GROSVENOR SOFTWARE

2 Beacon Close, Seaford, East Sussex, BN25 2JZ.

DRAGONDOS

A PROGRAMMER'S GUIDE

DRAGONDOS - A PROGRAMMER'S GUIDE

(c) 1985 Grosvenor Software

ABBREVIATIONS:

$	Hexadecimal
#	Number
FSN	File sector # (relative to file)
LSN	Logical sector # (relative to disk)
FIB	File Information Block

Dragon DOS supports 5 1/4 inch drives, single or double sided, with 40 or 80 tracks. 18 sectors per track, 256 bytes per sector. defaults: single sided, 40 tracks.

Directory track: 20
Alternate Directory track: 16
Number of "FIB's": 10 (i.e. max number of open files)

FLOPPY DISK CONTROLLER CHIP: WD 2797

Addresses of 2797 registers:

$FF40	Command reg.
$FF41	Track reg.
$FF42	Sector reg.
$FF43	Data reg.
$FF48	Latch reg.

DIRECTORY TRACK FORMAT:

Sector 1 - Bit Map etc.

bytes 0-179	bit map of free sectors (bit=1 = free)
byte 252	No. of tracks
byte 253	Sectors/side
byte 254	Complement of no. of tracks
byte 255	Complement of sectors/side

Sectors 3 onwards - DIRECTORY ENTRIES

Each sector of the directory track contains up to 10 DIR entries of 25 bytes each.

Directory entry layout:

+0	flag
+1-8	file name
+9-11	file name extension
+12-14	#1 extent
+15-17	#2 extent
+18-20	#3 extent
+21-23	#4 extent
+24	bytes in last sector ($00 = 256)

DIRECTORY TRACK cont.

Format of each extent entry

+0,1 first LSN in this extent
+2 Length (no. of sectors)
unused entries have length=0

Format of extra extent block

+0 flags
+1 extents (7 * 3 bytes)
+22 unused
+24 pointer

Format of flags byte

bit 0 sector unit
0=file name, 1=extension
bit 1 protection
1=protected
bit 3 end of dir
1=end
bit 5 continued?
1=more extents
bit 7 valid?
1=not valid

DOS MEMORY MAP

PAGE ZERO VARIABLES:

ADDRESS hex	CONTENTS
EA	Command
EB	Drive # (1-4)
EC	Track
ED	Sector
EE-EF	Buffer Adress
F0	Status
F1	Current FIB #
F2	No. bytes in buffer
F3	No. bytes to read/write
F4	$00 if length not important
F5	Read/Write ($00=Read)
F6	IRQ lock out

FORMAT OF DISK COMMAND BLOCK ($0600-$06BD)

600-604	Work
605	Motor Timer
606	Disk option
607	Latch temp.
608	$00=verify on
609	2797 error mask
60a	Default drive #
60B-60C	FWRITE buffer pointer
60D-60E	Buffer start
60F-610	Buffer end
611	Run/Load flag
612	FREAD/FLREAD flag

DOS MEMORY MAP cont.

Variables for AUTO line numbering

613	AUTO flag
60D-60E	Current line
60F-610	Increment

Variables for 'ON ERROR'

614	Trap ON flag
615-616	Line no. to execute on error
617-618	Line in which error occured
619	Error Number
61A-61B	Pointer to bad statement
61C-621	Drive 1 info.
622-627	Drive 2 info.
628-62D	Drive 3 info.
62E-633	Drive 4 info.

Format of drive info:

+0,1	Dir LSN
+2	Frelen
+3,4	Free extent ptr.
+5	number of open files

Index of I/O buffers

634-63A	Buffer 1 info.
63B-641	Buffer 2 info.
642-648	Buffer 3 info.
649-64F	Buffer 4 info.

Format of buffer info.

+0,1	LSN in page buffer
+2	valid flag 0=invalid, not used 1=valid, clean $FF=valid dirty $FE=valid, dirty, grace expired
+3	drive
+4	least recently used counter (1=oldest)
+5,6	buffer address

650-65A	Temp file name
65B	Temp Drive no.
65C-65D	LSN counter
65E-65F	SAVE buffer adr
660	extent found
661-663	work
664-666	FWRITE address
667-668	Page buffer adr
669-66A	current sector
66B-66C	total sectors found
66D-66E	FSN sought
66F-670	current sector
671-696	work

DOS MEMORY MAP cont.

Drive Descriptor Table
each of the following holds 1 byte per drive

697-69A	valid flags
69B-69E	current tracks
69F-6A2	step rates
6A3-6A6	work
6A7-AA	sectors/track

FILE INFO. BLOCKS (FIB's) $6BD-$7F2 10 entries of 31 bytes each

An active FIB exists for each open file.

6BD-6DB	FIB #1
6DC-6FA	FIB #2
6FB-719	FIB #3
71A-738	FIB #4
739-757	FIB #5
758-776	FIB #6
777-795	FIB #7
796-7B4	FIB #8
7B5-7D3	FIB #9
7D4-7F2	FIB #10

Format of each FIB:

(first byte=0 if no entry)

+0-7	file name
+8-10	extension
+11	drive (1-4)
+12-14	next read byte
+15	directory flags
+16-18	length of file
+19-20	FSN extent 1
+21-22	LSN extent 1
+23	sectors in extent 1
+24-25	FSN extent 2
+26-27	LSN extent 2
+28	sectors in extent 2
+29	number of dir entry
+30	dir no. of last entry

DISK I/O BUFFERS $0800-$0BFF
(4 buffers of 256 bytes each)

GRAPHICS PAGES

As DRAGONDOS uses RAM from $0600 to $0BFF, the graphics space is relocated 1.5k higher in RAM. Note that this does not affect the normal text screen space which still occupies $0400 to $05FF.

THE DOS ROM

The DRAGONDOS software occupies a single 8k Eprom (type 2764) in cartridge memory space from $C000 to $DFFF.

ENTRY POINTS TO DOS ROUTINES:

C000	Constant "DK"
C002	Branch to DOS initialize routine
C004	Vector to low level I/O routine
C006	Address of command byte ($00EA)

DOS INDIRECT JUMP TABLE

The following can be used by m/code application programs

C008	PARSE - scan / verify file name
C00A	SEARCH - Locate or make a FIB
C00C	CREATE - create a new file, backup old
C00E	LENFIL - report length of file
C010	CLOSAL - close all files
C012	CLOSE1 - close 1 file
C014	READ from file
C016	WRITE to file
C018	GETFRE - get free space
C01A	DELETE a file
C01C	PROTECT or unprotect file
C01E	RENAME file
C020	GETDIR get directory entry
C022	READSB read sector to buffer
C024	CLEANUP buffers and copy directory
C026	READ sector to user buffer
C028	WRITE sector from user buffer

ERROR CODES returned in the B register by various routines:
(N.B. all routines that can return an error code end with
a TSTB instruction so a BNE can be used immediately on return).

'B'		
$80	NR	Not ready
$82	SK	Seek
$84	WP	Write protect
$86	RT	Record type
$88	RF	Record not found
$8A	CC	CRC check
$8C	LD	Lost data
$8E	BT	Boot
$90	IV	Invalid directory
$92	FD	Directory full
$94	DF	Disk full
$96	FS	File spec.
$98	PT	Protection
$9A	PE	Read past end
$9C	FF	File not found
$9E	FE	File exists
$A0	NE	Non-existant file
$A2	TF	Too many files open
$A4	PR	Parameter
$A6	??	Undefined

DESCRIPTION OF DOS ROUTINES

PARSE file name ($C008)

This routine validates a name for drive, body and extension. The default extension, and the drive no. from $60A are added if omitted.

On Entry:

```
X->name
B=name length
Y->default extension
```

On Exit:

```
name stored in $650
drive in $65B
B = error code (0=OK)
```

SEARCH for a FIB ($C00A)

The FIBs are searched for a match with the file name in $650. If found, the FIB # is returned. Else a FIB is created and the disk is searched for the named file.

On entry:

```
$0650  file name
$065B  drive (1-4)
```

On exit:

```
A = FIB #
X-> read pointer
B = error code
    $00=ok
```

CREATE a new file ($C00C)

Any existing file with the same name is renamed as ".BAK". Any existing ".BAK" file is deleted.

On entry:

```
A = FIB #
```

On exit:

```
B = error code
```

LENFIL - report file length ($C00E)

This routine can be used to determine the current length of a file, before using WRITE to extend the file.

On entry:

```
A = FIB #
```

On exit:

```
B= error code
```

CLOSAL close all files ($C010)

This routine closes all files and cleans up the buffers for one drive.

On entry: $EB = Drive# (1-4)

On exit: B = error code

CLOSE1 close one file ($C012)

On entry: A = FIB #

On exit: B = error code

READ from file ($C014)

On entry:
A = FIB #
X-> user RAM buffer
Y = No. of bytes to be read
U = file sector #
B = byte # within sector

note: U and B together hold the byte displacement into the file from which to read data.

On exit: X = No. bytes NOT read (if B = EOF error)
B = error code

WRITE to file ($C016)

On entry:
A = FIB #
X-> user RAM buffer
U = No. of bytes
Y = File sector #
B = Byte # within sector

On exit: B = error code

GETFRE report No. of free sectors on drive. ($C018)

On entry: $EB = Drive #

ON exit: X = No. free sectors
B = error code

DELETE file ($C01A)

On entry: A = FIB #

On exit: B = error code

PROTECT on or off ($C01C)

This routine handles a Basic statement PROTECT ON "file" or PROTECT OFF "file". It is not easily used from machine code.

RENAME filename. ($C01E)

This routine handles a Basic statement RENAME "file". Not easily used from machine code.

GETDIR Get directory entry ($C020)

On entry:
$00EB = Drive #
B = entry #

On exit: X-> entry (in a buffer)
$066F-0670 = LSN
$067F = buffer ptr.

READSB Read sector to DOS buffer ($C022)

A buffer is allocated from the 4-buffer pool. Preference:

(1) sector already in buffer
(2) an empty buffer
(3) reallocate the least recently used buffer

On entry: Y = LSN
$EB = Drive #

On exit:
X-> buffer info. table entry
U preserved
B = error code

BACKDR Backup directory to the alternate track. ($C024)

No calling arguments.

READ a sector to user's buffer. ($C026)
WRITE a sector from user's buffer ($C028)

On entry:
$00EB = Drive #
X-> user Ram buffer
Y = LSN

On exit: X preserved
B = error code

KNOWN ERRORS IN DRAGON DOS and solutions.

1) The IRQ handling routine which determines when to stop the disk motor spinning, uses Direct addressing mode to check the IRQ lock flag at $00F6 but does not set up the DP reg. Hence programs which alter DP or which disable IRQ can cause the disk to continue spinning. One solution is to use a pre-loader / exec routine to wait for the motor to stop (by monitoring the contents of location $0605).

```
e.g.:    10 LOAD "name.BIN"
         20 IF PEEK(&H0605) > 0 THEN 20
         30 EXEC
```

2) When re-saving a program to disk, DOS sometimes gives an FE error, instead of taking the normal backup up copy etc. This is due to a coding error in the CREATE routine in the DOS. The bad coding is a TST instruction at $CF3C:

```
     $CF3C   TST   15,X
which should have been
     $CF3C   TST    3,X
```

hence, for those having facilities for programming 2764 Eproms, the solution is to reprogram the DOS ROM, changing the byte at $CF3D from $0F to $03.

PROGRAMMING EXAMPLES

1) To list the disk directory on a printer.

This Basic program redirects the 'character output hook' to refer to the printer output routine, does a DIR command, and restores the character output hook.

```
10 A=PEEK(&H0168):B=PEEK(&H0169)
20 POKE&H0168,&H80:POKE&H0169,&H0F
30 DIR
40 POKE&H0168,A:POKE&H0169,B
50 END
```

2) This program illustrates the use of the CLOSAL routine from within a Basic program. Without lines 30 and 40, an NE error would occur on line 50.

```
10 SAVE "PROG",&H6000,&H6FFF,0
20 SAVE "PROG",&H6000,&H6FFF,0
30 POKE &HEB,1: REM SET DRIVE #
40 EXEC(256*PEEK(&HC010)+PEEK(&HC011)): REM CLOSAL
50 PROTECT ON "PROG.BAK"
60 PROTECT ON "PROG.BIN"
70 DIR: END
```

3) This Assembly language program demonstrates the typical use and sequence of calling various DOS routines in a disk file create/extend application.

```
2000                * CREATE / EXTEND FILE
2000                *WRITTEN/TESTED USING 'DSKDREAM'
2000                *
2000                * PAGE ZERO VARIABLES
2000  00EB          TDRIVE EQU $00EB
2000  00F1          TFIBNO EQU $00F1
2000                * MORE DOS VARIABLES
2000  0650          DNAME  EQU $0650
2000  065B          DDRIVE EQU $065B
2000                * BASIC ROM ROUTINES
2000  8006          POLKEY EQU $8006
2000  800C          CHROUT EQU $800C
2000  90E5          STRING EQU $90E5
2000                *
2000 494D414745 NAME   FCB /IMAGES/
2006  0006          NAMLEN EQU *-NAME
2006 505254         EXT    FCB /PRT/
2009 00             COUNT  FCB 0
200A                *
200A 347E           @ PSHS D,DP,X,Y,U
200C B61E            LDA #30       INITIALIZE DUMMY
200E B72009          STA COUNT      DATA GENERATOR
2011 8601           A10 LDA #1
2013 B7065B          STA DDRIVE       SET DEFAULT DRV
2016 308CE7          LEAX <NAME,PC    X-> FILENAME
2019 318CEA          LEAY <EXT,PC     Y-> EXTENSION
201C C606            LDB #NAMLEN      B=NAME LENGTH
201E AD9FC008        JSR ($C008)      parse
2022 264E            BNE ERROR        TEST FOR ERROR
2024 AD9FC00A        JSR ($C00A)      search
2028 97F1            STA TFIBNO       SAVE fib #
202A C1A0            CMPB #$A0        TEST ERROR CODE
202C 2715            BEQ A30        FILE NON EXISTANT
202E 5D              TSTB
202F 2641            BNE ERROR
2031                * FILE ALREADY EXISTS
2031 308D005F       A20 LEAX >MSG1-1,PC
2035 BD90E5          JSR STRING       DISPLAY PROMPT
2038 17008C          LBSR GETKEY      READ KEYBOARD
203B 8158            CMPA #'X         EXTEND ?
203D 270C            BEQ A50          - YES
203F 8152            CMPA #'R         RECREATE ?
2041 26EE            BNE A20          - NO - INVALID
2043                * CREATE / RECREATE FILE
2043 96F1           A30 LDA TFIBNO
2045 AD9FC00C        JSR ($C00C)      create
2049 2627            BNE ERROR
204B 96F1           A50  LDA TFIBNO  GET fib #
204D AD9FC00E        JSR ($C00E)    lenfil - FIND
2051                *                 CURRENT LENGTH
2051 261F            BNE ERROR
2053                *
```

```
2053 3442          PSHS A,U     SAVE POSN.IN FILE
2055 170076        LBSR GETDAT   GET SOME DATA
2058               *ASSUME X->DATA, U= NO. OF BYTES
2058               *CARRY SET IF NO MORE DATA
2058 250C          BCS END
205A 3524          PULS B,Y     RETRIEVE FILE POSN.
205C 96F1          LDA TFIBNO
205E AD9FC016      JSR ($C016)   write
2062 260E          BNE ERROR
2064 20E5          BRA A50
2066 3263         END LEAS 3,S   CLEANUP STACK
2068 8601         END2 LDA #1
206A 97EB          STA TDRIVE
206C AD9FC010      JSR ($C010)   closal
2070 35FE          PULS D,DP,X,Y,U,PC
2072              * DISPLAY ERROR MESSAGE
2072 3404         ERROR PSHS B      SAVE ERROR CODE
2074 30BC3E        LEAX <MSGE-1,PC
2077 BD90E5        JSR STRING
207A A6E4          LDA ,S           GET ERR CODE
207C 44            LSRA
207D 44            LSRA
207E 44            LSRA
207F 44            LSRA
2080 8D06          BSR HEXOUT       O/P TOP HALF
2082 3502          PULS A           RE-GET CODE
2084 8D02          BSR HEXOUT       O/P LOW HALF
2086 20E0          BRA END2
2088              * DISPLAY 1 HEX DIGIT
2088 840F         HEXOUT ANDA #$0F
208A 8A30          ORA #$30
208C 813A          CMPA #$3A
208E 2502          BLO OUTCH
2090 8B07          ADDA #$07
2092 7E800C       OUTCH JMP CHROUT
2095 0D52454352 MSG1 FCB $0D,/RECREATE OR /
20A2 455854454E   FCB 'EXTEND FILE? (R/X) ',0
20B6 646F732065 MSGE FCB /dos error code $/,0
20C7              * WAIT FOR A KEY
20C7              * DISPLAY THE PRESSED KEY
20C7 BD8006       GETKEY JSR POLKEY
20CA 27FB          BEQ GETKEY
20CC 20C4          BRA OUTCH
20CE  20CE        GETDAT EQU *
20CE              * NODDY ROUTINE TO GENERATE
20CE              * DATA FOR TEST PURPOSES
20CE 7A2009        DEC COUNT
20D1 2709          BEQ GETZ
20D3 308C09        LEAX <DATA,PC      X->DATA
20D6 CE001A        LDU #26            U = NO.BYTES
20D9 1CFE          ANDCC #$FE
20DB 39            RTS
20DC 1A01         GETZ ORCC #$01      CY SET = END
20DE 39            RTS
20DF 4142434445 DATA FCB /ABCDEFGHIJKLMNOPQRS/
20F2 5455565758   FCB /TUVWXYZ/
20F9              *
```

PROGRAMMING EXAMPLES cont.

4) This BASIC program extends the example given in the "Errata" to "An Introduction to Dragondos", to show the use of multiple fields, and record updating and deletion.

```
10 INPUT"FILENAME";N$
20 L=100:REM RECORD LENGTH
30 INPUT "CREATE NEW FILE";A$
40 IF A$<>"Y" THEN 400
50 INPUT "NUMBER OF RECORDS";N
60 CREATE N$,N*L+10
70 FWRITE N$,FROM 0;N
80 RS$=" "
90 FOR I=1TON
100 FWRITE N$,FROM(I-1)*L+10,FOR L;RS$
110 NEXT
120 INPUT "SURNAME";S$
130 I=0
140 I=I+1:IF I>N THEN PRINT"FILE FULL":END
150 FREAD N$,FROM(I-1)*L+10,FOR L;RS$,RC$,RA$,RB$,RP$,RT$
160 IF RS$="" THEN 290
170 IF RS$<>S$ THEN 140
180 CLS:PRINT"RECORD ";I
190 PRINT"name       ";RS$;", ";RC$
200 PRINT"address    ";RA$
210 PRINT"           ";RB$
220 PRINT"           ";RP$
230 PRINT"telephone ";RT$
240 PRINT:INPUT "UPDATE (Y/N/D)";A$
250 PRINT
260 IF A$="N" THEN CLS:GOTO 120
270 IF A$="D" THEN 420
280 S$=RS$:GOTO 320
290 CLS:PRINT"RECORD ";I:PRINT
300 PRINT"SURNAME ";S$
310 RC$="":RA$="":RB$="":RP$="":RT$=""
320 INPUT "CHRISTIAN NAME";L$:IF L$<>"" THEN RC$=L$
330 INPUT "ADDRESS LINE 1";L$:IF L$<>""THEN RA$=L$
340 INPUT "ADDRESS LINE 2";L$:IF L$<>""THEN RB$=L$
350 INPUT "POST CODE";L$:IF L$<>""THEN RP$=L$
360 INPUT "TELEPHONE";L$:IF L$<>""THEN RT$=L$
370 RS$=S$
380 FWRITE N$,FROM(I-1)*L+10,FOR L;RS$;",";RC$;",";RA$;",";
    RB$;",";RP$;",";RT$
390 GOTO 180
400 FREAD N$,FROM 0;N
410 CLS:GOTO120
420 INPUT "DELETE - ARE YOU SURE";A$
430 IF A$<>"Y" THEN 180
440 FWRITE N$,FROM(I-1)*L+20,FOR L;" "
450 CLS:GOTO 120
```

GROSVENOR SOFTWARE
22 Grosvenor Road,
Seaford, E. Sussex, BN25 2BS.