PD6710/6722 — PC Card (PCMCIA) Controller Solution for Embedded Systems

Application Note

May 2001

As of May 2001, this document replaces the Basis Communications Corp. document *AN-PD1*.

Application Note

Contents

Figures

Tables

1.0 Introduction

The PD6710 and PD6722 are single-chip PC Card (PCMCIA) controllers capable of controlling one or two PC Card or compact Flash sockets, respectively. They are 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 PD6710, a complete single-socket PC Card controller solution with power-control circuitry can occupy less than 1.5 square inches (10 square centimeters) of board space. Similarly, with the PD6722, a complete dual-socket PC Card controller solution with power-control circuitry can occupy less than 2 square inches (13 square centimeters) of board space.

The PD67XX controllers are 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. The PD67XX controllers also offer special power-saving features such as Automatic Low-power Dynamic Mode and Suspend Mode. Both controllers are true mixed-voltage devices 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 ISA (or ISA-like) Bus or to PCMCIA sockets.

Note: In this document, PD67XX represents both the PD6710 and PD6722.

Other features of the PD67XX controllers are discussed throughout this application note.

2.0 Overview

The PD6710 and PD6722 can interface to an ISA (or ISA-like) bus or expansion bus on the host side. The host CPU can be an Intel X86, Motorola, or RISC-based CPU. This application notes shows an example of an ISA or ISA-like bus interface with an Intel X86 processor.

This document provides information to help system designers create a successful PD67XX motherboard PCMCIA host controller solution in an Intel® 80X86-compatible environment. For information on how to design the PD67XX PCMCIA controllers in a non-Intel environment, customers should contact their application engineer.

This application note presents specific pin connection information for motherboard and ISA Bus solutions. The following PD67XX controller interfaces are discussed:

- ISA Bus interface
- PCMCIA socket interface
- PCMCIA socket power control signals
- Power distribution pins

Section 3.0, “Hardware Description” on page 7 describes the PD67XX ISA Bus and PCMCIA Socket interfaces in enough detail to permit a successful interface between an ISA Bus and a PCMCIA connector. The power-saving modes of the PD67XX are discussed in Section 4.0, “Power-Saving Modes” on page 13.

Section 5.0, “Windows” on page 16 presents PCMCIA address windowing basics and Section 6.0, “Timing Registers” on page 21 presents examples of how a card memory window is set up and mapped between the ISA Bus memory address space and the PCMCIA card memory address space.

In “Appendix A” are schematics that demonstrate a PD67XX ISA Bus implementation, including power-control circuitry. Note that 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 ISA Bus interface, socket interface, PCMCIA socket power control, and power distribution.

3.1 ISA Bus Interface

The PD67XX ISA Bus interface signals are internally buffered and can be directly connected to the ISA Bus. The PD67XX signals are named the same as (or similarly to) most 80X86 motherboard ISA Bus signal names so connections can be easily identified.

The following bus control/status signals can be directly connected to their corresponding ISA Bus signals: **IOR*, IOW*, MEMR*, MEMW*, REFRESH*, IOCS16*, MEMCS16*, IOCHRDY, SBHE*, and ZWS***.

There are two methods of interfacing address lines to an 80X86 motherboard ISA Bus. The type of processor in the system determines the method to be used. For processors with LA address lines (e.g., Intel 80286, 80386, etc.), directly connect the PD67XX Address Lines **LA[23:17]** and **SA[16:0]** to the ISA Bus Address Lines LA[23:17] and SA[16:0]. For motherboards using Intel processors *without* the LA address lines (e.g., Intel 8080), tie ALE of the PD67XX high, ground LA[23:20] of the PD67XX, and then connect LA[19:17] and SA[16:0] of the PD67XX to SA[19:0] of the ISA Bus.

For 16-bit processors such as the 80X86 family, the PD67XX Data Lines **SD[15:0]** can be directly connected to the ISA Bus Data Lines SD[15:0]. These lines are used to transfer data during memory and I/O cycles. For 8-bit processors, connect the PD67XX Data Lines SD[7:0] to the ISA Bus Data Lines SD[7:0], tie the PD67XX SBHE* Pin high, and leave the PD67XX Data Lines SD[15:8] disconnected.

The **CLK** Signal of the PD67XX may be directly connected to the ISA Bus OSC Signal or to another 14.318-MHz 50%-duty-cycle source. The PD67XX uses an internal synthesizer to generate its internal 25-MHz clock from this 14.318-MHz input. If this on-board, 25-MHz clock synthesizer is to be internally bypassed, a 25-MHz 50%-duty-cycle clock source should be provided to the PD67XX CLK Pin.

The **PWRGOOD** Signal is the PD67XX active-low reset input. It should be connected to the Power Good Signal from the system power supply or reset circuit. If the system does not provide a Power Good Signal, an inverted ISA Bus RESETDRV Signal may be used.

SPKR_OUT*/C_SEL is a dual-function pin. While PWRGOOD is low, it is a configuration option input used to determine which sockets the PD67XX will control. The rising edge of PWRGOOD is used to latch the level on this pin as the configuration option selection as follows: If there is only one PD67XX in the system, the SPKR_OUT*/C_SEL Signal (PD6710 Pin 141 or PD6722 Pin 200) can be left open because it is internally pulled-up while PWRGOOD is low. This will configure the PD6710 socket interface as PCMCIA Socket 0 and the PD6722 socket interface as PCMCIA Sockets 0 and 1. If SPKR_OUT*/C_SEL is low while PWRGOOD is low, this will configure the PD6710 socket interface as PCMCIA Socket 2 and the PD6722 socket interface as PCMCIA Sockets 2 and 3. For more information on socket address, refer to the *PD6710/’22 Datasheet*.

After PWRGOOD goes active-high, SPKR_OUT*/C_SEL becomes a TTL-level-compatible digital output from each socket's speaker input. This pin should be connected, along with the core-logic speaker signal, to the input of the system's speaker driver or amplifier. If the core-logic digital speaker output is active-low, the SPKR_OUT*/C_SEL Signal should be XNOR'ed with the core-logic speaker output. Firmware should configure the Misc Control 1 Register (Index 16h) Bit 4 (Speaker Enable) to a '1' to enable the SPKR_OUT*/C_SEL Pin to function as an output to drive a speaker. On a PD6722, the output of SPKR_OUT*/C_SEL is the logical XNOR of the speaker inputs from each socket interface.

The **-VPP_VALID** Pin is a readable status input to the PD67XX. Its active-low level indicates that the PCMCIA VPP power supply is operating at a stable voltage. Some PCMCIA power-supply chips include a '-VPP STABLE' status output, which can be directly connected to -VPP_VALID. If the PCMCIA power-supply chip does not include a -VPP STABLE status output, -VPP_VALID can be tied low. On the PD6710/PD6722 ISA Bus demonstration board, the -VPP_VALID Signal is 'jumperable' to a logical '0' since the ISA demonstration board always has a stable PCMCIA VPP voltage available in the form of the card operating voltage or the +12 volts from the ISA Bus.

The **IRQ[15,14,12,11,10,9,7,5,4,3]** Signals are used to indicate an interrupt request from a card. These signals should be connected directly to the corresponding ISA Bus interrupt signals. The IRQ12 and IRQ15 Signals have a dual-function capability. They can operate as a normal interrupts by connecting directly to the ISA Bus interrupt signals or they can perform alternate functions, as described below.

If the Misc Control 2 Register (Index 1Eh) Bit 4 (Drive LED Enable) is set, the **IRQ12** Pin functions instead as an open-drain driver for disk-status LED signals from the PCMCIA sockets. The schematics in Appendix A show one possible solution using the IRQ12 Pin as a disk-status LED driver.

If the Misc Control 2 Register (Index 1Eh) Bit 7 (IRQ15 Is RI Out) is set, the **IRQ15** Pin functions instead as a debounced -RI ring indicate output from each socket's BVD1/-STSCG Pin. This allows ring-indicate status from some PCMCIA modem cards to be used to signal systems to go from standby to full operation. To enable a socket's -RI Signal Out to IRQ15, the corresponding socket's Interrupt and General Register Control Register (Index 3h) Bit 7 (Ring Indicate Enable) must be set to '1'. In a '386SL-based system, IRQ15 can then be connected to the '360SL -RI input. In the PD6722, if either cards' -RI input is active, IRQ15 will go low in this mode. Note that when IRQ15 is being used as a -RI output, debouncing circuitry prevents IRQ15 from going high again until a new ISA Bus cycle occurs.

How the **-INTR** output should be connected is usually a function of which Card-services/Socket-services software is to be integrated with the system. With most Card-services/Socket-services, the -INTR Pin of the PD67XX will not be used and can be left disconnected. Typically, a Card-service device driver will designate a particular IRQ Interrupt to indicate a card status change.

If the designer wishes to use the -INTR Pin for status change interrupts, it should be connected to the '360SL $\overline{\text{SMI}}$ input in a '386SL-based system (or to the $\overline{\text{NMI}}$ input in a core-logic-based system) to allow the PD67XX to generate system interrupts. If the designer chooses to wire-OR the -INTR Pin to the $\overline{\text{SMI}}$ or $\overline{\text{NMI}}$ input of the processor, along with similar signals from other devices in the system, an interrupt handler needs to be employed so the processor can service the interrupt as a PD67XX status change. PCMCIA Card-services/Socket-services software vendors can typically provide sample code to be integrated into system interrupt handlers for this purpose.

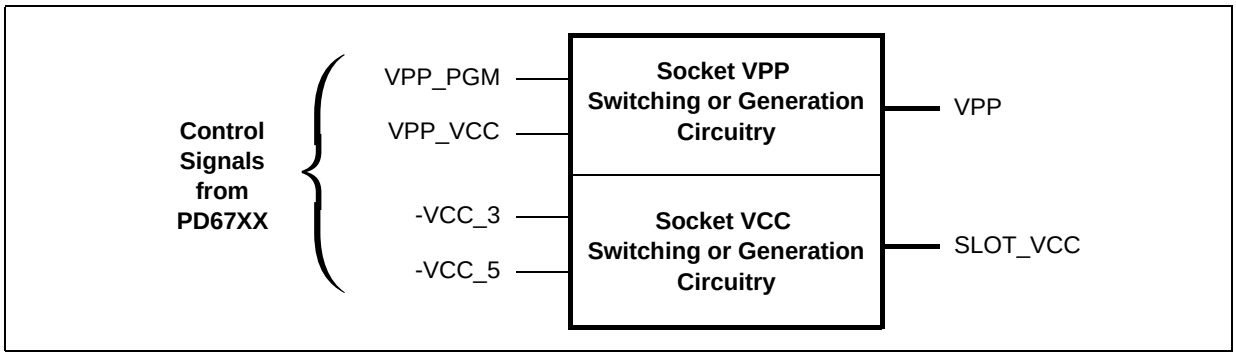
3.2 PCMCIA Socket Interface

The PD67XX host controller has a fully buffered PCMCIA socket interface. There is no need to use any additional socket buffering. All the socket interface signals can be directly connected to the socket connector.

3.3 PCMCIA Socket Power Control

The PD67XX power-control signals are used to the control PCMCIA socket power circuits. The signals are intended to be used to enable the power from the PCMCIA socket power source to 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. Refer to Figure 1 for a block diagram of the VCC and VPP Signals.

Figure 1. Block Diagram of Card-Socket Power Supply/Switching Circuit



The PD67XX **-VCC_3** and **-VCC_5** Signals for a socket interface are intended to be used to enable the power-switching circuitry that selects the power source applied to VCC Pins 17 and 51 of each PCMCIA socket. Both -VCC_3 and -VCC_5 are active-low signals and are mutually exclusive of each other. The Misc Control 1 Register (Index 16h) and the Power Control Register (Index 2h) are used to control -VCC_3 and -VCC_5. (See the *PD6710/'22 Datasheet* for more information about the registers and their function.) A low on -VCC_3 is intended to cause a PCMCIA socket power-supply circuit to provide +3.3 volts to the Card-socket VCC Pins 17 and 51; a low on -VCC_5 is intended to enable +5 volts to be provided to Card-socket VCC Pins 17 and 51.

When both -VCC_3 and -VCC_5 Pins for a socket interface are inactive-high, it is intended that a PCMCIA power supply would disconnect power sources from the socket's VCC Pins. Unlike other PCMCIA host controllers, there is no need to clamp a socket's VCC Pins to ground when both -VCC_3 and -VCC_5 are inactive-high; special interface circuitry in the PD67XX prevents card damage in this situation.

PD67XX socket-interface **VPP_VCC** and **VPP_PGM** Signals are used to control the voltage level to VPP Pins at the associated PCMCIA socket. To be PCMCIA compatible and function with both memory and I/O cards, the VPP switching/generation circuitry should be able to supply 0 volts, the selected card VCC voltage of +3.3 or +5 volts, and +12 volts to the socket's VPP pins. Both the VPP_VCC and VPP_PGM Signals are active-high signals that are mutually exclusive of each other. A high on VPP_VCC should be used to enable the PCMCIA socket VPP switching/

generation circuit to provide the socket's current VCC 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 socket's VPP-supply pins be at a logic '0' when programming voltages are not being applied. This can be accomplished by selecting a VPP switching or generation circuit that forces its VPP output to 0 volts when the socket interface's VPP_VCC and VPP_PGM Signals are both inactive-low. Alternatively, a bleed resistor can be placed from the socket's VPP 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 VCC and VPP to the PCMCIA socket in a system where +12 volts is already available.

The PD67XX host controller provides an input called **+5V_DET**. This signal can be used by applications to determine if an inserted card functions at +5 volts only or is capable of operating at +3.3 volts as well. The +5V_DET Pin should be connected to Pin 43 of the PCMCIA socket. The PD67XX internally pulls the +5V_DET Signal high. Cards that can support +3.3-volt operation will ground their Pin 43, and cards that work at +5 volts only, or that can work at +5 volts but choose to report their low-voltage capability through their card information structures, will leave their Pin 43 floating, causing +5V_DET to be active high.

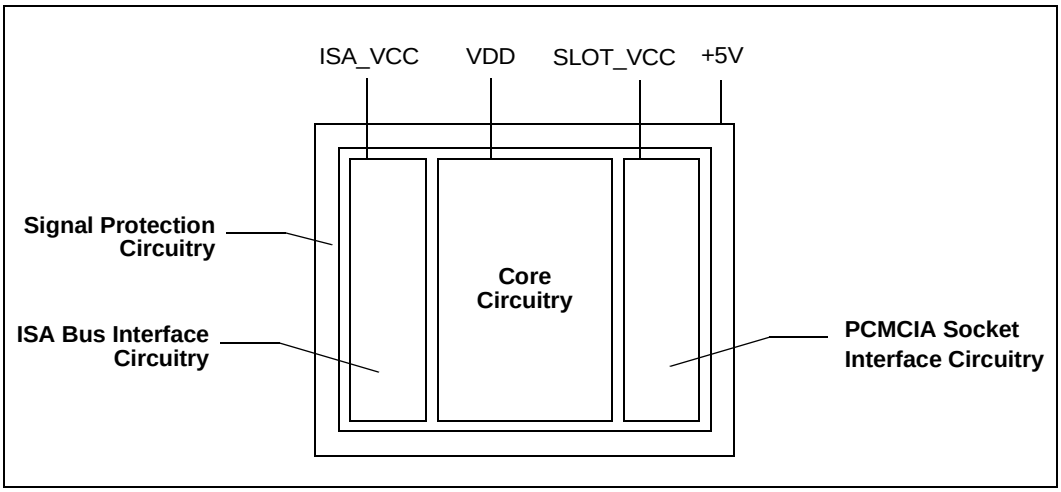
Note: Use of Pin 43 as +5V_DET is not yet officially approved as a PCMCIA Standard. Actual pin number assignment is subject to change pending approval by the 3.3/5V PCMCIA Working Group Committee.

3.4 Power Distribution

The PD67XX has four types of power-distribution pins that are used to provide power to various parts of the circuitry (Figure 2). These types are: ISA_VCC Power Pins, VDD Power Pins, SLOT_VCC Power Pins, and the +5V Power Pin.

Using these separate types of power pins permits the use of the PD67XX in a wide variety of applications where PCMCIA card voltages are dynamically changing between +3.3 and +5 volts or are being powered-on and -off.

Figure 2. PD67XX Powering of Internal Logic



The **ISA_VCC** Power Pins are used to power the ISA-bus-interface circuitry. The ISA_VCC Power Pins should be connected to the same power source used to drive other ISA Bus signals. When ISA_VCC is at +3.3 volts, ISA Bus interface signals operate at +3.3 volts. When ISA_VCC is at +5 volts, ISA-bus-interface signals operate at +5 volts.

The **VDD** Power Pins supply power to the main core circuitry of the PD67XX. Since VDD may be set at +3.3 or +5 volts, independent of the operating voltage at the ISA-bus or Card-socket interfaces, it is preferred that VDD be set 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 may be used to step-down the VDD voltage for power savings.

SLOT_VCC Power Pins supply power to the socket-interface pins and should be powered from the same card-power-circuit output that goes to the corresponding socket VCC Pins. This allows the respective socket-interface signals to operate at the same voltage as the card operating voltage.

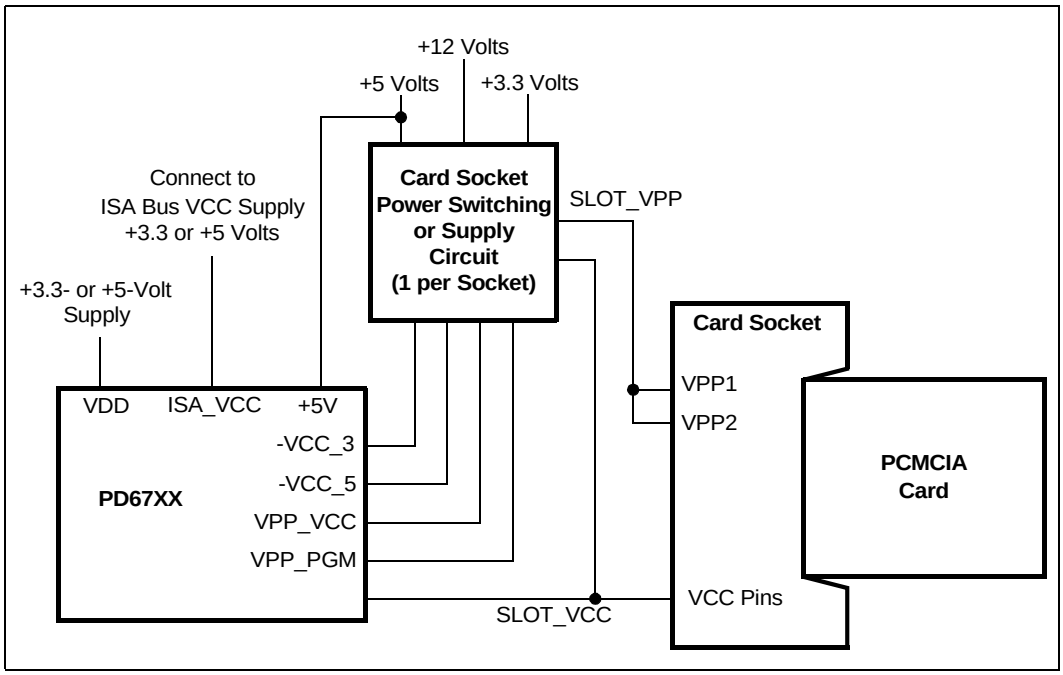
On the PD6722, which has two separate socket interfaces, each interface's SLOT_VCC Pins can operate independent of the other socket interface. This would prevent signal mismatch problems when, for example, a card in Slot A is operating at +5 volts while a card in Slot B is operating at +3.3 volts or is powered-off.

The power pin labeled '**+5V**' must be connected to the system +5-volt power supply. During conditions when the system +5-volt supply is not always powered-on, the +5V Pin must be connected to a supply whose voltage is at the same or higher potential than any of the voltages provided to the PD67XX other power pins.

Caution: If the other supply voltages (ISA_VCC, VDD, or SLOT_VCC) are raised more than 0.5 volts above the voltage applied to the +5V Pin, latch-up and/or damage to the PD67XX can result. Connecting the PD67XX power pins as previously described should prevent any potentially problematic situations. However, the possibility for latch-up conditions should be carefully analyzed in each design.

Refer to Figure 3 for a block diagram of the host and socket interfaces.

Figure 3. Typical Power Distribution for the PD67XX



4.0 Power-Saving Modes

In addition to the normal operating mode, which consumes very low power, the PD67XX supports three power-saving modes to further reduce the power consumption of the PCMCIA host controller. Refer to the *PD6710/'22 Datasheet* 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 by default enabled after chip reset. After reset, the host is configured for Automatic Low-power Dynamic Mode. This mode can be turned-off by writing to Misc Control 2 Register (Index 1Eh) and setting Bit 1 to a '0'. During periods of system inactivity in Automatic Low-power Dynamic Mode, the internal clock is turned off to most of the chip and the PCMCIA address lines are held at static values. Socket VCC and VPP Power Control Signals are not affected. Once activity occurs at the card socket (e.g., -INTR activation or status pins change) or ISA Bus cycles occur to either valid card addresses or PD67XX registers, the PD67XX restarts internal clocking. When ISA Bus cycles cease to access valid card address windows or the chip's internal registers, the PD67XX automatically stops internal clocks again and holds socket interface outputs static.

An example of a code segment that may be used to enable Automatic Low-power Dynamic Mode is shown below:

```
#define    67xxbase     0x3e0
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e

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

Note that when operating in Automatic Low-power Dynamic Mode, the system bus IOR* Signal must go inactive within 120 ns after IOCHRDY goes from low to high. This time is represented as time t1 in Figure 4. If the system core logic fails to meet this timing requirement, and Automatic Low-power Dynamic Mode is to be used, then the PD67XX Recovery Timing Registers must be programmed to adjust for the longest time that the system requires for IOR* to become inactive after IORCHDY becomes inactive.

If the maximum t1 time, $t1_{max}$, is greater than 120 ns, then a minimum value *n* is to be programmed into the Recovery Timing Registers. The value is calculated as follows:

$$n = \left(\frac{t1_{max}}{\text{Prescaler value}}\right) - 1 \qquad \text{(round-up } n \text{ to nearest intenger)}$$

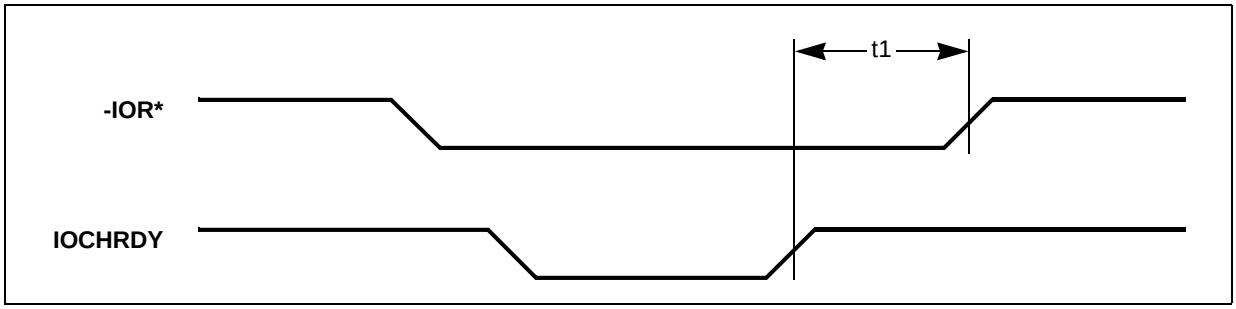
Note: See Chapter 6.0, "Timing Registers" for a description of the prescaler value. It is typically 40 ns, indicated by Bits 7 and 6 of the Recovery Timing Register being set to '0,0'. If Bits 7 and 6 are not set to '0,0', refer to Table 2 to determine the prescaler value (to be used in this equation) that corresponds to the bit settings.

For example, if $t1_{max}$ in a system implementation is 280 ns, then the minimum value programmed into the Recovery Timing Registers would be:

$$n = \left(\frac{280\ \text{ns}}{40\ \text{ns}}\right) - 1 = 6$$

In this example, Recovery Timing Registers (Index 3Ch/7Ch and 3Fh/7Fh) should be programmed to a minimum value of 06h.

Figure 4. Relationship Between IOCHRDY High and IOR* High for Automatic Low-Power Dynamic Mode Operation



4.2 Suspend Mode

During periods of operation when no card or chip access is needed, Suspend Mode may be used for increased power savings. The PD67XX is put into Suspend Mode by writing to the Misc Control 2 Register (Index 1Eh) and setting Bit 2 to a ‘1’. In Suspend Mode, the internal synthesizer and all the internal clocks are turned off and only access to the Index Register and the Misc Control 2 Register is supported. Both VCC and VPP Power Control Pins to each socket’s power-switching/ supply chipset are left unchanged (it is up to the system power-management software to suspend power to the socket before Suspend Mode if such an action is appropriate). Card-socket status changes and card interrupts are still detected and system interrupts are still generated when in Suspend Mode.

To return from Suspend Mode, the Misc Control 2 Register Bit 2 is set to a ‘0’. If the internal synthesizer is being used, other registers should not be accessed for 50 ms after return from Suspend Mode to allow the PD67XX to restart the internal clock synthesizer.

An example of a code segment that may be used to enable Suspend Mode is shown below:

```
#define    67xxbase     0x3e0
#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 Super Suspend Mode

If the Misc Control 2 Register Bit 2 is set to a ‘1’ and then the AEN Signal is held high, the PD67XX enters a ‘Super Suspend’ Mode. Super Suspend Mode retains all features of Suspend Mode, except access to the Misc Control 2 Register is not allowed. A key feature of Super Suspend Mode is that all ISA Bus inputs are disabled, shielding the PD67XX from the effects of floating ISA Bus signals in power-down modes of some systems.

4.4 Additional Notes on PCMCIA Subsystem Power Reduction

The PD67XX strictly controls power consumption during power-saving modes. The following measures can be utilized at the system-level to further minimize power consumption of the entire PCMCIA subsystem (including the PCMCIA power-switching/supply circuit and card):

- Controlling power to the cards (Power Control Register Bits 5:0)
- Enabling and disabling card interfaces (Power Control Register Bit 7)
- Externally turning-off clock sources to the PD67XX

5.0 Windows

The PD67XX provides memory and I/O address-translation features that allow transformation of system addresses into address ranges suitable for use with PCMCIA 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 PCMCIA Card- and Socket-service software or card-vendor-provided setup utilities.

5.1 Memory Windows

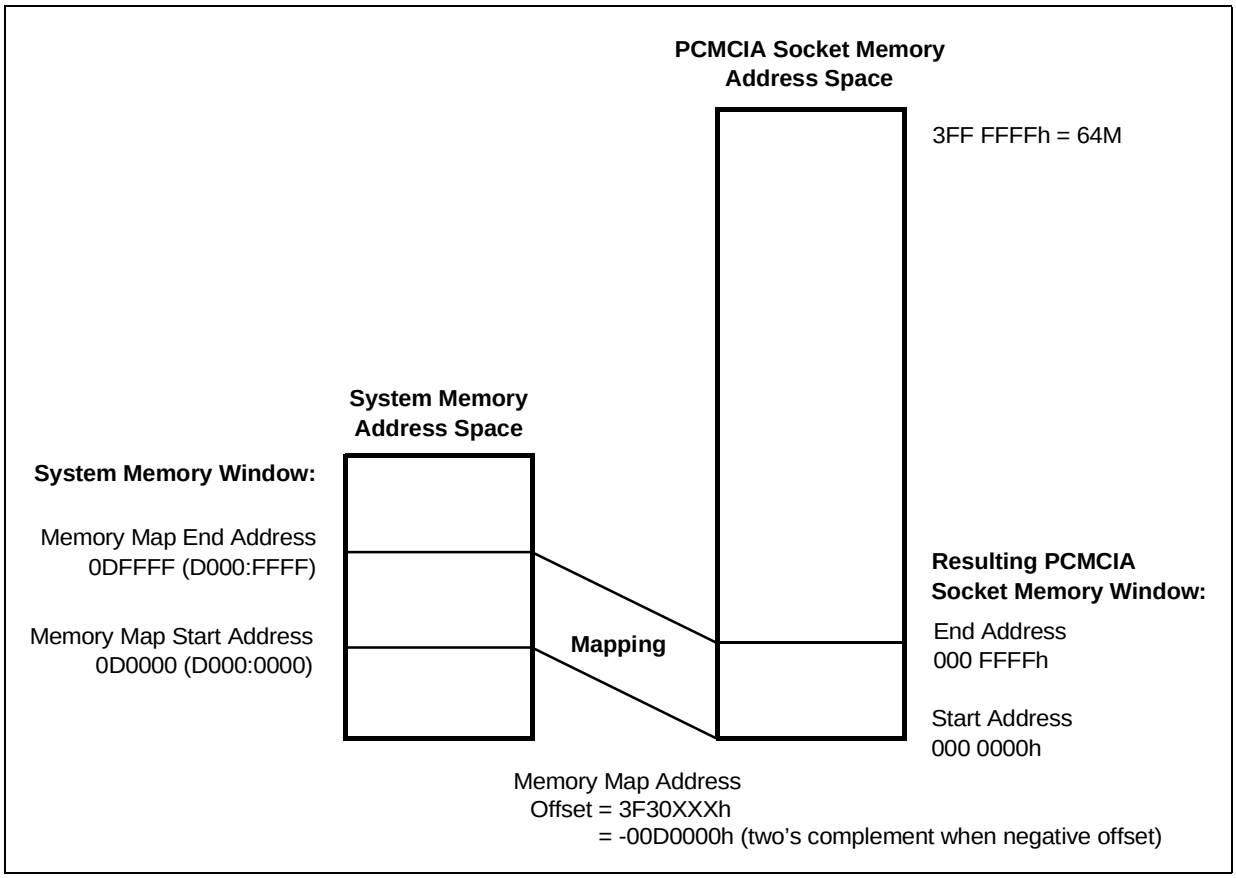
The PD67XX has five memory windows. These windows are accessed through the Socket-service software. A typical example of how a window is defined and accessed is described below.

Memory window sizes can be programmed on 4-Kbyte-increment boundaries. Offsets for these windows are provided so that the actual address used to access the PCMCIA peripheral can be different than the system ISA-bus address. For example, assume you want to access a memory card in your PCMCIA socket. The memory card has memory you wish to access at PCMCIA Addresses 0 through 0FFFF (64K bytes). A memory window could be specified in the computer's space that starts at 0D000:0 (0D000:0000) and continues through 0DFFFF (0D000:FFFF). This would allocate the memory space in the host (assuming it wasn't already being used by the computer) to the PC Card (PCMCIA) controller. To program the PCMCIA socket memory addresses to start at location 000 0000h for ISA-bus-memory addresses starting at 0D0000 (0D000:0000), an offset needs to be computed so that the ISA-bus-memory address plus the window's memory-map offset equals the PCMCIA address (000 0000h). This is done by taking the two's complement of 0D0000 (0D000:0000) to create the desired negative offset (remember that the 4-Kbyte-increment window resolution causes A11-A0 to map directly). This computed offset value is then programmed into the selected window's Memory Map Address Offset High and Memory Map Address Offset Low Registers. This then causes ISA-bus-memory accesses at Address 0D000:0 (0D000:0000) (for example) to occur at PCMCIA Address 0 and so on through the ISA-bus-memory map up to 0DFFFF (0D000:FFFF). Figure 5 shows how the mapping is done.

```
#define    67xxbase     0x3e0
#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 (67xxdata, 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);
```

Figure 5. Window Mapping Example of System Memory Address to PCMCIA Socket Memory Address



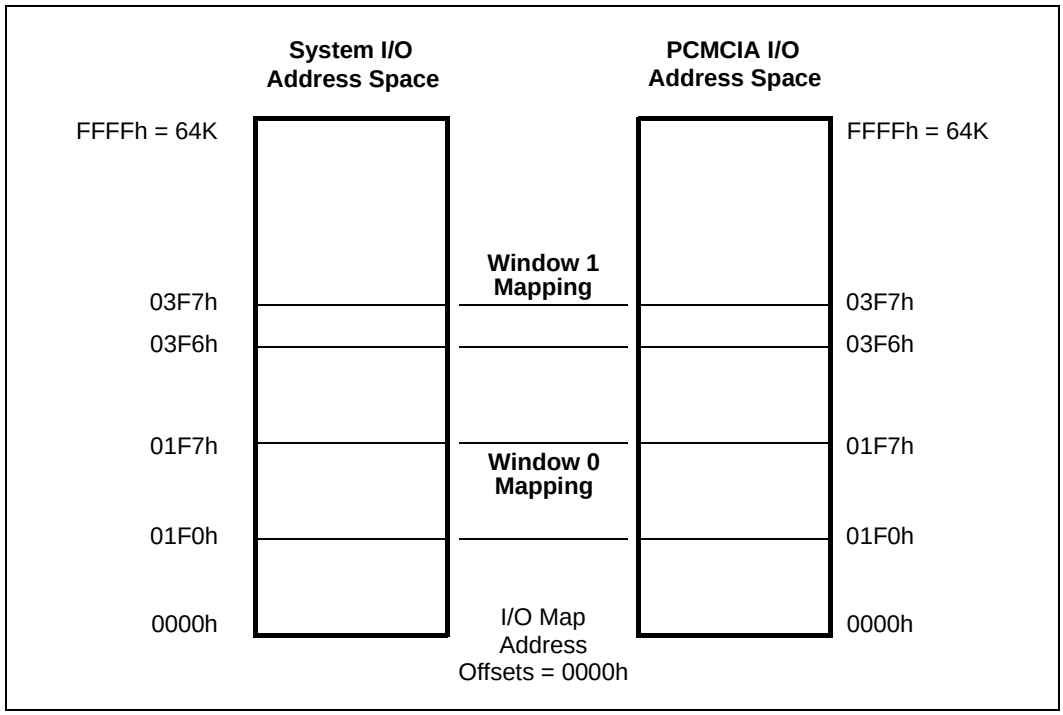
Below is a code segment (related to the example in Figure 5) that programs a PD62XX socket-interface Memory Window 0 as a 1-Mbyte window with a starting system address at location 0D0000 (0D000:D000) and a negative offset to create a 1-Mbyte socket memory window starting at PCMCIA address 000 0000h:

5.2 I/O Windows

The PD67XX supports two I/O windows per card socket. I/O-window mapping works similarly to that of memory windows. I/O window sizes can be specified in increments of even addresses on byte boundaries. Offsets are also provided from I/O windows so that the actual address used to access I/O on a PCMCIA card can be different than the system I/O address. A typical example of how an I/O window is defined and accessed is described below.

The example in Figure 6 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. A second I/O window has a start address at 03F6h and a stop address at 03F7h.

Figure 6. Window Mapping Example of System I/O Addresses Without Offset to PCMCIA I/O Address



Following is a code segment related to the example in Figure 6.

```
#define    67xxbase     0x3e0
#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 (67xxdata, 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 are used to program PCMCIA-card-access timing for the following timing parameters: Setup, Command, and Recovery. Programming the timing registers of the PD67XX for a specific timing value can relieve the CPU of the task of performing software timing loops to meet the required timing specifications for slower-access PC cards (e.g., Flash-memory cards). The PD67XX 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 can be individually programmed to use card access timing from either of the programmable timing register sets (Timing 0 or Timing 1).

Note: The timing-set register values should not be modified in the middle of PCMCIA Bus cycles. Timing registers should not be changed until the Write FIFO is empty because the timing changes begin immediately and may corrupt any in-progress cycles. This can be verified by checking that FIFO Control Register (Index 17h) Bit 7 (Empty Write FIFO) is set to a '1'.

After reset, the Timing 0 Register Set defaults to the values shown in Table 1 below. The Command Signals (i.e., -WE, -OE, -IOWR, and -IORD) are the signals used to control Setup, Command, and Recovery. The window widths are derived from the falling edge and rising edge of these signals. The register setting for the Timing Set 0 Register after reset is shown in Table 1 and the timing diagram is shown in Figure 7.

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

6.1 An Example: Setting Timing Register for Flash ROM

The user can change the values in the timing registers to vary the width of the Setup, Command, and Recovery Time for a particular application.

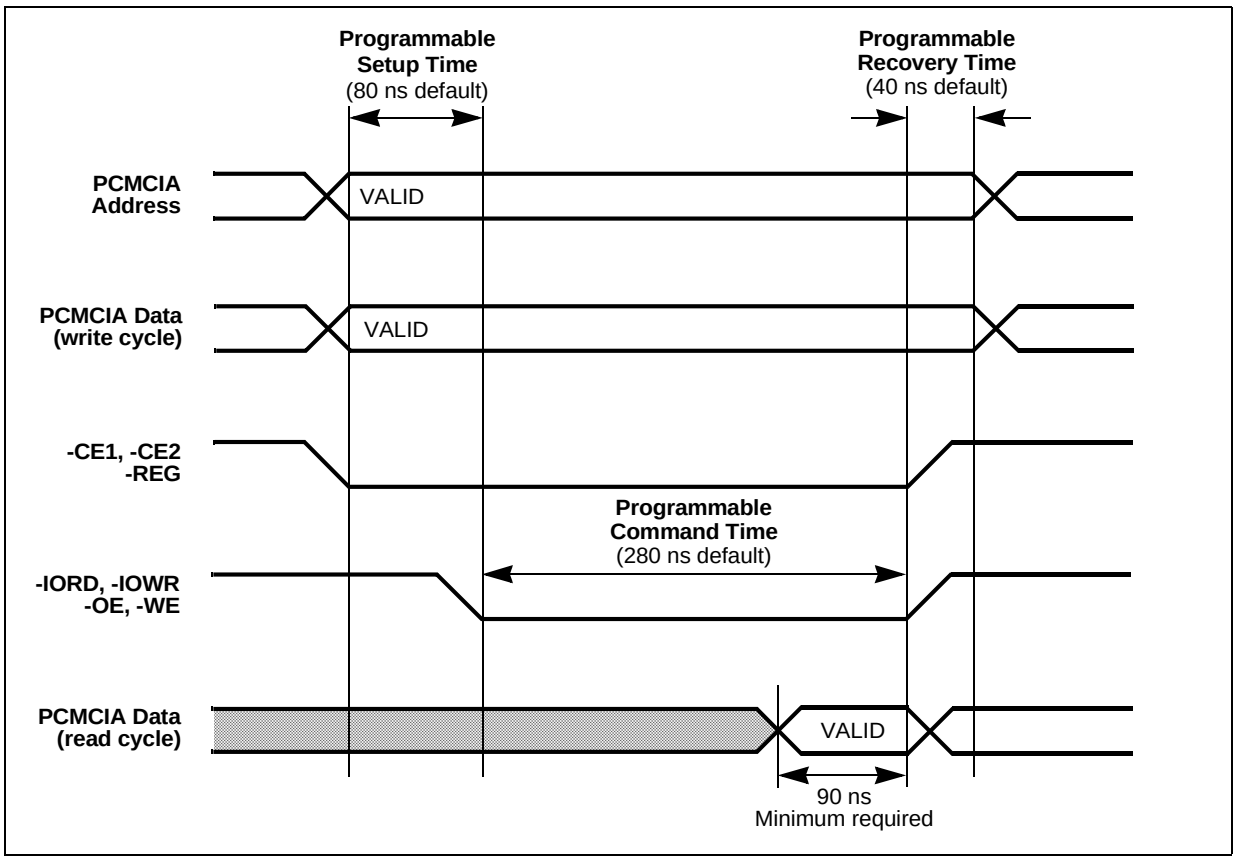
For example, consider programming a flash memory card that requires a minimum address Setup Time of 30 ns, a write pulse width of 150 ns, and a data Recovery Time of 50 ns. This particular flash-memory card has a write cycle time of 15 ms and will not accept another write until this unit of time has elapsed. The PD67XX can be configured as described in Section 6.2, "Programming the Timing 1 Register Set" on page 21 to accommodate this flash-timing specification. By programming the required cycle parameters into a timing-register set, flash-card programming can be accomplished without the system CPU having to calculate cycle periods.

6.2 Programming the Timing 1 Register Set

Each timing register has two variable fields that control the scale and duration of the timing parameter — the prescaler value and the multiplier value. Using the prescaler value and the multiplier value, the timing parameter is calculated as follows:

Timing parameter = (Prescaler value) × (Multiplier value) + 40 ns

Figure 7. Setup, Command, and Recovery Timing Diagram



Prescaler values are determined by Bits 7 and 6 of each of the three timing registers comprising Timing Set 0 and Timing Set 1. The possible values are as shown in Table 2

Table 2. Prescaler Values

Timing Parameter Register Bit 7, Bit 6	Prescaler Value
0,0	40 ns
0,1	640 ns
1,0	10.24 µs
1,1	327.68 µs

The multiplier is used to increase the timing parameter by a factor of the selected prescaler. The multiplier value is determined by Bits 5:0 of the three timing registers comprising Timing Set 0 and Timing Set 1. The six bits permit integer values from 0 to 63.

Based on the range of prescaler 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, which is large enough to accommodate the 15-ms write cycle time for flash ROM.

To select a Setup Time parameter close to 30 ns, choose '0,0' for Bits 7 and 6 (corresponding to a prescaler value of 40 ns), and with a multiplier of zero, this gives an address Setup Time of 40 ns:

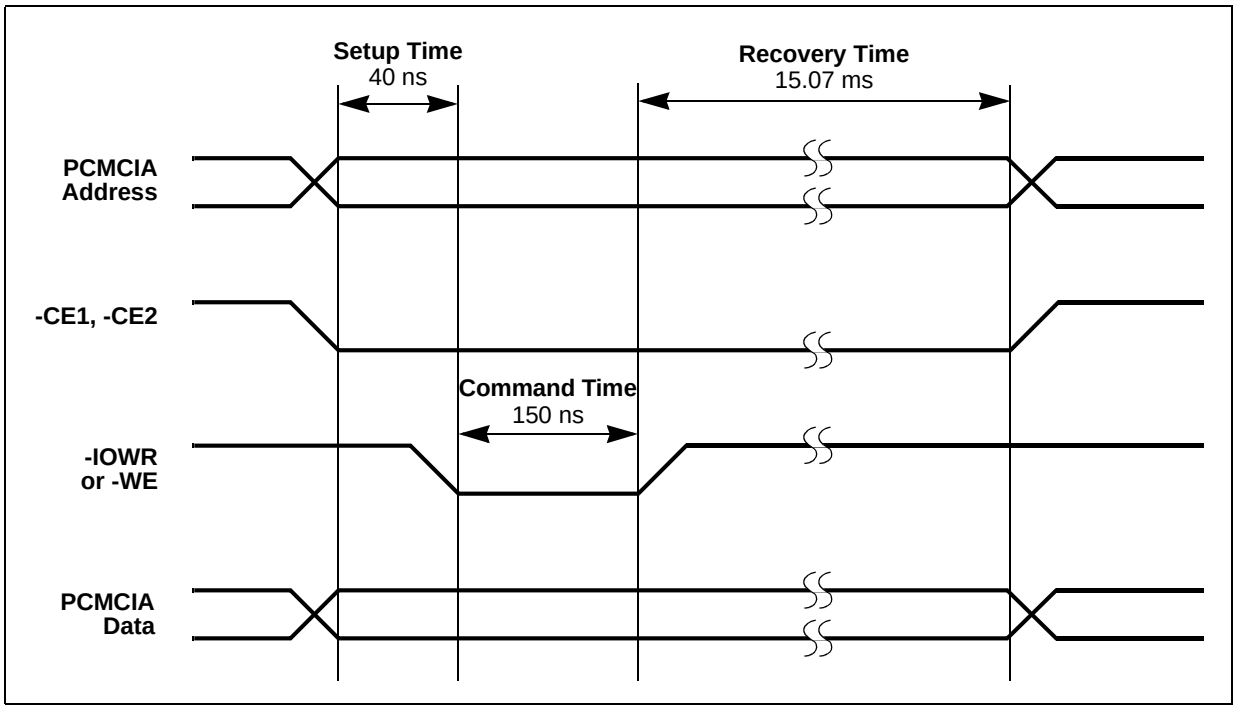
$$\text{Setup Time} = (40\ \text{ns}) \times (0) + 40\ \text{ns} = 40\ \text{ns}$$

The Command and Recovery Registers are configured similarly. After setting up the Timing 1 Registers for the example described, the Setup, Command, and Recovery Registers would have the values shown in Table 3 and Figure 8.

Table 3. Example of Flash ROM Write Timing Setting

Parameter	Register	Value	Time
Setup	3D	00	40 ns
Command	3E	04	160 ns
Recovery	3F	BE	15.07 ms

Figure 8. Timing Diagram of Example Flash ROM Write Timing Setting



The following code segment demonstrates the code needed to program the Timing 1 Registers for the Flash-ROM timing example:

```
#define    67xxbase     0x3e0
#define    67xxindex    67xxbase
#define    67xxdata     67xxindex + 1
#define    miscntrl2    0x1e
```

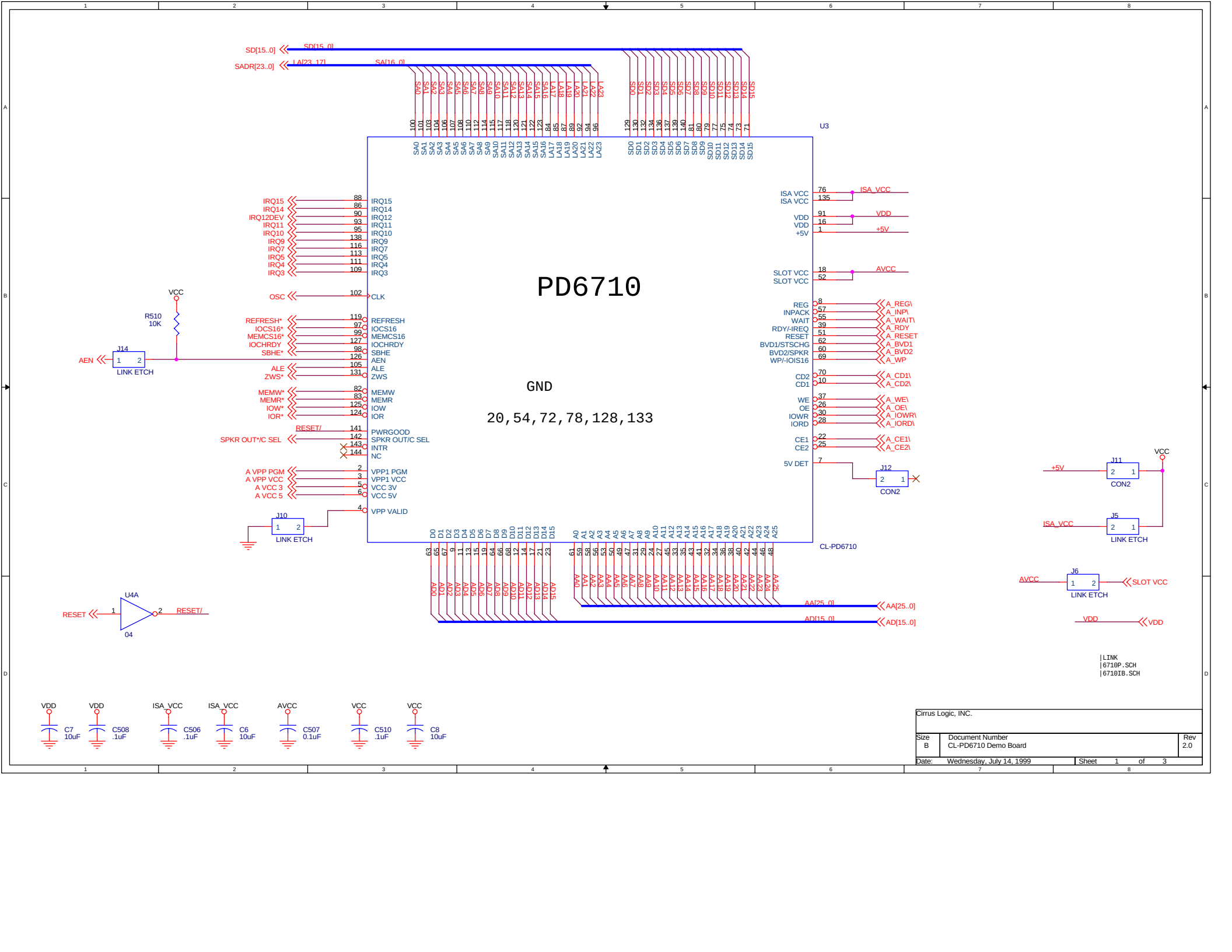
```
#define    TRUE         (1 ==1)
#define    FALSE        !TRUE

// 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 ( x == 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 150 ns
     Outp (67xxindex, 0x3e);
     Outp (67xxdata, 0x049nui);

     // Set Recovery Time window of 15.07 ms
     Outp (67xxindex, 0x3f);
     Outp (67xxdata, 0xbe);

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

}
```

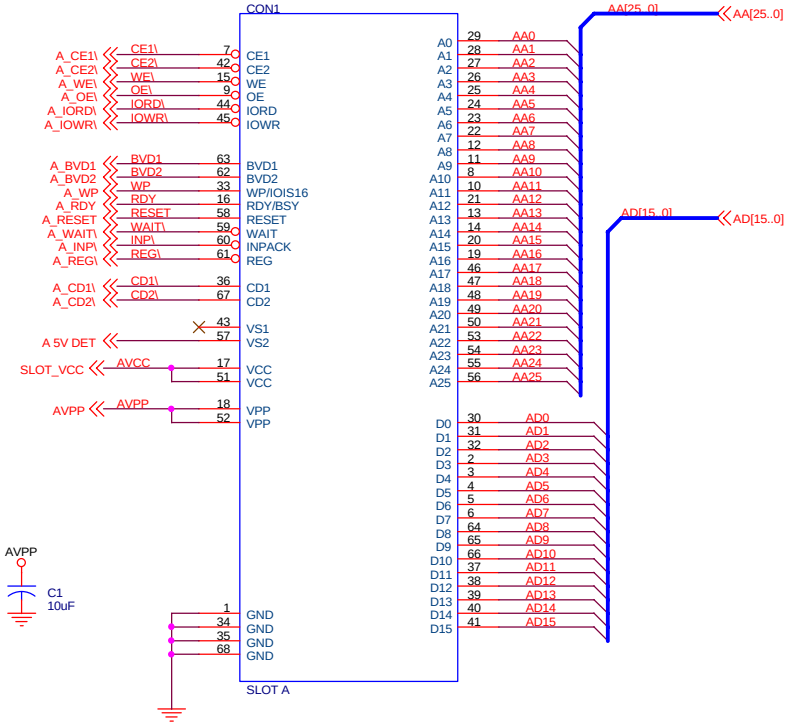
7.0 Appendix A

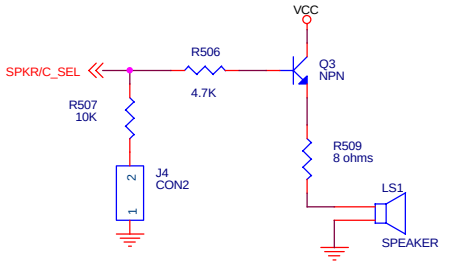
7.1 PD6710/'22 ISA Bus Demonstration Board PCMCIA Solution — Schematics

The schematics in Appendix A 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 PD6710:
  - Power-control logic
  - ISA Bus interface
- Sample circuitry for the PD6722:
  - Power-control logic
  - ISA Bus interface
  - PCMCIA Bus interface

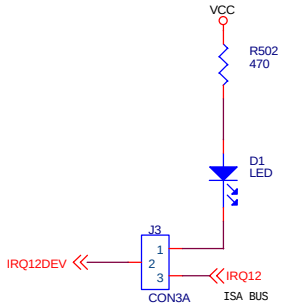

PD6710
GND
20,54,72,78,128,133
U3
CL-PD6710
SD[15..0]
SADR[23..0]
SD[15..0]
LA[23..17]
SA[16..0]
IRQ15
IRQ14
IRQ12DEV
IRQ11
IRQ10
IRQ9
IRQ7
IRQ5
IRQ4
IRQ3
OSC
CLK
REFRESH*
IOCS16*
MEMCS16*
IOCHRDY
SBHE*
ALE
ZWS*
MEMW*
MEMR*
IOW*
IOR*
RESET/
SPKR OUT*/C SEL
A VPP PGM
A VPP VCC
A VCC 3
A VCC 5
VCC
R510
10K
AEN
J14
LINK ETCH
J10
LINK ETCH
ISA VCC
VDD
+5V
SLOT VCC
REG
INPACK
WAIT
RDY/-IREQ
RESET
BVD1/STSCHG
BVD2/SPKR
WP/-IOIS16
CD2
CD1
WE
OE
IOWR
IORD
CE1
CE2
5V DET
ISA_VCC
VDD
AVCC
A_REG\
A_INP\
A_WAIT\
A_RDY
A_RESET
A_BVD1
A_BVD2
A_WP
A_CD1\
A_CD2\
A_WE\
A_OE\
A_IOWR\
A_IORD\
A_CE1\
A_CE2\
J12
CON2
AA[25..0]
AD[15..0]
U4A
RESET
RESET/
04
J11
CON2
J5
LINK ETCH
J6
LINK ETCH
SLOT VCC
LINK
6710P.SCH
6710IB.SCH
C7 10uF
C508 .1uF
C506 .1uF
C6 10uF
C507 0.1uF
C510 .1uF
C8 10uF
Cirrus Logic, INC.
Size B
Document Number CL-PD6710 Demo Board
Rev 2.0
Date: Wednesday, July 14, 1999
Sheet 1 of 3





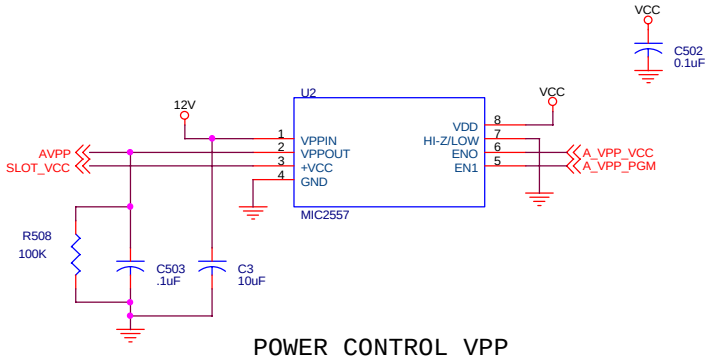
(OPTIONAL)

SPEAKER CIRCUIT



(OPTIONAL)

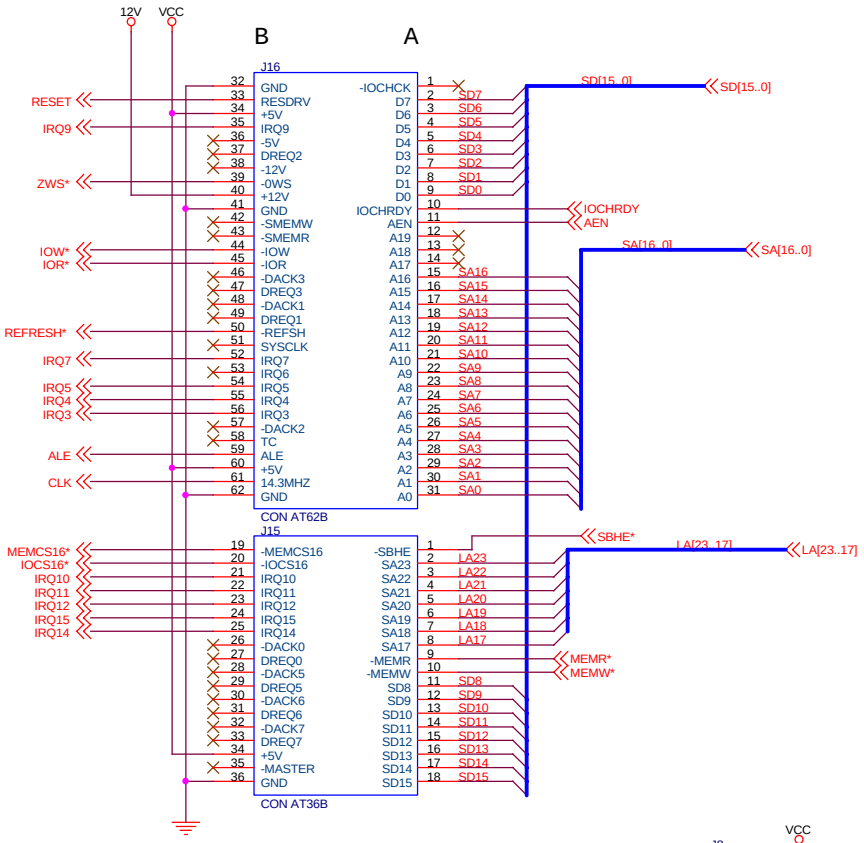
LED CIRCUIT
ATA MODE ONLY

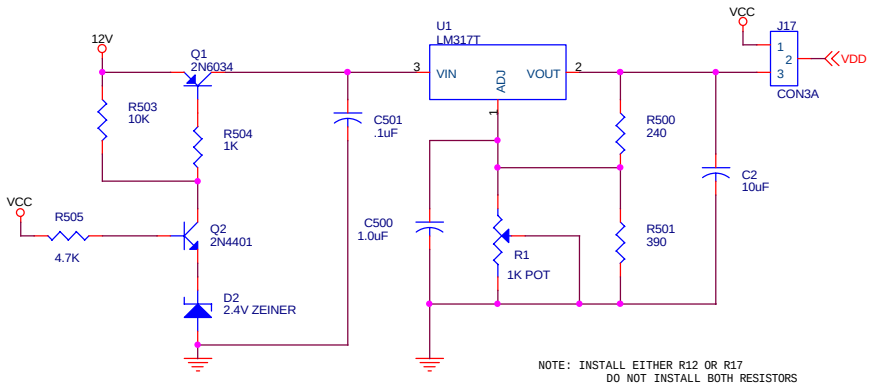


POWER CONTROL VPP

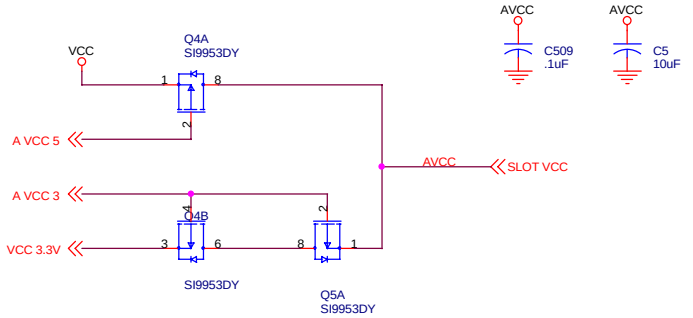
5 VOLT OR 12 VOLT SWITCH

CIRRUS LOGIC, INC.		
Title: POWER CONTROL LOGIC		
Size: B	Document Number: CL-PD6710 DEMO BOARD	Rev: 2.0
Date: Wednesday, July 14, 1999	Sheet 2 of 3	

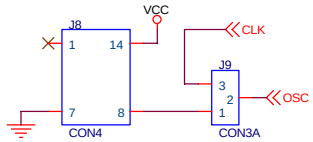




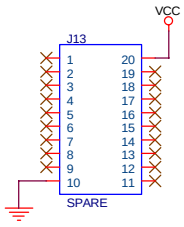
3.3V Volt Regulator



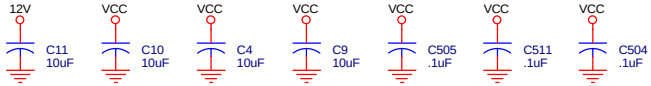
POWER CONTROL VCC



CLOCK SOURCE



SPARE



CIRRUS LOGIC, INC.		
Title: ISA BUS INTERFACE		
Size: B	Document Number: CL-PD6710 DEMO BOARD	Rev: 2.0
Date: Wednesday, July 14, 1999	Sheet 3 of 3	