PD6729 — A PCI Motherboard PC Card (PCMCIA) Controller Solution

Application Note

May 2001

As of May 2001, this document replaces the Basis Communications Corp. document *AN-PD7*.

Application Note

Contents

Figures

Tables

1.0 Introduction

The PD6729 is a single-chip PCMCIA interface controller capable of controlling two PCMCIA or compact Flash sockets, respectively. It is designed for use in embedded applications and notebook systems where reduced form factor and low power consumption are critical design objectives.

Current typical application examples include:

- Routers
- Access network servers
- PBXs
- Vending machines
- Portable handheld systems
- Data acquisition systems
- Settop boxes
- Integrated access devices
- DSLAMs
- Terminal servers
- Point of Sale terminals
- Navigation systems
- Measurement equipment

With the PD6729, a complete dual-socket PCMCIA solution with power-control circuitry can occupy less than 2 square inches (13 square centimeters) of board space.

The PD6729 controller is completely compatible with the standards of PCMCIA (Personal Card Memory International Association) Release 2.0 Standard as well as JEIDA (Japan Electronic Industry Development Association) Version 4.1 Standard, and PCI 2.1. The PD6729 controller also offers special power-saving features such as Automatic Low-power Dynamic Mode and Suspend Mode. The PD6729 is a true mixed-voltage device that can operate at +5 volts, +3.3 volts, or a combination of these at various interfaces. The controllers have full internal buffering and require no additional circuitry to interface to the PCI Bus or to PCMCIA sockets.

2.0 Overview

This application note provides information to help system designers interface the PD6729 to the PCI bus. For information on how to design the PD6729 PC Card (PCMCIA) Controller in a non-Intel® environment, contact an applications engineer.

This application note presents specific pin connection information for motherboard and PCI bus solutions. The following PD6729 controller interfaces are detailed:

- PCI bus interface
- PC Card socket interface
- PC Card socket power control signals
- Power distribution pins

Section 3.0, “Hardware Description” on page 7 describes the PD6729 PCI bus and PC Card socket interfaces with enough detail to permit a successful interface between the PCI bus and a PC Card connector. The power-saving modes of the PD6729 are discussed in Section 4.0, “Power-Saving Modes” on page 14.

Section 5.0, “Windows” on page 16 discusses PC Card address windowing basics, and Section 6.0, “Timing Registers” on page 21 presents an example of how the timing register is set up for a SRAM card.

“Appendix A” on page 26 presents schematics of a PD6729 PCI bus implementation, including the power-control circuitry.

Note: These sample circuits are provided for demonstration purposes only and do not represent the minimum component count achievable in a motherboard solution.

An OrCAD 4.1 database of this implementation is available. Customers can modify this database to meet the specific needs of their design.

3.0 Hardware Description

This section describes the signals of the PCI bus interface, PC Card socket interface, PC Card socket power control, and power distribution.

3.1 PCI Bus Interface

The PD6729 PCI bus interface signals are compliant with the PCI Specification Version 2.1, for drive capability and can be directly connected to the PCI bus. The PD6729 PCI pins are given the same as (or similar) conventional PCI bus signal names so that connections can be easily identified. In the PCI architecture, there are four required signal groups: address and data, interface control, error reporting, and system signals. Arbitration signals are only required if the device is master capable (and the PD6729 is a slave device). The following section describes how to interface these PCI pins to the PD6729.

- The following pins, with the exception of the Arbitration pins, are directly connected to the PCI bus:
- Address and data pins: **AD[31:0], C/BE[3:0],** and **PAR**
- Interface control pins are directly connected to corresponding PCI bus signals: **FRAME#, TRDY#, IRDY#, STOP#, DEVSEL#,** and **IDSEL#**
- System pins: **CLK** and **RST#**
- Error reporting pins: **PERR#** and **SERR#**
- Arbitration pins: **REQ#** and **GNT#** (not required because the PD6729 is a slave device)

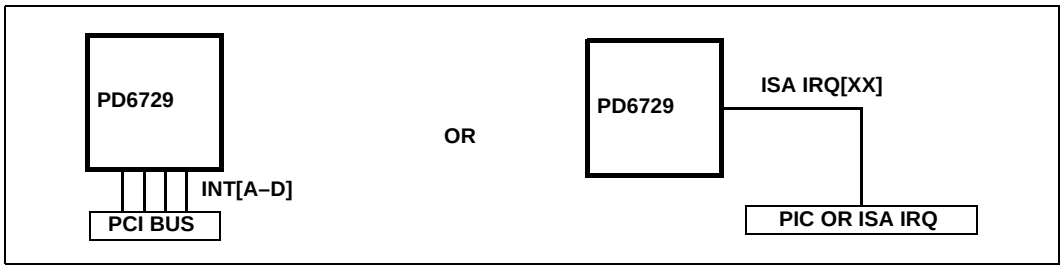
3.1.1 Optional Pins

Interrupts

The PD6729 can be configured to support either PCI (active-low) or ISA (active-high) interrupts. The configurable pins are: **INTA#/IRQ3**, **INTB#/IRQ4**, **INTC#/IRQ5**, and **INTD#/IRQ7**. In ISA interrupt mode, the PD6729 can support an additional six ISA interrupts, for a total of ten.

Depending on the system design, the designer can elect to support either ISA or PCI interrupts. One reason the ISA-interrupt scheme is more practical is that some software may not require modification. Most software applications expect each specific device to use specific interrupts. Take as an example, a modem configured to operate at COM2. The software application program would expect a COM2 device to use **IRQ3**. The software program may provide another ISA interrupt to be selected instead of **IRQ3**, but typically the selection of a PCI interrupt is not allowed. Figure 1 illustrates the PCI and ISA interrupt schemes.

Figure 1. ISA or PCI Interrupt Connections



The **PCI_CLK** signal from the PD6729 is directly connected to the system clock, which provides the timing for the transaction that occurs on the bus.

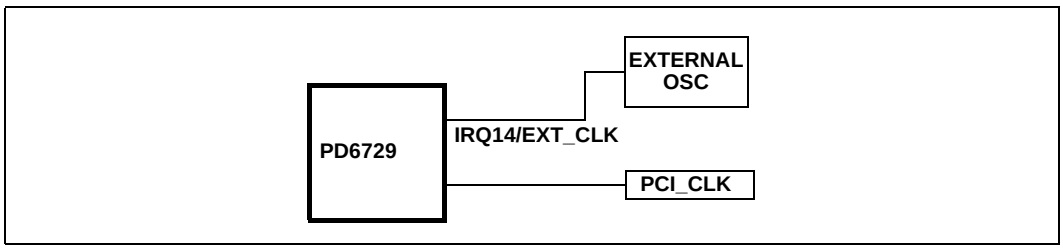
The PCI Specification Version 2.1 states that the clock can operate in a range from 0–33 MHz and that, when bus transactions are complete, the system could stop the clock. This may pose a problem for the PD6729 controller because the PD6729 incorporates a write FIFO. Data can be stored in this FIFO when the system determines to stop the clock.

To incorporate this functionality, implement one of the following steps:

1. **Ensure that all PD6729 operations complete before stopping the system clock (check the FIFO status to ensure that it is empty).**
2. **Add an external clock for when the system stops the PCI clock.**

When the system clock is stopped, the PD6729 has the capability to finish FIFO cycles with an external clock. The multiplex pin, **IRQ14/EXT_CLK**, can be configured as either an interrupt or an external clock input. When configured as an external clock, this input to the PD6729 uses the external clock to provide the internal timing to finish transactions to the PC Card socket when the system clock is stopped. This configuration is diagrammed in Figure 2.

Figure 2. External Clock Connection



3.1.2 Optional Signals

The PD6729 provides additional signals that a system designer can elect to implement in the system design. These signals are described below.

The **IRQ[15,14,12:10]** signals are ISA interrupts that indicate either Management or General Control interrupts. These interrupt lines are an alternative method to service interrupts generated by the host or card, but not through normal PCI interrupt requests.

SPKR_OUT* functions as an output to drive a speaker. On the PD6729, the output of SPKR_OUT* is the logical XNOR of the speaker inputs from each socket interface. In this mode, SPKR_OUT* can be used as an amplifier input to drive a speaker.

LED_OUT* functions as an open-drain driver to show when there is activity to/from the PC Card sockets.

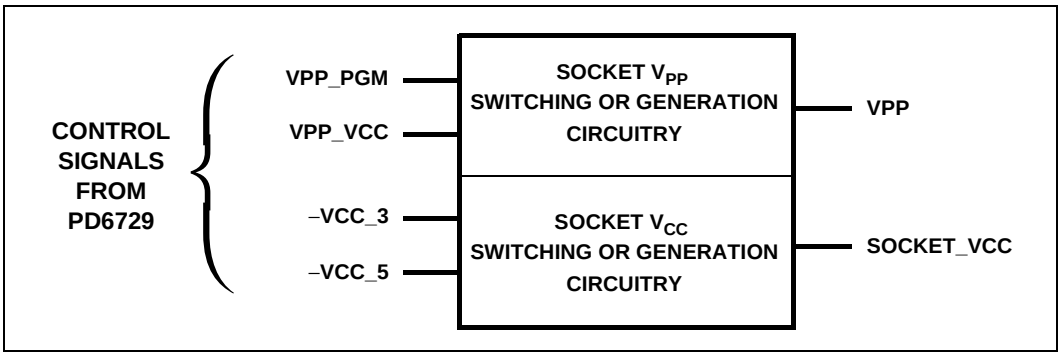
3.2 PC Card Socket Interface

The PD6729 host controller has a fully buffered PC Card socket interface. There is no need for additional socket buffering. All socket interface signals can be directly connected to the PC Card connector.

3.3 PC Card Socket Power Control

The PD6729 power-control signals control the PC Card socket power circuits. The signals enable the power from the PC Card socket power source to the socket **VCC** (the power to the card) and **VPP** (the programming and peripheral voltage) power pins. A description of these signals and how they are used is discussed in this section. Figure 3 is a block diagram of the V_{CC} and V_{PP} signals.

Figure 3. Block Diagram of Card-Socket Power Supply/Switching Circuit



–VCC_3 and –VCC_5

The PD6729 **–VCC_3** and **–VCC_5** control signals enable the power-switching circuitry that selects the power source applied to V_{CC} pins 17 and 51 of each PC Card socket. Both -VCC_3 and -VCC_5 are active-low signals, mutually exclusive of each other. The Miscellaneous Control 1 (Index 16h) and the Power Control (Index 2h) registers control -VCC_3 and -VCC_5. (See the *PD6729 Data Sheet* for more information about registers and register function.) A low on -VCC_3 causes the PC Card socket power-supply circuit to provide +3.3 volts to the card socket V_{CC} pins 17 and 51; a low on -VCC_5 enables +5 volts to be provided to card socket V_{CC} pins 17 and 51.

When both –VCC_3 and –VCC_5 on the socket interface are inactive (high), the PC Card power supply disconnects power sources from the card socket V_{CC} pins. Unlike other PC Card host controllers, there is no need to clamp a card socket V_{CC} pins to ground when both –VCC_3 and –VCC_5 are inactive (high); special interface circuitry in the PD6729 prevents card damage in this situation.

VPP_VCC and VPP_PGM

The PD6729 card socket interface **VPP_VCC** and **VPP_PGM** signals control the voltage level to the V_{PP} pins at the associated PC Card socket. To be PC Card-compatible and function with both memory and I/O cards, the V_{PP} switching/generation circuitry should be able to supply 0 volts, the selected card V_{CC} voltage of +3.3 or +5 volts, and +12 volts to the card socket V_{PP} pins. Both the VPP_VCC and VPP_PGM signals are active-high signals, mutually exclusive of each other. A high on VPP_VCC enables the PC Card socket V_{PP} switching/generation circuit to provide the card socket current V_{CC} voltage (as determined by the –VCC_3 and –VCC_5 control signals) to the card socket pins 18 and 52; a high on VPP_PGM enables +12 volts to be provided to card socket pins 18 and 52.

The PCMCIA Standard requires that a card socket V_{PP} supply pins be logic ‘0’ when programming voltages are not being applied. To accomplish this select a V_{PP} switching or generation circuit that forces the V_{PP} output to 0 volts when the card socket interface VPP_VCC and VPP_PGM signals are inactive (low). Alternatively, a bleed resistor can be placed from the card socket V_{PP} pins to ground, although this method may create unwanted power consumption for power-sensitive applications.

The schematics in Appendix A show a sample design on implementing the power-control supply logic for V_{CC} and V_{PP} to the PC Card socket in a system where +12 volts is already available.

VS1 and VS2

The PD6729 host controller provides two voltage sense pins per socket, **VS1** and **VS2**. These pins determine the card voltage level. The state of these pins is stored in the PD6729 internal registers and used by software to determine the voltage level applied to the card. No external resistors are required for the board design; these pins are internally pulled-up by the PD6729 host controller. If VS1 and VS2 are the voltage selects, the pins are connected as follows: VS1 directly connects to pin 43 of the PC Card socket; VS2 directly connects to pin 57 of the PC Card socket.

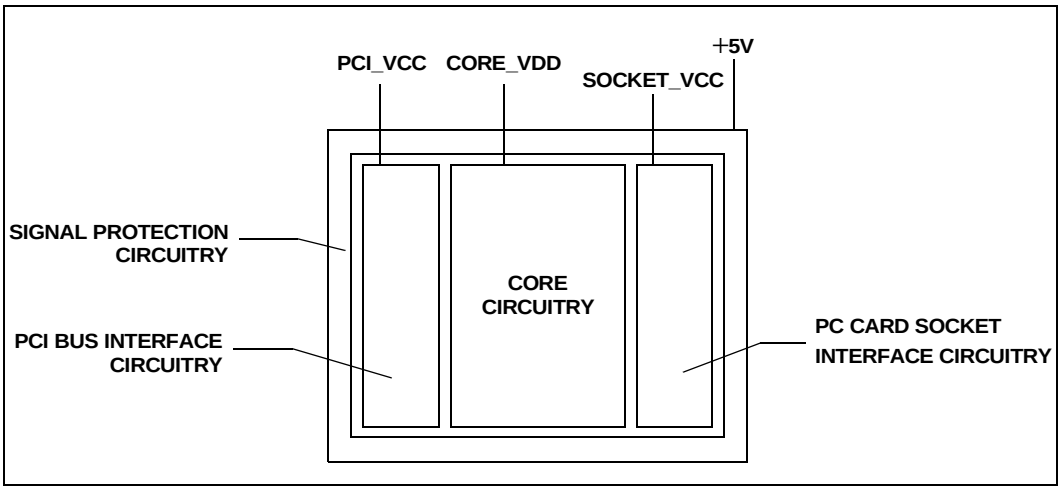
For further explantation of the VS1 and VS2 levels, refer to the PC Card Standard Specification.

3.4 Power Distribution

The PD6729 has four types of power-distribution pins that provide power to various parts of the circuitry (Figure 4). These power-distribution pins are the PCI_VCC, CORE_VDD, and SOCKET_VCC power pins, and the +5V power pin.

Having separate types of power pins allows the PD6729 to be used in a wide variety of applications where PC Card voltages are dynamically changing between +3.3 and +5 volts or are powered on and off.

Figure 4. PD6729 Powering of Internal Logic



PCI_VCC

The **PCI_VCC** power pins power the PCI bus interface circuitry. Connect the PCI_VCC power pins to the same power source that drives other PCI bus signals. When PCI_VCC is at +3.3 volts, the PCI bus interface signals operate at +3.3 volts. When PCI_VCC is at +5 volts, the PCI-bus-interface signals operate at +5 volts.

CORE_VDD

The **CORE_VDD** power pins supply power to the main core circuitry of the PD6729. Since CORE_VDD can be set at +3.3 or +5 volts independent of the operating voltage at the PCI bus or card socket interfaces, set CORE_VDD to +3.3 volts to reduce system power consumption. If there is no +3.3-volt supply available in the system, a regulator or diode circuit can step-down the V_{DD} voltage for power saving. The PD6729 PCI signals are CMOS, and the voltage applied to CORE_VDD determines the threshold levels to the PCI bus signals. If TTL threshold levels are required on the system PCI bus, the CORE_VDD (core voltage) is set to +3.3 volts. This makes the CMOS threshold level equivalent to TTL threshold levels.

SOCKET_VCC

The **SOCKET_VCC** power pins supply power to the socket-interface pins and should be powered from the same card power circuit output routed to the corresponding socket V_{CC} pins. This allows the respective socket interface signals to operate at the same voltage as the card operating voltage.

The PD6729 has two separate socket interfaces, each SOCKET_VCC pin can operate independent of the other socket interface. This prevents signal mismatch problems when, for example, a card in socket A is operating at +5 volts while a card in socket B is operating at +3.3 volts or is powered off.

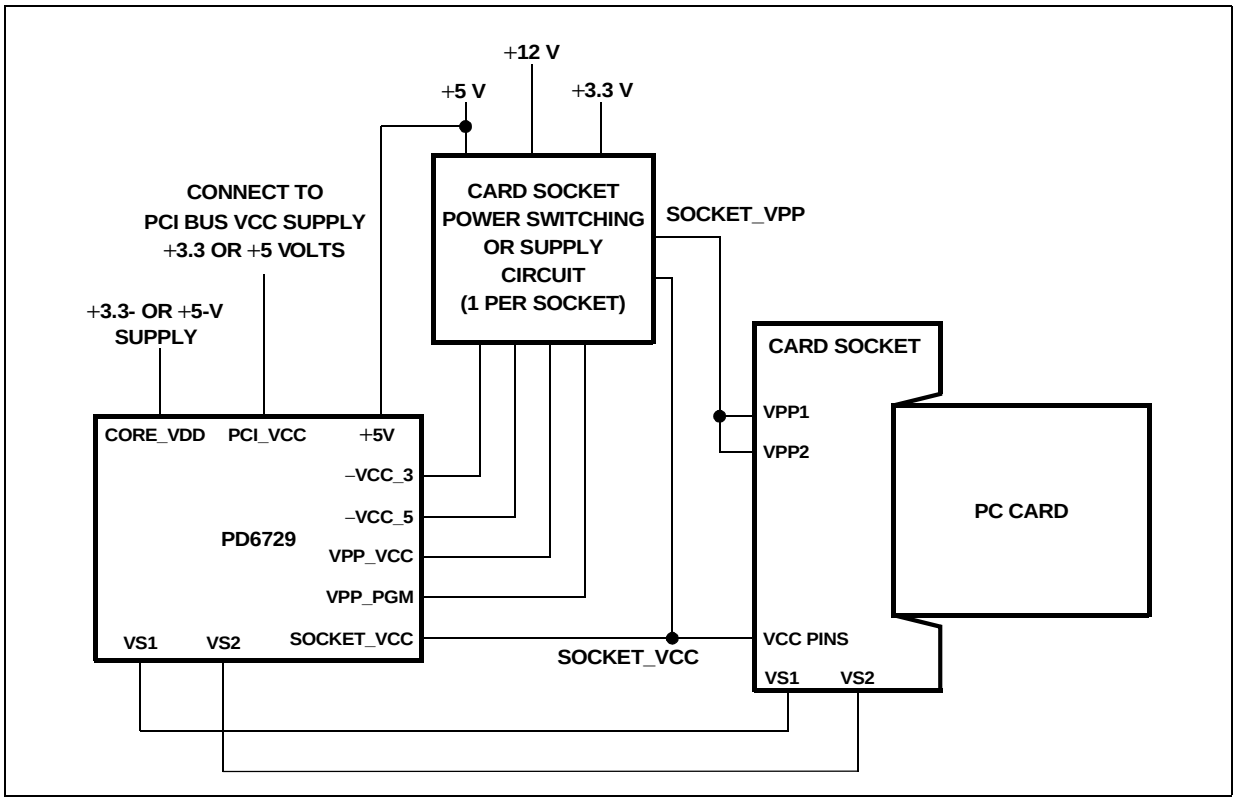
+5V

The power pin labeled +5V must be connected to the system +5-volt power supply. While the system +5-volt supply is not always powered on, the +5V pin must be connected to a supply where the voltage is at the same or higher than any of the voltages provided to the PD6729 other power pins.

Caution: If the other supply voltages (PCI_VCC, CORE_VDD, or SOCKET_VCC) are raised more than +0.5 volts above the voltage applied to the +5V pin, latch-up and/or damage to the PD6729 may result. Connect the PD6729 power pins as previously described to prevent any potential problematic situations. However, carefully analyze each design for the possibility of latch-up conditions.

Figure 5 is a block diagram of the host and socket interfaces.

Figure 5. Typical Power Distribution for the PD6729



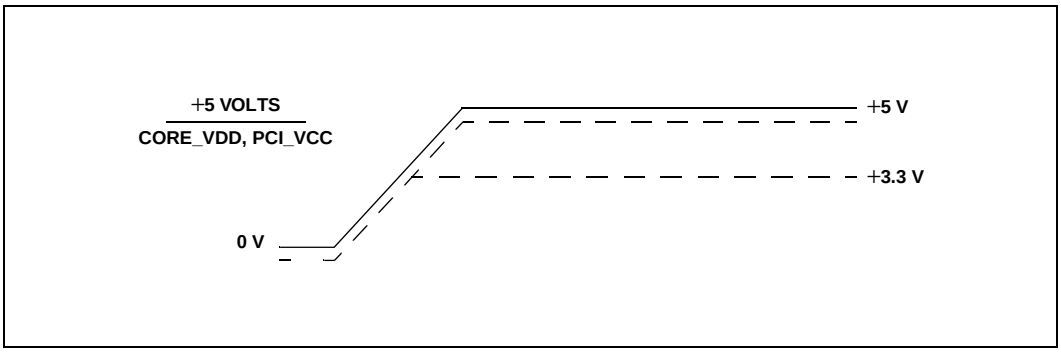
3.5 Power Sequencing

The correct power sequencing of the +5V, CORE_VDD, and PCI_VCC lines ensures that the +5V line is not more than 300 mV below any of the other pins on the device. This means that as the voltage supplies ramp-up, the +5V line should start concurrent to or before the other power supplies and ramp with or faster than the other power supplies. There is no timing requirement for this level difference so it is not necessary to bring the +5V line up and hold it for a length of time

and then apply the other power supplies. Instead, the +5V line *must* stay at or ahead of (or a maximum of 300 mV behind) the other power supplies. Figure 6 is an example of the PD6729 power plane.

Note: All signals input and output must not exceed +0.5 V above the voltage applied to the +5V pin.

Figure 6. Power Plane



4.0 Power-Saving Modes

In addition to the normal operating mode, which consumes very little power, the PD6729 supports two power-saving modes to further reduce power consumption of the PC Card (PCMCIA) Controller. Refer to the *PD6729 Data Sheet* for a full description of power-management features.

4.1 Automatic Low-Power Dynamic Mode

The Automatic Low-Power Dynamic mode is transparent to normal system operation and is enabled after chip reset by default. After reset, the host is configured for Automatic Low-Power Dynamic mode. This mode can be turned off by a write of '0' to bit 1 of Miscellaneous Control 2 register (Index 1Eh). When system is inactive in Automatic Low-Power Dynamic mode, the internal clock is turned off to most of the device, and the PC Card address lines are held at static values; the socket V_{CC} and V_{PP} power control signals are not affected. Once activity occurs at the card socket (for example, -INTR activation or status pins change) or PCI bus cycles occur to either valid card addresses or in the PD6729 registers, the PD6729 restarts the internal clock. When PCI bus cycles stop accessing valid card address windows or the PD6729 internal registers, the PD6729 automatically stops the internal clocks again and holds the card socket interface outputs static.

Shown below is an example of a code segment that enable Automatic Low-Power Dynamic mode.

```
#define    67xxbase     0xXXX        // XXX is value of configuration index 10h
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e

// code fragment
outp (67xxindex, miscntrl2);
x = inp (67xxdata);
x |= 0x2;
outp (67xxdata, x);
```

4.2 Suspend Mode

When no card or device accesses are requested, Suspend mode increases power savings. The PD6729 is put into Suspend mode by a write of '1' to bit 2 of the Misc Control 2 register (Index 1Eh). In Suspend mode, the internal synthesizer and all internal clocks are turned off. The V_{CC} and V_{PP} power control pins to each card socket power-switching/supply chip set are left unchanged (the system power-management software *must* suspend power to the card socket before Suspend mode is initiated). When in Suspend mode, card socket status changes and card interrupts are still detected and system interrupts are still generated.

To power up from Suspend mode, set bit 2 of the Misc Control 2 register to '0'.

Shown below is an example of a code segment that enable Suspend mode

```
#define    67xxbase     0xXXX       // XXX is value of configuration index 10h
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e
```

```
// code fragment
outp (67xxindex, miscntrl2);
x = inp (67xxdata);
x |= 0x4;
outp (67xxdata, x);
```

4.3 Additional Notes on PC Card Subsystem Power Reduction

The PD6729 strictly controls power consumption during power-saving modes. The following measures can further minimize power consumption at the system level of the entire PC Card subsystem (including the PC Card power-switching/supply circuit and card).

1. Controlling power to the cards (Power Control register bits 5:0)
2. Enabling and disabling card interfaces (Power Control register bit 7)
3. External shut down of clock sources to the PD6729

5.0 Windows

The PD6729 provides both memory and I/O windows, allowing access to address regions suitable for use with PC Card memory and I/O cards. These address translation features are organized as five independent memory windows per socket, and two independent I/O windows per socket.

Typical memory and I/O windows are configured using PC Card- and socket-service software or setup utilities provided by the card vendor.

5.1 Memory Windows

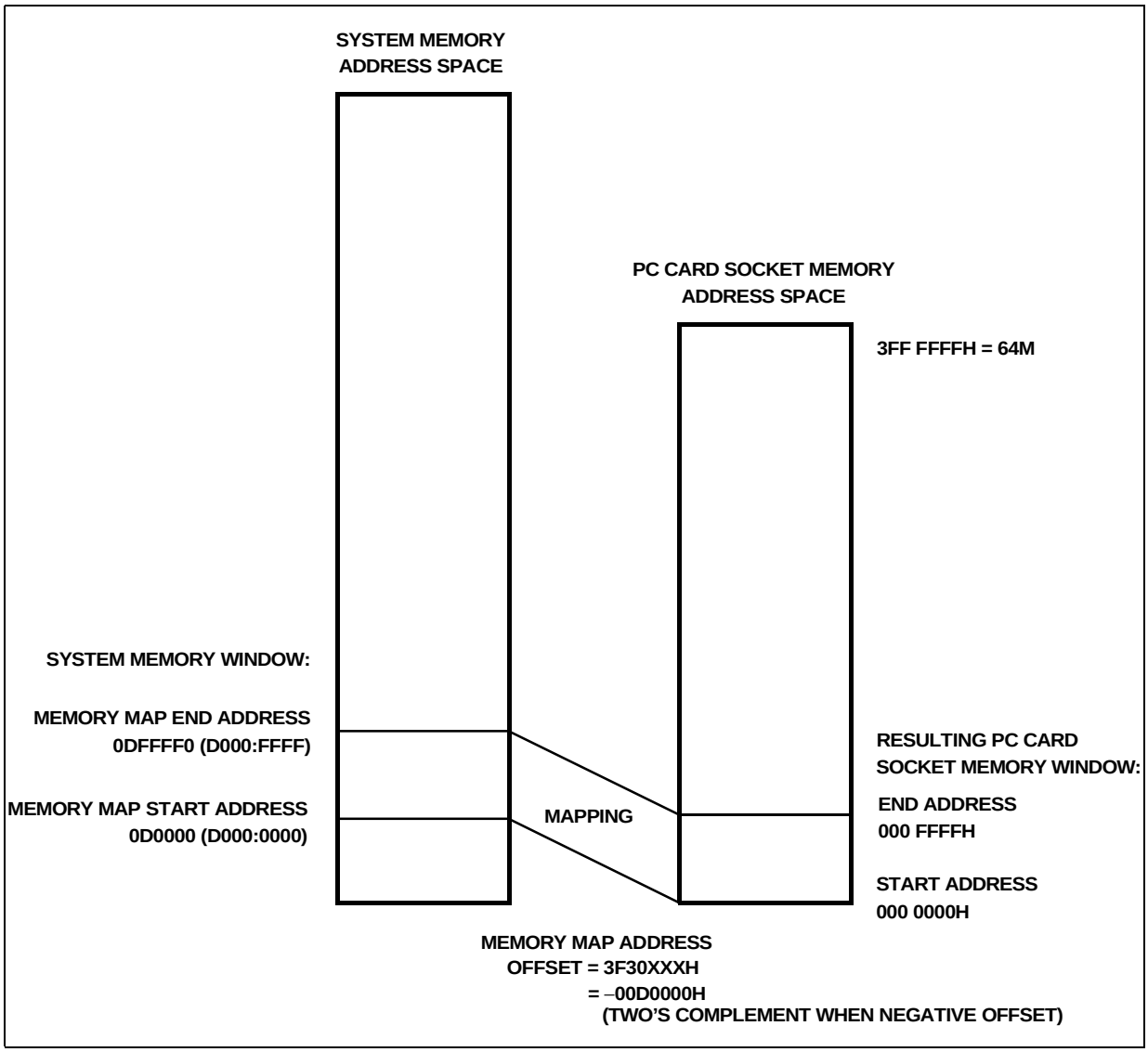
The PD6729 is configurable for up to five memory windows. These windows are accessed through the socket service software. The following is a typical example of how a window is defined and accessed.

Memory window sizes can be programmed on 4-Kbyte increment boundaries. Offsets for these windows are provided so that the actual PC Card access address is different than the system PCI bus address. For example, assume it is necessary to access a memory card in the PC Card socket. The memory card has accessible memory at PC Card addresses 0–0FFFF (64 Kbytes). A memory window could be specified in the computer space to start at 0D0000 (0D000:0000) and continue through 0DFFFF (0D000:FFFF). This allocates the memory space in the host (assuming it was not being used by the computer) to the PC Card (PCMCIA) Controller. To program the PC Card socket memory addresses to start at location 0000000h for PCI-bus-memory addresses starting at 0D0000 (0D0000:0000), an offset needs to be computed so that the PCI bus memory address, plus the window memory-mapped offset, equals the PC Card address (0000000h). To accomplish this, take the two's complement of D0000 (0D000:0000) to create the desired negative offset.

Note: The 4-Kbyte increment window resolution causes A[11:0] to map directly.

This computed offset value is then programmed into the selected window Memory-Mapped Address Offset High and Memory-Mapped Address Offset Low registers. PCI bus memory accesses are then at address 0D0000 (0D000:0000) (for example) to occur at PC Card address 0, and so on through the PCI bus memory map up to 0DFFFF (0D000:FFFF). Figure 7 demonstrates how the mapping is accomplished.

Figure 7. Window Mapping Example of System Memory Address to PC Card Socket Memory Address



Below is a code segment (related to the example in Figure 7) to program the PD6729 socket interface Memory Window 0 as a 1-Mbyte window with a starting system address at location 0D0000 (D000:0000h), and a negative offset to create a 1-Mbyte socket memory window starting at PC Card address 000 0000h:

```
#define    67xxbase     0xXXX        // XXX is value of configuration index 10h
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e

// code fragment

// Set socket power to Auto-power on when card is installed
```

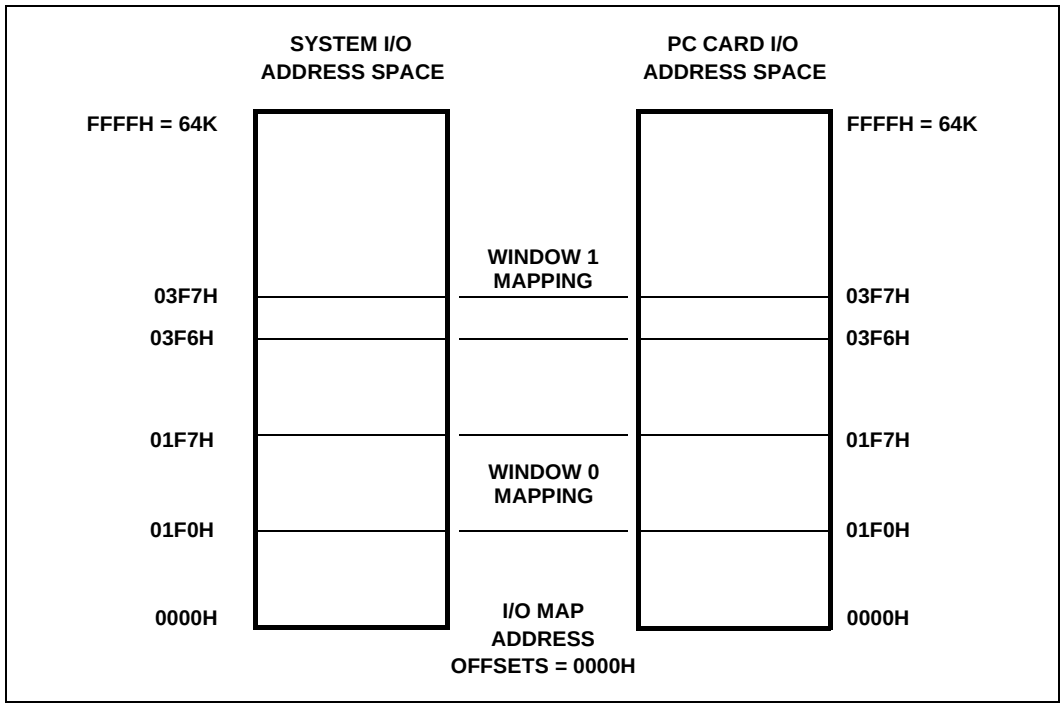
```
outp (67xxindex, 0x02);
outp (67xxdata, 0xb0);

// Disable memory window 0 during memory map window programming
outp (67xxindex, 0x06);
x = inp(67xxdata);
x &= 0xfe;
outp (67data, x);

// Program Memory Map 0 Start Address Low Byte (address lines A19-A12)
outp (67xxindex, 0x10);
outp (67xxdata, 0xd0)

// Program Memory Map 0 Start Address High Byte (address lines A23-A20) to all
zeros
// and enable 16-bit card memory cycles
outp (67xxindex, 0x11);
outp (67xxdata, 0x80);

// Program Memory Map 0 End Address Low Byte (address lines A19-A12)
outp (67xxindex, 0x12);
outp (67xxdata, 0xdf);

// Program Memory Map 0 End Address High Byte (A23-A20) and select card cycle
// Timing Register Set 0 for memory operations in this window
outp (67xxindex, 0x13);
outp (67xxdata, 0x00);

// Program Memory Map 0 Address Offset Low Byte (A19-A12)
outp (67xxindex, 0x14);
outp (67xxdata, 0x30);

// Program Memory Map 0 Address Offset High Byte (address lines A25-A20) and
// program Memory Window 0 to be common memory;  clear Write Protect to allow
// writes to card memory in Memory Window 0
outp (67xxindex, 0x15);
outp (67xxdata, 0x3f);

// Enable Memory Map 0, leave all other bits unchanged.
outp (67xxindex, 0x06);
x = inp (67xxdata);
x |= 1;
outp (67xxdata, x);
```

5.2 I/O Windows

The PD6729 supports up to two I/O windows per card socket. I/O-window mapping is similar to mapping memory windows. I/O window sizes are specified in even addresses increments on word boundaries. Offsets are provided from I/O windows so the actual address used to access I/O on a PC Card can be different than the system I/O address. A typical example of I/O window definition and access is described below.

Figure 8 shows two I/O windows programmed for IDE drive support with I/O-map offsets of zero. The first I/O window has a start address at 01F0h and a stop address at 01F7h. The second I/O window has a start address at 03F6h and a stop address at 03F7h.

Figure 8. Window Mapping Example of System I/O Addresses Without Offset to PC Card I/O Address



Following is a code segment related to the example in Figure 8.

```
#define    67xxbase     0xXXX     // XXX is value of configuration index 10h
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e

// code fragment

// Set socket power to Auto-power on when card is installed
outp (67xxindex, 0x02);
outp (67xxdata, 0xb0);

// Disable I/O windows 0 and 1 during I/O map window programming
outp (67xxindex, 0x06);
x = inp(67xxdata);
x &= 0x3f;
outp (67data, x);

// Program I/O Map 0 Start Address Low
outp (67xxindex, 0x08);
outp (67xxdata, 0xf0);

// Program I/O Map 0 Start Address High
outp (67xxindex, 0x09);
outp (67xxdata, 0x01);

//Program I/O Map 0 End Address Low
outp (67xxindex, 0x0a);
outp (67xxdata, 0xf7);
```

```
//Program I/O Map 0 End Address High
outp (67xxindex, 0x0b);
outp (67xxdata, 0x01);

// Program I/O Map 1 Start Address Low
outp (67xxindex, 0x0c);
outp (67xxdata, 0xf6);

// Program I/O Map 1 Start Address High
outp (67xxindex, 0x0d);
outp (67xxdata, 0x03);

//Program I/O Map 1 End Address Low
outp (67xxindex, 0x0e);
outp (67xxdata, 0xf7);

//Program I/O Map 1 End Address High
outp (67xxindex, 0x0f);
outp (67xxdata, 0x03);

// Enable I/O Map 0 and I/O Map 1
outp (67xxindex, 0x06);
x = inp (67xxdata);
x |= 0xcf;
outp (67xxdata, x);
```

6.0 Timing Registers

The timing registers program PC Card access timing for the following timing parameters: Setup, Command, and Recovery. When the PD6729 timing registers are programmed for card-specific requirements, meeting the card-specific timing is ensured.

The PD6729 has two timing register sets: Timing 0 and Timing 1. Each timing register set has a Setup Timing register, a Command Timing register, and a Recovery Timing register. Memory and I/O windows are individually programmable to use card access timing from either of the programmable timing register sets (Timing 0 or Timing 1).

Note: Do not modify the timing-set register values during PC Card bus cycles. Timing registers should not be changed until the Write FIFO is empty because the timing changes begin immediately and can corrupt any cycles in progress. This is verifiable by checking that the Empty Write FIFO bit (7) of the FIFO Control register (Index 17h) is set to '1'.

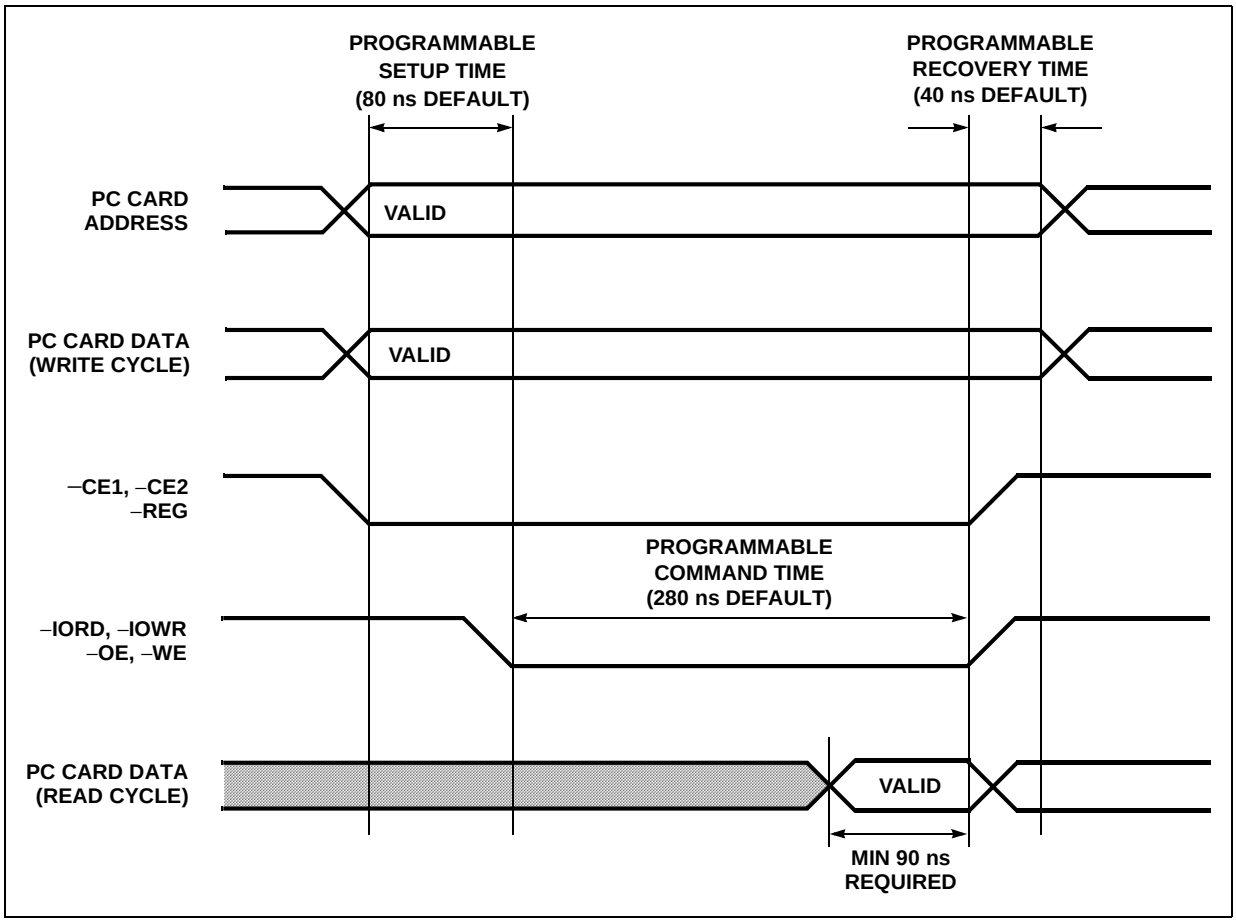
After reset, the Timing 0 register set defaults to the values shown in Table 1. The Command signals (that is, –WE, –OE, –IOWR, and –IORD) control Setup, Command, and Recovery. The window widths are derived from the falling and rising edges of these signals. The register setting for the Timing Set 0 register after reset is shown in Table 1. The timing diagram is shown in Figure 9 on page 22.

Table 1. Programmable Timing Set 0 Default Setting

Parameter	Register	Value	Time
Setup	3A	01	80 ns
Command	3B	06	280 ns
Recovery	3C	00	40 ns

NOTES:
1. Internal timing granularity = 1 ÷ PCI_CLK = 1 ÷ 25 MHz = 40 ns. If the PCI_CLK input is > 25 MHz, the Clock ÷ 2 bit in Index 1Eh, bit 4 must be set to '1'. The slot control interface is not designed to operate at frequencies above 25 MHz. Setting the Clock ÷ 2 bit to '1' accommodates PCI bus speeds > 25 KHz, but £ 33 Mhz. This doubles the timing to the PC Card socket.
   for example,
   PCI_CLK = 33 MHz (PCI_CLK > 25 MHz)
   Index 1Eh, bit 4 = 1
   Internal timing granularity = 1 ÷ (33 MHz ÷ 2) = 60 ns
2. The timing for this example is derived from a 25 MHz PCI_CLK input to the PD6729 controller.

Figure 9. Setup, Command, and Recovery Timing Diagram



6.1 Setting the Timing Registers for SRAM Card Example

The values in the timing registers can be altered to vary the width of the Setup, Command, and Recovery times for a particular application.

For example, consider programming a SRAM memory card that requires a minimum address setup time of 30 ns, a write pulse width of 150 ns, and a data recovery time of 50 ns.

6.2 Programming the Timing 1 Register Set

Each timing register has two variable fields to control the scale and duration of the timing parameter: prescalar value and the multiplier value. By using the prescalar value and the multiplier value, the timing parameter is calculated as follows:

$$\text{Timing parameter} = (N_{pres} \times N_{val}) + 1$$

Prescalar values are determined by bits 7 and 6 of each of the three timing registers comprising Timing Set 0 and Timing Set 1. These possible values are shown in Table 2.

Table 2. Prescalar Values

Timing Parameter Register		Prescalar Value
Bit 7	Bit 6	
0	0	1
0	1	16
1	0	256
1	1	4096

The multiplier increases the timing parameter by a factor of the selected prescalar. The multiplier value is determined by bits 5:0 of the three timing registers comprising Timing Set 0 and Timing Set 1. These six bits permit integer values from 0–63.

Based on the range of prescalar and multiplier values, the minimum timing parameter window width is 40 ns, and the maximum timing parameter window width is slightly greater than 20 ms.

To select a setup time parameter close to 30 ns, set bits 7 and 6 to ‘00’ (corresponding to a prescalar value of 40 ns) and, with a multiplier of zero, an address setup time of 40 ns is attainable.

$$\text{Setup Time} = (40\ \text{ns}) \times (0) + 40\ \text{ns} = 40\ \text{ns}$$

$$T_{CP} = 1 \div \text{PCI\_CLK frequency}$$

The Command and Recovery registers are similarly configured. After setting up the Timing 1 registers for the example described, the Setup, Command, and Recovery registers have the values shown in Table 3 and Figure 10.

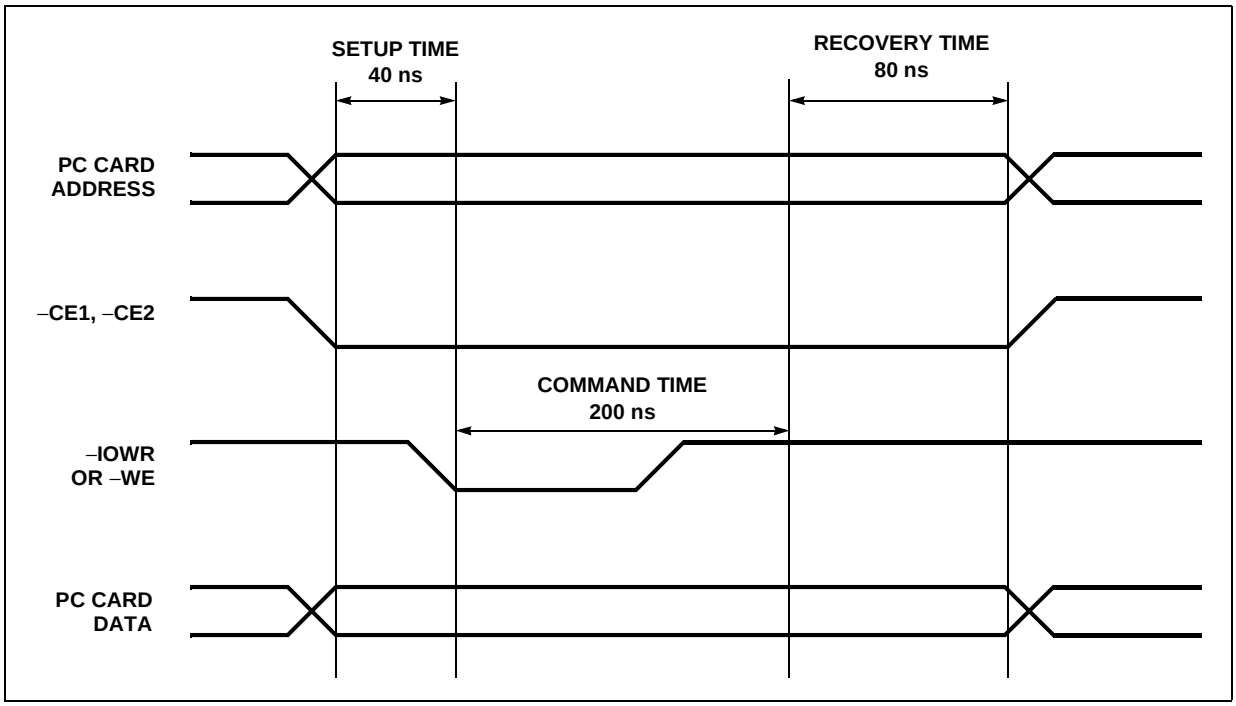
Table 3. Example of SRAM Card Write Timing Settings

Parameter	Register	Value	Time
Setup	3D	00	40 ns
Command	3E	04	200 ns
Recovery	3F	01	80 ns

NOTE:
1. The above example assumes a 25 MHz PCI CLK.
2. The internal clock period is the same as the PCI_CLK or EXT_CLK period if the MISC Control 2 register, bit 4 is ‘0’ and doubles the PCI_CLK or EXT_CLK if this bit is ‘1’.

Note: If PCI_CLK operates at a frequency > 25 MHz, the MISC Control 2 register, bit 4 *must* be set to ‘1’.

Figure 10. Timing Diagram of Example SRAM Card Write Timing Setting



Following is a code segment shows the code needed to program the Timing 1 register for SRAM timing examples:

```
#define    67xxbase     0xXXX    // XXX is value of configuration index 10h
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e
#define    TRUE         (1 ==1)
#define    FALSE        !TRUE

// code fragment
// function to check the fifo busy bit
BOOL check_fifo (void)
{
int I, x;
outp (67xxindex, 0x17);
for (I=0; I < 0x400; I++)
{
                        x = inp (67xxdata);
                        x &= 0x80;
                        if ( xxx == 0x80) return (TRUE);
}
return (FALSE);
}

// main code
if (check_fifo())
{
           // Set Setup Time window of 40 ns.
           Outp (67xxindex, 0x3d);
           Outp (67xxdata, 0x00);
```

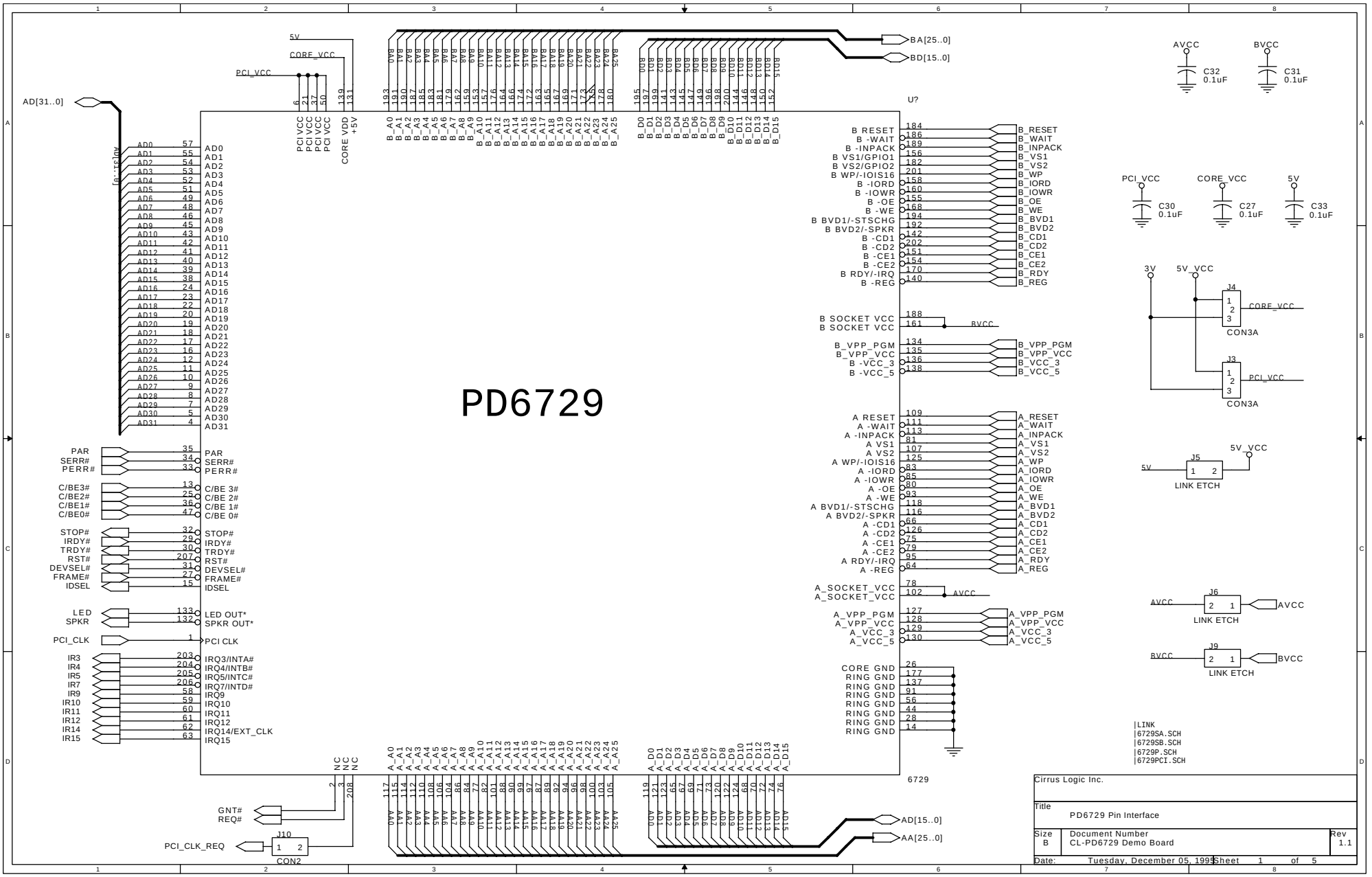
```
                // Set Command Time window of 200 ns
                Outp (67xxindex, 0x3e);
                Outp (67xxdata, 0x04);

                // Set Recovery Time window of 80 ms
                Outp (67xxindex, 0x3f);
                Outp (67xxdata, 01);

                // Set up Memory Window 0 to use Timing 1 register
                outp (67xxindex, 0x13);
                x = inp (67xxdata);
                x |= 0x40;
                outp (67xxdata, x);
}
```

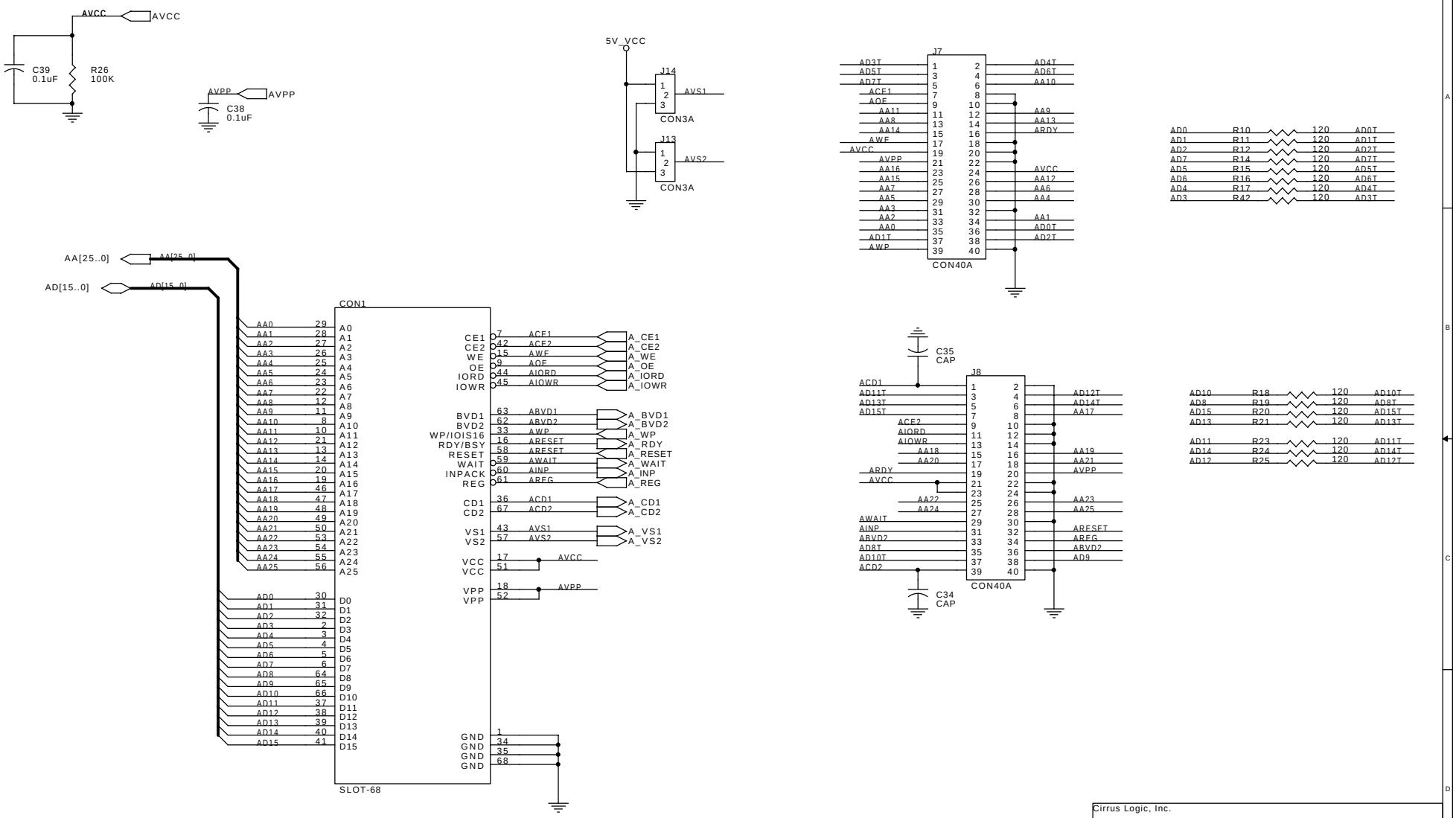
7.0 Appendix A

7.1 PD6729 PCI Bus Demonstration Board PC Card Solution — Schematics

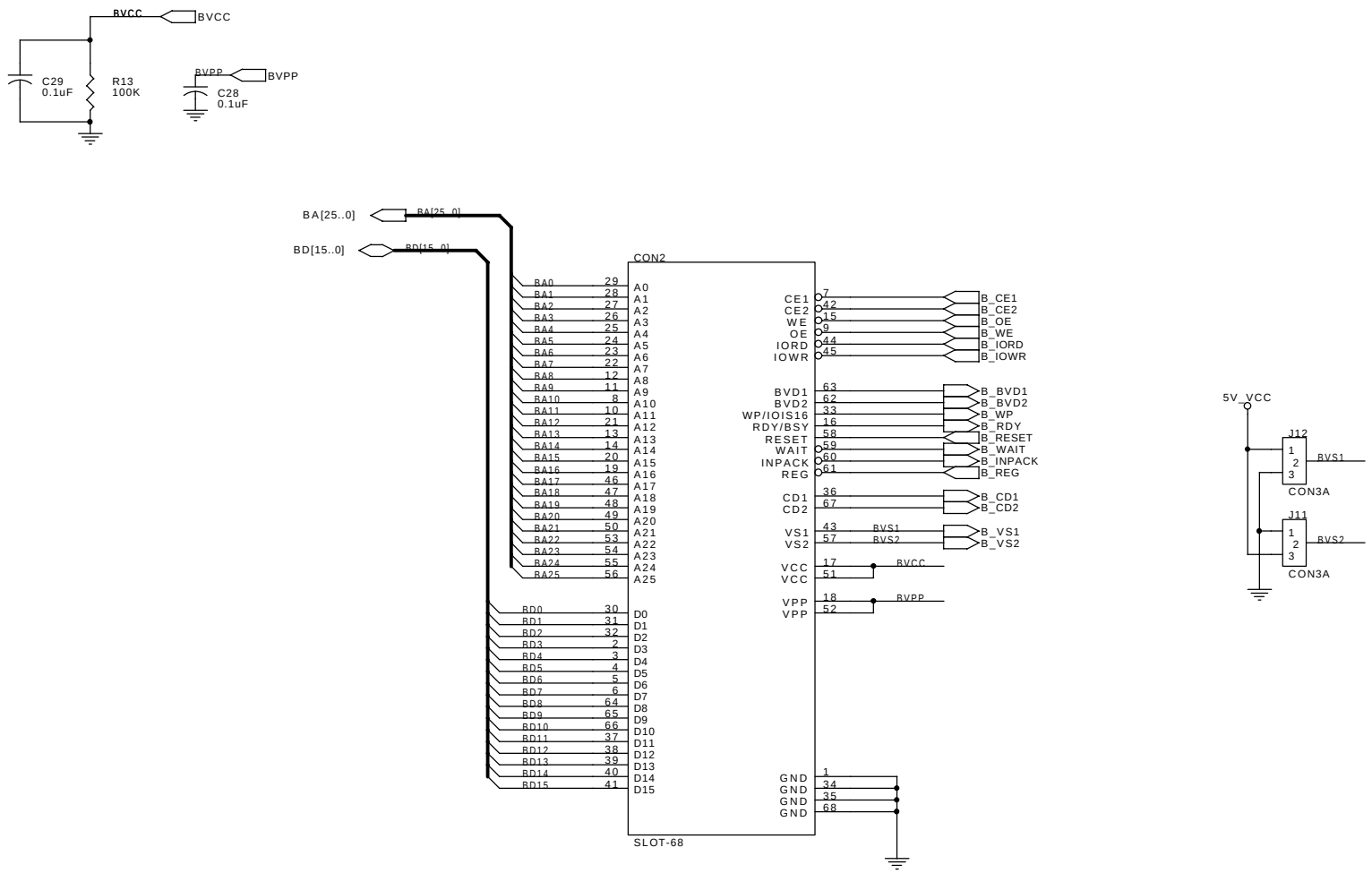
The schematics in this appendix show sample circuits. The signal connections used in evaluation board schematics can be applied to a motherboard solution. The evaluation board schematics provided include the following:

- Sample circuitry for the PD6729
  - — Power-control logic
  - — PCI bus interface
  - — PC Card bus interface

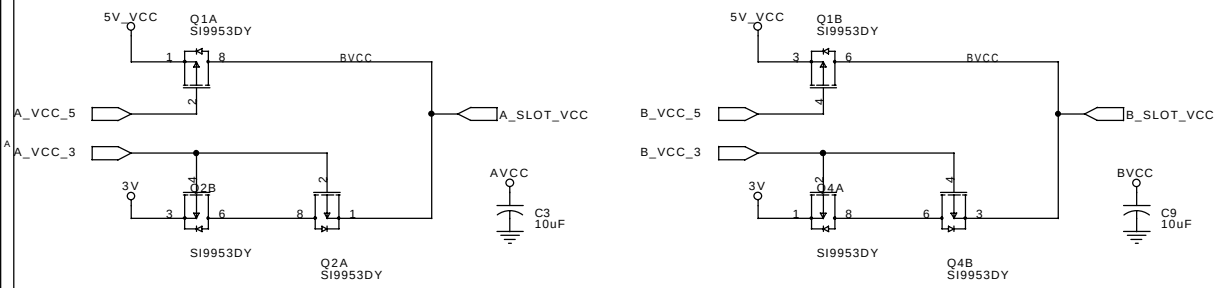

PD6729
U?
6729
5V
CORE_VCC
PCI_VCC
BA[25..0]
BD[15..0]
AD[31..0]
AD[15..0]
AA[25..0]
AVCC
BVCC
C32 0.1uF
C31 0.1uF
C30 0.1uF
C27 0.1uF
C33 0.1uF
3V
5V_VCC
J4
CON3A
J3
CON3A
J5
LINK ETCH
J6
LINK ETCH
J9
LINK ETCH
J10
CON2
PAR
SERR#
PERR#
C/BE3#
C/BE2#
C/BE1#
C/BE0#
STOP#
IRDY#
TRDY#
RST#
DEVSEL#
FRAME#
IDSEL
LED
SPKR
PCI_CLK
IR3
IR4
IR5
IR7
IR9
IR10
IR11
IR12
IR14
IR15
GNT#
REQ#
PCI_CLK_REQ
B RESET
B -WAIT
B -INPACK
B VS1/GPIO1
B VS2/GPIO2
B WP/-IOIS16
B -IORD
B -IOWR
B -OE
B -WE
B BVD1/-STSCHG
B BVD2/-SPKR
B -CD1
B -CD2
B -CE1
B -CE2
B RDY/-IRQ
B -REG
B SOCKET VCC
B SOCKET VCC
B_VPP_PGM
B_VPP_VCC
B -VCC_3
B -VCC_5
A RESET
A -WAIT
A -INPACK
A VS1
A VS2
A WP/-IOIS16
A -IORD
A -IOWR
A -OE
A -WE
A BVD1/-STSCHG
A BVD2/-SPKR
A -CD1
A -CD2
A -CE1
A -CE2
A RDY/-IRQ
A -REG
A_SOCKET_VCC
A_SOCKET_VCC
A_VPP_PGM
A_VPP_VCC
A_VCC_3
A_VCC_5
CORE GND
RING GND
|LINK
|6729SA.SCH
|6729SB.SCH
|6729P.SCH
|6729PCI.SCH
Cirrus Logic Inc.
Title
PD6729 Pin Interface
Size B
Document Number
CL-PD6729 Demo Board
Rev 1.1
Date: Tuesday, December 05, 1995
Sheet 1 of 5



Cirrus Logic, Inc.		
Title: PCMCIA Socket A Interface		
Size: B	Document Number: CL-PD6729 DEMO BOARD	Rev: 1.0
Date: Tuesday, December 05, 1995	Sheet 2 of 5	

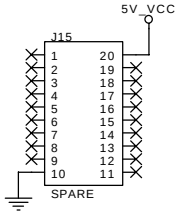


Cirrus Logic, Inc.		
Title: PCMCIA Socket B Interface		
Size: B	Document Number: CL-PD6729 DEMO BOARD	Rev: 1.0
Date: Tuesday, December 05, 1995	Sheet 3 of 5	

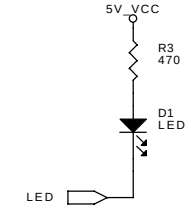


POWER CONTROL VCC

3.3 VOLT OR 5 VOLT SWITCH



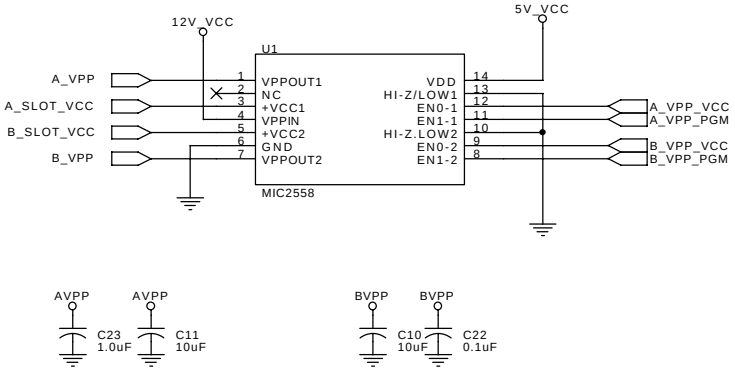
SPARE 20 PIN



(OPTIONAL)

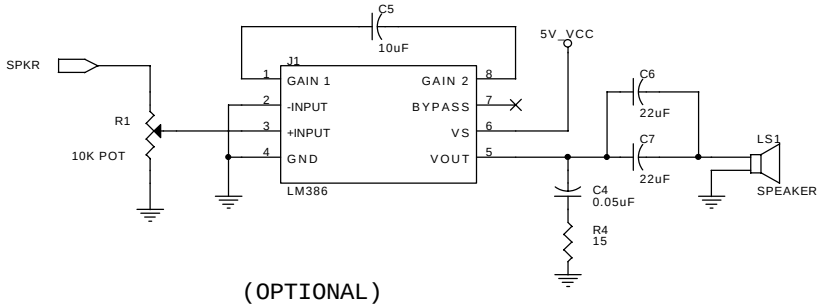
LED CIRCUIT

ATA MODE ONLY



POWER CONTROL VPP

5 VOLT OR 12 VOLT SWITCH



(OPTIONAL)

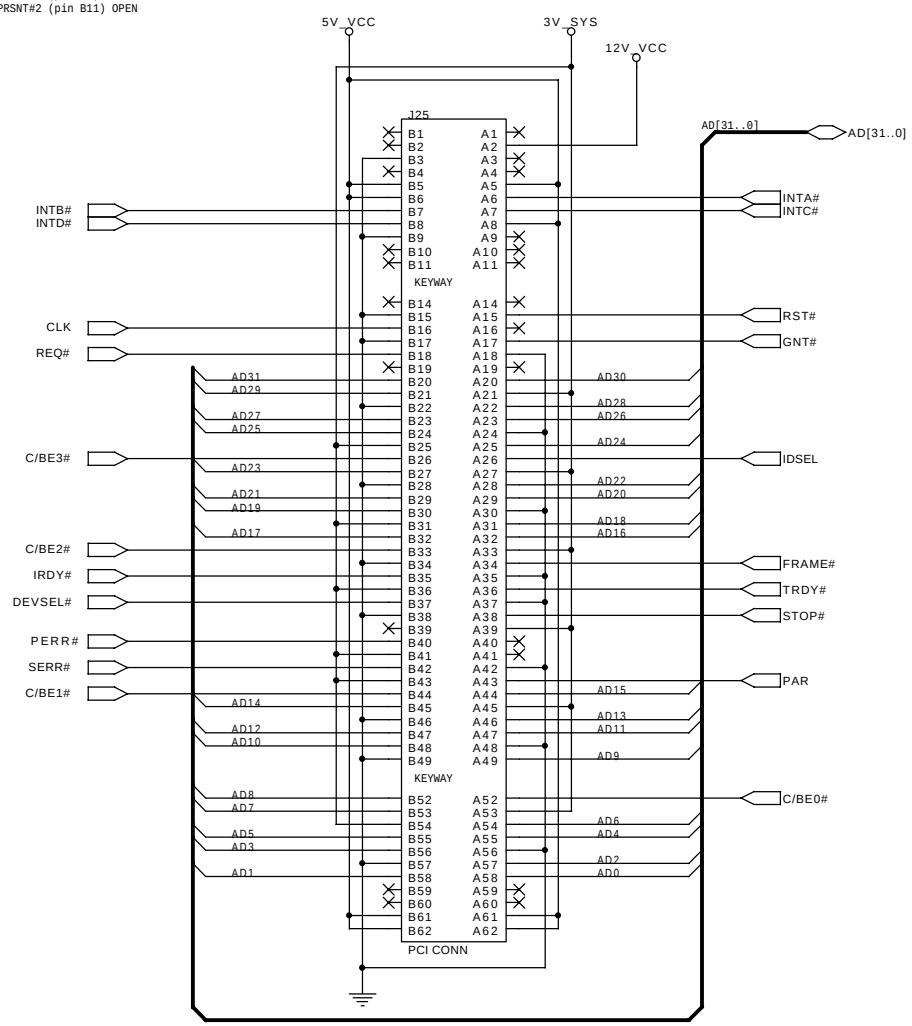
SPEAKER CIRCUIT

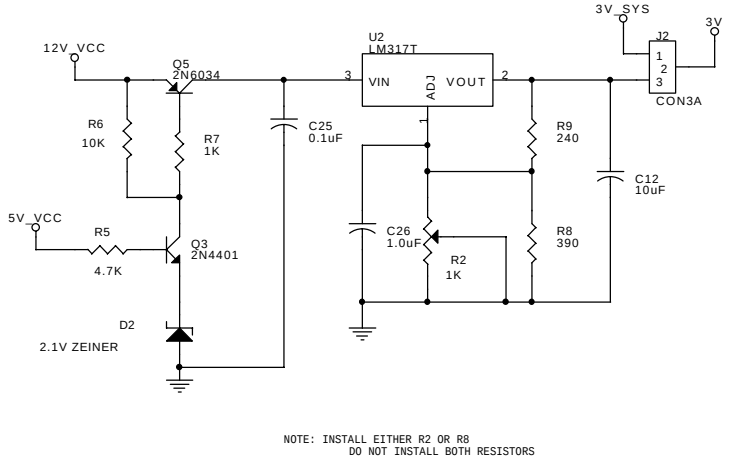
Note: The CL-PD6720-A-DM1-2 EVAL board uses the 12V supplied from the ISA bus as the 12V input for the VPP circuit. PCMCIA 2.01 specification requires a VPP voltage level of 12V +\- 5%. In actual applications we suggest that the designer use a 12V regulator circuit to guareent that the VPP voltage is within the PCMCIA 2.01 specification.

CIRRUS LOGIC, INC.		
Title: POWER CONTROL LOGIC		
Size: B	Document Number: CL-PD6729 DEMO BOARD	Rev: 1.1
Date: Tuesday, December 05, 1995	Sheet 4 of 5	

ADD IN CONFIGURATION 25 WATT MAX.

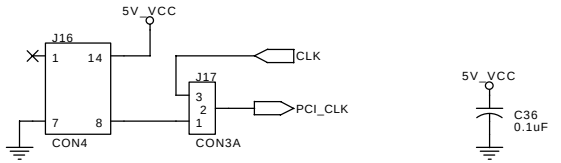
PRSNT#1 (pin B9) GND
PRSNT#2 (pin B11) OPEN



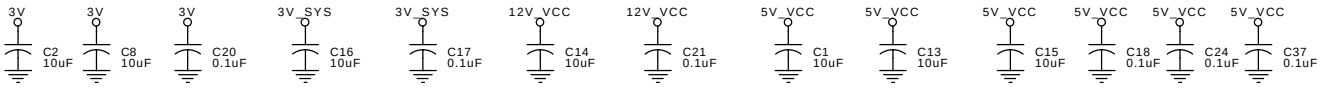


NOTE: INSTALL EITHER R2 OR R8
DO NOT INSTALL BOTH RESISTORS

3.3V Volt Regulator



EXTERNAL CLOCK SOURCE



CIRRUS LOGIC, INC.		
Title: ISA BUS INTERFACE		
Size: B	Document Number: CL-PD6729 DEMO BOARD	Rev: 1.0
Date: Tuesday, December 05, 1995	Sheet 5 of 5	