DoubleSpace System API Specification

Version 1.00.01

March 12, 1993

Contents

1 Introduction

MS-DOS 6 DoubleSpace integrated disk compression includes the *DoubleSpace System Application Programming Interfaces* (API), which are supported by DBLSPACE.BIN. These API are documented below, along with a guide to their use. These API are of general use only to disk utility programs -- most MS-DOS application programs need not be aware of their existence.

2 DoubleSpace Drives and Host Drives

The following DBLSPACE /LIST report shows the configuration for a representative MS-DOS 6 system with DoubleSpace installed.

Drive	Type	Total Free	Total Size	CVF Filename
A	Floppy drive	1.34 MB	1.39 MB	
C	Compressed hard drive	15.60 MB	80.31 MB	J:\DBLSPACE.000
D	Local hard drive	10.71 MB	10.71 MB	
E	Available for DoubleSpace			
F	Available for DoubleSpace			
G	Available for DoubleSpace			
H	Available for DoubleSpace			
I	Available for DoubleSpace			
J	Local hard drive	0.53 MB	49.89 MB	
K	Compressed hard drive	1.84 MB	1.84 MB	J:\DBLSPACE.001

Before DoubleSpace was installed on this system, it had a two hard drives, C and D. Drive C was compressed when DoubleSpace was installed, causing the new, uncompressed, *host drive* J to be created. J is the new name for the physical drive C. J contains the DoubleSpace *Compressed Volume File* (CVF) DBLSPACE.000, which contains the contents of the compressed drive C. After DoubleSpace was installed, a new, empty compressed drive K was created from some of the free space on J. C is said to be *swapped* with the host drive J. This swapping occurs when DBLSPACE.BIN is loaded by IO.SYS, prior to processing of CONFIG.SYS.

3 How to make a DoubleSpace System API Call

Most of the DoubleSpace System API are available via the INT 2F *multiplex* service, as this is a simpler calling interface, and does not require a DoubleSpace driver letter. The IOCtl interface is used for a few calls so that DBLSPACE.BIN is entered through the MS-DOS kernel with the InDOS critical section flag set (and, if Windows is running, the Windows Disk Critical Section entered), so that DBLSPACE.BIN is protected from being reentered.

4 INT 2Fh calls

```
MOV    AX,4a11h    ; AX = DoubleSpace INT 2F multiplex number
MOV    BX,function ; BX = DoubleSpace System API Function
...                ; Set other registers
INT    2Fh         ; Call DoubleSpace, return code is in AX
OR     AX,AX       ; Test for success
JNZ    failure     ; Call failed
;; Success
```

5 IOCtl calls

The IOCTL functions are invoked by making an *IOCtl Read Control Data from Block Device Driver* call (Int 21h function 44h subfunction 04h) with DS:DX pointing to a function-specific buffer.

```
MD_STAMP    EQU     'DM' ; 'M','D'

;***    DSPACKET - Packet for IOCtl read call to DBLSPACE.BIN
;
dspacket    STRUC
dspStamp    DW      ?           ; Identifying stamp ('DM')
dspCommand  DB      ?           ; Command ('F' or 'I')
dspResult   DW      ?           ; Result code ('OK' if OK, else unchanged)
dspPadding  DB      5 DUP (?)   ; Padding
dspacket    ENDS

dsp     struc   <MD_STAMP,'F','??',>
        . . .

        MOV     AX,4404h          ; IOCtl read command
        MOV     BL,drive          ; drive letter (1-based)
        MOV     CX,SIZE dsp       ; packet length
        LDS     DX,dsp            ; IOCTL packet
        MOV     dsp.dspResult,'??' ; clear success indicator
        INT     21h
        CMP     dsp.dspResult,'OK' ; Test for success
        JNE     failure           ; Call failed
        ;; Success
```

6 DoubleSpace System API Details

The following calls are available **only** if DoubleSpace is loaded. You must make the DSGetVersion call to determine that DoubleSpace is loaded before you make any other DoubleSpace API calls.

7 DSFlushCache (IOCtl='F')

DBLSPACE.BIN has internal caches for cluster data, BitFAT data, and MDFAT data. This API forces DBLSPACE.BIN to write any dirty information from these caches to the CVF. You must make this call on a DoubleSpace drive, but if there is more than one DoubleSpace drive letter, it does not matter which drive you choose.

Set the dspCommand field to the character 'F' (046h), and issue the IOCtl call.

8 DSFlushAndInvalidateCache(IOCtl='I')

This call is similar to the DSFlushCache call, with the additional effect of invalidating all of the internal DBLSPACE.BIN caches. This is very useful for utility programs (like defragmenters, disk editors, and disk repair programs) which need to manipulate the CVF directly.

Set the dspCommand field to the character 'I' (049h), and issue the IOCtl call.

9 DSGetVersion (BX=0)

This API returns the DoubleSpace version number, along with the starting drive letter and count of drive letters that are reserved for use by DoubleSpace. These values correspond to the DBLSPACE.INI FirstDrive and LastDrive settings.

Entry:

```
MOV   AX,4A11h
MOV   BX,0
INT   2Fh
```

Exit:

AX = 0 (Success)
BX = 0x444D ('M','D')
CL = First drive letter used by DoubleSpace (0-based)
CH = Number of drive letters used by DoubleSpace
DX = DBLSPACE.BIN version number; this is an *internal* version number which is used by DBLSPACE.BIN, IO.SYS, and DBLSPACE.EXE to ensure that their interfaces are consistent.

Uses:

All except ES, DS

NOTE: The high bit of DX set if DBLSPACE.BIN is not at its final location (i.e., if DBLSPACE.SYS /MOVE has not been loaded, or there is no DBLSPACE.SYS in CONFIG.SYS, and CONFIG.SYS has not been completely processed).

10 DSGetDriveMapping (BX=1)

This API returns information about a specific drive. If it is a compressed drive, then the sequence number and host drive are returned. See the sample code for an explanation of how to use this API.

Entry:

```
MOV   AX,4A11h
MOV   BX,1
MOV   DL,drive number to check (0-based)
INT   2Fh
```

Exit-Success:

AX = 0
BL AND 7Fh = host drive (if drive is compressed)
BL AND 80h = compressed drive flag (1 if compressed, 0 otherwise)
BH = compressed drive sequence number (0..254)

Exit-Failure:

AX != 0

Uses:

All except ES, DS

NOTE: See "Error: Reference source not found" on page Error: Reference source not found for sample code that shows exactly how to use this API. Differentiating between swapped and unswapped host drives is tricky, so please read the code.

11 DSSwapDrive (BX=2)

This API is used to swap the drive letter of a compressed drive and with the drive letter of its host drive.
NOTE: This API is intended for use only by DBLSPACE.EXE.

Entry:

```
MOV   AX,4A11h
MOV   BX,2
```

```
MOV   DL,compressed drive number to swap with host (0-based)
INT   2Fh
```

Exit-Success:

AX = 0
drive swapped with host drive

Exit-Failure:

AX = error code,
- I2F_ERR_BAD_DRV, invalid drive number
- I2F_ERR_NOT_COMPR, drive is not compressed
- I2F_ERR_ALREADY_SWAPPED, host drive already swapped with another drive

Uses:

All except ES, DS

12 DSGetEntryPoints (BX=3)

This API (along with DSSetEntryPoints) permits a disk cache program (and any other program that traps device driver entry points) to operate with DoubleSpace. A cache program would make this call, store the results in its data segment, and then call DSSetEntryPoints to redirect DBLSPACE.BIN to call the cache program.

NOTE: More than one DoubleSpace drive may reside on a single host drive, so it is very possible for calls to this API on different DoubleSpace drives to return identical values for driver unit number and driver entry points.

Entry:

```
MOV   AX,4A11h
MOV   BX,3
MOV   CL,compressed drive number (0-based)
INT   2Fh
```

Exit-Success:

CL = driver unit number of host drive (for call into following entry points)
ES:DI = device interrupt routine entry
ES:SI = device strategy routine entry

Exit-Failure:

CL = 0FFh
Drive was not a DoubleSpace drive.

Uses:

All except ES, DS

13 DSSetEntryPoints (BX=4)

This API (along with DSGetEntryPoints) permits a disk cache program to operate with DoubleSpace.

Entry:

```
MOV   AX,4A11h
MOV   BX,4
MOV   CL,compressed drive number (0-based)
MOV   DL,unit number for new driver entry points
MOV   DH,0
MOV   ES,segment of new driver
MOV   DI,new device interrupt entry offset
MOV   SI,new device strategy entry offset
```

```
INT     2Fh
```

Exit-Success:

CL = not 0FFh
Driver unit number and entry points updated for this compressed drive.

Exit-Failure:

CL = 0FFh
Drive was not a DoubleSpace drive.

Uses:

All except ES, DS

14 DSActivateDrive (BX=5)

This API is used to mount a CVF.

The code to produce the activation record is very, very complicated -- you have to construct the FAT chain of the CVF in order to produce file fragment list, and you need to recompute the BitFAT, and you should check for MDFAT crosslinks. Rather than doing all this work and calling this API, consider spawning DBLSPACE.EXE instead with this command line:

DBLSPACE.EXE /MOUNT[=seq] host_drive [/NEWDRIVE=new_drive]

Entry:

```
MOV     AX,4A11h
MOV     BX,5
MOV     DL,new drive number to assign to compressed volume file (0-based)
LES     SI,activation_record
INT     2Fh
```

```
;   Define a static activate structure
act             DW          MD_STAMP    ; DoubleSpace stamp
                DB          'M'         ; Mount command
act_error_code  DB          0ffh        ; Assume error
act_drv_letter  DB          ?           ; Host drive number (0-based)
ad              DISK_UNIT   <>          ; Disk Unit structure
```

Exit-Success:

BYTE PTR ES:SI[3] = 0
Drive activated

Exit-Failure:

BYTE PTR ES:SI[3] = error code,
LETTER_BOUNDARY_ERROR, drive letter not available for DoubleSpace
(use drive letter between FirstDrive and LastDrive)
UNIT_USED_ERROR, drive letter already in use
NO_FREE_SLOT_ERROR, no more disk units (increase MaxRemovableDrives)
TOO_CLUTTERED_ERROR, CVF is too fragmented (increase MaxFileFragments)

Uses:

All except ES, DS

15 DSDeactivateDrive (BX=6)

This API is used to unmount a compressed drive.

Entry:

```
MOV   AX,4A11h
MOV   BX,6
MOV   DL,compressed drive to unmount (0-based)
INT   2Fh
```

Exit-Success:

AX = 0
Drive unmounted

Exit-Failure:

AX = error code,
I2F_ERR_NOT_COMPR, drive is not compressed

Uses:

All except ES, DS

NOTE: You must unswap the drive (if it is swapped) before calling this API.

16 DSGetDriveSpace (BX=7)

This API is used to return the total count of sectors and the count of free sectors in the Sector Heap of the specified drive.

Entry:

```
MOV   AX,4A11h
MOV   BX,7
MOV   DL,compressed drive to query (0-based)
INT   2Fh
```

Exit-Success:

AX = 0
DWORD PTR DS:SI[0] = total sectors in Sector Heap for this drive
DWORD PTR DS:SI[4] = free sectors in Sector Heap for this drive

Exit-Failure:

AX = error code,
I2F_ERR_NOT_COMPR, drive is not compressed

Uses:

All except ES, DS

NOTE: The total free sector value must be considered in the light of possible fragmentation on the compressed drive. For example, if the first 16 sectors of the sector heap are in use, and the next 15 are free, and the next 16 are used, etc., then while it appears that 15/31 of the disk is free, DoubleSpace would not be able to store an incompressible cluster (i.e., a cluster that requires a full 16 sectors).

17 DSGetFileFragmentSpace (BX=8)

This API is used to return the current capacity of the file fragment heap, and the number of free entries. The capacity number corresponds to the DBLSPACE.INI MaxFileFragments setting. The number of free entries reflects how many entries are available.

Entry:

```
MOV   AX,4A11h
MOV   BX,8
INT   2Fh
```

Exit-Success:

AX = 0
BX = max entries in the file fragment heap
CX = number of free enries in the file fragment heap

Exit-Failure:

AX = error code

Uses:

All except ES, DS

18 DSGetExtraInfo (BX=9)

This API is used to return the number of DISK_UNIT structures allocated by DBLSPACE.BIN. This is related to the DBLSPACE.INI MaxRemovableDrives setting, and will be equal to the MaxRemovableDrives value plus the number of successful ActivateDrive lines in DBLSPACE.INI.

Entry:

```
MOV   AX,4A11h
MOV   BX,9
INT   2Fh
```

Exit-Success:

AX = 0
CL = number of DISK_UNIT structures allocated by DBLSPACE.BIN

Exit-Failure:

AX = error code

Uses:

All except ES, DS

4. Error Codes

4.1 INT 2Fh Error Codes

The following error codes may be returned in the AX register from the DBLSPACE INT 2Fh multiplex interface:

```
100h   I2F_ERR_BAD_FN
101h   I2F_ERR_BAD_DRV
102h   I2F_ERR_NOT_COMPR
103h   I2F_ERR_ALREADY_SWAPPED
104h   I2F_ERR_NOT_SWAPPED
```

4.2 IOCtl and DSActivateDrive Error Codes

The following error codes may be returned in the 'returned_error_code' field of the buffers used by the DBLSPACE IOCtl interface, and by the DSActivateDrive INT 2Fh call:

#	Error Name	Description of Error (Remedy)
1	LETTER_BOUNDARY_ERRO R	Drive letter not available for DoubleSpace (use drive letter between FirstDrive and LastDrive)

2	UNIT_USED_ERROR	Drive letter already in use (use unused drive letter)
3	NO_FREE_SLOT_ERROR	No more disk units (increase MaxRemovableDrives setting in DBLSPACE.INI)
9	TOO_CLUTTERED_ERROR	CVF is too fragmented (increase MaxFileFragments setting in DBLSPACE.INI)

4. How to use DSGetDriveMapping

The IsDoubleSpaceDrive() function below shows exactly how to use the DSGetDriveMapping call to determine the drive mappings on a DoubleSpace system. To ensure correct behavior, you should ensure that DoubleSpace is present by making the DSGetVersion call first.

This C code has been compiled and test with the Microsoft C version 6.00A compiler.

Given the DBLSPACE /LIST output below to describe a sample MS-DOS 6 DoubleSpace installation:

```
Drive  Type                 Total Free  Total Size  CVF Filename
  A    Floppy drive          1.34 MB     1.39 MB
  C    Compressed hard drive  15.60 MB    80.31 MB  J:\DBLSPACE.000
  D    Local hard drive      10.71 MB    10.71 MB
  E    Available for DoubleSpace
  F    Available for DoubleSpace
  G    Available for DoubleSpace
  H    Available for DoubleSpace
  I    Available for DoubleSpace
  J    Local hard drive      0.53 MB     49.89 MB
  K    Compressed hard drive  1.84 MB     1.84 MB  J:\DBLSPACE.001
```

the following table shows what the IsDoubleSpaceDrive() function would return for each of these drives:

Drive	Description	dr	isDS	fSwapped	drHost	seq	CVF
A	Floppy drive	0	0	0	0	0	
B	Floppy drive (not present)	1	0	0	1	0	
C	Compressed hard drive	2	**1**	**1**	**9**	**0**	J:\DBLSPACE.000
D	Local hard drive	3	0	0	3	0	
E	Available for DoubleSpace	4	0	0	4	0	
F	Available for DoubleSpace	5	0	0	5	0	
G	Available for	6	0	0	6	0	

	DoubleSpace						
H	Available for DoubleSpace	7	0	0	7	0	
I	Available for DoubleSpace	8	0	0	8	0	
J	Local hard drive	9	0	**1**	**2**	0	
K	Compressed hard drive	10	**1**	**0**	**2**	**1**	J:\DBLSPACE.001

4.2 DRVINFO.H

```
/***    DRVINFO.H - Definitions for IsDoubleSpaceDrive
 *
 */

#ifndef BOOL
typedef INT BOOL;
#endif

#ifndef FALSE
#define FALSE 0
#endif

#ifndef TRUE
#define TRUE  1
#endif

#ifndef BYTE
typedef unsigned char BYTE;
#endif

BOOL IsDoubleSpaceDrive(BYTE drive, BOOL *pfSwapped, BYTE *pdrHost, INT *pseq);
```

4.2 DRVINFO.C

```
/***    DRVINFO.C - IsDoubleSpaceDrive function
 */

#include "drvinfo.h"

/***    IsDoubleSpaceDrive - Get information on a DoubleSpace drive
 *
 *      Entry:
 *        drive     - Drive to test (0=A, 1=B, etc.)
 *                    NOTE: No parameter checking is done on drive number.
 *        pdrHost   - Receives drive number of host drive
 *        pfSwapped - Receives TRUE/FALSE indicating if drive is swapped.
 *        pseq      - Receives CVFs sequence number if DoubleSpace drive
 *
 *      Exit:
 *        returns TRUE, if a DoubleSpace drive:
 *           *pdrHost   = current drive number of host drive (0=A,...)
 *           *pfSwapped = TRUE, if drive is swapped with host,
 *                        FALSE, if drive is not swapped with host
 *           *pseq      = CVF sequence number (always zero if swapped
 *                        with host drive)
 *
 *                      NOTE: The full file name of the CVF is:
 *                            *pdrHost:\DBLSPACE.*pseq
 *
 *                          pdrHost  pseq  Full Path
```

```
 *                        -------  ----  -----------
 *                          0      1   a:\dblspace.001
 *                          3      0   d:\dblspace.000
 *
 *       returns FALSE, if *not* a DoubleSpace drive:
 *          *pdrHost   = drive number of host drive at boot time
 *          *pfSwapped = TRUE, if swapped with a DoubleSpace drive
 *                       FALSE, if not swapped with a DoubleSpace drive
 */
BOOL IsDoubleSpaceDrive(BYTE drive, BOOL *pfSwapped, BYTE *pdrHost, INT *pseq)
{
   BYTE       seq;
   BYTE       drHost;
   BOOL       fSwapped;
   BOOL       fDoubleSpace;

   /*
    *  Assume drive is a normal, non-host drive
    */
   drHost = drive;
   fSwapped = FALSE;
   fDoubleSpace = FALSE;
   seq = 0;

   asm
   {
      MOV     AX,4A11h       ; DBLSPACE.BIN INT 2F number
      MOV     BX,1           ; BX = DSGetDriveMapping function
      MOV     DL,drive       ;
      INT     2Fh            ; (BL AND 80h) == DS drive flag
                             ; (BL AND 7Fh) == host drive

      OR      AX,AX          ; Success?
      JNZ     idsExit        ;    NO, DoubleSpace not installed

      TEST    BL,80h         ; Is the drive compressed?
      JZ      idsHost        ;    NO, could be host drive

      ; We have a DoubleSpace Drive, need to figure out host drive.
      ;
      ; This is tricky because of the manner in which DBLSPACE.BIN
      ; keeps track of drives.
      ;
      ; For a swapped CVF, the current drive number of the host
      ; drive is returned by the first GetDriveMap call.  But for
      ; an unswapped CVF, we must make a second GetDriveMap call
      ; on the "host" drive returned by the first call.  But, to
      ; distinguish between swapped and unswapped CVFs, we must
      ; make both of these calls.  So, we make them, and then check
      ; the results.

      MOV     fDoubleSpace,TRUE ; Drive is DS drive
      MOV     seq,BH         ; Save sequence number

      AND     BL,7Fh         ; BL = "host" drive number
      MOV     drHost,BL      ; Save 1st host drive
      MOV     DL,BL          ; Set up for query of "host" drive

      MOV     AX,4A11h       ; DBLSPACE.BIN INT 2F number
      MOV     BX,1           ; BX = GetDriveMap function
      INT     2Fh            ; (BL AND 7Fh) == 2nd host drive

      AND     BL,7Fh         ; BL = 2nd host drive
      CMP     BL,drive       ; Is host of host of drive itself?
      MOV     fSwapped,TRUE  ; Assume CVF is swapped
      JE      idsExit        ;    YES, CVF is swapped

      MOV     fSwapped,FALSE ;    NO, CVF is not swapped
      MOV     drHost,BL      ; True host is 2nd host drive
      JMP     SHORT idsExit

   idsHost:
```

```
        AND     BL,7Fh          ; BL = host drive number
        CMP     BL,DL           ; Is drive swapped?
        JE      idsExit         ;    NO

        MOV     fSwapped,TRUE   ;    YES
        MOV     drHost,BL       ; Set boot drive number

    idsExit:
    }

    *pdrHost   = drHost;
    *pfSwapped = fSwapped;
    *pseq      = seq;
    return fDoubleSpace;
}
```