GPL INTERFACE SPECIFICATIONS FOR THE 99/4 DISK PERIPHERIAL

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

March 28, 1983

Contents

VERSION 2.0

Chapter 1

Introduction

The information contained in this document gives a complete specification of the interface between the 99/4 Disk Peripheral and the GPL interpreter.

NOTE

Throughout this document hexadecimal numbers are indicated by a preceeding 0 or a preceeding >. Therefor the numbers 010, >10, and decimal 16 are the same value.

The items in the transfer blocks which are enclosed braces {} are items returned by the subprogram.

Chapter 3

DISK DSR LEVEL CONCEPT

The disk DSR has been developed as a 3 Level software package, each level defines distinct options that can be used on higher levels. This section will give a brief overview of the levels used and of the features built-in to each level.

The three levels used are:

LEVEL 1 Definition of the basic disk funtions like sector READ/WRITE. Head control, drive selection, and track formatting.

LEVEL 2 Definition of the "file" concept. each file is addressable its' name and an offset of a 256-byte block relative to the beginning of the file.

LEVEL 3 Extension of the file concept to the level given in the file magement specifications. Introduction of the logical FIXED or VARIABLE length records, RELATIVE record or SEQUENTIAL files.

The following sections will each describe a level and the related subprogram calls to it.

3.1 Level 1 Subroutines

The lowest level routines in the DSR are called Level 1 Subroutines. These routines make the higher levels independent of the physical disk medium, e.g. changing the disk software for double density would only involve changing the subroutines on this level, as long as the physical sector size remains at 256 bytes.

The folowing routines are availble on this level:

- SECTOR READ/WRITE

- FORMAT DISK

The following sections will contain a description of these routines and thier call requirements. All parameters will be tranferred throught the FAC block in CPU RAM. This block is located in CPU RAM satarting at relative location 04A (currently >834A).

3.1.1 Sector READ/WRITE - SUBPROGRAM 010

The transfer block for this subprogram is:

004A	{ Sector Number }		004B
004C	Unit #	READ/WRITE	004D
004E	VDP Buffer start address		004F
0050	Sector Number		0051

ERROR CODES ARE RETURNED IN CPU LOCATION 050

The meaning of each entry is:

SECTOR NUMBER numer of the sector to be written or read. Sectors are addressed as logical sectors (0-359 for a standard minifloppy) rather than a track and record number, wich would require the knowledge of the physical layout of the floppy disk. The sector number has to be given in CPU RAM locations 050-051, and will be returned in CPU RAM locations 04A-04B.

UNIT # Indicates the disk drive that the operation is to be preformed. This entry has to be either 1,2, or 3.

READ/WRITE Indicates the direction of data flow.
0=WRITE
≠ 0=READ

VDP BUFFER START ADDRESS Indicates the start of VDP data buffer for data transfer. The number of bytes transferred will always be 256.

ERROR CODES WILL BE RETURNED IN CPU LOCATION 050.

3.1.2 Disk Formating - SUBPROGRAM 011.

This routine will format the entire disk on a given unit unless the disk in the unit has been hardware write protected. It can use any VDP memory, starting at the location given in the transfer block. The amount of memory used depends on the disk format. For the current single density format, the buffer memory used is a nominal 3125 bystes. This can vary with a disk motor speed to a maximum of 3300 bytes. To be compatible with double density versions of the disk controller, the minimum buffer size must be 8K.

The transfer block for this subprogram is:

004A	{ # of sectors/disk }			004B
004C	DSR Ver	Unit #	# of tracks	004D
004E	VDP Buffer start address			004F
0050	Density		# of sides	0051

The meaning of each entry is:

# OF SECTORS/DISK Is returned by the routine to provide comatibility between the current controller version and future (double density or SA200) versions.

DSR VERSION This is the MSNibble

- **0** indicates the format requires nothing special and can be done by any version of the DSR.
- **1** indicates the format requires the 2nd version of the DSR for 1 of 2 reasons. It may be a double sided format is requested or it may be because a # of tracks other than 35 or 40 is requested.
- **2** indicates the format requires features that are not on the currrent DSRs. (Density and perhaps double tracking if it is available on the next DSR.)

UNIT # Indicates the disk drive the operation is to be preformed. This entry must be 1,2, or 3.
This is the LSNibble

# OF TRACKS Indicates the number of tracks to be formated. In the current version this number must be 35 or 40!!! Upon return, this entry contains the number of sectors/tracks.

VDP BUFFER ADDRESS Indicates the start of the VDP buffer that can be used by the disk controler to write tracks.

DENSITY Only single is supported at this time.

# OF SIDES Indicates the number of sides to format.

3.2 Level 2 Subroutines

The Level 2 Subroutines are those that use the concept "file" rather than "logical record number". Notice that the file concept on this level is limited to an abstract type of file which has no properties such as "program file" or "data file". A file on this level is merely a collection of data, stored in logical blocks of 256 bytes each.

The logocal blocks on this level are accessed by file name and logical block offset. This offset starts block 0 and ends with block N-1 with a file length of N blocks.

3.2.1 Modify File Protection

The transfer block for this subprogram is:

004C	Unit #	Protect code	004D
004E	Pointer to the file name		004F

This protection bit for the indicated file will be set or reset according to the information given in CPU location 04D:

0= Reset the file protect bit. The file is no longer protected against modification/ deletion.

OFF= Set the file protection bit. Disallow SAVE and OPEN for OUTPUT, APPEND, or UPDATE mode.

3.2.2 File Rename Routine - SUBPROGRAM 013

The transfer block for this subprogram is:

004C	Unit #	Unused	004D
004E	Pointer to new name		004F
0050	Pointer to old name		0051

Both pointers, located at 04E and 050 in CPU RAM, point to the VDP location of the first character of the file name. The first pointer points to the new name, the second one to the original file name. Each name is left adjusted in a 10 character field, filled with spaces. Each name is located in VDP RAM and has to be a legal file name. No checks are being made to ensure legality of the name.

Since the rename has to be done on the same disk, only one unit # entry is required. This unit # is located in CPU RAM location 04C.

Error codes are returned at location CPU RAM 050. The error codes returned are identical to the standard file management error codes, i.e. only the upper three bits of the error byte are significant.

3.3 Direct File Access Routines

The direct file access routines can be used for accessing disk files without paying attention to the type of disk file (PROGRAM or DATA). The level of access is equivalent to the Level 2 disk software, which means that the access is performed on the basis of straight AUs. However, Level 3 information can be passed at file open time.

Since the input and output direct access subprograms can be used together to copy files, the user has to be careful with the information returned by the input file subprogram, since some of this information may be used by the output file subprogram.

3.3.1 Access Direct Input Files - SUBPROGRAM 014

The transfer block for this subprogram is:

004C	Unit #	Access Code	004D
004E	Pointer to File Name		004F
0050	Addt'l info		

The meaning of each entry is:

Unit # indicates the disk drive on which the operation is to be performed. Must be 1,2, or 3.

Access code an access code is used to indicate which function is to be performed, since this subprogram contains multiple functions. The following codes are used:

0→ Transfer file parameters. This will transfer Level 2 parameters to the additional information area (6 bytes). It also passes the number of AUs allocated for the file.

N→ When N is not equal to 0, this indicates the number of AUs to be read to a given file, starting at the AU indicated in the additional information block.

After the READ is complete, This entry contains the actual number of AUs read. If all AUs have been read, this entry will be 0.

Pointer to file name Contains a pointer to the first character of the 10 character filename., possibly padded to the right with spaces. This filename is NOT checked by the disk software.

Additional info Points to a 10 byte location in CPU RAM containing information for direct disk access:

Table 3.1: Additional Information Block

X	VDP Buffer Start Address	
X+2	# of first AU	
X+4	Status Flags	# records/AU
X+6	EOF offset	Log.Rec.Size
X+8	# of Level 3 records alloc.	

The VDP Buffer start address is where the information read from the disk is stored. The Buffer has to be able to store at least N+256 bytes, in which N is the access code.

The # of the first AU entry is the AU number at which the read should begin. If the access code =0 (parrameter passing), the total number of AUs allocated for the file wil be returned.

The remaining 6 bytes are explained in the Software Spec. for the 99/4 Disk Peripheral. The user must be careful when changing these bytes, since they directly affect Level 3 operation. If they are not modified consistently, weird things may happen.

ERROR CODES ARE RETURNED AT LOCATION 050 IN CPU RAM

3.3.2 Access Direct Output File - SUBPROGRAM 015

The transfer block for this subprogram is:

004C	Unit #	Access Code	004D
004E	Pointer to the file Name		004F
0050	Addt'l Info		

The meaning of each entry is:

Unit # is the drive being used, must be 1, 2, 3.

Access Code an access code is used to show wich function is to be preformed, since multiple functions are combined, the following codes are used:

0→ Create file and copy Level 3 parameters from addt'l info area.

N→ If N ≠ 0, the number of AUs to be written to the file, starting at the AU in the Addt'l Info Block.

Pointer to file name Contains a piointer to the first character of a 10 character filename. (Maybe padded with 0's). The filename is NOT checked ny the disk software.

Additional Info Points to a 10 byte location in CPU RAM containing addt'l info for direct disk access:

Table 3.2: Additional Information Block

X	VDP Buffer Start Address	
X+2	# of first AU	
X+4	Status Flags	# records/AU
X+6	EOF offset	Log.Rec.Size
X+8	# of Level 3 records alloc.	

The VDP Buffer start address is where the information read from the disk can be stored. It must be able to store N*256 bytes, Where N is the access code.

The # of first AU entery is the AU number at which the read should begin. If it is =0 (parameter passing), the total number of AUs to be used must be indicated.

The remaining 6 bytes are explained in Software Specs. for the 99/4 Disk Peripheral.

ERROR CODES ARE RETURNED TO LOCATION 050 IN CPU RAM

3.4 Buffer Allocation Routine - SUPROGRAM 016

The argument for this subprogram is the number of file buffers to be used. This argument is given in FAC+2 (CPU location 04C).

The effect of this routine is that an attempt is made to allocate enough VDP space for the disk usage to facilitate the simultaneous opening of the given number of files. This number is between 1-16.

The disk software automatically relocates all buffers that have been linked in the following manner:

Byte 1 Validation code

Byte 2-3 Top of memory before allocation of this buffer.

Byte 4 High byte of CPU address for the given buffer area. for programs this byte is 0.

The linkage to the first buffer area is made through the current top of memory, given in CPU location 070 (currently >8370).

The top of memory is also automatically updated after successful completion of this subprogram.

A check is made that the current request leaves at least 0800 bytes of VDP space for the screen and data storage. If this is not the case, or if the total number of buffers requested is 0 or >16, the requst is ignored and an error code will be indicated in CPU location 050 (currenly >8350).

Successful completion is shown by a 0 byte in CPU location 050. A nonzero here means tuff luck.