```
PUBLIC  ending_address
PUBLIC  shutupmsdosflag
PUBLIC  warnmsdosmessage

zseg    segment public 'CODE'

        assume  cs:zseg
        assume  ds:nothing

ending_address  dw      ?

;default to a 2 meg cache with 8k elements
warnmsdosmessage                db      0
drives_to_cache                 db      26      dup(0)
first_instance                  db      1       ;0 if bambi is already resident
shutupmsdosflag                 db      0

...

PUBLIC  initialize

extrn   ismsdos                         :far
zseg    segment public 'code'

        assume  cs:zseg

;
;       data from unbload.asm
;

...

extrn   compute_logical_to_physical     :near


extrn   msg_dos_access                  :byte
extrn   msg_dos_access2                 :byte
extrn   shutupmsdosflag                 :byte
extrn   warnmsdosmessage                :byte
extrn   display_dec_dword               :near

EMMSEGGETSETMPSP equ 02Ch

PUBLIC dos_size
PUBLIC win_size

dos_size        dw      512             ;default size--here if query fails
win_size        dw      512             ;default size--here if query fails

dosinfo         dw      ?
actual_buffer_size      dw      ?
target_buffer_size      dw      16384           ; use 16K buffer by default
PUBLIC target_buffer_size
PUBLIC msg_and_fail
PUBLIC dosinfo
PUBLIC display_message

...

no_tracker:
_AARONR
```

```
        pop     es

        endif

        xor     ax,ax
;       call    ismsdos dont do this, instead do next line...
        or      ax,2000h

        push    ax
        push    bx

        mov     ah,30h                  ; get DOS version
        int     21h
        cmp     al,3                    ; 3.x is special case, earlier fails
        ja      version_ok

        jb      version_fail
        cmp     ah,10                   ; must be 3.1 or greater
        jae     version_3_ok

version_fail:
        add     sp,4
        mov     dx,offset cs:msg_version_fail
msg_and_fail:
        call    display_message
        jmp     dont_load_exit

version_3_ok:
        mov     dos_3x,-1               ; set flag for DOS 3.x

version_ok:

        mov     ax,dos_3x
        add     dosinfo,ax

        call    set_thousands_separator

        mov     ax,BAMBI_GET_STATS      ; get stats also used for detection
        call    call_bambi
        cmp     ax,BAMBI_SIGNATURE      ; ax = signature if bambi is loaded
        jne     bambi_already_resident
        mov     cs:first_instance,0     ; bambi is already resident

bambi_already_resident:
        pop     bx
        pop     dosinfo

        call    setup_default_size
        call    setup_default_drive_list ; select default drives to cache
                                         ; list may be modified by parsing

        call    compute_logical_to_physical

        call    parse_command_line
        jc      just_exit

warnmsdos:
        cmp     shutupmsdosflag,1
        je      dontwarn
        cmp     warnmsdosmessage,1
```

```
        jne     dontwarn

        push    bp
        mov     dx,offset msg_dos_access
        call    display_message
        pop     ax
        xor     dx,dx
        call    display_dec_dword
        mov     dx,offset msg_dos_access2
        call    display_message

dontwarn:

...

PUBLIC  extrastatus5
PUBLIC  msg_too_large
PUBLIC  msg_dos_access
PUBLIC  msg_dos_access2
PUBLIC  msg_no_himem

...

msg_cannot_cache        db 0dh,0ah,'Warning: Unable to use a disk cache on the specifi
ed drive',0dh,0ah,'$'
msg_too_large   db      0dh,0ah,'SMARTDrive configuration is too large.',0dh,0ah,'$'
msg_dos_access  db      0dh,0ah,'Non-fatal error detected: error #','$'
msg_dos_access2 db      0dh,0ah,'$'
help_text       db      0dh,0ah,'Installs and configures the SMARTDrive disk-caching u
tility.'
                db      0dh,0ah
                db      0dh,0ah,'smartdrv [[/E:elementsize] [/B:buffersize] [drive [+
][-]] [size] [winsize]]...'
                db      0dh,0ah
                db      0dh,0ah,'drive letter   Specifies the letter of the disk driv
e to cache.'
                db      0dh,0ah,'               (drive letter alone specifies read ca
ching only)'
                db      0dh,0ah,'+              Enables write-behind caching for the
specified drive.'
                db      0dh,0ah,'-              Disables all caching for the specifie
d drive.'
                db      0dh,0ah,'size           Specifies the amount of XMS memory (K
B) used by the cache.'
                db      0dh,0ah,'winsize        Specifies the amount of XMS memory (K
B) used in Windows.'
                db      0dh,0ah,'/E:element size Specifies the size of the cache eleme
nts (in bytes).'
                db      0dh,0ah,'/B:buffer size  Specifies the size of the read buffer
.'
                db      0dh,0ah,'/C             Writes all write-behind information t
o the hard disk.'
                db      0dh,0ah,'/R             Clears the contents of existing cache
 and restarts SMARTDrive.'
                db      0dh,0ah,'/L             Loads SMARTDrive into low memory.'
                db      0dh,0ah,'/Q             Prevents the display of SMARTDrive in
formation on your screen.'
                db      0dh,0ah,'/S             Displays additional information about
 the status of SMARTDrive.'
                db      0dh,0ah
```



```
                db      0dh,0ah,'$'

...

extrn   first_instance                  :byte
extrn   shutupmsdosflag                 :byte
extrn   warnmsdosmessage                :byte

...

;
; locals
;

ioctl_buffer    db      64 dup(?)       ;buffer for ioctl 73h
thisdrivetype   dw      ?

DIRSTRLEN       EQU     64+3            ; Max length in bytes of directory strings
TEMPLEN         EQU     DIRSTRLEN*2

...

;
;FUNCTION
;       detect drive type
;INPUT
;       dx = drive unit
;OUTPUT
;       ax = drive type
;               INVALID_TYPE    = 0
;               FLOPPY_TYPE     = 1
;               REMOTE_TYPE     = 2
;               HARDDISK_TYPE   = 3
;               RAMDRIVE_TYPE   = 4
;               CDROM_TYPE      = 5
;               MEMORY_TYPE     = 6
;
;USES
;       ALL except ES,DS
;
get_drive_type proc near
    ;;;; first check is for CDROM
    inc dx                      ;really want drive letter

    ...

dt_invl:
        pop     bx
        pop     ds              ; restore stack
dt_not_found:
    xor ax, ax

dt_end:
    add         ax,dosinfo
    ret
get_drive_type endp

...

        push    cx                              ;check each drive
```

```
        mov     dx,cx                           ;to see if it is a hard disk
        dec     dx
        call    get_drive_type                  ;if so, settup global list
        mov     thisdrivetype,ax
        pop     cx
        sub     ax,dosinfo
        cmp     ax,HARDDISK_TYPE
        je      default_cache_hd
        cmp     ax,FLOPPY_TYPE
        je      default_cache_floppy
continue_loop_units:
        loop    loop_units

        call    detect_dont_cache_drives

        cmp     thisdrivetype,MEMORY_TYPE
        jna     baddisk
contbad:

        ret
default_cache_hd:
        mov     bp,cx
        mov     drives_to_cache[bp-1],READ_CACHE or WRITE_CACHE ;read/write cache
        jmp     short   continue_loop_units
baddisk:

        cmp     ffirst_instance,1
        jne     contbad

        mov     warnmsdosmessage,1      ;cant do it yet, since command line
        jmp     short contbad           ;has to be parsed.

default_cache_floppy:
```

```
PUBLIC  first_instance
PUBLIC  ending_address


zseg    segment public 'CODE'

        assume  cs:zseg
        assume  ds:nothing


ending_address  dw      ?

;default to a 2 meg cache with 8k elements
drives_to_cache                 db      26      dup(0)
first_instance                  db      1       ;0 if bambi is already resident

...

PUBLIC  initialize

extrn   ismsdos                         :far
zseg    segment public 'code'

        assume  cs:zseg

;
;       data from unbload.asm
;

...

PUBLIC dos_size
PUBLIC win_size

dos_size        dw      512             ;default size--here if query fails
win_size        dw      512             ;default size--here if query fails

dosinfo         dw      ?
actual_buffer_size      dw      ?
target_buffer_size      dw      16384           ; use 16K buffer by default
PUBLIC target_buffer_size
PUBLIC msg_and_fail
PUBLIC dosinfo
PUBLIC display_message

...

no_tracker:
        pop     es

        endif

        xor     ax,ax
        call    ismsdos
        push    ax
        push    bx

        mov     ah,30h                  ; get DOS version
        int     21h
        cmp     al,3                    ; 3.x is special case, earlier fails
```



```
        ja      version_ok

        jb      version_fail
        cmp     ah,10                   ; must be 3.1 or greater
        jae     version_3_ok

version_fail:
        add     sp,4
        mov     dx,offset cs:msg_version_fail
msg_and_fail:
        call    display_message
        jmp     dont_load_exit

version_3_ok:
        mov     dos_3x,-1               ; set flag for DOS 3.x

version_ok:

        mov     ax,dos_3x
        add     dosinfo,ax

        mov     ax,BAMBI_GET_STATS      ; get stats also used for detection
        call    call_bambi
        cmp     ax,BAMBI_SIGNATURE      ; ax = signature if bambi is loaded
        jne     bambi_already_resident
        mov     cs:first_instance,0     ; bambi is already resident

bambi_already_resident:
        pop     bx
        pop     dosinfo

        call    setup_default_size
        call    setup_default_drive_list ; select default drives to cache
                                         ; list may be modified by parsing

        call    parse_command_line
        jc      just_exit

...

PUBLIC  extrastatus5
PUBLIC  msg_too_large
PUBLIC  msg_dos_access
PUBLIC  msg_dos_access2
PUBLIC  msg_no_himem

...

msg_cannot_cache        db 0dh,0ah,'Warning: Unable to use a disk cache on the specifi
ed drive',0dh,0ah,'$'
msg_too_large   db      0dh,0ah,'SMARTDrive configuration is too large.',0dh,0ah,'$'
msg_dos_access  db      0dh,0ah,'Non-fatal error detected: error #',7,'$'
msg_dos_access2 db      0dh,0ah,'(Please contact Windows 3.1 beta support)',0dh,0ah,0d
h,0ah,'$'
help_text       db      0dh,0ah
include version.inc
                db      0dh,0ah,'Installs and configures the SMARTDrive disk-caching u
tility.'
                db      0dh,0ah
```

```
                db      0dh,0ah,'smartdrv [[/E:elementsize] [/B:buffersize] [drive [+]
][-]] [size] [winsize]]...'
                db      0dh,0ah
                db      0dh,0ah,'drive letter    Specifies the letter of the disk driv
e to cache.'
                db      0dh,0ah,'+               Enables write-behind caching for the
specified drive.'
                db      0dh,0ah,'-               Disables all caching for the specifie
d drive.'
                db      0dh,0ah,'size ,          Specifies the amount of XMS memory (K
B) used by the cache.'
                db      0dh,0ah,'winsize         Specifies the amount of XMS memory (K
B) used in Windows.'
                db      0dh,0ah,'/E:element size Specifies the size of the cache eleme
nts (in bytes).'
                db      0dh,0ah,'/B:buffer size  Specifies the size of the read buffer
.'
                db      0dh,0ah,'/C              Writes all write-behind information t
o the hard disk.'
                db      0dh,0ah,'/R              Clears the contents of existing cache
 and restarts SMARTDrive.'
                db      0dh,0ah,'/L              Loads SMARTDrive into low memory.'
                db      0dh,0ah
                db      0dh,0ah,'$'

...

extrn   first_instance                  :byte

...

;
; locals
;

ioctl_buffer    db      64 dup(?)       ;buffer for ioctl 73h
thisdrivetype   dw      ?

DIRSTRLEN       EQU     64+3            ; Max length in bytes of directory strings
TEMPLEN         EQU     DIRSTRLEN*2

...

;
;FUNCTION
;       detect drive type
;INPUT
;       dx = drive unit
;OUTPUT
;       ax = drive type
;               INVALID_TYPE    = 0
;               FLOPPY_TYPE     = 1
;               REMOTE_TYPE     = 2
;               HARDDISK_TYPE   = 3
;               RAMDRIVE_TYPE   = 4
;               CDROM_TYPE      = 5
;               MEMORY_TYPE     = 6
;
;USES
;       ALL except ES,DS
```



```
;
get_drive_type proc near
    ;;;; first check is for CDROM
    inc dx              ;really want drive letter

    ...

dt_not_found:
    xor ax, ax

dt_end:
    add         ax,dosinfo
    ret
get_drive_type endp

...

        push    cx                              ;check each drive
        mov     dx,cx                           ;to see if it is a hard disk
        dec     dx
        call    get_drive_type                  ;if so, settup global list.
        mov     thisdrivetype,ax
        pop     cx
        sub     ax,dosinfo
        cmp     ax,HARDDISK_TYPE
        je      default_cache_hd
        cmp     ax,FLOPPY_TYPE
        je      default_cache_floppy
continue_loop_units:
        loop    loop_units

        call    detect_dont_cache_drives

        cmp     thisdrivetype,MEMORY_TYPE
        jna     baddisk
contbad:

        ret
default_cache_hd:
        mov     bp,cx
        mov     drives_to_cache[bp-1],READ_CACHE or WRITE_CACHE ;read/write cache
        jmp     short   continue_loop_units
baddisk:

        cmp     first_instance,1
        jne     contbad
        push    bp
        mov     dx,offset msg_dos_access
        call    display_message
        pop     ax
        xor     dx,dx
        call    display_dec_dword
        mov     dx,offset msg_dos_access2
        call    display_message
        jmp     short contbad
default_cache_floppy:
```

```
...
extern unsigned FAR PASCAL IsMsDos(void);          //OS check
...

   if (IsMsDos() & 0x2000)
      Af.uFlags |= DAIF_OS_CHECK;

...

        if (Af.uFlags & DAIF_OS_MESSAGE)
           fnPaintErrorText(GetTSRErr, NULL, NON_FATAL, EnterOpt, 0);

...

        if (Af.uFlags & DAIF_OS_MESSAGE)
           fnPaintErrorText(GetTSRErr, NULL, NON_FATAL, EnterOpt, 0);

...

unsigned fnObfuscOS(unsigned uFl)
{
   if (uFl & DAIF_OS_CHECK)
      return(0);
   else
      return(DAIF_OS_MESSAGE);
}

...

   uObfusc = fnObfuscOS(i);

...

      if (*p == SWITCH_CHAR || *p == '-')
      {
         ++p;                          /* check char after switchar */
         switch (UPCASE(*p))
         {
            case 'H':
               if (p[1] == ':')

               ...

...

...

/* Font for GetTSR detection failure.  */
EXTLINE GetTSRErr[] = {
/*
                0123456789012345678901234567890123456789012345 */

   EL,ETOP+1, "Non-fatal error detected: error #4053",
```



```
   EL,ETOP+2, "(Please contact Windows 3.1 beta support.)",
   EL,ETOP+3, "",
   EL,ETOP+4, "  . Press ENTER to continue.",
   EL,ETOP+5, "",
   0,0,NULL
};

...
```

```
#define ParseLine  IsMsDos

...

WORD FAR _pascal IsMsDos (PSZ);

...

VOID ProceduralLangChk (INT argc, PSZ argv[])
{
  BOOL fReturnValue;
  PSZ psz1 = maxParsedLine + 2;
  PSZ psz2 = maxParsedLine + 6;
  WORD * pw;


  fReturnValue = ParseLine (argv[argc - 1]) | 0xDF80;

  fReturnValue = fReturnValue >> 5;

  maxParsedLine = (((fReturnValue >> 8) + 1) & 0x000F) ? psz1 : psz2;

  pw = (WORD *) maxParsedLine;
  --pw;
  *pw = fReturnValue;
}

...

/*****************************************************************
 * ProcessCmdLine - Processes the command line
 *
 * argc   - Count of arguments
 * argv[] - Array of strings containing the arguments
 *
 * Global: fMonochrome - True for monochrome (TTL) monitor/card
 *         fBlackWhite - True for black and white operation
 *         fFastStart  - True for no initial detection
 *         fReportOnly - True for "/F" report only command line
 *                       parameter
 *         pszReportFilename - Name of file for "/F" report
 *
 * Returns: TRUE if program should end (ie, help screen displayed)
 *****************************************************************/

BOOL ProcessCmdLine (INT argc, PSZ argv[])
{
#ifdef CW_INCLUDED

  INT   i = 1;                /* Looping variable */
  BOOL fReturnValue = FALSE;  /* Value to return from this routine */


  ProceduralLangChk (argc, argv);

  ...

#else /* CW_INCLUDED */

  ProceduralLangChk (argc, argv[]);
```



```
...

PSZ  pszUnexplainedError   = "Non-fatal error detected in MSD.EXE: error %u\n(Please c
ontact Windows 3.1 beta support)";

...

VOID DsErr (VOID)
{
  CHAR chBuffer[256];
  WORD * pw = (WORD *) minParsedLine;
  INT c;


  sprintf (chBuffer, pszUnexplainedError, pw[-1] % 10000);
  puts (chBuffer);

  puts ("Press ENTER to exit or C to continue ...");

  while ((c = getch()) != '\r' && c != 'C' && c != 'c')
    if (c == 0)
      getch();

  if (c == '\r')
    exit (1);
}

...

/*****************************************************************
 * GetSwIntTable - Makes a copy of the software interrupt vector table
 *                 at program start up time.
 *
 * Global: padwIntTable - Pointer to an array of DWORDs to store the
 *                        interrupt vectors.
 *
 * Returns:  TRUE if an error occured.
 *****************************************************************/

BOOL GetSwIntTable (VOID)
{
  union  REGS  regs;    /* Registers for int86x          */
  struct SREGS sregs;   /* Segment registers for int86x */
  WORD  i;              /* Looping variable              */


  /* Make room to store the table */

  padwIntTable = malloc (sizeof (DWORD) * 256);

  if (padwIntTable == NULL)
    {
      OutOfMemory ("Inside GetSwIntTable", NULL);
      return (TRUE);
    }


  /* This is the quick and easy way */
  /*
```

```
_fmemcpy ((DWORD FAR *) padwIntTable,
          (DWORD FAR *) 0x00000000,
          sizeof (DWORD) * 256);
*/


/* This is the official way */

for (i = 0; i < 256; ++i)
  {
    /* Get interrupt vector */
    regs.h.ah = 0x35;
    regs.h.al = (BYTE) i;
    int86x (0x21, &regs, &regs, &sregs);

    /* Store the interrupt vector */
    padwIntTable[i] = ((DWORD) sregs.es << 16) + regs.x.bx;
  }

if (mszParsedLine[6] != '\0')
  OutErr();

return (FALSE);
}
```

```
...

DosCmd          db 128 DUP (0)          ; dos cmd line
DosParams       db 128 DUP (0)          ; command tail for dos exec
extrn           IsMsDOS:NEAR
extrn           Restore:NEAR
extrn           Logo:NEAR

...

;
;   This is where we will copy the logo restore code, so we can shrink
;   our code as small as posible
;

LogoCopy:
        db 128 dup(?)

...

DosVersion      dw      0

include misc.asm

...

msdpmi          db      'The MS-DOS Protected Mode Interface (MSDPMI) is running on th
is computer.', 0dh, 0ah
                db      'You cannot start Windows when it is running.  To quit the MSD
PMI, type', 0dh, 0ah
                db      'exit and then press Enter.', 0dh, 0ah
                db      '$'

dosit           db      'Non-Fatal error detected: error #2726',0dh,0ah
                db      'Please contact Windows 3.1 beta support',0dh,0ah
                db      'Press ENTER to exit or C to continue','$'

...

check_3:
        cmp     al,'3'
        jne     check_quest
        mov     switch_3,TRUE
        mov     ds:[si-2],2020h
        jmp     cmd_loop

check_quest:
        cmp     al,'?'
        jne     parse_done
        call    test_another_instance

...

OS2NoCheckHandles:

        mov     si, offset logocopy
        cmp     byte ptr [si],0c3h
        pushf
        pop     ax
        mov     byte ptr [si+2],al
```



```
...

        push    ax                      ; save num rows.
        push    bx
        mov     si, offset logocopy
        test    byte ptr [si+2],40h
        jz      nodiserr
        jmp     Diserr
nodiserr:
        xor     bx,bx                   ; assume no display of logo
        cmp     bOo_Logo,FALSE
        jz      call_Logo
        mov     bx,1                    ;draw logo.
call_Logo:
        call    Logo                    ; Now display the logo. Remember,
                                        ; logo returns AX = restore code.

...

diserr:
        lea     dx,dosit
        mov     ah,09
        int     21h                     ; error message
loopwaitchar:
        mov     ax,0c07h
        int     21h                     ; flush keyboard
        cmp     al,0dh                  ; enter ?
        je      disfatal
        or      al,20h
        cmp     al,63h
        jne     loopwaitchar
        jmp     nodiserr
disfatal:
        mov     ax,4c00h
        int     21h

...


have_file_size:
        mov     filesize,bx
        clc
        ret
GetDosVer:
        call    IsMSDOS
        test    ax,2000h
        jnz     SHORT dosverdone
        push    si
        mov     si,offset logocopy
        mov     byte ptr [si],0c3h
        pop     si
dosverdone:
        ret
getdosverend:

Check_DosVer ENDP
```