高等学校计算机专业规划教材

计算机接口技术

王荣良　主编
孙德文　主审

Publishing House of Electronics Industry

北京 · BEIJING

内 容 简 介

本书以 CPU 为核心，通过论述计算机内各部件之间的关系，特别是它们与 CPU 之间的关系，从微型计算机的角度阐述了计算机基本工作原理、各种 I/O 接口以及常用 I/O 设备的结构与连接使用方法。全书共分 8 章，每章都附有习题。

本书有较好的系统性，既注重基本原理的讲解，又兼顾新技术，叙述力求深入浅出。本书可作为高等学校计算机专业教材，也可作为从事计算机应用的其他专业或工程人员的参考用书。

图书在版编目（CIP）数据

计算机接口技术/王荣良主编．—北京：电子工业出版社，2003.1
高等学校计算机专业规划教材
ISBN 7-5053-8190-3

Ⅰ.计… Ⅱ.王… Ⅲ.微型计算机—接口—高等学校—教材 Ⅳ.TP360.47

中国版本图书馆 CIP 数据核字（2002）第 097896 号

责任编辑：张云怡
印　　刷：
出版发行：电子工业出版社　http://www.phei.com.cn
　　北京市海淀区万寿路 173 信箱　邮编 100036
经　　销：各地新华书店
开　　本：787×1 092　1/16　印张：15.25　字数：390 千字
版　　次：2003 年 1 月第 1 版　2003 年 1 月第 1 次印刷
印　　数：5 000 册　定价：20.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。
联系电话：（010）68279077

前　言

计算机接口技术是计算机专业的核心课程，也是自动化控制及其他相关专业的必修课程。本课程的目标在于通过介绍微型计算机的基本结构和工作原理，重点阐述微型计算机内部各部件之间的接口关系以及微型计算机与外围设备的接口关系。通过对本课程的学习，学生应能对常规微机系统以及接口电路进行分析和设计。

由于本课程与微型计算机的发展有着紧密的联系，鉴于目前计算机技术的飞速发展，考虑到教学内容的相对稳定，本书的编写是以通用的微型计算机系统的结构为主要线索，以 Intel 系列 CPU 为典型 CPU，以 PC 系列微机及其外部设备作为具体实例，阐述接口的一般工作原理和性能特点。为了适应目前计算机技术飞速发展的需要，尽量避免所学知识陈旧，本课程教学以能力培养为主，注重基础知识和基本理论，并在此基础上介绍流行产品的具体实现，增强学生对新技术的综合分析能力和应用能力。

本书共分 8 章，可分为 3 大部分：微处理器部分、存储器接口部分和 I/O 接口部分。“微处理器”部分从一般的 CPU 到具体的 Intel 系列产品，在 CPU 内部结构和外部引脚功能两方面阐述 CPU 在计算机中的作用和核心地位。“存储器接口”部分重点针对半导体存储器构成的主存，通过对芯片外部特征的学习，解决 CPU 与存储器的连接问题。“I/O 接口”部分涉及了大量的知识点，包括 CPU 与 I/O 的数据交换方式、各种形式的 I/O 接口电路、常用的 I/O 接口芯片以及总线标准等，这部分知识既为深入理解计算机接口的工作原理提供了帮助，也为实际的开发应用提供了基础。

本书是编者在多年从事高校“接口技术”课程实践的基础上编写的，在编写过程中力求语言文字通俗易懂，叙述深入浅出。为了使学生加深对课程内容的理解，书中配有一定量的例题与习题。在本书的编写过程中，张蕙老师、陈昶老师等在资料收集和整理方面做了很多工作，并参与了部分章节的编写。黄国兴教授对本书大纲制定提出了很多具体指导意见。本书由孙德文教授担任主审，孙教授对书稿进行了认真的审阅，并提出了许多宝贵意见。在此一并向他们表示感谢！

由于计算机技术的不断更新，新名词层出不穷，加上本人水平有限，书中定会存在不足之处，恳请广大读者批评指正。

编　者

2002 年 12 月

目　　录

第 1 章　概　　述

微型计算机是计算机领域中发展最迅速、应用最广泛的一种。自 20 世纪 70 年代以来，微型计算机经历了一系列的更新换代后，其应用已经深入到了人们生产、生活、学习等领域。并且，以微处理器为核心的各种智能控制系统也展现出了广阔的应用前景，许多过去很难实现的功能在微型计算机的参与下都能得以实现。微型计算机的发展与普及对计算机的发展起着重要的作用，也对人们的社会活动起着重要的作用，学习与研究微型计算机及接口技术也有其实际意义。

1.1　微型计算机的基本结构

1.1.1　微处理器与微型计算机

计算机（Computer）由 5 大部分组成：运算器、控制器、存储器、输入设备和输出设备。运算器用于完成算术运算和逻辑运算；控制器根据指令代码完成译码和控制工作；存储器用于存储运行的程序代码和需要加工的数据及运算结果；输入设备用于为计算机提供程序和需要加工的原始数据；输出设备用于输出数据的加工结果。如图 1-1 所示，计算机在控制器的控制下，通过依次执行存放在存储器中的指令，完成程序所规定的工作。

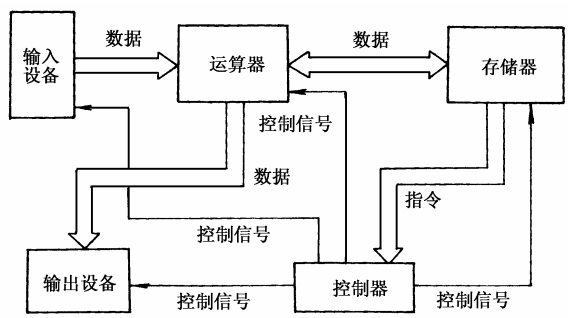


图 1-1　计算机结构

CPU（Central Processing Unit）即中央处理器，是计算机的核心。CPU 由运算器和控制器组成，用于实现对计算机的控制和运算工作。

计算机主机是指 CPU 和存储器中内存储器的组合。与主机相对应，计算机的外围设备是指计算机的输入设备、输出设备和存储器中的外存储器部分。

微处理器（Microprocessor）是微型计算机的核心。尽管各微处理器的性能指标不同，但具有相同的基本功能：

（1）可以进行算术运算和逻辑运算；

（2）可以保存少量数据；

（3）能对指令进行译码并执行规定的动作；

（4）能和存储器、外设交换数据；

（5）提供微型计算机所需要的地址和控制信号；

（6）可响应来自其他部件的中断请求以及对其他输入控制的处理。

与计算机的中央处理器相似，微处理器由运算部件、寄存器组、控制部件和内部数据总线组成。

微处理器内部的运算部件是专门用来处理各种数据信息的，可以进行加、减、乘、除等算术运算和与、或、非、异或等逻辑运算。较低档的微处理器不具有乘、除运算功能，可以通过程序来实现。

寄存器组主要用于暂存参加运行的数据以及运行的中间结果，这些寄存器可以与内存或 I/O 交换数据，也可为算术、逻辑运算单元提供运算数据以及存放运算结果。在寄存器组中，还有若干具有特殊用途的寄存器，如有的寄存器用于存放地址，有的用于完成各种寻址方式。

控制部件由指令寄存器、指令译码器及时序与控制逻辑电路组成。指令寄存器用于存放当前执行的指令代码供指令译码器译码。指令译码器产生的相应控制信号送到时序和控制逻辑电路，从而组合成微机系统包括微处理器外部所需要的时序和控制信号，以控制微型计算机各部件的协调工作。

内部数据总线为微处理器内部各部件之间的数据传送以及微处理器与外部存储器或 I/O 接口的数据交换提供了通道。

显然，微处理器与微型计算机是两个不同的概念，但必须指出的是，微型计算机与微型计算机系统也是两个不同的概念。图 1-2 所示为微处理器、微型计算机、微型计算机系统三者之间的关系。

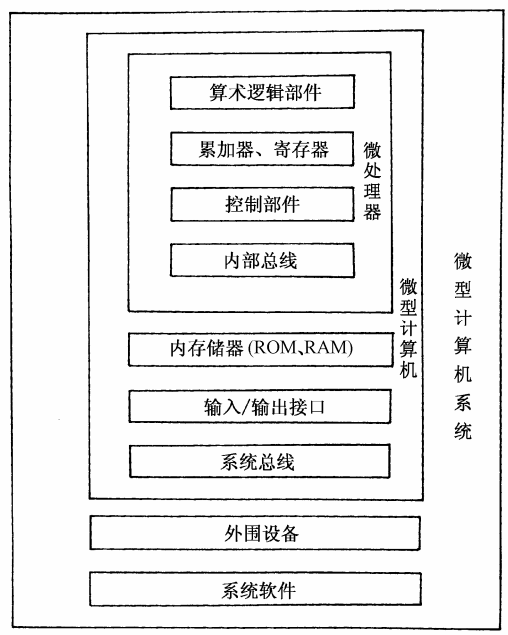


图 1-2　微处理器、微型计算机和微型计算机系统

可见，微处理器是集成了运算部件、控制部件、寄存器组、内部数据总线的集成电路芯片。由于微处理器也称 CPU 或μP，在本书以后的章节中，无特别的说明，CPU 就指微处理器。

微型计算机（Microcomputer）是指以微处理器为核心，配以内存储器（ROM 和 RAM）、I/O 接口以及用于连接的系统总线所组成的。显然，作为一个微型计算机，并不能直接使用。

微型计算机系统（Microcomputer System）是指由微型计算机配以相应的外部设备及其专用电路、电源、机架以及足够的系统软件所构成的系统。外部设备用来实现数据的输入/输出，包括 CRT 显示器、键盘、磁盘及磁盘驱动器和打印机等。系统软件包括操作系统和一系列系统应用程序，有了系统软件，才能发挥微型计算机系统中的硬件功能，并为用户使用计算机提供了方便。因此，人们通常使用的微机，严格地说是微型计算机系统。

1.1.2 微型计算机的典型结构

如上所述，微型计算机是以微处理器即 CPU 为核心，通过系统总线连接内存储器和 I/O 接口电路而构成的，如图 1-3 所示。

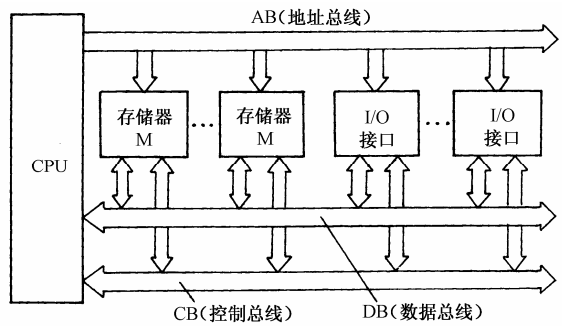


图 1-3 微型计算机结构

系统总线是一个公共的信息通道。微型计算机采用了总线结构，这种结构可以使得系统内部各部件之间的相互关系变为各部件之间面向总线的单一关系。一个部件只要符合总线标准，就可以连接到采用这种总线标准的系统中，使系统功能得到扩展。如图 1-3 所示，存储器模块通过总线与 CPU 相连，对存储器而言，只要拥有相同的总线接口标准，就可以很方便地通过系统总线连接到 CPU 上，从而扩充微型计算机的内存量；同样，CPU 通过 I/O 接口电路与外设相连，增加 I/O 接口电路意味着可以增加外设。因此，微型计算机采用的总线结构是一种有利于系统扩充的体系结构。

尽管各种类型的微型计算机的总线类型和标准有所不同，但大体上都包含了 3 种不同功能的总线：地址总线 AB（Address Bus）、数据总线 DB（Data Bus）和控制总线 CB（Control Bus）。

地址总线是专门用来传送地址信息的。因为地址总线是由 CPU 发送的，所以它是单向的。地址总线的位数决定了 CPU 可以直接寻址的内存范围。例如，某一微机的地址总线为 16 位，则表示该 CPU 所能寻址的最大内存容量为 2^{16}＝64KB。

数据总线用于传送数据。和地址总线不同的是，数据总线是双向总线，数据可以从其他部件传送到 CPU，也可以由 CPU 传送到其他部件。数据总线的位数与 CPU 的字长相对应，是一个很重要的指标。和其他类型的计算机一样，在微型计算机中，数据的含义也是广义的，数据总线上传送的不一定是作为计算机加工对象的数据，也可以是指令代码、状态码或者控制码。

控制总线用来传输控制信号，其中包括 CPU 送往存储器和 I/O 接口电路的控制信号，如读信号、写信号等；还包括其他部件送往 CPU 的信号，如时钟信号、中断请求信号等。

CPU 通过系统总线与存储器和 I/O 接口相连，也通过系统总线对存储器或 I/O 接口进行访问。当 CPU 在地址总线上提供存储器地址，用于选择具体的存储器单元，向控制总线提供存储器读、写控制信号，确定存储器访问的性质后，就可以在数据总线上进行数据交换，完成存储器读或写的操作了。采用同样的操作序列可以完成 CPU 对 I/O 接口的访问。

1.1.3 微机接口技术研究的内容

微型计算机是一种能够自动、高速、精确地完成数学运算和数据处理的电子设备，微型计算机的工作过程是完成对信息的输入、存储、传送、加工及输出的过程。微型计算机之所以能够自动、高速、精确地完成数学运算和数据处理，是因为微型计算机内部的核心部件——微处理器能够通过由程序预先编制好的工作顺序控制整个微型计算机的高速运转。

所谓接口，是指两个部件之间的连接点或边界。作为微机接口，涉及到微处理器与各外部设备之间的接口、微处理器与存储器之间的接口以及微型计算机之间的接口。两个部件之间的连接，包括物理电路上的连接，也包括逻辑上的连接，即信息的交换，如数据、命令、状态信息的交换。作为一个接口电路，实现物理电路上的连接是基础，其最终目标是实现有效的信息交换。

作为“微型计算机原理与接口”，就是以微型计算机为对象，研究它的组织结构以及内部各部件之间的接口关系，研究微型计算机对信息的加工过程和工作原理。

存储器接口涉及了微处理器与内存储器之间的关系以及主机与外存储器之间的连接关系，其中包括信号线的物理连接和信息交换的时序关系。

存储器接口解决微型计算机工作的最基本的要求。输入/输出接口解决微型计算机与外部设备交换信息的问题，有并行接口和串行接口，也有数字接口和模拟接口，既要考虑信号线的物理连接，又要考虑其时序关系。由于外部设备的多样性，微处理器在与外界进行信息交换时，还要考虑到采用不同的数据传送方式。

1.2 微型计算机的分类与发展

从 1946 年第一台电子计算机诞生以来，计算机的组织结构、表现形态及应用范围都发生了很大的变化。就微型计算机而言，尽管只有 20 多年的历史，但同样也反映了计算机的发展状况。了解微型计算机的分类，了解微型计算机的发展史，对掌握微型计算机的原理与结构、实际应用与发展趋势具有指导作用。

1.2.1 微型计算机的分类

经历了 20 多年的发展，微型计算机不仅数量众多，而且表现形式也各不相同。对微型计算机进行分类，可以从不同的角度进行。

1．按字长划分

传统的划分方法是根据所使用的微处理器的字长进行划分的，目前微型计算机可分为 4 位机、8 位机、16 位机和 32 位机。

4 位机中使用的字长为 4 位的微处理器，由于可以方便地处理 BCD 码，因此曾广泛地应用于电子计算器中。目前，随着对 4 位微机的指令系统、存储容量、输入/输出能力和运行速

度等方面性能的改善，4 位微机作为各种控制器已广泛应用于电子仪器、家用电器等应用领域。

8 位机在 20 世纪 80 年代初期和中期有着广泛地应用。由于 8 位微机可以很方便地表示字符和数字信息，运行速度较快，有较多的硬件支持和软件积累，可以配有操作系统和各种高级语言，适合于一般的数据处理。

16 位机的运行速度和数据处理能力明显强于 8 位机，并可配有功能强大的操作系统和多种高级语言，可以进行大量的数据处理的多任务控制。16 位机的性能已超过了过去的小型计算机。32 位机在系统结构、元器件技术等方面有很大的进展，其性能大大优于其他机种。目前，32 位机不仅用于过程控制、事务处理、科学计算等领域，还可以很好地工作于声音、图像处理等多媒体应用以及计算机辅助设计、计算机辅助制造等大数据量的应用领域。

2．按规模划分

按微型计算机的组织结构和规模，可分为单片机、个人计算机及工程工作站。

单片机是一种把能组成微型计算机的基本功能部件如微处理器、部分存储器、部分 I/O 接口以及定时器等集成在一片集成电路芯片上所构成的计算机。有的单片机还包含数/模转换器和模/数转换器。单片机具有体积小、功耗低等特点，主要应用于智能仪器、仪表及控制领域。因此，单片机也称微控制器。目前，Intel 公司、Motorola 公司等厂家都有大量的单片机系列产品。

个人计算机是指通常意义上的微型计算机系统，由计算机主机和键盘、显示器、鼠标、打印机等常用外部设备及系统软件组成。各种型号的微机性能差异很大，分别适用于家用、商用、教育等各种不同的应用领域。

工程工作站是一种微型化的功能强大的计算机，由高性能主机（包括高速处理器和大容量内存）、高分辨率显示器、快速的 I/O 设备及其他必要的仪器设备所组成，综合微型机和大型机的优点，既有速度快、内存大等特点，又有小巧灵活、轻便价廉等优点。工程工作站本身可作为一台计算机使用，能完成工程任务，又可以作为一个工作站联网。工程工作站特别适用于工程上的设计、计算、计划、模拟、分析以及各类常规的和非常规的数据处理。随着微机技术的发展和性能的提高，个人计算机与工程工作站之间的界线已越来越不明确。

3．按应用划分

根据应用范围和表现形式，微型计算机可分为通用计算机和专用计算机两种。

通用计算机是指传统意义上的微型计算机系统，具有基本的计算机结构与配置，体现通常的计算机功能。用户加载具体的应用软件以后，就可以完成相应的功能。根据需要，用户还可以在通用计算机上添加特定的硬件和相对应的软件，就可以让计算机完成特定的功能。

专用计算机是指为完成某一特定功能的计算机系统。这类计算机具有固定的用途，往往附属于某一具体的应用设备。作为专用计算机，有关计算机的功能通常不需要、也不可能由用户来随意添加或删除，而计算机的表现形式也不像一般的通用计算机。一般许多自动化程度很高的工作设备、仪器、仪表，甚至家用电器中都嵌有专用计算机。

1.2.2 微型计算机的发展

微型计算机的性能，从很大程度上是由微处理器的性能来代表的。在微型计算机 20 多年的发展与使用过程中，人们已经习惯用微处理器的型号来称呼使用这一型号微处理器的微型计算机了。因此，从某种意义上说，微处理器的发展史，也是一部微型计算机的发展史。

1．微处理器的发展

第一个微处理器是 1971 年美国 Intel 公司生产的 4004。Intel 4004 是一个 4 位微处理器，本来是为高级袖珍计算器设计的，但已经具备了一个微处理器所具有的体积小、重量轻、价格低廉等基本特点，在设计生产以后，取得了意外的成功。于是 Intel 公司对 Intel 4004 微处理器做了改进，正式生产了通用的 4 位微处理器 Intel 4040。

按传统的以微处理器的“字长”来划分微处理器发展的“代”，可分为四代。

第一代微处理器是由 4 位微处理器和低档 8 位微处理器为代表的，在 1971 年～1973 年期间设计生产，典型的产品有 Intel 4004、4040、8008 等。Intel 8008 是一个 8 位微处理器，是 Intel 公司于 1972 年设计生产的，这是因为 Intel 公司在 Intel 4040 芯片因其体积小、价格低廉、通用性强等特点引起许多部门和机构的兴趣和注意以后推出的一个升级芯片。Intel 公司一系列微处理器产品的推出，推动了微处理器芯片技术的发展，随后，出现了如 Zilog、Motorola 等许多从事微处理器的开发与生产的公司。

第二代微处理器是 1974 年～1978 年间设计生产的 8 位微处理器。在这一期间，处理器的设计生产技术已经相当成熟，同时配套的各类器件也很齐全。这一时期许多厂家设计生产了许多型号的微处理器，其中设计最成功、应用最广泛的是 Intel 公司的 8080/8085，Zilog 公司的 Z80，Motorola 公司的 6800/6802 和 Rockwell 公司的 6502。每一系列的微处理器产品不仅在性能和质量上有很大提高，同时各生产厂家为每一系列的微处理器配套设计生产了大量的外围集成电路芯片，如与 Z80 微处理器配套使用的 Z80PIO（并行接口芯片）、Z80SIO（串行接口芯片）、Z80CTC（定时器芯片），与 Intel 8085 配套使用的 Intel 8255（并行接口芯片）、Intel 8259（中断控制器）、Intel 8279（键盘显示控制器）等，这样，用户采用微处理器芯片和相应的外围接口芯片就可以很容易地构造微型计算机。从第二代起，提高集成度、提高功能与速度、增加外围接口电路的功能与种类成为微处理器发展的基本方向。

第三代微处理器的推出介于 1979 年～1981 年之间，这是一个 16 位微处理器的时代。在这一期间，超大规模集成电路工艺已经成熟，一片硅片上可以容纳几万个晶体管，16 位微处理器的功能已经可与过去中档小型计算机相比。其中，有代表性的三种芯片是 Intel 公司的 8086/8088、Motorola 公司的 M68000 和 Zilog 公司的 Z8000。众所周知，Intel 8088 微处理器在 1980 年被选作为 IBM PC 微型计算机的 CPU，IBM PC 微型计算机的诞生，对世界计算机技术的发展有着重大的影响。

1982 年以后，出现了 32 位微处理器芯片，进入了第四代微处理器时代。典型的代表产品有 Intel 公司的 Intel 80386、Motorola 公司的 MC68020 等。32 位微处理器经历了 10 多年的发展，无论在微处理器芯片本身的性能方面，以及与微处理器配套使用的外围接口芯片的开发方面，都有了很大的发展。

从构成微处理器芯片电路的集成度方面也可以反映出微处理器的发展状况。早期的微处理器芯片集成度大约仅在每片几千个晶体管，如最早的 Intel 8008，集成度仅在 2000 管/片左

右。以后微处理器芯片的集成度有了很大的提高，几乎每两年提高 1 倍。进入 32 位微处理器时代后，集成度都在每片芯片十万个晶体管以上。例如，标准 Intel 80386 微处理器芯片是一块 0.5×0.5 平方英寸的正方形硅片，其中集成了多达 275 000 个晶体管；Intel 80486 微处理器芯片是一块 0.4×0.65 平方英寸的长方型硅片，其中集成了 1 200 000 个晶体管；至于 Pentium 微处理器，是一块集成了 3 100 000 个晶体管的约 1 平方英寸大小的芯片。如今的微处理器芯片，集成度都在百万以上。

与集成度的发展相对应，微处理器的运行速度也迅速提高。以主频为例，最初的微处理器主频仅为 1～2MHz，作为典型的 8 位和 16 位微处理器，其主频值也只是在 10MHz 左右。32 位微处理器出现以后，特别是 Pentium 芯片的出现，主频从 33 MHz、75 MHz、100 MHz、200 MHz 直到如今超过 1 000 MHz 的微处理器芯片。

2．微型计算机的发展

微处理器性能的快速提高，得益于集成电路技术的飞速发展和大量新技术的涌现，包括中型、大型计算机技术在微处理器中的应用。微处理器性能的提高，也迅速提升了微型计算机的性能，推动了微型计算机技术的发展。

尽管在 20 世纪 70 年代已经有了微型计算机，但微型计算机得到广泛应用并且由此带动的微型计算机技术的迅速发展的起点还是 Apple 公司的 Apple Ⅱ计算机和 IBM 公司的 IBM PC 计算机。

1976 年创办的 Apple 公司在微型计算机发展史中起着不可磨灭的作用，Apple 公司从 1977 年推出 Apple Ⅱ以后，在美国以至世界微机市场上占有极大的份额。Apple 公司的成功，使一些以前专营中小型计算机或大型计算机的公司，也开始转向微型计算机的研制和开发工作。

特别值得一提的是“蓝色巨人”IBM 公司于 1981 年 8 月推出的 IBM PC 计算机采用了与以往 IBM 公司研制大中型计算机产品所不同的方法，在研制 IBM PC 机中广泛采用了现成的技术设备以及 IBM 公司以外的技术力量，如 IBM PC 计算机中采用了 Intel 公司的 Intel 8088 微处理器及其外围接口电路、采用了 TANDON 公司的磁盘驱动器、采用了 EPSON 公司的打印机、采用了 Microsoft 公司的操作系统，仅在 1 年多的时间内就完成其研制工作。1983 年 3 月推出了 IBM PC/XT 计算机，1984 年 8 月推出 IBM PX/AT 计算机。由于微型计算机本身所拥有的优点，加上 IBM 公司公开了 IBM PC 计算机的硬件和软件资料从而使用户很容易进行进一步的开发，以 IBM PC 为基本框架的 PC 机时代在近 20 年中发展迅猛，PC 机几乎影响到了人们的生产和生活方式。PC 机对计算机技术的发展和计算机应用的普及起到了重要的作用。

3．单片机的发展

作为在控制和信息家电中的应用，集微处理器和其他接口电路于一体的单片机也随着微处理器的发展而同步发展。作为微处理器的主要生产厂家，Intel 公司在 1976 年推出了第一代 8 位通用单片机 MCS-48 系列，1980 年相续推出第二代 8 位增强型单片机 MCS-51 系列，1983 年又推出 16 位单片机 MCS-96 系列，近年来已有 32 位单片机产品。

Motorola 公司于 1979 年生产了与 MC6800 兼容的 8 位单片机 MC6801，随后又生产了 MC6805，以后，Motorola 公司分别在这两个系列的单片机基础上进行发展。在 MC6801 的

基础上先后生产了性能更好、功能更强的系列产品，如 MC68701、MC68HC11 等；在 MC6805 的基础上，先后生产了价廉的 MC6804 以及高速度、低功耗的 MC68HC05 系列。另外，Motorola 公司还生产了 16 位的微控制器 MC68HC16 和 32 位微控制器 MC68322。

1.3 微型计算机的性能评价

评价一个微型计算机系统的性能优劣，需要从多方面进行综合考虑。但针对一般计算机的使用人员来说，一台计算机性能的好坏，主要是要看这台计算机的数据处理能力，包括运算速度、存储容量等。此外，系统的可靠性、通用性乃至价格都是评价一台计算机系统优劣的性能指标。

1.3.1 微处理器的性能

在评价一台微型计算机系统的性能时，微处理器的性能起着很重要的作用。微型计算机的发展，也是随着微处理器的发展而发展的，从目前对微型计算机的型号命名习惯上也常以微处理器的型号命名这一点就可以看出微处理器在微机系统中的地位。随着微处理器技术的发展，其结构也发生了很大的变化，可以从多角度来评价微处理器的性能，但最基本的评价指标还是字长与运算速度。

1．字长

在一台微型计算机系统内部，微处理器的性能往往在一定程度上反映了微型计算机系统的性能，甚至有的微机型号也是用微处理器的型号来表示的，如 486 微机、586 微机等。作为字长，即 CPU 中运算器一次能处理的最大数据位数，同样也是反映微机系统数据处理能力的重要技术指标。常见的字长有 8 位、16 位和 32 位等。字长越长，说明系统的运算精度越高，数据处理能力越强。

与字长相对应，总线的宽度，特别是数据总线的宽度同样也能反映系统性能。数据总线的宽度只有与 CPU 的字长相当，才能有效发挥出 CPU 数据处理的能力。除此以外，总线的数据传输速率等技术指标也对系统总体性能评价起一定的影响。

2．运算速度

运算速度的高低是衡量计算机系统的一个重要性能指标。无论是计算机系统，还是其核心 CPU，都是在追求高速度。主频反映了 CPU 的速度，在同一类 CPU 中，频率越高，CPU 的运算速度越快。主频的单位是兆赫兹（MHz），目前微机系统的主频都在 100MHz 以上。

反映微机系统运算速度的另一个单位是 MIPS，即每秒执行百万条指令数。显而易见，数值越大，计算机的速度越高。但在用 MIPS 衡量一个计算机系统的速度指标时，要注意测试 MIPS 所使用的是指令集中的哪些指令。因为不同指令的复杂程度不同，如执行一条加法指令与执行一条乘法指令所用的时间明显不同。

1.3.2 存储器的性能

存储容量反映了计算机系统所能存储的信息量。计算机的存储器系统主要分为内存储器

和外存储器两种。无论是内存储器还是外存储器，作为一个计算机系统，都希望能够存放尽可能多地数据，能够尽可能快地得到这些数据。

1．存储容量

存储器最基本的容量单位是字节（Byte）。由于存储器不仅用于长期存储信息，还为 CPU 加工信息提供场所，所以存储容量的增大，对提高系统的运行速度也有很大影响。

内存储器的主要作用是存放当前需运行的程序和加工的数据。通常衡量内存容量大小的单位是 MB。

外存储器的主要作用是为内存储器提供后备的程序和数据。衡量外存容量大小的单位通常是 GB。

2．存取速度

存储器的存取速度也称访问速度，也是衡量存储器性能的重要指标。在计算机运行时，有大量的存储器访问操作，存储器的存取速度直接影响到整个计算机系统的运行速度。如果微处理器的运行速度很快，但没有相对应的存储器访问速度，整个系统性能仍很难得到较好的改善。

目前，用于内存的半导体存储器芯片的存取速度大部分都是几十纳秒（ns）；在外存储器中，硬盘常用转速来衡量访问速度，光盘常用倍速来衡量访问速度，以 150MB/s 为基本速率，常有 24 倍速光驱、40 倍速光驱等。

1.3.3 I/O 设备的性能

I/O 设备多种多样，不同的设备有不同的评价指标。对于常用外设，其性能指标有速度、分辨率和颜色深度等。

1．速度

与其他微型计算机部件一样，I/O 设备速度的快慢影响着计算机最终的使用效益，追求高速度是共同的。在与人交互的常用外设中，大部分都能满足人的响应速度。相比较而言，打印机的响应速度较慢。对于以前常用的针式打印机，打印速度都是以每分钟打印多少字符来描述的。而目前常用的激光打印机和喷墨打印机则都习惯以用每分钟打印多少页来衡量。

2．分辨率

分辨率指标主要针对显示器、打印机、扫描仪等外围设备。对于显示器而言，分辨率是指屏幕上所显示出来的像素数目。显然，像素数目越多，分辨率越高。通常的描述是水平像素数乘垂直像素数，如 800×600、1024×768 等。

对于打印机和扫描仪，常用 DPI 来描述分辨率，DPI 是指每英寸的点数，如 600DPI、800DPI 等。

3．颜色深度

颜色深度是指外部设备所能支持的颜色数。显然，支持的颜色数越多，表现的颜色就越丰富，也就越接近真实感觉。无论显示器还是打印机，颜色深度都是以显示卡或打印机中描

述颜色的位数来决定的，以 2 为底数而以颜色位数为指数就能计算其颜色数，如 8 位颜色位的显示卡能显示 2^8（256）种颜色。

1.3.4 系统其他性能

1．通用性

通用性也称兼容性，是指一个微型计算机系统在软、硬件等方面的适应性。系统是否具有标准的硬件接口并能与绝大部分的通用外设相连接，以及系统是否能运行众多的系统软件和应用软件，都反映了系统的通用性。除了一些特殊的应用要求以外，微机系统的通用性越好，使用越方便。

2．稳定性

系统的稳定性决定了系统是否实用。一个计算机系统，无论其功能如何强大，只要其性能不稳定，其强大的功能也就无法体现出来。或者说，稳定性是系统功能的部分反映。

通常，衡量系统稳定性有两个性能指标：可靠性和可维性。

可靠性反映了系统能连续无故障工作的时间长短。理想状态是系统连续工作时间趋向于无穷大，这样系统永无故障。一般系统的运行规律是：系统正常运行、故障并修复、系统再正常运行、再出现故障并修复。随着系统的老化，系统每次出现故障并修复以后能正常运行的时间会越来越短，直至系统报废。通常，可靠性是用系统的平均无故障时间来评价的，平均无故障时间越长，可靠性越好。一般一个系统要获得高可靠性，在系统的硬件设计与软件设计中都要花费较大的代价。

系统的可维性反映了一个系统是否方便维护与维修。一个计算机系统能保证长时间不出故障很重要，但不能做到永远不出故障。因此，一旦出故障能在很短的时间内修复也就显得很重要。故障恢复时间越短，则因系统故障所造成的损失就越小。通常可维性也是用时间来衡量的，即一个系统故障修复所需要的平均时间。

3．软件性能

作为一个计算机系统，硬件系统提供了程序运行的基本环境，其功能的最终实现是需要由软件来完成的。优越的硬件性能只是一个基础，只有配置优质的软件系统才能使计算机性能得以充分发挥。由于软件系统有专门的评价指标和方法，因此不在这里详细阐述。

4．价格

高性能是计算机系统所追求的目标，同样，在价格上则希望相对低廉。衡量一个系统往往用性能价格比这一综合指标。性能价格比的取值反映了单位费用开销所能获得的实际功能的大小。在这里，性能既包括硬件性能又包括软件性能和使用性能等；而价格也不仅是指硬件价格，还要包括软件费用、维护费用等。

本 章 小 结

本章主要介绍 3 部分内容。首先介绍有关微型计算机的基本概念，包括微处理器、微型

计算机、微型计算机系统的定义。在此基础上又介绍了微型计算机的发展和分类，最后介绍了微机系统的性能评价方法。这些知识的掌握，对学习使用微型计算机是很有帮助的。

习 题 1

1.1 试说明微处理器（即微型计算机的 CPU）内部是由哪几部分组成的，并进一步阐述每一部分的功能。

1.2 试区分微型计算机与微型计算机系统这两个不同的概念。

1.3 通常，在微型计算机的系统总线上需要传送哪些信号？在总线上如何区分这些信息？

1.4 某一微型计算机的地址总线宽度为 20 位，则该计算机内存寻址范围应为多少？

1.5 如果 CPU 需要将其内部寄存器中的数据存放到存储器的某一个存储单元，结合微型计算机的结构，试说明这一操作的执行过程。

1.6 除了教材中所讨论的对微型计算机的分类方法以外，是否还能列举一些其他的分类方法。

1.7 举例说明生活中所见到的专用计算机。

1.8 如果你要购买一台微型计算机，会考虑哪些因素？

第2章　微 处 理 器

微处理器即微型计算机的CPU，是微型计算机的计算与控制中心。微处理器是通过执行程序来完成对微机系统的控制和数据加工的。传统的CPU由运算器和控制器两大部分组成，完成指令控制、操作控制、时间控制、数据加工的功能。目前，随着高密度集成电路技术的发展，早期放在CPU芯片之外的逻辑功能部件，如浮点运算器、高速缓存Cache等都移入了CPU内部，超标量技术、流水线技术等先进技术也引入到了CPU之中。因此，微处理器的发展典型地反映了微型计算机技术的发展。

2.1　微处理器的结构与时序

对微处理器的认识，应从两个方面来实现：内部结构与外部功能。了解内部结构，可以认识微处理器的工作原理；了解外部引脚及其相应的功能，可以认识微处理器如何与其他部件相连接。随着计算机技术的飞速发展，微处理器的产品更新速度也非常快。面对不同型号的微处理器，不同公司生产不同的微处理器系列，了解和掌握新技术是必要的，但认识微处理器的基本结构与功能也是必要的基础。

2.1.1　微处理器的基本结构

组成微处理器的最基本的部件是运算部件、控制部件、寄存器组和内部数据总线。目前，最先进的微处理器也是以这四个部件为基本部件，综合其他技术所构成的。如图2-1所示为一个典型的8位微处理器的内部结构。所谓8位微处理器，其基本特征是内部算术逻辑运算单元是8位的，内部数据总线是8位的，内部通用寄存器主要是8位的。下面以这个微处理器为例说明各功能部件的作用。

1．运算部件

如图2-1所示，算术逻辑单元（ALU）、累加器、锁存器、暂存寄存器及标志寄存器和十进制调整电路等构成了微处理器的运算部件。

从累加器存入锁存器的数据和暂存器中的数据通过ALU进行运算，结果通过内部数据总线存回累加器，或通过内部数据总线写入寄存器组中的某一个寄存器，也有可能输出到存储器或I/O接口。运算结果将影响标志寄存器和十进制调整电路，并对下一次运算产生作用。

参加运算的累加器和暂存寄存器中的数据，可能通过内部数据总线来自于寄存器组，也可能来自于CPU外部的存储器或I/O接口。

2．控制部件

控制部件由指令寄存器、指令译码器、定时与控制电路组成。

指令寄存器用于存放当前执行的代码。在取指令码阶段，微处理器将存于内存的指令读入微处理器内部的指令寄存器，以便进行分析译码。

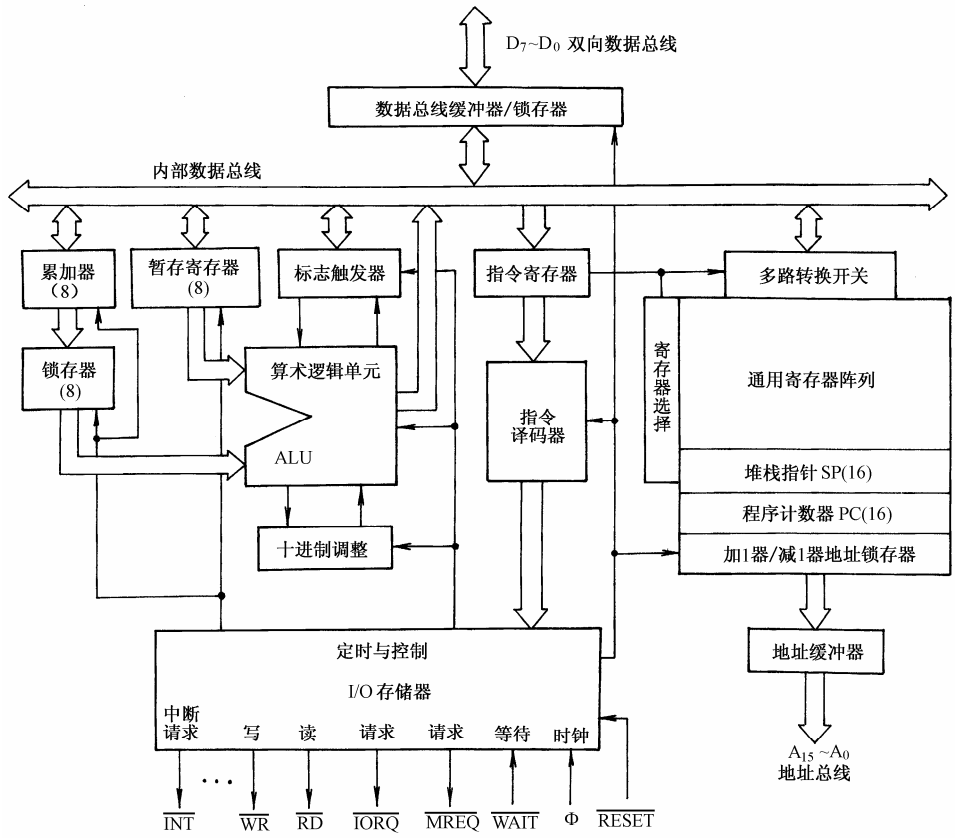


图 2-1　微处理器内部结构

指令译码器的功能是对存放于指令寄存器中的代码进行译码，从而产生表示当前指令功能如加法指令、数据传送指令、寄存器寻址操作等的控制码，并将这些译码结果传送给定时与控制电路。

定时与控制电路的功能是产生一系列的控制信号，这些控制信号分为两类，一类用于微处理器内部的控制，如对 ALU 提供控制以确定其运算类型，对累加器提供控制以确定累加器何时输出数据、何时锁存数据；另一类控制信号如读信号、写信号通过微处理器的控制总线输出到存储器或 I/O 接口，用于完成微处理器与外部的数据交换。

定时与控制电路产生各种控制信号的依据来自 3 个方面：指令译码器的控制码、时钟发生器的定时信号、状态寄存器或微处理器外部的状态反馈。

来自指令译码器的控制码可以通过定时与控制电路确定产生什么控制信号，如遇到加法指令，定时与控制电路会产生加法的控制信号，要求 ALU 执行加法操作。

定时信号确定了定时与控制电路的时序关系，也就确定了输出的各种控制信号的先后顺序，如加法操作的执行，先对锁存器和暂存寄存器发控制信号以提供运算数据，接着向 ALU 发加法控制信号以完成加法运算，最后向累加器发控制信号存入 ALU 的运算结果。

状态反馈会影响到控制信号的产生。如条件转移指令的功能是根据当前状态寄存器的某一状态位的值来确定是否执行转移操作。对于定时与控制电路来说，则是根据状态寄存器的值来确定定时与控制电路是否向程序计数器发控制信号，对程序计数器的值进行修改。

3．寄存器组

寄存器组包括通用寄存器和专用寄存器。

通用寄存器是在微处理器内部存放运算数据和中间结果的场所，通用寄存器可以通过内部数据总线与微处理器外部交换数据，可以为运算部件提供操作数或操作数地址，也可以在寄存器之间相互交换数据。

程序计数器 PC 和堆栈指针 SP 都属于专用寄存器。程序计数器 PC 存放着存储器地址，用于确定当前执行代码所在的位置。每取一次指令代码，PC 值加 1，指向下一内存单元。SP 作为堆栈指针，随着压栈操作和退栈操作而变化。

在寄存器组中与寄存器相关的还有寄存器选择电路和多路转换开关电路，这些电路在控制部件的作用下，以指令代码中的寄存器编码为依据选择某一个寄存器，并通过多路转换开关与内部数据总线交换数据。

4．内部数据总线

内部数据总线是微处理器内部传送数据的通道。寄存器组与运算部件相互之间以及它们与外部设备之间都是通过内部数据总线来传送数据的。内部数据总线与外部数据总线之间由数据总线缓冲器/锁存器相隔离。相似地，微处理器通过地址缓冲器向外部地址总线输出地址值。

每一条指令的执行都要经历取指令码和译码执行阶段。取指令码操作对于一条指令来说，定时与控制电路会产生相同的控制信号；在执行阶段，定时与控制电路会根据不同的指令产生相应的控制码。

每一条指令的取指令过程大致经历如下操作：

（1）程序计数器 PC 的值通过地址缓冲器输出到外部地址总线；

（2）PC 值加 1 后送 PC；

（3）通过数据总线缓冲器/锁存器将存储器提供的指令代码读到内部数据总线并写入指令寄存器。

以上操作都是在定时与控制电路产生的控制信号的控制下完成的。在指令代码到达指令寄存器以后，就可进行指令译码，并完成相应的指令操作。

如果是运算指令或微处理器内部的数据传送指令，定时与控制电路将要求 ALU 完成相应的操作，要求寄存器完成相应的数据传送。如果在指令执行过程中还需要内存中的其他数据，则微处理器会在控制部件的作用下通过地址缓冲器输出存储器地址，然后通过数据总线缓冲器/锁存器读入操作数据。如果指令执行过程中需要将运算结果写回存储器，则先在地址线上输出地址，然后再在数据线上输出数据。当然，无论是数据输入还是输出，控制部件都必须通过控制线向外部提供读、写控制信号及其他控制信号。

2.1.2 微处理器的基本引脚及功能

微处理器的外部功能是指微处理器的引脚功能。引脚功能可分为两方面：引脚静态功能和引脚动态功能。引脚静态功能是指一个微处理器有多少个外部引脚，每个引脚的具体意义与作用；引脚动态功能就是时序概念，反映微处理器各引脚在什么时刻有效，什么时刻无效，指令的执行过程与微处理器的时序是紧密相关的。

与微处理器外部总线相对应，微处理器的外部引脚分为 3 类：地址线、数据线和控制

线。

1．地址线

如图 2-1 所示，微处理器通过地址缓冲器引出的是地址线。地址信息用来确定微处理器将要访问的具体存储单元或某个 I/O 端口，因此，通常地址线是输出的。地址线的宽度反映了寻址范围，例如 16 根地址线可寻址存储器的范围是 2^{16}=64KB，20 根地址线可寻址存储器的范围是 1MB。

2．数据线

通过数据总线缓冲器/锁存器引出的是数据线。数据线用于传送微处理器与外部交换的信息，包括指令码和待加工数据。数据线的宽度有 8 位、16 位和 32 位等，其宽度是衡量微处理器的重要指标。由于数据在微处理器与存储器以及微处理器与 I/O 之间的传送是双向的，因此微处理器的数据线也是双向的。

3．控制线

地址线引脚用于输出地址值，数据线引脚用于数据交换，而控制线则与控制部件相关联。相对而言，控制线引脚最复杂，不同的控制线其意义完成不同。部分的控制线是输出的，其作用是实施微处理器对外部的控制；部分控制线是输入的，其作用是让微处理器了解外部的状态，以确定进一步的处理要求。常用的控制线及其功能如下：

$\overline{\text{WR}}$ ：写信号，输出。该信号有效表示微处理器当前处于写操作状态，$\overline{\text{WR}}$ 信号要求外部存储器或 I/O 接口接收数据总线上的输出数据。

$\overline{\text{RD}}$ ：读信号，输出。该信号有效表示微处理器正处于读操作状态，$\overline{\text{RD}}$ 信号要求外部存储器或 I/O 接口将选中的数据通过数据总线送到微处理器。

$\overline{\text{MREQ}}$ ：存储器访问信号，输出。该信号有效表示当前进行存储器访问操作。

$\overline{\text{IORQ}}$ ：I/O 访问信号，输出。该信号有效表示当前进行 I/O 访问操作。也有些微处理器将 $\overline{\text{MREQ}}$ 和 $\overline{\text{IORQ}}$ 两根控制信号线合并为一根控制信号线 M/$\overline{\text{IO}}$ 来描述当前微处理器的状态是存储器访问还是 I/O 访问。M/$\overline{\text{IO}}$ 为低（高）表示存储器访问，与之相对应 M/$\overline{\text{IO}}$ 为高（低）表示 I/O 访问。

$\overline{\text{INT}}$ ：中断请求信号，输入。该信号有效表示外部有中断请求。当 $\overline{\text{INT}}$ 有效时，在适当的条件下微处理器会暂停当前程序的执行，转入指定的中断服务操作。

$\overline{\text{INTA}}$ ：中断响应信号，输出。该信号有效表示微处理器收到外部中断请求信号以后同意进行中断服务。微处理器发 $\overline{\text{INTA}}$ 信号给中断源，通知中断源响应其中断请求。通常，微处理器在 $\overline{\text{INTA}}$ 信号有效期间读取中断向量，用于识别中断源。

HOLD：总线保持请求信号，输入。该信号有效表示除了本微处理器以外还有外部其他总线控制部件需要占用总线而向微处理器提出申请。微处理器应在合适的时刻响应这一请求。

HLDA：总线保持响应信号，输出。该信号有效表示微处理器对总线请求信号的响应。微处理器输出 HLDA 有效，然后释放总线控制权，由外部总线控制部件接管对总线的控制。

$\overline{\text{RESET}}$ ：复位信号，输入。该信号有效，将强制微处理器处于初始状态。所谓初始状态是指微处理器内部各寄存器被设置为初态，所有标志位全部清除。当复位信号 $\overline{\text{RESET}}$ 撤销以

后，微处理器开始新的指令周期，其执行的第一条指令代码取决于程序计数器 PC 或指令指针 IP 复位以后的初值。

$\overline{\text{WAIT}}$：等待信号，输入。该信号有效表示微处理器操作与部件操作时间上不匹配，要求微处理器适当等待。

2.1.3 微处理器的基本时序

微处理器引脚上的信号的输出是与时间有关的。在不同的时段，各引脚上的信号的输出是不相同的，这些信号都受一个统一的时钟信号的控制，也就是说，微处理器是在时钟脉冲的统一控制下，一个节拍一个节拍地工作的。从时序角度考虑，微处理器的执行工作可分为 3 种类型的周期：时钟周期、总线周期和指令周期。

时钟周期（Clock Cycle）也称为 *T* 状态，是微处理器动作处理的最小时间单位。时钟周期值的大小是由系统时钟确定的。如一个微处理器的系统时钟为 5MHz，则时钟周期为 200ns。

总线周期（Bus Cycle）是指微处理器对存储器或 I/O 端口完成一次读、写操作所需要的时间。通常，一个总线周期由若干个时钟周期所组成。总线周期也称机器周期。

指令周期（Instruction Cycle）反映了执行一条指令所需要的时间。一个指令周期通常由若个总线周期组成，对于读取指令代码，就是一个存储器读总线周期。在微型计算机中，由于指令的复杂程度不同，执行指令所需要的时间也不同，不同的指令周期其长短不一样。一般组成指令周期的总线周期数往往不一样，简单指令只需要一个总线周期，复杂指令就需要较多的总线周期了。

由于微处理器指令众多，而指令周期是由若干总线周期所组成的，这些总线周期反映微处理器对外部存储器或 I/O 进行读、写操作，种类十分有限。因此，对微处理器时序的讨论，重点在于对总线周期的讨论。从总线周期可以反映出微处理器对外部是如何操作的，也可以反映出微处理器与外部的关系。

一个微处理器最基本的总线周期是读总线周期和写总线周期，通过这两个总线周期，微处理器可以完成绝大部分的操作（对存储器的读、写操作和对 I/O 设备的读、写操作）。除此之外，微处理器还有中断请求与响应周期，总线请求与响应周期。

1．读操作总线周期

如图 2-2 所示为读操作总线周期，该总线周期是由四个时钟周期组成的。相关的引脚有地址线、数据线和读信号线。

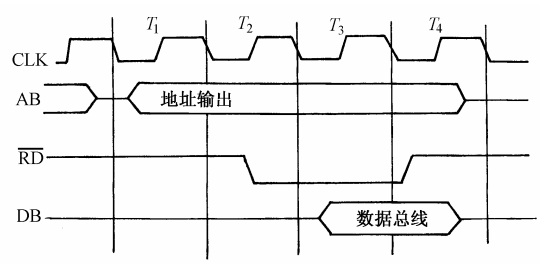


图 2-2　读总线周期

从第一个时钟周期起，微处理器输出有效地址并在地址总线上保持到本总线周期结束，该地址值用于选择某一确定的存储器单元或 I/O 端口。

从第二个时钟周期起，控制线$\overline{RD}$有效，$\overline{RD}$信号的作用是控制选中的存储器单元或I/O端口将有效数据输出到数据总线上。$\overline{RD}$信号有效一直维持到第四个时钟周期。

在$\overline{RD}$信号有效期间，数据总线上会出现有效数据。微处理器将在第三个时钟周期内，即$\overline{RD}$信号撤销之前将数据总线上的数据读入到微处理器内部。

当前总线周期是存储器读还是I/O读，取决于微处理器的输出控制信号$\overline{MREQ}$和$\overline{IORQ}$信号，若$\overline{MREQ}$信号有效，则为存储器读周期；若$\overline{IORQ}$信号有效，则为I/O读周期。

若微处理器所访问的存储器或外设工作速度较慢，不能满足上述的基本时序要求，也就是说在第三个时钟周期期间存储器或I/O还不能提供有效数据到数据总线，使得微处理器不能读到数据，这就需要产生$\overline{WAIT}$信号的电路，在T_3和T_4之间插入一个T_W状态，来解决微处理器与存储器或I/O之间的时间配合问题，如图2-3所示。

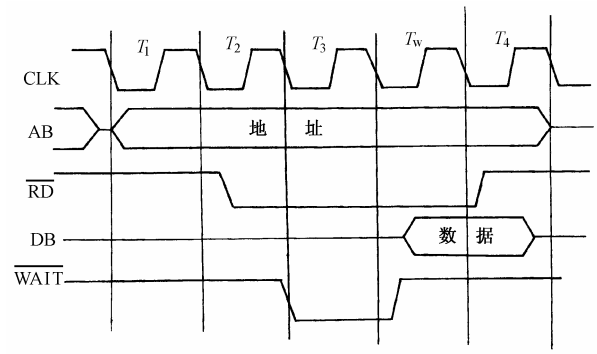


图2-3　具有等待状态的读总线周期

微处理器在T_3开始时刻采样$\overline{WAIT}$信号，如果有效，说明需要等待，因此微处理器在T_3周期结束以后不是进入T_4周期，而是插入一个T_W状态，即一个时钟周期，以后在每一个T_W状态开始时刻都采样$\overline{WAIT}$信号的状态，以决定是否还需要插入T_W状态。只有$\overline{WAIT}$信号撤销后，T_W状态才进入T_4周期。

2．写操作总线周期

同样，对于写总线周期，是一个微处理器将数据写入存储器或I/O端口的过程，如图2-4所示。

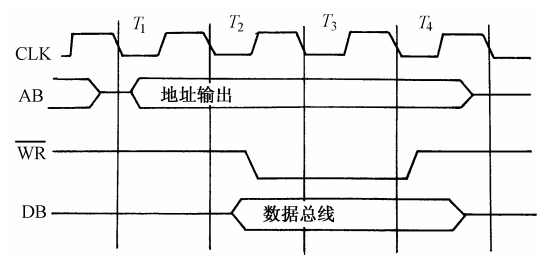


图2-4　写总线周期

首先，第一个时钟周期输出地址有效，用于选中某一个存储器单元或I/O端口；第二个总线周期微处理器输出有效数据，同时$\overline{WR}$写信号线输出有效，向存储器或I/O发写命

令；在完成写操作以后，$\overline{WR}$ 信号撤销，数据总线上的数据也跟着撤销，一个总线周期结束。

每一条指令的执行，都是由这些总线周期组成的。例如某一条指令的功能是将微处理器内部累加器中的值写入指定的存储器单元，执行这条指令可能就需要两个总线周期：读总线周期和写总线周期。

第一个读总线周期的功能是将存储器中的指令代码读入到微处理器内部的指令寄存器中。在这个总线周期中，地址总线上提供的是程序计数器 PC 的值，数据总线上的有效数据是指令代码。

第二个总线周期是写总线周期，其功能是将累加器中的数据通过数据总线写入到指定的存储器单元中。在这个总线周期中，地址总线上是微处理器提供的存储器单元地址，数据总线上是微处理器提供的累加器的值。

当前总线周期是存储器写还是 I/O 写，同样取决于微处理器的输出控制信号 $\overline{MREQ}$ 和 $\overline{IORQ}$ 信号，若 $\overline{MREQ}$ 有效，则为存储器写周期；若 $\overline{IORQ}$ 信号有效，则为 I/O 写周期。

与读操作总线周期一样，写操作总线周期也需要考虑微处理器与存储器及 I/O 的时间配合问题，以确定是否需要插入 T_W 状态。

3．中断请求与响应周期

当外部中断源通过 INT 引脚向微处理器发中断请求信号时，微处理器在允许中断的状态下执行完当前指令后，进入中断响应周期。在中断响应期间 $\overline{INTA}$ 信号有效，如图 2-5 所示。

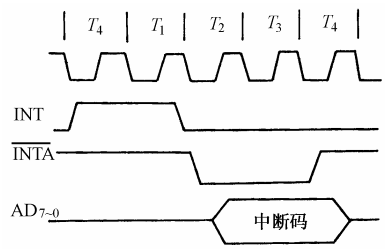


图 2-5　中断请求与响应周期时序

中断响应周期由 4 个 T 状态组成。从 T_1～T_4，INTA 为低电平有效，作为对中断请求设备的响应，该输出信号通知中断请求设备把中断向量送到数据总线的低 8 位。微处理器就是根据不同的中断向量来识别不同的中断源的。

4．总线请求与响应周期

当一个系统中具有多个总线控制主模块时，微处理器以外的其他总线控制主模块为了获得对总线的控制，需要向微处理器发出使用总线的请求；而微处理器在得到请求之后，如果同意让出总线，就需要向总线请求的主模块发应答信号。图 2-6 就是总线保持的请求与响应时序图。

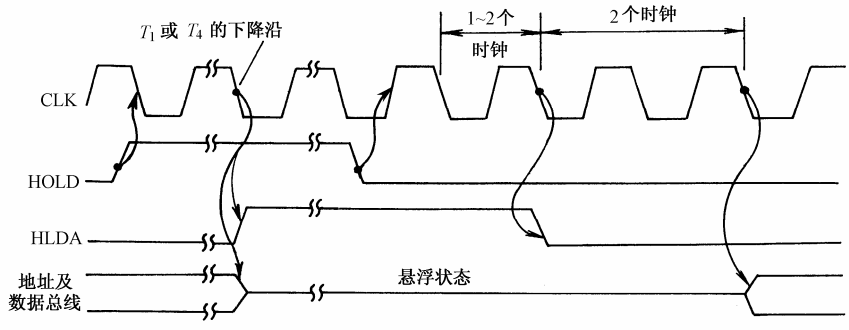


图 2-6　总线保持时序图

HOLD 是由其他总线控制主模块发给微处理器的。当某个总线控制主模块企图使用总线时，便发出高电平的 HOLD 信号，然后等待微处理器发总线保持回答信号 HLDA。

在每个时钟脉冲的上升沿处，微处理器会对 HOLD 引脚上的信号进行检测。如果检测到 HOLD 处于高电平状态，并且允许让出总线，那么在总线周期的 T_4 状态或者空闲状态 T_1 之后的下一个时钟周期内，微处理器会发出 HLDA 信号，从而微处理器便将总线让给提出总线保持请求的设备。此后，当总线请求模块将 HOLD 信号变低时，微处理器会收回总线控制权。

微处理器一旦让出总线控制权，便使数据总线、地址总线以及控制信号 $\overline{\text{RD}}$ 、 $\overline{\text{WR}}$ 等都处于浮空状态，这样，微处理器与数据总线、地址总线以及上述控制线暂时脱离关系。

实际上，总线保持请求可分为 3 个阶段：总线请求阶段、总线释放阶段和总线收回阶段。微处理器总是在 T_1 或 T_4 周期测试 HOLD 是否为高电平，一旦有效，且同意出让总线，就发 HLDA 信号，这是第一阶段。第二阶段，微处理器将数据总线、地址总线以及部分控制线置于浮空状态，出让总线给申请者。微处理器在出让总线以后，不断检测 HOLD 信号是否为低，以确定对方是否释放总线，一旦为低，则进入第三阶段，微处理器重新占用总线。

5．系统复位

微处理器的 RESET 引脚可以用来启动或再启动系统。当微处理器在 RESET 引脚上检测到一个脉冲的正沿时，它终结所有的操作，直到 RESET 信号变低。这时 CPU 内部各寄存器被初始化到复位状态，复位时序如图 2-7 所示。

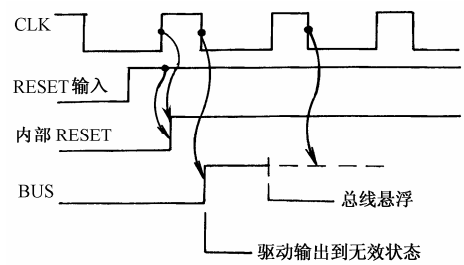


图 2-7　复位时序

在 RESET 为高电平复位期间，微处理器地址、数据、控制总线都处于浮空状态，直到 RESET 恢复为低电平，微处理器脱离复位状态，开始从处于初始状态的程序计数器或指令指针指向的存储单元中取指令执行。

2.2 16位微处理器

微型计算机的出现与发展，引起了世界范围内的计算机大普及浪潮。1971 年 Intel 4004 的 4 位微处理器组成的 MCS-4 是世界上第一台微型计算机，30 多年来，微型计算机获得惊人的飞跃式发展，从 4 位、8 位、16 位、32 位直至 64 位。微型计算机性能的提高，是由处理器所带动的。

微处理器性能的提高主要在于两方面：一是改进了微处理器芯片的半导体工艺技术，二是微处理器采用了更先进的技术。半导体技术进步已从 3μm 的线宽减少到 0.25μm 的线宽。线宽的减少使芯片上能集成更多的器件，也使处理器工作时钟频率大幅度提高。

Intel 微处理器的发展已经历了 8086、80286、80386、80486、Pentium、Pentium MMX、Pentium Pro、Pentium Ⅱ、Pentium Ⅲ、Pentium 4 等产品。其中 8086 和 80286 都是 16 位微处理器的典型产品，本节以 80286 为典型产品介绍微处理器。

80286 是一种高性能的 16 位微处理器，它有 16 位数据线、24 位地址线，支持多种数据类型，有很强的数值运算能力，指令丰富，性能优良。在 80286 内部，集成有存储器管理和存储器保护机构，因此 80286 能以两种不同的方式工作：实地址方式和保护虚地址方式。实地址方式的工作与 8086 基本相同。在保护方式下，通过应用存储管理方法，可使系统获得 1 000MB 的虚拟存储空间，并可以将此虚拟空间映像到 16MB 的物理存储器上。80286 的保护功能可以对存储器的段边界、属性及访问权等进行自动检查，通过四级保护环结构支持任务与任务之间以及用户与操作系统之间的保护，也支持任务中程序和数据的保密性，从而确保在系统中建立高可靠性的系统软件。80286 还具有高效的任务转换功能，可以适应多用户、多任务的需要。

80286 对 8086 等微处理器是向上兼容的。在实地址方式下，80286 的目标代码与 8086 软件兼容。在保护虚地址方式下，8086 的软件源代码可以使用 80286 存储器管理和保护机构支持的虚地址。80286 也可配置协处理器，以增加数值计算的能力与速度，它所支持的数值协处理器 80287 也对数值协处理器 8087 向上兼容。

2.2.1 80286CPU 的基本结构

80286 微处理器有四个独立的处理部件，它们分别是执行部件 EU、总线部件 BU、指令部件 IU 和地址部件 AU。各部件之间的联系如图 2-8 所示。整个 80286 采用流水线作业方式，使各部件能同时并行工作。

1．总线部件 BU

总线部件由地址锁存器和驱动器、协处理器扩展接口、总线控制器、数据收发器、预取器和六字节预取队列组成。

总线部件是微处理器与系统之间的一个高速接口，负责管理、控制总线操作，管理控制微处理器与存储器、外部设备之间的联系。在存、取代码和数据等期间内有效地满足微处理器对外部总线的传送要求，以最高的速率传送数据，即在两个处理器时钟周期内传送一个字。

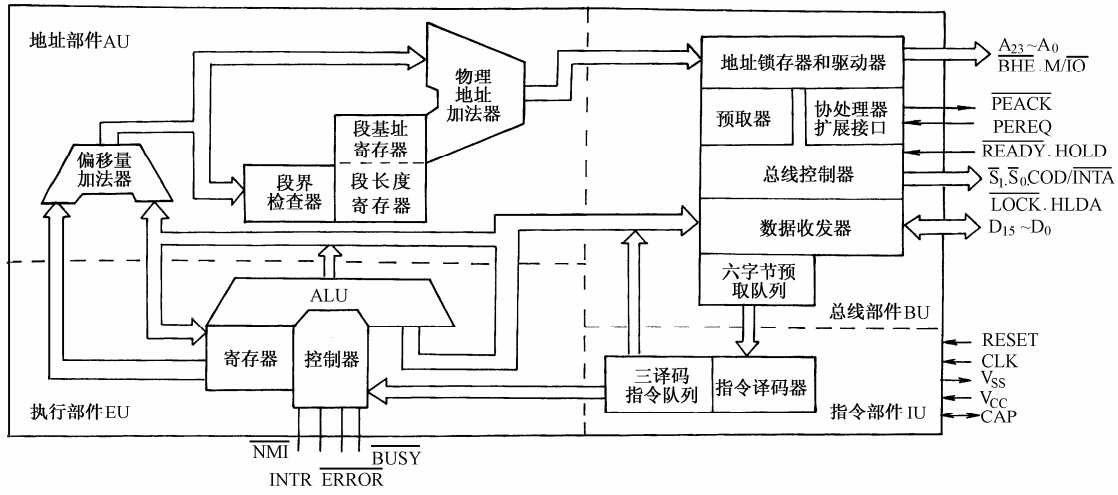


图 2-8　80286 微处理器结构图

80286 的总线部件还负责预取指令的操作。所谓预取指令，就是在指令执行之前把它从存储器中取出，并送入指令队列中，等待进一步的译码操作。80286 的指令预取队列最多可保存六字节的指令机器码。预取器总是力图使该队列装满代码的有效字节。每当队列有部件空闲或发生一次控制转移后，预取器便请求预取，以保持队列总被装满。

2．指令部件 IU

指令部件中设有指令译码器和译码指令队列。这个部件用来对指令译码，并做好执行部件执行的准备工作。当指令部件把来自六字节预取队列的指令译码后，就把它们存放到已译好码的指令队列中，准备执行。译好码的指令包含执行部件执行时需要的所有指令域。在 80286 微处理器中引入指令部件将进一步改善流水线操作，改变了如同 8086 等微处理器那样需要由指令执行部件译码的局面。指令部件连续译码，使得已译码的队列内总有几条已完成译码操作的指令字节等待执行。与此同时执行部件执行的总是事先译好的指令，使得译码部件和执行部件并行操作，从而大大提高了 80286 微处理器的工作速度。指令部件以每个时钟周期 1 个字节的速度接收数据。

3．执行部件 EU

执行部件由寄存器、控制部件、算术及逻辑运算单元 ALU 和微程序只读存储器构成。它负责执行指令。微处理器完成的算术运算、逻辑运算以及其他数据加工等操作均在这里实现。为此，它使用自己的资源，同时也与执行指令所必须的其他逻辑部件交换控制信息和时序信息。微程序只读存储器 ROM 规定了指令执行的内部微指令序列，指令在内部微指令序列的控制下执行。当一条指令的微指令序列快要完成时，ROM 就发出信号，让执行部件从指令队列里再取出下一条微指令序列的入口地址，即 ROM 地址，以控制执行下一条指令，这是微程序控制的典型做法。这项技术的采用，使得执行部件总是处于忙碌状态。

4．地址部件 AU

地址部件由偏移量加法器、段界检查器、段基地址寄存器、段长度寄存器和物理地址加法器等部件构成。它实施存储器管理及保护功能，计算出操作数的物理地址，同时检查保护权。在保护方式下，地址部件提供完全的存储管理、保护和虚拟存储等支持。为此，地址部件在存储器中建立操作系统控制表，以描述机器的全部存储器，然后由硬件如实地执行表中信息。

地址部件在检查访问权的同时完成地址转换。在这个部件内有一个高速缓冲寄存器，该寄存器保存着段的基地址、段长界限和当前正在执行的任务所用的全部虚拟存储段的访问权。为使从存储器中读出的信息减至最小，高速缓冲寄存器允许地址部件在一个时钟周期内完成它的功能。

80286 微处理器的 4 个部件构成一个有机的整体。总线部件把微处理器连接到外部系统总线，并且控制地址、数据及控制信号从微处理器输出或向微处理器输入。预取部件负责从指定的存储区域中取出指令，送入预取指令队列。该队列是预取器和指令译码器之间的一个缓冲。指令译码器将指令从队列中取出、译码后送入已译码指令队列。执行部件根据已译码的指令，按照所需步骤执行这条指令。执行指令过程中的有关寻址操作由地址单元完成。80286 微处理器内部的这种并行操作如图 2-9 所示。

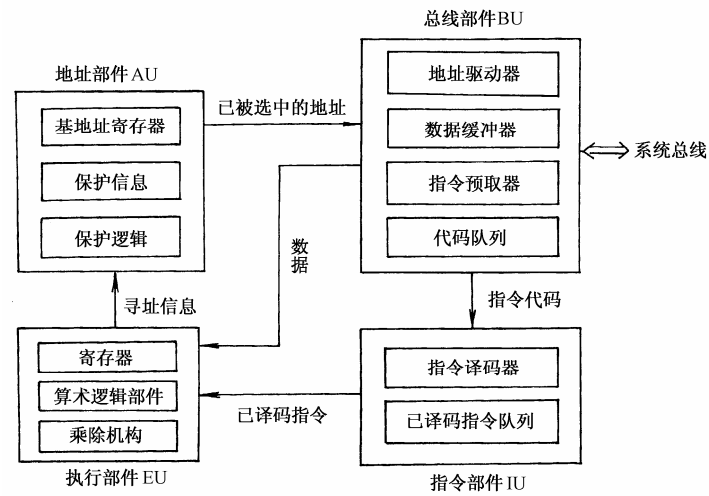


图 2-9　80286 微处理器内部的并行操作

2.2.2　80286 寄存器结构

在微处理器内部，往往包含有很多专用或通用寄存器，它们要么用于暂存数据、保存运算结果，要么用于保存控制信息或微处理器的有关状态信息。80286CPU 总共有 19 个内部寄存器，按其功能可分为通用寄存器、段寄存器、状态和控制寄存器以及系统地址寄存器等几类。图 2-10 给出了 80286 寄存器的构成。

1．通用寄存器

80286CPU 内部有 8 个 16 位通用寄存器，根据寄存器的功能不同，可归并为数据寄存器

和指针及变址寄存器。

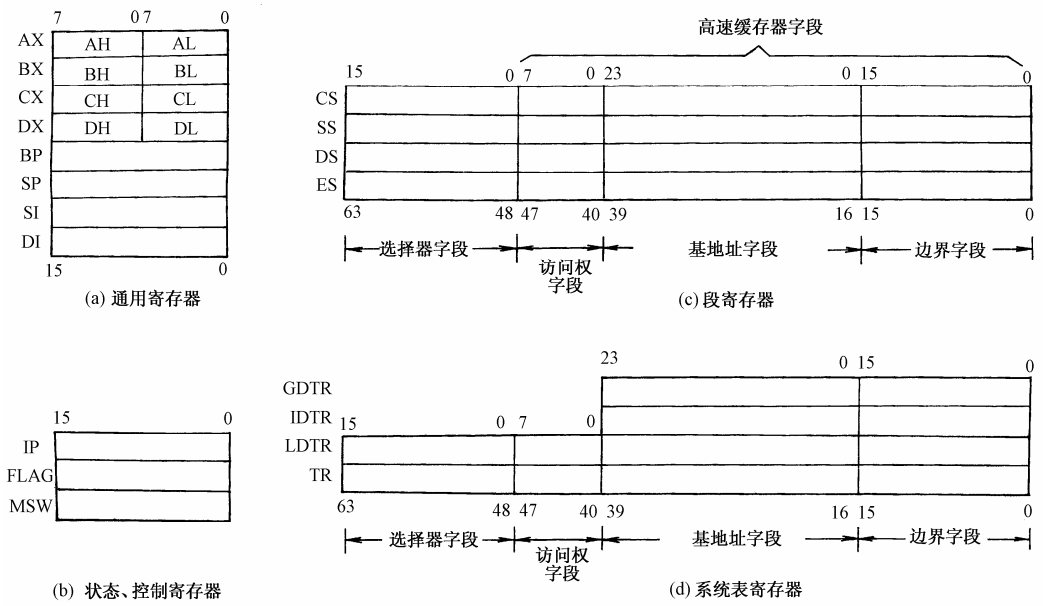


图 2-10　80286 寄存器

（1）数据寄存器组。数据寄存器组主要用于算术运算和逻辑运算，它包括 16 位寄存器 AX、BX、CX 和 DX。这些寄存器既可作为 16 位寄存器使用，也可以把每一个寄存器分成两个 8 位寄存器分别使用。把数据寄存器作为 8 位寄存器使用时，每个 8 位寄存器均有各自的名称，高 8 位部分称为 AH、BH、CH 和 DH 寄存器，低 8 位部分称为 AL、BL、CL 和 DL 寄存器。

各数据寄存器的主要用途如下：

累加寄存器 AX：用于 80286 中的数据字或数据字节的加减法、乘除法以及输入/输出等操作运算，某些数据串操作也是使用此累加器进行的。

基地址寄存器 BX：主要作用是为存储器进行寻址操作提供一个基准地址。

计数寄存器 CX：该寄存器在循环操作中做重复计数器用；在重复数据串操作中，做字符个数计数器用；在移位和循环移位中，用来控制移位次数。

数据寄存器 DX：主要用于字数据的乘法和除法，有时还用来提供输入/输出端口地址。

（2）指针与变址寄存器组。80286CPU 的指针与变址寄存器由 16 位寄存器 SP、BP、SI 和 DI 组成，其作用有两个：一是用于存放操作数，二是用于形成指令访问的存储器地址。

在形成存储器地址时，指针寄存器 SP、BP 和变址寄存器 SI、DI 在使用上是有差别的。一般情况下，通过指针寄存器寻址，访问的是堆栈内的栈区，SP 和 BP 中保存着进入当前堆栈的偏移量。而通过变址寄存器寻址，访问的是数据存储器。具体存储器地址的形成，都需要与段寄存器的内容一起确定。

2．段寄存器

在微处理器内部设置段寄存器，这是 8086、8088、80286 等系列微处理器结构的一个特点。

80286 系统将其可寻址的存储器划分为若干个不同的区域，这些区域称为段。不同的段中存放的信息可以不同。存放程序代码的区域称为代码段；存放数据信息的段称为数据段或附加段；用做开辟堆栈区域的段称为堆栈段。

80286 利用四个 16 位的段寄存器支持对不同段的访问。通过代码段寄存器 CS 访问代码段；通过数据段寄存器 DS 访问数据段；通过附加段寄存器 ES 访问附加段；通过堆栈段寄存器 SS 访问堆栈。程序可对段寄存器进行存取操作。

此外，80286 的各个段寄存器都带有 48 位的段高速缓存器，它们用来记录各段的属性参数，这些高速缓存器是 80286 进行存储管理时内部使用的存储器，其值不能由外部直接送入。

3．状态和控制寄存器

状态和控制寄存器由指令指针寄存器 IP、标志寄存器 FLAG 和机器状态字寄存器 MSW 组成。

（1）指令指针寄存器 IP。指令指针寄存器 IP 即通常 CPU 中的程序计数器 PC，用以提供指令的存储地址。此寄存器为 16 位，在 80286 存储器分段管理的情况下，指令的实际存储地址将由指令指针寄存器内容和代码段寄存器内容共同构成。与一般 CPU 的程序计数器 PC 一样，IP 也具有自动修改或强行变更内容的功能。

（2）标志寄存器 FLAG。标志寄存器的主要功能是用来记录 CPU 运行过程中的有关状态信息的。它一般是指算术运算指令、逻辑运行指令或其他指令操作后形成的有关结果特征。80286 的标志寄存器共有 16 位，但实际使用的标志只有 11 个，各标志在标志寄存器 FLAG 中的对应位置如图 2-11 所示。

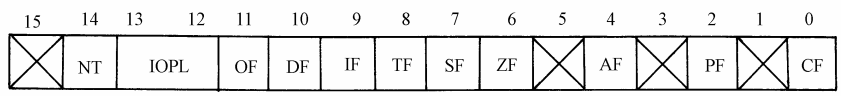


图 2-11　80286 标志寄存器

在这 11 个标志中可分为两组，一组是状态标志，另一组用于控制完成某些操作。

状态标志组，包括进位标志 CF、奇偶标志 PF、辅助进位标志 AF、零标志 ZF、符号标志 SF 和溢出标志 OF。它们的状态受算术逻辑运算指令或其他指令的影响，也为这些指令提供相应的信息。

控制标志位，包括自陷标志 TF、允许中断标志 IF、方向标志 DF、输入/输出特权级标志 IOPL 和嵌套任务标志 NT。当自陷标志 TF 置“1”时，80286 可处于单步操作方式，并且允许调试程序。当把允许中断标志 IF 置“1”时，便允许 CPU 去识别、响应外部的可屏蔽中断请求；当把 IF 置“0”时，则禁止外部中断。方向标志 DF 用来控制数据串的操作方向。

当 80286 工作于保护方式时，才可使用输入/输出特权级标志 IOPL 和嵌套任务标志 NT。输入/输出特权级标志有两位，用于指定输入/输出操作处于系统特权级的某一级。嵌套任务标志 NT 用于说明系统任务嵌套的情况。若 NT 置“1”，则说明该任务嵌套在另一个任务中执行，执行完成该任务后，要再返回到原来的任务中去，否则把 NT 置“0”。该位的置“0”与置“1”都是通过其他任务的控制转移来实现的。

（3）机器状态字寄存器 MSW。80286 机器状态字寄存器有 16 位，如图 2-12 所示。该寄存器仅用了低 4 位，其定义的含义如下：

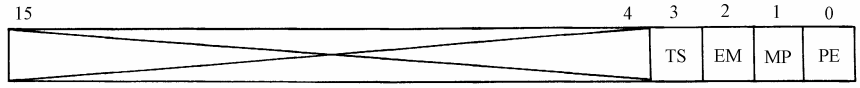


图 2-12　80286 机器状态寄存器

允许保护位 PE：该位设置了 80286CPU 的工作方式。当把 PE 置“1”时，80286 进入保护方式；若把 PE 复位，则 80286 在实地址方式下工作。

监控处理器位 MP：用 MP 位与 TS 位一起确定 WAIT 操作码是否将要生成一个协处理器不可用的异常 7，当然此时的 TS 应为 1。

协处理器仿真位 EM：若 EM 置“1”就会使所有的协处理器操作码生成协处理器不可用的异常 7；EM 复位，则所有协处理器操作码都在一个实际的 80287 协处理器上执行。

任务转换位 TS：每当完成一次任务转换就把 TS 置“1”，如果在 TS 置“1”时，也把 MP 置“1”，则处理器的操作码将会产生一个协处理器不可用的自陷。此时，自陷任务处理程序通常保存前一任务与 80287 相关的信息，装入现行任务的 80287 状态，并在返回到这个引起故障的协处理器操作码之前将 TS 清“0”。

4．系统地址寄存器

80286 有四个系统地址寄存器，它保存着操作系统需要的保护信息和地址转换表信息。这四个寄存器的名称和作用如下：

全局描述符表寄存器 GDTR（Global Descriptor Table Register）是 40 位的寄存器，由 24 位基址字段和 16 位边界字段组成，用于保存全局描述符表 GDT 的基地址和界限。

中断描述符表寄存器 IDTR（Interrupt Descriptor Table Register）也是 40 位，用于保存中断描述符表 IDT 的 24 位基地址和 16 位界限。

局部描述符表寄存器 LDTR（Local Descriptor Table Register）是 16 位寄存器，它保存着 16 位的 LDT 段的选择符。LDTR 像段寄存器那样带有高速缓冲存储器，该缓冲存储器共有 40 位，由 24 位基址字段和 16 位边界字段构成。

任务状态寄存器 TR（Task State Table Register）是 16 位寄存器，其内保存任务状态段 TSS 的 16 位选择符。TR 也带有 40 位高速缓冲存储器。

2.2.3　80286CPU 的引脚功能

80286 微处理器封装在 68 条引脚的正方形管壳中，采用四面引脚的方式，图 2-13 是 80286 芯片引脚的示意图。

1．数据线

D_{15}～D_0：数据总线。数据总线是双向传送的三态信号线。在存储器读周期、I/O 读周期、中断响应周期时是输入数据，在存储器写周期、I/O 写周期时是输出数据。在数据总线上，可能传送的是指令码、操作数、操作数地址、中断类型码等。当 80286CPU 释放总线时，数据总线处于高阻状态。

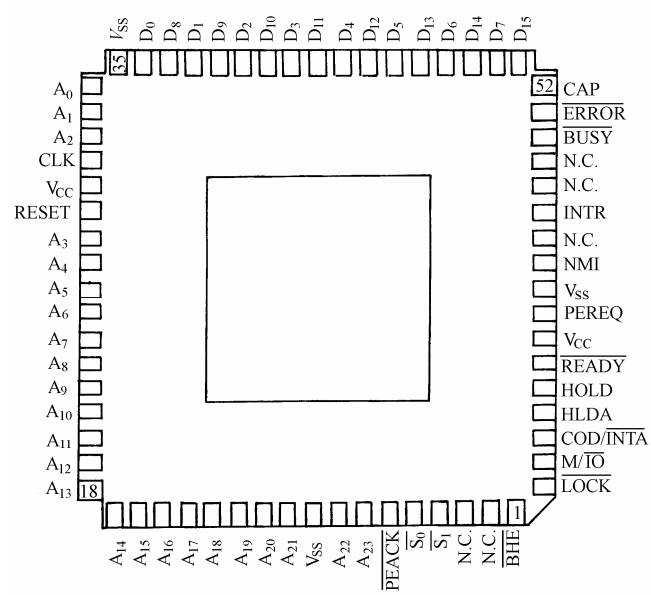


图 2-13　80286 引脚配置图

2．地址线

A_{23}～A_0 **：地址总线。地址总线是输出物理存储器地址或 I/O 端口地址的单向三态信号线，寻址范围为 16MB。当 80286CPU 释放总线时，地址总线处于高阻状态。当数据信息在数据总线低 8 位 D_7～D_0 上传输时，A_0 为低电平。**

$\overline{BHE}$ **：总线高位允许线，是一条低电平有效的三态输出信号线。该信号有效时，表示数据在数据总线的高字节有效 D_{15}～D_8 线上传输。BHE 信号线与地址线 A_0 一起确定当前数据总线是低字节有效还是高字节有效，或是 16 位数据线有效。**

3．控制线

M/$\overline{IO}$ **：存储器或输入/输出选择线。这是三态输出信号线，用于区分当前操作是存储器访问还是 I/O 访问。若此引脚为高电平，则为存储器访问；若此信号为低电平，则为 I/O 访问。**

COD/$\overline{INTA}$ **：代码或中断响应确认线。这也是一个三态输出信号线，用于对指令周期和读数周期加以区别，也对中断响应周期和输入/输出周期加以区别。对于** M/$\overline{IO}$ **的高电平状态，** COD/$\overline{INTA}$ **作为取指令代码的指示信号，当** COD/$\overline{INTA}$ **也为高电平时，表示取指令代码；对于** M/$\overline{IO}$ **的低电平状态，** COD/$\overline{INTA}$ **作为中断响应，当** COD/$\overline{INTA}$ **为低电平时，表示中断确认。**

$\overline{S_1}$、$\overline{S_0}$ **：总线周期状态线。这是两个三态输出的信号线，它们与信号线** COD/$\overline{INTA}$ **和** M/$\overline{IO}$ **一起，表明 80286 总线周期的状态，其关系如表 2-1 所示。总线周期状态信号提供给系统中的总线控制器，以产生存储器或输入/输出读、写所需要的命令控制信号。**

表 2-1　80286 总线周期状态定义

COD/$\overline{INTA}$	M/$\overline{IO}$	$\overline{S_1}$	$\overline{S_0}$	总线周期状态
0	0	0	0	中断响应
0	0	0	1	保留
0	0	1	0	保留
0	0	1	1	空闲状态
0	1	0	0	停机或暂停状态
0	1	0	1	读存储器数据
0	1	1	0	写存储器数据
0	1	1	1	空闲状态
1	0	0	0	保留
1	0	0	1	读 I/O 端口
1	0	1	0	写 I/O 端口
1	0	1	1	空闲状态
1	1	0	0	保留
1	1	0	1	读存储器中的指令代码
1	1	1	0	保留
1	1	1	1	空闲状态

$\overline{LOCK}$ ：总线封锁信号线。这是一个低电平有效的三态输出信号线，它表示在当前的总线周期后，另外的系统总线控制设备将不能得到系统总线的控制权。$\overline{LOCK}$ 信号可以直接由前缀指令“ lock ”激活，也可以由 80286 的硬件在存储器 XCHG 指令、中断响应或访问描述符表期间自动产生。

$\overline{READY}$ ：总线准备就绪信号线。这是一个低电平有效的输入信号线，它控制着总线周期的结束、请求建立和保持与系统时钟有关的时间，以满足正确操作的需要。若在一个总线周期的最后 $\overline{READY}$ 为有效电平，则结束总线周期；若 $\overline{READY}$ 为高电平，将再一次重复总线周期。

HOLD：总线保持请求。这是一个高电平有效的总线请求输入信号，用于其他总线控制设备向 80286 申请总线控制权。

HLDA：总线保持响应。这是一个高电平有效的总线响应输出信号，用于 80286 在收到总线请求信号以后发的响应信号。HOLD 和 HLDA 这一对信号用于控制 80286 局部总线的所有权。局部总线上的其他总线控制设备需要使用总线时，通过 HOLD 输入总线请求信号。当 80286CPU 允许转让总线控制权时，它将把所有的三态总线信号置于高阻状态，并送出总线响应信号 HLDA，通知申请总线的控制设备可以使用总线。此局部总线一直保持给提出请求的主设备，直到该主设备不用总线并撤销 HOLD 请求信号为止。随着 HOLD 请求信号的撤销，80286CPU 将撤销 HLDA 信号，并收回对局部总线的控制权，各三态信号脱离高阻状态，恢复到原来的状态，从而终止了总线保持响应状态。

INTR：中断请求信号线。这是一个高电平有效的输入信号。在 80286 系统中，一般由外部的中断控制器经 INTR 信号线向 CPU 发中断请求信号。在 80286 的内部如果没有对中断请求进行屏蔽，则执行中断响应周期，以便从外部读取中断类型码。INTR 是在每个总线周期的开始被采样，并必须在当前指令结束之前保持高电平有效至少两个系统时钟周期，以便在下一条指令之前实现中断。

NMI：非屏蔽中断请求线。这是由外部送来的高电平有效的非屏蔽中断请求信号，并且不受 CPU 内部中断允许标志位的控制。该信号可以异步于系统时钟，并且经内部同步后由系统时钟边缘触发。为了正确地识别这种中断请求，要求其输入必须是先有至少 4 个系统时钟

周期的低电平，而后保持至少 4 个系统时钟周期的高电平。

PEREQ：协处理器操作请求信号线。这是一个高电平有效的输入信号线，当 80286CPU 访问存储器的 ESC 指令时，数值协处理器 80287 送来的 PEREQ 信号被 80286 接收，以执行协处理器的一个数据操作数传输。

$\overline{\text{PEACK}}$ ：操作数响应线。这是一个低电平有效的输出信号线，当被请求的操作数正被传输时，$\overline{\text{PEACK}}$ 输出信号给协处理器，作为确认信号。

$\overline{\text{BUSY}}$ ：协处理器忙碌信号。这是一个低电平有效的输入信号，它把数值运算协处理器 80287 的工作状态告知 80287。从 $\overline{\text{BUSY}}$ 信号变为低电平，到 $\overline{\text{BUSY}}$ 信号再次变为高电平期间，80286 将不执行 ESC 指令和 WAIT 指令。

$\overline{\text{ERROR}}$ ：协处理器出错信号。这是数据运算协处理器 80287 要求 80286 进行异常处理时使用的输入信号。80286 收到此信号后，将产生类型 16 中断，进行异常处理。

RESET：系统复位端。这是高电平有效的输入信号，用于对 80286 内部进行初始化操作。

4．时钟及电源

CLK：系统时钟。这是外部送到 80286 的基本时钟信号。根据 80286 微处理器的要求，该信号有 12MHz、16MHz、20MHz 和 24MHz 等多种。通过 80286 内部的分频电路对 CLK 进行二分频处理，形成处理器时钟 PCLK，供内部各工作电路使用。

V_{CC}：+5V 电源。

V_{SS}：信号地。

CAP：电源基片滤波器电容器输入连接端。

2.2.4 80286CPU 总线操作与状态

80286 与其他微处理器一样，采用了总线周期和 T 状态构成的总线时序系统，但其处理器时钟 PCLK 与系统时钟 CLK 周期宽度不同，微处理器将系统时钟除以 2 来产生处理器时钟，并由它来确定状态。两种不同时钟用以满足系统中不同部件的需要。

1．总线状态

80286 系统总线有 3 种基本状态：空闲状态 T_i、传送状态 T_s 和执行状态 T_c。此外还有一种局部总线状态 T_h，也称保持状态，它表示 80286 在响应总线请求 HOLD 之后，已把局部总线的控制权转让给其他的总线设备。图 2-14 给出了这 4 种总线状态的相互关系。

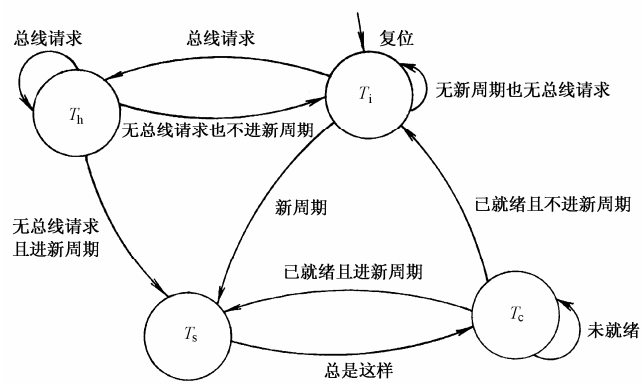


图 2-14　80286 总线状态

空闲状态 T_i 表示总线空闲，没有数据传送发生。传送状态 T_s 是总线周期的有效状态，在 T_s 期间，指令编码、地址和数据在 80286 输出引脚上是有效的，传送状态之后，总是进入执行命令状态 T_c。在 T_c 期间，存储器和 I/O 设备响应总线操作，将数据送入微处理器或接收写入的数据。为确保存储器或 I/O 设备有足够的时间做出响应，T_c 状态通常是可以重复的，由 READY 就绪信号来决定是否需要重复 T_c 状态。

在保持状态 T_h 期间，80286 将使所有的地址、数据和状态输出引脚浮空，以便使另外的总线主设备能使用局部总线。80286 的 HOLD 输入信号，用来使 80286 进入 T_h 状态。80286 的 HLDA 输出信号，表示微处理器已进入了 T_h 状态。

2．总线周期

80286 的每一个基本总线周期包含着两个处理器时钟周期，或称做两个总线状态。图 2-15 所示为基本总线周期的信号状况。

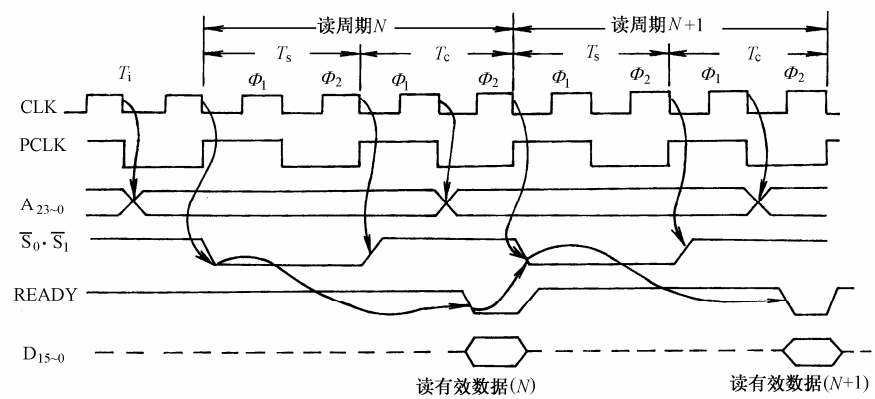


图 2-15　80286 基本总线周期

总线周期第一个状态是传送状态 T_s，它由 S_1、S_2 通知；第二个状态是命令执行状态 T_c。由图可见，80286 采用了流水线控制方式，其地址输出的定时也是流水线的。在任何一个 T_c 的节拍 2（Φ_2）期间，可发出下一个总线操作的地址，而在 T_c 节拍 1（Φ_1）期间现行地址有效，即下一个总线操作的第一个时钟周期与现行总线操作的最后一个时钟周期是重叠的。因此，下一个总线操作的地址译码和路径选择逻辑可以在下一总线操作之前进行操作。

80286 将各基本操作落实到总线周期，它所支持的总线操作有：存储器读、存储器写、I/O 读、I/O 写、中断响应和暂停/停机等 6 种，且每两个处理器时钟周期便可传送一个字。

3．80286CPU 子系统

80286 系统采用多用途的总线结构，具有一整套的支持组件，使得系统在大范围内有灵活的结构。

总的说来，80286 系统由微处理器、RAM、ROM、中断控制器、DMA 控制电路以及 I/O 电路等逻辑部件组成。图 2-16 表示了 80286CPU 构成的基本系统结构，它包括 80286 微处理器、一个 82284 时钟发生器、一个 82288 总线控制器以及两个 8259A 中断控制器等。另外，还有地址锁存器电路、数据接收/发送器电路和 I/O 译码电路。这些电路组成了系统的控制核心。

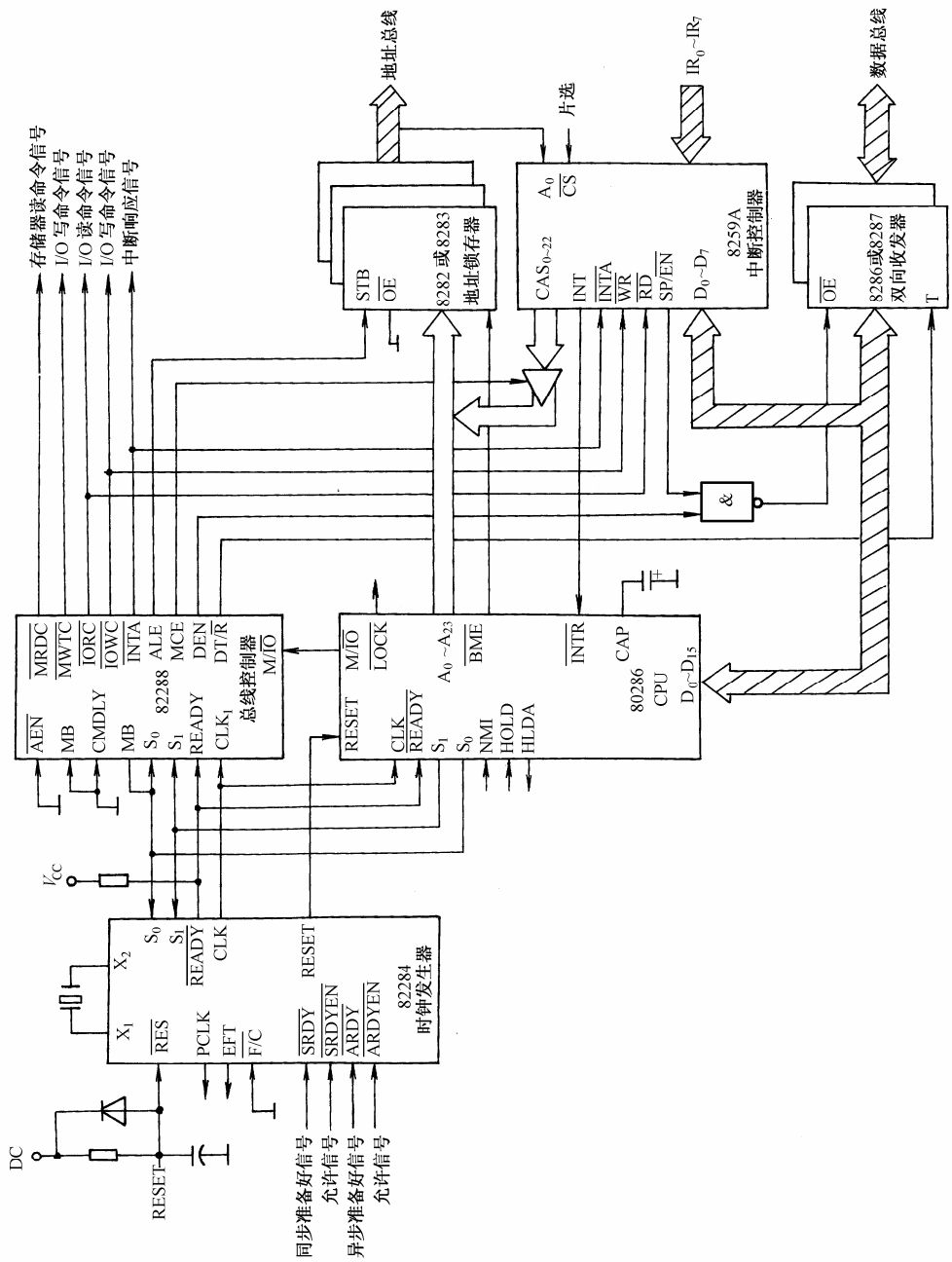


图2-16 80286的基本系统结构

8282 是 8 位锁存器，系统使用 8282 芯片对 24 位地址线以及 $\overline{\text{BHE}}$ 信号进行锁存，为地址总线提供稳定的地址值。8282 的地址锁存信号由总线控制器 82288 提供。

8286 是 8 位双向数据缓冲器，系统使用 8286 对 16 位的数据线进行数据驱动。8286 的方向控制由总线控制器 82288 的 DT/$\overline{\text{R}}$ 信号控制；8286 的输出允许控制分别由 82288 和中断控制器 8259A 控制，表示在正常的总线周期以及与中断相关的操作中 8286 处于数据允许输出状态，否则处于高阻状态。

时钟发生器 82284 是 Intel 公司专门为 80286 系统配套设计的单片时钟发生器，它能为系统提供处理器时钟 PCLK、系统时钟 CLK、准备就绪信号 $\overline{\text{READY}}$ 以及系统复位信号 RESET。其中，通过 X_1 和 X_2 之间连接的振荡晶体，使 82284 的内部晶体振荡电路工作，由 CLK 引脚输出时钟信号，同时 82284 还对 CLK 引脚输出的系统时钟进行二分频产生 PCLK 信号，这个信号与 80286CPU 的时钟是同步的。另外，82284 还提供与系统时钟同步的 $\overline{\text{READY}}$ 信号和 RESET 信号。就绪信号 $\overline{\text{READY}}$ 用以控制是否延长 80286 的总线周期，在 80286 总线周期的执行状态 T_c 的最后，如果 $\overline{\text{READY}}$ 为高电平，则 T_c 的后面再插入一个 T_c，以延长总线周期；如果 $\overline{\text{READY}}$ 为低电平，则总线周期结束。复位信号 RESET 用于向 80286 输入 16 个 CLK 以上宽度的脉冲信号，对 80286 内部状态进行初始化。80286 初始化复位以后，各寄存器的数值如表 2-2 所列。因此，复位以后的 80286 一开始是工作在实地址模式下，从存储器的物理地址 0FFFFF0H 中取第一条指令加以执行的。一般在这个单元定义一条 JMP 指令，把控制转换到执行对系统进行初始设置的程序。

表 2-2　80286 复位后的寄存器状态

寄存器名称	初　始　值
FLAG	0002H
MSW	0FFF0H
IP	0FFF0H
CS	0F000H
DS	0000H
SS	0000H
ES	0000H
CS 高速缓冲器	基地址＝0FF0000H，段长界限＝0FFFFH
DS 高速缓冲器	基地址＝000000H，段长界限＝0FFFFH
SS 高速缓冲器	基地址＝000000H，段长界限＝0FFFFH
ES 高速缓冲器	基地址＝000000H，段长界限＝0FFFFH
IDTR	基地址＝000000H，段长界限＝03FFH

82288 是专门为 80286 系统设计的总线控制器，它由状态信号译码电路、控制信号输入电路、命令输出电路和控制信号输出电路等部件组成，能为系统提供具有灵活时序选择的命令和控制信号。在 80286 系统中，总线控制器 82288 是一个核心组件，将根据 80286 的执行指令提供信号输出。82288 接收来自 CPU 的状态信号 $\overline{S_1}$、$\overline{S_0}$ 以及其他信号，确定 CPU 当前执行何种操作，以发出相应的命令输出和控制信号输出。其中，命令输出信号有：中断响应信号 $\overline{\text{INTA}}$、I/O 读信号 $\overline{\text{IORC}}$、I/O 写信号 $\overline{\text{IOWC}}$、存储器读信号 $\overline{\text{MRDC}}$ 和存储器写信号 $\overline{\text{MWTC}}$，它们都是低电平有效的信号。82288 的控制输出信号有：地址锁存允许 ALE、数据允许 DEN、数据发送或接收信号 DT/$\overline{\text{R}}$ 和主设备级连允许信号 MCE 等。图 2-17 为连续执行

存储器读和存储器写周期的波形图。

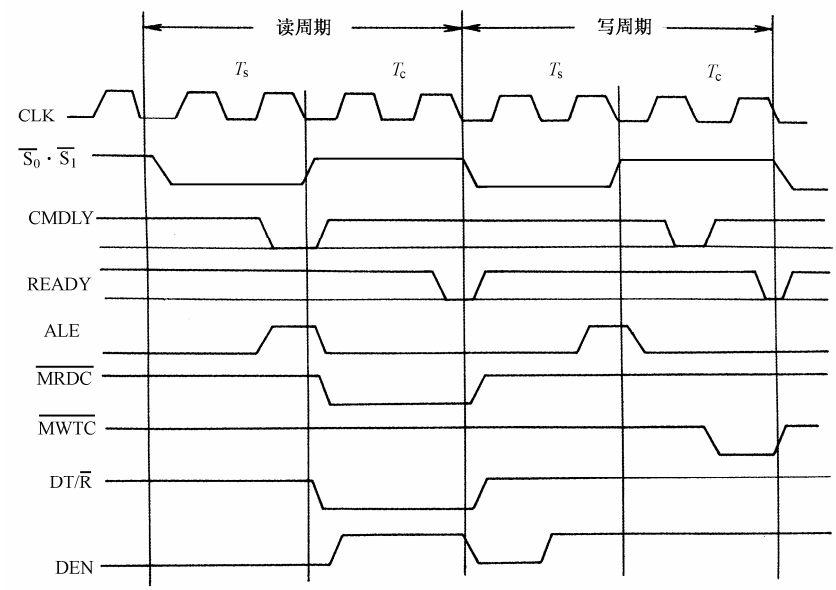


图 2-17　82288 输出信号波形

8259A 中断控制器用于接受外部中断请求，通过中断优先权的裁决，向 80286CPU 发中断请求，同时接收中断响应信号，并在中断响应周期中通过数据总线向 CPU 提供中断类型号，用于中断源的识别工作。

此外，系统还可以另加数值协处理器 80287。协处理器的加入使外部硬件在微处理器执行其他指令的同时完成专用的功能和数据传送操作。

2.2.5　保护方式与多任务

80286 具有实地址方式和保护虚地址方式两种工作方式。实地址方式相当于一个高性能的 8086CPU，有 16 位数据线，可访问 1MB 的内存地址空间，地址码将通过 80286 地址总线的低 20 位进行传送。

在保护虚地址方式中，80286 拥有与 8086 不同的功能。

1．虚拟地址管理

80286 的能力只有在保护虚地址管理方式下才能充分地发挥出来。保护方式是集实地址方式的能力、存储管理、对虚拟存储器的支持及对地址空间的保护为一体而建立起来的一种特殊工作方式，以便使 80286 支持多用户、多任务系统。

在保护虚地址方式下，80286 具有 16MB 的存储器寻址能力。通过微处理器内的保护虚地址结构，80286 对每个任务提供了最大为 1 024MB 的虚拟存储器空间。虚地址和实地址之间的转换由内部存储管理部件自动完成。

80286 在保护虚地址方式下，仍采用分段管理存储器的方法，使用逻辑地址表示存储空间中的特定位置，最终的物理地址由基地址和偏移量组成。与实地址方式不同的是，基地址不是直接由段寄存器提供，而是由段寄存器指定描述符表中的某个描述符，在描述符中有 24

位作为基地址。

80286 这种存储器管理方式，对存储器管理机构和保护机构提供了回旋余地，为 80286 多任务的实现提供了方便。

2．特权保护

所谓特权就是指在存储器访问中所拥有的优先权。在多任务下，为不同的用户、不同的任务、不同的过程设置优先权，可以有效地防止系统的混乱。

80286 设置了 4 级特权，编号为 0～3。0 级是操作系统核心；1 级是 I/O 驱动程序；2 级是操作系统扩展，如数据库系统等；3 级是用户程序。其中，0 级最高，3 级最低。这 4 级特权保护是在微处理器的硬件中实现的。

特权保护包括数据段与堆栈段访问的授权保护和代码段的特权保护。不同的任务，在对应的描述符中记录着相应的特权级，存储管理将会不断地判断其访问优先权，来决定当前访问是否合法。

3．多任务系统

80286 微处理器在保护模式下工作的主要特点就是具有严格的存储管理和任务管理功能，支持多任务操作。

多任务系统实际上是能够同时执行两个以上程序的系统。80286 本身并不能同时执行多个程序，只是在操作系统的调度下，将 80286 的执行时间进行划分，分别分配给不同的任务，各任务是分时、间隔运行的。80286 的任务状态段 TSS 是保护方式下使用的一种特殊段，对于每一个任务，都对应一个任务状态段 TSS。任务的定义、转换，都是通过对 TSS 的管理来实现的。

2.3 从80X86到Pentium

随着微型计算机的广泛应用，人们对微型计算机的性能提出了越来越高的要求；客观上由于大规模集成电路技术的迅速发展，使得微处理器及有关外围芯片的集成度不断提高，功能也越来越强。微型机系统从 8 位机、16 位机推进到 32 位机领域。特别是从 20 世纪 80 年代后期开始，由于超大规模集成电路集成度的不断提高，设计手段日益完善，加上体系结构设计概念的革新，新一代微处理器在各方面取得了巨大进展，就其运行速度而言，已可与大型机相比。由 32 位微处理器构成的超级微型机在实时控制、事务管理、工程计算、数据处理、人工智能以及计算机辅助设计、辅助制造等方面都得到了广泛应用。

从微型计算机总的发展情况看，一方面是迅速提高微处理器的性能，另一方面在系统设计上追求综合性能。当前微处理器获得高性能的主要方法是：提高集成度，更加全面地采用大中型计算机体系结构的设计技术。在提高微处理器性能的同时，在提高系统可靠性、利用微型机构造多机系统、充分利用资源及进行分布处理等方面也有重要进展。

就微处理器而言，新一代微处理器的设计融入了大型机的体系结构特点，如内部的存储管理、指令高速缓存和数据高速缓存、高度的并行性和流水线、大的寄存器组、多机处理接口甚至片上带有协处理器。

在新一代微处理器中，为了进一步提高运行速度，采用数据和指令分开存储和分开读、

写的方法，使用多个数据、地址总线，并使片内的存储管理部件和转换后备缓冲器与微处理器并行工作。

在新一代微处理器中，广泛采用精简指令系统计算机（RISC）体系结构。RISC 技术可以缩短计算机的设计周期、提高设计可靠性，特别是有较高的性能价格比，不仅应用在大型机、巨型机中，也越来越多地应用在微型机中。

新一代微处理器能全面支持多用户、多任务的操作系统，直接支持常用的高级语言。新一代微处理器还具有很强的联网功能，符合有关标准规定，支持多种形式的网络。

含有多个微处理器的单片芯片已经出现，系统集成已成为现实。多机处理与并行处理是并行度进一步提高的途径，而且多机处理与并行处理已可在单片超大规模集成电路芯片上实现。

随着计算机技术和半导体集成电路技术的不断提高，高档的微处理器与微型计算机正在迅猛发展。

2.3.1 80386/80486CPU 的结构及性能特点

80386 和 80486 都是 Intel 公司高性能的 32 位微处理器。80386 是 Intel 公司于 1985 年推出的产品，整个芯片采用 132 脚的陶瓷网格阵列封装。而 80486 是 Intel 公司 1989 年 4 月推出的 80386 的升级产品。

1．80386 的结构与特点

80386 微处理器由 6 部分组成，即总线接口部件、预取部件、指令译码部件、执行部件、分段部件和分页部件。其结构如图 2-18 所示。

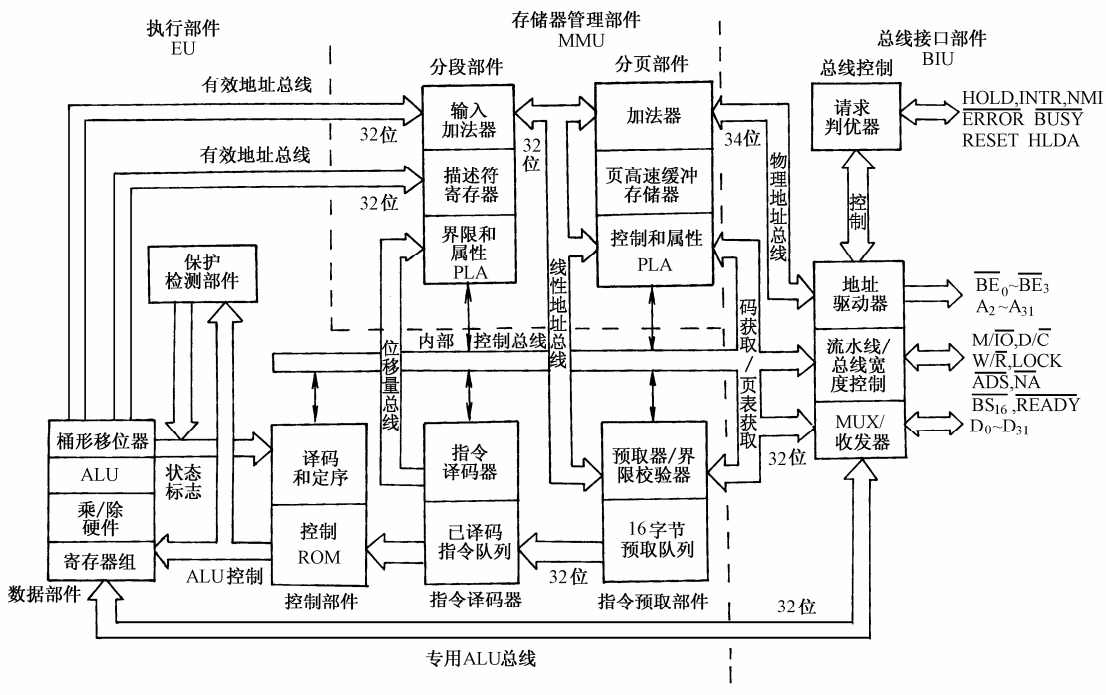


图 2-18　80386 的基本结构

（1）总线接口部件 BIU。总线接口部件是微处理器与系统的高速接口，它控制着 32 位的数据总线和地址总线。其功能是：在取指令、取数据、分段部件请求和分页部件请求时，有效地满足微处理器对外部总线的传送请求。总线接口部件被设计成能接收多个内部总线请求，并且能按优先权加以选择，这些动作是与当前的总线操作重叠进行的。

（2）预取部件 PU。预取部件的职责是从存储器预先取出指令。它有一个能容纳 16 条指令的队列，预取部件把取出的指令存放在队列中，以便于指令译码部件进行有效的译码。每当预取代码队列中有一部分已经变空，或者发生一次控制转移之后，预取部件就发出预取总线周期的请求信号。预取部件总线周期请求的优先级别低于与执行有关的取、存操作数的总线周期的请求级别，也低于页面没有命中的处理与分段专用总线周期请求的优先级别。当总线空闲周期到来时，就从存储器预取代码，并保持代码队列总是满的。

（3）指令译码部件 IDU。指令译码部件的职责是对指令进行译码，并且做好执行部件处理的准备工作。该部件从预取部件的指令队列中取出指令字节，对它们进行译码并将翻译好的代码存入自身的已译码指令队列。

（4）执行部件 EU。执行部件由控制部件、数据处理部件和保护测试部件组成。控制部件中包含着控制 ROM、译码电路等微程序驱动机构。数据处理部件中有 8 个 32 位通用寄存器、算术逻辑运算器 ALU、1 个 64 位桶形移位器、1 个乘除法器以及专用的控制逻辑，它执行控制部件所选择的数据操作。保护测试部件用在微程序控制下，执行所有静态的与段有关的违章检验。

为了提高执行部件的处理速度，把每条访问存储器的执行与前一条指令的执行部分地重叠，并且还把微指令的取指令操作和执行操作重叠起来，从而明显地提高了 80386 的工作速度。

（5）分段部件和分页部件。分段部件根据执行部件的要求，完成有效地址的计算，实现从逻辑地址到线性地址的转换。同时还要由保护测试部件完成总线周期分段的违章检查。转换好的线性地址与总线周期操作信息一起发送给分页部件。

分页部件将分段部件产生的线性地址转换成物理地址，这种转换是通过两级页面重定位机构来实现的。80386 中每一页为 4KB，每一段可以是一页，也可以是若干页。分页部件提供对物理地址的管理。

2．80486 的结构与特点

80486 基本上沿用了 80386 的体系结构，以保持与 86 系列微处理器在机器码级上的兼容性。80486 由 8 个基本部件组成：总线接口部件、指令预取部件、指令译码部件、执行部件、控制部件、存储管理部件、高速缓存部件和高性能浮点处理部件。图 2-19 为 80486 的结构框图。

从总体情况看，80486 有如下特点：

（1）80486 在 Intel 微处理器历史上首次采用了 RISC 技术，有效地优化了微处理器的性能。采用 RISC 技术并不意味着 80486 与 80386 等微处理器不兼容，实际上 80486 的指令并没有精简，强调的只是 RISC 技术。采用 RISC 技术的目的是使 80486 达到一个时钟周期执行一条指令。事实上，80486 能达到平均一个时钟周期执行 1.2 条指令。

（2）80486 采用突发总线同外部 RAM 进行高速数据交换。通常微处理器与 RAM 进行数据交换时，取 1 个地址，交换 1 个数据。采用突发总线后，每取得 1 个地址，便将这个地址及以后地址中的数据一起参与交换，从而大大加快了微处理器与 RAM 间的数据传送率。这

种技术尤其适用于图形显示和网络应用，因为在这两种情况下，所涉及的地址空间一般都是连续的。

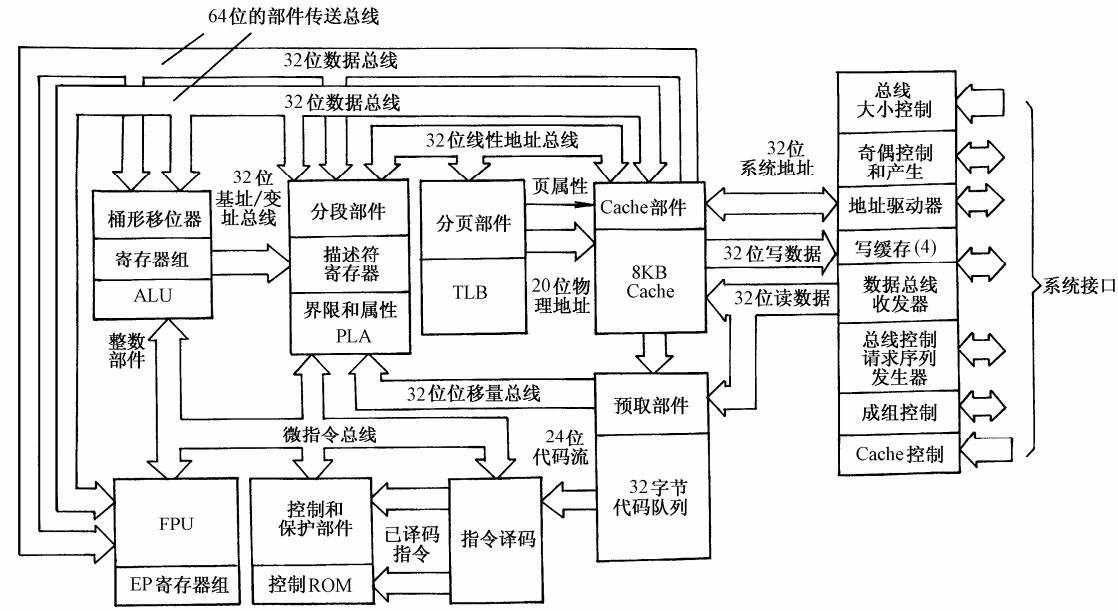


图 2-19　80486 的结构框图

（3）80486 微处理器中配置了 8KB 的高速缓存器（Cache）。高速缓存采用 4 路相连的实现方案，具有较高的命中率。高速缓存由指令和数据共用，若在高速缓存中找不到所需数据，可访问片外的存储器，一次可从片外调入 16B 的指令或数据。

（4）80486 微处理器内部还设置了一个数值协处理器，这就使得 80486 不再需要片外 80387 的支持而直接具有浮点数据处理能力。这个协处理器可以极高的速度进行浮点数运算，且与 80387 兼容。

此外，80486 在其高速缓存部件与协处理器之间设置了两条高速数据总线，这两条 32 位的总线也可作为一条 64 位的总线使用。高档 80486 芯片的数据总线宽度可达 128 位。如此宽的数据交换通道为数据高速处理提供了保证。

2.3.2　Pentium 系列 CPU 的结构及特点

Intel 微处理器发展至今已有六代。前四代是 8086、80286、80386、80486。第五代是 Pentium 和 Pentium MMX，第六代是 Pentium Pro、Pentium II 和 Pentium III。前四代的微处理器性能比较如表 2-3 所示。

表 2-3　Intel X86 微处理器性能

推出时间	CPU 类型	浮点处理器	指令集规模	工作时钟	数据线	地址线	物理空间	内部缓存	集成晶体管数目	引脚数
1978.6	8086	8087	133 条	6、8	16 位	20 位	1MB	无	29 000	40
1982.2	80286	80287	143 条	8、20	16 位	24 位	16MB	无	130 000	68
1985.10	80386	80387	154 条	12、33	32 位	32 位	4GB	无	275 000	132
1989.6	80486	内含	160 条	25、33	32 位	32 位	4GB	8KB	1 200 000	168

对于每一代微处理器，除上述所列的代表产品以外，还有一些其他产品，如 8088 是 8086 数据总线为 8 位的产品，80386SX 是 80386 外部总线与 80286 兼容的产品，80486SX 是 80486 片内不含浮点处理器的产品，80486DX2、80486DX4 是指 CPU 内部时钟频率是片外时钟频率的 2 倍频和 3 倍频的产品。

从表中可以看出 Intel 微处理器的发展特点：

（1）从 1978 年 6 月以来，Intel 公司遵循 MORE 定律，基本上每隔 18 个月就推出一个新的微处理器，在间隔期内对现有芯片的性能加以改进。

（2）8086 工作于实模式，80286 可工作于实模式和虚地址保护模式，80386 增加了虚拟 8086 模式。80386、80486 不仅能运行 16 位代码程序，也能运行 32 位代码程序，这是能采用 Windows NT 或 Windows 95 操作系统的最基本的硬件条件。

（3）性能的增加还包括：浮点运算器由片外到处理器片内；从片内无高速缓存器到有 8KB 的 Cache。

（4）微处理器时钟由 6MHz 发展到 33MHz，并且从 80486 起，微处理器普遍采用倍频技术，处理器的工作速度因时钟频率的加快而获得了很大的提高。

1993 年 3 月，Intel 公司推出 Pentium 微处理器，即“奔腾”芯片。Pentium 即 586，由于美国法院判决数字不能成为商标法律保护的对象，X86 就不再是 Intel 公司的专有商标了。所以 Intel 公司为 586 起名为 Pentium。Pentium 前 3 个字母“Pen”在拉丁文中代表“5”，象征 Intel 公司开发的第五代微处理器，而后面的字母“tium”的读音像某一种元素的名字。随着 Pentium 的出现，微处理器的发展又进入了一个新阶段。此后，Intel 公司又推出了 PentiumⅡ、PentiumⅢ等一系列产品。

1．Pentium 微处理器

Pentium 微处理器采用亚微米级的 CMOS，实现了 0.8μm 技术，一方面使器件的尺寸进一步减小，另一方面使芯片上集成的晶体管数达到 3 100 000 个。

在 Pentium 微处理器的体系结构上，采用了许多过去在大型机中才采用的技术，迎合了高性能微型机系统的需要。Pentium 体系结构如图 2-20 所示。

Pentium 微处理器采用的先进技术主要体现在超标量流水线设计、双高速缓存、分支预测、改善浮点运算等方面。

超标量流水线设计是 Pentium 处理器的核心。它由 U 和 V 两条指令流水线构成，每一流水线都拥有自己的 ALU、地址生成电路和与数据 Cache 的接口。这种流水线结构允许 Pentium 在单个时钟周期内执行两条整数指令，即实现指令并行，且 V 流水线总是接收 U 流水线的下一条指令。

Pentium 采用双 Cache 结构，每个 Cache 为 8KB，数据宽度为 32 位。两个 Cache 中，一个作为指令 Cache，另一个作为数据 Cache。数据 Cache 有两个接口，分别通向两条流水线，以便能在同一时刻与两个独立工作的流水线进行数据交换。双高速缓存的使用，大大节省了微处理器的时间。

Pentium 微处理器中还设置有分支目标缓冲器 BTB，它实际上是一个较小的高速缓存器，用于动态地预测程序分支。当一条指令导致程序分支时，BTB 会记住这条指令和分支目标的地址，并用这些信息预测这条指令再次产生分支时的路径，并预先从此预取，保证流水线的指令预取步骤不会空置。当 BTB 判断正确，分支程序即刻得到解码。

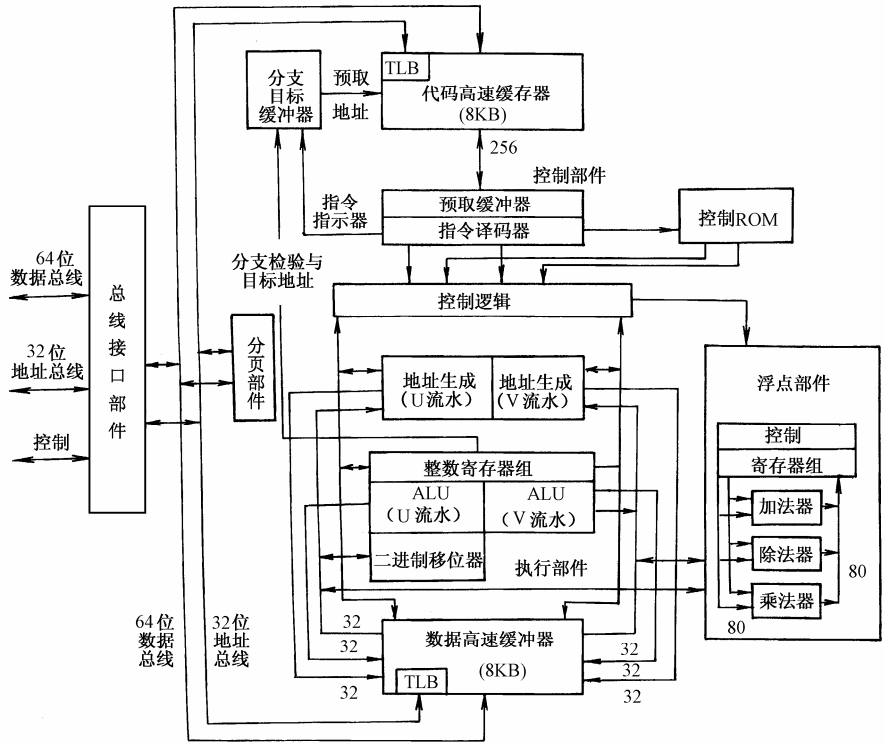


图 2-20　Pentium 结构示意图

为了加强浮点运算能力，Pentium **中的浮点运算部件在** 80486 **的基础上进行了彻底改进，其执行过程分为** 8 **级流水线，使每个时钟周期至少能完成一个浮点操作。浮点运算部件对一些常用指令采用了新的算法，并用电路进行固化，硬件的使用使得运算速度大为提高。**

2．Pentium Pro **微处理器**

Intel **公司于** 1995 **年底推出的** Pentium Pro **微处理器，是第六代微处理器系列的第一个产品。**Pentium Pro **比普通** Pentium **增加了** 8 **条指令，与** X86 **微处理器系列完全向下兼容。**Pentium Pro **微处理器具有** 64 **位数据线、**36 **位地址线。**197mm^2 **的芯片上集成了** 5 500 000 **个晶体管。**

Pentium Pro **主要有** 3 **大特点：动态执行技术、在片** L2 Cache**、支持多处理器系统。**

（1**）**Pentium Pro **采用了** RISC **技术，超标量与流水线相结合的核心结构实现了动态执行技术。每个时钟周期可执行** 3 **条指令，可推测执行** 30 **条指令，特别适合于多线程的** 32 **位程序运行。**

（2**）**Pentium Pro **处理器使用的是一种** 387 **管脚网格阵列（**PGA**）的陶瓷封装技术，片内除** CPU **外，集成的** L2 Cache **为** 256KB **或** 512KB**，这个** L2 Cache **能以处理器的工作时钟高速运行。**

（3**）**Pentium Pro **处理器支持不加附加逻辑的对称多处理，即不需要额外的逻辑电路就可支持** 4 **个** CPU**，这一结构对服务器、工作站实现多处理器系统特别有利。**

3．Pentium MMX 微处理器

1997 年 1 月 9 日，Intel 公司正式推出了 P55C 微处理器，即 Pentium MMX，其中“ MMX ”是“ Multi Media eXtension ”的英文缩写，意为“ 多媒体扩展 ”。这是为提高 PC 机处理多媒体和通信能力而推出的新一代处理器技术，它是通过在 Pentium 处理器中增加 4 种新的数据类型、8 个 64 位寄存器和 57 条新指令来实现的。

MMX 技术是 Intel 80X86 微处理器体系结构的重大革新，增加了很多技术，主要有：

（1）引入了新的数据类型。MMX 定义了 4 种新的 64 位数据类型及其紧缩（又称“ 压缩 ”）表示，它们是紧缩字节（8 个字紧缩在一个 64 位数据中）、紧缩字（4 个字紧缩在一个 64 位数据中）、紧缩双字（2 个双字紧缩在一个 64 位数据中）和 4 字（一个 64 位信息）。新增加的 8 个 64 位通用寄存器能够保存各类紧缩的 64 位数据。这对多媒体处理十分有用，例如处理一幅 256 级灰度的图像，图像像素数据通常以 8 位整数的字节表示，而用 MMX 技术，8 个这样的像素将紧缩为一个 64 位，可移入一个 MMX 寄存器。当一条 MMX 指令执行时，将从 MMX 寄存器中对所有 8 个像素值并行地完成其算术或逻辑运算，并将结果写入 MMX 寄存器，这样用 MMX 指令进行一次紧缩字节操作，相当于处理了 8 个像素。

（2）采用饱和运算。饱和运算也是 MMX 支持的一种新的运算，与常用的数据处理相比较，饱和运算的优点表现在：在常规运算中，上溢和下溢的结果均被截断，只有结果的低位能被返回，而在饱和运算中，上溢和下溢的结果都被截取为该数据类型的最大值和最小值。这种运算在图型处理中很有用，例如，在对一个暗色多面体按黑色做浓淡处理时，可以避免中间突现一个白色像素。

（3）具有积和运算能力。在多媒体应用程序中，必须处理大量数据。矢量点积和矩阵乘法是处理图像、音频、视频数据的基本算法，用 MMX 的 PMADDWD 指令（即积和运算）可以大大提高矢量点积的运算速度。这在音频和视频图像的压缩和解压缩中经常用到。

4．Pentium Ⅱ微处理器

1997 年 5 月 Intel 公司正式推出了 Pentium Ⅱ微处理器。它是 Pentium Pro 的先进性与 MMX 多媒体增强技术相结合的新型第六代微处理器。它采用 0.35μm 的 CMOS 半导体技术，片内集成 7 500 000 万个半导体元件，片内有 L1 Cache 为 32KB，L2 Cache 为 512KB。前期 Pentium Ⅱ的 4 档产品的工作频率分别为 233 MHz、266 MHz、300 MHz 和 333 MHz。

Pentium Ⅱ的优异性能与先进结构主要体现在以下 3 方面：

（1）动态执行技术与 MMX 技术。与 Pentium MMX 一样，Pentium Ⅱ也集成了 MMX 技术，增加了 57 条 MMX 指令，增强了音频、视频和图形等多媒体应用的处理能力，也加速了数据加密和数据压缩与解压过程。同时，与 Pentium Pro 一样，Pentium Ⅱ采用了先进的核心结构，具有包括数据流分析、转移预测和推测执行在内的动态执行技术。

（2）双重独立的总线结构。Pentium Ⅱ微处理器内部总线宽度已高达 32 位，外部为 64 位总线，即数据总线宽度为 64 位、地址总线宽度为 36 位，寻址空间为 64GB，虚拟地址空间为 64TB。

Pentium Ⅱ处理器核心外部采用了双重独立总线结构，即具有纠错功能的 64 位 CPU 总线负责与系统内存和 I/O 通信，具有可选纠错功能的专用总线负责与 L2 Cache 交换数据，这样解决了 Pentium 和 Pentium MMX 单一总线结构中由于 CPU 工作时钟速率成倍提高而在

CPU 和 L2 Cache 数据交换中所出现的瓶颈问题。

（3）SEC 单边接触封装技术。为了双重总线结构的需要，Pentium Ⅱ处理器封装采用了一种新型的单边接触 SEC（Single Edge Contact）卡式盒结构。SEC 卡是一块带金属外壳的印制电路板，上面集成有 Pentium ⅡCPU 芯片和 32KB 的 L1 Cache。CPU 芯片的管芯只有 203mm^2，采用的是一种 528 管脚的网格阵列（PLGA）封装技术。SEC 卡要插接到主板上被称为 Slot 1 的插槽中。

5．Pentium Ⅲ微处理器

Intel 公司于 1999 年 1 月正式宣布 Pentium Ⅲ处理器问世，并于 2 底正式上市。Pentium Ⅲ处理器采用 0.25μm 的 CMOS 半导体技术，处理器核心集成有 9 500 000 万个晶体管。一上市，Pentium Ⅲ即有主频为 450MHz、500MHz 和 550MHz 3 种型号的产品。Pentium Ⅲ的结构与 Pentium Ⅱ相仿，它与 Pentium Ⅱ的最大不同在于以下 3 点：

（1）Pentium Ⅲ也是采用双重独立总线结构，但是前端总线的时钟频率至少为 100MHz，处理器核心与 L2 Cache 之间专用的后端总线时钟频率最初是主频的一半，但以后的产品也有与主频同速的。

（2）Pentium Ⅲ处理器首次采用了 Intel 公司自行开发的流式单指令多数据扩展 SSE（Streaming SIMD Extension），这包括 70 条 SSE 指令集和新增加的 8 个 128 位单精度浮点数寄存器。SSE 技术使得 Pentium Ⅲ处理器在三维图像处理、语言识别、视频实时压缩等方面都有很大进步，而这个优点在互联网应用中得到了充分地体现。

（3）Pentium Ⅲ微处理器首次设置了处理器序列号 PSN（Processor Serial Number）。PSN 是一个 96 位的二进制数，制造芯片时它被编入处理器晶片的核心代码中，可以用软件读取但不能修改。PSN 的作用相当于处理器和系统的标识符，可用来加强资源跟踪、安全和内容管理。

本 章 小 结

微处理器是微型计算机的核心。由于微处理器产品的更新速度非常快，因此，对微处理器的学习，既要掌握微处理器的一般原理，又应该知道具体的典型产品。本章既有一般微处理器的阐述，又有流行 CPU 的介绍。

对于微处理器，有两个方面：内部结构和外部引脚。内部结构决定了微处理器的功能，外部引脚是微处理器功能的具体表现。由于微处理器的具体应用涉及到芯片间的连接，因此对外部引脚功能的了解十分重要。引脚功能既包括引脚的静态功能定义，也包括其动态功能（即时序关系）。

以 Intel 系列作为典型产品符合当前的实际需要。本章以 80286 这一 16 位的 CPU 为主，介绍了 CPU 的内部结构与外部引脚功能，同时也介绍了 Intel 系列 32 位微处理器的特点与功能。

习 题 2

2.1 一般微处理器内部由哪几部分组成，各自承担什么工作？

2.2 微处理器有哪些常用的外部控制信号线？这些控制信号线的不同组合，能代表哪些微处理器的常见总线操作？

2.3 说明指令周期、总线周期、时钟周期三者的关系。

2.4 与一般微处理器相比，80286 在结构上有什么特点？这些特点对提高性能有哪些好处？

2.5 80286CPU 有哪些总线状态，它们之间的关系如何？一个标准的 80286 总线周期由哪些总线状态组成？

2.6 构成 80286 基本的系统结构除了 CPU 以外，还需要哪些功能部件？各自具有什么功能？

2.7 试以 80286CPU 某一次存储器读操作为例，叙述一次总线操作的过程及各控制信号的变化情况。

2.8 与 80286CPU 相比较，分析 80386、80486 在体系结构上有些什么变化？

2.9 从 Pentium 系列微处理器的发展来看，在结构上主要运用哪些先进技术？了解 Pentium 4 微处理器的性能特点。

第 3 章　存储设备及接口

存储器是计算机的重要组成部分，分为内存储器和外存储器。内存储器在主机内部，也称主存或内存；而磁盘磁带等存储设备在主机外部，属于外存储器，也称辅助存储器，简称外存或辅存。

主存作为计算机记忆信息的装置，用于存放数据、符号等信息，并可随机取出这些信息供给计算机的其他部件。面对计算机信息处理量的激增，总是希望存储器能够容纳更多的信息；同时为了 CPU 能高速处理存储器中的信息，又希望存储器的存取速度尽量与 CPU 的速度匹配。从某种意义上说，CPU 执行指令的速度取决于主存储器的存取速度。

辅助存储器的速度允许慢一些，价格相对低廉，存储容量很大，大量静止的、待命的后援信息分布在辅存中。一旦需要，就可以将辅存中的信息调入主存供 CPU 访问。

因此，主存和辅存所构成的二级存储体系很好地形成了一个计算机的存储器系统：内存解决了能及时为 CPU 提供存放信息的空间，而外存则提供了存放大量的计算机后备数据的存储空间。二者很好地协调了有关存储器容量、速度、价格之间的矛盾。

除了二级存储体系以外，由高速缓冲存储器（Cache）、内存、外存组成的三级存储体系在存储器系统的整体性能上得到了更好的体现。

3.1　存储器概述

3.1.1　存储器分类

从不同的角度考虑，存储器有着不同的分类方法。例如，按存储器在计算机系统中的地位，可分为内存储器和外存储器；按存储特性，可分为易失性存储器和非易失性存储器；按寻址特征，可分为随机访问存储器、顺序访问存储器和直接访问存储器；按存储介质和存储器的工作原理，则可分为半导体存储器、磁介质存储器和光碟存储器，如图 3-1 所示为存储器分类示意图。

1．半导体存储器

半导体存储器具有体积小、集成度高、外围电路简单等优点，在计算机系统中被广泛应用。计算机系统的主存和各种外部设备的缓存，都是由半导体存储器担任的。

半导体存储器可分为易失性存储器和非易失性存储器两类。

（1）易失性存储器。易失性存储器主要是指随机访问存储器 RAM（Random Access Memory）。RAM 是计算机中用来存放数据、程序及运算结果，直接与 CPU 进行信息交换的场所。按工作原理可分为静态 RAM（SRAM）和动态 RAM（DRAM）两种。

SRAM 的基本工作原理是利用晶体管触发器的记忆功能和门电路的选通功能。存储单元的每一位都是由双稳态触发器和选通门电路组成的，而整个存储器由存储单元阵列和控制电路组成。SRAM 的主要特点是工作稳定、存取速度快、连接使用方便且不需要刷新。但由于

每一位存储单元都要使用好几个晶体管，因此功耗较大。由于集成度和价格原因，SRAM 很少用在计算机系统的主存中，主要用做高速缓冲存储器（Cache），或者用于小规模存储器系统如用于控制的小型计算机系统中的内存。

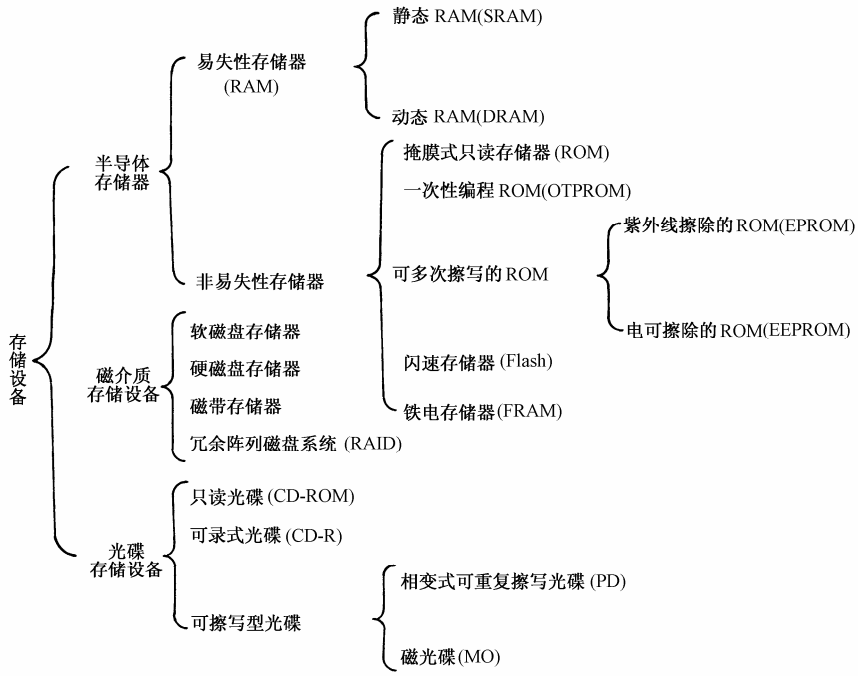


图 3-1　存储设备分类

SRAM 还有一些其他的表现形式，如双端口 SRAM 和 FIFO 存储器。双端口 SRAM 是具有两个可以独立访问任意存储单元端口的存储器，主要用于某一些具有两个或两个以上要求访问存储器的主控模块的系统。FIFO 存储器也是一种双端口 SRAM，在两个端口中，一个用于写入数据，另一个用于读出数据，写入和读出遵循先进先出原则。

DRAM 的存储原理是基于电容能存储电荷的特性。存储单元的每一位可用一个晶体管和一个与之相连的小电容来实现。若写入位为“1”，则电容被充电；写入位为“0”，电容不充电。读出时，用晶体管来读与之相连的电容电荷状态便可知“0”或“1”。

由于 DRAM 的每一个存储位仅用一个晶体管和一个小电容，它的集成度可以做得很高。就单个芯片的存储容量而言，DRAM 可以远远超过 SRAM；就相同容量的芯片而言，DRAM 的价格也大大低于 SRAM。这两个优点使 DRAM 成为计算机内存的主角。

但由于 DRAM 用电容来存储数据，而电容极其微小且有漏电存在，会使电容存储的电荷逐渐泄放掉，从而造成数据丢失。因此，需要对 DRAM 进行刷新，就是定时地对每个存储单元进行一次重新充电。刷新工作是按行进行的，在同一行的所有位同时被刷新。

由于需要刷新，一般连接 DRAM 的电路要复杂一些。另外，与 SRAM 相比，DRAM 的存取速度较慢。

（2）非易失性存储器。非易失性半导体存储器广泛用于程序、字符、常数的存储，种类很多，主要有以下几种：

掩膜式只读存储器 ROM 是生产厂家根据用户的要求用掩膜加工的方法制造的存储器芯

片，其内容不能修改。这是一种最早的 ROM，目前已被淘汰。

一次性编程的 OTPROM（One Time Programmable ROM）即可以一次性编程写入的 ROM，编程写入后，只能读取，不能再写入。目前由于可多次编程的 ROM 芯片的品种不断增加以及性能不断提高，一次性编程的 ROM 芯片已经很少使用了。

用紫外线擦除的可编程存储器 UV-EPROM（Ultraviolet EPROM）通常简称 EPROM，其性能是：未写入时，每个存储单元都呈“1”状态；在编程写入时，根据需要使相应位呈“0”状态；擦除时，用紫外线通过芯片顶部的石英窗口对芯片进行照射约 20min，芯片中所有“0”状态的存储单元全部变为“1”。

电可擦除的可编程存储器 EEPROM（Electrically EPROM）也称 E^2PROM，它不像 EPROM 那样必须脱机用紫外线来擦除，而是用电来擦除，可以直接由计算机联机进行编程和修改。它既可以整片擦除，也可以按字节进行擦除和再编程，从而克服了一般 EPROM 的缺点。

闪速存储器 Flash Memory 是一种可以用电快速擦写的非易失性存储器，简称 Flash。快速是相对 E^2PROM 而言的。从原理上看，Flash 存储器属于 ROM 型存储器，但它可以随时改写所存信息，从功能上看又相当于 RAM，使以前对 RAM 与 ROM 的划分变得模糊起来。但从存取速度和擦写的寿命两方面来看，它还赶不上 RAM，因此，在计算机中目前还是作为 ROM 来使用。在工业控制、办公设备等领域，Flash 可用做在线改写的 ROM。

铁电存储器 FRAM（Ferroelectric RAM）是一种非易失性存储器新技术，它克服了 Flash 存储器的擦写速度慢以及可写次数寿命不长等缺点而备受人们的关注。FRAM 的基本存储原理是：当铁电电容两端加电压时，根据电场方向不同，铁电电容的介质被极化为两种状态中的一种，当电场撤销以后，其极化状态仍能保持，即体现出非易失的特性。目前，国际上许多生产厂家都在大力开展 FRAM 的研究和开发工作，FRAM 极有可能成为一种理想的存储器。

2．磁介质存储器

磁介质存储器是利用涂敷在盘或带表面磁性材料的均匀薄层来存储数据的。

磁介质存储器要存储数据，光有磁介质是不够的，还必须有电磁进行相互转换的装置，即磁头。当磁头上的线圈中有电流流过时，会在其间隙处产生磁场，磁介质会在磁头间隙处被磁场磁化，磁化方向取决于磁头线圈中的电流方向。不同的磁化方向确定了存储在磁介质上的不同信息。随着磁介质以一定的速度与磁头做相对运动，就可以在磁介质的不同位置上记录相应的信息。这就是磁介质存储信息的基本原理。同样，在读取数据时，磁介质也与磁头做相对运动，磁介质上的磁化极性会在磁头线圈上产生电动势。不同的磁化方向产生不同的方向的感应电动势，表示读出不同数据。

磁介质存储器包括磁盘和磁带，其中磁盘分软磁盘和硬磁盘两种。

（1）软磁盘存储器。软磁盘简称软盘，软盘存储器包括软盘和软盘驱动器，软盘用于存储数据，驱动器用于读写数据。

软盘是一种表面涂有磁性材料的塑料圆盘，按其直径尺寸可分为 8 英寸、5.25 英寸和 3.5 英寸 3 种规格。其中 8 英寸软盘早已淘汰，5.25 英寸的软盘已基本淘汰，当前主流软盘是 3.5 英寸软盘。

软盘的双面都可以存储数据，每一面都被分隔成多个与盘片中心同轴且等宽的同心圆

环，每个圆环称为一个磁道。每个磁道又被均匀地分成若干段，称为扇区。每个扇区可以存储若干字节的数据，数据是沿磁道逐位串行存储的。

除普通软盘外，目前还出现了一些高容量的软盘存储器，如 ZIP 软盘及驱动器。ZIP 软磁盘存储器是美国 IOMEGA 公司生产的系列产品，软盘存储容量为 100MB，其软驱的数据传输速率可达 1.5MB/s，远高于普通软驱。

（2）硬磁盘存储器。硬磁盘存储器简称硬盘，其盘片和盘驱动器封装在一个密封腔内。硬盘由盘片、读写磁头、磁头驱动机构、主轴电机等部分构成，其基本原理与软盘驱动器相同，但结构要比软盘驱动器复杂得多，也精密得多。硬盘具有存储容量大、读写速度快的特点，是计算机系统最主要的存储设备。

除普通硬盘以外，磁盘阵列 RAID 也是一种很有特色的存储设备。RAID（Redundant Array of Independent Disk）即独立的冗余磁盘阵列，它是指用多台独立的硬盘存储器组成的一个快速的、超大容量的外存储子系统。RAID 具有存储容量大、可靠性高、内部阵列控制器拥有固化程序对磁盘阵列进行各项管理等功能。RAID 作为一种高可靠性海量存储系统，主要用于对存储器容量要求非常大、可靠性要求也非常高、单个硬盘不能胜任的系统，如大中型网络服务器和主机系统。

（3）磁带存储器。磁带是资格最老的存储介质之一，磁带存储技术，至今仍有广泛地应用。随着信息技术的发展，有大量的信息资料需要存档，需要进行数据备份保存，磁带存储器成为数据备份的首选目标。

磁带存储器之所以能成为数据备份的首选，是因为其具有如下优点：存储容量大、存储成本低、存储可靠性高。磁带存储器作为顺序访问的存储器，主要缺点就是存取速度慢，不利于联机使用，而作为数据备份，不需要随机访问，访问速度不是主要问题。

3．光介质存储器

光碟也称光盘，因其有存储容量大、性能可靠、便于携带与保存等特点，已被广泛应用。按数据的读写特性，光碟存储器分为只读光碟存储器（CD-ROM）、一次写入多次读取的光碟存储器（CD-R）及可重复擦写的光碟存储器（CD-RW）。

CD-ROM 是最早出现的一种光碟，是一种只读光碟，也是当前使用最为普遍的光碟。现在的微机都配置了 CD-ROM 驱动器。CD-ROM 碟片直径 120mm，厚度为 1.2mm，碟片表面第一层是涂漆保护层，第二层是铝反射层，再下一层才是聚碳酸酯压制的透明衬底。碟片上的信息就是通过模压在碟片上形成一系列凹坑点线的形式存储的。在读操作时，用激光对凹坑进行扫描，通过反射光的接收分析来辨别信息“0”或“1”。

CD-R 是一种可以一次性写入的光碟，通过专用的光刻录机对 CD-R 光碟实施写操作。CD-R 光碟可通过普通 CD-ROM 驱动器读取其中的信息。

可重复读写的光碟按数据存储原理和记录方式可分为相变光方式存储器和磁光方式存储器。后者因采用了光技术、磁技术、激光技术，故也称为光磁盘。

3.1.2 存储器主要技术指标

随着计算机制造技术的发展，存储器技术的发展也很快，各种存储器的性能都在不断提高。根据存储器应用场合的不同，对存储器的要求也有所差异。总体而言，对存储器的要求可从以下几方面考虑：

1．存取速度要快

存储器存取速度的快慢直接关系到整个系统的工作效率，尤其是作为直接运行程序的主存储器，更是希望其存取速度快，否则，处理速度再快的 CPU 也无法充分发挥其性能。因此，存取速度的快慢，应是首先要考虑的因素。

2．存储容量要大

存储器的存储容量越大，可以存储的信息越丰富，尤其是多媒体通信技术，要求存储器的容量很大。例如，一幅未经压缩的图像，就要存储数百兆字节。存储容量越大，存储的信息越多，计算机的运行速度也会越快。

3．功耗要低

存储器，特别是半导体存储器，都是由大规模集成电路组成的，集成度高，体积小，但是散热不容易，因此在保证速度的前提下应尽量减小功耗。

4．体积要小、重量要轻

体积越小，相对集成度则越高。无论是用于台式计算机，还是便携式计算机，体积小、重量轻同样是存储器所追求的性能指标。

5．可靠性要好

可靠性是指存储器对电磁场、湿度变化等因素造成干扰的抵抗能力，以及在高速使用时也能正确地存取。对于易失性存储器，在供电期间数据在未做修改的情况下存储稳定；对于非易失性存储器，非易失性要好，即断电以后，存储器中的数据仍能保持完整。

6．存取操作要方便

对存储器信息的存取通常都是根据需要，有选择地进行的，因此，希望存储器的存取操作越方便越好，想要读取或修改哪一部件存储内容，很快就能选中。

此外，任何产品，在性能相同的情况下，其价格越低，则产品越受欢迎，存储器也不例外。因此，对存储器的评价与选择，往往要综合考虑。对于存储器的要求，不可能各方面都能满足要求，所以需要根据应用的具体要求加以选择，并考虑其性能价格比。

3.1.3 半导体存储器结构

半导体存储器芯片的内部结构基本相同，都是由存储体和外围电路两部分组成的。存储体是由一系列按行/列排列的基本存储单元组成的，不同性质的半导体存储器，其基本存储单元电路有所不同。外围电路由地址译码器、I/O 电路、片选控制和输出驱动电路所组成。其中，地址译码器根据输入地址来选择存储单元，通常采用行/列双译码方式；I/O 电路介于数据总线与被选中的单元之间，用于控制被选中单元的读出或写入；片选控制电路用于控制本芯片是否被选中；输出驱动电路通常是三态输出，既便于连接数据总线，又具有驱动功能。

从内部结构可知，半导体存储器的外部引脚基本上分为 3 部分：地址线、数据线和控制线。地址线是输入的，用于连接内部地址译码器进行地址译码来选择存储单元；数据线是双

向的，用于连接存储单元进行数据交换；控制线基本上都是输入的，用于控制存储器芯片的操作，主要有读信号线、写信号线和片选信号线。

1．静态 RAM

静态 RAM 有多种型号，6264 是一个典型的芯片。6264 是一个 8KB×8 的 SRAM 芯片，有数据线 8 位 D_7～D_0，地址线 13 位 A_{12}～A_0，内部主要包括 256×256 的存储器矩阵、行/列地址译码器以及数据输入输出控制逻辑电路。在 13 位地址线中，A_{10}、A_2、A_1、A_0 用于列地址译码，其余地址线用于行译码。在存储器读周期，选中单元的 8 位数据经 I/O 控制电路输出；在存储器写周期，外部 8 位数据经输入数据控制电路的 I/O 控制电路写到所选中的存储单元中。当片选信号无效时，芯片数据线处于高阻状态。图 3-2 是 6264 芯片的内部结构图，表 3-1 提供了 6264 芯片引脚与芯片工作状态的关系。

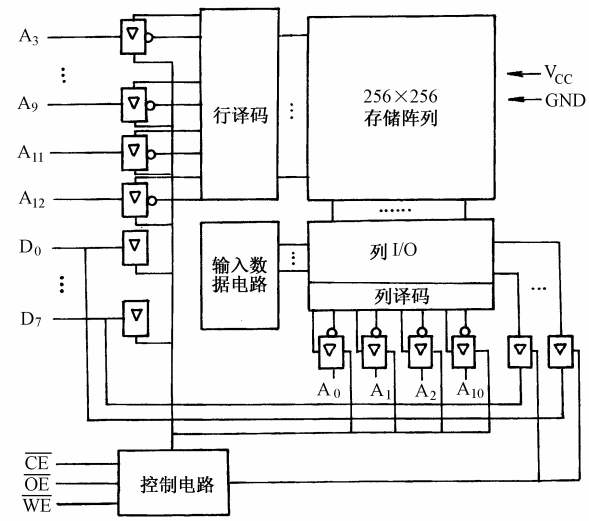


图 3-2　6264 内部结构图

表 3-1　6264 引脚功能与芯片工作状态

片选 $\overline{CE}$	输出控制 $\overline{OE}$	写信号 $\overline{WE}$	工 作 状 态
0	0	1	存储器读
0	1	0	存储器写
1	×	×	未选中

2．动态 RAM

尽管 DRAM 的工作原理与 SRAM 不同，但其芯片内部结构与 SRAM 基本相同，也是由存储体与外围电路组成的。由于 DRAM 集成度高，有动态刷新要求，故在具体构造上稍有区别：

（1）由于 SRAM 芯片的集成度高，存储单元多，因此地址线引脚也多，如 64KB 的存储器芯片就有 16 位地址输入线。为了减少封装引脚，地址线分行地址和列地址分时输入。这样，在 DRAM 的结构中，就要支持地址的行/列分时输入，其中包括具有行/列选通信号及相应的

控制电路、地址锁存器等。

（2）DRAM 的刷新是按行进行的，当行地址输入行地址锁存器以后，就可以对该一行内的所有存储单元进行一次刷新，也就是对存储的信号进行放大。为了完成刷新功能，在 DRAM 内部存储体电路中，每一列都有读出放大电路，以便在行地址选中以后对一行中的所有单元同时刷新。

4164 就是一个 64KB×1 的 DRAM 芯片，外部有 8 位地址输入线提供行地址和列地址、1 位数据输入线和 1 位数据输出线以及行列地址选通信号线等。图 3-3 是 4164 芯片的内部结构图。

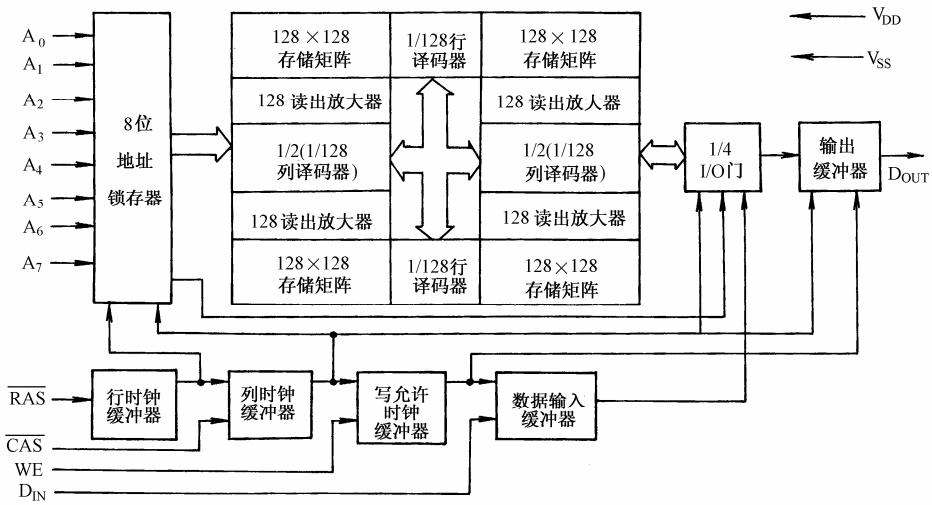


图 3-3　4164 内部结构图

64KB 的存储体是由 4 个 128×128 存储矩阵构成的，通过最高位的行地址和列地址选择不同的存储矩阵。每个 128×128 存储矩阵有 7 位行地址和 7 位列地址线进行选择。7 位行地址线经过译码产生 128 条选择线，分别选择 128 行；7 位列地址经过译码后也产生 128 条选择线，分别选择 128 列。

4164 的外部引脚有：

A_7～A_0：地址输入；

D_{IN}：数据输入线；

D_{OUT}：数据输出线；

$\overline{RAS}$：行地址选通，该信号有效，地址线输入行地址；

$\overline{CAS}$：列地址选通，该信号有效，地址线输入列地址；

$\overline{WE}$：写信号，当芯片选中时，低电平表示写状态，高电平表示读状态。

针对 4164 动态存储器，当 $\overline{RAS}$ 有效时，地址输入端输入行地址；当 $\overline{CAS}$ 接着有效后，输入列地址，同时芯片被选中，可以进行读写操作。图 3-4 和图 3-5 分别给出了 4164DRAM 的读操作时序和写操作时序。

当 4164DRAM 仅 $\overline{RAS}$ 有效时，芯片处于刷新状态，刷新行地址由 A_6～A_0 提供。图 3-6 是其刷新时序图。

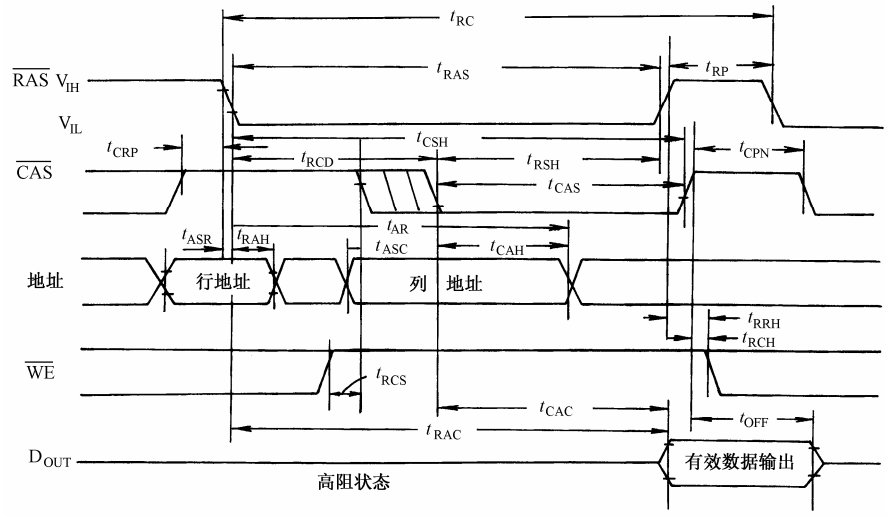


图 3-4　动态 RAM 读操作时序图

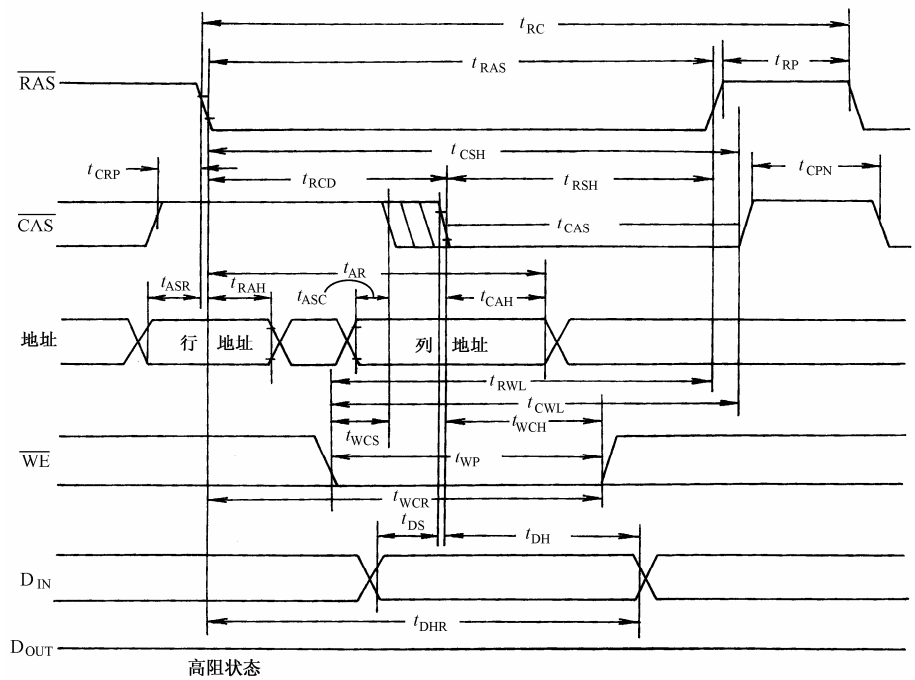


图 3-5　动态 RAM 写操作时序图

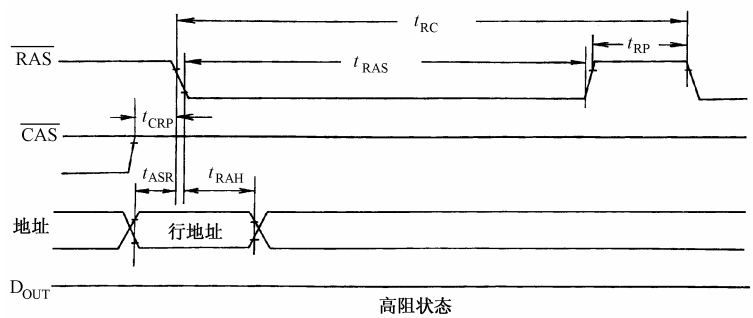


图 3-6　动态 RAM 刷新操作时序图

3．EPROM 存储器芯片

EPROM 的内部结构与 SRAM 基本相同，图 3-7 是 EPROM 27128 的结构图。27128 是一种 16KB×8 的紫外线可改写只读存储器芯片，与 SRAM 相比，具有编程逻辑电路而没有一般写控制电路。

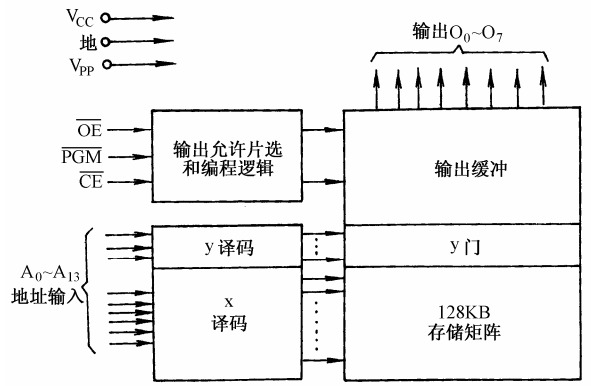


图 3-7　EPROM 27128 内部框图

EPROM 27128可工作于多种工作方式，其中主要的工作方式有：读方式、编程方式和备用方式。

（1）读方式。这是 27128 正常的使用方式，此时两条电源线 V_{CC} 和 V_{PP} 都接＋5V 电源，$\overline{PGM}$ 接高电平。每当要从一个地址单元读数据时，应先通过地址线 A_{13}～A_0 输入地址信号，接着片选信号 $\overline{CE}$ 和输出控制信号 $\overline{OE}$ 都有效，于是经过一段时间，指定单元的内容就可以读到数据线 O_7～O_0 上。图 3-8 为 EPROM 27128 的读操作时序图。

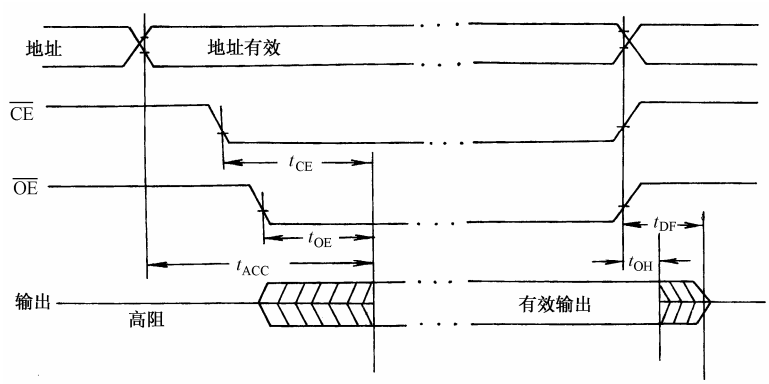


图 3-8　27128 读操作时序图

（2）编程方式。对 EPROM 的编程是以存储单元为单位进行的。编程时，从 O_7～O_0 输入待定写入的数据，电源 V_{CC} 接高电平，而 V_{PP} 接规定的编程电压值（一般为＋21V），$\overline{OE}$ 保持高电平，而编程控制信号 $\overline{PGM}$ 端必须提供一个 50ms 宽的负脉冲，完成一个单元的数据写入。图 3-9 所示为 27128 的编程时序。

（3）备用方式。当芯片未被选中时，为了降低芯片的功耗，设置了备用方式。当片选 $\overline{CE}$ 输入无效时，27128 芯片就处于备用方式，数据输出端为高阻状态。

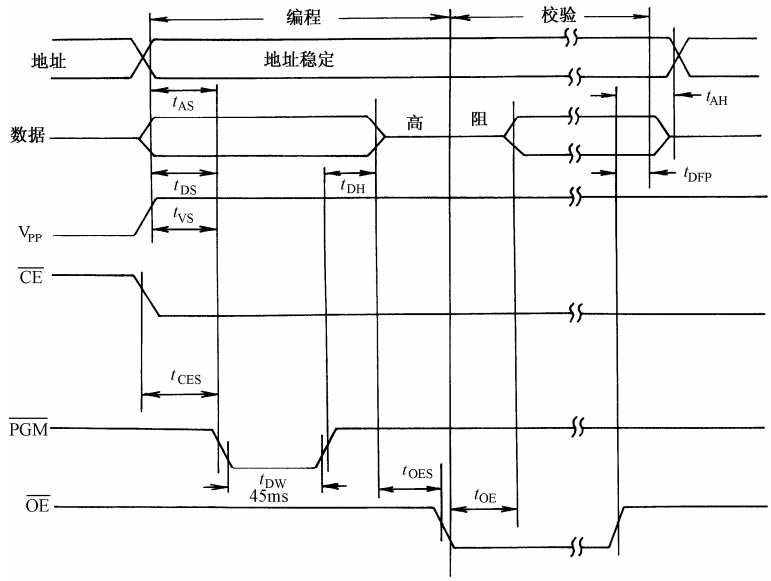


图 3-9　27128 的编程时序

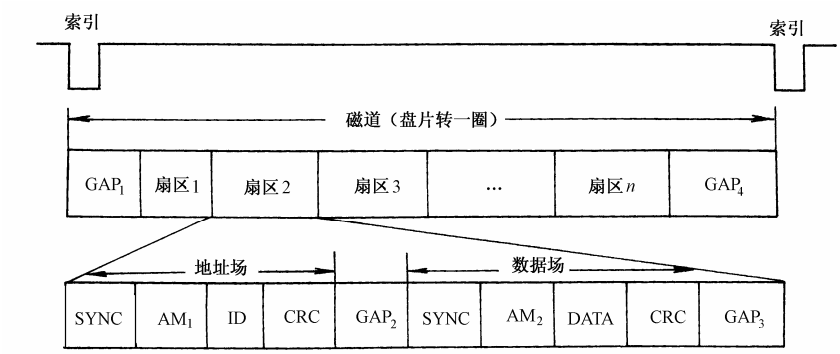
3.1.4　盘存储器数据格式

盘存储器主要涉及磁盘和光盘，两者都是以扇区为基本单位，其数据存储格式基本相同。

1．磁盘存储器

磁盘分软盘和硬盘两种。软盘双面都可以存储数据，每一面可分若干个磁道，每一个磁道分若干个扇区，每个扇区可存储若干字节。对于硬盘，由于有多片磁片，相同磁道组成了柱面，故磁道被柱面所取代。

为了保证磁盘读写数据的可靠性和通用性，对磁盘的读写必须按共同遵守的格式进行。磁盘的存取是以扇区为单位的，因此每一个扇区都有地址编号。磁盘地址由磁道号（柱面号）、磁头号、扇区号 3 部分组成。如图 3-10 所示为软盘记录格式，硬磁盘格式基本相同。



字节数	12	3	1	4	2	22	12	3	1	256 512 1 024	2	54 84 116
内容	00H	AIH	FEH	**	**	4EH	00H	AIH	FBH F8H	**	**	4EH

**：内容为可变

图 3-10　软磁盘记录格式

每一个磁道分为 3 大段：

（1）磁道首部。磁道首部由间隙 GAP_1 组成，为 32B 的 FFH，通过磁盘索引孔产生的索引脉冲来定位。磁道首部的作用是用于缓冲，以防止索引脉冲信号检测引起的误差。

（2）扇区段。扇区段由若干等长的扇区组成，扇区段分地址场、数据场两部分，扇区段之间是间隙 GAP。

地址场由同步字符（SYNC）、地址标志（AM_1）、地址字段（ID）和 CRC 校验码组成。SYNC 为 12B 的 00H，用于提供读盘时所需的同步时间。当检测到地址标志时，即表明其后为字段（ID），ID 占 4 个字节，依次为磁道号（C）、磁头号（H）、扇区号（R）和每个扇区字节数（N），最后两个字节的 CRC 校验码专门用于对地址字段进行校验。

地址场之后的间隙 GAP_2 为 22B 的 4EH，提供从读数据电路转换到写数据电路的切换时间。

数据场由同步字符（SYNC）、数据标志（AM_2）、数据字段（DATA）和 CRC 校验码组成。SYNC 的作用与地址场相同，数据标志表明后面数据字段的有效性。若检测到 AM_2 的最后一个字节为 FBH，则表明该扇区数据有效；若检测到 AM_2 的最后一个字节为 F8H，则表明该扇区数据是被删除的数据。DATA 字段存放数据字节，其长度与设置的磁道扇区数有关。CRC 校验码用于对数据字段的校验。

数据场之后是间隙 GAP_3，其作用是补偿由于主轴转速可能出现偏差而引起的扇区长度的变化。

（3）磁道尾部。磁道尾部为间隙 GAP_4，其作用是补偿由于主轴转速可能出现偏差而引起的总扇区段所占磁道长度的变化。

2．光盘存储器

光盘数据存储仍以扇区为单位，用于存储音频、文本、图形、图像等信号。每个扇区的物理格式如表 3-2 所列。

表 3-2　光盘扇区格式

	SYNC	Header	SubHeader	UserData	EDC	ECC
$FORM_1$	12B	4B	8B	2 048B	4B	276B
$FORM_2$	12B	4B	8B	2 324B	4B	

每个扇区为 2 352B，其中开始为 12B 同步码；后面为 4B 扇区头（以时、分、秒做地址的地址字节）；再往后是 8B 扇区子头，进一步说明该扇区的用户数据类型（音频、视频、文本等），格式形式（$FORM_1$、$FORM_2$），触发位，数据编码信息等；随后是用户数据；用户数据以后是 4B 的检错码。对于 $FORM_1$ 格式，用户数据为 2 048B，最后还有 276B 的纠错码；对于 $FORM_2$ 格式，用户数据为 2 324B，最后无纠错码。

3.2　半导体存储器接口

半导体存储器主要用于微机的内存。内存包含两部分：只读存储器 ROM 和随机存取存储器 RAM。其中，ROM 用于存放固化程序、表格、常数等，RAM 用于存放各种当前系统执行的程序，包括应用程序和当前处理的数据及结果。内存储器接口主要解决半导体存储器

芯片与 CPU 的连接问题。

3.2.1 半导体存储器接口的基本技术

存储器与 CPU 的连接可以从以下几个方面考虑。

1．信号线连接要求

用于内存的半导体存储器与 CPU 的连接，就是存储器芯片与 CPU 芯片的连接。就 CPU 而言，CPU 是通过地址总线 AB、数据总线 DB 以及控制总线 CB 与外部交换信息的，因此，存储器芯片与 CPU 的连接也就是与 CPU 3 种总线的连接。

（1）数据线的连接，一般系统的数据总线与存储器的数据线相连接；

（2）地址线的连接，通常系统的地址总线一部分用于选择存储器芯片而与存储器芯片的片选信号相连接，另一部分用于选择存储器芯片内的某一存储单元而与存储器芯片的地址线引脚相连接；

（3）控制线的连接，系统控制总线的读写控制需要与存储器芯片的控制线相连接。

2．地址分配要求

系统内存通常分为 ROM 和 RAM 两部分。其中 ROM 用于存放系统监控程序等固化程序及常数，RAM 可分为系统区和用户区两部分。系统区是监控程序或操作系统存放数据的区域，用户区又分为程序区和数据区两部分，分别用于存放用户程序和数据，所以，内存分配是一个重要的问题。就目前而言，单片的存储器芯片容量有限，计算机的内存储器系统需要由多个芯片组成，因此，针对存储器地址的分配，要知道哪些地址区域需要 ROM，哪些区域需要 RAM，即在具体电路中需要明确地址译码与片选信号的产生。以 Intel 8086CPU 为例，根据 Intel 8086 CPU 的特性，高地址区域应该是 ROM 区域，因为 Intel 8086CPU 复位以后执行的第一条指令一定在高地址区域；而低地址区域必须连接 RAM 芯片作为 RAM 系统区，因为低地址区域是 CPU 存放中断向量表等信息的系统区。

3．驱动能力考虑

在 CPU 的设计中，一般输出线的直流负载能力为带一个 TTL 负载，而在连接中，CPU 的每一根地址线或数据线，都有可能连接多片存储器芯片。所以，在存储器芯片与 CPU 的连接过程中，要考虑 CPU 外接有多少个存储器芯片以及 CPU 与存储器芯片的物理距离等因素。现在的存储器芯片都为 MOS 电路，直流负载都很小，主要的负载是电容负载，因此在小型系统中，CPU 可以与存储器芯片直接相连，在较大的系统中，就要考虑 CPU 是否需要加缓冲器，由缓冲器的输出再带负载。

3.2.2 静态 RAM 与 CPU 的连接

通常一片存储器芯片的存储容量不可能正好是 CPU 的内存寻址范围，所以一般都是由多片存储器芯片来构成计算机的内存储器系统。多片存储器芯片的组合，可能是为了满足 CPU 数据线宽度的需要，也可能是为了给 CPU 提供更大的存储空间。前者是对数据线的扩展，后者是对地址线的扩展。在实际应用中，常常需要这两方面同时进行扩展。

1．数据线上的扩展

图 3-11 所示为 1 024×1 的存储器芯片与 8 位 CPU 的连接。由于每片存储器芯片只有 1 位数据线，因此需要 8 片存储器芯片，才能满足 CPU 的 8 位数据线的要求，每一片存储器芯片连接 1 位数据线。

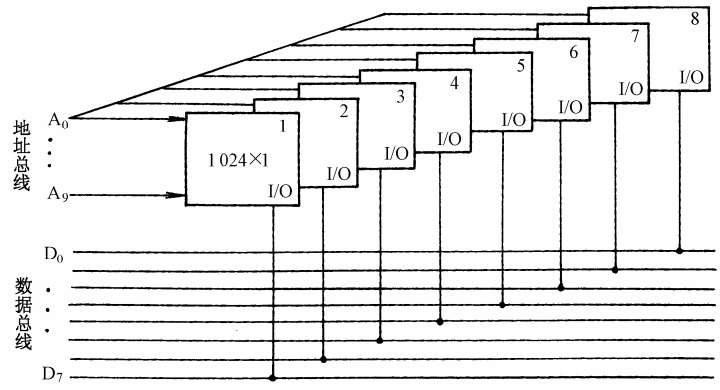


图 3-11　用 1 024×1 的存储器芯片组成的 1KB 的 RAM 电路

在这种结构中，每片存储器芯片除了数据线连至 CPU 不同位的数据线以外，地址线和控制线的连接都是相同的，因为 8 片芯片为 1 组，一旦被选中，则同时工作，或者输入，或者输出。

2．地址线上的扩展

图 3-12 所示是由两片 6264 静态 RAM 构成的 16KB 存储器。低位地址线 A_{12}～A_0 直接连至每一片 6264 芯片的地址输入端，高位地址线经译码以后产生片选信号，分别连接到两

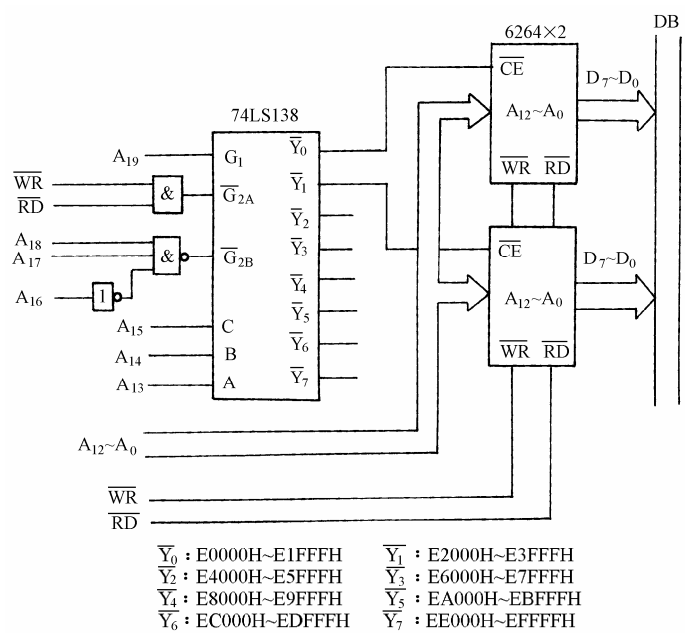


图 3-12　用 8KB×8 的存储器芯片组成的 16KB RAM 电路

片 6264 的片选输入端。地址译码器 74LS138 是一个常用的 3/8 译码器，当地址 A_{19}～A_{16}＝1110 时，该译码器被选中，也就是说，该译码器 Y_7～Y_0 输出的地址范围为 E0000H～EFFFFH。其中，当 A_{15}～A_{13}＝000 时，Y_0 输出有效，其地址范围为 E0000H～E1FFFH；当 A_{15}～A_{13}＝001 时，Y_1 输出有效，其地址范围为 E2000H～E3FFFH。

至于 6264 存储器芯片的数据线和控制线，则直接与 CPU 的数据线及对应的控制线相连接即可。

从图 3-12 可知，CPU 的地址线 A_{12}～A_0 负责了芯片内部存储单元的寻址，而不同存储芯片的选择，则是由地址线 A_{19}～A_{13} 来完成的。只有当 A_{19}～A_{13}＝1110000 时，选中第一片 RAM 芯片；当 A_{19}～A_{13}＝1110001 时，选中第二片 RAM 芯片。两者的区别在于地址线 A_{13}＝0 或 1。如果对上述电路做一简化，如图 3-13 所示，用 A_{13} 直接连接第一片芯片的片选输入端 $\overline{CE}$，用 A_{13} 经非门后接第二片芯片的 $\overline{CE}$ 端，结果又将如何？请读者自己分析。

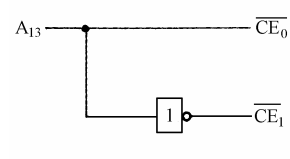


图 3-13　线选译码电路

当 CPU 访问的存储区域是在 E0000H～E1FFFH 地址范围内时，则第一片 RAM 芯片被选中；若访问的存储区域是 E2000H～E3FFFH，则第二片 RAM 芯片有效。这种只用一根地址线来实现片选的方法称为“线选”。显而易见，线选实现的电路简单，但缺点是出现了地址重叠。以第一片 RAM 芯片为例，不仅 E0000H～E1FFFH 是它的合法地址访问范围，E4000H～E5FFFH、E8000H～E9FFFH 以及更多的地址范围都对应这同一物理芯片。事实上，上述采用线选的方法，两片 RAM 芯片占用了 CPU 的全部地址资源。因此，除非是内存量小，无扩充需求的小系统，一般的计算机系统不采用线选方法来进行片选。

与线选对应的产生片选信号的方法称为全译码，如图 3-12 所示。全译码是所有的高位地址线都参加片选译码，因此尽管译码电路相对较复杂，但不会出现地址重叠，CPU 的地址与物理存储单元将一一对应。

如果是两片 RAM 芯片占据 CPU 不同的存储器地址区域，每片存储器芯片除了片选信号连接不同外，其他的控制信号以及地址线、数据线的连接都相同。

在控制信号的连接中必须注意：

（1）CPU 的写信号与 RAM 芯片的写信号相连，CPU 的读信号与 RAM 芯片的读信号或输出控制信号相连接。如果某些 RAM 芯片的读写控制信号是合为一根的信号线，则可根据其电平要求连接 CPU 的写信号线或读信号线。

（2）为了区别于 I/O 访问，保证只有在存储器读或写操作期间才能访问到存储器芯片，所以 CPU 与存储器的接口电路中要体现存储器访问这一操作性质，例如将 M/$\overline{IO}$ 信号参与控制用于片选的地址译码器，以保证只有在存储器访问期间地址译码器才工作。

3.2.3　动态 RAM 与 CPU 的连接

由于动态 RAM 的结构，特别是外部引脚与静态 RAM 有所不同，与 CPU 的连接要求也不相同。在与 CPU 相连构成动态 RAM 存储器系统时，有两方面必须考虑：

（1）动态 RAM 芯片的地址是分行、分列、分时输入的；

（2）动态 RAM 有刷新的要求。

因此，针对动态 RAM，除了在静态 RAM 连接中所要考虑的数据线的连接要求，地址线的译码要求等方面以外，还要考虑到如何提供地址以及在什么时候提供地址值。

与静态 RAM 的连接一样，CPU 输出的地址总线高位部分用于进行地址译码产生片选信号，地址总线的低位部分用于选择存储器内部的存储单元。但是，由于动态 RAM 的地址输入是分行、分列进行的，因此不能直接将 CPU 的低位地址线连至存储器的地址线输入，而是需要将这部分地址一分为二，按行、列分时输入存储器。

与此同时，由于动态 RAM 有刷新要求，既需要刷新控制信号，也需要为动态 RAM 提供刷新地址，因此，作为动态 RAM 的连接，还需要有一个产生刷新地址的电路，并通过选择电路，能在需要刷新的时候将刷新地址送入动态 RAM。

图 3-14 所示为动态 RAM 与 CPU 的接口电路。图中行地址由地址总线 A_7～A_0 提供，并且同刷新计数器的 7 位地址 RA_6～RA_0 通过一个多路开关（刷新多路器）送到“行/列多路器”，刷新控制信号控制刷新多路器只有在刷新操作时才选通刷新地址输出，否则输出行地址。列地址由 A_{15}～A_8 提供，且到行/列多路器。由多路控制信号控制多路器输出的是行地址或列地址送到存储器的地址线 A_7～A_0，由选通信号 $\overline{RAS}$ 和 $\overline{CAS}$ 选通送到行地址锁存器和列地址锁存器，在 $\overline{WE}$ 信号的作用下对所寻址的存储单元实现读操作或写操作。

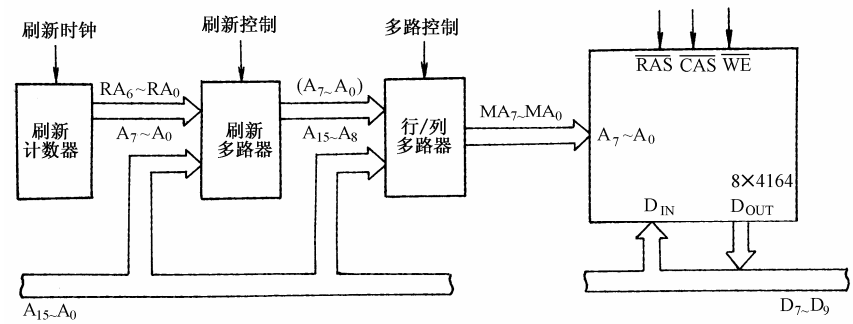


图 3-14　动态 RAM 与 CPU 的接口

PC/XT 的系统板上有 256KB 的 RAM，采用 64KB×1 位的芯片，则共有 4 组 36 个芯片，有关 RAM 的电路如图 3-15 所示。

PC/XT 中的 RAM 共有 4 组，每组 9 片芯片。所有存储器芯片的地址相连，然后连至行/列地址多路器的输出。每一组芯片的 $\overline{RAS}$ 和 $\overline{CAS}$ 输入端相连，分别接至相应的行和列地址选通信号。每个芯片的数据输入端（D_{IN}）和数据输出端（D_{OUT}）相连，然后每列的数据端相连，接到数据总线 MD_7～MD_0 上。每组中的第 9 个芯片存放奇偶校验位，作为奇偶校验使用。

2 选 1 多路开关 U_{39}、U_{40} 和 U_{41} 用做行/列地址切换。系统地址总线接到它们的输入端，选择控制信号端 S 接地址选择信号 ADDRSEL，此信号为低电平时，输出行地址，在执行读写操作时，$\overline{RAS}$ 信号将行地址锁存至动态 RAM 内部的地址锁存器中，当 ADDRSEL 变为高电平时，则输出列地址，从而实现地址切换，由随后的 $\overline{CAS}$ 把列地址锁存到列地址锁存器中。

数据收发器 U_9 把存储器局部数据总线与系统总线相连，它的允许工作端 $\overline{G}$ 接至信号 $\overline{RAM\ ADDRSEL}$ 上。当系统板上的 RAM 工作时，$\overline{RAM\ ADDRSEL}$ 为低电平，数据收发器 U_9 工作，允许数据传送。数据传送方向控制端 DIR 接 $\overline{XMEMR}$ 信号，在写操作时，数据由局部总线送至系统总线；在读操作时，数据由系统总线送至局部总线。

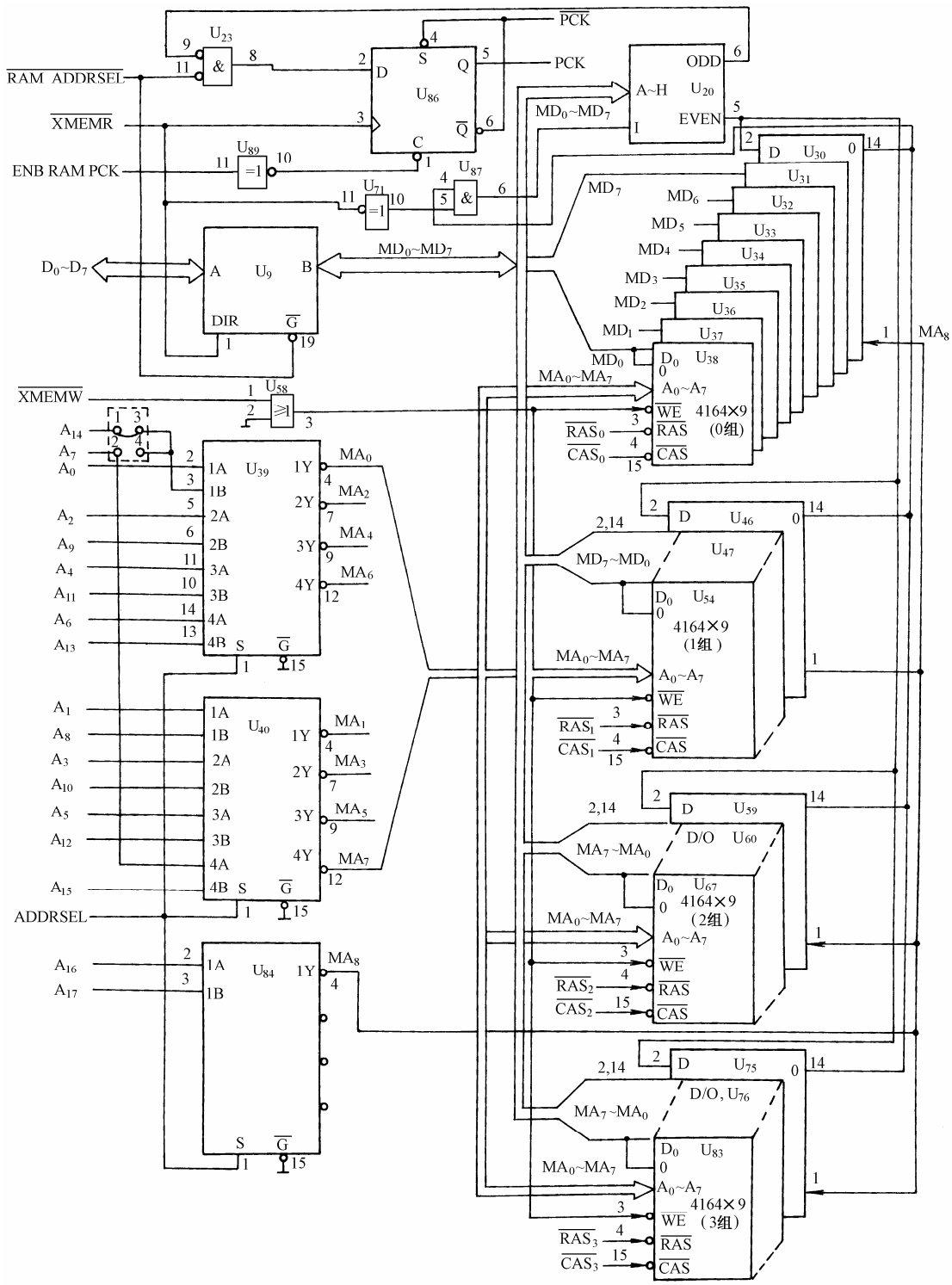


图 3-15　PC/XT 的 RAM 电路

在 PC/XT 的 RAM 电路中，控制信号 $\overline{\text{RAS}}$、$\overline{\text{CAS}}$ 以及 ADDRSEL 实现了对动态存储器的行、列地址的控制。三者的时序关系为：在执行存储器读总线周期或存储器写总线周期时，

首先行地址选通信号 $\overline{RAS}$ 有效，锁存行地址；隔 60ns 以后，ADDRSEL 由低电平变为高电平，使得 CPU 送到 RAM 芯片的地址由行地址切换为列地址；再隔 40ns 以后，$\overline{CAS}$ 有效，将列地址锁存入列地址锁存器。控制信号 $\overline{RAS}$、$\overline{CAS}$ 以及 ADDRSEL 由图 3-16 所示的地址译码电路产生。

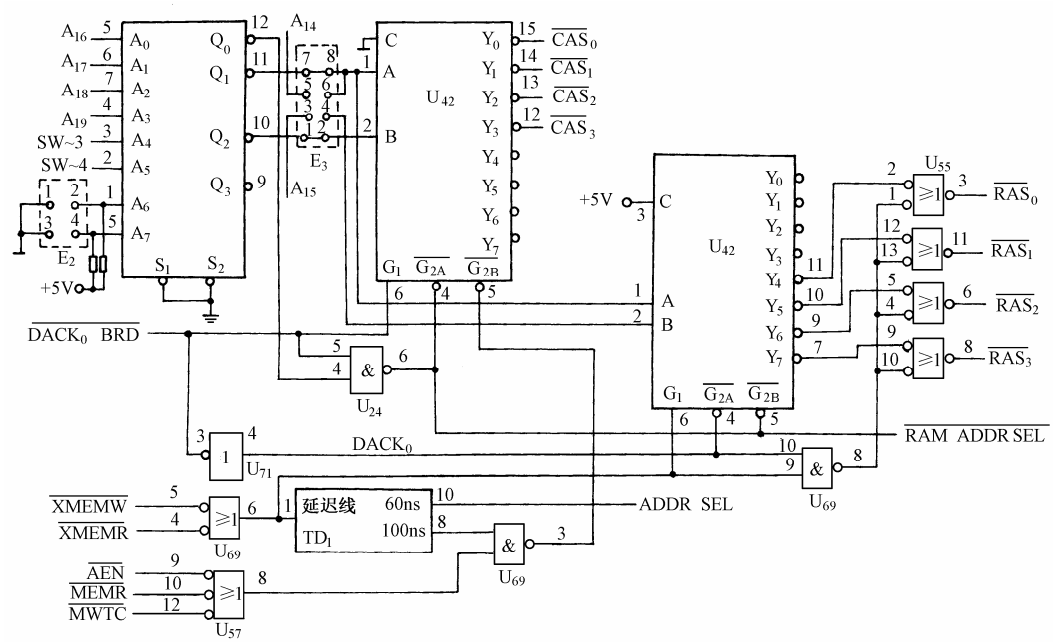


图 3-16　$\overline{RAS}$ 和 $\overline{CAS}$ 产生电路

从图 3-16 可知，通过定时电路 TD_1，$\overline{RAS}$、$\overline{CAS}$ 以及 ADDRSEL 可以依次有效。由 3/8 译码器产生的一系列行选通控制信号 $\overline{RAS}$ 和列选通控制信号 $\overline{CAS}$ 不仅体现了先后时序，同时也起到了存储器芯片的片选作用。

作为动态存储器的刷新操作，在 PC/XT 中采用 DMA 虚读方式，由 DMA 控制器产生刷新地址和控制信号。在存储器刷新期间（$DACK_0$ BRD 为低电平），所有的 $\overline{RAS}$ 输出都有效，刷新地址通过地址总线送到 RAM 芯片的地址线上，对所有的存储器进行刷新。

3.2.4　ROM 存储器与 CPU 的连接

由于常用的 EPROM 芯片的外部引脚与静态 RAM 相似，所以 CPU 与 EPROM 的连接也与静态 RAM 的连接相似，数据线直接相连，CPU 的地址线高位部分用于片选，低位部分连至 EPROM 的地址线输入端。对于控制线，由于 CPU 对 EPROM 不存在写操作，所以只需要考虑存储器读控制信号。针对 EPROM，要注意编程控制信号和编程电压的引脚输入，如 EPROM 27128 与 CPU 的连接，通常编程控制信号 $\overline{PGM}$ 输入高电平无效，V_{PP} 接+5V 电源。

显然，计算机系统在合闸上电后就能自动启动，必须把初始化程序和引导程序放在 ROM 中。IBM PC/XT 一般在系统板上安装了 40KB 的 ROM，它们占据了存储器的最高地址端 F6000H～FFFFFH，用于存放 BIOS 和 ROM BASIC。其中 BIOS 的功能就是对系统进行初始化，并为硬件功能调用提供接口。

图 3-17 所示为 PC/XT 的 ROM 系统控制电路图，系统板上 40KB 的 ROM 信息存放在两

片 ROM 芯片中，一片是 8KB 的芯片，一片是 32KB 的芯片。

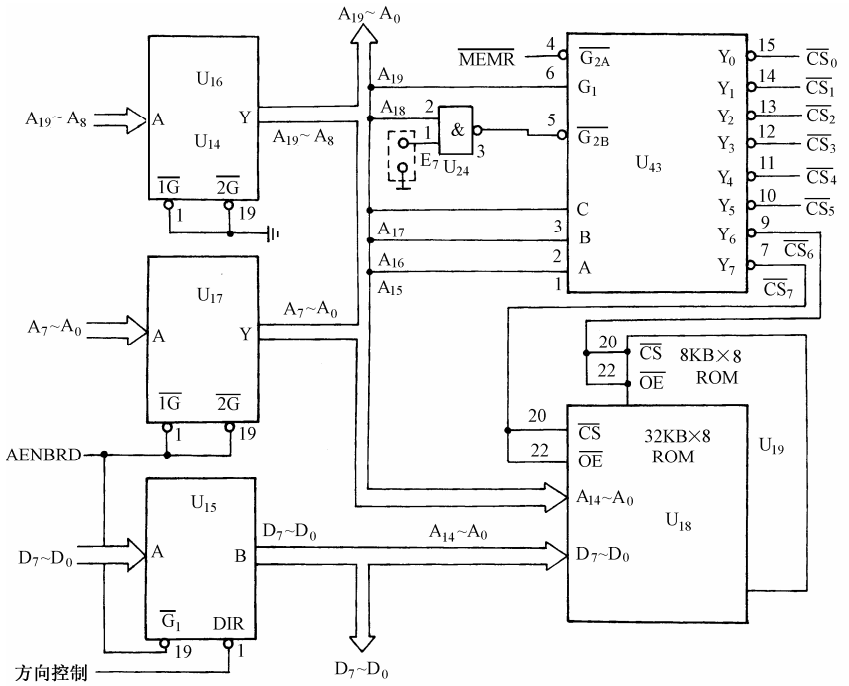


图 3-17　IBM PC/XT 的 ROM 子系统

系统的地址总线 A_{19}～A_0，经过 U_{17}、U_{16}、U_{14} 缓冲，形成 ROM 子系统中的地址总线，其中 A_{14}～A_0（32KB 地址）直接与 ROM 芯片的地址线 A_{14}～A_0 相连（8KB ROM 芯片地址线为 A_{12}～A_0），高位地址线与控制信号一起作为 ROM 芯片的片选信号。系统数据总线 D_7～D_0 经过 U_{15} 的缓冲，直接与两片 ROM 芯片的 D_7～D_0 相连。

片选信号由 3/8 译码器 74LS138（U_{43}）产生，它的 3 个允许工作控制端 G_1 直接连至 A_{19}，$\overline{G_{2A}}$ 直接连至系统的存储器读控制信号 $\overline{MEMR}$，$\overline{G_{2B}}$ 同与非门 U_{24} 相连，U_{24} 的两个输入端一个是 A_{18}，另一个是跨接线 E_7，通常情况下 E_7 断开，U_{24} 的输出就是 A_{18} 的反相信号。因此，U_{43} 能正常工作的条件是在存储器读周期，且 A_{19}＝1、A_{18}＝1。若跨接线 E_7 接地，则 U_{24} 的输出为高电平，禁止 U_{43} 工作，这时就禁止系统板上的基本 ROM 工作，用户可自己编写 BIOS，插入 I/O 通道工作。

U_{43} 的译码输入端 A、B、C 连至地址总线 A_{15}、A_{16}、A_{17}，译码输出 $\overline{CS_6}$ 的地址范围是 F0000H～F7FFFH，连至 8KB ROM 的 $\overline{CS}$ 和 $\overline{OE}$ 端，$\overline{CS_7}$ 的地址范围是 F8000H～FFFFFH，连至 32KB ROM 的 $\overline{CS}$ 和 $\overline{OE}$ 端。

3.2.5　16/32 位存储器的数据组织

由于 X86 微处理器在与主存储器的连接中，都会考虑到兼容问题。对于 16 位的微处理器不仅能完成双字节存取，还要满足单字节的访问；对于 32 位微处理器，也要能够具有单字节、双字节的数据访问功能。因此，X86 系统都是采用字节编址的存储器结构，微处理器不仅可以按字访问，也可以按字节访问。

1．16 位存储器组织

8086 和 80286 微处理器都是 16 位的微处理器，其主存储器采用按存储器分体的组成方法，即将存储器分成偶数地址的存储体和奇数地址的存储体，偶数地址存储体的数据线与数据总线 D_0～D_7 连接，而奇数地址存储体的数据线与数据总线 D_8～D_{15} 相连。偶数地址存储体和奇数地址存储体用 CPU 的 $\overline{BHE}$ 信号与地址线 A_0 区分。$\overline{BHE}$ 与地址线 A_0 的组合所对应的操作如表 3-3 所示。因此，通常用 A_0 作为偶数地址存储体的选通信号，$\overline{BHE}$ 作为奇数地址存储体的选通信号。这样，当 A_0＝0、$\overline{BHE}$ ＝1 时，只有偶数地址存储体工作，即低字节访问；当 A_0＝1、$\overline{BHE}$ ＝0 时，则奇数地址存储体工作，高字节访问；当 A_0＝0、$\overline{BHE}$ ＝0 时，奇偶数地址存储体同时工作，进行双字节访问。图 3-18 为存储器结构图。

表 3-3　$\overline{BHE}$ 和 A_0 的代码组合和对应的操作

$\overline{BHE}$	A_0	操　作	数 据 线
0	0	从偶地址开始读/写一个字	D_{15}～D_0
1	0	从偶地址单元读/写一个字节	D_7～D_0
0	1	从奇地址单元读/写一个字节	D_{15}～D_8

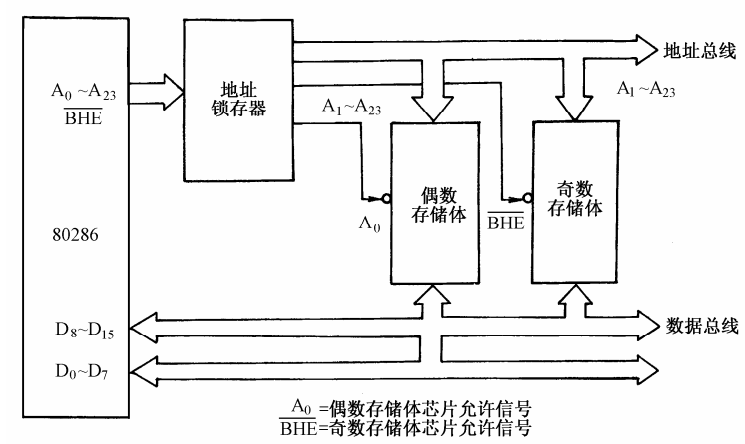


图 3-18　80286 存储器结构

2．32 位存储器组织

80386/80486 微处理器是 32 位微处理器，为了实现 8 位、16 位、32 位数据的访问，处理器设有 4 个引脚 $\overline{BE_3}$ ～ $\overline{BE_0}$，以控制不同的数据访问。$\overline{BE_3}$ ～ $\overline{BE_0}$ 由 CPU 根据指令类型产生，$\overline{BE_3}$ ～ $\overline{BE_0}$ 的作用如表 3-4 所示。

表 3-4　$\overline{BE_3}$ ～ $\overline{BE_0}$ 功能表

字 节 允 许				要访问的数据位			
$\overline{BE_3}$	$\overline{BE_2}$	$\overline{BE_1}$	$\overline{BE_0}$	D_{31}～D_{24}	D_{23}～D_{16}	D_{15}～D_8	D_7～D_0
1	1	1	0	—	—	—	D_7－D_0
1	1	0	1	—	—	D_{15}～D_8	—
1	0	1	1	—	D_{23}～D_{16}	—	—
0	1	1	1	D_{31}～D_{24}	—	—	—

续表

字 节 允 许				要访问的数据位			
1	1	0	0	—	—	D_{15}～D_8	D_7～D_0
1	0	0	1	—	D_{23}～D_{16}	D_{15}～D_8	—
0	0	1	1	D_{31}～D_{24}	D_{23}～D_{16}	—	—
1	0	0	0	—	D_{23}～D_{16}	D_{15}～D_8	D_7～D_0
0	0	0	1	D_{31}～D_{24}	D_{23}～D_{16}	D_{15}～D_8	—
0	0	0	0	D_{31}～D_{24}	D_{23}～D_{16}	D_{15}～D_8	D_7～D_0

80386/80486 微处理器设有 32 位地址，但直接输出 A_{31}～A_2，字节允许信号 $\overline{BE_3}$ ～$\overline{BE_0}$，实现了最低两位地址线的寻址功能，以选择不同字节。这样，在进行存储器系统设计时常把主存储器分为 4 个存储体，依次存放 32 位数据的不同字节，每个存储体的 8 位数据线依次并联到外部数据线 D_{31}～D_0 上。如图 3-19 所示为一个 32 位存储器系统，每个存储体的 15 位地址 A_{14}～A_0 接 CPU 的地址线 A_{16}～A_2，片选信号 $\overline{CE}$ 由高位地址的译码结果和 $\overline{BE_3}$ ～$\overline{BE_0}$ 相“与”后产生。一旦地址确定，A_{18}～A_2 确定 4 个存储体中的相同地址单元，$\overline{BE_3}$ ～$\overline{BE_0}$ 决定某一个或某几个字节单元选中，然后可对选中单元同时进行读/写操作。

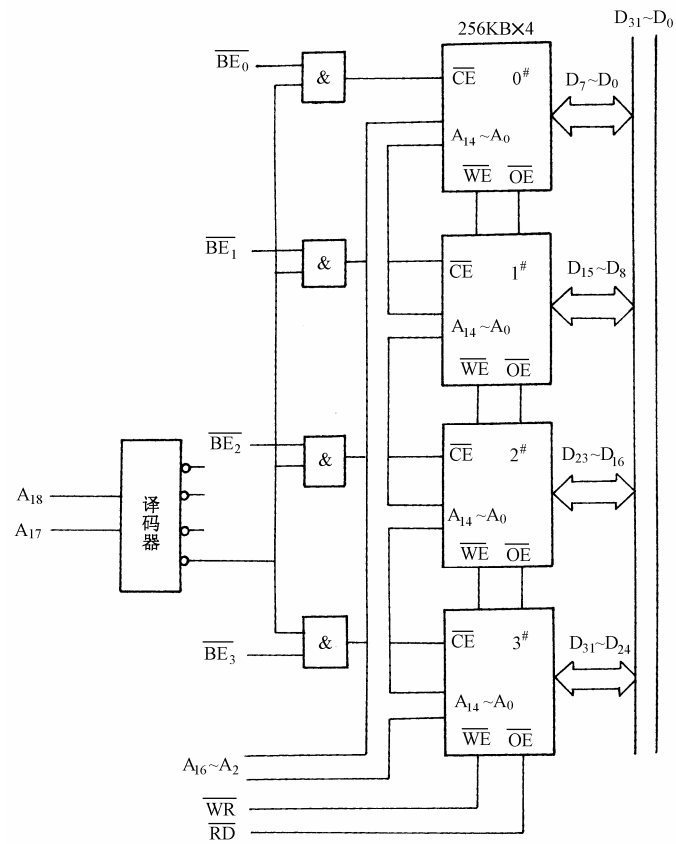


图 3-19　32 位字存储器系统组成

3．单列直插存储器 SIMM

SIMM（Single Inline Memory Module）是把存储器芯片焊接在条形印刷电路板上制成的，

俗称内存条。其容量有 256KB、512KB、1MB、4MB、8MB、16MB 甚至更大，单元位数有 8 位和 9 位两种，其中 9 位中有一位是奇偶校验位。目前，按印刷电路的连线可分为 30 线和 72 线两种。30 线内存条比较小，提供 8 位有效数据；72 线内存条体积比较大，提供 32 位有效数据。随着 Pentium 微处理器的普及，为了便于存储 32 位或 64 位数据，72 线内存条已成为主流。

（1）30 线 SIMM。30 线内存条提供 8 位或 9 位数据，在主机板上有插座。由 4 块内存条构成一组，可提供 32 位数据。30 位 SIMM 的引脚信号如表 3-5 所示。

表 3-5　30 线 SIMM 引脚信号

引　脚	信号名称	引　脚	信号名称	引　脚	信号名称	引　脚	信号名称
1	V_{SS}	9	V_{CC}	17	S_8	25	DQ_7
2	$\overline{CAS}$	10	DQ_2	18	NC（A_9）	26	Q_8
3	DQ_0	11	A_4	19	NC（A_{10}）	27	$\overline{RAS}$
4	A_0	12	A_5	20	DQ_5	28	$\overline{CAS}$
5	A_1	13	DQ_3	21	$\overline{WE}$	29	D_8
6	DQ_1	14	A_6	22	V_{CC}	30	V_{SS}
7	A_2	15	A_7	23	DQ_6		
8	A_3	16	DQ_4	24	NC		

如图 3-20 所示是由 256KB×4 位芯片构成的 256KB 内存条，或是由 1MB×4 位芯片构成的 1MB 内条。图中 D_8 是奇偶校验位输入，Q_8 是奇偶校验位输出。

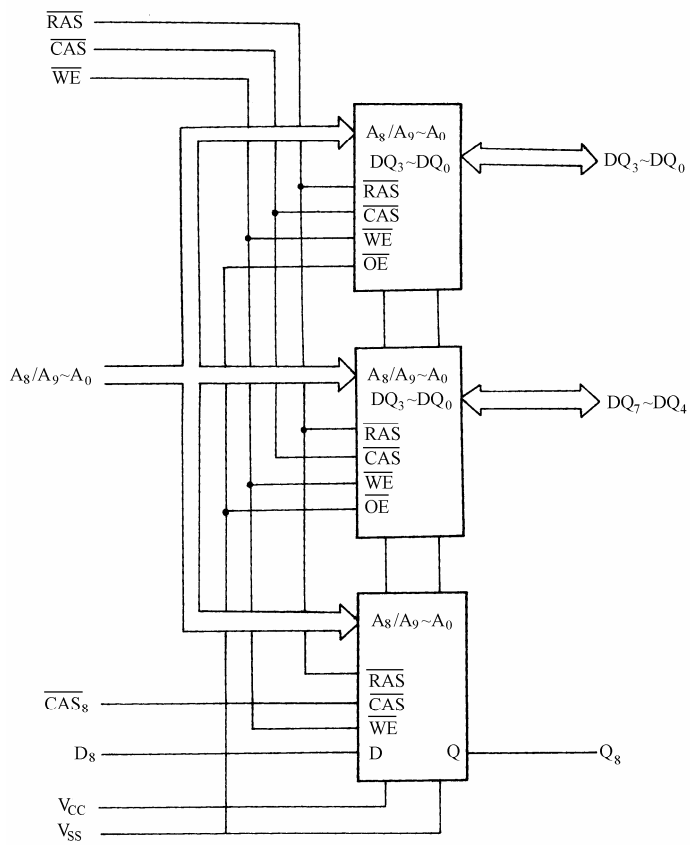


图 3-20　由 256KB（或 1MB）×4 位芯片构成的 8 位内存条

（2）72 线 SIMM。如图 3-21 所示为 72 线内存条，提供 32 位数据，在主机板上有其插座。若用两块构成一组，可提供 64 位数据。72SIMM 的引脚信号如表 3-6 所示。

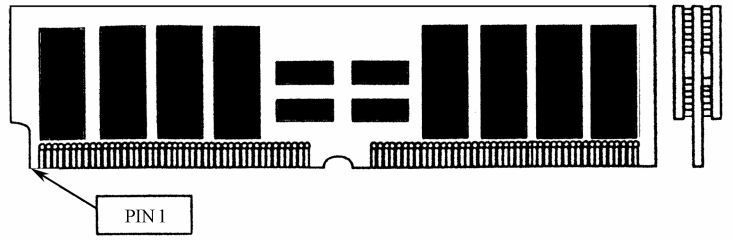


图 3-21　典型的 72 线 SIMM 结构图

表 3-6　72 线 SIMM 引脚信号

引　脚	信号名称	引　脚	信号名称	引　脚	信号名称	引　脚	信号名称
1	V_{SS}	19	NC	37	DQ_{17}	55	DQ_{12}
2	DQ_0	20	DQ_4	38	DQ_{35}	56	DQ_{30}
3	DQ_{18}	21	DQ_{22}	39	V_{SS}	57	DQ_{13}
4	DQ_1	22	DQ_5	40	$\overline{CAS_0}$	58	DQ_{31}
5	DQ_{19}	23	DQ_{23}	41	$\overline{CAS_2}$	59	V_{CC}
6	DQ_2	24	DQ_6	42	$\overline{CAS_3}$	60	DQ_{32}
7	DQ_{20}	25	DQ_{24}	43	$\overline{CAS_1}$	61	DQ_{14}
8	DQ_3	26	DQ_7	44	$\overline{RAS_0}$	62	DQ_{33}
9	DQ_{21}	27	DQ_{25}	45	NC	63	DQ_{15}
10	V_{CC}	28	A_7	46	NC	64	DQ_{34}
11	NC	29	NC	47	WE	65	DQ_{16}
12	A_0	30	V_{CC}	48	NC	66	NC
13	A_1	31	A_8	49	DQ_9	67	PD_1
14	A_2	32	NC	50	DQ_{27}	68	PD_2
15	A_3	33	NC	51	DQ_{10}	69	PD_3
16	A_4	34	$\overline{RAS_2}$	52	DQ_{28}	70	PD_4
17	A_5	35	DQ_{26}	53	DQ_{11}	71	NC
18	A_6	36	DQ_8	54	DQ_{29}	72	V_{SS}

一般主机板的存储器安装分为几个体，每个体有 2～4 个存储器插座，可安装 2～4 个内存条。对于 16 位处理器，使用 30 线 SIMM 时安装数量应为偶数；对于 32 位微处理器，使用 30 线 SIMM 时安装数量应为 4 的倍数；对于 Pentium 微处理器，其数据线为 64 位，若要一次能存取 64 位数据，采用 72 线内存条也应按偶数安装。

4．DIMM 内存条

DIMM（Double Inline Memory Module）内存条是双列直插的内存模块，为 168 线，提供 64 位数据线，此外还加 8 位校验位。

与 SIMM 不同，DIMM 上模块两面的引脚是分开的。内存条两端相同，但引脚排列有两处缺口，且左右不对称，主板上的内存插槽相应有两处凸出，可以防止方向插反。DIMM 内存条主要由存储芯片、印刷电路板、SPD（Serial Presence Detect）小芯片及一些电阻、电容组成。印刷电路板是多层板，一般为六层板。SPD 是一个 256B 的 EEPROM，里面保存着内存条的有关数据（如一些设置、模块周期信息等）。如果 PC 机主机是支持 SPD 的，开机后

就可以在 BIOS 的 DRAM 设定选项中看到 SPD 的设置。

DIMM 内存条容量有若干档次，早期多为 16MB、32MB 和 64MB，现在标准定位为 128MB 和 256MB。

3.2.6 存储器芯片连接中的时间估算

CPU 在执行取指令或是存储器读写操作时，是有固定时序的，因此需要考虑 CPU 与存储器芯片的速度匹配问题。也就是说，在 CPU 主频确定的情况下，如果 CPU 的读写存储器总线周期固定，就需选择合适的存储器芯片，以便在存储器访问总线周期中一定能完成存储器的读写。否则，CPU 需要插入一个或多个等待周期，以便 CPU 能正确实现存储器读写操作。

以 CPU 的存储器读操作为例，能够正确完成存储器读操作，对存储器芯片而言，必须满足两个时间参数：第一，有足够的地址译码时间，即存储器芯片接收到地址信息以后，有足够的时间选中相应的存储单元；第二，存储器芯片有足够的时间能将存储单元中的数据送到外部数据线，也就是存储器芯片从读控制信号有效到存储器有效数据输出之间的时间。这两个参数是存储器芯片的重要速度指标。

对于某一具体存储器芯片，图 3-22 所示为芯片读操作时的时序，其中 t_A 为读取时间，即上述第一条件规定的时间，t_{CO} 为输出控制有效到数据输出稳定时间，即第二条件规定的时间。

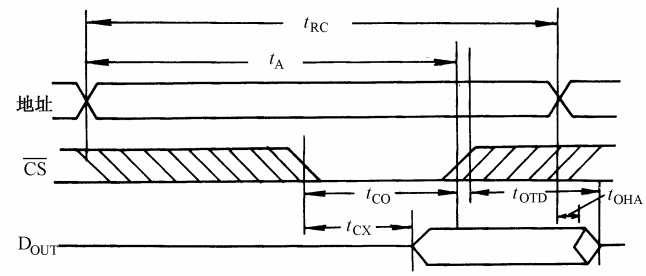


图 3-22 存储器芯片读操作

当 CPU 的主频确定以后，其时钟周期也已确定。若 CPU 的存储器读总线周期由 4 个时钟周期组成：CPU 在 T_1 输出地址有效，在 T_2 产生存储器读信号，CPU 将在 T_3 和 T_4 之间读取数据总线上的数据。也就是说，要求存储器在 T_3 和 T_4 之间已经向数据总线提供了有效的数据值。CPU 在 T_1 输出地址有效，从地址有效到 CPU 读数据总线有 2 个多时钟周期的时间；在 T_2 产生存储器读信号，从读控制信号有效到 CPU 读数据操作有 1 个多时钟周期的时间。只有这两个时间都比存储器芯片的地址译码时间参数和数据输出时间参数宽裕时，CPU 才能正确访问。只要两个参数中有一个不满足，就需要在 T_3 以后插入等待周期 T_W。

3.3 盘存储器接口

盘存储器是目前微型计算机最常用的外存储器，主要有磁盘存储器和光盘存储器。

3.3.1 软磁盘驱动器接口

从表面上看，软磁盘可以自由地插入软盘驱器以及取出，而硬磁盘与盘驱动器是合二为一的。软磁盘与硬磁盘在性能上有很大差异，但两者的读写原理相同，无论是软盘还是硬盘，都是由磁盘控制器通过磁盘驱动器完成对磁盘的读写操作的。

以软磁盘为例，软盘驱动器是精密的机电设备，其启、停及运行等都需要有控制信号对其进行严格的控制。由于软盘驱动器进行读取操作的数据传输速率远远低于 CPU 的工作速度，不可能由 CPU 直接控制，这就需要通过接口来完成对驱动器的控制。这个接口就是软驱控制器。软驱控制器主要由软盘控制器和一些辅助电路构成，其核心就是软盘控制器 FDC（Floppy Disk Controller）。

1．FDC 的主要功能

FDC 的主要功能有以下几个方面：接收来自主机的有关命令，检测 FDD（Floppy Disk Drive）的有关状态，对读、写的数据进行缓冲及处理。

写入软盘的数据来自内存，从软盘读出的数据也送到内存中，然后才能被 CPU 所使用，在盘与内存之间通过 DMA 方式传送数据。盘的读写操作首先是 CPU 对 DMA 控制器规定操作数据所在的内存位置、数量以及传送方向（读或写），然后 CPU 向 FDC 发命令。FDC 接收命令以后，经过命令识别和译码后，向 FDD 发出相应的读写控制和磁头的选择及控制工作，在完成寻道和扇区地址检测工作以后，就开始读盘或写盘。

FDC 检测 FDD 的有关状态，如盘写保护状态，并将检测结果发送给 CPU。一般 CPU 在发读写命令之前，要先检查 FDD 的有关状态，以便根据状态发出相应的命令。

FDC 对数据进行缓冲及处理功能包括数据串行与并行的转换、数据写入时的补偿以及数据读出时的锁相分离。

2．FDC 接口

无论何种型号的 FDD，它与软驱适配器的接口都已标准化。目前的微型计算机，其接口电路都集成在主板上，采用 34 芯扁平电缆与软驱连接。在 34 根线中，奇数号线均为地线，其余各线如图 3-23 所示。

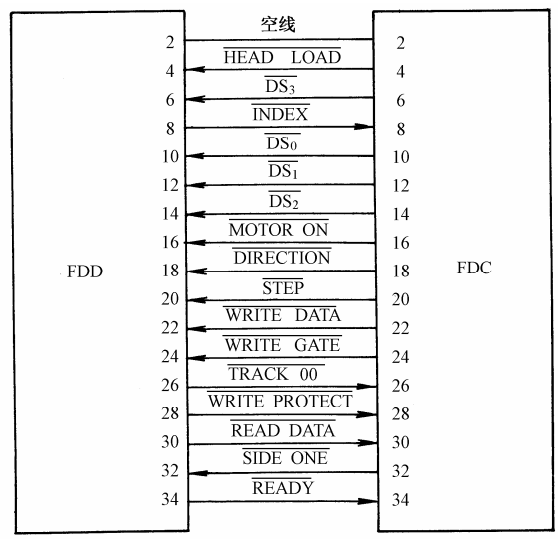


图 3-23　FDC 与 FDD 接口信号示意图

$\overline{DS_0}$ ～ $\overline{DS_3}$ ：驱动器选择信号。FDC 最多可带 4 台软驱，输出低电平为某一驱动器选中信号。

$\overline{MOTOR\ ON}$ ：电机启动信号。其为低电平时，被选中的当前驱动器主轴电机启动，并

加速到额定转速；当此信号为高电平时，主轴电机停转。

$\overline{\text{DIRECTION}}$：方向信号。用来控制磁头步进的方向，低电平时向内磁道方向步进，高电平时向外磁道方向步进。

$\overline{\text{STEP}}$：步进信号。以负脉冲作为输入信号，每出现一个负脉冲，磁头步进一个磁道。

$\overline{\text{SIDE ONE}}$：磁头选择信号。输出低电平或高电平，分别表示选择 1 磁道和 0 磁道。

$\overline{\text{WRITE GATE}}$：写选通信号。低电平有效时表示允许写。

$\overline{\text{WRITE DATA}}$：写数据信号。用于输出经 FDC 编码后的数据写入脉冲序列。

$\overline{\text{READ DATA}}$：读数据信号。读取磁盘上记录的序列脉冲，并输出到软盘适配器。

$\overline{\text{INDEX}}$：索引信号。用于标记磁道的起始位置。

$\overline{\text{TRACK 00}}$：零磁道信号。用于标记 0 磁道的位置。

通常在 PC 机中可以配置为由一个 FDC 控制两个 FDD，通过 34 芯扁平电缆与两个 FDD 相连接。三个连接插头，一端接 FDC，另两端分别接两个软驱 A 和 B，其连接如图 3-24 所示。

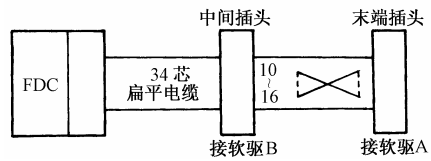


图 3-24　FDD 与 FDC 的连接示意图

为简化连接操作，规定接软驱 A 与 B 的差别在于：接驱动器 A 时，第 10～16 号这 7 根线扭转相接。

3.3.2　IDE 接口

IDE（Intelligent Drive Electronics）即智能驱动器电子接口，是个人计算机用来连接硬盘、光盘等设备的标准接口。目前计算机连接硬盘的接口有两类：IDE 接口 和 SCSI 接口。一般家用计算机和台式计算机上都采用 IDE 接口，而 SCSI 接口则多用于服务器或一些要求传输速度快的场合。

1．IDE 接口的特点

IDE 由 COMPAQ 公司推出，其最大特点是将原来接口中的控制器部分，包括控制器、缓冲器管理和错误校验器、缓冲器管理控制处理器、高速缓冲存储器等都集成到硬盘驱动器中去了，这样，在 IDE 接口的适配器电路中，不包含硬件控制器，其优点是可以消除驱动器和控制器之间的数据丢失问题，使数据传输十分可靠。1991 年美国 ANSI（American National Standards）正式把 IDE 接口命名为 ATA（AT Attachment）接口，但一般还是习惯于沿用“IDE 接口”的名称。

IDE 接口当初是针对 AT 系统设计的，支持超大容量高速硬盘。一个 IDE 接口可带两个硬盘驱动器，它用一条 40 线的单组电缆来转接信号。在硬盘连接之前，必须设置其工作模式，IDE 硬盘有 3 种工作模式：Spare（单机）、Master（主）和 Slave（从）。Spare 是指一条单排电缆只接一台硬盘机的状态，而接两块硬盘时，所接硬盘有两种工作状态，所接第一块为 Master 模式，第二块为 Slave 模式。

IDE 接口的硬盘机的基本特征是：

（1）最多接两台硬盘机；

（2）数据传输速率不超过 2MB/s；

（3）只能支持 528MB 的最大容量；

（4）最大传输带宽只有 8 位；

（5）只能轮流操作，不能并行处理。

硬盘机是 PC 机中发展最快的部件之一，存储容量和传输速度是硬盘机各项性能指标中提升最快的两项。这样，传统的 IDE 接口就远远不能满足硬盘机发展的要求。1993 年，硬盘机专业公司 Western Digital 开始着手制定扩充 IDE 规范，1994 年正式公布 EIDE 规范，其主要特征是：

（1）数据传输速率至少可达 12MB/s～18MB/s；

（2）支持硬盘的最大容量可达 8.4GB；

（3）可连接 4 台满足 EIDE 标准的外部设备；

（4）传输带宽为 16 位，可扩展到 32 位；

（5）EIDE 在内容及硬盘读写操作上可并行处理。

2．数据传送方式

IDE 接口的硬盘机与主机进行数据传送有两种方式：PIO 模式和 DMA 模式。

PIO（Programming Input/Output）模式通过 CPU 执行 I/O 端口指令来进行数据的读写，一般采用 I/O 串操作指令，以达到较高的数据传输速率。

在 DMA（Direct Memory Access）模式下，数据不经过 CPU 而直接在硬盘和内存之间传送，由于不需要 CPU 的参与，可获得高的数据传送速率。

3．IDE 接口信号

IDE 的接口信号如表 3-7 所示，共 40 条。除了对 AT 总线上的信号做必要的控制外，信号基本上原封不动地送往硬盘驱动器。

表 3-7　IDE 接口信号

引脚号	说明	引脚号	说明
01	复位（$\overline{\text{RESET}}$）	02	地（Ground）
03	数据信号线 7（DD_7）	04	数据信号线 8（DD_8）
05	数据信号线 6（DD_6）	06	数据信号线 9（DD_9）
07	数据信号线 5（DD_5）	08	数据信号线 10（DD_{10}）
09	数据信号线 4（DD_4）	10	数据信号线 11（DD_{11}）
11	数据信号线 3（DD_3）	12	数据信号线 12（DD_{12}）
13	数据信号线 2（DD_2）	14	数据信号线 13（DD_{13}）
15	数据信号线 1（DD_1）	16	数据信号线 14（DD_{14}）
17	数据信号线 0（DD_0）	18	数据信号线 15（DD_{15}）
19	地（Ground）	20	接口识别（key）
21	DMA 请求（DMARQ 请求）	22	地（Ground）
23	写信号（$\overline{\text{IOW}}$）	24	地（Ground）
25	读信号（$\overline{\text{IOR}}$）	26	地（Ground）
27	I/O 通道准备好（I/O : CH RDY）	28	同步信号（SPAYNC）
29	DMA 应答（DMACK）	30	地（Ground）

续表

引 脚 号	说 明	引 脚 号	说 明
31	中断请求信号 (INTRQ)	32	总线宽度识别 ($\overline{IOCS_{16}}$)
33	地址信号线 1 (DA_1)	34	诊断完成 (DPIAG)
35	地址信号线 0 (DA_0)	36	地址信号线 2 (DA_2)
37	片选 0 ($\overline{CS_1FX}$)	38	片选 1 ($\overline{CS_3FX}$)
39	驱动器激活 ($\overline{DASP}$)	40	地 (Ground)

各信号线功能如下：

DD_0～DD_{15}：双向数据线，直接取自系统数据线，用于传送数据、命令和状态信息。

DA_0～DA_2：地址线，直接取自系统地址线，用于选择控制器内部命令或控制寄存器。

$\overline{CS_1FX}$ ：命令寄存器组选择信号，由连接在 ISA 总线上的 IDE 适配器对系统地址译码后产生。

$\overline{CS_3FX}$ ：驱动器工作指示信号。

$\overline{DASP}$ ：驱动器工作指示信号。

$\overline{IOCS_{16}}$ ：16 位数据传送信号。

DPIAG：论断完成信号，表示主、从驱动器已完成自检。

$\overline{RESET}$ ：驱动器复位信号。

CSEL：主驱动器选择信号。

INTRQ：中断请求信号。

IORDY：通道就绪信号。

$\overline{DIOR}$ 和 $\overline{DIOW}$ ：驱动内部寄存器读写控制信号。

DMARQ 和 DAMCK：DMA 传送控制信号。

SPSYNC：主从驱动器同步信号。

3.3.3 SCSI 接口

随着个人计算机的快速发展，主机和外设的运行速度都有了极大的提高，而负责主机和外设间数据传输的接口技术的发展却远没跟上，它的数据传输率和可靠性已不能满足高速主机和高速外设间的通信要求，已成为系统数据传输的瓶颈。

要解决微型计算机系统数据传输的瓶颈问题，只有提高 I/O 接口的传输率和可靠性。方法之一就是在微型机系统上引入 SCSI 接口技术。接入 SCSI 接口的设备可以并行处理，占用 CPU 时间少，速度比 IDE 接口快，能接的设备也比 IDE 接口多。

1．SCSI 接口的标准分类

SCSI 接口是小型计算机系统接口（Small Computer System Interface）的缩写，它是在美国 Shugart 公司开发的 SASI 的基础上，增加了磁盘管理功能而成的。SCSI 接口作为输入输出接口，主要用于光盘机、磁带机、扫描仪、打印机等设备。

大部分 SCSI 控制器是以接口卡形式连接到 PC 主板上的，SCSI 控制器接口卡可连接 ISA、EISA、VESA 和 PCI 总线。仅个别高档主板在板上集成了 SCSI 控制器。

SCSI 标准是 1986 年审议完成的，称为 SCSI-1 标准，SCSI-1 是早期标准，数据传输率仅为 4MB/s，现已淘汰。1990 年又制定了 SCSI-2 标准，SCSI-2 分 Fast SCSI 和 Fast Wide SCSI。

SCSI-3 分 Ultra SCSI、Wide Ultra SCSI、Ultra-2 SCSI 和 Wide Ultra-2 SCSI，它们的速度都不相同。

（1）Fast SCSI 为 8 位总线，数据传输率为 10 MB/s。

（2）Fast Wide SCSI 采用 16 位数据总线，数据传输率为 20 MB/s。

（3）Ultra SCSI 为 8 位总线，数据传输率为 20 MB/s；

（4）Wide Ultra SCSI 采用 16 位总线，数据传输率为 40 MB/s。

（5）Ultra-2 SCSI 为 8 位总线，数据传输率为 40 MB/s;

（6）Wide Ultra-2 SCSI 为 16 位总线，数据传输率为 80 MB/s。

1998 年，7 家电脑系统和存储器产品的厂家共同宣布支持全新的 Ultra 160 MB/s 的 SCSI 接口。Ultra 160 MB SCSI 共有三大特点：

（1）双倍传输时钟（Double Transation Clocking），使传输速率由 80 MB/s 提升到 160 MB/s，比原先快了 1 倍；

（2）能够针对接口的效率进行自动侦测，具有主题区域确认（Domain Validation）功能，提高了管理能力；

（3）具有周期性循环冗余码校验（CRC）功能，增加了可靠性。

2．SISC 信号定义

SCSI 总线可以是 8 位、16 位和 32 位，若是 8 位总线，设备之间电缆为 50 芯扁平电缆，称为电缆 A。16 位和 32 位总线为“宽 SCSI”总线，属于 SCSI-2 标准，需要一根附加电缆，附加电缆为 68 芯电缆，称为电缆 B。这 68 条信号线称为 SCSI-2 的扩充信号。图 3-25 为 SCSI 设备连接图，其中启动设备为计算机的 SCSI 主适配器，目标设备是磁盘驱动器。

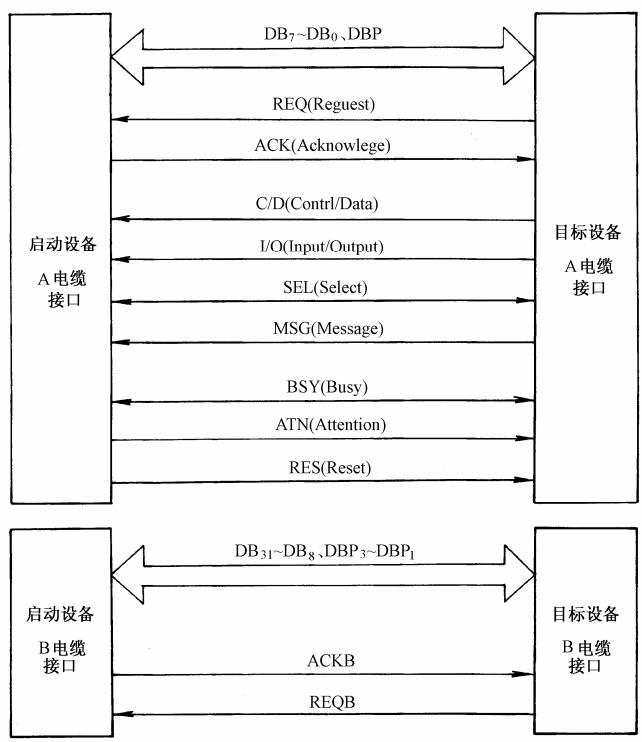


图 3-25　SCSI 设备连接

各信号线功能如下：

DB_7～DB_0：低 8 位数据线。

DBP：奇偶校验信号线。

REQ：目标设备向启动设备发出请求传输信息信号。

ACK：启动设备向目标设备发出的请求传输信息的响应信号。

C/D：由目标设备使用，表示总线上数据类型，命令或数据。

I/O：由目标设备使用，表示数据的方向，I/O=1 时表示启动设备接收数据。

SEL：线选信号，在选择和重选阶段使用。

MSG：在信息阶段由目标设备使用。

BSY：线选信号，表示总线正在被使用。

ATN：启动设备用该信号表示启动设备有信息需要传送。

RES：线选信号，使总线上的设备复位。

DB_{31}～DB_8：高 24 位数据线。

DBP_3～DBP_1：3 位奇/偶校验信号线。

ACKB：B 电缆中启动设备向目标设备发出的请求传输信息的响应信号。

REQB：B 电缆中目标设备向启动设备发出请求传输信息信号。

本 章 小 结

本章要求掌握的内容是存储器及接口。首先，作为计算机的存储器，有多种分类方法，通过不同类型的存储器，可以了解其不同的特性及其适用范围，以及对存储器性能的评价方法；其次，作为内存储器，主要是有关各类半导体存储器芯片与 CPU 的关系，涉及到信号线的连接，地址的分配与译码，时序关系的分析等；最后，针对外存储器介绍了常用的盘接口。

习 题 3

3.1 何为可改写的 ROM，除了 EPROM 以外，还有哪些产品，各有什么特点？

3.2 在 RAM 存储器中，动态 RAM 与静态 RAM 在结构与性能上有什么不同，二者分别适用于怎样的环境系统？

3.3 作为某一存储器系统，通常是如何来衡量其性能的？

3.4 作为静态 RAM 与 CPU 的连接，应该从哪些方面考虑？有 8 片 6264 存储器芯片，与一 8 位 CPU（有 20 根地址线）相连，构成 64KB 的存储区，地址范围 10000H～1FFFFH，试完成实现这一功能的电路图。

3.5 如果针对一 16 位的 CPU，按上题要求构成 64KB 存储区，其实现电路有什么不同？

3.6 分析图 3-15 的 PC/XT 的 RAM 电路，说明如何用第 9 位存储器芯片实现奇偶校验。

3.7 根据图 3-15 的 PC/XT 的 RAM 电路和图 3-16 的 $\overline{\text{RAS}}$ 和 $\overline{\text{CAS}}$ 产生电路，并对照存储器访问总线周期时序，说明 $\overline{\text{RAS}}$、$\overline{\text{CAS}}$、ADDRSEL 三者的时序关系，并绘出时序图。

3.8 所谓零等待就是在存储器访问过程中不需要插入等待周期 T_W。试分析 CPU 的时序和存储器芯片的参数，说明在什么条件下不需插入等待周期。如果需插入等待周期，如何确定 T_W 的个数。

3.9 什么叫内存条？内存条有哪些类形？

3.10 **磁盘存储器是以怎样的格式来存储信息的？**CPU **与软磁盘驱动器相连需要哪些电路？**

3.11 **作为磁盘接口，试说明** IDE **接口和** SCSI **接口特性的异同。**

第 4 章　输入／输出系统

输入/输出（Input/Output，简称 I/O）是指主机与外界交换信息，这种信息交换是通过输入/输出设备进行的。输入/输出系统，即 I/O 系统，包括 I/O 接口、I/O 设备、I/O 管理部件和与 I/O 有关的软件。各种外部设备通过输入输出接口与主机相连，并在接口的支持下实现各种方式的信息传送。输入和输出系统是计算机系统的重要组成部分之一，可以这样说，任何一台高性能计算机，如果没有高质量的输入/输出系统与之配合工作，计算机的高性能便无法发挥出来。

为了实现人机交互和各种形式的输入/输出，在不同的微机系统中，人们使用了各种各样的外部设备，常见的有：键盘、CRT 显示器、鼠标器、打印机和调制解调器（Modem）等，在一些控制场合，还会用到模/数或数/模转换器、发光二极管等。这些设备和装置的工作原理、信号形式、数据格式各异，为了把外设与 CPU 连接起来，必须有接口部件，以完成它们之间的速度匹配、信号匹配和某些控制功能。所以，它们不可能与 CPU 直接相连，需经过中间电路再与系统相连，这部分电路称为 I/O 接口电路，简称 I/O 接口。I/O 接口就是为使微处理器与输入/输出设备连接起来，并在二者之间正确进行信息交换而专门设计的逻辑电路。

4.1　I／O接口概述

4.1.1　I／O 接口的功能

接口是两个部件之间的连接点或边界，通过接口把 CPU 与外设连接在一起。因此，接口电路要面对 CPU 和外设两个方面，一般来说，I/O 接口应具有以下功能：

（1）数据缓冲和锁存功能。为了协调高速主机与低速外设间的速度不匹配，避免数据的丢失，接口电路中一般都设有数据锁存器或缓冲器。

在输出接口中，一般都要安排锁存环节（如锁存器），以便锁存输出数据，使较慢的外设有足够的时间进行处理，而 CPU 和总线可以去忙自己的其他工作；在输入接口中，一般要安排缓冲隔离环节（如三态门），只有当 CPU 选通时，才允许某个选定的输入设备将数据送到系统总线，其他的输入设备此时与数据总线隔离。

（2）信号转换功能。外设所需要的控制信号和它所能提供的状态信号往往和微机的总线信号不兼容，外设的电平和 CPU 规定的 0、1 电平不一致，因此，需要信号的转换。信号转换包括 CPU 的信号与外设的信号在逻辑上、时序配合上以及电平匹配上的转换，这些是接口电路应完成的重要任务之一。

（3）数据格式变换功能。CPU 处理的数据均是 8 位、16 位或 32 位的并行二进制数据，而外设的数据位宽度不一定与 CPU 总线保持一致，如串行通信设备只能处理串行数据。这时，接口电路应具有相应的数据变换功能。

（4）接收和执行 CPU 命令的功能。一般 CPU 对外设的控制命令是以代码形式发送到接口电路的控制寄存器中的，再由接口电路对命令代码进行识别和分析，并产生若干与所连外

设相适应的控制信号，并传送到 I/O 设备，使其产生相应的具体操作。

（5）设备选择功能。微机系统中一般接有多台外设，一种外设又往往要与 CPU 交换几种信息，因而一个外设接口中通常包含若干个端口，而 CPU 在同一时间内只能与一个端口交换信息，这时就要借助于接口电路中的地址译码电路对外设进行选择。只有被选中的设备或部件才能与 CPU 进行数据交换。

（6）中断管理功能。当外设需要及时得到 CPU 服务时，特别是在出现故障应得到 CPU 立即处理时，就要求在接口中设有中断控制器或优先级管理电路，使 CPU 能处理有关的中断事务，中断管理功能不仅使微机系统对外具有实时响应功能，又使 CPU 与外设并行工作，提高了 CPU 的工作效率。

对一个具体的接口电路来说，不一定要求它具备上述全部功能，不同的外设有不同的用途，其接口功能和内部结构是不同的。

接口电路应根据所连的外设功能进行设计，因此种类繁多，按功能可分为 3 类：

（1）与主机配套的接口，如中断控制、DMA 控制、总线裁决、存储管理等；

（2）专用外设接口，如软盘控制、硬盘控制、显示器控制、键盘控制等；

（3）通用 I/O 控制，如定时器、并行 I/O 接口、串行 I/O 接口等。

4.1.2 I/O 接口的基本结构

I/O 设备不同，其接口电路也不相同，CPU 与 I/O 设备间需传送的信号也不同。归纳起来，I/O 外设与 CPU 之间交换的信息有数据、状态及控制信号，如图 4-1 所示。

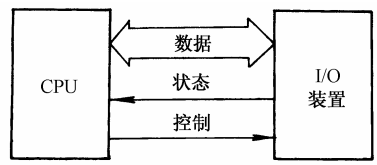


图 4-1　CPU 与外设之间的信息交换

（1）数据信息。根据不同的应用对象，数据信息可分为数字量、模拟量和开关量 3 种。

① 数字量：由键盘、CRT、打印机及磁盘等 I/O 外设与 CPU 交换的信息，是以二进制代码形式表示的数或以 ASCII 码表示的数或字符。

② 模拟量：模拟量是随时间变化的连续量，如温度、压力、电流、位移等。当计算机用于控制系统时，大量的现场信息经过传感器把非电量转换成电量，并经过放大处理得到模拟电压或电流。这些模拟量必须先经过 A/D 转换器转换后才能输入计算机；计算机输出的控制信号也必须先经过 D/A 转换器，把数字量转换成模拟量才能去控制执行机构。

③ 开关量：由两个状态组成的量。如开关的断开和闭合，机器的运转与停止，阀门的打开与关闭等。这些开关量用一位二进数即可表示，故对字长为 8 位（或 16 位）的计算机，一次可输入或输出 8 个（或 16 个）开关量。

（2）状态信号。状态信号是反映外设或接口电路当前工作状态的联络信号。CPU 通过对外设状态信号的读取，可得知其工作状态。如输入设备的数据是否准备好，输出是否空闲，若输出设备正在输出信息，则用 BUSY 信号通知 CPU 以便暂停数据的传输。因此，状态信号是 CPU 与 I/O 外设正确进行数据交换的重要条件。

（3）控制信号。控制信号是 CPU 用来控制 I/O 外设（包括 I/O 接口）工作的各种命令信

息。最常见的如 CPU 发出的读/写信号等。

需要指出的是，数据信息、控制信息和状态信息这三者的含义各不相同，应分别传送，但实际传送中，都是用输入、输出指令（IN、OUT 指令）在系统数据线上传送的。也就是说，把状态信息和控制信息当成一种特殊的数据信息通过数据总线在 CPU 与 I/O 接口之间传送。此时状态信息作为一种输入数据，控制信息作为一种输出数据。

因此，不同的外设对应的接口是不同的，但不论哪种接口，传送的都是上述 3 类信息，都必须具有以下基本部件：

（1）数据缓冲寄存器。数据缓冲寄存器分数据输入缓冲寄存器和数据输出缓冲寄存器两种。输入输出数据缓冲寄存器用来暂时存放输入设备输入的数据或 CPU 输出的数据。

对输入端口而言，由于输入缓冲器的输出是接到数据总线上的，所以，必须具有三态输出功能，当该端口未被选中时，其输出应处于“高阻”状态，与总线隔离；又由于外设输入到输入端口的数据与微处理器读取该端口数据往往不是同步的，因此输入时要求有相应的输入数据锁存器，利用它将来自输入设备的数据锁存起来，并一直稳定地保持到微处理器取走该数据。因此输入缓存器一般由锁存器和三态缓存器组成。

对输出端口而言，它是把来自 CPU 的数据通过系统总线输出到外部设备。CPU 的数据出现在总线上的时间很短，只是执行一条输出指令的时间，一般在纳秒级，而外设通常不可能在这么短的时间内将总线上的数据取走，因此要求输出端口必须有数据锁存器，将输出的数据保持足够长的时间。至于输出端口是否有三态缓冲功能则无关紧要，因为一般每根输出端口线只对应于一个输出设备。

（2）控制寄存器。控制寄存器用于存放处理器发来的控制命令和其他信息，以确定接口电路的工作方式和功能。由于现在的接口芯片大都具有可编程的特点，即可通过编程来选择或改变其工作方式和功能，这样，一个接口芯片就相当于具有多种不同的工作方式和功能，使用起来十分灵活、方便。控制寄存器是写寄存器，其内容只能由处理器写入，而不能读出。

（3）状态寄存器。状态寄存器用于保存外设或接口本身当前的工作状态信息。例如，输入设备通常用 READY 来表示当前是否可以向 CPU 提供数据，输出设备的空闲或忙状态常用 BUSY 来表示。状态寄存器的内容一般只能被 CPU 读出。

（4）内部定时与控制逻辑。用来产生内部工作所需的定时信号以及根据 CPU 的控制命令而产生的控制外设实现具体操作的控制信号。

图 4-2 是接口电路的基本结构框图。由于接口电路介于 CPU 和外设之间，它既要面对 CPU 又要面对外设，因此，在逻辑结构上分为两部分，一部分是与 CPU 相连接，这部分面向主机的逻辑是标准的逻辑，不同的接口差异不是很大。另一部分是与外设相连接的逻辑，随所连接的外设不同而差异较大。

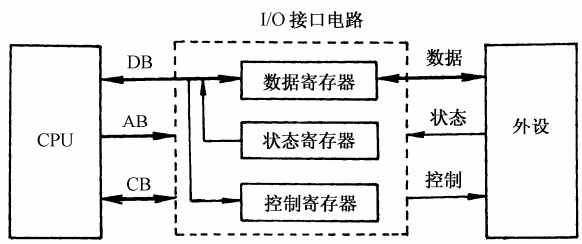


图 4-2　接口电路的基本结构

4.1.3 口地址及译码

一个计算机系统拥有多个 I/O 设备，也具有许多 I/O 接口，各种接口对应于不同的口地址，需要通过地址译码来识别。

1．I/O 端口和端口地址

通常，一个接口电路中包含有一组寄存器。CPU 和外设进行数据传输时，各类信息在接口中存入不同的寄存器。I/O 端口是指那些在 I/O 接口电路中能被 CPU 直接访问的寄存器或某些特定器件。CPU 通过这些端口发出命令，读取状态信息并传送数据。

一个接口电路可能含有一个或几个 I/O 端口，其中用于存放来自 CPU 和内存的数据或外设送 CPU 和内存的数据的端口称为数据端口，简称数据口；用来存放外部设备或接口本身当前工作状态的端口称为状态端口，简称状态口，CPU 通过对状态口的访问可以检测并了解外设或接口的当前状态；用来存放 CPU 发出的控制外设或接口执行具体操作命令的端口称为控制端口，简称控制口。

由此可见，CPU 对外设的输入输出操作，实际上可归结为对接口电路中 I/O 端口的读/写操作。端口的作用不同，特性也不同，有的端口只能读不能写，如状态口；有的端口只能写不能读，如控制口；有的端口既可读又可写，如用于输入/输出的数据口。

为了方便 CPU 对每个端口进行访问，系统给每个 I/O 端口均赋予一个地址，称为端口地址。CPU 要访问接口中的某一端口，需先将端口地址放入地址总线，用高位地址经地址译码选中该接口芯片，用低位地址选择具体要访问的端口，在执行读/写命令时，实现数据的传送。

2．I/O 端口及其编址方式

CPU 对外设的访问实质上是对 I/O 接口电路中相应的端口进行访问，因此和存储器一样，也需要由译码电路来形成 I/O 端口地址。I/O 端口的编址方式有两种，分别称为端口统一编址和端口独立编址。

（1）端口统一编址。端口统一编址又称存储器映像编址。这种方式把每一端口视为一个存储单元，将它们和存储单元联合在一起编排地址，即 I/O 和存储器使用同一个地址空间。这样，可利用访问内存指令去访问 I/O 端口，而不需要专门的 I/O 指令。CPU 采用存储器读写控制信号（如 $\overline{\text{MEMR}}$ 、 $\overline{\text{MEMW}}$），并通过不同的地址空间分配来确定是访问存储器还是访问 I/O 设备。

端口统一编址的特点是：简化指令系统，无需专门的 I/O 指令，但 I/O 端口地址占用了一部分存储器地址空间。

（2）端口独立编址。端口独立编址方式是指 I/O 设备的地址空间和存储器地址空间是独立的、分开的，也即 I/O 端口地址不占用存储器的地址空间。一般来说，因为 I/O 端口数较存储单元数少得多，所以 I/O 地址空间可小于存储器地址空间，CPU 只需用地址总线的低位部分对 I/O 端口进行寻址，如 PC/XT、AT 机仅使用 A_9～A_0 对 I/O 端口进行寻址。CPU 使用专门的 IN（输入）和 OUT（输出）等 I/O 指令来实现数据传送。工作时，CPU 对指令进行译码，区分是存储器读写操作还是 I/O 读写操作。

这种寻址方式的优点是将输入/输出指令和访问存储器的指令明显区分开，使程序清晰，可读性好；而且 I/O 指令长度短，执行的速度快，也不占用内存空间，I/O 地址译码电路比较

简单。不足之处是 CPU 指令系统中必须有专门的 IN 和 OUT 指令，这些指令的功能没有访问存储器的指令的功能强；I/O 端口数目有限。另外，CPU 要能提供区分存储器读/写和 I/O 读/写的控制信号。

3．I/O 端口地址译码

在进行 I/O 操作时，CPU 必须首先确定与自己交换信息的 I/O 端口，这就需要通过地址译码将系统地址总线上的地址代码变为 CPU 所需要的 I/O 端口，其具体电路就是 I/O 地址译码电路。

常见的 I/O 端口地址译码电路为固定式端口译码电路，所谓固定式端口地址译码是指接口中所用的端口地址是事先确定好的，不能改变。变量译码器就是一种常用的地址译码器件。

I/O 地址译码方法是灵活多样的，一般的做法是把地址分为两部分，一部分是用地址的高位与控制信号的不同组合，经译码电路产生 I/O 接口芯片的片选信号，实现接口芯片间的选择；另一部分是用地址的低位部分直接连到 I/O 接口芯片的端口选择端，实现具体端口的选择。

由于 CPU 对 I/O 端口进行操作时，除了用地址信息来选中端口外，还需要相应的控制信号（如 $\overline{RD}$、$\overline{WR}$、M/$\overline{IO}$ 等）决定操作方式和数据流向。因此，I/O 地址译码电路不仅与地址信号有关，还与相应的控制信号有关，它对地址信号和控制信号进行组合，产生对接口芯片的选择信号。所以，I/O 地址译码器的输入除地址信号外还有控制信号。

图 4-3 所示为 PC 机系统板上端口地址译码电路。该译码电路采用 3/8 译码器，对地址线 A_9~A_5 进行地址高位部分的译码，可译出 8 组端口地址，每组地址根据需要还可以进一步细分，最多可达 32 个端口地址。

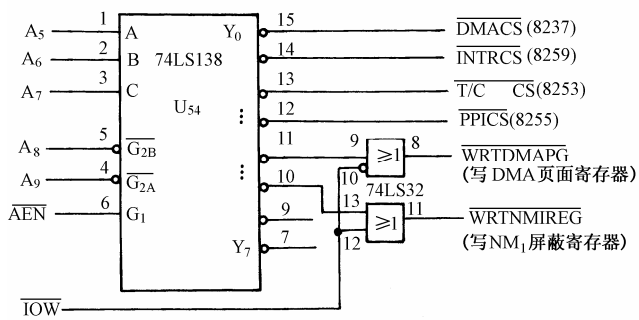


图 4-3　PC 机系统板口地址译码电路

4.1.4　数据传送的控制方式

在各种微型计算机系统中，CPU 与 I/O 接口间的数据传送方式不同，不同的外设需用不同的传送方式，而传送方式不同，CPU 对外设的控制方式也就不同。一般 CPU 与外设间的数据传送方式除了基本传送方式以外，还有中断控制方式、直接存储器存取方式和输入/输出处理器方式等。这些数据传送方式的特点各不相同，计算机系统应为不同的应用要求选择合适的数据传送方式。

1．基本传送方式

基本传送方式是在程序控制下的一种数据传送方式，即通过 CPU 执行程序中的 I/O 指令来完成数据传送。根据外设或接口的响应特征，基本传送方式又可进一步分为无条件传送和查询传送两种。

无条件传送是指数据传送前 CPU 不需了解端口的状态，直接进行数据的传送；查询传送是指数据传送前，CPU 必须先查询端口状态，待端口就绪后方可进行传送。

2．中断控制方式

数据传送请求由外设提出，CPU 视情况响应后，调用预先安排好的中断程序来完成数据传送。

3．直接存储器存取方式

直接存储器存取传送方式简称 DMA 方式，其请求由外设向 DMA 控制器（DMAC）提出，然后 DMA 控制器向 CPU 申请总线，最后 DMAC 利用系统总线来完成外设和存储器间的数据传送。该传送完全由硬件实现，一般在外设和存储器间进行，具有非常高的传送速率。

4．输入/输出处理器方式

输入/输出处理器方式简称 IOP 方式，它是采用 I/O 处理机进行数据的传送和处理的，即由 CPU 委托专门的 I/O 处理机来管理外设，完成传送和相应的数据处理。

4.2 基本I/O接口

CPU 的 I/O 访问与存储器访问十分相似，CPU 执行输入/输出指令，通过地址总线提供 I/O 端口地址选择 I/O 接口电路，通过控制总线提供控制信号确定操作性质，最后通过数据总线与接口电路进行数据交换。与此同时，I/O 接口电路还需与 I/O 设备进行数据传送。

4.2.1 简单的输入接口

CPU 执行 IN 指令的输入过程是在 I/O 设备已经将数据提供给 I/O 接口电路的前提下进行的，是 CPU 读取 I/O 接口电路数据端口中有效数据的过程。输入操作的具体步骤如下：

（1）CPU 把地址置于地址总线上，用于选择某一指定的输入端口；

（2）CPU 等待有效数据的出现；

（3）CPU 从数据总线上读取已稳定的输入数据。

有关输入过程的操作时序如图 4-4 所示。

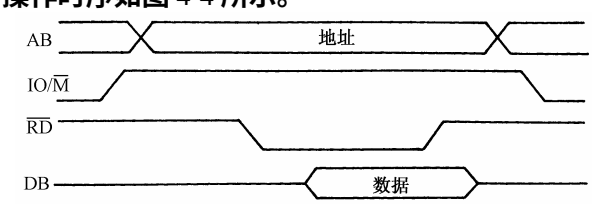


图 4-4　I/O 读总线周期时序

事实上，这一过程就是 I/O 读总线周期。以 IO/$\overline{M}$ 为高电平有效指明当前操作为 I/O 操作，CPU 先向地址总线发送稳定的地址值，然后发送 $\overline{RD}$ 控制信号，最后采样数据总线上的数值。因此，对于所有的输入/输出接口电路， 要求根据给定的地址值来确定哪一个接口电路被选中。对于选中的接口电路，应该在收到 $\overline{RD}$ 有效信号以后，及时将数据送数据总线供 CPU 读取。

从此过程可知，输入接口与 CPU 的数据总线的连接应该具有三态功能，通常处于高阻状态，以避免总线冲突。在 CPU 从数据总线上读取输入数据的时刻，就是选中的输入接口打开三态门向数据总线提供输入数据的时刻。因此，简单输入接口实质上是一个三态缓冲器，控制三态缓冲器的三态门打开的条件有 3 个：

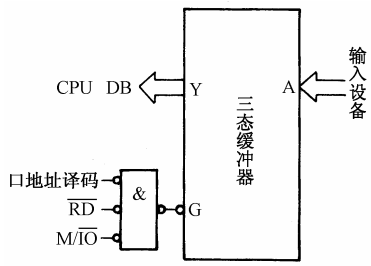


图 4-5　简单输入接口

（1） IO/$\overline{M}$ 为有效电平表示 I/O 操作；

（2）I/O 端口地址被选中；

（3） $\overline{RD}$ 为低电平读操作有效期间。

图 4-5 为一简单输入接口电路。三态缓冲器的输入连至外部输入设备的数据线，三态缓冲器的输出接 CPU 的数据总线。当 CPU 提供的 $\overline{RD}$ 信号、M/$\overline{IO}$ 信号和地址经译码后产生的端口选择信号有效时，三态门打开，外部数据送入 CPU 的系统数据总线。

4.2.2　简单的输出接口

同样，输出过程是 CPU 将需要输出的数据通过 OUT 指令写入到 I/O 接口电路的数据端口，然后接口电路为 I/O 设备提供一个稳定的输出数据以供读取。输出操作的具体步骤如下：

（1）CPU 将地址置于地址总线上，选择某一个指定的输出端口；

（2）CPU 将欲输出的数据置于数据总线上；

（3）CPU 等待数据传送操作的完成。

相似地，输出操作是通过 CPU 执行 OUT 输出指令的 I/O 写总线周期来实现的。在此期间输出接口电路接收 CPU 送往外部输出设备的数据。针对 I/O 写总线周期，接口电路需要输出锁存功能，在 CPU 向数据总线送数据的时刻，输出接口电路就应该锁存这一数据，以供外部设备从容地读取。数据锁存的时刻就是 CPU 执行 I/O 写操作的时刻。因此，简单输出接口实质上就是一个数据锁存器，控制数据锁存的选通条件有 3 个：

（1） M/$\overline{IO}$ 为有效电平表示 I/O 操作；

（2）I/O 端口地址被选中；

（3） $\overline{WR}$ 为低电平读操作有效期间。

图 4-6 为一简单输出接口电路。数据锁存器的输入连至 CPU 的数据总线，锁存器的输出接外部设备的数据线。当 CPU 提供的 $\overline{WR}$ 信号、M/$\overline{IO}$ 信号和地址经译码以后产生的端口选择信号有效以后，就可以触发锁存器锁存 CPU 同时发送的数据了。

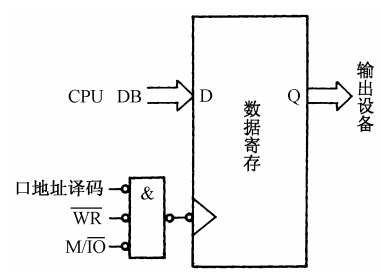


图 4-6　简单输出接口

另外，与存储器访问相同，在执行 I/O 操作时，同样需要考虑配合问题，当 I/O 接口电路的速度较慢时，需要考虑插入等待周期。

4.3 程序查询数据传送方式

无条件传送方式是指当 CPU 进行数据传送时，不用测试外设的状态，直接执行输入/输出指令进行传送的一种方式。采用这种方式的一个必要条件是在任何时刻总认为外设已处于输入设备准备就绪状态、输出设备准备就绪状态或空闲状态，这样就可以不查询外设而直接执行输入输出指令了，数据传送便可以立即进行。

在无条件传送方式下，程序设计较简单，不需查询和等待，硬件、软件较为节省。不过名为无条件传送，实际上对传送是有条件的，那就是传送不能太频繁，以保证每一次传送时，外设均处于就绪状态。所以无条件传送方式较少使用，只用在对一些简单外设的操作中，主要应用于定时为已知或固定不变的低速 I/O 接口（低速 I/O 设备），或无须等待时间的 I/O 设备，如开关、七段显示管等。而对于大部分的应用，通常会采用程序查询方式，CPU 通过了解 I/O 接口电路的状态来决定是否执行输入输出命令来完成 I/O 的交换。

4.3.1 I/O 接口状态的作用

程序查询数据传送方式也称条件传送。采用这种方式传送数据时，CPU 通过执行程序不断读取并测试外设的状态，如果外设数据端口处于准备好状态（输入设备）或者空闲状态（输出设备），则 CPU 执行输入指令或输出指令与外设交换信息。

I/O 接口的状态实质上反映了 I/O 设备的状态。由于 I/O 设备的速度通常远低于 CPU 的速度，为了能及时访问 I/O 设备，CPU 就要不断了解 I/O 设备的状态。一般情况下 I/O 设备的当前状态都是通过接口电路向 CPU 反映的。最常用的状态信号有 READY 和 BUSY。

READY 通常用于描述输入设备的状态，当外部设备向输入接口电路提供有效的输入数据后，READY 状态置为有效，告诉 CPU 外部数据已经准备好。当 CPU 读取接口电路和有效数据以后，READY 应置为无效，禁止 CPU 重复读取，同时又可请求外部设备提供新的数据。

BUSY 状态通常用于描述输出设备的状态。针对慢速的输出设备，CPU 不能连续不断地输出数据，以免外设来不及处理而丢失数据。因此，当外部设备正在进行数据处理而不能接受 CPU 的输出数据时，BUSY 置为有效；BUSY 为无效时，CPU 才能执行输出指令。一旦数据输出到接口电路，BUSY 应置为有效，直到外设数据处理完毕，才可重新接受 CPU 的数据。

因此，I/O 接口状态为 CPU 是否与接口电路交换数据提供了依据。同时，I/O 接口的状态也是随着 CPU 对接口电路的访问以及外设对接口电路的操作而发生变化的。作为程序查询接口电路，除了有传送数据的端口以外，还要有传送状态的端口。一个数据的传送过程由 3 个环节组成:

（1）CPU 从接口中读取状态字；

（2）CPU 检测状态字的对应位是否满足“就绪”条件，如果不满足，则回到前一步继续读取状态字；

（3）如状态字表明外设已处于“就绪”条件，就传送数据；否则，返回状态端口，循环等待，一直查询到准备好，才开始传送信息。

4.3.2 查询式输入接口

对于输入过程来说，当外设将数据准备好后，则将接口的状态端口中的“准备好”标志位置为有效。图 4-7 所示是查询式输入操作的工作流程。

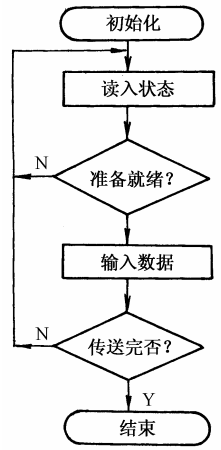


图 4-7 查询式输入操作的工作流程

查询式数据输入可分为 3 个步骤：CPU 从外设输入数据时先读取状态字，检查状态字是否表明数据准备就绪。若准备就绪，则执行输入指令读取数据，且使状态位清 0，这样，便可以开始下一个数据传输过程了。图 4-8 所示为一采用程序查询方式传送数据的输入接口电路，输入设备在数据准备好以后便往接口发一个选通信号 STB。这个选通信号有两个作用，一方面将外设的数据送到接口的锁存器中，另一方面使接口中的一个 D 触发器置 1，从而使接口中三态缓冲器的 READY 位置 1。数据信息和状态信息从不同的端口经过数据总线送到 CPU。

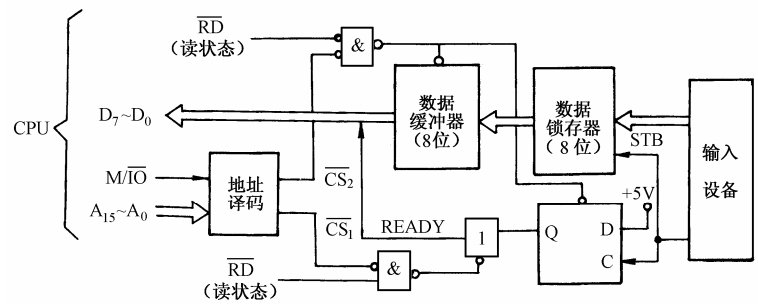


图 4-8 查询方式输入接口

设数据输入口地址为 $PORT_1$，状态口地址为 $PORT_2$，传送的数据字节数为 N，则查询式数据输入的程序如下：

```
        MOV   SI，   0          ；地址指针初始化为 0
        MOV   CX，   N          ；传送的字节数送 CX
START：  IN    AL，   PORT2     ；读状态口信息
        TEST  AL，   02H        ；READY 信息设在 D1，检查是否准备就绪
        JZ    START             ；若未准备好 READY=0，则转 START
        IN    AL，   PORT1      ；准备好则读数据口数据
        MOV   AL[SI]，AL        ；存数据
        INC   SI                    ；修改地址指针
        LOOP  START             ；未完，继续传送，已完则往下执行
```

4.3.3 查询式输出接口

对于输出过程来说，外设取走一个数据后，接口便将状态端口中的对应标志位置为有效，表示当前输出数据端口已经处于“空闲”状态，可以接收下一个数据了。图 4-9 所示为查询

式输出操作的工作流程。

与工作流程相对应，图 4-10 所示为查询方式进行输出的接口电路。当 CPU 要往一个外设输出数据时，先读取接口中的状态字，如果状态字表明外设有空或不忙，说明可以往外设输出数据，此时 CPU 才执行输出指令，否则 CPU 必须等待。

CPU 执行输出指令时，由选择信号 M/$\overline{IO}$ 和写信号 $\overline{WR}$ 产生的选通信号将数据总线上的数据送入接口锁存器，同时使 D 触发器置 1。

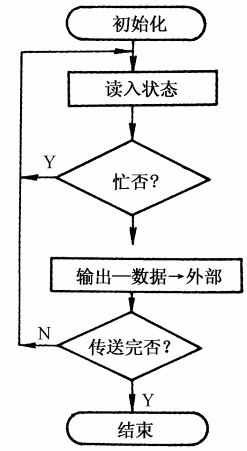


图 4-9　查询式输出操作的工作流程

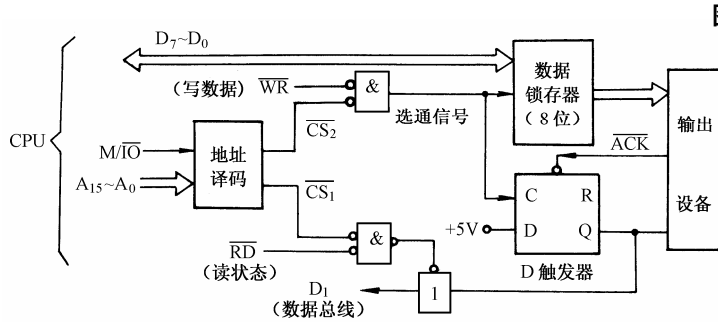


图 4-10　查询方式输出接口

D 触发器输出信号的作用一方面是为外设提供一个联络信号，告诉外设接口中已有数据可提取；另一方面是 D 触发器的输出信号将状态寄存器的对应标志位状态告诉 CPU，当前外设处于忙状态，从而阻止 CPU 输出新的数据。

当输出设备从接口中取走数据后，通常会送一个回答信号 ACK，ACK 使接口中的一个 D 触发器置 0；从而使状态寄存器中的对应标志位置 0，这样就可以开始下一个输出过程了。

4.4　中断传送方式

归纳起来，查询式输入/输出一般是通过下列过程来实现的：程序先对接口进行连续的检测，当检测到的状态表明接口中已有数据准备输入到 CPU　或者接口准备好从CPU 接收数据时，就可以进行输入/输出操作了。

显然，采用程序查询方式实现 CPU 与外设的数据交换，其优点是明显的：硬件电路简单，程序实现也简单。然而，其缺点也是明显的：第一，快速的 CPU 与慢速的 I/O 设备一起工作，CPU 的效率受慢速 I/O 的牵制，浪费 CPU 时间；第二，实时性差，因为由于一个外设的 I/O 未处理完毕，就不能处理下一个外设的 I/O 请求，故不能达到实时处理的要求。

中断概念的引入，就是为了提高 CPU 的效率，同时也改善了系统的实时性。

4.4.1 中断的基本概念

所谓中断是指当机器正在执行程序过程中，受外部逻辑要求，暂停执行原程序转而执行外部逻辑要求的程序，执行完后再返回原程序的过程。

1．中断源

引起中断的事件称为中断源，通常中断源有以下几种：

（1）一般的输入和输出设备，如打印机。

（2）数据通道中断源，如磁盘、磁带等。

（3）实时时钟，如定时器的输出信号。

（4）故障源，如电源掉电等。

（5）软件中断，如在调试程序时设置断点等。

2．中断的作用

中断概念的提出是为了解决快速的CPU与慢速的外部设备之间的矛盾。采用中断方式，主要有以下优点：

（1）并行处理。有了中断功能,可以使 CPU 和外部设备同时工作。CPU 启动外部设备工作以后，就继续执行主程序，同时外部设备也工作，当外部设备把数据准备好后，发出中断请求，CPU 执行中断服务程序进行输入/输出处理，处理完成以后 CPU 恢复执行主程序，外部设备也继续工作。有了中断功能，CPU 可命令多个外部设备同时工作。这样就大大提高了 CPU 的利用率，也提高了输入/输出的速度。

（2）实时处理。当计算机处于实时控制时，中断是一个十分重要的功能。现场的各个参数、信息可以在任何时间发出中断请求，要求 CPU 进行处理，CPU 可以根据需要随时响应。

（3）故障处理。计算机在运行过程中，往往会出现事先预料不到的情况，或出现一些故障，如电源突然掉电、存储器出错、运行溢出等。计算机就可以利用中断系统自行处理，而不必停机。

3．中断系统的功能

为了满足上述各种情况下的中断请求，中断系统应具有以下功能：

（1）能实现中断响应、中断服务及返回。当某一中断源发出申请时，CPU 能决定是否响应这一中断。若允许响应这一中断请求，CPU 能在保护断点（当前 PC 的值）以后将控制转移到相应的中断服务程序中去。中断处理完后能恢复断点，CPU 返回原中断处继续执行主程序。

（2）能实现中断优先权排队。当两个或多个中断源同时提出中断申请时，CPU 要能够根据各中断申请的轻重缓急情况分别处理，即每个中断源确定一个中断优先级别，保证首先处理优先级较高的中断申请。

（3）能实现中断嵌套。若在中断处理过程中，又有新的优先级较高的中断请求，CPU 应能暂停正在执行的中断服务程序，转去响应优先级较高的中断申请，结束后再返回原优先级较低的中断处理过程。这种情况称为中断的嵌套，也称为多重中断处理。

4.4.2 中断的处理过程

一个完整的中断处理过程可分为 4 部分：中断请求、中断响应、中断服务和中断返回。

1．中断请求

中断请求是由中断源提出的，中断源根据其本身工作的实际需要，向 CPU 发中断请求，CPU 将暂停当前执行的程序，转入执行中断服务程序。中断请求往往具有随机性，CPU 并不知道指令的执行将被中断请求所打断。对大多数 CPU 而言，并不是一发生中断请求 CPU 就马上响应，所以必须把中断请求信号锁存起来，并保持到 CPU 响应这个中断请求以后才能被清除。因此，要求每个中断源有一个中断请求触发器。如图 4-11 所示，当中断请求触发器为“1”状态时，表示申请中断；当中断请求触发器为“0”状态时，表示不申请中断。中断请求触发器的请求状态可由外设需要服务的信号建立，由 CPU 的中断响应信号撤销。

一般而言，每个中断源都有权利向 CPU 发中断请求，CPU 也可根据实际的要求对部分或全部的中断源实施中断屏蔽。为了便于控制，常常对每个中断请求触发器都设置一个中断屏蔽触发器，如图 4-12 所示，通过 CPU 对中断屏蔽触发器状态的设置来控制哪些中断源可以发中断请求，哪些中断源禁止向 CPU 申请中断。

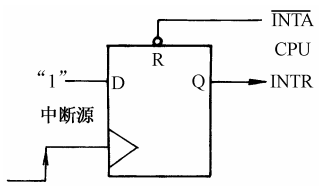


图 4-11　中断请求触发器

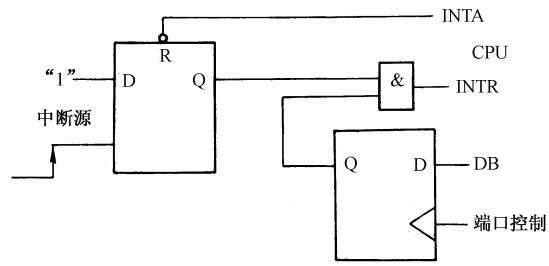


图 4-12　中断屏蔽触发器

2．中断响应

中断响应是指 CPU 获知中断源有中断请求到获得相应的中断服务程序入口地址这一段时刻。这是中断处理过程中关键的一个步骤，如何识别当前申请的中断源，如何确定哪一个中断源有权获得中断响应，最终如何得到正确的中断服务程序入口地址，都在这一阶段解决。

当有中断请求以后，CPU 并非立刻响应，中断响应是有条件的。对于可屏蔽的中断申请，CPU 要响应必须满足以下 3 个条件：首先，当前 CPU 处于中断开放状态，即 CPU 内部的中断允许触发器的状态是允许响应外部中断请求的；其次，当前无总线请求，因为若存在总线请求，CPU 会先释放总线的控制，也就是说，总线的操作优先权高于中断请求；第三，CPU 执行完当前的指令，也就是说，中断操作不能打断一条指令的执行。

如果申请的中断源是非屏蔽中断，则响应与否不受 CPU 内部的中断允许触发器的状态影响。CPU 执行完当前指令后，就可去处理中断服务了。

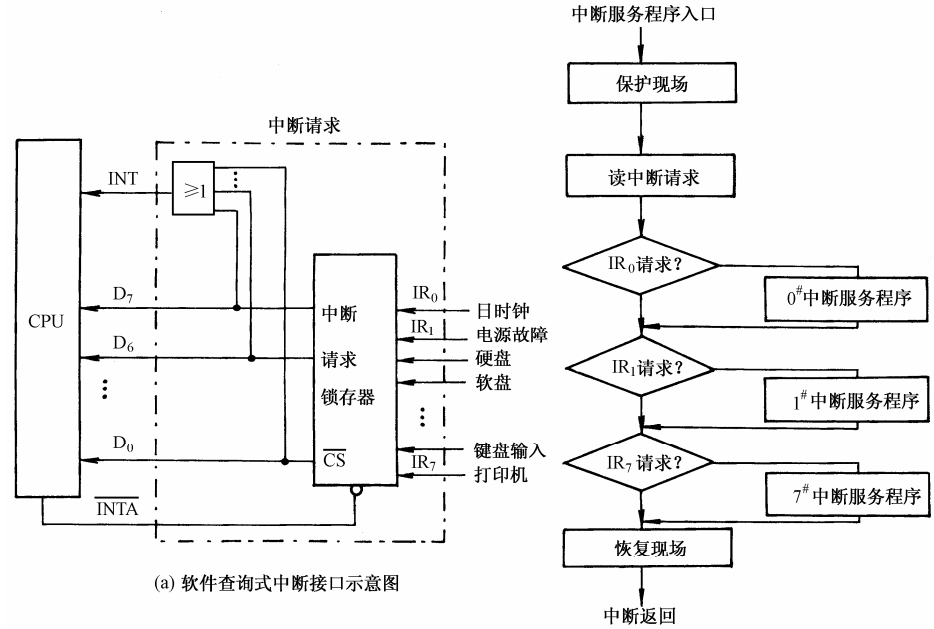
在满足中断响应的条件下，CPU 进入中断响应周期，其间，CPU 将自动完成下述 3 项工作：关中断、保护断点和识别中断源。

（1）关中断。CPU 在响应中断后，发出中断响应信号，同时内部自动关中断，以禁止其他的中断请求。

（2）保护断点。把断点处的地址值压入堆栈保留，以备中断处理完后能正确返回主程序断点。

（3）识别中断源。CPU 要对中断请求进行处理，必须要找到相应的中断服务程序入口地址，这就是中断源的识别。识别中断源有两种方法：

第一种方法称为查询中断。当外设没有提出中断请求时，CPU 照常执行主程序，只有在接收到中断请求信号以后才由 CPU 采用软件方法查询，以识别提出中断请求的设备。软件查询是用程序查询接在中断线的每一个外设上，查询程序依次读出每一个外设的中断状态位，通过测试该状态位来判断对应的外设是否发过中断请求，如果有，则转入相应的中断服务程序。图 4-13 所示就是一个管理 4 个外设的查询测试程序的流程图及对应的硬件示意图。



(a) 软件查询式中断接口示意图

(b) 软件查询式中断流程图

图 4-13　软件查询中断源

第二种方法称为向量中断，也称矢量中断，在具有向量中断的微机系统中，每个外设都预先指定一个中断向量，当 CPU 识别出某个外设请求中断并予以响应时，控制逻辑就将该外设的向量送入 CPU，以自动地提供相应的中断服务程序入口地址，转入中断服务。用向量中断来确定中断源主要是用硬件来实现的，微机系统中常用的可编程中断控制器都能提供中断向量。

3．中断服务

一旦 CPU 响应中断，就可转入中断服务程序了，中断处理要做好以下工作：

（1）保护现场。CPU 响应中断时自动完成断点和标志寄存器内容的保护，但主程序中使用的寄存器的保护则由用户视使用而定。由于中断程序中也要用到某些寄存器，若不保护这些寄存器在中断前的内容，中断服务程序会将其修改。这样，从中断服务程序返回主程序后，程序不能正确执行。由用户保护这些寄存器内容的功能称为保护现场，实际上是执行 PUSH 指令将需要保护的寄存器的内容推入堆栈。

（2）开中断。CPU 接收并响应一个中断后自动关闭中断，是为了不允许其他的中断来打断它。但在某些情况下，有比该中断更优先的情况要处理，此时，应停止对该中断的服务而

转入优先级更高的中断处理，故需要再开中断。若不允许响应更高级别的中断请求即不允许多重中断，在此也可不开中断。

（3）中断服务。中断服务的核心就是对某些中断的处理，如传送数据，处理掉电紧急保护，各种报警状态的控制处理等等。

（4）关中断。关中断对应于上述的开中断，因而在此处对应一个关中断过程，以便下面恢复现场的工作顺利进行而不被中断。

（5）恢复现场。在返回主程序前要将用户保护的寄存器内容从堆栈中弹出，以便返回主程序后继续正确执行主程序，恢复现场用退栈指令。要注意的是堆栈为先进后出的数据结构，注意保护现场时寄存器入栈的先后次序要与出栈时的次序相反。

4．中断返回

在中断服务工作结束后，返回主程序前，也就是中断服务程序的倒数第二条指令往往是开中断指令，最后一条是返回指令，执行返回指令，CPU 自动从现行堆栈中弹出断点地址，以便继续执行主程序。

4.4.3 中断的优先权

在一个微机系统中，常常遇到多个中断源同时申请中断。这时，CPU 必须首先确定为哪一个中断源服务，以及服务的顺序。另外系统若正在执行某一个中断，又有新的中断源发出中断申请，CPU 每次只能响应一个中断源的请求，那么，CPU 是否停止原中断程序的执行而去执行新的中断程序呢？

中断优先级是系统设计者根据各中断源工作性质的轻重缓急，给中断源安排的一个优先服务顺序，即级别。在系统中，有些中断服务是不允许被其他中断打断的，有些中断要优先处理，另外有些中断，在服务期间可以接受比它更需要紧急处理的中断等等。解决这些问题就是解决中断的优先排队问题。

通常，CPU 识别中断源和优先级的排队在系统中是同时解决的，因此，和中断源的识别方法一样，CPU 实现中断优先级排队的方法有两种：软件查询法和向量中断法。根据形成入口地址机制的不同，向量中断法又分为两种：硬件排队法和优先权编码法。

1．软件查询中断优先级

软件查询法又称程序查询法，在简单硬件接口电路（主要是一个中断请求信号锁存器）的支持下，通过查询顺序决定中断优先级别，先被查询的中断源具有高的优先级。使用这种方法需要设置一锁存器，将各中断源的信号保存下来，以便查询并对还没有服务的中断请求做一备忘录。

如图 4-13（a）所示，在软件查询过程中，首先将各外设的中断申请信号接收到中断请求寄存器中，各申请信号可通过或门相“或”后送到 CPU 中去，所有中断源共用一个中断类型号，所有中断请求状态位组成一个端口，赋予一个端口号。当 CPU 响应中断后进入查询程序，读取端口内容，对端口寄存器的内容进行逐位查询，查到哪个外设有中断申请，就转到相应的中断服务程序。查询流程图如图 4-13（b）所示。查询程序的查询顺序，决定了外设中断优先级的高低。

软件查询中断优先级方法的优点在于硬件简单、程序层次分明，可以用修改软件的方法

来改变中断优先级，而不必要更改硬件。软件查询确定优先级的缺点是响应中断速度慢、服务效率低，因为当优先级最低的外设有中断申请时，必须先将优先级高的设备查询一遍，若设备较多，优先级低的中断源可能很难得到服务。

例如，有 8 个中断源，将其中断请求信号相“或”后产生 INTR 信号发给 CPU 并将中断请求触发器组合成一个端口（中断寄存器），并赋予端口地址。

CPU 按程序设定，逐位查询，并转到相应的中断服务程序入口。查询程序一般有两种安排方法。

（1）屏蔽法：

```
IN      AL，20H          ；输入中断是否请求触发器的状态
TEST    AL，80H          ；先查询 D7 是否为 1
JNE     PWF              ；若为 1 则转 PWF
IN      AL，20H，
TEST    AL，40H          ；查询 D6 是否为 1
JNE     DISS             ；若为 1 则转 DISS
    …
```

（2）移位法：

```
XOR     AL，AL           ；清 AL（或 CIR AL）
IN      AL，20H          ；输入寄存器状态
RCL       AL，1              ；带进位左移 1 位，D7→CF
JC      PWF              ；若 D7=CF=1，则转 PWF
RCL       AL，1              ；再左移 1 位，D6→CF
JC      DISS             ；若 D6=CF=1，则转 DISS
   …
```

2．菊花链硬件排队法

菊花链排队法是向量中断的一种，这种方法是利用外设在系统中的物理位置来决定其中断优先级的，主要是利用硬件排队电路（菊花链电路）对中断源进行优先级排队，并将程序引导到相关的中断服务程序入口，电路如图 4-14 所示。

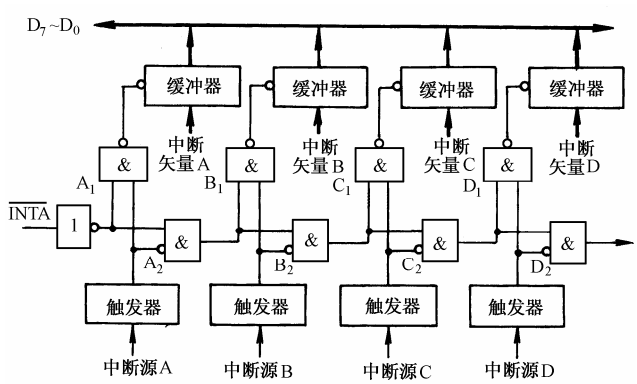


图 4-14　菊花链中断优先级排队电路

图中，来自 CPU 的中断响应信号 $\overline{\text{INTA}}$ 通过多个与门逐次向后传递，形成一个传送 $\overline{\text{INTA}}$ 信号的链条，称为“菊花链”。若中断源 B 发出中断请求（高电平信号）且 CPU 响应时，中断源 B 的中断申请被接收，其输出端通过与门 B_2 输出无效信号来封锁传送 $\overline{\text{INTA}}$ 信号的链条，使它不能向后传递，同时封锁中断源 C 和中断源 D 的中断请求。

在响应中断源 B 并为其服务期间，若中断源 A 发出中断申请，则 CPU 会挂起中断源 B 的服务转去接收优先级高的中断源 A 的中断申请并为其服务。待中断源 A 服务完毕后，再继续为中断源 B 服务。

显然，处于链头的中断源具有最高的优先权，每个中断源的中断优先权由它们在链条中的位置来决定。链式优先级排队电路使优先级别高的中断服务不被优先级别低的中断服务所打断，但可随时中断优先级别低的中断服务。

菊花链方式的特点是中断响应速度快，电路比较简单。但由于门电路的延迟作用，链条的长度即中断源的个数将受到限制，中断源的优先级也因硬件连接固定而不易被修改。

3．优先权编码法

优先权编码法需要采用专用的中断控制器。优先权编码法是向量中断的典型方法，该方法使用一个专门的中断优先级控制器来解决中断优先级的排队管理。图 4-15 所示是一个典型的向量中断优先级控制器原理框图。

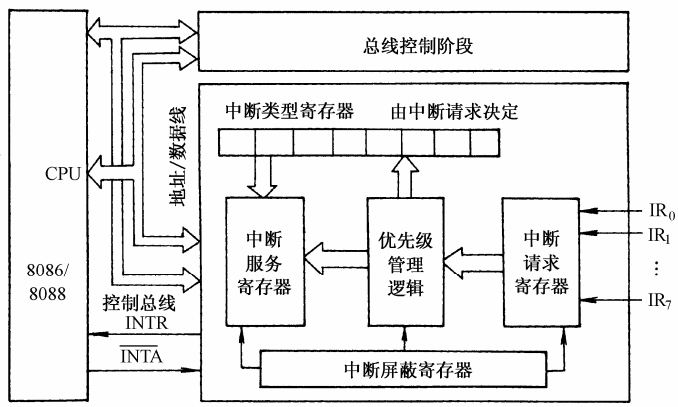


图 4-15　向量中断优先级控制器原理框图

中断控制器的核心电路是中断优先级管理逻辑电路，其实质是一个优先编码器。除此以外，中断控制器还有一个中断请求寄存器，一个中断服务寄存器，一个中断屏蔽寄存器和一个中断类型寄存器。从外设的 I/O 接口送来的中断申请信号可并行地送到中断请求寄存器的输入引脚（IR_0~IR_7）上，中断优先级管理电路从申请信号中确认出当前优先级最高的中断请求，并把申请寄存器的低 3 位转换成 3 位代码送到中断类型寄存器的低 3 位（即对应于中断请求的序号），并同时由中断类型寄存器将此代码送入中断服务寄存器使其相应位置“1”，此后，向 CPU 发出中断申请信号。如果中断允许标志 IF 为“1”，CPU 即发出中断响应信号 $\overline{\text{INTA}}$，从而进入中断响应周期。当中断控制器将中断类型码送给 CPU 后，CPU 将封锁所有低级中断，并根据类型码形成相应的中断服务程序入口地址，从而开始一个中断处理过程。中断处理结束，使中断服务寄存器对应位清“0”，中断级别较低的中断请求才能得到响应。

实际中的中断控制器是可编程的，可以通过软件来为各个中断请求信号分配优先级，通

过中断屏蔽寄存器可以控制中断源的屏蔽与开放，使用非常灵活。Intel 8259A 就是一个可编程的中断控制器。

4.4.4 80286 的中断系统

不同的微型计算机，其中断系统在工作原理上基本相同，但在具体安排、具体实现方法上却有不少差异。80286 的中断系统与 8086 中断系统基础相同，以后的 80X86 系列 CPU 都是在此基础上发展的。80286 的中断系统组织灵活且具有较强的处理功能，它可以处理多达 256 种不同类型的中断，其中断类型码编号为 0~255。中断源有的在微处理器内部，有的在微处理器外部；可以用软件启动中断，也可以用硬件启动中断；可以在 80286 的实地址模式下完成中断处理，也可以在保护模式下完成中断处理。

1．80286 系统的中断源

引起 80286 进行中断操作的原因是多方面的，概括起来大致可分为硬中断、内部中断和软中断 3 种。

（1）硬中断。硬中断是由 80286 外部给出中断信号而发生的中断。如图 4-16 所示，它包括可屏蔽中断、非屏蔽中断和数值处理器异常中断 3 种。它们通过引脚 INTR、NMI 和 $\overline{\text{ERROR}}$ 送入 80286 微处理器。

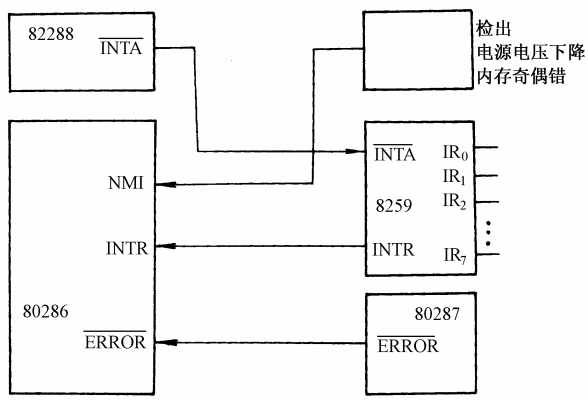


图 4-16　80286 硬中断

可屏蔽中断受微处理器内部标志寄存器的中断允许位 IF 的控制。IF 位实际上是 80286 微处理器可屏蔽中断的中断总开关，由它决定微处理器是否可以响应这种中断。当 IF＝1 时，80286 处于开中断状态，可以响应中断；当 IF＝0 时，处于关中断状态，不能响应中断。

外部的可屏蔽中断请求信号通常由可编程中断控制器送来，经 INTR 引脚输入微处理器中。80286 在每条指令结束时，检查 INTR 引脚上的电位，若为高电平且 IF＝1，它将进入中断响应总线周期。在此响应周期中，从中断控制器中读取 1 个字节的中断类型码，再进行中断的后继处理操作。

非屏蔽中断是一种不能用 IF 位加以禁止的中断，它常用来反映微型计算机硬件系统是否产生了致命性故障。例如，可在存储器中设置奇偶校验电路，一旦存储器操作中出现了奇偶校验错误，便发出非屏蔽中断请求，通知微处理器尽可能做出响应和处理。非屏蔽中断请求信号经 NMI 引脚送入微处理器中。这是一种触发脉冲信号，该信号被 80286 锁存后，将会无

条件地产生 2 类中断，即中断类型码为 2 的中断响应。当 80286 处理某个非屏蔽中断过程时，又收到了新的非屏蔽中断请求，该请求信号又将锁存，并待前一个非屏蔽中断处理完，且执行了返回操作系统命令后，才开始后一个非屏蔽中断的处理。

通常，用户可以使用开中断指令 STI 或关中断指令 CLI 来开放或禁止 INTR 线上的中断请求，但无法禁止 NMI 线上的非屏蔽中断请求。

在 80286 系统中配置了数值数据协处理器 80287 的情况下，浮点运算均由 80287 完成。80287 可对运算进行出错检查，一旦发现错误，便通过 $\overline{\text{ERROR}}$ 引脚向 80287 发出数值数据协处理器异常中断请求。该请求信号是一个低电平有效的信号。

（2）内部中断。所谓内部中断，指在执行指令的过程中，80286 内部确认必须进行异常处理时，自动产生的中断。这类中断主要用于处理除法运算溢出、非法指令运行以及保护故障等微处理器内部产生的随机事件。80286 可定义的内部中断如表 4-1 所示。

表 4-1　80286 的内部中断

中 断 类 型	中 断 原 因
0	除法溢出
1	单步调试中断
6	执行非法指令
7	无协处理器可用
8	双重异常
9	协处理器访问超越段界数据
10	非法访问任务状态段
11	段不存在
12	堆栈故障
13	一般存储器保护故障

其中，“双重异常”指前一个中断试图寻找处理程序时，又产生了一个意外的异常中断，这时就出现了双重异常。

实际上，80286 的内部中断都是一种异常中断。例如，指令要引用一个不在内存中的操作时，就会产生一个异常中断，通过异常中断处理，把该操作数所在的段从外存调入内存中，然后再启动该指令执行。

（3）软中断。80286 的硬中断和内部中断有一个共同的特点，那就是中断发生是随机的，不能由程序员预定时刻和位置。而软中断则不同，它是由中断指令引起的中断，用户可以通过对中断指令的安排，有意识地、准确地把微处理器引导到某个中断处理程序去运行。

在 80286 的软中断指令中，INT 0 是一条特殊的指令。该指令常被程序员安排在程序中的整型数据运算指令之后，监督运算的结果状态。如果整数运算发生溢出（OF＝1），INT 0 指令将引起中断 4，程序利用中断 4 来处理这种异常；如果整数运算没有溢出（OF＝0），INT 0 指令将不引起转移，微处理器继续执行 INT 0 后面的指令。

软中断指令中的 INT 3 指令是断点中断指令，也称自陷中断指令。它常用于关键地方中断程序的执行，这些地方就是断点。引入断点是为了更好地观察程序的运行状态。根据用户的需要，一般在程序中希望在中断的位置上设置 INT 3 指令。程序运行时，当执行到这条指令时就会引起类型为 3 的中断。由于该指令是一条单字节指令，应用起来十分方便。

数组上、下界检验指令 BOUND 是 80286 的另一条软中断指令，用法如图 4-17 所示。假设在定义数组时，数组 BUF 的最小偏移量 0 和最大偏移量 49 设定在 4 字节变量 VAR 中，指令

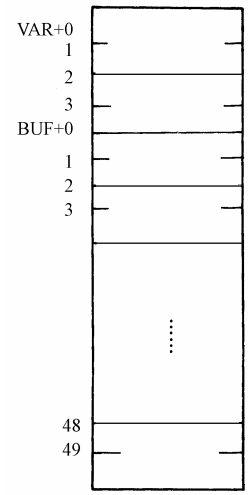


图 4-17　BOUND 指令的应用

BOUND　BX，VAR

将检查 BX 的值是否大于等于 VAR 中低位字所定义的值(0)且小于等于 VAR 高位字中所定义的值（49)。若满足这两个条件，则直接转移到 BOUND 指令的下一条指令去执行；否则，将产生类型 5 中断。BOUND 指令的第一操作数可以设置为 80286 的任何一个通用寄存器。

INT n 是 80286 中最为典型的软中断指令，指令中包含着中断类型码，中断类型码的理论值是 0~255。用户利用软中断指令可以很方便地实现与系统的连接，可以很方便地调用操作系统中的子模块，还可以方便地调用系统中的外部设备。

2．实模式下的中断

80286 实模式下的中断除了定义一些新的内部中断之外，其余与 8086、8088 基本相同。它采用了典型的向量中断方法，根据系统为每个中断规定的类型号，查找中断向量表，从中获得中断处理程序的入口地址，继而转去执行中断处理程序。

（1）中断向量表。在 80286 的向量中断中，中断向量表是中断类型号与该中断类型对应的中断处理子程序入口地址之间的连接表。实模式下的中断向量表共有 256 项，对应着 256 种不同类型的中断。表中的每一项都是一个双字指针，其高位字是中断处理程序入口的段基值 CS，低位字是入口地址的段内偏移地址值 IP。由于向量表每一项的长度为 4B，因此 256 个中断向量将占用 1KB 的存储空间。图 4-18 所示为实模式下的中断向量表。

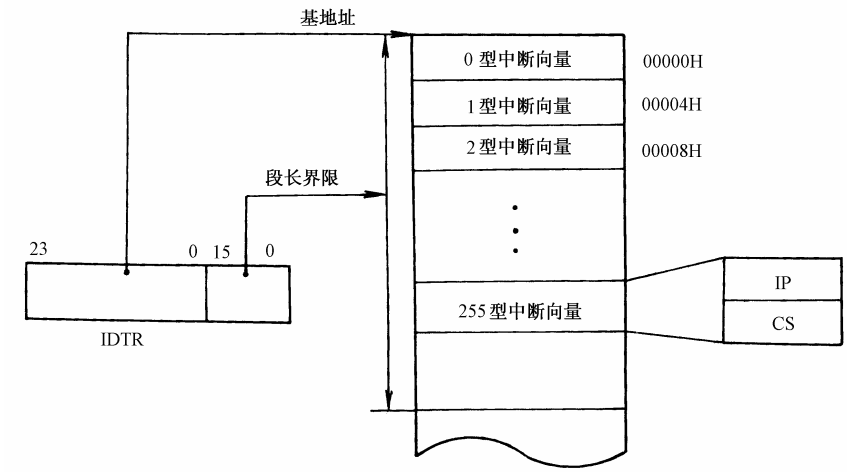


图 4-18　实模式下的中断向量表

实际上，实模式下中断向量表的基地址由中断描述符表寄存器 IDTR 决定。IDTR 本来用于在保护模式下去指定中断描述符表 IDT，在实模式下借用它指示中断向量表的位置。在实

模式下，通过使用装入中断描述符表寄存器指令 LIDT，可向 IDTR 中送入不同的内容，从而把中断向量表移动到 16MB 内的任意地址区域。由于 80286 微处理器在上电或复位时，IDTR 中的基地址字段被置为 000000H、段长界限字段被设为 00FFFFH，因此，实模式下 80286 的向量表配置在从最低物理地址开始的 1KB 存储空间上，即 000000H~0003FFH。这样就保持了 80286 的实模式中断与 8086、8088 的兼容性。

（2）实模式下的中断类型。80286 实模式下有多种中断类型，如表 4-2 所示。表中的“返回地址”一栏表示中断发生时压入堆栈的从中断程序返回的目的地址。“第一字节”表示发生中断的指令其下面一条指令的开始地址。而“没有关系”是指引起中断的原因和发生中断时执行的指令之间没有任何关系。

表 4-2　实模式下的中断类型

中断类型	中断名称	有关的指令	返回地址
0	除法错异常中断	DIV、IDIV	第一字节
1	单步中断	所有指令	下条指令
2	NMI 中断	所有指令	没有关系
3	断点中断	INT 3	下条指令
4	溢出异常中断	INT 0	下条指令
5	边界检查异常中断	BOUND	第一字节
6	无效的操作码异常中断	—	第一字节
7	扩充处理器无效异常中断	ESC	第一字节
8	中断表边界过小异常中断	LIDT	第一字节
9	扩充处理器段超越中断	ESC	没有关系
10~12	备用	—	—
13	段超越异常中断	存储器指令	第一字节
14、15	备用	ESC、WAIT	—
16	扩充处理器出错中断	—	第一字节
17~31	备用	—	—
32~255	用户定义	—	—

实模式下前 6 种类型的中断是 80286 规定的专用中断，其他的中断类型在系统中安排使用。但是，由于软件上的一些原因，对于某个具体的微机系统，其操作系统的一些低层模块可能已经定义了 00H~1FH 范围内的一些中断，这将引起一些矛盾。例如，BOUNDS 指令是使数组上、下边界检验软中断指令，它将引起类型 5 中断，而某个微机系统可能定义类型 5 中断是屏幕打印中断。此外，随着微机操作系统功能的加强，在不同的操作系统版本下，中断类型的安排也不尽相同。

（3）实模式下的中断过程。80286 每执行完一条指令都需要检查是否有中断发生。当然也有特殊情况，即执行那些将一定值代入 SS 寄存器的传送指令 MOV 或出栈指令 POP 时，执行之后就不去检查有没有发生中断，而是执行完下一条指令后才去检查。这是因为在修改完 SS 而未修改 SP 的状态下处理中断时，会将错误的区域作为堆栈来使用。为此，执行变更 SS 的指令之后就接着执行一条变更 SP 的指令，这种设计避免了上述错误的出现。

在有中断发生时，首先要取得中断类型码。对于内部中断的软中断，微处理器会自动形成中断类型码；对于外部的非屏蔽中断请求，也将自动产生中断类型码 2；若是数值协处理器异常故障，则将自动产生中断类型码 16；如果是外部的可屏蔽中断请求，且标志寄存器的

中断允许位 IF＝1，微处理器将进入中断响应周期，在该周期内从外部取入中断类型码。

80286 得到中断类型码后，接着便可形成中断向量地址。对于任一指定类型的中断，微处理器只要将其类型码乘以 4，便可得到对应的中断向量地址。接下来，将微处理器内的标志寄存器内容压栈，并清除标志寄存器的 IF 和 TF 位，以使微处理器关中断，屏蔽新的可屏蔽中断请求 INTR 以及禁止单步中断。然后保存断点，将 CS 和 IP 的内容压栈，并从向量表中取出双字指针分别送入 CS 和 IP 中。在新的程序指针的引导下，微处理器将转入中断处理程序运行。

中断处理程序也称中断服务程序，它完成对不同中断源请求的处理工作。在中断处理程序的最后，均安排有一条中断返回指令 IRET。IRET 指令从堆栈中弹出原指令指针 IP、代码段寄存器 CS 和标志寄存器 FLAG 的内容，从而使 80286 结束中断服务程序的运行，返回到被中断的原程序中去。

3．保护模式下的中断处理

在保护模式下，80286 所能处理的中断类型较实模式有所增加，表 4-3 列出了这些类型。

表 4-3　保护模式下的中断类型

中 断 类 型	中 断 名 称	返 回 地 址	可否再启动	出 错 代 码
0	除法错异常中断	第一字节	可	无
1	单步中断	下条指令	可	无
2	NMI 中断	没有关系	可	无
3	断点中断	下条指令	可	无
4	溢出异常中断	下条指令	可	无
5	边界检查异常中断	第一字节	可	无
6	无效的操作码异常中断	第一字节	可	无
7	扩充处理器无效异常中断	第一字节	可	无
8	双重错异常中断	一次错的返回地址	否	有
9	扩充处理器段超越中断	没有关系	否	无
10	无效 TSS 异常中断	第一字节	可	有
11	段不存在异常中断	第一字节	可	有
12	堆栈异常中断	第一字节	可	有
13	一般保护异常中断	第一字节	可	有
14、15	备用	—	—	—
16	扩充处理器出错中断	第一字节	可	无
17~31	备用	—	—	—
32~255	用户定义	—	—	—

表中的“返回地址”与实模式下的中断相同。“可否再启动”指的是若消除异常中断的原因以后，还能继续执行被中断的程序，则这种中断就是可再启动的。“除法错异常中断”是一种可再启动的中断，因为可以通过增加除数后再运行除法指令。可再启动性对于缺段所引起的异常来说具有明显的意义。在虚拟存储系统中，这样的异常中断并不算出错，它仅使操作系统将所缺的段调入内存，并继续运行程序。如果做不到这一点，在一个对缺段异常不能再启动的计算机中，要产生一个可靠的虚拟系统是相当困难的。

从表中可见，在保护模式中，许多中断还可以为中断服务程序提供一些附加的信息而不

仅仅是返回地址，它们定义了所产生的错误种类。对于这些中断，80286 在进入中断服务以前，把返回地址和 16 位的出错代码压入堆栈。

（1）中断描述符和中断描述符表。80286 保护模式下对中断的处理与实模式有着较大的差异。尽管两种模式都采用了向量中断，但在保护模式下，向量表中的每一项不是双字指针，而是 8B 的门描述符，通过这些门描述符，把 80286 引导到中断处理程序中。这样，保护模式下的中断向量表，往往被称为中断描述符表 IDT，表中的每一项便是中断描述符。

在整个 80286 系统中，中断描述符表只有一个。它可以被安排在内存空间的任何区域内，其基地址和大小仍由微处理器中的中断描述符表寄存器 IDTR 给定。系统初始化时，利用 LIDT 指令可以对中断描述符表寄存器 IDTR 设置固定的内容，以使其指向中断描述符表 IDT。该寄存器一旦被设定，在机器工作过程中将不再改变。保护模式下的中断描述符表的设定如图 4-19 所示。

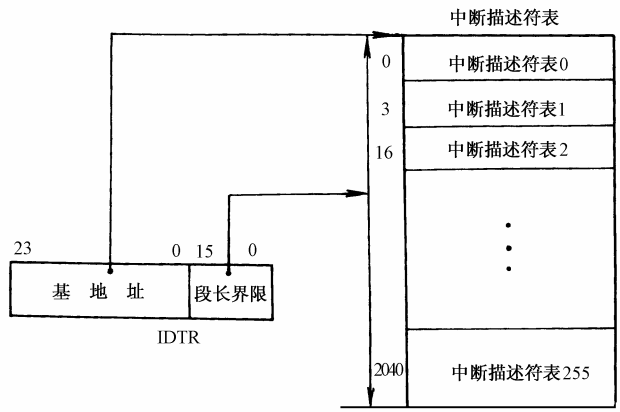


图 4-19　中断描述符表 IDT 和中断描述符表寄存器 IDTR

保护模式下的中断描述符是一种控制转移的系统描述符，它包括 3 种不同类型的门：自陷门、中断门和任务门。

自陷门是用于异常处理的标准控制门，除了不禁止中断外，通过自陷门访问中断处理程序的方法与实模式基本相似。这样做的好处是一旦发生了中断，也不会关闭由定时中断所构成的操作系统的时间片机构。

在自陷门中，提供了 16 位的供选择中断处理程序所在代码段的段选择符以及访问中断处理程序的 16 位偏移量，还包括反映有关特权级的信息。自陷门的格式如图 4-20（a）所示。

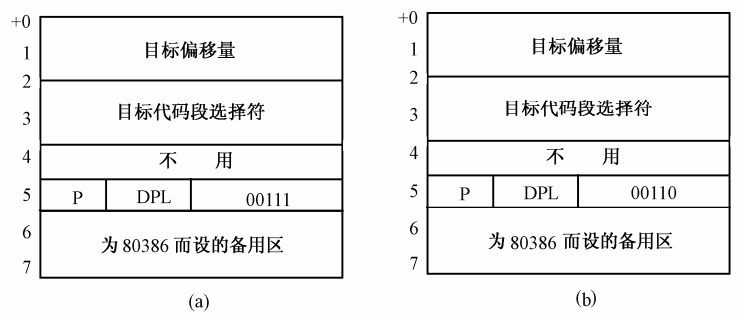


图 4-20　80286 的自陷门和中断门格式

80286 中断门的工作与自陷门一样，完成中断的控制转移功能。所不同的是，在调用所访问的中断处理程序之前，将会封锁，即复位 IF 位。中断门的格式如图 4-20（b）所示。

在中断系统中，任务门往往用做外部中断的标准门，由任务门访问的中断处理程序通过任务调用操作送入。在 80286 保护模式下，外部设备送来的中断信号通常是不能直接与现行任务相连的，只能完全与操作系统相连。实际上，80286 提供任务调用的主要原因就是要允许中断处理被正确、有效地执行。

（2）保护模式下的中断响应。与实模式一样，80286 每执行完一条指令都要查询有无中断发生。当有中断发生时，通过类似的方法获得中断类型码。从中断描述符表中偏移量为“中断类型码×8”处，取出对应的中断描述符。

微处理器依次将标志寄存器 FLAG 内容、代码段寄存器 CS 内容及指令指针 IP 内容压栈，根据已经得到的中断描述符形成中断处理程序代码段选择符和入口偏移量，并送入 CS 和 IP 中。需要特别指出，由于中断对过程的处理与中断对任务的处理不同，因而上述入口地址的形成步骤也不相同。

在进行中断处理之前，有些类型的中断还会将出错代码压栈。在中断描述符为中断门的情况下将 IF 复位，以封锁中断，还要清除单步中断标志位 TF 和嵌套任务标志位 NT。保护模式下的中断响应过程如图 4-21 所示。

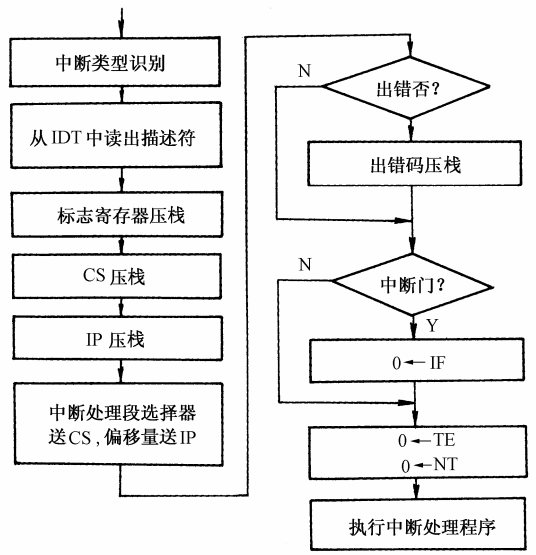


图 4-21　保护模式下的中断响应过程

（3）中断任务和中断过程。正如转移、调用指令能够调用一个过程或一个任务一样，一个中断或异常能够调用中断处理程序。中断处理程序可以是一个过程，也可以是一个任务。当 80286 响应一个中断或异常时，经中断类型的识别从中断描述符表 IDT 中读出对应的描述符。若读出的是自陷门或中断门，它将用与 CALL 指令对调用门的管理方法相同的方法去调用中断处理程序。如果找到的是任务门，它将用 CALL 指令对任务门产生一个任务转换。通过自陷门、中断门调用一个过程的操作如图 4-22 所示。

由图可见，微处理器通过查询中断描述符表 IDT 找到一个门，从中取出选择符在 GDT 或在当前 LDT 中找出可执行段的描述符；门中的偏移量指向中断或异常处理过程的开始点，从而可以进入中断或异常处理程序的过程。通过中断描述符表中的任务门实现任务转换的操

作如图 4-23 所示。

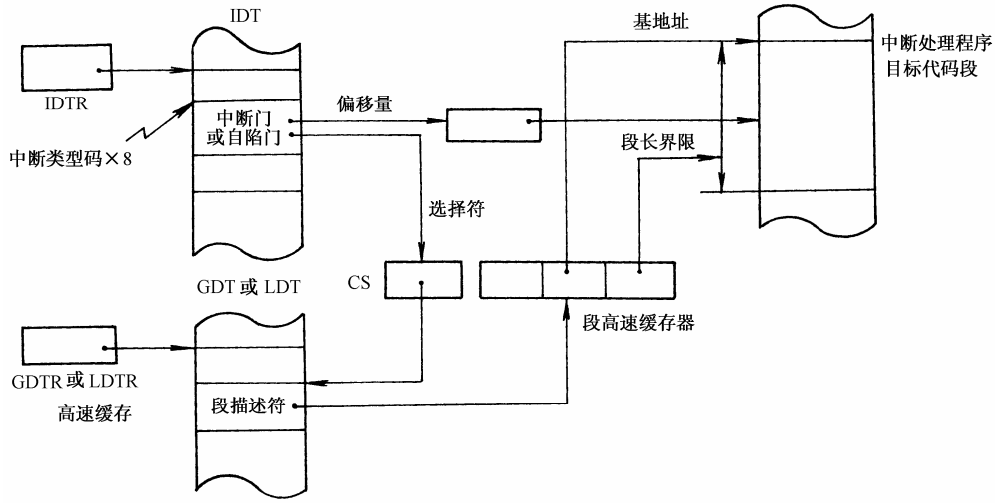


图 4-22　中断对过程的调用

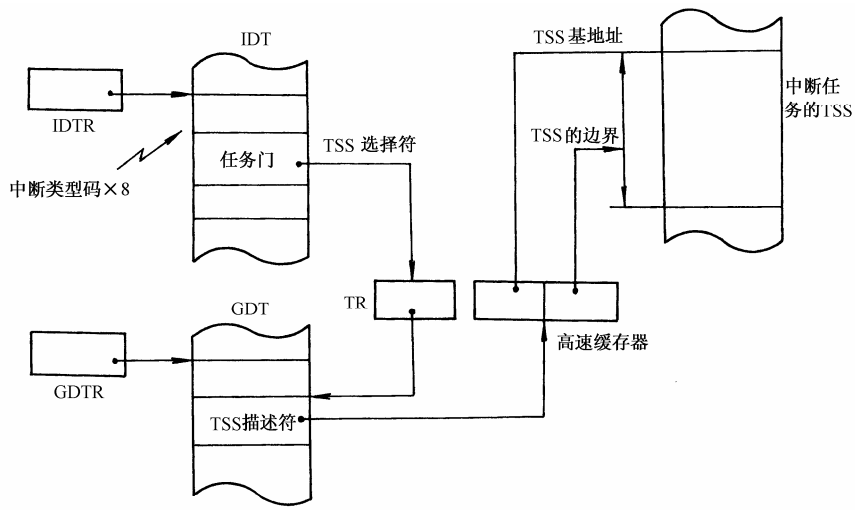


图 4-23　通过中断完成的任务转换

中断处理完毕后，通过中断返回指令 IRET 可使 80286 返回到被中断的程序或通过中断返回操作进行任务转换。80286CPU 为了有效地实现多任务管理，在保护模式下为每项任务设置一个任务状态段 TSS，用于记录任务的当前状态，也包括在 TSS 的反向链选择符中记录了反映任务转移目标的信息。因此，通过中断返回可以进行任务转换。

4.5　DMA传送方式与I/O处理机方式

利用程序中断，在一定程度上提高了微型计算机的效率。例如，某一外部设备 1s 能传送 100 个字节。若用程序查询方式，则在这 1s 内，CPU 全部用于查询和传送；若用中断方式，CPU 传送一个字节的服务程序需要 100μs，故传送 100 个字节只需 10ms，即只用 1s 的 1%，在另外 99%的时间内，CPU 仍可执行主程序。

但是，中断传送方式仍然是由 CPU 通过程序来传送数据的。对于某些外设，如磁盘、

CRT 显示器、高速模/数转换器等要求高速而大量地传送数据时，用程序控制的传送方式来传送数据往往无法满足速度的要求。因为在这种方式下，每传送一个字节就得把主程序停下来，转而去执行中断服务程序，在执行中断服务程序前要做好现场保护工作，执行完中断服务程序后还得恢复现场。由于数据传送过程始终受 CPU 的干预，CPU 需要读取以及执行一系列指令，每一字节数据都必须经过 CPU 的累加器才能输入/输出，这就从本质上限制了数据传送的速度。为提高数据传送速度，可采用 DMA 技术，实现外设与存储器的直接数据传送。

4.5.1 DMA 操作

DMA（Direct Memory Access）传送方式也称直接存储器存取方式， 能够在外部设备与内部存储器之间直接地相互交换数据。DMA 传送方式是一种不需要 CPU 干预也不需要软件介入的高速数据传送方式。在 DMA 传送方式中，对这一数据传送过程进行控制的硬件称为 DMA 控制器。

根据 DMA 传送方式可知，DMA 传送不需要 CPU 的干预，当然也不需要软件的介入。因此，在 DMA 操作期间，微机的系统总线就频繁用于内存储器与外部设备之间的数据传送。可见，DMA 传送操作具有速度快、效率高的特点，通常适用于数据传送速率高、数据传送量大的外部设备之间进行数据交换。

1．DMA 操作的基本方法

DMA 操作的基本方法有 3 种。

（1）周期挪用。一种实现 DMA 传送的方法是把 CPU 不访问存储器的那些周期“ 挪用 ”来进行 DMA 操作。在这种方式下，DMA 控制器可以使用总线而不用通知也不影响 CPU，这种方法称为“ 周期挪用 ”。使用这种方法的主要问题是如何识别可挪用的周期，以避免与 CPU 的操作发生冲突。有的 CPU 能够产生一个存储器是否被使用的信号，而在大多数的 CPU 中必须通过比较复杂的时序电路来加以识别。这种操作方式不影响也不减慢 CPU 的操作速度，但所需的电路较为复杂，而且数据的传送是不连续、不规则的，所以使用不太普遍。

（2）周期扩展。第二种 DMA 操作的方法是使用专门的时钟发生器/驱动器电路。当需要进行 DMA 操作时，由 DMA 控制器发出请求信号给时钟电路。于是，时钟电路把供给 CPU 的时钟周期加宽，而提供给存储器和 DMA 控制器的时钟周期不变，这样，CPU 在加宽的时钟周期内操作而不往下进行，而这加宽的时钟周期相当于若干个下沉的时钟周期，可以用来进行 DMA 操作。在加宽的时钟结束后，CPU 仍按下沉的时钟继续操作。这种操作方式可以使 CPU 操作和 DMA 操作同时进行，但会降低 CPU 的处理速度。

（3）CPU 停机。这是一种最常用也是最简单的 DMA 传送方式。在这种方式下，当 DMA 控制器要进行 DMA 传送时，向 CPU 发出 DMA 请求信号，迫使 CPU 在现行的总线周期结束后，使其地址、数据、控制总线引脚处于高阻状态，从而让出总线的控制权，并给出一个 DMA 的应答信号，使 DMA 控制器可以控制总线进行数据传送，直到 DMA 控制器完成数据传送使 DMA 请求信号无效以后，CPU 再恢复对系统总线的控制，继续进行被中断的操作。

在 CPU 停机的操作方式下，CPU 出让总线控制权的时间取决于 DMA 控制器保持 DMA 请求信号的时间。在进行 DMA 操作时，CPU 就处于空闲状态，所以在这种操作方式下 CPU 的利用率要降低，CPU 对中断的响应操作也会受到影响。

2．DMA 传送方式

DMA 控制器一般有 3 种基本的传送方式：

（1）单字节传送方式。每次 DMA 请求只传送一个字节，每传送完一个字节后释放总线，由 CPU 控制总线至少一个完整的总线周期。以后又测试 DMA 请求线 DREQ，若有效，则进入 DMA 周期。

（2）成组传送方式。每次 DMA 请求连续传送一个数据块，该数据块的长度由编程设定。DMA 请求信号 DREQ 保持到 DACK 响应信号有效，即开始传送。一旦开始传送，DMA 控制器一直不放弃总线控制权，直到预先设定长度的数据块传送完毕后才释放总线。

（3）请求传送方式。请求传送方式又称查询传送方式。该方式的传送类似于成组传送方式，但每传送一个字节后，DMA 控制器就检测 DREQ，若无效，则挂起；若有效，则继续 DMA 传送，直到一组数据传送结束或外部强制 DMA 控制器中止操作为止。

4.5.2 DMA 控制器的作用与结构

DMA 控制器也称 DMAC，是用于与 CPU 实现总线请求与应答以及在 DMA 操作期间实施对总线控制完成数据传送的硬件电路。

1．DMAC 的功能

一般 DMAC 应具有如下功能：

（1）能进行总线请求和应答。在需要 DMA 传送的时候，DMAC 能接受外设的 DREQ 请求信号，向 CPU 发总线请求 HOLD 信号，在合适的时间获得总线控制权，并向外设发 DMA 响应信号 DACK。

（2）能提供存储器的地址信息。在 DMA 操作期间，DMAC 能向内存储器提供地址值，用于指定传送数据的存储器单元地址，并且根据 DMA 传送的需要不断修改地址值。

（3）能提供控制信息。在 DMA 操作期间，DMAC 能向内存储器和 I/O 接口提供读信号或写信号，控制内存储器向 I/O 接口或 I/O 接口向内存储器传送数据。

（4）能控制 DMA 操作结束。在 DMA 操作期间，DMAC 能根据数据的传送量来判别 DMA 操作是否结束，在 DMA 操作结束时，撤销总线请求信号，以便由 CPU 接管总线。

2．DMAC 的结构

DMAC 的结构示意图如图 4-24 所示。一般 DMAC 内部有 3 个寄存器：

（1）地址寄存器。地址寄存器中存放的是当前 DMA 操作时传送数据所用的内存储器地址。在 DMA 操作期间，地址寄存器的值将送地址总线，以选择某一个存储单元。若当前是存储器→I/O 操作，则地址寄存器将指定的存储单元内容送数据总线；若当前是 I/O→存储器操作，则将数据总线上的数据送到地址寄存器指定的存储单元中。每传送一字节数据，地址寄存器的值要加 1。

（2）字节数寄存器。字节数寄存器也称计数器，用于存放 DMA 操作时需要传送数据的字节数。每传送 1 个字节，字节数寄存器的值减 1。DMAC 依据字节数寄存器的值是否为零来判断当前 DMA 操作是否结束。

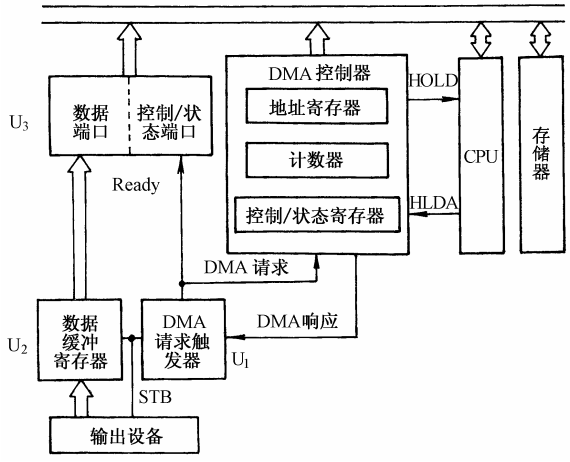


图 4-24　DMAC 结构

(3) 控制/状态寄存器。控制/状态寄存器用于设置 DMAC 的工作方式和当前的工作状态。CPU 可以向 DMAC 的控制/状态寄存器写控制命令来确定 DMA 操作是存储器→I/O 操作还是 I/O→存储器操作以及其他控制要求；CPU 可以读取控制/状态寄存器来了解当前 DMA 的工作情况。

3．DMA 的操作过程

用 DMA 方式进行外设与存储器间的数据传送，既可将外设中的数据经接口传送到内存，也可以将内存中的数据经接口传送到外设，微机中对磁盘的读写操作就是这种情况。一般说来，DMA 的操作过程可分为 3 个阶段：准备阶段（初始化阶段）、DMA 数据传送阶段和 DMA 结束阶段。

在 DMA 操作之前，DMAC 作为系统的一个接口部件，接受 CPU 送来的操作命令，即对 DMAC 的初始化，以规定传输类型（存储器和外设之间）、操作方式（单字节传送）、传送方向、内存的首地址、传送的字节数等。在完成初始化编程之后，DMAC 就准备进行 DMA 操作了。一般 DMA 的操作过程如下：

（1）外设已做好接收数据的准备，向 DMAC 发出请求信号 DREQ；

（2）DMAC 向 CPU 发出 DMA 操作请求 HRQ，该请求信号送到 CPU 的 HOLD 信号引脚；

（3）CPU 在完成当前的总线周期操作之后（若总线处于空闲状态，则立即做出响应）使数据总线、地址总线及部分控制信号处于三态状态，并发出响应信号 HLDA，指示 DMAC 可以使用总线；

（4）DMAC 将地址放入地址总线，该地址用来寻址内存储单元；

（5）DMAC 向 I/O 接口发出响应信号 DACK，该信号可作为接口接收数据的控制条件，有效时，表示允许接口接收数据；

（6）DMAC 分别向存储器和接口发出存储器读写和 I/O 读写控制信号，完成相应的存储器与 I/O 之间的数据传送；

（7）修改 DMAC 内部的地址寄存器和字节数寄存器的值；

（8）判别 DMA 操作是否结束，若结束则释放总线控制权。

图 4-25 为 DMA 工作过程的波形图。

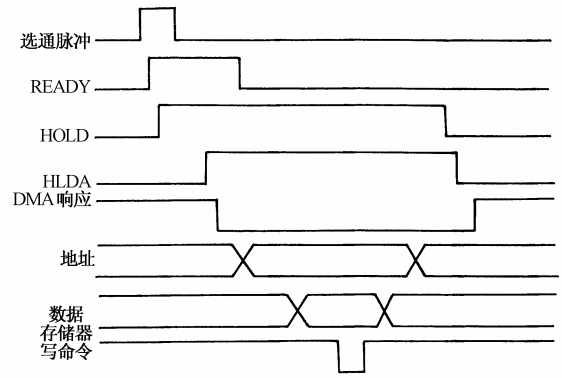


图 4-25　DMA 工作过程的波形图

4.5.3　I/O 处理机方式

采用 DMA 方式后，由于 DMAC 直接控制了数据的传送，在很大程度上提高了数据的传送速度和响应时间。但是，DMAC 只能实现对数据输入/输出传送的控制，而对输入/输出设备的管理和其他操作，诸如信息的变换、装配、拆卸和数码校验等功能操作仍需要由 CPU 来完成，为了使 CPU 完全摆脱管理、控制输入/输出的沉重负担，从 20 世纪 60 年代开始引入了 I/O 处理机的概念，提出了数据传送的 I/O 处理机方式。

在 I/O 处理机方式中，I/O 处理机几乎接管了原来由 CPU 承担的控制输入/输出操作及其他的全部功能。I/O 处理机有自己的指令系统，也能独立地执行程序，能对外设进行控制、对输入/输出过程进行管理，并能完成字与字之间的装配与拆卸、码制的转换、数据块的错误检测和纠错，以及格式变换等操作。同时，I/O 处理机还可以向 CPU 报告外设和外设控制器的状态，对状态进行分析，并对输入/输出系统出现的各种情况进行处理。上述操作都是同 CPU 程序并行执行的，为了使 CPU 的操作与输入/输出操作并行进行，必须使外设在任何时刻都能独立地工作，并且要让外设工作所需要的各种控制命令和定时信号与 CPU 无关，由外设控制器自动地独立形成。

4.6　并行接口

并行接口是微型计算机中最基本、最常用的电路。计算机内部各部件之间以及计算机与常用的外部设备之间，都采用并行接口实现数据传输。用数据锁存器或数据缓冲器就可以构成最简单的并行接口，在实际应用中，大量的通用可编程并行接口芯片被使用。

4.6.1　并行接口及其特点

并行通信就是传送数据的各位分别用一条线同时进行传输，而实现与外设并行通信的接口电路就是并行接口。一般情况下，数据是以计算机的字长为单位的，通常是 8 位、16 位或 32 位，同时在多根传输线上进行传送。

需要强调的是，所谓的并行和串行传送是指 I/O 接口与 I/O 设备或被控对象之间的通信方式，而不是指 I/O 接口与 CPU 之间的通信方式，因为 I/O 接口与 CPU（系统总线）间的数

据传送方式在任何情况下都是并行进行的。

1．并行接口的特点

并行通信是同时在多根传输线上以字节（字）为单位传送数据的。在并行传送方式下，外设（或被控对象）必须通过并行接口与系统总线相连。显然，并行通信传输速度快，信息率高，但它比串行通信所用的传输线多，通常用在传输距离短（几米至几十米），数据传输率要求较高的场合。在实际应用中，凡在 CPU 与外设之间同时需要两位以上信息传送时就要采用并行接口。因此，并行接口适合于外部设备与微机之间进行近距离、大量和快速的信息交换。如打印机接口，A/D、D/A 转换器接口等。

并行接口传送信息，不要求有固定的格式。在并行传送中对同步传送和异步传送没有严格的定义。

2．并行接口的一般结构

同一般的接口电路一样，并行接口也是一组能实现连接 CPU 与外部设备并加以控制的逻辑电路。一个并行接口可以设计为只做输出接口，也可作为输入接口。当然，也有既做输入又做输出的接口。针对后一种情况，可以有两种实现方法：一种方法是利用一个接口中的两个通路，一个为输入通路，另一个为输出通路；另一种方法是用一个双向通路，既可输入又可输出。这些都是具体电路的不同表现形式。

典型的并行接口的结构，从大的方面看，主要由 3 类端口寄存器组成，并行接口结构示意图如图 4-26 所示。

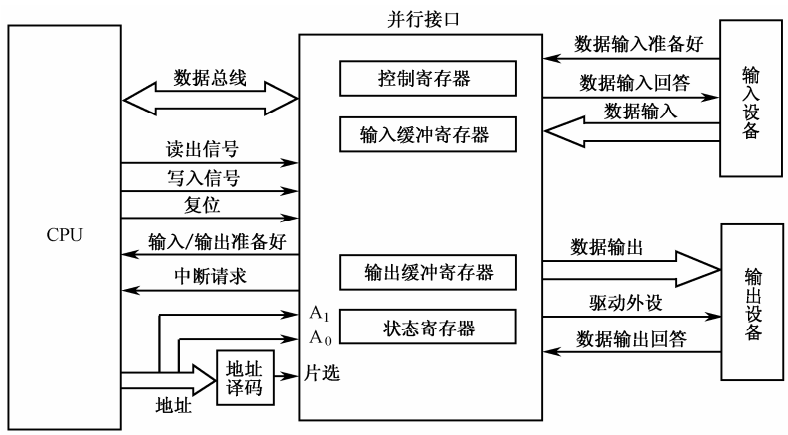


图 4-26　并行接口结构示意图

数据端口寄存器用于保存输入/输出传送中的数据，控制端口寄存器保存由 CPU 发来（对可编程接口而言）的控制信息，状态端口寄存器则保存当前外设的工作状态，提供与外设进行数据交换的应答信号。

3．并行接口的外部信号

并行接口电路的外部信号可分为两部分：与 I/O 设备相连的接口信号和与 CPU 相连的接口信号。与 I/O 设备相关联的接口电路信号有 3 种：

（1）数据信息，用于接口电路与 I/O 设备进行输入或输出的数据；

（2）控制信息，用于接口电路向 I/O 设备提供控制的信号；

（3）状态信息，用于接口电路接受 I/O 设备提供的状态信号。

与 CPU 相关的接口电路信号有：

（1）数据线，用于实现接口电路与 CPU 进行数据交换；

（2）地址线及地址译码信号，用于选择接口电路以及接口电路内部不同的寄存器；

（3）读写控制信号，用于确定 CPU 当前对接口电路的操作性质是读还是写；

（4）中断应答信号，用于实现中断请求和中断响应操作。

4.6.2 简单并行接口

采用数据缓冲器可以构成简单的输入接口；采用数据锁存器可以构成简单的输出接口。对于一些简单的输入/输出设备，如果没有较高要求的应答需要，使用这些简单并行接口是有效的。图 4-27 所示为一简单接口电路图，采用数据输出锁存器连接 LED 显示器，采用数据输入缓冲器连接开关。

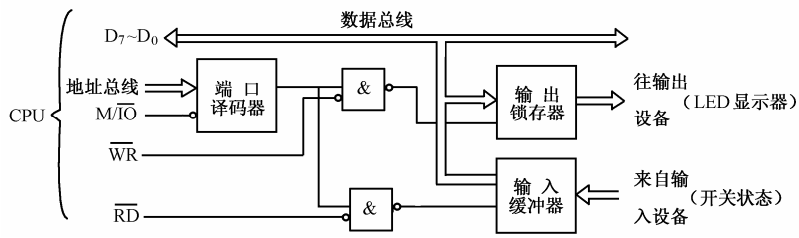


图 4-27 简单并行接口电路

简单外设作为输入设备，输入数据保持时间相对于 CPU 的处理时间要长得多，所以可直接使用三态缓冲器（如 74LS144）和数据总线相连。当执行输入的指令时，读信号 $\overline{RD}$ 有效，选择信号 M/$\overline{IO}$ 处于低电平状态，因而三态缓冲器被接通，使其中早已准备好的输入数据送到数据总线上，再到达 CPU。可见要求 CPU 在执行输入指令时，外设的数据是准备好的，即已经存在于三态缓冲器中，否则就会出错。

当简单外设作为输出设备时，一般都要求接口具有锁存功能，即应设有输出锁存器，以使 CPU 送出的数据在接口电路的输出端保持一些时间。其原因仍然是由于外设的速度比较慢，所以要求 CPU 送到接口的数据能保持和外设动作相适应的时间。如图 4-27 所示，CPU 执行输出指令时，M/$\overline{IO}$ 和 $\overline{WR}$ 信号有效，于是接口中的输出锁存器被选中，CPU 输出的信息经过数据总线送入输出锁存器中，输出锁存器保持这个数据，直到外设将它取走。显然，这里要求 CPU 在执行输出指令时，确信所选中的输出锁存器是空的。即输出时，输出设备已将上次送来的数据取走，接口准备好接收新的数据，否则就会出现错误。

4.6.3 联络信号的作用

联络信号是在接口电路和 I/O 设备之间发生作用的，它可以有效地反映接口电路和 I/O 设备的状态，从而解决 CPU 与 I/O 设备之间的同步问题。上述所涉及的状态信号和应答信号也是一种联络线。下面将对联络信号做进一步的说明。事实上，不同接口电路的联络信号的表示标识可能有所不同，但实质都是一样的。

接口电路中数据寄存器的状态是实现 CPU 与 I/O 设备之间数据传送的同步问题的关键。

数据寄存器有两种状态：满状态和空状态。当数据寄存器中新写入数据时，它处于满状态；当数据寄存器被读取一次时，它处于空状态。

对于输入操作而言，当外设送入一个数据时，数据输入寄存器为满状态；当 CPU 对接口电路的数据输入寄存器读一次后，寄存器的状态为空。

对于输出操作而言，当 CPU 向接口电路的数据输出寄存器写一个数据后，寄存器的状态为满；当外设取走暂存接口电路中的数据以后，数据输出寄存器处于空状态。

1．输入接口的联络信号

对于输入接口，定义数据输入寄存器的两种状态为：

IBF：数据输入寄存器满状态；

IBE：数据输入寄存器空状态。

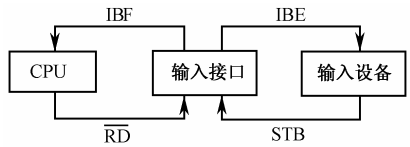


图 4-28　输入接口的联络信号

另外，输入设备通过选通 STB 信号将数据送入输入接口，CPU 通过 $\overline{RD}$ 读控制信号读取数据输入寄存器中的值。CPU、输入接口、输入设备三者的关系如图 4-28 所示。

输入设备在 IBE 即输入寄存器空的状态下通过发 STB 信号将数据送入接口电路；接口电路的输入寄存器状态由空变满，一方面禁止外设新送入其他数据，另一方面供 CPU 读取；CPU 在收到输入接口的 IBF 信号以后，执行 IN 指令，读取输入值，使输入寄存器为空。空状态 IBE 又可以通知输入设备输入新的数据了。

2．输出接口的联络信号

对于输出接口，定义数据输出寄存器的两种状态为：

OBF：数据输出寄存器满状态；

OBE：数据输出寄存器空状态。

另外，输出设备通过应答信号 ACK 告诉接口电路已读取数据，CPU 执行 OUT 指令，向接口电路发 $\overline{WR}$ 信号表示向接口电路中的输出寄存器写入新的数据。

如图 4-29 所示，CPU 通过 $\overline{WR}$ 写信号向数据输出寄存器写数据，并使数据寄存器的状态为满，其 OBF 信号的有效可促使输出设备读取数据。在读取完数据以后，输出设备通过发应答信号 ACK 通知接口电路，此时，接口电路的状态变为空，禁止输出紧接着读取下一个数据，同时 OBE 的有效信号可通知 CPU 再输出下一个数据，使数据输出寄存器的状态再一次为满状态。

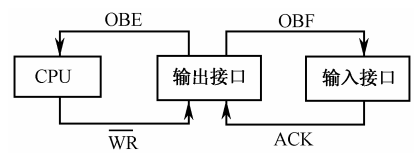


图 4-29　输出接口的联络信号

4.6.4　16/32 位并行接口

与 16/32 位存储器的连接相同，对于 16/32 位的 I/O 接口，16 或 32 位的接口电路数据线分别与系统数据总线相连，而端口则会覆盖 2 个或 4 个地址。以 16 位接口为例，16 位的数据口可分为两个 8 位口，一般低 8 位为偶地址端口，高 8 位为奇地址端口，高 8 位和低 8 位的选择是通过 16 位的 CPU 地址线 A_0 和 BHE 来实现的。当 A_0=0、$\overline{BHE}$ =1 时，选中低 8 位端口；当 A_0=1、$\overline{BHE}$ =0 时，选中高 8 位端口；只有当 A_0=0、$\overline{BHE}$ =0 时，才对 16 位口读

写。

32 位 CPU 一般设有 32 位数据线，与主存和外部设备传送数据时，每次可按 8 位、16 位、24 位或 32 位进行。不同字节的选择由地址 A_{32}~A_2 和字节选择信号 $\overline{BE}_3$ ~ $\overline{BE}_0$ 控制。在硬件连接时，可用 32 位数据线并行连接 4 个 8 位端口，用 $\overline{BE}_3$ ~ $\overline{BE}_0$ 参与端口选择，以确定并行传送数据的字节数，其连接如图 4-30 所示。

如果外部 8 位端口需要连接到 32 位的数据线上，可采用如图 4-31 所示的连接方法，用 $\overline{BE}_3$ ~ $\overline{BE}_0$ 控制 4 个门电路来实现内部 32 位数据线与外部 8 位数据线的连接。

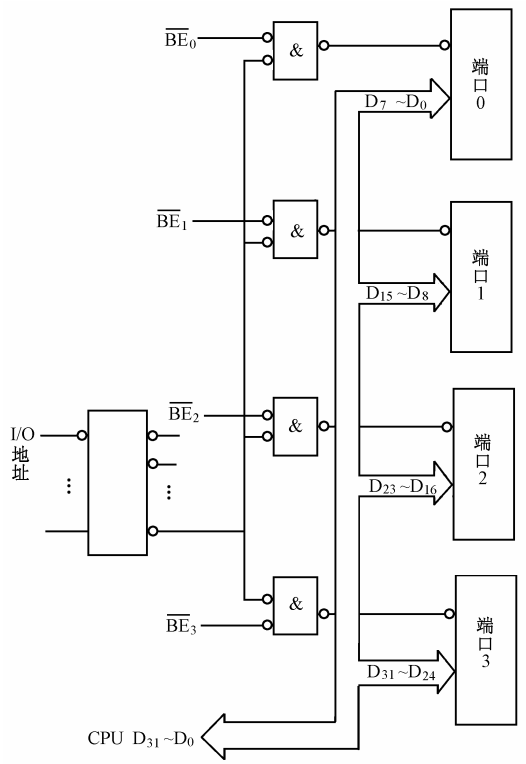


图 4-30　32 位 I/O 端口的连接

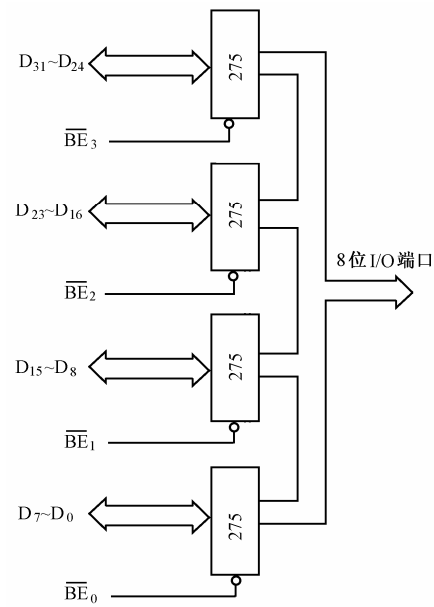


图 4-31　32 位数据线与 8 位数据线的连接

4.7　串行接口

计算机内部之间的信息交换，多采用并行方式，而计算机与计算机之间或计算机与外部设备之间，因多种因素的影响，常采用串行通信。随着计算机应用的普及与性能的提高，大多数计算机都备有各种各样的串行通信接口，这些接口是由相应的器件和线路来实现的。

4.7.1　串行接口及其特点

并行传送的数据是在多条并行传输线上同时由源到达目的地的，而串行传送，其数据是在单条1位宽的传输线上，一位一位地按顺序分时传送的。能够以串行方式传送数据的电路就是串行接口电路。

1．串行通信的特点

与并行传送相比，串行通信有以下明显特点：

（1）从距离上看，并行传送适用于近距离的数据传送，而串行通信适用于远距离传送，可由几米到数千千米。

（2）从速度上看，并行接口的数据传输速度明显比串行接口的传输速度快得多。

（3）从设备、费用角度来看，随着大规模和超大规模集成电路的发展、逻辑器件价格趋低，而通信线路费用趋高，因此对于远距离通信而言，串行通信的费用会低得多。另一方面，串行通信还可以利用现有的电话网络来实现远程通信，从而降低通信费用。显然，串行通信适用于远距离通信。

2．串行通信的传送方向

串行通信时，数据在两个站（如终端或微机）之间进行传送，按传送方向的不同，可分为 3 种传输制式，这就是单工方式（Simplex）、半双工方式（Half-Duplex）和全双工方式（Full-Duplex）。

（1）单工方式。这种方式只允许数据按照一个固定的方向传送。采用该方式时，已经确定了通信两点中的一点为接收端，另一端为发送端，这种确定是不可更改的，如图 4-32（a）所示。在参加通信的 A、B 两站中，A 站只能为接收器，B 站只能为发送器，反之不行。

（2）半双工方式。参加通信的 A、B 两端均具备接收或发送数据的能力。由于 A、B 是由一条信道相连，故在某一特定时刻，A、B 的传输方式是明确的，B 发 A 收或 A 发 B 收。决不允许 A 或 B 在同一时刻既发又收，如图 4-32（b）所示。

（3）全双工方式。全双工方式是用两条信道将 A、B 两端连接起来的，从而克服了单工或半双工方式带来的 A、B 两端不能既发又收的缺点。为了实现全双工传输的功能，A 端和 B 端必须分别具备一套完全独立的接收器和发送器，如图 4-32（c）所示。

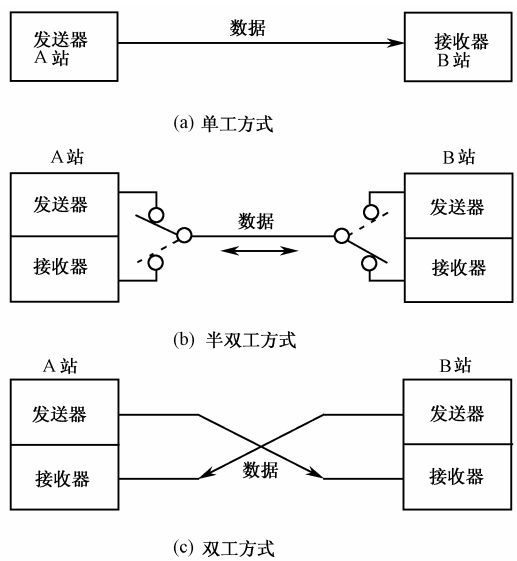


图 4-32　单工、半双工和全双工方式

3．信号的调制与解调

计算机通信传送的是数字信号，如图 4-33（a）所示，它要求传输线的频带很宽。而计算机在远程通信中通常使用电话线传送，不可能有这样宽的频带。如果数据线在这样的传输线上直接传送，经过传输线后，信号必然会发生畸变，如图 4-33（b）所示。因此，在发送端必须采用调制器把数字信号转换成模拟信号，而在接收端又必须用解调器检测发送端送来的模拟信号，并恢复为原来的数字信号，如图 4-33（c）所示。

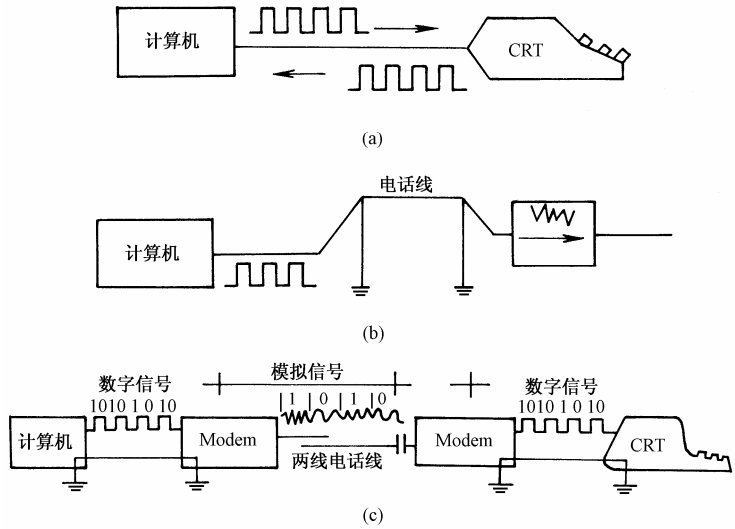


图 4-33　远程通信的信号调制与解调

Modem 即调制解调器，是计算机远程通信中的一种辅助设备，其功能是将发送的数字信号调制成模拟信号，同时又将接收的模拟信号解调成数字信号，使微型计算机可利用电话线实现远程通信。

4.7.2　串行通信的基本方式

在串行通信中，由于信息传输只占用一根传输线，因此这根线既作为数据线又作为联络线，也就是说要在一根数据线上既传送数据信息，又传送联络控制信息，这就是串行通信的基本特点。正因如此，在处理串行通信时，不仅需要考虑数据简单的输入/输出，更要考虑数据是如何有效传输的。实现数据的有效传输，除了保证经远距离传输信息不发生歧变以外，还要做到通过一根传输线，发送方可以明确地告诉接收方什么时候是有效数据，什么时候是无效数据。因此，串行通信有双方一系列严格、明确的规定。

1．同步通信与异步通信

由于在串行通信中，只有一根通信线用于传送信息。因此，在这根传输线上，应既能向接收方传送数据，又能向接收方表达数据的有效性，即需要及时地告知对方传送的有效数据什么时候开始，什么时候结束，以便接收方做相应的操作。这就需要收发双方有严格的通信格式规定，在串行通信中，有两种基本的通信方式：异步通信和同步通信。

（1）异步通信。异步通信的传输格式如图 4-34 所示。每个字符为一帧信息，由 4 部分组

成：

起始位：1 位，低电平表示；

数据位：5~8 位，低位在前，高位在后；

校验位：1 位，对数据进行校验，奇校验或偶校验；

停止位：1~2 位，高电平表示。

异步通信的特点是一个字符一个字符地传输，并且传送一个字符总是以起始位开始，以停止位结束，字符之间没有固定的时间间隔要求。

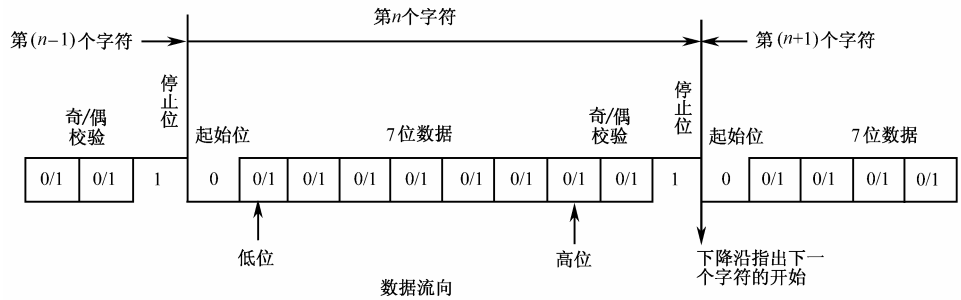


图 4-34　异步通信数据格式

例如，传送一个七位的 ASCⅡ码字符，再加上一个起始位、一个奇校验位、一个停止位，这样，组成一帧共 10 位。

如图 4-35 所示为字符“E”的 ASCⅡ码波形。

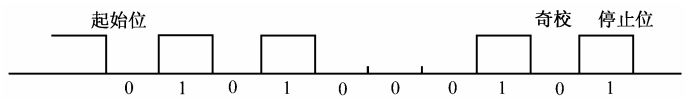


图 4-35　字符“E”的传送波形

其中，第 1 位为低电平，表示起始位；第 2~8 位为数据位，表示传送的是字符“E”的 ASCⅡ码 45H，这是一个由低到高表示的二进制值 1000101；第 9 位为奇校验位为 0，以保证数据位加校验位“1”的个数为奇数个；最后 1 位高电平表示停止位。

作为接收方，通过第一个低电平判断数据接收工作的开始，通过最后一个高电平知道数据传送的结束，通过校验位判断数据在传送过程中是否有误。

（2）同步通信。在异步通信中，各字符之间是异步的，而字符内部各位之间是同步的，即每个字符出现在数据流中的相对时间是随机的，接收端预先并不知道，而每个字符一开始发送，收发双方就以预先约定的时钟速率传送各位。

因此，在异步通信中，每个字符要用起始位和停止位作为开始和结束的标志，这样占用了一些时间，因而在数据块传送时，为了提高速度，就要设法去掉这些标志而采用同步传送。同步传送必须在数据块开始时用同步字符来指明，如图 4-36 所示。

图 4-36　同步通信数据格式

同步传送的基本特点：在通信的数据流中，字符与字符之间，字符内部的位与位之间都是同步的，接收端与发送端必须以同一时钟来实现双方的同步，因此硬件电路比较复杂，但传送速度高于异步传送速度。通常同步传送用于计算机之间的通信。

因此，与同步通信相比较，异步通信的效率相对比较低，但由于在通信过程中不需要传送相应的时钟信号，而是通过每一帧数据起始位和停止位来实现同步的，因此对支持通信的电路要求比较低，成为一种很实用的通信方式。

2．波特率

串行通信的收发双方必须使用相同的数据格式规定。例如，发送方以异步方式发送数据，且规定具体格式为 1 位起始位、7 位数据位、1 位奇校位和 1 位停止位。那么，接收方也要以相同的数据格式规定才能正确接收数据。然而，仅有这样的规定是不够的，因为当接收方收到一个低电平后，还需判断是 1 位低电平还是 2 位低电平或者更多。

如何才能在串行通信中规定 1 位信息的量呢？事实上，这与数据传输的速率有关。在串行通信过程中，收发双方除了规定相同的数据格式以外，还必须规定数据传输的速率。

波特率是指单位时间内传送二进制数据的位数，以 b/s 为单位，它是衡量串行数据传送速度的重要指标和参数。

例如，数据传送速率是 120B/s，而每个字符格式规定含有 10 位二进制数据（1 位起始位、7 位数据位、1 位校验位、1 位停止位），则传送的波特率为：

10×120=1 200b/s=1 200 波特

相应地，每个数据位的传送时间 T_d 为波特率的倒数：

T_d=1/1 200=0.833ms

最常用的标准波特率是 110、300、1 200、2 400、4 800、9 600、19 200 波特等。通常，CRT 终端能处理 9 600 波特的传输，打印机终端速度较慢，点阵打印机一般也能以 2 400 波特的速度来接收信号。

串行接口或终端直接传送串行信息位流的最大距离（波形不要发生畸变）与传输速率及传输线的电气特性有关，传输距离是随传输速率的增加而减少的。实际运用中，对远距离传送，一般都需加入通信设备调制解调器 Modem。

3．通信规程

随着计算机网络技术的不断发展和数据通信应用的日益普及，为了在国际间或范围广的地区内实现数据通信，有必要对数据编码、数据传输速度、同步方式、传输控制步骤、出错控制方式、通信报文格式及控制字符定义等问题做出统一的规定。这样，通信双方就如何交换信息所建立的一些规定和过程称为数据控制规程，或称传输控制规程。

目前采用的控制规程可分为两类：异步通信控制规程和同步通信控制规程。同步通信控制规程又可分为面向字符型（Character-Oriented）和面向比特型（Bit-Oriented）两大类。目前，对于每一类通信规程，都有相应的大规模集成电路的接口芯片实现。

（1）异步通信控制规程。异步通信控制规程的异步含义是发送器和接收器不共享共用的同步信号，也不在数据中传送同步信号。为了确认被传送的脉冲序列从某处到某处表示 1 个字符，其定时控制方法是在 1 个字符的首尾放置起始符号和停止符号，供接收端用起始符号和停止符号判断 1 个字符，所以这种通信控制规程有时也称起止同步控制方式。图 4-35 所示

为异步串行通信所使用的位格式。当发送器不工作或处于空闲时，线路处于连续的“1”状态（高电平的“传号”状态）。发送器可以在任一时刻通过传送起始位，即在线路上放置1位（传送1位信号所需时间）的“0”状态（低电平的“空号”状态），启动一个字符的传送，接着传送数据位（低位在前，高位在后），其后是可供选择的奇数或偶数检查位。最后，发送器维持线路处于“传号”状态，其维持时间可以为1、1.5或2位，它们称为停止位。

从起始位到停止位的时间周期称为1帧（FRAME）。在停止位以后，如果有另一个字符要传送，发送器可以立即传送1个新的起始位，否则发送器一直维持线路处在“传号”状态。自最后1个停止位起，发送器可以在任一时刻传送1个新的起始位。

在异步通信中是由数据时钟控制数据的发送和接收的，为了保证数据的正确传送，还必须保证发送和接收的双方有相同的数据传送速率。

（2）面向字符通信控制规程。面向字符型的控制规程始于1960年，它的特点是规定10字符作为传送控制专用，信息长度为8的整数倍，传输速率为200~4 800b/s。

这种协议的典型代表是IBM公司的二进制同步通信协议（BSC），一次传送由若干个字符组成的数据块，而不是只传送一个字符。面向字符的同步通信数据格式可采用单同步。双同步及外同步3种，如图4-37所示。

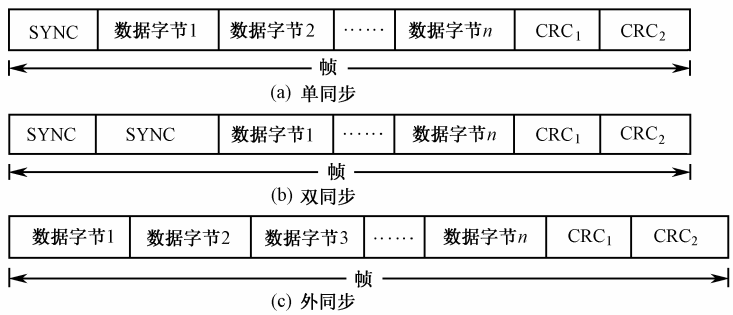


图4-37　面向字符型的数据格式

单同步是指在传送数据之前先传送一个同步字符“SYNC”，双同步则先传送两个同步字符“SYNC”。接收端检测到该同步字符，使接收方与发送端实现同步。当每一帧信息结束时均以两个字节的循环校验码CRC结束。外同步的格式中则无同步字符，而用一条专用控制线来传送同步字符。

（3）面向比特通信控制规程。面向比特通信控制规程的概念是由IBM公司在1969年提出的。它的特点是没有采用传送控制字符，而是采用某些位组合作为控制，其信息长度可变，传输速率在2 400波特以上。这一类中最有代表性的规程是IBM的同步数据链路控制规程SDLC（Synchronous Data Link Control），国际标准化组织ISO的高级数据链路控制规程HDLC（High Level Data Link Control），美国国家标准协会ANSI的先进数据通信规程ADCCP（Advanced Data Communications Control Procedure）。

SDLC是面向比特型的数据以帧为单位的数据传输规程。每帧由6个部分组成：第一部分是开始标志“7EH”（01111110）；第二部分是一个字节的地址场；第三部分是一个字节的控制场；第四部分是需要传送的数据，数据都是位的集合；第五部分是两个字节的循环校验CRC；最后部分又是“7EH”，作为结束标志。其数据格式如图4-38所示。

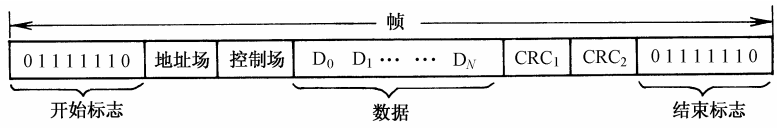


图 4-38　面向比特型的数据格式

4.7.3　串行接口电路的结构与功能

串行接口功能的实现可以由硬件实现，也可以由软件实现。就目前而言，绝大部分的串行接口都是由硬件来实现的，CPU 只要通过对串行接口电路的读写就可以实现串行数据的输入与输出。

1．串行接口的基本任务

串行接口电路是完成串行数据收发的具体接口电路。根据串行通信的要求，串行接口电路应具有如下功能：

（1）进行串、并行转换。串行传送数据是一位一位依次顺序传送的，而计算机处理数据是并行的。所以，当数据由计算机送至数据终端时，需要把并行数据转换为串行数据再传送。当计算机接收终端传来数据时，首先需要把串行数据转换为并行数据才能送入计算机进行处理。

（2）实行串行数据格式化。从 CPU 来的并行数据转换成串行数据后，接口电路要实现不同通信方式下的数据格式化。异步方式下，发送或接收数据时自动生成或去掉启动停止位；面向字符的同步方式下，接口所做的数据格式化则主要是在数据块前面加入同步字符。

（3）可靠性检验。为确保接收/发送数据的可靠性，在发送时，接口电路自动生成奇偶校验位或其他校验码；在接收时，接口电路检查字符的奇偶校验位或其他校验码，以确定是否发生传送错误。

2．串行接口的一般结构

串行接口有许多种类，典型的串行接口如图 4-39 所示，它包括 3 个部分：发送器、接收器和控制器。发送器用来把并行码转换为串行码；接收器用来把串行码转换为并行码；而控制器用来接收 CPU 的控制信号，执行 CPU 要求的操作，并输出状态信息和控制信息。

串行接口电路的发送器和接收器的核心电路都是移位寄存器。当输入时，由 RxD 来的串行数据先进入移位寄存器，然后并行输入给缓冲器，由数据总线输入至 CPU。在发送时，由 CPU 来的并行数据由缓冲器接收，然后送至移位寄存器，由 TxD 一位一位串行输出。

串行接口电路还有一些控制和状态信息。以异步通信为例，在接口电路工作时，接收器部分始终监视着 RxD 线，当发现一个起始位时，就开始了一个新的字符接收过程。

因为是异步传送，接口电路是用外部时钟与接收数据同步的。外部时钟的周期 T_c 和数据位的周期 T_d 之间的关系为：

$$T_c = T_d / K$$

其中，K 称为波特率因子，或称为波特率系数，通常取值为 $K=16$ 或 64。

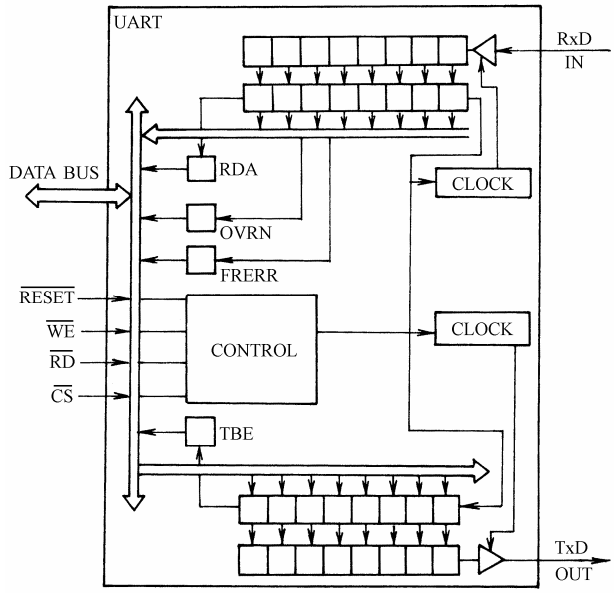


图 4-39　串行接口的结构

若 $K=16$，在每个时钟脉冲的上升沿采样接收数据线，若发现了第一个“0”，即表示起始位的开始，以后又连续采样到 8 个“0”，则确定为起始位而不是干扰信号，以后每隔 16 个时钟脉冲采样一次数据线，作为输入数据，如图 4-40 所示。

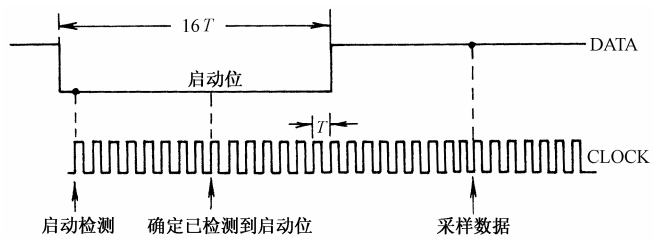


图 4-40　串行电路对数据的采样

3．接口电路的差错控制

串行数据在传输过程中，由于干扰而引起误码是不可避免的，这直接影响通信系统的可靠性。为了保证高效而无差错的传送数据，串行通信中的差错控制是一个十分重要的环节，通信中差错控制能力是衡量一个通信系统的重要内容。

差错控制可分为两部分：检错（发现错误）和纠错（纠正错误）。检错实质上是对传送的数据进行校验。在数据传送过程中，发送方根据发送的数据产生校验码，接收方根据收到的数据和校验码来判断传送的数据是否正确。常用的校验方法有：奇/偶校验和 CRC 循环冗余校验。

奇/偶校验主要用于对一个字符的传送进行校验。

发送时，在每个字符的最高位之后，都附加一个奇/偶校验位，这个校验位本身有可能是“1”或“0”。若是奇校验，则加上这个校验位之后，使所发送的任何字符中的“1”的个数始终为奇数；若是偶校验，则加上校验位之后，所发送的任何字符中的“1”的个数始终为偶

数。

在接收时，检查所接收的字符连同这个奇/偶校验位，其为“1”的个数是否符合规定（奇校验或偶校验），若不符合规定就置出错标志，供 CPU 查询及处理。

在串行接口电路中，通常设置 3 种出错校验：

（1）奇/偶校验错。接收方在进行奇/偶校验时发现的错误，即“1”的个数不符合规定，则为奇/偶校验错。

（2）帧格式错。接收方在收到字符时发现字符格式不符合规定，如缺少停止位，则说明帧格式出错。

（3）溢出错。接收方在接收数据时，将串行数据转换为并行数据供 CPU 读取。若接收方已经接收了第二个字符，但 CPU 还没有将前一个数据取走，于是出现数据丢失，这就是溢出错。

本 章 小 结

输入/输出接口是主机与外部设备进行数据交换的通道，主机可以采用不同的接口电路，以不同的数据通信形式，与种类繁多的外部设备相连接。本章主要介绍了 3 部分内容：I/O 接口的基本概念、主机与外部数据交换的方式以及最基本的接口形式。

作为 I/O 接口的基本概念，涉及接口电路的功能、结构、需要传送的信息类型以及基本输入/输出接口电路的实现。

主机与外部交换信息的形式，在程序控制下实现的有：无条件传送、程序查询、中断传送；不受程序控制的有 DMA 传送方式和 I/O 处理机方式。程序查询方式是由 CPU 不断访问 I/O 接口电路的状态，来决定是否进行数据传送；中断方式是外设根据自己的需要主动提出数据交换要求，因此，CPU 要涉及中断源识别、中断优先权处理、中断及返回等一系列的操作；DMA 传送是在 DAMC 的控制下摆脱 CPU 的干预，运行系统总线直接进行数据传送，数据传送速率高。不同的应用要求，可选择不同的数据传送方式。

CPU 与 I／O 信息交换的接口按其通信方式可分为两种：并行接口和串行接口。

并行接口适合近距离的高速数据传送。简单的并行接口解决数据的基本输入和输出，包括 16 位和 32 位的并行接口。对于 CPU 与外设之间在速度上的协调问题，则需要通过状态信号参与，状态信号作为 CPU 与外设的联络者，影响着双方的数据交换动作。

串行接口比较适合于远距离数据传送。串行接口涉及串行通信的基本概念、串行接口电路的基本结构与功能以及实现串行通信所需遵循的一系列数据格式规定及通信规程。

习 题 4

4.1 CPU 与外设交换数据时，为什么要通过 I/O 接口进行？I/O 接口电路有哪些主要功能？

4.2 I/O 接口的寻址方式有几种？各有什么特点？

4.3 通常，在 I/O 接口电路设计中，输入需要缓冲，输出需要锁存，为什么？

4.4 微型计算机对 I/O 端口编址时采用哪两种方法，分析其各自的特点。

4.5 某一个微机系统中，有 8 块 I/O 接口芯片，每个芯片占有 8 个端口地址，若起始地址为 9000H，8 块芯片的地址连续分布，用 74LS138 做译码器，试画出端口译码电路，并说明每块芯片的端口地址范围。

4.6　如果有多个 I/O 设备需要 CPU 访问，输入或输出数据，CPU 可否采用程序查询方式来实现，如何实现？

4.7　CPU 响应中断的条件是什么？简述中断处理过程。

4.8　什么是中断向量、中断类型号？识别中断源通常有哪些方法？

4.9　针对 80286 CPU，试说明当有中断请求后，CPU 是如何找到相应的中断服务程序入口地址的。

4.10　当有多个中断源发中断请求时，中断系统是如何解决优先权问题的？

4.11　说明常用的几种 DMA 操作的基本方法。

4.12　说明 DMA 控制器的作用，并进一步说明在 DMAC 的控制下实现 DMA 传送的操作过程。

4.13　分析程序查询传送方式、中断传送方式、DMA 传送方式这 3 种数据传送方式，阐述其各自的特点及适用范围。

4.14　比较并行接口和串行接口，分析各自的特点和适用的场合。

4.15　一般并行接口内部由哪些部件组成，各自功能如何？

4.16　在数据过程中，“ 满 ” 状态和 “ 空 ” 状态分别反映接口电路怎样的一种工作状态？

4.17　试为一个 16 位的 CPU，如 80286，设计一个 16 位的简单输入接口。

4.18　在串行异步通信中，每一帧数据由哪些信息位组成？串行接收器在接收数据过程中是如何识别起始位的？

4.19　若某一终端以 2 400 波特的速率发送异步串行数据，发送 1 位需要多少时间？假设一个字符包含 7 个数据位、1 个奇/偶校验位、1 个停止位，发送 1 个字符需要多少时间？

4.20　在串行通信中，什么叫单工、半双工、全双工工作方式？

4.21　起止式通信协议的特点及帧数据格式是怎样的？面向字符和面向比特通信协议有什么不同？各自的帧数据格式是怎样的？

4.22　什么叫波特率？什么叫波特率因子？常用的波特率有哪些？

第 5 章　可编程接口芯片及应用

CPU 要同外设交换信息，必须通过接口电路，在接口电路中多数具有如下电路单元：

（1）输入/输出数据锁存器和缓冲器，用以解决 CPU 与外设之间速度不匹配的矛盾，以及起隔离和缓冲作用；

（2）控制命令和状态寄存器，以存放 CPU 对外设的控制命令以及外设的状态信息；

（3）地址译码器，用来选择接口电路中的不同端口；

（4）读/写控制逻辑；

（5）中断控制逻辑。

随着大规模集成电路技术的迅速发展，微机系统中 CPU 与外设之间的接口电路已由早期的中小规模集成电路芯片组成的逻辑电路板发展为以大规模集成电路芯片为主的接口芯片。根据其功能，这些接口电路芯片有的是通用的，有的是专用的；有的用于 I／O 的数据交换，有的是为了完善与扩展 CPU 的功能。

5.1　可编程接口芯片概述

5.1.1　可编程接口芯片及其特点

微型计算机与各种外设或控制对象间的联系及信息交换都必须通过 I/O 接口电路来实现。所以 I/O 接口电路是微机应用系统中必不可少的重要组成部分。用微机组成一个实际的应用系统，除微机本身外，关键的技术就是接口技术，任何一个微机应用系统的研制和开发，关键是微机接口的研制和设计。

众所周知，在计算机系统的功能实现上，常常采用硬件和软件相结合的办法。一般而言，硬件实现的特点是速度快，但硬件开销大，灵活性差；用软件实现的特点是灵活性好，但速度慢。针对一个具有多种功能的 I/O 接口，既要保持硬件的快速性又要具有软件的灵活性，在微机接口的研制和设计中，常常用硬件来实现共同的基本部分，即接口电路；而用软件来实现特殊部分，即控制接口电路按要求工作的驱动程序。这些输入/输出设备的接口，通常设计成完全独立的通用接口芯片，以满足各种类型的外设与微机接口设计的需要。

所谓可编程实际上就是具有可选择性，可编程接口是指接口的工作方式和功能可以用编程的方法加以改变。例如，选择芯片中的哪一个或哪几个数据端口与外设连接，选择端口中哪一位或哪几位做输入，哪一位或哪几位做输出，选择端口与 CPU 之间采用哪种方式传送数据等，均可由用户在程序中写入方式字或控制字来进行指定，因此，它们具有广泛的适应性和很高的灵活性。这种可用软件的方法改变接口的工作方式及功能的接口芯片称为可编程接口芯片。

随着大规模集成电路技术的发展，现已生产了各种各样的通用可编程接口芯片。由于可编程芯片具有广泛的适应性及很高的灵活性，因此，在微机系统中得到了广泛的应用。

5.1.2 可编程接口芯片的分类

接口电路可以简单到只由一块中小规模的集成电路或一块大规模通用集成电路芯片组成，也可能复杂到不亚于系统主机板（如网络接口卡），其类型可谓多种多样，但其核心部分往往是一块或数块大规模集成电路芯片，这类芯片常被称为“接口芯片”。接口芯片按功能可分为以下 3 种：

（1）通用接口芯片：可支持通用的输入/输出及控制接口芯片。它适用于大部分外部设备，在某些专用的接口电路中也会用到它们。例如并行接口芯片 Intel 8212、Intel 8255A、Z80PIO 等，串行接口芯片 INS8250、Intel 8251、Z80SIO 等。

（2）面向微机系统的专用接口芯片：它们与 CPU 和系统配套使用，以增强其总体性能。如用来扩展系统中断能力的中断控制器 Intel 8259A，用来支持 DMA 数据高速传送的 DMA 控制器 Intel 8237/8257、Z80DMA；用来为系统提供时基信号的定时/计数器 Intel 8253/8254 等。

（3）面向外设的专用接口芯片：这类芯片一般针对某种外设而设计，仅用于某些特定的外设接口。例如 CRT 控制器 MC6845、Intel 8275 等可支持显示接口电路；软盘控制器μPD765、FD8271/8272 等可支持软盘驱动器接口电路；键盘/显示器接口芯片 Intel 8279 可支持简易键盘和数码显示器。

当前，许多接口电路芯片都具有可编程的特点，它们被称为可编程接口芯片。可编程（Programmable）的意思是指接口芯片的功能和工作方式可通过程序来进行设定。这些芯片往往具有多种功能和工作方式，可以通过程序来选取其中的一种，有的芯片还可选定引脚信号的有效方式。为设定芯片工作方式而编写的程序段一般被称为该接口芯片的初始化程序段。

由于可编程的芯片具有多种工作方式和内部资源，可以通过编程来加以设置和选用，并且还可以在系统运行的过程中随时加以改变，这就大大简化了接口的设计，也为灵活运用接口开辟了很大的空间。

显然，在讨论这类接口芯片的应用时，除了进行物理连接之外，还要考虑接口软件的编写。所谓接口软件是指管理、控制、驱动外设的程序，负责在外设和系统间进行信息交换。用户在编写这类程序时，可利用系统提供的软件资源，并处理好它与系统的关系，使它与监控程序或操作系统相协调。

随着微型计算机应用越来越广泛，外围设备的种类也越来越多，各类设备的组成和工作原理差异很大，与微机之间的连接和传输数据的方式也不相同。特别是当微机应用在数据采集和处理系统及过程控制系统时，与微机连接的常常是一些专用设备，对数据传输和处理的实时性要求很高。所以，微型计算机的输入输出是最具有多样性和复杂性的问题。为适应微型计算机广泛应用的需要，目前社会上有各种类型的 LSI/VLSI 可编程通用接口电路芯片。

可编程接口芯片根据数据传送方式的不同，可分为可编程并行接口芯片和可编程串行接口芯片两种。并行接口是在多根数据线上，以数据字节（字）为单位与输入/输出设备或被控对象传送信息的，如打印机接口，A/D、D/A 转换器接口，IEEE-488 接口，开关量接口，控制设备接口等；串行接口则是在一根线上以 1 位数据位为单位与 I/O 设备或通信设备传送信息，如 CRT 显示器、电传机及调制解调器接口等。

微机与外设交换信息都必须通过接口电路来实现，大规模集成电路的发展和应用，采用了各种各样通用的可编程的接口电路芯片，因此，学会典型的通用接口芯片的工作原理和使用方法是掌握微机接口技术的基础。

5.2 可编程并行接口芯片8255A

对于各种型号的 CPU 都有与其配套的并行接口芯片，如 Intel 公司生产的 8255A、Zilog 公司的 Z-80、Motorola 公司的 MC6820 等。它们的功能虽有差异，但工作原理基本相同。本节着重讨论 Intel 8255A 芯片。

5.2.1 8255A 的内部结构与引脚功能

可编程并行接口芯片 8255A 有如下基本特性与功能：

（1）8255A 是一个具有 3 个 8 位数据口（即 A 口、B 口、C 口，其中 C 口还可作为两个 4 位口来使用）的并行输入/输出接口芯片，它为 Intel 系列微机的 CPU 与外部设备提供了 TTL 电平兼容并行接口。三个数据口均可用软件来设置成输入口或输出口，并且与外设相连。C 口具有按位置位/复位的功能，为按位控制提供了强有力的支持。

（2）8255A 具有 3 种工作方式，即方式 0、方式 1 和方式 2。可适应 CPU 与外设间的多种数据传送方式，如查询方式和中断方式等，以满足用户的各种应用要求。

（3）8255A 具有两条功能强、内容丰富的控制命令（方式字和控制字），为用户根据外界条件（I/O 设备需要哪些信号线以及它能提供哪些状态线）来使用 8255A 构成多种接口电路和提供灵活方便的编程环境。8255A 执行命令过程中和执行命令完毕之后，所产生的状态可保留在状态字中以便查询。

（4）8255A 的 C 口是一个特殊的端口，除作为数据口外，当工作在方式 1 和方式 2 时，利用对 C 口的按位控制可为 A、B 口提供专门的联络控制信号；在 CPU 读取 8255A 的状态时，C 口可作为方式 1 和方式 2 的状态字。

（5）8255A 芯片内部主要由控制寄存器、状态寄存器和数据寄存器组成，因此以后的编程主要是对这 3 类寄存器进行访问。

1．8255A 的内部结构

8255A 内部结构方框如图 5-1 所示，它由 3 部分组成：外设接口部分（通道 A、B、C）；内部逻辑部分（A 组和 B 组控制电路）和 CPU 接口部分（数据总线缓冲器、读/写控制逻辑）。

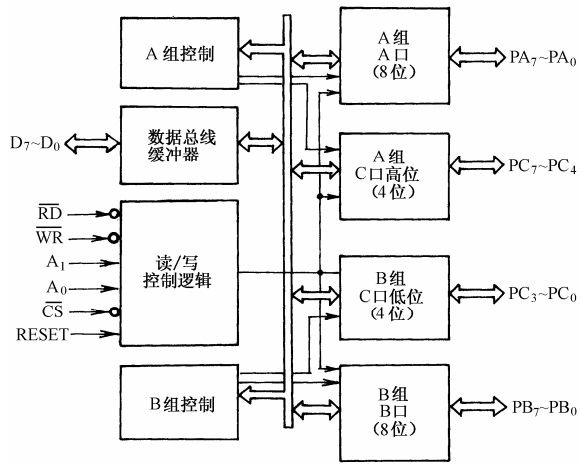


图 5-1　8255A 内部结构方框图

（1）外设接口部分。8255A 有 3 个 8 位数据通道 A、B、C，这 3 个通道在功能上各有特点，都可编程设置为输入或输出。

① 通道 A 具有一个 8 位数据输入锁存器和一个 8 位数据输出锁存器/缓冲器用来传送数据。作为输入端口或输出端口时数据均受到锁存。

② 通道 B 具有一个 8 位数据输入缓冲器和一个 8 位数据输出锁存器/缓冲器，也用来传送数据。用做输入端口时不会对数据进行锁存，而用做输出端口时数据受到锁存。

③ 通道 C 具有一个 8 位数据输入缓冲器和一个 8 位数据输出锁存器/缓冲器。一般作为控制或状态信息端口，可分成两个 4 位端口（高位口和低位口），可按位控制，分别和 A 通道、B 通道配合使用，用做输出控制信号和输入状态信号。当 CPU 与外设连接不需要联络控制线时，C 通道可以和 A、B 通道一样作为输入或输出的数据通道。

（2）内部逻辑。

① 内部逻辑主要用于接收来自内部总线上的控制字和接收来自读写控制逻辑电路的读/写命令。

② 有 A、B 两组控制电路。每组控制电路从读/写控制逻辑接收各种命令，从内部数据总线接收控制字并发出相应命令到各自的通道，控制各个通道的工作方式。还可根据 CPU 的命令字对通道 C 的每一位按位置位或复位。

A 组控制电路控制通道 A 和通道 C 的高 4 位（PC_7～PC_4）。

B 组控制电路控制通道 B 和通道 C 的低 4 位（PC_3～PC_0）。

（3）CPU 接口部分。这部分包括数据总线缓冲器和读/写控制逻辑。数据总线缓冲器是 8255A 与 CPU 数据总线的接口，是 8 位双向三态缓冲器，由读/写控制逻辑实施三态控制。所有数据的输入和输出，以及 CPU 写入 8255A 的控制字、从 8255A 读出的外设状态信息，都是通过这个缓冲器传送的。

读/写控制逻辑接收有关控制信号 RESET、$\overline{WR}$、$\overline{RD}$ 和地址总线的低位部分、地址译码信号 $\overline{CS}$，对 A 组和 B 组控制电路实施控制，管理内部和外部数据、状态或控制字的传送。

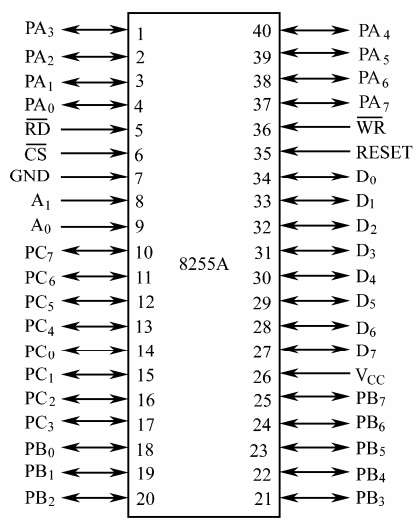


图 5-2　8255A 芯片引脚图

2．8255A 芯片的引脚及其功能

8255A 是一个单+5V 电源供电、40 个引脚的双列直插式芯片，图 5-2 是 8255A 的芯片引脚信号，其引脚信号分为：

（1）和外设的连接。

PA_7～PA_0：A 口数据信号线，双向，三态引脚。

PB_7～PB_0：B 口数据信号线，双向，三态引脚。

PC_7～PC_0：C 口数据信号线，双向，三态引脚。

（2）面向系统总线和 CPU 的连接。

RESET：复位信号，高电平有效。当 RESET 信号到来时，所有内部寄存器都被清除，同时 3 个数据端口被自动置为输入端口。

D_7～D_0：它们是 8255A 的数据线，和系统总线相连，用来传送数据和控制字。

$\overline{CS}$：片选信号，低电平有效。即当$\overline{CS}$端为低电平时，8255A 被选中。只有当$\overline{CS}$有效时，CPU 才能对 8255A 进行读写操作。

$\overline{RD}$：读信号，低电平有效。当$\overline{RD}$有效时，CPU 可以从 8255A 中读取数据。

$\overline{WR}$：写信号，低电平有效。当$\overline{WR}$有效时，CPU 可以往 8255A 中写入控制字或数据。

A_1、A_0：端口选择信号。8255A 内部有 3 个数据端口（I/O 端口）和 1 个控制端口，共 4 个端口。通过地址线 A_1A_0 寻址。规定当 A_1A_0 为 00 时，选中 A 端口；为 01 时，选中 B 端口；为 10 时，选中 C 端口；为 11 时，选中控制口。

（3）电源和地。

V_{CC}：+5V 电源；

GND：地线。

5.2.2 8255A 的编程控制

8255A 可以通过指令往控制端口中设置控制字来决定它的工作方式。8255A 的控制字分为两种：工作方式控制字（特征位 D_7=1，用于指定 3 个数据端口做输入/输出以及选择工作方式）和 C 口置位/复位控制字（特征位 D_7=0，用于指定 C 口的某一位置 1 或置 0）。由于两个控制字使用同一个端口地址写入，所以将最高位 D_7 作为标志位，用于区别这两种控制字。

1．工作方式的控制字

图 5-3 列出了 8255A 工作方式控制字各位的作用。最高位 D_7 必须为“1”，是方式控制字的特征位。当用输出指令将方式控制字写入 8255A 后，它被分别存于 A、B 两组控制寄存器中。

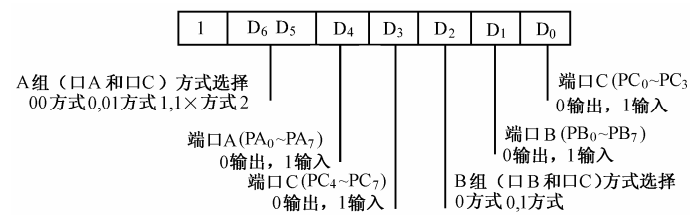


图 5-3　8255A 的工作方式控制字

D_6～D_3 控制 A 组（A 口及 C 口高 4 位）的工作方式及输入或输出；

D_2～D_0 控制 B 组（B 口及 C 口低 4 位）的工作方式及输入或输出。

例如，设 8255A 的控制端口地址为 00E6H，要把 A 口指定为方式 1 输入，PC_7～PC_4 定为输出，B 口指定为方式 0 输出，PC_3～PC_0 定为输入，则方式控制字应是 10110001B 或 B1H。若将此控制字的内容写入 8255A 的控制寄存器，即实现了对 8255A 工作方式的指定（或者说完成了对 8255A 的初始化），初始化的程序段为：

```
MOV    DX,    01E6H    ；控制端口地址 01E6H
MOV    AL,    0B1H     ；方式控制字
OUT    DX,    AL       ；送控制端口
```

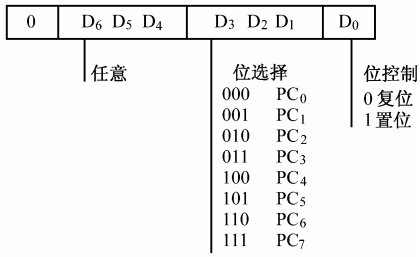


图 5-4　C 口的位控字

2．C 口的位控字

8255A 的 C 口具有位控功能，即允许 CPU 用输出指令单独对 C 口的某一位写“1”或“0”，C 口的位控字的格式如图 5-4 所示。这是通过向 8255A 的控制寄存器写入（注意不是直接对 C 口写入）一个位控制字来实现的。最高位 D_7 必须为“0”，是 C 口置位/复位控制字的特征位。D_0 位决定了是置“1”还是置“0”操作；D_3～D_1 位决定了对 C 口中的哪一位进行操作。

5.2.3　8255A 的三种工作方式

8255A 有三种工作方式，由方式控制字决定。A 口可以工作在方式 0、方式 1 和方式 2 三种方式下，而 B 口只能工作在方式 0 和方式 1 两种方式下。

1．方式 0

方式 0 是一种基本输入或输出方式，该方式适用于较简单的输入/输出场合，这种输入/输出可以不使用联络线，也可以由软件对其他端口的输入/输出控制来构造联络线，CPU 可以随时用输入/输出指令对指定端口进行读/写操作。该方式的特点是：

（1）使 8255A 分成彼此独立的两个 8 位端口（A 口、B 口）和两个 4 位端口（C 口高 4 位和低 4 位），4 个端口的输入/输出可有 16 种不同的组态，可适用于各种不同的应用场合。

（2）方式 0 规定输出有锁存能力，而输入数据不被锁存。

（3）方式 0 是单向的 I/O，即一次初始化指定了输入或输出，且不能改变；若改变，则须重新初始化。不能指定同一端口同时既作为输入又作为输出。

（4）这种方式下，无固定的 I/O 联络信号，联络信号线可由用户自行安排。这种方式只能用于无条件传送和查询传送，不能实现中断传送。

图 5-5 是一个在方式 0 下利用 C 口作为联络信号的 I/O 电路。此例中将 8255A 编程为如下状态：A 口输出，B 口输入，C 口高 4 位为输入，现仅用 PC_7、PC_6 两位输入外设的状态；C 口低 4 位为输出，用 PC_1、PC_0 输出选通及清除信号。对 8255A 写入的控制字为：8AH＝10001010B。

工作中，在给输出设备送数前，先通过 PC_7 查询设备状态，若准备好再从 A 口送出数据，然后用 PC_1 发选通信号使输出设备接收数据。从输入设备取数前，也先通过 PC_6 查询设备状态，准备好后，再从 B 口读入数据，然后从 PC_0 发出清除信号，以便输入后续字节。

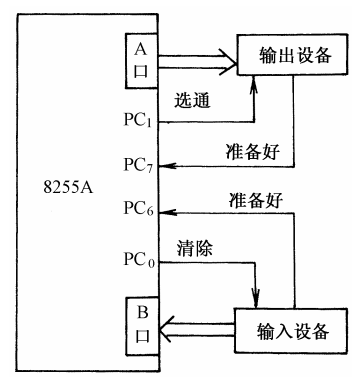


图 5-5　方式 0 下 I/O 的联络信号

2．方式 1

方式 1 为选通输入/输出方式，即可借助于选通（应答式）联络信号的 I/O 方式。在这种方式中，A 口和 B 口用于输入/输出的数据端口，C 口某些位用做接收或产生应答联络信号。方式 1 的特点是：

（1）有两组选通工作方式的端口，每组包含一个 8 位数据端口和 3 条控制线。只有 A 口和 B 口可作为数据端口，C 口的某些线被固定作为 A 口或 B 口与外设之间的联络信号线，其余的线只能定义为基本 I/O，即只能工作于方式 0。

（2）每组端口提供有中断请求逻辑和中断允许触发器。对中断允许触发器 INTE 的操作是通过对端口 C 的置位/复位控制字进行的。

（3）方式 1 在输入/输出数据时都被锁存。方式 1 可以用查询方式和中断传送方式进行数据的输入/输出。方式 1 输入时，各端口线的功能如图 5-6 所示。

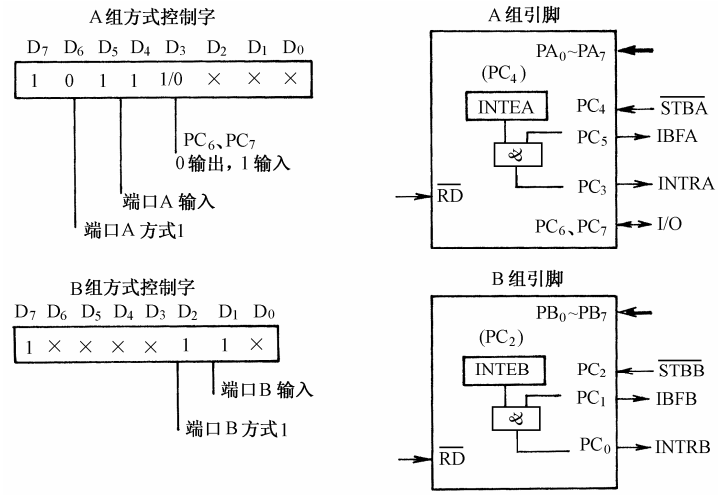


图 5-6　8255A 方式 1 输入时的控制字及信号

A 口、B 口和 PC_6、PC_7 作为数据口；端口 C 其余 6 位 PC_5～PC_0 作为控制口。A 口工作于方式 1 输入，固定用 PC_5～PC_3 做联络信号线；B 口工作于方式 1 输入，固定用 PC_2～PC_0 做联络信号线。各信号的作用说明如下：

$\overline{STB}$（STROBE）：选通信号，输入，低电平有效。它将外设的信号输入 8255A 的锁存器中。

IBF（INPUT BUFFER FULL）：输入缓冲器满信号，输出，高电平有效，这是 8255A 输出的状态信号，通知外设送来的数据已被接收。当 CPU 用输入指令读走数据后，此信号被清除。

INTR（INTERRUPT REQUEST）：中断请求信号，输出，高电平有效。当输入数据时，若 IBF 有效或输出数据时 $\overline{ACK}$ 有效，则 INTR 变成有效，以便向 CPU 发出中断请求。

INTE（INTERRUPT ENABLE）：中断允许位，INTE=0 禁止中断，可事先用位控方式写入。INTEA 写入 PC_4，INTEB 写入 PC_2。

方式 1 输出时，各端口线的功能如图 5-7 所示。

A 口、B 口、C 口的 PC_4、PC_5 作为数据口；PC_3～PC_0、PC_6、PC_7 作为控制口。A 口工作于方式 1 输出，所用的联络信号线为 PC_7、PC_6 和 PC_3，而 B 口工作于方式 1 输出时，使用 PC_2～PC_0 作为其联络信号线。各联络信号的作用如下所述：

$\overline{OBF}$（OUTPUT BUFFER FULL）：输出缓冲器满，低电平有效。当 $\overline{OBF}$ 有效时，表示 CPU 给指定端口写入一个字节数据，通知外设可以取数据。$\overline{OBF}$ 是由写信号 $\overline{WR}$ 的上跳沿置成有效电平的，而由 $\overline{ACK}$ 的有效信号使它恢复为高电平。

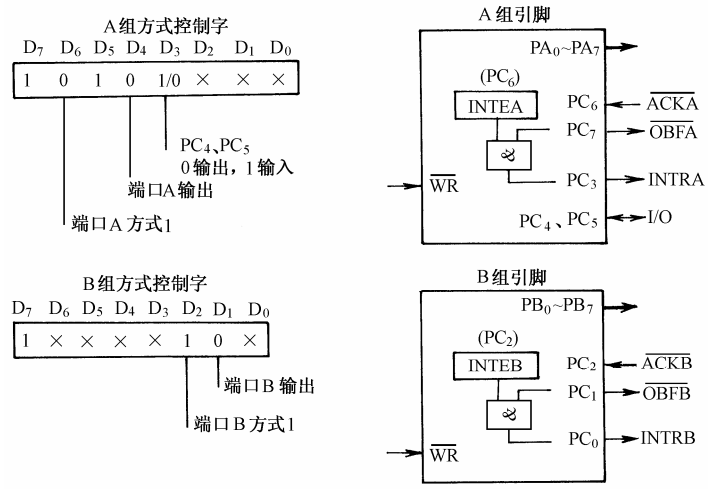


图 5-7　8255A 方式 1 输出时的控制字及信号

$\overline{\text{ACK}}$（ACKNOWLEDGE）：**应答信号，低电平有效。当外设得知** $\overline{\text{OBF}}$ **信号取数据时，要发出** $\overline{\text{ACK}}$ **信号选通，取走数据并清除** $\overline{\text{OBF}}$ **。A、B 两口的** $\overline{\text{ACK}}$ **信号分别由** PC_6 **及** PC_2 **提供。**

INTR **中断请求信号、**INTR **中断允许位，其作用及引出端都和方式** 1 **输入时相同。**

3．方式 2

方式 2 **为分时双向输入/输出方式（双向** I/O **方式），即同一端口的** I/O **线既可以作为输入也可以作为输出。方式** 2 **的主要特点为：**

（1**）**A **口可以工作于方式** 2**，此时** C **口有** 5 **条线固定为** A **口和外设之间的联络信号线。**C **口余下的** 3 **条线可以作为** B **口方式** 1 **下的联络线，也可以和** B **口一起成为方式** 0 **的** I/O **线。**

（2**）方式** 2 **在输入/输出数据时都被锁存。**

（3**）方式** 2 **可以用查询方式和中断传送方式进行数据的输入/输出。**

在方式 2 **时为双向传送设置的联络信号，实际上是方式** 1 **下输入和输出两种操作时的组合。只有中断申请信号** INTR **既可作为输入的中断申请，又可作为输出的中断申请。如图** 5-8 **是** 8255A **工作方式** 2 **的控制字及信号组合。**

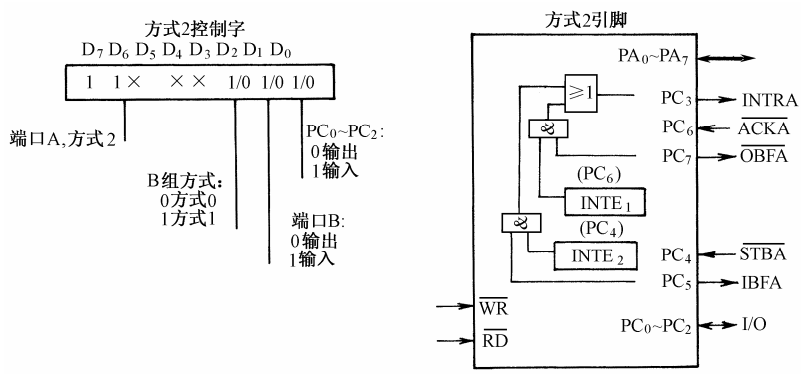


图 5-8　8255A 方式 2 的控制字及信号

5.2.4 可编程接口芯片 8255A 的应用

8255A 是一个多功能的芯片，应用非常广泛，下面以 8255A 在打印机接口和双机通信等方面的应用来介绍其基本使用方法。

1．8255A 在并行打印机接口电路中的应用

要求：为某应用系统配置一个打印机接口，CPU 通过接口采用查询方式把存放在 200H 单元开始的 256 个字符（ASCII）码送去打印。

分析：目前打印机一般采用并行接口 Centronics 标准（参见打印机接口标准），其主要接口信号与传送时序如图 5-9 所示。

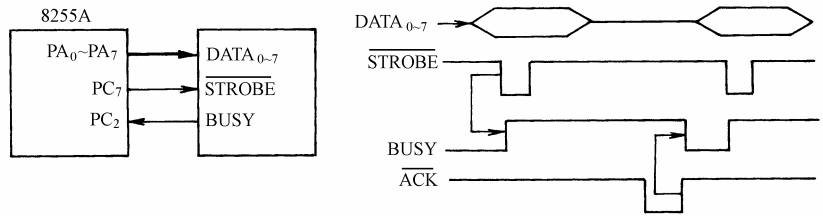


图 5-9　方式 0 的打印机接口

打印机接收主机传送数据的过程是：主机准备好输出打印的一个数据，首先查询 BUSY，若 BUSY=1（忙），则等待；若 BUSY=0（不忙），传送数据。通过并行接口把数据送给打印机的数据引脚 $DATA_0$～$DATA_7$，同时送出一个数据选通信号 $\overline{STROBE}$ 给打印机。打印机收到该信号后，把数据锁存到内部缓冲区，同时在 BUSY 信号线上发出忙信号。待打印机处理好输入数据后，打印机撤销忙信号，即置 BUSY=0，同时又向主机发出一个响应信号 $\overline{ACK}$，表示数据已经处理完毕。主机可利用 BUSY 信号或 $\overline{ACK}$ 信号决定是否输出下一个数据。

设计：采用 8255A 作为打印机接口电路，CPU 与 8255A 利用查询方式输出数据，选用 PA 口输出 8 位打印数据，工作方式为 0 方式，用 PC 口的 PC_7 引脚产生负脉冲选通信号 $\overline{STROBE}$，用 PC_2 引脚接收打印机忙信号（BUSY）并查询其状态。

假设 8255A 的 A、B 和 C 的 I/O 地址为 300H、301H 和 302H，控制端口的地址为 303H。驱动程序的程序段如下：

```
                                ；初始化程序段
    MOV   DX，303H              ；8255 命令（控制）端口
    MOV   AL，10000001B         ；工作方式字
    OUT   DX，AL                ；A 口 0 方式，输出；C4～C7 输出，C0～C3 输入
    MOV   AL，00001111B         ；PC7 位置高，使 STOBE =1
    OUT   DX，AL，
    MOV   SI，200H              ；打印字符的首字符
    MOV   CX，00H               ；打印字符的个数
L：  MOV   DX，302H              ；PC 口地址
    IN    AL，  DX              ；查 BUSY 是否为 0（PC2=0）
    AND   AL，   04H
```

```
        JNZ    L                    ；忙，则等待；不忙，则向 A 口送数
        MOV    DX，300H             ；PA 口地址
        MOV    AL，[SI]             ；从内存取数
        OUT    DX，AL               ；送数到 A 口
        MOV    DX，303H             ；8255 控制口
        MOV    AL，00001111B        ；置 STROBE 为低（PC7=0）
        OUT    DX，AL               ；负脉冲宽度（延时）
        NOP
        NOP
        MOV    AL，00001111B        ；置 STROBE 为高（PC7=1）
        OUT    DX，AL
        INC    SI                   ；内存地址加 1
        DEC    CX                   ；字符数减 1
        JNZ    L                    ；未完继续
        HLT                         ；已完，暂停
DATA1   DB     …                    ；256 个 ASCII 字符代码
```

2．双机并行通信接口设计

要求：在甲乙两台微机之间并行传送数据。甲机发送，乙机接收。甲机一侧的 8255A 采用方式 1 发送数据，乙机一侧的 8255A 采用方式 0 接收数据。两机的 CPU 与接口之间都采用查询方式交换数据。

分析：根据要求，双机均采用可编程并行接口芯片 8255A 构成接口电路，只是 8255A 的工作方式不同。

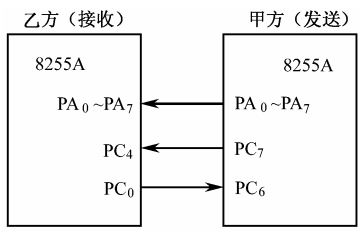


图 5-10　双机并行通信接口电路

设计：

（1）硬件连接，根据上述要求，接口电路如图 5-10 所示。甲机 8255A 是方式 1 发送，因此把 PA 口指定为输出，发出数据，用 PC_7 和 PC_6 引脚分别固定作为联络线 $\overline{OBF}$ 和 $\overline{ACK}$。乙机 8255A 是方式 0 接收，故把 PA 口定义为输入，接收数据，选用引脚 PC_4 和 PC_0 作为联络线。虽然两侧的 8255A 都设置了联络线，但有本质的区别，甲机 8255A 是方式 1，其联络线是固定的、不可替换的；乙机的 8255A 是方式 0，其联络线是不固定的、可选择的，例如可选择 PC_4 和 PC_1 或 PC_5 和 PC_2 等任意组合。

（2）软件编程，接口驱动程序包括发送与接收两个程序。

甲机发送程序：

```
        MOV    DX，303H             ；8255A 命令口
        MOV    AL，10100000B        ；初始化工作方式字
        OUT    DX，AL
        MOV    AL，ODH              ；置发送中断允许 INTEA=1
        OUT    DX，AL               ；PC6=1
```

```
        MOV    AX，   030H          ；发送数据内存首址
        MOV   ES，    AX
        MOV   BX，    00H
        MOV   CX，    3FFH         ；发送字节数
        MOV   DX，    300H         ；向 AD 送一个数，产生第一个 OBF 信号
        MOV   AL，    ES：[BX]     ；送给对方，以便获取对方的 ACK 信号
        OUT   DX ，   AL
        INC    BX                  ；内存加 1
        DEC   CX                   ；字节数减 1
L：     MOV   DX，    302H         ；8255A 状态口
        IN     AL，   DX           ；查发送中断请求，INTRA=1
        AND    AL，   08H          ；PC3=1
        JZ     L                   ；若无中断请求，则等待；有中断请求，则向 A 口写数据
        MOV    DX，   300H         ；8255A，A 口地址
        MOV    AL，   ES：[BX]     ；从内存取数
        OUT    DX，   AL           ；通过 A 口向乙机发送下一个数据
        INC    BX                  ；内存地址加 1
        DEC    CX                  ；字节数减 1
        JNZ    L                   ；字节未完，继续
        MOV   AX，   4C00H         ；已完，退出
        INT   21H                  ；返回 DOS
```

在上述发送程序中，是查状态字的中断请求 INTR 位（PC_3），实际上，也可以查发送缓冲器满 $\overline{OBF}$ （PC_7）的状态，只有当发送缓冲器空时，CPU 才能送下一个数据。

乙机接收程序：

```
        MOV     DX，    303H          ；8255A 命令口
        MOV     AL，    10011000B     ；初始化工作方式字
        OUT     DX，    AL
        MOV     AL，    00000001B     ；置 ACK =1（PC0=1）
        OUT     DX，    AL
        MOV     AX，    040H          ；接收数据内存首址
        MOV     ES，    AX
        MOV     BX，    00H
        MOV     CX，    3FF           ；接收字节数
L1：    MOV     DX，    302H          ；8255A PC 口
        IN      AL，    DX            ；查甲机的 OBF =0（PC4=0）
        AND     AL，    10H           ；查甲机是否有数据发来
        JNZ     L1                    ；若无数据发来，则等待，若有数据，则从 A 口读数
        MOV     DX，        300H      ；8255A  PA 口地址
        IN      AL，        DX        ；从 A 口读入数据
```

```
MOV     ES：        [BX]，AL        ；存入内存
MOV     DX，        303H            ；产生 ASK 信号并发回甲机
MOV     AL，        00000000B       ；PC0 置“0”
OUT     DX，        AL
NOP
NOP
MOV     AL，        00000001B       ；PC0 置“1”
INC     BX                          ；内存地址加 1
DEC     CX                          ；字节数减 1
JNZ     L1                          ；字节未完，则继续
MOV     AX,         4C00H           ；已完，退出
INT     21H                         ；返回 DOS
```

5.3 定时/计数技术概述

在微型计算机应用系统中经常要用到定时信号，以进行准确地定时、延时和计数控制。例如，许多微型计算机中动态存储器的刷新定时，要产生实时时钟信号以实现计数功能，对外部事件进行计数及统计外部事件发生的次数，系统日历时钟的计时以及喇叭的声源，都是用定时来产生的。又比如在计算机实时控制与处理系统中计算机主机需要每隔一定的时间就对处理对象进行采样，再对获得的数据进行处理，也要用到定时信号。

系统定时可分为两类：一类是微机本身运行的定时，称为内部定时，因而使计算机的每一种操作都是在精确的定时信号控制下按照严格的时间节拍执行的；另一类定时是外部设备（包括一些实时控制对象）实现某种功能时，在计算机与外设之间或者外设与外设之间的时间配合问题，称为外部定时。

计算机内部定时已在计算机设计时由其硬件结构确定了，它们具有固定的时序关系，是无法更改的，而且其他一切定时都应以此为基准。而外部定时则由于外设或被控对象完成的任务不同、内部结构不同而功能各异，所需要的定时信号也各不相同，不可能有统一的模式，因此往往需要由用户或研制者自行设定。由于定时的本质是计数，即把若干小段的计时单元累加起来，就获得一个固定的时间间隔。计数是定时的基础。

为获得所需要的定时，要求准确而稳定的时间基准，产生这种时间基准的方法通常有两种：软件定时和硬件定时。硬件定时又可分为不可编程（固定）硬件定时和可编程硬件定时。

软件定时是最简单的定时方法，实现软件定时的方法就是由 CPU 调用一个具有固定延时时间的延时子程序。由于延时程序中每条指令的执行时间是确定的，它所包含的时钟周期数也是固定的、已知的，将子程序中所有指令的时钟周期数量相加后再乘以时钟周期时间，就得到该子程序执行后所产生的延时时间。当子程序执行完毕后就可用输出指令输出一个信号作为定时控制输出。当延时时间常数较大时，可将延时程序设计成一个循环程序，通过控制循环次数和循环体内的指令来确定不同的延时时间，从而达到定时的目的。

软件定时方法简单、灵活，不需要增加硬件电路，但这种方法的缺点是降低了 CPU 的工作效率，在定时循环时间内，CPU 一直被占用，定时时间越长，占用 CPU 的时间就越长，效率就越低，浪费了 CPU 的资源。因此，软件定时方法常用于定时时间短、重复调用次数少

的场合，如键盘操作的消抖动延时。另外，在设计延时程序时，指令的选取，执行时间的计算都是比较麻烦的事，选择不好，会影响延时精度，这也是软件定时的不足之处。尽管如此，软件定时在实际中还是经常使用的。

硬件定时就是用专门的定时电路产生定时或延时信号。硬件定时又可分为不可编程的硬件定时和可编程的硬件定时。在实际中广泛使用的是利用可编程定时/计数器来实现定时或延时。这种可编程的硬件定时方式利用专门的定时/计数器（其核心是一个减 1 计数器）并通过软件来确定不同方式的定时计数功能，以满足 CPU 和外设（或控制对象）所需要的定时或延时要求。

硬件定时方法不占用 CPU 的时间，提高了 CPU 的工作效率，它的定时时间和范围完全由软件来确定和改变，使用灵活方便，而且定时精度高。因此这种方法在实际中获得了广泛的应用，特别是在微型计算机系统中是一个必备的接口部件。Intel 8253 就是一种能完成上述功能的计数/定时器芯片。

5.3.1 8253 的内部结构

Intel 8253 可编程计数/定时器芯片，其基本特性如下：

（1）8253 具有 3 个独立的 16 位计数器；

（2）可按二进制或十进制（BCD）计数；

（3）可由程序设置 6 种不同的工作方式；

（4）计数时钟频率（CLK）为 2MHz。

8253 的内部结构如图 5-11 所示，由数据总线缓冲器、读写控制逻辑、控制字寄存器和 3 个结构完全相同的计数器组成。

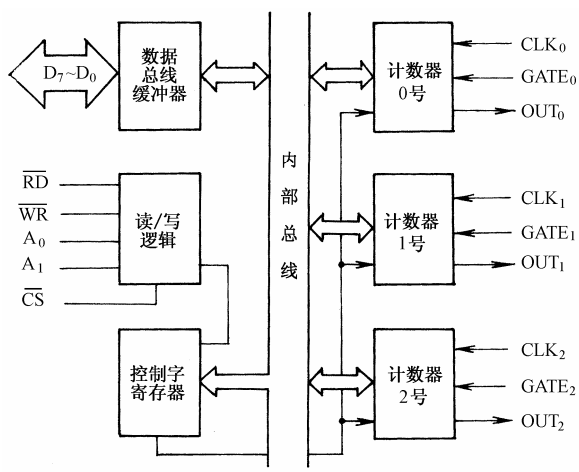


图 5-11　8253 内部结构图

1．数据总线缓冲器

数据总线缓冲器是 8253 连接 CPU 数据总线的 8 位双向三态缓冲器。CPU 读/写 8253 的所有数据都经过这个缓冲器。

CPU 用输出指令向 8253 写入方式控制字至控制寄存器，或写入计数值至某个计数器，都是经数据总线缓冲器和 8253 内部总线传送的。CPU 用输入指令读某个计数器时，指定计

数器的现行计数值经内部总线和缓冲器传送到系统数据总线上，读入 CPU。

2．读/写控制逻辑

读/写控制逻辑是 8253 内部操作的控制部件，它接收来自系统总线的输入信号，将其转换成 8253 内部操作和种种控制信号，选择读/写操作的对象（计数器或者控制寄存器），决定内部总线上数据传送的方向。

读/写逻辑接收 $\overline{CS}$ 芯片选择信号控制。$\overline{CS}$ 为 1 时读/写被禁止，CPU 不能对 8253 进行读/写操作，数据总线缓冲器呈高阻状态，与系统的数据总线脱离，这时芯片设置的工作方式和各计数器的现行工作不受影响。当 $\overline{CS}$ 为 0 时 CPU 可读/写计数器或写控制字到控制寄存器中。

3．控制字寄存器

当 $\overline{CS}$=0 且 A_1A_0=11 时，该寄存器接收来自 CPU 写入 8253 的方式控制字。控制字控制指定计数器的工作方式，选择二进制或二-十进制计数，规定读/写的具体方法。控制字寄存器只能写入，其内容不能读出。8253 控制字格式如图 5-12 所示。

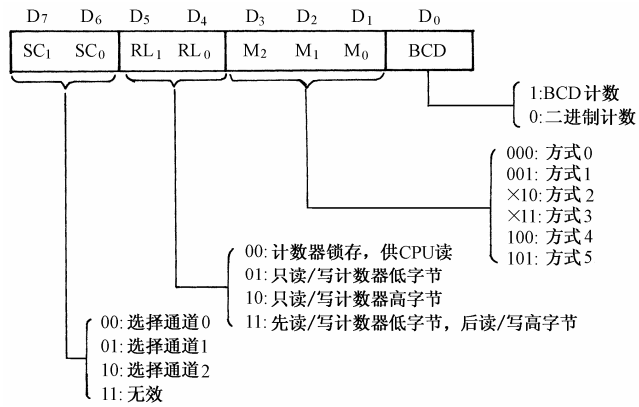


图 5-12　8253 的控制字

SC_1、SC_0：计数通道选择位。由于 8253 内部 3 个计数通道各有一个 8 位的控制字寄存器，而这 3 个控制字寄存器共用同一个控制端口地址，所以控制字中设置了 SC_1、SC_0 这两位来确定 CPU 当前发出的控制字是写入哪个计数通道的控制字寄存器中。

RL_1、RL_0：读/写操作方式位。这两位用来确定对选中的计数通道进行读/写的操作方式。由于 8253 的数据线只有 8 位，故传送 16 位数据时，要分两次进行。当 CPU 对 8253 进行 16 位读/写操作时，可以只读/写高 8 位或只读/写低 8 位，也可以读/写 16 位。读/写 16 位时，先读/写低 8 位，后读/写高 8 位，具体是哪种操作方式由 RL_1、RL_0 这两位的编码确定。

M_2、M_1、M_0：工作方式选择位。8253 的每个计数通道有 6 种不同的工作方式，即方式 0～方式 5，$M_2M_1M_0$ 这 3 类就是用来选择具体的工作方式的，具体选择如图 5-12 所示。

BCD：计数方式选择位。8253 的每个计数通道都有两种计数方式按二进制计数或按十进制（BCD 码）计数。BCD 位用来具体确定采用哪种计数方式。

4．计数器 0、计数器 1、计数器 2

每个计数器是一个 16 位减 1 计数器，可接收计数值寄存器预置的初值，3 个计数器的内部结构是相同的，它们的操作各自完全独立。

每个计数器都可对其 CLK 输入端输入的脉冲按照二进制或二-十进制从预置初值开始减 1 计数。当计数器减到 0 时，从 OUT 输出端输出一个脉冲信号。当 CLK 端输入周期恒定的脉冲时，计数器就完成了定时功能。

在计数的开始和整个计数过程中，计数器可以受 GATE 输入的门控信号控制。与外界相连的计数器输入/输出以及门控信号之间的关系取决于该计数器的工作方式。

计数器的初值必须在计数之前由 CPU 用输出指令预置，在计数过程中，CPU 能随时用输入指令读入计数器的当前计数值而不影响计数器的计数。

5.3.2　8253 芯片的引脚及其功能

8253 是双列直插式 24 脚封装的芯片，引脚如图 5-13 所示。

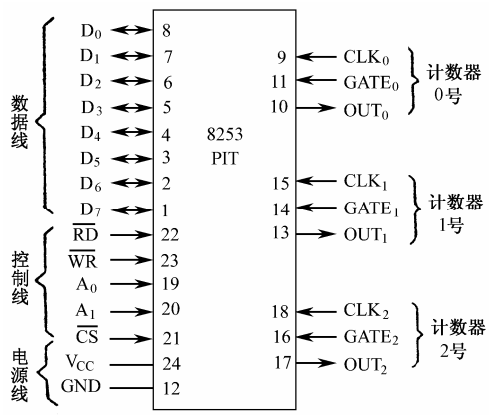


图 5-13　8253 引脚图

8253 的引脚可分为两部分：一部分是与 3 个计数器有关的引脚，另一部分是与 CPU 连接的引脚。

8253 的 3 个计数器功能是完全一样的，其内部逻辑和外部引脚都是相同的。每个计数器有 3 个引脚：脉冲输入、计数输出和门控输入。

CLK：脉冲输入引脚。计数器对该引脚输入的脉冲进行计数。8253 的基本工作方式是对 CLK 端输入的脉冲进行计数。每输入一个时钟信号，内部计数器便进行一次减 1 计数。CLK 输入脉冲可以是任何脉冲源提供的脉冲（只要它的周期不小于 380ns 即可），包括系统时钟和系统时钟分频后的脉冲。

GATE：门控脉冲输入引脚。这是外部控制计数器工作的门控脉冲输入引脚。通常，GATE 为 0 时禁止计数器工作，GATE 为 1 时允许计数器工作。GATE 也可启动或中止计数器／定时器的运行。

OUT：计数到 0／定时时间到输出引脚。不管工作于何种方式，当计数到 0 时，OUT 引脚定有输出，但在不同的工作方式下输出不同形式的信号。

用于与 CPU 及系统相连接的引脚有：

$\overline{CS}$：片选，输入，用于连接地址译码端。

$\overline{RD}$：读信号，输入，用于连接系统的读操作信号。

$\overline{WR}$：写信号，输入，用于连接系统的写操作信号。

A_1A_0：地址信号，输入，用于选择芯片内部寄存器。

一片 8253 占 4 个端口，对应片内的 3 个计数器和一个控制字寄存器。端口的选择由 8253 的 A_1 和 A_0 输入引脚的状态决定，它们通常接系统地址总线的 A_1 和 A_0。片选 $\overline{CS}$、读 $\overline{RD}$、写 $\overline{WR}$ 和 A_1A_0 对 8253 各端口的选择和读/写操作如表 5-1 所示。

表 5-1　8253 端口选择和读/写操作

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A_1	A_0	端口选择和操作
0	1	0	0	0	写入计数器 0
0	1	0	0	1	写入计数器 1
0	1	0	1	0	写入计数器 2
0	1	0	1	1	写方式控制字到控制字寄存器
0	0	1	0	0	读计数器 0
0	0	1	0	1	读计数器 1
0	0	1	1	0	读计数器 2
0	0	1	1	1	无操作，数据总线缓冲器三态
0	1	1	×	×	无操作，数据总线缓冲器三态
1	×	×	×	×	禁止，数据总线缓冲器三态

5.3.3　8253 的工作方式

8253 每个计数器有 6 种不同的工作方式，由控制字的 $D_3D_2D_1$ 确定。在不同的工作方式下，计数器完成不同的功能。不同的工作方式中，启动计数器的触发方式各不相同，有电平触发和边缘触发，计数过程中门控信号 GATE 对计数器操作的影响也不同；另外不同的工作方式其输出波形亦有不同。

1．方式 0——计数结束产生中断

8253 用做计数器时一般工作在方式 0。所谓计数结束产生中断，是指在计数值减到 0 时，输出端（OUT）产生的输出信号可作为中断申请信号，要求 CPU 进行相应的处理。方式 0 有如下特点：

（1）当控制字写进控制字寄存器确定了方式 0 时，计数器的输出（OUT 端口）保持低电平，一直保持到计数值减到 0 为止。

（2）计数初值装入计数器之后，在门控 GATE 信号为高电平时计数器开始减 1 计数。当计数器减到 0 时输出端 OUT 才由低变高，此高电平输出一直保持到该计数器装入新的计数值或再次写入方式 0 控制字为止。若要使用中断，可以用计数到 0 的输出信号向 CPU 发出中断请求，申请中断。

（3）GATE 为计数控制门，方式 0 的计数过程可由 GATE 控制暂停。即 GATE=1 时，允许计数；GATE=0 时，停止计数。GATE 信号的变化不影响输出 OUT 端口的状态。

（4）在计数过程中，可重新装入计数初值。如果在计数过程中，重新写入某一计数初值，则在写完新计数值后，计数器将从该值重新开始做减 1 计数。

方式 0 的时序波形如图 5-14 所示。

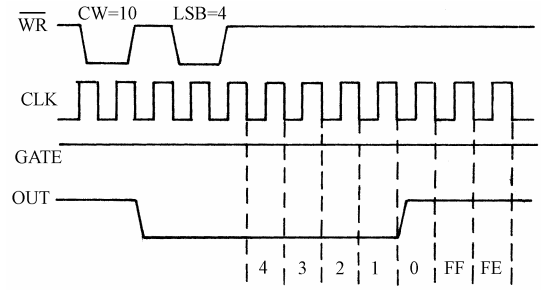


图 5-14　8253 方式 0 工作时序图

2．方式 1——可编程的单拍负脉冲

可编程的单拍负脉冲又称单稳态输出方式，简称单稳定时。方式 1 的特点是：

（1）CPU 写入控制字后，计数器输出 OUT 端为高电平作为起始电平，在写入计数值后计数器并不开始计数（不管此时 GATE 是高电平还是低电平），而要由外部门控 GATE 脉冲上升沿启动，并在上升沿之后的下一个 CLK 输入脉冲的下降沿开始计数。

（2）GATE 上升沿启动计数的同时，使输出 OUT 变低，每来一个计数脉冲，计数器做减 1 计数，直到计数减为 0 时，OUT 输出端再变为高电平。OUT 端输出的单拍负脉冲的宽度为计数初值乘以 CLK 端脉冲周期。设计数初值为 N，则单拍脉冲宽度为 N 个 CLK 时钟脉冲周期。

（3）如果在计数器未减到 0 时，GATE 又来了一个触发脉冲，则由下一个时钟脉冲开始，计数器将从初始值重新做减 1 计数。当减至 0 时，输出端又变为高电平。这样，使输出脉冲宽度延长。

方式 1 的时序波形如图 5-15 所示。

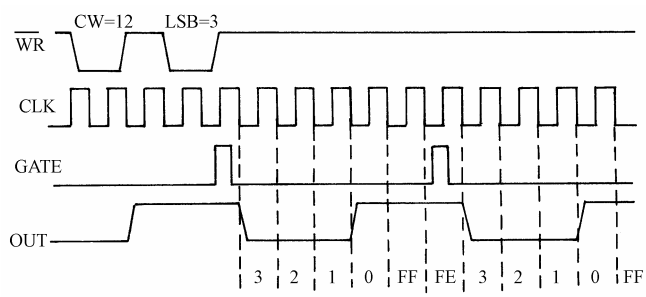


图 5-15　8253 方式 1 工作时序图

3．方式 2——分频脉冲发生器

方式 2 是一种具有自动预置计数初值 N 的脉冲发生器。从 OUT 端可以输出连续脉冲信号，脉冲宽度等于时钟脉冲周期，而计数初值 N 决定了输出端两个负脉冲之间的宽度即输出脉冲周期。方式 2 也叫 N 分频器，因为输出脉冲为输入脉冲的 N 分频，即出现 N 个输入脉冲才输出一个脉冲。方式 2 有如下特点：

（1）N 分频计数器方式是输出对输入脉冲按计数器计数初值 N 分频后的连续脉冲。

（2）当 CPU 写入控制字后，OUT 端输出为高电平作为起始电平，在写入计数值 N 后将立即自动开始对输入脉冲 CLK 计数，输出端仍一直为高；当计数器减到 1 时，输出变低，计数器减到 0 时又变为高，计数器重新按已写入的计数值 N 继续计数，周而复始，在 OUT 端输出一个 N 分频脉冲，其正脉冲宽度为（N-1）个输入脉冲时钟周期（是 N 个 CLK 时钟脉冲周期之和），而负脉冲输出宽度（持续时间）是一个 CLK 脉冲周期。

（3）GATE 用于控制计数，GATE=1，允许计数；GATE=0，停止计数。因此，可以用 GATE 来使计数器同步。

方式 2 的时序波形如图 5-16 所示。

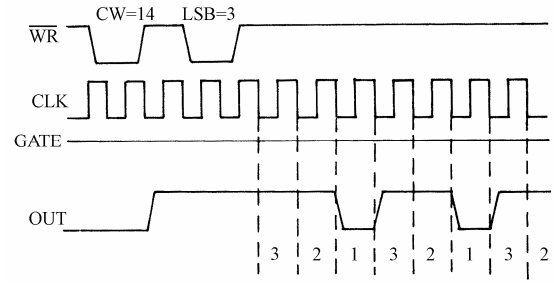


图 5-16　8253 方式 2 工作时序图

4．方式 3——分频方波发生器

方式 3 和方式 2 类似，其特点是：

（1）方式 3 常用于波特率发生器，其输出为方波或近似方波的矩形波。

（2）写入方式 3 控制字后输出为高电平。写入计数值后计数器自动开始对输入 CLK 脉冲计数，输出 OUT 仍保持为高；在计数完成一半时，输出 OUT 变为低电平，直到计数器全部完成时，输出 OUT 又变为高电平，并重复上述计数过程。

（3）当计数值 N 为偶数时，OUT 方波的高电平与低电平之比为 1∶1；若 N 为奇数，其高低电平之比为(N+1)/2∶(N-1)/2，即输出分频波高电平宽度为(N+1)/2 个 CLK 周期，低电平周期为(N-1)/2 个 CLK 周期。

方式 3 的时序波形如图 5-17 所示。

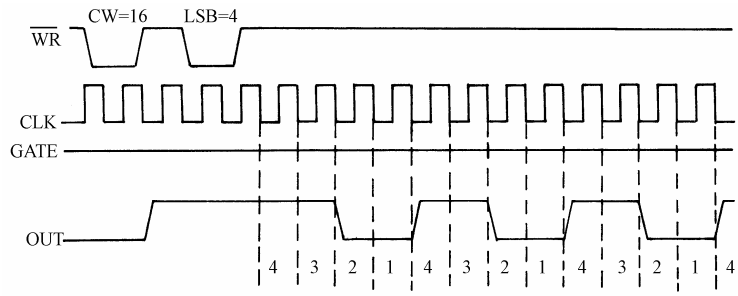


图 5-17　8253 方式 3 工作时序图

5．方式 4——软件触发选通脉冲发生器

方式 4 是类似于方式 0 的工作方式，计数器是靠置入新的计数初值这个软件操作来触发计数器工作的，故称为软件触发。方式 4 有如下特点：

（1）方式 4 是靠写入计数值来进行软件触发的“一次性有效”的选通脉冲发生器。写入控制字后输出端 OUT 变为高，并一直保持。在写入计数初值之后开始计数，当计数到 0 时输出端 OUT 变为低，维持一个 CLK 周期后又恢复为高，并一直保持为高，直到再次写入计数来进行“软件触发”才能再次开始。

（2）若 GATE=1，允许计数；GATE=0，停止计数。

（3）方式 4 的负脉冲输出常作为选通脉冲。

方式 4 的时序波形如图 5-18 所示。

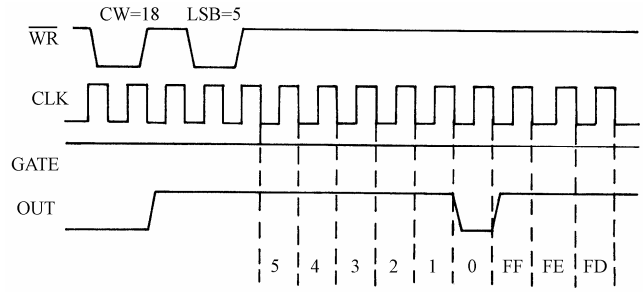


图 5-18　8253 方式 4 工作时序图

6．方式 5——硬件触发选通脉冲发生器

方式 5 有如下特点：

（1）方式 5 类似于方式 4，所不同的是 GATE 端输入信号的作用不同。方式 5 是硬件触发，是在外部硬件发出门控信号后才发生的。

（2）方式 5 是靠门控脉冲 GATE 的上升沿来进行触发的选通脉冲发生器。写入控制字后输出端 OUT 为高，这是初始电平；写入计数值后计数器并不开始计数，而要由门控脉冲 GATE 上升沿触发后才开始计数，计数到 0 时输出由高变低，一个 CLK 时钟周期后又恢复为高，并一直保持，直到下次门控脉冲触发再次开始计数。

（3）方式 5 的计数器可重新触发，在任何时候，当 GATE 信号的上升沿到来时，将把计数初值重新送入计数器，然后开始计数。

方式 5 的时序波形如图 5-19 所示。

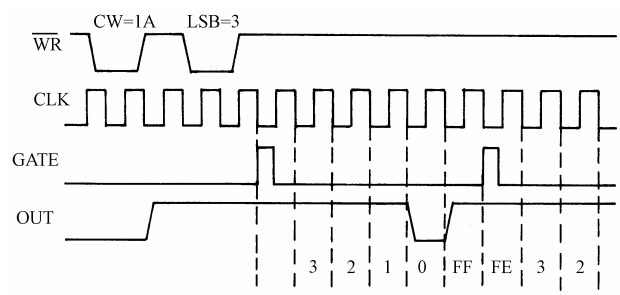


图 5-19　8253 方式 5 工作时序图

5.3.4 8253 的初始化编程

对于 8253 来说，刚加电时，处于一种未定义状态，为使其按照一定要求工作，需对芯片进行初始化，8253 的初始化编程有两项内容：一是先向控制寄存器写入控制字，以选定计数通道（3 个中的 1 个），规定该计数器的工作方式和计数方式以及计数初值的长度和装入顺序（初值写入方式）；二是向已选定的计数器按控制字的要求写入计数初值。

对 8253 初始化的要求是：

（1）对每个计数器，控制字必须写在计数值之前。这是因为计数器的读/写格式由它的控制字决定。计数值必须按控制字所规定的格式写入。若控制字规定只写 8 位，只需写入一次（8 位）计数值即可（规定写低 8 位则高 8 位自动置 0，规定写高 8 位则低 8 位自动置 0）；规定写 16 位时必须写两次，先写低 8 位，后写高 8 位。当初值为 0 时，也要分两次写入，因为在二进制计数时，“0”表示 65336，在 BCD 码计数时“0”表示 10 000=10^4。

（2）对所有方式计数器都可以在计数过程中或计数结束后改变计数值，重写计数值也必须遵守控制字所规定的格式，并且不会改变当前计数器的工作方式。

（3）计数值不能直接写到减 1 计数器中，而只能写入计数值寄存器中，并由写操作 $\overline{WR}$ 之后的下一个 CLK 脉冲将计数值寄存器的内容装入减 1 计数器开始计数。

（4）初始化编程必须明确各个计数器的控制字和计数值不是写到同一个地址单元。各个计数器的控制字各自独立确定，但它们都写入同一个端口地址（控制字寄存器）中，各个计数器的计数值则根据需要独立确定并写入各自计数器的相应寄存器中。

由于 3 个计数通道分别有各自的端口地址，因此，对这 3 个计数通道的初始化编程没有先后次序，根据使用要求对所选择的计数器编程即可。

例 5.1　若选择计数器 1，工作在方式 3，计数初值为 588H（2 个字节）采用二进制计数，则其控制字为 01110110=76H，设控制口地址为 43H，则将该控制字写入控制字寄存器的指令和写计数初值的指令如下：

```
MOV     AL，    76H
OUT     43H，   AL
MOV     AL，    88H
OUT     41H，   AL
MOV     AL，    05H
OUT     41，    AL
```

例 5.2　设一微机系统中 8253 的 3 个计数器的端口地址为 60H、62H 和 64H，控制口地址为 66H，要求计数器 0 为方式 1，按 BCD 计数；计数初值为 1800D，计数器 1 为方式 0，按二进制计数；计数初值为 1234H，计数器 2 为方式 3，按二进制计数；当计数初值为 65H 时，试分别写出计数器 0、1、2 的初始化程序。

计数器 0 的控制字为 00100011B=23H，计数器 0 的初始化程序为：

```
MOV     AL，    23H         ；计数器 0 的控制字
OUT     66H，   AL          ；控制字写入 8253 的控制器
MOV     AL，    18H         ；取计数初值的高 8 位，低 8 位 00 可不送
OUT     60H，   AL          ；计数初值送计数器 0 端口
```

计数器 1 的控制字为 01110000B=70H，计数器 1 的初始化程序为：

```
MOV     AL，    70H      ；计数器的控制字：方式 0，送高 8 位和低 8 位，二进制计数
OUT     66H，   AL       ；控制字写入 8253 的控制器
MOV     AL，    034H     ；取计数初值的低 8 位
OUT     62H，   AL       ；计数初值的低 8 位，写入计数器 1 端口
MOV     AL，    12H      ；取计数初值的高 8 位
OUT     62H，   AL       ；计数初值的高 8 位写入计数器 1 端口
```

计数器 2 的控制字为 10010110B=96H，计数器 2 的初始化程序为：

```
MOV     AL，    96H      ；计数器 2 的控制字为 96H，方式 3，只送低 8 位，二进制计数
OUT     66H，   AL       ；控制字写入 8253 的控制口
MOV     AL，    56H      ；计数初值的低 8 位
OUT     64H，   AL       ；计数初值的低 8 位写入计数器 2 的端口
```

例 5.3　要求读出计数器 2 的当前计数值，并检查是否为全“1”。

8253 在读取计数器的当前计数值时，必须分两步进行。首先发一锁存命令（即控制字中 RL_1RL_0=00），将当前计数值锁存到输出锁存器中。第二步执行读操作，即用 IN 指令将锁存器中内容读入 CPU。

假设计数初值只有低 8 位，设其程序段如下（控制口地址为 66H，计数器 2 的口地址为 64H）：

```
KEEP：MOV   AL，80H      ；计数器 2 的锁存命令
      OUT   66H，AL      ；锁存命令写入控制寄存器
      IN    AL，64H      ；读输出锁存器中的当前计数值（从计数器 2 端口读）
      CMP   AL，0FFH     ；比较当前计数值是否为全“1”
      JNE   KEEP         ；非全“1”继续读
      HLT                ；为全“1”暂停
```

5.3.5　8253 的应用举例

可编程定时器/计数器 8253 可与各种微型计算机应用系统相连构成一个完整的定时、计数器或脉冲发生器，应用于各种场合。在使用 8253 时有两项工作要做，一是要先根据实际应用要求，设计一个包含 8253 的硬件逻辑电路或接口；二是对 8253 进行初始化编程，只有初始化后 8253 才可以按要求正常工作。

8253 的 3 个计数通道是完全独立的，因此可以分别使用。使用时可以对它们分别进行硬件设计和软件编程，使它们工作在相同或不同的工作方式中。下面举例说明 8253 作为定时器和计数器的应用。

1．8253 定时功能的应用

例 5.4　将 8253 的计数器 1 作为 5ms 定时器，设输入时钟频率为 200kHz，试编写 8253 的初始化程序。

（1）计数初值 N 的计算：已知输入时钟 CLK 频率为 200kHz，则时钟周期为

$T=1/f=1/200\text{kHz}=5\mu\text{s}$，于是计数初值 N 为

$$N=5\text{ms}/T=5\text{ms}/5\mu\text{s}=1\,000$$

（2）确定控制字：按题意选计数器 1，按 BCD 码计数，工作于方式 0，由于计数初值 $N=1\,000$，控制字 D_5D_4 应为 11，于是 8253 的控制字为 01110001B=71H。

（3）选择 8253 各端口地址：设计数器 1 的端口地址为 3F82H，控制口地址为 3F86H。

（4）初始化程序如下：

```
NOV   AL，71H          ；控制字
MOV   DX，3F86H        ；控制口地址
OUT   DX，AL           ；控制字送 8253 控制寄存器
MOV   DX，3F82H        ；计数器 1 端口地址
MOV   AL，00               ；将计数初值 N=1 000 的低 8 位写入计数器 1
OUT       DX，AL
MOV   AL，10           ；将 N 的高 8 位写入计数器 1
OUT       DX，AL
```

例 5.5 以 80286 为 CPU 的某微机系统中使用了一块 8253 芯片，其通道端口地址为 308H、30AH 和 30CH，控制口地址为 30EH，3 个通道使用同一输入时钟，频率为 2MHz，要求完成如下功能：利用计数器 0 采用硬件触发，输出宽度等于时钟周期的单脉冲，定时常数为 36H；利用计数器 1 输出频率为 2kHz 的对称方波；利用计数器 2 产生宽度为 0.6ms 的单脉冲。

试设计该定时系统硬件电路和初始化程序。

（1）硬件电路设计。硬件电路设计主要是地址译码电路设计及 8253 与 CPU 间的连接。根据给定的端口地址可知，地址总线低位部分的 A_9～A_0 分别为：A_9A_8=11、A_7～A_4=0000、$A_3A_2A_1$=100～111、A_0=0，由它们经译码器译码产生 8253 的片选信号 $\overline{CS}$，8253 的数据线 D_7～D_0 必须与系统数据总线的低 8 位相连，8253 的端口选择信号 A_1A_0 应连系统地址的 A_2A_1。根据上述要求，译码器应选 3/8 译码器 74LS138。该译码器有 3 个代码输入端（C、B、A），输入 3 位代码决定译码信号从 Y_0～Y_7 中哪一个输出，本例中显然应以 Y_2 为输出端。

3/8 译码器有 3 个控制端 G_1、$\overline{G_{2A}}$、$\overline{G_{2B}}$，它们必须为 100 时译码器才能正常工作，根据以上分析可画出硬件逻辑电路，如图 5-20 所示。

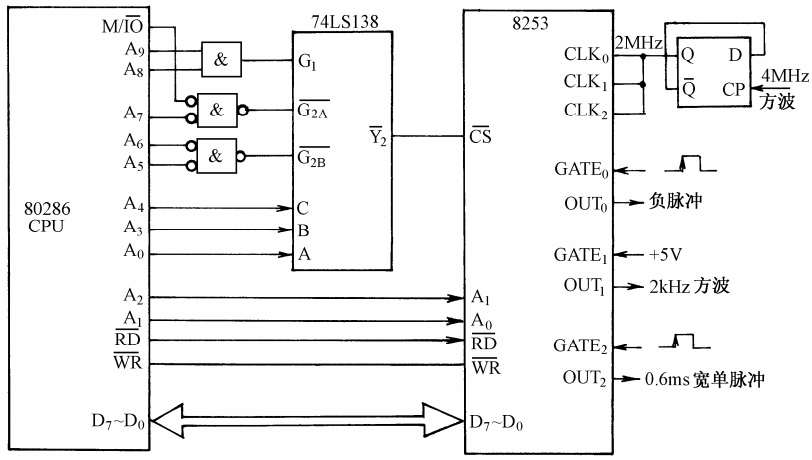


图 5-20　8253 定时信号产生电路

（2）初始化编程。根据题意要求，对 3 个通道的工作方式，计数初值确定如下：

由 CLK_0～CLK_2=2MHz 可得，时钟周期

$$T=1/f=1/2\text{MHz}=0.5\mu\text{s}$$

选计数器 0：选择方式 5，门控信号 GATE 应接一正跳变信号，OUT 端当计数为 0 时产生一个宽度等于时钟周期的单脉冲。计数系数为 36，用 BCD 计数。所以，计数器 0 的控制字应为 00011011B=1BH。

选计数器 1：选择方式 3，GATE 接+5V，CLK_1=2MHz，输出方波频率为 2kHz，所以计数器 N_1=2MHz/2kHz=1 000，采用 BCD 计数，于是计数器 1 的控制字为 01110111B=77H。

选计数器 2：选择方式 1，以构成一个单稳态电路，输出脉冲宽度由计数常数 N_2 决定，计数常数 N_2=600μs/0.5μs=1 200，采用 BCD 计数，于计数器 2 的控制字为 10110011B=B3H。

根据以上分析可得 3 个计数通道的初始化程序如下。

计数通道 0 的初始化程序：

```
MOV   DX，30EH        ；8253 的控制口地址
MOV   AL，1BH         ；计数通道 0 的控制字，低 8 位，方式 5，BCD 计数
OUT       DX，AL          ；控制字写入控制口
MOV   DX，308H        ；计数器 0 的端口地址
MOV   AL，36H         ；计数初值的低 8 位
OUT       DX，AL          ；低字节写入计数器 0 端口
```

计数通道 1 的初始化程序：

```
MOV   DX，30EH        ；8253 的控制口地址
MOV   AL，77H         ；计数通道 1 的控制字，先写低字节，后写高字节，方式 3，BCD 计数
OUT   DX，AL          ；控制字写入控制口
MOV   DX，30AH        ；计数通道 1 的端口地址
MOV   AL，00H         ；计数初值的低字节
OUT    DX，AL         ；低字节写入计数通道 1
MOV   AL，10              ；计数初值的高字节
OUT    DX，AL         ；高字节写入计数通道 1
```

计数通道 2 的初始化程序：

```
MOV   DX，30EH        ；8253 的控制口地址
MOV   AL，B3H         ；计数通道 2 的控制字，先写低字节，后写高字节，方式 1，BCD 计数
OUT    DX，AL         ；控制字写入控制口
MOV   DX，30CH        ；计数通道的端口地址
MOV   AL，00H         ；计数初值的低字节
OUT   DX，AL          ；低字节写入计数通道
MOV   AL，12          ；计数初值的高字节
OUT   DX，AL          ；高字节写入计数通道 2
```

2．8253 计数功能的应用

例 5.6　某 80286 系统中有一片 8253 芯片，利用计数通道 2 对外部事件计数，计满 360

次经 8259A 的 IR_1 向 CPU 发出中断信号，硬件电路如图 5-21 所示。计数器 2 的口地址为 5DH，控制口地址为 5FH。试编写 8253 的初始化程序。

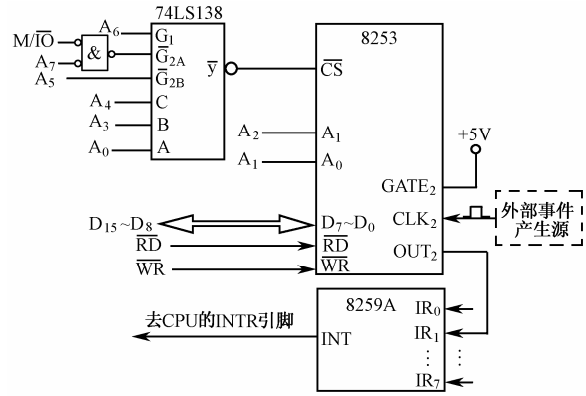


图 5-21　8253 计数功能原理图

（1）题意分析：对外部事件计数时，外部事件作为计数脉冲从通道 2 的 CLK_2 进入，采用方式 0 工作，计数常数即为外部发生的总事件为 360D，当计满 360 次时从 OUT_2 端输出一个正跳变信号作为中断申请信号送入 8259A 的 IR_1 输入端，由 8259A 向 CPU 送出中断申请。

（2）初始化编程：根据分析，计数通道 2 工作于方式 0，计数初值为 360D=168H，采用二进制计数，则其控制字为 10110000B=B0H。

计数通道 2 的初始化程序为：

```
MOV   AL，B0H      ；计数通道 2 的控制字
OUT       05FH，AL     ；控制字写入控制器
MOV   AL，68H      ；计数初值的低 8 位
OUT       5DH，AL      ；计数初值的低 8 位写入计数通道 2
MOV   AL，01H      ；计数初值的高 8 位
OUT       5DH，AL      ；高 8 位写入计数通道 2
```

5.4　可编程串行通信接口芯片8251A

随着大规模集成电路技术的发展，通用的可编程串行通信接口芯片种类越来越多，例如 INS 8250、MC 6850、MC 6852、MC 6854、Intel 8251、Intel 8273 等等。常用的通用串行接口芯片有两类，一种是仅用于异步通信的接口芯片，称为通用异步收发器 UART（Universal Asynchronous Receive-Transmitter），如 National INS 8250 就是这种器件，IBM PC 机中用 INS8250 做串行接口芯片。另一种芯片既可以工作于异步方式又可以工作于同步方式，称为同步异步收发器 USART（Universal Synchronous-Asynchronous Receiver-Transmitter ），如 Intel 8251A 就是这种器件。

5.4.1　8251A 的基本结构与功能

8251A 是一种可编程的通用同步/异步接收发送器，通常作为串行通信接口使用，其基本功能为：

（1）全双工、双缓冲器的接收/发送器。

（2）可工作在同步或异步工作方式。同步方式工作时，波特率在 0～64K 范围内；异步方式时，波特率在 0～19.2K 范围内。

（3）同步方式时，字符可选择为 5～8 位，可加奇偶校验位，可自动检测同步字符。异步方式时，字符可选择为 5～8 位，可加奇偶校验位，自动为每个字符添加一个启动位，并允许通过编程选择 1、1.5 或 2 位停止位。

8251A 的内部结构如图 5-22 所示。整个 8251A 有 5 个主要组成部分：接收器、发送器、调制解调控制逻辑、读/写控制逻辑和数据总线缓冲器。

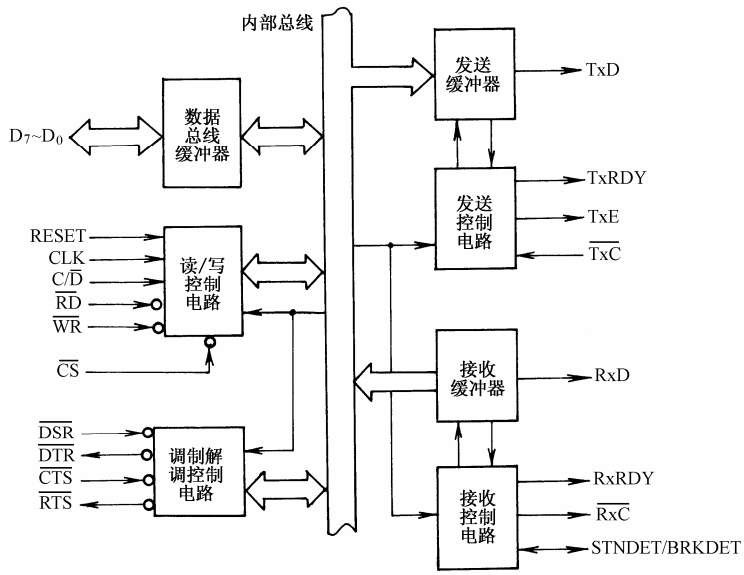


图 5-22　8251A 内部结构图

1．接收器

接收器包括接收缓冲器和接收控制电路两部分。接收器的功能是接收在 RxD 引脚上的串行数据，并按规定的格式把它转换成并行数据，存放在数据总线缓冲器中。

（1）接收缓冲器。接收缓冲器主要由移位寄存器和数据寄存器组成。相对应的外部引脚有：

RxD：接收数据输入端。

接收器接收传送到 RxD 引脚上的串行数据，并对串行数据流的特殊位（奇偶位、停止位等）和字符（同步字符）进行检查、处理，按规定的格式将串行数据转换为并行数据存放在缓冲器中。

接收移位寄存器和接收数据缓冲器组成了双缓冲器结构。

（2）接收控制逻辑。这一部分控制串行数据的接收，包括 3 条控制线：

RxRDY：接收器准备好，输出，高电平有效，

$\overline{\text{RxC}}$ ：接收时钟，输入，低电平有效；

SYNDET/BRKDET：同步检测/断点检测，输出/输入，高电平有效。

2．发送器

发送器包括发送缓冲器和发送控制电路两部分。

（1）发送缓冲器。发送数据缓冲器接收由 CPU 送来的并行数据，按初始化编程指定的数据格式转换成串行数据流送至发送移位寄存器，在 $\overline{\text{TxC}}$ 的下降沿从 TxD 引脚发送出去。发送缓冲器的相关引脚有：

TxD：发送数据输出端。

发送数据缓冲器和发送移位寄存器组成了发送的双缓冲器结构。

（2）发送控制逻辑。该部分控制串行数据的发送操作，包括 3 条控制线：

TxRDY：发送器准备好，输出，高电平有效；

TxE：发送器空，输出，高电平有效；

$\overline{\text{TxC}}$ ：发送时钟，输入，低电平有效。

3．读/写控制逻辑

读/写控制逻辑接收 CPU 的有关控制信号，据此确定对 8251A 的操作，该部分共有 6 条对外引线。

CLK：时钟，输入；

RESET：复位，输入，高电平有效。若 RESET 有效，8251A 被强行复位到空闲状态，只有在重新初始化后才能脱离空闲状态；

$\overline{\text{CS}}$ ：片选，输入，低电平有效。

C/$\overline{\text{D}}$ ：控制/数据端口选择信号，用来区分当前读写的是数据还是控制信息或状态信息。具体地说，CPU 在读操作时，若 C/$\overline{\text{D}}$ 为低电平，则读取的是数据，若 C/$\overline{\text{D}}$ 是高电平，则读取的是 8251A 当前的状态信息；CPU 在写操作时，若 C/$\overline{\text{D}}$ 为低电平，则写入的是数据，若 C/$\overline{\text{D}}$ 为高电平，则写入的是 CPU 对 8251A 的控制命令。

$\overline{\text{RD}}$ ：读控制，输入，低电平有效。当读信号为低电平时，用来通知 8251A，CPU 当前正在从 8251A 读取数据或者状态信息。

$\overline{\text{WR}}$ ：写控制，输入，低电平有效。当写信号为低电平时，用来通知 8251A，CPU 当前正在往 8251A 写入数据或控制信息。

4．数据总线缓冲

数据总线缓冲器也是三态的，双向，8 位缓冲器，经引脚 D_7～D_0 与系统的数据总线相连，是 8251A 与系统数据总线之间的接口。相关的引脚有：

D_7～D_0：数据线，三态，双向。实际上，数据线上不只传输一般的数据，也传输 CPU 对 8251A 的编程命令和 8251A 送往 CPU 的状态信息。

数据总线缓冲器包括：

（1）状态字缓冲寄存器。寄存 8251A 接收/发送操作的各种工作状态。

（2）发送数据缓冲寄存器。暂存由 CPU 送来的数据或控制字。8251A 没有独立的控制寄存器，写入的控制命令和发送的数据共用一个寄存器。

（3）接收数据缓冲寄存器。暂存接收到的准备送往 CPU 的数据。

与 $\overline{\text{RD}}$ 、 $\overline{\text{WR}}$ 、 C/$\overline{\text{D}}$ 相配合，具体的读写操作关系如表 5-2 所示。

表 5-2　$\overline{RD}$、$\overline{WR}$、C/$\overline{D}$ 的编码和对应的操作

C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	具体的操作
0	0	1	CPU 从 8251A 输入数据
0	1	0	CPU 往 8251A 输出数据
1	0	1	CPU 读取 8251A 状态
1	1	0	CPU 往 8251A 写入控制命令

5．调制解调控制逻辑

远程通信时，8251A 的 TxD 端数据经调制器调制后送传输线，传输线送来的信号经解调后送往 8251A 的 RxD 端。为了在 8251A 和调制解调器之间能正确的传送数据，8251A 调制解调控制逻辑产生 4 个相应的联络信号。

$\overline{DTR}$ ：数据终端准备好，输出，低电平有效；

$\overline{DSR}$ ：调制解调器准备好，输入，低电平有效；

$\overline{RTS}$ ：请求发送，输入，低电平有效；

$\overline{CTS}$ ：允许发送，输入，低电平有效。

当 8251A 不与调制解调器相接而是连接其他外设时，这 4 条线可以作为控制数据传输的联络线。

5.4.2　8251A 的编程

8251A 是一个可编程的多功能通信接口电路，在用它传送数据之前必须对它进行初始化编程，确定它的具体工作方式。改变 8251A 的工作方式，也必须要对其再次进行初始化编程。8251A 在编程时，CPU 发来的控制命令主要为工作方式字和工作命令字。

1．工作方式控制字

工作方式控制字在复位后写入，它详细规定了 8251A 的工作方式，其作用是对 8251A 的工作方式进行选择：异步/同步、数据格式、波特率系数，其格式如图 5-23 所示。

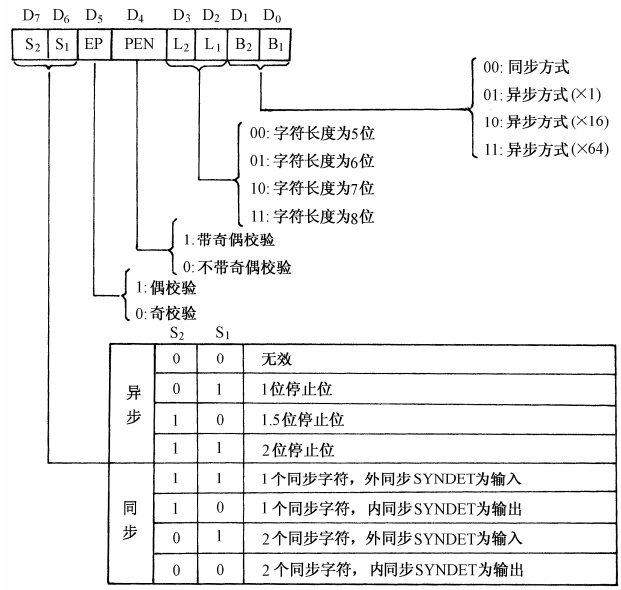


图 5-23　8251A 工作方式控制字格式

D_1D_0：确定是工作于同步方式还是异步方式。D_1D_0=00 为同步方式，当方式设为同步时，方式控制字后必须装入同步字符，并由同一个方式控制字规定装入单同步字符还是双同步字符；D_1D_0≠00 为异步方式，且有 3 种组合来选择输入的时钟频率与波特率之间的系数。

D_3D_2：确定每个字符的数据位（不包括奇/偶校验位）。

D_5D_4：确定是否校验和奇/偶校验的性质。

D_7D_6：含义因同步方式或异步方式而异。异步方式（D_1D_0≠00）时用来确定停止位个数。同步方式时 D_6 用来确定是内同步（SYNDET 脚为输出）还是外同步（SYNDET 为输入），D_7 用来确定同步字符个数。外同步方式时，同步字符只用于发送，接收时不作用。

例 5.7　某异步通信，数据位为 8 位，1 位起始位、2 位停止位、奇校验、波特率系数为 16。试编写其初始化程序。其工作方式控制字为 11011110B=0DEH，则其初始化程序为：

```
MOV   DX，309H    ；8251A 命令口
MOV   AL，0DEH
OUT   DX，AL
```

2．工作命令控制字

同步方式在同步字符之后（或异步方式在方式字后）接着写入命令控制字。命令字用于确定 8251A 的实际操作。写入方式字只规定了 8251A 的工作方式，并不能使 8251A 开始工作。只有写入命令字后才能使 8251A 处于相应的运行状态，接收或发送数据。异步方式在方式字之后（同步方式在同步字符之后）所写入的控制信息（$C/\overline{D}$=1）只能是命令控制字。只有在复位之后才能使 8251A 返回到写方式字状态，才能写入方式字改变 8251A 的工作方式。

工作命令控制字的格式如图 5-24 所示。

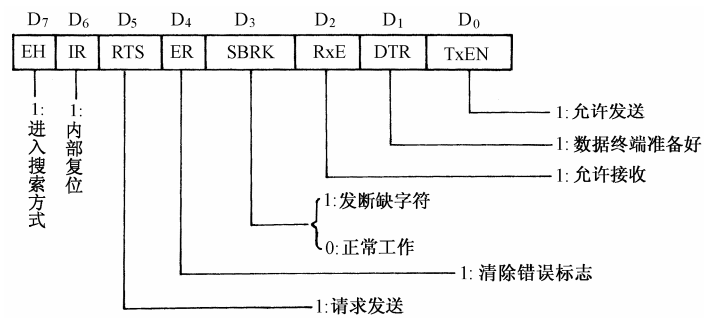


图 5-24　8251A 工作命令控制字格式

D_0：设置为 1 允许 8251A 开始发送操作。只有命令字的 D_0=1，引脚 TxDRY（通知 CPU 发送器准备好）才可能有效（为 1），可作为发送中断屏蔽位。

D_1：设置为 1 强制引脚 DTR 有效，表示数据终端准备好，通知调制解调器 8251A 已准备好。

D_2：设置为 1 允许 8251A 开始接收数据。只有命令字 D_2=1，RxRDY（通知 CPU 接收器准备好的引脚）才有可能为 1。允许接收时必须使错误标志复位。在同步方式时还必须指定进入同步搜索操作。

D_3：设置为 1 迫使 TxD 端发送低电平，以此作为断点字符。

D_4：设置为 1 则对状态字中的所有操作出错标志（FE、OE、PE）复位。

D_5：设置为 1 强制 $\overline{RTS}$ 引脚（请求发送）有效，向调制解调器提出发送请求。

D_6：设置为 1 强制 8251A 内部复位，使之回到准备接收方式字的状态。

D_7：只用于同步方式。为使 8251A 进入同步搜索操作，将输入的信息和同步字符比较，若一致则使 SYNDET/BRKDET（同步/断点检测）引脚有效，开始对数据做接收操作。

例 5.8　试编制程序，使 8251A 内部复位且允许接收/发送。

程序段如下：

```
MOV   DX，   309H
MOV   AL，   40H        ；D6=1，复位
OUT       DX，   AL
MOV   AL，   05H        ；D2=1，D0=1
OUT       DX，   AL
```

3．状态字

8251A 执行 CPU 命令进行数据传送后的状态字存放在状态寄存器中，CPU 可通过读入 8251A 的状态字进行分析和判断，以决定下一步该做什么。8251A 的状态字格式如图 5-25 所示。

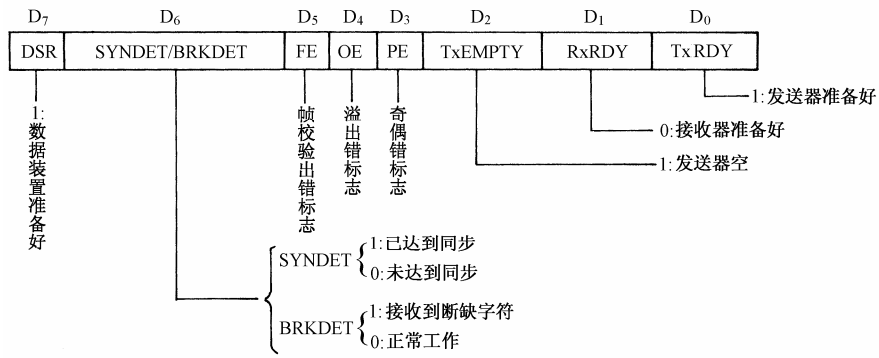


图 5-25　8251A 的状态控制字格式

状态字的作用是 8251A 向 CPU 送去数据传送操作中的各种状态信息。方式字、同步字符、命令字都是 CPU 写入 8251A 的，以控制 8251A 的工作方式和操作。那么，8251A 在发送、接收数据的过程中其实际工作状态如何呢？如何判断一个字符是否接收完全，接收的数据有没有错误？有什么类型的错误？发送缓冲器空了没有？发送移位寄存器空了没有等等，这些在发送/接收数据操作过程中的状态信息随时寄存在 8251A 的内部状态缓冲寄存器内，CPU 可以通过 I/O 读操作（$C/\overline{D}=1$）把状态字读入并加以分析，控制 CPU 和 8251A 之间的数据交换。

下面是状态控制字各位的含义：

D_0：TxRDY，发送器准备好；

D_1：RxRDY，接收准备好；

D_2：TxE，发送器空；

D_3：PE，奇/偶校验出错标志；

D_4：OE，溢出（覆盖）出错标志；

D_5：FE，帧格式出错标志；

D_6：SYNDET/BRKDET，同步检测；

D_7：DSR，数据装置准备好。

4．初始化编程

8251A 的方式字只是约定了双方的通信方式（同步/异步）、数据格式（数据位、停止位长度、校验特性、同步字符特性等）及传输速率（波特率因子）等参数，但没有规定传送的方向（发/收），故须由命令字来控制发/收。何时发/收取决于 8251A 的工作状态，即状态字。只有 8251A 进入发/收准备好的状态，才能真正开始数据的传送。

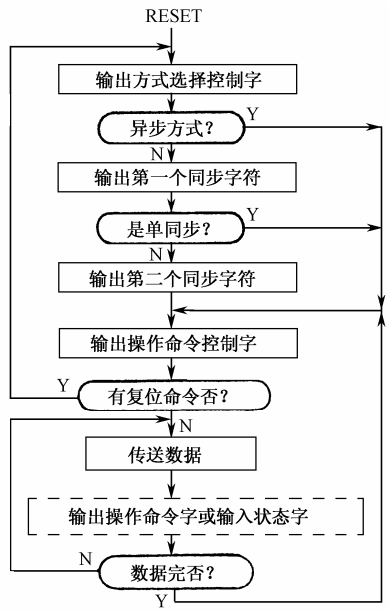


图 5-26　8251A 初始化流程图

方式字和命令字均无特征位标志，并且都是送到同一个命令口地址，所以，在向 8251A 写入方式字和命令字时，须按一定的顺序，这个顺序不能颠倒或改变，若改变了，就会使 8251A 不能正常工作。因此，8251A 只能根据写入的顺序来识别控制字和种类，这个顺序就是 8251A 的初始化流程。初始化编程写入控制字的顺序如图 5-26 所示。

例 5.9　8251A 工作于异步方式，方式选择控制字为 11111011B，工作命令控制字为 00010001B。试编写其初始化程序。

其初始化程序为：

```
    …
MOV   AL，      0FBH      ；8251A 方式选择字
OUT       CONTR，    AL
MOV   AL，      11H       ；8251A 操作命令字
OUT       CONTR，    AL
    …
```

5.4.3　8251A 的应用举例

在微机系统中多使用异步通信方式。下面以微机系统中两台微机之间进行双机串行通信的硬件连接和软件编程来说明 8251A 在实际中的应用。

在甲乙两台微机之间进行串行通信，甲机发送，乙机接收。要求把甲机上的数据（其长度为 2DH）传送到乙机中去。双方采用起止式异步方式，通信的数据格式为：字符长度为 8 位，2 位停止位，波特率因子为 64，无校验，波特率为 4 800。CPU 与 8251A 之间用查询方式交换数据，8251A 的端口地址分配是 309H，为命令／状态口，308H 为数据口。

由于是近距离传输，因此可以不设 Modem，两台微机之间直接通过 RS-232 标准接口连接即可，采用查询 I/O 方式，故收/发程序中只需检查发/收准备好的状态是否置位，即可收发 l 个字节。

整个设计工作分为两部分：硬件设计和软件设计。

1．硬件连接

根据分析，把两台微机都当做 DTE（数据终端设备），采用最简单的发送线 TxD、接收线 RxD 和地线 GND 3 根线连接就能进行通信了。采用 8251A 作为接口的主芯片再配置少量附加电路，如波特率发生器、RS-232C 与 TTL 电平转换电路、地址译码电路等就可构成一个串行通信接口，如图 5-27 所示。

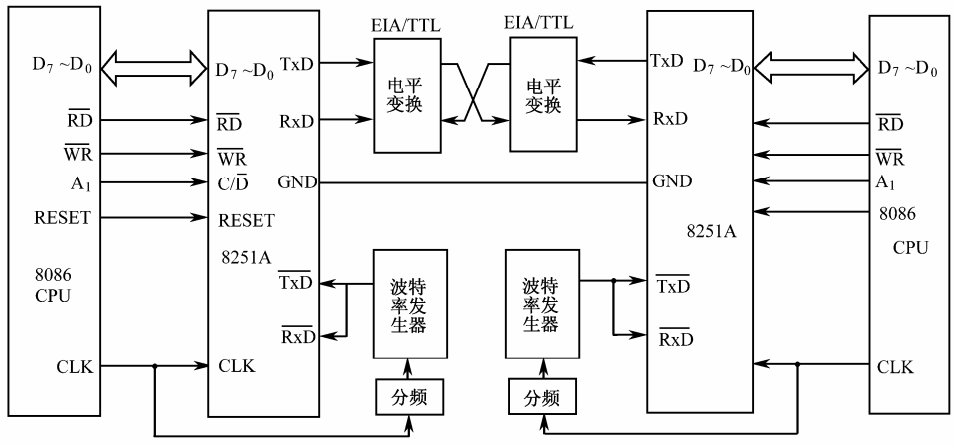


图 5-27　双机串行通信接口

2．软件编程

由题意可知，接收和发送程序应分别编写，每个程序段中包括 8251A 初始化、状态查询和输入/输出几部分。

对接收/发送方的 8251A 初始化时，首先要确定其方式选择控制字和工作命令控制字。根据题中的要求有：

发送方的方式选择控制字为 11001111B=CFH，工作命令控制字为 00111110B=37H；

接收方的方式选择控制字为 11001111B=CFH，工作命令控制字为 00010100B=14H。

发送端的发送程序（略去 STACK 和 DATA 段）为：

```
CSEG    SEGMENT
        ASSUME CS：CSEG
TRA     PROC FAR
START： MOV     DX，309H        ；控制口
        MOV     AL，00H         ；空操作
        OUT     DX，AL
        MOV     AL，40H         ；内部复位
        OUT     DX，AL
        NOP
        MOV     AL，0CFH        ；方式字（异步，2 位停止位，字符长度为 8 位，无
                                  校验，波特率因子为 64）
```

```
        OUT     DX，AL
        MOV     AL，37H          ；命令字（RTS 、ER、RxE、 DTR 和 TxEN 均置 1）
        OUT     DX，AL
        MOV     CX，2DH          ；传送字节数
        MOV     SI，300H         ；发送区首地址
L1：    MOV     DX，309H         ；状态口
        IN      AL，DX           ；查状态位 D0（TxRDY）是否为 1
        TEST    AL，38H          ；查错误
        JNZ     ERR              ；转出错处理
        AND     AL，01H
        JZ      L1               ；发送未准备好，则等待
        MOV     DX，308H         ；数据口
        MOV     AL，[SI]         ；发送准备好，则从发送区取 1 字节发送
        OUT     DX，AL
        INC     SI               ；修改内存地址
        DEC     CX               ；字节数减 1
        JNZ     L1               ；未发送完，继续
ERR：   （略）
        MOV     AX,4C00H         ；已送完，返回 DOS
        INT     21H
        TRA     ENDP
        CSEG    ENDS
END     START
```

接收方接收程序（略去 STACK 和 DATA 段）：

```
SCEG    SEGMENT
        ASSUME CS：REC
REC     PROC FAR
BEGIN： MOV     DX，309H         ；控制口
        MOV     AL，0AAH         ；空操作
        OUT     DX， AL
        MOV     AL，50H          ；内部复位
        OUT     DX， AL
        NOP
        MOV     AL，0CFH         ；方式字
        OUT     DX， AL
        MOV     AL，14H          ；命令字（ER、RxE 置 1）
        OUT     DX， AL
        MOV     CX， 2DH         ；传送字节数
        MOV     DI， 400H        ；接收区首址
```

```
L2：    MOV     DX，   309H        ；状态口
        IN      AL，   DX          ；查状态位 D2（RxRDY）＝17
        TEST    AL，   38H         ；查错误
        JNZ     ERR                ；转出错处理 .
        AND     AL，   02H
        JZ      L2                 ；接收未准备好，则等待
        MOV     DX，   308H        ；数据口
        IN      AL，   DX          ；接收准备好，则接收 1 个字节
        MOV     [DI]， AL          ；并存入接收区
        INC     DI                 ；修改内存
        LOOP    L2                 ；未接收完，继续
ERR：   （略）
        MOV     AX，   4C00H       ；已接收完，程序结束，退出
        INT     21H                ；返回 DOS
REC     ENDP
CSEG    ENDS
END BEGIN
```

5.5 可编程中断控制8259A

8259A 可编程中断控制器 PIC（Programmable Interrupt Controller）又称优先级中断控制器，具有多种工作方式，并可通过编程来加以选择。它可为 CPU 管理和处理 8 级矢量优先级中断，即单片可管理 8 级中断；与其他 8259A 芯片级联，可用 9 片构成多达 64 级主从式中断系统，从扩大中断功能；优先级方式在执行主程序的任何时间里都能够动态地改变，无论 8086CPU 工作在最小模式还是最大模式，都可以与之配套使用；能为 CPU 提供中断类型号，使 CPU 在中断响应过程中根据 8259A 提供的中断类型号找到中断服务程序的入口地址来实现程序转移。

5.5.1 8259A 的内部结构及引脚

8259A 的内部结构框图如图 5-28 所示，它由以下 8 部分组成：

（1）中断请求寄存器（IRR）。中断请求寄存器 IRR 是一个具有锁存功能的 8 位寄存器，该寄存器用来存放由外部输入的中断请求信号 IR_7～IR_0，当某个输入端为高电平时该寄存器的相应位置“1”。

（2）中断服务寄存器（ISR）。该寄存器是一个 8 位寄存器，与 8 级中断 IR_7～IR_0 相对应，用来记录正在处理中的中断请求，当任何一级中断被响应，CPU 正在执行它的中断服务程序时，ISR 寄存器中的相应位置“1”，一直保持到该级中断处理过程结束为止。多重中断情况下，ISR 寄存器中可有多位被同时置“1”。

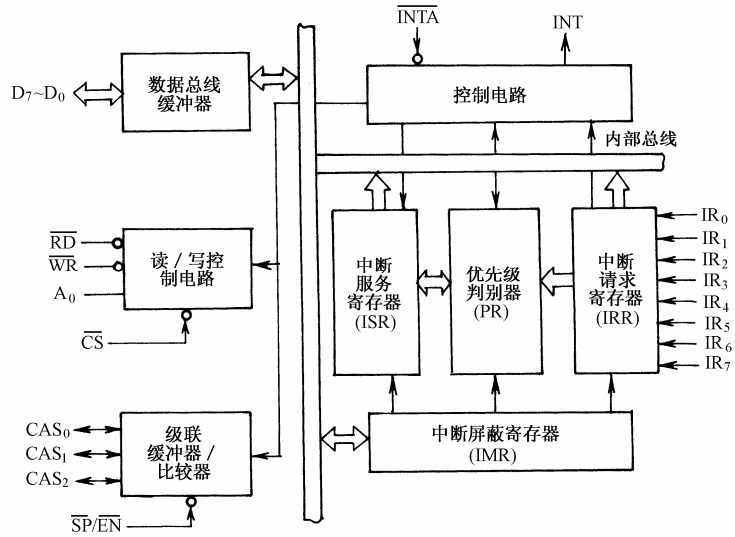


图 5-28　8259A 的内部结构框图

（3）优先级判别器（PR）。优先级判别器 PR 也叫优先级分析器，用来管理和识别各中断申请信号的优先级别。当输入端 IR_7～IR_0 中有多个中断请求信号同时产生时，由 PR 判断哪个中断请求具有最高优先级，并在 INTA 脉冲期间把它置入中断服务寄存器 ISR 的相应位。

（4）中断屏蔽寄存器（IMR）。IMR 寄存器是一个 8 位寄存器，与 8259A 处理的 8 级中断 IR_7～IR_0 相对应，该寄存器可对各个中断源进行屏蔽或开放。当某位置为“1”时，表示相应的中断源被屏蔽（禁止），为“0”则表示允许中断。

（5）级联缓冲器/比较器。级联缓冲器/比较器用于存储并比较系统中所用的全部 8259A 的输入信号，以实现多达 8 片的 8259A 的级联。一片 8259A 只能接收 8 级中断，当超过 8 级时，可用多片 8259A 级联使用构成主从关系。对于主 8259A，其级联信号 CAS_2～CAS_0 是输出信号，而对从 8259A，级联信号 CAS_2～CAS_0 是输入信号。此时，主 8259A 的 SP 端为“1”，从 8259A 的 SP 端为“0”，且从 8259A 的 INT 输出接到主 8259A 的中断输入端 IR 上，因而可把中断扩展到 64 级。

（6）控制电路。8259A 内部的控制电路，根据中断请求寄存器 IRR 的位置情况和优先级判别器 PR 的判定结果，向 8259A 内部其他部件发出控制信号，并向 CPU 发出中断请求信号 INT 和接收来自 CPU 的中断响应信号 $\overline{INTA}$，控制 8259A 进入中断服务状态。

（7）读/写控制逻辑。读/写控制逻辑接收 CPU 送来的读/写命令 $\overline{RD}$、$\overline{WR}$，片选信号 CS 以及端口选择信号 A_0，以实现 CPU 对 8259A 的读/写操作。因一片 8259A 中占两个 I/O 端口地址，用地址线 A_0 来选择端口，端口地址的高位由片选信号端 $\overline{CS}$ 输入。由读信号 $\overline{RD}$ 和写信号 $\overline{WR}$ 控制可将命令写入有关的控制寄存器，或读出内部寄存器的内容。

（8）数据总线缓冲器。数据总线缓冲器是一个双向 8 位三态缓冲器，由它构成 8259A 与 CPU 之间的数据接口。CPU 向 8259A 发送的数据、命令、控制字及 8259A 向 CPU 输送的数据、状态信息都要经过数据总线缓冲器。

至于 8259A 芯片的引脚，可分为以下 4 类：

（1）与外部设备连接的中断请求输入引脚 IR_0～IR_7；

（2）与 CPU 连接的数据通路和控制信号：D_0～D_7、$\overline{WR}$、$\overline{RD}$、$\overline{INTA}$、INT；

（3）用于 8259A 级联的引脚 CAS_0～CAS_2、$\overline{SP}/\overline{EN}$；

（4）端口地址选择信号 $\overline{CS}$、A_0。

8259A 芯片的引脚分配如图 5-29 所示。

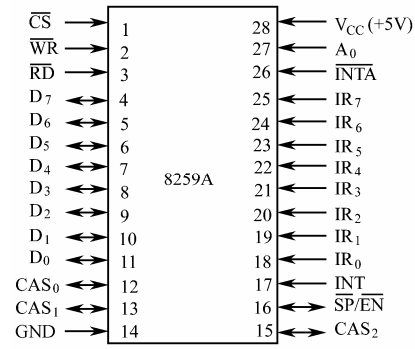


图 5-29　8259A 芯片的引脚

5.5.2　8259A 的工作方式

8259A 具有非常灵活的中断管理工作方式，这些工作方式都可以通过编程方法进行设置，可满足使用者的不同要求。

1．中断优先级设置方式

8259A 对中断优先级的管理是中断管理的核心问题，8259A 对中断优先级的管理可分为 4 种情况：完全嵌套方式、自动循环方式、中断屏蔽方式和特殊完全嵌套方式。

（1）完全嵌套方式。完全嵌套方式是 8259A 最常用和最基本的工作方式，如果对 8259A 进行初始化后没有设置其他优先级方式，则 8259A 就自动按完全嵌套方式工作。在此种方式下，8259A 的中断请求输入端引入的中断具有固定的优先级序列，IR_0 为最高优先级，IR_7 为最低优先级。高优先级的中断可中断低优先级的中断服务，从而实现嵌套中断。

（2）特殊完全嵌套方式。特殊完全嵌套方式和完全嵌套方式基本相同，不同之处在于：在特殊完全嵌套方式下，当处理某一级中断时，如果有同级的中断请求，也会给予响应，从而实现一种对同级中断请求的特殊嵌套。特殊完全嵌套方式用在 8259A 有级联的情况下，当任何一个 8259A 从片接收到一个中断请求且经本 8259A 判别确定为当前最高优先级时，则响应这一中断，通过 INT 端向 8259A 主片相应的 IR 端提出中断请求。如果这时 8259A 主片中 ISR 相应位已置“1”，则说明该 8259A 从片的其他输入端已提出过申请，且正在接受服务。8259A 从片的判优电路判别到刚申请的中断优先级最高，故应停止现行中断服务去为刚申请的中断服务。在 8259A 有级联的情况下，按完全嵌套方式管理优先级。显然接在主片 IR_3 上的从片比接在 IR_4 上的从片具有更高的优先级，而主片上 IR_0、IR_1、IR_2 上的中断比接在主片 IR_3 上的从片具有更高的优先级。

（3）优先级自动循环方式。在完全嵌套方式下，中断请求 IR_0～IR_7 的优先级别是固定不变的，使得从 IR_0 引入的中断总是具有最高优先级，在某些情况下可采用自动循环方式改变这种优先级别。在这种方式下，IR_0～IR_7 引入的中断轮流为最高优先级，任何一级中断被处理完后，它的优先级别就被修改为最低，而最高优先级分配给该中断的下一级中断。例如，现正为 IR_3 引入的中断服务，若服务完毕，则 IR_3 为最低优先级，IR_4 为最高优先级，IR_5 为次高优先级，依次排列。这种方式一般用在系统中多个中断源优先级相等的场合。

（4）优先级特殊自动循环方式。这种方式和优先级自动循环方式的不同在于：在优先级特殊自动循环方式中，一开始的最低优先级是由编程确定的，从而整个优先级顺序也由此而定，而不是像优先级自动循环方式中固定为 IR_7。例如，若设定 IR_2 为最低级，则 IR_3 就为最高级，其他依次类推。

2．中断结束方式

当 8259A 响应某一级中断而为其服务时，中断服务寄存器 ISR 的相应位置“1”，表示正在对外服务，同时也为中断优先级判别器 PR 提供判别依据。当有更高级的中断申请进入时，

ISR 的相应位又要置“1”，因此 ISR 寄存器中可有多位同时置“1”。

在中断服务结束时，ISR 中相应位应清“0”，以便再次接收同级别的中断。中断结束的管理就是用不同的方式使 ISR 中相应位清“0”，否则 8259A 的中断控制功能就会不正常。

8259A 中断结束方式可分为 3 种情况：

（1）自动中断结束方式。在自动中断结束方式中，系统一进入中断过程，8259A 在 CPU 中断响应总线周期的第二个中断响应信号 $\overline{\text{INTA}}$ 结束后，自动将 ISR 寄存器置“1”位清“0”。此时，该中断服务程序还在进行，但对 8259A 来说，它对本次中断的控制已经结束，这是一种最简单的中断结束方式。但是这种方式存在一个明显的缺点：任何一级中断在执行中断服务程序期间，在 8259A 中没有留下任何标志。如果在此过程中出现了新的中断请求，则只要 CPU 允许中断，不管出现的中断级别如何，都将打断正在执行的中断服务而被优先执行，从而产生重复嵌套，并且嵌套的深度也无法控制，这显然是不合理的。因此，只有在一些以预定速率发生中断，且不会发生同级中断互相打断或低级中断打断高级中断的情况下才能使用自动中断结束方式。

（2）普通中断结束方式。任何一级中断服务程序结束时，都会给 8259A 传送一个中断结束命令，8259A 将 ISR 寄存器中级别最高的置“1”位清“0”，表示当前正在处理的中断已经结束。这种方式只有在当前结束的中断确实是在尚未处理完的级别最高的中断时，才能使用。因此，此方式适用于全嵌套工作方式。

（3）特殊中断结束方式。由于在非全嵌套方式（如优先级自动循环、特殊循环等方式）下，无法根据 ISR 寄存器的内容来确定哪一级中断是最后响应和处理的，即无法从 ISR 的置“1”位置上确定当前的最高优先级，从而也就无法确定应将哪个置“1”位清“0”而结束中断。特殊的中断结束方式，就是 CPU 在中断服务程序结束时给 8259A 发出特殊中断结束命令的同时，将当前结束的中断级别也传送给 8259A，这被称做特殊的中断结束方式。在这种情况下，8259A 将 ISR 寄存器中指定级别的相应置“1”位清“0”。

3．屏蔽中断源的方式

CPU 可用 CLI 指令将 IF 清“0”，可以禁止所有的可屏蔽中断进入。但是，要想屏蔽某个或某几个中断源，就得用 8259A 的中断优先级管理的屏蔽方式来实现。8259A 对中断源的屏蔽方式有以下两种方法。

（1）普通屏蔽方式。在普通屏蔽方式下，CPU 向 8259A 的中断屏蔽寄存器 IMR 中发一个屏蔽字，若屏蔽字的某一位或几位为“1”，则与这些位相对应的中断源就被屏蔽，相应的中断请求就不能传到 PR，从而也就不能传到 CPU。屏蔽字中为“0”的位表示对应的中断源被允许。

（2）特殊屏蔽方式。特殊屏蔽方式通常用于多级中断嵌套中。采用这种方式是为了提高系统的实时性，临时改变固定的嵌套顺序，允许优先级别低的中断服务程序中断优先级别高的中断服务程序，即实现优先级的动态改变。这时可以使低优先级别的中断进入正在服务的高优先级别中。

5.5.3 8259A 的编程

在使用 8259A 时，除按各引脚规定的信号接好电路外，还必须用程序选定其工作状态，例如各中断请求信号的优先级分配、中断屏蔽、中断矢量等等。每一种状态都由一个命令字

或一个命令字中的某些位来规定。

8259A 的命令字分为初始化命令字 ICW（Initialization Command Word）和工作命令字 OCW（Operation Command Word）两种，因此 8259A 的编程也分为初始化编程和工作编程两步。在 8259A 内部，有相应的一组寄存器分别将这些命令字锁存，以控制其工作。

1．8259A 的初始化编程

8259A 必须先进行初始化编程，后进行工作编程。初始化命令共预置了 4 个命令字 ICW_1～ICW_4。8259A 的初始化编程完成的主要功能是：

（1）设定中断请求信号的触发方式，即高电平或上升沿触发；

（2）设定 8259A 的连接方式为单片或级联；

（3）设定 8259A 的中断类型码基值，即 IR_0 所对应的中断类型号；

（4）设定 8259A 的中断优先级管理方式；

（5）设定中断结束时的处理方式。

初始化命令字必须顺序填写，一旦发出就不能改变，工作后一般不再重复写。初始化编程由写入 ICW_1 开始，然后写入 ICW_2。至于是否写入 ICW_3 和 ICW_4 取决于工作方式（ICW_1 的有关位）。8259A 有两个端口，一个为偶地址，一个为奇地址。下面分析每个初始化命令字及其各位的作用。

（1）ICW_1——芯片控制初始化命令字。在地址 A_0=0 时，若对 8259A 写入一个 D_4=1 的字节，则启动了其初始化编程。写入的这个字节被当成 ICW_1，D_4=1 是其特征位，以区别 OCW_2 和 OCW_3 控制字的设置。其余各位的作用如图 5-30 所示。

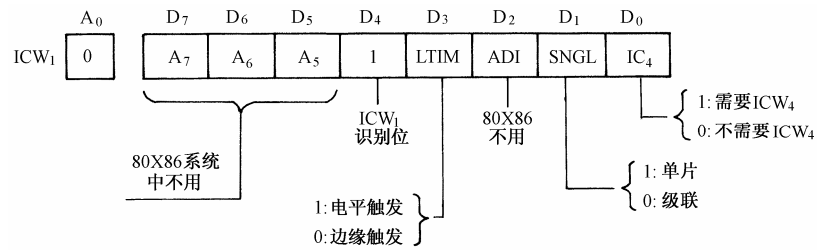


图 5-30　ICW_1 的作用

A_0=0，写入命令字的端口地址，ICW_1 必须写入 8259A 的偶地址端口。

D_7～D_5 和 D_2 这 4 位仅对 8080/8085 系统有意义（填 0）。

D_0=1 表示要送 ICW_4，否则不送。在不送 ICW_4 时，ICW_4 的各位默认值均为零。

D_1=0 表示系统中有多片 8259A 级联，否则表示单片工作，此时不送 ICW_3。

D_3 规定 IR_7～IR_0 信号的触发方式。D_3=1 为高电平触发，否则为上升沿触发。

写入 ICW_1 时，还自动将中断屏蔽寄存器 IMR 清零，并恢复各中断源的优先级为 IR_0 最高，IR_7 最低。

（2）ICW_2——设置中断类型码基值命令字。ICW_2 是中断矢量基值寄存器，用于设置中断类型号的命令字。ICW_2 的作用如图 5-31 所示。

A_0=1，ICW_2 必须写入 8259A 的奇地址端口中。

在工作于 8086/8088 系统中时，D_7～D_3 表示中断类型码的高五位。D_2～D_0 不需编程，其固定值为 000，在响应时由中断源序号填入相应值。

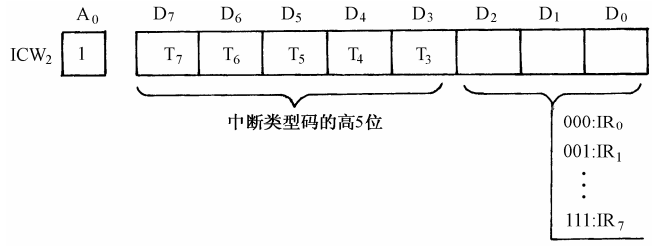


图 5-31　ICW_2 的作用

（3）ICW_3——标识主片/从片初始化命令字。ICW_3 是 8259A 的级联命令字，用于主片/从片的初始化命令字。单片 8259A 工作时不需写入，多片 8259A 级联时，有主片和从片之分，需要分别写入 ICW_3。主片/从片格式如图 5-32 所示。

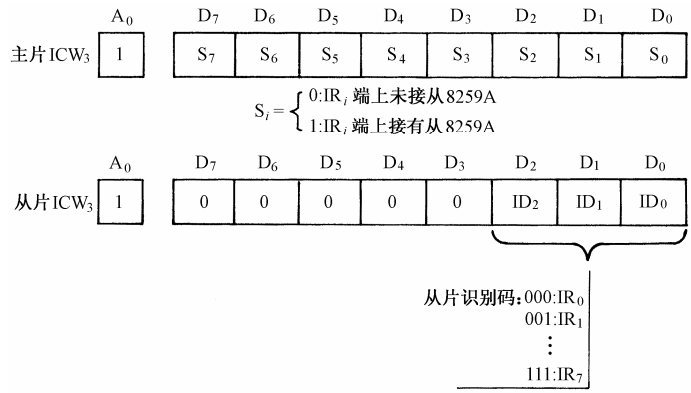


图 5-32　ICW_3 的作用

A_0=1，ICW_3 必须写入 8259A 的奇地址端口中；

主片 ICW_3 的 D_7～D_0 对应其 8 条中断请求线 IR_7～IR_0，若某根 IR 线上接有从 8259A 片，则 ICW_3 的相应位应写成 1，否则写 0。

各从片的 ICW_3 仅 D_2～D_0 有意义，作为其从片标识码，高位固定为 0，这个从片标识码必须和本片所接主片 IR 线的序号一致。

在中断响应时，主片通过级联线 CAS_2～CAS_0 送出被允许中断的从片标识码。各从片用自己的 ICW_3 和 CAS_2～CAS_0 比较，二者一致的从片被确定为当前中断源，才可发送自己的中断矢量。

例如，对主片，当 ICW_3=0F0H 时，表示 IR_7～IR_4 接有从片，而 IR_3～IR_0 未接从片。对从片，当 ICW_3=02H 时，表示从片接到主片的 IR_2（010）上。

（4）ICW_4——方式控制初始化命令字。ICW_4 是方式控制初始化命令字，当 ICW_1 中的 D_0 为 1 时，在初始化时需要设置 ICW_4。ICW_4 的作用如图 5-33 所示。

A_0=1，ICW_4 必须写入 8259A 的奇地址端口中。

D_7～D_5 这 3 位总为 0，用来表示 ICW_4 的标识码。

D_4 指定了中断的嵌套方式。D_4=0 为一般嵌套方式，当某个中断正在服务时，本级中断及更低级的中断都被屏蔽，只有更高的中断才能响应。对于单片 8259A 的中断系统，这种安排没有问题。但对多片 8259A 级联组成的中断系统，当某从片中的一个中断正在服务时，主片即将这个从片的所有中断屏蔽，因此即使本从片中有比正在服务的中断级别更高的中断源

发出请求，也不能得到响应，即不能进行中断嵌套。D_4=1 则是特殊嵌套方式，仅仅屏蔽比当前中断源级别低的中断，于是上述情况就可以进行中断嵌套。

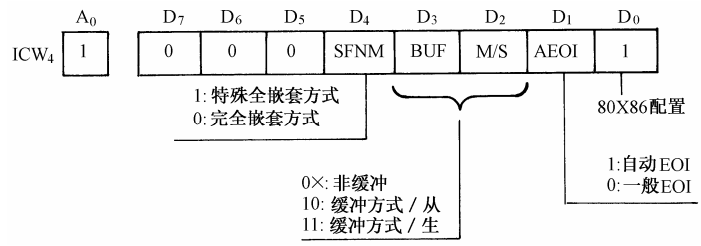


图 5-33　ICW_4 的作用

D_3 为数据缓冲选择。D_3=1 时 8259A 的数据线和系统总线之间要加上三态缓冲器，此时 8259A 的 $\overline{SP}/\overline{EN}$ 引脚变成输出线，作为缓冲器的选通信号。每当 8259A 的数据送往系统总线时，$\overline{SP}/\overline{EN}$ 引脚输出有效的低电平。这种情况用于多片 8259A 的级联，此时主从片的区分就不能再靠 SP/EN 固定接高或低电平了，而是使用 ICW_4 的 D_2 位。规定主片的 D_2=1，从片的 D_2=0；D_3=0 时不加缓冲器，D_2 无意义。

D_1 说明了中断结束的方式。D_1=0 是正常方式，即在中断服务结束时，CPU 向 8259A 送一个 EOI 命令字（OCW_2），于是中断服务寄存器 ISR 中与中断源相对应的位被清除。D_1=1 是自动 EOI 方式，即在中断响应时，在 8259A 送出中断矢量后自动将 ISR 相应位复位。

D_0 指定了系统中所用 CPU 的模式。D_0=0 时用 8080/8085CPU；D_0=1 时用 8086/8088CPU。

若在某种应用场合，正好需要 ICW_4 各位都为 0，则可以不写 ICW_4。因为 8259A 在进入初始化时，已自动将 ICW_4 全部复位。

2．8259A 的工作编程

8259A 在初始化编程后，应再进行工作编程，即写入工作命令字，用于对中断处理过程进行动态控制。工作命令字有 3 个，它们各有自己的特征位，因此对写入的顺序没有要求。在中断系统的工作中，某些工作命令字可能需要重复多次地写入。

（1）OCW_1——中断屏蔽操作命令字。OCW_1 用来实现对中断源的屏蔽功能，OCW_1 的内容直接写入屏蔽寄存器 IMR 中，OCW_1 的作用如图 5-34 所示。

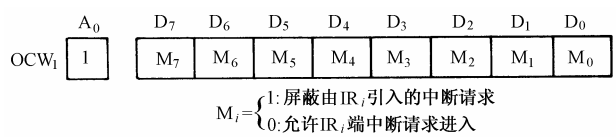


图 5-34　OCW_1 的作用

A_0=1，表示 OCW_1 必须写入 8259A 的奇地址端口中（3 个工作命令字中仅 OCW_1 占有奇地址）。

M_7～M_0 对应于中断屏蔽寄存器（IMR）各位，其每一位控制一根中断请求输入线，屏蔽字为“1”的位所对应的中断请求线被屏蔽；否则，被允许（在初始化开始时，默认屏蔽字各位全为 0）。

（2）OCW_2——优先级循环方式和中断结束方式操作命令字。OCW_2 的主要作用是设置中断结束方式（包括一般结束 EOI 和特殊结束 SEOI）和控制中断优先级的循环方式，如图 5-35

所示。

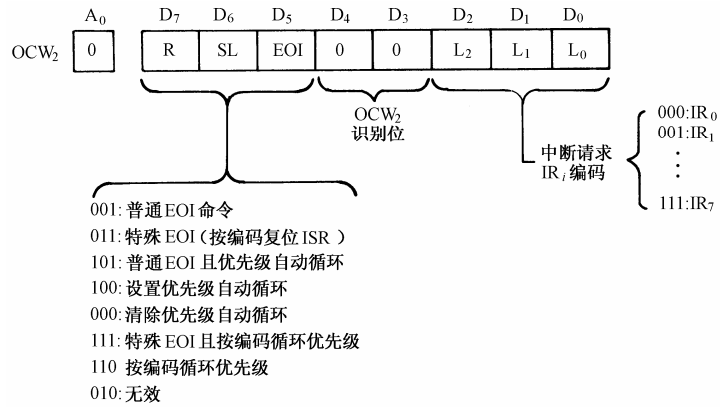


图 5-35　OCW_2 的作用

A_0=0 表示 OCW_2 应写入偶地址端口。

D_4、D_3 是 OCW_2 的标志位，它虽然和 OCW_3 都占有偶地址（A_0=0），但其特征为 D_4D_3=00，因此不会发生混淆。

D_7 位表示中断优先级循环。当 D_7=0 时，8 个中断请求 IR_7～IR_0 的优先级固定不变，IR_7 最低，IR_0 最高；当 D_7=1 时，优先级可以循环，即 IR_7 和 IR_0 首尾相接成一闭环，各级的优先级在其中循环移位，循环到什么情况停止还与其他位有关。

D_6 位表示特殊循环。当 D_6=1 时，最低 3 位 D_2～D_0 的二进制编码指定了一条外部中断请求线 IR_i（$0 \le i \le 7$）。此时若 D_7=1，则称为特殊循环，即优先级的循环移位一直进行到最低优先级对准 IR_i 为止，于是最高优先级也就移到 $IR_{(i-1)}$（若 i=0，则 i–1=7）。D_6=0 时，最低优先级自动循环到当前服务的中断请求线，D_2～D_0 无意义。

D_5 位是中断结束位。D_5=1 表示中断结束（EOI 命令）。当用 8259A 来实现中断管理时，中断服务程序结束时（返回指令 IRET 前）必须给 8259A 送一条 EOI 命令（即 D_5=1 的 OCW_2）。8259A 收到这条命令后，将中断服务寄存器中的相应位清除，然后才为其他中断源服务。若 D_6D_5=11，则称为特殊的中断结束（SEOI 命令），它将复位 ISR 中由 OCW_2 的 D_2～D_0 编码指定的位。

（3）OCW_3——特殊屏蔽方式和中断查询方式操作命令字。OCW_3 的作用是写入多功能操作命令字。OCW_3 有 3 项功能：设置/取消特殊屏蔽方式；设置中断查询方式；设置对 8259A 内部寄存器的读出命令。其各位的作用如图 5-36 所示。

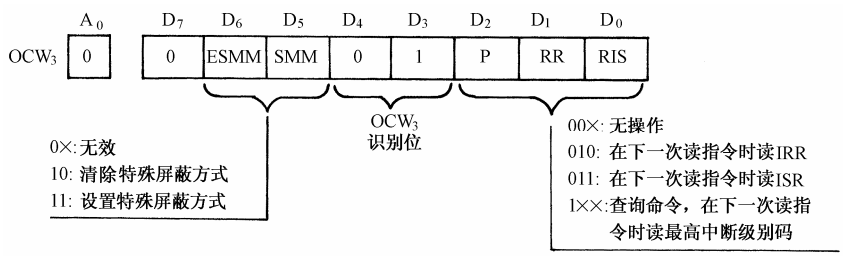


图 5-36　OCW_3 的作用

A_0=0，OCW_3 必须写入 8259A 的偶地址端口。

D_4D_3=01，写入 OCW_3 的地址和 OCW_2 相同，但其特征为 D_4D_3=01，以区别于 OCW_2。

OCW_3 经常用来配合读 8259A 内部寄存器的内容。D_1D_0=10，则将读入其中断请求寄存器的内容；若 D_1D_0=11，则读入的是中断服务寄存器的状态。

OCW_3 中的 D_2 位表示查询。8259A 也可以不工作于中断方式，而工作于查询方式，此时应写入 D_2=1。CPU 可以反复对 8259A 查询，但每次查询前都应先写相应的 OCW_3 命令字。

OCW_3 的 D_6D_5 用来控制特殊屏蔽功能。当 D_6D_5=11 时，设置特殊屏蔽；当 D_6D_5=10 时，清除特殊屏蔽。假设一个优先级较高的中断源正处在服务的过程中，若设置了特殊屏蔽功能，则允许优先级较低的中断源产生中断嵌套。

5.5.4 8259A 的应用

例 5.10 在某个 CPU 系统中接有一片 8259A，有一外设中断请求从 IR_7 引入，8259A 的端口地址如图 5-37 所示，试编写其初始化程序。

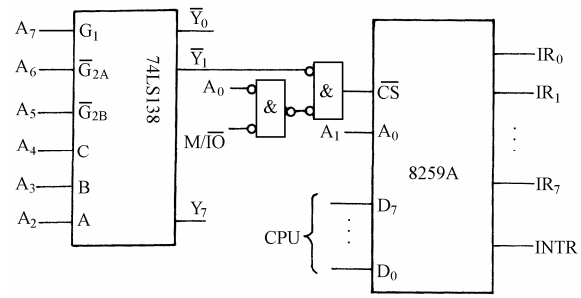


图 5-37 8259A 端口地址形成电路

从图 5-37 可知，8259A 具有两个端口地址，由 CPU 的地址线 A_1 控制，其端口地址的高位由译码器的输入端 Y_1 提供，组合逻辑电路可使其 8259A 为偶地址，因而 8259A 的命令和类型号等可由 8086CPU 的低 8 位数据线传送。

设中断为边沿触发，从 IR_7 引入中断的中断类型号为 C7H，端口地址为 84H 和 86H，8086CPU 获得中断类型号并乘以 4，到中断服务程序入口表中找到相应的中断服务程序入口地址，而后转入中断服务。

初始化程序包括对 8259A 的预置，以及将中断服务程序首地址填入中断矢量表中。初始化程序如下：

```
INTRRUP     SEGMENT AT 0
            ORG 0C7H*4
INTC7       LABEL DWORD
                …
            MAIN SEGMENT
                …
            CLI                         ；关中断
            MOV   AL，13H               ；写 ICW1，单片，边缘触发，要写 ICW4
            OUT         84H，AL
            MOV   AL，0C7H              ；写 ICW2
            OUT         86H，   AL
```

```
        MOV   AL，    01              ；写 ICW4
        OUT         86H，   AL
        STI                           ；开中断
        …
MAIN ENDS
        …
```

例 5.11　IBM PC/XT **系统中的** 8259A **的初始化编程。**

IBM PC/XT **系统启动时执行** BIOS **中的系统初始化程序，其中包括** 8259A **的编程（初始化编程和工作编程）。使用要求是：单片** 8259A **管理** 8 **级硬件中断，中断申请信号采用边沿触发，采用完全嵌套方式，**IR_0 **最高，**IR_7 **最低，中断类型码为** 08H～0FH**，非自动中断结束方式，端口地址为** 20H **和** 21H**。**

根据上述要求，8259A **在** IBM PC/XT **系统中的硬件连接如图** 5-38 **所示。由于是单片，所以级联信号** CAS_2～CAS_0 **可不用，** $\overline{SP}/\overline{EN}$ **接**+5V**。**

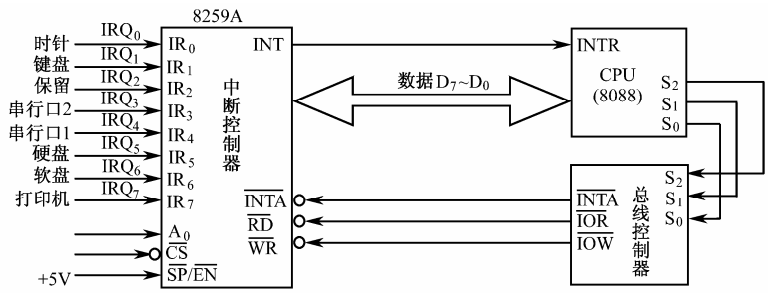


图 5-38　PC/XT **中** 8259A **的硬件连接**

根据上述硬件连接，8259A **初始化编程的有关部分程序如下：**

```
MOV   AL，    13H             ；写 ICW1，单片，边缘触发，要写 ICW4
OUT         20H，   AL
MOV   AL，    08H             ；写 ICW2，中断类型号为 08H
OUT         21H，   AL
MOV   AL，    09H             ；写 ICW4，8086/8088 系统，非自动结束方式
OUT         21H，   AL
MOV   AL，    0FFH            ；写 OCW1，屏蔽所有硬中断
OUT         21H，   AL
      …
```

程序中对 8259A **写入的** ICW_1=13H=00010011B**，表明外部中断请求信号为上升沿有效，单片** 8259A **工作，且后面还要写** ICW_4**。写入的** ICW_2 **是中断矢量，现为** 08H**，实际上** 8 **个中断源各自填入** D_2～D_0 3 **位，形成** 08H～0FH8 **个矢量。**ICW_4=09H=00001001B**，指定了系统中的** CPU **为** 8086/8088**，中断过程不自动结束，应写入一个含** EOI **命令的** OCW_2 **结束，数据线上有缓冲器，且取一般中断嵌套方式。最后送入** OCW_1=0FFH**，屏蔽所有硬中断，因为系统尚未初始化完毕，不能接收任何中断。**

5.6 DMA控制器Intel 8237A

为了能够实现高速率传送数据，人们提出了直接存储器存取方式（DMA）。8237A 是高性能的可编程 DMA 控制器，工作在 5MHz 时钟下的 8237A-5 传输速率可达 1.6MB/s。每片 8237A 内部有 4 个独立的通道，每个通道寻址及字节计数都可达 64KB。它们可以分时地为 4 个外部设备实现 DMA 操作，可用来实现内存到端口、端口到内存及内存到内存之间的高速数据传送。

5.6.1 8237A 的结构和引脚

1．8237A 的内部结构

8237A 的内部结构如图 5-39 所示，它主要由 5 个部分组成。

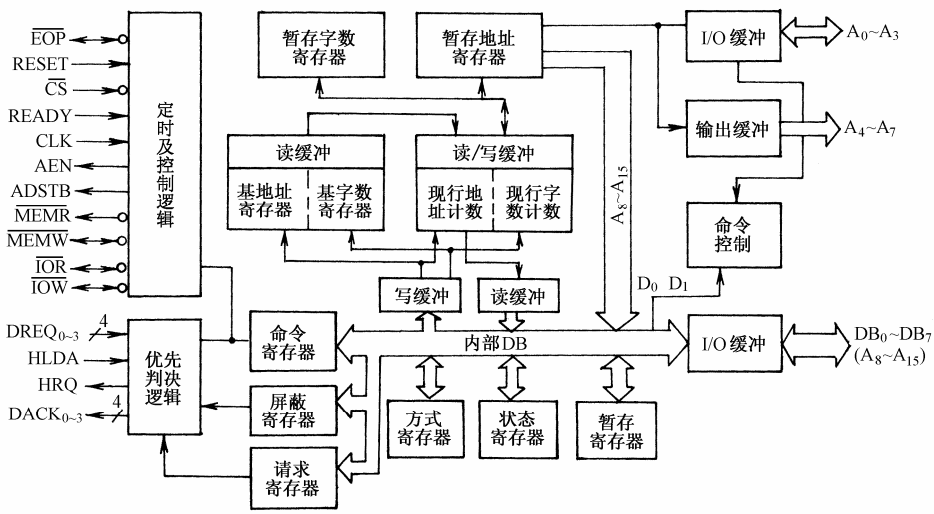


图 5-39 8237A 的内部结构

（1）时序与控制逻辑。8237A 处于从状态时，该部分电路接收系统送来的时钟、复位、片选和读/写控制等信号，完成相应的控制操作；处于主态时则向系统发出相应的控制信号。

（2）优先级编码电路。该部分电路根据 CPU 对 8237A 初始化时送来的命令，对同时提出 DMA 请求的多个通道进行排队判优，以决定哪一个通道的优先级最高。对优先级的管理有两种方式：固定优先级和循环优先级。无论采用哪种优先级管理，一旦某个优先级高的设备在服务时，其他通道的请求均被禁止，直到该通道服务结束为止。

（3）数据和地址缓冲器组。8237A 的 A_4～A_7、A_0～A_3 为地址线；DB_0～DB_7 在从态时传输数据信息，主态时传送地址信息。这些数据引线、地址引线都与三态缓冲器相连，因而可以接管或释放总线。

（4）命令控制逻辑。该部分电路处于从态时，接收 CPU 送的寄存器选择信号（A_0～A_3），选择 8237A 内部相应的寄存器；处于主态时，对方式字的最低两位（D_1D_0）进行译码，以确定 DMA 的操作类型。A_0～A_3 与 $\overline{IOR}$、$\overline{IOW}$ 配合可组成各种操作命令。

（5）内部寄存器组。8237A 内部的其余部分主要为寄存器。每个通道都有一个 16 位的

基地址寄存器、基字计数、当前地址寄存器和当前字计数器，都有一个 6 位的工作方式寄存器。8237A 有 4 个 DMA 通道，因此上述这几种寄存器在片内各有 4 个 DMA 通道。片内还各有一个命令寄存器、屏蔽寄存器、请求寄存器、状态寄存器和暂存寄存器。上述这些寄存器均是可编程寄存器。另外还有暂存字数寄存器和暂存地址寄存器等不可编程的寄存器。

2．8237A 的引脚功能

8237A 是一种 40 引脚的双列直插式器件，其引脚信号如图 5-40 所示。

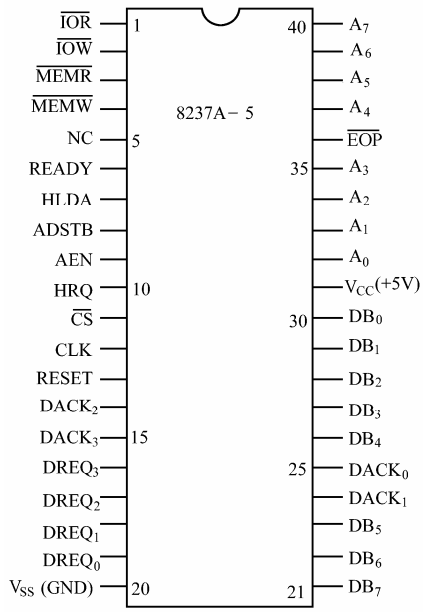


图 5-40　8237A 的引脚

CLK：输入，时钟信号。用来控制 8237A 内部操作的时序及数据传输的速率，对于 8237A-5 可用 5MHz。

$\overline{CS}$：输入，片选信号，低电平有效。当 CPU 控制总线时，用这个信号来选中 8237A 进行 I/O 设备的读/写操作。

RESET：输入，复位信号，高电平有效。复位后，除屏蔽寄存器被置 1（4 个通道全被屏蔽）外，其余所有的寄存器都被清零。

REDAY：输入，准备好信号，高电平有效。表示进入 DMA 的外设已为读写准备好，否则在总线周期中要插入等待状态 S_w。

AEN：输出，DMA 地址允许信号，高电平有效。当 AEN 为高电平时，允许 DMA 控制器送出地址信号而禁止 CPU 地址线接通系统总线；只有当 AEN 为低电平时，才允许 CPU 控制系统总线上的地址信号。

ADSTB：输出，地址选通，高电平有效。8237A 的数据线 DB_7～DB_0 供 DMA 地址信号 A_{15}～A_8 分时使用，当 ADSTB 信号有效时，DB_7～DB_0 上出现的是 DMA 地址的高字节，被此信号选通进入外部锁存器（例如 LS373）。

$\overline{MEMR}$：输出，DMA 存储器读信号，低电平有效。读出的数据可以直接传送给外部设备。

$\overline{MEMW}$：输出，DMA 存储器写信号，低电平有效。写入的数据可以直接来自外部设备。

$\overline{IOR}$：双向，I/O 读信号，低电平有效。当 8237A 作为从属器件时，$\overline{IOR}$ 信号作为输入，配合片选信号 $\overline{CS}$，由 CPU 读 8237A 内部寄存器。当 8237A 作为主控器件时，输出 $\overline{IOR}$ 信号，以读取外设的数据而写入存储器。

$\overline{IOW}$：双向，I/O 写信号，低电平有效。和 $\overline{IOR}$ 一样其信号传输方向由 8237A 在总线上的地位确定。

$\overline{EOP}$：双向，DMA 过程结束信号，低电平有效。若 8237A 中任一通道进入 DMA 过程，当其字节计数结束时，即输出 $\overline{EOP}$ 有效。若 DMA 计数未完，但外部输入一个有效的 $\overline{EOP}$ 信号，则强制结束 DMA 过程。只要 $\overline{EOP}$ 信号有效就会复位内部寄存器。

$DREQ_3$～$DREQ_0$：输入，外设对 8237A 的 4 个通道分别提出 DMA 请求信号，其有效极性可以由编程设定。在固定优先权情况下，$DREQ_0$ 优先级最高，然后依次下降，$DREQ_3$ 最低。当多个通道同时申请时，8237A 只能选择优先级最高的一个通道进行响应。DREQ 信号必须保持到响应信号 DACK 有效以后才能撤销。复位后是高电平有效。

$DACK_3$～$DACK_0$：输出，8237A 给外部的响应信号。其有效极性也可以编程设定，复位后规定低电平有效。

HRQ：输出，8237A 对 CPU 的总线请求信号，高电平有效。8237A 接受了任何一个通道有效的 DREQ 请求之后，就会产生 HRQ 信号。

HLDA：输入，CPU 回答 8237A 的总线响应信号，高电平有效。

DB_7～DB_0：双向数据线。CPU 用它对 8237A 内部寄存器进行读写。在 DMA 传送开始时，存储器地址的 A_{15}～A_8 位经过 DB_7～DB_0 线送出锁存。在同时利用通道 0 和通道 1 进行存储器到存储器的传送时，从原存储单元读出的数据还要经过数据线进入 8237A 内部暂存，然后再经数据线写入目的存储单元。

A_3～A_0：双向地址线。CPU 输出的 A_3～A_0 用来选择访问 8237A 内部的寄存器。8237A 输出的 A_3～A_0 是被读写的存储单元的最低 4 位。

A_7～A_4：三态输出，在 DMA 时用来输出被读写存储单元地址的 A_7～A_4 位。

5.6.2 8237A 的工作时序

8237A 的工作时序如图 5-41 所示。

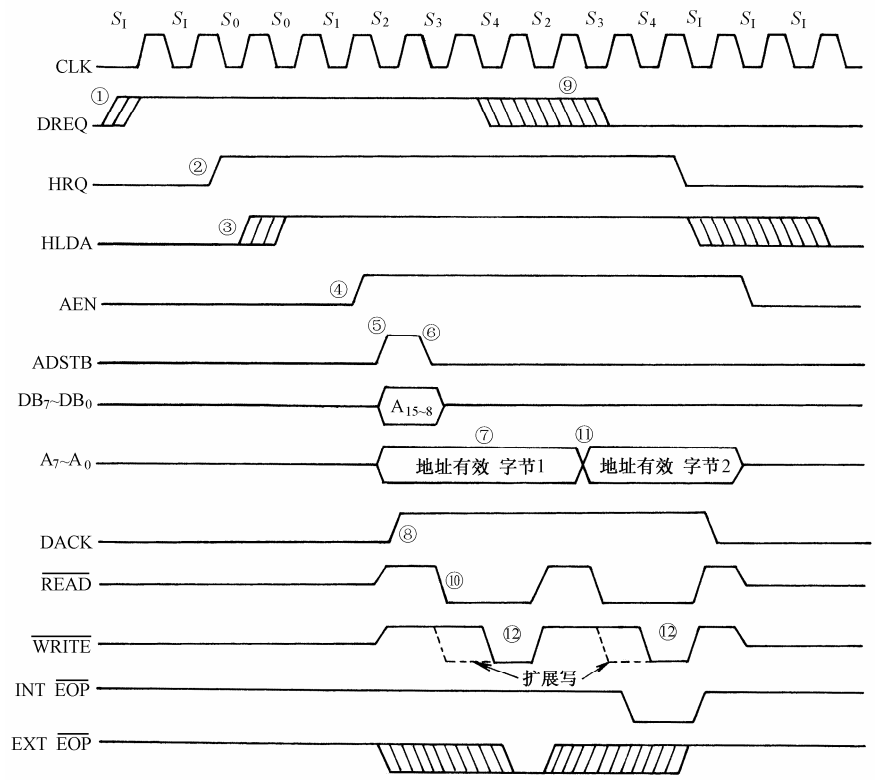


图 5-41　8237A 的工作时序

为了区别于 CPU 的时钟周期，DMA 的每一个时钟周期称为一个 *S* 状态。

1．空闲周期 S_I（IDLE CYCLE）：

S_I 状态是空闲状态。在进入 DMA 传输之前（8237A 的任一通道无请求时），8237A 一直处在连续的 S_I 状态。这时 8237A 作为从属器件，可以接受 CPU 的编程写入或读出。在 S_I 状态中 8237A 要不断地采样各 DREQ 信号。若有 DREQ 信号（一个或多个）产生，且经过屏蔽逻辑及优先级排队后仍有效，在 S_I 的上升沿产生 HRQ 信号，向 CPU 发出总线请求，同时结束 S_I 状态，进入 S_0 状态。

2．等待 CPU 响应周期 S_0

S_0 状态中 8237A 等待 CPU 的总线响应信号 HLDA，在 HLDA 信号有效之前，8237A 一直重复 S_0 状态。S_0 状态中的 8237A 还是从属器件，可以接收 CPU 的读/写操作。若在 S_0 的上升沿检测到 HLDA 信号有效，则进入 S_1 状态。

3．有效周期 S_1～S_4

DMA 传送的工作周期，S_1 产生 AEN 信号并锁存存储器地址 A_{15}～A_8；S_2 产生 DACK 信号并送出 16 位存储器地址 A_{15}～A_0；S_3 读数据；S_4 写数据。若外设速度太慢，可在 S_3 和 S_4 之间插入等待周期 S_W。

5.6.3 8237A 的编程

8237A 每个通道都有自己的模式寄存器，可选择不同的工作模式和操作类型。

1．8237A 的工作模式

如前所述，8237A 在 DMA 传送时有 4 种工作模式，即单字节传送模式、块传输模式、请求传输模式和级联模式。

2．操作类型

无论是单字节传送模式、块传输模式还是请求传输模式，其过程中数据的流向均可分为 3 种操作类型。

DMA 读：把数据由存储器传送到外设，由 $\overline{\text{MEMR}}$ 有效从存储器读出数据，由 $\overline{\text{IOW}}$ 有效把这一数据写入外设。

DMA 写：把外设输入的数据写入选中的存储器。由 $\overline{\text{IOR}}$ 有效从外设输入数据，由 $\overline{\text{MEMW}}$ 有效把数据写入存储器

DMA 校验：校验操作是一种空操作，8237A 并不进行任何读/写操作，但外部仍产生时序和地址信号，所有读/写控制信号无效，实际上没有数据的传递，仅仅用于校验电路工作是否正常。

3．内部寄存器寻址（8237A 内部寄存器的地址）

对 8237A 的内部寄存器进行读/写操作时，$\overline{\text{CS}}$ 必须为低电平才能选中 8237A 芯片，该信号由高位地址经译码后产生。8237A 的 A_3～A_0 线用于选择 8237A 内部的不同寄存器，这些寄存器占 16 个 I/O 端口地址。常把 8237A 的 A_3～A_0 与系统地址总线 A_3～A_0 相连，以产生

寄存器选择信号，而系统高位地址线经译码后形成 I/O 端口选择信号。选择 8237A 内部的一个端口后，再用 $\overline{IOR}$ 信号或 $\overline{IOW}$ 信号决定对某个内部寄存器是读出还是写入。CPU 对这些寄存器的寻址情况列于表 5-3 中。

表 5-3　8237A 寄存器的寻址

$A_3A_2A_1A_0$	通 道 号	读 操 作（$\overline{IOR}$）	写 操 作（$\overline{IOW}$）
0000	0	读当前地址寄存器	写基（当前）地址寄存器
0001		读当前字节数计数器	写基（当前）字节数计数器
0010	1	读当前地址寄存器	写基（当前）地址寄存器
0011		读当前字节数计数器	写基（当前）字节数计数器
0100	2	读当前地址寄存器	写基（当前）地址寄存器
0101		读当前字节数计数器	写基（当前）字节数计数器
0110	3	读当前地址寄存器	写基（当前）地址寄存器
0111		读当前字节数计数器	写基（当前）字节数计数器
1000	公共	读状态寄存器	写命令寄存器
1001		—	写请求寄存器
1010		—	写屏蔽寄存器某一位
1011		—	写模式寄存器
1100		—	清除高/低触发器
1101		读暂存寄存器	主清除（软件复位）
1110		—	清除屏蔽寄存器
1111		—	写屏蔽寄存器所有位

从表中可以看出，前 8 个地址（00H～07H，A_3=0）各通道单独占有。读操作只对当前地址寄存器或当前字节数计数器进行，写操作则都可以（不论基地址寄存器或当前地址寄存器，不论基字节数寄存器或当前字节数计数器）。

8237A 数据线为 8 位，而寄存器或计数器均为 16 位，故读/写操作分两次进行，由 8237A 内部高/低触发器确定高字节（“1”）或低字节（“0”）。

4．寄存器功能及编程

8237A 各寄存器对其工作起着不同的控制作用，在进行 DMA 传输前，必须对各寄存器写入一定的内容，以得到所要求的功能，即进行初始化编程。初始化的内容可分为数值型和功能型两类。

每个通道要把传输中将要访问的存储器的初始地址写入基地址寄存器和当前地址寄存器中；要传送的字节数写入基字节数计数器和当前字节数计数器，这就是初始化编程中的数值内容。功能编程的内容分别写入各功能寄存器中（命令 R、模式 R、请求 R、屏蔽 R 和状态 R）。

（1）命令寄存器。8237A 命令寄存器的功能如图 5-42 所示。

D_7 规定 DACK 信号的有效极性：0 为低电平有效，1 为高电平有效。

D_6 规定 DREQ 信号的有效极性：0 为高电平有效，1 为低电平有效。

D_5 和 D_3 两位用于选择工作时序：D_5=0 为正常时序写，D_5=1 为扩展写（若 D_3=1 则无效）；D_3=0 为正常时序，D_3=1 为压缩时序。

D_4 规定优先权编码方式：0 为固定优先权，1 为旋转优先权。

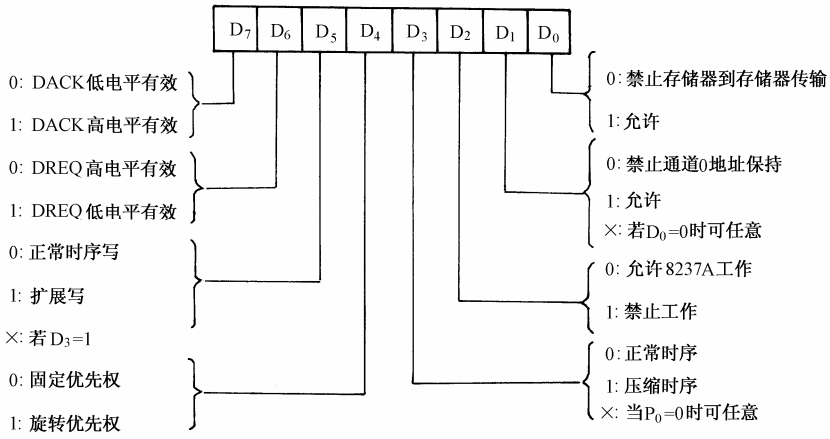


图 5-42　8237A **命令寄存器**

D_2 **对** 8237A **工作起控制作用：**0 **允许** 8237A **芯片工作，**1 **禁止** 8237A **芯片工作。**

D_1 **用于通道** 0 **地址保持控制：**0 **禁止通道** 0 **地址保持，**1 **允许通道** 0 **地址保持（若** D_0=0 **则无效）。**

D_0 **用于** DMA **传送控制：**0 **禁止存储器到存储器传输，**1 **允许传输。**

（2**）模式寄存器。**8237A **模式寄存器的功能如图** 5-43 **所示。**

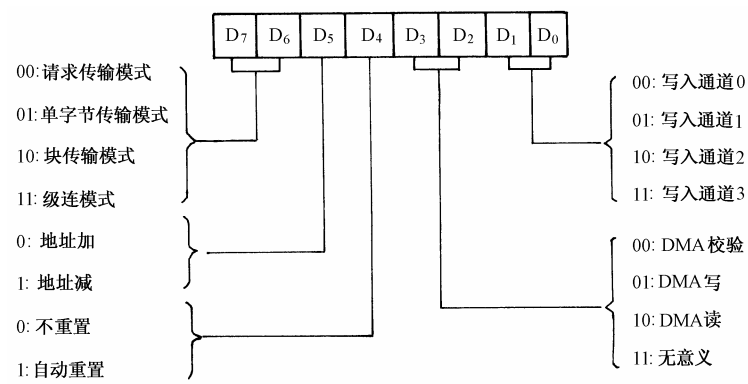


图 5-43　8237A **模式寄存器**

D_7D_6 **用于设置通道的工作模式：**00 **为请求传输模式，**01 **为单字节传输模式，**10 **为块传输模式，**11 **为级连模式。**

D_5 **用于指明** DMA **传送地址增减方向：**0 **为地址加** 1，1 **为地址减** 1。

D_4 **用于控制基值寄存器初值自动重置：**0 **为不重置，**1 **为自动重置。**

D_3D_2 **规定了操作类型：**00 **为** DMA **校验，**01 **为** DMA **写，**10 **为** DMA **读，**11 **无意义。**

D_1D_0 **用于指定写入的通道号：**00 **为写入通道** 0，01 **为写入通道** 1，10 **为写入通道** 2，11 **为写入通道** 3。

（3**）请求寄存器。**8237A **请求寄存器的功能如图** 5-44 **所示。每个通道都设有一个请求位，可用软件命令对其进行置位和复位操作。对请求位的置位等效于外部产生一个有效的** DREQ **信号。**

D_1D_0 **用于指定写入的通道号：**00 **为写入通道** 0，01 **为写入通道** 1，10 **为写入通道** 2，11 **为写入通道** 3。

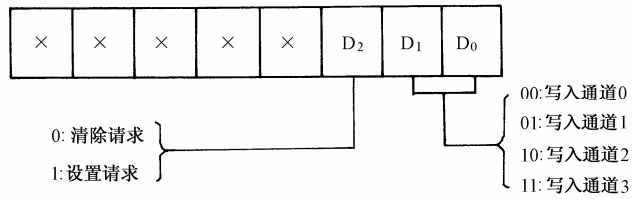


图 5-44　8237A 请求寄存器

D_2 用于设置和清除请求位：0 为清除请求，1 为设置请求。

（4）屏蔽寄存器。8237A 屏蔽寄存器的功能如图 5-45 所示。

当某位设置为 1 时，其外部对应的 DREQ 信号被屏蔽，不予响应。

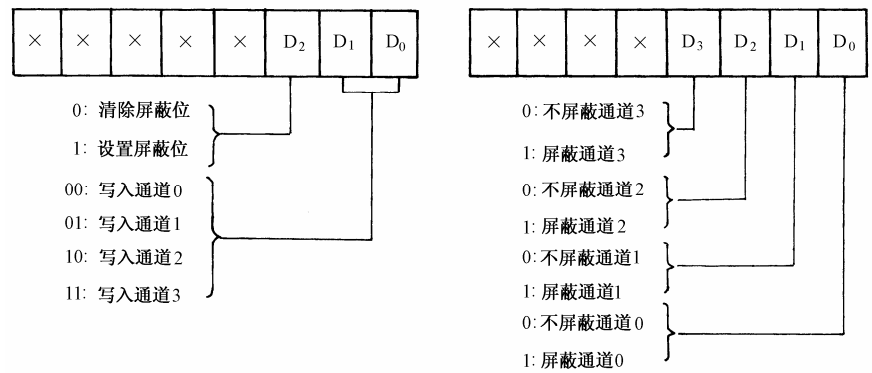


图 5-45　8237A 屏蔽寄存器

D_3～D_0 分别用于对 4 个通道进行屏蔽控制。一般情况下，通道在一次 DMA 过程结束后，就自动设置屏蔽位，若再次传输，必须用软件清除其屏蔽位（自动重置方式除外）。

（5）状态寄存器。8237A 状态寄存器的功能如图 5-46 所示。

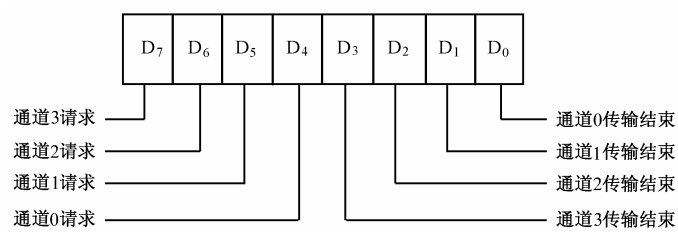


图 5-46　8237A 状态寄存器

状态寄存器的高 4 位记录各通道请求的情况，D_7～D_4 分别对应于通道 3～通道 0；状态寄存器的低 4 位反映各通道传输完成情况，D_3～D_0 分别对应于通道 3～通道 0。

5.6.4　8237A 的应用

8237A 的初始化编程工作步骤如下：

（1）输出主清除命令使 8237A 复位；

（2）写入基地址和当前地址寄存器，确定起始地址；

（3）写基字节数和当前字节数计数器，确定要传送的字节数；

（4）写入模式寄存器，指定工作方式；

（5）写入屏蔽寄存器；

（6）写入命令寄存器；

（7）写入请求寄存器。

例 5.12　若利用 8237A 芯片的通道 0，由外设（磁盘）输入 32KB 数据块，送至内存 8000H 开始的区域（增量传送），假设用块连续传送方式，传送完不自动初始化，外设的 DREQ 和 DACK 均为高电平有效。试编写其程序。

要编程首先要确定端口地址，地址的低 4 位用于区分 8237A 的内部寄存器；高 4 位地址经译码后，连至片选端 $\overline{CS}$。假定选中的高 4 位为 5。

模式控制字：84H（地址增量、写传送、数据块传送、非自动初始化、传送通道 0）；

屏蔽控制字：00H（清除、写通道 0）；

命令控制字：A0H（DACK 高有效、扩展写、正常时序、禁止 0 通道地址保持、DREQ 高有效、固定优先权、允许 8237A 工作、非存储器到存储器传送）。

程序清单如下：

```
MOV   AL，   00H
OUT       5DH，  AL        ；输出主清除命令
OUT       50H，  AL        ；写基地址和当前地址的低位地址 00H
MOV   AL，   80H
OUT   50H，  AL        ；写高位地址 80H
MOV   AL，   00H
OUT       51H，  AL        ；写基字节数和当前字节数的低 8 位
MOV   AL，   80H
OUT       51H，  AL        ；写字节数高 8 位
MOV   AL，   84H
OUT       5BH，  AL        ；写模式控制字
MOV   AL，   00H
OUT       5AH，  AL        ；写屏蔽控制字
MOV   AL，   0A0H
OUT   58H，  AL        ；写命令控制字
```

例 5.13　现假设用系统板上 8237A 芯片的通道 1，将内存起地址为 80000H 的 300H 个字节内容直接输出给外设。试编写其程序

程序清单如下：

```
MOV   AL，   4           ；命令字，禁止 8237A 工作（D2=1）
OUT   08，   AL          ；写命令寄存器（08H）
MOV   AL，   0
OUT   0DH，  AL          ；主清除，清除高/低触发器（0DH）
OUT   02，   AL          ；写低位地址 00，高/低触发器自动翻转
OUT   02，   AL          ；写高位地址 00，02H 基地址/当前地址寄存器
MOV   AL，   8           ；页面地址为 8（A19～A16=08H）
```

```
        OUT   83H，   AL          ；写入页面寄存器
        MOV   AX，    300H        ；传输字节数
        DEC   AX                  ；初始化时，写入的值减 1
        OUT   03，    AL          ；写字节数低位
        MOV   AL，    AH
        OUT   03，    AL          ；写字节数高位
        MOV   AL，    49H         ；模式字，单字节读，地址加 1（通道 1）
        OUT   0BH，   AL          ；模式字
        MOV   AL，    44H         ；命令字:DACK 和 DREQ 低有效
        OUT   08H，   AL          ；正常时序，固定优先权
        MOV   AL，    01          ；清除通道 1 屏蔽
        OUT   0AH，   AL          ；0AH 写屏蔽寄存器
WAIT：IN    AL，    08          ；读通道 1 状态
        AND   AL，    02          ；传输是否完成
        JZ    WAITF               ；没完成则等待
        MOV   AL，    05          ；完成后屏蔽通道 1
        OUT   0A，    AL
          …
```

5.7 CRT控制器MC6845

CRT 显示器是 20 世纪 70 年代发展起来的终端设备。根据 CRT 显示器的工作要求，需要周期性地对 CRT 显示器进行刷新工作及产生各种控制信号，一般的 CRT 显示器都需要通过 CRT 接口中的 CRT 控制器（CRTC）来完成显示器屏幕的刷新工作、各类同步信号以及提供字符或图形的显示信号。

随着超大规模集成电路的发展，使得微处理器占据了高性能和低价格两方面的优势，目前的显示终端几乎都用了微处理器，也就是说，CRT 显示终端本身就是一个含有 CPU 的控制系统。但是，专用 CRT 控制器的使用，仍大大减轻了 CRT 控制系统中 CPU 的负担。因此，CRT 控制器仍是当今用于 CRT 显示终端的主要控制电路。

作为 CRT 显示器，根据其光栅扫描原理，需要 CRT 接口电路提供水平同步信号、垂直同步信号、消隐信号以及显示内容和显示属性信息，经混合产生视频信号由 CRT 显示。CRT 控制器就是 CRT 接口电路中的核心控制芯片，用以产生各类定时信号、显示缓冲器和字符发生器的控制信号并产生相应的视频信号。MC6845 就是一个可编程的 CRT 控制器。

5.7.1 MC6845 的引脚功能

MC6845 的功能是一方面要按照一定的频率产生垂直同步信号 VS、水平同步信号 HS 和显示允许信号 DE，将这些信号送到视频信号处理电路；另一方面要按字符时钟周期读取显示存储器中的字符码，并且通过字符发生器转换成点阵码送至视频信号混合器。因此，MC6845 会产生访问显示存储器的地址 MA_{13}～MA_0 以及字符发生器的地址 RA_4～RA_0。

图 5-47 是 MC6845 芯片的引脚信号图，这些信号可以分为 4 类。

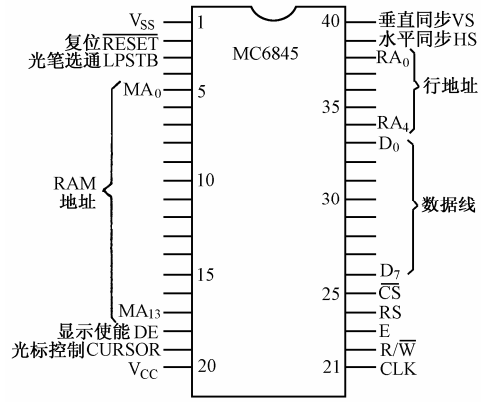


图 5-47　MC6845 引脚

1．与处理器的接口电路

D_7～D_0：8 位数据线，双向，可以在处理器和 CRTC 的内部寄存器之间进行数据传送。

E：读/写允许信号，当 E 信号有效时，数据线输入/输出缓冲器使能，CPU 可以将数据写入 CRTC 或从 CRTC 中读出。

$\overline{CS}$：片选信号，低电平有效，当 $\overline{CS}$ 为低电平时，MC6845 被选中，CPU 可以对 MC6845 的内部寄存器进行读/写操作。

RS：寄存器选择信号，当 RS 为 0 时，处理器可以对 MC6845 的内部地址寄存器进行读/写操作；当 RS 为 1 时，则对 MC6845 的内部数据寄存器进行读/写操作。

R/$\overline{W}$：读写控制信号，当 R/$\overline{W}$ 为高电平时，处理器可以对 MC6845 的内部寄存器进行读操作；当 R/$\overline{W}$ 为低电平时，处理器对 MC6845 的内部寄存器进行写操作。

2．CRT 控制信号

VS：垂直同步信号，高电平有效，用于确定显示文本的垂直位置，这个信号可以直接驱动监视器，也可以送到视频信号处理电路产生复合的视频信号。

HS：水平同步信号，高电平有效，用于确定显示文本的水平位置，这个信号可以直接驱动监视器，也可以送到视频信号处理电路产生复合的视频信号。

DE：显示允许信号，高电平有效，当 DE 有效时，表示 MC6845 当前正在提供显示存储器的一个单元地址。

3．与存储器的地址连接信号

MA_{13}～MA_0：显示存储器地址，输出，MC6845 通过这 14 位地址线对显示存储器进行寻址，因而最多可寻址 16KB 的显示存储器空间。

RA_4～RA_0：行地址，输出，MC6845 通过内部行地址计数器提供的 5 位地址实现对字符发生器的寻址。

4．其他引脚信号

CURSOR：光标控制信号，高电平有效，为外部视频处理逻辑电路指出一个正确的光标

位置，使得扫描线到达光标位置时，在屏幕上做光标显示。

CLK：时钟信号，时钟输入，一般是由外部点时钟计数器送来的进位信号，用于对 MC6845 中的所有功能电路进行同步。

LPSTB：光笔选通信号，以由低到高的上升沿为有效信号，此信号将当前显示存储器的地址锁存到光笔寄存器中。

V_{CC} 和 V_{SS}：电源，接＋5V 直流电源。

$\overline{\text{RESET}}$：复位信号，低电平有效，用于使 MC6845 进入复位状态。复位以后，MC6845 处于所有计数器被清除，MC6845 停止显示操作；所有输出信号处于低电平；内部各控制寄存器保持复位前的状态。

5.7.2 MC6845 的内部寄存器及作用

为了完成预定的功能，MC6845 内部主要有可编程的水平定时器和垂直定时器、线性地址发生器、光笔控制电路以及与微处理器总线有关的接口电路。

1．水平定时器

在水平定时器部分，水平计数器和各个可编程的内部寄存器通过比较器不断比较，从而产生下列结果：

（1）产生由相应寄存器中的参数所确定的具有一定频率和宽度的水平同步脉冲 HS，同时还确定水平同步脉冲在扫描周期中的位置；

（2）产生由相应寄存器中的参数所确定的一定频率和宽度的显示允许信号 DE。

2．垂直定时器

垂直定时器部分的字符行计数器和各垂直寄存器比较，从而产生如下结果：根据寄存器中的参数产生垂直同步脉冲 VS，并确定其频率以及在垂直扫描周期中的位置。

3．线性地址发生器

线性地址发生器是由输入时钟 CLK 驱动的，它根据字符要在屏幕上显示的位置来确定显示存储器中的字符地址。MA_{13}～MA_0 共 14 根地址线可以用来对多达 4 个 4KB 的页面或者 8 个 2KB 的页面进行寻址。利用起始地址寄存器，可以对 16KB 长的字符文本实现硬件滚动。

4．光标光笔控制电路

光标逻辑电路决定了光标在屏幕上的位置、大小和闪烁频率。

光笔选通信号变为高电平时，使得当前线性地址发生器的值锁存到光笔寄存器中，随后，处理器会读取光笔寄存器中的值。

微处理器是通过 MC6845 内部寄存器的访问来实现对 MC6845 的编程及工作方式的设置的。MC6845 的内部寄存器可以由微处理器通过数据线 D_7～D_0 及控制信号进行编程，这些控制信号是 R/$\overline{\text{W}}$、$\overline{\text{CS}}$、RS 和 E。

内部寄存器共有 19 个，如表 5-4 所示，为了减少占用的端口，其中有 1 个寄存器专门用做地址索引寄存器。在访问另外 18 个内部寄存器之前，只要先往地址索引寄存器中设置寄存器号，然后对数据端口进行访问就行了。这样，只要使用两个 I/O 端口（1 个索引端口，1 个

数据端口）便可以对 18 个内部参数寄存器进行编程了。

表 5-4　MC6845 内部寄存器一览表

$\overline{CS}$	RS	地址索引寄存器值	寄　存　器	寄存器名称	编 程 单 位	可　读　性	可　写　性
1	×	—	—	—	—	—	—
0	0	—	AR	地址索引寄存器	—	否	是
0	1	0	R_0	水平总时间寄存器	字符	否	是
0	1	1	R_1	水平显示时间寄存器	字符	否	是
0	1	2	R_2	水平同步位置寄存器	字符	否	是
0	1	3	R_3	水平同步宽度寄存器	字符	否	是
0	1	4	R_4	垂直总时间寄存器	字符行	否	是
0	1	5	R_5	垂直校正寄存器	扫描行	否	是
0	1	6	R_6	垂直可见部分寄存器	字符行	否	是
0	1	7	R_7	垂直同步位置寄存器	字符行	否	是
0	1	8	R_8	扫描方式寄存器	—	否	是
0	1	9	R_9	行扫描线数寄存器	扫描行	否	是
0	1	10	R_{10}	光标起始寄存器	扫描行	否	是
0	1	11	R_{11}	光标结束寄存器	扫描行	否	是
0	1	12	R_{12}	起始地址寄存器（高位）	—	否	是
0	1	13	R_{13}	起始地址寄存器（低位）	—	否	是
0	1	14	R_{14}	光标寄存器（高位）	—	是	是
0	1	15	R_{15}	光标寄存器（低位）	—	是	是
0	1	16	R_{16}	光笔寄存器（高位）	—	是	否
0	1	17	R_{17}	光笔寄存器（低位）	—	是	否

（1）地址索引寄存器：地址索引寄存器是一个 5 位的只读寄存器，它用来容纳其他 18 个寄存器的地址，从而实现对 MC6845 内部参数寄存器的间接寻址。当 RS 和 $\overline{CS}$ 都为低电平时，就选中了地址索引寄存器中指出的某个内部参数寄存器。

（2）水平总时间寄存器 R_0：这是一个 8 位的只写寄存器，它可以通过编程被写入扫过一条扫描线所用的时间，时间本身是用字符数来表示的，R_0 中写入的值为一行中的显示字符数加上回扫时间折合的字符数减 1。水平扫描总时间决定了水平同步信号 HS 的频率。

（3）水平显示部分时间寄存器 R_1：这是一个 8 位的只写寄存器，它的值决定了每行所显示的字符数，水平显示部分时间也是以字符为单位来表示的。当然，R_1 中的设置值应小于 R_0 中的设置值。

（4）水平同步位置寄存器 R_2：这是一个 8 位的只写寄存器，它控制水平扫描周期中同步脉冲 HS 的位置，而水平同步脉冲的位置又决定了画面在屏幕上的位置。R_2 中可以设置任何 8 位二进制数值，不过 R_2 和 R_3 的和必须小于 R_0 的值，而 R_2 的值又必须大于 R_1 的值。

（5）水平同步宽度寄存器 R_3：这是一个 8 位的只写寄存器，它的值决定了水平同步脉冲的宽度。水平同步脉冲 HS 的宽度可以通过编程设置为 1～15 个字符时钟周期，这样就可以和多数不同规格的监视器所要求的水平同步脉冲宽度相兼容了。

（6）垂直总时间寄存器 R_4：这是一个 7 位的只写寄存器，用于确定每一帧的字符总行数减 1 的值。

（7）垂直校正寄存器 R_5：这是一个 5 位的只写寄存器，当每一帧的扫描线不一定正好为每个字符行扫描行数的整数倍时，该寄存器就用于记录多余的扫描行数。因此，通过 R_4 和 R_5 可以计算总的扫描行数。

（8）垂直可见部分寄存器 R_6：这是一个 7 位的只写寄存器，用来指出 CRT 屏幕上的可见字符行数，R_6 的编程单位为字符行。

（9）垂直同步位置寄存器 R_7：这是一个 7 位的只写寄存器，用于控制画面在屏幕上的垂直位置。R_7 的编程单位也是字符行，当 R_7 中的值增加时，画面会上移，反之，画面下移。另外，垂直同步脉冲的宽度是固定的，为 16 条扫描线时间。

（10）扫描方式寄存器 R_8：MC6845 用内部寄存器 R_8 来控制对显示器是否用隔行扫描方式。R_8 是一个两位的寄存器，当 R_8 的第 0 位为“0”时，是普通同步方式，即逐行扫描方式；如果第 0 位为“1”，则为隔行扫描方式，其中，若 R_8 的第 1 位为“0”，则设置为第一种隔行扫描方式；若第 1 位为“1”，则设置为第二种隔行扫描方式。其方式设置关系如表 5-5 所示。

表 5-5　扫描方式设置

第 1 位	第 0 位	方　式
×	0	普通同步方式，即逐行扫描方式
0	1	第一种隔行扫描方式
1	1	第二种隔行扫描方式

采用隔行扫描方式时，信息帧总是被分为交替的偶数场和奇数场，也就是说两场之间的垂直同步延迟了半根扫描线的时间。在第一种隔行扫描方式中，两场的信息完全相同，这种方式对于字符显示很有用。在第二种扫描方式中，偶数场和奇数场分别包含了画面的偶数行信息和奇数行信息，因此两者包含的信息不同，这种方式常用于图形显示方式。

（11）字符行扫描线数寄存器 R_9：这是一个 5 位的只写寄存器，它决定了每个字符行包括行间间隔中的扫描线数，因此，它也控制着线性地址发生器的工作。R_9 的编程值比每个字符行的实际扫描线数少 1。

（12）光标起始寄存器 R_{10}：这是一个 7 位的只写寄存器，用来确定光标在字符行中的起始扫描线的位置，并决定光标的闪烁频率。R_{10} 和 R_{11} 在一起控制了光标的大小，R_{10} 的第 6、第 5 这两位决定了光标的闪烁频率，通过对这两位值的不同设置，可以确定不同的闪烁模式。具体如表 5-6 如示。

表 5-6　光标显示模式设置

R_{10} 的第 6 位	R_{10} 的第 5 位	光标显示模式
0	0	显示而不闪烁
0	1	不显示
1	0	按场频的 1／16 频率闪烁
1	1	按场频的 1／32 频率闪烁

（13）光标结束寄存器 R_{11}：这是一个 5 位的只写寄存器，用于决定光标在字符行中的最后一根扫描线的位置。R_{11} 与 R_{10} 这两个寄存器用来控制光标的大小，可以使光标在高度上跨越 32 根扫描线。

（14）起始地址寄存器 R_{12}、R_{13}：这是一对只写寄存器，共 14 位，用于控制垂直扫描开

始所对应的显示存储器地址和首址之间的差值。R_{12} 中存放位移量的高 6 位（MA_{13}～MA_8），R_{13} 中存放位移量的低 8 位（MA_7～MA_0）。可见，起始地址寄存器决定了在屏幕上显示存储器中哪个单元开始的内容。

（15）光标寄存器 R_{14}、R_{15}：这是一对可读可写的寄存器，通过对它们的编程可以将光标定位在显示存储器的任何位置。R_{14} 和 R_{15} 总共长 14 位，其中 R_{14} 中存放高 6 位地址（MA_{13}～MA_8），R_{15} 中存放低 8 位地址（MA_7～MA_0）。

（16）光笔寄存器 R_{16}、R_{17}：这是在系统连接光笔情况下才使用的一对寄存器，它们总长 14 位。这对寄存器在光笔选通脉冲 LPSTB 的上升沿处将当前显示缓冲区的地址写入，R_{16} 装入高 6 位地址（MA_{13}～MA_8），R_{17} 装入低 8 位地址（MA_7～MA_0）。

5.7.3 MC6845 的定时关系

MC6845 有着严格的定时要求，以产生各类定时信号。

1．光栅扫描定时

图 5-48 所示为典型的 CRT 各定时参数的含义，这些定时参数都是通过对内部寄存器 R_0～R_9 进行编程而设置的。

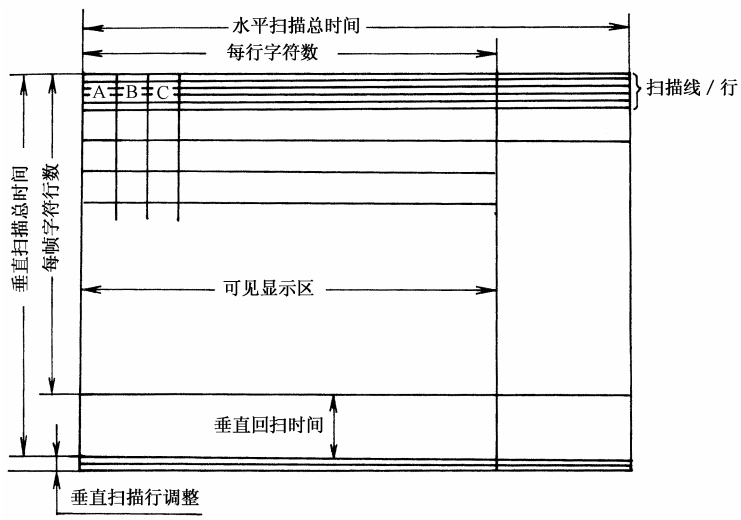


图 5-48　显示器定时参数的含义

R_0 表示水平扫描总时间，R_1 表示水平显示时间，两者的差就是非显示时间，正是这段时间，电子束又回到了屏幕的左边，所以也称为回扫时间。回扫时间本身是与监视器的水平扫描参数有关的。实际上，真正用电子束回扫的时间要小于非显示时间，因为电子束正向扫过屏幕左右边沿的时间也计算在非显示时间中。按照实际经验，一般认为，将水平非显示时间取水平扫描总时间的 20%比较合适。

同样，R_4 和 R_5 描述了垂直扫描的总时间，而 R_6 则是垂直可见部分的时间，两者之差是垂直回扫的时间。根据经验，垂直回扫时间为垂直扫描总时间的 1/3 比较合适。

在 CRT 的垂直回扫期间，需要一个垂直同步信号。垂直同步信号的频率常称为帧频，反映了每秒在屏幕上显示多少帧图像，通常帧频取 50Hz 或 60Hz。

在 CRT 的水平回扫期间，需要一个水平同步信号。水平同步信号的频率称为行频。行频反映了水平扫描一行所需要的时间量。当垂直扫描的频率确定，同时垂直扫描的总行数也确定时，两者相乘就可计算出行频。

字符频率反映了在一次水平扫描过程中每秒能完成多少个字符扫描的量。因此，用行频与一个扫描行中总的字符个数（包括正扫描与回扫描）相乘，就可得出字符频率。

由于每个字符都是由点阵组成其图像的，其点像是显示器的最基本显示单位。一个字符的水平点阵数反映了字符的宽度，垂直点阵数反映了字符的高度。用垂直点阵数可以计算垂直扫描行数和垂直扫描字符行数的关系。用水平点阵数乘以字符频率就是点像频率。点像频率也可称为点时钟频率，点时钟频率信号可以用来控制视频移位寄存器，使之将供显示的并行信号转化为串行信号移位输出到 CRT 显示器。显然，点时钟频率是 CRT 控制系统的主时钟，所有其他的定时信号都可以由点时钟信号分频得到。图 5-49 所示为典型 CRT 控制电路中的定时关系，以点时钟为基础，可以产生一系列的定时控制信号。

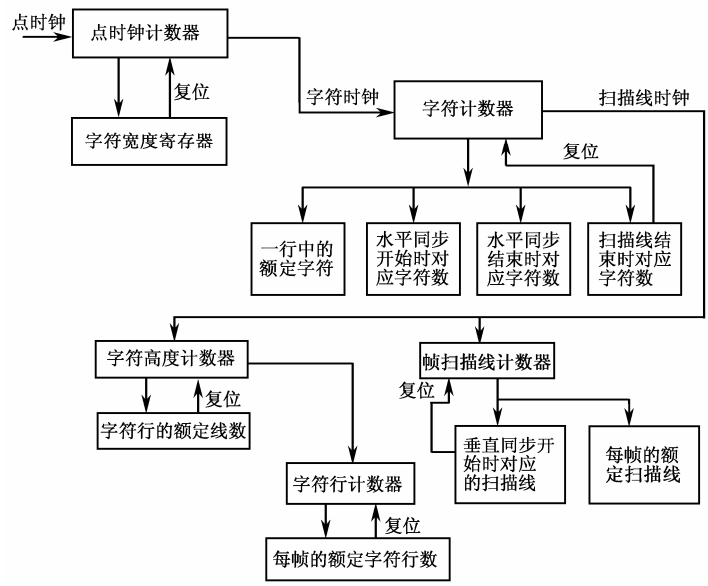


图 5-49　CRT 时钟定时关系

点时钟信号对点时钟计数器进行计数。点时钟计数器一边计数，一边与字符宽度寄存器进行比较，字符宽度是指一个字符在水平方向的点数，当比较结果相等时，点时钟计数器清 0 并使字符计数器加 1。

字符计数器对一条扫描线上已经扫描的字符进行计数，同时此计数器同 4 个不同的寄存器进行比较。R_1 反映一行中显示的字符数，R_2 和 R_3 反映同步脉冲的起始位置和结束位置，R_0 反映扫描行结束位置。比较相等后分别产生消隐信号和产生水平同步信号。当一个扫描行结束后，字符计数器清 0，并产生一个进位脉冲作为扫描线时钟，使字符高度计数器和帧扫描线计数器加 1。

字符高度计数器记录了当前字符行中已经扫描过的线数，当字符高度计数器的扫描线数和寄存器 R_9 中记录的字符行的额定线数相等时，字符高度计数器清 0，并产生进位脉冲，使字符行计数器加 1。

字符行计数器记录了当前一帧中已经显示的字符行数，当字符行计数器与每帧的额定字

符行数寄存器即 R_4 的值相等时，说明一帧的显示结束，此时，字符行计数器清 0。

帧扫描线计数器记录了一帧中已经扫过的扫描线数，它与垂直同步开始寄存器比较，相同时产生垂直同步信号。帧扫描线计数器同时与每帧的额定扫描线数值进行比较，结果相等则开始新一帧的扫描。

2．字符发生器

在显示存储器中存放的是 ASCⅡ码，为了能将字符按点阵形式显示在屏幕上，就需要进行转换，这要用到字符发生器。实际上，字符发生器是一个 ROM 存储器，其中存放了每个 ASCII 码的显示字型码。典型的字型是由 5×7 或 7×9 阵列组成的，图 5-50 表示出字母 H 的点阵形式，每个实点用逻辑“1”表示，每个虚点用逻辑“0”表示。

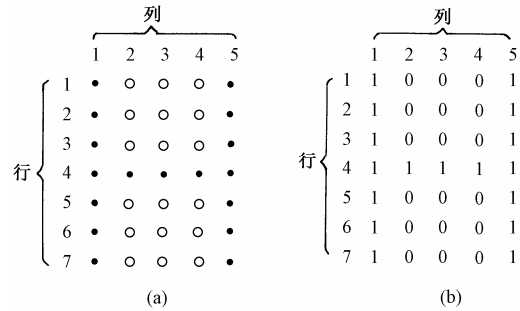


图 5-50　字符的点阵表示和逻辑电平表示

图 5-51 表示了字符发生器的工作原理。字符发生器用两组地址输入来选中其中的一个单元。一组用来给出某个字符的 ASCⅡ码位，如对于通常可显示的 128 个字符，字符发生器中存放了 128 个 ASCⅡ码字符对应的点阵，这样就需要有 128 个 ASCⅡ码位。另一组是行选择码位，行选择码位实际上就是字符行中扫描线的序号，决定了此刻输出字符点阵的哪一行。对于一个 5×7 点阵，即 5 个点宽，7 条扫描线高，所以每次访问字符发生器后，输出为 5 位，而构成一个字符需要对字符发生器访问 7 次。也就是说，显示一个字符的每一行点阵都要访问一次字符发生器，每一行点阵在字符发生器中存放的单元地址就是由字符的 ASCⅡ码和字符行的扫描线序号决定的。最终，字符发生器中输出的点阵在点时钟信号的控制下，通过视频移位寄存器由并行码转换为串行码送显示器，用于打开或关闭 CRT 的电子束。

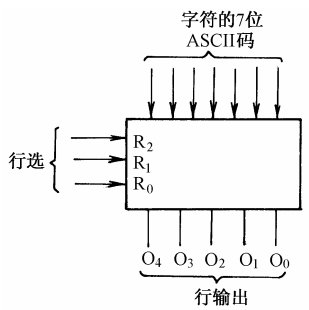


图 5-51　字符发生器的工作原理

因此，点时钟信号控制视频点阵信息的输出，字符频率信号可用于控制字符发生器的每一行点阵码的输出。由于一个字符行的所有扫描线全部完成扫描显示，一个完整的字符才显示完毕，字符行高度寄存器以字符行扫描线数寄存器为依据控制着显示存储器的地址变化，即 MC6845 的显示存储器地址输出值 MA_{13}～MA_0 的变化为：在字符频率信号的控制下，地址输出 MA_{13}～MA_0 加 1 指向下一个需要显示的 ASCⅡ码地址；一次行扫描结束后，地址 MA_{13}～MA_0 输出值为原行首地址值，而字符行的扫描线序号，即 RA_4～RA_0 的的值加 1；只有一个字符行扫描全部完成后，MA_{13}～MA_0 才输入下一个字符行的首地址，RA_4～RA_0 的输出为 0。

5.7.4 MC6845 应用举例

在 IBM PC/XT 系统中，CRT 显示器接口电路就是采用 MC6845 作为 CRT 控制器的。图 5-52 所示为单色显示器的接口电路。

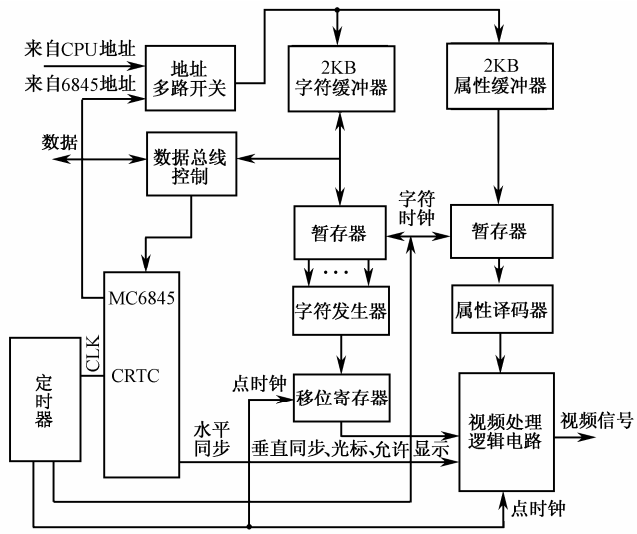


图 5-52 IBM PC/XT 单色显示器接口电路框图

在接口电路中，用 MC6845 作为 CRT 控制器，它的主要功能是产生水平同步信号、垂直同步信号以及产生访问显示存储器所需要的地址码。接口板中用 4KB 的 SRAM 作为显示存储器，还用 8KB 的 ROM 构成了字符发生器，用来存放 3 套不同字体的字符集，一个字符集包括 256 种不同的字符。

单色显示器接口板的主要功能如下：

（1）主机的 CPU 在显示器的水平回扫期间和垂直回扫期间，将要显示的字符送入显示存储器。

（2）MC6845 按照定时关系产生水平同步信号和垂直同步信号，送到视频信号处理逻辑电路中，并且产生地址码去读取显示存储器中的字符。

（3）从显示存储器中读出的字符作为一组地址，以 MC6845 提供的字符行扫描线序号作为另一组地址，去读取字符发生器中的点阵码。

（4）字形点阵码读出后送到移位寄存器，逐位移出送到视频处理逻辑电路，和字符属性、光标信号结合后成为视频输出信号。

IBM PC/XT 的单色显示器特性如下：

（1）屏幕格式为 80 字符/行×25 行；

（2）采用 9×14 的字符框，实际字符占 7×9 点阵；

（3）垂直分辨率为 350 线，水平分辨率为 720 点，屏幕刷新频率为 50Hz。

根据以上特性，表 5-7 列出了 CRT 接口电路中 MC6845 内部寄存器的对应参数，这些参数在初始化程序中进行设置。

表 5-7 单色显示器接口电路中 MC6845 的内部寄存器参数

寄 存 器	寄存器名称	编 程 单 位	参 数

R_0	水平总时间寄存器	字符	61H
R_1	水平显示时间寄存器	字符	50H
R_2	水平同步位置寄存器	字符	52H
R_3	水平同步宽度寄存器	字符	0FH
R_4	垂直总时间寄存器	字符行	1FH
R_5	垂直校正寄存器	扫描行	06H
R_6	垂直可见部分寄存器	字符行	19H
R_7	垂直同步位置寄存器	字符行	19H
R_8	扫描方式寄存器	—	02H
R_9	行扫描线数寄存器	扫描行	0DH
R_{10}	光标起始寄存器	扫描行	0BH
R_{11}	光标结束寄存器	扫描行	0CH
R_{12}、R_{13}	起始地址寄存器	—	00H
R_{14}、R_{15}	光标寄存器	—	00H

在 IBM 单色显示器接口电路所支持的 256 种字符代码中，包含了字符、数字和少量简单的游戏符号。在显示存储器中存放时，每个字符占用 1 个字节，偶地址单元的 1 个字节用来存放字符的 ASCII 码，奇地址单元的 1 个字节用来存放字符属性代码。属性代码的格式定义如图 5-53 所示。

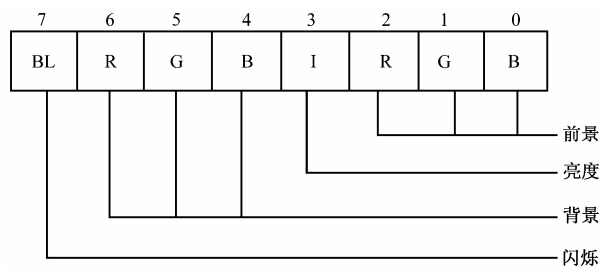


图 5-53　属性代码格式

显示器接口电路按这种定义对字符属性进行译码，根据闪烁位、光亮度、前景位和背景位的值产生字符属性功能。具体属性如表 5-8 所示。

表 5-8　字符属性功能

背景 RGB	前景 RGB	属性功能
000	000	不显示
000	001	加下横线
000	111	黑底白字
111	000	白底黑字

本 章 小 结

可编程接口芯片因其通用性强、使用方便灵活而成为微型计算机接口电路中的常用芯片。本章既介绍通用可编程接口芯片的一般概念与特征，也介绍了具体芯片的功能与使用方法。

典型的并行接口芯片 Intel 8255 是一个流行的可编程接口芯片，可根据应用的要求工作

于 3 种不同的工作方式以实现数据的输入或输出操作。串行接口电路是以 Intel 8251 为例的，这个芯片可以实现同步或异步的串行数据传送。Intel 8253 是一个定时器/计数器芯片，拥有 3 个独立的 16 位通道，可以通过编程设计其 6 种不同的工作方式。MC6845 作为一个 CRT 控制器，通过了解其内部寄存器的功能并设置合理的数据，可以对 CRT 的光栅扫描产生定时控制，并与 CRT 接口电路一起提供串行的显示信号。另外，可编程中断控制器 Intel 8259 和可编程 DMA 控制器 Intel 8237 都属于在同类产品中较典型的产品，能够对中断向量、中断方式、DMA 数据传送的存储器地址、DMA 工作方式等预先进行设置，这类芯片在 PC 系列机中都有实际的应用实例。

对于本章所列的可编程通用接口芯片，要求了解其功能，掌握其使用方法，其最终目的是通过对这些典型产品的了解学习相关接口电路的一般原理。

习 题 5

5.1 试说明可编程接口芯片的特点，并归纳对接口芯片进行工作方式设置等编程采用哪些方法。

5.2 可编程并行接口芯片 8255A 有哪几种工作方式？各适用于什么场合？端口 A、端口 B 和端口 C 各可以工作于哪几种工作方式？

5.3 8255A 的方式选择字和置位复位字都写入什么端口？用什么方式区分它们？

5.4 设 8255A 的 A 口、B 口、C 口和控制寄存器的端口地址为 80H、82H、84H 和 86H。若 A 口工作在方式 0 输入，B 口工作在方式 1 输出，C 口各位的作用是什么？控制字是什么？若 B 口工作在方式 0 输出，A 口工作在方式 1 输入，C 口各位作用是什么？控制字是什么？

5.5 设 8255A 的 A 口、B 口、C 口和控制寄存器的端口地址为 80H、82H、84H 和 86H。要求 A 口工作在方式 0 输入，B 口工作方式 0 输入，C 口高 4 位输入、低 4 位输出。试编写 8255A 的初始化程序。

5.6 有 8 个发光二极管，提供高电平，二极管发光；提供低电平，二极管熄灭。现要求 8 个发光二极管依次轮流点亮，每个点亮时间为 500ms。试用 8255A 设计完成该功能的电路及相应的程序。

5.7 8253 芯片的计数与定时功能有哪些区别？

5.8 Intel 8253 内部有 3 个定时计数器通道，CPU 在设置定时或计数初值时如何区分不同的通道？在设置工作方式时又如何区分是对哪一个通道进行设置的？

5.9 8253 的工作时钟频率为 3.993 6MHz，要求用 8253 产生一个频率为 1 200Hz 的方波，应当如何设置其工作方式？试写出相应的程序段。

5.10 某一应用系统中，8253 地址为 340H～343H，定时器 0 用做分频器（*N* 为分频系数），定时器 2 用做外部事件计数器，如何编制初始化程序？

5.11 GATE 信号在 8253 的各种工作方式中所起的作用是什么？

5.12 Intel 8251 是一个串行通信接口芯片，当 Intel 8251 工作于异步方式，波特率为 9 600b/s，波特率因子取 16 时，接受器频率应如何选择？

5.13 某数据装置通过 Intel 8251 芯片与微机相连，相互之间以异步方式传送数据。格式为：传送 ASC Ⅱ码字符，偶校验，两位停止位。传送波特率为 1 200b/s，TxC 和 RxC 的时钟频率为 1 200Hz，试确定其方式选择字和命令字。

5.14 单片 8259A 在完全嵌套中断工作方式下，要写哪些初始化命令字及操作命令字？

5.15 8259A 的初始化命令字和操作命令字的含义各是什么？二者有何差别？

5.16 试按下列要求对 8259A 进行初始化：系统 CPU 为 8086，系统中有一片 8259A，中断申请信号采

用电平触发，中断类型号为 60H，61H，…，67H，采用特殊嵌套，非缓冲方式，中断自动结束方式，8259A 的端口地址为 83H 和 84H。

5.17　8237A 具有几个 DMA 通道？每个通道有哪几种传送方式？各用于什么场合？什么叫自动预置方式？

5.18　8237A DMA 控制器的当前地址寄存器、当前字节寄存器、基地址寄存器和基字节寄存器各保存什么值？

5.19　试说明 MC6845 的输出引脚 MA_{13}～MA_0 和 RA_4～RA_0 的作用。

5.20　有一 CRT 显示器，屏幕显示 80×25 字符，每个字符由 8×8 点阵组成，帧频为 50Hz，水平扫描全程 100 个字符，垂直扫描全程 250 条扫描线。试完成 CRT 接口电路的逻辑框图，计算 CRTC 的点时钟频率，并根据要求设计 MC6845 中的各寄存器的取值。

第6章　模 拟 接 口

模拟接口是微型计算机与外部世界进行信息联系与转换的桥梁。模拟接口的核心器件是模/数转换器和数/模转换器，其中模/数转换器是将模拟量转换为数字量供 CPU 处理，而数/模转换器则是将计算机中的数字量转换为模拟量用于输出或控制。在计算机控制与测量仪表的应用中，大量使用模拟接口电路。

6.1　概述

在实际工程中，大量遇到的是连续变化的物理量。所谓“连续”，包含两方面的含义：一方面从时间上来说，它是随时间连续变化的；另一方面从数值上来说，它的数值也是连续变化的。通常称这种连续变化的物理量为模拟量。例如，温度、压力、流量、位移、转速以及连续变化的电流、电压等。而微型计算机只能处理数字量的信息，模拟接口的作用就是实现模拟量与数字量之间的转换。

6.1.1　模拟通道

1．模拟通道结构

一个典型的实时控制系统通常由模拟量输入通道、模拟量输出通道和微型计算机系统组成。其中，传感器、放大器、低通滤波、多路开关、采样保持、A/D 转换电路组成模拟量输入通道，D/A 转换电路、模拟控制电路组成了模拟输出通道，如图 6-1 所示。

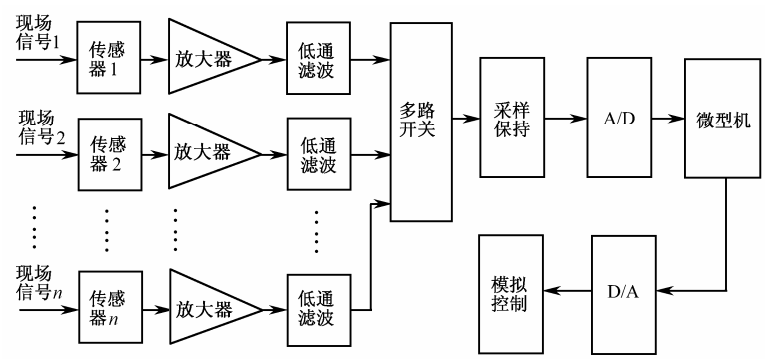


图 6-1　模拟输入/输出通道

（1）传感器。能够把生产过程的非电物理量转换成电流或电压的器件称为传感器。例如，热电耦能够把温度这个物理量转换成几毫伏或几十毫伏的电信号，因此可作为温度传感器。有些传感器不是直接输出电信号，而是把电阻值、电容值或电感值的变化作为输出量，反映相应的物理量的变化。例如，热敏电阻也可以作为温度传感器。

（2）量程放大器。通常 A/D 转换器的输入有以下几种电压等级：双极性为(0～±2.5)V、(0～±5)V、(0～±10)V；单极性为(0～5)V、(0～10)V、(0～20)V 等。不同的传感器的输出

电信号各不相同，因此需要经过信号处理环节，量程放大器就是将传感器输出的信号放大或处理成与 A/D 转换器输入要求相适应的电压范围。

（3）低通滤波器。由于传感器与现场信号相连接，处于恶劣工作环境，其输出叠加有干扰信号。因此信号处理包括低通滤波电路，以滤去干扰信号。

（4）多路转换开关。生产过程中，要监测或控制的模拟量往往不止一个，尤其是在数据采集系统中，需要采集的模拟量一般比较多，而且有不少模拟量是缓慢变化的信号。对于这类模拟信号的采集，为了降低成本，可以用多路模拟开关，使多个模拟信号共用一个 A/D 转换器轮流进行采样和转换。

（5）采样保持电路。在 A/D 转换器进行采样期间，保持输入信号不变的电路称为采样保持电路。由于输入模拟信号是连续变化的，而 A/D 转换器要完成一次转换是需要时间的，这段时间称为转换时间。不同类型的 A/D 转换芯片，其转换时间不同。对于变化较快的模拟输入信号来说，如果不采取措施，将会引起转换误差。显然，在同样频率的情况下，A/D 转换器的转换时间越慢，对模拟信号的转换精度的影响就越大。为了保证转换精度，可采用采样保持电路，使在 A/D 转换期间，保持采样输入信号的大小不变。

（6）A/D 转换器。这是模拟量输入通道的核心环节。其作用是将模拟输入量转换成数字量，以便由计算机读取，进行分析处理。

（7）D/A 转换器。D/A 转换器是模拟量输出通道的核心。D/A 转换器将微型计算机的处理结果转换为模拟量输出。

（8）模拟控制。作为控制用途的模拟输出一般都是经过直流驱动功放，来驱动直流伺服装置的。这种装置可能是直流电机或其他装置，根据 D/A 转换器输出的模拟量（电流、电压或频率等）大小来控制电机的转速。

实际上，在一个由计算机参与的控制系统中，计算机输出的控制信号可能是模拟量，但更多的是开关量，用“0”和“1”来驱动和控制。常用的功率开关接口器件及电路有：由功率晶体管或达林顿电路组成的大功率开关驱动电路、机械继电器或功率型光电耦合器、集成驱动芯片、固态继电器等。

2．采样保持电路

采样保持电路广泛应用于数据采集系统和实时控制系统中，其功能有以下两方面：

（1）采样跟踪状态。即在此期间应尽可能地接受输入信号，使输出和输入信号一致。

（2）保持状态。即把采样结束前瞬间的输入信号保持下来，使输出和保持的信号一致。由于 A/D 转换需要一定的时间，在此期间要求信号保持稳定。因此，当输入信号变化率较快时，都应采用采样保持电路；如果输入信号变化缓慢，则可不用保持电路。

采样保持电路的工作原理如图 6-2 所示。

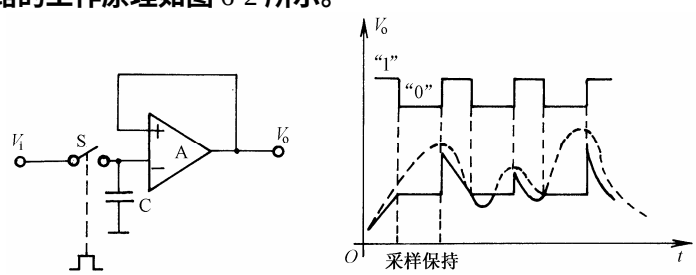


图 6-2　采样保持电路的工作原理图

此电路由模拟开关 S、存储元件 C 及运算放大器 A 所组成。模拟开关由数字指令控制，当数字指令为“1”时，模拟开关 S 接通，存储在 C 上的信号跟踪输入信号，经运算放大器输出，这就是采样过程，这段时间称为采样时间。当数字指令为“0”时，模拟开关 S 断开，存储电容 C 将 S 断开瞬间的输入信号保持下来，并通过放大器输出，即输出信号保持断开瞬间的输入信号，这个过程称为保持过程，这段时间称为保持时间。

微型计算机控制系统的特点是必须用采样开关将连续变化的模拟信号转变为离散的脉冲序列。香农采样定理指出：当采样器的采样频率高于或等于连续信号的最高频率的两倍时，原信号才能通过采样器无失真地复现出来。

3．量化与编码

采样后的信号经过量化才能输入到计算机，采样信号经量化后成为数字信号的过程称为量化过程。A/D 转换就是量化的过程，它把采样后的模拟信号转变成数字信号。

图 6-3 就是 3 位 A/D 转换器的量化示意图。3 位 A/D 转换器把模拟范围分割成 8 个离散区间，可把 0～7V 的采样信号量化为数字量，如输入 5V 被量化为 101 数字代码，这样，就可以输入给计算机进行加工处理了。

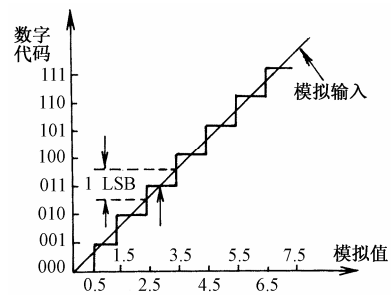


图 6-3　量化示意图

从图中可以看出，量化过程会产生（1/2）LSB（即 0.5V）的误差，要减少量化误差，一般是采取位数更多的 A/D 转换器，把模拟范围分割成更多的离散区间，以减少 LSB 的值。

在量化过程中，对双极性的信号通常有 3 种编码表示方式：

（1）符号表示法。这种方法类似于原码表示法，增加一位符号位，其他数值表示与单极性一样。通常，数值为正时，符号位用“0”表示；数值为负时，符号位用“1”表示。

（2）偏移二进制数。这是一种直接的二进制编码，用满刻度来加以偏移。正值（包括 0 在内）时符号位均为“1”，而在负值时符号位均为“0”。这样的编码常在微型计算机的双极性模拟量转换中使用。

（3）补码表示法。这种方法与计算机的补码表示方法相同，其符号位的特征和偏移二进制码相反，而数值部分相同。

6.1.2　模拟接口电路的性能指标

模拟接口电路的性能指标是反映模拟通道的性能，涉及到模拟通道中的所有元器件的性能。由于 A/D 转换器和 D/A 转换器是模拟接口电路中的主要器件，这里仅讨论两类器件的性能指标。

1．D/A 转换器的性能指标

D/A 转换器的性能可通过它的参数反映出来，这些参数是用户选择和使用 D/A 转换器的依据。D/A 转换器的主要性能指标可分为静态指标、动态指标、环境和工作条件指标 3 种。分辨率、精度、输出范围属于静态指标；建立时间、尖峰属于动态误差；环境和工作条件因素主要是指温度和电源电压的变化。

（1）分辨率。这是 D/A 转换器对微小输入数字量变化的敏感程度的描述，即输入数字的最低有效位 LSB 变化为 1 时所引起的输出模拟的变化，通常用数字量的位数来表示。当 D/A 转换器输入数字量的二进制数字为 N 位时，则该 D/A 转换器的分辨率为 2^N，常见的二进制位数有 8 位、10 位和 16 位等。

分辨率的另一种表示方法是输出模拟量的最小变化量相对于输出模拟量满度值的百分比。当二进制数字量为 N 位时，分辨率为 $1/2^N$ 或（$1/2^N$）×100%，如 8 位 D/A 转换器的分辨率为参考电压 V_R 的 1/256，用百分数表示约为 0.39% 。

常见的几种 D/A 转换器的分辨率如表 6-1 所示。

表 6-1　常见 D/A 转换器的分辨率

位数 N	用输入数字时的位数 N 表示	用 2 表示	用 1／2 表示	用（1/2 ）^{N}×100%表示
4	4 位	2^4	0.062 5	6.25%
6	6 位	2^6	0.016	1.6%
8	8 位	2^8	0.004	0.4%
10	10 位	2^{10}	0.001	0.1%
12	12 位	2^{12}	0.000 2	0.02%
14	14 位	2^{14}	0.000 06	0.006%
16	16 位	2^{16}	0.000 015	0.001 5%

（2）精度。D/A 转换器的精度是指其模拟输出电平与理想的输出值之间所存在的最大偏差，也就是 D/A 转换器实际的转换特性曲线与理想的转换特性曲线之间的最大偏差。

D/A 转换器的精度有绝对精度和相对精度之分。绝对精度是指对应于给定的满刻度数字量，D/A 转换器实际输出与理论值之间的误差。而相对精度是指在满刻度已校正的情况下，在整个量程范围内对应于任一数字量的模拟量输出与理论值之差。

通常 D/A 转换器的精度都用相对精度来描述。相对精度常用百分数来表示，或用最低位（LSB）的几分之几来表示。

在 D/A 转换器中，精度与分辨率是两个不同的概念，如一个 10 位的 D/A 转换器，它的分辨率能达到参考电压 V_R 的 1/1 024，约 0.01%，若它的精度只能达到 8 位 D/A 转换器的精度，那么，这个 10 位分辨率的 D/A 转换器就只能当做 8 位来使用。

在 D/A 转换中，影响转换精度的主要误差因素有失调误差、增益误差、非线性误差和微分非线性误差。

（3）输出范围。输出范围是指当 D/A 转换器的所有位全部由“0”变到“1”时所对应的输出电压值。如 8 位 D/A 转换器，输出电压的范围为 0～255×(V_R/256)。

实际上，当 D/A 转换器的位数以及输出范围确定以后，其步长也就确定了。所谓步长，是指两个相邻数字量间隔所对应的模拟量间隔。例如，某一 8 位 D/A 转换器的电压输出范围是 0～5V，其步长为：

$$\text{步长}=5\text{ V}/2^8=5\text{ V}/256=0.019\,531\,25\text{ V}$$

已知步长，就可以计算数字量与模拟量之间的关系了。

与输出范围相关的还有两个概念：标称满量程和实际满量程。

标称满量程是指相应于数字量标称值 2^N 的模拟输出量。显然，实际数字量最大为（2^N–1），比标称值小 1 个 LSB，因此，实际满量程要比标称满量程小 1 个 LSB 增量。例如，1 个 10

位 D/A 转换器，其标称满量程为＋5 V，而实际满量程为：

（1 023／1 024）×5 V＝4.995 117 187 5 V

（4）建立时间。建立时间是指在规定的误差范围内输出信号幅度达到要求的时间。当 D/A 转换器的输入数字信号发生变化时，输出电压也随着改变，但由于实际器件的工作速度有限，当输出电压达到稳定时已比输入数字信号发生变化的时刻延迟了一段时间，这段时间就是建立时间。根据转换时间可以计算 D/A 转换器的最大工作频率。如某 D/A 指标中的建立时间为 300 ns，输出稳定到全程范围的 0.2%，那么，该 D/A 在输入某一确定数字 300 ns 以后，它的输出模拟电压与无限长时间的稳定输出电压比较，误差不超过 0.2%。不同性能 D/A 转换器，其建立时间从几十纳秒～几微秒不等。

实际建立时间的长短不仅与转换器本身的转换速率有关，还与数字量变化的大小有关。输入数字从全“0”到全“1”或从全“1”到全“0”时，建立时间最长，称为满量程变化的建立时间。一般手册上给出的都是满量程变化的建立时间。

根据建立时间的长短，D/A 转换器可以分成以下几档：

超高速　　<100 ns

较高速　　100 ns～1 μs

高速　　1～10 μs

中速　　10～100 μs

低速　　>100 μs

（5）尖峰。尖峰是输入数码发生变化时产生的瞬时误差。尖峰的持续时间虽然很短，但幅值可能很大。在有些应用场合下，必须采用措施加以避免。

产生尖峰的原因是由于开关在换向过程中，“导通”的延迟时间与“截止”延迟时间不相等所致。若模拟开关“截止”则延迟时间较短，“导通”则延迟时间较长，而 D/A 转换器的输入数字是逐一增加的，可能出现图 6-4 所示的尖峰波形。例如当输入码由 011…11 变到 100…00 时，实际上只增加了 1LSB，由于开关电路从 1→0 比从 0→1 的响应要快，结果在转换过程中会出现 000…00 状态，使模拟输出向下猛跌，造成一个很大的尖峰误差。当然实际的尖峰大小还决定于电路各元件的响应速度和其他参数的影响。但是，凡是有从 1→0 的时刻都可能产生尖峰，而且发生的位数越高，尖峰的幅值一般也越大。

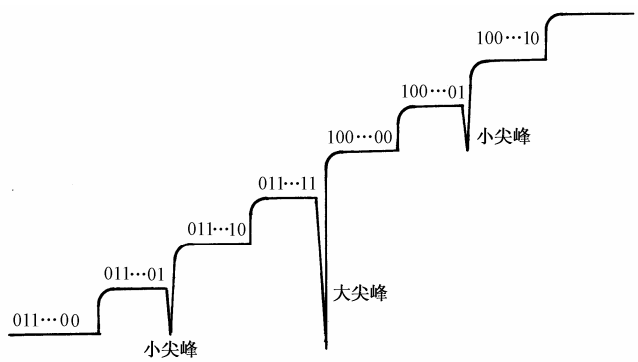


图 6-4　D/A 转换时产生的尖峰波形

（6）环境及工作条件影响指标。一般情况下，影响 D/A 转换精度的主要环境和工作条件因素是温度和电源电压的变化。

D/A 转换器的工作温度按产品等级分为军级、工业级和普通级，标准军级品可工作于-55～＋125℃，工业级工作温度为-25～＋85℃，普通级工作温度为 0～70℃。多数器件其静态、动态指标都是在 25℃的温度环境下测得的，环境温度对各项精度指标的影响用其温度系数来描述。温度系数是指在规定的范围内，温度每变化 1℃，D/A 转换器所引起的多种参数的变化量。

D/A 转换器受电源变化影响的指标为电源变化抑制比，这是用电源变化 1 V 时所产生的输出误差相对满量程的比值来描述的。

2．A/D 转换器的性能指标

A/D 转换器的性能可通过它的参数反映出来，这些参数是用户选择和使用 A/D 转换器的依据。衡量 A/D 转换器性能的主要参数是：

(1) 分辨率。分辨率是输入模拟量的最小变化量，通常用对应输出数字的最低有效位 LSB 变化 1 所需要的输入电压变化相对于输入电压满度值的百分比来表示。如 8 位 A/D 转换器的分辨率为 1/256，约为 0.39%。位数越多，分辨率越高。

(2) 精度。精度是指输入模拟信号的实际电压值与被转换成数字信号的理论电压值之间的差值，也叫绝对误差。当它用百分数表示时叫做相对误差。相对误差也常用最低有效值的位数 LSB 来表示。误差的主要来源有量化误差、零位偏差、增益误差和非线性误差等。例如，有一个 8 位 0～5 V 的 A/D 转换器，如果其相对误差为 1LSB，则其绝对误差为 19 mV，相对误差为 0.39%。一般来说，位数越多，其误差就越小。

(3) 转换时间。转换时间是 A/D 完成一次转换所需的时间。通常是 A/D 的最长转换时间的典型值。

(4) 电源灵敏度。电源灵敏度是指当电源电压发生变化时，会使 A/D 转换器的输出发生变化，这种变化的实际作用相当于 A/D 转换器输入量的变化，从而产生误差。通常 A/D 转换器对电源变化的灵敏度用相当于同样变化的模拟输入值的百分数来表示。

6.2 数/模转换接口

数模转换接口的功能是将 CPU 提供的数字信号转换为模拟信号。在实际应用中，主要考虑以下两方面的问题：

第一，D/A 转换器如何与 CPU 相连接，D/A 转换器输出的模拟量是电流还是电压，如何连接与使用。

第二，CPU 如何通过程序来控制 D/A 转换器实现预定的功能。

6.2.1 D/A 转换器与 CPU 的连接

D/A 转换器与 CPU 的连接时要考虑到 D/A 转换的具体特性。一般来说，D/A 转换器的数字量位数与 CPU 的数据总线位数是否一致以及 D/A 转换器内部是否拥有数据锁存器，都会影响到 D/A 转换器接口电路的构成。

由于 D/A 转换器将数字量转换为模拟量是一个过程，这个过程是需要时间的，并且在数字量转换为模拟量的过程中为了保证转换工作的正确进行，应该保证提供给转换电路的数字量保持不变。因此，需要有一个数据寄存器来保存待转换的数据。如果某一个 D/A 转换器芯

片本身不含有数据寄存器,那么在D/A转换器的接口电路中就应该提供这样一个数据寄存器。当CPU执行输出指令时,这个寄存器就可以保存供转换的数字量直到下一次寄存器的值被修改。如果D/A转换器芯片内部包含有数据寄存器,那么D/A转换器的数据输入线可以直接连在CPU的数据总线上,并正确处理好D/A转换器的其他控制信号,如数据锁存信号。

至于D/A转换器的数字量的位数,如果与CPU数据总线的位数相同,则D/A转换器的数据输入线可以与CPU的数据总线对应相连。

如果D/A转换器的数字量位数小于CPU的数据总线位数,则D/A转换器的数据输入线可以与CPU的部分数据线相连接(通常连接CPU的数据总线低位部分),这样,CPU在执行输出指令向D/A转换器提供转换数据时,要注意有效数据对应哪位,是数据线的低位还是高位。例如,一个16位的CPU与8位的D/A转换器相连接,若D/A转换器的8位数据线连到CPU的低8位数据总线上,则当CPU执行输出指令向D/A转换器写数据时,只有低8位数据是有意义的。

如果D/A转换器的数据位数大于CPU数据总线的位数,则CPU需要通过多次将数据分时写入D/A转换器,并设法同时将数字量提供给D/A转换器。如8位的CPU要连接16位的D/A转换器,CPU需要分两次将16位的转换数据写入到D/A转换器中,先写入低8位,后写入高8位,最后统一提供转换开始信号。

在下面的D/A转换器与CPU的连接举例中,假设都采用8位CPU。

1．不带锁存器的8位D/A转换芯片的使用

对于一个D/A转换部件来说,当等待转换的数据加到输入端时,在输出端也随之建立了相应的电流或电压。对于没有数据输入寄存器的D/A转换器来说,随着输入数据的变化,输出模拟量也随之变化,同理,当输入数据消失时,输出模拟量也会相应消失。以80286CPU为例,待转换的数据来自CPU,但CPU在执行输出指令时在数据总线上维持的有效数据只有2个时钟周期左右,这样,模拟量的输出时间也很短,难以在具体应用中体现实际意义。因此,为了能维持模拟量的输出,通常的做法是在D/A转换器之前增加一个数据锁存器,再与总线相连,如图6-5所示。

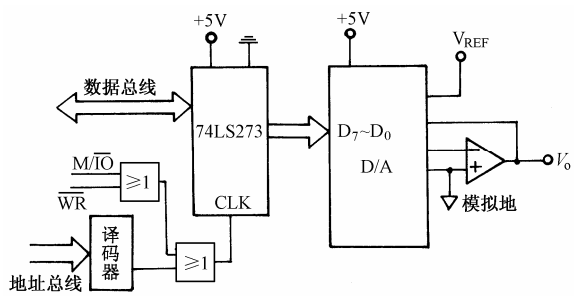


图6-5　不带数据锁存的D/A转换器与CPU的连接

图中,译码器的连接方法决定了锁存器的端口地址,当CPU用输出指令向这个端口输出一个数据时,就把数据送入这个锁存器,D/A转换器的输出得到相应的电压。

2．不带锁存器的 12 位 D/A 转换芯片的使用

针对 8 位 CPU，如果一个 D/A 转换器超过 8 位，这时，用一个锁存器就不够了。如一个 12 位的 D/A 转换芯片就需要用两个数据锁存器和总线相连。工作时，CPU 通过两条输出指令往两个锁存器对应的端口输出数据，就可以完成数据的传送了。具体的连接方法如图 6-6 所示。

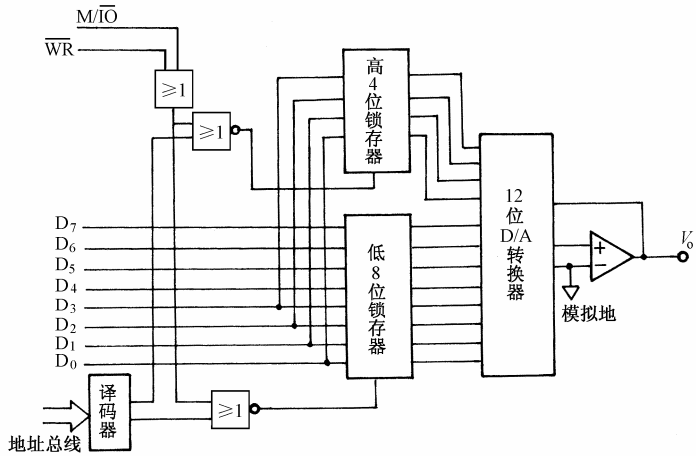


图 6-6　超过 8 位的 D/A 转换芯片的连接

但是，D/A 转换器只有在 CPU 执行两个输出指令后才能得到所需数据，而在第一次执行输出指令以后，D/A 转换器会得到一个局部的输入，因此，输出端会得到一个局部的、实际上不需要的模拟量输出。显然，这是一个需要避免的现象。因此，往往采用两级数据缓冲结构来解决 D/A 转换器与数据总线的连接问题。如图 6-7 所示，CPU 先用两条输出指令把数据送到第一级数据缓冲器，然后通过第三条输出指令使数据送到第二级数据缓冲器，从而使 D/A 转换器一次得到 12 位待转换的数据。

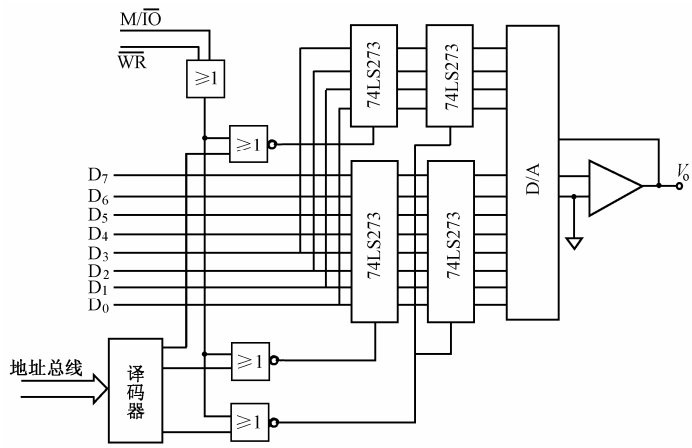


图 6-7　使用二级数据缓冲的 D/A 转换器与 CPU 的连接

在二级缓冲结构中，第一级缓冲器直接与 CPU 数据总线相连，CPU 将数据通过数据总线写入。第二级缓冲器的数据来源于第一级数据缓冲器，所以 CPU 执行输出指令的目的仅仅是为第二级缓冲器提供选通信号，以便能将第一级缓冲器的数据打入到第二级缓冲器中。具

体程序如下：

```
MOV   AL，      DATAL
OUT   PORTL，   AL              ；低 8 位数据送第一级缓冲器
MOV   AL，      DATAH
OUT   PORTH，   AL              ；高 8 位数据送第一级缓冲器
OUT   PORT，    AL              ；使数据打入第二级缓冲器
```

3．带数据锁存器的 D/A 转换芯片的使用

随着集成电路技术的发展，目前已能将精密电阻、模拟开关、数据锁存器，甚至包括基准电阻和运算放大器集成在同一片芯片上，和 8 位或 16 位微处理器兼容，可以直接连接 CPU。因此，数模转换器与 CPU 相关的接口电路信号有：

（1）数据线。用于实现数模转换器接口与 CPU 的数据交换。

（2）地址线及地址译码信号。用于选择数模转换器的接口地址或 D/A 电路内部不同的寄存器。

（3）写控制信号。用于确定 CPU 当前对数模转换器的写入。

DAC0832 就是一个内部具有锁存器的 D/A 转换器，CPU 可以直接通过系统总线与 DAC0832 相连接。

6.2.2 D/A 转换器应用举例

DAC0832 是美国国家半导体公司的 8 位 D/A 转换器，可以直接与常用微处理器相连。它采用 CMOS 工艺，是具有 20 个引脚的双列直插式器件。

DAC0832 的结构如图 6-8 所示。

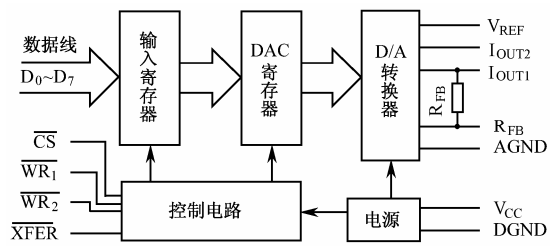


图 6-8　DAC0832 结构图

DAC0832 由 3 大部分组成：一个 8 位输入寄存器，一个 8 位 DAC 寄存器和一个 8 位 D/A 转换器。在 D/A 转换器中采用的是 R-2R 电阻网络，输出为电流信号。DAC0832 器件由于有两个可以分别控制的数据寄存器，使用时有较大的灵活性，可以根据需要连接成多种方式，如有两级输入锁存的双缓冲方式，只用一级输入锁存的单缓冲方式，或者完全直通的无缓冲器方式。

DAC0832 的引脚排列，如图 6-9 所示。DAC0832 的引脚可分为两部分：一部分与外设相连，另一部分与

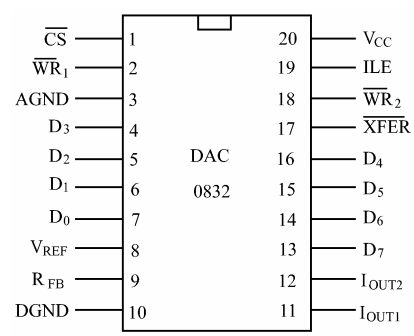


图 6-9　DAC0832 引脚功能图

CPU 相连。

用于同外设相连的引脚有：

I_{OUT1}：DAC 电流输出 1。当 DAC 寄存器为全 1 时，表示 I_{OUT1} 为最大值，当 DAC 寄存器为全 0 时，表示 I_{OUT1} 为 0。

I_{OUT2}：DAC 电流输出 2。I_{OUT2} 为常数减去 I_{OUT1}，即 $I_{OUT1}+I_{OUT2}$=常数。在单极性输出时，I_{OUT2} 通常接地。

R_{FB}：反馈电阻，它为外部运算放大器提供一个反馈电压。R_{FB} 可由内部提供，也可由外部提供。

用于同 CPU 连接的信号有：

D_7～D_0：数字输入量。D_0 是最低位（LSB），D_7 是最高位（MSB）。

$\overline{CS}$：片选信号引脚（低电平有效）。

ILE：输入锁存允许信号（高电平有效）。

$\overline{WR_1}$：写 1（低电平有效）。当 $\overline{WR_1}$ 为低电平时，用来将输入数据传送到输入锁存器；当 $\overline{WR_1}$ 为高电平时，输入锁存器中的数字被锁存；当 ILE 为高电平且 $\overline{CS}$ 和 $\overline{WR_1}$ 同时为低电平时，才能将锁存器中的数据进行更新。以上 3 个控制信号构成第一级输入锁存。

$\overline{WR_2}$：写 2（低电平有效）。该信号与 $\overline{XFER}$ 配合，可使锁存器中的数据传送到 DAC 寄存器中进行转换。

$\overline{XFER}$：传送控制信号（低电平有效）。$\overline{XFER}$ 将与 $\overline{WR_2}$ 配合使用，构成第二级输入锁存。

除此以外，DAC0832 还有 4 根引脚：

V_{REF}：参考电压输入，要求外部接一个精密的电源。其输入电压范围为±10 V。在四象限的应用中可作为模拟量输入。

V_{CC}：数字电路供电电压，一般为+5～+15 V。

AGND：模拟地。

DGND：数字地。这是两种不同的地，但在一般的情况下，这两个地最后总有一点接在一起，以提高抗干扰能力。

DAC0832 与 CPU，需要进行 3 方面的连接：数据总线的连接、地址总线的连接和控制总线的连接。对于 8 位 CPU，如 Intel 8088、DAC0832 的数据线 D_7～D_0 可直接连至 CPU 的数据总线。对于 CPU 的地址总线，应将 CPU 的地址线经译码以后连接 DAC0832 的片选端 $\overline{CS}$，不同的译码对应不同的端口地址。DAC0832 的引脚 $\overline{WR_1}$ 或 $\overline{WR_2}$ 应连接至 CPU 的 $\overline{WR}$ 或 $\overline{IOW}$，CPU 在执行 OUT 指令时能对 DAC0832 执行写操作。DAC0832 的 $\overline{CS}$ 与 $\overline{XFER}$ 输入一般采用地址译码电路。

DAC0832 有 3 种工作方式：

（1）双缓冲工作方式。DAC0832 芯片内有两种工作方式，在双缓冲工作方式下，CPU 要对 DAC 芯片进行两步写操作，先将数据写入输入寄存器，后将输入寄存器的数据写入 DAC 寄存器。其连接方式是：把 ILE 固定为高电平，$\overline{WR_1}$、$\overline{WR_2}$ 均接到 CPU 的 $\overline{IOW}$ 端，而 $\overline{CS}$ 和 $\overline{XFER}$ 分别接到两个端口的地址译码信号引脚。

双缓冲工作方式的优点是 DAC0832 的数据接收和启动转换可异步进行。可以在 D/A 转换的同时，进行下一个数据的接收，以提高模拟输出通道的转换速率，可实现多个模拟输出通道同时进行 D/A 转换。

（2）单缓冲工作方式。此方式是使两个寄存器中的一个处于直通状态，另一个工作于受锁存器控制状态。一般是使 DAC 寄存器处于直通状态，即把 $\overline{WR_2}$ 和 $\overline{XFER}$ 端都接数字地。此时，数据只要一写入 DAC 芯片，就立即进行数/模转换。此种工作方式可减少 1 条输出指令，在不要求多个模拟输出通道同时刷新模拟输出时可采用此种方式。

（3）直通工作方式。将 $\overline{CS}$、$\overline{WR_1}$ 和 $\overline{XFER}$ 引脚都直接接数字地，ILE 引脚为高电平时，芯片即处于直通状态。此时，8 位数字量一旦到达 D_0～D_7 输入端，就立即进行 D/A 转换并输出。但在此种方式下，DAC0832 不能直接和 CPU 的数据总线相连，故很少采用。

图 6-10 为 DAC0832 采用单缓冲方式与 CPU 的连接图。

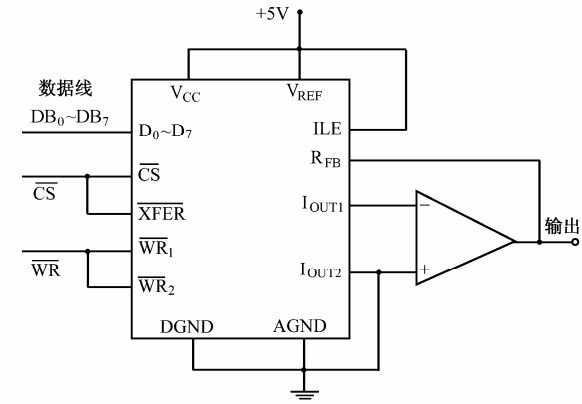


图 6-10　DAC0832 与 CPU 的连接

在图 6-10 的电路中，$\overline{WR_1}$ 与 $\overline{WR_2}$ 接在一起，与 CPU 的 $\overline{WR}$ 信号相连；$\overline{CS}$ 与 $\overline{XFER}$ 接在一起，与地址译码电路的片选信号相连；ILE 直接接高电平。设 DAC0832 的片选信号 $\overline{CS}$ 的地址为 80H，参考电压 V_{REF}=+5V，要使 DAC0832 输出 2V 的模拟电压，则相应的数字值应为 255×(2/5)=102，即 66H，可用下列指令实现：

```
MOV     AL，66H        ；数据送入 AL
OUT     80H，AL        ；输出到 D/A
```

用图 6-10 的电路，还可以输出各种波形。例如执行下列程序后，可以输出阶梯波。

```
L2：MOV     AH，10H
    MOV     AL，00H
L1：OUT     80H，AL
    INC     AL
    DJNZ    AH，L1
    JMP     L2
```

6.3　模/数转换接口

实现模拟量到数字量转换的模拟接口就是 A/D 转换接口。A/D 转换接口是一组经过控制能与微处理器的数据总线或并行口相连接的逻辑电路。它能接受来自外设的模拟信号，并将

其转换成相应的数字信号输出。

一般而言，A/D 转换接口电路应具有如下两方面的功能：

（1）实现与系统总线或并行口相连接，提供数据的输出功能，这是模数转换接口的基本功能。

（2）实现与外部设备相连接，具有处理外部设备提供的模拟量的输入能力，使 A/D 转换器能得到正常工作所需的模拟量。

要实现 A/D 转换的功能，除了必须的 A/D 转换器以外，还要有模拟信号的处理电路，使输入的模拟量能满足 A/D 转换器的要求；应具有相关的控制电路，实现 CPU 对 A/D 器件的控制；还必须具有相应的数据输出锁存器，以便 CPU 能读取转换结果。

因此，A/D 转换电路应由以下几个部分组成：A/D 转换电路、数据输出锁存器以及其他控制与信号处理电路。

6.3.1 A/D 转换器与 CPU 的连接

在 A/D 转换过程中，输入的模拟信号一般需处理后才能由 A/D 转换器进行转换，如使用采样保持电路、低通滤波器等电路。外设提供的模拟信号经低通滤波器滤除噪声后，加到采样保持电路。在采样保持信号的作用下，采样信号控制采样电路将输入的模拟信号离散化，并在保持信号的控制下，使该信号经保持电路后再送入 A/D 转换器进行模/数转换。A/D 转换器与 CPU 相关的接口电路信号有：

（1）数据线，用于实现 A/D 转换器与 CPU 的数据交换。

（2）地址线及地址译码信号，用于选择 A/D 转换器的输入通道及接口电路内部的锁存器。

（3）读、写控制信号，用于确定 CPU 当前对 A/D 转换器的读出或写入。

因此，A/D 转换器和 CPU 的连接，与 D/A 转换器的连接相似，仍然是从地址线、数据线、控制线 3 方面考虑。由于 A/D 转换器的数据位数是否与 CPU 的数据总线位数相同、是否带三态数据缓冲器、是否提供转换结束信号、是否支持多路模拟量的输入等各不相同，故 CPU 与 A/D 转换器的接口电路也有所不同。

1．A/D 转换器数据输出线与 CPU 数据总线的连接

A/D 转换器芯片一般有两种数据输出方式：提供数据三态输出控制和不提供数据三态输出控制。

对于提供数据三态输出的 A/D 转换器芯片，可以直接连接 CPU 的数据总线，由读信号和口地址译码信号控制三态门。在数/模转换结束后，CPU 通过执行一条输入指令产生读信号，就可以将数据从 A/D 转换器中取出了。

如果 A/D 转换器的数据输出端没有三态门或者三态门无法控制，则 A/D 转换器的数据输出不能直接连接到 CPU 的数据总线，而是必须通过 I/O 通道或者附加的三态门电路实现 A/D 转换器与 CPU 的连接。

如果 A/D 转换器输出的数据位数与 CPU 数据总线位数不相同，则要通过硬件连接与指令执行相配合，才能读取有效数据。

以 8 位 CPU 为例，如果 A/D 转换器的数据输出也是 8 位，则直接相连。如果 A/D 转换器的数据输出端小于 8 位，则与 CPU 数据总线的部分线相连，在 CPU 执行输出指令时要提取相对应的数据位。如果 A/D 转换器的数据输出位大于 8 位，接口电路要提供两个不同口地

址控制的数据输入端口，分两次将高字节和低字节数据读入 CPU 内部。

2．A/D 转换器启动信号的连接

A/D 转换器要求的启动信号一般有两种形式：电平启动信号和脉冲启动信号。

电平启动信号通常要求 A/D 转换器在整个转换过程中启动信号始终保持有效，如果中途撤回启动信号，那么就会停止转换或转换出错。为此，一般 CPU 要通过并行口来控制启动信号，或用触发器来保持启动信号在整个 A/D 转换期间处于有效电平状态。

针对脉冲启动信号，只需 CPU 在 A/D 转换开始时通过执行输出指令或输入指令对芯片产生脉冲信号，从而启动转换。

3．A/D 转换器转换结束信号的连接

A/D 转换结束，A/D 转换芯片会输出转换结束信号，通知 CPU 读取数据。

处理转换结束信号常用的方法有两种：程序查询方式和中断方式。

第一种方法是在启动 A/D 转换工作以后，程序就不断地读取 A/D 转换结束信号，如果发现结束信号有效，则认为完成一次转换，因而可用输入指令读取数据。

第二种方法是把转换结束信号作为中断请求信号，送到 CPU 的中断请求输入端或中断控制器的中断请求输入端。在中断服务程序响应时，CPU 读取 A/D 转换后的数据。

如果 A/D 转换器芯片不提供转换结束信号，则可采取固定的延迟程序方法。这种方法要预先精确地知道完成一次 A/D 转换所需要的时间。这样，CPU 在发出启动转换命令以后，执行一个固定的延迟程序，A/D 转换结束，于是 CPU 就可以读取转换数据了。

4．多路模拟量输入的处理

在一般的数/模转换应用中，多路的模拟量输入往往都用一个 A/D 转换器来实现数/模转换，这是因为通常对模拟量的采集频率都不是很高，从时间上完全有可能用一个 A/D 转换器来对多路的模拟量进行分时采样，从而降低 A/D 转换器的成本。

对于这样的应用要求，可以将多路模拟量的输入通过一个多选一的模拟开关将选中的某一路模拟量送到 A/D 转换器模拟量输入端。因此，在 A/D 转换的接口电路中，不仅需要多选一的模拟开关电路，还需要 CPU 对开关电路的控制。

部分 A/D 转换器芯片内部已经集成了多选一的开关电路，这样，在 CPU 与这类 A/D 转换器芯片的连接过程中，除了数据线的连接、启动信号的连接、转换结束信号的连接以外，还要考虑通过 CPU 的数据线或地址线对选择开关进行控制。

6.3.2 A/D 转换器应用举例

A/D 转换器根据转换原理的不同可以分成不同类别的转换器，每类转换器的特性各不相同。常用的转换有逐次逼近式 A/D 转换器、积分式 A/D 转换器和计数式 A/D 转换器。逐次逼近式 A/D 转换速度快、分辨率高，是一种常用的数据采集用 A/D 转换器芯片；双积分式 A/D 转换器转换率高，抗干扰性能好，适合于中等速度的系统；计数式 A/D 转换器速度低，价格低，适合于慢速系统。在实际应用中，应根据需要选择不同的 A/D 转换器。

AD570 是一个 8 位的 A/D 转换器芯片，工作时需要+5 V 和 -15 V 两挡电源，完成一次转换需要 25 μs。

AD570 内部在数据输出端带有三态门，但三态门是不能由外部控制的。在 A/D 转换结束时，三态门会自动接通，因此，从转换结束到取走数据这段时间内，输出数据线始终被占用，因此 AD570 就不能直接和系统总线相连。

AD570 的输入模拟电压可以是单极性，也可以是双极性，BIPOLAOFF 输入引脚就是用于选择极性的。若 BIPOLAOFF 引脚输入为低电平，则为单极性输入，模拟电压范围 0～+10 V；否则为双极性输入，模拟电压范围为 -5～+5 V。

AD570 要求用一个低电平作为启动信号，启动信号的输入端为 $B/\overline{C}$ （$BLK/\overline{CONV}$）。

AD570 的转换结束信号为 DR（DATA READY），每完成一次转换，芯片在 DR 引脚输出一个低电平，以通知 CPU 读取数据。

图 6-11 是 AD570 与 CPU 的连接图。AD570 和系统总线之间连接着并行接口 8255A。8255A 的端口 A 连接 A/D 转换器的数据输入端，工作于输入方式。端口 B 也工作于输入方式，PB_0 和转换结束信号 $\overline{DR}$ 相连，因此，用程序读取 PB_0 的值，就可以判断 A/D 转换是否完成。端口 C 工作在输出方式，PC_0 连接 A/D 转换器芯片的启动信号端 $B/\overline{C}$。在工作时，CPU 用输出指令将 PC_0 置为 0，这样，使 AD570 的 $B/\overline{C}$ 端得到一个低电平，从而启动转换。此后，用输入指令不断读取端口 B 的数据，并进行测试，以判断 PB_0 是否为 0，如果 PB_0 为 0，则说明完成一次 A/D 转换，CPU 可以从端口 A 中读取转换结果，否则继续进行查询。

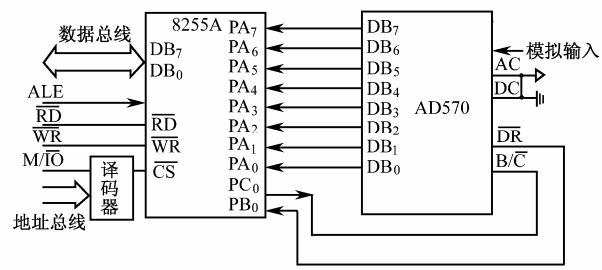


图 6-11　AD570 与系统总线的连接

实现程序查询式读取转换数据的程序如下：

```
START：  MOV   AL，     92H      ；方式字，使端口 A、B 为输入方式，C 为输出方式
         OUT   PORTCT，  AL       ；向控制口 PORTCT 送方式字
         MOV   AL，     01H
         OUT   PORTC，   AL       ；使 PC0 为 1，PORTC 为端口 C 的地址
         MOV   AL，     00H
         OUT   PORTC，   AL       ；使 PC0 为 0，启动 A/D 转换
L1：     IN    AL，     PORTB    ；读取端口 B 中的状态
         RCR   AL，     01       ；如 PB0 为 1，则再查询
         JC    L1
         MOV   AL，     01H
         OUT   PORTC，   AL       ；使 PC0 为 1，撤销启动信号
         IN    AL，     PORTA    ；读取转换数据
```

ADC0809 是逐次逼近型的 8 位 A/D 转换芯片。片内有 8 路模拟开关，可输入 8 个模拟量。ADC0809 为单极性，量程为 0～5 V，典型的转换速度为 100 μs。片内带有三态缓冲器，

可直接与 CPU 总线接口相连。ADC0809 的逻辑结构框图如图 6-12 所示，其结构分为 3 个部分。

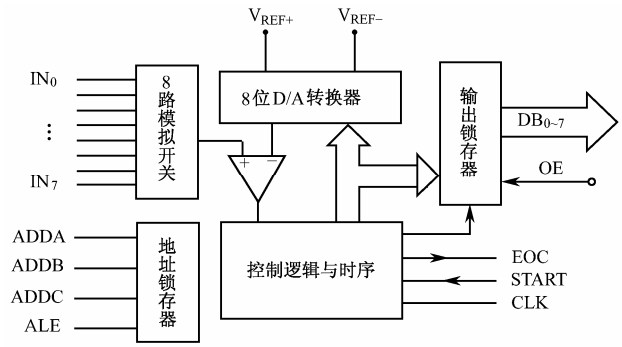


图 6-12　ADC0809 结构框图

模拟输入部分是由 8 路输入的多路转换开关和地址锁存与译码逻辑组成的，3 位地址用于选择 8 路模拟输入中的一路供 A/D 转换。转换器部分由逐位逼近寄存器 SAR、比较器、电阻网络和控制逻辑 4 部分组成，可实现逐位逼近的 A/D 转换。三态输出的数据缓冲器可将转换结果输出。

ADC0809 是一个 28 引脚的双列直插式集成电路芯片。其引脚功能如下：

IN_0～IN_7：8 路模拟输入端。

V_{REF+}、V_{REF-}：基准电压输入端。

ADDA、ADDB、ABBC：模拟输入的地址选择线。

地址选择线与模拟通道的关系如表 6-2 所示。

表 6-2　ADC0809 的通道选择

ADDC	ADDB	ADDA	选中的模拟通道
0	0	0	IN_0
0	0	1	IN_1
0	1	0	IN_2
0	1	1	IN_3
1	0	0	IN_4
1	0	1	IN_5
1	1	0	IN_6
1	1	1	IN_7

ALE：地址锁存信号，高电平有效。有效时，ADDA～ADDC 地址被锁存，从而选通相应的模拟通道，以便对选中的模拟通道进行 A/D 转换。

D_7～D_0：转换后的 8 位数字量输出线，三态。

START：转换启动信号，输入，高电平有效。

EOC：转换结束信号，输出，高电平有效，表示转换结束。

OE：输出允许控制端，高电平有效，表示将输出缓冲器中的数据送上数据总线。

CLK：时钟输入，用于逐次逼近 A/D 转换。

V_{CC}、GND：+5 V 电源和接地端。

从 ADC0809 的结构与引脚功能可以看出，ADC0809 的 8 路模拟开关通过控制 C、B、A 端口和地址锁存允许端口，可使其中一个通道被选中。逐次逼近型 A/D 转换器由比较器、控制逻辑、输出锁存缓冲器以及 D/A 电路组成。控制逻辑用来控制逐次逼近 A/D 转换器的转换过程。经过 8 次比较后，输出到 D/A 的数字量与输入模拟量所对应的数字量相等，此数字被送到输出锁存器中，同时发出转换结束信号 EOC（高电平有效），表示转换结束。此时，CPU 发出一个输出允许命令（OE 为高电平）就可读取数据了。

ADC0809 与 CPU 的连接，同样是 3 方面的连接：数据总线的连接、地址总线的连接和控制总线的连接。ADC0809 可以直接与 CPU 总线相连，也可以通过并行接口与 CPU 相连。下面是 ADC0809 与 CPU 直接相连时的一种接法。

对于 8 位 CPU，ADC0809 的数据线 D_7～D_0 可直接连至 CPU 的数据总线。

对于 CPU 的地址总线，应将 CPU 地址线的低 3 位接到 ADC0809 的通道选择端口 ADDA、ADDB、ADDC 上，其他地址经译码后，可用于选择不同的 ADC0809 电路的端口。

ADC0809 的引脚 START 与 ALE 接在一起，与 CPU 的 $\overline{IOW}$ 与高位地址译码输出端经与门后的输出端相连，CPU 在执行 OUT 指令时能对 ADC0809 执行写操作。CPU 的 $\overline{IOR}$ 与高位地址译码输出经与非门后与 ADC0809 的 OE 相连，CPU 在执行 IN 指令时能对 ADC0809 执行读操作。

图 6-13 为 ADC0809 与 CPU 连接的示意图。

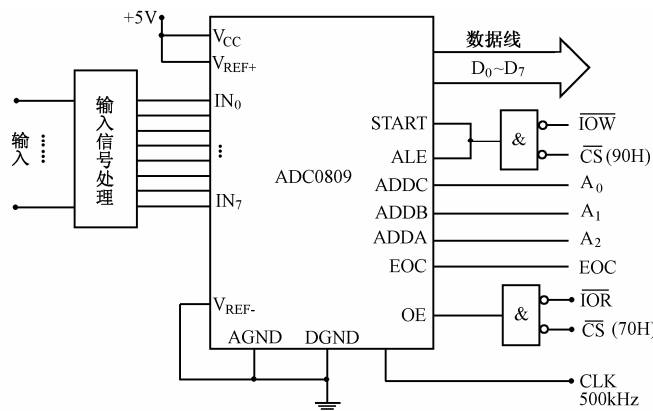


图 6-13　ADC0809 的连接

本 章 小 结

模拟接口可分为输入接口和输出接口。输入接口以 A/D 转换器为核心，其功能是将模拟量转换为数字量供 CPU 处理；输出接口的核心电路是 D/A 转换器，其功能是将 CPU 提供的数字量转换为模拟量输出。

无论 A/D 转换器还是 D/A 转换器，如何实现与 CPU 的连接是模拟接口电路的关键。在连接中，数据线如何连接，数据总线宽度与芯片的数据线宽度不一致如何处理，地址端口如何设计，以及数据的锁存与缓冲等都是需要解决的问题。

当然，完整的模拟接口电路除了 A/D 和 D/A 转换器芯片以外，还有传感器、放大器、滤波器、多路转换开关等一系列部件。此外，A/D 和 D/A 转换器芯片的性能也直接影响模拟接

口的具体应用。

习 题 6

6.1 什么是 D/A 转换器？什么是 A/D 转换器？

6.2 在 D/A 转换器的指标中，分辨率与精度有何差异？有一个 6 位 D/A 转换器，它的分辨率是多少？

6.3 试叙述模拟量输入通道的各组成部分及其功能。

6.4 对于一个量程范围为 0～+5 V 的 8 位 D/A 转换器芯片，怎样能输出一个正向的锯齿波？怎样能输出一个+1 V 的电平信号？若输出电平为+2 V、+3 V、+4 V，则又如何？

6.5 一个 10 位的 A/D 转换器与 8 位的 CPU 相连，除了数据线引脚以外，A/D 转换器还有电平启动信号和转换结束信号。试采用程序查询方式完成 A/D 转换器与 CPU 的连接，并给出其具体的电路图和实现程序。

6.6 有一个 8 位的 A/D 转换器，它的分辨率是多少？若 A/D 转换器的转换速度是 2 μs,则该 A/D 转换器在 1s 内最多可进行多少次转换？

6.7 如果 A/D 转换器 0809 要求与 CPU 相连，已知参考电压 V_{REF}=+5 V，若输入到 A/D 转换器的模拟电压为 1.6 V，从通道 0 加入，那么经 A/D 转换后得到的结果是多少？试写出相应的控制程序。

第7章　总 线 接 口

总线是一组互连信号线的集合，是一种在各模块间传送信息的公共通道。微型计算机系统大都采用总线结构，利用总线这一组公共信号线作为计算机各部件之间的通信线，实现芯片内部、印刷电路板各部件之间、机箱内各插件板之间、主机与外部设备之间或系统与系统之间的连接与通信。

总线是各部件连接的纽带，是计算机通信接口的重要技术。系统设计可面向总线进行，设计者只需要根据总线的规则去设计，将各部件按照总线接口的标准与总线连接而无需单独设计连线，因而简化了系统软件和硬件的设计，使系统易于扩充与升级。

7.1　总线概述

7.1.1　总线的分类

总线有不同的分类方法。根据总线所处的物理位置不同可将总线分为以下4种：片内总线、元器件级总线、系统总线和外总线，如图7-1所示。

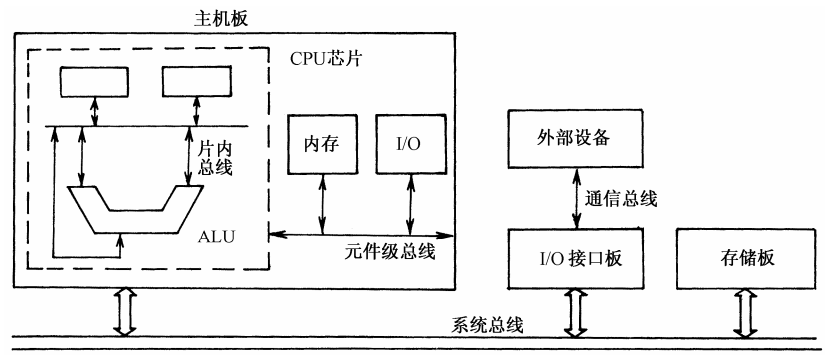


图7-1　微型计算机各级总线示意图

1．片内总线

片内总线是集成电路芯片内部用于连接各功能单元的信息通路。例如，微处理器芯片内部总线用于ALU及各种寄存器等功能单元之间的互相连接。片内总线一般由芯片生产厂家设计，用户不必关心。但随着微电子技术的发展，ASIC技术的大量运用，用户可以借助CAD技术，设计符合自己要求的专用芯片，在这种情况下，用户就必须掌握片内总线技术了。

2．元器件级总线

元器件级总线又称片总线或在板局部总线，是印刷电路板上连接各芯片之间的公共通路。例如，CPU板上CPU芯片与接口芯片之间的连接通路。元器件级总线与芯片引脚的关系密切，难以形成总线标准。

3．系统总线

系统总线又称内总线，是模块式微型计算机机箱内的底板总线，用以连接微型计算机系统的各插件板。例如，多处理器系统中各 CPU 板之间的通信通道就是系统总线。系统总线是微机系统最重要的一种总线，在组建或扩充微机系统时，就要选用恰当的系统总线，并按总线规范设计制作插件板。有的系统总线的性能与某种 CPU 芯片有关，是元件级总线经过重新驱动和扩展而成的，也有不少系统总线并不依赖于具体型号的 CPU，可为多种型号 CPU 及其配套芯片使用。常用的系统总线有 ISA、PCI 等。

4．外总线

外总线也称通信总线，其主要功能是用于微机系统之间、微机系统与仪器仪表或其他外部设备之间的连接。外总线有并行的，也有串行的，其数据传输速率一般比系统总线低。外总线往往并非微机系统专用，一般都是利用电子工业其他领域已有的总线标准，例如，RS-232C 是从 CCITT 远程通信标准中导出的。常用的外总线有 RS-232C、IEEE-488、SCSI 和 IDE 等。

7.1.2 总线信号类型

总线上的信号线通常有几十根至上百根，按其功能大致可分为以下几种：

1．地址总线

地址总线均为单向、三态总线，它是微机系统用于传送地址的信号线。地址总线的数目决定了直接寻址的范围。

2．数据总线

数据总线一般为双向、三态总线，用于传送数据和代码。

3．控制总线

控制总线是传送控制信号的总线，用于实现命令和状态的传送，包括读信号、写信号等，中断、DMA 控制信号、系统时钟、复位信号等也都是通过控制总线传送的。控制总线决定了总线功能的强弱和适应性，是一组很重要的信号线。控制总线根据不同的使用条件，其信号线可分为单向或双向，三态或非三态。

4．电源和地线

在总线中，电源线和地线决定了总线使用的电源种类及地线的分布和用法。

5．备用线

备用线是在总线中留给生产厂家和用户自行定义的信号线，其作用是为了功能的扩充和用户特殊技术要求的使用。备用线为总线的使用增加了灵活性。

7.1.3 总线规范

总线标准是人们在把各种不同的模块组成系统时所需要遵守的总线规范，它为各模块互连提供了透明的标准，任一方只需根据总线标准要求实现和完成接口功能，而不必考虑另一方的接口方式。采用总线标准可以为接口的软、硬件设计提供方便。对硬件结构来说，各模块的接口设计相对独立，只要达到功能要求便可，不必要求结构上的一致，而接口软件也可以进行模块化设计。采用总线标准可以简化系统设计和系统结构，提高系统可靠性，便于系统的扩充和更新。

总线形成标准，通常有两种方式：

（1）先有产品后有标准：一般是某一公司在为自己开发微机系统时所使用的一种总线，而其他兼容机厂商按其公布的总线规范开发相配套的产品并投入市场。由于这类产品的广泛使用，其总线标准也被国际工业界广泛支持，有的还被国际标准化组织承认并授予标准代号。PC 总线就属于这一类总线。

（2）先有标准后有产品：其总线标准是由国际权威机构或多家大公司联合组织研究人员根据技术和产品发展的需要进行研发并制定的。在总线标准确定以后，有关厂商推出相应的产品。USB 总线就是典型的例子。

在国际上，从事接纳和主持制定总线标准工作的有 IEEE（美国电气与电子工程师协会）、IEC（国际电工委员会）、ITU（国际电信联盟）和 ANSI（美国国家标准局）组织的专门标准化委员会，这些委员会一方面为适应不同应用水平要求，从事开发和制定总线标准或建议草案，另一方面对现有的由一些公司提出的并为国际工业界广泛支持的通用总线标准进行筛选、研究、修改和评价，给以统一编号，用做对该总线标准的认可。

每种总线都有详细的规范，以便大家共同遵守。总线规范一般包括如下部分：

（1）机械结构规范：规定模块尺寸、总线插头、连接器等规格；

（2）功能结构规范：规定总线接口引脚的定义、传输速率的设定、定时及信号格式和功能；

（3）电气规范：规定信号逻辑电平、负载能力及最大额定值及动态转换时间等。

7.1.4 总线传输周期

总线作为所有模块共同使用的公共通路，为了避免总线冲突，在任何时刻总线上只能进行一组信息交换，即在某一瞬间只能允许一个发送器向总线发送信息，不同的发送器需要在不同的时刻发送信息。当有多个模块都要使用总线进行数据传输时，只能采用分时复用的方式，将总线时间分成若干小段，每段完成模块之间一次完整而可靠的信息交换，这样的一次交换称为一个“传输周期”。

系统总线上的数据传输在主控模块的控制下进行，总线从属模块没有控制总线的能力，它对总线上的信号进行地址译码，接受和执行总线主控模块的命令信号。总线完成一次数据传输一般需要 4 个周期：申请阶段、寻址阶段、传输阶段和结束阶段。

1．申请阶段

需要使用总线的主控模块（CPU 或其他具有控制总线功能的模块）提出申请，由总线仲裁部分确定将下一个传输周期使用权交给哪一个主控模块。若总线上只有一个主控模块，则

不需要这一阶段。

2．寻址阶段

取得总线使用权以后，主控模块就通过总线发出本次传输周期计划访问的从属模块的地址及有关命令，以启动参与本次传输的从属模块。

3．传输阶段

在取得从属模块的响应后，主控模块和从属模块之间的数据就开始进行传输了，数据由源模块发出，经数据总线送至目的模块。

4．结束阶段

主/从模块的有关信息均从总线上撤除，从而让出总线。一般而言，本次总线操作的结束总是伴随着下一总线传输周期的开始。

7.1.5 总线的裁决方式

在一个微型计算机系统中，不仅有主处理器(CPU)，还可能有多个具有总线控制功能的模块或设备。在这样的一个系统中，完全有可能会出现多个主控设备同时申请占用总线的情况。这就要求系统必须具备总线裁决功能，以保证在任一时刻只能有一个主设备占用总线。总线的裁决工作在总线申请阶段完成，其主要功能有：

（1）用来规定在多处理器系统中各个主设备使用总线的优先权；

（2）当多个主设备请求使用总线时，判定请求总线的最高优先权主设备；

（3）当最高优先权主设备取得准许使用权后，控制总线使用权的正确转换，使原来占有总线的主设备交出总线控制权，转让给新的主设备。

总线的裁决工作可能由专用的总线裁决电路完成；也可能是每个主控模块都具有裁决功能，共同来完成裁决工作。经历总线裁决以后，只有获得了总线使用权的模块才能进行数据传送。

从裁决过程来看，总线裁决可分为固定型裁决和可变型裁决两种方式。

（1）固定型裁决：裁决周期与数据传送周期分时进行，只有在数据传送结束后才能进行总线裁决。

（2）可变型裁决：裁决周期和数据传送周期可重叠操作，这样主设备可在任何时候请求数据传送，与当前的状态无关。可变型裁决方式其裁决操作在时间上不影响数据传送操作，适用于高速系统。

根据总线控制器的位置，控制方式可分为集中式裁决方式和分散式裁决方式两种。

（1）集中式裁决是指系统中设有一个总线控制器，负责对系统中的所有主设备进行总线裁决和分配。集中式裁决的主要优点是裁决逻辑集中在一个模块内，这样便于逻辑设计的简化，主要缺点是如果总线控制器发生故障，将会造成系统致命性灾难。

（2）分散式裁决的系统中没有一个集中的裁决逻辑，它的裁决逻辑分布在系统的各个主设备内，由各个主设备自己进行判断，以获知本设备是否具有占用总线的权利。

具体的裁决方式可分为菊花链查询方式、计数器查询方式和独立请求方式。

1．菊花链查询方式

菊花链查询方式的基本思想是让各模块串行连接起来，依次传送总线响应信号，组成一个查询链。显然，在查询链中离总线控制器最近的设备具有最高优先权，可首先得到总线响应信号，如果这个模块有总线请求，则截断响应信号不再下传。否则依次串行传送总线响应信号，直到最高优先级的请求设备截取响应信号为止。

2．计数器查询方式

按计数器查询方式，系统中每个总线主控设备有一个惟一对应的查询号。主控设备的总线请求信号通过同一根请求线送至总线控制器。总线控制器在收到总线请求信号后，启动计数器计数，并发往各主控设备。各主控设备将收到的计数值与本设备的查询号相比较，若符合则发出信号，停止计数器计数。控制器通过计数查询电路即可查询到请求总线使用的主控设备。

3．独立请求方式

采用独立请求方式，每一个主控设备各有一对独立的总线请求线和总线响应线。当某一模块请求总线时，向总线控制器发出请求信号，总线控制器接收总线请求后，对优先模块在其相应的总线响应线上送回总线响应信号，该模块即可占用总线。

在独立请求方式下，各设备对总线可同时提出请求，因此总线的裁决时间最小。而且对设备的裁决服务也比较灵活，可根据需要定义各种优先级方式。由于各设备的请求线是独立的，还可利用总线控制器禁止某一模块的总线请求。

7.1.6 总线数据的传送方式

在总线数据传送阶段，需要有总线数据传送规程，这是总线通信的一个规约，规约包括：指定数据传送的主控设备和从属设备，规定数据传送的类型（如数据传送方向、数据宽度等），规定数据传送期间控制信号的时间关系。

总线数据传送的通信方式基本有 3 类：同步通信、异步通信和半同步通信。

1．同步通信方式

同步总线通信规程利用系统时钟作为各模块工作的时间标准，通信双方严格按时钟规定完成相应的操作。其主要优点是简单，每个模块什么时候发送或接收信息都由统一的时钟信号控制。但作为主控信号的系统时钟不适合较长距离的传输，否则会因为传输线效应使整个系统工作不正常。同时，同步总线在处理相对低速的从设备时也存在问题，由于数据传送必须在限定的时钟周期内完成，它不能适应那些存取时间较长的设备。为了满足这些低速设备的需要，不得不放慢时钟频率，而使高速设备迁就低速设备，结果降低了整个系统的性能。

2．异步通信方式

为了满足对高速设备能具有高速操作而对低速设备具有低速操作的要求，可采用异步通信总线。异步通信允许总线上的各模块有各自的时钟，这样在模块间进行通信时就不需要公

共的时钟了。但要实现不同速度模块间的配合，就必须增加应答信号线，应答信号常用请求（Request）和响应（Acknowledge）来表示。异步通信总线正是利用这两个应答信号的联络来保证可靠通信的。异步通信的优点是能在同一系统中兼容不同速度的设备。缺点是由于两个应答信号的相互联系，使得应答信号在一个数据传送期间内在总线上至少经过两次传输，降低了传送速率。

3．半同步通信方式

由于异步通信总线的传输延迟严重限制了数据的传送速率，而同步通信总线又不能满足不同速度设备的传送要求。因此有了半同步通信总线，这是一种结合同步总线和异步总线优点的总线方式。半同步通信总线能像同步通信总线一样采用统一的系统时钟同步双方的通信。同时，为了能像异步通信总线一样，允许不同速度的模块和谐地一起工作，而引入了一条“等待（Wait）”响应线。当慢速设备在规定时间内未完成操作时，该信号线有效，这时系统插入等待周期，以此来同步双方的通信。

7.2 PC总线

7.2.1 PC 系列总线及其发展

IBM PC系列总线是指IBM公司的PC系列微型计算机及其兼容机所用的系统总线。IBM PC系列微机采用开放式结构，即在底板上设置一些标准扩展插槽，将各种符合插槽运行的适配器板插入插槽即可扩充PC机的功能。这些插槽即系统总线，可用于各插件板之间的连接。

1．PC 系列总线发展概况

IBM PC 系列微型计算机总线，按其推出的先后顺序可分为 PC 总线、ISA 总线、MCA 总线、EISA 总线、VL 总线和 PCI 总线等。图 7-2 所示为 PC 系列总线的发展概况。

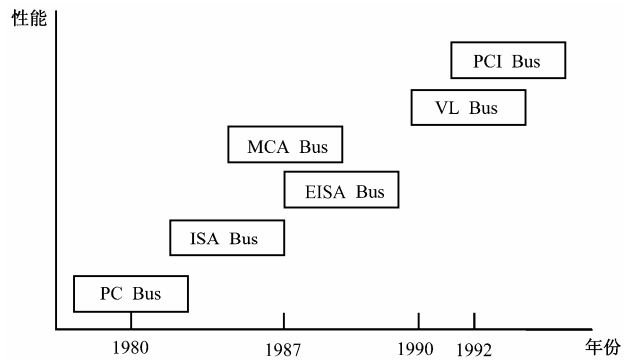


图 7-2 IBM PC 系列微机总线的发展概况

最早的系统总线是从 IBM PC 机时所用的 8 位扩展总线开始的。在 20 世纪 80 年代初期，IBM 公司吸取以前微型计算机的经验，在其推出的 IBM PC 机上成功地使用了一套总线结构，即在主机板上留有 6～8 个扩展插槽。利用这些插槽可以方便地插入各种功能的适配器板，如串行通信板、软盘驱动器板等，以扩充微机功能，从而为微机的扩充、组装和维护带来

了极大的方便。这种扩展插槽便是 PC 总线，它具有 8 位数据总线，传输速率也不是很高。

20 世纪 80 年代中期，IBM 公司在针对 16 位的 80286 CPU 设计 PC/AT 机时深感原 8 位 PC 总线的不足，便在原扩展槽的后面增加了一段，使其延伸成一个具有 36 条引脚的插槽，将数据线扩展到了 16 位。这种规格后来得到工业界的认可，并定义为工业总线标准（Industry Standard Architecture），这种标准的总线被称为 ISA 总线，又称 AT 总线。

IBM 公司推出 80386 CPU 后使得系统内部总线结构有了很大的变化，数据总线从 16 位增加到 32 位，CPU 的处理能力大大提高，32 位数据处理方式和 20MHz 的 CPU 工作频率使原来的 ISA 总线出现了瓶颈现象，通过总线与存储器、显示器、I/O 设备等传输数据的速度显得很慢。为了解决低性能总线与高性能 CPU 之间的矛盾，IBM 公司在 1987 年设计 PS/2 系列时采用了微通道技术，创造了一个全新的与 ISA 标准完全不同的系统总线，即 MCA（Micro Channel Architecture）总线标准。该标准定义了系统总线上的数据宽度为 32 位，并提供了突发方式的 DMA 传输，使得数据传输率为 ISA 总线的 4 倍。为了保持 IBM 公司在微型机领域的领导地位，该公司没有将这一标准公诸于世，以求垄断市场，这使得解决瓶颈问题的手段成为 IBM 公司独有，但也为 MCA 总线的推广设置了障碍。

随着 Intel 公司 80486 微处理器的推出，对于解决“瓶颈”问题的需求日益增加，为了突破 IBM 公司对 MAC 总线技术的封锁，1988 年以 Compaq 为代表的包括 HP、AST、EPSON、NEC、Olivetti、Tandy、Wyse 以及 ZDS 等 9 家计算机公司推出了一个新的总线标准，称为 EISA（Extended ISA）。EISA 不仅具有同 MCA 相似的功能，而且还可与传统的 ISA 100%兼容，这就意味着用户对 EISA 的投资不仅可享受 ISA 的资源，还能享受 EISA 的高性能资源。由于 EISA 的公开性，使其在应用领域得到了充分发展。

以后随着微处理器速度的提高、软件技术的发展，使得原有总线仍不能充分利用 CPU 的强大处理能力，仍然跟不上软件和 CPU 的发展速度。在系统运行的大部分时间内，CPU 都处于等待状态。特别是 CPU 在日益强大的处理能力和存储容量的支持和激励下，操作系统和应用程序变得越来越复杂，而显示卡和硬盘控制器等都是在 8 位或 16 位系统的 I/O 总线上，相对速度极高的 CPU 而言，传输数据的速度要低得多，从而影响了系统的整体工作效率。因此，为提高系统的整体性能，对显示、硬盘等子系统提出了越来越高的要求。要提高系统总线速度，关键是使其与微处理器同速，即将快速外设直接挂至 CPU 的局部总线上并以 CPU 的速度运行。于是，1992 年 VESA（Video Electronics Standards Association）视频电子标准协会与 60 余家附件卡制造商联合推出了一个基于 80486 CPU 的 32 位局部总线标准，简称 VL（VESA Local Bus）总线。VL 总线支持 16～66 MHz 的时钟频率，数据宽度为 32 位，可扩展到 64 位。这种总线可以使 I/O 速度随 CPU 速度的不断加快而加快，但由于 VL 总线是在 CPU 总线基础上扩展而成的，与 CPU 类型相关，因此开放性差。

在此期间，Intel 公司也推出了自己的局部总线 PCI（Peripheral Component Interconnect），它克服了 VL 总线的缺点，是当今广泛使用的总线标准。PCI 总线也定义了 32 位数据线，可扩展为 64 位，体积比原来的 ISA 总线还小，使用 33 MHz 时钟频率，可同时支持多组外围设备。

随着高档微型计算机的发展，且为了与早期微机系统兼容，如今微机系统结构大多采用不同总线构成的多总线结构，在其主机板上往往留有不同的总线插槽。常见的有 ISA-EISA 组合、ISA-VL 组合、ISA-PCI 组合、EISA-PCI 组合等。

2．PC 系列总线技术比较

PC 系列总线的发展实质上是 PC 系列微机发展的缩影。作为一种总线标准，是否能够得到认可，是否能够在微机产品上得到广泛地应用，取决于该总线标准是否符合当时的微机发展要求。要衡量一个总线标准的技术指标，除数据位的宽度、支持的数据传输速率以外，其规范性、可扩展性也是很重要的技术指标。表 7-1 为 PC 系列总线标准的性能比较。

表 7-1　PC 系列总线性能比较

总线	宽度	传输速率（Mb/s）	突发方式	自动配置	规范性	复杂性	可扩展性
PC	8	5	无	无	差	简单	较好
ISA	16	8	无	无	差	简单	较好
MCA	32	33	有限	有	好	复杂	较好
EISA	32	40	有限	有	差	复杂	差
VL	32	132	有限	无	差	简单	差
PCI	32/64	132/528	无限	有	较好	复杂	好

7.2.2　ISA 总线

ISA 总线是指 80286 CPU 的 IBM PC/AT 机中所使用的总线，又称 PC-AT 总线，它是在 8 位的 PC 总线基础上扩展而成的 16 位总线体系结构。实际上，在采用 80386 CPU 以上的 32 位微机系统中仍大量采用 ISA 总线作为外围总线。ISA 总线和 PC 总线都是一种原始总线，实际上是采用将微处理器芯片的总线缓冲后直接映射到系统总线上而形成的。

1．主要功能

ISA 总线是在原 PC 总线基础上扩展而成的，在不改变原设计的前提下增加了数条信号线，将数据线扩到 16 位、地址线扩到 24 位，可寻址空间为 16MB。

ISA 总线是一种多主控总线，即除主 CPU 以外，DMA 控制器、刷新控制器、带处理器的智能接口控制卡都可以成为 ISA 的主控设备。

ISA 总线支持 8 种类型的总线周期：存储器读、存储器写、I/O 读、I/O 写、中断请求与响应、DMA 传输、刷新和仲裁周期。

2．总线引脚定义

ISA 总线是在 PC 总线的基础上扩展一个 36 线插槽形成的。ISA 总线由同一轴线的基本插槽和扩展插槽两段组成。基本插槽有 62 条信号线，兼容 PC 总线；扩展插槽有 36 条信号线，为 ISA 新增加的信号。ISA 总线信号如 7-3 所示。

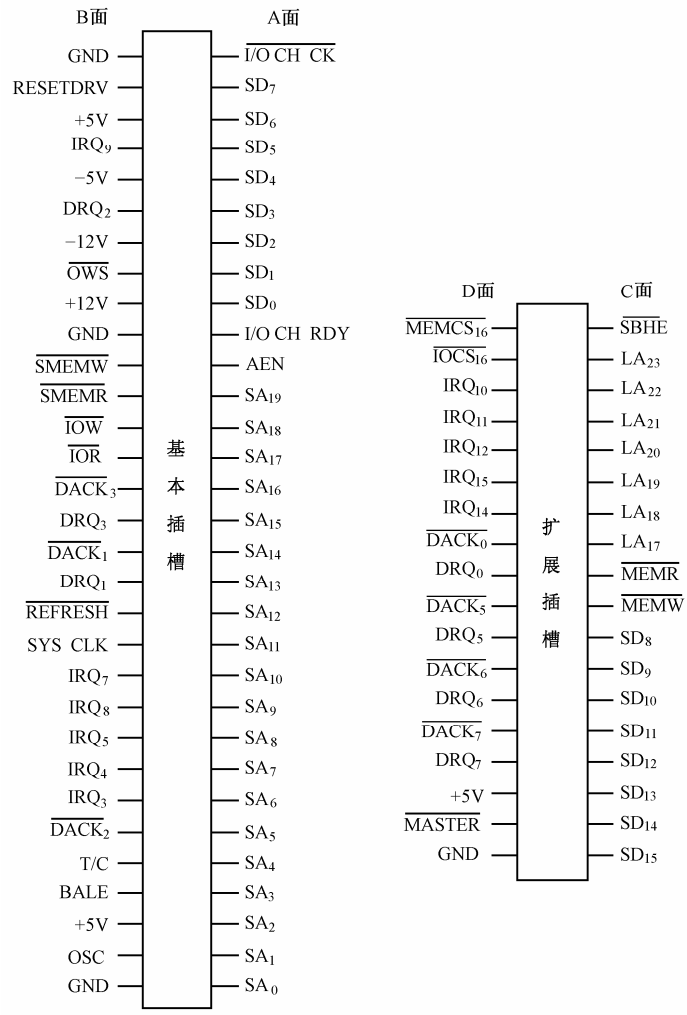


图 7-3　ISA 总线信号

（1）数据总线。

SD_7～SD_0：**数据总线低字节信号，8 位，双向，为微处理器、存储器和 I/O 端口提供了数据信息传输通道。其中，D_0 为最低有效位，D_7 为最高有效位，每次只能传输一个字节。**

SD_{15}～SD_8：**数据总线高 8 位信号。**

$\overline{SBHE}$：**数据总线高字节有效信号，用于表示当前数据总线传送的是高位字节** SD_{15}～SD_8。

$\overline{MEMCS_{16}}$：**存储器 16 位芯片选择信号，用来通知主机板，当前的数据传送是一个等待状态的 16 位存储器周期。**

$\overline{IOCS_{16}}$：**I/O 16 位芯片选择信号，用来通知主机板，当前的数据传送是一个等待状态的 16 位 I/O 周期。**

（2）地址总线。

SA_{19}～SA_0：**20 根地址总线，输出信号，用来寻址与系统总线相连接的存储器和 I/O 端口。**

LA_{23}～LA_{17}：**非锁定地址总线，也用于系统中存储器及 I/O 设备的寻址，这些信号的**

加入给系统提供多达 16MB 的寻址能力。

BALE：地址锁存允许信号，用于在主板上锁存从处理器来的有效地址。

AEN：地址允许信号，用来使主机板上的微处理器进入保持状态，以便进行 DMA 传送。

（3）控制总线。

$\overline{\text{MEMR}}$ ：存储器读命令，表示存储器将数据送上数据总线。

$\overline{\text{MEMW}}$ ：存储器写命令，表示存储器将当前数据总线上的数据存入。

$\overline{\text{IOR}}$ ：I/O 读命令，表示 I/O 设备将数据送上数据总线。

$\overline{\text{IOW}}$ ：I/O 写命令，表示 I/O 设备将当前数据总线上的数据读入。

$\overline{\text{SMEMR}}$ 、$\overline{\text{SMEMW}}$ ：存储器读、存储器写信号，指示存储器将数据送上数据总线或把总线上的数据存入存储器。

I/O CH RDY：通道准备好信号，可用于控制是否延长 I/O 或存储器周期。

$\overline{\text{I/O CH CK}}$ ：I/O 通道校验信号。

IRQ_3～IRQ_7、IRQ_9～IRQ_{12}、IRQ_{14}、IRQ_{15}：中断请求信号，用来对微处理器产生中断请求。

DRQ_0～DRQ_3，DRQ_5～DRQ_7：DMA 请求信号。

$\overline{DACK_0}$ ～$\overline{DACK_3}$ 、$\overline{DACK_5}$ ～$\overline{DACK_7}$ ：DMA 响应信号。

T/C：计数结束信号，表示 DMA 通道字计数器满。

$\overline{\text{REFRESH}}$ ：刷新信号，指示一个存储器刷新周期。

$\overline{\text{MASTER}}$ ：主设备信号，用于对系统进行控制。

（4）时钟、定时与电源。

OSC：振荡器信号。

SYS CLK：系统时钟信号。

RESETDRV：复位驱动信号。

$\overline{\text{OWS}}$ ：零等待状态信号。

＋5V、-5V、＋12V、-12V、GND：电源及地线。

7.2.3 PCI 总线

PCI 总线是 Intel 公司推出的一种局部总线，定义了 32 位数据线，且可扩展到 64 位。它体积小，支持无限突发操作，使用 33MHz 和 66 MHz 时钟频率，最大传输速率为 132～528Mb/s，支持并发工作方式。

1．PCI 体系结构特点

PCI 总线结构如图 7-4 所示，通过 PCI 桥路可将一些高速外设（如图形加速器、硬盘控制器、LAN 控制卡等）挂到 CPU 总线上，可支持高速 I/O 与 CPU 并行工作。PCI 桥路实现驱动 PCI 总线所需的全部控制，例如在与 CPU 总线接口方面引入了先进先出缓冲器，使 PCI 总线上的部件可以与 CPU 并行工作。在 PCI 总线上有一种特殊设备，即标准总线桥路，它将 PCI 总线转换为 ISA、EISA 等标准总线，以便与这些总线设备相连。

PCI 总线不依赖于某一具体的微处理器结构，因为它不与微处理器直接相连，而是通过 PCI 桥路与其相连。这样，对于不同的微处理器品种，只要更换相应的桥路即可。

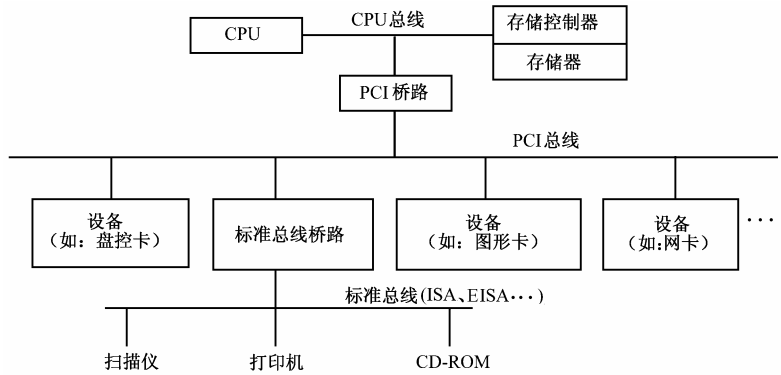


图 7-4　PCI 总线结构

PCI 总线支持无限突发读写方式，读写时后面可跟无数个数据，具有强大的数据突发传输能力。使用突发方式，由于外设与内存的数据传输往往是大块进行的，这意味着可从某一个地址起连续读写大量的数据，这样就能减少无谓的地址作业，提高传输效率。

PCI 数据线和地址线采用了多路复用结构，减少了针脚数。一般情况下，32 位字长仅做目标设备的接口只需 47 条引脚，作为总线控制者的设备接口再加 2 条引脚。并可有选择地增加信号线以扩展功能，如 64 位字长的接口卡需加 39 条引脚，资源锁定加 1 条引脚。

PCI 支持即插即用功能，能实现自动配置。在 PCI 器件上包含有配置寄存器，上面带有配置所需的器件信息。使外设适配器在和系统连接时能自动进行配置，无须人工干预。

2．PCI 总线信号定义

PCI 总线信号如图 7-5 所示，左边为必要信号，右边为可选信号。

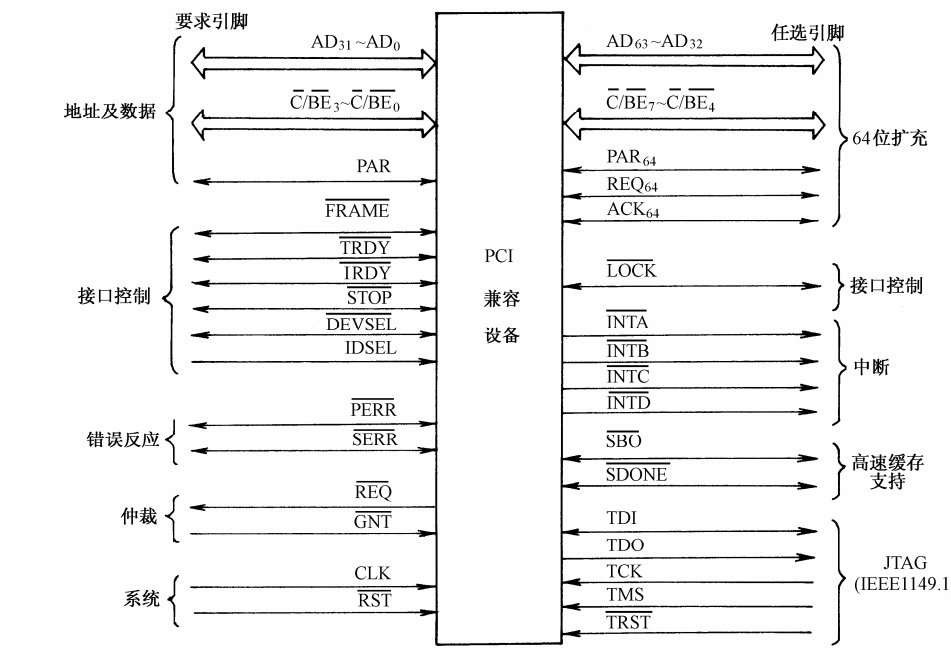


图 7-5　PCI 总线信号

按功能，引脚可分为以下几类：

（1）系统信号。

CLK：系统时钟信号线，该信号的频率为 PCI 总线的工作频率。除 $\overline{\text{RST}}$、$\overline{\text{INTA}}$、$\overline{\text{INTB}}$、$\overline{\text{INTC}}$ 和 $\overline{\text{INTD}}$ 信号外，其他 PCI 信号，都与 CLK 上升沿同步。

$\overline{\text{RST}}$：复位信号线。

（2）地址和数据信号。

AD_{31}～AD_0：地址、数据信号复用线。$\overline{\text{FRAME}}$ 有效，表示地址传送阶段开始，接下来为数据传送阶段（$\overline{\text{IRDY}}$ 和 $\overline{\text{TRDY}}$ 同时有效）。

$\overline{C}/\overline{BE_3}$～$\overline{C}/\overline{BE_0}$：总线指令和字节允许信号的复用线。在地址传送阶段是 4 位编码的总线指令。在数据传送阶段，用做字节允许标志，以决定数据线上哪些字节数据为有效数据。

PAR：为 AD 和 $\overline{C}/\overline{BE}$ 所指示的有效数据的校验位。

（3）接口控制信号。

$\overline{\text{FRAME}}$：周期帧信号，由当前总线控制者产生，表示一个总线传输的开始和延续。

$\overline{\text{IRDY}}$：主设备就绪（Initiator Ready），表明数据传输的启动者（主控者）已经准备好，等待完成当前的数据节拍。

$\overline{\text{TRDY}}$：目标设备就绪（Target Ready），说明数据传输的目标设备已经准备好，等待完成当前的数据节拍。

$\overline{\text{STOP}}$：停止信号。信号有效表明当前的目标设备要求总线控制者停止当前的数据传输。

IDSEL：初始化时的设备选择信号。由当前的总线控制者驱动。用于在配置空间内选择总线上的某个设备。

$\overline{\text{DEVSEL}}$：设备选择信号。每个目标设备在地址传送阶段进行地址译码，若被选中，则使该信号有效，用来向总线控制者报告已有目标设备被选中。

$\overline{\text{LOCK}}$：总线锁定信号。用来实现多处理器、多 PCI 总线主设备系统中存储数据的保护。

（4）仲裁信号（只对总线控制者有用）。

$\overline{\text{REQ}}$：总线请求信号，用来向总线仲裁器申请总线的控制权。

$\overline{\text{GNT}}$：总线响应信号，由总线仲裁器发出，通知申请总线控制权的设备已获得总线的控制权。

（5）错误反馈信号。

$\overline{\text{PERR}}$：奇/偶校验错误，该引脚用于反馈在除特殊周期以外的其他传输过程中的数据奇/偶校验错误。

$\overline{\text{SERR}}$：系统错误，用于反馈地址奇/偶校验错误、特殊周期指令中的数据奇/偶校验错误和将引起重大故障的其他系统错误。

（6）中断请求信号。

$\overline{\text{INTA}}$、$\overline{\text{INTB}}$、$\overline{\text{INTC}}$、$\overline{\text{INTD}}$：中断请求信号。一个 PCI 设备接口卡可有多个功能，可使用多个中断请求信号，最多为 4 个。单一功能的 PCI 设备接口卡只能使用一根中断请求线 $\overline{\text{INTA}}$。

（7）高速缓存支持。

$\overline{\text{SBO}}$：侦听回写信号（Snoop Backoff）。为了保证 Cache 和主存的内容一致，需采用 Cache 侦听技术。当侦听命令中 Cache 的一行有被修改过的数据时，该信号有效，直到此行数据被

写回到主存中为止。

$\overline{\text{SDONE}}$：侦听结束信号（Snoop Done）。

（8）64 位扩展信号线。

AD_{63}～AD_{32}：高 32 位的地址、数据复用线。

$\overline{C}/\overline{BE}_7$～$\overline{C}/\overline{BE}_4$：总线指令和高 32 位字节使能复用线。

PAR_{64}：高 32 位奇/偶校验位。

REQ_{64}：64 位数据传输申请信号。PCI 主设备在发送 $\overline{\text{FRAME}}$ 信号的同时置位该信号，表明本次申请的数据传输使用 64 位字长。

ACK_{64}：64 位数据传输许可信号。PCI 目标设备在发送 $\overline{\text{DEVSEL}}$ 信号的同时发送该信号，表明它许可 64 位数据传输，否则仍为 32 位。

7.3 RS-232总线接口

EIA-RS-232C 标准是由美国电子工业协会(EIA) 制定，于 1969 年公布的通信协议。在制定之初，EIA-RS-232C 就用于远程通信接口，即为远程通信连接数据终端设备 DTE 与数据通信设备 DCE 而制定。目前已广泛应用于计算机与外设之间的终端连接。由于该标准推出较早，并且对通信接口的有关问题如信号线的功能、电气特性等都有明确的规定，于是一般通信接口与设备制造厂商都生产与 RS-232C 兼容的设备，使得 RS-232C 成为计算机远程通信中被广泛采用的一种通信接口标准。

7.3.1 连接器及接口信号

EIA-RS-232C 标准只规定了采用一对物理连接器，对连接器本身的物理特性并没有定义，因而出现了各种的连接器，其引脚定义也各不相同。目前常用的连接器有 25 针的 DB-25 和 9 针的 DB-9 两种。图 7-6 和图 7-7 分别出示了这两种连接器的分配图。

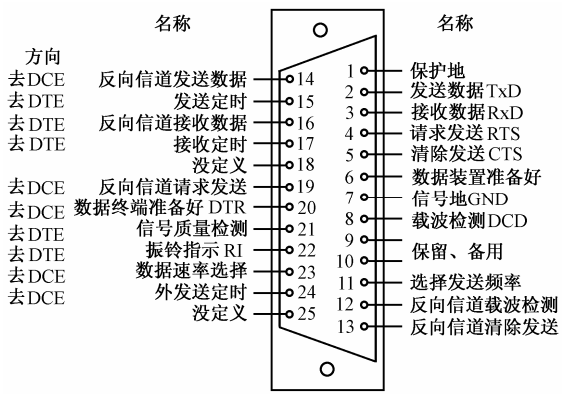


图 7-6 EIA-RS-232C 25 针连接器引脚分配图

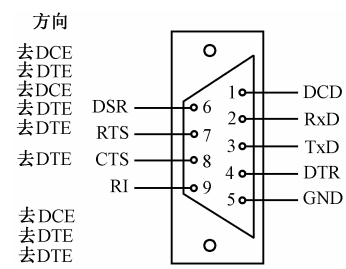


图 7-7 EIA-RS-232C 9 针连接器引脚分配图

早期的 PC 机都采用 25 针连接器，虽然 EIA-RS-232C 定义了 25 脚信号标准，但实际进行异步串行通信时，只需要 9 个电平信号，即 2 个数据信号、6 个控制信号和 1 个信号地。由于早期 PC 机还支持 20mA 电流环接口，另需 4 个电流信号，故采用 25 针连接器。以后的

PC 机串行口取消了电流环接口，故现常采用 9 针连接器。

EIA-RS-232C 接口中实际包括有两条信道：主信道和辅助信道。辅助信道的速率要比主信道低得多，可以在连接的两设备之间传送一些辅助的控制信息，但很少使用。即使是主信道，也不是所有的线都一定要使用的，最常用的是 9 条线，即 DB-9 连接器定义的信号线。下面介绍这 9 条信号线。

1．数据线

TxD（Transmitted Data）：发送数据，通过 TxD 信号线，DTE 将串行数据发送到 DCE。在不发送数据时，通常引脚保持逻辑“1”。

RxD（Received Data）：接收数据，通过 RxD 信号线，DTE 接收 DCE 发送的串行数据。

2．地线

GND：信号地，该引脚为所有的电路提供参考电位。

3．联络控制信号线

DSR（Data Set Ready）：数据装置准备好，该信号有效表示 DCE 可以使用，DCE 已与通信信道相连接。

DTR（Data Terminal Ready）：数据终端准备好，该信号有效表示 DTE 可以使用，DTE 准备发送数据至 DCE。在使用过程中，DSR、DTR 仅分别表示 DCE、DTE 设备本身是否准备好，并不说明通信链路可以开始通信。DTR 必须先接通，然后 DSR 才能变为接通状态。而能否开始通信由下面的控制信号决定。

RI（Ringing）：振铃指示信号，在 Modem 收到交换台的振铃呼叫信号时，使该信号有效，通知 DTE 已被呼叫。

DCD（Data Carrier Detection）：数据载波检测信号，表示 DCE 已接收到满足要求的载波信号，已接通通信链路，告知 DTE 准备接收数据。

RTS（Request To Send）：请求发送信号，表示 DTE 请求 DCE 发送数据。当 DTE 欲发送数据时，使该信号有效，向 DCE 请求发送，控制 DCE 进入发送状态。

CTS（Clear To Send）：清除发送信号，表示 DCE 准备好接收 DTE 发送来的数据，是对 RTS 的响应信号。当 DCE 已准备好接收 DTE 传来的数据、并向前发送时，该信号有效，通知 DTE 可以通过发送数据线 TxD 发送数据。

7.3.2 逻辑电平

EIA-RS-232C 标准对信号逻辑电平的规定如表 7-2 所示。对于数据信号：逻辑“1”（传号）的电平为-15～-3V，逻辑“0”（空号）的电平为+3～+15V；对于联络控制信号：接通状态（ON）即信号有效（逻辑“1”）的电平为+3～+15V，断开状态（OFF）即信号无效（逻辑“0”）的电平为-15～-3V。若传送的电平介于-3～+3V 之间或电平绝对值大于 15V 都是没有意义的。因此，在实际应用中，工作电压应保持在±(5～15)V，通常选择±12V。

表 7-2　EIA-RS-232C 信号的逻辑电平

	数据信号	联络控制信号
逻辑 1	-15～-3V	+3～+15V
逻辑 0	+3～+15V	-15～-3V

显然，EIA-RS-232C 电平与 TTL 电平是不兼容的。相互连接时需电平转换。在实际使用时电平转换芯片的电源几乎都采用±12V，因此信号一般约为±10V。目前已有专门的电平转换器件，可用来进行 TTL 电平和 EIA-RS-232C 电平之间的转换。图 7-8 中给出的 MC1488 和 MC1489 就是常用的电平转换器件。

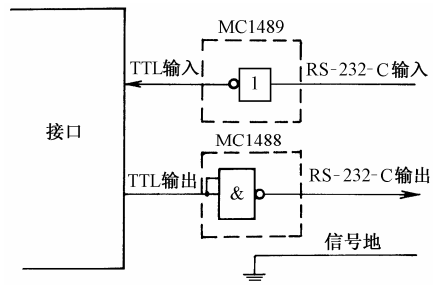


图 7-8　TTL 和 EIA-RS-232C 之间的电平转换

从图中可以看出，MC1488 将计算机或设备送出的 TTL 电平转换为 EIA-RS-232C 电平发送出去。在接收端再用 MC1489 将 EIA-RS-232C 电平还原成 TTL 电平连接计算机或设备。这样一来，在其传输线上传输的则是 RS-232C 电平，其信号电平可高达+10V，比 TTL 电平有更强的抗干扰性能。但即使使用了这样高的电平进行传输，RS-232C 所能直接连接的最大距离也仅为 100 英尺（约 30 米），通信速率仍低于 20Kb/s。

7.3.3　RS-232C 接口的连接方式

RS-232C 接口信号线传输距离短，一般小于 15m。当进行 15m 以上的远距离通信时，一般要加调制解调器（Modem）。进行远距离通信和近距离通信所需连接的 RS-232C 接口信号线不一样，进行近距离连接时，只需要 3 根信号线即可进行双向通信，而远距离通信需要加调制解调器，故需要使用较多的信号线。

1．远距离通信的连接

若在通信双方的调制解调器之间采用专用电话线进行通信，则需要使用如图 7-9 所示的信号线进行联络与控制。若双方调制解调器之间采用普通电话交换线进行通信，则还需要加接 RI 和 DTR 两个信号线进行联络，如图 7-10 所示。

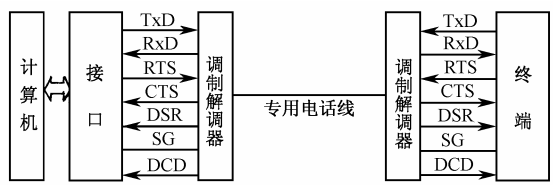


图 7-9　采用 Modem 和专用电话线通信时信号线的连接

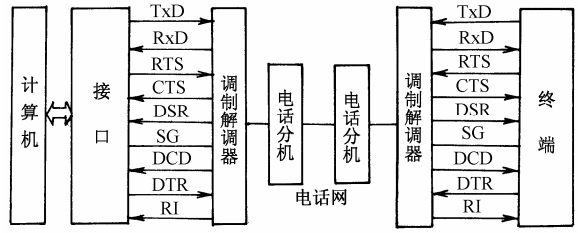


图 7-10　采用 Modem 与电话网通信时信号线的连接

2．近距离通信的连接

近距离通信的连接不需要调制解调器，只需要少量的几根信号线。常用的连接方式有以下 3 种：

（1）简化连接方式。通信双方直接连接，最简单的方式为三线连接法，如图 7-11 所示。此时连接双方的 DSR、CTS 已设置为接通状态。这种连接方式可以实现全双工通信。

（2）反馈连接方式。在计算机与外部设备的通信中，可采用反馈连接方式，通信一方的 DTR 与 DSR 相连。图 7-12 为计算机与打印机采用反馈连接方式的示意图。

（3）交叉连接方式。交叉连接方式适用于两台同一型号的计算机之间的通信，如图 7-13 所示。

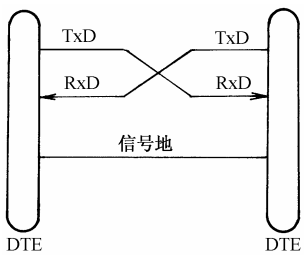


图 7-11　简化连接方式

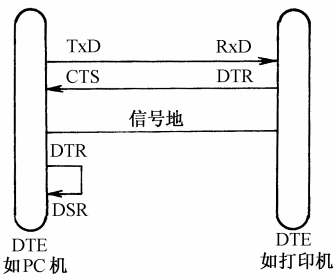


图 7-12　反馈连接方式

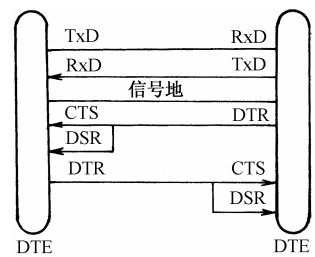


图 7-13　交叉连接方式

7.4　USB总线接口

计算机外部设备的日益丰富，扩展了计算机的应用范围，同时也给不同外设与主机间的连接带来了困难。打印机、扫描仪、游戏杆、数码相机层出不穷，串口、并口、游戏口、PS/2 口各式各样。连接的设备不同，传统的连接方式也不同，需要的跳线、设置也不同。通用串行接口 USB 总线就是在外部设备日益丰富，高速实时数据传输迫切需要的形势下推出的。

7.4.1　USB 概述

USB 是“Universal Serial Bus”的缩写，是计算机连接外围设备（如键盘、鼠标、打印机等）的输入/输出接口标准。1994 年，Intel、Compaq、Digital、IBM、Microsoft、NEC 等 7 家世界著名计算机公司和通信公司成立了 USB 论坛，经历了近两年的讨论才形成统一意见，于 1996 年 1 月 15 日正式颁布了“USB1.0 通用串行总线规范”，而把 USB 接口真正设计在主机板上也用了 1 年的时间，1997 年才开始有真正符合 USB 技术标准的外设出现。目前，USB 已成为微型计算机与外设连接普遍采用的标准接口。

1．USB 规范

USB 是一种针对 PC 结构的扩展工业标准，其主要规范是：

（1）数据传输速率有两种。用于连接打印机、扫描仪、交换机等设备的数据传输速率可达 12Mb/s；用于连接键盘、鼠标器、调制解调器等设备的数据传输速率为 1.5Mb/s，系统可以自动识别并设置传输速率。

（2）连接电缆种类有两种。传输速率为 12Mb/s 的采用带屏蔽双扭线；传输速率为 1.5Mb/s 的采用普通无屏蔽双扭线。

（3）USB 连接器为 4 芯插针，2 条用于信号连接，2 条用于电源馈电线路连接。

（4）包括转换器 HUB，最多可连接 127 个外设装置。

（5）连接结点之间的距离可达 5m。

2．USB 的特点

USB 规范公布以后，受到业界和用户的极大关注，PC 机制造商、外围设备生产厂以及大规模集成电路芯片制造厂商纷纷开发 USB 产品，许多新生产的 PC 机都具备 USB 接口，这是因为 USB 具有以下一系列的优点：

（1）USB 具有真正的“即插即用”特性，用户可以很容易地对外设实行安装和拆卸，主机可按外设的增减情况自动配置系统资源,同时用户可以在不关机的情况下进行外设的更换，外设装置驱动程序的安装、删除也实现了自动化。

（2）USB 具有很强的链接能力，最多可以链接 127 个外设到同一系统，这对一般的计算机系统来说是足够的。

（3）连接电缆轻巧，不需要占用 PC 机的槽口，并且 USB 电源能向外设提供 5V 的电源，因此接入的外设就不需要接专门的交流电源，给 USB 用户和厂商带来了方便。

（4）由于 USB 的智能机制都驻留在主机中，使得外设的设计制造过程比较简单，降低了设备制造的成本。

（5）USB 是一种开放性的不具专利版权的理想工业标准，它所制定的任何标准不为某家公司所独有，不存在专利版权问题，这也正是 USB 规范具有强大生命力的原因。

7.4.2 USB 的连接方法

USB 是一个万能插口，在功能上可以取代当前 PC 机上的串口和并口。USB 的连接，除了硬件连接以外，还包括系统的支持。

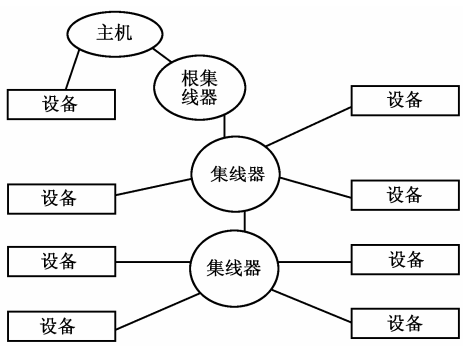


图 7-14 USB 体系

1．USB 系统的组成

一个完整的 USB 体系如图 7-14 所示。

USB 的硬件部分分为两层：最低层是 USB 设备，往上是 USB 主机控制器。

USB 软件部分分为 3 层：首先是 USB 主机控制器驱动程序（Host Controllet Driver），Windows 95 OSR2.1 以上版本及 Windows 98 提供了这个最低层的驱动程序；其次是 USB 设备驱动程序（USB

Device Driver），Windows 95/98 已经内建了一些常用的 USB 设备驱动程序，如 USB 音箱、USB Hub 等，其他没有内建的设备驱动程序可以在需要时装入；最上层是 USB 应用程序（USB Application 或 Client Driver Software），USB 设备需要有相应的应用程序才能发挥作用。

2．USB 的支持环境

USB 的支持环境包括主机硬件的 USB 接口、操作系统软件对 USB 的支持以及外设的 USB 设备。只有满足这 3 个方面的支持，USB 才能正常工作。目前生产的主板一般都采用支持 USB 功能的控制芯片组，主板上也安装有 USB 接口插座；Windows 98 操作系统已全面支持 USB 功能；到目前为止，已经有很多 USB 外设问世，如数字照相机、计算机电话、数字音箱、数字游戏杆、打印机、扫描仪、键盘、鼠标等。有了计算机的 USB 接口，并得到相应操作系统的支持，就可以方便地将 USB 外设连接在系统中了。

3．USB 的硬件结构

USB 采用四线电缆，其中两根是用来传送数据的串行通道，另两根为下接设备提供电源，如图 7-15 所示。

其中，D^+、D^-是串行数据通信线，采用差分传送，支持两种数据传输速率：对于高速且需要高带宽的外设，USB 采用全速 12Mb/s 传送数据；对于低速外设，USB 则采用 1.5Mb/s 的传输速率传送数据。USB 会根据外设的两种传输模式自动地动进行动态转换。VBUS 通常为＋5V 电源，GND 是地线。

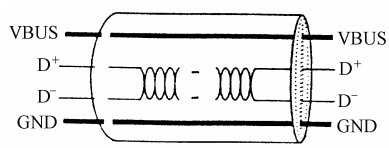


图 7-15　USB 总线的拓朴结构

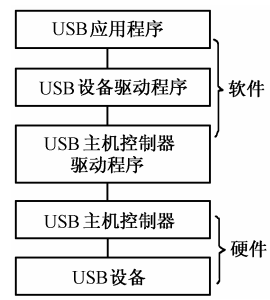


图 7-16　USB 系统结构

4．USB 的连接

USB 系统采用级联星型拓朴结构，该结构由 3 个基本部分组成：主机（Host）、集线器（Hub）和功能设备，其结构如图 7-16 所示。

（1）主机。主机也称根或根 Hub，它设计在主机板上或作为适配器安装在主机上，其中包含主控制器和根集线器，控制着 USB 总线上的数据和信息的流动。一个 USB 系统只有一个根集线器，可以下连集线器或功能设备。

（2）集线器。集线器是 USB 结构中的特定成分，它提供了额外的端口，可以下接功能设备，也可以连接其他的集线器。这种结构允许集线器自行供电或通过总线供电。

（3）功能设备。功能设备通过端口与 USB 总线连接。某些功能设备同时也可以作为 Hub 使用，如 USB 监视器可提供 USB 鼠标和 USB 键盘的端口。

本 章 小 结

本章涉及了两方面的知识：总线基础知识和具体总线介绍。

第一部分总线基础知识介绍了总线的类型、总线基本信号线、总线规范的概念以及运用总线进行信息交换所涉及的总线周期、总线裁决、总线传送方式等概念，这些知识是学习具体总线的基础。

第二部分介绍了目前常用的若干总线标准，其中包括 PC 系列的系统总线和两种串行外总线 RS-232C 和 USB。针对每一种总线，均介绍了其引脚功能定义及连接要求。对于具体总线标准的学习，既可以了解总线的具体使用方法，又可以加深对总线概念的理解。

习 题 7

7.1 为什么需要定义总线规范，总线规范通常是怎样产生的？

7.2 总线裁决方式有哪几种，各有什么特点？

7.3 与 ISA 总线相比较，PCI 总线有什么特点？

7.4 了解你使用的微型计算机，其主机板支持哪几种系统总线。

7.5 RS-232C 总线常用的连接方式有几种，各自适合怎样的连接要求？

7.6 USB 总线得到推广应用的原因是什么？

第 8 章　常用外设接口

外围设备的功能是在计算机和其他机器之间，特别是在计算机与用户之间提供联系。外围设备是通过输入/输出接口电路与主机相连的。由于外围设备的多样性，不同的设备与不同的接口电路相配合使用。常用的外围设备主要是用于人机交互的设备，如键盘、鼠标器、显示器和打印机等，相应的输入/输出接口电路也是很常用的，了解与掌握接口电路的组成原理以及它们的硬件、软件的设计思想是本课程的基本要求。

8.1　LED显示器及接口

七段 LED 显示器（Light Emitting Display）是一个很实用，同时也是很廉价的数字显示装置。LED 数码管由七段发光二极管组成，主要用于显示十六进制数字，从单板微型机、袖珍计算机到微型机控制系统及数字化仪器中都用 LED 数码管做输出显示器。

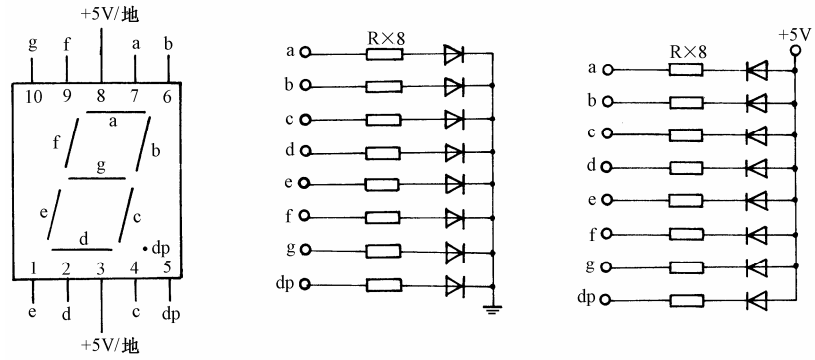
8.1.1　LED 显示器的工作原理

发光二极管是一种将电能转变成光能的半导体器件，每个 LED 的正向压降是恒定的，典型值为 1.6V 或 2.4V。正向导通时发光，发光时流过 2～20mA 的电流。一般说来，外加正向电压愈高，电流愈大，发光愈强，但如果电压太高，电流太大，将会烧坏发光二极管。因此，在使用时必须串入一限流电阻。

七段 LED 显示器由 7 个发光段构成，每段均是一个 LED 二极管。如图 8-1（a）所示，这 7 个发光段分别称为 a、b、c、d、e、f 和 g，通过控制不同段的点亮和熄灭，可显示十六进制数字 0～9 和 A、B、C、D、E、F，也能显示 H、L、P 等字符。多数七段 LED 显示器中实际有 8 个发光二极管，除 7 个构成 7 笔字形外，另外还有一个小数点 dp 位段，用来显示小数。有人也把这种显示器称做八段 LED 显示器。

LED 显示器有共阴极和共阳极两种结构，分别如图 8-1（b）和 8-1（c）所示。在共阳极结构中，各 LED 二极管的阳极被连在一起，使用时要将它与+5V 电源相连，而把各段的阴极连到器件的相应引脚上。当要点亮某一段时，只要将相应的引脚（阴极）接低电平即可。例如，要显示数字 5，只要将 a、f、g、c、d 段接低电平，其余段接高电平即可。对于共阴极结构的 LED 显示器，阴极连在一起后接地，各阳极段接到器件的引脚上，要想点亮某一段时，只要将相应引脚接高电平即可。

对于共阴极和共阳极两种不同的接法，为了显示同一个字符，对应的显示段码是不同的。表 8-1 列出了这两种不同接法下的字形段码关系表。表中的段码是以八段和 8 位字节数的对应关系为前提得到的，比如为了显示“3”，对共阴极应该使 D_7～D_0=01001111，即 4FH，对共阳极应该使 D_7～D_0=10110000，即 B0H。这就是表中对应于显示字符“3”的两个段码，其余依次类推。



(a) 结构和引脚　　(b) 共阴极接法　　(c) 共阳极接法

图 8-1　七段 LED 显示器

表 8-1　LED 字形表

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（d）	5EH	A1H
2	5BH	A4H	E	79H	86H
3	4FH	B0H	F	71H	8EH
4	66H	99H	·	80H	7EH
5	6DH	92H	P	73H	82H
6	7DH	82H	U	3EH	C1H
7	07H	F8H	T	31H	CEH
8	7FH	80H	Y	6EH	91H
9	6FH	90H	8.	FFH	00H
A	77H	88H	“灭”	00H	FFH
B（b）	7CH	83H	⋮	⋮	⋮

8.1.2　LED 显示接口

在一个 LED 显示器上显示一位十进制或十六进制数，都需要用 4 位二进制编码表示，并将这 4 位二进制数转换成 LED 的七段显示代码。转换的方法有两种：一种方法是采用专用的 LED 七段译码器，实现硬件译码；另一种方法是软件译码，通过程序直接向接口电路提供能显示的七段码。

在实际应用中，LED 显示器的使用往往不只一个。相对而言，单个 LED 显示器，其接口电路简单，对于多位的 LED 显示器，考虑到多位 LED 显示器同时显示，接口电路相对复杂。在实现上，可以将独立的一位 LED 显示器组合在一起构成多位显示，但更多的是采用程序控制动态扫描的方法来控制显示工作。

无论采用什么显示方法，由于 LED 显示器的一个段发光需在该段中流过平均约为 10～20mA 的电流，因此电路一般都需增加驱动器电路。针对硬件译码的电路，往往不需另加驱动器电路，这是因为一般在七段码译码器电路中已经包含了驱动器电路。

1．采用硬件译码的一位 LED 显示接口

利用专用接口芯片可驱动单个七段 LED 显示器，7447 就是这样一种接口芯片。7447 是

一个 BCD 码到七段码的译码电路，并具有驱动电路，可直接用于共阳极的 LED 显示。用 7447 驱动单个七段 LED 显示器的电路如图 8-2 所示，这种电路只能对 BCD 码数字 0～9 进行译码，不能用于显示十六进制数字 A～F。

7447 有 4 个 BCD 码输入端 A、B、C 和 D，其中 D 为最高有效位，A 为最低有效位，它们分别与输出端口中的 4 位相连。7447 的 7 个输出引脚 a～g 直接与 LED 的相应引脚相连，每个段中都串接一个限流电阻，其阻值为 150Ω。当从 A、B、C 和 D 端输入一个 BCD 码时，就能在 LED 上显示相应的数字。

针对共阴极 LED 显示器，也有对应的 BCD 码到七段码的译码器。

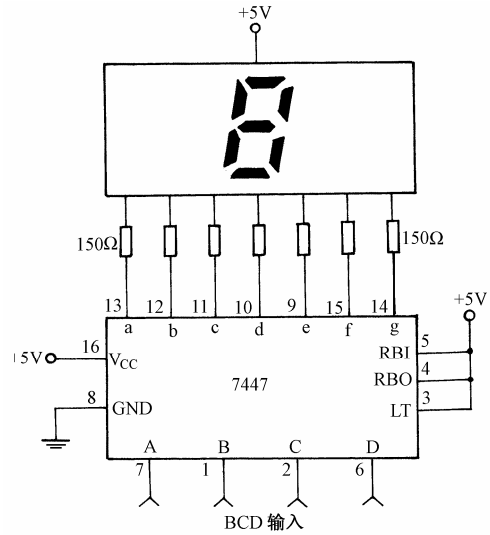


图 8-2　用 7447 驱动单个 LED 显示器

当然，采用 7447 等硬件译码电路，也可以实现多位 LED 的显示接口。

2．采用软件译码方法的一位 LED 显示接口

采用软件译码方法来控制 LED 的显示，就是直接由程序来产生七段 LED 显示编码，显然，对于这种实现方法其硬件电路结构简单，但需要通过程序实现要显示的 BCD 码到七段码之间的转换。图 8-3 就是这样一个 LED 接口电路。

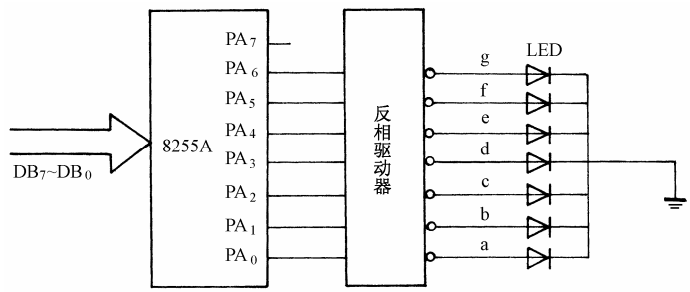


图 8-3　共阴极 LED 显示器接口电路

如图 8-3 所示，LED 数字显示器电路由 8255A 接口芯片、反相驱动器和共阴极 LED 显示器构成。CPU 送来的二进制数字代码从 8255A 的 A 口输出，经反相驱动器驱动后与 LED 相连。若要显示数字 0，应使 g 段熄灭，其余段均点亮。为此，可编程使 PA_6 输出高电平（经反相后输出低电平），其余位输出低电平（经反相后输出高电平），即数字 0 的编码为 01000000B＝40H，其中，PA_7 未被使用，也将它置为 0。同样，可求出 l 的编码为 79H，2 的编码为 24H 等等。将数字 0～F（也可以是 0～9）所对应的七段代码组成一个表，利用 XLAT 指令进行查表，就可求得各数字对应的七段代码值。把要显示的数字的七段代码从 8255A 输出，就可点亮相应的段，显示这个数字。

设十六进制数字的七段代码表的首地址为 TABLE，下面的程序段在 LED 显示器上显示十六进制数中的一位数字（程序中为 5）。

```
PORTA    EQU   F0H                ；A 口地址
COUNT    EQU   05H                ；要显示的数字
DATA     SEGMENT                  ；十六进制数字的七段代码表
TABLE    DB   40H，79H，24H，30H，19H，12H，02H，78H        ；0 1 2 3 4  5 6 7
         DB   00H，18H，08H，03H，46H，21H，06H，0EH        ；8 9 A b C d E F
DATA     ENDS                     ；十六进制数字到七段代码的转换程序
CODE     SEGMENT
                  …
DISPY：  MOV   BX，   OFFSET TABLE      ；七段代码表首地址
         LEA   AL，   COUNT[BX]         ；取 5 的位移量
         XLAT                           ；将被转换的七段代码放入 AL 寄存器
         MOV   DX，   PORTA
         OUT   DX，   AL                ；将七段代码值从 A 口输出，点亮相应数字
                  …
CODE     ENDS
END
```

如果想显示别的数字，只要修改第二行中“EQU”后的数字即可。例如，要显示数字 9，只要将第二行改成“COUNT　EQU　09H”即可。

3．多位 LED 显示电路

在实际系统中经常要显示多位数字。从一位 LED 数码管的显示方法可以看出，无论是采用硬件译码方法还是软件译码方法，当 CPU 向数据寄存器提供 BCD 码或七段字形码后，LED 就能保持显示字形，直到 CPU 对它进行第二次改写。这种显示方式比较适合于具有少量数码管的显示，因为随着数码管个数的增加，数据寄存器的数量也随之增加，同时占用的 I/O 口地址也增多。一般在多个数码管显示时，往往采用另一种设计方法，即动态显示技术。

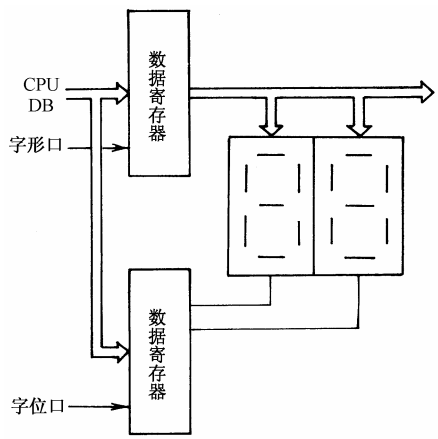


图 8-4　二位数码管动态显示接口电路

如图 8-4 所示是一个采用动态显示技术控制显示字形的二位数码管接口电路。由图可见，接口电路中共有两个数据输出寄存器与共阴极数码管相连。一个寄存器（口地址 10H）的输出连到每个数码管的 a～g 端，用于控制数码管显示的字形，通常称为字形口；另一个寄存器（口地址 20H）的输出分别连接两个数码管的 COM 端，用于控制不同的数码管被选中并点亮显示字形，通常称为字位口。

显然，当 CPU 向字位口输出一个全“1”时，共阴极数码管的 COM 端都为高电平，故数码管不可能有显示。

```
MOV   AL，    0FFH
OUT   20H，   AL
```

如果执行下列程序段：

```
MOV     AL，    06H
OUT     10H，   AL          ；显示数字“1”
MOV     AL，    0FEH
OUT     20H，   AL          ；选中低位数码管
```

结果将会在低位数码管上显示数字“1”。如果希望在高位数码管上显示“2”，则可以执行如下程序段：

```
MOV     AL，    5BH
OUT     10H，   AL          ；显示数字“2”
MOV     AL，    0FDH
OUT     20H，   AL          ；选中高位数码管
```

如果需要在两位数码管上都有显示，例如，显示数字“21”，则不能简单地将控制数码管显示的 COM 端同时设置为低电平，因为 COM 端同时为低确实保证了两个数码管同时都有显示，但显示的内容将是相同的。为了保证每个数码管同一时刻有不同内容的显示，动态显示技术利用了人眼视觉暂留效应的基本思想，使各个数码管轮流显示，每个数码管每次显示 1ms 左右，造成视觉上的稳定显示。

例如，实现显示“21”的程序段如下：

```
DISP：MOV   AL，    06H
      OUT   10H，   AL          ；输出数字“1”的显示代码值
      MOV   AL，    0FEH
      OUT   20H，   AL          ；选中第1位数码管 L1
      CALL  DELAY1MS            ；延时 1ms
      MOV   AL，    0FFH        ；关闭第1位数码管 L1
      OUT   20H，   AL
      MOV   AL，    5BH
      OUT   10H，   AL          ；输出数字“2”的显示代码值
      MOV   AL，    0FDH
      OUT   20H，   AL          ；选中第2位七段数码管 L2
      CALL  DELAY1MS            ；延时 1ms
      MOV   AL，    0FFH        ；关闭第二位数码管 L2
      OUT   20H，   AL
      JMP   DISP                ；继续扫描
```

这里，延时 1ms 的时间量很重要。因为如果延时的时间量较大，比如 1s，显示的效果是每个数码管在轮流显示，一个数码管显示时，则可以用肉眼明显观察到另一个数码管处于不显示状态；如果延时时间较短，比如 1μs，则每一个数码管不能得到稳定的显示效果，肉眼

所看到的是闪烁现象。

对于多于两个的数码管显示，可以采用相同的动态显示方法来实现。

8.2 键盘及接口

键盘是一种常用的输入设备，是由若干个按键组成的开关矩阵，人们可通过键盘输入数据和命令，实现简单的人机通信。键盘接口是将按键这一机械动作转化为可被计算机识别的电信号，供 CPU 读取。

8.2.1 键盘的工作原理

键盘由若干个键开关组成，可以通过按键向处理机输入数字、字符、文字和命令等。

常见的按键都存在两种状态：断开和闭合。当某一键被按下时，为闭合状态；键释放，为断开状态。键盘电路功能就是能将键的闭合和断开状态用“0”和“1”来表示，然后通过数据总线读到 CPU 内部进行键的识别。图 8-5 为单个键的输入电路。

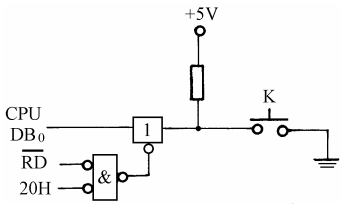


图 8-5 单键输入电路

图中，按键 K 通过三态门连到数据总线的 D_0 位，通过三态门，CPU 可以读取键的状态。当键处于断开状态时，三态门的输入值为“1”;当键处于闭合状态时，三态门的输入值为“0”。因此，用下列两条指令可以判别按键是否被按下。

```
IN      AL，20H
AND     AL，01H
```

程序执行结果若 AL 内容为零，则说明键被按下；若 AL 内容为非零，则说明键未被按下。

在实际使用过程中，一个键盘不可能只由一个键组成。对于多个键组成的键盘阵列，在处理方式上与单个键的处理原理基本相同，其关键技术在于如何进行键识别，也就是说，不仅要知道是否有键按下，还要知道是哪一个键被按下。

由于一般的并行口能够同时输入 8 位数据，因此小于 8 个键组成的键盘处理方式相对简单，只要每一个键的输入线连至并行口数据输入端即可。图 8-6 所示的就是一个由 4 个键组成的键盘阵列。

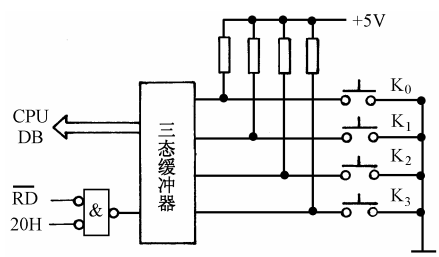


图 8-6 多键接口电路

图中，4 个键 K_0～K_3 分别连至三态缓冲器的 4 根输入线。CPU 读取三态缓冲器的值，并判别哪一线上为“0”，就可以知道当前是否有键被按下，并且是哪一个键被按下。键识别程序如下，在程序转移的目标处，KEY_0～KEY_3 分别表示 4 个键 K_0～K_3 的处理程序。

```
IN      AL, 20H
SHR     AL, 1
JNC     KEY0          ；转键 K0 的处理程序
SHR     AL, 1
```

```
JNC     KEY1        ；转键 K1 的处理程序
SHR     AL, 1
JNC     KEY2        ；转键 K2 的处理程序
SHR     AL, 1
JNC     KEY3        ；转键 K3 的处理程序
```

但是，采用这种结构设计键盘有一个很大的缺点，就是当键盘上的键较多时，引线太多，占用的 I/O 端口也太多。比如一个有 64 键的键盘，就需要 64 条连线和 8 个 8 位端口。所以，这种简单结构只能用在仅有几个键的小键盘中。

通常使用的键盘是矩阵结构的。矩阵式键盘是指键开关按行列排列，形成二维矩阵的结构，如 8×8、4×4 等，是常用的一种键盘结构。

对于 8×8 的 64 键键盘，采用矩阵方式只需用 16 条引线和 2 个 8 位端口便可完成键盘的连接。以 3×3 的 9 键键盘为例，如图 8-7 所示，这个矩阵分为 3 行 3 列，如果键 4 被按下，则第 l 行和第 l 列线接通而形成通路。如果第 l 行线接低电位，则由于键 4 的闭合，会使第 l 列线也输出低电位。矩阵式键盘工作时，就是按行线和列线的电平来识别闭合键的。

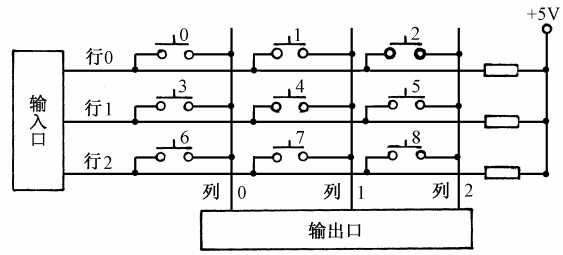


图 8-7　键盘的矩阵结构

8.2.2　键识别的方法

键盘结构的关键是如何把键盘上的按键动作转换为相应的编码，供处理器接收，这里存在一个键码识别的问题。

按照键码识别方式的不同，键盘可分为两大类:编码键盘和非编码键盘。编码键盘是指对每一个按键，由编码电路产生与其对应的惟一编码，也就是说，它是一种用硬件电路来识别按键代码的键盘。而非编码键盘是指利用简单的硬件和一套专用键盘程序来识别按键的位置，然后通过查表程序转换成相应的编码。这种键盘的响应速度虽然不及编码键盘，但因处理机的处理速度远远高于人工敲键的速度，所以这一点无关紧要。由于这种键盘可以通过软件编码为键盘的某些键进行重新定义，因此得到了广泛的使用。计算机中使用的主要是非编码键盘。

1．键识别方法

为了识别键盘上的闭合键，常用的键码识别方法有行扫描法、行反转法及行列扫描法等。

（1）行扫描法。行扫描法识别按键的基本原理是:先将所有的行线置 0，读列线的值，若此时列线上的值全为 1，说明无键按下。若有某位为 0，则说明对应这一列上有键按下，这时改变行扫描码，使行线逐行为 0，依次扫描。当读到某一列线的值为 0 时，就可根据此时的行扫描码和列线的值惟一地确定按键的位置，同时也就确定了该键的扫描码。

（2）行反转法。行反转法也是识别按键的常用方法。它的基本原理是：将行线接一个数据端口，先让它工作在输出方式；将列线也接到一个数据端口，先让它工作在输入方式。编写程序使 CPU 通过输出端口向各行线上输送低电平，然后读入列线值。如果此时有键被按下，则必定会使某列线值为 0。接着，程序再对两个端口进行方式设置，使接行线的端口改为输入方式，接列线的端口改为输出方式。并且，将刚才读得的列值从列线所接端口输出，再读取行线的输入值，那么，闭合键所在的行线值必定为 0。这样，当一个键被按下时，必定可以读得一对惟一的行值和列值。与之配合，行、列线所接的数据端口应能够改变输入、输出方式。8255A 的 3 个端口就具有这个功能。

这种方法要求 CPU 可读/写与行线和列线连接的接口，图 8-8 为利用 8255A 连接 $i\times j$ 键盘矩阵的示意图。

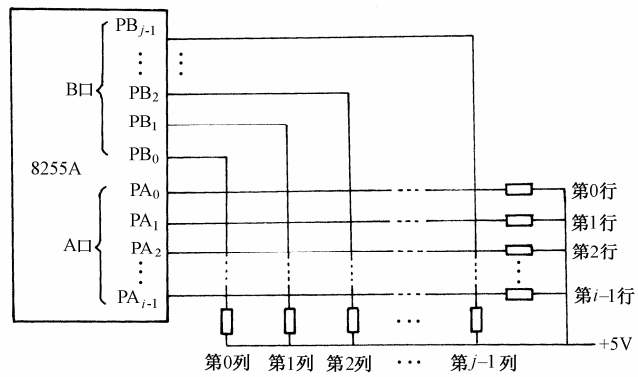


图 8-8　行反转法键盘连接

操作过程如下：

第一，设置 A 口输出，B 口输入。向 A 口写入全 0，然后读 B 口，若 B 口读入全 1，说明没有键被按下，B 口某一位读入为 0，则说明该列有键被按下。

第二，设置 A 口输入，B 口输出。将上一步由 B 口读入的数据再由 B 口输出，这时读 A 口，若某一位为 0，其余全为 1，则说明按下的键在该行，于是就确定了行号和列号，达到了键码识别的目的。

（3）行列扫描法。该方法基本思路为：通过计数译码，依次将各行输出为 0，其余为 1。在扫描每一行时，读列线，若全为 1，说明此行无键按下，若某一列为 0，说明有键按下，且行号和列号均可确定。然后用同样的方法，依次向列线扫描输出，读行线。如果两次所得的行号和列号分别相同。则确定了闭合键的键码。

2．键处理过程中出现的问题

在以上键码识别中，还必须考虑两个问题，一是抖动消除问题，二是重键处理问题。

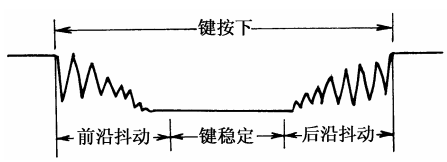


图 8-9　按键抖动波形

（1）抖动消除。在按键闭合、断开过程中，由于机械触点的弹性作用，在闭合和断开瞬间均有抖动现象，其电压信号波形如图 8-9 所示。抖动时间的长短与开关的机械特性有关，一般为 5～10ms。抖动问题不解决就会引起对闭合键的错误认识。

通常消除抖动的措施有硬件方法和软件方法两种：硬件方法可用硬件电路来实现，如用

RC 滤波电路消除抖动。软件方法可用延时的方法，即检测到有键被按下时，执行一个 20ms 的延时程序后再确认该键电平是否仍保持闭合状态，如保持则确认为真正键按下状态，从而消除了抖动影响。

（2）重键处理。重键是指两个或两个以上的按键同时被按下，或者一个按键按下后还未弹起，另一个按键已被按下的情况。处理的办法有两种，第一种是不停地扫描键盘，当有多个闭合键时不予识别，仅以最后检查到的一个闭合键为确认键。第二种方法是确认一个闭合键之后处于保持状态，只有当该键释放后再去处理，并开始识别其他键。

3．键处理流程

对于键盘阵列的处理，同样需要解决两方面的问题：判别是否有键按下以及判别哪一个键被按下。图 8-10 是实现键处理工作的程序流程。

从键处理工作流程中可以看出，键盘接口处理的主要任务有：

（1）检测是否有键被按下。通常的方法是：将键盘的所有列线全部接“ 0 ”，读入行线的值，若所得到的行线的值全为“ 1 ”，说明无键按下；若不全为“ 1 ”，说明已有键按下，因为按下的按键已将所连的列线与行线接通，使相应的行线变为“ 0 ”。

（2）去除键的机械抖动。在判别到键盘上有键闭合后，延时一段时间再判别键盘的状态，若仍有键闭合，则认为键盘上有一个键处于稳定的闭合状态，否则认为是键的抖动。

（3）确定被按下的键所在的行与列的位置。在确认有键被按下的情况下，为找出按下的键所在的行值与列值，可采用行扫描、行反转等方法，判别哪一个键闭合并读取相应的键值。

（4）使 CPU 对键的一次闭合仅做一次处理。为实现这个要求，可等待闭合的键释放以后再做键处理，这种方法可使 CPU 对键的一次闭合仅做一次处理。

根据此流程，可以通过编程来完成相应的键盘接口功能。

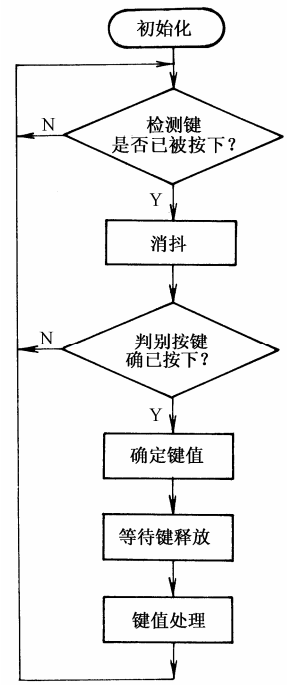


图 8-10　键处理程序流程

如图 8-7 中的 3×3 矩阵键盘接口电路，已知输出锁存器的输出口地址为 30H，三态门的输入口地址为 38H。在键盘中采用的键值编码方法是：行值×16＋列值。可用如下程序来实现该矩阵键盘的接口功能。

```
L1：MOV     AL，    00H
    OUT     30H，   AL          ；输出列线全为 0
    IN      AL，    38H         ；读入行值
    AND     AL，    07H
    CMP     AL，    07H         ；行值是否全为“ 1 ”
    JZ      L1                  ；是，无键按下，转 L1
    CALL DELAY10MS              ；延时 10ms 子程序
    MOV     AL，    00H
    OUT     AL，    30H         ；列线输出值全为 0
    IN      AL，    38H         ；读入行值
    AND     AL，    07H
```

```
        CMP     AL，    07H         ；行值是否全为“1”
        JZ      L1                  ；是，无键按下，转 L1
        MOV     AL，    FFH         ；不是，有键按下
        OUT     30H，   AL          ；列线输出值全为 1
        MOV     BL，    03H         ；移位次数
        MOV     AH，    FEH         ；使第 0 列输出为“0”
L2：MOV AL，    AH
        OUT     30H，   AL          ；输出列线的值
        IN      AL，    38H         ；读入行值
        AND     AL，    07H
        CMP     AL，    07H         ；行值是否全为“1”
        JNZ     L3                  ；不是，有键按下，转 L3
        SHL     AH，    1           ；左移一位
        DJNZ        BL，    L2          ；移位次数是否已到？未到则转 L2
        JMP     L1                  ；所有的列全部判别完毕，若无键按下，转 L1
L3：SHL AL，    4
        AND     AL，    F0H
        AND     AH，    0FH
        OR      AH，    AL          ；键值存放在 AH 中
        IN      AL，    38H         ；读入行值
        AND     AL，    07H
        CMP     AL，    07H         ；行值是否全为“1”
        JNZ     L3                  ；不是，等待键释放，转 L3
KEY_PROC：
                …                   ；键值处理程序段，键值在 AH 中
                JMP     L1          ；转到 L1，进行新一轮的键值扫描
```

8.3 CRT显示器及接口

阴极射线管（Cathode Ray Tube）显示器可用来显示字符、图形和图像，是实现良好的人机交互功能必不可少的输出设备。微型计算机显示系统由显示器（也称监视器）和显示适配器组成，是微型计算机的主要外围设备。

8.3.1 CRT 显示器的工作原理

CRT 显示器可分为两大类型，即单色 CRT 显示器和彩色 CRT 显示器。其中彩色显示器的内部结构如图 8-11 所示，主要由电子枪、偏转装置和荧光屏等 3 部分组成。下面以单枪三束荫罩式彩色显示器为例，介绍彩色显示器的工作原理。

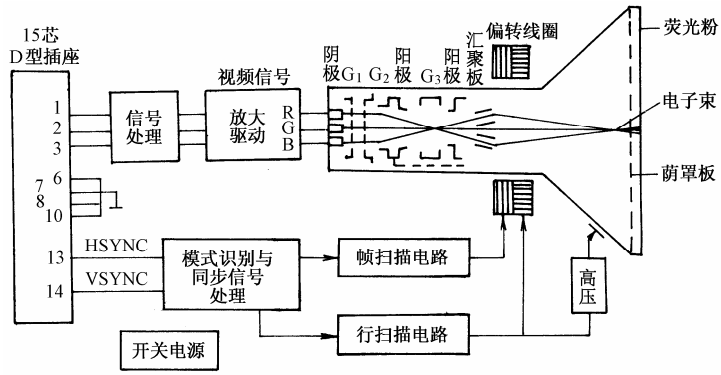


图 8-11　彩色显示器的内部结构

在阴极射线管的电子枪内有 3 个独立的阴极，排成一字形，发出 3 支平行的电子束。改变阴极和控制极 G_1 之间的电压，可控制电子束电流的强弱。屏蔽极 G_2 和阴极之间的电场使两边的电子束折向中心轴，经聚焦极 G_3 聚焦后由两侧射出，再经两对汇聚极板的静电场作用折向中心。最后，经阳极加速后的 3 束电子在排有竖条形孔的荫罩板的细缝中汇聚后分别准确地轰击涂在荧光屏上对应红绿蓝的三色荧光粉，使之产生不同颜色的光点。

帧/行扫描电路分别向垂直/水平偏转线圈提供帧频和行频锯齿波电流，从而在电子枪前部的管颈内产生两个互相垂直的，强度按帧频/行频变化的偏转磁场。电子束穿过这两个磁场时，受到垂直/水平两个方向的作用力而产生位移，即从左向右，从上向下扫描荧光屏，产生一幅幅光栅，如图 8-12 所示。水平扫描包括正程（显示）和逆程（消隐）。水平扫描率又称为行频，垂直扫描也包括正程和逆程。垂直扫描频率又称帧频。光栅扫描有逐行和隔行两种方式。为保证屏幕无闪烁感，可选择逐行扫描方式 CRT。整个屏幕被扫描线分成 m 行，每行有 n 个点，每个点为 1 个像素。整个屏幕有 $m \times n$ 个像素。图形是由电子束扫描时在屏幕上产生的不同亮度、不同颜色的光点（即像素）组成的。

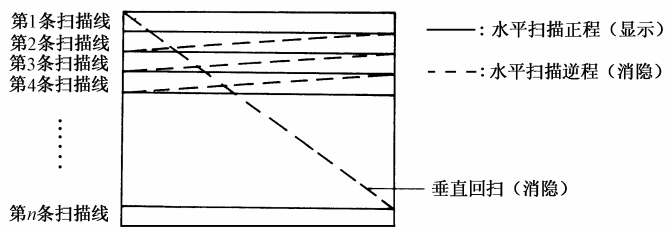


图 8-12　光栅扫描示意图

8.3.2　CRT 显示器接口

作为输出设备的显示器与系统主机的连接必然要通过接口电路，这就是 PC 系列机中的显示适配器。显示适配器通常制成接口电路卡，插在系统扩展槽上。它一方面通过系统总线与主机联系，接收主机提供的显示信息和显示控制命令；另一方面它通过视频接口向显示器输出视频信号，控制显示器的显示。需要说明的是，现在部分 PC 机的显示适配器是集成在系统主机板上的。

目前，CRT 显示器一般采用 37 芯 D 形插座与 CRT 显示卡连接，再通过 CRT 显示卡与主机连接。

1．视频显示标准

随着显示技术的不断发展和人们对显示效果要求的不断提高，陆续形成了一系列视频显示标准。从 MDA、CGA、EGA 到 VGA 等每一种视频标准都有相应的显示卡与之对应。这些标准实质上反映了各种视频显示图形卡的性能，即显示工作方式：屏幕显示规格、分辨率及显示色彩的种类。显然，在同样尺寸的显示器上，字符显示的列数、行数愈多，图形显示的分辨率愈高，可显示的色彩种类愈多，即表明相应的视频显示接口——图形显示卡的性能愈好。

（1）MDA 标准。MDA（Monochrome Display Adapter）是单色显示适配器。它是 IBM 规定的 PC 视频显示的第一个标准。该适配器仅支持文本模式（字符显示），不支持图形模式，且是黑白方式（或绿色、琥珀色，随显示器而异）显示。MDA 显示标准为显示方式 7，字符显示规格为 80 列×25 行，字符框为 9×14 点阵，而字符点阵为 7×9 点阵，故分辨率为 720×350。MDA 配置 4KB 显示缓冲存储器，绝对地址始于 B0000H，正好存放一帧字符显示信息。

（2）CGA 标准。CGA（Color Graphics Adapter）是彩色图形适配器。它与 MDA 相比增加了彩色显示和图形显示两大功能，它既支持文本模式，又支持图形模式。CGA 字符显示标准是方式 0～方式 3，采用的字符框为 8×8 点阵，字符点阵为 7×7。图形显示标准是方式 4～方式 6，最大分辨率为 640×200。CGA 配置 16KB 显示缓存，绝对地址始于 B8000H，40 列×25 行方式时可存放 8 帧显示字符，80 列×25 行方式时可存放 4 帧显示字符，图形方式时可存放 l 帧图形信息。

（3）EGA 标准。EGA（Enhanced Graphics Adapter）是增强图形适配器。它除兼容 MDA（方式 7）和 CGA（方式 0～方式 6）外，还支持增加的图形显示标准方式 13～方式 16，分辨率可达 640×350。在 EGA 与 MDA 和 CGA 兼容的各方式中，可使用的显示缓存仍为 4KB 和 16KB，其绝对地址分布也相同。但扩展的 4 种图形显示方式使用 4 个 16KB 或 4 个 64KB 的显示缓存，它们共同的起始地址为 A0000H。

（4）VGA 标准。VGA（Video Graphics Array）是视频图形阵列。它兼容 EGA 的所有显示标准，还扩展了新的图形显示标准方式 17～方式 19。与 EGA 类似，VGA 需要 4 个 64KB 的显示缓存才能支持所有的显示方式，它们都位于起始地址为 A0000H 的区域中，分辨率为 640×480。

（5）TVGA 标准。TVGA 是 Super VGA 产品，由 Trident 公司推出，它兼容 VGA 全部显示标准并扩展了若干字符显示和图形显示的新标准，具有更高的分辨率和更多的色彩选择，最大分辨率为 1 024×768。

随着各种图形软件，特别是 Windows 的日益流行，大量的图表和窗口需要显示，图形显示速度越来越重要，CPU 已不堪重负。正当微软分司竭力用软件的方法提高图形显示速度的同时，一些小型专业图形处理公司开发出了高速图像处理专用芯片，因此，采用图形加速芯片的图形加速卡很快流行开来。最初，图形加速卡只能用于图形加速，必须与 VGA 显示卡配合才能使用，到后来不仅集成了显示卡的功能，还具备各种动态视频的功能，成为目前显示卡的主流产品。

2．彩色图形适配器的结构与功能

无论何种标准的显示卡，其功能都是接收来自主机的数据，将其转变为视频信号，并与其产生的水平同步信号、垂直同步信号一起送 CRT 显示器显示。图 8-13 给出了一个 VGA 卡的结构示意图。

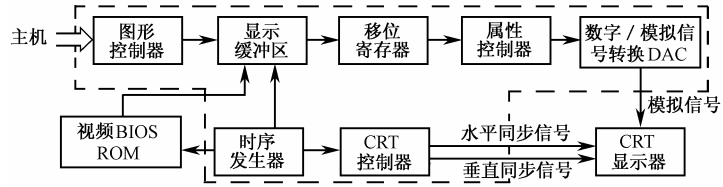


图 8-13　VGA 卡结构示意图

其中，图形控制器接收主机送来的显示数据，并可对数据进行与、或、异或和循环移位等逻辑运算，然后写入显示缓冲区。

显示缓冲区为一动态随机存储器，用来存放显示字符的 ASCⅡ码和属性代码、字符点阵信息或者是存放被显示图形的位图。VGA 卡中显示缓冲区容量为 256～512KB。

从显示缓冲区中读出的像素值经移位寄存器转变成串行信号送入属性控制器，属性控制器的基本功能是将像素值转换成颜色值。

数字/模拟信号转换 DAC 将选出的颜色值转换成模拟信号，并输出至 CRT 显示器。

CRT 控制器是 CRT 接口的核心控制部件，它的功能有两个，一是产生水平和垂直同步信号送 CRT 显示器，使 CRT 的电子束不断地从上到下、从左到右进行扫描，产生光栅；二是根据电子束在屏幕上的行列位置，自动计算并生成显示缓冲区的相应地址，不断地控制读出显示缓冲区中的像素值。

时序发生器则产生 CRT 控制器及动态存储器所需的时序信号，用来解决主机处理器和 VGA 的图形控制器访问显示缓冲区的时序冲突。

视频 BIOS 是一个只读存储器，里面除了固化视频控制程序外，还固化有不同字符集的字符点阵，在文本显示模式下充当字符发生器的角色。

目前流行的图形加速卡的基本结构如图 8-14 所示。

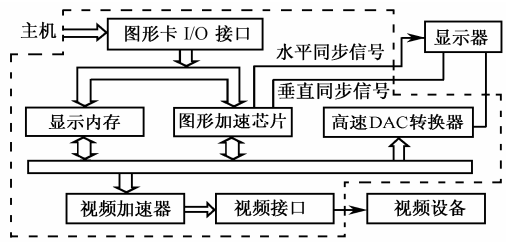


图 8-14　图形加速卡的结构示意图

图形加速卡看似与 VGA 卡差不多，但实际上有许多重大改变。首先，图形控制器变成了图形加速芯片，这是最重要的变化。前者只能在 CPU 指令下执行一些简单的控制任务，而后者不仅有控制作用，更主要的是加工处理显示内存要显示的图像，分担 CPU 画图的工作，因此具有智能性。其次，图形卡上是板卡总线方式，一般为 64 位，总线密度最大为 128 位，

超过了 PC 系统总线，这样使各部件之间的数据吞吐量大为增加。第三，有专门的视频加速电路，可以提高动态影像的解压缩速率。最后，DAC 有了高速转换加速器，使模拟信号输出速率增加；显存采用快速的内存芯片（如 SDRAM、SGRAM、VRAM 等）；与总线连接的数据线宽度增加，一般为 32 位的 VESA，最大的有 64 位的 PCI，比 VGA 的 ISA 和 EISA 有了较大提高。

如今，改用 AGP 总线，显示卡不是插在传统的总线上，而是插在重新设计的与 CPU 直接连通的插槽上。新设计比传统设计显示速度提高了 4 倍以上。

3．两种基本显示模式

在介绍视频显示标准时，曾提到有的标准仅支持文本模式，有的标准既支持文本模式又支持图形模式，下面就这两种基本显示模式做一简单描述。

（1）文本模式。文本模式又称字母/数字（A/N：Alphabetic/Numeric）模式，以字符为基本显示单位在屏幕上显示信息。例如，字符分辨率为 80×25，是指一屏能显示 25 行，每行显示 80 个字符。常见的字符分辨率还有 80×30、80×60、132×60、128×48 等。

在文本模式下，显示缓冲区内用两个字节地址空间为每个字符保存信息，其中一个字节用来保存字符的 ASCII 码（或扩充的 ASCII 码）值，另一字节则用来存放字符的属性（如前景色、背景色及闪烁等）。然而，在显示器上显示的内容都是以构成光栅的点的形式出现的，所以必须把显示缓冲区中保存的信息转换成可供显示器直接使用的矩形字符点阵。所谓矩形字符点阵是指每个字符由若干行和若干列的像素点组成。如一个字符的字模为 8×8 的点阵，则表示该字符框包括 8 行×8 列个像素点。

表示字符点阵的二进制代码存放在字符发生器中。字符发生器是一个只读存储区域，任何一个字符的点阵模式都可以从字符发生器中取到。常用的字符点阵有 8×8、8×14 和 9×16 等。每一种点阵构成的字符包括 256 个字符的字模，全部字模都存放在字符发生器中。

对文本模式来说，显示转换分两步进行：首先，按照字符的 ASCⅡ码信息为每个扫描行构造出精确的点阵形式，然后在这些点阵发送至显示器之前再加入相关的属性信息。

（2）图形模式。图形模式也称为全点可寻址（APA：All Points Accessable）模式，以像素为单位在屏幕上显示信息。如图形分辨率为“1 024×768”是指屏幕上水平扫描线的数目为 768，每行扫描线所含的点数为 1 024。

在图形模式下，显示缓冲区以位的形式为每个像素保存信息，每个像素仅具有独立的颜色属性，无形状，无闪烁，无背景颜色，也无数据。显示图形时，用描述像素颜色属性的二进制数的位数决定可同时显示的颜色数。当二进制位数为 1 时，可显示 2 种颜色；当二进制位数为 4 时，可显示 16 种颜色；当二进制位数为 8 时，可显示 256 种颜色。这种保存信息的格式与文本模式相比更接近于在显示器上显示的图形。

针对图形模式来说，显示转换所要做的工作是使用正确的属性，以正确的顺序向显示器发送像素信号。

4．CRT 显示器中断程序调用

在计算机系统中，视频显示卡实现的功能非常重要且相当复杂，用某些软件工具来帮助控制显示卡的运行是一种行之有效的方法。对于标准显示卡来说，这些软件工具是由 BIOS ROM 中的例行服务程序组成的，通过“INT 10H”的软中断可以调用它们。视频中断实现的

控制功能所覆盖的范围既大又细，大到可以对显示卡工作方式进行整体控制，细到在显示器上可以写一个单个字符或在显示器上放一单个的像素。表 8-2 是视频 BIOS 的基本功能表。

表 8-2　INT 10H 功能表

功　能　号	功　　能	入 口 参 数	出 口 参 数
00H	设置显示模式	AL=模式号	
01H	设置光标尺寸	CH=光标起始行，CL=光标结束行	
02H	设置光标位置	BH=页号，DH=行号，DL=列号	
03H	读光标位置	BH=页号	CH/CL=光标起始/结束行，DH/DL=行/列
04H	读光笔位置		BX=像素列号，CX=像素行号 DH=字符行号，DL=字符列号
05H	选择工作页面	AL=页面号	
06H	文本窗口上滚	AL=上滚行数，AL=0 初始化窗口 BH=字符填充属性 CH/CL=窗口左上角行/列坐标 DH/DL=窗口右下角行/列坐标	
07H	本文窗口下滚	AL=下滚行数，AL=0 初始化窗口 BH=字符填充属性 CH/CL=窗口左上角行/列坐标	
08H	读光标位置的字符和属性	BH=页号	AH/AL=属性/字符
09H	在光标位置写字符和属性	AL=字符 BH/BL=页号/属性 CX=重复写字符的个数	
0AH	在光标位置写字符	AL=字符 BH/BL=页号/图形模式时字符前景色，CX=重复写字符的个数	
0BH	置彩色调色板（CGA）		
0CH	写像素	AL=像素值（颜色） CX=像素列号，DX=像素行号	
0DH	读像素	BH=页号，CX=像素列号 DX=像素行号	AL=像素值（颜色）
0EH	以 TTY 方式写字，光标移动，解释命令字符	AL=字符、BX=页号/图形模式时，字符前景色，命令字符：07H 响铃，08H 退格，0AH 换行，0DH 回车	
0FH	读当前显示模式		AH/AL=每行字符数/当前显示模式 BH=工作页面号

5．程序设计举例

例 8.1　把整个屏幕作为窗口进行上滚操作，清除屏幕（字符显示方式，且字符分辨率为 $X \times Y$）。

实现上题目要求的程序如下：

```
MOV     CX，0              ；设置窗口左上角，坐标为 0 行、0 列
MOV     DH，X
MOV     DL，Y
DEC         DH
```

```
DEC     DL                  ；取得右下角的行号、列号
MOV     BH，7
MOV     AH，6
MOV     AL，0               ；清除窗口内容
INT     10H
HLT
```

例 8.2　利用 BIOS 中的写像素点功能画出一个填充的矩形（图形方式）。

```
        MOV     DX，100         ；像素点的 Y 坐标，同时兼计数
LOOP1： MOV     CX，100         ；像素点的 X 坐标，同时兼计数
LOOP2： MOV     AL，            ；设置 1#颜色
        MOV     AH，0CH
        INT     10H             ；在（X, Y）处画点
        DEC     CX
        JNZ     LOOP2
        DEC     DX
        JNZ     LOOP1
        HLT
```

8.4　鼠标器及接口

鼠标器是一个控制计算机屏幕上光标移动的小型手控输入设备。随着图形窗口软件的逐步普及，鼠标器已为广大用户普遍采用。在运用这些软件时，鼠标器比键盘更方便。人们只要用一只手握住鼠标，让它在桌面或专用的板子上滑动，就可以把鼠标器的运动方向和距离信息送给显示器，使屏幕上的鼠标指针随之移动。利用鼠标器的移动并配合鼠标器的按键，即可快速有效地进行各种操作。使用了鼠标器能使计算机的操作更容易、更有效。

8.4.1　鼠标器的工作原理

鼠标器是一种快速定位器，利用鼠标器可以方便地定位光标在显示屏幕上的位置。当鼠标器在平面上移动时，随着移动方向和快慢的变化，会产生两个在高低电平之间不断变化的脉冲信号，CPU 接收这两个脉冲信号并对其计数。根据接收到的两个脉冲信号的个数，CPU 控制屏幕上的鼠标器指针在横轴（*X*）、纵轴（*Y*）两个方向上移动距离的大小。

脉冲信号是由鼠标器内的半导体光敏器件产生的。根据结构的不同，鼠标器一般可分为光机式和光电式两种，也可以分别称之为机械式鼠标和光学式鼠标。

1．光机式鼠标器

光机式鼠标器在其基座底部凹处安有一个实心的橡胶球。内部有两个互相垂直的滚轴紧靠在橡胶球上。在两上滚轴的顶端各装有一个边缘开槽的光栅轮。光栅轮的两侧分别安装着由发光二极管和光敏三极管构成的光电检测电路。

当鼠标器在平面上移动，橡胶球与平面摩擦滚动时，带动滚轴及其上的光栅轮旋转。因

为光栅轮开槽处透光，未开槽处遮光，使得光敏三极管接收到的由发光二极管发出的光线时断时续，而产生不断变化的高低电平形成脉冲电信号。互相垂直的两个轴对应着屏幕平面上的 X 轴、Y 轴两个方向。脉冲信号的数量对应着位移的大小。

2．光电式鼠标器

光电式鼠标器没有橡胶球和带光栅轮的滚轴，而是在其基座上装有两对发光二极管和光敏接收管。其两对光电检测器互相垂直，光敏三极管检测发光二极管照射到鼠标下面垫板上产生的反射光的强度变化，从而确定鼠标器在 X、Y 两个方向上的位移。因此，光电式鼠标器需要工作在画有均匀网格的专用垫板上。

当发光二极管发出的光线照到网格线上时，被吸收而无反射光；若照到格内时，则有反射光，光敏三极管据此产生高低电平，形成脉冲信号。

8.4.2 鼠标器接口

鼠标器按接口分类主要有串行通信鼠标器、总线鼠标器、PS/2 鼠标器以及 USB 鼠标器。

（1）串行通信鼠标器。串行通信鼠标器一般采用 RS-232C 标准接口进行通信。PC 系列微型机上一般有两个串行通信接口 COM_1 和 COM_2。串行通信鼠标器不需要专门的电源线，而是由 RS-232C 串行通信接口线路中的 RTS 提供驱动，SGND 作为地线，使用 TxD 发送数据，DTR 作为联络信号线。在串行通信鼠标器控制板上配置有微处理器，其作用是判断鼠标器是否已启动工作，工作时组织输出 X、Y 方向串行位移数据。大多数鼠标采用 7 位数据位，1 位停止位，无奇/偶校验方式，以 1 200～2 400 b/s 的速率发送数据。对于串行通信鼠标器，通常带有一个 9 针的 D 型插头，只要将其插到微机的串行口 COM_1 上即可。有些型号的鼠标器带有 9～25 针转换插头，利用它可将鼠标器接在 COM_2 串行口上。

（2）总线式鼠标器。总线式鼠标器本身不带微处理器，但是需要在主机系统总线扩展槽中插一专用接口板。鼠标器与接口板之间采用 9 针插头连接。9 针插头如图 8-15 所示。其中，SW_1、SW_2、SW_3 为按键开关信号，X_A、X_B、Y_A、Y_B 分别表示 X、Y 方向上鼠标的位移量。接口板上配置有可编程器件，用来监视鼠标器的工作，置位内部数据寄存器，控制发送中断请求信号，调整对鼠标器信号采样的频率等。

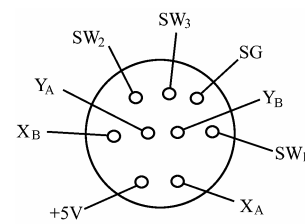


图 8-15　9 芯插头

（3）PS/2 鼠标器。PS/2 鼠标器是最早用在 IBM PS/2 系列微机上的鼠标，并由此而得名。它使用专用的鼠标接插座（6 芯 DIN 插型头），安装灵活方便，不占串口，但必须注意，使用 PS/2 鼠标时，主板上应有专门支持 PS/2 鼠标接口的插座。

（4）USB 鼠标器。USB 鼠标器也是一种串行鼠标器，与 RS-232 接口的鼠标器和 PS/2 接口的鼠标器不同，USB 鼠标器具有 USB 接口，即插即用，适用于具有 USB 接口的 PC 机。

微软公司为鼠标器提供了一个软件中断指令 INT 33H，只要加载了支持该标准的鼠标器驱动程序，在应用程序中便可直接调用鼠标器进行操作。INT 33H 有多种功能，可通过在 AX 中设置功能号来进行选择。INT 33H 功能调用如表 8-3 所示。

表 8-3　INT 33H 功能表

功　能	入口参数	出口参数	参数描述

AX=0H 初始化鼠标		AX BX	AX=-1 已安装鼠标，AX=0 未安装鼠标 BX=按键数目
AX=01H 显示光标			
AX=02H 关闭光标			
AX=03H 读取按键状态及光标位置		BX (CX, DX)	B_0、B_1、B_2 表示左、右、中按键，B_i=1 表示某键按下 光标位置（*X*，*Y*）
AX=04H 设置光标位置	(CX, DX)		光标位置（*X*，*Y*）
AX=05H 读取按键按下信息	BX	AX BX (CX，DX)	B_0、B_1、B_2 表示左、右、中按键，A_0、A_1、A_2 表示左、右、中按键状态 按键按下次数 光标位置（*X*，*Y*）
AX=06H 读取按键释放信息	BX	AX BX (CX，DX)	B_0、B_1、B_2 表示左、右、中按键，A_0、A_1、A_2 表示左、右、中按键状态 按键按下次数 光标位置（*X*，*Y*）
AX=07H 设光标横向移动范围	CX DX		*X* 最小值 *X* 最大值
AX=08H 设光标纵向移动范围	CX DX		*Y* 最小值 *Y* 最大值
AX=09H 定义图形光标形状	(BX, CX) EX:DX		光标基点（*X*，*Y*）坐标 光标图案首址(0～1FH：背景；20H～3FH：图案)
AX=0BH 读鼠标位移量	CX DX		*X* 方向位移量（-32 768～32 767），单位0.127mm（1/200in） *Y* 方向位移量（-32 768～32 767），单位0.127mm（1/200in）

8.5 打印机接口

打印机是计算机系统的基本输出设备之一，能直接简便地获得硬拷贝。打印机的种类很多，按照印字原理，目前常用的打印机有 3 种：针式打印机、喷墨打印机和激光打印机。

针式打印机是一种击打式打印机，它是靠打印机内部的机械电路控制打印头来完成打印动作的。打印头由若干根钢针及相应数目的电磁铁构成。钢针的数目一般为双列 24 根。每根钢针对应一块电磁铁，电磁铁根据电脉冲信号产生对钢针的吸合与释放动作，当钢针向前撞击时，就把色带上的油墨打印到纸上形成一个色点。

喷墨打印机是靠喷出的微小墨点在纸上组成图形、字符或汉字的，其主要技术环节是墨滴的形成及其充电和偏转。在喷墨打印机内部，使用充电电极施加一个静电场给墨滴充电，工作时，导电的墨水在墨水泵的高压力作用下进入喷嘴，通过喷嘴形成一束极细的高速射流，射击流通过高频振荡发生器断裂成连续均匀的墨水滴流。带不同电荷的墨滴通过加有恒定高压偏转电极形成的电场后垂直偏转到所需的位置。

激光打印机是激光、微电子和机械技术的综合应用。这是一种将激光扫描技术与电子照相技术相结合的非击打式打印输出设备，打印速度之快、打印质量之高是其他各类打印机所无法比拟的。在激光打印机内部，激光器输出的光源被聚焦成一个很细小的光点，沿着硒鼓进行横向重复扫描，使硒鼓上的电荷沉积情况发生变化，当硒鼓与普通纸接触时，由于静电电荷的作用，硒鼓表面的碳粉就会被吸附到纸上。这样，通过控制激光扫描，就可以在纸上打印出不同的字符和图形。

8.5.1 打印机接口信号

虽然各类打印机的打印原理不同，但与 CPU 的连接方式都是相同的。在打印机内部，除了打印机构以外，接口与控制电路用来连接主机，并实施对打印机的控制。主机向打印机传输的数据类型不外乎两种：一类是可打印的数据，包括可打印字符码或可打印的图形码；另一类是管理控制打印机工作的控制命令，如回车、换行等命令。

对可打印的数据来说，凡是属于可打印的字符码（如 ASCⅡ码），打印机接收以后均要从字符发生器中检索出相应的点阵数据，并存放到打印行缓冲区中；而对可打印的图形码，由于这种图形码本身就已经是点阵数据，所以可以不经过字符发生器直接写入行缓冲区。当行缓冲区装满或打印机接收到打印命令后，打印机控制部件将把行缓冲区中存储的点阵数据按当前打印头所处的列位置将数据发送到打印驱动电路，完成打印的一系列动作。

打印机有串行和并行之分，因此，它和主机之间的接口也有串行与并行两种。在串行接口中，早期的打印机采用的都是 RS-232 接口，现在的打印机通常都配有 USB 接口。而并行接口仍是目前打印机的首选接口。以并行接口为例，打印机与主机之间通过一根电缆线连接，电缆线的一头插座为 36 芯，与打印机相联，另一头为 25 芯，与主机并行接口相联。

目前，并行打印机的接口一般都是按照 Centronics 标准来定义插头插座的引脚。的 Centronics 接口是得到工业界大量支持的一个并行接口协议。这个协议规定了 36 脚簧式插头座为打印机标准插头座，并规定了 36 脚的信号含义。表 8-4 列出了 Centronics 标准中主要引脚信号的名称和功能。

表 8-4　Centronics 标准主要引脚信号

引　脚	名　称	方　向	功　能
1	$\overline{\text{STROBE}}$	入	数据选通，有效时接收数据
2～9	$DATA_1$～$DATA_8$	入	数据线
10	$\overline{\text{ACK}}$	入	响应信号，有效时准备接收数据
11	BUSY	出	忙信号，有效时不能接收数据
12	PE	出	纸用完
13	SLCT	出	选择联机，指出打印机不能工作
14	$\overline{\text{AUTOLF}}$	入	自动换行
31	$\overline{\text{INIT}}$	入	打印机复位
32	$\overline{\text{ERROR}}$	出	出错
36	$\overline{\text{SLCTIN}}$	入	有效时打印机不能工作

8.5.2 打印机接口逻辑及编程应用

PC 打印机并行接口内部有 3 个寄存器，分别对应 3 个端口地址，即数据口、控制口和状态口。主机可以分别对它们进行读/写操作，如图 8-16 所示。其中，控制寄存器的格式如图 8-17 所示，状态寄存器格式如图 8-18 所示。

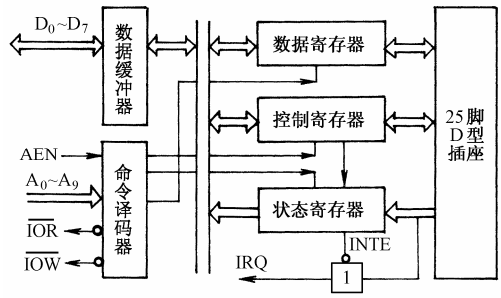


图 8-16 并行接口逻辑图

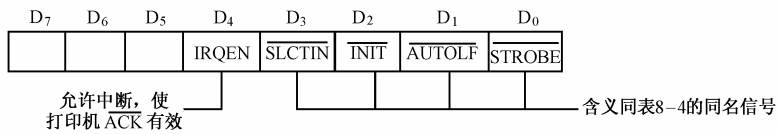


图 8-17 控制寄存器的格式

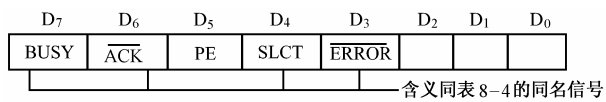


图 8-18 状态寄存器的格式

下面的程序通过对 3 个端口的编程，可以完成将 AL 中的字符送打印机输出的功能。3 个端口的地址分别为数据口 378H、状态口 379H 及控制口 37AH。

```
        MOV   DX，0378H
        OUT   DX，AL          ；将打印字符送数据口
        INC   DX
WAIT:   IN    AL，DX          ；读状态口
        TEST  AL，80H         ；检测 BUSY 位
        JNZ   WAIT            ；不忙则输出选通，忙则等待
NEXT-OUT: MOV AL，ODH
        INC   DX
        OUT   DX，AL
        MOV   AL，OCH
        OUT   DX，AL          ；在控制口写入，使选通有效
```

在 ROM BIOS 中固化有打印机 I/O 功能程序，可用软件中断 INT 17H 来调用。它又包括 3 个子功能：

（1）0 号功能——送入打印机一个字符（AH＝0）。

入口参数：AL=打印字符，DX=打印机号（0～2）；

出口参数：AH=打印机状态。

（2）1 号功能——初始化打印机（AH=1）。

入口参数：DX=打印机号（0～2）；

出口参数：AH=打印机状态。

（3）2 号功能——读打印机状态（AH＝2）。

入口参数：DX=打印机号（0～2）；

出口参数：AH=打印机状态。

上述 3 个功能调用返回的参数都是打印机的状态字节，其含义为：某位为 l，则反应不忙（D_7）、响应（D_6）、无纸（D_5）、选中（D_4）、出错（D_3）和超时错误（D_0）。

本 章 小 结

I/O 接口的重要性是显而易见的。在各类接口中，与用户关系最密切的就是人机接口。本章介绍的就是这些人机接口：LED 显示器接口、键盘接口、CRT 显示器接口、鼠标器接口和打印机接口。

LED 显示器是最廉价的显示器之一，很适合应用于简单、少量地数码显示。由七段发光二极管组成的 LED 显示器，显示关键的不同字形在于提供不同的二极管点亮组合。对于单个 LED 数码管，可以直接提供七段码，也可以由硬件电路来产生七段译码；对于多位 LED 的显示控制，可以是各显示位独立显示，也可以采用动态扫描的方法。不同显示方法的硬件接口电路和程序的编制方法都是不同的。

键的工作原理很简单：按键的动作引起电路的导通与断开，从而引起电信号的变化，接口电路就是通过电信号的变化来判断是否有键被按下或被释放。键盘接口的关键在于键的识别，就是能够快速准确地知道是哪个键被按下，尤其是在有很多键的键盘接口电路中。键识别的方法有很多种，如行扫描法、行反转法等。

光栅扫描是 CRT 显示器的基本工作原理。为了能控制 CRT 的光栅扫描以及产生相应的显示信号，CRT 接口电路需要不断地、周而复始地提供显示信息和定时控制时序。PC 机系列有多种显示模式，CRT 接口适配器也有从单色字符显示到彩色图像显示、从低分辨率到高分辨率的多种显示模式。

鼠标器的工作原理相对比较简单，通过串行接口电路与主机相连。在 PC 机系列中，可以通过 INT 33H 中断指令来实现鼠标器的各项功能调用。

各类打印机的工作原理不同，但打印机接口电路的功能是相同的：将主机需要打印的数据和控制信号送打印机。因此，接口电路基本上也是相同的。标准的打印机并行接口规定了 36 芯的数据线、控制线、状态和地线，相应的接口电路为用户提供数据端口、控制端口和状态端口。

习 题 8

8.1 由发光二极管（LED）组成的七段数码显示器有哪两种接法？不同接法对字符的显示有什么影响？

8.2　多位 LED 显示器采用动态扫描显示和静态显示有什么区别？

8.3　用 8255A 的 A 口和 B 口分别作为某一微机系统中 8 位七段 LED 显示器的段码和位码的输出端口，要求按动态扫描、分时显示的原理循环显示“20020901”8 个字符，试设计硬件接口电路和显示驱动程序。

8.4　行扫描法和行反转法都是键的常用方法。试针对图 8-7 所示的键盘阵列编写程序，采用行反转法完成键的识别工作。

8.5　消除按键抖动影响的方法有几种？如何保证一次按键动作程序只处理一次？试编写程序段来说明解决这些问题的方法。

8.6　说明 CRT 显示器光栅扫描的工作原理。

8.7　在图形适配器中，CRT 控制器的主要作用是什么，字符发生器内存放的是什么信息？

8.8　试说明光机式鼠标器和光电式鼠标器的工作原理。

8.9　Centroncs 接口的前 11 个信号线的功能是什么？它们是怎样配合数据输出的？

8.10　根据 Centroncs 接口信号，为打印机设计一个接口电路，CPU 可以通过中断方式向打印机传送打印数据，试设计电路图并编写相应的程序。