SOFTWARE SPECIFICATIONS FOR THE 99/4 DISK PERIPHERAL

CONSUMER GROUP
MAIL STATION 5890
2301 N. UNIVERSITY
LUBBOCK, TEXAS 79414

MARCH 28, 1983

Contents

Chapter 1

INTRODUCTION

This document is intended to give a final operational specification of the ROM based software for the TI 99/4 Disk peripheral. This ROM based software is also called the disk device service routine (DSR).

This hardware design features a Western Digital 1771 Floppy Disk controller. The disk controller has been designed to support any 5.25 floppy disk drive with the minimum head step of 20 milliseconds. More hardware details can be found in the Home Computer Disk Controller Product Specification. Beware that "disk controller" is sometimes used to refer to the software and sometimes to the hardware.

The disk peripheral is ROM based. The ROM code is executed by the TMS 9900 located in the main console. Access to the disk peripheral software is facilitated through the file management system, as specified in the File Management Specification for the 99/4 Home Computer.

Chapter 2

APPLICABLE DOCUMENTS

- File Management Specification for the 99/4 Home Computer (version 2.5, Revised 25 February 1983)
- Home Computer BASIC Language Specification (Revision 4.1, 12 April 1979)
- Home Computer Disk Peripheral Hardware Specification
- Functional Specification for the 99/4 Disk Peripheral (Version 3.0, Revised 28 March 1983)
- GPL Interface Specification for the 99/4 Disk Peripheral (Version 2.0, Revised 28 March 1983)

Chapter 3

SOFTWARE DESIGN CONSIDERATIONS

The disk peripheral software has been designed to support all options facilitated by the file management system, except for the SCRATCH RECORD option. The supported options include:

- Sequential and Relative record (random access) files
- Fixed and Variable length records
- INTERNAL and DISPLAY file types
- OUTPUT, INPUT, UPDATE, and APPEND access modes
- Program LOAD and SAVE functions

Aside from these functions, a separate disk utility cartridge, the Disk Manager, supports the following utility programs:

- Single disk backup
- Disk-to-disk copy/backup
- Disk initialization/formatting
- Disk catalog
- Disk rename
- Disk tests
- File copy
- File protection status
- Selective file deletion

The above mentioned utilities are accessible through menus, similar to the standard program selection menu featured by the 99/4 console. The utilities are currently available in three languages, English, French, and German. Provisions are taken for future expansion of these language capabilities.

More information about the Disk Manager is provided in the Texas Instruments Home Computer Disk Memory System manual.

Chapter 4

FUNCTIONAL OVERVIEW

This section will provide a quick sketch of the functions of the disk peripheral software in each of its implementation levels. Each level uses the features implemented in a lower level, and builds new features with the building blocks provided by the lower levels.

4.1 Level 1 Features

- Disk formatting functions
- Record read/write functions
- Soft error correction functions
- Communication with the FD 1771 chip

Level 1 is the only level that must be familiar with the hardware, thus this implementation level allows for abstraction from the disk hardware. Every higher level will only know the disk as a linear storage device, addressed by physical record number, disk unit number, and read or write operation. For test purposes this level utilizes a DX10 relative record file, each record of which represents one disk sector. This allows for DX10 simulation of the disk peripheral software without a need for availability of the actual hardware. A full disk simulation is operational under the DX10 system GPL simulator (also called a GPL debugger).

Future disk products that may wish to use the current disk software, such as the SA200 design, generally only need to replace this part of the disk software. All the higher levels have been designed to be independent of the actual physical disk structure known at this level, except for the sector size, which is assumed to be 256 bytes. For a single density 128 bytes sector design, the blocking factor would be 2, i.e. every sector as seen from the higher levels take up 2 sectors at Level 1.

4.2 Level 2 Features

* All Level 1 features, plus:

- Data access by filename and physical record displacement
- File creation and deletion
- Mixed hybrid file format

- Dynamically extendible file allocation

This level creates the actual file concept. Each file is known by its name and the displacement of the physical record within the file. Each physical record is defined as one disk sector (256 bytes).

A directory and a bitmap are maintained on every disk to allow for a file and data record management (creation and deletion). The file format available at this level is a mixture of contiguous and noncontiguous file formats, called the mixed hybrid format. Noncontiguous files (fragmented) carry a lot of overhead in the form of pointers to the location of each data record of the file, in case relative access is required. The files on this level are allocated in clusters of contiguous records. These clusters are expanded if possible, whenever new data records are requested. If a cluster can not be expanded any more, a new cluster is started.

4.3 Level 3 Features

* All Level 2 features, plus:

- Fixed and Variable record formats
- Relative and Sequential access methods
- Program and data files
- Internal and ASCII data types

The addition of relative/sequential access methods and fixed and variable record formats completes the disk management software. The software at this level takes care of the blocking of one or more logical records into a physical record. For relative access files it computes the physical record in which the logical record is located, updates that record, and passes the physical record back to Level 2 file update routines.

4.4 Utility Routines

The ROM code also provides the subprograms for special utility routines which do not use the standard file I/O system. These subprograms, which are provided through the GPL subprograms mechanism, are:

- Direct Level 2 file access
- Logical sector/Allocatable Unit (AU) access
- File rename
- File protection modification
- Disk formatting

These subprograms will be provided in the form of GPL subprograms, i.e. assembly language routines located in ROM. These GPL subprograms are specified in the GPL Interface Specification for the 99/4 Disk Peripheral.

Chapter 5

DETAILED OPERATIONAL SPECIFICATIONS

This section will go into the operational specifications in more detail. Record format and access methods, as well as file types will be discussed.

5.1 Record Formats

A file attribute specified at the time of creation is the record format. This attribute describes the logical organization of the file. Two such formats are currently supported by the disk software:

1. Variable length records
2. Fixed length records

5.1.1 Variable Length Records

In applications which must store data structures of unpredictable length, the variable length record format provides an economical way to use the disk space. Since the length of the records is variable, the length of each individual record must be recorded together with the data. This can be done by providing each record with a pointer to the next record, or equivalently, by providing each record with a header byte indicating the number of bytes required to represent the data structure.

Variable length records can also be used to record fixed length data structures in which repeated character strings are expected (i.e. trailing blanks). Such a fixed length structure can become variable by virtue of data compression. However, the current implementation of the disk peripheral software does not perform any data compression.

5.1.2 Fixed Length Records

Records of fixed (constant) length can be used if relative access to a particular records is desired. When the information structures to be recorded are almost or exactly equal in length to the length of the record size, fixed record lengths are appropriate, since there is no overhead associated with record headers or compression indicators. The records consist of an unmodified copy of the data as presented in the user's data buffer. This is obviously more efficient use of bulk storage devices where relative access is supported by the medium.

As we shall see in the next section, fixed length records are also very convenient for relative access, since their length is a known quantity.

5.2 Access Methods

Several methods of accessing data in files are supported. These methods are:

- Physical I/O
- Sequential access
- Relative access

5.2.1 Physical I/O

In the physical I/O method, the data on the disk is considered by the disk software to be organized in blocks of 256 bytes each. Each byte contains any of 256 possible 8-bit combinations, with no attempt to interpret at data transfer time. Any existence of records or files is completely ignored when this access method is used.

The rest of the disk software reduces all access methods to physical I/O, by converting logical record numbers to track/sector data, which can be used to specify the disk sector that is to be transferred by the physical I/O software. Physical I/O has been made available to GPL software only in the form of an assembly language subprogram.

5.2.2 Sequential Access

When the data records in a file are accessed strictly in the order of increasing addresses on the medium, the records are said to be sequentially accessed. This is typically the access method associated with magnetic tape or other linear storage media.

The data transfer parameters do not specify a physical record number. It is implied that the logical record currently indicated in some data transfer pointer, is the one desired. Rewind/Restore operations are implicitly or explicitly done, to set such beginning of the file, prior to first data transfer. As each logical record is transferred, the pointer moves to the first byte of the following one possible the length indicator).

5.2.3 Relative Access

This access method, also called random access, allows data referenced by logical record number. Logical data records may be accessed in any sequence, without regard to the order in which they were written, or their relative position in the file.

Since the disk software must be able to locate a record based solely on its number, relative access can be supported on indexed files or on fixed length files only. Indexed files are not supported in this implementation, so the relative access method is supported for the fixed length files only.

5.3 Library Organization

The library organization implemented in the disk software only supports a single level library. This implies that no file can be of the catalog type (a file pointing to other files). Each file can be identified by a single name, for example:

`DSK1.FILENAME`

specifies a file called FILENAME on the diskette in drive one.

Since this approach prohibits access of a catalog file as such, a semi-catalog file has been created. This file is of the fixed length, relative access type. It contains 128 records, each containing information about the associated catalog entry. This semi-catalog file, which will be described in more detail in SECTION B, can be accessed as:

`DSK1.` or `DSK.VOLNAME.` a general file, without a file name.

Notice that all general file operations have been defined for the catalog file. Only the standard OPEN, READ, and CLOSE calls are supported. All other operations, such as DELETE, RESTORE, WRITE, and EOF, are illegal, and will cause an error to be returned.

5.4 Internal Data Structure Overview

This section describes the internal data structure implemented on the disk peripheral. A description of the external data structure can be found in the File Management Specification for the 99/4 Home Computer.

5.4.1 Physical Device Format

The physical device is logically subdivided in Allocatable Units (AUs). An AU is defined to be an integral number of physical records on the device. The total number of AUs on any device should be less than 4096 (i.e., each AU can be addressed by a 12 bit word). AUs are numbered with zero origin, the first AU is numbered 0.

The physical record length is the block of data read from or written to the device at any one time. For the disk peripheral, both the AU and the physical record are currently equivalent to one disk sector (256 bytes).

5.4.2 Volume Information Block

The Volume Information Block (VIB) is located at AU number 0. If this AU is bad, the entire device is considered bad. This block contains configuration data as required by the disk software, such as available number of AUs, volume identification field, and format information.

Most of the VIB has been allocated for the Allocation Bit Map.

5.4.3 Allocation Bit Map

The Allocation Bit Map is used to indicate the availability of individual allocation units. A binary 1 in a bit position indicates that the allocation unit associated with that bit has been allocated. The first bit (bit 0) is associated with AU 0, the second bit with AU 1, etc. During disk initialization, bits corresponding to system reserved AUs, non-existant AUs, and bad AUs, are set to one. All other bits are set to zero.

5.4.4 File Descriptor Record

The File Descriptor Record (FDR) is used to map the filenames into physical locations of the files on the disk. Each entry contains information such as filename, file type, record type, data type, location, and size information for the file.

5.4.5 File Control Block

The File Control Block (FCB) is a copy of the FDR that is maintained in memory while the file is open. In addition to the FDR information, the FCB contains some up-to-date file information.

5.4.6 File Descriptor Index Record

The (FDIR) enables the system to keep track of the location of each FDR on the disk. It contains alphabetically sorted pointers to each FDR.

The FDIR is located at AU number 1. If this AU is bad, then the entire disk is considered bad.

Chapter 6

DETAILED DISKETTE FORMAT SPECIFICATION

The diskettes used on the Home Computer Disk Peripheral has the following specs:

- Capacity 92160 bytes per disk
  2304 bytes per track
  256 bytes per sector
  9 sectors per track

- Encoding method FM Single Density Recording

- Mini diskette type SA 104 (ANSI standard 5.25)

The specified diskette contains 360 sectors of 256 bytes each. In the remainder of this chapter each sector will be addressed as if the diskette was a linear medium, i.e. track 0 sector 0 will be sector 0 and track 39 sector 8 equals sector 359.

The following section contains a description of the logical structure on each diskette in terms of records.

6.1 Physical Diskette Format

The general diskette format used in the 99/4 Disk Peripheral is the following:

Sector 0 contains the VIB. This block contains general information like:

- Volume Name
- Number of available AUs
- Number of sectors/tracks
- Allocation Bit Map

Sector 1 contains pointers to FDRs. Sector 2 thru 359 contain FDRs and data blocks. The FDRs contain general information about the file, such as:

- File name
- File status data
- File data access blocks

6.1.1 Volume Information Block (VIB)

As mentioned earlier, this block contains general information about the disk. A more detailed description of each entry and its contents will be given in this section.

000 002 004 006 008	DISK VOLUME NAME		001 003 005 007 009
010	TOTAL NUMBER OF AUs		011
012	# SECTORS / TRACK	"D"	013
014	"S"	"K"	015
016	"PROTECTION"	# TRACKS / SIDE	017
018	# OF SIDES	DENSITY	019
020 054	RESERVED		021 055
056 254	ALLOCATION BIT MAP		057 255

Bytes 0-9 contain the volume name of the diskette. The volume name can be any combination of ten ASCII characters, except for the space or period characters and the null character (ASCII code 0). The name is space filled to the right in case of less than 10 characters. The volume name must contain at least one non-space character.

Bytes 10-11 give the total number of AUs on the volume. This datum should match the allocation bitmap.

Byte 12 indicates the number of sectors per track.

Bytes 13-15 contain ASCII code for "DSK", which is used by the disk manager software to check if the diskette has been initialized.

Byte 16 contains the ASCII code for "P" if the diskette is protected (a protected disk is also called a proprietary disk), otherwise this byte contains a >20.

Byte 17 indicates the number of tracks per side.

Byte 18 indicates the number of formatted sides on a disk.

Byte 19 indicates the density of the disk.

Bytes 20-55 are reserved for future expansion like date and time of creation. In the current version of the disk software these bytes are set to zero.

Bytes 56-255 contain the allocation bitmap. This 200 byte map can control up to 1600 256 byte records (total controllable capacity equals 400K bytes), which make it usable for a double density double sided disk. The disk allocation system uses a conventional method of allocating disk space called Bit Maps. Each bit in the bit maps represents one sector on the disk. A logical one in the bit maps means that the corresponding sector has been used. A zero means that the sector is still available.

The volume name can be used as an alternative to the actual disk drive name, i.e. the user can specify a disk drive in either of the following ways:

```
DSK.VOLNAME.FILENAME or DSK1.FILENAME
```

If the volume name is specified, rather than the physical drive number, the system will look in sequence on every drive in the system, until it finds the specified volume. If more than one volume of the same name exists, the drive with the lowest drive identification number will be assigned.

6.1.2 File Descriptor Index Record

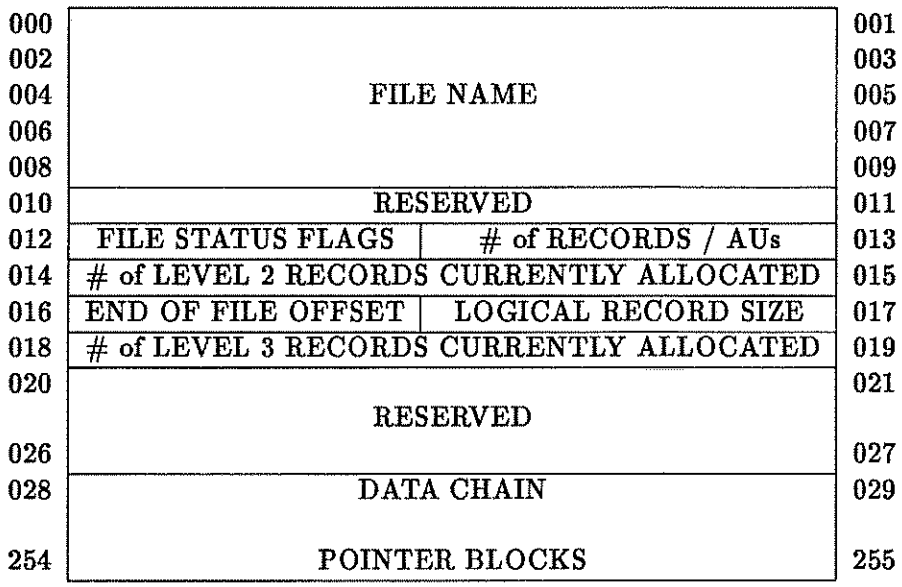
The FDIR contains up to 127 two byte entries, each pointing to a file descriptor record. These pointers are alphabetically sorted according to the file name in the associated file descriptor record. The pointer list starts at the beginning of this block, and ends with a zero entry.

Since the file descriptors are alphabetically sorted in this block, a binary search method can be used to find any given file name, limiting the maximum number of disk searches to seven if more than 63 files are defined. In general if between 2^{N-1} and 2^N files are defined, a file search will take at most N disk searches. To obtain faster directory search response times, the system will prefer to allocate data blocks in the area above AU number 34. Only if no AU can be allocated in that area will the disk data block allocator start allocating blocks in the AU area 2-33.

6.1.3 File Descriptor Records

The FDR contains general information about the file. All the information the system needs to know to access and update the file has to contained in the FDR.

THE PHYSICAL LAYOUT OF THE FDR IS:



Bytes 0-9 contain a file name up to ten characters in length.

Bytes 10-11 are reserved for future expansion of the data chain pointers through linkage to a data chain pointer block chain. In the current version these bytes are always zero.

Byte 12 contains the file status flags. These flags are to be interpreted as follows (bit 0 is the least significant bit):

Bit #	Description
0	Program/data file indicator 0=Data file 1=Program file
1	Binary/ASCII data 0=ASCII data (DISPLAY file) 1=Binary data (INTERNAL or program file)
2	Reserved for future data type expansion
3	PROTECT flag 0=Not protected 1=Protected
4-6	Reserved for future expansion
7	FIXED/VARIABLE flag 0=Fixed length records 1=Variable length records

Byte 13 contains the number of logical records per AU.

Bytes 14-15 contain the number of logical records allocated on level 2 (256 byte records).

Byte 16 contains the EOF offset within the highest physical AU for variable length record files and program files.

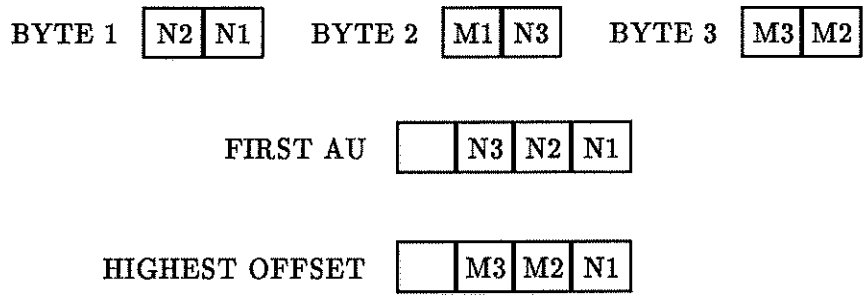
Byte 17 contains the logical record size in bytes. In case of variable record lengths, this entry will indicate the maximum allowable record size.

Bytes 18-19 contain the number of records allocated on level 3. For variable length records, this entry is replaced with the number of level 2 records actually used. (Note: These bytes in this entry are in reverse order.)

Bytes 20-27 have been reserved for future expansion. They will be fixed to 0 in this implementation of disk peripheral software.

Bytes 28-255 contain 3 byte blocks indicating the clusters that have been allocated for the file. The first 12 bits in each entry indicate the address of the first AU in the cluster. The second 12 bits indicate the highest logical record offset in the cluster of contiguous records. This indication has been chosen, rather than the number of data records in the chain, since it reduces the amount of computation required for relative record file access.

The following diagram shows how each 3 byte entry relates to the address of the first AU and the highest logical record offset in the cluster.



6.2 Data File Allocation

A data-file is built out of clusters of contiguous data records. Each data-file can contain up to 76 of those data record clusters. Each data cluster can contain at least one record. The disk software will allocate as many contiguous records as possible upon request. If a new record is requested, and no more records can be added to the current contiguous cluster, a new cluster of contiguous records is started. If 76 of those clusters have been allocated, and a new cluster is requested, the data-records on the disk have become too scattered, and the write-operation in aborted. Worst case this scheme still allows for a minimum of 19K bytes per file (76×256 bytes).

An additional advantage of this scheme is that each physical record within the file can be accessed at random, without any need for big areas of contiguous disk space. This means that as long as the logical records within a file have a fixed length, the file can be accessed either sequentially or at random. Therefore the disk software does not make any distinction between relative record

or sequential files. Note that this has some implications for sequential fixed length record access, since now the record number is being used, rather than the current record number and offset.

For variable length records, the length of the logical record will be stored together with the record itself. This means that, since we do not cross physical record boundaries for any file- or record-type, the maximum record length for a variable length record file is limited to 254 bytes. The end of a AU with variable length records will be marked with an "all ones" byte.

6.3 Program File Allocation

The allocation of a program file is identical to the allocation of a data file. The program segment is blocked into 256-byte records which are stored as a standard data-file. However, the disk software will mark a program file as such, and will not allow data access to program files and vice versa.

To avoid any problems with VDP memory wrap-around, the disk software will also note the actual number of bytes used in the last data record and it will return as many bytes as have been stored originally, even if this number is not a multiple of 256.

Chapter 7

MEMORY USAGE

Since the disk peripheral software will have to use buffer areas to buffer control information, the disk software will allocate part of VDP memory for its internal usage. This memory is continuously allocated and cannot be used by allocation programs, although its size can be changed with a GPL utility routine.

The allocated VDP memory can roughly be subdivided into three usage categories:

1. Drive control information
2. File allocation information
3. Data buffering

Each of these categories will be discussed in more detail in the next section.

7.1 Drive Control Information

In order to be able to control the disk drive hardware, the software has to know what the current status of each disk drive is before it can access it. All of this information is readily available, some through checking the actual current status of the drive.

The power up routine for the disk peripheral also takes initializing the internal track registers to -1. Whenever a drive is accessed, the internal track register will be loaded into the FD1771 chip, unless this value is -1, in which case the head is being restored to track zero, which also reinitializes the FD1771 controller to track zero.

7.2 File Allocation Information

The file allocation information is maintained in the File Control Blocks (FCBs). Each "open" file has an FCB associated with it.

The information maintained in the FCB is identical to the FDR information described in section 6.1.3. In addition, the disk software also maintains some dynamic information about each file. This information is stored in front of the standard information (i.e. the FDR starts at FDB location 6). The total length of an FDB is therefore 512+6=518 bytes, including its 256 byte data buffer (see next section).

The format of the FDR extension is outlined below.

-6	Current Logical Record Offset on Level 2		-5
-4	Physical Record Location of FDR		-3
-2	Logical Record Offset	Drive ID	-1

The meaning of each entry in this additional information block is:

PHYSICAL RECORD LOCATION OF FDR - Points to the physical sector location of the FDR on the disk. Important if we ever want to rewrite the FDR on the disk. Even though not required, it is still maintained during read-only access.

CURRENT LOGICAL RECORD OFFSET ON LEVEL 2 - Contains the physical record offset of the most recently processed physical record. Independent of READ or WRITE operations, this entry always contains the logical offset for Level 2 operation of the Data block that is currently in memory.

Notice that this approach causes fixed length sequential files to be accessed as relative access files on Level 2.

DRIVE ID - contains the drive number (1-3) of the drive on which the associated file resides. If the highest entry is set, the current data block has been modified and will have to be written back to the disk before closing the file, or accessing a new data block.

LOGICAL RECORD OFFSET - This entry contains the offset of the next logical record in the current physical record. If, during READ operations, this entry points to a byte count of >FF, this will indicate an end of record for the current physical record.

This entry is only used for variable length records. For fixed length record access, the actual position AU and the position within that AU is recomputed before every I/O operation. The logical record offset byte is therefore superfluous in this case.

During WRITE operations, this offset always points to the first free byte in the physical record. If the next logical record would leave less than one byte in the current record, a byte count of >FF will be written, and the logical record will be located in the next physical record. Note that the first logical record in a physical record can never cause that physical record to overflow, since the maximum logical record length is 254, and the physical record is 256.

Following this entry are the areas reserved for the FCBs and the data buffers. Each file has its own FCD and data buffer associated with it. To simplify the buffer allocation mechanism, the buffers are not allocated on demand, but rather as soon as a file is opened, an FCB and data buffer are associated with it for the entire "open" life of the file.

The information contained in the records is as follows:

- An ASCII string up to 10 characters in length containing the name of the file in the specified directory slot. For record 0 this is the name of the volume.
- A floating point type code between -5 and +5. A negative value means that the file is write protected. The individual codes are given in Figure 8-1.
- The number of AUs allocated for the file. Record 0 contain the total number of AUs on the disk.
- The number of bytes per logical record. For a program file this program entry is 0. Record 0 contains the remaining number of AUs in this entry.

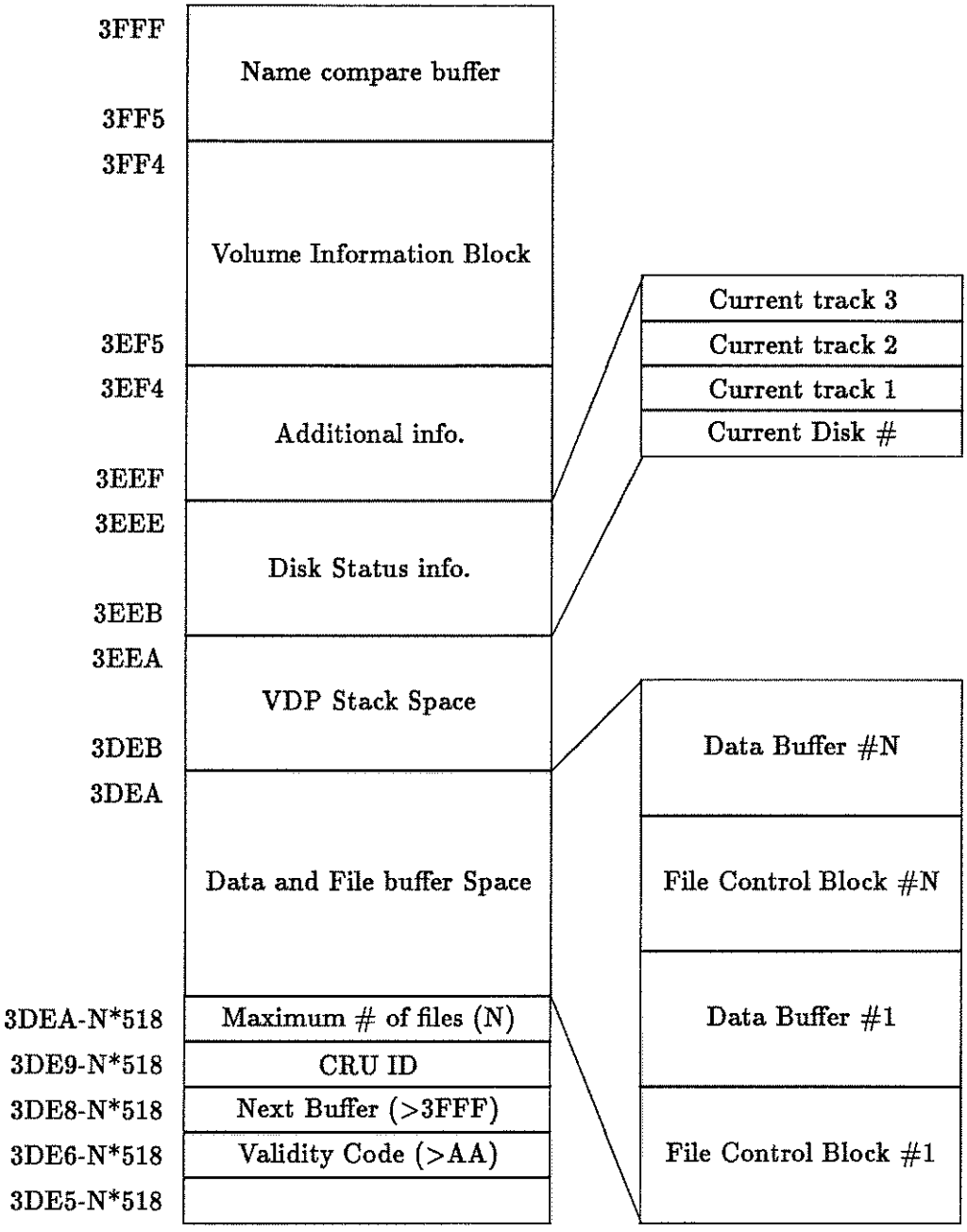


Figure 7.1: Disk VDP Memory Buffer Layout

7.3 Data Buffering

For the purpose of data buffering, the disk software will maintain one 265-byte buffer for each "open" file, located directly above the FCB buffer.

One of the VDP RAM buffers is continuously assigned to VIB processing. In case more than one drive is used for WRITE mode, the bit maps will be moved in and out of this buffer as demanded by the disk software. For every bit map operation, this buffer will be used to access the Volume Information Block.

Every Level 3 WRITE operation to a file will ultimately be passed on to Level 2 as a physical sector WRITE. To minimize the number of disk accesses, a flag will be set to indicate that the current data buffer has been modified. The data buffer will only be physically written to the disk if the next physical record access involves another physical record than the one currently residing in the data buffer. If the file is closed for further access, the last data buffer will be written on to the disk if required.

7.4 VDP Memory Layout

The VDP memory layout used for the disk peripheral in 16K VDP RAM system is outlined in figure 7-1. The memory block in this figure is reserved on power up by special power up code. The length of the entire area depends upon the maximum number of files that are allowed to be open at the same time. Each extra file takes up an extra 518 bytes.

The maximum number of files allowed to be open at the same time, which is initially 3, can be varied between 1 and 16 by calling a special GPL subprogram.

As for every peripheral, the disk peripheral identifies the area it reserved through its CRU address, which is unique for every peripheral. The area is validated with an >AA code followed by the address of the previous top of memory. Since the disk peripheral has the highest priority on power up, this entry will always point to the actual top of memory of the machine. However, the disk software does not use this fact, and will work equally well on other CRU locations.

The first entry behind the CRU ID contains the number of files for which the area has been reserved. This number directly determines the length of the reserved memory area.

The VDP stack area is used to simulate a stack machine on the TMS 9900. This gives the programmer the advantage of being able to use the multi-level stack oriented CALL/RETURN mechanism, rather than the single level BL mechanism used in the TMS 9900. The stack can also be used to PUSH and POP registers to and from, thereby greatly simplifying register usage.

The disk status information area is used to save the current track numbers of the three drives and the most recently accessed drive.

The additional information area (6 bytes) is a leftover from a previous implementation and serves no practical purpose any more. However, since there was a risk involved in removing this area, it has been left in. For the same reason the stack area has not been optimized to its minimum size.

The Volume Information Buffer is strictly reserved for VIBs. Only one buffer in the entire system is reserved for this purpose, although for future implementations one of the file buffers might be used to store two or more VIBs if not all reserved files are in use.

At the top of memory, an 11-byte buffer is reserved which is used for name comparison. Every high level entry point automatically saves the drive number and the 10-character file name in this entry. If less than 10 characters are available, the buffer is automatically padded with spaces.

7.3 Data Buffering

For the purpose of data buffering, the disk software will maintain one 265-byte buffer for each "open" file, located directly above the FCB buffer.

One of the VDP RAM buffers is continuously assigned to VIB processing. In case more than one drive is used for WRITE mode, the bit maps will be moved in and out of this buffer as demanded by the disk software. For every bit map operation, this buffer will be used to access the Volume Information Block.

Every Level 3 WRITE operation to a file will ultimately be passed on to Level 2 as a physical sector WRITE. To minimize the number of disk accesses, a flag will be set to indicate that the current data buffer has been modified. The data buffer will only be physically written to the disk if the next physical record access involves another physical record than the one currently residing in the data buffer. If the file is closed for further access, the last data buffer will be written on to the disk if required.

7.4 VDP Memory Layout

The VDP memory layout used for the disk peripheral in 16K VDP RAM system is outlined in figure 7-1. The memory block in this figure is reserved on power up by special power up code. The length of the entire area depends upon the maximum number of files that are allowed to be open at the same time. Each extra file takes up an extra 518 bytes.

The maximum number of files allowed to be open at the same time, which is initially 3, can be varied between 1 and 16 by calling a special GPL subprogram.

As for every peripheral, the disk peripheral identifies the area it reserved through its CRU address, which is unique for every peripheral. The area is validated with an >AA code followed by the address of the previous top of memory. Since the disk peripheral has the highest priority on power up, this entry will always point to the actual top of memory of the machine. However, the disk software does not use this fact, and will work equally well on other CRU locations.

The first entry behind the CRU ID contains the number of files for which the area has been reserved. This number directly determines the length of the reserved memory area.

The VDP stack area is used to simulate a stack machine on the TMS 9900. This gives the programmer the advantage of being able to use the multi-level stack oriented CALL/RETURN mechanism, rather than the single level BL mechanism used in the TMS 9900. The stack can also be used to PUSH and POP registers to and from, thereby greatly simplifying register usage.

The disk status information area is used to save the current track numbers of the three drives and the most recently accessed drive.

The additional information area (6 bytes) is a leftover from a previous implementation and serves no practical purpose any more. However, since there was a risk involved in removing this area, it has been left in. For the same reason the stack area has not been optimized to its minimum size.

The Volume Information Buffer is strictly reserved for VIBs. Only one buffer in the entire system is reserved for this purpose, although for future implementations one of the file buffers might be used to store two or more VIBs if not all reserved files are in use.

At the top of memory, an 11-byte buffer is reserved which is used for name comparison. Every high level entry point automatically saves the drive number and the 10-character file name in this entry. If less than 10 characters are available, the buffer is automatically padded with spaces.

Chapter 8

CATALOG FILE ACCESS

In order to enable access to a disk catalog from a user or application program, the CATALOG file has been added to the disk software.

The CATALOG file is a data file of the INTERNAL/FIXED type. The record length for this file is 38 bytes. DSKx. or DSK.volname. A standard datafile, but without a name.

The CATALOG file contains 128 records of 38 bytes. The data in this file is stored in an INTERNAL format (i.e. a length byte followed by a data-item). Each record contains four of those data-items:

- An ASCII string of up to 10 characters or a null-string
- Three numerics in standard 8-byte floating point notation

Record 0 contains information about the volume itself, whereas records 1-127 contain information about specific slots in the catalog. Record 1 contains information about file 1, 2 about file 2, etc.

If a specified catalog slot is empty, the file name will be the null-string, and all numeric entries will contain floating zeroes. The following figure shows the codes found in the CATALOG file.

1. Volume info record or empty catalog entry
2. DISPLAY/FIXED data file
3. DISPLAY/VARIABLE data file
4. INTERNAL/FIXED data file
5. INTERNAL/VARIABLE data file
6. Memory image file (program)

Table 8.1: CATALOG Type Codes