Software Protocol for Directly Connected Disks
7-for-8 version

Karl B. Young, Michael Hanlon
Version1.1, March 28, 1985

1.0 Introduction

This document will attempt to define the software protocol for devices (usually disks) connected directly over the external drive port of the Macintosh. Other aspects of such a connection have been discussed in two other documents: "DB19/IWM to Rigid Disk Interface Specification" dated 2/13, and "Notes on IWM Rigid Disk Interface Meeting" dated 2/27. In all probability, this document and those two should be combined into one great specification as soon as possible.

We will address what appears as sections II and III in the other two documents.

2.0 Handshake and Data Transmission

The biggest item in this section is that data transmission has been changed to be what we call 7-for-8 transmission, i.e., we get seven bytes of data for every eight bytes transmitted. A more detailed discussion of this must exist somewhere, but I don't know the reference. I will try to dig it up. Most significantly, the devices no longer need to worry about the HOBit being sent across the HDSEL line.

In order to support HOLDOFF under 7-for-8 transmission, the currently sending device (either Mac or the drive) will resume transmission (when HOLDOFF is deasserted) with the group that was interrupted. In this way, an SCC interrupt can be detected at the beginning of a group and serviced without worrying about finishing the group.

The actual handshakes for the drive will remain virtually the same. The one exception is that all transmissions to the drive under the new protocol will now receive a "Fast-ACK", whose timing is shown in the illustration entitled "Data Transmission from Mac to Drive" at the end of this document. The other direction for the handshake is illustrated in "Data Transmission from Drive to Mac".

3.0 Command, Status and Data Formats

I provide here the formats for Status, MultiBlock Read, and MultiBlock Write. The synchronizing bytes ("sync-bytes") are shown in bold-face before the data (NOTE: since the sync-bytes are not encoded in any 7-for-8 group, they will always have the hi-bit set). Items to notice are that all communications from the Macintosh are answered by the so-called "Fast-ACK" which notifies of correct transmission. A "Fast-NAK" (negative acknowledge) is always has the value $D5, while the "Fast-ACK" (positive acknowledge) has the same value as the sync-byte which was on the group just sent. A "Fast-ACK" or "Fast-NAK" is a type of sync-byte.

Status

From Mac:	<$AA><$03> <pad> <pad> <pad> <pad> <pad> <pad> <CHK>
From Drive (fast ACK):	<$AA>
From Drive:	<$AA><$83> <pad> <stat> <Identity Block> <pad> <pad> <CHK>

The identity block is 36 bytes long), and therefore a total of 42 bytes, or 6 groups, are sent. Assuming a truly "packed" structure, the ID block can be defined as follows:

```
ID_Block = packed record
    NameString:     packed array [1..13] of char;     { Name of device }
    Device_Type:    0..$FFFFFF;                       { Device code = $000110 }
    Firmware_Rev:   integer;                          { Revision # of firmware }
```

```
Capacity:          0..$FFFFFF;          { # of blocks on device }
Bytes_per_block:   integer;             { = 532 for René}
Num_cylinders:     integer;             { = 610 for René }
Num_heads:         byte;                { = 2 for René }
Sectors_per_track: byte;                { = 32 for René }
Possible_spares:   0..$FFFFFF;          { = 76 for René } ·
Num_spares:        0..$FFFFFF;          { Number of spared blocks }
Num_bad:           0..$FFFFFF;          { Number of bad blocks }
end;
```

MultiBlock Read

From Mac: **<$AA>** <$00> <count> <block# (3 bytes)> <pad> <CHK>
From Drive (fast ACK): **<$AA>**
note: the following occurs "count" times
From Drive: **<$AA>** <$80> <seq #> <stat> <532 bytes of data> <pad> <CHK>

Note that the command includes only one byte for the block count. The (possibly multiple) reponses will increment the sequence #, starting at 0. For example, if the Macintosh requests 10 blocks, the sequence numbers will go from 0 to 9.

MultiBlock Write

From Mac: **<$96>** <$01> <count> <block# (3 bytes)> <532 bytes of data> <pad> <CHK>
Fast Ack: **<$96>**
note: the following occurs "count-1" times
From Mac: **<$96>** <$01> <seq #> <3 byte pad> <532 bytes of data> <pad> <CHK>
Fast Ack: **<$96>**

In this case, when the block count is greater than 1, the sequence numbers start at $01 and continue up to one less than block count; this is because the first block (sequence number 0) is included with the command. The first block of data is sent with the command in order to optimize one-block writes, of which there seem to be a lot in the Macintosh.

Note that there are three bytes of padding between the sequence number and the write data when sending more than one block. This is so that the data will line up the same during each write-transmission, which should make it easier for coding.

MultiBlock Write-Verify

From Mac: **<$96>** <$02> <count> <block# (3 bytes)> <532 bytes of data> <pad> <CHK>
Fast Ack: **<$96>**
note: the following occurs "count-1" times
From Mac: **<$96>** <$02> <seq #> <3 byte pad> <532 bytes of data> <pad> <CHK>
Fast Ack: **<$96>**

The reads and writes are illustrated at the end of this document on the page entitled "Read/Write Protocol".

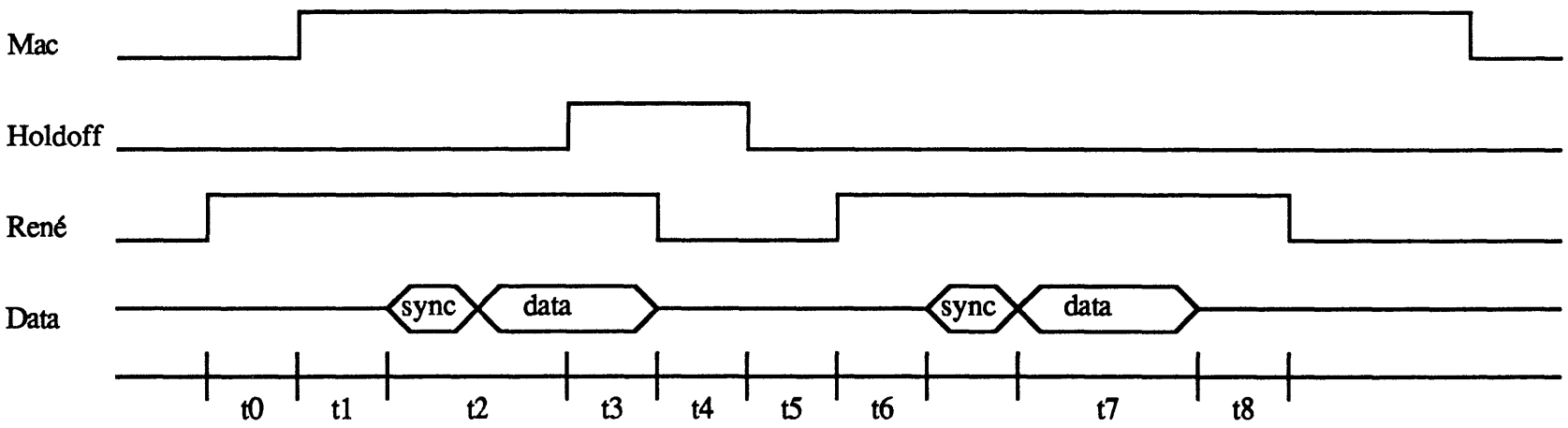
In order to facilitate testing, we reserve one other command (called "Diagnostic") which has the following format:

Diagnostic

From Mac: **<$AA>** <$04> <5-byte pad> <CHK>
From Drive (fast ACK): **<$AA>**

Commands that follow the diagnostic opcode to the drive will follow a special diagnostic protocol.

Data Transmission From Mac to Drive



t0 - René will wait forever for Mac to respond to handshake.

t1 - René will send sync within 33us.

t2 - Mac may assert holdoff anywhere in a group. That group will be ignored and will not be included in the checksum.

t3 - René will acknowledge the holdoff immediately after the last byte of the group is sent

t4 - René will wait forever fro holdoff to de-assert.

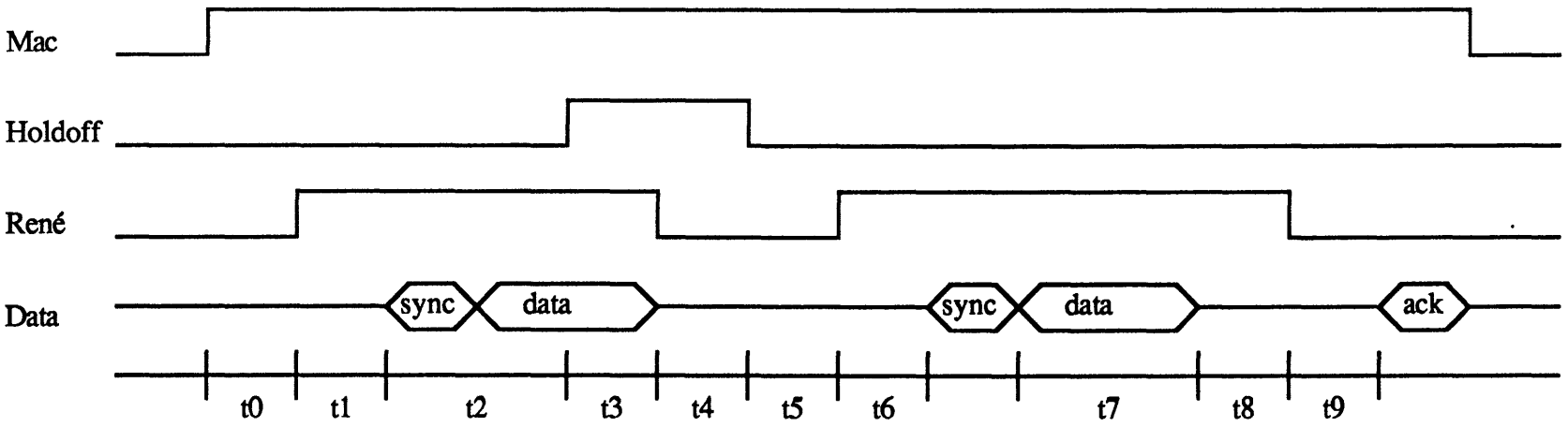
t5 - René will respond to de-assertion of holdoff within 18us.

t6 - Transmission starts with group that was interrupted by holdoff within 34us.

t7 - Data transmission time is dependent on the number of groups sent (8 bytes * byte time * number of groups).

t8 - René signals end of transmission within 3us of last byte loaded into IWM. Mac will received it one byte time later.

Data Transmission From Mac to Drive



t0 - René normally responds to Mac in 14us, but may take as long as 2 seconds if doing self test.

t1 - René will wait forever for a valid sync byte.

t2 - Mac may assert holdoff anywhere in a group. That group will be ignored and will not be included in the checksum.

t3 - René will acknowledge the holdoff immediately after the last byte of the group is received.

t4 - René will wait forever fro holdoff to de-assert.

t5 - René will respond to de-assertion of holdoff within 14us.

t6 - Same as t1. Transmission starts with group that was interrupted by holdoff.

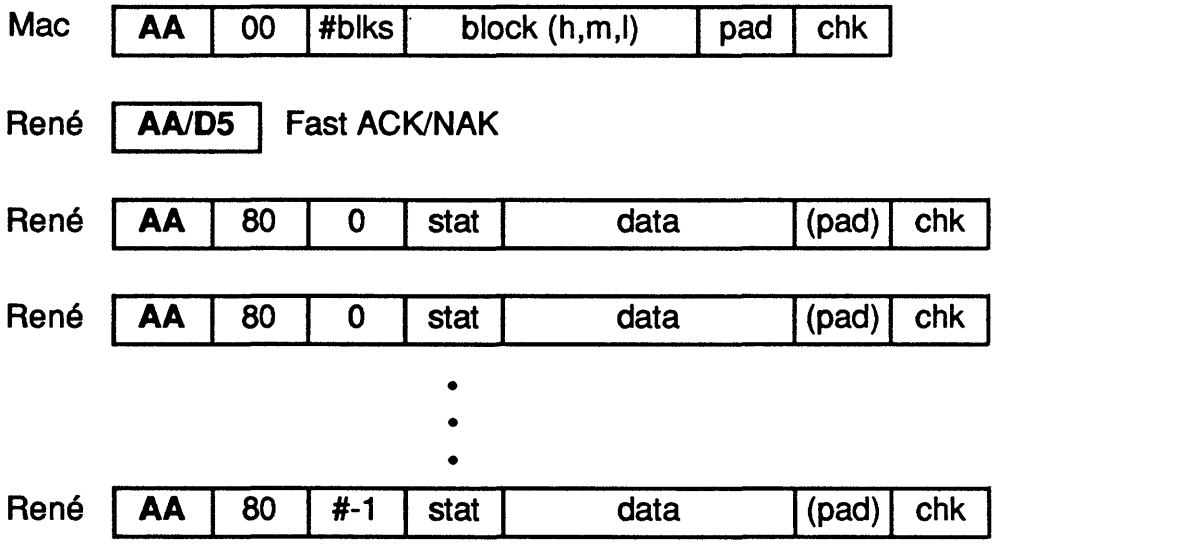
t7 - Data transmission time is dependent on the number of groups sent (8 bytes * byte time * number of groups).

t8 - René acknowledges end of transmission within 3us of last byte received.

t9 - René will send ACK/NAK 35-40us after handshake. Mac will receive it one byte time later.

Read/Write Protocol

Multi-Block Read Command:



Multi-Block Write-Write/Verify Command: