Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/Radar, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Radar). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

In order to generate this file, I need to scan the pages, split the double pages and remove any edge marks such as punch holes, clean up the pages, set the relevant pages to be all the same size and alignment. I then run Omnipage (OCR) to generate the searchable text and then generate the pdf file.

Hopefully after all that, I end up with a presentable file. If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you personally – I know that I would have liked to have found some of these files years ago – they would have saved me a lot of time !

Colin Hinson
In the village of Blunham, Bedfordshire.

SOFTWARE SPECIFICATION

FOR THE

99/4 DISK PERIPHERAL

Consumer Group
Mail Station 5890
2301 N. University
Lubbock, Texas 79414

TEXAS INSTRUMENTS
INCORPORATED

Date: March 28, 1983
Version 2.0

TABLE of CONTENTS

Paragraph Title

LIST of FIGURES

SECTION 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).

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

SECTION 2

APPLICABLE DOCUMENTS

File Management Specification for the TI-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)

SECTION 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 rename
* File protection status
* Selective file deletion

The above mentioned utilities are accessable through menus, similar to the standard program selection menu featured by the TI-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.

SECTION 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 FD1771 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. Smaller sector sizes can be easily supported by blocking the physical sectors in such a way that the total still adds up to 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 extendable 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 file and data-record management (creation and deletion). The file format available at this level is a mixture of contiguous and non-contiguous file formats, called the mixed hybrid format. Non-contiguous 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. In order to combine the advantage of the flexible allocation of non-contiguous files with the low overhead and easy access of contiguous files, 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 cannot 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 the 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 subprogram mechanism, are:

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

The 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.

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 the 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 subprogram mechanism, are:

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

The 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.

structures to be recorded are almost or exactly equal in length to the record size, fixed length records 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 a 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 access 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 the 256 possible 8-bit combinations, with no attempt to interpret at data transfer time. Any existance of records or files is completely ignored when this access method is being used.

The rest of the disk software reduces all access methods to physical I/O, by converting logical record numbers to physical 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 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 and 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 pointers to the beginning of the file, prior to the first data transfer. As each logical record is transferred, the pointer moves to the first byte of the following one (possibly the length indicator).

5.2.3 Relative Access.

This access method, also called random access, allows data reference 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 record files only. Indexed files are not supported in this implementation, so the relative access method is supported for fixed length record 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

which specifies a file called "filename" on the diskette in disk drive 1.

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. The semi-catalog file, which will be described in more detail in section 8, can be accessed as:

DSK1. or DSK.volname.

as a general file, with an empty filename.

Notice that not all general file operations have been defined for the catalog file. Only the standard OPEN, READ, and CLOSE calls are supported. All the 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 TI-99/4 Home Computer.

5.4.1 Physical Device Format.

The physical device is logically subdivided into 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 in a 12-bit word). AUs are numbered with zero origin, (i.e. the first AU is number 0).

The physical record length is the block of data read from or written to the device at one time. For the disk peripheral, both the AU and the physical record are currently equivalent to one diskette 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 will be 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.

A major part 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 allocation unit 0, the second bit (bit 1) with allocation unit 1, etc. During disk initialization, bits corresponding to system-reserved AUs, non-existing AUs, and bad AUs, are set to 1. All other bits are set to zero.

5.4.4 File Descriptor Record.

The File Descriptor Record (FDR) is used to map 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 File Descriptor Record 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 File Descriptor Index Record enables the system to keep track of the location of each file descriptor record on the disk. It contains alphabetically sorted pointers to each File Descriptor Record.

The File Descriptor Index Record is located at AU number 1. If this AU is bad, the entire disk is considered to be bad.

SECTION 6

DETAILED DISKETTE FORMAT SPECIFICATION

The diskette used on the Home Computer Disk Peripheral has the following specifications:

* 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 a total of 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 designated "sector 0"; 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 TI-99/4 Disk Peripheral is the following:

Sector 0 contains the Volume Information Block (VIB). This block contains general information about the diskette like:

* Volume Name
* Number of available AUs
* Number of sectors/track
* Allocation Bit Map

Sector 1 contains pointers to file descriptor records.

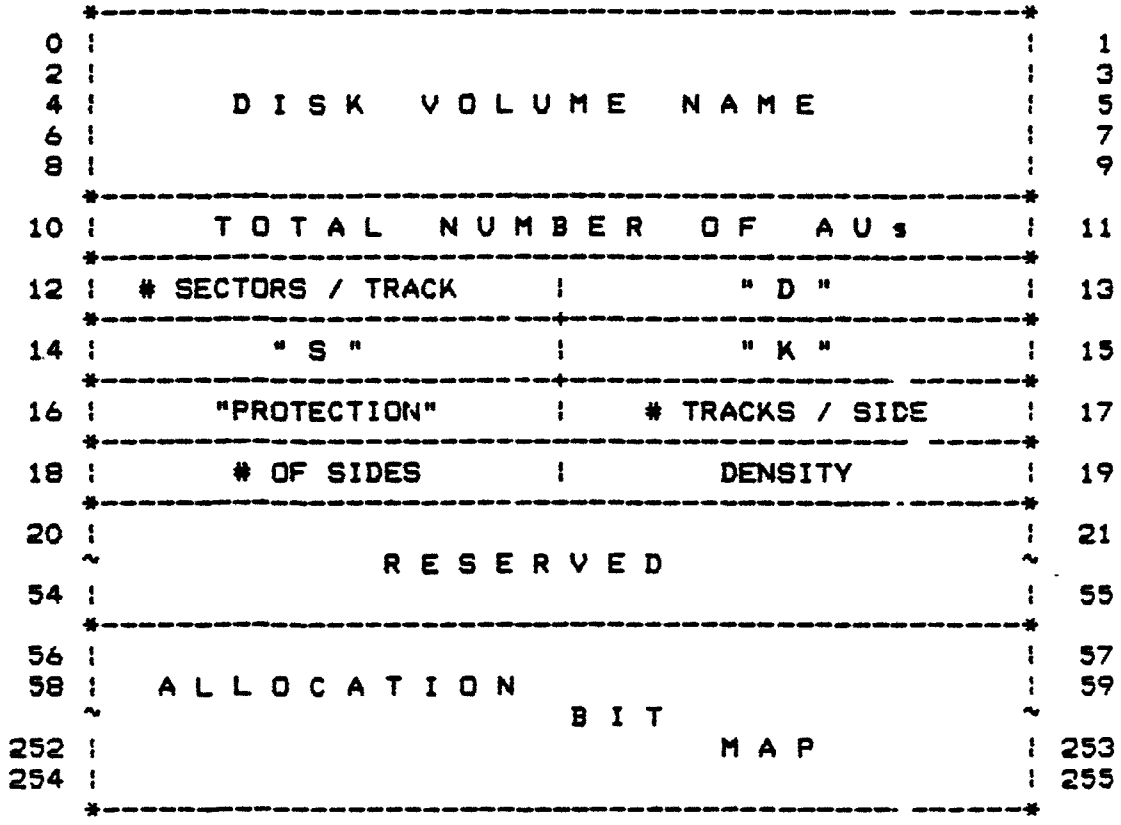
Sector 2 thru 359 contain File Descriptor Records and data blocks.

The File Descriptor Records contain general information about the file, such as:

* File name
* File status data
* File data access blocks

6.1.1 Volume Information Block.

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



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 allocation units (AUs) on the volume. This datum should match the allocation bit map.

Byte 12 indicates the number of sectors per track.

Bytes 13-15 contain the 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 the diskette.

Byte 19 indicates the density of the diskette.

Bytes 20-55 are reserved for future expansions 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 bit map. This 200 byte map can control up to 1600 256-byte records (total controllable storage capacity = 400K bytes), which make it useable for a double density, double sided diskette. 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 allocated. 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 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.

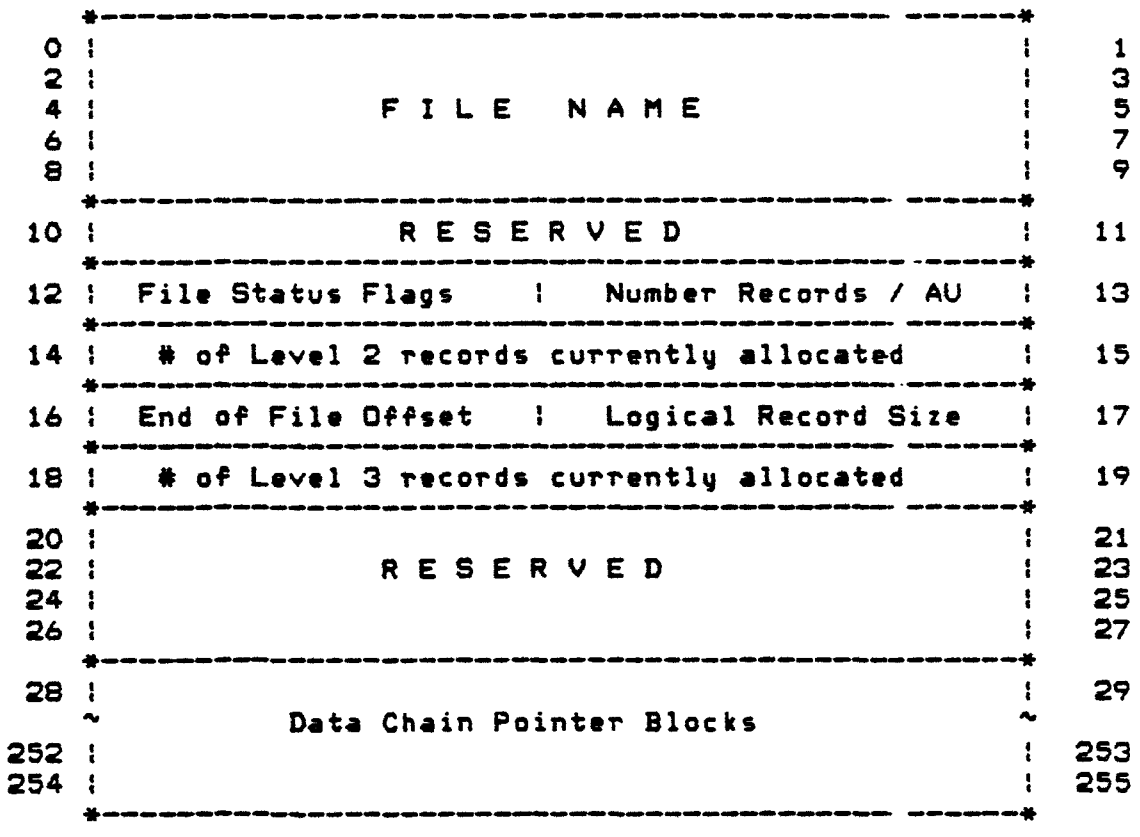
The File Descriptor Index Record contains up to 127 two byte entries, each pointing to a file descriptor record. These pointers are alphabetically sorted according to the filename in the associated file descriptor record. The pointer list starts at the beginning of this block, and ends with a zero entry.

Since the file descriptors are alphabetically sorted in this block, a binary search method can be used to find any given filename, limiting the maximum number of disk searches to 7 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 File Descriptor Record (FDR) contains general information about the associated file. All the information the system needs to know to access and update the file has to be contained within the file descriptor record.

The physical layout of an FDR is:



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

Bytes 10-11 are reserved for future extension of the number of data chain pointers through linkage to a data chain pointer block chain. In the current version these bytes are always 0.

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 length records, this entry will indicate the maximum allowable record size.

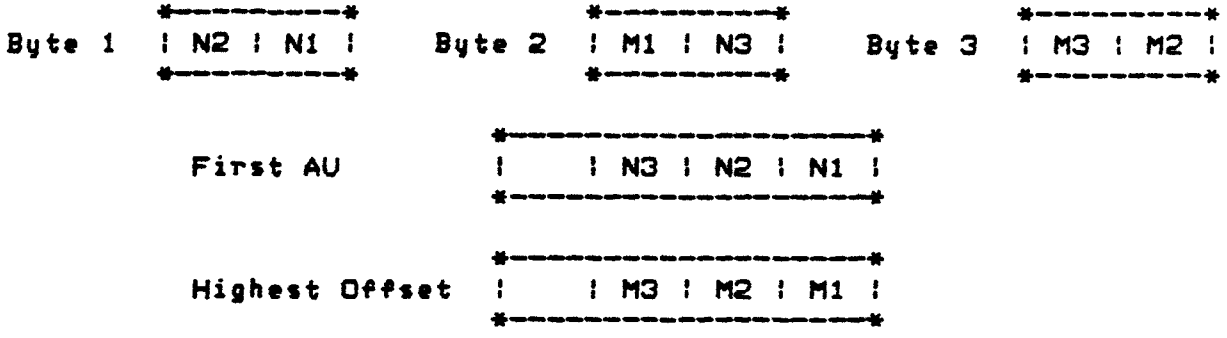
Bytes 18-19 contain the number of records allocated on Level 3. For variable length records, this entry is replaced with the number of Level 2 records actually used. (NOTE: The bytes in this entry are in reverse order.)

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 three 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 three 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 is 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 an 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 exactly as many bytes as have been stored originally, even if this number is not a multiple of 256.

SECTION 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 application 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 sections.

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.

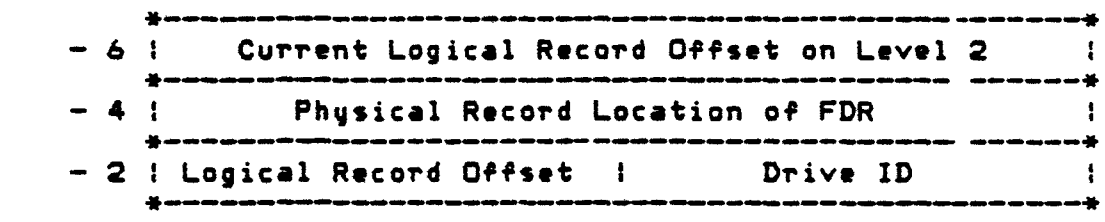
The power up routine for the disk peripheral also takes initializing the internal track registers to -1. Whenever a drive is accessed, the internal track register will be loaded into the FD1771 chip, unless this value is -1, in which case the head is being restored to track 0, which also re-initializes the FD1771 controller to track 0.

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 FDR 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.



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

Drive ID - Contains the drive number (1-3) of the drive on which the associated file resides. If the highest bit of this 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 length is 256.

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. Independant of READ or WRITE operations, this entry always contains the logical offset for Level 2 operation of the datablock that is currently in memory.

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

7.3 Data Buffering

For the purpose of data buffering, the disk software will maintain one 256-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 bitmap operation, this buffer will be used to access the Volume Information Block.

Every Level 3 WRITE operation to a file will ultimately be passed onto 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 onto the disk if required.

7.4 VDP Memory Layout

The VDP memory layout used for the disk peripheral in a 16K VDP RAM system is outlined in Figure 7-1. The memory block in this figure is reserved upon 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.

Following this entry are the areas reserved for the FCBs and the data buffers. Each file has its own FCB 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 databuffer are associated with it for the entire "open" life of the file.

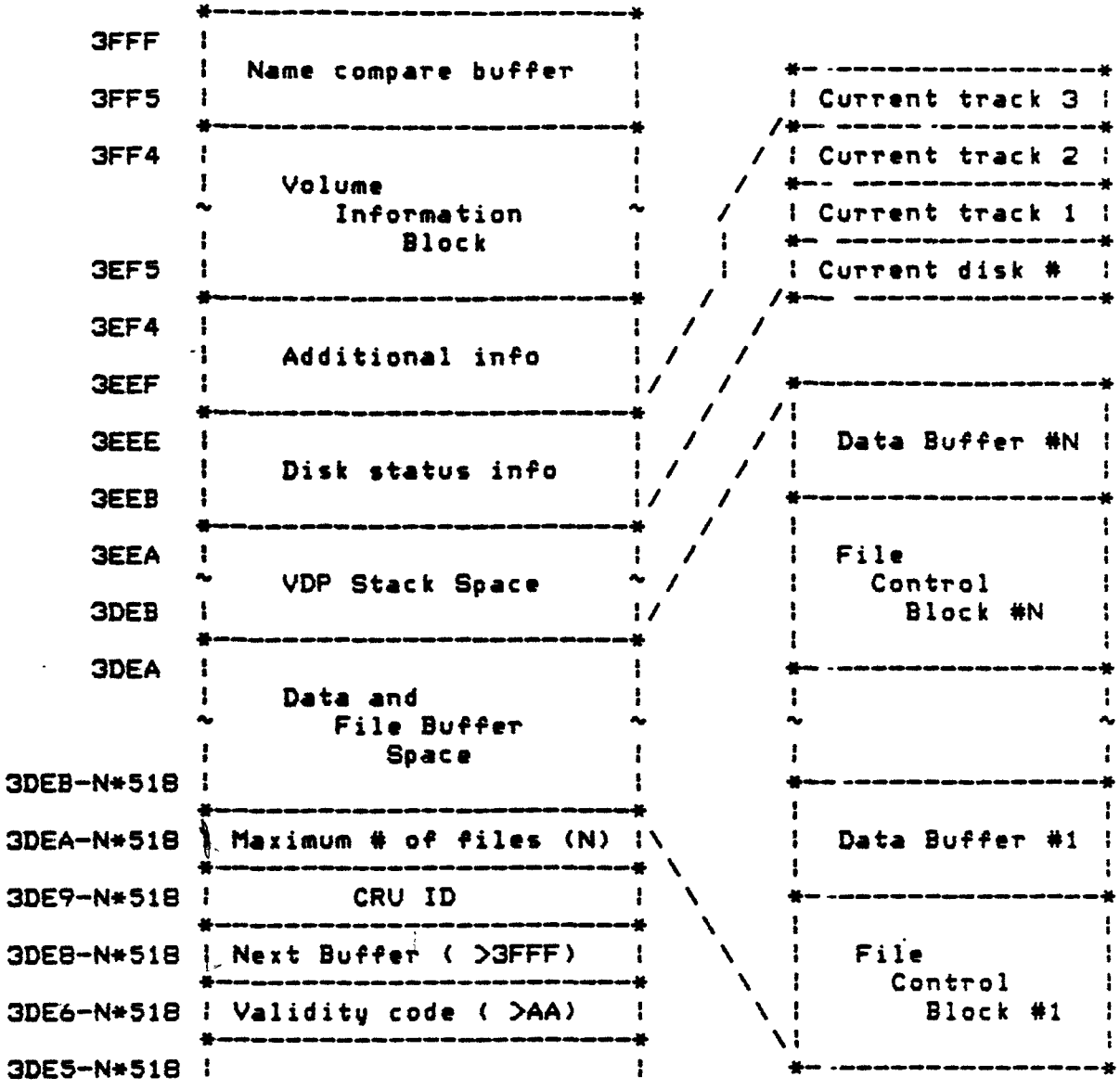


Figure 7-1 Disk VDP Memory Buffer Layout

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 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 filename in this entry. If less than 10 characters are available, the buffer is automatically padded with spaces.

SECTION 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 datafile of the INTERNAL/FIXED type. The record length for this file is 38 bytes. The file can be accessed under the name:

DSKx. or DSK.volname.

as 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 through 127 contain information about specific slots in the catalog. Record 1 contains information about file 1, record 2 about file 2, etc.

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 contains the total number of AUs on the disk.
* The number of bytes per logical record. For a program file this entry is 0. Record 0 contains the remaining

number of AUs in this entry.

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

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

Figure 8-1 CATALOG Type Codes