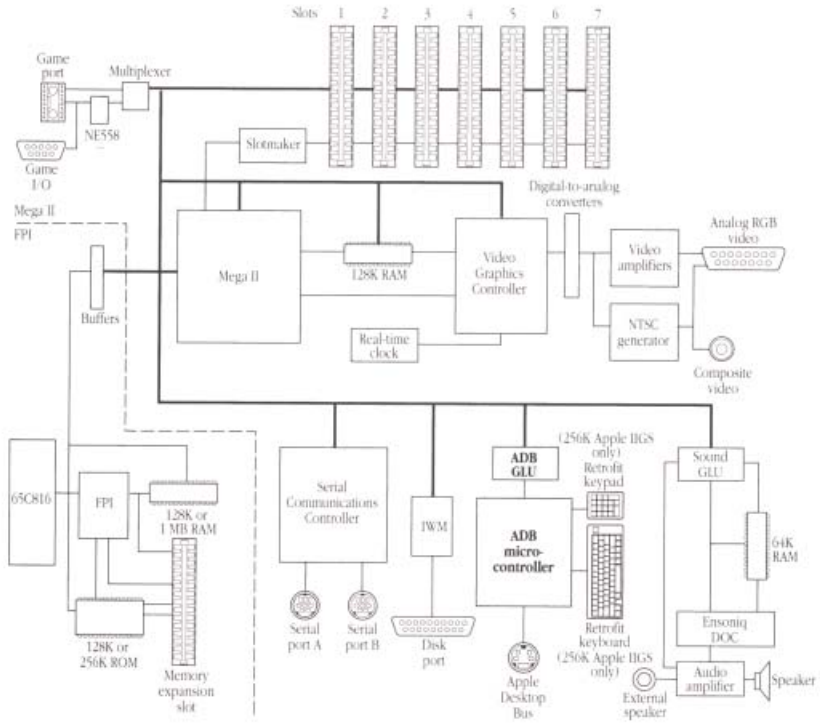
Chapter 6 The Apple Desktop Bus

The Apple Desktop Bus (ADB) is a method for connecting input devices (such as a keyboard or a mouse) with the Apple IIGS computer. The Apple Desktop Bus consists of the ADB GLU, the ADB microcontroller chip, and the Apple Desktop Bus cables. Figure 6-1 shows the relationship of the ADB components in the Apple IIGS computer.

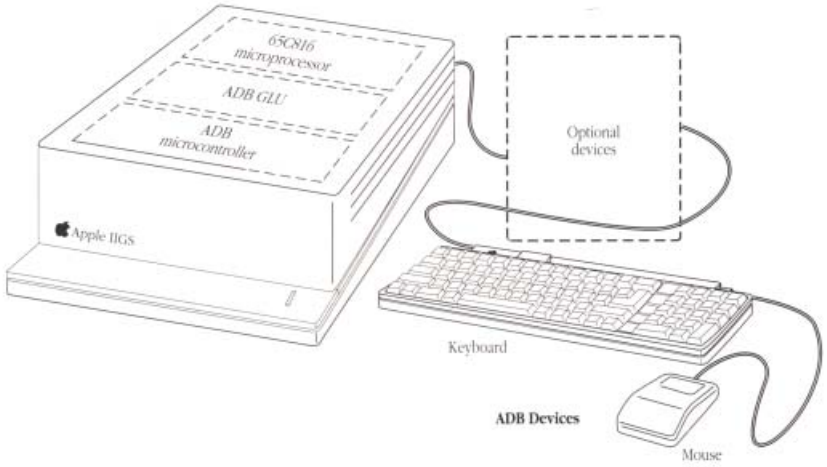
Figure 6-1 ADB components in the Apple IIGS



The ADB microcontroller controls devices on the bus by receiving commands from the 65C816 microprocessor, and then sending appropriate ADB commands to and receiving data from the input devices on the bus. Microcontroller commands (those received from the 65C816) are located in ROM. Figure 6-2 shows the relationship of ADB components in the host computer to the bus devices (keyboards, mouse devices, and so on).

For the remainder of this chapter, the computer will be referred to as the host, and the input devices (for example, a keyboard or a mouse) connected to the bus as devices.

■ **Figure 6-2** ADB components



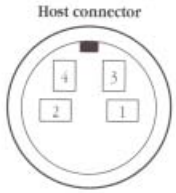
◆ *Note:* To keep compatibility with future Apple II products using the ADB, use the Apple Desktop Bus Tool Set in ROM. Directly accessing some of the ADB registers may cause the system to crash. (For more information on the ADB Tool Set, refer to the *Apple IIGS Toolbox Reference.*)

The input bus

All input devices share the input bus with the host. This bus consists of a 4-wire cable and uses 4-pin mini-DIN jacks at the host and at each device. Figure 6-3 shows the pin configuration of the ADB connectors and Table 6-1 lists the description of each pin. ADB devices may use the +5-volt power supplied by the bus, but must not draw more than 500 mA

total for all devices. All devices are connected in **parallel,** using the signal, power, and ground wires. Cables should be no longer than 5 meters, and total cable capacitance should not exceed 100 picoFarads per meter.

■ **Figure 6-3** Mini-DIN connector pin configuration used in the ADB



■ **Table 6-1** Pin assignments of the ADB connectors

Pin	Description
1	Data
2	Reserved
3	+5-volt **power supply** at 500 mA for all devices
4	Ground

The keyboard

The keyboard uses the Apple Desktop Bus to communicate with the processor. All input devices are connected to the ADB and are controlled by the keyboard microcontroller. This controller supports reading of the keyboard by standard Apple II application programs.

The Apple IIGS keyboard has automatic repeat, which means that if you press any key longer than you would during normal typing, the character code for that key will be sent continuously until you release the key. The speed at which the key repeats, and the delay before it repeats, may be set from the Control Panel. (See the section "Modifier Key Register," later in this chapter, for more information.)

Any number of modifier keys may be held down and an additional key pressed will be recognized. This is called *n-key rollover*. The alphanumeric keys do not have this ability. Any two alphanumeric keys pressed simultaneously will be recognized. A third key pressed will not be recognized. This is called *2-key lockout*.

Apple IIGS computers manufactured for sale outside the United States have slightly different standard keyboard arrangements. The different keyboards are shown in Appendix B.

ADB and the upgraded Apple IIe

The Apple IIGS can be used with a standard Apple IIe keyboard when an Apple IIe is upgraded with an Apple IIGS logic board. The Apple IIe keyboard is then used as the primary input device, rather than the ADB keyboard. The Apple IIe keyboard is read just like the standard keyboard in an Apple IIe, and is not subject to the Apple Desktop Bus protocols.

Reading the keyboard

The ADB microcontroller generates all 128 ASCII codes, so all the special character codes in the ASCII character set are available. Application programs obtain character codes from the keyboard by reading a byte from the keyboard data location shown in Table 6-2 (the Keyboard Data register).

■ **Table 6-2** Keyboard Data locations

Location		
Hex	**Dec**	**Description**
$C000	49152	Keyboard Data register and strobe
$C010	49168	Any-key-down flag and clear-strobe switch

Your programs can get the code for the last key pressed by reading the Keyboard Data register located at $C000. Table 6-2 gives this location in two different forms: The hexadecimal value, indicated by a preceding dollar sign ($), is used in assembly language; the decimal value is used in Applesoft BASIC.

The low-order seven bits of the byte at the keyboard data location contain the character code; the high-order bit of this byte is the strobe bit, described below.

Your program can find out whether any key is down, except the Reset, Control, Shift, Caps Lock, Command, and Option keys, by reading from location 49152 ($C000). The high-order bit (bit 7) of the byte you read at this location is called *any-key-down;* it is 1 if a key is down and 0 if no key is down. The value of this bit is 128; if a BASIC program gets this information with a PEEK, the value is 128 or greater if any key is down, and less than 128 if no key is down.

The strobe bit is the high-order bit of the Keyboard Data register. After any key has been pressed, the strobe bit is high. It remains high until you reset it by reading or writing at location $C010. This location is a combination flag and soft switch; the flag tells whether any key is down, and the switch clears the strobe bit. In this case, it doesn't matter whether the program reads or writes, and it doesn't matter what data the program writes: The only action that occurs is the resetting of the keyboard strobe. (See the *Apple IIGS Firmware Reference* for information on firmware for reading the keyboard.)

◆ *Note:* Any time you read the any-key-down flag, you also clear the keyboard strobe. If your program needs to read both the flag and the strobe, it must read the strobe bit first.

After the keyboard strobe has been cleared, it remains low until another key is pressed. Even after you have cleared the strobe, you can still read the character code at the keyboard location. The data byte has a different value, because the high-order bit is no longer set, but the ASCII code in the seven low-order bits is the same until another key is pressed. Appendix D contains the ASCII codes for the keys on the keyboard.

There are several special function keys that do not generate ASCII codes. For example, pressing the Control, Shift, or Caps Lock key directly alters the character codes produced by the other keys. In the Apple IIGS, you can determine the state of these modifier keys by reading the modifier key register within the ADB microcontroller, described later in this chapter.

The Control-Command-Reset key combination is different from all other key combinations on the ADB keyboard in that it generates a special key code. When the ADB microcontroller detects the reset key combination, the program currently running in memory is halted and the system asserts the RESET line and restarts the computer. This restarting process is called the *reset routine.* (To read about the reset routine, see the *Apple IIGS Firmware Reference.*)

The ADB microcontroller

The ADB microcontroller is an intelligent controller IC that oversees the Apple Desktop Bus. The M50740 microcontroller uses a superset of the 6502 instruction set, and contains 96 bytes of RAM and 3K of ROM. The ADB microcontroller operates asynchronously, transmitting commands and data to and receiving data from the bus devices. To ensure compatibility with future versions of ADB, use the ADB commands in the ROM toolbox to communicate with the ADB. To find out how to use the toolbox in the system ROM, see the *Apple IIGS Toolbox Reference*.

The ADB GLU

The ADB General Logic Unit (GLU) works together with the ADB microcontroller to form an intelligent input-device interface. The ADB GLU, located on the main logic board, uses two independent data buses that serve as a communications interface between the ADB microcontroller and the system bus. This interface is accomplished by using multiple internal read/write registers to store keyboard data, key modifiers, mouse X and Y coordinates, command data, and status information.

The ADB GLU registers

The ADB GLU contains five data and control registers. These registers are used for storing keyboard data and commands, key modifiers, mouse X and Y coordinates, and status information. The registers are

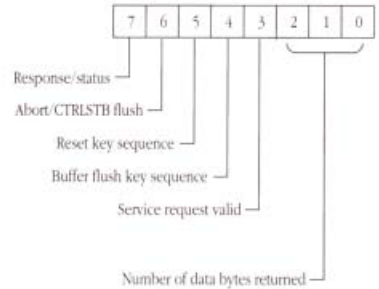
- ADB Command/Data register ($C026)
- Keyboard Data register ($C000)
- Modifier Key register ($C025)
- Mouse Data register ($C024)
- ADB Status register ($C027)

All registers except the status registers have a status flag that is set to 1 when the register is written to, and cleared to 0 when the register is read. Each of the data registers also has an interrupt flag that generates system interrupts, if interrupts are enabled. These status and interrupt flags are located in the status register. These registers are described in the following sections.

ADB Command/Data register

The ADB Command/Data register is a dual-function register used to communicate with ADB devices. To send a command to a device on the bus, write the command byte to this register at address $C026. To check the status of an ADB device, read this register at the same address. Figure 6-4 shows the format of the ADB Command/Data register when it is read. Table 6-3 gives a description of each bit.

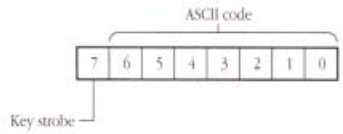
- **Figure 6-4** ADB Command/Data register at $C026



Keyboard Data register

The Keyboard Data register contains the ASCII value of the last key pressed on the keyboard. The high bit is set when a new key has been pressed. Figure 6-5 shows the format of this register. Table 6-4 gives a description of each bit.

- **Figure 6-5** Keyboard Data register at $C000



■ **Table 6-3** Bits in the ADB Command/Data register

Bit	Value	Description
7	1	When this bit is 1, the ADB microcontroller has received a response from an ADB device previously addressed.
	0	No response.
6	1	When this bit is 1, and only this bit in the register is 1, the ADB microcontroller has encountered an error and has reset itself. When this bit is 1 and bit 4 is also 1, the ADB microcontroller should clear the key strobe (bit 7 in the Keyboard Data register at $C000).
	0	–
5	1	When this bit is 1, the Control, Command, and Reset keys have been pressed simultaneously. This condition is usually used to initiate a cold **start up.**
	0	Reset key sequence has not been pressed.
4	1	When this bit is 1, the Control, Command, and **Delete keys** have been pressed simultaneously. This condition will result in the ADB microcontroller's flushing all internally buffered commands.
	0	Buffer flush key sequence has not been pressed.
3	1	When this bit is 1, a valid service request is pending. The ADB microcontroller will then poll the ADB devices and determine which has initiated the request.
	0	No service request pending.
2–0	–	The number of data bytes to be returned from the device is listed here.

■ **Table 6-4** Bits in the Keyboard Data register

Bit	Value	Description
7	–	This bit is 1 when a key has been pressed, and indicates that the ASCII value in bits 6 through 0 are valid. This bit must be cleared after reading the data by reading or writing to any address from $C010 through $C01F.
6–0	–	ASCII data from the keyboard.

Modifier Key register

The Modifier Key register contains bits that reflect the status of the modifier keys. These keys include the standard Shift, Control, Command, and Caps Lock keys, as well as keys on the numeric keypad. Figure 6-6 shows the format of this register. Table 6-5 gives a description of each bit.

■ **Table 6-5** Bits in the Modifier Key register

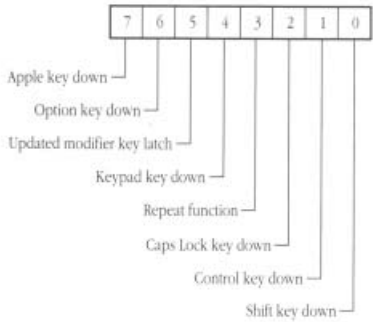
Bit	Value	Description
7	1	When this bit is 1, the Command key has been pressed.
	0	When this bit is 0, the Command key has not been pressed.
6	1	When this bit is 1, the Option key has been pressed.
	0	When this bit is 0, the Option key has not been pressed.
5	1	When this bit is 1, the modifier key latch has been updated, but no key has been pressed.
	0	–
4	1	When this bit is 1, a numeric keypad key has been pressed.
	0	When this bit is 0, a numeric keypad key has not been pressed.
3	1	When this bit is 1, a key is being held down.
	0	When this bit is 0, no key is being held down.
2	1	When this bit is 1, the Caps Lock key has been pressed.
	0	When this bit is 0, the Caps Lock key has not been pressed.
1	1	When this bit is 1, the Control key has been pressed.
	0	When this bit is 0, the Control key has not been pressed.
0	1	When this bit is 1, the Shift key has been pressed.
	0	When this bit is 0, the Shift key has not been pressed.

Mouse Data register

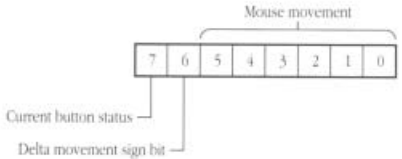
The ADB mouse, when moved, generates movement data that are transmitted to the host. These data, along with the mouse button status, are available in the Mouse Data register. Figure 6-7 shows the format of this register. Table 6-6 gives a description of each bit.

◆ *Note:* Read this register only twice in succession. The first read returns Y-coordinate data, and the second read returns X-coordinate data. Reading this register an odd number of times will result in a random movement of the cursor.

■ **Figure 6-6** Modifier Key register at $C025



■ **Figure 6-7** Mouse Data register at $C024



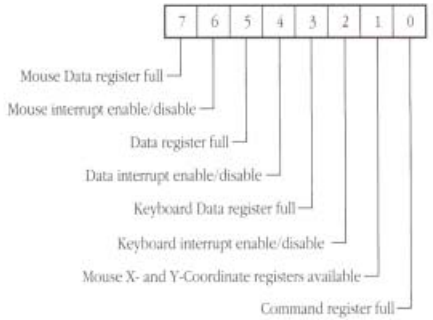
■ **Table 6-6** Bits in the Mouse Data register

Bit	Value	Description
7	1	Current mouse status: When this bit is 1, the mouse button is up.
	0	When this bit is 0, the mouse button is down.
6	1	Delta sign bit: if this bit is 1, the delta value is negative.
	0	If this bit is 0, the delta value is positive.
5–0	–	The relative mouse movement data are returned here.

ADB Status register

The ADB Status register, located at $C027, contains flags that relate to mouse and keyboard data and status. Figure 6-8 shows the format of the ADB Status register. Table 6-7 gives a description of each bit.

■ **Figure 6-8** ADB Status register at $C027



■ **Table 6-7** Bits in the ADB Status register

Bit	Value	Description
7	1	When this bit is 1, the Mouse Data register at $C024 is full (read-only bit)
	0	When this bit is 0, the Mouse Data register is empty.
6	1	When this bit is 1, the mouse interrupt is enabled (read/write bit), and an interrupt is generated when the Mouse Data register contains valid data.
	0	When this bit is 0, the mouse interrupt is disabled.
5	1	When this bit is 1, the Command/Data register contains valid data (read-only bit)
	0	When this bit is 0, the Command/Data register contains no valid data.
4	1	When this bit is 1, the Command/Data interrupt is enabled (read/write bit), and an interrupt is generated when the Command/Data register contains valid data.

■ **Table 6-7** Bits in the ADB Status register (Continued)

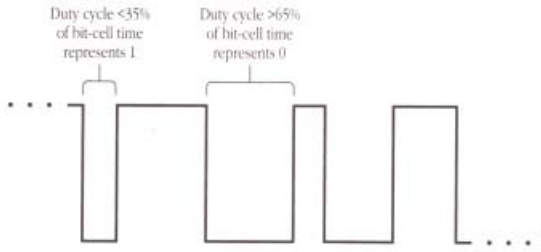
Bit	Value	Description
	0	When this bit is 0, the data interrupt is disabled.
3	1	When this bit is 1, the Keyboard Data register is full (read-only bit). This bit is set when the Keyboard Data register is written to.
	0	When this bit is 0, the Keyboard Data register is empty. This bit is cleared when the Keyboard Data register is read or the ADB Status register is read.
2	1	When this bit is 1, the keyboard data interrupt is enabled (read/write bit), and an interrupt is generated when the Keyboard Data register contains valid data
	0	When this bit is 0, the keyboard data interrupt is disabled.
1	1	When this bit is a 1, mouse X-coordinate data is available in the Mouse Data register (read-only bit)
	0	When this bit is a 0, mouse Y-coordinate data is available in the Mouse Data register.
0	1	Command register full (read-only bit). This bit is set when the Command/Data register is written to.
	0	Command register empty. This bit is cleared when the Command/Data register is read.

Bus communication

The host communicates with the devices on the bus by sending commands and/or data to a device. A device may respond to commands by sending data back to the host. This form of communication uses strings of bits, each making up a packet. A data transfer or transaction consists of a complete communication between the host and a device; for example, it may be a command packet sent by the host to request data from a device, followed by a data packet sent from the device to the host. Depending upon the type of command sent by the host, the device will send two to eight data bytes back.

Figure 6-9 shows how duty-cycle modulation represents bits on the bus. A low period of less than 35 percent of the bit-cell time is interpreted as a 1. A low period of greater than 65 percent of the bit-cell time is interpreted as a 0.

■ **Figure 6-9** Bit representation via duty-cycle modulation



Commands and global signals

Communications on the bus are of two types: Commands that are sent by the host to a specific device, and global signals that instruct all devices to perform a predefined function. The four commands in the command set are

- Listen
- Talk
- Device Reset
- Flush

Certain signals, however, are neither commands nor data transactions. These are special signals that the host or a device uses to broadcast status globally to all devices on the bus. There are four signals in the global signal group:

- Attention
- Sync
- Global Reset
- Service Request

The following paragraphs describe each of these device commands and global signals.

Transactions

A transaction is a bus communication between the host and a device, or vice versa. A command (Listen, Talk, Device Reset, Flush) initiates a transaction. A transaction consists

of a command packet sent by the host followed by a data packet of several bytes by either the host or a device. A command packet consists of

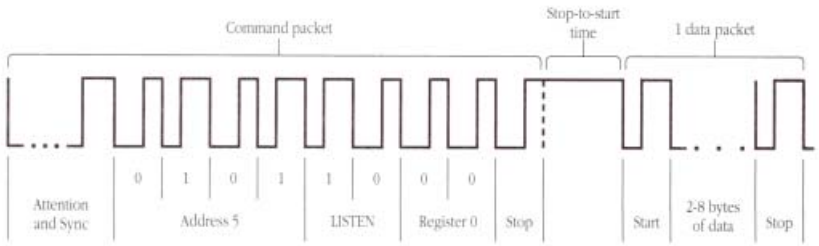
- an attention/sync signal
- one command byte
- one stop bit

A data packet consists of

- a start bit
- two to eight (8-bit) data bytes
- one stop bit

Figure 6-10 shows a typical transaction on the Apple Desktop Bus, consisting of a command packet followed by a data packet.

- **Figure 6-10** A typical transaction



To indicate the end of the command packet, the command ends with a stop bit after the last command bit-cell. Then the transaction is complete and the host releases its control of the bus. (The bus is always floating in a high state until a device or the host initiates a transaction.) The identical scheme is used for the data packet.

Note that the stop-bit–to–start-bit time is critical; the host requires this minimum turnaround delay to allow for internal overhead. Table 6-8 lists the timing maximum and minimum parameters for ADB signals.

- **Table 6-8** ADB timing specifications

Parameter	Minimum	Maximum	Unit
Bit-cell time	70	130	microseconds
"0" low time	60	70	percent of bit-cell time
"1" low time	30	40	percent of bit-cell time
Attention	560	1040	microseconds
Global Reset	2.8	5.2	milliseconds
Sync	60	70	percent of bit-cell time
Service request	140	260	microseconds
Stop bit to start bit time	140	260	microseconds

Commands

A command is sent by the host to one specific device address. Only the host can send commands. There are four commands: The Talk command is used to acquire data from a device; the Listen command is used to place data in a device register or to get a device to perform some new function; the Device Reset command reinitializes the device; and the Flush command is device specific.

A command is an 8-bit word that has a specific syntax (as shown in Table 6-9):

- A 4-bit field that specifies the address of the desired device. (The addresses range from 0 through 15 [address bits 3 through 0].)
- A 4-bit command and register address code.

◆ *Note:* To allow for future expansion of the command structure, Apple Computer, Inc. has reserved a group of instructions that are currently treated as no-ops (no operation performed). Use of commands not listed will result in possible incompatibility with future Apple products.

Talk

The Talk command is a request of the device to transmit the contents of one of the device's registers (0 through 3). When the host addresses a device to Talk, the device must respond with data before the host times out (does not receive data within the specified time). The selected device performs its data transaction and releases the bus.

■ **Table 6-9** Command byte syntax

	Device address				Command code		Register code		
Bit	7	6	5	4	3	2	1	0	Command
	x	x	x	x	0	0	0	0	Send Reset
	A_3	A_2	A_1	A_0	0	0	0	1	Flush
	x	x	x	x	0	0	1	0	Reserved
	x	x	x	x	0	0	1	1	Reserved
	x	x	x	x	0	1	x	x	Reserved
	A_3	A_2	A_1	A_0	1	0	r_1	r_0	Listen
	A_3	A_2	A_1	A_0	1	1	r_1	r_0	Talk

Note: x = ignored; r = register number; A3 through A0 = bits 11 through 8 of register 3.

Listen

The Listen command is a request of the device to store the data being transmitted in one of the internal registers (0 through 3). When the host addresses a device to Listen, the device receives the next data packet from the host and places it in the appropriate register. After the stop bit following the data is received, the transaction is complete and the host releases the bus. If the addressed device detects another command on the bus before it receives any data, the original transaction is immediately considered complete.

Send Reset

When a device receives a Send Reset command, it will clear all pending operations and data, and will initialize to the power-on state. The Send Reset is not device specific; it is sent to all devices on the bus simultaneously.

Flush

This is a device-specific command that will clear all pending commands from the device.

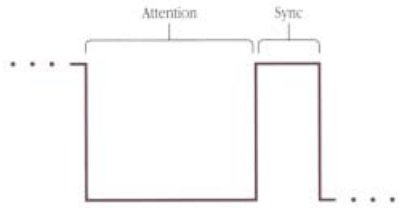
Broadcast signals

Broadcast signals are transmitted on the bus but do not address any specific device. This way, all devices receive the signals and respond simultaneously. There are four broadcast signals: Attention, Sync, Global Reset, and Service Request.

Attention and Sync

The start of every command is indicated by a long low Attention signal that the host sends on the bus. This signal is followed by a short high Sync pulse that signals the beginning of the initial bus timing. The falling edge of the sync pulse is used as a timing reference, after which the first command bit follows. Figure 6-11 shows the format of the Attention and Sync signals.

■ **Figure 6-11** Attention and Sync signals



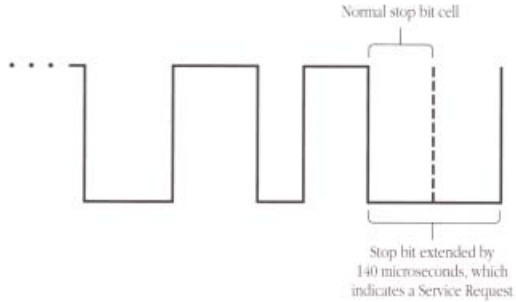
Global Reset

When the bus is held low for a minimum of 2.8 milliseconds, a Global Reset is initiated. Only the host may issue this signal, which forces all bus devices to reset. Note that the Global Reset signal differs from the Device Reset command: The Device Reset command addresses one specific device, and resets that device. The Global Reset signal is received by all devices and forces all devices to reset.

Service Request

A Service Request signal is used to inform the host that a device requires service, as, for example, when there are data to send to the host. Only a device can issue a Service Request. Following any command packet, a requesting device can signal a Service Request by holding the bus low during the low portion of the stop bit of the last command transaction. Holding the bus low in this manner lengthens the stop by a minimum of 140 microseconds beyond its normal bit-cell boundary. Figure 6-12 shows the format of the Service Request signal.

■ **Figure 6-12** Service Request



A device will signal a Service Request repeatedly until it is served. When a device has requested service (at which point the host does not know *which* device sent the request), the host will poll each of the devices by sending a Talk register 0 command (discussed later in this chapter), beginning with the last active device. Only the device that has data to send (the device that sent the Service Request) will respond to the Talk command.

When the host commands the requesting device to Talk, the device is considered served and does not send a Service Request signal again until it needs to be served again. The host can enable and disable the ability of a device to send a Service Request at any time. ADB mouse devices are prohibited by the Apple IIGS from issuing Service Requests. All other ADB devices may issue Service Requests, as long as the device has not been prohibited from sending a Service Request. (See description of register 3, later in this chapter.)

Error conditions

If the bus level remains low for a significant time period, all devices reset themselves and output a 1. If a command transaction is incomplete by staying high beyond the maximum bit-cell time, all devices ignore the command and wait for an Attention signal.

Apple Desktop Bus peripheral devices

All devices on the Apple Desktop Bus are slaves; only the host (the computer) may send commands. Devices transmit on the bus only after they have been requested by the host.

A device that receives a Talk command (and has data to send) sends the data and then releases control of the bus. If a device has been addressed but has no data to send, it does not respond and allows the host to time out (waiting for data, none arrives). The host may also send a Listen command to the addressed device followed by data in a separate packet.

Device registers

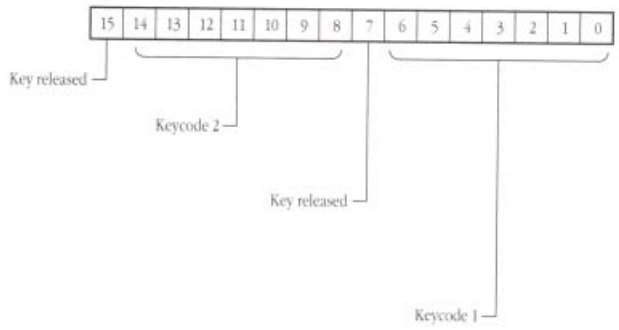
All devices have four locations to receive data. These are

- **Register 0**
  Talk: Data register, device specific
  Listen: Data register, device specific
- **Register 1**
  Talk: Data register, device specific
  Listen: Data register, device specific
- **Register 2**
  Talk: Data register, device specific
  Listen: Status or data, device specific
- **Register 3**
  Talk: Status information, including the device address handler
  Listen: Status information, including the device address handler

Register 0

Register 0 is a data register and contains data that will be sent to the host in response to a Talk register 0 command. Figures 6-13 and 6-14 show the format of register 0 as used in a keyboard and a mouse device, and Tables 6-10 and 6-11 describe each bit in these registers.

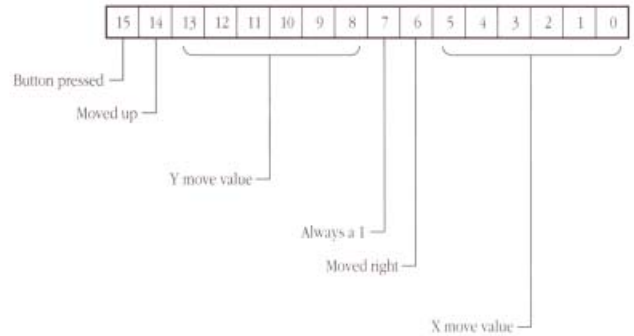
■ **Figure 6-13** Keyboard register 0



■ **Table 6-10** Bits in keyboard register 0

Bit	Value	Description
15	1	Key released indicator: When this bit is 1, the key represented by keycode 2 is up.
	0	When this bit is 0, the key represented by keycode 2 is down.
14–8	–	Keycode 2: The 7-bit ASCII value of the last key pressed.
7	1	Key released indicator: When this bit is 1, the key represented by keycode 1 is up.
	0	When this bit is 0, the key represented by keycode 1 is down.
6–0	–	Keycode 1: The 7-bit ASCII value of the last key pressed.

■ **Figure 6-14** Mouse register 0



■ **Table 6-11** Bits in mouse register 0

Bit	Value	Description
15	1	Button pressed indicator: When this bit is 1, the mouse button is up.
	0	When this bit is 0, the mouse button is down.
14	1	Movement indicator: When this bit is 1, the mouse has moved up along the Y axis.
	0	When this bit is 0, the mouse has moved down along the Y axis.
13–8	–	Y movement value: This field contains the value of the relative mouse movement along the Y (vertical) axis.
7	–	Always a 1.
6	1	Movement indicator: When this bit is 1, the mouse has moved left along the X axis.
	0	When this bit is 0, the mouse has moved right along the X axis.
5–0	–	X movement value: This field contains the value of the relative mouse movement along the X (horizontal) axis.

Register 1

Register 1, like register 0, is a data register, but it is device specific. The function of this register is not defined for use by the ADB protocol, but it may be used by the application program for any data function.

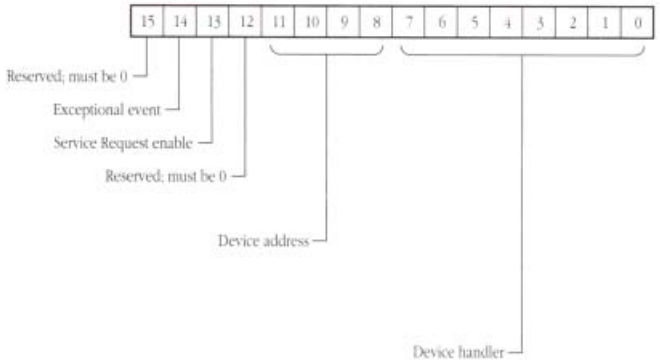
Register 2

This register is a data register and is device specific. In response to the Talk register 2 command, the device will send the contents of this register onto the bus. In response to the Listen register 2 command, the device will store the data sent to the device, as defined by the device's specification.

Register 3

This register is a status and command register that contains a handler code and the device address. The host may change the contents of this register with a Listen register 3 command. Figure 6-15 shows the format of register 3, and a description of each bit follows in Table 6-12.

■ **Figure 6-15** Device register 3



■ **Table 6-12** Bits in device register 3

Bit	Value	Description
15	–	Reserved; must be 0.
14	1	Exceptional event, device specific. In the ADB keyboard, this bit indicates that the Reset key has been pressed.
	0	No exceptional event has occurred.
13	1	Service Request enable: When this bit is 1, the device may transmit a Service Request. See the section "Service Request Enable/Disable," later in this chapter, for more details on this function.
	0	When this bit is 0, the device may not transmit a Service Request.
12	–	Reserved; must be 0.
11–8	–	Device address: This field contains the device's unique bus address. The possible addresses are listed in Table 6-14.
7–0	–	Device handler: This field contains the special handler code that defines the function of the device. See the next section, "Device Handlers," for a list of unique handlers.

Device handlers

Device handlers provide a means for a device to function in more than one manner. By sending a new handler ID, the device can be instructed to perform a new function.

There are two kinds of handlers: reserved, and all others. There are four reserved device handlers, listed in Table 6-13. Other handlers may be used for special device functions, but new device handlers defined by third-party developers must first be registered with Apple Computer, Inc.

Upon receiving a reserved device handler, the device will immediately perform the new function. The device will not store the reserved handler in register 3; only device-defined handler codes are stored. All unrecognized handlers are ignored.

Device addresses

Each peripheral device is preassigned a 4-bit command address, which identifies its device type. For example, all relative devices, such as a mouse, power up at address 3. Most devices have movable addresses. That is, the host can assign a new address to the device. The host

■ **Table 6-13** Reserved device handlers

Handler	Definition
$FF	Initiates a self-test in the device.
$FE	As Listen register 3 data, instructs the device to change the address field to the new address sent by the host if no collision has been detected.
$FD	As Listen register 3 data, instructs the device to change the address field to the new address sent by the host if the activator is pressed.
$00	As Listen register 3 data, instructs the device to change the address and enable fields to the new values sent by the host.
$00	As data sent in response to a Talk register 3 command, indicates that the device failed a self-test.

must assign a new address when two devices have the same default address (such as two mouse devices); one must be moved to a new address. A device will always default to its assigned address upon power-on or after it detects an ADB reset. Currently, eight addresses are predefined or reserved. The other eight addresses are available for movable devices. This means that ADB can support up to nine mouse devices, keyboards, or graphics tablets at the same time, each one with a unique address. Table 6-14 lists all the possible device addresses.

■ **Table 6-14** Device addresses

Address	Device class	Device type	Example
$00	Reserved	–	–
$01	Reserved	–	–
$02	Encoded devices	Movable	Keyboard
$03	Relative devices	Movable	Mouse
$04	Absolute devices	Movable	Graphics tablet
$05	Reserved	–	–
$06	Reserved	–	–
$07	Reserved	–	–
$08	Soft address	Movable	Any
$09	Soft address	Movable	Any
$0A	Soft address	Movable	Any
$0B	Soft address	Movable	Any
$0C	Soft address	Movable	Any
$0D	Soft address	Movable	Any
$0E	Soft address	Movable	Any
$0F	Soft address	Movable	Any

Collision detection

All devices must be able to detect collisions. If a device is attempting to output a bit and the data line is forced low by another device, it has lost a bit in collision with the other device. If another device sends data before the device is able to assert its start bit, it has lost a collision. The losing device should immediately stop transmitting and preserve the data that were being sent. A device sets an internal flag if it loses a collision.

◆ *Note:* Devices using internal clocks that operate within ±1 percent should attempt to assert their start bit at a random time within the limits of the bus turnaround time.

To aid in collision detection, the address field of register 3 is replaced with a random number in response to a Talk register 3 command. A device will change its device address to this new address as long as it has not detected a collision. A device that has detected a collision will not change its address during the next Listen register 3 command.

At the systems level, a host can change the addresses of normal devices by using this technique. By issuing a Talk register 3 command and following it with a Listen R3 command with a new address in bits 8 to 11 of the data packet, the host moves all devices that did not detect a collision to the new address. Typically, only one device will not detect a collision. This technique can be repeated at new addresses until the response to the Talk register 3 command is a time-out (no response). This process can be used to identify and relocate multiple devices of the same type after system initialization.

A normal device may have an optional *activator* on it. The activator can be a special key on a keyboard or a mouse button. At the application level, addresses can be changed by the host's displaying a message requesting a user to use the activator (hold down a key). By using the Listen register 3 command, the host can move the device with the activator pressed to a new address. This method can be used by an application program to identify and locate individual devices in multi-user applications. Also, certain reserved handlers are used to facilitate both address-changing methods.

Service Request enable/disable

It is possible to control the ability of a device to transmit a Service Request. To disable a device's ability to send a Service Request, set bit 12 in register 3 to 0; to enable it, set this bit to 1. This feature is useful in an application where the Service Request response time in a polled system is longer than desired. When only specific devices are required for an application, the others can be disabled.

1 MB Apple IIGS

The 1 MB Apple IIGS main logic board includes a redesigned ADB microcontroller. This IC has RAM expanded to 96 bytes, and ROM expanded to 4K. The new ROM code supports sticky keys and the ADB mouse functions. This information is provided here for developers who may have need to provide these functions in a third-party ADB device. For complete instructions on using sticky keys and ADB mouse, see the *Apple IIGS Owner's Guide*.

Sticky keys

The new ADB microcontroller provides a new feature useful to anyone who is limited to pressing only one key at a time. By enabling sticky keys, any combination of modifier keys (Shift, Command, Option, Control) can be achieved by pressing the keys sequentially rather than simultaneously. Table 6-15 lists these functions and the action required to implement each.

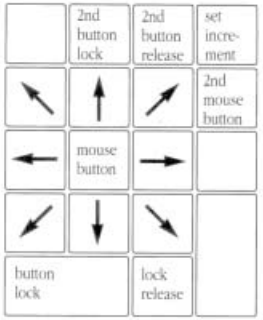
▪ **Table 6-15** Sticky keys functions

Function	Action
Enable sticky keys	Press Shift key five times
Enable a modifier key	Press the modifier key once
Lock down modifier key	Press the modifier key twice
Disable a modifier key	Press the modifier key a total of three times
Disable sticky keys	Press Shift key five times

ADB mouse

The ADB microcontroller provided with the 1 MB Apple IIGS includes the ADB mouse feature. This feature allows users to use the numeric keypad on the ADB keyboard to control mouse functions. Figure 6-16 shows the key functions, and Table 6-16 lists the mouse functions and the keys that implement these functions.

▪ **Figure 6-16** ADB mouse keypad



▪ **Table 6-16** ADB mouse functions

Function	Action
Enable ADB mouse.	Press Shift-Command-Clear key sequence.
Click mouse button.	Press keypad 5.
Lock down mouse button.	Press keypad 0.
Release mouse button.	Press keypad decimal.
Click second mouse button.	Press keypad "–".
Lock down second button.	Press keypad "=".
Release second button.	Press keypad "/".
Set cursor increment.	Press keypad "*" followed by 0–9.
Set cursor default.	Press keypad "*" twice.
Disable ADB mouse.	Press the Clear key.