**west side electronics**

P.O. BOX 636, CHATSWORTH, CALIFORNIA 91311 (213) 884-4794

# SUPERCLOCK II™

## OPERATING MANUAL

Revised 3/82.

Copyright 1981, 1982 West Side Electronics, Inc.

SUPERCLOCK II, SUPER-DOS, and TIME-CLOCK II are trademarks of West Side Electronics, Inc.
Apple, Apple II, and Applesoft are trademarks of Apple Computer, Inc.
Apple Clock is a trademark of Mountain Computer, Inc.
CP/M is a trademark of Digital Research, Inc.

All programs Copyright 1980,81,82 West Side Electronics, Inc.

```
*************************************************
*                                               *
*                     NOTE                      *
*                                               *
*     Before going  any  further, please make   *
*     copies of any disks that came with your   *
*     SUPERCLOCK II. Then store the originals   *
*     in a safe place.                          *
*                                               *
*                                               *
*************************************************
```

## INTRODUCTION

The SUPERCLOCK II represents an exciting addition to your Apple computer. Besides its usual function of supplying the date and time, the SUPERCLOCK II can also add new dimensions to the Apple by generating precise interrupts for foreground/background programming. Automatic date stamping of files in both BASIC and Pascal is also possible with the included software. The entire SUPERCLOCK II system will be described in this manual. Some portions may not be applicable if you did not purchase the optional software.

The following items should have been included in this package:

1 SUPERCLOCK II board
1 SUPERCLOCK SYSTEM diskette (SPC-D102)
1 Operating Manual

The SUPERCLOCK SYSTEM diskette contains utility and demonstration programs for using the clock in DOS 3.2, 3.3, Pascal, and CP/M. The front side is formatted as 16 sector and contains both Pascal and DOS 3.3 programs; the back side is 13 sector for DOS 3.2 operation and also serves as a backup copy. Please note that neither DOS 3.3 nor the Pascal files necessary for booting are present on this disk. Therefore, DO NOT ATTEMPT TO BOOT THE FRONT SIDE. The back side of the disk has the BASIC programs formatted in DOS 3.2.1. This side contains DOS and can be booted. Copy the appropriate side(s) of this diskette onto one or more blank disks and then prepare the copies as follows.

## PREPARING A 3.2.1 SUPERCLOCK II SYSTEM MASTER

DOS 3.2.1 users should make a copy of the back side of the SUPERCLOCK SYSTEM diskette. Then transfer the program UPDATE 3.2.1 supplied by Apple onto this copy.

## PREPARING A 3.3 SUPERCLOCK II SYSTEM MASTER

1. Make a copy of your DOS 3.3 System Master supplied by Apple and then boot it. 2. Insert the SUPERCLOCK II disk (SPC-D102) and type LOAD INSTALL (if you do not have Integer basic, type LOAD INSTALL.FP ). 3. Remove the SUPERCLOCK disk and replace it

with the copy of the DOS 3.3 Master. Type SAVE INSTALL and after it is finished type RUN (if you get a DISK FULL error, just UNLOCK and DELETE a large file such as PHONE LIST). 4. The INSTALL program is self-prompting and uses Apple's Master Create utility. See below for a complete description of the options available. After entering the required information (type HELLO for the greeting program's name), SUPER-DOS will be installed on your System Master. 5. When completed, hit ESC and then RETURN to re-boot the System Master, now with SUPER-DOS. 6. You can now place SUPER-DOS on any of your existing 3.3 DOS diskettes by BRUNning Master Create on this disk. One of the first disks you may wish to update is a copy of the SUPERCLOCK SYSTEM diskette since it has no DOS. The greeting program on this disk also called HELLO.

## INSTALL OPTIONS

SUPERCLOCK II SLOT - Must be set to the proper slot in which the clock is located (I-6). Default=4.

SUPERCLOCK II MODE - Must be set to the same mode as the clock (N=normal;A=ACE). Default=N.

TIME AND DATE - Specifies whether the time as well as the date should be stamped on each file. Default=N.

LANGUAGE CARD FIX - This is a fix for 3.3 DOS so that the Language Card need not be reloaded every time you boot. Default=Y.

## PASCAL UTILITY DISK

Make another copy of the SUPECLOCK SYSTEM diskette and label it WSE: (you can use any copy program, either from Pascal or DOS). This WSE: Pascal Utility Disk will contain the following files that allow you to use the SUPERCLOCK II in the Apple Pascal Operating environment:

WSE:

SYSTEM.LIBRARY - Complete library routines including UNIT

SUPERCLOCK. This library must be on the boot disk when compiling programs that use the clock.

AUTOBOOT.TEXT - Pascal program for automatic greeting and update of Filer.

SYSTEM.STARTUP - Compiled codefile for above program renamed to SYSTEM.STARTUP for immediate execution when booting.

SAMPLE1.TEXT - Short Pascal program to display time.

SAMPLE1.CODE - Compiled codefile for above.

LIBRARY/1.0 - Library for Pascal 1.0 users.

We suggest that you transfer WSE:SYSTEM.LIBRARY and SYSTEM.STARTUP onto a copy of APPLE1: and use this disk for booting. Apple Pascal 1.0 users should transfer WSE:LIBRARY/1.0 to APPLE1:SYSTEM.LIBRARY and then re-compile AUTOBOOT.TEXT into SYSTEM.STARTUP.

## INSTALLATION

Before installing the SUPERCLOCK II into your computer, examine the board to become familiar with the function of the dipswitches.

| SWITCH | ON (up) | OFF (down) |
|---|---|---|
| SET | Clock setting enabled | Setting disabled |
| ADJ | Resets seconds to 00 | Normal position |
| MODE | Selects ACE mode | Normal mode |

The last switch is unused and the normal position for all switches is down or OFF.

After turning off the computer, remove the cover by gently pulling up at the back. The SUPERCLOCK II can now be placed in any slot except 0; slot 4 is recommended. Do not replace the cover until the clock has been set.

## SETTING THE CLOCK

To set the clock, the switch marked "SET" must be moved to the ON position. Now run the clock setting program supplied. Follow the directions in the program to set the clock to the correct local time. Note that all data to and from the clock is expressed numerically. In particular, the day of week is represented by a number from 0-6 as shown below:

0 - Sunday
1 - Monday
2 - Tuesday
3 - Wednesday
4 - Thursday
5 - Friday
6 - Saturday

After setting the clock, return the "SET" switch to the OFF position to prevent accidental changes to the clock.

## READING THE CLOCK

The SUPERCLOCK II contains firmware that makes reading the clock from BASIC extremely easy. In short, whenever the SUPERCLOCK's slot is selected for input (i.e. IN#n in BASIC, n CTRL-K from the monitor, etc.), it will return a string of characters containing the date and time in this format (assuming NORMAL mode):

```
                  W MM/DD/YY HH;MM;SS
                  ↑  \      /  \     /
   (day of week┘   date      time   )
```

Semicolons are used in the time display because Applesoft cannot handle colons on input. Note that when reading the clock, the computer will echo the input (i.e. print it on the screen) just as if it were being entered from the keyboard. If you want to read the clock without having it echo to the screen, use the command PR#n. This sends all output to the SUPERCLOCK where it is "dumped." Don't forget to reset normal keyboard input and screen output with IN#0 and PR#0 after reading the clock. Of course, when using DOS, these commands must be preceded with a CTRL-D and placed in a PRINT statement.

When inputting the clock data into a string, both IN#n and PR#n should be used. Then read the clock with a statement in the form:

```
INPUT " ",T$    (Integer BASIC)

INPUT " ";T$    (Applesoft)
```

Note that a space is first printed to the clock board. This will ensure compatibility between Integer BASIC and Applesoft and also between other clock boards. Various components of the time can then be extracted using the appropriate string functions (see Integer BASIC and Applesoft manuals).

| NORMAL MODE | INTEGER | APPLESOFT |
|---|---|---|
| | WEEKDAY$=T$(1,1) | WEEKDAY$=LEFT$(T$,1) |
| | MONTH$=T$(3,4) | MONTH$=MID$(T$,3,2) |
| | DAY$=T$(6,7) | DAY$=MID$(T$,6,2) |
| | YEAR$=T$(9,10) | YEAR$=MID$(T$,9,2) |
| | HOUR$=T$(12,13) | HOUR$=MID$(T$,12,2) |
| | MINUTE$=T$(15,16) | MINUTE$=MID$(T$,15,2) |
| | SECOND$=T$(18,19) | SECOND$=RIGHT$(T$,2) |

Study the sample programs supplied for further clarification on how to use the SUPERCLOCK II.

## APPLE CLOCK EMULATION (ACE) MODE

To take advantage of the many programs written for the Apple Clock from Mountain Hardware, the SUPERCLOCK II has incorporated an Apple Clock Emulation mode. This mode is selected by moving the switch marked "MODE" to the ON position. The major difference between modes is in the format of the data presentation from the clock. The Apple Clock uses the following format:

```
                MM/DD HH;MM;SS.mmm
                \   /  \    /  \/
              ( date    time   milliseconds   )
```

In the ACE mode, the SUPERCLOCK II uses this format:

```
                MM/DD HH;MM;SS.WYY
                \   /  \    / ↑\/
              ( date    time  | └year          )
                          day of week
```

Thus the only difference between the two clocks is that the day of week and year digits appear instead of milliseconds. In this mode, the various components of the string can be extracted as shown below.

| ACE MODE | INTEGER | APPLESOFT |
|---|---|---|
| | WEEKDAY$=T$(16,16) | WEEKDAY$=MIO$(T$,16,1) |
| | MONTH$=T$(1,2) | MONTH$=LEFT$(T$,2) |
| | DAY$=T$(4,5) | DAY$=MID$(T$,4,2) |
| | YEAR$=T$(17) | YEAR$=RIGHT$(T$,2) |
| | HOUR$=T$(7,8) | HOUR$=MID$(T$,7,2) |
| | MINUTE$=T$(10,11) | MINUTE$=MID$(T$,10,2) |
| | SECOND$=T$(13,14) | SECOND$=MID$(T4,13,2) |

It is also possible to convert the string representation of the data from one mode to another. For example, the following line will allow an Applesoft program written for the Apple Clock to be used with a SUPERCLOCK II in either mode:

```
  IF MID$(T$,8,1)="/" THEN T$=MID$(T$,3,5)+MID$(T$,11,9)+".000"
```

This line should be added just after the portion of the program that reads the clock. If the program also has an Apple Clock slot-finding routine, it may not work when the SUPERCLOCK II is in the NORMAL mode. Therefore, you should replace it with the SUPERCLOCK II slot-finder.

## SUPERCLOCK II SLOT FINDER

The subroutine at the end of the clock setting programs can be used to automatically determine in which slot your SUPERCLOCK II is located. To allow compatibility with the Apple Clock and to promote consistency in programs that use the SUPERCLOCK II, the following bytes in the PROM firmware should be used by slot finding routines:

| TO FIND | BYTE | IS | AND | BYTE | IS |
|---|---|---|---|---|---|
| Clock in ACE mode* | 13 | 2C | | 15 | 68 |
| Clock in NORMAL mode | 13 | D7 | | 15 | C5 |
| Clock in either mode | 22 | F8 | | 24 | 68 |

*or Apple Clock

## ACCURACY AND 30 SECOND ADJUSTMENT

Your SUPERCLOCK II was adjusted at the factory to provide accuracy within a couple of minutes per year. If the clock is consistently gaining or losing time, you may wish to adjust the trimmer "C5" at the upper right corner of the board. Using a small screwdriver, turn the trimmer VERY SLIGHTLY clockwise to speed up the clock, counter-clockwise to slow it down.

Additionally, the SUPERCLOCK II can be synchronized to the correct time by using the switch marked "ADJ." Moving this switch to the ON position will momentarily reset the seconds to 00, adding one minute if the seconds were greater than 30. Return this switch to the OFF position after adjusting the clock.

## ON-BOARD BATTERY

The SUPERCLOCK II contains a NiCd rechargeable battery to maintain timekeeping when the computer is off. This battery is automatically charged whenever the computer is on and under normal use will not require attention. If you remove the SUPERCLOCK from the computer or leave it off for more than three months at a time, then the battery may require a full recharging. This can be accomplished by leaving the computer turned on for 24-36 hours.

## READING THE CLOCK FROM MACHINE LANGUAGE

This section is for those wishing to incorporate the SUPERCLOCK II into their machinge language programs. The simplest way to do this is by accessing the PROM firmware on the clock. For example, the program below will read the data from the clock into a string of RAM locations:

```
1000-   A5 38       LDA   $38
1002-   48          PHA
1003-   A5 39       LDA   $39
1005-   48          PHA
1006-   A9 00       LDA   #$00
1008-   85 38       STA   $38
100A-   A9 C4       LDA   #$C4
100C-   85 39       STA   $39
100E-   A2 13       LDX   #$13
1010-   20 18 FD    JSR   $FD18
1013-   9D 81 02    STA   $0281,X
1016-   CA          DEX
1017-   10 F7       BPL   $1010
1019-   68          PLA
101A-   85 39       STA   $39
101C-   68          PLA
101D-   85 38       STA   $38
101F-   60          RTS
```

Note that this routine assumes the clock to be in the ACE mode. In fact, the location of the RAM buffer was chosen to match the format used by the Apple Clock. Thus you may find this routine helpful in interfacing the SUPERCLOCK II to machine language programs written for the other clocks. When your program must handle the clock in either mode, it is usually best to write a complete driver routine, independent of the firmware. An example of this can be found in the VISIO.8 file on the SUPERCLOCK SYSTEM disk (it loads at $4C00; clock entry to read time is $4C06 and interrupt control is done through $4C03).

## INTERRUPTS

Refer to the data sheet of the 6820/6821 for complete details on interrupt handling. The following interrrupt frequencies are available:

| | | | |
|---|---|---|---|
| CA1 | - | 1024 Hz | (approx. 1 per mS) |
| CA2 | - | 1 Hz | (1 per second) |
| CB1 | - | 1/60 Hz | (1 per minute) |
| CB2 | - | 1/3600 Hz | (1 per hour) |

PIA addresses can be determined as follows:

| | | | |
|---|---|---|---|
| PORT A | - | $C080 + $n0 | |
| CRA | - | $C081 + $n0 | (n=SUPERCLOCK slot) |
| PORT B | - | $C082 + $n0 | |
| CRB | - | $C083 + $n0 | |

A simple example of interrupt handling may be found in our millisecond timing routine, MSRTN. The source code for this program is on the next page. Refer to it as you read the following description of the important sections.

Lines 270-590 comprise a slot-finding routine. Then lines 600-630 set up the Apple's IRQ interrupt vector at location $3FE-3FF. The address of your interrupt handling routine should be placed here, low byte first. Next we initialize the SUPERCLOCK's PIA in lines 640-780. The important points here are: 1) set up the Data Direction Registers, 2) enable the PIA to pass the appropriate interrupt signals, 3) enable the actual signals from the clock, and then, 4) clear the PIA interrupt flag(s). This last function is accomplished by performing a "dummy" read of the port associated with the given interrupt signal. Finally, the CLI instruction is executed and interrupts are enabled.

When an interrupt occurs, the CPU will begin executing your interrupt routine as pointed to by $3FE-3FF. This routine should always restore the accumulator from location $45 and perform another "dummy" read to clear the interrupt flag just prior to returning via the RTI instruction. Of course, any other registers used should be saved at the beginning, and then restored at the end, of your routine.

```
              0010 ;*******************************
              0020 ;*                             *
              0030 ;*  MSRTN :   SUPERCLOCK I[    *
              0040 ;*            MILLISECOND      *
              0050 ;*            ROUTINE          *
              0060 ;*                             *
              0070 ;*  BY JEFF MAZUR   11/10/80   *
              0080 ;*                             *
              0090 ;*******************************
              0100 ;
              0110 ;
              0120 TEMPL  EQU 00
              0130 TEMPH  EQU 01
              0140 COUNTL EQU 3CF
              0150 COUNTH EQU 3CE
              0160 IRQLOC EQU 3FE
              0170 ACC    EQU 45
              0180 ;
              0190 PORTA  EQU C080      ;BASE ADDRESS
              0200 CRA    EQU C081      ;+ SLOT OFFSET
              0210 PORTB  EQU C082
              0220 CRB    EQU C083
              0230 ;
              0240        ORG 280
              0250        OBJ 880
              0260 ;
0280  A501    0270 START  LDA TEMPH     ;SAVE TEMP
0282  48      0280        PHA
0283  A500    0290        LDA TEMPL
0285  48      0300        PHA
0286  A922    0310        LDA #$22      ;CHECK PROM
0288  8500    0320        STA TEMPL
028A  A2C7    0330        LDX #$C7      ;START @ SLOT7
028C  8601    0340 SEARCH STX TEMPH
028E  A002    0350        LDY #$2       ;3 BYTES
0290  B100    0360 LOOP   LDA (TEMPL),Y
0292  D9E602  0370        CMP REF,Y
0295  D005    0380        BNE NOMTCH
0297  88      0390        DEY
0298  10F6    0400        BPL LOOP
029A  300C    0410        BMI FOUND
029C  CA      0420 NOMTCH DEX
029D  E0C0    0430        CPX #$C0      ;SKIP SLOT 0
029F  D0EB    0440        BNE SEARCH
02A1  68      0450 RETURN PLA
02A2  8501    0460        STA TEMPH
02A4  68      0470        PLA
02A5  8500    0480        STA TEMPL
02A7  60      0490        RTS
02A8  78      0500 FOUND  SEI
02A9  8A      0510        TXA
02AA  0A      0520        ASL
02AB  0A      0530        ASL
02AC  0A      0540        ASL
02AD  0A      0550        ASL
02AE  A8      0560        TAY
02AF  18      0570        CLC
02B0  6980    0580        ADC #$80
02B2  8D0203  0590        STA MSRTN+02 ;MODIFY MSRTN
02B5  A903    0600        LDA #$03     ;<MSRTN
02B7  8DFF03  0610        STA IRQLOC+01
02BA  A900    0620        LDA #$00     ;>MSRTN
02BC  8DFE03  0630        STA IRQLOC
02BF  A900    0640 INIPIA LDA #$00     ;SELECT DDR
02C1  9981C0  0650        STA CRA,Y
02C4  9983C0  0660        STA CRB,Y
02C7  A9F0    0670        LDA #$F0     ;PORTA=INPUT
02C9  9980C0  0680        STA PORTA,Y
02CC  A9FF    0690        LDA #$FF     ;PORTB=OUTPUT
02CE  9982C0  0700        STA PORTB,Y
02D1  A905    0710        LDA #$05     ;SELECT MS
02D3  9981C0  0720        STA CRA,Y
02D6  A904    0730        LDA #$04     ;AND PORTS
02D8  9983C0  0740        STA CRB,Y
02DB  A92F    0750        LDA #$2F     ;ENABLE CLOCK
02DD  9982C0  0760        STA PORTB,Y
02E0  B980C0  0770        LDA PORTA,Y
02E3  4CA102  0780        JMP RETURN
              0790 ;
02E6  F80568  0800 REF    DFD F80568
              0810 ;
              0820 ;
              0830        ORG 300
              0840        OBJ 900
              0850 ;
0300  08      0860 MSRTN  PHP
0301  AD80C0  0870        LDA PORTA    ;CLR PIA FLAG
0304  A545    0880        LDA ACC      ;RESTORE ACC
0306  EECF03  0890        INC COUNTL   ;INC COUNTER
0309  D003    0900        BNE DONE
030B  EECE03  0910        INC COUNTH
030E  28      0920 DONE   PLP
030F  40      0930        RTI
              0940 ;
0310  00      0950        DFD 00
              0960 ;
0311  A900    0970 RESET  LDA #$00
0313  8DCF03  0980        STA COUNTL
0316  8DCE03  0990        STA COUNTH
0319  58      1000        CLI
031A  60      1010        RTS
              1020 ;
031B  78      1030 STOP   SEI
031C  60      1040        RTS
```

## SUPERCLOCK II AND PASCAL

Using the SUPERCLOCK II with Apple Pascal is greatly simplified by having the clock routines in the system library. The file NSE:SYSTEM.LIBRARY contains all of the normal routines plus the addition of UNIT SUPERCLOCK. You can give your programs access to the clock by simply adding the following statement just after the PROGRAM heading:

```
USES SUPERCLOCK;
```

This line effectively adds 3 external procedures which are described below. The following global variables are also declared:

```
DOW,MON,DAY,YR,HR,MIN,SEC   : INTEGER
TIME,DISKNAME,LASTBOOT      : STRING
CLOCKPRESENT                : BOOLEAN
```

### PROCEDURE READCLOCK;

This procedure will read the SUPERCLOCK II and assign to each variable data corresponding to the current date and time.

DOW - Day of week (0..6)
MON - Month (1..12)
DAY - Day of month (1..31)
YR - Year (0..99)
HR - Hour (0..23)
MIN - Minutes (0..59)
SEC - Seconds (0..59)

This data will also be formatted into a string by a call to the procedure TIMESTRING (see below).

### PROCEDURE TIMESTRING;

This procedure takes the data stored in the variables MON..SEC and creates a string in the format

```
MM/DD/YY HH:MM:SS
```

This string is then assigned to the variable TIME. Note that the day of week information is not used. This procedure can also be used to convert any integer data into a string; just set up the variables MON..SEC as desired before calling TIMESTRING.

PROCEDURE UPDATE (vol);

This procedure expects to be passed one parameter of type INTEGER. If the value of this parameter is a volume number for a disk drive (i.e.4,5,9,10,11,12), then the corresponding disk will be updated, if possible, with the current date and time. The volume name of the disk will also be read and assigned to the variable DISKNAME. Similarly, the date and time of last boot (or whenever the disk was last UPDATEd) is stored in the string LASTBOOT. Finally, the Filer is updated to reflect the current date for saving files.

Note that the seconds digits are not stored on the disk and that when UPDATE reads a time of last boot, it will show the seconds equal to 00. Also note that whenever the Filer writes to the disk, the time of last boot stored on the disk will be set to 00:00:00.

## SUPERCLOCK II AND CP/M

The SUPERCLOCK II may be used under the CP/M operating system with the appropriate software. From MBASIC or GBASIC, this is most easily done by POKEing in a short interface routine. Two examples of this are provided in the files CPMDEMO1 and 2. These are normal Apple text files which can be converted to the CP/M system by use of the APDOS utiity (see the CP/M documentation).

## DESCRIPTION OF FILES ON THE SUPERCLOCK II DISK (SPC-D102)

CLOCK FACE - Binary file containing the hi-res picture of a clock used by the HI-RES program. Do not BRUN.
CPMDEMOs - Text files of sample programs showing how to use the clock from CP/M Basic.
DATE & TIME - Demonstration of clock reading techniques.
HI-RES CLOCK - This is an Applesoft program that uses hi-res graphics to display a clock with moving hands.
INTERRUPT DEMO - Program that explains and executes STATUS.
MS DEMOs - Demonstrations of millisecond timing.
MSRTN - The actual millisecond routine used by above.
QUIZ DEMO - Shows how to put a time limit on input.
SET CLOCK - BASIC program to set clock for correct time.
SLOT-FINDER - Text file of SUPERCLOCK slot finding routine. Can be added to any Integer or Applesoft program by simply typing EXEC SLOT-FINDER.
STATUS - Routine that places time and date at the top of the screen using interrupts. Can be BRUN.
TIMEFILE - Text file used by TIMEFILER program.
TIMEFILER - Keeps track of when disk was last booted.
VISIPATCH - Program to modify VisiDex (from VisiCorp).

# ** SUPER-DOS **

NOTE: All RAM addresses given in this section are for a 48K (or Language System) Apple II or Apple II Plus. For 32K machines, subtract $4000 (16384 decimal); 16K machines subtract $8000 (32768 decimal).

## INTRODUCTION

SUPER-DOS, in conjunction with the SUPERCLOCK II , adds an exciting new dimension to Apple's Disk Operating System (DOS 3.2.1. or 3.3) - automatic time/date stamping of files as they are stored on the disk. This feature is similar to that found on most large computer systems and the Apple Pascal Operating System. Because we have incorporated this feature directly into the normal Apple DOS, there are no special instructions necessary for using SUPER-DOS. In fact, you probably will not be aware that you're using SUPER-DOS until you type CATALOG.

## USING SUPER-DOS

If you haven't already done so, boot your SUPERCLOCK II System Master (see page 1 for how this disk is made). Now practice saving and loading files under SUPER-DOS. Note how the current date (and time) is automatically stored in the catalog and that when accessing files, you do not need to type in the date as part of the filename. The only new restriction is that filenames are now limited to 21 characters (slightly less when time is also added) as opposed to the normal 30 - not much of a sacrifice.

After you are acquainted with the operation of SUPER-DOS, you can then use the MASTER CREATE (3.3) or UPDATE 3.2.1 program from Apple to transfer this new DOS to your existing diskettes. Note that only the DOS is changed; any programs on the disk will not be harmed. However, as with any software product, it is always a good idea to make a back-up copy of a disk before updating it.

## COMPATIBILITY WITH REGULAR DOS

After updating a regular DOS diskette, all programs on that disk will remain intact and there will be no problem using them with SUPER-DOS. Of course, there won't be a date associated with the exising files (created before the use of SUPER-DOS), but new files stored on the disk will automatically be dated. If a SAVE, BSAVE, RENAME, or OPEN command is performed on any exising file (whether it previously had a date or not), the current date will be stored in the catalog by SUPER-DOS. The INIT command will also cause the booting (or "HELLO") program to be saved with the date that the disk was initialized.

It is important to note that all files saved under SUPER-DOS are directly downward-compatible with regular DOS. If you ever happen to be running without SUPER-DOS and wish to access a dated file, you can either use the normal cursor copying function to include the date in the filename; or a simpler approach is to type:

```
POKE -19965,21
```

|  |  |  |
|---|---|---|
| For time and date | POKE -19965,15 | (Normal) |
|  | POKE -19965,18 | (ACE) |

This will make regular DOS ignore the date in all subsequent commands (i.e. until another regular DOS diskette is booted). This POKE is worth remembering if you don't plan on updating all of your diskettes.

You may also wish to know that if SUPER-DOS does not find a SUPERCLOCK II in the expected slot and mode, it will continue to function except that whenever it tries to store a date, it will just put in blanks. This could remove existing dates from files unless the updating feature is disabled (see next section). If you ever must remove the clock or change its defaults, either notify SUPER-DOS (with the appropriate POKEs) or boot up with a regular DOS diskette.

## FURTHER OPERATING HINTS

When transferring files between diskettes, it is best to use a copy program with a wildcard specification feature (such as Apple's FID or FISHHEAD). This will allow easy selection of the files and will assure that the original date of each file is left intact. LOADing and then SAVEing would of course transfer the file with the current date instead.

When using text files, any time the file is OPENed it will be updated. If you wish to defeat this feature (eg. when READing a text file), use the following commands:

|  | DOS 3.2.1 | DOS 3.3 |
|---|---|---|
| DISABLE UPDATING | POKE -17166,208 | POKE -18709,208 |
| RESTORE UPDATING | POKE -17166,144 | POKE -18709,144 |

If you wish to disable dating of new files, type:

|  | DOS 3.2.1 | DOS 3.3 |
|---|---|---|
| TURN OFF | POKE -17218,0 | POKE -18761,0 |
| TURN ON | POKE -17218,215 | POKE -18761,215 |

## OPTIONAL TIME-CLOCK II INSTRUCTIONS

NOTE: This disk is on 3.2 SUPER-DOS. The programs herein may be MUFFINed to 3.3 (preferably SUPER-DOS).

FIRST TIME USE. When you first boot the TIME-CLOCK II diskette, you will be presented with a blank menu of jobs. At this point use the A)dd command (just press the 'A' key) to add one or more jobs. For each job you will be asked for a job number (0-999), client name (0-13 characters), and the name of the program used for running this job. For example, if you used your Apple PIE word processing program to write contracts for ABC, Inc., you might enter:

```
JOB NUMBER: 100
CLIENT NAME: ABC, INC.
PROGRAM NAME: APPLE PIE
```

If the program you wish to run is on a protected disk or is otherwise incapable of being executed by a RUN or BRUN, you can type PR#6 (or other appropriate slot) for the program name. This will have the effect of booting the application program diskette.

RUNNING A PROGRAM. From the TIME-CLOCK II program, typing 'R' for Run followed by a job number, will automatically log on the desired job and, if the required program is on the same diskette, it will begin executing. Otherwise you will be prompted to place the appropriate application disk in the drive and hit RETURN. When you are finished with this job, you must replace the TIME-CLOCK II disk back in the drive (if it was removed) and RUN LOGOFF or just boot the disk. Application programs written in BASIC can usually be modified to automatically run LOGOFF when they are finished. Look through the program listing for where the program normally ends (usually with an END statement). Replace this with the following statement:

```
PRINT "dRUN LOGOFF"  (d=CTRL-D)
```

LOGOFF. This program calculates how much time you've spent on each session and records this information to the disk. You will have the option to change the STOP, and thus the ELAPSED, times if they are incorrect (eg. you took a 5 minute break while on the phone).

TIMER. This is a general purpose elapsed timer program that can be used with the TIME-CLOCK II program to record time spent on activities other than the computer. For example, a lawyer wishing to keep track of office visits could define a job number with the client's name, and then enter TIMER for the program name.

```
]LIST

10  REM   ** SUPERCLOCK II DATE & TIME **
20  HOME
30  GOSUB 30000
40  PRINT "       YOUR SUPERCLOCK IS IN SLOT ";SLOT
50 PA =  - 16256 + SLOT * 16
60 PB = PA + 2
65 CA = PA + 1:CB = PB + 1
70  INVERSE : VTAB 23: PRINT "          PRESS ANY KEY TO STOP          ";
    : NORMAL
80  VTAB 21: PRINT
90  PRINT " IN#";SLOT: REM  CTRL-D IN QUOTES FOR DOS
100  PRINT " PR#";SLOT
110  INPUT A$
120  PRINT " PR#0"
130  PRINT " IN#0"
140  IF  MID$ (A$,8,1) = "/" THEN 160
150 A$ =  MID$ (A$,16,1) + " " +  LEFT$ (A$,5) + "/" +  RIGHT$ (A$,2) +
     MID$ (A$,6,9): REM  CONVERT FROM ACE MODE IF NECESSARY
160  RESTORE : VTAB 6
200  REM   ** DAY OF WEEK ROUTINE **
210  FOR I = 1 TO 7
220  READ DUMMY$
230  IF I =  VAL ( LEFT$ (A$,1)) + 1 THEN DW$ = DUMMY$
240  NEXT I
250  DATA SUNDAY,MONDAY,TUESDAY
260  DATA WEDNESDAY,THURSDAY,FRIDAY,SATURDAY
270  REM
300  REM   ** MONTH ROUTINE **
310  FOR I = 1 TO  VAL ( MID$ (A$,3,2))
320  READ MO$
330  NEXT I
340  DATA  JANUARY,FEBRUARY,MARCH
350  DATA APRIL,MAY,JUNE
360  DATA  JULY,AUGUST,SEPTEMBER
370  DATA OCTOBER,NOVEMBER,DECEMBER
380  REM
400  REM   ** AM/PM ROUTINE **
410 P$ = ""
415 TA =  PEEK (CA):TB =  PEEK (CB): POKE CA,4: POKE CB,4
420  POKE PB,53
430 PM =  PEEK (PA)
440  POKE PB,47
445  POKE CA,TA: POKE CB,TB:PM = PM -  INT (PM / 16) * 16
450  IF PM > 7 THEN 480
460 P$ = " AM"
470  IF PM > 3 THEN P$ = " PM"
480  REM
500  PRINT : PRINT  SPC( 20 -  LEN (DW$) / 2);DW$;"     "
510 D$ = MO$ + " " +  MID$ (A$,6,2) + ", 19" +  MID$ (A$,9,2)
520  PRINT : PRINT  SPC( 20 -  LEN (D$) / 2);D$;"       "
530 TIME$ =  MID$ (A$,12,2) + ":" +  MID$ (A$,15,2) + ":" +  MID$ (A$,18
    ,2) + P$
540  PRINT : PRINT  SPC( 20 -  LEN (TIME$) / 2);TIME$
550  IF  PEEK ( - 16384) < 128 THEN 80
560  POKE  - 16368,0
570  END
30000  REM   SUPERCLOCK II FINDER
30010 I =  PEEK ( - 12289): REM   KILL ALL ROMS
30020 SLOT = 0:I = 1
30030  IF  PEEK ( - 16350 + I * 256) = 248 AND  PEEK ( - 16348 + I * 256
    ) = 104 THEN 30060
30040 I = I + 1: IF I < 8 THEN 30030
30050  PRINT "I CANNOT FIND A SUPERCLOCK II": GOTO 30070
30060 SLOT = I
30070 I =  PEEK ( - 12289): RETURN
```

## INTERNAL CONTROLS

There are six locations within the PIA accessible to the MPU data bus: two Peripheral Registers, two Data Direction Registers, and two Control Registers. Selection of these locations is controlled by the RS0 and RS1 inputs together with bit 2 in the Control Register, as shown in Table 1.

TABLE 1 – INTERNAL ADDRESSING

| | | Control Register Bit | | |
|---|---|---|---|---|
| RS1 | RS0 | CRA-2 | CRB-2 | Location Selected |
| 0 | 0 | 1 | X | Peripheral Register A |
| 0 | 0 | 0 | X | Data Direction Register A |
| 0 | 1 | X | X | Control Register A |
| 1 | 0 | X | 1 | Peripheral Register B |
| 1 | 0 | X | 0 | Data Direction Register B |
| 1 | 1 | X | X | Control Register B |

X = Don't Care

### INITIALIZATION

A low reset line has the effect of zeroing all PIA registers. This will set PA0-PA7, PB0-PB7, CA2 and CB2 as inputs, and all interrupts disabled. The PIA must be configured during the restart program which follows the reset.

Details of possible configurations of the Data Direction and Control Register are as follows.

### DATA DIRECTION REGISTERS (DDRA and DDRB)

The two Data Direction Registers allow the MPU to control the direction of data through each corresponding peripheral data line. A Data Direction Register bit set at "0" configures the corresponding peripheral data line as an input, a "1" results in an output.

### CONTROL REGISTERS (CRA and CRB)

The two Control Registers (CRA and CRB) allow the MPU to control the operation of the four peripheral control lines CA1, CA2, CB1 and CB2. In addition they allow the MPU to enable the interrupt lines and monitor the status of the interrupt flags. Bits 0 through 5 of the two registers may be written or read by the MPU when the proper chip select and register select signals are applied. Bits 6 and 7 of the two registers are read only and are modified by external interrupts occurring on control lines CA1, CA2, CB1 or CB2. The format of the control words is shown in Table 2.

TABLE 2 – CONTROL WORD FORMAT

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRA | IRQA1 | IRQA2 | CA2 Control | | | DDRA Access | CA1 Control | |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRB | IRQB1 | IRQB2 | CB2 Control | | | DDRB Access | CB1 Control | |

**Data Direction Access Control Bit (CRA-2 and CRB-2)** – Bit 2 in each Control register (CRA and CRB) allows selection of either a Peripheral Interface Register or the Data Direction Register when the proper register select signals are applied to RS0 and RS1.

**Interrupt Flags (CRA-6, CRA-7, CRB-6, and CRB-7)** – The four interrupt flag bits are set by active transitions of signals on the four Interrupt and Peripheral Control lines when those lines are programmed to be inputs. These bits cannot be set directly from the MPU Data Bus and are reset indirectly by a Read Peripheral Data Operation on the appropriate section

TABLE 3 – CONTROL OF INTERRUPT INPUTS CA1 AND CB1

| CRA-1 (CRB-1) | CRA-0 (CRB-0) | Interrupt Input CA1 (CB1) | Interrupt Flag CRA-7 (CRB-7) | MPU Interrupt Request $\overline{\text{IRQA}}$ ($\overline{\text{IRQB}}$) |
|---|---|---|---|---|
| 0 | 0 | ↓ Active | Set high on ↓ of CA1 (CB1) | Disabled — $\overline{\text{IRQ}}$ remains high |
| 0 | 1 | ↓ Active | Set high on ↓ of CA1 (CB1) | Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high |
| 1 | 0 | ↑ Active | Set high on ↑ of CA1 (CB1) | Disabled — $\overline{\text{IRQ}}$ remains high |
| 1 | 1 | ↑ Active | Set high on ↑ of CA1 (CB1) | Goes low when the interrupt flag bit CRA-7 (CRB-7) goes high |

Notes
1. ↑ indicates positive transition (low to high)
2. ↓ indicates negative transition (high to low)
3. The Interrupt flag bit CRA-7 is cleared by an MPU Read of the A Data Register, and CRB-7 is cleared by an MPU Read of the B Data Register
4. If CRA-0 (CRB-0) is low when an interrupt occurs (Interrupt disabled) and is later brought high, $\overline{\text{IRQA}}$ ($\overline{\text{IRQB}}$) occurs after CRA-0 (CRB-0) is written to a "one".

**Control of CA1 and CB1 Interrupt Input Lines (CRA-0, CRB-0, CRA-1, and CRB-1)** – The two lowest order bits of the control registers are used to control the interrupt input lines CA1 and CB1 Bits CRA-0 and CRB-0 are used to enable the MPU interrupt signals $\overline{\text{IRQA}}$ and $\overline{\text{IRQB}}$, respectively. Bits CRA-1 and CRB-1 determine the active transition of the interrupt input signals CA1 and CB1 (Table 3).

TABLE 4 – CONTROL OF CA2 AND CB2 AS INTERRUPT INPUTS
CRA5 (CRB5) is low

| CRA-5 (CRB-5) | CRA-4 (CRB-4) | CRA-3 (CRB-3) | Interrupt Input CA2 (CB2) | Interrupt Flag CRA-6 (CRB-6) | MPU Interrupt Request $\overline{\text{IRQA}}$ ($\overline{\text{IRQB}}$) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | ↓ Active | Set high on ↓ of CA2 (CB2) | Disabled — $\overline{\text{IRQ}}$ remains high |
| 0 | 0 | 1 | ↓ Active | Set high on ↓ of CA2 (CB2) | Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high |
| 0 | 1 | 0 | ↑ Active | Set high on ↑ of CA2 (CB2) | Disabled — $\overline{\text{IRQ}}$ remains high |
| 0 | 1 | 1 | ↑ Active | Set high on ↑ of CA2 (CB2) | Goes low when the interrupt flag bit CRA-6 (CRB-6) goes high |

Notes: 1 ↑ indicates positive transition (low to high)

2 ↓ indicates negative transition (high to low)

3 The Interrupt flag bit CRA-6 is cleared by an MPU Read of the A Data Register and CRB-6 is cleared by an MPU Read of the B Data Register

4 If CRA-3 (CRB-3) is low when an interrupt occurs (Interrupt disabled) and is later brought high, $\overline{\text{IRQA}}$ ($\overline{\text{IRQB}}$) occurs after CRA-3 (CRB-3) is written to a "one".

TABLE 5 – CONTROL OF CB2 AS AN OUTPUT
CRB-5 is high

| CRB-5 | CRB-4 | CRB-3 | CB2 Cleared | CB2 Set |
|---|---|---|---|---|
| 1 | 0 | 0 | Low on the positive transition of the first E pulse following an MPU Write B Data Register operation | High when the interrupt flag bit CRB-7 is set by an active transition of the CB1 signal |
| 1 | 0 | 1 | Low on the positive transition of the first E pulse after an MPU Write "B" Data Register operation. | High on the positive edge of the first "E" pulse following an "E" pulse which occurred while the part was deselected |
| 1 | 1 | 0 | Low when CRB-3 goes low as a result of an MPU Write in Control Register B | Always low as long as CRB-3 is low. Will go high on an MPU Write in Control Register B that changes CRB-3 to one |
| 1 | 1 | 1 | Always high as long as CRB-3 is high. Will be cleared when an MPU Write Control Register B results in clearing CRB-3 to zero | High when CRB-3 goes high as a result of an MPU Write into Control Register "B". |

```
                ;****************
                ;*              *
                ;* SUPERCLOCK II *
                ;*   I/O  DRIVER *
                ;*              *
                ;*    VER 2.3   *
                ;*              *
                ;****************
                ;
                ; J MAZUR  9/14/81
                ;
                ;
                CSWL    EQU 0036
                CSWH    EQU 0037
                KSWL    EQU 0038
                KSWH    EQU 0039
                STACK   EQU 0100
                ASAVE   EQU 0478
                PSAVE   EQU 04F8
                YSAVE   EQU 0578
                XSAVE   EQU 05F8
                NOSAVE  EQU 0678
                TBLPTR  EQU 06F8
                SLOT    EQU 07F8
                PORTA   EQU C080
                CRA     EQU C081
                PORTB   EQU C082
                CRB     EQU C083
                IORTS   EQU FF58
                ;
                ;
                        ORG C400
C400  08                PHP            ;SAVE P
C401  78                SEI            ;DISABLE INT
C402  2C58FF            BIT IORTS      ;SET V FLAG
C405  5000              BVC ENTR       ;ALWAYS TAKEN
                        ORG C406       ;BYTE SAVER
C406  B8                CLV            ;RE-ENTRY
C407  48        ENTR    PHA
C408  8A                TXA
C409  48                PHA
C40A  2058FF            JSR IORTS
C40D  BA                TSX
C40E  BD0001            LDA STACK,X    ;$CN
C411  D003              BNE SKIP       ;SKIP OVER
```

```
C473  9981C0            STA CRA,Y
C476  9983C0            STA CRB,Y
C479  A92F              LDA #$2F
C47B  9982C0            STA PORTB,Y
C47E  A9E4      DONE    LDA #$E4       ;SET COUNTER
C480  9DF806            STA TBLPTR,X   ;TO TABLE-7
C483  A906              LDA #$06       ;SET KSW FOR
C485  8538              STA KSWL       ;RE-ENTRY
C487  BC7806            LDY NOSAVE,X
C48A  A910              LDA #$10       ;SET HOLD
C48C  9982C0            STA PORTB,Y
C48F  B8                CLV
C490  EA        READ    NOP            ;V2.3 REMOVED
C491  EA                NOP            ;CODE TO STOP
C492  EA                NOP            ;BLINKING
C493  EA                NOP            ;CURSOR
C494  EA                NOP
C495  EA                NOP
C496  EA                NOP
C497  EA                NOP
C498  FEF806            INC TBLPTR,X
C49B  BCF806            LDY TBLPTR,X
C49E  B138              LDA (KSWL),Y
C4A0  1026              BPL RDCLOK
C4A2  C9FF              CMP #$FF       ;DONE?
C4A4  D015              BNE EXIT       ;IF NOT, EXIT
C4A6  A900              LDA #$00       ;RESTORE KSW
C4A8  8538              STA KSWL
C4AA  BDF804            LDA PSAVE,X    ;RECALL INIT.
C4AD  48                PHA            ;P AND PUSH
C4AE  BC7806            LDY NOSAVE,X
C4B1  A92F              LDA #$2F       ;RESTORE INT-
C4B3  9982C0            STA PORTB,Y
C4B6  A98D              LDA #$8D       ;ASCII CR
C4B8  2C58FF            BIT IORTS
C4BB  48        EXIT    PHA
C4BC  BC7805            LDY YSAVE,X    ;RESTORE REG
C4BF  BDF805            LDA XSAVE,X
C4C2  AA                TAX
C4C3  68                PLA
C4C4  5001              BVC RETN       ;RESTORE P ON
C4C6  28                PLP            ;FINAL EXIT
C4C7  60        RETN    RTS
C4C8  BC7806    RDCLOK  LDY NOSAVE,X
C4CB  9982C0            STA PORTB,Y
```

```
C413  D7D3C5  0046         DFD "WSE"
C416  8DF807  0047  SKIP   STA SLOT
C419  290F    0048         AND #$0F     ;$0N
C41B  AA      0049         TAX
C41C  98      0050         TYA          ;SAVE REG
C41D  9D7805  0051         STA YSAVE,X
C420  68      0052         PLA
C421  9DF805  0053         STA XSAVE,X
C424  68      0054         PLA
C425  9D7804  0055         STA ASAVE,X
C428  5004    0056         BVC CONT
C42A  68      0057         PLA
C42B  9DF804  0058         STA PSAVE,X
C42E  8A      0059  CONT   TXA
C42F  0A      0060         ASL
C430  0A      0061         ASL
C431  0A      0062         ASL
C432  0A      0063         ASL
C433  9D7806  0064         STA NOSAVE,X
C436  5058    0065         BVC READ
C438  A536    0066         LDA CSWL     ;DETERMINE
C43A  D022    0067         BNE CONT1    ;IF ENTERED
C43C  ACF807  0068         LDY SLOT     ;AS A READ
C43F  C437    0069         CPY CSWH     ;OR WRITE
C441  D018    0070         BNE CONT1
C443  A0C7    0071  WRITE  LDY #$C7     ;RETN LOC
C445  8436    0072         STY CSWL
C447  A438    0073         LDY KSWL
C449  D007    0074         BNE NOREAD
C44B  ACF807  0075         LDY SLOT
C44E  C439    0076         CPY KSWH
C450  F00C    0077         BEQ CONT1
C452  BDF804  0078  NOREAD LDA PSAVE,X
C455  48      0079         PHA
C456  BD7804  0080         LDA ASAVE,X
C459  2C58FF  0081         BIT IORTS
C45C  705D    0082         BVS EXIT     ;ALWAYS TAKEN
C45E  BC7806  0083  CONT1  LDY NOSAVE,X ;INIT. PIA
C461  B983C0  0084         LDA CRB,Y
C464  D018    0085         BNE DONE     ;IF NEEDED
C466  9981C0  0086         STA CRA,Y
C469  9980C0  0087         STA PORTA,Y
C46C  A9FF    0088         LDA #$FF
C46E  9982C0  0089         STA PORTB,Y
C471  A904    0090         LDA #$04
```

```
C4CE  EA      0136         NOP          ;WAIT A
C4CF  EA      0137         NOP          ;SHORT WHILE
C4D0  B980C0  0138         LDA PORTA,Y
C4D3  48      0139         PHA
C4D4  BDF806  0140         LDA TBLPTR,X
C4D7  C9EB    0141         CMP #$EB
C4D9  F004    0142         BEQ MASK     ;MASK D10 BIT
C4DB  C9F1    0143         CMP #$F1     ;ALSO H10 BIT
C4DD  D004    0144         BNE NOMASK
C4DF  68      0145  MASK   PLA
C4E0  2903    0146         AND #$03     ;REMOVE
C4E2  48      0147         PHA          ;STATUS BITS
C4E3  68      0148  NOMASK PLA
C4E4  290F    0149         AND #$0F     ;BCD DATA
C4E6  09B0    0150         ORA #$B0     ;APPLE ASCII
C4E8  30D1    0151         BMI EXIT     ;ALWAYS TAKEN
              0152  ;
C4EA  00      0153         DFD 00
              0154  ;
              0155  ;
C4EB  A036A0  0156  TABLE  DFD A036A0   ; W
C4EE  3A39AF  0157         DFD 3A39AF   ;MM/
C4F1  3837AF  0158         DFD 3837AF   ;DD/
C4F4  3C38A0  0159         DFD 3C38A0   ;YY
C4F7  3534BB  0160         DFD 3534BB   ;HH;
C4FA  3332BB  0161         DFD 3332BB   ;MM;
C4FD  3130FF  0162         DFD 3130FF   ;SS
```

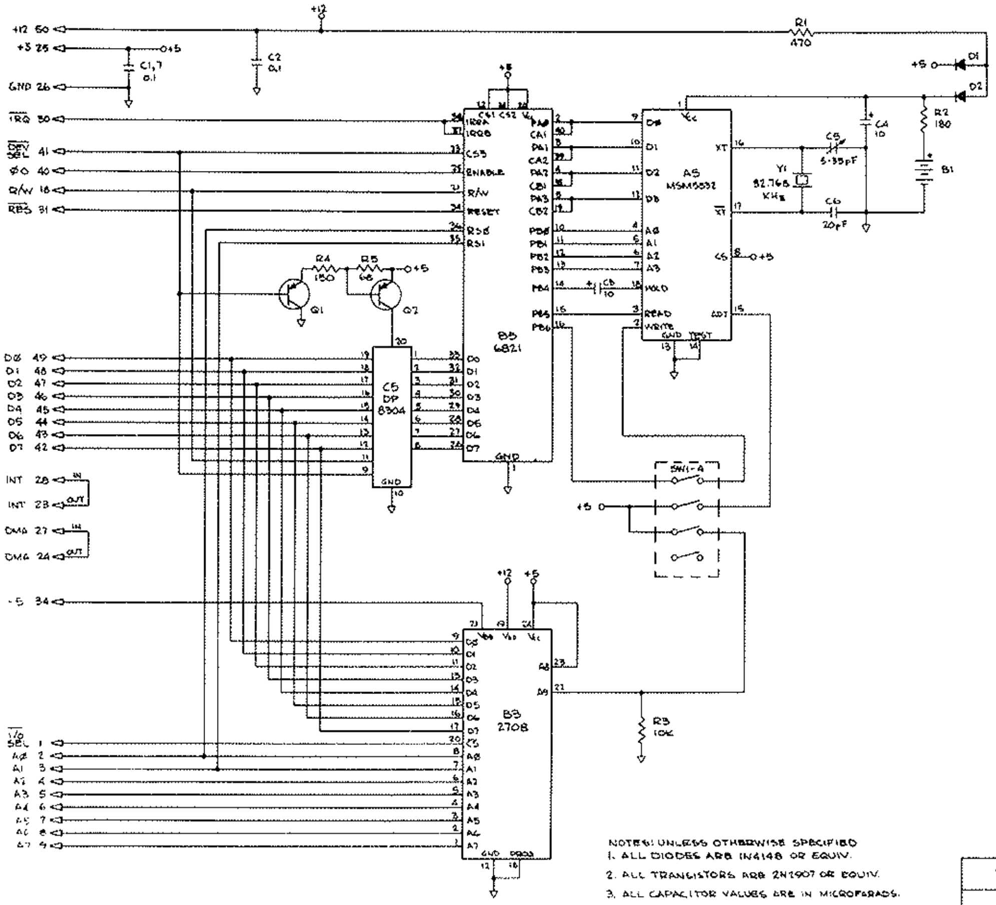
## ACE MODE CHANGES

```
C413  2CFF68  0047         DFD 2CFF68   ;ACE BYTES

C4D7  C9E9    0142         CMP #$E9
C4D9  F004    0143         BEQ MASK     ;MASK D10 BIT
C4DB  C9EC    0144         CMP #$EC     ;ALSO H10 BIT

C4EB  A03A39  0157  TABLE  DFD A03A39   ; MM
C4EE  AF3837  0158         DFD AF3837   ;/DD
C4F1  A03534  0159         DFD A03534   ; HH
C4F4  BB3332  0160         DFD BB3332   ;;MM
C4F7  BB3130  0161         DFD BB3130   ;;SS
C4FA  AE363C  0162         DFD AE363C   ;.WY
C4FD  3BFFFF  0163         DFD 3BFFFF   ;Y
```

| REVISIONS | | |
|---|---|---|
| LTR | DESCRIPTION | DATE |
| | | |



NOTES: UNLESS OTHERWISE SPECIFIED
1. ALL DIODES ARE IN4148 OR EQUIV.
2. ALL TRANSISTORS ARE 2N1907 OR EQUIV.
3. ALL CAPACITOR VALUES ARE IN MICROFARADS.
4. ALL RESISTOR VALUES ARE IN OHMS, 1/4W, ±5%.

WEST SIDE ELECTRONICS

SCHEMATIC -
SUPERCLOCK II

| DRAWN<br>J. BARNETT | DATE<br>1 DEC 80 | C | | REV |
|---|---|---|---|---|
| OK. | | SCALE NONE | SH 1 OF 1 | |