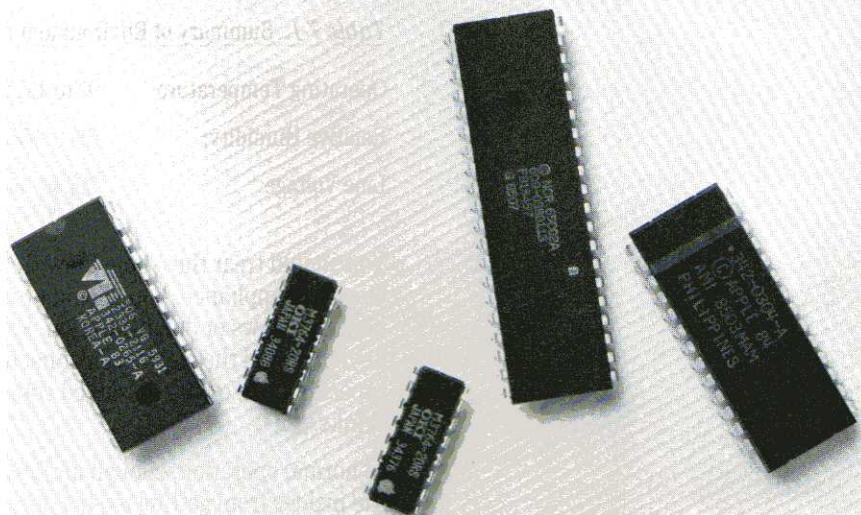


## Chapter 7

## Hardware Implementation



Most of this manual describes functions—what the Apple IIe does. This chapter, on the other hand, describes objects: the pieces of hardware the Apple IIe uses to carry out its functions. If you are designing a piece of peripheral hardware to attach to the Apple IIe, or if you just want to know more about how the Apple IIe is built, you should study this chapter.

## Environmental Specifications

The Apple IIe is quite sturdy when used in the way it was intended. Table 7-1 defines the conditions under which the Apple IIe is designed to function properly.

*Table 7-1.* Summary of Environmental Specifications

Operating Temperature:	0° to 45° C (30° to 115° F)
Relative Humidity:	5% to 85%
Line Voltage:	107 to 132 VAC

You should treat the Apple IIe with the same kind of care as any other electrical appliance. You should protect it from physical violence, such as hammer blows or defenestration. You should protect the mechanical keyboard and the electrical connectors inside the case from spilled liquids, especially those with dissolved contaminants, such as coffee and cola drinks.

In normal operation, enough air flows through the slots in the case to keep the insides from getting too hot, although some of the parts inside the Apple IIe normally get rather warm to the touch. If you manage to overheat your Apple IIe, by blocking the ventilation slots in the top and bottom for example, the first symptom will be erratic operation. The memory devices in the Apple IIe are sensitive to heat: when they get too hot, they occasionally change a bit of data. The exact result depends on what kind of program you are running and on just which bit of memory is affected.

## The Power Supply

The power supply in the Apple IIe operates on normal household AC power and provides enough low-voltage electrical power for the built-in electronics plus a full complement of peripheral cards, including disk controller cards and communications interfaces. The basic specifications of the power supply are listed in Table 7-2.

The Apple IIe's power cord should be plugged into a three-wire 110- to 120-volt outlet. You must connect the Apple IIe to a grounded outlet or to a good earth ground. Also, the line voltage must be in the range given in Table 7-2. If you try to operate the Apple IIe from a power source with more than 140 volts, you will damage the power supply.

*Table 7-2. Power Supply Specifications*

Line voltage:	107V to 132V AC
Maximum power consumption:	60W continuous 80W intermittent*
Supply voltages:	+5V $\pm$ 3% +11.8V $\pm$ 6% -5.2V $\pm$ 10% -12V $\pm$ 10%
Maximum supply currents:	+5V: 2.5A +12V: 1.5A continuous, 2.5A intermittent* -5V: 250mA -12V: 250mA
Maximum case temperature:	55° C (130° F)

\* Intermittent operation: The Apple IIe can safely operate for up to twenty minutes at the higher load if followed by at least ten minutes at normal load.

The Apple IIe uses a custom-designed switching-type power supply. It is small and lightweight, and it generates less heat than other types of power supplies do.

The Apple IIe's power supply works by converting the AC line voltage to DC and using this DC voltage to power a variable-frequency oscillator. The oscillator drives a small transformer with many separate windings to produce the different voltages required. A circuit compares the voltage of the +5-volt supply with a reference voltage and feeds an error signal back to the oscillator circuit. The oscillator circuit uses the error signal to control the frequency of its oscillation and keep the output voltages in their normal ranges.

The power supply includes circuitry to protect itself and the other electronic parts of the Apple IIe by turning off all four supply voltages whenever it detects one of the following malfunctions:

- any supply voltage short-circuited to ground
- the power-supply cable disconnected
- any supply voltage outside the normal range

Any time one of these malfunctions occurs, the protection circuit stops the oscillator, and all the output voltages drop to zero. After about half a second, the oscillator starts up again. If the malfunction is still occurring, the protection circuit stops the oscillator again. The power supply will continue to start and stop this way until the malfunction is corrected or the power is turned off.

### ▲Warning

If you think the power supply is broken, do not attempt to repair it yourself. The power supply is in a sealed enclosure because some of its circuits are connected directly to the power line. Special equipment is needed to repair the power supply safely, so see your authorized Apple dealer for service.

## The Power Connector

The cable from the power supply is connected to the main circuit board by a six-pin connector with a strain-relief catch. The connector pins are identified in Table 7-3 and Figure 7-13d.

*Table 7-3.* Power Connector Signal Specifications

Pin Number	Name	Description
1,2	Ground	Common electrical ground
3	+5V	+5V from power supply
4	+12V	+12V from power supply
5	-12V	-12V from power supply
6	-5V	-5V from power supply

## The 65C02 Microprocessor

The enhanced Apple IIe uses a 65C02 microprocessor as its central processing unit (CPU). The 65C02 in the Apple IIe runs at a clock rate of 1.023 MHz and performs up to 500,000 eight-bit operations per second. You should not use the clock rate as a criterion for comparing different types of microprocessors. The 65C02 has a simpler instruction cycle than most other microprocessors and it uses instruction pipelining for faster processing. The speed of the 65C02 with a 1MHz clock is equivalent to other types of microprocessors with clock rates up to 2.5MHz.

The 65C02 has a sixteen-bit address bus, giving it an address space of 64K (2 to the sixteenth power or 65536) bytes. The Apple IIe uses special techniques to address a total of more than 64K: see the sections "Bank-Switched Memory" and "Auxiliary Memory and Firmware" in Chapter 4 and the section "Switching I/O Memory" in Chapter 6.

See Appendix A for a description of the 65C02's instruction set and electrical characteristics.

*Table 7-4.* 65C02 Microprocessor Specifications

Type:	65C02
Register Complement:	8-bit Accumulator (A) 8-bit Index Registers (X,Y) 8-bit Stack Pointer (S) 8-bit Processor Status (P) 16-bit Program Counter (PC)
Data Bus:	Eight bits wide
Address Bus:	Sixteen bits wide
Address Range:	65,536 (64K)
Interrupts:	IRQ (maskable) NMI (non-maskable) BRK (programmed)
Operating Voltage:	+5V ( $\pm$ 5%)
Power Dissipation:	5 mW (at 1 MHz)

### 65C02 Timing

The operation of the Apple IIe is controlled by a set of synchronous timing signals, sometimes called clock signals. In electronics, the word *clock* is used to identify signals that control the timing of circuit operations. The Apple IIe doesn't contain the kind of clock you tell time by, although its internal timing is accurate enough that a program running on the Apple IIe can simulate such a clock.

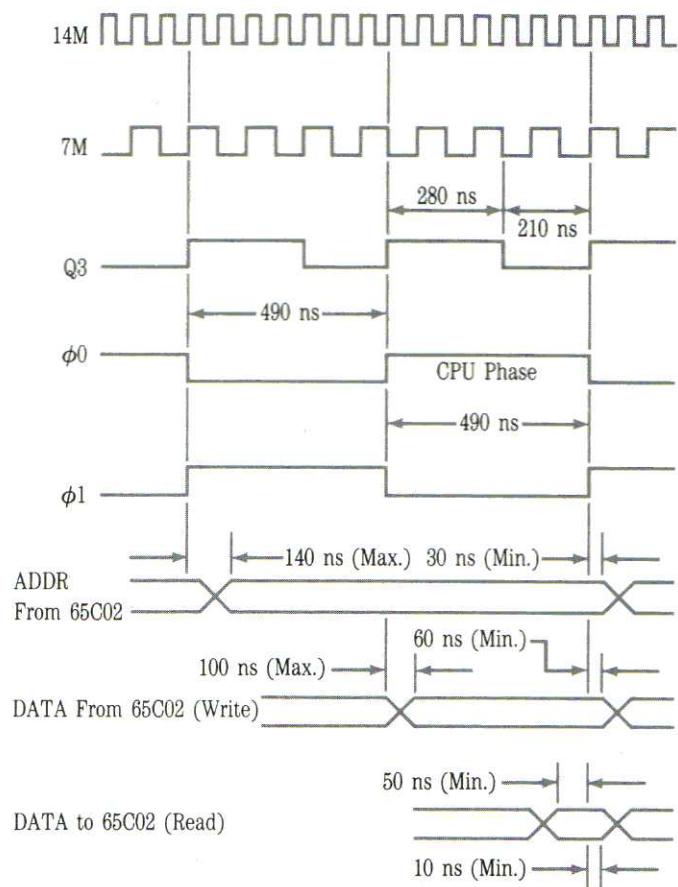
The frequency of the oscillator that generates the master timing signal is 14.31818 MHz. Circuitry in the Apple IIe uses this clock signal, called 14M, to produce all the other timing signals. These timing signals perform two major tasks: controlling the computing functions, and generating the video display. The timing signals directly involved with the operation of the 65C02 (and 6502 on the original version of the Apple IIe) are described in this section. Other timing signals are described in this chapter in the sections "RAM Addressing," "Video Display Modes," and "The Expansion Slots."

The main 65C02 timing signals are listed in Table 7-5, and their relationships are diagrammed in Figure 7-1. The 65C02 clock signals are  $\phi_1$  and  $\phi_0$ , complementary signals at a frequency of 1.02273 MHz. The Apple IIe signal named  $\phi_0$  is equivalent to the signal called  $\phi_2$  in the hardware manual. (It isn't identical: it's a few nanoseconds early.)

*Table 7-5.* 65C02 Timing Signal Descriptions

Signal Name	Description
14M	Master oscillator, 14.318 MHz; also 80-column dot clock
VID7M	Intermediate timing signal and 40-column dot clock
Q3	Intermediate timing signal, 2.045 MHz with asymmetrical duty cycle
$\phi_0$	Phase 0 of 65C02 clock, 1.0227 MHz; complement of $\phi_1$
$\phi_1$	Phase 1 of 65C02 clock, 1.0227 MHz; complement of $\phi_0$

*Figure 7-1.* 65C02 Timing Signals



The operations of the 65C02 are related to the clock signals in a simple way: address during  $\phi_1$ , data during  $\phi_0$ . The 65C02 puts an address on the address bus during  $\phi_1$ . This address is valid not later than 140 nanoseconds after  $\phi_1$  goes high and remains valid through all of  $\phi_0$ . The 65C02 reads or writes data during  $\phi_0$ . If the 65C02 is writing, the read/write signal is low during  $\phi_0$  and the 65C02 puts data on the data bus. The data is valid not later than 75 nanoseconds after  $\phi_0$  goes high. If the 65C02 is reading, the read/write signal remains high. Data on the data bus must be valid no later than 50 nanoseconds before the end of  $\phi_0$ .

## The Custom Integrated Circuits

Most of the circuitry that controls memory and I/O addressing in the Apple IIe is in three custom integrated circuits called the Memory Management Unit (MMU), the Input/Output Unit (IOU), and the Programmed Array Logic device (PAL). The soft switches used for controlling the various I/O and addressing modes of the Apple IIe are addressable flags inside the MMU and the IOU. The functions of these two devices are not as independent as their names suggest; working together, they generate all of the addressing signals. For example, the MMU generates the address signals for the CPU, while the IOU generates similar address signals for the video display.

### The Memory Management Unit

The circuitry inside the MMU implements these soft switches, which are described in the indicated chapters in this manual:

- Page 2 display (PAGE2): Chapter 2
- High resolution mode (HIRES): Chapter 2
- Store to 80-column card (80STORE): Chapter 2
- Select bank 2: Chapter 4
- Enable bank-switched RAM: Chapter 4
- Read auxiliary memory (RAMRD): Chapter 4
- Write auxiliary memory (RAMWRT): Chapter 4
- Auxiliary stack and zero page (ALTZP): Chapter 4
- Slot ROM for connector #3 (SLOTC3ROM): Chapter 6
- Slot ROM in I/O space (SLOTCXROM): Chapter 6

The 64K dynamic RAMs used in the Apple IIe use a multiplexed address, as described later in this chapter in the section “Dynamic-RAM Timing.” The MMU generates this multiplexed address for memory reading and writing by the 65C02 CPU. The pinouts and signal descriptions of the MMU are shown in Figure 7-2 and Table 7-6.

*Figure 7-2.* The MMU Pinouts

GND	1	40	A1
A0	2	39	A2
$\phi_0$	3	38	A3
Q3	4	37	A4
PRAS'	5	36	A5
RA0	6	35	A6
RA1	7	34	A7
RA2	8	33	A8
RA3	9	32	A9
RA4	10	31	A10
RA5	11	30	A11
RA6	12	29	A12
RA7	13	28	A13
R/W'	14	27	A14
INH'	15	26	A15
DMA'	16	25	+5V
EN80'	17	24	Cxxx
KBD'	18	23	RAMEN'
ROMEN2'	19	22	R/W' 245
ROMEN1'	20	21	MD7

*Table 7-6.* The MMU Signal Descriptions

Pin Number	Name	Description
1	GND	Power and signal common
2	A0	65C02 address input
3	$\phi_0$	Clock phase 0 input
4	Q3	Timing signal input
5	PRAS'	Memory row-address strobe
6-13	RA0-RA7	Multiplexed address output
14	R/W'	65C02 read-write control signal
15	INH'	Inhibits main memory (tied to +5 V)
16	DMA'	Controls data bus for DMA transfers
17	EN80'	Enables auxiliary RAM
18	KBD'	Enables keyboard data bits 0-6
19	ROMEN2'	Enables ROM (tied to ROMEN1')
20	ROMEN1'	Enables ROM (tied to ROMEN2')
21	MD7	State of MMU flags on data bus bit 7
22	RW'245	Controls 74LS245 data-bus buffer
23	RAMEN'	Enables main RAM
24	Cxxx	Enables peripheral-card memory
25	+5V	Power
26-40	A15-A1	65C02 address input

## The Input/Output Unit

The circuitry inside the Input/Output Unit (IOU) implements the following soft switches, all described in Chapter 2 in this manual:

- Page 2 display (PAGE2)
- High resolution mode (HIRES)
- Text mode (TEXT)
- Mixed mode (MIXED)
- 80-column display (80COL)
- Text display mode select (ALTCHAR)
- Any-key-down
- Annunciators
- Vertical blanking (VBL)

The 64K dynamic RAMs used in the Apple IIe require a multiplexed address, as described later in this chapter in the section “Dynamic-RAM Timing.” The IOU generates this multiplexed address for the data transfers required for display and memory refresh during clock phase 1. The way this address is generated is described later in this chapter in the section “Display Address Mapping.” The pinouts and signal descriptions for the IOU are shown in Figure 7-3 and Table 7-7.

**Figure 7-3.** The IOU Pinouts

GND	1	40	H0
GR	2	39	SYNC'
SEGA	3	38	WNDW'
SEGB	4	37	CLRGAT'
VC	5	36	RA10'
80VID'	6	35	RA9'
CASSO	7	34	VID6
SPKR	8	33	VID7
MD7	9	32	KSTRB
AN0	10	31	AKD
AN1	11	30	C0xx
AN2	12	29	A6
AN3	13	28	+5V
R/W'	14	27	Q3
RESET'	15	26	φ0
(n.c.)	16	25	PRAS'
RA0	17	24	RA7
RA1	18	23	RA6
RA2	19	22	RA5
RA3	20	21	RA4

**Table 7-7.** The IOU Signal Descriptions

Pin Number	Name	Description
1	GND	Power and signal common
2	GR	Graphics mode enable
3	SEGA	In text mode, works with VC (see pin 5) and SEGB to determine character row address
4	SEGB	In text mode, works with VC (see pin 5) and SEGA; in graphics mode, selects high-resolution when low, low-resolution when high
5	VC	Display vertical counter bit: in text mode, SEGA, SEGB and VC determine which of the eight rows of a character's dot pattern to display; in low-resolution, selects upper or lower block defined by a byte.
6	80VID'	80-column video enable
7	CASSO	Cassette output signal
8	SPKR	Speaker output signal
9	MD7	Internal IOU flags for data bus (bit 7)
10-13	AN0-AN3	Annunciator outputs
14	R/W'	65C02 read-write control signal
15	RESET'	Power on and reset output
16		Nothing is connected to this pin.
17-24	RA0-RA7	Video refresh multiplexed RAM address (phase 1)
25	PRAS'	Row-address strobe (phase 0)
26	φ0	Master clock phase 0
27	Q3	Intermediate timing signal
28	+5V	Power
29	A6	Address bit 6 from 65C02
30	C0xx	I/O address enable
31	AKD	Any-key-down signal
32	KSTRB	Keyboard strobe signal
33,34	VIDD7,VIDD6	Video display data bits
35,36	RA9',RA10'	Video display control bits
37	CLRGAT'	Color-burst gate (enable)
38	WNDW'	Display blanking signal
39	SYNC'	Display synchronization signal
40	H0	Display horizontal timing signal (low bit of character counter)

## The PAL Device

A Programmed Array Logic device, type PAL 16R8, generates several timing and control signals in the Apple IIe. These signals are listed in Table 7-8. The PAL pinouts are given in Figure 7-4.

Figure 7-4. The PAL Pinouts

14M	1	20	+5V
7M	2	19	PRAS'
3.58M	3	18	(n.c.)
H0	4	17	PCAS'
VID7	5	16	Q3
SEGB	6	15	$\phi_0$
GR	7	14	$\phi_1$
RAMEN'	8	13	VID7M
80VID'	9	12	LDPS'
GND	10	11	ENTMG

Table 7-8. The PAL Signal Descriptions

Pin Number	Name	Description
1	14M	14.31818 MHz master timing signal
2	7M	7.15909 MHz timing signal
3	3.58M	3.579545 MHz timing signal
4	H0	Horizontal video timing signal
5	VID7	Video data bit 7
6	SEGB	Video timing signal
7	GR	Video display graphics-mode enable
8	RAMEN'	RAM enable (CAS enable)
9	80VID'	Enable 80-column display mode
10	GND	Power and signal common
11	ENTMG	Enable master timing
12	LDPS'	Video shift-register load enable
13	VID7M	Video dot clock, 7 or 14 MHz
14	$\phi_1$	Phase 1 system clock
15	$\phi_0$	Phase 0 system clock
16	Q3	Intermediate timing and strobe signal
17	PCAS'	RAM column-address strobe
18	N.C.	(This pin is not used.)
19	PRAS'	RAM row-address strobe
20	+5V	Power

## Memory Addressing

The Apple IIe's microprocessor can address 65,536 locations. The Apple IIe uses this entire address space, and then some: some areas in memory are used for more than one function. The following sections describe the memory devices used in the Apple IIe and the way they are addressed. Input and output also use portions of the memory address space; refer to the section "Peripheral-Card Memory Spaces" in Chapter 6 for information.

**Figure 7-5.** The 2364 ROM Pinouts

+5V	1	28	+5V
A12	2	27	+5V
A7	3	26	+5V
A6	4	25	A8
A5	5	24	A9
A4	6	23	A11
A3	7	22	ROMENx'
A2	8	21	A10
A1	9	20	CE'
A0	10	19	MD7
MD0	11	18	MD6
MD1	12	17	MD5
MD2	13	16	MD4
GND	14	15	MD3

**Figure 7-6.** The 2316 ROM Pinouts

A7	1	24	+5V
A6	2	23	A8
A5	3	22	A9
A4	4	21	+5V
A3	5	20	KBD'
A2	6	19	GND
A1	7	18	ENKBD'
A0	8	17	(n.c.)
MD0	9	16	MD6
MD1	10	15	MD5
MD2	11	14	MD4
GND	12	13	MD3

**Figure 7-7.** The 2333 ROM Pinouts

VID4	1	24	+5V
VID3	2	23	VID5
VID2	3	22	RA9
VID1	4	21	GR
VID0	5	20	WNDW'
VC	6	19	RA10
SEGB	7	18	ENVID'
SEGA	8	17	D7
D0	9	16	D6
D1	10	15	D5
D2	11	14	D4
GND	12	13	D3

## ROM Addressing

In the Apple IIe, the following programs are permanently stored in two type 2364 8K by 7-bit ROMs (read-only memory):

- Applesoft editor and interpreter
- System Monitor
- 80-column display firmware
- self-test routines

These two ROMs are enabled by two signals called ROMEN1 and ROMEN2. The ROM enabled by ROMEN1, sometimes called the Diagnostics ROM, occupies the memory address space from \$C100 to \$DFFF. The address space from \$C300 to \$C3FF and from \$C800 to \$CFFF contains the 80-column display firmware. Those address spaces are normally assigned to ROM on a peripheral card in slot 3; for a discussion of the way the 80-column firmware overrides the peripheral card, see the section "Other Uses of I/O Memory Space" in Chapter 6. The pinouts of the 2364 ROMs are given in Figure 7-5.

Two other portions of the Diagnostics ROM, addressed from \$C100 to \$C2FF and from \$C400 to \$C7FF, contain the built-in self-test routines. These address spaces are normally assigned to the peripheral cards; when the self-test programs are running, the peripheral cards are disabled.

The remainder of the Diagnostics ROM, addressed from \$D000 to \$DFFF, contains part of the Applesoft BASIC interpreter.

The ROM enabled by ROMEN2, sometimes called the Monitor ROM, occupies the memory address space from \$E000 to \$FFFF. This ROM contains the rest of the Applesoft interpreter, in the address space from \$E000 to \$EFFF, and the Monitor subroutines, from \$F000 to \$FFFF.

The other ROMs in the Apple IIe are a type 2316 ROM used for the keyboard character decoder and a type 2333 ROM used for character sets for the video display. This 2333 ROM is rather large because it includes a section of straight-through bit-mapping for the graphics modes. This way, graphics display video can pass through the same circuits as text without additional switching circuitry. The 2316's pinout is given in Figure 7-6, and the 2333's pinout is given in Figure 7-7.

*Figure 7-8.* The 64K RAM Pinouts

+5V	1	16	GND
MDx	2	15	CAS'
R/W'	3	14	MDx
RAS'	4	13	RA1
RA7	5	12	RA4
RA5	6	11	RA3
RA6	7	10	RA2
+5V	8	9	RA0

## RAM Addressing

The RAM (programmable) memory in the Apple IIe is used both for program and data storage and for the video display. The areas in RAM that are used for the display are accessed both by the 65C02 microprocessor and by the video display circuits. In some computers, this dual access results in addressing conflicts (cycle stealing) that can cause temporary dropouts in the video display. This problem does not occur in the Apple IIe, thanks to the way the microprocessor and the video circuits share the memory.

The memory circuits in the Apple IIe take advantage of the two-phase system clock described earlier in this chapter in the section “65C02 Timing” to interleave the microprocessor memory accesses and the display memory accesses so that they never interfere with each other. The microprocessor reads or writes to RAM only during  $\phi_0$ , and the display circuits read data only during  $\phi_1$ .

## Dynamic-RAM Refreshment

The image on a video display is not permanent; it fades rapidly and must be refreshed periodically. To refresh the video display, the Apple IIe reads the data in the active display page and sends it to the display. To prevent visible flicker in the display, and to conform to standard practice for broadcast video, the Apple IIe refreshes the display sixty times per second.

The dynamic RAM devices used in the Apple IIe also need a kind of refresh, because the data is stored in the form of electric charges which diminish with time and must be replenished every so often. The Apple IIe is designed so that refreshing the display also refreshes the dynamic RAMs. The next few paragraphs explain how this is done.

The job of refreshing the dynamic RAM devices is minimized by the structure of the devices themselves. The individual data cells in each RAM device are arranged in a rectangular array of rows and columns. When the device is addressed, the part of the address that specifies a row is presented first, followed by the address of the column. Splitting information into parts that follow each other in time is called multiplexing. Since only half of the address is needed at one time, multiplexing the address reduces the number of pins needed for connecting the RAMs.

Different manufacturers’ 64K RAMs have cell arrays of either 128 rows by 512 columns or 256 rows by 256 columns. Only the row portion of the address is used in refreshing the RAMs.

Now consider how the display is refreshed. As described later in this chapter in the section “The Video Counters,” the display circuitry generates a sequence of 8,192 memory addresses in high-resolution mode; in text and low-resolution modes, this sequence is the 1,024 display-page addresses repeated eight times. The display address cycles through this sequence 60 times a second, or once every 17 milliseconds. The way the low-order address lines are assigned to the RAMs, the row address cycles through all 256 possible values once every two milliseconds. (See Figure 7-9.) This more than satisfies the refresh requirements of the dynamic RAMs.

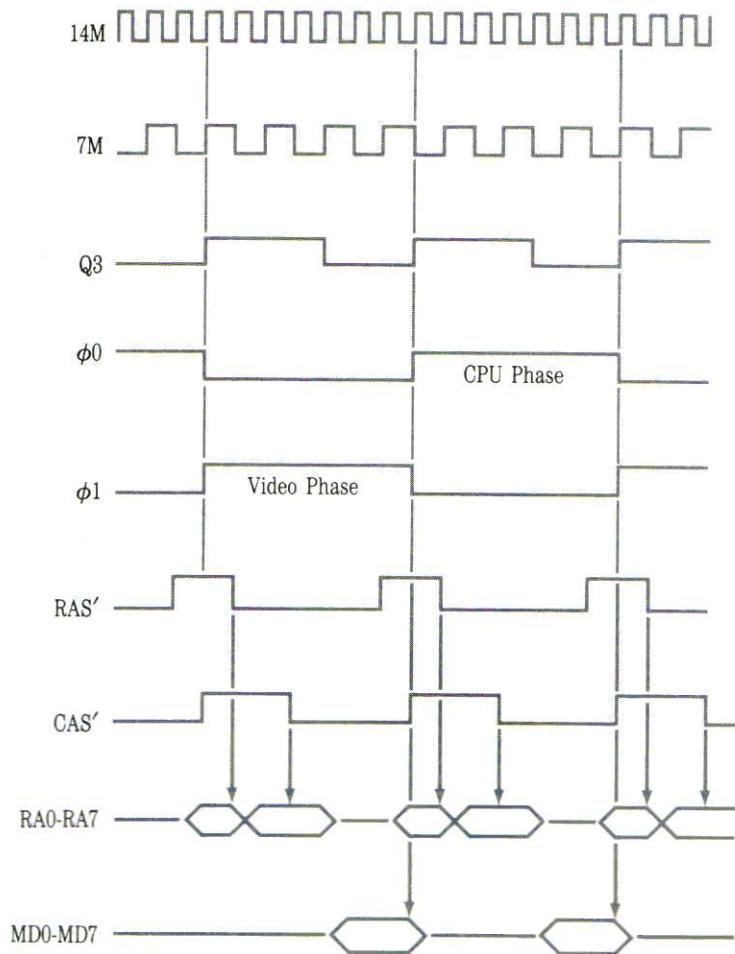
*Table 7-9.* RAM Address Multiplexing

Mux'd Address	Row Address	Column Address
RA0	A0	A9
RA1	A1	A6
RA2	A2	A10
RA3	A3	A11
RA4	A4	A12
RA5	A5	A13
RA6	A7	A14
RA7	A8	A15

### **Dynamic-RAM Timing**

The Apple IIe's microprocessor clock runs at a moderate speed, about 1.023 MHz, but the interleaving of CPU and display cycles means that the RAM is being accessed at a 2 MHz rate, or a cycle time of just under 500 nanoseconds. Data for the CPU is strobed by the falling edge of  $\phi_0$ , and display data is strobed by the falling edge of  $\phi_1$ , as shown in Figure 7-9.

*Figure 7-9.* RAM Timing Signals



The RAM timing looks complicated because the RAM address is multiplexed, as described in the previous section. The MMU takes care of multiplexing the address for the CPU cycle, and the IOU performs the same function for the display cycle. The multiplexed address is sent to the RAM ICs over the lines labelled RA0-RA7. Along with the other timing signals, the PAL device generates two signals that control the RAM addressing: row-address strobe (RAS) and column-address strobe (CAS).

*Table 7-10.* RAM Timing Signal Descriptions

Signal Name	Description
$\phi_0$	Clock phase 0 (CPU phase)
$\phi_1$	Clock phase 1 (display phase)
RAS	Row-address strobe
CAS	Column-address strobe
Q3	Alternate RAM/column-address strobe
RA0-RA7	Multiplexed address bus
MD0-MD7	Internal data bus

## The Video Display

The Apple IIe produces a video signal that creates a display on a standard video monitor or, if you add an RF modulator, on a black-and-white or color television set. The video signal is a composite made up of the data that is being displayed plus the horizontal and vertical synchronization signals that the video monitor uses to arrange the lines of display data on the screen.

**Video Standards:** Apple IIe's manufactured for sale in the U.S. generate a video signal that is compatible with the standards set by the NTSC (National Television Standards Committee). Apple IIe's manufactured for sale in European countries generate video that is compatible with the standard used there, which is called P.A.L. (for phase alternating lines). This manual describes only the NTSC version of the video circuits.

The display portion of the video signal is a time-varying voltage generated from a stream of data bits, where a 1 corresponds to a voltage that generates a bright dot, and a 0 to a dark dot. The display bit stream is generated in bursts that correspond to the horizontal lines of dots on the video screen. The signal named WNDW' is low during these bursts.

During the time intervals between bursts of data, nothing is displayed on the screen. During these intervals, called the blanking intervals, the display is blank and the WNDW' signal is high. The synchronization signals, called *sync* for short, are produced by making the signal named SYNC' low during portions of the blanking intervals. The sync pulses are at a voltage equivalent to blacker-than-black video and don't show on the screen.

### The Video Counters

The address and timing signals that control the generation of the video display are all derived from a chain of counters inside the IOU. Only a few of these counter signals are accessible from outside the IOU, but they are all important in understanding the operation of the display generation process, particularly the display memory addressing described in the next section.

The horizontal counter is made up of seven stages: H0, H1, H2, H3, H4, H5, and HPE'. The input to the horizontal counter is the 1 MHz signal that controls the reading of data being displayed. The complete cycle of the horizontal counter consists of 65 states. The six bits H0 through H5 count normally from 0 to 63, then start over at 0. Whenever this happens, HPE' forces another count with H0 through H5 held at zero, thus extending the total count to 65.

The IOU uses the forty horizontal count values from 25 through 64 in generating the low-order part of the display data address, as described later in this chapter in the section "Display Address Mapping." The IOU uses the count values from 0 to 24 to generate the horizontal blanking, the horizontal sync pulse, and the color-burst gate.

When the horizontal count gets to 65, it signals the end of a line by triggering the vertical counter. The vertical counter has nine stages: VA, VB, VC, V0, V1, V2, V3, V4, and V5. When the vertical count reaches 262, the IOU resets it and starts counting again from zero. Only the first 192 scanning lines are actually displayed; the IOU uses the vertical counts from 192 to 261 to generate the vertical blanking and sync pulse. Nothing is displayed during the vertical blanking interval. (The vertical line count is 262 rather than the standard 262.5 because, unlike normal television, the Apple IIe's video display is not interlaced.)

***Smooth Animation:*** Animation displays sometimes have an erratic flicker caused by changing the display data at the same time it is being displayed. You can avoid this on the Apple IIe by reading the vertical-blanking signal (VBL) at location \$C019 and changing display data while VBL is low only (data value less than 128).

## Display Memory Addressing

As described in Chapter 2 in the section “Addressing Display Pages Directly,” data bytes are not stored in memory in the same sequence in which they appear on the display. You can get an idea of the way the display data is stored by using the Monitor to set the display to graphics mode, then storing data starting at the beginning of the display page at hexadecimal \$400 and watching the effect on the display. If you do this, you should use the graphics display instead of text to avoid confusion: the text display is also used for Monitor input and output.

If you want your program to display data by storing it directly into the display memory, you must first transform the display coordinates into the appropriate memory addresses, as shown in the section “Video Display Pages” in Chapter 2. The descriptions that follow will help you understand how this address transformation is done and why it is necessary. They will not (alas!) eliminate that necessity.

The address transformation that folds three rows of forty display bytes into 128 contiguous memory locations is the same for all display modes, so it is described first. The differences among the different display modes are then described in the section “Video Display Modes.”

## Display Address Mapping

Consider the simplest display on the Apple IIe, the 40-column text mode. To address forty columns requires six bits, and to address twenty-four rows requires another five bits, for a total of eleven address bits. Addressing the display this way would involve 2048 (2 to the eleventh power) bytes of memory to display a mere 960 characters. The 80-column text mode would require 4096 bytes to display 1920 characters. The leftover chunks of memory that were not displayed could be used for storing other data, but not easily, because they would not be contiguous.

Instead of using the horizontal and vertical counts to address memory directly, the circuitry inside the IOU transforms them into the new address signals described below. The transformed display address must meet the following criteria:

- Map the 960 bytes of 40-column text into only 1024 bytes.
- Scan the low-order address to refresh the dynamic RAMs.
- Continue to refresh the RAMs during video blanking.

The transformation involves only horizontal counts H3, H4, and H5, and vertical counts V3 and V4. Vertical count bits VA, VB, and VC address the lines making up the characters, and are not involved in the address transformation. The remaining low-order count bits, H0, H1, H2, V0, V1, and V2 are used directly, and are not involved in the transformation.

The IOU performs an addition that reduces the five significant count bits to four new signals called S0, S1, S2, and S3, where S stands for sum. Figure 7-10 is a diagram showing the addition in binary form, with V3 appearing as the carry in and H5 appearing as its complement H5'. A constant value of 1 appears as the low-order bit of the addend. The carry bit generated with the sum is not used.

*Table 7-11.* Display Address Transformation

			V3 Carry in
H5'	V3	H4	H3 Augend
V4	H5'	V4	1 Addend
S3	S2	S1	S0 Sum

The requirements of the RAM refreshing are discussed earlier in this chapter in the section "Dynamic-RAM Refreshment."

If this transformation seems terribly obscure, try it with actual values. For example, for the upper-left corner of the display, the vertical count is 0 and the horizontal count is 24: H0, H1, H2, and H5 are 0's and H3, and H4 are 1's. The value of the sum is 0, so the memory location for the first character on the display is the first location in the display page, as you might expect.

Horizontal bits H0, H1, and H2 and sum bits S0, S1, and S2 make up the transformed horizontal address (A0 through A6 in Table 7-12). As the horizontal count increases from 24 to 63, the value of the sum (S3 S2 S1 S0) increases from 0 to 4 and the transformed address goes from 0 to 39, relative to the beginning of the display page.

The low-order three bits of the vertical row counter are V0, V1, and V2. These bits control address bits A7, A8, and A9, as shown in Table 7-12, so that rows 0 through 7 start on 127-byte boundaries. When the vertical row counter reaches 8, then V0, V1, and V2 are 0 again, and V3 changes to 1. If you do the addition in Table 7-11 with H equal to 24 (the horizontal count for the first column displayed) and V equal to 8, the sum is 5 and the horizontal address is 40: the first character in row 8 is stored in the memory location 40 bytes from the beginning of the display page.

*Figure 7-10. 40-Column Text Display Memory*

*Memory locations marked with an asterisk (\*) are reserved for use by peripheral I/O firmware: refer to the section “Peripheral-Card RAM Space” in Chapter 6.*

\$400	row 0	row 8	row 16	*
\$480	row 1	row 9	row 17	*
\$500	row 2	row 10	row 18	*
\$580	row 3	row 11	row 19	*
\$600	row 4	row 12	row 20	*
\$680	row 5	row 13	row 21	*
\$700	row 6	row 14	row 22	*
\$780	row 7	row 15	row 23	*

Figure 7-10 shows how groups of three forty-character rows are stored in blocks of 120 contiguous bytes starting on 127-byte address boundaries. This diagram is another way of describing the display mapping shown in Figure 2-5. Notice that the three rows in each block of 120 bytes are not adjacent on the display.

Table 7-12 shows how the signals from the video counters are assigned to the address lines. H0, H1, and H2 are horizontal-count bits, and V0, V1, and V2 are vertical-count bits. S0, S1, S2 and S3 are the folded address bits described above. Address bits marked with asterisks (\*) are different for different modes; see Table 7-13 and the four subsections under the section “Video Display Modes.”

*Table 7-12.* Display Memory Addressing

Memory Address Bit	Display Address Bit	Memory Address Bit	Display Address Bit
A0	H0	A8	V1
A1	H1	A9	V2
A2	H2	A10	**
A3	S0	A11	**
A4	S1	A12	**
A5	S2	A13	**
A6	S3	A14	**
A7	V0	A15	GND

\*\* For these address bits, see text and Table 7-13.

*Table 7-13.* Memory Address Bits for Display Modes

. means logical AND; ' means logical NOT.

Address Bit	Display Modes	
	Text and Low-Resolution	High-Resolution and Double-High-Resolution
A10	80STORE+PAGE2'	VA
A11	80STORE'.PAGE2	VB
A12	0	VC
A13	0	80STORE+PAGE2'
A14	0	80STORE'.PAGE2

## **Video Display Modes**

The different display modes all use the address-mapping scheme described in the previous section, but they use different-sized memory areas in different locations. The next four sections describe the addressing schemes and the methods of generating the actual video signals for the different display modes.

## **Text Displays**

The text and low-resolution graphics pages begin at memory locations \$0400 and \$0800. Table 7-13 shows how the display-mode signals control the address bits to produce these addresses. Address bits A10 and A11 are controlled by the settings of PG2 and 80STORE, which are set by the display-page and 80-column-video soft switches. Address bits A12, A13, and A14 are set to 0. Notice that 80STORE active inhibits PG2: there is only one display page in 80-column mode.

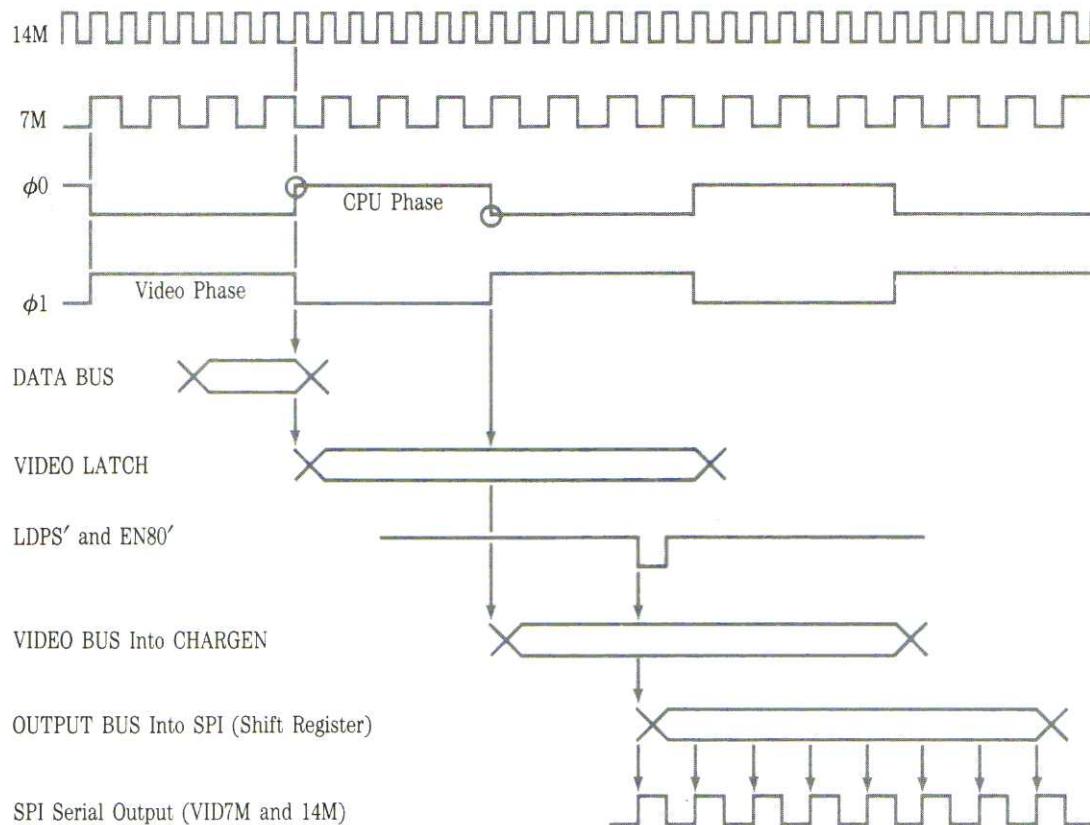
The bit patterns used for generating the different characters are stored in a 32K ROM. The low-order six bits of each data byte reach the character generator ROM directly, via the video data bus VID0-VID5. The two high-order bits are modified by the IOU to select between the primary and alternate character sets and are sent to the character generator ROM on lines RA9 and RA10.

The data for each row of characters are read eight times, once for each of the eight lines of dots making up the row of characters. The data bits are sent to the character generator ROM along with VA, VB, and VC, the low-order bits from the vertical counter. For each character being displayed, the character generator ROM puts out one of eight stored bit patterns selected by the three-bit number made up of VA, VB, and VC.

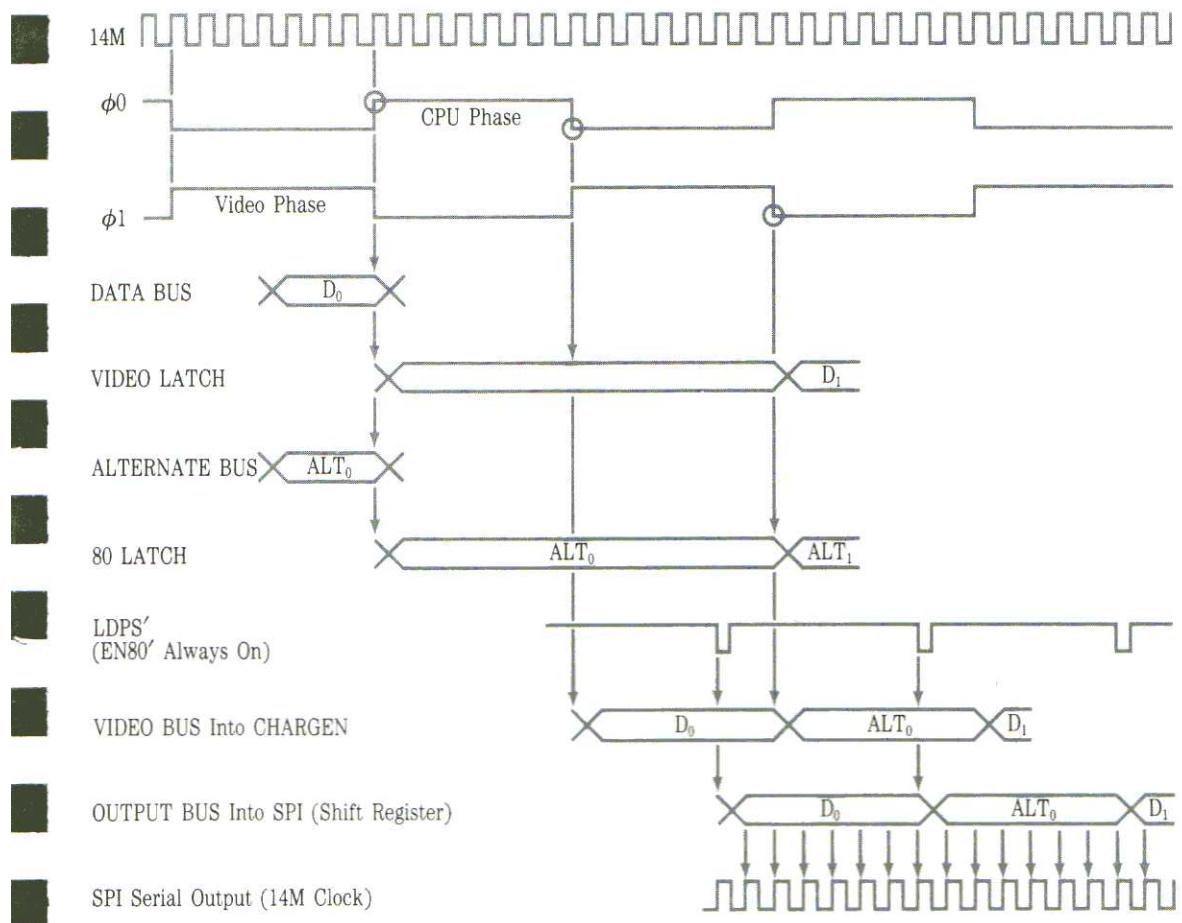
The bit patterns from the character generator ROM are loaded into the 74166 parallel-to-serial shift register and output as a serial bit stream that goes to the video output circuit. The shift register is controlled by signals named LDPS' (for load parallel-to-serial shifter) and VID7M (for video 7 MHz). In 40-column mode, LDPS' strobes the output of the character generator ROM into the shift register once each microsecond, and bits are sent to the screen at a 7 MHz rate.

The addressing for the 80-column display is exactly the same as for the 40-column display: the 40 columns of display memory on the 80-column card are addressed in parallel with the 40 columns in main memory. The data from these two memories reach the video data bus (lines VID0-VID7) via separate 74LS374 three-state buffers. These buffers are loaded simultaneously, but their outputs are sent to the character generator ROM alternately by  $\phi_0$  and  $\phi_1$ . In 80-column mode, LDPS' loads data from the character generator ROM into the shift register twice during each microsecond, once during  $\phi_0$  and once during  $\phi_1$ , and bits are sent to the screen at a 14 MHz rate. Figures 7-11a and 7-11b show the video timing signals.

*Figure 7-11a. 7 MHz Video Timing Signals*



**Figure 7-11b.** 14 MHz Video Timing Signals



## Low-Resolution Display

In the graphics modes, VA and VB are not used by the character generator, so the IOU uses lines SEGA and SEGB to transmit H0 and HIRES', as shown in Table 7-14.

*Table 7-14.* Character-Generator Control Signals

Display Mode	SEGA	SEGB	SEGC
Text	VA	VB	VC
Graphics	H0	HIRES'	VC

The low-resolution graphics display uses VC to divide the eight display lines corresponding to a row of characters into two groups of four lines each. Each row of data bytes is addressed eight times, the same as in text mode, but each byte is interpreted as two nibbles. Each nibble selects one of 16 colors. During the upper four of the eight display lines, VC is low and the low-order nibble determines the color. During the lower four display lines, VC is high and the high-order nibble determines the color.

The bit patterns that produce the low-resolution colors are read from the character-generator ROM in the same way the bit patterns for characters are produced in text mode. The 74166 parallel-to-serial shift register converts the bit patterns to a serial bit stream for the video circuits.

The video signal generated by the Apple IIe includes a short burst of 3.58 MHz signal that is used by an NTSC color monitor or color TV set to generate a reference 3.58 MHz color signal. The Apple IIe's video signal produces color by interacting with this 3.58 MHz signal inside the monitor or TV set. Different bit patterns produce different colors by changing the duty cycles and delays of the bit stream relative to the 3.58 MHz color signal. To produce the small delays required for so many different colors, the shift register runs at 14 MHz and shifts out 14 bits during each cycle of the 1-MHz data clock. To generate a stream of fourteen bits from each eight-bit pattern read from the ROM, the output of the shift register is connected back to the register's serial input to repeat the same eight bits; the last two bits are ignored the second time around.



Each bit pattern is output for the same amount of time as a character: .98 microseconds. Because that is exactly enough time for three and a half cycles of the 3.58 MHz color signal, the phase relationship between the bit patterns and the signal changes by a half cycle for each successive pattern. To compensate for this, the character generator ROM puts out one of two different bit patterns for each nibble, depending on the state of H0, the low-order bit of the horizontal counter.

### **High-Resolution Display**

The high-resolution graphics pages begin at memory locations \$2000 and \$4000 (decimal 8192 and 16384). These page addresses are selected by address bits A13 and A14. In high-resolution mode, these address bits are controlled by PG2 and 80STORE, the signals controlled by the display-page (PAGE2) and 80-column-video (80COL) soft switches. As in text mode, 80STORE inhibits addressing of the second page because there is only one page of 80-column text available for mixed mode.

In high-resolution graphics mode, the display data are still stored in blocks like the one shown in Figure 7-10, but there are eight of these blocks. As Table 7-12 and Table 7-13 show, vertical counts VA, VB, and VC are used for address bits A10, A11, and A12, which address eight blocks of 1024 bytes each. Remember that in the display, VA, VB, and VC count adjacent horizontal lines in groups of eight. This addressing scheme maps each of those lines into a different 1024-byte block. It might help to think of it as a kind of eight-way multiplexer: it's as if eight text displays were combined to produce a single high-resolution display, with each text display providing one line of dots in turn, instead of a row of characters.

The high-resolution bit patterns are produced by the character-generator ROM. In this mode, the bit patterns simply reproduce the eight bits of display data. The low-order six bits of data reach the ROM via the video data bus VID0-VID5. The IOU sends the other two data bits to the ROM via RA9 and RA10.

The high-resolution colors described in Chapter 2 are produced by the interaction between the video signal the bit patterns generate and the 3.58 MHz color signal generated inside the monitor or TV set. The high-resolution bit patterns are always shifted out at 7 MHz, so each dot corresponds to a half-cycle of the 3.58 MHz color signal. Any part of the video signal that produces a single white dot between two black dots, or vice versa, is effectively a short burst of 3.58 MHz and is therefore displayed as color. In other words, a bit pattern consisting of alternating 1's and 0's

gets displayed as a line of color. The high-resolution graphics subroutines produce the appropriate bit patterns by masking the data bits with alternating 1's and 0's.

To produce different colors, the bit patterns must have different phase relationships to the 3.58 MHz color signal. If alternating 1's and 0's produce a certain color, say green, then reversing the pattern to 0's and 1's will produce the complementary color, purple. As in the low-resolution mode, each bit pattern corresponds to three and a half cycles of the color signal, so the phase relationship between the data bits and the color signal changes by a half cycle for each successive byte of data. Here, however, the bit patterns produced by the hardware are the same for adjacent bytes; the color compensation is performed by the high-resolution software, which uses different color masks for data being displayed in even and odd columns.

To produce other colors, bit patterns must have other timing relationships to the 3.58 MHz color signal. In high-resolution mode, the Apple IIe produces two more colors by delaying the output of the shift register by half a dot (70 ns), depending on the high-order bit of the data byte being displayed. (The high-order bit doesn't actually get displayed as a dot, because at 7 MHz there is only time to shift out seven of the eight bits.)

As each byte of data is sent from the character generator to the shift register, high-order data bit D7 is also sent to the PAL device. If D7 is off, the PAL device transmits shift-register timing signals LDPS' and VID7M normally. If D7 is on, the PAL device delays LDPS' and VID7M by 70 nanoseconds, the time corresponding to half a dot. The bit pattern that formerly produced green now produces orange; the pattern for purple now produces blue.

*A Note About Timing:* For 80-column text, the shift register is clocked at twice normal speed. When 80-column text is used with graphics in mixed mode, the PAL device controls shift-register timing signals LDPS' and VID7M so that the graphics portion of the display works correctly even when the text window is in 80-column mode.

## **Double-High-Resolution Display**

Double-high-resolution graphics mode displays two bytes in the time normally required for one, but uses high-resolution graphics Page 1 in both main and auxiliary memory instead of text or low-resolution Page 1.

*Note:* There is a second pair of pages, high-resolution Page 2, which can be used to display a second double-high-resolution page.

Double-high-resolution graphics mode displays each pair of data bytes as 14 adjacent dots, seven from each byte. The high-order bit (color-select bit) of each byte is ignored. The auxiliary-memory byte is displayed first, so data from auxiliary memory appears in columns 0-6, 14-20, and so on, up to columns 547-552. Data from main memory appears in columns 7-13, 21-27, and so on, up to 553-559.

As in 80-column text, there are twice as many dots across the display screen, so the dots are only half as wide. On a TV set or low-bandwidth monitor (less than 14 MHz), single dots will be dimmer than normal.

*Note:* Except for some expensive RGB-type monitors, any video monitor with a bandwidth as high as 14 MHz will be a monochrome monitor. Monochrome means one color: a monochrome video monitor can have a screen color of white, green, orange, or any other single color.

The main memory and auxiliary memory are connected to the address bus in parallel, so both are activated during the display cycle. The rising edge of  $\phi_0$  clocks a byte of main memory data into the video latch, and a byte of auxiliary memory data into the 80 latch.

$\Phi_1 (\phi_1)$  enables output from the (auxiliary) 80 latch, and  $\phi_0$  enables output from the (main) video latch. Output from both latches goes to CHARGEN, where GR and SEGB' select high-resolution graphics. LDPS operates at 2 MHz in this mode, alternately gating the auxiliary byte and main byte into the parallel-to-serial shift register. VID7M is active (kept true) for double-high-resolution display mode, so when it is ANDed with 14M, the result is still 14M. The 14M serial clock signal gate shift register then outputs to VID, the video display hybrid circuit, for output to the display device.

## Video Output Signals

The stream of video data generated by the display circuits described above goes to a linear summing circuit built around transistor Q1 where it is mixed with the sync signals and the color burst. Resistors R3, R5, R7, R10, R13, and R15 adjust the signals to the proper amplitudes, and a tank circuit (L3 and C32) resonant at 3.58 MHz conditions the color burst.

The resulting video signal is an NTSC-compatible composite-video signal that can be displayed on a standard video monitor. The signal is similar to the EIA (Electronic Industries Association) standard positive composite video (see Table 7-15). This signal is available in two places in the Apple IIe:

- At the phono jack on the back of the Apple IIe. The sleeve of this jack is connected to ground and the tip is connected to the video output through a resistor network that attenuates it to about 1 volt and matches its impedance to 75 ohms.
- At the internal video connector on the Apple IIe circuit board near the RCA jack, J13 in Figure 7-13c. It is made up of four Molex-type pins, 0.25 inches tall, on 0.10 inch centers. This connector carries the video signal, ground, and two power supplies, as shown in Table 7-15.

*Table 7-15. Internal Video Connector Signals*

*Note: Pin 1 is the pin closest to the keyboard; pin 4 is at the back.*

Pin	Name	Description
1	GROUND	System common ground
2	VIDEO	NTSC-compatible positive composite video. White level is about 2.0 volts, black level is about 0.75 volts, and sync level is 0.0 volts. This output is not protected against short-circuits.
3	-5V	-5 volt power supply
4	+12V	+12 volt power supply

## Built-in I/O Circuits

The use of the Apple IIe's built-in I/O features is described in Chapter 2. This section describes the hardware implementation of all of those features except the video display described in the previous sections.

The IOU (Input/Output Unit) directly generates the output signals for the speaker, the cassette interface, and the annunciators. The other I/O features are handled by smaller ICs, as described later in this section.

The addresses of the built-in I/O features are described in Chapter 2 and listed in Table 2-2, Table 2-11, and Table 2-12. All of the built-in I/O features except the displays use memory locations between \$C000 and \$C070 (decimal 49152 and 49264). The I/O address decoding is performed by three ICs: a 74LS138, a 74LS154, and a 74LS251.

The 74LS138 decodes address lines A8, A9, A10, and A11 to select address pages on 256-byte boundaries starting at \$C000 (decimal 49152). When it detects addresses between \$C000 and \$C0FF, it enables the IOU and the 74LS154. The 74LS154 in turn decodes address lines A4, A5, A6, and A7 to select 16-byte address areas between \$C000 and \$C0FF. Addresses between \$C060 and \$C06F enable the 74LS251 that multiplexes the hand control switches and paddles; addresses between \$C070 and \$C07F reset the NE558 quadruple timer that interfaces to the hand controls, as described later in the section "Game I/O Signals."

## The Keyboard

The Apple IIe's keyboard is a matrix of keyswitches connected to an AY-3600-type keyboard decoder via a ribbon cable and a 26-pin connector. The AY-3600 scans the array of keys over and over to detect any keys pressed. The scanning rate is set by the external resistor-capacitor network made up of C70 and R32. The debounce time is also set externally, by C71.

The AY-3600's outputs include five bits of key code plus separate lines for **CONTROL**, **SHIFT**, any-key-down, and keyboard strobe. The any-key-down and keyboard-strobe lines are connected to the IOU, which addresses them as soft switches. The key-code lines, along with **CONTROL** and **SHIFT**, are inputs to a separate 2316 ROM. The ROM translates them to the character codes that are enabled onto the data bus by signals named KBD' and ENKBD'. The KBD' signal is enabled by the MMU whenever a program reads location \$C000, as described in the section "Reading the Keyboard" in Chapter 2.

*Table 7-16.* Keyboard Connector Signals

Pin Number	Name	Description
1,2,4,6,8,10, 23,25,12,22	Y0-Y9	Y-direction key-matrix connections
3	+5	+5 volt supply
5,7,9,15	n.c.	
1	LCNTL'	Line from [CONTROL] key
13	GND	System common ground
14,16,20,21, 19,26,17	X0-X7	X-direction key-matrix connections
24	LSHFT'	Line from [SHIFT] key

### Connecting a Keypad

There is a smaller connector wired in parallel with the keyboard connector. You can connect a ten-key numeric pad to the Apple IIe via this connector.

*Table 7-17.* Keypad Connector Signals

Pin Number	Name	Description
1,2,5,3,4,6	Y0-Y5	Y-direction key-matrix connections
7	n.c.	
9,11,10,8	X4-X7	X-direction key-matrix connections

## Cassette I/O

The two miniature phone jacks on the back of the Apple IIe are used to connect an audio cassette recorder for saving programs. The output signal to the cassette recorder comes from a pin on the IOU via resistor network R6 and R9, which attenuates the signal to a level appropriate for the recorder's microphone input. Input from the recorder is amplified and conditioned by a type 741 operational amplifier and sent to one of the inputs of the 74LS251 input multiplexer.

The signal specifications for cassette I/O are

- Input: 1 volt (nominal) from recorder earphone or monitor output. Input impedance is 12K ohms.
- Output: 25 millivolts to recorder microphone input. Output impedance is 100 ohms.

## The Speaker

The Apple IIe's built-in loudspeaker is controlled by a single bit of output from the IOU (Input Output Unit). The signal from the IOU is AC coupled to Q5, an MPSA13 Darlington transistor amplifier. The speaker connector is a Molex KK100 connector, J18 in Figure 7-13b, with two square pins 0.25 inches tall and on 0.10-inch centers.

A light-emitting diode is connected in parallel across the speaker pins such that, when the speaker is not connected, the diode glows whenever the speaker signal is on. This diode is used as a diagnostic indicator during assembly and testing of the Apple IIe.

*Table 7-18. Speaker Connector Signals*

Pin Number	Name	Description
1	SPKR	Speaker signal. This line will deliver about 0.5 watts into an 8-ohm speaker.
2	+5	+5V power supply. Note that the speaker is not connected to system ground.

## Game I/O Signals

Several I/O signals that are individually controlled via soft switches are collectively referred to as the game signals. Even though they are normally used for hand controls, these signals can be used for other simple I/O applications. There are five output signals: the four annunciations, numbered A0 through A3, and one strobe output. There are three one-bit inputs, called *switches* and numbered SW0 through SW2, and four analog inputs, called *paddles* and numbered PDL0 through PDL3.

The annunciation outputs are driven directly by the IOU (Input Output Unit). These outputs can drive one TTL (transistor-transistor logic) load each; for heavier loads, you must use a transistor or a TTL buffer on these outputs. These signals are only available on the 16-pin internal connector. (See Table 7-19.)

The strobe output is a pulse transmitted any time a program reads or writes to location \$C040. The strobe pin is connected to one output of the 74LS154 address decoder. This TTL signal is normally high; it goes low during  $\phi_0$  of the instruction cycle that addresses location \$C040. This signal is only available on the 16-pin internal connector. (See Table 7-19.)

The game inputs are multiplexed along with the cassette input signal by a 74LS251 eight-input multiplexer enabled by the C06X' signal from the 74LS154 I/O address decoder. Depending on the low-order address, the appropriate game input is connected to bit 7 of the data bus.

The switch inputs are standard low-power Schottky TTL inputs. To use them, connect each one to 560-ohm pull-down resistors connected to the ground and through single-pole, momentary-contact pushbutton switches to the +5 volt supply.

The hand-control inputs are connected to the timing inputs of an NE558 quadruple 555-type analog timer. Addressing \$C07X sends a signal from the 74LS154 that resets all four timers and causes their outputs to go to 1 (high). A variable resistance of up to 150K ohms connected between one of these inputs and the +5V supply controls the charging time of one of four 0.022-microfarad capacitors. When the voltage on the capacitor passes a certain threshold, the output of the NE558 changes back to 0 (low). Programs can determine the setting of a variable resistor by resetting the timers and then counting time until the selected timer input changes from high to low. The resulting count is proportional to the resistance.

The game I/O signals are all available on a 16-pin DIP socket labelled GAME I/O on the main circuit board inside the case. The switches and the paddles are also available on a D-type miniature connector on the back of the Apple IIe; see J8 and J15 in Figure 7-13d.

*Table 7-19.* Game I/O Connector Signals

Internal-Connector Pin Number	Back-Panel Connector Pin Number	Signal Name	Description
1	2	+5V	+5V power supply. Total current drain from this pin must not exceed 100mA.
2,3,4	7,1,6	PB0-PB2	Switch inputs. These are standard 74LS inputs.
5	-	STROBE'	Strobe output. This line goes low during $\phi_0$ of a read or write instruction to location \$C040.
6,10,7,11	5,8,4,9	PDL0-PDL3	Hand control inputs. Each of these should be connected to a 150K-ohm variable resistor connected to +5V.
8	3	GND	System ground.
15,14,13,12	-	AN0-AN3	Annunciators. These are standard 74LS TTL outputs and must be buffered to drive other than TTL inputs.
9,16	-	n.c.	Nothing is connected to these pins.

Chapter 6 describes the standards for programming peripheral cards for the Apple IIe.

## Expanding the Apple IIe

The main circuit board of the Apple IIe has eight empty card connectors or slots on it. These slots make it possible to add features to the Apple IIe by plugging in peripheral cards with additional hardware. This section describes the hardware that supports them, including all of the signals available on the expansion slots.

### The Expansion Slots

The seven connectors lined up across the back part of the Apple IIe's main circuit card are the expansion slots, also called peripheral slots or simply slots, numbered from 1 to 7. They are 50-pin PC-card edge connectors with pins on 0.10-inch centers. A PC card plugged into one of these connectors has access to all of the signals necessary to perform input and output and to execute programs in RAM or ROM on the card. These signals are described briefly in Table 7-20. The following paragraphs describe the signals in general and mention a few points that are often overlooked. For further details, refer to the schematic diagram in Figures 7-13a, 7-13b, 7-13c, and 7-13d.

### The Peripheral Address Bus

The microprocessor's address bus is buffered by two 74LS244 octal three-state buffers. These buffers, along with a buffer in the microprocessor's R/W' line, are enabled by a signal derived from the DMA' daisy-chain on the expansion slots. Pulling the peripheral line DMA' low disables the address and R/W' buffers so that peripheral DMA circuitry can control the address bus. The DMA address and R/W' signals supplied by a peripheral card must be stable all during  $\phi_0$  of the instruction cycle, as shown in Figure 7-12.

Another signal that can be used to disable normal operation of the Apple IIe is INH'. Pulling INH' low disables all of the memory in the Apple IIe except the part in the I/O space from \$C000 to \$CFFF. A peripheral card that uses either INH' or DMA' must observe proper timing; in order to disable RAM and ROM cleanly, the disabling signal must be stable all during  $\phi_0$  of the instruction cycle (refer to the timing diagram in Figure 7-12).

The peripheral devices should use I/O SELECT' and DEVICE SELECT' as enables. Most peripheral ICs require their enable signals to be present for a certain length of time before data is strobed into or out of the device. Remember that I/O SELECT' and DEVICE SELECT' are only asserted during  $\phi_0$  high.

### **The Peripheral Data Bus**

The Apple IIe has two versions of the microprocessor data bus: an internal bus, MD0-MD7, connected directly to the microprocessor; and an external bus, D0-D7, driven by a 74LS245 octal bidirectional bus buffer. The 65C02 is fabricated with MOS circuitry, so it can drive capacitive loads of up to about 130 pF. If peripheral cards are installed in all seven slots, the loading on the data bus can be as high as 500 pF, so the 74LS245 drives the data bus for the peripheral cards. The same argument applies if you use MOS devices on peripheral cards: they don't have enough drive for the fully-loaded bus, so you should add buffers.

### **Loading and Driving Rules**

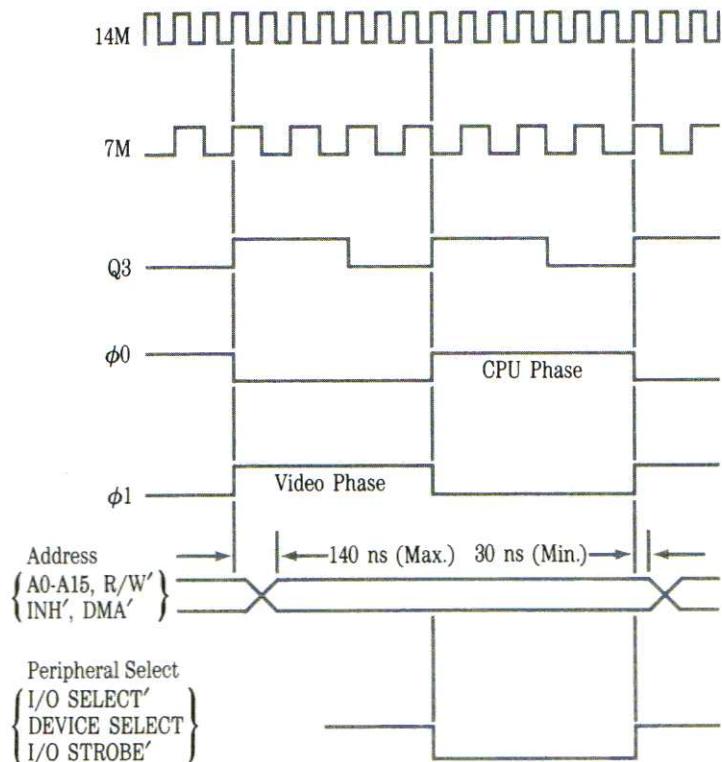
Table 7-20 shows the drive requirements and loading limits for each pin on the expansion slots. The address bus, the data bus, and the R/W' line should be driven by three-state buffers. Remember that there is considerable distributed capacitance on these busses and that you should plan on tolerating the added load of up to six additional peripheral cards. MOS devices such as PIAs and ACIAs cannot switch such heavy capacitive loads. Connecting such devices directly to the bus will lead to possible timing and level errors.

### **Interrupt and DMA Daisy Chains**

The interrupt requests (IRQ' and NMI') and the direct-memory access (DMA') signal are available at all seven expansion slots. A peripheral card requests an interrupt or a DMA transfer by pulling the appropriate output line low (active). If two peripheral cards request an interrupt or a DMA transfer at the same time, they will contend for the data and address busses. To prevent this, two pairs of pins on each connector are wired as a priority daisy chain. The daisy-chain pins for interrupts are INT IN and INT OUT, and the pins for DMA are DMA IN and DMA OUT, as shown for J1-J7 in Figure 7-13d.

Each daisy chain works like this: the output from each connector goes to the input of the next higher numbered one. For these signals to be useful for cards in lower numbered connectors, all of the higher numbered connectors must have cards in them, and all of those cards must connect DMA IN to DMA OUT and INT IN to INT OUT. Whenever a peripheral card uses pin DMA', it must do so only if its DMA IN line is active, and it must disable its DMA OUT line while it is using DMA'. The INT IN and INT OUT lines must be used the same way: enable the card's interrupt circuits with INT IN, and disable INT OUT whenever IRQ' or NMI' is being used.

*Figure 7-12. Peripheral-Signal Timing*



*Table 7-20.* Expansion Slot Signals

Pin	Name	Description
1	I/O SELECT	Normally high; goes low during $\phi_0$ when the 65C02 addresses location \$CnXX, where n is the connector number. This line can drive 10 LS TTL loads.*
2-17	A0-A15	Three-state address bus. The address becomes valid during $\phi_1$ and remains valid during $\phi_0$ . Each address line can drive 5 LS TTL loads.*
18	R/W'	Three-state read/write line. Valid at the same time as the address bus; high during a read cycle, low during a write cycle. It can drive 2 LS TTL loads.*
19	SYNC'	Composite horizontal and vertical sync, on expansion slot 7 <i>only</i> . This line can drive 2 LS TTL loads.*
20	I/O STROBE'	Normally high; goes low during $\phi_0$ when the 65C02 addresses a location between \$C800 and \$CFFF. This line can drive 4 LS TTL loads.
21	RDY	Input to the 65C02. Pulling this line low during $\phi_1$ halts the 65C02 with the address bus holding the address of the location currently being fetched. This line has a 3300 ohm pullup resistor to +5V.
22	DMA'	Input to the address bus buffers. Pulling this line low during $\phi_1$ disconnects the 65C02 from the address bus. This line has a 3300 ohm pullup resistor to +5V.
23	INT OUT	Interrupt priority daisy-chain output. Usually connected to pin 28 (INT IN).†
24	DMA OUT	DMA priority daisy-chain output. Usually connected to pin 22 (DMA IN).
25	+5V	+5-volt power supply. A total of 500mA is available for all peripheral cards.
26	GND	System common ground.
27	DMA IN	DMA priority daisy-chain input. Usually connected to pin 24 (DMA OUT).
28	INT IN	Interrupt priority daisy-chain input. Usually connected to pin 23 (INT OUT).
29	NMI'	Non-maskable interrupt to 65C02. Pulling this line low starts an interrupt cycle with the interrupt-handling routine at location \$03FB. This line has a 3300 ohm pullup resistor to +5V.

*Table 7-20—Continued.* Expansion Slot Signals

Pin	Name	Description
30	IRQ'	Interrupt request to 65C02. Pulling this line low starts an interrupt cycle only if the interrupt-disable (I) flag in the 65C02 is not set. Uses the interrupt-handling routine at location \$03FE. This line has a 3300 ohm pullup resistor to +5V.
31	RES'	Pulling this line low initiates a reset routine, as described in Chapter 4.
32	INH'	Pulling this line low during $\phi_1$ inhibits (disables) the memory on the main circuit board. This line has a 3300 ohm pullup resistor to +5V.
33	-12V	-12 volt power supply. A total of 200mA is available for all peripheral cards.
34	-5V	-5 volt power supply. A total of 200mA is available for all peripheral cards.
35	3.58M	3.58 MHz color reference signal, on slot 7 <i>only</i> . This line can drive 2 LS TTL loads.*
36	7M	System 7 MHz clock. This line can drive 2 LS TTL loads.*
37	Q3	System 2 MHz asymmetrical clock. This line can drive 2 LS TTL loads.*
38	$\phi_1$	65C02 phase 1 clock. This line can drive 2 LS TTL loads.*
39	$\mu$ PSYNC	The 65C02 signals an operand fetch by driving this line high during the first read cycle of each instruction.
40	$\phi_0$	65C02 phase 0 clock. This line can drive 2 LS TTL loads.*
41	DEVICE SELECT'	Normally high; goes low during $\phi_0$ when the 65C02 addresses location \$C0nX, where n is the connector number plus 8. This line can drive 10 LS TTL loads.*
42-49	D0-D7	Three-state buffered bi-directional data bus. Data becomes valid during $\phi_0$ high and remains valid until $\phi_0$ goes low. Each data line can drive one LS TTL load.*
50	+12V	+12 volt power supply. A total of 250mA is available for all peripheral cards.

\* Loading limits are for each card.

† On slot 7 *only*, this pin can be connected to the graphics-mode signal GR: see text for details.

### **Auxiliary Slot**

The large connector at the left side of the Apple IIe's main circuit card is the auxiliary slot. It is a 60-pin PC-card edge connector with pins on 0.10-inch centers. A PC card plugged into this connector has access to all of the signals used in producing the video display. These signals are described briefly in Table 7-21. For further details, refer to the schematic diagram in Figures 7-13a, 7-13b, 7-13c, and 7-13d.

Many of the internal signals that are not available on the expansion slots are on the auxiliary slot. By using both kinds of connectors, manufacturing and repair personnel can gain access to most of the signals needed for diagnosing problems in the Apple IIe.

### **80-Column Display Signals**

The additional memory needed for producing an 80-column text display is on the 80-column text card, along with the buffers that transfer the data to the video data bus, as described earlier in this chapter in the section "Text Displays." The signals that control the 80-column text data include the system clocks  $\phi_0$  and  $\phi_1$ , the multiplexed RAM address RA0-RA7, the RAM address-strobe signals PRAS' and PCAS', and the auxiliary-RAM enable signals, EN80' and R/W80. The EN80' enable signal is controlled by the 80STORE soft switch described in Chapter 4. Data is sent to the auxiliary memory via the internal data bus MD0-MD7; the data is transferred to the video generator via the video data bus VID0-VID7.

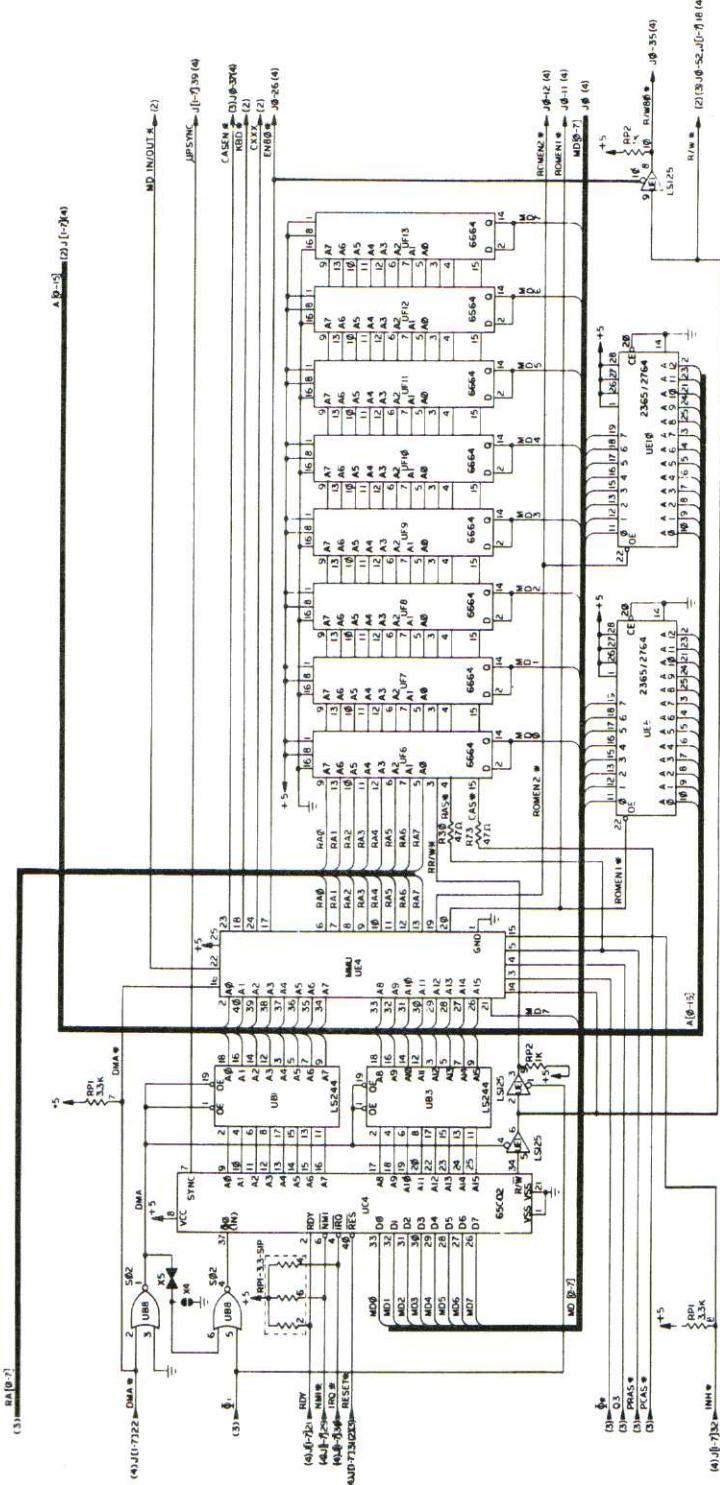
*Table 7-21.* Auxiliary Slot Signals

Pin	Name	Description
1	3.58M	3.58 MHz video color reference signal. This line can drive two LS TTL loads.
2	VID7M	Clocks the video dots out of the 74166 parallel-to-serial shift register. This line can drive two LS TTL loads.
3	SYNC'	Video horizontal and vertical sync signal. This line can drive two LS TTL loads.
4	PRAS'	Multiplexed RAM row-address strobe. This line can drive two LS TTL loads.
5	VC	Third low-order vertical-counter bit. This line can drive two LS TTL loads.
6	C07X'	Hand-control reset signal. This line can drive two LS TTL loads.
7	WNDW'	Video non-blank window. This line can drive two LS TTL loads.
8	SEGA	First low-order vertical counter bit. This line can drive two LS TTL loads.
51,10,49,48, 13,14,46,9	RA0-RA7	Multiplexed RAM-address bus. This line can drive two LS TTL loads.
11,12	ROMEN1, ROMEN2	Enable signals for the ROMs on main circuit board.
44,43,40,39, 21,20,17,16	MD0-MD7	Internal (unbuffered) data bus. This line can drive two LS TTL loads.
45,42,41,38, 22,19,18,15	VID0-VID7	Video data bus. This three-state bus carries video data to the character generator.
23	$\phi_0$	65C02 clock phase 0. This line can drive two LS TTL loads.
24	CLRGAT'	Color-burst gating signal. This line can drive two LS TTL loads.
25	80VID'	Enables 80-column display timing. This line can drive two LS TTL loads.
26	EN80'	Enable for auxiliary RAM. This line can drive two LS TTL loads.
27	ALTVID'	Alternative video output to the video summing amplifier.
28	SEROUT'	Video serial output from 74166 parallel-to-serial shift register.
29	ENVID'	Normally low; driving this line high disables the character generator such that the video dots from the shift register are all high (white), and alternative video can be sent out via ALTVID'. This line has a 1000 ohm pulldown resistor to ground.

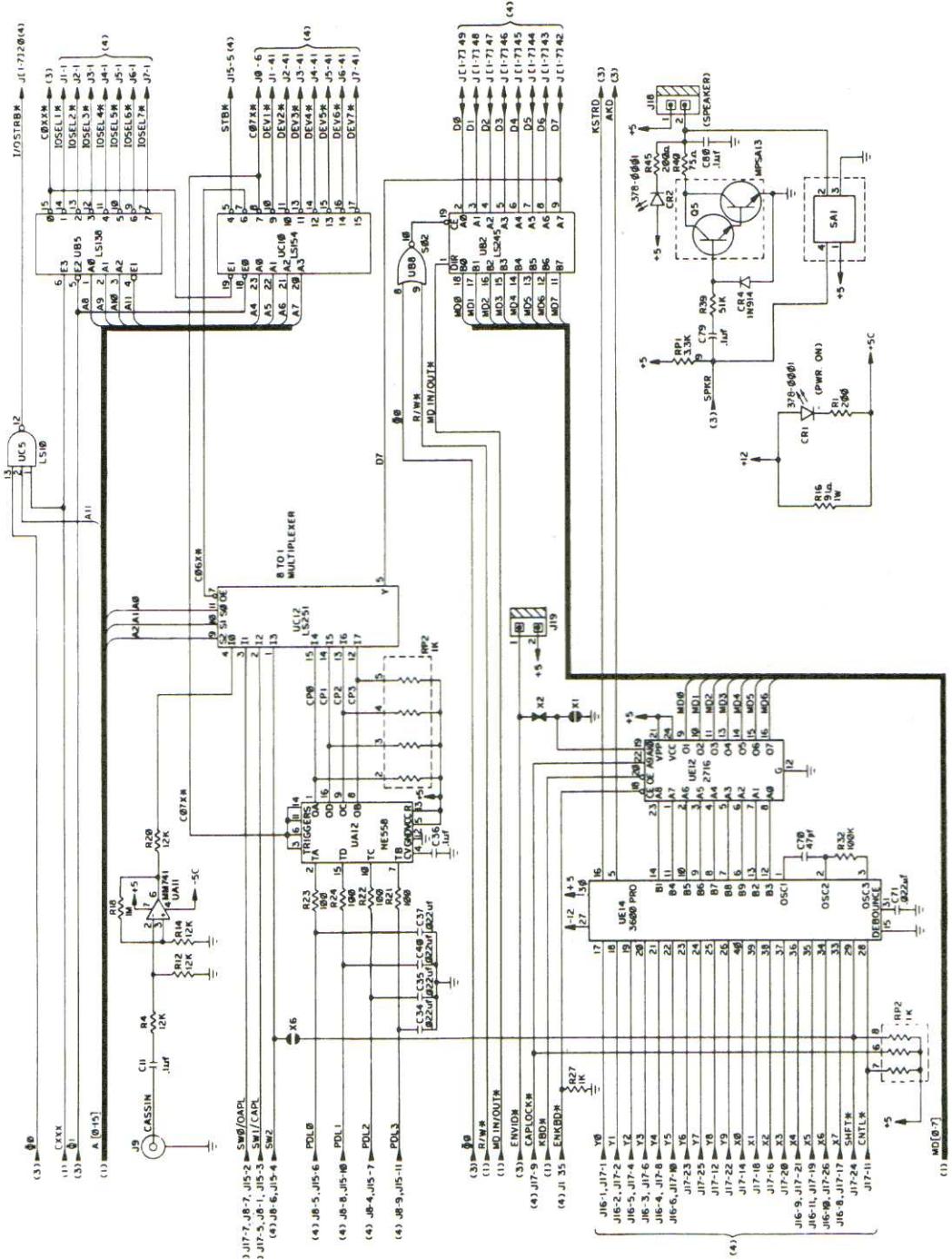
*Table 7-21—Continued.* Auxiliary Slot Signals

Pin	Name	Description
30	+5	+5 volt power supply.
31	GND	System common ground.
32	14M	14.3 MHz master clock signal. This line can drive two LS TTL loads.
33	PCAS'	Multiplexed column-address strobe. This line can drive two LS TTL loads.
34	LDPS'	Strobe to video parallel-to-serial shift register. This signal goes low to load the contents of the video data bus into the shift register. This line can drive two LS TTL loads.
35	R/W80	Read/write signal for RAM on the card in this slot. This line can drive two LS TTL loads.
36	ϕ1	65C02 clock phase 1. This line can drive two LS TTL loads.
37	CASEN'	Column-address enable. This signal is disabled (held high) during accesses to memory on the card in this slot. This line can drive two LS TTL loads.
47	H0	Low-order horizontal byte counter. This line can drive two LS TTL loads.
50	AN3	Output of annunciator number 3. This line can drive two LS TTL loads.
52	R/W'	65C02 read/write signal. This line can drive two LS TTL loads.
53	Q3	2 MHz asymmetrical clock. This line can drive two LS TTL loads.
54	SEGB	Second low-order vertical-counter bit. This line can drive two LS TTL loads.
55	FRCTXT'	Normally high; pulling this line low enables 14MHz video output even when GR is active.
56,57	RA9',RA10'	Character-generator control signals from the IOU. This line can drive two LS TTL loads.
58	GR	Graphics-mode enable signal. This line can drive two LS TTL loads.
59	7M	7 MHz timing signal. This line can drive two LS TTL loads.
60	ENTMG'	Normally low; pulling this line high disables the master timing from the PAL device. This line has a 1000 ohm pulldown resistor to ground.

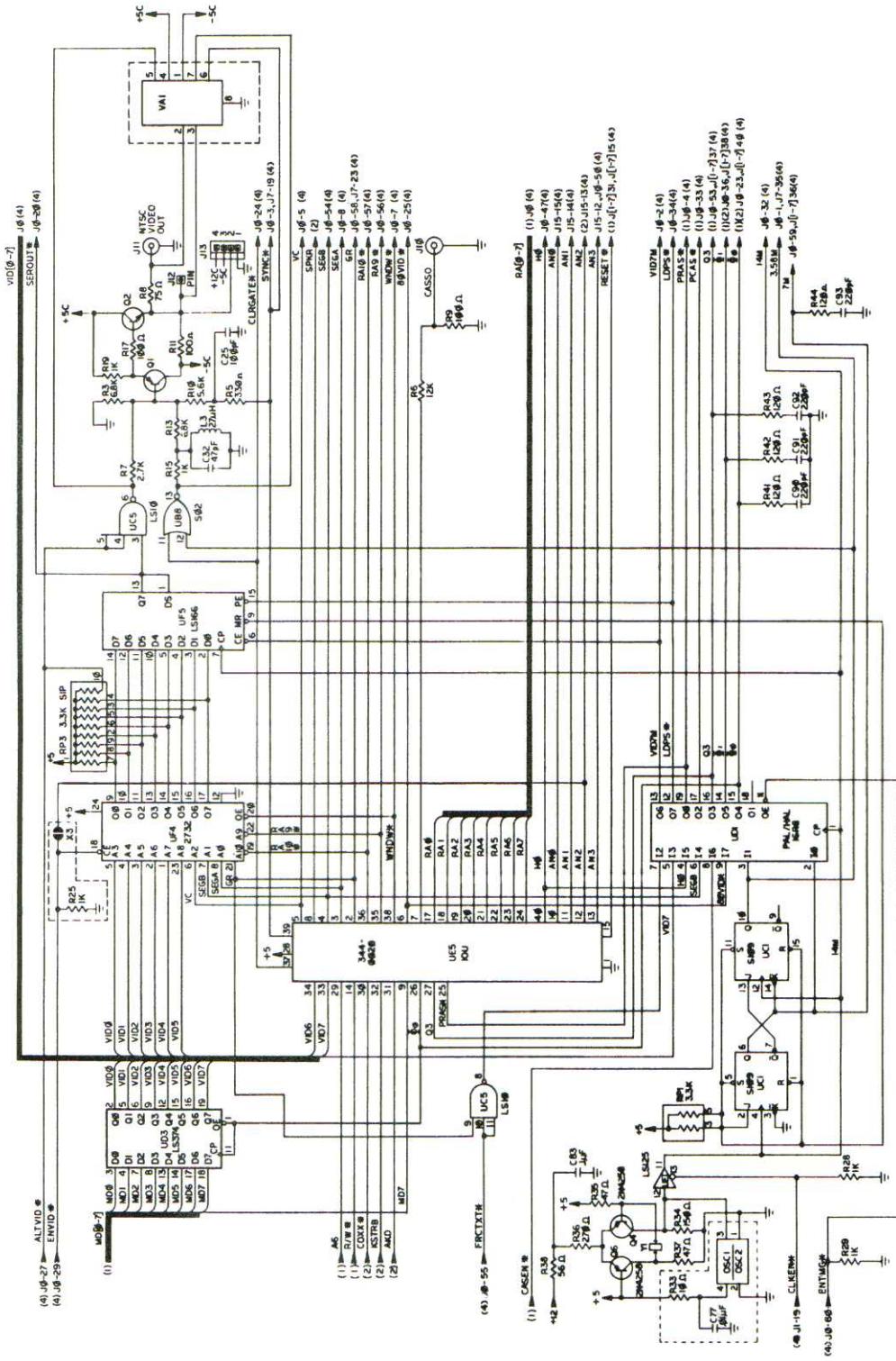
*Figure 7-13a.* Schematic Diagram, Part 1



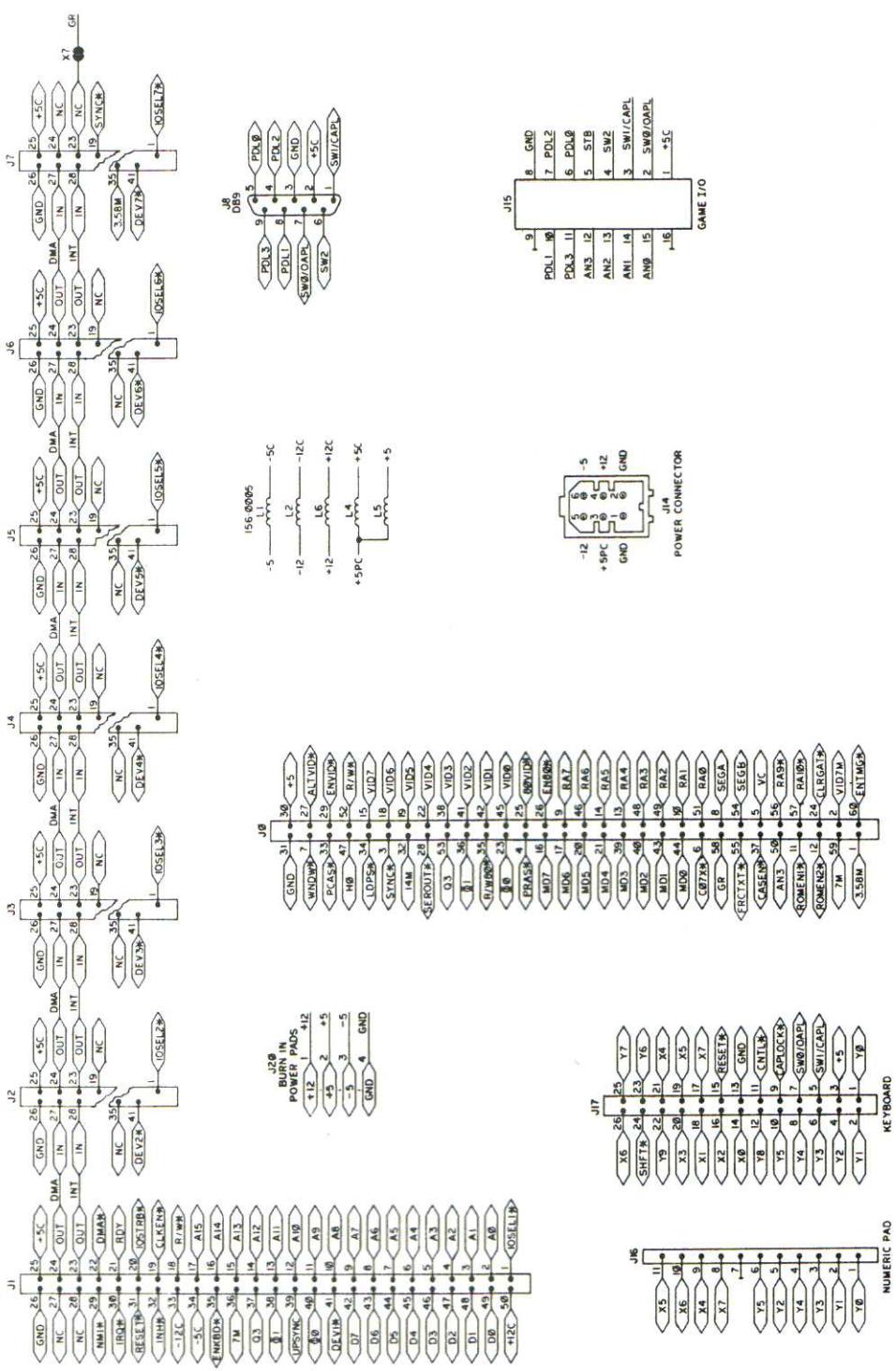
*Figure 7-13b.* Schematic Diagram, Part 2



*Figure 7-13c.* Schematic Diagram, Part 3



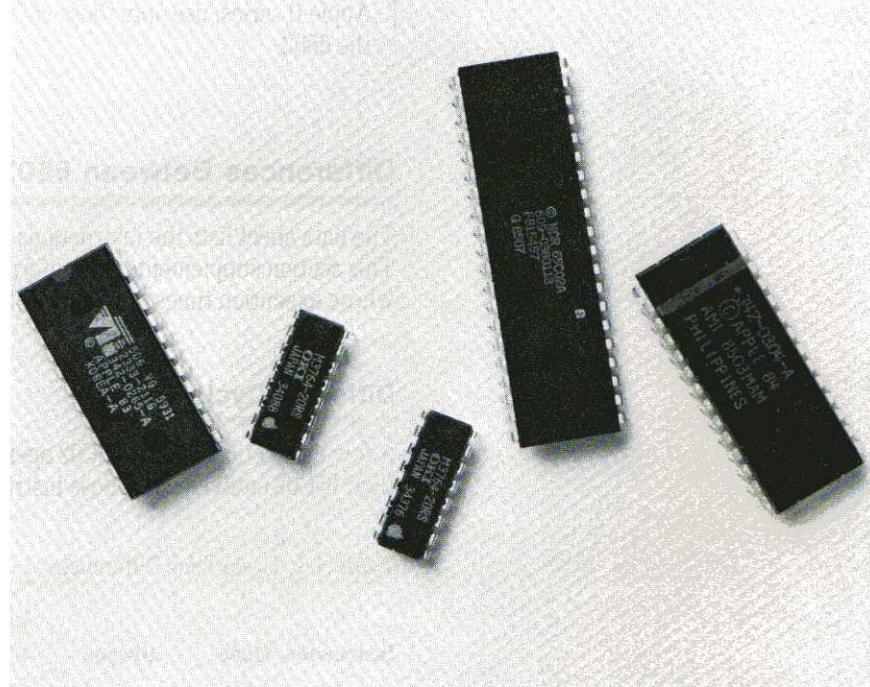
*Figure 7-14d.* Schematic Diagram, Part 4





## Appendix A

### The 65C02 Microprocessor



This appendix contains a description of the differences between the 6502 and the 65C02 microprocessors. It also contains the data sheet for the 65C02 microprocessor.

The 6502 microprocessor was used in the original Apple IIe, Apple II Plus, and Apple II. The 65C02 is a 6502 that uses less power and has ten new instructions and two new addressing modes. The 65C02 is used in both the enhanced Apple IIe and the Apple IIc.

In the data sheet tables, execution times are specified in number of cycles. One cycle time for the Apple IIe equals 0.978 microseconds, giving a system clock rate of about 1.02 MHz.

*Note:* If you want to write programs that execute on all computers in the Apple II series, use only those 65C02 instructions that are also present on the 6502.

## Differences Between 6502 and 65C02

The data sheet lists the instructions and addressing modes of the 65C02. This section supplements that information by listing those instructions whose execution times or results differ in the 6502 and the 65C02.

### Different Cycle Times

A few instructions on the 65C02 operate in different numbers of cycles than their 6502 equivalents. These instructions are listed in Table A-1.

*Table A-1.* Cycle Time Differences

Instruction/Mode	Opcode	6502 Cycles	65C02 Cycles
ASL Absolute, X	1E	7	6
DEC Absolute, X	DE	7	6
INC Absolute, X	FE	7	6
JMP (Absolute)	6C	5	6
LSR Absolute, X	5E	7	6
ROL Absolute, X	3E	7	6
ROR Absolute, X	7E	7	6

## Different Instruction Results

It is important to note that the BIT instruction when used in immediate mode (opcode \$89) leaves processor status register bits 7 (N) and 6 (V) unchanged on the 65C02. On the 6502, all modes of the BIT instruction have the same effect on the status register: the value of memory bit 7 is placed in status bit 7, and memory bit 6 is placed in status bit 6.

Also note that if the JMP indirect instruction (code \$6C) references an indirect address location that spans a page boundary, the 65C02 fetches the high-order byte of the effective address from the first byte of the next page, while the 6502 fetches it from the first byte of the current page. For example, JMP (\$02FF) gets ADL from location \$02FF on both processors. But on the 65C02, ADH comes from \$0300; on the 6502, ADH comes from \$0200.

## Data Sheet

The remaining pages of this appendix are copyright 1982, NCR Corporation, Dayton, Ohio, and are reprinted with their permission.



## NCR65C02

### ■ GENERAL DESCRIPTION

The NCR CMOS 6502 is an 8-bit microprocessor which is software compatible with the NMOS 6502. The NCR65C02 hardware interfaces with all 6500 peripherals. The enhancements include ten additional instructions, expanded operational codes and two new addressing modes. This microprocessor has all of the advantages of CMOS technology: low power consumption, increased noise immunity and higher reliability. The CMOS 6502 is a low power high performance microprocessor with applications in the consumer, business, automotive and communications market.

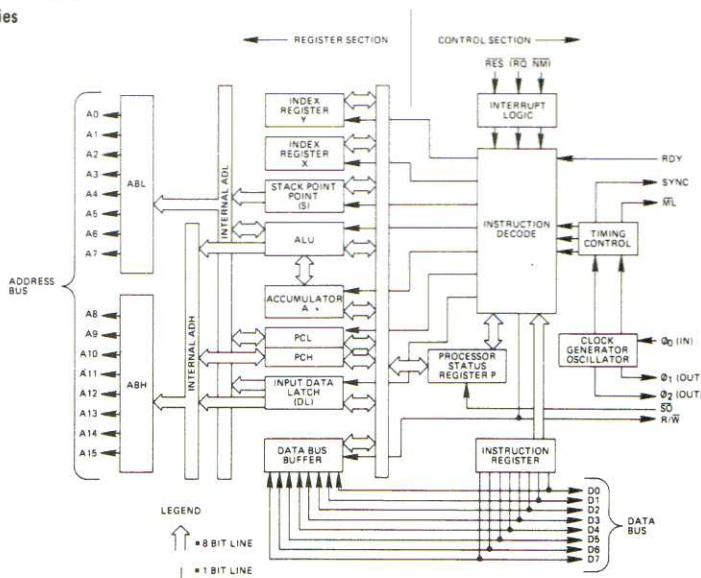
### ■ FEATURES

- Enhanced software performance including 27 additional OP codes encompassing ten new instructions and two additional addressing modes.
- 66 microprocessor instructions.
- 15 addressing modes.
- 178 operational codes.
- 1MHz, 2MHz operation.
- Operates at frequencies as low as 200 Hz for even lower power consumption (pseudo-static: stop during  $\Phi_2$  high).
- Compatible with NMOS 6500 series microprocessors.
- 64 K-byte addressable memory.
- Interrupt capability.
- Lower power consumption. 4mA @ 1MHz.
- +5 volt power supply.
- 8-bit bidirectional data bus.
- Bus Compatible with M6800.
- Non-maskable interrupt.
- 40 pin dual-in-line packaging.
- 8-bit parallel processing
- Decimal and binary arithmetic.
- Pipeline architecture.
- Programmable stack pointer.
- Variable length stack.
- Optional internal pullups for (RDY, IRQ, SO, NMI and RES)
- Specifications are subject to change without notice.

### ■ PIN CONFIGURATION

VSS	1	RES
RDY	2	$\Phi_2$ (OUT)
$\Phi_1$ (OUT)	3	SO
IRQ	4	$\Phi_0$ (IN)
ML	5	NC
NMI	6	NC
SYNC	7	NC
VDD	8	R/W
A0	9	D0
A1	10	D1
A2	11	D2
A3	12	D3
A4	13	D4
A5	14	D5
A6	15	D6
A7	16	D7
A8	17	A15
A9	18	A14
A10	19	A13
A11	20	A12
	21	VSS

### ■ NCR65C02 BLOCK DIAGRAM



## NCR65C02

### ■ ABSOLUTE MAXIMUM RATINGS:

RATING	SYMBOL	VALUE	UNIT
SUPPLY VOLTAGE	V <sub>DD</sub>	-0.3 to +7.0	V
INPUT VOLTAGE	V <sub>IN</sub>	-0.3 to +7.0	V
OPERATING TEMP.	T <sub>A</sub>	0 to +70	°C
STORAGE TEMP.	T <sub>STG</sub>	-55 to +150	°C

### ■ PIN FUNCTION

PIN	FUNCTION
A0 - A15	Address Bus
D0 - D7	Data Bus
IRQ *	Interrupt Request
RDY *	Ready
ML	Memory Lock
NMI *	Non-Maskable Interrupt
SYNC	Synchronize
RES *	Reset
SO *	Set Overflow
NC	No Connection
R/W	Read/Write
VDD	Power Supply (+5V)
VSS	Internal Logic Ground
Q <sub>0</sub>	Clock Input
Q <sub>1</sub> , Q <sub>2</sub>	Clock Output

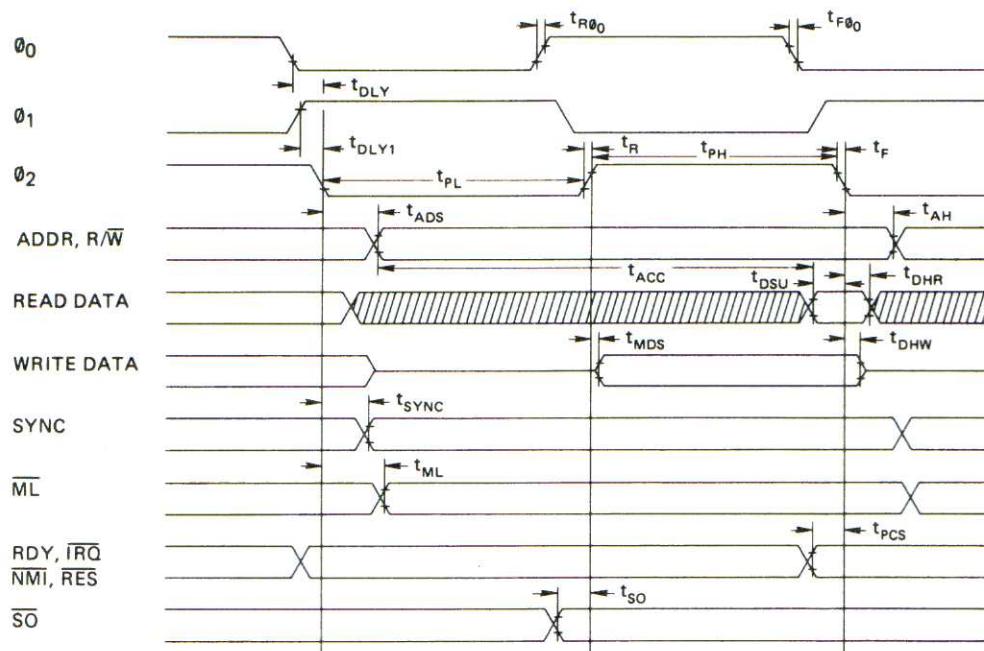
\*This pin has an optional internal pullup for a No Connect condition.

### ■ DC CHARACTERISTICS

	SYMBOL	MIN.	TYP.	MAX	UNIT
Input High Voltage Q <sub>0</sub> (IN)	V <sub>IH</sub>	V <sub>SS</sub> + 2.4	—	V <sub>DD</sub>	V
Input High Voltage RES, NMI, RDY, IRQ, Data, S.O.		V <sub>SS</sub> + 2.0	—	—	V
Input Low Voltage Q <sub>0</sub> (IN) RES, NMI, RDY, IRQ, Data, S.O.	V <sub>IL</sub>	V <sub>SS</sub> -0.3 —	—	V <sub>SS</sub> + 0.4 V <sub>SS</sub> + 0.8	V
Input Leakage Current (V <sub>IN</sub> = 0 to 5.25V, V <sub>DD</sub> = 5.25V) With pullups Without pullups	I <sub>IN</sub>	-30 —	—	+30 +1.0	μA
Three State (Off State) Input Current (V <sub>IN</sub> = 0.4 to 2.4V, V <sub>CC</sub> = 5.25V) Data Lines	I <sub>TSI</sub>	—	—	10	μA
Output High Voltage (I <sub>OH</sub> = -100 μAdc, V <sub>DD</sub> = 4.75V SYNC, Data, A0-A15, R/W)	V <sub>OH</sub>	V <sub>SS</sub> + 2.4	—	—	V
Out Low Voltage (I <sub>OL</sub> = 1.6mA, V <sub>DD</sub> = 4.75V SYNC, Data, A0-A15, R/W)	V <sub>OL</sub>	—	—	V <sub>SS</sub> + 0.4	V
Supply Current f = 1MHz Supply Current f = 2MHz	I <sub>DD</sub>	—	—	4 8	mA
Capacitance (V <sub>IN</sub> = 0, T <sub>A</sub> = 25°C, f = 1MHz) Logic Data A0-A15, R/W, SYNC Q <sub>0</sub> (IN)	C C <sub>IN</sub> C <sub>OUT</sub> C <sub>Q0</sub> (IN)	— — — —	— — — —	5 10 10 10	pF

## NCR65C02

### ■ TIMING DIAGRAM



Note: All timing is referenced from a high voltage of 2.0 volts and a low voltage of 0.8 volts.

### ■ NEW INSTRUCTION MNEMONICS

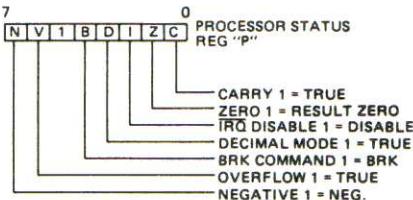
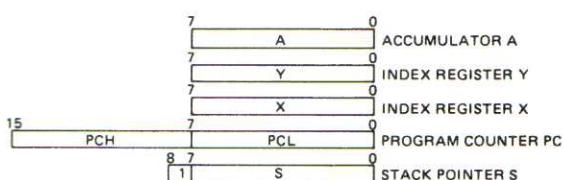
HEX	MNEMONIC	DESCRIPTION
80	BRA	Branch relative always [Relative]
3A	DEA	Decrement accumulator [Accum]
1A	INA	Increment accumulator [Accum]
DA	PHX	Push X on stack [Implied]
5A	PHY	Push Y on stack [Implied]
FA	PLX	Pull X from stack [Implied]
7A	PLY	Pull Y from stack [Implied]
9C	STZ	Store zero [Absolute]
9E	STZ	Store zero [ABS, X]
64	STZ	Store zero [Zero page]
74	STZ	Store zero [ZPG,X]
1C	TRB	Test and reset memory bits with accumulator [Absolute]
14	TRB	Test and reset memory bits with accumulator [Zero page]
0C	TSB	Test and set memory bits with accumulator [Absolute]
04	TSB	Test and set memory bits with accumulator [Zero page]

### ■ ADDITIONAL INSTRUCTION ADDRESSING MODES

HEX	MNEMONIC	DESCRIPTION
72	ADC	Add memory to accumulator with carry [(ZPG)]
32	AND	"AND" memory with accumulator [(ZPG)]
3C	BIT	Test memory bits with accumulator [ABS, X]
34	BIT	Test memory bits with accumulator [ZPG, X]
D2	CMP	Compare memory and accumulator [(ZPG)]
52	EOR	"Exclusive Or" memory with accumulator [(ZPG)]
7C	JMP	Jump (New addressing mode) [ABS(IND,X)]
B2	LDA	Load accumulator with memory [(ZPG)]
12	ORA	"OR" memory with accumulator [(ZPG)]
F2	SBC	Subtract memory from accumulator with borrow [(ZPG)]
92	STA	Store accumulator in memory [(ZPG)]

## NCR65C02

### ■ MICROPROCESSOR PROGRAMMING MODEL



### ■ FUNCTIONAL DESCRIPTION

#### Timing Control

The timing control unit keeps track of the instruction cycle being monitored. The unit is set to zero each time an instruction fetch is executed and is advanced at the beginning of each phase one clock pulse for as many cycles as is required to complete the instruction. Each data transfer which takes place between the registers depends upon decoding the contents of both the instruction register and the timing control unit.

#### Program Counter

The 16-bit program counter provides the addresses which step the microprocessor through sequential instructions in a program.

Each time the microprocessor fetches an instruction from program memory, the lower byte of the program counter (PCL) is placed on the low-order bits of the address bus and the higher byte of the program counter (PCH) is placed on the high-order 8 bits. The counter is incremented each time an instruction or data is fetched from program memory.

#### Instruction Register and Decode

Instructions fetched from memory are gated onto the internal data bus. These instructions are latched into the instruction register, then decoded, along with timing and interrupt signals, to generate control signals for the various registers.

#### Arithmetic and Logic Unit (ALU)

All arithmetic and logic operations take place in the ALU including incrementing and decrementing internal registers (except the program counter). The ALU has no internal memory and is used only to perform logical and transient numerical operations.

#### Accumulator

The accumulator is a general purpose 8-bit register that stores the results of most arithmetic and logic operations, and in addition, the accumulator usually contains one of the two data words used in these operations.

#### Index Registers

There are two 8-bit index registers (X and Y), which may be used to count program steps or to provide an index value to be used in generating an effective address.

When executing an instruction which specifies indexed addressing, the CPU fetches the op code and the base address, and modifies the address by adding the index register to it prior to performing the desired operation. Pre- or post-indexing of indirect addresses is possible (see addressing modes).

#### Stack Pointer

The stack pointer is an 8-bit register used to control the addressing of the variable-length stack on page one. The stack pointer is automatically incremented and decremented under control of the microprocessor to perform stack manipulations under direction of either the program or interrupts (NMI and IRQ). The stack allows simple implementation of nested subroutines and multiple level interrupts. The stack pointer should be initialized before any interrupts or stack operations occur.

#### Processor Status Register

The 8-bit processor status register contains seven status flags. Some of the flags are controlled by the program, others may be controlled both by the program and the CPU. The 6500 instruction set contains a number of conditional branch instructions which are designed to allow testing of these flags (see microprocessor programming model).

## NCR65C02

### ■ AC CHARACTERISTICS V<sub>DD</sub> = 5.0V ± 5%, T<sub>A</sub> = 0°C to 70°C, Load = 1 TTL + 130 pF

Parameter	Symbol	1MHz		2MHz		3MHz		Unit
		Min	Max	Min	Max	Min	Max	
Delay Time, $\theta_0$ (IN) to $\theta_2$ (OUT)	$t_{DLY}$	—	60	—	60	20	60	nS
Delay Time, $\theta_1$ (OUT) to $\theta_2$ (OUT)	$t_{DLY1}$	-20	20	-20	20	-20	20	nS
Cycle Time	$t_{CYC}$	1.0	5000*	0.50	5000*	0.33	5000*	μs
Clock Pulse Width Low	$t_{PL}$	460	—	220	—	160	—	nS
Clock Pulse Width High	$t_{PH}$	460	—	220	—	160	—	nS
Fall Time, Rise Time	$t_F, t_R$	—	25	—	25	—	25	nS
Address Hold Time	$t_{AH}$	20	—	20	—	0	—	nS
Address Setup Time	$t_{ADS}$	—	225	—	140	—	110	nS
Access Time	$t_{ACC}$	650	—	310	—	170	—	nS
Read Data Hold Time	$t_{DHR}$	10	—	10	—	10	—	nS
Read Data Setup Time	$t_{DSU}$	100	—	60	—	60	—	nS
Write Data Delay Time	$t_{MDS}$	—	30	—	30	—	30	nS
Write Data Hold Time	$t_{DHW}$	20	—	20	—	15	—	nS
SO Setup Time	$t_{SO}$	100	—	100	—	100	—	nS
Processor Control Setup Time**	$t_{PCS}$	200	—	150	—	150	—	nS
SYNC Setup Time	$t_{SYNC}$	—	225	—	140	—	100	nS
ML Setup Time	$t_{ML}$	—	225	—	140	—	100	nS
Input Clock Rise/Fall Time	$t_{FQ}, t_{RQ}$	—	25	—	25	—	25	nS

\*NCR65C02 can be held static with  $\theta_2$  high.

\*\*This parameter must only be met to guarantee that the signal will be recognized at the current clock cycle.

### ■ MICROPROCESSOR OPERATIONAL ENHANCEMENTS

Function	NMOS 6502 Microprocessor	NCR65C02 Microprocessor
Indexed addressing across page boundary.	Extra read of invalid address.	Extra read of last instruction byte.
Execution of invalid op codes.	Some terminate only by reset. Results are undefined.	All are NOPs (reserved for future use). Op Code      Bytes      Cycles X2            2            2 X3, X7, XB, XF 1            1 44            2            3 54, D4, F4    2            4 5C            3            8 DC, FC        3            4
Jump indirect, operand = XXFF.	Page address does not increment.	Page address increments and adds one additional cycle.
Read/modify/write instructions at effective address.	One read and two write cycles.	Two read and one write cycle.
Decimal flag.	Indeterminate after reset.	Initialized to binary mode (D=0) after reset and interrupts.
Flags after decimal operation.	Invalid N, V and Z flags.	Valid flag adds one additional cycle.
Interrupt after fetch of BRK instruction.	Interrupt vector is loaded, BRK vector is ignored.	BRK is executed, then interrupt is executed.

### ■ MICROPROCESSOR HARDWARE ENHANCEMENTS

Function	NMOS 6502	NCR65C02
Assertion of Ready RDY during write operations.	Ignored.	Stops processor during $\theta_2$ .
Unused input-only pins (IRQ, NMI, RDY, RES, SO).	Must be connected to low impedance signal to avoid noise problems.	Connected internally by a high-resistance to $V_{DD}$ (approximately 250 K ohm.)

## NCR65C02

### ■ ADDRESSING MODES

Fifteen addressing modes are available to the user of the NCR65C02 microprocessor. The addressing modes are described in the following paragraphs:

#### ■ Implied Addressing [Implied]

In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

#### ■ Accumulator Addressing [Accum]

This form of addressing is represented with a one byte instruction and implies an operation on the accumulator.

#### ■ Immediate Addressing [Immediate]

With immediate addressing, the operand is contained in the second byte of the instruction; no further memory addressing is required.

#### ■ Absolute Addressing [Absolute]

For absolute addressing, the second byte of the instruction specifies the eight low-order bits of the effective address, while the third byte specifies the eight high-order bits. Therefore, this addressing mode allows access to the total 64K bytes of addressable memory.

#### ■ Zero Page Addressing [Zero Page]

Zero page addressing allows shorter code and execution times by only fetching the second byte of the instruction and assuming a zero high address byte. The careful use of zero page addressing can result in significant increase in code efficiency.

#### ■ Absolute Indexed Addressing [ABS, X or ABS, Y]

Absolute indexed addressing is used in conjunction with X or Y index register and is referred to as "Absolute, X," and "Absolute, Y." The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes of the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields, resulting in reduced coding and execution time.

#### ■ Zero Page Indexed Addressing [ZPG, X or ZPG, Y]

Zero page absolute addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y." The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally, due to the "Zero Page" addressing nature of this mode, no carry is added to the high-order eight bits of memory, and crossing of page boundaries does not occur.

#### ■ Relative Addressing [Relative]

Relative addressing is used only with branch instructions;

it establishes a destination for the conditional branch. The second byte of the instruction becomes the operand which is an "Offset" added to the contents of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

#### ■ Zero Page Indexed Indirect Addressing [(IND, X)]

With zero page indexed indirect addressing (usually referred to as indirect X) the second byte of the instruction is added to the contents of the X index register; the carry is discarded. The result of this addition points to a memory location on page zero whose contents is the low-order eight bits of the effective address. The next memory location in page zero contains the high-order eight bits of the effective address. Both memory locations specifying the high- and low-order bytes of the effective address must be in page zero.

#### ■ \*Absolute Indexed Indirect Addressing [ABS(IND, X)] (Jump Instruction Only)

With absolute indexed indirect addressing the contents of the second and third instruction bytes are added to the X register. The result of this addition, points to a memory location containing the lower-order eight bits of the effective address. The next memory location contains the higher-order eight bits of the effective address.

#### ■ Indirect Indexed Addressing [(IND, Y)]

This form of addressing is usually referred to as Indirect, Y. The second byte of the instruction points to a memory location in page zero. The contents of this memory location are added to the contents of the Y index register, the result being the low-order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high-order eight bits of the effective address.

#### ■ \*Zero Page Indirect Addressing [(ZPG)]

In the zero page indirect addressing mode, the second byte of the instruction points to a memory location on page zero containing the low-order byte of the effective address. The next location on page zero contains the high-order byte of the effective address.

#### ■ Absolute Indirect Addressing [(ABS)] (Jump Instruction Only)

The second byte of the instruction contains the low-order eight bits of a memory location. The high-order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low-order byte of the effective address. The next memory location contains the high-order byte of the effective address which is loaded into the 16 bit program counter.

NOTE: \* = New Address Modes

## ■ SIGNAL DESCRIPTION

### Address Bus (A0-A15)

A0-A15 forms a 16-bit address bus for memory and I/O exchanges on the data bus. The output of each address line is TTL compatible, capable of driving one standard TTL load and 130pF.

### Clocks ( $\theta_0$ , $\theta_1$ , and $\theta_2$ )

$\theta_0$  is a TTL level input that is used to generate the internal clocks in the 6502. Two full level output clocks are generated by the 6502. The  $\theta_2$  clock output is in phase with  $\theta_0$ . The  $\theta_1$  output pin is 180° out of phase with  $\theta_0$ . (See timing diagram.)

### Data Bus (D0-D7)

The data lines (D0-D7) constitute an 8-bit bidirectional data bus used for data exchanges to and from the device and peripherals. The outputs are three-state buffers capable of driving one TTL load and 130 pF.

### Interrupt Request (IRQ)

This TTL compatible input requests that an interrupt sequence begin within the microprocessor. The IRQ is sampled during  $\theta_2$  operation; if the interrupt flag in the processor status register is zero, the current instruction is completed and the interrupt sequence begins during  $\theta_1$ . The program counter and processor status register are stored in the stack. The microprocessor will then set the interrupt mask flag high so that no further IRQs may occur. At the end of this cycle, the program counter low will be loaded from address FFFE, and program counter high from location FFFF, transferring program control to the memory vector located at these addresses. The RDY signal must be in the high state for any interrupt to be recognized. A 3K ohm external resistor should be used for proper wire OR operation.

### Memory Lock (ML)

In a multiprocessor system, the ML output indicates the need to defer the rearbitration of the next bus cycle to ensure the integrity of read-modify-write instructions. ML goes low during ASL, DEC, INC, LSR, ROL, ROR, TRB, TSB memory referencing instructions. This signal is low for the modify and write cycles.

### Non-Maskable Interrupt (NMI)

A negative-going edge on this input requests that a non-maskable interrupt sequence be generated within the microprocessor. The NMI is sampled during  $\theta_2$ ; the current instruction is completed and the interrupt sequence begins during  $\theta_1$ . The program counter is loaded with the interrupt vector from locations FFFA (low byte) and FFFB (high byte), thereby transferring program control to the non-maskable interrupt routine.

Note: Since this interrupt is non-maskable, another NMI can occur before the first is finished. Care should be taken when using NMI to avoid this.

### Ready (RDY)

This input allows the user to single-cycle the microprocessor on all cycles including write cycles. A negative transition to the low state, during or coincident with phase one ( $\theta_1$ ), will halt the microprocessor with the output address lines reflecting the current address being fetched. This condition will remain through a subsequent phase two ( $\theta_2$ ) in which the ready signal is low. This feature allows microprocessor interfacing with low-speed memory as well as direct memory access (DMA).

### Reset (RES)

This input is used to reset the microprocessor. Reset must be held low for at least two clock cycles after VDD reaches operating voltage from a power down. A positive transition on this pin will then cause an initialization sequence to begin. Likewise, after the system has been operating, a low on this line of at least two cycles will cease microprocessing activity, followed by initialization after the positive edge on RES.

When a positive edge is detected, there is an initialization sequence lasting six clock cycles. Then the interrupt mask flag is set, the decimal mode is cleared, and the program counter is loaded with the restart vector from locations FFFC (low byte) and FFFD (high byte). This is the start location for program control. This input should be high in normal operation.

### Read/Write (R/W)

This signal is normally in the high state indicating that the microprocessor is reading data from memory or I/O bus. In the low state the data bus has valid data from the microprocessor to be stored at the addressed memory location.

### Set Overflow (SO)

A negative transition on this line sets the overflow bit in the status code register. The signal is sampled on the trailing edge of  $\theta_1$ .

### Synchronize (SYNC)

This output line is provided to identify those cycles during which the microprocessor is doing an OP CODE fetch. The SYNC line goes high during  $\theta_1$  of an OP CODE fetch and stays high for the remainder of that cycle. If the RDY line is pulled low during the  $\theta_1$  clock pulse in which SYNC went high, the processor will stop in its current state and will remain in the state until the RDY line goes high. In this manner, the SYNC signal can be used to control RDY to cause single instruction execution.

## NCR65C02

### ■ INSTRUCTION SET — ALPHABETICAL SEQUENCE

ADC	Add Memory to Accumulator with Carry	LDX	Load Index X with Memory
AND	"AND" Memory with Accumulator	LDY	Load Index Y with Memory
ASL	Shift One Bit Left	LSR	Shift One Bit Right
BCC	Branch on Carry Clear	NOP	No Operation
BCS	Branch on Carry Set	ORA	"OR" Memory with Accumulator
BEQ	Branch on Result Zero	PHA	Push Accumulator on Stack
BIT	Test Memory Bits with Accumulator	PHP	Push Processor Status on Stack
BMI	Branch on Result Minus	*PHX	Push Index X on Stack
BNE	Branch on Result not Zero	*PHY	Push Index Y on Stack
BPL	Branch on Result Plus	PLA	Pull Accumulator from Stack
*BRA	Branch Always	PLP	Pull Processor Status from Stack
BRK	Force Break	*PLX	Pull Index X from Stack
BVC	Branch on Overflow Clear	*PLY	Pull Index Y from Stack
BVS	Branch on Overflow Set	ROL	Rotate One Bit Left
CLC	Clear Carry Flag	ROR	Rotate One Bit Right
CLD	Clear Decimal Mode	RTI	Return from Interrupt
CLI	Clear Interrupt/Disable Bit	RTS	Return from Subroutine
CLV	Clear Overflow Flag	SBC	Subtract Memory from Accumulator with Borrow
CMP	Compare Memory and Accumulator	SEC	Set Carry Flag
CPX	Compare Memory and Index X	SED	Set Decimal Mode
CPY	Compare Memory and Index Y	SEI	Set Interrupt Disable Bit
*DEA	Decrement Accumulator	STA	Store Accumulator in Memory
DEC	Decrement by One	STX	Store Index X in Memory
DEX	Decrement Index X by One	STY	Store Index Y in Memory
DEY	Decrement Index Y by One	*STZ	Store Zero in Memory
EOR	"Exclusive-or" Memory with Accumulator	TAX	Transfer Accumulator to Index X
*INA	Increment Accumulator	TAY	Transfer Accumulator to Index Y
INC	Increment by One	*TRB	Test and Reset Memory Bits with Accumulator
INX	Increment Index X by One	*TSB	Test and Set Memory Bits with Accumulator
INY	Increment Index Y by One	TSX	Transfer Stack Pointer to Index X
JMP	Jump to New Location	TXA	Transfer Index X to Accumulator
JSR	Jump to New Location Saving Return Address	TXS	Transfer Index X to Stack Pointer
LDA	Load Accumulator with Memory	TYA	Transfer Index Y to Accumulator

Note: \* = New Instruction

### ■ MICROPROCESSOR OP CODE TABLE

S	D	O	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	BRK	ORA ind, X			TSB* zpg	ORA zpg	ASL zpg		PHP	ORA imm	ASL A		TSB* abs	ORA abs	ASL abs		0	
1	BPL rel	ORA ind, Y	ORA*† (zpg)		TRB* zpg	ORA zpg, X	ASL zpg, X		CLC	ORA abs, Y	INA* A		TRB* abs	ORA abs, X	ASL abs, X		1	
2	JSR abs	AND ind, X			BIT zpg	AND zpg	ROL zpg		PLP	AND imm	ROL A		BIT abs	AND abs	ROL abs		2	
3	BMI rel	AND ind, Y	AND*† (zpg)		BIT* zpg, X	AND zpg, X	ROL zpg, X		SEC	AND abs, Y	DEA* A		BIT*† abs, X	AND abs, X	ROL abs, X		3	
4	RTI	EOR ind, X				EOR zpg	LSR zpg		PHA	EOR imm	LSR A		JMP abs	EOR abs	LSR abs		4	
5	BVC rel	EOR ind, Y	EOR*† (zpg)			EOR zpg, X	LSR zpg, X		CLI	EOR abs, Y	PHY*			EOR abs, X	LSR abs, X		5	
6	RTS	ADC ind, X			STZ* zpg	ADC zpg	ROR zpg		PLA	ADC imm	ROR A		JMP (abs)	ADC abs	ROR abs		6	
7	BVS rel	ADC ind, Y	ADC*† (zpg)		STZ* zpg, X	ADC zpg, X	ROR zpg, X		SEI	ADC abs, Y	PLY*		JMP*† abs (ind, X)	ADC abs, X	ROR abs, X		7	
8	BRA* rel	STA ind, X			STY zpg	STA zpg	STX zpg		DEY	BIT* imm	TXA		STY abs	STA abs	STX abs		8	
9	BCC rel	STA ind, Y	STA*† (zpg)		STY zpg, X	STA zpg, X	STX zpg, Y		TYA	STA abs, Y	TXS		STZ* abs	STA abs, X	STZ* abs, X		9	
A	LDY imm	LDA ind, X	LDX imm		LDY zpg	LDA zpg, X	LDX zpg, Y		TAY	LDA imm	TAX		LDY abs	LDA abs, X	LDX abs, Y		A	
B	BCS rel	LDA ind, Y	LDA*† (zpg)		LDY zpg, X	LDA zpg, X	LDX zpg, Y		CLV	LDA abs, Y	TSX		LDY abs, X	LDA abs, X	LDX abs, Y		B	
C	CPY imm	CMP ind, X			CPY zpg	CMP zpg	DEC zpg		INY	CMP imm	DEX		CPY abs	CMP abs	DEC abs		C	
D	BNE rel	CMP ind, Y	CMP*† (zpg)			CMP zpg, X	DEC zpg, X		CLD	CMP abs, Y	PHX*			CMP abs, X	DEC abs, X		D	
E	CPX imm	SBC ind, X			CPX zpg	SBC zpg	INC zpg		INX	SBC imm	NOP		CPX abs	SBC abs	INC abs		E	
F	BEQ rel	SBC ind, Y	SBC*† (zpg)			SBC zpg, X	INC zpg, X		SED	SBC abs, Y	PLX*			SBC abs, X	INC abs, X		F	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

Note: \* = New OP Codes

Note: † = New Address Modes

## ■ OPERATIONAL CODES, EXECUTION TIME, AND MEMORY REQUIREMENTS

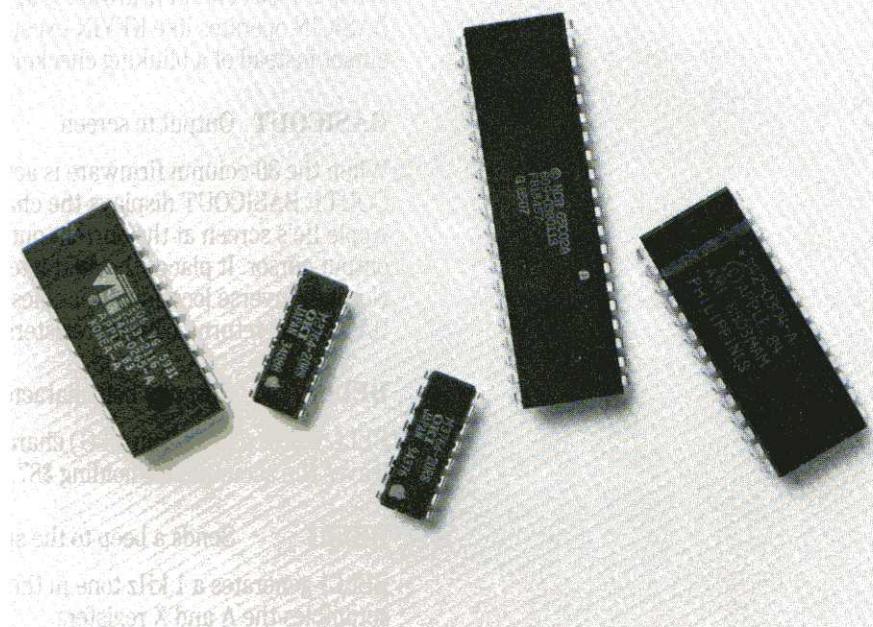
MNE	OPERATION	IMMEDIATE, ABSOLUTE, ZERO PAGE, ACCUMULATED, INDEXED, ZPG, X, ZPG, Y, ABS, X, ABS, Y, RELATIVE, (ABS), ABS (IND, X), ZPG)																PROCESSOR STATUS CODES					
		OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	OP n	MNE	
ADC A + M + C * A	(1,3)	69 2 2 6D 4 3 65 3 2							61 6 2 71 5 2 75 4 2			70 4 3 79 4 3						72 5 2	N V	Z C	ADC		
AND A $\wedge$ M $\wedge$ A	(1)	29 2 2 2D 4 3 25 3 2						21 6 2 31 5 2 35 4 2			30 4 3 39 4 3						32 5 2	N	Z C	AND			
ASL (C $\leftarrow$ I) B $\leftarrow$ 0	(1)	OE 6 3 06 5 2	OA 2 1					16 6 2			1E 6 3											ASL	
BCC Branch if C=0	(2)																					BCC	
BCS Branch if C=1	(2)																					BCS	
BEQ Branch if Z=1	(2)																					BEQ	
BIT A $\wedge$ M	(4,5)	89 2 2 2C 4 3 24 3 2										34 4 2		3C 4 3		F0 2 2						BIT	
BMI Branch if N=1	(2)																					BMI	
BNE Branch if Z=0	(2)																					BNE	
BPL Branch if N=0	(2)																					BPL	
BRA Branch Always	(2)								00 7 1												1 1	BRA	
BRK Break																						BRK	
BVC Branch if V=0	(2)																					BVC	
BVS Branch if V=1	(2)																					BVS	
CLC 0 $\leftarrow$ C								18 2 1														CLC	
CLD 0 $\leftarrow$ D									DB 2 1													CLD	
CLI 0 $\leftarrow$ I									58 2 1													CLI	
CLV 0 $\leftarrow$ V									BB 2 1													CLV	
CMP A - M	(1)	C9 2 2 CD 4 3 C5 3 2							C1 6 2	D1 5 2	D5 4 2						D2 5 2	N	Z C	CMP			
CPX X - M		EO 2 2 EC 4 3 E4 3 2																				CPX	
CPY Y $\leftarrow$ M		C0 2 2 CC 4 3 C4 3 2																				CPY	
DEA A $\leftarrow$ A								3A 2 1														DEA	
DEC M - 1 $\leftarrow$ M	(1)	CE 6 3 C6 5 2										D6 6 2		DE 6 3									DEC
DEX X $\leftarrow$ X																						DEX	
DEY Y $\leftarrow$ Y									CA 2 1													DEY	
EOR A $\vee$ M $\wedge$ A								41 6 2	51 5 2	55 4 2						52 5 2	N	Z C	EOR				
INA A $\leftarrow$ A								1A 2 1														INA	
INC M + 1 $\leftarrow$ M	(1)	EE 6 3 EE 5 2										F6 6 2		FE 6 3								INC	
INX X + 1 $\leftarrow$ X									EB 2 1													INX	
INY Y + 1 $\leftarrow$ Y									CB 2 1													INY	
JMP Jump to new loc								4C 3 3														JMP	
JSR Jump Subroutine								20 6 3														JSR	
LDA M $\leftarrow$ A	(1)	A9 2 2 AD 4 3 A5 3 2							A1 6 2	B1 5 2	B5 4 2						6C 6 3	7C 6 3				LDA	
LDX M $\leftarrow$ X	(1)	A2 2 2 AE 4 3 A6 3 2										B6 4 2		BD 4 3	B9 4 3		B2 5 2	N	Z C	LDX			
LDY M $\leftarrow$ Y	(1)	A0 2 2 AC 4 3 A4 3 2										BC 4 3		BE 4 3								LDY	
LSR 0 $\leftarrow$ I B $\leftarrow$ C	(1)	4E 6 3 46 5 2 4A 2 1										56 6 2		5E 6 3								Z C	
NOP PC $\leftarrow$ PC									EA 2 1													NOP	
ORA A $\vee$ M $\leftarrow$ M	(1)	09 2 2 DD 4 3 05 3 2										4B 3 1		1D 4 3	19 4 3		12 5 2	N	Z C	ORA			
PHA A $\leftarrow$ M <sub>1</sub> S $\leftarrow$ S <sub>1</sub>									08 3 1													PHA	
PHP P $\leftarrow$ M <sub>2</sub> S $\leftarrow$ S <sub>2</sub>																					PHP		
PHX X $\leftarrow$ M <sub>1</sub> S $\leftarrow$ S <sub>1</sub>												DA 3 1										PHX	
PHY Y $\leftarrow$ M <sub>2</sub> S $\leftarrow$ S <sub>2</sub>												5A 3 1										PHY	
PLA S $\leftarrow$ I S M <sub>1</sub> $\leftarrow$ A												6B 4 1										PLA	
PLP S $\leftarrow$ I S M <sub>2</sub> $\leftarrow$ P												28 4 1										PLP	
PLX S $\leftarrow$ I S M <sub>1</sub> $\leftarrow$ X												FA 4 1										PLX	
PLY S + 1 $\leftarrow$ S M <sub>1</sub> $\leftarrow$ Y									7A 4 1													PLY	
ROL I $\leftarrow$ I B $\leftarrow$ C	(1)	2E 6 3 26 5 2 2A 2 1										36 6 2		3E 6 3								Z C	
RRR I $\leftarrow$ I B $\leftarrow$ C	(1)	6E 6 3 66 5 2 2A 2 1										76 6 2		7E 6 3								Z C	
RTI Return from Inter.												40 6 1										RTI	
RTS Return from Subr.												60 6 1										RTS	
SBC A $\leftarrow$ C $\wedge$ A	(1,3)	E9 2 2 ED 4 3 E5 3 2										E1 6 2	F1 5 2	F5 4 2				F2 5 2	N V	Z C	SBC		
SEC 1 $\leftarrow$ C												3B 2 1										SEC	
SED 1 $\leftarrow$ D												FB 2 1										SED	
SEI 1 $\leftarrow$ I												7B 2 1										SEI	
STA A $\leftarrow$ M																					STA		
STX X $\leftarrow$ M									8E 4 3 86 3 2												STX		
STY Y $\leftarrow$ M									8C 4 3 84 3 2												STY		
STZ OO $\leftarrow$ M									9C 4 3 64 3 2												STZ		
TAX A $\leftarrow$ X												AA 2 1										TAX	
TAY A $\leftarrow$ Y												AB 2 1										TAY	
TRB A $\wedge$ M $\leftarrow$ M	(4)	1C 6 3 14 5 2											94 4 2		9E 5 3							Z TRB	
TSB A $\vee$ M $\leftarrow$ M	(4)	OC 6 3 04 5 2											74 4 2									Z TSB	
TSX S $\leftarrow$ X												BA 2 1										Z TSX	
TXA X $\leftarrow$ A												BA 2 1										Z TXA	
TXS X $\leftarrow$ S												BA 2 1										Z TXS	
TYA Y $\leftarrow$ A												98 2 1										Z TYA	

## Notes:

- Add 1 to "n" if page boundary is crossed.
- Add 1 to "n" if branch occurs to same page.
- Add 2 to "n" if branch occurs to different page.
- Add 1 to "n" if decimal mode.
- V bit equals memory bit 6 prior to execution.  
N bit equals memory bit 7 prior to execution.
- The immediate addressing mode of the BIT instruction leaves bits 6 & 7 (V & N) in the Processor Status Code Register unchanged.

## Appendix B

## Directory of Built-in Subroutines



Here is a list of useful subroutines in the Apple IIe's Monitor. To use these subroutines from machine-language programs, store data into the specified memory locations or microprocessor registers as required by the subroutine and execute a JSR to the subroutine's starting address. After the subroutine performs its function, it returns with the 65C02's registers changed as described.

## ▲ Warning

For the sake of compatibility between the Apple II Plus, Apple IIc, and the Apple IIe, do not jump into the middle of Monitor subroutines. The starting addresses are the same for all models of the Apple II, but the actual code is different.

<b>BASICIN</b>	Read the keyboard	\$C305
When the 80-column firmware is active, BASICIN is used instead of KEYIN. BASICIN operates like KEYIN except that it displays a solid, non-blinking cursor instead of a blinking checkerboard cursor.		
<b>BASICOUT</b>	Output to screen	\$C307
When the 80-column firmware is active, BASICOUT is used instead of COUT1. BASICOUT displays the character in the accumulator on the Apple IIe's screen at the current output cursor position and advances the output cursor. It places the character using the setting of the Normal/Inverse location. It handles control codes; see Table 3-3b. BASICOUT returns with all registers intact.		
<b>BELL</b>	Output a bell character	\$FF3A
BELL writes a bell (Control-G) character to the current output device. It leaves the accumulator holding \$87.		
<b>BELL1</b>	Sends a beep to the speaker	\$FBDD
BELL1 generates a 1 kHz tone in the Apple IIe's speaker for 0.1 second. It scrambles the A and X registers.		
<b>CLREOL</b>	Clear to end of line	\$FC9C
CLREOL clears a text line from the cursor position to the right edge of the window. CLREOL destroys the contents of A and Y.		

**CLEOLZ** Clear to end of line \$FC9E

CLEOLZ clears a text line to the right edge of the window, starting at the location given by base address BASL, which is indexed by the contents of the Y register. CLEOLZ destroys the contents of A and Y.

**CLREOP** Clear to end of window \$FC42

CLREOP clears the text window from the cursor position to the bottom of the window. CLREOP destroys the contents of A and Y.

**CLRSCR** Clear the low-resolution screen \$F832

CLRSCR clears the low-resolution graphics display to black. If you call CLRSCR while the video display is in text mode, it fills the screen with inverse-mode at-sign (@) characters. CLRSCR destroys the contents of A and Y.

**CLRTOP** Clear the low-resolution screen \$F836

CLRTOP is the same as CLRSCR (above), except that it clears only the top 40 rows of the low-resolution display.

**COUT** Output a character \$FDED

COUT calls the current character output subroutine. The character to be output should be in the accumulator. COUT calls the subroutine whose address is stored in CSW (locations \$36 and \$37), which is usually one of the standard character output subroutines, COUT1 or BASICOUT.

**COUT1** Output to screen \$FDF0

COUT1 displays the character in the accumulator on the Apple IIe's screen at the current output cursor position and advances the output cursor. It places the character using the setting of the Normal/Inverse location. It handles the codes for carriage return, linefeed, backspace, and bell. It returns with all registers intact.

**CROUT** Generate a carriage return character \$FD8E

CROUT sends a carriage return character to the current output device.

**CROUT1** Generate carriage return, clear rest of line \$FD8B

CROUT1 clears the screen from the current cursor position to the edge of the text window, then calls CROUT.

**GETLN** Get an input line with prompt \$FD6A

GETLN is the standard input subroutine for entire lines of characters, as described in Chapter 3. Your program calls GETLN with the prompt character in location \$33; GETLN returns with the input line in the input buffer (beginning at location \$0200) and the X register holding the length of the input line.

**GETLNZ** Get an input line \$FD67

GETLNZ is an alternate entry point for GETLN that sends a carriage return to the standard output, then continues into GETLN.

**GETLN1** Get an input line, no prompt \$FD6F

GETLN1 is an alternate entry point for GETLN that does not issue a prompt before it accepts the input line. If, however, the user cancels the input line, either with too many backspaces or with a **CONTROL-X**, then GETLN1 will issue the contents of location \$33 as a prompt when it gets another line.

**HLINE** Draw a horizontal line of blocks \$F819

HLINE draws a horizontal line of blocks of the color set by SETCOL on the low-resolution graphics display. Call HLINE with the vertical coordinate of the line in the accumulator, the leftmost horizontal coordinate in the Y register, and the rightmost horizontal coordinate in location \$2C. HLINE returns with A and Y scrambled, X intact.

**HOME** Home cursor and clear \$FC58

HOME clears the display and puts the cursor in the home position: the upper-left corner of the screen.

**IOREST** Restore all registers \$FF3F

IOREST loads the 65C02's internal registers with the contents of memory locations \$45 through \$49.

**IOSAVE** Save all registers \$FF4A

IOSAVE stores the contents of the 65C02's internal registers in locations \$45 through \$49 in the order A, X, Y, P, S. The contents of A and X are changed and the decimal mode is cleared.

<b>KEYIN</b>	Read the keyboard	\$FD1B
KEYIN is the keyboard input subroutine. It reads the Apple IIe's keyboard, waits for a keypress, and randomizes the random number seed at \$4E-\$4F. When a key is pressed, KEYIN removes the blinking cursor from the display and returns with the keycode in the accumulator. KEYIN is described in Chapter 3.		
<b>MOVE</b>	Move a block of memory	\$FE2C
MOVE copies the contents of memory from one range of locations to another. This subroutine is the same as the MOVE command in the Monitor, except that it takes its arguments from pairs of locations in memory, low-byte first. The destination address must be in A4 (\$42-\$43), the starting source address in A1 (\$3C-\$3D), and the ending source address in A2 (\$3E-\$3F) when your program calls MOVE. Register Y must contain \$00 when your program calls MOVE.		
<b>NEXTCOL</b>	Increment color by 3	\$F85F
NEXTCOL adds 3 to the current color (set by SETCOL) used for low-resolution graphics.		
<b>PLOT</b>	Plot on the low-resolution screen	\$F800
PLOT puts a single block of the color value set by SETCOL on the low-resolution display screen. The block's vertical position is passed in the accumulator, its horizontal position in the Y register. PLOT returns with the accumulator scrambled, but X and Y intact.		
<b>PRBLNK</b>	Print three spaces	\$F948
PRBLNK outputs three blank spaces to the standard output device. On return, the accumulator usually contains \$A0, the X register contains 0.		
<b>PRBL2</b>	Print many blank spaces	\$F94A
PRBL2 outputs from 1 to 256 blanks to the standard output device. Upon entry, the X register should contain the number of blanks to be output. If X=\$00, then PRBL2 will output 256 blanks.		
<b>PRBYTE</b>	Print a hexadecimal byte	\$FDDA
PRBYTE outputs the contents of the accumulator in hexadecimal on the current output device. The contents of the accumulator are scrambled.		

<b>PREAD</b>	Read a hand control	\$FB1E
PREAD returns a number that represents the position of a hand control. You pass the number of the hand control in the X register. If this number is not valid (not equal to 0, 1, 2, or 3), strange things may happen. PREAD returns with a number from \$00 to \$FF in the Y register. The accumulator is scrambled.		
<b>PRERR</b>	Print ERR	\$FF2D
PRERR sends the word <i>ERR</i> , followed by a bell character, to the standard output device. On return, the accumulator is scrambled.		
<b>PRHEX</b>	Print a hexadecimal digit	\$FDE3
PRHEX prints the lower nibble of the accumulator as a single hexadecimal digit. On return, the contents of the accumulator are scrambled.		
<b>PRNTAX</b>	Print A and X in hexadecimal	\$F941
PRNTAX prints the contents of the A and X registers as a four-digit hexadecimal value. The accumulator contains the first byte output, the X register contains the second. On return, the contents of the accumulator are scrambled.		
<b>RDCHAR</b>	Get an input character or escape code	\$FD35
RDCHAR is an alternate input subroutine that gets characters from the standard input subroutine, and also interprets the escape codes listed in Chapter 3.		
<b>RDKEY</b>	Get an input character	\$FD0C
RDKEY is the character input subroutine. It places a blinking cursor on the display at the cursor position and jumps to the subroutine whose address is stored in KSW (locations \$38 and \$39), usually the standard input subroutine KEYIN, which returns with a character in the accumulator.		
<b>READ</b>	Read a record from a cassette	\$FEFD
READ reads a series of tones at the cassette input port, converts them to data bytes, and stores the data in a specified range of memory locations. Before calling READ, the address of the first byte must be in A1 (\$3C-\$3D) and the address of the last byte must be in A2 (\$3E-\$3F).		

READ keeps a running exclusive-OR of the data bytes in CHKSUM (\$2E). When the last memory location has been filled, READ reads one more byte and compares it with CHKSUM. If they are equal, READ sends out a beep and returns; if not, it sends the string *ERR* through COUT, sends the beep, and returns.

**SCRN**      Read the low-resolution graphics screen      \$F871

SCRN returns the color value of a single block on the low-resolution graphics display. Call it with the vertical position of the block in the accumulator and the horizontal position in the Y register. Call it as you would call PLOT (above). The color of the block will be returned in the accumulator. No other registers are changed.

**SETCOL**      Set low-resolution graphics color      \$F864

SETCOL sets the color used for plotting in low-resolution graphics to the value passed in the accumulator. The colors and their values are listed in Table 2-6.

**SETINV**      Set inverse mode      \$FE80

SETINV sets the display format to inverse. COUT1 will then display all output characters as black dots on a white background. The Y register is set to \$3F, all others are unchanged.

**SETNORM**      Set normal mode      \$FE84

SETNORM sets the display format to normal. COUT1 will then display all output characters as white dots on a black background. On return, the Y register is set to \$FF, all others are unchanged.

**VERIFY**      Compare two blocks of memory      \$FE36

VERIFY compares the contents of one range of memory to another. This subroutine is the same as the VERIFY command in the Monitor, except it takes its arguments from pairs of locations in memory, low-byte first. The destination address must be in A4 (\$42-\$43), the starting source address in A1 (\$3C-\$3D), and the ending source address in A2 (\$3E-\$3F) when your program calls VERIFY.

**VLINE** Draw a vertical line of blocks \$F828

VLINE draws a vertical line of blocks of the color set by SETCOL on the low-resolution display. You should call VLINE with the horizontal coordinate of the line in the Y register, the top vertical coordinate in the accumulator, and the bottom vertical coordinate in location \$2D. VLINE will return with the accumulator scrambled.

**WAIT** Delay \$FCA8

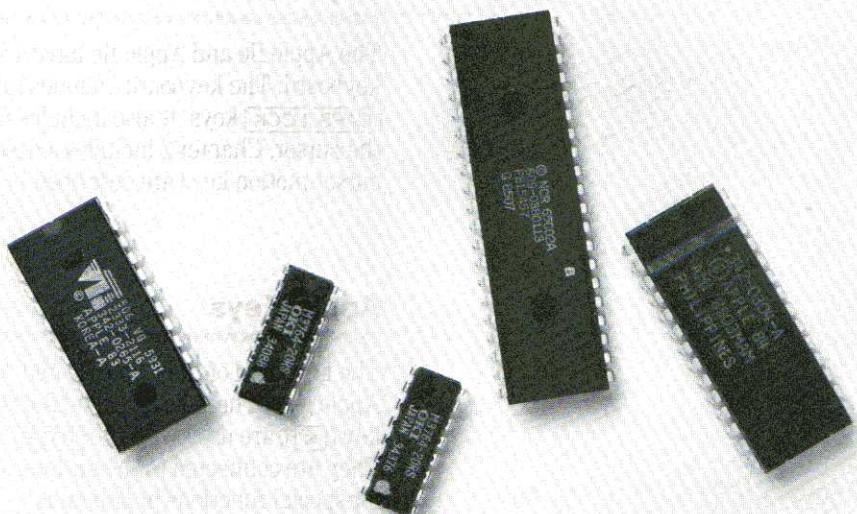
WAIT delays for a specific amount of time, then returns to the program that called it. The amount of delay is specified by the contents of the accumulator. The delay is  $1/2(26+27A+5A^2)$  microseconds, where A is the contents of the accumulator. WAIT returns with the accumulator zeroed and the X and Y registers undisturbed.

**WRITE** Write a record on a cassette \$FECD

WRITE converts the data in a range of memory to a series of tones at the cassette output port. Before calling WRITE, the address of the first data byte must be in A1 (\$3C-\$3D) and the address of the last byte must be in A2 (\$3E-\$3F). The subroutine writes a ten-second continuous tone as a header, then writes the data followed by a one-byte checksum.

## Appendix C

## Apple II Family Differences



This appendix lists the differences among the Apple II Plus, the original and the enhanced Apple IIe, and the Apple IIc.

If you're trying to write software to run on more than one version of the Apple II, this appendix will help you avoid unexpected problems of incompatibility.

The differences are listed here in approximately the order you are likely to encounter them: obvious differences first, technical details later. Each entry in the list includes references to the chapters in this manual where the item is described.

## Keyboard

The Apple IIe and Apple IIc have a full 62-key uppercase and lowercase keyboard. The keyboard includes fully-operational **SHIFT** and **CAPS LOCK** keys. It also includes four directional arrow keys for moving the cursor. Chapter 2 includes a description of the keyboard. The cursor-motion keys are described in Chapter 3.

## Apple Keys

The keyboard of the Apple IIe and Apple IIc have two keys marked with the Apple logo. These keys, called the Open-Apple key (□) and Solid-Apple key (■), are used with the **RESET** key to select special reset functions. They are connected to the buttons on the hand controls, so they can be used for special functions in programs.

The Apple II and the Apple II Plus do not have Apple keys.

## Character Sets

The Apple IIe and Apple IIc can display the full ASCII character set, uppercase and lowercase. For compatibility with older Apple II's, the standard display character set includes flashing uppercase instead of inverse-format lowercase; you can also switch to an alternate character set with inverse lowercase and uppercase, but no flashing. Chapter 2 includes a description of the display character sets. Chapter 3 tells you how to switch display formats.

The Apple IIc and the enhanced Apple IIe include a set of “graphic” text characters, called MouseText characters, that replace some of the inverse uppercase characters in the alternate character set of the original Apple IIe. MouseText characters are described in Chapter 2.

### 80-Column Display

---

With the addition of an 80-column text card, the Apple IIe can display 80 columns of text. The 80-column display is completely compatible with both graphics modes—you can even use it in mixed mode. (If you prefer, you can use an old-style 80-column card in an expansion slot instead.) Chapter 2 includes a description of the 80-column display.

The Apple IIc has a built-in extended 80-column card.

### Escape Codes and Control Characters

---

On the Apple IIe and Apple IIc, the display features mentioned above (and many others not mentioned) can be controlled from the keyboard by escape sequences and from programs by control characters. Chapter 3 includes descriptions of those escape codes and control characters.

### Built-in Language Card

---

The 16K bytes of RAM you add to the Apple II Plus by installing the Language Card is built into the Apple IIe and Apple IIc, giving the Apple IIe a standard memory size of 64K bytes. (The Apple IIc has a built-in extended 80-column text card as well, giving it a standard memory size of 128K bytes.) In the Apple IIe, this 16K-byte block of memory is called the bank-switched memory. It is described in Chapter 4.



## Auxiliary Memory

By installing the Apple IIe Extended 80-Column Text Card, you can add an alternate 64K bytes of RAM to the Apple IIe. Chapter 4 tells you how to use the additional memory. (The Extended 80-Column Text Card also provides the 80-column display option.)

The Apple IIc has a built-in extended 80-column text card.

## Auxiliary Slot

In addition to the expansion slots on the Apple II Plus, the Apple IIe has a special slot that is used either for the 80-Column Text Card or for the Extended 80-Column Text Card. This slot is identified in Chapter 1 and described in Chapter 7.

The Apple IIc has the functions of the auxiliary slot built in.

## Back Panel and Connectors

The Apple IIe has a metal back panel with space for several D-type connectors. Each peripheral card you add comes with a connector that you install in the back panel. Chapter 1 includes a description of the back panel; for details, see the installation instructions supplied with the peripheral cards.

The Apple IIc back panel has seven built-in connectors.

## Soft Switches

The display and memory features of the Apple IIe and the Apple IIc are controlled by soft switches like the ones on the Apple II Plus. On the Apple IIe and the Apple IIc, programs can also read the settings of the soft switches. Chapter 2 describes the soft switches that control the display features, and Chapter 4 describes the soft switches that control the memory features.

## Built-in Self-Test

---

The Apple IIe has built-in firmware that includes a self-test routine. The self-test is intended primarily for testing during manufacturing, but you can run it to be sure the Apple IIe is working correctly. The self-test is described in Chapter 4.

The Apple IIc also has built-in diagnostics.

## Forced Reset

---

Some programs on the Apple II Plus take control of the reset function to keep users from stopping the machine and copying the program. The Apple IIe and Apple IIc have a forced reset that writes over the program in memory. By using the forced reset, you can restart the Apple IIe (or Apple IIc) without turning power off and on and causing unnecessary stress on the circuits. The forced reset is described in Chapter 4.

## Interrupt Handling

---

Even though most application programs don't use interrupts, the Apple IIe (and Apple IIc) provide for interrupt-driven programs. For example, the 80-column firmware periodically enables interrupts while it is clearing the display (normally a long time to have interrupts locked out). Interrupts are discussed in Chapter 6.

## Vertical Sync for Animators

---

Programs with animation on the Apple IIe and Apple IIc can stay in step with the display and avoid flickering objects in their displays. Chapter 7 includes a description of the video generation and the vertical sync.

## Signature Byte

A program can find out whether it's running on an Apple IIe, Apple IIc, Apple III (in emulation mode), or on an older model Apple II by reading the byte at location \$FBB3 in the System Monitor. In the Apple IIe Monitor, this byte's value is \$06; in the Autostart Monitor (the standard Monitor on the Apple II Plus), its value is \$EA. (Note: if you start up with DOS and switch to Integer BASIC, the Autostart Monitor is active and the value at location \$FBB3 is \$EA, even on an Apple IIe.) Obviously, there are lots of other locations that have different values in the different versions of the Monitor; location \$FBB3 was chosen because it will have the value \$06 even in future revisions of the Apple IIe Monitor.

## Hardware Implementation

The hardware implementation of the Apple IIe is radically different from the Apple II and Apple II Plus. Three of the more important differences are

- the custom ICs: the IOU and MMU
- the video hardware, which uses ROM to generate both text and graphics
- the peripheral data bus, which is fully buffered.

### The Apple IIc

- shares some of the custom ICs of the Apple IIe
- has some new ones all its own
- lacks the slots of the Apple IIe, replacing some of them with built-in I/O ports.

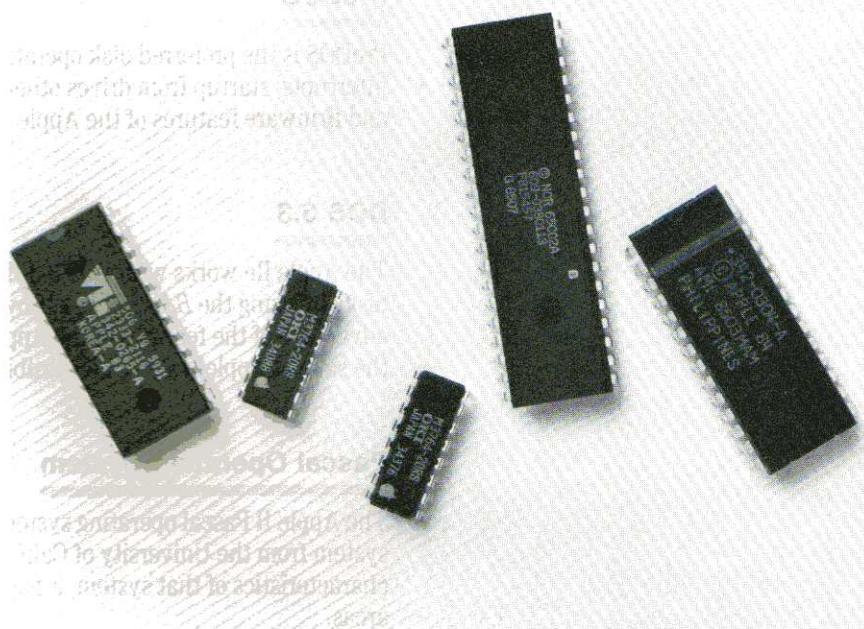
All of these features are described in Chapter 7.

For more information about the Apple IIc, see the *Apple IIc Reference Manual*.

Appendix D

---

## Operating Systems and Languages



This appendix is an overview of the characteristics of operating systems and languages when run on the Apple IIe. It is not intended to be a full account. For more information, refer to the manuals that are provided with each product.

## Operating Systems

This section discusses the operating systems that can be used with the Apple IIe.

### ProDOS

ProDOS is the preferred disk operating system for the Apple IIe. It supports interrupts, startup from drives other than a Disk II, and all other hardware and firmware features of the Apple IIe.

### DOS 3.3

The Apple IIe works with DOS 3.3. The Apple IIe can also access DOS 3.2 disks by using the *BASIC S* disk. However, neither version of DOS takes full advantage of the features of the Apple IIe. DOS support is provided only for the sake of Apple II series compatibility.

### Pascal Operating System

The Apple II Pascal operating system was developed from the UCSD Pascal system from the University of California at San Diego. While it shares many characteristics of that system, it has been extended by Apple in several areas.

Pascal versions 1.2 and later support interrupts and all the hardware and firmware features of the Apple IIe.

The Apple II Pascal system uses a disk format different than either ProDOS or DOS 3.3.

## **CP/M**

CP/M® is an operating system developed by Digital Research that runs on either the Intel 8080 or Zilog Z80® microprocessors. This means that a co-processor peripheral card, available from several manufacturers for the Apple IIe, is required to run CP/M. Several versions of CP/M from 1.4 through 3.0 and later can be run on an Apple IIe with an appropriate co-processor card.

## **Languages**

This section discusses special techniques to use, and characteristics to be aware of, when using Apple programming languages with the Apple IIe.

### **Assembly Language**

Programs written in assembly language have the potential of extracting the most speed and efficiency from your Apple IIe, but they also require the most effort on your part.

### **Applesoft BASIC**

The focus of the chapters in this manual is assembly language, and so most addresses and values are given in hexadecimal notation. Appendix E in this manual includes tables to help you convert from hexadecimal to the decimal notation you will need for BASIC.

In BASIC, use a PEEK to read a location (instead of the LDA used in assembly language), and a POKE (instead of STA) to write to a location. If you read a hardware address from a BASIC program, you get a value between 0 and 255. Bit 7 holds a place value of 128, so if a soft switch is on, its value will be equal to or greater than 128; if the switch is off, the value will be less than 128.

### **Integer BASIC**

Integer BASIC is not included in the Apple IIe firmware. If you want to run it on your Apple IIe, you must use DOS 3.3 to load it in to the system. ProDOS does not support Integer BASIC.

### **Pascal Language**

---

The Pascal language works on the Apple IIe under versions 1.1 and later of the Pascal Operating System. However, for best performance, use Pascal 1.2 or a later version.

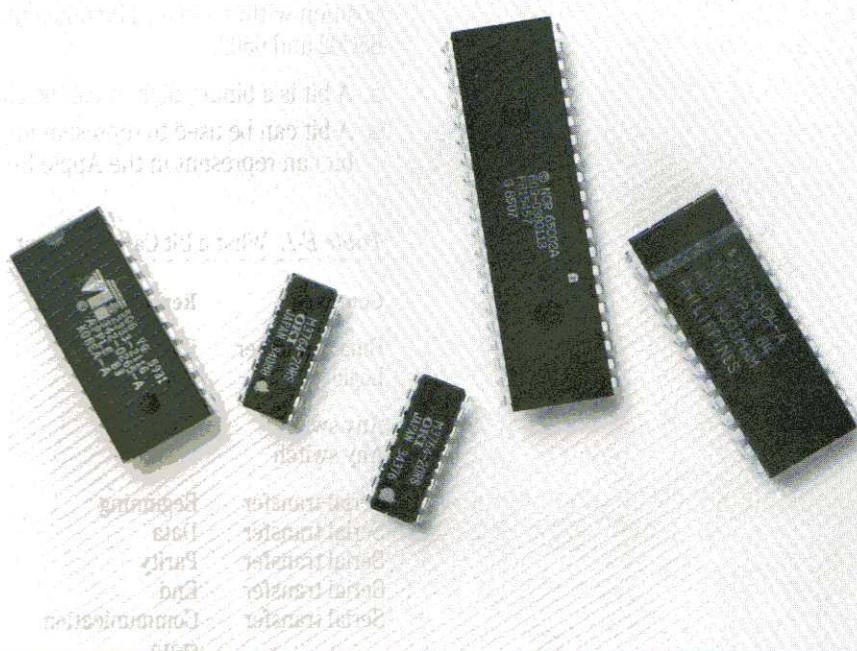
### **FORTRAN**

---

FORTRAN works under version 1.1 of the Pascal Operating System which does not detect or use certain Apple IIe features, such as auxiliary memory. Therefore, FORTRAN does not take advantage of these features.

## Appendix E

## Conversion Tables



Small chip	Medium chip
Large chip	Very large chip
Very large chip	Large chip
Medium chip	Small chip
Large chip	Medium chip

This appendix briefly discusses bits and bytes and what they can represent. It also contains conversion tables for hexadecimal to decimal and negative decimal, for low-resolution display dot patterns, display color values, and a number of 8-bit codes.

These tables are intended for convenient reference. This appendix is not intended as a tutorial for the materials discussed. The brief section introductions are for orientation only.

## Bits and Bytes

This section discusses the relationships between bit values and their position within a byte. The following are some rules of thumb regarding the 65C02 and 6502.

- A bit is a binary digit; it can be either a 0 or a 1.
- A bit can be used to represent any two-way choice. Some choices that a bit can represent in the Apple IIe are listed in Table E-1.

*Table E-1.* What a Bit Can Represent

Context	Representing	0 =	1 =
Binary number	Place value	0	1 x that power of 2
Logic	Condition	False	True
Any switch	Position	Off	On
Any switch	Position	Clear *	Set
Serial transfer	Beginning	Start	Carrier (no information yet)
Serial transfer	Data	0 value	1 value
Serial transfer	Parity	SPACE	MARK
Serial transfer	End		Stop bit(s)
Serial transfer	Communication state	BREAK	Carrier
P reg. bit N	Neg. result?	No	Yes
P reg. bit V	Overflow?	No	Yes
P reg. bit B	BRK command?	No	Yes
P reg. bit D	Decimal mode?	No	Yes
P reg. bit I	IRQ interrupts	Enabled	Disabled (masked out)
P reg. bit Z	Zero result?	No	Yes
P reg. bit C	Carry required?	No	Yes

\* Sometimes ambiguously termed *reset*.

- Bits can also be combined in groups of any size to represent numbers.  
Most of the commonly used sizes are multiples of four bits.
- Four bits comprise a nibble (sometimes spelled *nybble*).
- One nibble can represent any of 16 values. Each of these values is assigned a number from 0 through 9 and (because our decimal system has only ten of the sixteen digits we need) A through F.
- Eight bits (two nibbles) make a byte (Figure E-1).

*Figure E-1.* Bits, Nibbles, and Bytes

High Nibble								Low Nibble	
MSB	7	6	5	4	3	2	1	LSB	0
	\$80	\$40	\$20	\$10	\$08	\$04	\$02	\$01	
	128	64	32	16	8	4	2	1	
									Hexadecimal
									Decimal
Binary	Hex	Dec							
0000	\$00	0							
0001	\$01	1							
0010	\$02	2							
0011	\$03	3							
0100	\$04	4							
0101	\$05	5							
0110	\$06	6							
0111	\$07	7							
1000	\$08	8							
1001	\$09	9							
1010	\$0A	10							
1011	\$0B	11							
1100	\$0C	12							
1101	\$0D	13							
1110	\$0E	14							
1111	\$0F	15							

- One byte can represent any of  $16 \times 16$  or 256 values. The value can be specified by exactly two hexadecimal digits.
- Bits within a byte are numbered from bit 0 on the right to bit 7 on the left.
- The bit number is the same as the power of 2 that it represents, in a manner completely analogous to the digits in a decimal number.

- One memory position in the Apple IIe contains one eight-bit byte of data.
- How byte values are interpreted depends on whether the byte is an instruction in a language, part or all of an address, an ASCII code, or some other form of data.
- Two bytes make a word. The sixteen bits of a word can represent any one of  $256 \times 256$  or 65536 different values.
- The 65C02 uses a 16-bit word to represent memory locations. It can therefore distinguish among 65536 (64K) locations at any given time.
- A memory location is one byte of a 256-byte page. The low-order byte of an address specifies this byte. The high-order byte specifies the memory page the byte is on.

## Hexadecimal and Decimal

Use Table E-2 for conversion of hexadecimal and decimal numbers.

*Table E-2.* Hexadecimal/Decimal Conversion

Digit	\$x000	\$0x00	\$00x0	\$000x
F	61440	3840	240	15
E	57344	3584	224	14
D	53248	3328	208	13
C	49152	3072	192	12
B	45056	2816	176	11
A	40960	2560	160	10
9	36864	2304	144	9
8	32768	2048	128	8
7	28672	1792	112	7
6	24576	1536	96	6
5	20480	1280	80	5
4	16384	1024	64	4
3	12288	768	48	3
2	8192	512	32	2
1	4096	256	16	1

To convert a hexadecimal number to a decimal number, find the decimal numbers corresponding to the positions of each hexadecimal digit. Write them down and add them up.

**Examples:**

$$\begin{array}{ll} \$3C = ? & \$FD47 = ? \\ \$30 = 48 & \$F000 = 61440 \\ \$0C = 12 & \$D00 = 3328 \\ \hline & \$40 = 64 \\ & \$7 = 7 \\ \$3C = 60 & \hline \\ & \$FD47 = 64839 \end{array}$$

To convert a decimal number to hexadecimal, subtract from the decimal number the largest decimal entry in the table that is less than the number. Write down the hexadecimal digit (noting its place value) also. Now subtract the largest decimal number in the table that is less than the decimal remainder, and write down the next hexadecimal digit. Continue until you have zero left. Add up the hexadecimal numbers.

**Example:**

$$\begin{array}{lll} 16215 = \$ ? & & \\ 16215 - 12288 = 3927 & 12288 = \$7000 & \\ 3927 - 3840 = 87 & 3840 = \$F00 & \\ 87 - 80 = 7 & 80 = \$50 & \\ 7 & 7 = \$7 & \\ \hline & & \\ & 16215 = \$7F57 & \end{array}$$

## Hexadecimal and Negative Decimal

If a number is larger than decimal 32767, Applesoft BASIC allows and Integer BASIC requires that you use the negative-decimal equivalent of the number. Table E-3 is set up to make it easy for you to convert a hexadecimal number directly to a negative decimal number.

*Table E-3.* Hexadecimal to Negative Decimal Conversion

Digit	\$x000	\$\$0x00	\$\$00x0	\$\$000x
F	0	0	0	-1
E	-4096	-256	-16	-2
D	-8192	-512	-32	-3
C	-12288	-768	-48	-4
B	-16384	-1024	-64	-5
A	-20480	-1280	-80	-6
9	-24576	-1536	-96	-7
8	-28672	-1792	-112	-8
7		-2048	-128	-9
6		-2304	-144	-10
5		-2560	-160	-11
4		-2816	-176	-12
3		-3072	-192	-13
2		-3328	-208	-14
1		-3584	-224	-15
0		-3840	-240	-16

To perform this conversion, write down the four decimal numbers corresponding to the four hexadecimal digits (zeros included). Then add their values. The resulting number is the desired negative decimal number.

### Example:

```
$C010 = - ?  
$C000: -12288  
$ 000: - 3840  
$ 10: - 224  
$ 0: - 16  
-----  
$C010 - 16368
```

To convert a negative-decimal number to a positive decimal number, add it to 65536. (This addition ends up looking like subtraction.)

**Example:**

$-151 = + ?$

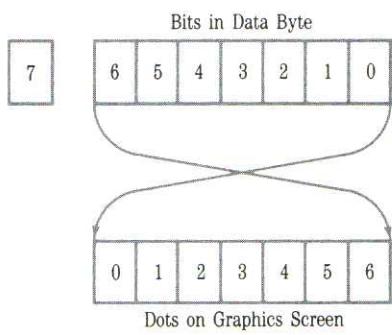
$$65536 + (-151) = 65536 - 151 = 65385$$

To convert a negative-decimal number to a hexadecimal number, first convert it to a positive decimal number, then use Table E-2.

## Graphics Bits and Pieces

Table E-4 is a quick guide to the hexadecimal values corresponding to 7-bit high-resolution patterns on the display screen. Since the bits are displayed in reverse order, it takes some calculation to determine these values. Table E-4 should make it easy.

*Table E-4. Hexadecimal Values for High-Resolution Dot Patterns*



Bit Pattern	x=0	x=1	Bit Pattern	x=0	x=1
x0000000	\$00	\$80	x0100000	\$02	\$82
x0000001	\$40	\$C0	x0100001	\$42	\$C2
x0000010	\$20	\$A0	x0100010	\$22	\$A2
x0000011	\$60	\$E0	x0100011	\$62	\$E2
x0000100	\$10	\$90	x0100100	\$12	\$92
x0000101	\$50	\$D0	x0100101	\$52	\$D2
x0000110	\$30	\$B0	x0100110	\$32	\$B2
x0000111	\$70	\$F0	x0100111	\$72	\$F2
x0001000	\$08	\$88	x0101000	\$0A	\$8A
x0001001	\$48	\$C8	x0101001	\$4A	\$CA
x0001010	\$28	\$A8	x0101010	\$2A	\$AA
x0001011	\$68	\$E8	x0101011	\$6A	\$EA
x0001100	\$18	\$98	x0101100	\$1A	\$9A
x0001101	\$58	\$D8	x0101101	\$5A	\$DA
x0001110	\$38	\$B8	x0101110	\$3A	\$BA
x0001111	\$78	\$F8	x0101111	\$7A	\$FA
x0010000	\$04	\$84	x0110000	\$06	\$86
x0010001	\$44	\$C4	x0110001	\$46	\$C6
x0010010	\$24	\$A4	x0110010	\$26	\$A6
x0010011	\$64	\$E4	x0110011	\$66	\$E6
x0010100	\$14	\$94	x0110100	\$16	\$96
x0010101	\$54	\$D4	x0110101	\$56	\$D6
x0010110	\$34	\$B4	x0110110	\$36	\$B6
x0010111	\$74	\$F4	x0110111	\$76	\$F6
x0011000	\$0C	\$8C	x0111000	\$0E	\$8E
x0011001	\$4C	\$CC	x0111001	\$4E	\$CE
x0011010	\$2C	\$AC	x0111010	\$2E	\$AE
x0011011	\$6C	\$EC	x0111011	\$6E	\$EE
x0011100	\$1C	\$9C	x0111100	\$1E	\$9E
x0011101	\$5C	\$DC	x0111101	\$5E	\$DE
x0011110	\$3C	\$BC	x0111110	\$3E	\$BE
x0011111	\$7C	\$FC	x0111111	\$7E	\$FE

The *x* represents bit 7. Zeros represent bits that are off; ones bits that are on. Use the first hexadecimal value if bit 7 is to be off, and the second if it is to be on.

For example, to get bit pattern 00101110, use \$3A; for 10101110, use \$BA.

*Table E-4—Continued.* Hexadecimal Values for High-Resolution Dot Patterns

Bit Pattern	x=0	x=1	Bit Pattern	x=0	x=1
x1000000	\$01	\$81	x1100000	\$03	\$83
x1000001	\$41	\$C1	x1100001	\$43	\$C3
x1000010	\$21	\$A1	x1100010	\$23	\$A3
x1000011	\$61	\$E1	x1100011	\$63	\$E3
x1000100	\$11	\$91	x1100100	\$13	\$93
x1000101	\$51	\$D1	x1100101	\$53	\$D3
x1000110	\$31	\$B1	x1100110	\$33	\$B3
x1000111	\$71	\$F1	x1100111	\$73	\$F3
x1001000	\$09	\$89	x1101000	\$0B	\$8B
x1001001	\$49	\$C9	x1101001	\$4B	\$CB
x1001010	\$29	\$A9	x1101010	\$2B	\$AB
x1001011	\$69	\$E9	x1101011	\$6B	\$EB
x1001100	\$19	\$99	x1101100	\$1B	\$9B
x1001101	\$59	\$D9	x1101101	\$5B	\$DB
x1001110	\$39	\$B9	x1101110	\$3B	\$BB
x1001111	\$79	\$F9	x1101111	\$7B	\$FB
x1010000	\$05	\$85	x1110000	\$07	\$87
x1010001	\$45	\$C5	x1110001	\$47	\$C7
x1010010	\$25	\$A5	x1110010	\$27	\$A7
x1010011	\$65	\$E5	x1110011	\$67	\$E7
x1010100	\$15	\$95	x1110100	\$17	\$97
x1010101	\$55	\$D5	x1110101	\$57	\$D7
x1010110	\$35	\$B5	x1110110	\$37	\$B7
x1010111	\$75	\$F5	x1110111	\$77	\$F7
x1011000	\$0D	\$8D	x1111000	\$0F	\$8F
x1011001	\$4D	\$CD	x1111001	\$4F	\$CF
x1011010	\$2D	\$AD	x1111010	\$2F	\$AF
x1011011	\$6D	\$ED	x1111011	\$6F	\$EF
x1011100	\$1D	\$9D	x1111100	\$1F	\$9F
x1011101	\$5D	\$DD	x1111101	\$5F	\$DF
x1011110	\$3D	\$BD	x1111110	\$3F	\$BF
x1011111	\$7D	\$FD	x1111111	\$7F	\$FF

## Eight-Bit Code Conversions

Tables E-5 through E-12 show the entire ASCII character set twice: once with the high bit off, and once with it on. Here is how to interpret these tables.

- The *Binary* column has the 8-bit code for each ASCII character.
- The first 128 ASCII entries represent 7-bit ASCII codes plus a high-order bit of 0 (SPACE parity or Pascal)—for example, 010010000 for the letter *H*.
- The last 128 ASCII entries (from 128 through 255) represent 7-bit ASCII codes plus a high-order bit of 1 (MARK parity or BASIC)—for example, 11001000 for the letter *H*.
- A transmitted or received ASCII character will take whichever form is appropriate if odd or even parity is selected—for example, 11001000 for an odd-parity *H*, 01001000 for an even-parity *H*.
- The *ASCII Char* column gives the ASCII character name.
- The *Interpretation* column spells out the meaning of special symbols and abbreviations, where necessary.
- The *What to Type* column indicates what keystrokes generate the ASCII character (where it is not obvious).
- The columns marked *Pri* and *Alt* indicate what displayed character results from each code when using the primary or alternate display character set, respectively. Boldface is used for inverse characters; italic is used for flashing characters.

Note that the values \$40 through \$5F (and \$C0 through \$DF) in the alternate character set are displayed as MouseText characters if MouseText is turned on.

The MouseText characters are shown in Table E-7.

**Note:** The primary and alternate displayed character sets in Tables E-5 through E-12 are the result of firmware mapping. The character generator ROM actually contains only one character set. The firmware mapping procedure is described in the section "Inverse and Flashing Text," in Chapter 3.

Table E-5. Control Characters, High Bit Off

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
0000000	0	\$00	NUL	Blank (null)	CONTROL-@	@	@
0000001	1	\$01	SOH	Start of Header	CONTROL-A	A	A
0000010	2	\$02	STX	Start of Text	CONTROL-B	B	B
0000011	3	\$03	ETX	End of Text	CONTROL-C	C	C
0000100	4	\$04	EOT	End of Transm.	CONTROL-D	D	D
0000101	5	\$05	ENQ	Enquiry	CONTROL-E	E	E
0000110	6	\$06	ACK	Acknowledge	CONTROL-F	F	F
0000111	7	\$07	BEL	Bell	CONTROL-G	G	G
0001000	8	\$08	BS	Backspace	CONTROL-H or →	H	H
0001001	9	\$09	HT	Horizontal Tab	CONTROL-I or TAB	I	I
0001010	10	\$0A	LF	Line Feed	CONTROL-J or ↓	J	J
0001011	11	\$0B	VT	Vertical Tab	CONTROL-K or ↑	K	K
0001100	12	\$0C	FF	Form Feed	CONTROL-L	L	L
0001101	13	\$0D	CR	Carriage Return	CONTROL-M or RETURN	M	M
0001110	14	\$0E	SO	Shift Out	CONTROL-N	N	N
0001111	15	\$0F	SI	Shift In	CONTROL-O	O	O
0010000	16	\$10	DLE	Data Link Escape	CONTROL-P	P	P
0010001	17	\$11	DC1	Device Control 1	CONTROL-Q	Q	Q
0010010	18	\$12	DC2	Device Control 2	CONTROL-R	R	R
0010011	19	\$13	DC3	Device Control 3	CONTROL-S	S	S
0010100	20	\$14	DC4	Device Control 4	CONTROL-T	T	T
0010101	21	\$15	NAK	Neg. Acknowledge	CONTROL-U or →	U	U
0010110	22	\$16	SYN	Synchronization	CONTROL-V	V	V
0010111	23	\$17	ETB	End of Text Blk.	CONTROL-W	W	W
0011000	24	\$18	CAN	Cancel	CONTROL-X	X	X
0011001	25	\$19	EM	End of Medium	CONTROL-Y	Y	Y
0011010	26	\$1A	SUB	Substitute	CONTROL-Z	Z	Z
0011011	27	\$1B	ESC	Escape	CONTROL-[ or ESC	[	[
0011100	28	\$1C	FS	File Separator	CONTROL-\	\	\
0011101	29	\$1D	GS	Group Separator	CONTROL-]	]	]
0011110	30	\$1E	RS	Record Separator	CONTROL-^	^	^
0011111	31	\$1F	US	Unit Separator	CONTROL-_	-	-

*Table E-6.* Special Characters, High Bit Off

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
0100000	32	\$20	SP	Space	[SPACE] bar		
0100001	33	\$21	!			!	!
0100010	34	\$22	"			"	"
0100011	35	\$23	#			#	#
0100100	36	\$24	\$			\$	\$
0100101	37	\$25	%			%	%
0100110	38	\$26	&			&	&
0100111	39	\$27	,	Closing Quote		,	,
0101000	40	\$28	(			(	(
0101001	41	\$29	)			)	)
0101010	42	\$2A	*			*	*
0101011	43	\$2B	+			+	+
0101100	44	\$2C	,	Comma		,	,
0101101	45	\$2D	-	Hypen		-	-
0101110	46	\$2E	.	Period		.	.
0101111	47	\$2F	/			/	/
0110000	48	\$30	0			0	0
0110001	49	\$31	1			1	1
0110010	50	\$32	2			2	2
0110011	51	\$33	3			3	3
0110100	52	\$34	4			4	4
0110101	53	\$35	5			5	5
0110110	54	\$36	6			6	6
0110111	55	\$37	7			7	7
0111000	56	\$38	8			8	8
0111001	57	\$39	9			9	9
0111010	58	\$3A	:			:	:
0111011	59	\$3B	;			;	;
0111100	60	\$3C	<			<	<
0111101	61	\$3D	=			=	=
0111110	62	\$3E	>			>	>
0111111	63	\$3F	?			?	?

**Table E-7.** Uppercase Characters, High Bit Off

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
1000000	64	\$40	@			@	apple
1000001	65	\$41	A			A	Α
1000010	66	\$42	B			B	Β
1000011	67	\$43	C			C	Γ
1000100	68	\$44	D			D	Δ
1000101	69	\$45	E			E	Ε
1000110	70	\$46	F			F	Φ
1000111	71	\$47	G			G	Γ
1001000	72	\$48	H			H	Η
1001001	73	\$49	I			I	Ι
1001010	74	\$4A	J			J	Ι
1001011	75	\$4B	K			K	Κ
1001100	76	\$4C	L			L	Λ
1001101	77	\$4D	M			M	Μ
1001110	78	\$4E	N			N	Ν
1001111	79	\$4F	O			O	Ο
1010000	80	\$50	P			P	Π
1010001	81	\$51	Q			Q	Ϙ
1010010	82	\$52	R			R	Ϙ
1010011	83	\$53	S			S	Ϻ
1010100	84	\$54	T			T	Ϻ
1010101	85	\$55	U			U	ϻ
1010110	86	\$56	V			V	ϻ
1010111	87	\$57	W			W	ϻ
1011000	88	\$58	X			X	ϻ
1011001	89	\$59	Y			Y	ϻ
1011010	90	\$5A	Z			Z	ϻ
1011011	91	\$5B	[	Opening Bracket		[	↶
1011100	92	\$5C	\	Reverse Slant		\	↷
1011101	93	\$5D	]	Closing Bracket		]	↷
1011110	94	\$5E	^	Caret		^	↑
1011111	95	\$5F	_	Underline		_	↔

*Table E-8.* Lowercase Characters, High Bit Off

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
1100000	96	\$60	`	Opening Quote		'	
1100001	97	\$61	a			!	a
1100010	98	\$62	b			"	b
1100011	99	\$63	c			#	c
1100100	100	\$64	d			\$	d
1100101	101	\$65	e			%	e
1100110	102	\$66	f			&	f
1100111	103	\$67	g			,	g
1101000	104	\$68	h			(	h
1101001	105	\$69	i			)	i
1101010	106	\$6A	j			*	j
1101011	107	\$6B	k			+	k
1101100	108	\$6C	l			,	l
1101101	109	\$6D	m			-	m
1101110	110	\$6E	n			.	n
1101111	111	\$6F	o			/	o
1110000	112	\$70	p			0	p
1110001	113	\$71	q			1	q
1110010	114	\$72	r			2	r
1110011	115	\$73	s			3	s
1110100	116	\$74	t			4	t
1110101	117	\$75	u			5	u
1110110	118	\$76	v			6	v
1110111	119	\$77	w			7	w
1111000	120	\$78	x			8	x
1111001	121	\$79	y			9	y
1111010	122	\$7A	z			:	z
1111011	123	\$7B	{	Opening Brace		;	{
1111100	124	\$7C		Vertical Line		<	
1111101	125	\$7D	}	Closing Brace		=	}
1111110	126	\$7E	~	Overline (Tilde)		>	~
1111111	127	\$7F	DEL	Delete/Rubout		?	DEL

**Table E-9.** Control Characters, High Bit On

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
10000000	128	\$80	NUL	Blank (null)	CONTROL+@	@	@
10000001	129	\$81	SOH	Start of Header	CONTROL+A	A	A
10000010	130	\$82	STX	Start of Text	CONTROL+B	B	B
10000011	131	\$83	ETX	End of Text	CONTROL+C	C	C
10000100	132	\$84	EOT	End of Transm.	CONTROL+D	D	D
10000101	133	\$85	ENQ	Enquiry	CONTROL+E	E	E
10000110	134	\$86	ACK	Acknowledge	CONTROL+F	F	F
10000111	135	\$87	BEL	Bell	CONTROL+G	G	G
10001000	136	\$88	BS	Backspace	CONTROL+H or ←	H	H
10001001	137	\$89	HT	Horizontal Tab	CONTROL+I or TAB	I	I
10001010	138	\$8A	LF	Line Feed	CONTROL+J or ↓	J	J
10001011	139	\$8B	VT	Vertical Tab	CONTROL+K or ↑	K	K
10001100	140	\$8C	FF	Form Feed	CONTROL+L	L	L
10001101	141	\$8D	CR	Carriage Return	CONTROL+M or RETURN	M	M
10001110	142	\$8E	SO	Shift Out	CONTROL+N	N	N
10001111	143	\$8F	SI	Shift In	CONTROL+O	O	O
10010000	144	\$90	DLE	Data Link Escape	CONTROL+P	P	P
10010001	145	\$91	DC1	Device Control 1	CONTROL+Q	Q	Q
10010010	146	\$92	DC2	Device Control 2	CONTROL+R	R	R
10010011	147	\$93	DC3	Device Control 3	CONTROL+S	S	S
10010100	148	\$94	DC4	Device Control 4	CONTROL+T	T	T
10010101	149	\$95	NAK	Neg. Acknowledge	CONTROL+U or →	U	U
10010110	150	\$96	SYN	Synchronization	CONTROL+V	V	V
10010111	151	\$97	ETB	End of Text Blk.	CONTROL+W	W	W
10011000	152	\$98	CAN	Cancel	CONTROL+X	X	X
10011001	153	\$99	EM	End of Medium	CONTROL+Y	Y	Y
10011010	154	\$9A	SUB	Substitute	CONTROL+Z	Z	Z
10011011	155	\$9B	ESC	Escape	CONTROL+[ or ESC	[	[
10011100	156	\$9C	FS	File Separator	CONTROL+\	\	\
10011101	157	\$9D	GS	Group Separator	CONTROL+]	]	]
10011110	158	\$9E	RS	Record Separator	CONTROL+^	^	^
10011111	159	\$9F	US	Unit Separator	CONTROL+-	—	—

*Table E-10.* Special Characters, High Bit On

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
10100000	160	\$A0	SP	Space	SPACE bar		
10100001	161	\$A1	!			!	!
10100010	162	\$A2	"			"	"
10100011	163	\$A3	#			#	#
10100100	164	\$A4	\$			\$	\$
10100101	165	\$A5	%			%	%
10100110	166	\$A6	&			&	&
10100111	167	\$A7	,	Closed Quote (acute accent)		,	,
10101000	168	\$A8	(			(	(
10101001	169	\$A9	)			)	)
10101010	170	\$AA	*			*	*
10101011	171	\$AB	+			+	+
10101100	172	\$AC	,	Comma		,	,
10101101	173	\$AD	-	Hyphen		-	-
10101110	174	\$AE	.	Period		.	.
10101111	175	\$AF	/			/	/
10110000	176	\$B0	0			0	0
10110001	177	\$B1	1			1	1
10110010	178	\$B2	2			2	2
10110011	179	\$B3	3			3	3
10110100	180	\$B4	4			4	4
10110101	181	\$B5	5			5	5
10110110	182	\$B6	6			6	6
10110111	183	\$B7	7			7	7
10111000	184	\$B8	8			8	8
10111001	185	\$B9	9			9	9
10111010	186	\$BA	:			:	:
10111011	187	\$BB	;			;	;
10111100	188	\$BC	<			<	<
10111101	189	\$BD	=			=	=
10111110	190	\$BE	>			>	>
10111111	191	\$BF	?			?	?

*Table E-11.* Uppercase Characters, High Bit On

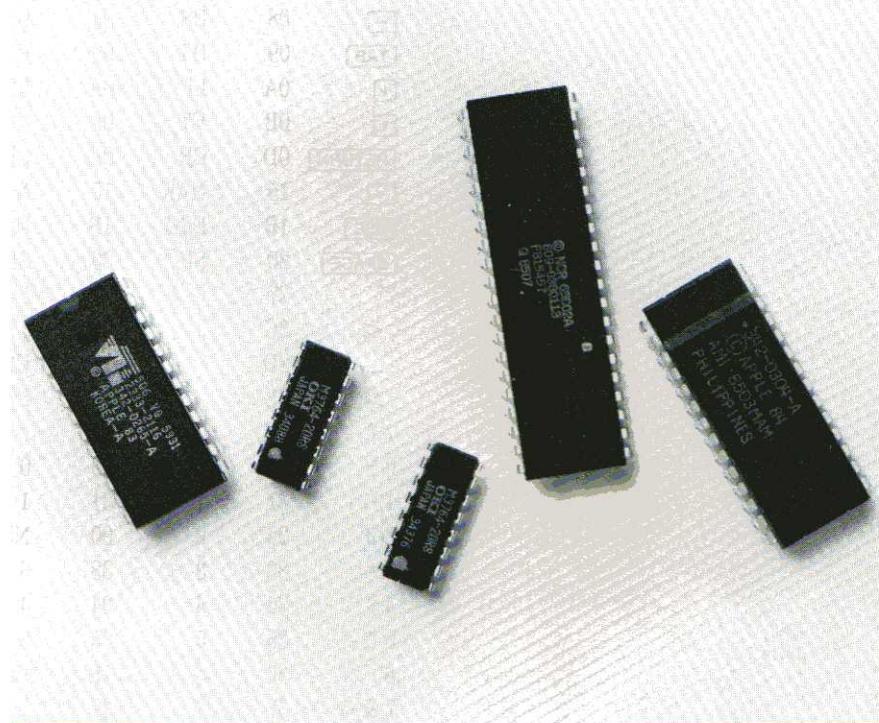
Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
11000000	192	\$C0	@			@	@
11000001	193	\$C1	A			A	A
11000010	194	\$C2	B			B	B
11000011	195	\$C3	C			C	C
11000100	196	\$C4	D			D	D
11000101	197	\$C5	E			E	E
11000110	198	\$C6	F			F	F
11000111	199	\$C7	G			G	G
11001000	200	\$C8	H			H	H
11001001	201	\$C9	I			I	I
11001010	202	\$CA	J			J	J
11001011	203	\$CB	K			K	K
11001100	204	\$CC	L			L	L
11001101	205	\$CD	M			M	M
11001110	206	\$CE	N			N	N
11001111	207	\$CF	O			O	O
11010000	208	\$D0	P			P	P
11010001	209	\$D1	Q			Q	Q
11010010	210	\$D2	R			R	R
11010011	211	\$D3	S			S	S
11010100	212	\$D4	T			T	T
11010101	213	\$D5	U			U	U
11010110	214	\$D6	V			V	V
11010111	215	\$D7	W			W	W
11011000	216	\$D8	X			X	X
11011001	217	\$D9	Y			Y	Y
11011010	218	\$DA	Z			Z	Z
11011011	219	\$DB	[	Opening Bracket	[	[	[
11011100	220	\$DC	\	Reverse Slant	\	\	\
11011101	221	\$DD	]	Closing Bracket	]	]	]
11011110	222	\$DE	^	Caret	^	^	^
11011111	223	\$DF	_	Underline	—	—	—

*Table E-12.* Lowercase Characters, High Bit On

Binary	Dec	Hex	ASCII Char	Interpretation	What to Type	Pri	Alt
11100000	224	\$E0	`	Open Quote		'	'
11100001	225	\$E1	a			a	a
11100010	226	\$E2	b			b	b
11100011	227	\$E3	c			c	c
11100100	228	\$E4	d			d	d
11100101	229	\$E5	e			e	e
11100110	230	\$E6	f			f	f
11100111	231	\$E7	g			g	g
11101000	232	\$E8	h			h	h
11101001	233	\$E9	i			i	i
11101010	234	\$EA	j			j	j
11101011	235	\$EB	k			k	k
11101100	236	\$EC	l			l	l
11101101	237	\$ED	m			m	m
11101110	238	\$EE	n			n	n
11101111	239	\$EF	o			o	o
11110000	240	\$F0	p			p	p
11110001	241	\$F1	q			q	q
11110010	242	\$F2	r			r	r
11110011	243	\$F3	s			s	s
11110100	244	\$F4	t			t	t
11110101	245	\$F5	u			u	u
11110110	246	\$F6	v			v	v
11110111	247	\$F7	w			w	w
11111000	248	\$F8	x			x	x
11111001	249	\$F9	y			y	y
11111010	250	\$FA	z			z	z
11111011	251	\$FB	{	Opening Brace		{	{
11111100	252	\$FC		Vertical Line			
11111101	253	\$FD	}	Closing Brace		}	}
11111110	254	\$FE	~	Overline (Tilde)		~	~
11111111	255	\$FF	DEL	Delete (Rubout)	DELETE	DEL	DEL

## Appendix F

### Frequently Used Tables



This appendix contains copies of the tables you will need to refer to frequently, for example, ASCII codes and soft-switch location. The figures all have their original figure numbers.

*Table 2-3. Keys and ASCII Codes*

*Note: Codes are shown here in hexadecimal; to find the decimal equivalents, refer to Table E-2.*

Key	Normal		Control		Shift		Both	
	Code	Char	Code	Char	Code	Char	Code	Char
DELETE	7F	DEL	7F	DEL	7F	DEL	7F	DEL
-	08	BS	08	BS	08	BS	08	BS
TAB	09	HT	09	HT	09	HT	09	HT
↓	0A	LF	0A	LF	0A	LF	0A	LF
↑	0B	VT	0B	VT	0B	VT	0B	VT
RETURN	0D	CR	0D	CR	0D	CR	0D	CR
→	15	NAK	15	NAK	15	NAK	15	NAK
ESC	1B	ESC	1B	ESC	1B	ESC	1B	ESC
SPACE	20	SP	20	SP	20	SP	20	SP
,	27	,	27	,	22	"	22	"
, <	2C	,	2C	,	3C	<	3C	<
- —	2D	-	1F	US	5F	-	1F	US
. >	2E	.	2E	.	3E	>	3E	>
/ ?	2F	/	2F	/	3F	?	3F	?
0 )	30	0	30	0	29	)	29	)
1 !	31	1	31	1	21	!	21	!
2 @	32	2	00	NUL	40	@	00	NUL
3 #	33	3	33	3	23	#	23	#
4 \$	34	4	34	4	24	\$	24	\$
5 %	35	5	35	5	25	%	25	%
6 ^	36	6	1E	RS	5E	^	1E	RS
7 &	37	7	37	7	26	&	26	&
8 *	38	8	38	8	2A	*	2A	*
9 (	39	9	39	9	28	(	28	(
;) :	3B	;	3B	;	3A	:	3A	:
= +	3D	=	3D	=	2B	+	2B	+
[ {	5B	[	1B	ESC	7B	{	1B	ESC
\ \}	5C	\	1C	FS	7C		1C	FS
] } ~	5D	]	1D	GS	7D	}	1D	GS
	60		60		7E		7E	

*Table 2-3—Continued.* Keys and ASCII Codes

*Note: Codes are shown here in hexadecimal; to find the decimal equivalents, refer to Table E-2.*

Key	Normal		Control		Shift		Both	
	Code	Char	Code	Char	Code	Char	Code	Char
A	61	a	01	SOH	41	A	01	SOH
B	62	b	02	STX	42	B	02	STX
C	63	c	03	ETX	43	C	03	ETX
D	64	d	04	EOT	44	D	04	EOT
E	65	e	05	ENQ	45	E	05	ENQ
F	66	f	06	ACK	46	F	06	ACK
G	67	g	07	BEL	47	G	07	BEL
H	68	h	08	BS	48	H	08	BS
I	69	i	09	HT	49	I	09	HT
J	6A	j	0A	LF	4A	J	0A	LF
K	6B	k	0B	VT	4B	K	0B	VT
L	6C	l	0C	FF	4C	L	0C	FF
M	6D	m	0D	CR	4D	M	0D	CR
N	6E	n	0E	SO	4E	N	0E	SO
O	6F	o	0F	SI	4F	O	0F	SI
P	70	p	10	DLE	50	P	10	DLE
Q	71	q	11	DC1	51	Q	11	DC1
R	72	r	12	DC2	52	R	12	DC2
S	73	s	13	DC3	53	S	13	DC3
T	74	t	14	DC4	54	T	14	DC4
U	75	u	15	NAK	55	U	15	NAK
V	76	v	16	SYN	56	V	16	SYN
W	77	w	17	ETB	57	W	17	ETB
X	78	x	18	CAN	58	X	18	CAN
Y	79	y	19	EM	59	Y	19	EM
Z	7A	z	1A	SUB	5A	Z	1A	SUB

*Table 2-2.* Keyboard Memory Locations

Location		
Hex	Decimal	Description
\$C000	49152	-16384
\$C010	49168	-16368
		Keyboard data and strobe
		Any-key-down flag and clear-strobe switch

*Table 2-4.* Video Display Specifications

Display modes:	40-column text; map: Figure 2-2 80-column text; map: Figure 2-3  Low-resolution color graphics; map: Figure 2-7  High-resolution color graphics; map: Figure 2-8  Double-high-resolution color graphics; map: Figure 2-9
Text capacity:	24 lines by 80 columns (character positions)
Character set:	96 ASCII characters (uppercase and lowercase)
Display formats:	Normal, inverse, flashing, MouseText (Table 2-5)
Low-resolution graphics:	16 colors (Table 2-6) 40 horizontal by 48 vertical; map: Figure 2-7
High-resolution graphics:	6 colors (Table 2-7) 140 horizontal by 192 vertical (restricted)  Black-and-white: 280 horizontal by 192 vertical; map: Figure 2-8
Double-high-resolution graphics:	16 colors (Table 2-8) 140 horizontal by 192 vertical (no restrictions)  Black-and-white: 560 horizontal by 192 vertical; map: Figure 2-9

*Table 2-8.* Double-High-Resolution Graphics Colors

Color	ab0	mb1	ab2	mb3	Repeated Bit Pattern
Black	\$00	\$00	\$00	\$00	0000
Magenta	\$08	\$11	\$22	\$44	0001
Brown	\$44	\$08	\$11	\$22	0010
Orange	\$4C	\$19	\$33	\$66	0011
Dark Green	\$22	\$44	\$08	\$11	0100
Gray 1	\$2A	\$55	\$2A	\$55	0101
Green	\$66	\$4C	\$19	\$33	0110
Yellow	\$6E	\$5D	\$3B	\$77	0111
Dark Blue	\$11	\$22	\$44	\$08	1000
Purple	\$19	\$33	\$66	\$4C	1001
Gray 2	\$55	\$2A	\$55	\$2A	1010
Pink	\$5D	\$3B	\$77	\$6E	1011
Medium Blue	\$33	\$66	\$4C	\$19	1100
Light Blue	\$3B	\$77	\$6E	\$5D	1101
Aqua	\$77	\$6E	\$5D	\$3B	1110
White	\$7F	\$7F	\$7F	\$7F	1111

*Table 2-9.* Video Display Page Locations

Display Mode			Highest Address			
			Hex	Dec	Hex	Dec
40-column text, low-resolution graphics	1 2 *	\$0400 \$0800	1024 2048	\$07FF \$0BFF	2047 3071	
80-column text	1 2 *	\$0400 \$0800	1024 2048	\$07FF \$0BFF	2047 3071	
High-resolution graphics	1 2	\$2000 \$4000	8192 16384	\$3FFF \$5FFF	16383 24575	
Double-high- resolution graphics	1 † 2 †	\$2000 \$4000	8192 16384	\$3FFF \$5FFF	16383 24575	

\* This is not supported by firmware; for instructions on how to switch pages, refer to the section “Display Mode Switching” in Chapter 2.

† See the section “Double-High-Resolution Graphics,” in Chapter 2.

**Table 2-10.** Display Soft Switches

Note: W means write anything to the location, R means read the location, R/W means read or write, and R7 means read the location and then check bit 7.

Name	Action	Hex	Function
ALTCHAR	W	\$C00E	Off: display text using primary character set
ALTCHAR	W	\$C00F	On: display text using alternate character set
RDALTCHAR	R7	\$C01E	Read ALTCHAR switch (1 = on)
80COL	W	\$C00C	Off: display 40 columns
80COL	W	\$C00D	On: display 80 columns
RD80COL	R7	\$C01F	Read 80COL switch (1 = on)
80STORE	W	\$C000	Off: cause PAGE2 on to select auxiliary RAM
80STORE	W	\$C001	On: allow PAGE2 to switch main RAM areas
RD80STORE	R7	\$C018	Read 80STORE switch (1 = on)
PAGE2	R/W	\$C054	Off: select Page 1
PAGE2	R/W	\$C055	On: select Page 2 or, if 80STORE on, Page 1 in auxiliary memory
RDPAGE2	R7	\$C01C	Read PAGE2 switch (1 = on)
TEXT	R/W	\$C050	Off: display graphics or, if MIXED on, mixed
TEXT	R/W	\$C051	On: display text
RDTEXT	R7	\$C01A	Read TEXT switch (1 = on)
MIXED	R/W	\$C052	Off: display only text or only graphics
MIXED	R/W	\$C053	On: if TEXT off, display text and graphics
RDMIXED	R7	\$C01B	Read MIXED switch (1 = on)
HIRES	R/W	\$C056	Off: if TEXT off, display low-resolution graphics
HIRES	R/W	\$C059	On: if TEXT off, display high-resolution or, if DHires on, double-high-resolution graphics
RDHIRES	R7	\$C01D	Read HIRES switch (1 = on)
IOUDIS	W	\$C07E	On: disable IOU access for addresses \$C058 to \$C05F; enable access to DHires switch *
IOUDIS	W	\$C07F	Off: enable IOU access for addresses \$C058 to \$C05F; disable access to DHires switch *
RDIUDIS	R7	\$C07E	Read IOUDIS switch (1 = off) †
DHires	R/W	\$C05E	On: (if IOUDIS on) turn on double-high-res.
DHires	R/W	\$C05F	Off: (if IOUDIS on) turn off double-high-res.
RDDHires	R7	\$C07F	Read DHires switch (1 = on) †

\* The firmware normally leaves IOUDIS on. See also †.

† Reading or writing any address in the range \$C070-\$C07F also triggers the paddle timer and resets VBLINT (Chapter 7).

*Table 3-1.* Monitor Firmware Routines

Location	Name	Description
\$C305	BASICIN	With 80-column firmware active, displays solid, blinking cursor. Accepts character from keyboard.
\$C307	BASICOUT	Displays a character on the screen; used when the 80-column firmware is active (Chapter 3).
\$FC9C	CLREOL	Clears to end of line from current cursor position.
\$FC9E	CLEOLZ	Clears to end of line using contents of Y register as cursor position.
\$FC42	CLREOP	Clears to bottom of window.
\$F832	CLRSCR	Clears the low-resolution screen.
\$F836	CLRTOP	Clears top 40 lines of low-resolution screen.
\$FDED	COUT	Calls output routine whose address is stored in CSW (normally COUT1, Chapter 3).
\$FDF0	COUT1	Displays a character on the screen (Chapter 3).
\$FD8E	CROUT	Generates a carriage return character.
\$FD8B	CROUT1	Clears to end of line, then generates a carriage return character.
\$FD6A	GETLN	Displays the prompt character; accepts a string of characters by means of RDKEY.
\$F819	HLINE	Draws a horizontal line of blocks.
\$FC58	HOME	Clears window; puts cursor in upper-left corner of window.
\$FD1B	KEYIN	With 80-column firmware inactive, displays checkerboard cursor. Accepts character from keyboard.
\$F800	PLOT	Plots a single low-resolution block on the screen.
\$F94A	PRBL2	Sends 1 to 256 blank spaces to the output device.
\$FDDA	PRBYTE	Prints a hexadecimal byte.
\$FF2D	PRERR	Sends <b>ERR</b> and Control-G to the output device.
\$FDE3	PRHEX	Prints 4 bits as a hexadecimal number.
\$F941	PRNTAX	Prints contents of A and X in hexadecimal.
\$FD0C	RDKEY	Displays blinking cursor; goes to standard input routine, normally KEYIN or BASICIN.
\$F871	SCRN	Reads color value of a low-resolution block.
\$F864	SETCOL	Sets the color for plotting in low-resolution.
\$FC24	VTABZ	Sets cursor vertical position.
\$F828	VLINE	Draws a vertical line of low-resolution blocks.

*Table 3-3a.* Control Characters With 80-Column Firmware Off

Control Character	ASCII Name	Apple IIe Name	Action Taken by COUT1
Control-G	BEL	bell	Produces a 1000 Hz tone for 0.1 second.
Control-H	BS	backspace	Moves cursor position one space to the left; from left edge of window, moves to right end of line above.
Control-J	LF	line feed	Moves cursor position down to next line in window; scrolls if needed.
Control-M	CR	return	Moves cursor position to left end of next line in window; scrolls if needed.

*Table 3-3b.* Control Characters With 80-Column Firmware On

Control Character	ASCII Name	Apple IIe Name	Action Taken by BASICOUT
Control-G	BEL	bell	Produces a 1000 Hz tone for 0.1 second.
Control-H	BS	backspace	Moves cursor position one space to the left; from left edge of window, moves to right end of line above.
Control-J	LF	line feed	Moves cursor position down to next line in window; scrolls if needed.
Control-K †	VT	clear EOS	Clears from cursor position to the end of the screen.
Control-L †	FF	home and clear	Moves cursor position to upper-left corner of window and clears window.
Control-M	CR	return	Moves cursor position to left end of next line in window; scrolls if needed.
Control-N †	SO	normal	Sets display format normal.
Control-O †	SI	inverse	Sets display format inverse.
Control-Q †	DC1	40-column	Sets display to 40-column.
Control-R †	DC2	80-column	Sets display to 80-column.
Control-S *	DC3	stop-list	Stops listing characters on the display until another key is pressed.

*Table 3-3b—Continued.* Control Characters With 80-Column Firmware On

Control Character	ASCII Name	Apple IIe Name	Action Taken by BASICOUT
Control-U †	NAK	quit	Deactivates 80-column video firmware.
Control-V †	SYN	scroll	Scrolls the display down one line, leaving the cursor in the current position.
Control-W †	ETB	scroll-up	Scrolls the display up one line, leaving the cursor in the current position.
Control-X	CAN	disable MouseText	Disable MouseText character display; use inverse uppercase.
Control-Y †	EM	home	Moves cursor position to upper-left corner of window (but doesn't clear).
Control-Z †	SUB	clear line	Clears the line the cursor position is on.
Control-[	ESC	enable MouseText	Map inverse uppercase characters to MouseText characters.
Control-\†	FS	forward space	Moves cursor position one space to the right; from right edge of window, moves it to left end of line below.
Control-]†	GS	clear EOL	Clears from the current cursor position to the end of the line (that is, to the right edge of the window).
Control-_	US	up	Moves cursor up a line, no scroll.

\* Only works from the keyboard.

† Doesn't work from the keyboard.

*Table 3-5.* Text Format Control Values

*Note:* These mask values apply only to the primary character set (see text).

Mask Value		
Dec	Hex	Display Format
255	\$FF	Normal, uppercase, and lowercase
127	\$7F	Flashing, uppercase, and symbols
63	\$3F	Inverse, uppercase, and lowercase

*Table 3-6.* Escape Codes

Escape Code	Function
<code>ESC @</code>	Clears window and homes cursor (places it in upper-left corner of screen), then exits from escape mode.
<code>ESC A</code> or <code>a</code>	Moves cursor right one line; exits from escape mode.
<code>ESC B</code> or <code>b</code>	Moves cursor left one line; exits from escape mode.
<code>ESC C</code> or <code>c</code>	Moves cursor down one line; exits from escape mode.
<code>ESC D</code> or <code>d</code>	Moves cursor up one line; exits from escape mode.
<code>ESC E</code> or <code>e</code>	Clears to end of line; exits from escape mode.
<code>ESC F</code> or <code>f</code>	Clears to bottom of window; exits from escape mode.
<code>ESC I</code> or <code>i</code> or <code>ESC ↑</code>	Moves the cursor up one line; remains in escape mode. See text.
<code>ESC J</code> or <code>j</code> or <code>ESC ←</code>	Moves the cursor left one space; remains in escape mode. See text.
<code>ESC K</code> or <code>k</code> or <code>ESC →</code>	Moves the cursor right one space; remains in escape mode. See text.
<code>ESC M</code> or <code>m</code> or <code>ESC ↓</code>	Moves the cursor down one line; remains in escape mode. See text.
<code>ESC 4</code>	If 80-column firmware is active, switches to 40-column mode; sets links to BASICIN and BASICOUT; restores normal window size; exits from escape mode.
<code>ESC 8</code>	If 80-column firmware is active, switches to 80-column mode; sets links to BASICIN and BASICOUT; restores normal window size; exits from escape mode.
<code>ESC CONTROL-D</code>	Disables control characters; only carriage return, line feed, BELL, and backspace have an effect when printed.
<code>ESC CONTROL-E</code>	Reactivates control characters.
<code>ESC CONTROL-Q</code>	If 80-column firmware is active, deactivates 80-column firmware; sets links to KEYIN and COUT1; restores normal window size; exits from escape mode.

*Table 3-10.* Pascal Video Control Functions

<b>Control-</b>	<b>Hex</b>	<b>Function performed</b>
E or e	\$05	Turns cursor on (enables cursor display).
F or f	\$06	Turns cursor off (disables cursor display).
G or g	\$07	Sounds bell (beeps).
H or h	\$08	Moves cursor left one column. If cursor was at beginning of line, moves it to end of previous line.
J or j	\$0A	Moves cursor down one row; scrolls if needed.
K or k	\$0B	Clears to end of screen.
L or l	\$0C	Clears screen; moves cursor to upper-left of screen.
M or m	\$0D	Moves cursor to column 0.
N or n	\$0E	Displays subsequent characters in normal video. (Characters already on display are unaffected.)
O or o	\$0F	Displays subsequent characters in inverse video. (Characters already on display are unaffected.)
V or v	\$16	Scrolls screen up one line; clears bottom line.
W or w	\$17	Scrolls screen down one line; clears top line.
Y or y	\$19	Moves cursor to upper-left (home) position on screen.
Z or z	\$1A	Clears entire line that cursor is on.
\ or \	\$1C	Moves cursor right one column; if at end of line, does Control-M.
} or ]	\$1D	Clears to end of the line the cursor is on, including current cursor position; does not move cursor.
^ or 6	\$1E	GOTOxy: initiates a GOTOxy sequence; interprets the next two characters as x+32 and y+32, respectively.
—	\$1F	If not at top of screen, moves cursor up one line.

**Table 4-6.** Bank Select Switches

*Note: R means read the location, W means write anything to the location, R/W means read or write, and R7 means read the location and then check bit 7.*

Name	Action	Hex	Function
	R	\$C080	Read RAM; no write; use \$D000 bank 2.
	RR	\$C081	Read ROM; write RAM; use \$D000 bank 2.
	R	\$C082	Read ROM; no write; use \$D000 bank 2.
	RR	\$C083	Read and write RAM; use \$D000 bank 2.
	R	\$C088	Read RAM; no write; use \$D000 bank 1.
	RR	\$C089	Read ROM; write RAM; use \$D000 bank 1.
	R	\$C08A	Read ROM; no write; use \$D000 bank 1.
	RR	\$C08B	Read and write RAM; use \$D000 bank 1.
RDBNK2	R7	\$C011	Read whether \$D000 bank 2 (1) or bank 1 (0)
RDLCRAM	R7	\$C012	Reading RAM (1) or ROM (0).
ALTZP	W	\$C008	Off: use main bank, page 0 and page 1.
ALTZP	W	\$C009	On: use auxiliary bank, page 0 and page 1.
RDALTZP	R7	\$C016	Read whether auxiliary (1) or main (0) bank

*Table 4-7.* Auxiliary-Memory Select Switches

Name	Function	Location			Notes
		Hex	Decimal		
RAMRD	Read auxiliary memory	\$C003	49155	-16381	Write
	Read main memory	\$C002	49154	-16382	Write
	Read RAMRD switch	\$C013	49171	-16365	Read
RAMWRT	Write auxiliary memory	\$C005	49157	-16379	Write
	Write main memory	\$C004	49156	-16380	Write
	Read RAMWRT switch	\$C014	49172	-16354	Read
80STORE	On: access display page	\$C001	49153	-16383	Write
	Off: use RAMRD, RAMWRT	\$C000	49152	-16384	Write
	Read 80STORE switch	\$C018	49176	-16360	Read
PAGE2	Page 2 on (aux. memory)	\$C055	49237	-16299	*
	Page 2 off (main memory)	\$C054	49236	-16300	*
	Read PAGE2 switch	\$C01C	49180	-16356	Read
HIRES	On: access high-res. pages	\$C057	49239	-16297	†
	Off: use RAMRD, RAMWRT	\$C056	49238	-16298	†
	Read HIRES switch	\$C01D	49181	-16355	Read
ALTZP	Auxiliary stack & z.p.	\$C009	49161	-16373	Write
	Main stack & zero page	\$C008	49160	-16374	Write
	Read ALTZP switch	\$C016	49174	-16352	Read

\* When 80STORE is on, the PAGE2 switch selects main or auxiliary display memory.

† When 80STORE is on, the HIRES switch enables you to use the PAGE2 switch to switch between the high-resolution Page-1 area in main memory or auxiliary memory.

*Table 4-8.* 48K RAM Transfer Routines

Name	Action	Hex	Function
AUXMOVE	JSR	\$C312	Moves data blocks between main and auxiliary 48K memory.
XFER	JMP	\$C314	Transfers program control between main and auxiliary 48K memory.