Display:

The Apple Computer outputs a composite video signal (composite of sync and video information) which can be applied to any standard raster-scan type video display monitor. The output level is adjustable with the potentiometer located near the video output Molex connector, J2. The additional two outside pins on the Molex connector supply +5 and +12 volts, to be used in future Apple accessories. The composite video signal can also be modulated at the proper RF frequency, with an inexpensive commercially available device, and applied to the antenna terminals of a home television receiver. Since the character format is 40 characters / line, all television receivers will have the necessary bandwidth to display the entire 40 characters. Two large manufacturers of video display monitors, which connect directly with the Apple Computer, are Motorola and Ball. The mating four-pin Molex connector is provided.

AC Power Sources:

Two incoming AC power sources are required for operation: 8 to 10 VAC (RMS) at 3 amps, and 28 VAC (RMS) Center-Tapped at 1 amp. These AC supplies enter the system at the Molex connector, J1. The 8 to 10 volts AC provides the raw AC for the +5 volt supply, while the 28 VCT supplies the raw AC for the +12 and -12 volt supplies, and the -5V supply is derived from the -12V regulated output.

The board, as supplied, requires no more than 1.5 amps DC from the +5V supply, while the regulator is capable of supplying 3 amps. The remaining 1.5 amps DC from the +5V supply is available for user hardware expansion (provided suitable transformer ratings are employed).

A suitable source of the raw AC voltages required, are two commercially available transformers; Stancor P/N P-8380 or equivalent (8 to 10 volts at 3 amps), and Stancor P/N P-8667 or equivalent (28VCT at 1 amp). Simply wire the secondaries to the mating six-pin Molex connector supplied, and wire the primaries in parallel, as shown in the schematic diagram (power supply section, Dwg. No. 00101, sheet 3 of 3.

TEST PROGRAM

After attaching the keyboard, display, and AC power sources, you can try a simple program to test if your system and the attachments are functioning together properly. While it does not test many possible areas of the microprocessor system, the test program will test for the correct attachment of the keyboard, display, and power supplies.

FIRST:
Hit the RESET button to enter the system monitor. A backslash should be displayed, and the cursor should drop to the next line.

SECOND:
Type- Ø : A9 b Ø b AA b 2Ø b EF b FF b E8 b 8A b 4C b 2 b Ø (RET)
(Ø is a zero, NOT an alpha "O"; b means blank or space; and (RET) hit the "return" key on the keyboard)

THIRD:
Type- Ø . A (RET)
(This should print out, on the display, the program you have just entered.)

FOURTH:
Type- R (RET)
(R means run the program.)

THE PROGRAM SHOULD THEN PRINT OUT ON THE DISPLAY A CONTINUOUS STREAM OF ASCII CHARACTERS. TO STOP THE PROGRAM AND RETURN TO THE SYSTEM MONITOR, HIT THE "RESET" BUTTON. TO RUN AGAIN, TYPE : R (RET).

SECTION II USING THE SYSTEM MONITOR

The Hex Monitor is a PROM program in locations FFØØ to FFFF (hex) which uses the keyboard and display to perform the front panel functions of examining memory, and running programs. The monitor program is entered by hitting (RESET), which displays backslash - return. A backslash alone (cursor remains on same line as backslash) indicates bad page 0 RAM.

Commands are typed on a "line-at-a-time" basis with editing. Each line may consist of any number of commands (up to 128 characters). None are executed until (RETURN) is typed. The (SHIFT-0) (backarrow) backspaces and echos an underline. The (ESC) cnacels a line and echos backslash-return.

One or more hexadecimal digits (0-9, A-F) are used for address and data values. Addresses use the four least significant digits of a group, and data values, the two least significant digits. The following examples illustrate the variety of acceptable commands:

1. Opening a location (examining the contents of a single address).

```
    USER TYPES/     4F (RET)
MONITOR TYPES/      ØØ4F: ØF (contents
                               of 4F)
```

2. Examining a block; from the last examined location, to a specified one.

```
    USER TYPES/     .5A (RET)
MONITOR TYPES/
    ØØ5Ø: ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7
    ØØ58: Ø8 Ø9 ØA
```

Note: 4F is still considered the most recently opened location.

3. Combining examples 1 and 2 to print a block of memory in a single command.

```
    USER TYPES/   4F.5A (RET)
MONITOR TYPES/
    ØØ5Ø: ØØ Ø1 Ø2 Ø3 Ø4 Ø5 Ø6 Ø7
    ØØ58: Ø8 Ø9 ØA
```

Note: Only the first location of the block (4F) is considered "opened".

4. Examining several individual locations at once.

```
    USER TYPES/   4F b 52 b 56 (RET)
MONITOR TYPES/    ØØ4F: ØF
                  ØØ52: Ø2
                  ØØ56: Ø6
```

Note: 56 is considered the most recently "opened" location. The "b" is a blank or comma, and is a delimiter for separation purposes only. A string of delimiters has the same effect as a single one (bbb is as effective as b).

5. Examining several blocks of memory at once.

```
    USER TYPES/    4F.52 b 56 b 58.5A
                   (RET)
MONITOR TYPES/     ØØ4F: ØF
                   ØØ5Ø: ØØ Ø1 Ø2
                   ØØ56: Ø6
                   ØØ58: Ø8 Ø9 ØA
```

Note: 58 is considered the most recently "opened" location. Refer to example 2.

6. Examining successive blocks.

```
    USER TYPES/    4F.52 (RET)
MONITOR TYPES/     ØØ4F: ØF
                   ØØ5Ø: ØØ Ø1 Ø2
    USER TYPES/    .55 (RET)
MONITOR TYPES/     ØØ53: Ø3 Ø4 Ø5
    USER TYPES/    .5A (RET)
MONITOR TYPES/     ØØ56: Ø6 Ø7
                   ØØ58: Ø8 Ø9 ØA
```

7. Depositing data in a single location.

```
    USER TYPES/    3Ø: AØ (RET)
MONITOR TYPES/     ØØ3Ø: FF (prior
                           contents)
```

Note: Location 3Ø is considered opened and now contains 3Ø.

8. Depositing data in successive locations from that last used in a deposit command.

```
    USER TYPES/    : A1 b A2 b A3 b A4
                   b A5 (RET)
```

(This deposits A1 in location 31, A2 in 32, and so on.)

9. Combining examples 7 and 8 in a single command.

```
    USER TYPES/    3Ø: AØ b A1 b A2 b
                   A3 b A4 b A5 (RET)
MONITOR TYPES/
    ØØ3Ø: FF (prior contents of location 3Ø)
```

10. Depositing data in successive locations with separate commands.

```
    USER TYPES/    3Ø: AØ b A1 (RET)
MONITOR TYPES/     ØØ3Ø: FF
    USER TYPES/    :A2 b A3 (RET)
    USER TYPES/    :A4 b A5 (RET)
```

NOTE: Capital letters enclosed in parenthesis represent single keystrokes.
Example: (RET) means hit the "return" key.

Note: A colon in a command means "start depositing data from the most recently deposited location, or if none, then from the most recently opened one.

11. Examining a block, then depositing into it.
```
    USER TYPES/      3Ø.35 (RET)
MONITOR TYPES/
    ØØ3Ø: AØ A1 A2 A3 A4 A5 A6
    USER TYPES/
    :B0 b B1 b B2 b B3 b B4 b B5 (RET)
```

Note: New data deposited beginning at most recently opened location (3Ø)

12. Run a program at a specified address.
```
    USER TYPES/      1ØFØ R (RET)
MONITOR TYPES/       1ØFØ: A9 (contents)
```

Note: The cursor is left immediately to the right of the "A9"; it is not returned to the next line.

13. Run at the most recently examined location.
```
    USER TYPES/      1ØFØ (RET)
MONITOR TYPES/       1ØFØ: A9
    USER TYPES/      R (RET)
```

14. Enter a program into memory and run it in one line.
```
    USER TYPES/
    4Ø: A9 b Ø b 2Ø b EF b FF b 38 b 69 b
    Ø b 4C b 4Ø b Ø R (RET)
MONITOR TYPES/       40: FF (prior con-
                          tents of 4Ø)
MONITOR TYPES/       4Ø: FF (prior con-
                          tents of 4Ø)
```

15. An "on line" error correction.
```
    USER TYPES/
    4Ø: A1 b A2 b A3A4A5A6 b A7
(data A6 will be loaded in location 42)
    USER TYPES/      4Ø5Ø6Ø7Ø: AA
(data AA will be loaded in location 6Ø7Ø)
```

16. Useful routines in monitor which can be accessed by user programs.

GETLINE: location FF1F:
monitor entry point
(jumping to FF1F will enter monitor and echo carriage return. You can then examine memory locations with the monitor.)

ECHO: location FFEF:
prints one byte (ASCII)
(data from "A" (accumulator), contents of "A" not disturbed. Example: 2Ø b EF b FF (JRS ECHO)).

PRBYTE: location FFDC:
prints one byte (HEX)
(data from "A", contents of "A" disturbed.)

PRHEX: location FFE5:
prints one hex digit
(data from four least significant bits of "A", contents of "A" disturbed.)

NOTE: RAM locations ØØ24 to ØØ2B are used as index pointers by the monitor, and are invalid for user use, when using monitor. Also, locations Ø2ØØ to Ø27F are used as input buffer storage, and are also invalid for user use when using the monitor.

6502 HEX MONITOR LISTING

```
FF00  D8          RESET       CLD              Clear decimal arithmetic mode.
FF01  58                      CLI
FF02  A0 7F                   LDY #$7F         Mask for DSP data direction register.
FF04  8C 12 D0                STY DSP          Set it up.
FF07  A9 A7                   LDA #$A7         KBD and DSP control register mask.
FF09  8D 11 D0                STA KBD CR       Enable interrupts, set CA1, CB1, for
FF0C  8D 13 D0                STA DSP CR         positive edge sense/output mode.
FF0F  C9 DF       NOTCR       CMP #$DF         "←"?
FF11  F0 13                   BEQ BACKSPACE    Yes.
FF13  C9 9B                   CMP #$9B         ESC?
FF15  F0 03                   BEQ ESCAPE       Yes.
FF17  C8                      INY              Advance text index.
FF18  10 0F                   BPL NEXTCHAR     Auto ESC if > 127.
FF1A  A9 DC       ESCAPE      LDA #$DC         "\".
FF1C  20 EF FF                JSR ECHO         Output it.
FF1F  A9 8D       GETLINE     LDA #$8D         CR.
FF21  20 EF FF                JSR ECHO         Output it.
FF24  A0 01                   LDY #$01         Initiallize text index.
FF26  88          BACKSPACE   DEY              Back up text index.
FF27  30 F6                   BMI GETLINE      Beyond start of line, reinitialize.
FF29  AD 11 D0    NEXTCHAR    LDA KBD CR       Key ready?
FF2C  10 FB                   BPL NEXTCHAR     Loop until ready.
FF2E  AD 10 D0                LDA KBD          Load character. B7 should be '1'.
FF31  99 00 02                STA IN, Y        Add to text buffer.
FF34  20 EF FF                JSR ECHO         Display character.
FF37  C9 8D                   CMP #$8D         CR?
FF39  D0 D4                   BNE NOTCR        No.
FF3B  A0 FF                   LDY #$FF         Reset text index.
FF3D  A9 00                   LDA #$00         For XAM mode.
FF3F  AA                      TAX              0→X.
FF40  0A          SETSTOR     ASL              Leaves $7B if setting STOR mode.
FF41  85 2B       SETMODE     STA MODE         $00 = XAM, $7B = STOR, $AE = BLOK XAM.
FF43  C8          BLSKIP      INY              Advance text index.
FF44  B9 00 02    NEXT ITEM   LDA IN, Y        Get character.
FF47  C9 8D                   CMP #$8D         CR?
FF49  F0 D4                   BEQ GETLINE      Yes, done this line.
FF4B  C9 AE                   CMP #$AE         "."?
FF4D  90 F4                   BCC BLSKIP       Skip delimiter.
FF4F  F0 F0                   BEQ SETMODE      Set BLOCK XAM mode.
FF51  C9 BA                   CMP #$BA         ":"?
FF53  F0 EB                   BEQ SETSTOR      Yes, set STOR mode.
FF55  C9 D2                   CMP #$D2         "R"?
FF57  F0 3B                   BEQ RUN          Yes, run user program.
FF59  86 28                   STX L            $00→ L.
FF5B  86 29                   STX H              and H.
FF5D  84 2A                   STY YSAV         Save Y for comparison.
FF5F  B9 00 02    NEXTHEX     LDA IN, Y        Get character for hex test.
FF62  49 B0                   EOR #$B0         Map digits to $0-9.
FF64  C9 0A                   CMP #$0A         Digit?
FF66  90 06                   BCC DIG          Yes.
FF68  69 88                   ADC #$88         Map letter "A"-"F" to $FA-FF.
FF6A  C9 FA                   CMP #$FA         Hex letter?
FF6C  90 11                   BCC NOTHEX       No, character not hex.
FF6E  0A          DIG         ASL
FF6F  0A                      ASL              Hex digit to MSD of A.
FF70  0A                      ASL
FF71  0A                      ASL
FF72  A2 04                   LDX #$04         Shift count.
FF74  0A          HEXSHIFT    ASL              Hex digit left, MSB to carry.
```

6502 HEX MONITOR LISTING (continued)

```
FF75  26 28                  ROL L              Rotate into LSD.
FF77  26 29                  ROL H              Rotate into MSD's.
FF79  CA                     DEX                Done 4 shifts?
FF7A  DØ F8                  BNE HEXSHIFT       No, loop.
FF7C  C8                     INY                Advence text index.
FF7D  DØ EØ                  BNE NEXTHEX        Always taken. Check next character for hex.
FF7F  C4 2A     NOTHEX       CPY YSAV           Check if L, H empty (no hex digits).
FF81  FØ 97                  BEQ ESCAPE         Yes, generate ESC sequence.
FF83  24 2B                  BIT MODE           Test MODE byte.
FF85  5Ø 1Ø                  BVC NOTSTOR        B6 = Ø for STOR, 1 for XAM and BLOCK XAM
FF87  A5 28                  LDA L              LSD's of hex data.
FF89  81 26                  STA (STL,X)        Store at current 'store index'.
FF8B  E6 26                  INC STL            Increment store index.
FF8D  DØ B5                  BNE NEXTITEM       Get next item. (no carry).
FF8F  E6 27                  INC STH            Add carry to 'store index' high order.
FF91  4C 44 FF  TONEXTITEM   JMP NEXTITEM       Get next command item.
FF94  6C 24 ØØ  RUN          JMP (XAML)         Run at current XAM index.
FF97  3Ø 2B     NOTSTOR      BMI XAMNEXT        B7 = Ø for XAM, 1 for BLOCK XAM.
FF99  A2 Ø2                  LDX #$Ø2           Byte count.
FF9B  B5 27     SETADR       LDA L-1,X          Copy hex data to
FF9D  95 25                  STA STL-1,X          'store index'.
FF9F  95 23                  STA XAML-1,X       And to 'XAM index'.
FFA1  CA                     DEX                Next of 2 bytes.
FFA2  DØ F7                  BNE SETADR         Loop unless X = Ø.
FFA4  DØ 14     NXTPRNT      BNE PRDATA         NE means no address to print.
FFA6  A9 8D                  LDA #$8D           CR.
FFA8  2Ø EF FF               JSR ECHO           Output it.
FFAB  A5 25                  LDA XAMH           'Examine index' high-order byte.
FFAD  2Ø DC FF               JSR PRBYTE         Output it in hex format.
FFBØ  A5 24                  LDA XAML           Low-order 'examine index' byte.
FFB2  2Ø DC FF               JSR PRBYTE         Output it in hex format.
FFB5  A9 BA                  LDA #$BA           ":".
FFB7  2Ø EF FF               JSR ECHO           Output it.
FFBA  A9 AØ     PRDATA       LDA #$AØ           Blank.
FFBC  2Ø EF FF               JSR ECHO           Output it.
FFBF  A1 24                  LDA (XAML,X)       Get data byte at 'examine index'.
FFC1  2Ø DC FF               JSR PRBYTE         Output it in hex format.
FFC4  86 2B     XAMNEXT      STX MODE           Ø→ MODE (XAM mode).
FFC7  A5 24                  LDA XAML
FFC8  C5 28                  CMP L              Compare 'examine index' to hex data.
FFCA  A5 25                  LDA XAMH
FFCC  E5 29                  SBC H
FFCE  BØ C1                  BCS TONEXTITEM     Not less, so no more data to output.
FFDØ  E6 24                  INC XAML
FFD2  DØ Ø2                  BNE MOD8CHK        Increment 'examine index'.
FFD4  E6 25                  INC XAMH
FFD6  A5 24     MOD8CHK      LDA XAML           Check low-order 'examine index' byte
FFD8  29 Ø7                  AND #$Ø7             For MOD 8 = Ø
FFDA  1Ø C8                  BPL NXTPRNT        Always taken.
FFDC  48        PRBYTE       PHA                Save A for LSD.
FFDD  4A                     LSR
FFDE  4A                     LSR
FFDF  4A                     LSR                MSD to LSD position.
FFEØ  4A                     LSR
FFE1  2Ø E5 FF               JSR PRHEX          Output hex digit.
FFE4  68                     PLA                Restore A.
FFE5  29 ØF     PRHEX        AND #$ØF           Mask LSD for hex print.
FFE7  Ø9 BØ                  ORA #$BØ           Add "Ø".
FFE9  C9 BA                  CMP #$BA           Digit?
```

```
FFEB  9Ø Ø2                  BCC ECHO        Yes, output it.
FFED  69 Ø6                  ADC #$Ø6        Add offset for letter.
FFEF  2C 12 DØ     ECHO      BIT DSP         DA bit (B7) cleared yet?
FFF2  3Ø FB                  BMI ECHO        No, wait for display.
FFF4  8D 12 DØ               STA DSP         Output character.  Sets DA.
FFF7  6Ø                     RTS             Return.
FFF8  ØØ ØØ (unused)
FFFA  ØØ ØF (NMI)
FFFC  ØØ FF(RESET)
FFFE  ØØ ØØ (IRQ)
```

HARDWARE NOTES

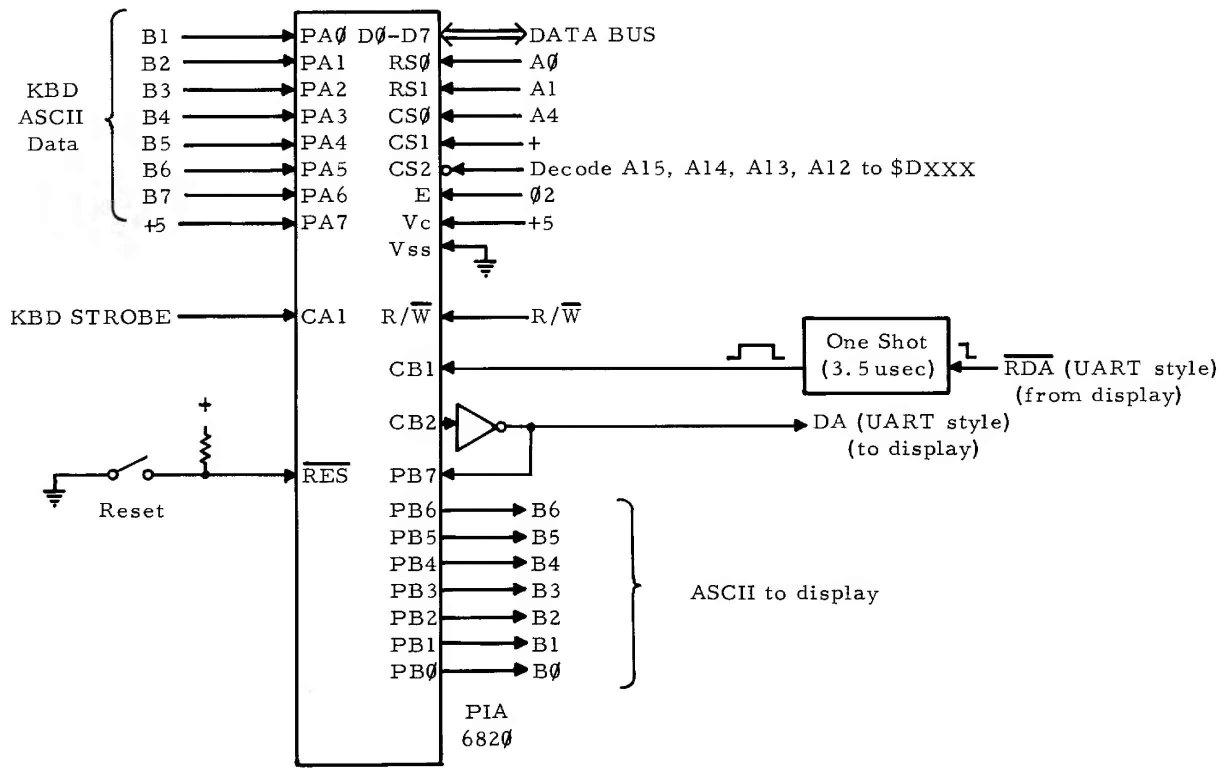
Page Ø Variables

XAML	24
XAMH	25
STL	26
STH	27
L	28
H	29
YSAV	2A
MODE	2B

Other Variables

IN	2ØØ-27F	
KBD	DØ1Ø	PIA
KBD CR	DØ11	PIA
DSP	DØ12	PIA
DSP CR	DØ13	PIA

KBD/DSP Interface


KBD ASCII Data
B1
B2
B3
B4
B5
B6
B7
+5
PAØ
PA1
PA2
PA3
PA4
PA5
PA6
PA7
DØ-D7
RSØ
RS1
CSØ
CS1
CS2
E
Vc
Vss
DATA BUS
AØ
A1
A4
+
Decode A15, A14, A13, A12 to $DXXX
Ø2
+5
KBD STROBE
CA1
R/$\overline{W}$
R/$\overline{W}$
CB1
One Shot (3.5 usec)
$\overline{RDA}$ (UART style) (from display)
CB2
DA (UART style) (to display)
+
$\overline{RES}$
PB7
Reset
PB6
PB5
PB4
PB3
PB2
PB1
PBØ
B6
B5
B4
B3
B2
B1
BØ
ASCII to display
PIA 682Ø