TO: Norman Leung
Conrad Chen

May 31, 1982

From: Peter Baum

1) I think that the document "Software Control of Disk II/IWM Controller" should also contain specific examples in certain section. These examples should be included in the sections where the operation requires time dependent code. The reason I suggest this is that the 65C02 has a few operations that use a different number of cycles then their counterparts in the NMOS 6502. If the developer decides to use his own sequence of instructions, they may not work properly with the other processor. It will also make things easier in the future if we ever decide to use a processor that is similar to the 6502, but has a few operations with different cycle times. A specific example would make things easier on the programmer.

2) It has been brought to my attention by a developer that we don't follow the guidelines exactly for the write self sync byte operation in ProDOS. Here is what I was told:

```
          Last-1 self sync byte
       40<
          Last self sync byte
***--> 36<
          1st address mark byte ($D5)
       32<
          2nd address mark byte ($AA)
       32<
          3rd address mark byte ($AD)
***--> 36<
          1st data nibble
       32<
          2nd data nibble
```

I have not checked this myself because I am not sure where the routines are located in ProDOS. The developer didn't think this would affect the operation of ProDOS, but I think we should check into it just to make sure.

3) I am also unclear from the document where the address mark and data mark fit in. Nowhere in the document does it tell when to write these bytes out, but it does mention something about reading them under the section titled "Read self sync". Specifically, shouldn't this be covered in the section on writing the self sync byte? Currently this section shows how to write the self sync bytes, followed directly by the first data byte. Is that correct

Peter Baum
20525 Mariani Ave.
Cupertino, Ca. 95014

Apple Computer
MS 22-W

May 24, 1984

Differences Between the IWM and the Apple // Disk Controller

The IWM (Integrated Woz Machine) is a single chip (LSI) implementation of the Disk II Controller state machine, designed and patented by Steve Wozniak. It is not an exact replica of the state machine, and though all of Apple's disk operating systems (DOS 3.3, ProDOS) work with it, 3rd party developers will find that some of their software will not work. This document describes all the known differences between the IWM, currrently used in the Apple //c, and the state machine used in the Apple II and //e disk controller card. This document only exists to describe differences and is not intended as a specification for the IWM or Apple // Disk Controller. The correct method of using the disk interface is described in another document, "Software Control of the Disk II or IWM Controller", which is required reading before perusing this document. The document is distributed to registered developers under the auspices of Apple // Developer Technical Support. (For more information contact: Apple Computer at above address - Certification/Registration Program, Mail Stop 23AF). This document will be updated as more differences are uncovered.

The Differences

1) When reading the write protect switch only the most significant bit (MSB) should be tested for the status. On the Disk Controller other bits can be tested, but this will not work on the IWM.

2) The IWM will not work properly in an Apple // if it is placed into a mode where both Q6H and Q7H are set and the motor is off (Q4L). If the chip is placed in this mode at any time then all subsequent operations of the IWM may not work properly.

3) The IWM does not require the false-read cycle, which occurs on indexed write instructions such as STA $C05E,X, to work properly. The Disk Controller state machine depends on the false read cycle during the STA Q7H,X instruction to store data properly into the shift register. This anamoly means that software which may work with the IWM will not work with the Disk Controller. This type of problem will generally appear if a store absolute instruction is used to set Q7H (STA $C0EF), since there is no false-read cycle on store absolute operations.

Other Notes of Interest

The IWM has never been sold in a Disk Controller card for the Apple // and there are no plans to do this. All Disk][Controller Cards, sold with either the Disk][or the Duodisk, use the state machine.

In the Apple //c the 65C02 processor, manufactured by GTE and NCR, is used. Some of the instructions use different cycle times than the corresponding instructions in the NMOS 6502 used in the Apple //e and][+. If these instructions are used as part of the 32usec. or 40usec. loops required in the disk write routines, then the write routines will not work properly in both the Apple //e (or][+) and Apple //c.

Other documents and books which may prove helpful:

Beneath Apple DOS, by Don Worth and Pieter Lechner, from Quality Software

Understanding the Apple II, by Jim Sather, from Quality Software

Software Control of the IWM or Disk][Controller , Apple Computer*

The 6502 False-read cycles, Apple Computer*

* -- These documents are distributed to registered developers under the auspices of Apple // Developer Technical Support. (For more information contact: Apple Computer at above address - Certification/Registration Program, Mail Stop 23AF).

Ray Klein
Sunrise Software
10855 Sorrento Valley Rd.
San Diego
Ca.
92121
Larry Roddenberry
-Home w/ Steve Park (Hayes=work)
6448 C Parton Ct.
Norcross
Ga.
30092
David Krankshaw
Scarborough Systems
25 No. Broadway
Tarrytown
N.Y.
10591

SEE BACK

IWM

Addendum to Software Control of the IWM or Disk II Controller

PREFACE

The purpose of this document is to briefly illustrate the instructions needed to operate the disk controller. The examples which are given show only the instructions that are needed to operate the disk drive and the timing sequence of these instruction. An example of executable code has not been given since the operation within the timing loops depends on other factors, such as the current disk function. The operations described include:

Selecting the Drive
Turning the Motor On
Sensing the Write Protect Switch
Reading a Data Byte
Writing a Data Byte
Writing a Self-Sync Byte
Reading a Self-Sync Byte
Seeking to Another Track

This document assumes that the reader has some knowledge of disk formats and operating systems. (A good primer for this type of information is the book 'Beneath Apple DOS'). For example, this document doesn't show how to format a disk, but assumes that the developer knows how a disk is formatted, including the address marks and address field format.

```
OURCE   FILE #01 =>IWM
000:                 1 *
000:                 2 *
000:                 3 * THIS ROUTINE WILL DETERMINE WHETHER A STATE MACHINE DISK II CONTROLLER
000:                 4 * OR AN IWM CONTROLLER IS INSTALLED IN THE SYSTEM.
000:                 5 * UPON EXIT FROM THE ROUTINE, Y=1 MEANS IWM CONTROLLER AND Y=0 MEANS
000:                 6 * STATE MACHINE DISK II CONTROLLER.
000:                 7 *
000:                 8 * ASSUME A MOTOR OFF INSTRUCTION [LDA $C088,X] HAS BEEN EXECUTED FOR
000:                 9 * TWO SECONDS BEFORE THE USER CALLS ON THIS ROUTINE.  OTHERWISE, A
000:                10 * TWO SECOND DELAY LOOP MUST BE ADDED AFTER THE FIRST MOTOR OFF
000:                11 * INSTRUCTION [LDA $C088,X]  AT THE BEGINNING OF THIS ROUTINE.
000:                12 *
000:                13 * THE ENABLED DISK DRIVE WILL CONTINUE TO BE ON FOR 1 SEC
000:                14 * AFTER EXIT FROM THIS ROUTINE.
000:                15 *
000:                16 *
---- NEXT OBJECT FILE NAME IS IWM.0
000:        1000    17            ORG   $1000
000:AE 41 10        18            LDX   SLOTX16       ;X REG=SLOT NO. X 16
003:BD 88 C0        19            LDA   $C088,X       ;MOTOR OFF
006:A0 00           20            LDY   #00           ;CLEAR REG. Y
008:BD 8D C0        21            LDA   $C08D,X       ;Q6H
00B:BD 8F C0        22            LDA   $C08F,X       ;Q7H, ADDRESS MODE REG.
00E:A9 04           23            LDA   #$04
010:9D 8F C0        24            STA   $C08F,X       ;DISABLE TIMER BIT IN MODE REG.
013:BD 8E C0        25            LDA   $C08E,X       ;Q7L, OUT OF WRITE MODE
016:BD 89 C0        26            LDA   $C089,X       ;MOTOR ON
019:48              27 LOOP       PHA                  ;10 MSEC DELAY LOOP
01A:68              28            PLA                  ;WAIT FOR THE MOTOR ON
01B:48              29            PHA                  ;SIGNAL TO BE ACTIVE IN
01C:68              30            PLA                  ;THE CONTROLLER CARD
01D:48              31            PHA
01E:68              32            PLA
01F:48              33            PHA
020:68              34            PLA
021:48              35            PHA
022:68              36            PLA
023:C8              37            INY
024:D0 F3   1019    38            BNE   LOOP          ;END OF DELAY LOOP
026:BD 8E C0        39            LDA   $C08E,X       ;Q7L, READ STATUS REG.
029:9D 88 C0        40            STA   $C088,X       ;MOTOR OFF
02C:29 1F           41            AND   #$1F          ;MASK 5 L.S. BITS
02E:C9 04           42            CMP   #$04          ;CHECK TIMER BIT
030:D0 09   103B    43            BNE   DISKII        ;DISK II CONTROLLER IF NOT EQ
032:C8              44 IWM        INY                  ;INCREMENT Y REG.
033:BD 8F C0        45            LDA   $C08F,X       ;Q7H, ADDRESS MODE REG.
036:A9 00           46            LDA   #$00
038:9D 8F C0        47            STA   $C08F,X       ;Q7H, RESTORE MODE REG.
03B:BD 8E C0        48 DISKII     LDA   $C08E,X       ;Q7L,
03E:BD 8C C0        49            LDA   $C08C,X       ;Q6L, RESTORE TO READ MODE
041:        1041    50 FIN        EQU   *
041:60              51 SLOTX16    DFB   $60
```

```
?03B DISKII          ?1041 FIN            ?1032 IWM              1019 LOOP
1041 SLOTX16
* SUCCESSFUL ASSEMBLY := NO ERRORS
* ASSEMBLER CREATED ON 15-JAN-84 21:28
* TOTAL LINES ASSEMBLED     51
* FREE SPACE PAGE COUNT    89
```