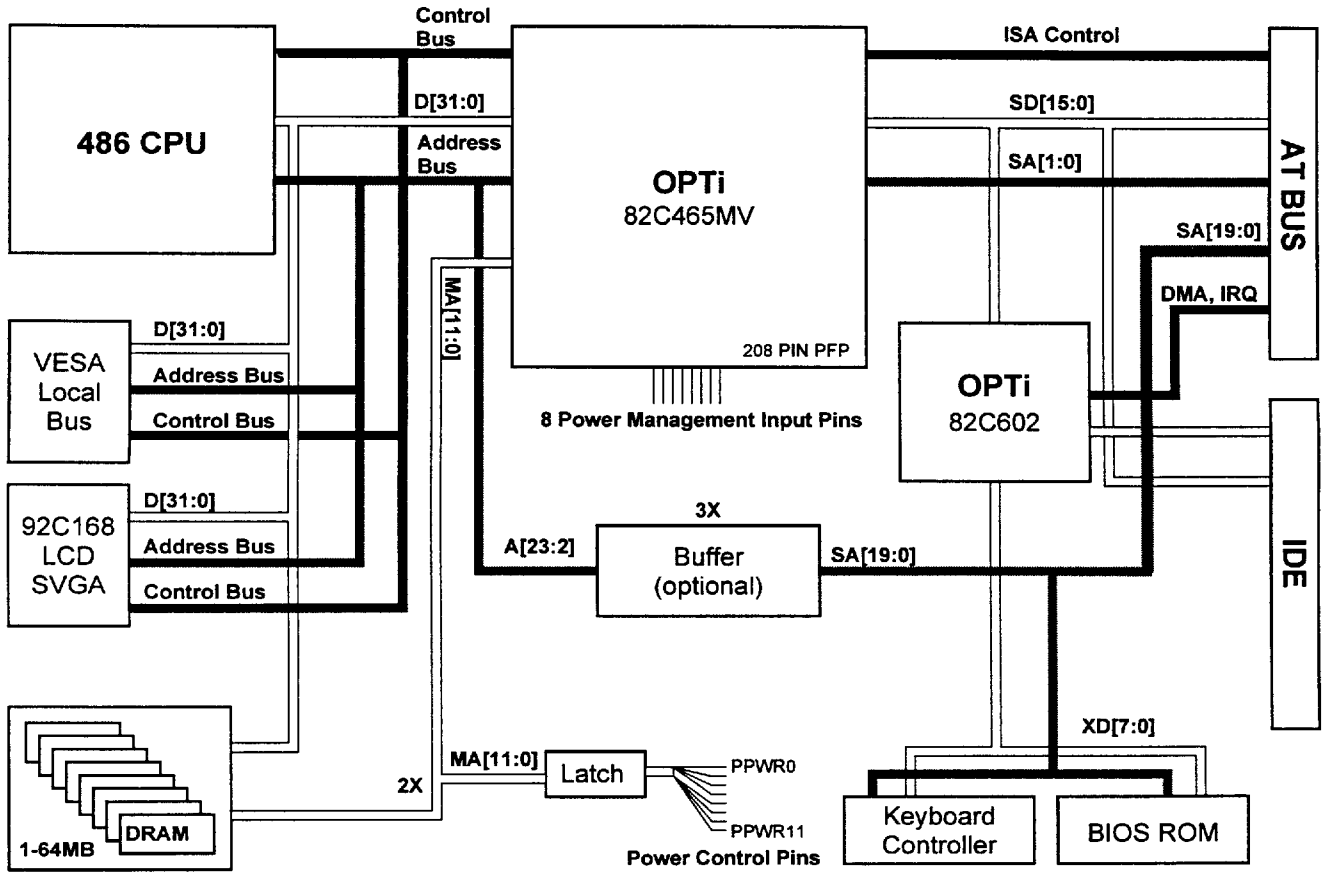
Single-Chip Mixed Voltage Notebook Solution

1.0 Overview

The OPTi 82C465MV chipset is a highly integrated ASIC that implements 32-bit AT-compatible core logic, along with power management and CPU thermal management hardware, in a single device. Its feature set provides an array of control and status monitoring options, all accessed through a simple and straightforward interface. All major BIOS vendors provide power management modules that are optimized for the OPTi power management unit and provide extensive software "hooks" that allow system designers to integrate their own special features with minimal effort.

The 82C465MV requires very little board space, implemented as a single 208-pin PQFP package in 0.8 micron CMOS technology. It can be used in conjunction with a 100-pin buffer chip to completely implement all the functions available on a typical desktop AT system.

Figure 1-1 System Block Diagram



1.1 Upgrade Comparison

The 82C465MV chipset has been replaced by the 82C465MVA and 82C465MVB derivatives. The following lists show the improvements mode in each chip.

NOTE This document covers the 82C465MV, 82C465MVA, and 82C465MVB. Features that apply only to the 82C465MVB are marked MVB, while features that apply to both the 82C465MVA and the 82C465MVB are marked MVA. Unmarked features apply to all three.

MVA

- No FLUSH required on entry into SMM
- Dual doze timers
- Local bus device can reset doze mode
- DMA state can be fully saved and restored when suspending
- DMA can trigger SMI
- HITM# and BOFF# can be connected directly (no external TTL)
- Fast RESET and A20M# generation can be disabled
- I/O trap address is saved
- Memory watchdog has additional control
- Floppy ports are shadowed
- AT bus address buffer can be disabled to save power
- The ATHOLD function is supported for docking station
- Software can generate hard reset
- High DRAM can be mode non-cacheable in L2 when using small tag RAM

MVB

- EDO DRAM is supported with no pin changes
- Includes Compact ISA support for the 82C852 PCMCIA controller
- AT bus refresh can be disabled to improve performance
- Four IDE drives are supported
- Cyrix M1sc CPUs are supported
- Type F DMA is supported
- Faster memory cycles are supported
- A20M# setting can be restored inside SMM
- Suspend refresh can be a narrower pulse
- Shadow RAM is now cacheable in L1

2.0 Features

2.1 CPU and VL-Bus Features

The 82C465MV offers the following CPU interface features.

- Supports AMD, Cyrix, Intel, and IBM 486-type CPUs, in 3.3V or 5V, including clock-tripled technology
- Provides a fail-safe thermal management scheme that predicts when CPU temperature is rising to unsafe levels and forces the system into a slower operating mode (Cool-down Clocking)
- Provides fast emulation of keyboard controller CPU reset and gate A20 control; supports port 092h as well
- Fully supports local bus implementations, including VL-bus masters
- Offers complete Microsoft APM operability, with CPU stop-clock support available.
- Supports next-generation processor stop clock protocol, to take advantage of the performance improvement possible when PLL start-up delay is reduced from milliseconds to nanoseconds
- Engages automatic internal resistors to eliminate the need for external pull up / pull down resistors on CPU address, data, and control lines; the resistors engage only when required, eliminating needless power consumption when the lines are driven
- Supports hybrid CPU interfaces to work with CPUs such as the IBM "Blue Lightning" processor that have mixed '386 and '486 aspects; supports all 32-bit data interfaces
- Provides SMBASE relocation to match the feature found in some CPUs that allows the SMBASE to be reprogrammed; each 64KB segment can be remapped to any 64KB-boundary segment in the first 640KB of address space
- Offers PCI compatibility in that the 82C465MV is fully compatible with the OPTi 82C82x family of PCI peripheral bridges, providing bus master support as well
- With the core operating at 5V, runs as fast as 50MHz with:
  - CPU and AT Bus I/O at 5V
  - CPU at 3.3V and AT Bus I/O at 5V
- With the core operating at 3.3V, runs as fast as 40MHz with:
  - CPU and AT Bus I/O at 3.3V.

2.2 DRAM/Cache Controller Features

The DRAM controller of the 82C465MV Single-Chip Notebook chipset runs from a 1X clock. It provides all of the performance features popular on powerful desktop systems and integrates power control for efficient operation.

- Provides L1 cache support for an on-CPU write-back cache such as that found on the Cyrix Cx486DX/DX2 processor and future L1 write-back CPUs from Intel and AMD.
- Provides power-managed L2 cache support with a high-performance, write-back external cache using the proven OPTi desktop 32-bit cache controller. Integral power control turns on the chip select lines only to cache chips actually being accessed, resulting in extremely low active-mode power consumption. Moreover, the cache can be flushed and completely turned off during low-power suspend mode.
- Operates up to five banks of DRAM, supporting any memory type in any bank, along with bank skipping (of intermediate banks) for automatic BIOS-based disabling of defective DRAM.
- Uses simplified memory programming scheme that supports up to 12x12 symmetrical along with asymmetrical DRAM such as 11x9 and 12x8 configurations. Symmetrical and asymmetrical types can be mixed.
- Allows any bank to use 256Kb, 512Kb, 1Mb, 2Mb, 4Mb, 8Mb, or 16Mb DRAM devices.
- Page-mode DRAM controller supports 3-2-2-2, 4-3-3-3, and 5-4-4-4 burst read memory cycles, zero- or one-wait-state DRAM write cycles.
- Supports two programmable non cacheable regions
- Offers fully programmable shadowing of ROM using DRAM in the C0000-FFFFFh region.
- Allows write-protected shadowing and caching of system and video BIOS.
- Allows normal or slow refresh, CAS before RAS refresh, and self-refresh DRAM support; minimum-pulse refresh cycles save power during suspend mode.

2.3 AT-Bus Features

The AT bus interface of the 82C465MV chip offers many improvements over previous generation OPTi notebook chipsets and competitive notebook chipsets.

- The internal 82C206 IPC offers a true single-chip notebook implementation and is based on the proven 82C463MV core.
- The 82C465MV implements a complete AT-compatible system with only one extra device, the OPTi 82C602 Notebook Companion chip, which contains a Real Time Clock (RTC) with 256 bytes of non volatile (backed up by battery) RAM and the equivalent of seven discrete TTL devices.
- The logic integrates an enhanced IDE interface running

at local-bus speeds (100% speed increase typical) based on the proven OPTi 82C611 core.

- Integral IDE support uses one external 74244 TTL device to control the IDE, with a second 7416245 device optional for complete power-down isolation of the IDE drive while the system is active.
- The IDE command scheme shares no AT-bus command lines, to prevent incompatibility with other AT-bus devices due to illegal "short pulse" cycles on AT bus.
- 8.00 MHz AT bus operation is available for implementations requiring exact adherence to the original AT standard.
- 16-bit decoding for internal I/O prevents conflicts for I/O peripherals addressed above 100h.
- Four programmable chip selects each decode ten address lines, A[9:0] for I/O addressing or A[23:14] for memory addressing. Memory address decoding allows simplified ROM chip select generation for applications such as Microsoft Windows in ROM.

2.4 Power Management Features

The synergistic incorporation of power management and system control features with the standard AT-subsystem controller of the 82C465MV chipset results in a compact design that handles multiple tasks with a simple, common interface.

The power management interrupt (PMI) scheme provides system management code with a quick means of identifying and handling events that affect power control and consumption.

- Recognizes 28 separate PMI events. Within these events, many sub-events are also identifiable for a high degree of power management monitoring flexibility.
- Eleven of the PMI events have individual timers to indicate inactivity timeout situations.
- Eight external inputs are available for monitoring asynchronous system events. These are in addition to the AT-compatible IRQ lines that can also be monitored as power management events.
- PMI generation on access allows SMI code to intercept status queries to powered-down devices that do not actually need to be restarted simply to return an "idle" status.
- Activity tracking register of eight events allows SMI or non SMI applications a means of determining whether activity has occurred since the last time the register was checked. Polling for I/O activity can then be used instead of multiple SMIs for less significant events.
- Memory watchdog monitoring allows accesses to memory ranges (specified as programmed) to cause an SMI. AT-bus memory devices that are not being accessed can be programmed to cause a timeout SMI so that unused peripherals can be powered down.
- Supports system-level low-power suspend, low-power suspend with zero-volt CPU suspend, or total system zero-volt suspend
- Twelve peripheral power control pins plus four user-definable I/O pins provides exceptional flexibility in peripheral device control.
- RTC alarm or modem ring can wake up the system from low-power suspend mode.
- Suspend current leakage control ensures that negligible power will be consumed in suspend mode without additional external buffering.

2.5 Backward Compatibility Features

The 82C465MV is application-compatible with the 82C463MV for the vast majority of applications.

- The register set and logic are derived from and are a superset of the popular OPTi 82C463MV notebook chipset.
- When used as a drop-in replacement for the 82C463MV, the 82C465MV allows continued operation with no required changes to the original 82C463MV BIOS. Optional programming needed to take advantage of performance improvements of 82C465MV logic can be run from an executable file.
- Many of the new 82C465MV features can be utilized with only minor changes to the 82C463MV BIOS; more extensive use of the new feature set requires some changes to hardware design as well.
- BIOS code need only check a single register to learn whether it is running on the 82C465MV or on an 82C463/463MV chipset.

NOTE The Sequencer of the 82C463MV has not been implemented in the 82C465MV. Contact OPTi for information regarding the conversion of Sequencer routines to SMI routines.

3.0 Signal Definitions

3.1 Pinout Options

The OPTi 82C465MV can alter its character significantly by simple strap options that are detected at hardware reset time. Figure 3-4, Figure 3-5, Figure 3-6 and Figure 3-7, shown later in this section, illustrate the pinouts of the 82C465MV in the following modes:

- 82C465MV standard configuration without L2 cache interface
- 82C463MV-compatible configuration
- 82C465MV enhanced configuration with L2 cache interface
- 386 interface configuration (Blue Lightning-compatible).

These modes are strap-selected interface options and are described below.

3.1.1 Strap-Selected Interface Options

The flexibility of the 82C465MV across a wide variety of applications is due in large part to its many interface strap options. The options listed in Table 3-1 are strap-selected at reset time as indicated. Normally, these pins are simply pulled up or down with a resistor to enable the desired function. The chip contains internal resistors that are enabled only at reset time to preset a default function, so that an external pull-up/pull-down may not be needed and can sometimes be avoided to save power. In most cases where an external resistor is required, the direction (pull-up or pull-down) coincides with the inactive state of the signal and therefore consumes negligible power. The internal resistors have a value of approximately 50KΩ.

Optionally, the system designer can strap these pins with tri-state drivers that enable their outputs only when the RST4# signal is active. The chip samples the lines on the rising edge of the RST1# input, at which time RST4# is still active.

Each strap option is described in detail in the section of this document that describes the affected subsystem.

Table 3-1 Strap Option Summary

Strap Option	Signal	Pin	Control Provided	Internal Up/ Dn at Reset	Default with No Strap
L2 Cache Support	SA0	146	Strap SA0 low w/ 4.7K to enable L2 cache interface	Up	No cache
Local-Bus IDE DBE Polarity	TRIS#	176	Strap DBE (TRIS) high for active low, low for active high	Up	TRIS# active low (for 82C463MV compatibility)
Input Clock 1X/ 2X Selection	ATCYC#	160	Strap ATCYC# high for 1X w/10K	Down	2X clock OSCCLK (for 82C463MV compatibility)
FBCLKOUT Delay	MA11/ DACKMUX2	77	Strap pin 77 high for delay w/10K	Down	No delay
CPUCLK Delay	RAS4# DACKMUX1	78	Strap pin 78 low for delay w/10K	Up	No delay
New Memory Control Interface	MDIR/ DACKMUX0	79	Strap pin 79 high for old 82C463MV-compatible scheme w/10K	Down	New interface
SL-Enhanced Enable	MP1:0	13, 23	Strap MP0 and MP1 high for STPCLK# operation w/10K	Down	No STPCLK# feature
CPU Clock 1X/ 2X Selection *	MP2	3	Strap MP2 high for 2X clock w/ 10K	Down	1X CPU clock
CPU Type 386/ 486 Selection	MP3	198	Strap MP3 high for 486 interface w/10K	Down	386-type interface

Table 3-1 Strap Option Summary (cont.)

Strap Option	Signal	Pin	Control Provided	Internal Up/ Dn at Reset	Default with No Strap
Resume Reset Selection *	SA1	148	Strap SA1 high for RSMRST# on EPMI2 w/10K, otherwise defaults to normal RST4# reset	Down	No RSMRST# on EPMI2 **
CA25/RDYI# Selection	SBHE#	161	Strap SBHE# high for pin 139 = CA25, otherwise pin 89 defaults to RDYI#	Down	Pin 139 is RDYI# **
CPU Interface Level *	NMI	120	Strap NMI low w/10K for 5V CPU; otherwise, 3.3V CPU assumed	Up	3.3V CPU interface
AT Bus Interface Level *	INTR	121	Strap INTR low w/10K for 3.3V AT bus; otherwise, 5V AT bus assumed	Up	5V AT bus interface

* Indicates that additional information is available on the following pages

** Indicates that heavy AT bus loading might require use of external 4.7K pull-down resistors. The internal pull-down resistor may not be sufficient to oppose external conductance to Vcc through devices on the AT bus that have this line pulled up.

3.1.1.1 Mixed Voltage Interface Options

Pins 120 and 121 provide strap options to select internal level translation of interface signals before the core logic interface. Pin 120 selects the CPU interface level, and pin 121 selects the AT bus interface level. With no straps, the option defaults to a 3.3V CPU and a 5V AT bus. The proper selections depend on the voltage applied to each power plane; the Vcc pins for each power plane are listed in the Power and Ground Pins table of the Pin Descriptions section that follows.

Table 3-2 shows the proper strap settings for each voltage mix.

Table 3-2 Strap Settings for Interface Voltages

CPU Interface Voltage	AT Bus Interface Voltage	Core Operating Voltage	Pin 120 (NMI)	Pin 121 (INTR)
3.3V	3.3V	3.3V or 5V	No strap	Strap
3.3V	5V	3.3V or 5V	No strap	No strap
5V	5V	5V	Strap	No Strap

Notes:
1) Pins 120 and 121 are never both strapped at the same time.
2) The combination of CPU interface at 5V and AT bus interface at 3.3V is not allowed, regardless of core voltage.
3) If the core is operating at 3.3V, both the CPU and AT-bus interfaces must be at 3.3V as well.

3.1.1.2 Resume Reset (RSMRST#) Function

Pin 148 strapping works in conjunction with bit 40h[0] to determine whether pin 185 acts as a reset line that toggles upon resuming from suspend mode. Refer to Reset Logic in the "System Functions" section for complete information on this option.

3.1.1.3 Reading the 1X/2X Strap Setting

The state of the 1X/2X CPU strapping selection can be read back through bit 35h[3].

Index	Name	7	6	5	4	3	2	1	0
35h	DRAM Control Register 2					MP2 Strapping Read-only 0 = 2X CPU 1 = 1X CPU			

3.1.2 Program Selected Interface Options

Several interface options that are not critical to system start-up are selectable through configuration register bits. Some of these options can even be switched dynamically, if external logic is in place to support such an arrangement.

3.1.2.1 DACKMUX Decoder Lines Source

The system designer must determine whether there is a need for the DACKMUX interface, the interface that generates the DACK#0-7 signals through an external decoder. Many portable systems use only DMA channel 2, which is available through dedicated DRQ2/DACK2# pins on the 82C465MV. However, if other DMA channels must be provided, the DACKMUX signal interface must be recovered by one of two means:

- Deleting the MA11, MDIR, and RAS4# signals of the memory interface. This option is described in the "Reduced Memory Configuration Signal Group" section.
- Relocating the EPMI1, EPMI2, LOWBAT, and LLOWBAT pins by programming bit A0h[3] = 1. Table 3-3 indicates the reassignment that takes place.

Table 3-3 Program-Selected DACKMUX Interface Recovery

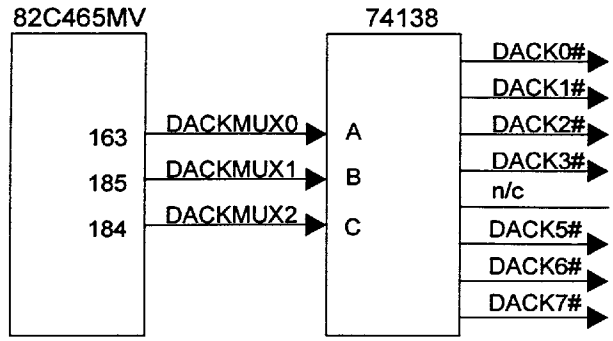
Pin	Normal Signal (Bit A0h[3] = 0)	Optional DACKMUX Replacement (Bit A0h[3] = 1)
184	EPMI1	DACKMUX2
185	EPMI2	DACKMUX1
163	LOWBAT	DACKMUX0
182	LLOWBAT	PMIMUX

DACKMUX Interface Option Enabling

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1					Enable Alternative DACKMUX Interface, See Table 3-3 0 = Disable 1 = Enable			

The hardware initialization BIOS code must select the DACKMUX replacement interface programmatically after reset by setting bit A0h[3] = 1. Figure 3-2 illustrates the typical connection.

Figure 3-2 Standard DACKMUX0-2 Connection (bit A0h[3] = 1)



3.1.2.2 EPMI Signal Source

If the DACKMUX pin functions have been moved to pins 184, 185, and 163 by setting bit A0h[3] = 1, then the displaced signal set LOWBAT, LLOWBAT, EPMI1, and EPMI2 can be recovered through an external multiplexer. This multiplexer will scan each input for status changes at the rate of once every 280ns. The signals are sampled through the multiplexer as shown in Table 3-4. One-half of an external 74153-type multiplexer is required to return the selected sample through pin 182 (PMIMUX) of the 82C465MV interface. See Figure 3 for a typical connection example.

Note that if just *one* of these power management input signals will be needed, it is easiest to eliminate the multiplexer and connect the required signal directly to the PMIMUX input of the 82C465MV. The unneeded EPMI inputs can simply be programmed to be ignored.

Table 3-4 PMIMUX Multiplex Option

74153 Mux Pin	Signal
A (input)	KBCLK
B (input)	KBCLK2
C0 (input)	LOWBAT
C1 (input)	LLOWBAT
C2 (input)	EPMI1
C3 (input)	EPMI2
Y (output)	PMIMUX

3.1.2.3 Additional EPMI Sources

The 82C465MV can use one-half of a 74153 multiplexer to provide two new signals, EPMI3 and EPMI4. This feature is convenient if one-half of the 74153 multiplexer is already being used to bring in LOWBAT, LLOWBAT, EPMI1, and EPMI2 as described above.

When this feature is enabled by setting bit A1h[5] = 1, pin 88 changes from the DRQ2 input to the EPMMUX input (the output of

the multiplexer). On the multiplexer itself, pins are defined as shown in Table 3-5.

EPMMUX Option Enabling

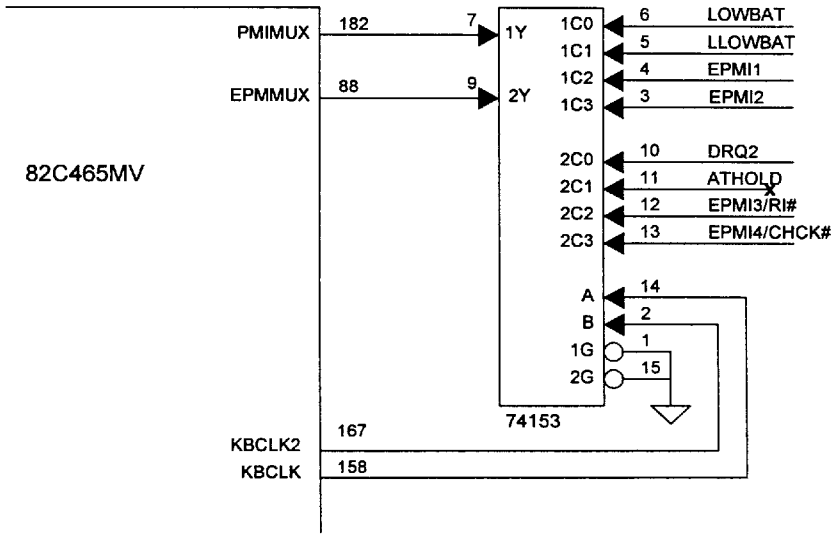
Index	Name	7	6	5	4	3	2	1	0
A1h	Feature Control Register 2			Pin 88 - for EPMI3-4 0 = DRQ2 1 = EPMMUX					

Table 3-5 EPMMUX Multiplex Option

74153 Mux Pin	Signal	Optional Signal
A (input)	KBCLK	
B (input)	KBCLK2	
C0 (input)	DRQ2	
C1 (input)	ATHOLD (MVA)	
C2 (input)	EPMI3	RI# when FAh[5]=1 (MVB)
C3 (input)	EPMI4	CHCK# when FAh[4]=1 (MVB)
Y (output)	EPMMUX	

Figure 3-3 illustrates how all EPMI pins can be multiplexed in using a single device when DACKMUX interface has displaced the EPMI-2, LOWBAT, and LLOWBAT signals.

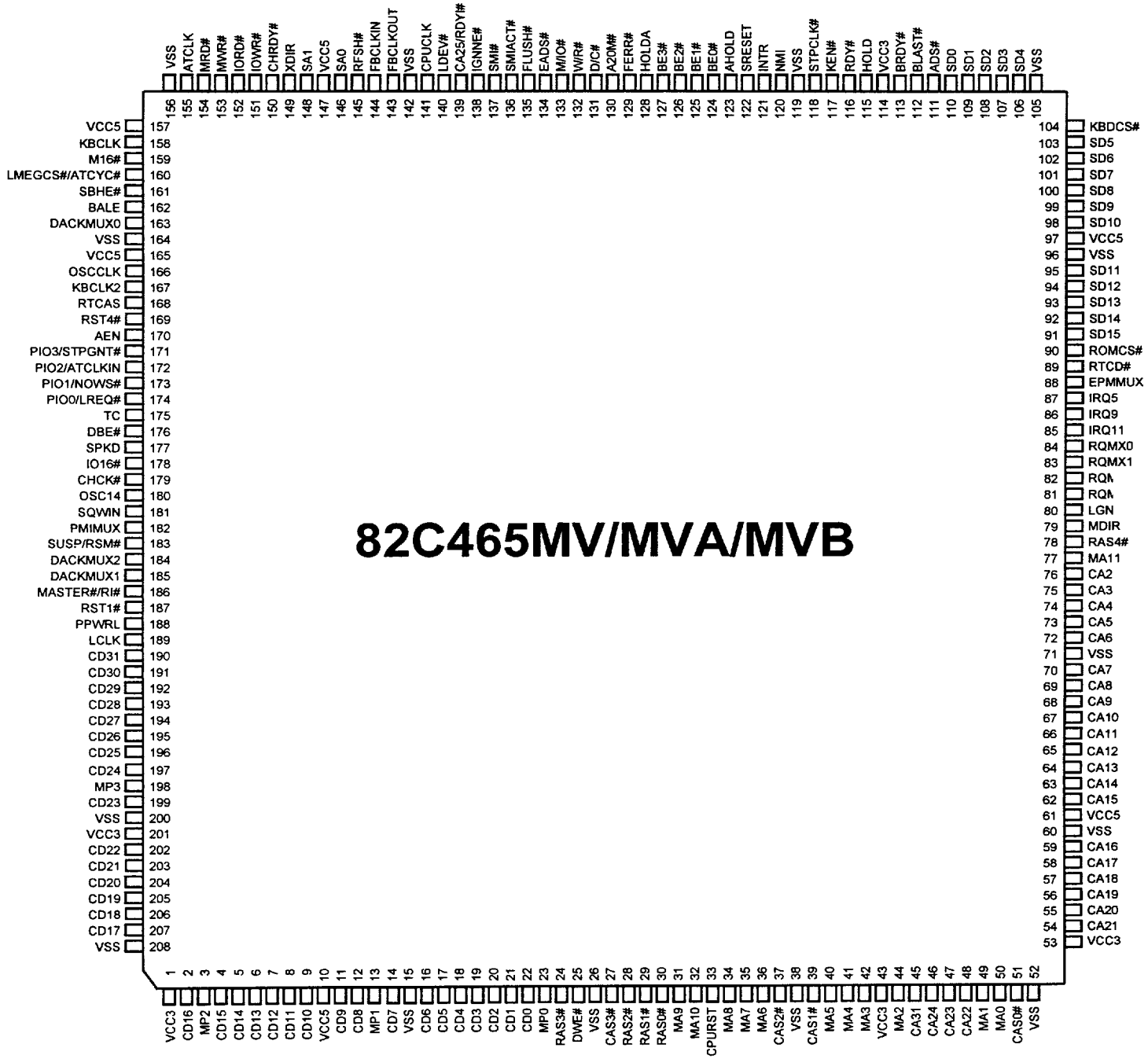
Figure 3-3 Multiplexed EPMI Input Connections



3.2 Standard Mode 82C465MV Interface

In its standard mode, the full complement of memory control options are available on the 82C465MV chip. Figure 3-4 illustrates the chip pinout in its standard mode, which requires strapping pin 198 high at reset. This figure also assumes that the full DACKMUX0-2 interface is required and is enabled through setting A0h[3] = 1.

Figure 3-4 Pin Diagram - Standard Mode



3.2.1 Reduced Memory Interface Signal Group Option

The standard 82C465MV DRAM controller hardware includes direct control of up to five banks of DRAM (RAS0#-RAS4#) and up to 12 bits of symmetrical or asymmetrical DRAM addressing (MA[11:0]), and provides memory data buffer direction control (MDIR).

To maintain backward compatibility with 82C463MV-based applications, three signals must be redefined: MA11, MDIR, and RAS4#. The three memory signals are disabled by a strap option: if pin 79 is pulled high at reset time, the chipset initialization logic will redefine three pins with their corresponding 82C463MV-located signals of DACKMUX0, 1, and 2. The memory features will not be available. Table 3-6 lists the pin changes.

Table 3-6 Strap-Selected Reduced Memory Interface Option

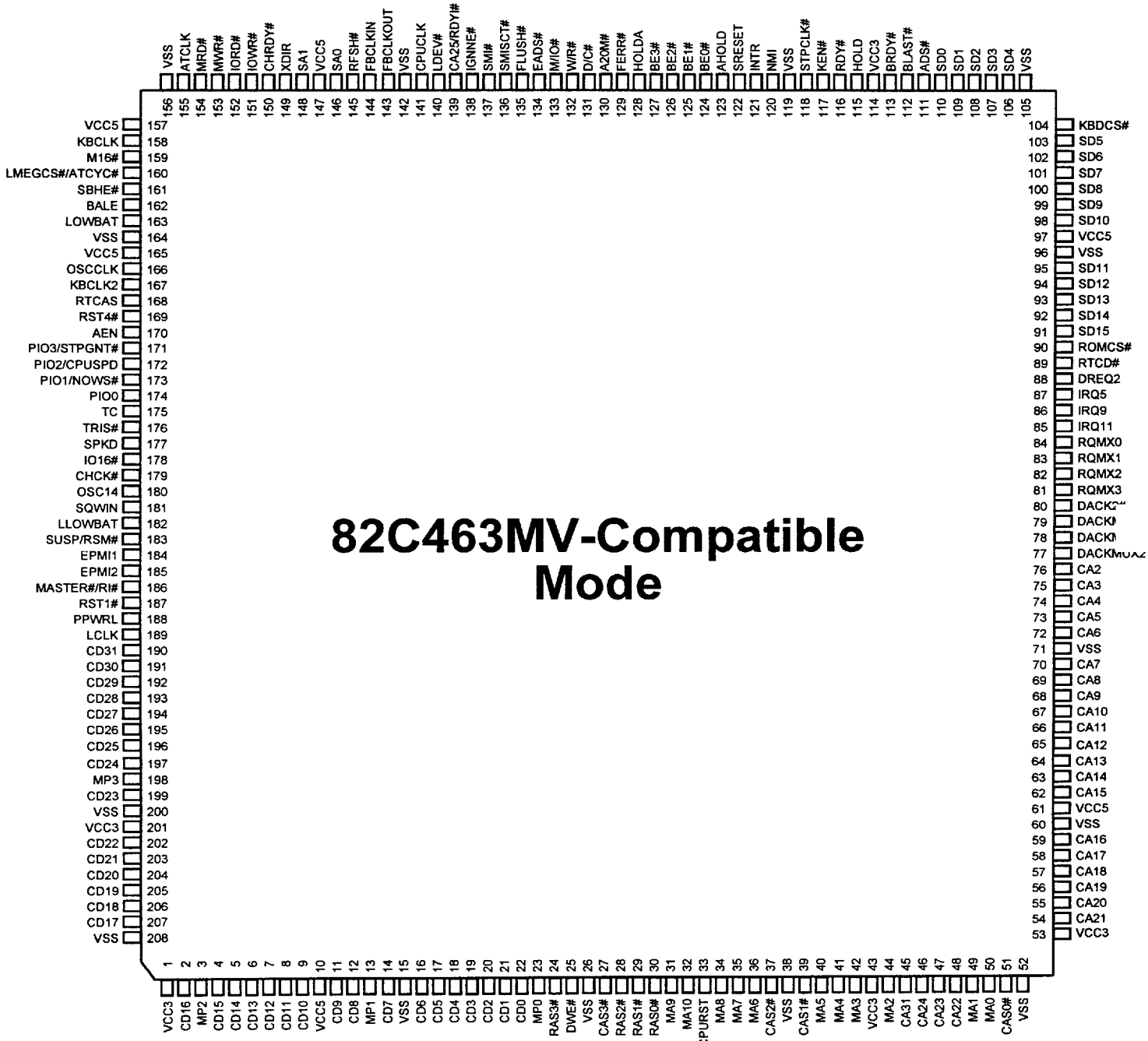
Pin	Normal 82C465MV (Pin 79 low at reset)	Reduced Memory Interface (Pin 79 high at reset)
77	MA11	DACKMUX2
78	RAS4#	DACKMUX1
79	MDIR	DACKMUX0

If pin 79 is left floating at reset, a weak internal pull-down resistor straps the line low and the 82C465MV memory interface signals will be available. If pin 79 is pulled high at reset, the old 82C463MV-compatible DACKMUX interface will be in effect. All 82C463MV designs required pin 79 to be pulled high at reset; consequently, replacing an 82C463MV chip with the 82C465MV part automatically establishes the proper interface scheme for backward compatibility.

NOTE When the DACKMUX0-2 signals are used from pins 77-79, they are provided on the CPU interface power plane. Therefore, in a mixed-voltage system, these will be 3.3V signals. The appropriate logic family must be considered when selecting the DACK decoder to avoid the high current associated with driving 3.3V signals to 5V-powered logic.

Figure 3-5 illustrates the 82C465MV pinout in its 82C463-MV compatible mode, selected by strapping pins 79 and 198 high at reset.

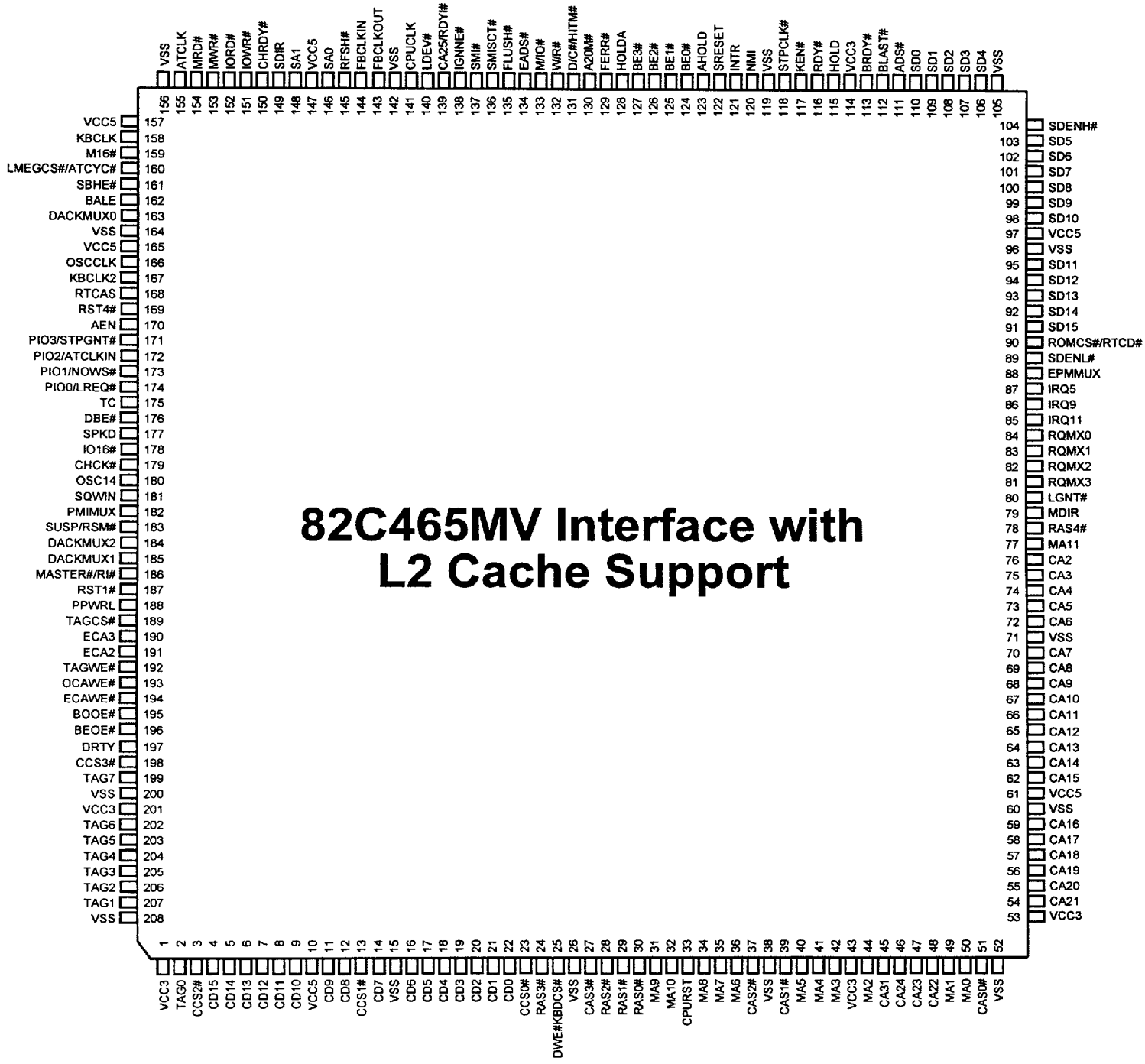
Figure 3-5 Pin Diagram - 82C463MV-Compatible Mode



3.2.2 82C465MV Interface with L2 Cache Support

An optional mode for the 82C465MV chipset provides a complete, direct set of cache control signals to an external write-back cache. This option is enabled by strapping pin 146 low and pin 198 high at reset. The external pin interface changes substantially with this option. Figure 3-6 illustrates the pinout in this mode.

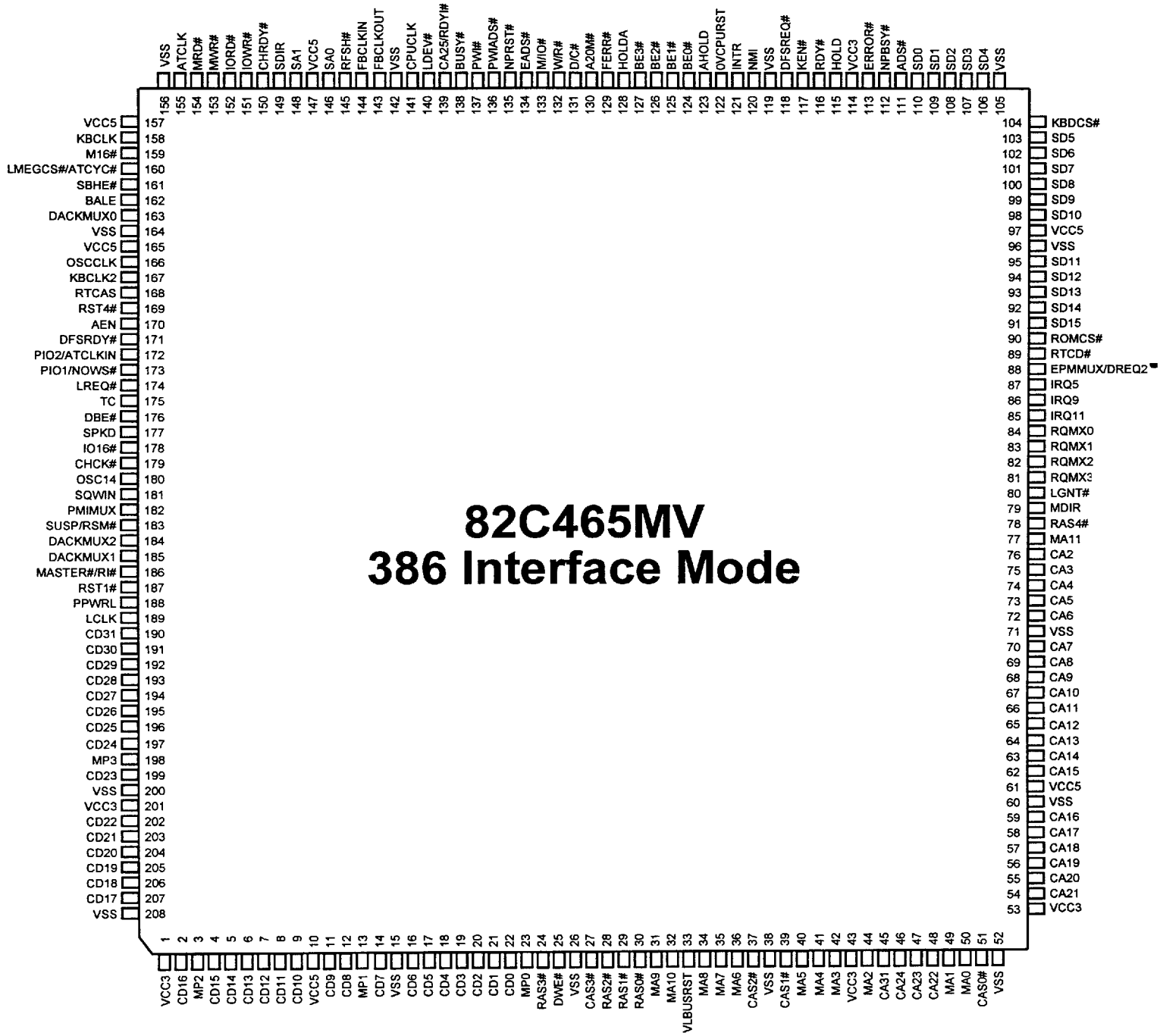
Figure 3-6 Pin Diagram - 82C465MV Interface with L2 Cache Support



3.2.3 82C465MV with 386 Interface

The 82C465MV chipset offers the option of a 386DX-type interface. This configuration supports processors like the IBM Blue Lightning. It is enabled by default with no strap pins. Figure 3-7 illustrates the 386 interface mode.

Figure 3-7 Pin Diagram - 386 Interface Mode



3.3 Pin Signal Characteristics

The signal types, drive capabilities, signal directions, power plane, and other information for each pin on the 82C465MV is provided in Table 3-7. The following legend applies to the table entries.

D = Driven (to the last state before suspend)
DH = Driven high
DL = Driven low
TS = Tri-stated
I = TTL-level input
IC = CMOS-level input
IS = Schmitt-trigger-level input
O = CMOS-level output
OD = Open-drain (open-collector) CMOS output
Drive mA = the maximum recommended steady state output current for each pin, assuming rated current loading on all pins at once.

Note that internal pull-up and pull-down resistors are engaged only during bus hold condition (which includes suspend mode), or only at hardware reset time (RST1# active) if so indicated. External pull-up resistors are suggested but may not be desirable in all cases (zero-volt suspend CPUs, for example).

Table 3-7 82C465MV Pin Characteristics

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
1	CPUVCC				--		CPUVCC
2	CD16/TAG0	I/O	Internal pull-down	4	TS		
3	MP2/CCS2#	I/O	Internal pull-down on reset	4	TS or DH	1	
4	CD15	I/O	Internal pull-down	4	TS		
5	CD14	I/O	Internal pull-down	4	TS		
6	CD13	I/O	Internal pull-down	4	TS		
7	CD12	I/O	Internal pull-down	4	TS		
8	CD11	I/O	Internal pull-down	4	TS		
9	CD10	I/O	Internal pull-down	4	TS		
10	COREVCC				--		COREVCC
11	CD9	I/O	Internal pull-down	4	TS		CPUVCC
12	CD8	I/O	Internal pull-down	4	TS		
13	MP1/CCS1#	I/O	Internal pull-down on reset	4	TS or DH	1	
14	CD7	I/O	Internal pull-down	4	TS		
15	GND				--		GND
16	CD6	I/O	Internal pull-down	4	TS		CPUVCC
17	CD5	I/O	Internal pull-down	4	TS		
18	CD4	I/O	Internal pull-down	4	TS		
19	CD3	I/O	Internal pull-down	4	TS		
20	CD2	I/O	Internal pull-down	4	TS		
21	CD1	I/O	Internal pull-down	4	TS		
22	CD0	I/O	Internal pull-down	4	TS		
23	MP0/CCS0#	I/O	Internal pull-down on reset	4	TS or DH	1	

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
24	RAS3#	O		8/12	D		CPUVCC
25	DWE# (+KBDCS#)	O		8/12	DH		
26	GND				--		GND
27	CAS3#	O		8	D		CPUVCC
28	RAS2#	O		8/12	D		
29	RAS1#	O		8/12	D		
30	RAS0#	O		8/12	D		
31	MA9	O		8/12	D		
32	MA10	O		8/12	D		
33	CPURST	O		8	DL		
34	MA8	O		8/12	D		
35	MA7	O		8/12	D		
36	MA6	O		8/12	D		
37	CAS2#	O		8	D		
38	GND				--		GND
39	CAS1#	O		8	D		CPUVCC
40	MA5	O		8/12	D		
41	MA4	O		8/12	D		
42	MA3	O		8/12	D		
43	CPUVCC				--		
44	MA2	O		8/12	D		
45	CA31	I	Internal pull-down		TS		
46	CA24	I	Internal pull-down		TS		
47	CA23	I/O	Internal pull-down	4	TS		
48	CA22	I/O	Internal pull-down	4	TS		
49	MA1	O		8/12	D		
50	MA0	O		8/12	D		
51	CAS0#	O		8	D		
52	GND				--		GND
53	CPUVCC				--		CPUVCC
54	CA21	I/O	Internal pull-down	4	TS		
55	CA20	I/O	Internal pull-down	4	TS		
56	CA19	I/O	Internal pull-down	4	TS		
57	CA18	I/O	Internal pull-down	4	TS		
58	CA17	I/O	Internal pull-down	4	TS		
59	CA16	I/O	Internal pull-down	4	TS		
60	GND				--		GND

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
61	COREVCC				--		COREVCC
62	CA15	I/O	Internal pull-down	4	TS		CPUVCC
63	CA14	I/O	Internal pull-down	4	TS		
64	CA13	I/O	Internal pull-down	4	TS		
65	CA12	I/O	Internal pull-down	4	TS		
66	CA11	I/O	Internal pull-down	4	TS		
67	CA10	I/O	Internal pull-down	4	TS		
68	CA9	I/O	Internal pull-down	4	TS		
69	CA8	I/O	Internal pull-down	4	TS		
70	CA7	I/O	Internal pull-down	4	TS		
71	GND				--		GND
72	CA6	I/O	Internal pull-down	4	TS		CPUVCC
73	CA5	I/O	Internal pull-down	4	TS		
74	CA4	I/O	Internal pull-down	4	TS		
75	CA3	I/O	Internal pull-down	4	TS		
76	CA2	I/O	Internal pull-down	4	TS		
77	MA11/DACKMUX2	O	Internal pull-down on reset	8	D/TS		
78	RAS4#/DACKMUX1/CDIR	O	Internal pull-up on reset	8	D/TS		
79	MDIR/DACKMUX0	O	Internal pull-down on reset	4	DH/TS		
80	DACK2#/LGNT#	O	External 20KΩ pull-up	4	TS		
81	RQMX3	IS					ATIOVCC
82	RQMX2	IS					
83	RQMX1	IS					
84	RQMX0	IS					
85	IRQ11	I					
86	IRQ9	I					
87	IRQ5	I					
88	DREQ2/EPMMUX	IS					
89	RTCD#/SDENL#	O	RTCD#: External 20KΩ pull-up	4	TS/DH		
90	ROMCS#(/+RTCD#)	O	External 20KΩ pull-up	4	TS		
91	SD15	I/O	External 10KΩ pull-up	8	TS		
92	SD14	I/O	External 10KΩ pull-up	8	TS		
93	SD13	I/O	External 10KΩ pull-up	8	TS		
94	SD12	I/O	External 10KΩ pull-up	8	TS		
95	SD11	I/O	External 10KΩ pull-up	8	TS		
96	GND				--		GND
97	ATIOVCC				--		ATIOVCC

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
98	SD10	I/O	External 10KΩ pull-up	8	TS		ATIOVCC
99	SD9	I/O	External 10KΩ pull-up	8	TS		
100	SD8	I/O	External 10KΩ pull-up	8	TS		
101	SD7	I/O	External 10KΩ pull-up	8	TS		
102	SD6	I/O	External 10KΩ pull-up	8	TS		
103	SD5	I/O	External 10KΩ pull-up	8	TS		
104	KBDCS#/SDENH#	O	KBDCS#: External 20KΩ pull-up	4	TS/DH		
105	GND				--		GND
106	SD4	I/O	External 10KΩ pull-up	8	TS		ATIOVCC
107	SD3	I/O	External 10KΩ pull-up	8	TS		
108	SD2	I/O	External 10KΩ pull-up	8	TS		
109	SD1	I/O	External 10KΩ pull-up	8	TS		
110	SD0	I/O	External 10KΩ pull-up	8	TS		
111	ADS#	I/O	External 10KΩ pull-up	4	TS		CPUVCC
112	BLAST#/NPBUSY#	I	External 10KΩ pull-up				
113	BRDY#/ERROR#	I/O	Internal CPU pull-up	4	TS		
114	CPUVCC						
115	HOLD	O		4	D	2	
116	RDY#	I/O	External 10KΩ pull-up	4	TS		
117	KEN#	O	Internal CPU pull-up	4	TS		
118	STPCLK#	O	Internal CPU pull-up	4	DL	4	
119	GND				--		GND
120	NMI	I/O	Internal pull-up on reset	4	DL		CPUVCC
121	INTR	I/O	Internal pull-up on reset	4	DL		
122	SRESET	O		8	DL		
123	AHOLD	O		4	DL		
124	BE0#	I/O	Internal pull-down	4	TS		
125	BE1#	I/O	Internal pull-down	4	TS		
126	BE2#	I/O	Internal pull-down	4	TS		
127	BE3#	I/O	Internal pull-down	4	TS		
128	HLDA	I					
129	FERR#	I	Internal pull-up				
130	A20M#/GA20	I/O	Internal CPU pull-up	4	TS		
131	DC#+HITM#	I	Internal pull-down			5	
132	W/R#	I/O	Internal pull-down	4			
133	M/IO#	I/O	Internal pull-down	4			
134	EADS#/NPRST#	O	Internal CPU pull-up	4	TS		

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
135	FLUSH#/SMIRDY#/HITM#	I/O	Internal CPU pull-up	4	TS		CPUVCC
136	SMIACT#/SMIADS#	I					
137	SMI#	I/O	Internal CPU pull-up	4	TS		
138	IGNNE#/BUSY#	O	Internal CPU pull-up	4	TS		
139	CA25/RDYI#	I	Internal pull-down				
140	LDEV#	I					
141	CPUCLK	O		8	DL		
142	GND						GND
143	FBCLKOUT	O		8			CPUVCC
144	FBCLKIN	IC					
145	RFSH#	I/O		8	TS		ATIOVCC
146	SA0	I/O	Internal pull-up on reset	8	TS		
147	VDDS						
148	SA1	I/O	Internal pull-down on reset	8	TS		
149	XDIR/SDIR	O		4	DH		
150	CHRDY	IS/OD		8	TS		
151	IOWR#	I/O		8	TS		
152	IORD#	I/O		8	TS		
153	MWR#	I/O		8	TS		
154	MRD#	I/O		8	TS		
155	ATCLK	O		8	DL		
156	GND						GND
157	COREVCC						COREVCC
158	KBCLK	O		4	runs		ATIOVCC
159	M16#	IS/O		8	TS		
160	LMEGCS#+ATCYC#	O	Internal pull-down on reset	4	TS		
161	SBHE#/DWR#	I/O	Internal pull-down on reset	8	TS		
162	BALE	O		8	DL		
163	LOWBAT/DACKMUX0	I/O	Internal pull-down disabled on suspend and when pin is output	4	TS		
164	GND	GND					GND
165	ATIOVCC						ATIOVCC
166	OSCCLK	IC					
167	KBCLK2	O		4	runs		
168	RTCAS	O		4	DL		
169	RST4#	O		4			
170	AEN	O		8	DL		

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
171	PIO3/ STPGNT#	I/O		4			ATIOVCC
172	PIO2/CPUSPD/ATCLKIN/ SABUFEN#	I/O		4			
173	PIO1/NOWS#/CMD#	I/O		4			
174	PIO0/LREQ#	I/O		4			
175	TC/DRD#	O		4	DL		
176	TRIS#/DBE#	O	Internal pull-up on reset	4			
177	SPKD	O		4	TS		
178	IO16#	IS					
179	CHCK#/KBCRSTIN	IS					
180	OSC14	I					
181	SQWIN	I					
182	LLOWBAT/PMIMUX	I					
183	SUSP/RSM	I					
184	EPMI1/DACKMUX2	I/O	Internal pull-up disabled on suspend **and** when pin is output	4	TS		
185	EPMI2/DACKMUX1	I/O	Internal pull-down disabled on suspend **and** when pin is output	4	TS	3	
186	MASTER#/RI/SEL#-ATB#	I					
187	RST1#	IS					
188	PPWRL	O		4	DL		CPUVCC
189	LCLK/TAGCS#/BOFF#	O		4	TS or DH	1	
190	CD31/ECA3	I/O	Internal pull-down	8	TS		
191	CD30/ECA2	I/O	Internal pull-down	8	TS		
192	CD29/TAGWE#	I/O	Internal pull-down	4	TS or DH	1	
193	CD28/OCAWE#	I/O	Internal pull-down	8	TS or DH	1	
194	CD27/ECAWE#	I/O	Internal pull-down	8	TS or DH	1	
195	CD26/BOOE#	I/O	Internal pull-down	8	TS or DH	1	
196	CD25/BEOE#	I/O	Internal pull-down	8	TS or DH	1	
197	CD24/DRTY	I/O	Internal pull-down	4	TS		
198	MP3/CCS3#	I/O	Internal pull-down on reset	4	TS or DH	1	
199	CD23/TAG7	I/O	Internal pull-down	4	TS		
200	GND						GND
201	CPUVCC						CPUVCC
202	CD22/TAG6	I/O	Internal pull-down	4	TS		
203	CD21/TAG5	I/O	Internal pull-down	4	TS		
204	CD20/TAG4	I/O	Internal pull-down	4	TS		
205	CD19/TAG3	I/O	Internal pull-down	4	TS		

Table 3-7 82C465MV Pin Characteristics (cont.)

Pin	Signal	I/O	Pull-up/pull-down, where	Drive mA	Bus hold/ suspend level	Note	Power Plane
206	CD18/TAG2	I/O	Internal pull-down	4	TS		CPUVCC
207	CD17/TAG1	I/O	Internal pull-down	4	TS		
208	GND						GND

Notes
1. These pins can be driven high during suspend if L2 cache is still powered, or can be tri-stated if the cache powered off, as selected through bit D0h[6].
2. These pins are driven low when zero-volt CPU suspend is selected through ADh[5].
3. The EPMI2 pin is driven during suspend if the chip is strapped for the RSMRST# option.
4. The CPU disconnects its STPGNT# pull-up during stop grant cycles.
5. D/C# still requires an external pull-up on the CPU side when the D/C#+HITM# gate is used for L2 cache.
6. Pins with dual drive capability are set through program registers.

3.4 Pin Descriptions

References throughout the pin descriptions depend on the chip strapping options made, as described in the "Strap-Selected Interface Options" section of this document. Refer to the table in that section to determine the features that are mutually exclusive. Internal pull-ups or pull-downs are not listed here, but in the "Pin Characteristics" section of this document. Table 3-8 provides the pin type legend for the pin descriptions that follow.

Table 3-8 Pin Description Legend

Pin Type	Legend
I	Input, TTL level
IC	Input, CMOS level
IS	Input, Schmitt-trigger TTL level
O	Output, CMOS level
OD	Output, Open-Drain (open-collector) CMOS level

3.4.1 Clock and Reset Interface

Name	Type	Pin No	Description
OSCCLK	IC	166	***Oscillator Clock Input:*** Fixed at the on-mode frequency of the CPU. ATCYC# strapping option also allows it to be 2X for a 1X CPU clock if desired.
FBCLKIN	IC	144	***Feedback Clock Input:*** This signal is used to clock most of the internal circuitry. This input should be connected to FBCLKOUT.
FBCLKOUT	O	143	***System Feedback Clock Output:*** This signal should be connected to the 82C465MV FBCLKIN input through a discrete passive component (33Ω resistor is typical). This output is 1X, regardless of whether a 1X or 2X CPU is being used.
CPUCLK	O	141	***CPU Clock Output:*** This output is 1X for 1X clock CPUs and 2X for 2X clock CPUs. It should be connected to the CPU clock input through a discrete passive component (33Ω resistor is typical).

Name	Type	Pin No	Description
SQWIN	I	181	***Square Wave Input:*** This clock input may be 32kHz or 128kHz. This clock input is used to clock the internal power management timers. This clock is also used to time refresh frequency, both active and during suspend (if suspend refresh is programmed). Set bit 40h[3] to indicate the input frequency used.
OSC14	I	180	***14.318MHz Clock Input:*** This signal goes to the IPC timer, times refresh pulse width except during suspend mode, and can be enabled to time AT bus operations. This input can be turned off during suspend mode if the KBCLK source is set to 32kHz (bit 66h[6]) or if no KBCLK and interrupt scanning is needed.
KBCLK	O	158	***Keyboard Controller Clock:*** Also used with KBCLK2 to multiplex IRQs, DRQs, and EPMI inputs.
KBCLK2	O	167	***Keyboard Controller Clock divided by two:*** Also used with KBCLK to multiplex interrupts and DMA requests.
RST1#	IS	187	***Reset One:*** Initiates a cold reset from the power supply or reset switch.
RST4#	O	169	***Reset Four:*** Indicates a cold reset to the system.
CPURST	O	33	***CPU Reset:*** Standard CPU reset (includes SMBASE reset for SMI CPUs).
SRESET	O	122	***CPU Soft Reset:*** Partially resets the CPU (the SMBASE is not reset for SMI CPUs). Used as the main CPU reset for IBM Blue Lightning.

3.4.2 CPU / VL-Bus Interface

Name	Type	Pin No	Description
CD[31:16] (No L2)	I/O	190-197, 199, 202-7	***Upper CPU Data Bus Word (No L2 Cache mode).***
Cache Signals (L2)	I/O		***Cache Data and Control Signals (L2 Cache mode):*** See L2 Cache Interface section.
CD[15:0]	I/O	2, 4-9, 11, 12, 14, 16-22	***Lower CPU Data Bus Word.***
ADS#	I/O	111	***Address Strobe:*** As an input, this pin from the CPU indicates the start of a valid address cycle. As an output, this signal is used to support the VESA local bus during ISA bus access of local memory (ISA masters and DMA).
M/IO#	I/O	133	***Memory or I/O:*** As an input, this signal from the CPU indicates whether the current cycle is a Memory or I/O access. As an output, this signal is used to support the VESA local bus during ISA bus access of local memory (ISA masters and DMA).
W/R#	I/O	132	***Write or Read:*** This signal from the CPU indicates whether the current cycle is a write or read access during ISA bus access of local memory (ISA masters and DMA).
D/C# +HITM#	I	131	***Data or Command:*** This signal from the CPU indicates whether the current cycle is a data or code access. ***Hit on Modified Line:*** CPU indication that the external master bus snoop initiated when EADS# went active has hit upon a modified internal cache line, requiring the CPU to update external DRAM before the master can continue.

Name	Type	Pin No	Description
CA31, CA24	I	45, 46	***CPU Address A31, A24 inputs.***
CA[23:10]	I/O	47, 48, 54-59, 62-67	***CPU Address Lines A23 to A10:*** These pins are inputs for CPU and master cycles. These pins are outputs for DMA cycles.
CA[9:2]	I/O	68-70, 72-76	***CPU Address Lines A9 to A2:*** These pins are inputs for CPU and master cycles. These pins are outputs for DMA and refresh cycles.
BE#[3:0]	I/O	127-124	***Byte Enables:*** For CPU cycles, these inputs are the CPU byte enables. For ISA bus access of local memory, they are outputs.
RDY#	I/O	116	***Ready:*** Indicates completion of the current VL-bus cycle. Local-bus devices can drive RDY# directly to the CPU as long as they use a tri-state driver that does not drive while LDEV# is inactive.
HOLD	O	115	***Hold Request:*** Requests system control from the CPU.
HLDA	I	128	***Hold Acknowledge:*** CPU acknowledges a hold request with this signal.
NMI	O	120	***Non Maskable Interrupt:*** Indicates the occurrence of a non maskable interrupt to the CPU.
INTR	O	121	***Interrupt:*** Indicates occurrence of a maskable interrupt to the CPU.
KEN#	O	117	***Cache Enable:*** Indicates to the CPU that current bus cycle is cacheable.
BRDY# (486)	I/O	113	***Burst Ready (486 mode):*** Indicates termination of a burst cycle. The 82C465MV controller logic also terminates non burst cycles with BRDY# on occasion.
ERROR# (386)	I/O		***Numeric Processor Error (386 mode):*** Indicates a calculation error from the numeric processor.
EADS# (486)	O	134	***External Address Strobe (486 mode):*** Indicates that an external bus master has placed a valid address on the CPU address bus, and is used by the CPU to invalidate internal cache (and generate HITM# if available).
NPRST# (386)	O		***Numerical Processor Reset (386 mode):*** This signal is used to reset the numeric co-processor.
AHOLD	O	123	***Address Hold Request:*** The CPU will float its address bus in the clock following AHOLD going active. The 82C465MV generates AHOLD after a cache write-back has been completed; AHOLD active while HITM# is inactive constitutes the BOFF# signal to the CPU.
BLAST# (486)	I	112	***Burst Last (486 mode):*** Indicates that the next BRDY# will complete the current burst cycle.
NPBUSY# (386)	I		***Numeric Processor Busy (386 mode):*** Indication from the numeric processor that the current operation has not completed.
LDEV#	I	140	***Local Device:*** Local devices drive this input to indicate that they are responding to the current cycle. This signal must be asserted by the end of the first T2 (with appropriate setup time) for local device recognition. This signal has an internal 50K pull up resistor.

Name	Type	Pin No	Description
CA25 (64MB)	I	139	***CA25:*** Address for increased local DRAM capacity. This pin must function as CA25 if DRAM memory size over 32MB is required.
RDYI# (32MB)	I		***RDYI# - (LRDY# from VL bus):*** Local devices drive this input to indicate that the current cycle is completed. The function of pin 139 is determined by pin 161 strapping. If CA25 is required as well as LRDY#, the VL-bus device can drive the CPU RDY# signal directly (LRDY# is defined as open-collector) up to 33MHz.
FERR#	I	129	***Floating Point Error:*** Driven by the co-processor to indicate an error condition when an unmasked exception occurs.
IGNNE# (486)	O	138	***Ignore Numeric Errors (486 mode):*** Instructs the CPU to ignore the numeric co-processor's error output.
BUSY# (386)	O		***Numeric Processor Busy (386 mode):*** Indicates that the numeric co-processor has not completed its current operation.
TAGCS# (L2)	O	189	***Tag Chip Select (L2):*** See L2 Cache Interface section.
BOFF# (No L2, D4h[0]=0)	O		***CPU Backoff (No L2, D4h[0]=0, new memory interface:*** This pin becomes BOFF# to the CPU when there is no L2 cache present and the 465 memory interface is enabled.
LCLK (No L2, D4h[0]=1)	O		***Local Bus Clock (No L2, D4h[0]=1):*** For 2X clock CPUs, this signal is the VESA local bus 1X clock. For 1X clock CPUs, this signal is a 2X clock output. (In this mode, the VESA local bus 1X clock comes from pin 143 FBCLKOUT)
STPCLK#	O	118	***Stop Clock:*** Requests a change in the clock frequency from the CPU.
SMI#	I/O	137	***System Management Interrupt:*** Request System Management Mode (SMM) operation from the CPU. For support of some CPUs, this pin is bi-directional. See bit 5Bh[4].
SMIACT#/ SMI-ADS#	I	136	***SMI Process Active or SMI Address Strobe:*** This pin is used to indicate that the CPU is operating is System Management Mode (SMM). Bit 5Bh[4] selects the function of this pin.
FLUSH#	O	135	***Cache Flush:*** The FLUSH# signal commands flushing of the internal CPU cache on entry to SMM.
SMIRDY#	O		***SMI Ready:*** The SMIRDY# signal responds to SMIADS# for CPUs that require this signal interface. Use bit 6Bh[6] to select this function.
HITM#	I		***Hit On Modified line:*** CPU indication that the external master bus snoop initiated when EADS# went active has hit upon a modified internal cache line, requiring the CPU to update external DRAM before the master can continue. Selected when D6h[4]=1.

3.4.3 DRAM Interface

Name	Type	Pin No	Description
STRP[3:0] (reset)	I	198, 3, 13, 23	***Strap Option Pins (at reset):*** During a system reset, these pins provide strapping options to determine system interface function. Refer to the "Strap-Selected Interface Options" section for complete details.
CCS[3:0]# (L2)	O		***Cache Chip Select 0-3 (L2):*** Refer to the "L2 Cache Interface" section.
MP[3:0] (No L2)	I/O		***Memory Parity [3:0] (No L2):*** Generate and check parity bits from DRAM (if used; usually not provided in notebook systems).
DWE# (No L2)	O	25	***DRAM Write Enable (No L2):*** For DRAM memory cycles, this signal is the DRAM write strobe.
DWE# + KBDCS# (L2)	O		***DWE# combined with KBDCS# (L2):*** This signal must be qualified with AEN low to decode the true KBDCS# signal. Refer to the separate KBDCS# description also.
CAS[3:0]#	O	27, 37, 39, 51	***Column Address Strobes 3 to 0:*** These outputs drive the CAS# inputs on DRAM bytes 3 to 0.
RAS[3:0]#	O	24, 28, 29, 30	***Row Address Strobes 3 to 0:*** These outputs drive the RAS# inputs on DRAM banks 3 to 0.
RAS4#	O	78	***Row Address Strobe 4:*** This output drives the RAS# input from DRAM bank 4. Note that RAS4# is not available if the pin 79 strapping option is used to defeat it.
CDIR	O		***Compact ISA Direction:*** Controls buffer direction for CISA cable drive buffer. Option available when bit F8h[1]=1.
MDIR	O	79	***Memory Buffer Direction Signal:*** Note that this signal is not available if the pin 79 strapping option is used to defeat it.
MA11	O	77	***Memory Address Signal MA11:*** Note that this signal is not available if the pin 79 strapping option is used to defeat it.
MA[10:0]	O	32, 31, 34-36, 40-42, 44, 49, 50	***Memory Address Signal MA10 to MA0 and Peripheral Power Control Signals:*** For DRAM cycles, these are MA addresses. For AT Bus cycles (ATCYC# low): 1) MA[11:0] are peripheral power control pins latched externally with signal PPWRL; 2) MA[9:6] are decoded by AEN and ATCYC# low to become programmable chip select signals CSG1#, CSG0#, CSG3#, and CSG2# respectively.

3.4.4 L2 Cache Interface

This interface is only available if the pin 146 strap option is set for L2 cache support. Refer to the "Strap-Selected Interface Options" section for strapping information.

Name	Type	Pin No	Description
DRTY	I/O	197	***Dirty bit.***
BEOE#	O	196	***Cache Output Enable, Even.***
BOOE#	O	195	***Cache Output Enable, Odd.***
ECAWE#	O	194	***Cache Write Enable, Even.***

Name	Type	Pin No	Description
OCAWE#	O	193	***Cache Write Enable, Odd.***
TAGWE#	O	192	***Tag RAM Write Enable.***
ECA2	O	191	***Cache CA2.***
ECA3	O	190	***Cache CA3.***
TAG[7:0]	I/O	199, 202-207, 2	***Tag RAM Data.***
TAGCS#	O	189	***Tag RAM Chip Select.***
CCS0-3#	O	23, 13, 3, 198	***Cache Chip Select 0-3.***
SDIR	O	149	***SD[15:0]-to-CD[31:16] buffer direction control.***
SDENH#	O	104	***SD[15:8]-to-CD[31:24] buffer enable.***
SDENL#	O	89	***SD[7:0]-to-CD[23:16] buffer enable.***

3.4.5 AT-Bus Interface

Name	Type	Pin No	Description
SD[15:0]	I/O	91-95, 98-103, 106-110	***AT Bus Data SD15 to SD0.***
ATCLK	O	155	***AT Bus Clock.***
BALE	O	162	***AT Bus Address Latch.***
MRD#	I/O	154	***AT Bus Memory Read Command.***
MWR#	I/O	153	***AT Bus Memory Write Command.***
IORD#	I/O	152	***AT Bus I/O Read Command.***
IOWR#	I/O	151	***AT Bus I/O Write Command.***
CHRDY	IS/OD	150	***AT Bus Channel Ready.***
M16#	IS/O	159	***16-bit Memory Slave:*** This AT Bus signal indicates a 16 bit memory slave is responding. Normally an input, this signal becomes an output when a master accesses local memory.
IO16#	IS	178	***16-bit I/O Slave:*** This AT Bus signal indicates a 16 bit I/O slave is responding.
STRP5-4 (reset)	I	148, 146	***Strap Options:*** Refer to "Strap-Selected Interface Options" section.
SA[1:0] (normal)	I/O		***AT Bus Address SA1 to SA0:*** Provide the remaining two SA bus address lines, which cannot be buffered directly from the CPU CA bus.

Name	Type	Pin No	Description
STRP6 (reset)	I	161	***Strap Options:*** Refer to "Strap-Selected Interface Options" section.
SBHE# (normal)	I/O		***System Byte High Enable:*** Indicates a transfer on the upper byte of the AT data bus SD[15:8].
DWR#	O		***Drive Write:*** Provides write command for local-bus IDE cycles when qualified by DBE.
RFSH#	I/O	145	***Refresh:*** Indicates AT-bus refresh cycles.
LMEGCS# + ATCYC#	O	160	***Lower Memory Chip Select and AT I/O Cycle Indicator:*** This signal is active when the memory cycle address is below 1MB. It is used to generate SMRD# and SMWR#. For AT bus I/O cycles, it is used to generate CSG0-3#.
CHCK#	IS	179	***AT Bus Channel Check:*** Provides the system with parity information about memory or devices on the AT bus. It indicates a non correctable system error and is one of the sources used to generate a CPU NMI.
KBCRSTIN	IS		***Keyboard Controller RESET Input:*** Used when bits 79h=11 (MVA). CHCK# can be recovered on EPMI4 (MVB). Internally synchronized to HLT to generate CPURST with the correct timing.

3.4.6 IPC (82C206) Interface

Name	Type	Pin No	Description
RQMX3	IS	81	***Multiplexed input signals of DRQ1, DRQ3, DRQ6, DRQ7.***
RQMX2	IS	82	***Multiplexed input signals of IRQ10, DRQ0, DRQ5, IRQ15.***
RQMX1	IS	83	***Multiplexed input signals of IRQ6, IRQ8, IRQ4, IRQ12.***
RQMX0	IS	84	***Multiplexed input signals of IRQ3, IRQ1, IRQ7, IRQ14.***
IRQ11	I	85	***Interrupt request channel 11.***
IRQ9	I	86	***Interrupt request channel 9.***
IRQ5	I	87	***Interrupt request channel 5.***
DREQ2	I	88	***DMA request channel 2.***
EPMMUX	I		***EPMI Input Mux:*** Option available when bit A1h[5] = 1.
DACK2# (486)	O	80	***DMA Channel Two Acknowledge (486 mode, 386 mode option).***
NPINT (386)	O		***Numeric Processor Interrupt (386 mode).***
LGNT#	O		***Local Bus Grant:*** Option available when bit A0h[5] = 1.
STRP7-9 (reset)	I	79-77	***Strap Options:*** Refer to "Strap-Selected Interface Options" section.
Memory Controls (normal)	O		***Memory Control Lines:*** Refer to "DRAM Interface" section.
DACKMUX0-2 (if mem. controls are defeated)	O		***Encoded DACK0-7# Signals (463-compatible solution):*** Available here only if memory controls interface is defeated through pin 79 strap option. Connect to 74138 decoder to derive DACK0#-DACK7# (DACK4# not valid). Note that pins 77-79 are on CPU I/O power plane and may be 3.3V signals.

Name	Type	Pin No	Description
EPMI pins (default)	I	163, 185, 184	***LOWBAT, EPMI2, EPMI1 Interface (at power on default):*** Refer to PMU Interface section for full details.
DACKMUX0-2 (if programmed)	O		***Encoded DACK0-7# Signals (preferred solution for new designs):*** Available here only if bit A0h[3] = 1. EPMI signals must be relocated to an external multiplexer. Connect to a 74138 decoder to derive DACK0#-DACK7# (DACK4# not valid). Note that these pins are on the AT I/O power plane and may be 5V signals.
TC/DRD#	O	175	***AT Bus Terminal Count/Drive Read:*** Provides read command for local bus IDE when qualified by DBE#.
AEN	O	170	***AT Bus Address Enable:*** Indicates that the DMA controller has taken control of the CPU address bus and the AT bus command lines.

3.4.7 PMU Interface

Name	Type	Pin No	Description
LOWBAT	I	163	***Low Battery Indication:*** Has programmable polarity: bit 40h[4].
DACKMUX0	I		***DACKMUX0:*** See "IPC Interface" section.
LLOWBAT	I	182	***Very Low Battery Indication:*** Has programmable polarity: bit 40h[5].
PMIMUX	I		***EPMI Multiplex Input:*** Refer to "Program-Selected Interface Options" section.
EPMI1	I	184	***External PMI Source One:*** See bit 40h[1] to select polarity.
DACKMUX2	I		***DACKMUX2:*** See "IPC Interface" section.
EPMI2	I	185	***External PMI Source Two:*** See bit 40h[2] to select polarity.
DACKMUX1	O		***DACKMUX1:*** See "IPC Interface" section.
RSMRST#	O		***Resume Reset:*** Selected by strap option on pin 148.
SUSP/RSM	I	183	***Suspend/Resume:*** Indirect suspend source (can cause SMI to initiate suspend); direct resume source (cannot be disabled).
TRIS# (no IDE)	O	176	***Suspend Mode Indication (no IDE):*** A '0' indicates that the system is in suspend mode.
DBE (IDE enabled)	O		***Data Buffer Enable (IDE):*** Enables the command line buffer to the IDE interface. Refer to the "IDE Interface" section.
PIO0 (no master)	I/O	174	***User definable I/O pin PIO0.***
LREQ# (VL-bus master)	I/O		***Local Bus Master Request:*** Option available when bit A0h[5] = 1.
PIO1	I/O	173	***User definable I/O pin PIO1.***
NOWS# program option	I		***AT Bus Zero Wait State:*** Option available when bit 66h[1] = 1.
CMD# program option	O		***Compact ISA Command:*** This signal generates ISA-like command timing to the CISA peripheral devices. Option available when bit F8h[0]=1.

Name	Type	Pin No	Description
PIO2	I/O	172	***User definable I/O pin PIO2.***
CPUSPD	O		***CPU Full Speed Indicator:*** A '1' indicates that the CPU is operating at full speed. Bit 66h[2] selects this option.
ATCLKIN	I		***AT Clocking Source Input:*** Bit A0h[1] = 1 enables this source.
SABUFEN#	O		***SA Bus Buffer Enable:*** Allows CA-to-SA bus buffer to be tristated when not in use to save power. Bit 79h[3]=1 enables this function (MVA).
PIO3	I/O	171	***User Definable I/O pin PIO3.***
STPGNT#	I		***CPU Stop Clock Grant Signal:*** Bit 66h[3] selects the function of this pin.
PPWRL	O	188	***Peripheral Power Control Signal Latch:*** This signal is used to control the external latching of the peripheral power control signals PPWR11-0 from MA[11:0]. This signal is pulsed during reset to pre-set the external latch.
MASTER#	I	186	***AT-Bus Master:*** Note: MASTER# is internally decipherable and need not be input. This pin should always be programmed as RI#.
RI#	I		***Ring Indicator:*** When the CISA interface is not needed (bit F8h[0]=0), this pin becomes the RI input and can be used to resume the system after entry to Suspend mode.
SEL#/ATB# + CLKRUN# program option	I		***Compact ISA Select.*** The CISA peripheral device (usually the 82C852) asserts SEL# to claim a CISA cycle after decoding its address on ALE active. The 82C465MVB can optionally inhibit the ISA command lines when it receives SEL#. ***AT Backoff.*** The CISA peripheral device asserts ATB# between cycles to generate an interrupt on the 82C465MVB. ***Clock Run.*** The CISA peripheral device asserts CLKRUN# to restart ATCLK when the 82C465MVB has issued a STPCLK broadcast cycle and stopped ATCLK.

3.4.8 Miscellaneous Signal Interface

Name	Type	Pin No	Description
A20M# (486)	O	130	***A20 Mask Control (486 mode).***
GA20 (386)	I/O		***Gated A20 line (386 mode):*** Can be forced to A20M# function.
RTCAS	O	168	***RTC Address Strobe:*** This signal is active when the system accesses port 70h.
SPKD	O	177	***Speaker signal.*** Uses special protocol when CISA is enabled.
ROMCS# (No L2)	O	90	***ROM Chip Select:*** For memory cycles in the proper range, this is the ROM chip select.
ROMCS# + RTCD# (L2)	O		***ROM Chip Select and RTCD# Signal:*** Combined signals. Valid as RTCD# only when AEN is low.
RTCD# (No L2)	O	89	***Real Time Clock (RTC) Data Strobe Qualifier:*** For I/O cycles at port 70h, this is used to generate the RTC data strobe and read/write signal.
SDENL# (L2)	O		***L2 Cache Control Signal:*** Refer to "L2 Cache Interface" section of this table.

Name	Type	Pin No	Description
KBDCS# (No L2)	O	104	***Keyboard Controller Chip Select:*** This signal is qualified with AEN to produce the keyboard chip select.
SDENH# (L2)	O		***L2 Cache Control Signal:*** Refer to "L2 Cache Interface" section of this table.
XDIR	O	149	***XD Bus Data Buffer Direction Control:*** A '1' indicates data transfer from the 82C465MV SD bus to the XD-bus device. Normally high, it is low for the following two conditions: 1) When ROMCS# and MRD# are both active, and 2) During reads from I/O ports 060h, 064h, 070h, and 071h.
SDIR	O		***L2 Cache Control Signal:*** Refer to "L2 Cache Interface" section of this table.

3.4.9 Power and Ground Pins

Name	Type	Pin No	Description
VDD CPU I/O	Power	1, 43, 53, 114, 201	+3.3V
VDDS AT I/O	Power	97, 147, 165	+5V
VDDCORE Core Vcc	Power	10, 61, 157	Same as VDDS
GND	GND	15, 26, 38, 52, 60, 71, 96, 105, 119, 142, 156, 164, 200, 208	Ground

4.0 Functional Description

4.1 463/465 Chipset Programming Comparison

Many OPTi chipsets are designed to be register-compatible with their immediate predecessors. In the case of the 82C465MV, virtually all of the programming registers in the 82C463MV are either directly available or are emulated.

However, certain 82C463MV features have been superseded by new 82C465MV registers. In such cases, only the recommended registers have been documented here. Programmers can refer to the 82C463MV Data Book for information on older registers, but are discouraged from using non current registers. The reason for this is that in the eventual successor to the 82C465MV, certain 82C463MV registers will no longer be available.

Code designed to accommodate both the 82C463MV and 82C465MV to determine the chipset type should read the product indicator bits 30h[7:6].

Index	Name	7	6	5	4	3	2	1	0
30h	Control Register 1	82C46x Product Indicator - read-only. 00 = 82C463/463MV 01 = 82C465MV 10 = 82C465MVA 11 = 82C465MVB							

4.2 CPU and VL-Bus Interface

The 82C465MV chipset is typically used to support a CPU with a 32-bit 486-type interface, but can also support 32-bit 386DX-type interfaces. The standard CPU signaling interface is provided and is known as the VESA local bus (VL bus). The fully featured VL-bus interface additionally allows for connection of high-speed peripheral devices such as the OPTi 92C168 Local-Bus LCD Controller or the OPTi 82C823/824 CardBus Controller chipset.

This section describes the CPU interface as part of the total VL-bus support provided by the 82C465MV chipset. All VESA-standard signals are provided either on dedicated pins or as options.

4.2.1 Basic Command Interface

The VL bus interface uses a standard command interface whose function and timing are described in CPU data books. The primary signal interface involves three cycle-type signals: M/IO#, W/R#, and D/C#; and a pair of cycle start and completion handshaking signals, ADS# and RDY#. Both 386 and 486 interfaces use these signals.

The 486 interface allows for read bursts (multiple high-speed sequential read cycles) and therefore provides BRDY# and BLAST# to control burst read cycles.

For the 486 interface, and for certain hybrid 386/486 interfaces as well, signals KEN#, AHOLD, EADS#, and FLUSH# are provided for cache control.

4.2.1.1 Cycle Signals

The CPU interface provides three status signals to indicate cycle type for the current cycle: M/IO#, to distinguish memory accesses from I/O access; W/R#, to distinguish writes from reads; and D/C#, to distinguish data accesses from code fetches. All of these signals are valid for a guaranteed amount of time before the CPU or VL-bus master sets its ADS# signal active. The rising edge of ADS# (when it goes from active to inactive) indicates that the CA bus address and the cycle type signals are valid.

Normally, the 82C465MV samples the state of ADS# according to the CPUCLK signal. All of the cycle type signals are guaranteed by the CPU manufacturer to be valid for a certain amount of time from a CPUCLK clock edge when ADS# is active. Therefore, if the 82C465MV logic sees ADS# low within this "window" it can determine the cycle type without actually waiting for the rising edge of ADS#. This logic improves performance dramatically, since a full CPUCLK is saved on each CPU cycle.

However, at high bus speeds (40-50MHz), the ADS# signal may not always be synchronized as well with the CPUCLK signal and only after the rising edge of ADS# will the controller logic be guaranteed to capture the correct cycle being signaled. There-

fore, the 82C465MV logic includes a control bit that allows the selection of whether the cycle type will sampled when ADS# is seen low to improve performance, or whether ADS# will be latched and the cycle sampled on the next clock edge to guarantee functionality. Bit D1h[6] selects this functionality, and defaults to latching ADS# so that booting at any speed will be possible. The BIOS should check for slower speed CPUs and clear this bit if possible for better performance.

ADS# Sampling Control

Index	Name	7	6	5	4	3	2	1	0
D1h	L2 Cache Control Register 2		ADS# sampling 0 = Sample on ADS# low 1 = Latch ADS#, sample on next cycle						

4.2.2 Local Device Interface

The 82C465MV allows VL-bus peripheral devices to share the local bus with the CPU and the numeric co-processor. The performance of these devices, which may include the video controller, LAN adapters, and other PC/AT controllers, will dramatically increase when allowed to operate in this high speed environment. These devices are responsible for their own address and bus cycle decoding and must operate properly at the elevated frequencies required for the local CPU bus.

4.2.2.1 LDEV# Operation

The LDEV# input signal to the 82C465MV indicates whether a local-bus device will be responding to the current cycle. If the access is not in the local DRAM range and the 82C465MV samples LDEV# active at the end of the first T2 clock cycle, it will allow the responding device to assume responsibility for terminating the current local cycle. Otherwise, the 82C465MV passes on the cycle to the AT bus. If the access is in the local DRAM range, the 82C465MV logic ignores LDEV#.

Normally, if a local bus device asserts LDEV# and then removes it without responding with its LRDY#, the 82C465MV will await a response forever and the system will hang. Bit ADh[7] is provided to avoid this situation.

LDEV# Control

Index	Name	7	6	5	4	3	2	1	0
ADh	Feature Control Register 3	Ignore unfinished LDEV# cycles 0 = wait 1 = ignore							

Ignore Unfinished LDEV# Cycles bit ADh[7] - allows LDEV# to be asserted to claim a cycle for the local bus, but then allows the cycle to be given up if no LRDY# (RDYI# or RDY#) is generated. When ADh[7] = 0 and LDEV# is sampled asserted, the logic gives up the cycle to the local bus and awaits RDY# from the local device. If the local device never actually responds, the system will hang. If ADh[7] = 1 and LDEV# is sampled asserted, the logic gives up the cycle to the local bus and awaits RDY# as usual. However, if no RDY# is returned before LDEV# goes inactive, the chipset logic takes back ownership of the cycle and forwards it to the AT bus.

Note that there is no "timeout" when the "ignore unfinished LDEV# cycles" option is selected. The 82C465MV chip will wait indefinitely as long as LDEV# remains active and no RDY# signal is returned. As soon as LDEV# goes inactive with no RDY# signal yet received, the 82C465MV takes back control of the cycle.

4.2.2.2 LRDY# Operation

When the local bus device has completed its operation, it uses the VL-bus local ready signal LRDY# to terminate the cycle. The LRDY# signal is defined in the VL-bus specification to be driven by a tri-state buffer. Therefore, at bus speeds up to 33MHz, LRDY# can simply be tied directly to the CPU RDY# input. Above 33MHz, the VL-bus device may not meet the timing requirements of CPU RDY#. In this case, LRDY# can be connected to the RDYI# input of the 82C465MV for re-synchronization to the CPU RDY# signal.

The 82C465MV provides bit ADh[0] to control whether the RDYI# input will be buffered to drive the CPU RDY# line directly, or will be latched and synchronized to the CPU RDY# input on the next clock. Refer to the Numeric Co-processor Interface section of this document for information on this bit.

4.2.2.3 VL-Bus Arbitration Logic

The 82C465MV provides arbitration among the various system resources: CPU, DMA, VL-bus masters, AT-bus masters, and refresh logic.

During cycles where the CPU is not the system master, the 82C465MV asserts HOLD to the CPU. The CPU responds to an active HOLD signal by generating HLDA after completing its current bus cycle and placing most of its output and I/O pins in a high impedance state. After the CPU relinquishes the bus, the 82C465MV responds by issuing the appropriate signal.

- For a refresh timer request, the 82C465MV logic runs a DRAM refresh cycle and AT-bus refresh cycle.
- For a DMA request or master request initiated by assertion of one of the DRQ lines, the chip provides the corresponding DACK# signal along with IOR#+MEMW# or IOWR#+MEMR#.
- For a VL-bus master request indicated on LREQ#, the chip responds with LGNT#.

The AT-bus controller in the 82C465MV arbitrates between hold and refresh requests, deciding which will own the bus once the CPU relinquishes control with the HLDA signal. The arbitration between refresh and DMA/masters is based on a FIFO priority. However, a refresh request (RFSH#) will be internally latched and serviced immediately after the DMA/master finishes its request if queued behind HOLD. HOLD must remain active to be serviced if the refresh request comes first.

4.2.3 VL-Bus Masters

The 82C465MV optionally provides one set of LREQ# and LGNT# signals to support bus masters. These signals are used in conjunction with devices such as the OPTi 82C822 VL-PCI Bridge chip or the OPTi 82C823/824 CardBus Controller chipset that must gain ownership of the VL bus for certain bus agents.

Support for a single bus master request is adequate for most applications, since the PCI subsystem can often combine multiple master requests into the single request line available. Consequently, this solution will not limit the PCI options if the 82C822 or 82C823/824 solutions are chosen.

The bus master support actually is provided on the VL bus. Therefore, any VL-bus master can make use of these signals.

4.2.3.1 Hardware Considerations

Enabling bus master support requires that pin 80, the dedicated DACK2# signal, be replaced by LGNT#, and that pin 174 (PIO0) be replaced by the LREQ# input from the VL bus. DACK2# is always available on the DACK decoder (decoding DACKMUX0-2), so only PIO0 is lost and no additional TTL is required.

4.2.3.2 Programming

The VL-bus master support feature is enabled by setting bit A0h[5] = 1. DACK2# is always available on DACK decoder, regardless of the setting of this bit.

Bus Master Enabling

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1			Enable Local Bus Master Support 0 = PIO0 and DACK2# 1 = LREQ# and LGNT#					

4.2.4 Data Bus Conversion/Data Path Logic

The 82C465MV performs data bus conversion when the CPU accesses 8- or 16-bit devices through 16- or 32-bit instructions. It also handles DMA and master cycles that transfer data between local DRAM or cache memory and locations on the AT or VL bus.

4.2.4.1 CPU Data Bus Multiplex Option

When the system design includes an external write-back cache, the signal pins normally used for the upper 16 lines of the CPU data bus are converted to cache tag data and control signals. Consequently, the 82C465MV must use an external 16-bit buffer to move data between CD31:16 and SD15:0 for AT-bus operations and internal register accesses.

This external buffer is controlled by three signals: SDENH#, SDENL#, and SDIR, that control a 16-bit '245-type buffer. If the CPU interface operates at 3.3V and the AT bus at 5V, this buffer must translate the level. The control signals mentioned will always be at the level of the Vcc supplied to AT I/O pads, which is appropriate for all standard designs.

Also when the external write-back cache is used, the XDIR signal is redefined and must be derived from IOW#, MEMR#, and the chip selects of all devices on the XD bus.

4.2.5 Numeric Co-processor Interface

The 82C465MV monitors FERR# and NPBUSY# to provide support for the 80387 co-processor (NPX) when the chipset is strapped for an 80386-type interface. There are no provisions for an external co-processor when a 486SX CPU is used.

4.2.5.1 Hardware Considerations

The NPX asserts FERR# during a power-on reset to indicate its presence. If the 82C465MV logic senses FERR# low when it asserts NPURST, it automatically generates LDEV# to itself whenever CA31 is high for a cycle. This feature allows the co-processor to generate its own RDY# to the CPU. The automatic co-processor recognition feature can be disabled through bit ADh[1].

The 82C465MV treats any access to the NPX address space as an AT cycle if the NPX is not installed, and generates its own RDY# to the CPU at the end of the AT cycle. Note that a VL-bus device can also respond with LDEV# to claim the cycle, and is responsible for generating LRDY# (RDY#) to the CPU.

When the NPX has completed its cycle, the NPX RDY# signal terminates the cycle by asserting:

1. CPU RDY# with a fast open-collector driver controlled by the NPX RDY# signal (not generally a practical option to implement)
2. RDYI# to the 82C465MV, which latches it and drives its RDY# output to the CPU low on the next clock (bit ADh[0] = 0)
3. RDYI# to the 82C465MV, which drives its RDY# signal to the CPU low on the same clock (bit ADh[0] = 1).

The co-processor asserts NPBUSY# while executing a floating-point calculation and asserts RDYI# to the 82C465MV when it is finished. If NPBUSY# is active and a co-processor error occurs (co-processor asserts FERR#), the 82C465MV latches NPBUSY# and generates IRQ13. Latched BUSY# and IRQ13 is cleared by a write command to I/O port 0F0h.

The RDYI# input is a strap-selectable option on the 82C465MV interface. Refer to the "Strap-Selected Interface Options" section for details on enabling the RDYI# input.

4.2.5.2 Programming

The RDYI# pin option must first be selected by strapping as noted above. Once RDYI# is available, it can function either as a direct input to the CPU RDY# signal or as a latched, delayed input, according to the setting of bit ADh[0]. Bit ADh[1] provides a means for overriding automatic co-processor detection if desired. The register at index ADh is described in the "Special CPU Interface Support" section.

4.2.6 Special CPU Interface Support

Certain CPUs operate internally with '486-type logic, yet present a '386 or '386/'486 hybrid signal interface. The IBM "Blue Lightning" processor uses this type of interface. The 82C465MV logic provides special features to handle mixed interfaces.

4.2.6.1 Ability to Cut CPU Power During Suspend

The 82C465MV will condition its outputs during suspend according to whether the connected CPU can be placed in a low-power mode during suspend, or can simply be powered down completely. The affected signals are listed in the "82C465MV Pin Signal Characteristics" section of this document. The 82C465MV reset logic generates a CPURST on exiting from suspend mode when this option is selected.

The option of complete CPU power-down on suspend is selected by writing bit ADh[5] = 1.

4.2.6.2 Programmable A20M# Functionality

Strapping the 82C465MV to provide a '386 interface automatically switches the function of pin 130 from A20M# to GA20. However, certain hybrid interface CPUs require a 386 interface with an A20M# input. The 82C465MV is programmable to force A20M# operation regardless of the interface selected. This option is enabled by writing bit ADh[4] = 1.

4.2.6.3 Programmable CPU RESET Functionality

Programmable reset functionality allows a CPU powered-down during suspend to restart operation on resume. The 82C465MV provides both soft and hard resets on SRESET whenever it is strapped for '386 interface operation. The RESET input to hybrid-interface CPUs must be connected to the 82C465MV SRESET signal for proper operation.

This reset option is enabled by writing bit ADh[3] = 1 before the first software-generated reset occurs. Operation is then as follows. The CPU has its power cut after entering suspend mode (using the PPWR0 power control latch signal to control the power MOSFET for the CPU). On resume, the PPWR0 line will turn on the CPU first. Then after the programmable delay associated with PPWR0, the SRESET line will be driven high to reset the CPU.

Note that this arrangement provides the VL bus with an independent reset signal, CPURST, so that VL-bus peripheral devices need not be inadvertently reset on a quick resume cycle.

The only important consideration in this type of arrangement is the software scheme. The SMM code that initiates the suspend operation must save the complete CPU context and be capable of restoring it without lapsing into a complete system reboot.

4.2.6.4 Programmable DACK2# Functionality

Strapping the 82C465MV to provide a '386 interface automatically switches the function of pin 80 from DACK2# to NPINT, requiring system designs to use DACK2# from the DACK decoder. However, since NPINT serves no purpose for most systems, many designers would prefer to keep the dedicated DACK2# signal. The 82C465MV is programmable to force DACK2# operation regardless of the interface selected. This option is enabled by writing bit ADh[2] = 1. Note that this bit setting is ignored if pin 80 is used for LGNT# (bit A0h[5] = 1); the LGNT# function always takes priority.

4.2.6.5 Cyrix Linear Burst Mode Support

Next-generation Cyrix M1sc CPUs provide a performance improvement through an optional "linear wrap mode" burst. The 82C465MVB part supports this feature through bit 3Fh[7]. Since the Cyrix CPU defaults to an Intel-compatible mode at power-up, the system can boot without problems.

Burst Mode Setting

Index	Name	7	6	5	4	3	2	1	0
3Fh	Misc Control Register	CPU Burst Mode 0=Intel 1=Cyrix linear (MVB)							

4.2.6.6 Programmable Exclusion of Co-processor Recognition

The 82C465MV automatically recognizes the presence of an external co-processor in a '386-type system by looking for an active FERR# line at reset time. If FERR# is sampled low at reset, any I/O cycle with A31 high automatically gets forwarded to the VL bus and the 82C465MV logic will not provide RDY#.

For special designs in which this arrangement conflicts with other devices mapped in high I/O space, the 82C465MV local-bus logic can be programmed to always pass these cycles through to the AT bus and generate RDY# to the CPU. This feature is enabled by writing bit ADh[1] = 1.

4.2.6.7 Programmable RDYI# Functionality

The 82C465MV normally takes the RDYI# input, synchronizes it through a flip-flop, and combines it with other sources to generate the open-collector RDY# signal to the CPU. This feature is provided for devices that cannot meet CPU RDY# setup times or

cannot tri-state their RDY# output after driving it inactive. The process effectively inserts a wait state and may reduce performance. Timing analysis indicates that the co-processor RDY# signal, for example, can be combined with CPU RDY# through a fast tri-state buffer and will meet the CPU RDY# setup requirement in many systems.

Therefore, the 82C465MV is programmable to inhibit RDYI# synchronization and simply pass RDYI# through a tri-state buffer to combine it with the existing RDY# output line to the CPU. This feature can be used in conjunction with an external co-processor that provides a direct-drive co-processor RDY# signal, such that when the 82C465MV recognizes a co-processor cycle, it will route the RDYI# signal directly to CPU RDY# and improve co-processor performance. This option is enabled by writing bit ADh[0] = 1. This bit has no function unless SBHE# is pulled low at system reset time to enable the RDYI# pin function.

Note that the co-processor RDYO# signal can also be tri-state buffered externally so that the RDYI#/CA25 input can be used in its CA25 capacity for 64MB on-board DRAM support.

Special CPU Feature Control Bits

Index	Name	7	6	5	4	3	2	1	0
ADh	Feature Control Register 3			CPU Power State in Suspend 0 = Powered 1 = 0 volt	Pin 130 in 386 mode 0 = GA20 1 = A20M#	SRESET Operation 0 = Normal 1 = Toggle on resume	Pin 80 in 386 mode 0 = NPINT 1 = DACK2#	Co-processor Recognition 0 = Enable 1 = Override	RDYI# Input 0 = Synchronized to RDY# 1 = Direct

4.3 System Functions

The 82C465MV handles clock generation, reset logic, and other basic system logic functions as described in the following sections.

4.3.1 Reset Logic

There are several signals involved in the 82C465MV system reset logic scheme.

4.3.1.1 RST1#

A low signal on input RST1# causes a hardware reset. The system must generate RST1# from the reset switch or from the power module signal representing "power good". This reset signal forces the system to begin execution in a known state. When RST1# is sensed active, the 82C465MV initiates a general reset cycle that lasts for 128 CPU clock cycles on all reset outputs.

4.3.1.2 RST4#

The RST4# output provides a peripheral reset signal that can be used to reset local devices directly, and can be inverted to provide RSTDRV to the AT bus.

4.3.1.3 CPURST and SRESET

The CPURST output provides a hard reset signal to the CPU, usually through its RESET input.

The SRESET output provides a soft reset signal to CPUs that can accommodate this function. Software generates reset by writing to the AT- or PS/2-compatible command ports. A soft reset is much faster (from the CPU perspective) than a hard reset. The SRESET signal duration is the same as that of CPURST. On the 82C465MV, SRESET also goes active during hardware resets in '386 interface mode.

Refer also to the "Special CPU Interface" section for information on how the CPURST and SRESET signals can be controlled during resume sequences.

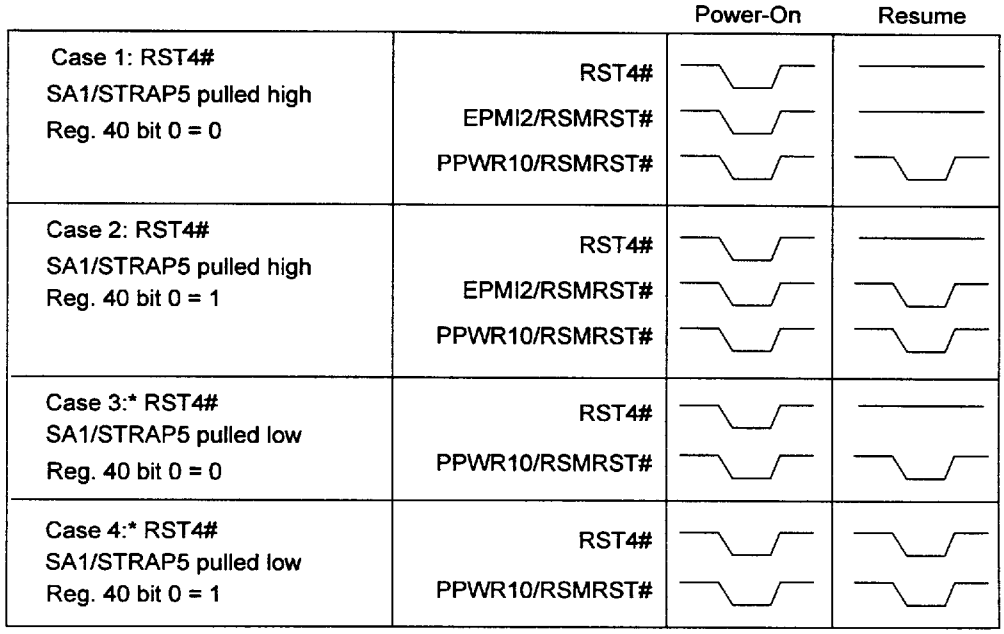
4.3.1.4 Resume Reset (RSMRST#) Function

The RSMRST# output supplies a reset signal only when resuming from suspend mode that is used to restart devices that were powered down on entering suspend mode. RSMRST# is provided on PPWR10 of the power control latch.

RSMRST# is optionally available on pin 185 (EPMI2) as well. Pin 148 strapping works in conjunction with bit 40h[0] to determine the reset lines that toggle upon resuming from suspend mode. Refer to the "Strap-Selected Interface Options" section for information on redefining pin 185 as RSMRST#. See Figure 4-1 for an illustration of the options.

Index	Name	7	6	5	4	3	2	1	0
40h	PMU Control Register 1								RSMRST# select 0 = Disable 1 = Enable See Figure 4-1

Figure 4-1 Resume Reset Function



* For cases 3 and 4, SA1/STRAP5 pulled low causes EPMI2/RSMRST# to have the function EPMI2 and is an input.

4.3.1.5 Rapid RESET Generation

The 82C465MV will monitor commands to I/O ports 060h and 064h, and intercept certain commands to port 060h, so that it can rapidly emulate the keyboard controller generation of the software reset signal. The decode sequence is software-transparent and requires no BIOS modifications to function.

The 82C465MV logic rapidly generates a fast CPU "warm reset" function when it detects a write of value FEh to port 064h, or a value of 1 to bit 0 of I/O port 092h.

NOTE Fast reset (ports 092h and 064h emulation) is generated on SRESET for the Intel SL Enhanced or the Cyrix Cx486S/S2 CPUs (MP0 and MP1 both strapped high) and on CPURST for other CPUs.

Port 092h is implemented in the logic according to the register layout shown.

System Control Port A (PS/2 Compatibility Port)

Port	Name	7	6	5	4	3	2	1	0
092h	System Control Port A	Don't care						Alternate Fast Gate A20 (r/w) 0 = No action 1 = Set Gate A20 active	Alternate Fast Reset (r/w) 0 = No reset 1 = Set reset active

Controlling Fast Reset

Index	Name	7	6	5	4	3	2	1	0
30h	Control Register 1							Fast Reset 0 = Wait for HLT 1 = Immed.	

Fast Reset Control (Port 092h or 064h) bit 30h[1] - Setting 30h[1] = 0 requires a Halt instruction before generation of CPURST (SRESET if Intel SL Enhanced CPU). Setting bit 30h[1] = 1 allows the reset to occur immediately without a Halt instruction.

4.3.1.6 Fast Reset Handling in SMM

In normal operating mode, fast reset I/O commands to the keyboard controller ports 060h and 064h are intercepted and handled by the 82C465MV logic. The keyboard control never receives KBDCS# and therefore does not know the fast reset status. In SMM, however, the 82C465MV logic does not inhibit KBDCS#. Therefore, a read of the keyboard controller ports to determine fast reset status will return the invalid status contained in the keyboard controller. Therefore, port 092h should always be used to determine the fast reset setting. Port 092h returns the current setting of both of the fast reset sources (port 092h and port 060/064h).

Note that generation of SRESET during SMM is prohibited on Intel and some other processors. Writing bit A0h[2] = 1 inhibits generation of SRESET during SMM.

Inhibition of SRESET in SMM

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1						SRESET Enable in SMM 0 = Enable 1 = Disable		

4.3.2 System Clock Generation

The 82C465MV chipset requires a minimum of three input clocks. These clocks are used to generate output clocks and to time internal operations as described below.

4.3.2.1 Input Clocks

OSCCLK. OSCCLK is the main input clock from which CPUCLK, FBCLKOUT, LCLK, and possibly ATCLK are derived. Its frequency is determined by the CPU as follows.

- If a 2X CPU is being used, OSCCLK is required to be 2X. Pin 3 (MP2) must be sensed high at reset (as described in the "Strap-Selected Interface Options" section) to indicate this condition. For example, a 33MHz 486DX, 2X clock CPU would require a 66MHz input on OSCCLK. The pin 160 strap option, which indicates whether the input clock is 1X or 2X, must indicate 2X clock (*not* pulled high) when pin 3 indicates a 2X CPU (pulled high).
- If a 1X CPU is being used, OSCCLK can be either a 1X or a 2X clock. Pin 3 must be sensed low at reset (pin 3 has an internal pull-down resistor at reset time so no external resistor is needed) to indicate a 1X CPU. With pin 3 low, OSCCLK can be 1X or 2X as selected by pin 160.

- If pin 160 is sensed high at reset time, OSCCLK is 1X. Pin 160 must be pulled up by a 10KΩ resistor if a 1X external clock is provided.
- If pin 160 is sensed low, OSCCLK is 2X. An *internal* pull-down resistor is engaged at reset time on pin 160 so no external strap is needed.

When a 1X CPU is used with a 2X input clock, the 2X clock is simply divided and the resulting 1X clock is passed on to the logic as if the 1X clock had been input directly. None of the internal 82C465MV logic uses the 2X input clock in this configuration. With a 1X CPU, a 1X input clock is recommended to save power.

OSC14. The OSC14 clock input (from a 14.31818MHz source) is used primarily for the system timer to retain AT compatibility. Most systems will also buffer the source of this clock and pass it on to the AT bus as the OSC14 signal to expansion devices. No synchronization to other system clocks is required.

The 82C465MV configuration bits 43h[3:0] can select the OSC14 signal (or other clocks) as the clock source for timing AT-bus cycles. The selection provides OSC14 divided by 2 for an effective AT clock rate of 7.2MHz as a close approximation of the 8MHz bus speed of the original AT bus.

The KBCLK and KBCLK2 signals are derived from OSC14. Their use is described in the section below.

ATCLKIN. The ATCLKIN option is selected through bit A0h[4]. If selected, the desired base clock is input on pin 172 (PIO2). This clock allows the AT bus clocking to be derived from any external input frequency. For example, if 8.0MHz AT bus operation is desired, ATCLKIN must be any multiple of 8MHz. 24MHz is often available and is commonly used for this function.

ATCLKIN Enabling

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1				Pin 172 0 = PIO2 (or CPUSPD) 1 = ATCLKIN				

SQWIN. The SQWIN signal should always be running, even in suspend mode. The clock input to SQWIN is used to:

- Time periodic DRAM refresh requests, both in active mode and, if enabled, in suspend mode
- Decrement the system activity counters (SQWIN is divided by program-selected values first)
- Control the sample rate of certain power management input pins
- To generate the KBCLK and KBCLK2 multiplexer clocks when no 14MHz source is available.

The SQWIN clock input can be 32KHz or 128KHz as selected by bit 40h[3].

4.3.2.2 Output Clocks

The 82C465MV chipset provides six clock outputs. It can control power consumption by adjusting many of these output clock frequencies dynamically as demand changes.

FBCLKOUT and CPUCLK. The clock input OSCCLK is used to generate signals FBCLKOUT and CPUCLK. CPUCLK is 2X for a 2X CPU and 1X for a 1X CPU. FBCLKOUT is always 1X.

CPUCLK may be divided internally using programmable divide rates to provide power control of the CPU and the 82C465MV chipset. Whenever CPUCLK is divided, FBCLKOUT is divided as well to maintain proper signal synchronization. The two clocks can be divided to achieve a slower full-speed clock and save system power. They can also be automatically slowed down (no software intervention required) during periods of low activity by the hardware doze mode feature described in this document.

- FBCLKOUT is the system clock and is fed back into pin FBCLKIN to provide a synchronized clock for most of the 82C465MV internal circuitry.
- CPUCLK is the clock to the CPU. In addition to being able to divide this clock, the logic can stop CPUCLK completely (known as the stop-clock feature) by writing a register bit. It will be restarted automatically when the 82C465MV logic detects an interrupt event such as IRQ, SMI, or EPMI. Note that FBCLKOUT is *not* stopped when CPUCLK is stopped.

Clock Phase. Both FBCLKOUT and CPUCLK must be kept in close phase alignment with each other; otherwise, the 82C465MV cycle controller may not sample the CPU control lines at the appropriate time. Clock phase alignment is affected primarily by board layout and CPU type. The provision for an external feedback clock (FBCLKOUT is fed back to FBCLKIN) allows any skew to be compensated for exact phase alignment. Usually, a series resistor or small inductor in series with either the CPU clock or the feedback clock is sufficient to realign the clocks. However, for certain CPUs and board layouts this correction is not sufficient and an external delay must be introduced.

The 82C465MV logic allows an internal gate delay to be programmed on either the CPU clock line or the feedback clock line through strapping options. This delay provides a coarse adjustment of clock skew; fine adjustment can then be completed using discrete components as described above.

- To delay FBCLKOUT by approximately 2ns:
  Pull pin 77 high at reset (can be strapped permanently with 10KΩ to AT bus Vcc)
- To delay CPUCLK by approximately 2ns:
  Pull pin 78 low at reset (can be strapped permanently with 10KΩ to ground).

Both straps can be used, but tend to cancel each other out. External resistors are not required to disable the delays, as internal resistors are activated on pins 77 and 78 at reset.

LCLK. LCLK can be used to provide a 1X clock for the local bus. Normally, FBCLKOUT is used to provide the local bus clock, but LCLK can be used if FBCLKOUT is too heavily loaded. LCLK is always a 1X clock, regardless of whether CPUCLK is 1X or 2X. However, the LCLK pin is redefined for L2 cache operation, in which case FBCLKOUT should be used.

ATCLK. The ATCLK is used for the AT bus clock. It is derived by dividing one of three sources: FBCLKIN, OSC14, or ATCLKIN. The source clock and divisor are selected through bits 43h[3:0]. If ATCLKIN is selected, a suitable clock must be provided on PIO2 and bit A0h[4] must be set to 1.

To avoid the incompatibilities introduced because OSCCLK can be either 1X or 2X, the clock divisors are based on double the FBCLKIN clock. For example, whether OSCCLK is a 2X 66MHz clock or a 1X 33MHz, FBCLKIN is 33MHz and the AT clock divisors are based on double that value. Therefore, selecting FBCLKIN/8 would result in an 8.33MHz AT clock in this case. This rule holds true in all cases except for the divide-by-1 setting, which can only select the actual OSCCLK clock input. For example, using a 2X 66MHz OSCCLK, the divide-by-1 AT clock is 66MHz; using a 1X 33MHz OSCCLK, the divide-by-1 AT clock is 33MHz.

AT Clock Rate Selection

Index	Name	7	6	5	4	3	2	1	0
43h	PMU Control Register 4					ATCLK generator source 0 = FBCLKIN 1 =ATCLKIN w/A0h[4] = 1	AT clock - Rate Selections 000 = /8 001 = /6 010 = /4 011 = /3	100 = 7.2 MHz 101 = /2 110 = /1 (/2 if 43h[3]=0) 111 = Stop	

Note to 82C463MV Programmers: PMU Control Register 4 (index 43h) changes slightly from its 82C463MV implementation, providing additional AT clock divisor selections and a new AT clock source option. These features are provided using reserved bits and selections, so that backward compatibility with 82C463MV software is maintained.

Table 4-1 indicates the appropriate ATCLK divisor settings for common input clock rates.

Table 4-1 Recommended Divisor Settings for Various Input Clock Frequencies

FBCLKIN frequency	Doubled frequency used for divisor calculation	/3	/4	/6	/8
12.5MHz	25MHz	8.1MHz			
16MHz	33MHz		8MHz		
20MHz	40MHz			6.6MHz	
25MHz	50MHz			8.3MHz	
33MHz	66MHz				8.25MHz
40MHz	80MHz				10MHz *
50MHz	100MHz				12.5MHz *

* Use 7.2MHz setting if these speeds are too fast for AT bus operations.

ATCLK can automatically be stopped if there is no AT bus activity through bits 5Eh[2:1].

AT Bus Clock Stretch Controls

Index	Name	7	6	5	4	3	2	1	0
5Eh	Clock Stretch Register						ATCLK when not in cycle 0 = Runs 1 = Stopped	AT clock stretch 0 = Async. 1 = Synch-ronous	

KBCLK and KBCLK2. The clock output KBCLK is provided for the keyboard controller. KBCLK is also used along with KBCLK2 (KBCLK divided by 2) to provide clocks for multiplexing interrupt and DMA requests. KBCLK is 7.2MHz, and KBCLK2 is 3.6MHz. When connected as the select inputs to 74153 multiplexers, they select each of the four multiplexed inputs sequentially once every 280ns.

4.3.3 A20M# Generation

The 82C465MV logic provides both AT-compatible and PS/2-compatible means of controlling the A20M# signal to the CPU.

- A20M# goes inactive (high) if either the AT-compatible port control is set to 1 (by writing D1h to port 064h followed by setting port 060h bit [1] = 1) or the PS/2-compatible port control is set to 1 (by setting port 092h bit [1] = 1).
- A20M# goes active if both the AT-compatible port control *and* the PS/2-compatible port control are cleared to 0.

The A20M# port control of port 092h is shown below.

System Control Port A (PS/2 Compatibility Port)

Port	Name	7	6	5	4	3	2	1	0
092h	System Control Port A	Don't care						Alternate Fast Gate A20 (r/w) 0 = No action 1 = Set Gate A20 active	Alternate Fast Reset (r/w) 0 = No reset 1 = Set reset active

A write to port 064h with data D0h will enable the status of GATEA20 (bit 1 of port 060h) and the system reset control (bit 0 of port 060h) to be readable in normal mode.

4.3.3.1 Rapid A20M# Generation

The 82C465MV will monitor commands to I/O ports 060h and 064h, and intercept certain commands to port 060h, so that it can rapidly emulate the keyboard controller generation of the A20M# signal. The decode sequence is software-transparent and requires no BIOS modifications to function.

The fast A20M# generation sequence occurs when the CPU writes the value D1h to port 064h, followed by writing the value 02h to port 060h. The logic inhibits KBDCS# on both the port 064h access and on the following port 060h access. Therefore, the 82C465MV logic can quickly switch its own A20M# line without waiting for the keyboard controller to do so. The I/O write command and output data still go to the SD and XD buses in both cases.

4.3.3.2 Inhibition of Fast A20M# and Fast Reset Generation

Due to a patent recently awarded against internal duplication of external keyboard controller functionality, system designers may prefer not to use the 82C465MV fast reset and fast A20M# generation features. Therefore, the 82C465MVA part provides the means to partially or completely disable this mechanism. Whether a system designer decides to use all, part, or none of the internal fast generation logic depends on his interpretation of the patent with regard to the intended system implementation.

The 82C465MV part blocks KBDCS# generation when it detects certain data write sequences to ports 060h and 064h, and generates the A20M# output or SRESET output as appropriate. The 82C465MVA logic can inhibit this fast generation logic at three different levels.

1. The logic permits KBDCS# to pass through regardless of whether it detects port 060/064h accesses. The chip continues to monitor port 060/064h accesses and generate A20M# and SRESET.
2. The logic permits KBDCS# to pass through as above, but does ***not*** monitor port 060h/064h accesses to generate A20M# and SRESET. Internal port 092h accesses can still generate A20M# and SRESET. The signals must be combined externally with the signals generated by the keyboard controller.
3. The logic operates as in item 2 above. In addition, pin 179 is redefined as the KBCRSTIN input for the reset signal output from the keyboard controller. The logic combines this control with the port 092h control as sources for SRESET. Pin 179 is normally used as CHCK#, the ISA bus NMI source; this option is lost when the pin is used as KBCRSTIN.

The register bits required for these options are shown below, along with a new shadow register for port 064h writes.

Fast Signal Generation Control Bits

Index	Name	7	6	5	4	3	2	1	0
79h	PMU Control Register 11						"Fast" Logic Functionality Level 00=Fully functional 01=Do not inhibit KBDCS# 10=Also. Disable reset from 060/064h 11=Also: Redefine pin 179 as KBCRSTIN (MVA)		
9Fh (MVA)	Port 064h Shadow Register	Shadows I/O writes to port 064h bits [7:0], regardless of whether KBDCS# is inhibited In this way, when an SMI occurs between a port 064h write and the subsequent write to port 060h, SMM code can access the keyboard controller as needed and then simply restore the port 064h value just before leaving SMM.							

4.3.3.3 A20M# Handling in SMM

On entry to SMM, the A20M# signal is forced high. The signal returns to its prior value on return to normal mode. Writes to either A20M# control port are blocked while in SMM.

Port 092h can be read at any time, including SMM. However, the GATEA20 setting cannot be read directly from ports 060/064h while in SMM. Therefore, a read-only bit is provided at A1h[7] to return the A20M# setting at any time. Bit A1h[7] provides an indication of the Gate A20 setting last made through the normal write sequence at I/O port 060/064h.

Index	Name	7	6	5	4	3	2	1	0
A1h	Feature Control Register 2	Port 060/4, Gate A20, Read-only.							

4.3.3.4 Port 060/064h A20 Setting Accessibility

On the 82C465MV and 82C465MVA parts, the port 060/064h A20M# setting can be read in SMM through bit A1h[7]. This feature is important to zero-volt suspend (suspend to disk) because SMM code must know the setting of A20M# in order to restore the value after resuming from a zero-volt suspend.

The complementary feature, however, is missing from the MV and MVA parts: There is no way to restore the port 060/064h bit setting from within SMM. Therefore, resume code currently has to restore the A20M# setting from outside SMM, which is not very clean.

The 82C465MVB redefines bit A1h[7] to be writable as well as readable. When written as 0, bit A1h[7] has no effect. When written as 1, bit A1h[7] toggles the current setting of the port 060/064h A20M# bit. If SMM code needs to restore A20M# to its original setting, it simply reads bit A1h[7] to determine the current setting, then writes back to register A1h with bit 7 set to 1 if it is necessary to toggle the bit. Software should then re-read A1h[7] to ensure that the bit has been restored to the desired value.

A20M# Setting within SMM

Index	Name	7	6	5	4	3	2	1	0
A1h	Feature Control Register 2	Port 060/4 A20M# bit Read: return current value; Write: toggle A20M# setting							

4.4 DRAM Controller

The 82C465MV uses an advanced memory controller to generate cycles to five banks of symmetrical or asymmetrical DRAM, manage the on-CPU write-back or write-through cache (L1), and manage an external write-back cache (L2).

During DRAM read or write cycles, a row address and a column address is required. In most DRAMs, the row address access time is longer than the column address access time. Therefore, bus performance can be improved by using page-mode DRAM operation. Memory locations sharing the same row address are on the same page; therefore, only a new column address is required. In page-mode operation, the row address strobe, RAS# can be kept active and only a new CAS# needs to be generated, thus reducing memory cycle time.

In a five-bank configuration, a maximum of five pages of memory can be kept active at the same time, since each bank has an independent RAS#. The effectiveness of page-mode operation depends heavily on the page size. A larger page size increases the chances of a page hit.

4.4.1 DRAM Controller Hardware Options

The standard DRAM controller hardware includes direct control of five banks of DRAM (RAS0#-RAS3#) and up to 12 bits of DRAM addressing (MA[11:0]). Through relocation of certain signals, certain of these features can be deleted to avoid using external TTL for external power management input signals. Refer to the "Interface Strapping Options" section of this document for information on the memory interface selection options.

The DRAM controller can also change its decoding according to the symmetry of the DRAM devices in use. The mapping of the CPU address (CA) signals to the memory address (MA) signals is controlled by the following inputs.

- Bank capacity, as set by bits [2:0] and [6:4] of registers A8h, A9h, and AAh
- Symmetrical or asymmetrical DRAM selection, as set by bits [3] and [7] of registers A8h, A9h, and AAh (ignored for 32MB and 64MB banks)
- Asymmetry selection, as set by bits [4:0] of register D3h (ignored for 32MB and 64MB banks).

The decoding is shown in Table 4-2, Table 4-3, and Table 4-4.

Table 4-2 Symmetrical DRAM Address Decoding

Memory Address	1MB		2MB		4MB		8MB		16MB		32MB		64MB	
	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
MA0	A2	A11	A2	A11	A2	A11	A2	A11	A2	A11	A2	A11	A2	A11
MA1	A3	A12	A3	A12	A3	A12	A3	A12	A3	A12	A3	A12	A3	A12
MA2	A4	A13	A4	A13	A4	A13	A4	A13	A4	A13	A4	A13	A4	A13
MA3	A5	A14	A5	A14	A5	A14	A5	A14	A5	A14	A5	A14	A5	A14
MA4	A6	A15	A6	A15	A6	A15	A6	A15	A6	A15	A6	A15	A6	A15
MA5	A7	A16	A7	A16	A7	A16	A7	A16	A7	A16	A7	A16	A7	A16
MA6	A8	A17	A8	A17	A8	A17	A8	A17	A8	A17	A8	A17	A8	A17
MA7	A9	A18	A9	A18	A9	A18	A9	A18	A9	A18	A9	A18	A9	A18
MA8	A10	A19	A10	A19	A10	A19	A10	A19	A10	A19	A10	A19	A10	A19
MA9	A21	A20	A21	A20	A21	A20	A21	A20	A21	A20	A21	A20	A21	A20
MA10	A23	A22	A23	A22	A23	A22	A23	A22	A23	A22	A23	A22	A23	A22
MA11	A25	A24	A25	A24	A25	A24	A25	A24	A25	A24	A25	A24	A25	A24

Table 4-3 Asymmetrical DRAM Decoding, Asymmetry bit D3h[4:0] = 0

Memory Address	1MB, 10x8		2MB, 11x8		4MB, 11x9		8MB, 12x9		16MB, 12x10	
	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
MA0	A2	A11	A2	A11	A2	A11	A2	A11	A2	A11
MA1	A3	A12	A3	A12	A3	A12	A3	A12	A3	A12
MA2	A4	A13	A4	A13	A4	A13	A4	A13	A4	A13
MA3	A5	A14	A5	A14	A5	A14	A5	A14	A5	A14
MA4	A6	A15	A6	A15	A6	A15	A6	A15	A6	A15
MA5	A7	A16	A7	A16	A7	A16	A7	A16	A7	A16
MA6	A8	A17	A8	A17	A8	A17	A8	A17	A8	A17
MA7	A9	A18	A9	A18	A9	A18	A9	A18	A9	A18
MA8	A10	A19	A10	A19	A10	A19	A10	A19	A10	A19
MA9	A21	A10	A21	A20	A21	A20	A21	A20	A21	A20
MA10	A23	A22	A23	A10	A23	A21	A23	A22	A23	A22
MA11	A25	A24	A25	A24	A25	A24	A25	A21	A25	A23

Table 4-4 Asymmetrical DRAM Decoding, Asymmetry bit D3h[4:0] = 1

Memory Address	1MB, 11x7		2MB, 12x7		4MB, 12x8		8MB, 12x9		16MB, 12x10	
	Col	Row	Col	Row	Col	Row	Col	Row	Col	Row
MA0	A2	A11	A2	A11	A2	A11	A2	A11	A2	A11
MA1	A3	A12	A3	A12	A3	A12	A3	A12	A3	A12
MA2	A4	A13	A4	A13	A4	A13	A4	A13	A4	A13
MA3	A5	A14	A5	A14	A5	A14	A5	A14	A5	A14
MA4	A6	A15	A6	A15	A6	A15	A6	A15	A6	A15
MA5	A7	A16	A7	A16	A7	A16	A7	A16	A7	A16
MA6	A8	A17	A8	A17	A8	A17	A8	A17	A8	A17
MA7	A9	A18	A9	A18	A9	A18	A9	A18	A9	A18
MA8	A10	A19	A10	A19	A10	A19	A10	A19	A10	A19
MA9	A21	A10	A21	A20	A21	A20	A21	A20	A21	A20
MA10	A23	A9	A23	A10	A23	A21	A23	A22	A23	A22
MA11	A25	A24	A25	A9	A25	A10	A25	A21	A25	A23

NOTE The shaded address lines should not be used for asymmetrical DRAM decoding, however, they are still output during the memory cycle.

4.4.2 DRAM Bus Drive Capability

By setting bit A1h[4] = 1, the drive capability of certain MA bus signals and DRAM control signals is increased by approximately 50%. Refer to the "AC Characteristics" section of this document for determining whether this additional drive, at the rated capacitance and speed of the DRAM load, will eliminate the need for extra bus buffers on the memory address lines.

Index	Name	7	6	5	4	3	2	1	0
A1h	Feature Control Register 2				Heavy-duty Memory Bus Drive 0 = Disable 1 = Enable				

4.4.3 Setting Up DRAM Operation

Programming the 82C465MV DRAM controller is a simple matter of:

1. Setting bit A0h[0] = 1 to enable simplified memory programming
2. Setting bit 31h[5] according to whether or not parity bit DRAM will be used (not an available option if L2 cache is part of design)
3. Writing the size and arrangement of each bank to its corresponding register at indexes A8-AAh
4. Setting the asymmetry selection for that bank in the register at index D3h if the DRAM is not symmetrical
5. Setting the cycle speed in the register at index 35h
6. Selecting the desired refresh rate through bits 67h[6:5]
7. Enabling DRAM refresh through bit 57h[7].

The register bits involved in setting up DRAM operation are shown below.

DRAM Setup Registers

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1								DRAM Mapping Enable 0 = Disable 1 = Enable
31h	Control Register 2			Parity check 0 = Enable 1 = Disable					
A8h	DRAM Bank Select Register 1	Bank 1 type 0 = Symmetrical 1 = Asymmetrical	Bank 1 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB			Bank 0 type 0 = Symmetrical 1 = Asymmetrical	Bank 0 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
A9h	DRAM Bank Select Register 2	Bank 3 type 0 = Symmetrical 1 = Asymmetrical	Bank 3 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100= 8MB 101 = 16MB 110 = 32MB 111 = 64MB			Bank 2 type 0 = Symmetrical 1 = Asymmetrical	Bank 2 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
AAh	DRAM Bank Select Register 3					Bank 4 type 0 = Symmetrical 1 = Asymmetrical	Bank 4 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
D3h	Asym. DRAM Select Register				Bank 4 asym. type 0 = 11x9 1 = 12x8	Bank 3 asym. type 0 = 11x9 1 = 12x8	Bank 2 asym. type 0 = 11x9 1 = 12x8	Bank 1 asym. type 0 = 11x9 1 = 12x8	Bank 0 asym. type 0 = 11x9 1 = 12x8

DRAM Setup Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
35h	DRAM Control Register 2	DRAM read wait states 00=3-2-2-2 01=4-3-3-3, 1ws pg. miss 10=4-3-3-3, 0 ws pg. miss 11=5-4-4-4		DRAM write wait states 00 = no wait states 01 = 1 wait state 10 = 1 wait state 11 = no wait states, RAS# 1/2 clock early (MVB)					
67h	PMU Control Register 9		Refresh rate Active or suspend mode '00' = 15us (30us in suspend if A1h[6] = 0) '01' = 30us '10' = 61us '11' = 122us						
57h	PMU Control Register 6	Refresh enable 0 = Disable 1 = Enable							

Parity Checking is always disabled if L2 cache option is enabled.

4.4.3.1 Faster Memory Cycles

On the 82C465MV and 82C465MVA chipsets, 3-2-2-2 burst reads with 0 wait state writes are possible only with 70ns or better DRAM. On the 82C465MVB part, RAS# can be programmed to come one-half clock early on page miss cycles to give a wider timing margin and permits the use of 80ns DRAM in this application. Bits 35h[5:4]=11, formerly a reserved combination, select this cycle.

DRAM Early RAS# Control

Index	Name	7	6	5	4	3	2	1	0
35h	DRAM Control Register 2			DRAM write wait states 00 = no wait states 01 = 1 wait state 10 = 1 wait state 11 = no wait states, RAS# 1/2 clock early					
3Fh	Misc. Control Register					Minimum Wait States for non L2 cache systems 0=1ws 1=0ws (MVB)			

4.4.3.2 DRAM Mapping Scheme Enable

Setting A0h[0] = 0 provides compatibility with BIOS code written for the 82C463MV chipset. The proper method for new 82C465MV designs is to set up the DRAM through indexes A8h, A9h, and D3h, then set bit A0h[0] = 1 before attempting to access DRAM.

4.4.3.3 DRAM Control Register II - Index 35h

DRAM Control Register 2 at index 35h is not strictly backward compatible with the 82C463MV chipset register due to changes in the memory controller timing selections. However, the 82C465MV memory controller will not fail if 82C463MV programming is used; only slower operation will occur. Refer to the 82C463MV Data Book for comparison of the index 35h bit select functions.

4.4.4 EDO DRAM Support

The 82C465MVB provides a dramatic performance improvement with the incorporation of EDO DRAM support in its memory controller. EDO DRAM latches its output data while its input address changes in order to save an additional clock on most memory read cycles. The performance of a system based on EDO DRAM is nearly as high as a system with L2 cache.

EDO DRAM requires special control of the DRAM WE# pin to extend the data output duration. However, it requires no additional pins. Bits 3Eh[4:0] enable EDO DRAM support separately for each bank as indicated below. Bits 3Eh[7:6] select the read wait state timing for EDO banks. Bits 35h[7:6] apply only to Standard DRAM.

EDO DRAM Selection

Index	Name	7	6	5	4	3	2	1	0
3Eh	DRAM Type Select Register	EDO DRAM read wait states 00=3-1-1-1 01=3-2-2-2 10=4-2-2-2 11=reserved (MVB)			Bank 4 DRAM 0=Standard 1=EDO (MVB)	Bank 3 DRAM 0=Standard 1=EDO (MVB)	Bank 2 DRAM 0=Standard 1=EDO (MVB)	Bank 1 DRAM 0=Standard 1=EDO (MVB)	Bank 0 DRAM 0=Standard 1=EDO (MVB)
35h	DRAM Control Register 2	Standard DRAM read wait states 00=3-2-2-2 01=4-3-3-3, 1ws pg miss 10=4-3-3-3, 0 ws pg miss 11=5-4-4-4							

4.4.5 DRAM Cycle Speed

The values typically used for DRAM cycle speed are shown in Table 4-5.

Table 4-5 Suggested DRAM Cycle Speed Settings

1X CPU Frequency	DRAM Speed	Read Cycle Timing	Write Cycle Timing
20MHz	80ns	3-2-2-2	0 ws
25MHz	80ns	3-2-2-2	0 ws
33MHz	70ns	3-2-2-2	1 ws
40MHz	70ns	4-3-3-3	1 ws
50MHz	70ns	5-4-4-4	1 ws

4.4.6 System ROM and Shadow RAM

Since accesses to local DRAM are much faster than those to ROM, the 82C465MV provides shadow RAM capability. With this feature, code from slow devices can be copied to local DRAM for faster access. All accesses to the specified EPROM space are redirected to the corresponding DRAM location.

The 82C465MV supports up to 256K bytes of ROM in the first 1MB of address space. The F000h segment is handled as one continuous 64K block, while the C000h, D000h and E000h segments each are divided into four 16K blocks that can be individually controlled. Segments C000h, E000h and F000h can be shadowed, cached, or both; segment D000h can be shadowed but not cached.

The procedure for configuring shadow RAM operating and loading the shadow RAM is as follows.

1. Select the blocks in the C00000-FFFFFh range whose access should generate ROMCS#. The ROM Select Registers shown below list the possible blocks. ROMCS# generation implies reads from the XD bus, so the XDIR signal will direct the ROM data onto the SD bus. If ROM is on the AT bus (such as for a video adapter), the corresponding ROMCS bit for that block should not be set.

ROM Select Registers

Index	Name	7	6	5	4	3	2	1	0
38h	Block Control Register 1				ROMCS for CC000	ROMCS for C8000	ROMCS for C4000	ROMCS for C0000	
37h	D/E000 Control Register	ROMCS for DC000	ROMCS for D8000	ROMCS for D4000	ROMCS for D0000				

ROM Select Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
31h	Control Register 2					ROMCS for EC000	ROMCS for E8000	ROMCS for E4000	ROMCS for E0000

NOTE See Table 4-6 for all bit settings in these registers

2. Globally enable loading of the shadow DRAM. The Write Destination Register below shows the control available: bit 36h[7] to enable writes to DRAM for all blocks C000-F000h, and bit 36h[6] for writes to DRAM in the C000h, D000h, and E000h blocks. Bit 36h[7] must be set to 1 before bit 36h[6] can be set to 1.

Write Destination Registers

Index	Name	7	6	5	4	3	2	1	0
36h	Shadow RAM Control Register 3	F000 write select dest. 0 = DRAM 1 = ROM don't care for F000 if 32h[7] = 0	C-D-E000 select dest. 0 = AT/ROM 1 = DRAM See Table 4-6						

Register bits 38h[4:1], 37h[7:4], and 31h[3:0] are set according to a common scheme, depending on the settings of bits 36h[7:6], as shown in Table 4-6.

Table 4-6 Access Control Bit Meanings for bits 38h[4:1], 37h[7:4], and 31h[3:0]

Bits 36h[7:6] Setting	Access Control Bit Selection
00	'0' = R/W from AT-Bus '1' = Read from ROMCS#; writes disabled
x1	'0' = Read from AT-Bus if not shadowed, write to DRAM '1' = Read from ROMCS# if not shadowed, write to DRAM See bits 33h[7:0] and 36h[3:0] for shadowing selection
10	'0' = R/W from AT-Bus '1' = R/W from ROMCS#

3. Copy the information to be shadowed from ROM to DRAM by simply reading from and then writing back to the same location for each byte of the block to be copied. The ROM need not be located in the physical BIOS ROM; it can also be ROM on the AT bus as selected in step 1.
4. Again using bits 36h[7:6], globally disable writes to the shadow DRAM.
5. Select the ROM blocks that have been shadowed in DRAM. The register bits shown below are used to select the blocks that have been shadowed. Enabling F000h reads to come from DRAM (bit 32h[7]) also automatically enables write protection for that DRAM block.

Shadow RAM Control Bits

Index	Name	7	6	5	4	3	2	1	0
33h	Shadow RAM Control Register 2	EC000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	E8000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	E4000 read select ROM/ RAM 0 = ROM 1 = Sh RAM	E0000 read select ROM/ RAM 0 = ROM 1 = Sh RAM	DC000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	D8000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	D4000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	D0000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM

Shadow RAM Control Bits (cont.)

Index	Name	7	6	5	4	3	2	1	0
36h	Shadow RAM Control Register 3					CC000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	C8000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	C4000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM	C0000 read select ROM/ RAM 0 = ROM 1 = Sh. RAM
32h	Shadow RAM Control Register 1	F0000 access 0 = DRAM 1 = ROM							

F0000h-FFFFFh access control bit 32h[7] - This bit serves a dual purpose. Setting bit 32h[7] = 0 allows reading from DRAM and write protect (enable shadowing) for the F000h block. Setting bit 32h[7] = 1 allows reading from ROMCS#, and writing to ROMCS# (if bit 36h[7] = 1) or to DRAM (if bit 36h[7] = 0)

6. Select the shadowed blocks that should be write protected. The bits used to select the blocks to be write-protected are shown below. While F000h was automatically write-protected in the previous step, it can be unprotected (for test purposes only) through bit A1h[1].

   The 82C465MV logic provides special handling for write-protected DRAM areas (ROM shadowed in RAM). Normally, shadow RAM selected as cacheable makes the RAM cacheable in both L1 and L2 cache. However, selecting shadow RAM areas as write-protected makes them cacheable only in L2 cache (if present). This provision prevents loss of cache coherency between L1 cache and external RAM (if a program were to try to write to write-protected memory that is cached in L1 cache, for example).

Write Protect Registers

Index	Name	7	6	5	4	3	2	1	0
32h	Shadow RAM Control Register 1				D000 block shadow control 0 = Wntable 1 = Protected	E000 block shadow control 0 = Writable 1 = Protected			
36h	Shadow RAM Control Register 3			C000 write protect 0 = Writable 1 = Protected					
A1h	Feature Control Register 2							F000 shadow test 0 = Read or write 1 = Read and write	

7. Read accesses to BIOS and other ROM code that has been shadowed will now come from DRAM. Write accesses will be either blocked if write protection has been engaged, or will be directed to the AT bus or to EPROM otherwise.

4.5 Cache Control

The 82C465MV manages several levels of cache: L1 write-through, L1 write-back (if supported by the CPU), and L2 write-back.

4.5.1 Global Enabling of Cacheability

The various levels of cache control must first be enabled individually. Then, setting bit 35h[1] = 0 globally enables all individually enabled cache control features.

Global Cache Control Enable

Index	Name	7	6	5	4	3	2	1	0
35h	DRAM Control Register 2							Global caching control 0 = Enable 1 = Disable	

4.5.2 Defining Non Cacheable Blocks

Registers 38-3Bh are used to define two non cacheable blocks. The starting address for these blocks must have the same granularity as the block size. For example, if a 512K-byte non cacheable block is selected, its starting address is a multiple of 512K bytes; consequently, only address bits of A[23:19] are significant and A[18:16] are "don't cares". Table 4-7 shows the valid starting address bits for all block sizes.

Table 4-7 Size and Valid Start Address Bits of Non Cacheable Memory Blocks

Bits 38h, 3Ah [7 6 5]	Block Size	Valid Starting Address Bits								
		A24	A23	A22	A21	A20	A19	A18	A17	A16
0 0 0	64KB	V	V	V	V	V	V	V	V	V
0 0 1	128KB	V	V	V	V	V	V	V	V	x
0 1 0	256KB	V	V	V	V	V	V	V	x	x
0 1 1	1MB	V	V	V	V	V	x	x	x	x
1 x x	Disabled									

Note: V = Valid Bit; x = "don't care"

The two non cacheable blocks are defined through the registers shown below.

Non Cacheable Block Registers

Index	Name	7	6	5	4	3	2	1	0
38h	Block Control Register 1	Non cacheable block 1 (ncb1) size, See Table 4-7							ncb1 A24
39h	Block Control Register 2	Non cacheable block 1 start address A[23:16]							
3Ah	Block Control Register 3	Non cacheable block 2 (ncb2) size, See Table 4-7							ncb2 A24
3Bh	Block Control Register 4	Non cacheable block 2 start address A[23:16]							

4.5.2.1 C000, E000, F000h Block Cache Enable

Certain blocks in the option ROM range C0000-FFFFFh can be made cacheable. The cacheability is effective only if the ROM in that range has been shadowed in DRAM.

- Bit 35h[0] determines whether the 32KB block from C0000 to C7FFFh will be cacheable.
- Bit 35h[2] determines whether the 64KB block from F0000 to FFFFFh will be cacheable.
- Bits 37h[3:0] determine the 16KB blocks from E0000 to EFFFFh that will be cacheable.

C000, E000, F000h Block Cache Enable

Index	Name	7	6	5	4	3	2	1	0
37h	D/E000 Control Register					EC00 16KB block cacheable 0 = yes 1 = no	E800 16KB block cacheable 0 = yes 1 = no	E400 16KB block cacheable 0 = yes 1 = no	E000 16KB block cacheable 0 = yes 1 = no
35h	DRAM Control Register 2						F000 64KB block cacheable 0 = yes 1 = no		C000 32KB block cacheable 0 = yes 1 = no

4.5.2.2 Cache Control of C000-F000h

The cache controller of the 82C465MVA chip can be used with either write-through or write-back CPUs, and with or without an external cache. The 82C465MV Data Book outlines the programming registers associated with L1 and L2 cache, but does not go into detail about their operation with regards to cacheable areas C000-F000h. This application note describes the operation of L2 cache in this region in greater detail.

4.5.2.2.1 Introduction

Cacheability of C0000-FFFFFh starts out very simply. First of all, any ROM in these regions must be shadowed in DRAM; otherwise the area cannot be cached at either L1 or L2. Second, Bit 35h[1] must be set to 0 to globally enable caching.

4.5.2.2.2 L1 Cache Control

L1 cacheability is interlinked with the 82C463MV-compatible memory mapping mode of the 82C465MVA chip. Bit A0h[0] defaults to 0, selecting 82C463MV compatibility at reset. In this mode, cacheability is available as follows.

- C0000h 32KB block: Cacheable in L1 when bit 35h[0]=0.
- C8000h 32KB and D0000h 64KB blocks: Never cacheable.
- E0000, E4000, E8000, and EC000h 16KB blocks: Cacheable in L1 when respective bits 37h[0-3]=0.
- F0000 64KB block: Cacheable in L1 when bit 35h[2]=0.

This programming scheme on the 82C463MV was not very secure, in that a shadowed region of ROM could be made cacheable in L1 yet write-protected in DRAM. In the case where an application program writes to the ROM region (as Windows does, for example), the content of L1 cache could be changed without updating the memory region in DRAM (which the 82C463MV would write-protect). This situation would become even more dangerous in the case of a system with L2 cache, since the DRAM, L2 cache, and L1 cache could all contain different values for the same memory space.

Therefore, the 82C465 series chip introduced greater protection against this type of programming when not in 82C463MV-compatible mode. When bit A0h[0] is set to 1, the 82C465MVA operates according to the new memory control scheme. This setting automatically introduces an additional layer of cacheability control, requiring that the region also be declared read/writeable before it becomes cacheable.

- C0000h 32KB block: Cacheable when bit 35h[0]=0 *and* bit 36h[5]=0 (C000h writeable).
- E0000, E4000, E8000, and EC000h 16KB blocks: Cacheable when bits 37h[0-3]=0 *and* 32h[3]=0 (E000h writeable).
- F0000 64KB block: Cacheable when bits 35h[2]=0 *and* A1h[1]=1 (F000h writeable).

This interlock ensures that the CPU cache contents will be coherent with the DRAM contents (or at least the L2 cache contents) at all times.

4.5.2.2.3 L2 Cache Control

L2 cacheability is not affected by the selection between 82C463MV-compatible memory mapping operation and 82C465MVA new memory mapping. L2 cacheability control works essentially the same for all regions.

- C0000, C4000, C8000, and CC000h 16KB blocks: Cacheable in L2 when respective bits D2h[0-3]=1, along with bit 36h[5]=0 or on any read cycle.
- D0000, D4000, D8000, and DC000h 16KB blocks: Cacheable in L2 when respective bits D2h[4-7]=1, along with bit 32h[4]=0 or on any read cycle.
- E0000, E4000, E8000, and EC000h 16KB blocks: Cacheable in L2 when respective bits D1h[0-3]=1, along with bit 32h[3]=0 or on any read cycle.
- F0000h 64KB block: Cacheable in L2 when bit 35h[2]=0, along with bit A1h[1]=1 or on any read cycle.

Note that, as with 82C465MVA L1 cache control, writes to a region are only cacheable if the region is not write-protected. However, unlike L1 cache, reads are cacheable in L2 regardless of the write-protect state of the region.

Example

Here is a practical example of the cacheability controls for a system with L1 writeback and L2 cache. The system is programmed for 82C465MV memory mapping so that the L1 cacheability interlock is in place.

1. The setup code shadows ROM code in the E0000-E7FFFh region, and sets bit 32h[3]=1 to write-protect the region. It also sets bits D1h[0-1]=11 to make the region cacheable in L2, and bit 37h[0]=0 to make the region cacheable in L1.
2. Application code reads the 1KB block starting from E0000h. The block is write-protected, so it will not be cached in L1. However, it will be cached in L2.
3. The application modifies the 1KB block and writes it back to E0000h. Since the block was never cached in L1, there is no effect for the CPU. Since the block is write-protected, the 82C465MVA prevents updating of both the L2 cache and the system DRAM. So all memory is still coherent.

If the system had been programmed for 82C463MV-compatible memory mapping mode, the write-protect status of the block would not have mattered. When the CPU read the block, it would have cached it in L1. When the CPU write the block, it would have updated the block in L1, but the L2 cache and the DRAM would have been write-protected. Therefore, system memory would no longer be coherent. If the CPU accesses data in that region again, it will not be the same data that an external master would access (even if the CPU performed a write-back cycle).

4.5.2.2.4 Verifying L2 Cache Operation

The following routine may prove useful in testing whether L2 cache is being read and written properly in an 82C465MV-based system. The procedure can be run from DEBUG commands if desired. Any errors indicate possible timing problems or RAM failures.

1. Program segments 0:0h through 3000:0h to be non cacheable. Using non cacheable block 2, this would involve setting bits 3Ah[7:5]=010 (256KB block) and bits 3Ah[0]+3Bh[7:0]=0 (start address 0:0). In this way, the debug program and all DOS routines in low memory will not involve cache.
2. Enter debug command `F 3000:0LFFFF 00 01 02 ... 0F` to fill all of segment 3000h with a repeating pattern.
3. Enter commands:

```
M 4000:0LFFFF 8000:0
M 5000:0LFFFF 8000:0
M 6000:0LFFFF 8000:0
M 7000:0LFFFF 8000:0
```

   to ensure that 9000:0 is *not* in L2 cache.

4. Enter `M 3000:0LFFFF 9000:0` to copy the pattern in segment 3000h to 9000h. L2 is unaffected.

5. Enter `M 9000:0LFFFF 8000:0` to copy the pattern into L2 as it is being read from segment 9000h.
6. Enter `C 9000:0LFFFF 3000:0` to compare the data written to L2 to the original data pattern in DRAM. Any mismatches indicate a failure.

From this point, there are two branches to the test. Taking Branch (a) will test whether DRAM is corrupted by new writes to L2 cache. Taking Branch (b) will test whether "dirty" data in L2 cache is getting written back to DRAM properly. To implement the test completely, it should go from step 1 through 9a the first time, and step 1 through 10b the second time.

Branch (A)

7. Enter `F 9000:0LFFFF 00` to fill L2 with data that is different from the original pattern.
8. Disable L2 cache by setting bit D0h[5]=0.
9. Enter `C 9000:0LFFFF 3000:0`. The comparison will take place against data in DRAM only, to determine whether it has been corrupted. Any mismatches indicate a failure.

Branch (B)

7. Enter `F 9000:0LFFFF 20 21 22 ... 2F` to fill segment 9000 with a new pattern.
8. Enter `M 5000:0LFFFF 8000:0` to force segment 9000 to be written back to DRAM (when the 5000 segment is read in, it has to occupy the spot where the segment 9000 data resides).
9. Enter `F 3000:0LFFFF 20 21 22 ... 2F` to fill segment 3000 with the same pattern written to segment 9000.
10. Enter `C 3000:0LFFFF 9000:0` to see whether the data forced out of L2 cache to segment 9000 in DRAM is still the same pattern originally entered. Any mismatches indicate a failure.

4.5.2.3 Cache Invalidation Feature

Caching of write-protected DRAM in L1 cache is automatically prevented on the 82C465MVA part, because the chip can only write-protect the external L2 cache and the DRAM. A write to on-CPU cache would result in a loss of coherency between L1 cache and any external memory.

On the 82C465MVB part, bit 3Fh[2] provides the option of invalidating the cache line instead. If bit 3Fh[2]=1, and shadow RAM is programmed to be cacheable in L1, a write to a shadow RAM location results in generation of EADS# and the invalidation of that cache line. This feature provides better performance. For example, frequently executed BIOS code shadowed at F000h can be cached in L1.

Write-Protected DRAM Cache Control

Index	Name	7	6	5	4	3	2	1	0
3Fh	Misc. Control Register						Invalidate L1 Cache Line on Writes to WP DRAM 0=Disable 1=Enable (MVB)		

4.5.3 L1 Write-Back Cache Support

The 82C465MV incorporates support for CPUs with a level-one (on-chip) write-back cache such as that found on the AMD 486Plus processor, the Cyrix Cx486DX processor, and the Intel P24D processor.

4.5.3.1 Hardware Considerations

L1 write-back cache support requires two additional signals, HITM# and BOFF#. The 82C465MV bus controller manages L1 cache without requiring dedicated pins by using the following scheme. The logic operates properly only if the 82C465MV registers have been programmed to recognize and generate L1 cache signals.

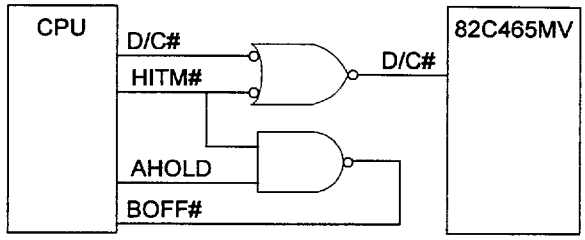
- HITM# is combined with the existing D/C# input to the bus controller. Since D/C# will always be high during a bus snoop cycle (after EADS# goes active), HITM# can be combined with D/C#. The D/C# input to the 82C465MV is a HITM# input for only one cycle, on the second or third clock after it sees EADS# active.

- BOFF# is generated externally from the AHOLD output qualified with the HITM# output of the 82C465MV logic. When the logic sets AHOLD active (high) along with HITM# inactive (high), BOFF# to the CPU must go low to request a restart of the current bus cycle.

Because HITM# does not have a dedicated input, it is monitored only during a specific window after EADS# occurs. The 82C465MV looks for HITM# to go active on the second or third clock edges after EADS#, according to programming; this provision accounts for the delay introduced by the external gate and ensures that all processors can meet the HITM# setup requirement of the cache support logic.

Figure 4-2 illustrates the typical circuit used to generate HITM# and BOFF#.

Figure 4-2 Generation of HITM# and BOFF#



4.5.3.2 Extra Programmable Pin Options

HITM# and BOFF# signal generation on the 82C465MVA part can be internal to avoid the requirement for two external gates. Note that these signals can be defined only if the new memory control interface has been strap selected on pin 79 at reset.

4.5.3.2.1 HITM# Input Option

HITM# can be directly input to the chip on the FLUSH# pin. Selecting the HITM# option will allow better performance for CPUs that cannot return HITM# fast enough for sampling on the second clock after EADS# when the external gate solution (D/C# + HITM#) is used. Note that until the HITM# option is selected, pin 135 is an output and will drive against the HITM# signal. However, both signals will be driving high in their normal state so no harm is done.

Note that the FLUSH# output can be eliminated only when SMBASE is relocated to A000h/B000h. This relocation eliminates the need for generating FLUSH# on entry to SMM, which is the only time the 82C465MV ever generates FLUSH#.

Pin Options

Index	Name	7	6	5	4	3	2	1	0
D6h	PMU Control Register 10				HITM# Source 0=D/C# 1=Pin 135 (MVA)				

4.5.3.2.2 BOFF# Output Option

Pin 189 (LCLK/TAGCS#) is redefined as BOFF# after reset under the following conditions:

1. Pin 79 (DACKMUX0) sensed low at reset, indicating that the new 82C465MV memory control interface must be enabled
2. Pin 146 (SA0) sensed high at reset, indicating that the L2 cache interface is not used.

The LCLK function of pin 189 should never be needed on 82C465MV designs, because it is the same 1X clock as FBCLKOUT. Also, the TAGCS# function of pin 189 for L2 cache should never be needed on designs using L1 write-back cache CPUs. Therefore, the pin changes function to accommodate the need for BOFF# on L1 write-back CPUs.

If desired, setting bit D4h[0]=1 will reassign pin 189 as LCLK even though conditions 1 and 2 above have been met.

Index	Name	7	6	5	4	3	2	1	0
D4h	Resistor Control Register 1								Redefine Pin 189 0=BOFF# 1=LCLK (MVA)

4.5.3.3 Programming

The L1 cache option must first be preset through setting bit A0h[1] = 1. The HITM# sensing is set according to the CPU used and speed of operation through bit D1h[7]. After cacheable ranges have been established as described in the previous section, the cache operation is then enabled by writing bit 35h[1] = 1.

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1							CPU Cache Operation Select 0 = Standard 1 = L1 write-back	
D1h	L2 Cache Control Register 2	L1 Cache HITM# sensing after EADS# 0 = 2nd clock 1 = 3rd clock							

L1 Cache HITM# Sensing bit D1h[7] - selects the cycle during which the 82C465MV looks for HITM# after it sees EADS# low. CPUs that cannot reliably meet the setup time requirements for HITM# to be returned on the second clock after EADS# should use the default third-clock setting.

0 = Sample HITM# (on D/C# input) on 2nd clock after sampling EADS# low

1 = Sample HITM# on 3rd clock after sampling EADS# low

4.5.3.4 Burst Write Feature

The 82C465MVA part provides the burst write feature for L1 write-back CPUs. The feature is enabled only when the L1 write-back feature is enabled. On the 82C465MVB part, bit 3Fh[4] allows the burst write feature to be enabled independently for non L1-write-back CPUs. Regardless of the setting of 3Fh[4], burst writes are always enabled when the L1 write-back feature is selected.

Burst Write Control

Index	Name	7	6	5	4	3	2	1	0
3Fh	Misc. Control Register				CPU Burst Write Support 0=Disable 1=Enable (MVB)				

4.5.4 L2 Cache Support

The 82C465MV incorporates support for an L2 (external) write-back cache. The L2 cache support is optional, and is engaged when the 82C465MV initialization logic sees SA0 low at hardware reset time. As soon as reset is complete, the logic redefines a large block of pins for L2 cache support. Figure 4-3 illustrates the connection of the L2 cache to the chip. Table 4-8 lists the "No Cache Support" interface versus the "L2 Cache Support" interface.

Figure 4-3 L2 Cache Connection - Two Bank Configuration

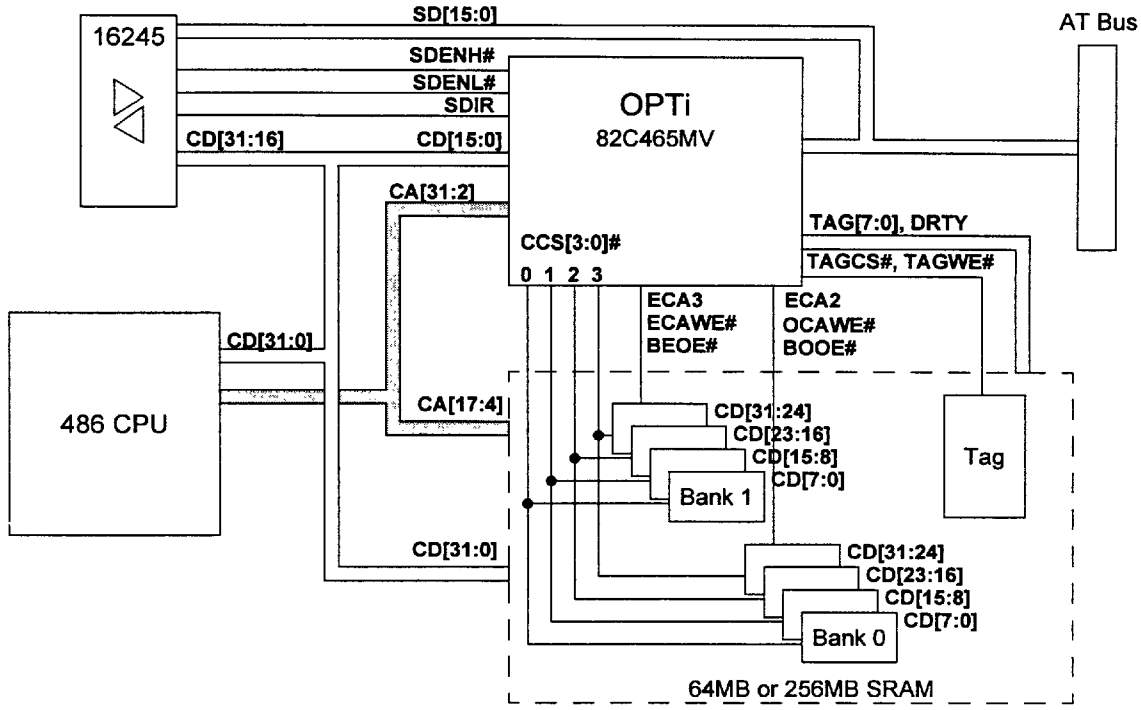


Table 4-8 L2 Cache Support Signal Correspondence

"L2 Cache Support" Signal	"No Cache Support" Signal	Impact on System Design
BEOE# (O) - Cache Output Enable, Even	CD25	External buffers required to move CD31:16 data down to SD15:0
BOOE# (O) - Cache Output Enable, Odd	CD26	
ECAWE# (O) - Cache Write Enable, Even	CD27	
OCAWE# (O) - Cache Write Enable, Odd	CD28	
TAGWE# (O) - Tag RAM write enable	CD29	
DRTY (I/O) - Dirty bit to/from tag RAM	CD24	
TAG7:0 (I/O) - Tag RAM data bus	CD23:16	
ECA2 (O) - Early CA2	CD30	
ECA3 (O) - Early CA3	CD31	
TAGCS# (O) - Tag RAM chip select	LCLK	LCLK no longer available
SDIR (O) - SD15:0 to CD31:16 buffer dir.	XDIR	XDIR must be derived externally
SDENH# (O) - SD15:8 to CD31:24 buffer en.	KBDCS#	None - combine with DWE#
SDENL# (O) - SD7:0 to CD23:16 buffer en.	RTCD#	None - combine with ROMCS#

Table 4-8 L2 Cache Support Signal Correspondence (cont.)

"L2 Cache Support" Signal	"No Cache Support" Signal	Impact on System Design
CCS0# (O - 3.3V) - Cache chip select 0	MP0	Memory parity checking is no longer available
CCS1# (O - 3.3V) - Cache chip select 1	MP1	
CCS2# (O - 3.3V) - Cache chip select 2	MP2	
CCS3# (O - 3.3V) - Cache chip select 3	MP3	

4.5.4.1 Performance

The L2 cache implementation is a full-performance scheme. Because the 82C465MV provides all control signals and the tag RAM data connections directly, there are no external buffer or gate delays and no extra clocks required for synchronization.

The integrated CCS0-3# select lines provide an additional feature. Besides eliminating the need to externally gate W/R# with each of the BE0-3# lines, these dedicated chip select lines go active only when the cache is actually being accessed. Consequently, the cache consumes less power when not actively being accessed.

4.5.4.2 L2 Cache Operation Details

The integrated cache controller uses a direct-mapped, bank-interleaved scheme to dramatically boost the overall performance of the local memory subsystem by caching writes as well as reads (write-back mode). Cache memory can be configured as one or two banks, and sizes of 64KB, 128KB, and 256KB are supported. The cache controller operates in non pipeline mode, with a fixed 16-byte line size (optimized to match a 486 burst line fill) in order to simplify the motherboard design without increasing cost or degrading system performance. For 486 systems, the secondary cache operates independently and in addition to the internal cache of the CPU.

4.5.4.2.1 Cache Bank Interleave

In order to support cache burst cycles at elevated frequencies and still utilize conventional-speed SRAMs, a bank-interleave cache access method is employed. The addresses are applied to the cache memory one cycle earlier, while cache output enable signals control even/odd bank selection and enable cache RAM data to the CPU data bus. Since the output enable time is about half of the address access time, the 82C465MV can achieve a high performance cache burst mode without using more expensive high-speed SRAMs.

The 82C465MV chip supports one or two cache banks. Two cache banks are required to interleave and optimally realize the performance advantages of this cache scheme. A selection of 128KB is a single-bank cache, while 64KB and 256KB cache sizes are two-bank configurations. When using a two-bank configuration, the even and odd banks receive mostly the same address lines; signals ECA3/ECA2, ECAWE#/OCAWE# and BEOE#/BOOE# are used to dictate the even or odd bank access.

4.5.4.2.2 Write-Back Cache

The write-back cache scheme derives its superior performance by optimizing write cycles. There is no performance penalty in the cache write cycle, since the cache controller does not need to wait for the much slower DRAM controller to finish its import before proceeding to the next cycle.

4.5.4.2.3 Tag RAM

A built-in tag comparator improves system performance while reducing component count on the system board. The comparator internally detects the cache hit/miss status by comparing the high-order address bits (for the memory cycle in progress) with the stored tag bits from previous cache entries (see Table 4-9). When a match is detected, and the location is cacheable, a cache-hit cycle takes place. If the comparator does not match, or a non cacheable location is accessed (based on the internal non cacheable region registers), the current cycles is a cache miss.

The tag is invalidated automatically during memory reads when the cache is disabled; each memory read will write into the corresponding tag location a non cacheable address (such as A0000h or B0000h of the video memory area). To flush the cache, simply disable the L2 cache and read a block of memory equal to the size of the cache. The advantage of this invalidation scheme is that no valid bit is necessary and expensive SRAM can be conserved.

Table 4-9 details CPU address bits that are stored as tags for the various cache sizes supported by the cache controller. The table also marks the high-order address bit in each configuration. This is the bit that can be eliminated in each configuration and replaced by the dirty bit if the design must use an 8-bit-wide tag SRAM instead of a 9-bit-wide device. Table 4-10 illustrates the consequences of this choice for each configuration.

Table 4-9 Correspondence Between Tag Bits and CPU Address Lines

Tag Bit	64KB Cache	128KB Cache	256KB Cache
7	CA23*	CA23	CA23
6	CA22	CA22	CA22
5	CA21	CA21	CA21
4	CA20	CA20	CA20
3	CA19	CA19	CA19
2	CA18	CA18	CA18
1	CA17	CA17	CA25*
0	CA16	CA24*	CA24

* Optional Tag Bit

Table 4-10 Maximum Cacheable System DRAM for Each Cache Configuration

Tag/Dirty SRAM Width	Maximum DRAM Possible		
	64KB Cache	128KB Cache	256KB Cache
9-bit	16MB	32MB	63MB*
8-bit	8MB	16MB	31MB*

* Not 64MB or 32MB because those addresses in the upper MB conflict with the value used to invalidate cache.

4.5.4.2.4 Dirty Bit Mechanism

The "dirty bit" is a mechanism for monitoring data coherency between the external cache subsystem and DRAM. Each tag entry has a corresponding dirty bit to indicate whether the data in the represented cache line has been modified since it was loaded from system memory. This bit allows the cache controller to determine whether the data in memory is "stale" and needs to be updated before a new memory location is allowed to overwrite the currently indexed cache entry. The write-back cycle causes an entire cache line (16 bytes) to be written back to memory, followed by a line burst from the new memory location into the cache and CPU. Normally, the performance advantage of completing fast writes to the cache outweigh the "write-back" read-miss penalties which are incurred while operating the write-back scheme.

Possible cache cycles are detailed below:

Cache Read-Hit. The secondary cache provides the data to the CPU directly. The cache controller follows the CPU burst protocol to fill the internal cache line of the processor.

Cache Read-Miss (DRTY bit negated): Import Cycle. The cache controller does not need to update system memory with the current data from the cache, because that data has not been modified (as shown by the dirty bit negation). The cache controller asserts TAGWE# to update the tag RAMs with the new address, and asserts BEOE#/BOOE# to update cache memory with data from the new DRAM line. Data is presented to the CPU and the secondary cache concurrently (following the 486 burst protocol).

Cache Read-Miss (DRTY bit asserted): Castout Cycle. The cache controller must update the system memory with data from the cache location that is going to be overwritten. The cache controller writes the 16-byte line from cache memory into DRAM, then reads the new line from DRAM into the cache memory and de-asserts the dirty bit. The cache controller asserts TAGWE# and BEOE#/BOOE# during this line fill. This new data is presented to the CPU and to the secondary cache concurrently (following the 486 burst protocol).

Cache Write-Hit. Because this is a write-back cache, the cache controller does not need to update the much slower DRAM memory. Instead, the controller updates the cache memory and sets the DRTY bit. DRTY may already be set, but that does not affect this cycle. The contents of the tag RAM remains unmodified.

Cache Write-Miss. The cache controller bypasses the cache entirely and writes the data directly into DRAM. The DRTY bit is unchanged. No import cycle to the cache takes place.

Table 4-11 shows recommended DATA and TAG SRAM speeds for relative CPU clock rates.

Table 4-11 SRAM Speed Requirements

Speed	Cache SRAM	TAG SRAM	DRAM speed	Write Wait States	Burst Read Timing	Bank Arrangement
16MHz	25ns	25ns	80ns	0	2-1-1-1	Single/double bank
20MHz	25ns	25ns	80ns	0	2-1-1-1	Single/double bank
25MHz	20ns	25ns	80ns	0	2-1-1-1	Single bank
25MHz	25ns	25ns	80ns	0	2-1-1-1	Double bank
33MHz	20ns	15ns	80ns	0	3-2-2-2	Single bank cache
33MHz	20ns (Note 2)	15ns	80ns	0	2-1-1-1	Double bank cache only
40MHz	20ns	15ns	80ns	1	3-2-2-2	Single/double bank
50MHz	20ns	15ns	80ns	1	3-2-2-2	Single/double bank

1. DRAM and cache cycles are at their minimum wait states.
2. 20ns SRAM with Tdoe ≤ 10ns

4.5.4.3 L2 Cache Arrangement

The 82C465MVA part provides two new bits for greater flexibility of L2 cache.

- Bit D1h[4] selects single-bank operation of L2 cache. The original 82C465MV part provided for single-bank operation of 128KB cache only. This new feature allows 64KB and 256KB cache to operate in a single bank mode.
- Bit D1h[5] allows the use of a 7-bit tag RAM address in order to use an 8-bit-wide tag SRAM instead of a 9-bit-wide SRAM. The original 82C465MV part allowed this arrangement, but since there was no way to indicate to the chipset to make upper DRAM non cacheable, the 8-bit tag RAM selection would severely limit the maximum possible system DRAM.

L2 Cache Arrangement Selection Bit

Index	Name	7	6	5	4	3	2	1	0
D1h	L2 Cache Control Register 2			L2 Tag RAM Size 0=8-bit 1=7-bit (MVA)	L2 Cache Arrangement 0=Two banks 1=One bank (MVA)				

4.5.4.4 Differences Between L2 Support and No Cache Support Modes

When the L2 cache feature is strap-selected, the following changes must be made to the system design.

- The CD31:16 inputs to the 82C465MV controller logic are redefined by the L2 cache interface. CD31:16 are used only for data exchanges between the SD15:0 bus and the CPU, and do not affect the CPU-to-memory interface. Enabling L2 support requires the use of a word-wide, level-translating transceiver between CD31:16 and SD15:0. Whenever bus exchanges take place between the ISA bus and the local CPU bus, the 82C465MV AT controller directs data onto the correct byte lanes as appropriate.
- The MP3:0 pins are used as the cache chip select lines CCS#0-3. Only the parity checking feature is lost, which is not important since parity DRAM is not generally used on portable systems.
- RTCD# is multiplexed with ROMCS# on the old ROMCS# pin. The new pin is valid as RTCD# when AEN is low, and as ROMCS# always. Only RTCD# needs to be qualified with AEN, as ROMCS# to the ROM is further qualified by MEMR# going active.
- The old RTCD# pin becomes the low-byte enable signal SDENL# to the CD-SD buffer.
- KBDCS# is multiplexed with DWE# on the old DWE# pin. The new pin is valid as KBDCS# when AEN is low, and as DWE# always. Only KBDCS# needs to be qualified with AEN, as DWE# to the DRAM is further qualified by one of the RAS# lines going active.
- The old KBDCS# pin becomes the high-byte enable signal SDENH# to the CD-SD buffer.
- The old XDIR pin becomes the CD-SD buffer direction signal SDIR. Systems that are designed using the OPTi 82C602 chip do not need an XDIR signal. For those designs using discrete logic for the XD bus, the XDIR function must be derived externally using W/R# from the CPU to control XD bus buffer direction, and ROMCS#/RTCD# ANDed with DWE#/KBDCS# to enable the buffer.
- The old LCLK pin becomes the TAGCS# control line to tag RAM. The LCLK function should not be needed in a new design since the 82C465MV clock FBCLKOUT runs at the proper speed for the VL bus (always 1X).

These functions are controlled separately from the L2 cache enable feature to allow systems to be designed with all the L2 cache support logic in place and operating, without actually having to install and enable the cache itself on every board. Not until the "Enable L2 Cache Operation" control bit is set in the Feature Control Register do the converted pins actually begin to function for cache support. The strapping option does, however, change pin definitions so that the CD-SD buffer is used in all transactions involving CD[31:16].

Table 4-12 illustrates the actual signal changes per pin.

Table 4-12 L2 Cache Support Option (strap-selected)

Pin	82C463MV Signal	82C465MV Signal w/ L2 Option Strapped	Pin	82C463MV Signal	82C465MV Signal w/ L2 Option Strapped
149	XDIR	SDIR	205	CD19	TAG3
89	RTCD#	SDENL#	204	CD20	TAG4
104	KBDCS#	SDENH#	203	CD21	TAG5
90	ROMCS#	ROMCS#+RTCD#	202	CD22	TAG6
25	DWE#	KBDCS#+DWE#	199	CD23	TAG7
23	MP0	CCS0#	197	CD24	DRTY
13	MP1	CCS1#	196	CD25	BEOE#
3	MP2	CCS2#	195	CD26	BOOE#

Pin	82C463MV Signal	82C465MV Signal w/ L2 Option Strapped	Pin	82C463MV Signal	82C465MV Signal w/ L2 Option Strapped
198	MP3	CCS3#	194	CD27	ECAWE#
189	LCLK	TAGCS#	193	CD28	OCAWE#
2	CD16	TAG0	192	CD29	TAGWE#
207	CD17	TAG1	191	CD30	ECA2
206	CD18	TAG2	190	CD31	ECA3

4.5.4.5 Hardware Considerations

The system designer should be aware of the following information when designing in L2 cache.

DWE# is on the CPU/memory interface and is therefore a 3.3V signal in a mixed-voltage system. Since the DWE# pin is shared by KBDCS#, KBDCS# is now a 3.3V signal also. The 7432 gate normally used to decode KBDCS# using AEN will have a 3.3V high input. If a 5V gate is used, the 3.3V inactive input at low-power suspend time could cause a current drain.

Also, the sense of SDIR is as follows: logic HIGH indicates an A-to-B exchange which is from CD31:16 to SD15:0; logic LOW indicates a B-to-A exchange which is from SD15:0 to CD31:16. System designers must be careful to orient the 'A' and 'B' sides of the buffer properly.

L2 cache is often placed on its own separate power plane so that it can be powered down when the system suspends. Bit D0h[6] is provided as explained below to select whether the cache control signals will be tri-stated or just driven inactive (high) during suspend.

4.5.4.6 Programming

Support for L2 cache is enabled by strapping SA0 low as described in the Strapping Option Summary section of this document. Any system designed for L2 cache should enable this support, whether the cache is actually installed or not. The size of the cache, the wait states, and the cacheable areas are programmed in L2 Cache Control Registers 1, 2, and 3. Then, the cache is actually enabled for operation by writing bit 5 = 1 in L2 Cache Control Register 1.

Since the MP3:0 pins are normally enabled for parity checking, the parity check function is automatically disabled when L2 cache support is enabled (SA0 pulled low at reset), regardless of the setting of the parity enable bit at index 31h.

NOTE When L2 cache is enabled, the 82C465MV part requires the DRAM controller to run no faster than 4-3-3-3 DRAM read cycles.

L2 Cache Registers

Index	Name	7	6	5	4	3	2	1	0
D0h	L2 Cache Control Register 1	L2 cache, CCS0-3# de-assert 0 = stop grant and suspend 1 = also btw. accesses	L2 cache controls state during suspend 0 = tri-state 1 = driven	L2 Cache Engage 0 = Disable 1 = Enable	Cache Size 0 0 = 64KB 0 1 = 128KB 1 0 = 256KB 1 1 = reserved		L2 Cache Write Wait State 0 = 1 ws 1 = no ws	L2 Cache Read Burst Wait State Control 0 = X-1-1-1 1 = X-2-2-2	L2 Cache First Read Wait State Control 0 = 3-X-X-X 1 = 2-X-X-X
D1h	L2 Cache Control Register 2					EC000-EFFFFh L2 cacheable 0 = No 1 = Yes	E8000-EBFFFh L2 cacheable 0 = No 1 = Yes	E4000-E7FFFh L2 cacheable 0 = No 1 = Yes	E0000-E3FFFh L2 cacheable 0 = No 1 = Yes
D2h	L2 Cache Control Register 3	DC000-DFFFFh L2 cacheable 0 = No 1 = Yes	D8000-DBFFFh L2 cacheable 0 = No 1 = Yes	D4000-D7FFFh L2 cacheable 0 = No 1 = Yes	D0000-D3FFFh L2 cacheable 0 = No 1 = Yes	CC000-CFFFFh L2 cacheable 0 = No 1 = Yes	C8000-CBFFFh L2 cacheable 0 = No 1 = Yes	C4000-C7FFFh L2 cacheable 0 = No 1 = Yes	C0000-C3FFFh L2 cacheable 0 = No 1 = Yes

L2 Chip Select Control bit D0h[7] - selects when the CCS0-3# and TAGCS# signals should go inactive. Some cache RAM may not be ready for access in time if its chip enable is turned off between cycles. Setting D0h[7] = 0 lets CCS0-3# and TAGCS# go inactive only during stop grant cycles and during suspend mode. Setting D0h[7] = 1 lets CCS0-3# and TAGCS# go inactive between cache accesses as well as during stop grant and suspend mode.

L2 Chip Select State During Suspend bit D0h[6] - allows cache to be flushed and turned off during suspend mode if desired to save power. Software must perform the flush.

L2 Cache Engage bit D0h[5] - enables L2 write-back cache operation if chip was strapped for L2 cache configuration. Otherwise, the bit setting has no effect. This bit would remain set to 0 on a system that was predisposed to accept cache but on which no cache was presently installed.

L2 Cache Read Burst Wait State Control bit D0h[1] - When D0h[1] = 0, read hit burst cycles run with no wait states (X-1-1-1). When D0h[1] = 1, read hit burst cycles run with one wait state (X-2-2-2).

L2 Cache First Read Wait State Control bit D0h[0] - When D0h[0] = 0, the first read hit cycle runs with no wait states (2-X-X-X). When D0h[0] = 1, the first read hit cycle runs with one wait state (3-X-X-X).

4.5.4.7 Timing Control Register

The superior performance of EDO DRAM requires timing that is tighter than for standard DRAM. This timing becomes especially critical during an L2 cache read miss cycle, because the DRAM read and L2 cache write occurs in the same clock cycle and from the same clock edge. Therefore, the 82C465MVB part provides bits 3Ch[2:0]. These bits control the number of gate delays inserted to delay the cache ECAWE#, OCAWE#, and TAGWE# signals so that the L2 cache write occurs after EDO DRAM read data is ready. The required setting for these bits depends on system layout.

Cache Timing Control

Index	Name	7	6	5	4	3	2	1	0
3Ch	Timing Control Register						L2 Cache WE# Delay (MVB) 000=No delay 001=1 gate delay . . 110=6 gate delays 111=7 gate delays		

4.6 Peripheral Interface Logic

The peripheral interface logic of the 82C465MV chip includes AT-bus control logic, the 82C206-type IPC, the integrated local-bus IDE controller, and the Compact ISA (CISA) interface.

4.6.1 AT Bus Logic

The 82C465MV handles all the typical aspects of AT bus operation. 8- or 16-bit transactions can take place on the AT bus, depending on the state of the IO16# and M16# lines during the transaction.

All AT-bus commands generated by the CPU will be passed first to the VL bus. If no local device responds by activating LDEV#, the 82C465MV bus controller runs the cycle on the AT bus. Even I/O accesses that are destined for internal devices such as the DMA controller and interrupt controller will be presented on the AT bus. It is therefore important that no external device attempt to respond to these cycles as well. Bus contention and invalid data could result.

Note that write accesses to internal configuration registers at 022h and 024h are also available outside the chip for any external logic that needs to record these transactions. Read accesses to these registers are available only on the CPU interface.

4.6.1.1 Hardware Considerations

The design of the ISA subsystem is heavily dependent on the load and the target power consumption. For example, a 3.3V AT subsystem is ideal in terms of power consumption characteristics, yet not all AT bus peripherals can operate at 3.3V. Therefore, the designer may need to separately buffer 3.3V and 5V buses and implement a control isolation mechanism. In addition the designer must consider the XD bus, which also can be either 3.3V or 5V, and determine whether this separate local peripherals bus is necessary as described below.

4.6.1.1.1 XD Bus Buffer Control

The logic provides an XDIR signal to control the movement of 8-bit ISA bus data to and from the SD[15:0] bus, always on the lower byte. When the XDIR signal is not available, as when the L2 cache interface is selected, the buffer direction can be derived from MEMR# and IOR#, and the buffer enable from ROMCS#, RTCD#, and KBDCS#.

Note that a separate XD bus is not always needed in a design. If there are not a large number of devices on the SD bus, the XD bus peripherals can simply be placed directly on the SD bus. The only absolute requirement for a separate XD bus is for designs in which SD and XD run at different voltages and the XD bus buffer acts also as a level translator.

4.6.1.1.2 SD Bus Buffer Control

The internal bus controller logic automatically splits word or double-word writes from the CPU to multiple eight-bit ISA-bus cycles. If the responding device asserts M16# or IO16#, the bus controller will transfer 16 bits at a time.

Conversely, CPU memory reads, which can be 32-bits wide, will automatically be assembled by the 82C465MV bus conversion logic either byte-by-byte or word-by-word, depending on whether M16# is asserted by the peripheral. The bytes or words are moved individually from the SD bus to the appropriate lanes on the CD bus as indicated by the CPU BE[3:0]# lines until the complete data word is ready. The 82C465MV logic then finally asserts RDY# to transfer the 8-, 16-, or 32-bit data back to the CPU.

4.6.1.1.3 SD-to-CD Bus Buffer Controller

When the L2 cache interface is implemented, the CD[31:16] signal interface is converted to various tag data and cache control signals. Since the CD[31:16] inputs are always moved down to the SD[15:0] bus anyway, this operation is readily achieved using an external 16-bit transceiver. In most cases the transceiver will also be a level-translator, since the CD bus is generally 3.3V and the SD bus is 5V. A 74FCT164245 level translator device or a 74LVT16245 5V-tolerant device are often used for this purpose as shown in Figure 4-3.

For any data exchange that requires a direct byte or word movement from CD[31:16] to SD[15:0], the bus controller sets the direction on SDIR to point to the SD bus, sets its internal SD bus buffer to input (so it can capture the data being written to the AT bus in case it needs to act on the data), and then enables SDENH# or SDENL# as appropriate. For direct movement from SD[15:0] to CD[31:16], the process is similar but the SDIR direction changes. Once again, the bus controller captures the data on its internal SD[15:0] lines in case it needs to act on the data.

For any operation that involves a byte-swap, the bus controller must direct the CD-SD buffer output to its internal SD[15:0] input. It then latches this data, performs the required byte swap, disables the SDENH# and SDENL# lines to the CD-SD buffer, and finally drives the correctly swapped data to the AT bus.

These operations take place at CPU clock speeds, not AT bus speeds; therefore, the impact on operational speed is negligible.

4.6.1.1.4 MASTER# Control

The internal logic can recognize any active DRQ and an active HLDA from the CPU as an indication that an AT-bus peripheral device has bus ownership. It can further determine that the device is a bus master, as opposed to a DMA slave, by looking at the state of AEN. Therefore, the 82C465MV can determine whether the current AT bus cycle is a master cycle without having to observe the MASTER# input pin.

The RI# pin was formerly shared with the MASTER# input on the 82C463 chipset and can continue to act as a MASTER# input if bit 30h[5] is written to '0'. However, this function need not be supported and should not be implemented in new designs. MASTER# from the AT bus should be used only to control the direction of any CA-to-SA bus buffers in use.

Pin 186 Function Select

Index	Name	7	6	5	4	3	2	1	0
30h	Control Register 1			Pin 186 function 0 = MSTR# 1 = RI					

4.6.1.2 AT Write Cycle Inhibition

The cycle enable bits listed below default to "enabled" to allow write accesses to the AT bus or EPROM to be generated normally. In those situations where ROMs are present, the write cycles can be blocked.

Cycle Enable Bits

Index	Name	7	6	5	4	3	2	1	0
32h	Shadow RAM Control Register 1		Enable D000 writes 0 = Disable 1 = Enable	Enable E000 writes 0 = Disable 1 = Enable					
36h	Shadow RAM Control Register 3				Enable C000 writes 0 = Disable 1 = Enable				

4.6.1.3 AT Bus Clock Options

The AT bus can run at exactly 8.0MHz on the 82C465MV if an appropriate external clock is provided. This feature requires a new input pin, ATCLKIN. If this feature is enabled, ATCLKIN will replace the PIO2 signal presently found on pin 172 of the 82C463MV.

ATCLKIN is simply an alternative clock source. It can be divided in the same way as the FBCLKIN source. An 8.0MHz clock will most likely be derived by dividing a 16MHz ATCLKIN by 2, or a 24MHz ATCLKIN by 3.

The ATCLKIN input for this function must be enabled first as explained in the "Clock Generation" section. The clock can then be selected by writing bit 43h[3] = 1, and using bits 43h[2:0] to choose the divisor. Bit 3 defaults to 0 on power-up.

NOTE The 82C465MV can run from either a 1X clock or a 2X clock (refer to the "Strap-Selected Interface Options" section for details). To remain compatible in both modes, the AT bus clock selections are based on the FBCLKIN (feedback) clock times 2. This scheme effectively maintains the same rate as would have been selected through 82C463MV programming.

4.6.1.4 AT-Bus Refresh Control

The 82C465MVB part allows refresh on the AT bus to be completely eliminated. Since very few AT bus devices actually make use of the refresh cycle, this bandwidth can be recovered to improve system performance. Setting bit 32h[2]=1 disables AT bus refresh and generation of addresses for refresh as well, but does not in any way affect local system DRAM refresh.

AT-Bus Refresh Control

Index	Name	7	6	5	4	3	2	1	0
32h (MVB)	Shadow RAM Control Register 1						Refresh on AT Bus 0=Disable 1=Enable (MVB)		

4.6.1.5 Programming

The AT bus requires a certain amount of programming in order to preset and optimize its operation for the peripheral devices chosen.

Index	Name	7	6	5	4	3	2	1	0
30h	Control Register 1				Turbo VGA, (NOWS#) 0 = Disable 1 = Enable		AT wait states 0 = None 1 = One		
31h	Control Register 2	Master byte swap 0 = Disable 1 = Enable							
32h	Shadow RAM Control Register 1								ALEs in bus conversion 0 = Multiple 1 = Single
A0h	Feature Control Register 1	Internal I/O Address Decoding 0 = 10-bit 1 = 16-bit			Pin 172 0 = PIO2 (or CPUSPD) 1 = ATCLKIN				

Turbo VGA forces zero wait state operation from memory accesses at addresses A0000h to B0000h, as if NOWS# were always active.

AT wait state control adds one extra wait state to each AT bus cycle, useful for slower devices.

Master byte swap control enables AT bus byte swapping for bus masters. While some AT-bus masters are capable of monitoring the IOCS16# and MEMCS16# AT-bus signals to determine where to drive data, others depend on the system to do the byte swap for them. This bit allows either type of bus master to be accommodated.

ALE control during bus conversion allows selection of a single ALE on the first cycle or an ALE on the subsequent command cycle as well when word accesses are split into two separate byte accesses. Most all newer peripheral devices require two separate ALEs.

Internal I/O address decode size selection allows addressing to wrap around every 400h ports or to end after 100h. System designs with peripheral devices at aliases of the 0-FFh range (for example, an I/O device addressed at 420h) should use 16-bit decoding to prevent the internal peripheral devices from responding at aliased addresses and conflicting with the external devices.

4.6.1.6 AT Bus Address Buffer Enable Signal

The 82C465MVA interface provides the SABUFEN# signal output as a strap-selected option on pin 172. Pin 172 is normally used for the CPUSPD output indicator, the ATCLKIN bus clock input function, or as general purpose I/O pin PIO2. As PIO2, pin 172 defaults to input mode at power-up time. When the new function is programmed, pin 172 becomes the SABUFEN# output. SABUFEN# is normally high, and drives low on any AT bus access (including DMA, refresh, etc.). This signal is enabled early in all cycles so that the CA[23:2] bits can be driven onto the SA[23:2] bus in time for decode by AT peripheral devices.

The 82C465MVA enables a weak internal pull-down resistor on pin 172 after reset to enable any connected buffer. Once index register 57h (where input or output is selected for each PIO pin) is written with any value, the internal pull-down resistor is disabled.

SABUFEN# Program Bit

Index	Name	7	6	5	4	3	2	1	0
79h	PMU Control Register 11					Pin 172 function 0=PIO2 or CPUSPD 1=Buffer enable pin SABUFEN# (MVA)			

4.6.1.7 Docking Station Attachment Feature

The "hot" attachment of a notebook computer to a docking station through the AT bus requires the ability to stop any AT cycle in progress and tristate the bus. The 82C465MVA part implements this feature through an unused interrupt line on the multiplexed EPMMUX input. On the 82C465MV part, inputs C0-C3 to EPMMUX are defined as DRQ2, RSVD, EPMI3, and EPMI4. The RSVD line is always shown as pulled low in the current schematics. On the 82C465MVA part, the C1 input is redefined from RSVD to ATHOLD.

Bringing ATHOLD high causes the 82C465MVA logic to stop the CPU operation after the current bus cycle is complete. Since ATHOLD is recognized through a multiplexer, there is a maximum latency of 280ns from assertion of ATHOLD to recognition by the logic. In addition, there is the time required for the system to complete its current cycle, which could take on the order of microseconds for certain AT-bus cycles. Finally, the AT bus signals are all tristated and will remain tristated as long as ATHOLD is high. Bringing ATHOLD low again restarts the system.

This feature is always enabled.

4.6.2 Programmed Hardware Reset

The 82C465MVA part allows generation of hardware reset signal CPURST by writing a register bit. This function is useful for restoring the registers to their default condition in certain situations. For example, if a flash BIOS has been reprogrammed, the system must be re-initialized and booted properly. Using this bit may eliminate the need for some hardware reset logic.

Programmed Hardware Reset Bit

Index	Name	7	6	5	4	3	2	1	0
ADh	Feature Control Register 3		Generate CPURST immediately 0=No 1=Yes (MVA)						

4.6.3 Integrated Peripheral Controller

The Integrated Peripheral Controller (IPC) includes two 8237 DMA controllers, two 8259 interrupt controllers, one 8254 timer/counter and one 74612 memory mapper. It is register-compatible with the 82C206 chip.

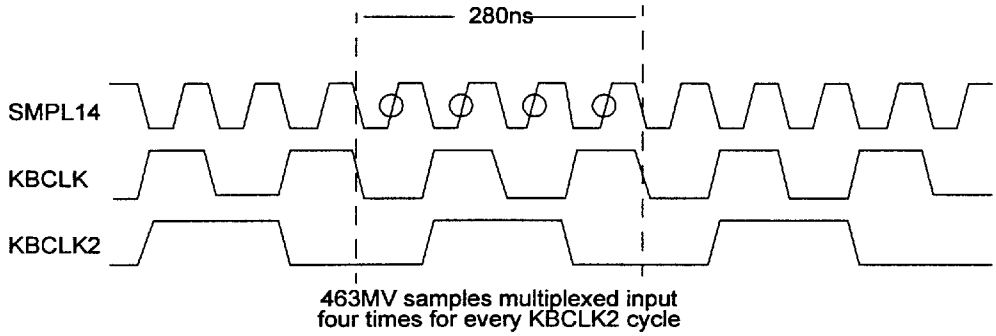
For information on the design architecture of this unit, refer to the separate document on the 82C206 IPC. This document is available on request from OPTi.

4.6.3.1 Hardware Considerations

The 82C465MV uses an external multiplexing scheme to read in many of the IRQ, DRQ, and external PMI inputs. The scheme uses the KBCLK and KBCLK2 outputs to toggle a 74153-type multiplexer through four distinct sampling phases. While the system is active, KBCLK and KBCLK2 are generated from the OSC14 input and sample each multiplexer input once every 280ns. During suspend, if OSC14 is not present the 32KHz signal is used to generate KBCLK and KBCLK2. Therefore, the inputs are sampled only once every 120us in this case.

Sampling occurs between multiplexer input switching. For example, when in active mode and running off OSC14, KBCLK/ KBCLK2 switch the multiplexer input every 70ns. The chipset samples the state of its input 35ns after the multiplexer has switched. Therefore, no "glitching" occurs on sampling. The sampling points are shown in Figure 4-4, where the "SMPL14" signal is a delayed internal version of the OSC14 input signal.

Figure 4-4 Multiplexed Input Sampling Points



External peripheral devices that generate IRQs or external PMIs must therefore generate a pulse of sufficient duration to be seen by the sampling logic. The OPTi 82C602 Notebook Companion chip incorporates a latching mechanism to ensure that IRQ inputs that pulse low will be held low for at least one complete KBCLK/KBCLK2 cycle (280ns or 120us).

4.6.3.2 IPC Configuration Programming

The sole configuration register of the IPC, separate from those of the 82C465MV, is accessed by first writing the register index of interest to I/O port 022h; the selected register information then becomes available for reading or writing at I/O port 023h as opposed to port 024h used by the 82C465MV configuration registers.

IPC Configuration Bits

Index	Name	7	6	5	4	3	2	1	0
01h	IPC Config. Register	IPC Register Access Wait States (ATCLKs) 00 = 1 wait states 01 = 2 wait states 10 = 3 wait states 11 = 4 wait states (default)		16-bit DMA Wait States* 00 = 1 wait state (default) 01 = 2 wait states 10 = 3 wait states 11 = 4 wait states		8-bit DMA Wait States* 00 = 1 wait state (default) 01 = 2 wait states 10 = 3 wait states 11 = 4 wait states		Delay DMA MEMR# one clock from system MEMR# 0 = Yes (AT-compatible - default) 1 = No	DMA Clock Select 0 = ATCLK/ 2, (default) 1= ATCLK

* Note that IOCHRDY can also be asserted by DMA devices to add wait states to DMA cycles.

4.6.3.3 Interrupt Controller Register Programming

The IPC provides two peripheral interrupt controllers that are register compatible with the 8259 part. The registers of this logic module are listed below. These registers are accessed directly through the I/O subsystem (no index/data method is used).

4.6.3.3.1 Initialization Command Words

The Initialization Command Words (ICWs) are shown first and must always be written in sequence starting with ICW1. Two I/O port groups are listed. The first group refers to INTC1, the interrupt controller for IRQs 0-7; the second refers to INTC2, the interrupt controller for IRQs 8-15.

INTC1 Initialization Command Words

Port	Name	7	6	5	4	3	2	1	0
020h	ICW1 (write-only)	Don't care			Always = 1	Trigger Mode 0 = Edge, 1 = Level	Don't care	Cascade Mode Select 0 = Yes (always), 1 = No	Don't care
021h	ICW2 (write-only)	V[7:3] - Upper bits of interrupt vector. For AT compatibility, write 08h.					Not used - lower bits of interrupt vector are generated by interrupt controller.		
021h	ICW3 (write-only)	S[7:0] - Slave mode controller connections. For AT compatibility, write 04h (IRQ2).							
021h	ICW4 (write-only)	Don't care			Enable Multiple Interrupts 0 = No, 1 = Yes	Don't care		Enable Auto End-of-Interrupt Command 0 = No 1 = Yes	Don't care

INTC2 Initialization Command Words

Port	Name	7	6	5	4	3	2	1	0
0A0h	ICW1 (write-only)	Don't care			Always = 1	Trigger Mode 0 = Edge, 1 = Level	Don't care	Cascade Mode Select 0 = Yes (always), 1 = No	Don't care
0A1h	ICW2 (write-only)	V[7:3] - Upper bits of interrupt vector. For AT compatibility, write 70h.					Not used - lower bits of interrupt vector are generated by interrupt controller.		
0A1h	ICW3 (write-only)	Don't care					ID[2] - Slave mode address. For AT compatibility, write 02h (IRQ2).		
0A1h	ICW4 (write-only)	Don't care			Enable Multiple Interrupts 0 = No 1 = Yes	Don't care		Enable Auto End-of-Interrupt Command 0 = No 1 = Yes	Don't care

Enable Multiple Interrupts can be enabled to allow INTC2 to fully nest interrupts without being blocked by INTC1. Correct handling of this mode requires the CPU to issue a non specific EOI command to zero when exiting an interrupt service routine. If the feature is disabled, no command need be issued.

Automatic End of Interrupt can be enabled to allow the interrupt controller to generate a non specific EOI command on the trailing edge of the second interrupt acknowledge cycle from the CPU. The feature allows the interrupt currently in service to be cleared automatically on exit from the service routine. This function should not be used with fully nested interrupts except by INTC1.

4.6.3.3.2 Operational Command Words

The Operational Command Words are used to program the interrupt controller during the course of normal operation. Two I/O port addresses are listed for each register. The first address refers to INTC1, the interrupt controller for IRQs 0-7; the second refers to INTC2, the interrupt controller for IRQs 8-15.

INTC1 and INTC2 Operational Command Words

Port	Name	7	6	5	4	3	2	1	0
021h, 0A1h	OCW1 Mask Register	IRQ7/15 0 = Enable, 1 = Mask	IRQ6/14 0 = Enable, 1 = Mask	IRQ5/13 0 = Enable, 1 = Mask	IRQ4/12 0 = Enable, 1 = Mask	IRQ3/11 0 = Enable, 1 = Mask	IRQ2/10 0 = Enable, 1 = Mask	IRQ1/9 0 = Enable, 1 = Mask	IRQ0/8 0 = Enable, 1 = Mask
020h, 0A0h	OCW2 Command Register (write-only)	000 = Disable auto-rotate, auto EOI mode 100 = Enable autorotate, auto EOI mode 001 = Generate non specific EOI 011 = Generate specific EOI 101 = Rotate on non specific EOI 111 = Rotate on specific EOI 110 = Set Priority 010 = No operation			Always = 0 for OCW2	Always = 0 for OCW2	L[2:0] - Interrupt level acted on by Set Priority and Rotate of Specific EOI		
020h, 0A0h	OCW3 Command Register (write-only)	Always = 0	Allow bit 5 changes 0 = No 1 = Yes	Special Mask Mode 0 = Disable 1 = Enable	Always = 0 for OCW3	Always = 1 for OCW3	Polled Mode 0 = Disable (generate interrupt) 1 = Enable (poll 020/ 0A0h for interrupt)	Allow bit 0 changes 0 = No 1 = Yes	In-Service Access 0 = 020/ 0A0h reads return IRR 1 = Return ISR
020h, 0A0h	Interrupt Request Register OCW3[0] = 0 (read-only)	IRQ7/15 Pending 0 = No 1 = Yes	IRQ6/14 Pending 0 = No 1 = Yes	IRQ5/13 Pending 0 = No 1 = Yes	IRQ4/12 Pending 0 = No 1 = Yes	IRQ3/11 Pending 0 = No 1 = Yes	IRQ2/10 Pending 0 = No 1 = Yes	IRQ1/9 Pending 0 = No 1 = Yes	IRQ0/8 Pending 0 = No 1 = Yes
020h, 0A0h	In-Service Register OCW3[0] = 1 (read-only)	IRQ7/15 In Service 0 = No 1 = Yes	IRQ6/14 In Service 0 = No 1 = Yes	IRQ5/13 In Service 0 = No 1 = Yes	IRQ4/12 In Service 0 = No 1 = Yes	IRQ3/11 In Service 0 = No 1 = Yes	IRQ2/10 In Service 0 = No 1 = Yes	IRQ1/9 In Service 0 = No 1 = Yes	IRQ0/8 In Service 0 = No 1 = Yes
020h, 0A0h	Polled Mode Register OCW3[2] = 1 (read-only)	Interrupt Pending 0 = No, 1 = Yes	Not used				IRQ[2:0] - Number of highest priority interrupt that is pending		

4.6.3.3.3 Interrupt Controller Shadow Registers

Values written to the interrupt controller are not always directly readable in the AT architecture. However, the 82C465MV shadows these values as they are written so that they can be read back later through the configuration registers. Table 4-13 lists the correspondence of shadow indexes to the write-only registers in the interrupt controllers.

Table 4-13 Interrupt Controller Shadow Register Index Values

Register	INTC1 Index	INTC2 Index
ICW1	80h	88h
ICW2	81h	89h
ICW3	82h	8Ah
ICW4	83h	8Bh
OCW2	85h	8Dh
OCW3	86h	8Eh

4.6.3.4 DMA Controller Programming Registers

The IPC provides two direct memory access controllers (DMAC1 and DMAC2) and their associated memory mappers that are register compatible with AT-type systems. The registers of this logic module are listed below. These registers are accessed directly through the I/O subsystem (no index/data method is used). Each DMAC has four DMA channels. Channels 0-3 are in DMAC1, channels 4-7 in DMAC2. Table 4-14 and Table 4-15 list the register locations.

Table 4-14 DMA Address and Count Registers

Name	DMA Channel 0 Address	DMA Channel 1 Address	DMA Channel 2 Address	DMA Channel 3 Address	DMA Channel 4 Address	DMA Channel 5 Address	DMA Channel 6 Address	DMA Channel 7 Address
Memory Address Register	000h R/W	002h R/W	004h R/W	006h R/W	0C0h R/W	0C4h R/W	0C8h R/W	0CCh R/W
Count Register	001h R/W	003h R/W	005h R/W	007h R/W	0C2h R/W	0C6h R/W	0CAh R/W	0CEh R/W
Page Address Register	087h R/W	083h R/W	081h R/W	082h R/W	08Fh R/W	08Bh R/W	089h R/W	08Ah R/W

Table 4-15 DMA Control and Status Registers

Command	Function	Command port address for DMA Channels 0-3	Command port address for DMA Channels 5-7
Mode Register	Sets the function type for each channel. Group can be read back - see "Reset Mode Register Readback Counter" command	Read/write 00Bh	Read/write 0D6h
Status Register	Returns channel request and terminal count information	Read 008h	Read 0D0h
Command Register	Sets the DMAC configuration	Write 008h, Read 00Ah	Write 0D0h, Read 0D4h
Request Register	Makes a software DMA request	Read/write 009h	Read/write 0D2h
Mask Register	Enables or masks DMA transfers on selected channels	Read/write 00Fh	Read/write 0DEh
Temporary Register	Not used in AT-compatible design	Read 00Dh	Read 0DAh

DMAC1 Control and Status Bits

Port	Name	7	6	5	4	3	2	1	0
008h	DMAC1 Status Register	Channel 3 request pending 0 = No 1 = Yes	Channel 2 request pending 0 = No 1 = Yes	Channel 1 request pending 0 = No 1 = Yes	Channel 0 request pending 0 = No 1 = Yes	Channel 3 reached terminal count 0 = No 1 = Yes	Channel 2 reached terminal count 0 = No 1 = Yes	Channel 1 reached terminal count 0 = No 1 = Yes	Channel 0 reached terminal count 0 = No 1 = Yes
00Bh	DMAC1 Mode Register	Mode Select 00 = Demand 01 = Single 10 = Block 11 = Cascade		Address Count 0 = Increment 1 = Decrement	Auto-Initialize 0 = Disable 1 = Enable	Transfer Select 00 = Verify 01 = Memory Write 10 = Memory Read 11 = Reserved		Channel Select 00 = Channel 0 01 = Channel 1 10 = Channel 2 11 = Channel 3	
009h	DMAC1, DMA Request Register	Reserved. Write as 0.					Request 0 = Clear 1 = Set	Channel Select 00 = Channel 0 01 = Channel 1 10 = Channel 2 11 = Channel 3	

DMAC1 Control and Status Bits (cont.)

Port	Name	7	6	5	4	3	2	1	0
008h	DMAC1 Command Register	DACK Active Sense 0 = Low 1 = High	DRQ Active Sense 0 = High 1 = Low	Extended Write 0 = Disable 1 = Enable	Rotating Priority 0 = Disable 1 = Enable	Compressed Timing 0 = Disable 1 = Enable	DMAC Operation 0 = Enable 1 = Disable	Channel 0 Address Hold 0 = Disable 1 = Enable	Memory-to-Memory 0 = Disable 1 = Enable
00Fh	DMAC1 Mask Register	Reserved. Write as 0.				Channel 3 0=Unmasked 1 = Masked	Channel 2 0=Unmasked 1 = Masked	Channel 1 0=Unmasked 1 = Masked	Channel 0 0=Unmasked 1 = Masked

DMAC2 Control and Status Bits

Port	Name	7	6	5	4	3	2	1	0
0D0h	DMAC2 Status Register	Channel 7 request pending 0 = No 1 = Yes	Channel 6 request pending 0 = No 1 = Yes	Channel 5 request pending 0 = No 1 = Yes	Channel 4 request pending 0 = No 1 = Yes	Channel 7 reached terminal count 0 = No 1 = Yes	Channel 6 reached terminal count 0 = No 1 = Yes	Channel 5 reached terminal count 0 = No 1 = Yes	Channel 4 reached terminal count 0 = No 1 = Yes
0D6h	DMAC2 Mode Register	Mode Select 00 = Demand 01 = Single 10 = Block 11 = Cascade		Address Count 0 = Increment 1 = Decrement	Auto-Initialize 0 = Disable 1 = Enable	Transfer Select 00 = Verify 01 = Memory Write 10 = Memory Read 11 = Reserved		Channel Select 00 = Channel 4 01 = Channel 5 10 = Channel 6 11 = Channel 7	
0D2h	DMAC2, DMA Request Register	Reserved. Write as 0.					Request 0 = Clear 1 = Set	Channel Select 00 = Channel 4 01 = Channel 5 10 = Channel 6 11 = Channel 7	
0D0h	DMAC2 Command Register	DACK Active Sense 0 = Low 1 = High	DRQ Active Sense 0 = High 1 = Low	Extended Write 0 = Disable 1 = Enable	Rotating Priority 0 = Disable 1 = Enable	Compressed Timing 0 = Disable 1 = Enable	DMAC Operation 0 = Enable 1 = Disable	Channel 0 Address Hold 0 = Disable 1 = Enable	Memory-to-Memory 0 = Disable 1 = Enable
0DEh	DMAC2 Mask Register	Reserved. Write as 0.				Channel 7 0=Unmasked 1 = Masked	Channel 6 0=Unmasked 1 = Masked	Channel 5 0=Unmasked 1 = Masked	Channel 4 0=Unmasked 1 = Masked

Table 4-16 DMA Commands

Command	Function	Command port address for DMA Channels 0-3	Command port address for DMA Channels 5-7
Set Single Mask Bits Register	Sets or clears individual mask register bits without having to do a read/modify/write of the Mask Register	Write 00Ah: bits [1:0] select the channel, bit [2] selects the new mask bit value	Write 0D4h: bits [1:0] select the channel, bit [2] selects the new mask bit value
Clear Mask	Unmasks all DMA channels at once	Write any value to 00Eh	Write any value to 0DCh
Reset Mode Register Readback Counter	Resets the Mode Register Readback function to start at register 0. The next four Mode Register reads then return channels 0, 1, 2, and 3 for that DMAC	Read 00Eh (then read 00Bh four times to get the Mode Register values)	Read 0DCh (then read 0D6h four times to get the Mode Register values)
Master Clear	Clears all values, masks all channels, just like a hardware reset	Write any value to 00Dh	Write any value to 0DAh
Clear Byte Pointer Flip-Flop	Resets the byte pointer flip-flop so that the next byte access to a word-wide DMA register is to the low byte	Write any value to 00Ch	Write any value to 0D8h

Table 4-16 DMA Commands (cont.)

Command	Function	Command port address for DMA Channels 0-3	Command port address for DMA Channels 5-7
Set Byte Pointer Flip-Flop	Sets the byte pointer flip-flop so that the next byte access to a word-wide DMA register is to the high byte	Read 00Ch	Read 0D8h

4.6.3.5 Determining DMA Status Before Suspend

The 82C465MVA is the first chip in the OPTi 82C46x family to allow DMA transfer status to be determined before suspending operation. In this way, complete system context can be saved to disk and restored at any time, even to the point of being able to reload and restart DMA operations such as DMA-driven audio applications.

4.6.3.5.1 Stopping DMA Activity in SMM

On receiving a suspend request, SMM code may want to stop DMA and determine the current state of all DMA peripheral devices. This would be difficult to do by masking the channels, since the mask register must be read first to determine the channels that are unmasked and then masked. By time this read/modify/write cycle is completed, transfer completion on a channel may have caused one of the unmasked channels to be masked.

The 82C465MVA logic provides a DMA SMI enable, bit D6h[6], that traps to SMM on any DMA request without actually servicing the request. This same bit can be set from within SMM to stop any DMA in progress. The DMA will not restart until bit DDh[4] is written to 1 to clear the event. If another DMA transfer then occurs, it too will be blocked if bit D6h[6] is still set to 1.

Once DMA is stopped, SMM code can read the "in-progress" bits to quickly determine the state of transfers on each channel. These bits are set on the first DRQ after a channel is unmasked, and cleared by a TC on that channel. Once known, this information allows the code to decide how to properly save and then restore the state of each channel as described in the following sections.

DMA Progress Bits

Index	Name	7	6	5	4	3	2	1	0
D6h	PMU Control Register 10		DMA Trap PMI#28 SMI 0=Disable 1=Enable	DMAC1 Byte Pointer Flip-Flop Read Only 0=Cleared 1=Set					
84h	DMA In-Progress Register Read-only	Channel 7 DMA in progress 0=No 1=Possibly	Channel 6 DMA in progress 0=No 1=Possibly	Channel 5 DMA in progress 0=No 1=Possibly	DMAC2 Byte Pointer Flip-Flop Read Only. 0=Cleared 1=Set (MVA)	Channel 3 DMA in progress 0=No 1=Possibly	Channel 2 DMA in progress 0=No 1=Possibly	Channel 1 DMA in progress 0=No 1=Possibly	Channel 0 DMA in progress 0=No 1=Possibly
DDh	PMU SMI Source Register 4				PMI #28 - DMA 0=Clear 1=Active (MVA)				

4.6.3.5.2 DMA Register Read-Back Provisions

The 82C465MVA part provides the means of reading those DMA register settings that are normally not accessible in order to restore the state after powering down the chip.

Saving Count and Address Registers

On the 82C465MVA part, only the current address and count values can be read back. The base values are not shadowed and cannot be read directly. However, this does not prohibit saving and restoring base values. Using the count registers as an example, when the SMM suspend code starts to execute the possible states of the DMA transfer are:

Not yet started. The current count and the base count will be identical.

Completed on an autoinitialized channel. The base count will be restored to the current count and the two will be identical.

Completed on a non autoinitialized channel. The base count is meaningless and can be restored to the current count (which will be FFFFh after DMA completion).

In mid-transfer on an autoinitialized channel. The current count and base count will be different. SMM code must perform the following steps.

1. Read back the current count.
2. Set the channel to "block verify" mode and make a software request to start the transfer.
3. Read back the current count, which now will reflect the base count instead (at the end of the transfer the current count will be autoinitialized to the base count).

In mid-transfer on a non autoinitialized channel. The base count is meaningless. Only the current count needs to be read back.

For all these cases, the register values can be read back directly from the DMA controller registers to which the base values were originally written.

Saving Mode and Mask Register Contents

The IPC in the 82C465MVA part provides the means of reading back the mode and mask registers. Perform the following steps, involving system I/O ports (not 82C465MVA configuration registers).

1. Read I/O port 00Eh to reset the mode register readback counter for DMAC 1
2. Do four successive reads from I/O port 00Bh to retrieve mode registers 0-3
3. Read I/O port 00Fh to return channel 0-3 mask bits in bits 0-3
4. Read I/O port 0DCh to reset the mode register readback counter for DMAC 2
5. Do four successive reads from I/O port 0D6h to retrieve mode registers 4-7
6. Read I/O port 0DEh to return channel 4-7 mask bits in bits 0-3.

Determining Programming and Transfer Progress

Aside from the registers already listed, the 82C465MVA logic shadows the byte pointer flip-flop and provides special "in-progress" indicators for each channel. Using this information along with the readable DMAC register information, SMM suspend code can save the state of the DMAC as follows.

1. Read and save the DMA mode and mask registers from the PMU.
2. Read the "in-progress" bits to determine whether DMA could be taking place on any channel. These bits are set on the first DRQ after a channel is unmasked, and cleared by a TC on that channel.
3. Read the PMU byte pointer flip-flop setting for each DMAC.
4. Clear the byte pointer flip-flops by writing the appropriate DMAC registers.
5. Read and save all current count and current address values from the DMAC.

The 82C465MVA part can now be powered down.

Restoring Registers on Resume

On resuming operation, the DMAC register values can be restored as follows.

1. Restore the DMAC mode and mask register values.
2. Restore the DMAC count and address values.
3. If either of the saved PMU byte pointer flip-flop settings indicates that a flip-flop was set, perform one read from an appropriate DMAC count or address register to set the flip-flop to its original state.

4.6.3.6 LDEV# Sense Control

When the 82C465MV chip runs DMA to a local-bus device (usually the video controller), the chip generates ADS# instead of the CPU. During a DMA write cycle (I/O read, local-bus write), the chip waits to sample LDEV# and ADS# low at the same time before it enables its buffer to drive SD bus data back to the local bus. If the local bus is heavily loaded, the data may not be ready in time for the local-bus device to latch it.

The 82C465MVA part provides a program bit to select a different sampling method in the case of DMA to the local bus. When bit D6h[3]=0, sampling is as with the 82C465MV. When bit D6h[3]=1, the buffer enable control depends only on LDEV# and not on ADS#. Effectively, the sampling window occurs one clock earlier. Local-bus devices that decode LDEV# from address and status alone should have no problem meeting the early sample requirement of this setting.

LDEV# Sampling for DMA to Local Bus

Index	Name	7	6	5	4	3	2	1	0
D6h	PMU Control Register 10					Local-bus DMA LDEV# sampling 0=Normal 1=Sample one clock sooner (MVA)			

4.6.3.7 Type F DMA Support

Improved DMA transfer performance is available on a channel-by-channel basis for those devices capable of shorter ISA command pulses. Normally the 82C465MVB DMA cycle width is 6 AT clocks for the read command and 4 AT clocks for the write command. Enabling Type F DMA for a channel changes this timing as follows.

- For ISA DMA devices: Read command (IOR# or MEMR#) is 2 AT clocks, Write command (IOW# or MEMW#) is 1 AT clock.
- For CISA DMA devices: CMD# is 3 AT clocks; MEMR# or MEMW# is also 3 AT clocks.

Type F DMA is controlled through the EISA register scheme. Only the bits shown are supported.

Type F DMA Control

I/O Port	Name	7	6	5	4	3	2	1	0
40Bh (0-3) (MVB)	EISA DMA Extended Mode Register	Not implemented	Not implemented.	Cycle Timing 00=ISA-compatible 01=ISA-compatible 10=ISA-compatible 11=Type F		Not implemented.	Not implemented.	DMA Channel 00=Channel 0 01=Channel 1 10=Channel 2 11=Channel 3	
4D6h (4-7) (MVB)	EISA DMA Extended Mode Register	Not implemented	Not implemented.	Cycle Timing 00=ISA-compatible 01=ISA-compatible 10=ISA-compatible 11=Type F		Not implemented.	Not implemented.	DMA Channel 00=Reserved 01=Channel 5 10=Channel 6 11=Channel 7	

4.6.3.8 Timer Programming Registers

The IPC provides an 8254-type timer with three channels that is register compatible with AT-type systems. The registers of this logic module are listed below. These registers are accessed directly through the I/O subsystem (no index/data method is used). Table 4-17 lists the register locations.

Table 4-17 Timer Control and Status Registers

Name	Function	Port address for timer channel 0	Port address for timer channel 1	Port address for timer channel 2
Counter Registers Access	Used to write and read the word-wide count. Writes always program the base value. Reads return either the instantaneous count value or the latched count value.	040h	041h	042h
Counter Mode Command	Selects the operational mode for each timer counter.	Write 043h		
Counter Latch Command	Latches the count from the selected register for reading at the associated counter register access port.	Write 043h then read 040h, 041h, and/or 042h		
Readback Command	Selects whether count or status, or both, will be latched for subsequent reading at the associated counter register access port. If both are selected, status is returned first. This command can latch information from more than one counter at a time.	Write 043h then read 040h, 041h, and/or 042h		

Timer Control Bits

Port	Name	7	6	5	4	3	2	1	0
043h	Counter Mode Command (write-only)	Counter Select 00 = Counter 0 01 = Counter 1 10 = Counter 2 11 = Readback command (see below)		Counter Access 00 = Counter latch command (see below) 01 = Read/write LSB only 10 = Read/write MSB only 11 = Read/write LSB followed by MSB		Mode Select 000 = M0) Interrupt on terminal count 001 = M1) Hardware retrig. one-shot X10 = M2) Rate generator X11 = M3) Square wave generator 100 = M4) Software-triggered strobe 101 = M5) Hardware-triggered strobe			Count Mode Select 0 = 16-bit binary 1 = 4-decade BCD
043h	Counter Latch Command (write-only)	Counter Select 00 = Counter 0 01 = Counter 1 10 = Counter 2 11 = Illegal		Counter latch command = 00		Don't care			
043h	Readback Command (write-only)	Readback command = 11		Latch count 0 = Yes 1 = No	Latch status 0 = Yes 1 = No	Counter 2 Select 0 = Yes 1 = No	Counter 1 Select 0 = Yes 1 = No	Counter 0 Select 0 = Yes 1 = No	Reserved. Write as 0.
043h	Status Byte (read-only)	OUT signal status	Null Count - counter contents valid 0 = Yes 1 = No (being updated)	Return bits [5:0] written in Counter Mode Command (see above)					

4.6.3.8.1 Shadow Registers To Support Timer

Values written to the timer are not always directly readable in the AT architecture. However, the 82C465MV shadows these values as they are written so that they can be read back later through the configuration registers. The values from index 90h to 96h are valid only when a Counter Mode Command byte for the counter has been written to the timer register at I/O port 043h. Setting bits 043h[5:4] = 11 starts the sequence.

Index	Name	7	6	5	4	3	2	1	0
90h	Channel 0 Low Byte	Timer Channel 0 count low byte, A[7:0]							
91h	Channel 0 High Byte	Timer Channel 0 count high byte, A[15:8]							
92h	Channel 1 Low Byte	Timer Channel 1 count low byte, A[7:0]							
93h	Channel 1 High Byte	Timer Channel 1 count high byte, A[15:8]							
94h	Channel 2 Low Byte	Timer Channel 2 count low byte, A[7:0]							
95h	Channel 2 High Byte	Timer Channel 2 count high byte, A[15:8]							
96h	Write Counter High/Low Byte Latch	Unused	Unused	Channel 2 read LSB toggle bit	Channel 1 read LSB toggle bit	Channel 0 read LSB toggle bit	Channel 2 write LSB toggle bit	Channel 1 write LSB toggle bit	Channel 0 write LSB toggle bit

4.6.3.9 Writing/Reading I/O Port 070h

The AT architecture does not allow the readback of the NMI enable bit settings and the RTC index value written at I/O port 070h. However, the 82C465MV logic makes the NMI Enable bit setting, along with the last RTC index value written to I/O port 070h, available for reading in its shadow register set.

RTC Index Register - I/O Port 070h

Port	Name	7	6	5	4	3	2	1	0
070h	RTC Index Register write-only	NMI Enable 0 = Disable 1 = Enable	RTC/CMOS RAM Index						

4.6.3.9.1 RTC Index Shadow Register

This shadow register is read as a normal 82C465MV configuration register: write 98h to I/O port 022h followed immediately by an I/O read at I/O port 024h.

Index	Name	7	6	5	4	3	2	1	0
98h	RTC Index Shadow Register read-only	NMI Enable Setting	CMOS RAM Index Last Written						

4.6.3.10 Additional Floppy Support

The 82C465MVA part allows floppy register writes to be shadowed for easier management of power-down operations. Register writes to primary FDC port addresses 3F2h and 3F7h, and to secondary FDC port addresses 372h and 377h, are always copied to the 82C465MVA shadow registers.

In addition, the 82C465MVA logic provides a new register bit to select whether additional FDC port accesses should be monitored by DSK_ACCESS. On the original 82C465MV part, only port 3F5h is monitored. If the new bit is set to enable additional monitoring, DSK_ACCESS will monitor I/O reads and writes to all primary and secondary FDC port addresses. Note that this

feature can be used in conjunction with the new I/O port address registers at indexes D6h and D7h to determine which register access caused the trap.

Floppy Shadow and Control Registers

Index	Name	7	6	5	4	3	2	1	0
9Bh	3F2h +3F7h Shadow Register	Shadows 3F2h[7] "Mode Select" bit (MVA)	Shadows 3F7h[1] "Disk Type" bit 1 (MVA)	Shadows 3F2h[5] "Drive 2 Motor" bit (MVA)	Shadows 3F2h[4] "Drive 1 Motor" bit (MVA)	Shadows 3F2h[3] "DMA Enable" bit (MVA)	Shadows 3F2h[2] "Soft Reset" bit (MVA)	Shadows 3F7h[0] "Disk Type" bit 0 (MVA)	Shadows 3F2h[0] "Drive Select" bit (MVA)
9Ch	372h +377h Shadow Register	Shadows 372h[7] "Mode Select" bit (MVA)	Shadows 377h[1] "Disk Type" bit 1 (MVA)	Shadows 372h[5] "Drive 2 Motor" bit (MVA)	Shadows 372h[4] "Drive 1 Motor" bit (MVA)	Shadows 372h[3] "DMA Enable" bit (MVA)	Shadows 372h[2] "Soft Reset" bit (MVA)	Shadows 377h[0] "Disk Type" bit 0 (MVA)	Shadows 372h[0] "Drive Select" bit (MVA)
9D-9Eh	Reserved								
D6h	PMU Control Register 10	DSK_ ACCESS 0=3F5h only 1=All FDC ports (3F2,4,5,7h and 372,4,5,7h) (MVA)							

4.6.3.11 IRQ8 Polarity

The recognition of the IRQ8 interrupt can be inverted through bit 50h[5]. In the normal AT architecture, IRQ8 is active low and driven by an open-collector output of the RTC against a pull-up resistor. If the 82C465MV chip is used in conjunction with the 82C602 Notebook Companion chip, IRQ8 polarity should be set to active high.

PMU Control Register - Index 50h

Index	Name	7	6	5	4	3	2	1	0
50h	PMU Control Register 5			IRQ8 polarity 0 = Active low 1 = Active hi					

4.6.4 Integrated Local-Bus Enhanced IDE Interface

Enhanced IDE support through the local bus is available on the 82C465MV as a register-programmable option. Logic from the proven OPTi 82C611 local-bus IDE controller is used to incorporate this option. Note, however, that the write posting and read prefetching features of the separate 82C611 device are not supported by the 82C465MV chip.

4.6.4.1 Hardware Considerations

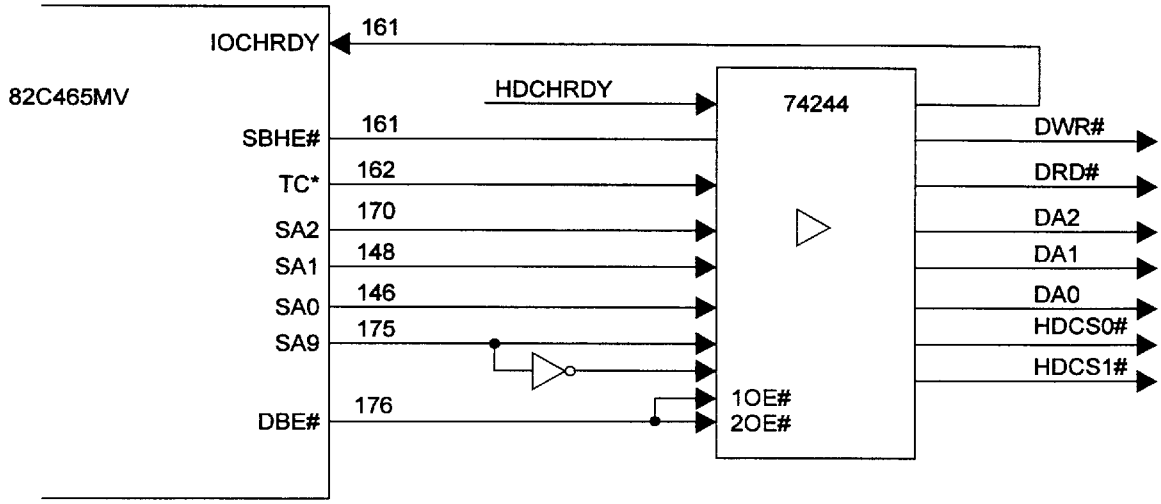
Local-bus IDE support requires seven pins. Six of the signals are shared signals that also go to their active state (from the perspective of the IDE) during non IDE cycles. Therefore, these signals must be qualified by DBE#; they cannot be connected directly to the IDE interface. The signals are defined as follows.

- Drive Read (DRD#) - provides the read strobe signal for the IDE drive; also switches the data buffer direction toward the SD bus during read cycles. The 82C465MV drives DRD# on the BALE line if Compact ISA is disabled and on TC is CISA is enabled. Refer to 4.6.5.
- Drive Write (DWR#) - provides the write strobe signal for the IDE drive. The 82C465MV drives DWR# on the SBHE# line.
- Address bits 1:0 (DA1:0) - provided directly from the 82C465MV SA1:0 signals.
- Address bits 2 and 9 (DA2 and DA9) - buffered version of CPU CA2 and CA9.
- Address bit 7 can be used for four drive support (MVB).
- Drive Buffer Enable (DBE#) - enables the buffer for control lines DRD# and DWR#, and address bits DA9, 2, 1, and 0 to the IDE drive, as well as the data bus buffers. DBE# replaces the TRIS# signal on pin 176.

The DRD# and DWR# signals and the other control signals are separated from the standard AT-bus I/O control signals to avoid the incompatibility that can occur when signals such as IOR# and IOW# are "short pulsed" as they would be for an IDE cycle. Short pulses on these lines can cause mis-operation on some AT peripheral devices, even if the read and write is not intended for those devices.

Figure 4-5 illustrates how the connections would typically be made.

Figure 4-5 Interface to Integrated IDE Controller



* DRD# comes out on BALE if Compact ISA interface is disabled. See Section 4.6.5.

4.6.4.2 Performance and Power

Enhanced IDE uses the SD bus for its data transfers, but does not use AT-bus transfers because of its dedicated DRD#, DWR#, and DBE# signals. Essentially, the local-bus IDE controller can run extremely short cycles because all timing aspects of the cycle are directly programmable to meet the capabilities of the drive being used.

The 82C465MV chipset implementation of local-bus IDE is designed to save power. The buffers to/from the IDE are tri-stated between cycles. Therefore, no power is wasted toggling the IDE data lines when the IDE is not in use.

An innovative method is used to handle port 3F7h bit 7 from the external floppy controller. Bit 7 from the FDC must be attached to bits [6:0] from the IDE controller whenever an I/O read of 3F7h takes place, and normally requires a separate IDED7 line from the IDE. On the 82C465MV, two separate cycles take place whenever the an I/O read from 3F7h takes place. First, the local-bus IDE cycle is run using DRD# and DBE#. Then an AT-bus I/O cycle is run. When the 82C465MV returns a value to the CPU, it provides bits [6:0] read from the IDE and bit 7 from the AT bus.

4.6.4.3 Signal Connection

With the IDE interface disabled, the chip pin functions remain compatible with those of the 82C463MV. When the IDE interface is enabled, the only function actually lost is TRIS#. TRIS# serves only to indicate that the system is in suspend mode, a status which can also be derived from the PPWR0-1 outputs. The various AT-bus signals that are used for IDE command and chip select lines are signals that would normally require qualification by other signals; toggled by themselves, they should not cause any action or conflicts on the AT bus.

4.6.4.4 DBE (TRIS) Polarity

The DBE# line (or the TRIS# line if the IDE interface is not enabled through bit ACh[3]) is normally an active low signal. Since this polarity may not be correct for all connected devices, the polarity can be inverted through a strap option. The inversion is valid for either the TRIS# or the DBE# function, changing them to TRIS and DBE respectively. The options available are as follows.

- Normal active-low sense, for DBE# (TRIS#):
  Do not strap TRIS# at reset (internal pull-up selects option)
- Active-high sense, for DBE (TRIS):
  Pull TRIS# low at reset (can be strapped permanently with 10KΩ to ground)

4.6.4.5 Programming

The controller can be enabled to respond to I/O accesses either in the 1F0-1F7h and 3F6-3F7h range, or in the 170-177h and 376-377h range, but never to both. I/O port references in this document list both ranges, but only one range can ever be active at a time (always selected by bit ACh[2] regardless of other mode settings).

There are two ways to program operation of the local bus IDE controller. "Basic" timing is the fixed timing selection available through bits ACh[7:4]. "Enhanced" timing refers to the precise control provided through the 611 register set.

4.6.4.5.1 Easy Programming Method (Basic)

Use bits ACh[7:4] to select the CPU speed and the mode of operation. The IDE controller logic will generate commands for the correct number of CPU clocks to approximate the selected mode timing.

IDE Controller Configuration

Index	Name	7	6	5	4	3	2	1	0
ACh	IDE Interface Config. Register	Chipset Input Clock Frequency 00 = 50MHz 01 = 40MHz 10 = 33MHz 11 = 20/25MHz		IDE Mode / Minimum Cycle Duration 00 = Mode 0 / 600ns 01 = Mode 1 / 383ns 10 = Mode 2 / 240ns 11 = Mode 3 / 180ns		IDE Interface Enable 0=Disable 1=Enable	IDE Port Address Select 0=1F0-7h, 3F6-7h 1=170-7h, 376-7h	3F7h[6:0] Source 0=local IDE 1=AT bus	Reserved. Write as 0.

IDE Interface Enable bit ACh[3] - When the IDE interface is disabled, TRIS# is available. When the IDE interface is enabled, pin 176 (TRIS#) becomes DBE# and TRIS# is no longer available. PPWR0-1 in their auto-toggle mode can perform essentially the same function as TRIS#.

Port 3F7 Decode Disable bit ACh[1] - prevents port 3F7 reads from being combined with IDE controller reads in situations where this arrangement causes problems. When ACh[1] = 0, 3F7h[7] comes from the AT bus; 3F7h[6:0] (or 377h[6:0] if bit 2 = 1) come from local-bus IDE. When ACh[1] = 1, 3F7h[7:0] come from the AT bus.

The setting in bits ACh[7:4] will select cycle timings according to the following scheme.

Table 4-18 Automatic Cycle Settings Available Through Bits ACh[7:4]

Bits ACh[7:4]	Expected Input Clock Frequency (MHz)	Setup Time (clocks)	Command Pulse (clocks)	Recovery Time (clocks)	Maximum Cycle Time (clocks)
0000	50	4	9	17	30
0001		3	7	10	20
0010		2	6	4	12
0011		2	5	2	9
8-bit		--	15	14	31

Bits ACh[7:4]	Expected Input Clock Frequency (MHz)	Setup Time (clocks)	Command Pulse (clocks)	Recovery Time (clocks)	Maximum Cycle Time (clocks)
0100	40	3	7	14	24
0101		3	6	7	16
0110		2	5	3	10
0111		2	4	2	8
8-bit		--	12	11	25
1000	33	3	6	11	20
1001		2	5	6	13
1010		2	4	2	8
1011		1	3	2	6
8-bit		--	10	9	20
1100	25	2	5	8	15
1101		2	4	4	10
1110		1	3	2	6
1111		1	2	2	5
8-bit		--	8	6	15

4.6.4.5.2 Precise Programming Method (Enhanced)

More precise control of IDE operation is available by using the "611" register set, so called because it is register-compatible with the register set used in the OPTi 82C611 stand-alone local-bus IDE controller. This register set is hidden behind the IDE drive I/O ports and is not normally accessible.

Timing 0 and Timing 1

The 611 register set supports two separate IDE drives on a single cable with independent timing requirements. Application software writes bit 1F6h[4] or 176h[4] to select between drive 0 and 1 on the cable. The 611 core tracks writes to this I/O port and switches its timing. The correspondence is ***not*** necessarily direct, however, between Drive 0 and Timing 0, for example. Each drive can select its timing from two sources, which are themselves selectable according to bit 1F3/173h[7].

"Basic" choices when bit 1F3/173h[7]=0:

1. The "easy method" timings from bits ACh[7:4]
2. Timing 0

"Enhanced" choices when bit 1F3/173h[7]=1:

1. Timing 1
2. Timing 0

The basic or enhanced timing choices are made as follows. Internal to the 82C465MVA, there is a single 611 register set. The 611 registers are available only when enabled through a special unlocking procedure. Timing choices are made according to the tables below through bits 1F3/173h[2-3] for drives 0 and 1, respectively. Once all programming is complete, the 611 register set again becomes hidden. From then on, accesses to the IDE port bit 1F6/176h[4] are tracked to determine the timing to use.

Table 4-19 82C465MVA Operation with Primary I/O Range Selected

ACh[2]	I/O Range	SA7 Value	IDE Drive Setting (IDE Head/Drive Select Register)	Drive Selected	Timing Selected by hidden 611 register bit
0	Primary	1	1F6h[4]=0	0	1F3h[2]
	1F0-7h, 3F6-7h		1F6h[4]=1	1	1F3h[3]

Table 4-20 82C465MVA Operation with Secondary I/O Range Selected

ACh[2]	I/O Range	SA7 Value	IDE Drive Setting (IDE Head/Drive Select Register)	Drive Selected	Timing Selected by hidden 611 register bit
1	Secondary	0	176h[4]=0	0	173h[2]
	170-7h, 376-7h		176h[4]=1	1	173h[3]

Subset Registers for Timing 0 and Timing 1

Within the single 611 register set, there are two subsets of registers to program the Read Pulse Width, Write Pulse Width, Read Recovery Time, and Write Recovery Time separately for Timing 0 and Timing 1. The register set loaded by writing to 1F0/1h or 170/1h is selected by bit 1F6/176h[0]. Setting this bit to 0 allows writes to 1F0/1h or 170/1h to program Timing 0; setting the bit to 1 allows programming of Timing 1.

"611" Register Set

Port	Name	7	6	5	4	3	2	1	0
1F0h 170h	Read Cycle Timing Register	Read Pulse Width - The value written to these bits, plus 1, selects the DRD# pulse width for a read from the 16-bit data register.*				Read Recovery Time - The value written to these bits, plus 2, determines the minimum time allowed between the end of DRD# and the start of the next IDE chip select (HDCS0-1#, derived from TC).*			
1F1h 171h	Write Cycle Timing Register	Write Pulse Width - The value written to these bits, plus 1, selects the DWR# pulse width for a write to the 16-bit data register *				Write Recovery Time - The value written to these bits, plus 2, determines the minimum time allowed between the end of DWR# and the start of the next IDE chip select (HDCS0-1#, derived from TC).*			
1F2h 172h	ID Register Write-only	82C611 Register Access 0x=Enable 10=Two 1F1/171h reads to enable (default) 11=Permanently disable		Reserved. Write as 0.				ID bits - These bits must always be written as 11	
1F3h 173h	Control Register 1	Timing Register Value Select 0=Basic 1=Enhanced	Reserved. Write as 0.			Drive 1 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0	Drive 0 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0	Reserved. Write as 0.	IDE Operation 0=Disable 1=Enable
1F5h 175h	Status Register Read-only	ISA 3F7h bit [7] status	Revision number - Returns 00 on present silicon revision.		IRQ14 status	Configuration register bits ACh[5:4] setting		Configuration register bits ACh[7:6] setting	
1F6h 176h	Secondary Setup and Hold Timing Register	Reserved. Write as 0.		Address setup time - The value written, plus 1, selects the address setup time.*		Channel ready hold time - The value written, plus 2, selects the delay for setting the command pulse inactive from when the controller sees IOCHRDY go high.*			Timing Register Load Select 0=Timing 0 1=Timing 1

* The value indicates the width in terms of FBCLKIN cycles.

4.6.4.5.3 Step-by-Step Programming Procedure

Each phase of 611 programming is described below. Before accessing the 611 register set, the basic interface must be set up as follows.

1. Enable the external IDE controller interface by setting bit ACh[3]=1.
2. Select the I/O range to be used through register ACh[2]. For the purposes of this example, bit ACh[2] is set to 0 to select the 1F0-7h and 3F6-7h range. If the 170-7h and 376-7h range is selected instead, use the x7x port instead of the xFx port for each of the following steps.
3. Use bit ACh[1] to select whether 3F7h accesses will be directed to the IDE drive. If enabled, only bits [6:0] will come from the local bus IDE interface; bit 7 always comes from the ISA bus because it belongs to the floppy disk controller (if present).

The basic IDE interface is now available. Bits ACh[7:4] can be used to select the fixed timings listed above. If these fixed timings are sufficient, there is no need to use the 611 register set.

Enabling Access to 611 Register Set

The 611 register set must be enabled through a very specific procedure.

1. Perform a **word** read of 1F1h two times. This operation makes the 611 register set accessible for the next I/O operation.
2. Write 00000011b (03h) to port 1F2h. This programming keeps the 611 registers accessible indefinitely.

The 611 register set is now accessible. No IDE operation can take place as long as the register set access is enabled.

Setting Up Enhanced 611 Timing

Once the 611 register set is unlocked, Timing 0 and Timing 1 can be programmed.

1. Write x1F6h with bit 0=0 to be able to program Timing 0.
2. Program the upper and lower nibbles of 1F0h and 1F1h with the correct values for Timing 0. The read and write pulse widths can be independently programmed to be as short as 1 clock, while the recovery time for these cycles can be as short as 2 clocks.
3. Write 1F6h with bit 0=1 to be able to program Timing 1.
4. Program 1F0h and 1F1h with the correct values for Timing 1.
5. Write 1F6h with bits [5:1] set to select the required address setup time and IOCHRDY recovery time. These values apply to both Timing 0 and Timing 1. Bit 0 can be left in any state.

The timing sets are now programmed. The next step is to assign one of the timing sets to each drive.

Associating Timing with Each Drive

Register 1F3h selects timing options for the drives, and is the last step necessary before "hiding" the 611 register set and enabling 611 operation. Prepare a programming byte in which:

1. Bit [7]=1 to enable Enhanced timing, thus allowing both drives to select between the precise timing sets Timing 0 and Timing 1.
2. Bit [2] selects Timing 0 or Timing 1 for drive 0, and bit [3] does the same for drive 1.
3. Bit [0]=1 to enable all of the programming established to this point.

Set all other bits to 0, and write this value to 1F3h.

Enabling IDE Operation and Hiding 611 Register Set

The 611 register set must be hidden from standard access before IDE operation can begin. Two options are available.

- Write 1F2h with 11000011b (C3h) to disable 611 register access and fully enable IDE operation, and also prevent any future access to the 611 register set until the next hardware reset.

- Write 1F2h with 10000011b (83h) to disable 611 register access and fully enable IDE operation, but leave open the future possibility of accessing the 611 register set by restarting this whole procedure.

Application software can now control the drive selection and the timing with which it will be accessed through bit 1F6h[4].

4.6.4.6 Four-Drive IDE Support

The 82C465MV and 82C465MVA parts provide local bus IDE controller logic that supports two separate IDE drives with independent timing requirements. However, both drives must be on a single cable that responds either to 1F0-7h and 3F6-7h (main I/O range) accesses or to 170-7h and 376-7h (auxiliary I/O range) accesses. Writing bit 1F6h[4] (main) or 176h[4] (auxiliary) selects between drive 0 and 1 in the selected range. Internally, there is only a single register set that is selected by SA7 according to the setting of bit ACh[2].

82C465MVA Operation with Primary I/O Range Selected

ACh[2]	I/O Range	IDE Drive Setting	Drive Selected	Timing Selected by
0	Primary	1F6h[4]=0	0	1F3h[2]
	1F0-7h, 3F6-7h	1F6h[4]=1	1	1F3h[3]

82C465MVA Operation with Secondary I/O Range Selected

ACh[2]	I/O Range	IDE Drive Setting	Drive Selected	Timing Selected by
0	Primary	1F6h[4]=0	0	1F3h[2]
	1F0-7h, 3F6-7h	1F6h[4]=1	1	1F3h[3]

The 82C465MVB part maintains backward compatibility with its predecessors, but allows both the primary and secondary I/O ranges to be claimed by the IDE controller; bit ACh[2] is ignored in this mode. The DBE# signal must then be qualified by SA7 (with external logic) to select between the two cables. There is still just a single set of internal registers that are common to both the 1Fx/3Fx and 17x/37x addresses, with the exception of bits 1F3h[3:2] and 173h[3:2], which are separate. In this way, either of the timing sets 0 and 1 can be programmed individually for any of the four drives. The ACh[5:4] setting can also be used.

Setting bit 3Fh[5]=1 enables four IDE drive support mode.

82C465MVB Operation with Four Drive Support Selected

I/O Range	IDE Drive Setting	Drive	SA7	Timing Selected by
Main Cable	1F6h[4]=0	0	1	1F3h[2]
1F0-7h, 3F6-7h	1F6h[4]=1	1		1F3h[3]
Auxiliary Cable	176h[4]=0	0	0	173h[2]
170-7h, 376-7h	176h[4]=1	1		173h[3]

4.6.4.6.1 Performance Improvement

Bits 1F3h/173h[5:4] allow extra control over the register settings normally available at 1F0-1h/170-1h. These bits are not controlled by bit 3Fh[5].

Four Drive IDE Control

Index	Name	7	6	5	4	3	2	1	0
3Fh	Misc Control Register			Four IDE Drive Support 0=Disable 1=Enable (MVB)					

Four Drive IDE Control (cont.)

Index	Name	7	6	5	4	3	2	1	0
1F3h (MVB)	Primary Cable IDE Control Register 1	Timing Register Value Select 0=Basic 1= Enhanced	Reserved. Write as 0.	Timing 1 Performance 0=465MV-compatible 1=Reduce cmd. pulse 1/2 clock, recovery time 1/2 clock	Timing 0 Performance 0=465MV-compatible 1=Reduce cmd. pulse 1/2 clock, recovery time 1/2 clock	Primary Cable Drive 1 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0	Primary Cable Drive 0 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0	Reserved. Write as 0.	IDE Operation 0=Disable 1=Enable
173h (MVB)	Secondary Cable IDE Control Register 1					Secondary Cable Drive 1 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0	Secondary Cable Drive 0 Timing Select -Basic: 0=ACh[5:4] 1=Timing 0 -Enhanced: 0=Timing 1 1=Timing 0		

4.6.5 Compact ISA Interface

The 82C465MVB chipset incorporates the OPTi Compact ISA (CISA) interface. This interface allows connection of any Compact ISA peripheral device, such as the OPTi 82C852 PCMCIA Controller. The Compact ISA Specification is a separate document that describes the interface in detail.

The Compact ISA implementation must deal with certain issues that are specific to the interface architecture.

- ATCLK cannot be stopped without a specific stop clock cycle, since CISA depends on clock edges to transfer interrupts. The 82C465MVB can be programmed to generate this stop clock cycle, both automatically and manually.
- The CISA interface generates an AT Backoff (ATB#) signal to the 465MVB to make an interrupt or DMA request. The CISA interface is required to backoff any ISA cycle it has already started as long as it has not yet asserted ALE. ATB# will come, at latest, one-half ATCLK before ALE# would be asserted. Once ATB# is asserted, the 82C465MVB must inhibit all DMA activity and must prevent an EOI command to the interrupt controller from taking effect until ATB# is de-asserted and the new DRQ/IRQ states are latched in.
- The 82C465MVB Compact ISA involves two mandatory signals and one optional signal.
  - CMD# (O) - Command, generated by the 82C465MVB to run CISA cycles. CMD# is on pin 173 and replaces the PIO1 function on the 82C465MVB part when the CISA interface is enabled. To eliminate the need for an external keeper resistor, the 82C465MVB implements a weak pull-up on this pin until its programming registers are written. A write to register 57h disables the pull-up resistor. Therefore, software should enable CMD# before writing register 57h.
  - SEL#/ATB# (I) - CISA peripheral device "selected" handshake input during AT cycles; AT backoff request between cycles; clock restart request during idle mode. SEL#/ATB# is on pin 186 and replaces the RI pin on the 82C465MVB part when the CISA interface is enabled. The CISA interface requires a pull-up resistor on this line, which is automatically enabled when the SEL#/ATB# function is selected.
  - CDIR (O) - Optional CISA buffer direction signal. For desktop-type designs where the CISA signals are buffered on the motherboard to connect through a long ribbon cable to 82C852 PCMCIA Controller(s). CDIR is on pin 78 and replaces the RAS4# function.

The Compact ISA Control Registers F8h, F9h, and FAh enable the interface and control various features.

NOTE The Compact ISA interface uses the ALE signal for all its cycles. The 82C465MV and 82C465MVA parts also use ALE as the DRD# signal for local bus IDE. Therefore, when CISA is enabled, the DRD# signal moves to TC from ALE so that ALE will not toggle except on ISA/CISA cycles. Use SA9 instead of TC on the IDE buffer to generate CS0# and CS1#.

Compact ISA Control Registers

Index	Name	7	6	5	4	3	2	1	0
F8h	Compact ISA Control Register 1 (MVB)	Inhibit MRD# and MWR# if SEL# asserted on memory cycle 0=No 1=Yes	Inhibit MRD# and MWR# if SEL# asserted on DMA cycle 0=No 1=Yes	Inhibit IORD# and IOWR# if SEL# asserted on I/O cycle 0=No 1=Yes	IRQ15 Assignment 0=IRQ15 1=RI	Reserved	Fast CISA memory cycle 0=Disable (ISA#=0) 1=Enable (ISA#=1)	CDIR Pin (pin 78) 0=RAS4# 1=CDIR	Compact ISA Interface (reassigns pins 173, 186) 0=Disable 1=Enable
F9h	Compact ISA Control Register 2 (MVB)	SPKD Signal Driving 0=Always, per AT spec. 1=Synchronously, per CISA spec.	End-of-Interrupt Hold - Delays 8259 recognition of EOI command to prevent false interrupts. 00=None 01=1 ATCLK 10=2 ATCLKs 11=3 ATCLKs		Stop Clock Count bits CC[2:0] - Stop clock cycle indication to CISA devices of how many ATCLKs to expect before the clock will stop. 000=Reserved 001=1 ATCLK (default) .. 111=7 ATCLKs			Generate CISA Stop Clock Cycle (if not already stopped): 00=Never 01=On STPCLK# cycles to the CPU (hardware) 10=Immediately (software) 11=Reserved	

Compact ISA Interface Enable bit - provides master control over whole interface and enables reassignment on pins 173 and 186 to support CISA. If this bit is 0, all Compact ISA functions are disabled and no address strobing occurs on the SD bus. No other Compact ISA register bits should be set when F8h[0]=0.

CDIR Pin Enable bit - selects whether pin 78 will be reassigned as CDIR to control CISA cable driver direction.

IRQ15 Assignment - reassigns IRQ15 from the PCMCIA slot so that it can generate a Ring Indicator (RI) SMI instead.

Inhibit Commands if SEL# Asserted - These bits control whether commands will be hidden from ISA bus peripheral devices if the cycle is claimed by a CISA device. The feature allows devices that use the same memory or I/O space to avoid conflict with each other; CISA devices always preempt ISA devices. A separate bit is provided for memory signals during DMA, which would allow fly-by transfers to function between a PCMCIA DMA card and an ISA memory device.

SPKD Signal Driving - selects the CISA scheme for shared audio outputs. Refer to the CISA specification for complete information.

4.6.5.1 CISA Stop Clock Cycle Generation

Bits F9h[1:0] enable the 82C465MVB to generate the Stop Clock Broadcast cycle on the CISA bus, after which it can stop the AT clock. There are two methods of generating a CISA stop clock cycle: hardware-controlled and software-controlled.

4.6.5.1.1 Hardware CISA Stop-Clock Control

Hardware-controlled CISA stop clock cycle generation occurs automatically, if ATCLK has not been stopped already, whenever bits F9h[1:0]=01 and the 82C465MVB chip receives a stop grant cycle (or STPGNT# signal such as SUSPA#) from the CPU. The chipset generates a stop request to the CPU when changing CPU speeds or stopping the CPU clock; the CPU responds with a stop grant.

When F9h[1:0]=01 to enable automatic stop clock cycle generation on CISA, address phase 1 of each CISA cycle will not be generated until the cycle is decoded to be an ISA cycle. The logic adds in one extra AT clock before the cycle starts to properly start the CISA interface. Inhibition of CISA phase 1 generation saves power by avoiding unnecessary toggling on the MAD bus.

When F9h[1:0]=00 to disable hardware stop clock mode, the 82C465MVB logic drives address phase 1 of each CISA cycle as soon as it detects ADS# active. In this mode, there is no AT clock start-up delay. Software stop-clock control can still be used to stop the clock and save power.

4.6.5.1.2 Software CISA Stop-Clock Control

Software-controlled CISA stop clock cycle generation occurs only when bits F9h[1:0] are written to 10. A CISA stop clock cycle is forced onto the CISA bus. Whenever bits F9h[1:0]=10, the bits F9h[7:2] written to this register are ignored so no "read/modify/write" procedure is required. This cycle is generated only once; the bits then revert to their previous setting (00 or 01).

Stop Clock Count bits CC[2:0] - indicate to CISA devices how many ATCLKs to expect before the clock will stop. The default setting of one ATCLK is correct for most applications.

4.6.5.2 Configuration Cycle Generation

The 82C465MVB part can be programmed to generate one CISA Configuration Cycle, the Stop Clock Broadcast cycle, automatically after a period of inactivity. In order to provide for future Configuration Cycle possibilities, the 82C465MVB CISA interface also includes a generic command generation scheme. This scheme takes advantage of the Scratchpad registers already present in the 82C465 series parts, and does not prevent their continued use as Scratchpad registers. They must be reprogrammed only in order to send out a Configuration Cycle.

To generate a Configuration Cycle:

1. Load the phase 1 word in registers 6C-6Dh.
2. Load the phase 2 word in registers 6E-6Fh.
3. Load the data phase word in registers 52-53h.
4. Write bit FAh[0]=1 to run the cycle.

The CISA interface will generate the desired Configuration Cycle. The cycle will always be a Broadcast (write) cycle, since there is no inherent means of receiving information back from the Configuration Cycle. Whenever bit FAh[0]=1, the bits FAh[7:1] written to this register are ignored so no "read/modify/write" procedure is required. Bit FAh[0] is automatically cleared to 0 after the cycle runs.

4.6.5.3 Driveback Cycle Handling

Normally the 82C465MVB will transfer the IRQ and DRQ information of an IRQ/DRQ Driveback Cycle to the interrupt and DMA controllers. However, bit FAh[2] allows driveback cycle information to simply be latched and an SMI generated. In this way, SMM code can determine how (or whether) to deal with the changed IRQ or DMA status. The information is latched in the new Scratchpad registers 7-10. These new registers can be used for general purpose storage if Compact ISA is disabled. PMI#36 is generated for this event, and is read/cleared through bit EAh[7].

CISA Cycle Generation Registers

Index	Name	7	6	5	4	3	2	1	0
52h	Scratchpad Register 1	General purpose storage byte For CISA Configuration Cycles: Data phase information, low byte (MVB)							
53h	Scratchpad Register 2	General purpose storage byte For CISA Configuration Cycles: Data phase information, high byte (MVB)							
6Ch	Scratchpad Register 3	General purpose storage byte For CISA Configuration Cycles: Address phase 1 information, low byte (MVB)							
6Dh	Scratchpad Register 4	General purpose storage byte For CISA Configuration Cycles: Address phase 1 information, high byte (MVB)							
6Eh	Scratchpad Register 5	General purpose storage byte For CISA Configuration Cycles: Address phase 2 information, low byte (MVB)							
6Fh	Scratchpad Register 6	General purpose storage byte For CISA Configuration Cycles: Address phase 2 information, high byte (MVB)							
FCh	Scratchpad Register 7	General purpose storage byte For CISA Driveback Cycle: IRQ phase information, low byte (read-only) (MVB)							
FDh	Scratchpad Register 8	General purpose storage byte For CISA Driveback Cycle: IRQ phase information, high byte (read-only) (MVB)							
FEh	Scratchpad Register 9	General purpose storage byte For CISA Driveback Cycle. DRQ phase information, low byte (read-only) (MVB)							
FFh	Scratchpad Register 10	General purpose storage byte For CISA Driveback Cycle: DRQ phase information, high byte (read-only) (MVB)							

CISA Cycle Generation Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
FAh	Compact ISA Control Register 3 (MVB)			Reassign EPMI3 as RI 0=No 1=Yes Use in case RI is assigned as SEL#/ATB#	Reassign EPMI4 as IOCHCK# 0=No 1=Yes Use in case IOCHCK# is assigned as KBCRSTIN	Resume from Suspend on SEL#/ATB# low 0=Disabled 1=Enabled	CMD# State during Suspend 0=Driven inactive (high) 1=Driven low	Driveback Cycle Handling 0=Pass DRQs and IRQs 1=Latch info and generate SMI	Configura-tion Cycle Generation 0=No action 1=Run cycle using Scratchpad
EAh	PMU Source Register 5	IRQ/DRQ Driveback Trap PMI#36 0=Inactive 1=Active Write 1 to clear (MVB)							
6Bh	Resume Source Register					CISA SEL#/ ATB# low caused resume read-only 0=No 1=Yes			

CMD# State During Suspend - If the CISA bus devices are to be powered down during suspend mode, setting bit FAh[2]=1 drives the CMD# line low with the same timing as the PPWR0-1 lines so that there is no current leakage path.

Resume from Suspend on SEL#/ATB# low - Setting bit FAh[3]=1 allows CISA devices in stop clock mode to resume system operation by generating an interrupt. During normal operation when CISA devices are in stop clock mode, the SEL#/ATB# line acts as a CLKRUN# signal. This bit also allows the same signal to act as RSM#.

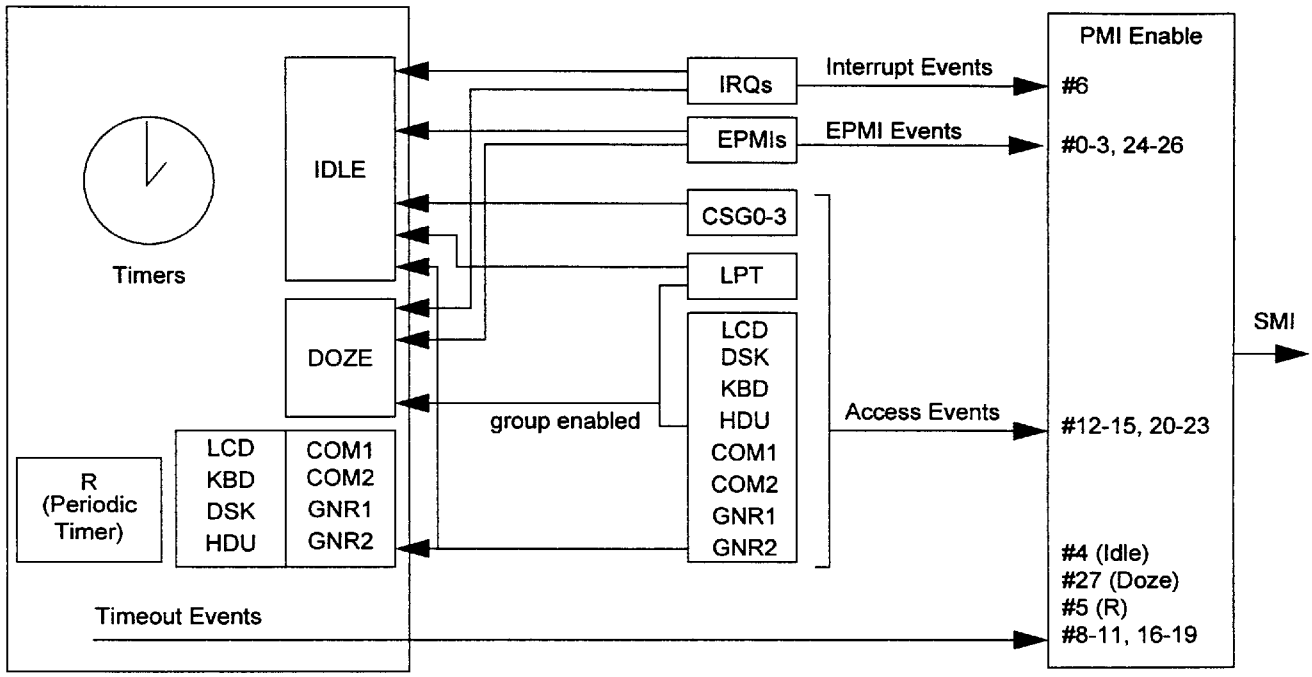
IRQ/DRQ Driveback Trap - If bit FAh[1]=1 and an IRQ/DRQ driveback cycle occurs, bit EAh[7] indicates that the SMI was caused by the interrupt driveback. No more IRQ driveback cycles will be serviced until this PMI is cleared by writing EAh[7]=1.

CISA SEL/ATB# Low Caused Resume - If bit FAh[3]=1 to allow resume from SEL#/ATB#, bit 6Bh[3] reads 1 to identify the resume source as CISA.

4.7 Power Management Unit

The 82C465MV provides a large amount of programmable logic for managing system power control on the most precise of levels. The basic concepts of the 82C465MV power management scheme involve activity monitoring through timeout events, access events, reload events, EPMI events, and interrupt events. These concepts are illustrated in Figure 4-6 and described in detail in the following sections.

Figure 4-6 Activity Monitoring Block Diagram



4.7.1 Activity Monitoring

Activity monitoring is based on timeouts of countdown timers, the events that can be enabled to reload the timers and delay a timeout, general access events, and the system management interrupts that can be generated in all cases.

4.7.1.1 Timers

The eleven 82C465MV timer registers all have **_TIMER** appended to their name.

- The **IDLE_TIMER** times long periods of inactivity across all selected system peripherals to determine, for example, when a full power down (called *suspend mode*) is appropriate.
- The **DOZE_TIMER** times short inactivity intervals (between keystrokes, for example) to put the system in an intermediate power-saving state called *doze mode*.
- The **R_TIMER** generates a periodic interrupt to allow system management code to poll for activity.
- Eight peripheral activity timers are available to monitor activity on specific peripheral devices so that system management software can put each peripheral device individually into a low power mode while the rest of the system continues to operate.

Simply loading the timer with a countdown value presets the timers. Then, the next access or interrupt event starts them counting down. A "dummy" access is needed in most cases to start the timer counting.

As each timer is clocked by its programmed source, it counts down to a *timeout* (zero) which generates a *power management interrupt* (PMI). The timeout PMI can, in turn, be enabled to generate a system management interrupt (SMI) on the SMI# line that goes from the 82C465MV to the CPU to switch it into system management mode (SMM).

4.7.1.2 Events

Each timer has one or more **events** that can reload it with its original value, holding off the timeout. The events can be: access events, those that are caused by CPU access to a certain I/O and/or memory range associated with that timer; or interrupt events, triggered by AT-bus IRQ events or special external power management inputs (EPMIs).

All events can be enabled individually to generate a PMI; access events generate separately numbered PMIs, while interrupt events are combined into a single PMI (PMI#6).

As opposed to timeout-caused PMIs, event PMIs can be enabled to:

- Reload the timer(s) and, if needed, restore the system clocks speed
- Generate an SMI
- Do both.

Because of the flexibility of the 82C465MV power management logic, the interaction among these mechanisms can become complex. It is important to keep in mind the basic goal of the logic in order to deal with power management effectively.

4.7.2 Timers

The 82C465MV logic implements eleven distinct timer circuits. Each timer has a clocking source associated with it. For all but the DOZE_TIMER these are named **SQW0**, **SQW1**, **SQW2**, or **SQW3**; the DOZE_TIMER circuit works differently than the rest and is described separately in the "Doze Mode" section of this document. Table 4-21 shows the frequencies that can be applied to the rest of the _TIMER counters.

The SQW0-3 timings are based on the SQWIN input to the 82C465MV logic, which can be either 32KHz or 128KHz as selected by bit 40h[3]. Bit 40h[6] provides a secondary range of time intervals and applies globally to all SQW0-3 selections.

Timer Control Bits

Index	Name	7	6	5	4	3	2	1	0
40h	PMU Control Register 1		Global timer divide 0 = div. by 1 1 = div. by 4			SQWIN frequency 0 = 32KHz 1 = 128KHz			

Table 4-21 lists the range of timeout delay that can be achieved by selecting each SWQx+bit 40h[6] combination.

Table 4-21 Time Interval Choices Applicable to _TIMER Settings

Choice	Bits	Bit 40h[6] = 0 - No base clock divisor			Bit 40h[6] = 1 - Divide base clock by 4		
		Frequency	Decrement timer every:	Maximum Delay	Frequency	Decrement timer every:	Maximum Delay
SQW0	00	32768Hz	30.5us	7.81ms	8.192KHz	0.122ms	31.25ms
SQW1	01	512Hz	1.95ms	0.5s	128Hz	7.8ms	2s
SQW2	10	16Hz	62.5ms	16s	4Hz	0.25s	64s
SQW3	11	0.5Hz	2s	8.5 min.	0.125Hz	8s	34 min.

The register bit locations for each timer are shown below. The timer source is selected by bit combinations as follows: 00 = SQW0, 01 = SQW1, 10 = SQW2, and 11 = SQW3.

Timer Clock Source Selection Registers

Index	Name	7	6	5	4	3	2	1	0
42h	Clock Source Register 1	Clock source for GNR_TIMER		Clock source for KBD_TIMER		Clock source for DSK_TIMER		Clock source for LCD_TIMER	
B2h	Clock Source Register 2	Clock source for HDU_TIMER		Clock source for COM2_TIMER		Clock source for COM1_TIMER		Clock source for GNR2_TIMER	
68h	Clock Source Register 3	Clock source for R_TIMER		Clock source for IDLE_TIMER					

4.7.2.1 Timeout Count and Timeout SMI

The Timer Source Registers listed below are used to load the initial timeout count. The following rules apply.

- A timeout count 5 or greater indicates the countdown value. Timeout count values 1-4 should not be used: since the logic can take up to four clocks to reload a timeout count value, an invalid timeout could occur in the meantime.
- Writing a timeout count of 0 disables the timer.
- A dummy access in the appropriate address range for that timer triggers counting. From then on, additional accesses will reload the timer with its initial value and delay a timeout.
- Reading the timer value will return only the value initially written, not the current count. This rule holds true for all but the R_TIMER, which does return the current count. Register 60h returns the initial value of R_TIMER.

When a timeout occurs, it can only trigger an SMI. Registers listed under the "Enabling of Events to Generate SMI" heading of the "System Management Interrupt" section enable each timeout event individually to cause an SMI.

Timer Source Registers

Index	Name	7	6	5	4	3	2	1	0
44h	LCD_TIMER	Time count byte for LCD_TIMER - monitors LCD_ACCESS Timeout generates PMI#8.							
45h	DSK_TIMER	Time count byte for DSK_TIMER - monitors DSK_ACCESS Timeout generates PMI#9.							
46h	KBD_TIMER	Time count byte for KBD_TIMER - monitors KBD_ACCESS. Timeout generates PMI#10.							
B4h	HDU_TIMER	Time count byte for HDU_TIMER - monitors HDU_ACCESS. Timeout generates PMI#19							
B5h	COM1_TIMER	Time count byte for COM1_TIMER - monitors COM1_ACCESS. Timeout generates PMI#17							
B6h	COM2_TIMER	Time count byte for COM2_TIMER - monitors COM2_ACCESS. Timeout generates PMI#18.							
47h	GNR1_TIMER	Time count byte for GNR1_TIMER - monitors GNR1_ACCESS. Timeout generates PMI#11							
B7h	GNR2_TIMER	Time count byte for GNR2_TIMER - monitors GNR2_ACCESS. Timeout generates PMI#16.							
4Fh	IDLE_TIMER	Time count byte for IDLE_TIMER - monitors selected IRQs and EPMIs. Timeout generates PMI #4.							
69h	R_TIMER	Time count byte for R_TIMER - starts to count after a non zero write to this register Unlike the other timer registers, a read from this register returns the current count Timeout generates PMI#5.							
41h [7 5]	DOZE_TIMER	Time count bits for DOZE_TIMER - monitors selected IRQs and EPMIs. Unlike the other timer registers, DOZE_TIMER uses its own time base selected through bits 41h[7:5] Timeout generates PMI#27							

4.7.3 ACCESS Events

CPU memory and I/O instructions to peripheral devices cause power management events known as ACCESS events.

- Most ACCESS events generate an access PMI directly, which in turn can be enabled to activate the SMI input to the CPU so that the event can be serviced.
- These same events can be programmed to reload an associated countdown timer, thus delaying a timeout PMI from occurring.
- Still other ACCESS events can only cause a timer reload, and cannot directly generate an SMI.

Table 4-22 lists all of the ACCESS events and how the access can reload its associated timer and reload the IDLE_TIMER.

Table 4-22 ACCESS Events and Their Enabling Bit Locations

ACCESS Mnemonic	Monitored Range	ACCESS PMI#	Enable SMI on Current Access	Enable SMI on Next Access	Enable Reload of IDLE_TIMER
LPT	Reads/writes in I/O address ranges 378-Fh, 278-Fh, and 3B8-Fh (LPT1, 2, and 3)	--	--	--	4Eh[5]
COM1	Reads/writes in I/O address range 3F8-Fh.	21	DEh[5]	DBh[1]	BEh[4]
COM2	Reads/writes in I/O address range 2F8-Fh.	22	DEh[6]	DBh[2]	BEh[5]
DSK	FDD accesses to I/O port 3F5h and/or HDD accesses to 1F0-1F7h+3F6h. Bits 57h[5:4] determine which ranges apply.	13	DEh[1]	5Bh[1]	4Eh[1]
KBD	Reads/writes to I/O ports 060h and 064h.	14	DEh[2]	5Bh[2]	4Eh[2]
LCD	Reads/writes in memory address range A0000-BFFFFh and/or I/O address range 3B0-3DFh. Bits 43h[7:6] and 5Fh[7:6]. determine which ranges apply.	12	DEh[0]	5Bh[0]	4Eh[0]
HDU	HDU accesses in the integrated IDE controller range: 1F0-7h + 3F6h (primary) or 170-7h + 376h (secondary). Bit ACh[2] determines which addresses apply.	23	DEh[7]	DBh[3]	BEh[2]
CSG0	Defined in 4Ah[7:0], 4Bh[7:0], BFh[4,0]	--	--	--	4Eh[6]
CSG1	Defined in 4Ch[7:0], 4Dh[7:0], BFh[5,1]	--	--	--	4Eh[7]
CSG2	Defined in BCh[7:0], BDh[7:0], BFh[6,2]	--	--	--	BEh[6]
CSG3	Defined in BAh[7:0], BBh[7:0], BFh[7,3]	--	--	--	BEh[7]
GNR1	Defined in bits 48h[7:0], 49h[7:0], and AEh[4,2,0]	15	DEh[3]	5Bh[3]	4Eh[3]
GNR2	Defined in bits B8h[7:0], B9h[7:0], and AEh[5,3,1]	20	DEh[4]	DBh[0]	BEh[3]

Note that enabled access events, except for the GNRx, COMx, and CSG events, can be globally enabled to reload the DOZE_TIMER by setting bit 41h[1] = 1. Refer to the "Doze Mode" section for details.

4.7.3.1 Serial (COMx) and Parallel Port (LPT) Access

Accesses to the LPT1, LPT2, and LPT3 I/O range group can be programmed to reload the IDLE_TIMER. For a greater degree of control, COM1 and COM2 can individually be enabled to cause COM1_ACCESS or COM2_ACCESS, reload the COM1_TIMER or COM2_TIMER, and reload the IDLE_TIMER.

4.7.3.2 AT-Bus Floppy and Hard Drive Access

DSK_ACCESS can come from either or both of two separate access types. If enabled, the DSK_ACCESS reload the DSK_TIMER, and the IDLE_TIMER as well if desired.

- Floppy accesses to I/O port 3F5h generate DSK_ACCESS if bit 57h[5] = 0.
- Hard disk accesses to 1F0-1F7h & 3F6h generate DSK_ACCESS if bit 57h[4] = 0. Both AT-bus IDE accesses and VL-bus IDE accesses will generate the access event.

Two separate and independent hard disk drives can be managed if the primary drive is on the AT bus or VL bus and the secondary drive is managed by the integrated IDE controller. Refer to the HDU_ACCESS event regarding access events from the integrated local-bus IDE controller.

Index	Name	7	6	5	4	3	2	1	0
57h	PMU Control Register 6			DSK_ ACCESS includes FDD 0 = Yes 1 = No	DSK_ ACCESS includes HDD 0 = Yes 1 = No				

4.7.3.3 Integrated Controller Hard Drive Access

Accesses to the integrated hard disk controller, in the primary range 1F0-1F7h & 3F6h or the secondary range 170-177h & 377h can cause HDU_ACCESS, reload the HDU_TIMER, and reload the IDLE_TIMER. Bit ACh[2] determines which addresses apply.

HDU_ACCESS is based solely on the decoding for the internal IDE controller. It is independent of the DSK_ACCESS decoding. Therefore, bit 57h[4] does not affect HDU_ACCESS. DSK_ACCESS can continue to monitor both floppy disk and primary external hard disk accesses if desired.

4.7.3.4 Keyboard Access

Keyboard accesses to I/O ports 060h and 064h can cause KBD_ACCESS, reload the KBD_TIMER, and reload the IDLE_TIMER.

4.7.3.5 LCD Controller Access

Video controller accesses are defined as accesses to I/O ports 3B0-3DFh and to memory locations A0000-BFFFFh if not masked by bits 43h[7:6].

The enabled accesses cause LCD_ACCESS, reload the LCD_TIMER, and reload the IDLE_TIMER if not masked in bits 5Fh[7:6].

LCD Controller Access Control

Index	Name	7	6	5	4	3	2	1	0
43h	PMU Control Register 4	LCD_ ACCESS includes I/O range 3B0h-3DFh 0 = Yes 1 = No	LCD_ ACCESS includes memory range A0000-BFFFFh 0 = Yes 1 = No						
5Fh	PMU Control Register 7	LCD_ ACCESS includes AT-bus video access 0 = Yes 1 = No	LCD_ ACCESS includes VL-bus video access 0 = No 1 = Yes						
D5h	Resistor Control Register 2								Generate BOFF# (AHOLD) on Next Access Trap 0 = Disable 1 = Enable

Generate BOFF# (AHOLD) on Next Access Trap. When the LCD_TIMER has timed out and the Next Access feature has been enabled on memory access, the memory access cannot easily be trapped. Usually, the access takes place to a dead LCD subsystem. SMI occurs but does not get serviced until several instructions later.

When bit D5h[0] = 1, a Next Access to the LCD causes the 82C465MV logic to generate AHOLD before generating SMI. If the AHOLD signal is externally gated to generate a BOFF# signal (as for L1 write-back cache control), the BOFF# signal should force the CPU to move back to the start of the current instruction. After SMI is asserted, BOFF# (AHOLD) is de-asserted. Depending on the CPU logic, this procedure might convince the CPU to perform the SMI service first. If so, the video subsystem can be completely powered down on a backlight timeout and restored without losing characters on the next video access.

4.7.3.6 Chip Select Generation (CSG) Access

The CSG#0-3 lines can be programmed to generate a chip select based on either memory or I/O decoding of reads and/or writes. Even if the external logic necessary to implement the chip select lines is not in place, the chip select events themselves can be individually enabled to reload the IDLE_TIMER through bits 4Eh[7:6] and BEh[7:6].

4.7.3.7 General Purpose (GNR) Access

Two programmable ranges, GNR1 and GNR2, are provided, each with its own separate timer, to allow any two I/O or memory ranges to be monitored. As an example, the COM3 I/O range 3E8-3EFh could be monitored for reads and writes in order to determine whether the connected UART was in active use. As another example, a network card that uses memory in the D800-DFFFh half-segment could be monitored to determine whether the memory is being accessed regularly and, if not, a query could be sent through the network to ensure that the connection was still valid.

4.7.3.7.1 Memory Watchdog Feature

The 82C465MV general-purpose access register sets, GNR1 and GNR2, can be monitored for activity and can generate an SMI when no activity has occurred in a given amount of time. As an option, either or both of these register sets can be assigned to monitor memory space instead. In this case, instead of the bit values corresponding to I/O address bits A[9:0], the values correspond to memory address bits A[23:14]. The bits that select I/O read or I/O write cycles instead indicate memory read or memory write cycles.

Example

To monitor memory write activity in the 16KB block from CC00:0 to CC00:3FFF requires first viewing the CC00 segment value as

0000 1100 1100 0000 0000 0000

to determine the value of the upper 10-bits, CA[23:14], which is

0000110011

to write into the A[9:0] GNR address decode bits. The bits are set by writing:

- Register B8h (A[8:1]) = 00011001b, or 19h
- Register B9h (A9 + write decode + read decode + A[5:1] mask bits) = 01000000b, or 40h
- Register AEh (GNR2 cycle + GNR1 cycle + GNR2 A0 + GNR1 A0 + GNR2 A0 mask + GNR1 A0 mask) = 011000b, or 18h (GNR1 values must also be considered).

The timer values must then be entered, the PMI enabled, and then a dummy write access must be made to the CC000-CFFFFh range to start GNR2_TIMER. If no accesses are occurring, the timer will eventually expire and generate an SMI. If enabled, the next write access to this range will also cause an SMI and will reload the timer.

General Purpose Access 1 Registers

Index	Name	7	6	5	4	3	2	1	0
48h	GNR1 Base Address Register	GNR1_ACCESS base address A[8:1] (I/O) A[22:15] (memory)							
49h	GNR1 Control Register	GNR1 base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	GNR1 Mask bits for address A[5:1] (I/O) or A[19:15] memory - a '1' in a particular bit means that the corresponding bit 48h[4:0] is not compared. This is used to determine address block size.				

General Purpose Access 1 Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
AEh	GNR_ ACCESS Feature Register				GNR1 cycle decode type 0 = I/O 1 = Memory		GNR1 base address A0 (I/O) A14 (mem)		GNR1 mask bit A0 (I/O) A14 (mem)

General Purpose Access 2 Registers

Index	Name	7	6	5	4	3	2	1	0
B8h	GNR2 Base Address Register	GNR2_ACCESS base address A[8:1] (I/O) A[22:15] (memory)							
B9h	GNR2 Control Register	GNR2 base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	GNR2 Mask bits for address A[5:1] (I/O) or A[19:15] memory - a '1' in a particular bit means that the corresponding bit B8h[4:0] is not compared. This is used to determine address block size.				
AEh	GNR_ ACCESS Feature Register			GNR2 cycle decode type 0 = I/O 1 = Memory		GNR2 base address A0 (I/O) A14 (mem)		GNR2 mask bit A0 (I/O) A14 (mem)	

4.7.3.7.2 Memory Watchdog Feature Extension

The 82C465MVA part provides extended granularity for the memory watchdog function of GNR1 and GNR2. In the 82C465MV part, the GNR registers provide bits A[23:14] for a minimum monitoring range of 16KB. The 82C465MV logic extends these registers by providing address and mask bits for A[13:2] for a minimum monitoring range of 4 bytes.

When these registers are used to extend the existing I/O decoding registers, they override the setting of the 10/16-bit I/O decoding bit A0h[7]. In other words, even if bit A0h[7]=1 such that upper address bits must be zero, the GNR1 and GNR2 registers still decode the full 16 bits of the address as long as those upper bits are not masked off (default).

These new registers are completely ignored until at least one of them is written in order to maintain compatibility with the 82C465MV part. Once any of registers 70-75h is written, they must all be written. This rule is especially important for the mask bits: address comparison would normally be masked for memory cycles, but unmasked for I/O cycles, to maintain backward compatibility. Therefore, there is no clear default available and the register values must always be written explicitly.

Note that on both the 82C465MV and 82C465MVA parts, the memory watchdog feature cannot detect access by a local bus master other than the CPU, nor by an ISA bus master.

Memory Watchdog Feature Extension Registers

Index	Name	7	6	5	4	3	2	1	0
70h	GNR1 Control Register 2	GNR1_ACCESS base address (MVA) A[13:6] for memory watchdog A[15:10] for I/O (right-aligned)							
71h	GNR1 Control Register 3	GNR1_ACCESS mask bits (MVA) Mask for A[13:6] for memory watchdog Mask for A[15:10] for I/O (right-aligned)							
72h	GNR1 Base Addr Register 4	GNR1_ACCESS base address (MVA) A[5:2] for memory watchdog Ignored for I/O				GNR1_ACCESS mask bits (MVA) Mask for A[5:2] for memory watchdog Mask for A[9:6] for I/O			
73h	GNR1 Control Register 2	GNR2_ACCESS base address (MVA) A[13:6] for memory watchdog A[15:10] for I/O (right-aligned)							
74h	GNR1 Control Register 3	GNR2_ACCESS mask bits (MVA) Mask for A[13:6] for memory watchdog Mask for A[15:10] for I/O (right-aligned)							

Memory Watchdog Feature Extension Registers

Index	Name	7	6	5	4	3	2	1	0
75h	GNR1 Base Addr. Register 4	GNR2_ACCESS base address (MVA) A[5:2] for memory watchdog Ignored for I/O				GNR2_ACCESS mask bits (MVA) Mask for A[5:2] for memory watchdog Mask for A[9:6] for I/O			

4.7.3.7.3 Improved Memory Watchdog Range

The memory watchdog feature of the 82C465MVA can decode with four-byte granularity. The 82C465MVB part allows reassignment of this granularity so that the decode bits can extend to the full addressing range of the chip. On the 82C465MVA part, only addresses up to 16MB can be decoded and then the decode repeats throughout the system address space.

Bits AEh[7:6] control the new feature.

Memory Decoding Control Bits

Index	Name	7	6	5	4	3	2	1	0
AEh	GNR_ ACCESS Feature Register	GNR2 memory decoding 0=A[5:2] 1=A[31:24] (MVB)	GNR1 memory decoding 0=A[5:2] 1=A[31:24] (MVB)						
72h	GNR1 Base Addr. Register 4	GNR1_ACCESS base address (MVB) A[5:2] for memory watchdog if AEh[6]=0, for A[31]+x+A[25]+[A24] if AEh[6]=1 Ignored for I/O				GNR1_ACCESS mask bits (MVB) Mask for A[5:2] for memory watchdog if AEh[6]=0 for A[31]+x+A[25]+[A24] if AEh[6]=1 Mask for A[9:6] for I/O			
75h	GNR2 Base Addr. Register 4	GNR2_ACCESS base address (MVB) A[5:2] for memory watchdog if AEh[7]=0, for A[31]+x+A[25]+[A24] if AEh[7]=1 Ignored for I/O				GNR2_ACCESS mask bits (MVB) Mask for A[5:2] for memory watchdog if AEh[7]=0, for A[31]+x+A[25]+[A24] if AEh[7]=1 Mask for A[9:6] for I/O			

4.7.4 Activity Tracking

The Activity Tracking Register at Index DFh allows events to be flagged even if they are not programmed to generate an SMI. In this way, code can check whether a keystroke occurred since the last time the register was checked, for example, without actually generating an SMI for every keystroke.

The Activity Tracking Register records activity on all eight ACCESS events. No type of enabling is needed for any of these events to be registered. Reading this register returns flags indicating whether any of the events has taken place and automatically resets the entire register. The register can be written if desired to set the selected bits. In this way, a read-modify-write code sequence can be used to clear selected bits only.

Activity Tracking Registers

Index	Name	7	6	5	4	3	2	1	0
DFh	Activity Tracking Register	HDU_ ACCESS activity 0 = No 1 = Yes	COM2_ ACCESS activity 0 = No 1 = Yes	COM1_ ACCESS activity 0 = No 1 = Yes	GNR2_ ACCESS activity 0 = No 1 = Yes	GNR1_ ACCESS activity 0 = No 1 = Yes	KBD_ ACCESS activity 0 = No 1 = Yes	DSK_ ACCESS activity 0 = No 1 = Yes	LCD_ ACCESS activity 0 = No 1 = Yes

4.7.5 Reloading IDLE_TIMER

The 82C465MV provides the IDLE_TIMER to monitor system-wide activity: I/O and memory accesses by the CPU, IRQs from AT peripherals, and EPMIs from power control and management subsystems. The occurrence of an enabled event in any one of these areas will reload the IDLE_TIMER. Once there is inactivity for a sufficiently long time, the IDLE_TIMER will expire.

Expiration of the IDLE_TIMER generates PMI#4, which can be enabled to generate an SMI to inform system management code that the system is idle and that entry into suspend mode is appropriate. Refer to the "Suspend Mode" section in this document for complete information. Expiration of the IDLE_TIMER cannot cause automatic (hardware-controlled) entry into suspend mode, since important CPU processing could be interrupted.

The register bits that enable each event individually to reload the IDLE_TIMER and delay entry into suspend mode are shown below. Registers 63h and A3h are write-only; reads return no useful information.

Idle Reload Source Registers

Index	Name	7	6	5	4	3	2	1	0
4Eh	Idle Reload Event Enable Register 1	CSG1_ ACCESS 0 = Disable 1 = Enable	CSG0_ ACCESS 0 = Disable 1 = Enable	LPT_ ACCESS 0 = Disable 1 = Enable		GNR1_ ACCESS 0 = Disable 1 = Enable	KBD_ ACCESS 0 = Disable 1 = Enable	DSK_ ACCESS 0 = Disable 1 = Enable	LCD_ ACCESS 0 = Disable 1 = Enable
BEh	Idle Reload Event Enable Register 2	CSG3_ ACCESS 0 = Disable 1 = Enable	CSG2_ ACCESS 0 = Disable 1 = Enable	COM2_ ACCESS 0 = Disable 1 = Enable	COM1_ ACCESS 0 = Disable 1 = Enable	GNR2_ ACCESS 0 = Disable 1 = Enable	HDU_ ACCESS 0 = Disable 1 = Enable		
63h	Idle Timeout Select Register 1	EPMI1 Level-triggered 0 = Disable 1 = Enable	IRQ13 0 = Disable 1 = Enable	IRQ8 0 = Disable 1 = Enable	IRQ7 0 = Disable 1 = Enable	IRQ5 0 = Disable 1 = Enable	IRQ4 0 = Disable 1 = Enable	IRQ3 0 = Disable 1 = Enable	IRQ0 0 = Disable 1 = Enable
A3h	Idle Timeout Select Register 2	IRQ15 0 = Disable 1 = Enable	IRQ14 0 = Disable 1 = Enable	IRQ12 0 = Disable 1 = Enable	IRQ11 0 = Disable 1 = Enable	IRQ10 0 = Disable 1 = Enable	IRQ9 0 = Disable 1 = Enable	IRQ6 0 = Disable 1 = Enable	IRQ1 0 = Disable 1 = Enable

4.7.6 External PMI Events

The 82C465MV logic can monitor a variety of inputs that are directly related to low-power, battery-operated system designs. Table 4-23 lists the external power management input (EPMI) pins provided. Note that all pins included here are considered external PMI pins, not just pins EPMI1-4.

Table 4-23 External PMI Source Summary

Name	Description
LOWBAT	Activity on low battery pin
LLOWBAT	Activity on very low battery pin
EPMI1	Activity on external power management input 1
EPMI2	Activity on external power management input 2
EPMI3	Activity on external power management input 3
EPMI4	Activity on external power management input 4
RESUME	SUSP/RSM input has been toggled while in suspend
SUSPEND	SUSP/RSM input has been toggled while system is active
RI	Activity detected on RI

4.7.6.1 Suspend/Resume Pin

The SUSP/RSM pin can be programmed to generate a PMI when changed from high to low. Bits 61h[5:4] select the debounce logic applied to the pin. Once the system is in suspend mode, the SUSP/RSM pin is **always** enabled to resume operation when brought from high to low.

4.7.6.2 EPMI Signal Relocation

The 82C465MV offers several pin configuration options. In general, the EPMI-type pins are the first to be relocated because they need not be monitored constantly; they can all be grouped together and monitored one at a time on a periodic basis. The possibilities are described in the "Program-Selected Interface Options" section of this document.

4.7.6.3 Programming

The chipset registers listed below are used to initialize the EPMI pins and enable them to cause PMI events.

EPMI Programming Registers

Index	Name	7	6	5	4	3	2	1	0
40h	PMU Control Register 1		Global timer divide 0 = div. by 1 1 = div. by 4	LLOWBAT polarity 0 = Active hi 1 = Active low	LOWBAT polarity 0 = Active hi 1 = Active low		EPMI2 pin polarity 0 = Active hi 1 = Active low	EPMI1 pin polarity 0 = Active hi 1 = Active low	
DBh	Next Access Event Gen. Register 2			EPMI4 pin polarity 0 = Active hi 1 = Active low	EPMI3 pin polarity 0 = Active hi 1 = Active low				
43h	PMU Control Register 4			LOWBAT pin sample rate - generates PMI each time sampled active 00 = 32s 01 = 64s 10 = 128s 11 = Reserved					
61h	Debounce Register	LOWBAT, LLOWBAT debounce rate select 00 = No debounce 01 = 250us 10 = 8ms 11 = 500ms		SUSP/RSM debounce rate select 00 = Reserved 01 = Latch high-to-low edge 10 = 4ms(low-to-high) 11 = 8ms(low-to-high)					
A1h	Feature Control Register 2			Pin 88 - for EPMI3-4 0 = DRQ2 1 = EPMMUX			Emergency Overtemp Sense 0 = Disable 1 = Enable		EPMI1-2 Status Latch 0 = Dynamic 1 = Latched

* 1) EPMI1 and EPMI2 need to be asserted until recognized by its SMI service routine, since these PMIs are not latched unless A1h[0] = 1.
2) If EPMI1 and EPMI2 are used to place the system into suspend, the EPMIx signal must be de-asserted before the suspend command (setting bit 50h[0] = 1) is written unless bit A1h[0] = 1.

Emergency Overtemp Sense Enable - Setting bit A1h[2] = 1 allows a level on the EPMI2 pin to force the chip into Cool-down Clocking mode as set by the Thermal Management registers. The Thermal Management feature itself does not need to be enabled to use this sense function. The polarity of the input is determined by bit 40h[2]. Once written to 1, this bit cannot be cleared without a hardware reset.

EPMI1-2 Status Latch - Setting bit A1h[0] = 1 allows the EPMI1-2 PMI events to be latched. The status returned by bits 5Ch[2:1] is also latched. Writing these same bits to 1 clears the status bits. Note that setting A1h[0] = 0 retains 82C463MV-compatible operation.

4.7.6.4 Power Management Event Status

The power management input pins can be monitored for their instantaneous state in the 82C465MV. This feature can be used to poll for power management status without generating an SMI. The bits of the Power Management Event Status Register return instantaneous pin status; the state is not latched. However, since the state of some of these pins is read through a multiplexer, the value from the multiplexer is held for the duration of the multiplexer cycle. Inputs that change state in less than the

280ns cycle time of the multiplexer may not be indicated accurately through this register. The bits return 0 if the input is inactive, 1 if the input is active.

Power Management Event Status

Index	Name	7	6	5	4	3	2	1	0
DAh	Power Mgt. Event Status Register (read-only)			LOWBAT state 0 = inactive 1 = active	LLOWBAT state 0 = inactive 1 = active	EPMI4 state 0 = inactive 1 = active	EPMI3 state 0 = inactive 1 = active	EPMI2 state 0 = inactive 1 = active	EPMI1 state 0 = inactive 1 = active

4.8 System Management Interrupt (SMI)

Most modern 80x86 processors offer a System Management Interrupt (SMI) that allows external logic to signal to the CPU that a high-priority event has occurred and must be serviced but should not in any way interfere with the application currently being processed. When the CPU senses its SMI input active, it saves the context of its current application and loads the context of its System Management Mode (SMM) handler routine from a protected part of RAM. SMM code can then proceed to determine the reason for the interrupt, service it appropriately, and return to application processing through a special RESUME instruction that restores the context as it originally was before the SMI. Entry to and exit from SMM is completely hardware-controlled.

The 82C465MV handles up to 28 Power Management Interrupt (PMI) events that can be selectively enabled to cause an SMI to the CPU. Since some of these PMI events are actually a single indication from a group of events (such as a single PMI#6 that indicates whether any of the selected IRQ lines has gone active), the effective number of events that can be indicated is actually much greater than 28.

The PMI events that can be programmed to generate an SMI are listed in Table 4-24, Table 4-25, and Table 4-26.

Table 4-24 IRQ and EPMI SMI Sources

Source	PMI Name	Description
#3	LOWBAT	Activity on low battery pin
#0	LLOWBAT	Activity on very low battery pin
#1	EPMI1	Activity on external power management input 1
#2	EPMI2	Activity on external power management input 2
#24	EPMI3	Activity on external power management input 3
#25	EPMI4	Activity on external power management input 4
#26	RI	Selected count activity detected on RI
#7	SUSP/RSM	SUSP/RSM input has been toggled
#6	INTRGRP - OR - RSMGRP	An interrupt from the INTRGRP set has occurred while the system was running, - OR - An interrupt from the RSMGRP has occurred and resumed the system from suspend mode

Table 4-25 Timeout Event SMI Sources

Source	PMI Name	Description
#4	IDLE_TIMER	IDLE_TIMER has timed out due to no I/O activity
#27	DOZE_TIMER	DOZE_TIMER has timed out due to inactivity of selected devices
#5	R_TIMER	R_TIMER has timed out on its normal periodic basis

Source	PMI Name	Description
#8	LCD_TIMER	LCD timer has timed out because of no LCD activity
#9	DSK_TIMER	Floppy (and/or external hard) disk timer has timed out because of no activity
#19	HDU_TIMER	Timeout has occurred because no access has occurred in the internal IDE range
#10	KBD_TIMER	Keyboard timer has timed out because of no controller accesses
#11	GNR1_TIMER	Timeout has occurred because the memory or I/O range selected by GNR1 has had no activity
#16	GNR2_TIMER	Timeout has occurred because the memory or I/O range selected by GNR2 has had no activity
#17	COM1_TIMER	Timeout has occurred because no access has occurred in the COM1 range
#18	COM2_TIMER	Timeout has occurred because no access has occurred in the COM2 range

Table 4-26 Access Event SMI Sources

Source	PMI Name	Description
#14	KBD_ACCESS	Keyboard controller has been accessed, either before or after timer timeout depending on current/next access setting
#12	LCD_ACCESS	LCD controller has been accessed, either before or after timer timeout depending on current/next access setting
#13	DSK_ACCESS	Floppy (or external hard) disk controller has been accessed, either before or after timer timeout depending on current/next access setting
#23	HDU_ACCESS	Internal IDE has been accessed, either before or after timer timeout depending on current/next access setting
#15	GNR1_ACCESS	GNR1 range has been accessed, either before or after timer timeout depending on current/next access setting
#20	GNR2_ACCESS	GNR2 range has been accessed, either before or after timer timeout depending on current/next access setting
#21	COM1_ACCESS	COM1 has been accessed, either before or after timer timeout depending on current/next access setting
#22	COM2_ACCESS	COM2 has been accessed, either before or after timer timeout depending on current/next access setting

4.8.1 SMI Presetting for Various CPU Type

The 82C465MV logic provides features to handle most types of SMM handling commonly in use. Therefore, presetting SMI operation for various processors involves several different concepts of system management mode entry, exit, and activity.

For all CPUs with SMI support, setting bit 30h[3] = 1 allows relocation of CPU-generated addresses during system management mode (SMM). The two 64KB DRAM segments at A000h and B000h are used to provide this relocated space, because these address ranges correspond to shadow memory of video access ranges which are always redirected to the VL bus or AT bus anyway. The DRAM at these locations would otherwise go unused.

Settings of the other SMI Initialization Registers shown below vary according to CPU type. Suggested settings for the most popular CPUs are described in the sections that follow.

SMI Initialization Registers

Index	Name	7	6	5	4	3	2	1	0
30h	Control Register 1					SMI address relocation 0 = Disable 1 = Enable			
5Bh	PMU Event Register 4	SMI to IRQ15 0 = Disable 1 = Enable			SMI type 0 = Intel 1 = Other				
6Bh	Resume Source Register		Pin 135 0 = FLUSH# 1 = SMIRDY#						
A0h	Feature Control Register 1						SRESET Enable in SMM 0 = Enable 1 = Disable		

4.8.1.1 Intel SL-Enhanced and AMD 486Plus CPU Settings

The Intel SL-Enhanced series CPUs and AMD 486Plus series CPUs use the SMIACT# pin to indicate operation in SMM. These CPUs generate addresses in the 3000h and 4000h segments to execute SMM code and access SMM data. To preset operation for these CPUs, pins 13 and 23 must be sensed high at hardware reset time as described in the "Strap-Selected Interface Options" section of this document. Then, the register bits must be set as follows.

1. Set bit 5Bh[7] = 0 to disable rerouting of SMI to IRQ15.
2. Set bit 5Bh[4] = 0 to enable Intel-type handling of SMM. This setting selects the SMI# pin for output only and defines pin 136 as SMIACT#.
3. Set bit A0h[2] = 1 to prevent SRESET from occurring within SMM. Intel and AMD CPUs require SRESET to be blocked during SMM.
4. Set bits AFh[7:4] to the CPU code segment SMBASE; these bits default to 3h for the 3000h segment. When in SMM, CPU accesses to the selected code segment will map to the DRAM at B000h.
5. Set bits AFh[3:0] to the CPU data segment SMBASE; these bits default to 4h for the 4000h segment. When in SMM, CPU accesses to the selected data segment will map to the DRAM at A000h.
6. Finally, set bit 30h[4] = 1 to enable the SMM remapping scheme.

Proceed to Section 4.8.2, *Loading Initial SMM Code and Data*, on page 102.

4.8.1.2 Cyrix and TI CPU Settings

Current Cyrix and Texas Instruments CPUs use the SMIADS# pin to indicate whether they are currently executing SMM code. These CPUs generate addresses in the 6000h and 7000h segments to execute SMM code and access SMM data. To preset operation for these CPUs, pins 13 and 23 must be sensed high at hardware reset time as described in the "Strap-Selected Interface Options" section of this document. Then, the register bits must be set as follows.

1. Set bit 5Bh[7] = 0 to disable rerouting of SMI to IRQ15.
2. Set bit 5Bh[4] = 1 to configure the SMI# pin as bidirectional and define pin 136 as SMIADS#.
3. Set bit 6Bh[6] = 1 to enable pin 135 as SMIRDY# for all Cyrix and TI CPUs except for the Cyrix Cx486DX, which does not require SMIRDY# but does require FLUSH#.
4. Set bit A0h[2] = 1 to prevent SRESET from occurring within SMM if desired.
5. Set bits AFh[7:4] to 7h for the CPU code segment SMBASE, which is 7000h on the Cyrix 486S/S2 and TI CPUs. When in SMM, CPU accesses to the selected code segment will map to the DRAM at B000h.

6. Set bits AFh[3:0] to 6h for the CPU data segment SMBASE, which is 6000h on the 486S/S2 CPU. When in SMM, CPU accesses to the selected data segment will map to the DRAM at A000h.
7. Finally, set bit 30h[4] = 1 to enable the SMM remapping scheme.

Proceed to Section 4.8.2, *Loading Initial SMM Code and Data*, on page 102.

4.8.1.3 AMD 486DXLV / IBM "Blue Lightning" CPU Settings

The AMD 486DXLV and IBM "Blue Lightning" CPUs use the SMIADS# pin to indicate whether they are currently executing SMM cycles. These CPUs generate addresses in the 6000h segment to access SMM data. To preset operation for these CPUs, pin 13 must be sensed low and pin 23 sensed high at hardware reset time as described in the "Strap-Selected Interface Options" section of this document. Then, the register bits must be set as follows.

1. Set bit 5Bh[7] = 0 to disable rerouting of SMI to IRQ15.
2. Set bit 5Bh[4] = 1 to configure the SMI# pin as bidirectional and define pin 136 as SMIADS#.
3. Set bit 6Bh[6] = 1 to enable pin 135 as SMIRDY#.
4. Set bit A0h[2] = 1 to prevent SRESET from occurring within SMM if desired.
5. Set bits AFh[7:4] to 7h for the CPU code segment SMBASE, which is 7000h on the 486DXLV and IBM CPUs. When in SMM, CPU accesses to the selected code segment will map to the DRAM at B000h.
6. Set bits AFh[3:0] to 6h for the CPU data segment SMBASE, which is 6000h on the 486DXLV and IBM CPUs. When in SMM, CPU accesses to the selected data segment will map to the DRAM at A000h.
7. Finally, set bit 30h[4] = 1 to enable the SMM remapping scheme.

Note that the AMD 486DXLV and IBM "Blue Lightning" processors jump to the reset vector, FFFFFFF0h, upon entering SMM. Bit 40h[7] is provided for BIOS code to determine whether the jump to this entry point is due to an SMI.

Index	Name	7	6	5	4	3	2	1	0
40h	PMU Control Register 1	Last jump to reset vector 0 = ADS# 1 = SMIADS#							

Proceed to the Section 4.8.2, *Loading Initial SMM Code and Data*, on page 102.

4.8.1.4 Non SMI CPU Settings

The IRQ15 SMI Select feature is provided for CPUs without an SMI pin to emulate SMM through IRQ15. For CPUs with an SMI, this feature must always be disabled.

Bit 5Bh[7] = 0 does not reroute SMIs and allows the IRQ15 pin to function as normal. Bit 5Bh[7] = 1 enables SMI to be internally connected to IRQ15 and disables the IRQ15 hardware pin function. Therefore, any enabled PMI events will trigger the internal IRQ15 signal.

4.8.2 Loading Initial SMM Code and Data

On system initialization, the system management code and data segments must be loaded from ROM with the appropriate information. This information will reside in the DRAM segments at physical starting addresses A0000h and B0000h and, once loaded, will be write-protected except when the system is operating in SMM.

Step 1: System Initialization (not in SMM)

On system initialization, the BIOS must load initial code and data into the protected SMM memory space. Normally the system will still be executing out of ROM at this point, but the memory subsystem is configured and enabled. The Dynamic SMI Relocation bit is used for this purpose (and for other reasons described in the "Run-Time SMI Address Relocation" section below).

Dynamic SMI Relocation Bit

Index	Name	7	6	5	4	3	2	1	0
31h	Control Register 2				Dynamic SMI relocation 0 = Normal 1 = Remap				

Bit 31h[4] is used as follows outside of SMM.

- Bit 31h[4] = 0: No Relocation. This setting prevents application software from accessing SMI memory space.
- Bit 31h[4] = 1: Remap CPU addresses in the 3000/4000h segments to SMI memory space, the DRAM segments at B000h/A000h. This setting provides the mechanism for initially loading SMI code to the B000h/A000h region.

The BIOS sets bit 31h[4] = 1. It can then load code and data into DRAM segments B000h and A000h by copying it to segments 3000h and 4000h, respectively. Even if the CPU in use defaults to different segments, such as CPUs that use 6000h and 7000h, this first load operation must be addressed to the 3000h and 4000h segments. Upon completing the loading of all initial SMM code and data, the BIOS clears bit 31h[4] to 0 to protect the SMM space.

Step 2: Software Generation of SMI

Having loaded the code and data, BIOS must now generate an SMI to enter SMM so that it can complete the SMM initialization process (by changing SMBASE if needed or performing system-specific tasks). To allow software SMI generation to take place, bit 59h[7] must be written to 1. Writing bit 50h[7] = 1 then asserts SMI to the CPU to start SMM operation. Writing bit 50h[7] = 0 clears the SMI. The SMI routine must clear this bit; otherwise, SMI requests will be generated continuously.

Software SMI Enable Registers

Index	Name	7	6	5	4	3	2	1	0
59h	PMU Event Register 2	Allow software SMI 0 = Disable 1 = Enable							
50h	PMU Control Register 5	Software start SMI 0 = Clear SMI 1 = Start SMI							

Step 3: Reprogramming SMBASE

Once the system has entered SMM for the first time, the CPU SMBASE value can be reprogrammed for future use. The 82C465MV SMBASE value must be subsequently updated to match the CPU SMBASE value. The operation should occur as follows.

1. SMM code updates the SMBASE value of the CPU register save area.
2. SMM code generates a RESUME instruction to return control to the BIOS initialization code. The new SMBASE value gets written to the CPU registers.
3. BIOS code rewrites the 82C465MV SMBASE register with the new value.

Using this procedure guarantees reliable results. If the 82C465MV SMBASE register is changed from within SMM, the change takes place on the next code fetch or data access and may not operate predictably.

4.8.2.1 SMBASE Register

The SMBASE Register at index AFh operates in conjunction with program option bit 5Bh[4]. Bit 5Bh[4] selects the function of the SMI# and SMIADS#/SMIACT# pins and also sets the base address of the SMI code and data segments in the system (SMBASE). Segments 3000h and 4000h are selected by default; these map to B000h and A000h, respectively, when bit 5Bh[4] = 0. Other processors use segments 6000h and 7000h, which map to A000h and B000h, respectively when bit 5Bh[4] = 1 (note that the order of A000h and B000h changes in this case).

The SMBASE Register provides independent relocation programming for the two segments associated with SMI. The mapping can be reprogrammed at any time. After reset, as long as the new SMBASE Register is not written, the mapping uses the value associated with the bit 5Bh[4] selection. Once the SMBASE Register at index AFh has been written with any value, all further mapping is selected by register AFh; changing bit 5Bh[4] could cause undesirable results.

SMBASE Register

Index	Name	7	6	5	4	3	2	1	0
AFh	SMBASE Register	SMM segment to be mapped to B000h 0 = 0:0 1 = 1000:0h ... 9 = 9000:0h (A-F illegal) defaults to 3h				SMM segment to be mapped to A000h 0 = 0:0 1 = 1000:0h ... 9 = 9000:0h (A-F illegal) defaults to 4h			

4.8.3 Run-Time SMI Address Relocation

The Dynamic SMI Relocation feature provides full memory access control while in SMM. SMI relocation at run time is controlled by bit 31h[4].

Index	Name	7	6	5	4	3	2	1	0
31h	Control Register 2				Dynamic SMI relocation 0 = Normal 1 = Remap				

4.8.3.1 Relocation with Standard Interface SMI

The standard interface SMI (bit 5Bh[4] = 0) uses the SMIACT# signal. Normally SMM code, the data at memory segments 3000h/4000h is not accessible by the CPU because these addresses are mapped to the SMI address space at B000h/A000h. However, bit 31h[4] can be used as follows to access this area during an SMI routine.

- Bit 31h[4] = 0: Relocate all accesses in the 3000h/4000h segment to the B000h/A000h SMI segment (normal operation).
- Bit 31h[4] = 1: Code fetches (CPU D/C# low) from the CPU in the 3000h/4000h segment will be translated from the SMI space at segment B000h/A000h. Memory data accesses (CPU D/C# high) in the 3000h/4000h space will not be translated to SMI space. This allows data in the 3000h/4000h memory space to be accessed and saved to disk.

4.8.3.2 Relocation with Alternative Interface SMI

The alternative mode SMI (bit 5Bh[4] = 1) uses the SMIADS# signal. In this mode, bit 31h[4] must be programmed to 0 while in SMM; setting it to 1 is illegal. For an SMIADS# cycle, accesses in the 6000h/7000h segment are relocated to the A000h/B000h SMI segment. For normal ADS# cycle, there is no relocation.

4.8.4 SMI Event Generation

The registers shown below control the events that are allowed to generate an SMI. The programming occurs as follows:

1. Timeout, access, and interrupt events must be programmed to generate a PMI.
2. The PMI event must be enabled to generate the SMI signal.
3. SMIs are globally unmasked to allow full operation.

This process is described in detail below.

4.8.4.1 Timeout Event Generation of SMI

For timeout events, simply loading a non zero timer value and generating a dummy access presets PMI generation on the next timeout. Refer to the "Timers" section for information on programming the timers.

4.8.4.2 Access Event Generation of SMI

Access events can be programmed to generate an SMI. The 82C465MV classifies accesses as Current Access or a Next Access depending on whether the timer associated with that access range is still running or has timed out, respectively. The available access event ranges are defined under the "ACCESS Event" heading of the "Power Management Unit" section.

- Next Access - occurs after a timeout, the first time software attempts to access the I/O and/or memory range that caused the timeout. The Next Access feature provides a way for I/O accesses, to a peripheral whose timer has timed out, to cause an SMI so that the peripheral can be powered up before the access takes place. Next access can also speed up system clocks if the system is in hardware (slow-clock) doze mode when the access occurs.
- Current Access - occurs any time this feature is enabled for an I/O and/or memory range, whether or not the device has timed out. The Current Access PMI can be programmed to cause an SMI, but cannot provide any automatic means of controlling system clocks.

If both the Current Access and Next Access features are enabled for an event, and the timer has timed out, an access will only cause a single SMI. Since both access types use the same PMI#, clearing either one clears both events.

Current and Next Access Registers

Index	Name	7	6	5	4	3	2	1	0
5Bh	PMU Event Register 4					GNR1 Next Access PMI #15 0 = Disable 1 = Enable	KBD Next Access PMI #14 0 = Disable 1 = Enable	DSK Next Access PMI #13 0 = Disable 1 = Enable	LCD Next Access PMI #12 0 = Disable 1 = Enable
DBh	Next Access Event Gen. Register 2	I/O Blocking Control 0 = Unblock 1 = Block I/O Next Access				HDU_ ACCESS PMI#23 on Next Access 0 = No 1 = Yes	COM2_ ACCESS PMI#22 on Next Access 0 = No 1 = Yes	COM1_ ACCESS PMI#21 on Next Access 0 = No 1 = Yes	GNR2_ ACCESS PMI#20 on Next Access 0 = No 1 = Yes
DEh	Current Access Event Gen Register	HDU_ ACCESS PMI #23 on Current Access 0 = No 1 = Yes	COM2_ ACCESS PMI#22 on Current Access 0 = No 1 = Yes	COM1_ ACCESS PMI#21 on Current Access 0 = No 1 = Yes	GNR2_ ACCESS PMI#20 on Current Access 0 = No 1 = Yes	GNR1_ ACCESS PMI#15 on Current Access 0 = No 1 = Yes	KBD_ ACCESS PMI #14 on Current Access 0 = No 1 = Yes	DSK_ ACCESS PMI #13 on Current Access 0 = No 1 = Yes	LCD_ ACCESS PMI #12 on Current Access 0 = No 1 = Yes

4.8.4.2.1 I/O Blocking

The I/O blocking bit DBh[7] operates as follows. This selection allows the I/O access that causes a Next Access PMI to be either blocked (if the peripheral is turned off, for example) or passed through. DBh[7] = 0 means the I/O will not be blocked; DBh[7] = 1 means the I/O on Next Access will be blocked and the CPU must be programmed to restart the I/O command if desired. The feature defaults to "blocked". Note that if an I/O read access is blocked on generation of SMI, a value of 0FFh is returned to the bus master.

4.8.4.3 No Flush Required on Entry to SMM

The 82C465MVA part provides a new feature that will allow the CPU SMBASE to be assigned to an address in the first 1MB of any boundary at or above 64MB (i.e. 64MB, 128MB, 192MB...). The CPU will recognize this region as being outside of cacheable memory space and will not attempt to access its on-board cache to retrieve SMM code and data. Therefore, the cache need not be flushed on generation of the SMIACT# signal as occurs in the current 82C465MV design. The feature works as in the following example.

Example

The new CPU SMBASE address will be selected as 64MB + segment 3000h for code, segment 4000h for data, in this example. The addresses will map to 4030000h and 4040000h respectively.

- During non SMI mode operations, the CPU will not generate the SMIACT# signal. Therefore, the 82C465MV will not remap 3000h/4000h accesses to B000h/A000h.
- The first SMI will take place to 30000h as usual and a cache flush will take place. During this first SMI the CPU SMBASE can be modified to 4030000h; this value will be loaded to the CPU on execution of the resume instruction and will take effect on the next SMI. Also during the initial SMI, SMM code sets bit D3h[5]=1 in the 82C465MVA registers to inhibit cache flush on entry to SMI.

NOTE The SMBASE register of the 82C465MV and 82C465MVA parts must be written *after* the resume instruction from the initial SMI, since it takes effect as soon as it is written.

- On subsequent SMIs, the CPU will generate the new address with bit A26 high to prevent it from attempting to access the code and data from its own internal cache. The 82C465MVA logic will not see bit A26; however, it will see SMIACT# active and will infer that the current cycle must come from protected SMM DRAM, not the normal DRAM segment indicated.

With the new feature enabled, SMM code has the additional choice of making DRAM data accesses either at the SMBASE DRAM segment or at the normal memory segment for that address. Bits D3h[7:6] are provided to select between the SMM data segment and the normal data segment for reads and writes, respectively. In this way, the MOVS instruction can be used by SMM code to copy data between segments with no intermediate storage of the data. This feature is especially important if the 82C465MVA SMBASE is reassigned to select A000h for the data segment. Bits D3h[7:6] could then be used to copy data between the SMM data segment and the video RAM at A000h.

SMM Flush Control Bits

Index	Name	7	6	5	4	3	2	1	0
D3h	Asym. DRAM Select Register	Segment for SMM data reads 0=A000h 1=SMBASE (MVA)	Segment for SMM data writes 0=A000h 1=SMBASE (MVA)	Cache Flush on SMI entry 0=Enable 1=Disable (MVA)					

4.8.4.4 Interrupt Event Generation of SMI

Asynchronous events from peripheral devices requesting service from the CPU are known as interrupt events. Interrupts in this context include both the traditional AT architecture IRQs and additional inputs known as external power management interrupts (EPMIs). For the 82C465MV logic, the desired interrupts are all grouped into a single event called INTRGRP. INTRGRP can then be enabled to cause an SMI.

If it is desired to generate an SMI from the INTRGRP event, setting bit 57h[6] = 1 will allow any of the selected interrupt events to generate PMI#6. Once in the SMI handler, the SMM code can read the registers at indexes 64h and A4h to determine which of the interrupt(s) caused the event. The IRQs will remain latched for reading in these registers until PMI#6 is cleared, at which time any latched sources are cleared. The INTRGRP IRQ Select Registers are shown below.

INTRGRP IRQ Select Registers

Index	Name	7	6	5	4	3	2	1	0
64h	INTRGRP IRQ Select Register 1	IRQ14 0 = Disable 1 = Enable	IRQ8 0 = Disable 1 = Enable	IRQ7 0 = Disable 1 = Enable	IRQ6 0 = Disable 1 = Enable	IRQ5 0 = Disable 1 = Enable	IRQ4 0 = Disable 1 = Enable	IRQ3 0 = Disable 1 = Enable	IRQ1 0 = Disable 1 = Enable
A4h	INTRGRP IRQ Select Register 2	Test bit Write as 0.	IRQ15 0 = Disable 1 = Enable	IRQ13 0 = Disable 1 = Enable	IRQ12 0 = Disable 1 = Enable	IRQ11 0 = Disable 1 = Enable	IRQ10 0 = Disable 1 = Enable	IRQ9 0 = Disable 1 = Enable	IRQ0 0 = Disable 1 = Enable
57h	PMU Control Register 6		INTRGRP generates PMI #6 0 = Disable 1 = Enable						

4.8.4.5 Enabling of Events to Generate SMI

The registers listed below allow PMI events that are enabled to generate timer timeouts, accesses, and interrupts to cause SMIs. Before setting the SMI Event Enable Registers shown below, timeouts, accesses, and interrupts must be individually enabled to generate PMI events as follows.

- For timeout events, loading a non zero timer value and generating a dummy access presets PMI generation on the next timeout.
- For current access events, the appropriate current access enable bit must be set to preset PMI generation on the following access.
- For next access events, the appropriate next access enable bit must be set. Then, a valid timeout must take place to preset PMI generation on the following access.
- For interrupt events, the corresponding INTRGRP bit must be set and INTRGRP must be enabled to generate PMI#6. Then, PMI#6 will occur on any enabled interrupt.

The PMIs should be enabled to generate SMIs through the register set below only after all desired PMI events have been enabled. Setting bit 5Bh[6] = 1 then unmasks all the SMIs previously enabled.

Note that a resume event can be enabled to generate PMI#6. Refer to the "Suspend and Resume" section for details on enabling resume events.

SMI Event Enable Registers

Index	Name	7	6	5	4	3	2	1	0
5Bh	PMU Event Register 4		Global SMI control 0 = Allow 1 = Mask						
58h	PMU Event Register 1	LOWBAT PMI #3 SMI 00 = Disable 11 = Enable		EPMI2 PMI #2 SMI 00 = Disable 11 = Enable		EPMI1 PMI #1 SMI 00 = Disable 11 = Enable		LLOWBAT PMI #0 SMI 00 = Disable 11 = Enable	
D9h	PMU Event Register 6			RI PMI #26 SMI 00 = Disable 11 = Enable		EPMI4 PMI #25 SMI 00 = Disable 11 = Enable		EPMI3 PMI #24 SMI 00 = Disable 11 = Enable	
5Ah	PMU Event Register 3	GNR1_TIMER PMI #11 SMI 00 = Disable 11 = Enable		KBD_TIMER PMI #10 SMI 00 = Disable 11 = Enable		DSK_TIMER PMI #9 SMI 00 = Disable 11 = Enable		LCD_TIMER PMI #8 SMI 00 = Disable 11 = Enable	
D8h	PMU Event Register 5	HDU_TIMER PMI #19 HDU_ACCESS PMI #23 SMI 00 = Disable 11 = Enable		COM2_TIMER PMI #18 COM2_ACCESS PMI #22 SMI 00 = Disable 11 = Enable		COM1_TIMER PMI #17 COM1_ACCESS PMI #21 SMI 00 = Disable 11 = Enable		GNR2_TIMER PMI# 16 GNR2_ACCESS PMI #20 SMI 00 = Disable 11 = Enable	
59h	PMU Event Register 2			INTRGRP/Resume PMI #6, Suspend PMI #7 SMI 00 = Disable 11 = Enable		R_TIMER PMI #5 SMI 00 = Disable 11 = Enable		IDLE_TIMER PMI #4 SMI 00 = Disable 11 = Enable	
DBh	Next Access Event Gen Register 2		Cool-down Clocking entry/exit PMI#25 SMI 0 = Disable 1 = Enable						

4.8.4.5.1 PMI#25 Triggers

The PMI#25 event is shared by both EPMI4 and the thermal management unit. Bits D9h[3:2] enable SMI for EPMI4 only. Bit DBh[6] enables SMI only for Cool-down Clocking entry and exit.

4.8.5 DRQ Generation of SMI

The 82C465MVA allows activity on the DRQ pins to generate an SMI. The SMI takes place *before* the DMA transfer occurs, allowing SMM code to emulate or modify the operation. Writing the bit to clear the PMI then allows any pending DMA operation to take place immediately.

There are certain latency limitations for DMA operations. For example, floppy disk DMA transfers generally must be serviced within 14us from receipt of DRQ2 in order to avoid an overrun condition. Entry into SMM requires a considerable amount of time in itself. Therefore, SMM routines that trap DMA accesses must be structured concisely so that the DMA cycle is allowed to occur before the latency limit is exceeded.

Index	Name	7	6	5	4	3	2	1	0
D6h	PMU Control Register 10	DSK_ ACCESS 0=3F5h only 1=All FDC ports (3F2,4,5,7h and 372,4,5,7h)	DMA Trap PMI#28 SMI 0=Disable 1=Enable	DMAC1 Byte Pointer Flip-Flop Read Only 0=Cleared 1=Set	HITM# Source 0=D/C# 1=Pin 135	Local-bus DMA LDEV# sampling 0=Normal 1=Sample one clock sooner	I/O Port Access Trapped - Read Only. 0=I/O Read 1=I/O Write	ACCESS Trap bit A9 - Read Only.	ACCESS Trap bit A8 - Read Only.
DDh	PMU SMI Source Register 4				PMI #28 - DMA 0=Clear 1=Active (MVA)	PMI #27 - DOZE_ TIMER 0=Clear 1=Active	PMI #26 - RI 0=Clear 1=Active	PMI #25 - EPMI4 pin/ Cool-down Clocking 0=Clear 1=Active	PMI #24 - EPMI3 pin 0=Clear 1=Active

4.8.6 Servicing an SMI

The register set shown below is used by SMM code to enable system events to cause SMIs, to determine the events that caused an active SMI, and to clear the events. Upon entry to SMM, the chip clears the SMI# signal to the CPU. Then, determining the source of the SMI is a simple procedure.

1. Read the registers at indexes 5Ch, 5Dh, DCh, and DDh. Any non zero bits indicate PMI sources. More than one can be active.
2. The PMI number will indicate the source of the service request. In case PMI#6 is indicated, also read registers 64h and A4h (described earlier in the "Interrupt Event Generation of SMI" section) to determine which IRQ line was responsible for the event.
3. Service the events in the order desired. Upon completion of each service, write a '1' back to the event source register bit to clear that event. Continue in this manner until all events are serviced and all the SMI service registers are clear.
4. Issue the proper CPU instruction to return from SMM operation.

If any events are still pending upon resume from SMM, the 82C465MV chip will issue a new SMI# immediately.

SMI Service Registers

Index	Name	7	6	5	4	3	2	1	0
5Ch	PMI Source Register 1 Write 1 to clear	PMI7 - Suspend 0 = Inactive 1 = Active	PMI6 - Resume or INTRGRP 0 = Inactive 1 = Active	PMI5 - R_TIMER timeout 0 = Inactive 1 = Active	PMI4 - IDLE_TMR timeout 0 = Inactive 1 = Active	PMI3 - LOWBAT 0 = Inactive 1 = Active	PMI2 - EPMI2 0 = Inactive 1 = Active	PMI1 - EPMI1 0 = Inactive 1 = Active	PMI0 - LLOWBAT 0 = Inactive 1 = Active
5Dh	PMI Source Register 2 Write 1 to clear	PMI15 - GNR1_ ACCESS 0 = Inactive 1 = Active	PMI14 - KBD_ ACCESS 0 = Inactive 1 = Active	PMI13 - DSK_ ACCESS 0 = Inactive 1 = Active	PMI12 - LCD_ ACCESS 0 = Inactive 1 = Active	PMI11 - GNR1_ TIMER 0 = Inactive 1 = Active	PMI10 - KBD_ TIMER 0 = Inactive 1 = Active	PMI9 - DSK_ TIMER 0 = Inactive 1 = Active	PMI8 - LCD_ TIMER 0 = Inactive 1 = Active
DCh	PMU SMI Source Register 3 Write 1 to clear	PMI #23 - HDU_ ACCESS 0 = Inactive 1 = Active	PMI #22 - COM2_ ACCESS 0 = Inactive 1 = Active	PMI #21 - COM1_ ACCESS 0 = Inactive 1 = Active	PMI #20 - GNR2_ ACCESS 0 = Inactive 1 = Active	PMI #19 - HDU_ TIMER 0 = Inactive 1 = Active	PMI #18 - COM2_ TIMER 0 = Inactive 1 = Active	PMI #17 - COM1_ TIMER 0 = Inactive 1 = Active	PMI #16 - GNR2_ TIMER 0 = Inactive 1 = Active

SMI Service Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
DDh	PMU SMI Source Register 4 Write 1 to clear				PMI #28 - DMA 0 = Clear 1 = Active (MVA)	PMI #27 - DOZE_ TIMER 0 = Inactive 1 = Active	PMI #26 - RI 0 = Inactive 1 = Active	PMI #25 - EPMI4 pin/ Cool-down Clocking 0 = Inactive 1 = Active	PMI #24 - EPMI3 pin 0 = Inactive 1 = Active

4.8.6.1 PMI Source Register Details

Registers 5Ch, 5Dh, DCh, and DDh indicate the SMI source. When a PMI event occurs, the corresponding bit will be set to '1' and the SMI# signal will then be generated. In the SMI service routine, SMM code must check these registers for the PMI source(s) and then clear them. Otherwise, for all but the EPMI pins the latched PMI source will generate SMI# continuously. SMI code normally clears only one event at a time to keep track of the events as they are serviced, but all events can be cleared at once if desired. Note that clearing bit 5Ch[6] will clear bit 5Ch[7] also.

Refer to the "Suspend and Resume" section of this document for information on PMI#6 when it is used to indicate a resume event.

4.8.6.2 EPMI Pin PMI Sources

The EPMI1-2 pin PMI source indicator bits behave a little differently than the rest of the PMI source indicator bits. For PMIs #1 & #2, the EPMI1-2 inputs are not latched by default, so bits 5Ch[2:1] are not latched. Therefore, an external device could trigger an SMI by toggling one of the EPMI1-2 lines, but if the device returns the EPMI line to its inactive state before SMM code reads bits 5Ch[2:1], the code would not be able to recognize the event that triggered the SMI. Likewise, an EPMI1-2 edge could initiate a resume from suspend mode, but then would not be recognized if the EPMI pin went to its inactive state.

Bit A1h[0] is provided to allow EPMI1-2 to be latched like other PMIs. If bit A1h[0] is written to 1, EPMI1-2 events will be latched at bits 5Ch[2:1]. Writing a '1' into the active bit(s) then clears the PMI.

For PMIs #24 & #25, the EPMI3-4 inputs are always latched, regardless of the A1h[0] setting.

4.8.6.3 I/O SMI Trap Indication

The 82C465MVA part provides a means for SMM code to determine the I/O port whose access caused the SMI, as well as a bit to indicate whether the access was a read or a write access.

I/O Access Trap Registers

Index	Name	7	6	5	4	3	2	1	0
D6h	PMU Control Register 10						I/O Port Access Trapped - Read Only. 0=I/O Read 1=I/O Write (MVA)	ACCESS Trap bit A9 - Read Only. (MVA)	ACCESS Trap bit A8 - Read Only. (MVA)
D7h	ACCESS Port Address Register	ACCESS Trap Address bits A[7:0] - These bits, along with A[9:8] in bits D6h[1:0], provide the 10-bit I/O address of the port access that caused the SMI trap. Bit D6h[2] indicates whether an I/O read or I/O write access was trapped. (MVA)							

4.8.6.4 Utility Registers

The registers below provide a general purpose storage region and a means of generating warning beeps on the system speaker without modifying the AT-compatible I/O ports.

Index	Name	7	6	5	4	3	2	1	0
52h	Scratchpad Register 1	General purpose storage byte							
53h	Scratchpad Register 2	General purpose storage byte							

Index	Name	7	6	5	4	3	2	1	0
6Ch	Scratchpad Register 3	General purpose storage byte							
6Dh	Scratchpad Register 4	General purpose storage byte							
6Eh	Scratchpad Register 5	General purpose storage byte							
6Fh	Scratchpad Register 6	General purpose storage byte							
51h	Beeper Control Register	General purpose storage bits						Beeper Control 00 = No action 01 = 1KHz 10 = Off 11 = 2KHz	

4.9 System Power Management

The power management unit logic provides two hardware means of controlling the CPU clock speed.

Doze mode hardware causes the CPU speed to slow down or stop to save power when there is no significant activity.

Thermal management hardware forces the CPU speed to be reduced to avoid overheating the CPU when the system is running at full speed for too long.

Both of these mechanisms engage the *stop clock* mechanism to slow down the CPU.

4.9.1 STPCLK# Mechanism to Change CPU Speed

Many CPUs now contain a phase-locked loop (PLL) frequency generator that takes the external clock frequency input and doubles or triples it. Any CPU that depends on an internal PLL for its frequency generation requires an input to allow it to stop operations in an orderly manner and in a known state until a new frequency can be established and locked onto. This type of CPU is referred to as a *PLL-based* CPU, as opposed to a *static* CPU whose clock can be changed at any time. Static CPUs usually (but not always) require a 2X clock input, while PLL-based CPUs almost always use a 1X clock input.

The CPU speed control input is called the stop clock pin (STPCLK#). The name is misleading, since the stop clock protocol must be followed even when switching between clock speeds, not just when stopping the clock altogether. However, the 82C465MV interface uses this same name for clarity. The STPCLK# mechanism is needed as follows.

- Static CPUs may or may not have a stop request input pin. In either case, the CPU can be sped up, slowed down, or stopped at any time. No protocol is required for doze mode on this type of CPU, but if it does have a stop clock input the STPCLK# feature should be activated to save additional power.
- PLL-based CPUs require a stop request signal from the power management unit requesting the CPU to stop all operations and disconnect from its clock input so that the clock can be changed or stopped. This signal is called STPCLK#, SUSP#, DFSREQ#, or similar.
- PLL-based CPUs also must tell the 82C465MV when they are actually ready to allow the clock to be stopped or changed. Some CPUs generate a special bus cycle to indicate this state; when they receive RDY# from the chipset they will go to their stop grant state. Other CPUs have a dedicated pin to indicate this condition to the chipset, called STPGNT#, SUSPA#, DFSRDY#, or similar.

In addition to using a STPCLK# mechanism to change speed, most CPUs provide a clock-stopping feature that allows the CPUCLK input to be completely stopped during periods of no activity. A system interrupt, such as initiated by a keystroke or a timer interrupt, can cause the 82C465MV to restart the CPU almost immediately. Stopping the CPU clock is usually initiated by software (APM for example), but could also be initiated by the hardware doze mechanism.

4.9.1.1 Hardware Considerations

To enable the STPCLK# request logic of the 82C465MV, the logic must sample pins 13 and 23 high at reset. These pins must be strapped high with 10KΩ resistors. Refer to the Strap-Selected Interface Options section for complete details.

4.9.1.2 Programming

To enable STPCLK# operation, the registers shown below must be programmed as follows.

- Bits 65h[6] and 66h[5] are set according to the slowed or stopped CPU clock desired:
  - For completely stopping the CPU clock: Set bit 65h[6] = 1 for latched STPCLK# operation and bit 66h[5] = 1 for stopping the CPU clock during doze mode.
  - For slowing down the CPU clock: Set bit 65h[6] = 0 for pulsed STPCLK# operation and bit 66h[5] = 0 for slowing the CPU clock during doze mode (the slowdown speed is set in bits 41h[4:2] as described later).
- Enable the STPCLK# logic mechanism by setting bit 61h[2] = 1 for any CPU that requires a clock change request signal (STPCLK#, DFSREQ#, SUSP#) to change operating frequency. Even for static CPUs whose frequency can be changed dynamically, if a STPCLK# input is provided it should be used to save power. The additional power savings is substantial, especially in software doze mode when the CPU clock is stopped.
- If STPCLK# is used, enable the stop grant protocol as needed. Almost all 1X CPUs require setting bit 66h[0] = 1 to recognize the stop grant cycle. In addition:
  - CPUs that have a stop grant pin (STPGNT#, DFSRDY#, SUSPA#) require setting bit 66h[3] = 1 and bit 57h[3] = 0 to enable the 82C465MV STPGNT# input on PIO3.
  - For CPUs that do not have a stop grant pin, the STPGNT# input function to the 82C465MV be disabled (bit 66h[3] = 0). Otherwise, the chipset may prematurely detect a "stop granted" condition and will change the clock unconditionally, likely hanging the CPU.
- Enable and set a switching delay through bits B0h[7:0] to select the precise minimum switching delay required for the CPU in use. If disabling the STPCLK# signal, set bits B0h[7:0] = 0 for no delay.

The STPCLK# sequence will now be observed any time the hardware doze feature changes the clock speed, or when APM commands the clock to stop. For example, during an APM stop clock operation the 82C465MV:

1. Asserts its STPCLK# output
2. Waits for either a stop grant cycle or a STPGNT# signal
3. Returns RDY# to the CPU
4. Stops the clock to the CPU
5. Awaits a restart event such as an interrupt
6. Restarts the clock to the CPU
7. Waits for the switching delay time programmed
8. De-asserts its STPCLK# output and continues operation.

The sequence for slowing the clock is similar except that instead of steps 4 through 6 above, the 82C465MV simply switches the clock speed.

The registers associated with the clock speed change mechanism are shown below. Additional detailed descriptive information follows the register bit listings.

Register Bits Associated with STPCLK# Feature

Index	Name	7	6	5	4	3	2	1	0
61h	Debounce Register						STPCLK# signal 0 = Disable 1 = Enable		
65h	DOZE Register		STPCLK# control 0 = Pulse 1 = Latch						

Register Bits Associated with STPCLK# Feature (cont.)

Index	Name	7	6	5	4	3	2	1	0
57h	PMU Control Register 6					PIO3 direction 0 = Input 1 = Output			
66h	PMU Control Register 8			Doze type 0 = Slow CPU clock 1 = Stop CPU clock		Pin 171 0 = PIO3 1 = STPGNT#			CPU clock change protocol required 0 = No 1 = Yes
B0h	Stop-Clock Delay Register	Stop Clock Delay 0 = Disable 1 = Enable	Stop Clock Delay Time Base 0 = 32KHz/4 (~122 us) 1 = FBCLK/4	Delay Count - This value multiplies the time base period selected in bit [6]. There is an additional 6 FBCLK delay for all selections, even "no delay." Sample approximate delays based on 32KHz/ 25MHz selections: 000000 = No delay 000001 = 122us/160ns000010 = 244us/320ns 000011 = 366us/480ns... 001000 = 976us/1.28us001001 = 1.1ms/1.44us... 111111 = 7.7ms/10.0us					

4.9.1.2.1 STPCLK# Pulse/Latch Control

The STPCLK# Pulse/Latch Control bit 65h[6] is meaningful only if the STPCLK# protocol is enabled by bit 61h[2] = 1 and bit 66h[0] = 1. The mechanism operates as follows.

- When STPCLK# is set to "pulse" for changing the frequency of the CPU clock (slowed CPUCLK mode), bits B0h[6:0] determine the duration of the pulse starting from the RDY# responding to the CPU stop grant cycle and ending with STPCLK# going inactive. The 82C465MV logic changes the clock speed just after generating RDY# to the CPU.
- When STPCLK# is set to "latch" for stopping the CPU clock (stopped CPUCLK mode), STPCLK# goes low and stays low. Just after generating the RDY# responding to the CPU stop grant cycle, the logic slows down and then stops the clock. Upon any enabled doze reset event, the 82C465MV logic restarts the clocks at doze speed, then brings them to full operating speed. Bits B0h[6:0] determine the start-up delay between the clocks returning to full speed and the STPCLK# signal going inactive.

The total time STPCLK# is active must also include the time from when the 82C465MV logic sets it active to the point where the CPU responds with a stop grant cycle, which is CPU-dependent.

4.9.1.2.2 Stop Clock Delay Selections

Most currently available CPUs specify a delay of 1ms from CPUCLK stable operation to STPCLK going inactive. To anticipate future CPUs that may require a clock stabilization delay time significantly greater or less than 1ms, the 82C465MV provides a register to offer total delay flexibility. The Stop Clock Delay Time Base bit B0h[6] provides two ranges: 32KHz/4 (for a 122us period), and OSCCLK/4 (for a 200ns period @ 20MHz, for example).

Note that when using the OSCCLK delay range, the values are calculated on the effective input frequency. If a 2X OSCCLK is used, it will be divided by 2 for an effective 1X input clock. In other words, if the input clock is 2X, the delay period becomes OSCCLK/8 to compensate.

Also note that regardless of delay setting, even for no delay, there will always be an additional 6 CPUCLKs of delay over any setting (whether based on CPUCLK or 32KHz). The logic requires this time to engage its sequencer.

4.9.2 Doze Mode

The 82C465MV power management unit includes doze mode control logic. Doze is the state in which the CPU and the 82C465MV chipset are fully alive and operational, yet running at a speed that is greatly reduced in order to save power. The 82C465MV engages doze mode when it sees no activity in certain pre-definable areas for a certain time period. Once initialized by software, the process is completely controlled by hardware. No further software intervention is needed, but an SMI can be generated if desired.

Even though doze mode is intended to operate independently without application or BIOS intervention, the 82C465MV provides logic hooks to software for software-based power control. The most common type of software-based power control follows the Microsoft Advanced Power Management (APM) specification, which allows applications to inform the operating system when

they are idle or do not require full processing power. The operating system, in turn, makes BIOS calls that can do any of the following:

- Turn off or put into a standby mode any unneeded peripherals
- Slow system clock speeds
- Turn off clocks to the CPU.

Therefore, the 82C465MV doze mode logic provides for three doze mode operations: hardware-controlled slowdown, software-controlled slowdown, and software-controlled slowdown with a stopped CPU clock.

The three modes are very similar and are outlined in the sections below, followed by descriptions of registers that the three modes have in common. Note, however, that the hardware mechanism of clock slowing or stopping is dependent on the type of CPU in use. The 82C465MV provides the stop clock logic described below.

4.9.2.1 Dual Doze Timer Reload Selections

The standard 82C465MV part can generate a doze timer timeout in as little as 2ms. However, this selection is not compatible with all applications. For example, stable operating speed might be desirable for at least 2 s after a keyboard interrupt in order to completely service the event and prevent delays on subsequent keystrokes. Another operation, such as video access, might allow a return to doze mode almost immediately.

Therefore, the 82C465MVA part provides a choice of two timeout timers for each device. When an interrupt for the device or access in the range associated with that device occurs, the event triggers a doze reset that reloads the selected timer for that device with the timeout value associated with that timer. Only when both timers have expired will the system return to doze mode operation.

On the original 82C465MV part, COM port, LPT port, and GNR accesses could not cause a doze reset. On the 82C465MVA version, these accesses can be programmed to enable a doze reset. However, to maintain backward compatibility with the 82C465MV and 82C463MV, the COM1&2, LPT, and GNR accesses point to the secondary doze timer at reset; this timer in turn is programmed for "no delay" at reset. The doze logic interprets the "no delay" setting as inhibiting doze reset for that source.

The SMI, EPMI1, and INTR signals are also potential sources of doze reset. However, these signals always use doze timeout 0 and cannot select doze timeout 1.

Regarding SMI generation on doze events: Doze timeout events on DOZE_0 and DOZE_1 can be individually programmed to generate an SMI through register bits D9h[7:6], which are redefined on the 82C465MVA part to select the timeout(s) that will cause an SMI. Doze reset events are all individually programmable to generate SMIs.

Doze Timeout Control Registers

Index	Name	7	6	5	4	3	2	1	0
76h	Doze Reload Select Register 1	LCD_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	KBD_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	DSK_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	HDU_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	COM1&2_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	LPT_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	GNR1_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	GNR2_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)
77h	Doze Reload Select Register 2	IRQ8 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ7 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ6 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ5 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ4 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ3 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ1 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ0 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)
78h	Doze Reload Select Register 3	LDEV# 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ15 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ14 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ13 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ12 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ11 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ10 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ9 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)

Doze Timeout Control Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
D9h	PMU Event Register 6	DOZE_TIMER PMI #27 SMI 00=Disable 01=Enable DOZE_0 (MVA) 10=Enable DOZE_1 (MVA) 11=Enable both							
41h	PMU Control Register 2	DOZE_0 Timeout Select 0 0 0 = 2 ms 0 0 1 = 4 ms 0 1 0 = 8 ms 0 1 1 = 32 ms 1 0 0 = 128 ms 1 0 1 = 512 ms 1 1 0 = 2 s 1 1 1 = 8 s							
79h	PMU Control Register 11	DOZE_1 Timeout Select (MVA) 0 0 0 = No delay (default) 0 0 1 = 1 ms 0 1 0 = 4 ms 0 1 1 = 16 ms 1 0 0 = 64 ms 1 0 1 = 256 ms 1 1 0 = 1 s 1 1 1 = 4 s							

4.9.2.2 Presetting Events to Reset Doze Mode

Before enabling doze mode operation, whether hardware or software doze mode, some preparation must be made for the event or events that will reset doze mode and bring the system back to full operation. Otherwise, especially in the case of APM stop clock mode, there would be no way to execute CPU instructions to restart the CPU clock.

Therefore, it is first necessary to choose the source or sources that will perform a doze reset. Doze reset will, if the system is currently in doze mode, restore the system clocks to full operating speed. Doze reset also reloads the doze timer with its original programmed value.

- Setting bit 41h[1] = 1 enables LCD_ACCESS, KBD_ACCESS, DSK_ACCESS, and HDU_ACCESS to reset doze mode and reload the DOZE_TIMER. If the DOZE_TIMER has timed out and switched operation to doze speed, this reload will change the system clocks back to their normal speed. Obviously, this bit has no effect if stopped (not just slowed) CPUCLK operation is programmed.
- Bit 65h[5] selects the EPMI1 pin as a doze reset trigger (level-triggered).
- Bits 62h[7:0], A2h[5:0], and 65h[3] define individual IRQs that can trigger a doze reset.
- Bit 65h[7] allows all enabled interrupts, i.e. any event that toggles the INTR signal to the CPU, to reset doze mode.

Once the doze mode reset events have been programmed, either hardware or software doze mode can be enabled as described in the following sections.

Register Bits that Select Doze Mode Reset Events

Index	Name	7	6	5	4	3	2	1	0
41h	PMU Control Register 2							LCD, DSK, KBD, HDU _ACCESS events reset doze mode 0 = Disable 1 = Enable	
62h	IRQ Doze Register 1 Write-only	IRQ13 doze reset 0 = Disable 1 = Enable	IRQ8 doze reset 0 = Disable 1 = Enable	IRQ7 doze reset 0 = Disable 1 = Enable	IRQ12 doze reset 0 = Disable 1 = Enable	IRQ5 doze reset 0 = Disable 1 = Enable	IRQ4 doze reset 0 = Disable 1 = Enable	IRQ3 doze reset 0 = Disable 1 = Enable	IRQ0 doze reset 0 = Disable 1 = Enable

Register Bits that Select Doze Mode Reset Events (cont.)

Index	Name	7	6	5	4	3	2	1	0
A2h	IRQ Doze Register 2 Write only			IRQ15 doze reset 0 = Disable 1 = Enable	IRQ14 doze reset 0 = Disable 1 = Enable	IRQ11 doze reset 0 = Disable 1 = Enable	IRQ10 doze reset 0 = Disable 1 = Enable	IRQ9 doze reset 0 = Disable 1 = Enable	IRQ6 doze reset 0 = Disable 1 = Enable
65h	DOZE Register	All interrupts to CPU reset doze mode 0 = Disable 1 = Enable		EPMI1 doze reset 0 = Disable 1 = Enable	SMI resets doze mode 0 = No 1 = Yes	IRQ1 doze reset 0 = Disable 1 = Enable			

4.9.2.3 LDEV# Doze Reset

Activity on the local bus can reset doze mode and cause a return to full operating speed. The 82C465MVA logic provides two bits to enable doze reset separately for local bus I/O accesses and local bus memory accesses. The doze reset is triggered by LDEV# going active and is qualified by the M/IO# signal.

Local Bus Doze Reset Registers

Index	Name	7	6	5	4	3	2	1	0
A2h	IRQ Doze Register 2 Write only	Local bus I/O access doze reset 0=Disable 1=Enable (MVA)	Local bus memory access doze reset 0=Disable 1=Enable (MVA)						
78h	Doze Reload Select Register 3	LDEV# 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)							

4.9.2.4 Doze Reset Inside SMM

The 82C465MVA part allows an SMI to reset doze mode if the clock is stopped from *within* system management mode. Bit 65h[4] in the 82C465MV part enables doze mode reset when the SMI signal goes active, but since SMI is masked on entry to SMM, an SMI triggered while the system is in SMM is not seen until some other trigger resets doze mode.

Setting bit 79h[4]=1 handles doze reset only from within SMM. Set bit 65h[4]=1 to handle SMI doze mode exit from outside of SMM.

Doze Reset Bit

Index	Name	7	6	5	4	3	2	1	0
79h	PMU Control Register 11				SMI resets doze mode if clock is stopped inside SMM 0=No 1=Yes				

4.9.2.5 Automatic (Hardware) Doze Mode

The chipset can be set up for hardware-controlled slowdown doze mode by programming the following information.

1. Set up the hardware and programming to consider the stop clock mechanism as described in the "Clock Speed Control for CPU Clocks" section.
2. Program the events that will reset doze mode as described in the "Presetting Events to Reset Doze Mode" section.
3. Select the timeout, that is, the time required after the last event before the system can be considered "inactive," in bits 41h[7:5]. A two second timeout is typical.

4. Select the clock divisor from bits 41h[4:2]. When choosing a divisor, keep in mind that most CPUs cannot run below 8MHz, while the limit for clock-tripled CPUs is usually 12.5MHz.
5. Set bit 66h[5] = 0 for slow-clock mode as opposed to stop-clock mode.
6. Set bit 65h[4] = 1 if chipset should exit doze mode for SMIs, or = 0 if the SMIs can run adequately at the doze speed. Note that since the clock change delay is typically 1ms, it may be practical to simply run the SMI code at the slower speed.
7. Finally, enable the hardware DOZE_TIMER by setting bit 41h[0] = 0.

After the selected period of inactivity, the CPU clock and the chipset clock will automatically be set to a low-speed mode. On the event of any of the enabled accesses, SMIs, or IRQs, the clock will again speed up for fully active operation.

Hardware Doze Mode Registers

Index	Name	7	6	5	4	3	2	1	0
41h	PMU Control Register 2	DOZE_TIMER timeout select 0 0 0 = 2 ms 0 0 1 = 4 ms 0 1 0 = 8 ms 0 1 1 = 32 ms 1 0 0 = 128 ms 1 0 1 = 512ms 1 1 0 = 2 s 1 1 1 = 8 s			Doze mode system clock speed 0 0 0 = OSCCLK / 1 0 0 1 = OSCCLK / 2 0 1 0 = OSCCLK / 4 0 1 1 = OSCCLK / 8 1 0 0 = OSCCLK / 16 1 0 1 = OSCCLK / 3 1 1 0 = reserved 1 1 1 = reserved				Doze control select 0 = Hardware 1 = Software
50h	PMU Control Register 5					Write = 1 to Start Doze Read: doze status 0 = counting 1 = timed out			

4.9.2.6 APM (Software) Doze Mode

The chipset can be set up for software-initiated, slowed or stopped CPUCLK doze mode in a very straightforward manner.

1. Set up the hardware and programming to consider the stop clock mechanism as described in the "Clock Speed Control for CPU Clocks" section.
2. Program the events that will reset doze mode as described in the "Presetting Events to Reset Doze Mode" section.
3. Set bit 65h[4] = 1 if chipset should exit doze mode for SMIs, or = 0 if the SMIs can run adequately at the doze speed. Bit 65h[4] must be set to 1 for stop clock operation or else the SMI will be missed altogether.
4. Select the clock divisor from bits 41h[4:2]. When choosing a divisor, keep in mind that most CPUs cannot run below 8MHz, while the limit for clock-tripled CPUs is usually 12.5MHz. If the CPU clock will be stopped, ignore the lower CPU limit and use a very low clock speed to save power to the chipset (whose clock is also slowed).
5. Disable the hardware DOZE_TIMER by setting bit 41h[0] = 1.

At this point the system is ready for APM control. When APM makes a call for low or very low power operation, the BIOS or power management code simply:

- For slow clock doze: Sets bit 65h[6] = 0 and bit 66h[5] = 0
  -- OR --
  For stop clock doze: Sets bit 65h[6] = 1 and bit 66h[5] = 1
- Sets bit 50h[3] = 1 to initiate the doze mode.

On the event of any of the enabled accesses, SMIs, or IRQs, the clock will again speed up for fully active operation.

Software Doze Mode Registers

Index	Name	7	6	5	4	3	2	1	0
41h	PMU Control Register 2				Doze mode system clock speed 0 0 0 = OSCCLK / 1 0 0 1 = OSCCLK / 2 0 1 0 = OSCCLK / 4 0 1 1 = OSCCLK / 8 1 0 0 = OSCCLK / 16 1 0 1 = OSCCLK / 3 1 1 0 = reserved 1 1 1 = reserved				Doze control select 0 = Hardware 1 = Software
50h	PMU Control Register 5					Write = 1 to Start Doze Read: doze status 0 = counting 1 = timed out			

4.9.2.7 Start Doze Bit

Bit 50h[3] serves two purposes: to start doze mode and to read the DOZE_TIMER status.

- Write: Start APM Doze mode
  '1' = start doze mode (if bit 40h[0] = 1)
  '0' = no effect

- Read: Hardware DOZE_TIMER timeout status bit
  '1' = Hardware DOZE_TIMER has timed out
  '0' = Hardware DOZE_TIMER still counting

4.9.2.8 Using Doze Timeout to Trigger an SMI

In addition to the ability to reset doze mode when an SMI is encountered, the 82C465MV has the ability to generate PMI#27 when the DOZE_TIMER times out. Setting bits D9h[7:6] = 11 enables the PMI to generate an SMI.

Index	Name	7	6	5	4	3	2	1	0
D9h	PMU Event Register 6	DOZE_TIMER PMI #27 SMI 00 = Disable 01 = Enable DOZE_0 (MVA) 10 = Enable DOZE_1 (MVA) 11 = Enable							

4.9.3 CPU Thermal Management Unit

Thermal management hardware is implemented in the 82C465MV for monitoring the level of CPU activity and the operating temperature of the device. A flexible hardware scheme assesses CPU activity to determine when it is necessary to enter Cool-down Clocking Mode. In addition, an external sensor can force the 82C465MV permanently into Cool-down Clocking Mode according to the parameters programmed for thermal management. In this way, a serious over-temperature condition cannot get out of control (as a result of "thermal runaway").

4.9.3.1 Prediction of Overtemp Activity

Thermal management logic is implemented in the 82C465MV for monitoring the level of CPU activity to determine its current draw, and thus approximate the operating temperature of the device. The most obvious way to do this would be to simply count the number of CPU clocks that occur in a given time period. However, a ripple counter running in the 20-40MHz range would consume a great deal of power. Instead, 82C465MV logic assesses CPU activity periodically and keeps track of how often the CPU exceeds the safety limits to determine whether automatic intervention is called for.

4.9.3.1.1 Operating Temperature Ranges

The 82C465MV thermal management algorithm identifies the temperature limits of any CPU through activity level values that correspond with idle, equilibrium, and thermal runaway conditions.

- Idle Condition - The CPU is cool to the touch. The 82C465MV is using APM stop clock or hardware doze mode to save power, and no real activity is taking place. This operational level constitutes the base level of activity, so it does not require a register to hold the value. It is associated in the 82C465MV thermal management scheme with "zero".
- Equilibrium Condition - The CPU is warm to the touch. It is operating at full speed for short bursts but frequently is at slow speed or stopped. The heat generated by the CPU is dissipated at the same rate that it is produced. Most active computer usage falls into this category, where APM or hardware doze mode operates frequently enough to allow safe operation. The 82C465MV thermal management scheme associates this temperature with Equilibrium Level bits EQL6:0. The value of these bits is referred to as EQL throughout this section.
- Thermal Runaway Condition - The CPU is hot to the touch. It is running at its full rated speed and generating heat faster than it can be dissipated, so its temperature increases. The program being used is active enough that APM or hardware doze mode cannot operate often enough to hold the temperature down. Operating in this mode for an extended period may cause damage to the CPU. The 82C465MV thermal management scheme associates this temperature with Overtemp Limit bits OTL7:0. The value of these bits is referred to as OTL throughout this section.

4.9.3.1.2 Accounting for CPU Activity

82C465MV logic frequently assesses CPU activity, according to the current CPU running mode. Each CPU run mode is associated with a *power level increment* as shown in Table 4-27. Using these values, the 82C465MV can keep a running count of the average power being used by the CPU in the internal 24-bit CPU Activity Counter (CNT).

The 82C465MV must also be able to account for the CPU type. For example, a clock-doubled or -tripled CPU at full speed will heat up much faster, yet at idle will cool down at approximately the same rate, as a non clock-multiplied CPU. The CPU Efficiency bits CPUE1:0 are used to indicate the relative CPU current consumption and are explained below.

Table 4-27 Power Levels Assigned to Each Operating Mode

Current CPU Operating Mode	Power Level Increment
Full Speed	+2
Divide by 2 or 3	+1
Divide by 4	0
Divide by 8	-1
Divide by 16 or more	-2
APM Stop Clock	-2

The thermal management hardware keeps track of CPU activity as follows. The logic checks the current CPU operating mode either 32k, 16k, 8k, or 4k times per second according to the CDHO1:0 bits setting as explained below. The thermal management logic notes the current operating mode to account for the instantaneous CPU current consumption by incrementing or decrementing the CPU Activity Counter as follows.

- For all CPU operating modes, the logic increments or decrements the counter by the value listed in Table 4-27 for the operating mode at that time.
- Only for those modes with a power level increment above zero, the logic also increments the CPU activity counter by the CPU Efficiency bits value.

For example, if the CPU is sampled at full speed (+2), and CPUE1:0 = 2, the activity counter will be incremented by 4. However, if the CPU is sampled at divide by 8 (-1), the counter will simply be decremented by 1. The CPUE value is not added in if the power level increment is zero or below because the cool-down rate is independent of CPU type.

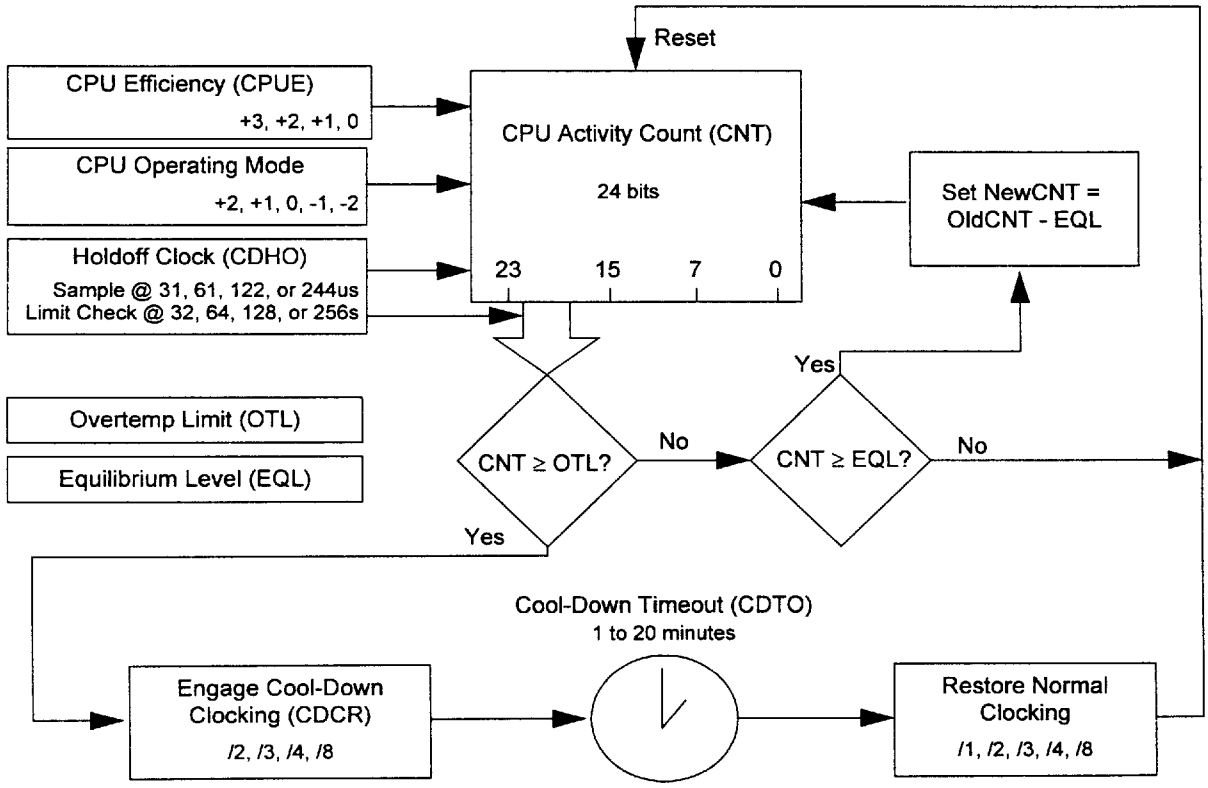
4.9.3.1.3 Determination of Operating Temperature Range

Periodically, the thermal management logic compares the upper byte of the CPU activity counter to the Overtemp Limit (OTL) and to the Equilibrium Level (EQL). This comparison takes place every 32, 64, 128, or 256 seconds as programmed in the Cool-down Holdoff bits CDHO1:0. These bits also select the frequency of sampling, at 32k, 16k, 8k, or 4k times per second respectively, such that the value accumulated in the CPU activity count will, on average, be the same whether a short or a long holdoff period is selected. The logic acts on the comparison results as follows.

- If the upper byte of the activity count is greater than or equal to OTL, the thermal management unit initiates Cool-down Clocking.
- If the upper byte of the activity count is above EQL but below OTL, EQL is subtracted from the upper byte of the CPU Activity Counter and the result is returned to the counter. This scheme handles operation consistently above equilibrium level that may, over time, result in excessively high CPU temperatures.
- If the upper byte of the activity count is equal to or below EQL, the CPU Activity Counter is simply cleared and the thermal management logic starts a new detection period from "zero" (idle condition).

Cool-down Clocking. Cool-down Clocking takes place according to the reduced clock rate programmed into the Cool-down Clock Rate bits CDCR1:0. The 82C465MV keeps the CPU running at the Cool-down Clocking Rate speed for the Cool-down Holdoff period multiplied by the Cool-down Timeout value programmed. At the end of the period, the activity counter is cleared and normal clocking is restored. No activity accounting takes place while Cool-down Clocking is engaged.

Figure 4-7 Thermal Management Block Diagram



4.9.3.2 Example

A 25/75MHz CPU is being used with clock tripling enabled. The Cool-down Clock Rate is set to divide-by-2 so that even when slowed down the minimum clock speed will be above 12MHz. The CPU Efficiency bits are set to 3 because of the clock tripler. Cool-down Holdoff is set for 64 seconds; Cool-down Timeout is set for 3x; the Equilibrium Level is set to 40h; and the Overtemp Limit is set to 7Fh.

Consider the situation after the CPU has run at full speed for one minute. The thermal management hardware is checking the CPU operating mode every 61us (16,384 times per second), and each time increments the CPU Activity Counter by 2 to account for the CPU running at full speed and then by 3 to account for the CPU Efficiency setting. Thus, the count is incremented by 16,384 x 5 each second for 64 seconds, for a total count of 500000h. The value of the high byte is 50h; this value is compared to EQL, which was set to 40h. Since the activity count exceeds the equilibrium level, EQL is subtracted from the activity count and the result (10h) remains in the counter. However, the activity count of 50h does not exceed the OTL value of 7Fh. Therefore, no other action is taken.

Now consider the situation after the CPU has run at full speed for four minutes. After each 64-second interval, the high-order activity count byte has been incremented by 50h, and after the OTL comparison is made the EQL value of 40h has been subtracted off for a net increase of 10h. So after four minutes of operation, when the OTL comparison is made, the activity count of 80h is compared to the OTL value of 7Fh and an over-limit situation is detected.

Therefore, the thermal management hardware engages Cool-down Clocking at 25MHz/2 for three minutes (3x 64 seconds). It then resets the CPU Activity Counter, resumes normal clocking and monitoring, and starts the whole detection process over again.

This situation would have been avoided if APM Stop Clock mode had been entered for at least 10 seconds during each 64-second period. In that case, the thermal management hardware would have counted (16384 x 5 x 54) + (16384 x -2 x 10) = 3E8000h each time, just under the 40h equilibrium limit. The CPU Activity Counter would have been reset each time and no thermal buildup would have been detected.

4.9.3.3 Programming

Before programming thermal management, the STPCLK# mechanism must be set up for the CPU being used as described in the "STPCLK# Mechanism to Change CPU Speed" section of this document. Enabling thermal management locks the STPCLK# bits and prevents them from being altered until the next hardware reset.

The Thermal Management option must be configured by setting the bits in registers A5h, A6h, and A7h, and then setting register A5h bit 7 = 1. The register setting order is not important, but bit 7 of A5h must be set to 1 only after all other settings have been made. Once bit 7 is set, none of the thermal management registers, nor bits 61h[2], 66h[0], and B0h[7:0], can be written again without resetting the 82C465MV chip.

4.9.3.3.1 SMI Generation

When the thermal management unit engages or disengages Cool-down Clocking, an SMI on PMI#25 can be generated. This feature is controlled through bit DBh[6]. When this SMI is being serviced, power management code can read bit A5h[7] to determine whether it was an entry into or an exit from Cool-down Clocking mode that caused the SMI.

PMI#25 is also shared with the EPMI4 event. If the SMI from EPMI4 is also enabled, on entry to the SMI software must check the state of the EPMI4 signal to determine whether the reason for the SMI is the external EPMI4 event or Cool-down Clocking entry/exit.

Thermal Management Registers

Index	Name	7	6	5	4	3	2	1	0
A5h	Thermal Management Register 1	Thermal Management 0 = Disable 1 = Enable	Equilibrium Level (EQL6:0) This count corresponds to equilibrium operation. If the CPU Activity Counter exceeds EQL, EQL is simply subtracted from the upper activity count byte and sampling continues. If the count is below EQL, the count is cleared						
A6h	Thermal Management Register 2	Overtemp Limit (OTL7:0) This count corresponds to an over-temperature situation. If the CPU Activity Counter exceeds OTL, the 82C465MV engages Cool-down Clocking.							

Thermal Management Registers (cont.)

Index	Name	7	6	5	4	3	2	1	0
A7h	Thermal Management Register 3	CPU Efficiency (CPUE1:0) 00 = Low power 01 = Moderate 10 = High 11 = Very high		Cool-down Holdoff (CDHO1:0) 00 = 32 seconds 01 = 64 seconds 10 = 128 seconds 11 = 256 seconds		Cool-down Clock Rate (CDCR1:0) 00 = /2 01 = /3 10 = /4 11 = /8		Cool-down Timeout (CDTO1:0) 00 = 2x CDHO 01 = 3x 10 = 4x 11 = 5x	

Thermal Management Enable bit A5h[7] - once written to 1, none of the thermal management registers can be cleared without a system reset. When read, this bit returns 1 only if Cool-down Clocking is currently taking place.

Cool-down Holdoff bits A7h[5:4] - specify the time period of observation to allow before checking whether to enable Cool-down Clocking.

Cool-down Clock Rate bits A7h[3:2] - specify the CPU clock divisor to use during Cool-down Clocking.

Cool-down Timeout bits A7h[1:0] - specify the length of time, in terms of the Cool-down Holdoff selected, that the Cool-down Clocking will continue before normal operation is restored.

4.9.4 Emergency Overtemp Sense

It is possible for an external sensor to force the 82C465MV into Cool-down Clocking mode according to the parameters programmed for thermal management. When low, the EPMI2 input can cause the 82C465MV to enter Cool-down Clocking mode. The existing SMI enable bits for EPMI2 are still operational regardless of the setting of the emergency overtemp enable bit. Therefore, an overtemp condition can also be programmed to cause an SMI so that the power management firmware will be made aware of the situation and instruct the user to shut down the system. In this way, a serious over-temperature condition cannot get out of control.

The chip will remain in Cool-down Clocking mode, using the rate specified in CPU Thermal Management Register 3 (defaults to divide-by-2), as long as the EPMI2 input remains triggered. The trigger specifications are the same as those for triggering the SMI: bit 40h[2] selects whether a high level or a low level is active, and this same bit selects whether a high level or a low level will engage Cool-down Clocking. The thermal management unit does not need to be enabled to use this feature.

EPMI2 can be used as the overtemp sense input even when the EPMI2 function has been moved to an external 74153 multiplexer (when the standard DACKMUX interface feature has been selected).

4.9.4.1 Programming

The emergency overtemp sense option is enabled by writing bit DBh[6] as explained in the SMI Event Generation section. Once written, this bit cannot be changed without a hard reset of the chip.

When bit DBh[6] = 1, entry into or exit from Cool-down Clocking mode causes PMI#25 and an SMI. If the EPMI2 event is also programmed to cause an SMI, the following situation can occur.

1. The thermal sensor input changes state, causing PMI#2 and an SMI.
2. Power management code services the SMI.
3. The thermal management unit generates a stop clock request.
4. Once the CPU generates a stop grant cycle, the chipset reduces the clock speed to enter Cool-down Clocking mode.
5. At this point, PMI#25 is generated along with another SMI.

Therefore, two SMIs will have been generated. On exit from Cool-down Clocking mode, only one SMI will be generated, since the active-to-inactive transition on EPMI2 does not cause another SMI. Power management software must be able to anticipate this situation and deal with it appropriately.

4.10 Suspend and Resume

The 82C465MV offers the ability to halt operations at extremely low power yet retain all its programming, called suspend. The chipset will respond to interrupts to determine that a return to normal operation, called resume, is necessary.

4.10.1 Suspend Mode

Suspend mode provides a significant level of power conservation. The suspend initiation event, either a key or button depression or a timeout SMI, calls a software routine in SMM code to save the current state of the system for complete restoration at some later time. In this mode, most system power can be shut down while still retaining the ability to restore the previous context. The 82C465MV can either stop the clock to a still-powered CPU, or engage leakage controls with a CPU that must be powered down during suspend. The leakage control function is enabled by either floating or driving low all critical interface nodes to reduce power consumption to a bare minimum.

The 82C465MV enters suspend mode when bit 50h[0] is set to 1. Software must control this event, even though a timer timeout may have initiated the process, because CPU processing must be completed in an orderly fashion.

Upon resuming from suspend mode, the controlling code must clear the Suspend PMI event, PMI#7, by writing bit 5Ch[7] = 1. Otherwise, the chip will never leave suspend mode the next time bit 50h[0] is set to 1.

The registers shown below select the state of various signals during suspend mode. Refer to the "Resume Event" section that follows to determine how to select the events that will cause the chip to resume operation after suspend.

Suspend Control Register Bits

Index	Name	7	6	5	4	3	2	1	0
ADh	Feature Control Register 3			CPU Power State in Suspend 0 = Powered 1 = 0 volt					
59h	PMU Event Register 2		Reload timers on resume 0 = No 1 = Yes						
50h	PMU Control Register 5								Start suspend (write-only) 1 = Enter suspend mode

Reload timers on resume - The system timers can be restarted to prevent false timeouts upon resuming from suspend mode.

4.10.1.1 Suspend Mode Power Savings

The 82C465MV can be preset in various ways before entering suspend mode in order to minimize the current consumption while suspended.

4.10.1.1.1 Resistor Control Registers

The Resistor Control Registers control and/or disable the automatic internal pull-down resistors on various lines during suspend mode. These registers are described in the "Chip-Level Power Conservation Features" section of this document.

4.10.1.1.2 Short-Pulse Refresh

The short pulse refresh mechanism should be engaged for lowest DRAM power consumption during suspend. The suspend refresh rate is selected through bits 67h[6:5], the same bits that select the refresh rate for active operation. However, there is a major difference between active mode refresh and suspend mode refresh: active-mode refresh pulse width is controlled by the OSCCLK input clock, while the suspend-mode refresh pulse width is generated from the SQWIN clock and therefore has the same pulse width as that clock. For a 32KHz input, this pulse width is 15us which will cause DRAM to consume extra power.

Setting bit A1h[6] = 1 allows the refresh pulse width to be narrowed to approximately 100ns and should always be selected if appropriate for the DRAM in use.

4.10.1.1.3 Self-Refresh DRAM

Suspend refresh can be eliminated altogether through bit 66h[7] for self-refresh DRAM. If "none" is selected for suspend refresh, only a single low pulse sequence is generated to put the DRAM into self-refresh mode.

4.10.1.1.4 Interrupt Scan Rate

The logic can scan for interrupts at a slower rate, at 122us intervals instead of 280ns intervals. This feature uses the SQWIN clock instead of the 14.318MHz clock to generate KBCLK and KBCLK2, so any keyboard controller in use must be able to tolerate this slower clocking speed. Bit 66h[6] controls the KBCLK rate to be applied during suspend mode.

Note that a major reduction of power consumption can come from switching off the 14.318MHz clock generator circuit of the clock generator chip in use when KBCLK runs from SQWIN.

4.10.1.1.5 Suspend Mode HOLD Control

Asserting HOLD to the CPU before stopping the clock is used to tri-state the CPU signals during suspend. This feature may or may not be useful, as determined by the system design.

- Setting suspend mode HOLD control bit 66h[4] = 0 drives HOLD high during suspend. This setting is useful if some devices on the VL bus are powered off during suspend. Many CPUs drive output signals to their last state while in stop grant state, which could power up unpowered devices on the bus. These CPUs tri-state the outputs if in stop grant state and give the system HLDA.
- Setting suspend mode HOLD control bit 66h[4] = 1 does not drive HOLD during suspend. This setting is useful in conjunction with the zero-volt CPU suspend option described in the "Special CPU Interface Support" section, because it allows all CPU interface signals to go low at suspend time.

The correct setting of the bit depends on the VL-bus power-down scheme used in the design.

Suspend Mode Power Saving Feature Bits

Index	Name	7	6	5	4	3	2	1	0
66h	PMU Control Register 8	Suspend refresh 0 = Slow 1 = None (for self-refresh DRAM)	Suspend KBCLK source 0 = 14MHz/2 1 = 32KHz/2		Assert HOLD during suspend 0 = Yes 1 = No				
A1h	Feature Control Register 2		Suspend Refresh Pulse Gen 0 = Wide 1 = ~100ns						

4.10.1.1.6 Suspend Refresh Pulse Width Control

The suspend mode refresh pulse on the 82C465 series part is generated by gate delays. These gate delays vary significantly according to the voltage at which the part operates. At 5V core operation, the pulse width might be 150-250ns. If the core operates at 3.3V, the pulse width might be 400-500ns. Therefore, the 82C465MVB part provides bit 3Fh[6] to select fewer gates in the delay path. At its default setting of 0, the pulse width is comparable with that of the 82C465MVA part. When set to 1, the pulse width is approximately one half of the normal value. The setting of 1 is probably the best setting for 3.3V core operation.

Suspend Refresh Pulse Width Control

Index	Name	7	6	5	4	3	2	1	0
3Fh	Misc Control Register		Suspend Refresh Pulse Width 0=Normal 1=Reduced (MVB)						

4.10.2 Resume Event

A certain set of interrupt events can be enabled to resume the system from suspend mode. The desired interrupts are grouped *into a single event, called* RSMGRP. RSMGRP can be enabled to generate an SMI if desired. The RI input and the SUSP/RSM input can also trigger a resume, and can also be enabled to generate an SMI if desired. Any one or more of the RSMGRP, RI, and SUSP/RSM events are called a resume event.

4.10.2.1 EPMI/IRQ Events

The registers at indexes 6Ah and B1h select the EPMI and IRQ source(s) that will be allowed to trigger the system out of suspend mode. Once selected, setting bit 5Fh[5] = 1 enables the RSMGRP globally. On resume, an SMI can be generated either from the EPMI events (through registers 58h and D9h) or the PMI#6 event (through register 59h). However, since the system usually is still in SMM when the resume takes place, SMI generation is not normally necessary.

The IRQ and EPMI resume enabling bits are shown in the Resume Event Registers below. Default for all bits is disabled. A rising edge on the enabled signal causes the resume event for all selections *except* IRQ8; it is a falling edge on IRQ8 that resumes operation.

4.10.2.2 SUSP/RSM and RI Events

When the RI input pin toggles enough to exceed the count set in bits 5Fh[3:0], and bit 5Fh[4] = 1, PMI#6 is generated to exit suspend-mode. Signal RI should be high for a minimum of 240ms and low for a minimum of 60ms when changing states. The RI input is sampled with a 32KHz clock; therefore rapid or unstable transitions may lead to unreliable counting.

The SUSP/RSM input pin is always enabled to resume the system, and should be pulled high if it will not be used in the system design. Resuming from SUSP/RSM generates PMI#6.

Resume Event Registers

Index	Name	7	6	5	4	3	2	1	0
6Ah	RSMGRP IRQ Register 1	EPMI2 resume 0 = Disable 1 = Enable	EPMI1 resume 0 = Disable 1 = Enable	IRQ8 resume 0 = Disable 1 = Enable	IRQ7 resume 0 = Disable 1 = Enable	IRQ5 resume 0 = Disable 1 = Enable	IRQ4 resume 0 = Disable 1 = Enable	IRQ3 resume 0 = Disable 1 = Enable	IRQ1 resume 0 = Disable 1 = Enable
B1h	RSMGRP IRQ Register 2	EPMI4 resume 0 = Disable 1 = Enable	EPMI3 resume 0 = Disable 1 = Enable	IRQ15 resume 0 = Disable 1 = Enable	IRQ14 resume 0 = Disable 1 = Enable	IRQ12 resume 0 = Disable 1 = Enable	IRQ11 resume 0 = Disable 1 = Enable	IRQ10 resume 0 = Disable 1 = Enable	IRQ9 resume 0 = Disable 1 = Enable
5Fh	PMU Control Register 7			RSMGRP IRQs can resume system 0 = No 1 = Yes	Transitions on RI can resume system 0 = No 1 = Yes	Number of RI transitions to cause resume			

Once a resume event has occurred, bits 6Bh[2:0] should be read to determine the source(s). If 6Bh[1] = 1, a read of register 6Ah and B1h will return the latched state of the any of the EPMI or IRQ lines that were originally enabled for resume triggering. The latched resume IRQ and EPMI source information in 6Ah and B1h is available until the PMI#6 bit (bit 5Ch[6]) is eventually written to 1 to clear the PMI generated. Bit 50h[1] = 1 as long as the resume PMI#6 remains active.

Resume Sources (read-only)

Index	Name	7	6	5	4	3	2	1	0
50h	PMU Control Register 5						Ready to resume Read only. 0 = Not in resume 1 = Ready to resume	PMU mode Read only. 0 = Suspend PMI not pending 1 = Suspend active (clear PMI#6)	

Resume Sources (read-only) (cont.)

Index	Name	7	6	5	4	3	2	1	0
6Bh	Resume Source Register						SUSP/RSM caused resume Read-only 0 = No 1 = Yes	RSMGRP caused resume Read-only 0 = No 1 = Yes	RI caused resume Read-only 0 = No 1 = Yes

4.10.3 Chip-Level Power Conservation Features

A central design goal of the 82C465MV was to incorporate power-reducing features wherever possible. To this end, several innovative methods of power conservation are implemented.

4.10.3.1 Automatic Keeper Resistors

Since there are times during normal operation in which the CPU tri-states many of its output signals and no other source is driving these signals, the lines tend to float between logic transition levels. When this results in an oscillation, a substantial amount of current is consumed. For this reason, external pull-up or pull-down resistors are typically connected on these lines. Even if resistors are integrated into the chipset itself, considerable current would be consumed during normal operation when the logic is active and is driving against these resistors.

The 82C465MV circuitry provides the option of internal 50KΩ pull-down resistors on certain lines. The resistors are automatically engaged only during bus hold and suspend mode. Also, the chip provides an option to enable the resistors all the time. It is theoretically possible that during long periods with IOCHRDY low, the temporary tri-state condition of the CPU and 82C465MV interfaces could consume more power than would be needed on average to leave the pull-down resistors engaged constantly. The system designer must determine the setting that is appropriate.

The resistors are on CA31, CA[25:2], CD[31:0], BE3:0#, W/R#, D/C#, and M/IO#. The resistors serve to prevent the buses from floating and consuming power during suspend mode, as well as to hold the upper address lines low and the CPU control lines inactive during bus hold cycles to prevent the logic from misinterpreting cycle types and destinations during DMA operations. This feature can be used safely with the zero-volt CPU suspend option, since the resistors always pull down when engaged. Setting bit A0h[6] = 1 enables this feature.

The 82C465MV additionally manages the FERR# pin when this feature is enabled, providing a pull-up resistor that is engaged while the chip is active and a pull-down resistor when the chip is in suspend mode. This feature eliminates the need for external control of FERR# depending on the type of CPU installed. The registers also control the DACKMUX0-2 signal states during suspend, regardless of where these signals have been relocated on the chip.

Resistor Control Registers

Index	Name	7	6	5	4	3	2	1	0
A0h	Feature Control Register 1		Automatic Internal Resistors 0 = Disable 1 = Enable						
D4h	Resistor Control Register 1	CA31, CA[25:2], BE[3:0]# Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved		CD[31:0] Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved		M/IO#, D/C#, W/R# Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved			
D5h	Resistor Control Register 2	DACKMUX Control During Suspend 00 = Pull resistor-equipped lines low, drive others low 01 = Tri-state all lines (463MV mode) 10 = Drive 100b on DACKMUX2-0 11 = Reserved		FERR# Pull-up Resistor Control 0 = Always enabled 1 = Always disabled					

Note that FERR# has an internal pull-*up* resistor.

4.10.3.2 Zero-Volt CPU Suspend

Most CPUs designed for portable applications consume negligible power when their input clock is stopped. Some CPUs draw too much power to be left powered when the system is in suspend mode, yet it is still desirable to obtain the rapid start-up available only through copying CPU context into DRAM and keeping DRAM alive. Therefore, the 82C465MV CPU interface provides a zero-volt suspend option.

Setting bit ADh[5] = 1 enables zero-volt CPU suspend. When set, the 82C465MV will condition its outputs during suspend assuming that the CPU has been powered down completely. The affected 82C465MV output signals are listed in the "Pin Description" section of this document. Signals that would normally be maintained high to the CPU while in suspend mode are instead tri-stated.

This feature is generally used in conjunction with a feature on the SRESET pin which, in this case, is used as the general-purpose CPU reset (not as a software reset). By setting bit ADh[3] = 1, the SRESET signal will toggle on resume from suspend to reset a CPU that has been powered-down. Refer to the "Special CPU Interface Support" section of this document for more information.

Note also that when bit ADh[5] = 1, CPURST will always be generated upon resuming.

Index	Name	7	6	5	4	3	2	1	0
ADh	Feature Control Register 3			CPU Power State in Suspend 0 = Powered 1 = 0 volt		SRESET Operation 0 = Normal 1 = Toggle on resume			

4.10.3.3 Clock Stretching

The register bits below are used to enable power-saving features for various system clocks. The CPU clock stretch functions should be enabled only for a totally static CPU, and then only when the CPU clock is set to /1 or /4. The ATCLK stretch function can be enabled for both 1X and 2X CPUs.

Index	Name	7	6	5	4	3	2	1	0
5Eh	Clock Stretch Register	Stretch memory code cycle 0 = Disable 1 = Enable	Stretch write cycle 0 = Disable 1 = Enable	Stretch read cycle 0 = Disable 1 = Enable	Stretch I/O cycle 0 = Disable 1 = Enable	Stretch memory data cycle 0 = Disable 1 = Enable	ATCLK when not in cycle 0 = Runs 1 = Stopped	AT clock stretch 0 = Async. 1 = Synchronous	

4.10.3.4 Stopping IPC Clock When Not In Use

Setting bit 50h[5] = 0 stops the clock going to the internal 82C206 IPC module. Primarily this setting affects the 8254-type Clock/Timer/Counter circuit. If the timer will not be used to maintain the DOS system clock, substantial power savings can be achieved by disabling this clock, and turning off the OSC14 clock generator if possible. Although KBCLK and KBCLK2 are derived from the 14MHz clock directly, they are not affected by this bit setting.

PMU Control Register - Index 50h

Index	Name	7	6	5	4	3	2	1	0
50h	PMU Control Register 5				14.3MHz to IPC 0 = Enable 1 = Disable				

4.10.3.5 Stopping KBCLK and KBCLK2

The 82C465MVA part allows the KBCLK and KBCLK2 outputs to be driven low. This feature is useful during suspend mode to stop the keyboard and IRQ/DRQ scanning clocks when those devices are powered down.

KBCLK/KBCLK2 Stop Control

Index	Name	7	6	5	4	3	2	1	0
79h	PMU Control Register 11								KBCLK/ KBCLK2 Control 0=Normal operation 1=Stopped in low state (MVA)

4.11 Power Control Latch and PIO Pins

There are 12 peripheral power pins (PPWR0-11) that are used to control power to individual peripherals through external 74373 latches. Each latch pin is controlled with its individual control bits in the configuration registers at indexes 54h, 55h, and ABh.

Four general purpose I/O pins are also provided for controlling or monitoring external devices without the need for additional TTL. These pins also offer certain preprogrammed functions that are commonly useful in system designs.

4.11.1 Power Control Latch

The value latched by PPWRL from the MA bus extends from PPWR0 through PPWR11, providing four useful power control signals for up to a dozen devices if all 12 bits are latched. MA11 is an optional signal, so PPWR11 is available only if MA11 is enabled as explained in the Memory Controller section of this document.

4.11.1.1 Hardware Considerations

The power control scheme uses the MA0-11 signals that normally address DRAM to additionally provide the inputs to a 74373-type latch. If all 12 signals will be used, two '373 devices (or a different combination, such as one '373 and one half of an '11873) are needed. The PPWRL signal from the 82C465MV is an active high signal and latches the MA0-11 signals on the latch output on its falling edge.

The pins PPWR0 and PPWR1 have a recovery delay time associated with them when doing the suspend/resume function. These two pins can be used as a delay control for some component that needs some time to become stable once power is restored. For example, after turning off the power to the clock oscillator during suspend mode, the resume function will restore power to the clock oscillator and wait until the clock has had time to stabilize before continuing the resume process.

PPWR10 provides the RSMRST# function. On hardware reset and on resuming from suspend mode, PPWR10 simply pulses low to generate a reset for any peripherals that were powered down during suspend. While also available directly from pin 185 as a strap-selected option, RSMRST# is always provided on PPWR10 regardless of the strapping option so that when pin 185 is reassigned from EPMI2 to DACKMUX1 (the Alternative DACKMUX Interface option), RSMRST# will still be available. Refer to the "Reset Logic" section for RSMRST# timing information.

During reset, the PPWRx latch signal (PPWRL) is pulsed to set the PPWRx signals to a known state. After reset PPWR0-3 and PPWR8-11 are set to '0'; PPWR4-7 are set to '1'. The PPWRx signals will remain in this state until they are updated by writing to configuration register indexes 54h, 55h, and ABh.

NOTE The MA0-11 pins and PPWRL are on the CPU power plane. In a mixed-voltage system, 3.3V-inputs to a 5V-powered latch will result in excessive current drain for any input that remains high during idle periods (around 1mA per input in suspend mode). The designer should make provisions to minimize the effect of this condition.

4.11.1.2 Programming

Register indexes 54h, 55h, and ABh set the power control latch outputs. The upper bits [7:4] of each register select whether the corresponding bits [3:0] should be used to change the latch; if the enable bit is 0, the current latch setting will not be changed when the register is written.

Index	Name	7	6	5	4	3	2	1	0
54h	Power Ctrl. Latch Register 1	Enable [3:0] to write latch lines PPWR3-0 0 = Disable 1 = Enable				Read/write data bits for PPWR3-0 - default 0000 0 = Latch output low 1 = Latch output high			
55h	Power Ctrl. Latch Register 2	Enable [3:0] to write latch lines PPWR7-4 0 = Disable 1 = Enable				Read/write data bits for PPWR7-4 - default 1111 0 = Latch output low 1 = Latch output high			
ABh	PMU Control Register 3	Enable [3:0] to write latch lines PPWR11-8 0 = Disable 1 = Enable				Read/write data bits for PPWR11-8 - default 0000 0 = Latch output low 1 = Latch output high			
68h	Clock Source Register 3					Resume recovery time 00 = 8ms 01 = 32ms 10 = 128ms 11 = 30us Ignored if BEh[0] = 1		PPWR1-0 auto toggle 0 = Disable 1 = Enable	
BEh	Idle Reload Event Enable Register 2								Override 68h[3:2] 0 = No 1 = Recover time 1s

4.11.1.2.1 Resume Recovery Time

The bits at 68h[3:2] determine the recovery time from PPWR1-0 active after a resume until the end of reset. The RSMRST# signal and/or RST4# is active during this recovery time. The clock is guaranteed to be active for at least the last 1/8 of the recovery time. These bits are not affected by bits 68h[1:0].

These bits can be overridden by setting bit BEh[0] = 1, in which case the resume recovery time will always be 1 second.

4.11.1.2.2 PPWR1-0 Suspend Auto Toggle Feature

Bits 68h[0] and 68h[1] enable PPWR0 and 1, respectively, to automatically toggle when entering and exiting suspend mode. Using PPWR0 as an example: When bit 0 = '1' and the chipset has gone into suspend mode, PPWR0 gets set to the inverse of bit 54h[0]; mask bit 54h[4] is ignored. When exiting suspend mode, PPWR0 is set to the bit 54h[0] setting, followed by the recovery time delay set in bits 68h[3:2] before continuing the resume operation.

4.11.2 Programmable I/O Pins

The programmable I/O (PIO) pins provide general purpose I/O pins for controlling and/or monitoring system operations. Several of these pins have specialized options that are linked into the 82C465MV logic.

- PIO3 can be redefined as the STPGNT# output for those CPUs that provide a specific acknowledge signal, not just a special stop grant cycle, to the chipset. Certain Cyrix and TI CPUs, and the IBM "Blue Lightning" CPU, provide this signal.
- PIO2 can be defined as an output to indicate when the CPU is in its full-speed mode, or as an input for an external AT bus clocking source (ATCLKIN).
- PIO1 can be redefined as the Zero Wait State input NOWS# from the AT bus. When enabled and active, this signal can reduce the time required to complete an AT-bus cycle to improve system speed.
- PIO0 can be redefined as the LREQ# input from the VL bus so that VL bus masters can request ownership of the bus. When PIO0 is redefined in this way, the DACK2# pin also gets redefined as the LGNT# output to the VL bus.

When they are not assigned special functions, the PIO pins are set for input or output and their data read or written through the registers at indexes 56h and 57h.

NOTE If a PIO pin will not be used, it should either be programmed as an output pin or should be pulled up externally if left as an input. Otherwise, during suspend mode the pin will float and cause high power consumption.

PIO Pin Registers

Index	Name	7	6	5	4	3	2	1	0
56h	PIO Pin Control Register	Write mask of 56h[3:0] 0 = Disable writes on corresponding bit [3:0] 1 = Enable bit [3:0] writes to PIO pins				Read/write data for PIO3-0 Read. Returns value present at pin. Write: Sets pin value if direction bit is set to 1.			
57h	PMU Control Register 6					PIO3 Direction 0 = Input 1 = Output	PIO2 Direction 0 = Input 1 = Output	PIO1 Direction 0 = Input 1 = Output	PIO0 Direction 0 = Input 1 = Output
66h	PMU Control Register 8					Pin 171 0 = PIO3 1 = STPGNT#	Pin 172 0 = PIO2 1 = CPUSPD	Pin 173 0 = PIO1 1 = NOWS#	
A0h	Feature Control Register 1			Enable Local Bus Master Support 0 = PIO0 and DACK2# 1 = LREQ# and LGNT#					

4.11.2.1 PIO3/STPGNT# Pin Select

When pin 171 is STPGNT#, also set bit 57h[3] to input mode. STPGNT# is for CPUs that use a dedicated stop grant signal pin to acknowledge a stop request.

4.11.2.2 PIO2/CPUSPD Pin Select

When pin 172 is CPUSPD, also set bit 57h[2] to output mode. CPUSPD indicates whether the CPU is at full speed or is slowed down. Note that output settings on PIO2 may conflict with ATCLKIN feature set by bit A0h[4].

4.11.2.3 PIO1/NOWS# Pin Select

When pin 173 is NOWS#, also set bit 57h[1] to input mode. NOWS# is the AT bus signal that requests a shorter cycle if possible.

4.11.3 Programmable Chip Select Feature

The 82C465MV provides programmable chip select features that require no chip signals to be sacrificed. A total of four programmable chip selects are available, and can decode either memory cycles or I/O cycles. For I/O chip select decoding, granularity can be specified to-the-byte, decoding a total of 10 bits. Bit A7h[7] determines whether the A[15:10] bits must be 0 or will be ignored. For ROM chip select decoding, granularity is to 16KB blocks anywhere in the AT address space (16MB).

Note that the memory chip select feature should be used cautiously for ROMs residing below 1MB. Since the ROM to be selected is on the SD bus, the XD bus buffer (if used) may be directed toward the 82C465MV chip for memory reads and could conflict with SD bus ROMs. Therefore, always set the ROMCS# generation registers (described in the "System ROM and Shadow RAM" section under the "DRAM Controller" heading in this document) to prevent XD bus ROMCS# conflicts with SD bus ROMs.

In hardware design, the chip select signals are decoded as follows. Using 7432 gates, first qualify ATCYC# with AEN to obtain a signal called CSG# here. Then:

- CSG0# = CSG# ORed with MA8
- CSG1# = CSG# ORed with MA9
- CSG2# = CSG# ORed with MA6
- CSG3# = CSG# ORed with MA7.

The required registers are shown below.

Programmable Chip Select 0 Registers

Index	Name	7	6	5	4	3	2	1	0
4Ah	Chip Select 0 Base Address Register	CSG0# base address A[8:1] (I/O) A[22:15] (memory)							
4Bh	Chip Select 0 Control Register	CSG0# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG0# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit 4Ah[3:0] is not compared. This is used to determine address block size.			
BFh	Chip Select 0 Granular Register				CSG0# base address A0 (I/O) A14 (mem)				CSG0# mask bit A0 (I/O) A14 (mem)
B3h	Chip Select 0 Cycle Type Register				CSG0# ROM width 0 = 8-bit 1 = 16-bit				CSG0# cycle type 0 = I/O 1 = ROMCS

Programmable Chip Select 1 Registers

Index	Name	7	6	5	4	3	2	1	0
4Ch	Chip Select 1 Base Address Register	CSG1# base address A[8:1] (I/O) A[22:15] (memory)							
4Dh	Chip Select 1 Control Register	CSG1# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG1# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit 4Ch[3:0] is not compared. This is used to determine address block size			
BFh	Chip Select 1 Granular Register			CSG1# base address A0 (I/O) A14 (mem)				CSG1# mask bit A0 (I/O) A14 (mem)	
B3h	Chip Select 1 Cycle Type Register			CSG1# ROM width 0 = 8-bit 1 = 16-bit				CSG1# cycle type 0 = I/O 1 = ROMCS	

Programmable Chip Select 2 Registers

Index	Name	7	6	5	4	3	2	1	0
BAh	Chip Select 2 Base Address Register	CSG2# base address A[8:1] (I/O) A[22:15] (memory)							
BBh	Chip Select 2 Control Register	CSG2# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG2# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit BAh[3:0] is not compared. This is used to determine address block size.			
BFh	Chip Select 2 Granular Register		CSG2# base address A0 (I/O) A14 (mem)				CSG2# mask bit A0 (I/O) A14 (mem)		
B3h	Chip Select 2 Cycle Type Register		CSG2# ROM width 0 = 8-bit 1 = 16-bit				CSG2# cycle type 0 = I/O 1 = ROMCS		

Programmable Chip Select 3 Registers

Index	Name	7	6	5	4	3	2	1	0
BCh	Chip Select 3 Base Address Register	CSG3# base address A[8:1] (I/O) A[22:15] (memory)							
BDh	Chip Select 3 Control Register	CSG3# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG3# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit BCh[3:0] is not compared. This is used to determine address block size.			
BFh	Chip Select 3 Granular Register	CSG3# base address A0 (I/O) A14 (mem)				CSG3# mask bit A0 (I/O) A14 (mem)			
B3h	Chip Select 3 Cycle Type Register	CSG3# ROM width 0 = 8-bit 1 = 16-bit				CSG3# cycle type 0 = I/O 1 = ROMCS			

5.0 Register Summary

All configuration registers of the 82C465MV are accessed using the index/data method: the configuration register index of interest is first written to port 022h; subsequently, the register data becomes available for reading or writing at port 024h. Port 022h must be rewritten each time, even if the index does not change. Port 022h is also readable to determine the last index value written.

Note that not all register bits are both readable and writable. Moreover, some register bits that can be written return different information when read. Therefore, it is good programming practice to maintain an up-to-date copy of all register settings in system RAM.

Register definitions that are specific to the 82C465MVA or 82C465MVB versions of the product are indicated for each bit with an MVA or and MVB, respectively, in parentheses. If a bit is designated as such, it is reserved for previous versions of the product. Definitions marked as MVA are also available in the MVB version. Registers without the MVA or MVB marking apply to all three versions.

The default hex value for each register at hardware reset time is indicated in parentheses under the Index number.

Index	Name	7	6	5	4	3	2	1	0
30h (40)	Control Register 1	82C46x Product Indicator - read-only. 00 = 82C463/463MV 01 = 82C465MV 10 = 82C465MVA 11 = 82C465MVB		Pin 186 function 0 = MSTR# 1 = RI	Turbo VGA (NOWS#) 0 = Disable 1 = Enable	SMI address relocation 0 = Disable 1 = Enable	AT wait states 0 = None 1 = One	Fast Reset 0 = Wait for HLT 1 = Immed	Reserved. Write as 0.
31h (40)	Control Register 2	Master byte swap 0 = Disable 1 = Enable	Reserved. Read-only Always reads 1	Parity check 0 = Enable 1 = Disable	Dynamic SMI relocation 0 = Normal 1 = Remap	ROMCS for EC000 See Table 4-6	ROMCS for E8000 See Table 4-6	ROMCS for E4000 See Table 4-6	ROMCS for E0000 See Table 4-6
32h (E4)	Shadow RAM Control Register 1	F0000 access 0=DRAM 1=ROM	Enable D000 writes 0 = Disable 1 = Enable	Enable E000 writes 0 = Disable 1 = Enable	D000 block shadow control 0=Writable 1=Protected	E000 block shadow control 0=Writable 1=Protected	Refresh on AT Bus 0=Disable 1=Enable (MVB)	Reserved. Write as 0.	ALEs in bus conversion 0=Multiple 1=Single
33h (00)	Shadow RAM Control Register 2	EC000 read select ROM/RAM 0 = ROM 1 = Sh RAM	E8000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	E4000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	E0000 read select ROM/RAM 0 = ROM 1 = Sh RAM	DC000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	D8000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	D4000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	D0000 read select ROM/RAM 0 = ROM 1 = Sh RAM
34h (05)	DRAM Control Register 1	DRAM banks 0 & 1 configuration (old scheme) Refer to 82C463MV Data Book for information				Reserved	DRAM banks 2 & 3 configuration (old scheme)		
35h (FF)	DRAM Control Register 2	Standard DRAM read wait states 00=3-2-2-2 01=4-3-3-3, 1ws pg miss 10=4-3-3-3, 0 ws pg miss 11=5-4-4-4		DRAM write wait states 00 = no wait states 01 = 1 wait state 10 = 1 wait state 11 = no wait states, RAS# 1/2 clock early (MVB)		MP2 Strapping Read-only 0=2X CPU 1=1X CPU	F000 64KB block cacheable 0=yes 1=no	Global caching control 0=Enable 1=Disable	C000 32KB block cacheable 0=yes 1=no
36h (10)	Sh RAM Control Register 3	F000 write select dest. 0 = DRAM 1 = ROM don't care for F000 if 32h[7] = 0	C-D-E000 select dest 0 = AT/ROM 1 = DRAM See Table 4-6	C000 write protect 0 = Writable 1 = Protected	Enable C000 writes 0 = Disable 1 = Enable	CC000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	C8000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	C4000 read select ROM/RAM 0 = ROM 1 = Sh. RAM	C0000 read select ROM/RAM 0 = ROM 1 = Sh. RAM
37h (0F)	D/E000 Control Register	ROMCS for DC000 See Table 4-6	ROMCS for D8000 See Table 4-6	ROMCS for D4000 See Table 4-6	ROMCS for D0000 See Table 4-6	EC00 16KB block cacheable 0 = yes 1 = no	E800 16KB block cacheable 0 = yes 1 = no	E400 16KB block cacheable 0 = yes 1 = no	E000 16KB block cacheable 0 = yes 1 = no

Index	Name	7	6	5	4	3	2	1	0
38h (80)	Block Control Register 1	Non cacheable block 1 (ncb1) size See Table 4-7			ROMCS for CC000 See Table 4-6	ROMCS for C8000 See Table 4-6	ROMCS for C4000 See Table 4-6	ROMCS for C0000 See Table 4-6	ncb1 A24
39h (00)	Block Control Register 2	Non cacheable block 1 start address A[23:16]							
3Ah (80)	Block Control Register 3	Non cacheable block 2 (ncb2) size See Table 4-7			Reserved. Write as 0.				ncb2 A24
3Bh (00)	Block Control Register 4	Non cacheable block 2 start address A[23:16]							
3Ch (00)	Timing Control Register	Reserved. Write as read.	Reserved Write as read.	Reserved. Write as read.	Reserved. Write as read.	Reserved. Write as read.	L2 Cache WE# Delay (MVB) 000=No delay 001=1 gate delay ... 110=6 gate delays 111=7 gate delays		
3Dh	Reserved								
3Eh (00)	DRAM Type Select Register	EDO DRAM read wait states 00=3-1-1-1 01=3-2-2-2 10=4-2-2-2 11=reserved (MVB)		Reserved. Write as read.	Bank 4 DRAM 0=Standard 1=EDO (MVB)	Bank 3 DRAM 0=Standard 1=EDO (MVB)	Bank 2 DRAM 0=Standard 1=EDO (MVB)	Bank 1 DRAM 0=Standard 1=EDO (MVB)	Bank 0 DRAM 0=Standard 1=EDO (MVB)
3Fh (00)	Misc Control Register	CPU Burst Mode 0=Intel 1=Cyrix linear (MVB)	Suspend Refresh Pulse Width 0=Normal 1=Reduced (MVB)	Four IDE Drive Support 0=Disable 1=Enable (MVB)	CPU Burst Write Support 0=Disable 1=Enable (MVB)	Minimum Wait States for non L2 cache systems 0=1ws 1=0ws (MVB)	Invalidate L1 Cache Line on Writes to WP DRAM 0=Disable 1=Enable (MVB)	Reserved. Write as read.	Reserved. Write as read.
40h (00)	PMU Control Register 1	Last jump to reset vector 0 = ADS# 1 = SMIADS#	Global timer divide 0 = div. by 1 1 = div. by 4	LLOWBAT polarity 0 = Active hi 1 = Active low	LOWBAT polarity 0 = Active hi 1 = Active low	SQWIN frequency 0 = 32KHz 1 = 128KHz	EPMI2 polarity 0 = Active hi 1 = Active low	EPMI1 polarity 0 = Active hi 1 = Active low	RSMRST# select 0 = Disable 1 = Enable See Figure 4-1
41h (00)	PMU Control Register 2	DOZE_0 timeout select 0 0 0 = 2 ms 0 0 1 = 4 ms 0 1 0 = 8 ms 0 1 1 = 32 ms 1 0 0 = 128 ms 1 0 1 = 512ms 1 1 0 = 2 s 1 1 1 = 8 s			Doze mode system clock speed 0 0 0 = OSCCLK / 1 0 1 = OSCCLK / 2 1 0 = OSCCLK / 4 1 1 = OSCCLK / 8 0 0 = OSCCLK / 16 0 1 = OSCCLK / 3 1 0 = reserved 1 1 = reserved			LCD, DSK, KBD, HDU _ACCESS events reset doze mode 0 = Disable 1 = Enable	Doze control select 0 = Hard-ware 1 = Software
42h (00)	Clock Source Register 1	Clock source for GNR_TIMER		Clock source for KBD_TIMER		Clock source for DSK_TIMER		Clock source for LCD_TIMER	
43h (00)	PMU Control Register 4	LCD_ ACCESS includes I/O range 3B0h-3DFh 0 = Yes 1 = No	LCD_ ACCESS includes memory A0000-BFFFFh 0 = Yes 1 = No	LOWBAT pin sample rate - generates PMI each time sampled active. 00 = 32s 01 = 64s 10 = 128s 11 = Reserved	ATCLK generator source 0 = FBCLKIN 1 = ATCLKIN w/A0h[4] = 1	AT clock - Rate selections 000 = /8 100 = 7.2 MHz 001 = /6 101 = /2 010 = /4 110 = /1 (/2 if 43h[3] = 0) 011 = /3 111 = Stop			
44h (00)	LCD_TIMER Register	LCD_TIMER count							
45h (00)	DSK_ TIMER Register	DSK_TIMER count							

Index	Name	7	6	5	4	3	2	1	0
46h (00)	KBD_ TIMER Register	KBD_TIMER count							
47h (00)	GNR1_ TIMER Register	GNR1_TIMER count							
48h (00)	GNR1 Base Address Register	GNR1_ACCESS base address A[8:1] (I/O) A[22:15] (memory)							
49h (00)	GNR1 Control Register	GNR1 base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	GNR1 Mask bits for address A[5:1] (I/O) or A[19:15] memory - a '1' in a particular bit means that the corresponding bit 48h[4:0] is not compared. This is used to determine address block size.				
4Ah (00)	Chip Select 0 Base Address Register	CSG0# base address A[8:1] (I/O) A[22:15] (memory)							
4Bh (00)	Chip Select 0 Control Register	CSG0# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG0# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit 4Ah[3:0] is not compared. This is used to determine address block size.			
4Ch (00)	Chip Select 1 Base Address Register	CSG1# base address A[8:1] (I/O) A[22:15] (memory)							
4Dh (00)	Chip Select 1 Control Register	CSG1# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG1# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit 4Ch[3:0] is not compared. This is used to determine address block size.			
4Eh (00)	Idle Reload Event Enable Register 1	CSG1_ ACCESS 0 = Disable 1 = Enable	CSG0_ ACCESS 0 = Disable 1 = Enable	LPT_ ACCESS 0 = Disable 1 = Enable	Reserved. Write as 0	GNR1_ ACCESS 0 = Disable 1 = Enable	KBD_ ACCESS 0 = Disable 1 = Enable	DSK_ ACCESS 0 = Disable 1 = Enable	LCD_ ACCESS 0 = Disable 1 = Enable
4Fh (00)	IDLE_ TIMER Register	IDLE_TIMER count							
50h (00)	PMU Control Register 5	Software start SMI 0 = Clear SMI 1 = Start SMI	Reserved. Write as 0.	IRQ8 polarity 0 = Active low 1 = Active hi	14.3MHz to IPC 0 = Enable 1 = Disable	Write = 1 to Start Doze Read: doze status 0 = counting 1 = timed out	Ready to resume Read only. 0 = Not in resume 1 = Ready to resume	PMU mode Read only. 0 = Nothing pending 1 = Suspend active (clear PMI#6)	Start suspend Write-only. 1 = Enter suspend mode
51h (00)	Beeper Control Register	General purpose storage bits						Beeper Control 00 = No action 01 = 1KHz 10 = Off 11 = 2KHz	
52h (00)	Scratchpad Register 1	General purpose storage byte For CISA Configuration Cycles: Data phase information, low byte (MVB)							
53h (00)	Scratchpad Register 2	General purpose storage byte For CISA Configuration Cycles: Data phase information, high byte (MVB)							
54h (X0)	Power Ctrl. Latch Register 1	Enable [3:0] to write latch lines PPWR3-0 0 = Disable 1 = Enable				Read/write data bits for PPWR3-0 - default 0000 0 = Latch output low 1 = Latch output high			
55h (XF)	Power Ctrl Latch Register 2	Enable [3:0] to write latch lines PPWR7-4 0 = Disable 1 = Enable				Read/write data bits for PPWR7-4 - default 1111 0 = Latch output low 1 = Latch output high			

Index	Name	7	6	5	4	3	2	1	0
56h (X0)	PIO Pin Control Register	Write mask of 56h[3:0] 0 = Disable writes on corresponding bit [3:0] 1 = Enable bit [3:0] writes to PIO pins				Read/write data for PIO3-0 Read: Returns value present at pin. Write: Sets pin value if direction bit is set to 1.			
57h (00)	PMU Control Register 6	Refresh enable 0 = Disable 1 = Enable	INTRGRP generates PMI #6 0 = Disable 1 = Enable	DSK_ACC includes FDD 0 = Yes 1 = No	DSK_ACC includes HDD 0 = Yes 1 = No	PIO3 Direction 0 = Input 1 = Output	PIO2 Direction 0 = Input 1 = Output	PIO1 Direction 0 = Input 1 = Output	PIO0 Direction 0 = Input 1 = Output
58h (00)	PMU Event Register 1	LOWBAT PMI #3 SMI 00 = Disable 11 = Enable		EPMI2 PMI #2 SMI 00 = Disable 11 = Enable		EPMI1 PMI #1 SMI 00 = Disable 11 = Enable		LLOWBAT PMI #0 SMI 00 = Disable 11 = Enable	
59h (00)	PMU Event Register 2	Allow software SMI 0 = Disable 1 = Enable	Reload timers on resume 0 = No 1 = Yes	Resume INTRGRP PMI #6, Suspend PMI #7 SMI 00 = Disable 11 = Enable		R_TIMER PMI #5 SMI 00 = Disable 11 = Enable		IDLE_TIMER PMI #4 SMI 00 = Disable 11 = Enable	
5Ah (00)	PMU Event Register 3	GNR1_TIMER PMI #11 SMI 00 = Disable 11 = Enable		KBD_TIMER PMI #10 SMI 00 = Disable 11 = Enable		DSK_TIMER PMI #9 SMI 00 = Disable 11 = Enable		LCD_TIMER PMI #8 SMI 00 = Disable 11 = Enable	
5Bh (00)	PMU Event Register 4	SMI to IRQ15 0 = Disable 1 = Enable	Global SMI control 0 = Allow 1 = Mask	Reserved. Write as 0	SMI type 0 = Intel 1 = Other	GNR1 Next Access PMI #15 0 = Disable 1 = Enable	KBD Next Access PMI #14 0 = Disable 1 = Enable	DSK Next Access PMI #13 0 = Disable 1 = Enable	LCD Next Access PMI #12 0 = Disable 1 = Enable
5Ch (00)	PMI Source Register 1 Write 1 to clear	PMI7 - Suspend 0 = Inactive 1 = Active	PMI6 - Resume or INTRGRP 0 = Inactive 1 = Active	PMI5 - R_TIMER timeout 0 = Inactive 1 = Active	PMI4 - IDLE_TMR timeout 0 = Inactive 1 = Active	PMI3 - LOWBAT 0 = Inactive 1 = Active	PMI2 - EPMI2 0 = Inactive 1 = Active	PMI1 - EPMI1 0 = Inactive 1 = Active	PMI0 - LLOWBAT 0 = Inactive 1 = Active
5Dh (00)	PMI Source Register 2 Write 1 to clear	PMI15 - GNR1_ ACCESS 0 = Inactive 1 = Active	PMI14 - KBD_ ACCESS 0 = Inactive 1 = Active	PMI13 - DSK_ ACCESS 0 = Inactive 1 = Active	PMI12 - LCD_ ACCESS 0 = Inactive 1 = Active	PMI11 - GNR1_ TIMER 0 = Inactive 1 = Active	PMI10 - KBD_ TIMER 0 = Inactive 1 = Active	PMI9 - DSK_ TIMER 0 = Inactive 1 = Active	PMI8 - LCD_ TIMER 0 = Inactive 1 = Active
5Eh (00)	Clock Stretch Register	Stretch memory code cycle 0 = Disable 1 = Enable	Stretch write cycle 0 = Disable 1 = Enable	Stretch read cycle 0 = Disable 1 = Enable	Stretch I/O cycle 0 = Disable 1 = Enable	Stretch memory data cycle 0 = Disable 1 = Enable	ATCLK when not in cycle 0 = Runs 1 = Stopped	AT clock stretch 0 = Async. 1 = Synchronous	Reserved. Write as 0.
5Fh (00)	PMU Control Register 7	LCD_ ACCESS includes AT-bus video access 0 = Yes 1 = No	LCD_ ACCESS includes VL-bus video access 0 = No 1 = Yes	RSMGRP IRQs can resume system 0 = No 1 = Yes	Transitions on RI can resume system 0 = No 1 = Yes	Number of RI transitions to cause resume			
60h (00)	R_TIMER Base Count Register Read-only	R_TIMER initial count							
61h (00)	Debounce Register	LOWBAT, LLOWBAT debounce rate select 00 = No debounce 01 = 250us 10 = 8ms 11 = 500ms		SUSP/RSM debounce rate select 00 = Reserved 01 = Latch high-to-low edge 10 = 4ms(low-to-high) 11 = 8ms(low-to-high)		Reserved. Write as 0.	STPCLK# signal 0 = Disable 1 = Enable	Reserved. Write as 0.	
62h (00)	IRQ Doze Register 1 Write-only	IRQ13 doze reset 0 = Disable 1 = Enable	IRQ8 doze reset 0 = Disable 1 = Enable	IRQ7 doze reset 0 = Disable 1 = Enable	IRQ12 doze reset 0 = Disable 1 = Enable	IRQ5 doze reset 0 = Disable 1 = Enable	IRQ4 doze reset 0 = Disable 1 = Enable	IRQ3 doze reset 0 = Disable 1 = Enable	IRQ0 doze reset 0 = Disable 1 = Enable

Index	Name	7	6	5	4	3	2	1	0
63h (00)	Idle Timeout Select Register 1	EPMI1 Level-triggered 0 = Disable 1 = Enable	IRQ13 0 = Disable 1 = Enable	IRQ8 0 = Disable 1 = Enable	IRQ7 0 = Disable 1 = Enable	IRQ5 0 = Disable 1 = Enable	IRQ4 0 = Disable 1 = Enable	IRQ3 0 = Disable 1 = Enable	IRQ0 0 = Disable 1 = Enable
64h (00)	INTRGRP IRQ Select Register 1	IRQ14 0 = Disable 1 = Enable	IRQ8 0 = Disable 1 = Enable	IRQ7 0 = Disable 1 = Enable	IRQ6 0 = Disable 1 = Enable	IRQ5 0 = Disable 1 = Enable	IRQ4 0 = Disable 1 = Enable	IRQ3 0 = Disable 1 = Enable	IRQ1 0 = Disable 1 = Enable
65h (00)	DOZE Register	All interrupts to CPU reset doze mode 0 = Disable 1 = Enable	STPCLK control 0 = Pulse 1 = Latch	EPMI1 doze reset 0 = Disable 1 = Enable	SMI resets doze mode 0 = No 1 = Yes	IRQ1 doze reset 0 = Disable 1 = Enable	Reserved. Write as 0.		
66h (00)	PMU Control Register 8	Suspend refresh 0 = Slow 1 = None (for self-refresh DRAM)	Suspend KBCLK source 0 = 14MHz/2 1 = 32KHz/2	Doze type 0 = Slow CPU clock 1 = Stop CPU clock	Assert HOLD during suspend 0 = Yes 1 = No	Pin 171 0 = PIO3 1 = STPGNT#	Pin 172 0 = PIO2 1 = CPUSPD	Pin 173 0 = PIO1 1 = NOWS#	CPU clock change protocol required 0 = No 1 = Yes
67h (00)	PMU Control Register 9	Reserved. Write as 0.	Refresh rate Active or suspend mode '00' = 15us (30us in suspend if A1h[6] = 0) '01' = 30us '10' = 61us '11' = 122us		Reserved. Write as 1.	Reserved. Write as 0.	Write: Select CPU frequency Read: Return current operating freq. 0 0 0 CPUCLK / 1 0 0 1 CPUCLK / 2 0 1 0 CPUCLK / 4 0 1 1 CPUCLK / 8 1 0 0 CPUCLK / 16 1 0 1 CPUCLK / 3 1 1 x Reserved		
68h (00)	Clock Source Register 3	R_TIMER clock source		IDLE_TIMER clock source		Resume recovery time 00 = 8ms 01 = 32ms 10 = 128ms 11 = 30us Ignored if BEh[0] = 1		PPWR1-0 auto toggle 0 = Disable 1 = Enable	
69h (00)	TIMER Register	R_TIMER count							
6Ah (00)	RSMGRP IRQ Register 1	EPMI2 resume 0 = Disable 1 = Enable	EPMI1 resume 0 = Disable 1 = Enable	IRQ8 resume 0 = Disable 1 = Enable	IRQ7 resume 0 = Disable 1 = Enable	IRQ5 resume 0 = Disable 1 = Enable	IRQ4 resume 0 = Disable 1 = Enable	IRQ3 resume 0 = Disable 1 = Enable	IRQ1 resume 0 = Disable 1 = Enable
6Bh (00)	Resume Source Register	Reserved Write as read	Pin 135 0=FLUSH# 1= SMIRDY#	Reserved. Write as read.	Reserved. Write as read.	CISA SEL#/ ATB# low caused resume read-only 0=No 1=Yes (MVB)	SUSP/RSM caused resume read-only 0=No 1=Yes	RSMGRP caused resume read-only 0=No 1=Yes	RI caused resume read-only 0=No 1=Yes
6Ch (00)	Scratchpad Register 3	General purpose storage byte For CISA Configuration Cycles. Address phase 1 information, low byte (MVB)							
6Dh (00)	Scratchpad Register 4	General purpose storage byte For CISA Configuration Cycles. Address phase 1 information, high byte (MVB)							
6Eh (00)	Scratchpad Register 5	General purpose storage byte For CISA Configuration Cycles: Address phase 2 information, low byte (MVB)							
6Fh (00)	Scratchpad Register 6	General purpose storage byte For CISA Configuration Cycles: Address phase 2 information, high byte (MVB)							
70h (00)	GNR1 Control Register 2	GNR1_ACCESS base address A[13 6] for memory watchdog A[15 10] for I/O (right-aligned) (MVA)							

Index	Name	7	6	5	4	3	2	1	0
71h (00)	GNR1 Control Register 3	GNR1_ACCESS mask bits Mask for A[13:6] for memory watchdog Mask for A[15:10] for I/O (right-aligned) (MVA)							
72h (00)	GNR1 Base Addr. Register 4	GNR1_ACCESS base address (MVA and MVB) A[5:2] for memory watchdog* Ignored for I/O * In MVB, if AEh[6]=0. For A[31]+x+A[25]+[A24] if AEh[6]=1				GNR1_ACCESS mask bits (MVA and MVB) Mask for A[5:2] for memory watchdog* Mask for A[9:6] for I/O * In MVB, if AEh[6]=0. For A[31]+x+A[25]+[A24] if AEh[6]=1			
73h (00)	GNR1 Control Register 2	GNR2_ACCESS base address A[13:6] for memory watchdog A[15:10] for I/O (right-aligned) (MVA)							
74h (00)	GNR1 Control Register 3	GNR2_ACCESS mask bits Mask for A[13:6] for memory watchdog Mask for A[15:10] for I/O (right-aligned) (MVA)							
75h (00)	GNR1 Base Addr. Register 4	GNR2_ACCESS base address (MVA and MVB) A[5:2] for memory watchdog* Ignored for I/O * In MVB, if AEh[7]=0 For A[31]+x+A[25]+[A24] if AEh[7]=1				GNR2_ACCESS mask bits (MVA and MVB) Mask for A[5:2] for memory watchdog* Mask for A[9:6] for I/O * In MVB, if AEh[7]=0. For A[31]+x+A[25]+[A24] if AEh[7]=1			
76h (00)	Doze Reload Select Register 1	LCD_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	KBD_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	DSK_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	HDU_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	COM1&2_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	LPT_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	GNR1_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)	GNR2_ ACCESS 0=DOZE_0 1=DOZE_1 Default is 1 (MVA)
77h (00)	Doze Reload Select Register 2	IRQ8 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ7 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ6 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ5 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ4 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ3 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ1 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ0 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)
78h (00)	Doze Reload Select Register 3	LDEV# 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ15 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ14 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ13 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ12 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ11 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ10 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)	IRQ9 0=DOZE_0 1=DOZE_1 Default is 0 (MVA)
79h (00)	PMU Control Register 11	DOZE_1 Timeout Select (MVA) 0 0 0 = No delay (default) 0 0 1 = 1 ms 0 1 0 = 4 ms 0 1 1 = 16 ms 1 0 0 = 64 ms 1 0 1 = 256 ms 1 1 0 = 1 s 1 1 1 = 4 s			SMI resets doze mode if clock is stopped inside SMM 0=No 1=Yes (MVA)	Pin 172 function 0=PIO2 or CPUSPD 1=Buffer enable pin SABUFEN# (MVA)	"Fast" Logic Functionality Level 00=Fully functional 01=Do not inhibit KBDCS# 10=Also: Disable reset from 060/064h 11=Also: Redefine pin 179 as KBCRSTIN (MVA)		KBCLK/ KBCLK2 Control 0=Normal operation 1=Stopped in low state (MVA)
7A-7Fh	Reserved								
80h	Shadow Register	Interrupt Controller 1 Shadow Register ICW1							
81h	Shadow Register	Interrupt Controller 1 Shadow Register ICW2							
82h	Shadow Register	Interrupt Controller 1 Shadow Register ICW3							
83h	Shadow Register	Interrupt Controller 1 Shadow Register ICW4							
84h	DMA In-Progress Register Read-only	Channel 7 DMA in progress 0=No 1=Possibly	Channel 6 DMA in progress 0=No 1=Possibly	Channel 5 DMA in progress 0=No 1=Possibly	DMAC2 Byte Pointer Flip-Flop Read Only. 0=Cleared 1=Set (MVA)	Channel 3 DMA in progress 0=No 1=Possibly	Channel 2 DMA in progress 0=No 1=Possibly	Channel 1 DMA in progress 0=No 1=Possibly	Channel 0 DMA in progress 0=No 1=Possibly
85h	Shadow Register	Interrupt Controller 1 Shadow Register OCW1							

Index	Name	7	6	5	4	3	2	1	0
86h	Shadow Register	Interrupt Controller 1 Shadow Register OCW2							
87h	Reserved								
88h	Shadow Register	Interrupt Controller 2 Shadow Register ICW1							
89h	Shadow Register	Interrupt Controller 2 Shadow Register ICW2							
8Ah	Shadow Register	Interrupt Controller 2 Shadow Register ICW3							
8Bh	Shadow Register	Interrupt Controller 2 Shadow Register ICW4							
8Ch	Reserved								
8Dh	Shadow Register	Interrupt Controller 2 Shadow Register OCW1							
8Eh	Shadow Register	Interrupt Controller 2 Shadow Register OCW2							
8Fh	Reserved								
90h	Channel 0 Low Byte	Timer Channel 0 count low byte, A[7:0]							
91h	Channel 0 High Byte	Timer Channel 0 count high byte, A[15:8]							
92h	Channel 1 Low Byte	Timer Channel 1 count low byte, A[7:0]							
93h	Channel 1 High Byte	Timer Channel 1 count high byte, A[15:8]							
94h	Channel 2 Low Byte	Timer Channel 2 count low byte, A[7:0]							
95h	Channel 2 High Byte	Timer Channel 2 count high byte, A[15:8]							
96h	Write Counter High/ Low Byte Latch	Unused	Unused	Channel 2 read LSB toggle bit	Channel 1 read LSB toggle bit	Channel 0 read LSB toggle bit	Channel 2 write LSB toggle bit	Channel 1 write LSB toggle bit	Channel 0 write LSB toggle bit
97h	Reserved								
98h	RTC Index Shadow Register read-only	NMI Enable Setting	CMOS RAM Index Last Written						
99-9Ah	Reserved								
9Bh (00)	3F2h+3F7h Shadow Register	Shadows 3F2h[7] "Mode Select" bit (MVA)	Shadows 3F7h[1] "Disk Type" bit 1 (MVA)	Shadows 3F2h[5] "Drive 2 Motor" bit (MVA)	Shadows 3F2h[4] "Drive 1 Motor" bit (MVA)	Shadows 3F2h[3] "DMA Enable" bit (MVA)	Shadows 3F2h[2] "Soft Reset" bit (MVA)	Shadows 3F7h[0] "Disk Type" bit 0 (MVA)	Shadows 3F2h[0] "Drive Select" bit (MVA)
9Ch (00)	372h+377h Shadow Register	Shadows 372h[7] "Mode Select" bit (MVA)	Shadows 377h[1] "Disk Type" bit 1 (MVA)	Shadows 372h[5] "Drive 2 Motor" bit (MVA)	Shadows 372h[4] "Drive 1 Motor" bit (MVA)	Shadows 372h[3] "DMA Enable" bit (MVA)	Shadows 372h[2] "Soft Reset" bit (MVA)	Shadows 377h[0] "Disk Type" bit 0 (MVA)	Shadows 372h[0] "Drive Select" bit (MVA)
9D-9Eh	Reserved								
9Fh (00)	Port 064h Shadow Register	Shadows I/O writes to port 064h bits [7:0], regardless of whether KBDCS# is inhibited. In this way, when an SMI occurs between a port 064h write and the subsequent write to port 060h, SMM code can access the keyboard controller as needed and then simply restore the port 064h value just before leaving SMM. (MVA)							
A0h (00)	Feature Control Register 1	Internal I/O Address Decoding 0 = 10-bit 1 = 16-bit	Automatic Internal Resistors 0 = Disable 1 = Enable	Enable Local Bus Master Support 0 = PIO0 and DACK2# 1 = LREQ# and LGNT#	Pin 172 0 = PIO2 (or CPUSPD) 1 = ATCLKIN	Enable Alternative DACKMUX Interface See Table 3-3 0 = Disable 1 = Enable	SRESET Enable in SMM 0 = Enable 1 = Disable	CPU Cache Operation Select 0 = Standard 1 = L1 write-back	DRAM Mapping Enable 0 = Disable 1 = Enable

Index	Name	7	6	5	4	3	2	1	0
A1h (00)	Feature Control Register 2	Port 060/4 Gate A20 Read-only. In MVB, Port 060/4 A20M# bit Read: return current value; Write: toggle A20M# setting	Suspend Refresh Pulse Gen. 0 = Wide 1 = ~100ns	Pin 88 - for EPMI3-4 0 = DRQ2 1 = EPMMUX	Heavy-duty Memory Bus Drive 0 = Disable 1 = Enable	Heavy-duty AT Bus Drive 0 = Disable 1 = Enable	Emergency Overtemp Sense 0 = Disable 1 = Enable	F000 shadow test 0 = Read or write 1 = Read and write	EPMI1-2 Status Latch 0 = Dynamic 1 = Latched
A2h (00)	IRQ Doze Register 2 Write only	Local bus I/O access doze reset 0=Disable 1=Enable (MVA)	Local bus memory access doze reset 0=Disable 1=Enable (MVA)	IRQ15 doze reset 0=Disable 1=Enable	IRQ14 doze reset 0=Disable 1=Enable	IRQ11 doze reset 0=Disable 1=Enable	IRQ10 doze reset 0=Disable 1=Enable	IRQ9 doze reset 0=Disable 1=Enable	IRQ6 doze reset 0=Disable 1=Enable
A3h (00)	Idle Timeout Select Register 2	IRQ15 0 = Disable 1 = Enable	IRQ14 0 = Disable 1 = Enable	IRQ12 0 = Disable 1 = Enable	IRQ11 0 = Disable 1 = Enable	IRQ10 0 = Disable 1 = Enable	IRQ9 0 = Disable 1 = Enable	IRQ6 0 = Disable 1 = Enable	IRQ1 0 = Disable 1 = Enable
A4h (00)	INTRGRP IRQ Select Register 2	Test bit Write as 0.	IRQ15 0 = Disable 1 = Enable	IRQ13 0 = Disable 1 = Enable	IRQ12 0 = Disable 1 = Enable	IRQ11 0 = Disable 1 = Enable	IRQ10 0 = Disable 1 = Enable	IRQ9 0 = Disable 1 = Enable	IRQ0 0 = Disable 1 = Enable
A5h (00)	Thermal Management Register 1	Thermal Management 0 = Disable 1 = Enable	Equilibnum Level (EQL6:0) This count corresponds to equilibrium operation. If the CPU Activity Counter exceeds EQL, EQL is simply subtracted from the upper activity count byte and sampling continues. If the count is below EQL, the count is cleared.						
A6h (00)	Thermal Management Register 2	Overtemp Limit (OTL7:0) This count corresponds to an over-temperature situation. If the CPU Activity Counter exceeds OTL, the 82C465MV engages Cool-down Clocking.							
A7h (00)	Thermal Management Register 3	CPU Efficiency (CPUE1:0) 00 = Low power 01 = Moderate 10 = High 11 = Very high		Cool-down Holdoff (CDHO1:0) 00 = 32 seconds 01 = 64 seconds 10 = 128 seconds 11 = 256 seconds		Cool-down Clock Rate (CDCR1:0) 00 = /2 01 = /3 10 = /4 11 = /8		Cool-down Timeout (CDTO1:0) 00 = 2x CDHO 01 = 3x 10 = 4x 11 = 5x	
A8h (00)	DRAM Bank Select Register 1	Bank 1 type 0 = Symmetrical 1 = Asymmetrical	Bank 1 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB			Bank 0 type 0 = Symmetrical 1 = Asymmetrical	Bank 0 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
A9h (00)	DRAM Bank Select Register 2	Bank 3 type 0 = Symmetrical 1 = Asymmetrical	Bank 3 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB			Bank 2 type 0 = Symmetrical 1 = Asymmetrical	Bank 2 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
AAh (00)	DRAM Bank Select Register 3	Reserved. Write as 0.				Bank 4 type 0 = Symmetrical 1 = Asymmetrical	Bank 4 Memory Size 000 = Not installed 001 = 1MB 010 = 2MB 011 = 4MB 100 = 8MB 101 = 16MB 110 = 32MB 111 = 64MB		
ABh (X0)	PMU Control Register 3	Enable [3:0] to write latch lines PPWR11-8 0 = Disable 1 = Enable				Read/write data bits for PPWR11-8 - default 0000 0 = Latch output low 1 = Latch output high			

Index	Name	7	6	5	4	3	2	1	0
ACh (00)	IDE Interface Config Register	Chipset Input Clock Frequency 00 = 50MHz 01 = 40MHz 10 = 33MHz 11 = 20/25MHz		IDE Command Pulse Duration 00 = 600ns 01 = 383ns 10 = 240ns 11 = 180ns		IDE Interface Enable 0 = Disable 1 = Enable	IDE Port Address Select 0 = 1F0-7h, 3F6-7h 1 = 170-7h, 376-7h	3F7h[6:0] Source 0 = local IDE 1 = AT bus	Reserved. Write as 0.
ADh (00)	Feature Control Register 3	Ignore un-finished LDEV# cycles 0=wait 1=ignore	Generate CPURST immediately 0=No 1=Yes (MVA)	CPU Power State in Suspend 0=Powered 1=0 volt	Pin 130 in 386 mode 0=GA20 1=A20M#	SRESET Operation 0=Normal 1=Toggle on resume	Pin 80 in 386 mode 0=NPINT 1=DACK2#	Coprocessor Recognition 0=Enable 1=Override	RDYI# Input 0=Synchro-nized to RDY# 1=Direct
AEh (03)	GNR_ ACCESS Feature Register	GNR2 memory decoding 0=A[5:2] 1=A[31:24] (MVB)	GNR1 memory decoding 0=A[5:2] 1=A[31:24] (MVB)	GNR2 cycle decode type 0=I/O 1=Memory	GNR1 cycle decode type 0=I/O 1=Memory	GNR2 base address A0 (I/O) A14 (mem)	GNR1 base address A0 (I/O) A14 (mem)	GNR2 mask bit A0 (I/O) A14 (mem)	GNR1 mask bit A0 (I/O) A14 (mem)
AFh (34)	SMBASE Register	SMM segment to be mapped to B000h 0 = 0:0 1 = 1000:0h ... 9 = 9000.0h (A-F illegal) defaults to 3h				SMM segment to be mapped to A000h 0 = 0:0 1 = 1000:0h ... 9 = 9000:0h (A-F illegal) defaults to 4h			
B0h (00)	Stop-Clock Delay Register	Stop Clock Delay 0 = Disable 1 = Enable	Stop Clock Delay Time Base 0 = 32KHz/4 (~122 us) 1 = FBCLK/4	Delay Count - This value multiplies the time base period selected in bit [6]. There is an additional 6 FBCLK delay for all selections, even "no delay." Sample approximate delays based on 32KHz/25MHz selections: 000000 = No delay 000001 = 122us/160ns 000010 = 244us/320ns 000011 = 366us/480ns ... 001000 = 976us/1.28us 001001 = 1 1ms/1.44us... 111111 = 7 7ms/10.0us					
B1h (00)	RSMGRP IRQ Register 2	EPMI4 resume 0 = Disable 1 = Enable	EPMI3 resume 0 = Disable 1 = Enable	IRQ15 resume 0 = Disable 1 = Enable	IRQ14 resume 0 = Disable 1 = Enable	IRQ12 resume 0 = Disable 1 = Enable	IRQ11 resume 0 = Disable 1 = Enable	IRQ10 resume 0 = Disable 1 = Enable	IRQ9 resume 0 = Disable 1 = Enable
B2h (00)	Clock Source Register 2	Clock source for HDU_TIMER		Clock source for COM2_TIMER		Clock source for COM1_TIMER		Clock source for GNR2_TIMER	
B3h (00)	Chip Select. Cycle Type Register	CSG3# ROM width 0 = 8-bit 1 = 16-bit	CSG2# ROM width 0 = 8-bit 1 = 16-bit	CSG1# ROM width 0 = 8-bit 1 = 16-bit	CSG0# ROM width 0 = 8-bit 1 = 16-bit	CSG3# cycle type 0 = I/O 1 = ROMCS	CSG2# cycle type 0 = I/O 1 = ROMCS	CSG1# cycle type 0 = I/O 1 = ROMCS	CSG0# cycle type 0 = I/O 1 = ROMCS
B4h (00)	HDU_TIMER Register	Time count byte for HDU_TIMER							
B5h (00)	COM1_TIMER Register	Time count byte for COM1_TIMER							
B6h (00)	COM2_TIMER Register	Time count byte for COM2_TIMER							
B7h (00)	GNR2_TIMER Register	Time count byte for GNR2_TIMER							
B8h (00)	GNR2 Base Address Register	GNR2_ACCESS base address A[8 1] (I/O) A[22:15] (memory)							
B9h (00)	GNR2 Control Register	GNR2 base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	GNR2 Mask bits for address A[5:1] (I/O) or A[19:15] memory - a '1' in a particular bit means that the corresponding bit B8h[4:0] is not compared. This is used to determine address block size				
BAh (00)	Chip Select 2 Base Address Register	CSG2# base address A[8:1] (I/O) A[22:15] (memory)							

Index	Name	7	6	5	4	3	2	1	0
BBh (00)	Chip Select 2 Control Register	CSG2# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG2# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit BAh[3:0] is not compared. This is used to determine address block size.			
BCh (00)	Chip Select 3 Base Address Register	CSG3# base address A[8:1] (I/O) A[22:15] (memory)							
BDh (00)	Chip Select 3 Control Register	CSG3# base address A9 (I/O) A23 (mem)	Write Decode 0 = Disable 1 = Enable	Read Decode 0 = Disable 1 = Enable	Chip select active 0 = w/cmd 1 = before ALE	CSG3# Mask bits for address A[4:1] (I/O) or A[18:15] memory - a '1' in a particular bit means that the corresponding bit BCh[3:0] is not compared. This is used to determine address block size.			
BEh (00)	Idle Reload Event Enable Register 2	CSG3_ ACCESS 0 = Disable 1 = Enable	CSG2_ ACCESS 0 = Disable 1 = Enable	COM2_ ACCESS 0 = Disable 1 = Enable	COM1_ ACCESS 0 = Disable 1 = Enable	GNR2_ ACCESS 0 = Disable 1 = Enable	HDU_ ACCESS 0 = Disable 1 = Enable	Reserved. Write as 0.	Override 68h[3:2] 0 = No 1 = Recover time 1s
BFh (0F)	Chip Select. Granular. Register	CSG3# base address A0 (I/O) A14 (mem)	CSG2# base address A0 (I/O) A14 (mem)	CSG1# base address A0 (I/O) A14 (mem)	CSG0# base address A0 (I/O) A14 (mem)	CSG3# mask bit A0 (I/O) A14 (mem)	CSG2# mask bit A0 (I/O) A14 (mem)	CSG1# mask bit A0 (I/O) A14 (mem)	CSG0# mask bit A0 (I/O) A14 (mem)
C0-CFh	Reserved								
D0h (C0)	L2 Cache Control Register 1	L2 cache CCS0-3# de-assert 0 = stop grant and suspend 1 = also btw accesses	L2 cache controls suspend state 0 = tri-state 1 = driven	L2 Cache Engage 0 = Disable 1 = Enable	Cache Size 0 0 = 64KB 0 1 = 128KB 1 0 = 256KB 1 1 = reserved		L2 Cache Write Wait State 0 = 1 ws 1 = nows	L2 Cache Read Burst Wait State Control 0 = X-1-1-1 1 = X-2-2-2	L2 Cache First Read Wait State Control 0 = 3-X-X-X 1 = 2-X-X-X
D1h (41)	L2 Cache Control Register 2	L1 Cache HITM# sensing after EADS# 0=2nd clock 1=3rd clock	ADS# sampling 0=Sample on ADS# low 1=Latch ADS#, sample on next cycle	L2 Tag RAM Size 0=8-bit 1=7-bit (MVA)	L2 Cache Arrangement 0=Two banks 1=One bank (MVA)	EC000-EFFFFh L2 cacheable 0=No 1=Yes	E8000-EBFFFh L2 cacheable 0=No 1=Yes	E4000-E7FFFh L2 cacheable 0=No 1=Yes	E0000-E3FFFh L2 cacheable 0=No 1=Yes
D2h (00)	L2 Cache Control Register 3	DC000-DFFFFh L2 cacheable 0 = No 1 = Yes	D8000-DBFFFh L2 cacheable 0 = No 1 = Yes	D4000-D7FFFh L2 cacheable 0 = No 1 = Yes	D0000-D3FFFh L2 cacheable 0 = No 1 = Yes	CC000-CFFFFh L2 cacheable 0 = No 1 = Yes	C8000-CBFFFh L2 cacheable 0 = No 1 = Yes	C4000-C7FFFh L2 cacheable 0 = No 1 = Yes	C0000-C3FFFh L2 cacheable 0 = No 1 = Yes
D3h (00)	Asym DRAM Select Register	Segment for SMM data reads 0=A000h 1=SMBASE (MVA)	Segment for SMM data writes 0=A000h 1=SMBASE (MVA)	Cache Flush on SMI entry 0=Enable 1=Disable (MVA)	Bank 4 asym. type 0=11x9 1=12x8	Bank 3 asym. type 0=11x9 1=12x8	Bank 2 asym. type 0=11x9 1=12x8	Bank 1 asym. type 0=11x9 1=12x8	Bank 0 asym. type 0=11x9 1=12x8
D4h (00)	Resistor Control Register 1	CA31, CA[25:2], BE[3:0]# Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved		CD[31:0] Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved		M/IO#, D/C#, W/R# Pull-down Resistor Control 00 = Automatic 01 = Always enabled 10 = Always disabled 11 = Reserved		Reserved. Write as 0.	Redefine Pin 189 0=BOFF# 1=LCLK (MVA)

Index	Name	7	6	5	4	3	2	1	0
D5h (00)	Resistor Control Register 2	DACKMUX Control During Suspend 00 = Pull resistor-equipped lines low, drive others low 01 = Tri-state all lines (463MV mode) 10 = Drive 100b on DACKMUX2-0 11 = Reserved		FERR# Pull-up Resistor Control 0 = Always enabled 1 = Always disabled	Reserved. Write as 0.				Generate BOFF# (AHOLD) on Next Access Trap 0 = Disable 1 = Enable
D6h (00)	PMU Control Register 10	DSK_ ACCESS 0=3F5h only 1=All FDC ports (3F2,4,5,7h and 372,4,5,7h) (MVA)	DMA Trap PMI#28 SMI 0=Disable 1=Enable (MVA)	DMAC1 Byte Pointer Flip-Flop Read Only. 0=Cleared 1=Set (MVA)	HITM# Source 0=D/C# 1=Pin 135 (MVA)	Local-bus DMA LDEV# sampling 0=Normal 1=Sample one clock sooner (MVA)	I/O Port Access Trapped - Read Only. 0=I/O Read 1=I/O Write (MVA)	ACCESS Trap bit A9 - Read Only. (MVA)	ACCESS Trap bit A8 - Read Only. (MVA)
D7h (00)	ACCESS Port Address Register	ACCESS Trap Address bits A[7:0] - These bits, along with A[9:8] in bits D6h[1:0], provide the 10-bit I/O address of the port access that caused the SMI trap. Bit D6h[2] indicates whether an I/O read or I/O write access was trapped (MVA)							
D8h (00)	PMU Event Register 5	HDU_TIMER PMI #19 HDU_ACCESS PMI #23 SMI 00 = Disable 11 = Enable		COM2_TIMER PMI #18 COM2_ACCESS PMI #22 SMI 00 = Disable 11 = Enable		COM1_TIMER PMI #17 COM1_ACCESS PMI #21 SMI 00 = Disable 11 = Enable		GNR2_TIMER PMI# 16 GNR2_ACCESS PMI #20 SMI 00 = Disable 11 = Enable	
D9h (00)	PMU Event Register 6	DOZE_TIMER PMI #27 SMI 00=Disable 01=Enable DOZE_0 (MVA) 10=Enable DOZE_1 (MVA) 11=Enable both		RI PMI #26 SMI 00=Disable 11=Enable		EPMI4 PMI #25 SMI 00=Disable 11=Enable		EPMI3 PMI #24 SMI 00=Disable 11=Enable	
DAh (00)	Power Mgt. Event Status Register (read-only)	Reserved Mask when reading	Reserved Mask when reading	LOWBAT state 0 = inactive 1 = active	LLOWBAT state 0 = inactive 1 = active	EPMI4 state 0 = inactive 1 = active	EPMI3 state 0 = inactive 1 = active	EPMI2 state 0 = inactive 1 = active	EPMI1 state 0 = inactive 1 = active
DBh (00)	Next Access Event Gen. Register 2	I/O Blocking Control 0 = Unblock 1 = Block I/O Next Access	SMI on Cool-down Clocking entry/exit 0 = Disable 1 = Enable	External EPMI4 pin polarity 0 = Active hi 1 = Active low	External EPMI3 pin polarity 0 = Active hi 1 = Active low	HDU_ ACCESS PMI#23 on Next Access 0 = No 1 = Yes	COM2_ ACCESS PMI#22 on Next Access 0 = No 1 = Yes	COM1_ ACCESS PMI#21 on Next Access 0 = No 1 = Yes	GNR2_ ACCESS PMI#20 on Next Access 0 = No 1 = Yes
DCh (00)	PMU SMI Source Register 3 Write 1 to clear	PMI #23 - HDU_ ACCESS 0 = Inactive 1 = Active	PMI #22 - COM2_ ACCESS 0 = Inactive 1 = Active	PMI #21 - COM1_ ACCESS 0 = Inactive 1 = Active	PMI #20 - GNR2_ ACCESS 0 = Inactive 1 = Active	PMI #19 - HDU_ TIMER 0 = Inactive 1 = Active	PMI #18 - COM2_ TIMER 0 = Inactive 1 = Active	PMI #17 - COM1_ TIMER 0 = Inactive 1 = Active	PMI #16 - GNR2_ TIMER 0 = Inactive 1 = Active
DDh (00)	PMU SMI Source Register 4	Reserved Write as 0.	Reserved. Write as 0.	Reserved. Write as 0	PMI #28 - DMA 0=Clear 1=Active (MVA)	PMI #27 - DOZE_ TIMER 0=Clear 1=Active	PMI #26 - RI 0=Clear 1=Active	PMI #25 - EPMI4 pin/ Cool-down Clocking 0=Clear 1=Active	PMI #24 - EPMI3 pin 0=Clear 1=Active
DEh (00)	Current Access Event Gen Register	HDU_ ACCESS PMI #23 on Current Access 0 = No 1 = Yes	COM2_ ACCESS PMI#22 on Current Access 0 = No 1 = Yes	COM1_ ACCESS PMI#21 on Current Access 0 = No 1 = Yes	GNR2_ ACCESS PMI#20 on Current Access 0 = No 1 = Yes	GNR1_ ACCESS PMI#15 on Current Access 0 = No 1 = Yes	KBD_ ACCESS PMI #14 on Current Access 0 = No 1 = Yes	DSK_ ACCESS PMI #13 on Current Access 0 = No 1 = Yes	LCD_ ACCESS PMI #12 on Current Access 0 = No 1 = Yes
DFh (00)	Activity Tracking Register	HDU_ ACCESS activity 0 = No 1 = Yes	COM2_ ACCESS activity 0 = No 1 = Yes	COM1_ ACCESS activity 0 = No 1 = Yes	GNR2_ ACCESS activity 0 = No 1 = Yes	GNR1_ ACCESS activity 0 = No 1 = Yes	KBD_ ACCESS activity 0 = No 1 = Yes	DSK_ ACCESS activity 0 = No 1 = Yes	LCD_ ACCESS activity 0 = No 1 = Yes
E0-E9h	Reserved								

Index	Name	7	6	5	4	3	2	1	0
EAh	PMU Source Register 5 (MVB)	IRQ/DRQ Driveback Trap PMI#36 0=Inactive 1=Active Write 1 to clear	Reserved. Write as read.	Reserved. Write as read.	Reserved. Write as read.	Reserved. Write as read.	Reserved Write as read	Reserved. Write as read.	Reserved. Write as read.
EB-F7h	Reserved								
F8h	Compact ISA Control Register 1 (MVB)	Inhibit MRD# and MWR# if SEL# asserted on memory cycle 0=No 1=Yes	Inhibit MRD# and MWR# if SEL# asserted on DMA cycle 0=No 1=Yes	Inhibit IORD# and IOWR# if SEL# asserted on I/O cycle 0=No 1=Yes	IRQ15 Assignment 0=IRQ15 1=RI	Reserved	Fast CISA memory cycle 0=Disable (ISA#=0) 1=Enable (ISA#=1)	CDIR Pin (pin 78) 0=RAS4# 1=CDIR	Compact ISA Interface (reassigns pins 173, 186) 0=Disable 1=Enable
F9h	Compact ISA Control Register 2 (MVB)	SPKD Signal Driving 0=Always, per AT spec. 1=Synchronously, per CISA spec.	End-of-Interrupt Hold - Delays 8259 recognition of EOI command to prevent false interrupts. 00=None 01=1 ATCLK 10=2 ATCLKs 11=3 ATCLKs		Stop Clock Count bits CC[2:0] - Stop clock cycle indication to CISA devices of how many ATCLKs to expect before the clock will stop. 000=Reserved 001=1 ATCLK (default) ... 111=7 ATCLKs			Generate CISA Stop Clock Cycle (if not already stopped): 00=Never 01=On STPCLK# cycles to the CPU (hardware) 10=Immediately (software) 11=Reserved	
FAh	Compact ISA Control Register 3 (MVB)	Reserved. Write as read	Reserved. Write as read.	Reassign EPMI3 as RI 0=No 1=Yes Use in case RI is assigned as SEL#/ATB#	Reassign EPMI4 as IOCHCK# 0=No 1=Yes Use in case IOCHCK# is assigned as KBCRSTIN	Resume from Suspend on SEL#/ATB# low 0=Disabled 1=Enabled	CMD# State during Suspend 0=Driven inactive (high) 1=Driven low	Driveback Cycle Handling 0=Pass DRQs and IRQs 1=Latch info and generate SMI	Configuration Cycle Generation 0=No action 1=Run cycle using Scratchpad
FBh	Reserved								
FCh	Scratchpad Register 7	General purpose storage byte For CISA Driveback Cycle. IRQ phase information, low byte (read-only) (MVB)							
FDh	Scratchpad Register 8	General purpose storage byte For CISA Driveback Cycle: IRQ phase information, high byte (read-only) (MVB)							
FEh	Scratchpad Register 9	General purpose storage byte For CISA Driveback Cycle: DRQ phase information, low byte (read-only) (MVB)							
FFh	Scratchpad Register 10	General purpose storage byte For CISA Driveback Cycle: DRQ phase information, high byte (read-only) (MVB)							

5.1 Index of Configuration Registers

6.0 Electrical Specification

6.1 Absolute Maximum Ratings

Symbol	Description	Min	Max	Units
VDD/VDDS	Operating voltage	3.0	6.5	V
VIH	Input high voltage	—	VDDS + 0.3	V
VIL	Input low voltage	-0.3	—	V
TA	Ambient Temperature	0	70	° C

NOTE Permanent device damage may occur if Absolute Maximum Ratings are exceeded.

6.2 DC Characteristics

6.2.1 5V DC Characteristics

(Ta = 0° C to 70° C, VDDS = 5V± 5%)

Symbol	Description		Min	Max	Units
VIH	Input high voltage	TTL Level	2.0		V
		CMOS Level	3.4		V
VT+	Schmitt Trigger Input Voltage (TTL)	Positive going threshold		2.2	V
VT-	Schmitt Trigger Input Voltage (TTL)	Negative going threshold	0.8		V
VIL	Input low voltage	TTL Level		0.8	V
		CMOS Level		0.8	V
VOH	Output high voltage (at rated drive current)	CMOS Level	2.4		V
VOL	Output low voltage (at rated drive current)	CMOS Level		0.45	V
IIH	Input leakage current, VIN = VDDS		-10	10	uA
IIL	Input leakage current, VIN = GND		-10	10	uA
IOZ	Tri-state leakage current 0.45V < VOUT < VDDS		-10	10	uA
ICC	Operating current	33MHz, on-mode*		100	mA
		suspend mode*		100	uA

* Assumes controlled leakage currents.

6.2.2 3.3V DC Characteristics

(Ta = 0° C to 70° C, V_{DD} = 3.3V± 5%)

Symbol	Description		Min	Max	Units
V_{IH}	Input high voltage		2.0		V
V_{T+}	Schmitt Trigger Input Voltage (TTL)	Positive going threshold		2.2	V
V_{T-}	Schmitt Trigger Input Voltage (TTL)	Negative going threshold	0.8		V
V_{IL}	Input low voltage			0.8	V
V_{OH}	Output high voltage (at rated drive current)		2.4		V
V_{OL}	Output low voltage (at rated drive current)			0.45	V
I_{IH}	Input leakage current, V_{IN} = V_{DD}		-10	10	uA
I_{IL}	Input leakage current, V_{IN} = GND		-10	10	uA
I_{OZ}	Tri-state leakage current 0.45V < Vout < Vdd		-10	10	uA
I_{CC}	Operating current	33MHz, on-mode*		50	mA
		suspend mode*		40	uA

* Assumes controlled leakage currents.

6.3 AC Characteristics

Symbol	Description	Min	Max	Units
C_{IN}	Input capacitance		10	pF
C_{OUT}	Output capacitance		10	pF
C_{IO}	I/O Capacitance		12	pF

6.4 Timing Characteristics

CPU interface = 3.3V, all other interfaces = 5V.

Sym.	Parameter	Min.	Typ.	Max.	Unit
6.4.1	**Cache Timing**				
t1	CPU Bus Definition Valid to BEOE#/BOOE# active delay (for 2-X-X-X leadoff cycles only)	10		20	ns
t2	CLK rising edge to BEOE#/BOOE# active delay	5		20	ns
t3	CLK rising edge to BEOE#/BOOE# inactive delay			20	ns
t4	CLK rising edge to BRDY# active delay	5		15	ns
t5	CLK rising edge to BRDY# inactive delay	5		15	ns
t7	CLK falling edge to ECAWE#/OCAWE# active delay			15	ns
t8	CLK falling edge to ECAWE#/OCAWE# inactive delay (0 wait state write, dirty)			20	ns
t9	CLK falling edge to ECAWE#/OCAWE# inactive delay (0 wait state write, not dirty, or 1 wait state write ,dirty)			16	ns
t9A	CLK rising edge to ECAWE#/OCAWE# active delay with DRAM at speed of 3-2-2-2			20	ns
t9B	CLK falling edge to ECAWE#/OCAWE# inactive delay with DRAM at speed of 3-2-2-2			17	ns
t10	CLK falling edge to TAGWE# active delay (for updating DIRTY bit)			15	ns
t11	CLK falling edge to TAGWE# inactive delay (for updating DIRTY bit)	5		15	ns
t12	CLK rising edge to BRDY# active delay (for cache write cycles)	5		20	ns
t13	CLK rising edge to BRDY# inactive delay (for cache write cycles)	5		10	ns
t14	CLK rising edge to TAGWE# active delay (for updating TAG)	5		15	ns
t15	CLK rising edge to TAGWE# inactive delay (for updating TAG)	5		15	ns
t16	CPU Bus Definition Valid to ECA3/ECA2 valid delay			16	ns
t17	CLK rising edge to ECA3/ECA2 invalid delay			10	ns
t161	HITM# signal setup time	10			ns
t162	ADS# active to HOLD active delay	10		30	ns
t167	FBCLK rising edge to EADS# active delay	5		25	ns
t168	FBCLK rising edge to EADS# inactive delay	5		25	ns
6.4.2	**DRAM Timing**				
t18	CLK rising edge to CAS# active delay	5		20	ns

Sym.	Parameter	Min.	Typ.	Max.	Unit
t19	CLK rising edge to CAS# inactive delay	5		20	ns
t20	CLK falling edge to CAS# active delay (for 3-2-2-2 cache burst cycles only)	5		20	ns
t21	CLK rising edge to CAS# inactive delay (for 3-2-2-2 cache burst cycles only)			15	ns
t22	CLK rising edge to RAS# inactive delay	5		20	ns
t23	CLK rising edge to RAS# active delay	5		20	ns
t24[1]	CLK rising edge to Column Address Valid delay	5		15	ns
t25	CLK rising edge to Row Address Hold Time	8		30	ns
t26	CLK rising edge to DWE# active delay	5	15	30	ns
t27	CLK rising edge to DWE# inactive delay	5	15	30	ns
t29	RAS# Precharge Time	80			ns
t30	CAS# Precharge Time	10			ns
t31	CAS# active to RAS#[1-0] active delay	25			ns
t32	CAS# inactive to RAS#[1-0] inactive delay	25			ns
t33	RAS# active to RAS# active delay (during refresh)			55	ns
t34	RAS# inactive to RAS# active delay (during refresh)			55	ns
6.4.3	**AT Bus Timing**				
t46	ATCLK falling edge to BALE active delay	5		30	ns
t47	ATCLK rising edge to BALE inactive delay	5		30	ns
t48	ATCLK falling edge to CMD active delay	5		30	ns
t49	ATCLK rising edge to CMD active delay	5		30	ns
t50	ATCLK rising edge to CMD inactive delay	5		30	ns
t51	M16# to ATCLK rising edge Setup Time	8			ns
t52	M16# from ATCLK rising edge Hold Time	8			ns
t53	IO16# to ATCLK rising edge Setup Time	10			ns
t54	IO16# from ATCLK rising edge Hold Time	10			ns
t55	CHRDY to ATCLK rising edge Setup Time	12			ns
t56	CHRDY from ATCLK rising edge Hold Time	12			ns
t57	FBCLK falling edge to Hold active delay	5		16	ns
t58	FBCLK rising edge to Hold inactive delay	5		16	ns
t59	ATCLK rising edge to REFRESH# active delay	8		30	ns
t60	ATCLK rising edge to REFRESH# inactive delay	8		30	ns
t70	0WS# to ATCLK falling edge Setup Time				
t71	0WS# from ATCLK falling edge Hold Time				

Sym.	Parameter	Min.	Typ.	Max.	Unit
t85	A[9:0] valid to KBDCS# active delay			30	ns
t86	A[9:0] invalid to KBDCS# inactive delay			30	ns
t106	CD[31:0] valid to SD[15:0] valid delay	9		17	ns
t107	CD[31:0] valid to MP[3:0] valid delay	8		18	ns
t108	CD[31:0] invalid to SD[15:0] invalid delay	8		18	ns
t109	CD[31:0] invalid to MP[3:0] invalid delay	8		18	ns
t110	SD[15:0] valid to CD[31:0] valid delay	8		16	ns
t111	SD[15:0] valid to MP[3:0] valid delay	12		20	ns
t112	SD[15:0] invalid to CD[31:0] invalid delay	8		16	ns
t113	SD[15:0] invalid to MP[3:0] delay	12		20	ns
t114	SD[15:8] valid to SD[7:0] delay	8		16	ns
t115	SD[15:8] invalid to SD[7:0] delay	9		18	ns
t116	CHCK# active to NMI delay		18		ns
t131	ATCLK edge to SDENL# delay		1/2 ATCLK		
t132	ATCLK edge to SDENH# delay		1/2 ATCLK		
t133	FBCLK rising edge to ROMCS# active delay			30	ns
t134	ATCLK falling edge to XDIR active delay	5		20	ns
t136	ATCLK rising edge to XDIR inactive delay	5		20	ns
t137	ATCLK rising edge to I/O CMD# inactive delay	5		20	ns
t138	ATCLK falling edge to I/O CMD# active delay	5		20	ns
t171	CPUCLK rising edge to SDENH# inactive delay	5			ns
t172	CPUCLK rising edge to SDENL# inactive delay	5			ns
t173	CPUCLK rising edge to SDIR inactive delay	5			ns
t174	AT bus turn-around time		1/2 ATCLK		ns
t175	Next ATCLK edge to SDENH# active delay			20	ns
t176	Next ATCLK edge to SDENL# active delay			20	ns
t177	Next ATCLK edge to SDIR active delay			20	ns
6.4.4	**Reset and Local Bus Timing**				
t102	FBCLK/CPUCLK rising edge to CPURST/SRESET active delay	8	10	15	ns
t105	FBCLK/CPUCLK rising edge to CPURST/SRESET inactive delay	8	10	15	ns
t117	ADS# Setup Time	8			ns
t170	ADS# Hold Time	5			ns
t118	LDEV# Setup Time	8			ns

Sym.	Parameter	Min.	Typ.	Max.	Unit
t119	LDEV# Hold Time	5			ns
t121	RDY# active to CPUCLK rising edge delay	8			ns
t122	RDY# inactive from CPUCLK rising edge delay	5			ns
t165	CPU RESET Time		190		CPUCLK
6.4.5	**Power Management Timings**				
t140	ATCLK rising edge to PIO active delay			40	ns
t141	ATCLK rising edge to MA bus valid delay			25	ns
t142	ATCLK rising edge to MA bus invalid delay			20	ns
t143	ATCLK rising edge to PPWRL active delay			25	ns
t144	ATCLK rising edge to PPWRL inactive delay			25	ns
t146	Stop clock delay setup in register B0h				
t150	Stop clock delay setup in register B0h				
t151	7/8 resume recovery time				
t152	1/8 resume recovery time				
t158	PPWRL width (ATCLK not running)		1/2 (15us)		SQWIN CLK
t169	PPWRL width (ATCLK running)		1		ATCLK
t180	RDY# inactive to Hold active delay		1		ATCLK

1. MA bus timing delays depend heavily on bus loading. Timing t24 assumes 12 DRAM devices on the bus. Add approximately a 7ns delay for each additional bank of 12 devices. If heavy-duty MA bus drive is enabled (bit A1h[4]=1), add a 5ns delay for each additional bank.

6.5 Timing Diagrams

Figure 6-1 ROM Cycle with SDENH#, SDENL# (L2 Cache Enabled)

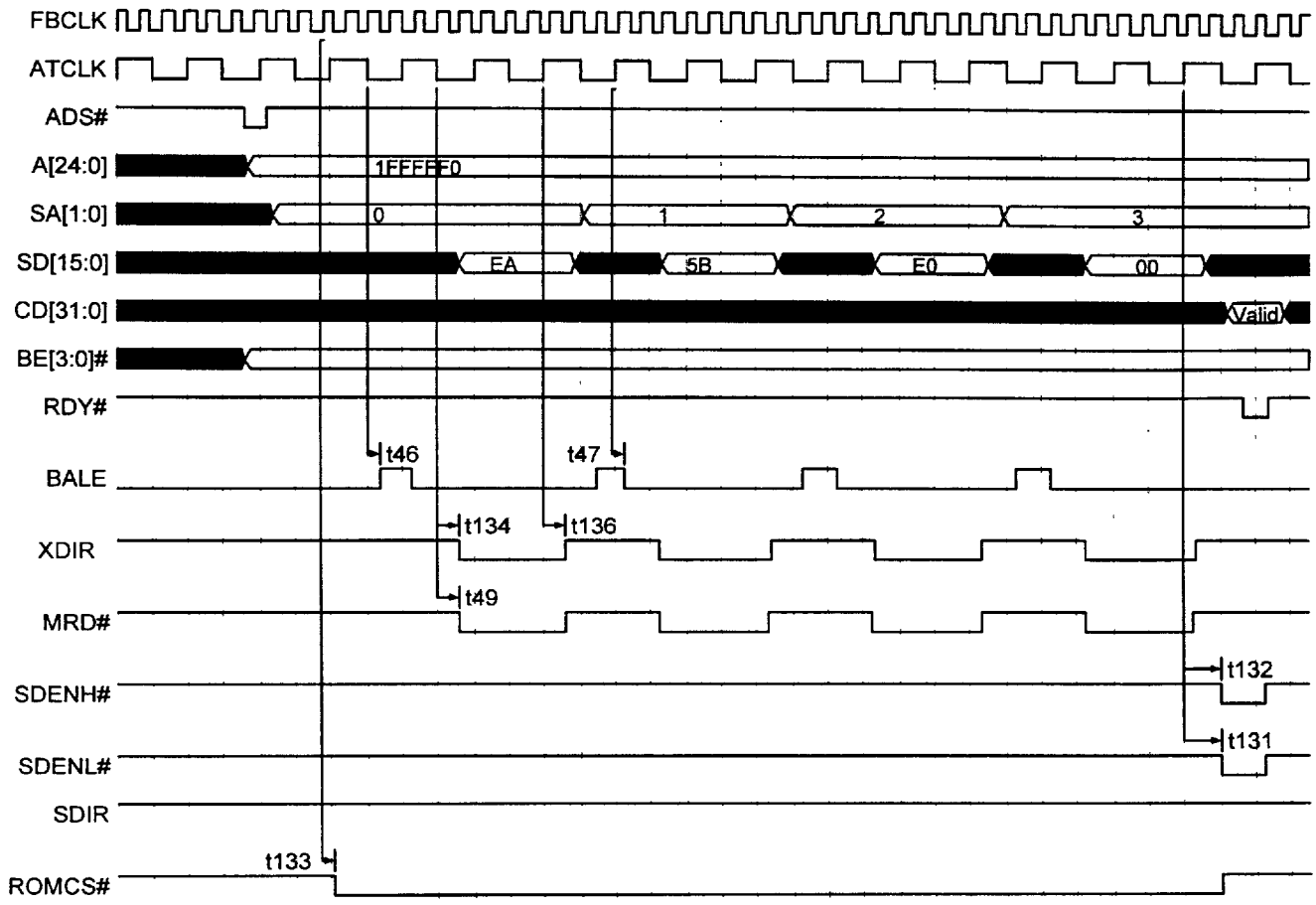


Figure 6-2 ISA Bus Cycle

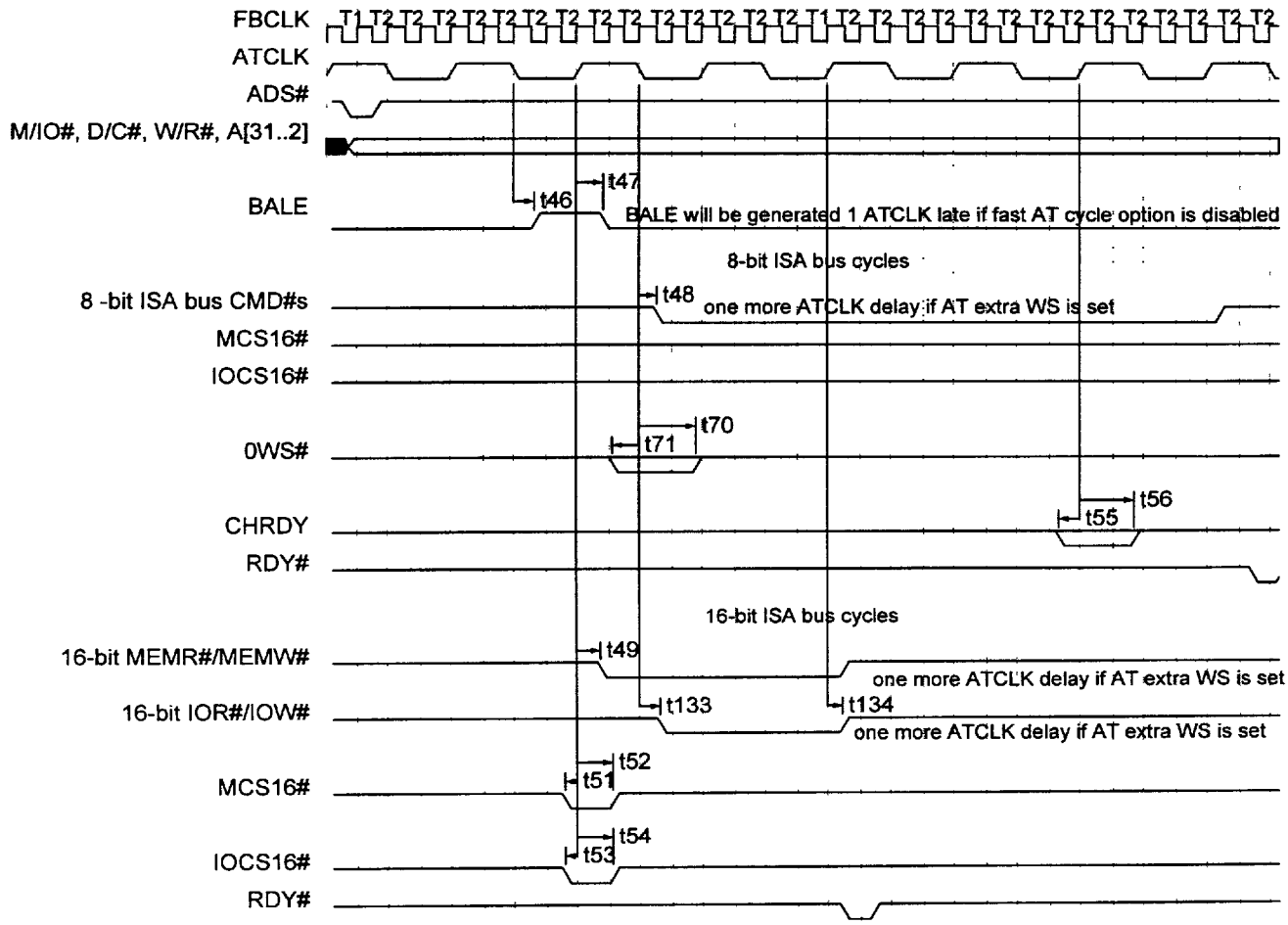


Figure 6-3 Keyboard Controller Access Cycle

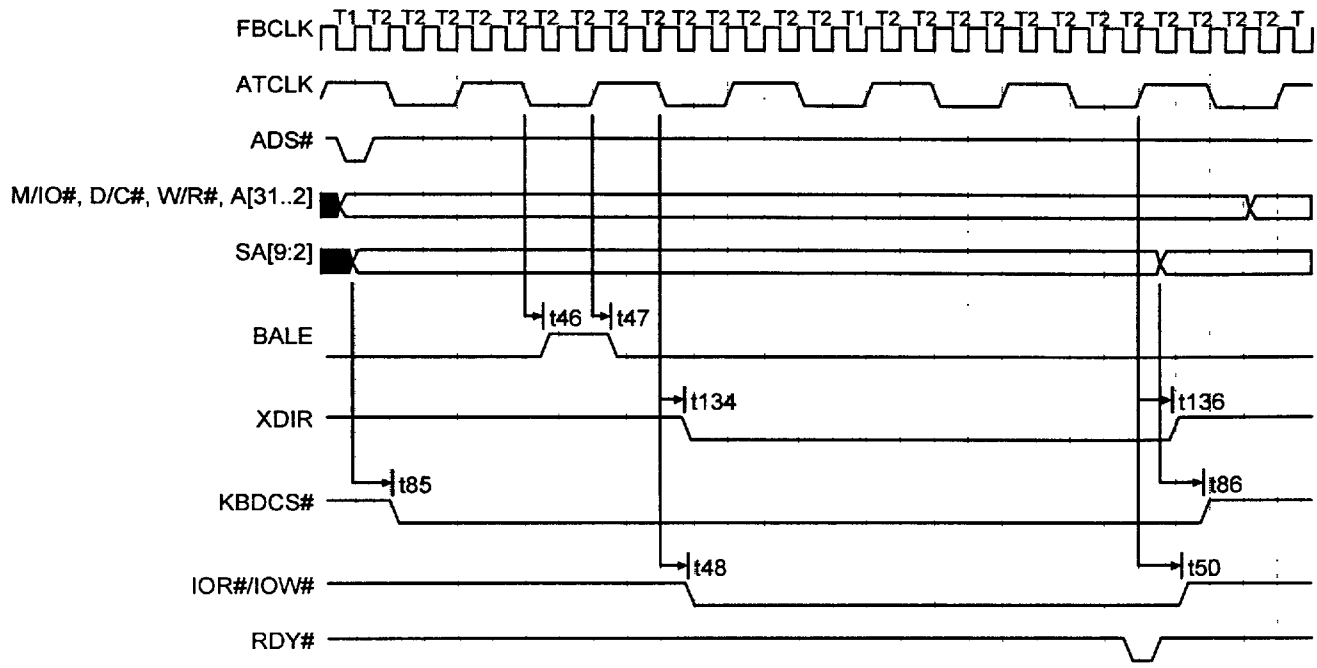


Figure 6-4 CD[31:0] to SD[15:0] and MP[3:0] Valid and Invalid Delay

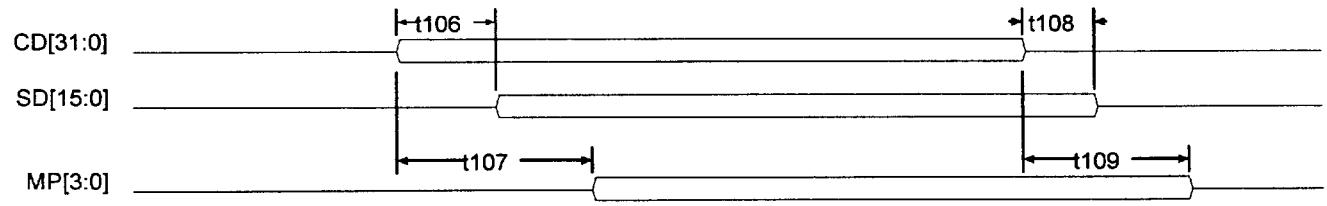


Figure 6-5 SD[15:0] to CD[31:0] and MP[3:0] Valid and Invalid Delay

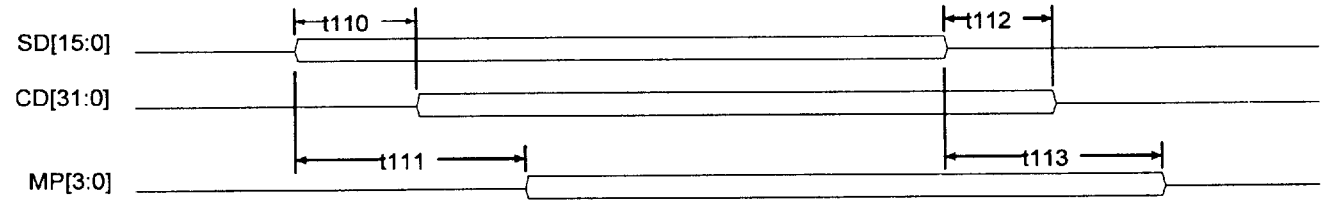


Figure 6-6 Data Valid and Invalid Delay Between SD[15:8] and SD[7:0] Data Swapping

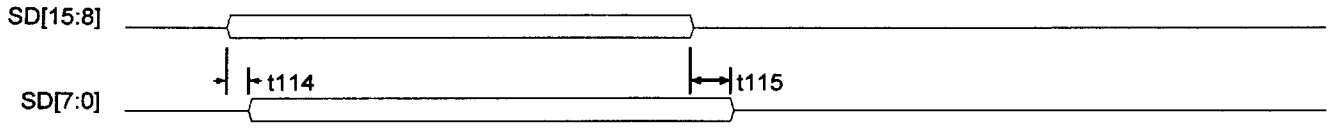


Figure 6-7 NMI Valid Delay Related to IOCHK#

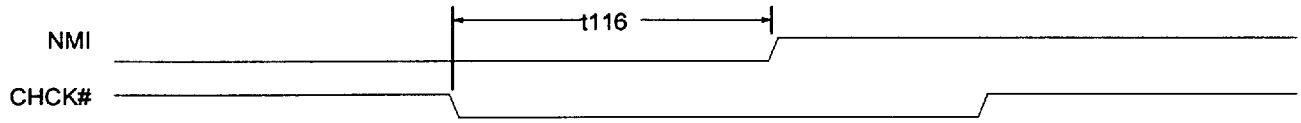


Figure 6-8 L2 Cache Read Miss - Dirty, Double Bank Cache

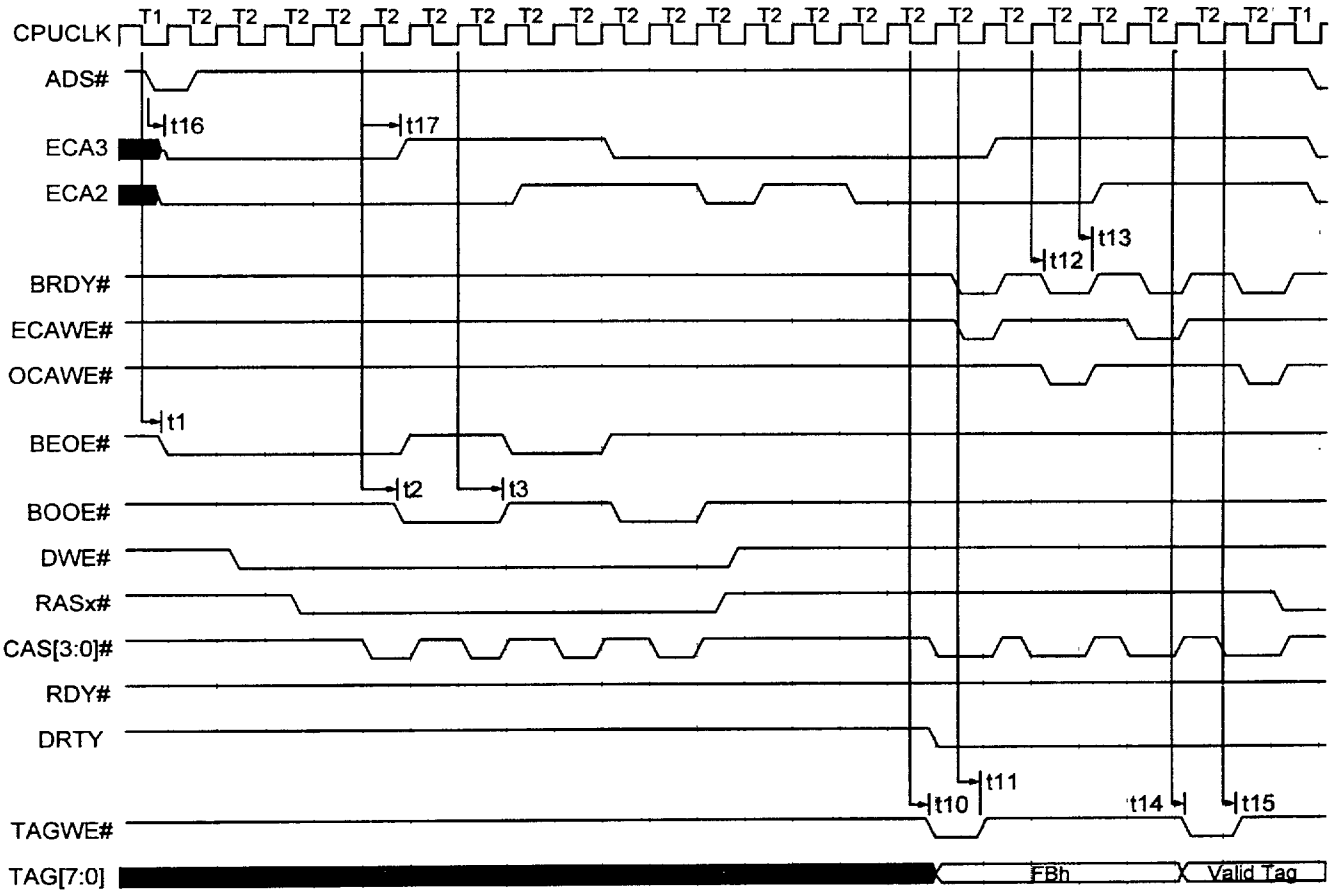


Figure 6-9 L2 Cache Write 0 Wait State, Not Dirty

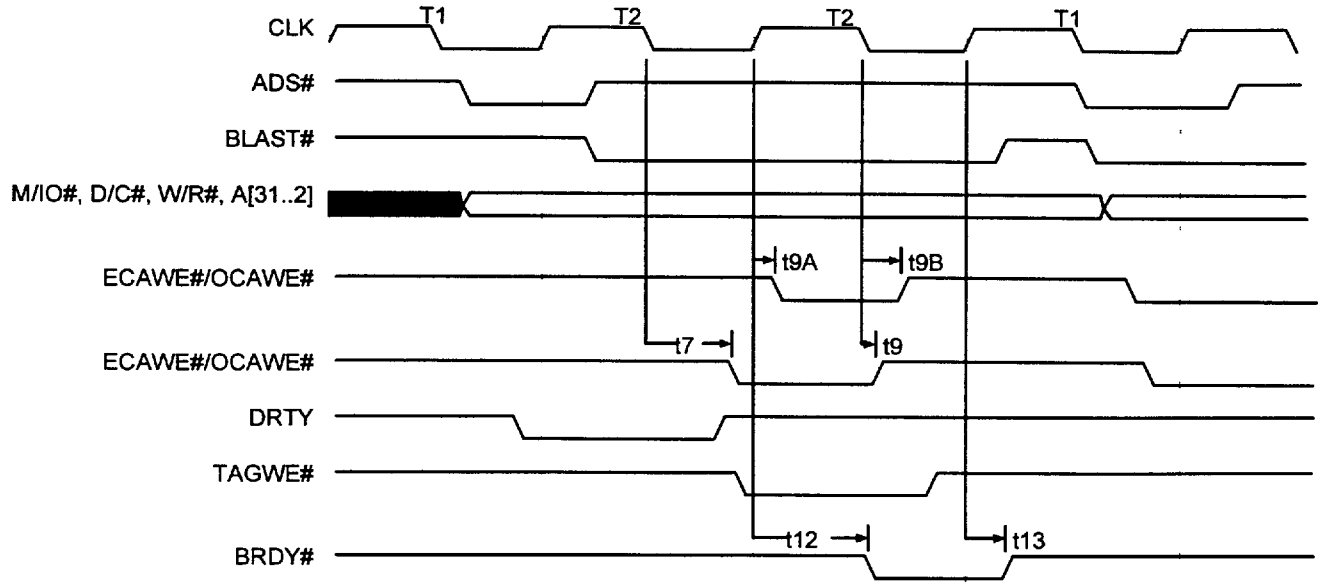


Figure 6-10 L2 Cache Write, Dirty

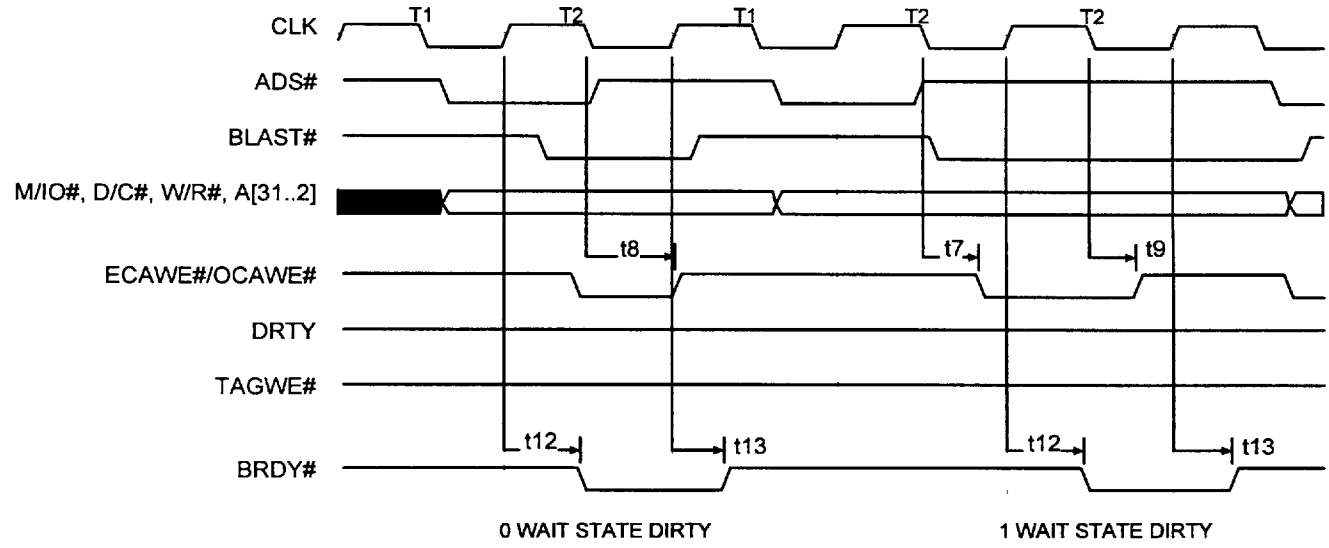


Figure 6-11 L2 Cache Burst Read 3-2-2-2 For Double Bank

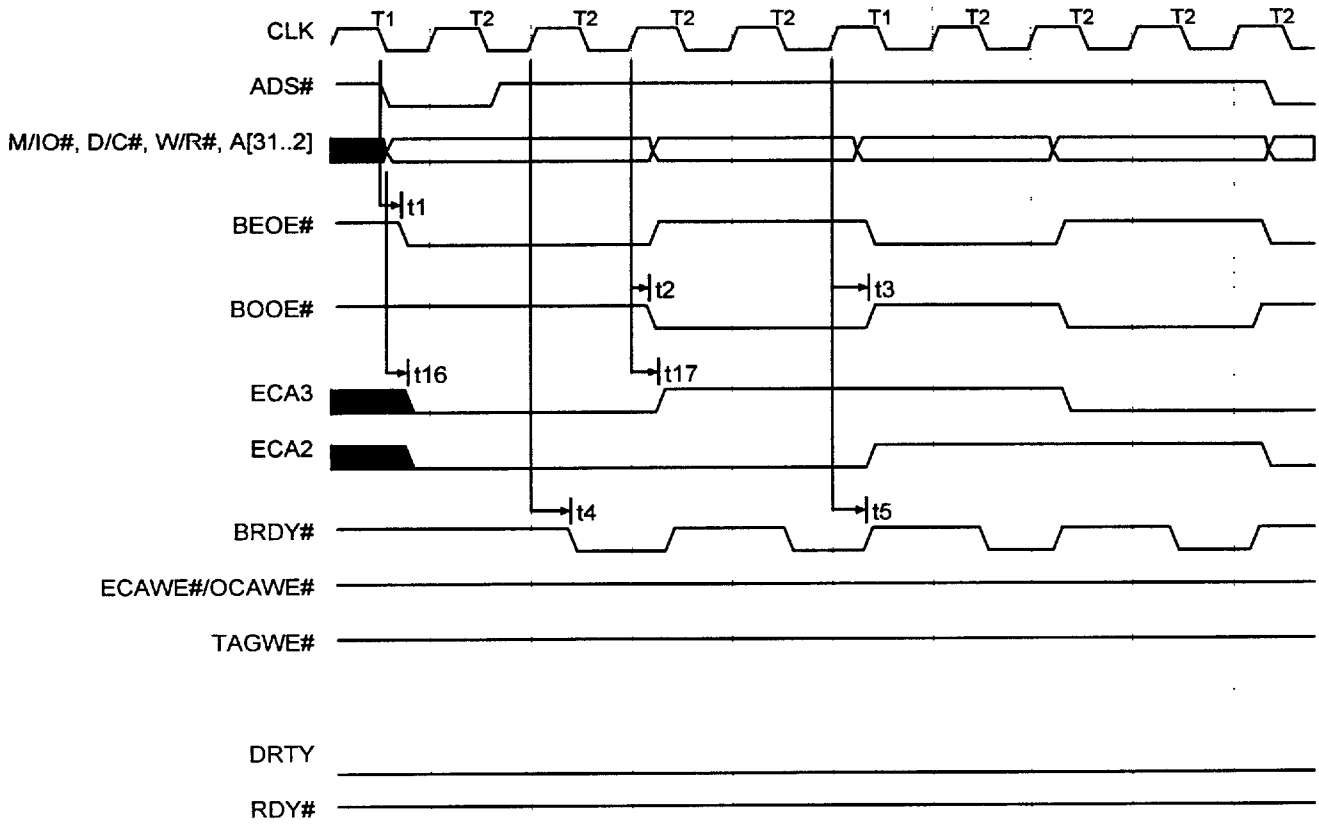


Figure 6-12 L2 Cache Burst Read 3-2-2-2 For Single Bank

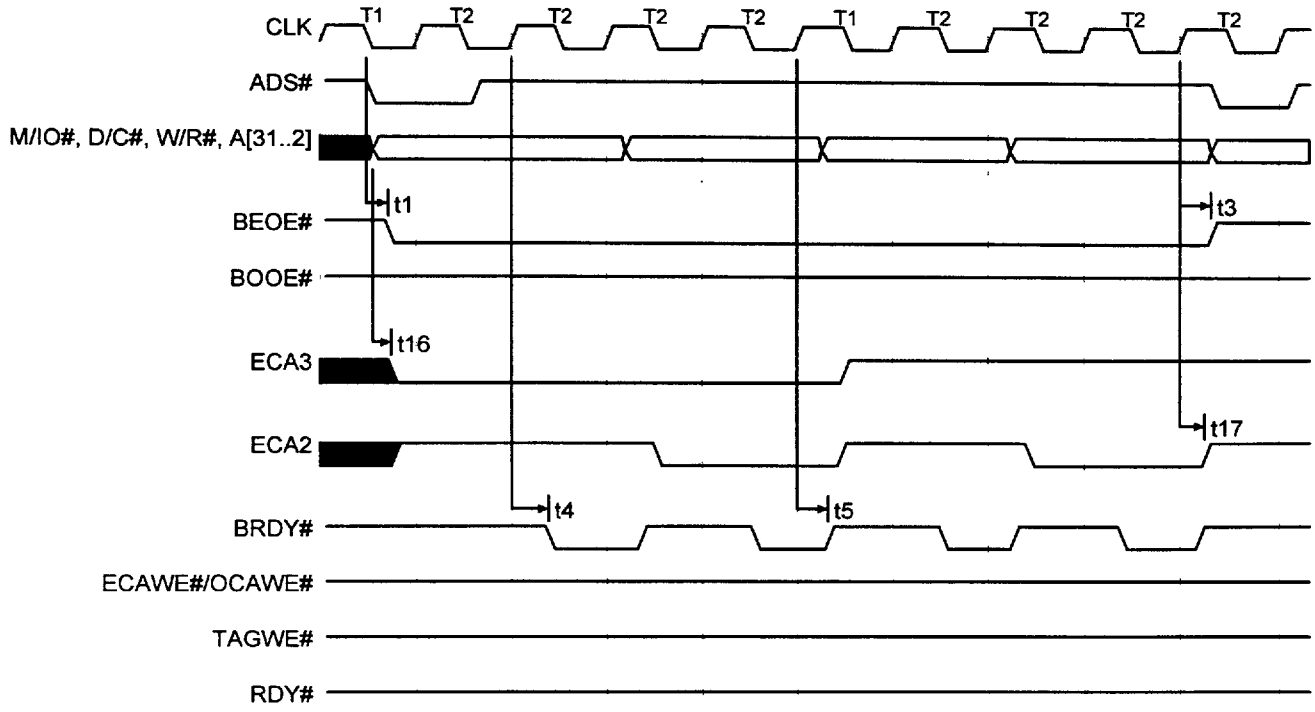


Figure 6-13 L2 Cache Burst Read 2-1-1-1 Cycle For Double Bank

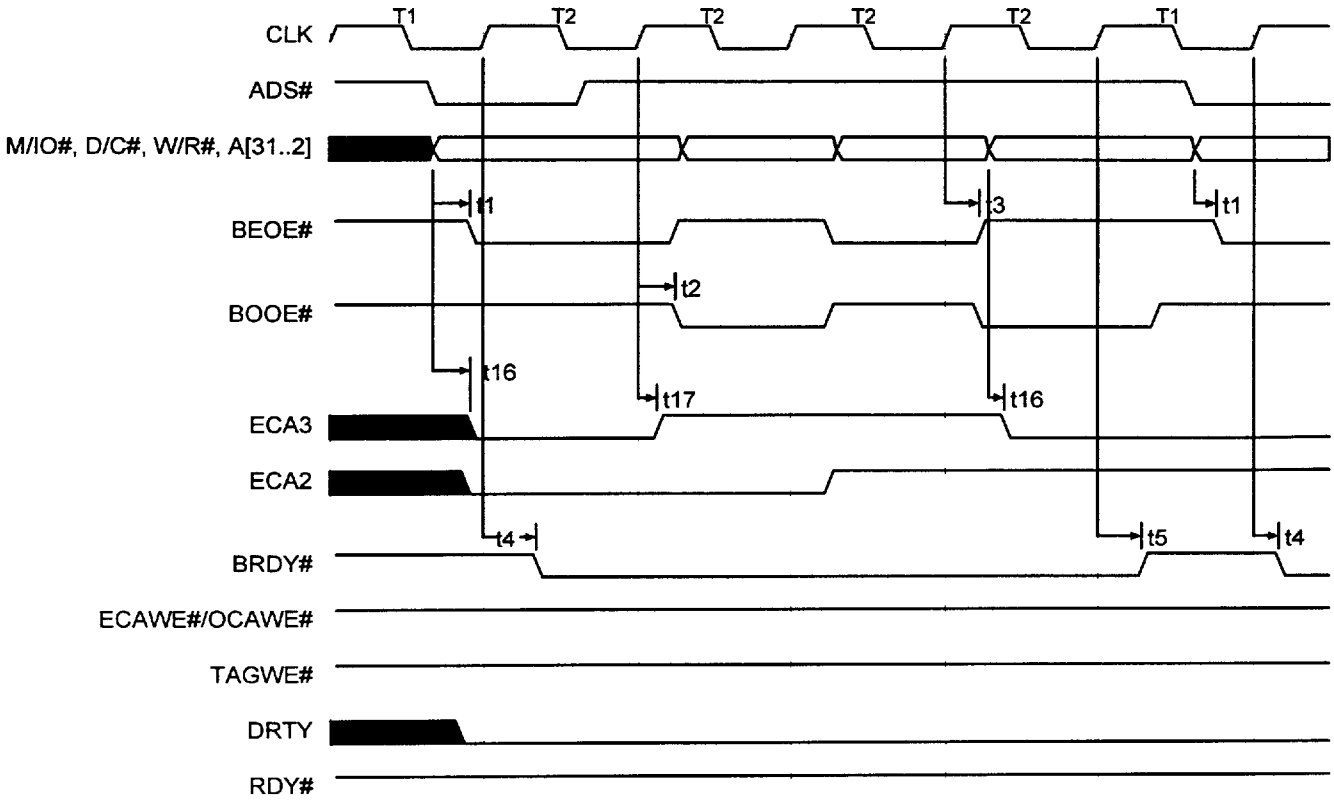
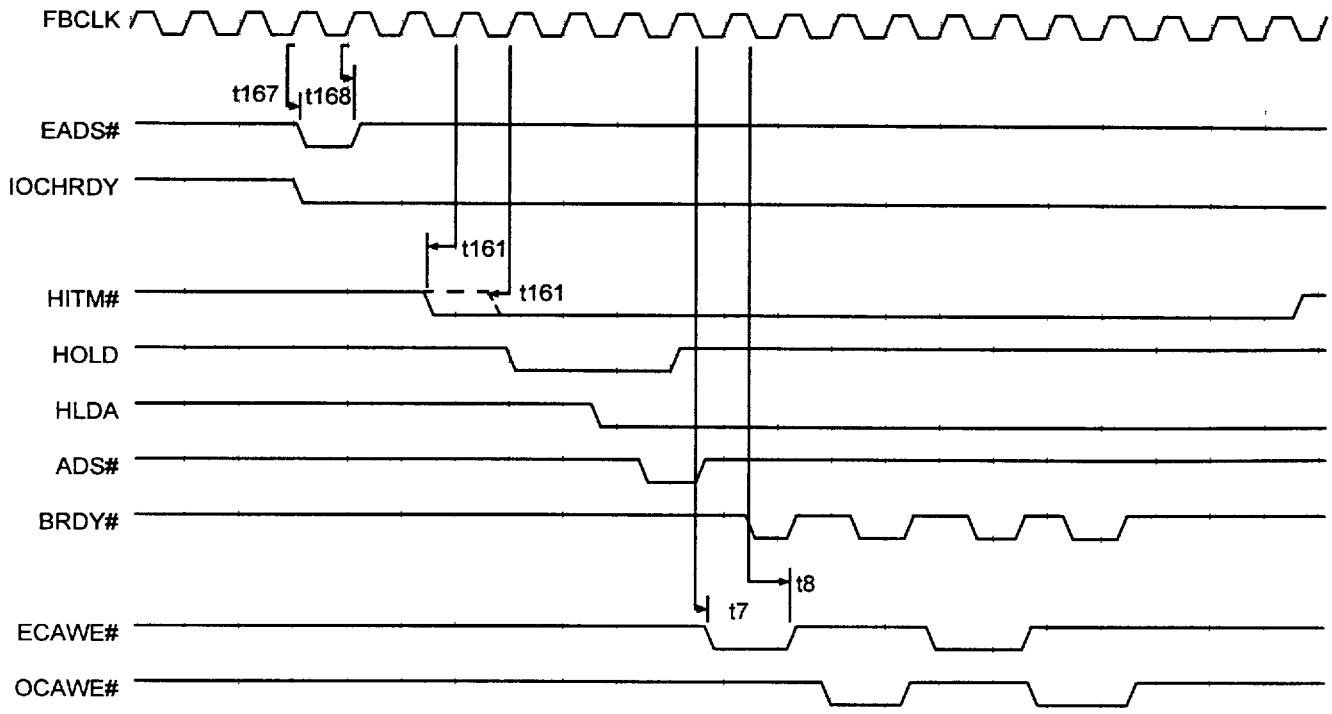
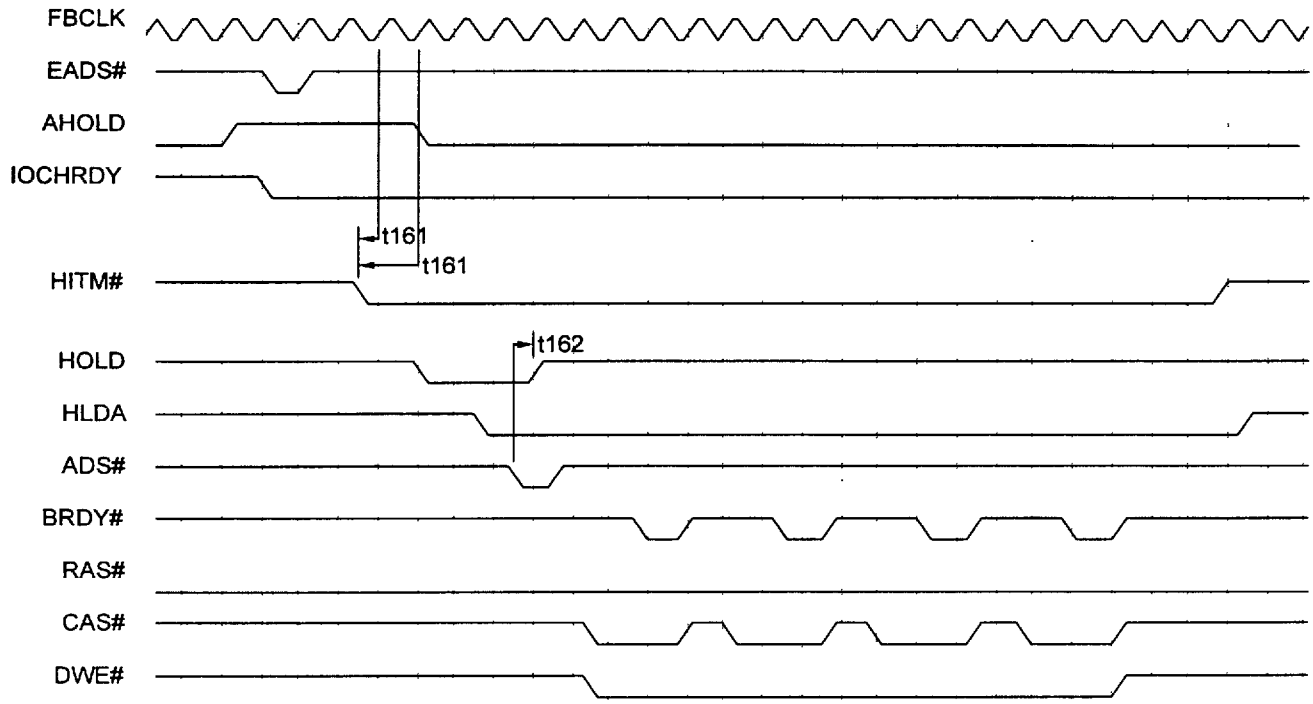


Figure 6-14 HITM# Active With L2 Cache Hit, L2 Cache 3-2-2-2 Write Cycle (Cache 0 Wait Write)



NOTE Sampling of HITM# is programmable to be in either second or third clock after EADS#.

Figure 6-15 HITM# Signal Active, Burst Write Back Cycle, 1 Wait State DRAM Write Setup



NOTE Sampling of HITM# is programmable to be in either second or third clock after EADS#.

Figure 6-16 Refresh Cycle

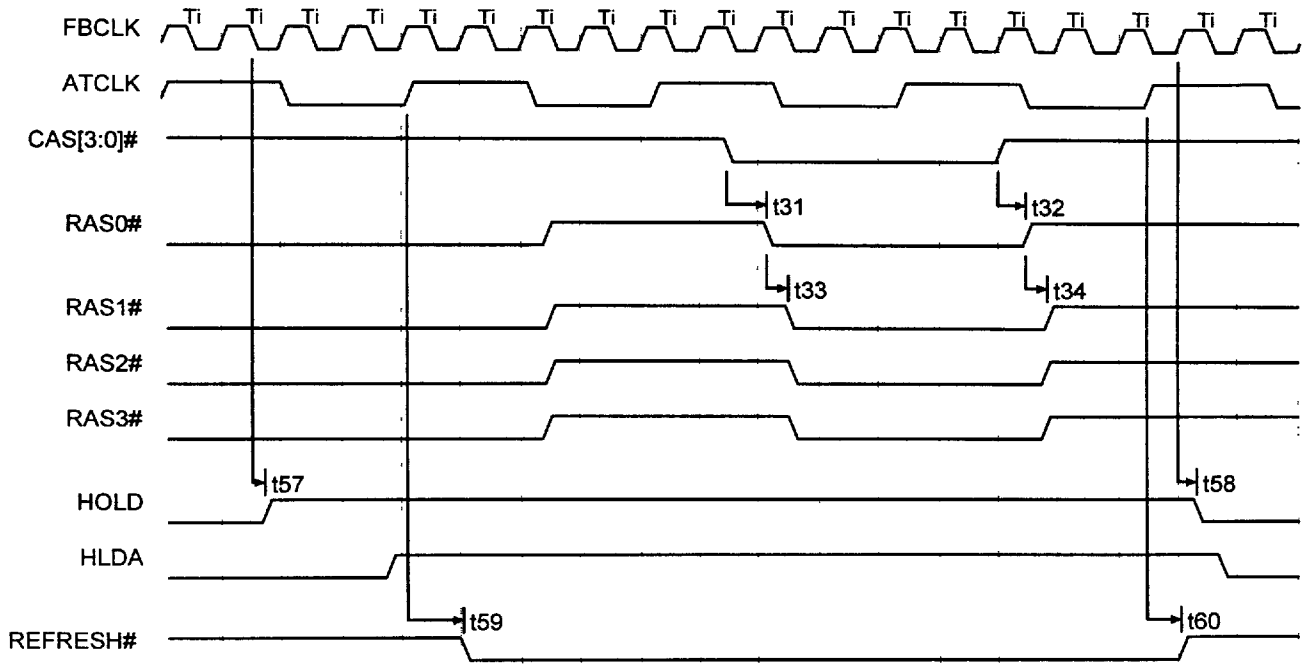


Figure 6-17 DRAM Burst Read 5-4-4-4 (Page Hit)

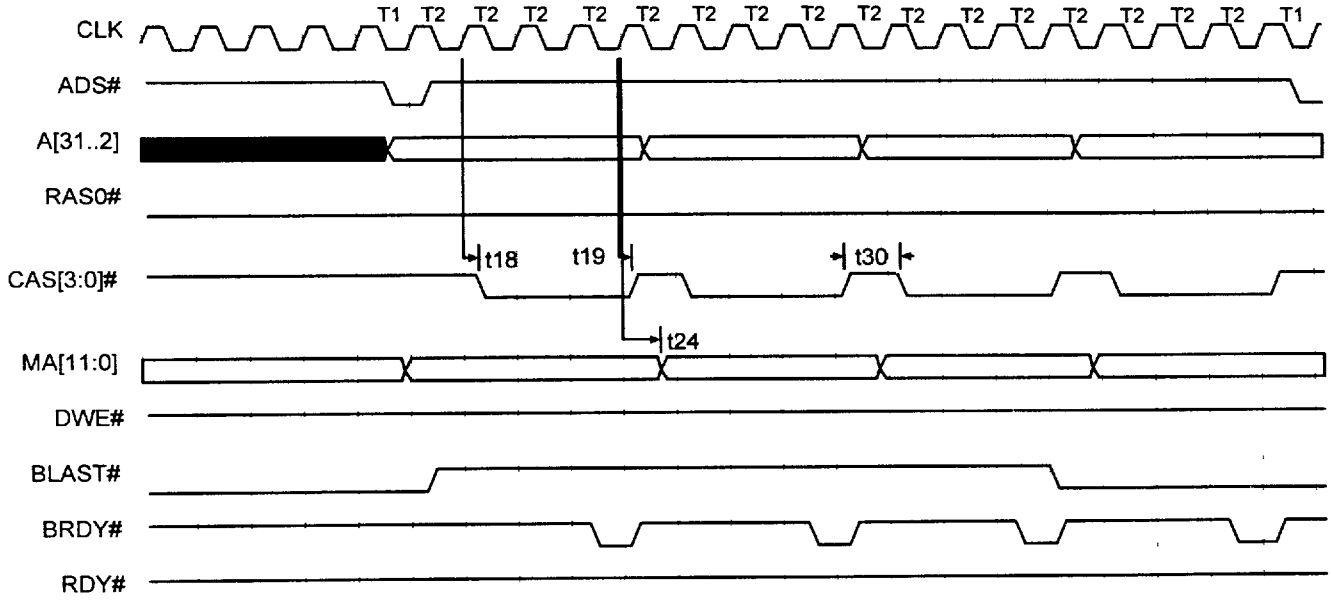


Figure 6-18 DRAM Burst Read 3-2-2-2 (Page Miss)

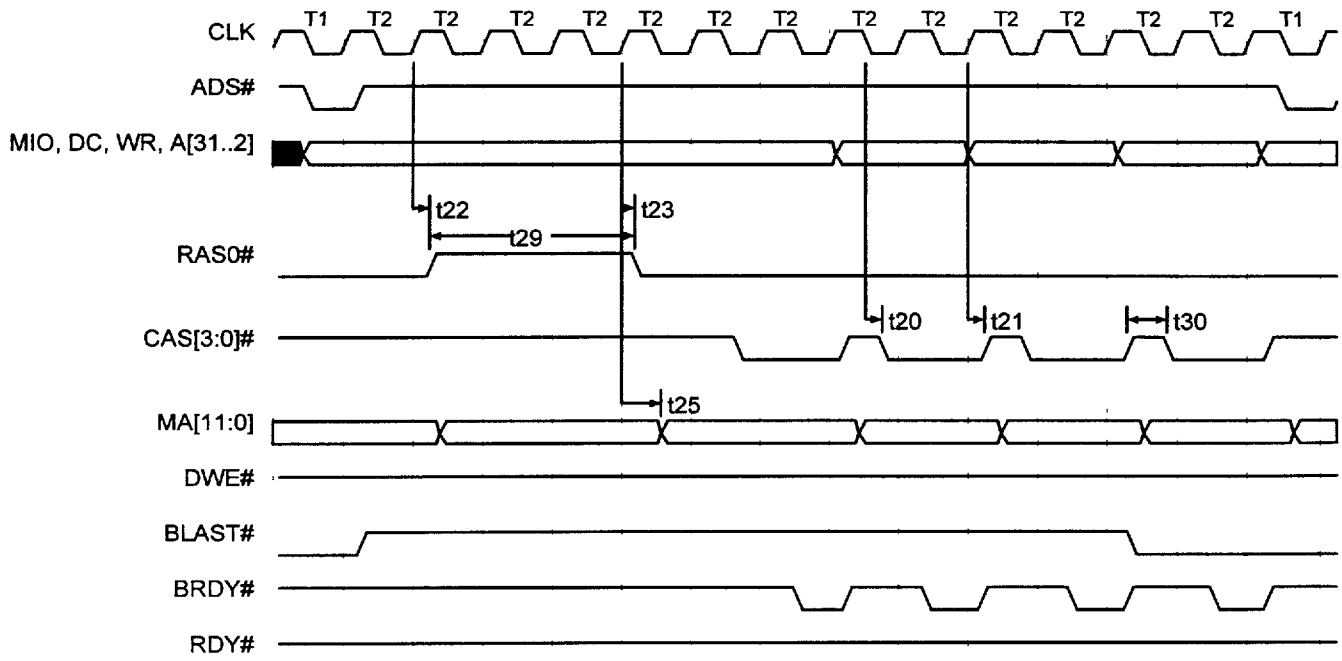


Figure 6-19 DRAM Burst Read 3-2-2-2 (Page Hit)

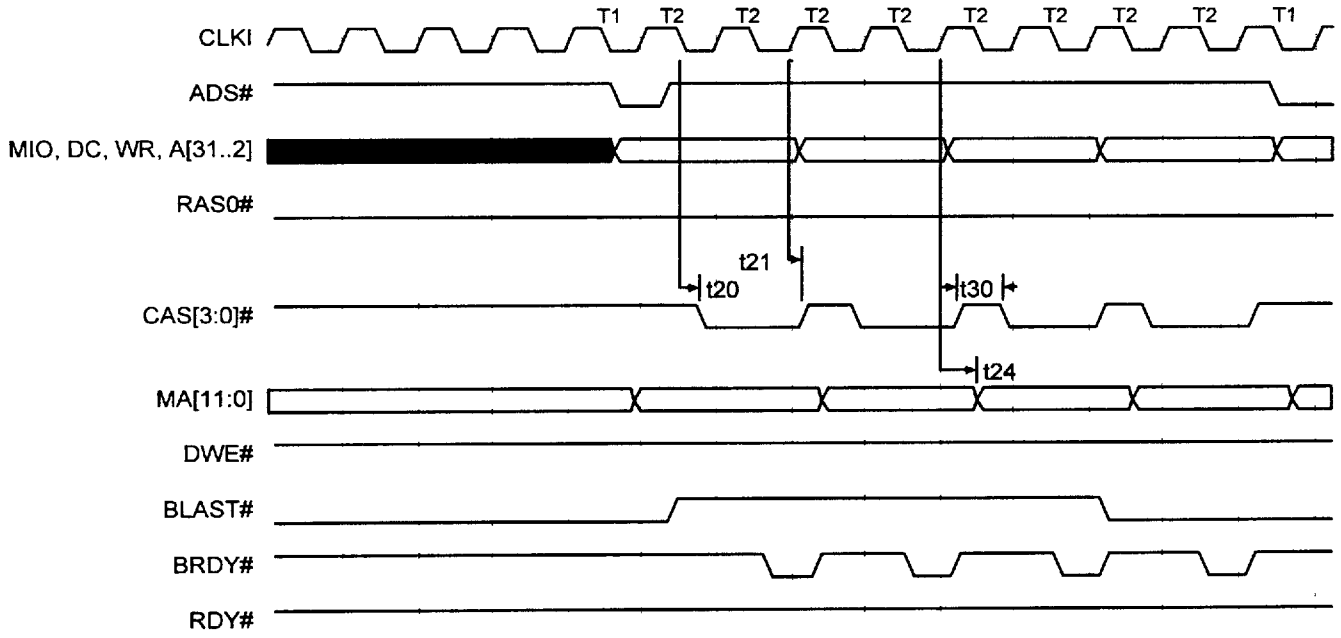


Figure 6-20 DRAM Burst Read 4-3-3-3 (Page Hit)

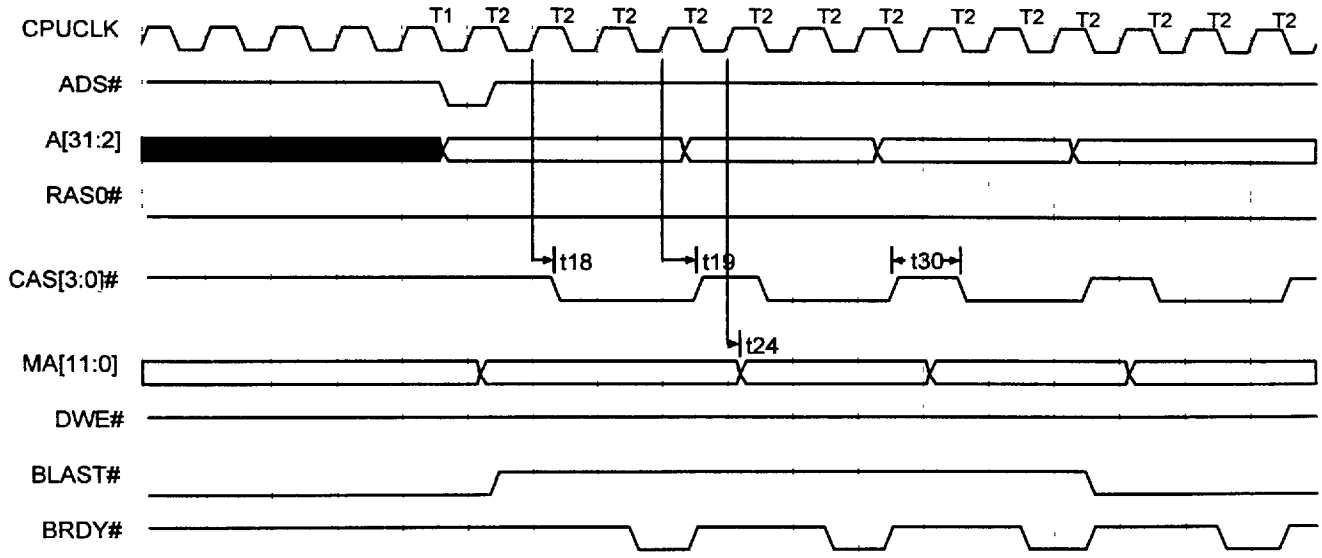


Figure 6-21 DRAM Burst Read 4-3-3-3, 1 Wait Page Miss

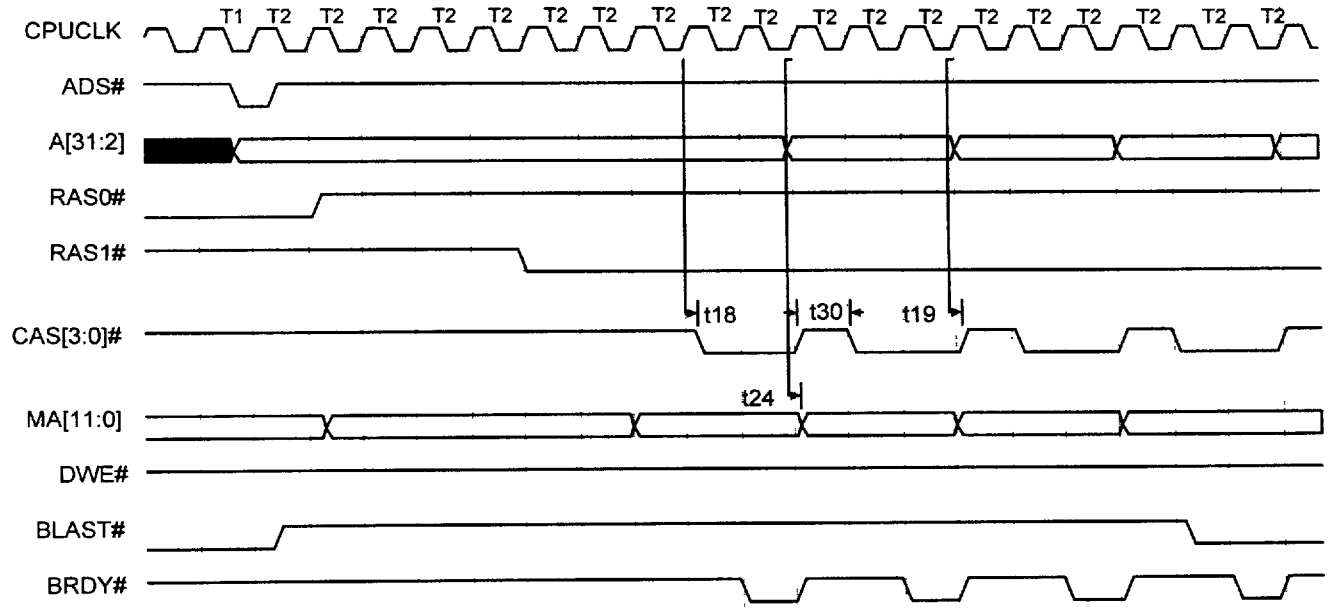


Figure 6-22 DRAM Burst Read 4-3-3-3, 0 Wait Page Miss

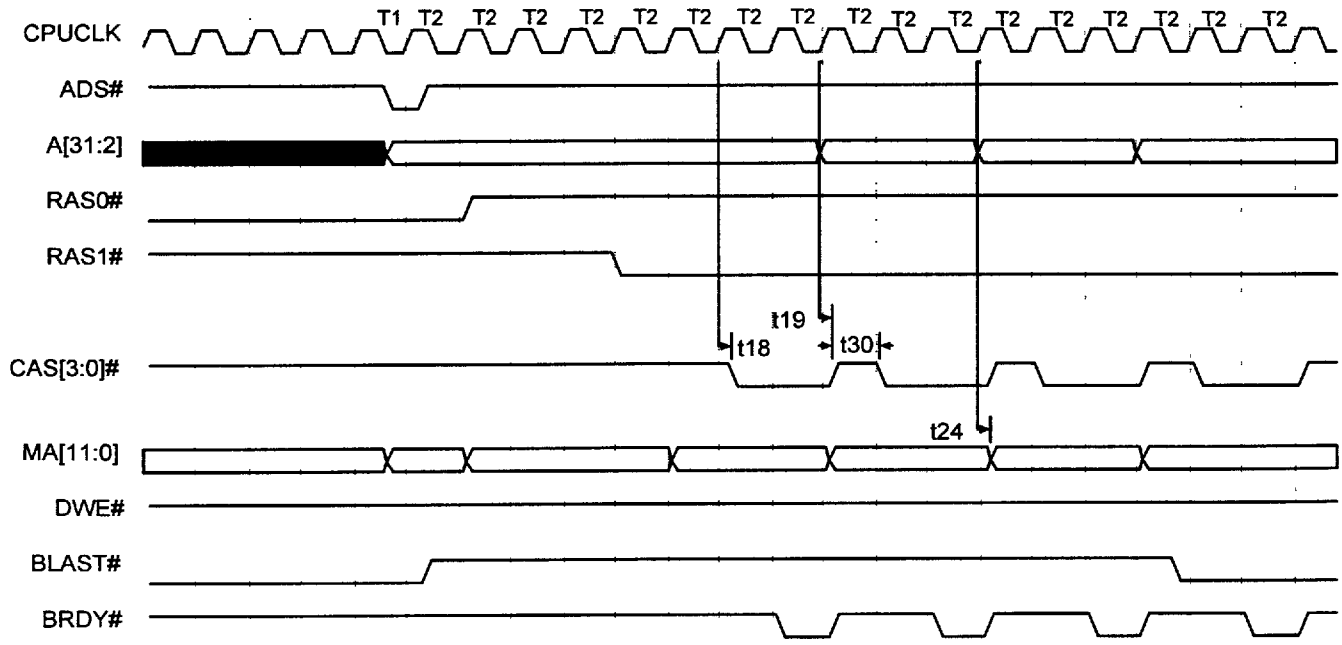
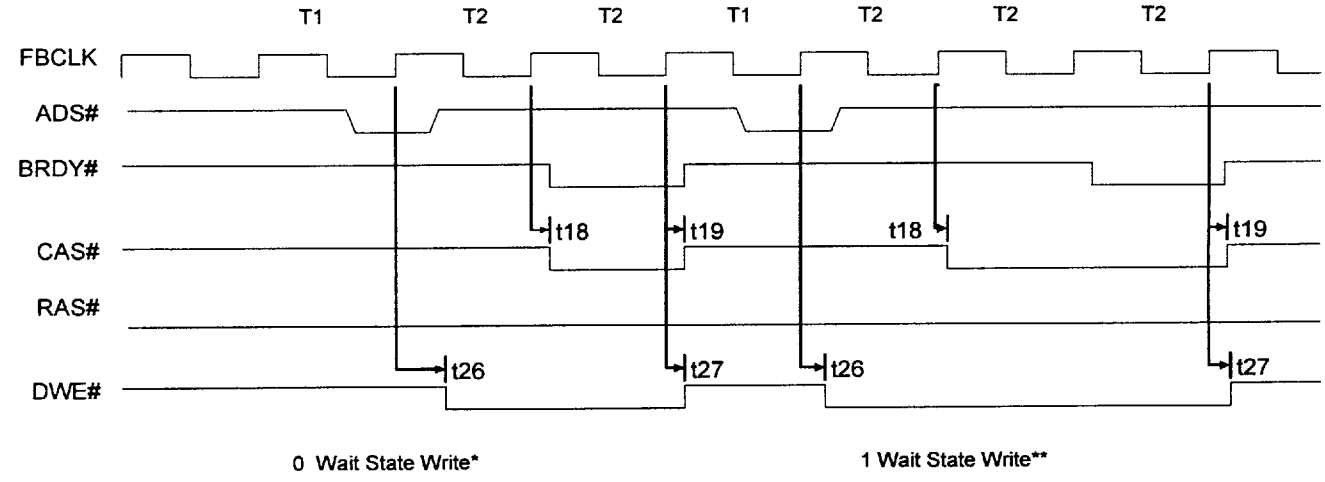


Figure 6-23 DRAM Write Wait State (without L2 Cache Support)



* 0 Wait State cycle takes three clocks to be completed.
** 1 Wait State cycle takes four clocks to be completed.

Figure 6-24 Single Local Bus Cycle

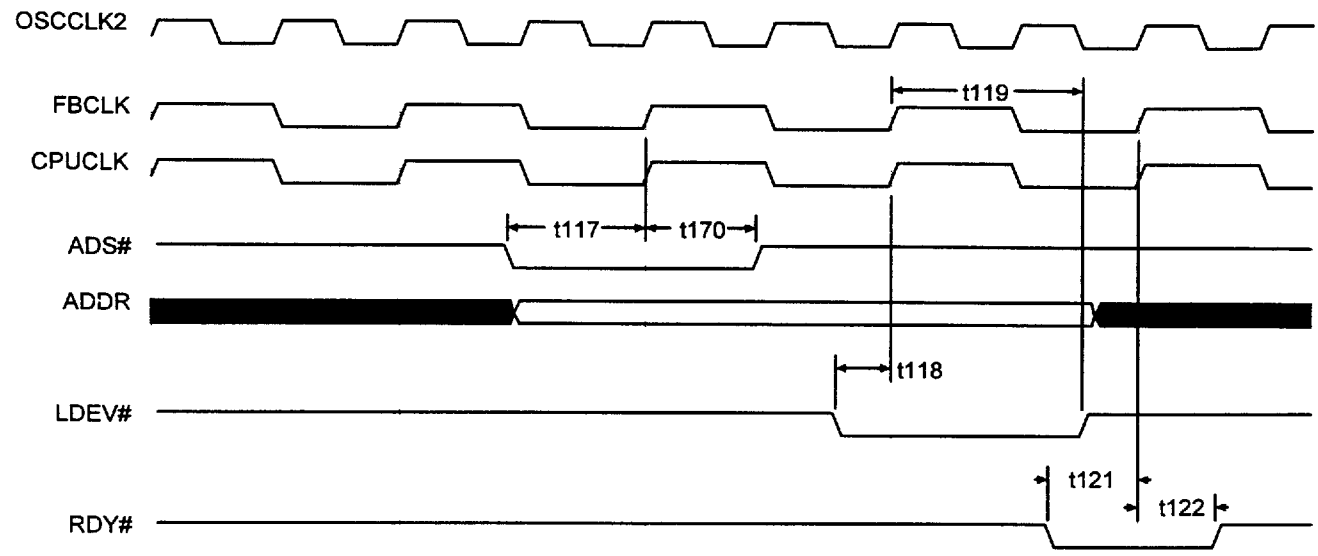


Figure 6-25 Reset Timing

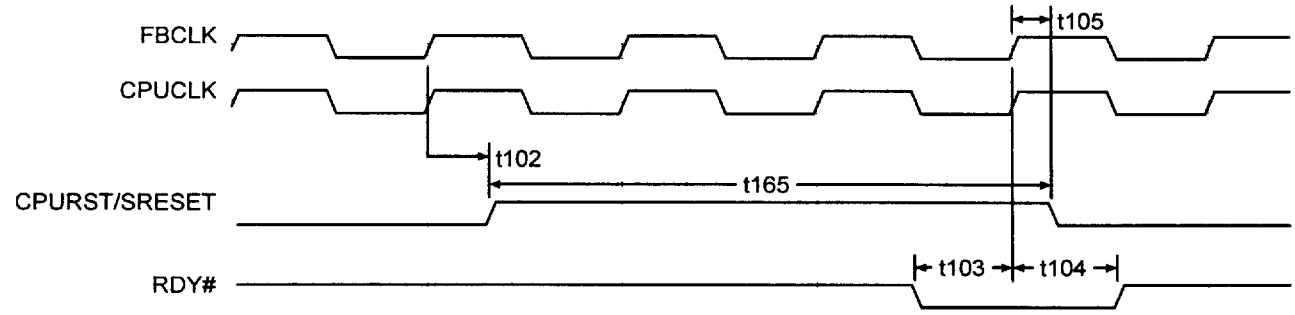


Figure 6-26 Low Word At Bus Memory Read To High Word Local Bus With SDENL#, SDENH#, and SDIR Active

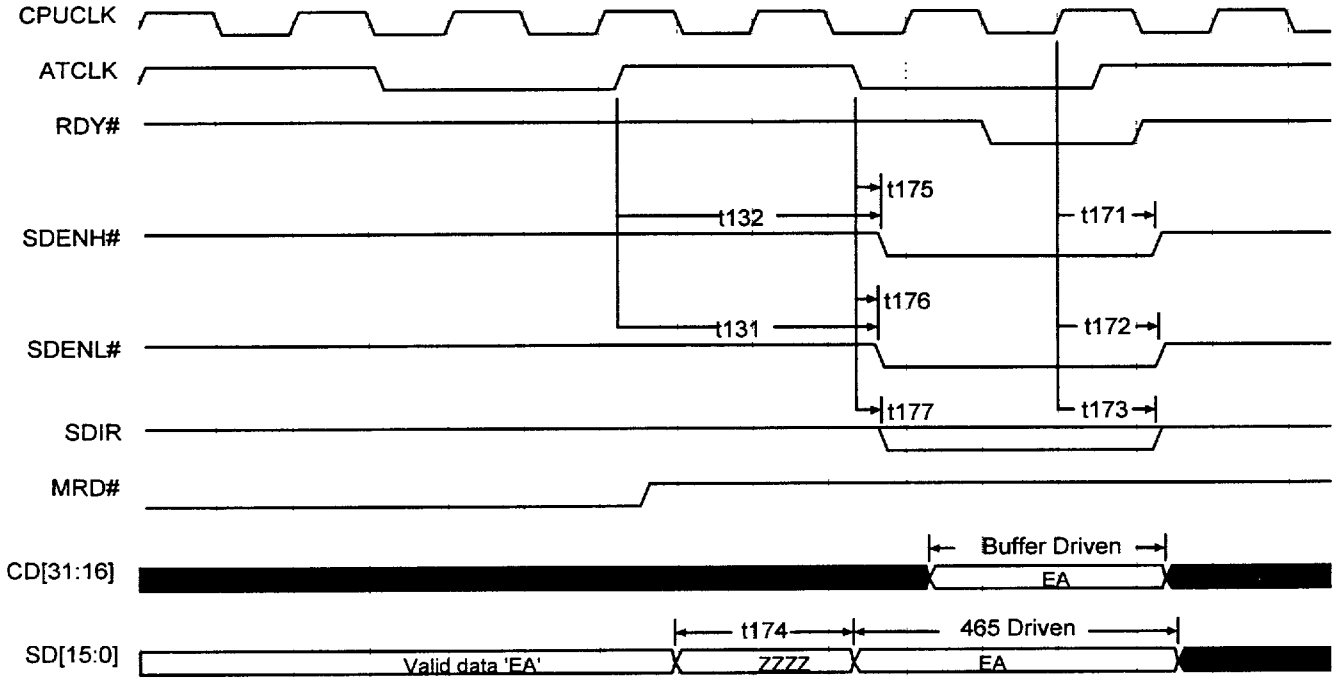


Figure 6-27 Write PPWR[3:0] With '1111'

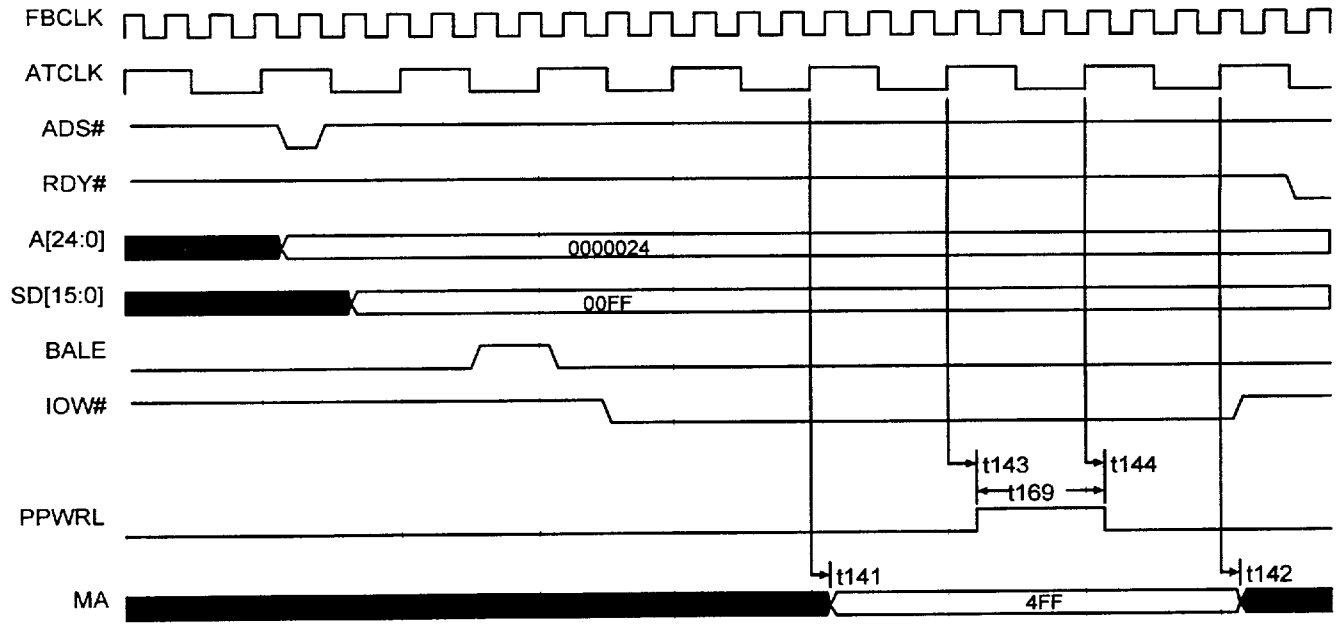


Figure 6-28 PIO Timing Example (PIO2)

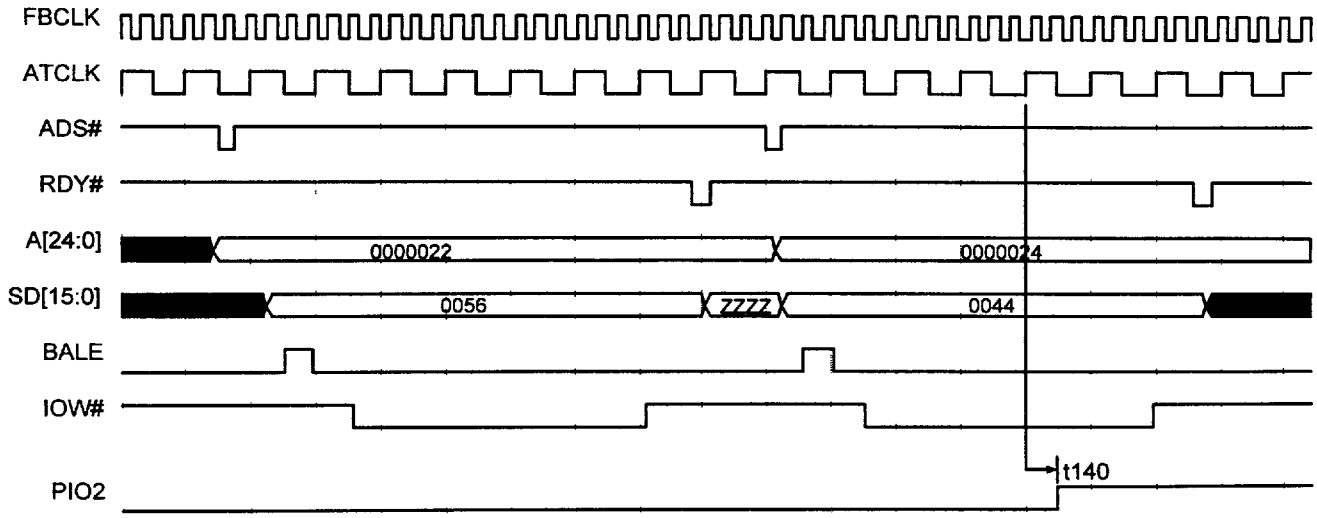
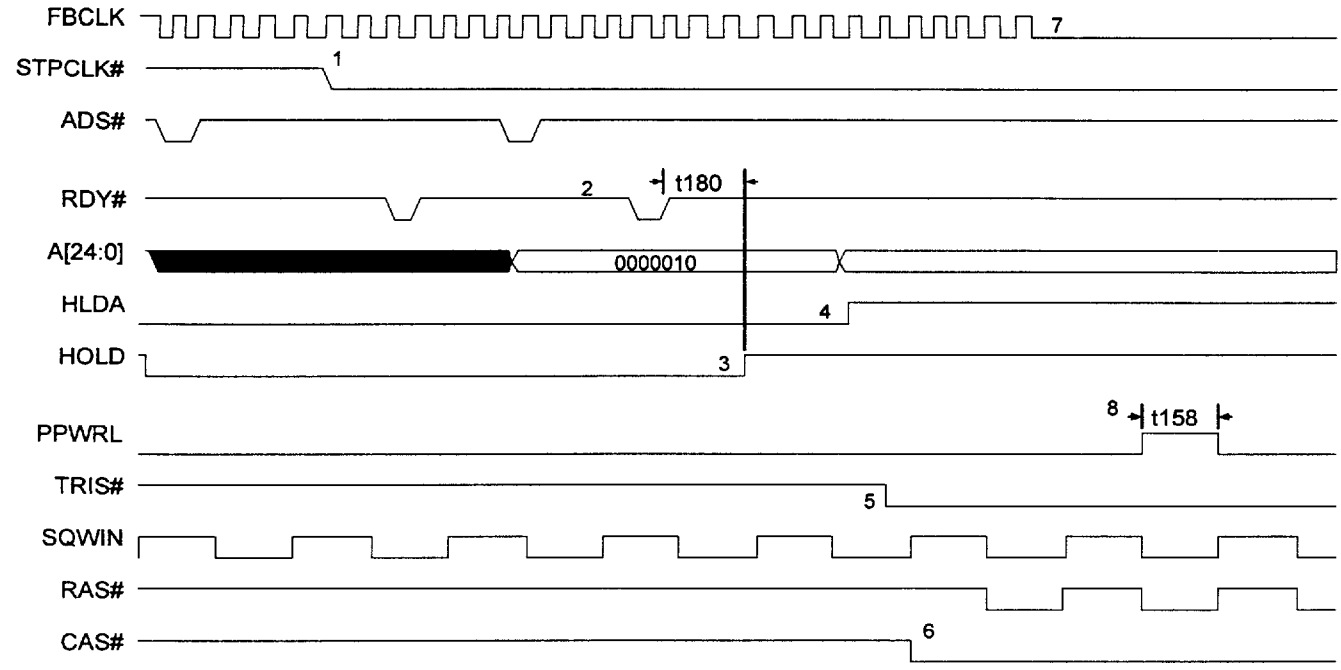


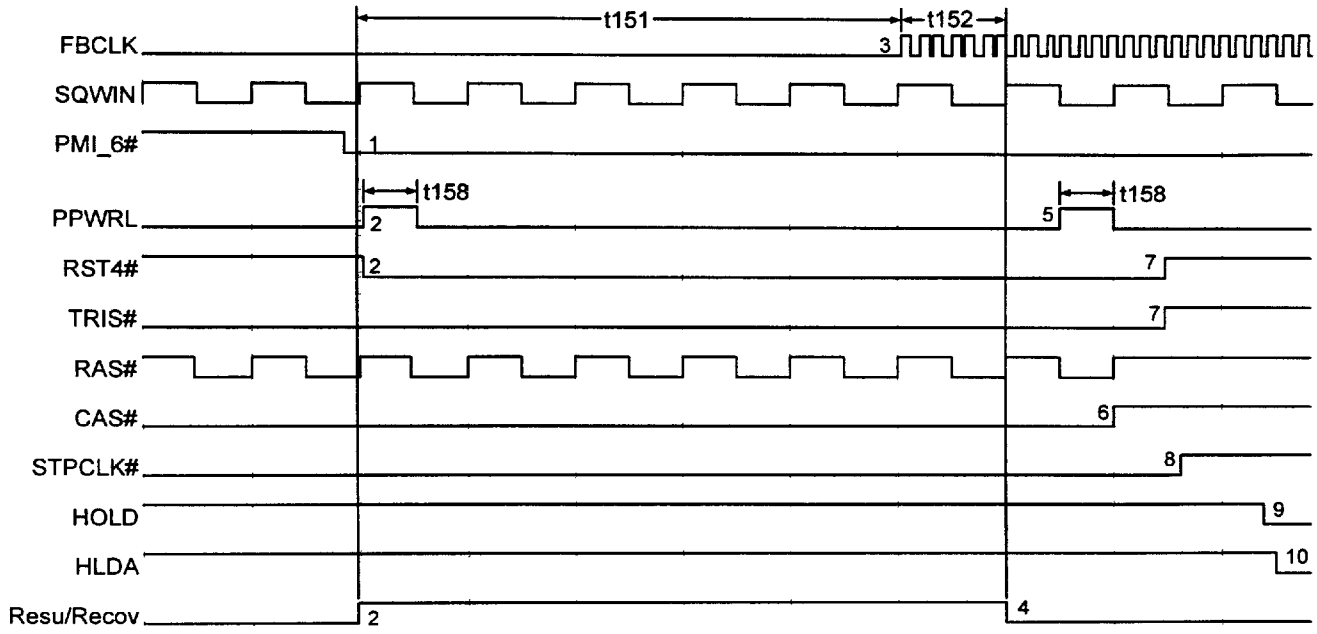
Figure 6-29 Suspend Sequence After Writing '1' To 50h[0]



NOTE The timing in this waveform is not proportional. The diagram is used to show the sequence of events 1 through 8 only.

1. 465MV asserts STPCLK# signal to CPU.
2. CPU responds with stop grant cycle.
3. One ATCLK after the RDY# signal is asserted, 465MV asserts HOLD to CPU.
4. CPU responds with HLDA.
5. 465MV asserts TRIS# and drives its output pins according to suspend state.
6. 465MV asserts suspend refresh within 1 1/2 32KHz clock (~45us). Normal refresh is still enabled within this period (not shown in the diagram).
7. 465MV stops the following clocks in low state: CPUCLK, FBCLK, ATCLK, KBCLK, KBCLK2
8. One 32KHz clock after suspend refresh, 465MV toggles PPWR0-1 to turn off clock generator.

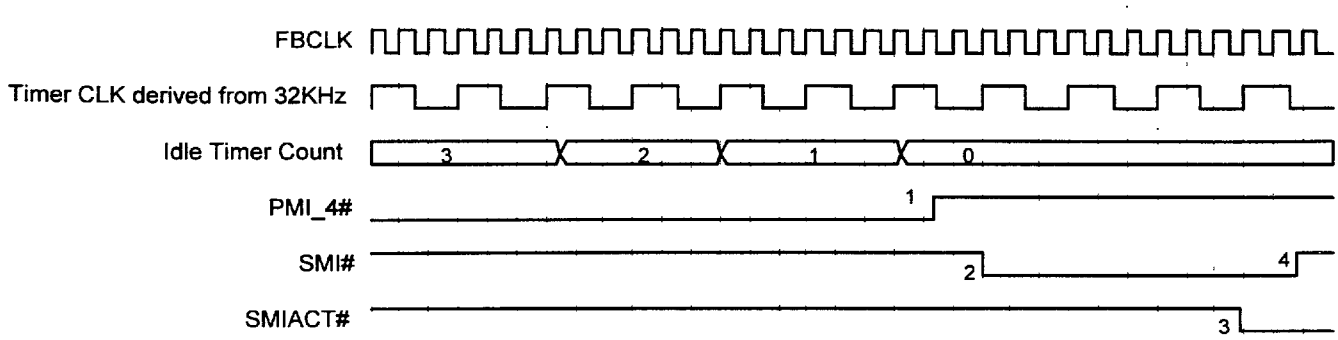
Figure 6-30 Resume Sequence



NOTE The timing in this waveform is not proportional. The diagram is used to show the sequence of events 1 through 10 only.

1. Resume source sets PMI6# active. (No SMI generated if already in SMM.)
2. PPWR0-1 toggle to enable clock generator. PPWR10 also set to low for resume reset. RST4# asserted for resume reset (also EPMI2, if enabled).
3. After 7/8 of resume recovery time, 465MV resumes CPUCLK, FBCLK, ATCLK.
4. Resume recovery time expires.
5. 1/2 32KHz clock later, 465MV toggles PPWR10 to end resume reset.
6. Suspend refresh ended, normal refresh started.
7. 465MV de-asserts TRIS# and drives its output pins back to normal state, and RST4#/EPMI2 de-asserted to end the resume reset.
8. 465MV de-asserts STPCLK# to the CPU.
9. 465MV de-asserts the HOLD signal.
10. CPU de-asserts the HLDA signal.

Figure 6-31 Timer Timeout and SMI Generation Sequence

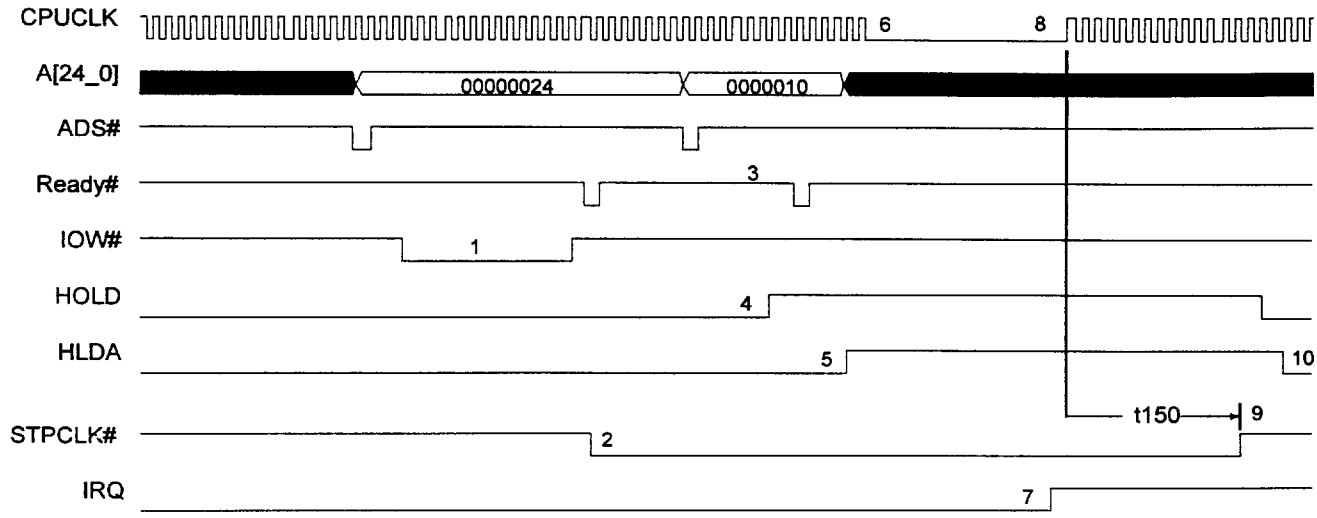


NOTE The timing in this waveform is not proportional. The diagram is used to show the sequence of events 1 through 4 only.

NOTE Idle Timer Count is not visible. It is shown here for illustrative purposes only. Other timers also behave in this manner.

1. Idle timer times out.
2. 465MV activates SMI# signal to CPU.
3. CPU returns SMIACT#.
4. 465MV de-asserts SMI# after SMIACT# goes active.

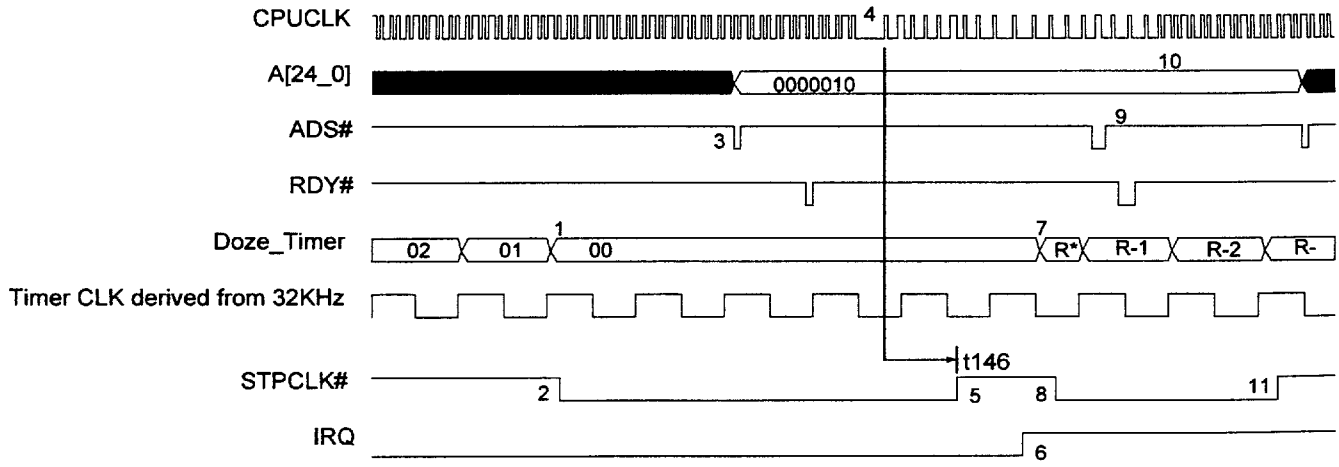
Figure 6-32 APM Stop Clock Sequence



NOTE The timing in this waveform is not proportional. The diagram is used to show the sequence of events 1 through 10 only.

1. Write 50h[3]=1 to enable 465MV to start stop-clock cycle.
2. 465MV drives STPCLK# low.
3. CPU stop grant cycle.
4. 465MV asserts HOLD.
5. CPU responds with HLDA.
6. 465MV stops CPUCLK or changes the clock according to bits 41h[4:2].
7. IRQ wakes up 465MV from stop clock and triggers resume speed sequence.
8. Both FBCLK and CPUCLK resume back to full speed.
9. After STPCLK# is de-asserted, 465MV releases HOLD.
10. CPU releases HLDA.

Figure 6-33 Doze Sequence



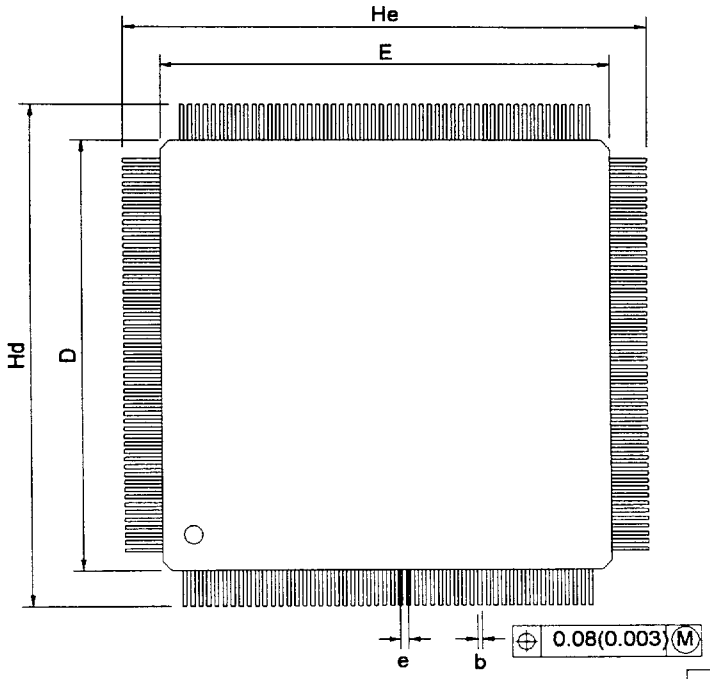
NOTE The timing in this waveform is not proportional. The diagram is used to show the sequence of events 1 through 11 only.

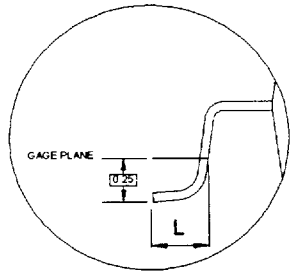
1. Doze timer times out
2. 465MV asserts STPCLK# signal.
3. CPU responds with stop grant cycle.
4. 465MV changes the FBCLK, CPUCLK according to bits 41h[4:2].
5. Stop clock delay according to register B0h.
6. IRQ line goes active to trigger the system back to normal speed.
7. Doze timer reload.
8. 465MV asserts stop clock signal.
9. CPU responds with stop grant cycle.
10. 465MV changes the clocks back to normal speed.
11. 465MV holds the STPCLK# signal active for the time selected in register B0h.

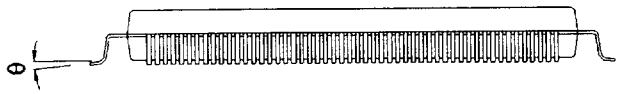
* R = Reload Value

7.0 Mechanical Package

Figure 7-1 208 Pin Plastic Quad Flat Pack (PQFP)







SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A_1	0.05	0.25	0.50	0.002	0.010	0.020
A_2	3.17	3.32	3.47	0.125	0.131	0.137
b	0.10	0.20	0.30	0.004	0.008	0.012
c	0.10	0.15	0.20	0.004	0.006	0.008
D	27.90	28.00	28.10	1.098	1.102	1.106
E	27.90	28.00	28.10	1.098	1.102	1.106
e		0.50			0.020	
Hd	30.35	30.60	30.85	1.195	1.205	1.215
He	30.35	30.60	30.85	1.195	1.205	1.215
L	0.50	0.60	0.75	0.020	0.024	0.030
L_1		1.30			0.051	
Y			0.08			0.003
θ	0		7	0		7

Dwg. No..	AS208PQFP-001		
Dwg. Rev.:	A1	Unit:	MM / INCH

Appendix A. Accessing the BBS

The OPTi BBS offers a wide range of useful files and utilities to our customers, from Evaluation PCB Schematics to HPGL/ PostScript format Databooks that you can copy directly to your laser printer. The only requirements for accessing and using the BBS is a modem and an honest response to our questionnaire.

A.1 Paging the SYSOP

Currently, Paging the SYSOP is not a valid choice for the OPTi BBS. Once a full-time SYSOP is created, then there will be hours available for paging the SYSOP and getting immediate help.

For now, you must send [C] Comments to the SYSOP with any questions or problems you are experiencing. They will be answered promptly.

NOTE Each conference has its own Co-SYSOP (the application engineer responsible for that product line), so specific conference questions can be addressed that way, but general, BBS-wide, questions should be sent to the SYSOP from the [0] - Private E-Mail conference.

A.2 System Requirements

The OPTi BBS will support any PC modem up to 14,400 baud, with 8 bits, no parity, and 1 stop bit protocol. The baud rate, handshaking, and system type will automatically be detected by the OPTi BBS.

A.3 Calling In/Hours of Operation

The OPTi BBS phone number is (408) 980-9774. The BBS is on-line 24 hours a day, seven days a week. Currently there is only one line, but as traffic requires additional lines will be installed.

A.4 Logging On for the First Time

To log on to the BBS for the first time,

1. Call (408) 980-9774 with your modem.
2. Enter your first name.
3. Enter your last name.
4. Verify that you have typed your name correctly.
5. Select a password (write it down).
6. Reenter the password to verify spelling.
7. You must then answer the questionnaire that follows.

After you have answered the questionnaire, you are given Customer rights. To change your profile (security level, password, etc.), you must send a [C]omment to the SYSOP explaining why.

After you have logged on for the first time, each subsequent log on will bypass the questionnaire and put you directly at the bulletin request prompt. As bulletins will be added on a regular basis in the future, it is recommended that you read the new bulletins on a regular basis.

A.5 Log On Rules and Regulations

- As a FULLUSER you can download from any conference.
- You will be limited to 45 minutes per day of access time (note that once a download has started, it will finish, even if the daily time limit is exceeded). If you have not entered any keystrokes after 5 minutes, you will automatically be logged off.
- You can upload to the Customer Upload Conference[1] only. This area is used for our customers/contacts to send data to OPTi. You will not be able to download any files from this area.

A.6 Using the BBS

This section will describe how to use the BBS on a daily basis.

The BBS is divided into Conferences that are specific to a product (for example, the Viper Desktop Chipset), or an application group (for example, the Field Application Conference is used by OPTi Field Application Engineers to send data to their contacts in the field). As a general rule, the files in the application specific areas will be for specific application and may contain a password. If a file is password protected, and you know you need that file, you must contact your OPTi sales representative for the password.

The files in the Product Conferences are released data that can be used for evaluating the OPTi product line.

To access a feature of the BBS, you should type the letter in brackets that precedes each menu item. This document places the appropriate letter in brackets whenever you are told to access a feature.

1. See Section A.6.5 for more information on uploading.

A.6.1 Reading Bulletins

The OPTi BBS will present you with a set of bulletins each time you log on that are global bulletins applying to OPTi in general. In addition to these, each Product Conference will have its own set of bulletins that apply to that product. These bulletins will announce new product information, documentation updates, and bug fixes and product alerts.

It is recommended that you read any new bulletins on a regular basis to keep up to date on the OPTi product line.

A.6.2 Sending/Receiving Messages

The Message Menu can be used to send and receive messages from OPTi employees, or other BBS users. The Message menu can also be used to attach files for the receiver to download after they read the message. This method will be used often to send customer specific files to OPTi customers.

Messages to the SYSOP depend upon the conference you are in. Each Product Conference sends Comments and Messages to the SYSOP to the Application Engineer responsible for that conference.

A.6.3 Finding Information

To find information on the OPTi BBS, you must use the [J] Join a Conference option and then list all of the conferences available. They are arranged by product number and name.

Once you are in the correct conference, you should read all applicable bulletins and messages. Then you can [L] List all the files that are available from the File Menu.

A.6.4 Downloading Files From OPTi

The easiest way to download files from OPTi, is to [L] List the files from the File Menu, the [M] Mark and files you want from the list. After you have marked all the files you need, you can [D] Download all the marked files and then logoff automatically.

A.6.5 Uploading Files To OPTi

There are two ways to upload a file to OPTi. The first is similar to the download option. You should [J] Join the Customer Upload Conference (this is the only conference that allows uploads from users) and [U] Upload the file to this conference.

If you are sending the file to a specific person, you should use the Message Menu to [E] Enter a new message to that person and then [A] Attach the file to the message. This way, the person receiving the message can download the file to his or her system without leaving behind a file that will not be used by anyone else on the BBS.

A.6.6 Logging Off

Once you have completed your visit to the OPTi BBS, you must say [G] Goodbye.

A.6.7 Logging Back on Again

To log back on to the BBS,

1. Call (408) 980-9774 with your modem.
2. Enter your first name.
3. Enter your last name.
4. Verify that you have typed your name correctly.
5. Enter your password.

You will not have to answer the questionnaire after the initial log-on. You will also be in the conference you were in when you last logged-on.

A.7 The Menus

There are four major menus that OPTi customers will use, the Main Menu, the File Menu, the Bulletin Menu and the Message Menu.

NOTE The following menus are for the Customer Profile (FULLUSER) only, if your user profile has been changed, you may see slightly different menus.

Figure A-1 The Main Menu

```
MAIN MENU:
[ J ] Join a conference              [ F ] File menu
[ M ] Message menu                   [ B ] Bulletin menu
[ C ] Comments to the sysop          [ U ] Userlog list
[ Y ] Your settings                  [ G ] Goodbye & logoff

Conf: "[0] - Private E-Mail", time on 0, with 45 remaining.
MAIN MENU: [J F M B C P U Y G] ?
```

Figure A-2 The Bulletin Menu

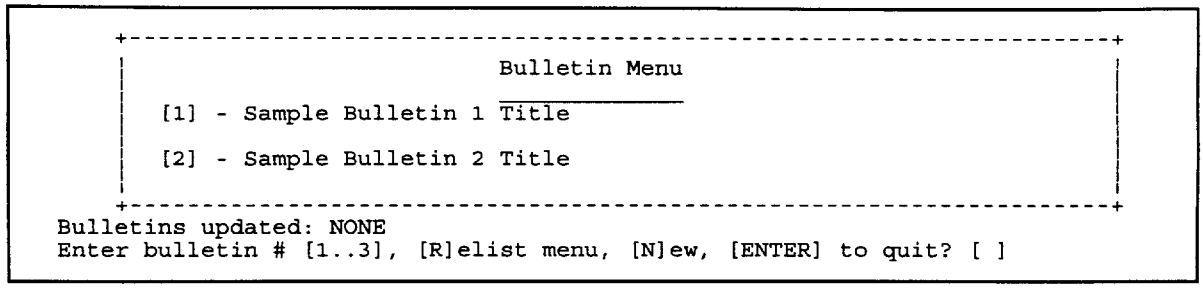


Figure A-3 The File Menu

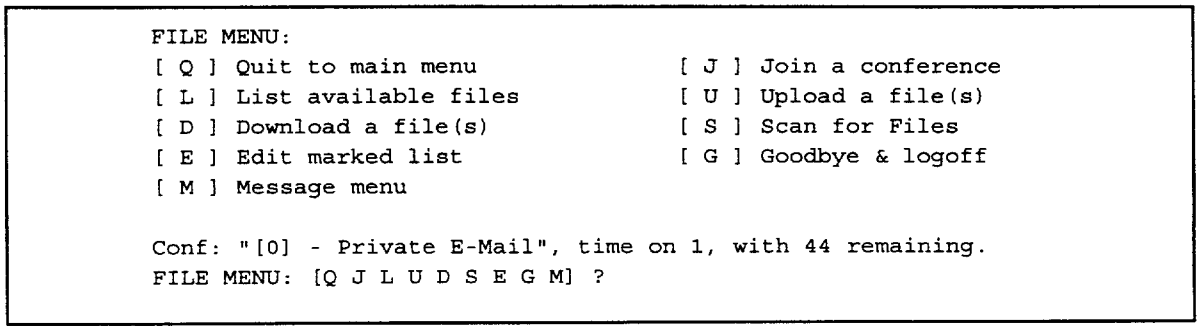
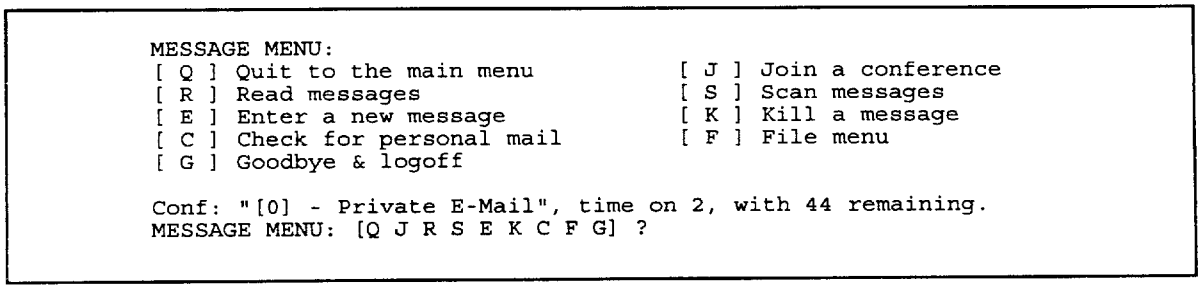


Figure A-4 The Message Menu



A.7.1 Menu Selections

- [B] Bulletin MenuMenu(s): main
  Access the Bulletin Menu.
- [C] Check for personal mailMenu(s): message
  See if you have any mail.
- [C] Comments to the sysopMenu(s): main
  Leave a private comment for the SYSOP.
- [D] Download a file(s)Menu(s): file
  Download a file from the BBS to your computer. If you have marked files it will display these files. If you have not marked any files, it will ask you for a file name. The file must be present in the current conference for you to be able to enter its name.
- [E] Edit Marked ListMenu(s): file
  Change the entries that you have selected as Marked for downloading.
- [E] Enter a new messageMenu(s): message
  Send a new message to someone on the BBS.
- [F] File MenuMenu(s): main, message
  Access the File Menu.
- [G] Goodbye and logoffMenu(s): main, message, file
  Logoff the system.
- [J] Join a ConferenceMenu(s): main, message, file
  Change conferences (product areas).
- [K] Kill a messageMenu(s): message

Delete a message.

- [L] List available filesMenu(s): file
  List the files in the current conference. Note that most conferences have sub-categories of files (Schematics, JOB, etc.) that you will be asked for (or you can press enter the list all of the categories).
- [M] Message MenuMenu(s): main, file
  Access the Message Menu.
- [Q] Quit to Main MenuMenu(s): message, file
  Leave current menu and return to the Main Menu.
- [R] Read MessagesMenu(s): message
  Read messages in the current conference or all conferences.
- [S] Scan for FilesMenu(s): file
  Scan for particular files (by name, or extension, etc.).
- [S] Scan messagesMenu(s): message
  Search for message by specific qualifier (date, sender etc.).
- [U] Upload a file(s)Menu(s): file
  Send a file from your computer to OPTi. This can only be done in the Customer Upload Conference.
- [U] Userlog ListMenu(s): main
  Lists the user database, in order of logon. This is useful if you are sending a message and are looking for the spelling of a persons name.
- [Y] Your settingsMenu(s): main
  Show you settings and allow you to make changes. These include password, name, address, etc.

Appendix B. Incompatibilities with the 82C463MV

The 82C465MV is intended to be as fully compatible as possible with the 82C463MV. However, certain architectural and programming features should be noted.

B.1 Power Plane Changes

The pins with alternative functions DACKMUX0-2 and DACK2# are 3.3V outputs on the 82C465MV; these were 5V outputs on the 82C463MV. Some existing 82C463MV designs could exhibit a slight increase in power consumption depending on the logic family used to interface to these signals. If the 82C465MV is used in a new design, these signals are relocated to a 5V plane and do not affect current draw.

B.2 Read Cycle Efficiency

On slower systems, the 82C463MV memory controller could be programmed to a 2-1-1-1 or 3-1-1-1 cycle due to the fact that the chipset used a 2X input clock. The 82C465MV uses only a 1X input clock to run its memory controller. Therefore, the fastest memory cycle possible on the 82C465MV is 3-2-2-2. In most practical applications, the speed difference is not detectable because most operation occurs from the CPU cache. Since burst read cycles are only used to refill the cache and this process takes place concurrently with internal processing, there is no substantial performance loss.

However, a perceived performance reduction can occur if an 82C463MV-compatible BIOS is used. The DRAM cycle controller bits changed meaning between the 82C463MV and the 82C465MV and settings that used to allow 3-1-1-1 and 3-2-2-2 operation now select 4-3-3-3 operation. Register 35h should be modified as soon as possible after boot, either through a BIOS modification or a simple executable file.

B.3 ADS# Sampling

On very fast systems, predictive sampling of ADS# to determine cycle type for better efficiency is not always effective. The safest way to determine cycle type is to latch the M/IO#, W/R#, and D/C# status on the rising edge of ADS#. The 82C463MV could not run above 33MHz, so this sampling was never an issue. But since the 82C465MV can run as high as 50MHz, it must default to the safe mode for latching status. A significant performance improvement can be made to slower systems, therefore, if the 82C463MV-compatible mode of sampling is enabled through setting bit D1h[6] = 1.

B.4 Removal of Sequencer

The 82C463MV contained a Sequencer that could perform limited power management functions on its own without CPU intervention. Because the vast majority of CPUs now available provide SMI control, the Sequencer feature has been removed.

B.5 Default Refresh Rate Change

Since the Sequencer has been removed, the global Sequencer enable bit 67h[5] has been redefined to form part of the refresh rate selection bits. Most 82C463MV-based BIOSes enabled this bit. With this setting, the default refresh rate on the 82C465MV (both active mode and suspend mode) will be every 31us instead of the recommended 15us. While most modern DRAM can handle slower refresh rates, the BIOS should be modified for the correct rate if necessary.

B.6 I/O Blocking Default Change

Bit DBh[7] defaults to 0 and disables I/O blocking on Next Access. It should be set to 1 to be compatible with 82C463MV software. The 82C463MV part had no bit to control this feature, which was always set to "block".

B.7 Suspend Mode DACKMUX Parking

Bits D5h[7:6] default to a state that parks the DACKMUX lines differently than the 82C463MV part, such that DACK2# sits low instead of DACK4#. These bits should be reset if needed (if DACK2# low will cause problems).

Appendix C. Board-Level Test Mode

The 82C465MV part can be forced to tri-state all of its outputs at any time for board-level testing. Once this mode has been commanded, all chip operating and status information becomes invalid. Therefore, the chip must be powered down and then restarted after this feature is used.

To enter test mode:

- Set pins 85, 86, and 139 low.
- Set pins 45, 87, 128, and 140 high.

Test mode is enabled by a decode of combinatorial logic. Therefore, the order in which these pins are brought high or low is unimportant.

To clear test mode, power down the system and clear the high/low settings used to enable test mode.

Appendix D. Compact ISA Specification

This document describes a new OPTi interface that will be used to interface the 82C852 PCMCIA Controller to OPTi system controller chipsets. This interface may also be used to interface OPTi peripheral products in the future. The interface is OPTi-proprietary, and may be licensed to others in the future.

D.1 Compact ISA Overview

The Compact ISA interface coexists with the standard ISA interface. Chips that support the Compact ISA interface enjoy a reduced ISA pin count because address signals and command information are strobed in on the SD[15:0] bus. ISA pins eliminated are:

- SA[23:0] (24 pins)
- IORD#, IOWR#, MRD#, MWR#, SMRD#, SMWR#, SBHE#, NOWS#, AEN, IO16#, M16# (11 pins)
- IRQ3, 4, 5, 6, 7, 10, 11, 12, 14, 15; DRQ/DACK#0, 1, 2, 3, 5, 6, 7, and TC (25 pins)

Compact ISA defines only two new signals, CMD# and SEL#/ATB#, for a total requirement of 22 pins. The pin count reduction over standard ISA is 58 pins. Compact ISA performance is comparable with that of 16-bit ISA bus peripheral devices. Moreover, Compact ISA does **not** interfere with standard ISA operations. The complete signal set of Compact ISA, referred to in the descriptions as CISA, is shown below.

Table D-1 Compact ISA (CISA) Interface Signals

Name	Type*	Description
MAD[15:0]	I/O	Multiplexed bus used to transfer address, command, data, IRQ, DRQ, DACK information.
ATCLK	I	Standard AT clock. CISA device uses rising edge to clock in the first (address) phase.
ALE	I	Standard AT address latch enable. CISA peripheral device uses rising edge of ALE to latch the second (address and command) phase. CISA host uses falling edge of ALE to latch CMD# from peripheral device.
CMD#	I	Command indication. Common to host and all devices on the CISA bus. The CISA host asserts CMD# during the data phase of the cycle to time the standard ISA command (IORD#/WR#, MRD#/WR#), and also asserts CMD# to acknowledge SEL#ATB#.
SEL#/ATB# (also CLKRUN#)	O tristate	Device selected / AT bus backoff request. Common to all peripheral devices on the CISA bus. When ALE is high, the CISA device asserts SEL# to indicate to the host that it is claiming the cycle. When ALE is low, the CISA device drives this signal to indicate that it has an interrupt and/or DMA request to make; the host acknowledges by asserting CMD#. After the host has preset the CISA device in a Stop Clock mode, the device can assert this signal asynchronously to restart the clock.
IOCHRDY	O tristate	Standard AT cycle extension request signal during memory and I/O cycles.
RSTDRV	I	Standard AT-bus reset signal.

* (Peripheral side)

D.2 Compact ISA Cycle Definition

The MAD[15:0] lines contain different information for each phase of the bus cycle. The use of these lines varies according to whether a memory cycle or an I/O cycle is being run. Certain cycle definition bits are common to all cycles, as shown in Table D-2.

Table D-2 Common MAD Bit Usage

Signal	Phase 1	Phase 2
MAD0	M/IO# indication bit; used to determine the cycle type.	W/R# indication bit
MAD1	I/D# indication bit. It is always 0 if M/IO#=1, and selects between I/O and DMA cycles if M/IO#=0.	SBHE# indication bit
MAD2	Usage varies.	ISA# timing indication bit; described in the "Performance Control" section of this document.

Retained Values
Entries marked "same" retain the same value as in the previous phase, in order to reduce transitions where possible. However, the CISA peripheral device decode logic must **not** assume that these values will be stable. The bits may be reassigned in the future.

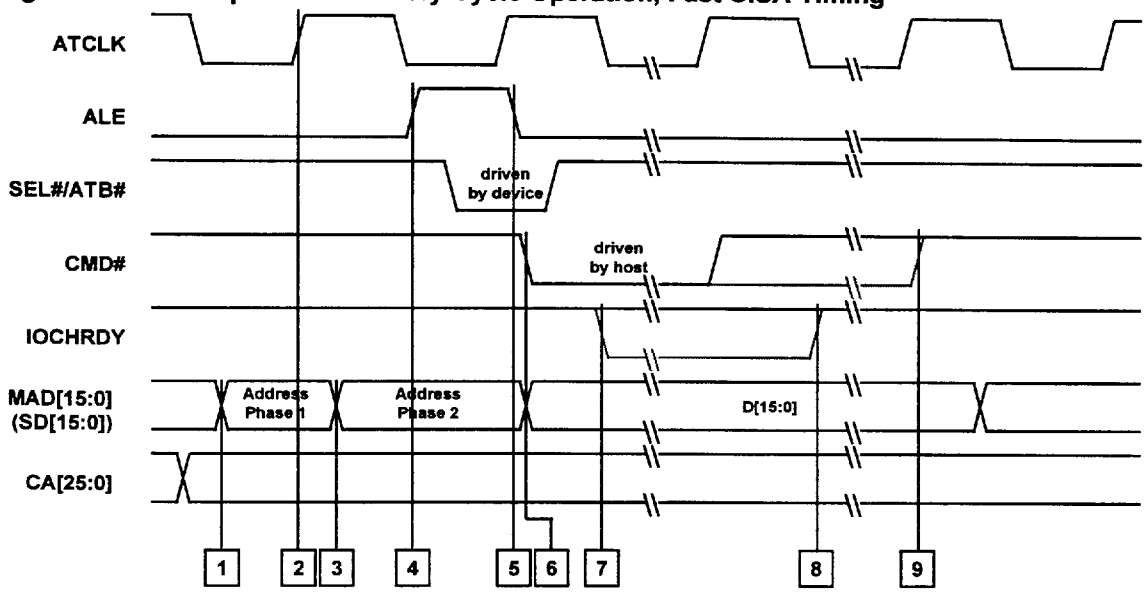
D.2.1 Memory Cycle

The MAD[15:0] bit meanings for each phase of a memory cycle are shown below. The M/IO# bit is always 1 for memory cycles.

Phase	MAD 15	MAD 14	MAD 13	MAD 12	MAD 11	MAD 10	MAD 9	MAD 8	MAD 7	MAD 6	MAD 5	MAD 4	MAD 3	MAD 2	MAD 1	MAD 0
1	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SA15	SA14	SA13	SA12	SA11	SA10	I/D# =0	M/IO# =1
2	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	same	same	same	ISA#	SBHE#	W/R#
3	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

The general structure of Compact ISA memory cycles is shown in Figure D-1 and Figure D-2.

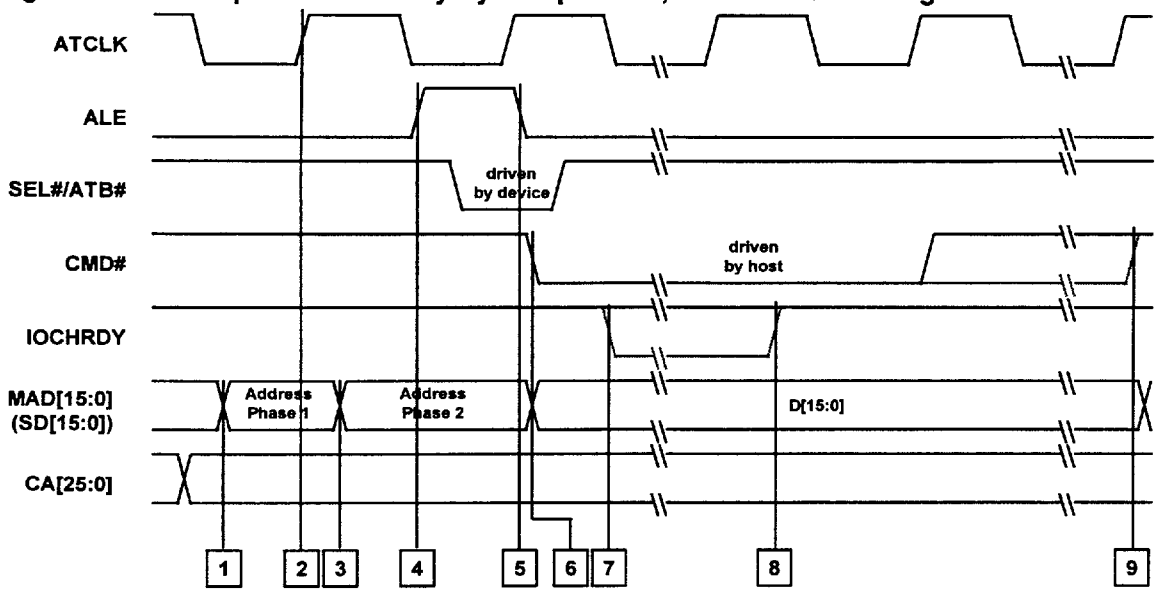
Figure D-1 Compact ISA Memory Cycle Operation, Fast CISA Timing*



1. CISA host gets address from the CPU address lines and byte enable lines. The host then drives out A[23:10] + M/IO# on MAD[15:0] with M/IO# high (memory).
2. CISA peripheral device latches address and M/IO# on the rising edge of ATCLK and decodes the information.
3. Host drives out remaining address + Command on MAD[15:0].
4. Host asserts ALE. If cycle belongs to CISA peripheral device, it asserts SEL# and latches the address and command from MAD[15:0] on the rising edge of ALE. Decice latches ISA#=1 at this time.
5. Host and other CISA devices recognize the SEL# function of SEL#/ATB# by seeing ALE high when sampling SEL#/ ATB# low on the rising edge of ATCLK. Host de-asserts ALE and stops driving address on this rising ATCLK edge.
6. For reads, the host tristates the MAD[15:0] buffers. For writes, it drives the write data onto MAD[15:0]. Host asserts CMD# synchronous to the rising edge of ATCLK and can optionally inhibit its MRD#/MWR# lines.
7. Cycle is 0 wait states as indicated by ISA#=1. CISA peripheral device can bring IOCHRDY low asynchronously after CDM# goes active to extend the cycle.
8. Device brings IOCHRDY high synchronous to the falling edge of ATCLK to allow cycle completion.
9. Host de-asserts CMD# on the same rising edge where it samples IOCHRDY high.

* Cycle optionally extended by IOCHRDY shown in gray.

Figure D-2 Compact ISA Memory Cycle Operation, Standard ISA Timing*



1. CISA host gets address from teh CPU address lines and byte enbable lines. The host then drives out A]23:10] + M/IO# on MAD[15:0] with M/IO# high (memory).
2. CISA peripheral device latches address and M/IO# on the rising edge of ATCLK and decodes the information.
3. Host drives out remaining address + Command on MAD[15:0].
4. Host asserts ALE. If cycle belongs to CISA peripheral device, it asserts SEL# and latches the address and command from MAD[15:0] on the rising edge of ALE. Device latches ISA#=0 at this time.
5. Host and other CISA devices recognize the SEL# function of SEL#/ATB# by seeing ALE high when sampling SEL#/ ATB# low on the rising edge of ATCLK. Host de-asserts ALE and stops driving address on this rising ATCLK edge.
6. For reads, the host tristates the MAD[15:0] buffers. For writes, it drives the write data onto MAD[15:0]. Host asserts CMD# synchronous to the rising edge of ATCLK and can optionally inhibit its MRD#/MWR# lines.
7. Cycle is not zero wait states, as indicated by ISA#=0. CISA peripheral device can bring IOCHRDY low asynchronously after CDM# goes active to extend the cycle further.
8. Device brings IOCHRDY high asynchronously to allow cycle completion.
9. Host de-asserts CMD# on the next rising edge of ATCLK after the rising edge ATCLK edge on which it samples IOCHRDY high.

D.2.2 I/O Cycle

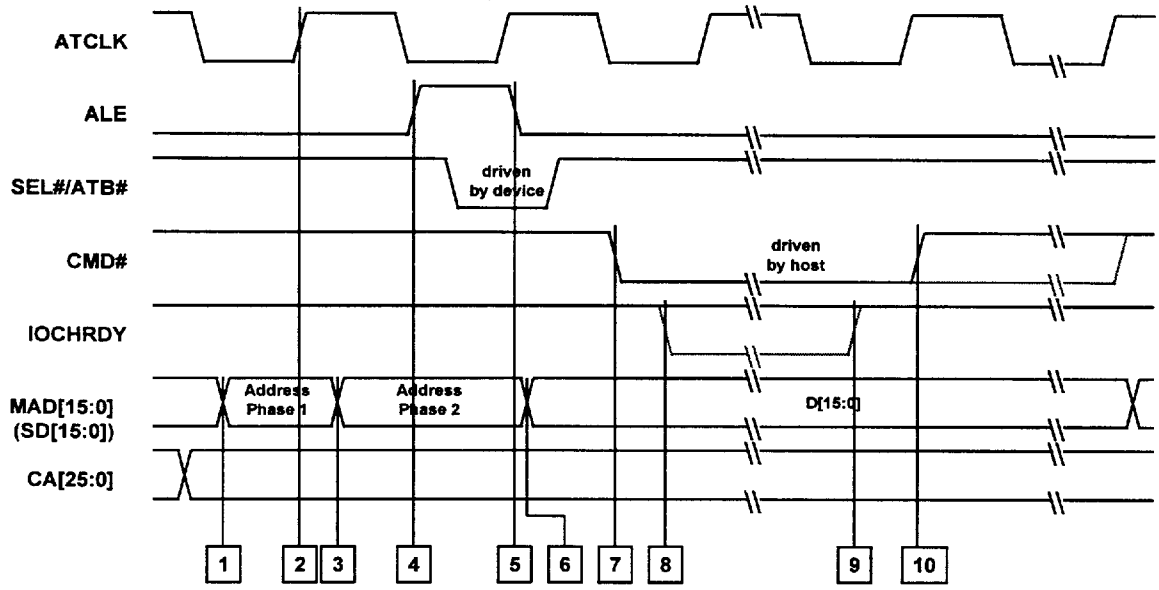
The MAD[15:0] bit meanings for each phase of an I/O cycle are shown below. The M/IO# bit is always 0, and the I/D# bit is always 1, for an I/O cycle.

Phase	MAD 15	MAD 14	MAD 13	MAD 12	MAD 11	MAD 10	MAD 9	MAD 8	MAD 7	MAD 6	MAD 5	MAD 4	MAD 3	MAD 2	MAD 1	MAD 0
1	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA15	SA14	SA13	SA12	SA11	SA10	I/D# =1	M/IO# =0
2	same	same	same	same	same	same	same	same	SA1	SA0	same	same	same	ISA# =0	SBHE#	W/R#
3	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

* Cycle optionally extended by IOCHRDY shown in gray.

The general structure of Compact ISA I/O cycles is shown in Figure D-3.

Figure D-3 Compact ISA I/O Cycle Operation*



1. CISA host gets address from the CPU address lines and byte enbable lines. The host then drives out A[15:2] + I/D#=1 + M/IO#=0 (I/O cycle).
2. CISA peripheral device latches address and M/IO# on the rising edge of ATCLK and decodes the information.
3. Host drives out remaining address + Command on MAD[15:0].
4. Host asserts ALE. If cycle belongs to CISA peripheral device, it asserts SEL# and latches the address and command from MAD[15:0] on the rising edge of ALE.
5. Host and other CISA devices recognize the SEL# function of SEL#/ATB# by seeing ALE high when sampling SEL#/ ATB# low on the rising edge of ATCLK. Host de-asserts ALE and stops driving address on this rising ATCLK edge.
6. For reads, the host tristates the MAD[15:0] buffers. For writes, it drives the write data onto MAD[15:0].
7. Host asserts CMD# synchronous to the falling edge of ATCLK to run the command and can optionally inhibit its IOR#/ IOW# lines.
8. Cycle is never zero wait state. CISA peripheral device can bring IOCHRDY low asynchronously after CDM# goes active, using standard ISA setup timing, to extend the cycle further. Note that if CISA periperhal device provides a bridge to another device (a PCMCIA slot, for example), the device on the secondary bus must be able to return IOCHRDY soon enough to meet setup timing on the CISA interface.
9. Device brings IOCHRDY high asynchronously to allow cycle completion.
10. Host de-asserts CMD# on the next falling edge of ATCLK after the rising edge ATCLK edge on which it samples IOCHRDY high.

D.2.3 DMA on the CISA/ISA Bus

DMA operations are handled very specifically for CISA peripheral devices. Both CISA memory devices and CISA DMA devices can be involved in a DMA transfer, possibly at the same time. The CISA host must handle each situation.

---

* Cycle optionally extended by IOCHRDY shown in gray.

The central consideration is that the CISA host must be able to distinguish between the DMA channels that are on the ISA bus and those that are on the CISA bus. This is a simple matter when the host also incorporates the DMA controller: because the host is responsible for latching the DRQ driveback information, it can determine on a cycle-by-cycle basis whether the DMA device being serviced is on CISA or on ISA according to whether it latched DRQ active for that channel from a CISA driveback cycle.

Because the host has this knowledge, the CISA DMA device does **not** need to assert SEL# on a DACK# cycle. The host already knows the cycle belongs to a CISA DMA device and does not need to see SEL# for the I/O portion of the cycle. This inhibition of SEL# is most important when a CISA memory device is responding to the memory portion of the cycle: the CISA memory device must respond as always with SEL#, and there would be contention (on deassertion) if the CISA DMA device asserted SEL# as well.

The host must foresee the following two situations.

- **DMA transfer between ISA DMA device and any memory device (system DRAM, ISA memory, or CISA memory) -** The host runs a standard CISA memory cycle (I/D#=0, M/IO#=1) along with the ISA memory-I/O cycle. If the selected memory is present on CISA, the device will respond to the access with SEL# as usual. The host **must** drop ALE if SEL# is returned.
- **DMA transfer between CISA DMA device and memory** - The host runs a CISA DACK# cycle (I/D#=0, M/IO#=0). If a CISA memory device claims this cycle it responds with SEL# as usual. The memory device can drive IOCHRDY low to extend the cycle if desired.

The CISA DACK# cycle is described below.

D.2.4 DACK# Cycle

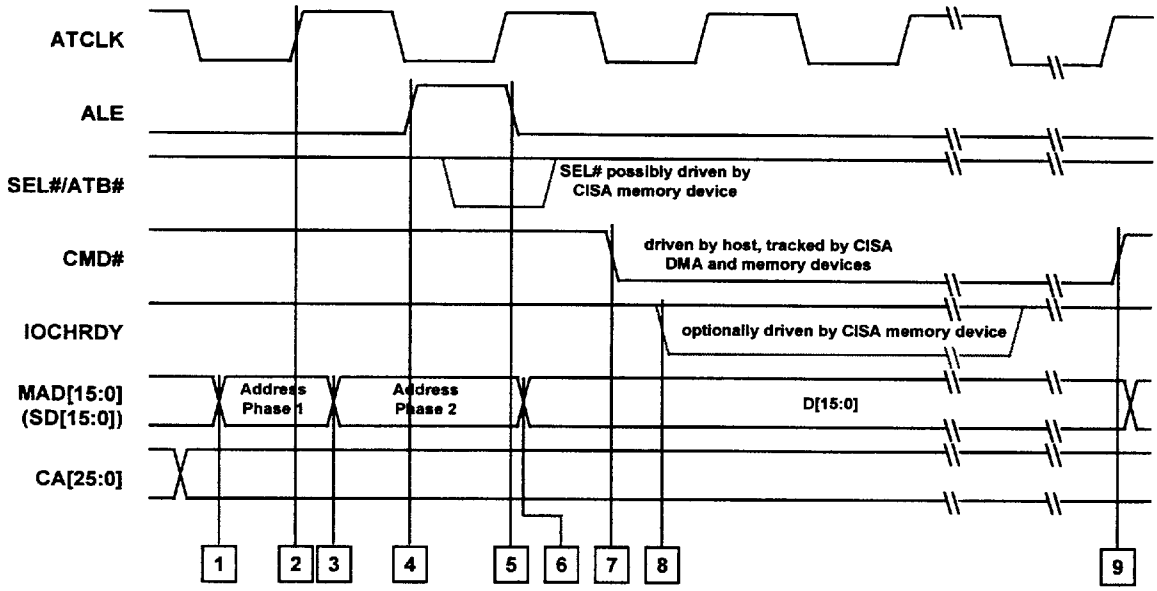
The DACK# cycle is unique in that it has properties of a memory cycle but is directed to an I/O device. Basically, the DACK# cycle is a memory cycle whose address must be decoded by any CISA memory device on the bus. SBHE# and W/R# reference the memory device, not the I/O device; the I/O device must assume the opposite sense of W/R# for its portion of the cycle. Only the memory device responds with SEL#; the DMA (I/O) device never responds. The CMD# timing will be the wider pulse of MEMW#/IOR# or MEMR#/IOW#.

The MAD[15:0] bit meanings for each phase of a DMA Acknowledge cycle are shown below. The M/IO# bit is always 0, and the I/D# bit is always 0, for a DACK# cycle. DMX2-0 encode the number of the DACK#. For example, DMX2-0 = 010 indicate DACK2# active. TC is high if the DACK# is being returned with the Terminal Count indication. Note that there is no ISA# bit, since there is no fast cycle possible.

Phase	MAD 15	MAD 14	MAD 13	MAD 12	MAD 11	MAD 10	MAD 9	MAD 8	MAD 7	MAD 6	MAD 5	MAD 4	MAD 3	MAD 2	MAD 1	MAD 0
1	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SA15	SA14	SA13	SA12	SA11	SA10	I/D# =0	M/IO# =0
2	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	DMX2	DMX1	DMX0	TC	SBHE#	W/R#
3	SD15	SD14	SD13	SD12	SD11	SD10	SD9	SD8	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0

The general structure of Compact ISA DACK# cycles is shown in Figure D-4.

Figure D-4 Compact ISA DACK# Cycle Operation



1. CISA host gets address form the CPU address lines and byte enable lines. Thge host then drives oput A[23:0] + I/D#=0 + M/IO#=0 (DACK# cycle).
2. CISA DAM device, and possibly CISA memory devicee, latches address and cycle type information on the rising edge of TACLK and decodes the information.
3. Host drives out remaining command information on MAD[15:0].
4. Host asserts ALE. CIDA DMA device does not assert SEL# but latches the address and command from MAD[15:0] on the rising edge of ALE. Any CISA memory device present latches address and command, decodes them, and asserts SEL# if appropriate.
5. Host de-asserts ALE and stops driving address on this rising ATCLK edge. Note that in a normal ISA cycle the host would keep ALE high.
6. For DMA I/O read, the host tristates the MAD[15:0] buffers. For DMA I/O write, it drives the write data onto MAD[15:0].
7. Host asserts CMD# synchronous to the falling edge of ATCLK to run the command and is required to inhibit its IOR#/ IOW# lines.
8. Only CISA memory devices can extend the cycle with IOCHRDY.
9. DACK# cycle is minimum 1.5 ATCLK. Host de-asserts CMD# synchronous to the rising edge of ATCLK.

D.2.5 Configuration Cycle

The CISA Configuration Cycle is a special cycle reserved for future expansion of CISA. The only configuration cycle currently defined is the Broadcast cycle; the only type of Broadcast cycle specified at this moment is the Stop Clock cycle.

The Stop Clock cycle indicates that the host will immediately put the CISA peripheral devices into a low-power mode in which they will no longer receive clocks. Therefore, the CISA peripheral device must enter into a state in which it can asynchronously signal that it needs the clocks restarted. CISA devices might need to generate an interrupt back to the system, which they cannot do if not receiving clocks.

The MAD[15:0] bit meanings for each phase of the Stop Clock configuration cycle are shown below.

In phase 1, the M/IO# bit is always 1, and the I/D# bit is always 1, for any configuration cycle. BRD is 1 to indicate a Broadcast cycle, and will always be zero for any other configuration cycle. The STP# bit is 0 to indicate a Stop Clock cycle, and will be 1 for all other cycles. Bits CC2:0 are the Clock Count bits that indicate to the CISA peripheral device how many rising clock edges

to expect after CMD# goes high before the clock is actually stopped. The other bits of phase 1 are reserved and should not be decoded.

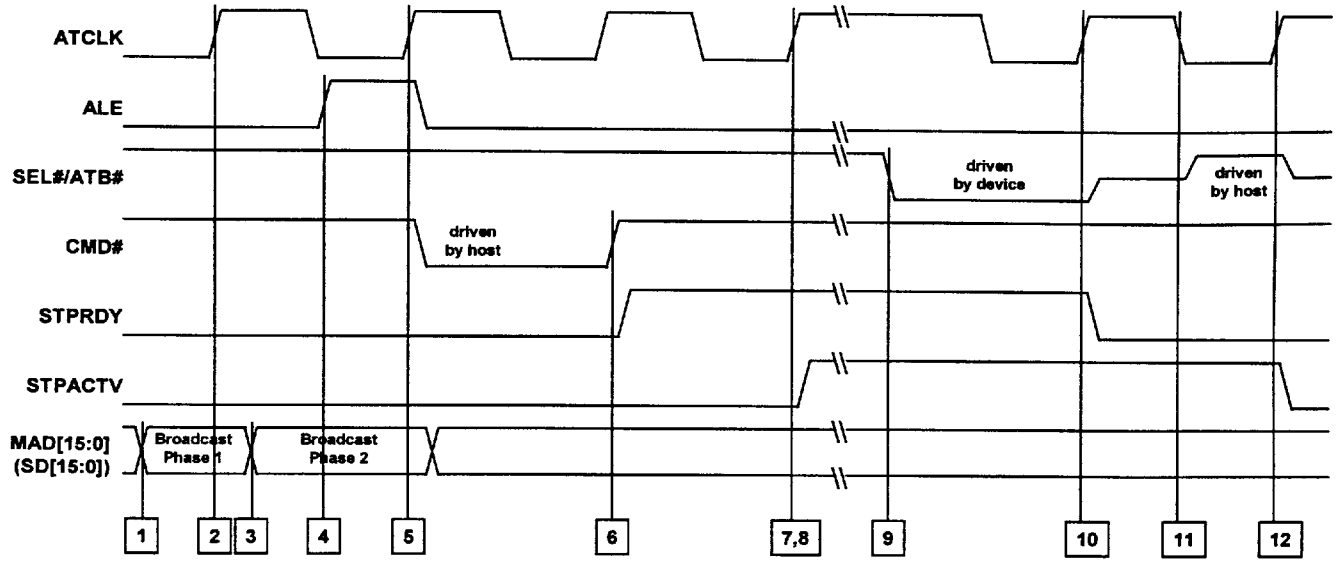
In phase 2, ISA#=1 indicating that this will be a fast cycle. SBHE#=0 to indicate 16 bits of data. W/R#=1 because the Stop Clock Broadcast cycle is always a write cycle.

The data phase of the Stop Clock cycle contains no useful data and should not be latched.

Phase	MAD 15	MAD 14	MAD 13	MAD 12	MAD 11	MAD 10	MAD 9	MAD 8	MAD 7	MAD 6	MAD 5	MAD 4	MAD 3	MAD 2	MAD 1	MAD 0
1	BRD =1	STP# =0	CC2	CC1	CC0	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	rsvd	I/D# =1	M/IO# =1
2	same	same	same	same	same	same	same	same	same	same	same	same	same	ISA# =1	SBHE# =0	W/R# =1
3	same	same	same	same	same	same	same	same	same	same	same	same	same	same	same	same

The general structure of Compact ISA Broadcast cycles is shown in Figure D-5.

Figure D-5 Compact ISA Configuration Cycle Operation



This example describes the Broadcast configuration cycle

1. CISA host initiates the Configuration cycle; it is not generate form ISA commands. The host drives out BRD=1 + I/D#=1 + M/IO#=0 (Broadcast configuration cycle).
2. CISA peripheral latches the command data on the rising edge of ATCLK and decodes the information.
3. Host drives out Clock Count, Stop Clock cycle indicator, and remaining command information on MAD[15:0].
4. Host asserts ALE. CISA devices latch clock count. CISA peripheral devices must NOT respond with SEL#.
5. Host asserts CMD# synchronous to the rising edge of ATCLK to run the command. The Broadcast configuration cycle is always zero wait states so ti completes in one ATCLK.
6. After the host de-asserts CMD#, the CISA peripheral device is internally in STPRDY state.
7. After the number of clocks specified by CC[2:0], the host stops the colck in its high state. In the example, CC[2:0]=001 (the minimum allowed) so the host will stop the clock on the nextrising ATCLK edge. Each additional count requires the host to wait one more clock.
8. The CISA peripheral device is also counting clocks while in STPRDY state. On the specified ATCLK edge the device is in STPACTV state. In STPACTV state, the CISA peripheral device gives SEL#/ATB# a third meaning: CLKRUN#. The device can assert CLKRUN# asynchronously at any time while in this mode to get the host to restart its clocks.
9. CISA peripheral device asserts CLKRUN# (SEL#/ATB#) on receipt of an interrupt to restart the clocks.
10. On next rising ATCLK clock edge, CISA perupheral device de-asserts CLKRUN# (SEL#/ATB#) but must not drive it high. Device has left STPRDY state but is still in STPACTV state and cannot nitiate or respond to any cycle.
11. On next falling ATCLK edge, the host drives SEL#/ATB# high for ½ ATCLK.
12. On next rising ATCLK edge, the host stops driving SEL#/ATB#. The CISA peripheral device leaves STPACTV state on this clock edge and can either generate an interrupt driveback cycle or can respond to cycles from the host.

D.3 Interrupt and DMA Request Drive-Back

Compact ISA provides the signal SEL#/ATB# to give the CISA peripheral device limited ownership of the bus. The SEL#/ATB# signal acts as ATB# (AT backoff) when asserted with ALE low. When the device asserts ATB# to the host, the host inhibits further AT bus operations and asserts the CMD# line to the CISA peripheral device to acknowledge that the device now owns the bus. The peripheral device can only drive two types of information onto the bus: interrupt requests and DMA requests.

Figure 6 illustrates the synchronous IRQ/DRQ driveback cycle.

D.3.1 Interrupt Requests

To drive interrupt requests, the CISA peripheral device drives the MAD[15:0] lines low for each IRQ line it wishes to assert. The host side IRQ generation circuitry samples ATB# and CMD# active on the rising ATCLK edge and latches the IRQ information on MAD[15:0].

The IRQ generation circuitry, whether external or built into the host, determines how to treat IRQ information. For pulse-type interrupts it could latch the IRQs and enable tri-state buffers to drive the lines low for 1-3 ATCLKs, for example.

D.3.2 DMA Requests

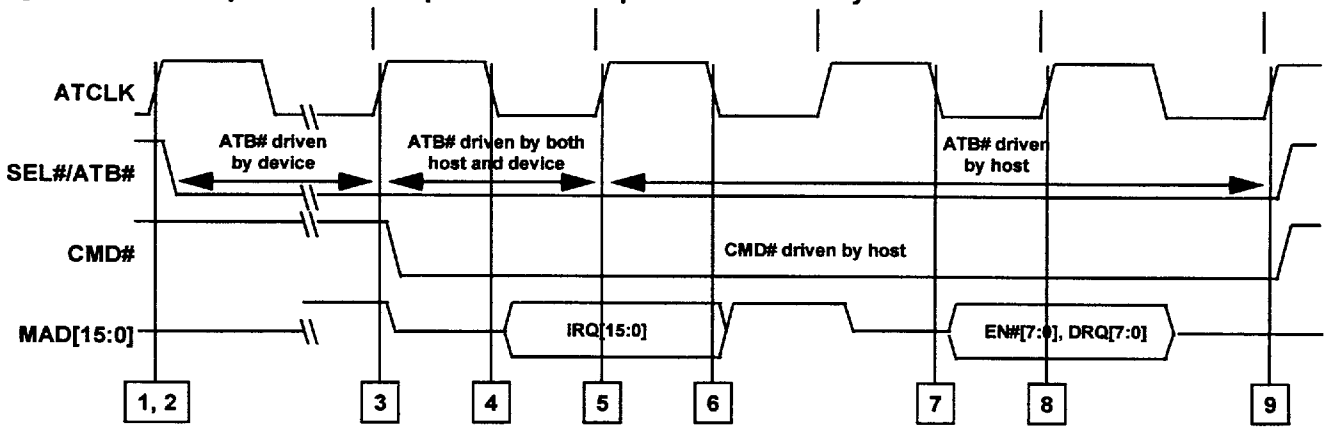
The CISA device must always precede the DRQ drive-back cycle with an IRQ drive-back cycle, even if no IRQs have changed state.

To make DMA requests, the CISA peripheral device drives the MAD[15:8] lines low for each DRQ it wishes to change. The device then sets the state of each MAD[7:0] line to correspond to the DRQ state desired. The host side DRQ generation circuitry samples ATB# and CMD# active on the next rising ATCLK edge after the edge on which IRQs were sampled, and latches the DRQ information on MAD[7:0] for the channels selected on MAD[15:8].

The desired DMA request line states are latched by the host and will remain in that state until cleared by another DRQ drive-back cycle. This scheme allows both DMA single transfer and DMA block transfer modes to be used. The CISA peripheral device must assert SEL#/ATB# immediately any time a DRQ line changes state (assuming the current cycle is finished). The CISA host, in turn, must immediately deassert all DRQ inputs to its DMA controller until the drive-back cycle is complete.

Phase	MAD 15	MAD 14	MAD 13	MAD 12	MAD 11	MAD 10	MAD 9	MAD 8	MAD 7	MAD 6	MAD 5	MAD 4	MAD 3	MAD 2	MAD 1	MAD 0
IRQ	IRQ15	IRQ14	IRQ13	IRQ12	IRQ11	IRQ10	IRQ9	IRQ8	IRQ7	IRQ6	IRQ5	IRQ4	IRQ3	IRQ2	IRQ1	IRQ0
DRQ	EN7#	EN6#	EN5#	Rsvd	EN3#	EN2#	EN1#	EN0#	DRQ7	DRQ6	DRQ5	Rsvd	DRQ3	DRQ2	DRQ1	DRQ0

Figure D-6 Compact ISA Interrupt and DMA Request Drive-Back Cycle



1. CISA Peripheral device must sample SEL#/ATB# and CMD# high, and ALE low, on TWO consecutive rising edges of ATCLK.
2. CISA peripherhal device asserts ATB# on rising edge of ATCLK to request AT backoff. If host was starting a cycle and was about to assert ALE on the next falling edge of ATCLK, it must abort the cycle and retry it later. Even if host is busy and cannot respond to drive back request immediately, it inhibits initiation of all I/O and DMA operations (EOI to PCI is blocked, for example).
3. As soon as AT bus operations have been completed and bus is available, host drives MAD[15:0] high for ½ ATCLK from a falling edge of ATCLK, then aserts CMD# after the net rising edge of ATCLK. The host drives ATB# low at this time.
4. CISA peripheral device(s) can drive interrupt data onto bus on next falling edge of ATCLK, driving low only those lines with IRQ activity and not actively driving high the other lines. In this way, multiple CISA devices can drive the lines in parallel.
5. Host IRQ generation circuitry uses rising edge of ATCLK, qualified by ATB# and CMD# low, to latch IRQs. The CISA device stops driving ATB# at this time. The host controls ATB# throughout the rest of the cycle.
6. CISA peripheral device drives any MAD[15:-0] lines it was driving low high for ½ ATCLK, then tristates the lines for ½ ATCLK.
7. CISA peripheral device drives DRQ information onto MAD[7:0] and at the same time drives low the corresponding lines MAD[15:8] to indicate which DRQ channels have a status change to be transferred.
8. Host DRQ generation circuitry uses next rising edge of ATLCK, qualified by ATB# and CMD# low AND previous IRQ cycle, to latch DRQs. The host DRQ generation circuitry ORs the DRQs with other system DRQs.
9. Host de-asserts CMD# and ATB# on rising edge of ATCLK.

D.4 Performance Control

Compact ISA performance is comparable with that of 16-bit ISA bus peripheral devices. In its simplest implementation, the CMD# signal is simply an AND of MRD#, MWR#, IOR#, and IOW# from the standard AT controller state machine.

Memory cycles are always assumed to be **zero wait state**. The CISA host detects a NOWS# command every time SEL# is generated. The CISA peripheral device can use its IOCHRDY line to extend the cycle and override the NOWS# status. All of this functionality is consistent with standard ISA operation.

I/O cycles cannot be made zero-wait-state cycles on the ISA bus, so by default are not zero-wait-state cycles on the CISA bus. However, performance improvement is possible if the CMD# duration is shortened to one ATCLK. Future PCMCIA I/O devices may be able to complete their cycles this quickly, for example. For zero-wait-state CISA I/O operation, the cycle timing would have to change from the standard ISA timing. The host can implement fast CISA timing as an option. However, all CISA slave devices are **required** to be able to accept fast CISA timing.

Fast CISA timing on the host side is defined as follows. If the CISA host is driving CMD# as derived from the logical AND of ISA command lines IOR#, IOW#, MRD#, and MWR#, it sets ISA#=0 to indicate that the CISA peripheral device must assume ISA timing. If the host is capable of performing fast CISA cycles, it can set ISA#=1. In this case, the CISA peripheral device must deassert IOCHRDY early to lengthen cycles.

Fast CISA timing on the device side is defined as follows. If the CISA host drives the ISA# bit low, the CISA peripheral device assumes normal ISA timing for CMD# and IOCHRDY. If the CISA host drives ISA# high, the CISA peripheral device must drop IOCHRDY low immediately upon receiving CMD# to lengthen the cycle; this is different from ISA timing.

The CISA peripheral device will have a programmable option to determine how IOCHRDY is deasserted. By default, the device might drop IOCHRDY on every cycle. For the example of a PCMCIA controller on the CISA bus, only when a fast PCMCIA card is inserted (as indicated in the CIS header of the card) would Card Services be allowed to enable the fast CISA timing option on the CISA peripheral device side.

D.5 Compatibility and Host Responsibilities

Compact ISA does **not** interfere with standard ISA operations or limit compatibility. This statement can be made with only the following restrictions:

- No device can drive the SD bus between ISA cycles. Devices capable of driving the SD bus must stay tri-stated at this time.
- ATCLK can be stopped only after a Stop Clock Broadcast configuration cycle. Slower-than-standard clock speeds are allowed if interrupt latency is not an issue.
- ISA bus masters cannot access CISA devices. Standard ISA masters are simply ignored by CISA devices since these masters cannot generate CMD# and so cannot run a CISA cycle. ISA bus masters can still take bus control and communicate with other ISA peripherals. CISA interrupt latency may be an issue if a bus master prevents the CISA host from responding to ATB# for an interrupt driveback cycle.
- No CISA bus master capability is currently defined. However, the presence of the SEL#/ATB# signal and its AT backoff feature leave open the possibility of future bus master capabilities.
- On receipt of an ATB# request, the CISA host must immediately inhibit all system DRQ activity (possibly by deasserting all DRQs to the DMA controller) until the drive-back cycle is complete. Otherwise, unwanted DMA cycles could occur.

D.6 Shared Speaker Signal Support (Optional)

Compact ISA provides a new scheme for the digital speaker output signal common to PCs and PCMCIA controllers. This scheme allows all digital audio outputs to be tied together without the XOR logic usually required.

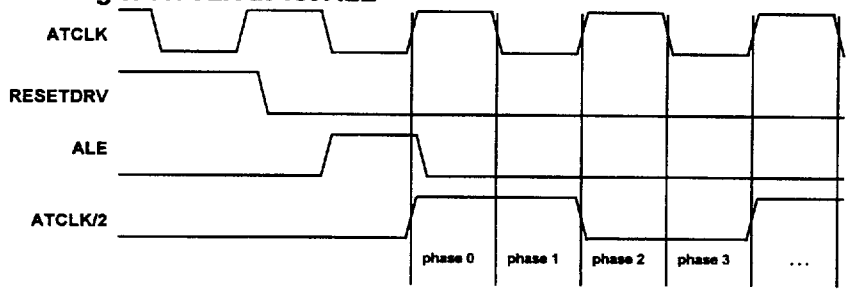
The standard specification for the speaker data output is a signal driven in both the low-to-high and high-to-low directions. The output cannot simply be respecified as open-collector, since there is no guarantee that software will leave the speaker output line from the system chipset in a high or tri-stated condition. If it leaves the signal driven low, no other open-collector devices connected on the line could toggle the signal. Moreover, open collector outputs tend to consume excessive power.

Compact ISA provides an efficient solution to the problem as described in the following sections.

D.6.1 Initial Synchronization

All CISA slave devices must tri-state their SPKR outputs at hard reset time and remain tri-stated until individually enabled. On the first ALE generated by the host, all participating CISA devices will synchronize to ATCLK and derive the signal ATCLK/2 that is in phase as shown in Figure D-7. Four distinct phases, 0 through 3, are the result. CISA slave SPKR outputs are still tri-stated at this point.

Figure D-7 Synchronizing to ATCLK at 1st ALE



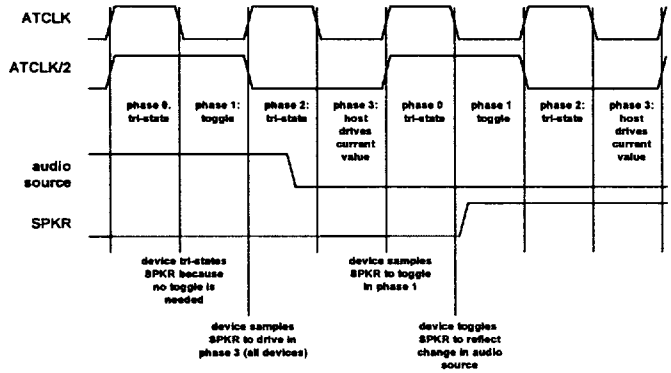
D.6.2 SPKR Sharing During Active Mode

The activities performed in each phase by the CISA host and the CISA slaves are as follows.

Phase	Slave	Host
On the rising ATCLK edge starting phase 0	Sample the state of SPKR.	Sample the state of SPKR. Tristate SPKR output.
During phase 0:	Maintain SPKR output tristated (as it was from previous phase).	Maintain SPKR output tristated.
On the falling ATCLK edge starting phase 1:	Sample digital audio source input.	
During phase 1:	If digital audio source input sampled on ATCLK edge has changed state since the previous phase 1 in which it was sampled, toggle SPKR. SPKR is toggled by driving the opposite of the SPKR value sampled in phase 0 onto the SPKR output.	
On the rising ATCLK edge starting phase 2:	Tristate SPKR output.	Tristate SPKR output. Sample the state of SPKR.
During phase 2:	Slave and host: Maintain SPKR output tristated.	
On the falling ATCLK edge starting phase 3:	No activity on this edge.	Drive SPKR pin to the value of SPKR sampled in phase 2.
During phase 3:	Maintain SPKR output tristated (as it was from previous phase).	Maintain SPKR output driven.

Figure D-8 illustrates the SPKR handling requirements.

Figure D-8 Shared SPKROUT Signal Management



D.6.3 SPKR Sharing During Stop Clock Mode

During Stop Clock mode, CISA devices handle SPKR as follows.

Slave: Tri-state SPKR. Referring to Figure 5, the exact period during which CISA slaves keep SPKR tri-stated is defined as the period during which both STPACTV and STPRDY are high.

Host: Drive or tri-state SPKR. It is recommended that the host drive SPKR low.

Note that even while CISA slave devices are in Stop Clock mode, they must remain synchronized to the correct phase of ATCLK. They do **not** resynchronize on the next ALE.

D.6.4 Audio Output Circuit Recommendations

The SPKR output must **never** be connected directly to a speaker or other low-impedance transducer. The shared SPKR implementation depends on an R-C time constant large enough that the signal will never change its level any appreciable amount across a period of 1.5 ATCLKs, the maximum number of clocks for which no device will be driving the SPKR line.

Three ATCLKs last for approximately 188ns. The R-C time constant of the design must be significantly larger than this value. Connecting an 8 ohm speaker directly would cause the line to begin a transition when it was tri-stated. Therefore, either capacitive coupling or an amplifier circuit with a high-impedance input is recommended.

D.7 Automatic Voltage Threshold Detection

Compact ISA devices are intended to work on either a traditional 5V ISA bus or on a local 3.3V ISA bus. Compact ISA designs are very power-conscious, so using external strap options on each CISA device to select the input buffer threshold may not be the best option.

Therefore, the Compact ISA host is required to use the ALE pin at reset to indicate the ISA bus voltage to CISA slaves. The correspondence is as follows.

- For a 5V ISA bus, the host must assert the ALE signal **high** when RSTDRV goes high, and must keep it asserted for at least 1/2 ATCLK and at most 1 ATCLK after RSTDRV goes low.
- For a 3.3V ISA bus, the host must keep the ALE signal **low** when RSTDRV goes high, and must maintain ALE low for at least 1/2 ATCLK after RSTDRV goes low.

This performance is **required** for CISA hosts, but CISA slave devices are not required to use the feature.

Appendix E. 602A Notebook Companion Chip

E.1 Features

E.1.1 General Features

The 82C602A, in 486 Notebook Mode, provides:

- Multiplexers for interrupt and DMA request scanning
- A byte-wide latch or tristate buffer
- A decoder for DMA acknowledge signal generation
- An XD-SD bus buffer
- An RTC with CMOS RAM
- Miscellaneous logic.

The attached circuit diagram illustrates the internal logic of the notebook mode.

E.1.2 Power-Saving Features

Signals with tristate options are listed below. See the attached circuit diagram for a logic representation of these tristate mechanisms.

- The signals SD[7:0] and XD[7:0] are tristated from a logic combination of ATTRIS# (pin 58), ROMCS# (pin 5), ROMCS#/RTCD# (pin 50), and DWE#/KBDCS# (pin 22).
- The signals DACK0-7#, KBDCS#, SMEMR#, and SMEMW# are tristated when the input signal ATTRIS# is low.
- The signals DO0# and DO0-7 are tristated when the input signal DTRIS# is low.

All other signals are driven to their normal state, usually inactive.

The power to the RTC can be disconnected during system suspend mode even while the rest of the chip remains powered. This feature results in extremely low standby power consumption and is described in the "Reducing Suspend Power Consumption" section that follows.

E.2 Overview

The notebook mode of the OPTi 82C602A chip provides general purpose multiplexers, latches, and logic to anticipate future integrated system designs. This mode is enabled through a strap option that operates as described below.

E.2.1 Modes/Chipset Support

The 82C602A must follow the strapping options show in Table E-1. The 82C602A will sense the XD[7:0] bits during reset to determine which mode it will enter. In order to achieve a '0' value during reset, place a 4.7KΩ pull-down resistor on the appropriate XD line. In order to achieve a '1' value, no external are needed since the 82C602A contains internal pull-up resistors on the XD bus.

The 82C602A is available by default in a 100-pin PQFP (plastic quad flat pack). It is also available in a 100-pin TQFP (thin quad flat pack) by special order for all notebook modes.

Table E-1 Mode Strapping Options

Mode/Chipset Supported	XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0
486 Notebook Mode/ 82C465MV	1	1	1	0	1	1	0	1
Viper Notebook Mode A (Viper NBA)	1	1	1	0	1	1	1	0
Viper Notebook Mode B (Viper NBB)	1	1	0	0	1	1	1	0

NOTE All other strap combinations are reserved for desktop modes.

E.2.2 Design Notes

The following information is important for proper incorporation of the 82C602A in system designs.

1. The 82C602A is a single-voltage part, usually selected as 5V. It has no provisions for 3.3V-to-5V translation. In most cases this limitation is unimportant, as the 82C602A is primarily an AT-bus interface device. However, for implementations that provide 3.3V input signals, the designer should be aware that 3.3V levels on 5V inputs draw excessive idle current (on the order of 1mA per input).

   For example, the '373 latch buffer could be used to buffer eight of the CPU address lines to the AT bus. However, when the system is put into suspend mode, any buffer inputs that remain at 3.3V will cause a current draw and create an undesirable situation for low-power suspend mode operation.

E.2.3 Reducing Suspend Power Consumption

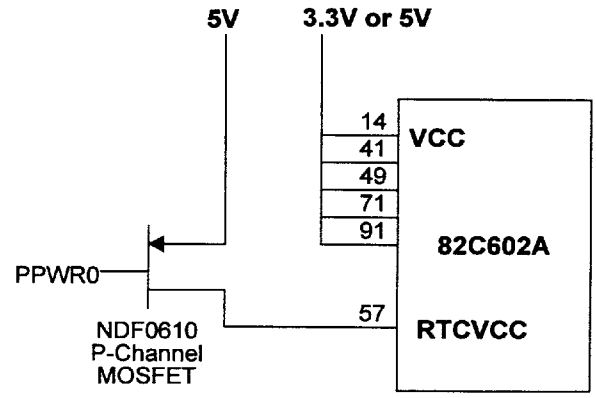
In its notebook modes, the 82C602A chip has been qualified for operation at 3.3V. However, the internal RTC still requires 5V for proper operation. RTCVCC, pin 57, provides Vcc to the RTC during active mode. The VBATT, pin 55, provides power only to maintain the RTC clock and CMOS data during power down modes and is connected to a 2.4V-4.0V battery.

The RTCVCC pin supplies analog circuit power to the RTC during run mode and must always be 5V, regardless of whether the rest of the chip is powered at 5V or 3.3V.

To save power during low-power suspend mode (when the rest of the chip is still powered), this pin may be disconnected from the supply voltage. It is important that the supply be *disconnected,* not simply brought to ground. A p-channel MOSFET is ideal for this purpose. The gate of the MOSFET can be controlled by PPWR0 or PPWR1 from the power control latch.

For example, PPWR0 may be used to switch off RTCVCC by using a P-channel MOSFET as shown in Figure E-1. The auto-toggle feature of the PPWR0 line must be enabled, 54h[0]=0, and 68h[0]=1. Setting bits 68h[3:2]=10 provides the necessary recovery time to the RTC analog Vcc. With this implementation, the MOSFET will switch off the power of the analog Vcc only during suspend mode; the only current flow through the analog Vcc is leakage current (less than 1μA).

Figure E-1 Example of RTCVCC Switching Circuit



The MOSFET used for testing at OPTi is a National Semiconductor NDF0610, which has a typical gate threshold voltage of -2.4V (-3.5V max / -1V min).

E.2.4 82C602A Power Consumption Measurements

Using the circuit of Figure E-1, the power consumption of the 82C602A Notebook Mode in use with the OPTi 463MV demonstration board is shown in the following two tables.

Table E-2 Typical Current Consumption Figures for RTC Power

	Normal	Suspend
Analog Vcc	< 1.5μA	~1μA
VBAT	~1μA	~1μA

Table E-3 Typical Current Consumption Figures for Digital Power

Digital Vcc = 3.3V		Digital Vcc = 5V	
Normal	Suspend	Normal	Suspend
< 4mA	< 30μA	< 4mA	250μA

E.2.5 Internal Real-Time Clock (RTC)

The internal RTC of the 82C602A is functionally compatible with the DS1285/MC146818B. The following sub-sections will give detailed functional and register features of the on-chip RTC of the 82C602A.

E.2.5.1 RTC Features

- System wake-up capability -- alarm interrupt output active in battery back-up mode
- 4.5V to 5.5V operation
- 114 bytes of general non-volatile storage
- Direct clock/calendar replacement for IBM®AT-compatible computers and other applications
- Less than 1.0µA load under battery operation
- 14 bytes for clock/calendar and control
- BCD or binary format for clock and calendar data
- Calendar in day of the week, day of the month, months, and years, with automatic leap-year adjustment
- Time of day in seconds, minutes, and hours
  - 12- or 24-hour format
  - Optional daylight saving adjustment
- Three individually maskable interrupt event flags:
  - Periodic rates from 122µs to 500ms
  - Time-of-day alarm once-per-second to-once-per-day
  - End-of-clock update cycle.

E.2.5.2 RTC Overview

The on-chip RTC is a low-power microprocessor peripheral providing a time-of-day clock and 100 year calendar with alarm features and battery operation. The RTC supports 3.3V systems. Other RTC features include three maskable interrupt sources, square-wave output, and 114 bytes of general non-volatile storage.

Wake-up capability is provided by an alarm interrupt, which is active in battery back-up mode.

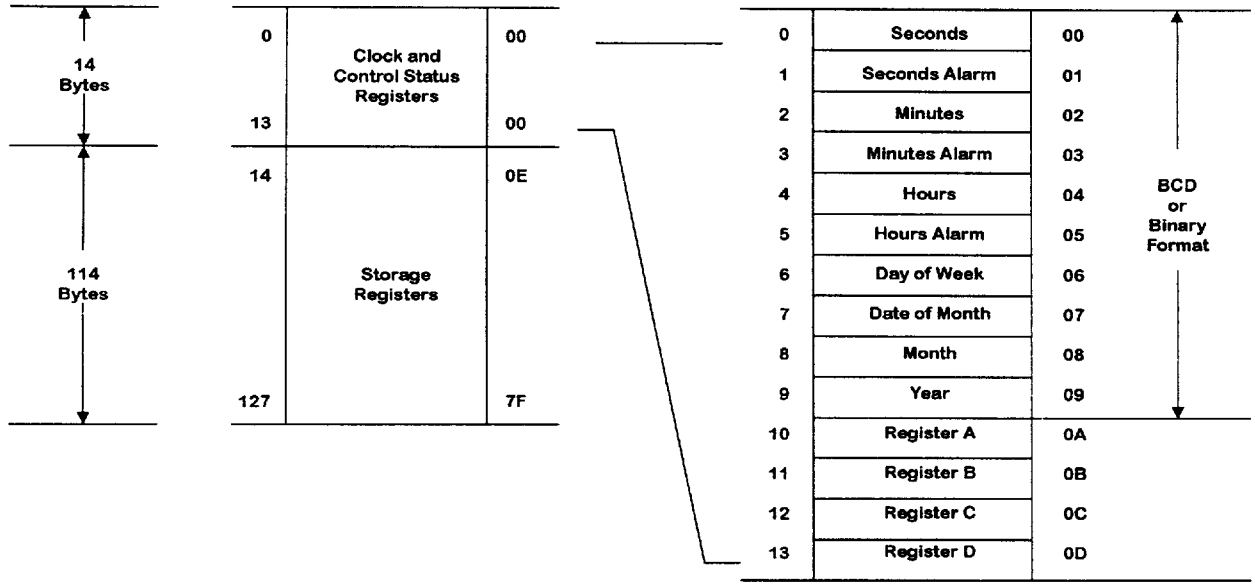
The RTC write-protects the clock, calendar, and storage registers during power failure. A back-up battery then maintains data and operates the clock and calendar.

The on-chip RTC is a fully compatible real-time clock for PC/AT-compatible computers and other applications. The only external components are a 32.768kHz crystal and a back-up battery.

E.2.5.3 RTC Address Map

The on-chip RTC provides 14 bytes of clock and control/status registers and 114 bytes of general non-volatile storage. Figure E-2 illustrates the address map for the RTC.

Figure E-2 RTC Address Map



E.2.5.4 Programming the RTC

The time-of-day, alarm, and calendar bytes can be written in either the BCD or binary format (see Table E-4).

These steps may be followed to program the time, alarm, and calendar:

1. Modify the contents of Register B:
   a. Write a 1 to the UTI bit to prevent transfers between RTC bytes and user buffer.
   b. Write the appropriate value to the data format (DF) bit to select BCD or binary format for all time, alarm, and calendar bytes.
   c. Write the appropriate value to the hour format (HF) bit.
2. Write new values to all the time, alarm, and calendar locations.
3. Clear the UTI bit to allow update transfers.

On the next update cycle, the RTC updates all ten bytes in the selected format.

Table E-4 Time, Alarm, and Calendar Formats

Address	RTC Bytes	Range		
		Decimal	Binary	Binary-Coded Decimal
0	Seconds	0-59	00h-3Bh	00h-59h
1	Seconds Alarm	0-59	00h-3Bh	00h-59h
2	Minutes	0-59	00h-3Bh	00h-59h
3	Minutes Alarm	0-59	00h-3Bh	00h-59h
4	Hours, 12-hour Format	1-12	01h-OCh am 81h-8Ch pm	01h-12h am 82h-92h pm
	Hours, 24-hour Format	0-23	00h-17h	00h-23h
5	Hours Alarm, 12-hour Format	1-12	01h-OCh am 81h-8Ch pm	01h-12h am 82h-92h pm
	Hours Alarm, 24-hour Format	0-23	00h-17h	00h-23h
6	Day of Week (1 = Sunday)	1-7	01h-07h	01h-07h
7	Day of Month	1-31	01h-1Fh	01h-31h
8	Month	1-12	01h-0Ch	01h-12h
9	Year	0-99	00h-63h	00h-99h

E.2.5.5 Square-wave Output

The RTC divides the 32.768kHz oscillator frequency to produce the 1Hz update frequency for the clock and calendar. Thirteen taps from the frequency divider are fed to a 16:1 multiplexer circuit. The output of this mux is fed to the SQW output and periodic interrupt generation circuitry. The four least-significant bits of Register A, RS[3:0], select among the 13 taps (see Table E-5).

E.2.5.6 Interrupts

The RTC allows three individually selected interrupt events to generate an interrupt request. These three interrupt events are:

1. The periodic interrupt, programmable to occur once every 122µs to 500ms.
2. The alarm interrupt, programmable to occur once-per-second to once-per-day, is active in battery back-up mode, providing a "wake-up" feature.
3. The update-ended interrupt, which occurs at the end of each update cycle.

Each of the three interrupt events is enabled by an individual interrupt enable bit in Register B. When an event occurs, its event flag bit in Register C is set. If the corresponding event enable bit is also set, then an interrupt request is generated. The interrupt request flag bit (INTF) of Register C is set with every interrupt request. Reading Register C clears all flag bits, including INTF, and makes INT# high-impedance.

Two methods can be used to process RTC interrupt events:

1. Enable interrupt events and use the interrupt request output to invoke an interrupt service routine.
2. Do not enable the interrupts and use a polling routine to periodically check the status of the flag bits.

The individual interrupt sources are described in detail in the following sub-sections.

Table E-5 Square-Wave Frequency/Periodic Interrupt Rate

Register A Bits				Square-Wave		Periodic Interrupt	
RS3	RS2	RS1	RS0	Frequency	Units	Period	Units
0	0	0	0	None		None	
0	0	0	1	256	Hz	3.90625	ms
0	0	1	0	128	Hz	7.8125	ms
0	0	1	1	8.192	kHz	122.070	µs
0	1	0	0	4.096	kHz	244.141	µs
0	1	0	1	2.048	kHz	488.281	µs
0	1	1	0	1.024	kHz	976.5625	µs
0	1	1	1	512	Hz	1.953125	ms
1	0	0	0	256	Hz	3.90625	ms
1	0	0	1	128	Hz	7.8125	ms
1	0	1	0	64	Hz	15.625	ms
1	0	1	1	32	Hz	31.25	ms
1	1	0	0	16	Hz	62.5	ms
1	1	0	1	8	Hz	125	ms
1	1	1	0	4	Hz	250	ms
1	1	1	1	2	Hz	500	ms

Periodic Interrupt
The mux output used to drive the SQW output also drives the interrupt generation circuitry. If the periodic interrupt event is enabled by writing a 1 to the periodic interrupt enable bit (PIE) in Register C, an interrupt request is generated once every 122µs to 500ms. The period between interrupts is selected by the same bits in Register A that select the square-wave frequency (see Table E-5). Setting OSC[2:0] in Register A to 011 does not affect the periodic interrupt timing.

Alarm Interrupt
The alarm interrupt is active in battery back-up mode, providing a "wake-up" capability. During each update cycle, the RTC compares the hours, minutes, and seconds bytes with the three corresponding alarm bytes. If a match of all bytes is found, the alarm interrupt event flag bit, AF in Register C, is set to 1. If the alarm event is enabled, an interrupt request is generated.

An alarm byte may be removed from the comparison by setting it to a "don't care" state. An alarm byte is set to a "don't care" state by writing a 1 to each of its two most significant bits. A "don't care" state may be used to select the frequency of alarm interrupt events as follows:

A. If none of the three alarm bytes is "don't care," the frequency is once per day, when hours, minutes, and seconds match.

B. If only the hour alarm byte is "don't care", the frequency is once per hour when minutes and seconds match.

C. If only the hour and minute alarm bytes are "don't care", the frequency is once per minute when seconds match.

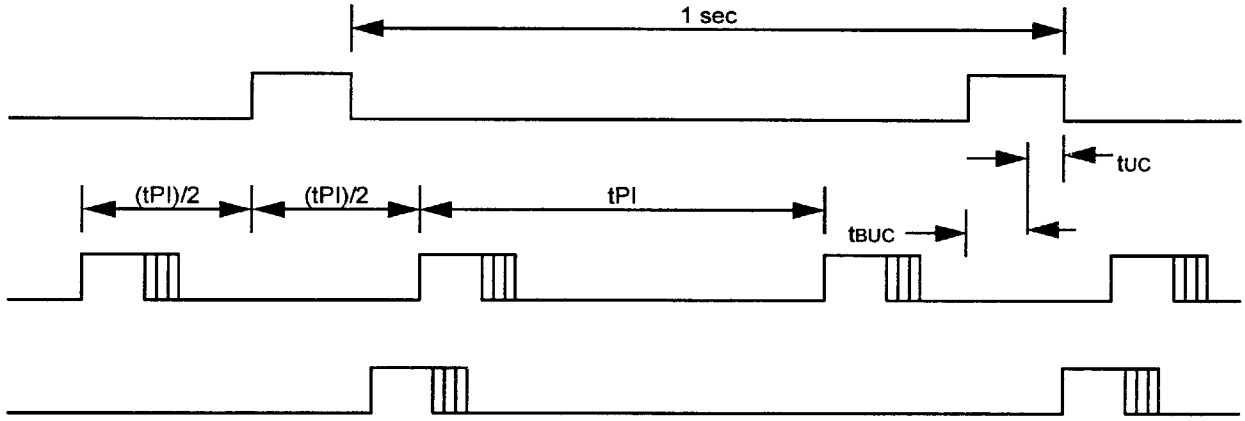
D. If the hour, minute, and second alarm bytes are "don't care", the frequency is once per second.

Update Cycle Interrupt
The update cycle ended flag bit (UF) in Register C is set to a 1 at the end of an update cycle. If the update interrupt enable bit (UIE) of Register B is 1, and the update transfer inhibit bit (UTI) in Register B is 0, then an interrupt request is generated at the end of each update cycle.

Accessing RTC bytes
Time and calendar bytes read during an update cycle may be in error. Three methods to access the time and calendar bytes without ambiguity are:

1. Enable the update interrupt event to generate interrupt requests at the end of the update cycle. The interrupt handler has a maximum of 999ms to access the clock bytes before the next update cycle begins (see Figure E-3).
2. Poll the update-in-progress bit (UIP) in Register A. If UIP = 0, the polling routine has a minimum of tBUC time to access the clock bytes (see Figure E-3).
3. Use the periodic interrupt event to generate interrupt requests every tPI time, such that UIP = 1 always occurs between the periodic interrupts. The interrupt handler has a minimum of tPI/2 + tBUC time to access the clock bytes (see Figure E-3).

Figure E-3 Update-Ended/Periodic Interrupt Relationship



RTC Time-Base Crystal

The RTC's time-base oscillator is designed to work with an external piezoelectric 32.768kHz crystal. A crystal can be represented by its electrical equivalent circuit and associated parameters as shown in Figure E-4 and Table E-6, respectively.

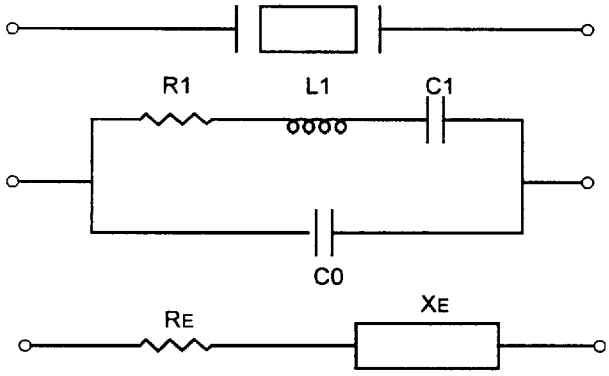
L1, C1, and R1 form what is known as the motional arm of the circuit. C0 is the sum of the capacitance between electrodes and the capacitance added by the leads and mounting structure of the crystal. The equivalent impedance of the crystal varies with the frequency of oscillation.

There are two frequencies at which the crystal impedance appears purely resistive ($X_E = 0$). They're indicated by two points on the graph, known as the series resonant (F_S) and anti-resonant (F_A) frequencies. Oscillators operating the crystal at the resonant frequency (F_S) are termed series resonant circuits, whereas those that operate the crystal around F_A are termed parallel resonant. The on-chip RTC uses a parallel resonant oscillator circuit. The frequency of oscillation in this mode lies between F_S and F_A and is dictated by the effective load capacitance appearing across the crystal inputs, as explained next.

Table E-6 Crystal Parameters

Parameter	Symbol	Value	Unit
Nominal Frequency	F	32.768	kHz
Load Capacitance	C_L	6	pF
Motional Inductance	L1	9076.66	H
Motional Capacitance	C1	2.6 x 10-3	pF
Motional Resistance	R1	27	Kohm
Shunt Capacitance	C0	1.1	pF

Figure E-4 Quartz Crystal Equivalent Circuit



RTC Oscillator

The parallel resonant RTC oscillator circuit is comprised of an inverting micro-power amplifier with a PI-type feedback network. Figure E-6 illustrates a block diagram of the oscillator circuit with the crystal as part of the PI-feedback network. The oscillator circuit ensures that the crystal is operating in the parallel resonance region of the impedance curve.

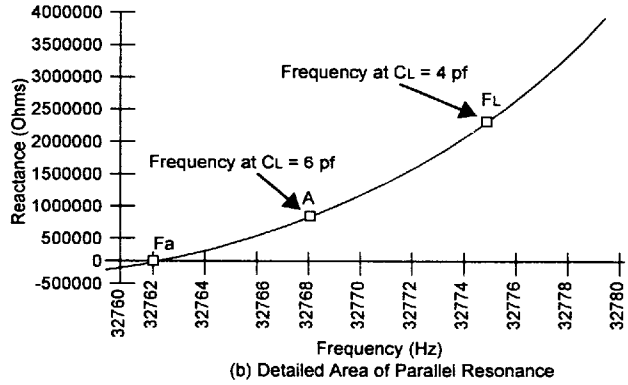
The actual frequency at which the circuit will oscillate depends on the load capacitance, C_L. This parameter is the dynamic capacitance of the total circuit as measured or computed across the crystal terminals.

A parallel resonant crystal like the DT-26 is calibrated at this load using a parallel oscillator circuit. C_L is computed from C_{L1} and C_{L2} as given below:

$$C_L = (C_{L1} * C_{L2}) / (C_{L1} + (C_{L2})$$

The RTC's C_{L1} and C_{L2} values are trimmed to provide approximately a load capacitance (C_L) of 6pF across crystal terminals. This is to match the specified load capacitance (6pf) at which the recommended DT-26 crystal is calibrated to resonate at the nominal frequency of 32.768kHz. Referring to the impedance graph of Figure E-5, "A" indicates the point of resonance when C_L equals the specified load capacitance of the crystal.

Figure E-5 Impedance Graph



(b) Detailed Area of Parallel Resonance

Time Keeping Accuracy

The accuracy of the frequency of oscillation depends on:

- Crystal frequency tolerance
- Crystal frequency stability
- Crystal aging
- Effective load capacitance in oscillator circuit
- Board layout

Crystal Frequency Tolerance. The frequency tolerance parameter is the maximum frequency deviation from the

nominal frequency (in this case 32.768kHz) at a specified temperature, expressed in ppm (parts per million) of nominal frequency. In the case of the Grade A DT-26 crystal, this parameter is ±20 ppm at 25°C.

Crystal Frequency Stability. This parameter, dependent on the angle and type of cut, is defined as the maximum frequency deviation from the nominal frequency over a specified temperature range, expressed in ppm or percentage of nominal frequency.

Figure E-7 shows a typical curve of frequency variation with temperature for the KDS DT-26 crystal.

Figure E-6 RTC Oscillator Circuit Block Diagram

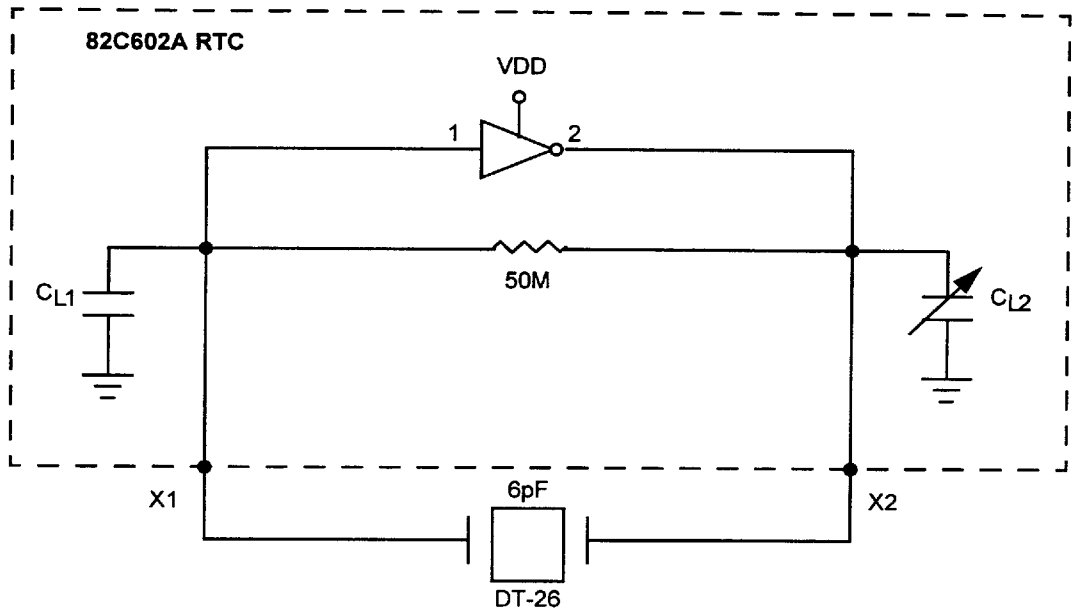
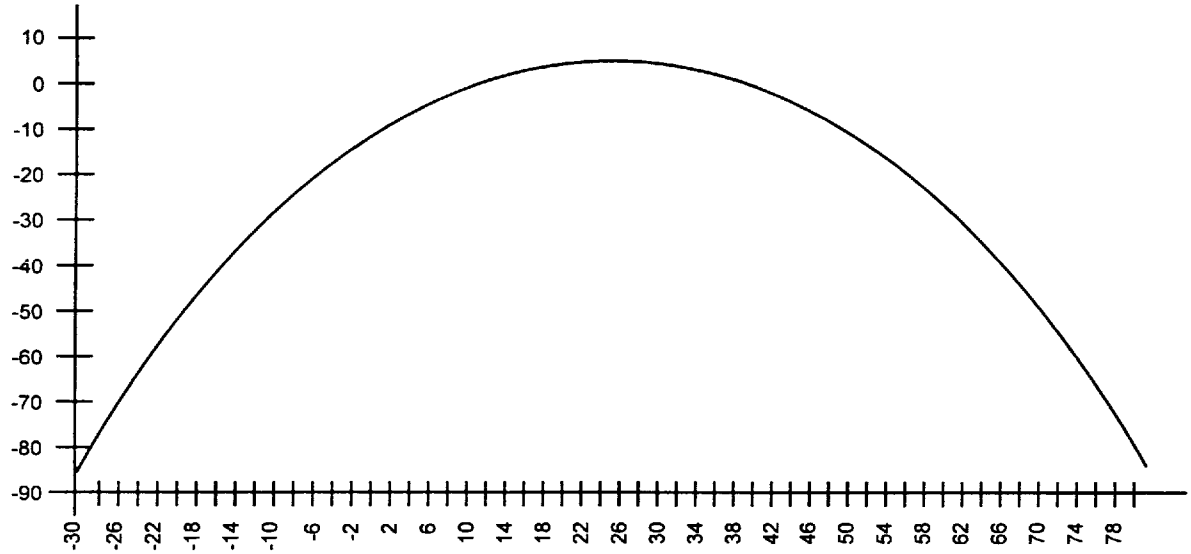


Figure E-7 Typical Temperature Characteristics



Crystal Aging. As a crystal ages, some frequency shift may be observed. Drift with age is specified to be typically 4 ppm for the first year and 2 ppm per year for the life of the KDS DT-26 crystal.

Load Capacitance. For a parallel resonant calibrated crystal, the crystal manufacturer specifies the load capacitance at which the crystal will "parallel" resonate at the nominal frequency. From the graph of Figure E-8, increasing the effective load capacitance by hanging additional capacitors on either of the X1 or the X2 pin will effectively lower the resonant frequency point "A" toward Fs. The deviation of the frequency Fl with load capacitance is given by:

$$F_L = F_S (1 + C1/2 (C0 + C_L))$$

where C1 is the crystal motional capacitance and C0 is the crystal shunt stray capacitance, as explained above. C_L is the effective load capacitance across the crystal inputs.

Allowing for capacitance due to board layout traces leading to the X1and X2 pins, the RTC is trimmed internally to provide an effective load capacitance of less than 6pF. Connecting a 6pF crystal directly to the X1 and X2 pins will cause the clock to oscillate approximately 24 ppm faster than the nominal frequency of 32.768kHz, for reason explained previously.

For maximum accuracy, it is recommended that a small trim capacitor (< 8pF) be hooked to the X2 pin to move the resonant point closer to the nominal frequency. The graph of Figure E-8 shows the variation of frequency with additional load capacitance on the X2 pin of the RTC.

Translating the data in Figure E-8 into a practical rule of thumb: for every additional 1.54pF capacitance on the X2 pin, the frequency will decrease by 0.8Hz or a $\Delta F/F$ of -24.4 ppm around 82.768kHz.

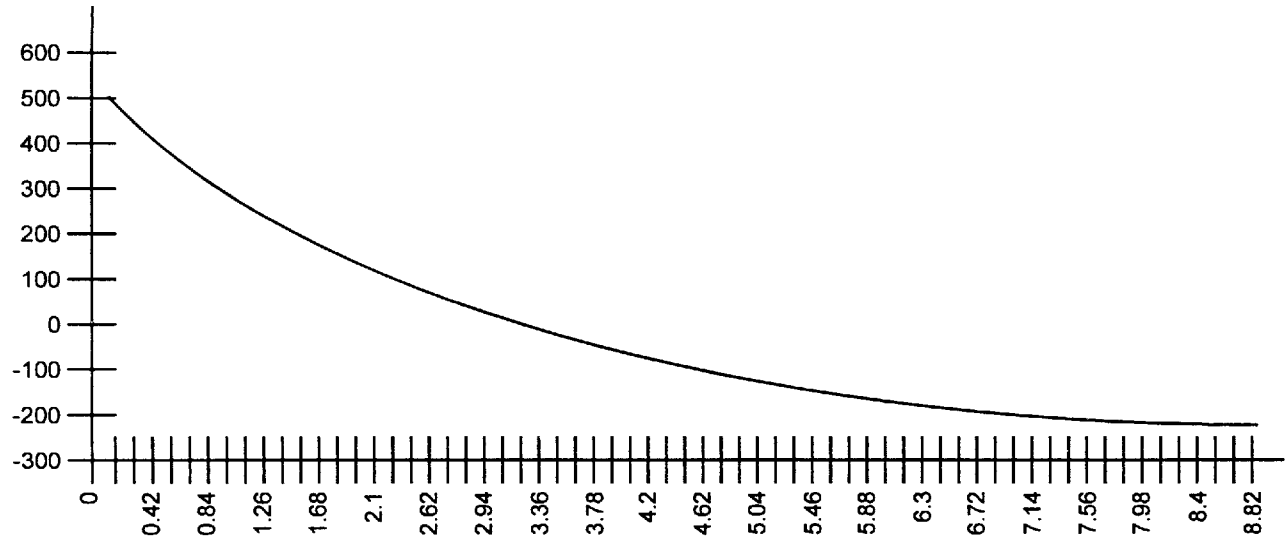
Board Layout. Given the high input impedance of the crystal input pins X1 and X2, care should be taken to route high-speed switching signal traces away from them. Preferably a ground-plane layer should be used around the crystal area to isolate capacitive-coupling of high frequency signals. The traces from the crystal leads to the X1 and X2 pins must be kept short with minimal bends. A good rule of thumb is to keep the crystal traces within 5mm of the X1 and X2 pins.

Finally, a 0.1µF ceramic by-pass capacitor should be placed close to the VCC pin of the RTC to provide an improved supply into the clock

Oscillator Start-up. Barring accuracy issues, the RTC will oscillate with any 32.768kHz crystal. When hooked to the X1 and X2 pins in certain configurations, however, passive components can lead to oscillator start-up problems:

- Excessive loading on the crystal input pins X1 and X2
- Use of a resistive feedback element across the crystal.

Figure E-8 Frequency Variation Versus Load Capacitance



Values above 10pF on either the X1 or X2 pin must be avoided. The feedback element is build into the RTC for start-up and no resistive feedback external to the part is required.

Oscillator Control
When power is first applied to the RTC and VCC is above VPFD, the internal oscillator and frequency divider are turned on by writing a 010 pattern to bits 4 through 6 of Register A. A pattern of 011 behaves as 010 but additionally transforms Register C into a read/write register. This allows the 32.768kHz output on the square-wave pin to be turned on. A pattern of 11X turns the oscillator on, but keeps the frequency divider disabled. Any other pattern to these bits keeps the oscillator off.

E.2.5.7 Power-Down/Power-Up Cycle

The RTC's power-up/power-down cycles are different. The RTC continuously monitors VCC for out-of-tolerance. During a power failure, when VCC falls below VPFD (2.53V typical), the RTC write-protects the clock and storage registers. The power source is switched to BC when VCC is less than VPFD and BC is greater than VPFD, or when VCC is less than VBC and VBC is less than VPFD. RTC operation and storage data are sustained by a valid back-up energy source. When VCC is above VPFD, the power source is VCC. Write-protection continues for tCSR time after VCC rises above VPFD.

The RTC continuously monitors VCC for out-of-tolerance. During a power failure, when VCC falls below VPFD (4.17V typical), the RTC write-protects the clock and storage registers. When VCC is below VBC (3V typical), the power source is switched to BC. RTC operation and storage data are sustained by a valid back-up energy source. When VCC is above VBC, the power source is VCC. Write-protection continues for tCSR time after VCC rises above VPFD.

E.2.5.8 Control/Status Registers

The four control/status registers of the RTC are accessible regardless of the status of the update cycle (see Table E-7).

Register A
Register A programs the frequency of the periodic event rate, and oscillator operation. Register A provides the status of the update cycle. See Table E-8 for Register A's format.

Register B
Register B enables the update cycle transfer operation, square-wave output-interrupt events, and daylight saving adjustment. Register B selects the clock and calendar data formats. See Table E-9 for Register B's format.

Register C
Register C is a read-only event status register. See Table E-10 for Register C's format.

Register D
Register D is a read-only data integrity status register. See Table E-11 for Register D's format.

Table E-7 Control/Status Registers Summary

Register	Loc. (Hex)	Read	Write	Bit Name and State on Reset															
				7 (MSB)		6		5		4		3		2		1		0 (LSB)	
A	0A	Yes	Yes[1]	UIP	NA	OSC2	NA	OSC1	0	OSC0	0	RS3	NA	RS2	NA	RS1	NA	RS0	NA
B	0B	Yes	Yes	UTI	NA	PIE	0	AIE	0	UIE	0	SQWE	0	DF	NA	HF	NA	DSE	NA
C	0C	Yes	No[2]	INTF	0	PF	0	AF	0	UF	0	-	0	32KE	0	-	0	-	0
D	0D	Yes	No	VRT	NA	-	0	-	0	-	0	-	0	-	0	-	0	-	0

NOTE NA = Not Affected
1. Except Bit 7
2. Read/write only when OSC[2:0] in Register A is 011 (binary).

Table E-8 Register A

Bit(s)	Type	Function
7	RO	UIP - Update-In-Progress: This read-only bit is set prior to the update cycle. When UIP equals 1, an RTC update cycle may be in progress. UIP is cleared at the end of each update cycle. This bit is also cleared when the update transfer inhibit (UTI) bit in Register B is 1.
6:4		OSC[2:0] - Oscillator Control Bits 2 through 0: These three bits control the state of the oscillator and divider stages. A pattern of 010 enables RTC operation by turning on the oscillator and enabling the frequency divider. A pattern of 11X turns the oscillator on, but keeps the frequency divider disabled. When 010 is written, the RTC begins its first update after 500ms.
3:0		RS[3:0] - Rate Select Bits 3 through 0: These bits select one of the 13 frequencies for the SQW output and the periodic interrupt rate, as shown in Table E-4.

Table E-9 Register B

Bit(s)	Type	Function
7	R/W	UTI - Update Transfer Inhibit: This bit inhibits the transfer of RTC bytes to the user buffer. 1 = Inhibits transfer and clears UIE0 = Allows Transfer
6	R/W	PIE - Periodic Interrupt Enable: This bit enables an interrupt request due to a periodic interrupt event. 1 = Enabled 0 = Disabled
5	R/W	AIE - Alarm Interrupt Enable: This bit enables an interrupt request due to an alarm interrupt event. 1 = Enabled 0 = Disabled
4	R/W	UIE - Update Cycle Interrupt Enable: This bit enables an interrupt request due to an update ended interrupt event. 1 = Enabled 0 = Disabled The UIE bit is automatically cleared when the UTI bit equals 1.
3	R/W	SQWE - Square-wave Enable: This bit enables the square-wave output. 1 = Enabled 0 = Disabled and Held Low
2	R/W	DF - Data Format: This bit selects the numeric format in which the time, alarm, and calendar bytes are represented. 1 = Binary 0 = BCD
1	R/W	HF - Hour Format: This bit selects the time-of-day and alarm hour format. 1 = 24-Hour Format 0 = 12-Hour Format
0	R/W	DSE - Daylight Saving Enable: This bit enables daylight saving time adjustments when written to 1. On the last Sunday in October, the first time the RTC increments past 1:59:59 AM, the time falls back to 1:00:00 am. On the first Sunday in April, the time springs forward from 2:00:00 am to 3:00:00 am.

Table E-10 Register C

Bit(s)	Type	Function
7	R/W	INTF - Interrupt Request Flag: This flag is set to a 1 when any of the following is true; AIE = 1 and AF = 1 PIE = 1 and PF = 1 UIE = I and UF=1 Reading Register C clears this bit.
6	R/W	PF - Periodic Event Flag: This bit is set to a 1 every tPI time, where tPI is the time period selected by the settings of RS[3:0] in Register A. Reading Register C clears this bit.
5	R/W	AF - Alarm Event Flag: This bit is set to a 1 when an alarm event occurs. Reading Register C clears this bit.
4	R/W	UF - Update Event Flag: This bit is set to a 1 at the end of the update cycle. Reading Register C clears this bit.
3:0	R/W	NU - Not Used - This bit is always set to 0.

Table E-11 Register D

Bit(s)	Type	Function
7	RO	VRT - Valid RAM and Time: 1 = Valid backup energy source 0 = Backup energy source is depleted When the backup energy source is depleted (VRT = 0), data integrity of the RTC and storage registers is not guaranteed.
6:0	RO	NU - Not Used - These bits are always set to 0.

E.3 Signal Definitions

Figure E-9 486 NB Mode Pin Diagram (100-Pin PQFP)

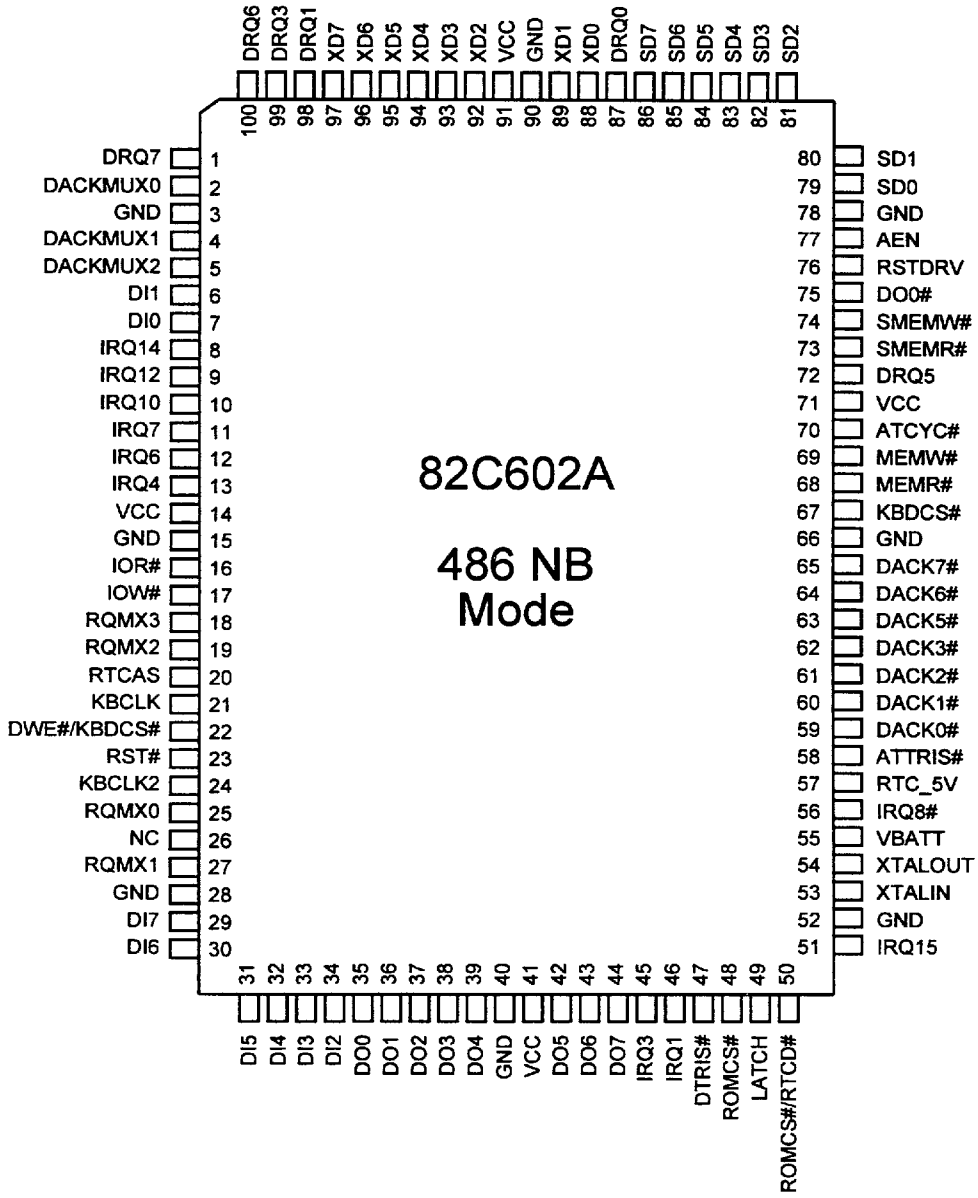


Figure E-10 486 NB Mode Pin Diagram (100-Pin TQFP)

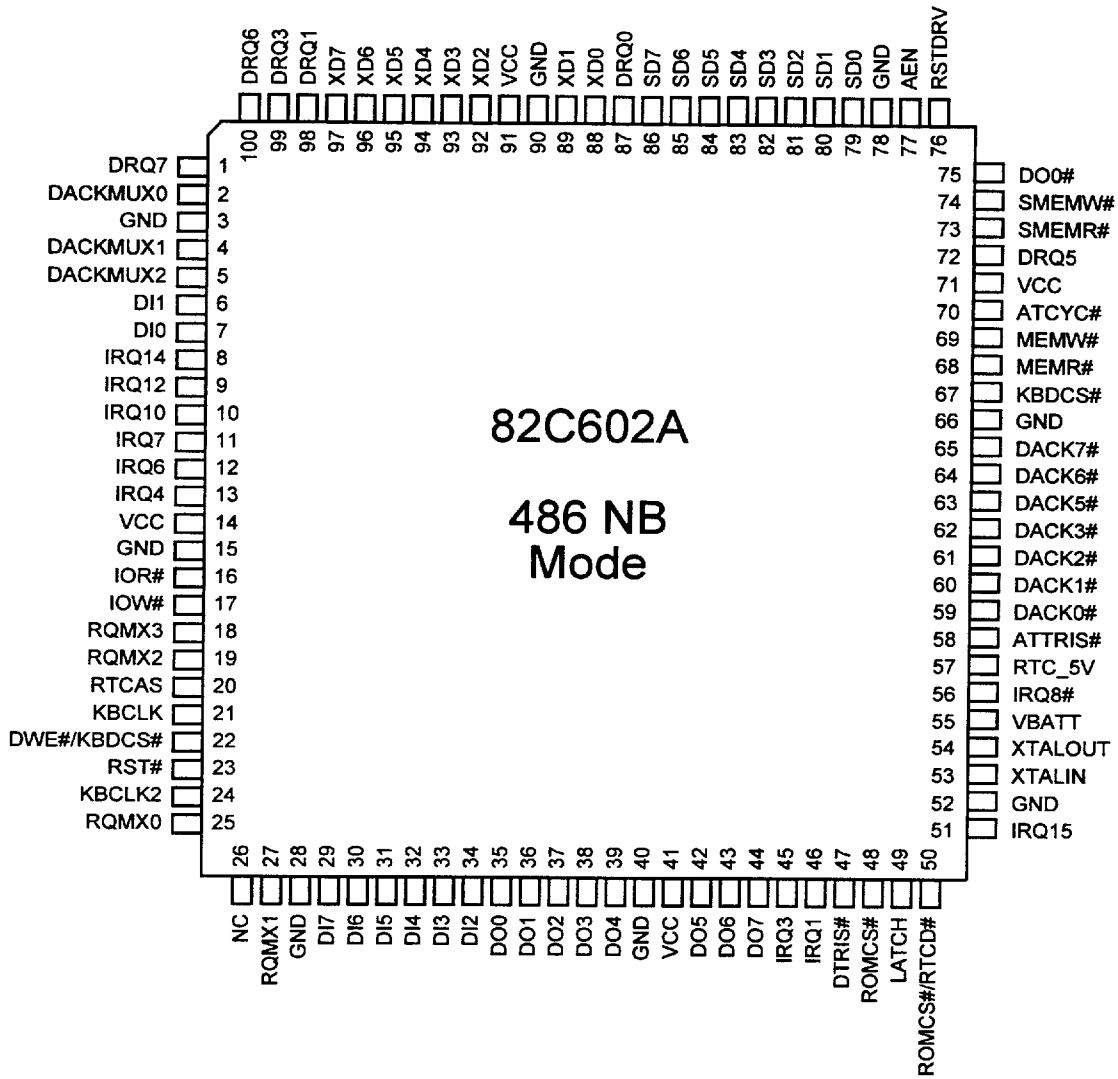


Table E-12 486 NB Mode - Numerical Pin Cross-Reference List

Pin #	Pin Name	Pin Type
1	DRQ7	I
2	DACKMUX0	I
3	GND	G
4	DACKMUX1	I
5	DACKMUX2	I
6	DI1	I
7	DI0	I
8	IRQ14	I
9	IRQ12	I
10	IRQ10	I
11	IRQ7	I
12	IRQ6	I
13	IRQ4	I
14	VCC	P
15	GND	G
16	IOR#	I
17	IOW#	I
18	RQMX3	O
19	RQMX2	O
20	RTCAS	I
21	KBCLK	I
22	DWE#/KBDCS#	I
23	RST#	I
24	KBCLK2	I
25	RQMX0	O

Pin #	Pin Name	Pin Type
26	NC	
27	RQMX1	O
28	GND	G
29	DI7	I
30	DI6	I
31	DI5	I
32	DI4	I
33	DI3	I
34	DI2	I
35	DO0	O
36	DO1	O
37	DO2	O
38	DO3	O
39	DO4	O
40	GND	G
41	VCC	P
42	DO5	O
43	DO6	O
44	DO7	O
45	IRQ3	I
46	IRQ1	I
47	DTRIS#	I
48	ROMCS#	I
49	LATCH	I
50	ROMCS#/RTCD#	I

Pin #	Pin Name	Pin Type
51	IRQ15	I
52	GND	G
53	XTALIN	I
54	XTALOUT	O
55	VBATT	I
56	IRQ8#	O
57	RTC_5V	I
58	ATTRIS#	I
59	DACK0#	O
60	DACK1#	O
61	DACK2#	O
62	DACK3#	O
63	DACK5#	O
64	DACK6#	O
65	DACK7#	O
66	GND	G
67	KBDCS#	O
68	MEMR#	I
69	MEMW#	I
70	ATCYC#	I
71	VCC	P
72	DRQ5	I
73	SMEMR#	O
74	SMEMW#	O
75	DO0#	O

Pin #	Pin Name	Pin Type
76	RSTDRV	O
77	AEN	O
78	GND	G
79	SD0	I/O
80	SD1	I/O
81	SD2	I/O
82	SD3	I/O
83	SD4	I/O
84	SD5	I/O
85	SD6	I/O
86	SD7	I/O
87	DRQ0	I
88	XD0	I/O
89	XD1	I/O
90	GND	G
91	VCC	P
92	XD2	I/O
93	XD3	I/O
94	XD4	I/O
95	XD5	I/O
96	XD6	I/O
97	XD7	I/O
98	DRQ1	I
99	DRQ3	I
100	DRQ6	I

Table E-13 486 NB Mode - Alphabetical Pin Cross-Reference List

Pin Name	Pin #	Pin Type
AEN	77	O
ATCYC#	70	I
ATTRIS#	58	I
DACK0#	59	O
DACK1#	60	O
DACK2#	61	O
DACK3#	62	O
DACK5#	63	O
DACK6#	64	O
DACK7#	65	O
DACKMUX0	2	I
DACKMUX1	4	I
DACKMUX2	5	I
DI0	7	I
DI1	6	I
DI2	34	I
DI3	33	I
DI4	32	I
DI5	31	I
DI6	30	I
DI7	29	I
DO0#	75	O
DO0	35	O
DO1	36	O
DO2	37	O

Pin Name	Pin #	Pin Type
DO3	38	O
DO4	39	O
DO5	42	O
DO6	43	O
DO7	44	O
DRQ0	87	I
DRQ1	98	I
DRQ3	99	I
DRQ5	72	I
DRQ6	100	I
DRQ7	1	I
DTRIS#	47	I
DWE#/KBDCS#	22	I
GND	3	G
GND	15	G
GND	28	G
GND	40	G
GND	52	G
GND	66	G
GND	78	G
GND	90	G
IOR#	16	I
IOW#	17	I
IRQ1	46	I
IRQ3	45	I

Pin Name	Pin #	Pin Type
IRQ4	13	I
IRQ6	12	I
IRQ7	11	I
IRQ8#	56	O
IRQ10	10	I
IRQ12	9	I
IRQ14	8	I
IRQ15	51	I
KBCLK	21	I
KBCLK2	24	I
KBDCS#	67	O
LATCH	49	I
MEMR#	68	I
MEMW#	69	I
NC	26	
ROMCS#	48	I
ROMCS#/RTCD#	50	I
RQMX0	25	O
RQMX1	27	O
RQMX2	19	O
RQMX3	18	O
RTCAS	20	I
RST#	23	I
RSTDRV	76	O
RTC_5V	57	I

Pin Name	Pin #	Pin Type
SD0	79	I/O
SD1	80	I/O
SD2	81	I/O
SD3	82	I/O
SD4	83	I/O
SD5	84	I/O
SD6	85	I/O
SD7	86	I/O
SMEMR#	73	O
SMEMW#	74	O
VBATT	55	I
VCC	14	P
VCC	41	P
VCC	71	P
VCC	91	P
XTALIN	53	I
XTALOUT	54	O
XD0	88	I/O
XD1	89	I/O
XD2	92	I/O
XD3	93	I/O
XD4	94	I/O
XD5	95	I/O
XD6	96	I/O
XD7	97	I/O

E.3.1 486 NB Mode Signal Descriptions

Refer to the 486 NB internal circuitry schematic in Section 4.0 for complete details.

E.3.1.1 Clock and Reset Interface Signals

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
RST#	23	I-S	***Reset:*** Reset input to the 82C602A logic.
RSTDRV	76	O (24mA)	***Reset Drive:*** Inverted RST#.

E.3.1.2 Interrupt/Control Interface Signals

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
IRQ1, IRQ3, IRQ4, IRQ6, IRQ7	46, 45, 13, 12, 11	I	Interrupt Request Bits 1, 3, 4, 6, and 7.
IRQ10, IRQ12, IRQ14, IRQ15	10, 9, 8, 51	I	Interrupt Request Bits 10, 12, 14, and 15.
IOR#	16	I	I/O Read
IOW#	17	I	I/O Write
MEMR#	68	I	Memory Read
MEMW#	69	I	Memory Write
SMEMR#	73	O (24mA)	SMEMR# with tristate control.
SMEMW#	74	O (24mA)	SMEMW# with tristate control.

E.3.1.3 ISA DMA Arbiter Interface Signals

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
DRQ[7:5] DRQ3, DRQ1, DRQ0	1, 100, 72, 99, 98, 87	I	DMA Request bits 7 through 5, 3, 1, and 0.
DACK[7:5]#, DACK[3:0]#	65:63, 62:59	O (6mA)	DMA Acknowledge bits 7 through 5, and 3 through 0.
DACKMUX[2:0]	5, 4, 2	I	Encoded DACKs
RQMX3	18	O (6mA)	Mux of DRQ1, DRQ3, DRQ6, DRQ7
RQMX2	19	O (6mA)	Mux of IRQ10, IRQ15, DRQ05
RQMX1	27	O (4mA)	Mux of IRQ4, IRQ6, IRQ8#, IRQ12

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
RQMX0	25	O (4mA)	Mux of IRQ1, IRQ3, IRQ7, IRQ14
DWE#/KBDCS#	22	i	DRAM Write Enable or Keyboard Chip Select

E.3.1.4 Bus Interface Signals

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
DI[7:0]	29:34, 6, 7	I	Data Buffer Inputs 7 through 0
DO[7:0]	44:42, 39:35	O (4mA)	Data Buffer Outputs 7 through 0
DO0#	75	O (8mA)	Inverted Data Buffer Output 0
DTRIS#	47	I	Data Buffer Tristate Control: When active, will tristate the data buffer.
ROMCS#	48	I	***ROM Chip Select:*** This signal, when active, will allow ROM on the XD bus to put information on the SD bus.
ROMCS#/RTCD#	50	I	***ROM Chip Select and RTC Command Line:*** This signal is used to enable accesses to the ROM and RTC from the 82C465MV.
SD[7:0]	86:79	I/O (24mA)	SD Bus Lines 7 through 0
XD[7:0]	97:92, 89, 88	I/O (6mA)	***XD Bus Data Lines 7 through 0:*** XD4 and XD1 must be sampled low during reset to enter the 486 Notebook Mode. A 2.2K pull-down resistor is recommended on these lines. All XD lines in the 82C602A have internal weak pull-up resistors and do not require any external pull-up resistors.
ATTRIS#	58	I	***Tristates AT Bus Outputs:*** This is used to tristate the AT bus during low power mode.
ATCYC	70	I	AT Cycle Indication

E.3.1.5 Real-Time Clock and Keyboard Interface Signals

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
RTCAS	20	I	***Real-time Clock Address Strobe:*** RTCAS is used to demultiplex the address/data bus. The falling edge of AS latches the address on XD[7:0].
RTC_5V	57	I	***Real-time Clock 5.0V:*** This pin must be connected to +5V. This input will prevent the lithium battery from being accessed during power-on.
VBATT	55	I	***Voltage Battery:*** This pin is connected to the CMOS and RTC battery.
IRQ8#	56	O	***Interrupt Request Bit 8:*** The alarm output interrupt generated by the internal RTC.
XTALIN	53	I	***Crystal Oscillator Input:*** 32.768KHz XTAL input.
XTALOUT	54	O	***Crystal Oscillator Output:*** 32.768KHz XTAL output.

Signal Name	Pin No.	Signal (Drive) Type	Signal Description
KBCLK	21	I	***Keyboard Clock:*** This input is used for demuxing interrupts and DMA requests.
KBCLK2	24	I	***Keyboard Clock / 2:*** This input is used for demuxing interrupts and DMA requests.
KBDCS#	67	O (6mA)	***KBDCS# qualified with AEN:*** Allows the system to access the keyboard controller.

E.3.1.6 Miscellaneous Interface Signals

Signal Name	Pin No.	Signal Type	Signal Description
NC	26		***No Connection:*** This pin should be left unconnected.
LATCH	49	I	***Data Buffer Latch:*** This signal controls the latching of information on the data bus.
AEN	77	I	***Address Enable:*** This input is used to ensure that the system has access to the real-time clock.

E.3.1.7 Power and Ground Pins

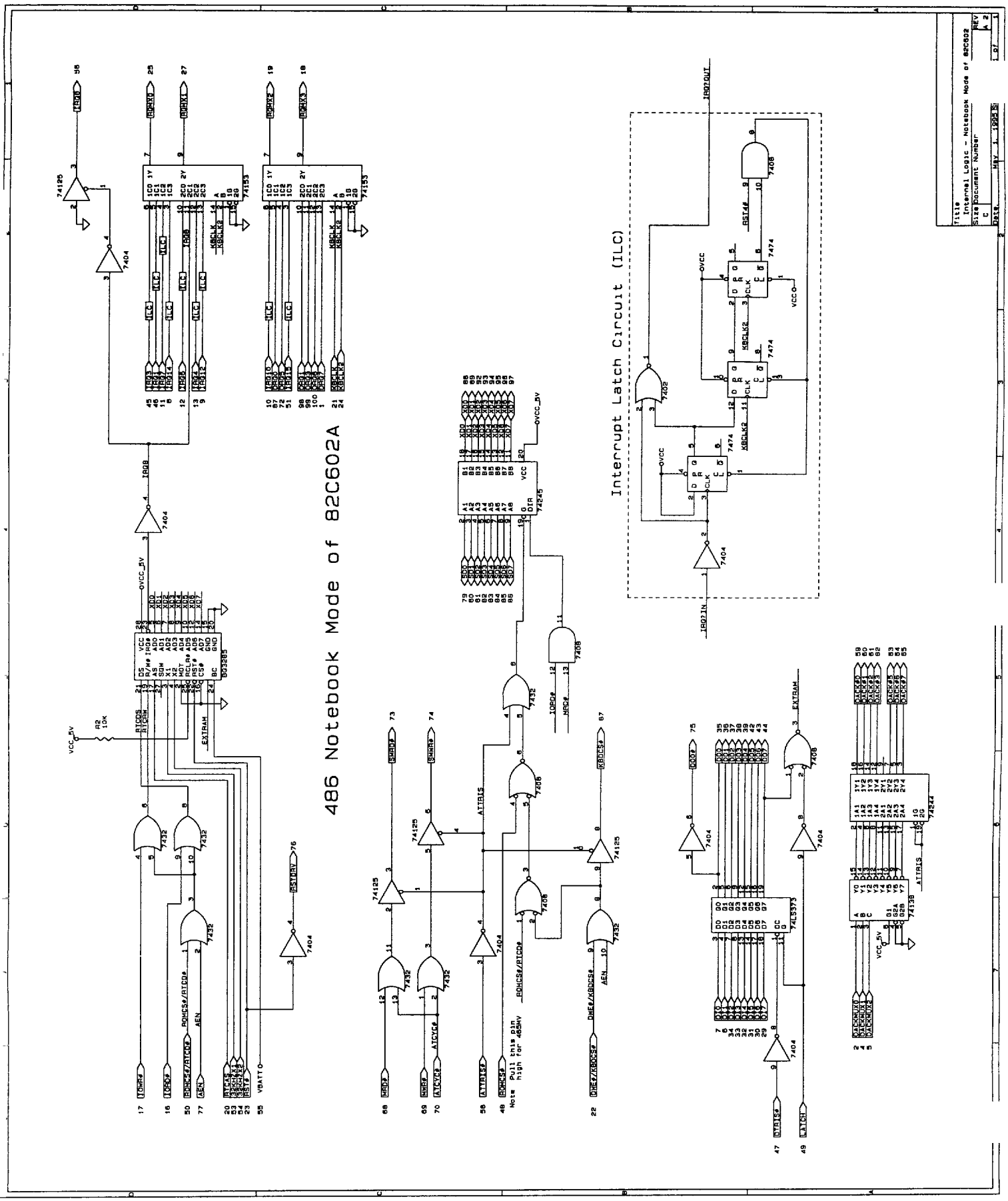
Signal Name	Pin No.	Signal Type	Signal Description
VCC	14, 41, 71, 91	P	Power Connection
GND	3, 15, 28, 40, 52, 66, 78, 90	G	Ground Connection

Legend:

G	Ground
I/O	Input/Output
G	Ground
OD	Open Drain
I/O	Input/Output
P	Power
Sch	Schmitt-trigger

E.4 Schematics

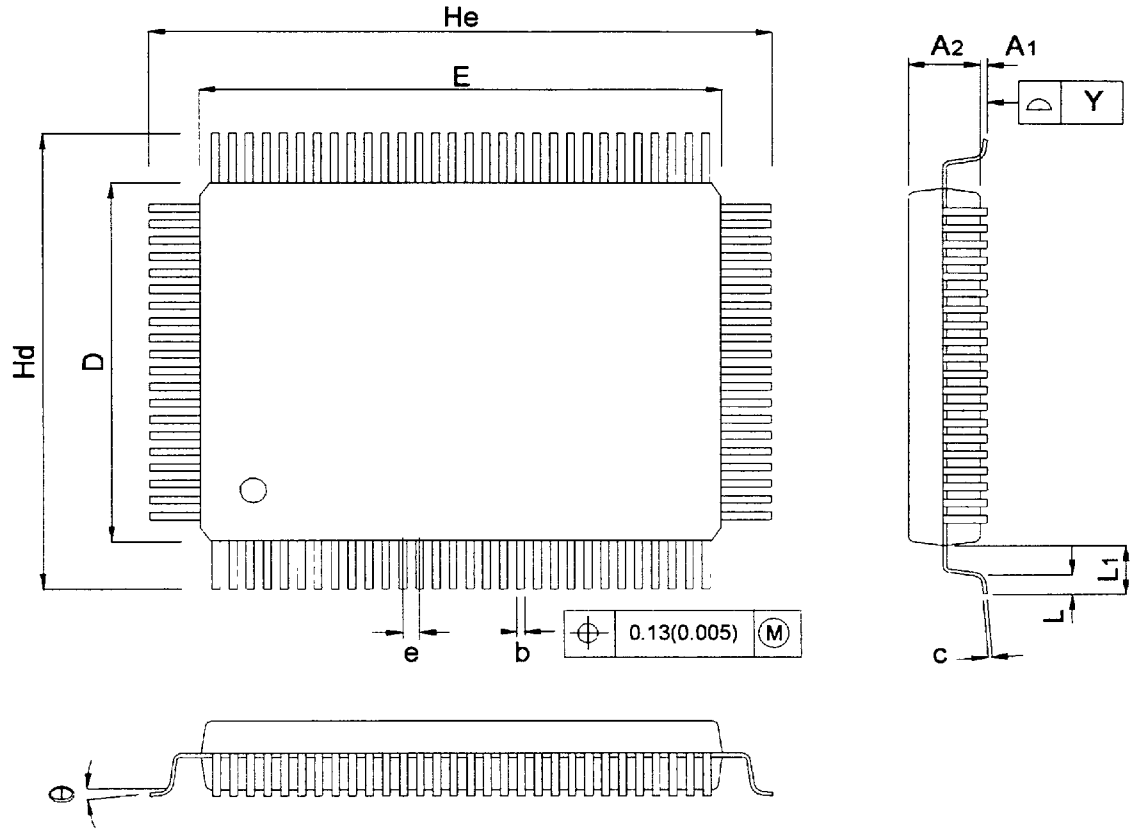
The next page shows a schematic of the internal circuitry for the 82C602A when in the 486 Notebook Mode.



E.5 82C602A Mechanical Package Outline

The 82C602A is available in a 100-pin TQFP by special order for all notebook modes; default packaging is 100-pin PQFP

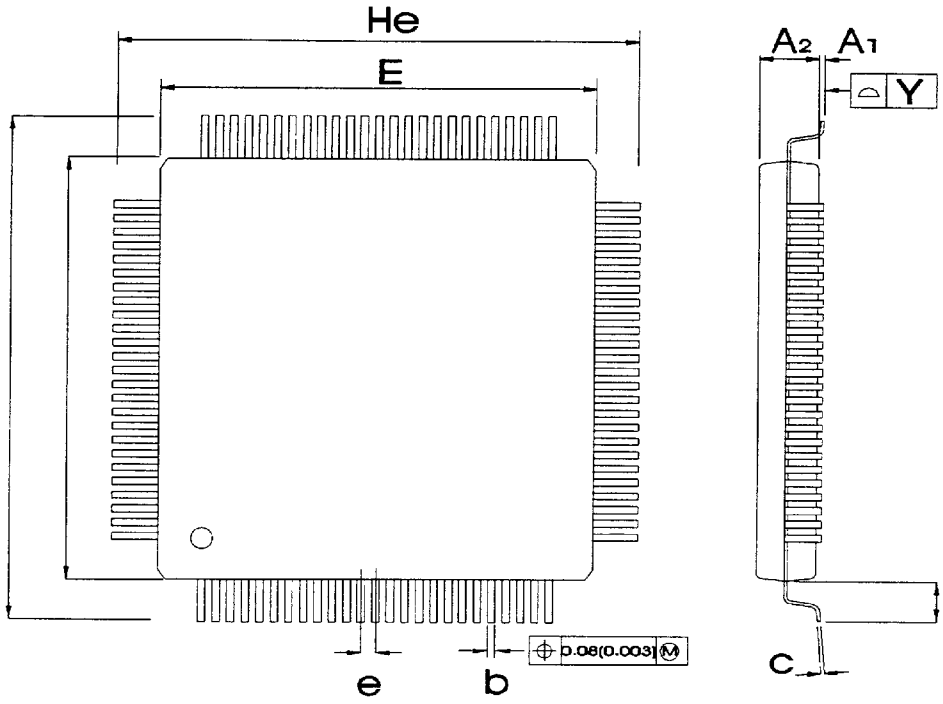
Figure E-11 82C602A 100-Pin Plastic Quad Flat Pack (PQFP)

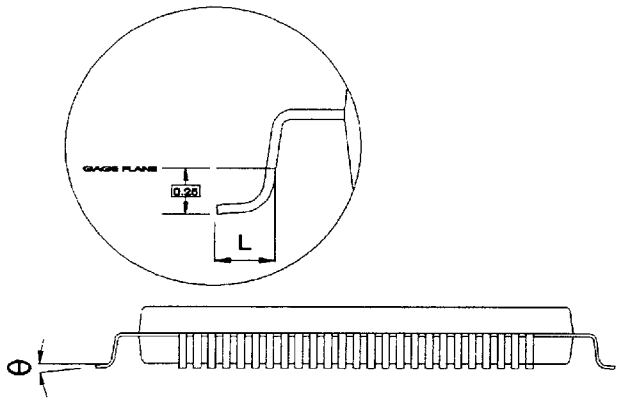


SYMBOL	MILLIMETER MIN.	MILLIMETER NOM.	MILLIMETER MAX.	INCH MIN.	INCH NOM.	INCH MAX.
A1	0.25	0.35	0.45	0.010	0.014	0.018
A2	2.57	2.72	2.87	0.101	0.107	0.113
b	0.20	0.30	0.40	0.008	0.012	0.016
c	0.10	0.15	0.20	0.004	0.006	0.008
D	13.90	14.00	14.10	0.547	0.551	0.555
E	19.90	20.00	20.10	0.783	0.787	0.791
e		0.65			0.026	
Hd	17.00	17.20	17.40	0.669	0.677	0.685
He	23.00	23.20	23.40	0.905	0.913	0.921
L	0.65	0.80	0.95	0.025	0.031	0.037
L1		1.60			0.063	
Y			0.08			0.003
θ	0		7	0		7

Dwg. No.: AS100PQFP-001
Dwg. Rev.: A0
Unit: MM / INCH

Figure E-12 82C602A 100-Pin Thin Quad Pack (TQFP)





SYMBOL	MILLIMETER			INCH		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A_1	0.05	0.10	0.15	0.002	0.004	0.006
A_2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
c	0.090		0.200	0.004		0.008
D	13.90	14.00	14.10	0.547	0.551	0.555
E	13.90	14.00	14.10	0.547	0.551	0.555
e		0.50			0.020	
Hd	15.90	16.00	16.10	0.626	0.630	0.634
He	15.90	16.00	16.10	0.626	0.630	0.634
L	0.45	0.60	0.75	0.018	0.024	0.030
L_1		1.00			0.039	
Y			0.08			0.003
θ	0		7	0		7

Dwg. No.: AS100TQFP-001	
Dwg. Rev.: A0	Unit: MM / INCH