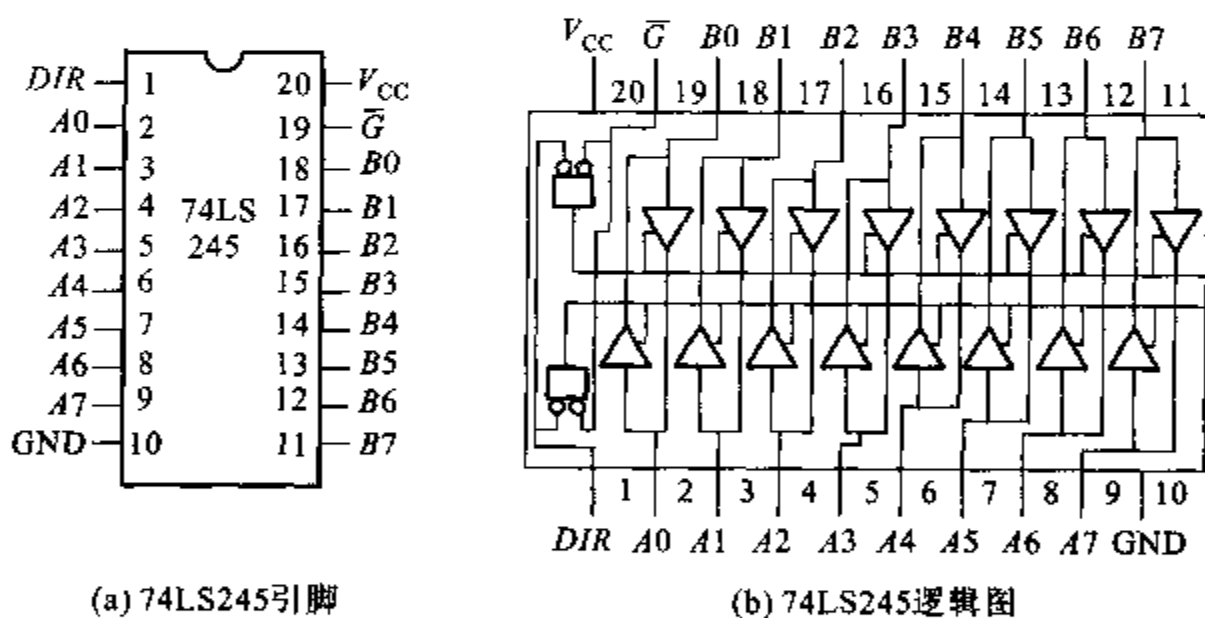


图 15-7 74LS245 的引脚和逻辑图

图 15-8 是 MCS-51 单片机应用系统总线驱动扩展电路。P0 口的双向驱动采用双向驱动器 74LS245, 如图 15-8(b) 所示; P2 口的单向驱动器采用 74LS244, 如图 15-8(a) 所示。

对于 P0 口的双向驱动器 74LS245, 使 \overline{G} 接地保证芯片一直处于工作状态, 而输入/输出的方向控制由单片机的数据存储器的读控制引脚 (\overline{RD}) 和程序存储器的取指控制引脚 (\overline{PSEN}) 通过与门控制 DIR 引脚实现。这种连接方法保证无论是读数据存储器中数据 (引脚 \overline{RD} 有效) 还是从程序存储器中取指令 (引脚 \overline{PSEN} 有效) 时, 都能保证对 P0 口的输入驱动; 除此以外的时间 (引脚 \overline{RD} 及 \overline{PSEN} 均无效), 保证对 P0 口的输出驱动。

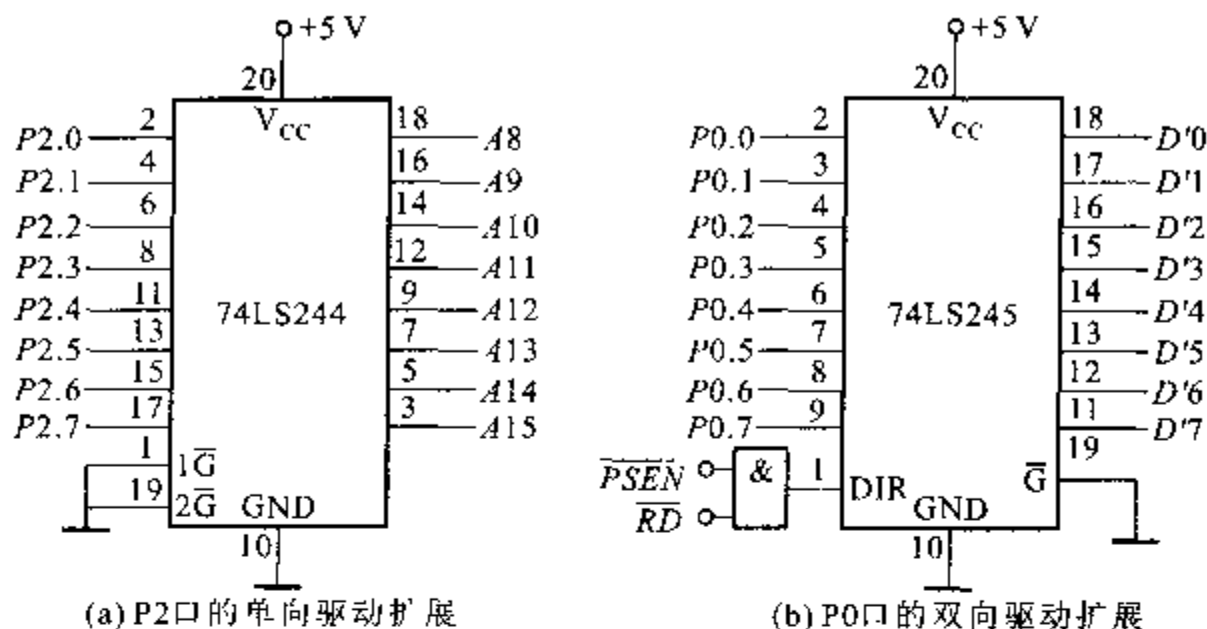


图 15-8 MCS-51 单片机应用系统中的总线驱动扩展

对于 P2 口, 因为只作地址输出口, 故 74LS244 的驱动门控制端 $1\bar{G}$ 、 $2\bar{G}$ 接地。

上面介绍了如何在总线上扩展驱动器, 下面简单介绍一下如何来估算驱动器的驱动能力。

总线驱动器驱动能力是以驱动同类门个数来度量的。驱动器驱动能力和驱动器负载性质有关。由于驱动器负载有交流和直流之分, 故总线驱动器驱动能力估算时应同时考虑交流和直流负载两方面的影响。

① 直流负载下驱动能力的估算

在直流负载下, 驱动器驱动能力主要取决于高电平输出时驱动器能提供的最大电流和低电平输出时所能吸收的最大电流。现设 I_{OH} 为驱动器在高电平输出时的最大输出电流, I_{IH} 为每个同类门负载所吸收的电流。 I_{OL} 为驱动器在低电平输出时的最大吸入电流, I_{IL} 为驱动器需要为每个同类门提供的吸入电流。显然, 如下关系满足时才能使驱动器可靠工作。

$$I_{OH} \geq \sum_{i=1}^{N_1} I_{IH}, I_{OL} \geq \sum_{i=1}^{N_2} I_{IL}$$

若: $I_{OH} = 15 \text{ mA}$, $I_{OL} = 24 \text{ mA}$, $I_{IH} = 0.1 \text{ mA}$ 和 $I_{IL} = 0.2 \text{ mA}$, 则根据上述 2 式求得 $N_1 = 150$ 和 $N_2 = 120$ 。因此, 驱动器的实际驱动能力应为 120 个同类门。

② 交流负载下驱动能力的估算

总线上传送的数据是脉冲型信号, 在同类门负载为容性(分布电容造成)时, 就必须考虑电容的影响。若: C_F 为驱动器的最大驱动电容, $C_i (i=1, 2, \dots, N)$ 为每个同类门的分布电容。为了满足同类门电容的交流效应, 驱动器负载电路应满足如下关系:

$$C_P \geq \sum_{i=1}^N C_i$$

若: $C_P = 15 \mu\text{F}$, C_i 不大于 $0.3 \mu\text{F}$, 则根据上式可求得 $N_3 = 50$ 。

综上所述, 驱动器驱动负载门的能力应从交流和直流负载两方面加以考虑。通常, 对于 TTL 负载, 主要应考虑直流负载特性, 因为 TTL 电流大, 分布电容小; 对于 MOS 型负载, 主要考虑交流特性, 因为 MOS 型负载的输入电流很小, 分布电容则是不容忽视的。

例如: 74LS245 驱动器常可驱动 100 多个 74LS $\times\times$ 门电路, 若把驱动负载的种种因素也考虑在内, 起码也能可靠驱动 50 个同类门。但为了保险起见, 74LS245 输出线上一般也只挂接 30 个左右同类门。因此, 驱动器不仅可以减轻主机负担, 增强单片机驱动负载的能力, 为负载电阻和分布电容提供较大的驱动电流, 而且也能够消除驱动器后面负载电路对主机芯片的干扰和影响, 较好的保证总线信号波形的完整性。

15.4.4 应用设计例 1——水温控制系统的设计

水温控制是经常遇到的过程控制。下面介绍以 89C51 为核心的水温控制系统的设计。本系统采用 3 位 LED 静态显示水温, 温度控制采用改进的 PID 数字控制算法。该控制系统具有如下基本功能:

- 温度控制的设定范围为 $35 \sim 85^\circ\text{C}$, 最小分辨率为 0.1°C 。
- 偏差 $\leq 0.6^\circ\text{C}$, 静态误差 $\leq 0.4^\circ\text{C}$ 。
- 实时显示当前的温度值。
- 命令按键 5 个: 复位键, 功能转换键, 加 1 键, 减 1 键。

1. 硬件电路设计

硬件电路从功能模块上来划分有:

- 主机电路
- 数据采集电路
- 键盘、显示电路
- 控制执行电路

(1) 硬件功能结构框图

硬件功能结构框图如图 15-9 所示。

(2) 数据采集电路的设计

主机采用 89C51, 系统时钟采用 12MHz , 内部含有 4KB 的闪烁存储器。无须外扩程序存储器。

本系统需要实时采集水温数据, 然后经过 A/D 转换为数字信号, 存入 89C51

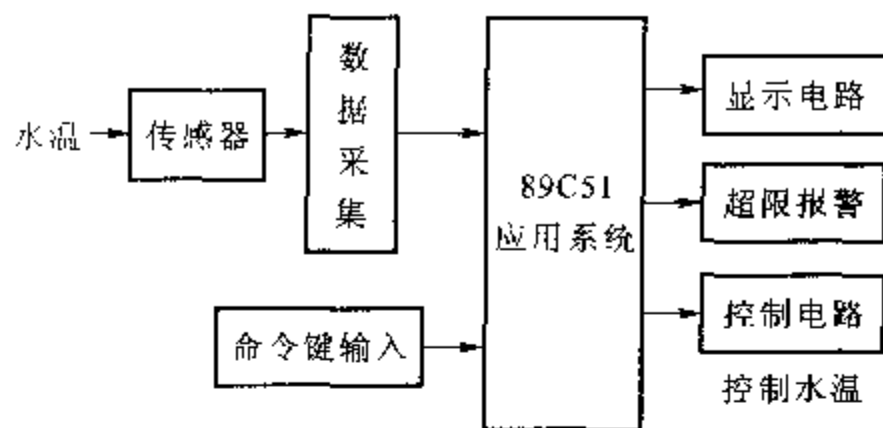


图 15-9 硬件结构框图

的内部数据存储器,送显示器显示,并与设定值进行比较,经过 PID 算法得到控制量并由单片机输出去控制电炉加热或开动风扇进行降温。

数据采集电路主要由温度传感器、A/D 转换器、放大电路等组成。具体电路如图 15-10 所示。

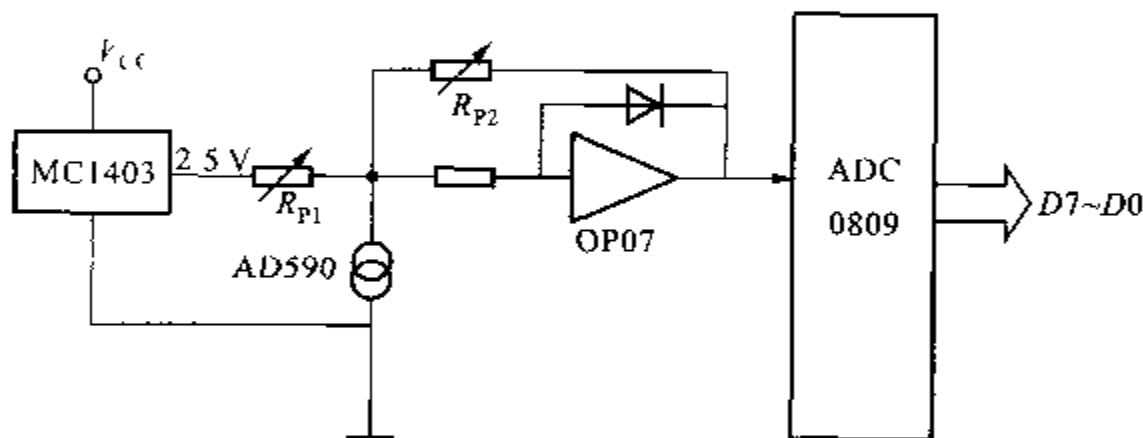


图 15-10 数据采集电路

温度传感器采用了常见的二端式电流型集成温度传感器 AD590。温度每变化 1°C ,其输出电流变化 $1\mu\text{A}$,在 25°C 时,其输出电流为 $298.2\mu\text{A}$ 。AD590 具有较高精度和重复性,测温的范围为: $-55\sim+150^{\circ}\text{C}$,重复性优于 0.1°C ,通过激光平衡调整,校准精度可达 $\pm 0.5^{\circ}\text{C}$ 。由于 AD590 的上述特点,使其在温度测控领域中得到广泛的应用。

A/D 转换器采用了 ADC0809。考虑到水温信号为缓变信号,足以满足转换速度的要求,而且还可以根据需要扩展最多测量 8 路温度信号。如果对 A/D 转换器的转换精度要求更高,可采用 12 位的 A/D 转换器,例如 AD574A 等。

放大电路采用低温漂、高精度的运算放大器 OP07 将温度传感器来的电压信号进行放大,以便于 A/D 转换器进行转换。

(3) 控制执行电路的设计

由单片机的输出来控制风扇或电炉。设计中要采用光电耦合器进行强电和弱电的隔离,但还要考虑到输出信号要对晶闸管进行触发,以便接通风扇或电炉

电路,所以晶闸管选用了既有光电隔离又有触发功能的 MC3041(请见 12.2.3 小节介绍)。其中使用引脚 P1.0 控制电炉电路,P1.1 控制风扇电路,如图 15-11 所示。

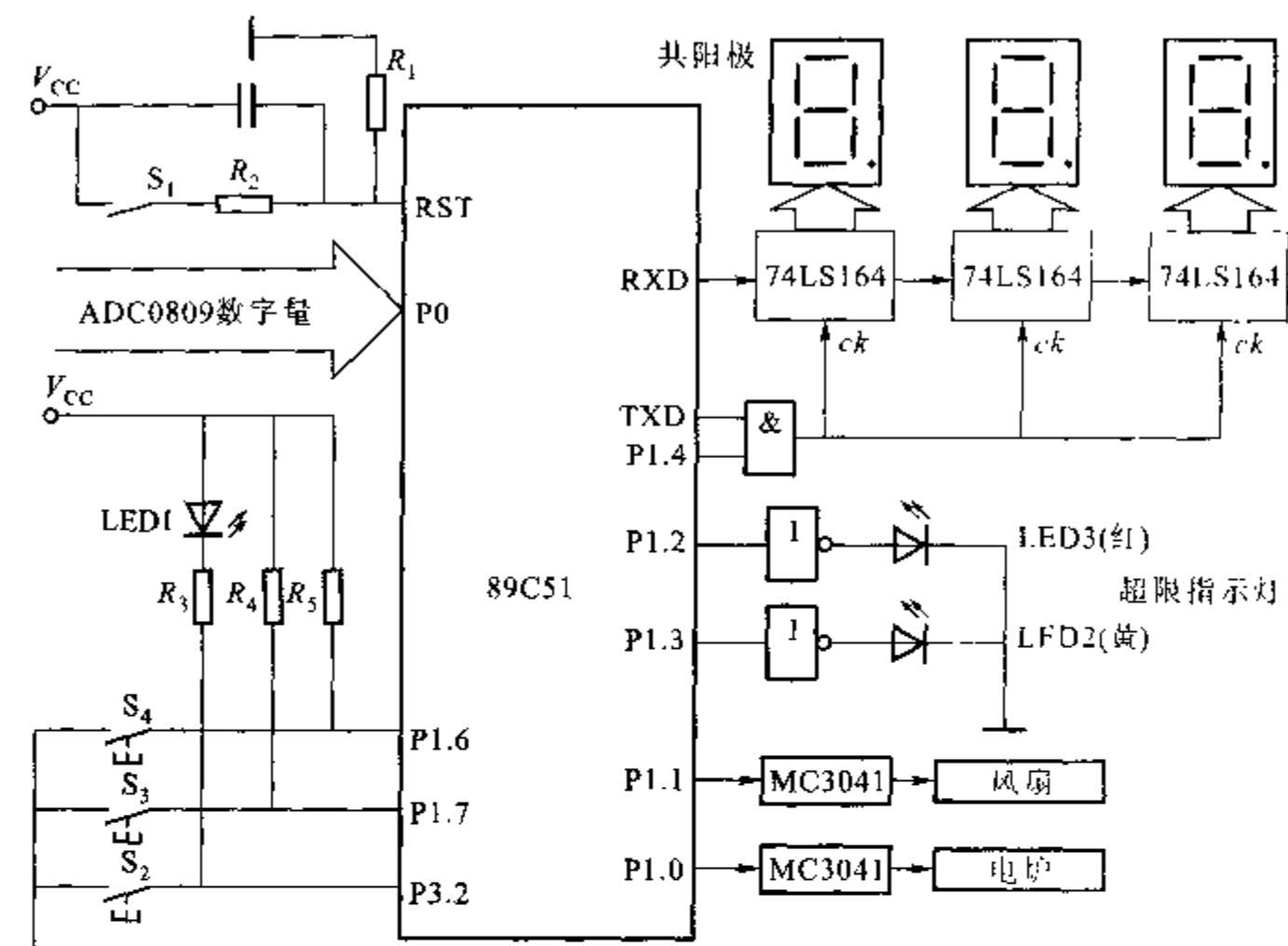


图 15-11 单片机的控制执行电路

此外,在设计中还要考虑到当水温超出所能控制的上下限温度时,要有超限报警,当温度低于 35°C 时黄色发光二极管亮,当温度高于 85°C 时红色发光二极管亮。

(4) 键盘与显示器电路的设计

键盘共有 4 个键,采用软件查询和外部中断相结合的方法来设计,当某个键按下时,低电平有效。4 个键 $S_1 \sim S_4$ 的功能定义如表 15-2 所示。

表 15-2 4 个键 $S_1 \sim S_4$ 的定义

按 键	键 名	功 能
S_1	复位键	使系统复位
S_2	功能转换键	按键按下,LED1 亮,显示温度设定值,按键松开,LED1 不亮,显示当前的温度值
S_3	加 1 键	设定的温度值加 1
S_4	减 1 键	设定的温度值减 1

按键 S_2 与 $\overline{INT0}$ (P3.2) 相连, 采用外部中断方式, 且优先级定为高优先级。 S_3 和 S_4 分别与 P1.7 和 P1.6 相连, 采用软件查询方式, S_1 为复位键, 与 RC 构成复位电路。

显示电路部分利用串行口来实现 3 位 LED 的共阳静态显示, 显示内容为温度的十位、个位以及小数点后的 1 位。利用串行口实现 LED 的共阳静态显示的工作原理及软件编程请见第 10 章中 10.3.2 小节的有关内容。

2. 软件设计

软件设计采用了模块化设计, 由主程序模块、功能实现模块和运算控制模块三大模块组成。

(1) 主程序模块

主程序流程如图 15-12 所示。在主程序中首先给定 PID 算法的参数值, 然后通过循环显示当前温度, 以等待中断, 并且使键盘外部中断为高优先级, 以便使主程序程序能实时响应键盘处理。软件设定定时器 T0 为 5 s 定时, 在无键按下时, 应每隔 5 s 响应 1 次, 以用来采集温度传感器经 A/D 转换的温度信号。设置定时器 T1 为嵌套在 T0 之中的定时中断, 初值由 PID 算法子程序提供, 以用来执行对电炉或风扇的控制。

(2) 功能实现模块

功能实现模块主要由 A/D 转换子程序、中断处理子程序、键盘处理子程序和显示子程序等组成。下面仅对几个中断处理子程序进行介绍。

① T0 中断子程序

该中断是单片机内部 5 s 定时中断, 为低优先级, 但却是最重要的子程序。在该中断响应中, 单片机要完成如下工作: A/D 转换和数据采集、数字滤波、判断是否超限、标度变换处理、显示当前温度、与设定值进行比较, 调用 PID 算法子程序并输出控制信号等。

② 键盘中断子程序

作为高优先级的功能控制键, 系统要实时准备响应该中断。在该中断的响应过程中, 系统要显示上一次的温度设定值, 并且可以通过 S_3 、 S_4 键来实现加 1、减 1 的输入修改。鉴于所控制的温度的上下限, 程序实现加 1 到 90, 再加 1 则为 40; 减 1 到 40 后, 再减 1 则为 90。

③ T1 中断子程序

T1 定时中断嵌套在 T0 中断之中, 为高优先级中断。T1 的定时初值由 PID

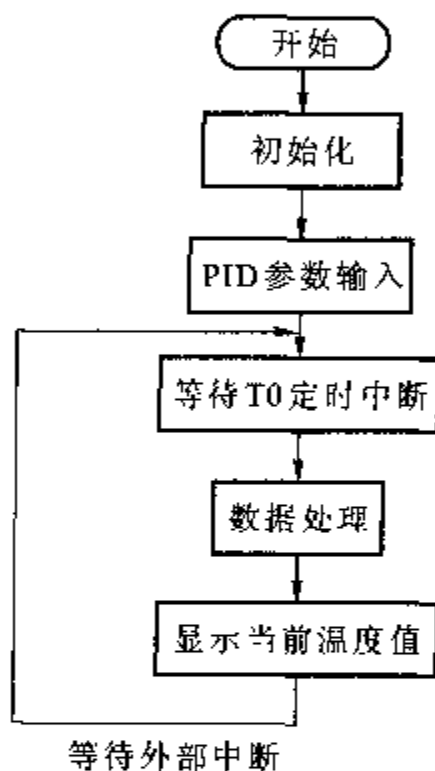


图 15-12 主程序流程

算法子程序提供, T1 的中断响应时间用于输出对电炉或风扇的控制信号。

(3) 运算控制模块

运算控制模块涉及标度变换、PID 算法以及该算法调用到的乘法子程序等。

① 标度变换子程序

该子程序的作用是将温度信号(00H~FFH)转换为对应的温度值, 以便显示或与设定值在相同量纲下进行比较。所用的线性标度变换式为:

$$Tx = [(Tm - T_0)(Nx - N_0) / (Nm - N_0)] + T_0$$

上式中: T_x 为实际测量的温度值, T_m 为 90, T_0 为 40, N_m 为 FEH, N_0 为 01H。采用定点数运算。

② PID 算法子程序

控制算法采用工业上常用的位置型 PID 数字控制, 并且结合本系统进行算法的改进, 形成变速积分 PID-积分分离 PID 控制相结合的自动识别的控制算法。控制算法如下:

$u_i(n) - u_r = e(n)$, 其中: $u_i(n)$ 为第 n 次温度采样值, u_r 为设定值。

若

$|e(n)| \geq \epsilon$, 采用 PD 算法;

$|e(n)| < \epsilon$, 采用 PID 算法。

该算法可大大减少超调, 而且可有效地克服积分饱和的影响, 使控制精度大大提高。

15.4.5 应用设计例 2——智能涡街流量计的设计

在连续生产过程中, 对管道内液体和气体的流量进行测量和控制, 是实现生产过程自动化的重要组成部分。下面介绍利用压电传感器和涡街原理开发的新型智能涡街流量计。

1. 基本工作原理及功能

涡街流量计是基于卡曼流体涡街原理制成的一种流体振荡型流量计。卡曼流体旋涡力学指出, 流过非流体阻力柱体(旋涡发生体)在雷诺数为 5 000~500 000 范围内时, 流体在旋涡发生体后出现稳定的且与流体方向相反的两侧内旋涡列, 一侧的旋涡分离频率与流速成正比, 这种关系通过斯特劳哈尔数 s_t 来表示:

$$f = s_t \times \bar{v} / (1 - 1.25d/D)d$$

式中 f 为旋涡分离频率(Hz), s_t 为斯特劳哈尔数, \bar{v} 为被测流体的平均流速, d 为与流体流动方向垂直的非流线体迎流宽度, D 为管道直径。旋涡在柱体两侧产生时, 柱体受到与流向垂直的交变升力, 交变升力使柱体内产生交变应力, 应力的方向变化频率与旋涡分离频率相同, 采用封装在柱体内部的压电元件, 通过正压电

效应将这种交变应力转换成变化频率与旋涡分离频率相同的交变电荷,通过信号调理电路将这种电荷信号进行变换处理后,输出与流量成正比的脉冲信号,再将该脉冲信号换算成相应的流量。

仪器包括两部分:(1)流量计部分,有时称流量变送器;(2)流量计的附加装置,通常称为流量积算仪。整机原理框图如图 15-13 所示。

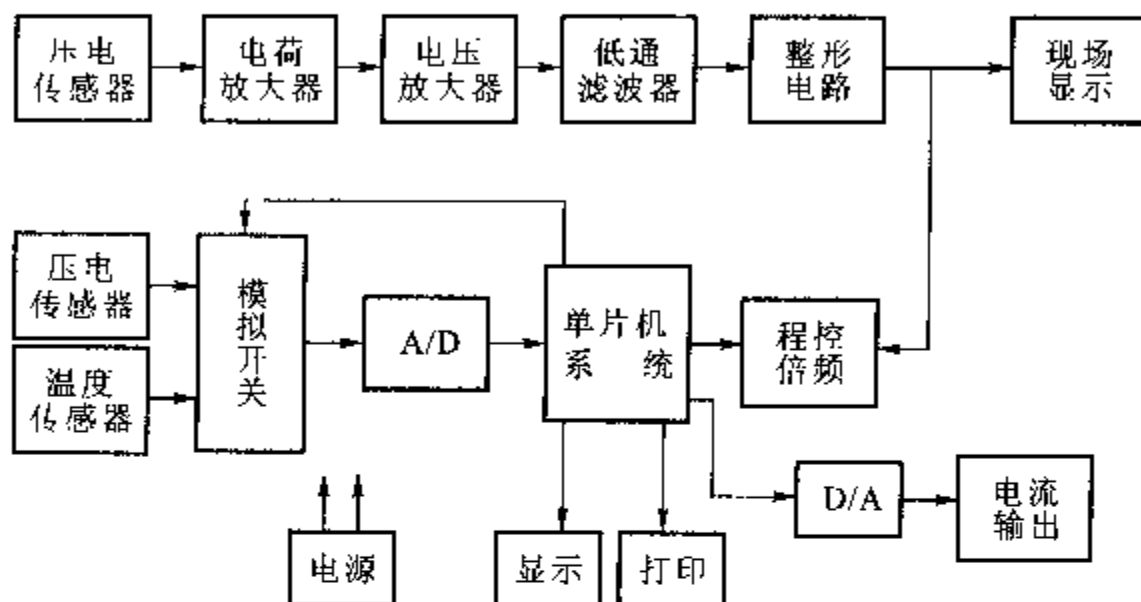


图 15-13 整机原理框图

仪器设置显示、预置、打印和选择功能键。可随时显示标况、瞬时和累计流量值以及现场温度、压力值和仪表常数 K 值,还可以实现现场瞬时显示。可根据用户需要分别显示体积流量、质量流量和重量流量。预置功能包括预置各种计算数字、仪表常数和被测介质的密度值和比重值等。可随时打印用户所需的瞬时、累计各种流量值、压力值、温度值和仪表常数等参数。

2. 硬件设计

(1) 变送器

从传感器的角度分析,压力式涡街流量计是一个压电测试系统。变送器的电子线路部分由完整的压电传感器的信号调理电路组成,其中包括电荷放大级、失调放大级、低通滤波器及施密特整形电路等。电荷放大级属于静电测试系统,为仪表的核心部分,应该采用高输入阻抗、高增益、低温漂放大器。

(2) 单片机系统结构及其硬件配置

根据设计要求,单片机应用系统包括:① 接受变送器送来的与流量成正比的脉冲,并对其定时、计数的电路;② 显示器与键盘接口电路;③ 温度、压力传感器送来的两路信号的数据处理转换电路;④ TP μ P-16A 打印机接口及报警二极管指示电路;⑤ 与流量成正比的控制电流的转换驱动电路;⑥ 外部存储器的扩展电路。单片机系统的整体框图如图 15-14 所示,现将其中主要电路介绍如下:

① 显示器/键盘接口

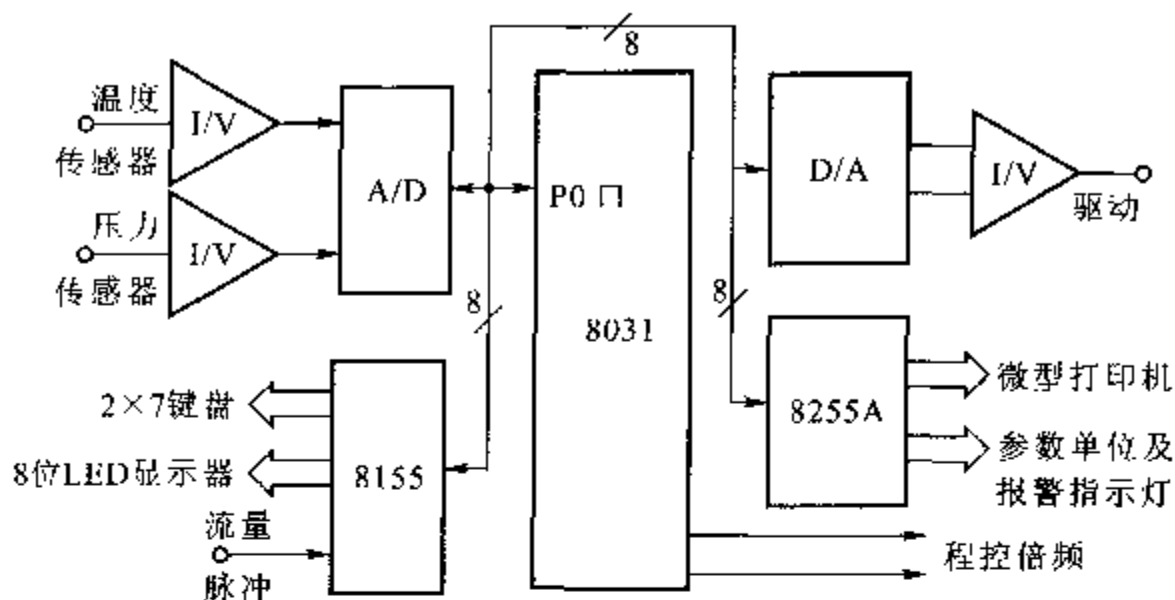


图 15-14 单片机应用系统框图

8031 外扩 1 片 8155 RAM/IO 扩展器同显示器/键盘相连接如图 15-15 所示。这是常用的 1 种显示器/键盘接口, 基本工作原理已在第 10 章介绍过。

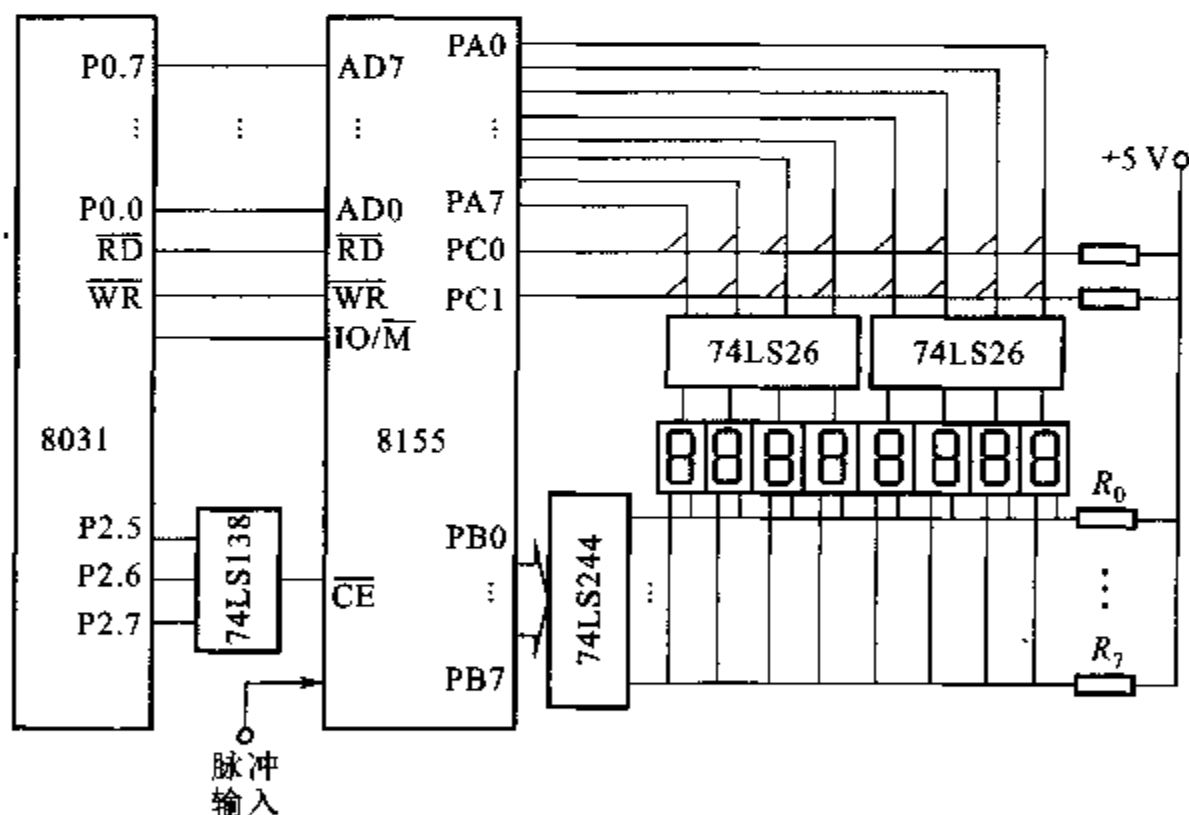


图 15-15 显示器/键盘接口电路

显示器段控信号由 PB 口提供, 但是由于 PB 口驱动功率不够, 所以在 PB 口和 LED 之间加入 1 片 74LS244 驱动器。74LS244 选通端 $\overline{1G}$ 、 $\overline{2G}$ 接地, 构成输出状态。接在 LED 与 74LS244 之间的电阻 $R_0 \sim R_7$, 用于调节 LED 亮度, 一般在 $47 \sim 100 \Omega$ 之间。8 个 LED 的位控信号为 PA 的 8 根 I/O 线, 连接到反相驱动器 74LS26 的输入端, 74LS26 的输出端连接到 LED 显示器(共阴极)的位控端。

仪器采用非编码键盘, 通过软件对键盘进行动态扫描来识别哪一键闭合。在

图 15-15 所示的键盘接口中,2 条行线接在 PC0、PC1 上,8 条列线接在 PA0~PA7 上。在工作时,键盘扫描程序依次向 PA0~PA7 发出低电平信号,并通过测试 PC0、PC1 的状态来判断键的闭合情况。

② 打印机接口

打印机选用 TP μ P-16A 微型点阵式打印机,该打印机使用 Model150 II 型打印机芯并用 1 片 8039 单片机(MCS-48 系列)对打印机进行内部管理,每行可打印的点阵字符为 16 个,其内部有 1 个包含 240 种字符的字库,并能打印图形和曲线。TP μ P-16A 采用 Centronics 标准接口,通过机后的 20 芯扁平电缆及插件与计算机连接。

单片机与打印机的接口,见图 15-16,是通过单片机外扩一片 8255 实现的。8255 的 PA 口接在打印机的数据线 DB0~DB7 上,PC0 与 TP μ P-16A 的 BUSY 端相连,PC7 接到微型打印机的 STB 上,此外 8255 的 PB 口和 PC4 用作显示被测测量指示灯和报警指示灯(指示灯采用发光二极管)的接口。

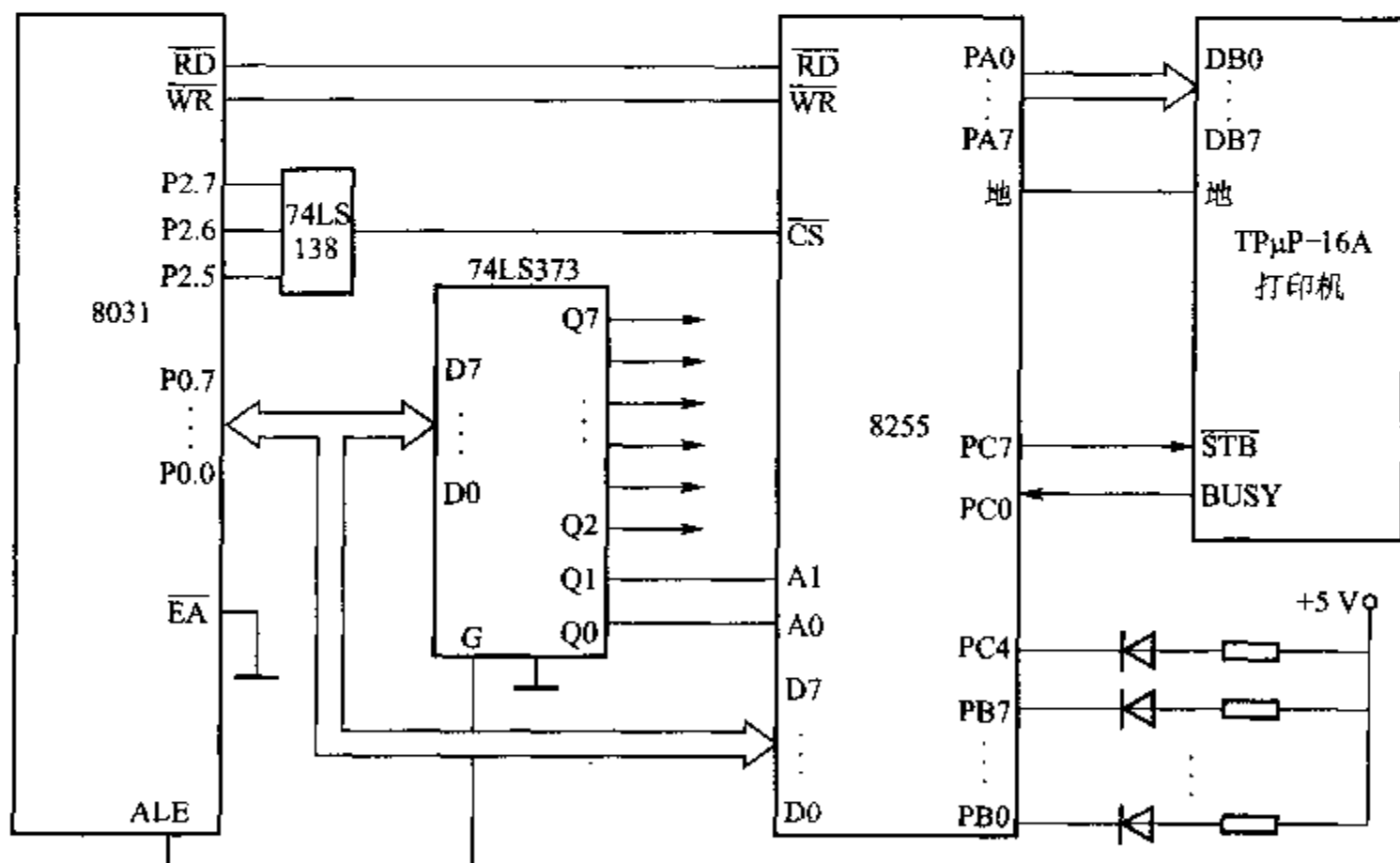


图 15-16 打印机接口电路

③ A/D 与 D/A 转换器与单片机的接口

本机采用监测被测介质温度和压力传感器,分别提供 0~10 mA 或 4~20 mA 的电流输出。

采用的 A/D 转换器为 ADC0809,因其模拟信号输入范围为 0~5 V,故应在其模拟电压信号输入端,设置 I/V 转换器,接口电路见图 15-17。

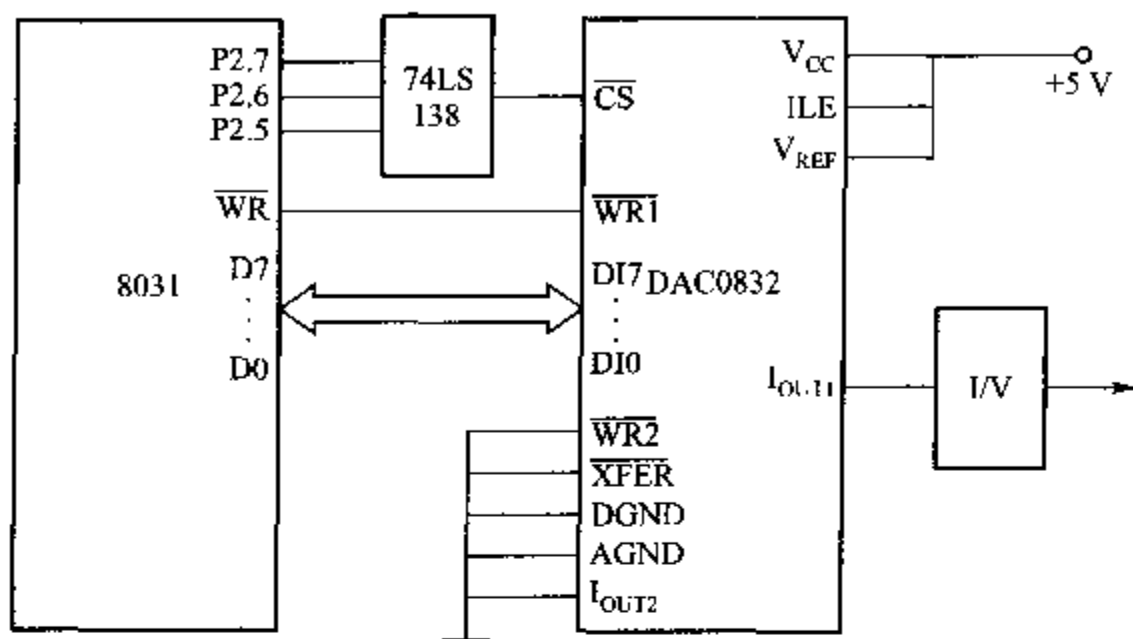


图 15-18 D/A 接口电路

出低 8 位地址时,地址由信号 ALE 的下降沿控制锁存到锁存器中,高 5 位由 $P2.0 \sim P2.4$ 引脚提供,锁存器采用 74LS373,其锁存控制端直接与 ALE 端相连。由程序存储器读选通信号 $PSEN$ 控制 EPROM 2764 的输出允许端 \overline{OE} 。

② 数据存储器的扩展

MCS-51 单片机内部 RAM 为 128 B,因其容量不能满足设计要求,故本机扩展 8 KB 静态 RAM 6264 1 片。本机外扩展的 RAM 和 EPROM 电路图如图 15-19 所示。

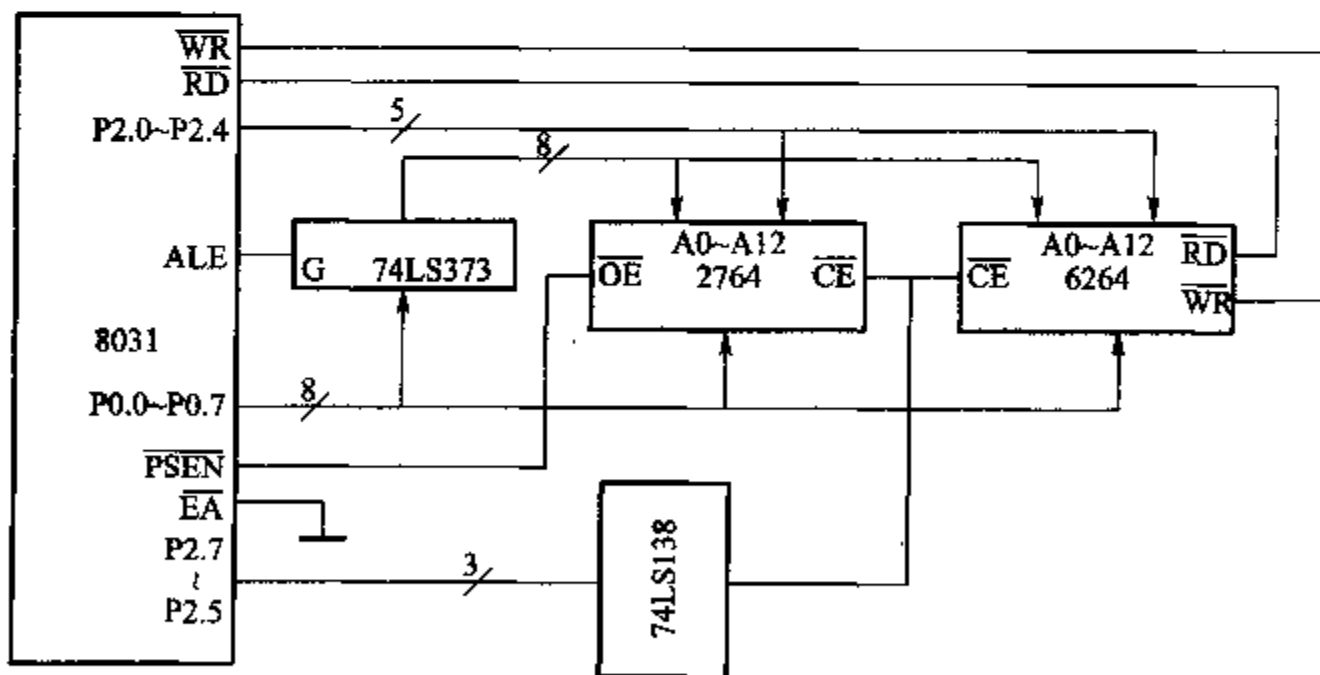


图 15-19 外部存储器扩展

从图中可以看出 EPROM 2764 与 RAM 6264 的地址范围是相同的,但是它们的控制信号是不一样的。2764 的读选通信号是 $PSEN$,而 6264 的读出或写入是

靠 \overline{RD} 或 \overline{WR} 信号控制,所以不会产生数据冲突的问题。

3. 程序设计

本机软件采用模块结构,分为如下四个主要部分:

(1) 主程序

主程序为本仪器的监控程序,是控制者。用户通过监控程序监控仪器工作。在程序运行中,必须首先对系统进行初始化,完成清零各工作单元,置计数器及标志位初值,自检指示灯,开中断,启动计数器等工作。

本仪器采用键盘和LED指示灯相配合,使仪器的各种功能清晰有序,共设11个按键,可分为功能键和数字键。功能键包括显示、打印、预置和选择等功能。

键盘子程序包括:扫描键盘子程序,其功能是寻找是否有键按下;输入键值程序;键值扫描子程序;表驱动程序;通用显示子程序等。键值扫描子程序的功能是根据按键的位置一行行一列列的扫描。表驱动程序的功能是判别按键是哪种功能键。通用显示子程序的功能是将显示缓冲区中的字码转换成段码送入显示器中,显示各种字形。几乎所有程序中都要用到这一程序。因此称为通用显示子程序,以便与显示功能块相区别。

(2) 中断服务程序

仪器的测量、转换和温度补偿等均采用中断方式同主程序相连,单片机内部的两个定时器/计数器均作为闸门使用。因为对流量传感器输出频率的测量是最重要的,所以定时器T0被用来测频,并定为高级中断。流量测频中断服务子程序流程图如图15-20所示。中断服务子程序T1的功能主要是定时对被测物体的温度和压力进行采样,该参数用于将工况流量转换为标况流量,该程序流程图如图15-21所示。

(3) 功能块程序

仪器通过键盘输入命令,可随时得到用户所需的结果,这就要用到功能程序块。功能程序块包括:显示、打印、锁定、清零等功能块。显示功能块的作用是根据用户的需要转入相应的入口,查找到相应的入口参数,再经过码制转换,送至显示缓冲区中。打印程序包括打印瞬时和累计流量。当打印瞬时流量时,还打印出当时的温度和压力值以及本机的仪表常数;当打印累计流量时,仅打印本机的仪表常数。锁定功能块的作用是在显示瞬时流量时,将此时此刻的流量值记录下来,并使其保持不变,这一功能由锁定键处理子程序完成。

实用计算机程序主要包括计算流量的程序。采用的是浮点制运算符程序,这些运算符程序可直接调用。

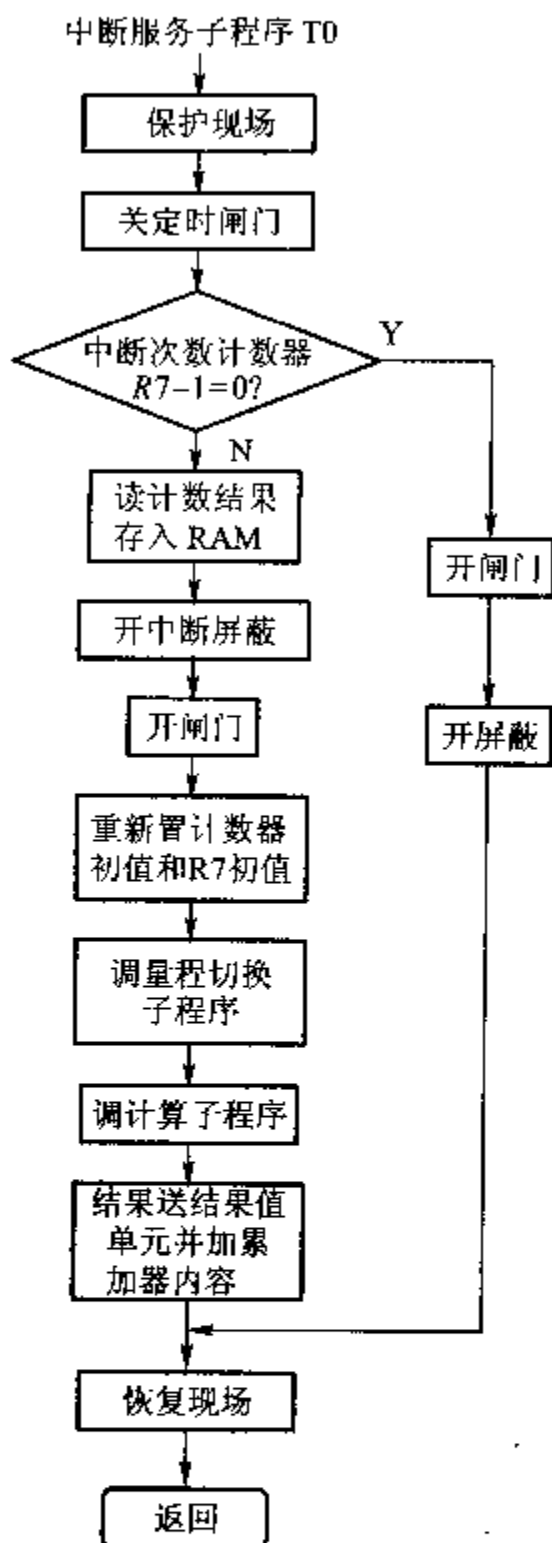


图 15-20 T0 的中断服务子程序框图

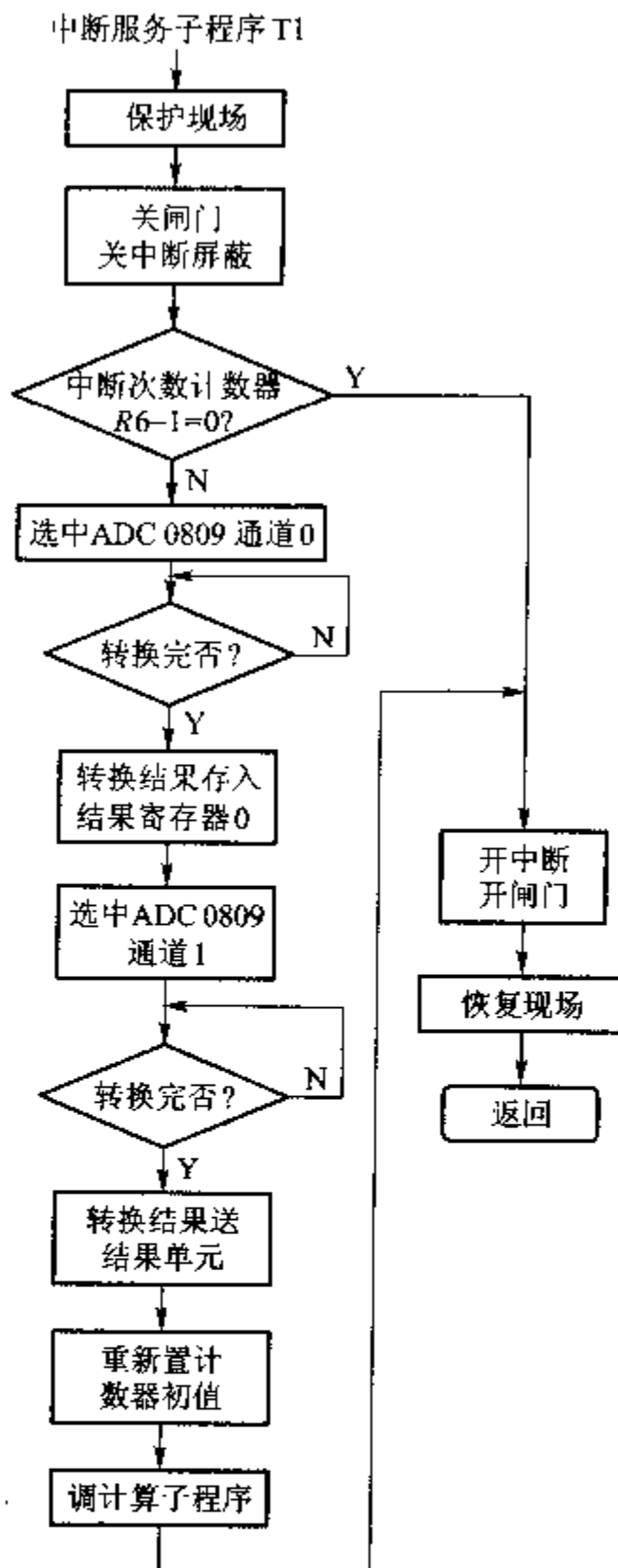


图 15-21 T1 的中断服务子程序框图

15.5 单片机应用系统的开发和调试

一个单片机系统经过总体设计,完成了硬件和软件设计开发。元器件安装后,在系统的程序存储器中放入编制好的应用程序,系统即可运行。但程序运行一次性成功几乎是不可能的,多少会出现一些硬件、软件上的错误,这就需要通过

调试来发现错误并加以改正。MCS-51 单片机虽然功能很强,但只是一个芯片,既没有键盘,又没有 CRT,LED 显示器,也没有任何系统开发软件(如编辑、汇编、调试程序等),也就是说 MCS-51 单片机本身无自开发能力。而编写、开发应用软件,对硬件电路进行诊断、调试,必须借助某种开发工具来模拟用户实际的单片机,并且能随时观察运行的中间过程而不改变运行中原有的数据、性能和结果,从而进行模仿现场的真实调试。完成这一在线仿真工作的开发工具就是单片机在线仿真器。一般也把仿真、开发工具称为仿真开发系统。

15.5.1 仿真开发系统简介

1. 仿真开发系统的功能

一般来说,仿真开发系统应具有如下最基本的功能:

- (1) 用户样机硬件电路的诊断与检查;
- (2) 用户样机程序的输入与修改;
- (3) 程序的运行、调试(单步运行、设置断点运行)、排错、状态查询等功能;
- (4) 将程序固化到 EPROM 芯片中。

不同的仿真开发系统都必须具备上述基本功能,但一个较完善的仿真开发系统还应具备:

(1) 有较全的开发软件。最好配有高级语言(C、PL/M 等),用户可用高级语言编制应用软件,由开发系统编译连接生成目标文件、可执行文件。同时要求用户可用汇编语言编制应用软件,开发系统自动生成目标文件,并配有反汇编软件,能将目标程序转换成汇编语言程序。有丰富的子程序可供用户选择调用。

(2) 有跟踪调试、运行的能力。仿真开发系统占用单片机的硬件资源尽可能得少。

(3) 为了方便模块化软件调试,还应配置软件转储、程序文本打印功能及设备。

2. 仿真开发系统的种类

目前国内使用较多的仿真开发系统大致分为如下两大类。

(1) 通用机仿真开发系统

此类仿真开发系统是目前国内使用最多的一类开发装置。这是一种通过通用计算机(PC 机)的并行口或串行口,外加在线仿真器的仿真开发系统。如图 15-22 所示。

在这种系统中,仿真开发系统不能独立完成开发任务,必须与 PC 机的并行口或串行口相连。

在调试用户样机时,仿真插头必须插入用户样机空出的单片机插座中。当仿



图 15-22 通用机开发系统

真开发系统通过串行口(或并行口)与 PC 机联机后,用户可利用组合软件,在计算机上编辑、修改源程序,然后通过 MCS-51 交叉汇编软件将其汇编成目标码,传送到仿真器的仿真 RAM 中。这时用户可用单步、断点、跟踪、全速等方式运行用户程序,系统状态实时地显示在屏幕上。待程序调试通过后,再使用专用的编程器,通过 PC 机,把程序写入到 EPROM 中。此类仿真开发系统的典型代表是南京伟福(Wave)公司的产品。配置不同的仿真头,可以仿真各种 1~16 位单片机。

通用机仿真开发系统中还有另一种结构,如上海复旦大学的 SICE-Ⅱ、SICE-Ⅳ 的在线仿真器。它们采用国际上流行的独立型仿真结构,与任何具有 RS-232C 串行接口(或并行口)PC 机相连,即可构成单片机仿真开发系统。系统中配备有 EPROM 读出/写入器、仿真插头和其他外设。

在与 PC 机联机调试用户样机时,与前面介绍的仿真开发系统的使用方法基本一样。与上一种仿真开发系统不同的是,该类仿真器采用模块化结构,配备有不同外设,如外存板、打印机、键盘/显示板等,用户可根据需要加以选用。在没有通用计算机支持的场合,利用键盘/显示板也可在现场完成仿真调试工作。

(2) 软件模拟开发系统

软件模拟开发系统,也称软件模拟器,这是一种完全用软件手段进行开发的系统。软件模拟开发系统与用户系统在硬件上无任何联系。通常这种系统是由通用 PC 机加模拟开发软件构成。用户在通用计算机上安装软件模拟器即可进行软件调试。使用者从南京伟福(Wave)的网站(www.wave.com)上即可下载该软件。

软件模拟器的工作原理是利用模拟开发软件在通用计算机上实现对单片机的硬件模拟、指令模拟、运行状态模拟,从而完成应用软件开发的全过程。单片机相应输入端由通用键盘相应的按键设定。输出端的状态则出现在 CRT 指定的窗口区域。在软件模拟器的支持下,通过指令模拟,可方便地进行编程、单步运行、设断点运行、修改等软件调试工作。调试过程中,运行状态、各寄存器状态、端口状态等都可以在 CRT 指定的窗口区域显示出来,以确定程序运行有无错误。

用软件模拟器调试软件不需任何在线仿真器,也不需要用户样机,直接就可

以在 PC 机上直接开发和调试 MCS-51 单片机软件。调试完毕的软件可以将机器码固化,完成一次初步的软件设计工作。对于实时性要求不高的应用系统,一般能直接投入运行。

软件模拟器的优点是开发效率较高,不需要附加的硬件开发装置成本。软件模拟器的最大缺点是不能进行硬件部分的诊断与实时在线仿真。

15.5.2 用户样机开发调试过程

完成一个用户样机,首先要完成硬件组装工作,然后进入软件设计、调试和硬件调试阶段。硬件组装就是在设计、制作完毕的印制板上焊好元件与插座,然后就可利用仿真开发工具进行软件设计、调试和硬件调试工作。

1. 用户样机软件的设计、调试

用户样机软件设计、调试的过程如图 15-23 所示,可分为以下几个步骤:

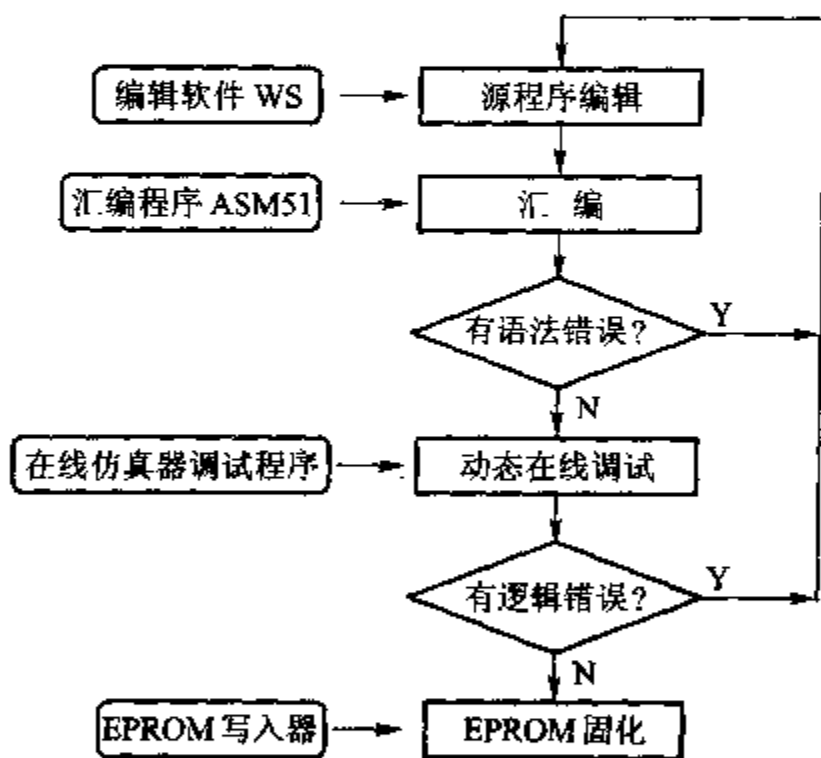


图 15-23 用户样机软件设计、调试的过程

第一步,建立用户源程序。用户通过开发系统的键盘、CRT 显示器及开发系统的编辑软件 WS,按照汇编语言源程序所要求的格式、语法规则,把源程序输入到开发系统中,并存在磁盘上。

第二步,在开发系统机上,利用汇编程序对第一步输入的用户源程序进行汇编,直至语法错误全部纠正为止。如无语法错误,则进入下一个步骤。

第三步,动态在线调试。这一步对用户的源程序进行调试。上述的第一步、第二步是一个纯粹的软件运行过程,而在这一步,必须要有在线仿真器配合,才能对用户源程序进行调试。用户程序分为与用户样机硬件无联系的程序以及与其样

机紧密关联的程序。

对于与用户样机硬件无联系的程序,例如计算程序,虽然已经没有语法错误,但可能有逻辑错误,使计算结果不对,这样必须借助于动态在线调试手段,如单步运行、设置断点等,发现逻辑错误,然后返回到第一步修改,直至逻辑错误被全部纠正为止。

对于与用户样机硬件紧密相关的程序段(如接口驱动程序),一定要先把在线仿真器的仿真插头插入用户样机的单片机插座中(如图 15-22 所示),进行在线仿真调试,仿真开发系统提供了单步、设置断点等调试手段,来对用户样机进行调试。有关部分程序段运行有可能不正常,可能软件逻辑上有问题,也可能硬件有故障,必须先通过在线仿真调试程序提供的调试手段,把硬件故障排除以后,再与硬件配合,对用户程序进行动态在线调试。对于软件的逻辑错误,则返回到第一步进行修改,直至逻辑错误被完全消除为止。在调试这类软件时,硬件调试与软件调试是不能完全分开的。许多硬件错误是通过软件的调试而发现和纠正的。

第四步,将调试完毕的用户程序通过 EPROM 编程器(也称 EPROM 写入器),固化在 EPROM 中。

15.5.3 用户样机硬件调试

对用户样机进行调试,首先要进行静态调试,静态调试的目的是排除明显的硬件故障。

1. 静态调试

静态调试工作分为两步:

第一步是在用户样机加电之前,先用万用表等工具,根据硬件逻辑设计图,仔细检查样机线路是否连接正确,并核对元器件的型号、规格和安装是否符合要求,应特别注意电源系统的检查,以防止电源的短路和极性错误,并重点检查系统总线(地址总线、数据总线、控制总线)是否存在相互之间短路或与其他信号线的短路。

第二步是加电后检查各芯片插座上有关引脚的电位,仔细测量各点电平是否正常,尤其应注意 8031 插座的各点电位,若有高压,与在线仿真器联机调试时,将会损坏在线仿真器。

具体步骤如下:

(1) 电源检查

当用户样机板连接或焊接完成之后,先不插主要元器件,通上电源。通常用 +5 V 直流电源(这是 TTL 电源),用万用表电压档测试各元器件插座上相应电源

引脚电压数值是否正确,极性是否符合。如有错误,要及时检查、排除,以使每个电源引脚的数值都符合要求。

(2) 各元器件电源检查

断开电源,按正确的元器件方向插上元器件。最好是分别插入,分别通电,并逐一检查每个元器件上的电源是否正确,直至最后插上全部元器件,通上电源后,每个元器件上电源应正确无误。

(3) 检查相应芯片的逻辑关系

检查相应芯片逻辑关系通常采用静态电平检查法。即在一个芯片信号输入端加入一个相应电平,检查输出电平是否正确。单片机系统大都是数字逻辑电路,使用电平检查法可首先检查出逻辑设计是否正确,选用的元器件是否符合要求,逻辑关系是否匹配,元器件连接关系是否符合要求等。

2. 联机仿真、在线动态调试

在静态调试中,对用户样机硬件进行了初步调试,只是排除了一些明显的静态故障。

用户样机中的硬件故障(如各个部件内部存在的故障和部件之间连接的逻辑错误)主要是靠联机在线仿真来排除的。

在断电情况下,除 8031 外,插上所有的元器件,并把在线仿真器的仿真插头插入样机上 8031 的插座,然后与开发系统的仿真器相连,分别打开样机和仿真器电源后便可开始联机在线仿真调试。

前面已经谈到,硬件调试和软件调试是不能完全分开的,许多硬件错误是在软件调试中发现和被纠正的。所以说,在上面介绍的软件设计过程中的第三步:动态在线调试中,也包括联机仿真、硬件在线动态调试以及硬件故障的排除。

开发系统的在线仿真器是一个与被开发的用户样机具有相同单片机芯片的系统,它是借助开发系统的资源来模拟用户样机中的单片机,对用户样机系统的资源如存储器、I/O 接口进行管理。同时仿真开发机还具有跟踪功能,它可将程序执行过程中的有关数据和状态在屏幕上显示出来,这给查找错误和调试程序带来了方便。同时,其程序运行的断点功能、单步功能可直接发现硬件和软件的问题。仿真开发系统和用户样机的连接如图 15-22 所示。

下面介绍在仿真开发机上如何利用简单调试程序检查用户样机电路。

利用仿真开发系统对用户样机的硬件检查,常常按其功能及 I/O 通道分别编写相应的简短的试验程序,来检查各部分功能及逻辑是否正确,下面作以简单介绍。

(1) 检查各地址译码输出

通常,地址译码输出是一个低电平有效信号。因此在选到某一个芯片时(无论是内存还是外设)其片选信号用示波器检查应该是一个负脉冲信号。由于使

用的时钟频率不同,其负脉冲的宽度和频率也有所不同。注意在使用示波器测量用户板的某些信号时,要将示波器电源插头上的地线断开,这是由于示波器测量探头一端连到外壳,在有些电源系统中,保护地和电源地连在一起,有时会将电源插座插反,将交流 220 V 直接引到测量端而将用户样机板全部烧毁,并且会殃及仿真开发机。

下面来讨论如何检查地址译码器输出,例如,一片 6116 存储芯片地址为 2000~27FFH,则可在开发机上执行如下程序:

```
LOOP:  MOV    DPTR, #2000H
        MOVX   A, @DPTR
        SJMP   LOOP
```

程序执行后,就应该从 6116 存储器芯片的片选端看到等间隔的一串负脉冲,就说明该芯片片选信号连接是正确的,即使不插入该存储器芯片,只测量插座相应片选引脚也应有上述结果。

用同样的方法,可将各内存及外设接口芯片的片选信号都逐一进行检查。如出现不正确现象,就要检查片选线连线是否正确,有无接触不好或错线、断线现象。

(2) 检查 RAM 存储器

检查 RAM 存储器可编译程序,将 RAM 存储器进行写入,再读出,将写入和读出的数据进行比较,发现错误,立即停止。将存储器芯片插上,执行如下程序:

```
        MOV    A, #00H
        MOV    DPTR, #RAM          ;首地址
LOOP:   MOVX   @DPTR, A
        MOV    R0, A
        MOVX   A, @DPTR
        CLR    C
        SUBB   A, R0
        JNZ    LOOP1
        INC    DPTR
        MOV    A, R0
        INC    A
        SJMP   LOOP
```

LOOP1: 出错停止

如一片 RAM 芯片的每个单元都出现问题,则有可能某些控制信号连接不正确,如一片 RAM 芯片中一个或几个单元出现问题,则有可能这一芯片本身是不好的,可换一片再测试一下。

(3) 检查 I/O 扩展接口

对可编程接口芯片如 8155、8255, 要首先对该接口芯片进行初始化, 再对其 I/O 端口进行 I/O 操作。初始化要按系统设计要求进行, 这个初始化程序调试好后就可作为正式编程的相应内容。程序初始化后, 就可对其端口进行读写。对开关量 I/O 来讲, 在用户样机板可利用钮子开关和发光二极管进行模拟, 也可直接接上驱动板进行检查。一般情况下, 用户样机板先调试, 驱动板单独进行调试, 这样故障排除更方便些。

如用自动程序检查端口状态不易观察时, 就可用开发系统的单步功能, 来单步执行程序, 检查内部寄存器的有关内容或外部相应信号的状态, 以确定开关量输入输出通道连接是否正确。

若外设端口连接一片 8255, 端口地址为 B000~B003H, A 口为方式 0 输入, B 口、C 口都为方式 0 输出, 则可用下述程序进行检查:

```

MOV    DPTR, #0B003H
MOV    A, #90H           ;90H 为方式控制字
MOVX   @DPTR, A
NOP
MOV    DPTR, #0B000H
MOVX   A, @DPTR          ;将 A 口输入状态读入累加器 A, 单
                           ;步执行完此步后暂停,
                           ;检查 PA 口外部开关状态同 A 中相
                           ;应位状态是否一致

CLR    C
MOV    A, #01H
INC    DPTR
LP:    MOVX   @DPTR, A    ;将 01H 送 B 口, 此指令执行完后, 暂
                           ;停。看 B 口连接的发
                           ;光二极管状态, 第 0 位是否是高
                           ;电平
RLC    A                  ;将 1 从 0 位移到第 1 位
JNZ    LP
INC    DPTR
RLC    A
LP1:   MOVX   @DPTR, A    ;将 01H 送 C 口, 此指令执行完后, 看
                           ;C 口第 0 位输出状态
RLC    A
JNZ    LP1

```

对锁存器和缓冲器, 可直接对端口进行读写, 不存在初始化的问题。

通过上面介绍的开发系统调试用户样机过程,可以体会到,离开了仿真开发系统就根本不可能进行用户样机的调试,而调试的关键步骤——动态在线仿真调试,又完全依赖于开发系统中的在线仿真器。所以说开发系统的性能优劣,主要取决于在线仿真器的性能优劣,在线仿真器所能提供的仿真开发的手段,直接影响设计者的设计、调试工作的效率。所以,它对于一个设计者来说,在了解了目前的开发系统的种类和性能之后,选择一个性能价格比高的开发系统,并能够熟练的使用它调试用户样机是十分重要的。

思考题及习题

1. 设计一个单片机测控系统,一般需要哪几个步骤?各步骤的主要任务是什么?
2. 画出 89C51 最小系统的电路图。
3. 一个单片机为核心的数据采集系统,都由哪些部分组成?各个组成部分的功能是什么?对它们都有哪些技术要求?
4. 单片机的功能是什么?
5. 仿真开发系统分为哪几类?各适合在什么场合下使用?
6. 为什么单片机应用系统的开发与调试离不开仿真开发系统?
7. 用软件模拟器能否对单片机应用系统中硬件部分进行调试与实时在线仿真?
8. 利用仿真开发系统对用户样机软件进行调试,需经哪几个步骤?各个步骤的作用是什么?

参 考 书 目

- 1 Intel. Microcontroller Handbook, 1988
- 2 Intel. Software Handbook, 1984
- 3 Analog Device Corp. Data-Acquisition Databook, 1991
- 4 张毅刚. MCS-51 单片机应用设计. 哈尔滨: 哈尔滨工业大学出版社, 1990
- 5 张毅刚. MCS-51 单片机应用设计. 哈尔滨: 哈尔滨工业大学出版社, 1997
- 6 张毅刚. 新编 MCS-51 单片机应用设计. 哈尔滨: 哈尔滨工业大学出版社, 2003
- 7 张毅刚. 单片微机原理及应用. 西安: 西安电子科技大学出版社, 1994
- 8 张毅刚. 8098 单片机应用设计. 北京: 电子工业出版社, 1993
- 9 张毅刚. 自动测试系统. 哈尔滨: 哈尔滨工业大学出版社, 2001
- 10 张毅刚. MCS-51 实用汇编子程序设计. 哈尔滨: 哈尔滨工业大学出版社, 2003
- 11 徐君毅等. 单片微型计算机原理及应用. 上海: 上海科学技术出版社, 1988
- 12 涂时亮. 单片机软件设计技术. 重庆: 科学文献出版社重庆分社, 1987
- 13 陈粤初等. 单片机应用系统设计与实践. 北京: 北京航空航天大学出版社, 1991
- 14 何立民. MCS-51 单片机应用系统设计. 北京: 北京航空航天大学出版社, 1990
- 15 李华. MCS-51 系列单片机实用接口技术. 北京: 北京航空航天大学出版社, 1993
- 16 何立民主编. 单片机应用系统的功率接口技术. 北京: 北京航空航天大学出版社, 1993
- 17 何为民. 低功耗单片微机系统设计. 北京: 北京航空航天大学出版社, 1994

- 18 何立民. 单片机应用技术选编. 北京:北京航空航天大学出版社,1993
- 19 王毅. 单片机器件应用手册. 北京:人民邮电出版社,1995
- 20 何立民. 单片机应用技术选编. 北京:北京航空航天大学出版社,1996
- 21 房小翠. 单片机使用系统设计技术. 北京:国防工业出版社,1999
- 22 胡汉才. 单片机原理及其接口技术. 北京:清华大学出版社,1996
- 23 王幸之. 单片机应用系统抗干扰技术. 北京:北京航空航天大学出版社,2000
- 24 李广弟. 单片机基础. 北京:北京航空航天大学出版社,2001
- 25 杨振江. 智能仪器与数据采集系统中的新器件及应用. 西安:西安电子科技大学出版社,2001