PCI 芯片 CH365 升级 ISA 板卡与设计 PCI 板卡

（本地硬件定址升级 ISA 板卡）

版本：2.1

http://wch.cn

1、概述

本文将介绍 ISA 板卡向 PCI 板卡的升级、PCI 板卡与 ISA 板卡软件设计的差别，另外还提供了一些常用的现成电路图，供设计 PCI 板卡时参考。

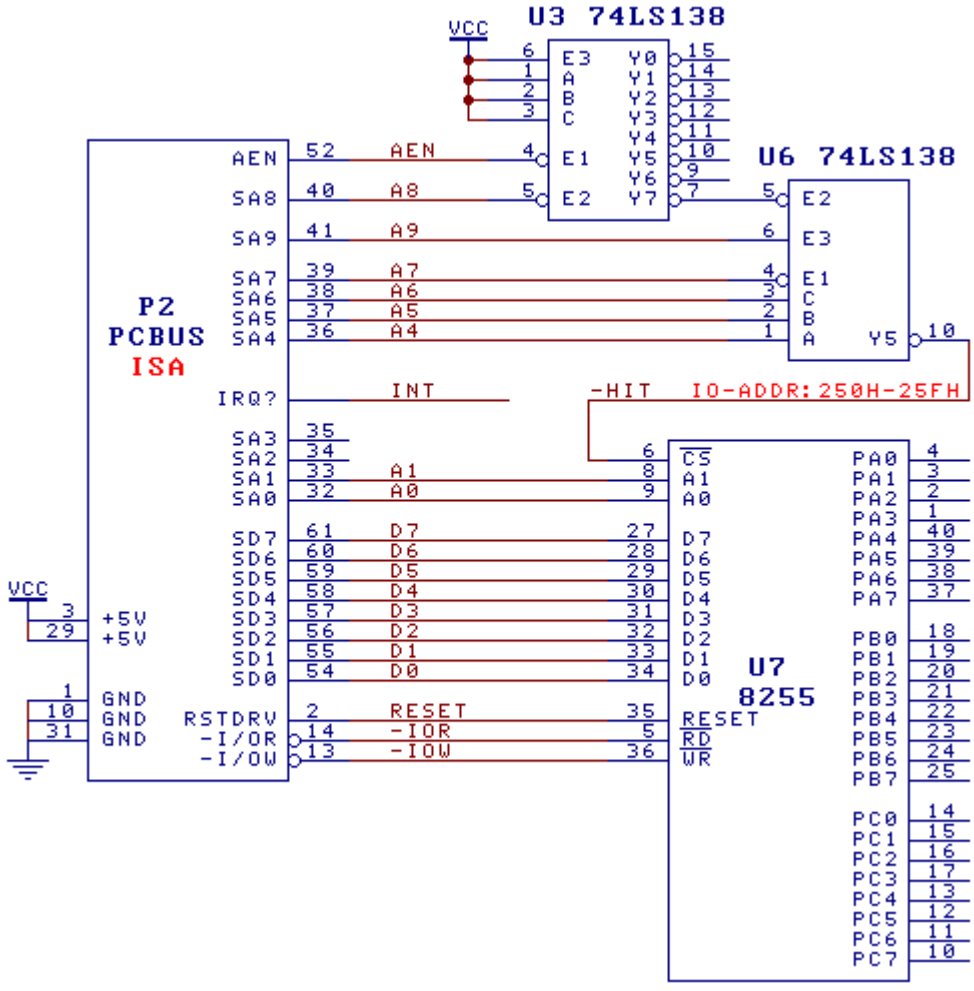
如果需要升级到 PCI-Express 板卡，请参考 CH367 和 CH368 芯片的相关资料。

2、升级 ISA 板卡

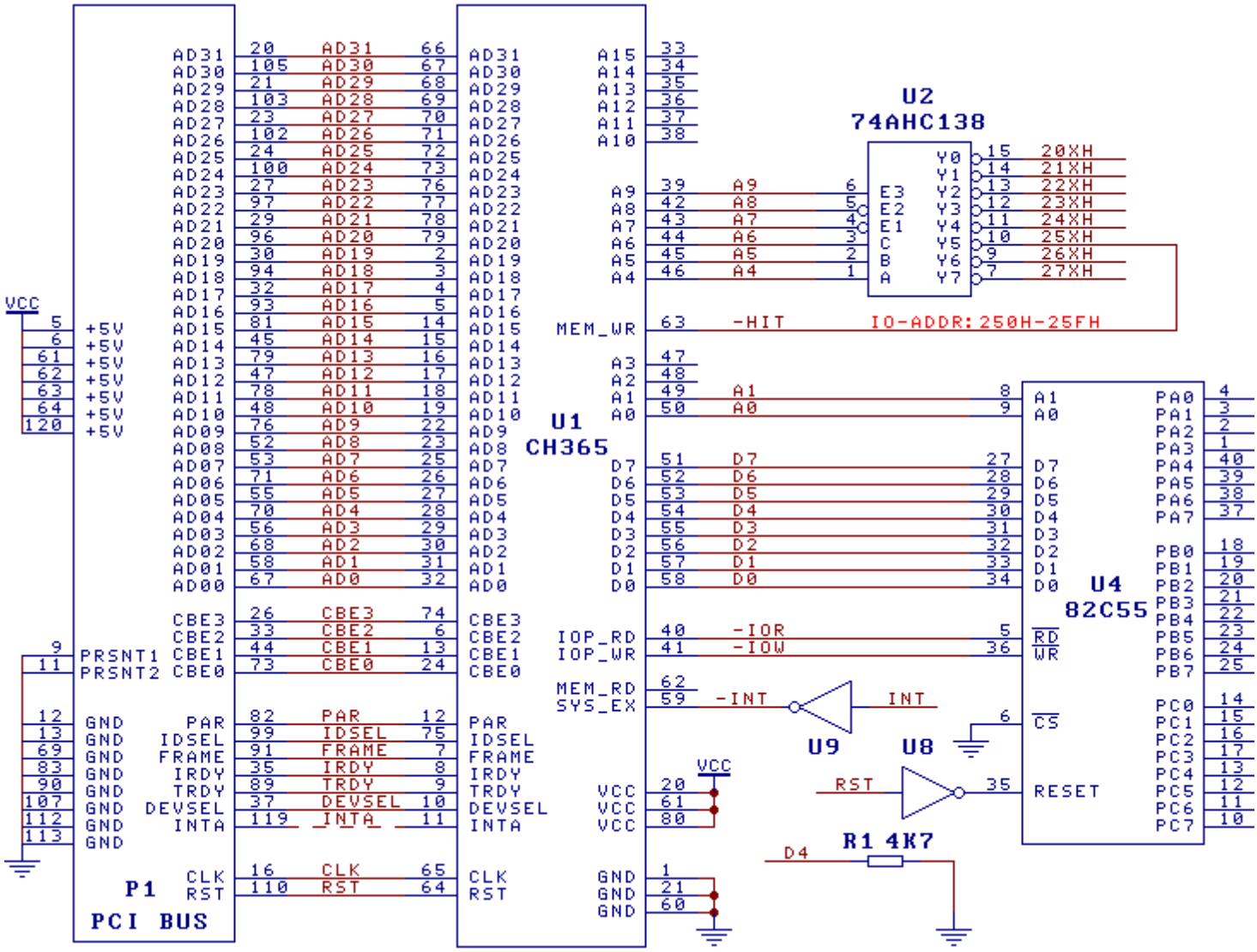
新的 PC 规范都去掉了 ISA 总线，很多以前设计的 ISA 板卡在新主板上无法使用，为了减少将 ISA 板卡移植到 PCI 总线的工作量，CH365 提供了本地硬件定址功能。具体的功能介绍在 CH365 手册中有详细说明，这里辅以原理图进行实例分析。

实例是一个 I/O 端口控制卡，通过 8255 与外部设备交换数据。板卡 I/O 端口的地址由译码器 U3 和 U6 进行级联译码，地址范围为 250H-25FH，由于 8255 只占用 4 个 I/O 地址，所以实际程序中只会使用 250H-253H 地址读写 8255。中断请求 INT 是高电平有效，该信号通常由单片机等器件根据需要进行驱动，一般电路中不一定需要中断。

2.1. 原 ISA 板卡的原理图



2.2. 新 PCI 板卡的原理图（升级后）



2.3. 升级说明

为了方便板卡制造商的小批量加工，CH365 尽量减少芯片的引脚，从而保持较大的引脚间距。同时为了支持较多的功能，CH365 对某些引脚进行了复用，并通过工作模式设定来选择复用引脚的实际功能。

PCI 板卡中，电阻 R1 用于工作模式设定，将 CH365 的数据线 D4 默认置为低电平，所以 CH365 的 MEM_WR 引脚被用于本地硬件定址请求输入 IOP_HIT。如果希望升级 ISA 板卡，但是又不希望修改任何应用程序，那么必须使用电阻 R1。本地硬件定址主要用于在不修改应用程序的前提下升级 ISA 卡。

译码器 U2 将 CH365 提供的地址进行译码，按图中连接的 I/O 地址是 250H-25FH。当计算机存取 I/O 端口地址 250H-25FH 时，译码电路输出有效，本地硬件定址输入-HIT 出现低电平，所以 CH365 执行 I/O 操作，输出低电平有效的-IOR 或-IOW 信号。由于 CH365 要求译码速度快于 20nS，所以 U2 选用比 74LS138 速度快的 74ALS138 或者 74AHC138，或者采用一个可编程器件，例如 16V8 等，译码速度最好小于 20nS。

在 ISA 板卡中，U7（8255）的-CS 是由译码电路驱动的；在 PCI 板卡中，由于 CH365 的-IOR 和-IOW 只在计算机存取指定的 I/O 端口地址（250H-25FH）时才有效，相当于已经经过初步的地址译码，所以 U4（82C55，不能用第一代 8255，速度太慢）的-CS 可以接地，保持一直有效，或者将剩余地址线 A2-A0 译码后再进一步产生-CS 信号。

PCI 总线的复位信号是低电平有效，所以通过反相器 U8 向 U4 提供与 ISA 总线相同的高电平有效的复位信号。

CH365 支持中断，具体请参考 CH365 的手册。由于 PCI 中断与 ISA 中断差别较大，如果 PCI 板卡中需要使用中断功能，那么必须修改应用程序，所以一般不会同时使用本地定址功能。在 CH365 的数据线 D3 上连接一个下拉电阻 R2 用于工作模式设定，可以使 CH365 的 SYS_EXT 引脚用于中断请求输入 INT_REQ，中断请求是低电平有效。原 ISA 中断请求经过反相器 U9 后作为 CH365 的中断请求-INT，但中断的具体工作方式有区别；当需要中断功能时，应该将 CH365 芯片的第 11 脚（PCI_STOP 或 INTA）连接到 PCI 总线的第 119 脚（INTA），图中以虚线表示此线根据需要连接。当不接该下拉电阻 R2 时（默认情况），CH365 不提供中断功能，SYS_EX 引脚为可以直接控制的输出信号线。

如果不需要中断功能（没有下拉电阻 R2），则上述 PCI 板卡具有与原 ISA 板卡相同的功能。一般情况下，原 ISA 板卡的应用程序可以不加修改，直接用于移植后的 PCI 板卡。

如果原 ISA 板卡工作时不需要驱动程序，那么新 PCI 板卡也不需要驱动程序。如果在 WINDOWS 中提供找到新硬件，可以不必安装驱动程序，或者安装一个假的驱动程序（提供一个不指定驱动程序的 INF 文件）。如果是新开发的 PCI 板卡或者原 ISA 板卡就需要驱动程序，那么可以参考 CH365 提供的通用驱动程序和动态链接库，在应用层或者驱动层实现所需要的功能。

PCI 总线的 I/O 端口地址一般是自动分配的，设备之间不会冲突；当采用本地硬件定址功能时，应该选用空闲 I/O 端口地址，防止与其它设备冲突。

如果不采用本地硬件定址功能，则原 ISA 板卡的应用程序应该作少量修改，包括：获取计算机自动分配给 CH365 的 I/O 端口基址；将 I/O 端口基址加上偏移地址后作为应用程序中各个 I/O 操作的端口地址。同时，PCI 板卡的本地二级译码电路可以简化甚至去掉，本实例中就可以直接去掉 U2 译码电路。

3、PCI 板卡软件设计

3.1. 升级 ISA 板卡

对于只使用 I/O 端口资源的 ISA 板卡，通过 CH365 的本地硬件定址功能一般都能做到 PCI 板卡直接使用原 ISA 板卡的软件，而不要修改任何程序。

对于使用存储器资源的 ISA 板卡，例如双口 RAM 接口的应用，在硬件成本适当增加的情况下可以不修改软件，简单方法是在计算机中多插一个“存储器地址设定卡”（在 BIOS 阶段运行），或者在计算机启动前执行设置存储器基址的工具程序 CH365MEM.EXE（在操作系统启动时执行），该方法通常用于初始测试，建议测试通过后针对 PCI 设计修改原 ISA 板卡的软件。

对于使用中断资源的 ISA 板卡，通常必须修改软件。

虽然原 ISA 板卡通过本地硬件定址功能或者“存储器地址设定卡”（或者工具程序）升级到 PCI 而可以不需要修改软件，但是在技术能力允许时建议修改软件，原因是：

① 由于 I/O 端口地址或者存储器地址是固定的，如果在同一台计算机插入多张同样的板卡，就需要使用跳线等办法选择各板卡的地址以避免冲突。

② 本地硬件定址所需要译码电路或者“存储器地址设定卡”增加了总成本，而设置存储器基址的工具程序只能用于 DOS 或者 WINDOWS 98 操作系统。

③ PCI 总线提供了 64KB 的 I/O 地址空间，而 ISA 只有 1KB，PCI 总线提供了 4GB 的存储器地址空间，而 ISA 通常只有 16MB，PCI 总线提供了至少 22 个中断（与操作系统和主板芯片组有关），而 ISA 通常只有 15 个中断。

④ 不能充分发挥 PCI 总线即插即用、灵活方便的特性。

3.2. 设计真正的 PCI 板卡

这里所指的真正的 PCI 板卡主要是指不使用本地硬件定址功能和“存储器地址设定卡”的 PCI 板卡，以下统一简称为 PCI 卡。

3.2.1. 基本原理

PCI 卡与 ISA 卡的一个重要区别就是：ISA 卡由设计人员作主，自由选择 I/O 端口地址、存储器 MEM 地址、中断号；而 PCI 卡由计算机（实际是由 BIOS 和操作系统 OS）自动分配 I/O 地址、MEM 地址以及中断号。

任何一个标准的 PCI 卡都有配置空间，其中通常有多个配置寄存器，其中一些寄存器是基址寄存器，例如 I/O 基址寄存器、MEM 基址寄存器、中断号寄存器。

基址寄存器是随时可以读写的寄存器 RAM，PCI 芯片厂家不能干涉也不能事先假定其内容（显卡是个例外，因为显示在 BIOS 之前工作）。以 I/O 基址寄存器为例。

⑴ PCI 芯片例如 CH365，具有 I/O 基址寄存器和 I/O 空间使能位（在命令寄存器中），一个 PCI 芯片可以有多个 I/O 基址寄存器，但是 I/O 空间使能位只有一个。当 I/O 空间使能位为 0 时，CH365 忽略任何 I/O 操作，所有 I/O 基址寄存器都没有作用，而当 I/O 空间使能位为 1 时，CH365 才有可能执行 I/O 操作。

⑵ 当计算机复位时（实际是 PCI 总线复位时），I/O 基址寄存器复位到 0000H，I/O 空间使能位也复位到 0。由于此时 I/O 空间使能位为 0，所以 CH365 不会执行任何 I/O 操作。

⑶ 当 BIOS 初始化时，向 CH365 的 I/O 基址寄存器中写入基址，例如 9400H，然后将 CH365 的 I/O 空间使能位设置为 1。对于其它 PCI 卡，BIOS 也执行相同操作，但是各个 PCI 卡的基址寄存器中的数值一定是不同的。

⑷ 由于 I/O 空间使能位为 1，所以当 CPU 执行 I/O 指令时，CH365 对 I/O 地址译码比较，如果发现 I/O 指令的地址与 I/O 基址寄存器相同，也是 9400H，就会响应并执行该 I/O 操作。由于 CH365 的 I/O 偏移地址范围是 00H-0FFH，所以 CH365 的 PCI 卡的 I/O 地址范围为 9400H-94FFH，只要 CPU 执行在此地址范围内的 I/O 操作，CH365 都会响应。

⑸ 其它 PCI 卡被 BIOS 分配的 I/O 基址数值与 CH365 不同，所以不会响应 I/O 地址为 9400H-94FF 的 I/O 操作，当然也就不会冲突。

⑹ 进入操作系统例如 WINDOWS 后，WINDOWS 也可以修改 I/O 基址，而一般情况下，WINDOWS 是保持 BIOS 自动分配给各个 PCI 板卡的基址。

⑺ 如果 WINDOWS 将 CH365 的 I/O 基址修改到 DC00H，那么在修改之后，只有当 CPU 在 DC00H 到 DCFFH 地址读写 I/O 时，CH365 才会响应并输入输出数据。

⑻ 应用程序也可以自己修改 CH365 配置空间中的基址寄存器，重新定位 CH365，前提是不要与其它 ISA 或者 PCI 卡的地址冲突。

⑼ 由此可见，PCI 板卡的 I/O 地址的译码由一个基址寄存器和一个比较器构成，改变基址寄存器中的数值就可以改变 PCI 板卡所占用的 I/O 地址，所以 PCI 板卡的 I/O 地址完全是动态分配的，而不是 ISA 板卡的固定物理连接。

⑽ 对于存储器 MEM，PCI 卡例如 CH365 也有 MEM 基址寄存器和相应的 MEM 空间使能位，其功能与用途与 I/O 基址类似。但是有一个重要的区别，就是 BIOS 或者操作系统（例如 WINDOWS）为 PCI 板卡自动分配的 MEM 基址通常都在 1MB 以上，而 DOS 下的程序不方便读写 1MB 以上的内存，所以如果 PCI 板卡在 DOS 下应用，可以由应用程序自己修改 MEM 基址，将其设置在 1MB 内存以下。“存储器地址设定卡”和设置存储器基址的工具程序就是基于这个原理将使用 MEM 资源的原 ISA 板卡升级到 PCI 的。

⑾ 对于中断，PCI 卡例如 CH365 也有中断号寄存器，但是该中断号寄存器没有物理作用，只是用于让操作系统或者应用程序知道当前使用的中断号。PCI 板卡只管向 PCI 总线申请中断，BIOS 或者 WINDOWS 会通过主板芯片组控制 PCI 路由，将 PCI 板卡申请的中断信号路由到相应的中断号，所以应用程序直接修改中断号寄存器通常没有作用。在使用 WINDOWS XP 和较新主板芯片组的计算机中，PCI 板卡的中断号还可以是 16 到 23，而不仅仅局限于 ISA 总线的 0 到 15。

⑿ 综上所述，PCI 板卡的 I/O 地址、MEM 地址、中断号完全是动态分配的，也是非常灵活的，在不同的计算机中，不同的 BIOS 下，不同的操作系统下，以及不同的 PCI 插槽中，同一块 PCI 板卡的 I/O 地址范围、MEM 地址范围和中断号都有可能是不同的。

3.2.2. 软件设计

产品设计人员可以获得自己设计的 PCI 板卡的 I/O 地址、MEM 地址、中断号。

根据原理中的说明，I/O 地址就在配置空间的 I/O 基址寄存器中、MEM 地址就在配置空间的 MEM

基址寄存器中。注意，I/O 基址寄存器的低 4 位通常是无效的，其值与基址无关，所以应该屏蔽掉，另外，MEM 基址寄存器的低 8 位也是无效的。

在没有操作系统时或者在 DOS 下，应用程序可以通过 PCI 中断 INT 1AH 调用，查询 PCI 板卡的配置寄存器，获得 I/O 地址、MEM 地址、中断号，并且在必要时可以重新设定 I/O 地址和 MEM 地址，前提是不能有地址冲突。

在调用 PCI 中断 INT 1AH 时，通常需要提供一个重要的参数，就是 PCI 板卡的物理地址，包括 BUS 号、DEVICE 号、FUNCTION 号，该值与 PCI 板卡所在的 PCI 插槽序号有关，通常靠近 CPU 的插槽中的 PCI 板卡的物理地址较小。

对于 PCI 扩展 ROM 应用,BIOS 会将 PCI 板卡的物理地址放在 AX 寄存器中传递给 ROM 初始化程序。对于 DOS 应用程序，通常需要自己通过搜索找到自己的 PCI 板卡，获得其物理地址。搜索也是由 PCI 中断 INT 1AH 提供，调用该中断前需要知道被搜索的 PCI 板卡的厂商识别码（Vendor ID）和设备识别码（Device ID）或者其它识别信息，而作为产品设计人员，必然已经知道自己所设计的 PCI 板卡的各种识别信息。

如果以 CH365 设计的 PCI 板卡的 I/O 基址寄存器中的数值为 E801H，那么该 PCI 板卡的 I/O 地址范围就是 E800H 到 E8FFH。

以下是 C 语言（TC 2.0）的示例，搜索 PCI 板卡。

```
union REGS mReg;
mReg.x.ax = 0xb102;
mReg.x.cx = 0x5049;  /* 默认的设备 ID，设计人员可以定义 */
mReg.x.dx = 0x4348;  /* 默认的厂商 ID，设计人员可以定义 */
mReg.x.si = 0;  /* 搜索第一个 */
int86 ( 0x1a, &mReg, &mReg );  /* 调用 PCI 的 BIOS */
if ( mReg.h.ah == 0 ) {  /* 调用成功，检测到 CH365 的 PCI 板卡 */
  dosCH365PciAddr = mReg.x.bx;  /* 保存 PCI 板卡的物理地址 */
  return( mReg.x.bx );  /* 返回 PCI 板卡的物理地址 */
}
 else /* 没有检测到 CH365 */
```

以下是 C 语言（TC 2.0）的示例，获取 I/O 基址，获取 MEM 基址方法类似。

```
union REGS mReg;
mReg.x.bx = dosCH365PciAddr;  /* PCI 卡的物理地址 */
mReg.x.ax = 0xb109;
mReg.x.di = 0x10;   /* I/O 基址寄存器在配置空间中的偏移地址 */
int86 ( 0x1a, &mReg, &mReg );  /* 调用 PCI 的 BIOS */
return( mReg.x.cx & 0xfff0 );  /* 返回 I/O 基址，低 4 位应该屏蔽 */
```

上述源程序可以在评估板资料或者产品光盘中的 CH365DOS.C 中找到。

在 WINDOWS 操作系统下,WINDOWS 屏蔽了很多硬件细节,所以不需要自己搜索和读取基址,WINDOWS 会主动将 I/O 基址和 MEM 基址及中断号通知该 PCI 卡的驱动程序。

如果使用 CH365 的通用 WDM 驱动程序和 DLL 动态链接库，应用程序完全不需要了解 PCI 板卡的 I/O 基址、MEM 基址等信息，只需要指定相应的偏移地址就可以读写 I/O 或者存储器，DLL 和 WDM 会自动加上基址。

PCI 板卡的应用程序设计与 ISA 板卡基本相同，只有 I/O 地址、MEM 地址不同，ISA 板卡的地址可能是事先定义的常数，而 PCI 板的地址是静态变量，在不同的环境下数值不同，但是在运行过程中保持不变，除非应用程序自己修改。

以下是 PCI 板卡的应用程序的必要步骤。

① 应用程序启动后获取可能需要用到的 I/O 基址、MEM 基址、中断号，可以参考上述方法，如果使用 CH365 的驱动程序和 DLL，那么直接调用相关 API。

② 将基址等保存在相应的变量中，例如 I/O 基址保存在 IO-BASE-ADDR 中。

③ 如果 CH365 接了一个 82C55,82C55 的 CS 直接接地,由于 82C55 占用 4 个偏移地址 00H-03H，所以将变量 IO-BASE-ADDR 加上 3 就可以向 82C55 写控制字，将变量 IO-BASE-ADDR 加上 1 就可以读写 82C55 的 B 口。

④ 实际 I/O 地址或者 MEM 地址不需要知道，因为在不同机器中，甚至不同插槽中，I/O 基址都有可能不同。如果一个计算机中同时插了 3 块相同的 PCI 板卡，那么它们的应用程序相同，只是 I/O 基址、MEM 基址不同。

3.2.3. PCI 中断共享

PCI 中断与 ISA 中断的最大区别在于 PCI 中断支持共享。

PCI 中断在硬件上是多个 PCI 板卡的中断请求线的“逻辑或”，PCI 中断是电平中断，支持中断共享的 PCI 板卡必须提供中断激活标志位，以便软件识别出是否中断。

假定 PCI 卡 A 和 PCI 卡 B 共用一个中断号 12，那么，卡 A 要中断或者卡 B 要中断时，CPU 都会收到 12 号中断请求，12 号中断描述符 IDT 只有一个，不过，为了支持中断共享，操作系统会建立一个链表，分别是卡 A 的中断服务程序和卡 B 的中断服务程序。

当 12 号中断产生时，由中断描述符 IDT 进入操作系统的中断处理程序。操作系统根据链表首先调用卡 A 的中断服务程序。

卡 A 的中断服务程序检查卡 A 自身的中断激活标志，如果是卡 A 的中断，那么卡 A 的程序就让卡 A 的硬件撤消中断请求，（PCI 是电平中断，所以要记忆直到中断服务，完成后要撤消），并且清除卡 A 硬件寄存器中所记忆的中断激活标志，并且然后执行中断服务，完成后退出卡 A 的中断服务，并通知操作系统“这是卡 A 的中断”。操作系统收到卡 A 的通知，则结束本次 12 号中断服务。

如果卡 A 的程序检查自身的中断激活标志，发现不是自己产生的中断，那么直接退出卡 A 的中断服务程序，并通知操作系统“这不是卡 A 的中断”，所以操作系统将查询链表，继续调用卡 B 的中断服务程序。

如果卡 B 的程序检查卡 B 自身的中断激活标志，发现是自己产生的中断，那么就会让卡 B 的硬件撤消中断请求以及清除激活标志，执行中断服务，然后通知操作系统“这是卡 B 的中断”，操作系统结束本次 12 号中断服务。

WINDOWS 屏蔽了硬件中断的最底层的细节，各种板卡的驱动程序只是通过相应的 API 接口以 WINDOWS 定义的格式处理“经过 WINDOWS 预处理”的硬件中断，例如，中断结束命令是 WINDOWS 处理的，而在 DOS 下是应用程序在中断退出时向 20H 端口写 20H。

并非所有中断号都支持共享，也并非所有 PCI 卡都支持中断共享，Windows 允许个别 PCI 卡独立占用中断。

3.2.4. 偏移地址

CH365 提供的 I/O 偏移地址是 00H-0FFH，所以在执行 I/O 操作时，CH365 同时提供 8 位地址线，外部电路可以对 A7-A0 进行偏移地址译码，产生进一步的二级片选。

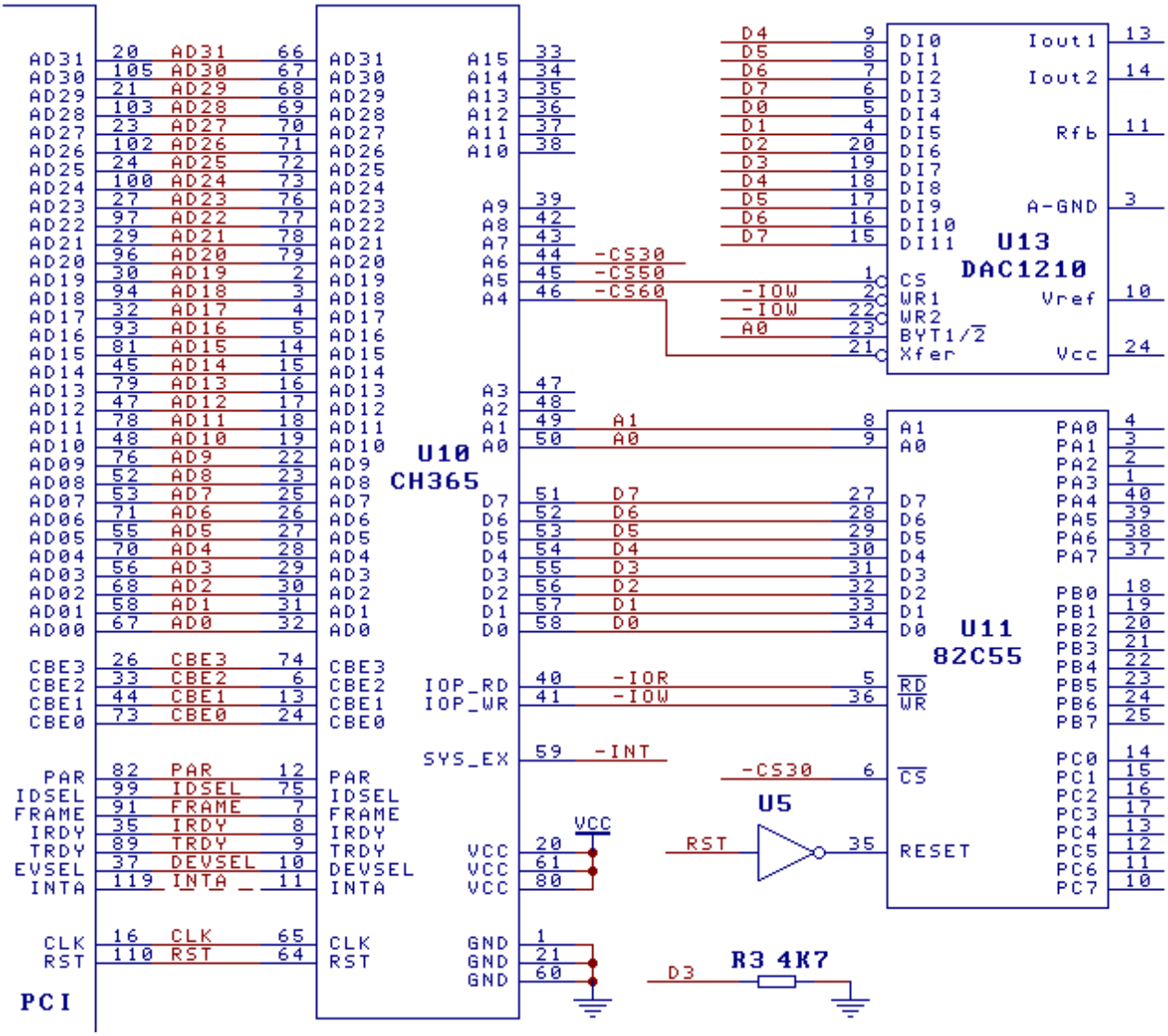
例如，计算机执行指令 OUTPORTB(0x1234,0x56)，则实际是向地址为 1234H 的 I/O 端口写出数据 56H。如果 CH365 的 I/O 基址是 1200H,则 CH365 响应 1200H-12FFH 之间的所有操作。对于地址 0x1234，CH365 的 IOP_WR 输出低电平，同时从 A7-A0 输出低 8 位地址 34H。如果外部电路有两组电路，例如两个 82C55，那么可以用 CH365 的 A7-A2 中的任何一个作为二级片选区分各自的偏移地址。

实际上，为了支持兼容 ISA 总线的本地硬件定址，CH365 在 I/O 操作过程中，总是提供 10 位 I/O 地址，也就是说，CH365 将 0x1234 地址的低 10 位从 A9-A0 输出，而 CH365 的 A15-A10 保持原先的值不变，因为外部电路不需要考虑高 6 位地址。

在存储器操作中，由于 MEM 容量较大，外部电路需要用到 A14-A11，所以 CH365 会通过 A14-A0 输出存储器操作中的低 15 位地址，只有 A15 保持不变。

4、PCI 板卡的电路图（图可能不完整，仅供参考）

4.1. 数模转换 DAC 与 I/O



说明：

① 如果不需要中断功能，那么应该去掉电阻 R3，CH365 的 SYS_EX 引脚应该悬空。

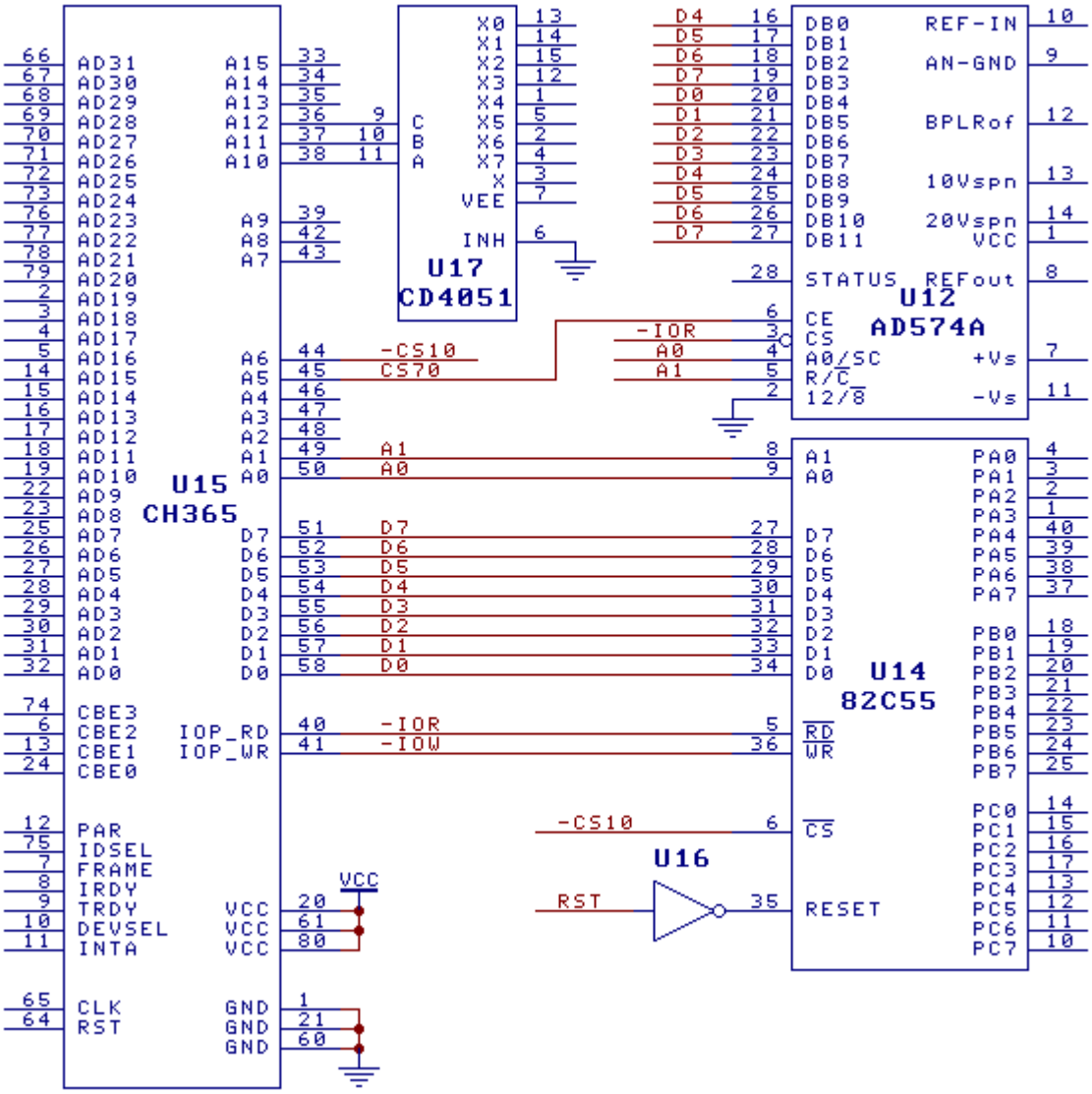
② 图中 U11 用于 I/O 输入输出控制，由于普通 8255 速度较慢，所以建议选用 82C55。

③ CH365 的负载有两个：U11/82C55 和 U13/DAC1210。当负载较多时，可以将 A4-A6 用一个 3-8 译码器（例如 74LS138）译码成 8 个二级片选。当负载较少时，例如本图中，可以直接用 CH365 的 A4-A6 引脚作为 3 个二级片选，都是低电平有效。其中：A4 作为片选的 I/O 偏移地址范围是 60H-6FH，A5 作为片选的 I/O 偏移地址范围是 50H-5FH，A6 作为片选的 I/O 偏移地址范围是 30H-3FH，实际 I/O 地址还要加上 I/O 基址。

④ 按图中的连接，82C55 的 A 口偏移地址是 30H，B 口偏移地址是 31H，C 口偏移地址是 32H，控制口的偏移地址是 33H。

⑤ 按图中的连接，DAC 数据输出方法是：先向偏移地址 51H 写入高 8 位数据，再向偏移地址 50H 写入低 4 位数据，最后向偏移地址 60H 写任意数据产生传送控制信号，用于将刚刚写入的 12 位数据传送到 DAC 寄存器。

4.2. 模数转换 ADC 与 I/O



说明：

① 因为 CH365 与 PCI 插槽之间的连接是标准电路，所以在本图中被省略。

② 图中 U14 用于 I/O 输入输出控制，由于普通 8255 速度较慢，所以建议选用 82C55。

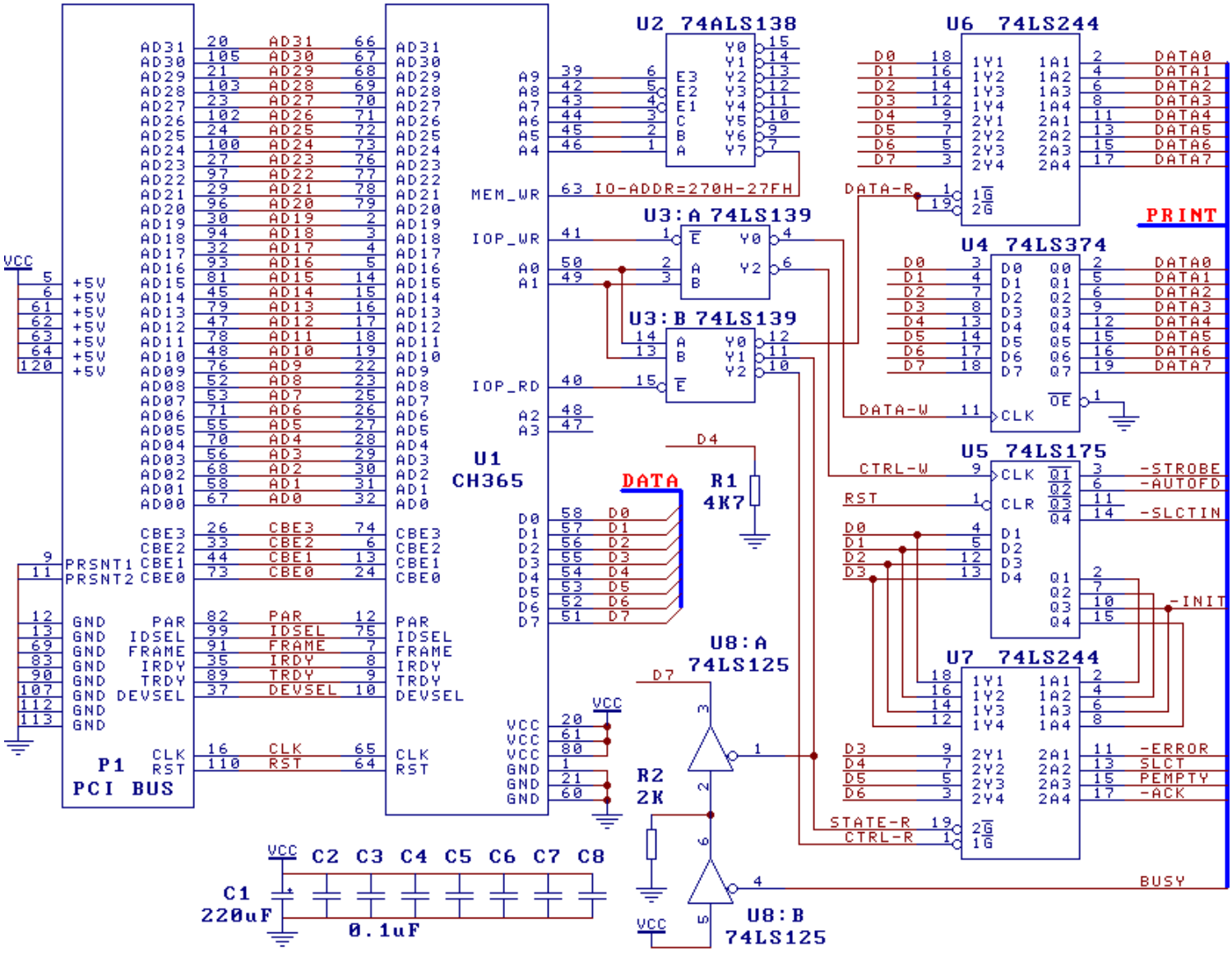
③ 图中用 CH365 的 A5 和 A6 作为外部电路的片选，A5 片选是高电平有效，A6 片选是低电平有效。其中：A5 作为片选的 I/O 偏移地址范围是 70H-7FH，A6 作为片选的 I/O 偏移地址范围是 10H-1FH，实际 I/O 地址还要加上 I/O 基址。

④ 按图中的连接，82C55 的 A 口偏移地址是 10H，B 口偏移地址是 11H，C 口偏移地址是 12H，控制口的偏移地址是 13H。

⑤ 按图中的连接,CH365 的 IOR 信号作为 U12 的低电平片选信号 CS,所以只有 CH365 的 A5=1 和 IOR=0 同时有效时，AD574 才会被选中。读取 I/O 偏移地址 70H 则启动 12 位 AD 转换，读取 I/O 偏移地址 71H 则启动 8 位 AD 转换，读取 I/O 偏移地址 72H 则获取转换结果的高 8 位数据，读取 I/O 偏移地址 73H 则获取转换结果的低 4 位数据。

⑥ CH365 的 A15-A10 都是可以自由独立控制的输出信号，并且是锁存后输出的信号。U17 用于选择 AD 通道，U17 的控制信号由 CH365 的地址线 A12-A10 提供。

4.3. 标准 PCI 打印口（单向并口）



说明：

① 因为 CH365 与 PCI 插槽之间的连接是标准电路，所以在本图中被省略。

② 电路中的 R1 用于启用 CH365 的本地硬件定址功能，所选 I/O 地址范围是 270H-27FH。

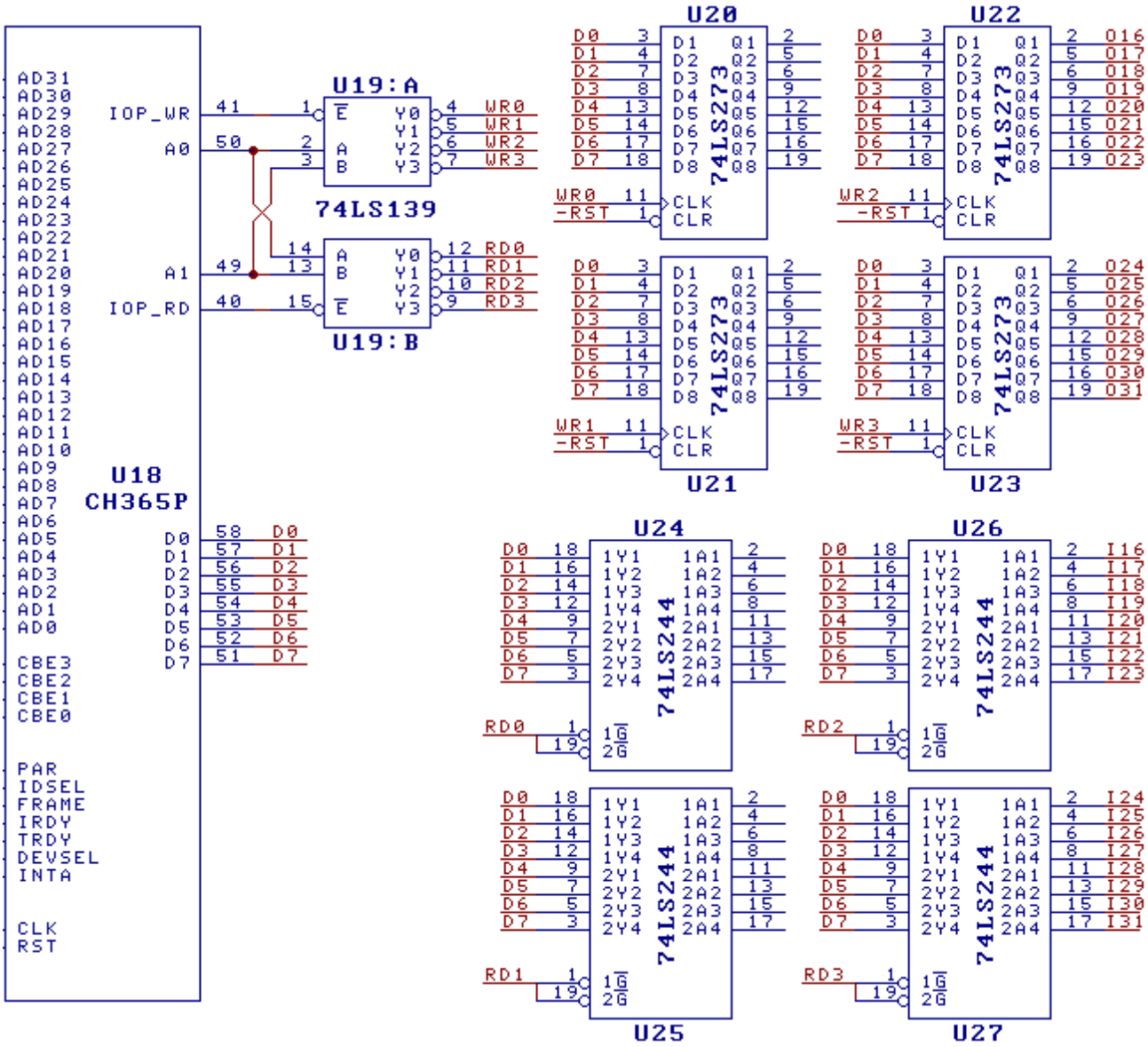
③ 该电路实现了标准的第二打印机端口，I/O 地址是 278H，实际地址还有 270H 等，由于 PCI 中断与 ISA 有所不同，所以该电路没有提供中断功能。

④ 该电路中简化了打印机口的输入输出，实际通常需要增加一些上拉电阻、下拉电阻、串接电阻和一些并联的抗干扰电容。

⑤ 为节约所用 IC，电路中由 U8:B 和 R2 实现 BUSY 的简单反相。

⑥ 该电路根据实际需要可以有很多变化或者一些改进。

4.4. 数字 I/O 控制卡（32 位输入和 32 位输出）



说明：

① 因为 CH365 与 PCI 插槽之间的连接是标准电路，所以在本图中被省略。

② 译码器 U19 对 2 位地址译码，为读和写分别产生 4 个本地片选，分别控制 4 个 I/O 芯片。

③ 由 U20/U21/U22/U23 组成 8*4=32 位数字信号输出，如果需要三态输出，可以选用 74LS374 和 574 等，图中使用 74LS273 可以支持上电复位。

④ 由 U24/U25/U26/U27 组成 8*4=32 位数字信号输入，如果需要输入锁存，可以选用 74LS373 和 573 等，图中使用 74LS244 或者 74LS245，自带上拉电流，悬空输入默认为高电平，如果选用 74HC244 或者 245 则不带上拉。

⑤ 该电路中简化了数字信号的输入输出，实际通常需要增加一些上拉电阻、下拉电阻、串接电阻和一些并联的抗干扰电容。

⑥ 该电路根据实际需要可以有很多变化或者一些改进。例如，减少为 16 位输入和 16 位输出，用两个 74LS138 代替 74LS139 实现 64 位输入和 64 位输出，等等。

4.5. 其它电路

CH365 连接单片机或者 DSP/MCU 等的电路

① 通过双向缓冲接口芯片 CH421 连接，请参考 CH421 芯片手册。
② 通过 I/O 扩展芯片 8255 连接，请参考 CH365 芯片手册中有关说明。
③ 通过双口 RAM 连接，双口 RAM 与普通 RAM 类似，请参考 CH424 芯片和 CH365 芯片手册中 RAM 连接。
④ 通过 4 位并口、I^2C 接口、SPI 接口等进行低速连接，请参考相关的完整说明文档。

CH365 连接 ROM 芯片或者大容量 FLASH 闪存的电路

① 连接容量 1KB 到 32KB 的 ROM 芯片，请参考 CH365 芯片手册。
② 连接容量 32KB 到 128KB 的 ROM 芯片，请参考 CH36X 扩展 ROM 说明文档。
③ 连接容量 128KB 以上的 ROM 或者 FLASH 闪存，请参考 CH36X 电子盘/终端卡方案。

CH365 设计接口卡的电路

① 设计 PCI 总线的 CAN 总线接口卡，请参考相关的完整说明文档。
② 设计 PCI 总线的 RS485 接口卡，可以参考单片机连接方案后由单片机扩展。
③ 设计数字 I/O 控制卡，可以参考 CH365 芯片手册中有关 I/O 扩展的说明，以及前面的例图。

5、常见问题和其它说明

5.1. 原理图

① 原理图中的 PCI 端口是针对 32 位的 PCI 总线，所以没有 CBE4-CBE7、AD32-AD63 等用于 64 位 PCI 的扩展引脚。
② 原理图中的 PCI 引脚计数是从元件面（B 侧）的左边（小）向右边（大），然后接着到焊接面（A 侧）的右边（大），再到左边（小）。
③ 图中所标 PIN61、PIN62 实际上是《PCI 结构》B 侧的 PIN61 和 PIN62。图中所标 PIN63、PIN64 实际上是《PCI 结构》A 侧的 PIN62 和 PIN61。图中的 PIN63-PIN121 对应于《PCI 结构》A 侧的 PIN62-PIN1。
④ PCI 保留引脚，有两种可能：以前版本定义的引脚，后来不用了，但又不能作为其它用途，所以保留；考虑以后升级或者可能有其它用途，但现在还不知道是什么用途，所以先保留。
⑤ 建议用 82C55、82C55A 代替 8255 芯片。CH365 输出的 RD 和 WR 的脉冲宽度是 240nS，可以设置为更小，但不能设置为更大，如果外部电路的响应速度较慢，那么就会不稳定。标准的 8255 芯片，速度是 400nS 而不能稳定工作，但是换成后期版本的相同功能的 82C55A 芯片，就可以正常工作，82C55A 的速度通常是 200nS 以下。

5.2. 两线串行接口

⑴ 为什么 CH365 的 SCL 输出频率是 260KHz？

因为绝大多数的兼容 I2C 接口都支持 400KHz 的速度（400KHz 以下都可以接受），只有个别的老设备才要 100KHz 以下。而 PCI 总线的 33MHz 分频 1/128 就是 260KHz。

⑵ CH365 的 SCL 和 SDA 是否需要上拉电阻?

CH365 的 A15 或者 SYS_EX 作为 SCL 时，都是输出线，所以完全不需要上拉电阻或者下拉。CH365 的 D7 是双向信号线，作为 SDA 信号线，必须要有上拉电阻，默认只有 CH365 内部提供的 40K，如果信号线分布电容较大（例如总线挂的 I2C 设备较多），那么 40K 对电容的充电时间较长，SDA 信号的波形的上升沿就不好，考虑到信号波形问题和抗干扰问题，可以仅在 D7 信号线使用上拉电阻（几 K），从而加快 SDA 信号线的充电速度。

⑶ I^2C 总线要求：SDA 线上的数据必须在时钟的高电平保持稳定；数据线的高或低电平状态只有在 SCL 线的时钟信号是低电平才能改变。CH365 是否符合？

因为作为 SDA 的 D7 是总线信号，变化比较频繁，所以应该在 I2C 空闲时将 SCL 默认置为低电平，

所以根据 SCL 信号线的不同，有两种可能性：

① 如果使用 A15 作为 SCL 信号线，那么最好将 D0 默认下拉，通过工作模式设定使 A15 复位后默认为低电平，而如果数据线 D0 没有下拉电阻，则 A15 在复位后默认为高电平，也就是 SCL 为高电平。

② 如果使用 SYS_EX 作为 SCL 信号线，那么 SYS_EX 信号线就不能用作其它用途，不过，SYS_EX 信号线复位后默认总是低电平，所以用于 SCL 时不需要任何特殊处理。在 SYS_EX 用于 SCL 的情况下，如果仍然需要硬件中断功能，也可以参考网站上关于 CH365 中断功能的说明另外使用一个三级管电路代替。

5.3. 其它问题

⑴ 为什么工具 DEBUG365 显示的存储器地址（或者由调用 DLL 动态库 API 获得的存储器地址）与在 WINDOWS 设备管理器中看到的存储器地址不一样？

由于 WINDOWS 支持虚拟内存，所以应用程序甚至驱动程序都应该使用虚拟线性地址，而不能直接使用物理地址，在 WINDOWS 设备管理器中看到的存储器地址是真正分配到的物理地址，而 DEBUG365 以及 DLL 提供的存储器地址都是经过映射后的虚拟线性地址。

⑵ 用 CH365 升级 ISA 卡需要修改我原来 ISA 卡的应用程序吗？

一般情况下，如果原 ISA 卡只使用了 I/O 端口资源（支持 I/O 读写），那么不需要修改原应用程序。大多数 ISA 卡属于这种情况。

如果使用了存储器资源（例如双端口 RAM），那么应该修改原应用程序，也可以不修改应用程序，但是需要额外硬件或者软件配合，参考本文其它叙述。

如果使用了中断资源，那么需要修改原应用程序。

⑶ 用 CH365 的本地硬件定址功能升级了 ISA 卡后，需要安装 CH365 的驱动吗？

实际上不需要，但是为了应付 WINDOWS 的提示，可以装个假的驱动，真的更好。

5.4. 简单调试

在 WINDOWS 下，可以使用评估板资料或者产品光盘中的 DEBUG365 等程序，在 DOS 下可以使用 DEBUG、CH365MEM 等工具。

通常由 BIOS 自动分配的 I/O 基址、MEM 基址在 WINDOWS 中不会变化，进入 WINDOWS 后可以在[我的电脑]的属性中查看硬件设备列表，找到 PCI 卡后，可以查看 PCI 板卡的 I/O 基址、MEM 基址等。

在 DEBUG 中，可以通过 I 和 O 命令直接以字节为单位读写 CH365 的 I/O 端口，例如，I/O 基址是 DC00H，那么 I DC01 命令就是读取 I/O 偏移地址 01H 的 I/O 端口。

CH365 提供的 MEM 与计算机的内存读写方式完全相同，如果 CH365 连接了双口 RAM，则在 DOS 下，可以先用 CH365MEM.EXE 工具将 CH365 的 MEM 地址设置为 1MB 以下，然后再由 DEBUG 进行读写，CH365MEM.EXE 与“存储器地址设定卡”原理相同。

CH365MEM 不能在 WINDOWS 中的 DOS 下运行，因为 CH365 默认的 MEM 地址是 PC 机自动分配的，可能在 4GB 内存顶端,例如 D8000000H,此时 D000H 到 DFFFH 区域是空闲的,所以进入 WINDOWS 后,WINDOWS 的 EMM386 等程序就会占用 D000H 到 DFFFH,所以在 WINDOWS 的 DOS 中,再用 CH365MEM.EXE 即使将 MEM 段地址修改到 D000H，仍然无法使用（已经被 WINDOWS 占用了）。但是如果使用“存储器地址设定卡”，由于在启动 WINDOWS 之前就先占用了 D000H 到 DFFFH，所以 WINDOWS 不会占用 D000H 到 DFFFH。

如果是通过 DEBUG 测试,那么可以在纯 DOS 环境下,或者在 WINDOWS 启动之前,执行 CH365MEM.EXE,将 CH365 的 MEM 段地址修改到 D800H，然后再读写 D800H 段，就可以正常了，这时再进入 WINDOWS，WINDOWS 也不会占用 D800H，所以 WINDOWS 下也可以读写 CH365 的 MEM。

在 WINDOWS 下无法直接读写计算机物理内存，所以需要通过驱动程序实现，在驱动程序中可以直接读写 CH365 的 MEM。CH365 的 DLL 提供了 WINDOWS 下的 API，可以通过驱动程序读取 I/O 和存储器。