VFW-III

VERSAFLOPPY WINCHESTER DISK CONTROLLER

OPERATIONS MANUAL

SD #7140174
REVISION A
SEPTEMBER 21, 1983

SD SYSTEMS
P.O. Box 28810, Dallas, Texas 75228

Preliminary

VFW-III

VERSAFLOPPY WINCHESTER DISK CONTROLLER

OPERATIONS MANUAL

SD #7140174
REVISION A
SEPTEMBER 21, 1983

TABLE OF CONTENTS

SECTION III
SOFTWARE DRIVERS

TABLE OF CONTENTS--Continued

ILLUSTRATION

SECTION I
INTRODUCTION

1.0 GENERAL

The SDSystems VFW-III is a high performance disk controller capable of running **both** Winchester hard disk and floppy disk drives. This unique design brings together the function of several controller boards onto a single S-100/IEEE-696 compliant board. Winchester drives designed for the industry standard ST506/412 interface comprising up to 16 read/write heads **and** floppy disk drives of single or double density, single or double sided and 5.25" minis or 8" full size configurations may be controlled concurrently by the VFW-III. Data may be interchanged with the controller via a DMA mode wherein the board becomes a Temporary Master capable of addressing any location within the 24 bits of address defined by the IEEE-696 specification, or via a programmed I/O mode wherein data is sequentially transferred through a single port of the controller. On-board processors transfer sector data between the internal sector buffer and the selected drive without need of the host processor's supervision, thus freeing the host for other activities such as servicing interrupts. Automatic retries, CRC generation/verification and optional error correction (on hard disk transfers) of a single burst error of five bits ensure data integrity and further unburden the CPU from monitoring these events.

1.1 FEATURES

- Versatile Disk Configurations:

  Three varied 5.25" Winchester hard disk drives utilizing the ST506/412 interface with up to 16 read/write heads may be controlled by the VFW-III, along with four floppy disk drives in any mixed configuration from single sided, single density, 5.25" minis to double sided, double density, 8" full size drives.

- Fully S-100/IEEE-696 Compliant:

  When operating as a Temporary Master, the controller can perform eight bit Direct Memory Accesses (DMA) for faster transfers of data between the sector buffer and any memory location within the extended memory addressing range of 16Mbytes (24 bit addressing).

- Data Integrity:

  Single burst errors of up to five bits are automatically corrected on data received from 5.25" Winchester drives formatted in ECC mode. Single burst errors of 20 bits and double burst errors of four bits can be detected from a 256 byte sector.

- Convenient Packaging:

  By supplying the function of two controllers in a single board, the VFW-III allows existing systems to expand to Winchester drives without sacrificing an S-100 slot and allows new systems to get more system performance into small card cages. Two regulators and heatsinks have been used to assure adequate cooling even under conditions where the 8 volts unregulated input is at its IEEE-696 maximum value of 11 volts DC. Care has been taken to place all cable connectors along the top edge of the board for optimum access.

- Flexible Data Storage and Transfers:

  Sector sizes of 128, 256, 512 and 1024 bytes and physical placement of the logical sectors for interleaving are both software selectable. The VFW-III processors supervise all disk to on-board single sector buffer transfers, requiring only that the host CPU issue the command sequence and later evaluate the command completion status. When Temporary Master DMA transfers are used, a completion interrupt may be issued after all sectors have transferred between the host memory and the selected drive. Using programmed I/O mode requires the host CPU to move sector data between the on-board buffer and host memory.

- Interruptable Data Transfers:

  When operating the VFW-III under programmed I/O transfer mode, interrupts may be serviced in the middle of a sector buffer to host transfers. On board processors transfer sector data between the internal sector buffer and the selected drive without need of the host processor's supervision. In this mode of operation, no interrupts need be missed.

1.2 SCOPE OF THIS MANUAL

This manual is divided into five sections, each directed toward a need of the user. Section I summarizes the board's features and defines the structure of the manual. Section II defines terms involved in disk operation, basic flow of information on the VFW-III and location of various circuits on the schematics. It is written with the novice in mind and is an easy introduction into rotating media storage devices. Information required to write or modify a software driver is presented in Section III. It includes the memory map of the controller and the command set to it. Section IV contains all information required to physically install the board into an existing system and the power and cooling requirements of the board. Specifications of the VFW-III are listed in the last section. Reference material is located in the appendices.

SECTION II
THEORY OF OPERATION

2.0 GENERAL

Section II is intended to be used as a light tutorial on the S-100 computer bus, the Winchester and floppy disk drives, and the interface that connects them together. The block diagram (Figure 2-1) illustrating the functions of the VFW-III is discussed in Section II, Subsection 2.4.

2.1 S-100 COMPUTER BUS

The S-100 computer bus has a 100 pin parallel backplane. This means that the slot in which a board is plugged into the bus is inmaterial. Information required for any card on the bus is presented to every location. An addressing scheme is used to specify which board is to respond to the following bus cycle. The S-100 defines two basic transfers; the memory access and the input/output (I/O) device access. The address of a memory location can be either 16 or 24 bits long for either 65,536 or 16,777,216 memory locations. The address of an I/O port can be either 8 or 16 bits long for a range of 256 or 65,536 locations. S-100 boards monitor the bus for an address within the range of locations which it alone contains. Once selected by the address, a board will perform the command presented on the control lines.

All addressing is done by one board at a time; either the permanent master or one of 16 temporary masters. The permanent master is usually the main CPU and will relinquish the bus using a priority scheme that ensures that only one master is controlling the bus at a given time. The VFW-III can speed data across the bus, avoiding the two step process wherein the CPU first reads from the controller and second writes into memory, by becoming a temporary master. This allows the VFW-III to transfer data directly between its single sector buffer and any memory location within in a range of 16,777,216 locations. Data transferred in this fashion is quicker but may not be chosen for an application where real-time interrupts must be handled quickly. TMA transfers disable the CPU from accessing the bus to service a pending interrupt until after the current sector is transferred. Operating the controller in a programmed I/O mode, where the CPU always controls the bus and moves data in the slower two-step process described above, allows immediate access to the bus for servicing interrupts in the middle of data transfers without disturbing them. In this mode, all transfers made with the controller are I/O transfers.

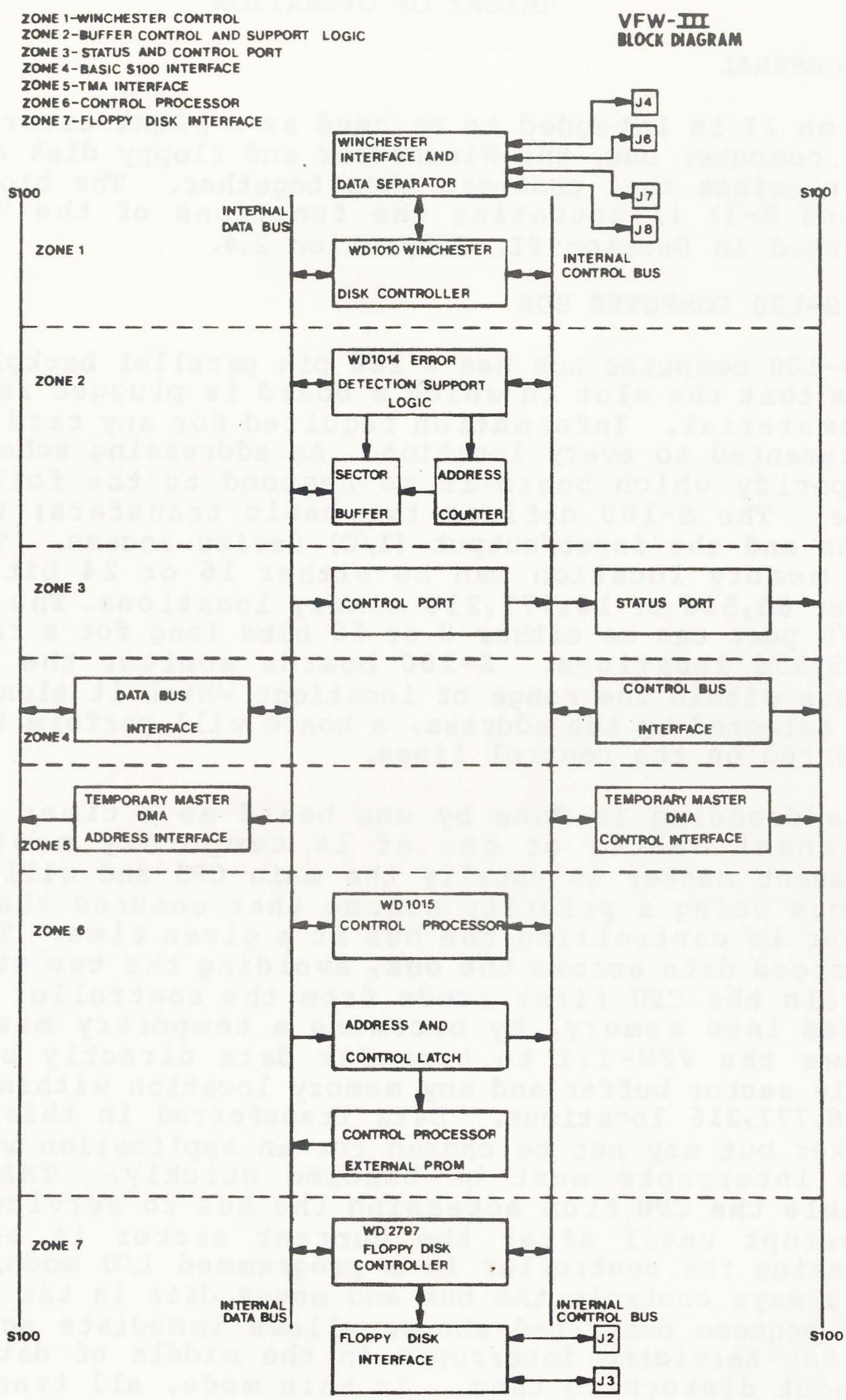


Figure 2-1. VFW-III BLOCK DIAGRAM

An interrupt bus is defined under the S-100 to stop the CPU to quickly do a time-dependent routine transparent to the present task. Interrupts are commonly used to get the CPU to transfer data with an input/output device and are more efficient than **polling status loops** where many wasted reads are made until the status changes, signalling an operation needed to take place. Status loops seldom respond as quick as interrupts.

Data lines exist for moving either 8 bit bytes or 16 bit words during memory or I/O transfers. All data transfers to or from the VFW-III will be 8 bit byte length only.

2.2 WINCHESTER DISK DRIVES

Winchester disk drives are rotating memory, random access storage devices which fulfill the need for fast, large, on-line data storage and retrieval. The word Winchester actually refers to the technology of the nickel-zinc formulated read/write head, but is commonly used to refer to 5.25" hard disk drives. The term "Winchester disk drive" is used interchangeably with "hard disk drive" throughout this manual.

A hard disk drive is a device which allows access to data stored magnetically within a vertical stack of several magnetic coated disks or **platters.** The stack is continuously spinning at speeds roughly 12 times as fast as a floppy disk drive with the read/write heads aerodynamically "flying" microns above each surface in the stack (two heads per platter; top and bottom). All heads are attached to an assembly sometimes called an **actuator arm,** which can be controlled to position the heads in or out along the radii of the stack. At any given position of the read/write heads, the magnetic media spinning below it that could be read or written would trace out a circle on the media. This circle is called a **track.** At each position of the actuator arm, its associated track is assigned a **track number.** Track numbers range from zero at the outermost concentric circle to the maximum number located nearest the stack axis. The movement of the actuator arm from one track to another is termed a **seek** and is issued by the controller as a series of pulses called **steps.** The time required to electically switch from one read/write head to another above or below it is considerably less than the time taken to seek from one track to its neighboring track. For this reason, when the end of one track has been reached, the continuation of information will be found on the same track number but on the next higher head number. The vertical stack of tracks accessed by all the heads of the actuator arm is referred to as a **cylinder.** Information in the present cylinder can be accessed the quickest. When consecutive data passes the end of a cylinder boundary, a seek must be issued and the head number is reset to zero.

The OEM manual or product specification of every drive gives the **unformatted capacity** of the entire drive and of an individual track. This is the number of locations where bits of information may reside. Since each of these bits cannot be uniquely addressed, a method is used that sacrifices some of the bits to identify blocks of continuous data. While this scheme is common among controller manufacturers, the identification blocks must be identical to allow information written by one company's controller to be read by another's. The VFW-III uses a Western Digital 40 pin chip to handle the **formatting** or writing of the pattern required for subsequent reads and writes. As this chip gains popularity in the market place, compatibility among controllers will increase. It is important to point out that more is involved than the physical access to the old data. The driving software must also be compatible.

Each track is subdivided into units called **sectors.** The **data field** which contains the information to be read or written is accompanied by an identification or **ID field** and gaps before, between and following these fields. The ID field contains the "address" of the sector and includes the cylinder number, the head number and the sector number. This is the information that the Central Processing Unit (CPU) or **host processor** must provide to the controller board in order to transfer the associated data field. The controller takes all action nessessary to locate the **target sector** and transfer the data field between the drive and the controller's on-board single sector buffer. All transfers between the host and the controller are independent of transfers between the controller and the drives. This means that the host needn't be "locked-up" as it waits to move data directly with the drive as some controllers require. If programmed I/O mode is used to move data with the controller, any higher level **interrupts** or conditions requiring immediate action may be handled in the middle of the transfer, only to complete it after the interrupt has been serviced.

An **index pulse** is used to the controller when a reference radius of the disk stack passes below the read/write heads. This reference is used to initiate formatting **physical sectors,** which are the actual sector fields placed consecutively along the track. To increase the **throughput** or average rate at which information is transferred on a track, a technique known as **sector interleaving** is used. If physical sectors' ID fields contain consecutive numbers (an interleave factor of 1), a dilemma occurs when two sectors are to be read. As the first sector spins under the read/write head, its ID field matches the target sector, and the following data field is transferred into the controller's single sector buffer. When the last bytes of data are read in, the controller begins to move the saved data into the host system's memory, at the same time that the next target sector is spinning past the read/write head. The disk controller

then returns, looking for the next target sector's ID field, and now must wait a full revolution for it to spin under again.

Sector interleaving makes a distinction between the physical sector and the **logical sector.** A logical sector is the next sector that the system would request, but its position is staggered around the disk so that while the buffer is moved into the host, unwanted sectors spin below the read/write head. Shortly after the controller returns, searching for the next logical sector, it spins under the head. The minimum interleave factor the VFW-III handles on a Winchester is 3; every third physical sector is the next logical sector. The table below illustrates an imaginary case where 16 sectors of 512 bytes length are formatted with an interleave factor of 3.

Physical Sector	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Logical Sector	1	12	7	2	13	8	3	14	9	4	15	10	5	16	11	6
1st Revolution	1	.	.	2	.	.	3	.	.	4	.	.	5	.	.	6
2nd Revolution	.	.	7	.	.	8	.	.	9	.	.	10	.	.	11	.
3rd Revolution	.	12	.	.	13	.	.	14	.	.	15	.	.	16	.	.

This shows that the entire track can be read in 3 revolutions with an interleave factor of 3 as opposed to 16 revolutions (1 per sector) for an interleave factor of 1 or 2. This increases the throughput by 5.3 times. The interleave factor is set only when a drive is formatted and is software selectable as explained in Section IV.

An important misconception about the interleave factor is that it should always be set to the minimum value handled successfully by a disk controller. The proper interleave factor is dependent not only upon the time it takes the controller to empty its sector buffer, but also upon the average time between consecutive sector requests. This time may be lengthened by interrupt servicing or other computations between sectors, as may occur when searching files. If it is set too low, the next sector will spin under before the controller begins searching again, resulting in 1 sector per revolution. This throughput is much more noticeable on flopy disk drives where the magnetic media is spinning 12 times slower. The proper procedure for determining the interleave factor is to run **benchmarks** or timing tests in the final system using typical application software.

2.3 FLOPPY DISK DRIVES

A floppy disk drive can be considered to be a simpler version of the hard disk drive discussed in Section II, Subsection 2.2. Only one removable magnetic disk is used in a floppy drive. A flexible plastic jacket protects and supports the media and has an open slot on front and back for the one or two read/write heads. Track densities are roughly 1/3 to 1/10 that of hard disk drives and rotational speeds are 1/12. Data

is stored on the media using the same method described for Winchesters (ID fields, data fields, gaps and interleave factors).

Floppy drives vary, but their characteristics are usually some combinations of the following:

5.25" or 8" sized diskettes and drives.
Single or double density recording (FM or MFM).
Single or double sided drives (1 or 2 heads).
48 or 96 tracks per inch track density.

The VFW-III can handle a total of four floppy drives in any combination of the above characteristics with the appropriate driving software.

Removability is perhaps the greatest advantage of floppy disks. Large amounts of data may be **backed-up** or saved as redundant copies. The portability of the diskette makes it a convenient way of transferring information between computers. Eight inch single sided, single density format is one of the few controller independent, standard formats. Using this format, files may be transferred to diverse computer systems.

2.4 VFW-III DISK CONTROLLER

The VFW-III is a single board, six layer disk controller capable of interfacing three hard disk drives and four floppy disk drives with minimal software differences. It is designed around the Western Digital 4 chip set: the WD1010, WD1014, WD1015 and WD2797. On-board processors and sector buffer make the board more bus efficient, handling all disk actions necessary to locate the target sector specified by the CPU. Error detection and correction are done by the controller's processors and require no work or supervision of the host.

The block diagram for the VFW-III shows connection to three busses: the S-100, the internal data bus and an internal control bus. Note that the S-100 bus is drawn on both sides for clarity. Function has been divided into seven zones vertically and each is discussed in the following subsections.

2.4.1 Zone 1: Winchester Control

The heart of the Winchester control circuit is the WD1010 Winchester Disk Controller. This circuit is located on the schematic on Sheets 4 and 5 and provides the hard disk drive positioning logic, write precompensation logic and the data separator. The three drives are daisy-chained to the control connector J4 for selection, positioning and status signals. Connectors J5, J6 and J7 link drives 1, 2 and 3, respectively, to transfer high frequency read and write data along differential lines. Jumper 3, used in conjunction with bit 2

of the control port (BASE + 0B), can be used to extend the head addressing to 4 bits or 16 heads. New drives utilizing smaller Whitney read/write heads and internal spindle motors are making more than 100Mbytes of data available in a single standard size 5.25" drive.

2.4.2 Zone 2: Buffer Control And Support Logic

This circuit is located on the top half of Sheet 2 of the schematic. The WD1014 Error Detection/Support Logic calculates the four bytes of appended ECC syndrome as data to be moved to or from the sector buffer which it controls. Assorted other signals are decoded within the WD1014 save space on the board. The address counter to the sector buffer is automatically incremented after every access and cleared when a command is issued to the command port at location BASE + 07H.

2.4.3 Zone 3: Status And Control Port

Located at the top left of Sheet 1 and the top center of Sheet 3 of the schematic, this circuit is used by the host to determine what the status of the board is and to define parameters for the next disk access. Section IV, Subsections 4.1.9 and 4.1.14 define their use in detail.

2.4.4 Zone 4: Basic S-100 Interface

Sheet 1 and the left side of Sheet 3 of the schematic contain this circuitry. Address decoding, data bus buffering, interrupts, resets, wait states and power are the functions handled by this interface. Twelve ports are mapped onto this controller, and it decodes standard input/output device addressing as specified by the IEEE-696. All data transfers are eight bits wide, one software definable vectored interrupt may be selected and on-board reset may follow either pin 75 or 99. Jumper 20, when installed, will assert two wait states for every port access to the board. This is used for operation on a 6 MHz bus.

2.4.5 Zone 5: TMA Interface

Most of Sheet 1 of the schematic is used to control temporary master access on the S-100 bus. The loadable address counters are used to select host memory for data transfers. The two buffers are used to control or hold the status and control signals of the bus during TMA, and the two programmable logic arrays are used to handle TMA priority resolution, and then to control both the internal and external busses.

2.4.6 Zone 6: Control Processor

The control processor circuit on the lower half of Sheet 2 is designed around the WD1015. This device intercepts commands from the bus and converts them to what would be required by

either the Winchester or floppy contoller chips. The external PROM enhances the internal program, offering more formats to be supported. The address latch is used both to address the PROM and to address the other 40 pin chips. Data is moved on the internal data bus by the control processor between the sector buffer and the Winchester and the floppy controllers. When ECC errors less than or equal to five bits are detected, the WD1015 will perform the correction within the sector buffer before passing it on to the host.

2.4.7 Zone 7: Floppy Disk Interface

The floppy disk interface on Sheet 3 allows control of both 8" and 5.25" drives of either density or head count. Drives are connected to J2 and J3 for 5.25" and 8", respectively. Since both cables are electrically common, unique unit numbers must exist on both cables and only one terminator may be connected.

SECTION III
SOFTWARE DRIVERS

3.0 GENERAL

This section is intended to supply all necessary information to write or modify a software driver for the SDSystems VFW-III. Provided herein are the complete memory map of the controller and the command set for the commands.

3.1 MEMORY MAP

The VFW-III is mapped onto the S-100/IEEE-696 using Standard Input/Output Device Addressing. Address lines A7 through A4 are compared with the I/O base address set by SW1 positions 5 through 8 to determine if the board is to be selected. If so, address lines A3 through A0 are needed to select locations within the memory map of the controller. These 16 locations are illustrated in the following table as offsets from the base address.

I/O Address	Read Access	Write Access
BASE+00	Sector buffer	Sector buffer
BASE+01	Error register	Write precompensation cylinder
BASE+02	Sector count	Sector count
BASE+03	Sector number	Sector number
BASE+04	Cylinder low	Cylinder low
BASE+05	Cylinder high	Cylinder high
BASE+06	Size/device/ head register	Size/device/head register
BASE+07	Status register	Command register
BASE+08	Not selected	DMA address least significant byte
BASE+09	Not selected	DMA address most significant byte
BASE+0A	Not selected	DMA address extended significant byte
BASE+0B	Not selected	VFW-III control port
BASE+0C	Not selected	Not selected
BASE+0D	Not selected	Not selected
BASE+0E	Not selected	Not selected
BASE+0F	Not selected	Not selected

Note that when connecting W20 to enable two wait states for 6 MHz operation, the controller will also assert the wait for address BASE+0C through BASE+0F, even though these locations are not used. Other I/O cards requiring wait states for 6 MHz but not incorporating them within their designs may be mapped here.

3.1.1 Sector Buffer (Read/Write And Increment)

A 1K byte wide sector buffer, capable of containing only one sector regardless of sector size, is mapped into location BASE+00. Except for READLONG and WRITELONG commands, all access to this location must be by block transfers equal in length to the sector size, using the quicker DMA mode or programmed I/O mode. On-board counters, which are reset at the issuance of a command, address the Random Access Memory (RAM), consecutively incrementing the address after each access. In programmed I/O mode, this port is simply continuously read or written by the host processor until a full sector length is transferred. In DMA mode, the transfer of data between the host's memory and the sector buffer is handled by making the VFW-III a temporary master. Provided that the absolute address has been loaded into registers BASE+08, +09 and +0A and that the control port, BASE+0B, has enabled DMA mode and selected the proper direction, then the controller will move the data around without need of the processor.

3.1.2 Error Register (Read Only)

In addition to normal error reporting, this register will also hold the results of on-board diagnostics initiated by a power-up or by a test command. The diagnostic sequence starts with the WD1015 and progresses to the WD2797 or until the first error occurs wherein the error code is posted and the test terminated. Error codes for the internal diagnostics and their meaning are tabulated below. Note that the error bit of the status register will never indicate an error at the completion of the diagnostics.

Error Code	Meaning
0	All diagnostics ran error-free; test passed
1	Error found within the WD2797 Floppy Disk Controller
2	Error found within the WD1010 Winchester Disk Controller
3	Error found within the sector buffer
4	Error found within the WD1014 or internal bus
5	Error found within the WD1015 Control Processor

The error register is also used to determine the type of error encountered during the last command and is only valid if the status register (Section III, Subsection 3.1.9) indicates an error exists. An active high bit specificies the errors illustrated in the following table.

Bit	Error	Bit	Error
0	No data address mark	1	Track 0 not found
2	Command aborted	3	Not used
4	ID field not found	5	Not used
6	Incorrectable error	7	Bad block detect

3.1.2.1 No Data Address Mark

A no data address mark error occurs if, after successfully finding the target sector's ID field or header, the subsequent byte marking the beginning of the sector's data field is not found with 16 bytes. Possible caused of this error include: bad diskette, media flaw after the ID field, or controller out of calibration.

3.1.2.2 Track 0 Not Found

Both floppy and Winchester drives assert a signal when their read/write heads are on track 0. During a restore command, this signal is used to put the read/write heads in a known position by continuously stepping to a lesser track number and then testing the track 0 signal. If the drive does not assert the signal before 1024 steps are attempted, the VFW-III will post this error. One probable reason for receiving the error in conjunction with Winchester operation is that jumper W21 is not connecting C-2, meaning the common line jumpered to option 2 as shipped from the factory. The jumper corrects a design flaw in the LSI chip WD1010-00 Control Processor that made it incompatible with drives that did not deactivate the signal SEEK COMPLETE on the rising edge of step. Other possible reasons for this error are a bad cable connection or a drive that has not had its read/write heads unlocked after shipping. Note that few 5.25" Winchester drives use head-locking for transport.

3.1.2.3 Command Aborted

Certain conditions must be reported by the disk drive before the operation of a valid command can commence. Failure of these conditions results in the posting of this error by the VFW-III. Interrogation of the status register will show one of the following signals in an abnormal state.

3.1.2.3.1 Status Register Bit 6: Ready

The normal state for this signal is active high, which indicates that the drive is ready to read, write or seek. When a drive is disconnected, powered down or spinning up, this signal is inactive and forces an error if a command is sent to it.

3.1.2.3.2 Status Register Bit 5: Write Fault

The normal state for this signal is inactive low. Write fault on a Winchester indicates that a condition exists at the drive that could cause improper writing on the disk such as bad DC voltages, improper head selection or write gate/write data problems. Floppy drives will assert this signal if a diskette is write protected.

3.1.2.3.3 Status Register Bit 4: Seek Complete

Seek complete must be active high prior to execution of all commands except seek. It is used to signal that the Winchester read/write heads are stationary over a legitimate track, and it goes inactive low as the heads are seeking to another track. After a power-on, if the read/write heads are not over track 0, a recalibration will be initiated by the drive, during which seek complete will go inactive. Floppy disk access forces this status bit active high since there is no seek complete signal on its interface.

3.1.2.4 Identification Field Not Found

If the target sector cannot be found by the drive after the retry sequence, this error is posted. A drive that has been formatted with too many sectors per track will cause this error as those sectors that were overwritten are sought. Media errors in the identification field and a VFW-III out of calibration also cause this error.

3.1.2.5 Uncorrectable Error

This error indicates that bad data was received from the disk drive. An uncorrectable error is set when CRC formatted sectors are read and the two bytes of appended CRC do not match the value calculated on the data. ECC formatted Winchesters will post this error if an error of more than five consecutive bits is received. The VFW-III automatically corrects single burst errors of five or less bits before transferring the data to the host.

3.1.2.6 Bad Block Detect

A bad block indicator in the identification field will force this error. Known media flaws that fall within the data field can be mapped out during the track format by inserting a bad block indicator within the identification field. Subsequent reads or writes to the sector will cause this error, allowing the host to take further action.

3.1.3 Write Precompensation Cylinder (Write Only)

Since Winchester drives differ greatly in their number of cylinders, this register is provided to specify on which cylinder the Winchester controller activates both the REDUCE

WRITE CURRENT signal and the write precompensation logic. Twelve nanoseconds is the fixed write precompensation delay from nominal. Floppy disk drive write precompensation is not handled in this register and is unaffected by it.

In order to address a range of 1024 cylinders by use of an eight bit register, the following method is used. The target cylinder must be divided by 4, shifting the bits to the right two places. For example, to load the register for a cylinder of 512 decimal, simply divide by 4 to get 128 decimal or 80 hexadecimal. Note that the two least significant bits of the resulting cylinder will be assumed to be zeroes.

```
1 0 0 0 0 0 0 0 0 0 binary = 200 hexadecimal = 512 decimal
--->1 0 0 0 0 0 0 0 binary =  80 hexadecimal = 128 decimal
1 0 0 0 0 0 0 0 1 1 binary = 203 hexadecimal = 515 decimal
--->1 0 0 0 0 0 0 0 binary =  80 hexadecimal = 128 decimal
```

3.1.4 Sector Count (Read/Write)

The sector count register is used during multiple sector reads or writes to specify the number of sectors requested. A decrement of this register occurs after every sector transfer. Format commands use this register to specify the number of sectors to format on the present track.

3.1.5 Sector Number (Read/Write)

The starting sector number for multiple sector transfers or the target sector for single sector transfers is written into this register. When an error is encountered, this register can be read to identify which sector of a multiple sector transfer the error occurred on.

3.1.6 Cylinder Low (Read/Write)

Seek, Read and Write commands cause this register and the next one to specify the cylinder at which the read/write heads are to be positioned. This register holds the eight least significant bits of a 10 bit cylinder number field.

3.1.7 Cylinder High (Read/Write)

Seek, Read and Write commands cause this register and the previous one to specify the cylinder at which the read/write heads are to be positioned. This register holds the two most significant bits of a 10 bit cylinder number field as illustrated below. Bit 8 of the cylinder field has an additional function when used with floppy disk drives. Disks formatted at 48 tracks per inch (TPI) can be read from a 96 TPI drive if bit 8 is set to 1.

Cylinder High								Cylinder Low								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	Cylinder Registers
x	x	x	x	x	x	9	8	7	6	5	4	3	2	1	0	Cylinder Number Field
x	x	x	x	x	x	x	*	x	x	x	x	x	x	x	x	Floppy 48 or 96 TPI bit

3.1.8 Size/Device/Head Register (Read/Write)

The Size/Device/Head register is used to select ECC or CRC operation, sector size, Winchester or floppy drive and head number. The following table illustrates the use of the bits of this register.

7	6	5	4	3	2	1	0	Size/device/head register bits
0	x	x	x	x	x	x	x	Select CRC operation
1	x	x	x	x	x	x	x	Select ECC Winchester operation
x	0	0	x	x	x	x	x	Sector size = 256 bytes
x	0	1	x	x	x	x	x	Sector size = 512 bytes
x	1	0	x	x	x	x	x	Sector size = 1024 bytes
x	1	1	x	x	x	x	x	Sector size = 128 bytes
x	x	x	0	0	x	x	x	Winchester drive 1 select *
x	x	x	0	1	x	x	x	Winchester drive 2 select *
x	x	x	1	0	x	x	x	Winchester drive 3 select *
x	x	x	*	*	0	0	0	Winchester head 0
x	x	x	*	*	0	0	1	Winchester head 1
x	x	x	*	*	0	1	0	Winchester head 2
x	x	x	*	*	0	1	1	Winchester head 3
x	x	x	*	*	1	0	0	Winchester head 4
x	x	x	*	*	1	0	1	Winchester head 5
x	x	x	*	*	1	1	0	Winchester head 6
x	x	x	*	*	1	1	1	Winchester head 7
x	x	x	1	1	x	x	x	Floppy disk drive select
x	x	x	1	1	0	0	0	Floppy drive 1 head 0
x	x	x	1	1	0	0	1	Floppy drive 1 head 1
x	x	x	1	1	0	1	0	Floppy drive 2 head 0
x	x	x	1	1	0	1	1	Floppy drive 2 head 1
x	x	x	1	1	1	0	0	Floppy drive 3 head 0
x	x	x	1	1	1	0	1	Floppy drive 3 head 1
x	x	x	1	1	1	1	0	Floppy drive 4 head 0
x	x	x	1	1	1	1	1	Floppy drive 4 head 1

3.1.9 Status Register (Read Only)

The status register is used to present command completion information to the host processor and is only valid when BUSY (bit 7) is inactive low. Interrupts issued from the VFW-III are cleared when this register is read. The following table illustrates the bits in the register.

0	Error encountered	1	Not used
2	ECC correction made	3	Data transfer requested
4	Seek complete	5	Write fault
6	Drive ready	7	VFW-III busy

3.1.9.1 Error Encountered (Bit 0)

An active high bit indicates that an error condition exists and that the host processor should next check the error register for further information. This bit is reset when the next command is issued.

3.1.9.2 ECC Correction Made (Bit 2)

This bit indicates with an active high that a correction has been successfully made on the single sector transfer or that one or more corrections have been made during a multiple sector transfer.

3.1.9.3 Data Transfer Requested (Bit 3)

The data transfer request goes active high when the VFW-III requires a sector's worth of data to be transferred between the host and the on-board sector buffer during a Read, Write or Format command.

3.1.9.4 Seek Complete (Bit 4)

Winchester drives force the signal active high when the selected drive's read/write heads are correctly positioned over a track and deactivate it during power-up and during head movement.

When floppy drives are selected, this bit is activated when the size/device/head register is reloaded.

3.1.9.5 Write Fault (Bit 5)

An active high of this bit indicates either that the selected Winchester drive is reporting a write fault condition or that the selected floppy disk drive to be written to is write protected. The command issuing the fault condition will be aborted.

3.1.9.6 Drive Ready (Bit 6)

This bit indicates when active high that the selected Winchester drive is reporting that it is ready to execute a command. Floppy disk drives do not report this status through the interface; so it will be forced active whenever floppies are selected.

3.1.9.7 VFW-III Busy (Bit 7)

Bit 7 must be inactive low for any bit within this register to be valid. An active level indicates that the VFW-III processor is busy and the internal data busy is inaccessible.

3.1.10 Command Register (Write Only)

The command register when written to, initiates the command sequence. All other register describing the command must be enterred before the command is issued. Writing to this location will also clear the interrupt.

3.1.11 TMA Least Significant Byte (Write Only)

This register is used to load the least significant byte (A7-A0) of the memory location where the VFW-III Temporary Master Access is to start. This register must be reloaded prior to any TMA.

3.1.12 TMA Most Significant Byte (Write Only)

This register is used to load the most significant byte (A15-A8) of the memory location where the VFW-III Temporary Master Access is to start. This register must be reloaded prior to any TMA.

3.1.13 TMA Extended Significant Byte (Write Only)

This register is used to load the extended significant byte (A23-A16) of the memory location where the VFW-III Temporary Master Access is to start. This register must be reloaded prior to any TMA.

3.1.14 VFW-III Control Port (Write Only)

The control port is used to supply additional information about the drive selection and the data transfer to take place. Use of these bits is illustrated below.

Bit	Meaning	Bit	Meaning
0	8" full size floppy select	1	Extended head selection
2	Single density select	3	Motor-on mode
4	Temporary master access direction	5	Temporary master access enable
6	External PROM disable	7	Not used

3.1.14.1 8" Full Size Floppy Select (Bit 0)

An active high on this signal is used to select 8" full sized floppy disk drives, and an inactive low selects 5.25" minis. When Winchesters are to be selected, the register bit may be either value.

1 = 8" floppy drives
0 = 5.25" floppy drives
X = 5.25" Winchester operation

3.1.14.2 Extended Head Selection (Bit 1)

If jumper W3 is changed from its factory setting such that pin 2 is connected to common, this signal will be driven onto the Winchester interface in place of the signal REDUCE WRITE CURRENT (J4 pin 2). This is the pin used by Winchester drives containing more than eight read/write heads to extend the head selection up to 16. The size/device/head register will still decode head select bits 2 to 0. This register is used as head select bit 3.

1 = Head numbers 8 to 15
0 = Head numbers 0 to 7

3.1.14.3 Single Density Select (Bit 2)

Bit 2 is used to choose single density or double density floppy disk access and has no effect on Winchester operation.

1 = Single density
0 = Double density
X = Winchester operation

3.1.14.4 Motor-On Mode (Bit 3)

This bit is used to select timing for either the MOTOR ON signal for 5.25" floppies or the HEAD LOAD signal for 8" floppies when jumper W1 is in its factory setting (pin 1 connected to common). Floppy drives requiring a spin down when inactivated need this bit set active high. After 3 seconds without an access, the motor will be turned off. Drives requiring a HEAD LOAD signal, such as full sized floppies, should have this bit set inactive low.

1 = Motor-on timing
0 = Head load timing
X = Winchester operation

3.1.14.5 Temporary Master Access Direction (Bit 4)

S-100/IEEE-696 Temporary Master Accesses (DMA) use this bit to determine if the direction is from the host into the on-board sector buffer or if it is the reverse.

1 = From the sector buffer to the host (read commands)
0 = From the host to the sector buffer (write and format commands)

3.1.14.6 Temporary Master Access Enable (Bit 5)

This bit enables the VFW-III when it needs data from the host for READ, WRITE and FORMAT commands to make a S-100/IEEE-696 Temporary Master Access for the data. The current values in the TMA address registers (BASE+08, +09 and +0A) are used as the starting locations for the host memory buffer. After an

access has occurred, the address registers will be pointing to the next location after the last byte addressed in host memory but may change while idle between commands. For this reason the address registers must be reloaded to the beginning of a block before issuing a command using TMA.

1 = TMA enabled (DMA mode)
0 = TMA disabled (Programmed I/O mode)

3.1.14.7 Extended PROM Disable (Bit 7)

An active high bit in this register will disable the external PROM containing single density format routines. It is advised that this bit be maintained as 0, enabling the external PROM.

1 = Disable external PROM
0 = Enable external PROM

3.2 VFW-III COMMAND SET

The command set for the VFW-III can be broken down into Type 1, Type 2, and Type 3. Type 1 commands involve no data transfers and include Restore, Seek and Diagnostic Test. Type 2 commands involve data transfers from the selected drive to the host's memory and include all Read commands. Type 3 commands involve transfers from the host's memory to the selected drive. All Write commands and the Format command are of Type 3. The commands are illustrated in the table below.

Restore	Type 1	0 0 0 1 S S S S	10h - 1Fh
Seek	Type 1	0 1 1 1 S S S S	70h - 7Fh
Test	Type 1	1 0 0 1 0 0 0 0	90h
Read	Type 2	0 0 1 0 D M L 0	20h - 2Eh
Write	Type 3	0 0 1 1 0 M L 0	30h - 36h
Format	Type 3	0 1 0 1 0 0 0 0	50h

S = Stepping rate:	Defined in Section III, Subsection 3.2.1	
D = Read Interrupt:	0-Programmed I/O	1-DMA
M = Multiple Sectors:	0-Single	1-Multiple
L = Long Mode:	0-Normal	1-Long

3.2.1 Type 1 Commands

Type 1 commands involve no data transfers, but most require stepping rates to be specified. The stepping rates in the following table are used as bit 3 through 0 within the Restore or Seek command and are saved internally for subsequent implied seeks.

Rate Number		B3-B0	Winchester	Floppy
00d	00h	0000b	0.035 ms	0.015 ms
01d	01h	0001b	0.5 ms	1.0 ms
02d	02h	0010b	1.0 ms	2.0 ms
03d	03h	0011b	1.5 ms	3.0 ms
04d	04h	0100b	2.0 ms	4.0 ms
05d	05h	0101b	2.5 ms	5.0 ms
06d	06h	0110b	3.0 ms	6.0 ms
07d	07h	0111b	3.5 ms	7.0 ms
08d	08h	1000b	4.0 ms	8.0 ms
09d	09h	1001b	4.5 ms	9.0 ms
10d	0Ah	1010b	5.0 ms	10.0 ms
11d	0Bh	1011b	5.5 ms	11.0 ms
12d	0Ch	1100b	6.0 ms	12.0 ms
13d	0Dh	1101b	6.5 ms	14.0 ms
14d	0Eh	1110b	7.0 ms	15.0 ms
15d	0Fh	1111b	7.5 ms	16.0 ms

3.2.1.1 Restore Command

A Restore command causes the selected drive to step the read/write head toward track 0 until the drive signal TRK 00 goes active. The stepping rates specified in bits 3 through 0 are saved for subsequent implied seeks and used as the step rate for this command unless a floppy step rate less than 8 ms is given. In that case the value chosen is stored, but the step rate for the restore is set to 8 ms. The cylinder high and cylinder low registers are cleared before stepping begins. After each step is issued, Ready and Write Fault are sampled for their normal levels. If no error conditions occur, the VFW-III will wait up to 16 revolutions for the Seek Complete to go active. The interface signal TRK 00 is next sampled to determine if the read/write heads have found track 0. If not, then a step is produced and the procedure is repeated until either track 0 is found or 1024 steps have been attempted.

Jumper 21 on the VFW-III has been incorporated to correct an inherent problem with the WD1010-00. This LSI chip requires the Seek Complete signal to be deactivated immediately following the rising edge of the step pulse. Many 5.25" Winchester drives deactivate Seek Complete after the falling edge of the step pulse, resulting in the WD1010-00 seeing an apparent early Seek Complete to which it responds with the next step pulse. Therefore, during a restore command on these drives, the chip would issue two steps at a time. Some drives make a "chattering" noise while continually trying to seek past track 0, after starting on an odd numbered track. With jumper 21 connecting pin 1 to common, this may occur if one of these drives is cabled in. If jumper 21 is connecting pin 2 to common, the Seek Complete from the drive is modified on-board to ensure that an early deactivation occurs. This jumper selection allows operation with either type of drive and is therefore the factory setting and recommended choice.

3.2.1.2 Seek Command

Due to the capability of implied seeks for Read and Write commands, the use of the Seek command is limited to positioning the read/write heads for formats, or issuing concurrent seeks to multiple drives using buffered seeks. Since a Seek command does not test the value of Seek Complete before or after the command, several seeks can be initiated and status checked to determine which drive finds its target cylinder first by monitoring bit 4 of the status register. Note that the stepping rate is both used and saved for subsequent implied seeks.

3.2.1.3 Diagnostic Test Command

The power-up self diagnostic test described in Section III, Subsection 3.1.2 can be reissued by use of the Diagnostic Test command. A test starts with the WD1015 and progresses to the WD2797 or until the first error is encountered wherein the error code is posted in the error register but an error is not signified in the status register as would normally be done. Error codes for the internal diagnostics and their meanings are tabulated below.

Error Code	Meaning
0	All diagnostics run and indicating no errors.
1	Error found within the WD2797.
2	Error found within the WD1010
3	Sector buffer error
4	Internal bus or WD1014 error
5	WD1015 error

3.2.2 Type 2 Commands

Commands moving data from the disk drives to the host's memory are of Type 2. These are the two read commands, Read and Readlong. Read commands will activate implied seeks at the last step rate to the target track if nessessary. All other registers describing the transfer must be loaded prior to executing Type 2 commands.

3.2.2.1 Read Command

This is the common Read command used to retrieve data from sectors on Winchesters or floppies. CRC will be checked and eight retries will be made before errors are reported. If ECC is formatted on a hard disk drive, data errors will force verification of the syndrome bytes followed by autocorrection of errors less than six bits long. If two syndromes do not match in eight attempts, errors will be posted and the read terminated. Bit 3 (D) is set active high to disable the interrupt request with every byte. This is used in TMA operations where the interrupt will be asserted after the buffer has been transferred. When using multiple sector

transfers, this mode will always be used. If bit 3 is set inactive low, an interrupt will be asserted for every byte request made while moving data into the sector buffer.

3.2.2.2 Readlong Command

The Readlong command permits recovery of the data field and the four bytes of appended ECC syndrome on Winchesters formatted in ECC mode. This can be used in testing or for host correction beyond the WD1015's limit of five bits.

3.2.3 Type 3 Commands

Transfers from host's memory through the sector buffer to the disk drive are performed by the three Type 3 commands. Implicit seeks will be executed as nessessary for the Write and the Writelong commands.

3.2.3.1 Write Command

Data is stored within the data field of the specified sector using this command. ECC or CRC are automatically appended to the data field as it is being written to the drive. It is recommended that, when using programmed I/O transfers, block moves be made without consulting the data request line of the status register between all bytes.

3.2.3.2 Writelong Command

This command, which can be used to test the ECC recovery, allows the writing of a host-supplied four bytes of syndrome and a data field.

3.2.3.3 Format Command

Before any new disk can be used, it must be formatted with a pattern of ID fields, data fields and gaps. No reads or writes will be completed on unformatted media. The Format command requests, from the host, a full sector of information containing the interleave table. It is important to understand that even if the interleave table is only 18 bytes long, as with nine sectors of 1024 bytes, a full sector must be transferred to the buffer. After the information fills the sector buffer, the track will be formatted according to the specified format after the next index pulse is found.

The interleave table has two entries per sector. The first byte is set to 00H for normal sectors and set to FFH if a sector is to be marked as bad. Sectors marked in this way will issue a bad block error when read to allow the operating system to map the sector request to a reserved sector. The second byte contains the logical sector number and is a function of the desired interleave factor. An example of the first few bytes of the example mentioned above is illustrated with an interleave factor of 2.

```
00   01   00   06   00   02   00   07   FF   03
00   08   00   04   00   09   00   05   xx   xx
xx   xx   xx   xx   xx   xx   xx   xx   xx   xx
```

Note that logical sector 3 (physical sector 5) has been marked as a bad block and that all other locations within the 1024 bytes transferred to the sector buffer are unimportant (marked here with "x" for don't-cares).

SECTION IV
INSTALLATION

4.0 GENERAL

Section IV is intended to assist in the installation of the VFW-III into existing systems. All information involved in the physical integration of the controller is supplied herein. Requirements of the host system, settings and adjustments to the board, and cabling to the disk drives are explained in the following subsections. Note that the board comes calibrated from the factory for standard floppy disk drives and 5 MHz data rate Winchester disk drives; therefore, it is properly calibrated for most applications. The procedure below is included for those exceptional cases requiring recalibration.

4.1 POWER AND COOLING

Section 3 of the IEEE publication, Std 696-1983, specifies the voltage levels of the +8 volt power pins of the S-100 that are to be considered compliant with its standard. Instantaneous minimum voltage must be greater than +7 volts, and instantaneous maximum voltage must not exceed +25 volts. Average maximum voltage must be less than +11 volts. This power is connected to the two voltage regulators of the VFW-III via S-100 pins 1 and 51. Two regulators are used to supply the typical 2.5 amps current required of the +5 volt level on-board, for heat dissipation reasons. At the average maximum voltage level of +11 volts, 15 watts must be dissipated. Two widely separated regulators with heatsinks below the IEEE-696 0.5" height limit allow operation up to +50°C ambient air temperature. Adequate air circulation must be provided to prevent the heatsinks from raising poorly circulated air beyond this maximum temperature.

4.2 CABLES

All cable connectors for the VFW-III are positioned on the top edge of the board between the two voltage regulators. Pin 1 for J2 through J8 is located in the lower left corner of each connector.

J2:	34 Pin	5.25" Floppy Disk Drive Connector
J3:	50 Pin	8" Floppy Disk Drive Connector
J4:	34 Pin	5.25" Winchester Disk Drive Control Connector
J5:	12 Pin	Test Connector
J6:	20 Pin	Device 1; Winchester Disk Drive Data Connector
J7:	20 Pin	Device 2; Winchester Disk Drive Data Connector
J8:	20 Pin	Device 3; Winchester Disk Drive Data Connector

4.2.1 Floppy Disk Drives

The SDSystems VFW-III controls four diverse drives concurrently through two connectors, J2 and J3, which are used to interface with 5.25" and 8" drives, respectively.

Leaving these connectors are 34 and 50 strand cables that attach the drives in a daisy-chain manner. Signals on J2 and J3 are electrically common; so unique device numbers must reside on each cable.

J2: 5.25" Floppy Disk		J3: 8" Floppy Disk	
Pin	Signal	Pin	Signal
6	DS4*	6	TG43*
8	INDEX*	14	SIDE*
10	DS1*	18	HLD*
12	DS2*	20	INDEX*
14	DS3*	22	READY*
16	MOTOR-ON*	26	DS1*
18	DIR*	28	DS2*
20	STEP*	30	DS3*
22	WR DATA*	32	DS4*
24	WR GATE*	34	DIR*
26	TRK 00*	36	STEP*
28	WR PROT*	38	WR DATA*
30	RD DATA*	40	WR GATE*
32	SIDE*	42	TRK 00*
		44	WR PROT*
		46	RD DATA*

4.2.2 Test Connector

Signals required for calibration and trouble-shooting are brought to connector J5 for easy access.

J5: Test Connector		J5: Test Connector	
Pin	Signal	Pin	Signal
1	VCO OUT	2	VCO IN
3	DRUN*	4	WPCD*
5	RDATA	6	RC*
7	T3	8	T2
9	T1	10	TEST*
11	GND	12	GND

4.2.3 Winchester Disk Drives

The VFW-III is designed to interface up to three Winchester hard disk drives supporting the ST506/412 standard interface. Each drive requires the connection of two cables: the 34 pin control cable and the 20 pin data cable. All drives are attached via the control cable in a daisy-chain manner to J4, and each is connected through a unique data cable to J6, J7 or J8.

J4: Winchester Control		J4: Winchester Control	
Pin	Signal	Pin	Signal
2	RWC* or HD3*	20	INDEX*
4	HD2*	22	READY*
6	WR GATE*	24	STEP*
8	SEEK COMPLETE*	26	DS1*
10	TRACK 00*	28	DS2*
12	WR FAULT*	30	DS3*
14	HD0*	34	DIR*
18	HD1*		

J6, J7, J8: Winchester Data		J6, J7, J8: Winchester Data	
Pin	Signal	Pin	Signal
11	GND	12	GND
13	+MFM WR DATA	14	-MFM WR DATA
15	GND	16	GND
17	+MFM RD DATA	18	-MFM RD DATA
19	GND	20	GND

4.3 JUMPERS AND SWITCHES

SDSystems increases the flexibility of the VFW-III with the inclusion of user selectable jumpers and switches. The following subsections define both these electives and the factory settings.

4.3.1 Jumpers

Twenty-one jumpers reside on the VFW-III and are labelled with a "Wnn" specification, where "nn" is the associated number. The following list gives the location, pinout, function and factory setting (preceded by an asterisk) for all jumpers.

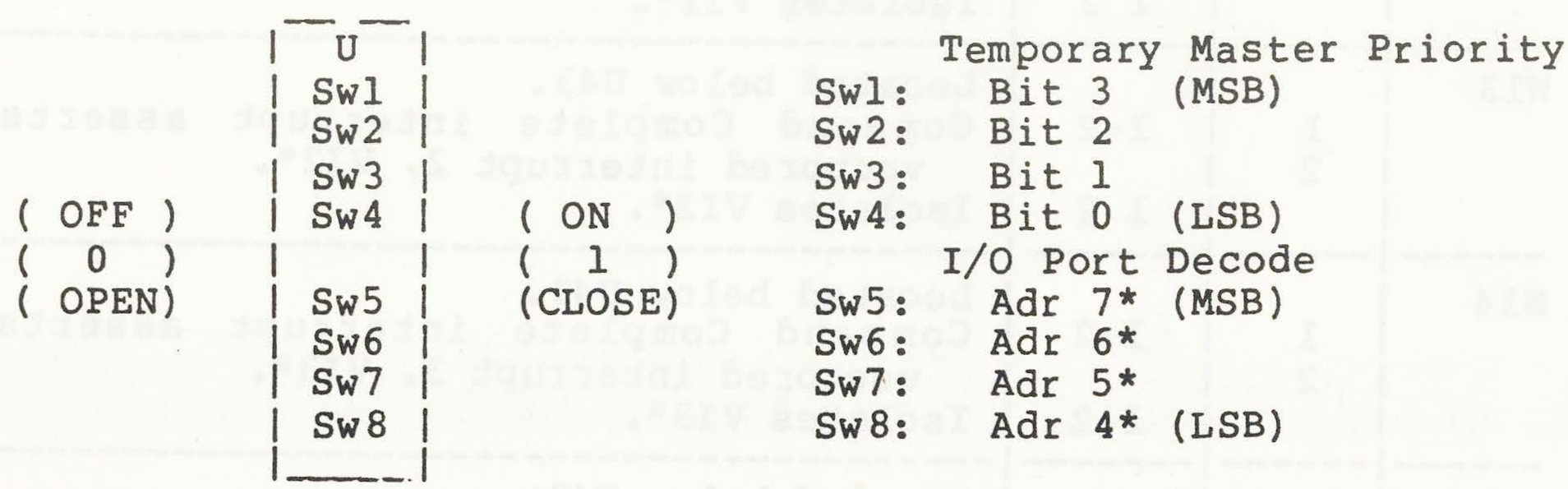
Jumper	Pinout	Setting	Location and Function
W1			Located between U16 and U8.
	1	*C-1	Connects the floppy motor-on line from
	C		the WD1015 to the floppy disks.
	2	C-2	Connects the head load signal from
			the WD2797 to the floppy disks.

Jumper	Pinout	Setting	Location and Function
W2			Located between U8 and U9.
	1	*C-1	Enables write precompensation on floppy
	C		disks starting at cylinder 44.
	2	C-2	Disables write precompensation on all
			floppy disks.
W3			Located between U12 and U13.
	1	*C-1	Connects the reduce write current line
	C		to pin 2 of the Winchester interface.
	2	C-2	Connects the extended bit of the head
			select address to pin 2.
W4			Located between U4 and U15.
	1 C	C-1	Unused at this time.
	2	C-2	Unused at this time.
W5			Located between U21 and U22.
	1	*1-2	Installs the 20 MHz oscillator (in etch).
	2	1 2	Isolates the 20 MHz oscillator.
W6			Located between VR2 and U26.
	1 2	*1-2	Installs VCO input signal (in etch).
		1 2	Isolates VCO input signal.
W7			Located between U28 and U29.
	1	*1-2	Enables WD1015 external addressing (in
	2		etch).
		1 2	Disables use of external PROM.
W8			Located between U16 and U30.
	1	*C-1	External 2716 PROM (in etch).
	2 C	C-2	External 2732 PROM.
	3	C-3	Host bank selection.
W9			Located between U29 and U30.
	1	1-2	Disables internal PROM.
	2	*1 2	Enables internal PROM.
W10			Located between U44 and U45.
	1	C-1	POC* resets the board.
	C	*C-2	RESET* resets the board (in etch).
	2		
W11			Located below U43.
	1	1-2	Command Complete interrupt asserts
	2		vectored interrupt 0, VI0*.
		1 2	Isolates VI0*.

Jumper	Pinout	Setting	Location and Function
W12			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 1, VI1*.
		1 2	Isolates VI1*.
W13			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 2, VI2*.
		1 2	Isolates VI2*.
W14			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 3, VI3*.
		1 2	Isolates VI3*.
W15			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 4, VI4*.
		1 2	Isolates VI4*.
W16			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 5, VI5*.
		1 2	Isolates VI5*.
W17			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 6, VI6*.
		1 2	Isolates VI6*.
W18			Located below U43.
	1 2	1-2	Command Complete interrupt asserts vectored interrupt 7, VI7*.
		1 2	Isolates VI7*.
W19			Located between U38 and U54.
	1 2	*1-2	Installs VCO output signal (in etch).
		1 2	Isolates VCO output signal.
W20			Located between U31 and U32.
	1 2	1-2	Installs 2 wait states on every VFW-III I/O access.
		*1 2	No wait states used.
W21			Located between U11 and U18.
	2 C 1	*C-1	Deactivates the Winchester Seek Complete signal early.
		C-2	Normal Seek Complete timing.

4.3.2 Switches

Input/Output (I/O) port addressing and the Temporary Master priority are selected via the eight position dip switch, SW1.



Switch positions Sw1 through Sw4 are used to select the priority of the VFW-III Temporary Master bus arbitration. The highest priority in the following table is 0F, and 00 is the lowest.

Pri	Sw1	Sw2	Sw3	Sw4
00	OFF	OFF	OFF	OFF
01	OFF	OFF	OFF	ON
02	OFF	OFF	ON	OFF
03	OFF	OFF	ON	ON
04	OFF	ON	OFF	OFF
05	OFF	ON	OFF	ON
06	OFF	ON	ON	OFF
07	OFF	ON	ON	ON

Pri	Sw1	Sw2	Sw3	Sw4
08	ON	OFF	OFF	OFF
09	ON	OFF	OFF	ON
0A	ON	OFF	ON	OFF
0B	ON	OFF	ON	ON
0C	ON	ON	OFF	OFF
0D	ON	ON	OFF	ON
0E	ON	ON	ON	OFF
0F	ON	ON	ON	ON

Switch positions Sw5 through Sw8 are used to select the base address used in the IEEE-696 Standard Input/Output Device Addressing. Sixteen consecutive locations from the base address are decoded by the VFW-III.

I/O	Sw5	Sw6	Sw7	Sw8
00	ON	ON	ON	ON
10	ON	ON	ON	OFF
20	ON	ON	OFF	ON
30	ON	ON	OFF	OFF
40	ON	OFF	ON	ON
50	ON	OFF	ON	OFF
60	ON	OFF	OFF	ON
70	ON	OFF	OFF	OFF

I/O	Sw5	Sw6	Sw7	Sw8
80	OFF	ON	ON	ON
90	OFF	ON	ON	OFF
A0	OFF	ON	OFF	ON
B0	OFF	ON	OFF	OFF
C0	OFF	OFF	ON	ON
D0	OFF	OFF	ON	OFF
E0	OFF	OFF	OFF	ON
F0	OFF	OFF	OFF	OFF

4.4 CALIBRATION

The VFW-III comes calibrated from the factory for standard floppy disk drives and 5 MHz data rate Winchester disk drives; therefore, it is properly calibrated for most applications. The procedure below is included for those exceptional cases requiring recalibration. Floppy adjustments can be made on a single trace oscilloscope, but Winchesters require a dual trace scope for proper results. All signals required for display are conveniently supplied on the 12 pin connector, J5.

4.4.1 Test Mode For The VFW-III

Prior to attempting any calibrations of Section IV, Subsection 4.4.2 or 4.4.3, the following procedure must be executed.

1) Remove the jumper connecting J5 pins 10 and 12 if installed.

2) Assert a RESET pulse on the S-100 bus (pin 75 active low).

3) Output a 01H to the VFW-III I/O port address, BASE+0BH. For factory setting of BASE equaling 50H, output the 01H to port 5BH.

4) Install the jumper connecting J5 pins 10 and 12.

The board is now in test mode wherein some signals are redefined to aid in calibration. For this reason it is imperative that the jumper connecting J5 pins 10 and 12 be removed before normal operation.

4.4.2 Floppy Disk Controller Calibration

Three points must be adjusted for proper operation of the WD2797 Floppy Disk Controller. These components are the trim capacitor C29 and the two trim potentiometers R7 and R8. Test connector pins used in this operation are:

Pin	Signal	Function
7	T3	Used to adjust the read pulse width to 1/8th of the read clock.
8	T2	Used to adjust the write precompensation timing.
9	T1	Used to adjust the center frequency of the internal voltage controlled oscillator.
10	TEST*	WD2797 input to enable test mode.
11	GND	Ground potential reference for the oscilloscope.
12	TESTGND*	A plug connecting pin 10 with this GND; enables test mode.

4.4.2.1 Floppy Disk Write Precompensation

After entering test mode per Section IV, Subsection 4.4.1, connect T2 (J5 pin 8) to a positive-edge triggered oscilloscope. Adjust the multiple revolution trim potentiometer, R7, until the active high pulse width of T2 equals the desired write precompensation value. Remove the jumper connecting J5 pins 10 and 12 before normal VFW-III operation is attempted.

4.4.2.2 Floppy Disk Read Pulse Width

After entering test mode per Section IV, Subsection 4.4.1, connect T3 (J5 pin 7) to a positive-edge triggered oscilloscope. Adjust the trim potentiometer, R8, until the active high pulse width of T3 equals 250 ns. Remove the jumper connecting J5 pins 10 and 12 before normal VFW-III operation is attempted.

4.4.2.3 Floppy Voltage Controlled Oscillator Center Frequency

After entering test mode per Section IV, Subsection 4.4.1, connect T1 (J5 pin 9) to a positive-edge triggered oscilloscope. Adjust the single revolution trim capacitor, C29, with a nonmetallic screwdriver until the period of the 50% duty cycle signal T1 equals 2 microseconds. Remove the jumper connecting J5 pins 10 and 12 before normal VFW-III operation is attempted.

4.4.3 Winchester Voltage Controlled Oscillator

Trim capacitor C47 must be adjusted for proper operation of the Winchester data separator circuit. The following procedure defines how to calibrate the VCO using an oscilloscope, a nonmetallic screwdriver and a cabled-up Winchester drive.

1) Connect the Winchester control cable to J4 and the data cable to J6, J7, or J8 as appropriate for device number 1, 2, or 3, respectively. Turn the Winchester power on and select the drive by writing to the SDH register at the VFW-III I/O BASE+06H (factory setting would be 56H).

Winchester Drive	Connector	SDH Value
1	J6	00
2	J7	08
3	J8	10

2) Connect one lead of a positive-edge triggered oscilloscope to VCO OUT (J5 pin 1) and either a second lead or a voltage meter to VCO IN (J5 pin 2).

3) Using the nonmetallic screwdriver, adjust the single revolution trim capacitor until a frequency of twice the drive's data rate "locks on." This can best be observed by displaying several clock periods; the oscilloscope will display clock periods farthest from the triggered edge as "fuzzy" until the phase-locked loop "locks on," resulting in a sharp image.

4) After the frequency is locked on, tune C47 until the voltage on VCO IN is at 2.5V±0.5V.

5) Connect RDATA and RC (J5 pins 5 and 6) to the oscilloscope and fine tune C47 until the rising edge of RDATA is centered between any two voltage transitions of RC.

The Winchester VCO is now calibrated. Remove the jumper between J5 pins 10 and 12 if not previously done.

4.5 NON-IEEE-696 BUSSES

S-100 busses not complying with the IEEE-696 standard may still control the VFW-III but may not take full advantage of its features. The most obvious change involves the specification of the Temporary Master Access (TMA) priority scheme.

4.5.1 Pin Assignments

The IEEE-696 standard specifies pins 20, 53 and 70 as new ground pins. Some older busses and modified systems may have these pins defined otherwise. If no better compromise exists, care should be taken to cut the connection between the pin and its feedthrough hole to the ground plane. The controller is a six layer board, and careless cutting could damage internal traces.

The factory setting for the reset pin is laid in etch on the solder side of the board at jumper W10. It is located between U44 and U45 and is set to reset the board when pin 75, RESET*, is active low. Removing this etch and connecting the common pad to pad 1 (see Section IV, Subsection 4.4.1) will force the board to be reset when POC* (power-on clear) is asserted low. Most S-100 boards support RESET*; so this change is seldom nessessary.

Pin 25 has been redefined to be pSTVAL* instead of ø2. The VFW-III only asserts this line when acting as a temporary master; this mode probably will not be used on non-IEEE-696 busses. Since this pin is not used as an input, operation in programmed I/O mode is unaffected. In this mode, the eight extra address lines of the TMA addressing circuit are not driven, thereby avoiding any conflicts on their pins.

4.5.2 Data Transfer Modes

The IEEE-696 defines a Direct Memory Access (DMA) scheme and its priority resolution procedure which allow several devices to make DMA transfers without collision. This method is not backward compatible and therefore probably cannot be used on non-IEEE-696 busses. With programmed I/O transfers, whereby all data is transferred through the Central Processing Unit (CPU) between its memory and the disk controller, no bus incompatibilities should exist. All transfers would be simple input/output commands to one of 12 ports.

SECTION V
SPECIFICATIONS

5.0 GENERAL

Section V contains the specifications for proper operation of the VFW-III controller.

5.1 POWER REQUIREMENTS

Section 3 of the IEEE publication, Std 696-1983, specifies the voltage levels of the +8 volt power pins of the S-100 that are to be considered compliant with its standard. Instantaneous minimum voltage must be greater than +7 volts, and instantaneous maximum voltage must not exceed +25 volts. Average maximum voltage must be less than +11 volts. Zero voltage potential or Ground must be supplied on pins 20, 50, 53, 70 and 100. For non-IEEE-696 busses refer to Section IV, Subsection 4.5 for information concerning possible conflicts with the new ground pins. The VFW-III typically requires 2.5 amps at 5 volts.

5.2 ENVIRONMENT

Ambient air temperature may range from 0 to 50o Celsius with a noncondensing, relative humidity between 20 and 80 percent. Adequate airflow must be maintained as discussed in Section IV, Subsection 4.1 to prevent the two voltage regulators from excessively heating local air beyond this maximum value. Busses with the average maximum voltage of 11 volts on the +8 volt line will force the heatsinks to dissipate 15 watts; so proper airflow is important.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A
SELECTED IEEE-696 SPECIFICATION SHEETS

NOTE: For additional information, see the complete document "IEEE Standard 696 Interface Devices."

IEEE-696 Bus Pin List

Pin No.	Signal & Type	Active Level		Description
1	+8 V (B)			Instantaneous minimum greater than 7 V, instantaneous maximum less than 25 V, average maximum less than 11 V.
2	+16 V (B)			Instantaneous minimum greater than 14.5 V, instantaneous maximum less than 35 V, average maximum less than 21.5 V.
3	XRDY (S)	H		One of two ready inputs to the current bus master. The bus is ready when both these ready inputs are true. See pin 72.
4	VIO* (S)	L	OC	Vectored interrupt line 0.
5	VII* (S)	L	OC	Vectored interrupt line 1.
6	VI2* (S)	L	OC	Vectored interrupt line 2.
7	VI3* (S)	L	OC	Vectored interrupt line 3.
8	VI4* (S)	L	OC	Vectored interrupt line 4.
9	VI5* (S)	L	OC	Vectored interrupt line 5.
10	VI6* (S)	L	OC	Vectored interrupt line 6.
11	VI7* (S)	L	OC	Vectored interrupt line 7.
12	NMI* (S)	L	OC	Nonmaskable interrupt.
13	PWRFAIL* (B)	L		Power fail bus signal.
14	TMA3* (M)	L	OC	Temporary master priority bit 3.
15	A18 (M)	H		Extended address bit 18.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
16	A16 (M)	H	Extended address bit 16.
17	A17 (M)	H	Extended address bit 17.
18	SDSB* (M)	L OC	The symbol to disable the 8 status signals.
19	CDSB* (M)	L OC	The signal to disable the 5 control output signals.
20	0 V (B)		Common with pin 100.
21	NDEF		Not to be defined. Manufacturer must specify any use in detail.
22	ADSB* (M)	L OC	The signal to disable the address signals.
23	DODSB* (M)	L OC	The control signal to disable the data output signals. (DO7-0 for 8 bit transfers, ED7-0 and OD7-0 for 16 bit transfers.)
24	ø (B)	A	The master timing signal for the bus.
25	pSTVAL* (M)	L	Status valid strobe.
26	pHLDA (M)	H	A control signal used in conjunction with HOLD* to coordinate bus master transfer operations.
27	RFU		Reserved for future use.
28	RFU		Reserved for future use.
29	A5 (M)	H	Address bit 5.
30	A4 (M)	H	Address bit 4.
31	A3 (M)	H	Address bit 3.
32	A15 (M)	H	Address bit 15 (most significant for nonextended addressing).

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
33	A12 (M)	H	Address bit 12.
34	A9 (M)	H	Address bit 9.
35	DO1 (M)/ED1 (M/S)	H	Data out bit 1, bidirectional data bit 1.
36	DO0 (M)/ED0 (M/S)	H	Data out bit 0, bidirectional data bit 0.
37	A10 (M)	H	Address bit 10.
38	DO4 (M)/ED4 (M/S)	H	Data out bit 4, bidirectional data bit 4.
39	DO5 (M)/ED5 (M/S)	H	Data out bit 5, bidirectional data bit 5.
40	DO6 (M)/ED6 (M/S)	H	Data out bit 6, bidirectional data bit 6.
41	DI2 (S)/OD2 (M/S)	H	Data in bit 2, bidirectional data bit 2.
42	DI3 (S)/OD3 (M/S)	H	Data in bit 3, bidirectional data bit 3.
43	DI7 (S)/OD7 (M/S)	H	Data in bit 7, bidirectional data bit 7.
44	sM1 (M)	H	The status signal which indicates that the current cycle is an op-code fetch.
45	sOUT	H	The status signal identifying the data transfer bus cycle to an output device.
46	sINP (M)	H	The status signal identifying the data transfer bus cycle from an input device.
47	sMEMR (M)	H	The status signal identifying bus cycles which transfer data from memory to a bus master, which are not interrupt acknowledge instruction fetch cycle(s).

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level		Description
48	sHLTA (M)	H		The status signal which acknowledges that a HLT instruction has been executed.
49	CLOCK (B)	A		2 MHz (+0.5%) 40-60% duty cycle. Not required to be synchronous with any other bus signal.
50	0 V (B)			Common with pin 100.
51	+8 V (B)			Common with pin 1.
52	-16 V (B)			Instantaneous maximum less than -14.5 V, instantaneous minimum greater than -35 V, average minimum greater than -21.5 V.
53	0 V (B)			Common with pin 100.
54	SLAVE CLR* (B)	L	OC	A reset signal to reset bus slaves. Must be active with POC* and may also be generated by externl means.
55	TMA0* (M)	L	OC	Temporary master priority bit 0.
56	TMA1* (M)	L	OC	Temporary master priority bit 1.
57	DMA2* (M)	L	OC	Temporary master priority bit 2.
58	sXTRQ* (M)	L		The status signal which requests 16-bit slaves to assert SIXTN*.
59	A19 (M)	H		Extended address bit 19.
60	SIXTN*	L	OC	The signal generated by 16-bit slaves in response to the 16 bit request signal sXTRQ*.
61	A20 (M)	H		Extended address bit 20.
62	A21 (M)	H		Extended address bit 21.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
63	A22 (M)	H	Extended address bit 22.
64	A23 (M)	H	Extended address bit 23.
65	NDEF		Not to be defined signal.
66	NDEF		Not to be defined signal.
67	PHANTOM* (M/S)	L OC	A bus signal which disables normal slave devices and enables phantom slaves--primarily used for bootstrapping systems without hardware front panels.
68	MWRT (B)	H	pWR*-sOUT (logic equation). This signal must follow pWR* by not more than 30 ns.
69	RFU		Reserved for future use.
70	0 V (B)		Common with pin 100.
71	RFU		Reserved for future use.
72	RDY (S)	H OC	See comments for pin 3.
73	INT* (S)	L OC	The primary interrupt request bus signal.
74	HOLD* (S)	L OC	The control signal used in conjunction with pHLDA to coordinate bus master transfer operations.
75	RESET* (B)	L OC	The reset signal to reset bus master devices. This signal must be active with POC* and may also be generated by external means.
76	pSYNC (M)	H	The control signal identifying BS_1.
77	pWR* (M)	L	The control signal signifying the presence of valid data on DO bus or data bus.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
78	pDBIN (M)	H	The control signal that requests data on the DI bus or data bus from the currently addressed slave.
79	AO (M)	H	Address bit 0 (least significant).
80	Al (M)	H	Address bit 1.
81	A2 (M)	H	Address bit 2.
82	A6 (M)	H	Address bit 6.
83	A7 (M)	H	Address bit 7.
84	A8 (M)	H	Address bit 8.
85	Al3 (M)	H	Address bit 13.
86	Al4 (M)	H	Address bit 14.
87	All (M)	H	Address bit 11.
88	DO2 (M)/ED2 (M/S)	H	Data out bit 2, bidirectional data bit 2.
89	DO3 (M)/ED3 (M/S)	H	Data out bit 3, bidirectional data bit 3.
90	DO7 (M)/ED7 (M/S)	H	Data out bit 7, bidirectional data bit 7.
91	DI4 (S)/OD4 (M/S)	H	Data in bit 4 and bidirectional data bit 12.
92	DI5 (S)/OD5 (M/S)	H	Data in bit 5 and bidirectional data bit 13.
93	DI6 (S)/OD6 (M/S)	H	Data in bit 6 and bidirectional data bit 14.
94	DIl (S)/ODl (M/S)	H	Data in bit 1 and bidirectional data bit 9.
95	DI0 (S)/OD0 (M/S)	H	Data in bit 0 (least significant for 8 bit data) and bidirectional data bit 8.

IEEE-696 Bus Pin List (Continued)

Pin No.	Signal & Type	Active Level	Description
96	sINTA (M)	H	The status signal identifying the bus input cycle(s) that may follow an accepted interrupt request presented on INT*.
97	sWO* (M)	L	The status signal identifying a bus cycle which transfers data from a bus master to a slave.
98	ERROR* (S)	L OC	The bus status signal signifying an error condition during present bus cycle.
99	POC* (B)	L	The power-on clear signal for all bus devices; when this signal goes low, it must stay low for at least 10 microseconds.
100	0 V (B)		System ground.

IEEE-696 Bus Layout--Quick Reference

pin 1	+8 V (B)		pin 51	+8 V (B)	
pin 2	+16 V (B)		pin 52	-16 V (B)	
pin 3	XRDY (S)	H	pin 53	0 V	
pin 4	VI0* (S)	L	pin 54	SLAVE CLR* (B)	L
pin 5	VI1* (S)	L	pin 55	TMA0* (M)	L
pin 6	VI2* (S)	L	pin 56	TMA1* (M)	L
pin 7	VI3* (S)	L	pin 57	TMA2* (M)	L
pin 8	VI4* (S)	L	pin 58	sXTRQ* (M)	L
pin 9	VI5* (S)	L	pin 59	A19	H
pin 10	VI6* (S)	L	pin 60	SIXTN* (S)	L
pin 11	VI7* (S)	L	pin 61	A20 (M)	H
pin 12	NMI* (S)	L	pin 62	A21 (M)	H
pin 13	PWRFAIL* (B)	L	pin 63	A22 (M)	H
pin 14	TMA3* (M)	L	pin 64	A23 (M)	H
pin 15	A18 (M)	H	pin 65	NDEF	
pin 16	A16 (M)	H	pin 66	NDEF	
pin 17	A17 (M)	H	pin 67	PHANTOM* (M/S)	L
pin 18	SDSB* (M)	L	pin 68	MWRT (B)	H
pin 19	CDSB* (M)	L	pin 69	RFU	
pin 20	0 V		pin 70	0 V	
pin 21	NDEF		pin 71	RFU	
pin 22	ADSB* (M)	L	pin 72	RDY (S)	H
pin 23	DODSB* (M)	L	pin 73	INT* (S)	L
pin 24	ø (B)	H	pin 74	HOLD* (M)	L
pin 25	pSTVAL* (M)	L	pin 75	RESET* (B)	L
pin 26	pHLDA (M)	H	pin 76	pSYNC	H
pin 27	RFU		pin 77	pWR* (M)	L
pin 28	RFU		pin 78	pDBIN (M)	H
pin 29	A5 (M)	H	pin 79	A0 (M)	H
pin 30	A4 (M)	H	pin 80	A1 (M)	H
pin 31	A3 (M)	H	pin 81	A2 (M)	H
pin 32	A15 (M)	H	pin 82	A6 (M)	H
pin 33	A12 (M)	H	pin 83	A7 (M)	H
pin 34	A9 (M)	H	pin 84	A8 (M)	H
pin 35	DO1 (M)/ED1 (M/S)	H	pin 85	A13 (M)	H
pin 36	DO0 (M)/ED0 (M/S)	H	pin 86	A14 (M)	H
pin 37	A10 (M)	H	pin 87	A11 (M)	H
pin 38	DO4 (M)/ED4 (M/S)	H	pin 88	DO2 (M)/ED2 (M/S)	H
pin 39	DO5 (M)/ED5 (M/S)	H	pin 89	DO3 (M)/ED3 (M/S)	H
pin 40	DO6 (M)/ED6 (M/S)	H	pin 90	DO7 (M)/ED7 (M/S)	H
pin 41	DI2 (S)/OD2 (M/S)	H	pin 91	DI4 (S)/OD4 (M/S)	H
pin 42	DI3 (S)/OD3 (M/S)	H	pin 92	DI5 (S)/OD5 (M/S)	H
pin 43	DI7 (S)/OD7 (M/S)	H	pin 93	DI6 (S)/OD6 (M/S)	H
pin 44	sM1 (M)	H	pin 94	DI1 (S)/OD1 (M/S)	H
pin 45	sOUT (M)	H	pin 95	DI0 (S)/OD0 (M/S)	H
pin 46	sINP	H	pin 96	sINTA (M)	H
pin 47	sMEMR	H	pin 97	sWO* (M)	L
pin 48	sHLTA (M)	H	pin 98	ERROR* (S)	L
pin 49	CLOCK (B)		pin 99	POC* (B)	
pin 50	0V		pin 100	0 V	

APPENDIX B
DISCLAIMER

SDSystems, INC. makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, SDSystems, INC. reserves the right to revise this publication and to make changes from time to time in the content hereof without obligation of SDSystems, INC. to notify any person of such revision or changes.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C
LIMITED WARRANTY

All SDSystems printed circuit board assemblies are warranted for a period of one (1) year from date of invoice to be free from defects of material and workmanship.

Should an SDSystems board fail to perform to specifications, obtain a Return Material Authorization (RMA) number from your distributor or from SDSystems. Include this number in all correspondence and with the returned product. Ship the item prepaid to SDSystems and it will, at our option, be repaired or replaced free of charge provided the unit is received during the warranty period.

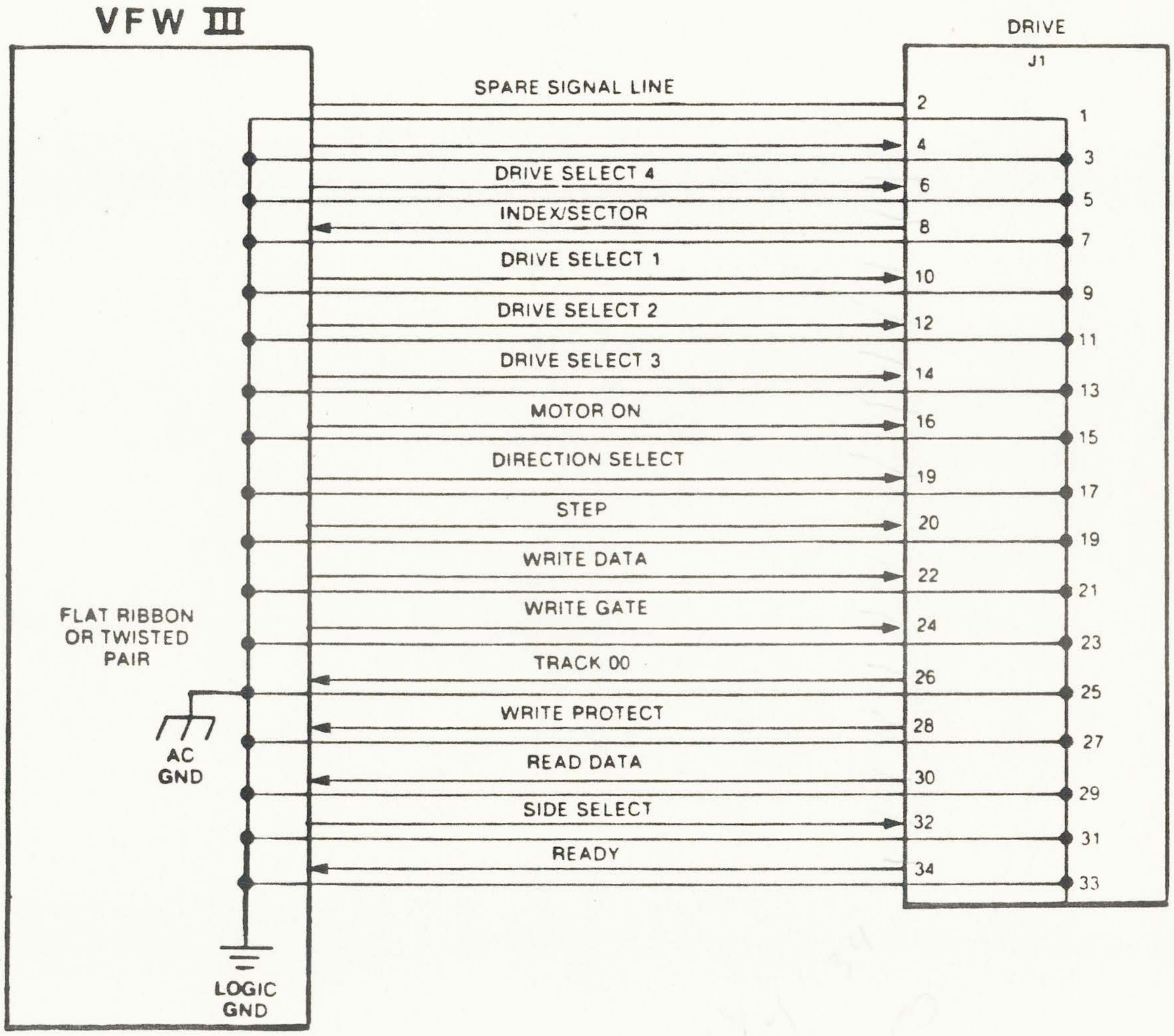
In order to validate this warranty, the enclosed warranty card must be returned to SDSystems. If no warranty card is on file at the time of product return, dated proof of purchase will be required.

This warranty is invalid if product has been misused or improperly modified. Modifications documented in the SDSystems unit publications may be performed without invalidating the warranty. All other modifications will invalidate the warranty. Warranty is limited to replacement of defective parts and no responsibility is assumed for damage to other equipment.

SDSYSTEMS MAKES NO WARRANTIES, GUARANTEES, OR REPRESENTATIONS, EXPRESSED OR IMPLIED, WITH RESPECT TO THE PRODUCTS COVERED HEREBY, EXCEPT AS EXPRESSED HEREIN, AND BUYER EXPRESSLY WAIVES ANY OTHER WARRANTIES, GUARANTEES, OR REPRESENTATIONS INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR USE. SDSYSTEMS NEITHER ASSUMES NOR AUTHORIZES ANY OTHER PERSON TO ASSUME FOR SELLER ANY OTHER LIABILITIES IN CONNECTION WITH THE SALE OF THE PRODUCTS. IN NO EVENT WILL SDSYSTEMS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D
FLOPPY INTERFACE



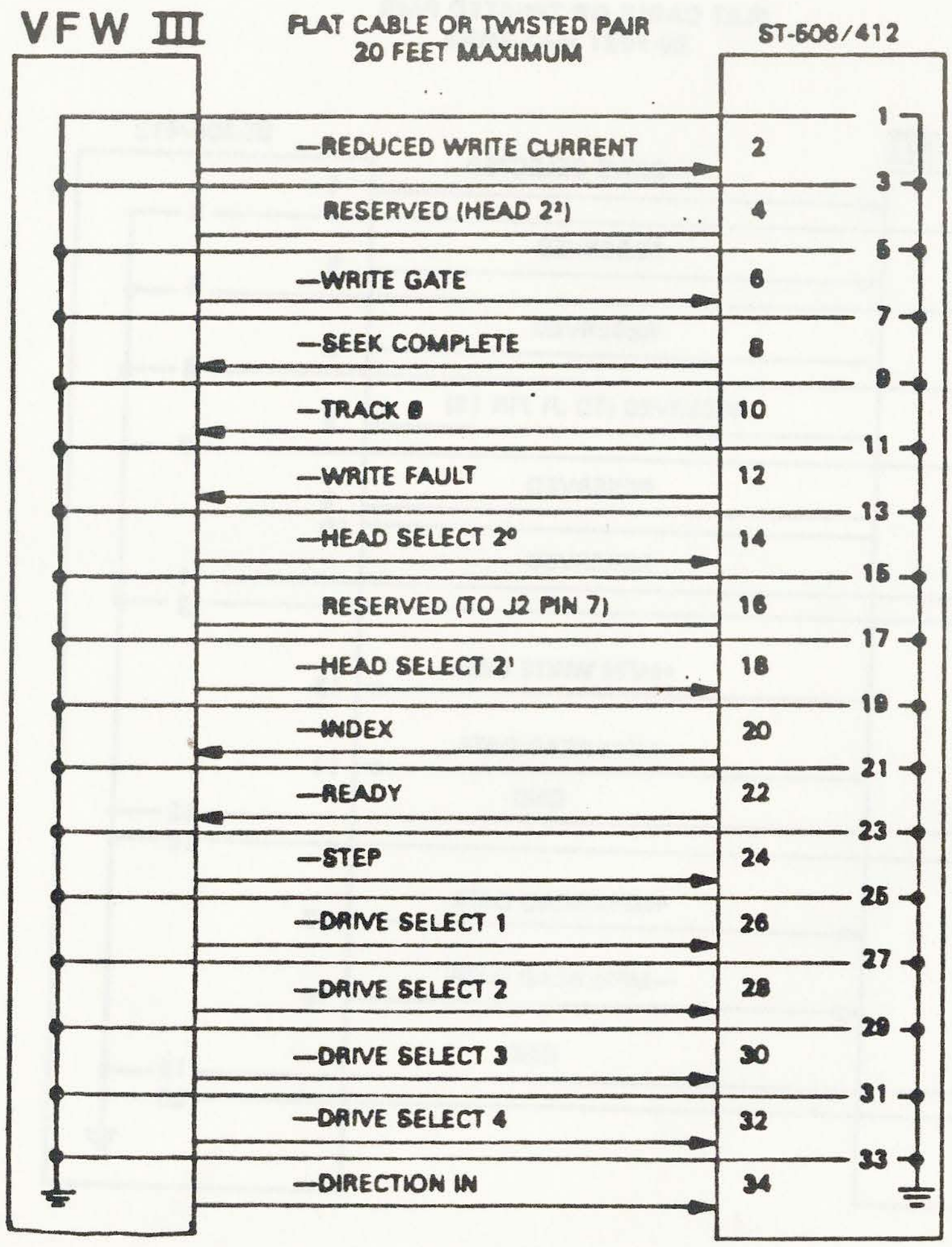
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E
WINCHESTER INTERFACE

Winchester Control Interface

GND RTN PIN	SIGNAL PIN	SIGNAL NAME
1	2	—REDUCED WRITE CURRENT
3	4	RESERVED (Head 2^3 in future products)
5	6	—WRITE GATE
7	8	—SEEK COMPLETE
9	10	—TRACK Ø
11	12	—WRITE FAULT
13	14	—HEAD SELECT 2^0
15	16	RESERVED (TO J2 PIN 7)
17	18	—HEAD SELECT 21
19	20	—INDEX
21	22	—READY
23	24	—STEP
25	26	—DRIVE SELECT 1
27	28	—DRIVE SELECT 2
29	30	—DRIVE SELECT 3
31	32	—DRIVE SELECT 4
33	34	—DIRECTION IN

CONTROL SIGNALS

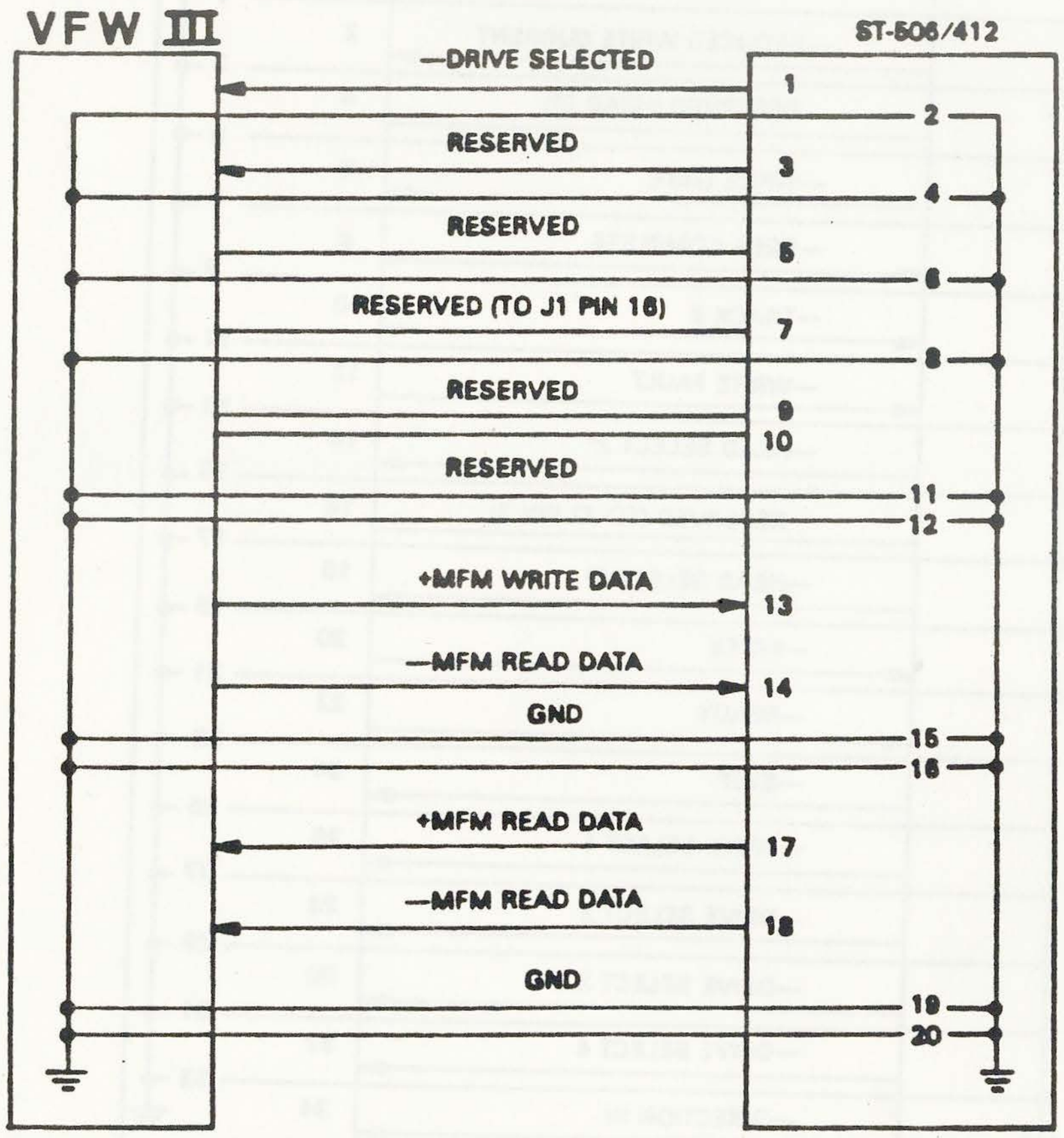


Winchester Data Interface

GND RTN PIN	SIGNAL PIN	SIGNAL NAME
2	1	— DRIVE SELECTED
4	3	RESERVED
6	5	RESERVED
8	7	RESERVED (TO J1 PIN 16)
	9, 10	RESERVED
12	11	GND
	13	+ MFM WRITE DATA
	14	— MFM WRITE DATA
16	15	GND
	17	+ MFM READ DATA
	18	— MFM READ DATA
20	19	GND

DATA SIGNALS

FLAT CABLE OR TWISTED PAIR
20 FEET MAXIMUM



APPENDIX F
PAL SPECIFICATIONS

```
PAL20L10 SPECIFICATION
VFW.001          REV B        4-27-83
SD #7250015

/HRE /HWE HA3 HA2 HA1 HA0 /DMARD /DMAWR DSKRD HCS /HBC GND
WAUP /DOEN WAUP2 /CCLK /CS0 A2 A1 A0 /WE /RE /DIEN VCC

IF(/WAUP2)/CCLK    = HCS*HA3*/HA2*HWE+DMARD
IF(/WAUP2)/CS0     = HCS*/HA3+DMARD
IF(/WAUP2)/A2      = /HA2+DMARD
IF(/WAUP2)/A1      = /HA1+DMARD
IF(/WAUP2)/A0      = /HA0+DMARD
IF(/WAUP)/WE       = HWE*HCS+/DSKRD*DMAWR
IF(/WAUP)/RE       = HRE*HCS+DSKRD*DMARD
IF(VCC)/DIEN       = /WAUP*HRE*HCS*HBC*/DMARD+/DSKRD*DMARD
IF(VCC)/DOEN       = /WAUP*HCS*/HRE*/DMARD+DSKRD*DMARD


PAL16L8 SPECIFICATION
VFW.002    REV A    4-29-83
SD #7250014

SW3 SW2 SW1 SW0 /APRIO /TERM0 NC NC NC GND
NC /T0 /TERM1 /TERM2 /TMA0 /TMA1 /TMA2 /TMA3 /ISMINE VCC

IF(SW3*APRIO)/TMA3   = SW3
IF(SW2*TERM2)/TMA2   = SW2
IF(SW1*TERM1)/TMA1   = SW1
IF(SW0*TERM0)/TMA0   = SW0
IF(VCC)/TERM2        = APRIO*SW3+APRIO*/TMA3
IF(VCC)/TERM1        = TERM2*SW2+TERM2*/TMA2
IF(VCC)/T0           = TERM1*SW1+TERM1*/TMA1
IF(VCC)/ISMINE       = TERM0*SW0+TERM0*/TMA0


PAL16R8 SPECIFICATION
VFW.003       REV0      4-11-83
SD #7250013

CLK DMAENB DRQ WAUP HLDA /HOLD READY /ISMINE NC1 GND
/ENB /APRIO /XFER2 /XFER1 NC2 /DMAWR /DMARD /DLY /PSYNC VCC

APRIO := /WAUP*DMAENB*DRQ*/HOLD*/HLDA+APRIO*/HLDA+ISMINE*HLDA*DRQ
XFER1 := ISMINE*HLDA+XFER2
XFER2 := XFER1*APRIO
PSYNC := DMAENB*DRQ*XFER2*/PSYNC*/DMARD
DLY   := PSYNC
DMARD := PSYNC+DLY+DMAWR
DMAWR := DLY+DMAWR*/READY
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G

FLOPPY DISK DRIVER

THIS PAGE INTENTIONALLY LEFT BLANK

```
;*****************************************************************************;
;                                                                             ;
;      SSS   DDDD    SSS   Y   Y   SSS   TTTTT  EEEEE  M    M   SSS         ;
;     S   S  D   D  S   S  Y   Y  S   S    T    E      MM  MM  S   S        ;
;     S      D   D  S       Y Y   S        T    E      M MM M  S            ;
;      SSS   D   D   SSS     Y     SSS     T    EEE    M    M   SSS         ;
;         S  D   D      S    Y        S    T    E      M    M      S        ;
;     S   S  D   D  S   S    Y    S   S    T    E      M    M  S   S        ;
;      SSS   DDDD    SSS     Y     SSS     T    EEEEE  M    M   SSS         ;
;                                                                             ;
;                                                                             ;
;                                SD SYSTEMS, INC.                             ;
;                                10111 MILLER RD.                             ;
;                                DALLAS, TX 75238                             ;
;                                                                             ;
;                               (214) - 340 - 0303                            ;
;                                                                             ;
;                            COPYRIGHT (C) 1983                               ;
;                                                                             ;
;*****************************************************************************;
;
        Title   'VFWDRVR  floppy disk driver for CP/M Plus 9/18/83'
;
;=============================================================================;
;  V/R  Description                                          Initials-Date    ;
;-----------------------------------------------------------------------------;
;  0.0a Initial test version for CPM3.0                      efj... 8/19/83   ;
;  0.0b All floppy ok- bnk/nbk, dma/pio, int/sta             efj... 9/05/83   ;
;  0.0c Winch up                                             efj... 9/13/83   ;
;  0.0d All but winch sparing                                efj... 9/18/83   ;
;=============================================================================;
;
;
;
        maclib  z80
        maclib  cond
        maclib  vfwconf
        maclib  ports
;
          IF FLOP0   ! public  vfwfd0 !  ENDIF ; FLOP0
          IF FLOP1   ! public  vfwfd1 !  ENDIF ; FLOP1
          IF FLOP2   ! public  vfwfd2 !  ENDIF ; FLOP2
          IF FLOP3   ! public  vfwfd3 !  ENDIF ; FLOP3
;
          IF WINCH00 ! public  vfwd00 !  ENDIF ; WINCH00
          IF WINCH01 ! public  vfwd01 !  ENDIF ; WINCH01
          IF WINCH02 ! public  vfwd02 !  ENDIF ; WINCH02
          IF WINCH03 ! public  vfwd03 !  ENDIF ; WINCH03
          IF WINCH10 ! public  vfwd10 !  ENDIF ; WINCH10
          IF WINCH11 ! public  vfwd11 !  ENDIF ; WINCH11
          IF WINCH12 ! public  vfwd12 !  ENDIF ; WINCH12
          IF WINCH13 ! public  vfwd13 !  ENDIF ; WINCH13
          IF WINCH20 ! public  vfwd20 !  ENDIF ; WINCH20
          IF WINCH21 ! public  vfwd21 !  ENDIF ; WINCH21
          IF WINCH22 ! public  vfwd22 !  ENDIF ; WINCH22
          IF WINCH23 ! public  vfwd23 !  ENDIF ; WINCH23
```

```
                ;
                  IF VFW$INT
                        public  ?vfw$cpl
                  ENDIF ; VFW$INT
                ;
                ;
                ;       bioskrnl.asm externals
                ;
                        extrn   @adrv,@rdrv,@trk,@sect,@dma,@cnt
                        extrn   ?pmsg,?pderr,?conin
                ;
                  IF BANKED
                        extrn   @dbnk
                  IF NOT DMA
                        extrn   @cbnk
                  ENDIF ; NOT DMA
                  ENDIF ; BANKED
                ;
                ;       move.asm externals
                ;
                  IF BANKED
                  IF DMA
                        extrn   ?maptab
                  ELSE  ; NOT DMA
                        extrn   ?bank
                  ENDIF ; NOT DMA
                  ENDIF ; BANKED
                ;
                ;       scb.asm externals
                        extrn   @ermde,@media
                ;
                        page
```

```
                        ;
                                dseg                    ; all of dma driver in dseg except dpb's
                        ;
                        ;       VFW$INIT
                        ;       Currently does nothing.
                        ;
                        vfw$init:
0000 C9                         ret
                        ;
                        ;
                        ;       VFW$LOGIN
                        ;       Initializes the vfw parameters in the dph passed on entry.
                        ;       Step rate is set with a restore according to info in page ff.
                        ;       Floppy- 5"/8" set according to info in page ff.
                        ;       If disk has a vfw type id in trk0 sctrn, the dpb is updated and
                        ;       from the one in the sector.
                        ;       If not, then if floppy, a dft dpb is move in instead; else if
                        ;       winch, an invalid drive is returned to bdos.
                        ;
                        ;       If an error occurs, and no retry requested, an invalid drive error
                        ;       is handed back to bdos.
                        ;       If no retry and user said not to ignore error, then an invalid drive
                        ;       error is returned to bdos; else, login proceeds with whatever is in
                        ;       the login sector buffer.
                        ;
                        ;
                        vfw$login:
                                pushix                  ; save ix
0001+DDE5
0003 D5                         push    d               ; get dph into ix
                                popix
0004+DDE1
                        ;
0006 3E20                       mvi     a,login
0008 327E05                     sta     func
                        ;
                        redo$log:
000B 21B0FF                     lxi     h,step$var      ; get disk info from page ff
000E 3A0000                     lda     @adrv           ; @adrv tells which entry
0011 85                         add     l
0012 6F                         mov     l,a
                        ;
0013 7E                         mov     a,m             ; get it
0014 0F0F0F0F                   rrc ! rrc ! rrc ! rrc   ; high nibble is step rate
0018 E60F                       ani     0000$1111b      ; strip it
                                stx     a,+step         ; save it- locally
001A+DD771A
001D 327A05                     sta     cur$step        ;  and globally
0020 F610                       ori     rs$cmd          ; make restore command
0022 321B06                     sta     rst$cmd         ; save it in restore task table
                        ;
0025 3EF0                       mvi     a,-10h          ; now locate other disk parms
0027 85                         add     l               ; 16 bytes earlier
0028 6F                         mov     l,a             ; hl must remain as is for later use
0029 7E                         mov     a,m
```

```
                        ;
                          IF LOADER
                        ;
                              ani     0011$1111b          ; strip nolog, no128
                              mov     m,a
                        ;
                          ENDIF ; LOADER
                        ;
002A E609                     ani     0000$1001b          ; get motor mode, flop type
002C 47                       mov     b,a                 ; save it in b
                        ;
                              ldx     a,+dfc              ; get last current +dfc
002D+DD7E1D
                              bitx    b$winch,+dtyp       ; check for winch
0030+DDCB1976
                              jrnz    set$r$dfc           ; no massage necessary for winch
0034+2017
                        ;
0036 E6F6                     ani     1111$0110b          ; strip motor mode, flop type
0038 B0                       ora     b                   ; set motor mode, flop type
0039 0E20                     mvi     c,flp$8             ; set up 8" for +dtyp
                              bit     b$5$or$8,b          ; check if 5 or 8 inch- set=8"
003B+CB40
                              jrnz    set$dfc
003D+2008
                        ;
003F 0E10                     mvi     c,flp$5             ; set 5" for +dtyp
                              bit     b$48$flg,m          ; check if 48tpi in 96tpi drive
0041+CB4E
                              jrz     set$dfc
0043+2802
                              setb    b$tpi$48,c          ; mark 48tpi in +dtyp
0045+CBC1
                        ;
                        set$dfc:
                              stx     a,+dfc              ; save new dfc
0047+DD771D
                              stx     c,+dtyp
004A+DD7119
                        ;
                        set$r$dfc:
004D 321A06                   sta     rst$dfc             ; note accum has +dfc
                        ;
                              ldx     a,+sdh              ; get last +sdh
0050+DD7E1C
0053 321606                   sta     rst$sdh             ; use it to restore
                        ;
                              bit     b$nolog,m           ; check if login is to be bypassed
0056+CB7E
0058 C22C01                   jnz     log$xit             ; if set, then skip login
                        ;
                        ;
                        restor:
005B 211106                   lxi     h,rst$tbl           ; pass cmd$act the restore table
005E CDA202                   call    cmd$act             ; and send it
                        ;
```

```
0061 CD0803             call    rdt0sn                  ; read in configuration sector
                        jrz     do$log                  ; if NZ, then hard error
0064+2825
                ;
                  IF LOADER
                ;
                        bitx    b$winch,+dtyp           ; floppy?
                        jrnz    log$err                 ; skip if not
                ;
                        lda     sav$stat                ; skip if busy or not rdy
                        xri     0100$0000b
                        ani     1100$0000b
                        jrnz    log$err
                ;
                        lxi     h,log$flg               ; have tried other type?
                        mvi     a,0ffh                  ; log$flg is 0 if not
                        xra     m
                        mov     m,a
                        ora     a
                        jrnz    chg$typ
                ;
                        lxi     h,drv$var               ; if already tried other type,
                        lda     @adrv                   ;  restore original condition
                        add     l
                        mov     l,a
                ;
                        lda     org$dv
                        mov     m,a
                        mvi     a,10
                        add     l
                        mov     l,a
                        lda     org$sv
                        mov     m,a
                        jmp     log$err
                ;
                chg$typ:
                        lxi     h,drv$var               ; else try 5" if tried 8" or vice versa
                        lda     @adrv
                        add     l
                        mov     l,a
                ;
                        mov     a,m                     ; save original drive var
                        sta     org$dv
                ;
                        bit     b$5$or$8,m
                        jrnz    chg8to5
                ;
                        setb    b$5$or$8,m              ; set as 8"
                        mvi     b,0a0h                  ; force 14 ms step on 8"
                        jmpr    set$stp
                chg8to5:
                        res     b$5$or$8,m              ; set as 5"
                        mvi     b,0d0h                  ; force 20 ms step on 5"
                set$stp:
                        mvi     a,10h
                        add     l
```

```
                        mov     l,a
                        mov     a,m
                        sta     org$sv                  ; save original step var
                        ani     0000$1111b
                        ora     b
                        mov     m,a
                        jmp     redo$log
                ;
                  ENDIF ; LOADER
                ;
                log$err:
0066 2A8005             lhld    err$info                ; pass bios$err status,error,trk,sctr
0069 228205             shld    sav$err$info
006C 210000             lxi     h,0
006F 227805             shld    cur$trk
0072 3A1F06             lda     r0n$sctr
0075 327B05             sta     cur$sctr
                ;
0078 CDC703             call    bios$err                ; else tell user
                        jrnz    inv$dri                 ; and exit if no retry, no ignore
007B+2006
007D B7                 ora     a                       ; else check for retry or ignore
                        jrz     do$log                  ; proceed if ignore
007E+280B
0080 C30B00             jmp     redo$log                ; else try again
                ;
                ;                                         else return invalid drive to bdos
                inv$dri:
                        popix                           ; restore ix
0083+DDE1
0085 E1                 pop     h                       ; pop call ipchl
0086 E1                 pop     h                       ; pop dph pointer
0087 210000             lxi     h,0                     ; set invalid drive
008A C9                 ret                             ; return to bdos
                ;
                ;
                do$log:
008B 218505             lxi     h,s0n$id                ; check if vfw or not vfw disk
008E 3E76               mvi     a,076h
0090 BE                 cmp     m
                        jrnz    not$vfw
0091+2004
0093 23                 inx     h
0094 BE                 cmp     m
                        jrz     set$skw
0095+2826
                ;
                not$vfw:                                ; if non vfw disk,
                        ldx     a,+dtyp                 ;  check for winch
0097+DD7E19
                        bit     b$winch,a
009A+CB77
                        jrnz    log$err                 ; no defaults allowed on winch
009C+20C8
                ;
                ;                                       ; else set SSSD
```

```
                        lxi     h,df5$info
                        bit     b$flp5,a                ; check 5" or 8"
00A1+CB67
                        jrnz    mov$dft                 ; keep if 5"
00A3+2003
00A5 214307             lxi     h,df8$info
                mov$dft:
00A8 7E                 mov     a,m
00A9 328C05             sta     s0n$lskw
                ;
00AC 23                 inx     h
00AD 119505             lxi     d,s0n$dpb
00B0 011100             lxi     b,dft$dpb$lgth
                        ldir
00B3+EDB0
                ;
00B5 11B505             lxi     d,s0n$ddb
00B8 011000             lxi     b,dft$ddb$lgth
                        ldir
00BB+EDB0
                ;
                set$skw:                                ; set up vfw parms in dph
00BD 3A8C05             lda     s0n$lskw                ; set logical skew
                        stx     a,+log$skw
00C0+DD771B
                ;
                        ldx     a,+sdh                  ; get last +sdh
00C3+DD7E1C
00C6 E61F               ani     0001$1111b              ; strip ECC/CRC, sector size
00C8 4F                 mov     c,a                     ; save it
00C9 3ABE05             lda     s0n$sdh                 ; get size info
00CC E6E0               ani     1110$0000b              ; mask all but ECC/CRC, size
00CE B1                 ora     c                       ; combine with rest of +sdh info
                        stx     a,+sdh                  ; and sav it back
00CF+DD771C
                ;
                        ldx     a,+dfc                  ; get last +dfc
00D2+DD7E1D
00D5 E6FB               ani     1111$1011b              ; strip density bit
00D7 4F                 mov     c,a                     ; save old +dfc
00D8 3ABD05             lda     s0n$dfc                 ; retrieve new density info
00DB E604               ani     0000$0100b              ; get density
00DD B1                 ora     c                       ; combine it with the rest
                        stx     a,+dfc                  ; and save it
00DE+DD771D
                ;
00E1 3ABA05             lda     s0n$nhd
                ;
00E4 4F                 mov     c,a                     ; put into c
00E5 AF                 xra     a                       ; clear accum and carry
00E6 0608               mvi     b,8
                chk$bit:                                ; now check if power of 2
                        rlcr    c
00E8+CB01
00EA CE00               aci     0
                        djnz    chk$bit
```

```
 369  00EC+10FA
 370                    ;
 371  00EE FE01                 cpi     1                       ; if only one bit set, then was 2**n
 372                       jrz     sav$nhds                ;  b7 not set -> shft instead of divide
 373  00F0+2802
 374                       setb    b$div$flg,c             ; else set divide flag
 375  00F2+CBF9
 376                    ;
 377                    sav$nhds:
 378                       stx     c,+nhds
 379  00F4+DD711E
 380                    ;
 381  00F7 3AB905               lda     s0n$spt                 ; get sctr/trk
 382                       stx     a,+spt                  ;  and save it
 383  00FA+DD771F
 384                    ;
 385  00FD 2AC105               lhld    s0n$scyl                ; get start cyl offset
 386                       stx     l,+scyll
 387  0100+DD7520
 388                       stx     h,+scylh
 389  0103+DD7421
 390  0106 11FFFF               lxi     d,-1                    ; check if valid drive
 391  0109 B7                   ora     a
 392                       dsbc    d                       ; winch is marked ffffh if invalid
 393  010A+ED52
 394  010C CA8300               jz      inv$dri                 ; return invalid drive if so
 395                    ;
 396                    mov$wpc:
 397  010F 3ABC05               lda     s0n$wpc
 398                       stx     a,+wpc
 399  0112+DD7722
 400                    ;
 401  0115 2A8A05               lhld    s0n$am
 402                       stx     l,+tms
 403  0118+DD7523
 404                       stx     h,+tms+1
 405  011B+DD7424
 406                    ;
 407  011E 219505               lxi     h,s0n$dpb               ; finally, move in new dpb
 408                       ldx     e,+dpbl
 409  0121+DD5E0C
 410                       ldx     d,+dpbh
 411  0124+DD560D
 412  0127 011100               lxi     b,dpb$lgth
 413                       ldir
 414  012A+EDB0
 415                    ;
 416                    ;
 417                    log$xit:
 418                       popix                   ; restore ix
 419  012C+DDE1
 420  012E C9                   ret                     ; and back to setdsk
 421                    ;
 422                    ;
 423                       page
```

```
                        ;
                        ;       VFW$RD
                        ;       Entry point for all floppy and winch disk reads
                        ;       Basically, control is passed on to SET$UP; control returns to
                        ;       here only so ix register is easily preserved.
                        ;
                        vfw$rd:
012F 3E80                       mvi     a,read          ; set read func
                                pushix                  ; free ix for use
0131+DDE5
0133 CD4301                     call    set$up          ;  and jump to common code
                                popix                   ; restore ix to entry status
0136+DDE1
0138 C9                         ret
                        ;
                        ;
                        ;       VFW$RD
                        ;       Entry point for all floppy and winch disk writes.
                        ;       Basically, control is passed on to SET$UP; control returns to
                        ;       here only so ix register is easily preserved.
                        ;
                        vfw$wrt:
0139 3E40                       mvi     a,write         ; set write func
                                pushix                  ; free ix for use
013B+DDE5
013D CD4301                     call    set$up
                                popix                   ; restore ix to entry status
0140+DDE1
0142 C9                         ret
                        ;
                        ;
                        ;       SET$UP
                        ;       Common code for read/write.
                        ;       Handles all single/multio determination and
                        ;       arranges for task table construction or physical i/o as needed.
                        ;       Entries requiring physical i/o return to caller through RD$WRT except
                        ;       for failures during multio track changes.  The latter is simply
                        ;       passed on through to caller.
                        ;
                        set$up:
0143 327E05                     sta     func            ; save func
0146 D5                         push    d               ; put dph pointer into ix
                                popix
0147+DDE1
                        ;
0149 3A7F05                     lda     cnt             ; check if multio in progress
014C B7                         ora     a
                                jrnz    do$mult         ; go do multio stuff if so
014D+201A
                        ;
014F CDC201                     call    bld$tsk         ; else need to set up task table
0152 3A0000                     lda     @cnt            ; check if multio starting
0155 B7                         ora     a               ; nonzero says yes
                                jrz     rd$wrt          ; else go do single read
0156+2832
```

```
                        ;
0158 47                 mov     b,a             ; save @cnt
0159 AF                 xra     a               ;  and reset it
015A 320000             sta     @cnt
                        ;
                        ldx     a,+log$skw      ; if wants multio, check if disk has log skw
015D+DD7E1B
0160 B7                 ora     a               ;  non zero if so
                        jrnz    rd$wrt          ; if so then no multio allowed
0161+2027
                        ;
                set$mult:
0163 78                 mov     a,b             ; save @cnt into cnt
0164 327F05             sta     cnt
                        jmpr    dcr$cnt         ; note that hl has address of cnt
0167+1819
                        ;
                do$mult:
0169 2A7805             lhld    cur$trk         ; check for track change
                        lbcd    @trk
016C+ED4B
016E+0000
0170 B7                 ora     a               ; clear carry
                        dsbc    b
0171+ED42
                        jrz     inr$scnt        ; if no change, then go check counts
0173+2809
                        ;
0175 CD8A01             call    rd$wrt          ; else, read the last track before going on
0178 C0                 rnz                     ;  if error from read, return to bdos
0179 CDC201             call    bld$tsk         ; set up new track, continue with count check
                        jmpr    dcr$cnt
017C+1804
                        ;
                inr$scnt:
017E 210606             lxi     h,tsk$scnt      ; increment #sectors to read
0181 34                 inr     m
                dcr$cnt:
0182 217F05             lxi     h,cnt
0185 35                 dcr     m               ; decrement count remaining
                        jrz     rd$wrt          ;  if zero must be time to read them in
0186+2802
                ;
0188 AF                 xra     a               ;  else set no error and
0189 C9                 ret                     ;   return to bdos
                ;
                page
```

```
                        ;
                        ;       RD$WRT
                        ;       Handles call for physical i/o and arranges for error recovery/reporting
                        ;       if needed.
                        ;
                        ;       Returns
                        ;               A=00:Z          no error (or error but user said to ignore it)
                        ;               A=01:NZ         hard error
                        ;               A=02:NZ         write protected (on writes only)
                        ;               A=ff:NZ         media change
                        ;
                        rd$wrt:
                        ;
                        redo$rw:                        ; redo entry assumes any necessary fixes to
                        ;                                  task table have been done
                        ;
018A 210506                     lxi     h,tsk$tbl       ; rd$wrt's use tsk$tbl for task info
                        ;
018D CDA202                     call    cmd$act         ; go execute, according to task
0190 C8                         rz                      ;  and return to bdos if no error (A=00:Z)
                        ;
0191 CD5B03                     call    proc$err        ; if error go do initial processing of it
                                jrnz    rep$err         ;  if no more to do with it, go report it (NZ)
0194+2012
                        ;
                                bitx    b$winch,+dtyp   ; winch, or floppy?
0196+DDCB1976
                                jrz     flop$err
019A+2807
019C CDA403                     call    chk$spare       ; if winch, try to get a spare
                                jrz     chk$more        ; if successful, go check multio continuation
019F+280F
                                jmpr    rep$err         ; else go report it
01A1+1805

                        flop$err:
01A3 CDA703                     call    chk$media       ; if floppy, go check for media change
                                jrnz    err$xit         ; if so, get out
01A6+2013
                        ;
                        rep$err:
01A8 CDC703                     call    bios$err        ; do bios error routine
                                jrnz    err$xit         ;  get out if NZ (A=01 or A=02)
01AB+200E
                        ;
01AD B7                         ora     a               ; else check if a retry or cont
                                jrnz    redo$rw         ; if A=01, then need a retry on same sector
01AE+20DA
                        ;                               ;  else, A=00
                        chk$more:                       ; need to continue with next sector, if any
01B0 210606                     lxi     h,tsk$scnt      ; is there more to do? (multio)
01B3 35                         dcr     m               ; decrement count
01B4 C8                         rz                      ; if zero, then no more to do (A=00:Z)
01B5 210706                     lxi     h,tsk$sctr      ; else increment sector #
01B8 34                         inr     m
```

```
                        jmpr    redo$rw         ; and go redo
01B9+18CF
                ;
                err$xit:
01BB F5                 push    psw             ; save cpm bdos code
01BC AF                 xra     a               ; always reset multio on errors
01BD 327F05             sta     cnt
01C0 F1                 pop     psw
01C1 C9                 ret                     ; return with whatever error code
                ;
                ;
                ;
                ;       BLD$TSK
                ;       Make tsk$tbl out of cp/m disk info
                ;
                bld$tsk:
01C2 210506             lxi     h,tsk$wpc       ; hl points to first variable in tsk$tbl
                ;
                ;                                       WPC
                        ldx     a,+wpc          ; set tsk$wpc from login info
01C5+DD7E22
01C8 77                 mov     m,a             ; save tsk$wpc
                ;
                ;                                       SCNT
01C9 23                 inx     h               ; advance pointer
01CA 3601               mvi     m,1             ; set sector count to 1
                ;
                ;                                       SCTR
01CC 23                 inx     h               ; advance pointer
                ;
                        ldx     a,+log$skw      ; logical skew?
01CD+DD7E1B
01D0 57                 mov     d,a             ; save it in d
01D1 B7                 ora     a               ; check for 0
01D2 3A0000             lda     @sect           ; put @sect in a
                        jrz     sav$sct         ; skip adjust if not log skew
01D5+2818
                ;
                        mov     b,a             ; save @sect in b
                        ldx     c,+spt          ; get sctr per track
01D8+DD4E1F
01DB 0C                 inr     c               ; increment it
01DC 3E01               mvi     a,1             ; start new sctr at 1
01DE 5F                 mov     e,a             ; set lp$strt
                ;
                adj$sct:
01DF 05                 dcr     b               ; decr sector count
                        jrz     sav$sct         ; loop until 0
01E0+280D
                ;
01E2 82                 add     d               ; add in skew
01E3 B9                 cmp     c               ; check for overflow
                        jrc     adj$sct         ; redo if not
01E4+38F9
                ;
01E6 91                 sub     c               ; else back it up
```

```
 639  01E7 3C              inr     a
 640  01E8 BB              cmp     e               ; check that we haven't been here before
 641                       jrnz    adj$sct         ; redo if not
 642  01E9+20F4
 643  01EB 3C              inr     a               ; else advance new sctr
 644  01EC 1C              inr     e               ; and flag
 645                       jmpr    adj$sct         ; and redo
 646  01ED+18F0
 647                  ;
 648                  sav$sct:
 649  01EF 77                  mov     m,a         ; save tsk$sctr
 650                  ;
 651                  ;                                    CYL
 652  01F0 23                  inx     h           ; advance pointer
 653  01F1 E5                  push    h           ; save tsk$tbl pointer
 654                  ;
 655  01F2 2A0000              lhld    @trk        ; get cp/m track number
 656  01F5 227805              shld    cur$trk     ; go ahead and save for multio checks
 657                  ;
 658                           ldx     a,+nhds     ; get nhds login info
 659  01F8+DD7E1E
 660                           bit     7,a
 661  01FB+CB7F
 662                           jrz     do$shft     ; if b7 not set, then just do shift
 663  01FD+2834
 664                  ;                              else do divide
 665                  ;
 666                  ;                           do 14 bit(max) by 4 bit divide
 667                  ;                                  hl=dividend
 668                  ;                                  de=divisor
 669                  ;                                  ac=quotient
 670                  ;                                  b=count: = 2...13
 671                  ;
 672  01FF E60F                ani     0000$1111b  ; strip divide flag
 673  0201 5F                  mov     e,a         ; divisor = nhds
 674  0202 010010              lxi     b,1000h     ; max times thru divide loop = 13, c=0
 675  0205 51                  mov     d,c
 676                           setb    2,l         ; force at least shl(de,1)
 677  0206+CBD5
 678                  shl$dvd:
 679  0208 05                  dcr     b           ; check divisor for leading 0's
 680  0209 29                  dad     h           ; and adjust loop count accordingly
 681                           jrnc    shl$dvd
 682  020A+30FC
 683  020C 78                  mov     a,b         ; save new loop count
 684  020D 05                  dcr     b           ; divisor is to be shl(nhds,b-1)
 685  020E EB                  xchg
 686                  shl$dvr:
 687  020F 29                  dad     h           ; shift divisor left
 688                           djnz    shl$dvr
 689  0210+10FD
 690  0212 EB                  xchg                ; and put it back in de
 691  0213 2A0000              lhld    @trk        ; get unadultered track
 692  0216 47                  mov     b,a         ; loop count back in b
 693  0217 AF                  xra     a           ; a=0
 694                  ;
```

```
                div$lp:
0218 B7                 ora     a               ; clear carry
0219 227C05             shld    divid           ; save dividend
                        dsbc    d               ; subtract divisor
021C+ED52
                        jrnc    no$car          ; if carry,
021E+3003
0220 2A7C05             lhld    divid           ;  restore dividend
                no$car:
0223 3F                 cmc                     ; complement carry
                        ralr    c               ; rotate carry into quotient
0224+CB11
0226 07                 rlc
                ;
                        srlr    d               ; shift divisor right
0227+CB3A
                        rarr    e               ; note carry is left undetermined
0229+CB1B
                ;
                        djnz    div$lp
022B+10EB
                ;                               ; final quotient is in ac, remainder in l
022D 67                 mov     h,a             ; put final quotient in hl,
022E 7D                 mov     a,l             ;  and remainder in a
022F 69                 mov     l,c
0230 C34802             jmp     sav$cyl
                ;
                ;
                do$shft:                        ; on entry, legal values for a=1,2,4,8 or 16
                                                ; @trk in hl, get hd#, shift trk# to get cyl#
0233 3D                 dcr     a               ; now a = 0000, 0001, 0011, 0111, or 1111
                ;
0234 47                 mov     b,a             ; save shft mask
0235 A5                 ana     l               ; accum now has head#
                shft$lp:
                        srlr    b               ; shift next bit of mask into carry
0236+CB38
                        jrnc    chk$48          ; exit when no bits fall out
0238+3006
                        srlr    h               ; shift cylh right into carry
023A+CB3C
                        rarr    l               ; shift carry into cyll
023C+CB1D
                        jmpr    shft$lp         ; repeat
023E+18F6
                chk$48:
                        bitx    b$tpi$48,+dtyp  ; is this 48tpi disk in 96tpi drive?
0240+DDCB1946
                        jrz     sav$cyl
0244+2802
                        setb    b$48$96,h       ; if so set bit0 of cylh
0246+CBC4
                ;
                sav$cyl:                        ; hl=relative cyl#, a=hd#
                        ldx     e,+scyll        ; get cyl start offset
0248+DD5E20
```

```
                        ldx     d,+scylh
0248+DD5621
024E 19                 dad     d               ; add it to cyl#
024F 44                 mov     b,h             ;  put absolute cyl# into bc
0250 4D                 mov     c,l
                ;
0251 E1                 pop     h               ; get tsk$tbl pointer
0252 71                 mov     m,c             ; save cyll
0253 23                 inx     h
0254 70                 mov     m,b             ; save cylh
                ;
                ;                                       SDH
                ;                                 head# is in accum from CYL routine
0255 23                 inx     h               ; advance pointer
                        ldx     d,+dfc
0256+DD561D
0259 5F                 mov     e,a             ; save head#
025A FE08               cpi     8               ; need hd adr3?
                        jrc     set$hd
025C+3804
025E E607               ani     0000$0111b      ; if so, then strip it
                        setb    b$dfc$hd3,d     ;  and set it
0260+CBCA
            set$hd:
                        orx     +sdh            ; make new sdh
0262+DDB61C
0265 77                 mov     m,a             ; save it
0266 78                 mov     a,b             ; is this cylinder 0?
0267 B1                 ora     c
                        jrnz    sdh$end         ; done if not
0268+201E
                ;
026A E5                 push    h               ; check if user doesn't want a 128 read
026B 21A0FF             lxi     h,drv$var       ; by checking b$no128 in drv$typ byte
026E 3A0000             lda     @adrv           ; for this drive
0271 85                 add     l
0272 6F                 mov     l,a
                        bit     b$no128,m
0273+CB76
0275 E1                 pop     h
                        jrnz    sdh$end         ; use as is if so
0276+2010
                ;
                ;       ldx     b,+dtyp         ; else, get disk type
                ;       bit     b$winch,b       ; winch?
                        bitx    b$winch,+dtyp
0278+DDCB1976
                        jrz     frc$128
027C+2804
027E 7B                 mov     a,e             ; on winch, only head 0 of cyl 0 is 128
027F B7                 ora     a
                        jrnz    sdh$end
0280+2006
            frc$128:
0282 7E                 mov     a,m             ; else retrieve sdh
0283 F660               ori     0110$0000b      ; and mark if for 128 byte sector
```

```
807  0285 77            mov   m,a
808                ;     bit   b$winch,b      ; floppy?
809                ;     jrnz  sdh$end
810                      setb  b$dfc$sd,d     ; if so, then do single density
811  0286+CBD2
812                sdh$end:
813  0288 42             mov   b,d           ; set aside new dfc
814                ;
815                ;                                 DMA
816  0289 23             inx   h             ; advance pointer
817                ;
818                  IF NOT DMA            ; if not dma, then just save @dma
819                ;
820                      lded  @dma          ; get cp/m dma address (16 bits)
821  028A+ED5B
822  028C+0000
823  028E 73             mov   m,e           ; and save it
824  028F 23             inx   h             ; advance pointer
825  0290 72             mov   m,d
826  0291 23             inx   h             ; advance pointer, skip dmah
827                ;
828                  ELSE  ; DMA
829                ;                           else, make 24 bit address from 16 bit address
830                ;
831                  IF NOT BANKED         ; if not banked, always 01xxxxh
832                ;
833                      lded  @dma          ; get cp/m dma address (16 bits)
834                      mov   m,e
835                      inx   h             ; advance pointer
836                      mov   m,d
837                      inx   h             ; advance pointer
838                      mvi   m,01          ; non banked always uses on board ram
839                ;
840                  ELSE  ; BANKED            if banked, have to calculate it
841                ;
842                      push  h             ; save tsk$tbl pointer
843                ;
844                      lxi   d,?maptab+2   ; memory layout table from move module
845                      lda   @dbnk         ; cp/m disk i/o bank
846                      mov   l,a
847                      mvi   h,0
848                      dad   h             ; *4 bytes per bank
849                      dad   h
850                      dad   d             ; address of bank base info
851                ;
852                      mov   a,m           ; get base address into hl
853                      cma                 ; note that least significant 12 bits are
854                      mov   e,a           ; complemented
855                      inx   h             ; also note that don't need 4 ms bits
856                      mov   a,m
857                      cma
858                      mov   h,a
859                      mov   l,e
860                      dad   h             ; shift left 4 times
861                      dad   h
862                      dad   h
```

```
                        dad     h
                        mov     a,h             ; dmah into a
                        mov     d,l             ; dmam into d
                        mvi     e,0             ; dmal into e
                        lhld    @dma            ; get CP/M 16 bit address
                        dad     d               ; add it in
                        xchg                    ; put new 24 bit address into de
                        jrnc    sav$dmal        ; if no carry, no adjust needed
                        inr     a               ; else adjust dmah
                sav$dmal:
                        pop     h               ; retrieve tsk$tbl pointer
                        mov     m,e             ; save dmal
                        inx     h               ; advance pointer
                        mov     m,d             ; save dmam
                        inx     h               ; advance pointer
                        mov     m,a             ; save dmah
                ;
                  ENDIF ; BANKED
                ;
                  ENDIF ; DMA
                ;
                ;                                       DFC
0292 23                 inx     h               ; advance pointer
                                                ;  dfc is in b from sdh routine
                ;
                  IF DMA                        ; if dma then set dma read bit, if needed
                ;
                        lda     func            ; read or write?
                        cpi     read
                        push    psw             ; save rd/wrt status for setting vfw$cmd
                        jrnz    sav$dfc
                        setb    b$dfc$rd,b      ; set dma read if read
                sav$dfc:
                ;
                  ENDIF ; DMA
                ;
0293 70                 mov     m,b             ; save dfc
                ;
                ;                                       CMD
0294 23                 inx     h               ; advance pointer
                ;
                  IF NOT DMA            ; if not dma, then rd/wrt check not done yet
                ;
0295 3A7E05             lda     func            ; read or write?
0298 FE80               cpi     read
                ;
                  ELSE  ; DMA
                ;
                        pop     psw             ; retrieve rd/wrt status
                ;
                  ENDIF ; DMA
                ;
029A 3E24               mvi     a,rd$cmd        ; use read command
                        jrz     sav$cmd
029C+2802
029E 3E34               mvi     a,wrt$cmd       ;  or write command as needed
```

```
                        sav$cmd:
02A0 77                         mov     m,a             ; save tsk$cmd
                        ;
02A1 C9                         ret
                        ;
                        ;
                        ;       CMD$ACT
                        ;       Performs the actual disk operation according to the instructions
                        ;       in the task table.
                        ;       On entry, hl has address of task table to be executed.
                        ;       Completion status is checked and saved for caller.
                        ;       Currently, software checks for not ready errors itself instead of
                        ;       relying on the vfw board.
                        ;
                        ;       hl must be preserved for caller
                        ;
                        ;       Return
                        ;               A=00:Z          no error (err$byt not valid)
                        ;               A=01:NZ         error (err$byt valid)
                        ;
                        cmd$act:
                        ;
                        bsy$chk:
02A2 DB57                       in      vfw$stat        ; wd1015 should never report busy here
                                bit     b$sta$bsy,a
02A4+CB7F
                                jrz     snd$tsk0        ; go on with command if not busy
02A6+280B
                        ;
02A8 3ED1                       mvi     a,bsy$err       ; else set up busy err
                        bsy$nrdy:
02AA 328005                     sta     stat$byt        ; save stat and error
02AD AF                         xra     a
02AE 328105                     sta     err$byt
02B1 3C                         inr     a
02B2 C9                         ret                     ;  and return A=01:NZ
                        ;
                        ;
                        snd$tsk0:                       ; all necessary info is set up in task table
02B3 E5                         push    h               ; preserve task table address for caller
02B4 0E50                       mvi     c,vfw$wpc-1     ; first port to write minus one
02B6 0606                       mvi     b,6             ; six ports before command port
                        outt0:
02B8 0C                         inr     c               ; advance port number
                                outi                    ; update vfw$wpc thru vfw$sdh
02B9+EDA3
                                jrnz    outt0
02BB+20FB
                        ;
                          IF DMA
                        ;
                                inr     c               ; skip command port
                        ;
                                mvi     b,4             ; four dma/control ports
                        outt1:
                                inr     c               ; advance port number
```

```
                        outi                    ; update vfw$dmal thru vfw$dfc
                        jrnz    outt1
                ;
                chk$step:                       ; check current step rate
                        lda     cur$step        ; get current step rate
                        cmpx    +step           ; compare to unit's
                        jrz     snd$tsk1        ; go on if same
                        ldx     a,+step
                        sta     cur$step        ; else save new step
                        ori     sk$cmd          ;  make seek command,
                        out     vfw$cmd         ;  and send it
                ;
                sk$wait:                        ; wait for not busy
                        in      vfw$stat
                        bit     b$sta$bsy,a
                        jrnz    sk$wait
                ;
                snd$tsk1:
                ;
                        mov     a,m             ; get command from table
                ;
                  IF VFW$INT                    ; if using interrupts,
                ;
                        lxi     h,?vfw$cpl      ;  clear completion status
                        mvi     m,0
                ;
                  ENDIF ; VFW$INT
                ;
                        out     vfw$cmd         ; issue command
                ;
                  IF VFW$INT                    ; use interrupt
                ;
                wait$done:
                        mov     a,m             ; completion interrupt will set done$flag
                        ora     a               ;  to 0ffh
                        jrz     wait$done
                ;
                        in      vfw$stat        ; get vfw status
                ;
                  ELSE  ; NOT VFW$INT
                ;
                get$stat:
                        in      vfw$stat        ;.get vfw status
                        bit     b$sta$bsy,a     ;  and wait on wd1015 not busy
                        jrnz    get$stat
                ;                               ;  if multio transfer, must check twice
                        in      vfw$stat        ; get vfw status
                        bit     b$sta$bsy,a     ;  and wait on wd1015 not busy
                        jrnz    get$stat
                ;
                  ENDIF ; NOT VFW$INT
                ;
                ;
                  ELSE  ; NOT DMA
                ;
02BD 5E                 mov     e,m             ; get dma address from task file into de
```

```
1031  02BE 23           inx   h
1032  02BF 56           mov   d,m
1033  02C0 23           inx   h
1034  02C1 23           inx   h                 ; skip dmah
1035  02C2 0E5B         mvi   c,vfw$dfc
1036                    outi                    ; send dfc data
1037  02C4+EDA3
1038              ;
1039              chk$step:                     ; check current step rate
1040  02C6 3A7A05       lda   cur$step          ; get current step rate
1041                    cmpx  +step             ; compare to unit's
1042  02C9+DDBE1A
1043                    jrz   snd$tsk1          ; go on if same
1044  02CC+2810
1045                    ldx   a,+step
1046  02CE+DD7E1A
1047  02D1 327A05       sta   cur$step          ; else save new step
1048  02D4 F670         ori   sk$cmd            ;  make seek command,
1049  02D6 D357         out   vfw$cmd           ;  and send it
1050              ;
1051              sk$wait:                      ; wait for not busy
1052  02D8 DB57         in    vfw$stat
1053                    bit   b$sta$bsy,a
1054  02DA+CB7F
1055                    jrnz  sk$wait
1056  02DC+20FA
1057              ;
1058              snd$tsk1:
1059              ;
1060  02DE 46           mov   b,m               ; pick command up from table
1061  02DF 23           inx   h
1062  02E0 7E           mov   a,m               ; pick up bank flag from table
1063              ;
1064  02E1 CD0000       call  exec$pio          ; go execute the command from common
1065              ;                               accum=cmd, de=dma address
1066              ;                               returns: accum=vfw$stat
1067                ENDIF ; NOT DMA
1068              ;
1069              ;
1070  02E4 328005       sta   stat$byt          ; save status for caller
1071                    bit   b$sta$err,a       ; set error / no error
1072  02E7+CB47
1073  02E9 DB51         in    vfw$err           ; input error byte
1074  02EB 328105       sta   err$byt           ; returning it in err$byt
1075              ;
1076              ;                               This is temporary call to check for not ready
1077  02EE CCF702       cz    rdychk            ; on 8" floppy. It can be removed once the
1078              ;                               wd1015 is fixed to do this check itself.
1079              ;
1080  02F1 E1           pop   h                 ; return with hl preserved
1081              ;
1082  02F2 3E00         mvi   a,00
1083  02F4 C8           rz                      ; if no error, return A=00:Z
1084  02F5 3C           inr   a
1085  02F6 C9           ret                     ; else return A=01:NZ
1086              ;
```

```
1087                    ;
1088                    ;       RDY$CHK
1089                    ;       This checks for not ready status on 8" floppies.
1090                    ;       If a not ready status is found, the error bit in stat$byt is set
1091                    ;       and an aborted command error is set in err$byt
1092                    ;       This routine will be obsolete if the wd1015 is corrected to do
1093                    ;       this check on its own.
1094                    ;
1095                    ;       Return
1096                    ;               Z               no ready error
1097                    ;               NZ              ready error (err$byt valid)
1098                    ;
1099                    rdy$chk:
1100  02F7 218005               lxi     h,stat$byt
1101  02FA 7E                   mov     a,m             ; retrieve status byte
1102                            bit     b$sta$rdy,a     ; check ready bit
1103  02FB+CB77
1104                            jrz     rdy$err         ; go set up not ready error if not ready
1105  02FD+2802
1106                    ;
1107  02FF AF                   xra     a               ; else return (Z)
1108  0300 C9                   ret
1109                    ;
1110                    rdy$err:
1111  0301 F601                 ori     stat$err        ; set err$bit in stat$byt
1112  0303 77                   mov     m,a             ;  and save it
1113  0304 23                   inx     h               ; increment hl to point at err$byt
1114  0305 3604                 mvi     m,abtd$cmd      ; set err$byt to aborted command
1115  0307 C9                   ret                     ; return (NZ)
1116                    ;
1117                    ;
1118                      IF NOT DMA
1119                    ;
1120                    ;
1121                    ;       EXEC$PIO
1122                    ;       Executes the vfw command in programmed io mode from common.
1123                    ;       The command is in reg b. DMA address is in de. Bank flag is in accum.
1124                    ;       Handles the bank switching, waits for completion.
1125                    ;
1126                    ;       Returns vfw$stat in accum.
1127                    ;
1128                            cseg
1129                    ;
1130                    exec$pio:
1131                    ;
1132                      IF VFW$INT
1133                    ;
1134                    ;
1135                      IF BANKED
1136                    ;
1137                            ora     a               ; need bank select?
1138                            push    psw             ; save for exit
1139                            lda     @dbnk           ; get dma bank
1140                            cnz     ?bank           ; ?bank saves all registers but psw
1141                    ;
1142                      ENDIF ; BANKED
```

```
1143                    ;
1144                            mvi     c,vfw$data      ; set up for data port access
1145                    ;
1146                            in      vfw$scnt        ; retrieve sector count
1147                            sta     scnt            ; and save it
1148                    ;
1149                            lxi     h,?vfw$cpl      ; address of completion status
1150                    ;
1151                    ;
1152                    ep$redo:
1153                    ;
1154                            mvi     a,1             ; set scnt to one
1155                            out     vfw$scnt
1156                    ;
1157                            mov     a,b             ; retrieve command
1158                            out     vfw$cmd         ; issue command
1159                    ;
1160                            mvi     m,0             ; reset completion status
1161                    ;
1162                            push    bc              ; save command
1163                    ;
1164                            cpi     wrt$cmd         ; check for read or write
1165                            jrz     wrt$sctr
1166                    ;
1167                    ;
1168                    ;       read or restore
1169                    ;
1170                    rd$wait:
1171                            mov     a,m             ; completion interrupt will set done$flag
1172                            ora     a               ;  to 0ffh
1173                            jrz     rd$wait
1174                    ;
1175                            xchg                    ; get dma address back into hl
1176                    ;
1177                    read$it:                        ; this entry works for read or restore
1178                            in      vfw$stat
1179                            bit     b$sta$drq,a
1180                            jrz     ep$chk          ; note restore should drop on through
1181                    ;
1182                            mvi     b,128
1183                            inir                    ; if drq set, need to read 128 more
1184                    ;
1185                            jmpr    read$it
1186                    ;
1187                    ;
1188                    ;       write
1189                    ;
1190                    wrt$sctr:
1191                            xchg                    ; dma address into hl
1192                    ;
1193                    snd$dat:
1194                            mvi     b,128           ; write 128 at a time
1195                            outir
1196                    ;
1197                            in      vfw$stat
1198                            bit     b$sta$drq,a     ; if drq still asserted,
```

```
1199                    jrnz    snd$dat         ;  then must need another 128 bytes
1200                ;
1201                    xchg                    ; get addr ?vfw$cpl back into hl
1202                ;
1203                wrt$wait:
1204                    mov     a,m             ; completion interrupt will set done$flag
1205                    ora     a               ;  to 0ffh
1206                    jrz     wrt$wait
1207                ;
1208                    xchg                    ; dma address into hl
1209                ;
1210                    in      vfw$stat        ; get status
1211                ;
1212                ep$chk:
1213                    mov     b,a             ; save status byte in b
1214                    bit     b$sta$err,a     ; check for err
1215                    jrnz    ep$end          ; exit if so
1216                    bit     b$sta$rdy,a     ; also for ready
1217                    jrz     ep$end          ; exit if not
1218                ;
1219                    lda     scnt            ; decrement scnt and exit if done
1220                    dcr     a
1221                    jrz     ep$end          ; exit if no more
1222                    sta     scnt            ; else save new scnt
1223                ;
1224                    in      vfw$sctr        ; update sector
1225                    inr     a
1226                    out     vfw$sctr
1227                ;
1228                    xchg                    ; addr ?vfw$cpl into hl
1229                ;
1230                    pop     bc              ; retrieve the command
1231                    jmpr    ep$redo         ; and redo
1232                ;
1233                ep$end:
1234                ;
1235                    pop     psw             ; adjust the stack
1236                ;
1237                  IF BANKED
1238                ;
1239                    pop     psw             ; retrieve bank flag
1240                    lda     @cbnk
1241                    cnz     ?bank           ; and restore the bank if needed
1242                ;
1243                  ENDIF ; BANKED
1244                ;
1245                    mov     a,b             ; return with status in accum
1246                    ret
1247                ;
1248                ;
1249                  ELSE  ; NOT VFW$INT
1250                ;
1251                ;
1252                  IF BANKED
1253                ;
1254                    ora     a               ; need bank select?
```

```
                        push    psw             ; save for exit
                        lda     @dbnk           ; get dma bank
                        cnz     ?bank           ; ?bank saves all registers but psw
                ;
              ENDIF ; BANKED
                ;
0000 78                 mov     a,b             ; get command
0001 D357               out     vfw$cmd         ; issue command
                ;
0003 0E50               mvi     c,vfw$data      ; set up for data port access
0005 EB                 xchg                    ; get dma address into hl
                ;
0006 FE34               cpi     wrt$cmd         ; check for read or write
                        jrz     wrt$sctr
0008+2819
                ;
                ;
                ;       read or restore
                ;
                rd$sctr:
                ;
                rd$wait:
000A DB57               in      vfw$stat        ;  and wait for not busy
                        bit     b$sta$bsy,a
000C+CB7F
                        jrnz    rd$wait
000E+20FA
                ;
                read$it:                        ; this entry works for read or restore
                        bit     b$sta$drq,a
0010+CB5F
                        jrz     rd$end          ; note restore should drop on through
0012+2808
                ;
0014 0680               mvi     b,128
                        inir                    ; if drq set, need to read 128 more
0016+EDB2
                ;
0018 DB57               in      vfw$stat
                        jmpr    read$it
001A+18F4
                ;
                rd$end:
                ;
001C DB57               in      vfw$stat        ; now check if vfw goes busy again
                        bit     b$sta$bsy,a
001E+CB7F
                        jrnz    rd$sctr         ; if so, then multio going on: go do it
0020+20E8
                ;
                ;
                  IF BANKED
                ;
                        mov     b,a             ; save status
                        pop     psw             ; retrieve bank flag
                        lda     @cbnk
```

```
                        cnz     ?bank           ; and restore the bank if needed
                        mov     a,b             ; retrieve status
                ;
                  ENDIF ; BANKED
                ;
0022 C9                 ret                     ; A=vfw$stat
                ;
                ;
                ;       write
                ;
                wrt$sctr:
                ;
                snd$dat:
0023 0680               mvi     b,128           ; write 128 at a time
                        outir
0025+EDB3
                ;
0027 DB57               in      vfw$stat
                        bit     b$sta$drq,a     ; if drq still asserted,
0029+CB5F
                        jrnz    snd$dat         ;  then must need another 128 bytes
002B+20F6
                ;
                wrt$wait:
002D DB57               in      vfw$stat        ; wait on busy to go not busy
                        bit     b$sta$bsy,a
002F+CB7F
                        jrnz    wrt$wait
0031+20FA
                ;
0033 DB57               in      vfw$stat        ; now check if vfw wants another sector
                        bit     b$sta$drq,a
0035+CB5F
                        jrnz    wrt$sctr        ; if so, then go do it
0037+20EA
                ;
                  IF BANKED
                ;
                        mov     b,a             ; save status
                        pop     psw             ; retrieve bank flag
                        lda     @cbnk
                        cnz     ?bank           ; and restore the bank if needed
                        mov     a,b             ; retrieve status
                ;
                  ENDIF ; BANKED
                ;
0039 C9                 ret                     ; A=vfw$stat
                ;
                  ENDIF ; NOT VFW$INT
                ;
                ;
                        dseg
                ;
                  ENDIF ; NOT DMA
                ;
                ;
```

```
1367                    ;       RD$T0Sn
1368                    ;       Read track 0, sector n, head 0 into 128 byte buffer using single
1369                    ;       density, where n=3 for floppies, or n=3+log'unit for winch.
1370                    ;       Note: b$tpi$48 can be ignored since reading cyl 0.
1371                    ;
1372                    ;       return
1373                    ;               A=00:Z          ok, no error from read
1374                    ;               A=01:NZ         error on read
1375                    ;               A=ff:Z          read ok, but had 8" n.rdy to rdy condition
1376                    ;
1377                    rdt0sn:
1378                            ldx     a,+sdh          ; update drive and/or unit type
1379  0308+DD7E1C
1380  030B F660                 ori     0110$0000b      ; force to size 128
1381  030D 322206               sta     r0n$sdh
1382                    ;
1383                            ldx     a,+dfc          ; update dma/floppy control
1384  0310+DD7E1D
1385  0313 F604                 ori     dfc$sdr         ; set sden bit and, if dma, rd bit
1386  0315 322606               sta     r0n$dfc
1387                    ;
1388  0318 3E03                 mvi     a,3             ; set sctr = 3
1389                            bitx    b$winch,+dtyp   ; winch? or floppy?
1390  031A+DDCB1976
1391                            jrz     sav$0ns
1392  031E+2805
1393                            addx    +unit           ;  adjust
1394  0320+DD86FE
1395  0323 E60F                 ani     0000$1111b      ; strip any misc
1396                    sav$0ns:
1397  0325 321F06               sta     r0n$sctr
1398                    ;
1399  0328 211D06               lxi     h,r0n$tbl       ; address of task table
1400  032B CDA202               call    cmd$act         ; execute the command
1401  032E C8                   rz                      ; return to caller if no error (A=00,Z)
1402                    ;                               ;  else accum = 01
1403                            bitx    b$flp8,+dtyp    ; check for 8" floppy
1404  032F+DDCB196E
1405                            jrnz    chk$rdy         ; if 8", go check ready again
1406  0333+2002
1407  0335 B7                   ora     a               ; else a=01, so set NZ
1408  0336 C9                   ret                     ;  and return error (A=01:NZ)
1409                    ;
1410                    chk$rdy:
1411  0337 3A8005               lda     stat$byt        ; else check for not ready err
1412                            bit     b$sta$rdy,a     ; test ready bit
1413  033A+CB77
1414                            jrz     frc$hd0         ; if zero, then not ready
1415  033C+2803
1416  033E 3E01                 mvi     a,01            ; else non zero,
1417  0340 C9                   ret                     ;  so return hard error (A=01:NZ)
1418                    ;
1419                    frc$hd0:                        ; if not ready on 8", then force hd0
1420  0341 3A2606               lda     r0n$dfc         ; retrieve unit's current dfc info
1421  0344 E6FE                 ani     1111$1110b      ; force 5"
1422  0346 322606               sta     r0n$dfc
```

```
1423  0349 CDA202          call    cmd$act         ; redo command, hl=r0n$tbl
1424                ;
1425                ;                               now try 8" read again
1426  034C 3A2606          lda     r0n$dfc         ; retrieve dfc info
1427  034F F601            ori     0000$0001b      ; force 8"
1428  0351 322606          sta     r0n$dfc
1429  0354 CDA202          call    cmd$act         ; redo command
1430                ;
1431  0357 C0              rnz                     ; if error, return it (A=01:NZ)
1432                ;
1433  0358 3EFF            mvi     a,0ffh
1434  035A C9              ret                     ; else return media change (A=ff:Z)
1435                ;
1436                ;
1437                ;       PROC$ERR
1438                ;       Called by RD$WRT.
1439                ;       Save status and error bytes for use by bios$err.
1440                ;       Also save sector # which caused the error.
1441                ;       Update task table info for retry.
1442                ;       Assumes tsk$tbl is the task table being used.
1443                ;       If error was not ready, no update of task table is done.
1444                ;       If sector in sav$sctr is not equal to sector in task table,
1445                ;       update sector and sector count in task table, and adjust dma address
1446                ;
1447                ;       Return
1448                ;               A=00:Z          OK to do CHK$'s (chk$spare or chk$media)
1449                ;               A=01:NZ         Skip CHK$'s, go directly to BIOS$ERR
1450                ;
1451                ;
1452                proc$err:
1453  035B DB53             in      vfw$sctr        ; first save the sector where it happened
1454  035D 327B05           sta     cur$sctr
1455                ;
1456  0360 2A8005           lhld    err$info        ; then save the current error info
1457  0363 228205           shld    sav$err$info    ;  for printing
1458                ;
1459                          bit     b$sta$bsy,l     ; check if its a busy err
1460  0366+CB7D
1461                          jrz     chk$rst         ;  if not then go check ready state
1462  0368+2803
1463                ;
1464  036A AF                 xra     a               ; else mark no CHK$'s
1465  036B 3C                 inr     a
1466  036C C9                 ret                     ; return A=01:NZ
1467                ;
1468                chk$rst:
1469                          bit     b$sta$rdy,l     ; if drive was ready, go update tsk$tbl
1470  036D+CB75
1471                          jrz     skp$fix         ;  then do CHK$'s but skip tsk fix
1472  036F+2831
1473                ;
1474                fix$tsk:
1475  0371 210706             lxi     h,tsk$sctr              ; hl -> sector in task table
1476  0374 47                 mov     b,a                     ;  accum has sector#, save it
1477  0375 96                 sub     m                       ; how many read correctly?
1478  0376 C8                 rz                              ; no update needed if none (A=00:Z)
```

```
1479                    ;
1480  0377 4F           mov     c,a             ; save # read correctly
1481  0378 70           mov     m,b             ; update task table to new start sector
1482  0379 2B           dcx     h               ; hl -> sector count in task table
1483  037A 96           sub     m               ; accum = (#correct - #requested)
1484                    neg                     ; accum = (#requested - #correct)
1485  037B+ED44
1486  037D 77           mov     m,a             ; update task table to new count
1487                    ;
1488                    ldx     a,+sdh          ; get sector size
1489  037E+DD7E1C
1490  0381 070707       rlc ! rlc ! rlc         ; shift sctr sz to bits 1,0
1491  0384 3C           inr     a               ; wrap around 128
1492  0385 E603         ani     0000$0011b      ; strip all but sz info
1493  0387 3C           inr     a               ; now 1=128, 2=256, 3=512, 4=1024
1494  0388 47           mov     b,a             ; for djnz instruction
1495  0389 214000       lxi     h,64                    ; start value in hl
1496                  fix00:
1497  038C 29           dad     h                       ; *2
1498                    djnz    fix00
1499  038D+10FD
1500                  ;
1501                  fix0:
1502  038F EB           xchg                    ; sector size into de
1503  0390 2A0B06       lhld    tsk$dmal        ; 2 lsb dma into hl
1504  0393 3A0D06       lda     tsk$dmah        ; msb dma into a
1505  0396 41           mov     b,c             ; success count into b
1506                  fix1:
1507  0397 19           dad     d               ; advance dma address
1508  0398 CE00         aci     0               ; adjust dmah if needed
1509                    djnz    fix1            ; advance 1 sector for each one read
1510  039A+10FB
1511                  ;
1512  039C 220B06       shld    tsk$dmal        ; and save the new dma address
1513  039F 320D06       sta     tsk$dmah
1514                  ;
1515                  skp$fix:
1516  03A2 AF           xra     a
1517  03A3 C9           ret                     ; return (A=00:Z)
1518                  ;
1519                  ;
1520                  ;       CHK$SPARE
1521                  ;       If appropriate, try to read a replacement for sector
1522                  ;       Skip attempt if busy, not ready, or seek error
1523                  ;       Else, read spare lists, search for replacement, and read it in
1524                  ;       if found.
1525                  ;
1526                  ;       Return
1527                  ;               A=00:Z          Spare found and read without error
1528                  ;               A=01:NZ         Return no replacement
1529                  ;
1530                  chk$spare:
1531                  ;       lda     sav$stat        ; first check if drive problem
1532                  ;       xri     0101$0000b      ; invert ready and seek
1533                  ;       ani     1101$0000b      ; check for busy, not ready, and seek fail
1534                  ;       jrnz    do$srch         ; go on if not
```

```
                        ;
03A4 AF                 xra     a               ; else mark no spare
03A5 3C                 inr     a
03A6 C9                 ret                     ; (A=01:NZ)
                        ;
                        ;do$srch:
                        ;       ldx     a,+sdh          ; first set up tsk tbl to read spare list
                        ;       sta     rsl$sdh
                        ;       ldx     a,+dfc
                        ;       sta     tsk$dfc
                        ;       mvi     a,0ah
                        ;       sta     rsl$sctr
                        ;
                        ;do search....
                        ;
                        ;
                        ;                               de = sy
                        ;       lhld    tsk$scnt
                        ;       push    h
                        ;       lhld    tsk$cyll
                        ;       push    h
                        ;       lda     tsk$sdh
                        ;       push    psw
                        ;       lda     tsk$dfc
                        ;       push    psw
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;       CHK$MEDIA
                        ;       Report media change if current disk's sided/density info doesn't match
                        ;       against the info from login or if 8" floppy goes n.rdy to rdy.
                        ;       after marking @media and +media.
                        ;       Always report no media change if winch.
                        ;       Return with bios$err error and task info unchanged.
                        ;
                        ;       Return
                        ;               A=00:Z          no media change detected
                        ;               A=ff:NZ         media change (@media, +media <- 0ffh)
                        ;
                        chk$media:
                        ;
03A7 CD0803             call    rdt0sn          ; go read in configuration sector
                        ;
03AA FEFF               cpi     0ffh            ; report media chg if 8" n.rdy to rdy (A=ff:NZ)
03AC+2010               jrnz    media$chg
                        ;
03AE DE01               sbi     01              ; if error on s0n, then return no media change
03B0 C8                 rz                      ;  and let caller handle original error
                        ;
03B1 2A8A05             lhld    s0n$mn          ; else check disk time id
                        ldx     e,+tms
03B4+DD5E23
```

```
1591                            ldx     d,+tms+1
1592    03B7+DD5624
1593    03BA AF                 xra     a
1594                            dsbc    d
1595    03BB+ED52
1596    03BD C8                 rz                      ; if still the same, return  (A=0:Z)
1597                    ;
1598                    media$chg:
1599    03BE F6FF               ori     0ffh            ; else return A=ff:NZ
1600                            stx     a,+media        ; also flag +media
1601    03C0+DD770B
1602    03C3 320000             sta     @media          ;  and @media
1603    03C6 C9                 ret
1604                    ;
1605                            page
```

```
1606
1607                    ;
1608                    ;       BIOS$ERR
1609                    ;       Massages status and error bytes for printing.
1610                    ;       If necessary, sets protect for floppy, fault for winch.
1611                    ;       Sets bdos err to 01 or 02 depending on protect bit.
1612                    ;       If @ermde in scb set to quiet mode, passes hard error on to bdos
1613                    ;       else prints ?Pderr, function, error info, and if aborted command,
1614                    ;       status info.
1615                    ;       Makes retry? request, and gets response.
1616                    ;       If no retry, gets ignore? request, and gets response.
1617                    ;
1618                    ;       Return
1619                    ;               A=00:Z          error is to be ignored.
1620                    ;               A=01:Z          retry requested
1621                    ;               A=01:NZ         no retry, return hard error to bdos
1622                    ;               A=02:NZ         no retry, return write protect to bdos
1623                    ;               A=ff:NZ         media change
1624                    ;
1625                    bios$err:
1626                    ;
1627                      IF LOADER
1628                    ;
1629                            mvi     a,01
1630                            sta     bdos$err
1631                    ;
1632                      ELSE  ; NOT LOADER
1633                    ;                                 massage status and error bytes
1634  03C7 1601                 mvi     d,01h           ; mark hard error as default for bdos return
1635                    ;
1636  03C9 3A8205               lda     sav$stat        ; retrieve status register
1637  03CC 4F                   mov     c,a             ; save it in c
1638  03CD E6D0                 ani     1101$0000b      ; pull busy, ready, and seek complete
1639  03CF EE50                 xri     0101$0000b      ; invert ready and seek complete
1640  03D1 47                   mov     b,a             ; save it in b
1641  03D2 3A7E05               lda     func            ; check for rd or wrt
1642  03D5 FE40                 cpi     write
1643  03D7 78                   mov     a,b             ; retrieve new stat$byt
1644                            jrnz    sav$sta         ; if read, leave fault/protect reset
1645  03D8+2012
1646                    ;
1647                            bit     b$sta$fwp,c     ; else check if fault/protect
1648  03DA+CB69
1649                            jrz     sav$sta         ; skip if not
1650  03DC+280E
1651                    ;
1652                            ldx     a,+dtyp         ; check drive type
1653  03DE+DD7E19
1654  03E1 E640                 ani     winch           ; if winch,
1655  03E3 3E20                 mvi     a,0010$0000b    ;  use b6 in stat$byt to mark write fault
1656                            jrnz    set$fwp
1657  03E5+2004
1658                    ;
1659  03E7 3E02                 mvi     a,0000$0010b    ; else use b1 in stat$byt to mark wrt protect
1660  03E9 1602                 mvi     d,02            ;  and set bdos error to write protect
1661                    set$fwp:
```

```
1662  03EB B0            ora   b              ; set the required bit
1663                sav$sta:
1664  03EC 328405        sta   prt$stat       ; and save the new status
1665                ;
1666  03EF 7A            mov   a,d            ; save the bdos return code
1667  03F0 329704        sta   bdos$err
1668                ;
1669  03F3 3A0000        lda   @ermde         ; check for quiet or not
1670  03F6 3C            inr   a
1671  03F7 CA5404        jz    hard$err       ; if so, then return hard error
1672                ;
1673                  ENDIF ; NOT LOADER
1674                ;
1675                ;
1676  03FA 2A0000        lhld  @trk           ; else update @trk,@sect  (multio)
1677  03FD E5            push  h              ; save @trk
1678  03FE 3A0000        lda   @sect          ; save @sect
1679  0401 F5            push  psw
1680  0402 2A7805        lhld  cur$trk        ; cur$trk has track which first caused error
1681  0405 220000        shld  @trk
1682  0408 3A7B05        lda   cur$sctr       ; cur$sctr has sector which first caused error
1683  040B 320000        sta   @sect
1684                ;
1685  040E CD0000        call  ?pderr         ; print message header
1686                ;
1687  0411 F1            pop   psw            ; must restore all in case retry succeeds
1688  0412 320000        sta   @sect          ; restore @sect
1689  0415 E1            pop   h
1690  0416 220000        shld  @trk           ; restore @trk
1691                ;
1692  0419 219C04        lxi   h,func$msgs    ; print function name
1693  041C 3A7E05        lda   func
1694  041F CD7D04        call  ptab$msg
1695                ;
1696  0422 21BC04        lxi   h,err$msgs     ; print all errors
1697  0425 3A8305        lda   sav$err
1698  0428 CD7D04        call  ptab$msg
1699                ;
1700  042B 210F05        lxi   h,stat$msgs    ; and any status msgs
1701  042E 3A8405        lda   prt$stat
1702  0431 F5            push  psw            ; save for busy err check
1703  0432 CD7D04        call  ptab$msg
1704  0435 F1            pop   psw            ; retrieve status
1705                ;
1706                  IF NOT LOADER
1707                ;
1708                      bit   b$sta$bsy,a    ; check for busy err
1709  0436+CB7F
1710                      jrnz  hard$err       ;  no retry request on busy err
1711  0438+201A
1712                ;
1713  043A 215105        lxi   h,retry$msg    ; send retry? msg
1714  043D 3E59          mvi   a,'Y'          ; <return> equals 'Y'
1715  043F CD5904        call  get$rsp
1716                ;
1717  0442 DE59          sbi   'Y'            ; Yes?
```

```
1718                            jrnz    chk$ignor
1719  0444+2003
1720  0446 3E01                 mvi     a,01
1721  0448 C9                   ret                     ;  then return to caller (A=01:Z)
1722                    ;
1723                    chk$ignor:
1724  0449 216105               lxi     h,ignor$msg
1725  044C 3E4E                 mvi     a,'N'           ; <return> equals 'N'
1726  044E CD5904               call    get$rsp
1727                    ;
1728  0451 DE59                 sbi     'Y'             ; Yes?
1729  0453 C8                   rz                      ;  then return (A=00:Z)
1730                    ;
1731                    ;                                 else fall through to return hard error
1732                    ;
1733                      ENDIF ; NOT LOADER
1734                    ;
1735                    hard$err:
1736  0454 3A9704               lda     bdos$err        ; retrieve bdos err
1737  0457 B7                   ora     a               ; set NZ
1738  0458 C9                   ret                     ;  A=(01, 02, or ff):NZ
1739                    ;
1740                    ;
1741                    get$rsp:                        ; message adr in hl, default for <return> in a
1742  0459 327C05               sta     ret$dft
1743  045C CD0000               call    ?pmsg           ; print message
1744                    redo$ci:
1745  045F CD0000               call    ?conin          ; get response
1746  0462 E65F                 ani     05fh            ; force upper case
1747  0464 FE59                 cpi     'Y'
1748                            jrz     echo$it
1749  0466+280B
1750  0468 FE4E                 cpi     'N'
1751                            jrz     echo$it
1752  046A+2807
1753  046C FE0D                 cpi     13
1754                            jrnz    redo$ci
1755  046E+20EF
1756  0470 3A7C05               lda     ret$dft
1757                    echo$it:
1758  0473 F5                   push    psw
1759  0474 219804               lxi     h,err$resp
1760  0477 77                   mov     m,a
1761  0478 CD0000               call    ?pmsg           ; echo response and do crlf
1762  047B F1                   pop     psw
1763  047C C9                   ret                     ; return user response in accum
1764                    ;
1765                    ;
1766                    ptab$msg:               ; table address in hl, bit map in a
1767  047D 110000               lxi     d,0             ; start index at 0
1768                    shift$lp:
1769  0480 B7                   ora     a               ; check if any bits set
1770  0481 C8                   rz                      ;  return when all 0
1771                    ;
1772  0482 87                   add     a               ; shift b7 into carry
1773                            jrnc    no$bit          ;  skip around print if not set
```

```
1774  0483+300E
1775                  ;
1776  0485 F5         no$bit:   push    psw             ; save error byte
1777  0486 E5                   push    h               ; save table addres
1778  0487 D5                   push    d               ; save current index
1779  0488 19                   dad     d               ; make entry pointer
1780  0489 5E                   mov     e,m             ;  and get entry address
1781  048A 23                   inx     h
1782  048B 56                   mov     d,m
1783  048C EB                   xchg
1784  048D CD0000               call    ?pmsg           ; go print it
1785  0490 D1                   pop     d               ;  and restore registers
1786  0491 E1                   pop     h
1787  0492 F1                   pop     psw
1788                  shift$lp:
1789  0493 1C                   inr     e               ; advance index
1790  0494 1C                   inr     e
1791                            jmpr    shift$lp
1792  0495+18E9
1793                  ;
1794                  ;
1795                  ;       BIOS$ERR MESSAGES
1796                  ;
1797  0497 00         bdos$err:       db      0       ; scratch space for setting up bdos err ret
1798                  ;
1799  0498 000A0D00   err$resp:       db      0,10,13,0
1800                  ;
1801  049C A204AA04   func$msgs       dw      read$msg,write$msg
1802  04A0 B304                       dw      login$msg
1803                  ;
1804  04A2 2C20526561read$msg:        db      ', Read,',0
1805  04AA 2C20577269write$msg:       db      ', Write,',0
1806  04B3 2C204C6F67login$msg:       db      ', Login,',0
1807                  ;
1808                  err$msgs:
1809  04BC CC04               dw      err$7
1810  04BE D804               dw      err$6
1811  04C0 7705               dw      null$msg
1812  04C2 E204               dw      err$4
1813  04C4 7705               dw      null$msg
1814  04C6 EB04               dw      err$2
1815  04C8 FD04               dw      err$1
1816  04CA 0505               dw      err$0
1817                  ;
1818  04CC 2042616420err$7:   db      ' Bad Block,',0
1819  04D8 204543432Ferr$6:   db      ' ECC/CRC,',0
1820  04E2 2049442045err$4:   db      ' ID Err,',0
1821  04EB 2041626F72err$2:   db      ' Aborted Command,',0
1822  04FD 2054523030err$1:   db      ' TR000,',0
1823  0505 2044414D20err$0:   db      ' DAM Err,',0
1824                  ;
1825                  stat$msgs:
1826  050F 1F05               dw      stat$7
1827  0511 2605               dw      stat$6
1828  0513 3205               dw      stat$5
1829  0515 3A05               dw      stat$4
```

```
1830  0517 7705              dw      null$msg
1831  0519 7705              dw      null$msg
1832  051B 4105              dw      stat$1
1833  051D 7705              dw      null$msg
1834                  ;
1835  051F 2042555359stat$7: db      ' BUSY,',0
1836  0526 204E6F7420stat$6: db      ' Not Ready,',0
1837  0532 204661756Cstat$5: db      ' Fault,',0
1838  053A 205365656Bstat$4: db      ' Seek,',0
1839  0541 2057726974stat$1: db      ' Write Protect,',0
1840                  ;
1841  0551 2052657472retry$msg:      db      ' Retry (Y/n) ? ',0
1842  0561 0749676E6Fignor$msg:      db      7,'Ignore error (N/y) ? '
1843  0577 00         null$msg:      db      0
1844                  ;
1845                          page
```

```
1846
1847                    ;
1848                    ;
1849                    ;
1850                    ;
1851  0050 =            vfw$bas         equ     P$vfw
1852                    ;
1853  0050 =            vfw$data        equ     vfw$bas+00h     ; data port
1854  0051 =            vfw$wpc         equ     vfw$bas+01h     ; write precompensation cylinder
1855  0051 =            vfw$err         equ     vfw$wpc         ; error return
1856  0052 =            vfw$scnt        equ     vfw$bas+02h     ; # sectors to rd/wrt
1857  0053 =            vfw$sctr        equ     vfw$bas+03h     ; starting sector #
1858  0054 =            vfw$cyll        equ     vfw$bas+04h     ; cylinder low
1859  0055 =            vfw$cylh        equ     vfw$bas+05h     ; cylinder high (only b1,b0 used)
1860  0056 =            vfw$sdh         equ     vfw$bas+06h     ; size, disk type, unit/head
1861  0057 =            vfw$cmd         equ     vfw$bas+07h     ; command
1862  0057 =            vfw$stat        equ     vfw$cmd         ; status
1863  0058 =            vfw$dmal        equ     vfw$bas+08h     ; 24bit dma low
1864  0059 =            vfw$dmam        equ     vfw$bas+09h     ; 24bit dma mid
1865  005A =            vfw$dmah        equ     vfw$bas+0ah     ; 24bit dma high
1866  005B =            vfw$dfc         equ     vfw$bas+0bh     ; dma/floppy control
1867                    ;
1868                    ;
1869  0010 =            rs$cmd          equ     0001$0000b      ; vfw restore command
1870  0070 =            sk$cmd          equ     0111$0000b      ; vfw seek command
1871                    ;
1872                      IF DMA
1873                    ;
1874                    rd$cmd          equ     0010$1100b      ; vfw read sector command
1875                    wrt$cmd         equ     0011$0100b      ; vfw write sector command
1876                    f$dfc$dft       equ     0010$0000b      ; default +dfc
1877                    dfc$sdr         equ     0001$0100b      ; single density dma read
1878                    ;
1879                      ELSE  ; NOT DMA
1880                    ;
1881  0024 =            rd$cmd          equ     0010$0100b      ; vfw read sector command
1882  0034 =            wrt$cmd         equ     0011$0100b      ; vfw write sector command
1883  0000 =            f$dfc$dft       equ     0000$0000b      ; default +dfc
1884  0004 =            dfc$sdr         equ     0000$0100b      ; single density read
1885                    ;
1886                      ENDIF ; NOT DMA
1887                    ;
1888  0080 =            read            equ     1000$0000b      ; note CP/M read call
1889  0040 =            write           equ     0100$0000b      ; note CP/M write call
1890  0020 =            login           equ     0010$0000b      ; note CP/M login call
1891                    ;
1892  0004 =            b$dfc$rd        equ     4               ; bit- dfc port read enable
1893  0002 =            b$dfc$sd        equ     2               ; bit- dfc port single density bit
1894  0001 =            b$dfc$hd3       equ     1               ; bit- dfc port fourth head addr line
1895                    ;
1896  0007 =            b$sta$bsy       equ     7               ; bit- sta port busy
1897  0006 =            b$sta$rdy       equ     6               ; bit- sta port ready
1898  0005 =            b$sta$fwp       equ     5               ; bit- sta port write fault/protect
1899  0003 =            b$sta$drq       equ     3               ; bit- sta port data request
1900  0000 =            b$sta$err       equ     0               ; bit- sta port error
1901                    ;
```

```
1902  0007 =        b$nolog         equ     7               ; bit- set -> skip login
1903  0006 =        b$no128         equ     6               ; bit- set -> skip 128 rd on cyl 0
1904  0001 =        b$48$flg        equ     1               ; bit- set -> 48tpi in 96tpi drive
1905  0000 =        b$5$or$8        equ     0               ; bit- set -> 8" floppy
1906                ;
1907  0007 =        b$div$flg       equ     7               ; bit- set -> div, not shift
1908                ;
1909  0000 =        b$48$96         equ     0               ; bit- cylh- set -> do 48 in 96 tpi drv
1910                ;
1911  0010 =        flp$5           equ     0001$0000b      ; for +dtyp- 5" floppy
1912  0020 =        flp$8           equ     0010$0000b      ; for +dtyp- 8" floppy
1913  0030 =        flop            equ     0011$0000b      ; for +dtyp- 5" or 8" floppy
1914  0040 =        winch           equ     0100$0000b      ; for +dtyp- winch
1915                ;
1916  0006 =        b$winch         equ     6               ; bit- +dtyp- winch
1917  0005 =        b$flp8          equ     5               ; bit- +dtyp- 8" flop
1918  0004 =        b$flp5          equ     4               ; bit- +dtyp- 5" flop
1919  0000 =        b$tpi$48        equ     0               ; bit- +dtyp- 48tpi in 96tpi drive
1920                ;
1921  0080 =        stat$bsy        equ     1000$0000b      ; for status busy check
1922  0001 =        stat$err        equ     0000$0001b      ; for setting error in status
1923  00D1 =        bsy$err         equ     1101$0001b      ; for reporting busy error
1924  0011 =        nrdy$err        equ     0001$0001b      ; for reporting not ready error
1925  0004 =        abtd$cmd        equ     0000$0100b      ; for setting error to aborted cmd
1926                ;
1927  FFA0 =        drv$var         equ     0ffa0h          ; b$nolog,b$no128,b$48$flg,b$5$or$8
1928  FFB0 =        step$var        equ     0ffb0h          ; step rates and relative position
1929                ;
1930  0578 0000     cur$trk:        dw      0               ; for multio to watch for track chg
1931  057A 00       cur$step:       db      0               ; last step rate sent to vfwiii
1932  057B 00       cur$sctr:       db      0               ; save sector# on which error occurred
1933  057C 0000     divid:          dw      0               ; temporary for divide routine
1934  057C =        ret$dft equ     divid                   ; temporary for get$rsp routine
1935  057E 00       func:           db      0               ; flag which cp/m function
1936  057F 00       cnt:            db      0               ; multio flag/countdown
1937                ;
1938                err$info:
1939  0580 00       stat$byt:       db      0               ; status and error are returned
1940  0581 00       err$byt:        db      0               ;  via these bytes
1941                ;
1942                sav$err$info:
1943  0582 00       sav$stat:       db      0               ; bios$err saves stat and err in
1944  0583 00       sav$err:        db      0               ;  these bytes
1945                ;
1946  0584 00       prt$stat:       db      0               ; status is printed from this byte
1947                ;
1948                ;
1949                s0n$buf:                                ; buffer for track 0 sector 3
1950                ;
1951  0585          s0n$id:         ds      2               ; set to 7676h identifies vfw disk
1952  0587          s0n$dt:         ds      2               ; set to create date (cpm date)
1953  0589          s0n$hr:         ds      1               ; set to create hour
1954  058A          s0n$mn:         ds      1               ; set to create min
1955  058B          s0n$sc:         ds      1               ; set to create sec
1956  058C          s0n$lskw:       ds      1               ; logical skew factor
1957  058D          s0n$pskw:       ds      1               ; physical skew factor
```

```
1958  058E            s0n$tskw:   ds   1          ; track change skew
1959  058F            s0n$rel:    ds   1          ; set to version# created under
1960  0590            s0n$sp0:    ds   5          ; spare
1961                  ;
1962  0595            s0n$dpb:    ds   17         ; disk parameter block
1963  05A6            s0n$sp1:    ds   15         ; spare
1964                  ;
1965                  s0n$ddb:                    ; disk definition block
1966  05B5            s0n$bps:    ds   2          ; bytes per sector
1967  05B7            s0n$ncyl:   ds   2          ; #cylinders per logical disk
1968  05B9            s0n$spt:    ds   1          ; #sctrs per track
1969  05BA            s0n$nhd:    ds   1          ; #heads
1970  05BB            s0n$sr:     ds   1          ; step rate- needless
1971  05BC            s0n$wpc:    ds   1          ; write precomp
1972  05BD            s0n$dfc:    ds   1          ; dma/floppy control
1973  05BE            s0n$sdh:    ds   1          ; size, disk#, head#
1974  05BF            s0n$gap:    ds   1          ; format gap
1975  05C0            s0n$skw:    ds   1          ; physical skew
1976  05C1            s0n$scyl:   ds   2          ; start cylinder offset
1977  05C3            s0n$sp2:    ds   2          ; spare
1978                  ;
1979  05C5            s0n$lbl:    ds   16         ; disk label
1980                  ;
1981                  s0n$wcl:                    ; winch start cyl's list
1982  05D5            s0n$nwd:    ds   1          ; #winch logical disks
1983  05D6            s0n$ws0:    ds   15         ; disk0 strt cyl, etc.
1984                  ;
1985  05E5            s0n$sp5:    ds   (s0n$buf+128)-s0n$sp5
1986                  ;
1987                  ;
1988                  tsk$tbl:                    ; task file image for rd/wrt
1989  0605 00         tsk$wpc:    db   0          ; write precompensation cylinder
1990  0606 00         tsk$scnt:   db   0          ; # sectors to rd/wrt
1991  0607 00         tsk$sctr:   db   0          ; starting sector #
1992  0608 00         tsk$cyll:   db   0          ; cylinder low
1993  0609 00         tsk$cylh:   db   0          ; cylinder high (only b1,b0 used)
1994  060A 00         tsk$sdh:    db   0          ; size, disk type, unit/head
1995  060B 00         tsk$dmal:   db   0          ; 24bit dma low
1996  060C 00         tsk$dmam:   db   0          ; 24bit dma mid
1997  060D 00         tsk$dmah:   db   0          ; 24bit dma high
1998  060E 00         tsk$dfc:    db   0          ; dma/floppy control
1999  060F 00         tsk$cmd:    db   0          ; command
2000                    IF NOT DMA
2001  0610 FF         tsk$bflg:   db   0ffh       ; read/writes need bank select
2002                    ENDIF ; NOT DMA
2003                  ;
2004                  ;
2005                  rst$tbl:                    ; task file image for restore
2006  0611 4D         rst$wpc:    db   77         ; **write precompensation cylinder
2007  0612 00         rst$scnt:   db   0          ; **# sectors to rd/wrt
2008  0613 00         rst$sctr:   db   0          ; **starting sector #
2009  0614 00         rst$cyll:   db   0          ; **cylinder low
2010  0615 00         rst$cylh:   db   0          ; **cylinder high (only b1,b0 used)
2011  0616 00         rst$sdh:    db   0          ; size, disk type, unit/head
2012  0617 00         rst$dmal:   db   0          ; **24bit dma low
2013  0618 00         rst$dmam:   db   0          ; **24bit dma mid
```

```
2014  0619 00     rst$dmah:   db   0              ; **24bit dma high
2015  061A 00     rst$dfc:    db   0              ; dma/floppy control
2016  061B 10     rst$cmd:    db   10h            ; command
2017                IF NOT DMA
2018  061C 00     rst$bflg:   db   000h           ; restore don't need bank select
2019                ENDIF ; NOT DMA
2020              ;
2021              ;
2022              r0n$tbl:                        ; task file image for rd/wrt
2023  061D 50     r0n$wpc:    db   50h            ; write precompensation cylinder
2024  061E 01     r0n$scnt:   db   1              ; # sectors to rd/wrt
2025  061F 03     r0n$sctr:   db   3              ; starting sector #
2026  0620 00     r0n$cyll:   db   0              ; cylinder low
2027  0621 00     r0n$cylh:   db   0              ; cylinder high (only b1,b0 used)
2028  0622 00     r0n$sdh:    db   0              ; size, disk type, unit/head
2029  0623 8505   r0n$dmal:   dw   s0n$buf ; 24bit dma low,mid
2030                IF BANKED
2031              r0n$dmah:     db   04           ; 24bit dma high - assumes bnk on xriv
2032                ELSE  ; NOT BANKED
2033  0625 01     r0n$dmah:   db   01             ; 24bit dma high - assumes bnk on board
2034                ENDIF ; NOT BANKED
2035  0626 30     r0n$dfc:    db   0011$0000b     ; dma/floppy control
2036  0627 24     r0n$cmd:    db   rd$cmd         ; command
2037                IF NOT DMA
2038  0628 00     r0n$bflg:   db   000h           ; read of id's don't need bank select
2039                ENDIF ; NOT DMA
2040              ;
2041              ;
2042              ;
2043  FFFE =      unit        equ  -02h           ; unit number
2044  0000 =      strnl       equ  00h            ; lsb sectran table address (or 0)
2045  0001 =      strnh       equ  01h            ; msb sectran table address (or 0)
2046  000B =      media       equ  0bh            ; media change flag
2047  000C =      dpbl        equ  0ch            ; lsb dpb address
2048  000D =      dpbh        equ  0dh            ; msb dpb address
2049              ;
2050  0019 =      dtyp        equ  19h            ; disk type- 5", 8", winch
2051  001A =      step        equ  1ah            ; unit's step rate
2052  001B =      log$skw     equ  1bh            ; logical skew
2053  001C =      sdh         equ  1ch            ; sdh port data
2054  001D =      dfc         equ  1dh            ; dma/floppy control port data
2055  001E =      nhds        equ  1eh            ; shift/divide flag and #hds
2056  001F =      spt         equ  1fh            ; max sector on track + 1
2057  0020 =      scyll       equ  20h            ; start cyl off low
2058  0021 =      scylh       equ  21h            ; start cyl off high
2059  0022 =      wpc         equ  22h            ; winch- write precomp cylinder
2060  0023 =      tms         equ  23h            ; disk time id- min, set
2061  0025 =      nxt$loc     equ  25h            ; next available
2062              ;
2063              ;
2064  0011 =      dpb$lgth    equ  17             ; length of a dpb
2065              ;
2066              ;
2067              ;           FLOPPY DISK PARAMETER HEADERS
2068              ;
2069              ;
```

```
                fxdph   macro   ?unit,?type
                ;
                        dw      vfw$wrt                 ; address of sector write
                        dw      vfw$rd                  ; address of sector read
                        dw      vfw$login               ; address of drive login
                        dw      vfw$init                ; address of drive init routine
                        db      ?unit                   ; physical unit number
                        db      ?type                   ; controller type
                        endm
                ;
                fdph    macro   ?dpb,?sdh
                        dw      0                       ; translate table address
                        db      0,0,0,0,0,0,0,0,0       ; bios scratch area
                        db      0                       ; media flag
                        dw      ?dpb                    ; address of dpb
                ;
                  IF LOADER
                ;
                        dw      csv0                    ; checksum allocated here
                        dw      alv0                    ; alloc vector allocated here
                        dw      drbcb0,dtbcb0,-1        ; dirbcb,dtabcb,hask alloc'd here
                        db      0                       ; hash bank
                ;
                  ELSE  ; NOT LOADER
                ;
                        dw      -2                      ; checksum allocated by gencpm
                        dw      -2                      ; alloc vector allocated by gencpm
                        dw      -2,-2,-2                ; dirbcb,dtabcb,hask alloc'd by gencpm
                        db      0                       ; hash bank
                ;
                  ENDIF ; NOT LOADER
                ;
                        db      flp$5                   ; dtype 5/8/winch
                        db      0                       ; step rate
                        db      0                       ; logical skew flag
                        db      ?sdh                    ; sdh port data
                        db      f$dfc$dft               ; dma/floppy control port data
                        db      0                       ; shft/divide flag and nhds
                        db      0                       ; sctr per track
                        dw      0                       ; log cyl start offset
                        db      77                      ; wrt precomp- dummy byte on flop
                        ds      2                       ; disk id- minutes, seconds
                        endm
                ;
                ;
                ;       FLOPPY DISK PARAMETER BLOCKS
                ;
                max$f$dpb       macro   ?dsm, ?drm
                ;
                        local   ?chksum
                ;
                ?chksum set     (?drm/4)+1
                ;
                        dw      64              ; (spt)
                        db      5,31,1          ; (blm bls exm)
                        dw      ?dsm            ; dsm - max # blocks per drive
```

```
2126                            dw      ?drm            ; drm
2127                            dw      0ffh            ; (alloc vector for directory)
2128                            dw      ?chksum         ; checksum size
2129                            dw      2               ; (offset for system tracks (off))
2130                            db      3,7             ; sector size, and shift mask
2131                        ;
2132                            endm
2133                        ;
2134                          IF FLOP0
2135                        ;
2136                            dseg
2137                        ;
2138                        xvfwfd0:
2139                            fxdph   00, floppy
2140  0629+3901
2141  062B+2F01
2142  062D+0100
2143  062F+0000
2144  0631+00
2145  0632+00
2146                        vfwfd0:
2147                            fdph    dpbf0, 0111$1000b
2148  0633+0000
2149  0635+0000000000
2150  063E+00
2151  063F+3A00
2152  0641+FEFF
2153  0643+FEFF
2154  0645+FEFFFEFFFE
2155  064B+00
2156  064C+10
2157  064D+00
2158  064E+00
2159  064F+78
2160  0650+00
2161  0651+00
2162  0652+00
2163  0653+0000
2164  0655+4D
2165  0656+
2166                        ;
2167                            cseg
2168                        ;                           this filled at login with correct data
2169                        dpbf0:
2170                            max$f$dpb       fdsm0, fdrm0
2171  003A+4000
2172  003C+051F01
2173  003F+FF01
2174  0041+FF00
2175  0043+FF00
2176  0045+4000
2177  0047+0200
2178  0049+0307
2179                        ;
2180                          ENDIF ; FLOP0
2181                        ;
```

```
                        ;
                          IF FLOP1
                        ;
                              dseg
                        ;
                        xvfwfd1:
                              fxdph   01, floppy
0658+3901
065A+2F01
065C+0100
065E+0000
0660+01
0661+00
                        vfwfd1:
                              fdph    dpbf1, 0111$1010b
0662+0000
0664+0000000000
066D+00
066E+4B00
0670+FEFF
0672+FEFF
0674+FEFFFEFFFE
067A+00
067B+10
067C+00
067D+00
067E+7A
067F+00
0680+00
0681+00
0682+0000
0684+4D
0685+
                ;
                      cseg
                ;                          this filled at login with correct data
                dpbf1:
                      max$f$dpb       fdsm1, fdrm1
004B+4000
004D+051F01
0050+FF01
0052+FF00
0054+FF00
0056+4000
0058+0200
005A+0307
                ;
                  ENDIF ; FLOP1
                ;
                ;
                  IF FLOP2
                ;
                      dseg
                ;
                xvfwfd2:
                      fxdph   02, floppy
```

```
2238  0687+3901
2239  0689+2F01
2240  068B+0100
2241  068D+0000
2242  068F+02
2243  0690+00
2244                    vfwfd2:
2245                            fdph    dpbf2, 0111$1100b
2246  0691+0000
2247  0693+0000000000
2248  069C+00
2249  069D+5C00
2250  069F+FEFF
2251  06A1+FEFF
2252  06A3+FEFFFEFFFE
2253  06A9+00
2254  06AA+10
2255  06AB+00
2256  06AC+00
2257  06AD+7C
2258  06AE+00
2259  06AF+00
2260  06B0+00
2261  06B1+0000
2262  06B3+4D
2263  06B4+
2264                    ;
2265                            cseg
2266                    ;                       this filled at login with correct data
2267                    dpbf2:
2268                            max$f$dpb       fdsm2, fdrm2
2269  005C+4000
2270  005E+051F01
2271  0061+FF01
2272  0063+FF00
2273  0065+FF00
2274  0067+4000
2275  0069+0200
2276  006B+0307
2277                    ;
2278                      ENDIF ; FLOP2
2279                    ;
2280                    ;
2281                      IF FLOP3
2282                    ;
2283                            dseg
2284                    ;
2285                    xvfwfd3:
2286                            fxdph   03, floppy
2287  06B6+3901
2288  06B8+2F01
2289  06BA+0100
2290  06BC+0000
2291  06BE+03
2292  06BF+00
2293                    vfwfd3:
```

```
2294                                    fdph    dpbf3, 0111$1110b
2295    06C0+0000
2296    06C2+0000000000
2297    06CB+00
2298    06CC+6D00
2299    06CE+FEFF
2300    06D0+FEFF
2301    06D2+FEFFFEFFFE
2302    06D8+00
2303    06D9+10
2304    06DA+00
2305    06DB+00
2306    06DC+7E
2307    06DD+00
2308    06DE+00
2309    06DF+00
2310    06E0+0000
2311    06E2+4D
2312    06E3+
2313                    ;
2314                            cseg
2315                    ;                       this filled at login with correct data
2316                    dpbf3:
2317                            max$f$dpb       fdsm3, fdrm3
2318    006D+4000
2319    006F+051F01
2320    0072+FF01
2321    0074+FF00
2322    0076+FF00
2323    0078+4000
2324    007A+0200
2325    007C+0307
2326                    ;
2327                      ENDIF ; FLOP3
2328                    ;
2329                    ;
2330                    ;
2331                    ;               WINCHESTER DISK PARAMETER HEADERS
2332                    ;
2333                    ;
2334                    wxdph   macro   ?unit,?type
2335                    ;
2336                            dw      vfw$wrt                 ; address of sector write
2337                            dw      vfw$rd                  ; address of sector read
2338                            dw      vfw$login               ; address of drive login
2339                            dw      vfw$init                ; address of drive init routine
2340                            db      ?unit                   ; physical unit number
2341                            db      ?type                   ; controller type
2342                            endm
2343                    ;
2344                    wdph    macro   ?dpb,?sdh
2345                            dw      0                       ; translate table address
2346                            db      0,0,0,0,0,0,0,0,0       ; bios scratch area
2347                            db      0                       ; media flag
2348                            dw      ?dpb                    ; address of dpb
2349                    ;
```

```
                        IF LOADER
                    ;
                            dw      0000h                   ; no checksum on winch
                            dw      alv0                    ; alloc vector allocated here
                            dw      drbcb0,dtbcb0,-1        ; dirbcb,dtabcb,hask alloc'd here
                            db      0                       ; hash bank
                    ;
                        ELSE  ; NOT LOADER
                    ;
                            dw      0000h                   ; no checksum on winch
                            dw      -2                      ; alloc vector allocated by gencpm
                            dw      -2,-2,-2                ; dirbcb,dtabcb,hask alloc'd by gencpm
                            db      0                       ; hash bank
                    ;
                        ENDIF ; NOT LOADER
                    ;
                            db      winch                   ; dtype 5/8/winch
                            db      0                       ; step rate
                            db      0                       ; logical skew flag
                            db      ?sdh                    ; sdh port data
                            db      f$dfc$dft               ; dma/floppy control port data
                            db      0                       ; shft/divide flag and nhds
                            db      0                       ; sctr per track
                            dw      0                       ; log cyl start offset
                            db      77                      ; wrt precomp- dummy byte on flop
                            ds      2                       ; disk id- minutes, seconds
                            endm
                    ;
                    ;
                    ;       WINCHESTER DISK PARAMETER BLOCKS
                    ;
                    ;
                    max$w$dpb       macro   ?dsm,?drm
                    ;
                            dw      72              ; (spt)
                            db      5,31,1          ; (blm bls exm)
                            dw      ?dsm            ; dsm - max # blocks per drive
                            dw      ?drm            ; drm
                            dw      0ffffh          ; (alloc vector for directory)
                            dw      8000h           ; no checksum on winch
                            dw      2               ; (offset for system tracks (off))
                            db      3,7             ; sector size, and shift mask
                    ;
                            endm
                    ;
                    ;
                      IF WINCH00
                    ;
                            dseg
                    ;
                    xvfwd00:
                            wxdph   0,winch0
06E5+3901
06E7+2F01
06E9+0100
06EB+0000
```

```
2406   06ED+00
2407   06EE+05
2408                    vfwd00:
2409                            wdph    dpbw00, 1110$0000b
2410   06EF+0000
2411   06F1+0000000000
2412   06FA+00
2413   06FB+7E00
2414   06FD+0000
2415   06FF+FEFF
2416   0701+FEFFFEFFFE
2417   0707+00
2418   0708+40
2419   0709+00
2420   070A+00
2421   070B+E0
2422   070C+00
2423   070D+00
2424   070E+00
2425   070F+0000
2426   0711+4D
2427   0712+
2428                    ;
2429                            cseg
2430                    ;                                       this filled at login with correct data
2431                    dpbw00:
2432                            max$w$dpb       wdsm00, wdrm00
2433   007E+4800
2434   0080+051F01
2435   0083+FF0C
2436   0085+FF03
2437   0087+FFFF
2438   0089+0080
2439   008B+0200
2440   008D+0307
2441                    ;
2442                      ENDIF ; WINCH00
2443                    ;
2444                    ;
2445                      IF WINCH01
2446                    ;
2447                            dseg
2448                    ;
2449                    xvfwd01:
2450                            wxdph   1,winch0
2451   0714+3901
2452   0716+2F01
2453   0718+0100
2454   071A+0000
2455   071C+01
2456   071D+05
2457                    vfwd01:
2458                            wdph    dpbw01, 1110$0000b
2459   071E+0000
2460   0720+0000000000
2461   0729+00
```

```
2462   072A+8F00
2463   072C+0000
2464   072E+FEFF
2465   0730+FEFFFEFFFE
2466   0736+00
2467   0737+40
2468   0738+00
2469   0739+00
2470   073A+E0
2471   073B+00
2472   073C+00
2473   073D+00
2474   073E+0000
2475   0740+4D
2476   0741+
2477                   ;
2478                                   cseg
2479                   ;                                       this filled at login with correct data
2480                   dpbw01:
2481                           max$w$dpb       wdsm01, wdrm01
2482   008F+4800
2483   0091+051F01
2484   0094+FF0C
2485   0096+FF03
2486   0098+FFFF
2487   009A+0080
2488   009C+0200
2489   009E+0307
2490                   ;
2491                     ENDIF ; WINCH01
2492                   ;
2493                   ;
2494                     IF WINCH02
2495                   ;
2496                           dseg
2497                   ;
2498                   xvfwd02:
2499                           wxdph   2,winch0
2500                   vfwd02:
2501                           wdph    dpbw02, 1110$0000b
2502                 ;
2503                           cseg
2504                 ;                                       this filled at login with correct data
2505                 dpbw02:
2506                           max$w$dpb       wdsm02, wdrm02
2507                 ;
2508                   ENDIF ; WINCH02
2509                 ;
2510                 ;
2511                   IF WINCH03
2512                 ;
2513                           dseg
2514                 ;
2515                 xvfwd03:
2516                           wxdph   3,winch0
2517                 vfwd03:
```

```
2518                    wdph    dpbw03, 1110$0000b
2519                ;
2520                    cseg
2521                ;                                   this filled at login with correct data
2522                dpbw03:
2523                    max$w$dpb       wdsm03, wdrm03
2524                ;
2525                  ENDIF ; WINCH03
2526                ;
2527                ;
2528                ;
2529                  IF WINCH10
2530                ;
2531                    dseg
2532                ;
2533                xvfwd10:
2534                    wxdph   0,winch1
2535                vfwd10:
2536                    wdph    dpbw10, 1110$1000b
2537                ;
2538                    cseg
2539                ;                                   this filled at login with correct data
2540                dpbw10:
2541                    max$w$dpb       wdsm10, wdrm10
2542                ;
2543                  ENDIF ; WINCH10
2544                ;
2545                ;
2546                  IF WINCH11
2547                ;
2548                    dseg
2549                ;
2550                xvfwd11:
2551                    wxdph   1,winch1
2552                vfwd11:
2553                    wdph    dpbw11, 1110$1000b
2554                ;
2555                    cseg
2556                ;                                   this filled at login with correct data
2557                dpbw11:
2558                    max$w$dpb       wdsm11, wdrm11
2559                ;
2560                  ENDIF ; WINCH11
2561                ;
2562                ;
2563                  IF WINCH12
2564                ;
2565                    dseg
2566                ;
2567                xvfwd12:
2568                    wxdph   2,winch1
2569                vfwd12:
2570                    wdph    dpbw12, 1110$1000b
2571                ;
2572                    cseg
2573                ;                                   this filled at login with correct data
```

```
2574                    dpbw12:
2575                            max$w$dpb       wdsm12, wdrm12
2576                    ;
2577                      ENDIF ; WINCH12
2578                    ;
2579                    ;
2580                      IF WINCH13
2581                    ;
2582                            dseg
2583                    ;
2584                    xvfwd13:
2585                            wxdph   3,winch1
2586                    vfwd13:
2587                            wdph    dpbw13, 1110$1000b
2588                    ;
2589                            cseg
2590                    ;                                       this filled at login with correct data
2591                    dpbw13:
2592                            max$w$dpb       wdsm13, wdrm13
2593                    ;
2594                      ENDIF ; WINCH13
2595                    ;
2596                    ;
2597                    ;
2598                      IF WINCH20
2599                    ;
2600                            dseg
2601                    ;
2602                    xvfwd20:
2603                            wxdph   0,winch2
2604                    vfwd20:
2605                            wdph    dpbw20, 1111$0000b
2606                    ;
2607                            cseg
2608                    ;                                       this filled at login with correct data
2609                    dpbw20:
2610                            max$w$dpb       wdsm20, wdrm20
2611                    ;
2612                      ENDIF ; WINCH20
2613                    ;
2614                    ;
2615                      IF WINCH21
2616                    ;
2617                            dseg
2618                    ;
2619                    xvfwd21:
2620                            wxdph   1,winch2
2621                    vfwd21:
2622                            wdph    dpbw21, 1111$0000b
2623                    ;
2624                            cseg
2625                    ;                                       this filled at login with correct data
2626                    dpbw21:
2627                            max$w$dpb       wdsm21, wdrm21
2628                    ;
2629                      ENDIF ; WINCH21
```

```
                ;
                ;
                  IF WINCH22
                ;
                        dseg
                ;
                xvfwd22:
                        wxdph   2,winch2
                vfwd22:
                        wdph    dpbw22, 1111$0000b
                ;
                        cseg
                ;                                       this filled at login with correct data
                dpbw22:
                        max$w$dpb       wdsm22, wdrm22
                ;
                  ENDIF ; WINCH22
                ;
                ;
                  IF WINCH23
                ;
                        dseg
                ;
                xvfwd23:
                        wxdph   3,winch2
                vfwd23:
                        wdph    dpbw23, 1111$0000b
                ;
                        cseg
                ;                                       this filled at login with correct data
                dpbw23:
                        max$w$dpb       wdsm23, wdrm23
                ;
                  ENDIF ; WINCH23
                ;
                ;
                        cseg                    ; this must be in common
                ;
                  IF VFW$INT
                ;
                ?vfw$cpl        db      0       ; completion interrupt sets this flag
                ;
                  IF NOT DMA
                ;
                scnt            db      0       ; temp for doing pio multio under interrupts
                ;
                  ENDIF ; NOT DMA
                ;
                  ENDIF ; VFW$INT
                ;
                ;
                        dseg
                ;
                ;       DEFAULT LOGIN INFO
                ;
                ;
```

```
2686                    df8$info:
2687                    ;
2688  0743 06           df8$lskw      db      6
2689                    ;
2690                    df8$dpb:
2691  0744 1A00               dw      26              ; spt
2692  0746 030700             db      3,7,0           ; blm bls exm
2693  0749 F200               dw      242             ; dsm
2694  074B 3F00               dw      63              ; drm
2695  074D C000               dw      0c0h            ; alloc vector for directory
2696  074F 1000               dw      16              ; checksum size cks
2697  0751 0200               dw      2               ; offset for system tracks (off)
2698  0753 0000               db      0,0             ; sector size, and shift mask
2699                    ;
2700  0011 =            dft$dpb$lgth  equ     $-df8$dpb
2701                    ;
2702                    df8$ddb:                              ; disk definition block
2703  0755 8000         df8$bps:      dw      128             ; bytes per sector
2704  0757 4D00         df8$ncyl:     dw      77              ; #cylinders per logical disk
2705  0759 1A           df8$spt:      db      26              ; #sctrs per track
2706  075A 01           df8$nhd:      db      1               ; #heads
2707  075B 03           df8$sr:       db      3               ; step rate- useless
2708  075C 13           df8$wpc:      db      77/4            ; write precomp
2709  075D 05           df8$dfc:      db      0000$0101b      ; dma/floppy control
2710  075E 60           df8$sdh:      db      0110$0000b      ; size, disk#, head#
2711  075F 42           df8$gap:      db      42h             ; format gap
2712  0760 01           df8$psku:     db      1               ; physical skew
2713  0761 0000         df8$scyl:     dw      0               ; start cylinder offset
2714  0763 0000         df8$sp2:      db      0,0             ; spare
2715                    ;
2716  0010 =            dft$ddb$lgth  equ     $-df8$ddb
2717                    ;
2718                    ;
2719                    df5$info:
2720                    ;
2721  0765 06           df5$lskw      db      6
2722                    ;
2723                    df5$dpb:
2724  0766 1200               dw      18              ; spt
2725  0768 030700             db      3,7,0           ; blm bls exm
2726  076B 4700               dw      71              ; dsm
2727  076D 3500               dw      53              ; drm
2728  076F C000               dw      0c0h            ; alloc vector for directory
2729  0771 0E00               dw      14              ; checksum size cks
2730  0773 0300               dw      3               ; offset for system tracks (off)
2731  0775 0000               db      0,0             ; sector size, and shift mask
2732                    ;
2733                    df5$ddb:                              ; disk definition block
2734  0777 8000         df5$bps:      dw      128             ; bytes per sector
2735  0779 5000         df5$ncyl:     dw      80              ; #cylinders per logical disk
2736  077B 12           df5$spt:      db      18              ; #sctrs per track
2737  077C 01           df5$nhd:      db      1               ; #heads
2738  077D 03           df5$sr:       db      3               ; step rate- useless
2739  077E 14           df5$wpc:      db      80/4            ; write precomp
2740  077F 04           df5$dfc:      db      0000$0100b      ; dma/floppy control
2741  0780 60           df5$sdh:      db      0110$0000b      ; size, disk#, head#
```

```
2742  0781 42      df5$gap:    db    42h          ; format gap
2743  0782 01      df5$pskw:   db    1            ; physical skew
2744  0783 0000    df5$scyl:   dw    0            ; start cylinder offset
2745  0785 0000    df5$sp2:    db    0,0          ; spare
2746               ;
2747               ;
2748               ;
2749                 IF LOADER
2750               ;
2751               log$flg     db    0
2752               org$dv      ds    1
2753               org$sv      ds    1
2754               ;
2755               csv0:  ds     64
2756               alv0:  ds     32
2757               ;
2758               ;       FLOPPY BCB'S
2759               ;
2760               drbcb0:
2761                       db    0ffh    ; drv
2762                       db    0,0,0   ; rec#
2763                       db    0       ; write flag
2764                       db    0       ; sequence number
2765                       dw    0       ; track number
2766                       dw    0       ; sector number
2767                       dw    buff0   ; buff ad
2768                       db    0       ; buffer bank
2769                       dw    0       ; link
2770               ;
2771               dtbcb0:
2772                       db    0ffh    ; drv
2773                       db    0,0,0   ; rec#
2774                       db    0       ; write flag
2775                       db    0       ; sequence number
2776                       dw    0       ; track number
2777                       dw    0       ; sector number
2778                       dw    buff1   ; buff ad
2779                       db    0       ; buffer bank
2780                       dw    0       ; link
2781               ;
2782               buff0  equ    2000h          ; this should be above loader but below load
2783               buff1  equ    2400h          ; 1k each
2784               ;
2785                 ENDIF ; LOADER
2786               ;
2787               ;
2788  0787                 end
```

```
ABTDCMD         0004  1114  1925#
ADJSCT          01DF   628#  635   641   645
B4896           0000   745  1909#
B48FLG          0001   172  1904#
B5OR8           0000   166   245   248   252  1905#
BANKED          0000    68    77   831  1135  1237  1252  1306  1347  2030
BC              0000  1162  1230
BDFCHD3         0001   772  1894#
BDFCRD          0004   894  1892#
BDFCSD          0002   810  1893#
BDIVFLG         0007   374  1907#
BDOSERR         0497  1630  1667  1736  1797#
BFLP5           0004   314  1918#
BFLP8           0005  1403  1917#
BIOSERR         03C7   275   569  1625#
BLDTSK          01C2   475   510   598#
BNO128          0006   788  1903#
BNOLOG          0007   192  1902#
BSTABSY         0007   943   989  1018  1022  1053  1278  1300  1336  1459  1708
                      1896#
BSTADRQ         0003  1179  1198  1284  1329  1342  1899#
BSTAERR         0000  1071  1214  1900#
BSTAFWP         0005  1647  1898#
BSTARDY         0006  1102  1216  1412  1469  1897#
BSYCHK          02A2   941#
BSYERR          00D1   948  1923#
BSYNRDY         02AA   949#
BTPI48          0000   176   741  1919#
BUBBLE          0006
BWINCH          0006   158   207   307   553   796  1389  1916#
CHK48           0240   732   740#
CHKBIT          00E8   364#  368
CHKIGNOR        0449  1718  1723#
CHKMEDIA        03A7   564  1577#
CHKMORE         01B0   558   577#
CHKRDY          0337  1405  1410#
CHKRST          036D  1461  1468#
CHKSPARE        03A4   557  1530#
CHKSTEP         02C6   978# 1039#
CMDACT          02A2   199   546   939# 1400  1423  1429
CNT             057F   470   493   518   589  1936#
CNTR3           0000
CURSCTR         057B   273  1454  1682  1932#
CURSTEP         057A   137   979   983  1040  1047  1931#
CURTRK          0578   271   498   656  1680  1930#
DCRCNT          0182   494   511   517#
DE              0002
DF5BPS          0777  2734#
DF5DDB          0777  2733#
DF5DFC          077F  2740#
DF5DPB          0766  2723#
DF5GAP          0781  2742#
DF5INFO         0765   313  2719#
DF5LSKW         0765  2721#
DF5NCYL         0779  2735#
DF5NHD          077C  2737#
```

```
ERRRESP         0498  1759  1799#
ERRXIT          01BB   565   570   586#
EXECPIO         0000  1064  1130#
FALSE           0000
FDFCDFT         0000  1876# 1883# 2106  2370
FDRM0           00FF  2170
FDRM1           00FF  2219
FDRM2           00FF  2268
FDRM3           00FF  2317
FDSM0           01FF  2170
FDSM1           01FF  2219
FDSM2           01FF  2268
FDSM3           01FF  2317
FIX0            038F  1501#
FIX00           038C  1496# 1498
FIX1            0397  1506# 1509
FIXTSK          0371  1474#
FLOP            0030  1913#
FLOP0           FFFF    40  2134
FLOP1           FFFF    41  2183
FLOP2           FFFF    42  2232
FLOP3           FFFF    43  2281
FLOPERR         01A3   555   563#
FLOPPY          0000  2139  2188  2237  2286
FLP5            0010   171  1911# 2102
FLP8            0020   165  1912#
FRC128          0282   798   804#
FRCHD0          0341  1414  1419#
FUNC            057E   124   465   890   906  1641  1693  1935#
FUNCMSGS        049C  1692  1801#
GETRSP          0459  1715  1726  1741#
HARD            0001
HARDERR         0454  1671  1710  1735#
HL              0004
IGNORMSG        0561  1724  1842#
INRSCNT         017E   505   514#
INVDRI          0083   276   284#  394
IO8             0000
IX              0004
IY              0004
LOADER          0000   146   205  1627  1706  2086  2350  2749
LOGERR          0066   208   213   234   267#  309
LOGIN           0020   123  1890#
LOGINMSG        04B3  1802  1806#
LOGSKW          001B   336   485   613  2052#
LOGXIT          012C   194   417#
MEDIA           000B  1600  2046#
MEDIACHG        03BE  1582  1598#
MOVDFT          00A8   316   319#
MOVWPC          010F   396#
NHDS            001E   378   658  2055#
NOBIT           0493  1773  1788#
NOCAR           0223   700   703#
NOTVFW          0097   297   304#
NRDYERR         0011  1924#
NULLMSG         0577  1811  1813  1830  1831  1833  1843#
```

```
NXTLOC          0025  2061#
OUTTO           02B8   961#  965
P300CIO         0078
P300PAR         0070
P300SCC         007C
PCIOA           007A
PCIOB           0079
PCIOC           0078
PCIOCTL         007B
PIO8            00A0
PIO8RTC         00A6
PIO8RTCRS       00A5
PROCERR         035B   549  1452#
PRTSTAT         0584  1664  1701  1946#
PSCCADATA       007F
PSCCASTAT       007E
PSCCBDATA       007D
PSCCBSTAT       007C
PTABMSG         047D  1694  1698  1703  1766#
PVDB            0000
PVDBDATA        0001
PVDBSTAT        0000
PVFW            0050  1851
PXRAM           00FF
RONBFLG         0628  2038#
RONCMD          0627  2036#
RONCYLH         0621  2027#
RONCYLL         0620  2026#
RONDFC          0626  1386  1420  1422  1426  1428  2035#
RONDMAH         0625  2031# 2033#
RONDMAL         0623  2029#
RONSCNT         061E  2024#
RONSCTR         061F   272  1397  2025#
RONSDH          0622  1381  2028#
RONTBL          061D  1399  2022#
RONWPC          061D  2023#
RAM             0004
RDCMD           0024   915  1874# 1881# 2036
RDEND           001C  1286  1297#
RDSCTR          000A  1274# 1302
RDTOSN          0308   201  1377# 1579
RDWAIT          000A  1170# 1173  1276# 1280
RDWRT           018A   478   488   508   520   539#
RDYCHK          02F7  1077  1099#
RDYERR          0301  1104  1110#
READ            0080   432   891   907  1888#
READIT          0010  1177# 1185  1283# 1294
READMSG         04A2  1801  1804#
REDOCI          045F  1744# 1754
REDOLOG         000B   126#  263   281
REDORW          018A   541#  574   583
REMOTE          0002
REPERR          01A8   550   560   568#
RESTOR          005B   197#
RETDFT          057C  1742  1756  1934#
RETRYMSG        0551  1713  1841#
```

```
ROM             0003
RSCMD           0010   138  1869#
RSTBFLG         061C  2018#
RSTCMD          061B   139  2016#
RSTCYLH         0615  2010#
RSTCYLL         0614  2009#
RSTDFC          061A   186  2015#
RSTDMAH         0619  2014#
RSTDMAL         0617  2012#
RSTDMAM         0618  2013#
RSTSCNT         0612  2007#
RSTSCTR         0613  2008#
RSTSDH          0616   190  2011#
RSTTBL          0611   198  2005#
RSTWPC          0611  2006#
SONBPS          05B5  1966#
SONBUF          0585  1949# 1985  2029
SONDDB          05B5   329  1965#
SONDFC          05BD   353  1972#
SONDPB          0595   324   407  1962#
SONDT           0587  1952#
SONGAP          05BF  1974#
SONHR           0589  1953#
SONID           0585   294  1951#
SONLBL          05C5  1979#
SONLSKW         058C   321   335  1956#
SONMN           058A   401  1588  1954#
SONNCYL         05B7  1967#
SONNHD          05BA   359  1969#
SONNWD          05D5  1982#
SONPSKW         058D  1957#
SONREL          058F  1959#
SONSC           058B  1955#
SONSCYL         05C1   385  1976#
SONSDH          05BE   343  1973#
SONSKW          05C0  1975#
SONSP0          0590  1960#
SONSP1          05A6  1963#
SONSP2          05C3  1977#
SONSP5          05E5  1985# 1985
SONSPT          05B9   381  1968#
SONSR           05BB  1970#
SONTSKW         058E  1958#
SONWCL          05D5  1981#
SONWPC          05BC   397  1971#
SONWS0          05D6  1983#
SASI            0000
SAVONS          0325  1391  1396#
SAVCMD          02A0   916   919#
SAVCYL          0248   720   743   748#
SAVERR          0583  1697  1944#
SAVERRINFO      0582   269  1457  1942#
SAVNHDS         00F4   372   377#
SAVSCT          01EF   618   630   648#
SAVSTA          03EC  1644  1649  1663#
SAVSTAT         0582   210  1636  1943#
```

```
SCYLH           0021  388  751  2058#
SCYLL           0020  386  749  2057#
SDH             001C  188  339   346  775  1378  1488  2053#
SDHEND          0288  780  791   802  812#
SETDFC          0047  168  174   179#
SETFWP          03EB 1656 1661#
SETHD           0262  769  774#
SETMULT         0163  491#
SETRDFC         004D  160  185#
SETSKW          00BD  301  334#
SETUP           0143  435  450   464#
SHFTLP          0236  729#  738
SHIFTLP         0480 1768# 1791
SHLDVD          0208  678#  681
SHLDVR          020F  686#  688
SKCMD           0070  984  1048  1870#
SKPFIX          03A2 1471 1515#
SKWAIT          02D8  987#  990  1051# 1055
SNDDAT          0023 1193# 1199  1323# 1331
SNDTSK0         02B3  945  957#
SNDTSK1         02DE  981  992# 1043  1058#
SPT             001F  382  622  2056#
STAT1           0541 1832 1839#
STAT4           053A 1829 1838#
STAT5           0532 1828 1837#
STAT6           0526 1827 1836#
STAT7           051F 1826 1835#
STATBSY         0080 1921#
STATBYT         0580  950 1070  1100  1411  1939#
STATERR         0001 1111 1922#
STATMSGS        050F 1700 1825#
STEP            001A  135  980   982  1041  1045  2051#
STEPVAR         FFB0  127 1928#
STRNH           0001 2045#
STRNL           0000 2044#
TMS             0023  402  404  1589  1591  2060#
TRUE            FFFF
TSKBFLG         0610 2001#
TSKCMD          060F 1999#
TSKCYLH         0609 1993#
TSKCYLL         0608 1992#
TSKDFC          060E 1998#
TSKDMAH         060D 1504 1513  1997#
TSKDMAL         060B 1503 1512  1995#
TSKDMAM         060C 1996#
TSKSCNT         0606  515  578  1990#
TSKSCTR         0607  581 1475  1991#
TSKSDH          060A 1994#
TSKTBL          0605  544 1988#
TSKWPC          0605  599 1989#
UNIT            FFFE 1393 2043#
VFWBAS          0050 1851# 1853  1854  1856  1857  1858  1859  1860  1861  1863
                     1864  1865  1866
VFWCMD          0057  985 1003  1049  1158  1262  1861# 1862
VFWCYLH         0055 1859#
VFWCYLL         0054 1858#
```

```
VFWD00          06EF   45  2408#
VFWD01          071E   46  2457#
VFWDATA         0050  1144  1264  1853#
VFWDFC          005B  1035  1866#
VFWDMAH         005A  1865#
VFWDMAL         0058  1863#
VFWDMAM         0059  1864#
VFWERR          0051  1073  1855#
VFWFD0          0633   40  2146#
VFWFD1          0662   41  2195#
VFWFD2          0691   42  2244#
VFWFD3          06C0   43  2293#
VFWINIT         0000   96#  2075  2339
VFWINT          0000   58   996  1005  1132  2668
VFWLOGIN        0001  116#  2074  2338
VFWRD           012F  431#  2073  2337
VFWSCNT         0052  1146  1155  1856#
VFWSCTR         0053  1224  1226  1453  1857#
VFWSDH          0056  1860#
VFWSTAT         0057   942   988  1012  1017  1021  1052  1178  1197  1210  1277
                      1293  1299  1328  1335  1341  1862#
VFWWPC          0051   959  1854#  1855
VFWWRT          0139   446#  2072  2336
WDRM00          03FF  2432
WDRM01          03FF  2481
WDRM02          03FF  2506
WDRM03          03FF  2523
WDRM10          03FF  2541
WDRM11          03FF  2558
WDRM12          03FF  2575
WDRM13          03FF  2592
WDRM20          03FF  2610
WDRM21          03FF  2627
WDRM22          03FF  2644
WDRM23          03FF  2661
WDSM00          0CFF  2432
WDSM01          0CFF  2481
WDSM02          0CFF  2506
WDSM03          0CFF  2523
WDSM10          0CFF  2541
WDSM11          0CFF  2558
WDSM12          0CFF  2575
WDSM13          0CFF  2592
WDSM20          0CFF  2610
WDSM21          0CFF  2627
WDSM22          0CFF  2644
WDSM23          0CFF  2661
WINCH           0040  1654  1914#  2366
WINCH0          0005  2401  2450  2499  2516
WINCH00         FFFF   45  2396
WINCH01         FFFF   46  2445
WINCH02         0000   47  2494
WINCH03         0000   48  2511
WINCH1          0006  2534  2551  2568  2585
WINCH10         0000   49  2529
WINCH11         0000   50  2546
```

```
WINCH12          0000   51  2563
WINCH13          0000   52  2580
WINCH2           0007 2603  2620  2637  2654
WINCH20          0000   53  2598
WINCH21          0000   54  2615
WINCH22          0000   55  2632
WINCH23          0000   56  2649
WINDSK           0005
WPC              0022  398   602  2059#
WRITE            0040  447  1642  1889#
WRITEMSG         04AA 1801  1805#
WRTCMD           0034  918  1164  1267  1875# 1882#
WRTSCTR          0023 1165  1190# 1268  1321# 1344
WRTWAIT          002D 1203# 1206  1334# 1338
XVFWD00          06E5 2400#
XVFWD01          0714 2449#
XVFWFD0          0629 2138#
XVFWFD1          0658 2187#
XVFWFD2          0687 2236#
XVFWFD3          06B6 2285#
?CONIN           0000   66  1745
?PDERR           0000   66  1685
?PMSG            0000   66  1743  1761  1784
@ADRV            0000   65   128   223   238   785
@CNT             0000   65   476   483
@DMA             0000   65   820   833   867
@ERMDE           0000   86  1669
@MEDIA           0000   86  1602
@RDRV            0000   65
@SECT            0000   65   617  1678  1683  1688
@TRK             0000   65   499   655   691  1676  1681  1690
```

APPENDIX H
PARTS LIST FOR VFW-III

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
0	VFW-III BOARD SCHEMATIC	0300731	
1	VFW-III PC BOARD	7000068	
1	SOCKET, 40 PIN	7060025	U29
1	SOCKET, 24 PIN	7060024	U6
2	PCB EJECTORS	7130072	
2	VOLTAGE REGULATOR	7160012	VR1, VR2
2	HEATSINK	7130224	
4	6-32 X 3/8 PPH SCREW	7130006	
4	6-32 NUT	7130007	
4	#6 LOCKWASHER	7130009	
1	2 X 8 HEADER	7170100	W11-W18
1	2 X 6 HEADER	7170085	J5
4	1 X 3 HEADER	7170021	W1-W3, W21
1	1 X 2 HEADER	7170018	W20
6	HEADER JUMPER	7170004	W1-W3, W14, W20, W21
3	2 X 10 CONNECTOR	7090199	J6-J8
2	2 X 17 CONNECTOR	7090198	J2, J4
1	2 X 25 CONNECTOR	7090197	J3
1	8 POSITION DIP SWITCH	7050002	SW1
1	TPQ 6700	7010544	U39
1	DELAY LINE	7010543	U20
1	4 MHZ CRYSTAL	7080002	Y1
1	20 MHZ CRYSTAL	7080022	Y2
1	2N2907 TRANSISTOR	7040043	Q1
8	1N4148 DIODE	7040001	CR1-CR8
2	4.7 UH INDUCTOR	7120011	L1, L2
2	7-60 PF TRIM CAP.	7030107	C29, C47
2	22 PF CAPACITOR	7030067	C12, C48
2	68 PF CAPACITOR	7030106	C25, C11
3	150 PF CAPACITOR	7030042	C20, C21, C33
1	300 PF CAPACITOR	7030044	C34
1	330 PF CAPACITOR	7030104	C24
1	0.0068 UF CAPACITOR	7030105	C19
1	0.01 UF CAPACITOR	7030046	C32
35	0.1 UF CAPACITOR	7030045	C2, C3, C5-C10, C13-C18, C22, C23, C27, C28, C30, C31, C35-C46, C49-C51
1	2.2 UF CAPACITOR	7030065	C4
4	10 UF CAPACITOR	7030080	C1, C26, C52, C53
1	200 OHM RES.	7020056	R23
3	220 OHM RES.	7020057	R13, R20, R33
1	270 OHM RES.	7020059	R18
2	330 OHM RES.	7020061	R15, R21
4	470 OHM RES.	7020065	R2, R3, R12, R14

QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
1	1K OHM 1% RES.	7020191	R19
7	1K OHM RES.	7020073	R5, R9, R10, R11, R24-R26
1	2K OHM RES.	7020080	R28
2	2.4K OHM RES.	7020082	R34, R35
1	3.3K OHM RES.	7020085	R27
6	4.7K OHM RES.	7020089	R1, R4, R22, R31, R36, R38
1	5.6K OHM RES.	7020091	R32
1	10K OHM RES.	7020097	R6
2	10K OHM TRIM RES.	7020220	R7, R8
2	22K OHM RES.	7020105	R29, R30
2	27K OHM RES.	7020107	R16, R17
1	100 OHM 8 PIN RES. NET.	7010593	RN4
1	150 OHM 6 PIN RES. NET.	7010551	RN2
1	220/330 8 PIN RES. NET.	7010552	RN3
1	1K OHM 6 PIN RES. NET.	7010556	RN6
1	4.7K OHM 6 PIN RES. NET.	7010348	RN1
1	4.7K OHM 10 PIN RES. NET.	7010434	RN5
2	74LS00	7010160	U34, U44
1	74S00	7010335	U41
3	74LS02	7010162	U15, U43, U55
3	74LS04	7010164	U2, U33, U37
1	74S04	7010336	U21
3	7406	7010007	U7, U11, U32
1	74LS14	7010172	U45
3	74S64	7010548	U4, U14, U25
2	74LS74	7010195	U13, U42
3	74S74	7010402	U22, U27, U40
1	74LS85	7010200	U35
1	74S124	7010408	U38
1	74LS125	7010526	U30
2	74LS132	7010217	U18, U46
1	74LS145	7010221	U1
1	74LS155	7010226	U52
1	74LS174	7010241	U19
1	74LS240	7010260	U58
2	74LS244	7010264	U48,U54
2	74LS245	7010265	U53,U56
1	74LS273	7010276	U57
1	74LS373	7010304	U5
2	74LS393	7010312	U16,U31
3	74LS461	7010549	U49-U51
1	26S02	7010545	U26
1	26LS31	7010546	U3
1	26LS32	7010547	U24
1	DS3487	7010555	U10
1	PAL16R8(VFW.003)	7250013	U47
1	PAL16L8(VFW.002)	7250014	U36
1	PAL20L10(VFW.001)	7250015	U17
1	2716(VFW.004)	7250016	U6
1	2K X 8 STATIC RAM	7010499	U8

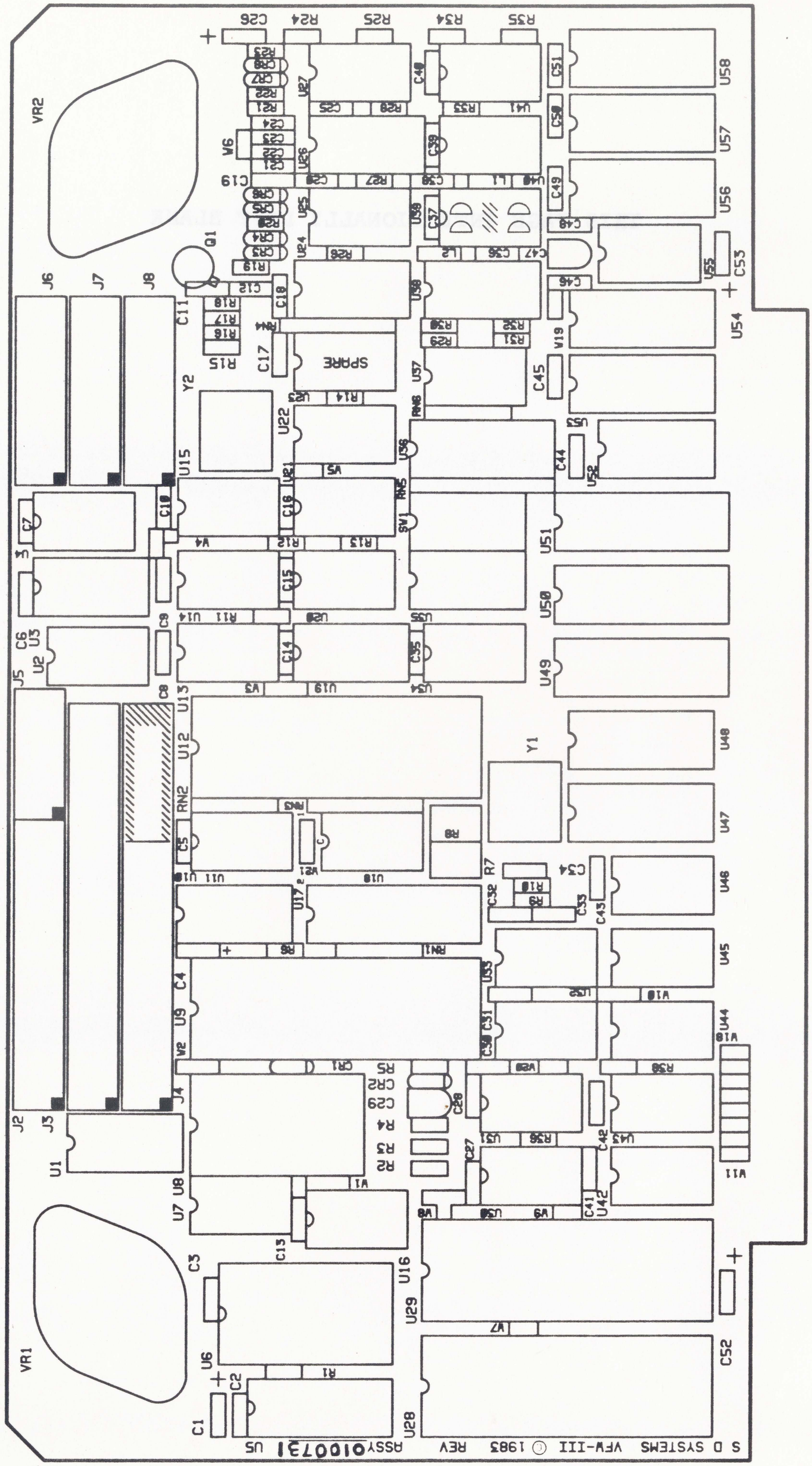
QTY REQD	DESCRIPTION	PART/SUB NUMBER	DESIGNATION
1	WD1010	7010539	U12
1	WD1014	7010541	U28
1	WD1015	7010542	U29
1	WD2797	7010540	U9

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I

PARTS PLACEMENT DIAGRAM

THIS PAGE INTENTIONALLY LEFT BLANK


S D SYSTEMS VFW-III © 1983 REV
ASSY 0100731 US
VR1
VR2
Y1
Y2
SPARE
J1
J2
J3
J4
J5
J6
J7
J8
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
U27
U28
U29
U30
U31
U32
U33
U34
U35
U36
U37
U38
U39
U40
U41
U42
U43
U44
U45
U46
U47
U48
U49
U50
U51
U52
U53
U54
U55
U56
U57
U58
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C39
C40
C41
C42
C43
C44
C45
C46
C47
C48
C49
C50
C51
C52
C53
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
R12
R13
R14
R15
R16
R17
R18
R19
R20
R21
R22
R23
R24
R25
R26
R27
R28
R29
R30
R31
R32
R33
R34
R35
R36
R37
R38
RN1
RN2
RN3
RN4
RN5
RN6
CR1
CR2
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21

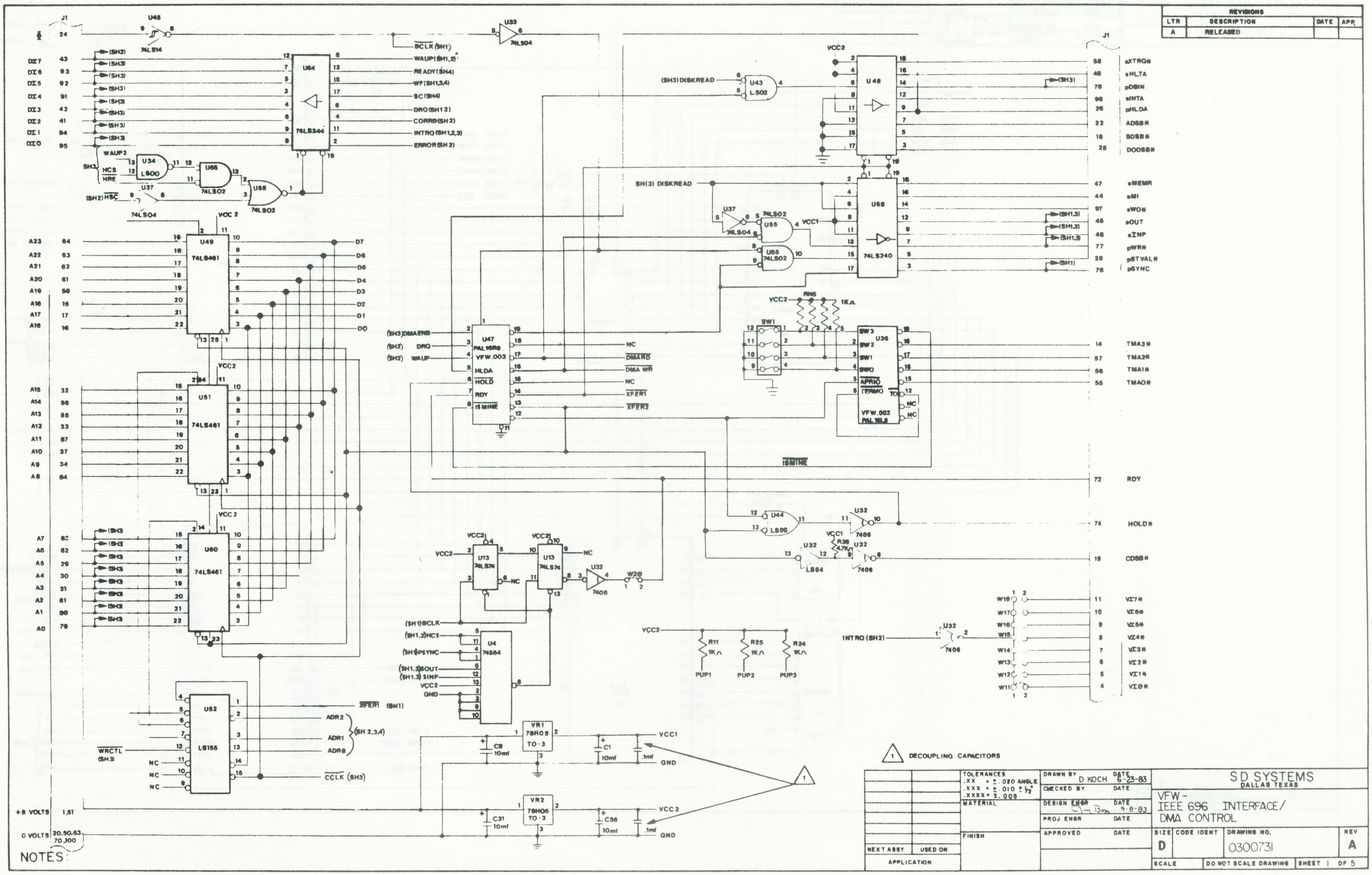
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J

SCHEMATIC

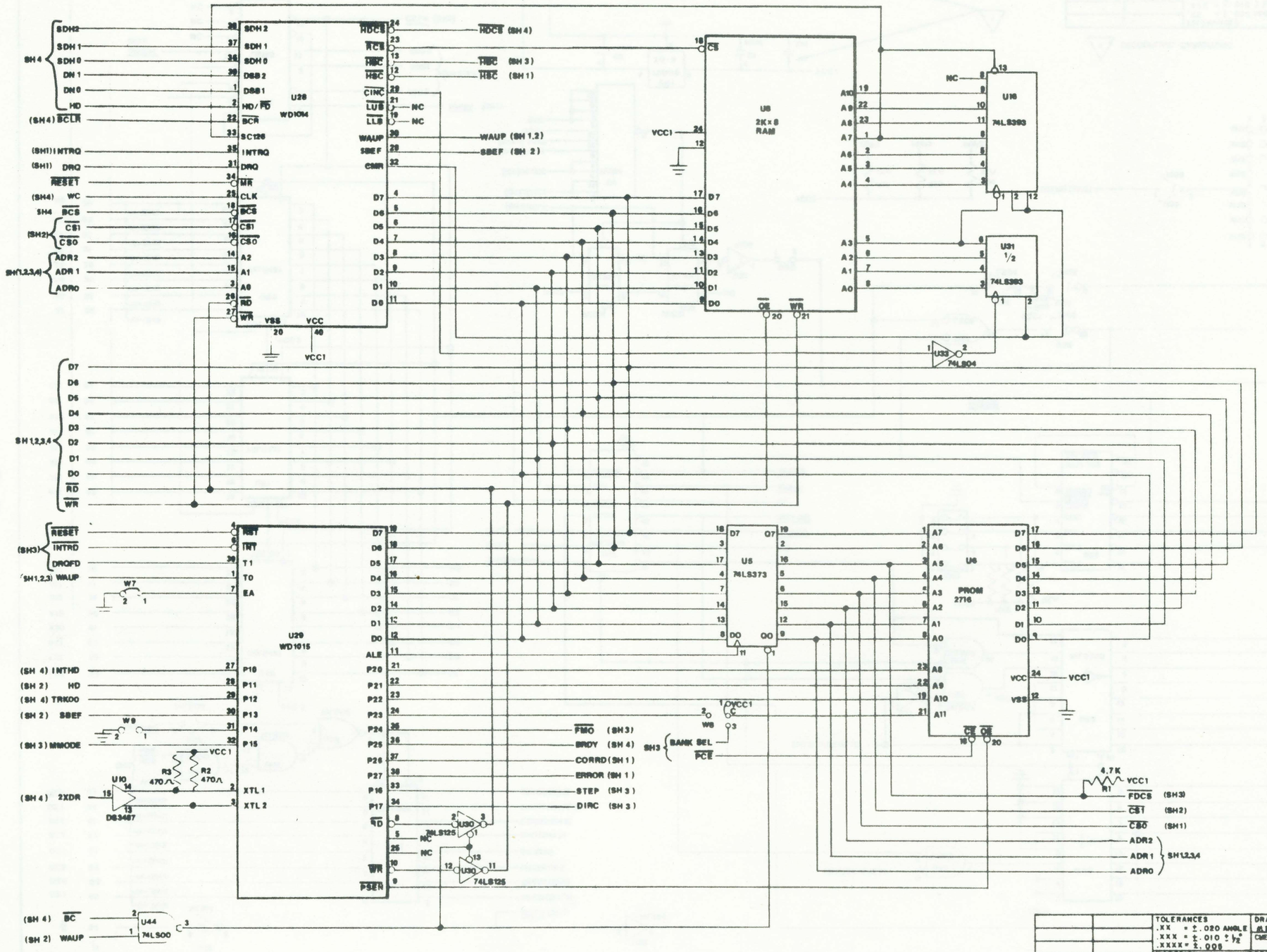
THIS PAGE INTENTIONALLY LEFT BLANK

REVISIONS			
LTR	DESCRIPTION	DATE	APP.
A	RELEASED		



TOLERANCES .XX = ± .020 ANGLE .XXX = ± .010 ± ½° .XXXX = ± .005	DRAWN BY D KOCH	DATE 6-23-83	SD SYSTEMS DALLAS TEXAS
MATERIAL	CHECKED BY	DATE	VFW - IEEE 696 INTERFACE/ DMA CONTROL
FINISH	DESIGN ENGR	DATE 9-8-83	
NEXT ASSY / USED ON	PROJ ENGR	DATE	SIZE D / CODE IDENT / DRAWING NO. 0300731 / REV A
APPLICATION	APPROVED	DATE	SCALE / DO NOT SCALE DRAWING / SHEET 1 OF 5

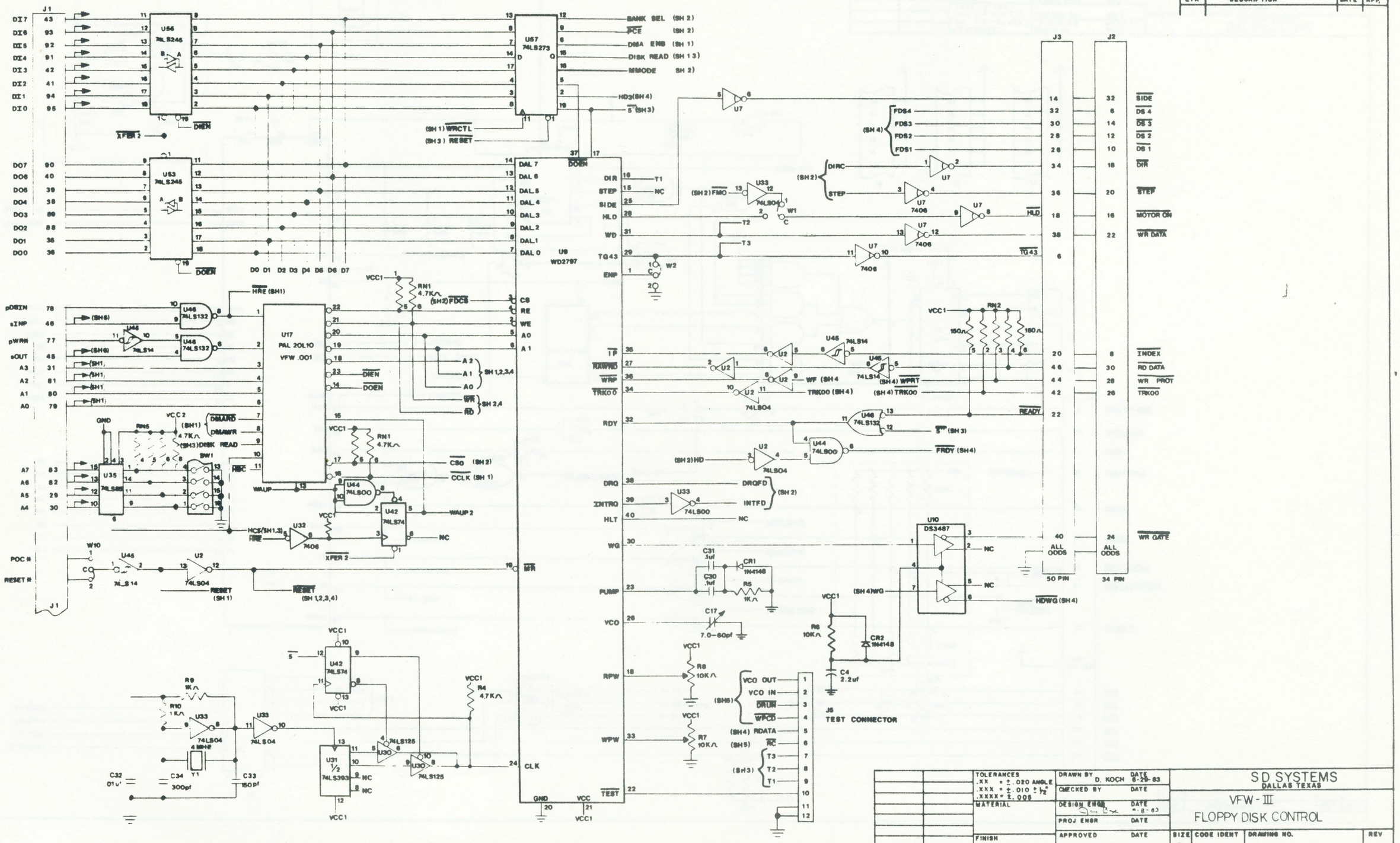
REVISIONS			
LTR	DESCRIPTION	DATE	APR.



NOTES:

		TOLERANCES .XX = ±.020 ANGLE .XXX = ±.010 ±½° .XXXX = ±.005	DRAWN BY ALBERT DOT DATE 9-7-83	SD SYSTEMS DALLAS TEXAS	
			CHECKED BY DATE	VFW-III CONTROL PROCESSORS AND BUFFERS	
		MATERIAL	DESIGN ENGR DATE 9-8-83		
			PROJ ENGR DATE		
		FINISH	APPROVED DATE	SIZE D	CODE IDENT	DRAWING NO. 0300731	REV A
NEXT ASSY	USED ON			SCALE	DO NOT SCALE DRAWING	SHEET 2 OF 5	
APPLICATION							

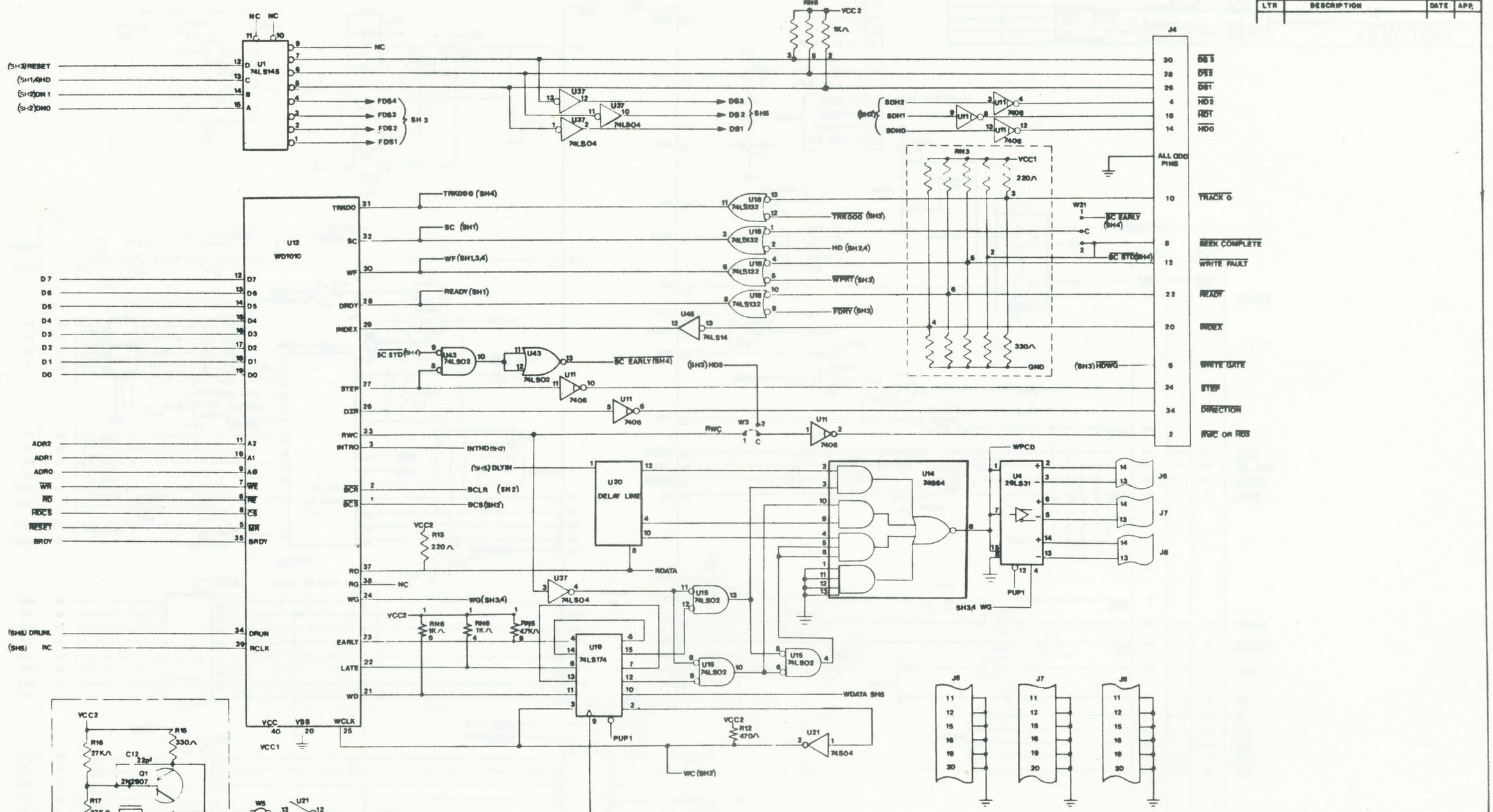
REVISIONS			
LTR	DESCRIPTION	DATE	APP.



NOTES:

NEXT ASSY	USED ON	TOLERANCES .XX = ± .020 ANGLE .XXX = ± .010 ± ½° .XXXX = ± .005	DRAWN BY D. KOCH	DATE 6-29-83	SD SYSTEMS DALLAS TEXAS
APPLICATION		MATERIAL	CHECKED BY	DATE	VFW - III FLOPPY DISK CONTROL
		FINISH	DESIGN ENGR	DATE 7-8-83	
			PROJ ENGR	DATE	
			APPROVED	DATE	SIZE D · CODE IDENT · DRAWING NO. 0300731 · REV A
			SCALE	DO NOT SCALE DRAWING	SHEET 3 OF 5

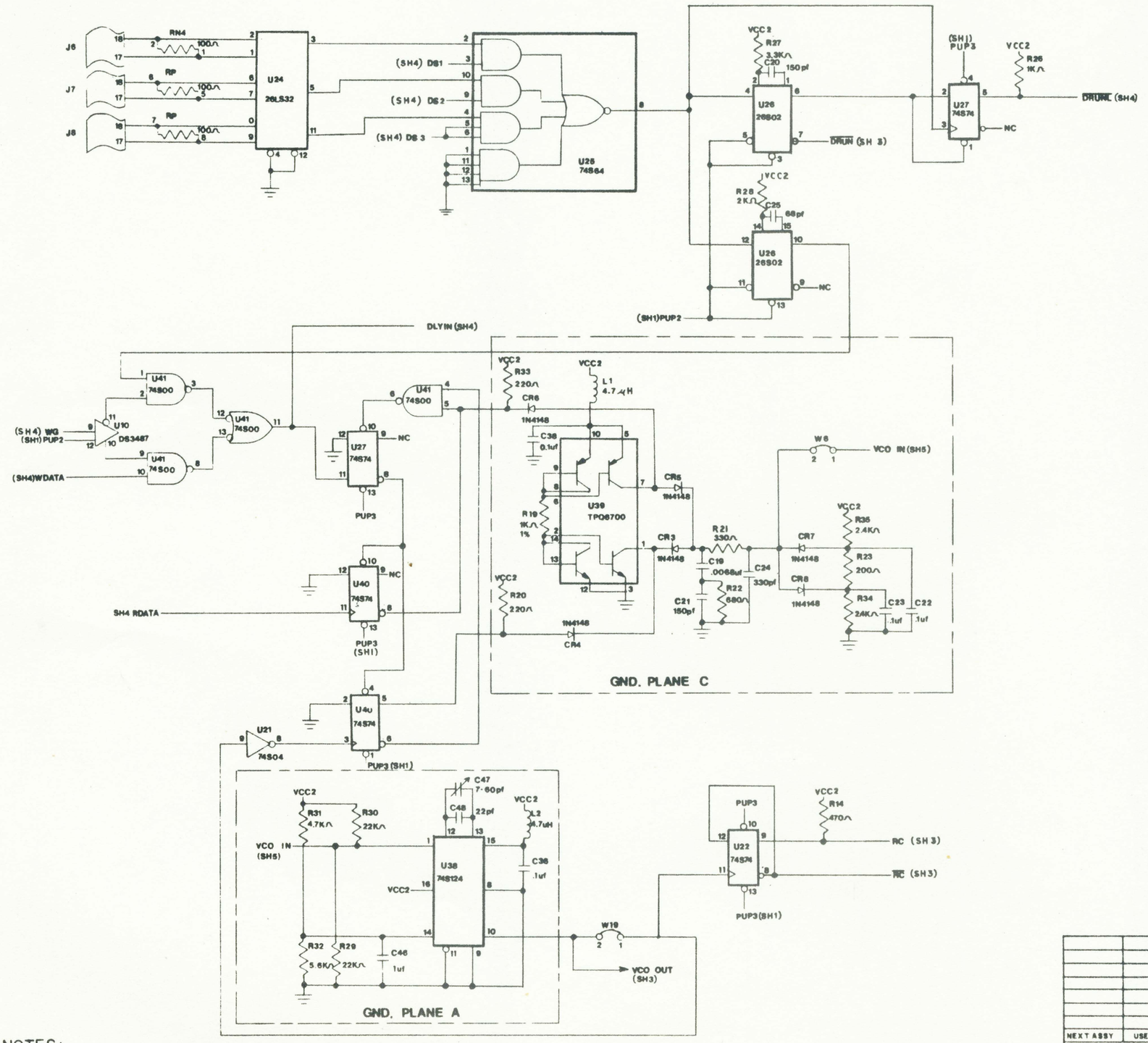
REVISIONS			
LTR	DESCRIPTION	DATE	APP.



NOTES:

		TOLERANCES .XX = ±.020 ANGLE .XXX = ±.010 ± ½° .XXXX = ±.005	DRAWN BY M Dalgarde DATE 9-7-83	SD SYSTEMS DALLAS TEXAS	
		MATERIAL	CHECKED BY DATE	VFW-III WINCHESTER DISK CONTROL	
			DESIGN ENGR Gary Box DATE 9-8-83		
		FINISH	PROJ ENGR DATE		
NEXT ASSY	USED ON		APPROVED DATE	SIZE D	CODE IDENT DRAWING NO. 030073I REV —
APPLICATION				SCALE	DO NOT SCALE DRAWING SHEET 4 OF 8

REVISIONS			
LTR	DESCRIPTION	DATE	APP.



NOTES:

		TOLERANCES .XX = ± .020 ANGLE .XXX = ± .010 ± ½° .XXXX = ± .005	DRAWN BY M GRANDE DATE 8-31-83 CHECKED BY DATE DESIGN ENGR DATE PROJ ENGR DATE	SD SYSTEMS DALLAS TEXAS	
		MATERIAL		VFW-III WINCHESTER DATA SEPARATOR	
		FINISH	APPROVED DATE	SIZE D	CODE IDENT	DRAWING NO. 0300731	REV A
NEXT ASSY	USED ON			SCALE	DO NOT SCALE DRAWING	SHEET 5 OF 5	
APPLICATION							

THIS PAGE INTENTIONALLY LEFT BLANK