IOSP

操作系统处理器

《舰船导航》编辑部

一九八五年十二月

目　录

80130/80130-2iAPX86/30、88/30、186/30、188/30 iRMX86操作系统处理器

●包含操作系统原语的高性能两片式数据处理器。

●标准的iAPX86/10、88/10指令系统加上任务管理，中断管理、信息传递、同步和存贮器定位原语。

●可全面扩充到iRMX86并与之兼容。

●支持五种操作系统数据类型：作业（JOB）、任务（TASK）、段(SEGMENT)、邮箱（MAILBOX）、域（REGION）。

●35种操作系统原语。

●内部操作系统定时器和从8级扩充到57级的中断控制逻辑。

●与8MHZ的8086/80150/80150-2/8088/80186/80188兼容，无需等待状态。

●多总线兼容接口。

Intel公司的iAPX86/30和88/30是一种两片式微处理器，它提供通用的CPU(8086)指令而又具有实时操作系统的支持，为多道程序运行和多任务应用奠定了基础。iAPX86/30由一片iAPX86/10(16位的8086CPU)和一片操作系统固件OSF（80130）组成，iAPX88/30由OSF和iAPX88/10（8位的8088CPU）组成，其中8086或8088可以由80186或80188来代替。

86/30和88/30都是N沟道、耗尽型、硅栅工艺（HMOS）组件，封装于40脚的管壳中。它们提供iAPX86/10、88/10处理器的全部功能，再加上35种操作系统原语、8级中断支持硬件、一个系统定时器、一个延时定时器以及一个波特速率产生器。

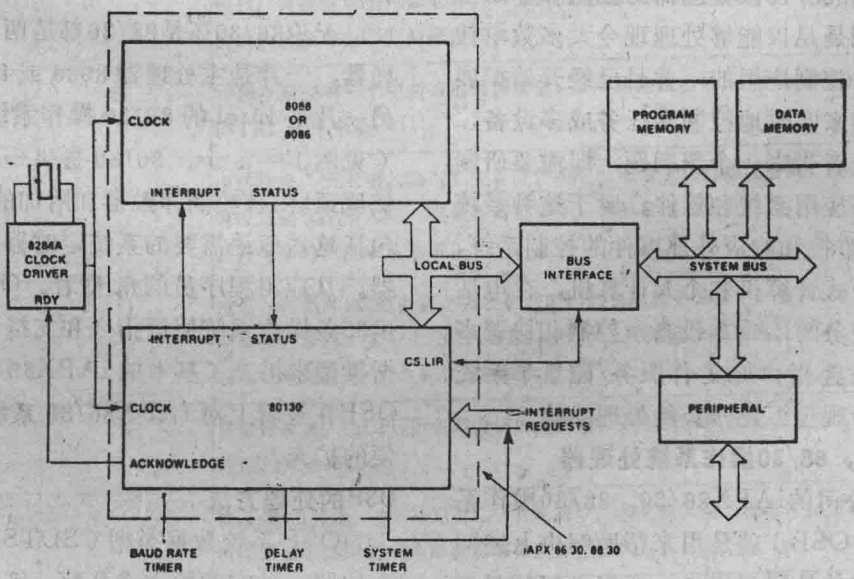

图1—1 iAPX86/30、88/30方框图

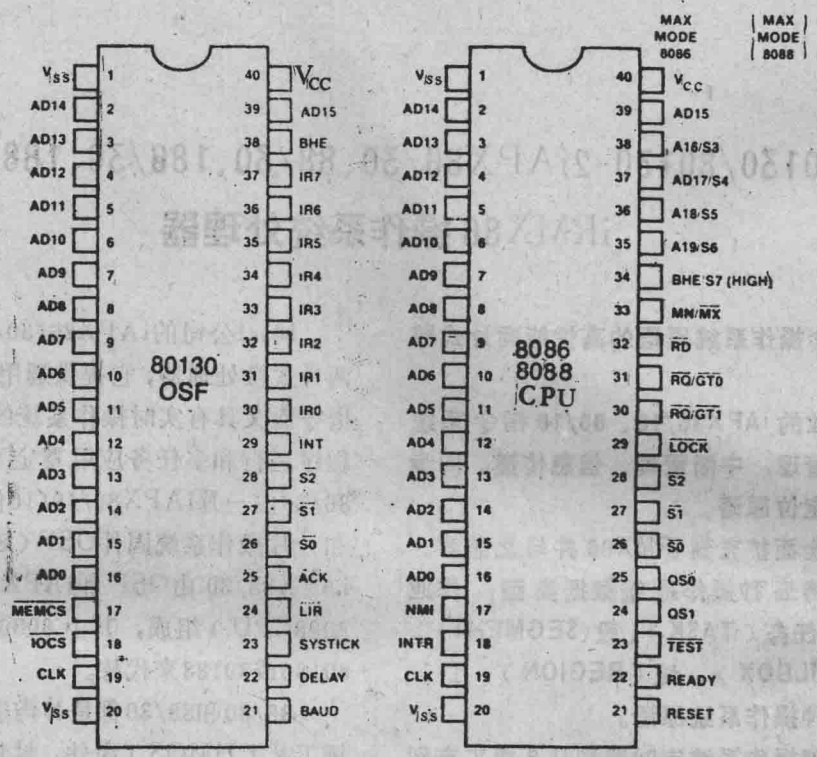

图1—2 iAPX86/30、88/30引脚排列

功能说明

86/10和88/10微处理器的性能和存贮空间已经证明是足以能够处理现今大多数单任务或单设备控制应用的，并且已经开始扩展它们的应用来实时地控制多任务或多设备，这就给设计者带来一个新问题，即需要研制实时多任务应用系统和软件。属于这种系统的有：实时监控和响应外部事件的控制系统，多功能的台式计算机和个人计算机，在电话局中管理话务的PABX设备，控制和协调多磁盘与多磁盘用户的文件服务/磁盘子系统以及象电子现金汇兑一类的处理系统。

iAPX86/30，88/30操作系统处理器

Intel公司的iAPX86/30，88/30操作系统处理器（OSP）就是用来帮助解决上述问题的，其目的是通过提供一套定义好的且调试好的操作系统原语来简化多任务应用系统的设计，这些原语可直接在硬件中运行，免除了程序员设计多任务操作系统原语的负担。

无论86/30还是88/30都是两片一套的处理器，一片是主处理器8086或8088CPU，另一片是Intel的80130操作系统固件OSF（见图1—1）。80130提供一套多任务内核原语、内核控制存贮器和附加的支持硬件，包括这些原语需要的系统定时器、中断控制器。从应用程序员的角度看，OSF通过提供35条操作系统原语指令和支持5种新的数据类型来扩充了基本的iAPX86/88结构，使OSF在逻辑上对iAPX86/88系统实现了方便的扩充。

OSP的处理方法

OSP系统数据类型（SDTS）和原语指令按照一种有效的方式分配、管理并共享低级的处理器资源，例如，OSP执行86/10或

表1—1 80130引脚说明

符　号	类　型	名 称 和 功 能
AD15—AD0	I/O	地址数据线：这16条引线构成分时传输的存贮器地址（T1）和数据（T2，T3，TW，T4）总线，高电平有效。对于一个被调用的原语来说，如果MEMCS或IOCS有效，则在总线周期的T1状态将地址内部锁存并转换成80130的内部地址。只要不被片选，80130的这些引脚就是浮置的，仅在读周期的T2—T4及INTA周期的T1期间80130才驱动这些引脚。
$\overline{BHE}$/S7	I	总线高位使能：80130根据来自处理器的$\overline{BHE}$信号来决定是响应高位字节还是低位字节，或者高低字节均响应。此信号为低电平有效，80130在ALE信号的后沿将$\overline{BHE}$锁存。80130的输出数据与$\overline{BHE}$的关系为 $\overline{BHE}$　A0 0　0　AD15—AD0为一字 0　1　AD15—AD8为高字节 1　0　AD7—AD0为低字节 1　1　AD7—AD0为高字节
$\overline{S_2}\overline{S_1}\overline{S_0}$	I	状态：对80130而言，状态线引脚仅作输入，其编码如下： $\overline{S_2}\overline{S_1}\overline{S_0}$ 0　0　0　INTA 0　0　1　IORD 0　1　0　IOWR 0　1　1　无效 1　0　0　取指 1　0　1　MEMRD 1　1　×　无效
CLK	I	时钟：系统时钟为处理器和总线控制提供基本定时。CLK是不对称的，占空比为33%，可提供最佳内部定时。80130使用该系统时钟作为SYSTICK和BAUD定时器的输入以求与主CPU的操作同步。
INT	O	中断：只要有一个中断请求保持有效，INT就是高电平。一般将INT连到CPU的INTR，用来中断CPU
IR7—IR0	I	中断请求：当IR输入端由低变高时产生中断请求（边缘触发方式，要求在中断被响应之前该IR输入端一直保持高电平）。如果是电平触发方式，则只要IR输入端是高电平就产生中断请求。
$\overline{ACK}$	O	应答：每当80130的资源正在被访问时，此引脚就是低电平。当80130正在提供中断向量信息时，第一个INTA和第二个INTA期间它也是低的。$\overline{ACK}$也可用作总线准备就绪应答与/或总线收发器控制信号。
$\overline{MEMCS}$	I	存贮器片选：当CPU正在取内核的原语时，此引脚必须被驱动为低电平，用AD13—AD0来选择指令。
$\overline{IOCS}$	I	输入/输出片选：在IORD和IOWR周期中，若此输入端为低电平，则表明80130内核原语正在访问某一外围功能部件，如下表所示：

		$\overline{BHE}$ A3 A2 A1 A0 0 × × × × 无效 × × × × 1 无效 × 0 1 × × 无效 1 0 0 × 0 中断控制器 1 1 0 0 0 系统信号定时器 1 1 0 1 0 延时计数器 1 1 1 0 0 波特速率定时器 1 1 1 1 0 定时器控制
$\overline{LIR}$	0	局部总线中断请求：当此中断请求连到非从输入端或连到一个编程为局部从的从输入端时，$\overline{LIR}$是低的。
Vcc		电源：+5V电源引脚。
Vss		地
SYSTICK	O	系统时钟信号：定时器0的输出，操作系统时钟基准。SYSTICK一般接到IR_2，实现操作系统定时中断。
DELAY	O	延时定时器：定时器1的输出，英特尔公司为将来的应用所保留。
BAUD	O	波特速率产生器：与8254Mode 3兼容的输出，80130定时器2的输出。

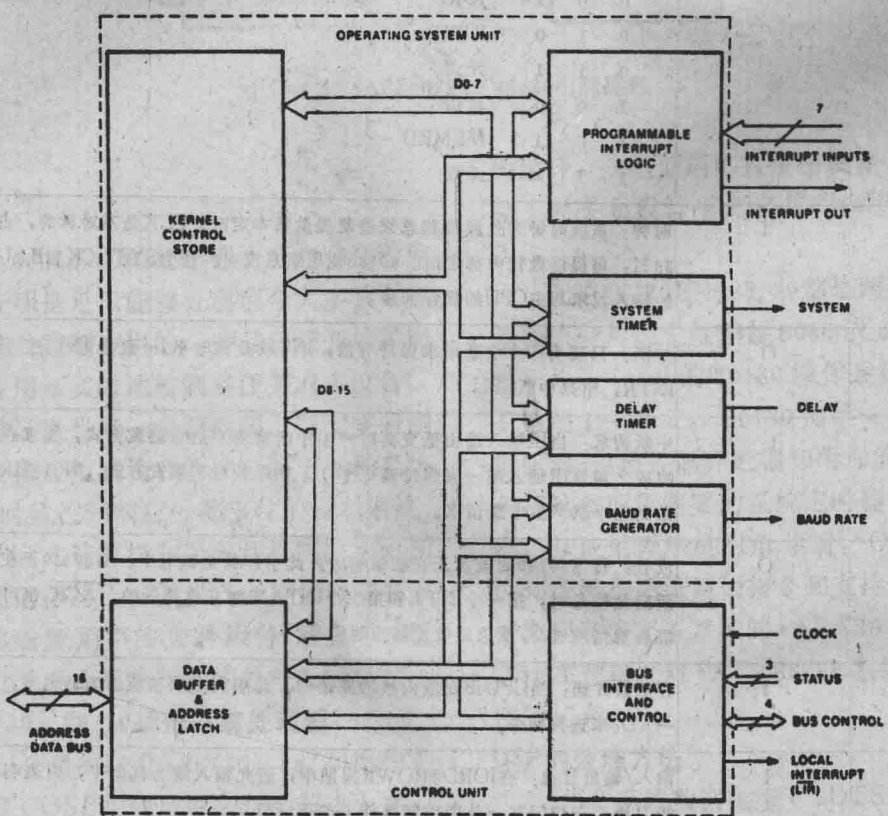

图1—3　OSF内部框图

86/20（8086+8087）的任务上下文关系管理（管理由硬件寄存器组和软件控制信息组成的任务状态映象）。无论任务是否正在执行，OSP都要管理整个任务的状态映象，可以生成任务，在规定的周期内将其置于静止状态或挂起，也可以执行任务，而当任务的功能完成之后，又可以动态地将它们删除。

操作系统处理器支持面向事件的系统设计。每一事件可以由一个单独响应的任务来处理，也可以与某个公共任务中其他密切相关的事件一起来处理。当外部事件和中断出现时，OSP中断处理器原语使用其内部中断控制器子系统实时地进行处理。多任务和多事件由OSP总调度程序进行协调，其抢先的以优先权为基础的调度算法和系统定时器组织并监控每一任务的处理，保证按事件的重要性有秩序地对它们进行处理。86/30还为任务间通信（通过邮箱）和互斥（通过域）提供原语，且具有多任务应用的基本功能。

编程语言支持

OSP的程序可以用Intel公司的iAPX 86，88系统标准语言，即ASM86/88或PL/M86/88来编写。

操作系统处理器支持包（iOSP86）为用任何一种PL/M86方式编写的应用程序提供接口库文件，该库还提供80130生成、初始化以及完整的用户文件编制方法。

OSF编程接口

OSF提供35种操作系统内核原语，它们执行多任务运行、中断管理、空闲存贮器管理、任务间通信以及同步。表1—4列出了每条原语，表1—5表示典型原语的执行特点。

OSP原语由CPU和OSF（80130）共同执行，当一个应用程序任务调用某OSP原语时，iAPXCPU的寄存器和堆栈被用来完成适当的功能并将结果转送到应用程序。

OSP原语调用序列

用标准的基于堆栈的调用序列来调用OSF原语，在调用一条原语之前，必须将其操作数参数放入任务堆栈，在SI寄存器中装入堆栈里最后一个参数的偏移量，并将该原语的入口编码装入AX，原语的调用由CPU的软件中断来实现（表1—4），图1—4表示调用原语的一个典型的ASM86序列。在PL/M中OSP程序员用调用语句来调用该原语。

```
                SAMPLE ASSEMBLY LANGUAGE PRIMITIVE CALL
PUSH P1                     ; PUSH PARAMETER1
PUSH P2                     ; PUSH PARAMETER2
  .                         ; . . .
  .                         ; . . .
  .                         ; . . .
PUSH PN                     ; PUSH PARAMETER N
PUSH BP                     ; STACK CALLING CONVENTION
MOV BP,SP
LEA SI,SS:NUM__BYTES__PARAM-2[BP]
                            ; SS:SI POINTS TO FIRST
                            ; PARAMETER ON STACK
MOV AX, ENTRY CODE          ; AX SETS PRIMITIVE ENTRY CODE
INT184                      ; OSF INTERRUPT
                            OSP PRIMITIVE INVOKED
POP BP
```

```
RET NUM_BYTES_PARAM_ ; POP PARAMETERS
                     ; CX CONTAINS EXCEPTION CODES
                     ; DL CONTAINS PARAMETER NUMBER
                     ;  THAT CAUSED EXCEPTION ( IF
                     ;  CX IS NON ZERO )
                     ; AX CONTAINS WORD RETURN VALUE
                     ; ES : BX CONTAINS POINTER
                     ;  RETURN VALUE
```

图 1—4　ASM/86 OSP调用规约

OSP功能说明

下面说明OSP的每一种主要功能，包括：Job和Task管理；中断管理；空闲存贮器管理；任务间通信；任务间同步；环境控制。

在说明中，由OSP所支持的系统数据类型（SDT）的第一个字母都大写，每种SDT的简要说明见表1—2

JOB和TASK管理

每个OSP JOB都是一个受控的环境，在该环境中执行应用程序并驻留OSF的系统数据类型。每个应用程序通常都是一个独立的OSP JOB，不管它有一个初始任务（最小值）还是多任务。JOB将系统存贮器划分为池（Pool），每存贮池提供一个存贮区域，在该区中OSP将分配由正在执行的TASK所产生的TASK状态映象及其他系统数据类型和TASK运行空间的空闲存贮器。OSP通过管理每一个任务所使用的资源来支持在一个JOB中执行多任务，这些资源包括CPU寄存器，NPX寄存器，堆栈，存贮数据类型以及可用的空闲存贮空间。

在生成一个TASK时，OSP要从其JOB环境的空闲存贮器中为它分配任务堆栈和数据区存贮器，并予置附加的任务属性，如任务的优先权等级和其出错处理程序单元（作为一种选择，调用CREATE TASK的程序可以事先给任务指定一定的堆栈和数据区）。任务优先权是0—255之间的一个整数（优先权号数越低，其任务的调度优先权就越高），通常，从128往前的优先级都安排给处理中断的那些任务；128往后的优先级不产生要被禁示的中断，这些优先级(128—225)适合于非中断任务。如果使用8087数值处理器扩充，分配给它的出错恢复中断级将比其上执行的任务的优先级更高，以便能正确地完成出错处理。

执行状态

任务有执行状态，OSP提供5种执行状态，它们是：运行（RUNNING），准备就绪（READY），静止（ASLEEP），挂起（SUSPENDED）和静止—挂起（ASLEEP-SUSPENDED）。

• 如果一个任务拥有处理器的控制，它就处于运行状态。

• 如果不是静止、挂起或静止—挂起，则该任务处于准备就绪状态。对于一个要成为运行（或执行）状态的任务来说，它必须是准备就绪状态中最高优先级的任务。

• 如果一个任务正在等待将要被认可的请求或定时事件的到来，则它处于静止状态。任务可以置本身为静止状态。

• 如果一个任务被另一个任务搁置起来或它自我停止，则它处于挂起状态。一个任务可以有多重挂起，挂起计数由OSP当作任务挂起深度来管理。

• 如果一个任务既在等待请求，又是挂起，则它处于静止—挂起状态。

任务属性，CPU寄存器的值，8087寄

存器的值（如果应用中配置了8087的话）均由OSP保留在任务的状态映象中，每个任务有一独立的任务状态映象。

调度

OSP根据优先等级将处理器时间分配给各任务，一个任务具有与系统中所有其他任务相对应的执行优生级，OSP为每个任务在其状态映象中保留执行优先级。当一个比正在执行的任务有更高优先权的任务准备好执行时，OSP将处理器的控制转移到更高优先权的任务。首先，OSP将退出任务（较低优先级的任务）的状态，包括CPU寄存器的值，保留在其任务状态映象中，然后，根据即将到来的（较高优先权的）任务状态映象重置CPU寄存器，最后它让CPU开始或恢复执行该较高优先级的任务。

任务的调度由OSP完成，OSP面向优先权的优先调度程序通过比较相应的优先权来决定执行哪一个任务，调度程序保证让准备就绪状态的最高优先级任务执行。在更高优先级的中断出现之前，只要不请求不可得到的资源，或者在它使得一些特殊的资源为一个更高级的正等待着这些资源的任务得到之前，此任务将一直继续执行。对于不可得到的资源，该任务将乐意等待下去。

通过接收一个信息，接受一种控制或一个中断以及定时输出，任务可转入准备就绪状态。OSP总是监督着系统中所有任务（和中断）的状态，仅当把CPU的资源专用于被执行的任务上时，抢先的调度才允许敏感外部环境。

定时等待

OSP定时器硬件支持定时等待和超时，因此在许多原语中，任务可以指定一个时间长度，这段时间用来等待事件的出现，等待能得到所需要的资源或从邮箱中接收一个信息。定时间隔（或系统定时）可以调整，下限为1 ms。

执行任务的应用控制

程序可以动态地改变任务的执行状态和优先级。一个任务可以在一段时间内使自身挂起，也可挂起另一任务的执行，过后再将其恢复。还可提供多重挂起，一个被挂起的任务可以再次挂起。

OSP的JOB和TASK管理原语如下：

CREATE JOB 划分系统资源并生成任务执行环境。

CREATE TASK 生成任务状态映象，指定任务代码指令流的地址、任务执行优先级和一些别的属性。

DELETE TASK 删除任务状态映象，撤消指令流的执行，废弃分配的堆栈资源，但不取消中断任务。

SUSPEND TASK 挂起指定的任务，如果任务原已挂起，则挂起深度加1，执行状态是SUSPEND。

RESUME TASK 任务深度减1，如果挂起深度是0，则此原语将任务的执行状态改变为READY或ASLEEP（当ASLEEP/SUSPEND时）。

SLEEP 在指定的系统定时内把提出请求的任务置于ASLEEP状态（该定时间隔可以配置到1 ms）。

SET PRIORITY 改变任务的优先级别。

中断管理

OSP支持256级中断（按中断矢量编排）和57种外部中断源，其中一种是NMI（非屏蔽中断）。OSP独立地管理每级中断，其中断系统为中断管理提供了两个机构：中断处理程序(INTERRUPT HANDER)和中断任务(INTERRUPT TASK)。中断处理程序阻塞所有的可屏蔽中断，且仅为需要很少处理时间的在服务中断所使用。在中断处理程序中，只可以使用一定的OSF中断管

理原语(DISABLE, ENTER INTERRUPT, EXIT INTERRUPT, GET LEVEL, SIGNAL INTERRUPT)和基本的CPU指令，其他的OSF原语不能用。中断任务允许发出全部OSP原语，且仅仅屏蔽较低优先权的中断。

分配到同一级别的中断处理程序和中断任务之间的工作流程由SIGNAL INTERRUPT和WAIT INTERRURT原语来调整。该流程是异步的，当中断处理程序给中断任务发信号时，它立即就可用来处理另一个中断。中断处理程序为中断任务排列的中断数目(级数)被限制到SET INTERRUPT原语中指定的值。当中断任务完成处理时，它发出WAIT INTERRUPT原语，并且立即作好了处理这一中断队列的准备，此中断队列是当中断任务还在执行的时候由中断处理器用重复的SIGNAL INTERRUPT原语建立的。假如在这一级没有中断，则队列是空的，而中断任务呈挂起状态，请参阅图1-5的例子和图1-6，1-7。

OSP的外部中断级直接与内部的任务调度优先权相对应，OSP保留着一个包含有任务和中断级别的优先权表格，正在执行中的任务的优先级自动地确定哪一些中断要被屏蔽。中断由中断级别编号来管理，OSP直接支持8级，使用从8259A总共可以扩充到57级。

```
/* CODE EXAMPLE A INTERRUPT TASK TO KEEP TRACK OF TIME-OF-DAY
DECLARE SECOND$COUNT BYTE,
   MINUTE$COUNT BYTE,
   HOURS$COUNT BYTE,
TIME$TASK: PROCEDURE,
   DECLARE TIME$EXCEPT$CODE WORD,
   AC$CYCLE$COUNT=0;
   CALL RQ$SET$INTERRUPT(AC$INTERRUPT$LEVEL, 01H),
      @AC$HANDLER, 0, @TIME$EXCEPT$CODE),
   CALL RQ$RESUME$TASK(INIT$TASK$TOKEN, @TIME$EXCEPT$CODE),
   DO HOUR$COUNT=0 TO 23,
      DO MINUTE$COUNT=0 TO 59,
         DO SECOND$COUNT=0 TO 59,
            CALL RQ$WAIT$INTERRUPT(AC$INTERRUPT$LEVEL,
               @TIME$EXCEPT$CODE),
            IF SECOND$COUNT MOD 5=0
               THEN CALL PROTECTED$CRT$OUT(BEL),
            END, /* SECOND LOOP */
         END, /* MINUTE LOOP */
      END, /* HOUR LOOP */
   CALL RQ$RESET$INTERRUPT(AC$INTERRUPT$LEVEL, @TIME$EXCEPT$CODE),
   END TIME$TASK,
/* CODE EXAMPLE B  INTERRUPT HANDLER TO SUBDIVIDE A.C.SIGNAL BY 60. */
DECLARE AC$CYCLE$COUNT BYTE,
AC$HANDLER: PROCEDURE INTERRUPT 59,
   DECLARE AC$EXCEPT$CODE WORD,
   AC$CYCLE$COUNT=AC$CYCLE$COUNT+1,
   IF AC$CYCLE$COUNT>=60 THEN DO,
```

```
        AC$CYCLE$COUNT = 0;
        CALL RQ$SIGNAL$INTERRUPT(AC$INTERRUPT$LEVEL,@AC$EXCEPT$CODE);
        END;
    END AC$HANDER;
```

图1-5 OSP举例

9种中断管理的OSP原语是：

DISABLE 阻塞外部中断级

ENABLE 使能外部中断级

ENTER INTERRUPT 为中断处理程序给出其自身的数据段，此段相对于被中断的任务的数据段是独立的。

EXIT INTERRUPT 执行结束中断的操作，用于不调用INTERRUPT TASK的中断处理程序。当中断处理程序放弃控制时，再次使能中断。

GET LEVEL 返回正在执行的中断处理程序级别编号。

RESET INTERRUPT 取消由SET INTERRUPT原语请求对一个中断级所作的原有安排，如果已安排了一个INTERRUPT TASK，也要将其删除，此中断级被阻塞。

SET INTERRUPT 给一个中断级指定INTERRUPT HANDLER或INTERRUPT TASK。

SIGNAL INTERRUPT 中断处理程序用它启动一个中断任务。

WAIT INTERRUPT 在中断处理程序执行SIGNAL INTERRUPT之前暂停正在调用的中断任务，此SIGNAL INTERRUPT是用来调用该中断任务的。如果一个任务的SIGNAL INTERRUPT已经产生，则要对该任务进行处理。

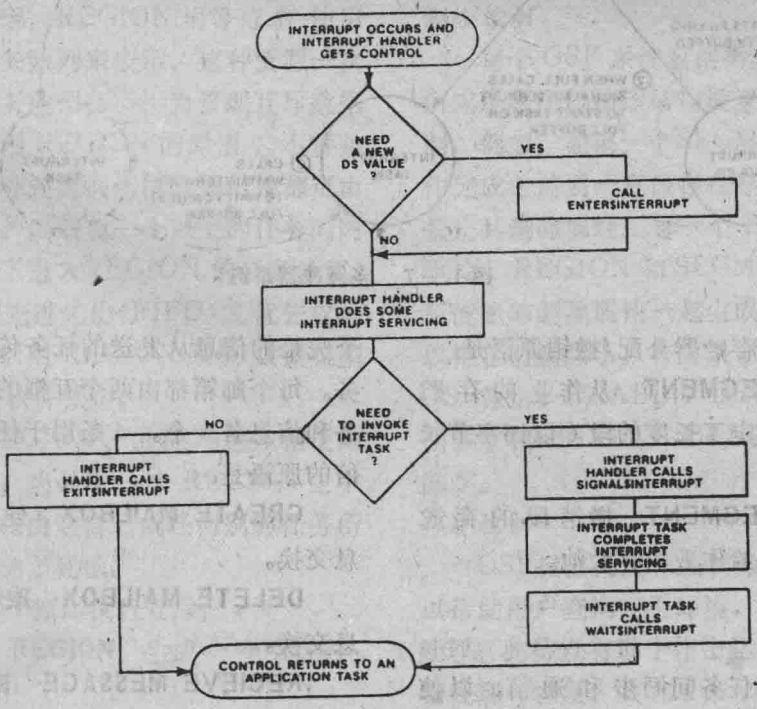

图1-6 中断处理流程图

空闲存贮器管理

OSP 的空闲存贮器管理程序管理每个作业所分配的存贮池（CREATE JOB原语从源JOB的存贮池中指定新JOB 的存贮池）。存贮池是作业资源的一部分，但还没有在该作业的任务之间分配，当在该作业中生成TASK，MAILBOX 或 REGION 的系统数据类型结构时，OSP为它从该作业的存贮池中隐含地指定存贮器，因此不需要分配存贮器的单独的调用。使用空闲存贮器管理的OSP原语隐含CREATE JOB，CREATE TASK,DELETE TASK,CREATE MAILBOX,DELETE MAILBOX,CREATE REGION和 DELETE REGION。当任务需要存贮器时，CREATE SEGMENT 原语直接指定一个存贮区域，例如，任务可以直接分配一个段作为存贮缓冲器，该段的长度是16字节到64K字节之间的16字节的任意倍数。程序员可以指定从 1 字节—64K字节任意数目的字节数，OSP 将自动地将该数值化归为合适的段长度。

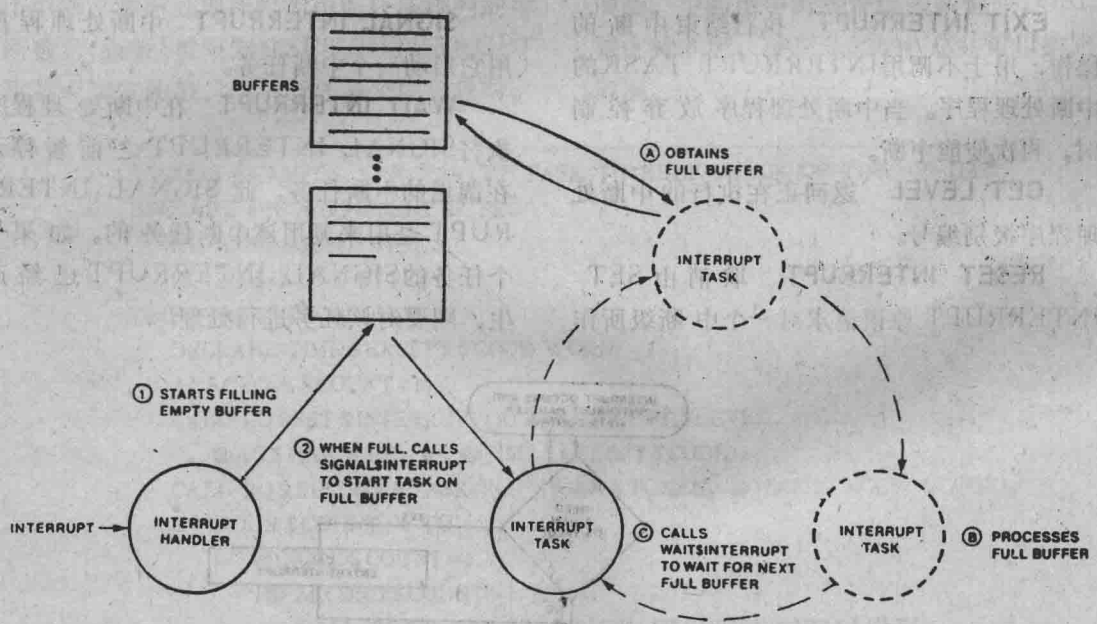

图 1—7. 多缓冲器举例

两种直接的存贮器分配/撤销原语是：

CREATE SEGMENT 从作业的存贮池中分配一个指定了长度的段（以16字节长度为间隔）。

DELETE SEGMENT 撤销段的存贮区，并将它归还给作业的存贮池。

任务间通信

OSP有内部任务间同步和通信，以使得任务可以彼此传递信息和共享信息。OSP的邮箱包含有受控的挂钩程序，它保证一个完整的信息从发送的任务传送到接收的任务。每个邮箱都由两个互锁的队列组成，任务和信息各一个。4 条用于任务间同步和通信的原语是：

CREATE MAILBOX 建立任务间的信息交换。

DELETE MAILBOX 取消任务间的信息交换。

RECIEVE MESSAGE 调用 TASK 从MAILBOX中接收一信息。

SEND MESSAGE 调用TASK 发送一

信息到MAILBOX。

CREATE MAILBOX 原语指派一个MAILBOX用于任务间的信息交换，当发出SEND MESSAGE指令时，OSP将在MAILBOX中以先进先出(FIFO)方式发出信息。同样，当一个任务发出 RECIEVE MESSAGE 原语时，OSP会接收到一个信息。建立该邮箱的任务可以让它用于其他任务。

如果得不到可用的信息，那末希望收到信息的那个任务可以有两种选择：等待信息或继续执行。

任务队列(FIFO或优先权)的排队管理方法决定在MAILBOX TASK 队列中哪一个TASK将从MAILBOX中接收信息，该方法在CREATE MAILBOX原语中指定。

任务间的同步和互斥

对于多道程序设计和多处理机系统来说，互斥是必不可少的。互斥由 REGION系统数据类型实现，REGION 用等待着使用某种资源的任务队列来表示，这种资源一次只能让一个任务使用。OSP 为管理互斥数据和资源提供使用 REGION 的原语，不管是精确的代码部分还是共用的数据结构都可由这些原语来保护，避免一个以上的任务同时使用它们。为了进入REGION 的任务检索，REGION支持先进先出(FIFO)或优先权排队规则，也可以用 REGION 系统数据类型(SDT)来实现软件锁定。

允许多 REGION，并按与进入相反的顺序自动退出，当处于一个 REGION 之中时，任务不可以由它自己或任何别的任务所中止，从而避免了死锁。

有 5 个OSP原语执行互斥：

CREATE REGION 生成一个 REGION(锁定)

SEND CONTROL 废止一个REGION

ACCEPT CONTROL 申请REGION，如果得不到，不等待。

RECIEVE CONTROL 申请REGION，如果不能立即得到，等待。

DELETE REGION 删除一个REGION

OSP 在优先权的 REGION 中为任务提供动态优先级调整。当一个较高优先级的任务发出一 REGION CONTROL 原语而一个较低先级的任务正在使用这同一 REGION 时，则使该较低优先权任务透明地，暂时地升高到等待着的任务的优先级，一直到它用SEND CONTROL 再次释放那个REGION 为止。此时，由于它不再使用该紧急的资源，任务将恢复其正常的优先级。

OSP的控制装置

OSP 包含有对多任务系统提供控制及予制能力的系统原语，利用这些原语控制SDT的删除和系统中空闲存贮器的恢复，允许询问操作系统状态，提供增加用户 SDT和类型管理程序的统一方法。

删除控制

每个 OSP 系统数据类型的删除，直接由应用程序员给该结构设置一删除属性来控制，例如，如果一个SEGMENT在 DMA操作完成之前要一直保存在存贮器中，那末应禁止其删除属性。每一个 TASK，MAILBOX，REGION 和 SEGMENT SDT 同其所使能的删除属性一起生成(也就是说，可以将它们删除)。有两条控制删除属性的OSP原语：ENABLE DELETION(使能删除)和DISABLE DELETION(禁止删除)。

环境控制

OSP提供询问和控制操作，这些操作可以帮助用户查询应用环境，执行灵活的例外处理。此特点有助于作出运行决定和应用出错处理与恢复。共有 5 条 OSP 环境控制原语。

操作系统扩充

OSP 的结构可容许新用户定义的系统数据类型和原语，以便将它们加到由内部系统数据类型所提供的OSP 功能上去。为用户定义的SDT 所生成的类型管理程序称作用户操作系统扩充，并由SET OS EXTENSION原语装配在系统中，一旦装配之后，类型管理程序的功能就可以被符合该OSP接口的用户原语所引用。为了更好地构成扩充结构，每个操作系统扩充都应该支持一种独立的用户定义的系统数据类型，并且每一操作系统扩充为用户提供的调用序列和程序接口与它为内部 SDT 提供的是相同的。扩充用的类型管理程序应当写得适合于应用的需要。将OSP的中断向量入口224—255保留给用户的操作系统扩充，OSP不占用这些入口。当给扩充指定了一个中断编号之后，该扩充的用户就可以用标准的OSP 调用序列（图1-4）和指定给该扩充的特殊软件中断编号来调用它。

ENABLE DELETION 删除一个指定的TASK, SEGMENT MAILBOX 或REGION SDT。

DISABLE DELETION 保护一指定的SEGMENT, TASK, MAILBOX或REGION SDT不被删除。

GET TYPE 为某种系统数据类型环境给定的一个标志，返回该类型码。

GET TASK TOKENS 回送有关现行任务环境信息给提出调用的任务。

GET EXCEPTION HANDLER 回送在调用任务的现场信息给处理器，包括它的地址和使用它的时间。

SET EXCEPTION HANDLER 为任务提供例外处理程序的地址和用法。

SET OS EXTENTION 修改为操作系统扩充保留的中断矢量入口(224—255)中的一个入口，以便指向用户扩充程序。

SIGNAL EXCEPTION 用于操作系统扩充的出错处理。

例外处理

OSP支持例外处理程序，这与CPU 的例外处理程序，例如溢出和非法操作是类似的，其目的是让OSP原语报告原语调用中的参数错误和原语用法上的错误。例外处理过程是灵活的，并可根据不同的应用分别编程，一般情况下，如果一个例外处理程序被调用，它将执行以下一项或几项功能：

- 记录该错误。
- 删除或中止引起这一例外的任务。
- 忽略该错误，认为它影响不大。

例外处理程序写成一个过程，如果使用PL/M86，则应该为此程序的编译指定控制模式，是Compact，Medium还是Large，运行例外处理程序的模式可以在SET EXCEPTION HANDLER原语中指定。图1-4表示来自原语调用的返回信息，用CX 返回标准的系统出错条件，表1-7使用OSP 的缺省EXCEPTION HANDLER 列出了这些条件。

硬件说明

80130与iAPX88/10或 86/10CPU 以紧耦合方式一起工作，80130驻留于CPU的多路复用局部总线上（图1-8），主处理器总是按最大模式配置，80130自动选择 88/30 或86/30操作方式。

从图1-8可见，86/30（或88/30）中的80130可以工作在5MHZ或 8MHZ 而不需要等待状态。当然也可以用需要等待状态的存贮器。80130还可与8087数值处理器扩充及8089输入输出处理器一起使用，提供整个8087的全上下文关系控制。

80130内部分为控制部件(CU)和操作系统部件 (OSU) 两大部分，OSU 含有支持OSP内核的设备，包括用于调度和定时等待的系统定时器，支持中断管理的中断控制

器。

iAPX86/30，iAPX88/30系统配置

80130既是输入/输出，又是映象到CPU局部总线上的存贮器，CPU的状态信号S0/—S2/同IOCS/(用BHE及AD_3—AD_0)或MEMCS/(用AD_{13}—AD_0)一起被译码，将这些引脚上的信号内部锁存。译码关系见表1-1。

存贮器映象

80130的存贮器映象沿着16K字节的边界对准，可从地址线A_{19}—A_{14}得到MEMCS/信号。除了最高的16K(FC000H—FFFF-FH)和最低的1K(00000H-003FFH)之外，80130可以驻留于任一16K字节的边界上。除了这一限制之外，80130的控制存贮代码是与位置无关的，从而使它与许多编码逻辑设计兼容。80130的内核控制存贮器对AD_{13}-AD_0进行译码。

I/O映象

80130的I/O映象必须沿着16字节的边界对准，用地址线A_{15}—A_4来得到IOCS/.

系统性能

表1-5列出了一些典型OSP原语的性能，这些时间参数是相对于8MHZ时钟的iAPX86/30给出的，其执行时间完全可以与小型计算机中类似功能的执行时间相比，而比前一代微型计算机块一个数量级。

初始化

使用OSP既需要应用系统初始化，也需要OSP专用的初始化或生成。生成是在用户提供给iOSP86支持包的数据库基础上进行的。OSP专用的初始化和生成信息区被指定在与80130存贮器映象单元相邻接的用户存贮器地址区(详见Intel公司微型系统元件手册应用注解130)。生成数据表明系统中是否配置8087，指出除了80130之外是否还配用从8259A中断控制器以及建立操作系统时间基准(定时间隔)。在生成区中还有例外处理程序控制参数、独立的应用系统生成区地址单元和所使用的OSP扩充。OSP应用系统生成区可以定位在用户存贮器中的任何地方，并且必须包含要被执行的应用指令代码的启始地址和由OSP空闲存贮器管理程序所管理的RAM存贮块地址。iAPX86/30和iAPX88/30操作系统处理器支持包(iOS-P-86)提供整个应用系统支持和必要的80130生成支持。

RAM要求

OSP管理本身的中断向量，中断向量被指定在低RAM存贮器区，还要求有堆栈空间和数据区工作RAM存贮器，这些存贮空间必须定位在用户RAM中。

OSP中断向量单元OH—3FFH必须是RAM，OSP需要该RAM中的2个字节，从空闲存贮器中动态地分配处理器工作存贮器。每个OSP任务应当分配大约300字节的堆栈。

典型的系统配置

图1-8表示一个典型的iAPX86/30或88/30 OSP系统配置，图中未画出的部分主要是随应用变化的子系统。该配置包括工作于最大模式的8086(或8088)，8284A时钟发生器和8288系统控制器。注意，80130被安排在锁存器或收发器的靠CPU那一边。关于系统的配置详见Intel应用注解130.

OSP定时器

将OSP定时器连接到数据总线的低8位上，按偶地址寻址，以相邻的两个字节来读取该定时器的内容，总是低字节LSB在前，高字节MSB在后。在读操作时，MSB总是被锁定。而且在被读之前保持锁定状态。定时器是不可控的。

波特速率发生器

波特速率发生器与8254兼容(方式3方波发生器)，其输出端BAUD原为高电平，并且在装入计数寄存器之前一直保持高电

平。在装入计数寄存器之后的时钟的第一个下降沿，产生内部计数器到计数寄存器的转换。该输出端保持高电平$\frac{N}{2}$（如果N是奇数，则为$\frac{N+1}{2}$），然后再变低$\frac{N}{2}$（如果N是奇数则为$\frac{N-1}{2}$）。在与该输出端为低电平时最终计数相对应的那个时钟下降缘，BAUD回到高电平并且计数寄存器被转移到内部计数器，然后重复整个过程。波特率列于表1-6。

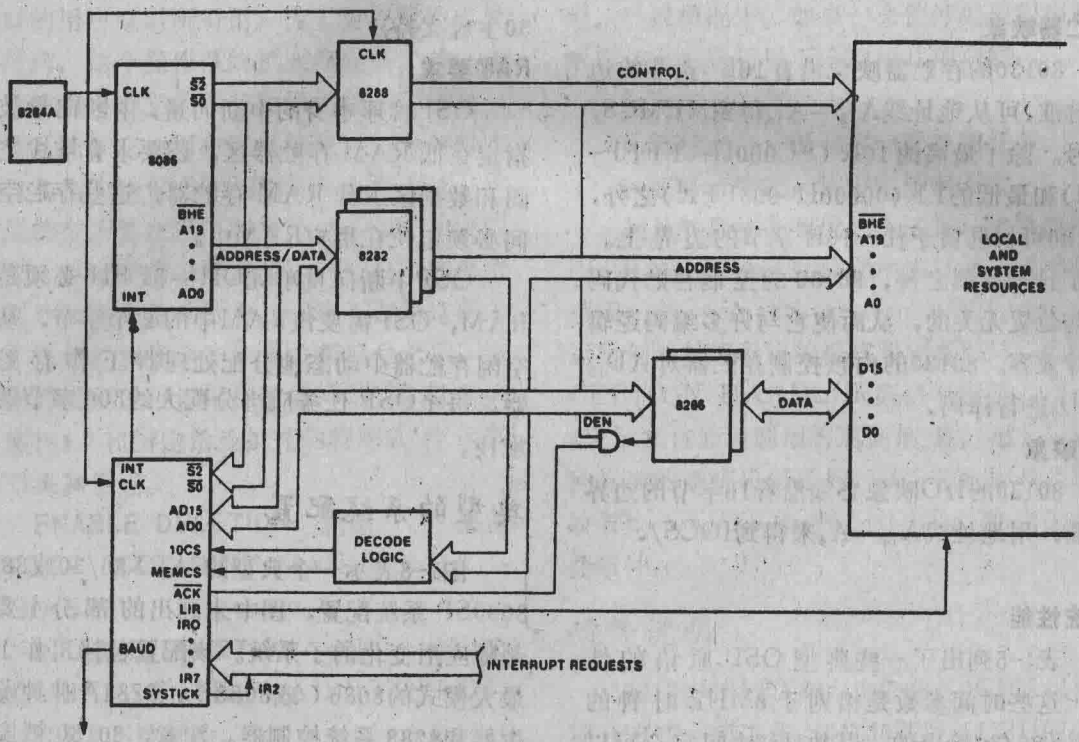

图1-8　典型的OSP配置

波特速率产生器定位在OCH(12)，与定位80130组件(OSF)的I/O空间中16字节边界相对应。定时器控制字定位在相对地址OEH(14)，定时器由IOCS=0寻址，定时器0和1分配给OSP使用，用户不能修改它。

对于大多数波特速率产生器的应用来说，都使用以下的命令字节：

OB6H 读/写波特速率延时值

用计数值52设置9600波特速率的典型序列如下（见表1-6）：

```
MOV AX, OB6H    ；为定时器3写延时作准备
OUT OSF+14, AX  ；控制字
MOV AX, 52
OUT OSF+12, AL  ；先写低字节
XCHG AL, AH
OUT OSF+12, AL  ；再写高字节
```

80130定时器与8254基本上是兼容的。

中断控制器

可编程中断控制器(PIC)也是80130的一个组成部分，它的8个输入引脚处理8级矢量中断，这8个引脚中必须有一个用于定时等待的SYSTICK定时功能，片外连线如图1-8所示。在80130初始化和配置序列中，80130的每一根中断引线可分别编程为电平

触发和边缘触发，外加的从8259中A中断控制器可将OSP外部中断的总数扩充到57。

除了标准的PIC功能之外，80130PIC部件还有一个$\overline{LIR}$输出信号，它为低电平时表示中断响应周期，$\overline{LIR}$ = 0用来控制8289总线裁决器的SYSB/$\overline{RESB}$引脚，这样不用请求系统总线就可响应局部总线非从中断。用户把中断系统作为配置的一部分来定义。

中断序列

OSP的中断序列如下：

1. IR输入端的低—高跳变（边缘触发方式）或高电平输入（电平触发方式）置位一级或多级中断。

2. 80130鉴别这些中断请求，如果合适，给CPU发INT信号。

3. CPU承认INT，并以由$\overline{S_2}$—$\overline{S_0}$编码的中断响应周期来响应。

4. 当收到CPU的第一个中断响应信号时，80130将最高优先级的中断置位，相应的边缘检测锁存器复位，在此周期内80130不驱动地址/数据总线，只是让$\overline{ACK}$ = 0并让被响应的IR输入端送出$\overline{LIR}$值。

5. 然后CPU启动第二个中断响应周期，在此周期中，80130在T_1时刻把相应的级联地址放到总线上，然后再在该总线上送出8位指示字节，CPU读取这一字节。当80130提供该指示字节时，$\overline{ACK}$在此周期内是低的，同时也为被响应的IR输入端送出$\overline{LIR}$值。

6. 中断响应周期到此完成。在中断处理程序的末尾调用一个适当的退出中断原语EXIT INTERRUPT（EOI）之前，ISR位一直保持置位状态。

osp应用举例

图1—5用一个化简了的例子表示使用OSP原语来记录一天的时间，系统设计用60 HZ A.C交流信号作为时间基准。每个A.C周期，电源发出一个TTL兼容信号，此信号用来驱动80130边缘触发中断申请输入脚当中的一根，中断处理程序响应该中断，记录一秒的A.C周期，中断任务计下秒数，一天之后将记数撤销。在典型的系统中，它可以每天执行一次数据记录操作。此中断处理程序和中断任务被写成单独的程序模块。

实际上，当中断59出现时，中断处理程序就响应它，直接将每次中断计数，记到60时，执行SIGNAL INTERRUPT，通知中断任务一秒已经过去。是SET INTERRUPT原语将中断处理程序(ACS HANDLER)指派给这一级的，然后，中断任务执行原语RESUME TASK以恢复应用任务（INITS TASKS TOKEN）。

该任务的主体是计数循环。中断处理程序（在中断级ACS INTERRUPT LEVEL）中的SIGNAL INTERRUPT原语向中断任务发出信号，当后者收到前者发来的信号之后，它将准确地执行一次循环，增加时间计数变量，然后执行等待中断原语WAIT INTERRUPT，在再次得到中断处理程序的通知之前，它一直等待。通常任务要等某一时间之后才能得到下一个信号，然而由于处理程序和任务之间的接口是异步的，处理程序也许已经将中断服务排好了队，任务编制者不必担心这种可能性。

当一天结束时，任务将退出循环并执行RESET INTERRUPT，这样就阻塞了该中断级，撤销了中断任务。此时osp收回此任务所占用的存贮器并调度另一个任务。在原语执行之后如果有一个例外出现，则该例外的编码就会在TIMES EXCEPS CODE中得到。

图1—5通过调用RESET INTERRUPT给出了一个典型的PL/M86调用过程。