IBM Personal System/2®
Hardware Interface Technical Reference

00F9809

IBM

Personal System/2 Hardware Interface Technical Reference

First Edition (May 1988)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

THE PUBLICATION OF THE INFORMATION CONTAINED HEREIN IS NOT INTENDED TO AND DOES NOT CONVEY ANY RIGHTS OR LICENSES, EXPRESS OR IMPLIED, UNDER ANY IBM PATENTS, COPYRIGHTS, TRADEMARKS, MASK WORKS OR ANY OTHER INTELLECTUAL PROPERTY RIGHTS.

Requests for copies of this publication and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

Preface

This technical reference provides hardware and software interface information specifically for IBM Personal System/2 products. This manual contains both a PS/2™ family overview and system-specific information, and should be used with the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

This manual is divided into the following:

"Micro Channel™ Architecture" describes the Micro Channel signals, timing, electrical characteristics, level-sensitive interrupts, and multidevice arbitration.

"Programmable Option Select" describes adapter setup, system configuration utilities, and adapter description files.

"Micro Channel Adapter Design" provides information such as adapter dimensions, power requirements, and design guidelines.

"Microprocessors and Instruction Sets" describes the various processors used in the Personal System/2 family. Also included is a quick reference for microprocessor assembly instruction sets.

"System Board I/O Controllers" describes the input and output interfaces of the system board controllers.

"Keyboards (101- and 102-Key)" has layouts of the 101- and 102-key keyboards. Keyboard scan-code sets and keyboard specifications are also provided.

"Characters and Keystrokes" supplies the decimal and hexadecimal values for ASCII characters.

"Power Supply" provides the electrical input/output specifications and describes the theory of operations.

Micro Channel and PS/2 are trademarks of the International Business Machines Corporation.

"Compatibility" provides hardware and software information to take into consideration, and provides suggestions to aid you in developing your programs.

System-specific information concerning hardware implementation and performance is also included.

Note: Information added to the system-specific area of this manual may have new information about subjects covered in other parts of this manual. Refer to the system-specific area for information that could affect your hardware or software development decisions.

A Bibliography is also provided.

Technical Reference Library

The technical reference library is intended for those who develop hardware and software products for IBM PS/2 systems and who understand computer architecture and programming concepts.

The technical reference library for Personal System/2 products that incorporate the Micro Channel architecture consists of the following:

- *IBM Personal System/2 Hardware Interface Technical Reference*: provides information to support the functions and architecture common to multiple models of the PS/2 family. In this manual, Type 1 refers to the initial hardware design level. Subsequent levels are designated as Type 2, Type 3, and so on.
- System-specific technical references: provide information concerning hardware implementation and performance for a given model.
- *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*: provides BIOS and Advanced BIOS interface information.
- Option and Adapter Technical References: provide hardware and programming information about individual PS/2 options and adapters.

Suggested Reading:

- *BASIC for the IBM Personal Computer*
- IBM *Disk Operating System (DOS)*
- IBM *Operating System/2*™
- *Macro Assembler for the IBM Personal Computer*
- *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference.*

Additional publications relating to the information contained in this manual are listed in the Bibliography.

Warning: In this technical reference, the term "Reserved" is used to describe certain signals, bits, and registers. Use of reserved areas can cause compatibility problems, loss of data, or permanent damage to the hardware.

When modifying a register, the state of the reserved bits must be preserved. When possible, read the register first and change only the bits required.

Operating System/2 is a trademark of the International Business Machines Corporation.

Notes:

Micro Channel Architecture

Description 1
Channel Definition 2
Signal Descriptions (16-Bit) 4
Signal Descriptions (32-Bit) 11
Signal Descriptions (Matched-Memory) 12
Signal Descriptions (Auxiliary Video Extension) 13
Micro Channel Connector (16-Bit) 15
Micro Channel Connector (Auxiliary Video Extension) 17
Micro Channel Connector (32-Bit Section) 18
Micro Channel Connector (Matched-Memory Extension) 19
Channel Signal Groups (16-Bit, 32-Bit, and Matched-Memory) 19
Channel Signal Groups (Auxiliary Video Extension) 22
Bus Ownership 23
Central Arbitration Control Point 23
Local Arbiters 25
Burst Mode 28
Preemption 29
Programmable Fairness and the Inactive State 30
Arbitration Bus Priority Assignments 30
Channel Support 32
Address Bus Translator 32
Data Bus Steering 32
Level-Sensitive Interrupt Sharing 34
Micro Channel Critical Timing Parameters 36
Basic-Transfer Cycle 36
Simplified Basic-Transfer Cycle 36
I/O and Memory Cycle 39
Default Cycle 40
Default Cycle Return Signals 42
Synchronous Special Case of Extended Cycle 43
Synchronous-Extended Cycle (300 Nanoseconds Minimum - Special Case) 44
Asynchronous-Extended Cycle (General Case) 47
Asynchronous-Extended Cycle (≥300 Nanoseconds Minimum - General Case) 48
DMA Timing 50
First Cycle After Grant 50
Single DMA Transfer (DMA Controller Controlled) 52
Burst DMA Transfer (DMA Controller Terminated) 54

Burst DMA Transfer (DMA Slave Terminated - Default Cycle 200 Nanoseconds) 56
Burst DMA Transfer (DMA Slave Terminated - Synchronous-Extended Cycle 300 Nanoseconds) 58
Burst DMA Transfer (DMA Slave Terminated - Asynchronous-Extended Cycle ≥300 Nanoseconds) 60
Arbitration Timing 62
Arbitration Cycle 62
Exiting from Inactive State 62
Configuration Timing 64
Additional Channel Timings 66
Auxiliary Video Extension Timing 67

Index 69

Figures

1. Micro Channel Connectors 3
2. I/O and Memory Transfer Controls 6
3. Channel Connector Voltage and Signal Assignments (8-Bit Section) 16
4. Channel Connector Voltage and Signal Assignments (16-Bit Section) 17
5. Auxiliary Video Extension 17
6. Channel Connector Voltage and Signal Assignments (32-Bit Section) 18
7. Channel Connector Voltage and Signal Assignments (Matched-Memory Extension) 19
8. Driver/Receiver Requirements and Options 20
9. Signal Driver Types 21
10. Distributed Arbitration Block Diagram 25
11. Local Arbiter Example 27
12. Burst Mode Timing 28
13. Preempt Timing 29
14. Arbitration Bus Priority Assignments for Compatibility . . . 31
15. Steering Control 32
16. Data Bus Steering Implementation 33
17. Typical Adapter Interrupt Sharing Implementation 34
18. Interrupt Sharing Sequence 35
19. Overview of the Basic-Transfer Cycle 38
20. Default I/O and Memory Cycle (200 Nanoseconds Minimum) 41
21. Default Cycle Return Signals (200 Nanoseconds Minimum) 42
22. Timing Sequence for the Synchronous Special Case of Extended Cycle 43
23. Synchronous-Extended Cycle (300 Nanoseconds Minimum - Special Case) 45
24. Timing Sequence for the Asynchronous-Extended Cycle (General Case) 47
25. Asynchronous-Extended Cycle (≥300 Nanoseconds Minimum - General Case) 49
26. First Cycle After Grant 51
27. Single DMA Transfer (DMA Controller Controlled) 53
28. Burst DMA Transfer (DMA Controller Terminated) 55
29. Burst DMA Transfer (DMA Slave Terminated - Default Cycle 200 Nanoseconds) 57

30. Burst DMA Transfer (DMA Slave Terminated - Synchronous-Extended Cycle 300 Nanoseconds) 59
31. Burst DMA Transfer (DMA Slave Terminated - Asynchronous-Extended Cycle ≥300 Nanoseconds) 61
32. Arbitration Cycle 63
33. Setup Cycle 65
34. Additional Channel Timings 66
35. Auxiliary Video Connector Timing (DAC Signals) 67

Description

The Micro Channel architecture consists of an address bus, a data bus, a transfer control bus, an arbitration bus, and multiple support signals. It uses synchronous and asynchronous procedures for control and data transfer between memory, I/O devices, and the controlling master. The controlling master can be a DMA controller, the system microprocessor, or a bus master. See "Bus Ownership" on page 23 for more information about controlling masters.

The following are characteristics of the Micro Channel architecture:

- An I/O address width of 16 bits allows 8-, 16-, or 32-bit I/O transfers within a 64KB range. A memory address width of 24 and 32 bits allows 8-, 16-, 24-, or 32-bit memory transfers within 16MB and 4GB ranges respectively (KB equals 1024 bytes; MB equals 1,048,576 bytes; GB equals 1,073,741,824 bytes).
- Support of a central arbitration control point that enables up to 15 devices and the system microprocessor to arbitrate for control of the channel.
- A direct memory access (DMA) procedure that supports multiple DMA channels with burst capability.
- Level-sensitive interrupts with interrupt sharing on all levels.
- Programmable Option Select (POS) registers that replace hardware jumpers and switches. These registers allow flexibility during system configuration.
- Channel connector extensions that support the growth of additional channel features.
- Improved electromagnetic compatibility.
- Error reporting.

Channel Definition

The channel provides all signal, power, and ground signals to adapters through 50-mil channel connectors.

The channel provides two basic types of connectors:

- 16-bit, which support 8- and 16-bit operations
- 32-bit, which support 8-, 16-, 24-, and 32-bit operations.

Pins 01 through 45 support 8-bit operations. Pins 46 and 47 are keys. Pins 48 through 58 provide additional power and signals to support 16-bit operations. Pins 59 through 89 are used with pins 01 through 58 to support 32-bit operations and are only present on systems with a 32-bit system microprocessor.

Side A of each connector is offset from side B by 2 pins, and every fourth pin on either side of each connector is at ac ground. This places each signal within 2.54 millimeters (0.1 inch) of a ground and minimizes current loop electromagnetic interference (EMI). The 50-mil connector reduces the required insertion force and matches the line spacing of surface mount technology.

Extensions to the basic 16- and 32-bit connectors are implemented on a system-by-system basis. Refer to the system-specific technical references for additions to or deviations from the information presented in this section.

Note: Adapter designs should not extend the card-edge connector beyond the basic 16- or 32-bit connector unless the signals provided by the extension are used by the adapter. Adapters for the Micro Channel architecture have special design criteria. See "Micro Channel Adapter Design."

The following is a diagram of the two basic types of channel connectors with optional extensions.

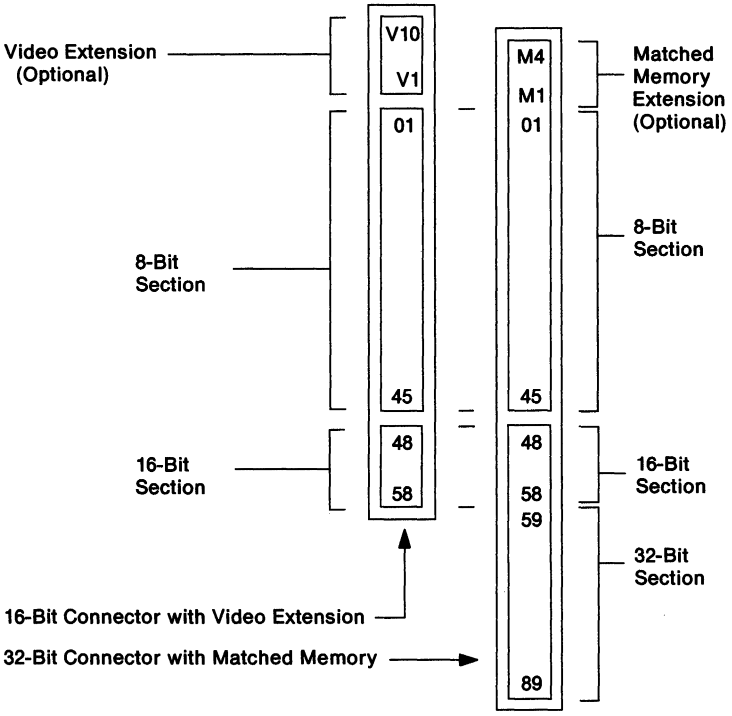

Video Extension (Optional)
8-Bit Section
16-Bit Section
V10
V1
01
45
48
58
16-Bit Connector with Video Extension
32-Bit Connector with Matched Memory
M4
M1
01
45
48
58
59
89
Matched Memory Extension (Optional)
8-Bit Section
16-Bit Section
32-Bit Section

Figure 1. Micro Channel Connectors

Warning: Any signals shown or described as "Reserved" should not be driven or received. These signals are reserved to allow compatibility with future implementations of the channel interface. Serious compatibility problems, loss of data, or permanent damage can result to features or the system if these signals are misused.

Signal Descriptions (16-Bit)

All of the logic signal lines are transistor-transistor logic (TTL) compatible. The following are the signals available on the channel. Timing information for the signals begins on page 36.

Reserved: Any signals shown or described as "Reserved" should not be driven or received. These signals are reserved to allow compatibility with future implementations of the channel interface. Serious compatibility problems, loss of data, or permanent damage can result to features or the system if these signals are misused.

A0 – A23: Address Bits 0 through 23: These lines are used to address memory and I/O slaves attached to the channel. A0 is the least-significant bit (LSB) and A23 is the most-significant bit (MSB). These 24 address lines allow access of up to 16MB of memory. Only the lower 16 address lines (A0 – A15) are for I/O operations, and all 16 lines must be decoded by the I/O slave. A0 through A23 are generated by the controlling master. Valid addresses generated by the controlling master are unlatched on the channel and, if required, must be latched by the slaves using either the leading or trailing edge of the '-address decode latch' signal (-ADL) or the leading edge of the 'command' signal (-CMD). A0 through A23 must be driven with tri-state drivers.

D0 – D15: Data Bits 0 through 15: These lines provide data bus bits 0 through 7 (low byte) and 8 through 15 (high byte) for the controlling master and slaves. D0 is the LSB and D15 the MSB. All 8-bit slaves on the channel must use D0 through D7 to communicate with the controlling master. During read cycles, data is valid on these lines after the leading edge but before the trailing edge of -CMD, and must remain valid until after the trailing edge of -CMD. However, during write cycles, data is valid as long as -CMD is active. D0 through D15 must be driven with tri-state drivers.

-ADL: -Address Decode Latch: This line, driven by the controlling master, is provided as a convenient way for the slave to latch valid addresses and status bits. This signal can be used by slaves to latch the address from the bus. -ADL is not active during matched-memory cycles. -ADL is driven with a tri-state driver.

-CD DS 16 (n): -Card Data Size 16: This line is driven by 16-bit and 32-bit memory, I/O, or DMA slaves to provide an indication on the channel of a 16-bit or 32-bit data port at the location addressed. The (n) indicates this signal line is unique to each channel connector (one independent signal line per connector). This signal is unlatched and derived as a valid address decode. All system logic receives this signal to support communication with 16- and 32-bit slaves. -CD DS 16 is not driven by 8-bit slaves. All 16- and 32-bit slaves must drive this signal. -CD DS 16 is driven with a totem-pole driver.

-DS 16 RTN: -Data Size 16 Return: This output signal is a negative OR of the -CD DS 16 signal from each channel connector. If any device drives its -CD DS 16 active, this output is active. This signal is provided to allow the controlling master to monitor the data size information. -DS 16 RTN must be driven with a bus driver.

-SBHE: -System Byte High Enable: This line indicates and enables transfer of data on the high byte of the data bus (D8 – D15), and is used with A0 to distinguish between high-byte transfers (D8 – D15) and low-byte transfers (D0 – D7). All 16-bit slaves decode this line, but 8-bit slaves do not. -SBHE is driven with a tri-state driver.

MADE 24: Memory Address Enable 24: This line indicates when an extended address is used on the bus. If a memory cycle is in progress and MADE 24 is inactive, an extended address greater than 16MB is being presented; if MADE 24 is active, an unextended address less than or equal to 16MB is being presented. This line is driven by the controlling master and decoded by all memory slaves, regardless of their address space size. MADE 24 is driven with a tri-state driver.

M/-IO: Memory/-Input Output: This signal distinguishes a memory cycle from an I/O cycle. When this signal is high, a memory cycle is in progress. When M/-IO is low, an I/O cycle is in progress. M/-IO is driven with a tri-state driver.

-S0, -S1: -Status Bits 0 and 1: These lines indicate the start of a channel cycle and also define the type of channel cycle. When used with M/-IO, memory read/write operations are distinguished from I/O read/write operations. These signals are latched by the slave, as required, using the leading edge of -CMD or the trailing edge of -ADL. -S0 and -S1 are driven with a tri-state driver.

Data is moved to or from the bus based on -CMD and a latched decode of the address, the status lines (-S0 exclusive or -S1), and M/-IO.

Slaves must support a full decode of -S0 and -S1. The following figure shows the proper states of M/-IO, -S0, and -S1 in decoding I/O and memory read/write commands.

M/-IO	-S0	-S1	Function
0	0	0	Reserved
0	0	1	I/O Write Command
0	1	0	I/O Read Command
0	1	1	Reserved
1	0	0	Reserved
1	0	1	Memory Write Command
1	1	0	Memory Read Command
1	1	1	Reserved

Figure 2. I/O and Memory Transfer Controls

An I/O Write command instructs an I/O slave to store the data on the data bus. The data must be valid on the bus from the leading edge of -CMD and must be held on the bus until after -CMD goes inactive. Addresses on the bus must be valid before -S0 goes active.

An I/O Read command instructs an I/O slave to drive its data onto the data bus. The data must be placed on the bus following the leading edge of -CMD, must be valid before the trailing edge of -CMD, and must be held on the bus until -CMD goes inactive. Addresses on the bus must be valid before -S1 goes active.

A memory Write command instructs the memory to read the data on the data bus. The data must be valid on the bus from the leading edge of -CMD and must be held on the bus until after -CMD goes inactive. Addresses on the bus must be valid before -S0 goes active.

A memory Read command instructs the memory to drive its data onto the data bus. The data must be placed on the bus following the leading edge of -CMD. The data must be valid before the trailing edge of -CMD, and must be held on the bus until -CMD goes inactive. Addresses on the bus must be valid before -S1 goes active.

-CMD: -Command: This signal is used to define when data is valid on the data bus. The trailing edge of this signal indicates the end of the bus cycle. This signal indicates to the slave how long data is valid on the bus. During write operations, the data is valid on the bus as long as -CMD is active. During read operations, the data is valid on the bus between the leading and trailing edges of -CMD, and must be held on the bus until after -CMD goes inactive. This signal can be used by the slaves to latch the address on the bus. Latched status lines gated by -CMD provide the timing control of valid data. Slaves should use transparent latches to latch address and status information with the leading edge of -CMD. -CMD is not active during matched-memory cycles. It must be driven with a tri-state driver.

-CD SFDBK (n): -Card Selected Feedback: When the controlling master addresses a memory slave or an I/O slave, the addressed slave drives -CD SFDBK active as a positive acknowledgment of its presence at the address specified. The (n) indicates this signal line is unique to each channel connector (one independent signal line per connector). This signal is unlatched by any slave with a valid select decode, and is driven by any slave selected by any select mechanism except -CD SETUP. The slave does not drive -CD SFDBK during the configuration cycle. -CD SFDBK is driven with a totem-pole driver.

Note: Memory supporting diagnostic software must not drive -CD SFDBK during the diagnostic operation.

CD CHRDY (n): Channel Ready: This line, normally active (ready), is pulled inactive (not ready) by a memory or I/O slave to allow additional time to complete a channel operation. The (n) indicates this signal line is unique to each channel connector (one independent signal line per connector). During a read operation, a slave ensures that data will be valid on the data bus within the time specified after releasing the line to a ready state. The slave also holds the data long enough for the controlling master to sample. A slave may also use this line during a write operation if more time is needed to store the data from the bus. This signal is derived with a valid address decode ANDed with status. CD CHRDY is driven with a totem-pole driver.

CHRDYRTN: Channel Ready Return: This output signal is a positive AND of the CD CHRDY signals. If all devices drive CD CHRDY active, this output is active. It is provided to allow the controlling master to monitor the ready information. CHRDYRTN must be driven with a bus driver.

ARB0 – ARB3: Arbitration Bus Priority Levels: These lines comprise the arbitration bus and are used to present priority levels for participants seeking control of the bus. ARB0 through ARB3, the least-significant through most-significant bits respectively, support up to 16 priority levels.

The highest hexadecimal value of the arbitration bus (hex F) has the lowest priority, and the lowest value (hex 0) has the highest priority. A participant is allowed to change the state of the arbitration bus only immediately after the rising edge of ARB/-GNT. All participants monitor the arbitration bus and the lower priority participants withdraw their priority levels by not activating less-significant arbitration bits.

The hexadecimal code of the highest priority requester is valid on the arbitration bus after a settling time. After the channel is granted to a requester, the highest priority participant continues to drive its priority lines. These bidirectional lines are active high and must be driven with open collector drivers.

ARB/-GNT: Arbitrate/-Grant: When high, this signal indicates an arbitration cycle is in process. When low, it is the acknowledgment from the central arbitration control point to an arbitrating bus participant (local arbiter) and the DMA controller that channel control has been granted. This signal is driven high by the central arbitration control point within a specified time after -S0, -S1, -BURST, and -CMD become inactive. The negative-to-positive transition of ARB/-GNT initiates an arbitration cycle; the positive-to-negative transition terminates the arbitration cycle. Only the central arbitration control point activates and deactivates this line. This signal must be used by all local arbiters to gate their address data and transfer control bus drivers off during arbitration cycles. ARB/-GNT is driven with a bus driver.

-PREEMPT: -Preempt: This signal is used by arbitrating bus participants (local arbiters) to request use of the channel through arbitration. Any local arbiter with a channel request activates -PREEMPT and causes an arbitration cycle to occur. A local arbiter removes its -PREEMPT upon being granted the channel. This bidirectional line must be driven with an open collector driver.

-BURST: -Burst: This signal indicates to the central arbitration control point the extended use of the channel for transferring a block of data. This type of data transfer is called a *burst cycle*. This line is shared by all local arbiters. -BURST is driven active by the local arbiter after being granted the channel. The local arbiter must deactivate -BURST during the last transfer cycle. -BURST must be driven with an open collector driver.

-TC: -Terminal Count: This line provides a pulse during a Read or Write command to indicate that the terminal count of the current DMA channel has been reached. This indicates to the DMA slave the last cycle to be performed of a preprogrammed DMA block transfer. -TC is available on the channel only during DMA operations. It is driven with a tri-state driver by the DMA controller.

-IRQ 3 − 7, -IRQ 9 − 12, and -IRQ 14 − 15: -Interrupt Request: These lines are used to signal that a device requires attention. They are prioritized with -IRQ 9 having the highest priority and -IRQ 7 having the lowest priority. The effective interrupt priority sequence is -IRQ (9 − 12, 14, 15, 3 − 7). An interrupt request is generated when a slave drives one of the 'interrupt request' signals low. The polarity of 'interrupt request' signals makes it possible for multiple slaves to share the same interrupt level. This is called *interrupt sharing*. These lines must be driven with an open collector driver.

-CD SETUP (n): -Card Setup: This signal is driven by system logic to individually select channel connectors during system configuration and error-recovery procedures. The (n) indicates this signal line is unique to each channel connector (one independent signal line per connector). When this signal is activated, a specific channel connector is selected and access to the adapter's configuration data space is obtained. The ID and configuration data can be obtained by an I/O read operation; the configuration data is stored by an I/O write operation. Each channel connector has a unique -CD SETUP. This line is driven with a totem-pole driver.

-CHCK: -Channel Check: This line is used to indicate a serious error (such as a parity error) that threatens continued operation of the system. -CHCK is driven active to indicate the error condition and must remain active until the -CHCK interrupt handler resets it. -CHCK is driven with an open collector driver to allow sharing.

AUDIO: Audio Sum Node: This line is an audio voltage sum node. It is used to drive audio signals from an adapter to the system audio output, or to transfer audio signals between adapters. The frequency response of the audio line is 50 Hz to 10 kHz ± 3 dB. The maximum signal amplitude is 2.5 Vac peak-to-peak, at a dc offset of 0 ± 50 millivolts. The noise level is limited to a maximum of 50 millivolts peak-to-peak.

AUDIO GND: Audio Ground: This is a separate ground return for the audio subsystem.

OSC: Oscillator: This line is a high-speed clock with a frequency of 14.31818 MHz ± 0.01%. The high-level pulse width (more than 2.3 Vdc) and the low-level pulse width (less than 0.8 Vdc) must not be less than 20 nanoseconds each.

CHRESET: Channel Reset: This signal is generated by the system logic to reset or initialize all adapters at power on or during a low line voltage condition. During a power-on sequence, CHRESET is active for a specified minimum time. The system can also activate this signal under program control. CHRESET is driven with a bus driver.

-REFRESH: -Refresh: This line is driven by the system logic and is used to indicate that a memory refresh operation is in progress. While this line is active, a memory read operation occurs. The address lines contain the memory locations being refreshed. Nine lines, A0 through A8, are activated. -REFRESH timing may be inconsistent and must not be used as a timing mechanism. -REFRESH is driven with a tri-state driver.

Signal Descriptions (32-Bit)

A24 – A31: Address Bits 24 through 31: These lines are used with A0 through A23 to address memory attached to the channel. A0 is the LSB and A31 is the MSB. These 32 address lines allow access of up to 4GB of memory. Only the lower 16 address lines (A0 through A15) are used for I/O operations. A24 through A31 are generated by the controlling master. Valid addresses generated by the controlling master are unlatched on the channel and, if required, must be latched by the slaves using either the leading or trailing edge of -ADL or the leading edge of -CMD. A0 through A31 must be driven with tri-state drivers.

-BE0 – -BE3: -Byte Enable 0 through 3: These lines are used during data transfers with 32-bit slaves to indicate which data bytes will be placed on the bus. Data transfers of 8, 16, 24, or 32 contiguous bits are controlled by -BE0 through -BE3 during transfers involving 32-bit slaves only. These lines are driven by the controlling master when TR 32 is inactive, and by the Central Translator Logic (for those operations involving a 16-bit master with a 32-bit slave) when TR 32 is active. These lines are unlatched on the bus and, if required, must be latched by 32-bit slaves. -BE0 through -BE3 are driven with tri-state drivers.

D16 – D31: Data Bits 16 through 31: These lines are used with D0 through D15 to provide data-bus bits to the controlling master and slaves. D0 is the LSB and D31 the MSB. All 32-bit transfers from the controlling master to 8-bit slaves are converted to four 8-bit transfers, and all are transmitted on lines D0 through D7. All 32-bit transfers from the controlling master to 16-bit slaves are converted to two 16-bit transfers, and all are transmitted on lines D0 through D15. During read cycles, data is valid on these lines after the leading edge of -CMD but before the trailing edge of -CMD. However, during write cycles, data is valid as long as -CMD is active. D0 through D31 must be driven with tri-state drivers.

-CD DS 32 (n): -Card Data Size 32: This line is driven by 32-bit slaves to provide an indication on the bus of a 32-bit data port at the location addressed. The (n) indicates this signal line is unique to a channel connector position (one independent signal per connector). -CD DS 32 is unlatched and derived from a valid address decode. All 32-bit slaves must drive this signal. -CD DS 32 is inactive for an 8- or 16-bit data port. -CD DS 32 must be driven with a totem-pole driver.

-DS 32 RTN: -Data Size 32 Return: This output signal is a negative OR of the -CD DS 32 signal from each channel connector. If any device drives its -CD DS 32 active, then this output is active. This signal is provided to allow controlling masters to monitor data size information. -DS 32 RTN must be driven with a bus driver.

TR 32: Translate 32: This line is driven inactive by 32-bit controlling masters and received by the Central Translator Logic. TR 32 can also be received by any 32-bit slave. When TR 32 is inactive, a 32-bit controlling master drives -BE0 through -BE3. When TR 32 is active, the Central Translator Logic drives -BE0 through -BE3. TR 32 must be driven by a tri-state driver. See "Channel Support" on page 32 for more information about Central Translator Logic.

Signal Descriptions (Matched-Memory)

Matched memory is a function that allows enhanced data transfer capabilities between the system microprocessor and its channel-resident memory. Matched memory is not supported by all systems. The following signal definitions are system specific and may not apply to all systems that support matched-memory cycles. For more information, refer to the system-specific technical reference for the system you are dealing with.

-MMC: -Matched Memory Cycle: This signal is driven by the system logic to indicate to the channel slaves that the system microprocessor is the controlling master and is able to run a matched-memory cycle.

-MMCR: -Matched Memory Cycle Request: This is a bus cycle control-input signal. -MMCR is driven by a 16- or 32-bit channel slave to request the faster cycle available on the system bus.

-MMC CMD: -Matched Memory Cycle Command: This output signal to the bus is generated for system microprocessor bus cycles only. -MMC CMD defines when data is valid on the bus during a matched-memory cycle.

Note: Adapter designs should not extend the card-edge connector beyond the basic 16- or 32-bit connector unless the signals provided by the extension are used by the adapter. Adapters for the Micro Channel architecture have special design criteria. See “Micro Channel Adapter Design.”

Signal Descriptions (Auxiliary Video Extension)

The following are signal descriptions for the auxiliary video extension of the channel connector.

VSYNC: Vertical Synchronization: This signal is the vertical synchronization signal to the display. See also the ESYNC description.

HSYNC: Horizontal Synchronization: This signal is the horizontal synchronization signal to the display. See also the ESYNC description.

BLANK: Blanking Signal: This signal is connected to the BLANK input of the video digital-to-analog converter (DAC). When active (0 Vdc), this signal tells the DAC to drive its analog color outputs to 0 Vdc. See also the ESYNC description.

P0 - P7: Palette Bits: These eight signals contain video information and comprise the picture element (PEL) address inputs to the video DAC. See also the EVIDEO description.

DCLK: Dot Clock: This signal is the PEL clock used by the DAC to latch the digital video signals, P7 through P0. The signals are latched into the DAC on the rising edge of DCLK.

This signal is driven through the EXTCLK input to the system board video when DCLK is driven by the adapter. If an adapter is providing the clock, it must also provide the video data to the DAC. See the EDCLK description.

ESYNC: External Synchronization: This signal is the output-enable signal for the buffer that drives BLANK, VSYNC, and HSYNC. ESYNC is tied to +5 Vdc through a pull-up resistor. When ESYNC is high, the system board video drives BLANK, VSYNC, and HSYNC. When ESYNC is pulled low, the adapter drives BLANK, VSYNC, and HSYNC.

EVIDEO: External Video: This signal is the output-enable signal for the buffer that drives P7 through P0. EVIDEO is tied to +5 Vdc through a pull-up resistor. When EVIDEO is high, the system board video drives P7 through P0. When it is pulled low, the adapter drives P7 through P0.

EDCLK: External Dot Clock: This signal is the output-enable signal for the buffer that drives DCLK. EDCLK is tied to +5 Vdc through a pull-up resistor. When EDCLK is high, the system board video is the source of DCLK to the DAC and the adapter. When EDCLK is pulled low, the adapter drives DCLK.

See "Video Subsystem" for more information.

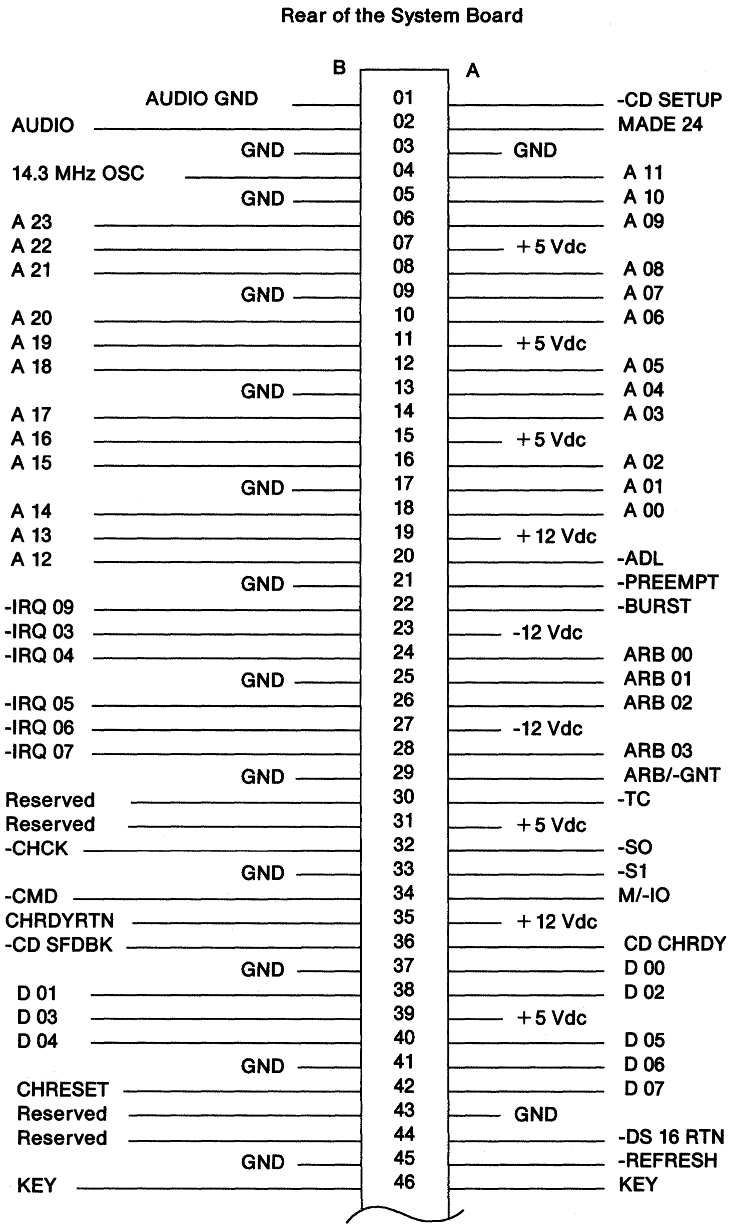
Micro Channel Connector (16-Bit)

The 16-bit Micro Channel connector has two sections:

- An 8-bit section
- A 16-bit section.

A key between the two sections is provided for mechanical alignment.

The following figures show the signals and voltages assigned to the 16-bit channel connector.



Rear of the System Board

B		A
AUDIO GND	01	-CD SETUP
AUDIO	02	MADE 24
GND	03	GND
14.3 MHz OSC	04	A 11
GND	05	A 10
A 23	06	A 09
A 22	07	+5 Vdc
A 21	08	A 08
GND	09	A 07
A 20	10	A 06
A 19	11	+5 Vdc
A 18	12	A 05
GND	13	A 04
A 17	14	A 03
A 16	15	+5 Vdc
A 15	16	A 02
GND	17	A 01
A 14	18	A 00
A 13	19	+12 Vdc
A 12	20	-ADL
GND	21	-PREEMPT
-IRQ 09	22	-BURST
-IRQ 03	23	-12 Vdc
-IRQ 04	24	ARB 00
GND	25	ARB 01
-IRQ 05	26	ARB 02
-IRQ 06	27	-12 Vdc
-IRQ 07	28	ARB 03
GND	29	ARB/-GNT
Reserved	30	-TC
Reserved	31	+5 Vdc
-CHCK	32	-SO
GND	33	-S1
-CMD	34	M/-IO
CHRDYRTN	35	+12 Vdc
-CD SFDBK	36	CD CHRDY
GND	37	D 00
D 01	38	D 02
D 03	39	+5 Vdc
D 04	40	D 05
GND	41	D 06
CHRESET	42	D 07
Reserved	43	GND
Reserved	44	-DS 16 RTN
GND	45	-REFRESH
KEY	46	KEY

Figure 3. Channel Connector Voltage and Signal Assignments (8-Bit Section)

The following are the signals and voltages assigned to the channel connector 16-bit section.

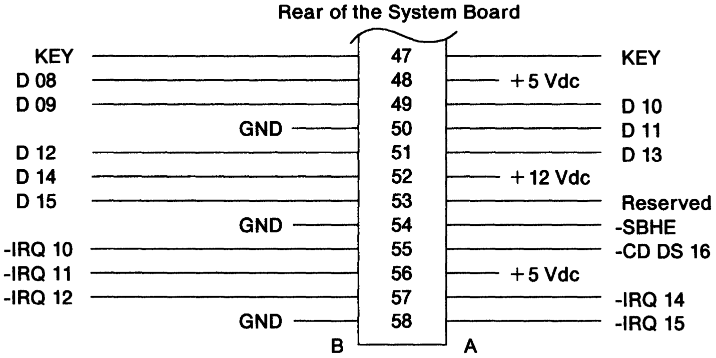


Figure 4. Channel Connector Voltage and Signal Assignments (16-Bit Section)

Micro Channel Connector (Auxiliary Video Extension)

This extension to the Micro Channel connector accommodates video adapters that interface with the system-board video subsystem.

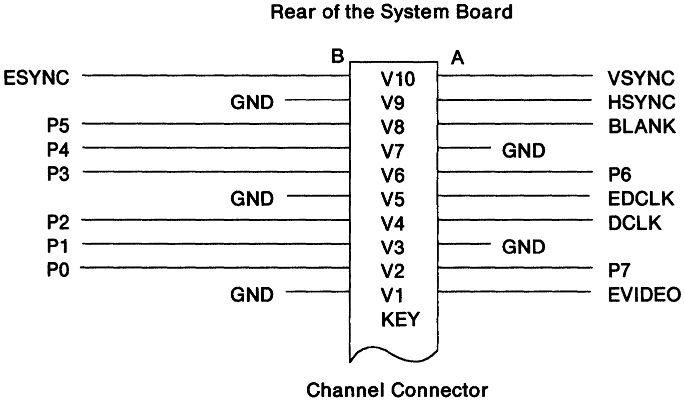
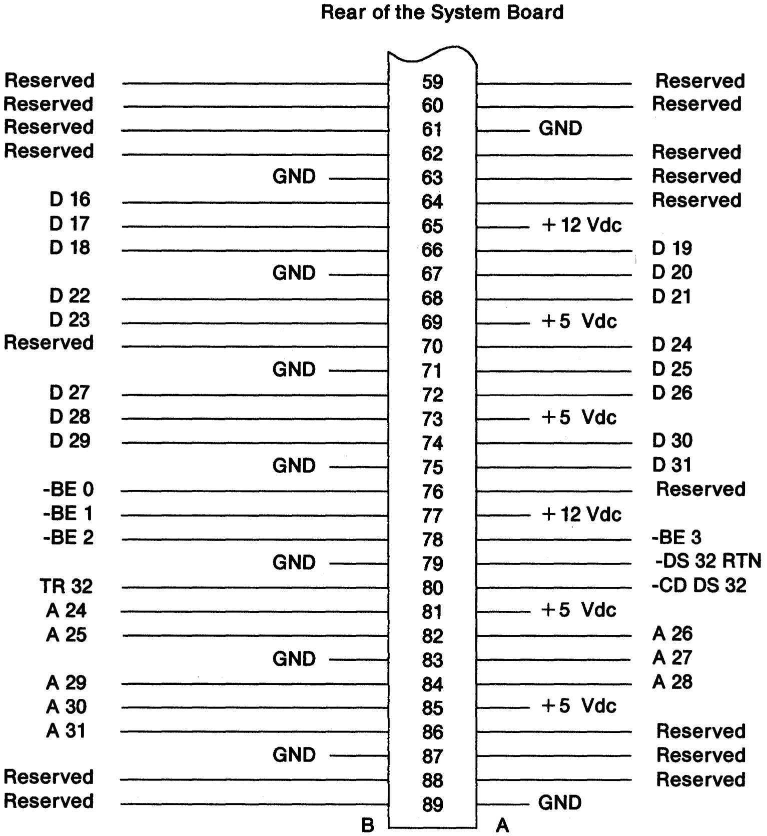


Figure 5. Auxiliary Video Extension

Micro Channel Connector (32-Bit Section)

This connector extends the 16-bit Micro Channel connector to accommodate 32-bit addressing and 32-bit data transfers.



B	Pin	A
Reserved	59	Reserved
Reserved	60	Reserved
Reserved	61	GND
Reserved	62	Reserved
GND	63	Reserved
D 16	64	Reserved
D 17	65	+12 Vdc
D 18	66	D 19
GND	67	D 20
D 22	68	D 21
D 23	69	+5 Vdc
Reserved	70	D 24
GND	71	D 25
D 27	72	D 26
D 28	73	+5 Vdc
D 29	74	D 30
GND	75	D 31
-BE 0	76	Reserved
-BE 1	77	+12 Vdc
-BE 2	78	-BE 3
GND	79	-DS 32 RTN
TR 32	80	-CD DS 32
A 24	81	+5 Vdc
A 25	82	A 26
GND	83	A 27
A 29	84	A 28
A 30	85	+5 Vdc
A 31	86	Reserved
GND	87	Reserved
Reserved	88	Reserved
Reserved	89	GND

Figure 6. Channel Connector Voltage and Signal Assignments (32-Bit Section)

Micro Channel Connector (Matched-Memory Extension)

This extension provides additional signals to accommodate matched-memory cycles. The following figure shows a connector with a typical set of matched-memory signals. Refer to the system-specific technical reference for the system you are dealing with for further information.

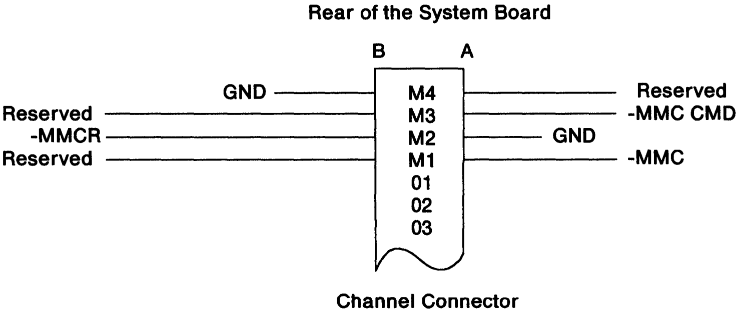


Figure 7. Channel Connector Voltage and Signal Assignments (Matched-Memory Extension)

Channel Signal Groups (16-Bit, 32-Bit, and Matched-Memory)

The following figure lists digital 16-bit, 32-bit, and matched-memory Micro Channel signals and shows what type of driver or receiver is required to be compatible with the system board. The 'audio' and 'audio ground' signals are analog signals; for further information about these signals refer to page 10.

Signal Name	Sys Logic	DMA Cntlr	Bus Master	DMA Slave	MEM Slave	I/O Slave	Driver Type
	D/R	D/R	D/R	D/R	D/R	D/R	Signal Group
A(0-23)	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
D(0-15)	D/R	D/R	D/R	D/R	D/R	D/R	TS (2)
-ADL	D/-	D/-	D/-	-/O	-/O	-/O	TS (1)
-CD DS 16 (n)	-/R	-/R	-/R	#/-	#/-	#/-	TP (3)
-DS 16 RTN	D/-	-/R	-/R	-/-	-/-	-/-	BD (4)
-SBHE	D/-	D/-	D/-	-/#	-/#	-/#	TS (1)
MADE 24	D/-	D/-	D/-	-/-	-/R	-/-	TS (1)
M/-IO	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
-S0,-S1	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
-CMD	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
-CD SFDBK (n)	-/R	-/-	-/-	D/-	D/-	D/-	TP (3)
CD CHRDY (n)	O/R	-/-	-/-	O/-	O/-	O/-	TP (3)
CHRDYRTN	D/-	-/R	-/R	-/-	-/-	-/-	BD (4)
ARB(0-3)	O/R	-/R	D/R	D/R	-/-	-/-	OC (5)
-BURST	O/R	D/R	D/-	#/-	-/-	-/-	OC (5)
-PREEMPT	O/R	-/-	D/R	D/#	-/-	-/-	OC (5)
ARB/-GNT	D/-	-/R	-/R	-/R	-/-	-/-	BD (4)
-TC	-/-	D/-	-/-	-/O	-/-	-/-	TS (1)
-IRQ (*)	-/R	O/-	O/-	O/-	-/-	O/-	OC (6)
-CD SETUP (n)	D/-	-/#	-/R	-/R	-/R	-/R	TP (7)
-CHCK	-/R	#/-	D/O	D/-	D/-	D/-	OC (6)
-REFRESH	D/R	-/O	-/O	-/O	-/R	-/O	TS (1)
OSC	D/-	-/O	-/O	-/O	-/O	-/O	CD (7)
CHRESET	D/-	-/O	-/R	-/R	-/R	-/R	BD (4)
A(24-31)	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
-BE(0-3)	D/-	D/-	D/-	-/R	-/R	-/R	TS (1)
D(16-31)	D/R	D/R	D/R	D/R	D/R	D/R	TS (2)
-CD DS 32 (n)	-/R	-/R	-/R	#/-	#/-	#/-	TP (3)
-DS 32 RTN	D/-	-/-	-/O	-/-	-/-	-/-	BD (4)
TR 32	-/-	-/-	-/O	-/-	O/-	O/-	TS (1)
-MMC	D/-	-/-	-/-	-/-	-/O	-/O	**
-MMCR	-/R	-/-	-/-	-/-	O/-	O/-	**
-MMC CMD	D/-	-/-	-/-	-/-	-/O	-/O	**

KEY

D = Drive Enabled	OC = Open Collector
O = Optional	TS = Tri-State
R = Receive Enabled	TP = Totem-Pole
- = Not Implemented	BD = Bus Driver
# = Some are Required	CD = Clock Driver

* IRQ (9-12, 14, 15, 3-7)

** See the system-specific technical references for more information.

Figure 8. Driver/Receiver Requirements and Options

The following figure describes the signal driver types.

Signal Group	Driver Type
1, 2	Tri-state (TS) with 24 mA sinking capacity.
3	Totem-pole (TP) with current sinking capacity of 6 mA.
4	Bus driver (BD) with current sinking capacity of 24 mA.
5, 6	Open collector (OC) with 24 mA sinking capacity.
7	Unique drivers: Totem-pole (TP) or Tri-state (TS) with current sinking capacity of 6 mA. Clock driver (CD) with 24 mA sinking capacity.

Figure 9. Signal Driver Types

The following notes apply to the driver and receiver options listed on page 20.

Notes:

1. During the Reset state, an active CHRESET must degate all bus drivers.
2. During the Reset state, the state of all signals is unknown.
3. -CD SETUP is driven to only one channel connector at a time.
4. All pull-up resistors are provided by the system logic and pulled up to +5 Vdc.
5. Loading Current: A maximum of 1.6 milliamperes per channel connector, except signal group 5. The maximum loading current of group 5 is 1.0 milliamperes per channel connector.
6. Loading Capacitance (average capacitance across a 0.1- to 2.3-volt interval):
   - 15 pF maximum permitted for adapter for OSC and Group 5 signals (see Figure 9 for Group 5 definition) and 20 pF maximum permitted for adapter for all other signals. (The value refers to the capacitance from the adapter side of the connector to the adapter driver/receiver.)
   - Total capacitance seen by the driver is 200 pF maximum for Group 5 signals and OSC, and 240 pF maximum for all other signals.

7. An open collector can be either an open-collector device or a tri-state device wired with the input grounded and using the 'enable' line to control the output.

8. The electromagnetic interference (EMI) potential of a bus driver increases as the transition time of its voltage decreases. Therefore, the drivers with output transitions greater than 1 Vdc per nanosecond should be used only to meet channel timing requirements. The figure on page 20 lists the role of the driver or receiver while a given operation is being performed. The names of the physical packaging of the logic should not be confused with the performed functions.

Channel Signal Groups (Auxiliary Video Extension)

An adapter using the auxiliary video extension must not exceed the following loading limits for any auxiliary video-extension signal pin it is receiving:

- C max = 15.0 pF
- I_{IL} min = −1.6 mA
- I_{IH} max = 50.0 µA.

An adapter using the auxiliary video extension must meet the following minimum requirements for any auxiliary video-extension signal pin it is driving:

- C min = 150.0 pF
- I_{OL} min = 10.0 mA - VSYNC and HSYNC
  = 2.0 mA - All other signals
- I_{OH} max = −4.0 mA - VSYNC and HSYNC
  = −0.25 mA - All other signals.

Bus Ownership

Bus ownership is controlled by the central arbitration control point based on prioritized arbitration of up to 16 devices. These arbitrating devices can be DMA slaves, bus masters, or the system microprocessor. If either a bus master or the system microprocessor wins the arbitration, it owns the bus and becomes the controlling master. If a DMA slave wins the arbitration, the supporting DMA controller owns the bus and becomes the controlling master.

An adapter can incorporate either a master function, a slave function, or a combination of both. For example, an adapter might be designed to operate primarily as a DMA slave. However, it would probably also respond to certain I/O read and I/O write operations from the system microprocessor, making it an I/O slave. If the adapter contained RAM or ROM that was in the system microprocessor address space, it would be a memory slave when that memory was accessed.

Typically, a slave is selected by a decode of:

- An address
- Status (-S0 exclusive or -S1)
- MADE 24 (if it is a memory slave)
- M/-IO.

The decode is latched at the leading or trailing edge of -ADL or the leading edge of -CMD. A DMA slave can also be selected by a latched decode of the same signals, using the arbitration level in place of the address.

Central Arbitration Control Point

The central arbitration control point (central arbiter) gives devices on the channel the ability to share and control the system. It allows burst data transfers and prioritization of control between devices. This central arbiter supports up to 16 devices, such as a DMA slave, a bus master, and the system microprocessor

Note: Information about programming the central arbitration control point can be found in the system-specific technical references.

The central arbiter uses seven signals to coordinate arbitration for all devices from a single arbitration point on the system board. These signals are -PREEMPT, ARB/-GNT, -BURST, and ARB0 through ARB3.

Arbitrating devices (local arbiters) requesting use of the system channel, drive -PREEMPT active. The central arbiter initiates an arbitration cycle when the present device releases the channel. The central arbiter indicates an arbitration cycle by driving ARB/-GNT to the arbitrate state. The requesting local arbiters then drive their assigned 4-bit arbitration level onto the arbitration bus. When an arbitrating device sees a more significant bit low (inactive) on the arbitration bus than those driven low by itself, it stops driving its lower-order bits onto the arbitration bus. The arbitrating device driving the lowest arbitration level thereby wins control of the channel when ARB/-GNT goes to the grant state.

Arbitrating devices with multiple transfers to perform must signal the central arbiter by driving -BURST active until all transfers have been completed or until another device drives -PREEMPT active, in which case further transfers are postponed until the device wins the system channel again. Because -PREEMPT and ARB0 through ARB3 may be driven by multiple devices, they must be driven through an open collector driver. ARB/-GNT is driven by the central arbiter only.

The central arbiter recognizes an end-of-transfer when both status signals (-S0 and -S1), -BURST, and -CMD are inactive. Control of the channel is then transferred to the next higher priority device or to the system microprocessor by default.

The interaction between the central arbitration control point and the local arbiters is called distributed arbitration. The following is a block diagram of distributed arbitration.

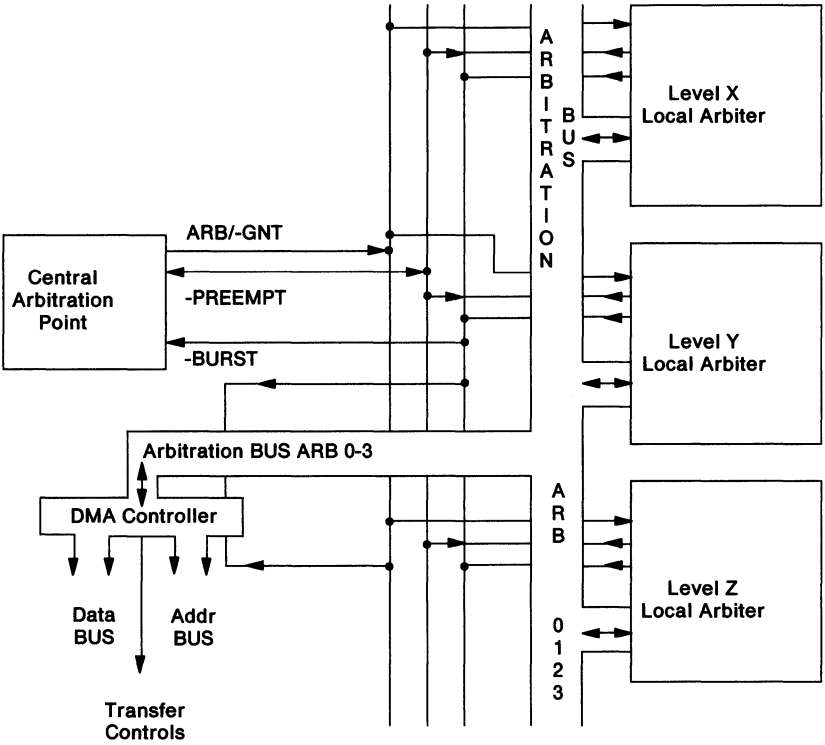


Figure 10. Distributed Arbitration Block Diagram

Local Arbiters

Devices requesting the use of the channel must implement logic to drive the arbitration bus in a way that allows all competing devices to recognize the winner. This logic is known as a local arbiter. An arbitrating device should compete for control of the channel only if it has driven -PREEMPT active, and ARB/-GNT has subsequently gone to the arbitrate state. A competing local arbiter drives its arbitration level onto the arbitration bus, then compares, on a bit-by-bit basis, its arbitration level with the value appearing on the arbitration bus beginning with the most significant bit, ARB3. If the competing local arbiter detects a mismatch on one of the bits, it should immediately cease driving all lower-order bits. If the local arbiter subsequently recognizes a match on that bit, it may continue driving lower-order bits until another mismatch is detected. Because the arbitration bus is driven by open-collector drivers, multiple arbiters can safely drive the bus. The following is an example of bus arbitration.

1. Two devices with arbitration levels 1010 and 0101 (hex A and 5) compete for the channel. Both devices drive their arbitration levels on the bus, which now appears as 0000.

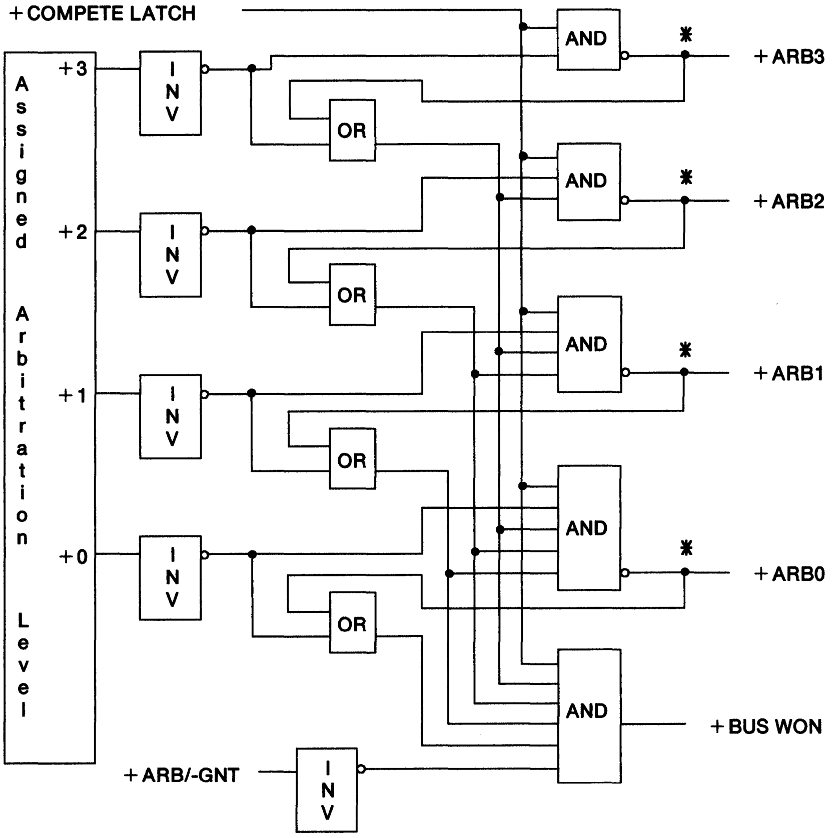
2. The first device (1010) detects a mismatch on ARB3 and stops driving all lower-order arbitration bus bits (ARB2 and ARB0 in this case).

3. The second device (0101) detects a mismatch on ARB2 and stops driving the lower-order arbitration bit (ARB1, in this case). The arbitration bus now shows 0111.

4. The second device now sees a match on ARB2 and resumes driving ARB1 of the arbitration bus.

5. The arbitration bus now shows a value of 0101, and the second device wins control of the channel.

The following is a simplified example of a local arbiter.



* Open-Collector Driven

Figure 11. Local Arbiter Example

Burst Mode

One of the most efficient ways for a device, such as a fixed disk drive, to transfer data is in bursts. These bursts are often separated by long inactive periods. The burst mode makes these devices more efficient.

To use the burst mode, the local arbiter activates -BURST and does not release it until after the leading edge of the last -CMD pulse in the burst sequence. The following diagram shows a burst operation without interference.

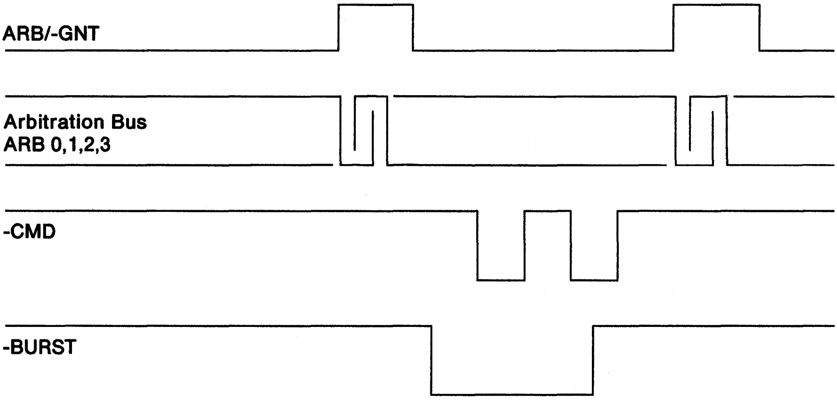


Figure 12. Burst Mode Timing

Preemption

Whenever an arbitrating device needs service, it activates -PREEMPT. The following timing diagram shows -PREEMPT occurring during a burst operation.

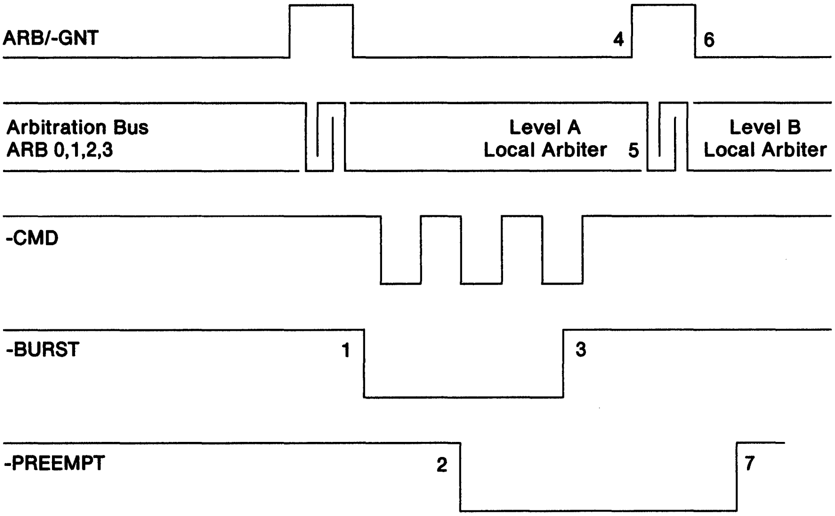


Figure 13. Preempt Timing

The sequence is as follows:

1. Device A gains control of the channel.
2. Device B, nearing an overrun condition, requests preemption.
3. Device A, still in control of the channel, completes any partial transfers and removes -BURST. Device A does not participate in the next arbitration cycle if the fairness feature is active. See "Programmable Fairness and the Inactive State" on page 30.
4. When the central arbitration control point recognizes the end of transfer, it removes the grant.
5. Arbitration for channel control begins.
6. When ARB/-GNT is in the grant state, the new local arbiter gains control of the channel.
7. Device B, the preempting device, removes -PREEMPT in response to the grant.

If an arbitrating device holds -BURST active for more than 7.8 microseconds after an active -PREEMPT, an error condition may exist, and a channel time-out may occur. ARB/-GNT is driven high immediately and takes control of the channel from the controlling master. An NMI is driven active. The channel remains in the arbitration state under the control of the system microprocessor until released by a NMI handler.

Programmable Fairness and the Inactive State

A programmable *fairness* feature in bursting DMA slaves and bursting masters allows each device a share of the channel time. If the fairness feature is active and an arbitrating device that owns the channel is preempted, the device enters the Inactive state and must wait for an inactive -PREEMPT and an inactive (trailing) edge of status to compete for the channel again. The fairness feature allows the system to service all arbitrating devices in order of priority before the same device can gain control of the channel again.

Arbitration Bus Priority Assignments

The following figure identifies the arbitration level assignments that have been maintained to ensure hardware and software compatibility with existing products. The functions with the lowest arbitration level have the highest priority.

ARB Level	Compatibility Assignment
-2	Memory Refresh
-1	NMI
0	DMA Channel 0 (Programmable to any arbitration level)
1	DMA Channel 1
2	DMA Channel 2
3	DMA Channel 3
4	DMA Channel 4 (Programmable to any arbitration level)
5	DMA Channel 5
6	DMA Channel 6
7	DMA Channel 7
8	Available
9	Available
A	Available
B	Available
C	Available
D	Available
E	Available
F	System Microprocessor

Figure 14. Arbitration Bus Priority Assignments for Compatibility

DMA channels can be masked in order to install a bus master at the assignment specified for that DMA channel. Information about programming the central arbitration control point can be found in the system-specific technical references.

NMI service is executed at a priority level higher than 0, called -1. Memory refresh is prioritized at -2, two levels higher than 0. Levels -1 and -2 are reached on the system board only while ARB/-GNT is in the arbitrate state.

When the central arbitration control point receives a level -1 request (NMI, a system-board internal signal), it may activate -PREEMPT, wait for the end of transfer, and then place ARB/-GNT in the arbitrate state, denying channel activity to arbitrating devices. The grant is then given to the level -1 request, and ARB/-GNT is held in the arbitrate state until the operation is complete and the NMI is reset.

Channel Support

The 32-bit bus requires unique logic to permit masters with 16-bit data to communicate with slaves with 32-bit data.

Address Bus Translator

A 32-bit slave uses the '-byte enable' channel signals (-BE0 through -BE3) as part of its address instead of A0 and -SBHE. A 16-bit master does not provide these four '-byte enable' signals; the system generates them when a 16-bit master has control of the bus.

Data Bus Steering

Eight-bit masters are not supported; however, an 8-bit microprocessor can be used as a 16-bit master if it does its own data steering of the two low-order data bytes.

A 32-bit slave writes data to and reads data from data bits 0 through 31. A 16-bit master does not use data bits 16 through 31; the system board logic must cross data over from the low 16 data lines (D0 through D15) to the high data lines (D16 through D31) and back at the appropriate times. Compensation for the added delay is the responsibility of the 32-bit slave.

A0	-SBHE	Description
0	1	Byte 0 Only (D0 – D7)
1	0	Byte 1 Only (D8 – D15)
0	0	Byte 0 and Byte 1 (D0 – D15)
1	1	Invalid

Figure 15. Steering Control

TR 32: This signal is driven inactive by 32-bit masters only. When TR 32 is active, it is used by:

- The Central Translator Logic to drive -BE0 through -BE3
- The Central Steering Logic to perform bus steering.

When TR 32 is active, 32-bit slaves can use it to recognize that the controlling master is not 32-bit and compensate for additional delay attributable to the Central Steering Logic.

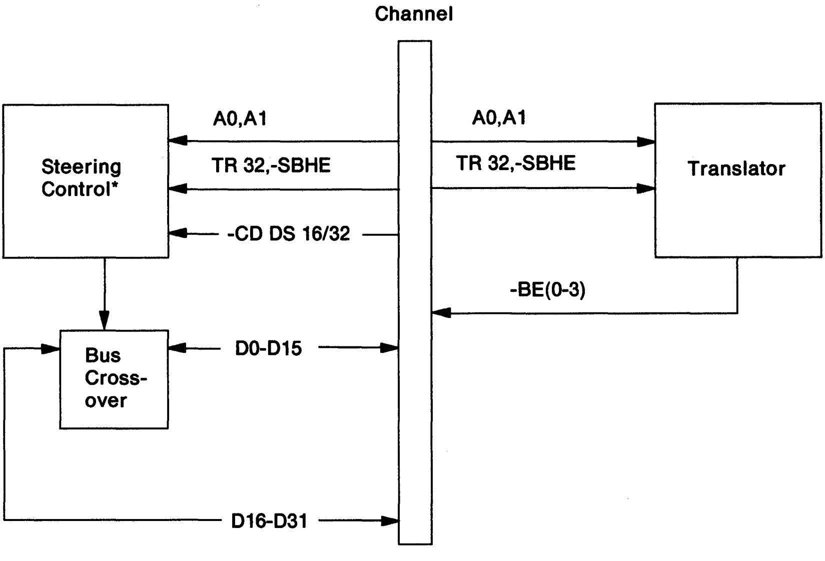
Central Steering Logic: Central Steering Logic uses A0, A1, -SBHE, -CD DS 16, and -CD DS 32 to steer data in support of 16-bit masters communicating with 32-bit slaves.

Central Translator Logic: Central Translator Logic translates A0, A1, and -SBHE to -BE0 through -BE3, when TR 32 is active.

-BE(0-3): Signals -BE0 through -BE3 are:

- Driven by a 32-bit master that has control of the bus
- Created by the Central Translator Logic when a 16-bit master has control of the bus
- Used by 32-bit slaves only.

The following block diagram shows the implementation of data bus steering.



* For 16-bit devices to 32-bit devices

Figure 16. Data Bus Steering Implementation

Level-Sensitive Interrupt Sharing

The main objectives of level-sensitive interrupt sharing are to:

- Simplify the logic-sharing design of adapters
- Reduce transient sensitivity of the interrupt controller
- Provide compatibility with existing software
- Allow for a mixture of sharing and nonsharing hardware on the same interrupt level.

Each adapter designed for the Micro Channel architecture uses a level-sensitive, active-low, interface mechanism. This mechanism, an open-collector driver (or tri-state driver gated active-low), drives the interrupt request line for levels assigned for the adapter function.

Note: Designers may want to limit the number of devices that share an interrupt level because of performance and latency reasons.

An adapter must hold the level-sensitive interrupt active until it is reset as a result of servicing the interrupt (reset). Service routines must not issue an End of Interrupt instruction (EOI) to the interrupt controller until the interrupt line of the device being serviced is reset. All adapters must also provide an interrupt-pending latch that is readable at an I/O address bit position and can be reset by normal servicing of the device.

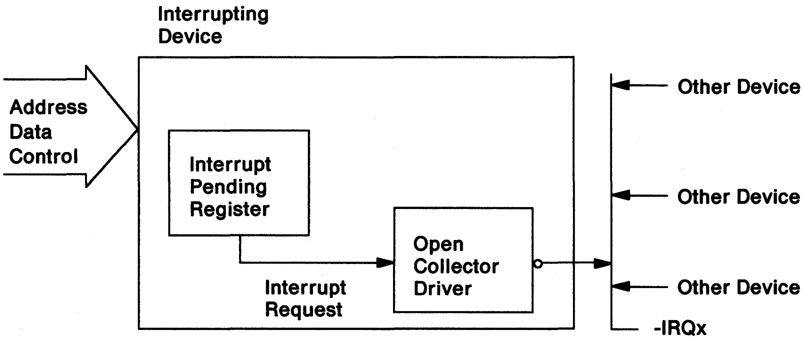


Figure 17. Typical Adapter Interrupt Sharing Implementation

Level-sensitive interrupts are interlocked between the hardware and software that support the interrupt service. Lost or spurious interrupts are more easily isolated. The following figure shows the sequence of interrupt sharing and the interaction of hardware and software when an interrupt is serviced by the system microprocessor.

A bus master can also service interrupts in a similar manner. Interrupt requests to the interrupt controller can be masked off and the request can be serviced by a bus master.

Hardware Operation	**Software Operation**
1. An interrupt condition sets hardware interrupt line X to an active (low) level with an open-collector driver, and sets an interrupt-pending latch readable by code.	
2. An interrupt controller presents the interrupt to the supporting microprocessor.	
	3. The supporting microprocessor begins executing code at the beginning of the appropriate chain of interrupt handlers.
	4. The interrupt-handler code reads the interrupt-pending latch of the first device in the chain. If the latch is not pending, the next device in the chain is tested. When a reporting card is detected, the handler executes the appropriate service routine.
	5. The interrupt service routine operates the device hardware.
6. The adapter hardware resets the interrupt-pending latch and the hardware interrupt line because of the interrupt service routine actions.	
	7. The interrupt service routine finishes executing code resetting the interrupt controller as its final action (End of Interrupt).
8. The interrupt controller resets.	
9. If an interrupt is pending (IRQ active by another device), the interrupt controller sets immediately and the sequence starts again.	

Figure 18. Interrupt Sharing Sequence

Micro Channel Critical Timing Parameters

This section provides timing diagrams for Micro Channel operations. All timings are related to a nominal cycle. The cycle may be changed by systems and adapters in various ways. Developers should ensure that hardware and software designs operate over the ranges specified and do not depend on a given performance level.

Basic-Transfer Cycle

This section provides the specification for critical timing parameters for the basic-transfer cycle.

Simplified Basic-Transfer Cycle

Most masters, including DMA controllers, transfer data with the same control sequence. Except for matched-memory transfers, the signals appear on the channel in the following sequence:

1. Address bus, MADE 24, M/-IO, and -REFRESH (if applicable) become valid, beginning the cycle.
2. The 'status' signals, S0 and S1, become valid.
3. The 'address decode latch' signal (-ADL) becomes valid. A slave may latch decodes of address, status (S0 exclusive or S1), and M/-IO.
4. In response to an unlatched address decode, MADE 24, and M/-IO, the adapter returns:
   - -CD SFDBK
   - -CD DS 16 (if the device is capable of 16-bit operations)
   - -CD DS 16 and -CD DS 32 (if the device is capable of 32-bit operations).
5. In response to an unlatched address decode, MADE 24, M/-IO, and status, the adapter drives CD CHRDY inactive if the cycle is to be extended.
6. Write data appears on the bus (for the write cycle).

7. -CMD becomes active and -ADL becomes inactive. A slave must latch decodes of address, status (S0 exclusive or S1), and M/-IO if they were not latched at the fall of -ADL.

8. The 'status' signals become inactive.

9. The 'address' signals become invalid in preparation for the next cycle.

10. In response to an address change:

    - -CD SFDBK is set inactive by the device.
    - -CD DS 16 is set inactive by the device.
    - -CD DS 32 is set inactive by the device.

11. If CD CHRDY has been set inactive, the system holds in this state until CD CHRDY is set active. This line should not be held inactive longer than specified.

12. The device places data on the bus in preparation for the trailing edge of -CMD (for the read cycle).

13. The address, 'status' signals, and M/-IO for the next cycle may become valid.

14. -CMD goes inactive, ending the cycle.

Note: The address and status can be overlapped with the preceding cycle to minimize the memory access time impact on performance.

The sequence for the basic-transfer cycle is as follows.

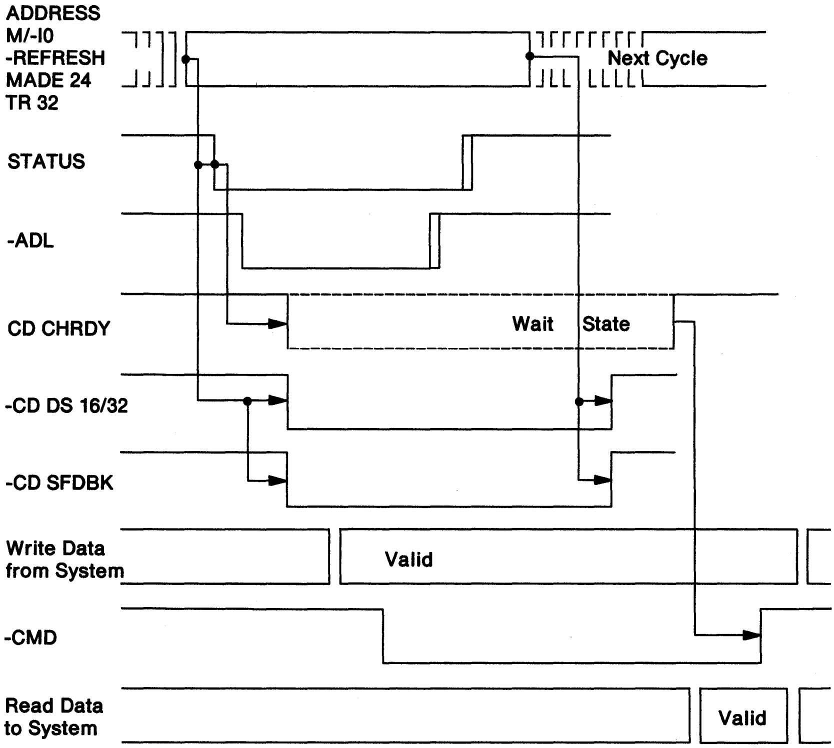


Figure 19. Overview of the Basic-Transfer Cycle

I/O and Memory Cycle

The timing diagrams for the basic I/O and memory cycle appear on the following pages in this sequence:

- Default cycle (200 nanoseconds minimum)
- Synchronous-extended cycle (300 nanoseconds minimum) - Special case
- Asynchronous-extended cycle (≥300 nanoseconds minimum) - General case.

The timing diagrams for the matched-memory cycle appear in the system-specific technical reference for those models that support matched-memory cycles.

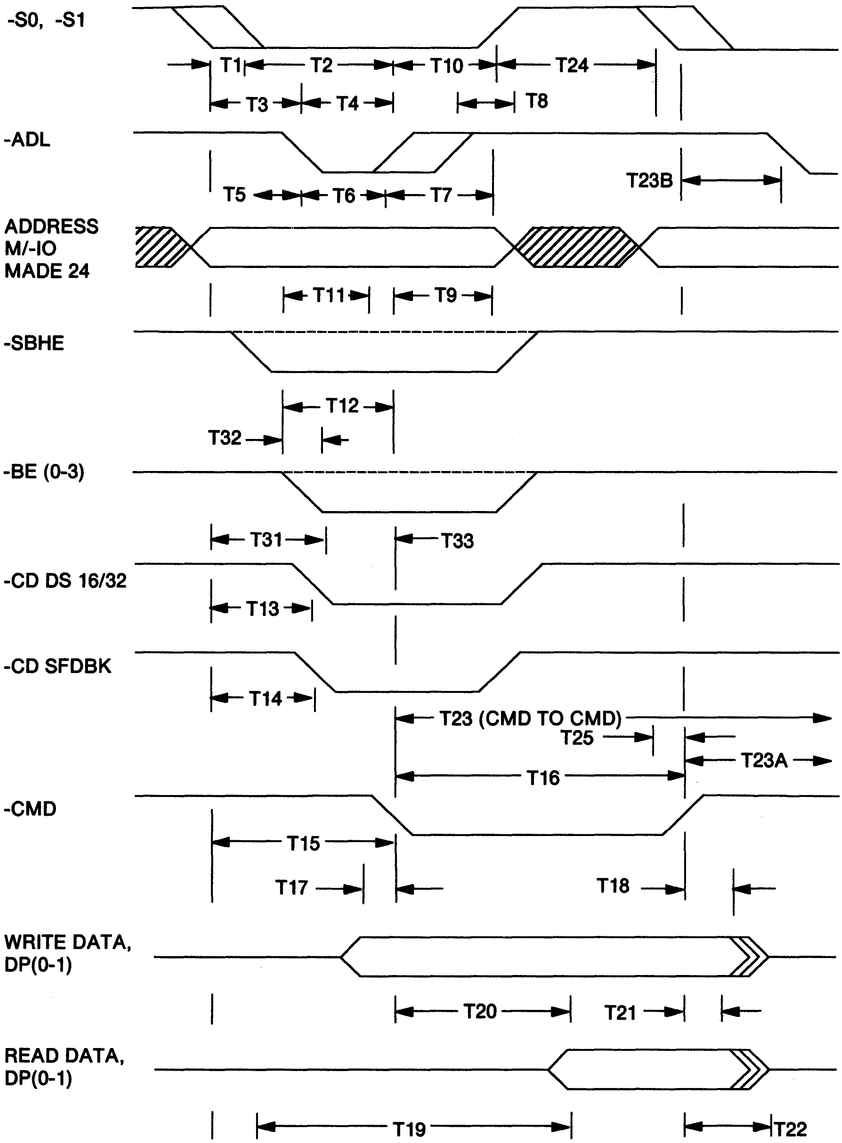
Whether a default, a synchronous-extended, or an asynchronous-extended cycle is performed depends on how a slave uses CD CHRDY.

A default cycle occurs when a slave does not hold CD CHRDY inactive longer than the time specified after address valid and status active.

A synchronous-extended cycle occurs when a slave releases CD CHRDY synchronously within the specified time after the leading edge of -CMD. The slave provides the read data within a specified time from -CMD.

An asynchronous-extended cycle occurs when a slave releases CD CHRDY asynchronously. However, the slave provides the read data within the specified time from CD CHRDY release.

Default Cycle



	Timing Parameter	Min/Max	Note
T1	Status active (low) from ADDRESS,M/-IO,-REFRESH valid	10 / - ns	
T2	-CMD active (low) from Status active (low)	55 / - ns	2
T3	-ADL active (low) from ADDRESS,M/IO,-REFRESH valid	45 / - ns	
T4	-ADL active (low) to -CMD active (low)	40 / - ns	
T5	-ADL active (low) from Status active (low)	12 / - ns	
T6	-ADL pulse width	40 / - ns	
T7	Status hold from -ADL inactive (high)	25 / - ns	2
T8	ADDRESS,M/-IO,-REFRESH,-SBHE hold from -ADL inactive	25 / - ns	2
T9	ADDRESS,M/-IO,-REFRESH,-SBHE hold from -CMD active (low)	30 / - ns	3
T10	Status hold from -CMD active	30 / - ns	2
T11	-SBHE setup to -ADL inactive	40 / - ns	2
T12	-SBHE setup to -CMD active	40 / - ns	2
T13	-CD DS 16/32 active (n) (low) from ADDRESS,M/-IO,-REFRESH valid	- / 55 ns	3
T14	-CD SFDBK active (low) from ADDRESS,M/-IO,-REFRESH valid	- / 60 ns	1
T15	-CMD active (low) from ADDRESS valid	85 / - ns	2
T16	-CMD pulse width	90 / - ns	
T17	Write data setup to -CMD active (low)	0 / - ns	
T18	Write data hold from -CMD inactive (high)	30 / - ns	
T19	Status to Read data valid (Access Time)	- / 125 ns	
T20	Read data valid from -CMD active (low)	- / 60 ns	
T21	Read data hold from -CMD inactive (high)	0 / - ns	
T22	Read data bus tri-state from -CMD inactive (high)	- / 40 ns	
T23	-CMD active to next -CMD active	190 / - ns	4
T23A	-CMD inactive to next -CMD active	80 / - ns	
T23B	-CMD inactive to next -ADL active	40 / - ns	
T24	Next Status active (low) from Status inactive	30 / - ns	
T25	Next Status active (low) to -CMD inactive	- / 20 ns	
T31	-BE(0-3) active from ADDRESS valid (32-bit masters only)	- / 40 ns	
T32	-BE(0-3) active from -SBHE, A0, A1 active	- / 30 ns	
T33	-BE(0-3) active to -CMD active	10 / - ns	

Figure 20. Default I/O and Memory Cycle (200 Nanoseconds Minimum)

Notes:

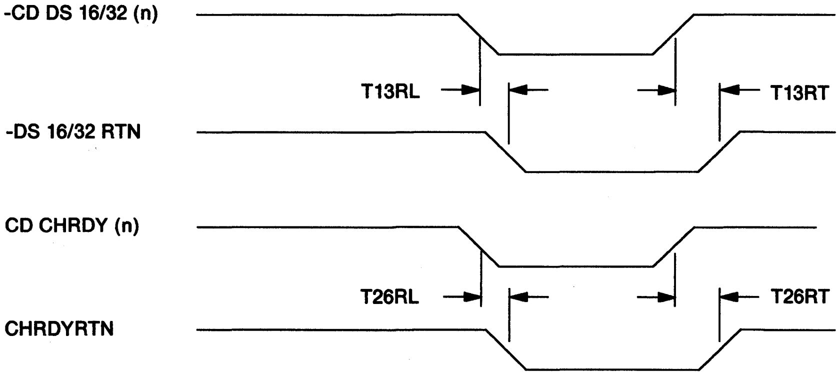
1. When slaves are selected by the controlling master, they drive -CD SFDBK. Slaves do not drive -CD SFDBK when they are selected by the '-card setup' signal.

2. Slaves should use transparent latches to latch information with the leading or trailing edge of -ADL or with the leading edge of -CMD.

3. -CD DS 16/32 and -CD SFDBK must be driven by *unlatched address decodes* because the next address may enter the current cycle early.

4. Any controlling master, including the DMA controller, can operate at a performance level lower than the one specified. Designers should not design to a given performance level because the level can be reduced by CD CHRDY, a lower microprocessor rate, a lower DMA controller rate, or by system contention.

Default Cycle Return Signals



	Timing Parameter	Min/Max	Note
T13RL	-CD DS 16/32 (n) active to -DS 16/32 RTN active	- / 20 ns	1
T13RT	-CD DS 16/32 (n) inactive to -DS 16/32 RTN inactive	- / 20 ns	1
T26RL	CD CHRDY (n) inactive to CHRDYRTN inactive	- / 20 ns	2
T26RT	CD CHRDY (n) active to CHRDYRTN active	- / 20 ns	2

Figure 21. Default Cycle Return Signals (200 Nanoseconds Minimum)

Notes:

1. These signals, -DS 16 RTN and -DS 32 RTN, are developed from a negative OR of signals received from each channel slave.

2. This signal, CHRDYRTN is developed from a positive AND of CD CHRDY signals received from each channel slave.

Synchronous Special Case of Extended Cycle

A synchronous-extended cycle occurs when a slave releases CD CHRDY synchronously within the specified time after the leading edge of -CMD. The slave provides the read data within a specified time from -CMD. The timing sequence is illustrated by the following figure.

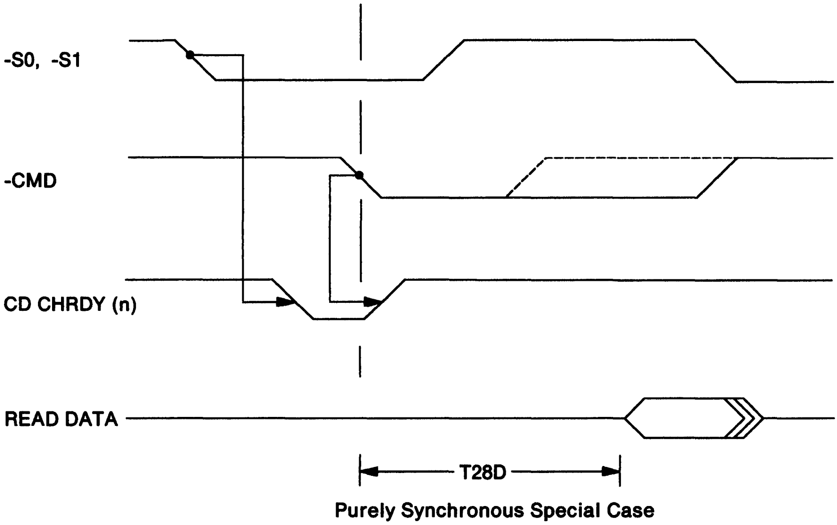
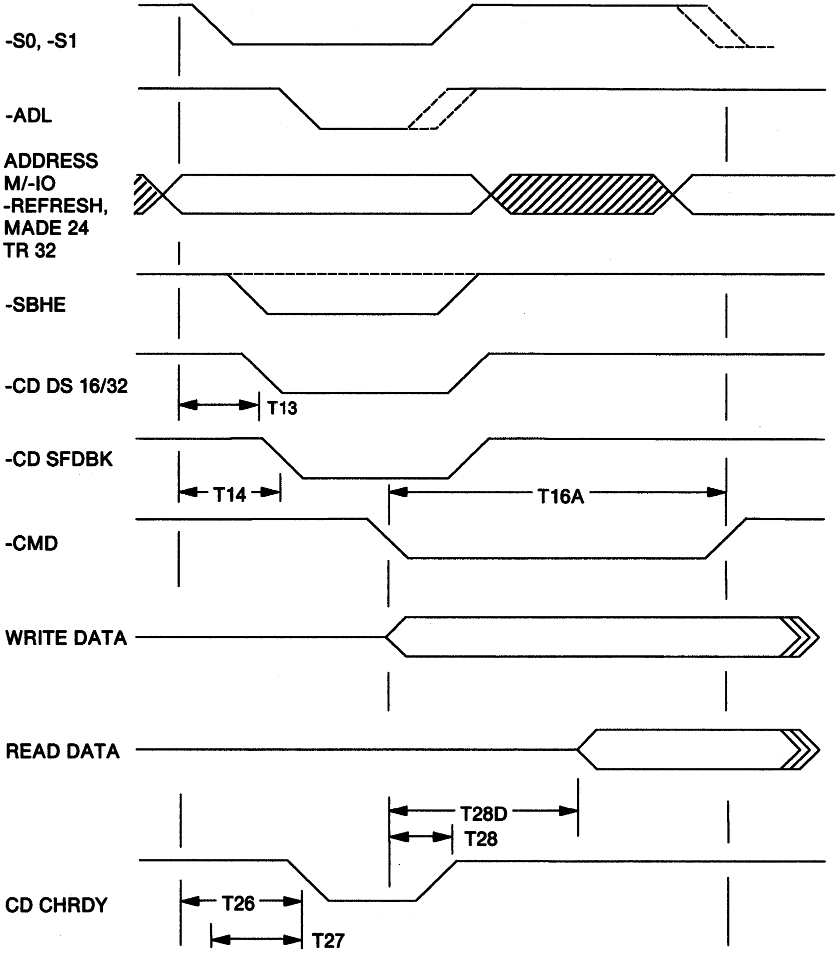


Figure 22. Timing Sequence for the Synchronous Special Case of Extended Cycle

Synchronous-Extended Cycle (300 Nanoseconds Minimum - Special Case)



	Timing Parameter	Min/Max	Note
T13	-CD DS 16/32 (n) active (low) from ADDRESS,M/-IO,-REFRESH valid	- / 55 ns	2
T14	-CD SFDBK (n) active (low) ADDRESS,M/-IO,-REFRESH valid	- / 60 ns	2
T16A	-CMD pulse width	190 / - ns	
T26	CD CHRDY (n) inactive (low) from ADDRESS valid	- / 60 ns	3, See T27
T27	CD CHRDY (n) inactive (low) from Status active	0 / 30 ns	3
T28	CD CHRDY (n) release (high) from -CMD active (low)	0 / 30 ns	1
T28D	Read Data valid from -CMD active (when used with T28)	0 / 160 ns	1

This figure shows only the parameters additional to the default cycle. All other parameters are the same as the default cycle.

Figure 23. Synchronous-Extended Cycle (300 Nanoseconds Minimum - Special Case)

Notes:

1. CD CHRDY is released by a slave performing a 300-nanosecond extended cycle that is synchronous with the leading edge of -CMD. Since CD CHRDY is generally an asynchronous signal, this is called a purely synchronous special case.

2. This is the same as default cycle timing (listed here for emphasis).

3. T27 is valid only when status becomes active 30 nanoseconds or more after the address is valid.

4. If status overlaps with a previous -CMD, then the CD CHRDY state is not valid during the overlapped period.

5. Slaves must not hold CD CHRDY inactive (low) more than 3.5 microseconds.

Notes:

Asynchronous-Extended Cycle (General Case)

An asynchronous-extended cycle occurs when a slave releases CD CHRDY asynchronously. However, the slave provides the read data within the specified time from CD CHRDY release. The timing sequence is illustrated by the following figure.

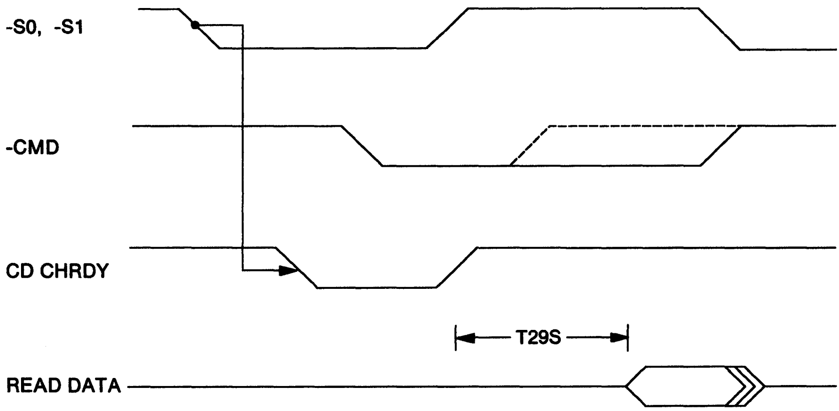
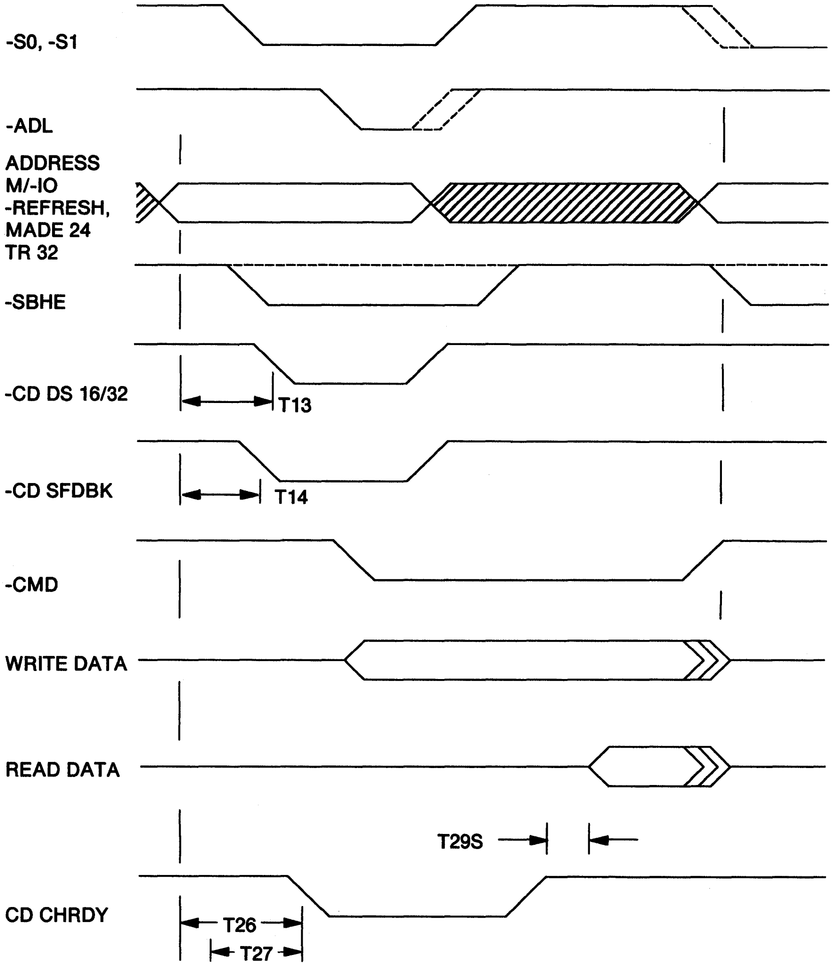


Figure 24. Timing Sequence for the Asynchronous-Extended Cycle (General Case)

Asynchronous-Extended Cycle (≥300 Nanoseconds Minimum - General Case)



	Timing Parameter	Min/Max	Note
T13	-CD DS 16/32 (n) active (low) from ADDRESS,M/-IO,-REFRESH valid	- / 55 ns	2
T14	-CD SFDBK (n) active (low) ADDRESS,M/-IO,-REFRESH valid	- / 60 ns	2
T26	CD CHRDY (n) inactive (low) from ADDRESS valid	- / 60 ns	See T27
T27	CD CHRDY (n) inactive (low) from Status active	0 / 30 ns	
T29S	Read data from slave valid from CD CHRDY (n) active (high)	- / 60 ns	1

This figure shows only the parameters additional to the default cycle. All other parameters are the same as the default cycle.

Figure 25. Asynchronous-Extended Cycle (≥300 Nanoseconds Minimum - General Case)

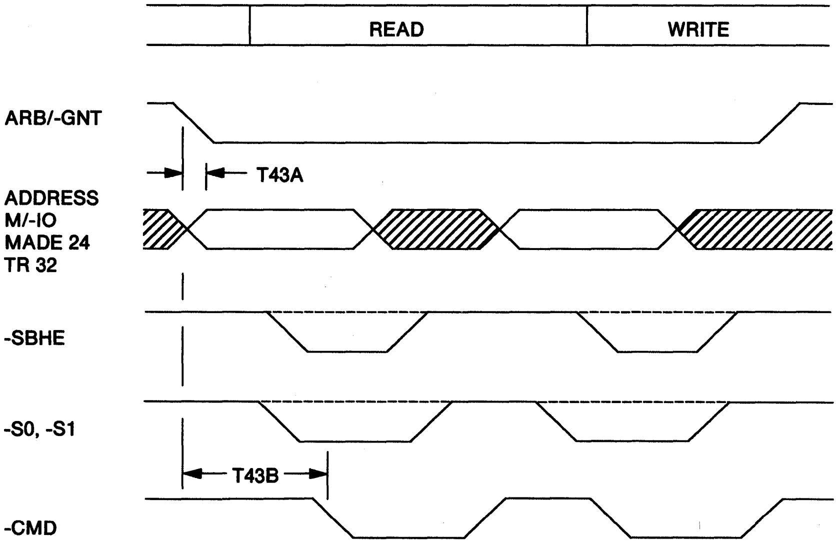
Notes:

1. CD CHRDY is released asynchronously by a slave performing a 300-nanosecond minimum cycle. The slave must present the Read data within the time specified after the release of CD CHRDY.
2. This is the same as default cycle timing (listed here for emphasis).
3. T27 is valid only when status becomes active 30 nanoseconds or more after the address is valid.
4. If status overlaps with the previous -CMD, then the CD CHRDY state is not valid during the overlapped period.
5. Slaves must not hold CD CHRDY inactive (low) more than 3.5 microseconds.

DMA Timing

This section provides the specification for critical timing parameters for DMA timing.

First Cycle After Grant

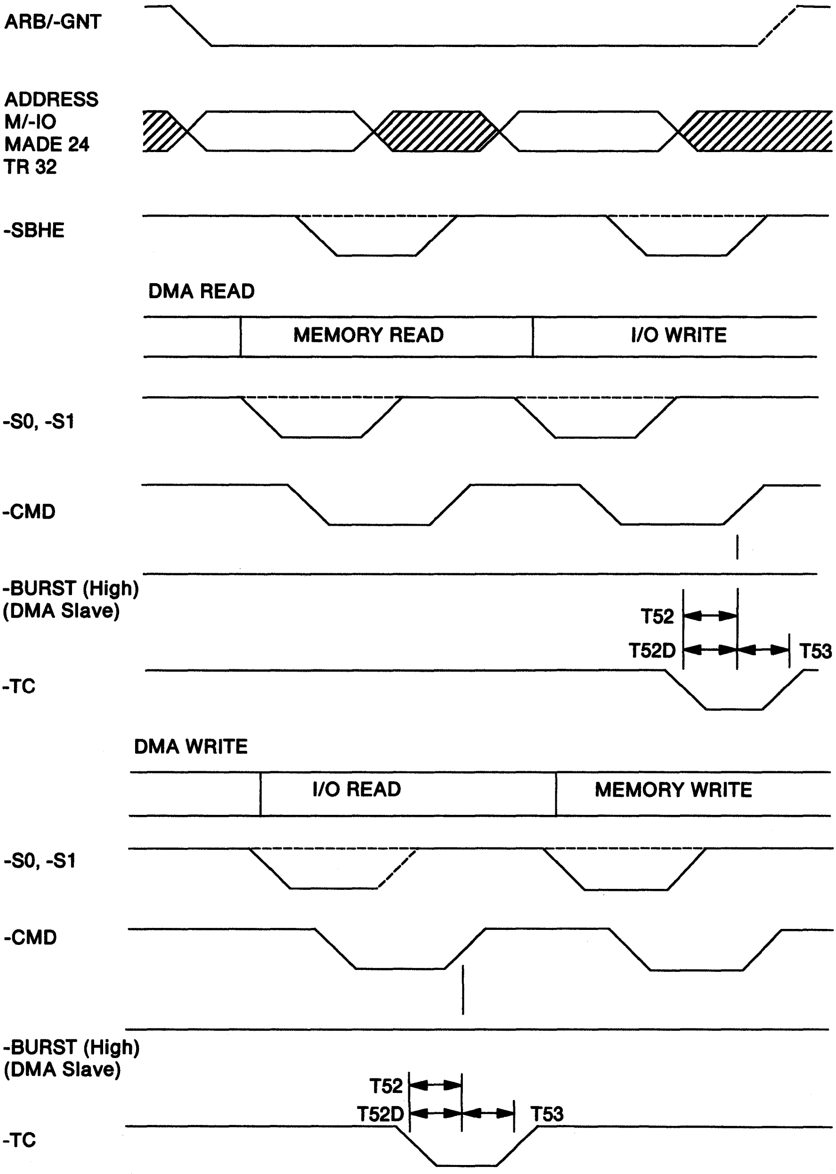


	Timing Parameter	Min/Max
T43A	ADDRESS valid from ARB/-GNT low	0 / - ns
T43B	-CMD active from ARB/-GNT low	115 / - ns

Figure 26. First Cycle After Grant

Note: A DMA controller must allow 30 nanoseconds after the grant, for a slave to generate an internal acknowledgment that it has been selected. During the first cycle, the DMA controller must allow this additional 30 nanoseconds before sampling -CD DS 16/32 RTN and CHRDYRTN, if it places an address on the bus within 30 nanoseconds after the grant. However, if the DMA controller places the address on the bus 30 nanoseconds after the grant, the additional 30 nanosecond allowance is not needed.

Single DMA Transfer (DMA Controller Controlled)



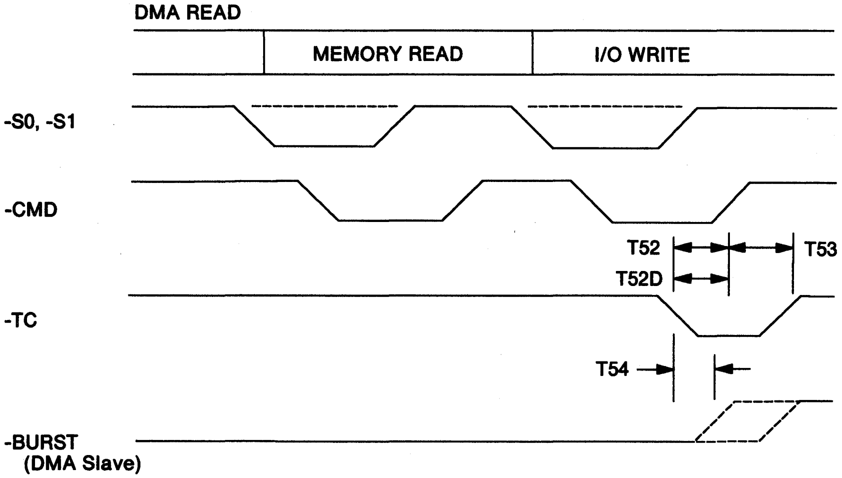
	Timing Parameter	Min/Max	Note
T52	-TC setup to -CMD inactive	30 / - ns	
T52D	-TC setup to -CMD inactive	15 / - ns	2
T53	-TC hold to -CMD inactive	10 / - ns	

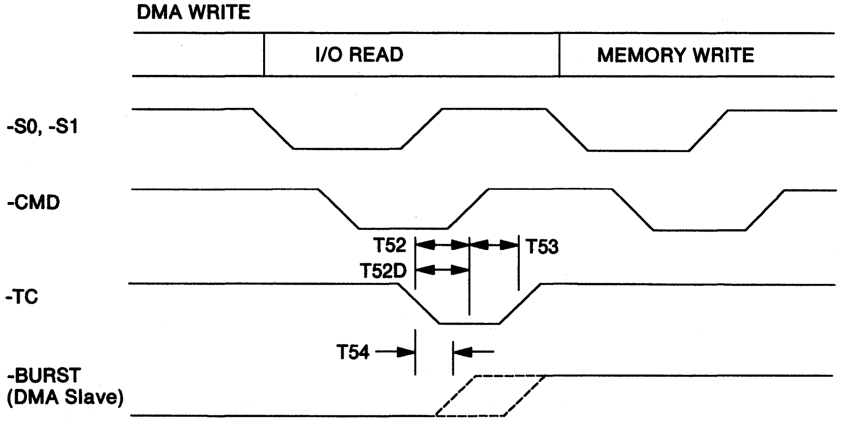
Figure 27. Single DMA Transfer (DMA Controller Controlled)

Notes:

1. Only those timing parameters additional to those specified for the basic-transfer cycle are included here.
2. Only for devices using a 200-nanosecond minimum default cycle.

Burst DMA Transfer (DMA Controller Terminated)





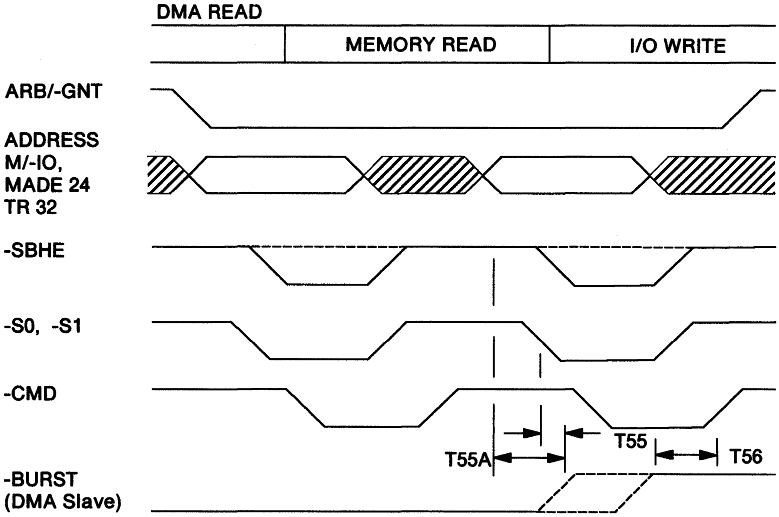
	Timing Parameter	Min/Max	Note
T52	-TC setup to -CMD inactive	30 / - ns	
T52D	-TC setup to -CMD inactive	15 / - ns	2
T53	-TC hold from -CMD inactive	10 / - ns	
T54	-BURST released by the DMA slave from -TC active	- / 30 ns	

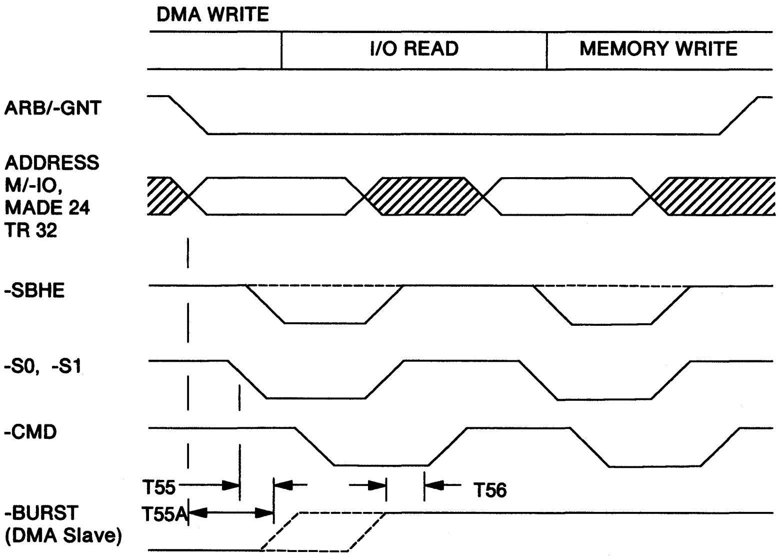
Figure 28. Burst DMA Transfer (DMA Controller Terminated)

Notes:

1. Only those timing parameters additional to those specified for the basic-transfer cycle are included here.
2. Only for devices using a 200-nanosecond minimum default cycle.

Burst DMA Transfer (DMA Slave Terminated - Default Cycle 200 Nanoseconds)





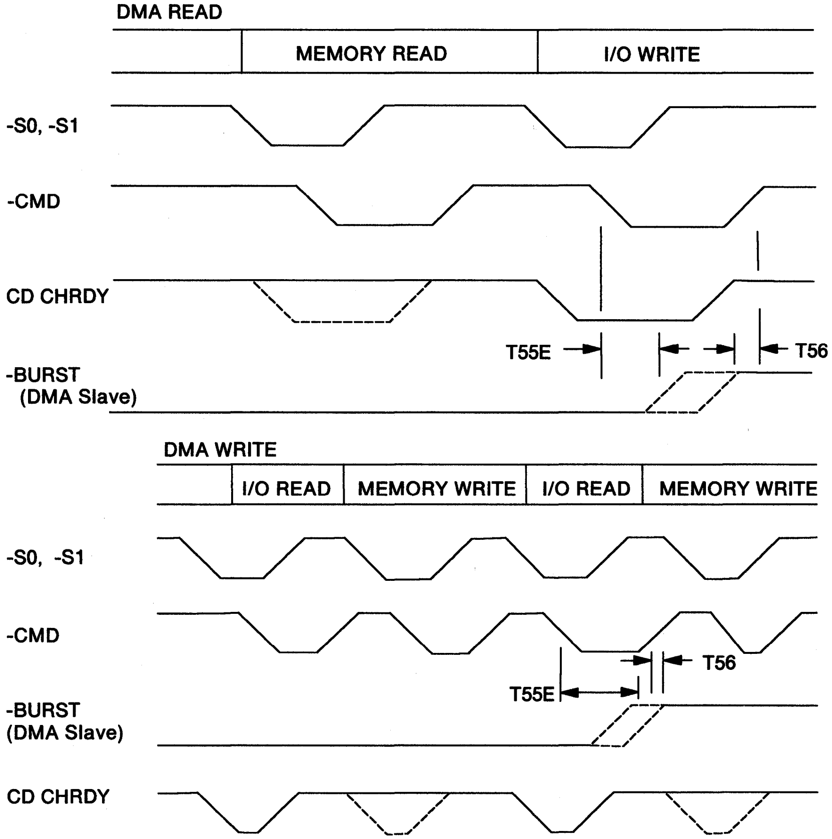
	Timing Parameter	Min/Max	Note
T55	-BURST released by the DMA slave from the last I/O Status active (default cycle only)	- / 40 ns	2
T55A	-BURST released by the DMA slave from the last I/O ADDRESS valid (default cycle only)	- / 70 ns	2
T56	-BURST inactive (high) setup to -CMD inactive	35 / - ns	3

Figure 29. Burst DMA Transfer (DMA Slave Terminated - Default Cycle 200 Nanoseconds)

Notes:

1. Only those timing parameters additional to those specified for the basic-transfer cycle are included here.
2. If, after releasing -BURST and upon receiving -SBHE, the DMA slave has another cycle to perform, it must redrive -BURST.
3. -BURST inactive (high) setup time to the end of -CMD (T56) must be guaranteed during the last I/O write cycle to prevent the DMA controller from starting the next cycle. This setup time (T56) is guaranteed by the sum of -BURST release by the DMA slave (T55/T55A) and the -BURST resistor-capacitor restoration time. The resistor-capacitor restoration time must not exceed 70 nanoseconds. T56 is the same for the default and extended cycles.

Burst DMA Transfer (DMA Slave Terminated - Synchronous-Extended Cycle 300 Nanoseconds)



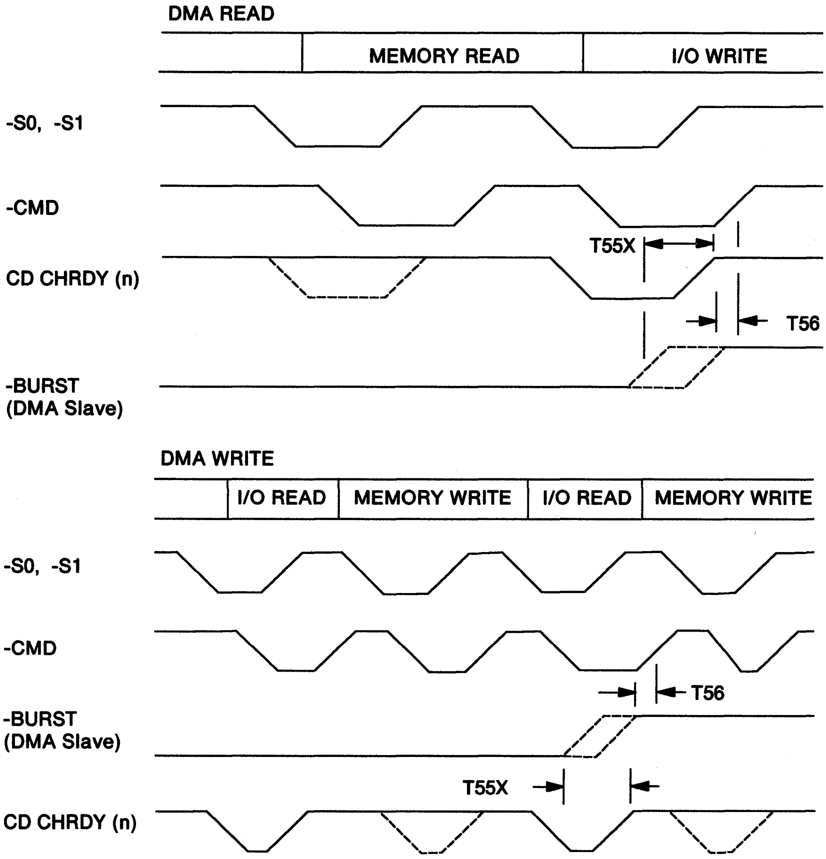
	Timing Parameter	Min/Max	Note
T55E	-BURST released by the DMA slave from the last -CMD active (extended cycles only)	- / 80 ns	
T56	-BURST inactive (high) setup to -CMD inactive	35 / - ns	2

Figure 30. Burst DMA Transfer (DMA Slave Terminated - Synchronous-Extended Cycle 300 Nanoseconds)

Notes:

1. Only those timing parameters additional to those specified for the basic-transfer cycle are included here.
2. -BURST inactive (high) setup time to the end of -CMD (T56) must be guaranteed during the last I/O write cycle to prevent the DMA controller from starting the next cycle. This setup time (T56) is guaranteed by the sum of -BURST release by the DMA slave (T55E) and the -BURST resistor-capacitor restoration time. The resistor-capacitor restoration time must not exceed 70 nanoseconds. T56 is the same for the default and extended cycles.

Burst DMA Transfer (DMA Slave Terminated - Asynchronous-Extended Cycle ≥300 Nanoseconds)



	Timing Parameter	Min/Max	Note
T55X	-BURST released by the DMA slave before CD CHRDY (n) active (high) (Async-Extended cycles only)	50 / - ns	
T56	-BURST inactive (high) setup to -CMD inactive	35 / - ns	2

Figure 31. Burst DMA Transfer (DMA Slave Terminated - Asynchronous-Extended Cycle ≥300 Nanoseconds)

Notes:

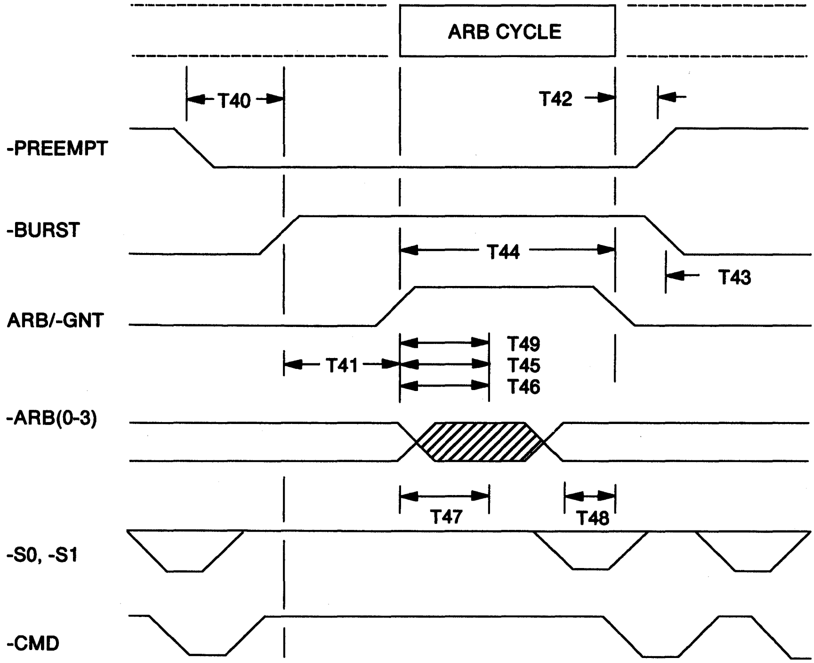
1. Only those timing parameters additional to those specified for the basic-transfer cycle are included here.

2. -BURST (high) setup time to the end of -CMD (T56) must be guaranteed during the last I/O write cycle to prevent the DMA controller from starting the next cycle. This setup time (T56) is guaranteed by the sum of -BURST release by the DMA slave (T55X) and the -BURST resistor-capacitor restoration time. The resistor-capacitor restoration time must not exceed 70 nanoseconds. T56 is the same for the default and extended cycles.

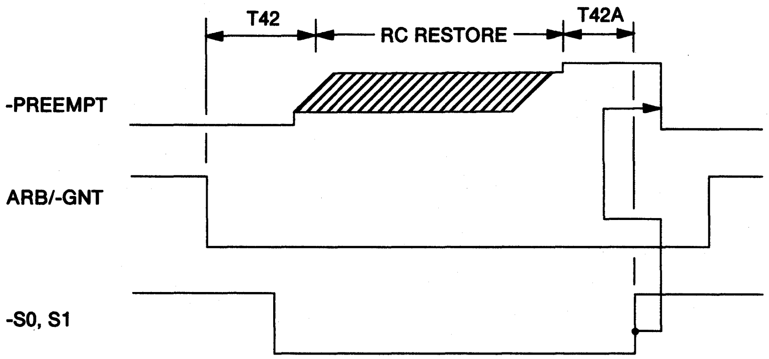
Arbitration Timing

This section provides the specification for critical timing parameters for arbitration protocol.

Arbitration Cycle



Exiting from Inactive State



	Timing Parameter	Min/Max	Note
T40	-PREEMPT active (low) to End of Transfer	0 / 7.8 µs	1
T41	ARB/-GNT high from End of Transfer	30 / - ns	6
T42	-PREEMPT inactive (high) from ARB/-GNT low	0 / 50 ns	
T42A	-PREEMPT inactive (high) to Status inactive	20 / - ns	5
T43	-BURST active (low) from ARB/-GNT low (By Bursting DMA slave)	- / 50 ns	4
T44	ARB/-GNT high	100 / - ns	2
T45	Driver turn-on delay from ARB/-GNT high	0 / 50 ns	3
T45A	Driver turn-on delay from lower priority line	0 / 50 ns	3
T46	Driver turn-off delay from ARB/-GNT high	0 / 50 ns	3
T47	Driver turn-off delay from higher priority line	0 / 50 ns	3
T48	Arbitration bus stable before ARB/-GNT low	10 / - ns	
T49	Tri-state drivers from ARB/-GNT high	- / 50 ns	

Figure 32. Arbitration Cycle

Notes:

1. The intent of this parameter is to limit the maximum non-preemptive ownership of the bus.

2. The value shown applies only to the special case implementation involving the central arbitration control point, and is provided for pulse width and portability considerations only. Arbitration can be extended by refresh or error recovery procedures. An arbiter should decode a win of the grant by a combined decode of the arbitration bus and the ARB/-GNT. The minimum arbitration time can be 100 nanoseconds when a level 0 arbiter and the central arbitration control point coordinate. In this special case, the central arbitration control point can terminate arbitration prematurely at 100 nanoseconds.

3. T45, T45A, T46, and T47 must be satisfied by ARB (0-3) drivers of all arbitrating bus participants.

4. This parameter applies to all bus winners.

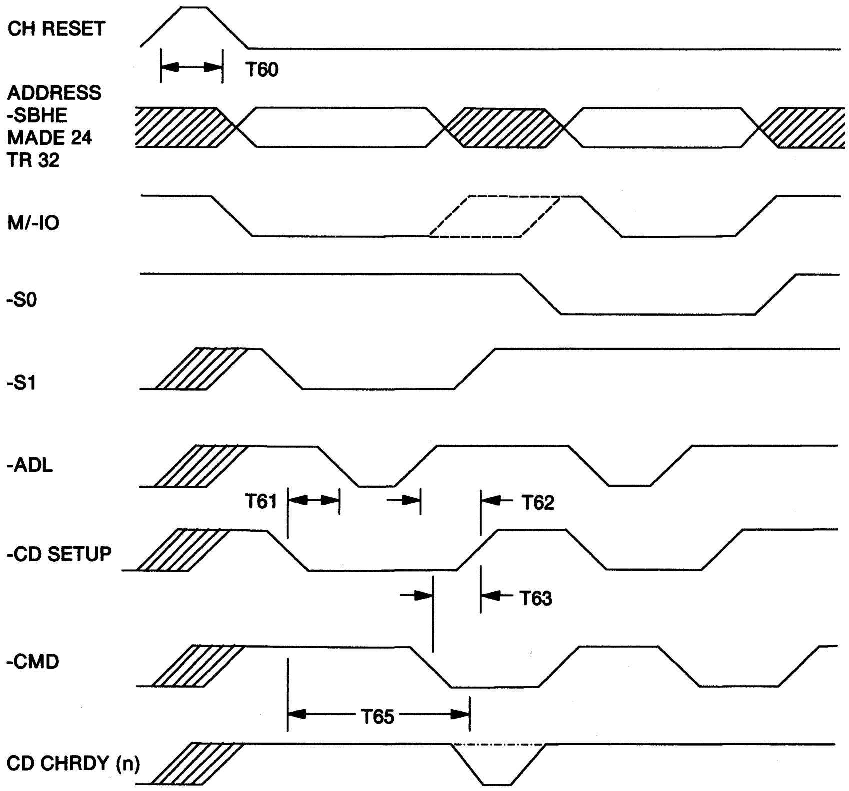
5. This represents the timing requirement after the resistor-capacitor line delay. This window is available for devices to detect inactive -PREEMPT and exit from the Inactive state after the rising edge of status.

6. Because no maximum is specified, a controlling master must degate bus drivers at the end-of-transfer condition. The end-of-transfer condition must be held stable until arbitration begins.

Configuration Timing

This section provides the specification for critical timing parameters for the system configuration protocol.

Setup Cycle



	Timing Parameter	Min/Max	Note
T60	CHRESET active (high) pulse width	100 / - ms	
T61	-CD SETUP (n) active (low) to -ADL active (low)	15 / - ns	
T62	-CD SETUP (n) hold from -ADL inactive (high)	25 / - ns	
T63	-CD SETUP (n) hold from -CMD active (low)	30 / - ns	
T65	CD CHRDY (n) inactive (low) from -CD SETUP (n) active	- / 100 ns	3

Figure 33. Setup Cycle

Notes:

1. Only those timing parameters that are different or additional to those specified for the basic-transfer cycle are included here.
2. The setup cycle is 300 nanoseconds minimum (default). A valid non-adapter selecting address must be present on the bus during system configuration.
3. A slave is allowed to extend the setup cycle beyond 300 nanoseconds using CD CHRDY. The slave qualifies the leading edge of CD CHRDY with active status.
4. Setup cycles are restricted to 8-bit transfers.

Additional Channel Timings

Timing Parameter	Min/Max	Note
-CD CHRDY inactive	- / 3.5 µs	
Card ID = 0000 (Indicating not ready)	- / 1 s	1
Retention of bus ownership after -PREEMPT active	- / 7.8 µs	2
Data steering (high 16-bit/low 16-bit data crossover)	- / 15 ns	3
Exiting Inactive state (driving -PREEMPT) after -PREEMPT inactive and the rising edge of Status	0 / -	4

Figure 34. Additional Channel Timings

Notes:

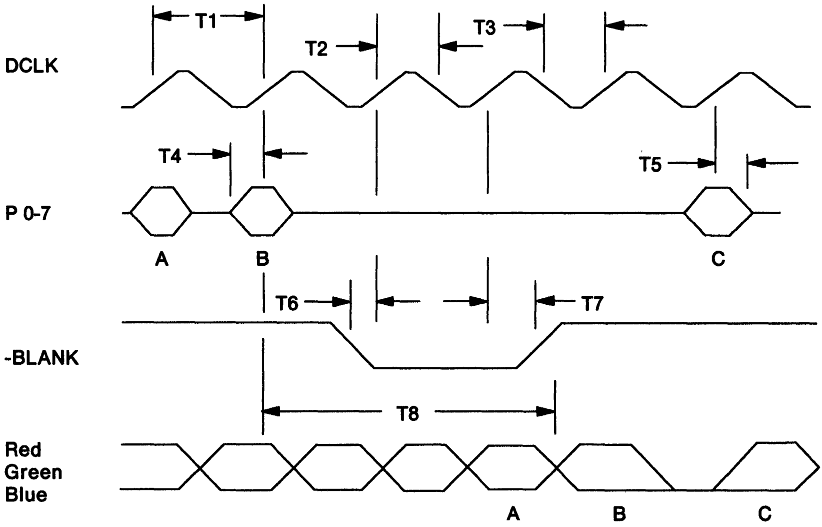
1. An adapter may issue an ID of hex 0000 for up to 1 second after channel reset to indicate it is not ready. Any adapter that continues to issue an adapter ID of hex 0000 (not ready) for more than 1 second is considered defective.

2. Applies to bursting masters with the fairness feature active.

3. When a 16-bit master accesses a 32-bit slave, the 32-bit slave is responsible to compensate for this added delay.

4. Applies to bursting masters (with the fairness feature active) that were preempted and have entered the Inactive state.

Auxiliary Video Extension Timing



Symbol	Description	Min.	Max.
T1	PEL Clock Period (tclk)	28 ns	10,000 ns
T2	Clock Pulse Width High (tch)	7 ns	10,000 ns
T3	Clock Pulse Width Low (tcl)	9 ns	10,000 ns
T4	PEL Set-Up Time (tps)	4 ns	-
T5	PEL Hold Time (tph)	4 ns	-
T6	Blank Set-Up Time (tbs)	4 ns	-
T7	Blank Hold Time (tbh)	4 ns	-
T8	Analog Output Delay (taod)	3(T1) + 5 ns	3(T1) + 30 ns

Figure 35. Auxiliary Video Connector Timing (DAC Signals)

Note: See “Video Subsystem” for additional video timing information.

Notes:

Index

A

additional channel timings 66
address bus 4, 11
address bus translator 32
-ADL signal 4
ARB/-GNT signal 8, 28
arbiter mismatch 25
arbiter refresh 31
arbitration level 30, 31
arbitration timing 62
ARB0 - ARB3 signal 8
asynchronous-extended cycle 47, 48
AUDIO GND signal 10
AUDIO signal 10
auxiliary video extension 17
auxiliary video extension timing 67
auxiliary video signals 13, 22
A0 - A23 signals 4
A24 - A31 signals 11

B

basic-transfer cycle 36
-BE signals 11, 32, 33
BLANK signal 13
-BURST signal 9
burst data transfers 23
burst DMA transfer 54
burst mode 28
burst mode timing 28
bus ownership 23
bus priority assignments 30
bus, address 4
bus, data 4
byte enable 32

C

CD CHRDY signal 7
-CD DS 16 signal 5
-CD DS 32 signal 11
-CD SETUP signal 9
-CD SFDBK signal 7
central arbitration control point 23
central steering logic 33
central translator logic 33
channel connector diagram 3
channel connector voltages and signal assignments 15
channel connectors 2, 15, 17
channel definition 2
channel support 32
-CHCK signal 9
CHRDYRTN signal 7
CHRESET signal 10
-CMD signal 7, 28
connector, auxiliary video extension 17
connector, matched-memory extension 19
connector, 16 bit 15
connector, 32-bit 18
connectors, channel 2
controlling master 1, 23
critical timing parameters 36
cycle, asynchronous-extended 47, 48
cycle, default 40
cycle, synchronous-extended 44

D

data bus 4, 11
data bus steering 32, 33
DCLK signal 13
default cycle 40
default cycle return 42
description, micro channel 1
distributed arbitration 25
distributed arbitration block diagram 25
DMA channels 31
DMA timing 50
-DS 16 RTN signal 5
-DS 32 RTN signal 12
D0 - D15 signals 4
D16 - D31 signal 11

E

EDCLK signal 14
electromagnetic interference 2
ESYNC signal 13
EVIDEO signal 14

F

fairness feature 30
first cycle after grant 50

H

HSYNC signal 13

I

I/O cycle 39
inactive state 30
inactive state, exiting 62
interrupt pending latch 34
interrupt sharing 34
-IRQ 14-15 signal 9
-IRQ 3-7 signal 9
-IRQ 9-12 signal 9

L

level-sensitive interrupts 34
local arbiters 24, 25, 27

M

M/-IO signal 5
MADE 24 signal 5
master, controlling 1, 23
matched-memory connector 19
matched-memory signal description 12
memory cycle 39
memory refresh 31
micro channel description 1
-MMC signal 12
-MMC CMD signal 12
-MMCR signal 12
multiple transfers 24

N

NMI 30
NMI service 31
NMI signal 31
nominal cycle 36

O

OSC signal 10
ownership, bus 23

P

-PREEMPT signal 8, 29
preempt timing 29
preemption 29
P0 - P7 signals 13

R

-REFRESH signal 10, 31
reserved signals 3, 4

S

-SBHE signal 5
setup cycle timing 64
signal descriptions, auxiliary video 13
signal descriptions, matched-memory 12
signal descriptions, 16-bit 4
signal descriptions, 32-bit 11
signal groups 19
signals, reserved 3, 4
single DMA transfer 52
slave 23
steering control diagram 32
synchronous-extended cycle 43, 44
system configuration timing 64
-S0, -S1 signals 5

T

-TC signal 9
timing parameters 36
timing, DMA 50
TR 32 signal 12, 32

V

VSYNC signal 13

Numerics

16-bit connectors 2
32-bit connectors 2

Notes:

Programmable Option Select

Description 1
Card Selected Feedback 3
System Board Setup 3
Adapter Setup 4
Adapter Identification 6
Required Fields 6
Adapter Selection for Setup 7
Adapter POS Implementation 7
POS Implementation Procedure 9
System Configuration Utilities 10
Automatic Configuration Utility 11
Change Configuration Utility 12
View Configuration Utility 13
Backup and Restore Configuration Utilities 13
Copy an Option Diskette Utility 13
Adapter Description Files 14
Format 14
Syntax 14
Example 20

Figures

1. POS I/O Address Space 3
2. POS I/O Address Decode 4
3. Channel-Check Active Indicator 5
4. Typical Adapter Implementation of POS 8
5. Subaddressing POS Extension Example 9
6. Syntax Symbol Key 14
7. Adapter Description File Syntax 15

Description

Programmable Option Select (POS) eliminates the need for switches from the system board and adapters by replacing their function with programmable registers.

The System Configuration utilities (described on page 10) automatically create configuration data for the system board and each adapter. This is achieved by reading a unique adapter ID number from each adapter, matching it with an adapter description file (ADF), and configuring the system accordingly. The resulting data and adapter ID numbers are stored in battery-backed CMOS RAM.

This data permits the power-on self-test (POST) to automatically configure the system whenever the system is powered on. The POST first verifies that the configuration has not changed by reading the adapter ID numbers and comparing them with the values stored in the battery-backed CMOS. If the configuration has changed, it is necessary to rerun the System Configuration utilities.

The adapters and the system board setup functions all share I/O addresses hex 0100 through 0107.

Warning:

- IBM recommends that programmable options be set only through the System Configuration utilities. Directly setting the POS registers or CMOS RAM POS parameters can result in multiple assignment of the same system resource, improper operation of the feature, loss of data, or possible damage to the hardware.
- Application programs should avoid using the adapter identification (ID) whenever possible. Software compatibility problems with systems and options may result.
- If an adapter and the system board are in setup mode at the same time, bus contention will occur, no useful programming can take place, and damage to the hardware can occur.
- After setup operations are complete, the Adapter Enable/Setup register (hex 0096) should be set to hex 00, and the System Board Enable/Setup register (hex 0094) should be set to hex FF.
- The channel reset bit (bit 7) in the Adapter Enable/Setup register must be 0 to program the adapters.

Setup functions respond to I/O addresses hex 0100 through 0107 only when their unique setup signal is active.

The system board does not support 16-bit I/O operations to 8-bit POS registers. Using 16-bit I/O instructions on 8-bit POS registers will cause erroneous data to be written to or read from the registers. Only 8-bit transfers are supported for setup operations.

The following figure shows the organization of the I/O address space used by the POS.

Address (Hex)	Function
0100	POS Register 0 - Adapter Identification Byte (Low byte)
0101	POS Register 1 - Adapter Identification Byte (High byte)
0102	POS Register 2 - Option Select Data Byte 1
	Bit 0 is designated as Card Enable.
0103	POS Register 3 - Option Select Data Byte 2
0104	POS Register 4 - Option Select Data Byte 3
0105	POS Register 5 - Option Select Data Byte 4
	Bit 7 is designated as channel check active.
	Bit 6 is designated as channel check status available.
0106	POS Register 6 - Subaddress Extension (Low byte)
0107	POS Register 7 - Subaddress Extension (High byte)

Figure 1. POS I/O Address Space

Bits 6 and 7 of address hex 0105 and bit 0 of address hex 0102 are fixed. All other fields within the address range of hex 0102 and 0105 are free form.

Card Selected Feedback

When the adapter is addressed, it responds by setting the Card Selected Feedback signal (-CD SFDBK) to 0. -CD SFDBK is derived by the adapter from the address decode, and driven by a totem pole driver. -CD SFDBK is latched by the system board and made available on subsequent cycles. -CD SFDBK may be used by automatic configuration or diagnostics to verify operation of an adapter at a given set of addresses. -CD SFDBK enables diagnostics to verify the operation of the adapter.

System Board Setup

The integrated I/O functions of the system board use POS information during the setup procedure. The bit assignments and functions may vary from system to system (refer to the system board setup information in the system-specific technical reference for the system you are dealing with).

Adapter Setup

The '-card setup' signal (-CD SETUP *(n)*) is unique for each channel connector. When -CD SETUP *(n)* is active, adapters recognize setup read and write operations. The adapter decodes -CD SETUP and all three low-order address bits (A0 through A2) to determine the POS register to be read from or written to. -CD SETUP is enabled by an I/O operation on the address range 0100 through 0107. The figures in this section show the complete address.

The setup routine (Automatic Configuration) obtains adapter information from ADFs and uses I/O addresses hex 0100 through 0107 to address the POS bytes of the adapter. The following figure shows the organization of the address space used by POS during adapter setup operations.

Address		**Address Bit**			
(Hex)	**-CD SETUP**	**A2**	**A1**	**A0**	**Function**
0100 (POS Register 0)	0	0	0	0	Adapter Identification Byte (Least-significant byte)
0101 (POS Register 1)	0	0	0	1	Adapter Identification Byte (Most-significant byte)
0102 (POS Register 2)	0	0	1	0	Option Select Data (Byte 1)*
0103 (POS Register 3)	0	0	1	1	Option Select Data (Byte 2)
0104 (POS Register 4)	0	1	0	0	Option Select Data (Byte 3)
0105 (POS Register 5)	0	1	0	1	Option Select Data (Byte 4)*
0106 (POS Register 6)	0	1	1	0	Subaddress Extension (Least-significant byte)
0107 (POS Register 7)	0	1	1	1	Subaddress Extension (Most-significant byte)

* These bytes contain one or more bits with specific assignments.

Figure 2. POS I/O Address Decode

Bytes hex 0100 and 0101 are 8-bit read-only. Bytes hex 0102 through 0107 are 8-bit read-only and read-write.

All bits in bytes hex 0102 through 0105 are free-form, adapter-dependent, and implemented except for the following:

- **Hex 0102, Bit 0:** Card Enable (CDEN): When this bit is set to 0, the adapter is disabled, responding only to setup read and write operations, and 'channel reset'. It does not respond to I/O or memory read or write operations, nor does it make any interrupt requests. When this bit is set to 1, the adapter is fully enabled.

- **Hex 0105, Bit 7:** Channel-Check Active Indicator (-CHCK): System memory and I/O functions that report a channel-check must set a channel-check active indicator to identify the source of the error. This indicator is bit 7 of address hex 0105 of each adapter POS address space. This bit can be interrogated by the nonmaskable interrupt (NMI) handler responding to 'channel check' for each adapter position until all reporting adapters have been identified. The following figure shows a typical implementation of the channel-check active indicator.

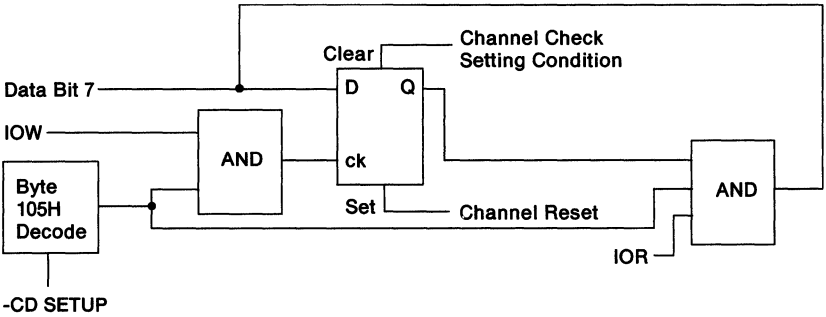


Figure 3. Channel-Check Active Indicator

The indicator is set to 0 on a channel-check condition or when bit 7 of POS Register 5 is set to 0. The indicator is set to 1 on a channel reset, or when bit 7 of hex 0105 is 1. This bit may be reset by any action occurring during the channel-check service routine. If the channel-check active indicator is used by an attachment, hex 0105 bit 6 must be used to indicate whether additional status is available through bytes hex 0106 and 0107.

- **Hex 0105, Bit 6:** Channel-Check Status Indicator: When set to 0, this bit indicates channel-check exception status is available from POS Registers 6 and 7. When set to 1, this bit indicates no status is available. Registers 6 and 7 may be the status, a pointer to status, or a command port to present the address elsewhere (for example, in a subaddress area).

  Bit 6 is required by all devices supporting the channel-check active indicator (bit 7). If a device does not use the channel-check active indicator, bit 6 may be defined to contain device-unique information. If a device uses the channel-check active indicator, but does not report status, bit 6 must be set to 1.

Adapter Identification

Each adapter has a unique 2-byte adapter ID. This enables diagnostic programs, configuration utilities, and POST routines to initialize the adapter when the system is powered on or reset.

To minimize the need for hardware, only bits driven to 0 require drivers. Pull-up resistors on the system board provide a 1 for each remaining bit. See "Micro Channel Adapter Design" for more information.

Required Fields

Several fields are not assigned specific bit locations within the *free form* POS bytes. However, the following are required if the adapter supports the function:

- **Fairness Enable Bit:** All bursting devices using the arbitration mechanism must support the *fairness* feature through a programmable fairness-enable bit. The default state of this bit is 1, requiring all devices to honor the fairness feature. When fairness-enable is set to 0, the fairness feature is disabled.

- **Arbitration Level Field:** All devices (bursting and nonbursting) using the arbitration mechanism must support a programmable arbitration level through a 4-bit allocation. This field allows incorrectly prioritized devices to be reassigned by diagnostic or system programs to reduce impacts on performance. Only one device may be assigned to each arbitration level.

- **Device ROM Segment Address Field:** All I/O devices containing memory-mapped I/O ROM must support a programmable Device ROM Segment Address field. This field can be up to 4 bits and provides the ROM of a device a starting address at any one of sixteen 8KB segments.

- **I/O Device Address Field:** All I/O devices that can simultaneously reside in a system with a device of the same type must support programmable I/O device addresses. This field eliminates addressing conflicts.

Adapter Selection for Setup

Each channel position has a unique 'setup' line (-CD SETUP) associated with it. See the system-specific technical references for more information.

Adapter POS Implementation

The following figure shows how an adapter typically implements POS. All designs must latch the least-significant bit of the device-dependent option-select byte. Bit 7 of POS Register 5 is set to 1 unless -CHCK is active from the adapter. The remaining bits can be implemented as required.

Note: Any adapter that POS does not completely initialize should implement a second enable, which is activated by adapter ROM routines or loadable software. The card-disable function (POS Register 2, bit 0) must override a second enable.

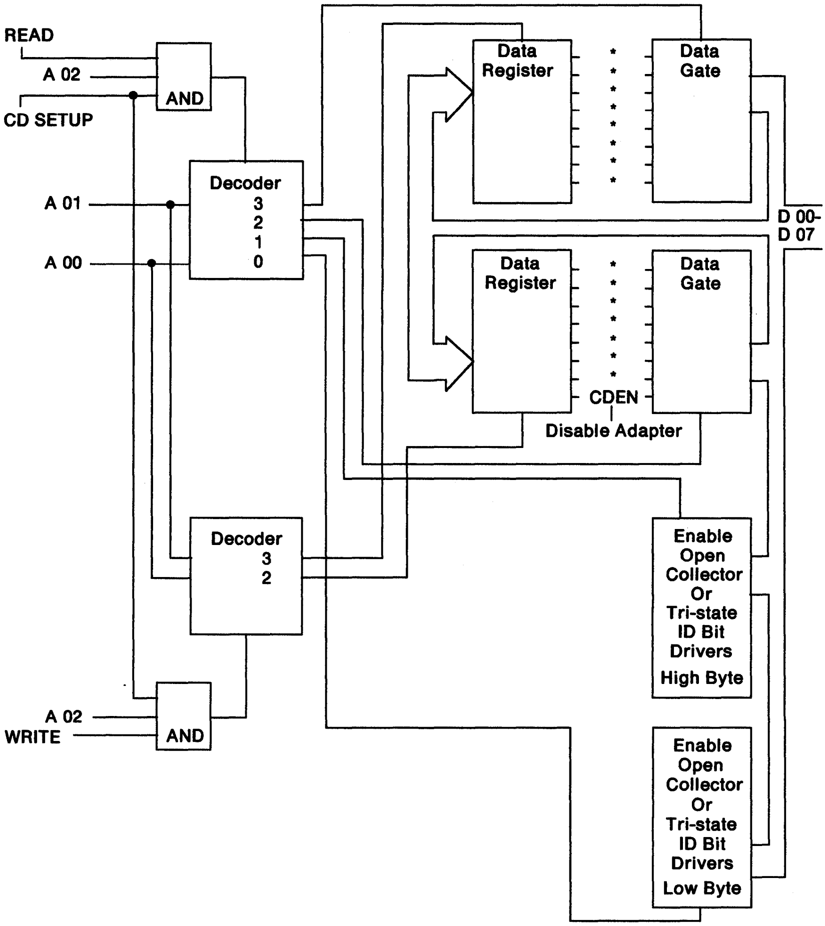


Figure 4. Typical Adapter Implementation of POS

The POS subaddressing extension allows the subaddressing of a block of initial program load (IPL) or setup information. Subaddressing bits (SAD00 through SAD15) are used to address RAM, a register stack, or other devices.

The following figure shows the subaddressing extension for memory. The counter registers increment after each least-significant byte of option-select information is written.

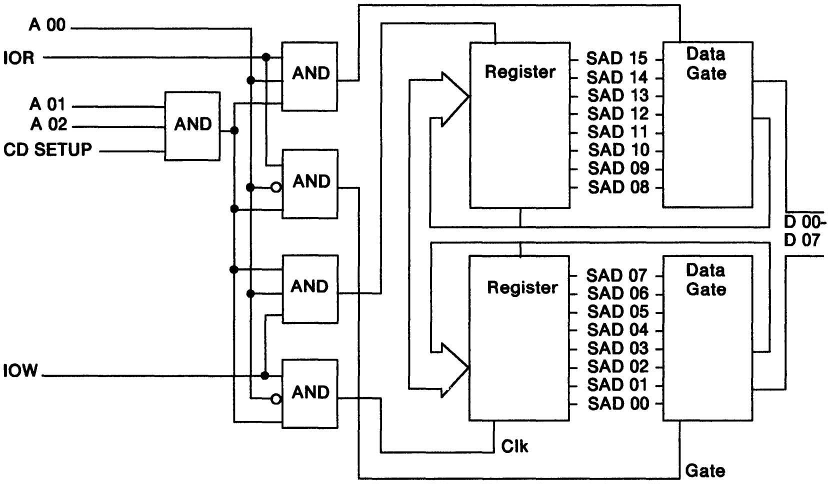


Figure 5. Subaddressing POS Extension Example

POS Implementation Procedure

Although the design of POS circuitry is the designer's choice, the following is an example of a typical POS implementation.

1. Disable interrupts.
2. Select the adapter for subsequent setup cycles. See the system-specific technical references for more information.
3. Read the adapter ID by an I/O read at hex 0100 and hex 0101.
4. Disable the adapter and place it in setup by performing an I/O write to hex 0102 with bit 0 off.
5. Write POS data to hex 0103, 0104, and 0105 in any order.
6. With bit 0 set to 1, write POS data to hex 0102.
7. Deselect the adapter. See the system-specific technical references for more information.
8. Enable interrupts.

The system microprocessor can communicate with the adapter, provided the adapter is enabled (bit 0 at hex 0102 set to 1). After the adapter has been set up, a subsequent I/O write does not affect these latches or permit the ID circuitry of the adapter to operate, unless the adapter is returned to setup.

System Configuration Utilities

Each system has a Reference Diskette containing the System Configuration utilities. These utilities identify the installed hardware and interpret the system resources (I/O ports, memory, interrupt levels, and arbitration levels) for each device. The System Configuration utilities are contained within the Set Configuration program.

The Reference Diskette enables the user to configure the system in one of two ways:

- Running the Automatic Configuration utility after a configuration error is displayed
- Selecting Set Configuration from the Main Menu.

The Set Configuration program uses information contained in adapter description files (ADFs) to track and allocate system resources. Each ADF describes the resources that can be allocated to a specific adapter and the POS setting used to indicate those resource assignments. When more than one device is configured to the same resource and that resource cannot be shared, only one of the conflicting devices is enabled.

ADF data for the system board and some adapters is contained on the Reference Diskette. Before new adapters are installed, their associated ADFs must be merged onto a backup copy of the Reference Diskette by selecting Copy an Option Diskette from the Main Menu and following the instructions on the screen.

Each adapter contains a 16-bit adapter ID and one to four 8-bit POS registers. Adapter IDs and the POS information are stored in CMOS RAM by the Set Configuration program. The CMOS RAM locations used to hold this information are not the same for all systems. The Set Configuration program determines the system type and handles differences between systems such as the CMOS RAM storage and the number of available adapter slots. If the system is identified as having only a 64-byte CMOS RAM, adapter IDs and POS data are stored in the 64-byte CMOS RAM. Each adapter slot is allocated 2 bytes for an adapter ID and 4 bytes for POS data. If the system type is identified as having a 2KB CMOS RAM extension, adapter IDs and POS data are stored in the 2KB CMOS RAM.

Automatic Configuration Utility

Automatic Configuration can be run after a configuration error has occurred or by selecting Run Automatic Configuration from the Set Configuration menu. Each time the system is powered on, the POST compares the configuration of the system to the configuration indicated by CMOS RAM. If differences between the two are detected, an error is displayed and logged in system RAM. POST error message files contained on the Reference Diskette display text that provides further information about the POST error.

Note: The Reference Diskette must be installed when the system is powered on to receive POST error messages.

If a configuration error is caused by a battery failure or a bad cyclic-redundency check (CRC), Automatic Configuration is run immediately after the POST error is displayed. If the error is caused by a change to the configuration, the user is given a choice to either run Automatic Configuration or continue to the Main Menu. If the user continues with the Main Menu, the changed areas of the system are configured and CMOS RAM is updated when Set Configuration is selected from the Main Menu.

Depending on the source of the error, Automatic Configuration either reconfigures the entire system or configures only the areas of the system that have been changed since the last time a configuration was performed. The following POST errors cause the system to be completely reconfigured:

- 161 (battery failure)
- 162 (bad CMOS CRC).

The following POST errors cause only the areas of the system that have changed to be reconfigured:

- 162 (system configuration error not caused by a bad CMOS CRC)
- 164 (memory configuration)
- 165 (adapter configuration).

An adapter is considered previously configured when the POS data stored in CMOS RAM matches a POS setting in the appropriate ADF. If an ADF for an installed adapter cannot be found, the adapter is configured as an empty slot.

During Automatic Configuration, devices are configured to the first nonconflicting values as defined in the ADF. Adapters are configured in the order of the channel position in which they are installed. The system board is configured first, followed by each adapter slot starting with slot 1. If the interrupt level is the only resource defined for a specific adapter item, the choice of interrupt levels that are least used by other adapters are assigned.

Automatic Configuration does not backtrack to previously configured adapters to resolve resource conflicts. If conflicts can be resolved, they must be done by choosing a nonconflicting resource option through the Change Configuration utility. Any adapter having a resource conflict that cannot be resolved by the Set Configuration program is disabled; the program sets bit 0 of POS Register 2 (hex 0102) to 0 in CMOS RAM.

Change Configuration Utility

The Change Configuration utility allows the user to change the default configuration settings from those set by Automatic Configuration. This utility is used to resolve unusual conflicts or to set items for personal preference.

The user interface is through scrolling and paging screens. Changes are made by rotating field value names through a set of choices using the F5 (Previous) and F6 (Next) keys. Changes are not saved in CMOS until the F10 (Save) key is pressed. Help text is provided for each item by pressing the F1 (Help) key.

Resource conflicts are indicated by an asterisk (*) next to the conflicting items and also by the "* Conflicts" string in the upper right corner of the Change Configuration window. Conflicts with fixed resources have the asterisk (*) to the left of the slot number of the adapter. Adapters with conflicts are disabled; the program sets bit 0 of POS Register 2 (hex 0102) to 0 in CMOS RAM.

View Configuration Utility

The View Configuration utility is provided to view the configuration. This is the Change Configuration utility with the change capabilities disabled.

Backup and Restore Configuration Utilities

The Backup Configuration utility provides a means to back up the configuration data to a file on the Reference Diskette. If the battery fails or the battery is changed, the user can use the Restore utility to restore the configuration.

Note: A copy of the Reference Diskette that is not write-protected is needed for this backup and restore process.

Copy an Option Diskette Utility

The Copy an Option Diskette utility is a separate program from the Set Configuration program and is accessible through the Main Menu. This utility is used to merge the following files from an option diskette onto a backup copy of the Reference Diskette: *.adf, *.dgs, *.pep, COMMAND.COM, DIAGS.COM, CMD.COM, and SC.EXE. The files found on the option diskette are compared to the files on the backup copy of the Reference Diskette. If the file does not exist on the Reference Diskette, it is copied. If the file already exists on the Reference Diskette, the dates of the two files are compared. The file on the option diskette is copied only if it has a date later than the corresponding file on the Reference Diskette.

The option diskette must be a DOS formatted diskette.

Adapter Description Files

Adapter description files provide POS information and system resource information for Automatic Configuration. The adapter description files also provide text for System Configuration utilities, help screens, and prompts. This section provides guidelines for developing the adapter description files.

Format

- File names: @CARDID.adf (high byte of the adapter ID first).
- Type of file: ASCII text.
- Not case sensitive: Key words can be lowercase, uppercase, or mixed. The case is preserved within the user interface text strings.
- Blanks, tabs, new lines: These are considered as white space and ignored, except when in text strings for the Change Configuration user interface.
- Comments: Lines beginning with semicolons are comments and are ignored.

Syntax

The following figure shows the meaning of special symbols used in the adapter description file syntax.

Symbol	Meaning
{ }	an optional item
{ }*	0,1,2,... items allowed
{ }+	1,2,3,... items allowed
x | y	either x or y allowed
x{n}	n x's required
[0-9]+	one or more decimal digits

Figure 6. Syntax Symbol Key

```
adf_file => card_id card_name nbytes {fixed_resources} {named_item}*
```

This defines the contents of an ADF. The following definitions describe each portion of an ADF in detail.

```
card_id => AdapterId number
```

Each ADF must contain a card_id. The character string 'AdapterId' is a keyword and must be present in the ADF. The Configuration program looks for this ID, which must match the ID used in the filename.

Example: `AdapterId 0DEFFh`

```
card_name => AdapterName string
```

Each ADF must contain a card_name. The character string 'AdapterName' is a keyword and must be present in the ADF. The string following the 'AdapterName' keyword is displayed as the adapter name in the Change Configuration and View Configuration screens. The length of the AdapterName string is limited to (74 - (length of 'SlotX - ')) characters.

(US English length = 66 characters)

Example: `AdapterName "IBM Multi-Protocol Communications Adapter"`

```
nbytes => NumBytes number
```

Each ADF must contain an nbytes. The character string 'NumBytes' is a keyword and must be present in the ADF. This is used to define the number of POS bytes used by the adapter. The number of POS bytes used should include all bytes from POS [0] to the last POS byte used. (If POS [3] is the only POS byte defined, NumBytes should be set to 4.)

Example: `NumBytes 4`

```
fixed_resources => FixedResources pos_setting resource_setting
```

A fixed_resources is not a requirement for ADFs. It is used to define resources required by an adapter and corresponding POS data. The character string 'FixedResources' is a keyword and must be present in the ADF only if the adapter needs to define resources that it requires.

Example: `FixedResources  POS[1]=XXXX01XXb int 3`

Figure 7 (Part 1 of 5). Adapter Description File Syntax

```
named_item => NamedItem prompt {named_choice}+  help
```

A named_item is not a requirement for ADFs. The named_item is used to define a field providing a choice of one or more resources. Each choice sets specified POS bits to a unique setting used to identify resources assigned to the adapter. The character string 'NamedItem' is a keyword and must be present in the ADF only if the adapter can be configured to use different resources. The adapter determines the resources it is configured to by how the POS bytes are set. When a 'NamedItem' is defined in an ADF it must be accompanied by a prompt (defined later), at least one named_choice (defined later), and help (defined later). Example:

```
NamedItem
Prompt "Communications Port"
choice "SDLC_1"   pos[0]=XXX1000Xb  io 0380h-038ch  int 3 4
choice "SDLC_2"   pos[0]=XXX1001Xb  io 03a0h-03ach  int 3 4
choice "BISYNC_1" pos[0]=XXX1100Xb  io 0380h-0389h  int 3 4
choice "BISYNC_2" pos[0]=XXX1101Xb  io 03a0h-03a9h  int 3 4
choice "SERIAL_1" pos[0]=XXX0000Xb  io 03f8h-03ffh  int 4
choice "SERIAL_2" pos[0]=XXX0001Xb  io 02f8h-02ffh  int 3
choice "SERIAL_3" pos[0]=XXX0010Xb  io 3220h-3227h  int 3
choice "SERIAL_4" pos[0]=XXX0011Xb  io 3228h-322fh  int 3
choice "SERIAL_5" pos[0]=XXX0100Xb  io 4220h-4227h  int 3
choice "SERIAL_6" pos[0]=XXX0101Xb  io 4228h-422fh  int 3
choice "SERIAL_7" pos[0]=XXX0110Xb  io 5220h-5227h  int 3
choice "SERIAL_8" pos[0]=XXX0111Xb  io 5228h-522fh  int 3
Help
"This port can be assigned as a:  primary (SDLC1) or
secondary (SDLC2) sdlc, primary (BISYNC1) or secondary
(BISYNC2) bisync, or as a serial port (Serial 1 through Serial
8).  Use the F5=Previous and the F6=Next keys to change this
assignment in the 'Change configuration' window.  Conflicting
assignments are marked with an asterisk and must be changed
to use the adapter."
```

```
prompt => Prompt string
```

The prompt is required when a NamedItem is defined. The prompt is used to define a title for a NamedItem field. The character string 'Prompt' is a keyword and must be present in the ADF when there is a NamedItem present. The string following the 'Prompt' keyword appears after the adapter name in the Change Configuration and View Configuration screens. Following the prompt string is a field that can be toggled in the Change Configuration screen if two or more named_items are defined. The length of the prompt string cannot exceed 38 characters. Example: (See the example for named_item).

Figure 7 (Part 2 of 5). Adapter Description File Syntax

```
named_choice => Choice choice_name pos_setting resource_setting
```

At least one named_choice is required when a NamedItem is defined. The character string 'Choice' is a keyword and must be present in the ADF when there is a NamedItem present. One or more named_choices must follow a prompt. Each named_choice must contain a string describing the current choice in the prompt field. Each named_choice must define a pos_setting, for at least one POS byte, which will uniquely identify the resource_setting defined in the named_choice. The length of the named_choice string cannot exceed 28 characters.
Example: (See the example for named_item).

```
help => Help string
```

The help is a string of text used to give the user assistance at a prompt. This text is displayed in the Change Configuration and View Configuration screen when the cursor is at the associated prompt and the F1 key is pressed. The character string 'Help' is a keyword and must be present in the ADF when there is a NamedItem present. The string following the keyword 'Help' is the text describing the prompt defined in the same NamedItem as the help. The length of the help string is limited to 1000 characters.
Example: (See the example for named_item).

```
pos_setting => {pos_byte_setting}+
```

The pos_setting must contain at least one pos_byte_setting. See the definition of pos_byte_setting for more information.

```
pos_byte_setting => pos[number]=pos_bit{8}b
```

This is the definition of the pos_byte_setting in the ADF.
The character string 'pos' is a keyword and must be present in a pos_byte_setting, followed by a number in brackets. The number in brackets refers to the following POS bytes:

number = 0, POS byte at port 102h
number = 1, POS byte at port 103h
number = 2, POS byte at port 104h
number = 3, POS byte at port 105h

The end bracket must be followed by an equal sign and then a bit definition of the POS byte (See pos_bit for information on the bit definition). The bit definition must define all 8 bits of the byte.
Bit 0 of pos[0] should always be defined as X in the ADF.
Example: `pos[0]=XXX1001Xb`

Figure 7 (Part 3 of 5). Adapter Description File Syntax

pos_bit => x | X | 0 | 1

A pos_bit can be defined as a mask bit (x or X), a clear bit (0), or a set bit (1).

Example: pos[0]=XXX1001Xb

resource_setting =>
    {ioblock_list} {interrupt_list} {arb_list} {memaddr_list}

The resource_setting defines a list of system resources. They may be fixed resources required by the adapter or they may be resources the adapter uses when configured to a specific choice in a named_item. The resources can consist of the following:

Range of I/O addresses (limited to 16).
List of interrupt levels (limited to 16).
List of arbitration levels (limited to 16).
Range of memory addresses (limited to 2).

Example: (See the following resource definitions).

ioblock_list => IO {range}+

The ioblock_list must be a list of one or more ranges of I/O addresses. The character string 'IO' is a keyword and must be present in the ioblock_list.

Example: io 4220h-4227h

Interrupt_list => INT {number}+

The interrupt_list must be a list of one or more interrupt levels. The character string 'INT' is a keyword and must be present in the interrupt_list.

Example: INT 3 4

arb_list => ARB {number}+

The arb_list must be a list of one or more arbitration levels. The character string 'ARB' is a keyword and must be present in the arb_list.

Example: ARB 1

memaddr_list => MEM {range}+

The memaddr_list must be a list of one or more ranges of RAM or ROM addresses. The character string 'MEM' is a keyword and must be present in the memaddr_list. This keyword is used to allocate memory address space in the hex 000C0000 through hex 000DFFFF range.

Example: MEM 0CC000h - 0CDFFFh

range => number - number

Figure 7 (Part 4 of 5). Adapter Description File Syntax

```
number => [0-9]+ {d} | [0-9a-f]+ h | [0-9A-F]+ H

string => " [ascii except for "]+ "
```

A string is a set of ASCII characters beginning with a double quote (") and ending with a double quote.
Example:

```
"This port can be assigned as a:  primary (SDLC1) or
secondary (SDLC2) sdlc port, primary (BISYNC1) or secondary
(BISYNC2) bisync port, or as a serial port (Serial 1 through
Serial 8).  Use the F5=Previous and the F6=Next keys to change
this assignment in the 'Change configuration' window.
Conflicting assignments are marked with an asterisk and must
be changed to use the adapter."
```

Figure 7 (Part 5 of 5). Adapter Description File Syntax

Example

The following is an example of an adapter description file for the IBM Personal System/2 Multiprotocol Communications Adapter/A. The name of the file for this adapter is @DEFF.adf. An explanation of each numbered item begins on page 21.

```
AdapterId 0DEFFh 1

AdapterName "IBM Multiprotocol Communications Adapter" 2

NumBytes 2 3

NamedItem 4

Prompt "Communications Port"

choice "SDLC_1"   pos[0]=XXX1000Xb  io 0380h-038ch  int 3 4
choice "SDLC_2"   pos[0]=XXX1001Xb  io 03a0h-03ach  int 3 4
choice "BISYNC_1" pos[0]=XXX1100Xb  io 0380h-0389h  int 3 4
choice "BISYNC_2" pos[0]=XXX1101Xb  io 03a0h-03a9h  int 3 4
choice "SERIAL_1" pos[0]=XXX0000Xb  io 03f8h-03ffh  int 4
choice "SERIAL_2" pos[0]=XXX0001Xb  io 02f8h-02ffh  int 3
choice "SERIAL_3" pos[0]=XXX0010Xb  io 3220h-3227h  int 3
choice "SERIAL_4" pos[0]=XXX0011Xb  io 3228h-322fh  int 3
choice "SERIAL_5" pos[0]=XXX0100Xb  io 4220h-4227h  int 3
choice "SERIAL_6" pos[0]=XXX0101Xb  io 4228h-422fh  int 3
choice "SERIAL_7" pos[0]=XXX0110Xb  io 5220h-5227h  int 3
choice "SERIAL_8" pos[0]=XXX0111Xb  io 5228h-522fh  int 3

Help

"This port can be assigned as a: primary (SDLC1) or
secondary (SDLC2) sdlc port, primary (BISYNC1) or secondary
(BISYNC2) bisync port, or as a serial port (Serial 1 through
Serial 8).  Use the F5=Previous and the F6=Next keys to
change this assignment.  Conflicting assignments are marked
with an asterisk and must be changed."
```

```
NamedItem 5

Prompt "Arbitration Level for SDLC"

choice "Level_1"  pos[1]=XXXX0001b  arb  1
choice "Level_0"  pos[1]=XXXX0000b  arb  0
choice "Level_2"  pos[1]=XXXX0010b  arb  2
choice "Level_3"  pos[1]=XXXX0011b  arb  3
choice "Level_4"  pos[1]=XXXX0100b  arb  4
choice "Level_5"  pos[1]=XXXX0101b  arb  5
choice "Level_6"  pos[1]=XXXX0110b  arb  6
choice "Level_7"  pos[1]=XXXX0111b  arb  7
choice "Level_8"  pos[1]=XXXX1000b  arb  8
choice "Level_9"  pos[1]=XXXX1001b  arb  9
choice "Level_10" pos[1]=XXXX1010b  arb 10
choice "Level_11" pos[1]=XXXX1011b  arb 11
choice "Level_12" pos[1]=XXXX1100b  arb 12
choice "Level_13" pos[1]=XXXX1101b  arb 13
choice "Level_14" pos[1]=XXXX1110b  arb 14

Help

"This assignment need only be changed if it is in conflict
with another assignment.  Conflicting assignments are marked
with an asterisk.  Use the F5=Previous and the F6=Next keys
to change arbitration level assignments.  Using arbitration
levels, this adapter accesses memory directly without
burdening the computer's main microprocessor.  An
arbitration level of 0 has the highest priority, and
increasing levels have corresponding decreased priority"
```

1 The card_id for this adapter is hex 0DEFF. This is an ASCII representation of the ID generated by the adapter. The high byte is followed by the low byte. The card_id is required for all ADFs.

2 The card_name is “IBM Multiprotocol Communications Adapter.” The card_name is required for all ADFs.

3 The nbytes (NumBytes 2) in this file indicates the adapter uses two POS bytes located at hex 0102 and 0103.

4 This is the first named_item for the adapter. The title of the field is “Communications Port.” The user can toggle between the 12 named_choices. Each named_choice has a unique pos_setting assigned to it in bit locations 1 through 4 of POS byte hex 0102 (pos [0]). Also shown is a resource_setting that corresponds to the pos_setting of the named_choice. The resources allocated in this named_item are I/O addresses and interrupt levels. A help string for this named_item is provided below the last named_choice.

5 This is the second named_item for the adapter. The title of the field is “Arbitration Level for SDLC.” The user can toggle between the 15 named_choices. Each named_choice has a unique pos_setting assigned to it in bit locations 0 through 3 of POS byte hex 0103 (pos [1]). Also shown is a resource_setting that corresponds to the pos_setting of the named_choice. The resources allocated in this named_item are arbitration levels. A help string for this named_item is provided below the last named_choice.

Index

A

adapter configuration, order of 12
adapter description files 1
adapter ID number 1
adapter identification 6, 10
adapter POS implementation 7
adapter setup 4, 7
address decode 3
address space, POS 3
arbitration level field 6
automatic configuration 1, 4, 11

B

backup configuration utility 13
battery failure 11
bus contention 2

C

card enable 4
card selected feedback register 3
CD SETUP 4
CD SFDBK 3
change configuration utility 12
channel-check active indicator 5
channel-check status indicator 5
CMOS RAM, card ID bytes 10
configuration error 11
configuration utilities 6
copy an option diskette utility 13
CRC error 11

D

device ROM segment address field 6

F

fairness enable bit 6
fairness feature 6

H

help text 12

I

I/O address decode 4
I/O device address field 6
initial program load 8

P

POS I/O address space 3, 4
POS overview 1
POS registers 3
POST error message files 11
power-on self-test 1

R

reference diskette 10
restore configuration utility 13

S

SAD00 - SAD15 8
set configuration program 10
setup procedures 2
setup, system board 3
subaddressing bits 8
subaddressing extension 8
system board setup 3
system configuration utilities 2, 10
system resources 10

V

view configuration 13

Micro Channel Adapter Design

General Guidelines 1
Dimensions 1
Power 19
Voltage Regulation 20
General Design Considerations 20
  Safety 20
  Thermal 21
  Electromagnetic Compatibility 21
  Diagnostics 22
Design Guidelines 22

Index 25

Figures

1. Adapter Dimensions (8- or 16-Bit) 2
2. Connector Dimensions (8- or 16-Bit) 3
3. Adapter Dimensions (8- or 16-Bit with Video Extension) . . . 4
4. Connector Dimensions (8- or 16-Bit with Video Extension) . . 5
5. Adapter Dimensions (32-Bit) 6
6. Connector Dimensions (32-Bit) 7
7. Adapter Dimensions (32-Bit with Matched Memory) 8
8. Connector Dimensions (32-Bit with Matched Memory) 9
9. Connector (Common Detail) 10
10. Typical Adapter Assembly 11
11. Adapter Holder 12
12. Adapter Retainer 13
13. Adapter Bracket 15
14. Channel Load Current 19
15. Channel Voltage Regulation 20
16. Vendor ID Assignments 23

General Guidelines

This section provides some basic guidelines to design adapters for the Micro Channel architecture 16- and 32-bit products. Topics include physical specifications, power requirements and limitations, and configuration program support.

The system board provides channel connectors to support the following types of adapters. Some systems do not support all types of adapters. See the system-specific technical references for more information.

- 16-bit adapter
- 16-bit adapter with video extension
- 32-bit adapter
- 32-bit adapter with matched-memory extension.

Connector contacts are not required for signals not used by an adapter. See “Micro Channel Architecture” for more information on channel connectors and signals.

Dimensions

The following figures show the dimensions of each type of adapter and the associated mounting hardware. The tolerances shown include all individual process tolerances and are not cumulative. The maximum height for components mounted on the adapter is 15 millimeters (0.6 inch) on the component (A) side. The maximum height for pins and components on the B side of the adapter is 2 millimeters (0.078 inch). Adapters using CMOS technology should have all plated connector contacts the same length to reduce the exposure of incorrect bias to modules.

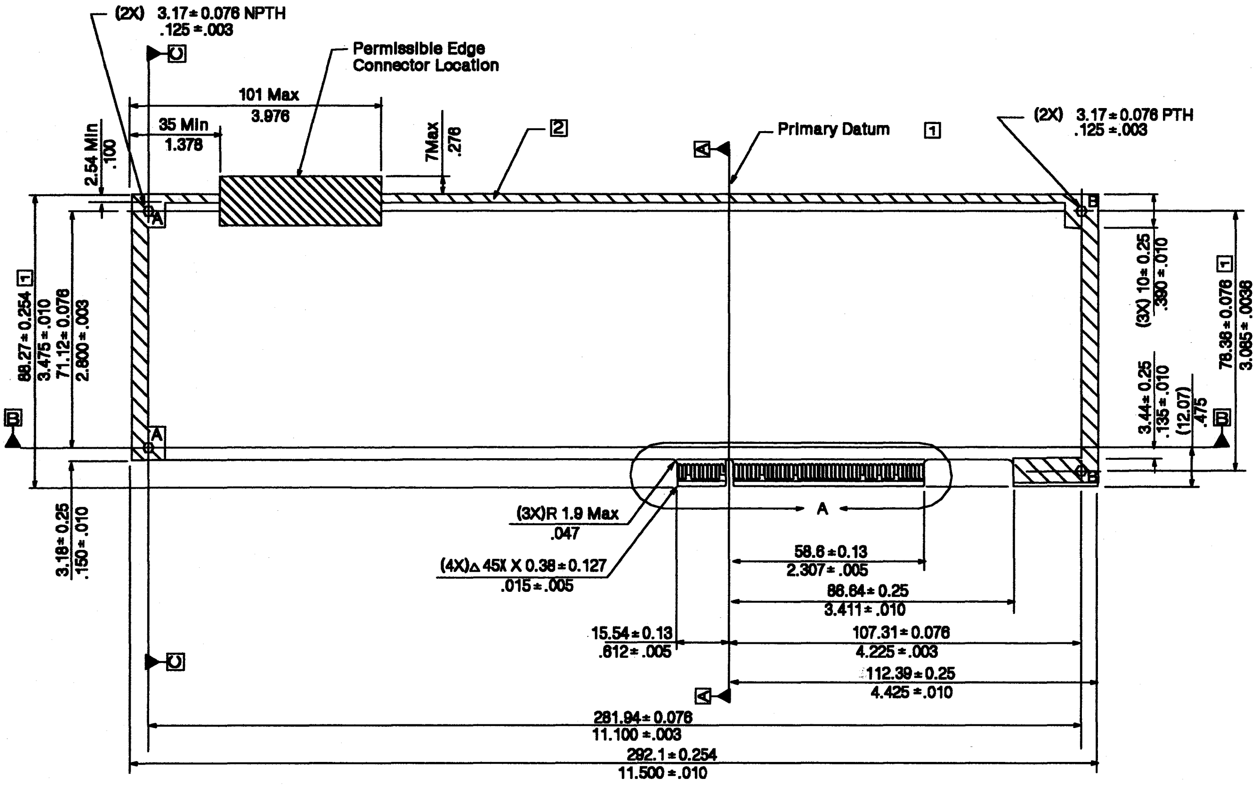
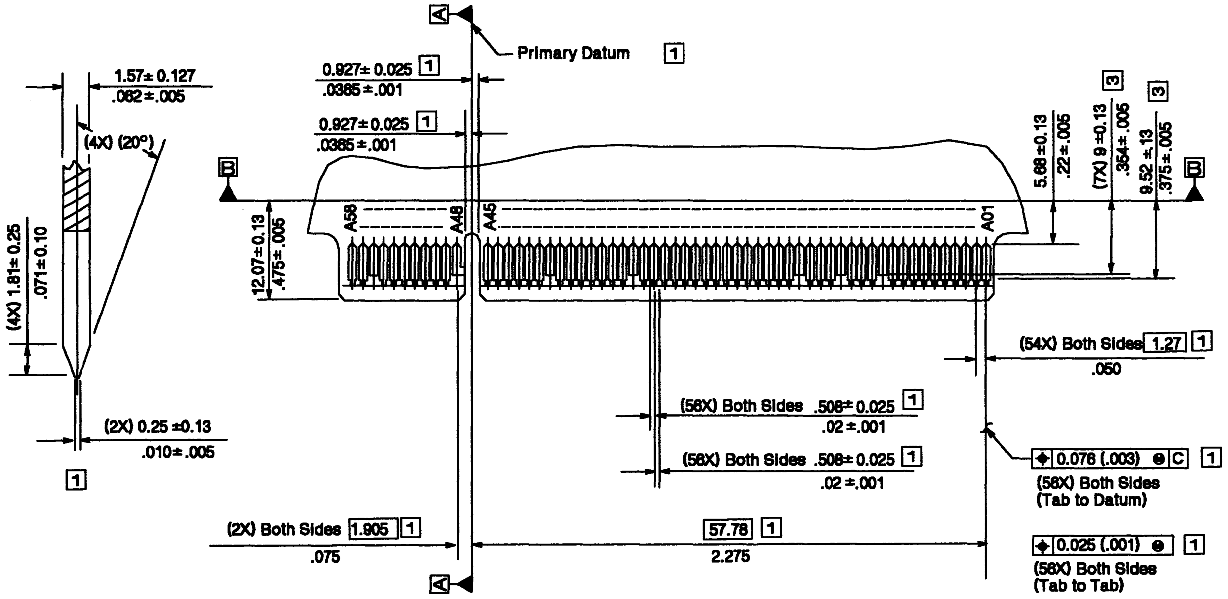


Figure 1. Adapter Dimensions (8- or 16-Bit)



1 Dimensions Critical to Function

2 Component Free Area Both Sides

3 Cards Using CMOS Technology Should Make All Card Tabs the Same Length to Reduce the Exposure of Incorrect Bias to the Modules

Figure 2. Connector Dimensions (8- or 16-Bit)

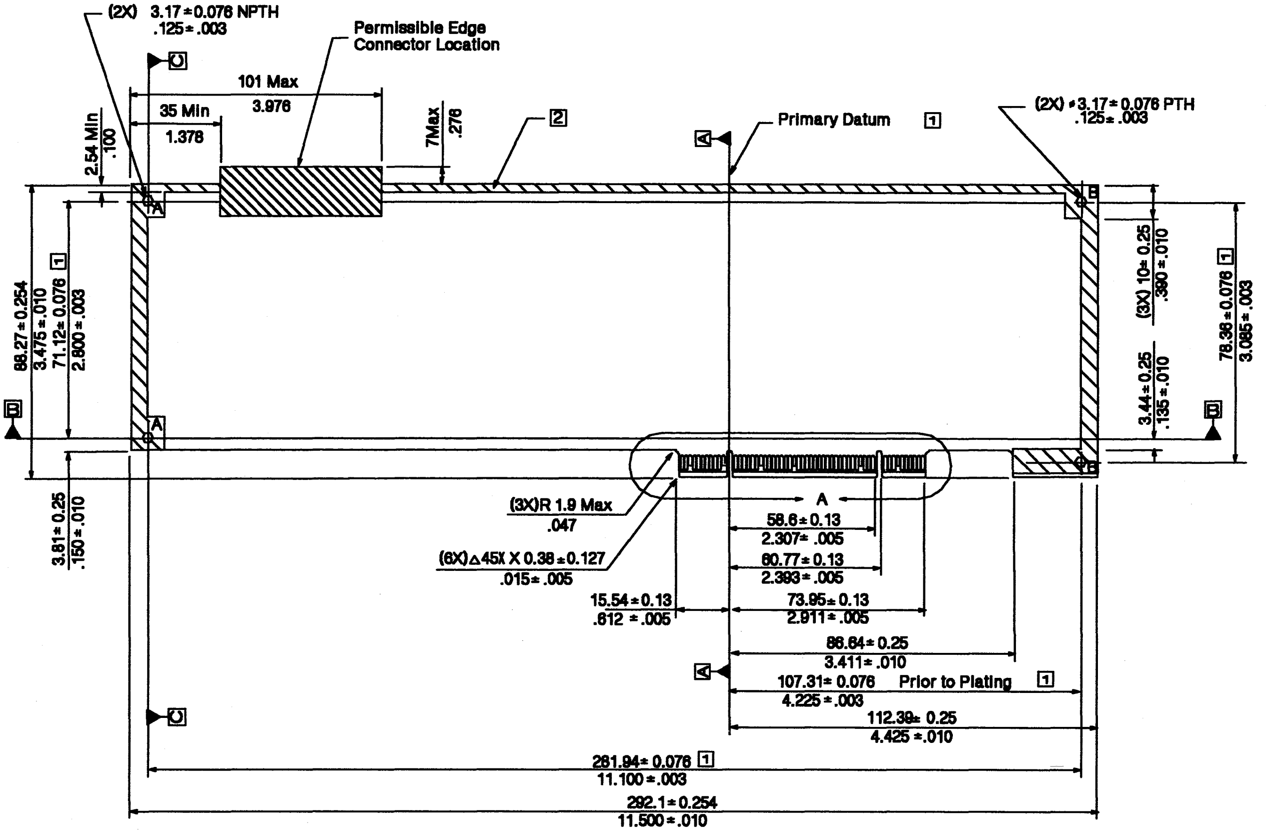
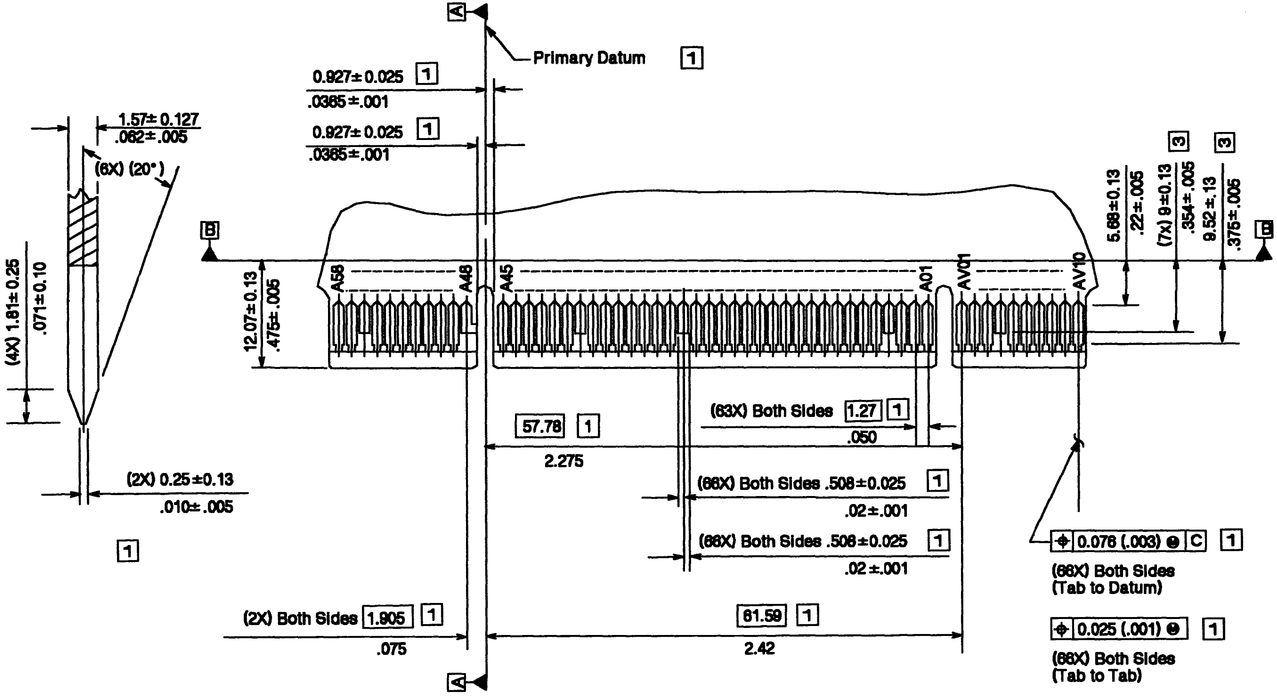


Figure 3. Adapter Dimensions (8- or 16-Bit with Video Extension)

Figure 4. Connector Dimensions (8- or 16-Bit with Video Extension)



1 Dimensions Critical to Function

2 Component Free Area Both Sides

3 Cards Using CMOS Technology Should Make All Card Tabs the Same Length to Reduce the Exposure of Incorrect Bias to the Modules

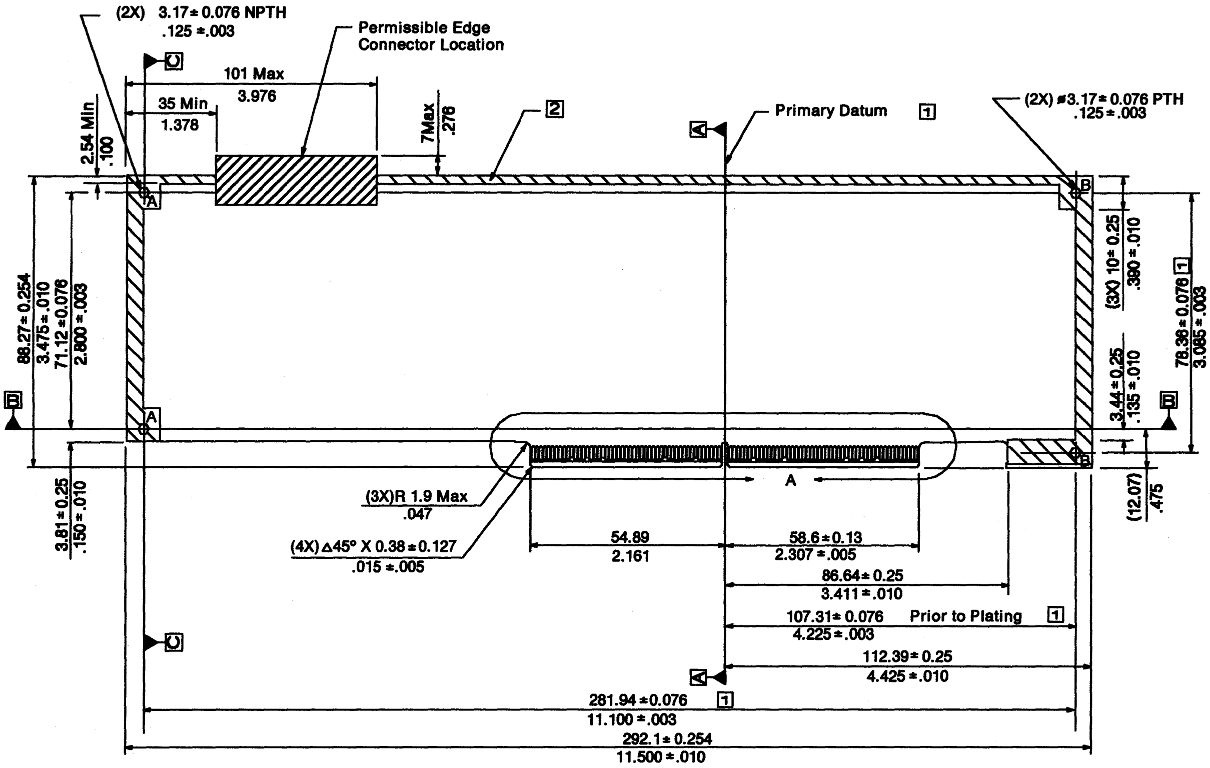
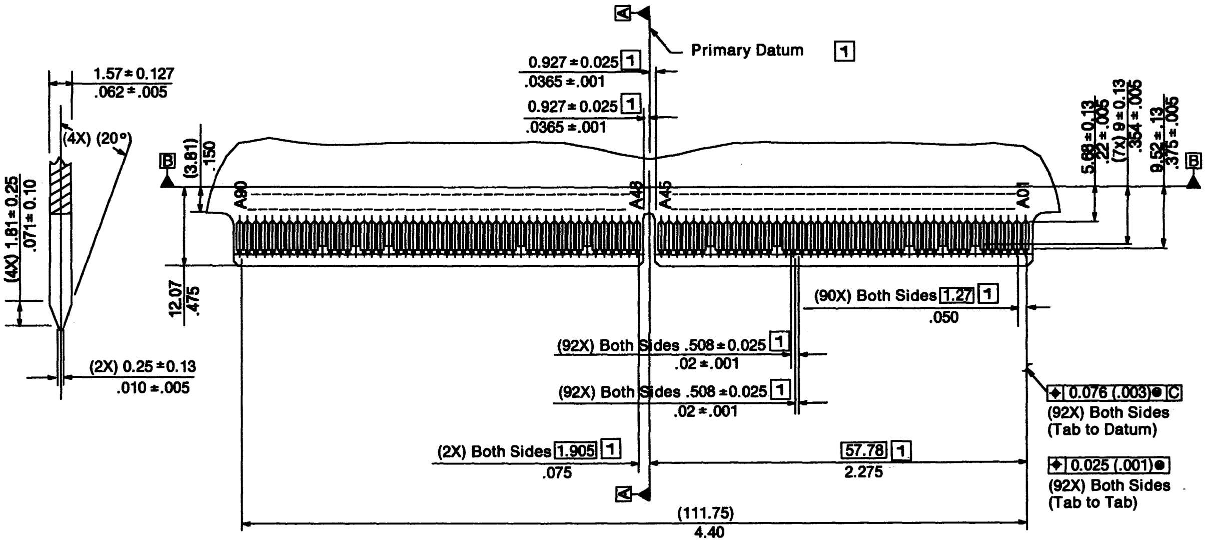


Figure 5. Adapter Dimensions (32-Bit)



1 Dimensions Critical to Function

2 Component Free Area Both Sides

3 Cards Using CMOS Technology Should Make All Card Tabs the Same Length to Reduce the Exposure of Incorrect Bias to the Modules

Figure 6. Connector Dimensions (32-Bit)

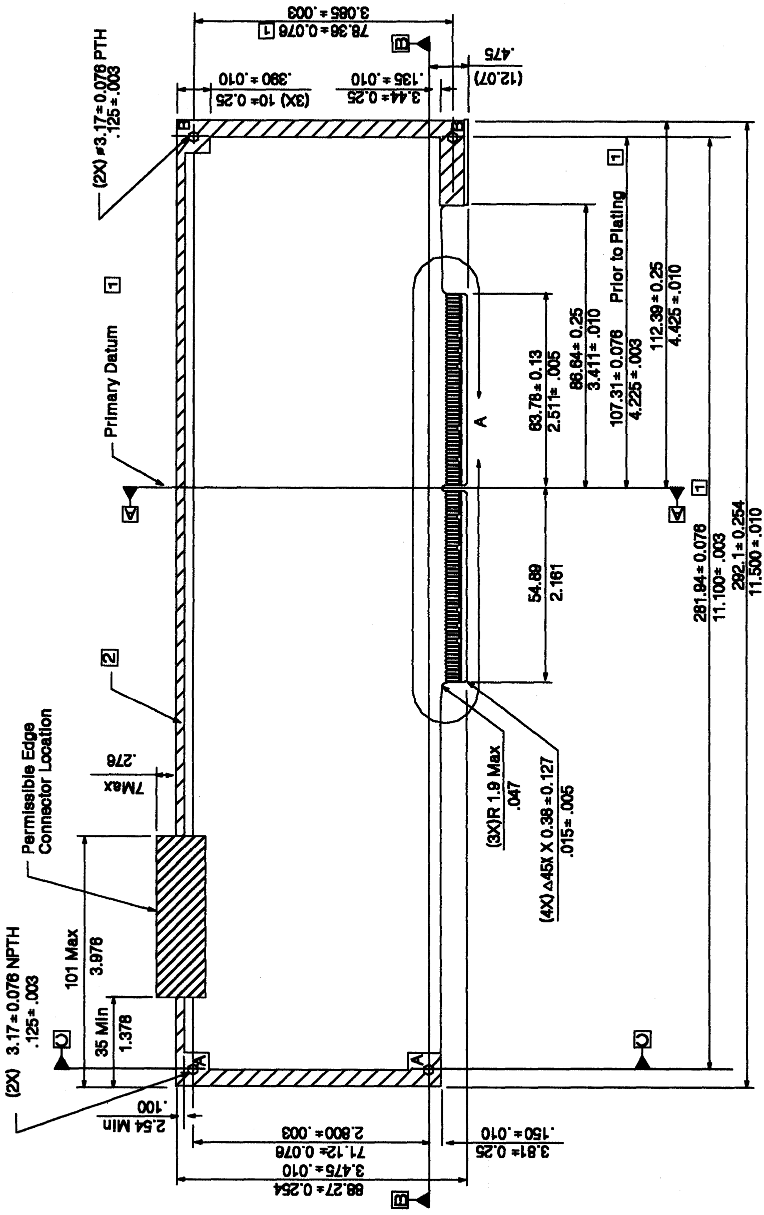
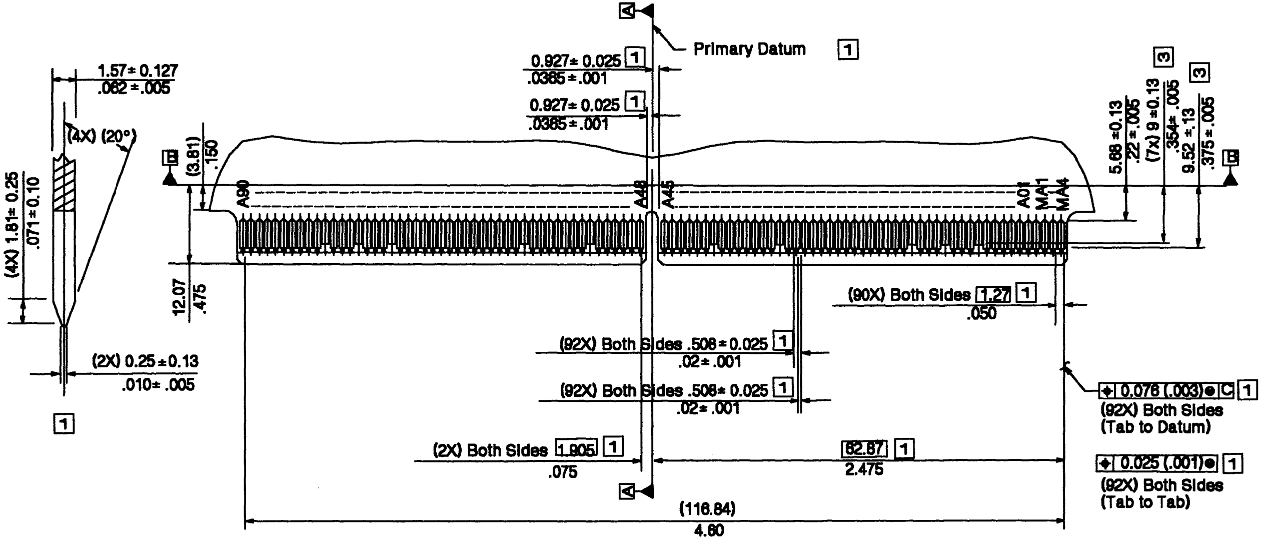


Figure 7. Adapter Dimensions (32-Bit with Matched Memory)

Figure 8. Connector Dimensions (32-Bit with Matched Memory)



1 Dimensions Critical to Function

2 Component Free Area Both Sides

3 Cards Using CMOS Technology Should Make All Card Tabs the Same Length to Reduce the Exposure of Incorrect Bias to the Modules

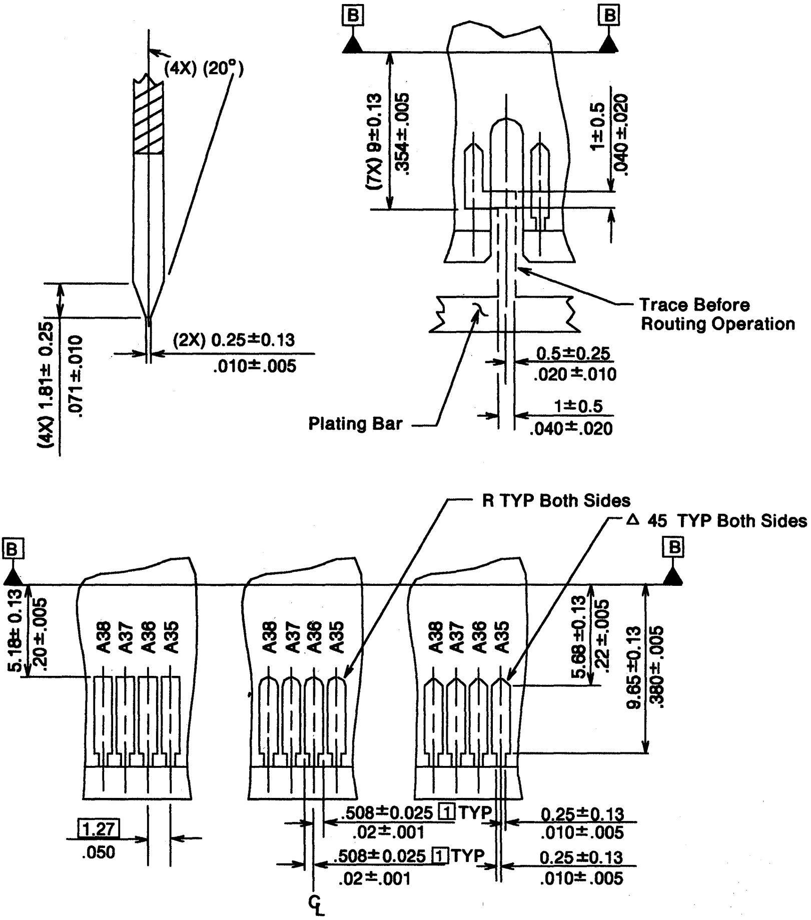
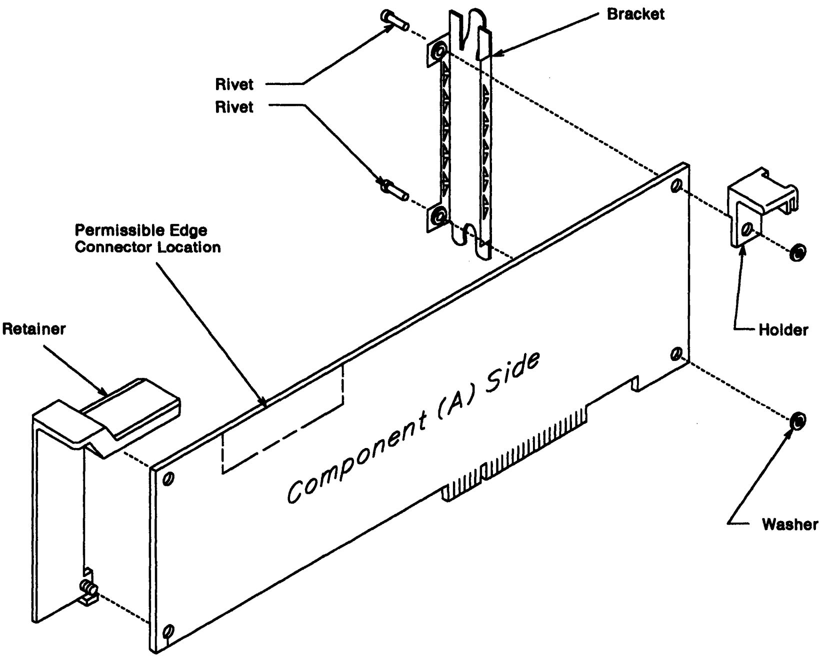


Figure 9. Connector (Common Detail)



Materials:
Holder and Retainer - Polycarbonate UL 94 V-0
Bracket - AISI Type 302 1/4 Hard Stainless Steel

Figure 10. Typical Adapter Assembly

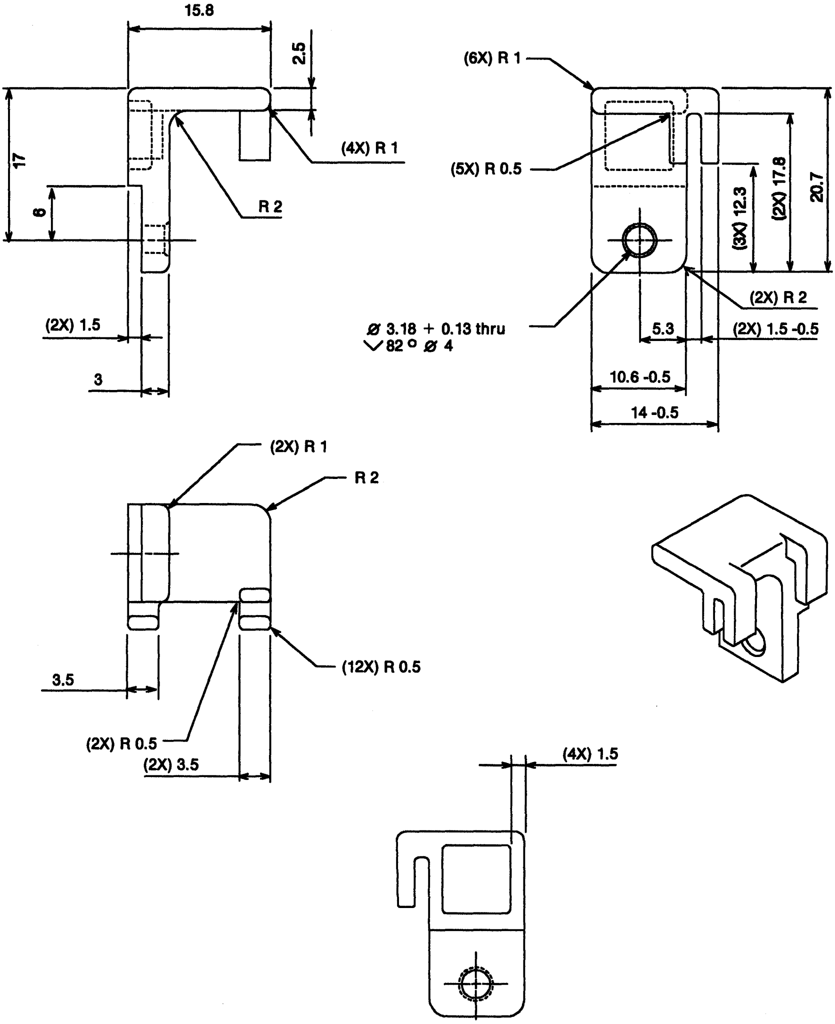


Figure 11. Adapter Holder

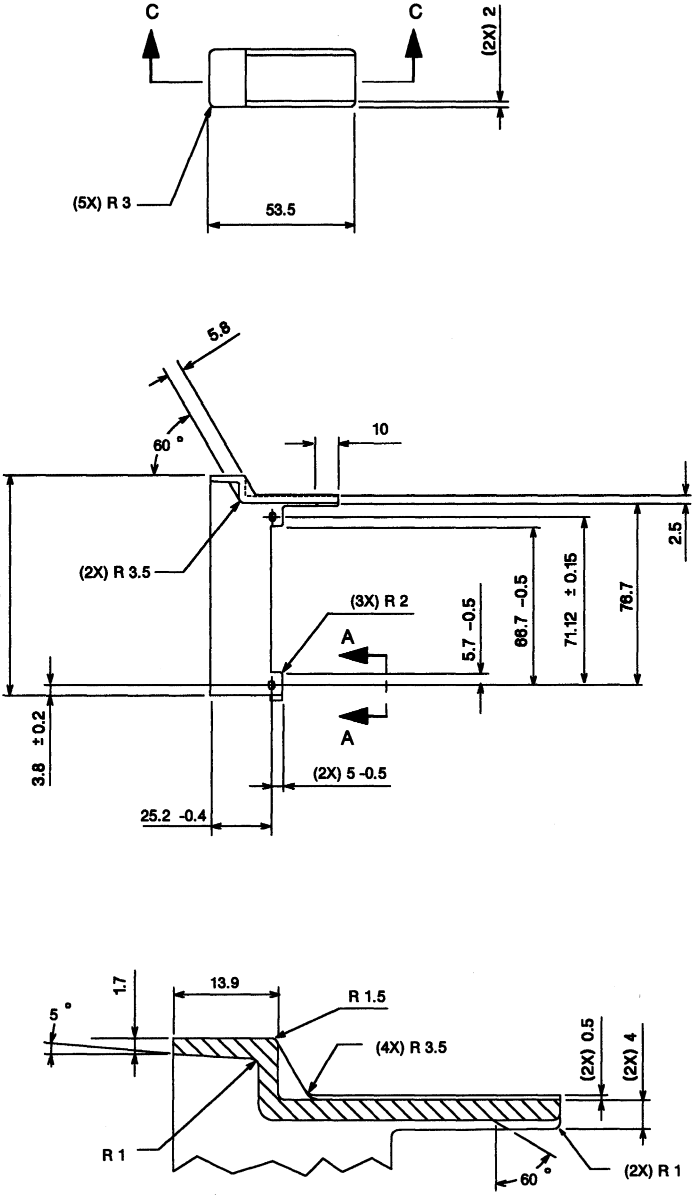
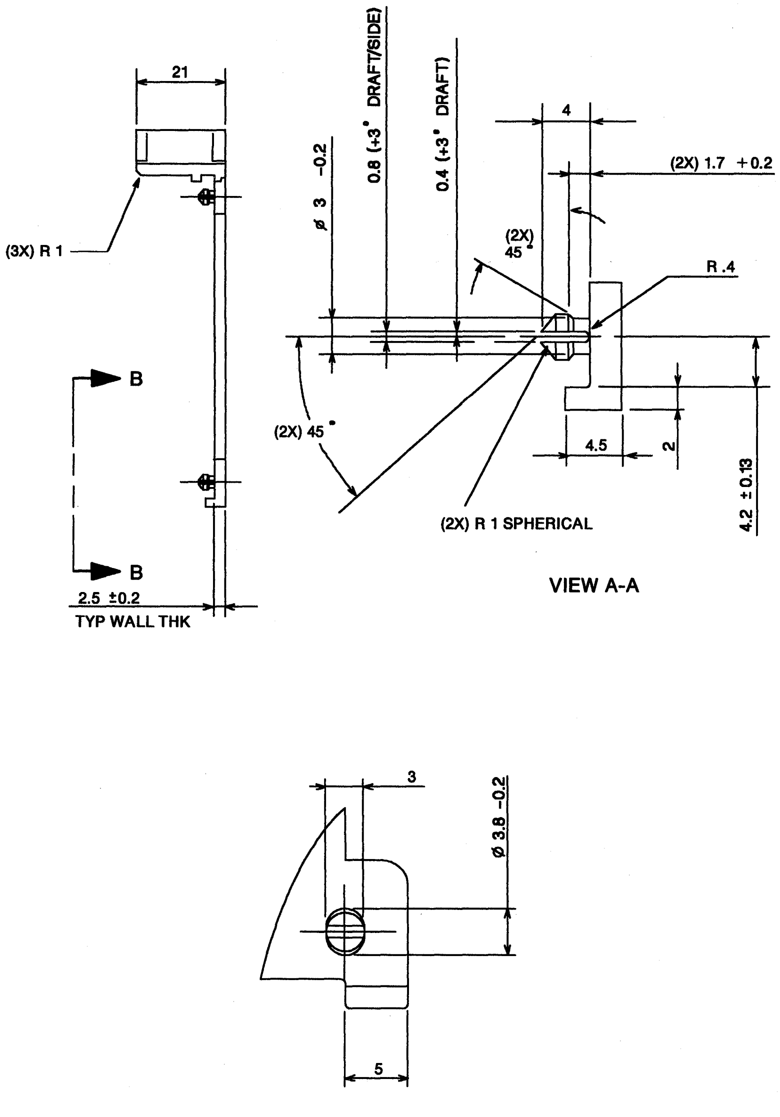


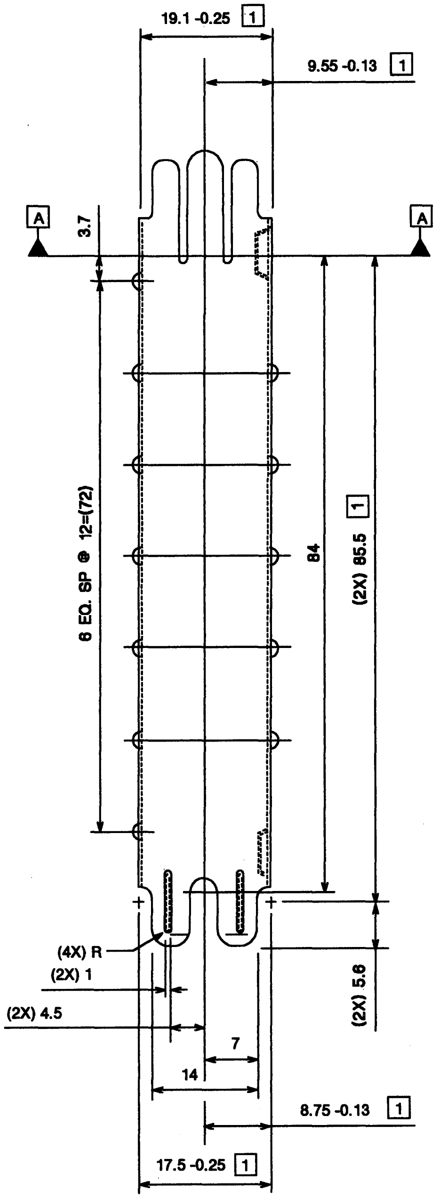
Figure 12 (Part 1 of 2). Adapter Retainer



VIEW A-A

VIEW B-B

Figure 12 (Part 2 of 2). Adapter Retainer



1 DIMENSION IS CRITICAL TO FUNCTION.

Figure 13 (Part 1 of 4). Adapter Bracket

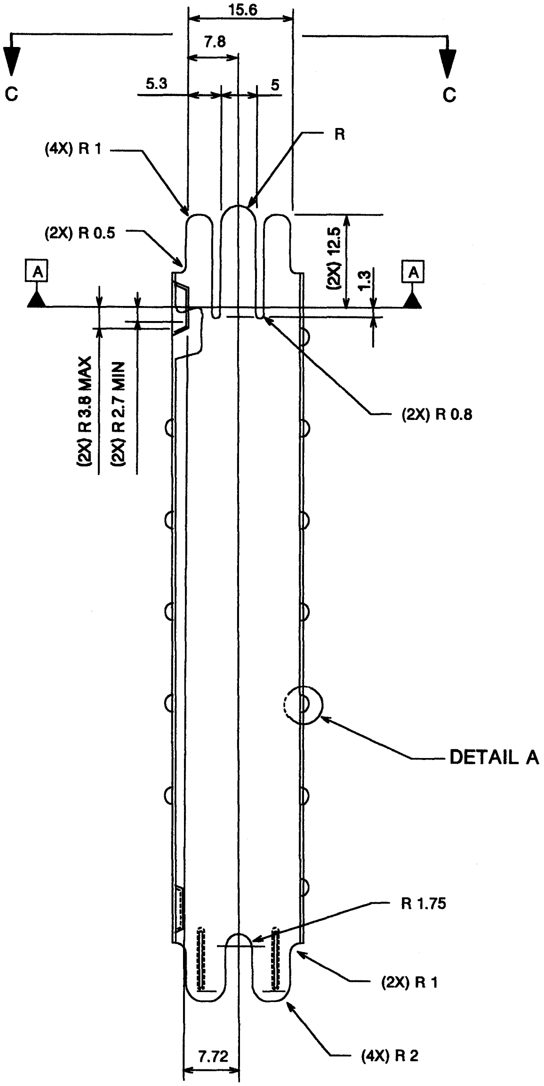


Figure 13 (Part 2 of 4). Adapter Bracket

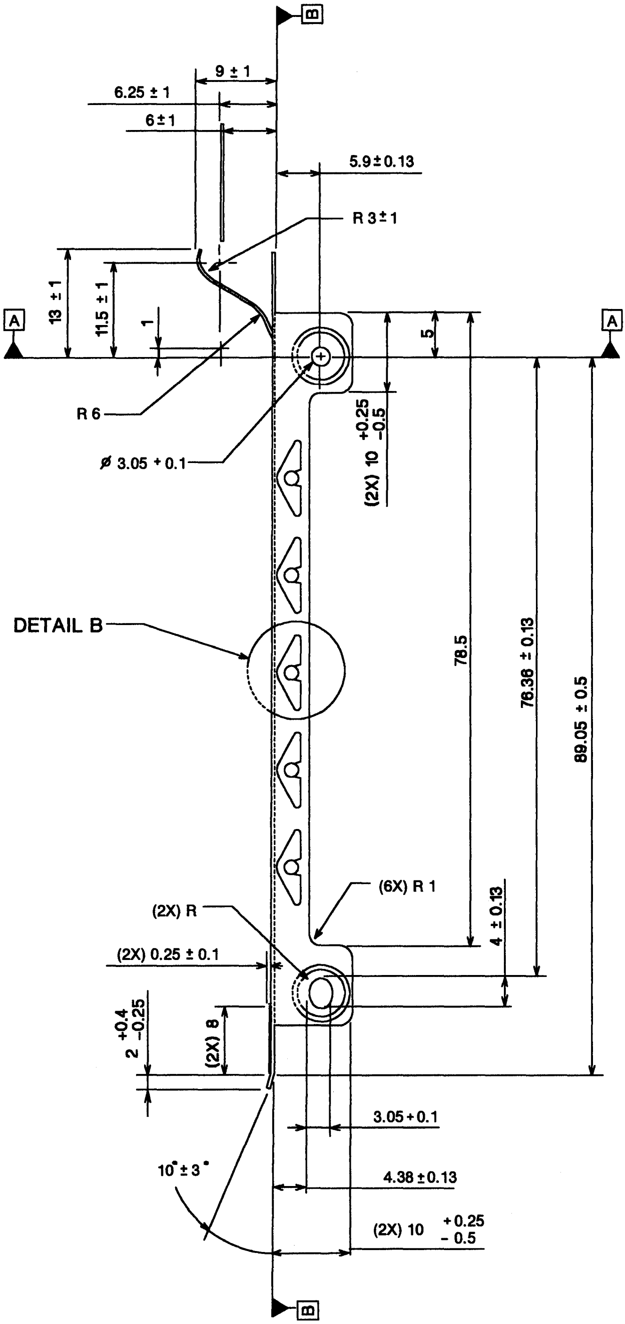
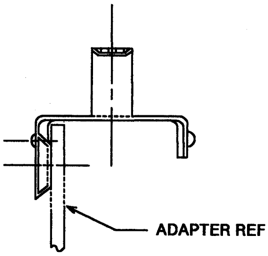
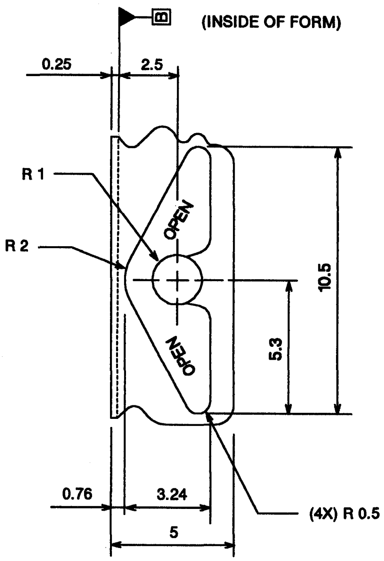


Figure 13 (Part 3 of 4). Adapter Bracket

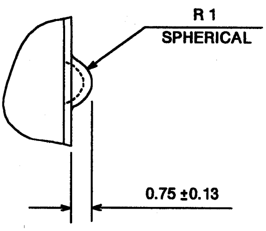


VIEW C-C



DETAIL B

SCALE 10/1
(12X)



DETAIL A

SCALE 10/1
(12X)

Figure 13 (Part 4 of 4). Adapter Bracket

Power

The allowable load current for each voltage present on each channel connector is as follows.

Supply Voltage	Maximum Current* Per 16-Bit Connector	Maximum Current Per 16/32-Bit Connector
+ 5.0 Vdc	1.6 A	2.0 A
+12.0 Vdc	0.175 A	0.175 A
−12.0 Vdc	0.040 A	0.040 A

* This maximum current may not apply to all systems.

Figure 14. Channel Load Current

The following formulas can be used to determine maximum statistical current values.

$$(I_{TC_1} + \cdots + I_{TC_n}) + \sqrt{((I_{MC_1} - I_{TC_1})^2 + \cdots + (I_{MC_n} - I_{TC_n})^2)}$$

Where:

- I_{MC} is the maximum current for each component on a given adapter.
- I_{TC} is the typical current for each component on a given adapter and the sum of all I_{TC} equals I_{TA}.

Note: If I_{MC} or I_{TC} is not available, estimate by using: $I_{TC} = 0.7 \times I_{MC}$

The total channel current is also determined in a statistical manner. Total Channel Current =

$$(I_{TA_1} + \cdots + I_{TA_n}) + \sqrt{((I_{MA_1} - I_{TA_1})^2 + \cdots + (I_{MA_n} - I_{TA_n})^2)}$$

Where:

- I_{MA} is a maximum statistical current for a given adapter.
- I_{TA} is the typical current for a given adapter.

Voltage Regulation

The voltage regulation at the channel connector is shown in the following figure.

Voltage	Pins	Tolerance
Ground	A3, B3, B5, B9, B13, B17, B21, B25, B29,	N/A
	B33, B37, B41, A43, B45, B50, B54, B58	N/A
	A61, B63, B67, B71, B75, B79, B83, B87, A89, BM4*, AM2*	N/A
+ 5.0 Vdc	A7, A11, A15, A31, A39, A48, A56, A69, A73, A81, A85	+5% −4.5%
+12.0 Vdc	A19, A35, A52, A65, A77,	+5% −4.5%
−12.0 Vdc	A23, A27	+10% −9.5%
Ground (Auxiliary Video)	BV1, AV3, BV5, AV7, BV9	N/A

* These connectors are in the matched-memory portion of the connector.

Figure 15. Channel Voltage Regulation

The tolerance includes all power distribution losses in both power and ground planes, up to the pins of the channel connector. It does not include the drop due to the connector (30 milliohm maximum per contact), or the drop due to distribution within the adapter.

General Design Considerations

Each designer must take the precautions necessary to protect the safety of the end user, provide reliable operation of the device, and ensure the device does not interfere with the operation of the system or any other installed devices. The design considerations described in this section are not the **only** considerations, but rather those that might otherwise be overlooked.

Safety

Avoid exposed high-voltage or current points, sharp edges, and exposed components that operate at high temperatures. Devices must not channel dc power outside the system unit in any manner that violates Underwriters Laboratory and Canadian Standards Association guidelines.

Note: Canadian Standards Association C22.2, paragraph 4.11.3, number 154 requires protection of conductors of external interconnecting cords and cables connected to secondary circuits.

IBM does not support installing or removing adapters or components when the system power is on.

Thermal

The system unit is cooled internally by low-volume forced air. Adapter designs must allow for adequate air space between the adapters. Avoid using internal cables as a mechanism for signal communication inside the system unit, they can interfere with the air flow. If internal cables are required, they must be positioned to minimize the impact on airflow. The maximum height for components mounted on the adapter should not exceed the dimensions specified under "Dimensions" on page 1. The adapter design should avoid clustering of high-temperature components. No component should exceed its maximum thermal rating.

Electromagnetic Compatibility

Adhere to the following guidelines to reduce electromagnetic compatibility (EMC) problems.

- The adapter end brackets make a continuous 360° connection to the outside "skin" of the system unit cabinet. A similar 360° connection to the inside skin should also be provided. The adapter bracket must not be used as a dc voltage return path, a logic-ground connection, or an audio ground connection.

- The end bracket at the rear of the adapter is isolated from dc ground on the adapter. The bracket must be grounded through a screw connection to the system unit and designed as shown in Figure 13 on page 15.

- All connector ground pins must be connected to the interplane ground at the channel connector, and the +5 Vdc power must be immediately connected to the +5 Vdc power plane.

- All adapters must provide nonsegmented internal power and ground planes.

- Each surface-mount technology module position should provide a decoupling capacitor pad with minimal connection inductance. Pin-in-hole modules should be decoupled if they drive or contain edge-triggered logic. Capacitors can range between 0.01 and 0.10 microfarad and should be low-inductance ceramic or a layered design.

- Internal cables should be avoided as a mechanism for signal communication inside the system unit. The channel should not be extended outside the system unit, except by an adapter.

- Clocks should be properly imbedded and terminated. When clocks, strobes, and handshakes are generated or received, care should be taken to control the rise-and-fall times to minimize radiation.

- External cables should connect through 360° shielded D-shell or equivalent connectors. Avoid the use of "pigtail" shield connections. Shield terminations should be connected to the external shield of the cable connector. Do not bring the shield through the connector and connect it to either logic ground or the inside skin of the cabinet.

- High-current power within the system unit should provide adjacent return paths to allow the maximum cancelation of radiated magnetic fields by the mutual coupling between the supply and return lines.

Diagnostics

All writable registers typically are readable at the same address. External interfaces typically include 100% diagnostic wrap capability by electronic switching or an external wrap tool.

Design Guidelines

Adapters designed for the Micro Channel architecture must comply with the following design guidelines:

- Each I/O adapter must decode all 16 lines of the I/O address.

- Each memory adapter must decode all 24 lines of the memory address and MADE 24.

- Each 32-bit memory adapter must decode all 32 lines of the memory address if MADE 24 is inactive.

- Each adapter must replace the function of switches and jumpers with registers that incorporate POS logic.

- Each adapter must issue an *adapter ID* to the data bus when interrogated.

  To minimize the number of required drivers, only the logical 0 bits in the adapter ID need to be driven. This provides 39,202 combinations with 8 drivers or less.

  The following figure shows the recommended ID values for vendors. ID values 8100 to FFFE are assigned for IBM products only.

  Note: Programs should not make decisions based on the high nibble of the Adapter ID groupings.

ID	Definition
0000	Device Not Ready
0001 to 0FFF	Bus Master
5000 to 5FFF	Direct Memory Access Devices
6000 to 6FFF	Direct Program Control (Including Memory-Mapped I/O)
7000 to 7FFF	Storage*
8000 to 80FF	Video
FFFF	Device Not Attached

* Multiple-function adapters containing storage typically respond as storage.

Figure 16. Vendor ID Assignments

- Each enabled adapter must return a '-card selected feedback' signal (-CD SFDBK) to the system microprocessor when an access is made to the address space of the adapter, or when the adapter is selected by arbitration level. -CD SFDBK must not be generated when the '-card setup' (-CD SETUP) line is active.

- Each adapter design must be capable of degating all outputs to the system board (including -CD SFDBK, -CD DS 16/32, interrupts, and so on) if bit 0 of POS Register 2 (hex 0102) is set to 0.

- Each adapter design must implement an open-collector driver (or a tri-state driver gated negative-active) to drive the interrupt request line. The design must also implement a status register (readable at an I/O address bit position) that remains active when the interrupt is set, and stays active until reset by the service routine. The adapter must hold the level-sensitive interrupt active until it is reset as a direct result of servicing the interrupt. The service routine must not reset the interrupt controller until after the interrupt bus signal has been reset by the adapter.

- Following a reset, each adapter must set bit 0 (Card Enable) of its POS Register 2 to 0.

- If applicable, the adapter can reside at an alternate address (corresponding to one selected by switches on a Personal Computer-type adapter).

- All adapters must provide adapter description files (.adf) on a 3.5-inch diskette for system configuration.

- To provide maximum portability, devices designed for arbitration level 0 or 1 should have limited bandwidth or short bursts so diskette overruns can be prevented or recovered by retry operations. The diskette drive controller on arbitration level 2 may be held inactive by devices on levels 0 and 1, by a refresh operation, and by the previous controlling master. The diskette drive controller should not be held inactive for more than 12 microseconds to prevent overrun.

- Adapter designs should not extend the card-edge connector beyond the basic 16- or 32-bit connector unless the signals provided by the extension are used by the adapter.

Index

A

adapter bracket 15
adapter design 1
adapter design considerations 20
adapter diagnostics 22
adapter dimensions 1, 2
adapter holder 12
adapter retainer 13
assembly, typical adapter 11
assignments, recommended 23

B

bracket, adapter 15

C

channel load current 19
circuit protection, secondary 21
CMOS technology 1
compatibility, electromagnetic 21
component height 1
conductor protection 21
connector description 10
connector dimensions 3
connector, common detail 10

D

design considerations 20
design guidelines 22
design, adapter 1
diagnostics 22
dimensions, adapter 1, 2
dimensions, connector 3
dimensions, video adapter 4
dimensions, video connector 5

E

electromagnetic compatibility 21

F

formula, maximum statistical current 19
formula, total channel current 19

G

guidelines, design 22

H

height of components 1
holder, adapter 12

L

load current 19

M

maximum statistical current formula 19
micro channel adapter design 1

P

power specifications 19
protection of conductors 21

R

recommended assignments 23
regulation, voltage 20
retainer, adapter 13

S

safety 20
secondary circuit protection 21
specifications, power 19

T

technology, CMOS 1
thermal 21
total channel current formula 19
typical adapter assembly 11

V

video adapter dimensions 4
video connector dimensions 5
voltage regulation 20

Numerics

16-bit adapter 1
16-bit adapter with video 1
32-bit adapter 1
32-bit adapter with
  matched-memory extension 1

Microprocessors and Instruction Sets

80286 Microprocessor 1
  Real-Address Mode 1
  Protected Virtual Address Mode 2
80287 Math Coprocessor 2
  Programming Interface 3
  Hardware Interface 3
80386 Microprocessor 5
  Real Address Mode 5
  Protected Virtual Address Mode 6
  Virtual 8086 Mode 7
  80386 Paging Mechanism 8
80387 Math Coprocessor 10
  Programming Interface 11
  Hardware Interface 12
80286 Microprocessor Instruction Set 14
  Data Transfer 14
  Arithmetic 18
  Logic 22
  String Manipulation 24
  Control Transfer 26
  Processor Control 30
  Protection Control 32
80287 Math Coprocessor Instruction Set 35
  Data Transfer 35
  Comparison 37
  Constants 38
  Arithmetic 39
  Transcendental 40
  Processor Control 41
Introduction to the 80386 Instruction Set 43
  Code and Data Segment Descriptors 43
  Prefixes 44
  Instruction Format 45
  Encoding 46
  Address Mode 47
  Operand Length (w) Field 50
  Segment Register (sreg) Field 51
  General Register (reg) Field 51
  Operation Direction (d) Field 52

Sign-Extend (s) Field 52
Conditional Test (tttn) Field 52
Control, Debug, or Test Register (eee) Field 53
80386 Microprocessor Instruction Set 54
Data Transfer 54
Segment Control 57
Flag Control 58
Arithmetic 59
Logic 64
String Manipulation 68
Repeated String Manipulation 69
Bit Manipulation 71
Control Transfer 72
Conditional Jumps 73
Conditional Byte Set 78
Interrupt Instructions 80
Processor Control 81
Processor Extension 82
Prefix Bytes 82
Protection Control 83
Introduction to the 80387 Instruction Set 86
80387 Usage of the Scale-Index-Base Byte 86
Instruction and Data Pointers 87
New Instructions 89
80387 Math Coprocessor Instruction Set 90
Data Transfer 90
Comparison 91
Constants 93
Arithmetic 94
Transcendental 95
Processor Control 96

Index 99

Figures

1. 80287 Data Types 3
2. 80386 Addressing 7
3. Paging Mechanism 8
4. Data Type Classifications and Instructions 10
5. 80387 Data Types 12
6. 2-Bit Register Field 34
7. 3-Bit Register Field 34
8. 80287 Encoding Field Summary 35
9. 80386 Code and Data Segment Descriptor Format 43
10. Instruction Format 45
11. 80386 Instruction Set Encoding Field Summary 46
12. Effective Address (16-Bit and 32-Bit Address Modes) 47
13. Scale Factor (s-i-b Byte Present) 48
14. Index Registers (s-i-b Byte Present) 48
15. Base Registers (s-i-b Byte Present) 49
16. Effective Address (32-Bit Address Mode — s-i-b Byte Present) 50
17. Operand Length Field Encoding 51
18. Segment Register Field Encoding 51
19. General Register Field Encoding 51
20. Operand Direction Field Encoding 52
21. Sign-Extend Field Encoding 52
22. Conditional Test Field Encoding 53
23. Control, Debug, and Test Register Field Encoding 53
24. 80387 Encoding Field Summary 86
25. Instruction and Pointer Image (16-Bit Real Address Mode) 88
26. Instruction and Pointer Image (16-Bit Protected Mode) . . . 88
27. Instruction and Pointer Image (32-Bit Real Address Mode) 88
28. Instruction and Pointer Image (32-Bit Protected Mode) . . . 89

Notes:

80286 Microprocessor

The 80286 microprocessor subsystem has the following:

- 24-bit address
- 16-bit data interface
- Extensive instruction set, including string I/O
- Hardware fixed-point multiply and divide
- Two operational modes:
  - 8086-compatible Real Address
  - Protected Virtual Address.
- 16MB (MB equals 1,048,576 or 2^{20} bytes) of physical address space
- 1GB (GB equals 1,073,741,824 or 2^{30} bytes) of virtual address space.

Real-Address Mode

In the real-address mode, the address space of the system microprocessor is a contiguous array of up to 1MB. The system microprocessor generates 20-bit physical addresses to address memory.

The segment portion of the pointer is interpreted as the upper 16 bits of a 20-bit segment address; the lower 4 bits are always 0. Therefore, segment addresses begin on multiples of 16 bytes.

All segments in the real-address mode are 64KB (KB equals 1024 bytes) and can be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (for example, a word with its low-order byte at offset hex FFFF and its high-order byte at hex 0000). If, in the real-address mode, the information contained in the segment does not use the full 64KB, the unused end of the segment can be overlaid by another segment to reduce physical memory requirements.

Protected Virtual Address Mode

The protected virtual address mode (hereafter called protected mode) offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

The protected mode provides a virtual address space of 1GB for each task mapped into a 16MB physical address space. The virtual address space may be larger than the physical address space, because any use of an address that does not map to a physical memory location will cause a restartable exception.

Like the real-address mode, the protected mode uses 32-bit pointers, consisting of 16-bit selector and offset components. The selector specifies an index into a memory-resident table rather than the upper 16 bits of a real address. The 24-bit base address of the desired segment is obtained from a table in memory. The 16-bit offset is added to the segment base address to form the physical address. The system microprocessor automatically refers to the tables whenever a segment register is loaded with a selector. All instructions that load a segment register refer to the table without additional program support. Each entry in a table is 8-bytes.

80287 Math Coprocessor

The optional 80287 Math Coprocessor enables the system to perform high-speed arithmetic, logarithmic, and trigonometric operations. The coprocessor works in parallel with the microprocessor. The parallel operation decreases operating time by allowing the coprocessor to do mathematical calculations while the microprocessor continues to do other functions.

The coprocessor works with seven numeric data types, which are divided into the following three classes:

- Binary integers (three types)
- Decimal integers (one type)
- Real numbers (three types).

Programming Interface

The coprocessor offers extended data types, registers, and instructions to the microprocessor. The coprocessor has eight 80-bit registers, which provide the equivalent capacity of forty 16-bit registers. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access, improving speed, and increasing bus availability. The register space can be used as a stack or as a fixed register set. When used as a stack, only the top two stack elements are operated on.

The following figure shows representations of large and small numbers in each data type.

Data Type	**Bits**	**Significant Digits (Decimal)**	**Approximate Range (Decimal)**
Word Integer	16	4	$-32{,}768 \leq x \leq +32{,}767$
Short Integer	32	9	$-2 \times 10^{9} \leq x \leq +2 \times 10^{9}$
Long Integer	64	19	$-9 \times 10^{18} \leq x \leq +9 \times 10^{18}$
Packed Decimal	80	18	$-9..99 \leq x \leq +9..99$ (18 digits)
Short Real *	32	6 - 7	$8.43 \times 10^{-37} \leq x \leq 3.37 \times 10^{38}$
Long Real *	64	15 - 16	$4.19 \times 10^{-307} \leq x \leq 1.67 \times 10^{308}$
Temporary Real **	80	19	$3.4 \times 10^{-4932} \leq x \leq 1.2 \times 10^{4932}$

* The Short Real and Long Real Data Types Correspond to the Single- and Double-precision Data Types.

** The Temporary Real Data Type Corresponds to the Extended-precision Data Type.

Figure 1. 80287 Data Types

Hardware Interface

The coprocessor uses the same clock generator as the microprocessor and operates in the asynchronous mode. The coprocessor is wired so that it functions as an I/O device through I/O port addresses hex 00F8, 00FA, and 00FC. The microprocessor sends opcodes and operands through these I/O ports. It also receives and stores results through the same I/O ports. The coprocessor 'busy' signal informs the microprocessor that it is executing; the microprocessor Wait instruction forces the microprocessor to wait until the coprocessor is finished executing.

The coprocessor detects six different exception conditions that can occur during instruction execution:

- Invalid operation
- Denormal operand
- Zero-divide
- Overflow
- Underflow
- Precision.

If the appropriate exception mask bit within the coprocessor is not set, the coprocessor activates the 'error' signal. The 'error' signal generates a hardware interrupt (IRQ 13) causing the 'busy' signal to be held in the busy state. The 'busy' signal may be cleared by an 8-bit I/O Write command to address hex 00F0, with D7 through D0 equal to 0. This action also clears IRQ 13.

The power-on self-test code in the system ROM enables IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the non-maskable interrupt (NMI) vector. This maintains code compatibility across the IBM Personal Computer and Personal System/2 product lines. The NMI handler reads the coprocessor status to determine if the coprocessor generated the NMI. If it wasn't generated by the coprocessor, control is passed to the original NMI handler.

The coprocessor has two operating modes: real-address mode and protected mode. They are similar to the two modes of the microprocessor. The coprocessor is in the real-address mode if reset by a power-on reset, system reset, or I/O write operation to port hex 00F1. This mode is compatible with the 8087 Math Coprocessor used in IBM Personal Computers. The coprocessor is placed in the protected mode by executing the SETPM ESC instruction. It is placed back in the real-address mode by an I/O write operation to port hex 00F1, with D7 through D0 equal to 0.

Detailed information for the internal functions of the 80287 Math Coprocessor is in the books listed in the Bibliography. Also see "Compatibility" for more information.

80386 Microprocessor

The 80386 microprocessor subsystem has the following:

- 32-bit address
- 32-bit data interface
- Extensive instruction set, including string I/O
- Hardware fixed-point multiply and divide
- Three operational modes:
  - Real Address
  - Protected Virtual Address
  - Virtual 8086.
- 4GB of physical address space
- 8 general-purpose 32-bit registers
- 64TB (TB equals 1,099,511,627,776 or 2^{40} bytes) of total virtual-address space.

Real Address Mode

In the real-address mode, the address space of the system microprocessor is a contiguous array of up to 1MB. The system microprocessor generates 20-bit physical addresses to address memory.

The segment portion of the pointer is interpreted as the upper 16 bits of a 20-bit segment address; the lower 4 bits are always 0. Therefore, segment addresses begin on multiples of 16 bytes.

All segments in the real-address mode are 64KB and can be read, written, or executed. An exception or interrupt can occur if data operands or instructions attempt to wrap around the end of a segment (for example, a word with its low-order byte at offset hex FFFF and its high-order byte at hex 0000). If, in the real-address mode, the information contained in the segment does not use the full 64KB, the unused end of the segment can be overlaid by another segment to reduce physical memory requirements.

Protected Virtual Address Mode

The protected virtual-address mode offers extended physical and virtual memory address space, memory protection mechanisms, and new operations to support operating systems and virtual memory.

The protected mode provides up to 64TB of virtual address space for each task mapped into a 4GB physical address space.

From a programmer′s point of view, the main difference between the real-address mode and protected mode is the increased address space and the method of calculating the base address. The protected mode uses 32- or 48-bit pointers, consisting of 16-bit selector and 16- or 32-bit offset components. The selector specifies an index into one of two memory-resident tables, the global descriptor table (GDT) or the local descriptor table (LDT). These tables contain the 32-bit base address of a given segment. The 32-bit effective offset is added to the segment base address to form the physical address. The system microprocessor automatically refers to the tables whenever a segment register is loaded with a selector. All instructions that load a segment register refer to the memory-resident tables without additional program support. The memory-resident tables contain 8-byte values called descriptors.

The paging option provides an additional way of managing memory in the very large segments of the 80386. Paging operates in the protected mode only, beneath segmentation. The paging mechanism translates the protected linear address (which comes from the segmentation unit) into a physical address. When paging is not enabled, the physical address is the same as the linear address. The following figure shows the 80386 addressing mechanism.

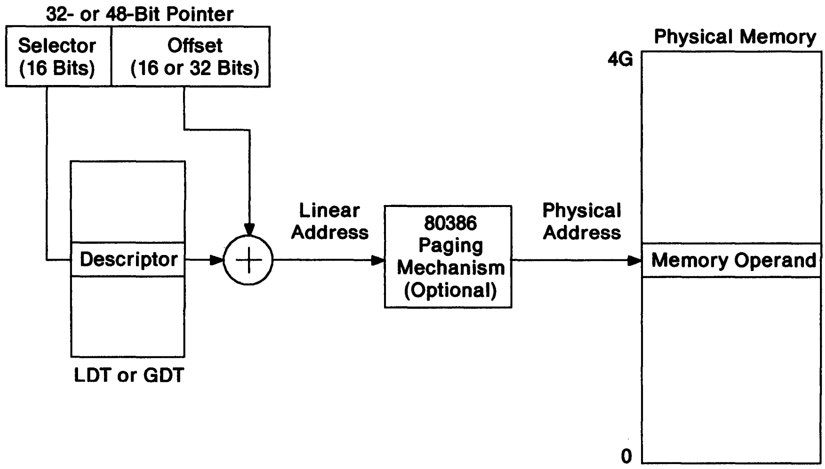


Figure 2. 80386 Addressing

Virtual 8086 Mode

The virtual-8086 mode ensures compatibility of programs written for 8086- and 8088-based systems by establishing a protected 8086 environment within the 80386 multitasking framework.

Since the address space of an 8086 is limited to 1MB, the logical addresses generated by the virtual-8086 mode lie within the first 1MB of the 80386 linear address space. To support multiple virtual-8086 tasks, paging can be used to give each virtual-8086 task a 1MB address space anywhere in the 80386 physical address space.

On a task-by-task basis, the value of the virtual-8086 flag (VM86 flag in the Flags register) determines whether the 80386 behaves as an 80386 or as an 8086. Some instructions, such as Clear Interrupt Flag, can disrupt all operations in a multitasking environment. The 80386 raises an exception when a virtual-8086 mode task attempts to execute an I/O instruction, interrupt-related instruction, or other sensitive instruction. Anytime an exception or interrupt occurs, the 80386 leaves the virtual 8086 mode, making the full resources of the 80386 available to an interrupt handler or exception handler. These handlers can determine if the source of the exception was a virtual-8086 mode task by inspecting the VM86 flag in the Flags image on the stack. If the source is a virtual-8086 mode task, the handler calls on a routine in the operating system to simulate an 8086 instruction and return to the virtual-8086 mode.[1]

80386 Paging Mechanism

The 80386 uses two levels of tables to translate the linear address from the segmentation unit into a physical address. There are three components to the paging mechanism:

- Page directory
- Page tables
- Page frame (the page itself).

The following figure shows how the two-level paging mechanism works.

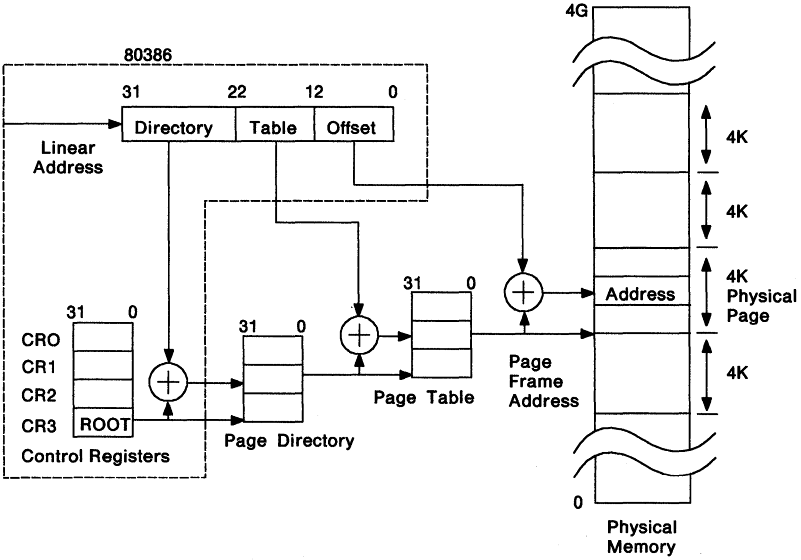


Figure 3. Paging Mechanism

CR2 is the Page-Fault Linear-Address register. It holds the 32-bit linear address that caused the last detected page fault.

[1] The routine in the operating system, called a *virtual machine monitor*, simulates a limited number of 8086 instructions.

CR3 is the Page Directory Physical Base Address register. It contains the physical starting address of the page directory.

The page directory is 4KB and allows up to 1024 page-directory entries. Each page-directory entry contains the address of the next level of tables, the page tables, and information about the page tables. The upper 10 bits of the linear address (A22 through A31) are used as an index to select the correct page-directory entry.

Each page table is 4KB and holds up to 1024 page-table entries. Page-table entries contain the starting address of the page frame and statistical information about the page. Address bits A12 through A21 are used as an index to select one of the 1024 page-table entries. The upper 20 bits of the page-frame address (from the page-table entry) are linked with the lower 12 bits of the linear address to form the physical address. The page-frame address bits become the most-significant bits; the linear-address bits become the least-significant bits.

80387 Math Coprocessor

The optional 80387 Math Coprocessor enables the system to perform high-speed arithmetic, logarithmic, and trigonometric operations. The 80387 effectively extends the 80386 register and instruction set for existing data types and also adds several new data types. The following figure shows the four data type classifications and the instructions associated with each.

Classification	Size	Instructions
Integer	16, 32, 64 Bits	Load, Store, Compare, Add, Subtract, Multiply, Divide
Packed BCD*	80 Bits	Load, Store
Real	32, 64 Bits	Load, Store, Compare, Add, Subtract, Multiply, Divide
Temporary Real	80 Bits	Add, Subtract, Multiply, Divide, Square Root, Scale, Remainder, Integer Part, Change, Sign, Absolute Value, Extract Exponent and Significand, Compare, Examine, Test, Exchange Tangent, Arctangent, $2^{X}-1$, Y*Log$_2$ (X+1), Y*Log$_2$ (X), Load Constant (0.0, π, etc.), Sine, Cosine, Unordered Compare

* BCD = Binary-coded decimal

Figure 4. Data Type Classifications and Instructions

The 80386/80387 configuration fully conforms to the ANSI[2] and IEEE[3] floating-point standard and are upward, object-code compatible from 80286/80287- and 8086/8087-based systems.

[2] American National Standards Institute

[3] Institute of Electrical and Electronics Engineers

Programming Interface

The 80387 is not sensitive to the processing mode of the 80386. The 80387 functions the same whether the 80386 is executing in real-address mode, protected mode, or virtual-8086 mode. All memory access is handled by the 80386; the 80387 merely operates on instructions and values passed to it by the 80386.

All communication between the 80386 and 80387 is transparent to application programs. The 80386 automatically controls the 80387 whenever a numeric instruction is executed. All physical and virtual memory is available for storage of instructions and operands of programs that use the 80387. All memory address modes, including use of displacement, base register, index register, and scaling are available for addressing numeric operands.

The coprocessor has eight 80-bit registers. The total capacity of these eight registers is equivalent to twenty 32-bit registers. This register space allows constants and temporary results to be held in registers during calculations, thus reducing memory access, improving speed, and increasing bus availability. The register set can be used as a stack or as a fixed register set. When it is used as a stack, only the top two stack elements are operated on.

The following figure shows the seven data types supported by the 80387 Math Coprocessor.

Data Type	**Range**	**Precision**
Word Integer	10^4	16 Bits
Short Integer	10^9	32 Bits
Long Integer	10^{19}	64 Bits
Packed BCD	10^{18}	18 Digits (2 digits per byte)
Single Precision (Short Real)	$10^{\pm 38}$	24 Bits
Double Precision (Long Real)	$10^{\pm 308}$	53 Bits
Extended Precision (Temporary Real)	$10^{\pm 4932}$	64 Bits

Figure 5. 80387 Data Types

Hardware Interface

The 80387 Math Coprocessor uses the same clock generator as the 80386 system microprocessor. The coprocessor is wired so that it functions as an I/O device through I/O port addresses hex 00F8, 00FA, and 00FC. The system microprocessor sends opcodes and operands through these I/O ports. The coprocessor 'busy' signal informs the system microprocessor that it is executing; the system microprocessor Wait instruction forces the system microprocessor to wait until the coprocessor is finished executing.

The coprocessor detects six different exception conditions that can occur during instruction execution:

- Invalid operation
- Denormal operand
- Zero-divide
- Overflow
- Underflow
- Precision.

If the appropriate exception mask bit within the coprocessor is not set, the coprocessor activates the 'error' signal. The 'error' signal generates a hardware interrupt (IRQ 13) causing the ‘busy’ signal to be held in the busy state. The ‘busy’ signal may be cleared by an 8-bit I/O Write command to address hex 00F0, with D7 through D0 equal to 0. This action also clears IRQ 13.

The power-on self-test code in the system ROM enables IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the ‘busy’ signal latch and then transfers control to the address pointed to by the (NMI) vector. This maintains code compatibility across the IBM Personal Computer and Personal System/2 product lines. The NMI handler reads the status of the coprocessor to determine if the coprocessor generated the NMI. If it wasn't generated by the coprocessor, control is passed to the original NMI handler.

Detailed information about the internal functions of the 80387 Math Coprocessor is in the books listed in the Bibliography. Also see “Compatibility” for more information.

80286 Microprocessor Instruction Set

Data Transfer

MOV = Move

Register to Register/Memory

1 0 0 0 1 0 0 w	mod reg r/m

Register/Memory to Register

1 0 0 0 1 0 1 w	mod reg r/m

Immediate to Register/Memory

1 1 0 0 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1

Immediate to Register

1 0 1 1 wreg	data	data if w = 1

Memory to Accumulator

1 0 1 0 0 0 0 w	addr-low	addr-high

Accumulator to Memory

1 0 1 0 0 0 1 w	addr-low	addr-high

Register/Memory to Segment Register

1 0 0 0 1 1 1 0	mod 0 reg r/m

Segment Register to Register/Memory

1 0 0 0 1 1 0 0	mod 0 reg r/m

PUSH = Push

Memory

1 1 1 1 1 1 1 1	mod 1 1 0 r/w

Register

0 1 0 1 0 reg

Segment Register

0 0 0 reg 1 1 0

Immediate

0 1 1 0 1 0 s 0	data	data if s = 0

PUSHA = Push All

0 1 1 0 0 0 0 0

POP = Pop

Register/Memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m

Register

0 1 0 1 1 reg

Segment Register

0 0 0 reg 1 1 1	reg ≠ 0 1

POPA = Pop All

0 1 1 0 0 0 0 1

XCHG = Exchange

Register/Memory with Register

1 0 0 0 0 1 1 w	mod reg r/m

Register with Accumulator

1 0 0 1 0 reg

IN = Input From

Fixed Port

1 1 1 0 0 1 0 w	port

Variable Port

1 1 1 0 1 1 0 w

OUT = Output To

Fixed Port

1 1 1 0 0 1 1 w	port

Variable Port

1 1 1 0 1 1 1 w

XLAT = Translate Byte to AL

1 1 0 1 0 1 1 1

LEA = Load EA to Register

10001101	mod reg r/m

LDS = Load Pointer to DS

11000101	mod reg r/m mod ≠ 11

LES = Load Pointer to ES

11000100	mod reg r/m mod ≠ 11

LAHF = Load AH with Flags

10011111

SAHF = Store AH with Flags

10011110

PUSHF = Push Flags

10011100

POPF = Pop Flags

10011101

Arithmetic

ADD = Add

Register/Memory with Register to Either

0 0 0 0 0 0 dw	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 sw	mod 0 0 0 r/m	data	data if sw = 0 1

Immediate to Accumulator

0 0 0 0 0 1 0 w	data	data if w = 1

ADC = Add with Carry

Register/Memory with Register to Either

0 0 0 1 0 0 dw	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 sw	mod 0 1 0 r/m	data	data if sw = 0 1

Immediate to Accumulator

0 0 0 1 0 1 0 w	data	data if w = 1

INC = Increment

Register/Memory

1 1 1 1 1 1 1 w	mod 0 0 0 r/m

Register

0 1 0 0 0 reg

SUB = Subtract

Register/Memory with Register to Either

0 0 1 0 1 0 dw	mod reg r/m

Immediate from Register/Memory

1 0 0 0 0 0 sw	mod 1 0 1 r/m	data	data if sw = 0 1

Immediate from Accumulator

0 0 1 0 1 1 0 w	data	data if w = 1

SBB = Subtract with Borrow

Register/Memory with Register to Either

0 0 0 1 1 0 dw	mod reg r/m

Immediate from Register/Memory

1 0 0 0 0 0 sw	mod 0 1 1 r/m	data	data if sw = 0 1

Immediate from Accumulator

0 0 0 1 1 1 0 w	data	data if w = 1

DEC = Decrement

Register/Memory

1 1 1 1 1 1 1 w	mod 0 0 1 r/m

Register

0 1 0 0 1 reg

CMP = Compare

Register/Memory with Register

0 0 1 1 1 0 1 w	mod reg r/m

Register with Register/Memory

0 0 1 1 1 0 0 w	mod reg r/m

Immediate with Register/Memory

1 0 0 0 0 0 s w	mod 1 1 1 r/m	data	data if sw = 0 1

Immediate with Accumulator

0 0 1 1 1 1 0 w	data	data if w = 1

NEG = Change Sign

1 1 1 1 0 1 1 w	mod 0 1 1 r/m

AAA = ASCII Adjust for Add

0 0 1 1 0 1 1 1

DAA = Decimal Adjust for Add

0 0 1 0 0 1 1 1

AAS = ASCII Adjust for Subtract

0 0 1 1 1 1 1 1

DAS = Decimal Adjust for Subtract

0 0 1 0 1 1 1 1

MUL = Multiply (Unsigned)

1 1 1 1 0 1 1 w	mod 1 0 0 r/m

IMUL = Integer Multiply (Signed)

1 1 1 1 0 1 1 w	mod 1 0 1 r/m

IIMUL = Integer Immediate Multiply (Signed)

0 1 1 0 1 0 s 1	mod reg r/m	data	data if s = 0

DIV = Divide (Unsigned)

1 1 1 1 0 1 1 w	mod 1 1 0 r/m

IDIV = Integer Divide (Signed)

1 1 1 1 0 1 1 w	mod 1 1 1 r/m

AAM = ASCII Adjust for Multiply

1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0

AAD = ASCII Adjust for Divide

1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0

CBW = Convert Byte to Word

1 0 0 1 1 0 0 0

CWD = Convert Word to Doubleword

1 0 0 1 1 0 0 1

Logic

Shift/Rotate Instructions

Register/Memory by 1

1 1 0 1 0 0 0 w	mod T T T r/m

Register/Memory by CL

1 1 0 1 0 0 1 w	mod T T T r/m

Register/Memory by Count

1 1 0 0 0 0 0 w	mod T T T r/m	count

T T T	Instruction
0 0 0	ROL
0 0 1	ROR
0 1 0	RCL
0 1 1	RCR
1 0 0	SHL/SAL
1 0 1	SHR
1 1 1	SAR

AND = And

Register/Memory and Register to Either

0 0 1 0 0 0 dw	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 0 w	mod 100 r/m	data	data if w = 1

Immediate to Accumulator

0 0 1 0 0 1 0 w	data	data if w = 1

TEST = AND Function to Flags; No Result

Register/Memory and Register

1 0 0 0 0 1 0 w	mod reg r/m

Immediate Data and Register/Memory

1 1 1 1 0 1 1 w	mod 0 0 0 r/m	data	data if w = 1

Immediate Data and Accumulator

1 0 1 0 1 0 0 w	data	data if w = 1

Or = Or

Register/Memory and Register to Either

0 0 0 0 1 0 d w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 0 w	mod 0 0 1 r/m	data	data if w = 1

Immediate to Accumulator

0 0 0 0 1 1 0 w	data	data if w = 1

XOR = Exclusive OR

Register/Memory and Register to Either

0 0 1 1 0 0 d w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 0 w	mod 1 1 0 r/m	data	data if w = 1

Immediate to Accumulator

0 0 1 1 0 1 0 w	data	data if w = 1

NOT = Invert Register/Memory

1 1 1 1 0 1 1 w	mod 0 1 0 r/m

String Manipulation

MOVS = Move Byte Word

1010010w

CMPS B/W = Compare Byte/Word

1010011w

SCAS = Scan Byte/Word

1010111w

LODS = Load Byte/Word to AL/AX

1010110w

STOS = Store Byte/Word from AL/AX

1010101w

INS = Input Byte/Word from DX Port

0110110w

OUTS = Output Byte/Word to DX Port

0110111w

REP/REPNE, REPZ/REPNZ = Repeat String

Repeat Move String

1 1 1 1 0 0 1 1	1 0 1 0 0 1 0 w

Repeat Compare String (z/Not z)

1 1 1 1 0 0 1 z	1 0 1 0 0 1 1 w

Repeat Scan String (z/Not z)

1 1 1 1 0 0 1 z	1 0 1 0 1 1 1 w

Repeat Load String

1 1 1 1 0 0 1 1	1 0 1 0 1 1 0 w

Repeat Store String

1 1 1 1 0 0 1 1	1 0 1 0 1 0 1 w

Repeat Input String

1 1 1 1 0 0 1 1	0 1 1 0 1 1 0 w

Repeat Output String

1 1 1 1 0 0 1 1	0 1 1 0 1 1 1 w

Control Transfer

CALL = Call

Direct within Segment

1 1 1 0 1 0 0 0	disp-low	disp-high

Register/Memory Indirect within Segment

1 1 1 1 1 1 1 1	mod 0 1 0 r/m

Direct Intersegment

1 0 0 1 1 0 1 0	Segment Offset	Segment Selector

Indirect Intersegment

1 1 1 1 1 1 1 1	mod 0 1 1 r/m (mod ≠ 11)

JMP = Unconditional Jump

Short/Long

1 1 1 0 1 0 1 1	disp-low

Direct within Segment

1 1 1 0 1 0 0 1	disp-low	disp-high

Register/Memory Indirect within Segment

1 1 1 1 1 1 1 1	mod 1 0 0 r/m

Direct Intersegment

1 1 1 0 1 0 1 0	Segment Offset	Segment Selector

Indirect Intersegment

1 1 1 1 1 1 1 1	mod 1 0 1 r/m (mod ≠ 11)

RET = Return from Call

Within Segment

1 1 0 0 0 0 1 1

Within Segment Adding Immediate to SP

1 1 0 0 0 0 1 0	data-low	data-high

Intersegment

1 1 0 0 1 0 1 1

Intersegment Adding Immediate to SP

1 1 0 0 1 0 1 0	data-low	data-high

JE/JZ = Jump on Equal/Zero

0 1 1 1 0 1 0 0	disp

JL/JNGE = Jump on Less/Not Greater, or Equal

0 1 1 1 1 1 0 0	disp

JLE/JNG = Jump on Less, or Equal/Not Greater

0 1 1 1 1 1 1 0	disp

JB/JNAE = Jump on Below/Not Above, or Equal

0 1 1 1 0 0 1 0	disp

JBE/JNA = Jump on Below, or Equal/Not Above

0 1 1 1 0 1 1 0	disp

JP/JPE = Jump on Parity/Parity Even

0 1 1 1 1 0 1 0	disp

JO = Jump on Overflow

0 1 1 1 0 0 0 0	disp

JS = Jump on Sign

0 1 1 1 1 0 0 0	disp

JNE/JNZ = Jump on Not Equal/Not Zero

0 1 1 1 0 1 0 1	disp

JNL/JGE = Jump on Not Less/Greater, or Equal

0 1 1 1 1 1 0 1	disp

JNLE/JG = Jump on Not Less, or Equal/Greater

0 1 1 1 1 1 1 1	disp

JNB/JAE = Jump on Not Below/Above, or Equal

0 1 1 1 0 0 1 1	disp

JNBE/JA = Jump on Not Below, or Equal/Above

0 1 1 1 0 1 1 1	disp

JNP/JPO = Jump on Not Parity/Parity Odd

0 1 1 1 1 0 1 1	disp

JNO = Jump on Not Overflow

0 1 1 1 0 0 0 1	disp

JNS = Jump on Not Sign

0 1 1 1 1 0 0 1	disp

LOOP = Loop CX Times

1 1 1 0 0 0 1 0	disp

LOOPZ/LOOPE = Loop while Zero/Equal

1 1 1 0 0 0 0 1	disp

LOOPNZ/LOOPNE = Loop while Not Zero/Not Equal

1 1 1 0 0 0 0 0	disp

JCXZ = Jump on CX Zero

1 1 1 0 0 0 1 1	disp

ENTER = Enter Procedure

1 1 0 0 1 0 0 0	data-low	data-high	L

LEAVE = Leave Procedure

1 1 0 0 1 0 0 1

INT = Interrupt

Type Specified

11001101	type

Type 3

11001100

INTO = Interrupt on Overflow

11001110

IRET = Interrupt Return

11001111

BOUND = Detect Value Out of Range

01100010	mod reg r/m

Processor Control

CLC = Clear Carry

11111000

CMC = Complement Carry

11110101

STC = Set Carry

11111001

CLD = Clear Direction

11111100

STD = Set Direction

11111101

CLI = Clear Interrupt

11111010

STI = Set Interrupt Enable Flag

11111011

HLT = Halt

11110100

WAIT = Wait

10011011

LOCK = Bus Lock Prefix

11110000

CTS = Clear Task Switched Flag

00001111	00000110

ESC = Processor Extension Escape

11011TTT	mod LLL r/m

Protection Control

LGDT = Load Global Descriptor Table Register

00001111	00000001	mod 0 1 0 r/m

SGDT = Store Global Descriptor Table Register

00001111	00000001	mod 0 0 0 r/m

LIDT = Load Interrupt Descriptor Table Register

00001111	00000001	mod 0 1 1 r/m

SIDT = Store Interrupt Descriptor Table Register

00001111	00000001	mod 0 0 1 r/m

LLDT = Load Local Descriptor Table Register from Register/Memory

00001111	00000000	mod 0 1 0 r/m

SLDT = Store Local Descriptor Table Register from Register/Memory

00001111	00000000	mod 0 0 0 r/m

LTR = Load Task Register from Register/Memory

00001111	00000000	mod 0 1 1 r/m

STR = Store Task Register to Register/Memory

00001111	00000000	mod 0 0 1 r/m

LMSW = Load Machine Status Word from Register/Memory

00001111	00000001	mod 1 1 0 r/m

SMSW = Store Machine Status Word

00001111	00000001	mod 100 r/m

LAR = Load Access Rights from Register/Memory

00001111	00000010	mod reg r/m

LSL = Load Segment Limit from Register/Memory

00001111	00000011	mod reg r/m

ARPL = Adjust Requested Privilege Level from Register/Memory

01100011	mod reg r/m

VERR = Verify Read Access; Register/Memory

00001111	00000000	mod 100 r/m

VERW = Verify Write Access

00001111	00000000	mod 101 r/m

The effective address (EA) of the memory operand is computed according to the mod and r/m fields:

If mod = 11, then r/m is treated as a reg field.
If mod = 00, then disp = 0, disp-low and disp-high are absent.
If mod = 01, then disp = disp-low sign-extended to 16 bits, disp-high is absent.
If mod = 10, then disp = disp-high:disp-low.

If r/m = 000, then EA = (BX) + (SI) + DISP
If r/m = 001, then EA = (BX) + (DI) + DISP
If r/m = 010, then EA = (BP) + (SI) + DISP
If r/m = 011, then EA = (BP) + (DI) + DISP
If r/m = 100, then EA = (SI) + DISP
If r/m = 101, then EA = (DI) + DISP
If r/m = 110, then EA = (BP) + DISP
If r/m = 111, then EA = (BX) + DISP

The disp field follows the second byte of the instruction (before data if required).

Note: An exception to the above statements occurs when mod=00 and r/m=110, in which case EA = disp-high; disp-low.

Segment Override Prefix

```
0 0 1 reg 1 1 0
```

The 2-bit and 3-bit reg fields are defined in the following figures.

Reg	Segment Register	Reg	Segment Register
00	ES	10	SS
01	CS	11	DS

Figure 6. 2-Bit Register Field

16-Bit (w = 1)	8-Bit (w = 0)
000 AX	000 AL
001 CX	001 CL
010 DX	010 DL
011 BX	011 BL
100 SP	100 AH
101 BP	101 CH
110 SI	110 DH
111 DI	111 BH

Figure 7. 3-Bit Register Field

The physical addresses of all operands addressed by the BP register are computed using the SS Segment register. The physical addresses of the destination operands of the string primitive operations (those addressed by the DI register) are computed using the ES segment, which may not be overridden.

80287 Math Coprocessor Instruction Set

The following is an instruction-set summary for the 80287 Math Coprocessor.

The following figure shows abbreviations used in the summary.

Field	Description	Bit Information
escape	80286 Extension Escape	Bit Pattern = 11011
MF	Memory Format	00 = 32-Bit Real 01 = 32-Bit Integer 10 = 64-Bit Real 11 = 16-Bit Integer
ST(0)	Current Stack Top	
ST(i)	ith Register Below the Stack Top	
d	Destination	0 = Destination is ST(0) 1 = Destination is ST(i)
P	Pop	0 = No pop 1 = Pop ST(0)
R	Reverse*	0 = Destination (op) source 1 = Source (op) destination

* When d = 1, reverse the sense of R.

Figure 8. 80287 Encoding Field Summary

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 0 0 0 r/m

Long Integer Memory to ST(0)

escape 1 1 1	mod 1 0 1 r/m

Temporary Real Memory to ST(0)

escape 0 1 1	mod 1 0 1 r/m

BCD Memory to ST(0)

escape 1 1 1	mod 1 0 0 r/m

ST(i) to ST(0)

escape 0 0 1	1 1 0 0 0 ST(i)

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 0 1 0 r/m

ST(0) to ST(i)

escape 1 0 1	1 1 0 1 0 ST(i)

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 0 1 1 r/m

ST(0) to Long Integer Memory

escape 1 1 1	mod 1 1 1 r/m

ST(0) to Temporary Real Memory

escape 0 1 1	mod 1 1 1 r/m

ST(0) to BCD Memory

escape 1 1 1	mod 1 1 0 r/m

ST(0) to ST(i)

escape 1 0 1	1 1 0 1 1 ST(i)

FXCH = Exchange ST(i) and ST(0)

escape 0 0 1	1 1 0 0 1 ST(i)

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 0 1 0 r/m

ST(i) to ST(0)

escape 0 0 0	1 1 0 1 0 ST(i)

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 0 1 1 r/m

ST(i) to ST(0)

escape 0 0 0	1 1 0 1 1 ST(i)

FCOMPP = Compare ST(1) to ST(0) and Pop Twice

escape 1 1 0	1 1 0 1 1 0 0 1

FTST = Test ST(0)

escape 0 0 1	1 1 1 0 0 1 0 0

FXAM = Examine ST(0)

escape 0 0 1	1 1 1 0 0 1 0 1

Constants

FLDZ = Load + 0.0 into ST(0)

escape 0 0 1	1 1 1 0 1 1 1 0

FLD1 = Load + 1.0 into ST(0)

escape 0 0 1	1 1 1 0 1 0 0 0

FLDPI = Load π into ST(0)

escape 0 0 1	1 1 1 0 1 0 1 1

FLDL2T = Load $\log_2$ 10 into ST(0)

escape 0 0 1	1 1 1 0 1 0 0 1

FLDL2E = Load $\log_2$ e into ST(0)

escape 0 0 1	1 1 1 0 1 0 1 0

FLDLG2 = Load $\log_{10}$ 2 into ST(0)

escape 0 0 1	1 1 1 0 1 1 0 0

FLDLN2 = Load $\log_e$ 2 into ST(0)

escape 0 0 1	1 1 1 0 1 1 0 1

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 0 0 0 r/m

ST(i) and ST(0)

escape dP0	1 1 0 0 0 ST(i)

FSUB = Subtraction

Integer/Real Memory with ST(0)

escape MF 0	mod 1 0 R r/m

ST(i) and ST(0)

escape dP 0	1110R r/m

FMUL = Multiplication

Integer/Real Memory with ST(0)

escape MF 0	mod 0 0 1 r/m

ST(i) and ST(0)

escape dP 0	1 1 0 0 1 r/m

FDIV = Division

Integer/Real Memory with ST(0)

escape MF 0	mod 1 1 R r/m

ST(i) and ST(0)

escape dP 0	1 1 1 1 R r/m

FSQRT = Square Root of ST(0)

escape 0 0 1	1 1 1 1 1 0 1 0

FSCALE = Scale ST(0) by ST(1)

escape 0 0 1	1 1 1 1 1 1 0 1

FPREM = Partial Remainder of ST(0) ÷ ST(1)

escape 0 0 1	1 1 1 1 1 0 0 0

FRNDINT = Round ST(0) to Integer

escape 0 0 1	1 1 1 1 1 1 0 0

FXTRACT = Extract Components of ST(0)

escape 0 0 1	1 1 1 1 0 1 0 0

FABS = Absolute Value of ST(0)

escape 0 0 1	1 1 1 0 0 0 0 1

FCHS = Change Sign of ST(0)

escape 0 0 1	1 1 1 0 0 0 0 0

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 0 0 1	1 1 1 1 0 0 1 0

FPATAN = Partial Arctangent of ST(1) ÷ ST(0)

escape 0 0 1	1 1 1 1 0 0 1 1

F2XM1 = $2^{ST(0)}$ -1

escape 0 0 1	1 1 1 1 0 0 0 0

FYL2X = ST(1) x Log_2 [ST(0)]

escape 0 0 1	1 1 1 1 0 0 0 1

FYL2XP1 = ST(1) x Log_2 [ST(0) + 1]

escape 0 0 1	1 1 1 1 1 0 0 1

Processor Control

FINIT = Initialize NPX

escape 0 1 1	1 1 1 0 0 0 1 1

FSETPM = Enter Protected Mode

escape 0 1 1	1 1 1 0 0 1 0 0

FSTSW AX = Store Control Word

escape 1 1 1	1 1 1 0 0 0 0 0

FLDCW = Load Control Word

escape 0 0 1	mod 1 0 1 r/m

FSTCW = Store Control Word

escape 0 0 1	mod 1 1 1 r/m

FSTSW = Store Status Word

escape 1 0 1	mod 1 1 1 r/m

FCLEX = Clear Exceptions

escape 0 1 1	1 1 1 0 0 0 1 0

FSTENV = Store Environment

escape 0 0 1	mod 1 1 0 r/m

FLDENV = Load Environment

escape 0 0 1	mod 1 0 0 r/m

FSAVE = Save State

escape 1 0 1	mod 1 1 0 r/m

FRSTOR = Restore State

escape 1 0 1	mod 1 0 0 r/m

FINCSTP = Increment Stack Pointer

escape 0 0 1	1 1 1 1 0 1 1 1

FDECSTP = Decrement Stack Pointer

escape 0 0 1	1 1 1 1 0 1 1 0

FFREE = Free ST(i)

escape 1 0 1	1 1 0 0 0 ST(i)

FNOP = No Operation

escape 0 0 1	1 1 0 1 0 0 0 0

Introduction to the 80386 Instruction Set

The 80386 instruction set is an extended version of the 8086 and 80286 instruction sets. The instruction sets have been extended in two ways:

- The instructions have extensions that allow operations on 32-bit operands, registers, and memory.
- A 32-bit addressing mode allows flexible selection of registers for base and index as well as index scaling capabilities (x2, x4, x8) for computing a 32-bit effective address. The 32-bit effective address yields a 4GB address range.

  Note: The effective address size must be less than 64KB in the real-address or virtual-address modes to avoid an exception.

Code and Data Segment Descriptors

Although the 80386 supports all 80286 Code and Data segment descriptors, there are some differences in the format. The 80286 segment descriptors contain a 24-bit base address and a 16-bit limit field, while the 80386 segment descriptors have a 32-bit base address, a 20-bit limit field, a default bit, and a granularity bit.

<table>
<tr><th>31 24</th><th colspan="5">23 16</th><th>15 08</th><th>07 00</th><th>Offset</th></tr>
<tr><td colspan="6">Segment Base (SB) Bits 15-0</td><td colspan="2">Segment Limit (SL) Bits 15-0</td><td>0</td></tr>
<tr><td>SB Bits 31-24</td><td>G</td><td>D</td><td>0</td><td>0</td><td>SL 19-16</td><td>Access Rights Byte</td><td>SB Bits 23-16</td><td>4</td></tr>
</table>

Figure 9. 80386 Code and Data Segment Descriptor Format

Note: Bits 31 through 16 shown at offset 4 are set to 0 for all 80286 segment descriptors.

The default (D) bit of the code segment register is used to determine whether the instruction is carried out as a 16-bit or 32-bit instruction. Code segment descriptors are not used in either the real-address mode or the virtual-8086 mode. When the system microprocessor is operating in either of these modes, a D-bit value of 0 is assumed and operations default to a 16-bit length compatible with 8086 and 80286 programs.

The granularity (G) bit is used to determine the granularity of the segment length (1 = page granular, 0 = byte granular). If the value of the 20 segment-limit bits is defined as *N*, a G-bit value of 1 defines the segment size as follows:

Segment size = (*N* + 1) x 4KB

4KB represents the size of a page.

Prefixes

Two prefixes have been added to the instruction set. The Operand Size prefix overrides the default selection of the operand size; the Effective Address Size prefix overrides the effective address size. The presence of either prefix toggles the default setting to its opposite condition. For example:

- If the operand size defaults to 32-bit data operations, the presence of the Operand Size prefix sets it for 16-bit data operations.
- If the effective address size is 16-bits, the presence of the Effective Address Size prefix toggles the instruction to use 32-bit effective address computations.

The prefixes are available in all 80386 modes, including the real-address mode and the virtual-8086 mode. Since the default of these modes is always 16 bits, the prefixes are used to specify 32-bit operations. If needed, either or both of the prefixes may precede any opcode bytes and affect only the instruction they precede.

Instruction Format

The instructions are presented in this format:

Opcode	Mode Specifier	Address Displacement	Immediate Data

Term	Description
Opcode	The opcode may be one or two bytes in length. Within each byte, smaller encoding fields may be defined.
Mode Specifier	Consists of the "mod r/m" byte and the "scale-index-base" (s-i-b) byte. The mod r/m byte specifies the address mode to be used. Format: mod T T T r/m The "s-i-b" byte is optional and can be used only in 32-bit address modes. It follows the mod r/m byte to fully specify the manner in which the effective address is computed. Format: ss index base
Address Displacement	Follows the "mod r/m" byte or "s-i-b" byte. It may be 8, 16, or 32 bits.
Immediate Data	If specified, follows any displacement bytes and becomes the last field of the instruction. It may be 8, 16, or 32 bits. The term "8-bit data" indicates a fixed data length of 8 bits. The term "8-, 16-, or 32-bit data" indicates a variable data length. The length is determined by the w field and the current operand size. If w = 0, the data is always 8 bits. If w = 1, the size is determined by the operand size of the instruction.

Figure 10. Instruction Format

The instructions use a variety of fields to indicate register selection, the addressing mode, and so on. The following figure is a summary of the fields.

Field Name	Description	Bit Information
w	Specifies if data is byte or full size. (Full size is either 16 or 32 bits.)	1
d	Specifies the direction of data operation.	1
s	Specifies if an immediate data field must be sign-extended.	1
reg	General address specifier.	3
mod r/m	Address mode specifier (effective address can be a general register).	2 for mod; 3 for r/m
ss	Scale factor for scaled index address mode.	2
index	General register to be used as an index register.	3
base	General register to be used as base register.	3
sreg2	Segment register specifier for CS, SS, DS, and ES.	2
sreg3	Segment register specifier for CS, SS, DS, ES, FS, and GS.	3
tttn	For conditional instructions; specifies a condition asserted or a condition negated.	4

Figure 11. 80386 Instruction Set Encoding Field Summary

Encoding

This section defines the encoding of the fields used in the instruction sets.

Address Mode

The first addressing byte is the "mod r/m" byte. The effective address (EA) of the memory operand is computed according to the mod and r/m fields. The mod r/m byte can be interpreted as either a 16-bit or 32-bit addressing mode specifier. Interpretation of the byte depends on the address components used to calculate the EA. The following figure defines the encoding of 16-bit and 32-bit addressing modes with the mod r/m byte.

mod r/m	16-Bit Mode	32-Bit Mode (No s-i-b byte)
00 000	DS:[BX + SI]	DS:[EAX]
00 001	DS:[BX + DI]	DS:[ECX]
00 010	SS:[BP + SI]	DS:[EDX]
00 011	SS:[BP + DI]	DS:[EBX]
00 100	DS:[SI]	s-i-b present (see Figure 16 on page 50)
00 101	DS:[DI]	DS:d32
00 110	d16	DS:[ESI]
00 111	DS:[BX]	DS:[EDI]
01 000	DS:[BX + SI + d8]	DS:[EAX + d8]
01 001	DS:[BX + DI + d8]	DS:[ECX + d8]
01 010	SS:[BP + SI + d8]	DS:[EDX + d8]
01 011	SS:[BP + DI + d8]	DS:[EBX + d8]
01 100	DS:[SI + d8]	s-i-b present (see Figure 16 on page 50)
01 101	DS:[DI + d8]	SS:[EBP + d8]
01 110	SS:[BP + d8]	DS:[ESI + d8]
01 111	DS:[BX + d8]	DS:[EDI + d8]
10 000	DS:[BX + SI + d16]	DS:[EAX + d32]
10 001	DS:[BX + DI + d16]	DS:[ECX + d32]
10 010	SS:[BP + SI + d16]	SS:[EDX + d32]
10 011	SS:[BP + DI + d16]	DS:[EBX + d32]
10 100	DS:[SI + d16]	s-i-b present (see Figure 16 on page 50)
10 101	DS:[DI + d16]	SS:[EBP + d32]
10 110	SS:[BP + d16]	DS:[ESI + d32]
10 111	DS:[BX + d16]	DS:[EDI + d32]

Figure 12. Effective Address (16-Bit and 32-Bit Address Modes)

The displacement follows the second byte of the instruction (before data, if required).

The scale-index-base (s-i-b) byte can be specified as a second byte of addressing information. The s-i-b byte is specified when using a

32-bit addressing mode and the mod r/m byte has the following values:

- r/m = 100
- mod = 00, 01, or 10.

When the s-i-b byte is present, the 32-bit effective address is a function of the mod, ss, index, and base fields. The following figures show the scale factor, Index register selected, and base register selected when the s-i-b byte is present.

ss	Scale Factor
00	1
01	2
10	4
11	8

Figure 13. Scale Factor (s-i-b Byte Present)

Index	Index Register	
000	EAX	
001	ECX	
010	EDX	
011	EBX	
100	No Index Register	The ss field must equal 00 when the index field is 100; if not, the effective address is undefined.
101	EBP	
110	ESI	
111	EDI	

Figure 14. Index Registers (s-i-b Byte Present)

base	Base Register	
000	EAX	
001	ECX	
010	EDX	
011	EBX	
100	ESP	
101	EBP	If mod = 00, then EBP is not used to form the EA; immediate 32-bit address displacement follows the mode specifier byte.
110	ESI	
111	EDI	

Figure 15. Base Registers (s-i-b Byte Present)

The scaled-index information is determined by multiplying the contents of the Index register by the scale factor. The following example shows the use of the 32-bit addressing mode with scaling where:

- EAX is the base of ARRAY_A
- ECX is the index of the desired element
- 2 is the scale factor.

```
; ARRAY_A is an array of words
MOV EAX, offset ARRAY_A
MOV ECX, element_number
MOV BX, [EAX][ECX*2]
```

The following figure defines the encoding of the 32-bit addressing mode when the s-i-b byte is present.

Note: The mod field is from the mod r/m byte. The base field and scaled-index information are from the s-i-b byte.

Mod Base	32-Bit Address Mode
00 000	DS:[EAX + (scaled index)]
00 001	DS:[ECX + (scaled index)]
00 010	DS:[EDX + (scaled index)]
00 011	DS:[EBX + (scaled index)]
00 100	SS:[ESP + (scaled index)]
00 101	DS:[d32 + (scaled index)]
00 110	DS:[ESI + (scaled index)]
00 111	DS:[EDI + (scaled index)]
01 000	DS:[EAX + (scaled index) + d8]
01 001	DS:[ECX + (scaled index) + d8]
01 010	DS:[EDX + (scaled index) + d8]
01 011	DS:[EBX + (scaled index) + d8]
01 100	SS:[ESP + (scaled index) + d8]
01 101	SS:[EBP + (scaled index) + d8]
01 110	DS:[ESI + (scaled index) + d8]
01 111	DS:[EDI + (scaled index) + d8]
10 000	DS:[EAX + (scaled index) + d32]
10 001	DS:[ECX + (scaled index) + d32]
10 010	DS:[EDX + (scaled index) + d32]
10 011	DS:[EBX + (scaled index) + d32]
10 100	SS:[ESP + (scaled index) + d32]
10 101	SS:[EBP + (scaled index) + d32]
10 110	DS:[ESI + (scaled index) + d32]
10 111	DS:[EDI + (scaled index) + d32]

Figure 16. Effective Address (32-Bit Address Mode – s-i-b Byte Present)

Operand Length (w) Field

For an instruction performing a data operation, the instruction is executed as either a 32-bit or 16-bit operation. Within the constraints of the operation size, the w field encodes the operand size as either one byte or full operation.

w	16-Bit Data Operation	32-Bit Data Operation
0	8 Bits	8 Bits
1	16 Bits	32 Bits

Figure 17. Operand Length Field Encoding

Segment Register (sreg) Field

The 2-bit segment register field (sreg2) allows one of the four 80286 segment registers to be specified. The 3-bit segment register (sreg3) allows the 80386 FS and GS segment registers to be specified.

sreg2	sreg3	Segment Register
00	000	ES
01	001	CS
10	010	SS
11	011	DS
--	100	FS
--	101	GS
--	110	Reserved
--	111	Reserved

Figure 18. Segment Register Field Encoding

General Register (reg) Field

The General register is specified by the reg field, which may appear in the primary opcode bytes as the reg field of the mod reg r/m byte, or as the r/m field of the mod reg r/m byte when mod = 11.

reg	16-Bit (without w)	16-Bit (w = 0)	16-Bit (w =1)	32-Bit (without w)	32-Bit (w =0)	32-Bit (w =1)
000	AX	AL	AX	EAX	AL	EAX
001	CX	CL	CX	ECX	CL	ECX
010	DX	DL	DX	EDX	DL	EDX
011	BX	BL	BX	EBX	BL	EBX
100	SP	AH	SP	ESP	AH	ESP
101	BP	CH	BP	EBP	CH	EBP
110	SI	DH	SI	ESI	DH	ESI
111	DI	BH	DI	EDI	BH	EDI

Figure 19. General Register Field Encoding

The physical addresses of all operands addressed by the BP register are computed using the SS Segment register. For string primitive operations (those addressed by the DI register), addresses of the destination operands are computed using the ES segment, which may not be overridden.

Operation Direction (d) Field

The operation direction (d) field is used in many two-operand instructions to indicate which operand is the source and which is the destination.

d	Direction of Operation
0	Register/Memory <-- Register The "reg" field indicates the source operand; "mod r/m" or "mod ss index base" indicates the destination operand.
1	Register<-- Register/Memory The "reg" field indicates the destination operand; "mod r/m" or "mod ss index base" indicates the source operand.

Figure 20. Operand Direction Field Encoding

Sign-Extend (s) Field

The sign-extend (s) field appears primarily in instructions having immediate data fields. The s field affects only 8-bit immediate data being placed in a 16-bit or 32-bit destination.

s	8-Bit Immediate Data	16/32-Bit Immediate Data
0	No effect on data	No effect on data
1	Sign-extend 8-bit data to fill 16-bit or 32-bit destination	No effect on data

Figure 21. Sign-Extend Field Encoding

Conditional Test (tttn) Field

For conditional instructions (conditional jumps and set-on condition), the conditional test (tttn) field is encoded, with n indicating whether to use the condition (n = 0) or its negation (n = 1), and ttt defining the condition to test.

tttn	Condition	Mnemonic
0000	Overflow	O
0001	No Overflow	NO
0010	Below/Not Above or Equal	B/NAE
0011	Not Below/Above or Equal	NB/AE
0100	Equal/Zero	E/Z
0101	Not Equal/Not Zero	NE/NZ
0110	Below or Equal/Not Above	BE/NA
0111	Not Below or Equal/Above	NBE/A
1000	Sign	S
1001	Not Sign	NS
1010	Parity/Parity Even	P/PE
1011	Not Parity/Parity Odd	NP/PO
1100	Less Than/Not Greater or Equal	L/NGE
1101	Not Less Than/Greater or Equal	NL/GE
1110	Less Than or Equal/Not Greater Than	LE/NG
1111	Not Less or Equal/Greater Than	NLE/G

Figure 22. Conditional Test Field Encoding

Control, Debug, or Test Register (eee) Field

The following shows the encoding for loading and storing the Control, Debug, and Test registers (eee).

eee Code	Interpreted as Control Register	Interpreted as Debug Register	Interpreted as Test Register
000	CR0	DR0	---
001	---	DR1	---
010	CR2	DR2	---
011	CR3	DR3	---
100	---	---	---
101	---	---	---
110	---	DR6	TR6
111	---	DR7	TR7

Figure 23. Control, Debug, and Test Register Field Encoding

80386 Microprocessor Instruction Set

Data Transfer

MOV = Move

Register to Register/Memory

1 0 0 0 1 0 0 w	mod reg r/m

Register/Memory to Register

1 0 0 0 1 0 1 w	mod reg r/m

Immediate to Register/Memory

1 1 0 0 0 1 1 w	mod 0 0 0 r/m	8-, 16-, or 32-bit data

Immediate to Register (Short Form)

1 0 1 1 w reg	8-, 16-, or 32-bit data

Memory to Accumulator (Short Form)

1 0 1 0 0 0 0 w	full 16- or 32-bit displacement

Accumulator to Memory (Short Form)

1 0 1 0 0 0 1 w	full 16- or 32-bit displacement

Register/Memory to Segment Register

1 0 0 0 1 1 1 0	mod sreg3 r/m

Segment Register to Register/Memory

1 0 0 0 1 1 0 0	mod sreg3 r/m

MOVSX = Move with Sign Extension

Register from Register/Memory

00001111	1011111w	mod reg r/m

MOVZX = Move with Zero Extension

Register from Register/Memory

00001111	1011011w	mod reg r/m

PUSH = Push

Register/Memory

11111111	mod 110 r/m

Register (Short Form)

01010 reg

Segment Register (ES, CS, SS, or DS) Short Form

000 sreg2 110

Segment Register (FS or GS)

00001111	10 sreg3 000

Immediate

011010s0	8-, 16-, or 32-bit data

PUSHA = Push All

01100000

POP = Pop

Register/Memory

1 0 0 0 1 1 1 1	mod 0 0 0 r/m

Register (Short Form)

0 1 0 1 1 reg

Segment Register (ES, SS, or DS) Short Form

0 0 0 sreg2 1 1 1

Segment Register (FS or GS)

0 0 0 0 1 1 1 1	1 0 sreg3 0 0 1

POPA = Pop All

0 1 1 0 0 0 0 1

XCHG = Exchange

Register/Memory with Register

1 0 0 0 0 1 1 w	mod reg r/m

Register with Accumulator (Short Form)

1 0 0 1 0 reg

IN = Input From:

Fixed Port

1 1 1 0 0 1 0 w	port number

Variable Port

1 1 1 0 1 1 0 w

OUT = Output To:

Fixed Port

1 1 1 0 0 1 1 w	port number

Variable Port

1 1 1 0 1 1 1 w

LEA = Load EA to Register

1 0 0 0 1 1 0 1	mod reg r/m

Segment Control

LDS = Load Pointer to DS

1 1 0 0 0 1 0 1	mod reg r/m

LES = Load Pointer to ES

1 1 0 0 0 1 0 0	mod reg r/m

LFS = Load Pointer to FS

0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 0	mod reg r/m

LGS = Load Pointer to GS

0 0 0 0 1 1 1 1	1 0 1 1 0 1 0 1	mod reg r/m

LSS = Load Pointer to SS

0 0 0 0 1 1 1 1	1 0 1 1 0 0 1 0	mod reg r/m

Flag Control

CLC = Clear Carry Flag

11111000

CLD = Clear Direction Flag

11111100

CLI = Clear Interrupt Enable Flag

11111010

CLTS = Clear Task Switched Flag

00001111	00000110

CMC = Complement Carry Flag

11110101

LAHF = Load AH Into Flag

10011111

POPF = Pop Flags

10011101

PUSHF = Push Flags

10011100

SAHF = Store AH Into Flags

10011110

STC = Set Carry Flag

11111001

STD = Set Direction Flag

11111101

STI = Set Interrupt Enable Flag

11111011

Arithmetic

ADD = Add

Register to Register

000000dw	mod reg r/m

Register to Memory

0000000w	mod reg r/m

Memory to Register

0000001w	mod reg r/m

Immediate to Register/Memory

100000sw	mod 0 0 0 r/m	8-, 16-, or 32-bit data

Immediate to Accumulator (Short Form)

0000010w	8-, 16-, or 32-bit data

ADC = Add with Carry

Register to Register

0 0 0 1 0 0 d w	mod reg r/m

Register to Memory

0 0 0 1 0 0 0 w	mod reg r/m

Memory to Register

0 0 0 1 0 0 1 w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 s w	mod 0 1 0 r/m	8-, 16-, or 32-bit data

Immediate to Accumulator (Short Form)

0 0 0 1 0 1 0 w	8-, 16-, or 32-bit data

INC = Increment

Register/Memory

1 1 1 1 1 1 1 w	mod 0 0 0 r/m

Register (Short Form)

0 1 0 0 0 reg

SUB = Subtract

Register from Register

0 0 1 0 1 0 d w	mod reg r/m

Register from Memory

0 0 1 0 1 0 0 w	mod reg r/m

Memory from Register

0 0 1 0 1 0 1 w	mod reg r/m

Immediate from Register/Memory

1 0 0 0 0 0 s w	mod 1 0 1 r/m	8-, 16-, or 32-bit data

Immediate from Accumulator (Short Form)

0 0 1 0 1 1 0 w	8-, 16-, or 32-bit data

SBB = Subtract with Borrow

Register from Register

0 0 0 1 1 0 d w	mod reg r/m

Register from Memory

0 0 0 1 1 0 0 w	mod reg r/m

Memory from Register

0 0 0 1 1 0 1 w	mod reg r/m

Immediate from Register/Memory

1 0 0 0 0 0 s w	mod 0 1 1 r/m	8-, 16-, or 32-bit data

Immediate from Accumulator (Short Form)

0 0 0 1 1 1 0 w	8-, 16-, or 32-bit data

DEC = Decrement

Register/Memory

1 1 1 1 1 1 1 w	mod 0 0 1 r/m

Register (Short Form)

0 1 0 0 1 reg

CMP = Compare

Register with Register

0 0 1 1 1 0 d w	mod reg r/m

Memory with Register

0 0 1 1 1 0 0 w	mod reg r/m

Register with Memory

0 0 1 1 1 0 1 w	mod reg r/m

Immediate with Register/Memory

1 0 0 0 0 0 s w	mod 1 1 1 r/m	8-, 16-, or 32-bit data

Immediate with Accumulator (Short Form)

0 0 1 1 1 1 0 w	8-, 16-, or 32-bit data

NEG = Change Sign

1 1 1 1 0 1 1 w	mod 0 1 1 r/m

AAA = ASCII Adjust for Add

0 0 1 1 0 1 1 1

AAS = ASCII Adjust for Subtract

0 0 1 1 1 1 1 1

DAA = Decimal Adjust for Add

0 0 1 0 0 1 1 1

DAS = Decimal Adjust for Subtract

0 0 1 0 1 1 1 1

MUL = Multiply (Unsigned)

Accumulator with Register/Memory

1 1 1 1 0 1 1 w	mod 1 0 0 r/m

IMUL = Integer Multiply (Signed)

Accumulator with Register/Memory

1 1 1 1 0 1 1 w	mod 1 0 1 r/m

Register with Register/Memory

0 0 0 0 1 1 1 1	1 0 1 0 1 1 1 1	mod reg r/m

Register/Memory with Immediate to Register

0 1 1 0 1 0 s 1	mod reg r/m	8-, 16-, or 32-bit data

DIV = Divide (Unsigned)

Accumulator by Register/Memory

1 1 1 1 0 1 1 w	mod 1 1 0 r/m

IDIV = Integer Divide (Signed)

Accumulator by Register/Memory

1 1 1 1 0 1 1 w	mod 1 1 1 r/m

AAD = ASCII Adjust for Divide

1 1 0 1 0 1 0 1	0 0 0 0 1 0 1 0

AAM = ASCII Adjust for Multiply

1 1 0 1 0 1 0 0	0 0 0 0 1 0 1 0

CBW = Convert Byte to Word

1 0 0 1 1 0 0 0

CWD = Convert Word to Doubleword

1 0 0 1 1 0 0 1

Logic

Shift/Rotate Instructions
Not Through Carry (ROL, ROR, SAL, SAR, SHL, and SHR)

Register/Memory by 1

1 1 0 1 0 0 0 w	mod T T T r/m

Register/Memory by CL

1 1 0 1 0 0 1 w	mod T T T r/m

Register/Memory by Immediate Count

1 1 0 0 0 0 0 w	mod T T T r/m	8-bit data

Shift/Rotate Instructions Through Carry (RCL and RCR)

Register/Memory by 1

1 1 0 1 0 0 0 w	mod T T T r/m

Register/Memory by CL

1 1 0 1 0 0 1 w	mod T T T r/m

Register/Memory by Immediate Count

1 1 0 0 0 0 0 w	mod T T T r/m	8-bit data

T T T	Instruction
0 0 0	ROL
0 0 1	ROR
0 1 0	RCL
0 1 1	RCR
1 0 0	SHL/SAL
1 0 1	SHR
1 1 1	SAR

SHLD = Shift Left Double

Register/Memory by Immediate

0 0 0 0 1 1 1 1	1 0 1 0 0 1 0 0	mod reg r/m	8-bit data

Register/Memory by CL

0 0 0 0 1 1 1 1	1 0 1 0 0 1 0 1	mod reg r/m

SHRD = Shift Right Double

Register/Memory by Immediate

0 0 0 0 1 1 1 1	1 0 1 0 1 1 0 0	mod reg r/m	8-bit data

Register/Memory by CL

0 0 0 0 1 1 1 1	1 0 1 0 1 1 0 1	mod reg r/m

AND = And

Register to Register

0 0 1 0 0 0 d w	mod reg r/m

Register to Memory

0 0 1 0 0 0 0 w	mod reg r/m

Memory to Register

0 0 1 0 0 0 1 w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 s w	mod 1 0 0 r/m	8-, 16-, or 32-bit data

Immediate to Accumulator (Short Form)

0 0 1 0 0 1 0 w	8-, 16-, or 32-bit data

TEST = AND Function to Flags; No Result

Register/Memory and Register

1 0 0 0 0 1 0 w	mod reg r/m

Immediate Data and Register/Memory

1 1 1 1 0 1 1 w	mod 0 0 0 r/m	8-, 16-, or 32-bit data

Immediate Data and Accumulator (Short Form)

1 0 1 0 1 0 0 w	8-, 16-, or 32-bit data

OR = Or

Register to Register

0 0 0 0 1 0 d w	mod reg r/m

Register to Memory

0 0 0 0 1 0 0 w	mod reg r/m

Memory to Register

0 0 0 0 1 0 1 w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 s w	mod 0 0 1 r/m	8-, 16-, or 32-bit data

Immediate to Accumulator (Short Form)

0 0 0 0 1 1 0 w	8-, 16-, or 32-bit data

XOR = Exclusive OR

Register to Register

0 0 1 1 0 0 d w	mod reg r/m

Register to Memory

0 0 1 1 0 0 0 w	mod reg r/m

Memory to Register

0 0 1 1 0 0 1 w	mod reg r/m

Immediate to Register/Memory

1 0 0 0 0 0 s w	mod 1 1 0 r/m	8-, 16-, or 32-bit data

Immediate to Accumulator (Short Form)

0 0 1 1 0 1 0 w	8-, 16-, or 32-bit data

NOT = Invert Register/Memory

1 1 1 1 0 1 1 w	mod 0 1 0 r/m

String Manipulation

CMPS = Compare Byte Word

1 0 1 0 0 1 1 w

INS = Input Byte/Word from DX Port

0 1 1 0 1 1 0 w

LODS = Load Byte/Word to AL/AX/EAX

1010110w

MOVS = Move Byte Word

1010010w

OUTS = Output Byte/Word to DX Port

0110111w

SCAS = Scan Byte Word

1010111w

STOS = Store Byte/Word from AL/AX/EX

1010101w

XLAT = Translate String

11010111

Repeated String Manipulation

Repeated by Count in CX or ECX

REPE CMPS = Compare String (Find Non-Match)

11110011	1010011w

REPNE CMPS = Compare String (Find Match)

11110010	1010011w

REP INS = Input String

11110010	0110110w

REP LODS = Load String

11110010	1010110w

REP MOVS = Move String

11110010	1010010w

REP OUTS = Output String

11110010	0110111w

REPE SCAS = Scan String (Find Non-AL/AX/EAX)

11110011	1010111w

REPNE SCAS = Scan String (Find AL/AX/EAX)

11110010	1010111w

REP STOS = Store String

11110010	1010101w

Bit Manipulation

BSF = Scan Bit Forward

0 0 0 0 1 1 1 1	1 0 1 1 1 1 0 0	mod reg r/m

BSR = Scan Bit Reverse

0 0 0 0 1 1 1 1	1 0 1 1 1 1 0 1	mod reg r/m

BT = Test Bit

Register/Memory, Immediate

0 0 0 0 1 1 1 1	1 0 1 1 1 0 1 0	mod 1 0 0 r/m	8-bit data

Register/Memory, Register

0 0 0 0 1 1 1 1	1 0 1 0 0 0 1 1	mod reg r/m

BTC = Test Bit and Complement

Register/Memory, Immediate

0 0 0 0 1 1 1 1	1 0 1 1 1 0 1 0	mod 1 1 1 r/m	8-bit data

Register/Memory, Register

0 0 0 0 1 1 1 1	1 0 1 1 1 0 1 1	mod reg r/m

BTR = Test Bit and Reset

Register/Memory, Immediate

0 0 0 0 1 1 1 1	1 0 1 1 1 0 1 0	mod 1 1 0 r/m	8-bit data

Register/Memory, Register

0 0 0 0 1 1 1 1	1 0 1 1 0 0 1 1	mod reg r/m

BTS = Test Bit and Set

Register/Memory, Immediate

00001111	10111010	mod 1 0 1 r/m	8-bit data

Register/Memory, Register

00001111	10101011	mod reg r/m

Control Transfer

CALL = Call

Direct within Segment

11101000	full 16- or 32-bit displacement

Register/Memory Indirect within Segment

11111111	mod 0 1 0 r/m

Direct Intersegment

10011010	offset, selector

Indirect Intersegment

11111111	mod 0 1 1 r/m

JMP = Unconditional Jump

Short

11101011	8-bit disp.

Direct within Segment

11101001	full 16- or 32-bit displacement

Register/Memory Indirect within Segment

1 1 1 1 1 1 1 1	mod 1 0 0 r/m

Direct Intersegment

1 1 1 0 1 0 1 0	offset, selector

Indirect Intersegment

1 1 1 1 1 1 1 1	mod 1 0 1 r/m

RET = Return from Call

Within Segment

1 1 0 0 0 0 1 1

Within Segment Adding Immediate to SP

1 1 0 0 0 0 1 0	16-bit displacement

Intersegment

1 1 0 0 1 0 1 1

Intersegment Adding Immediate to SP

1 1 0 0 1 0 1 0	16-bit displacement

Conditional Jumps

JO = Jump on Overflow

8-Bit Displacement

0 1 1 1 0 0 0 0	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 0 0 0 0	full 16- or 32-bit displacement

JNO = Jump on Not Overflow

8-Bit Displacement

01110001	8-bit disp.

Full Displacement

00001111	10000001	full 16- or 32-bit displacement

JB/JNAE = Jump on Below/Not Above or Equal

8-Bit Displacement

01110010	8-bit disp.

Full Displacement

00001111	10000010	full 16- or 32-bit displacement

JNB/JAE = Jump on Not Below/Above or Equal

8-Bit Displacement

01110011	8-bit disp.

Full Displacement

00001111	10000011	full 16- or 32-bit displacement

JE/JZ = Jump on Equal/Zero

8-Bit Displacement

01110100	8-bit disp.

Full Displacement

00001111	10000100	full 16- or 32-bit displacement

JNE/JNZ = Jump on Not Equal/Not Zero

8-Bit Displacement

01110101	8-bit disp.

Full Displacement

00001111	10000101	full 16- or 32-bit displacement

JBE/JNA = Jump on Below or Equal/Not Above

8-Bit Displacement

01110110	8-bit disp.

Full Displacement

00001111	10000110	full 16- or 32-bit displacement

JNBE/JA = Jump on Not Below or Equal/Above

8-Bit Displacement

01110111	8-bit disp.

Full Displacement

00001111	10000111	full 16- or 32-bit displacement

JS = Jump on Sign

8-Bit Displacement

01111000	8-bit disp.

Full Displacement

00001111	10001000	full 16- or 32-bit displacement

JNS = Jump on Not Sign

8-Bit Displacement

0 1 1 1 1 0 0 1	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 0 0 1	full 16- or 32-bit displacement

JP/JPE = Jump on Parity/Parity Even

8-Bit Displacement

0 1 1 1 1 0 1 0	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 0 1 0	full 16- or 32-bit displacement

JNP/JPO = Jump on Not Parity/Parity Odd

8-Bit Displacement

0 1 1 1 1 0 1 1	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 0 1 1	full 16- or 32-bit displacement

JL/JNGE = Jump on Less/Not Greater or Equal

8-Bit Displacement

0 1 1 1 1 1 0 0	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 1 0 0	full 16- or 32-bit displacement

JNL/JGE = Jump on Not Less/Greater or Equal

8-Bit Displacement

0 1 1 1 1 1 0 1	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 1 0 1	full 16- or 32-bit displacement

JLE/JNG = Jump on Less or Equal/Not Greater

8-Bit Displacement

0 1 1 1 1 1 1 0	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 1 1 0	full 16- or 32-bit displacement

JNLE/JG = Jump on Not Less or Equal/Greater

8-Bit Displacement

0 1 1 1 1 1 1 1	8-bit disp.

Full Displacement

0 0 0 0 1 1 1 1	1 0 0 0 1 1 1 1	full 16- or 32-bit displacement

JCXZ = Jump on CX Zero

1 1 1 0 0 0 1 1	8-bit disp.

JECXZ = Jump on ECX Zero

1 1 1 0 0 0 1 1	8-bit disp.

Note: The operand size prefix differentiates JCXZ from JECXZ.

LOOP = Loop CX Times

1 1 1 0 0 0 1 0	8-bit disp.

LOOPZ/LOOPE = Loop with Zero/Equal

1 1 1 0 0 0 0 1	8-bit disp.

LOOPNZ/LOOPNE = Loop while Not Zero

1 1 1 0 0 0 0 0	8-bit disp.

Conditional Byte Set

SETO = Set Byte on Overflow

To Register/Memory

0 0 0 0 1 1 1 1	1 0 0 1 0 0 0 0	mod 0 0 0 r/m

SETNO = Set Byte on Not Overflow

To Register/Memory

0 0 0 0 1 1 1 1	1 0 0 1 0 0 0 1	mod 0 0 0 r/m

SETB/SETNAE = Set Byte on Below/Not Above or Equal

To Register/Memory

0 0 0 0 1 1 1 1	1 0 0 1 0 0 1 0	mod 0 0 0 r/m

SETNB = Set Byte on Not Below/Above or Equal

To Register/Memory

0 0 0 0 1 1 1 1	1 0 0 1 0 0 1 1	mod 0 0 0 r/m

SETE/SETZ = Set Byte on Equal/Zero

To Register/Memory

0 0 0 0 1 1 1 1	1 0 0 1 0 1 0 0	mod 0 0 0 r/m

SETNE/SETNZ = Set Byte on Not Equal/Not Zero

To Register/Memory

00001111	10010101	mod 000 r/m

SETBE/SETNA = Set Byte on Below or Equal/Not Above

To Register/Memory

00001111	10010110	mod 000 r/m

SETNBE/SETA = Set Byte on Not Below or Equal/Above

To Register/Memory

00001111	10010111	mod 000 r/m

SETS = Set Byte on Sign

To Register/Memory

00001111	10011000	mod 000 r/m

SETNS = Set Byte on Not Sign

To Register/Memory

00001111	10011001	mod 000 r/m

SETP/SETPE = Set Byte on Parity/Parity Even

To Register/Memory

00001111	10011010	mod 000 r/m

SETNP/SETPO = Set Byte on Not Parity/Parity Odd

To Register/Memory

00001111	10011011	mod 000 r/m

SETL/SETNGE = Set Byte on Less/Not Greater or Equal

To Register/Memory

00001111	10011100	mod 000 r/m

SETNL/SETGE = Set Byte on Not Less/Greater or Equal

To Register/Memory

00001111	01111101	mod 000 r/m

SETLE/SETNG = Set Byte on Less or Equal/Not Greater

To Register/Memory

00001111	10011110	mod 000 r/m

SETNLE/SETG = Set Byte on Not Less or Equal/Greater

To Register/Memory

00001111	10011111	mod 000 r/m

ENTER = Enter Procedure

11001000	16-bit displacement	8-bit level

LEAVE = Leave Procedure

11001001

Interrupt Instructions

INT = Interrupt

Type Specified

11001101	type

Type 3

11001100

INTO = Interrupt 4 if Overflow Flag Set

11001110

BOUND = Interrupt 5 If Detect Value Out of Range

01100010	mod reg r/m

IRET = Interrupt Return

11001111

Processor Control

HLT = Halt

11110100

MOV = Move to and from Control/Debug/Test Registers

CR0/CR2/CR3 from Register

00001111	00100010	11 eee reg

Register from CR0-3

00001111	00100000	11 eee reg

DR0-3, DR6-7 from Register

00001111	00100011	11 eee reg

Register from DR0-3, DR6-7

00001111	00100001	11 eee reg

TR6-7 from Register

00001111	00100110	11 eee reg

Register from TR6-7

00001111	00100100	11 eee reg

NOP = No Operation

10010000

WAIT = Wait until BUSY Pin is Negated

10011011

Processor Extension

ESC = Processor Extension Escape

11011TTT	mod L L L r/m

Note: TTT and LLL bits are opcode information for the coprocessor.

Prefix Bytes

Address Size Prefix

01100111

Operand Size Prefix

01100110

LOCK = Bus Lock Prefix

11110000

Note: The use of LOCK is restricted to an exchange with memory, or bit test and reset type of instruction.

Segment Override Prefix

CS:

00101110

DS:

0 0 1 1 1 1 1 0

ES:

0 0 1 0 0 1 1 0

FS:

0 1 1 0 0 1 0 0

GS:

0 1 1 0 0 1 0 1

SS:

0 0 1 1 0 1 1 0

Protection Control

ARPL = Adjust Requested Privilege Level from Register/Memory

0 1 1 0 0 0 1 1	mod reg r/m

LAR = Load Access Rights from Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 1 0	mod reg r/m

LGDT = Load Global Descriptor Table Register

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 0 1 0 r/m

LIDT = Load Interrupt Descriptor Table Register

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 0 1 1 r/m

LLDT = Load Local Descriptor Table Register to Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 0 1 0 r/m

LMSW = Load Machine Status Word from Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 1 1 0 r/m

LSL = Load Segment Limit from Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 1 1	mod reg r/m

LTR = Load Task Register from Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 0 0 1 r/m

SGDT = Store Global Descriptor Table Register

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 0 0 0 r/m

SIDT = Store Interrupt Descriptor Table Register

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 0 0 1 r/m

SLDT = Store Local Descriptor Table Register to Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 0 0 0 r/m

SMSW = Store Machine Status Word

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 1	mod 1 0 0 r/m

STR = Store Task Register to Register/Memory

0 0 0 0 1 1 1 1	0 0 0 0 0 0 0 0	mod 0 0 1 r/m

VERR = Verify Read Access; Register/Memory

00001111	00000000	mod 1 0 0 r/m

VERW = Verify Write Access

00001111	00000000	mod 1 0 1 r/m

Introduction to the 80387 Instruction Set

The 80387 instructions use many of the same fields defined earlier in this section for the 80386 instructions. Additional fields used by the 80387 instructions are defined in the following figure.

Field	Description	Bit Information
escape	80386 Extension Escape	Bit Pattern = 11011
MF	Memory Format	00 = 32-bit Real 01 = 32-bit integer 10 = 64-bit Real 11 = 16-bit integer
ST(0)	Current Stack Top	
ST(i)	ith register below the stack top	
d	Destination	0 = Destination is ST(0) 1 = Destination is ST(i)
P	Pop	0 = No pop 1 = Pop ST(0)
R	Reverse*	0 = Destination (op) source 1 = Source (op) destination

* When d = 1, reverse the sense of R.

Figure 24. 80387 Encoding Field Summary

Within the 80387 Instruction Set:

- Temporary (Extended) Real is 80-bit Real.
- Long Integer is a 64-bit integer.

80387 Usage of the Scale-Index-Base Byte

The "mod r/m" byte of an 80387 instruction can be followed by a scale-index-base (s-i-b) byte having the same address mode definition as in the 80386 instruction. The mod field in the 80387 instruction is never equal to 11.

Instruction and Data Pointers

The parallel operation of the 80386 and 80387 may allow errors detected by the 80387 to be reported after the 80386 has executed the ESC instruction that caused the error. The 80386/80387 provides two pointer registers to identify the failing numeric instruction. The pointer registers supply the address of the failing numeric instruction and the address of its numeric memory operand when applicable.

Although the pointer registers are located in the 80386, they appear to be located in the 80387 because they are accessed by the ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR. Whenever the 80386 decodes a new ESC instruction, it saves the address of the instruction along with any prefix bytes that may be present, the address of the operand (if present), and the opcode.

The instruction and data pointers appear in one of four available formats:

- 16-bit Real Mode/Virtual 8086 Mode
- 32-bit Real Mode
- 16-bit Protected Mode
- 32-bit Protected Mode

The Real Mode formats are used whenever the 80386 is in the Real Mode or Virtual 8086 Mode. The Protected Mode formats are used when the 80386 is in the Protected Mode. The Operand Size Prefix can also be used with the 80387 instructions. The operand size of the 80387 instruction determines whether the 16-bit or 32-bit format is used.

Note: FSAVE and FRSTOR have an additional eight fields (10 bytes per field) that contain the current contents of ST(0) through ST(7). These fields follow the instruction and data pointer image shown in the following figures.

The following figures show the instruction and data pointer image format used in the various address modes. The ESC instructions FLDENV, FSTENV, FSAVE, and FRSTOR are used to transfer these values between the 80386/80387 registers and memory.

Bits

15 … 8		7 … 0	Offset in Block
Control Word			0
Status Word			2
Tag Word			4
Instruction Pointer (IP) Bits 15-0			6
IP Bits 19-16	0	Opcode Bits 10-0	8
Operand Pointer (OP) Bits 15-0			A
OP Bits 19-16	0	0 0 0 0 0 0 0 0 0 0 0	C

Figure 25. Instruction and Pointer Image (16-Bit Real Address Mode)

Bits

15 … 8 7 … 0	Offset in Block
Control Word	0
Status Word	2
Tag Word	4
Instruction Pointer Offset	6
CS Selector	8
Operand Offset	A
Operand Selector	C

Figure 26. Instruction and Pointer Image (16-Bit Protected Mode)

Bits

31 … 24	23 … 16	15 … 8	7 … 0	Offset in Block
Reserved		Control Word		0
Reserved		Status Word		4
Reserved		Tag Word		8
Reserved		IP Bits 15-0		C
0 0 0 0	IP Bits 31-16	0	Opcode Bits 10-0	10
Reserved		Operand Pointer Bits 15-0		14
0 0 0 0	Operand Pointer Bits 31-16	0 0 0 0 0 0 0 0 0 0 0 0		18

Figure 27. Instruction and Pointer Image (32-Bit Real Address Mode)

Bits

31 24	23 16	15 8	7 0	Offset in Block
Reserved		Control Word		0
Reserved		Status Word		4
Reserved		Tag Word		8
Instruction Pointer Offset				C
Reserved		CS Selector		10
Data Operand Offset				14
Reserved		Operand Selector		18

Figure 28. Instruction and Pointer Image (32-Bit Protected Mode)

New Instructions

Several new instructions are included in the 80387 instruction set that are not available to the 80287 or 8087 math coprocessors. The new instructions are:

FUCOM (Unordered Compare Real)
FUCOMP (Unordered Compare Real and Pop)
FUCOMPP (Unordered Compare Real and Pop Twice)
FPREM1 (IEEE Partial Remainder)
FSINE (Sine)
FCOS (Cosine)
FSINCOS (Sine and Cosine).

80387 Math Coprocessor Instruction Set

The following is an instruction set summary for the 80387 coprocessor. In the following, the bit pattern for escape is 11011.

Data Transfer

FLD = Load

Integer/Real Memory to ST(0)

escape MF 1	mod 0 0 0 r/m

Long Integer Memory to ST(0)

escape 1 1 1	mod 1 0 1 r/m

Temporary Real Memory to ST(0)

escape 0 1 1	mod 1 0 1 r/m

BCD Memory to ST(0)

escape 1 1 1	mod 1 0 0 r/m

ST(i) to ST(0)

escape 0 0 1	1 1 0 0 0 ST(i)

FST = Store

ST(0) to Integer/Real Memory

escape MF 1	mod 0 1 0 r/m

ST(0) to ST(i)

escape 1 0 1	1 1 0 1 0 ST(i)

FSTP = Store and Pop

ST(0) to Integer/Real Memory

escape MF 1	mod 0 1 1 r/m

ST(0) to Long Integer Memory

escape 1 1 1	mod 1 1 1 r/m

ST(0) to Temporary Real Memory

escape 0 1 1	mod 1 1 1 r/m

ST(0) to BCD Memory

escape 1 1 1	mod 1 1 0 r/m

ST(0) to ST(i)

escape 1 0 1	1 1 0 1 1 ST(i)

FXCH = Exchange ST(I) and ST(0)

escape 0 0 1	1 1 0 0 1 ST(i)

Comparison

FCOM = Compare

Integer/Real Memory to ST(0)

escape MF 0	mod 0 1 0 r/m

ST(i) to ST(0)

escape 0 0 0	1 1 0 1 0 ST(i)

FCOMP = Compare and Pop

Integer/Real Memory to ST(0)

escape MF 0	mod 0 1 1 r/m

ST(i) to ST(0)

escape 0 0 0	1 1 0 1 1 ST(i)

FCOMPP = Compare ST(1) to ST(0) and Pop Twice

escape 1 1 0	1 1 0 1 1 0 0 1

FUCOM = Unordered Compare Real

escape 1 0 1	1 1 1 0 0 ST(i)

FUCOMP = Unordered Compare Real and Pop

escape 1 0 1	1 1 1 0 1 ST(i)

FUCOMPP = Unordered Compare Real and Pop Twice

escape 0 1 0	1 1 1 0 1 0 0 1

FTST = Test ST(0)

escape 0 0 1	1 1 1 0 0 1 0 0

FXAM = Examine ST(0)

escape 0 0 1	1 1 1 0 0 1 0 1

Constants

FLDZ = Load +0.0 Into ST(0)

escape 0 0 1	1 1 1 0 1 1 1 0

FLD1 = Load +1.0 Into ST(0)

escape 0 0 1	1 1 1 0 1 0 0 0

FLDPI = Load π Into ST(0)

escape 0 0 1	1 1 1 0 1 0 1 1

FLDL2T = Load $\log_2$ 10 Into ST(0)

escape 0 0 1	1 1 1 0 1 0 0 1

FLDL2E = Load $\log_2$ e Into ST(0)

escape 0 0 1	1 1 1 0 1 0 1 0

FLDLG2 = Load $\log_{10}$ 2 Into ST(0)

escape 0 0 1	1 1 1 0 1 1 0 0

FLDLN2 = Load $\log_e$ 2 Into ST(0)

escape 0 0 1	1 1 1 0 1 1 0 1

Arithmetic

FADD = Addition

Integer/Real Memory with ST(0)

escape MF 0	mod 0 0 0 r/m

ST(i) and ST(0)

escape d P 0	1 1 0 0 0 ST(i)

FSUB = Subtraction

Integer/Real Memory with ST(0)

escape MF 0	mod 1 0 R r/m

ST(i) and ST(0)

escape d P 0	1 1 1 0 R r/m

FMUL = Multiplication

Integer/Real Memory with ST(0)

escape MF 0	mod 0 0 1 r/m

ST(i) and ST(0)

escape d P 0	1 1 0 0 1 r/m

FDIV = Division

Integer/Real Memory with ST(0)

escape MF 0	mod 1 1 R r/m

ST(i) and ST(0)

escape d P 0	1 1 1 1 R r/m

FSQRT = Square Root of ST(0)

escape 0 0 1 1 1 1 1 1 0 1 0

FSCALE = Scale ST(0) by ST(1)

escape 0 0 1 1 1 1 1 1 1 0 1

FPREM = Partial Remainder of ST(0) ÷ ST(1)

escape 0 0 1 1 1 1 1 1 0 0 0

FPREM1 = IEEE Partial Remainder

escape 0 0 1 1 1 1 1 0 1 0 1

FRNDINT = Round ST(0) to Integer

escape 0 0 1 1 1 1 1 1 1 0 0

FXTRACT = Extract Components of ST(0)

escape 0 0 1 1 1 1 1 0 1 0 0

FABS = Absolute Value of ST(0)

escape 0 0 1 1 1 1 0 0 0 0 1

FCHS = Change Sign of ST(0)

escape 0 0 1 1 1 1 0 0 0 0 0

Transcendental

FPTAN = Partial Tangent of ST(0)

escape 0 0 1 1 1 1 1 0 0 1 0

FPATAN = Partial Arctangent of ST(1) ÷ ST(0)

escape 0 0 1	1 1 1 1 0 0 1 1

FSIN = Sine

escape 0 0 1	1 1 1 1 1 1 1 0

FCOS = Cosine

escape 0 0 1	1 1 1 1 1 1 1 1

FSINCOS = Sine and Cosine

escape 0 0 1	1 1 1 1 1 0 1 1

F2XM1 = $2^{ST(0)}$ -1

escape 0 0 1	1 1 1 1 0 0 0 0

FYL2X = ST(1) x Log_2 [ST(0)]

escape 0 0 1	1 1 1 1 0 0 0 1

FYL2XP1 = ST(1) x Log_2 [ST(0) + 1]

escape 0 0 1	1 1 1 1 1 0 0 1

Processor Control

FINIT = Initialize NPX

escape 0 1 1	1 1 1 0 0 0 1 1

FSTSW AX = Store Control Word

escape 1 1 1	1 1 1 0 0 0 0 0

FLDCW = Load Control Word

escape 0 0 1	mod 1 0 1 r/m

FSTCW = Store Control Word

escape 0 0 1	mod 1 1 1 r/m

FSTSW = Store Status Word

escape 1 0 1	mod 1 1 1 r/m

FCLEX = Clear Exceptions

escape 0 1 1	1 1 1 0 0 0 1 0

FSTENV = Store Environment

escape 0 0 1	mod 1 1 0 r/m

FLDENV = Load Environment

escape 0 0 1	mod 1 0 0 r/m

FSAVE = Save State

escape 1 0 1	mod 1 1 0 r/m

FRSTOR = Restore State

escape 1 0 1	mod 1 0 0 r/m

FINCSTP = Increment Stack Pointer

escape 0 0 1	1 1 1 1 0 1 1 1

FDECSTP = Decrement Stack Pointer

escape 0 0 1	1 1 1 1 0 1 1 0

FFREE = Free ST(i)

escape 1 0 1	1 1 0 0 0 ST(i)

FNOP = No Operation

escape 0 0 1	1 1 0 1 0 0 0 0

Index

A

AAA instruction 20, 63
AAD instruction 21, 64
AAM instruction 21, 64
AAS instruction 20, 63
ADC instruction 18, 60
ADD instruction 18, 59
address mode 47
address size prefix 82
address space 1, 2, 5, 6
AND instruction 22, 66
arithmetic instructions 18, 39, 59, 94
ARPL instruction 33, 83

B

bit manipulation instructions 71
BOUND instruction 30, 81
BSF instruction 71
BT instruction 71
BTC instruction 71
BTR instruction 71
BTS instruction 72

C

CALL instruction 26, 72
CBW instruction 21, 64
CLC instruction 30, 58
CLD instruction 31, 58
CLI instruction 31, 58
clock generator 3
CLTS instruction 58
CMC instruction 30
CMP instruction 20, 62
CMPS instruction 24, 68
comparison instructions 37, 91
conditional byte set 78
conditional jumps 73
conditional test field 52
constants 38, 93
control register field 53
control transfer instructions 26, 72
CTS instruction 31
CWD instruction 21, 64

D

DAA instruction 20, 63
DAS instruction 20, 63
data pointers 87
data transfer 35
data transfer instructions 14, 35, 54, 90
data types 2, 3, 12
data types, math coprocessor 3
debug register field 53
DEC instruction 19, 62
descriptors 6, 43
DIV instruction 21, 64

E

effective address 47
effective address size prefix 44
encoding field summary 46, 86
encoding, instructions 46
ENTER instruction 29, 80
ESC instruction 31, 82
exception 1, 5
exception conditions 4

F
FABS instruction 40, 95
FADD instruction 39, 94
FCHS instruction 40, 95
FCLEX instruction 42, 97
FCOM instruction 37, 91
FCOMP instruction 37, 92
FCOMPP instruction 37, 92
FCOS instruction 96
FDECSTP instruction 42, 98
FDIV instruction 39, 94
FFREE instruction 42, 98
FINCSTP instruction 42, 97
FINIT instruction 96
FINT instruction 41
flag control instructions 58
FLD instruction 35, 90
FLDCW instruction 41, 97
FLDENV instruction 42, 97
FLDLG2 instruction 38, 93
FLDLN2 instruction 38, 93
FLDL2E instruction 38, 93
FLDL2T instruction 38, 93
FLDPI instruction 38, 93
FLDZ instruction 38, 93
FLD1 instruction 38, 93
FMUL instruction 39, 94
FNOP instruction 42, 98
FPATAN instruction 40, 96
FPREM instruction 40, 95
FPREM1 instruction 95
FPTAN instruction 40, 95
FRNDINT instruction 40, 95
FRSTOR instruction 42, 97
FSAVE instruction 42, 97
FSCALE instruction 40, 95
FSETPM instruction 41
FSIN instruction 96
FSINCOS instruction 96
FSQRT instruction 40, 95
FST instruction 36, 90
FSTCW instruction 41, 97
FSTENV instruction 42, 97
FSTP instruction 36, 91
FSTSW AX instruction 41, 96
FSTSW instruction 41, 97
FSUB instruction 39, 94
FTST instruction 38, 92
FUCOM instruction 92
FUCOMP instruction 92
FUCOMPP instruction 92
FXAM instruction 38, 92
FXCH instruction 37, 91
FXTRACT instruction 40, 95
FYL2X instruction 41, 96
FYL2XP1 instruction 41, 96
F2XM1 instruction 41, 96

G
general register field 51
global descriptor table 6

H
hardware interface 3, 12
HLT instruction 31, 81

I
I/O ports 3
IDIV instruction 21, 64
IIMUL instruction 21
IMUL instruction 21, 63
IN instruction 16, 56
INC instruction 18, 60
INS instruction 24, 68
instruction pointers 87
instruction set, math
  coprocessor 86
INT instruction 30, 80
interrupt 1, 5
interrupt instructions 80
interrupts 4
INTO instruction 30, 80

IRET instruction 30, 81

J

JB/JNAE instruction 27, 74
JBE/JNA instruction 28, 75
JCXZ instruction 29, 77
JE/JZ instruction 27, 74
JECXZ instruction 77
JL/JNGE instruction 27, 76
JLE/JNG instruction 27, 77
JMP instruction 26, 72
JNB/JAE instruction 28, 74
JNBE/JA instruction 28, 75
JNE/JNZ instruction 28, 75
JNL/JGE instruction 28, 77
JNLE/JG instruction 28, 77
JNO instruction 29, 74
JNP/JPO instruction 29, 76
JNS instruction 29, 76
JO instruction 28, 73
JP/JPE instruction 28, 76
JS instruction 28, 75

L

LAHF instruction 17, 58
LAR instruction 33, 83
LDS instruction 17, 57
LEA instruction 17, 57
LEAVE instruction 29, 80
LES instruction 17, 57
LFS instruction 57
LGDT instruction 32, 83
LGS instruction 57
LIDT instruction 32, 83
LLDT instruction 32, 84
LMSW instruction 32, 84
local descriptor table 6
LOCK instruction 31, 82
LODS instruction 24, 69
logic instructions 22, 64
LOOP instruction 29, 78
LOOPNZ/LOOPNE instruction 29, 78
LOOPZ/LOOPE instruction 29, 78
LSL instruction 33, 84
LSS instruction 57
LTR instruction 32, 84

M

memory, virtual 2
mode, protected 2
mode, real address 5
mode, real-address 1
MOV instruction 14, 54, 81
MOVS instruction 24, 69
MOVSX instruction 55
MOVZX instruction 55
MUL instruction 21, 63

N

NEG instruction 20, 63
NOP instruction 82
NOT instruction 24, 68

O

operand length field 50
operand size 44
operand size prefix 44, 82
operation direction field 52
OR instruction 23, 67
OUT instruction 16, 57
OUTS instruction 24, 69

P

page directory 8
page directory physical base address register 9
page fault linear address register 8
page frame 8
page tables 8

paging 6
paging mechanism, 80386 8
pointer 1, 2, 5
pointer image 87
pointer registers 87
POP instruction 15, 56
POPA instruction 16, 56
POPF instruction 17, 58
prefix bytes 82
processor control instructions 30, 81, 96
processor extension 82
programming interface 3
protected mode 2, 6, 11
protected virtual address mode 2, 6
protection control instructions 32, 83
PUSH instruction 15, 55
PUSHA instruction 15, 55
PUSHF instruction 17, 58

R
real address mode 5, 11
real-address mode 1
registers 3
REP INS instruction 70
REP LODS instruction 70
REP MOVS instruction 70
REP OUTS instruction 70
REP STOS instruction 70
REP/REPNE, REPZ/REPNZ instructions 25
REPE CMPS instruction 69
REPE SCAS instruction 70
repeated string manipulation instructions 69
REPNE CMPS instruction 70
REPNE SCAS instruction 70
RET instruction 27, 73
rotate instructions 22, 64

S
SAHF instruction 17, 59
SBB instruction 19, 61
scale-index-base byte 47
SCAS instruction 24, 69
segment control instructions 57
segment descriptors 43
segment override prefix 82
segment register field 51
segments 1, 5
SETB/SETNAE instruction 78
SETBE/SETNA instruction 79
SETE/SETZ instruction 78
SETL/SETNGE instruction 79
SETLE/SETNG instruction 80
SETNB instruction 78
SETNBE/SETA instruction 79
SETNE/SETNZ instruction 79
SETNL/SETGE instruction 80
SETNLE/SETG instruction 80
SETNO instruction 78
SETNP/SETPO instruction 79
SETNS instruction 79
SETO instruction 78
SETP/SETPE instruction 79
SETS instruction 79
SGDT instruction 32, 84
shift instructions 22, 64
SHLD instruction 65
SHRD instruction 66
SIDT instruction 32, 84
sign extend field 52
SLDT instruction 32, 84
SMSW instruction 33, 84
STC instruction 30, 59
STD instruction 31, 59
STI instruction 31, 59
STOS instruction 24, 69
STR instruction 32, 84
string manipulation instructions 24, 68
SUB instruction 19, 61

T

TEST instruction 23, 67
test register field 53
transcendental instructions 95
two-level paging 8

V

VERR instruction 33, 85
VERW instruction 33, 85
virtual memory 2, 6
virtual 8086 flag 7
virtual 8086 mode 7, 11

W

WAIT instruction 31, 82

X

XCHG instruction 16, 56
XLAT instruction 16
XOR instruction 23, 68

Numerics

80286 microprocessor 1
80287 math coprocessor 2
80386 microprocessor 5
80386 paging mechanism 8
80387 clock generator 12
80387 coprocessor 86
80387 exception conditions 12
80387 I/O ports 12
80387 math coprocessor 10

Notes:

Direct Memory Access Controller (Type 1)

Description 1
DMA Controller Operations 2
  Data Transfers between Memory and I/O Devices 2
  Read Verifications 2
DMA I/O Address Map 3
Byte Pointer 4
DMA Registers 4
  Memory Address Register 4
  I/O Address Register 5
  Transfer Count Register 5
  Temporary Holding Register 5
  Mask Register 6
  Mode Register 7
  Extended Mode Register 7
  Status Register 8
  DMA Extended Function Register (Hex 0018) 9
  Arbus Register 9
DMA Extended Operations 10
  Extended Commands 11

Figures

1. DMA I/O Address Map 3
2. DMA Registers 4
3. Set/Clear Single Mask Bit Using 8237 Compatible Mode . . . 6
4. DMA Mask Register Write Using 8237 Compatible Mode . . . 6
5. 8237 Compatible Mode Register 7
6. Extended Mode Register 8
7. Status Register 8
8. DMA Extended Function Register (Hex 0018) 9
9. Arbus Register 9
10. DMA Extended Address Decode 10
11. DMA Extended Commands 11
12. DMA Channel 2 Programming Example, Extended Commands 12

Description

The Direct Memory Access (DMA) controller allows I/O devices to transfer data directly to and from memory. This frees the system microprocessor of I/O tasks, resulting in a higher throughput.

The DMA controller is software programmable. The system microprocessor can address the DMA controller and read or modify the internal registers to define the various DMA modes, transfer addresses, transfer counts, channel masks, and page registers.

The functions of the DMA controller can be grouped into two categories: program mode and DMA transfer mode.

Program mode is when the system microprocessor accesses the DMA controller within the specific address range. These addresses are identified in Figure 1 on page 3. During program mode, the DMA registers can be read from or written to.

Transfer mode is when the DMA controller performs data transfer. This is initiated when a DMA slave has won the arbitration bus and the DMA controller has been programmed to service the winning request in process. Data transfers can be a single transfer, or multiple transfers (burst).

Deactivation of CD CHRDY by a device can extend accesses for slower I/O or memory devices.

The DMA controller supports the following:

- Register and program compatibility with the IBM Personal Computer AT® DMA channels (8237 compatible mode)
- 16MB (MB equals 1,048,576 bytes) 24-bit address capability for memory and 64KB (KB equals 1024 bytes) 16-bit address capability for I/O
- Eight independent DMA channels capable of transferring data between memory and I/O devices

Personal Computer AT is a registered trademark of the International Business Machines Corporation.

- DMA operation with a separate read and write cycle for each transfer operation
- Channel programmable for byte or word transfer
- Extended operations:
  - Extended program control
  - Extended Mode register
- 8- and 16-bit DMA slaves only
- Programmable arbitration levels for two channels.

DMA Controller Operations

The DMA controller does two types of operations:

- Data transfers between memory and I/O devices
- Read verifications.

Data Transfers between Memory and I/O Devices

The DMA controller performs serial transfers for all read and write operations. These transfers can be between memory and I/O on any channel. Data is read from a device and latched in the DMA controller before it is written back to a second device. The memory address needs to be specified only for a DMA data transfer. A programmable 16-bit I/O address can be provided during the I/O portion of the transfer as a programmable option. If the programmable 16-bit I/O address is not selected, the I/O address is forced to hex 0000 during the I/O transfer.

Read Verifications

The DMA controller can do a memory-read operation without a transfer. The address and the count are updated, and the terminal count is provided.

DMA I/O Address Map

Address		Bit	Byte
(Hex)	Description	Description	Pointer
0000	Channel 0, Memory Address Register	00-15	Used
0001	Channel 0, Transfer Count Register	00-15	Used
0002	Channel 1, Memory Address Register	00-15	Used
0003	Channel 1, Transfer Count Register	00-15	Used
0004	Channel 2, Memory Address Register	00-15	Used
0005	Channel 2, Transfer Count Register	00-15	Used
0006	Channel 3, Memory Address Register	00-15	Used
0007	Channel 3, Transfer Count Register	00-15	Used
0008	Channel 0-3, Status Register	00-07	
000A	Channel 0-3, Mask Register (Set/Reset)	00-02	
000B	Channel 0-3, Mode Register (Write)	00-07	
000C	Clear Byte Pointer (Write)	N/A	
000D	DMA Controller Reset (Write)	N/A	
000E	Channel 0-3, Clear Mask Register (Write)	N/A	
000F	Channel 0-3, Write Mask Register	00-03	
0018	Extended Function Register (Write)	00-07	
001A	Extended Function Execute	00-07	Used *
0081	Channel 2, Page Table Address Register **	00-07	
0082	Channel 3, Page Table Address Register **	00-07	
0083	Channel 1, Page Table Address Register **	00-07	
0087	Channel 0, Page Table Address Register **	00-07	
0089	Channel 6, Page Table Address Register **	00-07	
008A	Channel 7, Page Table Address Register **	00-07	
008B	Channel 5, Page Table Address Register **	00-07	
008F	Channel 4, Page Table Address Register **	00-07	
00C0	Channel 4, Memory Address Register	00-15	Used
00C2	Channel 4, Transfer Count Register	00-15	Used
00C4	Channel 5, Memory Address Register	00-15	Used
00C6	Channel 5, Transfer Count Register	00-15	Used
00C8	Channel 6, Memory Address Register	00-15	Used
00CA	Channel 6, Transfer Count Register	00-15	Used
00CC	Channel 7, Memory Address Register	00-15	Used
00CE	Channel 7, Transfer Count Register	00-15	Used
00D0	Channel 4-7, Status Register	00-07	
00D4	Channel 4-7, Mask Register (Set/Reset)	00-02	
00D6	Channel 4-7, Mode Register (Write)	00-07	
00D8	Clear Byte Pointer (Write)	N/A	
00DA	DMA Controller Reset (Write)	N/A	
00DC	Channel 4-7, Clear Mask Register (Write)	N/A	
00DE	Channel 4-7, Write Mask Register	00-03	

* Used Only During Extended Functions, see "Extended Commands" on page 11.

** Upper Byte of Memory Address Register.

Figure 1. DMA I/O Address Map

Byte Pointer

A byte pointer gives 8-bit ports access to consecutive bytes of registers greater than 8 bits. For program I/O, the registers which use it are the Memory Address registers (3 bytes), the Transfer Count registers (2 bytes), and the I/O Address registers (2 bytes). Interrupts should be masked off when programming DMA controller operations.

DMA Registers

All system microprocessor access to the DMA controller must be 8-bit I/O instructions. The following figure lists the name and size of the DMA registers.

Register	Size (Bits)	Quantity of Registers	Allocation
Memory Address	24	8	1 per Channel
I/O Address	16	8	1 per Channel
Transfer Count	16	8	1 per Channel
Temporary Holding	16	1	All Channels
Mask	4	2	1 for Channels 7 - 4 1 for Channels 3 - 0
Arbus	4	2	1 for Channel 4 1 for Channel 0
Mode	8	8	1 per Channel
Status	8	2	1 for Channel 7 - 4 1 for Channel 3 - 0
Function	8	1	All Channels
Refresh	9	1	Independent of DMA

Figure 2. DMA Registers

Memory Address Register

Each channel has a 24-bit Memory Address register, which is loaded by the system microprocessor. The Mode register determines whether the address is incremented or decremented. The Mode register can be read by the system microprocessor in successive I/O byte operations. To read this register, the microprocessor must use the extended DMA commands.

I/O Address Register

Each channel has a 16-bit I/O Address register, which is loaded by the system microprocessor. The bits in this register do not change during DMA transfers. This register can be read by the system microprocessor in successive I/O byte operations. To read this register, the microprocessor must use the extended DMA commands.

Typically, a DMA slave is selected for DMA transfers by a decode of the arbitration level, status (-S0 exclusively ORed with -S1), and M/-IO. In this case, the respective I/O address register must have a value of zero.

A DMA slave may be selected based on a decode of the address rather the arbitration level. In this case, the respective I/O address register must have the proper I/O address value.

Transfer Count Register

Each channel has a 16-bit Transfer Count register, which is loaded by the system microprocessor. The transfer count determines how many transfers the DMA channel will execute before reaching the terminal count. The number of transfers is always 1 more than the count specifies. If the count is 0, the DMA controller does one transfer. This register can be read by the system microprocessor in successive I/O byte operations. To read this register, the system microprocessor can use only the extended DMA commands.

Temporary Holding Register

This 16-bit register holds the intermediate value for the serial DMA transfer taking place. A DMA operation requires the data to be held in the register before it is written back. This register is not accessible by the system microprocessor.

Mask Register

Bit	Function
7 - 3	Reserved = 0
2	0 Clear Mask Bit 1 Set Mask Bit
1, 0	00 Select Channel 0 or 4 01 Select Channel 1 or 5 10 Select Channel 2 or 6 11 Select Channel 3 or 7

Figure 3. Set/Clear Single Mask Bit Using 8237 Compatible Mode

Bit	Function
7 - 4	Reserved = 0
3	0 Clear Channel 3 or 7 Mask Bit 1 Set Channel 3 or 7 Mask Bit
2	0 Clear Channel 2 or 6 Mask Bit 1 Set Channel 2 or 6 Mask Bit
1	0 Clear Channel 1 or 5 Mask Bit 1 Set Channel 1 or 5 Mask Bit
0	0 Clear Channel 0 or 4 Mask Bit 1 Set Channel 0 or 4 Mask Bit

Figure 4. DMA Mask Register Write Using 8237 Compatible Mode

Each channel has a corresponding mask bit that, when set, disables the DMA from servicing the requesting device. Each mask bit can be set to 0 or 1 by the system microprocessor. A system reset or DMA Controller Reset command sets all mask bits to 1. A Clear Mask Register command sets mask bits 0 - 3 or mask bits 4 - 7 to 0.

When a device requesting DMA cycles wins the arbitration cycle, and the mask bit is set to 1 on the corresponding channel, the DMA controller does not execute any cycles in its behalf and allows external devices to provide the transfer. If no device responds, the bus times out and causes a nonmaskable interrupt (NMI). This register can be programmed using the 8237 compatible mode commands (used by the IBM Personal Computer AT) or the extended DMA commands.

Mode Register

The Mode register for each channel identifies the type of operation that takes place when that channel transfers data.

Bit	Function
7, 6	Reserved = 0
5, 4	Reserved = 0
3, 2	00 Verify Operation 01 Write Operation 10 Read Operation 11 Reserved
1, 0	Channel Accessed 00 Select Channel 0 or 4 01 Select Channel 1 or 5 10 Select Channel 2 or 6 11 Select Channel 3 or 7

Figure 5. 8237 Compatible Mode Register

The Mode register is programmed by the system microprocessor, and its contents are reformatted and stored internally in the DMA controller. In the 8237 compatible mode, this register can only be written.

Extended Mode Register

Besides the 8237 compatible mode, all channels support an 8-bit Extended Mode register. The Extended Mode register can be programmed and read by the system microprocessor.

The DMA controller supports an Extended Mode register for each channel that can be programmed and read by the system microprocessor. This register is used whenever a DMA channel requests a DMA data transfer.

The DMA channel must be programmed to match the transfer size of the DMA slave on the channel. Bit 6 of this register is used to program the size of the DMA transfer.

Bit	**Function**
7	Reserved = 0
6	0 = 8-Bit Transfer 1 = 16-Bit Transfer
5	Reserved = 0
4	Reserved = 0
3	0 = Read Memory Transfer 1 = Write Memory Transfer
2	0 = Read Verifications Operation 1 = Data Transfer Operation
1	Reserved = 0
0	0 = I/O Address equals 0000H 1 = Use programmed I/O Address

Figure 6. Extended Mode Register

Status Register

The Status register, which can be read by the system microprocessor, contains information about the status of the devices. This information tells which channels have reached the terminal count and which channels have requested the bus since the last time the register was read.

Bit	**Function**
7	Channel 3 or 7 Request
6	Channel 2 or 6 Request
5	Channel 1 or 5 Request
4	Channel 0 or 4 Request
3	TC on Channel 3 or 7
2	TC on Channel 2 or 6
1	TC on Channel 1 or 5
0	TC on Channel 0 or 4

Figure 7. Status Register

Bits 3 through 0 in each Status register are set every time a terminal count is reached by a corresponding channel. Bits 7 through 4 are set when a corresponding arbitration level has controlled the bus. All bits are cleared by a system reset or following a system microprocessor Status Read command. This register can be read using the 8237 commands or extended DMA commands.

DMA Extended Function Register (Hex 0018)

This 8-bit register minimizes I/O address requirements and provides the extended program functions. The system microprocessor loads this register using I/O write operations. See "Extended Commands" on page 11 for more information.

Bit	Function
7 - 4	Program Command (DMA Extended Commands)
3	Reserved = 0
2 - 0	Channel Number (0 through 7)

Figure 8. DMA Extended Function Register (Hex 0018)

Arbus Register

This register is used for virtual DMA operations.

Bit	Function
7 - 4	Reserved
3 - 0	Arbitration Level

Figure 9. Arbus Register

Virtual DMA channel operation permits programming of the arbitration level assignment for channels 0 and 4 using the two 4-bit Arbus registers. These registers enable the system microprocessor to dynamically reassign the arbitration ID value by which the DMA controller responds to bus arbitration for DMA requests. This allows channels 0 and 4 to service devices at any arbitration level. The value of arbitration level hex F is reserved.

DMA Extended Operations

The function register supports an extended set of commands for the DMA channels. The extended command hex 8 programs the Arbus registers; the upper 4 bits of the Extended Function register are set to a value of 8 to select the Arbus register, and the lower 4 bits are set to the channel number (0 or 4). If channel 0 = 1 - 3, 5 - 7 then channel 0 is active; if channel 0 = 4 then channel is inactive. The system microprocessor uses the following addresses to gain control of the internal DMA registers.

I/O Address (Hex)	Command
0018	Write Extended Function Register
0019	Reserved
001A	Execute Extended Function Register
001B	Reserved

Figure 10. DMA Extended Address Decode

The system microprocessor uses the following steps to write to or read from any of the DMA internal registers:

1. Write to the Extended Function register by executing an I/O Write instruction to address hex 0018, with the proper data to indicate the function and the channel number. The internal byte pointer is always reset to 0 when an I/O write to address hex 0018 is detected.
2. Execute the Extended Function command by doing an I/O Read or I/O Write instruction to address hex 001A. The byte pointer automatically increments and points to the next byte each time port address hex 001A is used. This step is not required for Direct commands because they are executed when the Out command to address hex 0018 is detected.

Extended Commands

The following figure shows the available extended command set contained in the Extended Function register.

Registers/Bits Accessed	Bits	Extended Command (Hex) (7-4 *)	Byte Pointer
I/O Address Register	00-15	0	Used
Reserved		1	
Memory Address Register Write	00-23	2	Used
Memory Address Register Read	00-23	3	Used
Transfer Count Register Write	00-15	4	Used
Transfer Count Register Read	00-15	5	Used
Status Register Read	00-07	6	
Mode Register	00-07	7	
Arbus Register	00-07	8	
Mask Register Set Single Bit **		9	
Mask Register Reset Single Bit **		A	
Reserved		B	
Reserved		C	
Master Clear **		D	
Reserved		E	
Reserved		F	

* Bits 7-4 of the Extended Function Register.

** Direct commands to the Extended Function register

Figure 11. DMA Extended Commands

The following is an example showing the programming of DMA channel 2 using the 8237 compatible mode and the extended mode. In this example, to perform each step, write the data indicated to the corresponding addresses.

Program Step	**8237 Compatible Mode**	**Extended Mode**
	Address/Data	**Address/Data**
Set Channel Mask Bit	(000AH) x6H	(0018H) 92H
Clear Byte Pointer	(000CH) xxH	(0018H) 22H
Write Memory Address	(0004H) xxH	(001AH) xxH
Write Page Table Address	(0081H) xxH	(001AH) xxH
Clear Byte Pointer	(000CH) xxH	(0018H) 42H
Write Register Count	(0005H) xxH	(001AH) xxH
Write Register Count	(0005H) xxH	(001AH) xxH
Write Mode Register	(000BH) xxH	(0018H) 72H
		(001AH) xxH
Clear Channel 2 Mask Bit	(000AH) x2H	(0018H) A2H

x's represent data.

Figure 12. DMA Channel 2 Programming Example, Extended Commands

Interrupt Controller (Type 1)

Description 1
Interrupt Assignments 1
Interrupt Sharing 2
Interrupt Controller Registers 2
Interrupt Request Register and In-Service Register 3
Interrupt Mask Register 3
Initialization Command Registers and Operation Command Registers 3
Modes of Operation 4
Fully-Nested Mode 4
Special Fully-Nested Mode 5
Automatic Rotation Mode 6
Specific Rotation Mode 6
Special Mask Mode 7
Poll Mode 7
Level-Sensitive Mode 8
Programming the Interrupt Controller 8
Initialization Command Byte 1 9
Initialization Command Byte 2 9
Initialization Command Byte 3 10
Initialization Command Byte 4 10
Operation Command Byte 1 11
Operation Command Byte 2 11
Operation Command Byte 3 12

Figures

1. Interrupt Level Assignments by Priority 1
2. Automatic Rotation Mode 6
3. Specific Rotation Mode when IRQ5 Has the Lowest Priority . 7
4. Poll Mode Status Byte 8
5. Initialization Command Byte 1 9
6. Initialization Command Byte 2 9
7. Initialization Command Byte 3 10
8. Initialization Command Byte 4 10
9. Operation Command Byte 1 11
10. Operation Command Byte 2 11
11. Operation Command Byte 2 (Bits 7 - 5) 12
12. Operation Command Byte 2 (Bits 2 - 0) 12
13. Operation Command Byte 3 12
14. Operation Command Byte 3 (Bits 6 and 5) 13
15. Operation Command Byte 3 (Bits 1 and 0) 13

Description

The system provides 16 levels of hardware interrupts. Any interrupt can be masked, including the nonmaskable interrupt. The interrupt controller must be initialized to the level-sensitive mode; the edge-triggered mode is not supported. Attempts to set the controller to the edge-triggered mode will result in level-sensitive operation. For more information on nonmaskable interrupt, see the system-specific technical references.

Interrupt Assignments

The following figure shows the interrupt assignments, interrupt levels, and their functions. The interrupt levels are listed by order of priority, from highest (NMI) to lowest (IRQ 7). See system-specific technical references for masking interrupts.

Level	Master Function	Level	Slave Function
NMI	Channel Check *		
IRQ 0	Timer		
IRQ 1	Keyboard		
IRQ 2	Cascade Interrupt Control —	IRQ 8	Real Time Clock
		IRQ 9	Redirect Cascade
		IRQ 10	Reserved
		IRQ 11	Reserved
		IRQ 12	Auxiliary Device
		IRQ 13	Math Coprocessor Exception
		IRQ 14	Fixed Disk
		IRQ 15	Reserved
IRQ 3	Serial Alternate		
IRQ 4	Serial Primary		
IRQ 5	Reserved		
IRQ 6	Diskette		
IRQ 7	Parallel Port		

IRQ 8 through 15 are cascaded through IRQ 2

* For Channel Check and other System Specific Functions, Refer to the System-Specific Technical References.

Figure 1. Interrupt Level Assignments by Priority

Interrupt Sharing

Hardware interrupt IRQ9 is defined as the replacement interrupt level for the cascade level IRQ2. Program interrupt sharing should be implemented on IRQ2, interrupt hex 0A. The following processing occurs to maintain compatibility with the IRQ2 used by IBM Personal Computer products:

1. A device drives the interrupt request active on IRQ2 of the channel.
2. This interrupt request is mapped in hardware to IRQ9 input on the slave interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ9 (interrupt hex 71) interrupt handler.
4. The interrupt handler performs an end of interrupt (EOI) to the slave interrupt controller and passes control to the IRQ2 (interrupt hex 0A) interrupt handler.
5. The IRQ2 interrupt handler, when handling the interrupt, causes the device to reset the interrupt request prior to performing an EOI to the master interrupt controller that finishes servicing the IRQ2 request.

Note: Prior to the programming of the interrupt controllers, interrupts should be disabled with a CLI instruction. This includes the Mask register, EOIs, initialization command bytes, and operation command bytes.

Interrupt Controller Registers

The interrupt controller contains the following registers:

- Interrupt Request register
- In-Service register
- Interrupt Mask register
- Initialization Command registers
- Operation Command registers.

Interrupt Request Register and In-Service Register

These registers handle incoming interrupt requests. The Interrupt Request register stores all interrupt levels requesting service. The In-Service register stores all interrupt levels currently being serviced. A priority resolver prioritizes the bits in the Interrupt Request register and strobes the bit with the highest priority into the corresponding bit of the In-Service register.

Both registers can be read by issuing a Read Register command through Operation Command Byte 3, and then reading port hex 0020 or 00A0. The controller keeps track of the last register selected; therefore, subsequent reads of the same register do not require another Operation Command Byte 3 to be written. See "Operation Command Byte 3" on page 12 for more information.

Note: After initialization, the controller is set to read the Interrupt Request register.

Interrupt Mask Register

The Interrupt Mask register contains bits that mask each of the interrupt request lines of the Interrupt Request register. Lower priority levels are not affected when a higher priority level is masked.

The contents of this register are placed on the output data bus when -READ is active and port hex 0021 or 00A1 is accessed.

Initialization Command Registers and Operation Command Registers

These registers store commands from the system microprocessor that define initialization parameters and operating modes. See "Programming the Interrupt Controller" on page 8 for more information.

Modes of Operation

The interrupt controller can be programmed to operate in a variety of modes through the Initialization Command bytes and the Operation Command bytes.

Fully-Nested Mode

In the fully-nested mode, interrupts are prioritized from 0 (highest priority) to 7 (lowest priority). This mode is automatically entered after initialization unless another mode has been defined.

Note: The priorities can be changed by rotating the priorities through Operation Command Byte 2.

A typical interrupt request occurs in the following manner:

1. One or more 'interrupt request' lines are set active causing the corresponding bits in the Interrupt Request register to be set to 1.
2. The interrupt controller evaluates the requests and sends an interrupt to the system microprocessor, if appropriate.
3. The system microprocessor responds with an 'interrupt acknowledge' pulse to the interrupt controller.
4. The controller prioritizes the unmasked bits in the Interrupt Request register and strobes the bit with the highest priority into the corresponding bit of the In-Service register. No data is sent to the system microprocessor.

   Note: If an interrupt request is not present (for example, the duration of the request was too short), the interrupt controller issues an interrupt 7.
5. The system microprocessor sends a second 'interrupt acknowledge' pulse to the interrupt controller.
6. The interrupt controller responds by releasing the interrupt vector on the data bus, where it is read by the system microprocessor.
7. The highest priority in-service bit remains set to 1 until the proper End of Interrupt command is issued by the interrupt subroutine. If the source of the interrupt request is the slave interrupt controller, the End of Interrupt command must be issued twice, once for the master and once for the slave. When the in-service

bit is set to 1, all other interrupts with the same or lower priority are inhibited; interrupts with a higher priority cause an interrupt, but the interrupt is acknowledged only if the system-microprocessor interrupt input has been re-enabled by software.

The End of Interrupt command has two forms, specific and nonspecific. The controller responds to a nonspecific End of Interrupt command by resetting the highest in-service bit of those set. In a mode that uses a fully-nested interrupt structure, the highest in-service bit set is the level that was just acknowledged and serviced. In a mode that can use other than the fully-nested interrupt structure, a specific End of Interrupt command is required to define which in-service bit to reset.

Note: An in-service bit masked by an Interrupt Mask register bit cannot be reset by a nonspecific End of Interrupt command when in the special mask mode. See "Special Mask Mode" on page 7 for more information.

Special Fully-Nested Mode

The special fully-nested mode is used when the priority in the slave interrupt controller must be preserved. This mode is similar to the normal nested mode with the following exceptions:

- When the slave's interrupt request is in service, the slave can still generate additional interrupt requests of a higher priority that are recognized by the master, and initiate interrupts to the system microprocessor.

- Upon completion of the interrupt service routine, software must send a nonspecific End of Interrupt command to the slave and read the slave's In-Service register to ensure that the interrupt just serviced was the only one generated by the slave. If the register is not empty, additional interrupts are pending, and an End of Interrupt command must not be sent to the master. If the register is empty, a nonspecific End of Interrupt command can be sent to the master.

The special fully-nested mode is selected through Initialization Command Byte 4. See "Initialization Command Byte 4" on page 10 for more information.

Automatic Rotation Mode

The automatic rotation mode accommodates multiple devices having the same interrupt priority. After a device is serviced, it is assigned the lowest priority and must wait until all other devices requesting an interrupt are serviced once before the first device is serviced again.

The following example shows the status and priorities of the In-Service register bits before and after bit 4 of the Interrupt-Request register is serviced by a Rotation on Nonspecific End of Interrupt command.

In-Service Register Bits	**Status before Service**	**Priority before Rotate**	**Status after Service**	**Priority after Rotate**
7	0	7 (Lowest)	0	2
6	1 (Pending)	6	1 (Pending)	1
5	0	5	0	0 (Highest)
4	1 (Pending)	4	0 (Serviced)	7 (Lowest)
3	0	3	0	6
2	0	2	0	5
1	0	1	0	4
0	0	0 (Highest)	0	3

Figure 2. Automatic Rotation Mode

The automatic rotation mode is selected by issuing a Rotation on Nonspecific End of Interrupt command through Operation Command Byte 2. See “Operation Command Byte 2” on page 11 for more information.

Specific Rotation Mode

The specific rotation mode allows the application programs to change the priority levels by assigning the lowest priority to a specific interrupt level. Once the lowest-level priority is selected, all other priority levels change. The following example compares the normal-nested mode to the specific rotation mode with bit 5 of the Interrupt Request register set to the lowest priority.

Interrupt Request Register Bits	Nested Mode Priority Level	Specific Rotation Mode Priority Level
7	7 (Lowest)	1
6	6	0 (Highest)
5	5	7 (Lowest)
4	4	6
3	3	5
2	2	4
1	1	3
0	0 (Highest)	2

Figure 3. Specific Rotation Mode when IRQ5 Has the Lowest Priority

The specific rotation mode is selected by issuing a Rotate on Specific End of Interrupt command or a Set Priority command through Operation Command Byte 2. See "Operation Command Byte 2" on page 11 for more information.

Special Mask Mode

The special mask mode allows application programs to selectively enable and disable any interrupt or combination of interrupts at any time during its execution. The special mask mode is selected through Operation Command Byte 3. Once the controller is in the special mask mode, setting a bit in Operation Command Byte 1 sets a corresponding bit in the Interrupt Mask register. Each bit set in the Interrupt Mask register masks the corresponding interrupt channel. Interrupt channels above and below a masked channel are not affected. See "Operation Command Byte 1" on page 11 and "Operation Command Byte 3" on page 12 for more information.

Poll Mode

Before the poll mode can be used, a CLI instruction must be issued to disable the system-microprocessor interrupt input. Devices are serviced by software issuing a Poll command through Operation Command Byte 3. The first 'read' pulse following a Poll command is interpreted by the controller as an 'interrupt acknowledge' pulse; the controller sets the appropriate in-service bit and reads the priority level. The byte placed on the data bus during a 'read' pulse is shown in the following figure.

Bit	Function
7	Interrupt Present
6-3	Undefined
2-0	Highest Priority Level

Figure 4. Poll Mode Status Byte

Bit 7 This bit is set to 1 if an interrupt is present.

Bits 6 - 3 These bits are not used and may be set to either 0 or 1.

Bits 2 - 0 These bits contain the binary code of the highest priority level requesting service.

Level-Sensitive Mode

The interrupt controller cannot be placed in the edge-triggered mode. In the level-sensitive mode, interrupt requests are recognized by a high level on the interrupt-request input. Interrupt requests must be removed before the End of Interrupt command is issued to prevent a second interrupt from occurring.

Programming the Interrupt Controller

Before the system can be used, the interrupt controller must be programmed with four sequential initialization commands. When a command is issued to a master at port hex 0020 (or a slave at port hex 00A0) with bit 4 set to 1, the command is recognized as Initialization Command Byte 1. Initialization Command Byte 1 is the first of four initialization commands required to program the interrupt controller. The following events occur during the initialization sequence:

1. The level sense circuit is set to the level-sensitive mode. (Following the initialization procedure, interrupts are generated by a high level on the interrupt-request input.)
2. The Interrupt Mask register is cleared.
3. IRQ7 is assigned priority 7.
4. The slave mode address is set to 7.

5. The special mask mode is cleared and the controller is set to read the Interrupt Request register.

Once the interrupt controller is programmed by the initialization command bytes, the controller can be programmed by the operation command bytes to operate in other modes.

Note: The master interrupt controller must be initialized before the slave interrupt controller. Failure to do so will cause unexpected results.

Initialization Command Byte 1

This is the first byte of the 4-byte initialization command sequence. This byte is issued to either the master (port hex 0020) or the slave (port hex 00A0).

Bit	Function
7 - 5	Reserved – Must be set to 0.
4	Initialization Command Byte 1 Identifier – Must be set to 1.
3	Level-Sensitive Mode – Must be set to 1.
2	Call Address Interval of 8 – Must be set to 0.
1	Cascade Mode – Must be set to 0.
0	4-byte Initialization Command Sequence – Must be set to 1.

Figure 5. Initialization Command Byte 1

Initialization Command Byte 2

This byte defines the address of the interrupt vector. Bits 7 through 3 define the five high-order bits of the interrupt vector address. Bits 2 through 0 are initialized to 0 and replaced by the hardware interrupt level when an interrupt occurs. This byte is issued to either the master (port hex 0021) or the slave (port hex 00A1).

Bit	Function
7 - 3	Bits 7 - 3 of the Interrupt Vector Address
2 - 0	Initialized to 0

Figure 6. Initialization Command Byte 2

Initialization Command Byte 3

This byte loads a value into an 8-bit slave register.

- In the master device mode, this byte has a value of hex 04 to identify interrupt 02 as a slave providing the input request.
- In the slave device mode, this byte has a value of hex 02 to tell the slave that it is using hardware interrupt 02 to communicate with the master. The slave compares this value to the cascade input; if they are equal, the slave releases the Interrupt Vector Address on the data bus.

This byte is issued to either the master (port hex 0021) or the slave (port hex 00A1).

Bit	Slave Function	Master Function
7 - 3	0	0
2	1	0
1	0	1
0	0	0

Figure 7. Initialization Command Byte 3

Initialization Command Byte 4

This byte is issued to either the master (port hex 0021) or the slave (port hex 00A1).

Bit	Function
7 - 5	Reserved – Must be set to 0.
4	Special Fully-Nested Mode
3, 2	Reserved – Must be set to 0.
1	Normal End of Interrupt – Must be set to 0.
0	80286/80386 Microprocessor Mode – Must be set to 1.

Figure 8. Initialization Command Byte 4

Operation Command Byte 1

This byte controls the individual bits in the Interrupt Mask register.

This byte is issued to either the master (port hex 0021) or the slave (port hex 00A1).

Bit	Function
7 - 0	Interrupt Mask Bits 7 - 0

Figure 9. Operation Command Byte 1

Bits 7 - 0 When set to 1, these bits inhibit their respective interrupt request input signals.

Operation Command Byte 2

This byte controls the interrupt priority and End of Interrupt command.

This byte is issued to either a master (port hex 0020) or a slave (port hex 00A0).

Bit	Function
7	Rotate Mode
6	Set Interrupt Level
5	End of Interrupt Mode
4, 3	Reserved – Must be set to 0
2 - 0	Interrupt Level (When bit 6 = 1)

Figure 10. Operation Command Byte 2

Bits 7 - 5 These bits define the rotate mode, end of interrupt mode, or a combination of the two, as shown in Figure 11 on page 12.

Bit 7	Bit 6	Bit 5	Function
0	0	0	Reserved
0	0	1	Nonspecific End of Interrupt Command
0	1	0	No Operation
0	1	1	Specific End of Interrupt Command*
1	0	0	Reserved
1	0	1	Rotate on Nonspecific End of Interrupt Command
1	1	0	Set Priority Command**
1	1	1	Rotate on Specific End of Interrupt Command**

* Bits 0, 1, and 2 are the binary level of the in-service bit to be reset.

** Bits 0, 1, and 2 are the binary level of the lowest priority device.

Figure 11. Operation Command Byte 2 (Bits 7 - 5)

Bits 4, 3 These bits are reserved and must be set to 0.

Bits 2 - 0 These bits define the hardware interrupt level to be acted upon when bit 6 is set to 1.

Bit 2	Bit 1	Bit 0	Function
0	0	0	Interrupt Level 0
0	0	1	Interrupt Level 1
0	1	0	Interrupt Level 2
0	1	1	Interrupt Level 3
1	0	0	Interrupt Level 4
1	0	1	Interrupt Level 5
1	1	0	Interrupt Level 6
1	1	1	Interrupt Level 7

Figure 12. Operation Command Byte 2 (Bits 2 - 0)

Operation Command Byte 3

This byte is issued to either a master (port hex 0020) or a slave (port hex 00A0).

Bit	Function
7	Reserved – Must be set to 0.
6, 5	Special Mask Mode Bits
4	Reserved – Must be set to 0.

Figure 13 (Part 1 of 2). Operation Command Byte 3

Bit	Function
3	Reserved – Must be set to 1.
2	Poll Command
1, 0	Read Register Command

Figure 13 (Part 2 of 2). Operation Command Byte 3

Bit 7 This bit is reserved and must be set to 0.

Bits 6, 5 These bits enable the special mask mode, as shown in the following figure.

Bit 6	Bit 5	Function
0	0	No Action
0	1	No Action
1	0	Normal Mask Mode
1	1	Special Mask Mode

Figure 14. Operation Command Byte 3 (Bits 6 and 5)

Bit 4 This bit is reserved and must be set to 0.

Bit 3 This bit is reserved and must be set to 1.

Bit 2 When set to 1, this bit sets the Poll command.

Bits 1, 0 These bits determine the register to be read on the next 'read' pulse, as shown in the following figure.

Bit 1	Bit 0	Function
0	0	No Action
0	1	No Action
1	0	Read Interrupt Request Register
1	1	Read In-Service Register

Figure 15. Operation Command Byte 3 (Bits 1 and 0)

Notes:

System Timers (Type 1)

Description 1
Channel 0 - System Timer 2
Channel 2 - Tone Generation for Speaker 2
Channel 3 - Watchdog Timer 4
Counters 0, 2, and 3 5
Programming the System Timers 5
Counter Write Operations 5
Counter Read Operations 6
Registers 6
Count Register - Channel 0 (Hex 0040) 6
Count Register - Channel 2 (Hex 0042) 6
Control Byte Register - Channel 0 or 2 (Hex 0043) 7
Count Register - Channel 3 (Hex 0044) 9
Control Byte Register - Channel 3 (Hex 0047) 9
Counter Latch Command 10
System Timer Modes 11
Mode 0 - Interrupt on Terminal Count 11
Mode 1 - Hardware Retriggerable One-Shot 12
Mode 2 - Rate Generator 13
Mode 3 - Square Wave 13
Mode 4 - Software Retriggerable Strobe 15
Mode 5 - Hardware Retriggerable Strobe 16
Operations Common to All Modes 17

Figures

1. Counters 1
2. Audio Subsystem Block Diagram 3
3. System Timer/Counter Registers 6
4. Select Counter Bits, Port Hex 0043 7
5. Read/Write Counter Bits, Port Hex 0043 7
6. Counter Mode Bits, Port Hex 0043 8
7. Select Counter, Port Hex 0047 9
8. Read/Write Counter, Port Hex 0047 9
9. Counter Latch Command 10
10. Minimum and Maximum Initial Counts, Counters 0, 2 17

Description

The system has three programmable timers/counters: Channel 0 is the System Timer, Channel 2 is the Tone Generator for the speaker, and Channel 3 is the Watchdog Timer. Channel 0 and Channel 2 are similar to Channel 0 and Channel 2 of the IBM Personal Computer, IBM Personal Computer XT™, and the IBM Personal Computer AT®. Channel 3 does not have a counterpart in earlier IBM personal computer systems. The following is a block diagram of the counters.

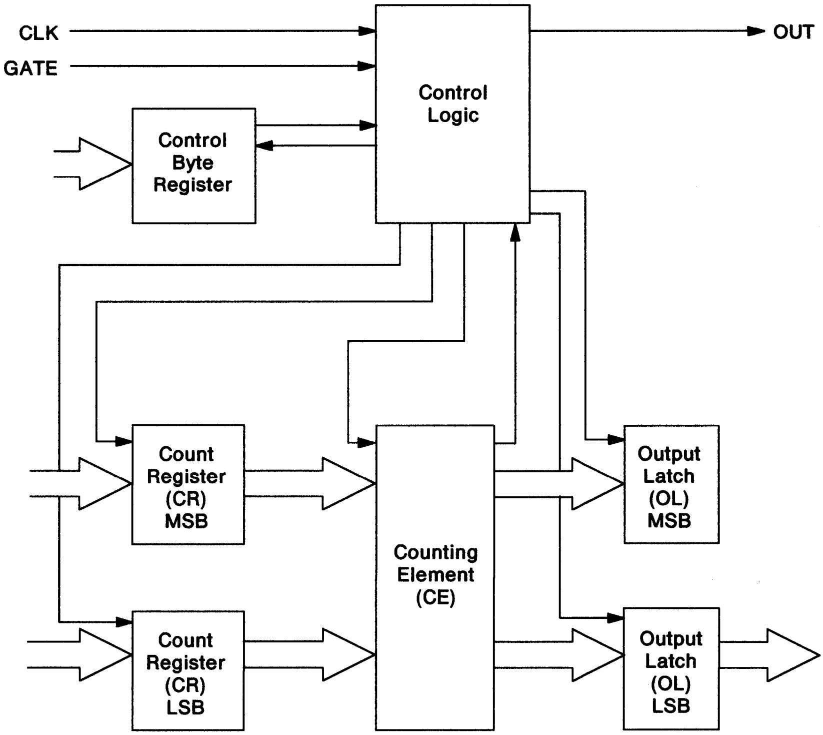


Figure 1. Counters

Personal Computer XT is a trademark of the International Business Machines Corporation.

Personal Computer AT is a registered trademark of the International Business Machines Corporation.

Channel 0 - System Timer

- GATE 0 is always enabled.
- CLK IN 0 is driven by a 1.193 MHz signal.
- CLK OUT 0 indirectly drives 'interrupt request 0' signal (IRQ 0).

  IRQ 0 is driven by a latch that is set by the rising edge of the 'clock out 0' signal (CLK OUT 0). The latch may be cleared by a system reset, an interrupt acknowledge cycle with a vector of hex 08, or an I/O write to System Control Port B (hex 0061) setting bit 7 to 1.

  Signals derived from CLK OUT 0 are used to gate and clock Channel 3.

Channel 2 - Tone Generation for Speaker

- GATE 2 is controlled by bit 0 of port hex 0061.
- CLK IN 2 is driven by a 1.193 MHz signal.
- CLK OUT 2 has two connections. One is to input port hex 0061, bit 5. CLK OUT 2 is also logically ANDed with port hex 0061, bit 1 (speaker data enable). The output of the AND gate drives the 'audio sum node' signal.

The audio subsystem is a speaker driven by a linear amplifier. The linear amplifier input node can be driven from the following sources:

- System-timer Channel 2 can be enabled to drive the speaker using bit 1 of I/O port hex 0061 set to 1. For information about system timer Channel 2 see "Description" on page 1.
- The channel using the 'audio sum node' signal.

The following block diagram shows the audio subsystem.

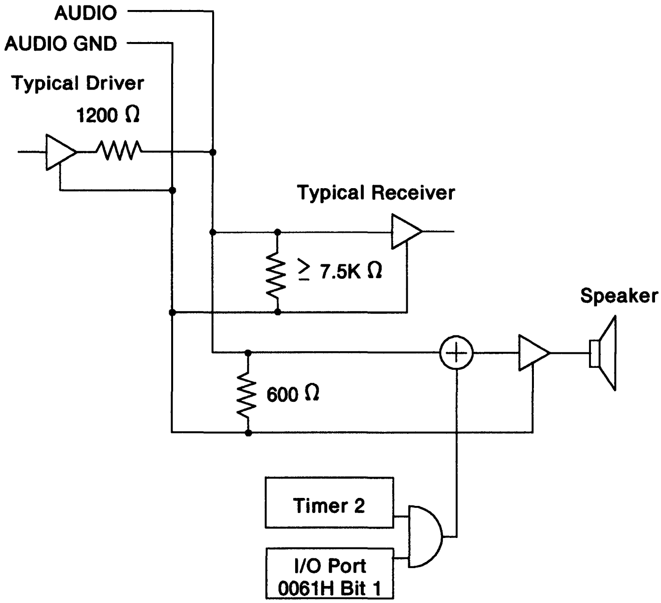


Figure 2. Audio Subsystem Block Diagram

Each audio driver must have a 1200 ohm source impedance, and a 7.5 kilohm or greater impedance is required for each audio receiver. Volume control is provided by the driver. Output level is a function of the number of drivers and receivers that share the AUDIO line.

Logic ground is connected to AUDIO GND at the amplifier.

Channel 3 - Watchdog Timer

This channel operates only in Mode 0, and counts in 8-bit binary.

- GATE 3 is tied to IRQ 0.
- CLK IN 3 is tied to CLK OUT 0 inverted.
- CLK OUT 3, when high, drives the NMI active.

The Watchdog Timer detects when IRQ 0 is active for more than one period of CLK OUT 0. If IRQ 0 is active when a rising edge of CLK OUT 0 occurs, the count is decremented. When the count is decremented to 0, an NMI is generated. Thus, the Watchdog Timer can be used to detect when IRQ 0 is not being serviced. This is useful for detecting error conditions.

BIOS interfaces are provided to enable and disable the Watchdog Timer. When the Watchdog Timer times out, it causes an NMI and sets System Control Port A (hex 0092), bit 4 to 1. This bit may be set to 0 by using the BIOS interface to disable the Watchdog Timer.

Note: The NMI stops all arbitration on the bus until bit 6 of the Arbitration register (I/O address hex 0090) is set to 0. This can result in lost data or an overrun error on some I/O devices.

If the Watchdog Timer is used to detect “tight looping” software tasks that inhibit interrupts, some I/O devices may be overrun (not serviced in time). The operating system may be required to restart these devices.

When the Watchdog Timer is enabled, the ‘inhibit’ signal (INHIBIT) is active only when IRQ 0 is pending for longer than one period of CLK OUT 0. When INHIBIT is active, any data written to Channel 0 or Channel 3 is ignored. INHIBIT is never active if the Watchdog Timer is disabled.

The Watchdog Timer operation is defined only when Channel 0 is programmed in Mode 2 or Mode 3. The operation of the Watchdog Timer is undefined when Channel 0 is programmed in any other mode.

Counters 0, 2, and 3

Each counter is independent. Counters 0 and 2 are 16-bit down counters that can be preset. They can count in binary or binary coded decimal (BCD). Counter 3 is an 8-bit down counter that can be preset. It counts in binary only.

Programming the System Timers

The system treats the programmable interval timer as an arrangement of five external I/O ports. Three ports are treated as count registers and two are control registers for mode programming. Counters are programmed by writing a control word and then an initial count. All control words are written into the Control Word registers, which are located at address hex 0043 for counters 0 and 2, and address hex 0047 for counter 3. Initial counts are written into the Count registers, not the Control Byte registers. The format of the initial count is determined by the control word used.

After the initial count is written to the Count register, it is transferred to the counting element, according to the mode definition. When the count is read, the data is presented by the output latch.

Counter Write Operations

The control word must be written before the initial count, and the count must follow the count format specified in the control word.

A new initial count may be written to the counters at any time without affecting the counter's programmed mode. Counting is affected as described in the mode definitions. The new count must follow the programmed count format.

Counter Read Operations

The counters can be read using the Counter Latch command (see "Counter Latch Command" on page 10).

If the counter is programmed for two-byte counts, two bytes must be read. The two bytes need not be read consecutively; read, write, or programming operations of other counters may be inserted between them.

Note: If the counters are programmed to read or write two-byte counts, the program must not transfer control between writing the first and second byte to another routine that also reads or writes into the same counter. This will cause an incorrect count.

Registers

I/O Address (Hex)	Register
0040	Count Register - Channel 0 (Read/Write)
0042	Count Register - Channel 2 (Read/Write)
0043	Control Byte Register - Channel 0 or 3 (Write)
0044	Count Register - Channel 3 (Read/Write)
0047	Control Byte Register - Channel 3 (Write)

Figure 3. System Timer/Counter Registers

Count Register - Channel 0 (Hex 0040)

The control byte is written to port hex 0043 indicating the format of the count (least-significant byte only, most-significant byte only, or least-significant byte followed by most-significant byte). This must be done before writing the count to port hex 0040.

Count Register - Channel 2 (Hex 0042)

The control byte is written to port hex 0043 indicating the format of the count (least-significant byte only, most-significant byte only, or least-significant byte followed by most-significant byte). This must be done before writing the count to port hex 0042.

Control Byte Register - Channel 0 or 2 (Hex 0043)

This is a write-only register. The following gives the format for the control byte (port hex 0043) for counters 0 and 2.

Bits 7, 6 These bits select Counter 0 or 2.

Bits 7 6	Function
0 0	Select Counter 0
0 1	Reserved
1 0	Select Counter 2
1 1	Reserved

Figure 4. Select Counter Bits, Port Hex 0043

Bits 5, 4 These bits distinguish a counter latch command from a control byte. If a control byte is selected, these bits also determine the method in which each byte is read or written.

Bits 5 4	Function
0 0	Counter Latch Command
0 1	Read/Write Counter bits 0 - 7 only
1 0	Read/Write Counter bits 8 - 15 only
1 1	Read/Write Counter bits 0 - 7 first, then bits 8 - 15

Figure 5. Read/Write Counter Bits, Port Hex 0043

Bits 3 - 1 These bits select the mode.

Bits 3 2 1	Function
0 0 0	Mode 0 - Interrupt on Terminal Count
0 0 1	Mode 1 - Hardware Retriggerable One Shot
X 1 0	Mode 2 - Rate Generator
X 1 1	Mode 3 - Square Wave
1 0 0	Mode 4 - Software Retriggerable Strobe
1 0 1	Mode 5 - Hardware Retriggerable Strobe

Don't care bits (X) should be set to 0.

Figure 6. Counter Mode Bits, Port Hex 0043

Bit 0 When set to 1, this bit selects the binary coded decimal method of counting. When set to 0, it selects the 16-bit binary method.

Count Register - Channel 3 (Hex 0044)

The control byte is written to port hex 0047 indicating the format of the count (least-significant byte only). This must be done before writing the count to port hex 0044.

Control Byte Register - Channel 3 (Hex 0047)

This is a write-only register. The following gives the format for the control byte (port hex 0047) for counter 3.

Bits 7, 6 These bits select Counter 3.

Bits 7 6	Function
0 0	Select Counter 3
0 1	Reserved
1 0	Reserved
1 1	Reserved

Figure 7. Select Counter, Port Hex 0047

Bits 5, 4 These bits distinguish a counter latch command from a control byte.

Bits 5 4	Function
0 0	Counter Latch Command Select Counter 0
0 1	R/W Counter Bits 0 - 7 Only
1 0	Reserved
1 1	Reserved

Figure 8. Read/Write Counter, Port Hex 0047

Bits 3 - 0 These bits are reserved and must be written as 0.

Counter Latch Command

The Counter Latch command is written to the Control Byte register. Bits 7 and 6 select the counter, and bits 5 and 4 distinguish this command from a control byte. The following figure shows the format of the Counter Latch command.

Bit	**Function**
7, 6	Specifies the counter to be latched
5, 4	00 Specifies the Counter Latch command
3 - 0	Reserved = 0

Figure 9. Counter Latch Command

The count is latched into the selected counter's output latch when the Counter Latch command is received. This count is held in the latch until it is read by the system microprocessor (or until the counter is reprogrammed). After the count is read by the system microprocessor, it is automatically unlatched, and the output latch returns to following the counting element. Counter Latch commands do not affect the programmed mode of the counter in any way. All subsequent latch commands issued to a given counter before the count is read, are ignored. A read cycle to the counter latch returns the value latched by the first Counter Latch command.

System Timer Modes

The following definitions are used when describing the timer modes.

CLK pulse A rising edge, then a falling edge on the counter CLK input.

Trigger A rising edge on a counter's input GATE.

Counter Load The transfer of a count from the Counter register to the counting element.

Mode 0 - Interrupt on Terminal Count

Event counting can be done using Mode 0. Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0. If GATE is equal to 1 when the control byte and initial count are written to the counter, the sequence is as follows:

1. The control byte is written to the counter, and OUT goes low.
2. The initial count is written.
3. Initial count is loaded on the next CLK pulse. The count is not decremented for this CLK pulse.

   The count is decremented until the counter reaches 0. For an initial count of N, the counter reaches 0 after N+1 CLK pulses.
4. OUT goes high.

OUT remains high until a new count or new Mode 0 control byte is written into the counter.

If GATE equals 0 when an initial count is written to the counter, it is loaded on the next CLK pulse even though counting is not enabled. After GATE enables counting, OUT goes high N CLK pulses later.

If a new count is written to a counter while counting, it is loaded on the next CLK pulse. Counting then continues from the new count. If a 2-byte count is written to the counter the following occurs:

1. The first byte written to the counter disables the counting. OUT goes low immediately, and there is no delay for the CLK pulse.
2. When the second byte is written to the counter, the new count is loaded on the next CLK pulse. OUT goes high when the counter reaches 0.

Mode 1 - Hardware Retriggerable One-Shot

The sequence for Mode 1 is as follows:

1. OUT is high.
2. On the CLK pulse following a trigger, OUT goes low and begins the one-shot pulse.
3. When the counter reaches zero, OUT goes high.

OUT remains high until the CLK pulse following the next trigger.

The counter is armed by writing the control byte and initial count to the counter. When a trigger occurs, the counter is loaded. OUT goes low on the next CLK pulse, starting the one-shot pulse. For an initial count of N, a one-shot pulse is N CLK pulses long. The one-shot pulse repeats the count of N for the next triggers. OUT remains low N CLK pulses following any trigger. GATE does not affect OUT. The current one-shot pulse is not affected by a new count written to the counter, unless the counter is retriggered. If the counter is retriggered, the new count is loaded and the one-shot pulse continues.

Note: Mode 1 is valid only for Counter 2.

Mode 2 - Rate Generator

This mode causes the counter to perform a divide-by-N function. Counting is enabled when GATE equals 1, and disabled when GATE equals 0.

The sequence for Mode 2 is as follows:

1. OUT is high.
2. The initial count decrements to 1.
3. OUT goes low for one CLK pulse.
4. OUT goes high.
5. The counter reloads the initial count.
6. The process is repeated.

If GATE goes low during the OUT pulse, OUT goes high. On the next CLK pulse a trigger reloads the counter with the initial count. OUT goes low N CLK pulses after the trigger. This allows the GATE input to be used to synchronize the counter.

The counter is loaded on the CLK pulse after a control byte and initial count are written to the counter. OUT goes low N CLK pulses after writing the initial count. This allows software to synchronize the counter.

The current counting sequence is not affected by a new count being written to the counter. If the counter receives a trigger after a new count is written and before the end of the current count, the new count is loaded on the next CLK pulse, and counting continues from the new count. If the trigger is not received by the counter, the new count is loaded following the current counting cycle.

Mode 3 - Square Wave

Mode 3 is similar to Mode 2 except for the duty cycle of OUT. Counting is enabled when GATE is equal to 1, and disabled when GATE is equal to 0. An initial count of N results in a square wave on OUT. The period of the square wave is N CLK pulses. If OUT is low and GATE goes low, OUT goes high. On the next CLK pulse, a trigger reloads the counter with the initial count.

The counter is loaded on the CLK pulse following the writing of a control byte and the initial count.

The current counting sequence is not affected by a new count being written to the counter. If the counter receives a trigger after a new count is written, and before the end of the current count's half-cycle of the square wave, the new count is loaded on the next CLK pulse, and counting continues from the new count. If the trigger is not received by the counter, the new count is loaded following the current half-cycle.

The way Mode 3 is implemented depends on whether the count written is odd or even. If the count is even, OUT begins high and the following applies:

1. The initial count is loaded on the first CLK pulse.
2. The count is decremented by 2 on succeeding CLK pulses.
3. The count decrements to 0.
4. OUT changes state.
5. The counter is reloaded with the initial count.
6. The process repeats indefinitely.

If the count is odd, the following applies:

1. OUT is high.
2. The initial count minus 1 is loaded on the first CLK pulse.
3. The count is decremented by 2 on succeeding CLK pulses.
4. The count decrements to 0.
5. One CLK pulse after the count reaches 0, OUT goes low.
6. The counter is reloaded with the initial count minus 1.
7. Succeeding CLK pulses decrements the count by 2.
8. The count decrements to 0.
9. OUT goes high.
10. The counter is reloaded with the initial count minus 1.
11. The process repeats indefinitely.

Mode 3, using an odd count, causes OUT to go high for a count of (N+1)/2 and low for a count of (N-1)/2.

Mode 3 may operate such that OUT is initially set low when the control byte is written. For this condition, Mode 3 operates as follows:

1. OUT is low.
2. The count decrements to half of the initial count.
3. OUT goes high.
4. The count decrements to 0.
5. OUT goes low.
6. The process repeats indefinitely.

This process results in a square wave with a period of N CLK pulses.

Note: If OUT needs to be high after the control byte is written, the control byte must be written twice. This applies only to Mode 3.

Mode 4 - Software Retriggerable Strobe

Counting is enabled when GATE equals 1, and disabled when GATE equals 0. Counting begins when an initial count is written.

The sequence for Mode 4 is as follows:

1. OUT is high.
2. The control byte and initial count are written to the counter.
3. The initial count is loaded on the next CLK pulse. The count is not decremented for this clock pulse.
4. The count is decremented to 0. For an initial count of N, the counter reaches 0 after N+1 CLK pulses.
5. OUT goes low for one CLK pulse.
6. OUT goes high.

GATE should not go low one-half CLK pulse before or after OUT goes low. If this occurs, OUT remains low until GATE goes high.

If a new count is written to a counter while counting, it is loaded on the next CLK pulse. Counting then continues from the new count. If a 2-byte count is written, the following occurs:

1. Writing the first byte does not affect counting.
2. The new count is loaded on the CLK pulse following the writing of the second byte.

The Mode 4 sequence can be retriggered by software. The period from when the new count of N is written to when OUT strobes low is (N+1) pulses.

Mode 5 - Hardware Retriggerable Strobe

The sequence for Mode 5 is as follows:

1. OUT is high.
2. The control byte and initial count are written to the counter.
3. Counting is triggered by a rising edge of GATE.
4. The counter is loaded on the CLK pulse following the trigger. This CLK pulse does not decrement the count.
5. The count decrement to 0.
6. OUT goes low for one CLK pulse. This occurs (N+1) CLK pulses after the trigger.
7. OUT goes high.

The counting sequence can be retriggered. OUT strobes low (N+1) pulses after the trigger. GATE does not affect OUT.

The current counting sequence is not affected by a new count being written to the counter. If the counter receives a trigger after a new count is written and before the end of the current count, the new count is loaded on the next CLK pulse, and counting continues from the new count.

Note: Mode 5 is valid only on Counter 2.

Operations Common to All Modes

Control bytes written to a counter cause all control logic to reset. OUT goes to a known state. This does not take a CLK pulse.

The falling edge of the CLK pulse occurs when new counts are loaded and counters are decremented.

Counters do not stop when they reach zero. In Modes 0, 1, 4, and 5 the counter wraps to the highest count, and continues counting. Modes 2 and 3 are periodic; the counter reloads itself with the initial count and continues from there.

The GATE is sampled on the rising edge of the CLK pulse.

The following shows the minimum and maximum initial counts for the counters.

Mode	Minimum Count	Maximum Count
0	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
1	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
2	2	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
3	2	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
4	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)
5	1	0 = 2^{16} (Binary Counting) or 10^4 (BCD Counting)

Figure 10. Minimum and Maximum Initial Counts, Counters 0, 2

Counter 3 can use only Mode 0, Interrupt on Terminal Count. The minimum initial count is 1 and the maximum is hex FF.

Notes:

Diskette Drive Controller (Type 1)

Description 1
Diskette Drive Controller Registers 2
  Status Register A (Hex 03F0) 2
  Status Register B (Hex 03F1) 3
  Digital Output Register (Hex 03F2) 3
  Digital Input Register (Hex 03F7 - Read) 4
  Configuration Control Register (Hex 03F7 - Write) 4
  Diskette Drive Controller Status Register (Hex 03F4) 5
  Data Register (Hex 03F5) 5
Diskette Drive Controller Programming Considerations 6
  Controller Commands 7
    Read Data Command 9
    Read Deleted Data Command 10
    Read a Track Command 11
    Read ID Command 12
    Write Data Command 13
    Write Deleted Data Command 14
    Format a Track Command 15
    Scan Equal Command 16
    Scan Low or Equal Command 17
    Scan High or Equal Command 18
    Recalibrate Command 19
    Sense Interrupt Status Command 19
    Specify Command 20
    Sense Drive Status Command 20
    Seek Command 21
    Invalid Command Status 21
  Command Status Registers 22
    Status Register 0 22
    Status Register 1 23
    Status Register 2 24
    Status Register 3 25
Signal Descriptions 26
  Output Signals 26
  Input Signals 28
  Power Sequencing 28
Connector 29

Index 30

Figures

1. Status Register A (Hex 03F0) 2
2. Status Register B (Hex 03F1) 3
3. Digital Output Register (Hex 03F2) 3
4. Digital Input Register (Hex 03F7 - Read) 4
5. Configuration Control Register (Hex 03F7 - Write) 4
6. Data Rate Selection 4
7. Diskette Drive Controller Status Register (Hex 03F4) 5
8. Command Symbols, Diskette Drive Controller 8
9. Read Data Command 9
10. Read Data Result 9
11. Read Deleted Data Command 10
12. Read Deleted Data Result 10
13. Read a Track Command 11
14. Read a Track Result 11
15. Read ID Command 12
16. Read ID Result 12
17. Write Data Command 13
18. Write Data Result 13
19. Write Deleted Data Command 14
20. Write Deleted Data Result 14
21. Format a Track Command 15
22. Format a Track Result 15
23. Scan Equal Command 16
24. Scan Equal Result 16
25. Scan Low or Equal Command 17
26. Scan Low or Equal Result 17
27. Scan High or Equal Command 18
28. Scan High or Equal Result 18
29. Recalibrate Command 19
30. Sense Interrupt Status Command 19
31. Sense Interrupt Status Result 19
32. Specify Command 20
33. Sense Drive Status Command 20
34. Sense Drive Status Result 20
35. Seek Command 21
36. Invalid Command Result 21
37. Status Register 0 22
38. Status Register 0 (Bits 7, 6) 22
39. Status Register 1 23

40. Status Register 2 24
41. Status Register 3 25
42. Diskette Drive Connector Voltage and Signal Assignments 29

Notes:

Description

The diskette drive controller and connector reside on the system board. The diskette drive controller supports:

- Two drives
- 1MB unformatted media, 720KB formatted
  (MB equals 1,048,576 bytes; KB equals 1024 bytes)
- 2MB unformatted media, 1.44MB formatted
- 250,000 bits-per-second mode
- 500,000 bits-per-second mode.

Precompensation of 125 nanoseconds is provided for all tracks.

The controller supports both high- and low-density media. The 720KB and 1.44MB formatted diskette densities are supported for single and multithread operations.

Warning: 32-bit operations to the video subsystem can cause a diskette overrun in the 1.44MB mode because data width conversions may require more than 12 microseconds. If an overrun occurs and BIOS returns an error code, retry the operation.

When the diskette drive controller is switched from one density to another, the following occurs:

- The clock rate changes:
  - 8 MHz for high density
  - 4 MHz for low density
- The programmed step rate changes
- Write current changes (reduced write current is used in the high-density mode)
- Number of sectors per track changes:
  - 18 sectors per track in the high-density mode
  - 9 sectors per track in the low-density mode

Warning: The controller does not check to see that the media supports the density selected. 1MB media can not be reliably formatted to the 2MB density. Loss of data can result.

Diskette Drive Controller Registers

Three registers indicate the status of signals used in diskette operations; two registers control certain interface signals. The Diskette Drive Controller Status register and the Data register are also accessed by the system.

Status Register A (Hex 03F0)

This read-only register shows the status of signals on the diskette drive interface. For additional information about signals on the diskette drive interface see "Signal Descriptions" on page 26.

Bit	Function
7	Interrupt Pending
6	-2nd Drive Installed
5	Step
4	-Track 0
3	Head 1 Select
2	-Index
1	-Write Protect
0	Direction

Figure 1. Status Register A (Hex 03F0)

Status Register B (Hex 03F1)

This read-only register shows the status of signals on the diskette drive interface. For additional information about signals on the diskette drive interface see "Signal Descriptions" on page 26.

Bit	Function
7, 6	Reserved
5	Drive Select
4	Write Data – Set to 1 by a positive transition of the '-Write Data' signal.
3	Read Data – Set to 1 by a positive transition of the '-Read Data' signal.
2	Write Enable
1	Motor Enable 1
0	Motor Enable 0

Figure 2. Status Register B (Hex 03F1)

Digital Output Register (Hex 03F2)

This write-only register controls drive motors, drive selection, and feature enable. All bits are set to 0 by a reset.

Bit	Function
7, 6	Reserved
5	Motor Enable 1
4	Motor Enable 0
3	Reserved
2	Diskette Controller Reset
1	Reserved
0	Drive Select (0 = Drive 0; 1 = Drive 1)

Figure 3. Digital Output Register (Hex 03F2)

Digital Input Register (Hex 03F7 - Read)

This read-only register senses the state of the '-diskette change' and '-high density select' signals on the diskette drive interface. For additional information about these signals see "Signal Descriptions" on page 26.

Bit	Function
7	Diskette Change
6 - 1	Reserved
0	-High Density Select

Figure 4. Digital Input Register (Hex 03F7 - Read)

Configuration Control Register (Hex 03F7 - Write)

This write-only register sets the transfer rate.

Bit	Function
7 - 2	Reserved
1, 0	Data Rate Select

Figure 5. Configuration Control Register (Hex 03F7 - Write)

Bits 7 - 2 Reserved.

Bits 1, 0 These bits select the data rate as shown in the following figure.

Bits 1 0	Data Rate
0 0	500,000-bits per second mode
0 1	Reserved
1 0	250,000-bits per second mode
1 1	Reserved

Figure 6. Data Rate Selection

Diskette Drive Controller Status Register (Hex 03F4)

This read-only register facilitates the transfer of data between the system microprocessor and the controller.

Bit	Function
7	Request for Master
6	Data Input/Output
5	Non-DMA Mode
4	Diskette Controller Busy
3	Reserved
2	Reserved
1	Drive 1 Busy
0	Drive 0 Busy

Figure 7. Diskette Drive Controller Status Register (Hex 03F4)

Bit 7 When this bit is set to 1, the Data register is ready to transfer data with the system microprocessor.

Bit 6 This bit indicates the direction of data transfer between the diskette drive controller and the system microprocessor. If this bit is set to 1, the transfer is to the system microprocessor; if this bit is set to 0, the transfer is to the controller.

Bit 5 When this bit is set to 1, the controller is in the non-DMA mode.

Bit 4 When this bit is set to 1, a Read or Write command is being executed.

Bits 3, 2 These bits are reserved.

Bit 1 When this bit is set to 1, diskette drive 1 is in the seek mode.

Bit 0 When this bit is set to 1, diskette drive 0 is in the seek mode.

Data Register (Hex 03F5)

This read/write register passes data, commands, and parameters, and provides diskette-drive status information.

Diskette Drive Controller Programming Considerations

Each command is initiated by a multibyte transfer from the system microprocessor; the result can be a multibyte transfer back to the system microprocessor. Each command consists of three phases:

- *Command Phase:* The system microprocessor writes a series of command bytes to the controller directing it to perform a specific operation.
- *Execution Phase:* The controller performs the specified operation.
- *Result Phase:* After the operation is complete, status information is available to the system microprocessor through a sequence of read commands.

Controller Commands

The following is a list of the commands issued to the diskette drive controller:

- Read Data command
- Read Deleted Data command
- Read a Track command
- Read ID command
- Write Data command
- Write Deleted Data command
- Format a Track command
- Scan Equal command
- Scan Low or Equal command
- Scan High or Equal command
- Recalibrate command
- Sense Interrupt Status command
- Specify command
- Sense Drive Status command
- Seek command.

Notes:

1. Bits 6 and 7 in Status Register 0 are used to indicate command status. When an invalid command is processed, this information is returned to the system microprocessor in the Invalid Command Status byte.
2. Diskette BIOS may not support all commands listed. Whenever possible, the diskette hardware should be accessed through the diskette BIOS.

The following symbols are used in the figures showing the format of individual commands. The command formats start on page 9.

Symbol	Name	Description
HD	Head	The selected head number is 0 or 1.
HLT	Head Load Time	HLT is the head load time in the selected drive (2 to 254 milliseconds in 2-millisecond increments).
HUT	Head Unload Time	HUT is the head unload time after a Read or Write operation (16 to 240 milliseconds in 16-millisecond increments).
MF	FM or MFM Mode	0 selects FM mode and 1 selects MFM.
MT	Multitrack	1 selects multitrack operation. (Head 0 and head 1 will be read or written.)
ND	Non-DMA Mode	ND indicates an operation in the non-DMA mode.
PTN	Present Track Number	PTN is the track number at the completion of a Sense Interrupt Status command (present position of the head).
SK	Skip	SK stands for skip deleted-data address mark.
SRT	Step Rate	SRT is the stepping rate for the diskette drive (1 to 16 milliseconds in 1-millisecond increments).
US	Unit Select	Indicates the drive number selected. 0 selects drive 0; 1 selects drive 1.

Figure 8. Command Symbols, Diskette Drive Controller

The following commands are issued to the controller by the system. An X in the figures indicates that the bit can be either 0 or 1.

Read Data Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	0	0	1	1	0
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 9. Read Data Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 10. Read Data Result

Read Deleted Data Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	0	1	1	0	0
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 11. Read Deleted Data Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 12. Read Deleted Data Result

Read a Track Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	MF	SK	0	0	0	1	0
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 13. Read a Track Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 14. Read a Track Result

Read ID Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	MF	0	0	1	0	1	0
Byte 1	X	X	X	X	X	HD	0	US

Figure 15. Read ID Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 16. Read ID Result

Write Data Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	0	0	0	1	0	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 17. Write Data Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 18. Write Data Result

Write Deleted Data Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	0	0	1	0	0	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Data Length							

Figure 19. Write Deleted Data Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 20. Write Deleted Data Result

Format a Track Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	MF	0	0	1	1	0	0
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Number of Data Bytes in Sector							
Byte 3	Sectors per Track							
Byte 4	Gap Length							
Byte 5	Data							

Figure 21. Format a Track Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 22. Format a Track Result

Scan Equal Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	0	0	0	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 23. Scan Equal Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 24. Scan Equal Result

Scan Low or Equal Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	1	0	0	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 25. Scan Low or Equal Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 26. Scan Low or Equal Result

Scan High or Equal Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	MT	MF	SK	1	1	1	0	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	Track Number							
Byte 3	Head Address							
Byte 4	Sector Number							
Byte 5	Number of Data Bytes in Sector							
Byte 6	End-of-Track							
Byte 7	Gap Length							
Byte 8	Scan Test							

Figure 27. Scan High or Equal Command

Result Phase

Byte 0	Status Register 0
Byte 1	Status Register 1
Byte 2	Status Register 2
Byte 3	Track Number
Byte 4	Head Address
Byte 5	Sector Number
Byte 6	Number of Data Bytes in Sector

Figure 28. Scan High or Equal Result

Recalibrate Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	1	1	1
Byte 1	X	X	X	X	X	0	0	US

Figure 29. Recalibrate Command

Result Phase: This command has no result phase.

Sense Interrupt Status Command

Command Phase

Byte 0	0	0	0	0	1	0	0	0

Figure 30. Sense Interrupt Status Command

Result Phase

Byte 0	Status Register 0
Byte 1	Present Track Number

Figure 31. Sense Interrupt Status Result

Specify Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	0	0	1	1
Byte 1	SRT	SRT	SRT	SRT	HUT	HUT	HUT	HUT
Byte 2	HLT	HLT	HLT	HLT	HLT	HLT	HLT	ND

Figure 32. Specify Command

Result Phase: This command has no result phase.

Sense Drive Status Command

Command Phase

Byte 0	0	0	0	0	0	1	0	0
Byte 1	X	X	X	X	X	HD	0	US

Figure 33. Sense Drive Status Command

Result Phase

Byte 0	Status Register 3

Figure 34. Sense Drive Status Result

Seek Command

Command Phase

	7	6	5	4	3	2	1	0
Byte 0	0	0	0	0	1	1	1	1
Byte 1	X	X	X	X	X	HD	0	US
Byte 2	New Track Number for Seek							

Figure 35. Seek Command

Result Phase: This command has no result phase.

Invalid Command Status

Result Phase: The following status byte is returned to the system microprocessor when an invalid command has been received.

Byte 0	Status Register 0

Figure 36. Invalid Command Result

Command Status Registers

This section provides definitions for status registers 0 through 3.

Status Register 0

Bit	Function
7, 6	Interrupt Code
5	Seek End
4	Equipment Check
3	Not Ready
2	Head Address
1	Reserved = 0
0	Unit Select (0 = Drive 0; 1 = Drive 1)

Figure 37. Status Register 0

Bits 7, 6 These bits indicate the command interrupt status.

Bits 7 6	Function
0 0	Normal Termination of Command
0 1	Abnormal Termination of Command
1 0	Invalid Command Issued
1 1	Reserved

Figure 38. Status Register 0 (Bits 7, 6)

Bit 5 This bit is set to 1 when the diskette drive completes the Seek command.

Bit 4 This bit is set to 1 if the '-track 0' signal fails to occur after the Recalibrate command is issued.

Bit 3 This bit is set to 1 when the diskette drive is in the not-ready state and a Read or Write command is issued. This bit is also set to 1 if a Read or Write command is issued to head 1 of a single-sided diskette drive.

Bit 2 This bit indicates the state of the '-head select' signal after the command was performed. When set to 1, head 1 was selected; when set to 0, head 0 was selected.

Bit 1 Reserved. This bit must be set to 0.

Bit 0 When is set to 1, this bit indicates the command was issued to drive 1. When set to 0, this bit indicates the command was issued to drive 0.

Status Register 1

Bit	Function
7	End-of-Track
6	Reserved = 0
5	Data Error
4	Overrun
3	Reserved = 0
2	No Data
1	Not Writable
0	Missing Address Mark

Figure 39. Status Register 1

Bit 7 This bit is set to 1 when the controller tries to gain access to a sector beyond the final sector of a track.

Bit 6 Reserved.

Bit 5 This bit is set to 1 when a CRC error is detected in the ID or data field.

Bit 4 This bit is set to 1 if the system does not service the diskette drive controller within 12 microseconds during data transfers.

Bit 3 Reserved. This bit is always set to 0.

Bit 2 This bit is set to 1 when:

- The controller cannot find the sector specified in the ID register during the execution of a Read Data, Write Deleted Data, or Scan command
- The controller cannot read the ID field without an error during the execution of a Read ID command
- The starting sector cannot be found during the execution of a Read Track command.

Bit 1 This bit is set to 1 when the '-write-protect' signal is active during a Write Data, Write Deleted Data, or Format Track command.

Bit 0 This bit is set to 1 if the controller cannot detect an address mark. When this occurs, bit 0 of Status Register 2 indicates whether the missing address mark is an ID-address mark or a data-address mark.

Status Register 2

Bit	Function
7	Reserved = 0
6	Control Mark
5	Data Error in Data Field
4	Wrong Track
3	Scan Equal Hit
2	Scan Not Satisfied
1	Bad Track
0	Missing Address Mark in Data Field

Figure 40. Status Register 2

Bit 7 Reserved. This bit is always set to 0.

Bit 6 This bit is set to 1 when the controller encounters a sector that has a deleted data-address mark during a Read Data or Scan command.

Bit 5 This bit is set to 1 if the controller detects an error in the data.

Bit 4 This bit is set to 1 when the track number on the media is different from the track number issued by the command. When this occurs, bit 2 of Status register 1 is also set to 1.

Bit 3 This bit is set to 1 if the contiguous sector data is the same as the data supplied by the system during the execution of a Scan command.

Bit 2 This bit is set to 1 if the contiguous sector data is not the same as the data supplied by the system during the execution of a Scan command.

Bit 1 This bit is set to 1 when the track number on the media is hex FF and the track number value stored in the ID register is not hex FF. When this occurs, bit 2 of Status register 1 is also set to 1.

Bit 0 This bit is set to 1 when the controller cannot find a data-address mark. This bit is set to 0 when an ID-address mark cannot be found. Bit 0 in Status Register 0 is also set if either address mark cannot be found.

Status Register 3

Bit	Function
7	Reserved
6	Write Protect
5	Reserved
4	Track 0
3	Reserved
2	Head Address
1, 0	Reserved

Figure 41. Status Register 3

Bit 7 Reserved.

Bit 6 This bit indicates the status of the '-write-protect' signal from the diskette drive. When this bit is set to 1, the '-write-protect' signal is active.

Bit 5 Reserved.

Bit 4 This bit indicates the status of the '-track 0' signal from the diskette drive. When this bit is set to 1, the '-track 0' signal is active.

Bit 3 Reserved.

Bit 2 This bit indicates the status of the '-head 1 select' signal from the diskette drive. When this bit is set to 1, the '-head 1 select' signal is active.

Bits 1, 0 Reserved.

Signal Descriptions

The diskette-drive controller interface-signal sequences and timings are compatible with the industry standard 3.5-inch diskette drive interface. All interface signals are TTL compatible at the driver/receivers both in the rise and fall times and the interface levels.

The following describes the interface to the diskette drive.

Output Signals

All output signals to the diskette drive operate between 5 Vdc and ground with the following definitions:

- The inactive level is 2.0 Vdc minimum.
- The active level is 0.8 Vdc maximum.

All input signals from the drive can sink 4.0 milliamperes at the active level.

- The inactive level is 3.7 Vdc minimum.
- The active level is 0.4 Vdc maximum.

-HIGH DENSITY SELECT: When this signal is active, the 2MB mode is selected. Diskettes are formatted with 18 sectors per track and a capacity of 1.44MB. When this signal is inactive, the 1MB mode is selected. Diskettes are formatted with 9 sectors per track and a capacity of 720KB. This signal is not used by the 720KB diskette drive.

-DRIVE SELECT 0 - 1: The drive-select signals enable or disable all drive interface signals except -MOTOR ENABLE. When a drive select signal is active, the drive is enabled. When it is inactive, all controlled inputs are ignored and all drive outputs are disabled.

-MOTOR ENABLE 0 - 1: When this signal is made active, the spindle starts to turn. There must be a 500-millisecond delay after -MOTOR ENABLE 0 or -MOTOR ENABLE 1 becomes active before a read, write, or seek operation. When inactive, this signal causes the spindle motor to decelerate and stop.

-DIRECTION: When this signal is active, -STEP moves the heads toward the drive spindle. When this signal is inactive, -STEP moves the heads away from the drive spindle. This signal is stable for 1 microsecond before and after the trailing edge of the -STEP pulse.

Note: After a direction change, a 15-millisecond delay is required before the next '-step' pulse.

-STEP: A 1-microsecond active pulse of this signal causes the read/write heads to move one track. The state of -DIRECTION at the trailing edge of -STEP determines the direction of motion.

Note: Before a read or write operation, a 15-millisecond seek settle time must be allowed.

-WRITE DATA: A 250-nanosecond pulse of this signal causes a bit to be written if -WRITE ENABLE is active. All tracks have a write-precompensation of 125 nanosecond.

-WRITE ENABLE: When active, this signal enables the write current circuits and -WRITE DATA controls the writing of information. Motor-start and head-settle times must be observed before this signal becomes active.

-HEAD 1 SELECT: When active, this signal selects the upper head; when inactive, this signal selects the lower head.

Input Signals

All inputs from the drive can sink 4.0 milliamperes at the active level.

- The inactive level is 3.7 Vdc minimum.
- The active level is 0.4 Vdc maximum.

-INDEX: When the drive senses the index, it generates an active pulse of at least 1 millisecond on this line. This signal is gated to the interface only when a diskette is in the drive.

-TRACK 0: This signal is active when the read/write head is on track 0. Track 0 is determined by a sensor, not a track counter.

The drive can seek to track 0 under control of the system even if a diskette is not inserted. This allows system software to determine how many drives are attached to the system. Software selects each drive and attempts to recalibrate that drive to track 0. The track 0 indication determines whether or not each drive is installed in the system.

-WRITE PROTECT: When this signal is active, a write-protected diskette is in the drive; therefore, the drive will not write data.

-READ DATA: Each bit detected provides a 250-nanosecond active pulse on this line for the 250,000 bits-per-second rate or a 125-nanosecond pulse for the 500,000 bits-per-second rate.

-DISKETTE CHANGE: This signal is active at power-on and latched inactive when a diskette is present, the drive is selected, and a step pulse occurs. This signal goes active when the diskette is removed from the drive. The presence of a diskette is determined by a sensor.

Power Sequencing

-WRITE ENABLE is turned off and is kept off before power is switched on or off. The read/write heads return to track 0 when the system power is switched on.

Connector

Signals and voltages are transferred between the system board and the diskette drives through a cable or printed-circuit board. Both the cable and the printed-circuit board provide a 2- by 20-pin connector for each diskette drive. The locator key is between pins 34 and 36.

The following figure shows the signals and voltages for the connector at the diskette drive.

Pin	I/O	Signal	Pin	I/O	Signal
1	N/A	-2nd Drive Installed	2	O	-High Density Select
3	N/A	Reserved	4	N/A	Reserved
5	N/A	Ground	6	N/A	Reserved
7	N/A	Signal Ground	8	I	-Index
9	N/A	Signal Ground	10	O	Reserved
11	N/A	Signal Ground	12	O	-Drive Select
13	N/A	Ground	14	O	Reserved
15	N/A	Signal Ground	16	O	-Motor Enable
17	N/A	Signal Ground	18	O	-Direction In
19	N/A	Signal Ground	20	O	-Step
21	N/A	Signal Ground	22	O	-Write Data
23	N/A	Signal Ground	24	O	-Write Enable
25	N/A	Signal Ground	26	I	-Track 0
27	N/A	Signal Ground	28	I	-Write Protect
29	N/A	Signal Ground	30	I	-Read Data
31	N/A	Signal Ground	32	O	-Head 1 Select
33	N/A	Signal Ground	34	I	-Diskette Change
35	N/A	Ground	36	N/A	Ground
37	N/A	Ground	38	O	+5 Vdc
39	N/A	Ground	40	O	+12 Vdc

Figure 42. Diskette Drive Connector Voltage and Signal Assignments

Index

C

clock rate 1
command phase 6
command status registers 22
command summary 6, 7
command symbols 8
commands 6
  format a track 15
  invalid command status 21
  read a track 11
  read data 9
  read deleted data 10
  read id 12
  recalibrate 19
  scan equal 16
  scan high or equal 18
  scan low or equal 17
  sense drive status 20
  sense interrupt status 19
  specify 20
  write data 13
  write deleted data 14
configuration control register 4
connector 29
connector voltages, diskette drive 29
controller registers 2

D

data register 5
data width conversions 1
digital input register 4
digital output 3
digital output register 3
diskette command symbols 8
diskette drive connector 29
diskette drive controller 6
diskette drive controller status register 5

E

execution phase 6

F

format a track command 15

I

input signals 28
invalid command status 21

M

multibyte transfer 6

O

output signals 26
output, digital 3

P

precompensation 1
programming considerations 6

R

read a track command 11
read data command 9
read deleted data command 10
read id command 12
recalibrate command 19
registers
  configuration control 4
  data 5
  digital input 4

registers *(continued)*
- digital output 3
- diskette drive controller status 5
- status register A 2
- status register B 3
- status register 0 22
- status register 1 23
- status register 2 24
- status register 3 25

registers, command status 22
registers, controller 2
result phase 6

S

scan equal command 16
scan high or equal command 18
scan low or equal command 17
seek 21
seek command 21
sense drive status command 20
sense interrupt status command 19
signal descriptions 26
specify command 20
status register A 2
status register B 3
status register 0 22
status register 1 23
status register 2 24
status register 3 25
step rate 1
switching density 1

W

write data command 13
write deleted data command 14

Notes:

Keyboard and Auxiliary Device Controller (Type 1)

Description 1
Keyboard Password 2
Controller Status Register 3
  Input and Output Buffers 5
  Input and Output Ports 5
  Controller Commands 7
Keyboard and Auxiliary Device Programming Considerations . . 11
Auxiliary Device/System Timings 12
  System Receiving Data 12
  System Sending Data 13
Signals 15
Connector 15

Figures

1. Controller Status Register, Read Port Hex 0064 3
2. Input Port Bit Definitions 5
3. Output Port Bit Definitions 6
4. Controller Command Byte 7
5. Command A9 Test Results 9
6. Command AB Test Results 9
7. Bit Definitions of Auxiliary-Device Data Stream 12
8. Receiving Data Timings 13
9. Sending Data Timings 14
10. Keyboard and Auxiliary Device Signals 15
11. Keyboard and Auxiliary Device Connector Information . . . 15

Description

Input to the keyboard and auxiliary device controller is through two connectors at the rear of the system unit. One connector is dedicated to the keyboard, the other is available for an auxiliary device. An auxiliary device can be any type of serial input device compatible with the controller interface. The device types include:

- Mouse
- Touchpad
- Trackball
- Keyboard.

The controller receives the serial data, checks the parity, translates keyboard scan codes (see bit 6 of the Controller Command byte on page 7), and presents the data to the system as a byte of data at data port address hex 0060. The interface interrupts the system when data is available or waits for polling from the system microprocessor.

Address hex 0064 is the command/status port. When the system reads port hex 0064, it receives status information from the controller. When the system writes to the port, the controller interprets the byte as a command.

Secondary circuit protection is provided on the system board for the +5 Vdc line to the keyboard and auxiliary device.

Keyboard Password

The controller provides a keyboard password mechanism. Three commands are available for keyboard password operation:

A4	Test Password Installed
A5	Load Password
A6	Enable Password.

The Test Password Installed command determines if a keyboard password is currently installed. The controlling program can use this command to determine if a keyboard password is loaded before enabling the password.

The Load Password command allows the system microprocessor to set a keyboard password in the controller at any time. Any existing password is lost, and the new password becomes active. The password must be installed in scan-code format.

To set the controller to the secure mode, the system microprocessor issues the Enable Password command. Once the Enable Password command is issued, the controller does not pass any information to the system microprocessor. It intercepts the keyboard data stream and continuously compares it to the installed password pattern. Keyboard and auxiliary device data are not passed to the system microprocessor until a match occurs. Then, the state of the controller is restored and data is passed to the system microprocessor.

The keyboard password can be changed at any time. A command to verify the installed password is not provided. Also, commands are not accepted by the controller when the keyboard password is active.

Controller Status Register

The following figure shows the Controller Status register.

Bit	Function
7	Parity Error
6	General Time-Out
5	Auxiliary Device Output Buffer Full
4	Inhibit Switch
3	Command/Data
2	System Flag
1	Input Buffer Full
0	Output Buffer Full

Figure 1. Controller Status Register, Read Port Hex 0064

Note: Controller commands C1 and C2 place data in bits 7 through 4 of the Controller Status register. See commands C1 and C2 on page 10 for more information.

Bit 7 When set to 0, this bit indicates the last byte of data received from the keyboard had odd parity. When set to 1, this bit indicates the last byte had even parity. The keyboard should send with odd parity.

Bit 6 When set to 1, this bit indicates that a transmission was started by the keyboard but did not finish within the programmed receive time-out delay or a transmission was started by the controller and one of the following three errors occurred:

1. If the transmit byte was not clocked out within the specified time limit, this will be the only error.
2. If the transmit byte was clocked out but a response was not received within the programmed time limit, this will be the only error.
3. If the transmit byte was clocked out but the response was received with a parity error, the transmit time-out and parity error bits are set to 1.

Bit 5 This bit works in conjunction with bit 0. When this bit and bit 0 are set to 1, auxiliary device data is in the output buffer. When this bit is set to 0 and bit 0 is set to 1, keyboard or command controller response data is in the output buffer.

Bit 4 When set to 0, this bit indicates the password state is active and the keyboard is inhibited. When set to 1, this bit indicates the password state is inactive and the keyboard is not inhibited. See "Keyboard Password" on page 2 for more information.

Bit 3 The keyboard controller input buffer may be addressed as either address hex 0060 or 0064. Address hex 0060 is defined as the data port, and address hex 0064 is defined as the command/status port. Writing to address hex 0064 sets this bit to 1; writing to address hex 0060 sets this bit to 0. The controller uses this bit to determine if the byte in its input buffer should be interpreted as a command byte or a data byte.

Bit 2 This bit is set to 0 or 1 by writing to the system flag bit (bit 2) in the Controller Command byte. This bit is set to 0 after a power-on reset.

Bit 1 When set to 1, this bit indicates that data has been written into the buffer, but the controller has not read the data. When the controller reads the input buffer, this bit returns to 0. When set to 0, this bit indicates the keyboard controller input buffer (address hex 0060 or 0064) is empty.

Bit 0 When set to 1, this bit indicates the controller has placed data into its output buffer, but the system microprocessor has not yet read the data. When the system microprocessor reads the output buffer (address hex 0060), this bit returns to 0. When set to 0, this bit indicates the keyboard controller output buffer has no data.

Input and Output Buffers

The output buffer is an 8-bit, read-only register at address hex 0060. When the output buffer is read, the controller sends information to the system microprocessor. The information can be keyboard scan codes, auxiliary device data, or data bytes from a controller command.

The input buffer is an 8-bit, write-only register at address hex 0060 or address hex 0064. When the input buffer is written to, the Input Buffer Full bit (bit 1) in the Controller Status Byte is set to 1. Data written to the input buffer through address hex 0064 is interpreted as a controller command. Data written to address hex 0060 is sent to the keyboard, unless the controller expects a data byte following a controller command. Bit 3 of the Controller Status register indicates whether the contents of the input buffer is a command or a data byte.

Data should be written to the controller input buffer only if the Input Buffer Full bit (bit 1) in the Controller Status register (address hex 0064) is 0.

Input and Output Ports

The input port consists of two signals driven to the controller by the keyboard and auxiliary device. The output port consists of eight signals driven by the controller to the keyboard, auxiliary device, or system interface. The following figures show the input port and the output port bytes.

Bit	Function
7 - 2	Reserved = 0
1	Auxiliary Data In
0	Keyboard Data In

Figure 2. Input Port Bit Definitions

Bit 7 - 2 Reserved.

Bit 1 This bit reflects the state of the 'data' line driven by the auxiliary device. For more information on the auxiliary device 'data' line, see Figure 7 on page 12.

Bit 0 This bit reflects the state of the 'data' line driven by the keyboard.

Bit	Function
7	Keyboard Data Out
6	Keyboard Clock Out
5	IRQ12
4	IRQ01
3	Auxiliary Clock Out
2	Auxiliary Data Out
1	Gate Address Line 20
0	Reset Microprocessor

Figure 3. Output Port Bit Definitions

Bit 7 This bit reflects the state of the 'data' line driven by the controller to the keyboard.

Bit 6 This bit reflects the state of the 'clock' line driven by the controller to the keyboard.

Bit 5 When set to 1, this bit indicates an interrupt has been generated by data from the auxiliary device in the output buffer. When the system reads the data from address hex 0060, this bit will be set to 0.

Bit 4 When set to 1, this bit indicates an interrupt has been generated by data from the keyboard or a command in the output buffer. When the system reads the data form address hex 0060, this bit will be set to 0.

Bit 3 This bit reflects the state of the 'clock' line driven by the controller to the auxiliary device.

Bit 2 This bit reflects the state of the 'data' line driven by the controller to the auxiliary device.

Bit 1 When set to 1, this bit indicates that the system address line A20 will be set to 0 so that memory accesses above 1 MB will wrap around the low memory. This bit is set to 0 at power-on.

Bit 0 This bit reflects the state of the 'data' line driven by the keyboard. When set to 0, this bit resets the system microprocessor.

Controller Commands

A command is a data byte written to the controller through address hex 0064. The following are the recognized commands, shown in hex values:

20 - 3F Read the Controller RAM: This command causes the controller to return the data in the internal address specified by bits 5 through 0 of this command. Internal address 0 is assigned as the Controller Command byte. Command hex 20 requests a Read of the Controller Command byte. Data will be output to port hex 0060 by the controller.

Bit	Function
7	Reserved = 0
6	IBM Keyboard Translate Mode
5	Disable Auxiliary Device
4	Disable Keyboard
3	Reserved = 0
2	System Flag
1	Enable Auxiliary Interrupt
0	Enable Keyboard Interrupt

Figure 4. Controller Command Byte

Bit 7 This bit is reserved and must be set to 0.

Bit 6 When this bit is set to 1, the controller translates the incoming scan codes to scan-code set 1. When this bit is set to 0, the controller passes the keyboard scan codes without translation. The default scan-code set for the keyboard is scan-code set 2.

Bit 5 When set to 1 by a write operation, this bit disables the auxiliary device interface by driving the 'clock' line low. Data is not sent or received.

Bit 4 When set to 1 by a write operation, this bit disables the keyboard interface by driving the 'clock' line low. Data is not sent or received.

Bit 3 This bit is reserved and must be set to 0.

Bit 2 The value written to this bit is placed in the system flag bit of the Controller Status register.

Bit 1 When set to 1 by a write operation, this bit causes the controller to generate an interrupt (IRQ 12) when it places auxiliary device data into its output buffer.

Bit 0 When set to 1 by a write operation, this bit causes the controller to generate an interrupt (IRQ 1) when it places keyboard or command controller response data into its output buffer.

60 - 7F Write the Controller RAM: bits 5 through 0 of the command specify the address. Command hex 0060 writes the Controller Command byte: The next byte of data written to address hex 0060 is placed in the Controller Command byte. For more information about the Controller Command byte, see Controller Command 20 on page 7.

A4 Test Password Installed: This command checks for a password currently installed in the controller. The test result is placed in the output buffer (address hex 0060 and IRQ 01). Hex FA means the password is installed; hex F1 means it is not installed.

A5 Load Password: This command initiates the Password Load procedure. Following this command the controller takes input from the data port until a null (0) is detected. The null terminates password entry.

A6 Enable Password: This command enables the controller password feature. This command is valid only when a password pattern is currently loaded in the controller.

A7 Disable Auxiliary Device Interface: This command sets bit 5 of the Controller Command byte to 1. This disables the auxiliary device interface by driving the 'clock' line low. Data is not sent or received.

A8 Enable Auxiliary Device Interface: This command sets bit 5 of the Controller Command byte to 0, releasing the auxiliary device interface.

A9 Auxiliary Device Interface Test: This command causes the controller to test the auxiliary device 'clock' and 'data' lines. The test result is placed in the output buffer (address hex 0060 and IRQ 01) as shown in the following figure.

Test Result (Hex)	Meaning
00	No error detected
01	The auxiliary device 'clock' line is stuck low.
02	The auxiliary device 'clock' line is stuck high.
03	The auxiliary device 'data' line is stuck low.
04	The auxiliary device 'data' line is stuck high.

Figure 5. Command A9 Test Results

AA Self-Test: This command causes the controller to perform internal diagnostic tests. A hex 55 is placed in the output buffer if no errors are detected.

AB Keyboard Interface Test: This command causes the controller to test the keyboard 'clock' and 'data' lines. The test result is placed in the output buffer (address hex 0060 and IRQ 01) as shown in the following figure.

Test Result (Hex)	Meaning
00	No error detected
01	The keyboard 'clock' line is stuck low.
02	The keyboard 'clock' line is stuck high.
03	The keyboard 'data' line is stuck low.
04	The keyboard 'data' line is stuck high.

Figure 6. Command AB Test Results

AC This command is reserved.

AD Disable Keyboard Interface: This command sets bit 4 of the Controller Command byte to 1. This disables the keyboard interface by driving the 'clock' line low. Data is not sent or received.

AE Enable Keyboard Interface: This command clears bit 4 of the Controller Command byte to 0, releasing the keyboard interface.

C0 Read Input Port: This command causes the controller to read its input port and place the data in its output buffer. This command should be used only if the output buffer is empty.

C1 Poll Input Port Low: This command causes the controller to read its input port bits 0 - 3, and place the data in bits 4 - 7 of the Controller Status register.

C2 Poll Input Port High: This command causes the controller to read its input port bits 4 - 7 and place the data in bits 4 - 7 of the Controller Status register.

D0 Read Output Port: This command causes the controller to read its output port and place the data in its output buffer. This command should be used only if the output buffer is empty.

D1 Write Output Port: The next byte of data written to address hex 0060 is placed in the controller output port.

Note: Bit 0 of the controller output port is connected to System Reset. This bit should not be set to 0.

D2 Write Keyboard Output Buffer: The next byte written to address hex 0060 input buffer is written to address hex 0060 output buffer as if initiated by the keyboard. An interrupt occurs if the interrupt is enabled in the Controller Command byte.

D3 Write Auxiliary Device Output Buffer: The next byte written to address hex 0060 input buffer is written to address hex 0060 output buffer as if initiated by an auxiliary device. An interrupt occurs if the interrupt is enabled in the Controller Command byte.

D4 Write to Auxiliary Device: The next byte written to address hex 0060 input buffer is transmitted to the auxiliary device.

E0 Read Test Inputs: This command causes the controller to read its keyboard 'clock' in (T0) and auxiliary device 'clock' in (T1) inputs. This data is placed in the output buffer. Data bit 0 represents T0 and data bit 1 represents T1.

F0 - FF Pulse Output Port: Bits 0 through 3 of the controller output port may be pulsed low for approximately 6 microseconds. Bits 0 through 3 of this command indicate which bits are to be pulsed; 0 indicates the bit should be pulsed, 1 indicates the bit should not be modified.

Note: Bit 0 of the controller output port is connected to System Reset. Pulsing this bit resets the system microprocessor.

Keyboard and Auxiliary Device Programming Considerations

The following are some programming considerations for the keyboard and auxiliary device controller:

- Address hex 0064 (Controller Status register) can be read at any time.
- Address hex 0060 should be read only when the Output Buffer Full bit in the Controller Status register is 1.
- The auxiliary-device output buffer full bit in the Controller Status register indicates the data in address hex 0060 came from the auxiliary device. This bit is valid only when the output buffer full bit is set to 1.
- Address hex 0060 and address hex 0064 should be written to only when the Controller Status register input buffer full bit and output buffer full bit are set to 0.
- Auxiliary devices connected to the controller should be disabled before initiating a command that generates output. If output is generated, any value in the output buffer is overwritten.
- An external latch holds the level-sensitive interrupt request until the system microprocessor reads address hex 0060.

Auxiliary Device/System Timings

Data transmissions to and from the auxiliary device connector consist of an 11-bit data stream sent serially over the 'data' line. The following figure shows the function of each bit.

Bit	Function
11	Stop bit (always 1)
10	Parity bit (odd parity)
9	Data bit 7 (most-significant)
8	Data bit 6
7	Data bit 5
6	Data bit 4
5	Data bit 3
4	Data bit 2
3	Data bit 1
2	Data bit 0 (least-significant)
1	Start bit (always 0)

Figure 7. Bit Definitions of Auxiliary-Device Data Stream

The parity bit is either 1 or 0, and the 8 data bits, plus the parity bit, always have an odd number of 1′s.

System Receiving Data

The following describes the typical sequence of events when the system is receiving data from the auxiliary device. A graphic representation showing the timing relationships is presented in Figure 8 on page 13.

1. The auxiliary device checks the 'clock' line. If the line is inactive, output from the device is not allowed.

2. The auxiliary device checks the 'data' line. If the line is inactive, the controller receives data from the system.

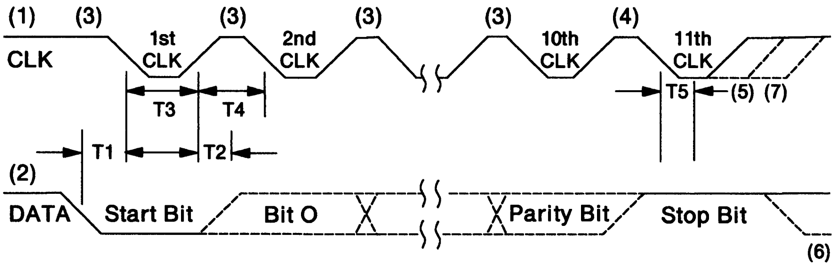
3. The auxiliary device checks the 'clock' line during the transmission at intervals not exceeding 100 microseconds. If the device finds the system holding the 'clock' line inactive, the transmission is terminated. The system can terminate transmission anytime during the first 10 clock cycles.

4. A final check for terminated transmission is performed at least 5 microseconds after the 10th clock.

5. The system can hold the ‘clock’ signal inactive to inhibit the next transmission.

6. The system can set the ‘data’ line inactive if it has a byte to transmit to the device. When the ‘data’ line is inactive, the system has data to transmit. The ‘data’ line is set inactive when the start bit (always 0) is placed on the ‘data’ line.

7. The system raises the ‘clock’ line to allow the next transmission.



	Timing Parameter	Min/Max
T1	Time from DATA transition to falling edge of CLK	5/25 µs
T2	Time from rising edge of CLK to DATA transition	5/T4 - 5 µs
T3	Duration of CLK inactive	30/50 µs
T4	Duration of CLK active	30/50 µs
T5	Time to auxiliary device inhibit after clock 11 to ensure the auxiliary device does not start another transmission	>0/50 µs

Figure 8. Receiving Data Timings

System Sending Data

The following describes the typical sequence of events when the system is sending data to the auxiliary device. A graphic representation showing the timing relationships is presented in Figure 9 on page 14.

1. The system checks for an auxiliary device transmission in process. If a transmission is in process and beyond the 10th clock, the system must receive the data.

2. The auxiliary device checks the ‘clock’ line. If the line is inactive, an I/O operation is not allowed.

3. The auxiliary device checks the 'data' line. If the line is inactive, the system has data to transmit. The 'data' line is set inactive when the start bit (always 0) is placed on the 'data' line.

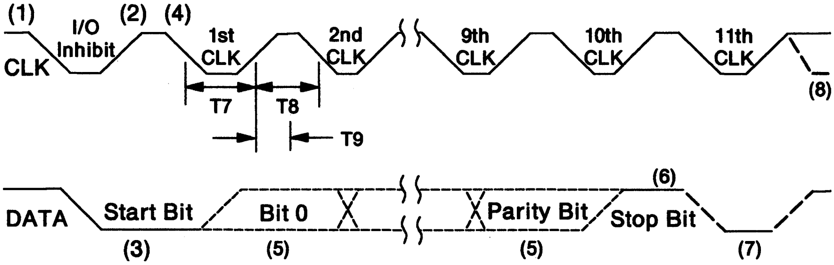
4. The auxiliary device sets the 'clock' line inactive. The system then places the first bit on the 'data' line. Each time the auxiliary device sets the 'clock' line inactive, the system places the next bit on the 'data' line until all bits are transmitted.

5. The auxiliary device samples the 'data' line for each bit while the 'clock' line is active. Data must be stable within 1 microsecond after the rising edge of the 'clock' line.

6. The auxiliary device checks for a positive level stop bit after the 10th clock. If the 'data' line is inactive, the auxiliary device continues to clock until the 'data' line becomes active, then clocks the line-control bit, and at the next opportunity sends a Resend command to the system.

7. The auxiliary device pulls the 'data' line inactive, producing the line-control bit.

8. The system can pull the 'clock' line inactive, inhibiting the auxiliary device.



	Timing Parameter	Min/Max
T7	Duration of CLK inactive	30/50 µs
T8	Duration of CLK active	30/50 µs
T9	Time from inactive to active CLK transition, used to time when the auxiliary device samples DATA	5/25 µs

Figure 9. Sending Data Timings

Signals

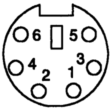
The keyboard and auxiliary device signals are driven by open-collector drivers pulled to 5 Vdc through 10-kilohm resistors. The following lists the characteristics of the signals.

Sink Current	20 mA	Maximum
High-Level Output Voltage	5.0 Vdc minus pullup	Minimum
Low-Level Output Voltage	0.5 Vdc	Maximum
High-Level Input Voltage	2.0 Vdc	Minimum
Low-Level Input Voltage	0.8 Vdc	Maximum

Figure 10. Keyboard and Auxiliary Device Signals

Connector

The keyboard and auxiliary device connectors use 6-pin miniature DIN connectors. The signals and voltages are the same for both connectors, and are assigned as shown in the following figure.



Pin	I/O	Signal Name
1	I/O	Data
2	NA	Reserved
3	NA	Ground
4	NA	+5 Vdc
5	I/O	Clock
6	NA	Reserved

Figure 11. Keyboard and Auxiliary Device Connector Information

Notes:

Serial Port Controller (Types 1 and 2)

Description 1
Communications Application 2
Programmable Baud-Rate Generator 3
Registers 3
  Transmitter Holding Register (Hex nF8) 4
  Receiver Buffer Register (Hex nF8) 5
  Divisor Latch Registers (Hex nF8 and nF9) 5
  Interrupt Enable Register (Hex nF9) 6
  Interrupt Identification Register (Hex nFA) 7
  FIFO Control Register (Hex nFA) 9
  Line Control Register (Hex nFB) 11
  Modem Control Register (Hex nFC) 12
  Line Status Register (Hex nFD) 14
  Modem Status Register (Hex nFE) 17
  Scratch Register (Hex nFF) 18
FIFO Modes of Operation 18
  FIFO Interrupt Mode Operation 18
  FIFO Polled Mode Operation 20
Serial Port Controller Programming Considerations 21
Signal Descriptions 21
  Modem-Control Input Signals 21
  Modem-Control Output Signals 22
Voltage Interchange Information 22
Connector 23

Index 24

Figures

1. Serial Port Controller Block Diagram 2
2. Serial Port Data Format 3
3. Serial Port Register Addresses 4
4. Transmitter Holding Register 4
5. Receiver Buffer Register 5
6. Divisor Latch Register, Low Byte (Hex nF8) 5
7. Divisor Latch Register, High Byte (Hex nF9) 6
8. Baud Rates at 1.8432 MHz 6
9. Interrupt Enable Register (Hex nF9) 7
10. Interrupt Identification Register (Hex nFA) 8
11. Interrupt Control Functions 9
12. FIFO Control Register (Hex nFA) 10
13. Trigger Level 10
14. Line Control Register (Hex nFB) 11
15. Stop Bits 12
16. Word Length 12
17. Modem Control Register (Hex nFC) 12
18. Line Status Register (Hex nFD) 14
19. Modem Status Register (Hex nFE) 17
20. Voltage Levels 23
21. Serial Port Connector Signal and Pin Assignments 23

Description

The serial port controller is programmable and supports asynchronous communications. The controller automatically adds and removes start, stop, and parity bits. A programmable baud-rate generator allows operation from 50 baud to 19,200 baud. The controller supports 5-, 6-, 7- and 8-bit characters with 1, 1.5, or 2 stop bits. A prioritized interrupt system controls transmit, receive, error, and line status and data-set interrupts.

The serial port controller provides the following functions:

- Full double buffering in the character mode, eliminating the need for precise synchronization
- False-start bit detection
- Line-break generation and detection
- Modem control functions:
  Clear to send (CTS)
  Request to send (RTS)
  Data set ready (DSR)
  Data terminal ready (DTR)
  Ring indicator (RI)
  Data carrier detect (DCD).

Two types of serial port controllers have been used on the system boards. To programs, the Type 1 controller appears to be identical to the serial portion of the IBM Personal Computer AT IBM Personal Computer Serial/Parallel Adapter. The Type 2 controller incorporates all functions of the Type 1 and also provides support of the first-in-first-out (FIFO) mode.

Note: Some systems using the Type 2 controller do not support the FIFO mode. For information about individual systems refer to the system-specific technical reference manuals.

Support for the Type 1 controller is restricted to the functions that are identical to the NS16450. Using the Type 1 controller in the FIFO mode may result in nondetectable data errors. See "Registers" on page 3 for detailed FIFO information.

The following figure is a block diagram of the serial port controller.

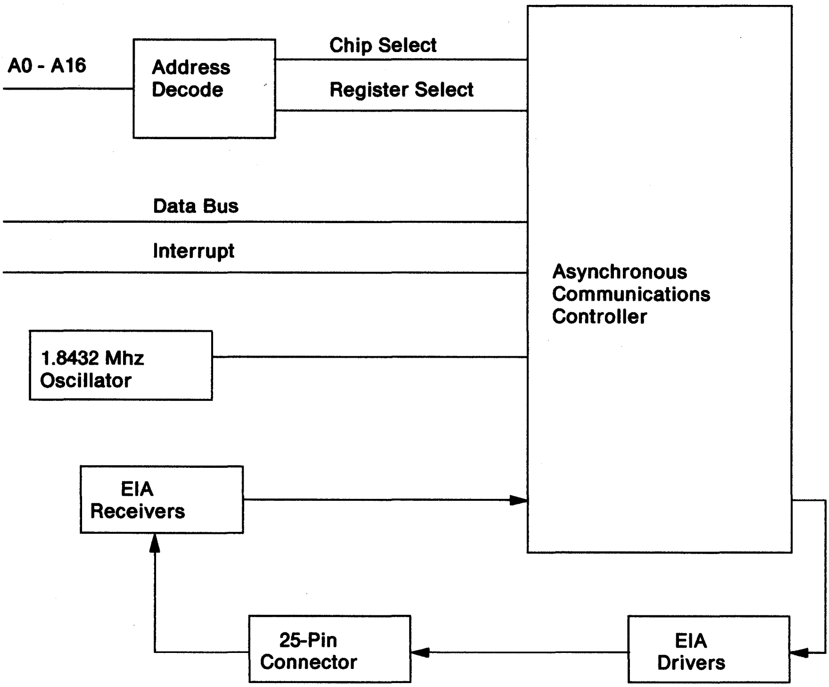


Figure 1. Serial Port Controller Block Diagram

Communications Application

The serial output port can be addressed as either serial output port 1 (Serial 1) or serial output port 2 (Serial 2). In this section, serial port register addresses contain an **n**. The **n** can be either 03 for Serial 1, or 02 for Serial 2. The port assignments are controlled by POS during system board setup.

Two interrupt lines are provided to the system: Interrupt level 4 (IRQ4) is for Serial 1 and interrupt level 3 (IRQ3) is for Serial 2. For the serial port controller to send interrupts to the interrupt controller, bit 3 of the Modem Control register must be set to 1. Any interrupts allowed by the Interrupt Enable register will cause an interrupt.

The data format is shown in the following figure.

Marking	Start Bit	D0	D1	D2	D3	D4	D5	D6	D7	Parity Bit	Stop Bit

Figure 2. Serial Port Data Format

Data bit 0 (D0) is the first bit to be sent or received. The controller automatically inserts the start bit, the correct parity bit (if programmed to do so), and the stop bits (1, 1.5, or 2 depending on the command in the Line Control register).

Programmable Baud-Rate Generator

The controller has a programmable baud-rate generator that can divide the clock input (1.8432 MHz) by any divisor from 1 to 65,535. The output frequency of the baud-rate generator is the baud rate multiplied by 16. Two 8-bit latches store the divisor in a 16-bit binary format. The divisor latches are loaded during setup to ensure desired operation of the baud-rate generator. When either of the divisor latches is loaded, a 16-bit baud counter is immediately loaded. This prevents long counts on the first load.

Registers

The controller has several accessible registers. These control the operations of the controller and transmit and receive data. The system programmer can gain access to or control any of the controller registers through the system microprocessor.

The bit definitions of the Interrupt Enable register, Interrupt Identification register, and Line Status register have been modified from the NS16450 registers, and a FIFO Control register has been added to support the FIFO mode.

Note: Using the Type 1 controller in the FIFO mode may result in nondetectable data errors. See "Interrupt Identification Register (Hex nFA)" on page 7 to see how programs can determine if the FIFO mode can be safely used.

Specific registers are selected according to the following figure.

DLAB State *	Port Address (Hex)	R/W	Register
0	0nF8 **	W	Transmitter Holding Register **n**
0	0nF8 **	R	Receiver Buffer Register **n**
1	0nF8 **	R/W	Divisor Latch, Low Byte
1	0nF9 **	R/W	Divisor Latch, High Byte
0	0nF9 **	R/W	Interrupt Enable Register
X	0nFA **	R	Interrupt Identification Register
X	0nFA **	W	FIFO Control Register
X	0nFB **	R/W	Line Control Register
X	0nFC **	R/W	Modem Control Register
X	0nFD **	R	Line Status Register
X	0nFE **	R	Modem Status Register
X	0nFF **	R/W	Scratch Register

* The DLAB state is controlled by bit 7 of the Line Control register.
** The **n** determines the port selected; 3 is for Serial 1, and 2 is for Serial 2.

Figure 3. Serial Port Register Addresses

Transmitter Holding Register (Hex nF8)

The Transmitter Holding register contains the character to be sent. Bit 0 is the least-significant bit and the first bit sent serially, as shown in the following figure.

Bit	Function
7	Data Bit 7
6	Data Bit 6
5	Data Bit 5
4	Data Bit 4
3	Data Bit 3
2	Data Bit 2
1	Data Bit 1
0	Data Bit 0

Figure 4. Transmitter Holding Register

Receiver Buffer Register (Hex nF8)

The Receiver Buffer register contains the received character. Bit 0 is the least-significant bit and the first bit received serially, as shown in the following figure.

Bit	Function
7	Data Bit 7
6	Data Bit 6
5	Data Bit 5
4	Data Bit 4
3	Data Bit 3
2	Data Bit 2
1	Data Bit 1
0	Data Bit 0

Figure 5. Receiver Buffer Register

Divisor Latch Registers (Hex nF8 and nF9)

The Divisor Latch registers are used to program the baud-rate generator. The values in these two registers form the divisor of the clock input (1.8432 MHz), which establishes the desired baud rate.

Bit	Function
7	Bit 7
6	Bit 6
5	Bit 5
4	Bit 4
3	Bit 3
2	Bit 2
1	Bit 1
0	Bit 0

Figure 6. Divisor Latch Register, Low Byte (Hex nF8)

Bit	Function
7	Bit 7
6	Bit 6
5	Bit 5
4	Bit 4
3	Bit 3
2	Bit 2
1	Bit 1
0	Bit 0

Figure 7. Divisor Latch Register, High Byte (Hex nF9)

The following figure illustrates the use of the baud-rate generator with a frequency of 1.8432 MHz. For baud rates of 19,200 and below, the error obtained is minimal.

Note: Data speed should not exceed 19,200 baud.

Desired Baud Rate	Divisor Used to Generate 16x Clock (Decimal)	(Hex)	Percent of Error Difference between Desired and Actual
50	2304	0900	--
75	1536	0600	--
110	1047	0417	0.026
134.5	857	0359	0.058
150	768	0300	--
300	384	0180	--
600	192	00C0	--
1200	96	0060	--
1800	64	0040	--
2000	58	003A	0.69
2400	48	0030	--
3600	32	0020	--
4800	24	0018	--
7200	16	0010	--
9600	12	000C	--
19200	6	0006	--

Figure 8. Baud Rates at 1.8432 MHz

Interrupt Enable Register (Hex nF9)

This 8-bit register allows the four types of controller interrupts to separately activate the 'chip interrupt' output signal. The interrupt system can be totally disabled by setting bits 0 through 3 of the Interrupt Enable register to 0. Similarly, by setting the appropriate bits of this register to 1, selected interrupts can be enabled.

Disabling the interrupts inhibits the 'chip interrupt' output signal from the controller. All other system functions operate normally, including the setting of the Line Status and Modem Status registers.

Bit	Function
7 - 4	Reserved = 0
3	Modem-Status Interrupt
2	Receiver-Line-Status Interrupt
1	Transmitter-Holding-Register-Empty Interrupt
0	Received-Data-Available Interrupt (Character and FIFO Mode) and Time-Out Interrupts (FIFO Mode Only)

Figure 9. Interrupt Enable Register (Hex nF9)

Bits 7 - 4 These bits are reserved and always set to 0.

Bit 3 When set to 1, this bit enables the modem-status interrupt.

Bit 2 When set to 1, this bit enables the receiver-line-status interrupt.

Bit 1 When set to 1, this bit enables the transmitter-holding-register-empty interrupt.

Bit 0 When set to 1, this bit enables the received-data-available interrupt. In the FIFO mode, this bit also enables the time-out interrupts.

Interrupt Identification Register (Hex nFA)

To minimize programming overhead during data character transfers, the controller prioritizes interrupts into four levels:

- Priority 1 - Receiver-line-status
- Priority 2 - Received-data-available
- Priority 2 - Time-out (FIFO mode only)
- Priority 3 - Transmitter-holding-register-empty
- Priority 4 - Modem status.

Information about a pending interrupt is stored in the Interrupt Identification register. When this register is addressed, the pending interrupt with the highest priority is held and no other interrupts are acknowledged until the system microprocessor services that interrupt.

Bit	Function
7, 6	FIFO Registers Enabled
5, 4	Reserved = 0
3	Interrupt ID, Bit 2
2	Interrupt ID, Bit 1
1	Interrupt ID, Bit 0
0	Interrupt Pending = 0

Figure 10. Interrupt Identification Register (Hex nFA)

Bits 7,6 Programs can determine whether a Type 1 or a Type 2 controller is present by reading these two bits when bit 0 of the FIFO Control register is set to 1. If bits 7 and 6 are set to 1, the Type 2 controller is present and FIFO support is provided. If bit 6 is set to 0, the controller is a Type 1 and FIFO mode should not be used.

Note: Some systems using the Type 2 controller do not support the FIFO mode. For information about individual systems refer to the system-specific technical reference manuals.

Bits 5, 4 These bits are reserved and always set to 0.

Bit 3 In the FIFO mode, this bit is set to 1, along with bit 2, to indicate that a time-out interrupt is pending. In the character mode, this bit is always set to 0.

Bits 2, 1 These two bits identify the pending interrupt with the highest priority, as shown in Figure 11 on page 9.

Bit 0 When this bit is set to 1, no interrupt is pending, and polling (if used) continues. When this bit is set to 0, an interrupt is pending, and the contents of this register can be used as a pointer to the appropriate interrupt service routine.

This bit can be used in either hard-wired, prioritized, or polled conditions to indicate if an interrupt is pending.

Bits 3 2 1 0	Priority	Type	Cause	Interrupt Reset Control
0 0 0 1	-	None	None	-
0 1 1 0	Highest	Receiver Line Status	Overrun, Parity, or Framing Error or Break Interrupt	Read the Line Status Register.
0 1 0 0	Second	Received Data Available	Data is in the Receiver Buffer or the Trigger Level Has Been Reached.	Read the Receiver Buffer Register or the FIFO Register Drops Below the Trigger Level.
1* 1 0 0	Second	Character Time-Out Indication	No Characters Have Been Removed From or Put Into the Receiver FIFO Register During the Last Four Character Times, and at Least 1 Character is in it at This Time.	Read the Receiver Buffer Register.
0 0 1 0	Third	Transmitter Holding Register Empty	Transmitter Holding Register is Empty.	Read the Interrupt Identification Register or Write to Transmitter Holding Register.
0 0 0 0	Fourth	Modem Status	Change in Signal Status From Modem.	Read the Modem Status Register.

* FIFO Mode Only

Figure 11. Interrupt Control Functions

FIFO Control Register (Hex nFA)

The FIFO Control register is a write-only register at the same location as the read-only Interrupt Identification register. The FIFO Control register enables the FIFO registers, clears the FIFO registers, and sets the Receiver FIFO register trigger level.

Note: The Transmitter and Receiver FIFO registers are not accessible serial controller registers.

The contents of the FIFO Control register are shown in the following figure.

Bit	**Function**
7, 6	Receiver FIFO Register Trigger
5 - 3	Reserved = 0
2	Transmitter FIFO Register Reset
1	Receiver FIFO Register Reset
0	FIFO Enable

Figure 12. FIFO Control Register (Hex nFA)

Bits 7, 6 These bits indicate the trigger level for the receiver-FIFO-register interrupt, as shown in the following figure.

Bits	**Receiver FIFO Register**
7 6	**Trigger Level**
0 0	01 Byte
0 1	04 Bytes
1 0	08 Bytes
1 1	14 Bytes

Figure 13. Trigger Level

Bits 5 - 3 These bits are reserved and always set to 0.

Bit 2 When this bit is set to 1, all bytes in the Transmitter FIFO register are cleared and its counter logic is reset to 0. The Transmitter Shift register is not cleared. The 1 written to this bit position is self-clearing.

Bit 1 When this bit is set to 1, all bytes in the Receiver FIFO register are cleared and its counter logic is cleared to 0. The Transmitter Shift register is not cleared. The 1 written to this bit position is self-clearing.

Bit 0 When this bit is set to 1, both the Transmitter and Receiver FIFO registers are enabled. When set to 0, this bit clears all bytes in both FIFO registers. When the mode changes from the FIFO mode to the character mode or the character mode to the FIFO mode, data is automatically cleared from the FIFO registers. This bit must be set to 1 when other FIFO Control register bits are written or the bits will not be programmed.

Line Control Register (Hex nFB)

The format of asynchronous communications is programmed through the Line Control register.

Bit	Function
7	Divisor Latch Access Bit
6	Set Break
5	Stick Parity
4	Even Parity Select
3	Parity Enable
2	Number of Stop Bits
1	Word Length Select, Bit 1
0	Word Length Select, Bit 0

Figure 14. Line Control Register (Hex nFB)

Bit 7 This bit must be set to 1 during a read or write operation to gain access to the divisor latches of the baud-rate generator. This bit must be set to 0 to gain access to the Receiver Buffer, Transmitter Holding, or Interrupt Enable registers.

Bit 6 When this bit is set to 1, set break is enabled. The serial output is forced to the spacing state and remains there regardless of other transmitter activity. When this bit is set to 0, set break is disabled.

Bit 5 When bits 5, 4, and 3 are set to 1, the parity bit is sent and checked as a logical 0. When bits 5 and 3 are set to 1, and bit 4 is set to 0, the parity bit is sent and checked as a logical 1. If bit 5 is set to 0, stick parity is disabled.

Bit 4 When this bit and bit 3 are set to 1, an even number of logical 1's are transmitted and checked in the data word bits and parity bit. When this bit is set to 0, and bit 3 is set to 1, an odd number of logical 1's are transmitted and checked in the data word bits and parity bit.

Bit 3 When set to 1, a parity bit is generated (transmit data) or checked (receive data) between the last data-word bit and stop bit of the serial data. (The parity bit produces an even or odd number of 1's when the data-word bits and the parity bit are summed.)

Bit 2 This bit, with bits 0 and 1, specifies the number of stop bits in each serial character sent or received, as shown in the following figure.

Bit 2	Word Length *	Number of Stop Bits
0	N/A	1
1	5-Bits	1.5
1	6-Bits	2
1	7-Bits	2
1	8-Bits	2

* Word length is specified by bits 1 and 0 in this register.

Figure 15. Stop Bits

Bits 1, 0 These bits specify the number of bits in each serial character sent or received. Word length is selected, as shown in the following figure.

Bit 1 0	Word Length
0 0	5-Bits
0 1	6-Bits
1 0	7-Bits
1 1	8-Bits

Figure 16. Word Length

Modem Control Register (Hex nFC)

This 8-bit register controls the data exchange with the modem, data set, or peripheral device emulating a modem.

Bit	Function
7 - 5	Reserved = 0
4	Loop
3	Out 2
2	Out 1
1	Request-to-Send
0	Data-Terminal-Ready

Figure 17. Modem Control Register (Hex nFC)

Bits 7 - 5 These bits are reserved and always set to 0.

Bit 4 This bit provides a local loopback feature for diagnostic testing of the serial controller. When bit 4 is set to 1:

- Transmitter-serial output is set to the marking state.
- Receiver-serial input is disconnected.
- Output of the Transmitter Shift register is “looped back” to the Receiver Shift register input.

  Note: The Transmitter and Receiver Shift registers are not accessible serial controller registers.

- The modem control inputs (-CTS, -DSR, -DCD, and -RI) are disconnected.
- The modem control outputs (-DTR, -RTS, -OUT 1, and -OUT 2) are internally connected to the four modem control inputs.
- The modem control output pins are forced inactive.

When the serial port is in diagnostic mode, transmitted data is immediately received. This mode allows the system microprocessor to verify the transmit-data and receive-data paths of the serial port.

When the serial port is in diagnostic mode, the receiver and transmitter interrupts are fully operational. The modem control interrupts are also operational, but their sources are the lower 4 bits of the Modem Control register instead of the four modem control input signals. The interrupts are still controlled by the Interrupt Enable register.

Bit 3 This bit controls the ‘-output 2’ signal (-OUT 2), which is an auxiliary user-designated interrupt enable signal. -OUT 2 controls the interrupt signal to the channel. Setting this bit to 1 enables the interrupt. Setting this bit to 0 disables the interrupt.

Bit 2 This bit controls the ‘-output 1’ signal (-OUT 1), which is an auxiliary user-designated output signal. When this bit is set to 1, -OUT 1 is forced active. When this bit is set to 0, -OUT 1 is forced inactive.

Bit 1 This bit controls the ‘-request to send’ signal (-RTS) modem control output. When this bit is set to 1, -RTS is forced active. When this bit is set to 0, -RTS is forced inactive.

Bit 0 This bit controls the '-data terminal ready' signal (-DTR) modem control output. When this bit is set to 1, -DTR is forced active. When this bit is set to 0, -DTR is forced inactive.

Line Status Register (Hex nFD)

This 8-bit read-only register provides the system microprocessor with status information about the data transfer. Writing to this register can produce unpredictable results.

Bit	Function
7	Error in Receiver FIFO Register
6	Transmitter Shift Register Empty
5	Transmitter Holding Register Empty
4	Break Interrupt
3	Framing Error
2	Parity Error
1	Overrun Error
0	Data Ready

Figure 18. Line Status Register (Hex nFD)

Bit 7 This bit is set to 1 when there is at least one parity error, framing error, or break indication in the Receiver FIFO register. If there are no subsequent errors in the Receiver FIFO register, this bit is set to 0 when the system microprocessor reads the Line Status register. In the character mode, this bit is always set to 0.

Bit 6 This bit is set to 1 when the Transmitter Holding register and the Transmitter Shift register are both empty. This bit is set to 0 when either the Transmitter Holding register or the Transmitter Shift register contains a data character.

In the FIFO mode, this bit is set to 1 when the Transmitter FIFO register and the Transmitter Shift register are both empty.

Bit 5 This bit indicates that the serial port controller is ready to accept a new character for transmission. This bit is set to 1 when a character is transferred from the Transmitter Holding register to the Transmitter Shift register. This bit is set to 0 when the system microprocessor loads the Transmitter Holding register.

This bit also causes the controller to issue an interrupt to the system microprocessor when bit 1 in the Interrupt Enable register is set to 1.

In the FIFO mode, this bit is set to 1 when the Transmitter FIFO register is empty. It is set to 0 when at least 1 byte is written to the Transmitter FIFO register.

Bit 4 This bit is set to 1 when the received data input is held in the spacing state for longer than a fullword transmission time (that is, the total time of start bit + data bits + parity + stop bits). This bit is set to 0 when the system microprocessor reads the contents of the Line Status register.

When a break interrupt occurs, only one zero character is loaded into the Receiver FIFO register. The next character is loaded after the receiver serial input changes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a receiver-line-status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

Bit 3 This bit is set to 1 when the stop bit, following the last data bit or parity bit, is at a spacing level. This indicates that the received character did not have a valid stop bit. This bit is set to 0 when the system microprocessor reads the contents of the Line Status register.

Note: In the FIFO mode, the framing error (or parity error for bit 2) is associated with the particular character in the Receiver FIFO register it applies to. The error is indicated to the system microprocessor when its associated character is at the top of the Receiver FIFO register.

Bit 2 This bit is set to 1 when a parity error is detected (the received character does not have the correct even or odd parity, as selected by the even-parity-select bit). This bit is set to 0 when the system microprocessor reads the contents of the Line Status register. See the note in bit 3 for more information.

Bit 1 When set to 1, this bit indicates that data in the Receiver Buffer register was not read by the system microprocessor before the next character was transferred into the Receiver Buffer register, destroying the previous character. This bit is set to 0 when the system microprocessor reads the contents of the Line Status register.

If the FIFO mode data continues to fill the Receiver FIFO register beyond the trigger level, an overrun error will occur only after the Receiver FIFO register is full and the next character has been completely received in the Receiver Shift register. An overrun error is indicated to the system microprocessor when it happens. The character in the Receiver Shift register is overwritten, but it is not transferred to the Receiver FIFO register.

Bit 0 This bit is the receiver data-ready indicator. It is set to 1 when a complete incoming character has been received and transferred into the Receiver Buffer register or the Receiver FIFO register. This bit is set to 0 by reading the Receiver Buffer register or the Receiver FIFO register.

Modem Status Register (Hex nFE)

This 8-bit register provides the current state of the control lines from the modem (or external device) to the system microprocessor. Also, bits 3 through 0 of this register provide change information. These four bits are set to 1 whenever a control input from the modem changes state. They are set to 0 whenever the system microprocessor reads this register.

Bit	Function
7	Data-Carrier-Detect
6	Ring Indicator
5	Data-Set-Ready
4	Clear-to-Send
3	Delta-Data-Carrier-Detect
2	Trailing Edge Ring Indicator
1	Delta-Data-Set-Ready
0	Delta-Clear-to-Send

Figure 19. Modem Status Register (Hex nFE)

Bit 7 This bit is the inverted '-data carrier detect' signal (-DCD) modem control input. If bit 4 of the Modem Control register is set to 1, this bit is equivalent to bit 3 in the Modem Control register.

Bit 6 This bit is the inverted '-ring indicator' signal (-RI) modem control input. If bit 4 of the Modem Control register is set to 1, this bit is equivalent to bit 2 in the Modem Control register.

Bit 5 This bit is the inverted '-data set ready' signal (-DSR) modem control input. If bit 4 of the Modem Control register is set to 1, this bit is equivalent to bit 0 in the Modem Control register.

Bit 4 This bit is the inverted '-clear to send' signal (-CTS) modem control input. If bit 4 of the Modem Control register is set to 1, this bit is equivalent to bit 1 in the Modem Control register.

Bit 3 When set to 1, this bit indicates that the '-data carrier detect' signal (-DCD) modem control input has changed state since the last time it was read by the system microprocessor.

Note: Whenever bit 0, 1, 2, or 3 is set to 1, a modem status interrupt is generated.

Bit 2 When set to 1, this bit indicates that the '-ring indicator' signal (-RI) modem control input has changed from an active condition to an inactive condition.

Bit 1 When set to 1, this bit indicates that the '-data set ready' signal (-DSR) modem control input has changed state since the last time it was read by the system microprocessor.

Bit 0 When set to 1, this bit indicates that the '-clear to send' signal (-CTS) modem control input has changed state since the last time it was read by the system microprocessor.

Scratch Register (Hex nFF)

This register does not control the serial port controller in any way. It is used by the system microprocessor to temporarily hold data.

FIFO Modes of Operation

When in the FIFO mode, the controller can operate in either the interrupt mode or the polled mode.

FIFO Interrupt Mode Operation

When the Receiver FIFO register and the receiver interrupts are enabled (FIFO Control register bit 0 and Interrupt Enable register bit 0 are set to 1), receiver interrupts occur as follows:

1. A received-data-available interrupt is issued to the system microprocessor when the Receiver FIFO register has reached its programmed trigger level; it is cleared as the Receiver FIFO register drops below the programmed trigger level.
2. The Interrupt Identification register's received-data-available indication also occurs when the Receiver FIFO register's trigger level is reached and, like the interrupt, it is cleared when the Receiver FIFO register drops below the trigger level.
3. The receiver-line-status interrupt (Interrupt Identification register = hex 06), has higher priority than the received-data-available interrupt (Interrupt Identification register = hex 04).

4. Bit 0 (data ready) of the Line Status register is set to 1 when a character is transferred from the Receiver Shift register to the Receiver FIFO register. It is set to 0 when the Receiver FIFO register is empty.

When the Receiver FIFO register and receiver interrupts are enabled, register time-out interrupts occur as follows:

1. A FIFO time-out interrupt occurs if the following conditions exist:

    - At least 1 character is in the Receiver FIFO register.

    - The last character was received more than four continuous character times ago (if 2 stop bits are programmed, the second one is included in this time delay).

    - The most recent system microprocessor read of the Receiver FIFO register was longer than four continuous character times ago.

    This causes a maximum character-received to interrupt-issued delay of 160 ms at 300 baud with a 12-bit character.

2. Character times are calculated by using the 'receiver clock' (RCLK) input for a clock signal (this makes the delay proportional to the baud rate).

3. When a time-out interrupt has occurred, it is cleared, and the timer is reset when the system microprocessor reads one character from the Receiver FIFO register.

4. When a time-out interrupt has not occurred, the time-out timer is reset after a new character is received or after the system microprocessor reads the Receiver FIFO register.

When the Transmitter FIFO register and transmitter interrupts are enabled (FIFO Control register bit 0 and Interrupt Enable register bit 1 are set to 1), transmitter interrupts occur as follows:

1. The transmitter-holding-register interrupt (02) occurs when the Transmitter FIFO register is empty; it is cleared when the Transmitter Holding register is written to (1 to 16 characters may be written to the Transmitter FIFO register while this interrupt is being serviced) or the Interrupt Identification register is read.

2. The transmitter-FIFO-register-empty indications are delayed one character time minus the last stop bit time whenever both of the following occur:

- Bit 5 (transmitter holding register empty) of the Line Status register is set to 1.
- There have not been at least 2 bytes in the Transmitter FIFO register at the same time since the last time bit 5 of the Line Status register was set to 1.

The first transmitter interrupt after changing bit 0 in the FIFO Control register is immediate, if enabled.

Character time-out and Receiver FIFO register trigger-level interrupts have the same priority as the current received-data-available interrupt; transmitter-FIFO-register-empty has the same priority as the current transmitter-holding-register-empty interrupt.

FIFO Polled Mode Operation

To put the controller in the FIFO polled mode, set bit 0 of the FIFO Control register to 1 and set bits 0 through 3 of the Interrupt Identification register to 0. The Receiver and Transmitter FIFO registers are controlled separately; either one or both can be in the polled mode of operation.

In the FIFO polled mode of operation, the system microprocessor checks Receiver and Transmitter FIFO register status through the Line Status register:

- Line Status register bit 0 is set to 1, as long as 1 byte is in the Receiver FIFO register.
- Line Status register bits 1 through 4 specify which errors have occurred. Character error status is handled the same way as when in the interrupt mode; the Interrupt Identification register is not affected, because bit 2 of the Interrupt Enable register is set to 0.
- Line Status register bit 5 indicates when the Transmitter FIFO register is empty.
- Line Status register bit 6 indicates that both the Transmitter FIFO register and Transmitter Shift register are empty.
- Line Status register bit 7 indicates if any errors in the Receiver FIFO register.

There is no trigger level reached or time-out condition indicated in the FIFO polled mode; however, the Receiver and Transmitter FIFO registers are still fully capable of holding characters.

Serial Port Controller Programming Considerations

The serial port uses either the Type 1 or Type 2 serial communications controller. The following should be considered when programming the serial controller:

- The Type 1 serial controller does not support the FIFO mode. For more information, refer to the "Interrupt Identification Register (Hex nFA)" on page 7.
- Some systems using the Type 2 controller do not support the FIFO mode. For more information, refer to the system-specfic technical reference manuals.
- The serial port can be configured to either Serial 1 or Serial 2 using the system configuration utilities programs.
- Before changing the Line Control register, make sure the Transmitter Holding register is empty.

See "Compatibility" for additional programming considerations.

Signal Descriptions

Modem-Control Input Signals

The following are input signals from the modem or external device to the controller. Bits 7 through 4 in the Modem Status register indicate the condition of these signals. Bits 3 through 0 monitor these signals to indicate when the modem changes state.

-Clear to Send (-CTS): When active, this signal indicates that the modem is ready for the serial port to transmit data.

-Data Set Ready (-DSR): When active, this signal indicates that the modem or data set is ready to establish the communications link and transfer data with the controller.

-Ring Indicator (-RI): When active, this signal indicates that the modem or data set detected a telephone ringing signal.

-Data Carrier Detect (-DCD): When active, this signal indicates that the modem or data set detected a data carrier.

Modem-Control Output Signals

The following are controller output signals. All are set inactive by a master reset operation. These signals are controlled by bits 3 through 0 in the Modem Control register.

-Data Terminal Ready (-DTR): When active, this signal informs the modem or data set that the controller is ready to communicate.

-Request to Send (-RTS): When active, this signal informs the modem or data set that the controller is ready to send data.

-Output 1 (-OUT 1): This signal is pulled high.

-Output 2 (-OUT 2): This is a user-designated output. This signal controls interrupts to the system.

Voltage Interchange Information

The signal is considered in the *marking* condition when the voltage on the interchange circuit, measured at the interface point, is more negative than -3 Vdc with respect to signal ground. The signal is considered in the *spacing* condition when the voltage is more positive than +3 Vdc with respect to signal ground. The region between +3 Vdc and -3 Vdc is defined as the transition region and is considered an invalid level. Voltage that is more negative than -15 Vdc or more positive than +15 Vdc is also considered an invalid level.

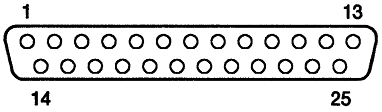
Interchange Voltage	**Binary State**	**Signal Condition**	**Interface Control Function**
Positive Voltage	Binary 0	Spacing	On
Negative Voltage	Binary 1	Marking	Off

Figure 20. Voltage Levels

Connector

The hardware interface uses the standard D-shell connector and pin assignments defined for RS-232C. The voltage levels are EIA only. Current loop interface is not supported.

The following figure shows the pin configuration and signal assignments for the serial port in a communications environment.



Pin No.	**I/O**	**Signal Name**	**Pin No.**	**I/O**	**Signal Name**
1	N/A	Not Connected	14	N/A	Not Connected
2	O	Transmit Data	15	N/A	Not Connected
3	I	Receive Data	16	N/A	Not Connected
4	O	Request to Send	17	N/A	Not Connected
5	I	Clear to Send	18	N/A	Not Connected
6	I	Data Set Ready	19	N/A	Not Connected
7	N/A	Signal Ground	20	O	Data Terminal Ready
8	I	Data Carrier Detect	21	N/A	Not Connected
9	N/A	Not Connected	22	I	Ring Indicator
10	N/A	Not Connected	23	N/A	Not Connected
11	N/A	Not Connected	24	N/A	Not Connected
12	N/A	Not Connected	25	N/A	Not Connected
13	N/A	Not Connected			

Figure 21. Serial Port Connector Signal and Pin Assignments

Index

A

applications 2
asynchronous communications 1

B

baud-rate generator 3, 6
block diagram 2
break interrupt 15

C

communications applications 2
compatibility 21
connector 23
considerations, programming 21

D

data format 3
data speed 6
descriptions, signal 21
diagnostic capabilities 13
diagnostic mode 13
divisor latch access bit (DLAB) 11
divisor latch registers 5
DLAB (divisor latch access bit) 4, 11

F

FIFO control register 9
FIFO modes 18
framing error 15

I

interface 23
interrupt enable register 6
interrupt identification register 7
interrupts 6
interrupts, modem control 13

L

line control register 11
line status register 14

M

marking condition, signal 22
modem control register 12
modem status register 17

O

output port 1 (Serial 1) 2
output port 2 (Serial 2) 2
overrun error 16

P

parity error 15
pin assignments 23
programming considerations 21

R

receiver buffer register 5

S
scratch register 18
signal descriptions 21
spacing condition, signal 22
support 1

T
transition region, signal 22
transmitter holding register 4

V
voltage interchange information 22
voltages 23

W
word length 12

Notes:

Parallel Port Controller (Type 1)

Description 1
Parallel Port Programmable Option Select 2
Parallel Port Extended Mode 2
Parallel Port Controller Programming Considerations 2
Data Port 3
Status Port 4
Parallel Control Port 5
Parallel Port Timing 6
Signal Descriptions 7
Connector 8

Figures

1. Parallel Port Controller 1
2. Parallel Port Address Assignments 2
3. Data Port 3
4. Status Port 4
5. Parallel Control Port 5
6. Parallel-Port Timing Sequence 6
7. Data and Interrupt Signals 7
8. Control Signals 7
9. Parallel Port Connector Signal and Pin Assignments 8

Description

The parallel port allows the attachment of devices that transfer 8 bits of parallel data at standard transistor-transistor levels. It has a 25-pin, D-shell connector. The primary function of the parallel port is to attach a printer with a parallel interface to the system.

This port may be addressed as parallel port 1, 2, or 3, and is compatible with IBM Personal Computer parallel port implementations. The parallel port has an extended mode that supports bidirectional input and output. The port also supports level-sensitive interrupts and a readable interrupt-pending status.

The following figure is a block diagram of the parallel port controller.

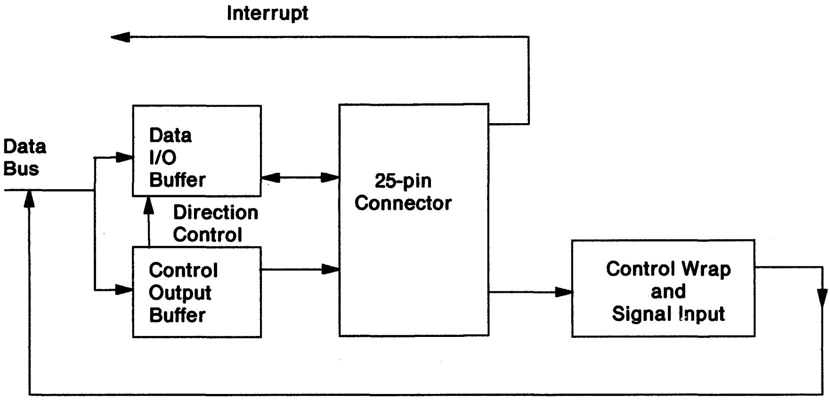


Figure 1. Parallel Port Controller

Parallel Port Programmable Option Select

The parallel port can be configured to the same three address spaces used by IBM Personal Computer products. These addresses are selected through Programmable Option Select (POS) registers during system board setup.

The address assignments for each configuration are shown in the following figure.

	Data Port (Hex Address)	**Status Port (Hex Address)**	**Control Port (Hex Address)**	**Reserved Address**	**IRQ**
Parallel 1	03BC	03BD	03BE	03BF	7
Parallel 2	0378	0379	037A	037B	7
Parallel 3	0278	0279	027A	027B	7

Figure 2. Parallel Port Address Assignments

Parallel Port Extended Mode

The extended mode option is selected through the POS function during system board setup. The extended mode makes the parallel port an 8-bit parallel bidirectional interface. Direction is determined by bit 5 of the Parallel Control port. See "Parallel Control Port" on page 5.

Parallel Port Controller Programming Considerations

The following are some considerations for programming the parallel port controller.

The interface responds to five I/O instructions: two output and three input. In the compatible mode, the output instructions transfer data into two latches whose output is presented on the pins of the D-shell connector. In the extended mode, the 8-bit data latch output to the D-shell connector is controlled by bit 5 in the Parallel Control port.

In the compatible mode, two of the three input instructions allow the system microprocessor to read back the contents of the two latches.

In the extended mode, the read-back of the 8-bit data in the Data Address is controlled by bit 5 in the Parallel Control port. The third input instruction allows the system microprocessor to read the real-time status of a group of pins on the connector.

The extended mode can be used by externally attached equipment.

During the power-on self test (POST), the parallel port is configured as an output port. POST status information is written to this port during the power-on initialization or the initialization caused by a reset from the keyboard (Ctrl, Alt, Del).

The following is a detailed description of each interface-port instruction.

Data Port

The Data port is the 8-bit data port for both the compatible and extended modes. In the compatible mode, a write operation to this port immediately presents data to the connector pins. In the compatible mode, a read operation from this port produces the last data written to it.

In the extended mode, a write operation to this port latches the data, but the data is only presented to the connector pins if the direction bit was set to 0 (Write) in the Parallel Control port. A read operation in the extended mode produces either:

- The data previously written if the direction bit in the Parallel Control port is set to 0 (Write).
- The data on the connector pins from another device if the direction bit in the Parallel Control port is set to 1 (Read).

Bit	Port Data
7 - 0	Data

Figure 3. Data Port

Bits 7 - 0 These bits represent the 'data' (D7 - D0) signal lines.

Status Port

The Status port is a read-only port in both modes. A read operation to this port presents the system microprocessor with the interrupt-pending status of the interface, and the real-time status of the connector pins, as shown in the following figure. An interrupt is pending when bit 2 (-IRQ STATUS) is set to 0.

Bit	Port Data
7	-BUSY
6	-ACK
5	PE
4	SLCT
3	-ERROR
2	-IRQ STATUS
1, 0	Reserved

Figure 4. Status Port

Bit 7 This bit represents the state of the '-busy' signal (-BUSY). When this signal is active, the printer is busy and cannot accept data.

Bit 6 This bit represents the current state of the printer '-acknowledge' signal (-ACK) . When this bit is set to 0, the printer has received a character and is ready to accept another.

Bit 5 This bit represents the current state of the printer 'paper end' signal (PE). When this bit is set to 1, the printer has detected the end of the paper.

Bit 4 This bit represents the current state of the 'select' signal (SLCT). When this bit is set to 1, the printer has been selected.

Bit 3 This bit represents the current state of the printer '-error' signal (-ERROR). When this bit is set to 0, the printer has encountered an error condition.

Bit 2 When this bit is set to 0, the printer has acknowledged the previous transfer using the '-acknowledge' signal. An interrupt is pending when bit 2 (-IRQ STATUS) is set to 0.

Bits 1, 0 These bits are reserved.

Parallel Control Port

The Parallel Control port is a read/write port. A write operation to this port latches bits 0 through 5 of the bus. Bit 5 is the direction control bit used in the extended mode only. A read operation to the Parallel Control port presents the system microprocessor the data that was last written to it, except for the write-only direction bit.

Bit	Port Data
7, 6	Reserved = 0
5	Direction
4	IRQ EN
3	Pin 17 (SLCT IN)
2	Pin 16 (-INIT)
1	Pin 14 (AUTO FD XT)
0	Pin 1 (STROBE)

Figure 5. Parallel Control Port

Bits 7, 6 These bits are reserved and must be set to 0.

Bit 5 This write-only bit controls the direction of the data port. When this bit is set to 0, the data port is written to. When this bit is set to 1, the data port is read from.

Bit 4 This bit enables the parallel port interrupt. When this bit is set to 1, an interrupt occurs when the '-acknowledge' signal changes from active to inactive.

Bit 3 This bit controls the 'select in' signal (SLCT IN). When this bit is set to 1, the printer is selected.

Bit 2 This bit controls the 'initialize printer' signal (-INIT). When this bit is set to 0, the printer starts.

Bit 1 This bit controls the 'automatic feed XT' signal (AUTO FD XT). When this bit is set to 1, the printer automatically spaces the paper up one line for every line return.

Bit 0 This bit controls the 'strobe' signal (STROBE) to the printer. When this bit is set to 1, data is clocked into the printer.

Parallel Port Timing

Timing for the parallel port depends on the devices connected to the port. The following figure shows the sequence for typical parallel-port signal timing.

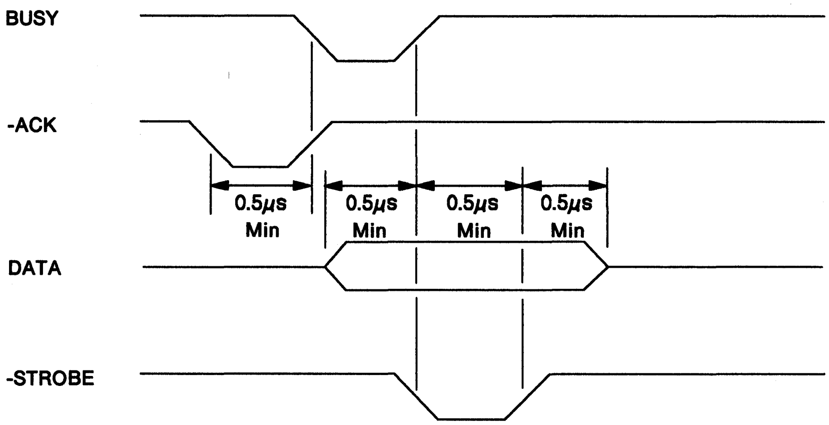


Figure 6. Parallel-Port Timing Sequence

For specific signal timing parameters, refer to the specifications for the equipment connected to the parallel port connector.

Signal Descriptions

The following figures list characteristics of the signals.

Sink Current	24 mA	Maximum
Source Current	15 mA	Maximum
High-Level Output Voltage	2.4 Vdc	Minimum
Low-Level Output Voltage	0.5 Vdc	Maximum

Figure 7. Data and Interrupt Signals

Pins 1, 14, 16, and 17 are driven by open collector drivers pulled to 5 Vdc through 4.7 kilohm resistors.

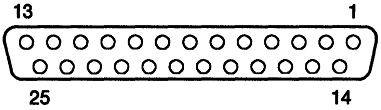
Sink Current	20 mA	Maximum
Source Current	0.55 mA	Maximum
High-Level Output Voltage	5.0 Vdc minus pullup	Minimum
Low-Level Output Voltage	0.5 Vdc	Maximum

Figure 8. Control Signals

Connector

The parallel port connector is a standard 25-pin, D-shell connector. The data lines on the connector are driven by drivers capable of sourcing 15 milliamps and sinking 24 milliamps.

The following figure shows the signal and pin assignments for the parallel port controller connector.



Pin No.	I/O	Signal Name	Pin No.	I/O	Signal Name
1	I/O	-STROBE	14	O	-AUTO FD XT
2	I/O	Data 0	15	I	-ERROR
3	I/O	Data 1	16	O	-INIT
4	I/O	Data 2	17	O	-SLCT IN
5	I/O	Data 3	18	NA	Ground
6	I/O	Data 4	19	NA	Ground
7	I/O	Data 5	20	NA	Ground
8	I/O	Data 6	21	NA	Ground
9	I/O	Data 7	22	NA	Ground
10	I	-ACK	23	NA	Ground
11	I	BUSY	24	NA	Ground
12	I	PE	25	NA	Ground
13	I	SLCT			

Figure 9. Parallel Port Connector Signal and Pin Assignments

Video Subsystem (Type 1)

Description 1
Major Components 4
ROM BIOS 4
Support Logic 4
Video Graphics Array Components 5
CRT Controller 5
Sequencer 5
Graphics Controller 5
Attribute Controller 7
Modes of Operation 8
Display Support 10
Programmable Option Select 10
Alphanumeric Modes 11
Graphics Modes 14
320 x 200 Four-Color Graphics (Modes Hex 4 and 5) 14
640 x 200 Two-Color Graphics (Mode Hex 6) 16
640 x 350 Graphics (Mode Hex F) 17
640 x 480 Two-Color Graphics (Mode Hex 11) 18
16-Color Graphics Modes (Mode Hex 10, D, E, and 12) . . . 18
256-Color Graphics Mode (Mode Hex 13) 18
Video Memory Organization 20
Memory Modes 20
Modes Hex 0, 1 21
Modes Hex 2, 3 22
Modes Hex 4, 5 23
Mode Hex 6 24
Mode Hex 7 25
Mode Hex D 26
Mode Hex E 28
Mode Hex F 30
Mode Hex 10 31
Mode Hex 11 32
Mode Hex 12 33
Mode Hex 13 34
Memory Operations 35
Write Operations 35
Read Operations 36
Registers 37
General Registers 38

Miscellaneous Output Register 38
Input Status Register 0 40
Input Status Register 1 40
Feature Control Register 41
Video Subsystem Enable Register 41
Sequencer Registers 42
Sequencer Address Register 42
Reset Register 43
Clocking Mode Register 43
Map Mask Register 45
Character Map Select Register 46
Memory Mode Register 48
CRT Controller Registers 49
Address Register 50
Horizontal Total Register 50
Horizontal Display-Enable End Register 51
Start Horizontal Blanking Register 51
End Horizontal Blanking Register 51
Start Horizontal Retrace Pulse Register 53
End Horizontal Retrace Register 53
Vertical Total Register 54
Overflow Register 55
Preset Row Scan Register 56
Maximum Scan Line Register 57
Cursor Start Register 58
Cursor End Register 58
Start Address High Register 59
Start Address Low Register 59
Cursor Location High Register 59
Cursor Location Low Register 60
Vertical Retrace Start Register 60
Vertical Retrace End Register 60
Vertical Display Enable End Register 61
Offset Register 62
Underline Location Register 62
Start Vertical Blanking Register 63
End Vertical Blanking Register 63
CRT Mode Control Register 64
Line Compare Register 67
Graphics Controller Registers 68
Address Register 68
Set/Reset Register 69
Enable Set/Reset Register 69

Color Compare Register 70
Data Rotate Register 71
Read Map Select Register 72
Graphics Mode Register 72
Miscellaneous Register 74
Color Don't Care Register 75
Bit Mask Register 75
Attribute Controller Registers 76
Address Register 76
Internal Palette Registers 0 through F 77
Attribute Mode Control Register 78
Overscan Color Register 80
Color Plane Enable Register 80
Horizontal PEL Panning Register 81
Color Select Register 82
VGA Programming Considerations 83
Programming the Registers 86
RAM Loadable Character Generator 87
Creating a Split Screen 88
Video Digital-to-Analog Converter 90
Device Operation 90
Video DAC to System Interface 91
Programming Considerations 92
Auxiliary Video Connector 94
Signal Descriptions 96
Connector Timing 98
Display Connector 99
Signal Timing 100

Index 107

Figures

1. Video Subsystem 3
2. Graphics Controller 6
3. Attribute Controller 7
4. BIOS Video Modes 8
5. Double Scanning and Border Support 9
6. IBM Direct-Drive Analog Displays 10
7. Character/Attribute Format 12
8. Attribute Byte Definitions 12
9. BIOS Color Set 13
10. Video Memory Format 14
11. PEL Format, Modes Hex 4 and 5 15
12. Color Selections, Modes Hex 4 and 5 15
13. PEL Format, Mode Hex 6 16
14. Bit Definitions C2,C0 17
15. Compatible Color Coding 19
16. 256KB Video Memory Map 20
17. Data Flow for Write Operations 35
18. Color Compare Operations 36
19. Video Subsystem Register Overview 37
20. General Registers 38
21. Miscellaneous Output Register, Hex 03CC/03C2 38
22. Display Vertical Size 39
23. Clock Select Definitions 39
24. Input Status Register 0, Hex 03C2 40
25. Input Status Register 1, Hex 03DA/03BA 40
26. Feature Control Register, Hex 03DA/03BA and 03CA 41
27. Video Subsystem Enable Register, Hex 03C4 41
28. Sequencer Registers 42
29. Sequencer Address Register 42
30. Reset Register, Index Hex 00 43
31. Clocking Mode Register, Index Hex 01 43
32. Map Mask Register, Index Hex 02 45
33. Character Map Select Register, Index Hex 03 46
34. Character Map Select A 47
35. Character Map Select B 47
36. Memory Mode Register, Index Hex 04 48
37. Map Selection, Chain 4 48
38. CRT Controller Registers 49
39. CRT Controller Address Register, Hex 03B4/03D4 50

40. Horizontal Total Register, Index Hex 00 50
41. Horizontal Display Enable-End Register, Index Hex 01 . . . 51
42. Start Horizontal Blanking Register, Index Hex 02 51
43. End Horizontal Blanking Register, Index Hex 03 51
44. Display Enable Skew 52
45. Start Horizontal Retrace Pulse Register, Index Hex 04 . . . 53
46. End Horizontal Retrace Register, Index Hex 05 53
47. Vertical Total Register, Index Hex 06 54
48. CRT Overflow Register, Index Hex 07 55
49. Preset Row Scan Register, Index Hex 08 56
50. Maximum Scan Line Register, Index Hex 09 57
51. Cursor Start Register, Index Hex 0A 58
52. Cursor End Register, Index Hex 0B 58
53. Start Address High Register, Index Hex 0C 59
54. Start Address Low Register, Index Hex 0D 59
55. Cursor Location High Register, Index Hex 0E 59
56. Cursor Location Low Register, Index Hex 0F 60
57. Vertical Retrace Start Register, Index Hex 10 60
58. Vertical Retrace End Register, Index Hex 11 60
59. Vertical Display Enable End Register, Index Hex 12 61
60. Offset Register, Index Hex 13 62
61. Underline Location Register, Index Hex 14 62
62. Start Vertical Blanking Register, Index Hex 15 63
63. End Vertical Blanking Register, Index Hex 16 63
64. CRT Mode Control Register, Index Hex 17 64
65. CRT Memory Address Mapping 65
66. Line Compare Register, Index Hex 18 67
67. Graphics Controller Register Overview 68
68. Graphics Controller Address Register, Hex 03CE 68
69. Set/Reset Register, Index Hex 00 69
70. Enable Set/Reset Register, Index Hex 01 69
71. Color Compare Register, Index Hex 02 70
72. Data Rotate Register, Index Hex 03 71
73. Operation Select Bit Definitions 71
74. Read Map Select Register, Index Hex 04 72
75. Graphics Mode Register, Index Hex 05 72
76. Write Mode Definitions 73
77. Miscellaneous Register, Index Hex 06 74
78. Video Memory Assignments 74
79. Color Don't Care Register, Index Hex 07 75
80. Bit Mask Register, Index Hex 08 75
81. Attribute Controller Register Addresses 76
82. Address Register, Hex 03C0 76

83. Internal Palette Registers, Index Hex 00 - 0F 77
84. Attribute Mode Control Register, Index Hex 10 78
85. Overscan Color Register, Index Hex 11 80
86. Color Plane Enable Register, Index Hex 12 80
87. Horizontal PEL Panning Register, Index Hex 13 81
88. Image Shifting 81
89. Color Select Register, Index Hex 14 82
90. Character Table Structure 87
91. Character Pattern Example 88
92. Split Screen Definition 88
93. Screen Mapping within the Display Buffer Address Space . 89
94. Video DAC Register 90
95. Auxiliary Video Connector 95
96. Auxiliary Video Connector Timing (DAC Signals) 98
97. Display Connector 99
98. Display Connector Signals 99
99. Vertical Size 100
100. Vertical Timing, 350 Lines 101
101. Vertical Timing, 400 Lines 102
102. Vertical Timing, 480 Lines 103
103. Horizontal Timing, 80 Column with Border 104
104. Horizontal Timing, 40/80 Column, without Border 105

Description

System video is generated by the IBM Video Graphics Array (VGA) and its associated circuitry, which consists of a video buffer, a video digital-to-analog converter (DAC), and test circuitry. The 256KB video memory is mapped as four planes of 64Kb by 8 bits (maps 0 through 3). The video DAC drives the analog output to the display connector. The test circuitry is used to test for the type of display attached, color or monochrome.

The video subsystem supports all video modes available on the IBM Monochrome Display Adapter, IBM Color/Graphics Monitor Adapter, and IBM Enhanced Graphics Adapter. When a monochrome display is attached, the colors for the color modes appear as shades of gray.

The new modes available are:

- 640 x 480 16- and 2-color graphics
- 720 x 400 16-color and monochrome alphanumeric
- 360 x 400 16-color alphanumeric
- 320 x 200 256-color graphics.

In the 200-scan-line modes, the data for each scan line is scanned twice. This double scanning allows the 200-scan-line image to be displayed as 400 scan lines.

The video subsystem serves as the interface between the system microprocessor and video memory. When the system microprocessor writes to or reads from video memory, all data passes through the video subsystem.

The video subsystem controls the access to video memory from the system and the cathode-ray tube (CRT) controller. Therefore, programs do not need to wait for horizontal retrace to update the display buffer in order to preserve screen appearance. The system performs better when accessing the display buffer during nonactive display times because there is less interference from the CRT controller.

The video subsystem also controls the system addresses assigned to video memory. Up to three different starting addresses can be programmed for compatibility with previous video adapters.

In the graphics modes, the mode determines the way video information is formatted into memory, and the way memory is organized.

In alphanumeric modes, the system writes the ASCII character code and attribute data to video memory maps 0 and 1, respectively. Memory map 2 contains the character font loaded by BIOS during an alphanumeric mode set. The font is used by the character generator to create the character image on the display.

Three fonts are contained in ROM. Two of these fonts have dot patterns that are the same as previous IBM display adapters. The third font is a new 8-by-16 character font. Up to eight 256-character fonts can be loaded into video memory map 2 at one time; two fonts can be active at any one time, allowing a 512-character font.

The video subsystem formats the information in video memory and sends the output to the video DAC. For color displays, the video DAC sends three analog color signals (red, green, and blue) to the display connector. For monochrome displays, BIOS translates the color information in the DAC, and the DAC drives the summed signal onto the green output.

The auxiliary video connector allows video data to be passed between the video subsystem and an adapter plugged into the channel connector. The video subsystem can be disabled through the POS registers. When it is disabled, the video subsystem will not respond to video memory or I/O reads or writes, and the video from the adapter can directly drive the video DAC.

Note: Compatibility with other hardware is best achieved by using the BIOS interface or operating system interface whenever possible.

The following is a block diagram of the video subsystem, which is part of the system board.

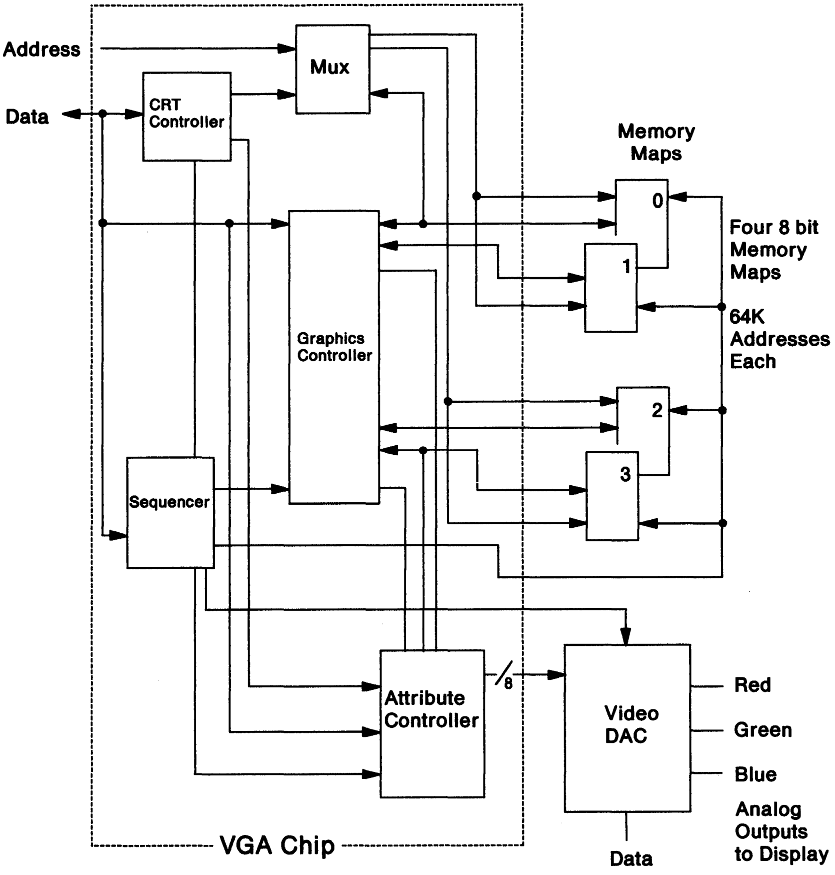


Figure 1. Video Subsystem

Major Components

Most of the logic for the video subsystem is contained in one module, the video graphics array (VGA). This module contains all circuits necessary to generate the timing for the video memory, and generates the video information going to the video DAC. The major components are: ROM BIOS, the support logic, and the VGA.

ROM BIOS

Software support is provided by BIOS on the system board. BIOS contains the character fonts and the system interface to run the video subsystem.

Support Logic

The support logic consists of the video memory, the clocks, and the video DAC. The video memory consists of 256KB and its use and mapping depend on the mode selected.

Two clock sources (25.175 MHz and 28.322 MHz) provide the dot rate. The clock source is selected in the Miscellaneous Output register.

The video DAC contains the color palette that is used to convert the video data into the video signal sent to the display. Three analog signals (red, green, blue) are output from the DAC.

The maximum number of colors displayed is 16 out of 256K (K equals 1024), except for mode hex 13, which can display 256 colors. The maximum number of gray shades is 16 out of 64, except for mode hex 13, which can display all 64 shades.

Video Graphics Array Components

The VGA has four major functional areas: the CRT controller, the sequencer, the graphics controller, and the attribute controller.

CRT Controller

The CRT controller generates horizontal and vertical synchronization signal timings, addressing for the regenerative buffer, cursor and underline timings, and refresh addressing for the video memory.

Sequencer

The sequencer generates basic memory timings for the video memory and the character clock for controlling regenerative buffer fetches. It allows the system to access memory during active display intervals by periodically inserting dedicated system microprocessor memory cycles between the display memory cycles. Map mask registers in the sequencer are available to protect entire memory maps from being changed.

Graphics Controller

The graphics controller is the interface between the video memory and the attribute controller during active display times, and between video memory and the system microprocessor during memory accesses.

During active display times, memory data is latched and sent to the attribute controller. In graphics modes, the memory data is converted from parallel to serial bit-plane data before being sent; in alphanumeric modes, the parallel attribute data is sent.

During system accesses of video memory, the graphics controller can perform logical operations on the memory data before it reaches video memory or the system data bus. These logical operations are composed of four logical write modes and two logical read modes. The logical operators allow enhanced operations, such as a color compare in the read mode, individual bit masking during write modes, internal 32-bit writes in a single memory cycle, and writing to the display buffer on nonbyte boundaries.

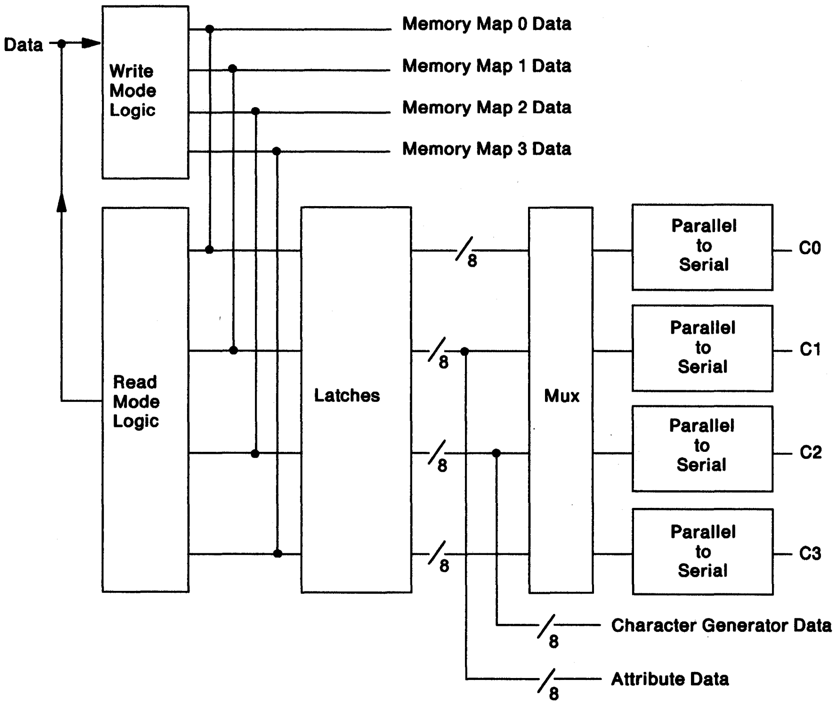


Figure 2. Graphics Controller

Attribute Controller

The attribute controller takes in data from video memory through the graphics controller and formats it for display. Attribute data in alphanumeric mode and serialized bit-plane data in graphics mode are converted to an 8-bit color value.

Each color value is selected from an internal color palette of 64 possible colors (except in 256-color mode). The color value is used as a pointer into the video DAC where it is converted to the analog signals that drive the display.

Blinking, underlining, cursor insertion, and PEL panning are also controlled in the attribute controller.

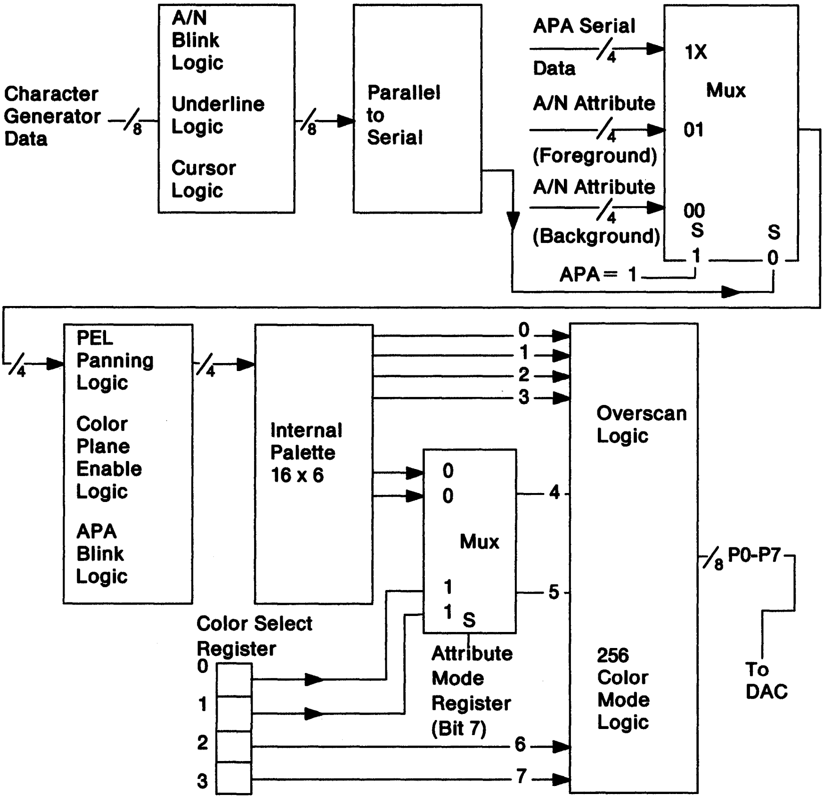


Figure 3. Attribute Controller

Modes of Operation

Certain modes on previous IBM display adapters distinguished between monochrome and color displays. For example, mode 0 was the same as mode 1 with the color burst turned off. Because color burst is not supported by the PS/2 video, the mode pairs are exactly the same. The support logic for VGA recognizes the type of display, and adjusts the output accordingly.

Mode 3+ is the default mode with a color display attached and mode 7+ is the default mode with a monochrome display attached.

The following figure describes the alphanumeric (A/N) and all points addressable (APA) graphics modes supported by BIOS. Each color is selected from 256K possibilities, and gray shades from 64 possibilities. The variations within the basic BIOS modes are selected through BIOS calls that set the number of scan lines. The scan line count is set before the mode call is made.

Mode (Hex)	**Type**	**Colors**	**Alpha Format**	**Buffer Start**	**Box Size**	**Max. Pgs.**	**Freq.**	**Vert. PELs**
0, 1	A/N	16	40 x 25	B8000	8 x 8	8	70 Hz	320 x 200
0*, 1*	A/N	16	40 x 25	B8000	8 x 14	8	70 Hz	320 x 350
0+, 1+	A/N	16	40 x 25	B8000	9 x 16	8	70 Hz	360 x 400
2, 3	A/N	16	80 x 25	B8000	8 x 8	8	70 Hz	640 x 200
2*, 3*	A/N	16	80 x 25	B8000	8 x 14	8	70 Hz	640 x 350
2+, 3+	A/N	16	80 x 25	B8000	9 x 16	8	70 Hz	720 x 400
4, 5	APA	4	40 x 25	B8000	8 x 8	1	70 Hz	320 x 200
6	APA	2	80 x 25	B8000	8 x 8	1	70 Hz	640 x 200
7	A/N	-	80 x 25	B0000	9 x 14	8	70 Hz	720 x 350
7+	A/N	-	80 x 25	B0000	9 x 16	8	70 Hz	720 x 400
D	APA	16	40 x 25	A0000	8 x 8	8	70 Hz	320 x 200
E	APA	16	80 x 25	A0000	8 x 8	4	70 Hz	640 x 200
F	APA	-	80 x 25	A0000	8 x 14	2	70 Hz	640 x 350
10	APA	16	80 x 25	A0000	8 x 14	2	70 Hz	640 x 350
11	APA	2	80 x 30	A0000	8 x 16	1	60 Hz	640 x 480
12	APA	16	80 x 30	A0000	8 x 16	1	60 Hz	640 x 480
13	APA	256	40 x 25	A0000	8 x 8	1	70 Hz	320 x 200

* Enhanced modes from the IBM Enhanced Graphics Adapter.
+ Enhanced modes

Figure 4. BIOS Video Modes

Border support and double scanning depend on the mode selected. The following shows which modes use double scanning and which support a border.

Mode (Hex)	**Double Scan**	**Border Support**
0, 1	Yes	No
0*, 1*	No	No
0+, 1+	No	No
2, 3	Yes	Yes
2*, 3*	No	Yes
2+, 3+	No	Yes
4, 5	Yes	No
6	Yes	Yes
7	No	Yes
7+	No	Yes
D	Yes	No
E	Yes	Yes
F	No	Yes
10	No	Yes
11	No	Yes
12	No	Yes
13	Yes	Yes

Figure 5. Double Scanning and Border Support

Display Support

The video subsystem supports direct-drive analog displays. The displays must have a horizontal sweep frequency of 31.5 kHz, and a vertical sweep frequency capability of 50 to 70 Hz. Displays that use a digital input, such as the IBM Color Display, are *not* supported. The following figure summarizes the display characteristics.

Parameter	Color	Monochrome
Horizontal Scan Rate	31.5 kHz	31.5 kHz
Vertical Scan Rate	50 to 70 Hz	50 to 70 Hz
Video Bandwidth	28 MHz	28 MHz
Maximum Horizontal Resolution	720 PELs	720 PELs
Maximum Vertical Resolution	480 PELs	480 PELs

Figure 6. IBM Direct-Drive Analog Displays

Since the color and monochrome displays run at the same sweep rate, all modes work on both displays. The vertical gain of the display is controlled by the polarity of the vertical and horizontal synchronization pulses. This is done so 350, 400, or 480 lines can be displayed without adjusting the display. See "Signal Timing" on page 100 for more information.

Programmable Option Select

The video subsystem supports programmable option select (POS). The video subsystem is placed in the setup mode through bit 5 of the System Board Enable/Setup register (hex 0094).

While the video subsystem is in the setup mode, only POS Register 2 (hex 0102) is used; bit 0 of this register is the video enable bit. When this bit is set to 0, the video subsystem does not respond to commands, addresses, or data. If video output is being generated when the video enable bit is set to 0, the output is still generated. For information on BIOS calls to enable or disable the video, see the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

Note: Whenever the video subsystem is in setup mode, access to the video DAC registers is disabled.

Alphanumeric Modes

The alphanumeric modes are modes 0 through 3 and 7. The mode chart lists the variations of these modes. The data format for alphanumeric modes is the same as the data format on the IBM Color/Graphics Monitor Adapter, the IBM Monochrome Display Adapter, and the IBM Enhanced Graphics Adapter.

BIOS initializes the video subsystem according to the selected mode and loads the color values into the video DAC. These color values can be changed to give a different color set to select from. Bit 3 of the attribute byte may be redefined by the Character Map Select register to act as a switch between character sets, giving the programmer access to 512 characters at one time.

When an alphanumeric mode is selected, the BIOS transfers character font patterns from the ROM to map 2. The system stores the character data in map 0, and the attribute data in map 1. In the alphanumeric modes, the programmer views maps 0 and 1 as a single buffer. The CRT controller generates sequential addresses, and fetches one character code byte and one attribute byte at a time. The character code and row scan count are combined to make up the address into map 2, which contains the character font. The appropriate dot patterns are then sent to the attribute controller, where color is assigned according to the attribute data.

Every display-character position in the alphanumeric mode is defined by two bytes in the display buffer. Both the color/graphics and the monochrome emulation modes use the following 2-byte character/attribute format.

Display Character Code Byte								Attribute Byte							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Even Address								Odd Address							

Figure 7. Character/Attribute Format

See "Characters and Keystrokes" for characters loaded during a BIOS mode set.

The functions of the attribute byte are defined in the following table. Bit 7 can be redefined in the Attribute Mode Control register to give 16 possible background colors; its default is to control character blinking. Bit 3 can be redefined in the Character Map Select register to select between two character fonts; its default is to control foreground color selection.

Bit	Color	Function
7	B/I	Blinking or Background Intensity
6	R	Background Color
5	G	Background Color
4	B	Background Color
3	I/CS	Foreground Intensity or Character Font Select
2	R	Foreground Color
1	G	Foreground Color
0	B	Foreground Color

Figure 8. Attribute Byte Definitions

For more information about the attribute byte, see "Character Map Select Register" on page 46 and "Attribute Mode Control Register" on page 78.

The following are the color values loaded by BIOS for the 16-color modes.

I	R	G	B	Color
0	0	0	0	Black
0	0	0	1	Blue
0	0	1	0	Green
0	0	1	1	Cyan
0	1	0	0	Red
0	1	0	1	Magenta
0	1	1	0	Brown
0	1	1	1	White
1	0	0	0	Gray
1	0	0	1	Light Blue
1	0	1	0	Light Green
1	0	1	1	Light Cyan
1	1	0	0	Light Red
1	1	0	1	Light Magenta
1	1	1	0	Yellow
1	1	1	1	White (High Intensity)

Figure 9. BIOS Color Set

Both 40-column and 80-column alphanumeric modes are supported. The features of the 40-column alphanumeric modes (all variations of modes hex 0 and 1) are:

- 25 rows of 40 characters
- 2,000 bytes of video memory per page
- One character byte and one attribute byte per character.

The features of the 80-column alphanumeric modes (all variations of modes hex 2, 3, and 7) are:

- 25 rows of 80 characters
- 4,000 bytes of video memory per page
- One character byte and one attribute byte per character.

Graphics Modes

This section describes the graphics modes supported in BIOS. The colors in this section are generated when the BIOS is used to set the mode. BIOS initializes the video subsystem and the DAC palette to generate these colors. If the DAC palette is changed, different colors are generated.

320 x 200 Four-Color Graphics (Modes Hex 4 and 5)

Addressing, mapping, and data format are the same as the 320 x 200 PEL mode of the IBM Color/Graphics Monitor Adapter. The display buffer is configured at hex B8000. Bit image data is stored in memory maps 0 and 1. The two bit planes (C0 and C1) are each formed from bits from both memory maps.

Features of this mode are:

- A maximum of 200 rows of 320 PELs,
- Double-scanned to display as 400 rows
- Memory-mapped graphics
- Four colors for each PEL
- Four PELs per byte
- 16,000 bytes of read/write memory.

The video memory is organized into two banks of 8,000 bytes each using the following format. Address hex B8000 contains the PEL information for the upper-left corner of the display area.

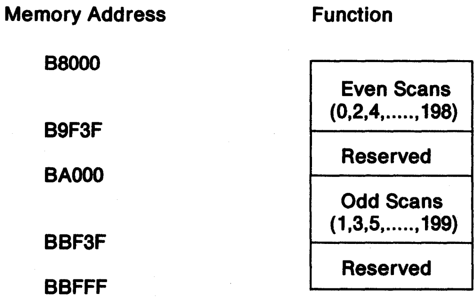


Figure 10. Video Memory Format

The following figure shows the format for each byte.

Bit	Function
7	C1 - First Display PEL
6	C0 - First Display PEL
5	C1 - Second Display PEL
4	C0 - Second Display PEL
3	C1 - Third Display PEL
2	C0 - Third Display PEL
1	C1 - Fourth Display PEL
0	C0 - Fourth Display PEL

Figure 11. PEL Format, Modes Hex 4 and 5

The color selected depends on the color set that is used. Color set 1 is the default. For information on changing the color set, see the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference*.

Bits	Color Selected	
C1 C0	Color Set 1	Color Set 0
0 0	Black	Black
0 1	Light Cyan	Green
1 0	Light Magenta	Red
1 1	Intensified White	Brown

Figure 12. Color Selections, Modes Hex 4 and 5

640 x 200 Two-Color Graphics (Mode Hex 6)

Addressing, scan-line mapping, and data format are the same as the 640 x 200 PEL black and white mode of the IBM Color/Graphics Monitor Adapter. The display buffer is configured at hex B8000. Bit image data is stored in memory map 0 and comprises a single bit plane (C0). Features of this mode are:

- A maximum of 200 rows of 640 PELs
- Double scanned to display as 400 rows
- Same addressing and scan-line mapping as 320 x 200 graphics
- Two colors for each PEL
- Eight PELs per byte
- 16,000 bytes of read/write memory.

The following shows the format for each byte.

Bit	Function
7	First Display PEL
6	Second Display PEL
5	Third Display PEL
4	Fourth Display PEL
3	Fifth Display PEL
2	Sixth Display PEL
1	Seventh Display PEL
0	Eighth Display PEL

Figure 13. PEL Format, Mode Hex 6

The bit definition for each PEL is 0 equals black and 1 equals intensified white.

640 x 350 Graphics (Mode Hex F)

This mode emulates the EGA graphics with the monochrome display and the following attributes: black, video, blinking video, and intensified video. A resolution of 640 x 350 uses 56,000 bytes of video memory to support the four attributes. This mode uses maps 0 and 2; map 0 is the video bit plane (C0), and map 2 is the intensity bit plane (C2). Both planes reside at address hex A0000.

The two bits, one from each bit plane, define one PEL. The bit definitions are given in the following table.

C2	C0	PEL Color
0	0	Black
0	1	White
1	0	Blinking White
1	1	Intensified White

Figure 14. Bit Definitions C2,C0

Memory is organized with successive bytes defining successive PELs. The first eight PELs displayed are defined by the byte at hex A0000, the second eight PELs at hex A0001, and so on. The most-significant bit in each byte defines the first PEL for that byte.

Since both bit planes reside at address hex A0000, the user must select the plane to update through the Map Mask register of the sequence controller (see "Video Memory Organization" on page 20).

640 x 480 Two-Color Graphics (Mode Hex 11)

This mode provides two-color graphics with the same data format as mode 6. Addressing and mapping are shown under "Video Memory Organization" on page 20.

The bit image data is stored in map 0 and comprises a single bit plane (C0). The video buffer starts at hex A0000. The first byte contains the first eight PELs; the second byte at hex A0001 contains the second eight PELs, and so on. The bit definition for each PEL is 0 equals black and 1 equals intensified white.

16-Color Graphics Modes (Mode Hex 10, D, E, and 12)

These modes support 16 colors. For all modes, the bit image data is stored in all four memory maps. Each memory map contains the data for one bit plane. The bit planes are C0 through C3 and represent the following colors:

C0 = Blue
C1 = Green
C2 = Red
C3 = Intensified

The four bits define each PEL on the screen by acting as an address (pointer) into the internal palette in the VGA.

The display buffer resides at address hex A0000. The Map Mask register selects any or all of the maps to be updated when the system writes to the display buffer.

256-Color Graphics Mode (Mode Hex 13)

This mode provides graphics with the capability of displaying 256 colors at one time.

The display buffer is sequential, starts at address hex A0000, and is 64,000 bytes long. The first byte contains the color information for the upper-left PEL. The second byte contains the second PEL, and so on, for 64,000 PELs (320 x 200). The bit image data is stored in all four memory maps and comprises four bit planes. The four bit planes are sampled twice to produce eight bit-plane values that address the video DAC.

In this mode, the internal palette of the video subsystem is loaded by BIOS and should not be changed. The first 16 locations in external palette, which is in the video DAC, contain the colors compatible with the alphanumeric modes. The second 16 locations contain 16 evenly spaced gray shades. The next 216 locations contain values based on a hue-saturation-intensity model tuned to provide a usable, generic color set that covers a wide range of color values.

The following figure shows the color information that is compatible with the colors in other modes.

PEL Bits 7 6 5 4 3 2 1 0	Color Output
0 0 0 0 0 0 0 0	Black
0 0 0 0 0 0 0 1	Blue
0 0 0 0 0 0 1 0	Green
0 0 0 0 0 0 1 1	Cyan
0 0 0 0 0 1 0 0	Red
0 0 0 0 0 1 0 1	Magenta
0 0 0 0 0 1 1 0	Brown
0 0 0 0 0 1 1 1	White
0 0 0 0 1 0 0 0	Dark Gray
0 0 0 0 1 0 0 1	Light Blue
0 0 0 0 1 0 1 0	Light Green
0 0 0 0 1 0 1 1	Light Cyan
0 0 0 0 1 1 0 0	Light Red
0 0 0 0 1 1 0 1	Light Magenta
0 0 0 0 1 1 1 0	Yellow
0 0 0 0 1 1 1 1	Intensified White

Figure 15. Compatible Color Coding

Each color in the palette can be programmed to one of 256K different colors.

The features of this mode are:

- A maximum of 200 rows with 320 PELs
- Double scanned to display as 400 rows
- Memory-mapped graphics
- 256 of 256K colors for each PEL
- One byte per PEL
- 64,000 bytes of video memory.

Video Memory Organization

The display buffer consists of 256KB of dynamic read/write memory configured as four 64KB memory maps.

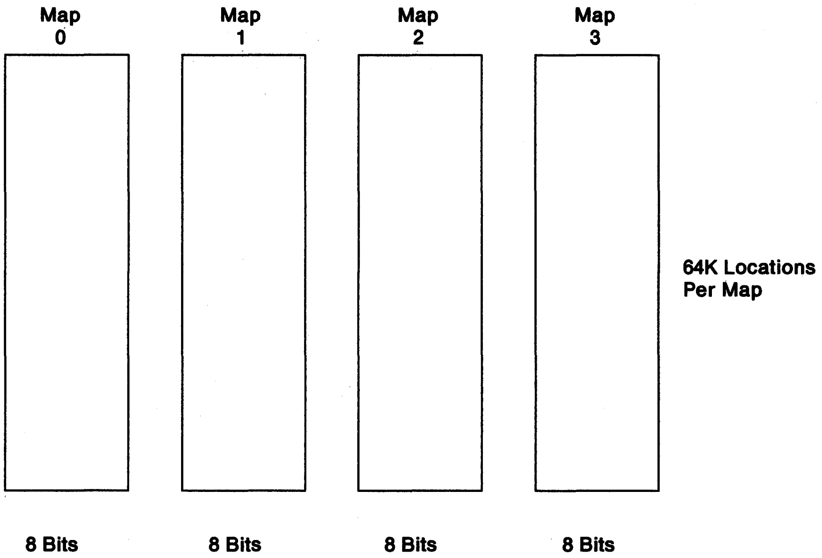


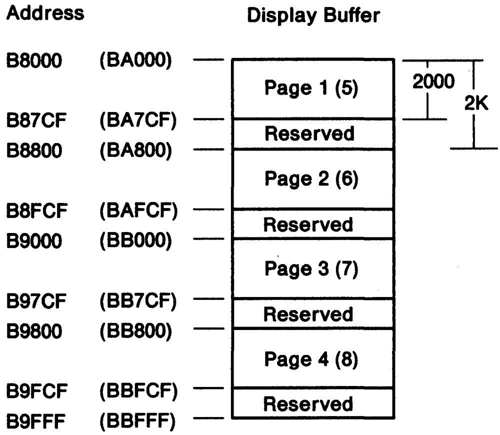
Figure 16. 256KB Video Memory Map

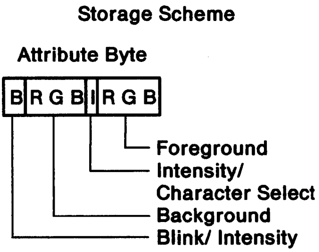
The starting address and size of the display buffer can be changed to maintain compatibility with other display adapters and application software. There are three configurations used by other adapters:

Address hex A0000 for a length of 64KB
Address hex B0000 for a length of 32KB
Address hex B8000 for a length of 32KB.

The following pages show the memory organization for each of the BIOS modes.

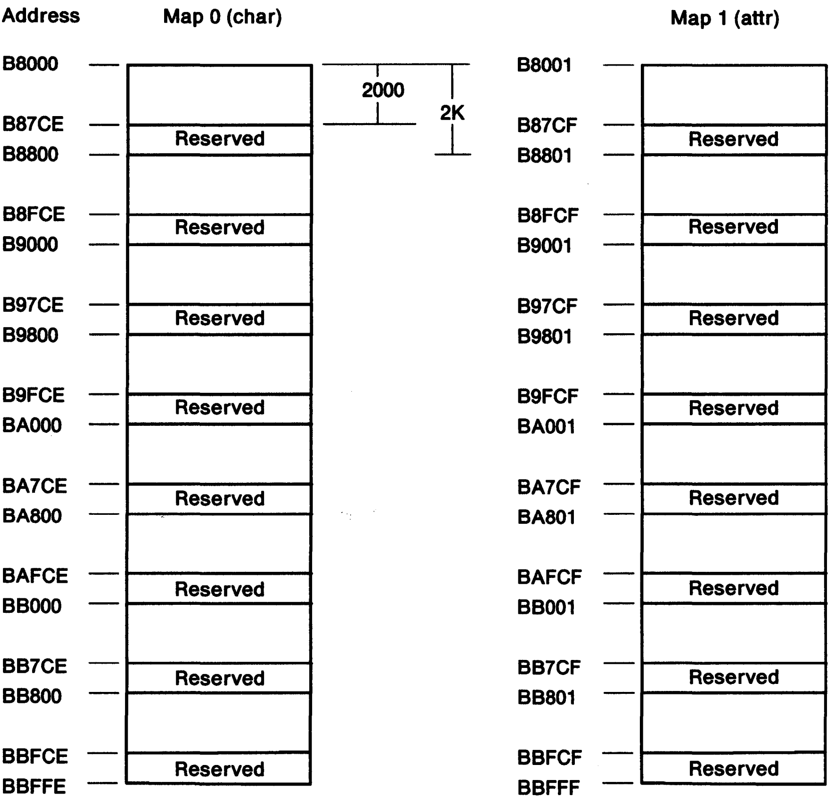
Modes Hex 0, 1



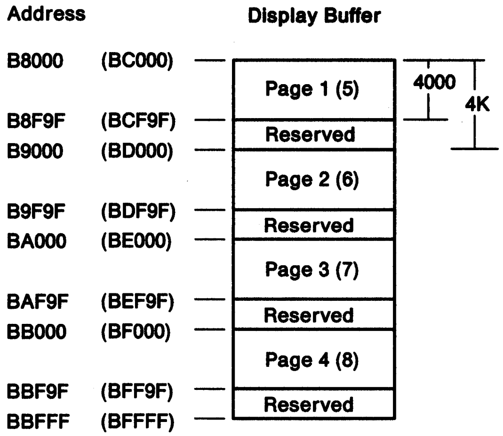


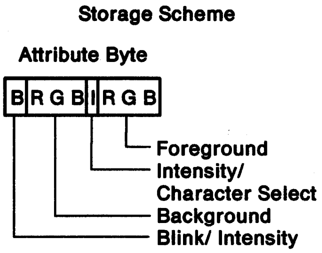
-16 colors per character

Character Byte
-Format is one character per byte



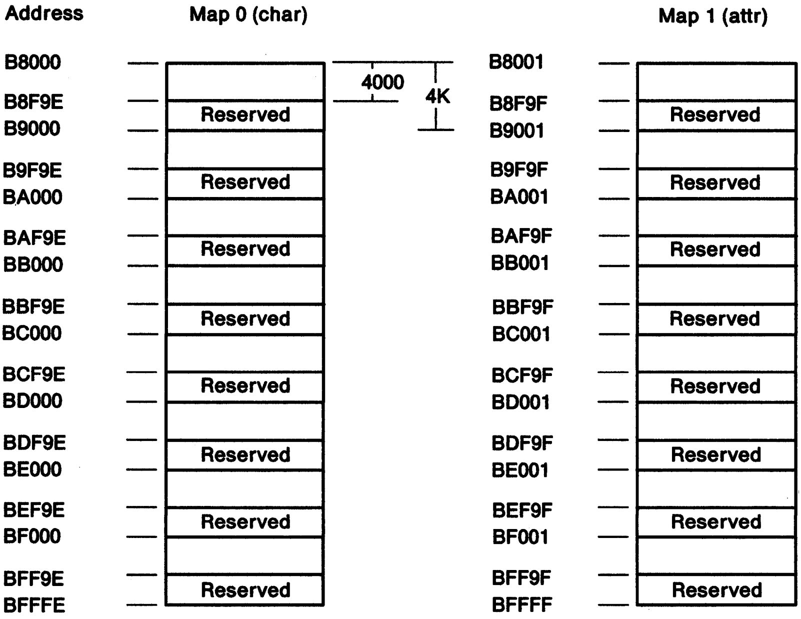
Modes Hex 2, 3





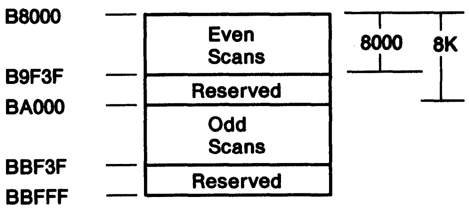
-16 colors per character

Character Byte
-Format is one character per byte

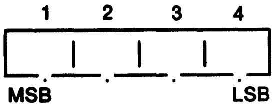


Modes Hex 4, 5

Address Display Buffer

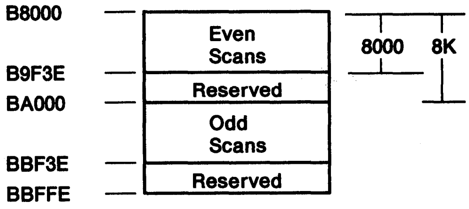


Storage Scheme

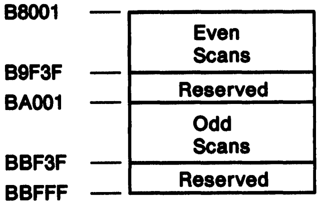


- 4 PELs per byte
- 4 colors per PEL
- First PEL is the two MSBs

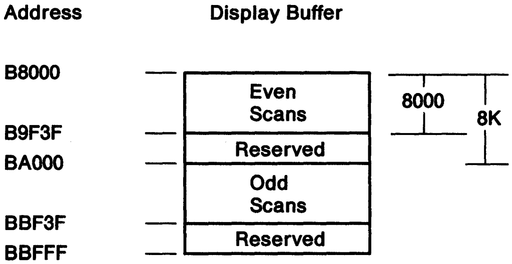
Address Map 0 (C0)

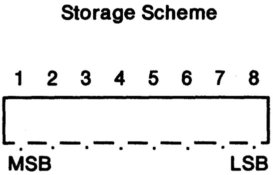


Map 1 (C1)

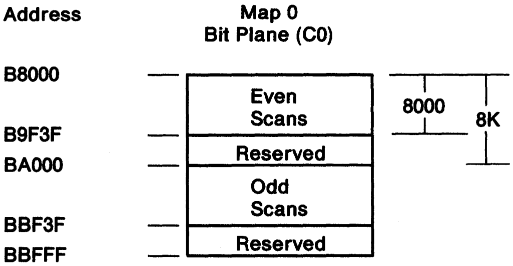


Mode Hex 6



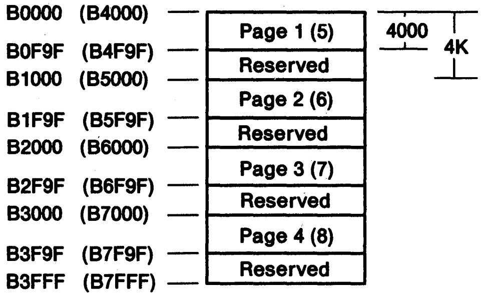


- Eight PELs per byte
- Two colors per PEL
- First PEL is MSB

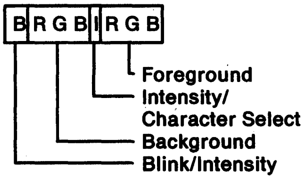


Mode Hex 7

Address Display Buffer Storage Scheme



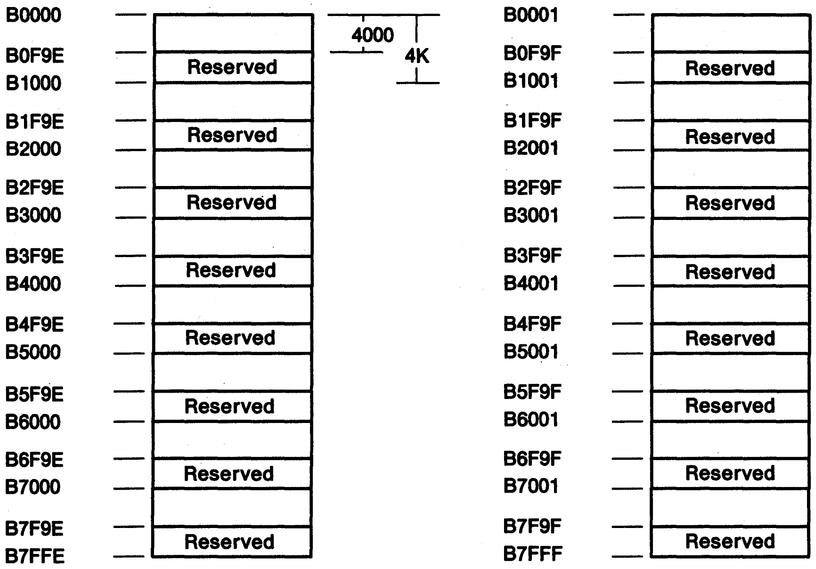
Attribute Byte



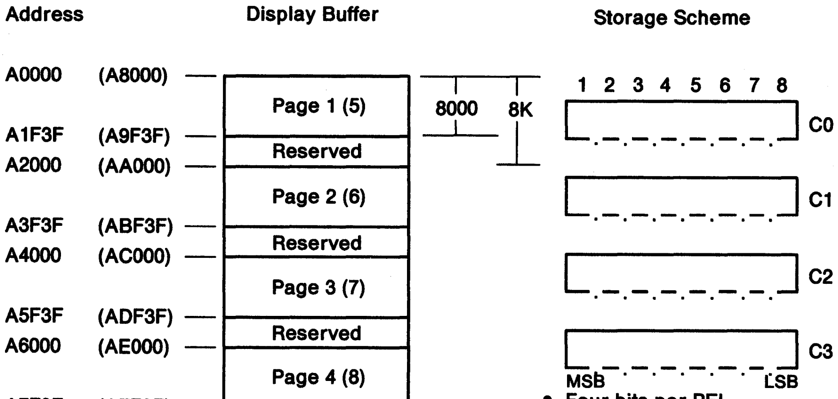
-Four attributes per character

Character Byte
-Format is one character per byte.

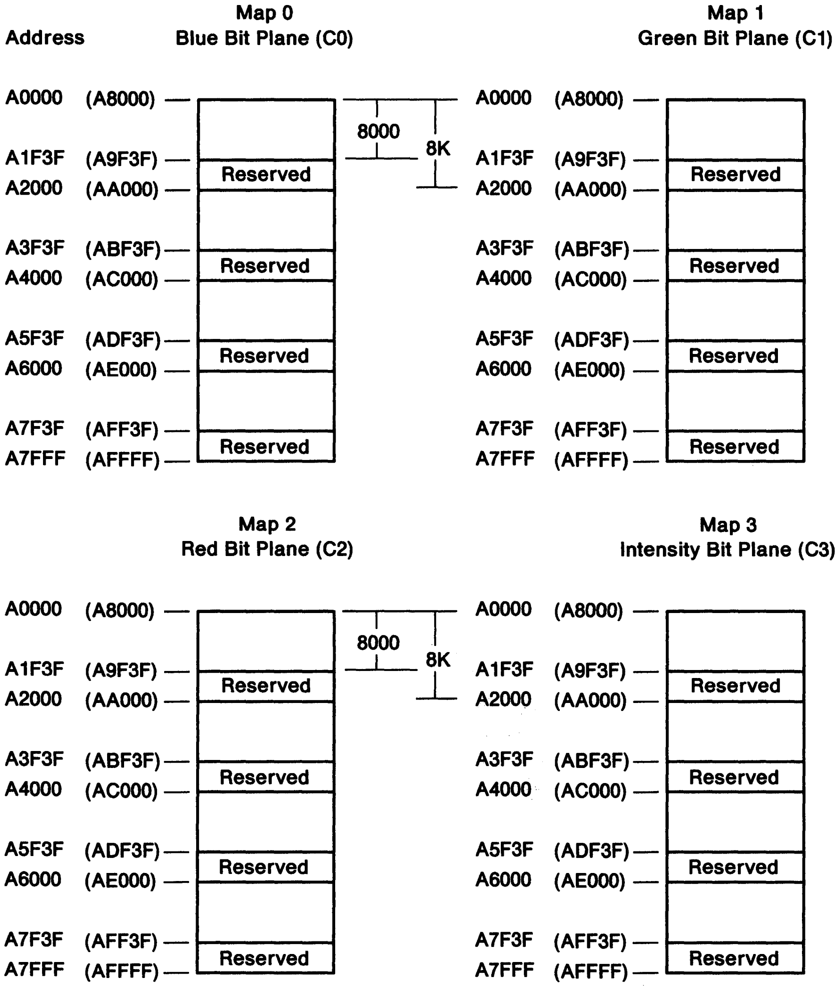
Address Map 0 (char) Map 1 (attr)



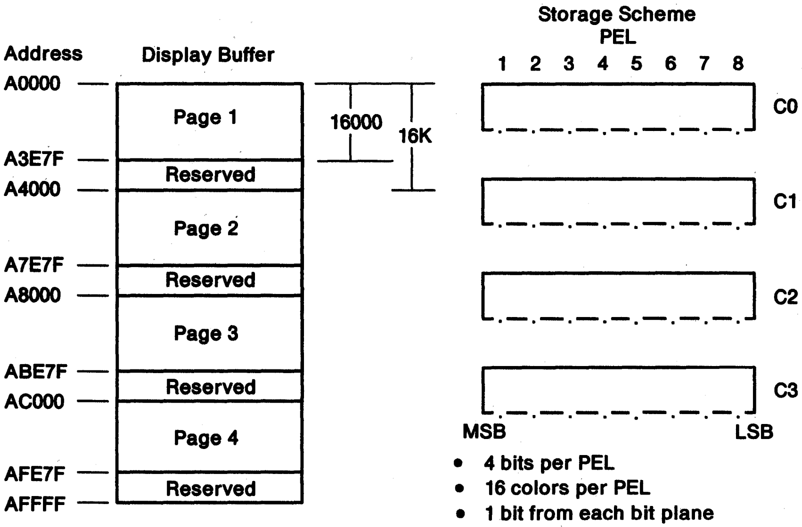
Mode Hex D



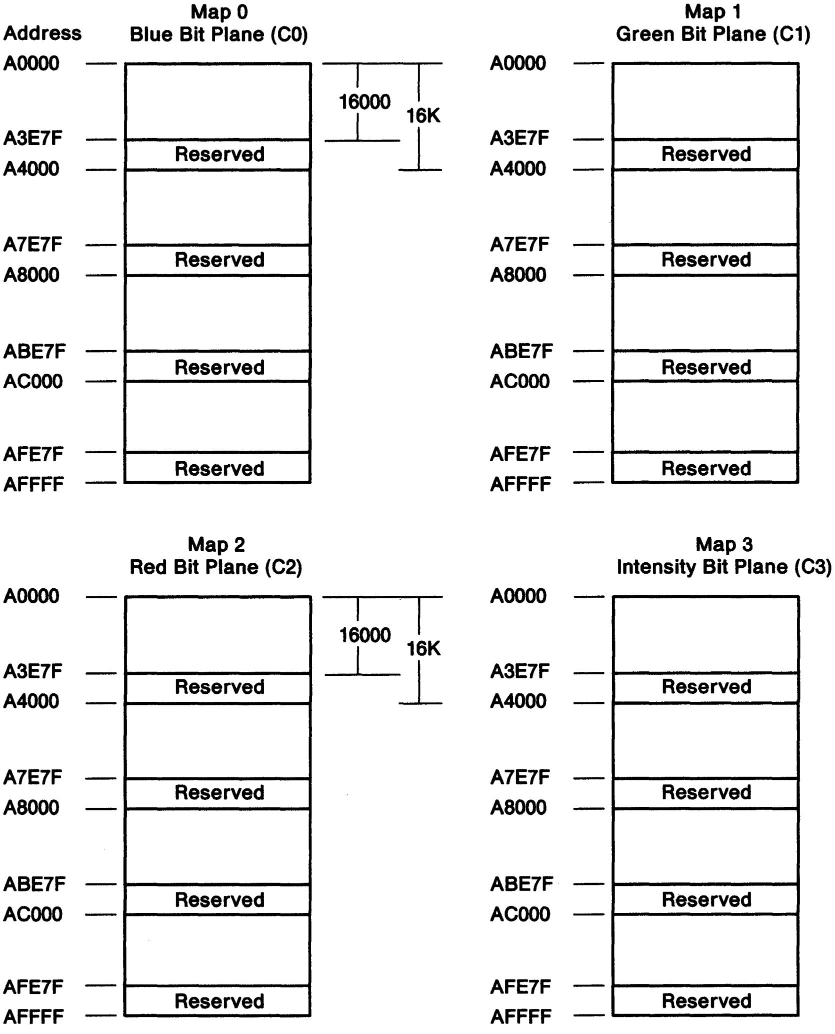
- Four bits per PEL
- 16 colors per PEL
- One bit from each bit plane (C3,C2,C1,CO) per PEL
- First PEL is MSB of all four bit planes


Address
Map 0
Blue Bit Plane (C0)
Map 1
Green Bit Plane (C1)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
8000
8K
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
Map 2
Red Bit Plane (C2)
Map 3
Intensity Bit Plane (C3)
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved
8000
8K
A0000 (A8000)
A1F3F (A9F3F)
A2000 (AA000)
A3F3F (ABF3F)
A4000 (AC000)
A5F3F (ADF3F)
A6000 (AE000)
A7F3F (AFF3F)
A7FFF (AFFFF)
Reserved
Reserved
Reserved
Reserved

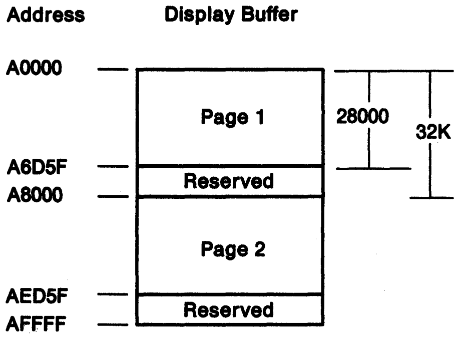
Mode Hex E

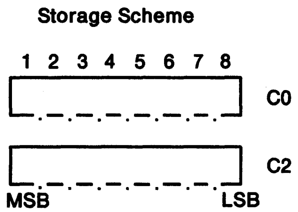


- 4 bits per PEL
- 16 colors per PEL
- 1 bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is MSB of all 4 bit planes

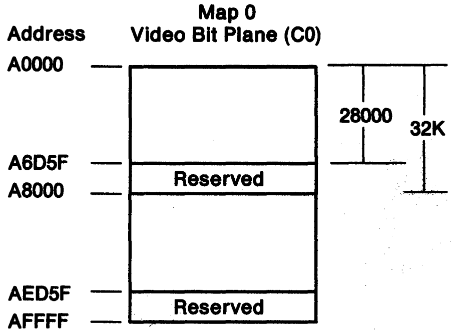

Address
Map 0
Blue Bit Plane (C0)
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
16000
16K
Map 1
Green Bit Plane (C1)
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
Map 2
Red Bit Plane (C2)
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved
16000
16K
Map 3
Intensity Bit Plane (C3)
A0000
A3E7F
A4000
A7E7F
A8000
ABE7F
AC000
AFE7F
AFFFF
Reserved
Reserved
Reserved
Reserved

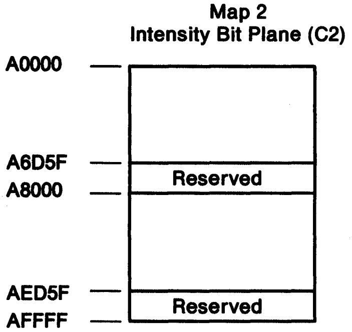
Mode Hex F



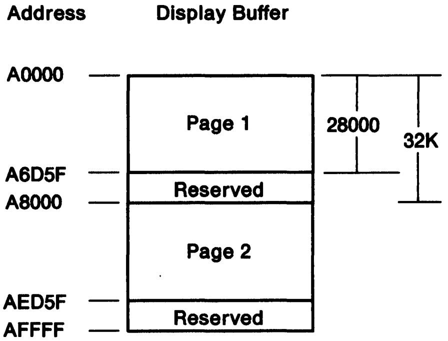


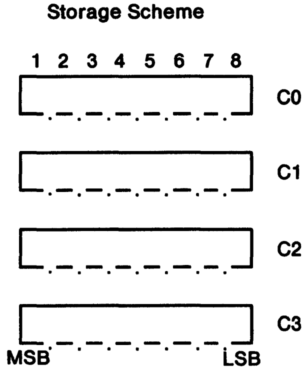
- Two bits per PEL
- Four attributes per PEL
- One bit from each bit plane (C2,C0)
- First PEL is MSB of video and intensity bit planes





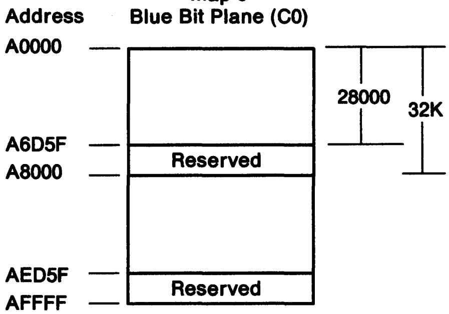
Mode Hex 10



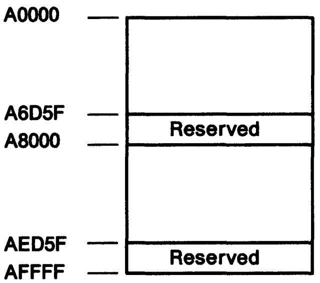


- Four bits per PEL
- 16 colors per PEL
- One bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is MSB of all four bit planes

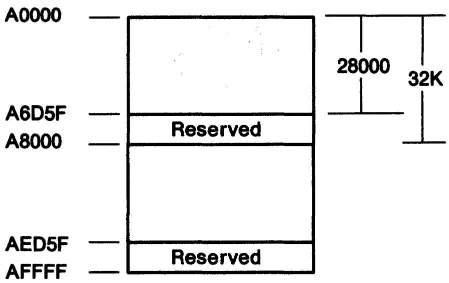
Map 0
Blue Bit Plane (C0)



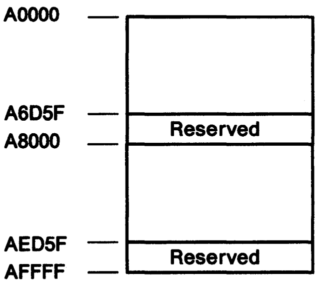
Map 1
Green Bit Plane (C1)



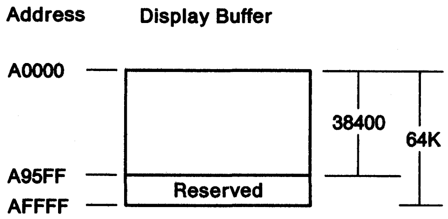
Map 2
Red Bit Plane (C2)

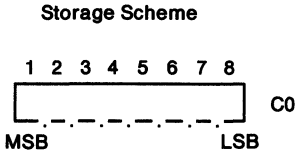


Map 3
Intensity Bit Plane (C3)

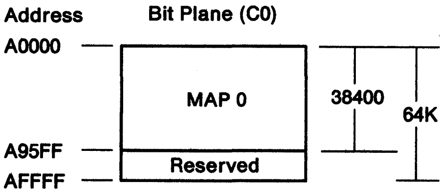


Mode Hex 11

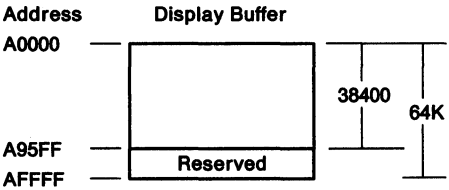


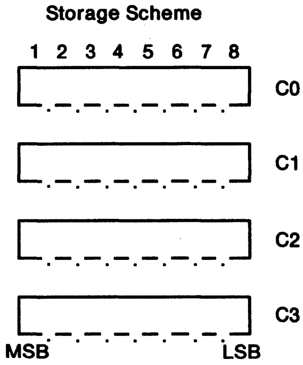


- One bit per PEL
- Two attributes per PEL
- First PEL is MSB

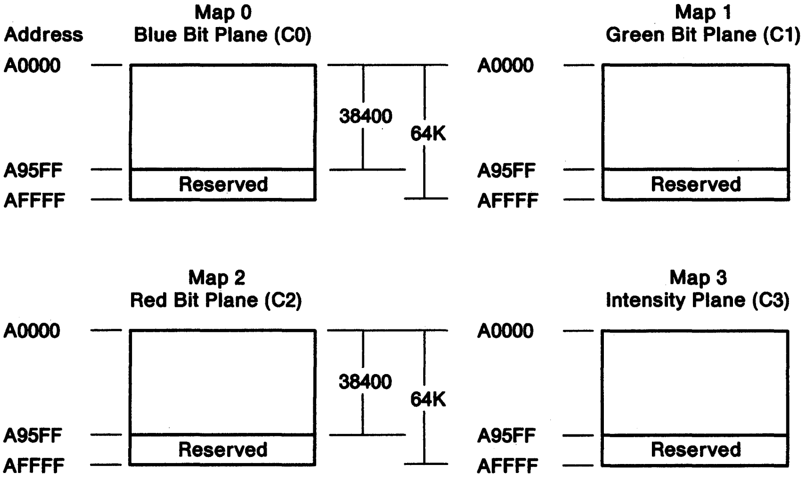


Mode Hex 12

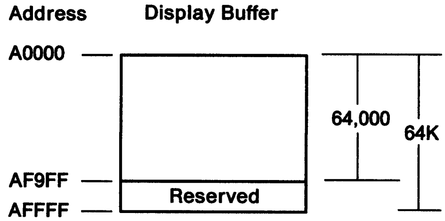


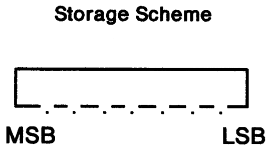


- Four bits per PEL
- 16 colors per PEL
- One bit from each bit plane (C3,C2,C1,C0) per PEL
- First PEL is MSB of all four bit planes

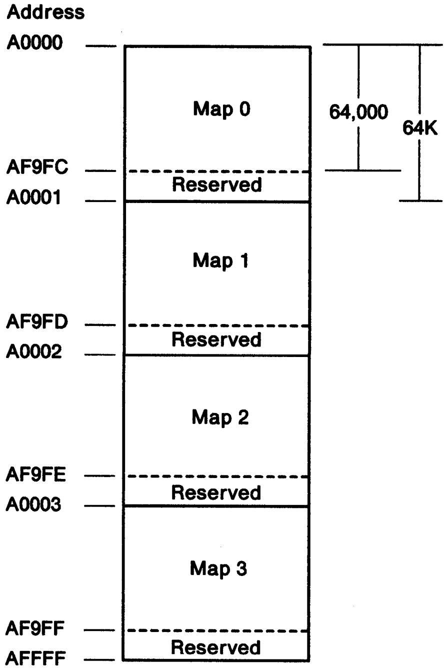


Mode Hex 13





- 8 Bits per PEL
- 256 Colors per PEL
- 1 PEL per Byte
- First PEL is at Address A0000



Memory Operations

Write Operations

When the system is writing to the display buffer, the maps are enabled by the logical decode of the memory address and the Map Mask register. The addresses used for video memory depend on the mode selected. The data flow for a system Write operation is illustrated in the following figure.

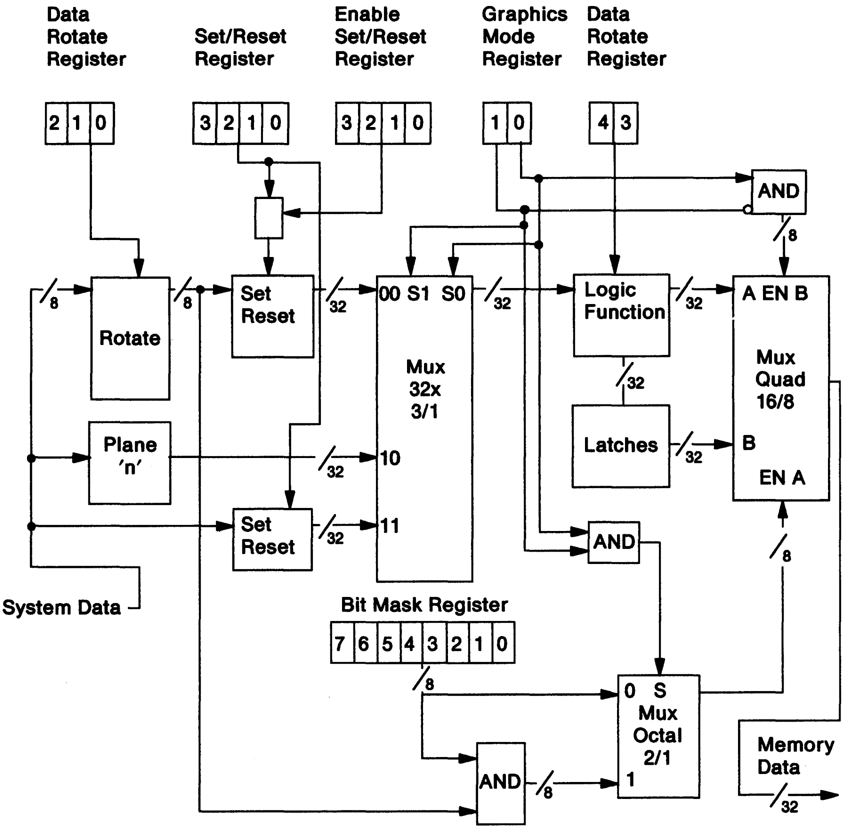


Figure 17. Data Flow for Write Operations

Read Operations

The two ways to read the video buffer are selected through the Graphics Mode register in the graphics controller. The Mode 0 Read operation returns the 8-bit value determined by the logical decode of the memory address and, if applicable, the Read Map Select register. The Mode 1 Read operation returns the 8-bit value resulting from the Color Compare operation controlled by the Color Compare and Color Don't Care registers. The data flow for the Color Compare operation is shown in the following figure.

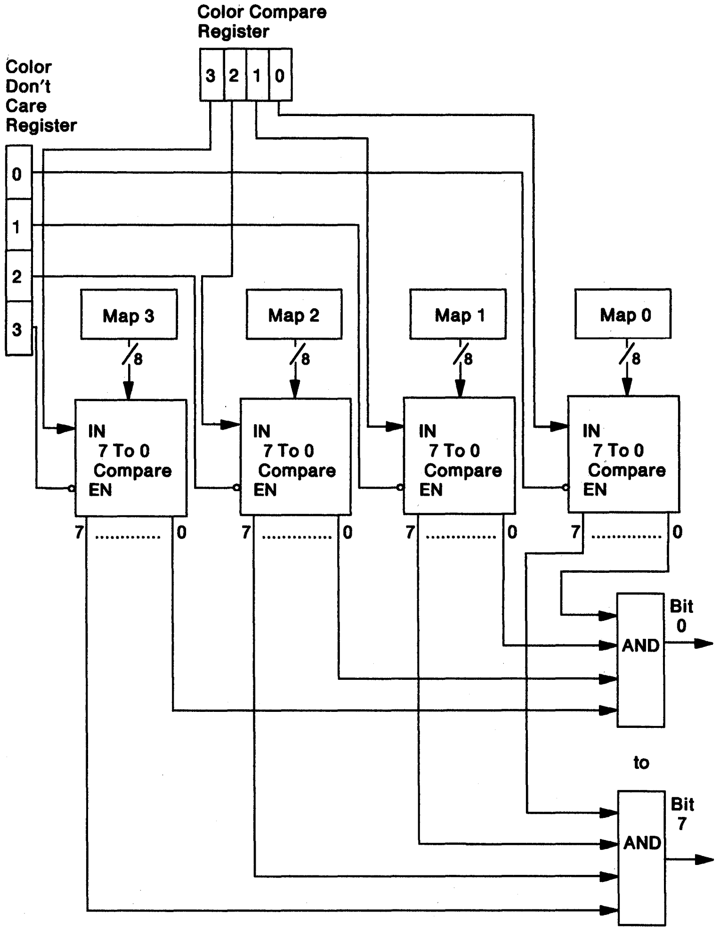


Figure 18. Color Compare Operations

Registers

There are six groups of registers in the video subsystem. All video registers are readable except the system data latches and the attribute address flip-flop. The following figure lists the register groups, their I/O addresses with the type of access (read or write), and page reference numbers.

The question mark in the address can be a hex B or D depending on the setting of the I/O address bit in the Miscellaneous Output register, described in “General Registers” on page 38.

Note: All registers in the video subsystem are read/write. The value of reserved bits in these registers must be preserved. Read the register first and change only the bits required.

Registers	**R/W**	**Port Address**	**Page Reference**
General Registers			38
Sequencer Registers			42
Address Register	R/W	03C4	
Data Registers	R/W	03C5	
CRT Controller Registers			49
Address Register	R/W	03?4	
Data Registers	R/W	03?5	
Graphics Controller Registers			68
Address Register	R/W	03CE	
Data Registers	R/W	03CF	
Attribute Controller Registers			76
Address Register	R/W	03C0	
Data Registers	W	03C0	
	R	03C1	
Video DAC Palette Registers			90
Write Address	R/W	03C8	
Read Address	W	03C7	
Data	R/W	03C9	
PEL Mask	R/W	03C6	

Figure 19. Video Subsystem Register Overview

General Registers

Register	Read Address	Write Address
Miscellaneous Output Register	03CC	03C2
Input Status Register 0	03C2	-
Input Status Register 1	03?A	-
Feature Control Register	03CA	03?A
Video Subsystem Enable Register	03C3	03C3

Figure 20. General Registers

Miscellaneous Output Register

The read address for this register is hex 03CC and its write address is hex 03C2.

Bit	Function
7	Vertical Sync Polarity
6	Horizontal Sync Polarity
5, 4	Reserved
3, 2	Clock Select
1	Enable RAM
0	I/O Address Select

Figure 21. Miscellaneous Output Register, Hex 03CC/03C2

Bit 7 When set to 0, this bit selects a positive 'vertical retrace' signal. This bit works with bit 6 to determine the vertical size.

Bit 6 When set to 0, this bit selects a positive 'horizontal retrace' signal. Bits 7 and 6 select the vertical size as shown in the following figure.

Bits 7 6	Vertical Size
0 0	Reserved
0 1	400 lines
1 0	350 lines
1 1	480 lines

Figure 22. Display Vertical Size

Bits 5, 4 Reserved.

Bits 3, 2 These two bits select the clock source according to the following figure. The external clock is driven through the auxiliary video extension. The input clock should be kept between 14.3 MHz and 28.4 MHz.

Bits 3 2	Function
0 0	Selects 25.175 MHz clock for 640/320 Horizontal PELs
0 1	Selects 28.322 MHz clock for 720/360 Horizontal PELs
1 0	Selects External Clock
1 1	Reserved

Figure 23. Clock Select Definitions

Bit 1 When set to 0, this bit disables address decode for the display buffer from the system.

Bit 0 This bit selects the CRT controller addresses. When set to 0, this bit sets the CRT controller addresses to hex 03Bx and the address for the Input Status Register 1 to hex 03BA for compatibility with the monochrome adapter. When set to 1, this bit sets CRT controller addresses to hex 03Dx and the Input Status Register 1 address to hex 03DA for compatibility with the color/graphics adapter. The Write addresses to the Feature Control register are affected in the same manner.

Input Status Register 0

The address for this read-only register is address hex 03C2.

Bit	Function
7	CRT Interrupt
6, 5	Reserved
4	Switch Sense Bit
3 - 0	Reserved

Figure 24. Input Status Register 0, Hex 03C2

Bit 7 When set to 1, this bit indicates a vertical retrace interrupt is pending.

Bits 6, 5 Reserved.

Bit 4 This bit is used by BIOS in determining the type of display attached.

Bits 3 - 0 Reserved.

Input Status Register 1

The address for this read-only register is address hex 03DA or 03BA.

Bit	Function
7 - 4	Reserved
3	Vertical Retrace
2, 1	Reserved
0	Display Enable

Figure 25. Input Status Register 1, Hex 03DA/03BA

Bits 7 - 4 Reserved.

Bit 3 When set to 1, this bit indicates a vertical retrace interval. This bit can be programmed, through the Vertical Retrace End register, to generate an interrupt at the start of the vertical retrace.

Bits 2, 1 Reserved.

Bit 0 When set to 1, this bit indicates a horizontal or vertical retrace interval. This bit is the real-time status of the inverted 'display enable' signal. Programs have used this status bit to restrict screen updates to the inactive display intervals in order to reduce screen flicker. The video subsystem is designed to eliminate this software requirement; screen updates may be made at any time without screen degradation.

Feature Control Register

This register's write address is hex 03DA or 03BA; its read address is hex 03CA. All bits are reserved.

Bit	Function
7 - 0	Reserved

Figure 26. Feature Control Register, Hex 03DA/03BA and 03CA

Video Subsystem Enable Register

This register is at address hex 03C3. Accessing this register does not affect the video POS enable bit described in "Programmable Option Select" on page 10.

Bit	Function
7 - 1	Reserved
0	Video Subsystem Enable

Figure 27. Video Subsystem Enable Register, Hex 03C4

Bits 7 - 1 Reserved.

Bit 0 When this bit is set to 1, the I/O and memory address decoding for the video subsystem are enabled. When set to 0, this bit disables the video I/O and memory address decoding.

Sequencer Registers

The Address register is at address hex 03C4 and the data registers are at address hex 03C5. All registers within the sequencer are read/write.

Register	Index (Hex)
Sequencer Address	-
Reset	00
Clocking Mode	01
Map Mask	02
Character Map Select	03
Memory Mode	04

Figure 28. Sequencer Registers

Sequencer Address Register

The Address register is at address hex 03C4. This register is loaded with an index value that points to the desired sequencer data register.

Bit	Function
7 - 3	Reserved
2 - 0	Sequencer Address

Figure 29. Sequencer Address Register

Bits 7 - 3 Reserved.

Bits 2 - 0 These bits contain the index value that points to the data register to be accessed.

Reset Register

This read/write register has an index of hex 00; its address is hex 03C5.

Bit	Function
7 - 2	Reserved
1	Synchronous Reset
0	Asynchronous Reset

Figure 30. Reset Register, Index Hex 00

Bits 7 - 2 Reserved.

Bit 1 When set to 0, this bit commands the sequencer to synchronously clear and halt. Bits 1 and 0 must be 1 to allow the sequencer to operate. To prevent the loss of data, bit 1 must be set to 0 during the active display interval before changing the clock selection. The clock is changed through the Clocking Mode register or the Miscellaneous Output register.

Bit 0 When set to 0, this bit commands the sequencer to asynchronously clear and halt. Resetting the sequencer with this bit can cause loss of video data.

Clocking Mode Register

This read/write register has an index of hex 01; its address is hex 03C5.

Bit	Function
7, 6	Reserved
5	Screen Off
4	Shift 4
3	Dot Clock
2	Shift Load
1	Reserved
0	8/9 Dot Clocks

Figure 31. Clocking Mode Register, Index Hex 01

Bits 7, 6 Reserved.

Bit 5 When set to 1, this bit turns off the display and assigns maximum memory bandwidth to the system. Although the display is blanked, the synchronization pulses are maintained. This bit can be used for rapid full-screen updates.

Bit 4 When this bit and bit 2 are set to 0, the video serializers are loaded every character clock. When this bit is set to 1, the serializers are loaded every fourth character clock, which is useful when 32 bits are fetched per cycle and chained together in the shift registers.

Bit 3 When set to 0, this bit selects the normal dot clocks derived from the sequencer master clock input. When this bit is set to 1, the master clock will be divided by 2 to generate the dot clock. All other timings are affected because they are derived from the dot clock. The dot clock divided by 2 is used for 320 and 360 horizontal PEL modes.

Bit 2 When this bit and bit 4 are set to 0, the video serializers are loaded every character clock. When this bit is set to 1, the video serializers are loaded every other character clock, which is useful when 16 bits are fetched per cycle and chained together in the shift registers.

Bit 1 Reserved.

Bit 0 When set to 0, this bit directs the sequencer to generate character clocks 9 dots wide; when set to 1, it directs the sequencer to generate character clocks 8 dots wide. The 9 dot mode is for alphanumeric modes 0+, 1+, 2+, 3+, 7 and 7+ only; the 9th dot equals the 8th dot for ASCII codes hex C0 through DF. All other modes must use 8 dots per character clock. See the line graphics character bit in the Attribute Mode Control register on page 78.

Map Mask Register

This read/write register has an index of hex 02; its address is hex 03C5.

Bit	Function
7 - 4	Reserved
3	Map 3 Enable
2	Map 2 Enable
1	Map 1 Enable
0	Map 0 Enable

Figure 32. Map Mask Register, Index Hex 02

Bits 7 - 4 Reserved.

Bits 3 - 0 When set to 1, these bits enable system access to the corresponding map. If all maps are enabled, the system can write its 8-bit value to all four maps in a single memory cycle. This substantially reduces the system overhead during display updates in graphics modes.

Data scrolling operations can be enhanced by enabling all maps and writing the display buffer address with the data stored in the system data latches. This is a Read-Modify-Write operation.

When odd/even modes are selected, maps 0 and 1 and maps 2 and 3 should have the same map mask value.

When chain 4 mode is selected, all maps should be enabled.

Character Map Select Register

This register's index is hex 03; its address is hex 03C5. In alphanumeric modes, bit 3 of the attribute byte normally defines the foreground intensity. This bit can be redefined as a switch between character sets allowing 512 displayable characters. To enable this feature:

1. Set the extended memory bit in the Memory Mode register (hex 04) to 1.
2. Select different values for character map A and character map B.

This function is supported by BIOS and is a function call within the character generator routines.

Bit	Function
7, 6	Reserved
5	Character Map A Select (MSB)
4	Character Map B Select (MSB)
3, 2	Character Map A Select
1, 0	Character Map B Select

Figure 33. Character Map Select Register, Index Hex 03

Bits 7, 6 Reserved.

Bit 5 This bit is the most-significant bit for selecting the location of character map A.

Bit 4 This bit is the most-significant bit for selecting the location of character map B.

Bits 3, 2 These bits and bit 5 select the location of character map A. Map A is the area of map 2 containing the character font table used to generate characters when attribute bit 3 is set to 1. The selection is shown in the following figure.

Bits 5 3 2	Map Selected	Table Location
0 0 0	0	1st 8KB of Map 2
0 0 1	1	3rd 8KB of Map 2
0 1 0	2	5th 8KB of Map 2
0 1 1	3	7th 8KB of Map 2
1 0 0	4	2nd 8KB of Map 2
1 0 1	5	4th 8KB of Map 2
1 1 0	6	6th 8KB of Map 2
1 1 1	7	8th 8KB of Map 2

Figure 34. Character Map Select A

Bits 1, 0 These bits and bit 4 select the location of character map B. Map B is the area of map 2 containing the character font table used to generate characters when attribute bit 3 is set to 0. The selection is shown in the following figure.

Bits 4 1 0	Map Selected	Table Location
0 0 0	0	1st 8KB of Map 2
0 0 1	1	3rd 8KB of Map 2
0 1 0	2	5th 8KB of Map 2
0 1 1	3	7th 8KB of Map 2
1 0 0	4	2nd 8KB of Map 2
1 0 1	5	4th 8KB of Map 2
1 1 0	6	6th 8KB of Map 2
1 1 1	7	8th 8KB of Map 2

Figure 35. Character Map Select B

Memory Mode Register

This register's index is hex 04; its address is hex 03C5.

Bit	Function
7 - 4	Reserved
3	Chain 4
2	Odd/Even
1	Extended Memory
0	Reserved

Figure 36. Memory Mode Register, Index Hex 04

Bits 7 - 4 Reserved.

Bit 3 This bit controls the map selected during system Read operations. When set to 0, this bit enables system addresses to sequentially access data within a bit map by using the Map Mask register. When set to 1, this bit causes the two low-order bits to select the map accessed as shown in following figure.

Address Bits A1 A0	Map Selected
0 0	0
0 1	1
1 0	2
1 1	3

Figure 37. Map Selection, Chain 4

Bit 2 When this bit is set to 0, even system addresses access maps 0 and 2, while odd system addresses access maps 1 and 3. When this bit set to 1, system addresses sequentially access data within a bit map, and the maps are accessed according to the value in the Map Mask register (hex 02).

Bit 1 When set to 1, this bit enables the video memory from 64KB to 256KB. This bit must be set to 1 to enable the character map selection described for the previous register.

Bit 0 Reserved.

CRT Controller Registers

A data register is accessed by writing its index to the Address register at address hex 03D4 or 03B4, and then writing the data to the access port at address hex 03D5 or 03B5. The I/O address used depends on the setting of the I/O address select bit (bit 0) in the Miscellaneous Output register, which is described in "General Registers" on page 38. The following figure shows the variable part of the address as a question mark.

Note: When modifying a register, the setting of reserved bits must be preserved. Read the register first and change only the bits required.

Register	**Address (Hex)**	**Index (Hex)**
Address	03?4	-
Horizontal Total	03?5	00
Horizontal Display Enable End	03?5	01
Start Horizontal Blanking	03?5	02
End Horizontal Blanking	03?5	03
Start Horizontal Retrace Pulse	03?5	04
End Horizontal Retrace	03?5	05
Vertical Total	03?5	06
Overflow	03?5	07
Preset Row Scan	03?5	08
Maximum Scan Line	03?5	09
Cursor Start	03?5	0A
Cursor End	03?5	0B
Start Address High	03?5	0C
Start Address Low	03?5	0D
Cursor Location High	03?5	0E
Cursor Location Low	03?5	0F
Vertical Retrace Start	03?5	10
Vertical Retrace End	03?5	11
Vertical Display Enable End	03?5	12
Offset	03?5	13
Underline Location	03?5	14
Start Vertical Blanking	03?5	15
End Vertical Blanking	03?5	16
CRT Mode Control	03?5	17
Line Compare	03?5	18

Figure 38. CRT Controller Registers

Address Register

This register is at address hex 03B4 or 03D4, and is loaded with an index value that points to the data registers within the CRT controller.

Bit	Function
7 - 5	Reserved
4 - 0	Index 4 - 0

Figure 39. CRT Controller Address Register, Hex 03B4/03D4

Bits 7 - 5 Reserved.

Bits 4 - 0 These bits are the index that points to the data register accessed through address hex 03D5 or 03B5.

Horizontal Total Register

This register's index is hex 00; its address is hex 03D5 or 03B5. It defines the total number of characters in the horizontal scan interval including the retrace time. The value directly controls the period of the 'horizontal retrace' signal. A horizontal character counter in the CRT controller counts the character clock inputs; comparators are used to compare the register value with the character's horizontal width to provide horizontal timings. All horizontal and vertical timings are based on this register.

Bit	Function
7 - 0	Horizontal Total

Figure 40. Horizontal Total Register, Index Hex 00

Bits 7 - 0 The value of these bits is the total number of characters minus 5.

Horizontal Display-Enable End Register

This register's index is hex 01; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Horizontal Display Enable End

Figure 41. Horizontal Display Enable-End Register, Index Hex 01

Bits 7 - 0 These bits define the length of the 'horizontal display-enable' signal, and determine the number of character positions per horizontal line. The value of these bits is the total number of displayed characters minus 1.

Start Horizontal Blanking Register

This register's index is hex 02; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Start Horizontal Blanking

Figure 42. Start Horizontal Blanking Register, Index Hex 02

Bits 7 - 0 This value is the horizontal character count where the 'horizontal blanking' signal goes active.

End Horizontal Blanking Register

This register's index is hex 03; its address is hex 03D5 or 03B5. It determines when the 'horizontal blanking' signal will go active.

Bit	Function
7	Reserved
6, 5	Display Enable Skew Control
4 - 0	End Blanking

Figure 43. End Horizontal Blanking Register, Index Hex 03

Bit 7 Reserved.

Bits 6, 5 These two bits determine the amount of skew of the ‘display enable’ signal. This skew control is needed to provide sufficient time for the CRT controller to read a character and attribute code from the video buffer, to gain access to the character generator, and go through the Horizontal PEL Panning register in the attribute controller. Each access requires the ‘display enable’ signal to be skewed one character clock so that the video output is synchronized with the horizontal and vertical retrace signals. The skew values are shown in the following figure.

Bits	
6 5	**Amount of Skew**
0 0	No character clock skew
0 1	One character clock skew
1 0	Two character clock skew
1 1	Three character clock skew

Figure 44. Display Enable Skew

Bits 4 - 0 These bits are the five low-order bits of a 6-bit value that is compared with the value in the Start Horizontal Blanking register to determine when the ‘horizontal blanking’ signal will go inactive. The most-significant bit is bit 7 in the End Horizontal Retrace register (index hex 05).

To program these bits for a signal width of W, the following algorithm is used: the width W, in character clock units, is added to the value from the Start Horizontal Blanking register. The six low-order bits of the result are the 6-bit value programmed.

Start Horizontal Retrace Pulse Register

This register's index is hex 04; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Start Horizontal Retrace Pulse

Figure 45. Start Horizontal Retrace Pulse Register, Index Hex 04

Bits 7 - 0 These bits are used to center the screen horizontally by specifying the character position where the 'horizontal retrace' signal goes active.

End Horizontal Retrace Register

This register's index is hex 05; its address is hex 03D5 or 03B5.

Bit	Function
7	End Horizontal Blanking, Bit 5
6, 5	Horizontal Retrace Delay
4 - 0	End Horizontal Retrace

Figure 46. End Horizontal Retrace Register, Index Hex 05

Bit 7 This bit is the most-significant bit of the end horizontal blanking value in the End Horizontal Blanking register (index hex 03).

Bits 6, 5 These bits control the skew of the 'horizontal retrace' signal. The value of these bits is the amount of skew provided (from 0 to 3 character clock units). For certain modes, the 'horizontal retrace' signal takes up the entire blanking interval. Some internal timings are generated by the falling edge of the 'horizontal retrace' signal. To ensure that the signals are latched properly, the 'retrace' signal is started before the end of the 'display enable' signal and then skewed several character clock times to provide the proper screen centering.

Bits 4 - 0 These bits are compared with the Start Horizontal Retrace register to give a horizontal character count where the 'horizontal retrace' signal goes inactive.

To program these bits with a signal width of W, the following algorithm is used: the width W, in character clock units, is added to the value in the Start Retrace register. The five low-order bits of the result are the 5-bit value programmed.

Vertical Total Register

This register's index is hex 06; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Vertical Total

Figure 47. Vertical Total Register, Index Hex 06

Bits 7 - 0 These are the eight low-order bits of a 10-bit vertical total. The value for the vertical total is the number of horizontal raster scans on the display, including vertical retrace, minus 2. This value determines the period of the 'vertical retrace' signal.

Bits 8 and 9 are in the Overflow register (index hex 07).

Overflow Register

This register's index is hex 07; its address is hex 03D5 or 03B5.

Bit	Function
7	Vertical Retrace Start, Bit 9
6	Vertical Display Enable End, Bit 9
5	Vertical Total, Bit 9
4	Line Compare, Bit 8
3	Vertical Blanking Start, Bit 8
2	Vertical Retrace Start, Bit 8
1	Vertical Display Enable End, Bit 8
0	Vertical Total, Bit 8

Figure 48. CRT Overflow Register, Index Hex 07

Bit 7 Bit 9 of the Vertical Retrace Start register (index hex 10).

Bit 6 Bit 9 of the Vertical Display Enable End register (index hex 12).

Bit 5 Bit 9 of the Vertical Total register (index hex 06).

Bit 4 Bit 8 of the Line Compare register (index hex 18).

Bit 3 Bit 8 of the Start Vertical Blanking register (index hex 15).

Bit 2 Bit 8 of the Vertical Retrace Start register (index hex 10).

Bit 1 Bit 8 of the Vertical Display Enable End register (index hex 12).

Bit 0 Bit 8 of the Vertical Total register (index hex 06).

Preset Row Scan Register

This register's index is hex 08; its address is hex 03D5 or 03B5.

Bit	Function
7	Reserved
6	Byte Panning 1
5	Byte Panning 0
4 - 0	Starting Row Scan Count

Figure 49. Preset Row Scan Register, Index Hex 08

Bit 7 Reserved.

Bits 6, 5 These two bits control byte panning in multiple shift modes. (Current BIOS modes do not use multiple shift operation.) These bits are used in PEL-panning operations, and should normally be set to 0.

The PEL Panning register in the attribute controller provides panning of up to eight individual PELs. In single-byte shift modes, to pan to the next higher PEL (8 or 9), the CRT controller start address is incremented and the PEL Panning register is reset to 0. In multiple shift modes, the byte-panning bits are used as extensions to the PEL Panning register. This allows panning across the width of the video output shift. For example, in the 32-bit shift mode, the byte pan and PEL-panning bits provide up to 31 bits of panning capability. To pan from position 31 to 32, the CRT controller start address is incremented and the panning bits, both PEL and byte, are reset to 0.

Bits 4 - 0 These bits specify the row scan count for the row starting after a vertical retrace. The row scan counter is incremented every horizontal retrace time until the maximum row scan occurs. When the maximum row scan is reached, the row scan counter is cleared (not preset).

Note: The CRT controller latches the start address at the start of the vertical retrace. These register values should be loaded during the active display time.

Maximum Scan Line Register

This register's index is hex 09; its address is hex 03D5 or 03B5.

Bit	Function
7	200 to 400 Line Conversion
6	Line Compare, Bit 9
5	Start Vertical Blanking, Bit 9
4 - 0	Maximum Scan Line

Figure 50. Maximum Scan Line Register, Index Hex 09

Bit 7 When this bit is set to 1, 200-scan-line video data is converted to 400-scan-line output. To do this, the clock in the row scan counter is divided by 2, which allows the 200-line modes to be displayed as 400 lines on the display (this is called double scanning; each line is displayed twice). When this bit is set to 0, the clock to the row scan counter is equal to the horizontal scan rate.

Bit 6 Bit 9 of the Line Compare register (index hex 18).

Bit 5 Bit 9 of the Start Vertical Blanking register (index hex 15).

Bits 4 - 0 These bits specify the number of scan lines per character row. The value of these bits is the maximum row scan number minus 1.

Cursor Start Register

This register's index is hex 0A; its address is hex 03D5 or 03B5.

Bit	Function
7, 6	Reserved
5	Cursor Off
4 - 0	Row Scan Cursor Begins

Figure 51. Cursor Start Register, Index Hex 0A

Bits 7, 6 Reserved.

Bit 5 When set to 1, this bit disables the cursor.

Bits 4 - 0 These bits specify the row within the character box where the cursor begins. The value of these bits is the first line of the cursor minus 1. When this value is greater than that in the Cursor End register, no cursor is displayed.

Cursor End Register

This register's index is hex 0B; its address is hex 03D5 or 03B5.

Bit	Function
7	Reserved
6, 5	Cursor Skew Control
4 - 0	Row Scan Cursor Ends

Figure 52. Cursor End Register, Index Hex 0B

Bit 7 Reserved.

Bits 6, 5 These bits control the skew of the cursor. The skew value delays the cursor by the selected number of character clocks from 0 to 3. For example, a skew of 1 moves the cursor right one position on the screen.

Bits 4 - 0 These bits specify the row within the character box where the cursor ends. If this value is less that the cursor start value, no cursor is displayed.

Start Address High Register

This register's index is hex 0C; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	High Byte of the Start Address

Figure 53. Start Address High Register, Index Hex 0C

Bits 7 - 0 These are the eight high-order bits of a 16-bit value that specifies the starting address for the regenerative buffer. The start address points to the first address after the vertical retrace on each screen refresh.

Note: The CRT controller latches the start address at the start of the vertical retrace. These register values should be loaded during the active display time.

Start Address Low Register

This register's index is hex 0D; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Low Byte of the Start Address

Figure 54. Start Address Low Register, Index Hex 0D

Bits 7 - 0 These are the eight low-order bits of the starting address for the regenerative buffer.

Cursor Location High Register

This register's index is hex 0E; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	High Byte of the Cursor Location

Figure 55. Cursor Location High Register, Index Hex 0E

Bits 7 - 0 These are the eight high-order bits of the 16-bit cursor location.

Cursor Location Low Register

This register's index is hex 0F; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Low Byte of the Cursor Location

Figure 56. Cursor Location Low Register, Index Hex 0F

Bits 7 - 0 These are the eight low-order bits of the cursor location.

Vertical Retrace Start Register

This register's index is hex 10; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Vertical Retrace Pulse

Figure 57. Vertical Retrace Start Register, Index Hex 10

Bits 7 - 0 These are the eight low-order bits of the 9-bit start position for the 'vertical retrace' pulse; it is programmed in horizontal scan lines. Bit 8 is in the Overflow register (index hex 07).

Vertical Retrace End Register

This register's index is hex 11; its address is hex 03D5 or 03B5.

Bit	Function
7	Protect Registers 0-7
6	Select 5 Refresh Cycles
5	-Enable Vertical Interrupt
4	-Clear Vertical Interrupt
3 - 0	Vertical Retrace End

Figure 58. Vertical Retrace End Register, Index Hex 11

Bit 7 When set to 1, this bit disables write access to the CRT controller registers at index 00 through 07. The line compare bit in the Overflow register (index hex 07) is not protected.

Bit 6 When set to 1, this bit generates five memory refresh cycles per horizontal line. When set to 0, this bit selects three refresh cycles. Selecting five refresh cycles allows use of the VGA chip with 15.75 kHz displays. This bit should be set to 0 for supported operations. It is set to 0 by a BIOS mode set, a reset, or a power on.

Bit 5 When set to 0, this bit enables a vertical retrace interrupt. The vertical retrace interrupt is IRQ2. This interrupt level can be shared; therefore, to determine whether the video generated the interrupt, check the CRT interrupt bit in Input Status Register 0.

Bit 4 When set to 0, this bit clears a vertical retrace interrupt. At the end of the active vertical display time, a flip-flop is set to indicate an interrupt. An interrupt handler resets this flip-flop by first setting this bit to 0, then resetting it to 1.

Bits 3 - 0 The Vertical Retrace Start register is compared with these four bits to determine where the 'vertical retrace' signal goes inactive. It is programmed in units of horizontal scan lines. To program these bits with a signal width of W, the following algorithm is used: the width W, in horizontal scan units, is added to the value in the Start Vertical Retrace register. The four low-order bits of the result are the 4-bit value programmed.

Vertical Display Enable End Register

This register's index is hex 12; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Vertical Display Enable End

Figure 59. Vertical Display Enable End Register, Index Hex 12

Bits 7 - 0 These are the eight low-order bits of a 10-bit value that defines the vertical-display-enable end position. The two high-order bits are contained in the Overflow register (index hex 07). The 10-bit value is equal to the total number of scan lines minus 1.

Offset Register

This register's index is hex 13; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Logical Line Width of the Screen

Figure 60. Offset Register, Index Hex 13

Bits 7 - 0 These bits specify the logical line width of the screen. The starting memory address for the next character row is larger than the current character row by 2 or 4 times the value of these bits. Depending on the method of clocking the CRT controller, this address is either a word or doubleword address.

Underline Location Register

This register's index is hex 14; its address is hex 03D5 or 03B5.

Bit	Function
7	Reserved
6	Doubleword Mode
5	Count By 4
4 - 0	Start Underline

Figure 61. Underline Location Register, Index Hex 14

Bit 7 Reserved.

Bit 6 When this bit is set to 1, memory addresses are doubleword addresses. See the description of the word/byte mode bit (bit 6) in the CRT Mode Control register on page 64.

Bit 5 When this bit is set to 1, the memory-address counter is clocked with the character clock divided by 4, which is used when doubleword addresses are used.

Bits 4 - 0 These bits specify the horizontal scan line of a character row on which an underline occurs. The value programmed is the scan line desired minus 1.

Start Vertical Blanking Register

This register's index is hex 15; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Start Vertical Blanking

Figure 62. Start Vertical Blanking Register, Index Hex 15

Bits 7 - 0 These are the eight low-order bits of a 10-bit value that specifies the starting location for the 'vertical blanking' signal. Bit 8 is in the Overflow register (index hex 07) and bit 9 is in the Maximum Scan Line register (index hex 09). The 10-bit value is the horizontal scan line count where the 'vertical blanking' signal becomes active minus 1.

End Vertical Blanking Register

This register's index is hex 16; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	End Vertical Blanking

Figure 63. End Vertical Blanking Register, Index Hex 16

Bits 7 - 0 This register specifies the horizontal scan count where the 'vertical blanking' signal becomes inactive. The register is programmed in units of the horizontal scan line.

To program these bits with a 'vertical blanking' signal of width W, the following algorithm is used: the width W, in horizontal scan line units, is added to the value in the Start Vertical Blanking register minus 1. The eight low-order bits of the result are the 8-bit value programmed.

CRT Mode Control Register

This register's index is hex 17; its address is hex 03D5 or 03B5.

Bit	Function
7	Hardware Reset
6	Word/Byte Mode
5	Address Wrap
4	Reserved
3	Count By Two
2	Horizontal Retrace Select
1	Select Row Scan Counter
0	CMS 0

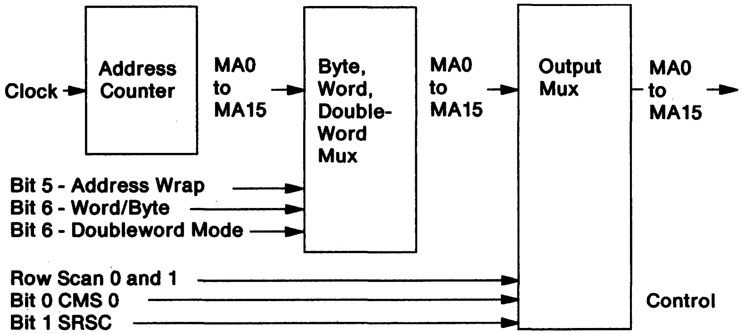
Figure 64. CRT Mode Control Register, Index Hex 17

Bit 7 When set to 0, this bit disables the horizontal and vertical retrace signals and forces them to an inactive level. When set to 1, this bit enables the horizontal and vertical retrace signals. This bit does not reset any other registers or signal outputs.

Bit 6 When this bit is set to 0, the word mode is selected. The word mode shifts the memory-address counter bits down one bit; the most-significant bit of the counter appears on the least-significant bit of the memory address outputs.

The doubleword bit in the Underline Location register (hex 14) also controls the addressing. When the doubleword bit is 0, the word/byte bit selects the mode. When the doubleword bit is set to 1, the addressing is shifted by two bits.

When set to 1, bit 6 selects the byte address mode. See the following figures for address output details.



Memory	**Modes of Addressing**		
Address Outputs	**Byte**	**Word**	**Doubleword**
MA 0/RFA 0	MA 0	MA 15 or 13	MA 12
MA 1/RFA 1	MA 1	MA 0	MA 13
MA 2/RFA 2	MA 2	MA 1	MA 0
MA 3/RFA 3	MA 3	MA 2	MA 1
MA 4/RFA 4	MA 4	MA 3	MA 2
MA 5/RFA 5	MA 5	MA 4	MA 3
MA 6/RFA 6	MA 6	MA 5	MA 4
MA 7/RFA 7	MA 7	MA 6	MA 5
MA 8/RFA 8	MA 8	MA 7	MA 6
MA 9	MA 9	MA 8	MA 7
MA 10	MA 10	MA 9	MA 8
MA 11	MA 11	MA 10	MA 9
MA 12	MA 12	MA 11	MA 10
MA 13	MA 13	MA 12	MA 11
MA 14	MA 14	MA 13	MA 12
MA 15	MA 15	MA 14	MA 13

Figure 65. CRT Memory Address Mapping

Bit 5 This bit selects the memory-address bit, bit MA 13 or MA 15, that appears on the output pin MA 0, in the word address mode. If the VGA is not in the word address mode, bit 0 from the address counter appears on the output pin, MA 0.

When set to 1, this bit selects MA 15. In odd/even mode, this bit should be set to 1 because 256KB of video memory is installed on the system board. (Bit MA 13 is selected in applications where only 64KB is present. This function maintains compatibility with the IBM Color/Graphics Monitor Adapter.)

Bit 4 Reserved.

Bit 3 When this bit is set to 0, the address counter uses the character clock. When this bit is set to 1, the address counter uses the character clock input divided by 2. This bit is used to create either a byte or word refresh address for the display buffer.

Bit 2 This bit selects the clock that controls the vertical timing counter. The clocking is either the horizontal retrace clock or horizontal retrace clock divided by 2. When this bit is set to 1, the horizontal retrace clock is divided by 2.

Dividing the clock effectively doubles the vertical resolution of the CRT controller. The vertical counter has a maximum resolution of 1024 scan lines because the vertical total value is 10-bits wide. If the vertical counter is clocked with the horizontal retrace divided by 2, the vertical resolution is doubled to 2048 scan lines.

Bit 1 This bit selects the source of bit 14 of the output multiplexer. When this bit is set to 0, bit 1 of the row scan counter is the source. When this bit is set to 1, the bit 14 of the address counter is the source.

Bit 0 This bit selects the source of bit 13 of the output multiplexer. When this bit is set to 0, bit 0 of the row scan counter is the source, and when this bit is set to 1, bit 13 of the address counter is the source.

The CRT controller used on the IBM Color/Graphics Adapter was capable of using 128 horizontal scan-line addresses. For the VGA to obtain 640-by-200 graphics resolution, the CRT controller is programmed for 100 horizontal scan lines with two scan-line addresses per character row. Row scan address bit 0 becomes the most-significant address bit to the display buffer. Successive scan lines of the display image are displaced in 8KB of memory. This bit allows compatibility with the graphics modes of earlier adapters.

Line Compare Register

This register's index is hex 18; its address is hex 03D5 or 03B5.

Bit	Function
7 - 0	Line Compare Target

Figure 66. Line Compare Register, Index Hex 18

Bits 7 - 0 These bits are the eight low-order bits of the 10-bit compare target. When the vertical counter reaches the target value, the internal start address of the line counter is cleared. This creates a split screen where the lower screen is immune to scrolling. Bit 8 is in the Overflow register (index hex 07), and bit 9 is in the Maximum Scan Line register (index hex 09).

Graphics Controller Registers

The Address register for the graphics controller is at address hex 03CE. The data registers are at address hex 03CF. All registers are read/write.

Register Name	Address (Hex)	Index (Hex)
Address	03CE	
Set/Reset	03CF	00
Enable Set/Reset	03CF	01
Color Compare	03CF	02
Data Rotate	03CF	03
Read Map Select	03CF	04
Graphics Mode	03CF	05
Miscellaneous	03CF	06
Color Don't Care	03CF	07
Bit Mask	03CF	08

Figure 67. Graphics Controller Register Overview

Address Register

The Address register is at address hex 03CE. This register is loaded with the index value that points to the desired data register within the graphics controller.

Bit	Function
7 - 4	Reserved
3 - 0	Register Index

Figure 68. Graphics Controller Address Register, Hex 03CE

Bits 7 - 4 Reserved.

Bits 3 - 0 These bits contain the index value that points to the data registers.

Set/Reset Register

This register's index is hex 00; its address is hex 03CF.

Bit	Function
7 - 4	Reserved
3	Set/Reset Map 3
2	Set/Reset Map 2
1	Set/Reset Map 1
0	Set/Reset Map 0

Figure 69. Set/Reset Register, Index Hex 00

Bits 7 - 4 Reserved.

Bits 3 - 0 When write mode 0 is selected, the system writes the value of each set/reset bit to its respective memory map. For each Write operation, the set/reset bit, if enabled, is written to all eight bits within that map. Set/reset operation can be enabled on a map-by-map basis through the Enable Set/Reset register.

Enable Set/Reset Register

The index for this register is hex 01; its address is hex 03CF.

Bit	Function
7 - 4	Reserved
3	Enable Set/Reset Map 3
2	Enable Set/Reset Map 2
1	Enable Set/Reset Map 1
0	Enable Set/Reset Map 0

Figure 70. Enable Set/Reset Register, Index Hex 01

Bits 7 - 4 Reserved.

Bits 3 - 0 These bits enable the set/reset function used when write mode 0 is selected in the Graphics Mode register (index hex 05). When the bit is set to 1, the respective memory map receives the value specified in the Set/Reset register. When Set/Reset is not enabled for a map, that map receives the value sent by the system.

Color Compare Register

This register's index is hex 02; its address is hex 03CF.

Bit	Function
7 - 4	Reserved
3	Color Compare Map 3
2	Color Compare Map 2
1	Color Compare Map 1
0	Color Compare Map 0

Figure 71. Color Compare Register, Index Hex 02

Bits 7 - 4 Reserved.

Bits 3 - 0 These bits are the 4-bit color value to be compared when the read mode bit in the Graphics Mode register is set to 1. When the system does a memory read, the data returned from the memory cycle will be a 1 in each bit position where the four maps equal the Color Compare register. If the read mode bit is 0, the data is returned without comparison.

The color compare bit is the value that all bits of the corresponding map's byte are compared with. Each of the eight bit positions in the selected byte are compared across the four maps, and a 1 is returned in each position where the bits of all four maps equal their respective color compare values.

Data Rotate Register

This register's index is hex 03; its address is hex 03CF.

Bit	Function
7 - 5	Reserved
4, 3	Function Select
2 - 0	Rotate Count

Figure 72. Data Rotate Register, Index Hex 03

Bits 7 - 5 Reserved.

Bits 4, 3 Data written to the video buffer can be operated on logically by data already in the system latches.

Data can be any of the choices selected by the write mode bits except system latches, which cannot be modified. If rotated data is selected also, the rotate is performed before the logical operation. The logical operations selected are shown in the following table.

Bits 4 3	Function
0 0	Data Unmodified
0 1	Data ANDed with Latched Data
1 0	Data ORed with Latched Data
1 1	Data XORed with Latched Data

Figure 73. Operation Select Bit Definitions

Bits 2 - 0 In write mode 0, these bits select the number of positions the system data is rotated to the right during a system Memory Write operation. To write data that is not rotated in mode 0, all bits are set to 0.

Read Map Select Register

This register's index is hex 04; its address is hex 03CF.

Bit	Function
7 - 2	Reserved
1, 0	Map Select

Figure 74. Read Map Select Register, Index Hex 04

Bits 7 - 2 Reserved.

Bits 1, 0 These bits select the memory map for system Read operations. This register has no effect on the color compare read mode. In odd/even modes, the value can be a binary 00 or 01 to select the chained maps 0, 1 and the value can be a binary 10 or 11 to select the chained maps 2, 3.

Graphics Mode Register

This register's index is hex 05; its address is hex 03CF.

Bit	Function
7	Reserved
6	256-Color Mode
5	Shift Register Mode
4	Odd/Even
3	Read Mode
2	Reserved
1, 0	Write Mode

Figure 75. Graphics Mode Register, Index Hex 05

Bit 7 Reserved.

Bit 6 When set to 0, this bit allows bit 5 to control the loading of the shift registers. When set to 1, this bit causes the shift registers to be loaded in a manner that supports the 256-color mode.

Bit 5 When set to 1, this bit directs the shift registers in the graphics controller to format the serial data stream with even-numbered bits from both maps on even-numbered maps, and odd-numbered bits from both maps on the odd-numbered maps. This bit is used for modes 4 and 5.

Bit 4 When set to 1, this bit selects the odd/even addressing mode used by the IBM Color/Graphics Monitor Adapter. Normally, the value here follows the value of Memory Mode register bit 2 in the sequencer.

Bit 3 When this bit is set to 1, the system reads the results of the comparison of the four memory maps and the Color Compare register.

When this bit is set to 0, the system reads data from the memory map selected by the Read Map Select register, or by the two low-order bits of the memory address (this selection depends on the chain-4 bit in the Memory Mode register of the sequencer).

Bit 2 Reserved.

Bits 1, 0 The write mode selected and its operation are defined in the following figure. The logic operation specified by the function select bits is performed on system data for modes 0, 2, and 3.

Bits	
1 0	**Mode Description**
0 0	Each memory map is written with the system data rotated by the count in the Data Rotate register. If the set/reset function is enabled for a specific map, that map receives the 8-bit value contained in the Set/Reset register.
0 1	Each memory map is written with the contents of the system latches. These latches are loaded by a system Read operation.
1 0	Memory map *n* (0 through 3) is filled with eight bits of the value of data bit *n*.
1 1	Each memory map is written with the 8-bit value contained in the Set/Reset register for that map (the Enable Set/Reset register has no effect). Rotated system data is ANDed with the Bit Mask register to form an 8-bit value that performs the same function as the Bit Mask register in write modes 0 and 2 (see also Bit Mask register on page 75).

Figure 76. Write Mode Definitions

Miscellaneous Register

This register's index is hex 06; its address is hex 03CF.

Bit	Function
7 - 4	Reserved
3	Memory Map 1
2	Memory Map 0
1	Odd/Even
0	Graphics Mode

Figure 77. Miscellaneous Register, Index Hex 06

Bits 7 - 4 Reserved.

Bits 3, 2 These bits control the mapping of the regenerative buffer into the system address space. The bit functions are defined in the following figure.

Bits 3 2	Addressing Assignment
0 0	A0000 for 128KB
0 1	A0000 for 64KB
1 0	B0000 for 32KB
1 1	B8000 for 32KB

Figure 78. Video Memory Assignments

Bit 1 When set to 1, this bit directs the system address bit, A0, to be replaced by a higher-order bit. The odd map is then selected when A0 is 1, and the even map when A0 is 0.

Bit 0 This bit controls alphanumeric mode addressing. When set to 1, this bit selects graphics modes, which also disables the character generator latches.

Color Don't Care Register

This register's index is hex 07; its address is hex 03CF.

Bit	Function
7 - 4	Reserved
3	Map 3 is Don't Care
2	Map 2 is Don't Care
1	Map 1 is Don't Care
0	Map 0 is Don't Care

Figure 79. Color Don't Care Register, Index Hex 07

Bits 7 - 4 Reserved.

Bits 3 - 0 These bits select whether a map is going to participate in the color compare cycle. When the bit is set to 1, the bits in that map are compared.

Bit Mask Register

This register's index is hex 08; its address is hex 03CF.

Bit	Function
7 - 0	-Bit Mask 7 - 0

Figure 80. Bit Mask Register, Index Hex 08

Bits 7 - 0 When the bit is set to 1, the corresponding bit position in each map can be changed. When the bit set to 0, the bit position in the map is masked to prevent change, provided that the location being written was the last location read by the system microprocessor.

The bit mask applies to write modes 0 and 2. To preserve bits using the bit mask, data must be latched internally by reading the location. When data is written to preserve the bits, the most current data in the latches is written in those positions. The bit mask applies to all maps simultaneously.

Attribute Controller Registers

Each register for the attribute controller has two addresses. Address hex 03C0 is the write address and hex 03C1 is the read address. The individual data registers are selected by writing their index to the Address register.

Register Name	Write Address	Read Address	Index
Address	03C0	03C1	-
Internal Palette			00 - 0F
Attribute Mode Control			10
Overscan Color			11
Color Plane Enable			12
Horizontal PEL Panning			13
Color Select			14

Figure 81. Attribute Controller Register Addresses

Address Register

This register is read through address hex 03C1, and written to through address hex 03C0.

The attribute controller registers do not have an input bit to control selection of the address and data registers. An internal address flip-flop controls this selection. Reading Input Status Register 1 clears the flip-flop and selects the Address register.

After the Address register has been loaded with the index, the next Write operation to 03C0 loads the data register. The flip-flop toggles for each Write operation to address hex 03C0. It does not toggle for Read operations to 03C1. (Also see "VGA Programming Considerations" on page 83.)

Bit	Function
7, 6	Reserved
5	Internal Palette Address Source
4 - 0	Register Index

Figure 82. Address Register, Hex 03C0

Bits 7, 6 Reserved.

Bit 5 This bit is set to 0 to load color values to the registers in the internal palette. For normal operation of the attribute controller, set this bit to 1, which allows the video data to use the internal palette for output.

Bits 4 - 0 These bits contain the index to the data registers in the attribute controller.

Internal Palette Registers 0 through F

These registers are at indexes hex 00 through 0F. Their write address is hex 03C0; their read address is hex 03C1.

Bit	Function
7, 6	Reserved
5	P5
4	P4
3	P3
2	P2
1	P1
0	P0

Figure 83. Internal Palette Registers, Index Hex 00 - 0F

Bits 7, 6 Reserved.

Bits 5 - 0 These 6-bit registers allow a dynamic mapping between the text attribute or graphic color input value and the display color on the CRT screen. When set to 1, this bit selects the appropriate color. The Internal Palette registers should be modified only during the vertical retrace interval to avoid problems with the displayed image. These internal palette values are sent off-chip to the video DAC, where they serve as addresses into the DAC registers. (Also see the attribute controller block diagram on page 7.)

Attribute Mode Control Register

This read/write register is at index hex 10. Its write address is hex 03C0; its read address is hex 03C1.

Bit	Function
7	P5, P4 Select
6	PEL Width
5	PEL Panning Compatibility
4	Reserved
3	Enable Blink/-Select Background Intensity
2	Enable Line Graphics Character Code
1	Mono Emulation
0	Graphics/-Alphanumeric Mode

Figure 84. Attribute Mode Control Register, Index Hex 10

Bit 7 This bit selects the source for the P5 and P4 video bits that act as inputs to the video DAC. When this bit is set to 0, P5 and P4 are the outputs of the Internal Palette registers. When this bit is set to 1, P5 and P4 are bits 1 and 0 of the Color Select register. For more information, refer to "VGA Programming Considerations" on page 83.

Bit 6 When this bit is set to 1, the video data is sampled so that eight bits are available to select a color in the 256-color mode (hex 13). This bit is set to 0 in all other modes.

Bit 5 When this bit is set to 1, a successful line-compare in the CRT controller forces the output of the PEL Panning register to 0 until a vertical synchronization occurs, at which time the output returns to its programmed value. This bit allows a selected portion of a screen to be panned.

When this bit is set to 0, line compare has no effect on the output of the PEL Panning register.

Bit 4 Reserved.

Bit 3 When this bit is set to 0, the most-significant bit of the attribute selects the background intensity (allows 16 colors for background). When set to 1, this bit enables blinking.

Bit 2 When this bit is set to 0, the ninth dot will be the same as the background. When set to 1, this bit enables the special line-graphics character codes for the monochrome emulation mode. This emulation mode forces the ninth dot of a line graphic character to be identical to the eighth dot of the character. The line-graphics character codes for the monochrome emulation mode are hex C0 through hex DF.

For character fonts that do not utilize these line-graphics character codes, bit 2 should be set to 0 to prevent unwanted video information from displaying on the CRT screen.

BIOS will set this bit, the correct dot clock, and other registers when the 9-dot alphanumeric mode is selected.

Bit 1 When this bit is set to 1, monochrome emulation mode is selected. When this bit is set to 0, color emulation mode is selected.

Bit 0 When set to 1, this bit selects the graphics mode of operation.

Overscan Color Register

This read/write register is at index hex 11. Its write address is hex 03C0; its read address is hex 03C1. This register determines the border (overscan) color.

Bit	Function
7 - 0	P7 - P0

Figure 85. Overscan Color Register, Index Hex 11

Bits 7 - 0 These bits select the border color used in the 80-column alphanumeric modes and in the graphics modes other than modes 4, 5, and D.

Color Plane Enable Register

This read/write register is at index hex 12. Its write address is hex 03C0; its read address is hex 03C1.

Bit	Function
7 - 4	Reserved
3 - 0	Enable Color Plane

Figure 86. Color Plane Enable Register, Index Hex 12

Bits 7 - 4 Reserved.

Bits 3 - 0 Setting a bit to 1, enables the corresponding display-memory color plane.

Horizontal PEL Panning Register

This read/write register is at index hex 13. Its write address is hex 03C0; its read address is hex 03C1.

Bit	Function
7 - 4	Reserved
3 - 0	Horizontal PEL Panning

Figure 87. Horizontal PEL Panning Register, Index Hex 13

Bits 7 - 4 Reserved.

Bits 3 - 0 These bits select the number of PELs that the video data is shifted to the left. PEL panning is available in both alphanumeric and graphics modes. The following figure shows the number of bits shifted for each mode.

	Number of PELs Shifted to the Left		
Register Value	Mode 13	A/N Modes *	All Other Modes
0	0	1	0
1	-	2	1
2	1	3	2
3	-	4	3
4	2	5	4
5	-	6	5
6	3	7	6
7	-	8	7
8	-	0	-

* Only mode 7 and the A/N modes with 400 scan lines.

Figure 88. Image Shifting

Color Select Register

This read/write register is at index hex 14. Its write address is hex 03C0; its read address is hex 03C1.

Bit	Function
7 - 4	Reserved
3	S_color 7
2	S_color 6
1	S_color 5
0	S_color 4

Figure 89. Color Select Register, Index Hex 14

Bits 7 - 4 Reserved.

Bits 3, 2 In modes other than mode hex 13, these are the two most-significant bits of the 8-bit digital color value to the video DAC. In mode hex 13, the 8-bit attribute is the digital color value to the video DAC. These bits are used to rapidly switch between sets of colors in the video DAC. (For more information, refer to “VGA Programming Considerations” on page 83.)

Bits 1, 0 These bits can be used in place of the P4 and P5 bits from the Internal Palette registers to form the 8-bit digital color value to the video DAC. Selecting these bits is done in the Attribute Mode Control register (index hex 10). These bits are used to rapidly switch between colors sets within the video DAC.

VGA Programming Considerations

The following are some programming considerations for the VGA:

- The following rules must be followed to guarantee the critical timings necessary to ensure the proper operation of the CRT controller:
  - The value in the Horizontal Total register must be at least hex 19.
  - The minimum positive pulse width of the 'horizontal synchronization' signal must be four character clock units.
  - The End Horizontal Retrace register must be programmed such that the 'horizontal synchronization' signal goes to 0 at least one character clock time before the 'horizontal display enable' signal goes active.
  - The End Vertical Blanking register must be set to a minimum of one horizontal scan line greater than the line-compare value.
- When PEL panning compatibility is enabled in the Attribute Mode Control register, a successful line compare in the CRT controller forces the output of the Horizontal PEL Panning register to 0′s until a vertical synchronization occurs. When the vertical synchronization occurs, the output returns to the programmed value. This allows the portion of the screen indicated by the Line Compare register to be operated on by the Horizontal PEL Panning register.
- A write to the Character Map Select register becomes valid on the next whole character line. This will prevent deformed character images when changing character generators in the middle of a character scan line.
- For mode hex 13, the attribute controller is configured so that the 8-bit attribute in video memory becomes the 8-bit address (P0 - P7) into the video DAC. The user should not modify the contents of the Internal Palette registers when using this mode.

- The following is the sequence for accessing the attribute data registers:

  1. Disable interrupts.
  2. Reset the flip-flop for the Attribute Address register.
  3. Write the index.
  4. Access the data register.
  5. Enable interrupts.

- The Color Select register in the attribute controller section allows the programmer to rapidly switch color sets in the video DAC. Bit 7 of the Attribute Mode Control register controls the number of bits in the Color Select register used to address the color information in the video DAC (either two or four bits are used). By changing the value in the Color Select register, an application can switch color sets in graphics and alphanumeric modes (mode hex 13 does not use this feature).

  Note: For multiple color sets, the user must load the color values.

- An application that saves the video state must store the four bytes of information contained in the system microprocessor latches in the graphics controller subsection. These latches are loaded with 32 bits from video memory (8 bits per map) each time the system reads from video memory. The application needs to:

  1. Use write mode 1 to write the values in the latches to a location in video memory that is not part of the display buffer, such as the last location in the address range.
  2. Save the values of the latches by reading them back from video memory.

     Note: If memory addressing is in the chain-4 or odd/even mode, reconfigure the memory as four sequential maps prior to performing the sequence above.

BIOS provides support for completely saving and restoring the video state. Refer to the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for more information.

- The Horizontal PEL Panning register controls the number of PELs shifted left. Further panning, beyond that shown under the register control, can be accomplished by changing the start-address value in the CRT Controller registers, Start Address High and Start Address Low. The sequence is:

  1. Use the Horizontal PEL Panning register to shift the maximum number of bits to the left.
  2. Increment the start address.
  3. Set the Horizontal PEL Panning register so that no bits are shifted.

     The screen will now be shifted one PEL to the left of the position it was in at the end of Step 1. Step 1 through Step 3 are repeated as often as necessary.

- When using a split-screen application that scrolls a second screen on top of the first screen and operating in a mode with 200 scan lines, the Line Compare register (CRT Controller register hex 19) must contain an even value. This is a requirement of the double scanning logic in the CRT controller.

- If the value in the Cursor Start register (CRT Controller register hex 0A) is greater than that in the Cursor End register (CRT Controller register hex 0B), the cursor is not displayed.

- In 8-dot character modes, the underline attribute produces a solid line across adjacent characters. In 9-dot character modes, the underline across adjacent characters is dashed. In 9-dot modes with the line-graphics characters (C0 - DF character codes), the underline is solid.

Programming the Registers

Each of the video components has an address register and a number of data registers. The data registers have addresses common to all registers for that component. The individual registers are selected by a pointer (index) in its Address register. To write to a data register, the Address register is loaded with the index of the desired data register, then the data register is loaded by writing to the common I/O address.

The general registers do not share a common address; they each have their own I/O address.

See "Video DAC to System Interface" on page 91 for details on programming the video DAC.

For compatibility with the IBM Enhanced Graphics Adapter (EGA), the internal video subsystem palette is programmed the same as the EGA. Using BIOS to program the palette will produce a color compatible to that produced by the EGA. Mode hex 13 (256 colors) is programmed so that the first 16 locations in the DAC produce compatible colors.

When BIOS is used to load the color palette for a color mode and a monochrome display is attached, the color palette is changed. The colors are summed to produce shades of gray that allow color applications to produce a readable screen.

Modifying the following bits must be done while the sequencer is held in a synchronous reset through its Reset register. The bits are:

- Bits 3 and 0 of the Clocking Mode register.
- Bits 3 and 2 of the Miscellaneous Output register.

RAM Loadable Character Generator

The character generator is RAM loadable and can support characters up to 32 scan lines high. Three character fonts are stored in BIOS, and one is automatically loaded when an alphanumeric mode is selected. The Character Map Select register can be programmed to redefine the function of bit 3 of the attribute byte to be a character-font switch. This allows the user to select between any two character sets residing in map 2, and effectively gives the user access to 512 characters instead of 256. Character fonts can be loaded offline, and up to eight fonts can be loaded at any one time.

The structure of the character fonts is described in the following figure. The character generator is in map 2 and must be protected using the map mask function.

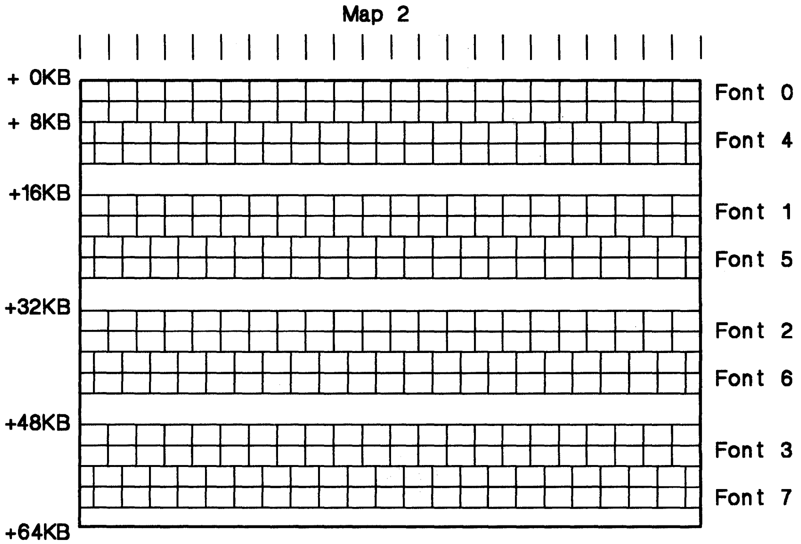


Figure 90. Character Table Structure

The following figure illustrates the structure of each character pattern. If the CRT controller is programmed to generate 16 row scans, then 16 bytes must be filled in for each character in the font. The example below assumes eight row scans per character.

Address	Byte Image								Data
CC * 32 + 0				X	X				18H
1			X	X	X	X			3EH
2		X	X			X	X		66H
3		X	X			X	X		66H
4		X	X	X	X	X	X		7EH
5		X	X			X	X		66H
6		X	X			X	X		66H
7		X	X			X	X		66H

Figure 91. Character Pattern Example

CC equals the value of the character code. For example, hex 41 equals an ASCII "A."

Creating a Split Screen

The VGA hardware supports a split screen. The top portion of the screen is designated as screen A, and the bottom portion is designated as screen B, as in the following figure.

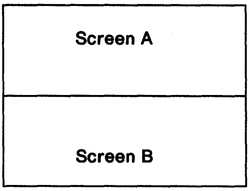


Figure 92. Split Screen Definition

The following figure shows the screen mapping for a system containing a 32KB alphanumeric storage buffer, such as the VGA.

Information displayed on screen A is defined by the Start Address High and Low registers of the CRT controller. Information displayed on screen B always begins at video address hex 0000.

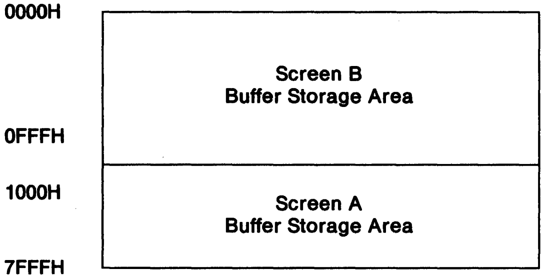


Figure 93. Screen Mapping within the Display Buffer Address Space

The Line Compare register of the CRT controller performs the split screen function. The CRT controller has an internal horizontal scan line counter and logic that compares the counter value to the value in the Line Compare register and clears the memory address generator when a comparison occurs. The linear address generator then sequentially addresses the display buffer starting at location 0. Each subsequent row address is determined by the 16-bit addition of the start-of-line latch and the Offset register.

Screen B can be smoothly scrolled onto the display by updating the Line Compare register in synchronization with the 'vertical retrace' signal. Screen B information is not affected by scrolling operations that use the Start Address registers to scroll through the screen A information.

When PEL-panning compatibility is enabled (Attribute Mode Control register), a successful line comparison forces the output of the Horizontal PEL Panning register to 0′s until vertical synchronization occurs. This feature allows the information on screen B to remain unaffected by PEL-panning operations on screen A.

Video Digital-to-Analog Converter

The video digital-to-analog converter (DAC) integrates the function of a color palette with three internal DACs for driving an analog display.

The DAC has 256 registers containing 18 bits each to allow the display of up to 256 colors from a possible 256K colors. Each output signal is driven by a 6-bit DAC.

Register Name	R/W	Address (in Hex)
Palette Address (Write Mode)	R/W	03C8
Palette Address (Read Mode)	W	03C7
DAC State	R	03C7
Palette Data	R/W	03C9
PEL Mask	R	03C6

Figure 94. Video DAC Register

Device Operation

The palette address (P7 - P0) and the blanking input are sampled on the rising edge of the PEL clock. After three more PEL clock cycles, the video reflects the state of these inputs.

During normal operation the palette address is used as a pointer to one of the 256 data registers in the palette. The value in each data register is converted to an analog signal for each of the three outputs (red, green, blue). The blanking input is used to force the video output to 0 volts. The blanking operation is independent of the palette operation.

Each data register is 18 bits wide: 6 bits each for red, green, and blue. The data registers are accessible through the system interface.

Video DAC to System Interface

The Palette Address register holds an 8-bit value that is used to address a location within the video DAC. The Palette Address register responds to two addresses; the address depends on the type of palette access, Read or Write. Once the address is loaded, successive accesses to the data register automatically increment the address register.

For palette Write operations, the address for the Palette Address register is hex 03C8. A write cycle consists of three successive bytes written to the Data register at address hex 03C9. The six least-significant bits of each byte are concatenated to form the 18-bit palette data. The order is red value first, then green, then blue.

For palette Read operations, the address for the Palette Address register is hex 03C7, which is read only. A read cycle consists of three successive bytes read from the Data register at address hex 03C9. The six least-significant bits of each byte contain the corresponding color value. The order is red value first, then green, then blue.

If the Palette Address register is written to during a Read or Write cycle, a new cycle is initialized and the unfinished cycle is terminated. The effects of writing to the Data register during a Read cycle or reading from the Data register during a Write cycle are undefined and can change the palette contents.

The DAC State register is a read-only register at address hex 03C7. Bits 1 and 0 return the last active operation to the DAC. If the last operation was a Read operation, both bits are set to 0. If the last operation was a Write, both bits are set to 1.

Reading the Read Palette Address register at hex 03C8 or the DAC State register at hex 03C7 does not interfere with read or write cycles.

Programming Considerations

- As explained in "Video DAC to System Interface," the effects of writing to the Data register during a read cycle or reading from the Data register during a write cycle are undefined and may change the palette contents. Therefore, the following sequence must be followed to ensure the integrity of the color palette during accesses to it:

  1. Write the address to the PEL Address register.
  2. Disable Interrupts.
  3. Write or read three bytes of data.
  4. Go to Step 3, repeat for the desired number of locations.
  5. Enable interrupts.

  Note: The above sequence assumes that any interrupting process will return the DAC in the correct mode (write or read). If this is not the case, the sequence shown below should be followed:

  1. Disable interrupts.
  2. Write the address to PEL Address register.
  3. Write or read three bytes of data.
  4. Go to Step 2, repeat for the desired number of locations.
  5. Enable interrupts.

- The minimum time from one Read or Write command to the DAC to the next Read or Write command is 6 dot clocks. Depending on how the dot clock is derived, this time can be up to 480 nanoseconds (dot clock divided by 2). To ensure enough time, assembler language programmers can place a JMP $+2 between successive accesses to the DAC.

- To prevent “snow” on the screen, an application reading data from or writing data to the DAC registers should ensure that the blank input to the DAC is asserted. This can be accomplished either by restricting data transfers to retrace intervals (use Input Status register 1 to determine when retrace is occurring) or by using the Screen Off bit located in the Clocking Mode register in the sequencer.

  Note: BIOS provides read and write interfaces to the Video DAC.

- Do not write to the PEL Mask register (hex 03C6). Palette information can be changed as a result. This register is correctly initialized to hex FF during a mode set.

Auxiliary Video Connector

The auxiliary video connector is a 20-pin connector located in-line with one of the channel connectors on the system board. This connector allows video data to be passed to and from an adapter. The video buffers can be turned off, and video output from the adapter can be sent through the video DAC to the display connector. The full channel is available for use by the adapter. Video output can be passed in only one direction at a time. The 'dot clock' signal cannot drive both EXTCLK to the VGA and PCLK to the DAC.

The pin assignments and a functional block diagram of the auxiliary video connector and timing diagrams for the display control signals appear on the following pages.

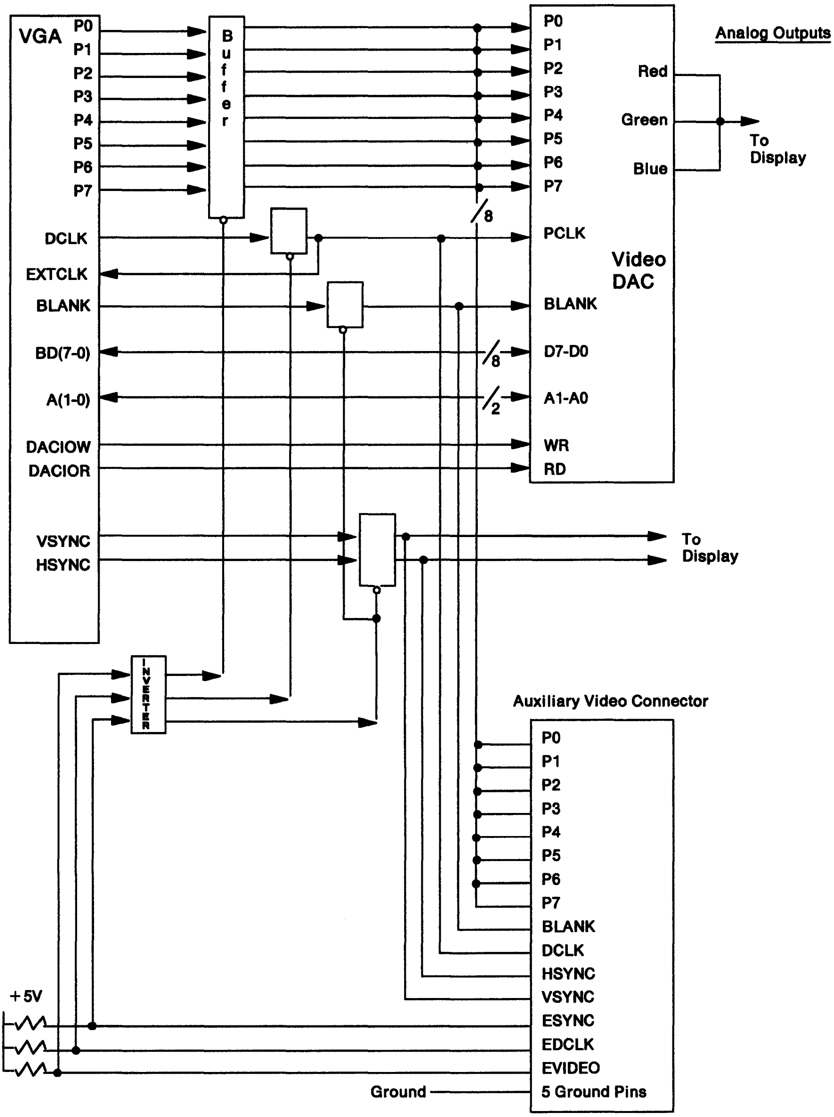


Figure 95. Auxiliary Video Connector

Signal Descriptions

The following are signal descriptions for the Auxiliary Video extension of the channel connector.

VSYNC: Vertical Synchronization: This signal is the vertical synchronization signal to the display. Also see the ESYNC description.

HSYNC: Horizontal Synchronization: This signal is the horizontal synchronization signal to the display. Also see the ESYNC description.

BLANK: Blanking Signal: This signal is connected to the BLANK input of the video DAC. When active (0 Vdc), this signal tells the DAC to drive its analog color outputs to 0 Vdc. Also see the ESYNC description.

P7 - P0: Palette Bits: These eight signals contain video information and comprise the PEL address inputs to the video DAC. See also the EVIDEO description.

DCLK: Dot Clock: This signal is the PEL clock used by the DAC to latch the digital video signals, P7 through P0. The signals are latched into the DAC on the rising edge of DCLK.

This signal is driven through the EXTCLK input to the VGA when DCLK is driven by the adapter. If an adapter is providing the clock, it must also provide the video data to the DAC. Also see the EDCLK description.

ESYNC: External Synchronization: This signal is the output-enable signal for the buffer that drives BLANK, VSYNC, and HSYNC. ESYNC is tied to +5 Vdc through a pull-up resistor. When ESYNC is high, the VGA drives BLANK, VSYNC, andHSYNC. When ESYNC is pulled low, the adapter drives BLANK, VSYNC, andHSYNC.

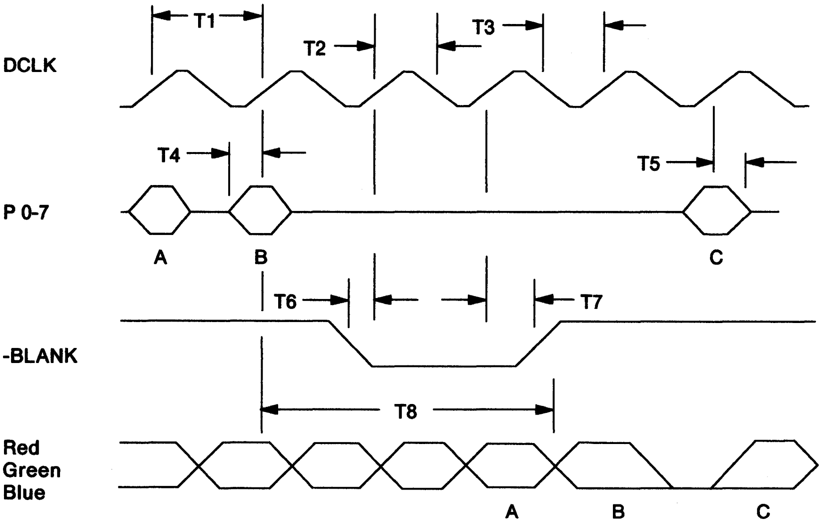
EVIDEO: External Video: This signal is the output-enable signal for the buffer that drives P7 through P0. EVIDEO is tied to +5 Vdc through a pull-up resistor. When EVIDEO is high, the VGA drives P7 through P0. When it is pulled low, the adapter drives P7 through P0.

EDCLK: External Dot Clock: This signal is the output-enable signal for the buffer that drives DCLK. EDCLK is tied to +5 Vdc through a pull-up resistor.

When EDCLK is high, the VGA is the source of DCLK to the DAC and the adapter. The Miscellaneous Output register (see "System Board I/O Controllers") should not select clock source 2 (010 binary) when EDCLK is high.

When EDCLK is pulled low, the adapter drives DCLK. If the adapter is driving the clock, it must also provide the video data to the DAC, and the Miscellaneous Output register must select clock source 2 (010 binary).

Connector Timing



Symbol	Description	Min.(ns)	Max.(ns)
T1	PEL Clock Period	28	10,000
T2	Clock Pulse Width High	7	10,000
T3	Clock Pulse Width Low	9	10,000
T4	PEL Set-Up Time	4	-
T5	PEL Hold Time	4	-
T6	Blank Set-Up Time	4	-
T7	Blank Hold Time	4	-
T8	Analog Output Delay	3(T1) + 5	3(T1) + 30

Figure 96. Auxiliary Video Connector Timing (DAC Signals)

Display Connector

The synchronization signals to the display are TTL levels. The video signals are 0 to 0.7 volts.

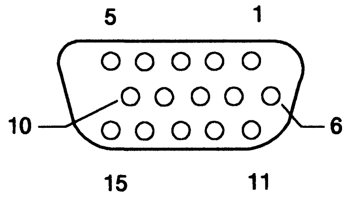


Figure 97. Display Connector

			Display Type	
Pin	**I/O**	**Output**	**Monochrome**	**Color**
1	O	Red	No Pin	Red
2	O	Green	Mono	Green
3	O	Blue	No Pin	Blue
4	N/A	Reserved	No Pin	No Pin
5	N/A	Ground	Self Test	Self Test
6	N/A	Red Ground	Dummy Pin	Red Ground
7	N/A	Green Ground	Mono Ground	Green Ground
8	N/A	Blue Ground	No Pin	Blue Ground
9	N/A	Plug	No Pin	No Pin
10	N/A	Ground	Ground	Ground
11	I	Monitor Sense 0	No Pin	Ground
12	I	Monitor Sense 1	Ground	No Pin
13	O	Hsync	Hsync	Hsync
14	O	Vsync	Vsync	Vsync
15	N/A	Reserved	No Pin	No Pin

Figure 98. Display Connector Signals

Signal Timing

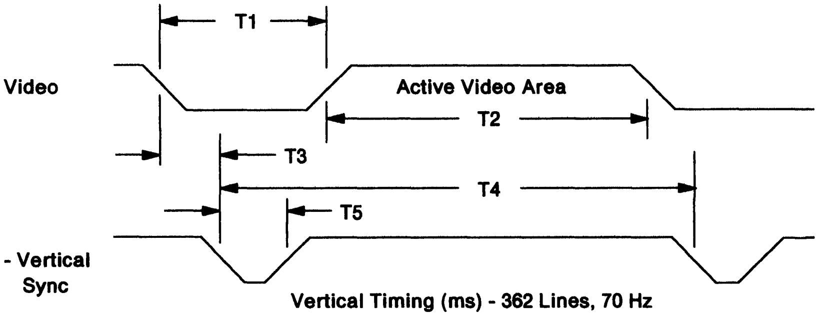
BIOS sets the video subsystem according to the video mode. All modes use a 70-Hz, vertical retrace except for modes 11 and 12. These two modes use 60 Hz.

The video subsystem generates the signal timings required by the displays according to the mode selected. The following timing diagrams represent only the vertical frequencies set by BIOS.

Note: The vertical size of the display is encoded using the polarity of the synchronization signals, as shown in the following figure.

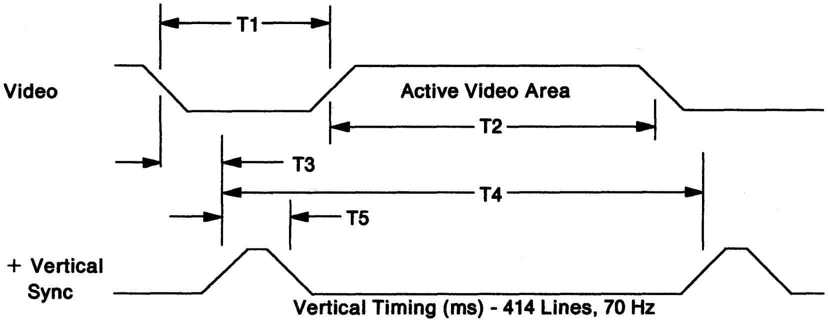
VSYNC Polarity	HSYNC Polarity	Vertical Size
+	+	Reserved
+	–	400 lines
–	+	350 lines
–	–	480 lines

Figure 99. Vertical Size



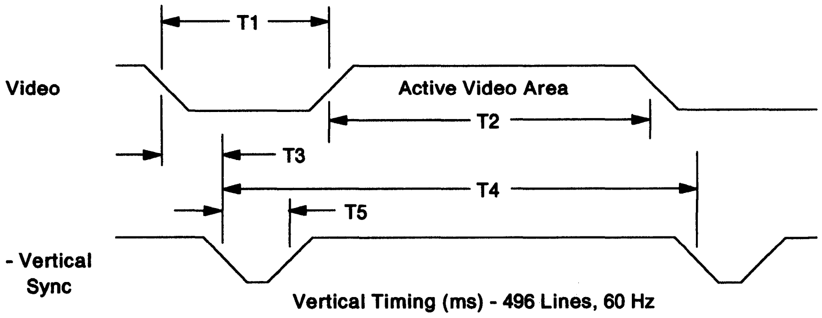
Symbol	Signal Time
T1	2.765 ms
T2	11.504 ms
T3	0.985 ms
T4	14.268 ms
T5	0.064 ms

Figure 100. Vertical Timing, 350 Lines



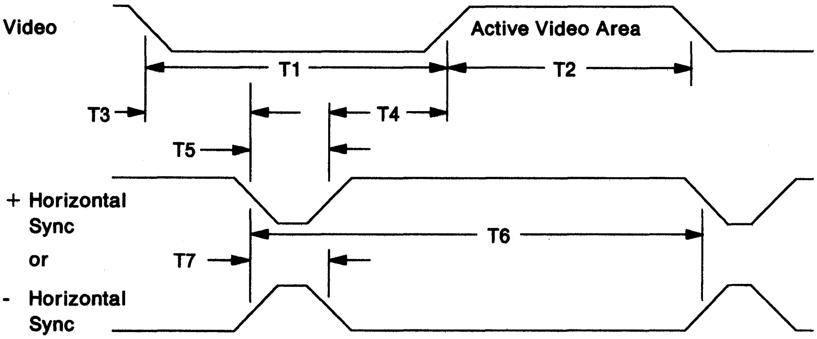
Symbol	Signal Time
T1	1.112 ms
T2	13.156 ms
T3	0.159 ms
T4	14.268 ms
T5	0.064 ms

Figure 101. Vertical Timing, 400 Lines



Symbol	Signal Time
T1	0.922 ms
T2	15.762 ms
T3	0.064 ms
T4	16.683 ms
T5	0.064 ms

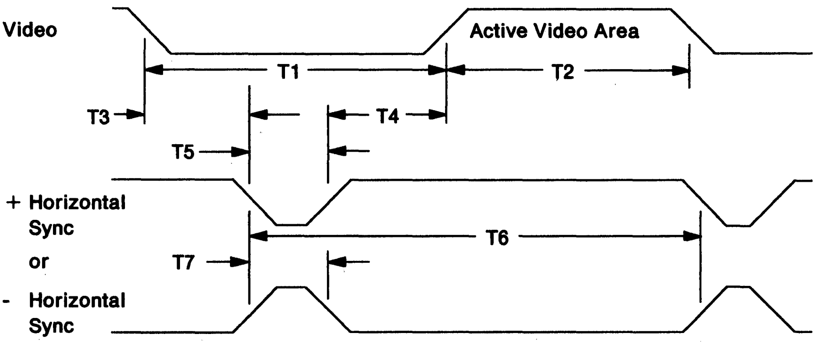
Figure 102. Vertical Timing, 480 Lines



Horizontal Timing(μs) – 80 Column, With Border

Symbol	Signal Time
T1	5.720 μs
T2	26.058 μs
T3	0.318 μs
T4	1.589 μs
T5	3.813 μs
T6	31.778 μs
T7	3.813 μs

Figure 103. Horizontal Timing, 80 Column with Border



Horizontal Timing(μs) — 40/80 Column, No Border

Symbol	Signal Time
T1	6.356 μs
T2	25.422 μs
T3	0.636 μs
T4	1.907 μs
T5	3.813 μs
T6	31.778 μs
T7	3.813 μs

Figure 104. Horizontal Timing, 40/80 Column, without Border

Notes:

Index

A

address registers
  attribute controller 76
  palette 90
  sequencer 42
address select 74
all points addressable (APA) modes 14
  mds.graphics 14
alphanumeric memory mapping 22
alphanumeric modes 11
AND data select 71
asynchronous reset 43
attribute controller 7
  blink enable 78
  block diagram 7
  registers 76
attribute controller registers 76
  address 76
  color plane enable 80
  color select 82
  horiz PEL panning 81
  internal palette 0-F 77
  mode control 78
  overscan color 80
  reserved bits 37
attribute definition 12
auxiliary video connector 94
  connector timing 98
  signals 96

B

BIOS 4
BIOS video modes 8
bit mask register 75
blanking signal 96
blink enable 78
block diagram
  address mapping 65
  attribute controller 7
  auxiliary video connector 95
  graphics controller 6
  memory read 36
  memory write 35
  subsystem 3
border color select 80
buffer address 59
buffer address select 74

C

chain 4 bit 48
character generator 46, 79
character generator, RAM loadable 87
character map select register 46
clocking mode register 43
color compare operations 36
color compare register 70
color don't care register 75
color graphics modes 640 x 480, 2 color 18
color modes 8
color plane enable register 80
color select register 82
color/graphics modes 14
compare, line 67
compatibility 86
components, subsystem 4

connector timing 98
connector, display 99
considerations, programming 83
control cursor 58
control cursor skew 58
controller
  attribute 7
  CRT 5
  graphics 5
count by two 64
count by 4 62
creating a split screen 88
CRT controller 5
  address mapping 65
  horiz skew 52
  registers 49
CRT controller compatibility 67
CRT controller registers 49
  address 50
  cursor location 59
  cursor start and end 58
  end horiz blanking 51
  end horiz retrace 53
  end vert blank 63
  horizontal end 51
  horizontal total 50
  line compare 67
  max scan line 57
  mode control 64
  offset 62
  overflow 55
  preset row scan 56
  reserved bits 37
  start address 59
  start horiz blanking 51
  start horiz retrace 53
  start vert blank 63
  start vert total 54
  underline location 62
  vert retrace end 60
  vert retrace start 60
  vertical end 61
CRT mode control register 64
cursor location register 59
cursor off bit 58
cursor skew 58
cursor start/end register 58

D

DAC palette 90
  operation 90
  programming considerations 92
  register 90
  state register 91
DAC state register 91
data rotate register 71
DCLK signal 96
description 1
device operation, DAC 90
digital-to-analog converter (DAC) 90
display connector 99
display connector timing 100
display support 10
display synchronization signals 100
display, vertical gain 39
display, vertical size 100
don't care, color 75
double scanning 57
doubleword mode 62

E

EDCLK signal 97
enable bit, video subsystem 10
enable blink 78
enable line graphics 79
enable register 41
enable set/reset register 69
end horizontal blanking register 51
end horizontal retrace register 53
end vertical blanking register 63
ESYNC signal 96

EVIDEO signal 97

F

feature control 41
font, RAM loadable 87
format, video
  mode F 17
  mode 11 18
  mode 13 19
  mode 4,5 14
  mode 6 16
  16 color modes 18
full-screen updates 44

G

general description 1
general registers 38
  input status 40, 41
  miscellaneous output 38
  reserved bits 37
  video enable 41
graphics controller 5
  block diagram 6
  register 68
graphics controller registers 68
  bit mask 75
  color compare 70
  color don't care 75
  data rotate 71
  enable set/reset 69
  miscellaneous 74
  mode 72
  read map select 72
  reserved bits 37
  set/reset 69
graphics mode register 72
graphics mode, 256-color 18
graphics modes 14
graphics modes 640 x 200, 2-color 16
graphics modes 640 x 350 17
graphics modes, 16-color 18
graphics modes, 320 x 200, four-color 14
guidelines, programming 83

H

horizontal display enable end register 51
horizontal PEL panning register 81
horizontal retrace select 64
horizontal skew 52
horizontal total register 50
HSYNC signal 96

I

image shift 81
input status register 40
interface, system to DAC 91
internal palette select 77
interrupt enable 61

L

line compare register 67
line graphics 79
line graphics character 78
logic, support 4
logical operator select 71

M

major components 4
map mask register 45
map select register 46
map select, read 72
mapping, memory 65
mapping, split screen 89
maps, memory 21
maximum scan line register 57
memory address select 74
memory maps 21, 65
memory mode register 48

memory read/writes 35
methods for programming 86
miscellaneous output register 38
miscellaneous register 74
mode control register (CRT) 64
mode control register, attribute 78
mode hex F, 640 x 350 graphics 17
mode hex 11, 640 x 480 two color graphics 18
mode hex 13, 256-color 18
mode register, graphics 72
mode 16-color graphics 18
mode 6, 640 x 200 two-color 16
modes
- alphanumeric 11
- BIOS 8
- count by two 64
- count by 4 62
- doubleword 62
- memory maps 20
- odd/even 73
- read and write 73
- word/byte 64
- 512 char font 87

modes 4 and 5, 320 x 200 four-color 14
modes, new 1
monochrome modes 8

N

new modes 1

O

odd/even address bit 48
odd/even mode 73
offset register 62
OR data select 71
organization, video memory 20, 21
overflow register, CRT controller 55
overscan color register 80

P

palette address select 77
palette operation 90
palette registers 77
palette, DAC 90
PEL panning 81
polarity, sync 38
POS information 10
preset row scan register 56
programming considerations 83
programming registers 86
protect registers 61
P4,P5 select 78
P7 - P0 signal 96

R

RAM loadable character generator 87
read map select register 72
read modes 73
read/writes, memory 35
refresh select 61
regen buffer address 59
registers 37
- attribute controller 76
- CRT controller 49
- DAC 90
- feature control 41
- general 38
- input status 40
- miscellaneous output 38
- programming 86
- reserved bits 37
- video enable 41
- write protect 61

reset bit 64
reset register 43
retrace polarity 38
ROM BIOS 4
rotate data 71
row scan counter 64

S
scanning, double 57
screen off bit 44
select address 74
select horizontal retrace 64
select operator 71
select P4,P5 78
select row scan 64
sequencer 5
  registers 42
sequencer registers 42
  address 42
  clocking mode 43
  map mask 45
  map select 46
  memory mode 48
  reserved bits 37
  reset 43
set/reset register 69
setting modes 8
setup mode 10
shift image 81
signal, auxiliary video 96
skew control, cursor 58
skew control, horizontal 52
source select 77
specifications, display 10
split screen, creating 88
start address high register 59
start address low register 59
start address register 59
start horizontal blanking register 51
start horizontal retrace pulse register 53
start vertical blanking register 63
support logic 4
support, display 10
sync polarity 39, 100
synchronization signals 100
synchronous reset 43
system interface, DAC to 91

T
timing, display 100
timing, video 98

U
underline location register 62
updates, full-screen 44

V
vertical display enable end register 61
vertical gain 39
vertical interrupt 60
vertical retrace end register 60
vertical retrace interrupt 61
vertical retrace start register 60
vertical size 100
vertical total register 54
VGA programming considerations 83
video connector 95, 99
video DAC programming considerations 92
video enable bit 10
video graphics array 1, 5
video modes 8
video registers 86
video subsystem enable register 41
VSYNC signal 96

W
word/byte mode 64
write modes 73
write operations 35

Numerics

16 background colors 78
512 character fonts 87

Keyboards (101- and 102-Key)

Description 1
Keyboard Layouts 1
101-Key Keyboard 2
102-Key Keyboard 3
Belgian Keyboard 4
Canadian French Keyboard 5
Danish Keyboard 6
Dutch Keyboard 7
French Keyboard 8
German Keyboard 9
Italian Keyboard 10
Latin American Keyboard 11
Norwegian Keyboard 12
Portuguese Keyboard 13
Spanish Keyboard 14
Swedish Keyboard 15
Swiss Keyboard 16
U.K. English Keyboard 17
U.S. English Keyboard 18
Sequential Key-Code Scanning 19
Buffer 19
Keys 19
Power-On Routine 20
Power-On Reset (POR) 20
Basic Assurance Test 20
Commands from the System 21
Commands to the System 27
Scan Codes 28
Set 1 Scan-Code Tables 28
Set 2 Scan-Code Tables 31
Set 3 Scan Code Tables 35
Clock and Data Signals 38
Data Stream 38
Data Output 39
Data Input 40
Encode and Usage 41
Extended Functions 44
Shift States 46
Special Handling 48

System Reset 48
Break 48
Pause 48
Print Screen 48
System Request 49
Other Characteristics 49
Cables and Connectors 50
Specifications 51

Index 52

Figures

1. Keyboard Commands from the System 21
2. Set All Keys Commands 23
3. Set Key Type Commands 24
4. Set/Reset Status Indicators 24
5. Typematic Rate 26
6. Keyboard Commands to the System 27
7. Keyboard Scan Codes, Set 1 29
8. Keyboard Scan Codes, Set 1 30
9. Keyboard Scan Codes, Set 1 31
10. Keyboard Scan Codes, Set 1 31
11. Keyboard Scan Codes, Set 1 31
12. Keyboard Scan Codes, Set 2 32
13. Keyboard Scan Codes, Set 2 33
14. Keyboard Scan Codes, Set 2 33
15. Keyboard Scan Codes, Set 2 34
16. Keyboard Scan Codes, Set 2 34
17. Keyboard Scan Codes, Set 3 35
18. Keyboard Data Stream Bit Definitions 39
19. Character Codes 42
20. Special Character Codes 44
21. Keyboard Extended Functions 45
22. Keyboard Connectors Signal and Voltage Assignments . . . 50

Notes:

Description

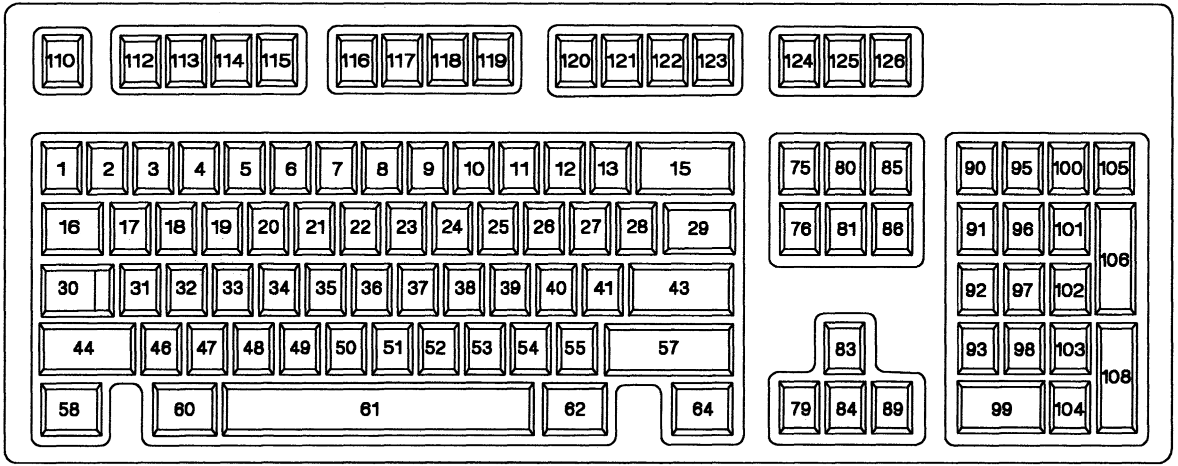
The keyboard has 102 keys (101 in the U.S.). At system power-on, the keyboard monitors the signals on the 'clock' and 'data' lines and establishes its line protocol. A bidirectional serial interface in the keyboard converts the 'clock' and 'data' signals and sends this information to and from the keyboard through the keyboard cable.

Keyboard Layouts

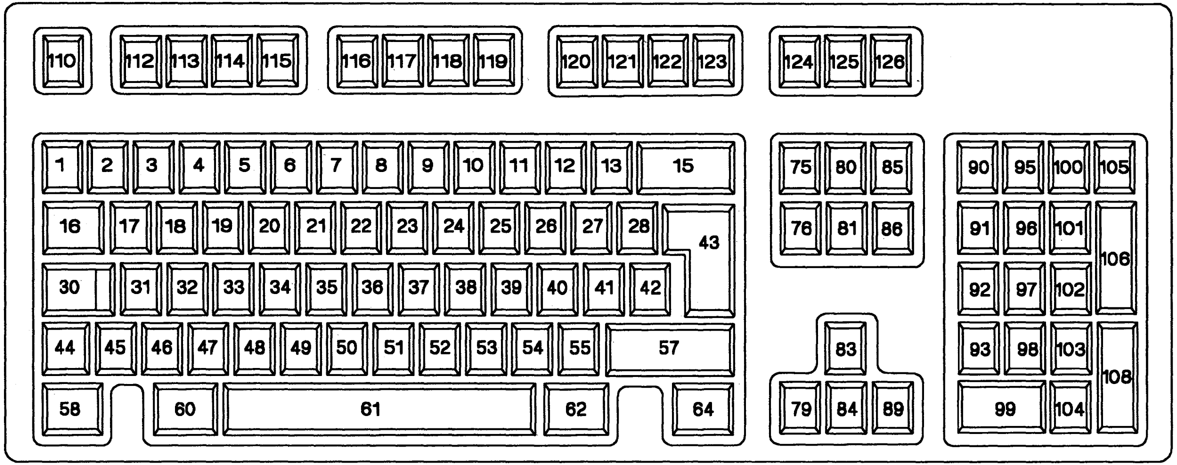
Keyboard layouts are in alphabetic order on the following pages. Nomenclature is on both the top and front face of the keybuttons.

- Belgian
- Canadian French
- Danish
- Dutch
- French
- German
- Italian
- Latin American
- Norwegian
- Portuguese
- Spanish
- Swedish
- Swiss
- U.K. English
- U.S. English.

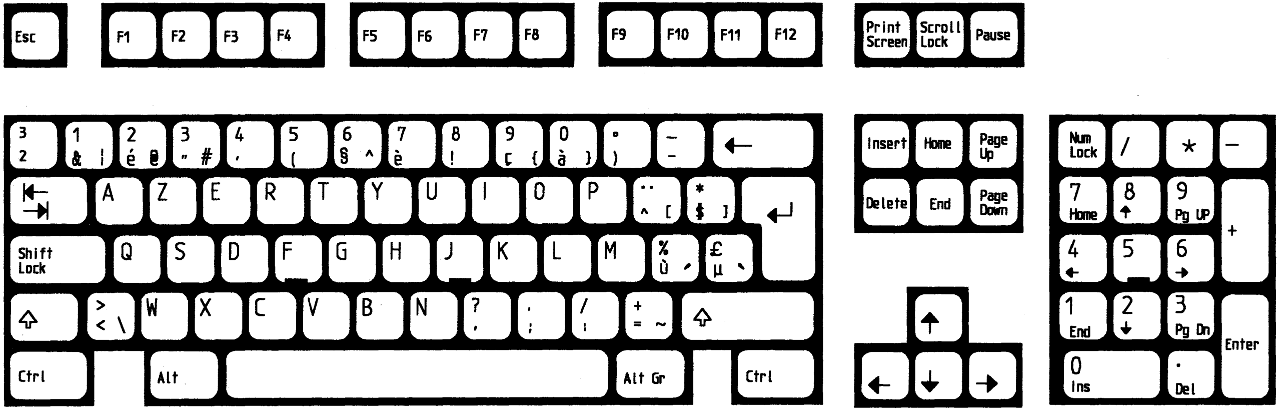
101-Key Keyboard



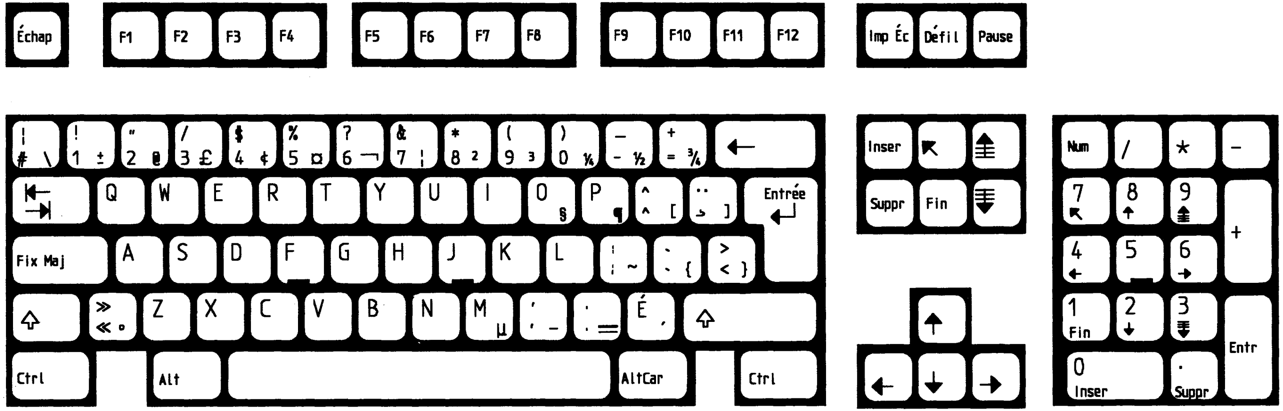
102-Key Keyboard



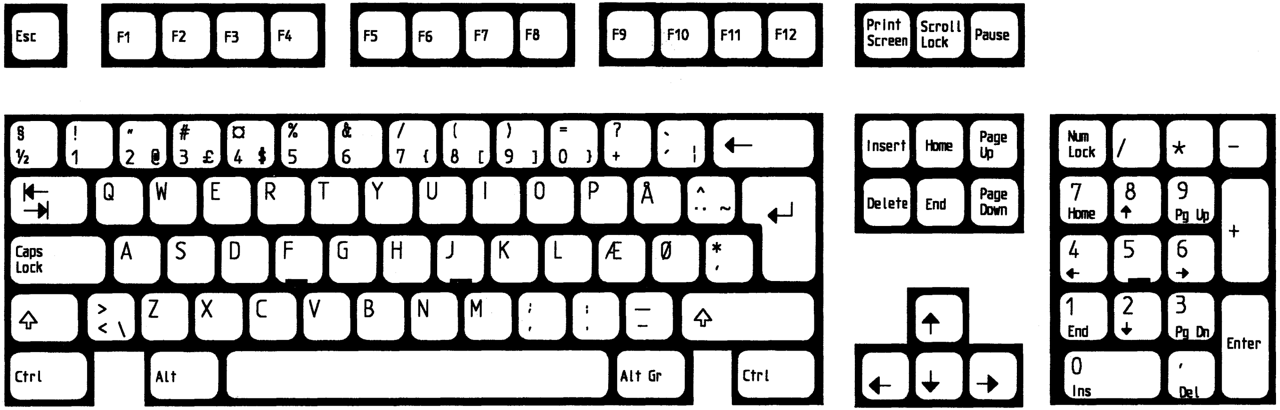
Belgian Keyboard



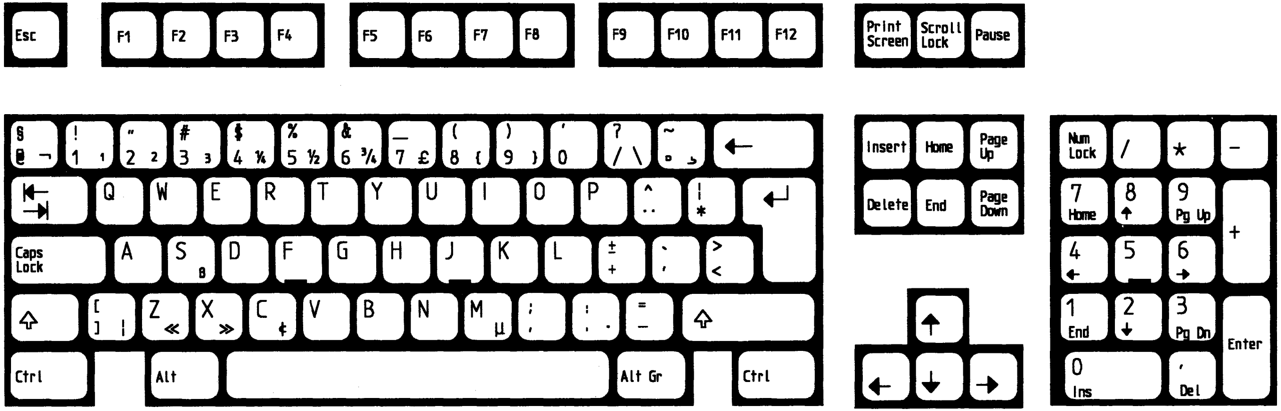
Canadian French Keyboard



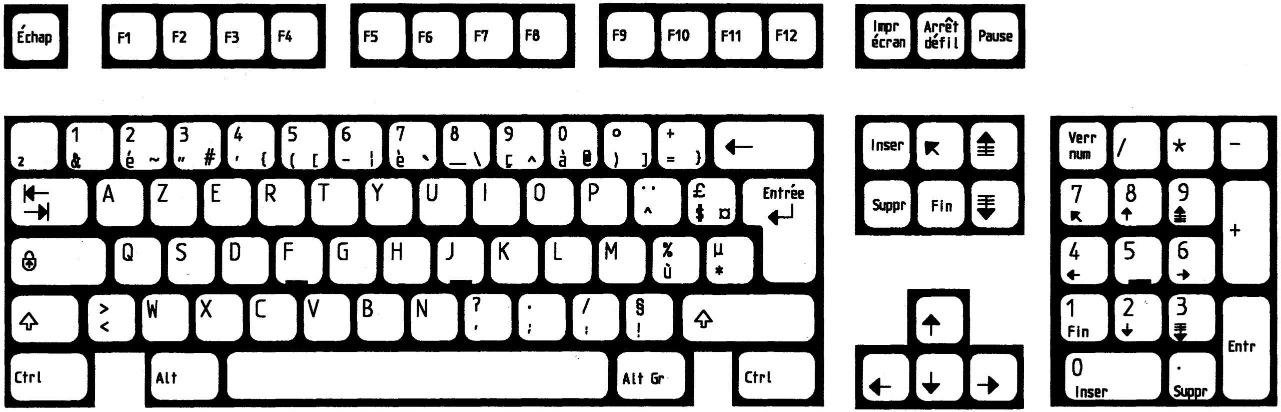
Danish Keyboard



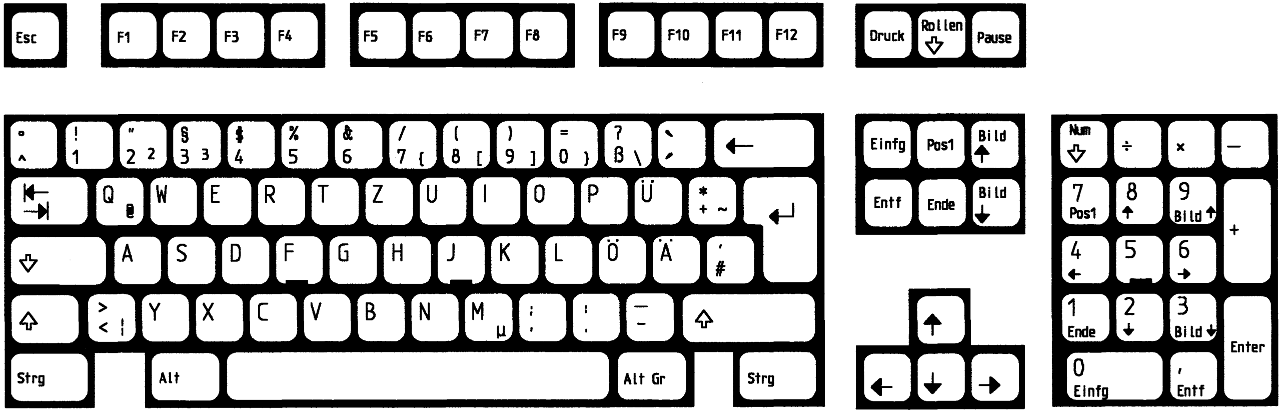
Dutch Keyboard



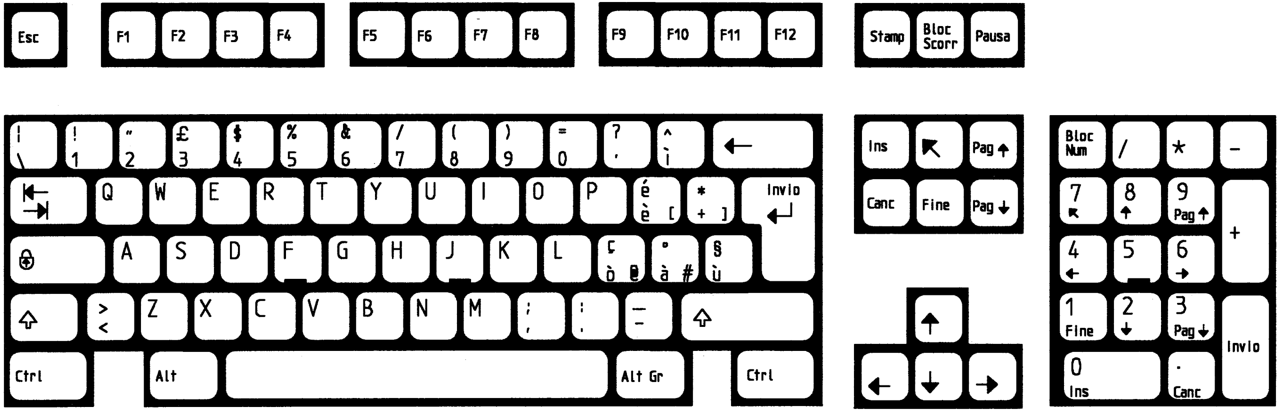
French Keyboard



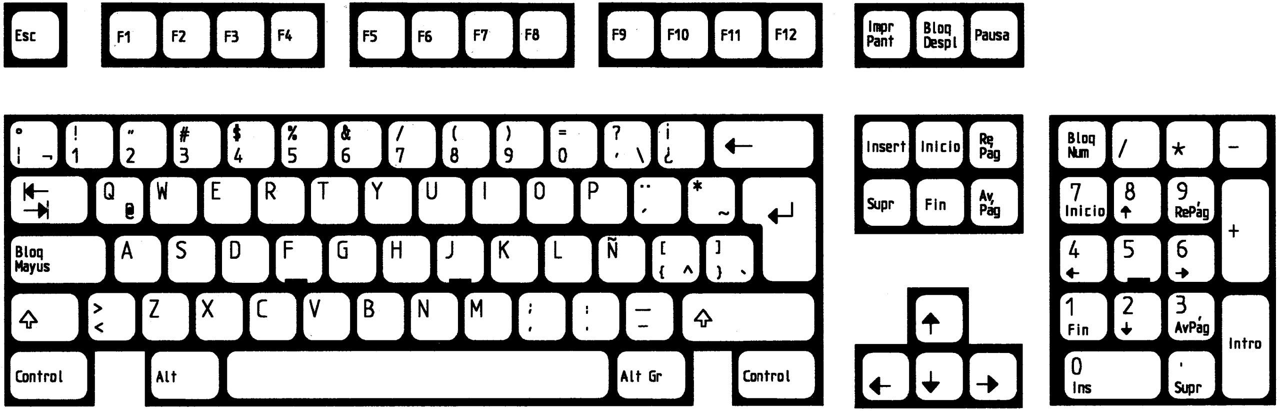
German Keyboard



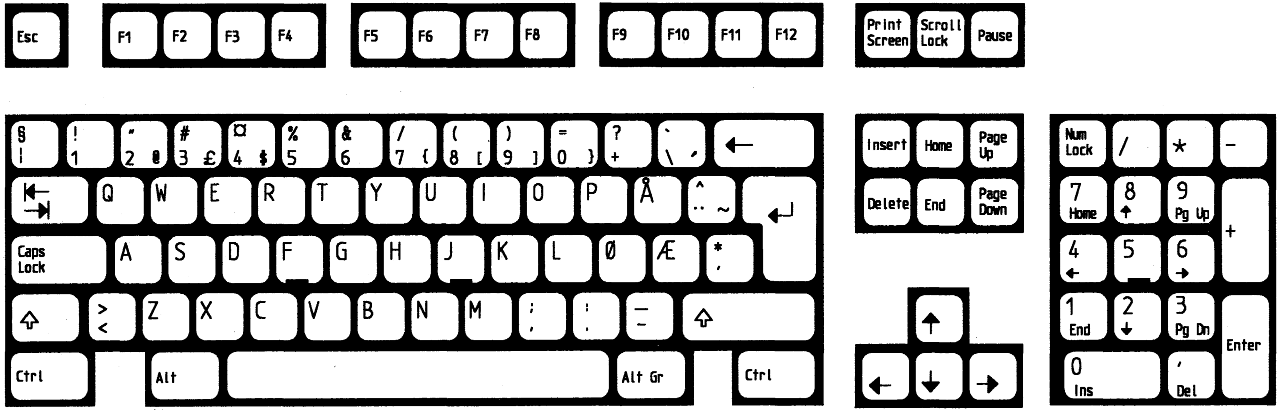
Italian Keyboard



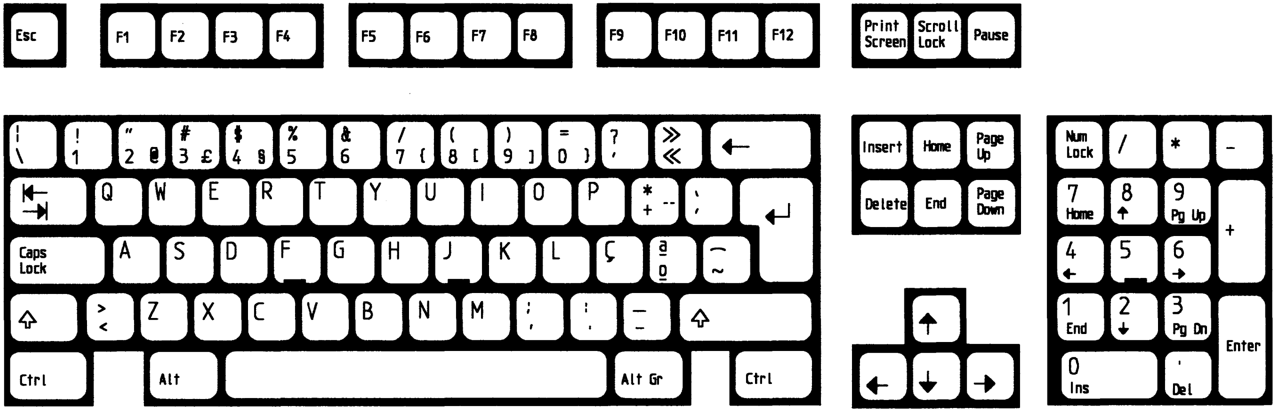
Latin American Keyboard



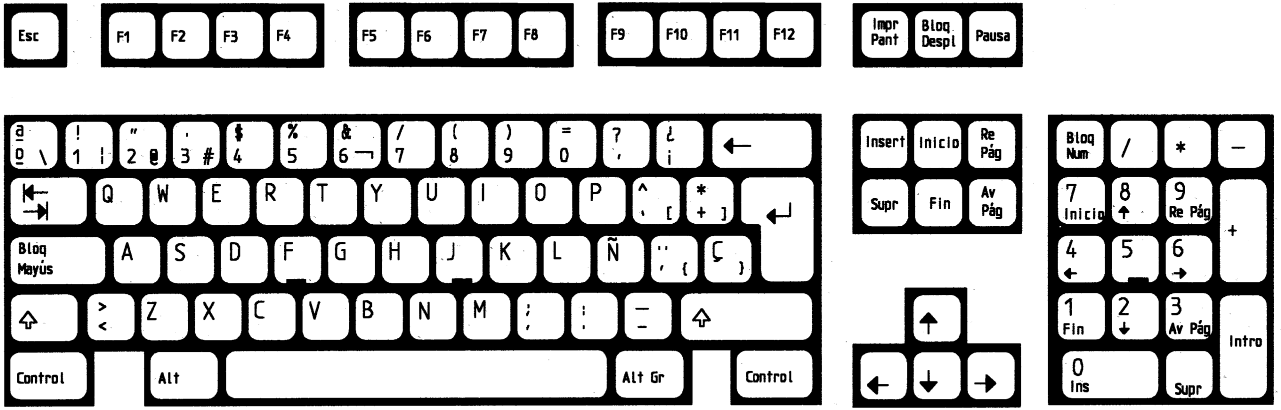
Norwegian Keyboard



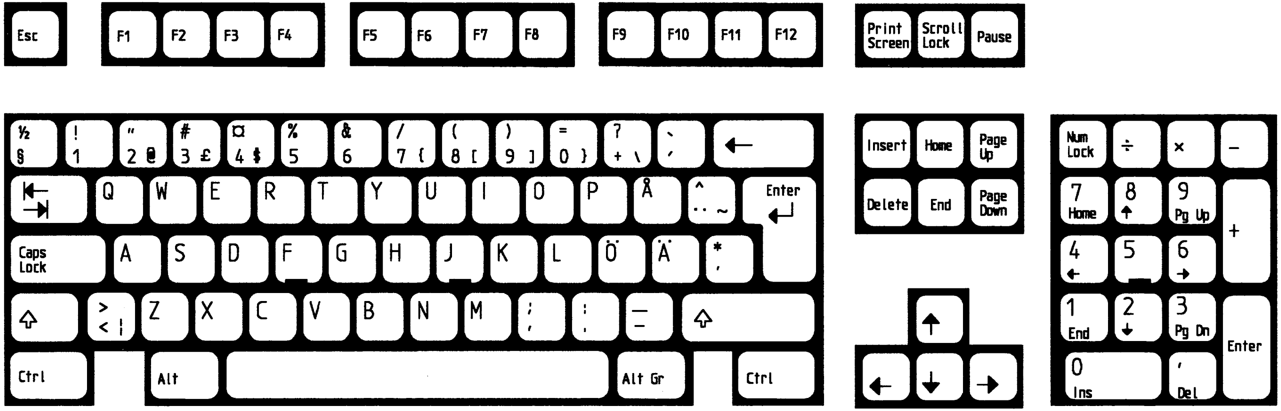
Portuguese Keyboard



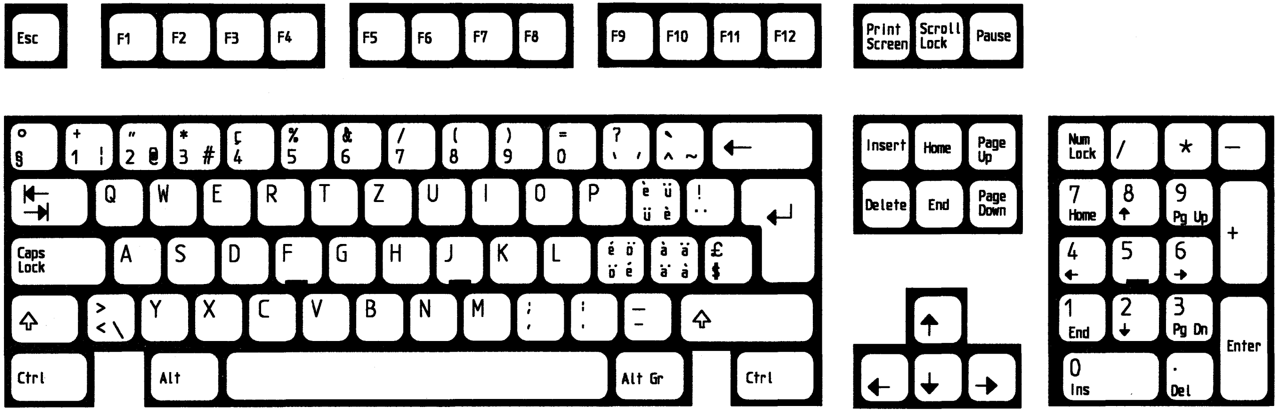
Spanish Keyboard



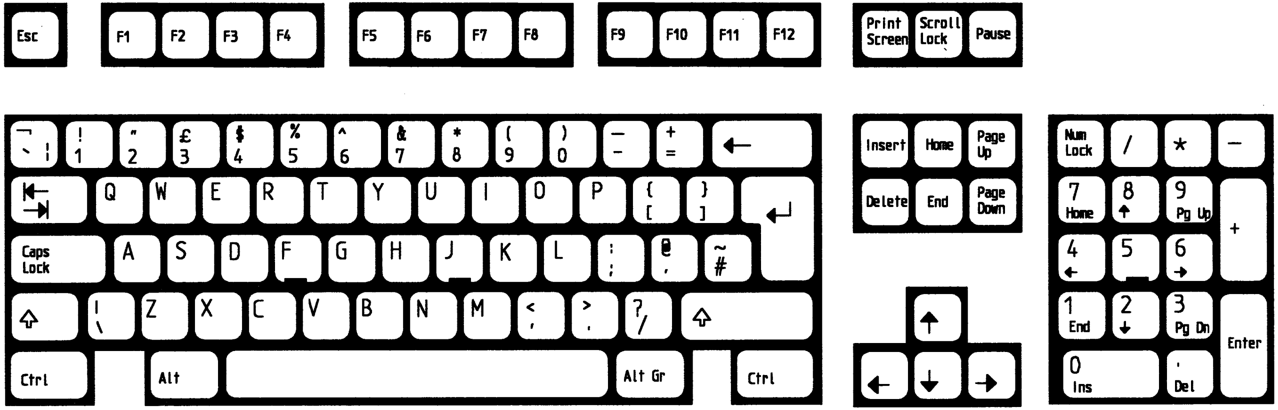
Swedish Keyboard



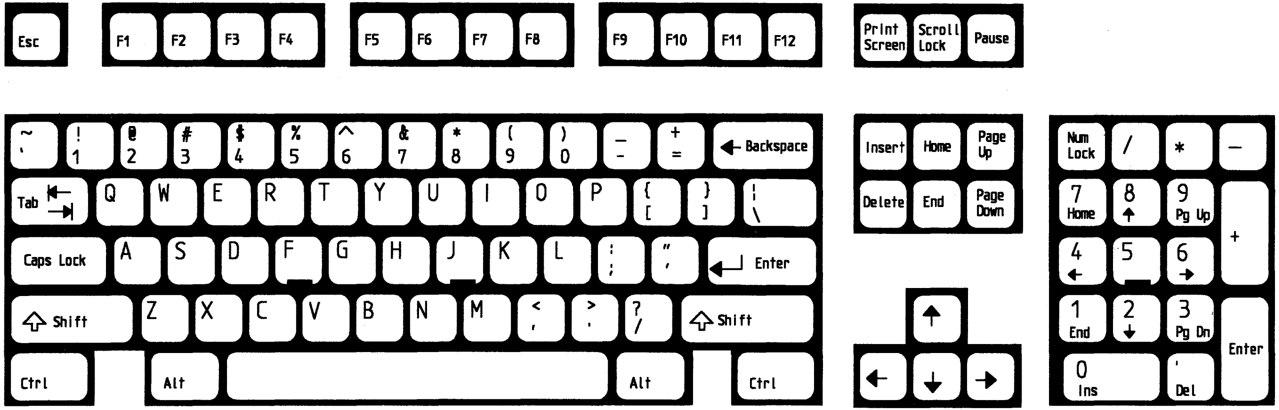
Swiss Keyboard



U.K. English Keyboard



U.S. English Keyboard



Sequential Key-Code Scanning

The keyboard detects all keys pressed and sends each scan code in the correct sequence. When not being serviced by the system, the keyboard stores the scan codes in its buffer.

Buffer

A 16-byte first-in-first-out (FIFO) buffer in the keyboard stores the scan codes until the system is ready to receive them. A buffer-overrun condition occurs when more than 16 bytes are placed in the keyboard buffer. An overrun code replaces the 17th byte. If more keys are pressed before the system allows keyboard output, the additional data is lost.

When the keyboard is allowed to send data, the bytes in the buffer are sent as in normal operation, and new data entered is detected and sent. Response codes do not occupy a buffer position.

If keystrokes generate a multiple-byte sequence, the entire sequence must fit into the available buffer space, or the keystroke is discarded and a buffer-overrun condition occurs.

Keys

Except for the Pause key, all keys are *make/break*. The make scan code of a key is sent to the keyboard controller when the key is pressed. When the key is released, its break scan code is sent.

Also, except for the Pause key, all keys are *typematic*. When a key is pressed and held down, the keyboard sends the make code for that key, delays 500 milliseconds ±20%, and begins sending a make code for that key at a rate of 10.9 characters per second ±20%. The typematic rate and delay can be modified (see "Set Typematic Rate/Delay (Hex F3)" on page 25).

If two or more keys are held down, only the last key pressed repeats at the typematic rate. Typematic operation stops when the last key pressed is released, even if other keys are still held down. If a key is pressed and held down while keyboard transmission is inhibited, only the first make code is stored in the buffer. This prevents buffer overflow because of typematic action.

Note: Scan-code set 3 allows key types to be changed by the system. See "Set 3 Scan Code Tables" on page 35 for the default settings.

Power-On Routine

The following activities take place when power is first applied to the keyboard:

Power-On Reset (POR)

The keyboard logic generates a 'power-on reset' signal (POR) when power is first applied to the keyboard. POR takes a minimum of 150 milliseconds and a maximum of 2.0 seconds from the time power is first applied to the keyboard.

Basic Assurance Test

The basic assurance test (BAT) consists of a keyboard processor test, a checksum of the read-only memory (ROM), and a random-access memory (RAM) test. During the BAT, activity on the 'clock' and 'data' lines is ignored. The LEDs are turned on at the beginning and off at the end of the BAT. The BAT takes a minimum of 300 milliseconds and a maximum of 500 milliseconds. This is in addition to the time required by the POR.

On satisfactory completion of the BAT, a completion code (hex AA) is sent to the system, and keyboard scanning begins. If a BAT failure occurs, the keyboard sends an error code to the system. The keyboard is then disabled pending command input. Completion codes are sent between 450 milliseconds and 2.5 seconds after POR, and between 300 and 500 milliseconds after a Reset command is acknowledged.

Immediately following POR, the keyboard monitors the signals on the keyboard 'clock' and 'data' lines and sets the line protocol.

Commands from the System

The following figure shows the commands that the system may send and their hexadecimal values.

Command	Hex Value
Set/Reset Status Indicators	ED
Echo	EE
Invalid Command	EF
Select Alternate Scan Codes	F0
Invalid Command	F1
Read ID	F2
Set Typematic Rate/Delay	F3
Enable	F4
Default Disable	F5
Set Default	F6
Set All Keys - Typematic	F7
- Make/Break	F8
- Make	F9
- Typematic/Make/Break	FA
Set Key Type - Typematic	FB
- Make/Break	FC
- Make	FD
Resend	FE
Reset	FF

Figure 1. Keyboard Commands from the System

These commands can be sent to the keyboard at any time. The keyboard responds within 20 milliseconds, except when performing the BAT, or executing a Reset command.

Note: Mode 1 accepts only the Reset command.

The following commands are in alphabetic order. They have different meanings when issued by the keyboard (see "Commands to the System" on page 27).

Default Disable (Hex F5): The Default Disable command resets all conditions to the power-on default state. The keyboard responds with ACK, clears its output buffer, sets the default key types (scan-code set 3 operation only) and typematic rate/delay, and clears the last typematic key. The keyboard stops scanning and awaits further instructions.

Echo (Hex EE): Echo is a diagnostic aid. When the keyboard receives this command, it issues a hex EE response if previously enabled, it continues scanning.

Enable (Hex F4): On receipt of this command, the keyboard responds with ACK, clears its output buffer, clears the last typematic key, and starts scanning.

Invalid Command (Hex EF and F1): Hex EF and hex F1 are invalid commands and are not supported. If one of these is sent, the keyboard does not acknowledge the command but returns a Resend command and continues in its prior scanning state. No other activities occur.

Read ID (Hex F2): This command requests identification information from the keyboard. The keyboard responds with ACK, stops scanning, and sends the two keyboard ID bytes. The second byte must follow completion of the first by no more than 500 microseconds. After the output of the second ID byte, the keyboard resumes scanning.

Resend (Hex FE): The system sends this command when it detects an error in any transmission from the keyboard. It is sent only after a keyboard transmission and before the system allows the next keyboard output. When a Resend command is received, the keyboard sends the previous output again (unless the previous output was Resend command, in which case the keyboard sends the last byte before the Resend command).

Reset (Hex FF): The system issues a Reset command to start a program reset and a keyboard internal self-test. The keyboard acknowledges the command with an ACK and ensures the system accepts ACK before executing the command. The system signals acceptance of ACK by raising the 'clock' and 'data' lines for a minimum of 500 microseconds. The keyboard is disabled from the time it receives the Reset command until ACK is accepted, or until another command is sent that overrides the previous command.

Following acceptance of ACK, the keyboard is reinitialized and performs the BAT. After returning the completion code, the keyboard defaults to scan-code set 2.

Select Alternate Scan Codes (Hex F0): This command instructs the keyboard to select one of three sets of scan codes. The keyboard acknowledges receipt of this command with ACK and clears both the output buffer and the typematic key (if one is active). The system then sends the option byte and the keyboard responds with another ACK. An option byte value of hex 01 selects scan code set 1, hex 02 selects scan code set 2, and hex 03 selects scan code set 3.

An option byte value of hex 00 causes the keyboard to acknowledge with an ACK and send a byte telling the system which scan code set is currently in use. To prevent the controller from translating this byte, disable the keyboard controller translate mode.

After establishing the new scan code set, the keyboard returns to the scanning state it was in before receiving the Select Alternate Scan Codes command.

Set All Keys (Hex F7, F8, F9, FA)

These commands instruct the keyboard to set all keys to a condition listed in the following figure.

Hex Value	Command
F7	Set All Keys - Typematic
F8	Set All Keys - Make/Break
F9	Set All Keys - Make
FA	Set All Keys - Typematic/Make/Break

Figure 2. Set All Keys Commands

The keyboard responds with ACK, clears its output buffer, sets all keys to the condition indicated by the command, and continues scanning (if it was previously enabled). Although these commands can be sent using any scan-code set, they affect only the operation of scan-code set 3.

Set Default (Hex F6): The Set Default command resets all conditions to the power-on default state. The keyboard responds with ACK, clears its output buffer, sets the default key types (scan-code set 3 operation only) and typematic rate/delay, clears the last typematic key, and continues scanning.

Set Key Type (Hex FB, FC, FD): These commands instruct the keyboard to set individual keys to a condition listed in the following figure.

Hex Value	Command
FB	Set Key Type - Typematic
FC	Set Key Type - Make/Break
FD	Set Key Type - Make

Figure 3. Set Key Type Commands

The keyboard responds with ACK, clears its output buffer, and prepares to receive key identification. The system identifies each key by its scan-code value, as defined in scan-code set 3. Only scan code set 3 values are valid for key identification. The type of each identified key is set to the value indicated by the command.

These commands can be sent using any scan code set, but affect only the operation of scan code set 3.

Set/Reset Status Indicators (Hex ED): Three status indicators on the keyboard—Num Lock, Caps Lock, and Scroll Lock—are accessible by the system. The keyboard activates or deactivates these indicators when it receives a valid command-code sequence from the system. The command sequence begins with the command byte (hex ED). The keyboard responds with ACK, stops scanning, and waits for the option byte from the system. The bit assignments for this option byte are as follows.

Bit	Function
7 - 3	Reserved (must be 0's)
2	Caps Lock Indicator
1	Num Lock Indicator
0	Scroll Lock Indicator

Figure 4. Set/Reset Status Indicators

If a bit for an indicator is set to 1, the indicator is turned on. If a bit is set to 0, the indicator is turned off.

The keyboard responds to the option byte with ACK, sets the indicators and, if the keyboard was previously enabled, continues scanning. The state of the indicators reflect the bits in the option byte and can be activated or deactivated in any combination. If another command is received in place of the option byte, execution of the Set/Reset Mode Indicators command is stopped, with no change to the indicator states, and the new command is processed.

Immediately after power-on, the lights default to the Off state. If the Set Default and Default Disable commands are received, the lamps remain in the state they were in before the command was received.

Set Typematic Rate/Delay (Hex F3): The system issues the Set Typematic Rate/Delay command to change the typematic rate and delay. The keyboard responds to the command with ACK, stops scanning, and waits for the system to issue the rate/delay value byte. The keyboard responds to the rate/delay value byte with another ACK, sets the rate and delay to the values indicated, and continues scanning (if it was previously enabled). Bits 6 and 5 indicate the delay, and bits 4, 3, 2, 1, and 0 (the least-significant bit) the rate. Bit 7, the most-significant bit, is always 0. The delay is equal to 1 plus the binary value of bits 6 and 5, multiplied by 250 milliseconds ±20%.

The period (interval from one typematic output to the next) is determined by the following equation:

Period = $(8 + A) \times (2^B) \times 0.00417$ seconds ±20%.
where:
A = binary value of bits 2, 1, and 0.
B = binary value of bits 4 and 3.

The typematic rate (make codes per second) is 1 for each period.

Bit	Typematic Rate ± 20%	Bit	Typematic Rate ± 20%
00000	30.0	10000	7.5
00001	26.7	10001	6.7
00010	24.0	10010	6.0
00011	21.8	10011	5.5
00100	20.0	10100	5.0
00101	18.5	10101	4.6
00110	17.1	10110	4.3
00111	16.0	10111	4.0
01000	15.0	11000	3.7
01001	13.3	11001	3.3
01010	12.0	11010	3.0
01011	10.9	11011	2.7
01100	10.0	11100	2.5
01101	9.2	11101	2.3
01110	8.6	11110	2.1
01111	8.0	11111	2.0

Figure 5. Typematic Rate

The default values for the system keyboard are as follows:

Typematic rate = 10.9 characters per second ± 20%.

Delay = 500 milliseconds ± 20%.

The execution of this command stops without change to the existing rate if another command is received instead of the rate/delay value byte.

Commands to the System

The following figure shows the commands that the keyboard may send to the system, and their hexadecimal values.

Command	Hex Value
Key Detection Error/Overrun	00 (Code Sets 2 and 3)
Keyboard ID	83AB
BAT Completion Code	AA
BAT Failure Code	FC
Echo	EE
Acknowledge (ACK)	FA
Resend	FE
Key Detection Error/Overrun	FF (Code Set 1)

Figure 6. Keyboard Commands to the System

The commands the keyboard sends to the system are described in alphabetic order. They have different meanings when issued by the system.

Acknowledge (Hex FA): The keyboard issues ACK to any valid input other than an Echo, or Resend command. If the keyboard is interrupted while sending ACK, it discards ACK and accepts and responds to the new command.

BAT Completion Code (Hex AA): Following satisfactory completion of the BAT, the keyboard sends hex AA. Any other code indicates a failure of the keyboard.

BAT Failure Code (Hex FC): If a BAT failure occurs, the keyboard sends this code, stops scanning, and waits for a system response or reset.

Echo (Hex EE): The keyboard sends this code in response to an Echo command.

Keyboard ID (Hex 83AB): The Keyboard ID consists of 2 bytes, hex 83AB. The keyboard responds to the Read ID command with ACK, stops scanning, and sends the 2 ID bytes. The low byte is sent first followed by the high byte. Following the output of the Keyboard ID, the keyboard begins scanning. Because of keyboard controller translation, the keyboard ID may not be returned to the system as hex 83AB.

Key Detection Error (Hex 00 or FF): The keyboard sends a key detection error character if conditions in the keyboard make it impossible to identify a switch closure. If the keyboard is using scan-code set 1, the code is hex FF. For sets 2 and 3, the code is hex 00.

Overrun (Hex 00 or FF): An overrun character is placed in the keyboard buffer and replaces the last code when the buffer capacity has been exceeded. The code is sent to the system when it reaches the top of the buffer queue. If the keyboard is using scan code set 1, the code is hex FF. For sets 2 and 3, the code is hex 00.

Resend (Hex FE): The keyboard issues a Resend command following receipt of an invalid input, or any input with incorrect parity. If the system sends nothing to the keyboard, no response is required.

Scan Codes

The following figures list the key numbers of the three scan code sets and their hexadecimal values. The system defaults to scan set 2, but can be switched to set 1 or set 3 (see "Select Alternate Scan Codes (Hex F0)" on page 23).

Set 1 Scan-Code Tables

In scan-code set 1, each key is assigned a base scan code and, sometimes extra codes to generate artificial shift states in the system. The typematic scan codes are identical to the base scan code for each key.

The following figure shows the codes sent for the keys, regardless of any shift states in the keyboard or system. Refer to "Keyboard Layouts" beginning on page 1 to determine the character associated with each key number.

Key Number	Make Code	Break Code	Key Number	Make Code	Break Code
1	29	A9	47	2D	AD
2	02	82	48	2E	AE
3	03	83	49	2F	AF
4	04	84	50	30	B0
5	05	85	51	31	B1
6	06	86	52	32	B2
7	07	87	53	33	B3
8	08	88	54	34	B4
9	09	89	55	35	B5
10	0A	8A	57	36	B6
11	0B	8B	58	1D	9D
12	0C	8C	60	38	B8
13	0D	8D	61	39	B9
15	0E	8E	62	E0 38	E0 B8
16	0F	8F	64	E0 1D	E0 9D
17	10	90	90	45	C5
18	11	91	91	47	C7
19	12	92	92	4B	CB
20	13	93	93	4F	CF
21	14	94	96	48	C8
22	15	95	97	4C	CC
23	16	96	98	50	D0
24	17	97	99	52	D2
25	18	98	100	37	B7
26	19	99	101	49	C9
27	1A	9A	102	4D	CD
28	1B	9B	103	51	D1
29 *	2B	AB	104	53	D3
30	3A	BA	105	4A	CA
31	1E	9E	106	4E	CE
32	1F	9F	108	E0 1C	E0 9C
33	20	A0	110	01	81
34	21	A1	112	3B	BB
35	22	A2	113	3C	BC
36	23	A3	114	3D	BD
37	24	A4	115	3E	BE
38	25	A5	116	3F	BF
39	26	A6	117	40	C0
40	27	A7	118	41	C1
41	28	A8	119	42	C2
42 **	2B	AB	120	43	C3
43	1C	9C	121	44	C4
44	2A	AA	122	57	D7
45 **	56	D6	123	58	D8
46	2C	AC	125	46	C6

* 101-key keyboard only, ** 102-key keyboard only.

Figure 7. Keyboard Scan Codes, Set 1

The remaining keys send a series of codes that are dependent on the state of the various shift keys (Ctrl, Alt, and Shift), and the state of Num Lock (On or Off). Because the base scan code is identical to another key, an extra code (hex E0) has been added to the base code to make it unique.

	Base Case, or		
Key	**Shift + Num Lock**	**Shift Case**	**Num Lock on**
No.	**Make/Break**	**Make/Break ***	**Make/Break**
75	E0 52	E0 AA E0 52	E0 2A E0 52
	/E0 D2	/E0 D2 E0 2A	/E0 D2 E0 AA
76	E0 53	E0 AA E0 53	E0 2A E0 53
	/E0 D3	/E0 D3 E0 2A	/E0 D3 E0 AA
79	E0 4B	E0 AA E0 4B	E0 2A E0 4B
	/E0 CB	/E0 CB E0 2A	/E0 CB E0 AA
80	E0 47	E0 AA E0 47	E0 2A E0 47
	/E0 C7	/E0 C7 E0 2A	/E0 C7 E0 AA
81	E0 4F	E0 AA E0 4F	E0 2A E0 4F
	/E0 CF	/E0 CF E0 2A	/E0 CF E0 AA
83	E0 48	E0 AA E0 48	E0 2A E0 48
	/E0 C8	/E0 C8 E0 2A	/E0 C8 E0 AA
84	E0 50	E0 AA E0 50	E0 2A E0 50
	/E0 D0	/E0 D0 E0 2A	/E0 D0 E0 AA
85	E0 49	E0 AA E0 49	E0 2A E0 49
	/E0 C9	/E0 C9 E0 2A	/E0 C9 E0 AA
86	E0 51	E0 AA E0 51	E0 2A E0 51
	/E0 D1	/E0 D1 E0 2A	/E0 D1 E0 AA
89	E0 4D	E0 AA E0 4D	E0 2A E0 4D
	/E0 CD	/E0 CD E0 2A	/E0 CD E0 AA

*** If the left Shift key is held down, the AA/2A shift make and break are sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.**

Figure 8. Keyboard Scan Codes, Set 1

Key No.	Scan Code Make/Break	Shift Case Make/Break *
95	E0 35/E0 B5	E0 AA E0 35/E0 B5 E0 2A

* If the left Shift key is held down, the AA/2A shift make and break are sent with the other scan codes. If the right Shift key is held down, B6/36 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Figure 9. Keyboard Scan Codes, Set 1

Key No.	Scan Code Make/Break	Ctrl Case, Shift Case Make/Break	Alt Case Make/Break
124	E0 2A E0 37 /E0 B7 E0 AA	E0 37/E0 B7	54/D4

Figure 10. Keyboard Scan Codes, Set 1

Key No.	Make Code	Ctrl Key Pressed
126 *	E1 1D 45 E1 9D C5	E0 46 E0 C6

* This key is not typematic. All associated scan codes occur on the make of the key.

Figure 11. Keyboard Scan Codes, Set 1

Set 2 Scan-Code Tables

In scan-code set 2, each key is assigned a unique 8-bit make scan code, which is sent when the key is pressed. Each key also sends a break code when the key is released. The break code consists of 2 bytes, the first of which is the break code prefix, hex F0; the second byte is the same as the make scan code for that key. The typematic scan code for a key is the same as the key's make code.

The following figure shows the codes sent for the keys, regardless of any shift states in the keyboard or system. Refer to “Keyboard Layouts” beginning on page 1 to determine the character associated with each key number.

Key Number	Make Code	Break Code	Key Number	Make Code	Break Code
1	0E	F0 0E	47	22	F0 22
2	16	F0 16	48	21	F0 21
3	1E	F0 1E	49	2A	F0 2A
4	26	F0 26	50	32	F0 32
5	25	F0 25	51	31	F0 31
6	2E	F0 2E	52	3A	F0 3A
7	36	F0 36	53	41	F0 41
8	3D	F0 3D	54	49	F0 49
9	3E	F0 3E	55	4A	F0 4A
10	46	F0 46	57	59	F0 59
11	45	F0 45	58	14	F0 14
12	4E	F0 4E	60	11	F0 11
13	55	F0 55	61	29	F0 29
15	66	F0 66	62	E0 11	E0 F0 11
16	0D	F0 0D	64	E0 14	E0 F0 14
17	15	F0 15	90	77	F0 77
18	1D	F0 1D	91	6C	F0 6C
19	24	F0 24	92	6B	F0 6B
20	2D	F0 2D	93	69	F0 69
21	2C	F0 2C	96	75	F0 75
22	35	F0 35	97	73	F0 73
23	3C	F0 3C	98	72	F0 72
24	43	F0 43	99	70	F0 70
25	44	F0 44	100	7C	F0 7C
26	4D	F0 4D	101	7D	F0 7D
27	54	F0 54	102	74	F0 74
28	5B	F0 5B	103	7A	F0 7A
29 *	5D	F0 5D	104	71	F0 71
30	58	F0 58	105	7B	F0 7B
31	1C	F0 1C	106	79	F0 79
32	1B	F0 1B	108	E0 5A	E0 F0 5A
33	23	F0 23	110	76	F0 76
34	2B	F0 2B	112	05	F0 05
35	34	F0 34	113	06	F0 06
36	33	F0 33	114	04	F0 04
37	3B	F0 3B	115	0C	F0 0C
38	42	F0 42	116	03	F0 03
39	4B	F0 4B	117	0B	F0 0B
40	4C	F0 4C	118	83	F0 83
41	52	F0 52	119	0A	F0 0A
42 **	5D	F0 5D	120	01	F0 01
43	5A	F0 5A	121	09	F0 09
44	12	F0 12	122	78	F0 78
45 **	61	F0 61	123	07	F0 07
46	1A	F0 1A	125	7E	F0 7E

* 101-key keyboard only, ** 102-key keyboard only.

Figure 12. Keyboard Scan Codes, Set 2

The remaining keys send a series of codes that are dependent on the state of the shift keys (Ctrl, Alt, and Shift), and the state of Num Lock (On or Off). Because the base scan code is identical to another key, an extra code (hex E0) is added to the base code to make it unique.

Key No.	Base Case, or Shift + Num Lock Make/Break	Shift Case Make/Break *	Num Lock on Make/Break
75	E0 70	E0 F0 12 E0 70	E0 12 E0 70
	/E0 F0 70	/E0 F0 70 E0 12	/E0 F0 70 E0 F0 12
76	E0 71	E0 F0 12 E0 71	E0 12 E0 71
	/E0 F0 71	/E0 F0 71 E0 12	/E0 F0 71 E0 F0 12
79	E0 6B	E0 F0 12 E0 6B	E0 12 E0 6B
	/E0 F0 6B	/E0 F0 6B E0 12	/E0 F0 6B E0 F0 12
80	E0 6C	E0 F0 12 E0 6C	E0 12 E0 6C
	/E0 F0 6C	/E0 F0 6C E0 12	/E0 F0 6C E0 F0 12
81	E0 69	E0 F0 12 E0 69	E0 12 E0 69
	/E0 F0 69	/E0 F0 69 E0 12	/EO F0 69 E0 F0 12
83	E0 75	E0 F0 12 E0 75	E0 12 E0 75
	/E0 F0 75	/E0 F0 75 E0 12	/E0 F0 75 E0 F0 12
84	E0 72	E0 F0 12 E0 72	E0 12 E0 72
	/E0 F0 72	/E0 F0 72 E0 12	/E0 F0 72 E0 F0 12
85	E0 7D	E0 FO 12 E0 7D	E0 12 E0 7D
	/E0 F0 7D	/E0 F0 7D E0 12	/E0 F0 7D E0 F0 12
86	E0 7A	E0 F0 12 E0 7A	E0 12 E0 7A
	/E0 F0 7A	/E0 F0 7A E0 12	/E0 F0 7A E0 F0 12
89	E0 74	E0 F0 12 E0 74	E0 12 E0 74
	/E0 F0 74	/E0 F0 74 E0 12	/E0 F0 74 E0 F0 12

* If the left Shift key is held down, the F0 12/12 shift make and break are sent with the other scan codes. If the right Shift key is held down, F0/59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Figure 13. Keyboard Scan Codes, Set 2

Key No.	Scan Code Make/Break	Shift Case Make/Break *
95	E0 4A/E0 F0 4A	E0 F0 12 E0 4A/E0 F0 4A E0 12

* If the left Shift key is held down, the F0 12/12 shift make and break are sent with the other scan codes. If the right Shift key is held down, F0 59/59 is sent. If both Shift keys are down, both sets of codes are sent with the other scan code.

Figure 14. Keyboard Scan Codes, Set 2

Key No.	Scan Code Make/Break	Ctrl Case, Shift Case Make/Break	Alt Case Make/Break
124	E0 12 E0 7C /E0 F0 7C E0 F0 12	E0 7C/E0 F0 7C	84/F0 84

Figure 15. Keyboard Scan Codes, Set 2

Key No.	Make Code	Ctrl Key Pressed
126 *	E1 14 77 E1 F0 14 F0 77	E0 7E E0 F0 7E

* This key is not typematic. All associated scan codes occur on the make of the key.

Figure 16. Keyboard Scan Codes, Set 2

Set 3 Scan Code Tables

In scan-code set 3, each key is assigned a unique 8-bit make scan code, which is sent when the key is pressed. Each key also sends a break code when the key is released. The break code consists of 2 bytes, the first of which is the break-code prefix, hex F0; the second byte is the same as the make scan code for that key. The typematic scan code for a key is the same as the key's make code. With this scan-code set, each key sends only one scan code, and no keys are affected by the state of any other keys.

The following figure shows the codes sent for the keys, regardless of any shift states in the keyboard or system. Refer to "Keyboard Layouts" beginning on page 1 to determine the character associated with each key number.

Key Number	Make Code	Break Code	Default Key State
1	0E	F0 0E	Typematic
2	16	F0 16	Typematic
3	1E	F0 1E	Typematic
4	26	F0 26	Typematic
5	25	F0 25	Typematic
6	2E	F0 2E	Typematic
7	36	F0 36	Typematic
8	3D	F0 3D	Typematic
9	3E	F0 3E	Typematic
10	46	F0 46	Typematic
11	45	F0 45	Typematic
12	4E	F0 4E	Typematic
13	55	F0 55	Typematic
15	66	F0 66	Typematic
16	0D	F0 0D	Typematic
17	15	F0 15	Typematic
18	1D	F0 1D	Typematic
19	24	F0 24	Typematic
20	2D	F0 2D	Typematic
21	2C	F0 2C	Typematic
22	35	F0 35	Typematic
23	3C	F0 3C	Typematic
24	43	F0 43	Typematic
25	44	F0 44	Typematic
26	4D	F0 4D	Typematic
27	54	F0 54	Typematic
28	5B	F0 5B	Typematic

Figure 17 (Part 1 of 3). Keyboard Scan Codes, Set 3

Key Number	Make Code	Break Code	Default Key State
29 *	5C	F0 5C	Typematic
30	14	F0 14	Make/Break
31	1C	F0 1C	Typematic
32	1B	F0 1B	Typematic
33	23	F0 23	Typematic
34	2B	F0 2B	Typematic
35	34	F0 34	Typematic
36	33	F0 33	Typematic
37	3B	F0 3B	Typematic
38	42	F0 42	Typematic
39	4B	F0 4B	Typematic
40	4C	F0 4C	Typematic
41	52	F0 52	Typematic
42 **	53	F0 53	Typematic
43	5A	F0 5A	Typematic
44	12	F0 12	Make/Break
45 **	13	F0 13	Typematic
46	1A	F0 1A	Typematic
47	22	F0 22	Typematic
48	21	F0 21	Typematic
49	2A	F0 2A	Typematic
50	32	F0 32	Typematic
51	31	F0 31	Typematic
52	3A	F0 3A	Typematic
53	41	F0 41	Typematic
54	49	F0 49	Typematic
55	4A	F0 4A	Typematic
57	59	F0 59	Make/Break
58	11	F0 11	Make/Break
60	19	F0 19	Make/Break
61	29	F0 29	Typematic
62	39	F0 39	Make only
64	58	F0 58	Make only
75	67	F0 67	Make only
76	64	F0 64	Typematic
79	61	F0 61	Typematic
80	6E	F0 6E	Make only
81	65	F0 65	Make only
83	63	F0 63	Typematic
84	60	F0 60	Typematic
85	6F	F0 6F	Make only
86	6D	F0 6D	Make only
89	6A	F0 6A	Typematic
90	76	F0 76	Make only
91	6C	F0 6C	Make only
92	6B	F0 6B	Make only

Figure 17 (Part 2 of 3). Keyboard Scan Codes, Set 3

Key Number	Make Code	Break Code	Default Key State
93	69	F0 69	Make only
95	77	F0 77	Make only
96	75	F0 75	Make only
97	73	F0 73	Make only
98	72	F0 72	Make only
99	70	F0 70	Make only
100	7E	F0 7E	Make only
101	7D	F0 7D	Make only
102	74	F0 74	Make only
103	7A	F0 7A	Make only
104	71	F0 71	Make only
105	84	F0 84	Make only
106	7C	F0 7C	Typematic
108	79	F0 79	Make only
110	08	F0 08	Make only
112	07	F0 07	Make only
113	0F	F0 0F	Make only
114	17	F0 17	Make only
115	1F	F0 1F	Make only
116	27	F0 27	Make only
117	2F	F0 2F	Make only
118	37	F0 37	Make only
119	3F	F0 3F	Make only
120	47	F0 47	Make only
121	4F	F0 4F	Make only
122	56	F0 56	Make only
123	5E	F0 5E	Make only
124	57	F0 57	Make only
125	5F	F0 5F	Make only
126	62	F0 62	Make only

* 101-key keyboard only.
** 102-key keyboard only.

Figure 17 (Part 3 of 3). Keyboard Scan Codes, Set 3

Clock and Data Signals

The keyboard and system communicate over the 'clock' and 'data' lines. The source of each of these lines is an open-collector device on the keyboard that allows either the keyboard or system to force a line to an inactive (low) level. When no communication is occurring, the 'clock' line is at an active (high) level. The state of the 'data' line is held active (high) by the keyboard.

When the system sends data to the keyboard, it forces the 'data' line to an inactive level and allows the 'clock' line to go to an active level.

An inactive signal will have a value of at least 0, but not more than +0.7 volts. A signal at the inactive level is a logical 0. An active signal will have a value of at least +2.4, but not more than +5.5 volts. A signal at the active level is a logical 1. Voltages are measured between a signal source and the dc network ground.

When the keyboard sends data to, or receives data from the system, it generates the 'clock' signal to time the data. The system can prevent the keyboard from sending data by forcing the 'clock' line to an inactive level; the 'data' line may be active or inactive during this time.

During the BAT, the keyboard allows the 'clock' and 'data' lines to go to an active level.

Data Stream

Data transmissions to and from the keyboard consist of an 11-bit data stream (Mode 2) sent serially over the 'data' line. The following figure shows the functions of the bits.

Bit	Function
11	Stop bit (always 1)
10	Parity bit (odd parity)
9	Data bit 7 (most-significant)
8	Data bit 6
7	Data bit 5
6	Data bit 4
5	Data bit 3
4	Data bit 2
3	Data bit 1
2	Data bit 0 (least-significant)
1	Start bit (always 0)

Figure 18. Keyboard Data Stream Bit Definitions

The parity bit is either 1 or 0, and the 8 data bits, plus the parity bit, always have an odd number of 1′s.

Note: Mode 1 is a 9-bit data stream that does not have a parity bit or stop bit, and the start bit is always 1.

Data Output

When the keyboard is ready to send data, it first checks for a keyboard-inhibit or system request-to-send status on the 'clock' and 'data' lines. If the 'clock' line is inactive (low), data is stored in the keyboard buffer. If the 'clock' line is active (high) and the 'data' line is inactive (request-to-send), data is stored in the keyboard buffer, and the keyboard receives system data.

If the 'clock' and 'data' lines are both active, the keyboard sends the 0 start bit, 8 data bits, the parity bit, and the stop bit. Data is valid before the trailing edge and beyond the leading edge of the clock pulse. During transmission, the keyboard checks the 'clock' line for an active level at least every 60 milliseconds. If the system lowers the 'clock' line from an active level after the keyboard starts sending data, a condition known as *line contention* occurs, and the keyboard stops sending data. If line contention occurs before the leading edge of the 10th clock signal (parity bit), the keyboard buffer returns the 'clock' and 'data' lines to an active level. If contention does not occur by the 10th clock signal, the keyboard completes the transmission. Following line contention, the system may or may not request the keyboard to resend the data.

Following a transmission, the system can inhibit the keyboard until the system processes the input, or until it requests that a response be sent.

Data Input

When the system is ready to send data to the keyboard, it first checks to see if the keyboard is sending data. If the keyboard is sending, but has not reached the 10th 'clock' signal, the system can override the keyboard output by forcing the keyboard 'clock' line to an inactive (low) level. If the keyboard transmission is beyond the 10th 'clock' signal, the system must receive the transmission.

If the keyboard is not sending, or if the system elects to override the keyboard's output, the system forces the keyboard 'clock' line to an inactive level for more than 60 microseconds while preparing to send data. When the system is ready to send the start bit (the 'data' line will be inactive), it allows the 'clock' line to go to an active (high) level.

The keyboard checks the state of the 'clock' line at intervals of no more than 10 milliseconds. If a system request-to-send signal (RTS) is detected, the keyboard counts 11 bits. After the 10th bit, the keyboard checks for an active level on the 'data' line, and if the line is active, forces it inactive, and counts one more bit. This action signals the system that the keyboard has received its data. On receipt of this signal, the system returns to a ready state, in which it can accept keyboard output, or goes to the inhibited state until it is ready.

If the keyboard 'data' line is found at an inactive level following the 10th bit, a framing error has occurred, and the keyboard continues to count until the 'data' line becomes active. The keyboard then makes the 'data' line inactive and sends a Resend command.

Each system command or data transmission to the keyboard requires a response from the keyboard before the system can send its next output. The keyboard will respond within 20 milliseconds unless the system prevents keyboard output. If the keyboard response is invalid or has a parity error, the system sends the command or data again. However, two-byte commands require special handling. If hex F3 (Set Typematic Rate/Delay), hex F0 (Select Alternate Scan Codes), or hex ED (Set/Reset Mode Indicators) have been sent and acknowledged, and the value byte has been sent but the response is invalid or has a parity error, the system resends both the command and the value byte.

Encode and Usage

The keyboard routine, provided in the ROM BIOS, is responsible for converting the keyboard scan codes into what is called *Extended ASCII*. The extended ASCII codes returned by the ROM routine are mapped to the U.S. English keyboard layout. Some operating systems may make provisions for alternate keyboard layouts by providing an interrupt replacement routine, which resides in the read/write memory. This section discusses only the ROM routine.

Extended ASCII encompasses 1-byte character codes with possible values of 0 to 255, an extended code for certain extended keyboard functions, and functions handled within the keyboard routine or through interrupts.

The character codes are passed through the BIOS keyboard routine to the system or application program. In the following figure "-1" means the combination is suppressed in the keyboard routine. The codes are returned in the AL register.

Key	Base Case	Uppercase	Ctrl	Alt
1	'	~	-1	(*)
2	1	!	-1	(*)
3	2	@	Null(000) (*)	(*)
4	3	#	-1	(*)
5	4	$	-1	(*)
6	5	%	-1	(*)
7	6	^	RS(030)	(*)
8	7	&	-1	(*)
9	8	*	-1	(*)
10	9	(	-1	(*)
11	0	)	-1	(*)
12	-	_	US(031)	(*)
13	=	+	-1	(*)
15	Backspace (008)	Backspace (008)	Del(127)	(*)
16	→| (009)	|← (*)	(*)	(*)
17	q	Q	DC1(017)	(*)
18	w	W	ETB(023)	(*)
19	e	E	ENQ(005)	(*)
20	r	R	DC2(018)	(*)
21	t	T	DC4(020)	(*)
22	y	Y	EM(025)	(*)
23	u	U	NAK(021)	(*)
24	i	I	HT(009)	(*)
25	o	O	SI(015)	(*)
26	p	P	DLE(016)	(*)
27	[	{	Esc(027)	(*)
28	]	}	GS(029)	(*)
29	\	|	FS(028)	(*)
30 Caps Lock	-1	-1	-1	-1
31	a	A	SOH(001)	(*)
32	s	S	DC3(019)	(*)
33	d	D	EOT(004)	(*)
34	f	F	ACK(006)	(*)
35	g	G	BEL(007)	(*)
36	h	H	BS(008)	(*)
37	j	J	LF(010)	(*)
38	k	K	VT(011)	(*)
39	l	L	FF(012)	(*)
40	;	:	-1	(*)
41	'	"	-1	(*)

Figure 19 (Part 1 of 2). Character Codes

Key	Base Case	Uppercase	Ctrl	Alt
43	CR(013)	CR(013)	LF(010)	(*)
44 Shift (Left)	-1	-1	-1	-1
46	z	Z	SUB(026)	(*)
47	x	X	CAN(024)	(*)
48	c	C	ETX(003)	(*)
49	v	V	SYN(022)	(*)
50	b	B	STX(002)	(*)
51	n	N	SO(014)	(*)
52	m	M	CR(013)	(*)
53	,	<	-1	(*)
54	.	>	-1	(*)
55	/	?	-1	(*)
57 Shift (Right)	-1	-1	-1	-1
58 Ctrl (Left)	-1	-1	-1	-1
60 Alt (Left)	-1	-1	-1	-1
61	Space	Space	Space	Space
62 Alt (Right)	-1	-1	-1	-1
64 Ctrl (Right)	-1	-1	-1	-1
90 Num Lock	-1	-1	-1	-1
95	/	/	(*)	(*)
100	*	*	(*)	(*)
105	-	-	(*)	(*)
106	+	+	(*)	(*)
108	Enter	Enter	LF(010)	(*)
110	Esc	Esc	Esc	(*)
112	Null (*)	Null (*)	Null (*)	Null(*)
113	Null (*)	Null (*)	Null (*)	Null(*)
114	Null (*)	Null (*)	Null (*)	Null(*)
115	Null (*)	Null (*)	Null (*)	Null(*)
116	Null (*)	Null (*)	Null (*)	Null(*)
117	Null (*)	Null (*)	Null (*)	Null(*)
118	Null (*)	Null (*)	Null (*)	Null(*)
119	Null (*)	Null (*)	Null (*)	Null(*)
120	Null (*)	Null (*)	Null (*)	Null(*)
121	Null (*)	Null (*)	Null (*)	Null(*)
122	Null (*)	Null (*)	Null (*)	Null(*)
123	Null (*)	Null (*)	Null (*)	Null(*)
125 Scroll Lock	-1	-1	-1	-1
126	Pause(**)	Pause(**)	Break(**)	Pause(**)

(*) Refer to "Extended Functions" on page 44.
(**) Refer to "Special Handling" on page 48.

Figure 19 (Part 2 of 2). Character Codes

The following figure is a list of keys that have meaning only in Num Lock, Shift, or Ctrl states.

The Shift key temporarily reverses the current Num Lock state.

Key	Num Lock	Base Case	Alt	Ctrl
91	7	Home (*)	-1	Clear Screen
92	4	← (*)	-1	Reverse Word(*)
93	1	End (*)	-1	Erase to EOL(*)
96	8	↑ (*)	-1	(*)
97	5	(*)	-1	(*)
98	2	↓ (*)	-1	(*)
99	0	Ins	-1	(*)
101	9	Page Up (*)	-1	Top of Text and Home
102	6	→ (*)	-1	Advance Word (*)
103	3	Page Down (*)	-1	Erase to EOS (*)
104	.	Delete (*,**)	(**)	(**)
105	-	Sys Request	-1	-1
106	+	+ (*)	-1	-1

(*) Refer to "Extended Functions."
(**) Refer to "Special Handling" on page 48.

Figure 20. Special Character Codes

Extended Functions

For certain functions that cannot be represented by a standard ASCII code, an extended code is used. A character code of 000 (null) is returned in AL. This indicates that the system or application program should examine a second code, which indicates the actual function. Usually, but not always, this second code is the scan code of the primary key that was pressed. This code is returned in AH.

The following figure is a list of the extended codes and their functions.

Second Code	**Function**
1	Alt Esc
3	Null Character
14	Alt Backspace
15	|← (Back-tab)
16-25	Alt Q, W, E, R, T, Y, U, I, O, P
26-28	Alt [ ] ↵
30-38	Alt A, S, D, F, G, H, J, K, L
39-41	Alt ; ' '
43	Alt \
44-50	Alt Z, X, C, V, B, N, M
51-53	Alt , . /
55	Alt Keypad *
59-68	F1 to F10 Function Keys (Base Case)
71	Home
72	↑ (Cursor Up)
73	Page Up
74	Alt Keypad -
75	← (Cursor Left)
76	Center Cursor
77	→ (Cursor Right)
78	Alt Keypad +
79	End
80	↓ (Cursor Down)
81	Page Down
82	Ins (Insert)
83	Del (Delete)
84-93	Shift F1 to F10
94-103	Ctrl F1 to F10
104-113	Alt F1 to F10
114	Ctrl PrtSc (Start/Stop Echo to Printer)
115	Ctrl ← (Reverse Word)
116	Ctrl → (Advance Word)
117	Ctrl End (Erase to End of Line-EOL)
118	Ctrl PgDn (Erase to End of Screen-EOS)
119	Ctrl Home (Clear Screen and Home)
120-131	Alt 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = keys 2-13
132	Ctrl PgUp (Top 25 Lines of Text and Cursor Home)
133-134	F11, F12
135-136	Shift F11, F12
137-138	Ctrl F11, F12
139-140	Alt F11, F12
141	Ctrl Up/8

Figure 21 (Part 1 of 2). Keyboard Extended Functions

Second Code	Function
142	Ctrl Keypad -
143	Ctrl Keypad 5
144	Ctrl Keypad +
145	Ctrl Down/2
146	Ctrl Ins/0
147	Ctrl Del/.
148	Ctrl Tab
149	Ctrl Keypad /
150	Ctrl Keypad *
151	Alt Home
152	Alt Up
153	Alt Page Up
155	Alt Left
157	Alt Right
159	Alt End
160	Alt Down
161	Alt Page Down
162	Alt Insert
163	Alt Delete
164	Alt Keypad /
165	Alt Tab
166	Alt Enter

Figure 21 (Part 2 of 2). Keyboard Extended Functions

Shift States

Most shift states are handled within the keyboard routine and are not apparent to the system or application program. In any case, the current status of active shift states is available by calling an entry point in the BIOS keyboard routine. The following keys result in altered shift states:

Shift: This key temporarily shifts keys 1 through 13, 16 through 29, 31 through 41, and 46 through 55, to uppercase (base case if in Caps Lock state). Also, the Shift key temporarily reverses the Num Lock or non-Num Lock state of keys 91 through 93, 96, 98, 99, and 101 through 104.

Ctrl: This key temporarily shifts keys 3, 7, 12, 15 through 29, 31 through 39, 43, 46 through 52, 75 through 89, 91 through 93, 95 through 108, 112 through 124, and 126 to the Ctrl state. The Ctrl key is also used with the Alt and Del keys to initiate the system-reset function, with the Scroll Lock key to initiate the break function, and with the Num Lock key to initiate the pause function. The system-reset, break, and pause functions are described under “Special Handling” on page 48.

Alt: This key temporarily shifts keys 1 through 29, 31 through 43, 46 through 55, 75 through 89, 95, 100, and 105 through 124 to the Alt state. The Alt key is also used with the Ctrl and Del keys to cause a system reset.

The Alt key also allows the user to enter any character code from 1 to 255. The user holds down the Alt key and types the decimal value of the characters desired on the numeric keypad (keys 91 through 93, 96 through 99, and 101 through 103). The Alt key is then released. If the number is greater than 255, a modulo-256 value is used. This value is interpreted as a character code and is sent through the keyboard routine to the system or application program. Alt is handled internally in the keyboard routine.

Caps Lock: This key shifts keys 17 through 26, 31 through 39, and 46 through 52 to uppercase. When Caps Lock is pressed again, it reverses the action. Caps Lock is handled internally in the keyboard routine. When Caps Lock is pressed, it changes the Caps Lock mode indicator. If the indicator was on, it goes off; if it was off, it goes on.

Scroll Lock: When interpreted by appropriate application programs, this key indicates that the cursor-control keys will cause windowing over the text rather than moving the cursor. When the Scroll Lock key is pressed again, it reverses the action. The keyboard routine simply records the current shift state of the Scroll Lock key. It is the responsibility of the application program to perform the function. When Scroll Lock is pressed, it changes the Scroll Lock mode indicator. If the indicator was on, it goes off; if it was off, it goes on.

Num Lock: This key shifts keys 91 through 93, 96 through 99, and 101 through 104 to uppercase. When Num Lock is pressed again, it reverses the action. Num Lock is handled internal to the keyboard routine. When Num Lock is pressed, it changes the Num Lock mode indicator. If the indicator was on, it goes off; if it was off, it goes on.

Shift Key Priorities and Combinations: If combinations of the Alt, Ctrl, and Shift keys are pressed and only one is valid, the priority is: Alt key first, Ctrl key, and Shift key third. The only valid combination is Alt and Ctrl, which is used in the system-reset function.

Special Handling

System Reset

The combination of Alt, Ctrl, and Del keys results in the keyboard routine that starts a system reset or restart. System reset is handled by system BIOS.

Break

The combination of the Ctrl and Pause/Break keys results in the keyboard buffer being cleared, then the keyboard routine signals interrupt 1B, and finally the extended characters AL = hex 00, and AH = hex 00 are stored in the buffer.

Pause

The Pause key causes the keyboard interrupt routine to loop, waiting for any character or function key to be pressed. This provides a method of temporarily suspending an operation, such as listing or printing, and then resuming the operation. The method is not apparent to either the system or the application program. The key stroke used to resume operation is discarded. Pause is handled internally in the keyboard routine.

Print Screen

The Print Screen key results in an interrupt invoking the print-screen routine. This routine works in the alphanumeric or graphics mode, with unrecognizable characters causing blanks.

System Request

When the System Request (Alt and Print Screen) key is pressed, a hex 8500 is placed in AX, and an interrupt hex 15 is executed. When the System Request key is released, a hex 8501 is placed in AX, and another interrupt hex 15 is executed. If an application is to use System Request, the following rules must be observed:

```
Save the previous address.

    Overlay interrupt vector hex 15.

    Check AH for a value of hex 85:

        If yes, process may begin.
        If no, go to previous address.
```

The application program must preserve the value in all registers, except AX, on return. System Request is handled internally in the keyboard routine.

Other Characteristics

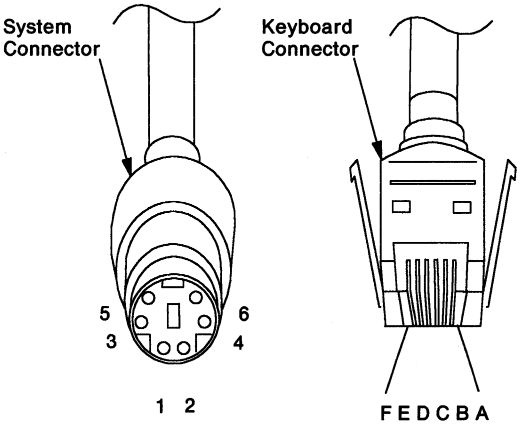
The keyboard routine does its own buffering, and the keyboard buffer is large enough to support entries by a fast typist. However, if a key is pressed when the buffer is full, the key will be ignored and the alarm will sound.

The keyboard routine also suppresses the typematic action of the following keys: Ctrl, Shift, Alt, Num Lock, Scroll Lock, Caps Lock, and Ins.

During each interrupt hex 09 from the keyboard, an interrupt hex 15, function (AH) = hex 4F is generated by the BIOS after the scan code is read from the keyboard adapter. The scan code is passed in the AL register with the carry flag set. This is to allow an operating system to intercept each scan code before it is being handled by the interrupt hex 09 routine and have a chance to change or act on the scan code. If the carry flag is changed to 0 on return from interrupt hex 15, the scan code is ignored by the interrupt handler.

Cables and Connectors

The keyboard cable connects to the system with a 6-pin miniature DIN connector and to the keyboard with a 6-position connector. The following figures show the pin configuration and signal assignments.



DIN Connector Pins	Signal Name	Keyboard Connector Pins
1	+KBD DATA	B
2	Reserved	F
3	Ground	C
4	+5.0 Vdc	E
5	+KBD CLK	D
6	Reserved	A
Shield	Frame Ground	Shield

Figure 22. Keyboard Connectors Signal and Voltage Assignments

Specifications

Specifications for the keyboard are as follows:

Power Requirements

- +5 Vdc ± 10%
- 275 mA.

Size

- Length: 492 millimeters (19.4 inches)
- Depth: 210 millimeters (8.3 inches)
- Height: 58 millimeters (2.3 inches), legs extended.

Weight

- 2.25 kilograms (5.0 pounds).

Index

A

acknowledge (ACK) command 27
alternate key 47
ASCII, extended 41

B

basic assurance test (BAT) 20
BAT completion code command 27
BAT failure code 27
Belgian keyboard 4
break code 19
break key 48
buffer overrun condition 19

C

Canadian keyboard 5
caps lock key 47
character codes 42
clock and data signals 38
combinations, shift key 47
commands from the system 21
control key 46
ctrl state 44

D

Danish keyboard 6
data and clock signals 38
data input, keyboard 40
data output, keyboard 39
data stream 38
default disable command 21
delay, typematic 19
Dutch keyboard 7

E

echo command 22, 27
enable command, keyboard 22
encode, keyboard 41
extended ASCII 41
extended codes, keyboard 44

F

FIFO (first-in-first-out) 19
French keyboard 8

G

German keyboard 9

I

invalid command 22
Italian keyboard 10

K

key detection error command 28
key encode 41
key-code scanning 19
keyboard buffer overrun condition 19
keyboard description 1
keyboard ID command 27
keyboard layouts 1

L

Latin American keyboard 11
line contention, keyboard 39
line protocol, keyboard 20

M

make/break codes 19
mode, keyboard data stream 21

N

Norwegian keyboard 12
num lock state 44
number lock key 47

O

outputs, keyboard 28
overrun command 28
overrun condition, keyboard
buffer 19

P

pause key 48
Portuguese keyboard 13
power-on reset (POR) 20
power-on routine, keyboard 20
print screen key 48
priorities, shift key 47
protocol, keyboard 20

R

rate, typematic 19, 25
read ID command, keyboard 22
resend command 22, 28
reset command 22
reset, power-on (POR) 20
reset, system 48
routine, keyboard 49

S

scan code set 2 31
scan code set 3 35
scan codes, keyboard 28
scanning, key-code 19
scroll lock key 47
select alternate scan codes
command 23
sequential key-code scanning 19
set all keys command 23
set default command 23
set key type command 24
set typematic rate/delay 25
set/reset status indicators
command, keyboard 24
shift key 46
shift key priorities 47
shift state 44, 46
Spanish keyboard 14
stream, data 38
Swedish keyboard 15
Swiss keyboard 16
system request key 49
system reset 48

T

test, basic assurance (BAT) 20
typematic delay 19
typematic keys 19
typematic rate 19, 25

U

U.K. English keyboard 17
U.S. English keyboard 18

Numerics

101-key keyboard 2
102-key keyboard 3

Characters and Keystrokes

Character Codes 1
Quick Reference 8

Notes:

Character Codes

The following figures show the decimal values, hexadecimal values, and keystrokes for each character. The notes referred to in the figures are on page 7.

Value		As Characters		
Hex	**Dec**	**Symbol**	**Keystrokes**	**Notes**
00	0	Blank (Null)	Ctrl 2	
01	1	☺	Ctrl A	
02	2	☻	Ctrl B	
03	3	♥	Ctrl C	
04	4	♦	Ctrl D	
05	5	♣	Ctrl E	
06	6	♠	Ctrl F	
07	7	●	Ctrl G	
08	8	◘	Ctrl H, Backspace, Shift Backspace	
09	9	○	Ctrl I	
0A	10	◙	Ctrl J, Ctrl ↵	
0B	11	♂	Ctrl K	
0C	12	♀	Ctrl L	
0D	13	♪	Ctrl M, ↵, Shift ↵	
0E	14	♫	Ctrl N	
0F	15	☼	Ctrl O	
10	16	►	Ctrl P	
11	17	◄	Ctrl Q	
12	18	↕	Ctrl R	
13	19	‼	Ctrl S	
14	20	¶	Ctrl T	
15	21	§	Ctrl U	
16	22	▬	Ctrl V	
17	23	↨	Ctrl W	

Value		As Characters		
Hex	**Dec**	**Symbol**	**Keystrokes**	**Notes**
18	24	↑	Ctrl X	
19	25	↓	Ctrl Y	
1A	26	→	Ctrl Z	
1B	27	←	Ctrl [Esc, Shift Esc, Crtl Esc	
1C	28	∟	Ctrl	
1D	29	↔	Ctrl]	
1E	30	▲	Ctrl 6	
1F	31	▼	Ctrl -	
20	32	Blank Space	Space Bar, Shift, Space, Ctrl Space, Alt Space	
21	33	!	!	Shift
22	34	"	"	Shift
23	35	#	#	Shift
24	36	$	$	Shift
25	37	%	%	Shift
26	38	&	&	Shift
27	39	'	'	Shift
28	40	(	(	Shift
29	41	)	)	
2A	42	*	*	Note 1
2B	43	+	+	Shift
2C	44	,	,	
2D	45	-	-	
2E	46	.	.	Note 2

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
2F	47	/	/	
30	48	0	0	Note 3
31	49	1	1	Note 3
32	50	2	2	Note 3
33	51	3	3	Note 3
34	52	4	4	Note 3
35	53	5	5	Note 3
36	54	6	6	Note 3
37	55	7	7	Note 3
38	56	8	8	Note 3
39	57	9	9	Note 3
3A	58	:	:	Shift
3B	59	;	;	
3C	60	<	<	Shift
3D	61	=	=	
3E	62	>	>	Shift
3F	63	?	?	Shift
40	64	@	@	Shift
41	65	A	A	Note 4
42	66	B	B	Note 4
43	67	C	C	Note 4
44	68	D	D	Note 4
45	69	E	E	Note 4
46	70	F	F	Note 4
47	71	G	G	Note 4
48	72	H	H	Note 4
49	73	I	I	Note 4
4A	74	J	J	Note 4

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
4B	75	K	K	Note 4
4C	76	L	L	Note 4
4D	77	M	M	Note 4
4E	78	N	N	
4F	79	O	O	Note 4
50	80	P	P	Note 4
51	81	Q	Q	Note 4
52	82	R	R	Note 4
53	83	S	S	Note 4
54	84	T	T	Note 4
55	85	U	U	Note 4
56	86	V	V	Note 4
57	87	W	W	Note 4
58	88	X	X	Note 4
59	89	Y	Y	Note 4
5A	90	Z	Z	Note 4
5B	91	[	[	
5C	92	\	\	Note 4
5D	93	]	]	
5E	94	^	^	Shift
5F	95	–	–	Shift
60	96	‘	‘	
61	97	a	a	Note 5
62	98	b	b	Note 5
63	99	c	c	Note 5
64	100	d	d	Note 5
65	101	e	e	Note 5
66	102	f	f	Note 5

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
67	103	g	g	Note 5
68	104	h	h	Note 5
69	105	i	i	Note 5
6A	106	j	j	Note 5
6B	107	k	k	Note 5
6C	108	l	l	Note 5
6D	109	m	m	Note 5
6E	110	n	n	Note 5
6F	111	o	o	Note 5
70	112	p	p	Note 5
71	113	q	q	Note 5
72	114	r	r	Note 5
73	115	s	s	Note 5
74	116	t	t	Note 5
75	117	u	u	Note 5
76	118	v	v	Note 5
77	119	w	w	Note 5
78	120	x	x	Note 5
79	121	y	y	Note 5
7A	122	z	z	Note 5
7B	123	{	{	Shift
7C	124	¦	¦	Shift
7D	125	}	}	Shift
7E	126	~	~	Shift
7F	127	△	Ctrl-	

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
80	128	Ç	Alt 128	Note 6
81	129	ü	Alt 129	Note 6
82	130	é	Alt 130	Note 6
83	131	â	Alt 131	Note 6
84	132	ä	Alt 132	Note 6
85	133	à	Alt 133	Note 6
86	134	å	Alt 134	Note 6
87	135	ç	Alt 135	Note 6
88	136	ê	Alt 136	Note 6
89	137	ë	Alt 137	Note 6
8A	138	è	Alt 138	Note 6
8B	139	ï	Alt 139	Note 6
8C	140	î	Alt 140	Note 6
8D	141	ì	Alt 141	Note 6
8E	142	Ä	Alt 142	Note 6
8F	143	Å	Alt 143	Note 6
90	144	É	Alt 144	Note 6
91	145	æ	Alt 145	Note 6
92	146	Æ	Alt 146	Note 6
93	147	ô	Alt 147	Note 6
94	148	ö	Alt 148	Note 6
95	149	ò	Alt 149	Note 6
96	150	û	Alt 150	Note 6
97	151	ù	Alt 151	Note 6
98	152	ÿ	Alt 152	Note 6
99	153	Ö	Alt 153	Note 6
9A	154	Ü	Alt 154	Note 6

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
9B	155	¢	Alt 155	Note 6
9C	156	£	Alt 156	Note 6
9D	157	¥	Alt 157	Note 6
9E	158	Pt	Alt 158	Note 6
9F	159	ƒ	Alt 159	Note 6
A0	160	á	Alt 160	Note 6
A1	161	í	Alt 161	Note 6
A2	162	ó	Alt 162	Note 6
A3	163	ú	Alt 163	Note 6
A4	164	ñ	Alt 164	Note 6
A5	165	Ñ	Alt 165	Note 6
A6	166	ª	Alt 166	Note 6
A7	167	º	Alt 167	Note 6
A8	168	¿	Alt 168	Note 6
A9	169	⌐	Alt 169	Note 6
AA	170	¬	Alt 170	Note 6
AB	171	½	Alt 171	Note 6
AC	172	¼	Alt 172	Note 6
AD	173	¡	Alt 173	Note 6
AE	174	< <	Alt 174	Note 6
AF	175	> >	Alt 175	Note 6
B0	176	░	Alt 176	Note 6
B1	177	▒	Alt 177	Note 6
B2	178	▓	Alt 178	Note 6
B3	179	│	Alt 179	Note 6
B4	180	┤	Alt 180	Note 6
B5	181	╡	Alt 181	Note 6
B6	182	╢	Alt 182	Note 6
B7	183	╖	Alt 183	Note 6
B8	184	╕	Alt 184	Note 6
B9	185	╣	Alt 185	Note 6
BA	186	║	Alt 186	Note 6
BB	187	╗	Alt 187	Note 6
BC	188	╝	Alt 188	Note 6
BD	189	╜	Alt 189	Note 6
BE	190	╛	Alt 190	Note 6
BF	191	┐	Alt 191	Note 6
C0	192	└	Alt 192	Note 6
C1	193	┴	Alt 193	Note 6
C2	194	┬	Alt 194	Note 6
C3	195	├	Alt 195	Note 6
C4	196	─	Alt 196	Note 6
C5	197	┼	Alt 197	Note 6
C6	198	╞	Alt 198	Note 6
C7	199	╟	Alt 199	Note 6
C8	200	╚	Alt 200	Note 6
C9	201	╔	Alt 201	Note 6
CA	202	╩	Alt 202	Note 6
CB	203	╦	Alt 203	Note 6
CC	204	╠	Alt 204	Note 6
CD	205	═	Alt 205	Note 6
CE	206	╬	Alt 206	Note 6
CF	207	╧	Alt 207	Note 6
D0	208	╨	Alt 208	Note 6

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
D1	209	╤	Alt 209	Note 6
D2	210	╥	Alt 210	Note 6
D3	211	╙	Alt 211	Note 6
D4	212	╘	Alt 212	Note 6
D5	213	╒	Alt 213	Note 6
D6	214	╓	Alt 214	Note 6
D7	215	╫	Alt 215	Note 6
D8	216	╪	Alt 216	Note 6
D9	217	┘	Alt 217	Note 6
DA	218	┌	Alt 218	Note 6
DB	219	█	Alt 219	Note 6
DC	220	▄	Alt 220	Note 6
DD	221	▌	Alt 221	Note 6
DE	222	▐	Alt 222	Note 6
DF	223	▀	Alt 223	Note 6
E0	224	α	Alt 224	Note 6
E1	225	β	Alt 225	Note 6
E2	226	Γ	Alt 226	Note 6
E3	227	π	Alt 227	Note 6
E4	228	Σ	Alt 228	Note 6
E5	229	σ	Alt 229	Note 6
E6	230	μ	Alt 230	Note 6
E7	231	τ	Alt 231	Note 6
E8	232	Φ	Alt 232	Note 6
E9	233	θ	Alt 233	Note 6
EA	234	Ω	Alt 234	Note 6
EB	235	δ	Alt 235	Note 6

Value		As Characters		
Hex	Dec	Symbol	Keystrokes	Notes
EC	236	∞	Alt 236	Note 6
ED	237	φ	Alt 237	Note 6
EE	238	ε	Alt 238	Note 6
EF	239	∩	Alt 239	Note 6
F0	240	≡	Alt 240	Note 6
F1	241	±	Alt 241	Note 6
F2	242	≥	Alt 242	Note 6
F3	243	≤	Alt 243	Note 6
F4	244	⌠	Alt 244	Note 6
F5	245	⌡	Alt 245	Note 6
F6	246	÷	Alt 246	Note 6
F7	247	≈	Alt 247	Note 6
F8	248	°	Alt 248	Note 6
F9	249	●	Alt 249	Note 6
FA	250	•	Alt 250	Note 6
FB	251	√	Alt 251	Note 6
FC	252	ⁿ	Alt 252	Note 6
FD	253	²	Alt 253	Note 6
FE	254	■	Alt 254	Note 6
FF	255	BLANK	Alt 255	Note 6

Notes:

1. Asterisk (*) can be typed by pressing the * key or, in the shift state, pressing the 8 key.
2. Period (.) can be typed by pressing the . key or, in the shift or Num Lock state, pressing the Del key.
3. Numeric characters 0-9 can be typed by pressing the numeric keys on the keyboard or, in the shift or Num Lock state, pressing the numeric keys in the keypad portion of the keyboard.
4. Uppercase alphabetic characters (A-Z) can be typed by pressing the character key in the shift state or the Caps Lock state.
5. Lowercase alphabetic characters (a-z) can be typed by pressing the character key in the normal state or in Caps Lock and shift state combined.
6. The three digits are typed on the numeric keypad while pressing the Alt key. Character codes 001-255 may be entered in this fashion (with Caps Lock activated, character codes 97-122 display in uppercase).

Quick Reference

DECIMAL VALUE	➡	0	16	32	48	64	80	96	112
⬇	HEXA-DECIMAL VALUE	0	1	2	3	4	5	6	7
0	0	BLANK (NULL)	►	BLANK (SPACE)	0	@	P	‘	p
1	1	☺	◄	!	1	A	Q	a	q
2	2	☻	↕	"	2	B	R	b	r
3	3	♥	‼	#	3	C	S	c	s
4	4	♦	¶	$	4	D	T	d	t
5	5	♣	§	%	5	E	U	e	u
6	6	♠	▬	&	6	F	V	f	v
7	7	•	↨	’	7	G	W	g	w
8	8	◘	↑	(	8	H	X	h	x
9	9	○	↓	)	9	I	Y	i	y
10	A	◙	→	*	:	J	Z	j	z
11	B	♂	←	+	;	K	[	k	{
12	C	♀	∟	,	<	L	\	l	¦
13	D	♪	↔	—	=	M	]	m	}
14	E	♫	▲	.	>	N	^	n	~
15	F	☼	▼	/	?	O	_	o	△

DECIMAL VALUE	➡	128	144	160	176	192	208	224	240
⬇	HEXA-DECIMAL VALUE	8	9	A	B	C	D	E	F
0	0	Ç	É	á	░	└	╨	∝	≡
1	1	ü	æ	í	▒	┴	╤	β	±
2	2	é	Æ	ó	▓	┬	╥	Γ	≥
3	3	â	Ô	ú	│	├	╙	π	≤
4	4	ä	ö	ñ	┤	─	╘	Σ	⌠
5	5	à	ò	Ñ	╡	┼	╒	σ	⌡
6	6	å	û	a̲	╢	╞	╓	μ	÷
7	7	ç	ù	o̲	╖	╟	╫	ϒ	≈
8	8	ê	ÿ	¿	╕	╚	╪	Φ	°
9	9	ë	Ö	⌐	╣	╔	┘	Θ	•
10	A	è	Ü	¬	║	╩	┌	Ω	•
11	B	ï	¢	½	╗	╦	█	δ	√
12	C	î	£	¼	╝	╠	▄	∞	n
13	D	ì	¥	¡	╜	═	▌	ϕ	2
14	E	Ä	₧	«	╛	╬	▐	∈	■
15	F	Å	ƒ	»	┐	╧	▀	∩	BLANK 'FF'

Notes:

Power Supply (Types 1 and 2)

Description 1
Input Protection 1
Ground Leakage Current 1
Outputs 2
Output Protection 2
Voltage Sequencing 2
No-Load Operation 2
Power Supply Connector (Type 1) 3
Power Supply Connectors (Type 2) 4

Figures

1. Power Supply Connector (Type 1) 3
2. Power Supply Connector Voltages and Signal Assignments . 3
3. System Board Power Supply Cable Connector (Type 2) 4
4. System Board Power Supply Connector Voltage and Signal Assignments 4
5. Fixed Disk Drive Power Supply Cable Connectors (Type 2) . 5
6. Fixed Disk Drive Power Supply Connectors Voltage and Signal Assignments 5

Description

The power supply converts the ac input voltage to three dc outputs. A description of the input voltage ranges to the power supply is in the system-specific technical references. The power supply provides power for the following:

- System board
- Channel adapters
- Diskette drives
- Fixed disk drives
- Auxiliary device
- Keyboard.

The power switch and two light-emitting diodes (LEDs) are on the front of the system unit. The green LED indicates that the power supply is operating. The yellow LED indicates fixed disk drive activity.

Input Protection

The input power line is protected against an overcurrent condition by an internal fuse.

Ground Leakage Current

The system unit ground leakage current does not exceed 500 microamperes at any nominal input voltage.

Outputs

The power supply provides three voltages: +5, +12, and −12 Vdc.

Output Protection

A short circuit placed on any dc output (between outputs or between an output and dc return) latches all dc outputs into a shutdown state with no damage to the power supply.

If an overvoltage fault occurs (internal to the power supply), the power supply latches all dc outputs into a shutdown state before any output exceeds 130% of its nominal value.

If either of these shutdown states is actuated, the power supply returns to normal operation only after the fault has been removed and the power switch has been turned off for at least ten seconds.

Voltage Sequencing

At power-on time, the output voltages track within 50 milliseconds of each other when measured at the 50% points.

No-Load Operation

The power supply is capable of operation with "no load" on the outputs. The output regulation is ± 25% and the 'power-good' signal can be either active or inactive.

Power Supply Connector (Type 1)

The Type 1 power supply uses a 50-pin card-edge connector mounted on the side of the power supply. The system board plugs into the card-edge connector, eliminating the need for separate cabling.

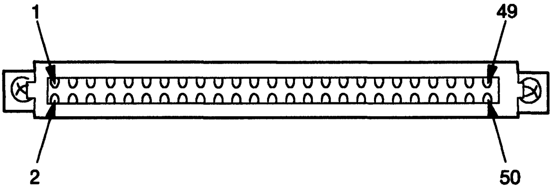


Figure 1. Power Supply Connector (Type 1)

Pin	I/O	Signal	Pin	I/O	Signal
1	N/A	−12 Vdc	2	N/A	Signal Ground
3	N/A	+12 Vdc	4	N/A	Signal Ground
5	N/A	+12 Vdc	6	N/A	Signal Ground
7	N/A	+12 Vdc	8	N/A	Signal Ground
9	N/A	+12 Vdc	10	N/A	Signal Ground
11	N/A	+12 Vdc	12	N/A	Signal Ground
13	N/A	+12 Vdc	14	N/A	Signal Ground
15	N/A	+5 Vdc	16	N/A	Signal Ground
17	N/A	+5 Vdc	18	N/A	Signal Ground
19	N/A	+5 Vdc	20	N/A	Signal Ground
21	N/A	+5 Vdc	22	N/A	Signal Ground
23	N/A	+5 Vdc	24	N/A	Signal Ground
25	N/A	+5 Vdc	26	N/A	Signal Ground
27	N/A	+5 Vdc	28	N/A	Signal Ground
29	N/A	+5 Vdc	30	N/A	Signal Ground
31	N/A	+5 Vdc	32	N/A	Signal Ground
33	N/A	+5 Vdc	34	N/A	Signal Ground
35	N/A	+5 Vdc	36	N/A	Signal Ground
37	N/A	+5 Vdc	38	N/A	Signal Ground
39	N/A	+5 Vdc	40	N/A	Signal Ground
41	N/A	+5 Vdc	42	N/A	Signal Ground
43	N/A	+5 Vdc	44	N/A	Signal Ground
45	N/A	+5 Vdc	46	N/A	Signal Ground
47	N/A	+5 Vdc	48	N/A	Signal Ground
49	I	System Status	50	O	Power Good

Figure 2. Power Supply Connector Voltages and Signal Assignments

Power Supply Connectors (Type 2)

The Type 2 power supply uses a 15-pin connector for the system board and two 4-pin connectors for the two fixed-disk-drive locations.

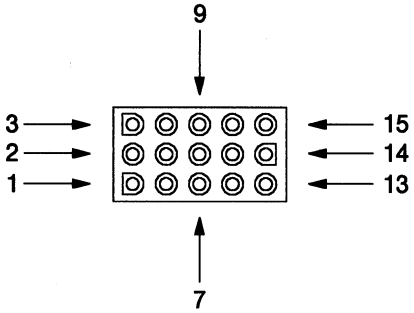


Figure 3. System Board Power Supply Cable Connector (Type 2)

Pin	I/O	Signal	Pin	I/O	Signal
1	N/A	+5 Vdc	2	N/A	Signal Ground
3	N/A	+12 Vdc	4	N/A	+5 Vdc
5	N/A	Signal Ground	6	N/A	Signal Ground
7	N/A	+5 Vdc	8	N/A	Signal Ground
9	N/A	−12 Vdc	10	N/A	+5 Vdc
11	N/A	Signal Ground	12	O	Power Good
13	N/A	+5 Vdc	14	N/A	Signal Ground
15	I	System Status			

Figure 4. System Board Power Supply Connector Voltage and Signal Assignments

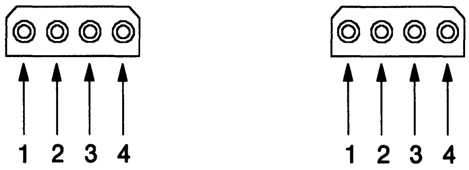


Figure 5. Fixed Disk Drive Power Supply Cable Connectors (Type 2)

Pin	I/O	Signal	Pin	I/O	Signal
1	N/A	+12 Vdc	2	N/A	Signal Ground
3	N/A	Signal Ground	4	N/A	+5 Vdc

Figure 6. Fixed Disk Drive Power Supply Connectors Voltage and Signal Assignments

Notes:

Power Supply (Type 3)

Description 1
Inputs 1
Outputs 2
Output Protection 2
Voltage Sequencing 2
No-Load Operation 2
'Power Good' Signal 2
Power Supply Connectors 3

Notes:

Description

The Type 3 power supply provides power for the following:

- System board
- Channel adapters
- Diskette drive
- Fixed disk drive
- Auxiliary device
- Keyboard.

Inputs

The Type 3 power supply operates with two ranges of input power. They are selected through the use of a mechanical switch. The following table shows both ranges of input power.

Table 1-1. Input Voltage (Vac)

Nominal	Range
100 - 125	90 - 137
200 - 240	180 - 265

The system-unit-ground leakage current does not exceed 500 microamperes at any nominal input voltage.

Refer to the system-specific technical references for additional electrical information.

Outputs

The power supply provides three voltages to the system board: +5, +12, and −12 Vdc.

Output Protection

A short circuit placed on any dc output (between outputs or between an output and dc return) latches all dc outputs into a shut-down state with no damage to the power supply.

If an overvoltage fault occurs (internal to the power supply), the supply latches all dc outputs into a shut-down state before any output exceeds 130% of its nominal value.

If either of these shut-down states is actuated, the power supply returns to normal operation only after the fault has been removed and the power switch has been turned off for at least 10 seconds.

Voltage Sequencing

At power-on time, all voltage must be within the regulation tolerance for a minimum of 200 milliseconds before the 'power good' signal goes active.

No-Load Operation

The power supply is capable of operation with no load on the outputs. The output regulation is ± 25%, and the 'power good' signal can be either active or inactive.

'Power Good' Signal

A 'power good' signal indicates proper operation of the power supply and is active (high) during normal operation. The 'power good' signal is pulled up with a 10-kilohm resistor on the system board.

This signal also resets system logic. At power-on time, this signal is low for a minimum of 200 milliseconds but not greater than 650 milliseconds after all outputs have reached specified operating limits.

Power Supply Connectors

The power supply is attached to the system board through a 12-pin connector (J7) and a 6-pin connector (J14). The following table shows the signals and voltages assigned to the power supply output connectors.

Table 1-2. Power Supply Connectors

J7 Pin	Assignments	J14 Pin	Assignments
1	Power Good	1	Ground
2	Ground	2	Ground
3	+12 Vdc	3	+5 Vdc
4	−12 Vdc	4	+5 Vdc
5	Ground	5	+5 Vdc
6	Ground		
7	Ground		
8	Ground		
9	No Connection		
10	+5 Vdc		
11	+5 Vdc		
12	+5 Vdc		

Notes:

Compatibility

Description 1
Application Guidelines 1
Hardware Interrupts 1
Software Interrupts 3
High-Level Language Considerations 3
Assembler Language Programming Considerations 3
Opcodes 4
80286 Anomalies 8
80386 Anomalies 8
ROM BIOS and Operating System Function Calls 15
Software Compatibility 18
Multitasking Provisions 19
Interfaces 19
Classes 20
Time-Outs 21
Machine-Sensitive Programs 22
Math Coprocessor Compatibility 22
80287 to 80387 Compatibility 23
Exceptions 24
8087 to 80287 Compatibility 25
Diskette Drives and Controller 27
Fixed Disk Drives and Controller 28

Index 29

Figures

1. String Instruction/Register Size Mismatch 11
2. Write and Format Head Settle Time 17
3. Functional Code Assignments 21
4. Math Coprocessor Software Compatibility 23
5. Diskette Drive Read, Write, and Format Capabilities 27

Description

The differences in system microprocessors, math coprocessors, general system architecture, and diskette drive capabilities must be taken into consideration when designing application programs exclusively for a specific model or programs compatible across the IBM Personal Computer and Personal System/2 product lines. This section discusses these major differences and provides some suggestions to aid you in developing your program.

Application Guidelines

Use the following information to develop application programs for the IBM Personal Computer and Personal System/2 products. Whenever possible, BIOS should be used as an interface to hardware in order to provide maximum compatibility and portability of applications across systems.

Hardware Interrupts

Hardware interrupts are level-sensitive for systems using the Micro Channel architecture while systems using the Personal Computer type I/O channel design have edge-triggered hardware interrupts. The interrupt controller clears its in-service register bit when the interrupt routine sends an End-of-Interrupt (EOI) command to the controller. The EOI command is sent whether the incoming interrupt request to the controller is active or inactive.

In level-sensitive systems, the interrupt-in-progress latch is readable at an I/O address bit position. This latch is read during the interrupt service routine and may be reset by the read operation or may require an explicit reset.

Note: Designers may want to limit the number of devices sharing an interrupt level for performance and latency considerations.

The interrupt controller on level-sensitive systems requires the interrupt request to be inactive at the time the EOI command is sent; otherwise, a "new" interrupt request will be detected and another microprocessor interrupt caused.

To avoid this problem, a level-sensitive interrupt handler must clear the interrupt condition (usually by a Read or Write to an I/O port on the device causing the interrupt). After clearing the interrupt condition, a JMP $+2 should be executed prior to sending the EOI command to the interrupt controller. This ensures that the interrupt request is removed prior to re-enabling the interrupt controller. Another JMP $+2 should be executed after sending the EOI command, but prior to enabling the interrupt through the Set Interrupt Enable Flag (STI) command.

In the level-sensitive systems, hardware prevents the interrupt controllers from being set to the edge-triggered mode.

Hardware interrupt IRQ9 is defined as the replacement interrupt level for the cascade level IRQ2. Program interrupt sharing should be implemented on IRQ2, interrupt hex 0A. The following processing occurs to maintain compatibility with the IRQ2 used by IBM Personal Computer products:

1. A device drives the interrupt request active on IRQ2 of the channel.
2. This interrupt request is mapped in hardware to IRQ9 input on the second interrupt controller.
3. When the interrupt occurs, the system microprocessor passes control to the IRQ9 (interrupt hex 71) interrupt handler.
4. This interrupt handler performs an EOI command to the second interrupt controller and passes control to IRQ2 (interrupt hex 0A) interrupt handler.
5. This IRQ2 interrupt handler, when handling the interrupt, causes the device to reset the interrupt request prior to performing an EOI command to the master interrupt controller that finishes servicing the IRQ2 request.

Software Interrupts

With the advent of software interrupt sharing, software interrupt routines must daisy chain interrupts. Each routine must check the function value and if it is not in the range of function calls for that routine, it must transfer control to the next routine in the chain. Because software interrupts are initially pointed to 0:0 before daisy chaining, it is necessary to check for this case. If the next routine is pointed to 0:0 and the function call is out of range, the appropriate action is to set the carry flag and do a RET 2 to indicate an error condition.

High-Level Language Considerations

The IBM-supported languages of IBM C, BASIC, FORTRAN, COBOL, and Pascal are the best choices for writing compatible programs.

If a program uses specific features of the hardware, that program may not be compatible with all IBM Personal Computer and Personal System/2 products. Specifically, the use of assembler language subroutines or hardware-specific commands (for example In, Out, Peek, and Poke) must follow the assembler language rules. See "Assembler Language Programming Considerations."

Any program that requires precise timing information should obtain it through an operating system or language interface; for example, TIME$ in BASIC. If greater precision is required, the assembler techniques in "Assembler Language Programming Considerations" are available. The use of programming loops may prevent a program from being compatible with other IBM Personal Computer products, IBM Personal System/2 products, and software.

Assembler Language Programming Considerations

This section describes fundamental differences between the systems in the Personal Computer and Personal System/2 product lines that may affect program development.

Opcodes

The following opcodes work differently on systems using either the 80286 or 80386 microprocessor than they do on systems using the 8088 or 8086 microprocessor.

- PUSH SP

  The 80286 and 80386 microprocessors push the current stack pointer; the 8088 and 8086 microprocessors push the new stack pointer, that is, the value of the stack pointer after the PUSH SP instruction is completed.

- Single step interrupt (when TF=1) on the interrupt instruction (Opcode hex CC, CD):

  The 80286 and 80386 microprocessors do not perform a single-step interrupt on the INT instruction; the 8088 and 8086 microprocessors do perform a single-step interrupt on the INT instruction.

- The divide error exception (interrupt 0):

  The 80286 and 80386 microprocessors push the CS:IP of the instruction that caused the exception; the 8088 and 8086 microprocessors push the CS:IP of the instruction following the instruction that caused the exception.

- Shift counts for the 80286 and 80386 microprocessors:

  Shift counts are masked to 5 bits. Shift counts greater than 31 are treated mod 32. For example, a shift count of 36 shifts the operand four places.

- LOCK prefix:

  When the LOCK prefix is used with an instruction, the system microprocessor executes the entire instruction before allowing interrupts. If a Repeat String Move instruction is locked, interrupts may be disabled for a long duration.

  The 8088, 8086, and 80286 microprocessors allow the LOCK prefix to be used with most instructions. However, the 80386 microprocessor restricts the use of LOCK to the following instructions:

  - Bit Test and Set Memory, Register/Immediate
  - Bit Test and Reset Memory, Register/Immediate

- Bit Test and Complement Memory, Register/Immediate
- XCHG Register, Memory
- XCHG Memory, Register
- ADD, OR, ADC, SBB, Memory, Register/Immediate
- AND, SUB, XOR Memory, Register/Immediate
- NOT, NEG, INC, DEC Memory.

An undefined opcode trap (INT 6) is generated if the LOCK prefix is used in the 80386 environment with an instruction not listed.

When the 80286 is operating in the virtual memory mode, the LOCK prefix is IOPL-sensitive. Since the 80386 restricts the use of the LOCK prefix to a specific set of instructions, the LOCK prefix is not IOPL-sensitive in the 80386 environment.

- Multiple lockout instructions:

  There are several microprocessor instructions that, when executed, lock out external bus signals. DMA requests are not honored during the execution of these instructions. Consecutive instructions of this type prevent DMA activity from the start of the first instruction to the end of the last instruction. To allow for necessary DMA cycles, as required by the diskette controller in a multitasking system, multiple lock-out instructions must be separated by a JMP SHORT $+2.

- Back-to-back I/O commands:

  Back-to-back I/O commands to the same I/O ports do not permit enough recovery time for some I/O adapters. To ensure enough time, a JMP SHORT $+2 must be inserted between IN/OUT instructions to the same I/O adapters.

  Note: MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
OUT  IO_ADD,AL
```

- I/O commands followed by an STI instruction:

  I/O commands followed immediately by an STI instruction do not permit enough recovery time for some system board and channel operations. To ensure enough time, a JMP SHORT $+2 must be inserted between the I/O command and the STI instruction.

Note: MOV AL,AH type instruction does not allow enough recovery time. An example of the correct procedure follows:

```
OUT  IO_ADD,AL
JMP  SHORT $+2
MOV  AL,AH
STI
```

- NT bit and IOPL bits:

  When the 80286 is operating in the Real Address mode, the NT and IOPL bits in the flag register cannot be changed; the bits are zero.

  The 80386 allows the NT bit and the IOPL bits to be modified by POP stack into flags, and other instructions, while operating in the Real Address mode. This has no effect on the Real Address mode operation. However, upon entering Protected Mode operation, the NT bit should be cleared to prevent erroneous execution of the IRET instruction. If NT is set, the IRET attempts to perform a task switch to the previous task.

- Overlap of OUT and following instructions:

  The 80386 has a delayed write to memory and delayed output-to-I/O capability. It is possible for the actual output cycle to I/O devices to occur after the completion of instructions following the Out instruction. Under certain conditions, this may cause some programs to behave in an undesirable manner. For example, an interrupt handler routine may output an EOI command to the interrupt controller to drop the interrupt request. If the interrupt handler has an STI instruction following the output instruction, the 80386 may re-enable interrupts before the interrupt controller drops the interrupt request. This could cause the interrupt routine to be reentered.

  To avoid this problem, either of the following procedures may be used:

  - Place a JMP SHORT $+2 instruction between the OUT instruction and the STI instruction, or

  - Read back the status from the interrupt controller before executing the STI instruction.

- Math coprocessor instructions:

  In 80386-based systems, the mode of the microprocessor and math coprocessor are tightly coupled. This is not the case for 80286-based systems. The 80286-based systems require the math coprocessor FSETPM instruction to be executed to enable the 80287 to operate in the Protected mode. The 80287 remains in the Protected mode until it is reset.

  The mode of the 80287 determines the format in which the math coprocessor state information is saved by the FSTENV and FSAVE instructions. In the Protected mode, the instruction and data operand pointers are saved as selector/offset pairs; in the Real Address mode, the physical address and opcode are saved.

  If the FSETPM instruction is encountered in the 80386 environment, it is ignored. The formatting is performed by the 80386, which internally maintains the instruction and data operand pointers. The Real Address mode format image is saved when the 80386 is operating in the Real Address mode or Virtual 8086 mode. The Protected mode format is used otherwise.

  See also "Math Coprocessor Compatibility" on page 22 for more information.

- Use of 32-bit registers and the 32-bit addressing mode:

  It is possible to use the 32-bit registers and 32-bit addressing mode in all operating modes of the 80386 through the use of the operand-size prefix or address-size prefix.

  In a multitasking environment, extreme care must be taken to avoid conflicts with other tasks that use extended registers. If the operating system saves the extended 32-bit registers and new segment registers in the task context save area, conflicts will be avoided; if the operating system does not provide this function, another method must be implemented.

  One possible method is to disable the interrupts while using the extended registers. The extended registers should be saved before use and restored immediately after use while the interrupts are still disabled. The time that interrupts are disabled should be kept as short as possible.

- Operand Alignment:

  When multiple bus cycles are required to transfer a multibyte logical operand (for example, a word operand beginning at an address not evenly divisible by 2), the 80386 transfers the highest order bytes first.

  This characteristic may affect adapters with memory-mapped I/O that require or assume that sequential memory accesses are made to the memory I/O ports.

  This problem may be avoided by using a REP MOVB(yte) instead of a REP MOVSW(ord).

80286 Anomalies

In the Protected mode, when any of the null selector values (hex 0000, 0001, 0002, and 0003) are loaded into the DS or ES registers with a MOV or POP instruction or a task switch, the 80286 always loads the null selector hex 0000 into the corresponding register.

If a coprocessor (80287) operand is read from an "executable and readable" and conforming (ERC) code segment, and the coprocessor operand is sufficiently near the segment limit that the second or subsequent byte lies outside the limit, an interrupt 9 will not be generated.

The following describes the operation of all 80286 parts:

- Instructions longer than 10 bytes (instructions using multiple redundant prefixes) generate an interrupt 13 (General Purpose Exception) in both the Real Address mode and Protected mode.
- If the second operand of an ARPL instruction is a null selector, the instruction generates an interrupt 13.

80386 Anomalies

The following describes anomalies that apply to the B-1 stepping level of the 80386 microprocessor.

80386 Real Address Mode Operation

- FSAVE/FSTENV opcode field incorrect:

The opcode of some numeric instructions is saved incorrectly in the FSAVE/FSTENV format image when the 80386 is operating in the Real Address mode or Virtual 8086 mode.

The power-on self-test (POST) code in the system ROM enables hardware interrupt 13 and sets up its vector (INT hex 75) to point to a math coprocessor exception routine in ROM. Any time this routine is executed as a result of an exception, it repairs the opcode field by performing the following sequence:

1. Clears the 'busy' signal latch
2. Executes FNSTENV (save image on stack)
3. Extracts instruction pointer from FSTENV memory image
4. Skips over prefix bytes until opcode is found
5. Inserts correct opcode information in the memory image
6. Executes FLDENV to restore the corrected opcode field
7. Writes the EOI command to the interrupt controller
8. Transfers control to the address pointed to by the NMI handler.

Any math coprocessor application containing an NMI handler should require its NMI handler to read the status of the coprocessor to determine if the NMI was caused by the coprocessor. If the interrupt was not generated by the coprocessor, control should be passed to the original NMI handler.

Applications do not require any modification for this errata because the BIOS exception routine repairs the opcode field after exceptions. However, if a debugger is used to display the math coprocessor state information, the opcode field will contain an incorrect value for some math coprocessor instructions.

- Single stepping repeated MOVS:

  If a repeated MOVS instruction is executed when single-stepping is turned on (TF = 1 in the EFLAGS register), a single-step interrupt is taken after two move steps on the 80386 microprocessor. The 8088, 8086, and 80286 microprocessors take a single-step interrupt after every iteration step. However, for the 80386, if a data breakpoint is encountered on the first iteration of a repeated MOVS, the data break is not taken until after the second iteration. Data breakpoints encountered on the second and subsequent iterations stop immediately after the step causing a break.

- Wrong register size for string instructions:

  One of the (E)CX, (E)SI, or (E)DI registers will not be updated properly if certain string and loop instructions are followed by instructions that either:

  - Use a different address size (that is, either the string instruction or the following instruction uses an address size prefix), or
  - Reference the stack (such as PUSH/POP/CALL/RET) and the "B" bit in the SS descriptor is different from the address size used by the instructions.

  The size of the register (16 bits or 32 bits) is taken from the instruction following the string instruction rather than from the string instruction itself. This could result in one of the following conditions:

  - Only the lower 16 bits of a 32-bit instruction updated (if the 32-bit string instruction was followed by an instruction using a memory operand addressed with a 16-bit address).
  - All 32 bits of a register updated rather than just the lower 16 bits.

  The following is a list of the instructions and the affected registers:

Instruction	**Register**
REP MOVS	(E)SI
MOVS	(E)DI
STOS	(E)DI
INS	(E)DI
REP INS	(E)CX

Notes:

1. A 32-bit effective address size specified with a string instruction indicates that the 32-bit ESI and EDI registers should be used for forming addresses, and the 32-bit ECX register should be used as the count register.
2. A 32-bit operand size on a repeated string move (MOVS) should be used only if the compiler or programmer can guarantee that the strings do not overlap destructively. An 8-bit or 16-bit MOVS has a predictable effect when the strings overlap destructively.

Figure 1. String Instruction/Register Size Mismatch

The problem only occurs if instructions with different address sizes are mixed, or if a code segment of one size is used with a stack segment of the other size.

To avoid this problem, add a NOP instruction after each of the instructions listed in Figure 1 on page 10 and ensure that the NOP instruction has the same address size as the string/loop instruction. If necessary, an address size prefix hex 67 may precede the NOP instruction.

- Wrong ECX update with REP INS:

  ECX (or CX in a 16-bit address size) is not updated correctly in the case of a REP INS[1] followed by an early start instruction[2]. After executing any repeat-prefixed instruction, the contents of ECX is supposed to be 0, but in the case of an REP INS instruction, ECX is not updated correctly and its contents become hex FFFFFFFF for 32-bit address size operations and hex 0FFFF for 16-bit address size operations. INS is still executed the correct number of times and EDI is updated properly.

  To avoid this problem, one of the following procedures may be used:

  - Insert an explicit MOV ECX,0 (or MOV CX,0) instruction after any REP INS instruction. This ensures that the contents of ECX or CX are 0.
  - Do not rely on the count in ECX (or CX) after a REP INS instruction but instead, move a new count into ECX (or CX) before using it again.

- Test register access fails:

  Accessing the Translate Lookaside Buffer (TLB) test registers, TR6 and TR7, may not function properly.

  Avoid using test registers TR6 and TR7 to test the TLB.

[1] REP INS refers to any input string instruction with a repeat prefix.

[2] An early start instruction refers to PUSH, POP, or memory reference instructions.

80286 Compatible Protected Mode Operation

- Math coprocessor Save/Restore environment operands:

  If either of the last two bytes of an FSAVE/FRSTOR or FSTENV/FLDENV is not accessible, the instruction cannot be restarted. An FNINIT instruction must be issued to the math coprocessor before any other math coprocessor instruction can be executed. This problem arises only in demand-paged systems, or demand-segmented systems that increase the segment size on demand.

- Wrap-around math coprocessor operands:

  The 80386 architecture does not permit a math coprocessor operand, or any other operand, to wrap around the end of a segment. If such an instruction is issued in a protected segmented system, and the operand starts and ends in valid parts of a segment, but passes through an inaccessible region of the segment, the math coprocessor may be put in an indeterminate state. Under these conditions, an FCLEX or FINIT must be sent before any other math coprocessor instruction is issued.

- Load Segment Limit instruction cannot precede PUSH/POP:

  If the instruction executed immediately after a Load Segment Limit (LSL) instruction does a stack operation, the value of (E)SP may be incorrect after the operation.

  Note: Stack operations resulting from noninstruction sources, such as exceptions or interrupts following the LSL, do not corrupt (E)SP.

  To avoid this problem, make sure that the instruction following an LSL instruction is never one that does a push to or pop off the stack. This includes PUSH, POP, RET, CALL, ENTER, and other such instructions. This can be achieved by always following an LSL instruction with a NOP instruction. Even if a forbidden instruction is used, (E)SP may be updated correctly since the problem is data-dependent and only occurs if the LSL operation succeeds (that is, sets the ZF flag).

- LSL/LAR/VERR/VERW malfunction with a NULL selector:

  An LSL, LAR, VERR, or VERW executed with a NULL selector (that is, bits 15 through 2 of the selector set to 0) operates on the descriptor at entry 0 of the Global Descriptor Table (GDT) instead of unconditionally clearing the ZF flag.

  This problem can be avoided by filling in the "NULL descriptor" (that is, the descriptor at entry 0 of the GDT) with all zeroes, which is an invalid descriptor type.

  The access to the "NULL descriptor" is made but fails since the descriptor has an invalid type. The failure is reported with ZF cleared, which is the desired behavior.

80386 Extended Protected Mode Operation

The following problems exist for operation in the Virtual 8086 mode.

- Task switch to Virtual 8086 mode does not set prefetch limit:

  The 80386 prefetch unit limit is not updated when doing a task switch to the Virtual 8086 mode. This may cause an incorrect segment limit violation to be reported if the microprocessor instruction fetches the segment limit that existed before the task switch.

  This problem can be avoided by using an IRET with the appropriate items on the stack to start the Virtual 8086 task in place of the task switch method.

- FAR jump near page boundary in Virtual 8086 mode:

  When paging is enabled in the Virtual 8086 mode, and a direct FAR jump (opcode EA) instruction is located at the end of a page (or within 16 bytes of the end), and the next page is not cached in the TLB internal to the 80386, the FAR jump instruction leaves the prefetcher limit at the "end" of the old code segment instead of setting it at the "end" of the new code segment. This can allow execution off the end of the new segment to trigger a segment limit violation, or cause a spurious GP fault if the old and new segments overlap and a prefetch crosses the old segment limit.

  There is no way to detect code "walking off" the end of a code segment. However, the spurious GP fault can be avoided by simply performing an IRET back to the instruction causing the fault. The IRET will set the prefetch limit correctly, provided the exception handler has the ability to determine a spurious GP fault from a "real" GP fault.

The following problems exist for operations with paging enabled.

- Coprocessor operands:

  To avoid having a nonstartable instruction involving math coprocessor operands in demand-paged systems, ensure the operands do not cross page boundaries. This can be accomplished by aligning math coprocessor operands in 128-byte boundaries within a segment, and aligning the start of segments on 128-byte physical boundaries.

- Page fault error code on stack is not reliable:

  When a page fault (exception 14) occurs, the three defined bits in the error code can be unreliable if a certain sequence of prefetches occurred at the same time.

  Although the page-fault error code pushed onto the page-fault handler stack is sometimes unreliable, the page-fault linear address stored in register CR2 is always correct. The page-fault handler should refer to the page-fault linear address in register CR2 to access the corresponding page table entry and thereby determine whether the page fault was due to a page-not-present condition or a usage violation.

- I/O relocated in paged systems:

  When paging is enabled (PG = 1 in CR0), accessing I/O addresses in the range hex 00001000 to hex 0000FFFF, or accessing a 80387 math coprocessor using ESC instructions (I/O addresses hex 800000F8 to hex 800000FF) can generate incorrect I/O addresses on A12 through A31 if the I/O address is the same as a memory linear address that is mapped by the TLB.

  The physical address corresponding to the memory linear address mapped by the TLB is ANDed with the I/O address, causing the I/O address to be incorrect in most cases.

  A suggested method for handling normal I/O addresses between hex 00000 and hex 0FFFF is as follows: The operating system is required to map the lowest (first) 64KB of linear address space to 16 pages, which are defined such that bits 12 through 15 of the linear and physical addresses are equal. This requires that the pages be aligned on a 64KB physical boundary (the physical address associated with the first page has address bits 15 through 0 equal to 0).

A suggested method for handling the math coprocessor I/O addresses requires that the memory page at linear address hex 80000000 always be marked "not present" so it cannot be cached in the TLB. This may be accomplished in one of the following ways:

  - Require the operating system to handle a 4KB "hole" in the linear address space at the 2GB boundary.
  - Restrict the linear address space to a 2GB maximum instead of 4GB. No segments will have a linear address above the 2GB boundary.

- Spurious page level protection fault:

  This problem only occurs when the page table and the directory entries that map the stacks for the inner levels of a task are marked as supervisor access only, and an external bus HOLD comes during the cycle that pops (E)SP off the stack during an inter-level RET or IRET.

  This problem can be avoided by marking the pages that map the inner level stacks (level 0, 1, and 2) to permit the user read access. The segmentation protection mechanism can be used to prevent user access to the linear addresses containing these stacks, if required.

ROM BIOS and Operating System Function Calls

For maximum portability, programs should perform all I/O operations through operating system function calls. In environments where the operating system does not provide the necessary programming interfaces, programs should access the hardware through ROM BIOS function calls, if permissible.

- In some environments, program interrupts are used for access to these functions. This practice removes the absolute addressing from the program. Only the interrupt number is required.
- The coprocessor detects six different exception conditions that can occur during instruction execution. If the appropriate exception mask within the coprocessor is not set, the coprocessor

sets the 'error' signal. This 'error' signal generates a hardware interrupt 13 (IRQ 13) causing the 'busy' signal to be held in the busy state. The 'busy' signal can be cleared by an 8-bit I/O Write command to address hex 00F0 with bits D0 through D7 equal to 0.

The power-on self-test code in the system ROM enables hardware IRQ 13 and sets up its vector to point to a routine in ROM. The ROM routine clears the 'busy' signal latch and then transfers control to the address pointed to by the NMI vector. This maintains code compatibility across the IBM Personal Computer and Personal System/2 product lines. The NMI handler reads the status of the coprocessor to determine if the NMI was caused by the coprocessor. If the interrupt was not caused by the coprocessor, control is passed to the original NMI handler.

- In systems using the 80286 or 80386 microprocessor, IRQ 9 is redirected to INT hex 0A (hardware IRQ 2). This ensures that hardware designed to use IRQ 2 will operate in these systems. See "Hardware Interrupts" on page 1 for more information.

- The system can mask hardware sensitivity. New devices can change the ROM BIOS to accept the same programming interface on the new device.

- In cases where BIOS provides parameter tables, such as for video or diskette, a program can substitute new parameter values by building a new copy of the table and changing the vector to point to that table. However, the program should copy the current table, using the current vector, and then modify those locations in the table that need to be changed. In this way, the program does not inadvertently change any values that should be left the same.

- The Diskette Parameters table pointed to by INT hex 1E consists of 11 parameters required for diskette operation. It is recommended that the values supplied in ROM be used. If it becomes necessary to modify any of the parameters, build another parameter block and modify the address at INT hex 1E (0:78) to point to the new block.

  The parameters were established to allow:

  - Some models of the IBM Personal Computer to operate both the 5.25-inch high capacity diskette drive (96 tracks per inch) and the 5.25-inch double-sided diskette drive (48 tracks per inch).

- Some models of the Personal System/2 to operate both the 3.5-inch 1.44M diskette drive and the 3.5-inch 720KB diskette drive.

The Gap Length Parameter is not always retrieved from the parameter block. The gap length used during diskette read, write, and verify operations is derived from within diskette BIOS. The gap length for format operations is still obtained from the parameter block.

Note: Special considerations are required for format operations. Refer to the diskette section of the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for the required details.

If a parameter block contains a head settle time parameter value of 0 milliseconds, and a write or format operation is being performed, the following minimum head settle times are enforced.

Drive Type	Head Settle Time
5.25-Inch Diskette Drives:	
Double Sided (48 TPI)	20 ms
High Capacity (96 TPI)	15 ms
3.5-Inch Diskette Drives:	
720K	20 ms
1.44M	15 ms

Figure 2. Write and Format Head Settle Time

Read and verify operations use the head settle time provided by the parameter block.

If a parameter block contains a motor-start wait parameter of less than 500 milliseconds (1 second for a Personal Computer product) for a write or format operation, diskette BIOS enforces a minimum time of 500 milliseconds (1 second for a Personal Computer product). Verify and write operations use the motor-start time provided by the parameter block.

- Programs may be designed to reside on both 5.25-inch or 3.5-inch diskettes. Since not all programs are operating-system dependent, the following procedure can be used to determine the type of media inserted into a diskette drive:

1. Verify Track 0, Head 0, Sector 1 (1 sector): This allows diskette BIOS to determine if the format of the media is a recognizable type.

   If the verify operation fails, issue the reset function (AH=0) to diskette BIOS and try the operation again. If another failure occurs, the media needs to be formatted or is defective.

2. Verify Track 0, Head 0, Sector 16 (1 sector).

   If the verify operation fails, either a 5.25-inch (48 TPI) or 3.5-inch 720KB diskette is installed. The type can be determined by verifying Track 78, Head 1, Sector 1 (1 sector). A successful verification of Track 78 indicates a 3.5-inch 720KB diskette is installed; a verification failure indicates a 5.25-inch (48 TPI) diskette is installed.

   Note: Refer to the *DOS Technical Reference* for the File Allocation Table parameters for single-sided and double-sided diskettes.

3. Read the diskette controller status in BIOS starting with address 40:42. The fifth byte defines the head that the operation ended with. If the operation ended with head 1, the diskette is a 5.25-inch high capacity (96 TPI) diskette; if the operation ended with head 0, the diskette is a 3.5-inch 1.44M diskette.

Software Compatibility

To maintain software compatibility, the interrupt polling mechanism used by IBM personal computer products is retained. Software that interfaces with the reset port for the IBM personal computer positive-edge interrupt sharing[3] does not create interference. Level-sensitive interrupt hardware allows several devices to simultaneously set a common interrupt line active (low) without interference.

Application code that deals directly with the interrupt controller may try to reset the controller to the positive edge-sensitive mode when

[3] Hex address 02FX or 06FX, where X is the interrupt level.

relinquishing control. The interrupt control circuitry of the system board prevents setting the controller to the edge-sensitive mode by blocking positive edge-sensitive commands to the interrupt controllers.

Multitasking Provisions

The BIOS contains a feature to assist multitasking implementation. "Hooks" are provided for a multitasking dispatcher. Whenever a busy (wait) loop occurs in the BIOS, a hook is provided for the program to break out of the loop. Also, whenever BIOS services an interrupt, a corresponding wait loop is exited, and another hook is provided. Thus a program can be written that employs the bulk of the device driver code. The following is valid only in the Real Address mode and must be taken by the code to allow this support.

- The program is responsible for the serialization of access to the device driver. The BIOS code is not reentrant.
- The program is responsible for matching corresponding Wait and Post calls.

Warning: 32-bit operations to the video subsystem can cause a diskette overrun in the 1.44M mode because data width conversions may require more than 12 microseconds. If an overrun occurs, BIOS returns an error code and the operation should be retried.

Interfaces

There are four interfaces to be used by the multitasking dispatcher:

Startup: First, the startup code hooks interrupt hex 15. The dispatcher is responsible to check for function codes of AH= hex 90 or 91. The "Wait" and "Post" sections describe these codes. The dispatcher must pass all other functions to the previous user of interrupt hex 15. This can be done by a JMP or a CALL. If the function code is hex 90 or 91, the dispatcher should do the appropriate processing and return by the IRET instruction.

Serialization: It is up to the multitasking system to ensure that the device driver code is used serially. Multiple entries into the code can result in serious errors.

Wait: Whenever the BIOS is about to enter a busy loop, it first issues an interrupt hex 15 with a function code of hex 90 in AH. This signals a wait condition. At this point, the dispatcher should save the task status and dispatch another task. This allows overlapped execution of tasks when the hardware is busy. The following is an outline of the code that has been added to the BIOS to perform this function.

```
MOV AX, 90XXH        ; wait code in AH and
                     ; type code in AL
INT 15H              ; issue call
JC  TIMEOUT          ; optional: for time-out or
                     ; if carry is set, time-out
                     ; occurred
NORMAL TIMEOUT LOGIC ; normal time-out
```

Post: Whenever the BIOS has set an interrupt flag for a corresponding busy loop, an interrupt hex 15 occurs with a function code of hex 91 in AH. This signals a Post condition. At this point, the dispatcher should set the task status to "ready to run" and return to the interrupt routine. The following is an outline of the code added to BIOS that performs this function.

```
MOV AX, 91XXH        ; post code AH and
                     ; type code AL
INT 15H              ; issue call
```

Classes

The following types of wait loops are supported:

- The class for hex 0 to 7F is serially reusable. This means that for the devices that use these codes, access to the BIOS must be restricted to only one task at a time.
- The class for hex 80 to BF is reentrant. There is no restriction on the number of tasks that can access the device.
- The class for hex C0 to FF is noninterrupt. There is no corresponding interrupt for the wait loop. Therefore, it is the responsibility of the dispatcher to determine what satisfies this condition to exit from the loop.

Function Code Classes

Type Code (AL)	Description
00H->7FH	Serially reusable devices; the operating system must serialize access.
80H->0BFH	Reentrant devices; ES:BX is used to distinguish different calls (multiple I/O calls are allowed simultaneously).
0C0H->0FFH	Wait-only calls; there is no complementary Post for these waits--these are time-out only. Times are function-number dependent.

Function Code Assignments: The following are specific assignments for the Personal System/2 BIOS. Times are approximate. They are grouped according to the classes described under “Function Code Classes.”

Type Code (AL)	Time Out	Description
00H	Yes (12 seconds)	Fixed Disk
01H	Yes (2 seconds)	Diskette
02H	No	Keyboard
0FCH	Yes	Fixed Disk Reset
0FDH	Yes (500-ms Read/Write)	Diskette Motor Start
0FEH	Yes (20 seconds)	Printer

Figure 3. Functional Code Assignments

The asynchronous support has been omitted. The serial and parallel controllers generate interrupts, but BIOS does not support them in the interrupt mode. Therefore, the support should be included in the multitasking system code if that device is to be supported.

Time-Outs

To support time-outs properly, the multitasking dispatcher must be aware of time. If a device enters a busy loop, it generally should remain there for a specific amount of time before indicating an error. The dispatcher should return to the BIOS wait loop with the carry bit set if a time-out occurs.

Machine-Sensitive Programs

Programs can select machine-specific features, but they must first identify the machine and model type. IBM has defined methods for uniquely determining the specific machine type. The location of the machine model bytes can be found through interrupt 15 function code (AH) = hex C0. See the *IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference* for a listing of model bytes for IBM Personal Computer and Personal System/2 products.

Math Coprocessor Compatibility

IBM systems use three math coprocessors: the 8088- and 8086-based systems use the 8087, the 80286-based systems use the 80287, and the 80386-based systems use the 80387. In the Real Address mode and Virtual 8086 mode, the 80386/80387 is upward object-code compatible with software for the 8086/8087 and 80286/80287 Real-Address mode systems; in the Protected mode, the 80386/80387 is upward object-code compatible with software for the 80286/80287 Protected-mode systems. However, if a math coprocessor instruction other than FINIT, FSTSW, or FSTCW is executed by an 80386-based system without an 80387 present, the 80386 waits indefinitely for a response from the 80387. This causes the system to stop processing without providing an error indication. To prevent this problem, software should check for the presence of the 80387 before executing math coprocessor instructions. The BIOS equipment function should be used when possible as the method for detecting the presence of the math coprocessor.

The only other differences of operation that may appear when 8086/8087 programs are ported to a Protected-mode 80386/80387 system (*not* using the Virtual 8086 mode), are in the format of operands for the administration instructions FLDENV, FSTEN, FRSTOR, and FSAVE. These instructions are normally used only by exception handlers and operating systems, not by application programs.

Operating Modes	Software Written for:		
	8087 Real	80287 Real	80287 Protected
8087 Real Mode	Yes	Yes*	No
80287 Real Mode	Yes*	Yes	No
80387 Real Mode	Yes***	Yes**	No
80387 8086 Virtual Mode	Yes***	Yes**	No
80287 Protected Mode	No	Yes**	Yes
80387 Protected Mode	No	No	Yes**

* See "8087 to 80287 Compatibility."
** See "80287 to 80387 Compatibility."
***See "8087 to 80287 Compatibility" and "80287 to 80387 Compatibility."

Figure 4. Math Coprocessor Software Compatibility

Many changes have been designed into the 80387 to directly support the IEEE standards in hardware. These changes result in increased performance by eliminating the need for software that supports the IEEE standard.

80287 to 80387 Compatibility

The following summarizes the differences between the 80387 and 80287 Math Coprocessors, and provides details showing how 80287 software can be ported to the 80387 Math Coprocessor:

Note: Any migration from 8087 directly to the 80387 must also take into account the differences between the 8087 and the 80287. This information is provided on page 25.

- The 80387 supports only affine closure for infinity arithmetic, not projective closure.
- Operands for FSCALE and FPATAN are no longer restricted in range (except ±∞); F2XM1 and FPTAN accept a wider range of operands.
- Rounding control is in effect for FLD *constant*.
- Software cannot change entries of the tag word to values (other than empty) that differ from actual register contents.
- In conformance with the IEEE standard, the 80387 does not support special data formats pseudozero, pseudo-NaN, pseudoinfinity, and unnormal.

Exceptions

When the overflow or underflow exception is masked, the only difference from the 80287 is in rounding when overflow or underflow occurs. The 80387 produces results that are consistent with the rounding mode.

For exceptions that are not masked, a number of differences exist due to the IEEE standard and to functional improvements to the architecture of the 80387:

- There are fewer invalid-operation exceptions due to denormal operands, because the instructions FSQRT, FDIV, FPREM, and conversions to BCD or to integer normalize denormal operands before proceeding.
- The FSQRT, FBSTP, and FPREM instructions may cause underflow, because they support denormal operands.
- The denormal exception can occur during the transcendental instructions and the FXTRACT instruction.
- The denormal exception no longer takes precedence over all other exceptions.
- When the operand is zero, the FXTRACT instruction reports a zero-divide exception and leaves $-\infty$ in ST(1).
- The status word has a new bit (SF) that signals when invalid-operation exceptions are due to stack underflow or overflow.
- FLD *extended precision* no longer reports denormal exceptions, because the instruction is not numeric.
- FLD *single/double precision* when the operand is denormal converts the number to extended precision and signals the denormalized operand exception. When loading a signaling NaN, FLD *single/double precision* signals an invalid-operation exception.
- The 80387 only generates quiet NaNs (as on the 80287); however, the 80387 distinguishes between quiet NaNs and signaling NaNs. Signaling NaNs trigger exceptions when they are used as operands; quiet NaNs do not (except for FCOM, FIST, and FBSTP, which also raise IE for quiet NaNs).

- Most 80387 numeric instructions are automatically synchronized by the 80386. No explicit Wait instructions are required for these instructions. To maintain compatibility with systems using the 8087, an explicit Wait is required before each numeric instruction.
- The FLDENV and FRSTOR instructions should be followed by an explicit Wait when used in the 80387 environment. An explicit Wait is not required after these instructions in the 80287 environment.
- The 80287 FSETPM (set Protected mode) instruction performs no useful purpose in the 80387 environment; if encountered, it is ignored.
- The format of the FSAVE and FSTENV instructions is determined by the current mode of the 80386; the Real Address mode format is used when the 80386 is in the Real Address mode, and the Protected mode format is used when the 80386 is in the Protected mode.
- The following applies only to the B1 stepping level 80386: An interrupt 9 does not occur for an operand outside a segment size; an interrupt 13 occurs.

8087 to 80287 Compatibility

The 80287 operating in the Real Address mode can execute 8087 software without major modifications. However, because of differences in the handling of numeric exceptions by the 80287 and the 8087, exception-handling routines *may* need to be changed.

The following summarizes the differences between the 80287 and 8087 Math Coprocessors, and provides details showing how 8087 software can be ported to the 80287 Math Coprocessor.

- The 8087 instructions FENI/FNENI and FDISI/FNDISI perform no useful function in the 80287 environment. If the 80287 encounters one of these opcodes in its instruction stream, the instruction is effectively ignored; none of the 80287 internal states are updated. While 8086 code containing these instructions may be executed on an 80287, it is unlikely that the exception-handling routines containing these instructions will be completely portable to the 80287.

- The ESC instruction address saved in the 80287 includes any leading prefixes before the ESC opcode. The corresponding address saved in the 8087 does not include leading prefixes.

- In the Protected mode, the format of the 80287 saved instruction and address pointers is different from the format of the 8087. The instruction opcode is not saved in the Protected mode; exception handlers have to retrieve the opcode from memory if needed.

- Interrupt 7 occurs in the 80286 when executing ESC instructions with either TS (task switched) or EM (emulation) of the 80286 MSW set (TS = 1 or EM = 1). If TS is set, then a Wait instruction also causes interrupt 7. An exception handler should be included in 80286 code to handle these exceptions.

- Interrupt 9 occurs if the second or subsequent words of a floating-point operand fall outside a segment size. Interrupt 13 occurs if the starting address of a numeric operand falls outside a segment size. An exception handler should be included in the 80286 code to report these programming errors.

- Most 80287 numeric instructions are automatically synchronized by the 80286. The 80286 automatically tests the 'busy' signal from the 80287 to ensure that the 80287 has completed its previous instruction before executing the next ESC instruction. Explicit Wait instructions are not required to ensure this synchronization. An 8087 used with 8086 and 8088 system microprocessors requires explicit Waits before each numeric instruction to ensure synchronization. Although 8086 software having explicit Wait instructions executes perfectly on the 80286 without reassembly, these Wait instructions are unnecessary.

  The processor control instructions for the 80287 may be coded using either a WAIT or No-WAIT form of the mnemonic. The WAIT forms of these instructions cause the assembler to precede the ESC instruction with a microprocessor Wait instruction.

Diskette Drives and Controller

The following figure shows the read, write, and format capabilities for each type of diskette drive.

Diskette Drive Type	**160/180KB Mode**	**320/360KB Mode**	**1.44MB Mode**	**720KB Mode**
5.25-Inch Diskette Drive:				
Single Sided (48 TPI)	R W F	---	---	---
Double Sided (48 TPI)	R W F	R W F	---	---
3.5-Inch Diskette Drive:				
720KB Drive	---	---	---	R W F
1.44MB Drive	---	---	R W F	R W F
R-Read W-Write F-Format				

Figure 5. Diskette Drive Read, Write, and Format Capabilities

Notes:

1. 5.25-inch diskettes designed for the 1.2M mode cannot be used in either a 160/180KB or a 320/360KB diskette drive.
2. 3.5-inch diskettes designed for the 1.44M mode cannot be used in a 720KB diskette drive.

Warning: 32-bit operations to the video subsystem can cause a diskette overrun in the 1.44M mode because data width conversions may require more than 12 microseconds. If an overrun occurs, BIOS returns an error code and the operation should be retried.

Copy Protection

The following methods of copy protection may not work on systems using the 3.5-inch 1.44M diskette drive.

- Bypassing BIOS Routines:
  - Data Transfer Rate: BIOS selects the proper data transfer rate for the media being used.
  - Diskette Parameters Table: Copy protection, which creates its own Diskette Parameters table, may not work on these drives.

- Diskette Drive Controls:
  - Rotational Speed: The time between two events on a diskette is a function of the controller.
  - Access Time: Diskette BIOS routines must set the track-to-track access time for the different types of media used in the drives.
  - Diskette Change Signal: Copy protection may not be able to reset this signal.
- Write Current Control—Copy protection that uses write current control will not work because the controller selects the proper write current for the media being used.

Detailed information about specific diskette drives is available in separate technical references.

Fixed Disk Drives and Controller

Reading from and writing to the fixed disk drive is initiated in the same way as with IBM Personal Computer products; however, new functions are supported. Detailed information about specific fixed disk drives and fixed disk adapters is available in system-specific technical references.

Index

A

application guidelines 1
assembler language programming considerations 3

B

back-to-back I/O commands 5
BASIC 3
busy loop 20
bypassing BIOS 27

C

classes, wait loop 20
COBOL 3
condition, wait 20
copy protection 27

D

diskette change signal 28
diskette compatibility 27
diskette write current 28
divide error exception 4

E

edge-triggered interrupts 1
exception, divide error 4

F

fixed disk controller 28
FORTRAN 3
function calls, operating system 15
function calls, ROM BIOS 15
function codes, multitasking 21

H

hardware interrupts 1
high-level language considerations 3

I

I/O commands and STI instructions 5
IBM C 3
interfaces, multitasking 19
interrupt, single step 4
IRQ 2 16
IRQ 9 16

L

level-sensitive interrupts 1
loop, busy 20

M

machine model bytes 22
machine-sensitive programs 22
math coprocessor compatibility 22
model bytes 22
multiple lockout instructions 5

O

opcodes 4
operating system 3
operating system function calls 15

P
paging 13
parameter tables, BIOS 16
Pascal 3
polling mechanism 18
post 20
provisions, multitasking 19
PUSH SP 4

S
serialization, multitasking 19
shift counts 4
single step interrupt 4
software interrupts 3
startup, multitasking 19
STI instruction 5
system compatibility 1

T
time-outs 21

W
wait condition 20
wait loop classes 20
write current, diskette 28

Numerics
80286 anomalies 8
80386 anomalies 8

Bibliography

Microprocessor and Peripheral Handbook. INTEL Corporation, *210844.001*

Introduction to the iAPX 286. INTEL Corporation, *210308.001*

iAPX 286 Operating Systems Writer's Guide. INTEL Corporation, *121960.001*

iAPX 286 Programmer's Reference Manual. INTEL Corporation, *210498.001*

iAPX 286 Hardware Reference Manual. INTEL Corporation, *210760.001*

Numeric Processor Extension Data Sheet. INTEL Corporation, *210920*

80287 Support Library Reference Manual. INTEL Corporation, *122129*

80386 Hardware Reference Manual. INTEL Corporation, *231732.001*

Introduction to the 80386. INTEL Corporation, *231252.001*

80386 Programmer's Reference Manual. INTEL Corporation, *230985.001*

80386 System Software Writer's Guide. INTEL Corporation, *231499*

National Semiconductor Corporation, NS16550

Motorola Microprocessor's Data Manual. Motorola Inc. *Series B*

Notes:

Printed in the
United States of America

References in this
Publication to IBM
products or services do not
imply that IBM intends
to make them available
outside the United States.

IBM

Personal System/2®
Hardware Interface
Technical Reference

68X2330
S68X-2330-00