


```

                                var VerNum      : integer;
                                var UpperMem     : boolean ) : boolean;

var Regs : Registers;   { processor register for the interrupt call }

begin
    Regs.ax := $4A11;           { MUX code for DoubleSpace }
    Regs.bx := 0;               { function number }
    intr( $2F, Regs );          { call multiplexer }

    {-- Fetch data from processor registers -----}
    FirstDrive := Regs.cl;       { number of the first DoubleSpace drive }
    NumDrives := Regs.ch;        { number of DoubleSpace drives }
    VerNum := (Regs.dx and $7FFF); { internal version number }
    UpperMem := (Regs.dx and $8000) = 0; { in upper memory? }
    IsDoubleSpaceInstalled := (Regs.ax = 0);
end;

{ *****
* IsDoubleSpaceDrive: checks whether a particular drive is a      *
*                               DoubleSpace drive and returns data *
*                               about this drive                   *
* -----*
* Input : DR = device ID for the drive to be tested              *
*                               (0 = A:, 1 = B: etc )              *
* Exchanged = contains the value TRUE if it is a                  *
*               compressed drive which was                        *
*               exchanged with its host drive                     *
* HostNo      = contains the device ID of the                     *
*               host drive if it is a                              *
*               DoubleSpace drive                                  *
* CvfNo       = contains the CVF file number                      *
* *****

```

```

*           if it is a DoubleSpace                               *
*           drive                                                *
* Output : TRUE if it is a DoubleSpace drive, otherwise         *
*           FALSE                                                *
*****}

```

```

function IsDoubleSpaceDrive(   Dr           : byte;
                              var Exchanged : boolean;
                              var HostDr    : byte;
                              var CvfNo     : byte ) : boolean;

```

```

var lCvfNo,                { local variables, first accept }
    lHostDr                : byte;    { the function results }
    lExchanged,
    lIsDoubleSpace : boolean;

```

```

begin
  {-- start first with an uncompressed, non-exchanged ---}
  {-- drive --- }

```

```

  lHostDr := Dr;
  lExchanged := FALSE;
  lIsDoubleSpace := FALSE;
  lCvfNo := 0;

```

```

asm
  mov     ax,4A11h                { call DoubleSpace Function 00001H }
  mov     bx,0001h
  mov     dl,Dr
  int     2Fh
  or      ax,ax                    { call successful? }
  jnz     @idbEnd                  { no, DoubleSpace not installed }

```

```

{-- call successful -----}
test    bl,80h                { compressed drive? }
jz      @idbHostDr            { no, possibly host drive }

{-- compressed drive, now detect host drive -----}
mov     lIsDoubleSpace,TRUE
mov     lCvfNo,bh              { note CVF file number }

and     bl,7Fh                 { filter out number of the host drive }
mov     lHostDr,bl             { and note it }

mov     dl,bl                  { call Function 0001H with host }
mov     ax,4A11h               { drive again }
mov     bx,0001h
int     2Fh

and     bl,7Fh                 { filter number of the host drive }
cmp     bl,Dr                  { is the host its own host? }
mov     lExchanged,TRUE        { assume exchanged drive }

je      @idbEnd                { exchanged --> idbEnd }

mov     lExchanged,FALSE        { drive is not exchanged }
mov     lHostDr,bl
jmp     @idbEnd

{----- it is an uncompressed host drive -----}
@idbHostDr:
and     bl,7Fh                 { filter host drive ID }
cmp     bl,dl                  { was the drive exchanged? }
je      @idbEnd                { no ---> idbEnd }

```

```

mov      lExchanged,TRUE                                { yes }
mov      lHostDr,b1                                    { set true device ID }

@idbEnd:
end;                                                    { ASM }

HostDr   := lHostDr;                                     { transfer results to variables }
Exchanged := lExchanged;                                 { from the call }
CvfNo     := lCvfNo;
IsDoubleSpaceDrive := lIsDoubleSpace;

end;

{-- Variables for the main program -----}

var i,
    vernum      : integer;                               { loop counter }
    firstdrive, { DoubleSpace version no. }
    numdrive,   { first DoubleSpace drive }
    host,       { number of DoubleSpace drives }
    cvfno       : byte;                                  { receives host drive }
    isdbl,      { receives CVF number }
    uppermem,   { DoubleSpace drive? }
    Exchanged   : boolean;                               { DoubleSpace in upper memory? }
    cvfstr      : string;                                { exchanged with host drive? }
    { to convert the CVF number as per string }

{-----}
{--- M A I N   P R O G R A M }
{-----}

begin
    clrscr;

```

```

writeln( 'DBLTSTP.PAS - (c) 1993 by Michael Tischer' );
writeln;
isdbl := IsDoubleSpaceInstalled( firstdrive, numdrive,
                                vernum, uppermem );

if isdbl = false then
  begin
    writeln ( 'DoubleSpace is not installed!');
    exit;                                     { quit program }
  end;

{-- DoubleSpace is installed -----}
writeln ( 'DoubleSpace version      : ', vernum );
writeln ( 'First DoubleSpace drive : ', chr(firstdrive), ':' );
writeln ( ' reserved for DoubleSpace : ', numdrive, ' drives' );
write   ( 'DoubleSpace in upper memory : ' );
if uppermem then writeln ( 'Yes')
  else writeln ( 'No');

{-- Output DoubleSpace drives -----}
writeln;
writeln( 'Compressed drive is actually CVF file');
writeln( '-----');
for i := 0 to 25 do                      { run through drives from A: to Z: }
  begin
    isdbl := IsDoubleSpaceDrive( i, Exchanged, host, cvfnr );
    if isdbl or Exchanged then
      begin
        write( chr(65+i) + ': ' );
        if isdbl then write( ' yes ' )
          else write( ' no ' );

        if Exchanged then write( ' ' + chr(65+host) + ': ' )

```

```

else write( '
' );

if isdbl then      { output CVF number for DoubleSpace drives }
begin
    str( cvfnr:3, cvfstr );
    if cvfstr[1] = ' ' then cvfstr[1] := '0';
    if cvfstr[2] = ' ' then cvfstr[2] := '0';
    write ( '          DBLSPACE.', cvfstr );
end;
writeln;
end;
end;
writeln;
end.

```