Intel 单片微机手册

上海市微型电脑应用协会

前　言

单片机是在一块硅片上把 CPU、ROM 和 RAM，以及 I/O 电路集成在一起的智能器件，只要外接 5V 电源和 LC 振荡器或晶体就能工作，单片机具有价格低廉、使用方便、体积小和重量轻等特点，因此获得普遍应用，美国 Intel 公司在世界上最早研制并大量生产单片机，单片机又称微控制器(Microcontroller)，在工业控制方面得到广泛应用，在仪器、仪表和家用电器产中用单片机作控制器，可以大幅度提高质量和性能，因此单片机受到极大的重视，成为微型计算机中销量最大的产品。

本书主要阐述 MCS-96 系列 16 位单片机、MCS-51 系列和 MCS-48 系列 8 位单片机，以及带有通讯控制器的 RUPI 系列单片机，详细介绍这些单片机的结构原理和使用方法，还给出了这些单片机的技术性能、具体参数和应用示例。本书对微机系统和工业自动控制的设计使用人员，以及高等院校师生都有参考价值。本书由林匡定、顾良士、孙德和项湜伍等同志翻译，由韦众成副教授审校。

由于时间仓促和水平有限，错漏之处请读者不吝指正。

译校者

目　录

第一章　MCS-96 介绍

1.0　微控制器的持续发展

从 1976 年推出世界标准 8048(MCS-48)微控制器以来，Intel 公司于 1980 年推出了工作性能明显优于 8048 的 8051(MCS-51)，由于 8051 有种种优点，这种微控制器的应用程序库发生了显著的进步，这些通用芯片的应用场合从键盘和终端一直到控制汽车引擎，8051很快地就成了微控制器的第二代世界标准。

如今，半导体处理技术发展到了一个新的阶段，已经有可能在单块硅片上集成十万个以上的晶体管，Intel 公司的微控制器设计人员采用一些最新的处理技术成果，制造出新一代的单片微控制器 MCS-96。8096(MCS-96 的商品器件编号)在单片微处理器中具有已经有过的最高系统集成度，它集成了十二万个以上的晶体管来实现高性能的16位的CPU、8K 字节的程序存贮器、232 字节的数据存贮器以及模拟和数字两种类型的 I/O 特性，图 1-1 说明了 Intel 公司单片微控制器的发展历程，

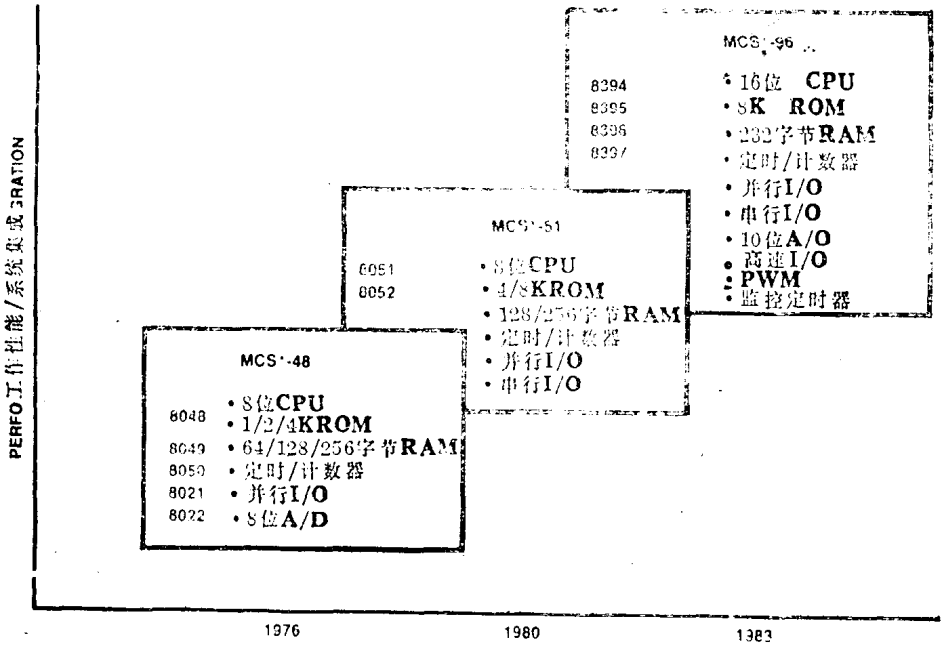

图 1-1　Intel 公司微控制器的发展历程

1.1　MCS-96 简介

8096 中的 16 位高性能 CPU 与程序和数据存贮器紧密地联系在一起，它们和若干个 I/O 特性一起集成在单块硅片上，该 CPU 支持位、字节和字操作，对 32 位双字的支持形成了指令集中的一个子集，输入频率为 12 MHz 时，8096 能够在 1.0 μs 内执行一次 16

位加，也能在 6.5μs 内执行一次 16×16 的乘法或 32/16 的除法操作。

提供有四个高速触发器输入端来记录外部事件发生的时刻，分辨率为 2 μs（输入 12 MHz 晶体频率时）。最多可以提供六个高速脉冲发生器输出端，用来在预定的时刻触发外部事件，高速输出单元能够同时执行定时器功能，除了两个16位的硬件定时器之外，最多可以有四个这样的 16 位软件定时器同时投入运行。

任选的片内 A/D 转换器可以转换最多四个(48 脚型)或八个(68 脚型)模拟输入通道成 10 位数字量，片内还提供串行端口、监控(watchdog)定时器和脉冲宽度调制(PWM)输出信号，表 1.1 是对 MCS-96 特性和好处的小结。

表 1.1 MCS-96 特性和好处小结

特　　性	好　　处
16 位 CPU	高处理能力
8K 字节 ROM	大程序空间可用于较复杂的大型程序
232 字节 RAM	大的板上寄存器堆
硬件乘/除	提供良好的运算能力，输入 12MHz 时能在 6.5μs 内完成 16 位乘 16 位或 32 位除以 16 位的操作
6 种址寻方式	为编程和据处理提供更大的灵活性
高度 I/O 单元　4 条专用 I/O 线 4 条可编程 I/O 线	能够产生和测量具有高分辨率(12MHz 时为 2μs)的脉冲
10 位 A/D 转换	读取外部模拟输入
全双工串行口	提供与其它处理器或系统的串行连接
最多 40 个 I/O 口	提供 TTL 兼容的数字数据 I/O，包括采用标准 8 或 16 位外设的系统扩充
可编程的 8 级优先中断系统	响应异步事件
PWM 输出	提供可编程的、占空比可变的脉冲序列，并用来产生模拟输出
监控定时器	提供能力在软件出错或硬件故障时进行恢复
48 脚(双列直插)和 68 脚(扁平封装，引脚格栅阵列)	有多种封装形式可供挑选，可以满足特定应用对 I/O 数目和封装大小的要求

8096 的 16 位 CPU 和所有 I/O 特性以及接口资源都在单块硅片上，它代表了微控制器世界中系统集成度的最高水平。它必将在一些原先需要多块芯片的场合得到应用，

1.2 MCS-96 应用

MCS-96 产品是独立的高性能单片微控制器，可以用于一些有实时要求的复杂场合，象工业控制、仪器和智能计算机外设等。其应用之广泛可涉及所有的工业领域(参看表 1.2)，由于具有高性能 16 位 CPU、高速数学处理能力和高速 I/O，8096 很适合用于复杂的电机控制和转轴控制系统。这样的例子包括三相大功率交流电机和机器人等。

选用10位A/D转换器时，MCS-96可以用于数据采集系统和闭环模拟控制器，由于在单一芯片中包括了模拟和数字I/O处理，它可以具有相当高的系统集成度。

这种芯片非常适合使用仪器产品的领域，比如说气体色层谱仪等，它们既需要模拟处理，又需要高速数字量输出。同样是这些特性，使得它为在导弹导航和控制等空间应用中的理想元件。

表 1.2 MCS-96的广泛应用

工业
电机控制
机器人
离散的和连续的过程控制
数值控制
智能换能器
仪器
医学仪器
液体和气体色层谱仪
示波器
消费品
录象机
光盘驱动器
高级电视游戏
导航和控制
导弹控制
鱼雷导航控制
智能装弹系统
空间导航系统
数据处理
绘图仪
彩色和黑白复印机
温氏盘驱动器
磁带驱动器
打击式和非打击式打印机
远程通讯
调制解调器
智能线路卡控制
汽车
点火控制
传动控制
防滑制动
发射控制

1.3 MCS-96系列开发支持工具

该产品系列得到许多Intel公司的软件和硬件开发工具的支持，这些工具缩短了产品的开发周期，因此使得产品能更快地推入市场，

1.3.1 MCS-96软件开发程序包

提供开发系统支持的8096软件开发程序专门设计用于MCS-96系列单片微控制器，组成该程序包的程序有符号宏汇编程序ASM-96、连接/重定位程序RL-96，以及程序库LIB-96。在各种高级语言中，提供的PLM-96带有浮点数学程序包，还有一些高级语言也正在为MCS-96产品系列开放。

1.3.2 ASM-96宏汇编程序

8096的这种宏汇编程序可以将符号汇编语言指令翻译成机器可执行的目标代码。ASM-96允许程序员采用模块方式编写程序。模块化的程序把十分复杂的程序划分成一个一个较小的功能单元，使得编码、调试和修改都比较容易。各个模块以后可以用RL-96实用程序再连接和定位成为一个程序模块，这一实用程序选出的各个输入目标模块组合成单一输出目标模块，它还给各输入段分配内存，并将各个浮动地址转换成绝对地址，然后，它产生一份打印文件，其中包括连接小结、符号表清单和中间交叉查找清单，另一实用程序LIB-96帮助建立、修改和检查各种库文件，ASM-96运行于Intellec系列Ⅲ或Ⅳ，

1.3.3 PL/M-96

PL/M-96编译程序将PL/M-96语言转译成8096的浮动目标模块。这样做可以提高程序员的工作效率和应用程序的可靠性。这种高级语言的设计和这种机器的体系结构相匹配，因此效率很高，不会因为低效的编码而降低程序员的工作效率，因为该语言和编译程

序均已针对 8096 及其应用环境优化，所以采用 PL/M-96 开发软件冒的风险较少。

1.3.4 硬件开发支持：iSBE-96

iSBE-96 是 MCS-96 产品的硬件执行和调试工具。它包括常驻在 8096 系统中的监控/调试程序。该开发系统通过两根带状电缆与用户 8096 系统连接。一根电缆用于 8096 的 I/O 端口，另一根用于存贮器总线。iSBE-96 通过串行连接接受 Intellec 系列Ⅱ或其它计算机系统控制。iSBE-96 的电源可以通过将其插入 MULTIBUS 插卡槽，或通过外部电源来供给。iSBE-96 组装在一块标准 MULTIBUS 板上。

iSBE-96 提供实时硬件仿真所需的最常用的各种特性。用户可以修改和显示内存、设置断点、带和不带断点执行，以及改变内存映象。此外，用户还可以单步执行系统程序。

1.3.5 MCS-96 工作站

该工作站为设计工程师或系统设计师提供有关 MCS-96 系列的各种经验。整个过程包括对 Intel 8096 体系结构、系统定时和 I/O 设计等方面的介释。控制台会话使得用户可以对 MCS-96 产品系列和支持工具获得深入的了解。

1.3.6 Insite 库

Intel 的 Insite 库含有好几个应用程序，其中一个非常有用的程序就是 8096 的软件模拟程序 SIM-96。它可以实现对用户系统的软件模拟。读模拟程序允许用户设置断点、检查和修改内存、反汇编目标代码，以及单步执行代码等。

1.4 MCS-96 系列产品

尽管 8096 是整个这本手册中用来代表各种 MCS-96 产品的最常用商品器件编号，这一产品系列实际上却是包括有八种配置，因而包括 8096 在内共有八个器件编号。这些性能各异的产品使得用户能够在 I/O 数目和封装尺寸等方面更大地满足应用的要求。任选项有板上 8K 字节掩膜编程存贮器、10 位 A/D 转换器，以及 48 或 68 两种引脚封装形式。

表 1.3 概括了 MCS-96 产品系列中所有各种当前产品。

表 1.3 MCS-96 系列产品

任选项		68脚	48脚
数字 I/O	无 ROM	8096	8094
	有 ROM	8396	8394
模拟和数字 I/O	无 ROM	8097	8095
	有 ROM	8397	8395

48脚型以双列直插封装形式提供

68脚型有两种封装，塑料扁平封装和引脚格栅阵列。

第二章　体系结构

2.0 引言

为了讲述 8096 的操作，可将其划分成几个部分，它们分别是 CPU、可编程高速 I/O 单元、模/数转换器、串行口，以及用于数/模转换的脉冲宽度调制(PWM)输出。除了这些功能单元以外，还有一些部分用来支持芯片总的一些操作，象时钟发生器和反偏发生器等。这一 CPU 和可编程 I/O 和使得 8096 与其它微控制器有很大的差别，因此首先讨论这一 CPU。

2.1 CPU 操作

8096 中 CPU 的主要部分是寄存器堆和寄存器/算术逻辑单元(RALU)。和外部世界的通讯通过各个专用功能寄存器(SFR)或存贮控制器来实现。RALU 并不使用累加器，而是直接在由寄存器堆和 SFR 组成的 256 字节寄存器空间中执行。通过各个 SFR 直接控制 I/O 从而有可能实现高效的 I/O 操作。这种结构的好处是能够快速改变上下文、不存在累加器瓶颈，以及快速吞吐和 I/O 时间。

2.1.1 CPU 总线

一个控制单元和两条总线将寄存器堆和 RALU 连接起来。图 2-1 给出这一 CPU 及其

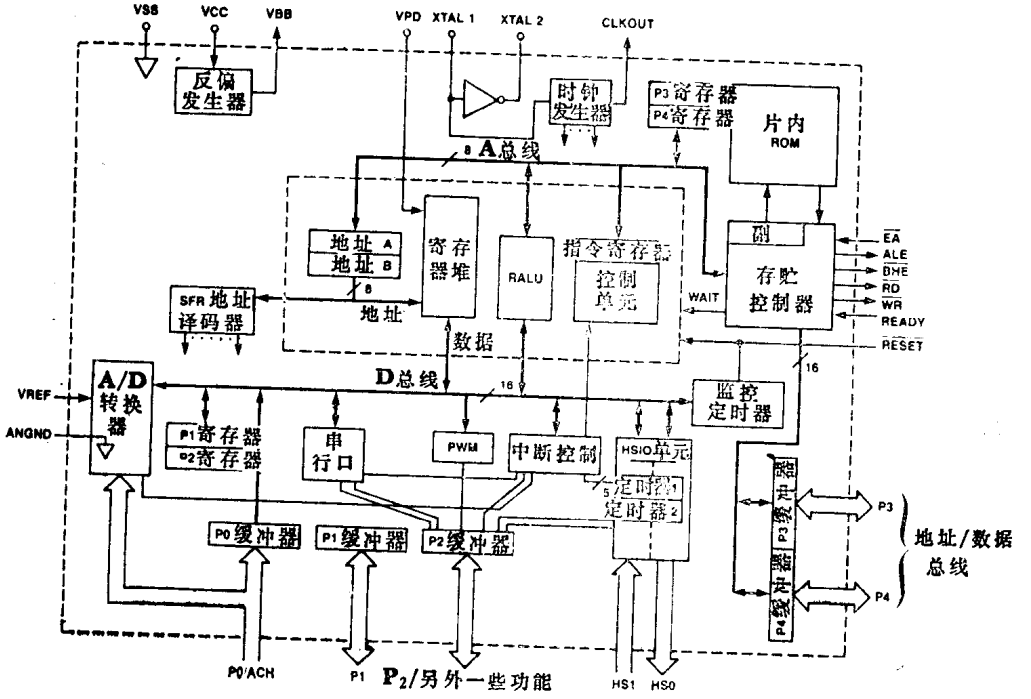

图 2-1　框图(为简化起见，连接端口寄存器与端口缓冲器各线未画出)

主要的总线连接。两条总线是8位宽度的A总线和16位宽度的D总线。D总线仅在RALU和寄存器堆或各个专用功能寄存器之间传送数据。A总线用作上述传送的地址总线，或者用作与存贮控制器相连的多路复用地址/数据总线。无论是对内部ROM还是外部存贮器的访问，都通过存贮控制器进行。

在存贮控制器的内部有一个副程序计数器(副PC)，它随时跟踪着CPU中的PC。由于大多数程序从内存的取指都以副PC为参照，处理机就节省了许多时间，因为很少有地址需要发送给存贮控制器。一旦发生地址转移，副PC中就装入新值，处理继续下去。从内存读取数据同样通过存贮控制器进行，只是副PC在这一操作中一无所事。

2.1.2 CPU寄存器堆

该寄存器堆有232字节的RAM，可以字节、字或双字形式存取。因为这些单元中的每一个都可被RALU使用，所以基本上就有了232个"累加器"。寄存器堆中的第一个字留作堆栈指针用，因此在有堆栈操作发生时不可用它来保存数据。访问寄存器堆和SFR用的地址由CPU硬件暂时贮存在两个8位的地址寄存器中。

2.1.3 RALU控制

给RALU的指令从A总线取得，并暂存于指令寄存器中。控制单元译码这些指令，产生适当的信号序列，使RALU执行需要的各种功能。图2-1中指出了这一指令寄存器和控制单元。

2.1.4 RALU

8096执行的大多数计算操作发生在RALU中。RALU示于图2-2中，它含有一个17

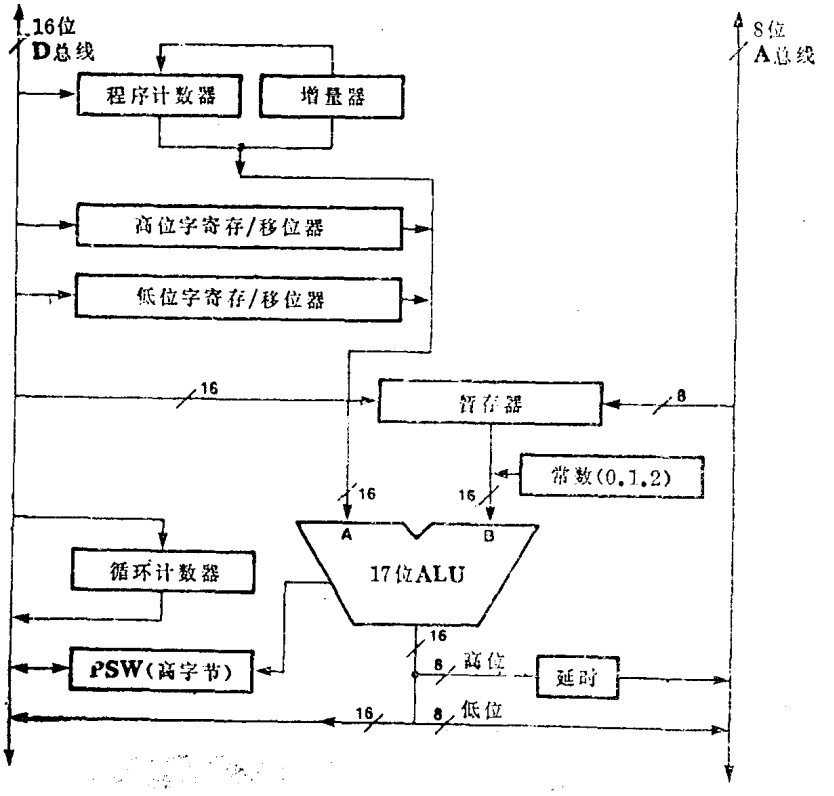

图 2-2 RALU框图

位的 ALU、程序状态字(PSW)，程序计数器(PC)、循环计数器，以及三个暂存器。所有这些寄存器都是 16 位或 17 位(16 位加 1 位符号扩充)宽度。其中有些寄存器能够执行一些简单的操作，减轻了 ALU 的工作负担。

有一个独立的增量器用于 PC 的增量；然而，转移必须通过 ALU 处理。暂存器中有两个具备自己的移位逻辑。这两个寄存器用于需要逻辑移位的一些操作，包括正规化、乘和除等。"低位字"寄存器仅在移位双字量时使用，"高位字"在有任何移位操作时都要用到。在许多指令中，它还用作暂存器。重复的移位由 5 位的循环计数器计数。

有一个暂存器用来贮存双操作数指令中的第二操作数，比如说乘法中的乘数和除法中的除数。在减法操作时，这一寄存器的输出可以在放到 ALU 的"B"端之前先行取补。图 2-2 中的延时单元用来将 16 位总线转换成 8 位总线。需要这一单元的原因是因为所有的地址和指令都在 8 位的 A 总线上传送。为了加快某些计算，有一些常数，象 0、1 和 2，贮存在 RALU 中备用。在 RALU 需要形成2的补码，或者执行增量或减量指令时，这些常数随手可得。

2.2 基本时序

8096 要求输入的时钟频率在 6.0MHz 和 12MHz 之间才能正常工作。这一频率可以直接加至 XTAL1。换一种做法，因为 XTAL1 和 XTAL2 是一个倒相器的输入和输出端，所以有可能采用晶体来产生时钟。振荡器部分的框图示图 2-3 中。该电路的细节及其使用建议可在 4.1 节中找到。

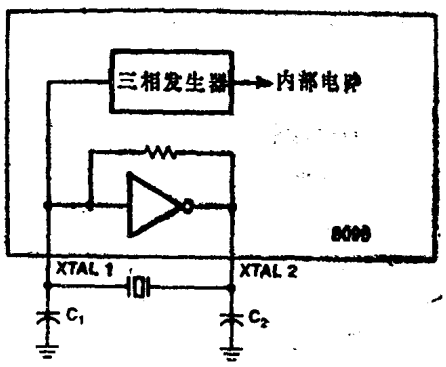

图 2-3 振荡器框图

2.2.1 内部时序

晶体或外部振荡器频率被三分频而产生图 2-4 所示的三相内部时序。每一内部时相经每三个振荡器周期重复一次，因此三个振荡器周期称为一个"状态时间"，它是8096操作的基本时间度量单位。大多数内部操作都和 A、B 或 C 三相之一同步，每一时相都具有三分之一占空比。A 相对外表示成 CLKOUT，这是 68 脚器件上提供的一个信号。B、C

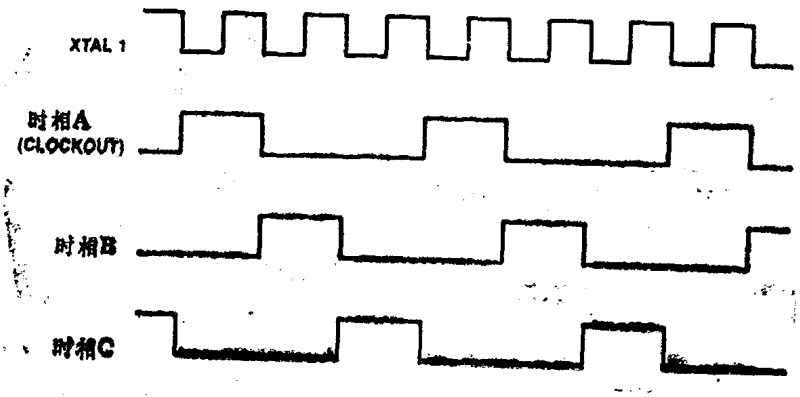

图 2-4 内部时序和 XTAL1 的关系

两相不对外提供。XTAL1、CLKOUT 和 A、B、C 三相的关系示于图 2-4 中。有必要注意的是图中未计及传播延时。这一些时序和其它时序的关系可在 4.1、4.4 和 4.6 节中见到。

RESET 线可以用来在确切的时刻启动8096，从而可以为测试设备和多芯片系统提供同步信号。这一特性的用法在 2.15 和 4.1 节中 RESET 项下有完整的介释。

2.3 存贮空间

8096 的可寻址存贮空间达 64K 字节，其中大部分可供用户作程序或数据内存使用。具有特定功能的一些单元是 0000H 到 00FFH 和 1FFEH 到 2010H。所有其它单元均可用作程序或数据存贮器，也可用作存贮器映象外设。存贮器映象示于图 2-5 中。

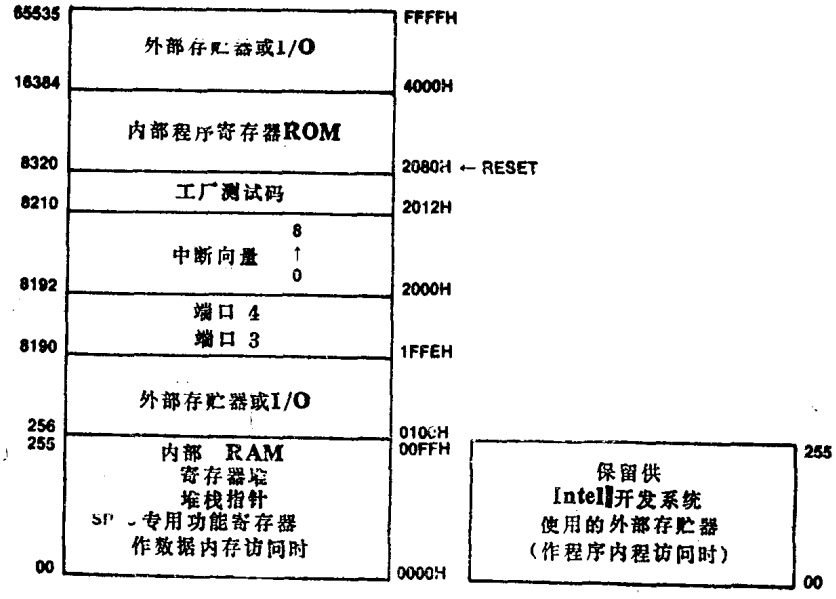

图 2-5 存贮器映象

2.3.1 寄存器堆

单元 00H 到 0FFH 含有寄存器堆和 SFR。有关这一部分存贮空间的完整信息可在“RAM 空间”一节中见到。没有什么代码可以从这一内部 RAM 区执行。如果企图从单元 000H 到 0FFH 中执行指令，该指令就从外部存贮器中读取。这一外部存贮器区域保留供 Intel 开放工具使用。非屏蔽中断(NMI)的执行会强行调用外部单元 0000H，因此，NMI 指令也保留供 Intel 开发工具使用。

2.3.2 保留的存贮空间

单元 1FFEH 和 1FFFH 分别保留作为端口 3 和端口 4 使用。这是为了在系统中使用外部存贮器时可以容易地重构这些端口。重构这些 I/O 端口实例在 4.6.7 节中给出。如果不打算重构端口3和4，那么这些单元可以和任何其他外部存贮单元同样使用。

九个中断向量贮存在单元 2000H 到 2011H。正如 2.5 节中介释的那样，第九个向量供 Intel 开发系统使用。内部单元 2012H 到 2077H 保留用作 Intel 公司的工厂测试代码。外部存贮器中的这些单元仍可供使用。

8096 总清后从单元2 080H 开始取指。选择这一单元是为了使系统接在寄存器堆之后可含有多达 8K 的 RAM。有关总清的进一步信息可在 2.15 节中见到。

2.3.3 内部 ROM

如果在订货时要求配置 ROM，从 2080H 到 3FFFH 的内部存贮单元就被用户指定，单元 2000H 到 2011H 中的中断向量也同时指定。从内部 ROM 读取指令和数据的条件是该器件带有 ROM，$\overline{EA}$ 接至高电平，并且地址在 2000H 到 3FFFH 之间。任何其它时刻的数据，都从内部 RAM 空间或外部存贮器中读取，指令则从外部存贮器中读取。

2.3.4 存贮控制器

RALU 和存贮器的对话(除了在寄存器堆和 SFR 范围内的单元以外)通过存贮器来进行，它通过 A 总线和几条控制线与 RALU 相连。因为 A 总线为 8 位宽度，因此该存贮控制器就利用副程序计数器来工作，不必总是从 RALU 取得指令地址。这一副 PC 在每次取指后增量。发生跳转或调用时，该副 PC 必须在继续取指之前由 A 总线对其重新装载。

除了具有一个副 PC 之外，该存贮控制器还含有一个 3 字节的队列，用来帮助加快执行速度。这一队列对 RALU 和用户透明，除非在外部总线周期中强行插入等待状态。表 3-3 和表 3-4 所示指令执行时间给出的是没有等得状态插入的正常执行时间。重新装载副 PC，然后取得新指令流的首字节，共计占用 4 个状态时间。这一点在表 3-4 中的跳转发生/不发生时间中有所反映。

2.3.5 系统总线

外部存贮器通过 AD0 到 AD15 这几条线编址，它们形成一条 16 位的多路复用(地址/数据)数据总线。这些线的引脚和端口 3 和端口 4 合用。地址锁存允许(ALE)的下降沿用来给透明的锁存器(74LS373)提供时钟，以便对总线作多路分配。一种典型的电路和所需要的时序在 4.6 节中给出。因为8096的外部存贮器可以按字节两种方式寻址，所以译码由总线高位允许($\overline{BHE}$)和地址/数据线 0(AD0)这两条线控制。$\overline{BHE}$ 线必须作透明性锁存，就和地址的做法完全一样。

为了避免在介释存贮系统时发生混淆，给多路分配后的地址/数据信号起名是合理的做法。这些地址信号将称为 MA0 到 MA15(M 指存贮器，A 指地址)，数据信号则称为 MD0 到 MD15(M 指存贮器，D 指数据。)

在 $\overline{BHE}$ 有效(低电平)时和数据总线高位字节所连之存单元应被选中。在MA0 为低电平时，和数据总线低位字节相连之存贮单元被选中。采用这种方式来访问16位宽度的存贮器，可以仅指向低位(偶数)字节(MA0 = 0，$\overline{BHE}$ = 1)，仅指向高位(奇数)字节 (MA0 = 1，$\overline{BHE}$ = 0)，或同时指向这两个字节(MA0 = 0，$\overline{BHE}$ = 0)如果一个存贮块仅用于读时，$\overline{BHE}$ 和 MA0 不需要译码。

图 2-6 给出有关下面这些外部存贮器操作的理想波形。确切的时序性能请参看最后的数据表。在外部存贮器取指开始时，地址锁存允许线(ALE)电平升高，地址送上 AD0-AD15，并且 $\overline{BHE}$ 设置成需要的状态。然后 ALE 下降，地址从引脚上取走，并且 $\overline{RD}$(读)信号降低。在这种情优下，READY 线可能被拖低来保持住处理机达几个附加的状态时间。

2.3.6 READY

在上述情优中，READY 线可用来保持处理机，以便访问慢速的存贮器，或者用于

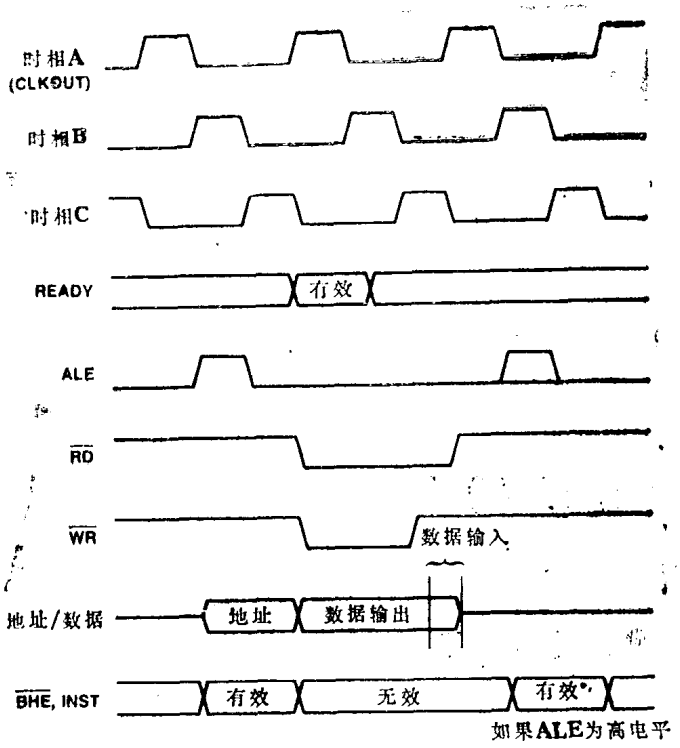

图 2-6 外部存贮器时序

DMA。对 READY 线的采样从内部讲发生在时相 2 中，这是一个产生 CLKOUT 的信号。在 CLKOUT 变成低电平之前有一段 READY 必须保持稳定的最小时间。如果在这一部分正要进入非就绪状态时，这一置位时间未能得到保证，该器件的操作就会变得无法确定。

因为 READY 与 CLKOUT 同步，8096 处于未就绪状态的时间有可能长达 CLKOUT 的几倍，尽管 READY 线在任何时刻都有可能转为高电平。存在一个保持 8096 于未就绪状态的最大时间，典型值在 1μS 的数量级。确切的数值依特定的器件和要求的温度范围而异，可参阅数据表。来自外部存贮器的数据送上总线并趋于稳定的时刻必须在 RD 的上升沿之前一段时间，这段时间的最小值就是建立时间(set-up time)。$\overline{RD}$ 的上升沿将信息锁存在 8096 中。如果读取的是数据，在地址有效时 INST 引脚就处于低电平；如果是指令，这段时间中 INST 引脚就处于高电平。

写入外部存贮器的时序要求和从外部存贮器读时的要求相似。主要的差别就在用到的是写($\overline{WR}$)信号，而不是读($\overline{RD}$)信号。在 $\overline{WR}$ 线的下降沿以前的时序都相同，在 $\overline{WR}$ 下降时 8096 取走地址而将数据送上总线。如前所述，此时 READY 线必须保持在需要的状态。在 $\overline{WR}$ 线趋于高电平时，数据应该锁存进外部存贮器在写过程中，INST一直处于低电平，因为不能写指令。存贮器访问的确切时序性能可在数据表中查到。

2.4 RAM 空间

8096 的内部寄存器单元分成两组，一组是寄存器堆，另一组是专用功能寄存器组(SRFs)。RALU 能够操作这 256 个内部寄存器单元中的任何一个。单元 00H 到 17H 用来访问 SFRs。单元 18H 和 19H 存放堆栈指针。剩下的 230 个单元的使用没有什么限制，只是不能从其中取出代码来执行。

2.4.1 专用功能寄存器组

8096 中所有的 I/O 都通过 SFRs 来控制。其中许多寄存器具有双重功能。一种功能在它们被读取时起作用，另一种在写入时起作用。图 2-7 给出这些寄存器的地址和名称。

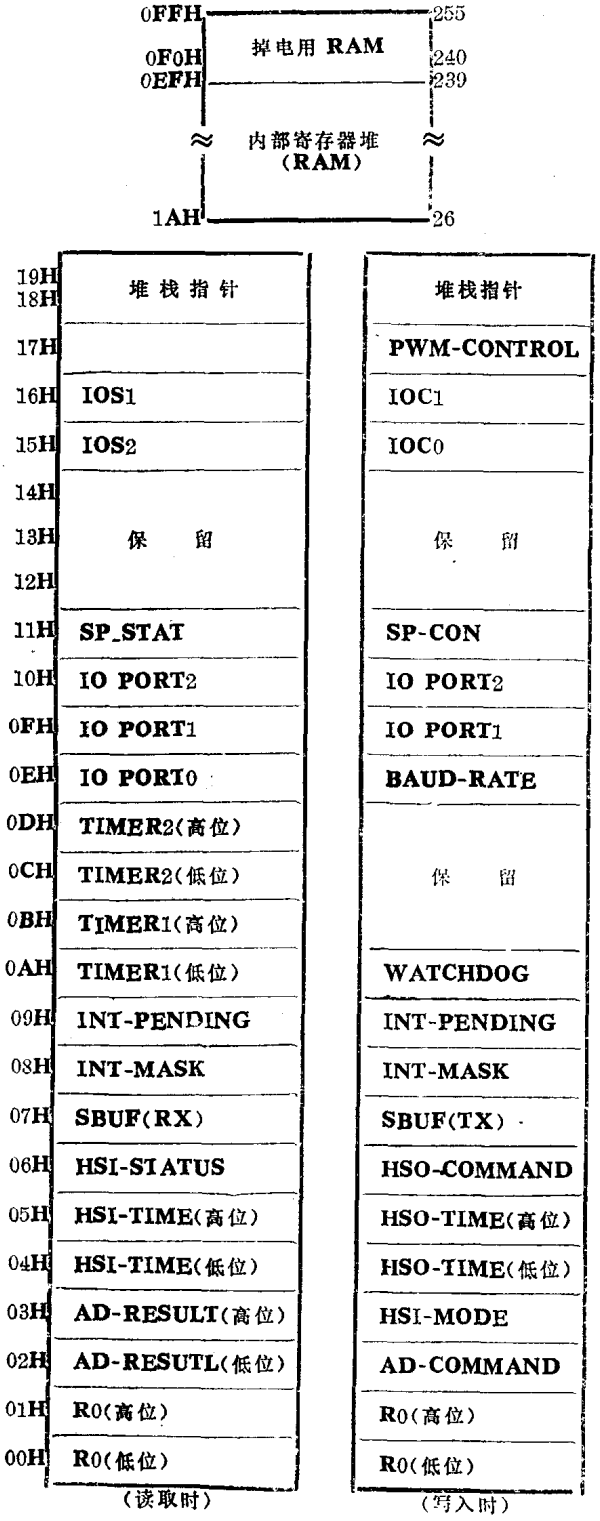

图 2-7 寄存器堆存贮器映象

每一个寄存器的作用小结于图 2-8 中，后面的章节中还有更完整的描述。

在 SFR 空间中有几个寄存器标记着“保留”字样。这些寄存器保留用于今后的扩充和测试目的。读写这些寄存器会产生不希望有的结果。比如说，写至单元 000CH 会将两个定时器都置成 0FFFXH，该特性在测试这一器件时使用，程序中不应该用它。

2.4.2 掉电

上面的 16 个 RAM 单元(0F0H 到 0FFH)从 VCC 和 VPD 这两个引脚获得电源。如

寄存器	描述	章节
RO	零寄存器——总是读作零，变址时作基址用，计算和比较时作常数用。	3.2.7
AD-RESULT	A/D 结果——A/D 转换的结果	2.9.3
AD-COMMAND	A/D 命令寄存器——控制 A/D	2.9.2
HSI-MODE	HSI 方式寄存器——设置高速输入单元的工作方式。	2.7.1
HSI-TIME	HSI 时间——保存高速输入单元触发的时间。	2.7.4
HSO-TIME	HSO 时间——设置高速输出执行命令寄存器中命令的时间。	2.8.3
HSO-COMMAND	HSO 命令寄存器——确定在 HSO 时间寄存器中装载的时间值到来时将发生什么。	2.8.2
HSI-STATUS	HSI 状态寄存器——指出在 HSI 时间寄存器中装载的时间值到来时哪一个 HSI 引脚被检测。	2.7.4
SBUF(RX)	串行口接收缓冲器，	2.11
SBUF(TX)	串行口发送缓冲器，	2.11
INT-MASK	中断屏蔽寄存器——允许或禁止各个中断。	2.5.2 3.6.2
INT-PENDING	中断悬挂寄存器——指明何时已有一个中断信号在诸源之一发生。	2.5.2 3.6.2
WATCHDOG	监控定时寄存器——周期性写入，防止发生每 64K 状态时间一次的自动总清。	2.14
TIMER1	定时器 1	2.6.1 2.7-8
TIMER2	定时器 2	2.6.2 2.7-8
IOPORT0	端口 0 寄存器——端口 0 各引脚电平	2.12.1
BAUD-RATE	保存波特率的寄存器，该寄存器顺序装入。	2.11.4
IOPORT1	端口 1 寄存器——用于端口 1 读写。	2.12.2
IOPORT2	端口 2 寄存器——用于端口 2 读写。	2.12.3
SP-STAT	串行口状态——指明串行口的状态。	2.11.3
SP-CON	串行口控制——用来设置串行口的工作方式。	2.11.1
IOS0	I/O 状态寄存器 0——含有 HSO 状态信息。	2.13.4
IOS1	I/O 状态寄存器 1——含有定时器和 HSI 的信息	2.13.5 3.7.2
IOC0	I/O 控制寄存器 0——控制将 HSI 各引脚改换成另一种功能，如定时器 2 复位源和时钟源。	2.13.2
IOC1	I/O 控制寄存器 1——控制将端口 2 各引脚改换成另一种功能，如定时器中断和 HSI 中断。	2.13.3
PWM-CONTROL	脉冲宽度调制控制寄存器——设置 PWM 脉冲间期	2.10 4.3.2

图 2-8 SFR 小结

希望在发生掉电时仍保持这些单元的存贮内容有效，所需要做的只是保持 VPD 引脚的电压。保持这些 RAM 单元的内容有效所需要的电流大约是 1 毫安(确切的数值参阅数据表)。

为使 8096 进入掉电工作方式，$\overline{\text{RESET}}$ 引脚变为低电平。2 个状态时间以后该器件就会复位。有必要在掉电的瞬时控制不让写入 RAM。这样就有可能从 VCC 引脚撤去电源。而 VPD 引脚必须仍在规定的电压范围内。8096 可以维持在这种状态下达任意长时间，并且这 16 个 RAM 字节能保持其值不变。

为使 8096 脱离掉电工作方式，在 VCC 加电时 $\overline{\text{RESET}}$ 要保持在低电平。在振荡器和反偏发生器工作稳定后两个状态时间(大约 1ms)，$\overline{\text{RESET}}$ 引脚可以变为高电平。在 $\overline{\text{RESET}}$ 电平升高后 10 个状态时间，8096 将开始从单元 2080H 起执行代码。图 2-9 给出了掉电过程的时序图。为了确保 2 个状态时间的最小复位时间(与 CLKOUT 同步)能够满足，推荐采用 10 个 XTAL1 周期。有关实际硬件连接方面的建议在 4.1 节中给出。复位在 2.15 节讨论。

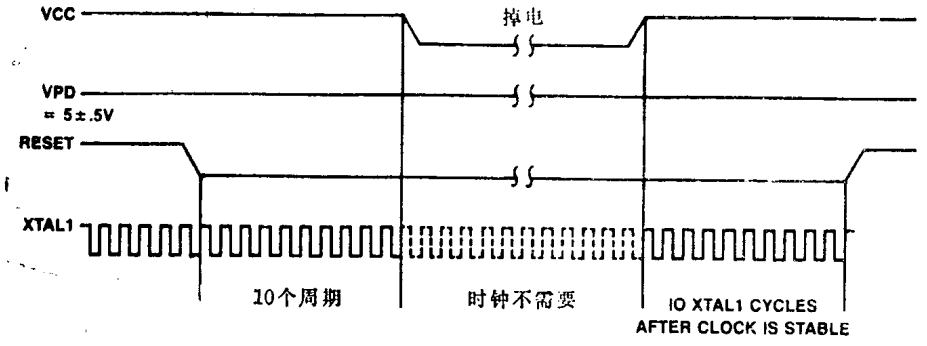

图 2-9 掉电时序

2.5 中断结构

2.5.1 中断源

8096有八个中断源。允许中断时，发生于这些中断源中的任何一个中断都会引起对地址贮存在该中断源的向量单元中的单元的一次调用。这些中断源及其各自的向量单元列于图 2-10 中。除了这八个标准的中断外，有一条 TRAP 指令可以起软件产生的中断的作用。这一条指令目前没有得到 MCS-96汇编程序的支持，仅仅保留供 Intel 开发系统使用。这些中断源中有许多都可以用几种方式获得，图 2-11 给出这些中断的所有各种可能来源。

中断源	向量单元		优先级
	(高位字节)	低位字节)	
软件	2011H	2010H	不可申请
外部中断	200FH	200EH	7(最高级)
串行口	200DH	200CH	6
软件定时器	200BH	200AH	5
HSI.0	2009H	2008H	4
高速输出端	2007H	2006H	3
HSI 数据可用	2005H	2004H	2
A/D转换完成	2003H	2002H	1
定时器溢出	2001H	2000H	0 (最低级)

图 2-10 中断向量单元

2.5.2 中断控制

中断系统框图示于图2-12。每一个中断源都要检测是否有 0 到 1 的跳变。如果这一跳变发生，地址为 0009H 的中断悬挂寄存器

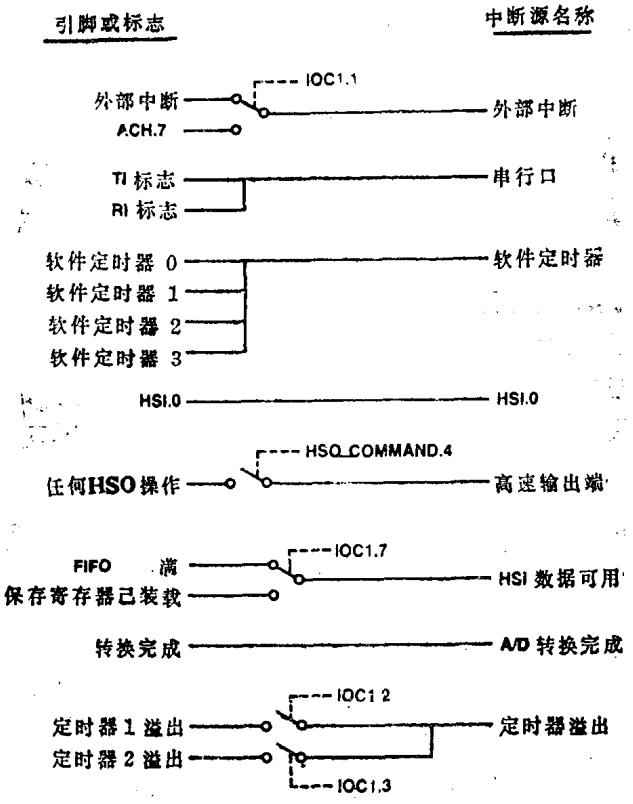

图 2-11 所有可能的中断源

中相应的位置位。因为这一寄存器是可写的，所以就有可能通过在这一寄存器中置位来产生软件中断，也可以通过将该寄存器中的某些位清除来撤消悬挂着的中断。即使在禁止中断时，这一悬挂寄存器仍可置位。

在写悬挂寄存器清除中断时必须谨慎。如果在该位清除时，中断已开始响应，就会有一个 4 个状态时间的“部分”中断周期发生。这是因为8096将不得不去取正常指令流的下一条指令，而不是如同它原来要做的那样继续处理中断过程。对程序的影响基本上就等于加入了一个 NOP。避免发生这种情况的方法是利用双操作数立即逻辑指令来清除这些位，因为 8096 在执行这些“读/改/写”指令时不会马上响应中断。

一个中断的允许与禁止通过地址为0008H 的中断屏蔽寄存器来实现。如果该寄存器的某一位为 1，相应的中断就被允许，否则就被禁止。即使一个中断处于屏蔽状态，它仍旧可以被悬挂。因此在解除对一个中断的屏蔽之前，或许有必要先清除其悬挂位。

中断屏蔽寄存器同时也是 PSW 的低位字节。利用“EI”(允许中断)和“DI”(禁止中断)指令同时允许或禁止所有的中断。EI 和 DI 置位或清除 PSW9，也就是中断允许位，它们并不影响屏蔽寄存器的内容。

2.5.3 中断优先级编程

优先级编码器检测出所有既已被悬挂又是允许的中断，挑选出其中具有最高优先级的一个来。各优先级示于图 2-10 中(7 是最高级，0 是最低级)。中断发生器于是对地址由中断向量指定的单元发出调用。这应单元应该是中断服务例程(ISR)的起始单元。