〔19〕中华人民共和国专利局

〔51〕Int.Cl.⁴
G06F 15/16

CN 86 1 02261 A

〔12〕发明专利申请公开说明书

〔11〕CN 86 1 02261 A

〔43〕公开日 1986年11月12日

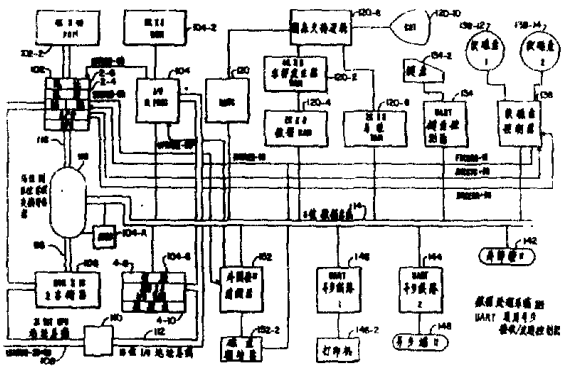
〔21〕申请号 86 1 02261
〔22〕申请日 86.4.2
〔30〕优先权
〔32〕85.4.3 〔33〕美国 〔31〕719,772
〔71〕申请人 霍尼韦尔信息系统公司
地址 美国马萨诸塞州02154
〔72〕发明人 加里·J·格斯 索马斯·S·黑施
霍马斯·O·霍尔蒂

〔74〕专利代理机构 中国国际贸易促进委员会专利代理部
代理人 陈景峻

〔54〕发明名称 通过外壳支持多操作系统的桥接设备

〔57〕摘要

披露了用于自动地从一个操作系统转换到另一个引导装入装置与方法。本发明提出了一种用于自动地将信息，例如时刻，从一个操作系统传送给另一个的方法与硬件。



242/8604635/21

权　利　要　求　书

1.在一个计算机系统中，该系统具有至少一个处理器，一个与所说的处理器以及与一个诸如磁盘存储器那样的档案式存储器系统相耦合的主存储器，所说的计算机系统是在一个从所说的档案式存储器装入到所说的主存储器中的第一操作系统的管理下运行的，一个用于自动地将一个所谓第二操作系统从所说的档案式存储器引导装入到所说主存储器的引导装入装置包括：

(a) 用来存储被所说第一和第二操作系统使用的信息的第一装置；

(b) 与所说第一装置耦合的用来引导装入第一与第二操作系统的第二装置；

(c) 与所说第一和第二装置耦合的用来自动地将信息从所说第一操作系统传送给所说第二操作系统的第三装置。

2.如权利要求1 中所述的计算机系统，包括与所说第三装置耦合的用作从所说主储存器卸下所说第一操作系统的第四装置。

3.如权利要求2 中所述的计算机系统，包括与所说第一装置耦合的所说计算机系统中的管理通道定时器装置（ SCT ）。

4.如权利要求3 中所述的计算机系统，包括与所说 SCT耦合的用来更新所说 SCT的第五装置。

5.如权利要求4 中所述的计算机系统，包括与所说第五装置耦合的用来防止对所说 SCT的更新操作进行中断的第六装置。

6.如权利要求5 中所述的计算机系统，其中所说 SCT由两个寄存器，一个高位字与一个低位字组成，并包括向所说寄存器装入信息和从所说寄存器卸下信息的固件装置。

7.在一个计算机系统中，该系统具有至少一个处理器，一个与所说处理器耦合的主存储器，一个诸如磁盘存储器那样的档案式储存器系统，所

说的计算机系统是由一个从所说的档案式储存器系统装入到所说的主储存器中的第一操作系统管理的，所说的计算机系统还包括用来存放被所说的第一操作系统在所说的计算机系统中工作时所用的信息的操作系统寄存器，用来自动地将一个第二操作系统从所说的档案式储存器引导装入到所说主储存器的引导装入方法包括下列步骤：

(a) 在所说的被第一操作系统使用的操作系统寄存器中存入关键信息；

(b) 将从所说的操作系统寄存器存入的第一信息加以保存；

(c) 对操作系统寄存器进行初始化，以使它们不含信息；

(d) 将第一操作系统从所说计算机系统的主储存器卸下；

(e) 将第二操作系统引导装入所说计算机系统的主储存器；

(f) 将所保存的关键信息装入所说操作系统寄存器以为第二操作系统所用。

8.如权利要求7 中所述的方法，其中有一个管理通道定时器(SCT)包含在所说的计算机系统中供存储时刻用，此外还包含将所说 SCT更新为准确时刻的步骤。

9.如权利要求8 所述的方法，包括防止在所说的 SCT的更新过程中对它进行中断的步骤。

10. 在由一个第一操作系统控制下执行程序的计算机系统中，一个用来自动地引导装入第二操作系统的方法包括下列步骤：

(a) 保存被所说第一操作系统使用的预定信息；

(b) 引导装入所说第二操作系统；以及

(c) 自动地将先前保存的预定信息传送给所说第二操作系统。

说　　明　　书

通过外壳支持多操作系统的桥接设备

本发明一般地涉及计算机系统，特别是有多处理器的计算机系统，其中每个处理器能够在各自独立的操作系统支持下进行工作，其中计算机系统具有一种完善了的初始程序装入程序（引导装入程序）的方法与硬件。

本发明的参考文献为：

1.直接型多路控制设施，序列号503,963,1983年6 月13日。

2.具有独立运行系统的微计算机系统。

计算机系统一般由硬件与软件组成。没有必要的软件，计算机硬件本身是不能完成任何的计算机任务的。需要各种类型的软件以使计算机有条不紊地完成其作业与任务。这些程序通常分为两大类—操作系统程序和应用程序。操作系统通常由诸如管理程序（也称为执行程序或监督程序）、作业操作程序、汇编程序、编译程序、编辑程序等程序组成。应用程序是用户程序，用来完成用户的任务，例如工资单程序，存货清单程序等等。应用程序通常是由操作系统进行管理的，亦即可以对它们进行编译、调度、编辑等等。操作系统也对应用程序进行翻译或编译，以便计算机理解它们。

然而在执行某个程序之前，首先必须将它送入计算机的存储器中。因为没有任何程序的机器是“哑巴”机器，那么怎样才能将必要的程序送入计算机的存储器中呢？一般利用机器操作台上的一个装入按钮启动一个程序，由这个微程序进行控制，将很少量的数据从零地址开始送入存储器中。通常这种初始程序装入抗张过程只送入很少量的数据，例如一张卡片或七个字。于是这就开始了一个程序，它的第一条指令是由初始程序装入微程序读入的。由这个微程序导入的信息包含用来将更多的信息读入存储器的指令，它是一个更通用的装入程序的一部分。用这种方法，机器“引导”

自己首先送入一个装入程序，然后利用它从外部设备将其它的数据与程序装入计算机系统的主存储器中。虽然引入的初始信息块可能相当小，然而一般包含有能够引入更多指令的一些指令，直到引入能够控制一般性的程序及数据输入的装入程序。

在一个多道程序多处理系统中，当发出了“引导”操作信号后，输入/输出（I/O）处理器就进行控制。通常它必须停止所有正在进行中的I/O操作、将包含在它的I/O存储器中的全部通道寄存器复位、此外它必须使主处理器将其全部寄存器与存储管理单元复位，并将其主存储区清零。（通常是在小于装入一个新的操作系统时执行这个初始化过程。）然后这个利用“引导装入”程序的I/O处理器将“装入操作系统与应用程序。然后计算机系统便准备好执行程序并完成指定给它的作业与任务。

在一个多道程序多处理系统中，一个计算机系统可以在任何规定的时间在一个操作系统的控制运行。如果需要改变操作系统，一般还需要进行上述初始化过程。可见，这是耗时的。此外，当操作系统改变时，没有办法从原先的操作系统将信息传给当前的操作系统。因此，停止一个操作系统并“引导装入”另一个操作系统的过程便变成无能为力了。因此为了提高从一个操作系统转到另一个操作系统的效能，所需要的是一种能够将信息从一个操作系统传送到另一个操作系统，从而“桥接”引导装入过程的方法与手段。

本发明的第一个目的是提出一种改进了的数字计算机系统。

本发明的另一个目的是为一个计算机系统提出一种改善了的“引导”装入方法与装置。

本发明还有一个目的是提出一种改善了的“引导”装入方法与装置，利用它将计算机系统从一个操作系统转换到另一个操作系统。

本发明还有一个目的是提出一种改善了的方法与装置，它用来在“引导”装入操作期间或其后将信息从一个操作系统传送给另一个操作系统。

根据本发明的一个实施例，上述这些目的是利用硬件和固件来实现的，这种硬件和固件用来在一个操作系统即将从主存中被撤消并用另外一个操作系统来代替它时存贮一些必要的信息（例如时刻）并把这些信息从一个操作系统传递给另一操作系统。

在本发明的一个实施例中，利用了一个管理通道定时器（以下是指SCT226）。这个SCT是一个分辨度为1秒的32位实时计数器。装入软件时，它便启动，但是不能用软件使它停止。一旦已经启动，它就（向上）计数，直到停电下降。经过软件“引导”键的操作，SCT便有了准确的时间。在微系统6/10上SCT有两组如下所示的I/O功能码对：

功能码（十六进制）	功能
FC-10	输入SCT高位字
FC-12	输入SCT低位字
FC-11	输出SCT高位字
FC-13	输出SCT低位字
FC-14	输入G.P.（通用寄存器）高位字
FC-16	输入G.P.低位字
FC-15	输出G.P.高位字
FC-17	输出G.P.低位字

为了使SCT能够不被中断地更新状态，则当发出输入高SCT（FC-10）与输出低STC（FC-13）时SCT正处在更新过程中的话，上述的输入与输出就成为NAK（也就是说将CPUi指示器201a复位并防止I/O中断更新过程，因此I/O必须再次试行中断）*。

* 此处术语ACK不严格地作为成功地达到了I/O指令目的的同义词，而NAK则用来指示其失败。

为了给SCT提供相关联的装入/卸载功能，实现了下述机构。对于输入，执行FC-10h时，SCT的两半部分都由固件读出，高位字返回，

低位字保存在固件的某个工作单元中。执行FC-12 时只是将这个工作单元中的当前内容送回。FC-12 永远不会被NAK，即能够发生中断。

对于输出操作，执行FC-11 时，只是将输出数据保存固件的某个工作存储单元中。FC-11 也永远不会被NAK。执行FC-13 时，这个工作存储单元中的内容被装入SCT的高位字，FC-13 输出字被装入SCT的低位字，如果SCT尚未在运行的话，它就被启动。

在另一个实施例中，在I/0处理器的RAM（随机存储器）空间中提供两个附加字。这些字用来将信息从一个操作系统传送给另一个正被导入计算机的操作系统，这里因为在引导操作中并不对这些字进行初始化而且当第2 个操作系统被引导装入计算机时，它可以利用这些字中的信息。这些字利用功能码FC-14 至FC-17 。此外，这些字不需要不可分割地进行更新或是对低位与高位寄存器做出上述那样的反应。

因此本发明提出一种方法与硬件，用来将时刻从一个操作系统传送另一个；它也将其它有用的信息，例如调用它并把它引导装入到计算机中的目的传送给第2 个系统。

图1 是体现本发明的计算机系统，微系统6/10的高级框图。

图2 是本发明的高级框图。

图3 是位于I/0处理器的RAM空间中的信箱图。

图4 A,4 B,4 C是被LSI-6处理器在输入/ 输出（I/0）操作中所用的指令格式。

图5 是SCT计数器的格式。

图6 A,6 B是在本发明中所用的通用寄存器的格式。

图6 C-6 D是本发明中所用的寄存器格式。

现在参看图1,它是数据处理系统100 的总体图，它包含一个固件控制的作为应用处理器的中央处理单元（CPU)102与一个作为输入/ 输出处理器的I/0微处理器104 。CPU102 是Honeywell LSI-6型，它运

行Honeywell MOD200 或，MOD400 操作系统，或者任何其它的操作系统。由CPU102 执行存在一个64k ×16bit(位) 字或更大的主存贮器106 内的软件程序来完成系统应用工作。CPU102 执行软件指令时用到的微程序存在一个4k×48位字的只读存贮器(ROM)102-2中。

与I/O微处理器104 配合的是一个8k×8 位字的只读存贮器(ROM) 104-2 与一个 32k×8 位字节的随机存储器(RAM)104-6。ROM 104-2存储了为使数据处理系统100 启动和初始化所必需的固件程序。RAM104-6 存储了表格通信控制程序以及用来对一个通用异步接收/发送控器(UART)144进行仿真的固件、用来控制一些设备的固件，这些设备包括：通过UART134 控制的键盘134-2 、通过软磁盘控制器(FDC)138控制的软磁盘138-12,138-14 、通过UART146 控制的打印机146-2 以及阴极射线管控制器(CTRC)120。RAM104-6 也包括一些由CPU102 以及I/O微处理器104 用作“信箱区”以便相互间进行通信的存储单元。

CPU102 与主存储器106 之间通过21位地址总线108 和16位数据总线116 进行耦合。根据CPU102 所规定的地址，通过数据总线116 在CPU102 与主存储器106 之间传送数据。

数据总线116 耦合到总线交换寄存器118 。耦合到总线交换寄存器118 的还有8 位数据总线114 。总线交换寄存器118 从数据总线116 接收16位数据字，并通过数据总线114 将其作为两个8 位字节发送，同时它也从数据总线114 接收8 位字节，并通过数据总线116 将其发送。I/O微处理器104， RAM104-6 ，CRTC120 UART134 、144 、146,以及FDC138 都耦合到数据总线114 上。

16位I/O地址总线112 通过收发机110 耦合到地址总线108,它也耦合到I/O微处理器104 以及I/O RAM104-6,以此方式能够使CPU102 以及I/O微处理器104 二者都能对主存储器106 和RAM104-6 寻址。

耦合到数据总线114 的还有一个用来控制一个磁盘驱动器152-2 的外围接口适配器152，还有用来通过一个异步端口148 接收与发送数据字符的异步线路UART144，还有用来存储那些在CRT120-10上显示的字符的一个2k×8 位字的数据随机存储器（RAM）120-4，还有用来存储属性字符的一个2k×8 位字的属性随机存储器（RAM）120-6。属性字符通常用来完成CRT120-10的这些显示功能：如加下线字符或字符域以便使某些选定的字符或字符域闪烁或以较强亮度显示。存储在RAM120-4中的字符代码加给一个4k×8 位字的字符发生器随机存取存储器（RAM）120-2，后者为在CRT120-10屏幕上显示出字符，产生与数据的光栅条相对应的代码。一个图象支持逻辑120-8 与CRTC120 、字符发生器RAM120-2 以及属性RAM120-6 相耦合以便在CRT120-10表面产生字符的线条。

FDC138 一般是一台NECu PD765 单/双密度软磁盘控制器，在NEC Electronics U. S. A. Inc.（美国NEC电子公司），Microcomputer Division 微计算机部，1 Native Executive Park, Natick, Massachusetts 01760出版的NEC1982年的目录中对它作了叙述。

I/O微处理器104 通常是一个Motorola MC68B098位微处理单元。PIA152 通常是一个Motorola MC68B21外围接口适配器。CRTC120 通常是一个Motorola MC68B45 CRT控制器。

对I/O微处理器104，PIA152 与CRTC120 在由Motorola Semiconductor Products Inc.（Motorola 半导体产品公司），3501 Bluestein Boulevard, Austin, Texas 78721 出版的Motorala Microprocessor Data Manual（Motorola 微处理器数据手册），1981年版中有所叙述。

UART134，144与146 是Synertics 2661 通用异步接收/发送控

制器，Synertics Corporation（Synertics公司），811 East Argues Avenue，Sunnyvale，California 94086 出版的 Synertics MOS Microprocessor Data Processor Manual（Synertics MOS 微处理器数据处理器手册）中对它们作了叙述。

在大容量存储器和主存储器之间传送数据时涉及到逻辑元件与固件称为数据多路控制（DMC）装置。在其时间内数据进行发送的总线周期称为DMC周期。微系统6/10系统以及这些结构在1983年6 月13日发布的，题为“直接型多路控制设施”的美国专利N.503,963中有更详细的叙述，该专利也是授给本发明的同一受让人的。

现在参看图2,这是本发明的结构框图。不包括8086个人机的基本 Honeywell 微系统6/10系统方案包含一个主存储器205,它可以由一个 Honeywell LSI-6处理器201 与一个Motorola 6809型处理器203 存取。与Motorola 6809型处理器耦合的还有随机存储器（RAM）204 。由于LSI-6处理器是一个16位处理器，随机存储器205 是通过一条16位总线211 与它耦合的。Motorola 6809处理器是一个8 位机，RAM 204 也是8 位宽度的，它们与主存储器是通过一条双总线208 耦合的，该双总线由两条8 位型式的总线组成。从而LSI-6处理器201 可以对主存储器205 与RAM 204进行存取；Motorola 6809型处理器203 也能对主存储器205 与RAM 204进行存取。这个由LSI-6处理器201 、主存储器205 、RAM 204以及Motorola 6809型处理器203 组成的系统，是基本的微系统6/10系统。它是固件驱动的，因而动作很快，只利用主存储器205 的大约20—25% 的存储周期；即利用可用存储器带宽的25% 。LSI-6处理器201 是这个计算机系统的主力部件。它基本上是在Honeywell MOD 400或MOD 200操作系统，或任何其它的操作系统的管理下进行工作的，并执行与这些操作系统兼容的用户系统。另一方面，Motorola 6809

型处理器203 作为一个输入/ 输出处理器工作，在99% 时间内，它利用其自己的局部存储器204 进行工作。根据设计，当LSI-6处理器201 不使用主存储砌205 时，它可以利用主存储器205 。

一个 Intel 8086 型微处理器202 通过总线210 也与主存储器205 耦合。 Intel 8086 处理器202 能够利用MSDOS或CPM-86 操作系统，它们可以执行为 IBM个人计算机写的商品软件，与此相应，作为选用部件微系统6/10系统具有 IBM个人计算机仿真器板。8086处理器202 是软件驱动的，慢得多，大约需用主存储器周期的70—75% 。因此，如果处理器201 、202 和203 中的每一个在它们各自的操作系统控制下独立而并行地运行并处理独立的用户程序的话，就有必要使各个处理器之间对于主存储器带宽的要求保持一定的关系。除这个要求外，必须在主存储器205 中给处理器201 与202 配给各自的存储器空间。这是利用一个地址寄存器A 223 以及与加法器22耦合的一个基地址寄存器225 和比较器229 实现的。这个基地址寄存器存有偏移量，它是与从8086处理器202 来的、并存在地址寄存器A223 的地址通过加法器224 相加。因此主存储器中所有被 Intel 8086 处理器202 请求的地址，都偏离主存储器205 的基地址一个予定的偏移量。比较器209 要确定分配给各个处理器的予定的RAM空间的边界不被超过。由于LSI-6处理器201 是直接请求存储器的地址，并设有偏移量，所以处理器201 与202 在主存储器中有其各自的工作空间。另外，在8086存储器205 基地址的下面还有一个ROM空间。这个ROM空间由8086处理器201 来的负地址寻址。在这个空间，数据只能由8086从主存储器205 读出，但是在执行期间数据不能写入这个区域。因此在这个空间只存储只读程序，诸如为8086处理器201 的BIOS。仅受 Intel8086型的处理器202 由于通过比较器229 确定了边界而不能对留给LSI-6处理器201 的存储空间进行存取，然而，LSI-6处理器却不同，它在这方面并不受限制，它可以对主存储器空间205 中包括8086空间在内的任何

区域进行访问。因此，为了使8086型处理器202 与ＬＳ I-6处理器201 进行通信，在留给8086处理器202 的ＲＡＭ空间中配有一个信号灯/ 信箱212 。对ＲＡＭ存储器204 与其它外围设备，如ＵＳＡＲＴ、软磁盘等进行存取的操作是通过在ＬＳ I-6处理器中201 中执行的一个程序完成的。ＬＳ I-6处理器201 通过ＲＯＭ201-2 中的一个固件程序，利用位于 I/ ＯＲＡＭ空间204 的信箱213 对诸如上面提到的一些物理设备进行寻址，而不对它们直接寻址。（对系统这一方面的情况在下面叙述图3 时还要进一步加以说明。）

然而主存储器空间205 中的信箱区212 是用来在处理器201 与202 间进行通信联系的。在运行中ＬＳ I-6处理器负责对输入/ 输出（ I/ Ｏ）操作初始化。因此当 Intel 8086 型处理器202 希望对一个 I/ O设备进行存取时，需要给信箱212 送一个信息。正常运行时，ＬＳ I-6处理器201 对信箱212 进行管理，并根据请求着手为8086服务。在ＬＳ I-6与 I/ O设备之间的所有通信是由ＬＳ I-6处理器201 通过在ＲＯＭ201-2 中的ＬＳ I-6固件，以及由 I/ O处理器203 完成的。ＬＳ I-6处理器201-2 通过位于 I/ O处理器ＲＡＭ204 中的信箱213 顺序地与 I/ O处理器203 通信。（信箱213 的结构如图3 所示，下面将对它的用途与结构进行说明。）

这个硬件及方法，与辅助的硬件和固件以及与辅助的功能代码一起共同用来将信息从一个操作系统传送给另一个，因而便于从一个操作系统转换到另一个。相应地加上一个管理通道定时器（ ＳＣＴ）226 与固件（下面说明），用来存储时刻，并使定时器以一种的间隔更新内容。此外还拥有一些Ｇ．Ｐ．寄存器225，用于存储由一个操作系统使用，并在引导装入过程中被传送到另一个操作系统的其它信息。计算机系统初始化以及引导装入时，这些硬件寄存器不被清零。

因此可见在 Level 6软件与 I/ O的一些装置之间的所有内部通信是

通过在ROM 201-2中的LSI-6固件，以及通过I/O处理器203实现的，的，二者通过在I/O处理器的RAM 204中的信箱区213进行相互通信。

现在参看图3,该图示出了信箱区213.的信箱图。信箱图有13个字节，每一个字节有8位。偶数字节存在信箱左侧，奇数字节存在右侧。每个字所完成的功能表示在右侧。

信箱的字节1-7用来在ROM 201-2中的LSI-6固件与I/O处理器203之间传送为执行I/O指令所需的信息。LSI-6处理器201将信息装入字节2-7,并装入一个硬件寄存器(未示出),然后置上一个中断I/O处理器的硬件位(未示出)。当I/O处理器将I/O命令处理完毕后，它在字节1中置一个响应码，然后使硬件位复位。

字节0-1

字节0的第2-7位定义LSI-6处理201的当前中断级。由LSI-6固件保持这个数据的当前状态。如果当前中断级在数值上低于或等于某个I/O子系统的中断级，则不接收该I/O子系统来的中断。只要所请求的中断尚未被应用处理器服务，I/O处理器还将在以后某一时刻重新试图中断。这个重新要求中断的操作将一直延续到中断被接受或中断条件复位为止。

字节1的第9位是I/O处理器的忙位。如果一个附属的I/O子系统(即打印机、终端等)与level 6系统软件请求一个附加的功能，而该功能并不能立即被处理的话，便由I/O处理器给忙位置1。这个动作是与在一标准的lecel 6总线上的NAK应答同一类型的信息响应。

字节1的第10位是不存在的资源位。当Level 6系统软件对某个在所用的微系统6/10结构中所没有子系统或设备进行寻址时，便由I/O处理器将它置1。

字节1的其余位与本发明无关，在此不作讨论。

字节2 与3

第0-9 位对由 I/ O命令进行寻址的设备定义通道数。如果应用处理器(L S I-6) 201 插入一个无效的通道号的话，字节1 中的N位便由 I/ O处理器置位。字节3 的第10-15 位定义 I/ O命令的功能代码。

对用来实现本发明的功能代码，将在以后较详细地讨论。

字节4 与5

这些字节包含 I/ O命令数据。

字节6 与7

这些位定义了20位地址中的最低的16位。4 位最高地址以及字节偏移量通过一个硬件寄存器(未示出) 被传送给 I/ O处理器。

中断字节9, A， B

出现中断请求时， I/ O处理器203 便将信息装入字节 A和 B，并给 L S I-6处理器201 送一个硬件中断位。 L S T-6处理器取出字节 A与 B，在字节9 中置入它的响应值并清除硬件中断位。

字节9 是中断信箱区用的控制字节。字节9 的第8 位是标记位(F)，当它由 I/ O处理器置1 时表示有一个需要服务的外部中断。当 L S I-6处理器取走了中断数据，它就将标记(F) 位复位。如果中断不被接受(N A K)，L S I-6处理器便将第9 位置位。

字节 A，字节 B的第8 、9 位

字节 A，以及字节 B中的第8 、9 位包含了产生中断的设备的通道号。字节 B的第10-15 位包含了指定给该通道的中断等级。

寄存器与暂存存储器

下面对包含数据多路控制(DM C) 寄存器以及暂存存储器的字节进行定义。

字节 C

第0-3 位定义准备将哪个 Lenvel 寄存器的信息传送给 I/ O信箱或

从 I/ O信箱将信息传给哪个 Lenvel 6 寄存器。如果第0-3 位的内容是数值0 至7,则在 L S I-6处理器201 中的一个16位的 C P U R寄存器的信息将被送到字节 E与 F，或者后者的信息将被传送给前者。如果第0-3 位的数值是A至 F，则一个20位的 C P U B寄存器的信息将被送到字节10，11和13，或者后者的信息将被传送给前者。

字节 C的（ S）位是传送方向标记位。0 状态表示方向是进入信箱；而1 状态则表示方向是进入 L S I-6处理器。

字节 C的（T）位表示在紫节 E到11与13中存储了哪一类型的信息。当它为0 时，这个信息便是数据多路控制（ DM C）信息，当它为1 时，这个信息便是 Lenvel 6 寄存器信息，在这种情况下，由字节 E、F、11与13传送一个范围与地址。

字节 D

（ F）位——当它被置为1 时——表示有一个寄存器的传送操作正在进行中。

R F U是备用空间。

字节 E和 F

这些字节用来传送数据。它们或者包含了一次DM C操作的范围，或者包含了16位 Level 6寄存器的内容。

字节10，11和13

这些字节用在数据传送中。它们包含或者是一次DM C操作的地址或者是一个20位 Level 6寄存器的内容。字节10和11包含了最低的16位地址位，字节13包含最高的4 位。

现在参看图5,图中是 S C T计数器。 S C T计数器由两个寄存器401 、402 组成，每个寄存器能够存两个36位字，总共32位。 S C T寄存器的分辨度为1 秒。 S C T寄存器位于 I/ O R AM205 中，用数字226 标示。 S C T的内容是由 L S I-6软件装入的，该软件利用了以后要叙述的 I/

O指令。一旦被软件装入后，它就不能由软件停止，而由下面要叙述的固体每秒维护一次。因此SCT在整个软件“引导装入”操作中保持了以秒为单位的准确的时间。

参看图4A,4B与4C，它们表示了通常存放在主存储器105中的典型的指令格式，这些指令供LS I-6处理器在输入/输出（I/O）操作中使用。图4A与4B的第0至8位表示I/O指令的类型，而第9至15位表示正在传送的信息从何处来或到何处去的地址。在本发明中，作为例子，需要传送的信息是时刻，它被从SCT226传送到主存储器205或传送到LS I-6处理器201中的一个寄存器以供新装入的操作系统使用。进行传送的方向由功能码表示。功能码及其功能如下如示：

功能代码（十六进制）	功能
FC-10	输入SCT高位字
FC-11	输出SCT高位字
FC-12	输入SCT低位字
FC-13	输出SCT低位字

如果需要将信息根据I/O指令中的地址字节传送到主存储器中的某个位置的话，就利用代码10或者12。功能代码10将SCT的高位字信息传送出去；而功能代码12则将SCT的低位字信息传送出去。这个信息首先被送到信箱区213,然后通过I/O指令的数据字被送到LS I-6处理器。

如果需要反向传送信息，例如当SCT由LS I-6设定为某一日期时，就在I/O指令中利用功能代码11和13，过程就反向。

为了给SCT提供相关联的装入/卸载功能，需要根据下述过程。对于输入，执行FC-10时，将由固件读出SCT的两半部分，高位字返回，而低位字则保存在固件的一个工作单元中。执行FC-12时，只有将该工作单元的当前内容返回。FC-12始终不被NAK。

对于输出，执行FC-11时，只是将输出数据保存在固件工作单元中。

FC- 11始终不被NAK。执行FC- 13时将该工作单元的内容装入SCT的高位字中，FC- 13输出字装入SCT低位字，SCT被启动，如果尚未在运行中的话。

现在参看图 4C，这是另一种类型 I/ O指令的格式，其中前10位用来存通道号。这个号数表示信息来去的通道。（在这是结构中，典型的通道是信箱212,213,对它们每一个都予先指定了通道号。）上面所述的功能码表示本发明中的信息流向，它位于执行用指令的第10-15 位。

参看图 6A与 6B，它们表示通用寄存器225 的格式。两个寄存器都有相似的格式，都是16位长。这些寄存器在引导装入之前、之中或之后都是不可擦的。这些寄存器是由 I/ O指令利用适当的功能码装入的，它们包含可以传送给新装入的操作系统的编码信息，而传送操作也是通过 I/ O指令利用适当的功能代码实现的。可以传送给一个新操作系统的典型信息是，为了确定正确的固件(right firmware)是否位于该参照系统关键表(operating system's key table)所在的位置信息。传送给新操作系统的另一个典型信息是需要执行的功能，即字处理、计算等。

用来将信息传送到G. P. 寄存器225 或从那里传送来的信息代码如下如示：

功能代码（十六位）	功能
FC-14	输入G. P. 高位字
FC-16	输入G. P. 低位字
FC-15	输出G. P. 高位字
FC-17	输出G. P. 低位字

现在参看图 6C，这是位于RAM204 中的SCT- LOCK226a的格式。SCT- LOCK是一个8 位的单字节字。当SCT- LOCK被置为0 时，I/ O指令在传送信息时便不成功，软件必须再次争取进行这项操作，如果SCT- LOCK被置为1，I/ O指令便可成功地完成其目

标。

如果 I/ O指令成功的话，便通过一个称为ＡＣＫ的过程而将i 指示器201a置位；而如果 I/ O指令不能达到目标的话，便通过一个称为ＮＡＫ的过程而将i 指示器201a复位。（因而必须指出，在这里术语ＡＣＫ是成功的同义词，而术语ＮＡＫ是 I/ O指令的目标不能完成的同义词）。

现在参看图 6D与 6E，这里分别表示了ＳＣＴ- IN- ＳＡＶＥ与ＳＣＴ- ＯＵＴ- ＳＡＶＥ寄存器的格式。每个格式由两个8 位的字节组成。二者都位于6809处理器的ＲＡＭ204 中，各自用来保留输入或输出信息。

与上述硬件组合相联系，有如下三个相关的固件程序：

Ａ．分别对ＳＣＴ- ＨＩＧＨ和ＳＣＴ- ＬＯＷ实施输入/ 输出的固件程序（ I/ Os）。 I/ Os 是在不可屏蔽中断（ＮＭ Ｉ）级中被执行的，这表示 I/ Os 可以中断ＳＣＴ固件处理程序（ＳＣＴ handler fimware routine)(将在下面叙述),但反之不然。因此对ＳＣＴ固件处理程序，将ＳＣＴ- ＬＯＣＫ置位便足够了，而对于 I/ Os 则只要对它进行检查就足够了。

Ｂ． ＳＣＴ固件处理程序，它每秒将ＳＣＴ定时器226 增值一次。

Ｃ． Start- ＳＣＴ（启动ＳＣＴ）调用一个通用启动定时器程序（Grneral Purpose- ＳＴＡＲＴ ＴＩＭＥＲ ＲＯＵＴＩＮＥ。这个程序给16个通用定时器226d中的一个装入某个给定值并作出它在运行的标志。它也为这个定时器在一张表格中存入处理程序的地址。所有标志出在运行中的定时器或者60赫兹（标准的）或者以50赫兹可选择的（如果电源频率为50赫兹的话）的速率连续减值。到定时器达到0 时，便作出它不在运行的标志，并使指定给此定时器的处理程序是工作，此处理程序是一个子程序。

下面详细列出实际的固件程序。它们的功能流程如下所示：

I/ O固件过程

过程— Input - SCT- HIGH

如果SCT- LOCK被锁住，即等于(0)

NACK I/ O（I/ O未完成的软件将再试）

如果SCT- LOCK未被锁住，即等于1(1)，于是便复制SCT-LOW，并将它保存在SCT- IN- SAVE中

SCT- HIGH返回到 I/ O并调用ACK

结束 Input- SCT- HIGH

过程— Input- SCT- LOW

返回SCT- IN- SAVE到 I/ O并ACK（即，给i 指示器置位）

结束 Input- SCT- LOW

过程—Output- SCT- HIGH

复制 I/ O字到SCT- OUT- SAVE并ACK

结束Output- SCT- HIGH

过程—output- SCT- LOW

如果SCT- LOCK被锁住，即等于（0）

NAK I/ O（软件将再试）

复制SCT- OUT- SAVE并保存在SCT- HIGH中

复制 I/ O字到SCT- LOW

启动SCT

ACK I/ O

结束Output- SCT- LOW

SCT- HANDLER过程

SCT- LOCK被封锁，即等于(0)(这将SCT- LOCK置0)

SCT- LOW等于SCT- LOW+1(这使SCT- LOW定时器增值)

如果SCT- LOW等于0,则

SCT- HIGH等于SCT- HIGH+1

SCT- LOCK不被封锁,等于(1)

新的start- SCT过程

结束SCT- HANDLER

Start- SCT过程

利用下列参数调General Purpose start timer routine (通用启动定时器程序):

定时器数等于SCT- TIMER

处理程序等于SCT- HADLER (上述过程)

如果是60赫兹的机器的话,频率等于60

如果是50赫兹的机器的话,频率等于50

结束Start- SCT

在说明了本发明的一个最佳实施例后,本技术的行家们可以知道,对所述的发明尚可有许多形式与变动,然而仍然在申请权利要求的本发明的范围之内。因此只能根据本权利要求所指范围限定本发明。

SEQUENCE: $$SDS

LINE #					ADDRESS (HEX)	IMAGE (HEX)
10000*440	\					
10000*450	\ SUPERVISORY CHANNEL FUNCTION CODES 10# TO 13#					
10000*460	\ SUPV CHAN TIMER IN/OUT					
10000*470	\					
10000*480	*SUPV-MORE	TFR	(R,A)	\ COPY FC	58A9	1F 98
10000*490	$	ANDI	(R,FF#)	\ STRIP VARIABLE BITS	58AB	C4 F8
10000*5C0	$	CMPI	(R,10#)	\ CK FC = 10#-17#	58AD	C1 10
10000*510	$	LBNE	($IO-NONX)	\ NON-EXISTENT RESOURCE IF NOT	58AF	10 26 00 9D
10000*520	\					
10000*530	$	BITI	(A,04#)	\ TEST IF 2ND GRP OF FUNC. CODES	58B3	85 04
10000*540	$	BNE	($SUPV-:017)	\ BR IF SO	58B5	26 4C
10000*550	\					
10000*560	$	LDAI	(R,40#)	\ NAK CODE	58B7	C6 40
10000*570	$	BITI	(A,01#)	\ CK IN/OUT FC BIT	58B9	85 01
10000*580	$	BNE	($SUPV-:[illegible]	\ 0-IT OR FC-[illegible]	58BB	26 1F
10000*590	\					
10000*600	\ FC 10 * FC [illegible]			\ SET-LO		
10000*610						
10000*620	$	BITI	(A,02#)	\ CK IF FC-12	58CD	85 02
10000*630	$	BNE	($SUPV-12)	\ BR IF SO	58CF	26 12
10000*640	\					
10000*650	\ FC-13 = INPUT SET-HI = CK COPY THEN READ BOTH HI & LO, SAVE LO					
10000*660	\					
10000*670	$	TSTE	($SCT-LOCC)	\ NO NEED TO COPY IT - RETURN IN MMI	58C1	79 09 C5
10000*680	$	LBEQ	($SUPV-RSP)	\ NAK IF LOCKED (=0)	58C4	10 27 FF 9F
10000*690	\					
10000*700	$	LDDE	($SCT-2)	\ GET SCT-LO	58C8	FC 08 B8
10000*710	$	STXE	($SCT9F-2)	\ SAVE IT FOR FC-12	58CB	FD 08 C3
10000*720	\					
10000*730	$	LDDF	($SCT-0)	\ GET SCT-HI	58CE	FC 09 B9
10000*740	$	BRA	($SUPV-IN)	\ RETURN IT	58D1	20 03
10000*750	\					
10000*760	\ FC-12 = JUST RETURN SAVED SCT-LO					
10000*770	\					
10000*780	$SUPV-12	LDDE	($SCT9F-2)	\ GET SAVED SCT-LO	58D3	FC 09 C3
10000*790	\					
10000*800	\ PUT RETURN IN -SX AND ACK					
10000*810	\					
10000*820	$SUPV-IN	STXE	($:[illegible]-3)	\ PUT DATA WORD IN MBR	58D6	F9 CE 04
10000*830	$	[illegible]		\ ACK CODE	58D9	5F
10000*840	$	BRA	($SUPV-RSP)	\ &) ACK & END IT	58DA	20 A9
10000*850	SKIP REF					

```
MODEL: HERCULES                                          RTL/6000 FILE EDIT                    01/17/25      8.034  PAGE:  12
REVISION: 000.00                                          MICROPROGRAM SECTION                 OOC_#: /R9_3/ASYNC-9.531

                                                        SEQUENCE: $$ROS

 LINE #                                                                                  ADDRESS   IMAGE
10002860   \ SUPV FC-11 & FC-13 - OUTPUT SCT-HI & SCT-LO                                  (HEX)     (HEX)
10002870   \
10002880   $SUPV-1113 BITI   (A,02#)          \ CK IF FC-13                                 580C   85 02
10002890   $          BNE    ($SUPV-13)       \ BR IF SO                                    580E   26 0A
10002900   \
10002910   \ SUPV FC-11 - OUTPUT SCT-HI - JUST SAVE FOR LATER
10002920   \
10002930   $          LDDE   ($IOMX-3)        \ GET OUTPUT WORD                             58E0   FC 0E 04
10002940   $          STWE   ($SCT8F-0)       \ SAVE IT FOR LATER                           58E3   FD 08 C1
10002950   $          CLRB                    \ ACK CODE                                    58E6   5F
10002960   $          LBRA   ($SUPV-RSP)      \ GO ACK AND END I/O                          58E7   16 FF 90
10002970   \
10002980   \ SUPV FC-13 - OUTPUT SCT-LO - CK LOCK, THEN LOAD BOTH SCT-HI & SCT-LO
10002990   \  ALSO START SCT TIMER
10003000   \
10003010   $SUPV-13   TSTE   (*SCT-LOCK)      \ TEST LOCK                                   58EA   70 08 C5
10003020   $          LBEQ   ($SUPV-RSP)      \ NAK IF LOCKED - CODE LOADED AT -MORE        58ED   10 27 FF 96
10003030   \
10003040   $          LDDE   ($SCT8F-0)       \ SAVED SCT-HI OUTPUT                         58F1   FC 08 C1
10003050   $          STWE   ($SCT-0)         \ TO SCT-HI                                   58F4   FD 08 89
10003060   \
10003070   $          LDDE   ($IOMX-3)        \ THIS FC DATA                                58F7   FC 0E 04
10003080   $          STWE   ($SCT-2)         \ TO SCT-LO                                   58FA   FD 08 88
10003090   \
10003100   $          BSR    ($ST-SCTMR)      \ START SCT TIMER                             58FD   D2 38
10003110   \
10003120   $          CLRB                    \ ACK CODE                                    58FF   5F
10003130   $          LBRA   ($SUPV-RSP)      \ ACK FC-13                                   5900   16 FF 94
10003140     SKIP HOF
```

SEQUENCE: SSROS

LINE #					ADDRESS (HEX)	IMAGE (HEX)
10003150	\					
10003160	\ 2ND GROUP OF SUPV CHAN IO'S - FC14-FC17					
10003170	\ TRANSPARENT EXCEPT THAT LSB OF FC16 IS LOADED W/ BOOT ID CODE					
10003180	\					
10003190	$SUPV-1417	LDXI	($SCP-14)	\ PT TO 1ST OF GROUP	5903	8E 08 BD
10003200	$	BITI	(A,02#)	\ TEST IF 2ND PAIR OF GRP	5906	85 02
10003210	$	BEQ	($SUPV-14XX)	\ BR IF NOT	5908	27 02
10003220	$	LEXXC	(X,2)	\ STEP PTR TO 2ND WORD	590A	30 02
10003230	$SUPV-14XX	BITI	(A,01#)	\ TEST IF OUTPUT FC	590C	85 01
10003240	$	BNE	($SUPV-1517)	\ BR IF SO	590E	26 05
10003250	\					
10003260	$	LDDXR	(X,N)	\ GET WORD	5910	EC 84
10003270	$	LARA	($SUPV-IK)	\ GIVE TO CP	5912	16 FF C1
10003280	\					
10003290	$SUPV-1517	LDDE	($IOPX-3)	\ GET OUTPUT WORD FROM IO MPX	5915	FC 0E 04
10003300	$	STWXR	(X,N)	\ STOW IT	5918	ED 84
10003310	$	CLRB		\ ACK CODE	591A	5F
10003320	$	LARA	($SUPV-RSP)	\ GO ACK LSI-6 & END UP	591B	16 FF 69
10003330	SKIP HOF					

```
MODEL: HERCULES                                          RTL/6000 FILE EDIT                    01/17/25      8.034  PAGE:   14
REVISION: COC.00                                           MICROPROGRAM SECTION                  OOC.#: /R9.3/ASYNC-9.531

                                                          SEQUENCE: SSROS                    ADDRESS   IMAGE
 LINE #                                                                                        (HEX)     (HEX)
10003340   \ SUPV CHAN TIMER HANDLER - BUMP SCT & RE-CALL TIMER                                591E   7F 08 C5
10003350   \                                                                                   5921   FC 09 88
10003360   $SCTHAN     CLRE   ($SCT-LOCK)    \ SET SCT LOCK                                   5924   C3 00 01
10003370   $           LDDE   ($SCT-2)       \ SCT-LO                                         5927   FD 08 88
10003380   $           ADWI   (1)            \ +1                                             592A   26 09
10003390   $           STWE   ($SCT-2)
10003400   $           BNE    ($SCTHAN-8)    \ AROUND IF LO HALF NOT NOW ZERO                 592C   FC 08 89
10003410   \                                                                                   592F   C3 00 01
10003420   $           LDDE   ($SCT-0)       \ SCT-HI                                         5932   FD 08 89
10003430   $           ADWI   (1)            \ +1                                             5935   C6 01
10003440   $           STWE   ($SCT-0)                                                        5937   F7 08 C5
10003450   \
10003460   $SCTHAN-8   LDAI   (8,01H)        \ LOCK CODE
10003470   $           STAE   (8,$SCT-LOCK)  \ RESET SCT LOCK                                 593A   C6 3C
10003480   \                                                                                   593C   B0 08 48
10003490   \ START SCT TIMER                  :                                              593F   85 05
10003500   \                                                                                   5941   27 02
10003510   $ST-SCTMR   LDAI   (8,60)         \ FOR 60 HZ                                      5943   C6 32
10003520   $           LDAE   (A,$TSR2)      \ 7300 STATUS REG                                5945   4F
10003530   $           BITI   (A,HZ50)       \ TEST IF SCREEN AT 50 HZ                        5946   1F 02
10003540   $           BEQ    ($STSCT-8)     \ BR IF NOT                                      5948   86 02
10003550   $           LDAI   (8,50)         \ USE 50 HZ VALUE IF 50                          594A   17 FE A9
10003560   $STSCT-8    CLRA                  \ FOR WORD VALUE                                 594D   0D C4
10003570   $           TFR    (D,Y)          \ FOR CALL                                       594F   39
10003580   $           LDAI   (A,SCTTMR)     \   ALSO
10003590   $           LBSR   ($GO-STTMR)    \ START TIMER FOR ONE SECOND TIMEOUT
10003600   $           CNSTI6 ($SCTHAN-V)    \ VECTOR FOR HANDLER BELOW
10003610   $           RTS
10003620   \
10003630     SKIP HOF
```

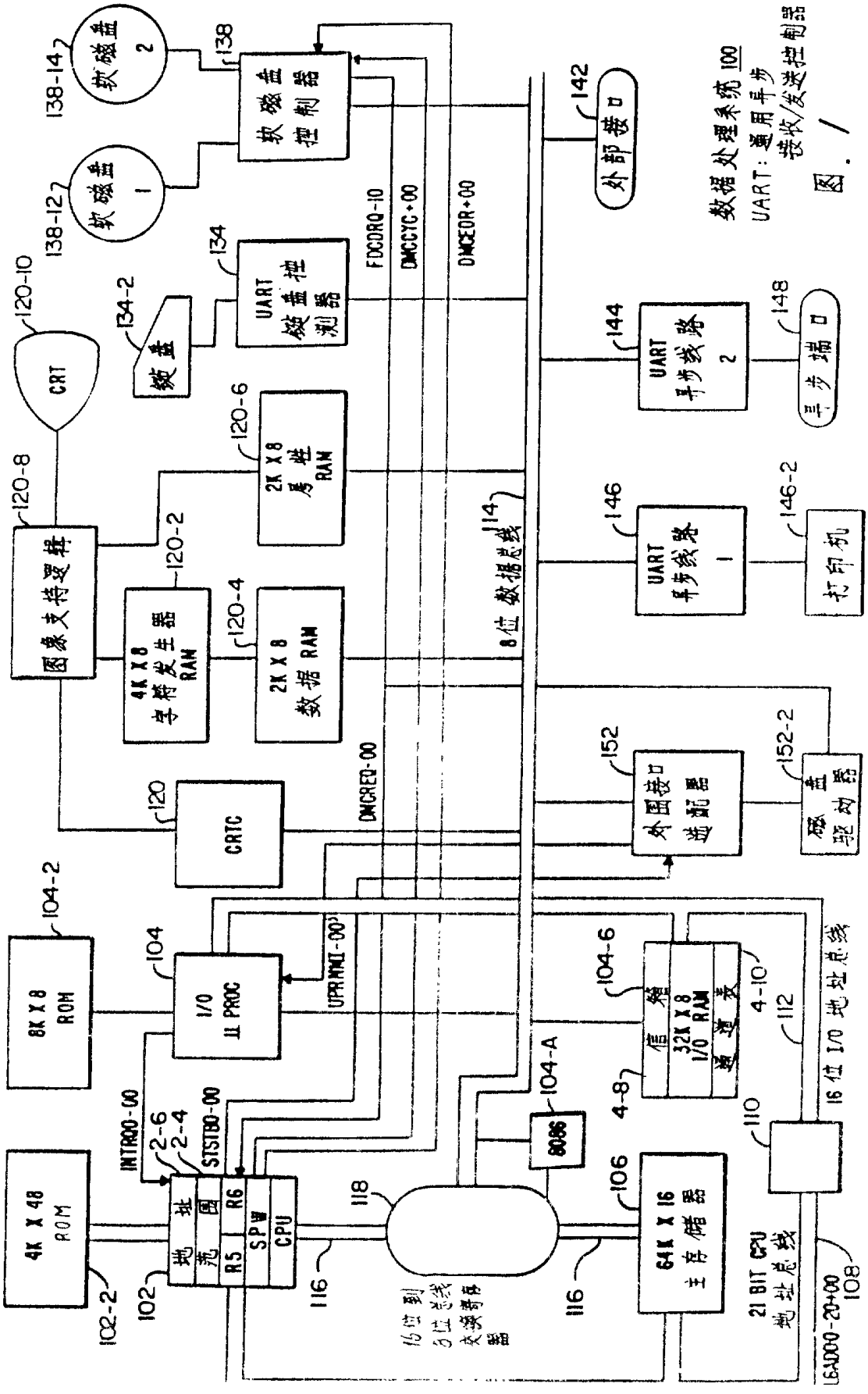

数据处理系统 100
UART：通用异步
接收/发送控制器
软磁盘 2
软磁盘 1
软磁盘控制器
外部接口
UART 异步线路 2
异步端口
UART
键盘
CRT
2K X 8 易挥发 RAM
4K X 8 字符发生器 RAM
2K X 8 数据 RAM
图象支持逻辑
UART 异步线路 1
打印机
8位 数据总线
CRTC
外围接口适配器
磁盘驱动器
8K X 8 ROM
I/O 11 PROC
32X X 8 I/O RAM
16 位 I/O 地址总线
8086
4K X 48 ROM
地址 范围 R5 R6 SPW CPU
64K X 16 主存储器
21 BIT CPU 地址总线
INTR00-00
STSTB0-00
UPRNMI-00
DMCREQ-00
FDCDRQ-10
DMCCYC+00
DMCEOR+00
16400-20+00

图. 1

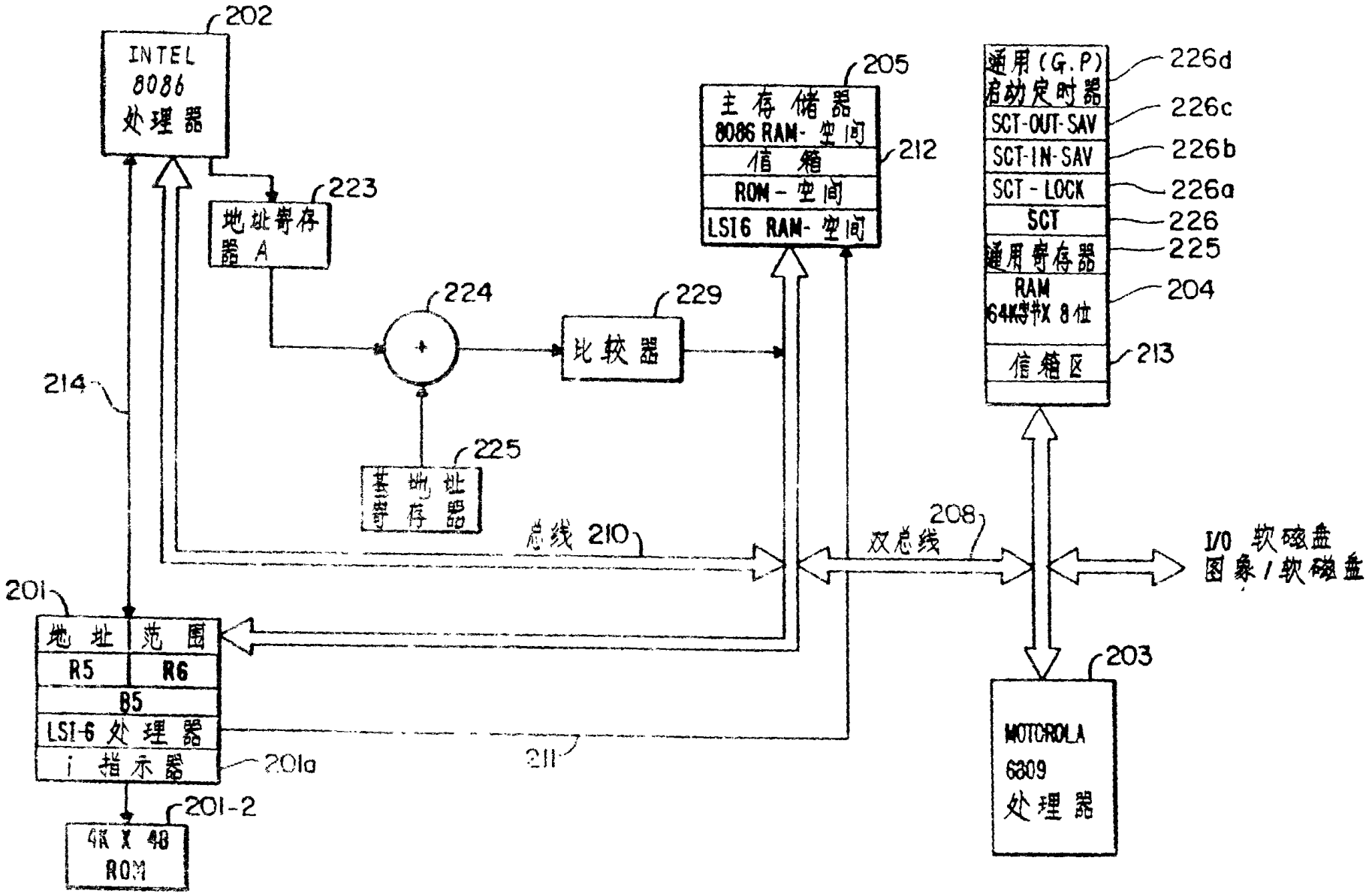

202
INTEL
8086
处理器
223
地址寄存器 A
224
229
比较器
225
基地址寄存器
214
205
主存储器
8086 RAM-空间
信箱
ROM-空间
LSI6 RAM-空间
212
通用(G.P)启动定时器
SCT-OUT-SAV
SCT-IN-SAV
SCT-LOCK
SCT
通用寄存器
RAM
64K字节X 8位
信箱区
226d
226c
226b
226a
226
225
204
213
总线 210
双总线 208
I/O 软磁盘
图象/软磁盘
201
地址
范围
R5
R6
B5
LSI-6 处理器
i 指示器
201a
211
203
MOTOROLA
6809
处理器
201-2
4K X 48
ROM

图 2

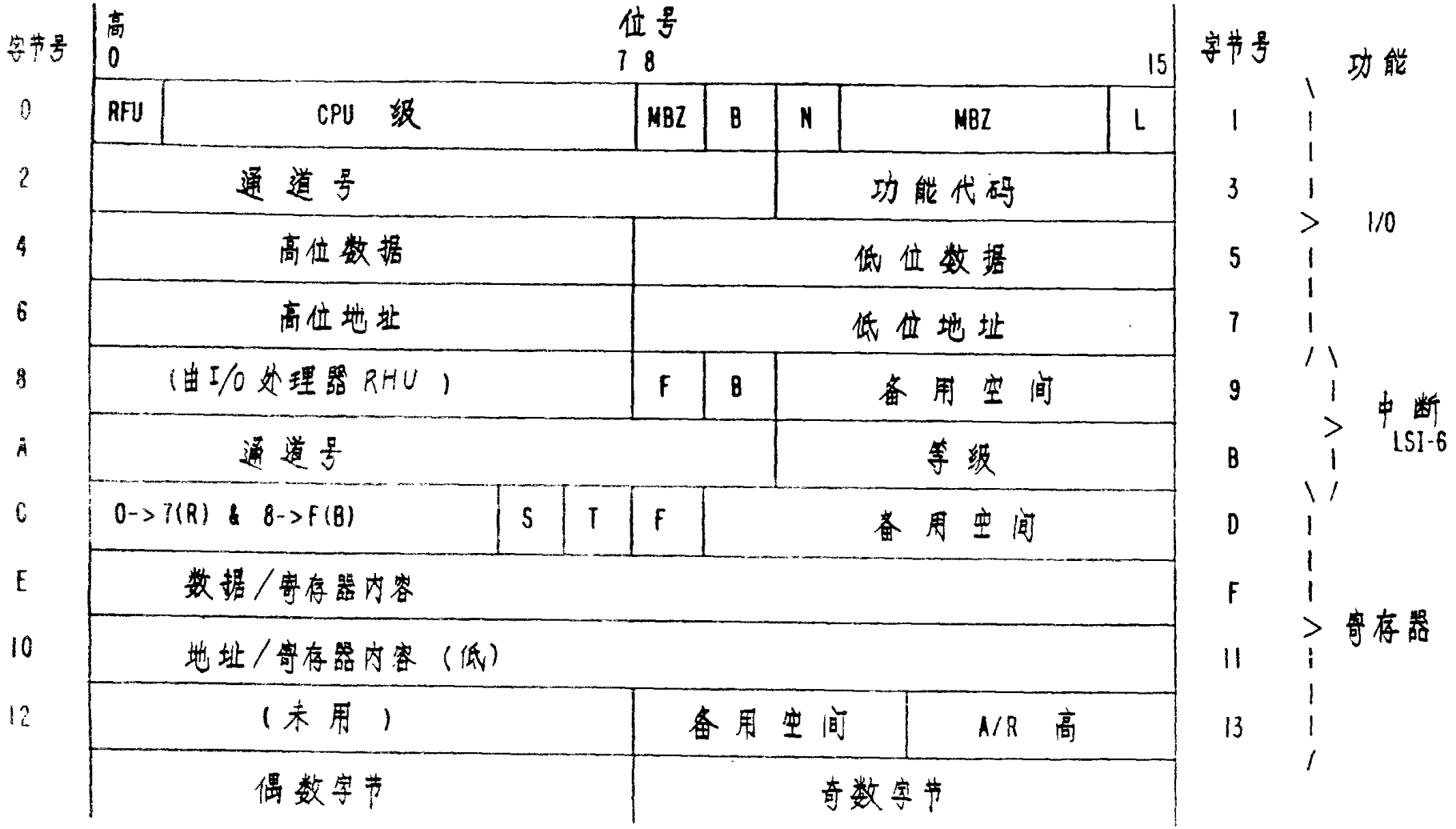

字节号
位号
高
0
7 8
15
字节号
功能
RFU
CPU 级
MBZ
B
N
MBZ
L
通道号
功能代码
高位数据
低位数据
高位地址
低位地址
(由I/O处理器RHU)
F
B
备用空间
通道号
等级
0->7(R) & 8->F(B)
S
T
F
备用空间
数据/寄存器内容
地址/寄存器内容 (低)
(未用)
备用空间
A/R 高
偶数字节
奇数字节
0
2
4
6
8
A
C
E
10
12
1
3
5
7
9
B
D
F
11
13
I/O
中断
LSI-6
寄存器

图3

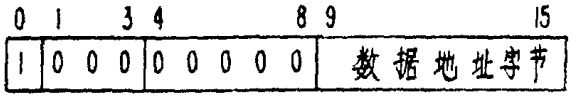


图. 4A

图. 4B

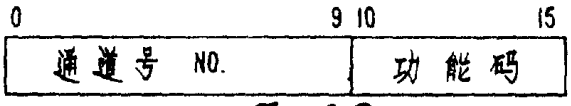


图. 4C

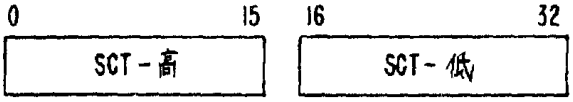


图. 5

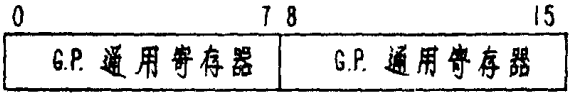


图. 6A

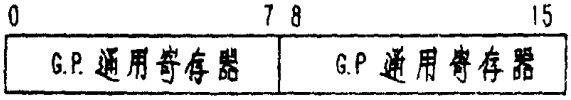


图. 6B

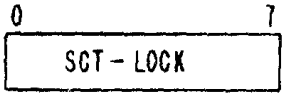


图. 6C

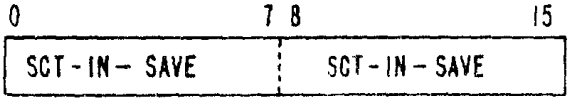


图. 6D

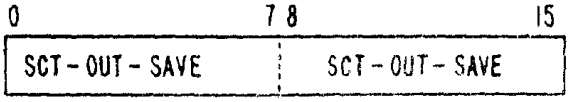


图. 6E