APPLE II MINI MANUAL

CONTENTS

NOTES ON INTERFACING WITH THE HOME TV

Accessories are available to aid the user in connecting the Apple II system to a home color TV with a minimum of trouble. These units are called "RF Modulators" and they generate a radio frequency signal corresponding to the carrier of one or two of the lower VHF television bands; 61.25 MHz (channel 3) or 67.25 MHz (channel 4). This RF signal is then modulated with the composite video signal generated by the Apple II.

Users report success with the following RF modulators:

the "PixieVerter" (a kit)
ATV Research
13th and Broadway
Dakota City, Nebraska 68731

the "TV-1" (a kit)
UHF Associates
6037 Haviland Ave.
Whittier, CA 90601

the "Sup-r-Mod" by (assembled & tested)
M&R Enterprises
P.O. Box 1011
Sunnyvale, CA 94088

the RF Modulator (a P.C. board)
Electronics Systems
P.O. Box 212
Burlingame, CA 94010

Most of the above are available through local computer stores.

The Apple II owner who wishes to use one of these RF Modulators should read the following notes carefully.

All these modulators have a free running transistor oscillator. The M&R Enterprises unit is pre-tuned to Channel 4. The PixieVerter and the TV-1 have tuning by means of a jumper on the P.C. board and a small trimmer capacitor. All these units have a residual FM which may cause trouble if the TV set in use has a IF pass band with excessive ripple. The unit from M&R has the least residual FM.

All the units except the M&R unit are kits to be built and tuned by the customer. All the kits are incomplete to some extent. The unit from Electronics Systems is just a printed circuit board with assembly instructions. The kits from UHF Associates and ATV do not have an RF cable or a shielded box or a balun transformer, or an antenna switch. The M&R unit is complete.

Some cautions are in order. The Apple II, by virtue of its color graphics capability, operates the TV set in a linear mode rather than the 100% contrast mode satisfactory for displaying text. For this reason, radio frequency interference (RFI) generated by a computer (or peripherals) will beat with the carrier of the RF modulator to produce faint spurious background patterns (called "worms") This RFI "trash" must be of quite a low level if worms are to be prevented. In fact, these spurious beats must be 40 to 50db below the signal level to reduce worms to an acceptable level. When it is remembered that only 2 to 6 mV (across 300Ω) is presented to the VHF input of the TV set, then stray RFI getting into the TV must be less than 50µV to obtain a clean picture. Therefore we recommend that a good, co-ax cable be used to carry the signal from any modulator to the TV set, such as RG/59u (with copper shield), Belden #8241 or an equivalent miniature type such as Belden #8218. We also recommend that the RF modulator be enclosed in a tight metal box (an unpainted die cast aluminum box such as Pomona #2428). Even with these precautions, some trouble may be encountered with worms, and can be greatly helped by threading the coax cable connecting the modulator to the TV set repeatedly through a Ferrite toroid core. Apple Computer supplies these cores in a kit, along with a 4 circuit connector/cable assembly to match the auxilliary video connector found on the Apple II board. This kit has order number A2M010X. The M&R "Sup-r-Mod" is supplied with a coax cable and toroids.

Any computer containing fast switching logic and high frequency clocks will radiate some radio frequency energy. Apple II is equipped with a good line filter and many other precautions have been taken to minimize radiated energy. The user is urged not to connect "antennas" to this computer; wires strung about carrying clocks and/data will act as antennas, and subsequent radiated energy may prove to be a nuisance.

Another caution concerns possible long term effects on the TV picture tube. Most home TV sets have "Brightness" and "Contrast" controls with a very wide range of adjustment. When an un-changing picture is displayed with high brightness for a long period ,a faint discoloration of the TV CRT may occur as an inverse pattern observable with the TV set turned off. This condition may be avoided by keeping the "Brightness" turned down slightly and "Contrast" moderate.

THE APPLE II SWITCHING POWER SUPPLY

Switching power supplies generally have both advantages and peculiarities not generally found in conventional power supplies. The Apple II user is urged to review this section.

Your Apple II is equipped with an AC line voltage filter and a three wire AC line cord. It is important to make sure that the third wire is returned to earth ground. Use a continuity checker or ohmmeter to ensure that the third wire is actually returned to earth. Continuity should be checked for between the power supply case and an available water pipe for example. The line filter, which is of a type approved by domestic (U.L. CSA) and international (VDE) agencies must be returned to earth to function properly and to avoid potential shock hazards.

The APPLE II power supply is of the "flyback" switching type. In this system, the AC line is rectified directly, "chopped up" by a high frequency oscillator and coupled through a small transformer to the diodes, filters, etc., and results in four low voltage DC supplies to run APPLE II. The transformer isolates the DC supplies from the line and is provided with several shields to prevent "hash" from being coupled into the logic or peripherals. In the "flyback" system, the energy transferred through from the AC line side to DC supply side is stored in the transformer's inductance on one-half of the operating cycle, then transferred to the output filter capacitors on the second half of the operating cycle. Similar systems are used in TV sets to provide horizontal deflection and the high voltages to run the CRT.

Regulation of the DC voltages is accomplished by controlling the frequency at which the converter operates; the greater the output power needed, the lower the frequency of the converter. If the converter is overloaded, the operating frequency will drop into the audible range with squeels and squawks warning the user that something is wrong.

All DC outputs are regulated at the same time and one of the four outputs (the +5 volt supply) is compared to a reference voltage with the difference error fed to a feedback loop to assist the oscillator in running at the needed frequency. Since all DC outputs are regulated together, their voltages will reflect to some extent unequal loadings.

For example; if the +5 supply is loaded very heavily, then all other supply voltages will increase in voltage slightly; conversely, very light loading on the +5 supply and heavy loading on the +12 supply will cause both it and the others to sag lightly. If precision reference voltages are needed for peripheral applications, they should be provided for in the peripheral design.

In general, the APPLE II design is conservative with respect to component ratings and operating termperatures. An over-voltage crowbar shutdown system and an auxilliary control feedback loop are provided to ensure that even very unlikely failure modes will not cause damage to the APPLE II computer system. The over-voltage protection references to the DC output voltages only. The AC line voltage input must be within the specified limits, i.e., 107V to 132V.

> Under no circumstances, should more than 140 VAC be applied to the input of the power supply. Permanent damage will result.

Since the output voltages are controlled by changing the operating frequency of the converter, and since that frequency has an upper limit determined by the switching speed of power transistors, there then must be a minimum load on the supply; the Apple II board with minimum memory (4K) is well above that minimum load. However, with the board disconnected, there is no load on the supply, and the internal over-voltage protection circuitry causes the supply to turn off. A 9 watt load distributed roughly 50-50 between the +5 and +12 supply is the nominal minimum load.

Nominal load current ratios are: The +12V supply load is ½ that of the +5V.
The - 5V supply load is 1/10 that of the +5V.
The -12V supply load is 1/10 that of the +5V.

The supply voltages are +5.0 ± 0.15 volts, +11.8 ± 0.5 volts, -12.0 ± 1V, -5.2 ± 0.5 volts. The tolerances are greatly reduced when the loads are close to nominal.

The Apple II power supply will power the Apple II board and all present and forthcoming plug-in cards, we recommend the use of low power TTL, CMOS, etc. so that the total power drawn is within the thermal limits of the entire system. In particular, the user should keep the total power drawn by any one card to less than 1.5 watts, and the total current drawn by all the cards together within the following limits:

+ 12V - use no more than 250 mA
+ 5V - use no more than 500 mA
- 5V - use no more than 200 mA
- 12V - use no more than 200 mA

The power supply is allowed to run indefinetly under short circuit or open circuit conditions.

> CAUTION: There are dangerous high voltages inside the power supply case. Much of the internal circuitry is NOT isolated from the power line, and special equipment is needed for service. NO REPAIR BY THE USER IS ALLOWED.

GETTING STARTED WITH YOUR APPLE II BOARD

INTRODUCTION

ITEMS YOU WILL NEED:

Your APPLE II board comes completely assembled and thoroughly tested. You should have received the following:

a. 1 ea. APPLE II P.C. Board complete with specified RAM memory.

b. 1 ea. d.c. power connector with cable.

c. 1 ea. 2" speaker with cable.

d. 1 ea. Preliminary Manual

e. 2 ea. Demonstration cassette tapes.

f. 2 ea. 16 pin headers plugged into locations A7 and J14.

In addition you will need:

g. A color TV set (or B & W) equipped with a direct video input connector for best performance or a commercially available RF modulator such as a "Pixi-verter"tm. Higher channel (7-13) modulators generally provide better system performance than lower channel modulators (2-6).

h. The following power supplies (NOTE: current ratings do not include any capacity for peripheral boards.):

1. +12 Volts with the following current capacity:

   a. For 4K or 16K systems - 350mA.

   b. For 8K, 20K or 32K - 550mA.

   c. For 12K, 24K, 36K or 48K - 850mA.

2. +5 Volts at 1.6 amps

3. -5 Volts at 10mA.

4. OPTIONAL: If -12 Volts is required by your keyboard. (If using an APPLE II supplied keyboard, you will need -12V at 50mA.)

i. An audio cassette recorder such as a Panasonic model RQ-309 DS which is used to load and save programs.

j. An ASCII encoded keyboard equipped with a "reset" switch.

k. Cable for the following:

1. Keyboard to APPLE II P.C.B.

2. Video out 75 ohm cable to TV or modulator

3. Cassette to APPLE II P.C.B. (1 or 2)

Optionally you may desire:

1. Game paddles or pots with cables to APPLE II Game I/O connector. (Several demo programs use PDL(0) and "Pong" also uses PDL(1).

m. Case to hold all the above

Final Assembly Steps

1. Using detailed information on pin functions in hardware section of manual, connect power supplies to d.c. cable assembly. Use both ground wires to miminize resistance. With cable assembly disconnected from APPLE II mother board, turn on power supplies and verify voltages on connector pins. Improper supply connections such as reverse polarity can severely damage your APPLE II.

2. Connect keyboard to APPLE II by unplugging leader in location A7 and wiring keyboard cable to it, then plug back into APPLE II P.C.B.

3. Plug in speaker cable.

4. Optionally connect one or two game paddles using leader supplied in socket located at J14.

5. Connect video cable.

6. Connect cable from cassette monitor output to APPLE II cassette input.

7. Check to see that APPLE II board is not contacting any conducting surface.

8. With power supplies turned off, plug in power connector to mother board then recheck all cableing.

POWER UP

1. Turn power on. If power supplies overload, immediately turn off and recheck power cable wiring. Verify operating supply voltages are within ±3% of nominal value.

2. You should now have random video display. If not check video level pot on mother board, full clockwise is maximum video output. Also check video cables for opens and shorts. Check modulator if you are using one.

3. Press reset button. Speaker should beep and a "*" prompt character with a blinking cursor should appear in lower left on screen.

4. Press "esc" button, release and type a "@" (shift-P) to clear screen. You may now try "Monitor" commands if you wish. See details in "Monitor" software section.

RUNNING BASIC

1. Turn power on; press reset button; type "control B" and press return button. A ">" prompt character should appear on screen indicating that you are now in BASIC.

2. Load one of the supplied demonstration cassettes into recorder. Set recorder level to approximately 5 and start recorder. Type "LOAD" and return. First beep indicates that APPLE II has found beginning of program; second indicates end of program followed by ">" character on screen. If error occurs on loading, try a different demo tape or try changing cassette volume level.

3. Type RUN and carriage return to execute demonstration program. Listings of these are included in the last section of this manual.

APPLE II MANUAL

SOFTWARE SECTION

CONTENTS

System Monitor Commands

Apple II contains a powerful machine level monitor for use by the advanced programmer. To enter the monitor either press RESET button on keyboard or CALL-151 (Hex FF65) from Basic. Apple II will respond with an "*" (asterisk) prompt character on the TV display. This action will not kill current BASIC program which may be re-entered by a C^C (control C). NOTE: "adrs" is a four digit hexidecimal number and "data" is a two digit hexidecimal number. Remember to press "return" button at the end of each line.

Command Format	Example	Description
Examine Memory		
adrs	*CØF2	Examines (displays) single memory location of (adrs)
adrs1.adrs2	*1Ø24.1Ø48	Examines (displays) range of memory from (adrs1) thru (adrs2)
(return)	* (return)	Examines (displays) next 8 memory locations.
.adrs2	*.4Ø96	Examines (displays) memory from current location through location (adrs2)
Change Memory		
adrs:data data data	*A256:EF 20 43	Deposits data into memory starting at location (adrs).
:data data data	*:FØ A2 12	Deposits data into memory starting after (adrs) last used for deposits.
Move Memory		
adrs1<adrs2. adrs3M	*1ØØ<BØ1Ø.B41ØM	Copy the data now in the memory range from (adrs2) to (adrs3) into memory locations starting at (adrs1).
Verify Memory		
adrs1<adrs2. adrs3V	*1ØØ<BØ1Ø.B41ØV	Verify that block of data in memory range from (adrs2) to (adrs3) exactly matches data block starting at memory location (adrs1) and displays differences if any.

Command Format	Example	Description
Cassette I/O		
adrs1.adrs2R	*3ØØ.4FFR	Reads cassette data into specified memory (adrs) range. Record length must be same as memory range or an error will occur.
adrs1.adrs2W	*8ØØ.9FFW	Writes onto cassette data from specified memory (adrs) range.
Display		
I	*I	Set inverse video mode. (Black characters on white background)
N	*N	Set normal video mode. (White characters on black background)
Dis-assembler		
adrsL	*C8ØØL	Decodes 2Ø instructions starting at memory (adrs) into 65Ø2 assembly nmenonic code.
L	*L	Decodes next 2Ø instructions starting at current memory address.
Mini-assembler		
(Turn-on)	*F666G	Turns-on mini-assembler. Prompt character is now a "!" (exclamation point).
$(monitor command)	!$C8ØØL	Executes any monitor command from mini-assembler then returns control to mini-assembler. Note that many monitor commands change current memory address reference so that it is good practice to retype desired address reference upon return to mini-assembler.
adrs:(65Ø2 MNEMONIC instruction)	!CØ1Ø:STA 23FF	Assembles a mnemonic 65Ø2 instruction into machine codes. If error, machine will refuse instruction, sound bell, and reprint line with up arrow under error.

Command Format	Example	Description
(space) (65Ø2 mnemonic instruction)	! STA 01FF	Assembles instruction into next available memory location. (Note space between "!" and instruction)
(TURN-OFF)	! (Reset Button)	Exits mini-assembler and returns to system monitor.

Monitor Program Execution and Debugging

adrsG	*3ØØG	Runs machine level program starting at memory (adrs).
adrsT	*8ØØT	Traces a program starting at memory location (adrs) and continues trace until hitting a breakpoint. Break occurs on instruction ØØ (BRK), and returns control to system monitor. Opens 65Ø2 status registers (see note 1).
adrsS	*CØ5ØS	Single steps through program beginning at memory location (adrs). Type a letter S for each additional step that you want displayed. Opens 6502 status registers (see Note 1).
(Control E)	*E^{C}	Displays 65Ø2 status registers and opens them for modification (see Note 1).
(Control Y)	*Y^{C}	Executes user specified machine language subroutine starting at memory location (3F8).

Note 1:

6502 status registers are open if they are last line displayed on screen. To change them type ":" then "data" for each register.

```
Example:  A = 3C  X = FF  Y = ØØ  P = 32  S = F2
          *: FF               Changes A register only
          *:FF ØØ 33          Changes A, X, and Y registers
```

To change S register, you must first retype data for A, X, Y and P.

Hexidecimal Arithmetic

data1+data2	*78+34	Performs hexidecimal sum of data1 plus data2.
data1-data2	*AE-34	Performs hexidecimal difference of data1 minus data2.

Command Format	Example	Description
Set Input/Output Ports		
(X) (Control P)	*5P^C	Sets printer output to I/O slot number (X). (see Note 2 below)
(X) (Control K)	*2K^C	Sets keyboard input to I/O slot number (X). (see Note 2 below)

Note 2:

Only slots 1 through 7 are addressable in this mode. Address Ø (Ex: ØP^C or ØK^C) resets ports to internal video display and keyboard. These commands will not work unless Apple II interfaces are plugged into specificed I/O slot.

Multiple Commands

	*1ØØL 4ØØG AFFT	Multiple monitor commands may be given on same line if separated by a "space".
	*LLLL	Single letter commands may be repeated without spaces.

BASIC COMMANDS

Commands are executed immediately; they do not require line numbers.Most Statements (see Basic Statements Section) may also be used as commands. Remember to press Return key after each command so that Apple knows that you have finished that line. Multiple commands (as opposed to statements) on same line separated by a " : " are NOT allowed.

COMMAND NAME

AUTO *num*

Sets automatic line numbering mode. Starts at line number *num* and increments line numbers by 10. To exit AUTO mode, type a control X*, then type the letters "MAN" and press the return key.

AUTO *num1, num2*

Same as above execept increments line numbers by number *num2*..

CLR

Clears current BASIC variables; undimensions arrays. Program is unchanged.

CON

Continues program execution after a stop from a control C*. Does not change variables.

DEL *num1*

Deletes line number *num1*.

DEL *num1, num2*

Deletes program from line number *num1* through line number *num2*.

DSP *var*

Sets debug mode that will display variable *var* every-time that it is changed along with the line number that caused the change. (NOTE: RUN command clears DSP mode so that DSP command is effective only if program is continued by a CON or GOTO command.)

HIMEM: *expr*

Sets highest memory location for use by BASIC at location specified by expression *expr* in decimal. HIMEM: may not be increased without destroying program. HIMEM: is automatically set at maximum RAM memory when BASIC is entered by a control B*.

GOTO *expr*

Causes immediate jump to line number specified by expression *expr*.

GR

Sets mixed color graphics display mode. Clears screen to black. Resets scrolling window. Displays 40x40 squares in 15 colors on top of screen and 4 lines of text at bottom.

LIST

Lists entire program on screen.

LIST *num1*

Lists program line number *num1*.

LIST *num1, num2*

Lists program line number *num1* through line number *num2*.

LOAD *expr.*
Reads (Loads) a BASIC program from cassette tape. Start tape recorder before hitting return key. Two beeps and a " > " indicate a good load. "ERR" or "MEM" FULL ERR" message indicates a bad tape or poor recorder performance.

LOMEM: *expr*
Similar to HIMEM: except sets lowest memory location available to BASIC. Automatically set at 2048 when BASIC is entered with a control B*. Moving LOMEM: destroys current variable values.

MAN
Clears AUTO line numbering mode to all manual line numbering after a control C* or control X*.

NEW
Clears (Scratches) current BASIC program.

NO DSP *var*
Clears DSP mode for variable *var.*

NO TRACE
Clears TRACE mode.

RUN
Clears variables to zero, undimensions all arrays and executes program starting at lowest statement line number.

RUN *expr*
Clears variables and executes program starting at line number specified by expression *expr.*

SAVE
Stores (saves) a BASIC program on a cassette tape. Start tape recorder in record mode prior to hitting return key.

TEXT
Sets all text mode. Screen is formated to display alpha-numeric characters on 24 lines of 40 characters each. TEXT resets scrolling window to maximum.

TRACE
Sets debug mode that displays line number of each statement as it is executed.

* Control characters such as control X or control C are typed by holding down the CTRL key while typing the specified letter. This is similiar to how one holds down the shift key to type capital letters. Control characters are NOT displayed on the screen but are accepted by the computer. For example, type several control G's. We will also use a superscript C to indicate a control character as in X^C.

BASIC Operators

Symbol	Sample Statement	Explanation
Prefix Operators		
( )	1Ø X= 4*(5 + X)	Expressions within parenthesis ( ) are always evaluated first.
+	2Ø X= +4*5	Optional; +1 times following expression.
-	3Ø ALPHA = -(BETA +2)	Negation of following expression.
NOT	4Ø IF NOT B THEN 2ØØ 5Ø 1=NOT NOT 1	Logical Negation of following expression; Ø if expression is true (non-zero), 1 if expression is false (zero).
Arithmetic Operators		
↑	6Ø Y = X↑3	Exponentiate as in X^3. NOTE: ↑ is shifted letter N.
*	7Ø LET DOTS=A*B*N2	Multiplication. NOTE: Implied multiplication such as (2 + 3)(4) is not allowed thus N2 in example is a variable not N * 2.
/	8Ø PRINT GAMMA/S	Divide
MOD	9Ø 5 = 12 MOD 7 1ØØ X = X MOD(Y+2)	Modulo: Remainder after division of first expression by second expression.
+	11Ø P = L + G	Add
-	12Ø XY4 = H-D	Substract
=	13Ø HEIGHT=15 14Ø LET SIZE=7*5 15Ø A(8) = 2 155 ALPHA$ = "PLEASE"	Assignment operator; assigns a value to a variable. LET is optional

Relational and Logical Operators

The numeric values used in logical evaluation are "true" if non-zero, "false" if zero.

Symbol	Sample Statement	Explanation
=	16Ø IF D = E THEN 5ØØ	Expression "equals" expression.
=	17Ø IF A$(1,1)= "Y" THEN 5ØØ	String variable "equals" string variable.
# or < >	18Ø IF ALPHA #X*Y THEN 500	Expression "does not equal" expression.
#	19Ø IF A$ # "NO" THEN 5ØØ	String variable "does not equal" string variable. NOTE: If strings are not the same length, they are considered un-equal. < > not allowed with strings.
>	2ØØ IF A>B THEN GO TO 5Ø	Expression "is greater than" expression.
<	21Ø IF A+1<B-5 THEN 1ØØ	Expression "is less than" expression.
>=	22Ø IF A>=B THEN 1ØØ	Expression "is greater than or equal to" expression.
<=	23Ø IF A+1<=B-6 THEN 2ØØ	Expression "is less than or equal to" expression.
AND	24Ø IF A>B AND C<D THEN 2ØØ	Expression 1 "and" expression 2 must both be "true" for statements to be true.
OR	25Ø IF ALPHA OR BETA+1 THEN 2ØØ	If either expression 1 or expression 2 is "true", statement is "true".

BASIC FUNCTIONS

Functions return a numeric result. They may be used as expressions or as part of expressions. PRINT is used for examples only, other statements may be used. Expressions following function name must be enclosed between two parenthesis signs.

FUNCTION NAME

ABS *(expr)*	3ØØ PRINT ABS(X)	Gives absolute value of the expression *expr*.
ASC *(str$)*	31Ø PRINT ASC("BACK") 32Ø PRINT ASC(B$) 33Ø PRINT ASC(B$(4,4)) 335 PRINT ASC(B$(Y))	Gives decimal ASCII value of designated string variable *str$* . If more than one character is in designated string or sub-string, it gives decimal ASCII value of first character.
LEN *(str$)*	34Ø PRINT LEN(B$)	Gives current length of designated string variable *str$*;i.e., number of characters.
PDL *(expr)*	35Ø PRINT PDL(X)	Gives number between Ø and 255 corresponding to paddle position on game paddle number designated by expression *expr* and must be legal paddle (Ø,1,2,or 3) or else 255 is returned.
PEEK *(expr)*	36Ø PRINT PEEK(X)	Gives the decimal value of number stored of decimal memory location specified by expression *expr*. For MEMORY locations above 32676, use negative number; i.e., HEX location FFFØ is -32751
RND *(expr)*	37Ø PRINT RND(X)	Gives random number between 0 and (expression *expr* -1) if expression *expr* is positive; if minus, it gives random number between Ø and (expression *expr* +1).
SCRN(*expr1, expr2)*	380 PRINT SCRN (X1,Y1)	Gives color (number between Ø and 15) of screen at horizontal location designated by expression *expr1* and vertical location designated by expression *expr2* Range of expression *expr1* is Ø to 39. Range of expression *expr2* is Ø to 39 if in standard mixed colorgraphics display mode as set by GR command or Ø to 47 if in all color mode set by POKE -163Ø4 ,Ø: POKE - 163Ø2,Ø.
SGN *(expr)*	39Ø PRINT SGN(X)	Gives sign (not sine) of expression *expr* i.e., -1 if expression *expr* is negative, zero if zero and +1 if *expr* is positive.

BASIC STATEMENTS

Each BASIC statement must have a line number between Ø and 32767. Variable names must start with an alpha character and may be any number of alphanumeric characters up to 1ØØ. Variable names may not contain buried any of the following words: AND, AT, MOD, OR, STEP, or THEN. Variable names may not begin with the letters END, LET, or REM. String variables names must end with a $ (dollar sign). Multiple statements may appear under the same line number if separated by a : (colon) as long as the total number of characters in the line (including spaces) is less than approximately 15Ø characters
Most statements may also be used as commands. BASIC statements are executed by RUN or GOTO commands.

NAME		
CALL *expr*	1Ø CALL-936	Causes execution of a machine level language subroutine at decimal memory location specified by expression *expr* Locations above 32767 are specified using negative numbers; i.e., location in example 1Ø is hexidecimal number $FC53
COLOR=*expr*	3Ø COLOR=12	In standard resolution color (GR) graphics mode, this command sets screen TV color to value in expression *expr* in the range Ø to 15 as described in Table A. Actually expression *expr* may be in the range Ø to 255 without error message since it is implemented as if it were expression *expr* MOD 16.
DIM *var1 (expr1)* *str$ (expr2)* *var2 (expr3)*	5Ø DIM A(2Ø),B(1Ø) 6Ø DIM B$(3Ø) 7Ø DIM C Illegal: 8Ø DIM A(3Ø) Legal: 85 DIM C(1ØØØ)	The DIM statement causes APPLE II to reserve memory for the specified variables. For number arrays APPLE reserves approximately 2 times *expr* bytes of memory limited by available memory. For string arrays -*str$*- *(expr)* must be in the range of 1 to 255. Last defined variable may be redimensioned at any time; thus, example in line 80 is illegal but 85 is allowed.
DSP *var*	Legal: 9Ø DPS AX:DSP L Illegal: 1ØØ DSP AX,B 102 DSP AB$ 104 DSP A(5) Legal: 105 A=A(5): DSP A	Sets debug mode that DSP variable *var* each time it changes and the line number where the change occured.

NAME	EXAMPLE	DESCRIPTION
END	11Ø END	Stops program execution. Sends carriage return and "> " BASIC prompt) to screen.
FOR *var*= *expr1* TO*expr2* STEP*expr3*	11Ø FOR L=Ø to 39 12Ø FOR X=Y1 TO Y3 13Ø FOR I=39 TO 1 15Ø GOSUB 1ØØ *J2	Begins FOR...NEXT loop, initializes variable *var* to value of expression *expr1* then increments it by amount in expression *expr 3* each time the corresponding "NEXT" statement is encountered, until value of expression *expr 2* is reached. If STEP *expr3* is omitted, a STEP of +1 is assumed. Negative numbers are allowed.
GOSUB *expr*	14Ø GOSUB 5ØØ	Causes branch to BASIC subroutine starting at legal line number specified by expression *expr* Subroutines may be nested up to 16 levels.
GOTO *expr*	16Ø GOTO 2ØØ 17Ø GOTO ALPHA+1ØØ	Causes immediate jump to legal line number specified by expression *expr*.
GR	18Ø GR 19Ø GR: POKE -163Ø2,Ø	Sets mixed standard resolution color graphics mode. Initializes COLOR = Ø (Black) for top 4Øx4Ø of screen and sets scrolling window to lines 21 through 24 by 4Ø characters for four lines of text at bottom of screen. Example 19Ø sets all color mode (4Øx48 field) with no text at bottom of screen.
HLIN *expr1*, *expr2*AT*expr3*	2ØØ HLIN Ø,39 AT 2Ø 21Ø HLIN Z,Z+6 AT I	In standard resolution color graphics mode, this command draws a horizontal line of a predefined color (set by COLOR=) starting at horizontal position defined by expression *expr1* and ending at position *expr2* at vertical position defined by expression *expr3* . *expr1* and *expr2* must be in the range of Ø to 39 and *expr1* < = *expr2* . *expr3* be in the range of Ø to 39 (or Ø to 47 if not in mixed mode).

Note: HLIN Ø, 19 AT Ø is a horizontal line at the top of the screen extending from left corner to center of screen and HLIN 2Ø,39 AT 39 is a horizontal line at the bottom of the screen extending from center to right corner.

IF *expression* THEN *statement*	22Ø IF A > B THEN PRINT A 23Ø IF X=Ø THEN C=1 24Ø IF A#1Ø THEN GOSUB 2ØØ 25Ø IF A$(1,1)# "Y" THEN 1ØØ Illegal: 26Ø IF L > 5 THEN 5Ø: ELSE 6Ø Legal: 27Ø IF L > 5 THEN 5Ø GO TO 6Ø	If *expression* is true (non-zero) then execute *statement*; if false do not execute *statement*. If *statement* is an expression, then a GOTO *expr* type of statement is assumed to be implied. The "ELSE" in example 26Ø is illegal but may be implemented as shown in example 27Ø.
INPUT *var1, var2, str$*	28Ø INPUT X,Y,Z(3) 29Ø INPUT "AMT", DLLR 3ØØ INPUT "Y or N?", A$	Enters data into memory from I/O device. If number input is expected, APPLE wil output "?"; if string inout is expected no "?" will be outputed. Multiple numeric inputs to same statement may be separated by a comma or a carriage return. String inputs must be separated by a carriage return only. One pair of " " may be used immediately after INPUT to output prompting text enclosed within the quotation marks to the screen.
IN# *expr*	31Ø IN# 6 32Ø IN# Y+2 33Ø IN# 0	Transfers source of data for subsequent INPUT statements to peripheral I/O slot (1-7) as specified as by expression *expr*. Slot Ø is not addressable from BASIC. IN#Ø (Example 33Ø) is used to return data source from peripherial I/O to keyboard connector.
LET	34Ø LET X=5	Assignment operator. "LET" is optional
LIST *num1, num2*	35Ø IF X > 6 THEN LIST 5Ø	Causes program from line number *num1* through line number *num2* to be displayed on screen.
NEXT *var1, var2*	36Ø NEXT I 37Ø NEXT J,K	Increments corresponding "FOR" variable and loops back to statement following "FOR" until variable exceeds limit.
NO DSP *var*	38Ø NO DSP I	Turns-off DSP debug mode for variable
NO TRACE	39Ø NO TRACE	Turns-off TRACE debug mode

PLOT *expr1, expr2*	4ØØ PLOT 15, 25 4ØØ PLT XV,YV	In standard resolution color graphics, this command plots a small square of a predefined color (set by COLOR=) at horizontal location specified by expression *expr1* in range Ø to 39 and vertical location specified by expression *expr2* in range Ø to 39 (or Ø to 47 if in all graphics mode) NOTE: PLOT Ø Ø is upper left and PLOT 39, 39 (or PLOT 39, 47) is lower right corner.
POKE *expr1, expr2*	42Ø POKE 2Ø, 4Ø 430 POKE 7*256, XMOD255	Stores decimal number defined by expression *expr2* in range of Ø 255 at decimal memory location specified by expression *expr1* Locations above 32767 are specified by negative numbers.
POP	44Ø POP	"POPS" nested GOSUB return stack address by one.
PRINT *var1, var, str$*	45Ø PRINT L1 46Ø PRINT L1, X2 47Ø PRINT "AMT=";DX 48Ø PRINT A$;B$; 49Ø PRINT 492 PRINT "HELLO" 494 PRINT 2+3	Outputs data specified by variable *var* or string variable *str$* starting at current cursor location. If there is not trailing "," or ";" (Ex 45Ø) a carriage return will be generated. Commas (Ex. 46Ø) outputs data in 5 left justified columns. Semi-colon (Ex. 47Ø) inhibits print of any spaces. Text imbedded in " " will be printed and may appear multiple times.
PR# *expr*	500 PR# 7	Like IN#, transfers output to I/O slot defined by expression *expr* PR# Ø is video output not I/O slot Ø.
REM	510 REM REMARK	No action. All characters after REM are treated as a remark until terminated by a carriage return.
RETURN	52Ø RETURN 53Ø IFX= 5 THEN RETURN	Causes branch to statement following last GOSUB; i.e., RETURN ends a subroutine. Do not confuse "RETURN" statement with Return key on keyboard.

RNDC

TAB *expr*	53Ø TAB 24 54Ø TAB I+24 55Ø IF A#B THEN TAB 2Ø	Moves cursor to absolute horizontal position specified by expression *expr* in the range of 1 to 4Ø. Position is left to right
TEXT	55Ø TEXT 56Ø TEXT: CALL-936	Sets all text mode. Resets scrolling window to 24 lines by 4Ø characters. Example 56Ø also clears screen and homes cursor to upper left corner
TRACE	57Ø TRACE 580 IFN > 32ØØØ THEN TRACE	Sets debug mode that displays each line number as it is executed.
VLIN *expr1, expr2* AT *expr3*	59Ø VLIN Ø, 39AT15 6ØØ VLIN Z,Z+6ATY	Similar to HLIN except draws vertical line starting at *expr1* and ending at *expr2* at horizontal position *expr3*.
VTAB *expr*	61Ø VTAB 18 62Ø VTAB Z+2	Similar to TAB. Moves cursor to absolute vertical position specified by expression *expr* in the range 1 to 24. VTAB 1 is top line on screen; VTAB24 is bottom.

SPECIAL CONTROL AND EDITING CHARACTERS

"Control" characters are indicated by a super-scripted "C" such as G^C. They are obtained by holding down the CTRL key while typing the specified letter. Control characters are NOT displaved on the TV screen. B^C and C^C must be followed by a carriage return. Screen editing characters are indicated by a sub-scripted "E" such as D_E. They are obtained by pressing and releasing the ESC key then typing specified letter. Edit characters send information only to display screen and does not send data to memory. For example, U^C moves to cursor to right and copies text while A_E moves cursor to right but does not copy text.

CHARACTER — DESCRIPTION OF ACTION

RESET key — Immediately interrupts any program execution and resets computer. Also sets all text mode with scrolling window at maximum. Control is transfered to System Monitor and Apple prompts with a "*" (asterisk) and a bell. Hitting RESET key does NOT destroy existing BASIC or machine language program.

Control B — If in System Monitor (as indicated by a "*"), a control B and a carriage return will transfer control to BASIC, scratching (killing) any existing BASIC program and set HIMEM: to maximum installed user memory and LOMEM: to 2048.

Control C — If in BASIC, halts program and displays line number where stop occurred*. Program may be continued with a CON command. If in System Monitor, (as indicated by "*"), control C and a carraige return will enter BASIC without killing current program.

Control G — Sounds bell (beeps speaker)

Control H — Backspaces cursor and deletes any overwritten characters from computer but not from screen. Apply supplied keyboards have special key "←" on right side of keyboard that provides this functions without using control button.

Control J — Issues line feed only

Control V — Compliment to H^C. Forward spaces cursor and copies over written characters. Apple keyboards have "→" key on right side which also performs this function.

Control X — Immediately deletes current line.

* If BASIC program is expecting keyboard input, you will have to hit carriage return key after typing control C.

CHARACTER	DESCRIPTION OF ACTION
A_E	Move cursor to right
B_E	Move cursor to left
C_E	Move cursor down
D_E	Move cursor up
E_E	Clear text from cursor to end of line
F_E	Clear text from cursor to end of page
$@_E$	Home cursor to top of page, clear text to end of page.

Table A: APPLE II COLORS AS SET BY COLOR =

Note: Colors may vary depending on TV tint (hue) setting and may also be changed by adjusting trimmer capacitor C3 on APPLE II P.C. Board.

Ø = Black
1 - Magenta
2 = Dark Blue
3 = Light Purple
4 = Dark Green
5 = Grey
6 = Medium Blue
7 = Light Blue

8 = Brown
9 = Orange
1Ø = Grey
11 = Pink
12 = Green
13 = Yellow
14 = Blue/Green
15 = White

Special Controls and Features

Hex	BASIC Example	Description
Display Mode Controls		
CØ50	1Ø POKE -163Ø4,Ø	Set color graphics mode
CØ51	2Ø POKE -163Ø3,Ø	Set text mode
CØ52	3Ø POKE -163Ø2,Ø	Clear mixed graphics
CØ53	4Ø POKE -163Ø1,Ø	Set mixed graphics (4 lines text)
CØ54	5Ø POKE -163ØØ,Ø	Clear display Page 2 (BASIC commands use Page 1 only)
CØ55	6Ø POKE -16299,Ø	Set display to Page 2 (alternate)
CØ56	7Ø POKE -16298,Ø	Clear HIRES graphics mode
CØ57	8Ø POKE -16297,Ø	Set HIRES graphics mode
TEXT Mode Controls		
ØØ2Ø	9Ø POKE 32,L1	Set left side of scrolling window to location specified by L1 in range of Ø to 39.
ØØ21	1ØØ POKE 33,W1	Set window width to amount specified by W1. L1+W1<4Ø. W1>Ø
ØØ22	11Ø POKE 34,T1	Set window top to line specified by T1 in range of Ø to 23
ØØ23	12Ø POKE 35,B1	Set window bottom to line specified by B1 in the range of Ø to 23. B1>T1
ØØ24	13Ø CH=PEEK(36) 14Ø POKE 36,CH 15Ø TAB(CH+1)	Read/set cusor horizontal position in the range of Ø to 39. If using TAB, you must add "1" to cusor position read value; Ex. 140 and 150 perform identical function.
ØØ25	16Ø CV=PEEK(37) 17Ø POKE 37,CV 18Ø VTAB(CV+1)	Similar to above. Read/set cusor vertical position in the range Ø to 23.
ØØ32	19Ø POKE 5Ø,127 2ØØ POKE 5Ø,255	Set inverse flag if 127 (Ex. 19Ø) Set normal flag if 255(Ex. 2ØØ)
FC58	21Ø CALL -936	($@_E$) Home cusor, clear screen
FC42	22Ø CALL -958	(F_E) Clear from cusor to end of page

Hex	BASIC Example	Description
FC9C	23Ø CALL -868	(E_E) Clear from cusor to end of line
FC66	24Ø CALL -922	(J^C) Line feed
FC7Ø	25Ø CALL -912	Scroll up text one line

Miscellaneous

CØ3Ø	36Ø X=PEEK(-16336) 365 POKE -16336,Ø	Toggle speaker
CØØØ	37Ø X=PEEK(-16384	Read keyboard; if X>127 then key was pressed.
CØ1Ø	38Ø POKE -16368,Ø	Clear keyboard strobe - always after reading keyboard.
CØ61	39Ø X=PEEK(16287)	Read PDL(Ø) push button switch. If X>127 then switch is "on".
CØ62	4ØØ X=PEEK(-16286)	Read PDL(1) push button switch.
CØ63	41Ø X=PEEK(-16285	Read PDL(2) push button switch.
CØ58	42Ø POKE -16296,Ø	Clear Game I/O ANØ output
CØ59	43Ø POKE -16295,Ø	Set Game I/O ANØ output
CØ5A	44Ø POKE -16294,Ø	Clear Game I/O AN1 output
CØ5B	45Ø POKE -16293,Ø	Set Game I/O AN1 output
CØ5C	46Ø POKE -16292,Ø	Clear Game I/O AN2 output
CØ5D	47Ø POKE -16291,Ø	Set Game I/O AN2 output
CØ5E	48Ø POKE -16290,Ø	Clear Game I/O AN3 output
CØ5F	49Ø POKE -16289,Ø	Set Game I/O AN3 output

APPLE II BASIC ERROR MESSAGES

*** SYNTAX ERR
Results from a syntactic or typing error.

*** > 32767 ERR
A value entered or calculated was less than -32767 or greater than 32767.

*** > 255 ERR
A value restricted to the range 0 to 255 was outside that range.

*** BAD BRANCH ERR
Results from an attempt to branch to a non-existant line number.

*** BAD RETURN ERR
Results from an attempt to execute more RETURNs than previously executed GOSUBs.

*** BAD NEXT ERR
Results from an attempt to execute a NEXT statement for which there was not a corresponding FOR statement.

*** 16 GOSUBS ERR
Results from more than 16 nested GOSUBs.

*** 16 FORS ERR
Results from more than 16 nested FOR loops.

*** NO END ERR
The last statement executed was not an END.

*** MEM FULL ERR
The memory needed for the program has exceeded the memory size allotted.

*** TOO LONG ERR
Results from more than 12 nested parentheses or more than 128 characters in input line.

*** DIM ERR
Results from an attempt to DIMension a string array which has been previously dimensioned.

*** RANGE ERR
An array was larger than the DIMensioned value or smaller than 1 or HLIN,VLIN, PLOT, TAB, or VTAB arguments are out of range.

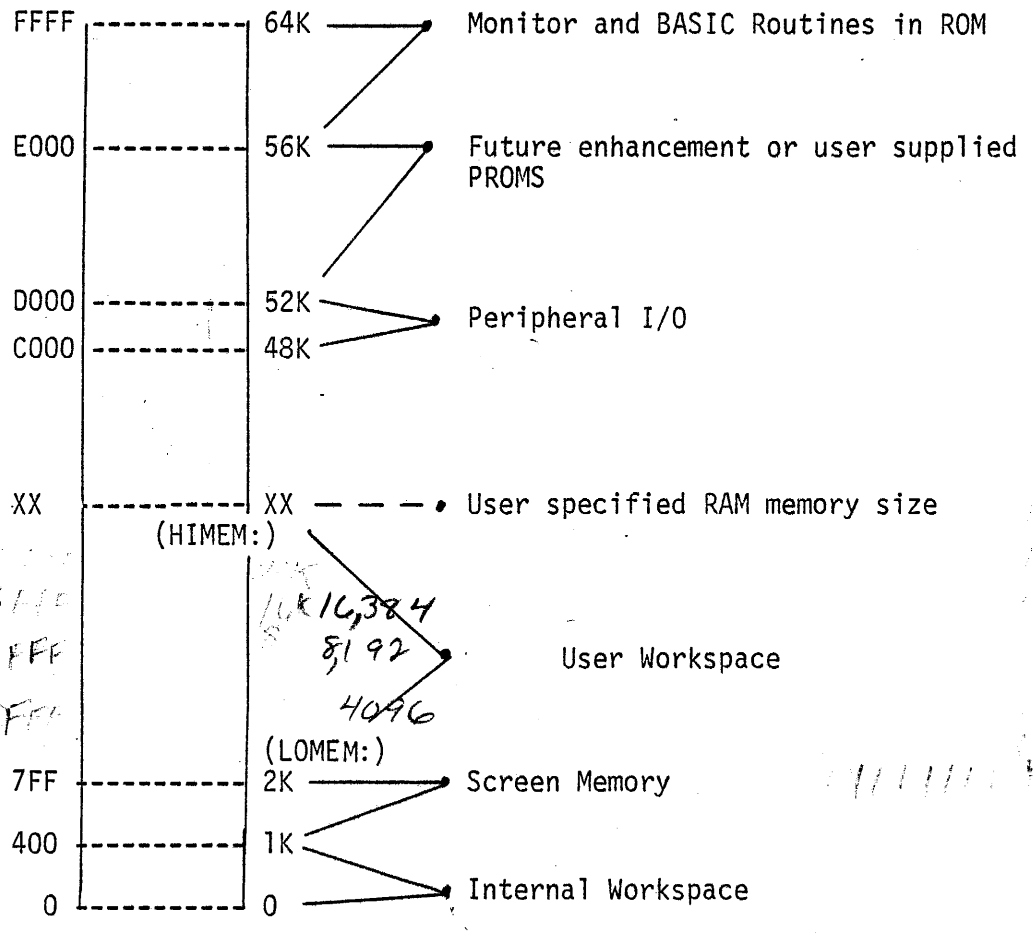
*** STR OVFL ERR
The number of characters assigned to a string exceeded the DIMensioned value for that string.

*** STRING ERR
Results from an attempt to execute an illegal string operation.

RETYPE LINE
Results from illegal data being typed in response to an INPUT statement. This message also requests that the illegal item be retyped.

9000 / 6000 > = 4000 (6000)

Simplified Memory Map



A
B
C
D
E
F

APPLE II MANUAL

HARDWARE SECTION

CONTENTS

*Future Addition

INTERFACING THE APPLE

This section defines the connections by which external devices are attached to the APPLE II board. Included are pin diagrams, signal descriptions, loading constraints and other useful information.

TABLE OF CONTENTS

Figure 1A APPLE II Board-Complete View

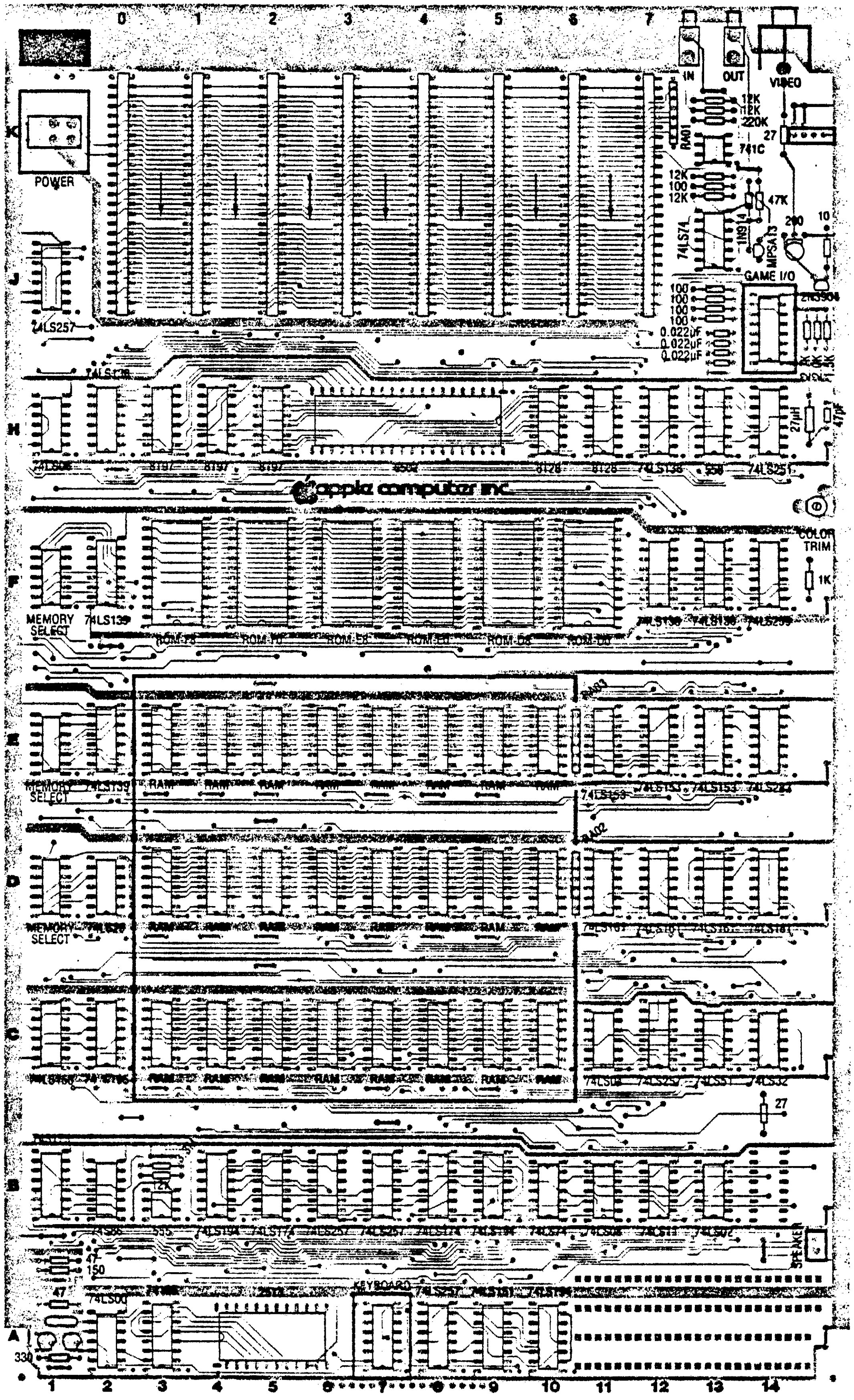
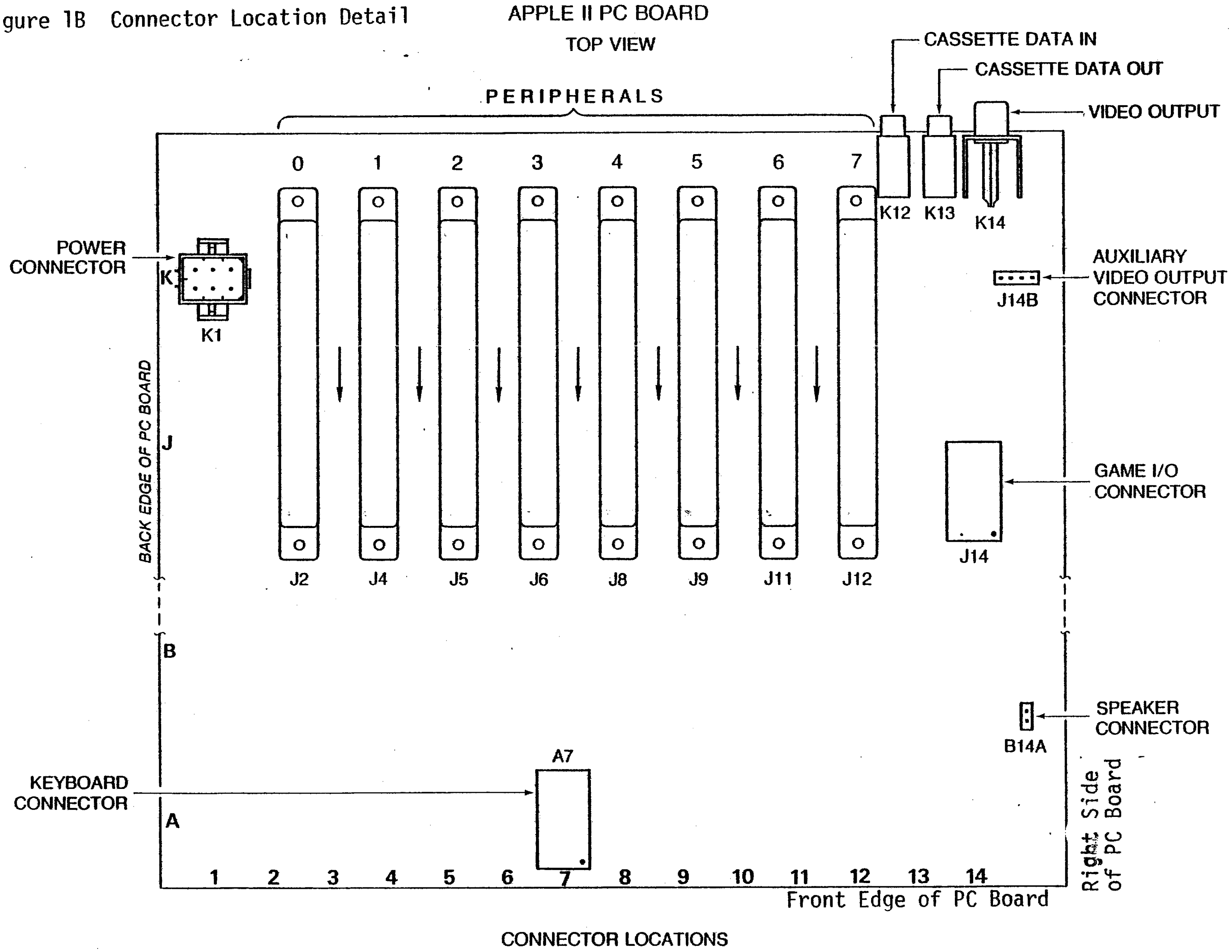


Figure 1B Connector Location Detail



CASSETTE JACKS

A convenient means for interfacing an inexpensive audio cassette tape recorder to the APPLE II is provided by these two standard (3.5mm) miniature phone jacks located at the back of the APPLE II board.

CASSETTE DATA IN JACK: Designed for connection to the "EARPHONE" or "MONITOR" output found on most audio cassette tape recorders. V_{IN}=1Vpp (nominal), Z_{IN}=12K Ohms. Located at K12 as illustrated in Figure 1.

CASSETTE DATA OUT JACK: Designed for connection to the "MIC" or "MICROPHONE" input found on most audio cassette tape recorders. V_{OUT}=25 mV into 100 Ohms, Z_{OUT}=100 Ohms. Located at K13 as illustrated in Figure 1.

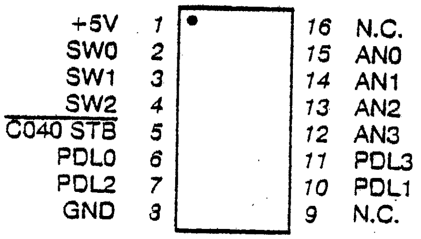
GAME I/O CONNECTOR

The Game I/O Connector provides a means for connecting paddle controls, lights and switches to the APPLE II for use in controlling video games, etc. It is a 16 pin IC socket located at J14 and is illustrated in Figure 1 and 2.

Figure 2 GAME I/O CONNECTOR

TOP VIEW

(Front Edge of PC Board



LOCATION J14

SIGNAL DESCRIPTIONS FOR GAME I/O

ANØ-AN3: 8 addresses (CØ58-CØ5F) are assigned to selectively "SET" or "CLEAR" these four "ANNUNCIATOR" outputs. Envisioned to control indicator lights, each is a 74LSxx series TTL output and must be buffered if used to drive lamps.

CØ4Ø STB: A utility strobe output. Will go low during $\emptyset_2$ of a read or write cycle to addresses CØ4Ø-CØ4F. This is a 74LSxx series TTL output.

GND: System circuit ground. 0 Volt line from power supply.

NC: No connection.

PDLØ-PDL3: Paddle control inputs. Requires a Ø-15ØK ohm variable resistance and +5V for each paddle. Internal 1ØØ ohm resistors are provided in series with external pot to prevent excess current if pot goes completely to zero ohms.

SWØ-SW2: Switch inputs. Testable by reading from addresses CØ61-CØ63 (or CØ69-CØ6B). These are uncommitted 74LSxx series inputs.

+5V: Positive 5-Volt supply. To avoid burning out the connector pin, current drain MUST be less than 100mA.

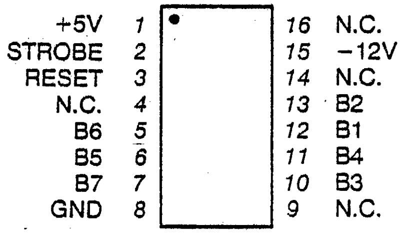
KEYBOARD CONNECTOR

This connector provides the means for connecting as ASCII keyboard to the APPLE II board. It is a 16 pin IC socket located at A7 and is illustrated in Figures 1 and 3.

Figure 3 KEYBOARD CONNECTOR

TOP VIEW

(Front Edge of PC Board)



LOCATION A7

SIGNAL DESCRIPTION FOR KEYBOARD INTERFACE

B1-B7: 7 bit ASCII data from keyboard, positive logic (high level= "1"), TTL logic levels expected.

GND: System circuit ground. 0 Volt line from power supply.

NC: No connection.

RESET: System reset input. Requires switch closure to ground.

STROBE: Strobe output from keyboard. The APPLE II recognizes the positive going edge of the incoming strobe.

+5V: Positive 5-Volt supply. To avoid burning out the connector pin, current drain MUST be less than 100mA.

-12V: Negative 12-Volt supply. Keyboard should draw less than 50mA.

PERIPHERAL CONNECTORS

The eight Peripheral Connectors mounted near the back edge of the APPLE II board provide a convenient means of connecting expansion hardware and peripheral devices to the APPLE II I/O Bus. These are Winchester #2HW25C0-111 (or equivalent) 50 pin card edge connectors with pins on .10" centers. Location and pin outs are illustrated in Figures 1 and 4.

SIGNAL DESCRIPTION FOR PERIPHERAL I/O

AØ-A15: 16 bit system address bus. Addresses are set up by the 6502 within 300nS after the beginning of $Ø_1$. These lines will drive up to a total of 16 standard TTL loads.

$\overline{\text{DEVICE SELECT}}$: Sixteen addresses are set aside for each peripheral connector. A read or write to such an address will send pin 41 on the selected connector low during $Ø_2$ (500nS). Each will drive 4 standard TTL loads.

DØ-D7: 8 bit system data bus. During a write cycle data is set up by the 6502 less than 300nS after the beginning of $Ø_2$. During a read cycle the 6502 expects data to be ready no less than 100nS before the end of $Ø_2$. These lines will drive up to a total of 8 total low power schottky TTL loads.

$\overline{\text{DMA}}$: Direct Memory Access control output. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

DMA IN: Direct Memory Access daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.

DMA OUT: Direct Memory Access daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.

GND: System circuit ground. 0 Volt line from power supply.

$\overline{\text{INH}}$: Inhibit Line. When a device pulls this line low, all ROM's on board are disabled (Hex addressed DØØØ through FFFF). This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

INT IN: Interrupt daisy chain input from higher priority peripheral devices. Will present no more than 4 standard TTL loads to the driving device.

INT OUT: Interrupt daisy chain output to lower priority peripheral devices. This line will drive 4 standard TTL loads.

$\overline{\text{I/O SELECT}}$: 256 addresses are set aside for each peripheral connector (see address map in "MEMORY" section). A read or write of such an address will send pin 1 on the selected connector low during $\emptyset_2$ (500nS). This line will drive 4 standard TTL loads.

$\overline{\text{I/O STROBE}}$: Pin 20 on all peripheral connectors will go low during $\emptyset_2$ of a read or write to any address C8ØØ-CFFF. This line will drive a total of 4 standard TTL loads.

$\overline{\text{IRQ}}$: Interrupt request line to the 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.

NC: No connection.

$\overline{\text{NMI}}$: Non Maskable Interrupt request line to the 6502. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output. It is active low.

Q_3: A 1MHz (nonsymmetrical) general purpose timing signal. Will drive up to a total of 16 standard TTL loads.

RDY: "Ready" line to the 6502. This line should change only during $\emptyset_1$, and when low will halt the microprocessor at the next READ cycle. This line has a 3K Ohm pullup to +5V and should be driven with an open collector output.

$\overline{\text{RES}}$: Reset line from "RESET" key on keyboard. Active low. Will drive 2 MOS loads per Peripheral Connector.

R/$\overline{W}$: READ/WRITE line from 6502. When high indicates that a read cycle is in progress, and when low that a write cycle is in progress. This line will drive up to a total of 16 standard TTL loads.

USER 1: The function of this line will be described in a later document.

$\emptyset_0$: Microprocessor phase 0 clock. Will drive up to a total of 16 standard TTL loads.

$\emptyset_1$: Phase 1 clock, complement of $\emptyset_0$. Will drive up to a total of 16 standard TTL loads.

7M: Seven MHz high frequency clock. Will drive up to a total of 16 standard TTL loads.

+12V: Positive 12-Volt supply.

+5V: Possitive 5-Volt supply

-5V: Negative 5-Volt supply.

-12V: Negative 12-Volt supply.

POWER CONNECTOR

The four voltages required by the APPLE II are supplied via this AMP #9-35Ø28-1,6 pin connector. See location and pin out in Figures 1 and 5.

PIN DESCRIPTION

GND: (2 pins) system circuit ground. Ø Volt line from power supply.

+12V: Positive 12-Volt line from power supply.

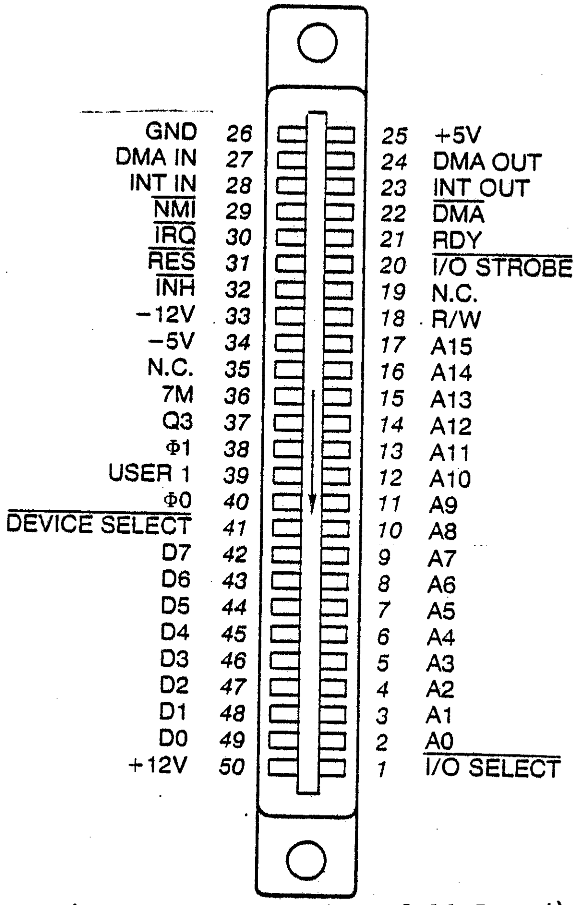
+5V: Positive 5-Volt line from power supply.

-5V: Negative 5-Volt line from power supply.

-12V: Negative 5-Volt line from power supply.

Figure 4 PERIPHERAL CONNECTORS
(EIGHT OF EACH)

TOP VIEW
PINOUT (Back Edge of PC Board)

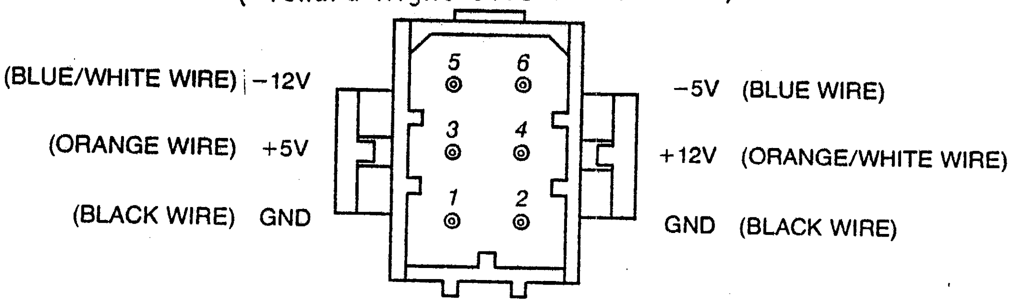


GND	26	25	+5V
DMA IN	27	24	DMA OUT
INT IN	28	23	INT OUT
$\overline{\text{NMI}}$	29	22	$\overline{\text{DMA}}$
$\overline{\text{IRQ}}$	30	21	RDY
$\overline{\text{RES}}$	31	20	$\overline{\text{I/O STROBE}}$
$\overline{\text{INH}}$	32	19	N.C.
−12V	33	18	R/W
−5V	34	17	A15
N.C.	35	16	A14
7M	36	15	A13
Q3	37	14	A12
Φ1	38	13	A11
USER 1	39	12	A10
Φ0	40	11	A9
$\overline{\text{DEVICE SELECT}}$	41	10	A8
D7	42	9	A7
D6	43	8	A6
D5	44	7	A5
D4	45	6	A4
D3	46	5	A3
D2	47	4	A2
D1	48	3	A1
D0	49	2	A0
+12V	50	1	$\overline{\text{I/O SELECT}}$

(Toward Front Edge of PC Board)
LOCATIONS J2 TO J12

Figure 5 POWER CONNECTOR

TOP VIEW
PINOUT (Toward Right Side of PC Board)



(BLUE/WHITE WIRE)	−12V	5	6	−5V	(BLUE WIRE)
(ORANGE WIRE)	+5V	3	4	+12V	(ORANGE/WHITE WIRE)
(BLACK WIRE)	GND	1	2	GND	(BLACK WIRE)

LOCATION K1

SPEAKER CONNECTOR

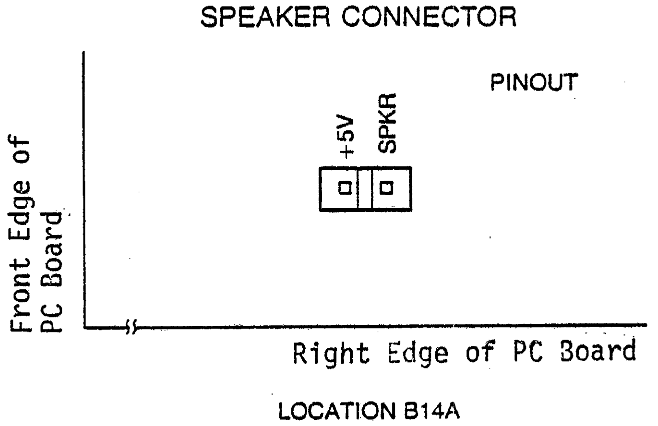
This is a MOLEX KK 1ØØ series connector with two .25" square pins on .1Ø" centers. See location and pin out in Figures 1 and 6.

SIGNAL DESCRIPTION FOR SPEAKER

+5V: System +5 Volts

SPKR: Output line to speaker. Will deliver about .5 watt into 8 Ohms.

Figure 6



VIDEO OUTPUT JACK

This standard RCA phono jack located at the back edge of the APPLE II P.C. board will supply NTSC compatible, EIA standard, positive composite video to an external video monitor.

A video level control near the connector allows the output level to be adjusted from Ø to 1 Volt (peak) into an external 75 OHM load.

Additional tint (hue) range is provided by an adjustable trimmer capacitor.

See locations illustrated in Figure 1.

AUXILIARY VIDEO OUTPUT CONNECTOR

This is a MOLEX KK 1ØØ series connector with four .25" square pins on .1Ø" centers. It provides composite video and two power supply voltages. Video out on this connector is not adjustable by the on board 200 Ohm trim pot. See Figures 1 and 7.

SIGNAL DESCRIPTION

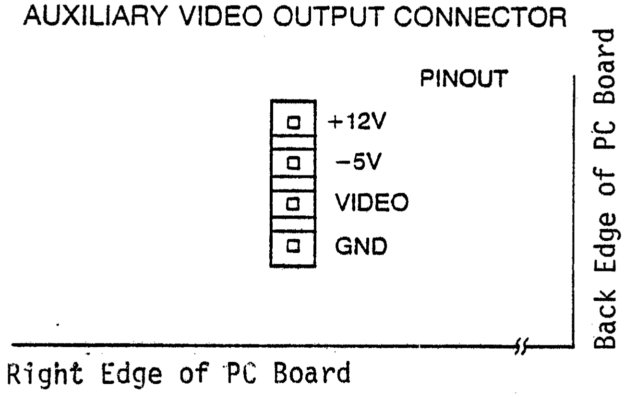
GND: System circuit ground. Ø Volt line from power supply.

VIDEO: NTSC compatible positive composite VIDEO. DC coupled emitter follower output (not short circuit protected). SYNC TIP is Ø Volts, black level is about .75 Volts, and white level is about 2.Ø Volts into 47Ø Ohms. Output level is non-adjustable.

+12V: +12 Volt line from power supply.

-5V: -5 Volt line from power supply.

Figure 7 AUXILIARY VIDEO OUTPUT CONNECTOR



LOCATION J14B

MEMORY

TABLE OF CONTENTS

INTRODUCTION

APPLE II is supplied completely tested with the specified amount of RAM memory and correct memory select jumpers. There are five different sets of standard memory jumper blocks:

1. 4K 4K 4K BASIC
2. 4K 4K 4K HIRES
3. 16K 4K 4K
4. 16K 16K 4K
5. 16K 16K 16K

A set of three each of one of the above is supplied with the board. Type 1 is supplied with 4K or 8K systems. Both type 1 and 2 are supplied with 12K systems. Type 1 is a contiguous memory range for maximum BASIC program size. Type 2 is non-contiguous and allows 8K dedicated to HIRES screen memory with approximately 2K of user BASIC space. Type 3 is supplied with 16K, 20K and 24K systems. Type 4 with 30K and 36K systems and type 5 with 48K systems.

Additional memory may easily be added just by plugging into sockets along with correct memory jumper blocks.

The 6502 microprocessor generates a 16 bit address, which allows 65536 (commonly called 65K) different memory locations to be specified. For convenience we represent each 16 bit (binary) address as a 4-digit hexadecimal number. Hexadecimal notation (hex) is explained in the Monitor section of this manual.

In the APPLE II, certain address ranges have been assigned to RAM memory, ROM memory, the I/O bus, and hardware functions. The memory and address maps give the details.

MEMORY SELECT SOCKETS

The location and pin out for memory select sockets are illustrated in Figures 1 and 8.

HOW TO USE

There are three MEMORY SELECT sockets, located at D1, E1 and F1 respectively. RAM memory is assigned to various address ranges by inserting jumper wires as described below. All three MEMORY SELECT sockets MUST be jumpered identically! The easiest way to do this is to use Apple supplied memory blocks.

Let us learnby example:

If you have plugged 16K RAMs into row "C" (the sockets located at C3-C1Ø on the board), and you want them to occupy the first 16K of addresses starting at ØØØØ, jumper pin 14 to pin 1Ø on all three MEMORY SELECT sockets (thereby assigning row "C" to the ØØØØ-3FFF range of memory).

If in addition you have inserted 4K RAMs into rows "D" and "E", and you want them each to occupy the first 4K addresses starting at 4ØØØ and 5ØØØ respectively, jumper pin 13 to pin 5 (thereby assigning row "D" to the 4ØØØ-4FFF range of memory), and jumper pin 12 to pin 6 (thereby assigning row "E" to the 5ØØØ-5FFF range of memory). Remember to jumper all three MEMORY SELECT sockets the same.

Now you have a large contiguous range of addresses filled with RAM memory. This is the 24K addresses from ØØØØ-5FFF.

By following the above examples you should be able to assign each row of RAM to any address range allowed on the MEMORY SELECT sockets. Remember that to do this properly you must know three things:

1. Which rows have RAM installed?

2. Which address ranges do you want them to occupy?

3. Jumper all three MEMORY SELECT sockets the same!

If you are not sure think carefully, essentially all the necessary information is given above.

INSTALLING YOUR OWN RAM

THE POSSIBILITIES

The APPLE II computer is designed to use dynamic RAM chips organized as 4096 x 1 bit, or 16384 x 1 bit called "4K" and "16K" RAMs respectively. These must be used in sets of 8 to match the system data bus (which is 8 bits wide) and are organized into rows of 8. Thus, each row may contain either 4096 (4K) or 16384 (16K) locations of Random Access Memory depending upon whether 4K or 16K chips are used. If all three rows on the APPLE II board are filled with 4K RAM chips, then 12288 (12K) memory locations will be available for storing programs or data, and if all three rows contain 16K RAM chips then 49152 (commonly called 48K) locations of RAM memory will exist on board!

RESTRICTIONS

It is quite possible to have the three rows of RAM sockets filled with any combination of 4K RAMs, 16K RAMs or empty as long as certain rules are followed:

1. All sockets in a row must have the same type (4K or 16K RAMs.

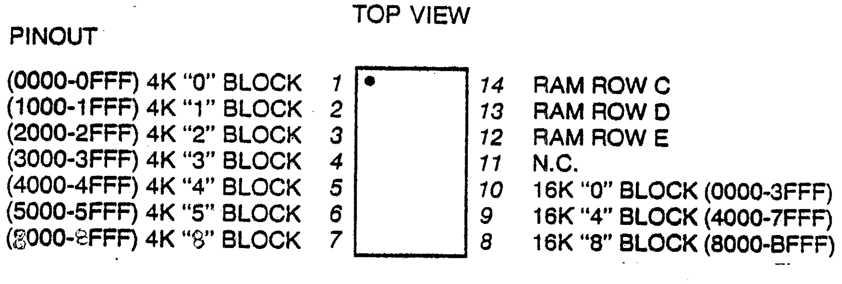
2. There MUST be RAM assigned to the zero block of addresses.

ASSIGNING RAM

The APPLE II has 48K addresses available for assignment of RAM memory. Since RAM can be installed in increments as small as 4K, a means of selecting which address range each row of memory chips will respond to has been provided by the inclusion of three MEMORY SELECT sockets on board.

Figure 8

MEMORY SELECT SOCKETS


TOP VIEW
PINOUT
(0000-0FFF) 4K "0" BLOCK 1
(1000-1FFF) 4K "1" BLOCK 2
(2000-2FFF) 4K "2" BLOCK 3
(3000-3FFF) 4K "3" BLOCK 4
(4000-4FFF) 4K "4" BLOCK 5
(5000-5FFF) 4K "5" BLOCK 6
(8000-8FFF) 4K "8" BLOCK 7
14 RAM ROW C
13 RAM ROW D
12 RAM ROW E
11 N.C.
10 16K "0" BLOCK (0000-3FFF)
9 16K "4" BLOCK (4000-7FFF)
8 16K "8" BLOCK (8000-BFFF)

LOCATIONS D1, E1, F1

Memory Address Allocations in 4K Bytes

0000	text and color graphics display pages, 6502 stack, pointers, etc.	8000	
1000		9000	
2000	high res graphics display primary page " "	A000	
3000	" " " "	B000	
4000	high res. graphics display secondary page " "	C000	addresses dedicated to hardware functions " "
5000	" " " "	D000	ROM socket D0: spare " ROM socket D8: spare "
6000		E000	ROM socket E0: BASIC " ROM socket E8: BASIC "
7000		F000	ROM socket F0: BASIC utility ROM socket F8: monitor

I/O and ROM Address Detail

HEX		
ADDRESS	ASSIGNED FUNCTION	COMMENTS
C00X	Keyboard input.	Keyboard strobe appears in bit 7. ASCII data from keyboard appears in the 7 lower bits.
C01X	Clear keyboard strobe.	
C02X	Toggle cassette output.	
C03X	Toggle speaker output.	
C04X	"$\overline{\text{C040 STB}}$"	Output strobe to Game I/O connector.
C050	Set graphics mode	
C051	" text "	
C052	Set bottom 4 lines graphics	
C053	" " " " text	
C054	Display primary page	
C055	" secondary page	
C056	Set high res. graphics	
C057	" color "	
C058	Clear "AN0"	Annunciator 0 output to Game I/O connector.
C059	Set "	
C05A	Clear "AN1"	Annunciator 1 output to Game I/O connector.
C05B	Set "	
C05C	Clear "AN2"	Annunciator 2 output to Game I/O connector.
C05D	Set "	
C05E	Clear "AN3"	Annunciator 3 output to Game I/O connector.
C05F	Set "	

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C060/8	Cassette input	State of "Cassette Data In" appears in bit 7.
C061/9	"SW1"	State of Switch 1 input on Game I/O connector appears in bit 7.
C062/A	"SW2"	State of Switch 2 input on Game I/O connector appears in bit 7.
C063/B	"SW3"	State of Switch 3 input on Game I/O connector appears in bit 7.
C064/C	Paddle 0 timer output	State of timer output for Paddle 0 appears in bit 7.
C065/D	" 1 " "	State of timer output for Paddle 1 appears in bit 7.
C066/E	" 2 " "	State of timer output for Paddle 2 appears in bit 7.
C067/F	" 3 " "	State of timer output for Paddle 3 appears in bit 7.
C07X	"$\overline{\text{PDL STB}}$"	Triggers paddle timers during $\emptyset_2$.
C08X	$\overline{\text{DEVICE SELECT}}$ 0	Pin 41 on the selected Peripheral Connector goes low during $\emptyset_2$.
C09X	" 1	
C0AX	" 2	
C0BX	" 3	
C0CX	" 4	
C0DX	" 5	
C0EX	" 6	
C0FX	" 7	
C10X	" 8	Expansion connectors.
C11X	" 9	"
C12X	" A	"

HEX ADDRESS	ASSIGNED FUNCTION	COMMENTS
C13X	$\overline{\text{DEVICE SELECT}}$ B	"
C14X	" C	"
C15X	" D	"
C16X	" E	"
C17X	" F	"
C1XX	$\overline{\text{I/O SELECT}}$ 1	Pin 1 on the selected Peripheral Connector goes low during ϕ_2.
C2XX	" 2	
C3XX	" 3	NOTES: 1. Peripheral Connector 0 does not get this signal. 2. $\overline{\text{I/O SELECT}}$ 1 uses the same addresses as $\overline{\text{DEVICE SELECT}}$ 8-F.
C4XX	" 4	
C5XX	" 5	
C6XX	" 6	
C7XX	" 7	
C8XX	" 8, $\overline{\text{I/O STROBE}}$	Expansion connectors.
C9XX	" 9, "	
CAXX	" A, "	
CBXX	" B, "	
CCXX	" C, "	
CDXX	" D, "	
CEXX	" E, "	
CFXX	" F, "	
D000-D7FF	ROM socket D0	Spare.
D800-DFFF	" " D8	Spare.
E000-E7FF	" " E0	BASIC.
E800-EFFF	" " E8	BASIC.
F000-F7FF	" " F0	1K of BASIC, 1K of utility.
F800-FFFF	" " F8	Monitor.

SYSTEM TIMING

SIGNAL DESCRIPTIONS

14M: Master oscillator output, 14.318 MHz +/- 35 ppm. All other timing signals are derived from this one.

7M: Intermediate timing signal, 7.159 MHz.

COLOR REF: Color reference frequency used by video circuitry, 3.580 MHz.

$\emptyset_0$: Phase 0 clock to microprocessor, 1.023 MHz nominal.

$\emptyset_1$: Microprocessor phase 1 clock, complement of $\emptyset_0$, 1.023 MHz nominal.

$\emptyset_2$: Same as $\emptyset_0$. Included here because the 6502 hardware and programming manuals use the designation $\emptyset_2$ instead of $\emptyset_0$.

Q3: A general purpose timing signal which occurs at the same rate as the microprocessor clocks but is nonsymmetrical.

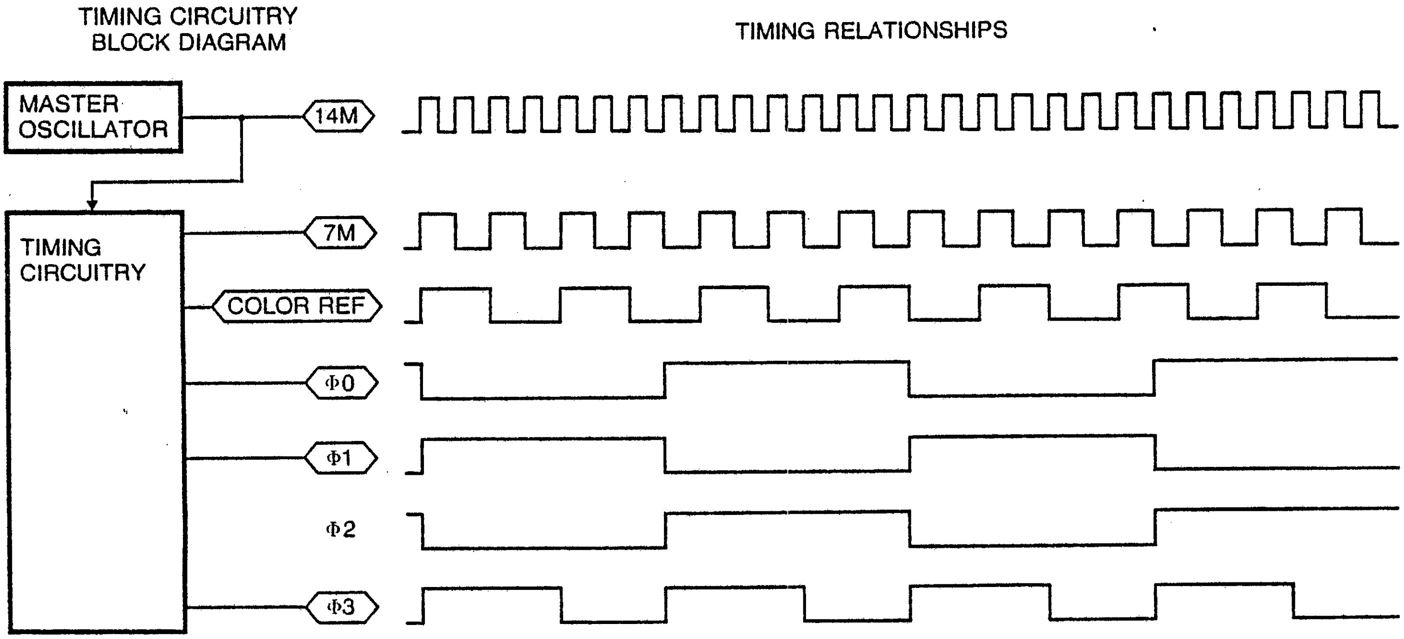
MICROPROCESSOR OPERATIONS

ADDRESS: The address from the microprocessor changes during $\emptyset_1$, and is stable about 300nS after the start of $\emptyset_1$.

DATA WRITE: During a write cycle, data from the microprocessor appears on the data bus during $\emptyset_2$, and is stable about 300nS after the start of $\emptyset_2$.

DATA READ: During a read cycle, the microprocessor will expect data to appear on the data bus no less than 100nS prior to the end of $\emptyset_2$.

SYSTEM TIMING DIAGRAM



TIMING CIRCUITRY BLOCK DIAGRAM

TIMING RELATIONSHIPS

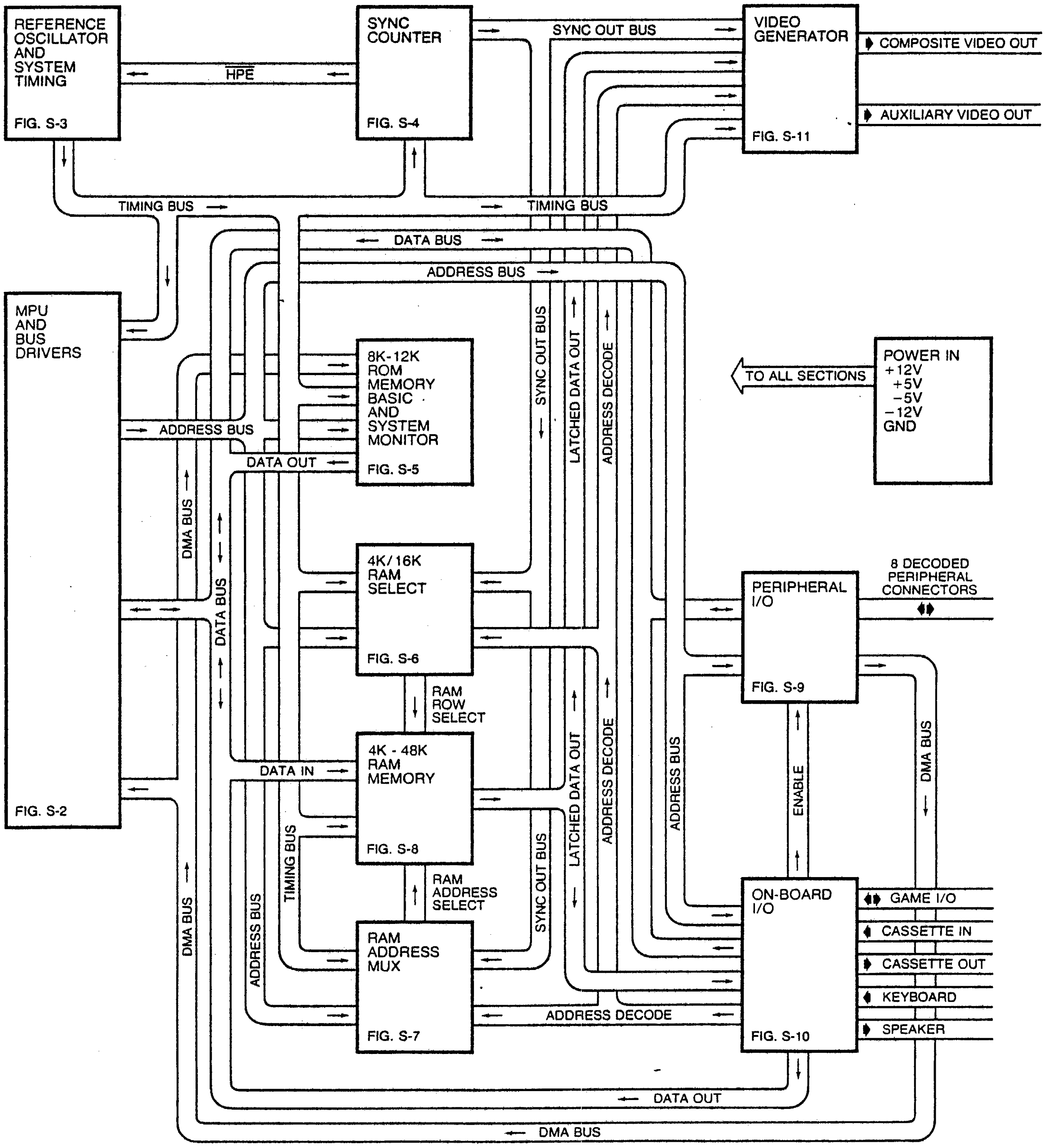


FIGURE S-1 APPLE II SYSTEM DIAGRAM

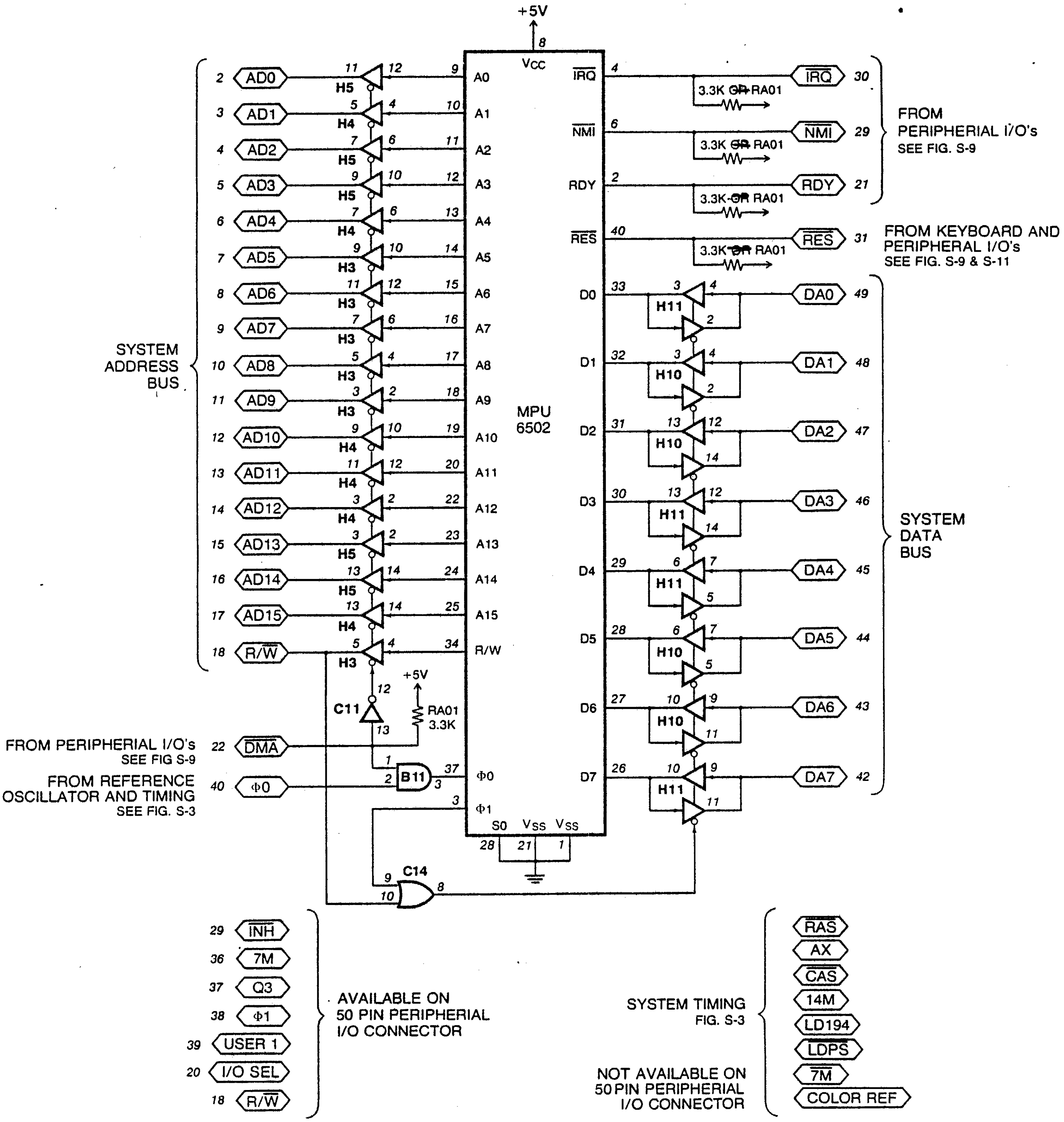


FIGURE S-2 MPU AND SYSTEM BUS

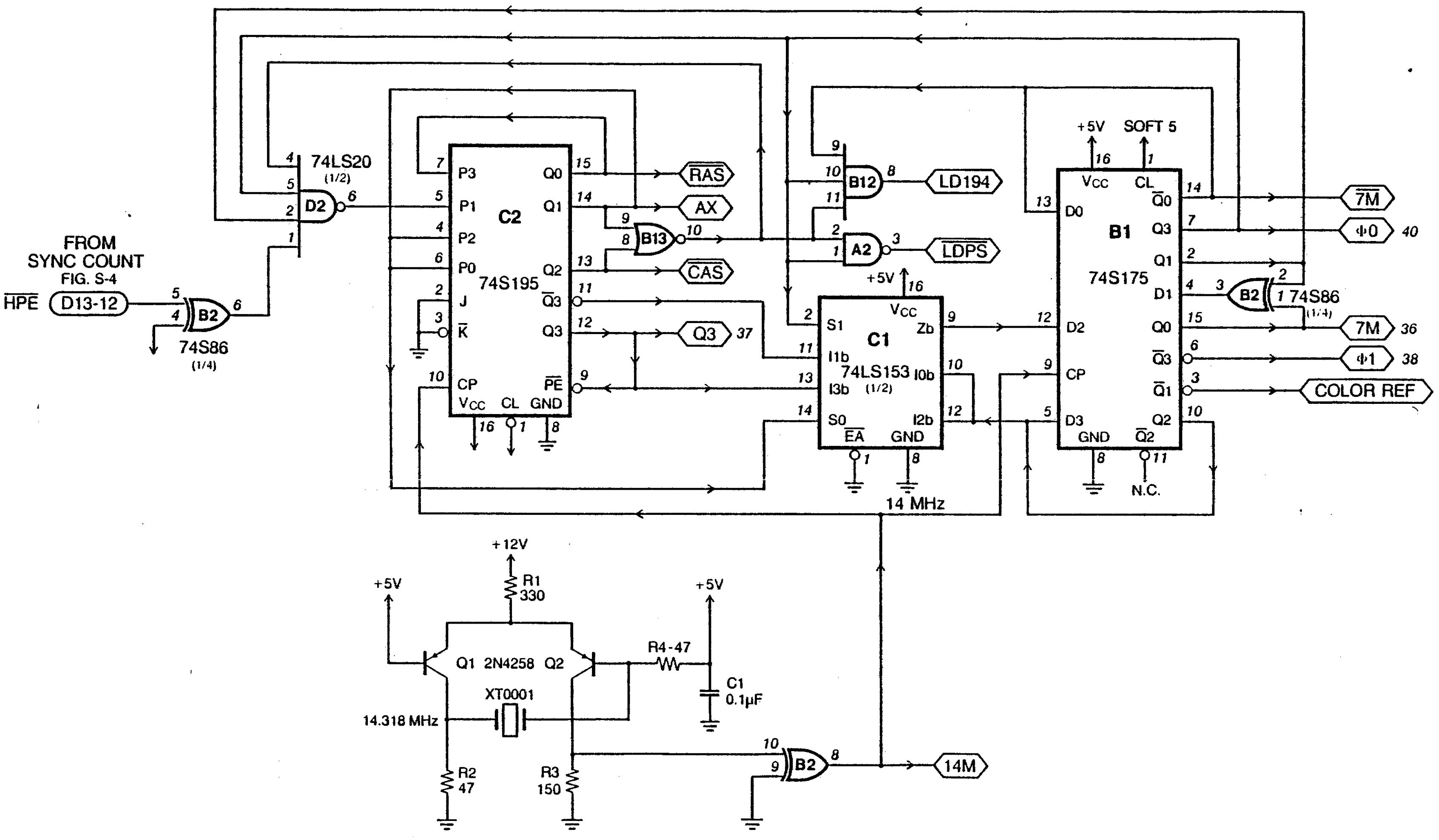


FIGURE S-3   REFERENCE OSCILLATOR AND SYSTEM TIMING

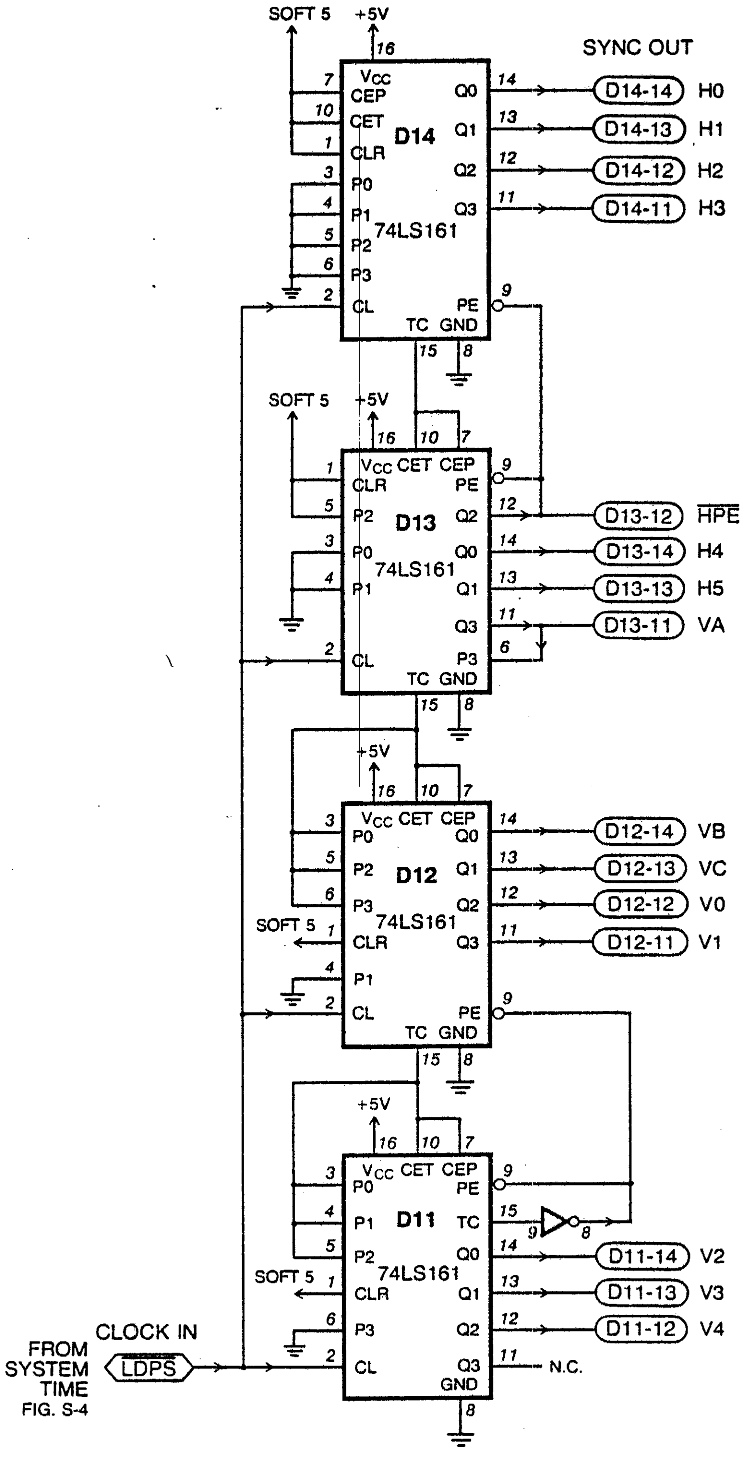


FIGURE S-4 SYNC COUNTER

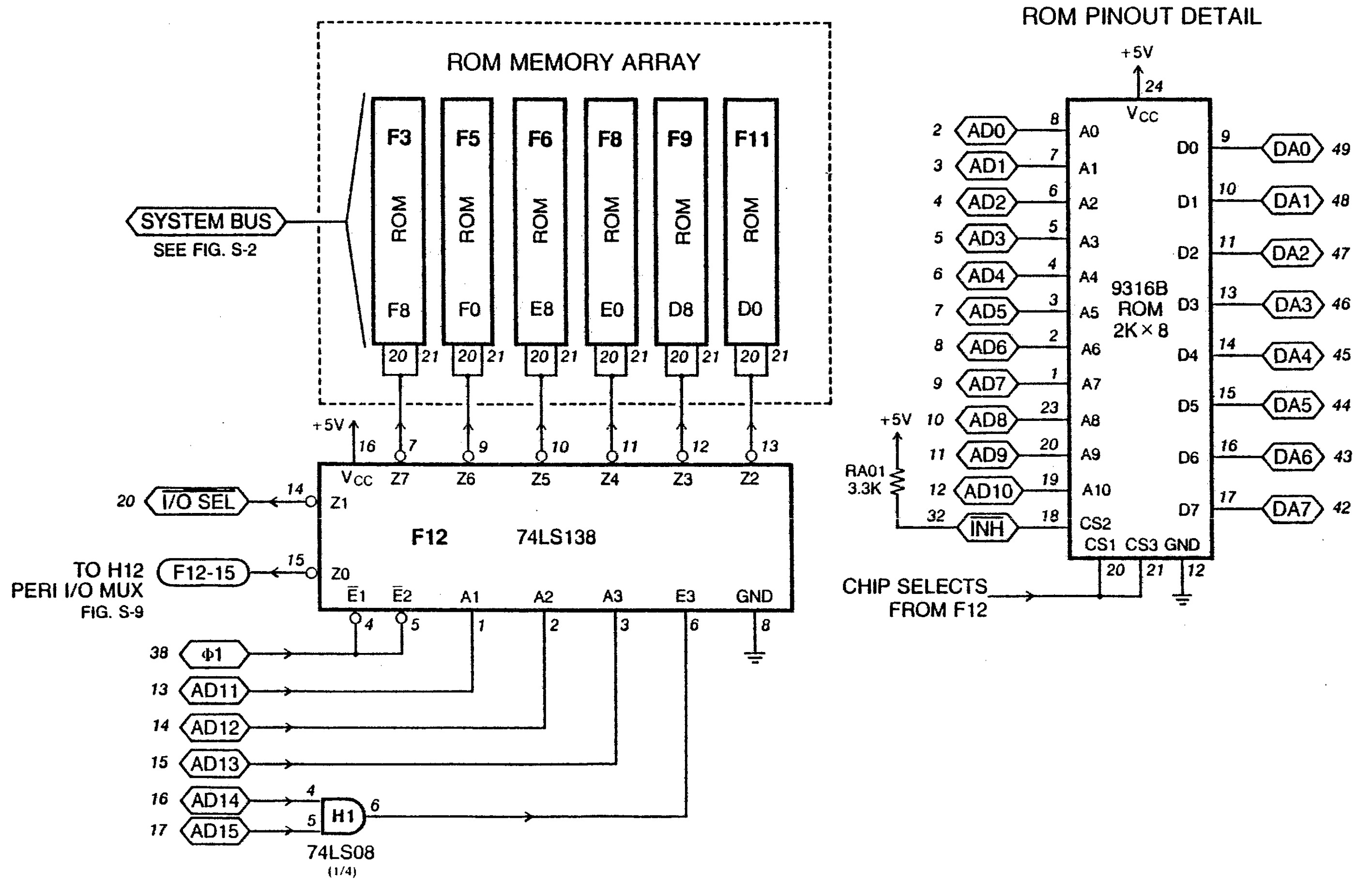


FIGURE S-5 ROM MEMORY

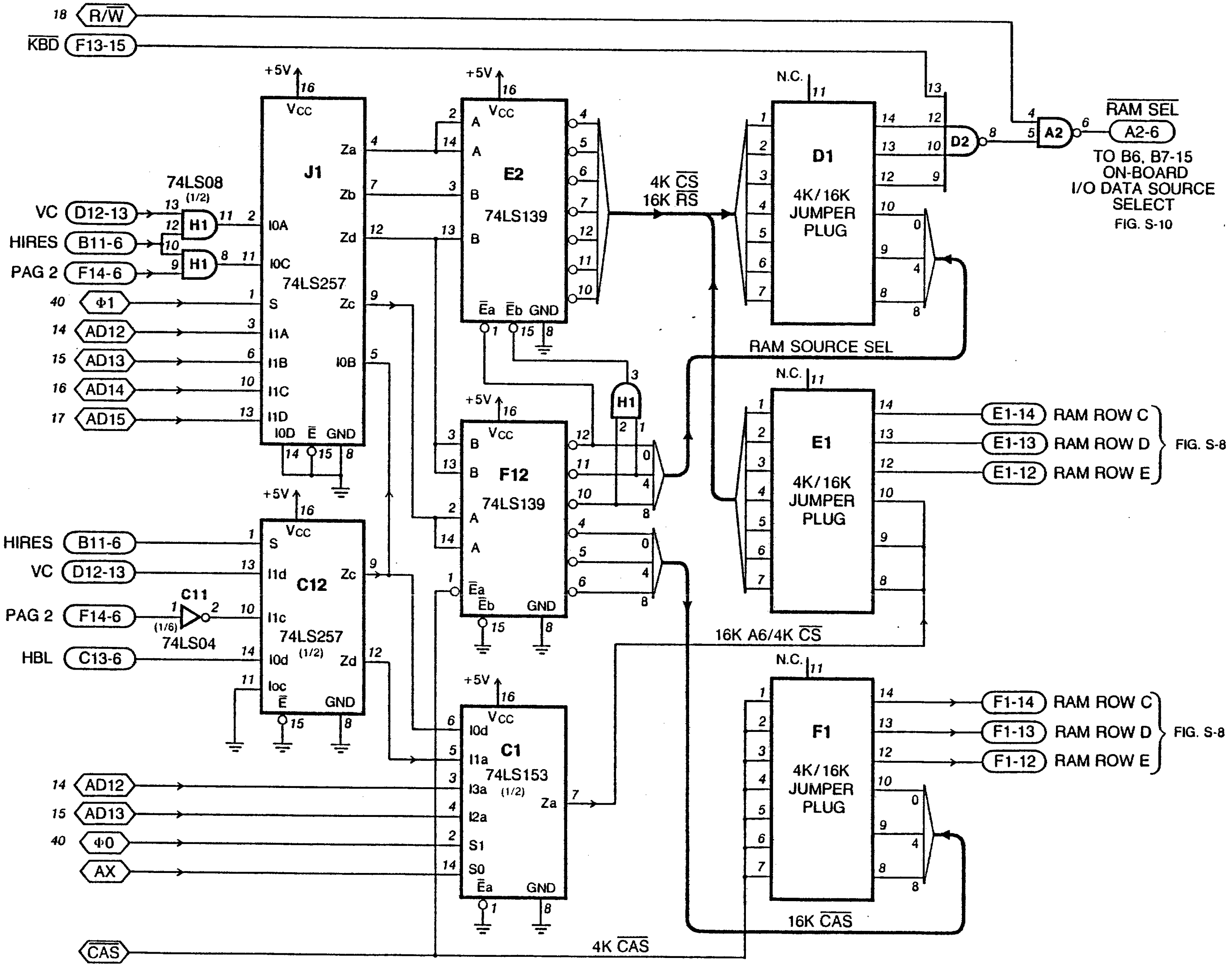


FIGURE S-6 4K/16K RAM SELECT

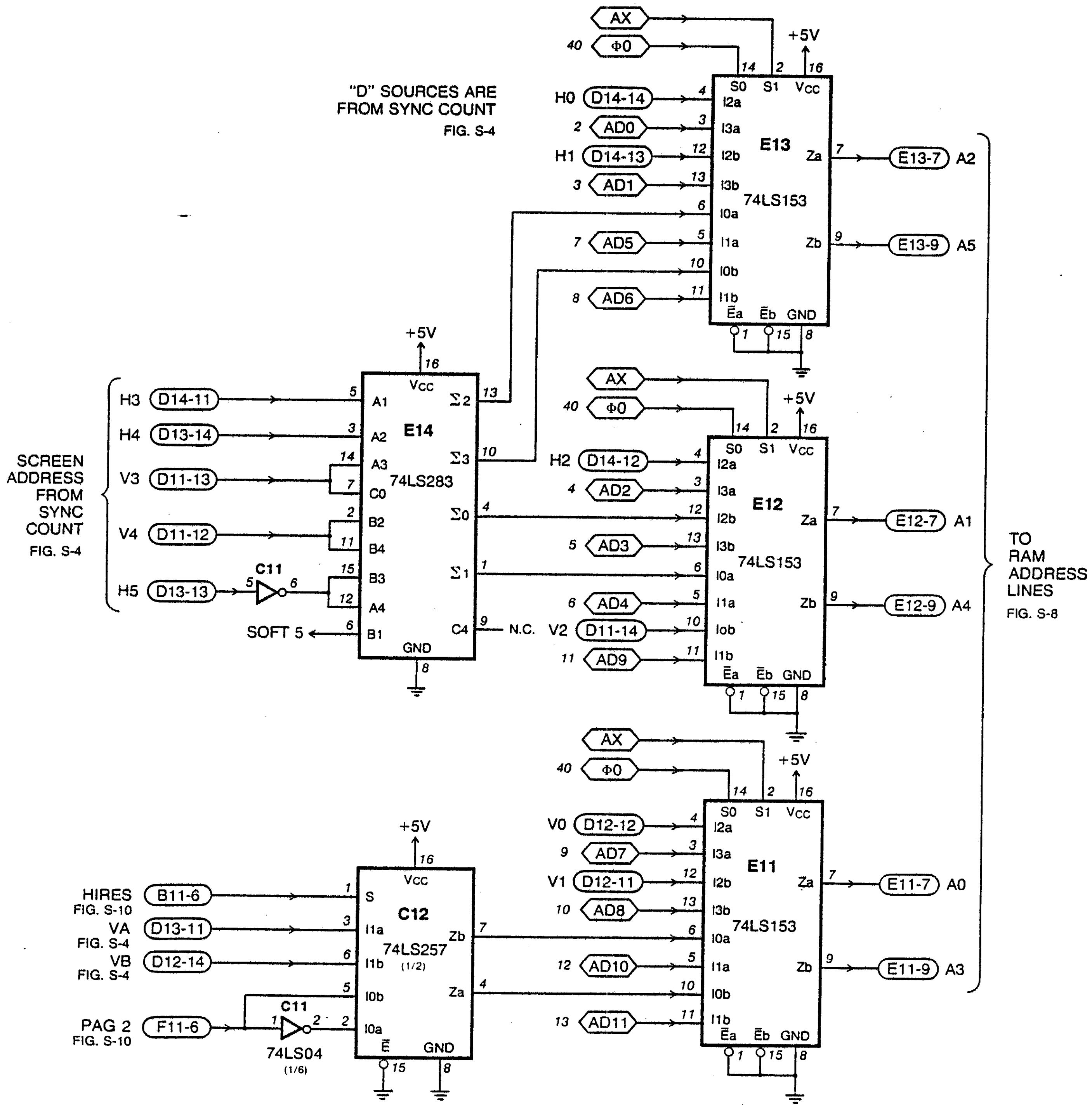


FIGURE S-7 RAM ADDRESS MUX

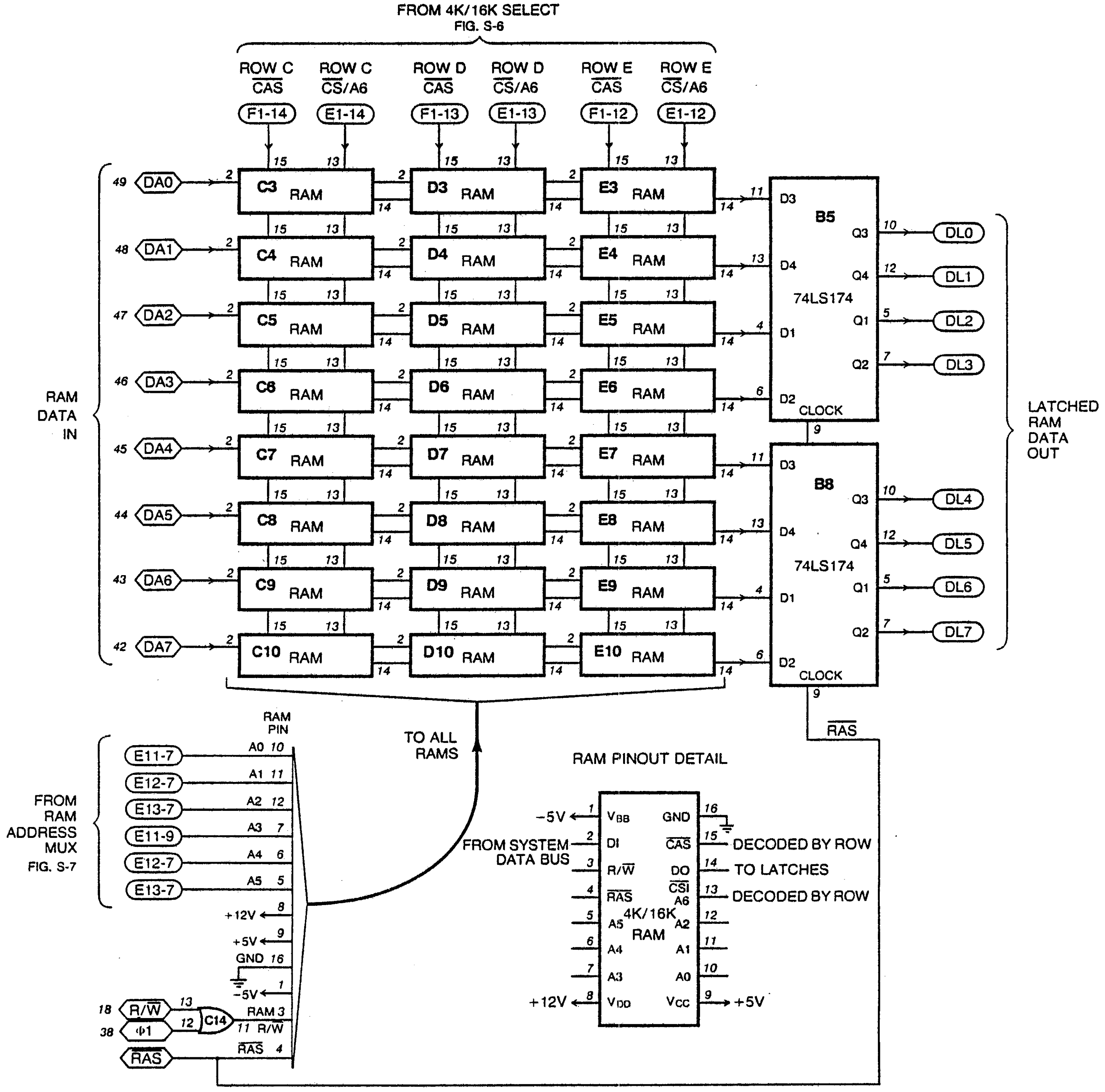


FIGURE S-8 4K TO 48K RAM MEMORY WITH DATA LATCH

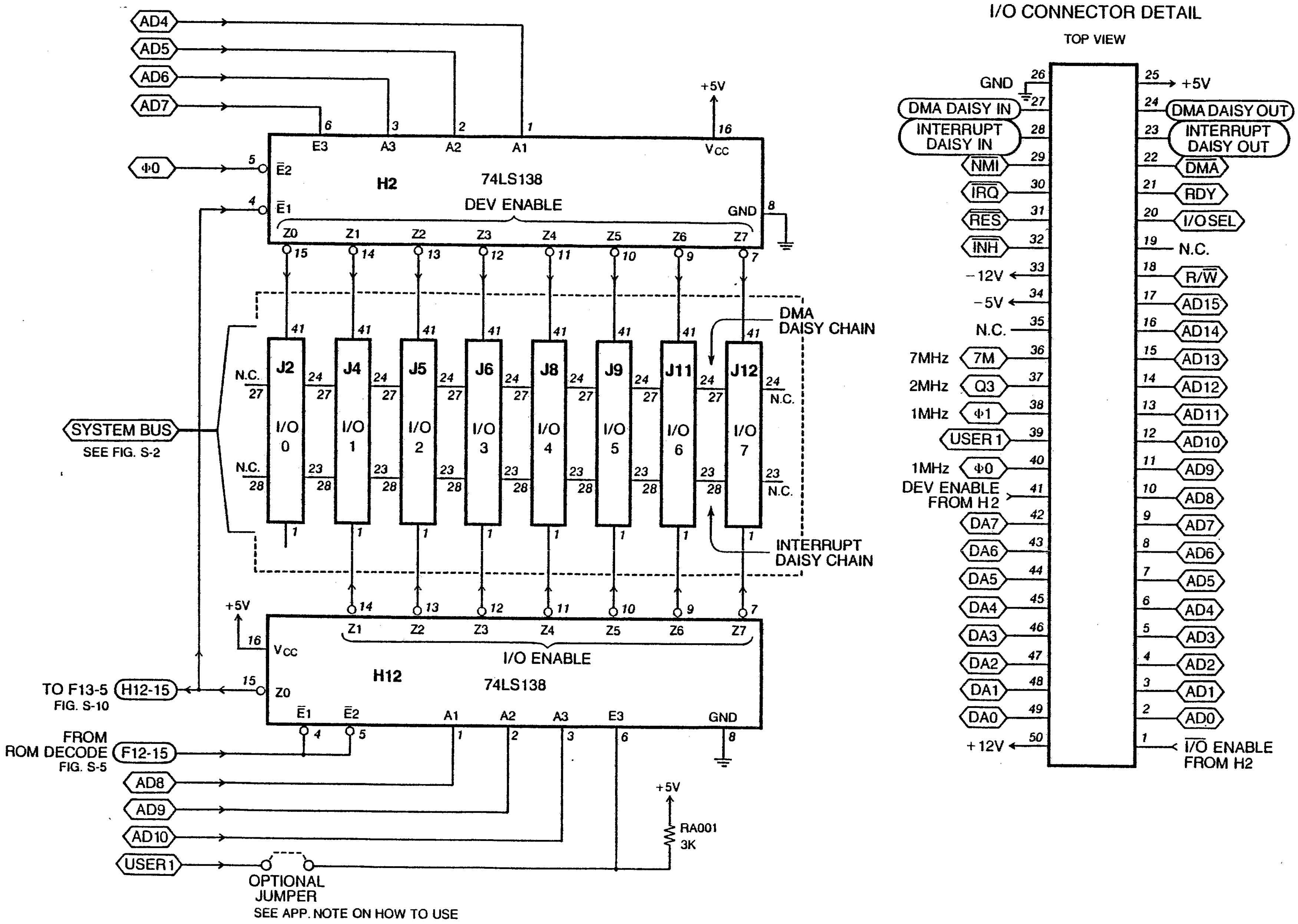


FIGURE S-9 PERIPHERIAL I/O CONNECTOR PINOUT AND CONTROL LOGIC

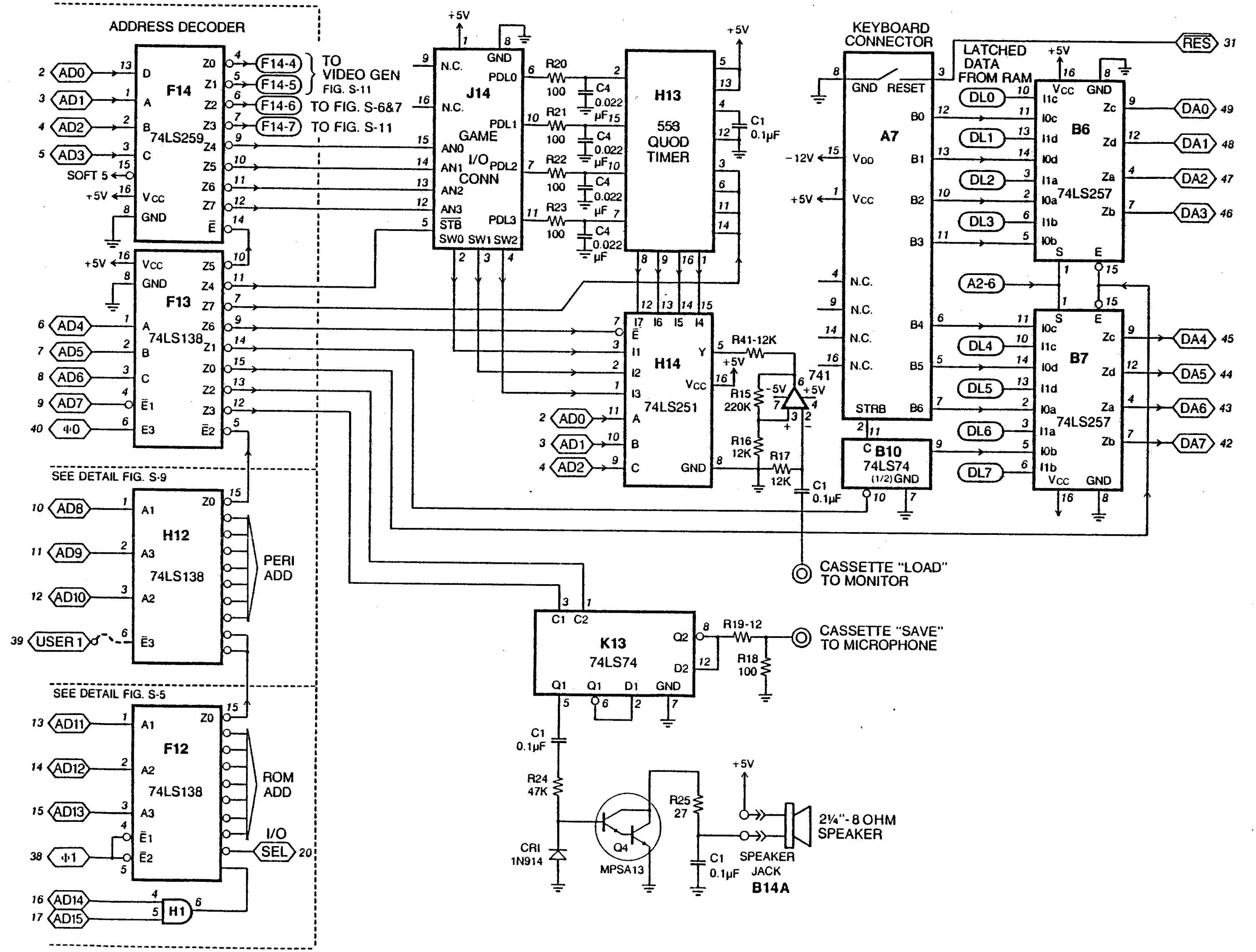


FIGURE S-10 ON-BOARD I/O

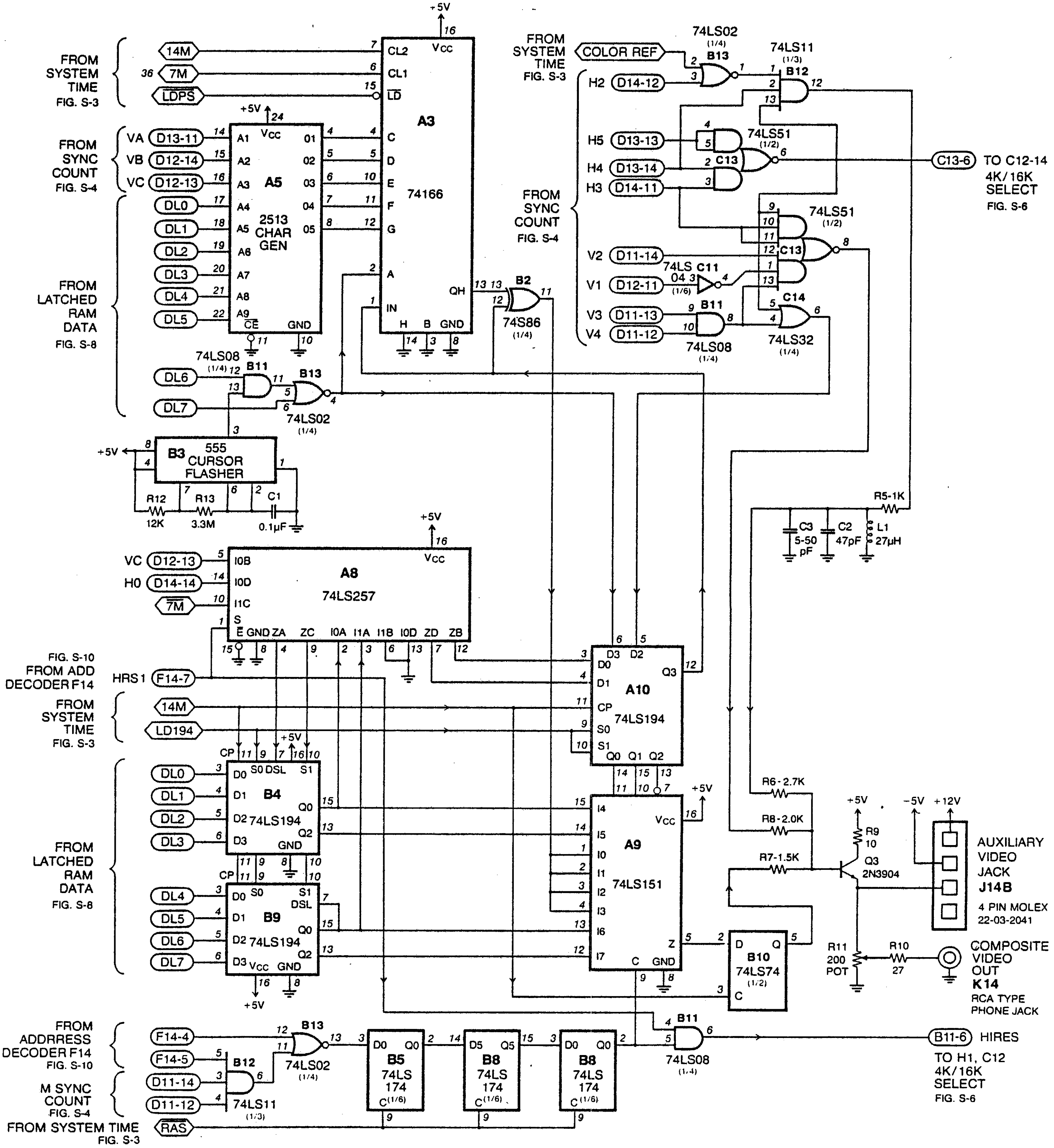


FIGURE S-11 APPLE COMPUTER VIDEO GENERATOR

APPLE II MANUAL

PROGRAM LISTING FOR CASSETTE TAPE DEMOS

CONTENTS

NOTES:

a. Color math will run by itself if a control P is typed in response to a "?". Manual operation will resume if any keyboard character is typed.

b. Breakout will "play itself" if PDL(Ø) switch is depressed when ball hits top or bottom wall.

c. Pong is included as an example from APPLE-cation Note 1: "How to Program "Pong" type games in Apple BASIC" which is now being printed.

COLOR GRAPHICS DEMO LISTING

```
10 DIM C(4): GOSUB 2ØØØØ: TEXT
   :Z1=2ØØØ: CALL -936: VTAB 4
   : TAB 8: PRINT "4K COLOR DEMOS"
   : PRINT : PRINT "1 LINES": PRINT
   "2 CROSS": PRINT "3 WEAVING"

20 PRINT "4 TUNNEL": PRINT "5 CIRCL
   E": PRINT "6 SPIRAL **": PRINT
   "7 TONES ** ": PRINT "8 SPRING"

30 PRINT "9 HYPERBOLA": PRINT
   "1Ø ??????": PRINT : PRINT
   "** NEEDS PDL(Ø) CONNECTED"
   : PRINT
40 PRINT "TYPE CTL C & RTN TO STOP"
   : PRINT : INPUT "WHICH DEMO # "
   ,I: GR : IF I>Ø AND I<11 THEN
   GOTO 1ØØ*I: GOTO 1Ø
60 INPUT "WHICH DEMO WOULD YOU LIKE
   ",I: GR : IF I AND I<RØ THEN
   GOTO 1ØØ*I: GOTO 1Ø
1ØØ I=(I+1) MOD 8Ø:J=I+(I>39)*(
    79-I-I): GOSUB 2ØØØ: GOTO 1ØØ

2ØØ I=(I+1) MOD 4Ø:J=IZ GOSUB 2ØØØ
    :J=39-I: GOSUB 2ØØØ: GOTO 2ØØ

3ØØ J=(J+1) MOD 22: FOR I=1 TO
    1295: COLOR=I MOD J+7: PLOT
    (2*I) MOD 37,(3*I) MOD 35: NEXT
    I: FOR Z=Ø TO Z1: NEXT Z: GOTO
    3ØØ
4ØØ FOR I=1 TO 4:C(I)= RND (16)
    : NEXT I:
41Ø FOR I=3 TO 1 STEP -1:C(I+1)
    =C(I): NEXT I:C(1)= RND (16
    ): FOR I=1 TO 5: FOR J=1 TO
    4
42Ø COLOR=C(J):L=J*5+14+I:K=39-
    L: HLIN K,L AT K: VLIN K,L AT
    L: HLIN K,L AT L: VLIN K,L AT
    K: NEXT J,I: GOTO 41Ø
5ØØ Z=2Ø: GOTO 9ØØ
6ØØ COLOR= RND (16): FOR I=Ø TO
    18 STEP 2:J=39-I: HLIN I,J AT
    I: GOSUB 64Ø: VLIN I,J AT J:
     GOSUB 64Ø
61Ø HLIN I+2,J AT J: GOSUB 64Ø:
     VLIN I+2,J AT I+2: GOSUB 64Ø
    : NEXT I: GOTO 62Ø
62Ø COLOR= RND (16): FOR I=18 TO
    Ø STEP -2:J=3Y-I: VLIN I+2,
    J AT I+2: GOSUB 64Ø: HLIN I+
    2,J AT J: GOSUB 64Ø
63Ø VLIN I,J AT J: GOSUB 64Ø: HLIN
    I,J AT I: GOSUB 64Ø: NEXT I:
     GOTO 6ØØ
64Ø K=I+7:L=K*K*5+K*26+7Ø:L=32767
    /L*( PDL (Ø)/1Ø): POKE Ø,K:
     POKE 1,L MOD 256: POKE 24,
    L/256+1: CALL 2: RETURN
```

COLOR GRAPHICS DEMO LISTING - Continued

```
  700 I= RND (3Ø)+3:J=I*I*5+I*26+
      7Ø:K=32767/J*( PDL (Ø)/1Ø):
       POKE Ø,I: POKE 1,K MOD 256
      : POKE 24,(K>255)+1: CALL 2
      : GOTO 7ØØ
  8ØØ X=3:YA=1ØØØ:YP=YA:L=2Ø:IW=4
      :DEL=1: COLOR=6: HLIN Ø,39 AT
      4: COLOR=9: GOSUB 88Ø: COLOR=
      12: VLIN 5,LM-2 AT X
  81Ø YN=2*YA-YP-YA/IW: COLORJØ: GOSUB
      88Ø: VLIN 5,39 AT X:X=X+1: IF
      X<39 THEN 82Ø:X=3: VLIN 5,39
       AT 1: VLIN 5,39 AT 2
  82Ø YP=YA:YA=YN:Y=YA/1ØØ: COLOR=
      12: GOSUB 88P: COLOR=9: VLIN
      5,LM-2 AT X: COLORJ15: PLOT
      X-2,LM: FOR I=Ø TO DEL: NEXT
      I: GOTO 81Ø
  88Ø LM=L-Y:L1=LM-1:L2=LM+Q: VLIN
      L1,L2 AT X-1: VLIN L1,LR AT
      X: VLIN L1,L2 AT X+1: RETURN

  9ØØ I=(I+1) MOD 16: FOR Y=Ø TO
      39: FOR X=Ø TO 39: COLOR=I+
      ( ABS (2Ø-X)-Z)*( ABS (2Ø-Y)
     @-Z)/25: PLOT X,Y: NEXT X,Y:
       FOR J=Ø TO Z1: NEXT J: GOTO
      9ØØ
 1ØØØ K=(K+1) MOD 326ØØ: FOR IJØ TO
      1479: COLOR=I MOD 9+6: PLOT
      I/37,(K*I) MOD 37: NEXT I: FOR
      Z=Ø TO Z1: NEXT Z: GOTO 1ØØØ

 11ØØ J=(J+1) MOD 22: FOR I=1 TO
      12Y5: COLOR=I MOD J+7: PLOT
      2*I MOD 37,3*I MOD 35: NEXT
      I: FOR Z=Ø TO 3ØØØ: NEXT Z:
       GOTO 11ØØ
 12ØØ K=(K+1) MOD 326ØØ: FOR I=Ø TO
      1479: COLOR=I MOD 9+6: PLOT
      I/37,(K*I) MOD 37: NEXT I: FOR
      Z=Ø TO 3ØØØ: NEXT Z: GOTO 12ØØ

 2ØØØ COLOR= RND (16): HLIN Ø,39 AT
      J: COLOR= RND (16): VLIN Ø,
      39 AT J: RETURN
2ØØØØ POKE 2,173: POKE 3,48: POKE
      4,192: POKE 5,165: POKE 6,Ø
      : POKE 7,32: POKE 8,168: POKE
      9,252: POKE 1Ø,165: POKE 11
      ,1: POKE 12,2Ø8: POKE 13,4
2ØØØ5 POKE 14,198: POKE 15,24: POKE
      16,2TØ: POKE 17,5: POKE 18,
      198: POKE 19,1: POKE 2Ø,76:
       POKE 21,2: POKE 22,Ø: POKE
      23,96: RETURN
```

COLOR MATH

```
  5 DIM A$(54): GOSUB 1Ø42: REM COLO
    RMATH:M.MARKKULA
 1Ø GR : FOR P=Ø TO 24 STEP 12
 2Ø A= RND (1Ø):B= RND (1Ø)
 3Ø S= RND (4):Q= RND (14)+1:Q=
    Q+(Q>9): IF Q]LQ THEN 3Ø: COLOR=
    Q:LQ=Q: IF S=1 AND B<A THEN
    34
 32 D=A: A=B: B=D
 34 IF S#3 THEN 45:B=B+ NOT B:A=
    A*B:X=P:Y=Ø:N=A/1Ø: COLOR=Q*
     NOT NOT N: GOSUB 9ØØ: COLOR=
    Q: GOTO 7Ø
 45 Y=Ø
 7Ø X=P+6:N=A MOD 1Ø: GOSUB 9ØØ
    :Y=8:N=B: GOSUB 9ØØ
1ØØ X=P: GOSUB 1Ø2Ø+S+S
11Ø HLIN P,P+11 AT 16
12Ø X=P+6:Y=18:I=Ø
121 I=Q-I: COLOR=I: GOSUB 1Ø28:
    Z=Ø
122 Z=Z+1: IF PEEK (-16384)>127
     THEN 13Ø:Z=Z+1: GOTO 121+(
    Z<1ØØ)
13Ø O= PEEK (-16384)#1T4: COLOR=
    Ø: GOSUB 1Ø28
17Ø IF O THEN 171: GOSUB 3ØØØ:C=
    RE: GOTO 174
171 INPUT C: IF C>=Ø THEN 174: PRINT
    "NO MINUS SIGNS";
172 PRINT "...TRY AGAIN": GOSUB
    2ØØØ: GOTO 12Ø
174 IF C<1ØØ THEN 18Ø: PRINT "TOO MU
    CH";: GOTO 172
18Ø N=C/1Ø: COLOR=Q:X=P:Y=18
2ØØ IF NOT N THEN 22Ø: GOSUB 9ØØ

22Ø N=C MOD 1Ø:X=P+6: GOSUB 9ØØ

25Ø GOSUB 3ØØØ: IF C#RE THEN 31Ø
    : COLOR=Ø: GOSUB 1Ø45Z GOSUB
    1Ø43: NEXT P: GOTO 1Ø
31Ø X1=X:Y1=Y: GOSUB 1Ø42
315 Y=Y1:X=X1
32Ø GOSUB@2ØØØ: FOR Z=1 TO 5ØØ:
     NEXT Z: COLOR=Ø: FOR M=0 TO
    1Ø: VLIN Y,Y+7 AT X-6+M: NEXT
    M: COLOR=Q
35Ø GOTO 12Ø
9ØØ IF N=1 OR N=4 OR N=7 OR N=8
     THEN 92Ø
91Ø GOSUB 1Ø16: COLOR=Ø
92Ø GOSUB 1ØØØ+N+N: COLOR=Q: RETURN
```

COLOR MATH - Continued

```
1000 HLIN X+1,X+3 AT Y+3: RETURN

1002 VLIN Y,Y+6 AT X+4: RETURN
1004 VLIN Y+1,Y+2 AT X: PLOT X+4
     ,Y+4: RETURN
1006 VLIN Y+2,Y+4 AT X: PLOT X+1
     ,Y+4: RETURN
1008 VLIN Y,Y+3 AT X: HLIN X+1,X+
     3 AT Y+3: VLIN Y,Y+6 AT X+4
     : RETURN
1010 VLIN Y+1,Y+2 AT X+4: PLOT X,
   @ Y+4: RETURN
1012 VLIN Y,Y+2 AT X+4: RETURN
1014 HLIN X+1,X+4 AT Y: VLIN Y+1
     ,Y+6 AT X+4: RETURN
1016 VLIN Y,Y+6 AT X: HLIN X+1,X+
     3 AT Y: HLIN X+1,X+3 AT Y+3
     : HLIN X+1,X+3 AT Y+V: VLIN
     Y,Y+6 AT X+4: RETURN
1018 VLIN Y+4,Y+7 AT X: RETURN
1020 HLIN X+1,X+3 AT Y+3: VLIN Y+
     1,Y+5 AT X+2: RETURN
1022 HLIN X+1,X+3 AT Y+3: RETURN

1024 FOR Z=0 TO 4: PLOT X+Z,Y+1+
     Z: PLOT X+4-Z,Y+1+Z: NEXT Z:
      RETURN
1026 HLIN X,X+4 AT Y+3: PLOT X+2
     ,Y+1: PLOT X+R,Y+5: RETURN

1028 VLIN Y,Y+1 AT X: HLIN X+1,X+
     4 AT Y: VLIN Y+1,Y+3 AT X+4
     : HLIN X+2,X+4 AT Y+3: PLOT
     X+2,Y+4: PLOT X+2,Y+6: RETURN

1042 A$="42026118203350373330:40456450
   @ 3673610742790726842942042O"
     : GOTO 1044
1043 A$="42026118203350373330:40153156
     159150264268261277272268429"

1044 COLOR= RND (15)+1
1045 X=P:Y=R7: FOR Z=1 TO 52 STEP
     3:Z1= ASC(A$(Z))-176+X: HLIN
     Z1,Z1+ ASC(A$(Z+1))-176 AT
     ASC(A$(Z+2))-176+Y: NEXT Z:
      RETURN
2000 FOR Z=1 TO 50:M= PEEK (-16336
     )- PEEK (-16336)+ PEEK (-163S6
     ): NEXT Z: RETURN
2010 FOR Z=1 TO 52 STEP 3:Z1= ASC(
     A$(Z,Z))-176+X: HLIN Z1,Z1+
      ASC(A$(Z+Q,Z+1))-176 AT Y+
      ASC(A$(Z+2,Z+2))-176: NEXT
     Z
3000 RE=A+ NOT S*B-S*B: IF S>1 THEN
     RE=A*B/(S/3*B*B+(S=2)): RETURN
```

BREAKOUT

```
 5 TEXT : CALL -936: VTAB 4: TAB
   1Ø: PRINT "*** BREAKOUT GAME ***
          ": PRINT
 7 PRINT "  OBJECT IS TO DESTROY AL
   L BRICKS WITH 5 BALLS": FOR
   N=1 TO 7ØØØ: NEXT N
1Ø DIM A$(2Ø),B$(2Ø): GR : PRINT
   : INPUT "HI, WHAT'S YOUR NAME? "
   ,A$: A=1: B=13: C=9: D=6: E=15: PRINT
   "STANDARD COLORS, ";A$;
2Ø INPUT "? ",B$: IF B$#"N" AND
   B$#"NO" THEN 3Ø: FOR I=Ø TO
   3Y: COLOR=I/2*(I<32): VLIN
   Ø,39 AT I
25 NEXT I: POKE 34,2Ø: PRINT :
    PRINT : PRINT : FOR I=Ø TO
   15: VTAB 21+I MOD 2: TAB I+
   I+1: PRINT I;: NEXT I: POKE
   34,22: VTAB 24: PRINT : PRINT
   "BACKGROUND";
27 GOSUB 1ØØ: A=E: PRINT "EVEN BRICK
   ";: GOSUB 1ØØ: B=E: PRINT "ODD BR
   ICK";: GOSUB 1ØØ: C=E: PRINT
   "PADDLE";: GOSUB 1ØØ: D=E: PRINT
   "BALL";: GOSUB 1ØØ
3Ø POKE 34,2Ø: COLOR=A: FOR I=
   Ø TO 39: VLIN Ø,39 AT I: NEXT
   I: FOR I=2Ø TO 34 STEP 2: TAB
   I+1: PRINT I/2-9;: COLOR=B:
    VLIN Ø,39 AT I: COLOR=C: FOR
   J=I MOD 4 TO 39 STEP 1 4
35 VLIN J,J+1 AT I: NEXT J,I: TAB
   5: PRINT "SCORE = Ø": PRINT
   : PRINT : POKE 34,21: S=Ø: P=
   S: L=S: X=19: Y=19: X=19
4Ø COLOR=A: PLOT X,Y/3: X=19: Y=
    RND (12Ø): V=-1: W= RND (5)-
   2: L=L+1: IF L>5 THEN 14Ø: TAB
   6: PRINT "BALL #";L: PRINT
   : FOR I=1 TO 1ØØ: GOSUB 2ØØ
   : NEXT I: M=1: N=Ø
5P J=Y+W: IF J>=Ø AND J<12Ø THEN
   6Ø: W=-W: J=Y: FOR I=1 TO 6: K=
    PEEK (-16336): NEXT I
55 IF PEEK (-16287)>127 THEN SW=
   1-SW
6Ø I=X+V: IF I<Ø THEN 4ØØ: GOSUB
   2ØØ: COLOR=A: K=J/3: IF I>39
    THEN 7Ø: IF SCRN(I,K)=A THEN
   9Ø: IF I THEN 12Ø: N=N+1: V=(
   N>9)+1: W=(K-P)*2-5: M=1
65 Z= PEEK (-16336)- PEEK (-16336
   )+ PEEK (-16336)- PEEK (-16336
   )+ PEEK (-16336)- PEEK (-16336
   )+ PEEK (-16336): GOTO 9Ø
7Ø FOR I=1 TO 6: M= PEEK (-16336
   ): NEXT I: I=X: M=Ø
8Ø V=-V
9Ø PLOT X,Y/3: COLOR=E: PLOT I,
   K: X=I: Y=J: GOTO 5Ø
```

BREAKOUT - Continued

```
 99 PRINT "INVALID.  REENTER";
100 INPUT " COLOR (Ø TO 15)",E:
   @ IF E<Ø OR E>15 THEN 99: RETURN

120 IF M THEN V= ABS (V): VLIN
"    K/2*2,K/2*2+1 AT I: S=S+I/2-
    9: VTAB 2Q: TAB 13: PRINT S
123 Q= PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16366)- PEEK (-1633V
    )+ PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )+ PEEK (-16336)- PEEK (-16336
    )
124 IF S<72Ø THEN 8Ø
13Ø PRINT "CONGRATULATIONS, YOU WIN.
    ": GOTO 15Ø
14Ø PRINT "YOUR SCORE OF ";S;" IS "
    ;: GOTO 141+S/1ØØ
141 PRINT "TERRIBLE!": GOTO 15Ø

142 PRINT "LOUSY.": GOTO 15Ø
143 PRINT "POOR.": GOTO 15Ø
144 PRINT "FAIR.": GOTO 15Ø
145 PRINT "GOOD.": GOTO 15Ø
146 PRINT "VERY GOOD.": GOTO 15Ø

147 PRINT "EXCELLENT.": GOTO 15Ø

148 PRINT "NEARLY PERFECT."
15Ø PRINT "SAME COLORS";: GOTO
    2Ø
2ØØ IF SW THEN 22Ø: Q=( PDL (Ø)-
    5)/6: IF Q<Ø THEN Q=Ø
2Ø5 IF Q>=34 THEN Q=34: COLOR=D:
     VLIN Q,Q+5 AT Ø: COLOR=A: IF
    P>Q THEN 21Ø: IF Q THEN VLIN
    Ø,Q-1 AT Ø: P=Q: RETURN
21Ø IF P=Q THEN RETURN : IF Q#S4 — Q<34
     THEN VLIN Q+6,39 AT Ø: P=Q:
     RETURN
22Ø Q=(Y-5)/3+ RND (3)* SGN (W)
    *(X<1Ø AND V<Ø): IF Q<Ø THEN
    Q=Ø: GOTO 2Ø5
4ØØ FOR I=1 TO 8Ø: Q= PEEK (-16336
    ): NEXT I: GOTO 4Ø
```

PONG WITH BRICKS

```
  Ø TEXT : CALL -936: VTAB V: TAB
    6: PRINT "APPLE PONG WITH BRICKS
    ": PRINT : PRINT "EXAMPLE OF HOW
     TO WRITE YOUR OWN GAME"
  2 PRINT "+2PTS FOR BRICK, -1PT FOR
     MISS": FOR N=1 TO 7ØØØ: NEXT
    N: GOTO 5ØØ
  5 PAD=Ø
 1Ø NP= PDL (PAD)*34/256: IF NP=
    P(PAD) THEN 3Ø
 2Ø COLOR=3: VLIN NP,NP+6 AT PAD*
    39: COLOR=Ø: IF NP<P(PAD) THEN
    VLIN NP+6,39 AT PAD*39: IF
    NP>P(PAD) THEN VLIN Ø,NP-1 AT
    PAD*39:P(PAD)=NP
 3Ø PAD=PAD+1: IF PAD<2 THEN 1Ø
    : IF F THEN 53Ø
 4Ø NX=X+XV:NY=Y+YV: IF NX<Ø OR
    NX>39 THEN 4ØØ: IF NY<3 OR
    NY>116 THEN 1ØØ: IF SCRN(NX,
    NY/3)#Ø THEN 2ØØ
 5Ø COLOR=Ø: PLOT X,Y/3: COLOR=
    15: PLOT NX,NY/3:X=NX:Y=NY:
     GOTO 5
1ØØ YV=-YV:NY=Y: FOR N=1 TO 6:N1=
     PEEK (-16336): NEXT N: GOTO
    5Ø
2PØ XV=-XV: IF SCRN(NX,NY/3)#3 THEN
    3ØØ:YV=((NY/3)-P(NX>=39))-3
    :NX=X: FOR N=1 TO 5:N1= PEEK
    (-16336): NEXT N: GOTO 5Ø
3ØØ COLOR=Ø: PLOT NX,NY/3:PL=(XV<
@   Ø):SC(PL)=SC(PL)+1: VTAB 22
    : TAB 1Ø+2Ø*( NOT PL): PRINT
    SC(PL);:N= PEEK (-16336)+ PEEK
    (-16336)+ PEEK (-16336): GOTO
    5Ø
4ØØ PL=(NX>39):SC(PL)=SC(PL)+1:
     VTAB 22: TAB 1Ø+2Ø*( NOT PL)
    : PRINT SC(PL);
41Ø FOR N=1 TO 15:N1= PEEK (-16336
    )- PEEK (-16336): NEXT N: COLOR=
    Ø: PLOT X,Y/3: IF SC(PL)>14
     THEN 6ØØ: GOTO 52Ø
5ØØ GR : PRINT : PRINT : PRINT
    : PRINT : DIM SC(1),P(1):SC(
    Ø)=Ø:SC(1)=Ø:P(Ø)=Ø:P(1)=Ø
51Ø COLOR=13: FOR J=16 TO 24 STEP
    2: FOR K=16 TO 24 STEP 2: IF
    NOT (J MOD 2) AND J MOD 4=K MOD
    4 THEN PLOT J,K: NEXT K,J:XV=
    (2* RND (2))-1:Y= RND (2Ø)+
    1Ø
515 COLOR=8: HLIN Ø,39 AT Ø: HLIN
    Ø,39 AT 39
52Ø F=1: FOR N=1 TO 5Ø: GOTO 5
53Ø NEXT NZF=Ø:X=2Ø:YV= RND (7)
    -3: GOTO 5
6ØØ VTAB 22: TAB 7+(2Ø* NOT PL)
    : PRINT "WINNER";: END
```