LiuTaoTao's Win95 programming page

Other

- Debug in Win95
- Back Doors of win95
- Debug Register
- CR0
- EFLAG
- misc

Win95 Programming resources

- Unauthorized Windows 95 Update

Download:

- VtoolsD 95 v2.03
- Win95 DDK
- companion disk of Unauthorized Windows 95
- companion disk of Undocumented Windows

EMAIL me if you can help me: mailto:liutt@371.net

Hello World

- Tutorial1: Win32 ASM sample
- Sample 32bit PE in ASM make in BC++5.0
- NE, WinMain, BC++5.0
- NE, cpp + asm, BCC5 + TASM5
- PE, WinMain, BC++5.0
- Sample 16/32bit DLL make in BC++5.0
- Call the 16/32bit DLL in NE/PE make in BC++5.0
- Sample 32bit DLL make in BC++5.02
- GUI PE
-

DDK tutorial

- DDK98: DynaLoad VxD & Loader, in ASM
- DDK98: DynaLoad VxD & Loader, in ASM & CPP

```
test.cpp:

#include "stdio.h"
void main()
{
        printf("Hello World");
}
```

```
1. NE, BCC 5.0:
        BCC -tWE -ml test.cpp

2. PE, BCC32 5.0:
        BCC32 test.cpp

3. PE, CL 5.0:
        CL test.cpp
```

Basic Structure

- 0x1 K32OBJ_SEMAPHORE
- 0x2 K32OBJ_EVENT
- 0x3 K32OBJ_MUTEX
- 0x4 K32OBJ_CRITICAL_SECTION
- 0x5 K32OBJ_PROCESS PDB: Process DataBase
- 0x6 K32OBJ_THREAD R3TCB: Ring3 Thread Control Block
- 0x7 K32OBJ_FILE
- 0x8 K32OBJ_CHANGE
- 0x9 K32OBJ_CONSOLE
- 0xA K32OBJ_SCREEN_BUFFER
- 0xB K32OBJ_MEM_MAPPED_FILE
- 0xC K32OBJ_SERIAL
- 0xD K32OBJ_DEVICE_IOCTL
- 0xE K32OBJ_PIPE
- 0xF K32OBJ_MAILSLOT
- 0x10K32OBJ_TOOLHELP_SNAPSHOT
- 0x11K32OBJ_SOCKET
- VMCB Virtual Machine Control Block
- K16TDB Win16 Task Data Block
- EDB Environment DataBase
- 16bit Module 16bit NE module
- R0CONTEXT Ring0 Context
- TIB FS: Thread Information Block
- CONTEXT Context defined in winnt.h
- Mutex
- Heap
- Client_Reg_Struc
- Other structures

- WND
- MSGQUEUE

- TCB Ring0 Thread Control Block
- DDB Device Data Block
- DeviceInfo

Ring0 CONTEXT

R0CONTEXT
who know this structure ?
use in _ContextSwitch

```
struct R0CONTEXT{
        DWORD PGTPTR;       //00 Page Table PTR
        WORD  Tables;       //04 Page Tables
        DWORD pStruct1;     //08
        DWORD hNextContext; //0c tail_to_head
        }
struct Struct1{
        DWORD min_Page; //00 << 12=min address
        DWORD max_Page; //04 << 12=max address
        DWORD ?;
        DWORD Mutex;    //0c
        DWORD hMutex;   //10
        }



        }
```

see also:

PDB	offset 1c	goo
TDBX	offset 8	goo
_ContextSwitch		

Win32 VxD services

01 VMM
2a VWIN32
2b VCOMM
VxDCall

See Also:

_Register_Win32_Services
If you want to make a VxD with Win32VxDService, you must call this service during init to register it.

0001 VMM.vxd, max 28

```
0x010000     K0_3_PageReserve            ;
0x010001     K0_5_PageCommit             ;
0x010002     K0_3_PageDecommit           ;
0x010003     K0_1_PagerRegister          ;
0x010004     K0_2_PageQuery              ;
0x010005     K0_2_HeapAllocate           ;
0x010006     K0_0_ContextCreate          ;
0x010007     K0_1_ContextDestroy         ;
0x010008     K0_4_PageAttach             ;
0x010009     K0_2_PageFlush              ;
0x01000a     K0_2_PageFree               ;
0x01000b     K0_1_ContextSwitch          ;
0x01000c     K0_3_HeapReallocate         ;
0x01000d     K0_4_PageModifyPermissions  ;
0x01000e     K0_3_PageQuery              ;
0x01000f     K0_0_GetCurrentContext      ;
0x010010     K0_2_HeapFree               ;
0x010011     K0_3_RegOpenKey             ;
0x010012     K0_3_RegCreateKey           ;
0x010013     K0_1_RegCloseKey            ;
0x010014     K0_2_RegDeleteKey           ;
0x010015     K0_5_RegSetValue            ;
0x010016     K0_2_RegDeleteValue         ;
0x010017     K0_4_RegQueryValue          ;
0x010018     K0_4_RegEnumKey             ;
0x010019     K0_8_RegEnumValue           ;
0x01001a     K0_6_RegQueryValueEx        ;
0x01001b     K0_6_RegSetValueEx          ;
0x01001c     K0_1_RegFlushKey            ;
0x01001d     K0_6_1d                     ;
0x01001e     K0_2_GetDemandPageInfo      ;
0x01001f     K0_2_BlockOnID              ;
```

```
0x010020     K0_1_SignalID               ;
0x010021     K0_3_RegLoadKey             ;
0x010022     K0_2_RegUnloadKey           ;
0x010023     K0_3_RegSavKey              ;
0x010024     K0_2_RegRemapPreDefKey      ;
0x010025     K0_5_PageChangePaper        ;
0x010026     K0_5_RegQueryMultipleValues ;
0x010027     K0_4_RegReplaceKey          ;
0x010028     K0_0_28                     ;

--------------------------------------

002a VWIN32.VXD, max 4e


0x2a0000     K0_0_GetVersion" ,
0x2a0001     K0_a_LoadVWin32CodePointers" ,
0x2a0002     K0_0_GetSystemTime" ,
0x2a0003     K0_8_GetVWin32PointersFromCaller" ,
0x2a0004     K0_1_BlockOnSemaphore_1" ,
0x2a0005     K0_1_Signal_Sema_NoSwtch",
0x2a0006     K0_3_Create_Semaphore",
0x2a0007     K0_1_Destroy_Semaphore" ,
0x2a0008     K0_9_VWIN32_CreateThread" ,
0x2a0009     K0_1_Sleep",
0x2a000a     K0_2_WakeThread" ,
0x2a000b     K0_2_TerminateThread" ,
0x2a000c     K0_1_2a000c",
0x2a000d     K0_3_VWIN32_QueueUserApc" ,
0x2a000e     K0_0_VWIN32_Initialize" ,
0x2a000f     K0_4_VWIN32_QueueKernelApc" ,
0x2a0010     K0_2_Int21_Call",                go
0x2a0011     K0_6_IFSMgr_Win32DupHandle",
0x2a0012     K0_1_BlockThreadSetBit" ,
0x2a0013     K0_2_AdjustThreadExecPrio",
0x2a0014     K0_2_GetThreadContext",
0x2a0015     K0_2_SetThreadContext",
0x2a0016     K0_5_ReadProcessMemory" ,
0x2a0017     K0_5_WriteProcessMemory" ,
0x2a0018     K0_1_VMCPD_Get_CR0_State" ,
0x2a0019     K0_2_VMCPD_Set_CR0_State" ,
0x2a001a     K0_1_SuspendThread",
0x2a001b     K0_1_ResumeThread",
0x2a001c     K0_0_2A001c",
0x2a001d     K0_1_WaitCrst",
0x2a001e     K0_1_WakeCrst",
0x2a001f     K0_b_2a001f",
0x2a0020     K0_1_VMCPD_Get_Version" ,
0x2a0021     K0_2_Set_Thread_Win32_Pri",
0x2a0022     K0_3_Boost_With_Decay" ,
0x2a0023     K0_4_Set_Inversion_Pri" ,
0x2a0024     K0_2_Release_Inversion_Pri_ID",
0x2a0025     K0_1_Release_Inversion_Pri" ,
```

```
0x2a0026     K0_2_Attach_Thread_to_Group",
0x2a0027     K0_2_Set_Thread_Static_Boost" ,
0x2a0028     K0_2_Set_Group_Static_Boost" ,
0x2a0029     K0_2_Int31_Call",                go
0x2a002a     K0_1_Int41_Call" ,
0x2a002b     K0_0_BlockForTermination" ,
0x2a002c     K0_1_TerminationHandler" ,
0x2a002d     K0_2_2a002d",
0x2a002e     K0_3_BlockSingleWnod" ,
0x2a002f     K0_5_BlockMultipleWnod" ,
0x2a0030     K0_2_VWIN32_SetEvent" ,
0x2a0031     K0_0_2A0031",
0x2a0032     K0_1_2a0032",
0x2a0033     K0_1_InitUserAPCList" ,
0x2a0034     K0_1_2a0034",
0x2a0035     K0_1_Signal_Sema_NoSwth" ,
0x2a0036     K0_1_SysCtl_Krnl32_initialized"
0x2a0037     K0_3_CommonFaultPopup" ,
0x2a0038     K0_0_VWIN32_ForceCrsts" ,
0x2a0039     K0_2_2a0039",
0x2a003a     K0_0_FreezeAllThreads" ,
0x2a003b     K0_0_UnFreezeAllThreads" ,
0x2a003c     K0_1_IFSMgr_Ring0_FileIO",
0x2a003d     K0_2_Attach_Thread_To_Group",
0x2a003e     K0_0_ActiveTimeBiasSet" ,
0x2a003f     K0_5_ModifyPagePermission" ,
0x2a0040     K0_4_2a0040",
0x2a0041     K0_2_ForceLeaveCrst" ,
0x2a0042     K0_3_ForceEnterCrst" ,
0x2a0043     K0_2_VMCPD_SetThreadExcptType",
0x2a0044     K0_1_VTD_Get_Real_Time" ,
0x2a0045     K0_0_Sysctl_SET_DEVICE_FOCUS",
0x2a0046     K0_1_UnFreezeThread" ,
0x2a0047     K0_1_VMM_Replace_Global_Env" ,
0x2a0048     K0_0_Sysctl_KERNEL32_SHUTDOWN",
0x2a0049     K0_3_SubUnk4" ,
0x2a004a     K0_2_VWIN32_AddSysCrst" ,
0x2a004b     K0_3_VWIN32_AddSysCrst" ,
0x2a004c     K0_1_Cancel_Time_Out" ,
0x2a004d     K0_1_2a004d",
0x2a004e     K0_1_Set_Reflect_Kkey" ,

-------------------------------------------------
2a001a  1   K0_SuspendThread    Ring0TCB
2a001b  1   K0_ResumeThread             Ring0TCB
-------------------------------------------------
```

```
002b VCOMM.vxd, max 1b

00     K0_0_VCOMMGetVersion
```

```
01      K0_1_VCOMMOpenComm
02      K0_3_VCOMMSetupComm
03      K0_2_VCOMMEscapeCommFunction
04      K0_2_VCOMMGetCommMask
05      K0_2_VCOMMGetCommProperties
06      K0_2_VCOMMGetCommState
07      K0_2_VCOMMGetCommTimeouts
08      K0_2_VCOMMPurgeComm
09      K0_2_VCOMMSetCommEventMask
0a      K0_2_VCOMMSetCommState
0b      K0_2_VCOMMSetCommTimeouts
0c      K0_2_VCOMMTransmitCommChar
0d      K0_4_VCOMMWaitCommEvent
0e      K0_2_VCOMMGetModemStatus
0f      K0_6_VCOMMWriteComm
10      K0_6_VCOMMReadComm
11      K0_3_VCOMMClearCommError
12      K0_1_VCOMMCloseComm
13      K0_1_VCOMMGetLastError
14      K0_2_VCOMMDequeueRequest         //do nothing
15      K0_1_VCOMMQueryFriendlyName
16      K0_3_VCOMMGetCommConfig
17      K0_3_VCOMMSetCommConfig
18      K0_2_VCOMMGetWin32Error
19      K0_1_VCOMMFlushFileBuffers
1a      K0_8_VCOMMDeviceIOControl
```

---

back to VWIN32's Win32VxDcall

```
// VxDCall is probably the most important undocumented Win32 API
DWORD (WINAPI *VxDCall)(DWORD srvc, DWORD eax, DWORD ecx);

#define VWIN32_INT21_CALL       0x2A0010
#define VWIN32_INT31_CALL       0x2A0029
#define DosCall(eax, ecx)       VxDCall(VWIN32_INT21_CALL, (eax), (ecx))
#define DPMICall(eax, ecx)      VxDCall(VWIN32_INT31_CALL, (eax), (ecx))

        // Get linear base address for TDB by calling
        // DPMI INT 31h function 6 (Get Selector Base).
        // Even though we're a 32-bit app, we get the 16-bit
        // DPMI services, since we're calling indirectly.
        _asm mov bx, tdb
        DPMICall(0x0006, 0);
        _asm mov word ptr tdb_base+2, cx
        _asm mov word ptr tdb_base, dx
        printf("TDB base: %08lXh\n", tdb_base);
```

```
        // Call DOS INT 21h function 62h (Get PSP)
        DosCall(0x6200, 0);
        _asm mov psp, bx
        printf("PSP: %04Xh\n\n", psp);

DWORD GetSelectorBase(WORD sel)
{
    #define VXDCALL_ORD         1
    #define VWIN32_INT31_CALL   0x2A0029
    #define DPMICall(eax, ecx)  VxDCall(VWIN32_INT31_CALL, (eax), (ecx))

    static DWORD (WINAPI *VxDCall)(DWORD srvc, DWORD eax, DWORD ecx) =
        (DWORD (WINAPI *)(DWORD,DWORD,DWORD)) 0;
    if (! VxDCall)
        if (! (VxDCall = GetK32ProcAddress(VXDCALL_ORD)))
            return -1;

    _asm mov bx, sel
    DPMICall(0x0006, 0);    // DPMI Get Selector Base function
    // put CX:AX return value into EAX
    _asm mov ax, cx
    _asm shl eax, 16
    _asm mov ax, dx
}

see also:
        chgdir.c from Unauthorized Windows95
        context.cpp     go
```

File Format

.SYM Microsoft Debug Symbol file
PE Win32 EXE file format
NE Win16 EXE file format LX attach\lxexe.txt

Selector

In krnl386.exe, there are many API about selector

But in kernel32.dll, no API about selector available.

See Also:

- DPMI
- source:getBase in cpp1.cpp

```
krnl386.exe APIs about selector:

 170: ALLOCCSTODSALIAS
 171: ALLOCDSTOCSALIAS
 172: ALLOCALIAS
 173: __ROMBIOS
 174: __A000H
 175: ALLOCSELECTOR
 176: FREESELECTOR
 177: PRESTOCHANGOSELECTOR
 178: __WINFLAGS
 179: __D000H
 180: LONGPTRADD
 181: __B000H
 182: __B800H
 183: __0000H
 184: GLOBALDOSALLOC
 185: GLOBALDOSFREE
 186: GETSELECTORBASE
 187: SETSELECTORBASE
 188: GETSELECTORLIMIT
 189: SETSELECTORLIMIT
 190: __E000H
 191: GLOBALPAGELOCK
 192: GLOBALPAGEUNLOCK
 193: __0040H
 194: __F000H
 195: __C000H
 196: SELECTORACCESSRIGHTS
```

```
mov     ecx,es
lar     eax,ecx
jnz     @@1
cmp     ah, 0F3h
;// Present, ring 3, writeable, accessed data segment
jnz     @@1
```

DR7 - Debug control

```
    3                       2                       1
 1  0  9 8  7  6  5 4  3  2  1 0  9  8  7 6  5 4   3  2 1 0   9   8   7   6   5   4   3   2   1
[LEN3][RW3][LEN2][RW2][LEN1][RW1][LEN0][RW0][0 0][GD][0 0 1][GE][LE][G3][L3][G2][L2][G1][L1][G0][
```

L? Local breakpoint DR? (0=off)
G? Global Breakpoint DR? (0=off)

GD Avaible on 486i+ - setting it enables a breakpoint on access to
debug registers the GD bit is cleared by the processor on entry to the exception
handler

RW? Condition on DR? - 0 0 = execution
0 1 = Data write
1 0 = Undefined
1 1 = Data Read/writes but NOT execution

LEN? - Operand length
0 0 = Byte or instruction execution
0 1 = Word
1 0 = Undefined
1 1 = Dword

****
DR{0-4} - must contain *LINEAR* address of address!
e.g. xor eax,eax |mov ax,cs | shl ax, 5 | mov ebx,offset breakhere | add eax,ebx | mov dr0,eax
****
DR6 Debug status register

Bits

```
   3                   2                           1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6   5    4    3   2 1 0 9 8 7 5 4   3    2    1    0
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 [BT] [BS] [BD] [0 1 1 1 1 1 1 1 [B3] [B2] [B1] [B0]
```

B{0-3} set for the triggering DR{0-3}
BD - set if next instruction will access any debug register
BS - set if exception caused by Trap-flag
BT - When a task with Trap bit set in the TSS has been switched to

---

```
;drx.asm from Stone, bug fixed by me.
;Can only run in DOS, not for Win95.
;tasm drx
;tlink drx

.model small            ; It's a flaw of mine ... I really like this model
                        ; I know I should do a .com with the tiny model..
                        ; but I just love the small :>
.stack 100h             ; Plenty stack ;>
.386P                   ; We must be in priveledgemode since the debugregisters
                        ; can only be accessed from priveledged mode

.data
.code
start:
        mov ax, @data            ; Make DS&ES point to the DATA
        mov ds,ax
        mov es,ax

        xor ax,ax               ; Hook int 1 - the debug exception
        mov ds,ax
        mov bx, offset int1h
        mov word ptr ds:[4h], bx
        mov word ptr ds:[6],cs

        mov     eax,dr7
        and     eax,not 0F0003h ;set execute bp,1 byte len
        or      eax,3           ;local
```

```
        mov dr7,eax                     ; Execute, dr0

        xor     ebx,ebx
        mov bx,offset flaf              ; calculate linear address of flaf
        xor eax,eax
        mov ax,cs
        shl eax,4
        add eax,ebx

        mov dr0, eax                    ; Let breakpoint trigger on execution of flaf
        nop
        nop                     ;necessary!!

flaf:
        nop
        mov ax,4c00h                    ; terminate!
        int 21h

int1h:
        mov ah,2h                       ; Write "a" on the screen when the BP triggers
        mov dl,'a'
        int 21h
        xor     eax,eax
        mov     dr0,eax         ;avoid dead lock
        iret
end start
```

Articles

- About Debug Symbol
- 32bit DPMI in Win95/98
- ××SOFT-ICE2.64×κL'×

---

about speed of IDA

IDAW is a Win32 console program. It use standard console display
API's, and all these API's are very slowly.

We can speed this by write video memory directly just as we do in
DOS. Every Win32 console program has a correspondent VM. From the
VMcb structure we can get 'CB_High_Linear' which is the base memory
of the VM. So the console screen memory is
        CB_High_Linear + B8000h
Write directly to this memory will speed your console program.

This is about page memory management of Win95.

In vmm.vxd of Win95, I find:

```
----------------------------------------
pIfVMcb         proc near               ; CODE XREF: _ModifyPageBits+92p
                                        ; mmMapPhys+73p ...

a_LineAddress   = dword ptr  8

                push    ebp
                mov     ebp, esp
                mov     edx, [ebp+a_LineAddress]
                mov     eax, edx
                shr     eax, 0Ch
                shl     eax, 2
                lea     ecx, [eax+0FF800000h]
                and     ecx, 0FFFFF3FFh
                shr     ecx, 0Ah
                test    byte ptr [ecx+0FF800000h], 1
                jz      short loc_8CCD
                test    byte ptr [eax+0FF800000h], 1
                jz      short loc_8CCD
                cmp     dword ptr [edx+10h], 'bcMV'
                mov     eax, 1
                jz      short loc_8CCF

loc_8CCD:                               ; CODE XREF: pIfVMcb+24j pIfVMcb+2Dj
                sub     eax, eax

loc_8CCF:                               ; CODE XREF: pIfVMcb+3Bj
```

```
                pop     ebp
                retn    4
pIfVMcb         endp
----------------------------------------
To make it clear, I change it to:
----------------------------------------
pIfVMcb         proc stdcall LineAddress:DWORD

                mov     edx, [LineAddress]
                mov     eax, edx
                shr     eax, 0Ch
                lea     ecx, [eax*4+0FF800000h]
                shr     ecx, 0Ch
                test    byte ptr [ecx*4+0FF800000h], 1          ;PDE
                jz      short @@1

                test    byte ptr [eax*4+0FF800000h], 1          ;PTE
                jz      short @@1

                cmp     dword ptr [edx+10h], 'bcMV'
                mov     eax, 1
                jz      short @@2
@@1:
                sub     eax, eax
@@2:
                ret
pIfVMcb         endp
----------------------------------------
```

Can you understand it ?
I think I understand it today. If you already understand it,
you needn't read next part.

From books, we can get that if we want to know a linear address
is readable, we can

```
        mov     ecx,[LinearAddress]
        shr     ecx,16h                 ;=IPDE

        mov     eax,cr3
        and     eax,0fffff000h

        Phys2Linear     eax

        mov     eax,[eax+ecx*4]         ;PDE
        test    eax,1                   ;test if PTE present
        jz      ERROR

        and     eax,0fffff000h

        Phys2Linear     eax

        mov     ecx,[LinearAddress]

        shr     ecx,12
        and     ecx,3ffh                ;=IPTE
```

```
        mov     eax,[eax+ecx*4]         ;=PTE
        test    eax,1                   ;test if page present
        jz      ERROR
```

This is to say:

```
        LinAddr = ( (IPDE<<10 +IPTE) <<12 )+ AddrInPage
        PDE = Phys2Lin(CR3 & 0xfffff000) + IPDE*4
        PTE = Phys2Lin(*PDE & 0xfffff000) + IPTE*4
```

But from the pIfVMcb, it seems:

```
        PDE = 0xff800000 + ( PTE >> 12 )*4
        PTE = 0xff800000 + (LinAddr >> 12)*4
```

Is this true ? I find it YES!

```
        LinAddr = ( (IPDE<<10 +IPTE) <<12 )+ AddrInPage
        LinAddr >> 12 = IPDE << 10 +IPTE
        PDE = Phys2Lin(CR3 & 0xfffff000) + IPDE*4
            = 0xffbfe000 + IPDE*4
            = 0xffbfe000 - 0x3fe000 + 0xff800*4 + (LinAddr >>12 >>10)
            = 0xff800000 + ( 0xff800000 + (LinAddr >> 12)*4 ) >> 12 *4
        PTE = Phys2Lin(*PDE & 0xfffff000) + IPTE*4
            = 0xff800000 + (LinAddr >> 12)*4
            = 0xff800000 + IPTE*4 + (IPDE << 10)*4
            = ( 0xff800000 + (IPDE << 10)*4 ) + IPTE*4
```

so if we assume

```
        Phys2Lin(*PDE & 0xfffff000) == 0xff800000 + (IPDE << 10)*4
```

then all is OK. I think it must be.

So, if we want to know if a linear address is readable, we can

```
                mov     eax, [LineAddress]
                shr     eax, 12
                lea     ecx, [eax*4+0FF800000h]
                shr     ecx, 12
                test    byte ptr [ecx*4+0FF800000h], 1          ;PDE
                jz      ERROR
                test    byte ptr [eax*4+0FF800000h], 1          ;PTE
                jz      ERROR
```

Micro$oft is really clever!

But why no book say this ?

Exports Functions

32Bit

- Kernel32.dll
- User32.dll

16Bit

- krnl386.exe
- gdi.exe
- user.exe

See also

- Win32 VxD Services

Interrupt in Win95

- Int 22h
- Int 2fh
- Int 31h
- Int 68h

Can you help me ?

- about Html Help Workshop

How can I make a .COL file ?
How to add a subsets selection just as MSDN98 ?
How to add a 'locate' menu item ?

If I know a 32bit proc(proc32) is thunk to a 16bit proc(proc16).
How can I get the address of proc16 from proc32 ?

How to make a directory tree in HTML ?

'Font Editor' from VC++ 1.0
can edit a .FNT file. We can make a .FON file from this .FNT.

You can use Font Editor to create and edit only raster fonts.

After creating a new font with Font Editor, you must add the new font to a font resource file. For information about creating adding fonts to the font resource file, see the Microsoft Windows Guide to Programming.

Borland 'Resource Workshop' can edit a .FON file directly.

My Source

- exp.cpp Get ProcAddress from a module with ordinary number. Get ProcName from a module with ordinary number.
- context.cpp
  Walk Context of Win95 via Win32 VxD service.
- other source

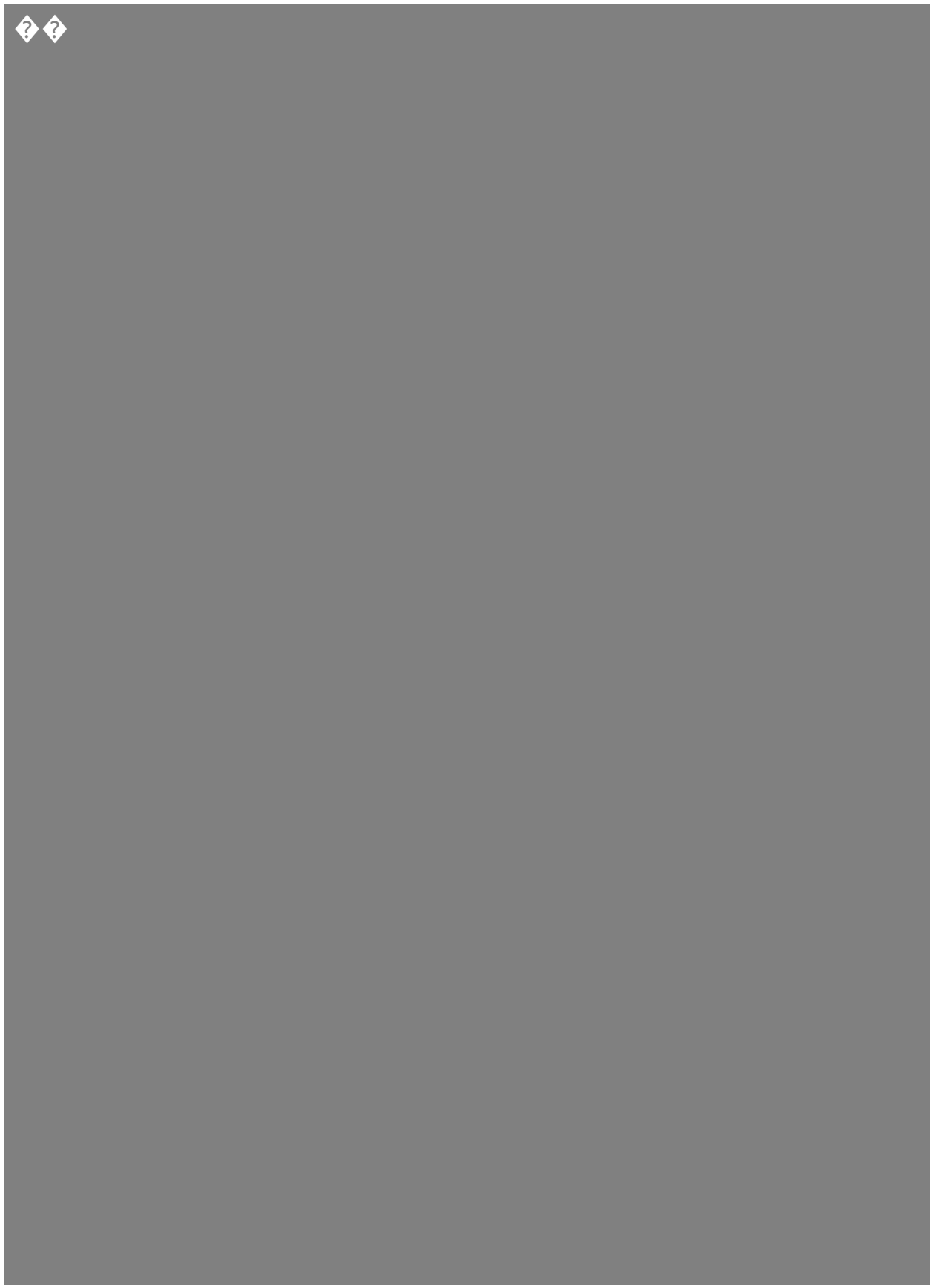
Good Books:
For Beginners:

- System Programming for Windows95, by Walter Oney

Advanced:

- Unauthorized Windows95, by Andrew Schulman
- Windows95 System Programming Secrets, by Matt Pietrek
  中文译本：
- Windows95开发指南，电子工业出版社，ISBN7-5053-2897-2/TP.966
- Windows95系统编程奥秘，电子工业出版社，ISBN7-5053-3278-3/TP.1226

```
??

RC.EXE

sdk32\mstools\binw16\rc.exe

7-11-95 9:50 46,605 RC.EXE

Microsoft (R) Windows Resource Compiler Version 4.00
Copyright (C) Microsoft Corp. 1985-1993. All rights reserved.


Usage: rc [switches] .RC input file [.EXE output file]
Switches:

    -r Create a .RES file only; don't process .EXE
    -l *Create an application that uses LIM 3.2 EMS
    -e *Create a driver which uses EMS memory
    -m *Set Multiple Instance flag
    -p *Create a Private Library
    -t *Create a protected mode only application
    -k *Keep segments in .DEF file order (do not sort segments for fast
    load)
    -40 *Mark as 4.0 (or above) app (default)
    -31 *Mark as 3.1 (or above) app
    -30 *Mark as 3.0 (or above) app
    -v Verbose (print progress messages)
    -d Define a symbol
    -fo Rename .RES file
    -fe Rename .EXE file
    -i Add a path for INCLUDE searches
    -x Ignore INCLUDE environment variable
    -z Skip check for RCINCLUDE statements

    * == Cannot be used when -r is specified.
```



DLL16:

MSC600 & MSC700 can not make any Windows program without SDK16. They even do not
include windows.h. So MSVC++1.0 is a good choice to make DLL16.

Make DLL16 for Win95 need RC.EXE 4.0 from SDK32.
Only this RC.EXE support '-40' option.

Files:

- sync.htm
- Win32 VxD Services
- Page Memory Meanage
- Articles
- Interrupt in Win95
- User.exe export API

Microsoft Thunk compiler:

sdk32\mstools\bin\i386\thunk.exe
98ddk\bin\win98\thunk.exe

7-11-95 9:50 225,280 THUNK.EXE

Microsoft (R) Thunk Compiler Version 1.8 for Windows 95. May 11 1995 13:16:19

Thunk compiler usage
thunk [{-|/}options] infile[.ext]

where options include:
? Display this help screen
h Display this help screen
o <name> Override default output filename
p<n> Change 16-bit structure alignment (default = 2)
P<n> Change 32-bit structure alignment (default = 4)
t <name> Override default stem name

Nx <name> Name segment or class where x is
C32 32-bit code segment name
C16 16-bit code segment name

VxD overall

VxD calls
System Control Messages
DDB
DeviceInfo for dynamic VxD
VxDLocationList for static VxD

Walk VxD with DDB next:

Name	Vers	ID	DDB	Control	V86 API	PM API	#Srvc
VMM	4.10	1h	C0011360	C00027A8	C0002F50	C0002F50*	446
VCACHE	3. 1	48Bh	C004E2CC	C004D608	C004E1D8	C004E1D8	29
PERF	4. 0	48h	C005926C	C00592BC			5
VPOWERD	4.10	26h	C14264C8	C14265D0	C1814E20*	C1814E20*	34
VPICD	4.10	3h	C0051BD0	C004FF39	C0050D94	C0050D94	32
VrtwD	1. 4	33A2h	C006D0B4	C006A870	C006A8AB	C006A8AB	2
VTD	4. 0	5h	C005274C	C00529EE	C023477E	C023477E	18
VWIN32	4.10	2Ah	C002F8A8	C002FA64		C029689C*	54
VXDLDR	3. 0	27h	C004F3BC	C004F31C	C0251074	C0251074	19
NTKERN	4.16	4Bh	C002E740	C002EB55		C028B330	15
CONFIGMG	4. 0	33h	C0020A30	C0020AB4	C025F21C	C025F21C*	124
PCI	4.10	43h	C1422ED0	C1422F20			4
ISAPNP	4.10	3Ch	C1425190	C14251E0			0
BIOS	4.10	3Dh	C1424360	C14243B4	C1804084*	C1804084*	4
VCDFSD	3. 0	41h	C004F1EC	C004F180			4
IOS	3.10	10h	C003DEB4	C003A600	C02B5E18	C02B4294	21
PAGEFILE	4. 0	21h	C0073568	C00734C8		C023B606*	10
PAGESWAP	2.10	7h	C0056F14	C0056E84			10
PARITY	1. 0	8h	C0058208	C0058148			0
REBOOT	4. 0	9h	C0053630	C005325C		C02348EC*	5
VPBIOSD	1. 0	66h	C0070E20	C0070B48	C0070B72	C0070DF3	2
VDD	2. 0	Ah	C0076FF0	C0076890	C025BA3B	C025BA3B*	24
IGAMINI	3. 1		C14296C0	C1429A80			0
VSD	2. 0	Bh	C0055AEC	C005592C			4
COMBUFF	1. 0		C0032840	C003269C			0
VCD	3.10	Eh	C0073938	C00735D4		C023B890	13
MMD	3. 0	1025h	C0070AA4	C0070448	C0070A7E	C0070A7E	1
VMOUSE	4. 0	Ch	C0072CB4	C0072418	C02C0748	C02C3C1C*	12
MSMINI	1. 0		C00733D0	C0072F4C			0
ENABLE	4.10	37h	C007526C	C0074F82	C02C5647	C02C55B9*	10
VKD	2.10	Dh	C0076508	C00752E8		C02587CC*	25
VPD	3. 0	Fh	C0074270	C0073A00		C0073C9A	0
INT13	3.10	20h	C0074DC8	C007446A			5
PharLap	1. 0		C0070388	C006FC4C			0
VMCPD	4. 5	11h	C0058C98	C00582FC		C0058346	13
BIOSXLAT	1. 0	13h	C00582A8	C0058260			0
VNETBIOS	3. 0	14h	C006FA9C	C006ED20			8
NDIS	5. 0	28h	C001863C	C001910F			161
PPPMAC	3. 0	499h	C14387A4	C1438969	C1830E8C*	C1830E8C*	13
VTDI	3. 0	488h	C1433844	C14333E6			15
VIP	3. 0	489h	C1457474	C14495BA	C144943B*		14

```
MSTCP     3. 0    48Ah   C1464F50   C14592DA   C14592F9*             1
VDHCP     3. 0    49Ah   C1469798   C146AB2C   C18607FA*  C18607FA*  6
VNBT      3. 0    49Bh   C147CE00   C147D085   C147D1C7*  C147D1C7*  1
DOSMGR    4. 0     15h   C0057110   C0056F6C   C0257144              19
VMPOLL    4. 0     18h   C00573B0   C0057290                        4
VFIXD     1. 0           C006E3B4   C006E44F   C006E445   C006E43B   0
JAVASUP   1. 0           C00207AC   C002080C                        0
VCOMM     1. 0     2Bh   C003255C   C0032250   C029C170   C029C170*  36
VCOND     1. 0     38h   C004F110   C004F160   C02B8F58*  C02B8F96*  2
VTDAPI    4. 0    442h   C00591F0   C0059161              C02BF8C8*  0
VMSGD     1. 0     62h   C007134C   C0070EA4              C0070ECC*  0
VFLATD    4. 0    11Fh   C00782B0   C0077FD8              C02C8528   2
mmdevldr  1. 0    44Ah   C142545C   C14254B8   C1817CF8*  C1817CF8*  7
Display1  1. 0           C142A0D0   C0076941                        0
DiskTSD   3.10           C142D7D4   C142D570                        0
voltrack  3.10     90h   C143098C   C1430420                        0
BIGMEM    4. 0           C1423CFC   C1423A50                        0
SPAP      1. 0           C14404DC   C1440530   C1440584*  C1440596*  0
HSFLOP    3.10           C147F5F0   C14802B0                        0
ESDI_506  3.10     8Dh   C1482D10   C1480F90                        0
LPTENUM   4.10           C142F780   C142F1F0                        0
SERENUM   4.10           C14668F8   C14668D0                        0
vjoyd     1. 3    449h   C142C8E4   C142C93C              C188AB90*  19
FIOLOG    1. 0           C14A01A0   C14A02C8                        0
TRW       0.65   1999h   C14C1C8C   C14BFC43   C14BFDD1*  C14BFDDA*  0
VDMAD     4. 0      4h   C00555D0   C005387C                        36
V86MMGR   1. 0      6h   C0056CFC   C00560DF              C0238020   25
SPOOLER   1. 0     2Ch   C003FD8C   C003FCA8                        17
UDF       3. 0     4Dh   C0040370   C00405F0                        0
VFAT      3. 0    486h   C0041B18   C004D53C                        0
VDEF      3. 0           C004F8B8   C004F66C                        0
IFSMGR    3. 0     40h   C0037B80   C00329DC                        124
VFBACKUP  4. 0     36h   C00321B8   C0031AF8   C0031B58   C0031B59   10
SHELL     4. 0     17h   C0057ED4   C0057BA9   C02BF6F0   C02BF6F0*  29

And walk VxDs with DeviceInfo:

--DDB--- --VxDName-
c14c1c8c TRW
c14a01a0 FIOLOG
c142c8e4 VJOYD
c14668f8 SERENUM
c142f780 LPTENUM
c1482d10 ESDI_506
c147f5f0 HSFLOP
c147ce00 VNBT
c1469798 VDHCP
c1464f50 MSTCP
c1457474 VIP
c1433844 VTDI
c14404dc SPAP
c14387a4 PPPMAC
       1 wmidrv
c1423cfc BIGMEM
c143098c voltrack
```

```
c142d7d4 DiskTSD
       0 CDFS
       0 APIX
c14296c0 IGA
c142545c MMDEVLDR
c14264c8 VPOWERD
c1425190 ISAPNP
       1 PCIMP
c1422ed0 PCI
       1 hidvkd
c1424360 BIOS
       1 swenum
       1 wdmfs
```

So,

1. DeviceInfo list contains all Dynamic load VxDs,WDMs
2. WDM do not have DDB?
3. static VxDs not in DeviceInfo list, how to get their DeviceInfo?

and,

1. I search DDB of static VxD in whole memory range, and find no one followed by 'DLVX', so its sure static VxDs do not have DeviceInfo.
   Then how can I get its load info ?

   Real part of WinICE hooks int68 ax=5080,5081 to get this at load time.

1.
2. Have a look at:
            debugsys.inc   from Win95 DDK
3. Debug interrupt:
        int 2fh
        int 41h   go
        int 68h
4. VxD services:
        VMMcall Test_Debug_Installed

5. Dot Commands:
        .?
        .R [#] ------- Displays the registers of the current thread
        .VM [#] ------ Displays complete VM status
        .VC [#] ------ Displays the current VMs control block
        .VH [#] ------ Displays a VMM linked list, given list handle
        .VR [#] ------ Displays the registers of the current VM
        .VS [#] ------ Displays the current VM's virtual mode stack
        .VL ---------- Displays a list of all valid VM handles
        //.T ---------- Toggles the trace switch
        //.S [#] ------ Displays short logged exceptions starting at #, if specified
        //.SL [#] ------ Displays long logged exceptions just #, if specified
        //.LQ ---------- Display queue outs from most recent
        .DS ---------- Dumps protected mode stack with labels
        .VMM --------- Menu VMM state information
        . -- Display device specific info

see also:
        int41_0070 -- Register dot command (32 bit code)   go
        int41_0071 -- Register dot command (called by 16 bit code)   go
        int41_0072 -- Unregister dot command   go

        In Debug Version of Win95:   Retail:
        _Debug_Printf_Service   call int41/ax=73   ret
        In_Debug_Chr   call int41/ax=01
        Is_Debug_Chr   call int41/ax=03
        Out_Debug_Chr   call int41/ax=00

                                Retail
        _Debug_Flags_Service   ret 4
        Queue_Debug_String   ret 8

- Int 22

Int 22h
Int 22h service is a VMM fault hooked in VMM.vxd.

```
INT 22h - Win32 Protected mode interface requests API
        AX=02h -- Converts a physical address to a linear address in curent context
                  entry:
                      ECX=phisycal address
                  returns:
                      ESI=linear address
                      AX= 1 if success , otherwise 0
        AX=07h -- Check to see if an address is within a VxD object
                   entry:
                      DS:ESI = buffer to receive object name
                      BX  =  thread number
                      EDX  = linear address to query
                   returns:
                      If EAX == 0, EDX = base address of object
                      If EAX != 0, error
        AX=08h --  Get PDE for a specific context
                    entry:
                     BX = thread number
                     EDX = linear address
                    returns:
                     if EAX == 0, ECX = PDE
                     if EAX != 0, error
        AX=0Ah --    Get LDT base
                    entry:
                     BX = Thread ID
                    returns:
                      if EAX == 0
                      EDI = pointer to LDT
                      ECX = LDT limit
                      if EAX != 0, error
I find these:
        AX=00h -- Return AX=0f386h means debug installed.

        AX=01h -- Debug Query.
                entry:
                        esi: points to Dot cmd lines(after the dot).

        AX=02h -- stc and return

        AX=03h -- verify memory, esi:addr,cx:len

        AX=04h -- nothing
        AX=05h -- nothing

        AX=06h -- Get VxD service address
                entry:
                        ebx= VxD service num,(00010001 for Get_Cur_VM_Handle)
                returns:
                        eax= VxD service address, 0 for error


        AX=09h -- ?
                entry:


        AX=0bh -- Return AX= current Thread ID

        AX=0ch -- ?
                entry:
                        ebx=
```

```
Win32Asm .inc file for TASM by Ghiribzzo        -win32t2.zip


/*
 *  CR0 bit assignments
 */
#define PE_BIT          0       /* 1 = Protected Mode */
#define PE_MASK         (1 << PE_BIT)
#define MP_BIT          1       /* 1 = Monitor Coprocessor */
#define MP_MASK         (1 << MP_BIT)
#define EM_BIT          2       /* 1 = Emulate Math Coprocessor */
#define EM_MASK         (1 << EM_BIT)
#define TS_BIT          3       /* 1 = Task Switch occured */
#define TS_MASK         (1 << TS_BIT)
#define ET_BIT          4       /* 1 = 387 present, 0 = 287 present */
#define ET_MASK         (1 << ET_BIT)
#define PG_BIT          31      /* 1 = paging enabled, 0 = paging disabled */
#define PG_MASK         (1 << PG_BIT)


/*
 *  EFLAGs bit assignments
 */
#define CF_BIT          0
#define CF_MASK         (1 << CF_BIT)
#define PF_BIT          2
#define PF_MASK         (1 << PF_BIT)
#define AF_BIT          4
#define AF_MASK         (1 << AF_BIT)
#define ZF_BIT          6
#define ZF_MASK         (1 << ZF_BIT)
#define SF_BIT          7
#define SF_MASK         (1 << SF_BIT)
#define TF_BIT          8
#define TF_MASK         (1 << TF_BIT)
#define IF_BIT          9
#define IF_MASK         (1 << IF_BIT)
#define DF_BIT          10
#define DF_MASK         (1 << DF_BIT)
#define OF_BIT          11      /* Overflow flag */
#define OF_MASK         (1 << OF_BIT)
#define IOPL_MASK       0x3000  /* IOPL flags */
#define IOPL_BIT0       12
#define IOPL_BIT1       13
#define NT_BIT          14      /* Nested task flag */
#define NT_MASK         (1 << NT_BIT)
#define RF_BIT          16      /* Resume flag */
#define RF_MASK         (1 << RF_BIT)
#define VM_BIT          17      /* Virtual Mode flag */
#define VM_MASK         (1 << VM_BIT)
#define AC_BIT          18      /* Alignment check */
#define AC_MASK         (1 << AC_BIT)
#define VIF_BIT         19      /* Virtual Interrupt flag */
#define VIF_MASK        (1 << VIF_BIT)
#define VIP_BIT         20      /* Virtual Interrupt pending */
```

```
#define VIP_MASK        (1 << VIP_BIT)
```

- Make Files
- IDC Structure defines
- IDC Enum defines
- a .ASP file

Articls

- How to create kernel mode thread in Win95?
- How to hook int21?

```
Words:
        APC     Asynchronous Procedure Call
        Crst    Critical Section
        IFSMgr  the Installable File System Manager
        IT      Information Technology
        mutex   mutual exclusion
        RAD     Rapids Application Delevopment
        TLS     Thread Local Storage
        VMCPD   the Virtual Math Coprocessor Device
```

```
.sym -> .nms    nmsym /load file.sym
        .sym -> .map    ??
        .map -> .sym    msym    (softicew)
        .dbg -> .map    dbg2map (softicew)
```

```
VOID  _cdecl COAX_Timer_Handler();
        VMM_SEMAPHORE hSemaphore;
        void CreateThread(void)
        {

VWIN32_CreateRing0Thread((PVOID)Ring0ThreadHandler,4096,NULL,FALSE);
                hSemaphore = Create_Semaphore(0);
                Wait_Semaphore(hSemaphore,BLOCK_THREAD_IDLE);
        }

        // this is the RIng 0 thread Handler.  The OS calls this handler
immidiately.
        //also note that U cannot pass parameters, so if U wana access
any local variables,, U will have to make them global.
        VOID Ring0ThreadHandler(void)
        {

            Out_Debug_String("Inside Ring0ThreadHandler\n\r");
             CallDelayRoutine(1Sec);                    // and Pseudo
Function that calls a delay fro 1 sec .
            Signal_Semaphore(hSemaphore); // see that the hSemaphore is
globally declated.

// it is quite possible that the Semaphore may be

// signalled before U start waiting on it.. so  I Included

//a delay
            return ;
```

```
        }

        The parametes to the function Create ring 0 thread is defined in
the DDK documentation.
        [Felix Durairaj]         Felix Durairaj FelixDu@ATTACHMATE.COM
```

```
BeginProc PM_Int21Handler

     ;Your handler code here

     ;Return this way to allow the call to continue down the
     ; Int21h chain

     mov ecx, [SaveInt21_CS]      ;Set up the CS and IP values that
     mov edx, [SaveInt21_IP]      ; were saved when we hooked Int 21h
     VMMJmp   Simulate_Far_Jmp    ;Jump to the next handler in the chain

EndProc   PM_Int21Handler


If you want to return from the interrupt without calling down
the chain, do the following:

BeginProc PM_Int21Handler

     ;Your handler code here

     VMMCall   Simulate_Iret      ;Simulate a return from the interrupt

EndProc   PM_Int21Handler

This should work much better.

Terry Peterson
petersot@motsat.sat.mot.com
```

```
To do page-in you must
 set the flag of page as in transition
 alloc phys memory
 if not enough free
 than steal from other process or cache subsystem
 In the page
 then change the flag as present
 This is in theory
```

```
>         My question is:  Is there a way to call this 16-bit app more
> quickly, without interference
>         from the system?  The only Windows call made by the 16-bit app
> during this call is
>         DriverCallback(), which can be used in interrupt handlers.  Would
>         it
> be possible to
>         set this 16-bit app up as an interrupt handler and set that
> interrupt from the VXD?
```

> Tom Flaherty

Maybe this would help you.
You do not need to go by VPIC.

```
//----------------------------------------------------------------------------
void
signalEvent(DWORD dwApplicationID, DWORD dwPostMessageContext)
 {
  CLIENT_STRUCT saveRegs;
  WORD          stack;

  dwSignalRunning++;

         Save_Client_State(&saveRegs);

         Begin_Nest_Exec();

         //
         // user mode signal handler is defined as
         // void WINAPI VxdEventHandler( LPVOID lpvContext, WORD uApplicationID);
         //

         Simulate_Push( HIWORD(dwPostMessageContext) );
         Simulate_Push( LOWORD(dwPostMessageContext) );
         Simulate_Push( HIWORD(dwApplicationID) );
         Simulate_Push( LOWORD(dwApplicationID) );

         Simulate_Far_Call(uPostMessageSegment, uPostMessageOffset);
         Resume_Exec();

         End_Nest_Exec();

         Restore_Client_State(&saveRegs);

  dwSignalRunning--;
 }
```

Norbert Kawulski norbert@STOLLMANN.DE

```
//

// Build 16bit DLL with BC++5.0

// bcc -tWDE bcdll.cpp

// Build 32bit DLL with BC++5.0

// bcc32 -tWDE bcdll.cpp

//or

// \bc4\bin\bcc -c bcdll.cpp
// \bc4\bin\tlink /Twd bcdll.obj \bc5\lib\import.lib

//---------------------------------------------------------------------------
#include <windows.h>

#define DLLEXPORT __declspec(dllexport)

extern "C" void DLLEXPORT WINAPI Message(LPSTR msg);


//
// Display a message passed from the calling application
//
extern "C" void DLLEXPORT WINAPI Message(LPSTR msg)
   {
        MessageBox(NULL, msg, "Borland C++ DLL", MB_OK|MB_APPLMODAL);
   }

BOOL WINAPI DllEntryPoint(HINSTANCE hDLLInst, DWORD fdwReason, LPVOID lpvReserved)
{

/*
    switch (fdwReason)
    {
        case DLL_PROCESS_ATTACH:
            // The DLL is being loaded for the first time by a given process.
            // Perform per-process initialization here.  If the initialization
            // is successful, return TRUE; if unsuccessful, return FALSE.


            // In this DLL, we want to initialize a critical section used
            // by one of its exported functions, DLLFunction2().

//            InitializeCriticalSection(&gCriticalSection);

            break;

        case DLL_PROCESS_DETACH:
            // The DLL is being unloaded by a given process.  Do any
            // per-process clean up here, such as undoing what was done in
            // DLL_PROCESS_ATTACH.  The return value is ignored.


            // In this DLL, we need to clean up the critical section we
            // created in the DLL_PROCESS_ATTACH message.

//            DeleteCriticalSection(&gCriticalSection);
```

```
            break;

        case DLL_THREAD_ATTACH:
            // A thread is being created in a process that has already loaded
            // this DLL.  Perform any per-thread initialization here.  The
            // return value is ignored.

            break;

        case DLL_THREAD_DETACH:
            // A thread is exiting cleanly in a process that has already
            // loaded this DLL.  Perform any per-thread clean up here.  The
            // return value is ignored.

            break;
    }
*/
    return TRUE;
}
```

- IMTE
- MODREF
- TDBX
- PMcb
- LIST
- Event
- Exception_Handler_Struc

- PROCESS_INFORMATION
- PROCESS_HEAP_ENTRY
- HEAPENTRY32
- PROCESSENTRY32
- THREADENTRY32
- MODULEENTRY32
- CRITICAL_SECTION
- PUSHAD

```
#define K32OBJ_SEMAPHORE            0x1
#define K32OBJ_EVENT                0x2
#define K32OBJ_MUTEX                0x3
#define K32OBJ_CRITICAL_SECTION     0x4
#define K32OBJ_PROCESS              0x5
#define K32OBJ_THREAD               0x6
#define K32OBJ_FILE                 0x7
#define K32OBJ_CHANGE               0x8
#define K32OBJ_CONSOLE              0x9
#define K32OBJ_SCREEN_BUFFER        0xA
#define K32OBJ_MEM_MAPPED_FILE      0xB
#define K32OBJ_SERIAL               0xC
#define K32OBJ_DEVICE_IOCTL         0xD
#define K32OBJ_PIPE                 0xE
#define K32OBJ_MAILSLOT             0xF
#define K32OBJ_TOOLHELP_SNAPSHOT    0x10
#define K32OBJ_SOCKET               0x11
```

PUSHAD

```
/*
 *  Registers as they appear on the stack after a PUSHAD.
 */

struct Pushad_Struc {
    ULONG Pushad_EDI;                   //00 Client's EDI
    ULONG Pushad_ESI;                   //04 Client's ESI
    ULONG Pushad_EBP;                   //08 Client's EBP
    ULONG Pushad_ESP;                   //0c ESP before pushad
    ULONG Pushad_EBX;                   //10 Client's EBX
    ULONG Pushad_EDX;                   //14 Client's EDX
    ULONG Pushad_ECX;                   //18 Client's ECX
    ULONG Pushad_EAX;                   //1c Client's EAX
};


include vmm.inc

pmcb_s  struc
PMcb_Flags   dd  ?  ; //00 control-block flags
PMcb_Parent  dd  ?  ; //04 parent of the protected-mode application
```

```
//debug:
//      WORD    PMcb_PSP;       //08
//      WORD    PMcb_signature; //0a    0cbb0h

//      ?       PMAPP_CB_Area;  //offset from PMAPPCBArea


pmcb_s  ends


Application control block. Contains information about a protected-mode application.

see also:
        Get_Cur_PM_App_CB
```

```
struct list
        First           dd      ?               //00 first node
        _4              dd      ?               //04
        Signature       dd      ' LAR'          //10h Signature

List_Node structure
        hList   dd      ?       //00    hList this node belong to
        Next    dd      ?       //-08 next node

from DDK95:

struct nodelist_s {
    struct nodelist_s *nl_Next;         // next node element
    struct nodelist_s *nl_Previous;     // previous node element
    struct devnode_s  *nl_ItsDevNode;   // device node represented
    struct Log_Conf   *nl_Test_Req;     // test resource alloc request
    ULONG  nl_ulSortDWord;              // sort order
};


Contains information about an element in a node list.
Although additional members can be appended to this
structure, the first three members must not be changed.


struct nodelistheader_s {
    struct nodelist_s *nlh_Head;  // First node element
    struct nodelist_s *nlh_Tail;  // Last node element
};


Contains information about the elements in a node list.
```

```
        include vxdldr.inc
struct DeviceInfo {
    struct DeviceInfo *DI_Next;
    UCHAR  DI_Loaded;
    struct VxD_Desc_Block *DI_DDB;
    USHORT DI_DeviceID;
    CHAR   *DI_ModuleName;
    ULONG  DI_Signature;
    ULONG  DI_ObjCount;
    struct ObjectInfo *DI_ObjInfo;
    ULONG  DI_V86_API_CSIP;
    ULONG  DI_PM_API_CSIP;
};

see also:
```

```
        VxDcall VXDLDR_GetDeviceList
```

```
struct  Event{
        DWORD   EV_Flags;       //08h
        DWORD   EV_?            //0c
        DWORD   EV_hTimeOut     //10
        DWORD   EV_R0TCB        //14
        DWORD   EV_oEventName;  //1ch offset of event name
        }
```

```
EXCEPTION_RECORD                STRUC
        ExceptionCode           DD ?
        ExceptionFlags          DD ?            ; bit 0=1..uncontinuable
        EXCEPTION_RECORD_PTR    DD ?
        ExceptionAddress        DD ?
        NumberParameters        DD ?
        EXCEPTION_MAXIMUM_PARAMETERS  EQU 15    ; pro forma
        ExceptionInformation    DD EXCEPTION_MAXIMUM_PARAMETERS DUP (?)
EXCEPTION_RECORD                ENDS
;-------------------------------------------------------------------------------

STATUS_WAIT_0                   EQU 000000000H
STATUS_ABANDONED_WAIT_0         EQU 000000080H
STATUS_USER_APC                 EQU 0000000C0H
STATUS_TIMEOUT                  EQU 000000102H
STATUS_PENDING                  EQU 000000103H
STATUS_SEGMENT_NOTIFICATION     EQU 040000005H
STATUS_GUARD_PAGE_VIOLATION     EQU 080000001H
STATUS_DATATYPE_MISALIGNMENT    EQU 080000002H   ; exception 17
STATUS_BREAKPOINT               EQU 080000003H   ; exception 3
STATUS_SINGLE_STEP              EQU 080000004H   ; exception 1
STATUS_ACCESS_VIOLATION         EQU 0C0000005H   ; typically exception 13
STATUS_IN_PAGE_ERROR            EQU 0C0000006H
STATUS_NO_MEMORY                EQU 0C0000017H
STATUS_ILLEGAL_INSTRUCTION      EQU 0C000001DH
STATUS_NONCONTINUABLE_EXCEPTION EQU 0C0000025H
STATUS_INVALID_DISPOSITION      EQU 0C0000026H
STATUS_ARRAY_BOUNDS_EXCEEDED    EQU 0C000008CH   ; exception 5
STATUS_FLOAT_DENORMAL_OPERAND   EQU 0C000008DH
STATUS_FLOAT_DIVIDE_BY_ZERO     EQU 0C000008EH
STATUS_FLOAT_INEXACT_RESULT     EQU 0C000008FH
STATUS_FLOAT_INVALID_OPERATION  EQU 0C0000090H
STATUS_FLOAT_OVERFLOW           EQU 0C0000091H
STATUS_FLOAT_STACK_CHECK        EQU 0C0000092H
STATUS_FLOAT_UNDERFLOW          EQU 0C0000093H
STATUS_INTEGER_DIVIDE_BY_ZERO   EQU 0C0000094H   ; exception 0
STATUS_INTEGER_OVERFLOW         EQU 0C0000095H   ; exception 4
STATUS_PRIVILEGED_INSTRUCTION   EQU 0C0000096H   ; typically exception 13
STATUS_STACK_OVERFLOW           EQU 0C00000FDH
STATUS_CONTROL_C_EXIT           EQU 0C000013AH

Exception_Handler_Struc
include vmm.inc

Exception_Handler_Struc struc
EH_Reserved     dd      ?
EH_Start_EIP    dd      ?
EH_End_EIP      dd      ?
EH_Handler      dd      ?
Exception_Handler_Struc ends
```

```
The Exception_Handler_Struc structure contains information about a ring-0 exception handler.
EH_Reserved

Reserved; must be zero.
EH_Start_EIP
Specifies the starting address of the exception handler's supported range of addresses.
EH_End_EIP
Specifies the ending address of the exception handler's supported range of addresses.
EH_Handler
Points to the exception handler.

See also Install_Exception_Handler

include winbase.h,winnt.h

#define EXCEPTION_ACCESS_VIOLATION          ((DWORD   )0xC0000005L)
#define EXCEPTION_DATATYPE_MISALIGNMENT     ((DWORD   )0x80000002L)
#define EXCEPTION_BREAKPOINT                ((DWORD   )0x80000003L)
#define EXCEPTION_SINGLE_STEP               ((DWORD   )0x80000004L)
#define EXCEPTION_ARRAY_BOUNDS_EXCEEDED     ((DWORD   )0xC000008CL)
#define EXCEPTION_FLT_DENORMAL_OPERAND      ((DWORD   )0xC000008DL)
#define EXCEPTION_FLT_DIVIDE_BY_ZERO        ((DWORD   )0xC000008EL)
#define EXCEPTION_FLT_INEXACT_RESULT        ((DWORD   )0xC000008FL)
#define EXCEPTION_FLT_INVALID_OPERATION     ((DWORD   )0xC0000090L)
#define EXCEPTION_FLT_OVERFLOW              ((DWORD   )0xC0000091L)
#define EXCEPTION_FLT_STACK_CHECK           ((DWORD   )0xC0000092L)
#define EXCEPTION_FLT_UNDERFLOW             ((DWORD   )0xC0000093L)
#define EXCEPTION_INT_DIVIDE_BY_ZERO        ((DWORD   )0xC0000094L)
#define EXCEPTION_INT_OVERFLOW              ((DWORD   )0xC0000095L)
#define EXCEPTION_PRIV_INSTRUCTION          ((DWORD   )0xC0000096L)
#define EXCEPTION_IN_PAGE_ERROR             ((DWORD   )0xC0000006L)
#define EXCEPTION_ILLEGAL_INSTRUCTION       ((DWORD   )0xC000001DL)
#define EXCEPTION_NONCONTINUABLE_EXCEPTION  ((DWORD   )0xC0000025L)
#define EXCEPTION_STACK_OVERFLOW            ((DWORD   )0xC00000FDL)
#define EXCEPTION_INVALID_DISPOSITION       ((DWORD   )0xC0000026L)
#define EXCEPTION_GUARD_PAGE                ((DWORD   )0x80000001L)


set up Structured Exception Handler( in Ring3 ):

        push    offset32 my_Exception_Handler   ; address of our handler
        push    dword [fs:00000000]             ; save previous head of the list
        mov     [fs:00000000], esp              ; prepend our handler to the list
.....
        pop     dword [fs:00000000]
        pop     eax                             ;or pop anything else

        ...

my_Exception_Handler:
        push    ebp                     ; our handler, let's set up the stack
        mov     ebp, esp                ; frame cos we're gonna access

Expt_Status     =       [ebp+4+4]
Context         =       [ebp+4+0ch]

        mov     ebx, Expt_Status        ; check whether it was an
        cmp     dword [ebx], 0x80000004 ; EXCEPTION_SINGLE_STEP or not
        jne     ????
                        184=128+56=80h+38h=b8h
        mov     eax, Context            ;
        mov     ebx, [eax+184]          ; Eip_in_Context
        cmp     byte [ebx], 0x9D        ;
        jne     .a                      ;
        mov     ecx, [eax+196]          ; EFlags_in_Context
        test    [ecx], word 0x100       ; TF ?
        jz      .b                      ;
```

```
        mov     esp, ebp
        pop     ebp
        ret


-----
0040BC4D  PUSH      EBP
0040BC4E  MOV       EBP,ESP
0040BC50  SUB       ESP,18
0040BC53  MOV       EAX,[EBP+10]                        ;context
0040BC56  MOV       ECX,[EAX+000000B8]                  ;eip_in_context
0040BC5C  MOV       [EBP-4],ECX
0040BC5F  MOV       EDX,[EBP+08]                        ;Expt_Status
0040BC62  MOV       EAX,[EDX]
0040BC64  MOV       [EBP-18],EAX
0040BC67  CMP       [DWORD EBP-18],80000003             ;Expt_Brkpnt
0040BC6E  JE        0040BDC1
0040BC74  CMP       [DWORD EBP-18],80000004             ;Expt_Single_Step
0040BC7B  JE        0040BDE1
0040BC81  CMP       [DWORD EBP-18],C0000005             ;Access_Violation
0040BC88  JE        0040BC8F
0040BC8A  JMP       0040BE01

0040BC8F  MOV       ECX,[EBP+10]
0040BC92  CMP       [DWORD ECX+000000B0],00400000
0040BC9C  JNE       0040BD1B
0040BC9E  MOV       EDX,[EBP+10]
0040BCA1  MOV       [DWORD EDX],00010017
0040BCA7  MOV       EAX,[EBP+10]
0040BCAA  MOV       [DWORD EAX+18],00002355
0040BCB1  MOV       ECX,[0040C270]
0040BCB7  MOV       [EBP-8],ECX
0040BCBA  MOV       EDX,[EBP-8]
0040BCBD  SUB       EDX,01
0040BCC0  MOV       [EBP-8],EDX
0040BCC3  MOV       EAX,[EBP+10]
0040BCC6  MOV       ECX,[EBP-8]
0040BCC9  MOV       [EAX+04],ECX
0040BCCC  MOV       EDX,[EBP-8]
0040BCCF  SUB       EDX,02
0040BCD2  MOV       [EBP+F0],EDX
0040BCD5  MOV       EAX,[EBP+10]
0040BCD8  MOV       ECX,[EBP+F0]
0040BCDB  MOV       [EAX+08],ECX
0040BCDE  MOV       EDX,[EBP-8]
0040BCE1  MOV       [EBP+EC],EDX
0040BCE4  MOV       EAX,[EBP+10]
0040BCE7  MOV       ECX,[EBP+EC]
0040BCEA  MOV       [EAX+0C],ECX
0040BCED  MOV       EDX,[EBP+F0]
0040BCF0  MOV       [EBP+F4],EDX
0040BCF3  MOV       EAX,[EBP+10]
0040BCF6  MOV       ECX,[EBP+F4]
0040BCF9  MOV       [EAX+10],ECX
0040BCFC  MOV       EDX,[EBP+10]
0040BCFF  MOV       [DWORD EDX+000000A4],BFF70000
0040BD09  MOV       EAX,[EBP+10]
0040BD0C  MOV       [DWORD EAX+000000B0],BFF70000
0040BD16  JMP       0040BDBF
0040BD1B  MOV       ECX,[EBP+10]
0040BD1E  CMP       [DWORD ECX+000000B0],BFF70000
0040BD28  JNE       0040BD79
0040BD2A  MOV       EDX,[EBP+10]
0040BD2D  MOV       [DWORD EDX],00010007
0040BD33  MOV       EAX,[EBP+10]
0040BD36  MOV       [DWORD EAX+000000A4],77F00000
```

```
0040BD40  MOV       ECX,[EBP+10]
0040BD43  MOV       [DWORD ECX+000000B0],77F00000
0040BD4D  MOV       EDX,[EBP+10]
0040BD50  MOV       EAX,[EDX+000000B8]
0040BD56  ADD       EAX,01
0040BD59  MOV       ECX,[EBP+10]
0040BD5C  MOV       [ECX+000000B8],EAX
0040BD62  MOV       EDX,[EBP+10]
0040BD65  MOV       EAX,[EDX+000000B8]
0040BD6B  ADD       EAX,01
0040BD6E  MOV       ECX,[EBP+10]
0040BD71  MOV       [ECX+000000B8],EAX
0040BD77  JMP       0040BDBF
0040BD79  MOV       EDX,[EBP+10]
0040BD7C  CMP       [DWORD EDX+000000B0],00
0040BD83  JE        0040BDB8
0040BD85  MOV       EAX,[EBP+10]
0040BD88  CMP       [DWORD EAX+000000A4],00
0040BD8F  JNE       0040BDB8
0040BD91  MOV       ECX,[EBP+10]
0040BD94  MOV       [DWORD ECX],00010017
0040BD9A  MOV       EDX,[EBP+10]
0040BD9D  MOV       EAX,[EBP+10]
0040BDA0  MOV       ECX,[EAX+000000B0]
0040BDA6  MOV       [EDX+000000A4],ECX
0040BDAC  MOV       EDX,[EBP+10]
0040BDAF  MOV       [DWORD EDX+18],00000000
0040BDB6  JMP       0040BDBF
0040BDB8  MOV       EAX,00000001
0040BDBD  JMP       0040BE0A
0040BDBF  JMP       0040BE08
@@1:
0040BDC1  MOV       EAX,[EBP+10]
0040BDC4  MOV       [DWORD EAX],00010007
0040BDCA  MOV       ECX,[EBP+10]
0040BDCD  MOV       EDX,[ECX+000000B8]
0040BDD3  ADD       EDX,01
0040BDD6  MOV       EAX,[EBP+10]
0040BDD9  MOV       [EAX+000000B8],EDX
0040BDDF  JMP       0040BE08

0040BDE1  MOV       ECX,[EBP+10]
0040BDE4  MOV       [DWORD ECX],00010007
0040BDEA  MOV       EDX,[EBP+10]
0040BDED  MOV       EAX,[EDX+000000B8]
0040BDF3  ADD       EAX,01
0040BDF6  MOV       ECX,[EBP+10]
0040BDF9  MOV       [ECX+000000B8],EAX
0040BDFF  JMP       0040BE08

0040BE01  MOV       EAX,00000001
0040BE06  JMP       0040BE0A

0040BE08  XOR       EAX,EAX

0040BE0A  MOV       ESP,EBP
0040BE0C  POP       EBP
0040BE0D  RET       0010
```

```
crst          struc      ;Critical_Section
Type             dd ?             //00    = 4
Count            dd ?             //04
Owner_R3Thread   dd ?             //08
mc               dd ?             //0c
LockSemaphore    dd ?             //10 0 or 1
m14              dd ?
m18              dw ?
                 db ? ;
                 db ? ;
m1c              dd ?
pConsole         dd ?             //20 point to K32obj 9
          ends

see also:
         PDB     offset 60h       goo
```

PDB

STARTUPINFO
include winbase.h

```
typedef struct _STARTUPINFOA {
    DWORD   cb;
    LPSTR   lpReserved;
    LPSTR   lpDesktop;
    LPSTR   lpTitle;
    DWORD   dwX;
    DWORD   dwY;
    DWORD   dwXSize;
    DWORD   dwYSize;
    DWORD   dwXCountChars;
    DWORD   dwYCountChars;
    DWORD   dwFillAttribute;
    DWORD   dwFlags;
    WORD    wShowWindow;
    WORD    cbReserved2;
    LPBYTE  lpReserved2;
    HANDLE  hStdInput;
    HANDLE  hStdOutput;
    HANDLE  hStdError;
} STARTUPINFOA, *LPSTARTUPINFOA;
```

MODREF Module reference

```
typedef struct _MODREF
{
    PMODREF         pNextModRef;    //00h end with 0
    DWORD           un1;            //04h number of ?
    DWORD           un2;            //08h Ring0TCB ?
    DWORD           un3;            //0Ch
    WORD            mteIndex;       //10h
    WORD            un4;            //12h
    DWORD           un5;            //14h
    PVOID           ppdb;           //18h Pointer to process database
    DWORD           un6;            //1Ch
    DWORD           un7;            //20h
    DWORD           un8;            //24h
} MODREF, *PMODREF;
```

IMTE Internal Module Table Entry

```
typedef struct _IMTE
{
    DWORD           un1;            //00h
    PIMAGE_NT_HEADERS   pNTHdr;     //04h
```

```
    DWORD           un2;            //08h
    PSTR            pszFileName;    //0Ch
    PSTR            pszModName;     //10h
    WORD            cbFileName;     //14h
    WORD            cbModName;      //16h
    DWORD           un3;            //18h
    DWORD           cSections;      //1Ch
    DWORD           un5;            //20h
    DWORD           baseAddress;    //24h
    WORD            hModule16;      //28h goo
    WORD            cUsage;         //2Ah
    DWORD           offset?;        //2Ch ??? ptr struc1,and struc1[18h]->PDB
                                    //    this is _MODREF ?
    PSTR            pszFileName2;   //30h
    WORD            cbFileName2;    //34h
    DWORD           pszModName2;    //36h
    WORD            cbModName2;     //3Ah
} IMTE, *PIMTE;
```

see also:
PDB offset 2a goo

EDB Environment DataBase
W95SPS p87

```
typedef struct _ENVIRONMENT_DATABASE    //why not name EDB
{
PSTR    pszEnvironment;     //00h Pointer to Environment
DWORD   un1;                //04h
PSTR    pszCmdLine;         //08h Pointer to command line
PSTR    pszCurrDirectory;   //0Ch Pointer to current directory
LPSTARTUPINFOA pStartupInfo;//10h Pointer to STARTUPINFOA struct go
HANDLE  hStdIn;             //14h Standard Input
HANDLE  hStdOut;            //18h Standard Output
HANDLE  hStdErr;            //1Ch Standard Error
DWORD   un2;                //20h
DWORD   InheritConsole;     //24h
DWORD   BreakType;          //28h
DWORD   BreakSem;           //2Ch
DWORD   BreakEvent;         //30h
DWORD   BreakThreadID;      //34h
DWORD   BreakHandlers;      //38h
} ENVIRONMENT_DATABASE, *PENVIRONMENT_DATABASE;
```

```
typedef struct _HANDLE_TABLE_ENTRY
{
    DWORD   flags;      // Valid flags depend on what type of object this is
    PVOID   pObject;    // Pointer to the object that the handle refers to
} HANDLE_TABLE_ENTRY, *PHANDLE_TABLE_ENTRY;

typedef struct _HANDLE_TABLE
{
    DWORD   cEntries;               // Max number of handles in table
    HANDLE_TABLE_ENTRY array[1];    // An array (number is given by cEntries)
```

```
} HANDLE_TABLE, *PHANDLE_TABLE;
```

PDB

```
typedef struct _PROCESS_DATABASE        //PDB
{
DWORD   Type;                   //00h KERNEL32 object type (5)
DWORD   cReference;             //04h Number of references to process
DWORD   un1;                    //08h
DWORD   someEvent;              //0Ch An event object (What's it used for???)
DWORD   TerminationStatus;      //10h Returned by GetExitCodeProcess
DWORD   un2;                    //14h
DWORD   DefaultHeap;            //18h Address of the process heap
DWORD   MemoryContext;          //1Ch pointer to the process's context goo
DWORD   flags;                  //20h go
DWORD   pPSP;                   //24h Linear address of PSP?
WORD    PSPSelector;            //28h
WORD    MTEIndex;               //2Ah *4+ModuleList=IMTE goo
WORD    cThreads;               //2Ch
WORD    cNotTermThreads;        //2Eh
WORD    un3;                    //30h
WORD    cRing0Threads;          //32h number of ring 0 threads
HANDLE  HeapHandle;             //34h Heap to allocate handle tables out of
                                //    This seems to always be the KERNEL32 heap
HTASK   W16TDB;                 //38h Win16 Task Database selector goo
DWORD   MemMapFiles;            //3Ch memory mapped file list (?)
PEDB    pEDB;                   //40h Pointer to Environment Database go
PHANDLE_TABLE pHandleTable;     //44h Pointer to process handle table
PPDB    ParentPDB;              //48h Parent process database
PMODREF MODREFlist;             //4Ch Module reference list go
DWORD   ThreadList;             //50h Threads in this process
DWORD   DebuggeeCB;             //54h Debuggee Context block?
DWORD   LocalHeapFreeHead;      //58h Head of free list in process heap
DWORD   InitialRing0ID;         //5Ch
CRITICAL_SECTION    crst;       //60h defined in winnt.h goo 24h len
                                //    yes, this structure is 24h len
DWORD   pConsole;               //84h Pointer to console for process
DWORD   tlsInUseBits1;          //88h  // Represents TLS indices 0 - 31
DWORD   tlsInUseBits2;          //8Ch  // Represents TLS indices 32 - 63
DWORD   ProcessDWORD;           //90h
PPDB    ProcessGroup;           //94h
DWORD   pExeMODREF;             //98h pointer to EXE's MODREF
DWORD   TopExcFilter;           //9Ch Top Exception Filter?
DWORD   BasePriority;           //A0h Base scheduling priority for process
DWORD   HeapOwnList;            //A4h Head of the list of process heaps
DWORD   HeapHandleBlockList;//A8h Pointer to head of heap handle block list
DWORD   pSomeHeapPtr;           //ACh normally zero, but can a pointer to a
                                //moveable handle block in the heap
DWORD   pConsoleProvider;       //B0h Process that owns the console we're using?
WORD    EnvironSelector;        //B4h Selector containing process environment
WORD    ErrorMode;              //B6H SetErrorMode value (also thunks to Win16)
DWORD   pevtLoadFinished;       //B8h Pointer to event LoadFinished?
WORD    UTState;                //BCh
} *PPDB, PROCESS_DATABASE, *PPROCESS_DATABASE;
```

```
        Pid     =       Pdb ^ Obfuscator

see also:
        R3TCB   offset 8        goo
        GetCurrentProcessId  returns PDB xor Unobsfucator
        VWIN32_GetCurrentProcessHandle  returns PDB in Ring0


PDB_Flags:
        { 0x00000001, "fDebugSingle" },
        { 0x00000002, "fCreateProcessEvent" },
        { 0x00000004, "fExitProcessEvent" },
        { 0x00000008, "fWin16Process" },
        { 0x00000010, "fDosProcess" },
        { 0x00000020, "fConsoleProcess" },
        { 0x00000040, "fFileApisAreOem" },
        { 0x00000080, "fNukeProcess" },
        { 0x00000100, "fServiceProcess" },
        { 0x00000800, "fLoginScriptHack" },
        { 0x00200000, "fSendDLLNotifications" },
        { 0x00400000, "fDebugEventPending" },
        { 0x00800000, "fNearlyTerminating" },
        { 0x08000000, "fFaulted" },
        { 0x10000000, "fTerminating" },
        { 0x20000000, "fTerminated" },
        { 0x40000000, "fInitError" },
        { 0x80000000, "fSignaled" },
```

R3TCB TDBX TIB

```
//==================================
// WIN32WLK - Matt Pietrek 1995
//==================================
typedef struct _SEH_record
{
    struct _SEH_record *pNext;
    FARPROC             pfnHandler;
} SEH_record, *PSEH_record;
```

TIB Thread Information Block
This structure is always at offset 10h of R3TCB.
In runtime of PE, FS: points to this structure.
so you can get Ring3TCB by

```
                        Ring3TCB = GetLinearAddress(FS)-10h
```

or

```
                        Ring3TCB = (WORD)FS:[18h] - 10h
```

```
// This is semi-documented in the NTDDK.H file from the NT DDK
typedef struct _TIB
{
PSEH_record pvExcept;       //00h Head of exception record list
PVOID   pvStackUserTop;     //04h Top of user stack
PVOID   pvStackUserBase;    //08h Base of user stack
WORD    pW16TDB;            //0Ch W16 Task DataBase go
WORD    pvThunksSS;         //0Eh SS selector used for thunking to 16 bits
DWORD   SelmanList;         //10h
PVOID   pvArbitrary;        //14h Available for application use
PTIB    ptibSelf;           //18h Linear address of TIB structure, = R3TCB + 10h
WORD    TIBFlags;           //1Ch
WORD    Win16MutexCount;    //1Eh
DWORD   DebugContext;       //20h
DWORD   pCurrentPriority;   //24h
DWORD   pvQueue;            //28h Message Queue selector
PVOID*  pvTLSArray;         //2Ch Thread Local Storage array
} TIB, *PTIB;
```

TDBX

```
typedef struct _TDBX
{
    DWORD   ptdb;               //00h  PTHREAD_DATABASE R3TCB
                                //04    for vmm4.03
    DWORD   ppdb;               //04h  PPROCESDS_DATABASE PDB
    DWORD   ContextHandle;      //08h  CONTEXT goo
    DWORD   un1;                //0Ch
    DWORD   TimeOutHandle;      //10h
    DWORD   WakeParam;          //14h
    DWORD   BlockHandle;        //18h
    DWORD   BlockState;         //1Ch
    DWORD   SuspendCount;       //20h
    DWORD   SuspendHandle;      //24h
    DWORD   MustCompleteCount;  //28h
```

```
    DWORD   WaitExFlags;        //2Ch go
    DWORD   SyncWaitCount;      //30h
    DWORD   QueuedSyncFuncs;    //34h
    DWORD   UserAPCList;        //38h
    DWORD   KernAPCList;        //3Ch
    DWORD   pPMPSPSelector;     //40h
    DWORD   BlockedOnID;        //44h
    DWORD   un2[7];             //48h
    DWORD   TraceRefData;       //64h
    DWORD   TraceCallBack;      //68h
    DWORD   TraceEventHandle;   //6Ch
    WORD    TraceOutLastCS;     //70h
    WORD    K16TDB;             //72h goo
    WORD    K16PDB;             //74h This 16bit sel ptr to PSP!
    WORD    DosPDBSeg;          //76h
    WORD    ExceptionCount;     //78h
                                //289816h ???
                                //569595h ???
} TDBX, *PTDBX;
```

How to get TDBX from Ring0TCB ?
I think I find it:
TDBX = Ring0TCB[8c] !!!
see also:
R3TCB offset 60h goo
R0TCB offset 8ch goo

TDBX_flags

```
                                // 0x00000001 - WAITEXBIT
                                // 0x00000002 - WAITACKBIT
                                // 0x00000004 - SUSPEND_APC_PENDING
                                // 0x00000008 - SUSPEND_TERMINATED
                                // 0x00000010 - BLOCKED_FOR_TERMINATION
                                // 0x00000020 - EMULATE_NPX
                                // 0x00000040 - WIN32_NPX
                                // 0x00000080 - EXTENDED_HANDLES
                                // 0x00000100 - FROZEN
                                // 0x00000200 - DONT_FREEZE
                                // 0x00000400 - DONT_UNFREEZE
                                // 0x00000800 - DONT_TRACE
                                // 0x00001000 - STOP_TRACING
                                // 0x00002000 - WAITING_FOR_CRST_SAFE
                                // 0x00004000 - CRST_SAFE
                                // 0x00040000 - BLOCK_TERMINATE_APC
```

R3TCB
This is Ring3 Thread Control Block

```
typedef struct _THREAD_DATABASE
{
DWORD   Type;               //00h = 6
DWORD   cReference;         //04h
PPROCESS_DATABASE pProcess; //08h PDB goo
DWORD   someEvent;          //0Ch An event object (What's it used for???)

DWORD   pvExcept;           //10h This field through field 3CH is a TIB
                            //     structure (see TIB.H) go
DWORD   TopOfStack;         //14h
```

```
DWORD   StackLow;           //18h
WORD    W16TDB;             //1Ch goo
WORD    StackSelector16;    //1Eh Used when thunking down to 16 bits
DWORD   SelmanList;         //20h  ptr to PROCESSENTRY32 struc go
DWORD   UserPointer;        //24h
PTIB    pTIB;               //28h  go
WORD    TIBFlags;           //2Ch  TIBF_WIN32 = 1, TIBF_TRAP = 2
WORD    Win16MutexCount;    //2Eh
DWORD   DebugContext;       //30h
PDWORD  pCurrentPriority;   //34h
DWORD   MessageQueue;       //38h go
DWORD   pTLSArray;          //3Ch

PPROCESS_DATABASE pProcess2;//40h Another copy of the thread's process???go
DWORD   Flags;              //44h go
DWORD   TerminationStatus;  //48h Returned by GetExitCodeThread
WORD    TIBSelector;        //4Ch
WORD    EmulatorSelector;   //4Eh
DWORD   cHandles;           //50h
DWORD   WaitNodeList;       //54h
DWORD   un4;                //58h
DWORD   Ring0Thread;        //5Ch  goo
PTDBX   pTDBX;              //60h  goo
DWORD   StackBase;          //64h
DWORD   TerminationStack;   //68h
DWORD   EmulatorData;       //6Ch
DWORD   GetLastErrorCode;   //70h
DWORD   DebuggerCB;         //74h
DWORD   DebuggerThread;     //78h
PCONTEXT    ThreadContext;  //7Ch  go
DWORD   Except16List;       //80h
DWORD   ThunkConnect;       //84h
DWORD   NegStackBase;       //88h
DWORD   CurrentSS;          //8Ch
DWORD   SSTable;            //90h
DWORD   ThunkSS16;          //94h
DWORD   TLSArray[64];       //98h
DWORD   DeltaPriority;      //198h
// The retail version breaks off somewhere around here.
// All the remaining fields are most likely only in the debug version
DWORD   un5[7];             //19Ch
DWORD   pCreateData16;      //1B8h
DWORD   APISuspendCount;    //1BCh # of times SuspendThread has been called
DWORD   un6;                //1C0h
DWORD   WOWChain;           //1C4h
WORD    wSSBig;             //1C8h
WORD    un7;                //1CAh
DWORD   lp16SwitchRec;      //1CCh
DWORD   un8[2];             //1D0h
DWORD   Mutex?[4];                      //1D8h max 4 level

DWORD   hMutex[4];          //1E8h max 4 level,hMutex of each level

DWORD   un9;                //1F8h
DWORD   ripString;          //1FCh
DWORD   LastTlsSetValueEIP[64]; // 200h (parallel to TlsArray, contains EIP
                                //      where TLS value was last set from)

} THCB, THREAD_DATABASE, *PTHREAD_DATABASE;
```

for VMMversion >= 403h, offset of PDB will be 38h, not 8 ?

see also:
W16TDB offset 54h go
TDBX offset 00 goo

R3TCB_flags

```
                            // 0x00000001 - fCreateThreadEvent
                            // 0x00000002 - fCancelExceptionAbort
                            // 0x00000004 - fOnTempStack
                            // 0x00000008 - fGrowableStack
                            // 0x00000010 - fDelaySingleStep
                            // 0x00000020 - fOpenExeAsImmovableFile
                            // 0x00000040 - fCreateSuspended
                            // 0x00000080 - fStackOverflow
                            // 0x00000100 - fNestedCleanAPCs
                            // 0x00000200 - fWasOemNowAnsi
                            // 0x00000400 - fOKToSetThreadOem
```

About Virtual Machine (VM)

- size of CB_High_Linear is 10fff0h

```
include:        vmm.h   vmm.inc

typedef struct cb_s {
//      DWORD VM_count?                 //-70
//      DWORD VM_execpriority           //-68
//      DWORD VM_List_of_?              //-44
//      QWORD VM_savIDT                 //-36
//      DWORD VM_hTimeOut               //-2c Time_Slice_Wake_Sys_VM
//      DWORD VM_Ring0TCB               //-24
//      DWORD VM_TCB1                   //-20 Thread of ?
//      DWORD VM_TCB                    //-1c Thread of VM


    DWORD CB_VM_Status;                 //00 VM status flags */
    DWORD CB_High_Linear;               //04 Address of VM mapped high */
    DWORD CB_Client_Pointer;            //08
    DWORD CB_VMID;                      //0c
    DWORD CB_Signature;                 //10 'VMcb'
//      DWORD VM_Exec_Time;             //14
//      WORD  VM_ldt;                   //1a
//      WORD  VM_Event_List_Count       //1c
//      WORD  VM_LDT_sel;               //1e
//      DWORD VM_Event_List             //24
//      DWORD VM_MemBlock;              //28
//      DWORD VM_Next_VMHandle;         //2c

debug:
//      DWORD VM_hheapVMCB              //-74
//      BYTE  VM_fIntsStatus            //-6c
//      DWORD VM_App_Num;               //-6a
//      DWORD VM_PM_APP_CB;             //-40
//      WORD  VM_LDTcpq;                //-38
//      QWORD VM_PM_Idt                 //-36
//      DWORD VM_pthcbThread            //-24 Ring0TCB
//      DWORD VM_ThreadListHead         //-20 see Get_Next_Thread_Handle
                                        //    the first R0TCB
//      DWORD VM_SuspendCount           //-18
//      DWORD VM_ppteVM                 //-14
//      DWORD VM_pdeVM                  //-10
//      DWORD VM_Int_Enable_Count;      //-08
//      DWORD VM_EnableFirstMB          //14
//      DWORD VM_Exec_Time;             //18
//      DWORD VM_TS_Sched_Count         //1c
//      WORD  VM_LDTsel;                //1e
//      DWORD VM_MMGR_Flags;            //28
//      DWORD VM_LDTAddr;               //2c
//      DWORD VM_List_Linke;            //30 see Get_Next_Thread_Handle,Get_Next_VM_Handle
                                        //   next VM
//      DWORD VM_Inst_Buf_ptr           //38
//      DWORD VM_hVM                    //3c
//      DWORD VM_quota_left             //40
//      DWORD VM_TrapMask               //58 Enable_Local_Trap

debug:
```

```
        DWORD VM_hPage?         //-3c

}*VMHANDLE, **PVMHANDLE;


#define VMCB_ID 0x62634D56              /* VMcb */

/*
 *  VM status indicates globally interesting VM states
 */

#define VMSTAT_EXCLUSIVE_BIT    0x00    /* VM is exclusive mode */
#define VMSTAT_EXCLUSIVE        (1L << VMSTAT_EXCLUSIVE_BIT)
#define VMSTAT_BACKGROUND_BIT   0x01    /* VM runs in background */
#define VMSTAT_BACKGROUND       (1L << VMSTAT_BACKGROUND_BIT)
#define VMSTAT_CREATING_BIT     0x02    /* In process of creating */
#define VMSTAT_CREATING         (1L << VMSTAT_CREATING_BIT)
#define VMSTAT_SUSPENDED_BIT    0x03    /* VM not scheduled */
#define VMSTAT_SUSPENDED        (1L << VMSTAT_SUSPENDED_BIT)
#define VMSTAT_NOT_EXECUTEABLE_BIT 0x04 /* VM partially destroyed */
#define VMSTAT_NOT_EXECUTEABLE  (1L << VMSTAT_NOT_EXECUTEABLE_BIT)
#define VMSTAT_PM_EXEC_BIT      0x05    /* Currently in PM app */
#define VMSTAT_PM_EXEC          (1L << VMSTAT_PM_EXEC_BIT)
#define VMSTAT_PM_APP_BIT       0x06    /* PM app present in VM */
#define VMSTAT_PM_APP           (1L << VMSTAT_PM_APP_BIT)
#define VMSTAT_PM_USE32_BIT     0x07    /* PM app is 32-bit */
#define VMSTAT_PM_USE32         (1L << VMSTAT_PM_USE32_BIT)
#define VMSTAT_VXD_EXEC_BIT     0x08    /* Call from VxD */
#define VMSTAT_VXD_EXEC         (1L << VMSTAT_VXD_EXEC_BIT)
#define VMSTAT_HIGH_PRI_BACK_BIT 0x09   /* High pri background */
#define VMSTAT_HIGH_PRI_BACK    (1L << VMSTAT_HIGH_PRI_BACK_BIT)
#define VMSTAT_BLOCKED_BIT      0x0A    /* Blocked on semaphore */
#define VMSTAT_BLOCKED          (1L << VMSTAT_BLOCKED_BIT)
#define VMSTAT_AWAKENING_BIT    0x0B    /* Woke up after blocked */
#define VMSTAT_AWAKENING        (1L << VMSTAT_AWAKENING_BIT)
#define VMSTAT_PAGEABLEV86BIT   0x0C    /* part of V86 is pageable (PM app) */
#define VMSTAT_PAGEABLEV86_BIT  VMSTAT_PAGEABLEV86BIT
#define VMSTAT_PAGEABLEV86      (1L << VMSTAT_PAGEABLEV86BIT)
#define VMSTAT_V86INTSLOCKEDBIT 0x0D    /* Locked regardless of pager type */
#define VMSTAT_V86INTSLOCKED_BIT VMSTAT_V86INTSLOCKEDBIT
#define VMSTAT_V86INTSLOCKED    (1L << VMSTAT_V86INTSLOCKEDBIT)
#define VMSTAT_IDLE_TIMEOUT_BIT 0x0E    /* Scheduled by time-slicer */
#define VMSTAT_IDLE_TIMEOUT     (1L << VMSTAT_IDLE_TIMEOUT_BIT)
#define VMSTAT_IDLE_BIT         0x0F    /* VM has released time slice */
#define VMSTAT_IDLE             (1L << VMSTAT_IDLE_BIT)
#define VMSTAT_CLOSING_BIT      0x10    /* Close_VM called for VM */
#define VMSTAT_CLOSING          (1L << VMSTAT_CLOSING_BIT)
#define VMSTAT_TS_SUSPENDED_BIT 0x11    /* VM suspended by */
#define VMSTAT_TS_SUSPENDED     (1L << VMSTAT_TS_SUSPENDED_BIT)
#define VMSTAT_TS_MAXPRI_BIT    0x12    /* this is fgd_pri 10,000 internally*/
#define VMSTAT_TS_MAXPRI        (1L << VMSTAT_TS_MAXPRI_BIT)

#define VMSTAT_USE32_MASK       (VMSTAT_PM_USE32 | VMSTAT_VXD_EXEC)

VMSTAT_EXCLUSIVE              1         'Excl    ',0
VMSTAT_BACKGROUND             2         'Back    ',0
VMSTAT_CREATING               4         'Creat   ',0
VMSTAT_SUSPENDED              8         'Susp    ',0
VMSTAT_NOT_EXECUTEABLE  0x00010         'Not_Ex  ',0
VMSTAT_PM_EXEC          0x00020         'PM_Exe  ',0
VMSTAT_PM_APP           0x00040         'PM_App  ',0
VMSTAT_PM_USE32         0x00080         'PM_U32  ',0
VMSTAT_VXD_EXEC         0x00100         'VxD_Ex  ',0
VMSTAT_HIGH_PRI_BACK    0x00200         'HPriBk  ',0
VMSTAT_BLOCKED          0x00400         'Block   ',0
VMSTAT_AWAKENING        0x00800         'Awake   ',0
```

```
VMSTAT_PAGEABLEV86      0x01000         'PgV86   ',0
VMSTAT_V86INTSLOCKED    0x02000         'V86ILk  ',0
VMSTAT_IDLE_TIMEOUT     0x04000         'IdleTOut',0
VMSTAT_IDLE             0x08000         'Idle    ',0
VMSTAT_CLOSING          0x10000         'Closing ',0
VMSTAT_TS_SUSPENDED     0x20000
VMSTAT_TS_MAXPRI        0x40000

VMSTAT_SWAPPING         0x2000000


see also:
        Ring0TCB        offset 14h      goo
        VMMcall Suspend_VM
        VMMcall Resume_VM
        Get_Initial_Thread_Handle       go
        Int 2f 1683h    go

include shell.inc

SGVMI_Windowed          EQU     00000004H
SGVMI_ALTTABdis         EQU     00000020H
SGVMI_ALTESCdis         EQU     00000040H
SGVMI_ALTSPACEdis       EQU     00000080H
SGVMI_ALTENTERdis       EQU     00000100H
SGVMI_ALTPRTSCdis       EQU     00000200H
SGVMI_PRTSCdis          EQU     00000400H
SGVMI_CTRLESCdis        EQU     00000800H
SGVMI_HasHotKey         EQU     00004000H
SGVMI_Polling           EQU     00001000H
SGVMI_FastPaste         EQU     00020000H
SGVMI_NoHMA             EQU     00002000H
SGVMI_XMS_Lock          EQU     00010000H
SGVMI_EMS_Lock          EQU     00008000H
SGVMI_V86_Lock          EQU     00040000H
SGVMI_ClsExit           EQU     40000000H

SGVMI_Windowed          The virtual machine runs in a window.
SGVMI_ALTTABdis         ALT+TAB is reserved.
SGVMI_ALTESCdis         ALT+ESC is reserved.
SGVMI_ALTSPACEdis       ALT+SPACEBAR is reserved.
SGVMI_ALTENTERdis       ALT+ENTER is reserved.
SGVMI_ALTPRTSCdis       ALT+PRTSC is reserved.
SGVMI_PRTSCdis          PRTSC is reserved.
SGVMI_CTRLESCdis        CTRL+ESC is reserved.
SGVMI_HasHotKey         Has a shortcut key.
SGVMI_Polling           Polling detection enabled.
SGVMI_FastPaste         Allow fast paste enabled.
SGVMI_NoHMA             No HMA.
SGVMI_XMS_Lock          XMS hands locked.
SGVMI_EMS_Lock          EMS hands locked.
SGVMI_V86_Lock          V86 memory locked.
SGVMI_ClsExit           Close on exit enabled.
```

```
mov     ax, 1683h
int     2Fh


; - Multiplex - MS WINDOWS -  3+ - GET CURRENT VIRTUAL MACHINE ID
; Return: BX = current virtual machine (VM) ID
```

```
//==================================
// SHOW16 - Matt Pietrek 1995
// FILE: TDB.H
//==================================

        Win16 Task Data Block

typedef struct TDB              //name: K16TDB
{
    HTASK   TDB_next;           //00 next task in dispatch queue
                                //   end with 0, where is the first ?
    WORD    TDB_taskSP;         //02 Saved SS:SP for this task
    WORD    TDB_taskSS;         //04
    WORD    TDB_nEvents;        //06 Task event counter
    BYTE    TDB_priority;       //08 Task priority (0 is highest)
    BYTE    TDB_thread_ordinal; //09 ordinal number of this thread
    WORD    TDB_thread_next;    //0a next thread
    WORD    TDB_thread_tdb;     //0c the real TDB for this task
    WORD    TDB_thread_list;    //0e list of allocated thread structures
    WORD    TDB_thread_free;    //10 free list of available thread structures
    WORD    TDB_thread_count;   //12 total count of tread structures
    WORD    TDB_FCW;            //14 Floating point control word
    WORD    TDB_flags;          //16 Task flags
    WORD    TDB_ErrMode;        //18 Error mode for this task
    WORD    TDB_ExpWinVer;      //1a Expected Windows version for this task
    WORD    TDB_HInstance;      //1c instance handle of task
    WORD    TDB_HMODULE;        //1e module database for task
                                //   selector ptr to 'NE'
                                //   Mod16 go
    WORD    TDB_Queue;          //20 Task Event Queue pointer
    WORD    TDB_Parent;         //22 TDB of the task that started this up
    WORD    TDB_SigAction;      //24 Action for app task signal
    FARPROC TDB_ASignalProc;    //26 App's signal procedure address
    FARPROC TDB_USignalProc;    //2a User's Task Signal procedure address
    FARPROC TDB_GNotifyProc;    //2e Task global discard notify proc.
    FARPROC TDB_INTVECS[7];     //32 Task specfic hardware interrupts
    DWORD   TDB_CompatFlags;    //4e Compatibility flags
    WORD    TDB_FS_selector;    //52 Same selector as FS (points inside TCB)
    DWORD   TDB_ring3_thread_db;//54 32 bit address of KERNEL32 thread database
                                //   Ring3TCB go
    WORD    TDB_thunk_stack_ss; //58 selector used for stack during thunking
    WORD    TDB_filler[3];      //5a appears to be unused
    WORD    TDB_PSP;            //60 MSDOS Process Data Block (aka, the PSP)
    LPBYTE  TDB_DTA;            //62 MSDOS Disk Transfer Address
    BYTE    TDB_Drive;          //66 MSDOS current drive
    char    TDB_Directory[65];  //67 MSDOS current directory
    WORD    TDB_Validity;       //a8 initial AX to be passed to a task
    HTASK   TDB_Yield_to;       //aa DirectedYield arg stored here
    WORD    TDB_LibInitSeg;     //ac segment address of libraries to init
    WORD    TDB_LibInitOff;     //ae MakeProcInstance thunks live here.
    WORD    TDB_MPI_Sel;        //b0 Code selector for thunks
    WORD    TDB_more_thunks;    //b2 selector of segment with more MPI thunks
    WORD    TDB_PT_sig;         //b4 'PT'
    WORD    TDB_unused1;        //b6
    WORD    TDB_next_MPI_thunk; //b8
    char    TDB_MPI_Thunks[0x38];   //ba
    char    TDB_ModName[8];     //f2 Name of Module.
    WORD    TDB_sig;            //fa 'TD' Signature word to detect bogus code
```

```
    DWORD   TDB_unused2;
    char    TDB_current_directory[0x110];   //100h Current directory
}  FAR * LPTDB;
```

```
see also:
        PDB     offset 38h      goo
        THCB    offset 1ch      goo
        TDBX    offset 72h      goo

#define TDB_FLAGS_WIN32     0x0010
#define TDB_FLAGS_WINOLDAP  0x0001

In Win32:

    HANDLE id = GetCurrentThreadId();

    id = TIDToTDB(id);
    // Win16 TDB is at offset 1Ch in Ring 3 THCB
    // I'm a little surprised it's so easy to get at this!
    WORD tdb = *((WORD *) (((BYTE *) id) + 0x1c));
-------------------------------------------------------------
HWND hwnd = GetWindow(GetDesktopWindow(), GW_CHILD);
DWORD   thcb = GetWindowThreadProcessId(hwnd,0);
DWORD    thcb = TIDToTDB(htask);
WORD     tdb16 = *((WORD *) (thcb + 0x1C));  // Win16 TDB -- see CHGDIR.C
-------------------------------------------------------------
void GetProcessNameFromHTask( HTASK hTask, PSTR pszBuffer )
{
    pszBuffer[0] = 0;

    try
    {
        __asm
        {
            push ds
            push ds
            pop  es
            mov ds, word ptr [hTask]
            mov esi, 0F2h
            mov edi, [pszBuffer]
            mov ecx, 2
            cld
            rep movsd
            mov byte ptr es:[edi], 0
            pop ds
        }
    }
    __except( 1 ){}

}
```

```
        Mod16   structure

See also:
        NE overall

//==================================
// SHOW16 - Matt Pietrek 1995
// FILE: HMODULE.H
//==================================
#pragma pack (1)
```

```
typedef struct
{
    WORD    sector_offset;      // Offset to logical sector
    WORD    segment_length;     // Size in bytes of segment
    WORD    flags;              // flags for segment
    WORD    alloc_size;         // Segment allocation size
    WORD    handle;             // Global heap handle assigned by loader
} SEGMENT_RECORD, FAR * LPSEGMENT_RECORD;
```

```
typedef enum
{                               // Segment type constants
  CODE      = 0x0000,           // Code segment type
  DATA      = 0x0001,           // Data segment type
} SEGMENT_TYPES;
```

NE

```
typedef struct NE    //I name it Mod16
{
   WORD     ne_signature;           //00 'NE'
   WORD     ne_usage;               //02 reference count of module
   WORD     ne_npEntryTable;        //04 near pointer to entry table
   HMODULE  ne_npNextExe;           //06 next module database
   WORD     ne_npAutoData;          //08 near pointer to DGROUP segment entry
   WORD     ne_npFileInfo;          //0a near pointer to OFSTRUCT with file name
   WORD     ne_flags;               //0c
   WORD     ne_autodata;            //0e segment index of DGROUP segment
   WORD     ne_heap;                //10
   WORD     ne_stack;               //12
   DWORD    ne_csip;                //14
   DWORD    ne_sssp;                //18
   WORD     ne_cseg;                //1c
   WORD     ne_cModules;            //1e
   WORD     ne_cbNonResNamesTab;    //20
   WORD     ne_segtab;              //22
   WORD     ne_rsrcTab;             //24
   WORD     ne_resNamesTab;         //26 Pascal string of ModuleName
   WORD     ne_modRefTab;           //28
   WORD     ne_importedNamesTab;    //2a
   DWORD    ne_nonResNamesTab;      //2c
   WORD     ne_cMovEnt;             //30
```

```
   WORD     ne_align;               //32
   WORD     ne_cres;                //34
   unsigned char ne_exetyp;         //36
   unsigned char ne_flagsother;     //37
   WORD     ne_importedNamesTab2;   //38
   WORD     ne_importedNamesTab3;   //3a
   WORD     ne_swaparea;            //3c minimum code swap area size
   WORD     ne_expver;              //3e expected windows version num
   DWORD    ne_Win32BaseAddr1;      //40 (Win32 only) Base addr of module
   DWORD    ne_Win32BaseAddr2;      //44 (Win32 only) Base addr of module
   DWORD    ne_Win32ResourceAddr;   //48 (Win32 only) Base addr of resources
//      DWORD ???               //54h
} MODULE, FAR * LPMODULE;
```

see also:

W16TDB offset 1eh go

IMTE offset 28h goo

WORD w=ne:segindex;
WORD b=ne:segtab
WORD sel=ne:[b+(w-1)*10+8]

```
typedef struct
{
    WORD    firstEntry;
    WORD    lastEntry;
    WORD    nextBundle;
} ENTRY_BUNDLE_HEADER, FAR *LPENTRY_BUNDLE_HEADER;
```

```
typedef struct
{
    BYTE    segType;
    BYTE    flags;
    BYTE    segNumber;
    WORD    offset;
} ENTRY, FAR * LPENTRY;
```

```
typedef struct
{
    WORD    ID;
    WORD    count;
    DWORD   function;
} RESOURCE_TYPE, FAR *LPRESOURCE_TYPE;
```

```
typedef struct
```

```
{
    WORD    offset;
    WORD    length;

    WORD    flags;
    WORD    ID;
    WORD    handle;
    WORD    usage;
} RESOURCE_INFO, FAR * LPRESOURCE_INFO;

#define NEAPPTYP        0x0700  // Application type mask
#define NEWINAPI        0x0300  // Uses windowing API
#define NEWINCOMPAT     0x0200  // Compatible with windowing API
#define NENOTWINCOMPAT  0x0100  // Not compatible with windowing API
#define NENONRES        0x0080  // Contains non-resident code segments
#define NELIM32         0x0010  // Uses LIM 3.2 API
#define NEPROT          0x0008  // Runs in protected mode only
#define NEPPLI          0x0004  // Per-Process Library Initialization

// Target operating systems

#define NE_UNKNOWN      0
#define NE_OS2          1   // Microsoft/IBM OS/2
#define NE_WINDOWS      2   // Microsoft Windows
#define NE_DOS4         3   // Microsoft European MS-DOS 4.x
#define NE_DEV386       4   // Microsoft Windows 386

#define MODFLAGS_DLL                0x8000
#define MODFLAGS_CALL_WEP           0x4000
#define MODFLAGS_SELF_LOADING       0x0800
#define MODFLAGS_APPTYPE            0x0300
                                    // 0x0300 = Uses Windows API
                                    // 0x0200 = Can be run in a window
                                    // 0x0100 = full-screen text app

#define MODFLAGS_IMPLICIT_LOAD      0x0040  // DLL is implicitly loaded
#define MODFLAGS_WIN32              0x0010
#define MODFLAGS_AUTODATA           0x0002
#define MODFLAGS_SINGLEDATA         0x0001


typedef struct tagOFSTRUCT_EXT
{
    WORD cBytes;            //00 This is a single BYTE in the regular OFSTRUCT
    BYTE fFixedDisk;        //02
    UINT nErrCode;          //03
    BYTE reserved[4];       //05
    char szPathName[128];   //09
} OFSTRUCT_EXT;
```

Context

```
VMM:
00010123        _ContextCreate
00010124        _ContextDestroy
00010132        _ContextSwitch
00010157        _GetCurrentContext

VWIN32:
002A0003        _VWIN32_Get_Thread_Context
002A0004        _VWIN32_Set_Thread_Context
002A000C        VWIN32_GetContextHandle

See Also:
        Sample: context.cpp
        Source: Get aliase address of another context
        Sample: Context walk via VxD

Win32 VxD Services:
        0x010006     K0_0_ContextCreate          ;
        0x010007     K0_1_ContextDestroy         ;
        0x01000b     K0_1_ContextSwitch          ;
        0x01000f     K0_0_GetCurrentContext      ;
        0x2a0014     K0_2_GetThreadContext",
        0x2a0015     K0_2_SetThreadContext",


BOOL GetThreadContext(
    HANDLE  hThread,    // handle of thread with context
    LPCONTEXT  lpContext        // address of context structure
   );
BOOL SetThreadContext(
    HANDLE  hThread,    // handle of thread with context
    CONST CONTEXT *  lpContext  // address of context structure
   );
```

```
include winnt.h

#if !defined(RC_INVOKED)

#define CONTEXT_i386    0x00010000    // this assumes that i386 and
#define CONTEXT_i486    0x00010000    // i486 have identical context records

#define CONTEXT_CONTROL         (CONTEXT_i386 | 0x00000001L) // SS:SP, CS:IP, FLAGS, BP
#define CONTEXT_INTEGER         (CONTEXT_i386 | 0x00000002L) // AX, BX, CX, DX, SI, DI
#define CONTEXT_SEGMENTS        (CONTEXT_i386 | 0x00000004L) // DS, ES, FS, GS
#define CONTEXT_FLOATING_POINT  (CONTEXT_i386 | 0x00000008L) // 387 state
#define CONTEXT_DEBUG_REGISTERS (CONTEXT_i386 | 0x00000010L) // DB 0-3,6,7

#define CONTEXT_FULL (CONTEXT_CONTROL | CONTEXT_INTEGER |\
                      CONTEXT_SEGMENTS)

#endif

#define SIZE_OF_80387_REGISTERS      80
typedef struct _FLOATING_SAVE_AREA {    // 70h len
    DWORD   ControlWord;        //00
```

```
    DWORD   StatusWord;         //04
    DWORD   TagWord;            //08
    DWORD   ErrorOffset;        //0c
    DWORD   ErrorSelector;      //10
    DWORD   DataOffset;         //14
    DWORD   DataSelector;       //18
    BYTE    RegisterArea[SIZE_OF_80387_REGISTERS];      //1c
    DWORD   Cr0NpxState;        //6c
} FLOATING_SAVE_AREA;

typedef FLOATING_SAVE_AREA *PFLOATING_SAVE_AREA;

//
// Context Frame
//
//  This frame has a several purposes: 1) it is used as an argument to
//  NtContinue, 2) is is used to constuct a call frame for APC delivery,
//  and 3) it is used in the user level thread creation routines.
//
//  The layout of the record conforms to a standard call frame.
//

typedef struct _CONTEXT {

    //
    // The flags values within this flag control the contents of
    // a CONTEXT record.
    //
    // If the context record is used as an input parameter, then
    // for each portion of the context record controlled by a flag
    // whose value is set, it is assumed that that portion of the
    // context record contains valid context. If the context record
    // is being used to modify a threads context, then only that
    // portion of the threads context will be modified.
    //
    // If the context record is used as an IN OUT parameter to capture
    // the context of a thread, then only those portions of the thread's
    // context corresponding to set flags will be returned.
    //
    // The context record is never used as an OUT only parameter.
    //

    DWORD ContextFlags; //00

    //
    // This section is specified/returned if CONTEXT_DEBUG_REGISTERS is
    // set in ContextFlags.  Note that CONTEXT_DEBUG_REGISTERS is NOT
    // included in CONTEXT_FULL.
    //

    DWORD   Dr0;                //04
    DWORD   Dr1;                //08
    DWORD   Dr2;                //0c
    DWORD   Dr3;                //10
    DWORD   Dr6;                //14
    DWORD   Dr7;                //18

    //
    // This section is specified/returned if the
    // ContextFlags word contians the flag CONTEXT_FLOATING_POINT.
    //

    FLOATING_SAVE_AREA FloatSave;       //1c
```

```
    //
    // This section is specified/returned if the
    // ContextFlags word contians the flag CONTEXT_SEGMENTS.
    //

    DWORD   SegGs;              //8c
    DWORD   SegFs;              //90
    DWORD   SegEs;              //94
    DWORD   SegDs;              //98

    //
    // This section is specified/returned if the
    // ContextFlags word contians the flag CONTEXT_INTEGER.
    //

    DWORD   Edi;                //9c
    DWORD   Esi;                //a0
    DWORD   Ebx;                //a4
    DWORD   Edx;                //a8
    DWORD   Ecx;                //ac
    DWORD   Eax;                //b0

    //
    // This section is specified/returned if the
    // ContextFlags word contians the flag CONTEXT_CONTROL.
    //

    DWORD   Ebp;                //b4
    DWORD   Eip;                //b8
    DWORD   SegCs;              //bc MUST BE SANITIZED
    DWORD   EFlags;             //c0 MUST BE SANITIZED
    DWORD   Esp;                //c4
    DWORD   SegSs;              //c8

} CONTEXT;


typedef CONTEXT *PCONTEXT;
```

CRITICAL_SECTION
 24h len(doc only 18h)

```
include winbase.h
typedef RTL_CRITICAL_SECTION CRITICAL_SECTION;

include winnt.h
typedef struct _LIST_ENTRY {
   struct _LIST_ENTRY *Flink;
   struct _LIST_ENTRY *Blink;
} LIST_ENTRY, *PLIST_ENTRY, *RESTRICTED_POINTER PRLIST_ENTRY;

typedef struct _RTL_CRITICAL_SECTION_DEBUG {
    WORD   Type;                                //00
    WORD   CreatorBackTraceIndex;               //02
    struct _RTL_CRITICAL_SECTION *CriticalSection;//04
    LIST_ENTRY ProcessLocksList;                //08
    DWORD EntryCount;                           //10H
    DWORD ContentionCount;                      //14H
    DWORD Spare[ 2 ];                           //18H
} RTL_CRITICAL_SECTION_DEBUG, *PRTL_CRITICAL_SECTION_DEBUG;

#define RTL_CRITSECT_TYPE 0
#define RTL_RESOURCE_TYPE 1

typedef struct _RTL_CRITICAL_SECTION {
    PRTL_CRITICAL_SECTION_DEBUG DebugInfo;      //00

    //  The following three fields control entering and exiting the critical
    //  section for the resource
    LONG LockCount;                             //04
    LONG RecursionCount;                        //08
    HANDLE OwningThread;        // from the thread's ClientId->UniqueThread
    HANDLE LockSemaphore;                       //10H
    DWORD Reserved;                             //14H
} RTL_CRITICAL_SECTION, *PRTL_CRITICAL_SECTION;
```

```
Semaphore       struc
        WORD    ?               //00
        WORD    signature;      //02    'SE'
        DWORD   ?               //04
        DWORD   ?               //08
        DWORD   ?               //0c

        DWORD   oWhoCreate      //14 eip after call createSema

        ends
```

```
R3Mutex         struc
m0              dd ?            //00
Count           dd ?            //04
Owner_R3Thread  dd ?            //08
mc              dd ?            //0c
m10             dd ?
m14             dd ?
m18             dw ?
                db ? ;
                db ? ;
m1c             dd ?
nLevel          dd ?            //20
R3Mutex         ends

see also:
        PDB     offset 60h      goo
```

```
Mutex           struc
m0              db ? ;                  //00
                db ? ; undefined
Signature       dw ?                    ;//02 'MX'
m4              dd ?
                db ? ; undefined
                db ? ; undefined
                db ? ; undefined
                db ? ; undefined
Count           dd ?                    ;//0c EnterMutex count
Owner_R0TCB     dd ?            ;//10
priority        dd ?            ;//14
Owner_ThreadID  dw ?            ;//18
                dw ?
retAddr         dd 10 dup(?)    ;//1c ret addr of _EnterMutex,_EnterMustComplete
Last_EIP_to_acqire      dd      ?       ;//48h
Mutex           ends

Events
     9C    CreateEventA
     9D    CreateEventW
    218    OpenEventA
    219    OpenEventW
    24A    ResetEvent
    284    SetEvent
    22D    PulseEvent
        2D1    WaitForMultipleObjects
    2D2    WaitForMultipleObjectsEx
    2D3    WaitForSingleObject
    2D4    WaitForSingleObjectEx
        002A000E        _VWIN32_SetWin32Event
        002A000F        _VWIN32_PulseWin32Event
        002A0010        _VWIN32_ResetWin32Event                 ????
        002A0011        _VWIN32_WaitSingleObject
        002A0012        _VWIN32_WaitMultipleObjects

Semaphores
     AE    CreateSemaphoreA
     AF    CreateSemaphoreW
    221    OpenSemaphoreA
    222    OpenSemaphoreW
    247    ReleaseSemaphore
         00010027        Wait_Semaphore
         00010028        Signal_Semaphore
Mutexes
    21D    OpenMutexA
    21E    OpenMutexW
        00010101        _CreateMutex
        00010102        _DestroyMutex
        00010103        _GetMutexOwner
        0001012E        _EnterMutex             go
        0001012F        _LeaveMutex             go
        0001017F        Force_Mutexes_Free
        00010180        Restore_Forced_Mutexes
        000100FA        _BlockOnID              go
        00010127        _SignalID               go
Critical sections
     B9    KERNEL32!DeleteCriticalSection
     C3    KERNEL32!EnterCriticalSection
    1D8    KERNEL32!InitializeCriticalSection
    1EE    KERNEL32!LeaveCriticalSection
    203    KERNEL32!MakeCriticalSectionGlobal
    245    KERNEL32!ReinitializeCriticalSection
    2BF    KERNEL32!UninitializeCriticalSection

0001001F        Begin_Critical_Section
00010020        End_Critical_Section
00010021        End_Crit_And_Suspend
00010022        Claim_Critical_Section
00010023        Release_Critical_Section
```

```
00010029        Get_Crit_Section_Status
000100F1        Get_Crit_Status_No_Block        go
00010145        Get_Crit_Status_Thread

MustComplete
00010135        _EnterMustComplete
00010136        _LeaveMustComplete
00010137        _ResumeExecMustComplete
```

Get_Crit_Status_No_Block

```
 000100F1
include vmm.inc

VMMcall Get_Crit_Status_No_Block

mov     [VMHandle], ebx
mov     [Claims], ecx
jc      high_priority


Retrieves the claim count and handle of the owner of the critical section.
Unlike the Get_Crit_Section_Status service, this service returns immediately
(without blocking) even if a delayed request to release the critical section
is pending. This service is only available for Windows version 3.1 or later.
Uses Flags.

·       Returns the critical section claim count in ECX and the handle of the
virtual machine owning the critical section EBX. If ECX is 0, EBX is the handle
of the current virtual machine. The carry flag is set if the current virtual
machine has an execution priority greater than or equal to Critical_Section_Boost,
such as during a hardware interrupt simulation.

In some cases, this service may indicate that the critical section is currently
owned even when it will be released before returning to the virtual machine.

This is an asynchronous service; it may be called at interrupt time.

see also:
        Get_Crit_Section_Status
```

_BlockOnID

```
include vmm.inc

VMMcall _BlockOnID, <ThreadID, Flags>



        VMMCall _BlockOnID,<345678,0>           ;this will change es
Flags:
        BLOCK_FORCE_SVC_INTS            0x20
        BLOCK_THREAD_IDLE               0x10
        BLOCK_POLL                      8
        Block_Enable_Ints               4
        Block_Svc_If_Ints_Locked        2
        Block_Svc_Ints                  1

see also:
        _SignalID       go
        Suspend_VM
        Wait_Semaphore
        K32!SuspendThread
```

_SignalID

```
include vmm.inc

VMMCall _SignalID, <ID>
```

Signals an ID. Threads currently blocked on this ID will be unblocked.

see also:
_BlockOnID go

_EnterMutex
include vmm.inc

VMMcall _EnterMutex, <MutexHandle, Flags>

Enters a mutex. If the mutex is unowned, the calling thread becomes the owner and execution conti

· No return value.

MutexHandle

Handle of the mutex.

Flags

Action to take when interrupts occur while the virtual machine is blocked for the mutex. This par

Block_Enable_Ints Service interrupts in the virtual machine even if the virtual machine doe
Block_Svc_If_Ints_Locked Service events and simulated interrupts in the virtual machine if
Block_Svc_Ints Service events and simulated interrupts in the virtual machine if the thread bloc

Block_Thread_Idle Consider the thread to be idle if it blocks for the mutex.

The Block_Poll value is reserved and must not be used with this service.

If the mutex has a priority boost and the mutex is entered for the first time, the execution prio

---

_LeaveMutex
include vmm.inc

cCall _LeaveMutex, <MutexHandle>

Leaves a mutex. If the reentry count of the mutex is greater than one,
this function decrements the reentry count. If the reentry count is one,
it becomes unowned and the highest priority thread waiting for the mutex
is released.

· No return value.

MutexHandle

Handle of the mutex.

This service causes a context switch if the execution priority of the
released thread is higher priority than the current thread.
Only the thread owning a mutex can leave it.
When the critical section is freed, any threads waiting for the critical
section are released.

```
//==================================
// WALKHEAP - Matt Pietrek 1995
// FILE: HEAPW32.H
//==================================

typedef struct tagHEAP_ARENA_DEBUG
{
        DWORD   size;   // Size of block, including arena, OR'ed with A0000000
                                    // If bottom bit is set, block is free
        union
        {
                DWORD   alloc_EIP;      // 0x04 // If in-use block
                DWORD   prev;           // 0x04 // if free block
        }a;

        WORD    threadID;       // 0x08 (Free blocks are 0xFEFE)
        WORD    signature;      // 0x0A 0x4842 = "BH", 0x4846 = "FH"

        union
        {
                DWORD   checksum;       // 0x0C // If in-use block
                DWORD   next;           // 0x0C // If free block
        }b;
} HEAP_ARENA_DEBUG, *PHEAP_ARENA_DEBUG;


typedef struct tagHEAP_ARENA_RETAIL
{
        DWORD   size;   // Size of block, including arena, OR'ed with A0000000
                                    // If bottom bit is set, block is free
} HEAP_ARENA_RETAIL, *PHEAP_ARENA_RETAIL;


typedef struct tagFREE_HEAP_ARENA_DEBUG
{
        HEAP_ARENA_DEBUG arena;
        DWORD   freeBlockChecksum;              // 0x10 - only present if a free block
} FREE_HEAP_ARENA_DEBUG, *PFREE_HEAP_ARENA_DEBUG;


typedef struct tagFREE_HEAP_ARENA_RETAIL
{
        HEAP_ARENA_RETAIL       arena;
        DWORD                           prev;
        DWORD                           next;
} FREE_HEAP_ARENA_RETAIL, *PFREE_HEAP_ARENA_RETAIL;


typedef struct tagFREE_LIST_HEADER_DEBUG
{
        DWORD                                   dwMaxBlockSize;
        FREE_HEAP_ARENA_DEBUG   freeArena;
} FREE_LIST_HEADER_DEBUG, *PFREE_LIST_HEADER_DEBUG;

typedef struct tagFREE_LIST_HEADER_RETAIL
{
        DWORD                                   dwMaxBlockSize;
        FREE_HEAP_ARENA_RETAIL  freeArena;
} FREE_LIST_HEADER_RETAIL, *PFREE_LIST_HEADER_RETAIL;

typedef struct tagHEAP_HEADER_DEBUG
{
        DWORD   dwSize;         // 0x00 total size of heap (defaults to 1MB + 4K)
        DWORD   nextBlock;      // 0x04 next reserved block of memory in this heap

        FREE_LIST_HEADER_DEBUG  freeListArray[4];       // 0x8
                                                        // 0x08 start of array of free list struc
                                                        // 0x18 bytes long.  Array of these.  Siz
                                                        // are 0x20, 0x80, and 0x200, and < 0xFFF

        HANDLE                          nextHeap;                       // 0x68  Next heap in pro
```

```
        PCRITICAL_SECTION       pCriticalSection;       // 0x6C - for heap synchronization

        CRITICAL_SECTION        criticalSection;        // 0x70

        DWORD   unknown1[14];

        DWORD   creating_EIP;   // 0xC0
        DWORD   checksum;               // 0xC4 checksum
        WORD    creating_thread_ordinal;        // 0xC8
        WORD    unknown2;               // 0xCA

        BYTE    flags;                  // 0xCC HEAP_xxx flags
        BYTE    unknown3;               // 0xCD filler
        WORD    signature;              // 0xCE = 'HI'
} HEAP_HEADER_DEBUG, *PHEAP_HEADER_DEBUG;

typedef struct tagHEAP_HEADER_RETAIL
{
        DWORD   dwSize;         // 0x00 total size of heap (defaults to 1MB + 4K)
        DWORD   nextBlock;      // 0x04 next reserved block of memory in this heap

        FREE_LIST_HEADER_RETAIL freeListArray[4];       // 0x8
                                                        // 0x08 start of array of free list struc
                                                        // 0x18 bytes long.  Array of these.  Siz
                                                        // are 0x20, 0x80, and 0x200, and < 0xFFF

        HANDLE                          nextHeap;                       // 0x48  Next heap in pro

        PCRITICAL_SECTION       pCriticalSection;       // 0x4C - for heap synchronization

        CRITICAL_SECTION        criticalSection;        // 0x50

        DWORD   unknown1[2];                                    // 0x68

        BYTE    flags;                  // 0x70 HEAP_xxx flags
        BYTE    unknown2;               // 0x71 filler
        WORD    signature;              // 0x72
} HEAP_HEADER_RETAIL, *PHEAP_HEADER_RETAIL;
```

PROCESSENTRY32

PROCESS_INFORMATION

```
//winbase.h ----------------------------------------

typedef struct _PROCESS_INFORMATION {
    HANDLE hProcess;
    HANDLE hThread;
    DWORD dwProcessId;
    DWORD dwThreadId;
} PROCESS_INFORMATION, *PPROCESS_INFORMATION, *LPPROCESS_INFORMATION;
```

PROCESS_HEAP_ENTRY

```
//winbase.h ----------------------------------------

typedef struct _PROCESS_HEAP_ENTRY {
    PVOID lpData;
    DWORD cbData;
    BYTE cbOverhead;
    BYTE iRegionIndex;
    WORD wFlags;
    union {
        struct {
            HANDLE hMem;
            DWORD dwReserved[ 3 ];
        } Block;
        struct {
            DWORD dwCommittedSize;
            DWORD dwUnCommittedSize;
            LPVOID lpFirstBlock;
            LPVOID lpLastBlock;
        } Region;
#if defined(__cplusplus) || defined(_ANONYMOUS_UNION)
    };
#else
    }u;
#endif
} PROCESS_HEAP_ENTRY, *LPPROCESS_HEAP_ENTRY, *PPROCESS_HEAP_ENTRY;
```

HEAPENTRY32

```
//tlhelp32.h----------------------------------------
typedef struct tagHEAPENTRY32
{
    DWORD  dwSize;
    HANDLE hHandle;     // Handle of this heap block
    DWORD  dwAddress;   // Linear address of start of block
    DWORD  dwBlockSize; // Size of block in bytes
    DWORD  dwFlags;
    DWORD  dwLockCount;
    DWORD  dwResvd;
    DWORD  th32ProcessID;   // owning process
    DWORD  th32HeapID;      // heap block is in
} HEAPENTRY32;
```

PROCESSENTRY32

```
//tlhelp32.h----------------------------------------
```

```
typedef struct tagPROCESSENTRY32
{
    DWORD   dwSize;             //00
    DWORD   cntUsage;           //04
    DWORD   th32ProcessID;          //08 this process
    DWORD   th32DefaultHeapID;  //0c
    DWORD   th32ModuleID;           //10 associated exe
    DWORD   cntThreads;         //14
    DWORD   th32ParentProcessID;    //18 this process's parent process
    LONG    pcPriClassBase;         //1c Base priority of process's threads
    DWORD   dwFlags;            //20
    char    szExeFile[MAX_PATH];    //24 Path
} PROCESSENTRY32;

see also:
        struct THCB     offset 20h      go
        Process32First          go
        Process32Next
```

THREADENTRY32

```
//tlhelp32.h-------------------------------------------
typedef struct tagTHREADENTRY32
{
    DWORD   dwSize;
    DWORD   cntUsage;
    DWORD   th32ThreadID;       // this thread
    DWORD   th32OwnerProcessID; // Process this thread is associated with
    LONG    tpBasePri;
    LONG    tpDeltaPri;
    DWORD   dwFlags;
} THREADENTRY32;
```

MODULEENTRY32

```
//tlhelp32.h-------------------------------------------
typedef struct tagMODULEENTRY32
{
    DWORD   dwSize;
    DWORD   th32ModuleID;       // This module
    DWORD   th32ProcessID;      // owning process
    DWORD   GlblcntUsage;       // Global usage count on the module
    DWORD   ProccntUsage;       // Module usage count in th32ProcessID's context
    BYTE  * modBaseAddr;        // Base address of module in th32ProcessID's context
    DWORD   modBaseSize;        // Size in bytes of module starting at modBaseAddr
    HMODULE hModule;            // The hModule of this module in th32ProcessID's context
    char    szModule[MAX_MODULE_NAME32 + 1];
    char    szExePath[MAX_PATH];
} MODULEENTRY32;
```

ClientReg

```
// Client Register Structure

// The following structures represent the data which EBP points to when
// VxD routines are entered - both VxD control calls and traps.
// The structures are defined in three forms - DWORD, WORD, and BYTE offsets

struct Client_Reg_Struc {
    DWORD       Client_EDI;                     //00
    DWORD       Client_ESI;                     //04
    DWORD       Client_EBP;                     //08
    DWORD       Client_res0;
    DWORD       Client_EBX;                     //10
    DWORD       Client_EDX;                     //14
    DWORD       Client_ECX;                     //18
    DWORD       Client_EAX;                     //1c
```

```
    DWORD       Client_Error;           //20
    DWORD       Client_EIP;                     //24h
    WORD        Client_CS;                      //28h
    WORD        Client_res1;
    DWORD       Client_EFlags;          //2c
    DWORD       Client_ESP;                     //30
    WORD        Client_SS;
    WORD        Client_res2;
    WORD        Client_ES;
    WORD        Client_res3;
    WORD        Client_DS;
    WORD        Client_res4;
    WORD        Client_FS;                      //40h
    WORD        Client_res5;
    WORD        Client_GS;                      //44h
    WORD        Client_res6;
    DWORD       Client_Alt_EIP;         //48h
    WORD        Client_Alt_CS;          //4ch
    WORD        Client_res7;
    DWORD       Client_Alt_EFlags;      //50h
    DWORD       Client_Alt_ESP;         //54h
    WORD        Client_Alt_SS;          //58h
    WORD        Client_res8;
    WORD        Client_Alt_ES;          //5ch
    WORD        Client_res9;
    WORD        Client_Alt_DS;          //60h
    WORD        Client_res10;
    WORD        Client_Alt_FS;          //64h
    WORD        Client_res11;
    WORD        Client_Alt_GS;          //68h
    WORD        Client_res12;
};


struct Client_Word_Reg_Struc {
    WORD        Client_DI;
    WORD        Client_res13;
    WORD        Client_SI;
    WORD        Client_res14;
    WORD        Client_BP;
    WORD        Client_res15;
    DWORD       Client_res16;
    WORD        Client_BX;
    WORD        Client_res17;
    WORD        Client_DX;
    WORD        Client_res18;
    WORD        Client_CX;
    WORD        Client_res19;
    WORD        Client_AX;
    WORD        Client_res20;
    DWORD       Client_res21;
    WORD        Client_IP;
    WORD        Client_res22;
    DWORD       Client_res23;
    WORD        Client_Flags;
    WORD        Client_res24;
    WORD        Client_SP;
    WORD        Client_res25;
    DWORD       Client_res26[5];
    WORD        Client_Alt_IP;
    WORD        Client_res27;
    DWORD       Client_res28;
    WORD        Client_Alt_Flags;
    WORD        Client_res29;
    WORD        Client_Alt_SP;
```

```
};



struct Client_Byte_Reg_Struc {
    DWORD       Client_res30[4];               /* EDI, ESI, EBP, ESP at pushall */
    BYTE        Client_BL;
    BYTE        Client_BH;
    WORD        Client_res31;
    BYTE        Client_DL;
    BYTE        Client_DH;
    WORD        Client_res32;
    BYTE        Client_CL;
    BYTE        Client_CH;
    WORD        Client_res33;
    BYTE        Client_AL;
    BYTE        Client_AH;
};

typedef union tagCLIENT_STRUC {
    struct Client_Reg_Struc       CRS;
    struct Client_Word_Reg_Struc  CWRS;
    struct Client_Byte_Reg_Struc  CBRS;
    } CLIENT_STRUCT;

typedef CLIENT_STRUCT* PCLIENT_STRUCT;

// Define offsets into client register struct for VMM Map_Flat service

#define CLIENT_DI       0
#define CLIENT_SI       4
#define CLIENT_BP       8
#define CLIENT_BX       16
#define CLIENT_DX       20
#define CLIENT_CX       24
#define CLIENT_AX       28
#define CLIENT_IP       36
#define CLIENT_CS       40
#define CLIENT_SP       48
#define CLIENT_SS       52
#define CLIENT_ES       56
#define CLIENT_DS       60
#define CLIENT_FS       64
#define CLIENT_GS       68
```

```
        Wnd structure

See also:
        wnd.arj
        ShowWnd, Win95Scrit     Matt Pietrek 1995


//==================================
// SHOWWND - Matt Pietrek 1995
// FILE: HWND32.H
//==================================
typedef struct _RECTS
{
    WORD    left;
    WORD    top;
    WORD    right;
    WORD    bottom;
} RECTS, *PRECTS, *LPRECTS;
```

WND32

```
W95SPS p172
typedef struct _WND32
{
struct _WND32 *hWndNext;    //00h (GW_HWNDNEXT) HWND of next sibling window
struct _WND32 *hWndChild;   //04h (GW_CHILD) First child window
struct _WND32 *hWndParent;  //08h Parent window handle
struct _WND32 *hWndOwner;   //0Ch Owning window handle
RECTS   rectWindow;         //10h Rectangle describing entire window
RECTS   rectClient;         //18h Rectangle for client area of window
WORD    hQueue;             //20h Application message queue handle go
WORD    hrgnUpdate;         //22h window region needing an update
WORD    wndClass;           //24h handle to an INTWNDCLASS
WORD    hInstance;          //26h hInstance of creating application
WNDPROC lpfnWndProc;        //28h Window procedure address
DWORD   dwFlags;            //2Ch internal state flags
DWORD   dwStyleFlags;       //30h WS_XXX style flags
DWORD   dwExStyleFlags;     //34h WS_EX_XXX extended style flags
DWORD   moreFlags;          //38h flags
HANDLE  ctrlID;             //3Ch GetDlgCtrlId or hMenu
WORD    windowTextOffset;   //40h Offset of the window's text in atom heap
WORD    scrollBar;          //42h DWORD associated with the scroll bars
WORD    properties;         //44h Handle for first window property
WORD    hWnd16;             //46h Actual HWND value for this window
struct _WND32 *lastActive;  //48h Last active owned popup window
HANDLE  hMenuSystem;        //4Ch handle to the system menu
DWORD   un1;                //50h
WORD    un2;                //54h
WORD    classAtom;          //56h See also offs. 2 in the field 24 struct ptr
DWORD   alternatePID;       //58h
DWORD   alternateTID;       //5Ch
} WND32, *PWND32;


DWORD_FLAGS WndStyles[]=
{
{ 0x80000000L, "WS_POPUP" },
```

```
{ 0x40000000L, "WS_CHILD" },
{ 0x20000000L, "WS_MINIMIZE" },
{ 0x10000000L, "WS_VISIBLE" },
{ 0x08000000L, "WS_DISABLED" },
{ 0x04000000L, "WS_CLIPSIBLINGS" },
{ 0x02000000L, "WS_CLIPCHILDREN" },
{ 0x01000000L, "WS_MAXIMIZE" },
{ 0x00800000L, "WS_BORDER" },
{ 0x00400000L, "WS_DLGFRAME" },
{ 0x00200000L, "WS_VSCROLL" },
{ 0x00100000L, "WS_HSCROLL" },
{ 0x00080000L, "WS_SYSMENU" },
{ 0x00040000L, "WS_THICKFRAME" },
{ 0x00020000L, "WS_MINIMIZEBOX" },
{ 0x00010000L, "WS_MAXIMIZEBOX" },
};

DWORD_FLAGS WndExStyles[]=
{
{ 0x00000001L, "WS_EX_DLGMODALFRAME" },
{ 0x00000004L, "WS_EX_NOPARENTNOTIFY" },
{ 0x00000008L, "WS_EX_TOPMOST" },
{ 0x00000010L, "WS_EX_ACCEPTFILES" },
{ 0x00000020L, "WS_EX_TRANSPARENT" },
{ 0x00000040L, "WS_EX_MDICHILD" },
{ 0x00000080L, "WS_EX_TOOLWINDOW" },
{ 0x00000100L, "WS_EX_WINDOWEDGE" },
{ 0x00000200L, "WS_EX_CLIENTEDGE" },
{ 0x00000400L, "WS_EX_CONTEXTHELP" },
{ 0x00001000L, "WS_EX_RIGHT" },
{ 0x00002000L, "WS_EX_RTLREADING" },
{ 0x00004000L, "WS_EX_LEFTSCROLLBAR" },
{ 0x00010000L, "WS_EX_CONTROLPARENT" },
{ 0x00020000L, "WS_EX_STATICEDGE" },
{ 0x00040000L, "WS_EX_APPWINDOW" },

typedef struct _WNDCLASS {    // wc
    UINT    style;              //00
    WNDPROC lpfnWndProc;        //02
    int     cbClsExtra;         //06
    int     cbWndExtra;         //08
    HANDLE  hInstance;          //0a
    HICON   hIcon;              //0c
    HCURSOR hCursor;            //0e
    HBRUSH  hbrBackground;      //10
    LPCTSTR lpszMenuName;       //12
    LPCTSTR lpszClassName;      //16
} WNDCLASS;
```

```
        DWORD   d=pGet_Module16("USER");
        prtl("User.exe is %x",d);

        d=getBase(0,d);

        d=d + pd(d+0x10000+h);
```

```
        PSTR p=(PSTR)d;

        prtl("wnd addr is %x",p);

        disp_struct(WND_str,p);
```

```
//==================================
// SHOWWND - Matt Pietrek 1995
// FILE: MSGQUEUE.H
//==================================

#pragma pack(1)

// type == LT_USER_VWININFO(0x1B), offset 10h in msg queue
typedef struct _PERQUEUEDATA
{
WORD    npNext;         //00h a USER heap handle (type == LT_USER_VWININFO)
WORD    un2;            //02h
WORD    un3;            //04h
WORD    npQMsg;         //06h type == LT_USER_QMSG
WORD    un5;            //08h
WORD    un6;            //0Ah
WORD    un7;            //0Ch
WORD    un8;            //0Eh
WORD    un9;            //10h
WORD    un10;           //12h
WORD    somehQueue1;    //14h a msg queue handle
WORD    somehQueue2;    //16h a msg queue handle
DWORD   hWndCapture;    //18h
DWORD   hWndFocus;      //1Ch
DWORD   hWndActive;     //20h
} PERQUEUEDATA, *PPERQUEUEDATA;

// type == LT_USER_QMSG(0x1A)
typedef struct _QUEUEMSG
{
WORD    hWnd;           //00h
WORD    msg;            //02h
WORD    wParamLow;      //04h
DWORD   lParam;         //06h
DWORD   messageTime;    //0Ah  GetMessageTime
DWORD   messagePos;     //0Eh  GetMessagePos
WORD    wParamHigh;     //12h  HIWORD of wParam for 32 bit apps
DWORD   extraInfo;      //14h  GetMessageExtraInfo
WORD    nextQueueMsg;   //18h  Near offset to next QUEUEMSG
} QUEUEMSG, *PQUEUEMSG;

// LT_USER_PROCESS(0x1D), offset 16h in the msg queue
// All queues belonging to the same process have a pointer to this struct
typedef struct _QUEUEPROCESSDATA
{
WORD    npNext;     //00h ptr to next QUEUEPROCESSDATA
WORD    un2;        //02h type == LT_USER_SUBSYSTEM
WORD    flags;      //04h
WORD    un3;        //06h
DWORD   processId;  //08h e.g., GetCurrentProcessId
WORD    un5;        //0Ch
WORD    hQueue;     //0Eh an hQueue belonging to this process (which one?)
} QUEUEPROCESSDATA, *PQUEUEPROCESSDATA;

MSGQUEUE
```

```
W95SPS p166
typedef struct _MSGQUEUE
{
WORD    nextQueue;  //00h  next queue in the list
WORD    hTask;      //02h  Task that this queue is associated with
WORD    headMsg;    //04h  Near ptr to head of linked list of QUEUEMSGs
WORD    tailMsg;    //06h  Near ptr to end of list of QUEUEMSGs
WORD    lastMsg;    //08h  Near ptr to last msg retrieved (not really!)
WORD    cMsgs;      //0Ah  Number of messages
BYTE    un1;        //0Ch  ???
BYTE    sig[3];     //0Dh  "MJT" (Jon Thomason?)
WORD    npPerQueue; //10h  16 bit offset in USER DGROUP to PERQUEUEDATA
                    //     type == LT_USER_VWININFO???
WORD    un2;        //12h  ???
WORD    un2_5;      //14h  ??
WORD    npProcess;  //16h  near pointer in USER DGROUP to a QUEUEPROCESSDATA
DWORD   un3[3];     //18h  ???
DWORD   messageTime;//24h  retrieved by GetMessageTime()
DWORD   messagePos; //28h  retrived by GetMessagePos()
WORD    un4;        //2Ch  ??? (seems to always be 0)
WORD    lastMsg2;   //2Eh  Near ptr to last retrieved QUEUEMSG
DWORD   extraInfo;  //30h  returned by GetMessageExtraInfo()
DWORD   un5[2];     //34h  ???
DWORD   threadId;   //3Ch  See GetWindowProcessThreadId
WORD    un6;        //40h  ??
WORD    expWinVer;  //42h  Version of Windows this app expects
DWORD   un7;        //44h  ???
WORD    ChangeBits; //48h  high order word returned by GetQueueStatus
WORD    WakeBits;   //4Ah  low order word returned by GetQueueStatus
WORD    WakeMask;   //4Ch  The QS_xxx bits that GetMessage/PeekMessage are
                    //     waiting for
WORD    un8;        //4Eh  ???
WORD    hQueueSend; //50h  App that's in SendMessage to this queue
DWORD   un9;        //52h  ???
WORD    sig2;       //56h  "HQ"
} MSGQUEUE, *PMSGQUEUE;

#pragma pack()
```

```
include:        <vmm.h> <vmm.inc>

Ring0 Thread Control Block

typedef struct tcb_s{
//      DWORD   TCB_ExecTime            //-4c
//      BYTE    TCB_nMustComplete       //-3b _EnterMutex
//      BYTE    TCB_?                   //-3a
//      BYTE    TCB_mustComplete;       //-33   _EnterMustComplete
//      DWORD   TCB_BasePriority        //-24 see Get_Thread_Win32_Pri
//      DWORD   TCB_CurrentPriority;    //-20
//      DWORD   TCB_InversionPriority;  //-1c
//      DWORD   TCB_TimeDecayBoost;     //-10

//      DWORD   TCB_whocallEnterMutex;  //-08 code addr after _EnterMutex,_EnterMustComplete

    ULONG   TCB_Flags;          //00 Thread status flags go
    ULONG   TCB_Reserved1;      //04 Used internally by VMM
    ULONG   TCB_Reserved2;      //08 Used internally by VMM
    ULONG   TCB_Signature;      //0c 'THCB'
    ULONG   TCB_ClientPtr;      //10 Client registers of thread go
    ULONG   TCB_VMHandle;       //14 VM that thread is part of goo
    USHORT  TCB_ThreadId;       //18 Unique Thread ID
    USHORT  TCB_PMLockOrigSS;   //1a Original SS:ESP before lock stack
    ULONG   TCB_PMLockOrigESP;  //1c
    ULONG   TCB_PMLockOrigEIP;  //20 Original CS:EIP before lock stack
    ULONG   TCB_PMLockStackCount;       //24
    USHORT  TCB_PMLockOrigCS;   //28
    USHORT  TCB_PMPSPSelector;  //2a
    ULONG   TCB_ThreadType;     //2c dword passed to VMMCreateThread
    USHORT  TCB_pad1;           //30 reusable; for dword align
    UCHAR   TCB_pad2;           //32 reusable; for dword align
    UCHAR   TCB_extErrLocus;    //33 extended error Locus
    USHORT  TCB_extErr;         //34 extended error Code
    UCHAR   TCB_extErrAction;   //36    Action
    UCHAR   TCB_extErrClass;    //37    Class
    ULONG   TCB_extErrPtr;      //38 "  pointer
//      DWORD   TCB_ExecPriority        //3c
//      DWORD   TCB_Next                //40h
//      DWORD   TCB_fEvent              //44h see _AtEvent
//      DWORD   TCB_?48                 //48h see _AtEvent
//      DWORD   TCB_TDBX                //8ch goo


debug:
//      DWORD   TCB_Next                //44h see Get_Next_Thread_Handle
//      DWORD   TCB_ListOf_NestExecStatus       //4ch see Get_Nest_Exec_Status
//      DWORD   TCB_StackFrame          //50h
//      DWORD   TCB_hPage?              //54h
//      BYTE    TCB_fBlockOnID          //76h
//      DWORD   TCB_TimeSliceGra        //7ch
//      DWORD   TCB_nTimeSlice          //-10h see Set_Thread_Static_Boost
} TCB,*PTCB;


see also:
        Ring3TCB        offset 5c       goo
        Get_Cur_Thread_Handle
        Get_Sys_Thread_Handle
        Get_Next_Thread_Handle

#define SCHED_OBJ_ID_THREAD         0x42434854    // THCB in ASCII
```

```
/*
 *  Thread status indicates globally interesting thread states.
 *  Flags are for information only and must not be modified.
 */
```

```
#define THFLAG_SUSPENDED_BIT        0x03   // Thread not scheduled
#define THFLAG_SUSPENDED                   (1L << THFLAG_SUSPENDED_BIT)
#define THFLAG_NOT_EXECUTEABLE_BIT  0x04   // Thread partially destroyed
#define THFLAG_NOT_EXECUTEABLE             (1L << THFLAG_NOT_EXECUTEABLE_BIT)
#define THFLAG_THREAD_CREATION_BIT  0x08   // Thread in status nascendi
#define THFLAG_THREAD_CREATION             (1L << THFLAG_THREAD_CREATION_BIT)
#define THFLAG_THREAD_BLOCKED_BIT   0x0A   // Blocked on semaphore
#define THFLAG_THREAD_BLOCKED              (1L << THFLAG_THREAD_BLOCKED_BIT)
#define THFLAG_RING0_THREAD_BIT     0x1C   // thread runs only at ring 0
#define THFLAG_RING0_THREAD                (1L << THFLAG_RING0_THREAD_BIT)
#define THFLAG_CHARSET_BITS         0x10   // Default character set
#define THFLAG_CHARSET_MASK                (3L << THFLAG_CHARSET_BITS)
#define THFLAG_ANSI                        (0L << THFLAG_CHARSET_BITS)
#define THFLAG_OEM                         (1L << THFLAG_CHARSET_BITS)
#define THFLAG_UNICODE                     (2L << THFLAG_CHARSET_BITS)
#define THFLAG_RESERVED                    (3L << THFLAG_CHARSET_BITS)
```

```
THFLAG_SUSPENDED        0x00000008
THFLAG_NOT_EXECUTEABLE  0x00000010
THFLAG_THREAD_CREATION  0x00000100
THFLAG_THREAD_BLOCKED   0x00000400
THFLAG_OEM              0x00010000
THFLAG_UNICODE          0x00020000
THFLAG_RING0_THREAD     0x10000000
????                    0x00000001
????                    0x00000800
????                    0x00001000
TCB_Nuke_Pending        0x00002000
????                    0x00100000
????                    0x00400000
????                    0x01000000
THFLAG_Swapping         0x02000000
THFLAG_CallOnMyStack    0x04000000      ;see _Call_On_My_Stack
????                    0x08000000

struct ThreadDataSlot{
        TDBX            //00
        Suspend_Count   //20h   max 7f
        }
```

```
include vmm.inc
// Device Data Block structure
Vxd_Desc_Block
```

```
typedef struct tagDDB {         //edh len
    DWORD       DDB_Next;                       //00 addr of next VxD in chain, or 0
    WORD        DDB_SDK_Version;        //04
    WORD        DDB_Req_Device_Number;  //06 Device ID (or Undefined_Device_ID)
    BYTE        DDB_Dev_Major_Version;  //08 Major version number
    BYTE        DDB_Dev_Minor_Version;  //09 Minor version number
    WORD        DDB_Flags;                      //0a Flags for init calls complete go
    BYTE        DDB_Name[8];            //0c Device name,padded with spaces
    DWORD       DDB_Init_Order;         //14 Init ordder (Undefined_Init_Order)
    DWORD       DDB_Control_Proc;       //18 Offset of control procedure
    DWORD       DDB_V86_API_Proc;       //1c Offset of API procedure (if present)
    DWORD       DDB_PM_API_Proc;        //20 Offset of API procedure (if present)
    DWORD       DDB_V86_API_CSIP;       //24 CS:IP of API entry point (if present)
                                                    // V86 mode seg:ofs callback addr
    DWORD       DDB_PM_API_CSIP;        //28 CS:IP of API entry point (if present)
                                                    // prot mode sel:ofs callback add

    DWORD       DDB_Reference_Data;     //2c Reference data from real mode
    DWORD       DDB_VxD_Service_Table_Ptr;  //30 Pointer to service table (if present)
    DWORD       DDB_VxD_Service_Table_Size; //34 Number of services (if any)
    DWORD       DDB_Win32_Service_Table;//38 Pointer to Win32 services (if any)
    DWORD       DDB_Prev;               //3c Ptr to prev 4.0 DDB
    DWORD       DDB_Size;               //40 INIT <SIZE(VxD_Desc_Block)> Reserved
    DWORD       DDB_Reserved1;        /* INIT <'Rsv1'> Reserved */
    DWORD       DDB_Reserved2;        /* INIT <'Rsv2'> Reserved */
    DWORD       DDB_Reserved3;        /* INIT <'Rsv3'> Reserved */
} DDB, *PDDB, VMMDDB, *PVMMDDB, **PPVMMDDB;
```

DDB_Flags

```
/*
 *  Flag values for DDB_Flags
 */

#define DDB_SYS_CRIT_INIT_DONE_BIT      0
#define DDB_SYS_CRIT_INIT_DONE          (1 << DDB_SYS_CRIT_INIT_DONE_BIT)
#define DDB_DEVICE_INIT_DONE_BIT        1
#define DDB_DEVICE_INIT_DONE            (1 << DDB_DEVICE_INIT_DONE_BIT)

#define DDB_HAS_WIN32_SVCS_BIT          14
#define DDB_HAS_WIN32_SVCS              (1 << DDB_HAS_WIN32_SVCS_BIT)
#define DDB_DYNAMIC_VXD_BIT             15
#define DDB_DYNAMIC_VXD                 (1 << DDB_DYNAMIC_VXD_BIT)

#define DDB_DEVICE_DYNALINKED_BIT       13
#define DDB_DEVICE_DYNALINKED           (1 << DDB_DEVICE_DYNALINKED_BIT)

DDB_SYS_CRIT_INIT_DONE  =       1
DDB_DEVICE_INIT_DONE    =       2
DDB_HAS_WIN32_SVCS              =       4000h           ;has Win32_VxD_Service
see also:
        VMMCall VMM_GetDDBList
        VMMCall Get_DDB go

        Structure DeviceInfo    go
        VxDcall VXDLDR_GetDeviceList    270005
```

DeviceInfo

```
struct DeviceInfo {
    struct DeviceInfo *DI_Next;    //00
    UCHAR  DI_Loaded;              //04
    struct VxD_Desc_Block *DI_DDB; //05 DDB
    USHORT DI_DeviceID;            //09
    CHAR   *DI_ModuleName;         //0b
    ULONG  DI_Signature;           //0f 'DLVX'
    ULONG  DI_ObjCount;            //13
    struct ObjectInfo *DI_ObjInfo; //17 ObjectInfo
    ULONG  DI_V86_API_CSIP;        //1b
    ULONG  DI_PM_API_CSIP;         //1f
};

See also:
        VxD overall     go
        VxDcall VXDLDR_GetDeviceList    270005
        structure DDB (Device Data Block)              go
        int68 ax=5080h,5081h    go
        VxDLocationList for static VxD
```

```
0030:C14BFEA0 80 02 4A C1 01 8C B3 58-C1 99 19 F0 F3 5A C1 58  ..J....X.....Z.X
0030:C14BFEB0 56 4C 44 03 00 00 00 10-EC 4B C1 F7 20 A2 FD E4  VLD......K.. ...
0030:C14BFEC0 03 3B 00 01 00 00 00 00-00 00 00 00 00 00 00 00  .;..............
0030:C14BFED0 30 02 4A C1 F2 E7 41 C0-80 74 28 C5 01 00 03 10  0.J...A..t(.....
0030:C14BFEE0 20 00 A8 14 41 C0 88 F2-9E FF 04 F2 72 00 00 00   ...A.......r...
0030:C14BFEF0 00 00 00 00 CA F1 41 C0-80 77 28 C5 01 00 03 10  ......A..w(.....
0030:C14BFF00 20 00 44 15 41 C0 74 F2-9E FF 04 B2 82 00 00 00   .D.A.t.........
0030:C14BFF10 50 47 4A C1 56 2C 44 C0-00 7B 28 C5 01 00 03 10  PGJ.V,D..{(.....
```

Contains information about a VxD.

DI_Next

Address of the next **DeviceInfo** structure in the list of devices maintained by VXDLDR. When there are no more devices, this element is NULL.

DI_Loaded

Nonzero if the VxD is currently loaded.

DI_DDB

Address of the **VxD_Desc_Block** structure for the VxD.

DI_DeviceID

The device identifier for the VxD.

DI_ModuleName

Address of the name of the VxD module.

DI_Signature

A unique value used by the system to verify the structure.

DI_ObjCount

Number of **ObjectInfo** structures pointed to by the **DI_ObjInfo** member.

DI_ObjInfo

Address of an array of **ObjectInfo** structures. Each **ObjectInfo** structure describes one of the VxD's memory objects.

DI_V86_API_CSIP

Save area for the virtual-86 mode entry point.

DI_PM_API_CSIP

Save area for the protected-mode entry point.

ObjectInfo

```
struct ObjectInfo {
    ULONG OI_LinearAddress;  // starting address of object
    ULONG OI_Size;           // size of object (in bytes)
    ULONG OI_ObjType;        // see below
    ULONG OI_Resident;       // see below
};
```

Contains information about a memory object in a VxD.

OI_ObjType

Object type, which can be one of the following values.

Constant	Value	Meaning
RCODE_OBJ	-1	Real-mode stub (ignored for dynamically loadable VxDs)

LCODE_OBJ	0x01	Locked code segment
LDATA_OBJ	0x02	Locked data segment
PCODE_OBJ	0x03	Pageable code segment
PDATA_OBJ	0x04	Pageable data segment
S CODE_OBJ	0x05	Static code segment (dynamically loadable VxDs only)
SDATA_OBJ	0x06	Static data segment (dynamically loadable VxDs only)
CODE16_OBJ	0x07	16-bit V86 segment
ICODE_OBJ	0x11	Initialization-time only code segment
IDATA_OBJ	0x12	Initialization-time only data segment

ICODE16_OBJ 0x13	Initialization-time only 16-bit V86 segment

OI_Resident

Whether or not the memory object is static. This value is zero for VxDs that are not dynamically loadable.

The **DeviceInfo** structure includes the address of an array of **ObjectInfo** structures that describe a VxDs memory objects.

Get_Initial_Thread_Handle

```
include vmm.inc

mov     ebx, [VMHandle]
VMMCall Get_Initial_Thread_Handle
mov     [hThread], edi
```

Returns a handle identifying a thread that was created at the same time that the virtual machine

· Returns a handle in EDI that identifies the VM's initial thread.

VMHandle

Identifies the virtual machine for which the thread was created.

This service can be called at interrupt time.

---

back to VMM

00010129 **_Register_Win32_Services**

If you want to make a VxD with Win32VxDService, you must
call this service during init to register it.

```
include vmm.inc

push    DDB_Offset
push    OFFSET32 Service_Table
VMMCall _Register_Win32_Services

------from vmm.vxd------
_Register_Win32_Services proc near

a_DDB           = dword ptr  8
a_pWin32VxDTable= dword ptr  0Ch

        push    54h ; 'T'
        VMMcall _Debug_Flags_Service
        push    ebp
        mov     ebp, esp
        push    eax
        push    edx
        mov     eax, [ebp+a_DDB]
        cmp     [eax+DDB.DDB_SDK_version], 400h
        jnb     short loc_E1098
        push    offset aRegister_win32
        VMMcall _Trace_Out_Service

loc_E1098:
        jb      short loc_E10B6
        mov     edx, [ebp+a_pWin32VxDTable]
        mov     [eax+DDB.DDB_Win32_VxD_Service_Table], edx
        or      [eax+DDB.DDB_Flags], 4000h
        movzx   eax, [eax+DDB.DDB_Device_ID]
        cmp     eax, 40h ; '@'
        jnb     short loc_E10B6
        mov     ds:Win32ServicesProvidersTable[eax*4], edx

loc_E10B6:
        pop     edx
        pop     eax
        leave
        retn
_Register_Win32_Services endp
```

10088 Get_Cur_PM_App_CB

```
include vmm.inc

mov     ebx, VMHandle
VMMcall Get_Cur_PM_App_CB
mov     [ControlBlock], edi
```

Retrieves a pointer to the application control block for a protected-mode application. Uses EDI.

Returns the address of the application control block in EDI.

VMHandle

Handle of the virtual machine in which the protected-mode application is running.

It is an error to call this service if the virtual machine is not running a protected-mode applic

See also:
PMcb

System_Control

```
include vmm.inc

mov     eax, Message    ; system control message
mov     ebx, VM         ; VM handle (if needed by message)
mov     esi, Param1     ; message-specific parameter
mov     edi, Param2     ; message-specific parameter
mov     edx, Param3     ; message-specific parameter
VMMcall System_Control
jc      error           ; carry flag set if error
```

Sends system control messages to all the virtual devices and,
depending on the message, to the VMM. Uses Flags, and possibly
other registers depending on the service.

Returns with the carry flag clear if successful.

see also:
System Control Messages go

Create_VM May only be sent by the virtual shell device.
VM_Init
Destroy_VM May only be sent by the virtual shell device.
Destroy_VM2
VM_Terminate
VM_Terminate2
VM_Suspend
VM_Suspend2
VM_Resume
VM_Not_Executeable
VM_Not_Executeable2
VM_Critical_Init

Set_Device_Focus
  May be sent by any virtual device.
  If the device ID is zero, all devices
  with a focus that can be set, must
  set their focus to the specified virtual
  machine.
End_PM_App May only be sent by the virtual MS-DOS manager.
Thread_Init
Create_Thread
Destroy_Thread

```
        Terminate_Thread
        Thread_Not_Executeable
        Sys_VM_Init
        Sys_VM_Terminate
        Sys_VM_Terminate2
        System_Exit
        System_Exit2
        Sys_Critical_Init
        Sys_Critical_Exit
        Sys_Critical_Exit2
```

```
Hook_Sys_Ctl    proc
        pusha
        mov     eax,10093h      ;System_Control
        mov     esi, OFFSET32 newSysCtl
        VMMcall Hook_Device_Service

        jnc     @@4
        d_msg_l 'Hook System_Control error!'
@@4:
        popa
        ret
        endp
UnHook_Sys_Ctl  proc
        pusha
        mov     eax,10093h      ;System_Control
        mov     esi, OFFSET32 newSysCtl
        VMMcall UnHook_Device_Service

        popa
        ret
        endp

BeginProc newSysCtl, HOOK_PROC, sav_SysCtl

        cmp     eax,1dh         ;Create_Thread message
        jz      @@1
        jmp     [sav_SysCtl]
@@1:
        call    [sav_SysCtl]
        jc      @@e
                ;after the message,edi=Ring0TCB to create
        pusha
        mov     eax,edi
        push    eax
        vmmcall Get_Cur_Thread_Handle
        pop     eax
        xchg    eax,edi

        d_msg_l '`SysControl`: Thread_Create find! TCB=#edi,owner=#eax'
        popa
        clc
@@e:
        ret
EndProc newSysCtl
```

Get_DDB

```
        PDDB    pddb;
        pddb = Get_DDB(0,"VMM     ");
        pddb = Get_DDB(1,0);
        pddb = VMM_GetDDBList();
All are same.
see also:       VMMCall VMM_GetDDBList
```

Format of .SYM

See also:
About Debug Symbol

```
98DDK   provide

        MAPSYM 6.02
        SYMLIB 1.25
        HDR             94.1.5

------predefined structures------

typedef struct tagSYMNAME {
    BYTE        length;
    char        name[1];
} SYMNAME, *PSYMNAME;

typedef struct tagSYMSYMBOL {
    WORD        offset;         //DWORD if 32bit
    SYMNAME     symName;
} SYMSYMBOL, *PSYMSYMBOL;

typedef struct tagSYMLINEHEAD {
    WORD        nextlinepos;    //00 *16h
    WORD        reserv1;        //02
    WORD        linerecpos;     //04 +SymLineHead
    WORD        f32;            //06 1:16bit 2:32bit
    WORD        cline;          //08 numlines
    SYMNAME     srcFileName;    //0a
} SYMLINEHEAD, *PSYMLINEHEAD;

typedef struct tagSYMLINE {
    WORD        Lineoffset;     //00 DWORD if 32bit
    WORD        LineNo;         //02
} SYMLINE, *PSYMLINE;

typedef struct tagSYMFILEHEADER {
    WORD        fileSize;       //00 *16h
    WORD        reserved1;      //02
    DWORD       reserved2;      //04
    WORD        absoff;         //08
    WORD        cseg;           //0a
    WORD        segpos;         //0c *16h
    BYTE        cbnamemax;      //0e
    SYMNAME     tableName;      //0f
} SYMFILEHEADER, *PSYMFILEHEADER;

typedef struct tagSYMHEADER {
    WORD        nextOffset;     //00  *10h
    WORD        numSyms;        //02  csym
    WORD        symoff;         //04  +offset SYMHEADER
```

```
    WORD        segment;        //06  loadseg
    BYTE        reserved2[6];   //08
    WORD        f32;            //0e  0:16bit 1:32bit
    WORD        linepos;        //10  *10h
    WORD        reserved3;      //12  00 ff
    SYMNAME     symName;
} SYMHEADER, *PSYMHEADER;

#define SIZEOFSYMFILEHEADER   16
#define SIZEOFSYMHEADER       21
#define SIZEOFSYMBOL           3

#define SYM_SEGMENT_NAME       0
#define SYM_SYMBOL_NAME        1
#define SYM_SEGMENT_ABS        2
#define SYM_SYMBOL_ABS         3


-----File Format---------

SYMFILEHEADER

SYMHEADER
   SYMSymbol
   SYMSymbol
   SYMSymbol
   ..
   SYMSymbol
SYMHEADER
   SYMSymbol
   SYMSymbol
   SYMSymbol
   ..
   SYMSymbol
...
SymLineHead
        SymLine
        SymLine
        SymLine
SymLineHead
        SymLine
        SymLine
        SymLine
MapVersion

----- Sample ------------
T1.cpp---

        // test for TRW's sym function

        int proc1(int i,int j,int k);

        /* compile with BC++ 5.0:
         * bcc32 /v /y /M t1.cpp t2.cpp
         */
```

```
        long L1=23455671;
        static char msg[]="this is a string";
        void main()
        {
                int i = 2;
                int j = 4;

                int k = i+j;
                j = proc1(i,j,k);
            i = j+k;
        }
```

T2.cpp---

```
        int proc1(int i,int j,int k)
        {
                int m=234;
                for( int h=1;h<5;h++ ){
                        i+=h;
                        }
                return i+j+k+m;
        }
```

MAP---

```
 Start         Length     Name                    Class
 0001:00000000 000005014H _TEXT                   CODE
 0002:00000000 00000107CH _DATA                   DATA
 0002:0000107C 000000000H _TLSCBA                 TLSCBA
 0002:0000107C 00000001EH _INIT_                  INITDATA
 0002:0000109A 000000000H _INITEND_               INITDATA
 0002:0000109A 00000000CH _EXIT_                  EXITDATA
 0002:000010A6 000000000H _EXITEND_               EXITDATA
 0002:000010A8 000000000H CONST                   CONST
 0002:000010A8 000000298H _BSS                    BSS
 0002:00001340 000000000H _BSSEND                 BSS

  Address         Publics by Name

 0001:000045E4       operator delete(void*)
 0001:000045F4       operator delete[](void*)
 0001:00001FC5       __tpdsc__[Bad_typeid]
 0001:00001FE1       __tpdsc__[Bad_cast]
....
 0001:0000009C       proc1(int,int,int)
 0001:00002088  Idle terminate()
 0001:0000007C       _main
 0001:000042F0       _malloc

....
 0001:00003B50       ___utoa
 0001:00002B54       ___vprinter
 0001:000032F4       ___write
```

```
  Address         Publics by Value

 0001:00000000  Idle __acrtused
 0001:00000074  Idle __GetExceptDLLinfo
 0001:0000007C       _main
 0001:0000009C       proc1(int,int,int)
....
 0002:00001324       __C0argc
 0002:00001328       __C0argv
 0002:0000132C       __C0environ
 0002:00001338  Idle __stkbase
 0002:0000133C  Idle __isWindows

SYM---
                     Display of File T1.SYM

000000: 97 00 00 00 00 00 00 00  12 00 02 00 02 00 23 02 ..............#.
000010: 54 31 00 00 00 00 00 00  00 00 00 00 00 00 00 00 T1..............

000020: 68 00 61 00 96 05 01 00  00 00 00 00 00 00 00 00 h.a.............
000030: 00 00 00 FF 05 5F 54 45  58 54 7C 00 05 5F 6D 61 ....._TEXT|.._ma
000040: 69 6E 9C 00 05 70 72 6F  63 31 C0 00 07 5F 6D 65 in...proc1..._me
000050: 6D 63 68 72 E0 00 07 5F  6D 65 6D 63 70 79 04 01 mchr..._memcpy..
000060: 08 5F 6D 65 6D 6D 6F 76  65 50 01 07 5F 6D 65 6D ._memmoveP.._mem
000070: 73 65 74 80 01 07 5F 73  74 72 63 6D 70 AC 01 07 set..._strcmp...
....
000510: 65 73 73 61 67 65 48 65  6C 70 65 72 A4 49 07 5F essageHelper.I._
000520: 5F 61 62 6F 72 74 B8 49  06 5F 61 62 6F 72 74 24 _abort.I._abort$
000530: 4A 05 5F 65 78 69 74 3C  4A 06 5F 5F 65 78 69 74 J._exit J.__exit
000540: 74 4A 0A 5F 5F 69 6E 69  74 77 69 6C 64 90 4C 06 tJ.__initwild.L.
000550: 5F 72 61 69 73 65 08 4D  0E 5F 5F 63 72 65 61 74 _raise.M.__creat
000560: 65 5F 73 68 6D 65 6D 10  4D 10 5F 5F 69 6E 69 74 e_shmem.M.__init
000570: 5F 65 78 69 74 5F 70 72  6F 63 BC 4D 09 5F 5F 73 _exit_proc.M.__s
000580: 74 61 72 74 75 70 24 4F  09 5F 5F 63 6C 65 61 6E tartup$O.__clean
000590: 75 70 9C 4F 0B 5F 5F 74  65 72 6D 69 6E 61 74 65 up.O.__terminate
0005A0: B0 4F 08 5F 67 65 74 64  61 74 65 DC 4F 08 5F 67 .O._getdate.O._g
0005B0: 65 74 74 69 6D 65 1A 00  22 00 2A 00 34 00 3E 00
0005C0: 49 00 53 00 5D 00 67 00  71 00 7C 00 8E 00 9E 00
0005D0: AF 00 C4 00 D7 00 EE 00  06 01 1C 01 32 01 47 01
0005E0: 5B 01 70 01 86 01 95 01  B2 01 C7 01 DE 01 EE 01
0005F0: 04 02 18 02 24 02 3C 02  56 02 62 02 6F 02 79 02
000600: 86 02 94 02 9F 02 A8 02  B1 02 BC 02 CD 02 DB 02
000610: E7 02 F4 02 FE 02 09 03  18 03 27 03 38 03 45 03
000620: 53 03 60 03 6E 03 79 03  85 03 9B 03 A6 03 B0 03
000630: BC 03 CA 03 D7 03 E5 03  F1 03 FE 03 0A 04 17 04
000640: 21 04 2D 04 3A 04 4A 04  52 04 62 04 6C 04 77 04
000650: 82 04 93 04 A3 04 B5 04  C6 04 D7 04 E5 04 FC 04
000660: 06 05 0F 05 17 05 20 05  2D 05 36 05 47 05 5A 05
000670: 66 05 72 05 80 05 8B 05  00 00 00 00 00 00 00 00

000680: 00 00 2D 00 87 02 02 00  00 00 00 00 00 00 00 00 ..-.............
000690: 00 00 00 FF 05 5F 44 41  54 41 58 00 08 5F 5F 5F .....DATAX.._____
0006A0: 69 73 44 4C 4C 59 00 08  5F 5F 5F 69 73 47 55 49 isDLLY.._____isGUI
0006B0: 62 00 0B 5F 5F 68 49 6E  73 74 61 6E 63 65 90 00 b.__hInstance..
0006C0: 23 5F 5F 5F 64 65 62 75  67 67 65 72 44 69 73 61 #___debuggerDisa
0006D0: 62 6C 65 54 65 72 6D 69  6E 61 74 65 43 61 6C 6C bleTerminateCall
```

```
0006E0: 62 61 63 6B 9C 01 0A 74  79 70 65 69 6E 66 6F 3A back...typeinfo:
....
0008B0: 79 70 65 90 12 0B 5F 5F  61 74 65 78 69 74 74 62 ype..__atexittb
0008C0: 6C 18 13 0B 5F 5F 45 78  63 52 65 67 50 74 72 1C l..__ExcRegPtr.
0008D0: 13 07 5F 5F 6F 73 65 6E  76 20 13 07 5F 5F 6F 73 ..__osenv ..__os
0008E0: 63 6D 64 24 13 08 5F 5F  43 30 61 72 67 63 28 13 cmd$..__C0argc(.
0008F0: 08 5F 5F 43 30 61 72 67  76 2C 13 0B 5F 5F 43 30 .__C0argv,..__C0
000900: 65 6E 76 69 72 6F 6E 1A  00 25 00 30 00 3E 00 64 environ..
000910: 00 71 00 7C 00 88 00 92  00 9D 00 A7 00 B4 00 C4 .q.|............
000920: 00 D1 00 E0 00 ED 00 FC  00 0C 01 1B 01 2B 01 39 .............+.9
000930: 01 46 01 54 01 6A 01 78  01 86 01 94 01 9E 01 AD .F.T.j.x........
000940: 01 B6 01 C4 01 D0 01 DE  01 F5 01 04 02 0F 02 1B ................
000950: 02 28 02 33 02 41 02 4F  02 59 02 63 02 6E 02 79 .(.3.A.O.Y.c.n.y
000960: 02 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 ................

000970: 00 00 02 06
-------------------------------------------
hdr -v -y t1.sym
----------------

t1.sym:         DOS .sym file
nextmappos abstype entryseg cabs  absoff  cseg  segpos cbnamemax name
  000970   16bit     0000   0000  000012  0002  000020     23    T1
segno nextsegpos loadseg csym  symoff  symtype linepos  name
  01    000680     0001  0061  0005b6  16bit    000000  _TEXT
  02    000000     0002  002d  000907  16bit    000000  _DATA
segment 01 symbols:
  001a  01:007c  _main
  0022  01:009c  proc1
  002a  01:00c0  _memchr
  0034  01:00e0  _memcpy
  003e  01:0104  _memmove
....
  055a  01:4dbc  __startup
  0566  01:4f24  __cleanup
  0572  01:4f9c  __terminate
  0580  01:4fb0  _getdate
  058b  01:4fdc  _gettime
segment 02 symbols:
  001a  02:0058  ___isDLL
  0025  02:0059  ___isGUI
  0030  02:0062  __hInstance
....
  024f  02:131c  __osenv
  0259  02:1320  __oscmd
  0263  02:1324  __C0argc
  026e  02:1328  __C0argv
  0279  02:132c  __C0environ
mapsym version 6.02


-------------------------------------------
trw.sym:        DOS .sym file
nextmappos abstype entryseg cabs  absoff  cseg  segpos cbnamemax name
  013360   16bit     0000   0000  000013  0003  000020     32    TRW
segno nextsegpos loadseg csym  symoff  symtype linepos  name
```

```
  01    012f30     0001  03f8  00415f  32bit    004950  PELEDATA
  02    013050     0002  0007  013003  16bit    013020  INITDAT0
  03    000000     0003  0010  01322b  16bit    013250  PCODE
segment 01 symbols:
  001d  01:00000000  c_Test1
  0029  01:0000000c  @CheckPageRange$qiul
  0042  01:00000064  @Dump_PTE$quli
  0055  01:00000188  @LinAddr_to_PageNum$qpv
  0071  01:00000198  @Test1$qv

00000000:  36 13 00 00-00 00 00 00-13 00 03 00-02 00 32 03
00000010:  54 52 57 00-00 00 00 00-00 00 00 00-00 00 00 00  TRW

00000020:  F3 12 F8 03-3F 41 01 00-00 00 00 00-00 00 01 00
00000030:  95 04 00 FF-08 50 45 4C-45 44 41 54-41 00 00 00  ? ?& ;PELEDATA
00000040:  00 07 63 5F-54 65 73 74-31 0C 00 00-00 14 40 43   & c_Test1
00000050:  68 65 63 6B-50 61 67 65-52 61 6E 67-65 24 71 69  heckPageRange$qi
00000060:  75 6C 64 00-00 00 0E 40-44 75 6D 70-5F 50 54 45  uld   Dump_PTE
00000070:  24 71 75 6C-69 88 01 00-00 17 40 4C-69 6E 41 64  $quli?  @LinAd
00000080:  64 72 5F 74-6F 5F 50 61-67 65 4E 75-6D 24 71 70  dr_to_PageNum$qp

00004950:  A6 04 00 00-16 00 01 00-3C 00 0A 73-72 63 5C 74           src\t
00004960:  32 2E 63 70-70 00 00 00-1B 00 05 00-1D 00 0A 00  2.cpp
00004970:  1E 00 0C 00-20 00 12 00-24 00 21 00-25 00 2A 00
00004980:  26 00 33 00-27 00 40 00-28 00 5B 00-2A 00 5D 00

segment 01 line info:
  nextlinepos=004a60  linerecpos=004966  cline=60  src\t2.cpp
    line 27    01:0000  src\t2.cpp
    line 29    01:0005  src\t2.cpp
    line 30    01:000a  src\t2.cpp
    line 32    01:000c  src\t2.cpp
    line 36    01:0012  src\t2.cpp
    line 37    01:0021  src\t2.cpp
    line 38    01:002a  src\t2.cpp

0000DA00:  B4 0D 00 00-17 00 02 00-2F 00 0B 73-72 63 5C 69           src\i
0000DA10:  64 74 2E 61-73 6D 00 58-FF 00 00 17-00 59 FF 00  dt.asm
0000DA20:  00 18 00 5E-FF 00 00 19-00 65 FF 00-00 1B 00 6A
0000DA30:  FF 00 00 1C-00 6F FF 00-00 1F 00 75-FF 00 00 20

  nextlinepos=00db40  linerecpos=00da17  cline=47  src\idt.asm
    line 23    01:0000ff58  src\idt.asm
    line 24    01:0000ff59  src\idt.asm
    line 25    01:0000ff5e  src\idt.asm
```

HDR

Prints header information from executable files.

Syntax

hdr [*options*] *filename*

Parameters

-d Prints data relocation records.
-f Prepends the file name on every output line.
-h Prints the file header (default).
-m Print only memory segment table information.
- Prints specified tables.
j[*options*]

d Prints debug table.
e Prints entry table.
f Prints extended attributes.
i Prints import module table.
I Prints import procedure table.
r Prints resident name table.
R Prints resource table.

-p Prints file seek positions.
-R Same as **-r**, but also follows relocation chains.
-r Prints both text and data relocations.
-S Prints the file segment table with a header.
-s Prints the symbol table.
-T Prints the file type.
-t Prints the text relocation records.
-y Dumps DOS .SYM file.
-v Prints everything (verbose).

-? Prints usage summary.

filename Specifies the filename

SYMLIB

You can use SYMLIB to manage symbol libraries (.SYM files). With SYMLIB you can create new symbol files, add symbols, remove symbols, and update symbols.

Syntax

symlib symlib1{+*symlib2*|-{*symlib2*|*symbol*}| *symlib2*|-+*symlib2*}

Parameters

symlib1	Specifies the symbol library (.SYM file) to be created or modified.
symlib2	Specifies the symbol library (.SYM file) to be added to or updated in *symlib1*
symbol	Specifies the symbol to be removed from *symlib1*

Note There should be no space between the '+' or '-' and the following argument.

Examples

The following example adds **vmm.sym** to **win386.sym**.

```
symlib win386.sym vmm.sym
```

or

```
symlib win386.sym+vmm.sym
```

The following example updates the symbols in win386.sym with those in vmm.sym.

```
symlib win386.sym-+vmm.sym
```

The following example deletes the symbol **vmm** from win386.sym.

```
symlib win386.sym-vmm.sym
```

Creating Symbol Files

Symbol files provide the information the debugger needs to display functions, structures, variables, and absolute symbols by name rather than number. To prepare symbol files, perform the following steps:

1. Compile or assemble your source files, using the appropriate command-line option to generate object files with line-number information. For more information about compiler and assembler options, see the documentation that accompanied your compiler and assembler.
2. Link the compiled code with the standard libraries (as needed), using the appropriate linker option to create a symbol map (.MAP) file that includes PUBLIC symbols. You may also want to use the linker option for display of line-number information. For more information about linker options, see the documentation that accompanied your linker.
3. Run the Microsoft Symbol File Generator (MAPSYM.EXE) to create a symbol file for symbolic debugging. MAPSYM converts the contents of your application's symbol map (.MAP) file into a form suitable for loading with the debugger; then MAPSYM copies the result to a symbol (.SYM) file.

Following is the command-line syntax for MAPSYM:

mapsym [**/l**][**/n**] *mapfilename*

/l	Directs MAPSYM to display information on the screen about the conversion. The information includes the names of groups defined in the application, the application start address, the number of segments, and the number of symbols per segment.
/n	Directs MAPSYM to ignore line-number information in the map file. The resulting symbol file contains no line-number information.
mapfilename	Specifies the filename for a symbol map file that was created during linking. If you do not give a filename extension, .MAP is assumed. If you do not give a full path, the current directory and drive are assumed. MAPSYM creates a new symbol file having the same name as the map file but with the .SYM extension.

In the following example, MAPSYM uses the symbol information in FILE.MAP to create FILE.SYM in the current directory on the current drive:

```
mapsym /l file.map
```

Information about the conversion is sent to the screen. MAPSYM always places the new symbol file in the current directory on the current drive. MAPSYM can process up to 10,000 symbols for each segment in the application and up to 1024 segments. If you have many components to debug, you can combine multiple symbol files into a single file by using the Symbol File Librarian (SYMLIB.EXE). This creates a symbol library file and lets you add, remove, or replace .SYM files in it.

```
PE on INTERNET:

        http://www.microsoft.com/win32dev/base/pefile.htm
        http://www.eccentrica.org/Mammon/pefile.html

typedef struct _IMAGE_NT_HEADERS {
    DWORD Signature;                //00    'PE'

//    IMAGE_FILE_HEADER FileHeader;
    WORD    Machine;                //04
    WORD    NumberOfSections;       //06
    DWORD   TimeDateStamp;          //08
    DWORD   PointerToSymbolTable;   //0c
    DWORD   NumberOfSymbols;        //10
    WORD    SizeOfOptionalHeader;   //14
    WORD    Characteristics;        //16

//    IMAGE_OPTIONAL_HEADER OptionalHeader;
    WORD    Magic;                      //+18
    BYTE    MajorLinkerVersion;         //+1a
    BYTE    MinorLinkerVersion;
    DWORD   SizeOfCode;                 //+1c
    DWORD   SizeOfInitializedData;      //+20
    DWORD   SizeOfUninitializedData;
    DWORD   AddressOfEntryPoint;
    DWORD   BaseOfCode;                 //+2c
    DWORD   BaseOfData;                 //+30
    DWORD   ImageBase;                  //+34
    DWORD   SectionAlignment;
    DWORD   FileAlignment;
    WORD    MajorOperatingSystemVersion; //+40
    WORD    MinorOperatingSystemVersion;
    WORD    MajorImageVersion;
    WORD    MinorImageVersion;
    WORD    MajorSubsystemVersion;      //+48
    WORD    MinorSubsystemVersion;
    DWORD   Win32VersionValue;          //+4c
    DWORD   SizeOfImage;                //+50
    DWORD   SizeOfHeaders;
    DWORD   CheckSum;                   //+58
    WORD    Subsystem;                  //+5c
    WORD    DllCharacteristics;
    DWORD   SizeOfStackReserve;         //+60
    DWORD   SizeOfStackCommit;
    DWORD   SizeOfHeapReserve;
    DWORD   SizeOfHeapCommit;
    DWORD   LoaderFlags;                //+70
    DWORD   NumberOfRvaAndSizes;        //+74

//    IMAGE_DATA_DIRECTORY DataDirectory[IMAGE_NUMBEROF_DIRECTORY_ENTRIES];
    DWORD   EXPORT_VirtualAddress;      //+78
    DWORD   EXPORT_Size;                //+7c

    DWORD   IMPORT_VirtualAddress;      //+80
    DWORD   IMPORT_Size;                //+84

    DWORD   RESORC_VirtualAddress;      //+88
    DWORD   RESORC_Size;                //+8c

    DWORD   EXCEPT_VirtualAddress;      //+90
    DWORD   EXCEPT_Size;                //+94
```

```
    DWORD   SECURT_VirtualAddress;       //+98
    DWORD   SECURT_Size;                 //+9c

} PE;

#define IMAGE_DIRECTORY_ENTRY_EXPORT         0   // Export Directory
#define IMAGE_DIRECTORY_ENTRY_IMPORT         1   // Import Directory
#define IMAGE_DIRECTORY_ENTRY_RESOURCE       2   // Resource Directory
#define IMAGE_DIRECTORY_ENTRY_EXCEPTION      3   // Exception Directory
#define IMAGE_DIRECTORY_ENTRY_SECURITY       4   // Security Directory
#define IMAGE_DIRECTORY_ENTRY_BASERELOC      5   // Base Relocation Table
#define IMAGE_DIRECTORY_ENTRY_DEBUG          6   // Debug Directory
#define IMAGE_DIRECTORY_ENTRY_COPYRIGHT      7   // Description String
#define IMAGE_DIRECTORY_ENTRY_GLOBALPTR      8   // Machine Value (MIPS GP)
#define IMAGE_DIRECTORY_ENTRY_TLS            9   // TLS Directory
#define IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG   10   // Load Configuration Directory
#define IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT  11   // Bound Import Directory in headers
#define IMAGE_DIRECTORY_ENTRY_IAT           12   // Import Address Table
```

```
        NE exe file format
See also:
        NE overall
        neexe1.htm
        neexe.txt

/* NE.H -- New Executable (segmented executable) format */

#define ENEWHDR     0x003CL         /* offset of new EXE header */
#define EMAGIC      0x5A4D          /* old EXE magic id:  'MZ'  */
#define NEMAGIC     0x454E          /* new EXE magic id:  'NE'  */
#define LEMAGIC         0x454C                  /* linear executable */
#define LXMAGIC         0x584C                  /* IBM OS/2 2.0 linear executable */
#define PEMAGIC         0x4550                  /* NT portable executable */
#define W3MAGIC         0x3357                  /* WIN386.EXE signature */
#define SZMAGIC         0x5A53                  /* SZ -- compressed file */

#define DLL_FLAG    0x8000
#define APPLOAD         0x0800                  /* self-loading Win app */
#define NEMATH          0x0080
#define NEI386          0x0040
#define NEI286          0x0020
#define NEI086          0x0010

#define NSTYPE          0x0007
#define NSCODE          0x0000
#define NSDATA          0x0001
#define NSRELOC         0x0100
#define NS32BIT         0x2000

#pragma pack(1)

typedef struct
{
    unsigned    ne_magic;       //00 ID='NE'
    char        ne_ver;                 //02 Linker major version
    char        ne_rev;                 //03 Linker minor version
    unsigned    ne_enttab;              //04 NE offset of entry table
    unsigned    ne_cbenttab;    //06 #bytes in entry table
    long        ne_crc;                 //08 File load CRC (0 in Borland's TPW)
    unsigned    ne_flags;               //0c DLL vs. program, APPLOAD, etc.
    unsigned    ne_autodata;    //0e logical segment# (LSN) of DGROUP
    unsigned    ne_heap;                //10 Initial local heap size
    unsigned    ne_stack;               //12 Initial stack size
    void far *  ne_csip;                //14 initial entry point (LSN:ofs)
    void far *  ne_sssp;                //18 Initial stack pointer (SS:SP)
    unsigned    ne_cseg;                //1c number of segments
    unsigned    ne_cmod;                //1e number of WORDs in module ref tbl
    unsigned    ne_cbnrestab;   //20 #bytes in nonres name table
    unsigned    ne_segtab;              //22 NE offset of segment table
    unsigned    ne_rsrctab;     //24 *aligned* NE offset of resource table
    unsigned    ne_restab;      //26 NE offset of resident name table
    unsigned    ne_modref;              //28 NE offset of module reference table
    unsigned    ne_impname;             //2a NE offset of imported names table
    long        ne_nonrestab;   //2c file (NOT NE) offset of nonres name tbl
        unsigned        ne_cmovent;             //30 Count of moveable entry point listed in entr
        unsigned        ne_align;               //32 segment align shift count: segs, rsrc
        unsigned        ne_cres;                //34 Number of resource table entries
        unsigned char   ne_exetyp;      //36 target OS; 1=OS/2 2=Windows
        unsigned char ne_flagsothers; //37 Other OS/2 EXE flags, bitmapped
        unsigned        ne_fastload;    //38 Offset to return thunks or start of gangload
        unsigned        ne_lenfastload; //3a offset to segment reference thunks or length
                                of gangload area.
        unsigned        ne_reserved;    //3c Minimum code swap area size
        unsigned char   ne_expwinvermin;        //3e expected Windows version
        unsigned char   ne_expwinvermaj;
} NEWEXE;

typedef struct {
        unsigned short ns_sector;
        unsigned short ns_cbseg;
```

```
        unsigned short ns_flags;
        unsigned short ns_minalloc;
        } SEG;

typedef struct {
        unsigned char ep_flags;
        unsigned short ep_int_3f;
        unsigned char ep_segnum;
        unsigned short ep_offset;
        } ENTRY_MOVEABLE;

typedef struct {
        unsigned char ep_flags;
        unsigned short ep_offset;
        } ENTRY_FIXED;

typedef struct {
        union {
                ENTRY_MOVEABLE ep_moveable;
                ENTRY_FIXED ep_fixed;
                } ep_union;
        } ENTRY_POINT;

typedef struct {
        unsigned char eb_nentry;
        unsigned char eb_segnum;
        } BUNDLE_HEADER;

typedef struct {
        BUNDLE_HEADER eb_hdr;
        ENTRY_POINT eb_epoint [1] ;
        } ENTRY_BUNDLE;

typedef struct {
        char nr_stype;
        char nr_flags;
        unsigned short nr_soff;
        union {
                struct {
                        unsigned char nr_segno;
                        unsigned char nr_res;
                        unsigned short nr_entry;
                        } nr_intref;
                struct {
                        unsigned short nr_mod;
                        unsigned short nr_proc;
                        } nr_import;
                struct {
                        unsigned short nr_ostype;
                        unsigned short nr_osres;
                        } nr_osfix;
                } nr_union;
        } RELOC_ITEM;

#define NRSTYP          0x0f
#define NRSBYT          0x00
#define NRSSEG          0x02
#define NRSPTR          0x03
#define NRSOFF          0x05
#define NRSPTR48        0x0b
#define NRSOFF32        0x0d

#define NRADD           0x04
#define NRRTYP          0x03
#define NRRINT          0x00
#define NRRORD          0x01
#define NRRNAM          0x02
#define NRROSF          0x03

#define MOVEABLE        0xFF
```

```
-------------------------------------------

The NE EXE files are the new exe files used by windows and OS/2 executables.
They contain a small MZ EXE which prints "This program requires Microsoft
Windows" or something similar but Some files contain both DOS and Windows
versions of the executable. The position of the new EXE header can be found
in the old exe header - see the MZ EXE topic for further information. All
offsets within this header are from the start of the header if not noted
otherwise.

OFFSET              Count TYPE   Description
0000h                   2 char   ID='NE'
0002h                   1 byte   Linker major version
0003h                   1 byte   Linker minor version
0004h                   1 word   Offset of entry table (see below)
0006h                   1 word   Length of entry table in bytes
0008h                   1 dword  File load CRC (0 in Borland's TPW)
000Ch                   1 byte   Program flags, bitmapped :
                                 0-1 - DGroup type :
                                       0 - none
                                       1 - single shared
                                       2 - multiple
                                       3 - (null)
                                   2 - Global initialization
                                   3 - Protected mode only
                                   4 - 8086 instructions
                                   5 - 80286 instructions
                                   6 - 80386 instructions
                                   7 - 80x87 instructions
000Dh                   1 byte   Application flags, bitmapped
                                 0-2 - Application type
                                       1 - Full screen (not aware of
                                           Windows/P.M. API)
                                       2 - Compatible with Windows/P.M. API
                                       3 - Uses Windows/P.M. API
                                   3 - OS/2 family application
                                   4 - reserved?
                                   5 - Errors in image/executable
                                   6 - "non-conforming program" whatever
                                   7 - DLL or driver (SS:SP info invalid, CS:IP
                                       points at FAR init routine called with
                                       AX=module handle which returns AX=0000h
                                       on failure, AX nonzero on successful
                                       initialization)
000Eh                   1 byte   Auto data segment index
0010h                   1 word   Initial local heap size
0012h                   1 word   Initial stack size
0014h                   1 dword  Entry point (CS:IP),
                                 CS is index into segment table
0018h                   1 dword  Initial stack pointer (SS:SP)
                                 SS is index into segment table
001Ch                   1 word   Segment count
001Eh                   1 word   Module reference count
0020h                   1 word   Size of nonresident names table in bytes
0022h                   1 word   Offset of segment table (see below)
0024h                   1 word   Offset of resource table
0026h                   1 word   Offset of resident names table
0028h                   1 word   Offset of module reference table
002Ah                   1 word   Offset of imported names table
                                 (array of counted strings, terminated with a
                                  string of length 00h)
002Ch                   1 dword  Offset from start of file to nonresident
                                 names table
0030h                   1 word   Count of moveable entry point listed in
                                 entry table
0032h                   1 word   File alignment size shift count
                                  0 is equivalent to 9 (default 512-byte pages)
0034h                   1 word   Number of resource table entries
0036h                   1 byte   Target operating system
                                  0 - unknown
                                  1 - OS/2
                                  2 - Windows
```

```
                                3 - European MS-DOS 4.x
                                4 - Windows 386
                                5 - BOSS (Borland Operating System Services)
0037h                  1 byte   Other OS/2 EXE flags, bitmapped
                                0 - Long filename support
                                1 - 2.x protected mode
                                2 - 2.x proportional fonts
                                3 - Executable has gangload area
0038h                  1 word   Offset to return thunks or start of gangload
                                area - whatever that means.
003Ah                  1 word   offset to segment reference thunks or length
                                of gangload area.
003Ch                  1 word   Minimum code swap area size
003Eh                  2 byte   Expected Windows version (minor version first)

EXTENSION:DLL,EXE,FOT
OCCURENCES:PC
PROGRAMS:
REFERENCE:Windows 3.1 SDK Programmer's Reference, Vol 4.
SEE ALSO:EXE,MZ EXE
```

```
Export functions of Krnl386.exe
NEdump.exe, sort

   1: FATALEXIT
   2: EXITKERNEL
   3: GETVERSION
   4: LOCALINIT
   5: LOCALALLOC
   6: LOCALREALLOC
   7: LOCALFREE
   8: LOCALLOCK
   9: LOCALUNLOCK
  10: LOCALSIZE
  11: LOCALHANDLE
  12: LOCALFLAGS
  13: LOCALCOMPACT
  14: LOCALNOTIFY
  15: GLOBALALLOC
  16: GLOBALREALLOC
  17: GLOBALFREE
  18: GLOBALLOCK
  19: GLOBALUNLOCK
  20: GLOBALSIZE
  21: GLOBALHANDLE
  22: GLOBALFLAGS
  23: LOCKSEGMENT
  24: UNLOCKSEGMENT
  25: GLOBALCOMPACT
  26: GLOBALFREEALL
  27: GETMODULENAME
  28: GLOBALMASTERHANDLE
  29: YIELD
  30: WAITEVENT
  31: POSTEVENT
  32: SETPRIORITY
  33: LOCKCURRENTTASK
KERNEL . 34     SETTASKQUEUE
  35: GETTASKQUEUE
  36: GETCURRENTTASK
  37: GETCURRENTPDB
KERNEL . 38     SETTASKSIGNALPROC
;; in 2.1, KERNEL.39 was SETTASKSWITCHPROC
;; in 2.1, KERNEL.40 was SETTASKINTERCHANGE
  41: ENABLEDOS
  42: DISABLEDOS
;; in 2.1, KERNEL.43 was ISSCREENGRAB
```

```
;; in 2.1, KERNEL.44 was BUILDPDB
  45: LOADMODULE
  46: FREEMODULE
  47: GETMODULEHANDLE
  48: GETMODULEUSAGE
  49: GETMODULEFILENAME
  50: GETPROCADDRESS
  51: MAKEPROCINSTANCE
  52: FREEPROCINSTANCE
  53: CALLPROCINSTANCE
  54: GETINSTANCEDATA
  55: CATCH
  56: THROW
  57: GETPROFILEINT
  58: GETPROFILESTRING
  59: WRITEPROFILESTRING
  60: FINDRESOURCE
  61: LOADRESOURCE
  62: LOCKRESOURCE
  63: FREERESOURCE
  64: ACCESSRESOURCE
  65: SIZEOFRESOURCE
  66: ALLOCRESOURCE
  67: SETRESOURCEHANDLER
  68: INITATOMTABLE
  69: FINDATOM
  70: ADDATOM
  71: DELETEATOM
  72: GETATOMNAME
  73: GETATOMHANDLE
  74: OPENFILE
  75: OPENPATHNAME
  76: DELETEPATHNAME
KERNEL . 77 RESERVED1               ;; AnsiNext
KERNEL . 78 RESERVED2               ;; AnsiPrev
KERNEL . 79 RESERVED3               ;; AnsiUpper
KERNEL . 80 RESERVED4               ;; AnsiLower
  81: _LCLOSE
  82: _LREAD
  83: _LCREAT
  84: _LLSEEK
  85: _LOPEN
  86: _LWRITE
KERNEL . 87 RESERVED5               ;; lstrcmp
  88: LSTRCPY
  89: LSTRCAT
  90: LSTRLEN
  91: INITTASK
```

```
  92: GETTEMPDRIVE
  93: GETCODEHANDLE
  94: DEFINEHANDLETABLE
  95: LOADLIBRARY
  96: FREELIBRARY
  97: GETTEMPFILENAME
  98: GETLASTDISKCHANGE
  99: GETLPERRMODE
 100: VALIDATECODESEGMENTS
 101: NOHOOKDOSCALL
 102: DOS3CALL
 103: NETBIOSCALL
 104: GETCODEINFO
 105: GETEXEVERSION
 106: SETSWAPAREASIZE
 107: SETERRORMODE
 108: SWITCHSTACKTO
 109: SWITCHSTACKBACK
 110: PATCHCODEHANDLE
 111: GLOBALWIRE
 112: GLOBALUNWIRE
 113: __AHSHIFT
 114: __AHINCR
 115: OUTPUTDEBUGSTRING
 116: INITLIB
 117: OLDYIELD
KERNEL . 118    GETTASKQUEUEDS
KERNEL . 119    GETTASKQUEUEES
 120: UNDEFDYNLINK
 121: LOCALSHRINK
 122: ISTASKLOCKED
 123: KBDRST
 124: ENABLEKERNEL
 125: DISABLEKERNEL
 126: MEMORYFREED
 127: GETPRIVATEPROFILEINT
 128: GETPRIVATEPROFILESTRING
 129: WRITEPRIVATEPROFILESTRING
 130: FILECDR
 131: GETDOSENVIRONMENT
 132: GETWINFLAGS
 133: GETEXEPTR
 134: GETWINDOWSDIRECTORY
 135: GETSYSTEMDIRECTORY
 136: GETDRIVETYPE
 137: FATALAPPEXIT
 138: GETHEAPSPACES
KERNEL . 139    DOSIGNAL
```

```
KERNEL . 140    SETSIGHANDLER
KERNEL . 141    INITTASK1
 142: GETPROFILESECTIONNAMES
 143: GETPRIVATEPROFILESECTIONNAMES
 144: CREATEDIRECTORY
 145: REMOVEDIRECTORY
 146: DELETEFILE
 147: SETLASTERROR
 148: GETLASTERROR
 149: GETVERSIONEX
 150: DIRECTEDYIELD
KERNEL . 151    WINOLDAPCALL
 152: GETNUMTASKS

 154: GLOBALNOTIFY
 155: GETTASKDS
 156: LIMITEMSPAGES
 157: GETCURPID
 158: ISWINOLDAPTASK
 159: GLOBALHANDLENORIP
KERNEL . 160    EMSCOPY
 161: LOCALCOUNTFREE
 162: LOCALHEAPSIZE
 163: GLOBALLRUOLDEST
 164: GLOBALLRUNEWEST
 165: A20PROC
 166: WINEXEC
 167: GETEXPWINVER
 168: DIRECTRESALLOC
 169: GETFREESPACE
 170: ALLOCCSTODSALIAS
 171: ALLOCDSTOCSALIAS
 172: ALLOCALIAS
 173: __ROMBIOS
 174: __A000H
 175: ALLOCSELECTOR
 176: FREESELECTOR
 177: PRESTOCHANGOSELECTOR
 178: __WINFLAGS
 179: __D000H
 180: LONGPTRADD
 181: __B000H
 182: __B800H
 183: __0000H
 184: GLOBALDOSALLOC
 185: GLOBALDOSFREE
 186: GETSELECTORBASE
 187: SETSELECTORBASE
```

```
188: GETSELECTORLIMIT
189: SETSELECTORLIMIT
190: __E000H
191: GLOBALPAGELOCK
192: GLOBALPAGEUNLOCK
193: __0040H
194: __F000H
195: __C000H
196: SELECTORACCESSRIGHTS
197: GLOBALFIX
198: GLOBALUNFIX
199: SETHANDLECOUNT
200: VALIDATEFREESPACES
201: REPLACEINST
202: REGISTERPTRACE
203: DEBUGBREAK
204: SWAPRECORDING
205: CVWBREAK
206: ALLOCSELECTORARRAY
207: ISDBCSLEADBYTE
208: K208
209: K209
210: K210
211: K211

213: K213
214: K214
215: K215
216: REGENUMKEY
217: REGOPENKEY
218: REGCREATEKEY
219: REGDELETEKEY
220: REGCLOSEKEY
221: REGSETVALUE
222: REGDELETEVALUE
223: REGENUMVALUE
224: REGQUERYVALUE
225: REGQUERYVALUEEX
226: REGSETVALUEEX
227: REGFLUSHKEY
228: K228
229: K229
230: GLOBALSMARTPAGELOCK
231: GLOBALSMARTPAGEUNLOCK
232: REGLOADKEY
233: REGUNLOADKEY
234: REGSAVEKEY
235: INVALIDATENLSCACHE
```

```
 236: GETPRODUCTNAME
 237: K237

 310: LOCALHANDLEDELTA
 311: GETSETKERNELDOSPROC
KERNEL . 313    GETLASTCRITICALERROR
 314: DEBUGDEFINESEGMENT
 315: WRITEOUTPROFILES
KERNEL . 316    GETFREEMEMINFO

 318: FATALEXITHOOK
 319: FLUSHCACHEDFILEHANDLE
 320: ISTASK
KERNEL . 321    PRESTOCHANGOSELECTOR

 323: ISROMMODULE
 324: LOGERROR
 325: LOGPARAMERROR
 326: ISROMFILE
KERNEL . 327    K327                            HandleParamError
 328: _DEBUGOUTPUT
KERNEL . 329    K329                            DebugFillBuffer

 332: THHOOK

 334: ISBADREADPTR
 335: ISBADWRITEPTR
 336: ISBADCODEPTR
 337: ISBADSTRINGPTR
 338: HASGPHANDLER
KERNEL . 339    DIAGQUERY
KERNEL . 340    DIAGOUTPUT
 341: TOOLHELPHOOK
KERNEL . 342    __GP
KERNEL . 343    REGISTERWINOLDAPHOOK
KERNEL . 344    GETWINOLDAPHOOKS
 345: ISSHAREDSELECTOR
 346: ISBADHUGEREADPTR
 347: ISBADHUGEWRITEPTR
 348: HMEMCPY
 349: _HREAD
 350: _HWRITE
 351: BUNNY_351
 352: LSTRCATN
 353: LSTRCPYN
 354: GETAPPCOMPATFLAGS
 355: GETWINDEBUGINFO
 356: SETWINDEBUGINFO
```

```
 360: OPENFILEEX
 361: PIGLET_361

 406: WRITEPRIVATEPROFILESTRUCT
 407: GETPRIVATEPROFILESTRUCT

 411: GETCURRENTDIRECTORY
 412: SETCURRENTDIRECTORY
 413: FINDFIRSTFILE
 414: FINDNEXTFILE
 415: FINDCLOSE
 416: WRITEPRIVATEPROFILESECTION
 417: WRITEPROFILESECTION
 418: GETPRIVATEPROFILESECTION
 419: GETPROFILESECTION
 420: GETFILEATTRIBUTES
 421: SETFILEATTRIBUTES
 422: GETDISKFREESPACE
 432: FILETIMETOLOCALFILETIME

 491: REGISTERSERVICEPROCESS

 513: LOADLIBRARYEX32W
 514: FREELIBRARY32W
 515: GETPROCADDRESS32W
 516: GETVDMPOINTER32W
 517: CALLPROC32W
 518: _CALLPROCEX32W

 627: ISBADFLATREADWRITEPTR

undoc:
        625:    Get_hMsgQueue

        C16ThkSL01      = KERNEL.631
        ThunkConnect16  = KERNEL.651
```

```
Exports from KERNEL32.dll

ordinal    name

     32    AddAtomA
     66    AddAtomW
     67    AllocConsole
     68    AllocLSCallback
     69    AllocSLCallback
     6A    AreFileApisANSI
     6B    BackupRead
     6C    BackupSeek
     6D    BackupWrite
     6E    Beep
     6F    BeginUpdateResourceA
     70    BeginUpdateResourceW
     71    BuildCommDCBA
     72    BuildCommDCBAndTimeoutsA
     73    BuildCommDCBAndTimeoutsW
     74    BuildCommDCBW
     75    CallNamedPipeA
     76    CallNamedPipeW
     77    Callback12
     78    Callback16
     79    Callback20
     7A    Callback24
     7B    Callback28
     7C    Callback32
     7D    Callback36
     81    Callback4
     7E    Callback40
     7F    Callback44
     80    Callback48
     82    Callback52
     83    Callback56
     84    Callback60
     85    Callback64
     86    Callback8
     87    ClearCommBreak
     88    ClearCommError
     89    CloseHandle
     8A    CloseProfileUserMapping
     8B    CloseSystemHandle
     8C    CommConfigDialogA
     8D    CommConfigDialogW
     8E    CompareFileTime
     8F    CompareStringA
     90    CompareStringW
     91    ConnectNamedPipe
     92    ContinueDebugEvent
     93    ConvertDefaultLocale
     94    ConvertToGlobalHandle
     95    CopyFileA
     96    CopyFileW
     97    CreateConsoleScreenBuffer
```

```
    98    CreateDirectoryA
    99    CreateDirectoryExA
    9A    CreateDirectoryExW
    9B    CreateDirectoryW
    9C    CreateEventA
    9D    CreateEventW
    9E    CreateFileA                    go
    9F    CreateFileMappingA
    A0    CreateFileMappingW
    A1    CreateFileW
    A2    CreateIoCompletionPort
    A3    CreateKernelThread
    A4    CreateMailslotA
    A5    CreateMailslotW
    A6    CreateMutexA
    A7    CreateMutexW
    A8    CreateNamedPipeA
    A9    CreateNamedPipeW
    AA    CreatePipe
    AB    CreateProcessA
    AC    CreateProcessW
    AD    CreateRemoteThread
    AE    CreateSemaphoreA
    AF    CreateSemaphoreW
    B0    CreateSocketHandle
    B1    CreateTapePartition
    B2    CreateThread
    B3    CreateToolhelp32Snapshot             go
    B4    DebugActiveProcess
    B5    DebugBreak
    B6    DefineDosDeviceA
    B7    DefineDosDeviceW
    B8    DeleteAtom
    B9    DeleteCriticalSection
    BA    DeleteFileA
    BB    DeleteFileW
    BC    DeviceIoControl
    BD    DisableThreadLibraryCalls
    BE    DisconnectNamedPipe
    BF    DosDateTimeToFileTime
    C0    DuplicateHandle
    C1    EndUpdateResourceA
    C2    EndUpdateResourceW
    C3    EnterCriticalSection
    C4    EnumCalendarInfoA
    C5    EnumCalendarInfoW
    C6    EnumDateFormatsA
    C7    EnumDateFormatsW
    C8    EnumResourceLanguagesA
    C9    EnumResourceLanguagesW
    CA    EnumResourceNamesA
    CB    EnumResourceNamesW
    CC    EnumResourceTypesA
    CD    EnumResourceTypesW
    CE    EnumSystemCodePagesA
```

```
 CF    EnumSystemCodePagesW
 D0    EnumSystemLocalesA
 D1    EnumSystemLocalesW
 D2    EnumTimeFormatsA
 D3    EnumTimeFormatsW
 D4    EraseTape
 D5    EscapeCommFunction
 D6    ExitProcess
 D7    ExitThread
 D8    ExpandEnvironmentStringsA
 D9    ExpandEnvironmentStringsW
 DA    FT_Exit0
 DB    FT_Exit12
 DC    FT_Exit16
 DD    FT_Exit20
 DE    FT_Exit24
 DF    FT_Exit28
 E0    FT_Exit32
 E1    FT_Exit36
 E2    FT_Exit4
 E3    FT_Exit40
 E4    FT_Exit44
 E5    FT_Exit48
 E6    FT_Exit52
 E7    FT_Exit56
 E8    FT_Exit8
 E9    FT_Prolog            go
 EA    FT_Thunk
 EB    FatalAppExitA
 EC    FatalAppExitW
 ED    FatalExit
 EE    FileTimeToDosDateTime
 EF    FileTimeToLocalFileTime
 F0    FileTimeToSystemTime
 F1    FillConsoleOutputAttribute
 F2    FillConsoleOutputCharacterA
 F3    FillConsoleOutputCharacterW
 F4    FindAtomA
 F5    FindAtomW
 F6    FindClose
 F7    FindCloseChangeNotification
 F8    FindFirstChangeNotificationA
 F9    FindFirstChangeNotificationW
 FA    FindFirstFileA
 FB    FindFirstFileW
 FC    FindNextChangeNotification
 FD    FindNextFileA
 FE    FindNextFileW
 FF    FindResourceA
100    FindResourceExA
101    FindResourceExW
102    FindResourceW
103    FlushConsoleInputBuffer
104    FlushFileBuffers
105    FlushInstructionCache
```

```
  106    FlushViewOfFile
  107    FoldStringA
  108    FoldStringW
  109    FormatMessageA
  10A    FormatMessageW
  10B    FreeConsole
  10C    FreeEnvironmentStringsA
  10D    FreeEnvironmentStringsW
  10E    FreeLSCallback
  10F    FreeLibrary
  110    FreeLibraryAndExitThread
  111    FreeResource
  112    FreeSLCallback
  113    GenerateConsoleCtrlEvent
  114    GetACP
  115    GetAtomNameA
  116    GetAtomNameW
  117    GetBinaryType
  118    GetBinaryTypeA
  119    GetBinaryTypeW
  11A    GetCPInfo
  11B    GetCommConfig
  11C    GetCommMask
  11D    GetCommModemStatus
  11E    GetCommProperties
  11F    GetCommState
  120    GetCommTimeouts
  121    GetCommandLineA
  122    GetCommandLineW
  123    GetCompressedFileSizeA
  124    GetCompressedFileSizeW
  125    GetComputerNameA
  126    GetComputerNameW
  127    GetConsoleCP
  128    GetConsoleCursorInfo
  129    GetConsoleMode
  12A    GetConsoleOutputCP
  12B    GetConsoleScreenBufferInfo
  12C    GetConsoleTitleA
  12D    GetConsoleTitleW
  12E    GetCurrencyFormatA
  12F    GetCurrencyFormatW
  130    GetCurrentDirectoryA
  131    GetCurrentDirectoryW
  132    GetCurrentProcess
  133    GetCurrentProcessId          go
  134    GetCurrentThread
  135    GetCurrentThreadId
  136    GetDateFormatA
  137    GetDateFormatW
  138    GetDaylightFlag
  139    GetDefaultCommConfigA
  13A    GetDefaultCommConfigW
  13B    GetDiskFreeSpaceA
  13C    GetDiskFreeSpaceW
```

```
    13D    GetDriveTypeA
    13E    GetDriveTypeW
    13F    GetEnvironmentStrings
    140    GetEnvironmentStringsA
    141    GetEnvironmentStringsW
    142    GetEnvironmentVariableA
    143    GetEnvironmentVariableW
    144    GetErrorMode
    145    GetExitCodeProcess
    146    GetExitCodeThread
    147    GetFileAttributesA
    148    GetFileAttributesW
    149    GetFileInformationByHandle
    14A    GetFileSize
    14B    GetFileTime
    14C    GetFileType
    14D    GetFullPathNameA
    14E    GetFullPathNameW
    14F    GetHandleContext
    150    GetHandleInformation
    151    GetLSCallbackTarget
    152    GetLSCallbackTemplate
    153    GetLargestConsoleWindowSize
    154    GetLastError
    155    GetLocalTime
    156    GetLocaleInfoA
    157    GetLocaleInfoW
    158    GetLogicalDriveStringsA
    159    GetLogicalDriveStringsW
    15A    GetLogicalDrives
    15B    GetMailslotInfo
    15C    GetModuleFileNameA
    15D    GetModuleFileNameW
    15E    GetModuleHandleA                      ret edx=?
    15F    GetModuleHandleW
    160    GetNamedPipeHandleStateA
    161    GetNamedPipeHandleStateW
    162    GetNamedPipeInfo
    163    GetNumberFormatA
    164    GetNumberFormatW
    165    GetNumberOfConsoleInputEvents
    166    GetNumberOfConsoleMouseButtons
    167    GetOEMCP
    168    GetOverlappedResult
    169    GetPriorityClass
    16A    GetPrivateProfileIntA
    16B    GetPrivateProfileIntW
    16C    GetPrivateProfileSectionA
    16D    GetPrivateProfileSectionNamesA
    16E    GetPrivateProfileSectionNamesW
    16F    GetPrivateProfileSectionW
    170    GetPrivateProfileStringA
    171    GetPrivateProfileStringW
    172    GetPrivateProfileStructA
    173    GetPrivateProfileStructW
```

```
   174    GetProcAddress                          go
   175    GetProcessAffinityMask
   176    GetProcessFlags
   177    GetProcessHeap
   178    GetProcessHeaps
   179    GetProcessShutdownParameters
   17A    GetProcessTimes
   17B    GetProcessVersion
   17C    GetProcessWorkingSetSize
   17D    GetProductName
   17E    GetProfileIntA
   17F    GetProfileIntW
   180    GetProfileSectionA
   181    GetProfileSectionW
   182    GetProfileStringA
   183    GetProfileStringW
   184    GetQueuedCompletionStatus
   185    GetSLCallbackTarget
   186    GetSLCallbackTemplate
   187    GetShortPathNameA
   188    GetShortPathNameW
   189    GetStartupInfoA
   18A    GetStartupInfoW
   18B    GetStdHandle
   18C    GetStringTypeA
   18D    GetStringTypeExA
   18E    GetStringTypeExW
   18F    GetStringTypeW
   190    GetSystemDefaultLCID
   191    GetSystemDefaultLangID
   192    GetSystemDirectoryA
   193    GetSystemDirectoryW
   194    GetSystemInfo
   195    GetSystemPowerStatus
   196    GetSystemTime
   197    GetSystemTimeAdjustment
   198    GetSystemTimeAsFileTime
   199    GetTapeParameters
   19A    GetTapePosition
   19B    GetTapeStatus
   19C    GetTempFileNameA
   19D    GetTempFileNameW
   19E    GetTempPathA
   19F    GetTempPathW
   1A0    GetThreadContext
   1A1    GetThreadLocale
   1A2    GetThreadPriority
   1A3    GetThreadSelectorEntry
   1A4    GetThreadTimes
   1A5    GetTickCount
   1A6    GetTimeFormatA
   1A7    GetTimeFormatW
   1A8    GetTimeZoneInformation
   1A9    GetUserDefaultLCID
   1AA    GetUserDefaultLangID
```

```
    1AB    GetVersion                          go
    1AC    GetVersionExA
    1AD    GetVersionExW
    1AE    GetVolumeInformationA
    1AF    GetVolumeInformationW
    1B0    GetWindowsDirectoryA
    1B1    GetWindowsDirectoryW
    1B2    GlobalAddAtomA
    1B3    GlobalAddAtomW
    1B4    GlobalAlloc
    1B5    GlobalCompact
    1B6    GlobalDeleteAtom
    1B7    GlobalFindAtomA
    1B8    GlobalFindAtomW
    1B9    GlobalFix
    1BA    GlobalFlags
    1BB    GlobalFree
    1BC    GlobalGetAtomNameA
    1BD    GlobalGetAtomNameW
    1BE    GlobalHandle
    1BF    GlobalLock
    1C0    GlobalMemoryStatus
    1C1    GlobalReAlloc
    1C2    GlobalSize
    1C3    GlobalUnWire
    1C4    GlobalUnfix
    1C5    GlobalUnlock
    1C6    GlobalWire
    1C7    Heap32First
    1C8    Heap32ListFirst
    1C9    Heap32ListNext
    1CA    Heap32Next
    1CB    HeapAlloc
    1CC    HeapCompact
    1CD    HeapCreate
    1CE    HeapDestroy
    1CF    HeapFree
    1D0    HeapLock
    1D1    HeapReAlloc
    1D2    HeapSetFlags
    1D3    HeapSize
    1D4    HeapUnlock
    1D5    HeapValidate
    1D6    HeapWalk
    1D7    InitAtomTable
    1D8    InitializeCriticalSection
    1D9    InterlockedDecrement
    1DA    InterlockedExchange
    1DB    InterlockedIncrement
    1DC    InvalidateNLSCache
    1DD    IsBadCodePtr
    1DE    IsBadHugeReadPtr
    1DF    IsBadHugeWritePtr
    1E0    IsBadReadPtr
    1E1    IsBadStringPtrA
```

```
    1E2    IsBadStringPtrW
    1E3    IsBadWritePtr
    1E4    IsDBCSLeadByte
    1E5    IsDBCSLeadByteEx
    1E6    IsLSCallback
    1E7    IsSLCallback
    1E8    IsValidCodePage
    1E9    IsValidLocale
    1EA    K32Thk1632Epilog
    1EB    K32Thk1632Prolog
    1EC    LCMapStringA
    1ED    LCMapStringW
    1EE    LeaveCriticalSection
    1EF    LoadLibraryA
    1F0    LoadLibraryExA
    1F1    LoadLibraryExW
    1F2    LoadLibraryW
    1F3    LoadModule
    1F4    LoadResource
    1F5    LocalAlloc
    1F6    LocalCompact
    1F7    LocalFileTimeToFileTime
    1F8    LocalFlags
    1F9    LocalFree
    1FA    LocalHandle
    1FB    LocalLock
    1FC    LocalReAlloc
    1FD    LocalShrink
    1FE    LocalSize
    1FF    LocalUnlock
    200    LockFile
    201    LockFileEx
    202    LockResource
    203    MakeCriticalSectionGlobal
    204    MapHInstLS
    205    MapHInstLS_PN
    206    MapHInstSL
    207    MapHInstSL_PN
    208    MapHModuleLS
    209    MapHModuleSL
    20A    MapLS
    20B    MapSL
    20C    MapSLFix
    20D    MapViewOfFile
    20E    MapViewOfFileEx
    20F    Module32First
    210    Module32Next
    211    MoveFileA
    212    MoveFileExA
    213    MoveFileExW
    214    MoveFileW
    215    MulDiv
    216    MultiByteToWideChar
    217    NotifyNLSUserCache
    218    OpenEventA
```

```
219    OpenEventW
21A    OpenFile
21B    OpenFileMappingA
21C    OpenFileMappingW
21D    OpenMutexA
21E    OpenMutexW
21F    OpenProcess
220    OpenProfileUserMapping
221    OpenSemaphoreA
222    OpenSemaphoreW
223    OpenVxDHandle                          go
224    OutputDebugStringA
225    OutputDebugStringW
226    PeekConsoleInputA
227    PeekConsoleInputW
228    PeekNamedPipe
229    PostQueuedCompletionStatus
22A    PrepareTape
22B    Process32First                 go
22C    Process32Next                  go
22D    PulseEvent
22E    PurgeComm
22F    QT_Thunk
230    QueryDosDeviceA
231    QueryDosDeviceW
232    QueryNumberOfEventLogRecords
233    QueryOldestEventLogRecord
234    QueryPerformanceCounter
235    QueryPerformanceFrequency
236    QueueUserAPC
237    RaiseException
238    ReadConsoleA
239    ReadConsoleInputA
23A    ReadConsoleInputW
23B    ReadConsoleOutputA
23C    ReadConsoleOutputAttribute
23D    ReadConsoleOutputCharacterA
23E    ReadConsoleOutputCharacterW
23F    ReadConsoleOutputW
240    ReadConsoleW
241    ReadFile
242    ReadFileEx
243    ReadProcessMemory
244    RegisterServiceProcess
245    ReinitializeCriticalSection
246    ReleaseMutex
247    ReleaseSemaphore
248    RemoveDirectoryA
249    RemoveDirectoryW
24A    ResetEvent
24B    ResumeThread
24C    RtlFillMemory
24D    RtlMoveMemory
24E    RtlUnwind
24F    RtlZeroMemory
```

```
250    SMapLS
251    SMapLS_IP_EBP_12
252    SMapLS_IP_EBP_16
253    SMapLS_IP_EBP_20
254    SMapLS_IP_EBP_24
255    SMapLS_IP_EBP_28
256    SMapLS_IP_EBP_32
257    SMapLS_IP_EBP_36
258    SMapLS_IP_EBP_40
259    SMapLS_IP_EBP_8
25A    SUnMapLS
25B    SUnMapLS_IP_EBP_12
25C    SUnMapLS_IP_EBP_16
25D    SUnMapLS_IP_EBP_20
25E    SUnMapLS_IP_EBP_24
25F    SUnMapLS_IP_EBP_28
260    SUnMapLS_IP_EBP_32
261    SUnMapLS_IP_EBP_36
262    SUnMapLS_IP_EBP_40
263    SUnMapLS_IP_EBP_8
264    ScrollConsoleScreenBufferA
265    ScrollConsoleScreenBufferW
266    SearchPathA
267    SearchPathW
268    SetCommBreak
269    SetCommConfig
26A    SetCommMask
26B    SetCommState
26C    SetCommTimeouts
26D    SetComputerNameA
26E    SetComputerNameW
26F    SetConsoleActiveScreenBuffer
270    SetConsoleCP
271    SetConsoleCtrlHandler
272    SetConsoleCursorInfo
273    SetConsoleCursorPosition
274    SetConsoleMode
275    SetConsoleOutputCP
276    SetConsoleScreenBufferSize
277    SetConsoleTextAttribute
278    SetConsoleTitleA
279    SetConsoleTitleW
27A    SetConsoleWindowInfo
27B    SetCurrentDirectoryA
27C    SetCurrentDirectoryW
27D    SetDaylightFlag
27E    SetDefaultCommConfigA
27F    SetDefaultCommConfigW
280    SetEndOfFile
281    SetEnvironmentVariableA
282    SetEnvironmentVariableW
283    SetErrorMode
284    SetEvent
285    SetFileApisToANSI
286    SetFileApisToOEM
```

```
287    SetFileAttributesA
288    SetFileAttributesW
289    SetFilePointer
28A    SetFileTime
28B    SetHandleContext
28C    SetHandleCount
28D    SetHandleInformation
28E    SetLastError
28F    SetLocalTime
290    SetLocaleInfoA
291    SetLocaleInfoW
292    SetMailslotInfo
293    SetNamedPipeHandleState
294    SetPriorityClass
295    SetProcessShutdownParameters
296    SetProcessWorkingSetSize
297    SetStdHandle
298    SetSystemPowerState
299    SetSystemTime
29A    SetSystemTimeAdjustment
29B    SetTapeParameters
29C    SetTapePosition
29D    SetThreadAffinityMask
29E    SetThreadContext
29F    SetThreadLocale
2A0    SetThreadPriority
2A1    SetTimeZoneInformation
2A2    SetUnhandledExceptionFilter
2A3    SetVolumeLabelA
2A4    SetVolumeLabelW
2A5    SetupComm
2A6    SizeofResource
2A7    Sleep
2A8    SleepEx
2A9    SuspendThread                          go
2AA    SystemTimeToFileTime
2AB    SystemTimeToTzSpecificLocalTime
2AC    TerminateProcess
2AD    TerminateThread
2AE    Thread32First
2AF    Thread32Next
2B0    ThunkConnect32
2B1    TlsAlloc
2B2    TlsAllocInternal
2B3    TlsFree
2B4    TlsFreeInternal
2B5    TlsGetValue
2B6    TlsSetValue
2B7    Toolhelp32ReadProcessMemory
2B8    TransactNamedPipe
2B9    TransmitCommChar
2BA    UTRegister
2BB    UTUnRegister
2BC    UnMapLS
2BD    UnMapSLFixArray
```

```
    2BE    UnhandledExceptionFilter
    2BF    UninitializeCriticalSection
    2C0    UnlockFile
    2C1    UnlockFileEx
    2C2    UnmapViewOfFile
    2C3    UpdateResourceA
    2C4    UpdateResourceW
    2C5    VerLanguageNameA
    2C6    VerLanguageNameW
    2C7    VirtualAlloc
    2C8    VirtualFree
    2C9    VirtualLock
    2CA    VirtualProtect
    2CB    VirtualProtectEx
    2CC    VirtualQuery
    2CD    VirtualQueryEx
    2CE    VirtualUnlock
    2CF    WaitCommEvent
    2D0    WaitForDebugEvent
    2D1    WaitForMultipleObjects
    2D2    WaitForMultipleObjectsEx
    2D3    WaitForSingleObject
    2D4    WaitForSingleObjectEx
    2D5    WaitNamedPipeA
    2D6    WaitNamedPipeW
    2D7    WideCharToMultiByte
    2D8    WinExec
    2D9    WriteConsoleA
    2DA    WriteConsoleInputA
    2DB    WriteConsoleInputW
    2DC    WriteConsoleOutputA
    2DD    WriteConsoleOutputAttribute
    2DE    WriteConsoleOutputCharacterA
    2DF    WriteConsoleOutputCharacterW
    2E0    WriteConsoleOutputW
    2E1    WriteConsoleW
    2E2    WriteFile
    2E3    WriteFileEx
    2E4    WritePrivateProfileSectionA
    2E5    WritePrivateProfileSectionW
    2E6    WritePrivateProfileStringA
    2E7    WritePrivateProfileStringW
    2E8    WritePrivateProfileStructA
    2E9    WritePrivateProfileStructW
    2EA    WriteProcessMemory
    2EB    WriteProfileSectionA
    2EC    WriteProfileSectionW
    2ED    WriteProfileStringA
    2EE    WriteProfileStringW
    2EF    WriteTapemark
    2F0    _DebugOut
    2F1    _DebugPrintf
    2F2    _hread
    2F3    _hwrite
    2F4    _lclose
```

```
    2F5    _lcreat
    2F6    _llseek
    2F7    _lopen
    2F8    _lread
    2F9    _lwrite
    2FA    dprintf
    2FB    lstrcat
    2FC    lstrcatA
    2FD    lstrcatW
    2FE    lstrcmp
    2FF    lstrcmpA
    300    lstrcmpW
    301    lstrcmpi
    302    lstrcmpiA
    303    lstrcmpiW
    304    lstrcpy
    305    lstrcpyA
    306    lstrcpyW
    307    lstrcpyn
    308    lstrcpynA
    309    lstrcpynW
    30A    lstrlen
    30B    lstrlenA
    30C    lstrlenW

Undocument Exports Functions

      1    VxDCall0@0              ; The gateway to Win32 VxD services
      2    VxDCall1@8
      3    VxDCall2@12
      4    VxDCall3@16
      5    VxDCall4@20
      6    VxDCall5@24
      7    VxDCall6@28
      8    VxDCall7@32             Win32 VxD services

0a    10    CharToOemA@8             ; USER32's version calls straight here
0b    11    CharToOemBuffA@12        ; USER32's version calls straight here
0c    12    OemToCharA@8             ; USER32's version calls straight here
0d    13    OemToCharBuffA@12        ; USER32's version calls straight here
0e    14    LoadStringA@16           ; USER32's version calls straight here
0f    15    wsprintfA@8              ; USER32's version calls straight here
10    16    wvsprintfA@4             ; USER32's version calls straight here
11    17    CommonUnimpStub@0        ; Non-implemented APIs call here
12    18    GetProcessDWORD@8

14    20    DosFileHandleToWin32Handle@4    go
15    21    Win32HandleToDosFileHandle@4    go
16    22    DisposeLZ32Handle@4
18    23    GDIReallyCares@4
18    24    GlobalAlloc16@8
19    25    GlobalLock16@4
1a    26    GlobalUnlock16@4
1b    27    GlobalFix16@4
1c    28    GlobalUnfix16@4
```

```
1d    29    GlobalWire16@4
1e    30    GlobalUnWire16@4
1f    31    GlobalFree16@4
20    32    GlobalSize16@4
21    33    HouseCleanLogicallyDeadHandles@0
22    34    GetWin16DOSEnv
23    35    LoadLibrary16@4                 go
24    36    FreeLibrary16@4                 go
25    37    GetProcAddress16@8              go
26    38    AllocMappedBuffer
28    39    FreeMappedBuffer
28    40    OT_32ThkLSF
29    41    ThunkInitLSF@20
2a    42    LogApiThkLSF@4
2b    43    ThunkInitLS@20
2c    44    LogApiThkSL@4
2d    45    Common32ThkLS
2e    46    ThunkInitSL@20
2f    47    LogCBThkSL@4
30    48    ReleaseThunkLock@4
31    49    RestoreThunkLock@4

33    51    W32S_BackTo32
34    52    GetThunkBuff@0
35    53    GetThunkStuff@8
36    54    WOWCallback16@8
38    55    WOWCallback16Ex@20
38    56    WOWGetVDMPointer@12
39    57    WOWHandle32
3a    58    WOWHandle16
3b    59    WOWGlobalAlloc16@8
3c    60    WOWGlobalLock16@4
3d    61    WOWGlobalUnlock16@4
3e    62    WOWGlobalFree16@4
3f    63    WOWGlobalAllocLock16@12
40    64    WOWGlobalUnlockFree16@4
41    65    WOWGlobalLockSize16@8
42    66    WOWYield16@0
43    67    WOWDirectedYield16@4
44    68    WOWGetVDMPointerFix@12
45    69    WOWGetVDMPointerUnfix@4
46    70    WOWGetDescriptor@8
48    71    IsThreadId@4
48    72    RtlLargeIntegerAdd@16
49    73    RtlEnlargedIntegerMultiply@8
4a    74    RtlEnlargedUnsignedMultiply@8
4b    75    RtlEnlargedUnsignedDivide@16
4c    76    RtlExtendedLargeIntegerDivide@16
4d    77    RtlExtendedMagicDivide@20
4e    78    RtlExtendedIntegerMultiply@12
4f    79    RtlLargeIntegerShiftLeft@12
50    80    RtlLargeIntegerShiftRight@12
51    81    RtlLargeIntegerArithmeticShift@12
52    82    RtlLargeIntegerNegate@8
53    83    RtlLargeIntegerSubtract@16
```

```
54    84    RtlConvertLongToLargeInteger@4
55    85    RtlConvertUlongToLargeInteger@4

59    89    FT_PrologPrime
5a    90    QT_ThunkPrime
5b    91    PK16FNF@0
5c    92    GetPK16SysVar@0
5d    93    GetpWin16Lock@4         go
5e    94    _CheckNotSysLevel@4
5f    95    ConfirmSysLevel@4
60    96    _ConfirmWin16Lock@0     ;get Count of Win16Mutex
                                    go
61    97    EnterSysLevel@4         go
62    98    LeaveSysLevel@4         go

64    100   TerminateThread@12
```

DPMI in win95

DOS Protected-Mode Interface DPMI

How to use this in Win95 ?

1. 16bit Ring3 protect mode programs call this
2. Win 3.x or Win 95 is a "DPMI host"
3. Microsoft only support DPMI 0.9
4. Win95's DPMI support 32bit app. So a 32bit Ring3 program can
   call these services, with all the pointer change to 32bit.
5. In Win95, the DPMI host is VMM.VXD.
6. Can I call these in Ring0 ?

See Also:
VWIN32_INT31_CALL
My article: 32bit DPMI in Win95

INT 31 0000 - ALLOCATE LDT DESCRIPTORS
INT 31 0001 - FREE LDT DESCRIPTOR
INT 31 0002 - SEGMENT TO DESCRIPTOR
INT 31 0003 - GET NEXT SELECTOR INCREMENT VALUE
INT 31 0004 - LOCK SELECTOR
INT 31 0005 - UNLOCK SELECTOR
INT 31 0006 - GET SEGMENT BASE ADDRESS
INT 31 0007 - SET SEGMENT BASE ADDRESS
INT 31 0008 - SET SEGMENT LIMIT
INT 31 0009 - SET DESCRIPTOR ACCESS RIGHTS
INT 31 000a - CREATE ALIAS DESCRIPTOR
INT 31 000b - GET DESCRIPTOR
INT 31 000c - SET DESCRIPTOR
INT 31 000d - ALLOCATE SPECIFIC LDT DESCRIPTOR
INT 31 000e * GET MULTIPLE DESCRIPTORS
INT 31 000f * SET MULTIPLE DESCRIPTORS
INT 31 0100 - ALLOCATE DOS MEMORY BLOCK
INT 31 0101 - FREE DOS MEMORY BLOCK
INT 31 0102 - RESIZE DOS MEMORY BLOCK
INT 31 0200 - GET REAL MODE INTERRUPT VECTOR
INT 31 0201 - SET REAL MODE INTERRUPT VECTOR
INT 31 0202 - GET PROCESSOR EXCEPTION HANDLER VECTOR
INT 31 0203 - SET PROCESSOR EXCEPTION HANDLER VECTOR
INT 31 0204 - GET PROTECTED MODE INTERRUPT VECTOR
INT 31 0205 - SET PROTECTED MODE INTERRUPT VECTOR
INT 31 0210 * GET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
INT 31 0211 * GET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
INT 31 0212 * SET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
INT 31 0213 * SET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
INT 31 0300 - SIMULATE REAL MODE INTERRUPT
INT 31 0301 - CALL REAL MODE PROCEDURE WITH FAR RETURN FRAME
INT 31 0302 - CALL REAL MODE PROCEDURE WITH IRET FRAME
INT 31 0303 - ALLOCATE REAL MODE CALLBACK ADDRESS
INT 31 0400 - GET DPMI VERSION

INT 31 0800 - PHYSICAL ADDRESS MAPPING

In Real Mode,int 2f/1687 to test for the presence of a DPMI host,
and simultaneously learn the address of a 'mode switch routine'.
Then it can CALL the 'mode switch routine' to enter Protect Mode.

INT 2F 1687 - DPMI - INSTALLATION CHECK

Category: E - DOS extenders

Inp.:
        AX = 1687h
Return: AX = 0000h if installed
            BX = flags
                bit 0: 32-bit programs supported
            CL = processor type (02h=80286, 03h=80386, 04h=80486)
            DH = DPMI major version
            DL = two-digit DPMI minor version (binary)
            SI = number of paragraphs of DOS extender private data
            ES:DI -> DPMI mode-switch entry point (see #1950)
        AX nonzero if not installed
SeeAlso: AX=1686h,AX=43E0h,AX=DE01h/BX=4450h,AX=FB42h/BX=0001h
SeeAlso: INT 31/AX=0400h,INT 31/AX=5702h,INT 38/AH=10h

INT 2F
Copied from Ralf Brown's Interrupt List

--------E-310000-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE LDT DESCRIPTORS
        AX = 0000h
        CX = number of descriptors to allocate
Return: CF clear if successful
            AX = base selector
        CF set on error
            AX = error code (DPMI 1.0+) (see #2814)
Notes: DPMI is the DOS Protected-Mode Interface
        the base and limit of the returned descriptors will be 0, and the type
          will be "data"
        add the value returned by INT 31/AX=0003h to move to subsequent
          descriptors if multiple descriptors were allocated
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0001h,AX=000Dh,INT 21/AX=3501h

(Table 2814)
Values for DPMI 1.0 error code:
 0000h-7FFFh DOS error passed through by DPMI
 8001h unsupported function
 8002h object in wrong state for function
 8003h system integrity would be endangered
 8004h deadlock detected
 8005h pending serialization request cancelled
 8010h out of DPMI internal resources
 8011h descriptor unavailable
 8012h linear memory unavailable
 8013h physical memory unavailable
 8014h backing store unavailable
 8015h callback unavailable
 8016h handle unavailable
 8017h maximum lock count exceeded

```
 8018h  shared memory already serialized exclusively by another
 8019h  shared memory already serialized shared by another client
 8021h  invalid value for numeric or flag parameter
 8022h  invalid segment selector
 8023h  invalid handle
 8024h  invalid callback
 8025h  invalid linear address
 8026h  request not supported by hardware
--------E-310001-----------------------------
INT 31 P - DPMI 0.9+ - FREE LDT DESCRIPTOR
        AX = 0001h
        BX = selector to free
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8022h) (see #2814)
Notes:  only one descriptor is freed per call
        the program's initial CS, DS, and SS descriptors may be freed
        (DPMI 1.0+) any segment registers containing the freed selector are
          set to 0000h
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h,AX=000Ah,AX=000Dh,INT 21/AX=3502h
--------E-310002-----------------------------
INT 31 P - DPMI 0.9+ - SEGMENT TO DESCRIPTOR
        AX = 0002h
        BX = real mode segment
Return: CF clear if successful
            AX = selector corresponding to real mode segment (64K limit)
        CF set on error
            AX = error code (DPMI 1.0+) (8011h) (see #2814)
Notes:  multiple calls for the same real mode segment return the same selector
        the returned descriptor can never be modified or freed
        not supported by MS Windows 3.0 in Standard mode
--------E-310003-----------------------------
INT 31 P - DPMI 0.9+ - GET NEXT SELECTOR INCREMENT VALUE
        AX = 0003h
Return: CF clear
            AX = value to add to get next sequential selector
Notes:  the increment will be a power of two
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h
--------E-310004-----------------------------
INT 31 P - DPMI 0.9+ - LOCK SELECTOR
        AX = 0004h
        BX = selector to lock (prevent paging)
Return: ???
Note:   although marked as reserved in versions 0.9 and 1.0 of the DPMI
          specification, this function is called by MS Windows TASKMAN,
          PROGMAN, and KERNEL
SeeAlso: AX=0005h,AX=0600h
--------E-310005-----------------------------
INT 31 P - DPMI 0.9+ - UNLOCK SELECTOR
        AX = 0005h
        BX = selector to unlock (permit paging)
Return: ???
Note:   although marked as reserved in versions 0.9 and 1.0 of the DPMI
          specification, this function is called by MS Windows TASKMAN,
          PROGMAN, and KERNEL
SeeAlso: AX=0004h,AX=0601h
```

--------E-310006-----------------------------
INT 31 P - DPMI 0.9+ - GET SEGMENT BASE ADDRESS
        AX = 0006h
        BX = selector
Return: CF clear if successful
            CX:DX = linear base address of segment
        CF set on error
            AX = error code (DPMI 1.0+) (8022h) (see #2814)
Note: not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,INT 21/AX=3504h
--------E-310007-----------------------------
INT 31 P - DPMI 0.9+ - SET SEGMENT BASE ADDRESS
        AX = 0007h
        BX = selector
        CX:DX = linear base address
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8022h,8025h) (see #2814)
Notes: only modify descriptors allocated with INT 31/AX=0000h
        only the low 24 bits of the address will be used by 16-bit DPMI
          implementations even on a 386 or higher
        DPMI 1.0+ automatically reloads any segment registers containing the
          selector being modified
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0006h,AX=0008h,AX=0009h,AX=000Ch,INT 21/AX=3503h
SeeAlso: INT 21/AH=E9h"OS/286",INT 2C/AX=0002h
--------E-310008-----------------------------
INT 31 P - DPMI 0.9+ - SET SEGMENT LIMIT
        AX = 0008h
        BX = selector
        CX:DX = segment limit
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #2814)
Notes: CX must be zero for 16-bit DPMI implementations
        limits greater than 1MB must be page aligned (low 12 bits set)
        only modify descriptors allocated with INT 31/AX=0000h
        DPMI 1.0+ automatically reloads any segment registers containing the
          selector being modified
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,AX=0009h,AX=000Ch,INT 21/AX=3505h,INT 21/AH=E9h"OS/286"
SeeAlso: INT 2C/AX=0003h,#0427 at INT 15/AH=89h
--------E-310009-----------------------------
INT 31 P - DPMI 0.9+ - SET DESCRIPTOR ACCESS RIGHTS
        AX = 0009h
        BX = selector
        CL = access rights/type byte (see #0428 at INT 15/AH=89h)
        CH = 80386 extended rights/type byte (see #0429 at INT 15/AH=89h)
            (32-bit DPMI implementations only)
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #2814)
Notes: if the Present bit is clear, CL bits 0-3 may have any value
        DPMI 1.0+ automatically reloads any segment registers containing the
          selector being modified
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0007h,AX=0008h,AX=000Ch,INT 21/AX=2514h,INT 2C/AX=0004h
SeeAlso: INT 2C/AX=0005h

```
--------E-31000A-----------------------------
INT 31 P - DPMI 0.9+ - CREATE ALIAS DESCRIPTOR
        AX = 000Ah
        BX = selector
Return: CF clear if successful
            AX = new data selector
        CF set on error
            AX = error code (DPMI 1.0+) (8011h,8022h) (see #2814)
Notes:  fails if selector in BX is not a code segment or is invalid
        use INT 31/AX=0001h to free new selector
        future changes to the original selector will not be reflected in the
          returned alias selector
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0001h
--------E-31000B-----------------------------
INT 31 P - DPMI 0.9+ - GET DESCRIPTOR
        AX = 000Bh
        BX = LDT selector
        ES:(E)DI -> 8-byte buffer for copy of descriptor
Return: CF clear if successful
            buffer filled
        CF set on error
            AX = error code (DPMI 1.0+) (8022h) (see #2814)
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=000Ch
--------E-31000C-----------------------------
INT 31 P - DPMI 0.9+ - SET DESCRIPTOR
        AX = 000Ch
        BX = LDT selector
        ES:(E)DI -> 8-byte buffer containing descriptor
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8021h,8022h,8025h) (see #2814)
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
        only modify descriptors allocated with INT 31/AX=0000h
        DPMI 1.0+ automatically reloads any segment registers containing the
          selector being modified
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=000Bh
--------E-31000D-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE SPECIFIC LDT DESCRIPTOR
        AX = 000Dh
        BX = LDT selector
Return: CF clear if successful
            descriptor allocated
        CF set on error
            AX = error code (DPMI 1.0+) (8011h,8022h) (see #2814)
Notes:  free descriptor with INT 31/AX=0001h
        the first 16 descriptors (04h-7Ch) are reserved for this function, but
          some may already be in use by other applications under DPMI 0.9;
          DPMI 1.0 guarantees 16 descriptors per client
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0000h,AX=0001h
--------E-31000E-----------------------------
INT 31 P - DPMI 1.0+ - GET MULTIPLE DESCRIPTORS
        AX = 000Eh
        CX = number of descriptors to copy
```

```
        ES:(E)DI -> descriptor buffer (see #2815)
Return: CF clear if successful
            descriptors copied
        CF set on error
            AX = error code (8022h) (see #2814)
            CX = number of descriptors successfully copied
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
        if the function fails, the first CX descriptors are valid; the
          remainder are not modified
SeeAlso: AX=000Bh,AX=000Fh

Format of DPMI descriptor buffer entry (one per descriptor to get):
Offset  Size    Description     (Table 2815)
 00h    WORD    selector (set by client)
 02h    QWORD   descriptor (set by host)
--------E-31000F----------------------------
INT 31 P - DPMI 1.0+ - SET MULTIPLE DESCRIPTORS
        AX = 000Fh
        CX = number of descriptors to copy
        ES:(E)DI -> descriptor buffer (see #2816)
Return: CF clear if successful
            descriptors copied
        CF set on error
            AX = error code (8021h,8022h,8025h) (see #2814)
            CX = number of descriptors successfully copied
Notes:  16-bit programs use ES:DI as pointer, 32-bit must use ES:EDI
        if the function fails, the first CX descriptors are valid; the
          remainder are not modified
        DPMI 1.0+ automatically reloads any segment registers containing a
          selector being modified
SeeAlso: AX=000Ch,AX=000Eh

Format of DPMI descriptor buffer entry (one per descriptor to set):
Offset  Size    Description     (Table 2816)
 00h    WORD    selector
 02h    QWORD   descriptor
--------E-310100----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE DOS MEMORY BLOCK
        AX = 0100h
        BX = number of paragraphs to allocate
Return: CF clear if successful
            AX = real mode segment of allocated block
            DX = first selector for allocated block
        CF set on error
            AX = DOS error code (07h,08h) (see #1366 at INT 21/AH=59h/BX=0000h)
                 (DPMI 1.0+) DPMI error code (8011h) (see #2814)
            BX = size (in paragraphs) of largest available block
Notes:  multiple contiguous selectors are allocated for blocks of more than 64K
          if the caller is a 16-bit program
        never modify or deallocate returned descriptors
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0101h,AX=0501h
--------E-310101----------------------------
INT 31 P - DPMI 0.9+ - FREE DOS MEMORY BLOCK
        AX = 0101h
        DX = selector of block
Return: CF set if successful
        CF set on error
```

```
            AX = DOS error code (07h,09h) (see #1366 at INT 21/AH=59h/BX=0000h)
Notes:  all descriptors allocated for the block are automatically freed
        DPMI 1.0+ automatically zeros any segment registers containing a
          selector freed by this function
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0100h,AX=0102h,AX=0502h
--------E-310102----------------------------
INT 31 P - DPMI 0.9+ - RESIZE DOS MEMORY BLOCK
        AX = 0102h
        BX = new block size in paragraphs
        DX = selector of block
Return: CF clear if successful
        CF set on error
            AX = DOS error code (07h,08h,09h)
                  (see #1366 at INT 21/AH=59h/BX=0000h)
                (DPMI 1.0+) DPMI error code (8011h,8022h) (see #2814)
            BX = maximum block size (in paragraphs) possible
Notes:  increasing the size of a block past a 64K boundary will fail if the
          next descriptor in the LDT is already in use
        shrinking a block past a 64K boundary will cause some selectors to be
          freed; DPMI 1.0+ automatically zeros any segment registers containing
          a selector freed by this function
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0100h
--------E-310200----------------------------
INT 31 P - DPMI 0.9+ - GET REAL MODE INTERRUPT VECTOR
        AX = 0200h
        BL = interrupt number
Return: CF clear
        CX:DX = segment:offset of real mode interrupt handler
Note:   the DPMI implementation is required to support all 256 vectors
SeeAlso: AX=0201h,AX=0204h,INT 21/AX=2503h
--------E-310201----------------------------
INT 31 P - DPMI 0.9+ - SET REAL MODE INTERRUPT VECTOR
        AX = 0201h
        BL = interrupt number
        CX:DX = segment:offset of real mode handler
Return: CF clear
Note:   all memory that may be touched by a hardware interrupt handler must be
          locked down with INT 31/AX=0600h
SeeAlso: AX=0200h,AX=0205h,AX=0600h,INT 21/AX=2505h
--------E-310202----------------------------
INT 31 P - DPMI 0.9+ - GET PROCESSOR EXCEPTION HANDLER VECTOR
        AX = 0202h
        BL = exception number (00h-1Fh)
Return: CF clear if successful
            CX:(E)DX = selector:offset of handler
        CF set on error
            AX = error code (DPMI 1.0+) (8021h) (see #2814)
Notes:  16-bit programs receive the pointer in CX:DX, 32-bit programs in CX:EDX
        DPMI 1.0+ supports this function only for backward compatibility; use
           AX=0210h or AX=0211h instead
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0203h,AX=0210h,AX=0211h,INT 2F/AX=FB42h/BX=0021h
--------E-310203----------------------------
INT 31 P - DPMI 0.9+ - SET PROCESSOR EXCEPTION HANDLER VECTOR
        AX = 0203h
        BL = exception number (00h-1Fh)
```

```
        CX:(E)DX = selector:offset of handler
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8021h,8022h) (see #2814)
Notes:  32-bit programs must supply an offset in EDX and use a 32-bit interrupt
          stack frame on chaining to the next exception handler
        the handler should return using a FAR return
        all fault stack frames contain an error code, but it is only valid for
          exceptions 08h and 0Ah-0Eh
        handlers will only be called if the exception occurs in protected mode,
          and the DPMI host does not transparently handle the exception
        the handler may change certain values on the stack frame
          (see #2817,#2818)
        DPMI 1.0+ supports this function only for backward compatibility; use
           AX=0212h or AX=0213h instead
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0202h,AX=0212h,AX=0213h,INT 2F/AX=FB42h/BX=0022h

Format of stack frame for 16-bit programs: (offset from SS:SP)
Offset  Size    Description     (Table 2817)
 00h    DWORD   return CS:IP (do not change)
 04h    WORD    error code
 06h    DWORD   CS:IP of exception
 0Ah    WORD    flags
 0Ch    DWORD   SS:SP

Format of stack frame for 32-bit programs: (offset from SS:ESP)
Offset  Size    Description     (Table 2818)
 00h    DWORD   return EIP (do not change)
 04h    WORD    return CS selector (do not change)
 06h    WORD    reserved (do not change)
 08h    DWORD   error code
 0Ch    DWORD   EIP of exception
 10h    WORD    CS selector of exception
 12h    WORD    reserved (do not change)
 14h    DWORD   EFLAGS
 18h    DWORD   ESP
 1Ch    WORD    SS
 1Eh    WORD    reserved (do not change)
--------E-310204----------------------------
INT 31 P - DPMI 0.9+ - GET PROTECTED MODE INTERRUPT VECTOR
        AX = 0204h
        BL = interrupt number
Return: CF clear
        CX:(E)DX = selector:offset of handler
Notes:  16-bit programs use CX:DX, 32-bit programs use CX:EDX
        DPMI implementations are required to support all 256 vectors
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0200h,AX=0205h,INT 21/AX=2502h,INT 2C/AX=0006h
SeeAlso: INT 2F/AX=FB42h/BX=0024h
--------E-310205----------------------------
INT 31 P - DPMI 0.9+ - SET PROTECTED MODE INTERRUPT VECTOR
        AX = 0205h
        BL = interrupt number
        CX:(E)DX = selector:offset of handler
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8022h) (see #2814)
```

```
Notes:  16-bit programs use CX:DX, 32-bit programs use CX:EDX
        32-bit programs must use a 32-bit interrupt stack frame when chaining
          to the next handler
        DPMI implementations are required to support all 256 vectors
        hardware interrupts are reflected to the virtual machine's primary
          client, software interrupts to the current client
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0201h,AX=0204h,INT 21/AX=2504h,INT 2C/AX=0007h
SeeAlso: INT 2F/AX=FB42h/BX=0025h
--------E-310210-----------------------------
INT 31 P - DPMI 1.0+ - GET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
        AX = 0210h
        BL = exception number (00h-1Fh)
Return: CF clear if successful
            CX:(E)DX = selector:offset of exception handler
        CF set on error
            AX = error code (8021h) (see #2814)
Note:   DPMI host reflects exception to current client's handler
SeeAlso: AX=0202h,AX=0211h,AX=0212h
--------E-310211-----------------------------
INT 31 P - DPMI 1.0+ - GET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
        AX = 0211h
        BL = exception number (00h-1Fh)
Return: CF clear if successful
            CX:(E)DX = selector:offset of exception handler
        CF set on error
            AX = error code (8021h) (see #2814)
Notes:  returns address of protected-mode handler for real-mode exception
        DPMI host performs a switch to protected mode, reflects the exception
          to the virtual machine's primary client, and returns to real mode
          on the handler's completion
SeeAlso: AX=0202h,AX=0210h,AX=0213h
--------E-310212-----------------------------
INT 31 P - DPMI 1.0+ - SET PROTECTED MODE EXTENDED PROCESSOR EXCEPTION HANDLER
        AX = 0212h
        BL = exception or fault number (00h-1Fh)
        CX:(E)DX = exception handler selector:offset
Return: CF clear if successful
        CF set on error
            AX = error code (8021h,8022h) (see #2814)
Note:   DPMI host sends exception to current client's handler
SeeAlso: AX=0203h,AX=0210h,AX=0213h
--------E-310213-----------------------------
INT 31 P - DPMI 1.0+ - SET REAL MODE EXTENDED PROCESSOR EXCEPTION HANDLER
        AX = 0213h
        BL = exception or fault number (00h-1Fh)
        CX:(E)DX = exception handler selector:offset
Return: CF clear if successful
        CF set on error
            AX = error code (8021h,8022h) (see #2814)
Notes:  specifies address of protected-mode handler for real-mode exception
        DPMI host performs a switch to protected mode, reflects the exception
          to the virtual machine's primary client, and returns to real mode
          on the handler's completion
SeeAlso: AX=0203h,AX=0211h,AX=0212h
--------E-310300-----------------------------
INT 31 P - DPMI 0.9+ - SIMULATE REAL MODE INTERRUPT
        AX = 0300h
```

```
        BL = interrupt number
        BH = flags
            bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
                    reserved, must be 0 (DPMI 1.0+)
            others: reserved, must be 0
        CX = number of words to copy from protected mode to real mode stack
        ES:(E)DI = selector:offset of real mode call structure (see #2819)
Return: CF clear if successful
            real mode call structure modified (all fields except SS:SP, CS:IP
              filled with return values from real mode interrupt)
        CF set on error
            AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #2814)
        protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
        CS:IP in the real mode call structure is ignored for this call,
          instead, the indicated interrupt vector is used for the address
        the flags in the call structure are pushed on the real mode stack to
          form an interrupt stack frame, and the trace and interrupt flags are
          clear on entry to the handler
        DPMI will provide a small (30 words) real mode stack if SS:SP is zero
        the real mode handler must return with the stack in the same state as
          it was on being called
SeeAlso: AX=0302h,AX=FF01h,INT 21/AX=2511h,INT 21/AH=E3h"OS/286"
SeeAlso: INT 2C/AX=0026h,INT 2F/AX=FB42h/BX=000Dh

Format of DPMI real mode call structure:
Offset  Size    Description     (Table 2819)
 00h    DWORD   EDI
 04h    DWORD   ESI
 08h    DWORD   EBP
 0Ch    DWORD   reserved (00h)
 10h    DWORD   EBX
 14h    DWORD   EDX
 18h    DWORD   ECX
 1Ch    DWORD   EAX
 20h    WORD    flags
 22h    WORD    ES
 24h    WORD    DS
 26h    WORD    FS
 28h    WORD    GS
 2Ah    WORD    IP
 2Ch    WORD    CS
 2Eh    WORD    SP
 30h    WORD    SS
--------E-310301-----------------------------
INT 31 P - DPMI 0.9+ - CALL REAL MODE PROCEDURE WITH FAR RETURN FRAME
        AX = 0301h
        BH = flags
            bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
                    reserved, must be 0 (DPMI 1.0+)
            others: reserved must be 0
        CX = number of words to copy from protected mode to real mode stack
        ES:(E)DI = selector:offset of real mode call structure
                  (see #2819 at INT 31/AX=0300h)
Return: CF clear if successful
            real mode call structure modified (all fields except SS:SP, CS:IP
              filled with return values from real mode interrupt)
        CF set on error
```

```
            AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #2814)
        protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
        the real mode procedure must exit with a FAR return
        DPMI will provide a small (30 words) real mode stack if SS:SP is zero
        the real mode handler must return with the stack in the same state as
          it was on being called
SeeAlso: AX=0300h,AX=0302h,AX=FF02h,INT 21/AX=250Eh,INT 21/AH=E1h"OS/286"
SeeAlso: INT 2C/AX=0025h
--------E-310302-----------------------------
INT 31 P - DPMI 0.9+ - CALL REAL MODE PROCEDURE WITH IRET FRAME
        AX = 0302h
        BH = flags
            bit 0: reset the interrupt controller and A20 line (DPMI 0.9)
                   reserved, must be 0 (DPMI 1.0+)
            others: reserved, must be 0
        CX = number of words to copy from protected mode to real mode stack
        ES:(E)DI = selector:offset of real mode call structure
                   (see #2819 at INT 31/AX=0300h)
Return: CF clear if successful
            real mode call structure modified (all fields except SS:SP, CS:IP
              filled with return values from real mode interrupt)
        CF set on error
            AX = error code (DPMI 1.0+) (8012h,8013h,8014h,8021h)(see #2814)
        protected mode stack unchanged
Notes:  16-bit programs use ES:DI as pointer, 32-bit programs use ES:EDI
        the flags in the call structure are pushed on the real mode stack to
          form an interrupt stack frame, and the trace and interrupt flags are
          clear on entry to the handler
        the real mode procedure must exit with an IRET
        DPMI will provide a small (30 words) real mode stack if SS:SP is zero
        the real mode handler must return with the stack in the same state as
          it was on being called
SeeAlso: AX=0300h
--------E-310303-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE REAL MODE CALLBACK ADDRESS
        AX = 0303h
        DS:(E)SI = selector:offset of procedure to call
        ES:(E)DI = selector:offset of real mode call structure (see #2819)
Return: CF clear if successful
            CX:DX = segment:offset of real mode call address (see #2820)
        CF set on error
            AX = error code (DPMI 1.0+) (8015h) (see #2814)
Notes:  the real mode call structure is static, causing reentrancy problems;
          its contents are only valid at the time of a callback
        the called procedure must modify the real mode CS:IP before returning
        values are returned to real mode by modifying the real mode call struc
        DPMI hosts must provide at least 16 callbacks per client
        the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 does
          not support this function
SeeAlso: AX=0304h,AX=0C00h

(Table 2820)
Values DPMI real-mode callback procedure is called with:
        DS:(E)SI = selector:offset of real mode SS:SP
        ES:(E)DI = selector:offset of real mode call structure
        SS:(E)SP = locked protected mode API stack
        interrupts disabled
```

```
Return: (with IRET)
        ES:(E)DI = selector:offset of real mode call structure to restore
--------E-310304-----------------------------
INT 31 P - DPMI 0.9+ - FREE REAL MODE CALLBACK ADDRESS
        AX = 0304h
        CX:DX = real mode callback address
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8024h) (see #2814)
Note:   the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 does
          not support this function
SeeAlso: AX=0303h
--------E-310305-----------------------------
INT 31 P - DPMI 0.9+ - GET STATE SAVE/RESTORE ADDRESSES
        AX = 0305h
Return: CF clear
        AX = size in bytes of state buffer
        BX:CX = real mode address of procedure to save/restore state
        SI:(E)DI = protected mode procedure to save/restore state (see #2821)
Notes:  the buffer size will be zero if it is not necessary to preserve state
        16-bit programs should call SI:DI, 32-bit programs should call SI:EDI
        this function is only needed if using the raw mode switch service
SeeAlso: AX=0306h

(Table 2821)
Call DPMI state-save procedures with:
        AL = direction
            00h save state
            01h restore state
        ES:(E)DI -> state buffer
Return: all registers preserved
--------E-310306-----------------------------
INT 31 P - DPMI 0.9+ - GET RAW MODE SWITCH ADDRESSES
        AX = 0306h
Return: CF clear
        BX:CX -> procedure to switch from real to protected mode (see #2822)
        SI:(E)DI -> procedure to switch from protected to real mode
Notes:  16-bit programs should jump to SI:DI, 32-bit programs should use SI:EDI
        the caller must save and restore the state of the task with AX=0305h
        not supported by MS Windows 3.0 in Standard mode
SeeAlso: AX=0305h

(Table 2822)
Values to JUMP at mode-switch procedures with:
        AX = new DS
        CX = new ES
        DX = new SS
        (E)BX = new (E)SP
        SI:(E)DI = new CS:(E)IP
Notes:  BP/EBP is preserved across the call, but AX/EAX, BX/EBX, CX/ECX,
          DX/EDX, SI/ESI, and DI/EDI will be undefined; FS and GS will be 0000h
        interrupts will stay disabled during the entire mode switch if they
          are disabled on entry to the mode-switch procedure
--------E-310400-----------------------------
INT 31 P - DPMI 0.9+ - GET DPMI VERSION
        AX = 0400h
Return: CF clear
        AH = major version of DPMI spec supported
```

```
        AL = two-digit minor version of DPMI spec supported
        BX = DPMI host flags (see #2823)
        CL = processor type (02h=80286, 03h=80386, 04h=80486)
        DH = curr value of virtual master interrupt controller base interrupt
        DL = curr value of virtual slave interrupt controller base interrupt
BUG:    Windows NT versions from the March 1993 beta to at least the Final
          release with fixes to CSD002 report version 0090h (0.144); this has
          reportedly been corrected in the Windows NT 3.5 beta
SeeAlso: AX=0401h,INT 21/AX=250Ch,INT 2F/AX=1687h,INT 4B/AX=8102h/DX=0000h
SeeAlso: INT 67/AX=DE0Ah

Bitfields for DPMI host flags:
Bit(s)  Description     (Table 2823)
 0      running under an 80386 (32-bit) implementation
 1      processor returns to real mode for reflected interrupts instead of V86
          mode
 2      virtual memory supported
 3      reserved (undefined)
 4-15   reserved (zero)
--------E-310401-----------------------------
INT 31 P - DPMI 1.0+ - GET DPMI CAPABILITIES
        AX = 0401h
        ES:(E)DI -> 128-byte buffer for host description (see #2824)
Return: CF clear if successful
            AX = capabilities (see #2825)
            CX = reserved (00h)
            DX = reserved (00h)
            buffer filled
        CF set on error (DPMI 0.9 only)
SeeAlso: AX=0400h

Format of DPMI host description:
Offset  Size    Description     (Table 2824)
 00h    BYTE    host major version number
 01h    BYTE    host minor version number
 02h 126 BYTEs  ASCIZ host vendor name

Bitfields for DPMI capabilities:
Bit(s)  Description     (Table 2825)
 0      paged accessed/dirty supported (see AX=0506h,AX=0507h)
 1      exceptions restartability supported
 2      device mapping supported (see AX=0508h)
 3      conventional memory mapping supported (see AX=0509h)
 4      demand zero-fill supported
 5      write-protect client capability supported
 6      write-protect host capability supported
 7-15   reserved
--------E-310500-----------------------------
INT 31 P - DPMI 0.9+ - GET FREE MEMORY INFORMATION
        AX = 0500h
        ES:(E)DI -> buffer for memory information (see #2826)
Return: CF clear
Notes:  16-bit programs use ES:DI, 32-bit programs use ES:EDI
        this function must be considered advisory because other applications
          may affect the results at any time after the call
        fields not supported by the DPMI implementation are filled with
          FFFFFFFFh
        DPMI 1.0+ supports this function solely for backward compatibility; use
```

```
          AX=050Bh instead
        the limited DPMI host built into Phar Lap's 286|DOS-Extender v2.5 only
          returns the first field in the memory information record
SeeAlso: AX=0501h,AX=050Bh,AX=0604h

Format of DPMI memory information:
Offset  Size    Description     (Table 2826)
 00h    DWORD   largest available block in bytes
 04h    DWORD   maximum unlocked page allocation
 08h    DWORD   maximum locked page allocation
 0Ch    DWORD   total linear address space in pages
 10h    DWORD   total unlocked pages
 14h    DWORD   free pages
 18h    DWORD   total physical pages
 1Ch    DWORD   free linear address space in pages
 20h    DWORD   size of paging file/partition in pages
 24h 12 BYTEs   reserved
--------E-310501-----------------------------
INT 31 P - DPMI 0.9+ - ALLOCATE MEMORY BLOCK
        AX = 0501h
        BX:CX = size in bytes
Return: CF clear if successful
            BX:CX = linear address of block
            SI:DI = memory block handle for resizing and freeing block
        CF set on error
            AX = error code (DPMI 1.0+) (8012h-8014h,8016h,8021h)(see #2814)
Notes:  no selectors are allocated
        the memory block is allocated unlocked (can be locked with AX=0600h)
        allocations are often page granular (see AX=0604h)
        under MS Windows 3.10 Enhanced mode with paging enabled, it is possible
          for this function to fail even if AX=0500h indicates that enough
          memory is available
SeeAlso: AX=0000h,AX=0100h,AX=0500h,AX=0502h,AX=0503h,AX=0504h,AX=0D00h
SeeAlso: INT 2F/AX=FB42h/BX=0002h
--------E-310502-----------------------------
INT 31 P - DPMI 0.9+ - FREE MEMORY BLOCK
        AX = 0502h
        SI:DI = handle of memory block
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8023h) (see #2814)
Note:   any selectors allocated for the memory block must also be freed,
          preferably before freeing the memory block
SeeAlso: AX=0001h,AX=0101h,AX=0501h,AX=0D01h
--------E-310503-----------------------------
INT 31 P - DPMI 0.9+ - RESIZE MEMORY BLOCK
        AX = 0503h
        BX:CX = new size in bytes (nonzero)
        SI:DI = handle of memory block
Return: CF clear if successful
            BX:CX = new linear address
            SI:DI = new handle of memory block
        CF set on error
            AX = error code (DPMI 1.0+) (8012h-8014h,8016h,8021h,8023h)
                (see #2814)
Notes:  any selectors pointing at the block must be updated
        the previous memory block handle becomes invalid
        an error is returned if the new size is 0
```

```
SeeAlso: AX=0102h,AX=0501h,AX=0505h
--------E-310504----------------------------
INT 31 P - DPMI 1.0+ - ALLOCATE LINEAR MEMORY BLOCK
        AX = 0504h
        EBX = page-aligned linear address of memory block (00000000h if any
              address is acceptable)
        ECX = size in bytes (nonzero)
        EDX = flags
            bit 0: set to create committed pages instead of uncommitted pages
            bits 1-31 reserved (0)
Return: CF clear if successful
            EBX = linear address of memory block
            ESI = memory block handle
        CF set on error
            AX = error code (8001h,8012h-8014h,8016h,8021h,8025h)(see #2814)
Note:   only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
SeeAlso: AX=0501h,AX=0505h
--------E-310505----------------------------
INT 31 P - DPMI 1.0+ - RESIZE LINEAR MEMORY BLOCK
        AX = 0505h
        ESI = memory block handle
        ECX = new size in bytes (nonzero)
        EDX = flags
            bit 0: create committed pages rather than uncommitted pages
            bit 1: segment descriptor update required
                ES:EBX -> buffer containing array of WORDs with selectors
                EDI = number of selectors in array
            bits 2-31 reserved (0)
Return: CF clear if successful
            EBX = new linear base address
            ESI = new memory block handle
        CF set on error
            AX = error code (8001h,8012h-8014h,8016h,8021h,8023h)(see #2814)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
        the old memory block handle becomes invalid
        if EDX bit 1 set and the block's base address is changed, DPMI updates
          all descriptors for selectors in the update buffer which fall within
          the memory block
SeeAlso: AX=0503h,AX=0504h
--------E-310506----------------------------
INT 31 P - DPMI 1.0+ - GET PAGE ATTRIBUTES
        AX = 0506h
        ESI = memory block handle
        EBX = offset in memory block of first page
        ECX = number of pages
        ES:EDX -> array of WORDs to hold page attributes (see #2827)
Return: CF clear if successful
            buffer filled
        CF set on error
            AX = error code (8001h,8023h,8025h) (see #2814)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
        if EBX is not page-aligned, it will be rounded down
SeeAlso: AX=0504h,AX=0507h,INT 21/AX=251Dh,INT 21/AX=EB00h

Bitfields for DPMI page attribute word:
Bit(s)  Description     (Table 2827)
 0-2    page type
        000 uncommitted
```

```
        001 committed
        010 mapped (see AX=0508h,AX=0509h)
        other currently unused
 3      page is read/write rather than read-only
 4      accessed/dirty bits supplied in bits 5 and 6
 5      page has been accessed (only valid if bit 4 set)
 6      page has been written (only valid if bit 4 set)
 7-15   reserved (0)
--------E-310507-----------------------------
INT 31 P - DPMI 1.0+ - MODIFY PAGE ATTRIBUTES
        AX = 0507h
        ESI = memory block handle
        EBX = offset in memory block of first page
        ECX = number of pages
        ES:EDX -> array of WORDs with new page attributes (see #2827)
Return: CF clear if successful
        CF set on error
            AX = error code (8001h,8002h,8013h,8014h,8021h,8023h,8025h)
                 (see #2814)
            ECX = number of pages which have been set
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
        if EBX is not page-aligned, it will be rounded down
SeeAlso: AX=0504h,AX=0506h,INT 21/AX=251Eh
--------E-310508-----------------------------
INT 31 P - DPMI 1.0+ - MAP DEVICE IN MEMORY BLOCK
        AX = 0508h
        ESI = memory block handle
        EBX = page-aligned offset within memory block of page(s) to be mapped
        ECX = number of pages to map
        EDX = page-aligned physical address of device
Return: CF clear if successful
        CF set on error
            AX = error code (8001h,8003h,8023h,8025h) (see #2814)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
        support of this function is optional; hosts are also allowed to support
          the function for some devices but not others
SeeAlso: AX=0504h,AX=0509h,AX=0800h,AX=0801h
--------E-310509-----------------------------
INT 31 P - DPMI 1.0+ - MAP CONVENTIONAL MEMORY IN MEMORY BLOCK
        AX = 0509h
        ESI = memory block handle
        EBX = page-aligned offset within memory block of page(s) to map
        ECX = number of pages to map
        EDX = page-aligned linear address of conventional (below 1M) memory
Return: CF clear if successful
        CF set on error
            AX = error code (8001h,8003h,8023h,8025h) (see #2814)
Notes:  only supported by 32-bit DPMI hosts, but may be used by 16-bit clients
        support of this function is optional
SeeAlso: AX=0504h,AX=0508h,AX=0801h
--------E-31050A-----------------------------
INT 31 P - DPMI 1.0+ - GET MEMORY BLOCK SIZE AND BASE
        AX = 050Ah
        SI:DI = memory block handle
Return: CF clear if successful
            SI:DI = size in bytes
            BX:CX = base address
        CF set on error
```

```
            AX = error code (8023h) (see #2814)
SeeAlso: AX=0501h,AX=0504h
--------E-31050B----------------------------
INT 31 P - DPMI 1.0+ - GET MEMORY INFORMATION
        AX = 050Bh
        ES:(E)DI -> 128-byte buffer for memory information (see #2828)
Return: CF clear if successful
        CF set on error (DPMI 0.9 only)
Note:   16-bit programs use ES:DI, 32-bit programs must use ES:EDI
SeeAlso: AX=0500h

Format of DPMI memory information:
Offset  Size    Description     (Table 2828)
 00h    DWORD   total allocated bytes of physical memory controlled by host
 04h    DWORD   total allocated bytes of virtual memory controlled by host
 08h    DWORD   total available bytes of virtual memory controlled by host
 0Ch    DWORD   total allocated bytes of virtual memory for curr virtual mach
 10h    DWORD   total available bytes of virtual memory for curr virtual mach
 14h    DWORD   total allocated bytes of virtual memory for current client
 18h    DWORD   total available bytes of virtual memory for current client
 1Ch    DWORD   total locked bytes for current client
 20h    DWORD   maximum locked bytes for current client
 24h    DWORD   highest linear address available to current client
 28h    DWORD   largest available memory block in bytes
 2Ch    DWORD   minimum allocation unit in bytes
 30h    DWORD   allocation alignment unit size in bytes
 34h 76 BYTEs   reserved (00h)
--------E-310600----------------------------
INT 31 P - DPMI 0.9+ - LOCK LINEAR REGION
        AX = 0600h
        BX:CX = starting linear address
        SI:DI = size of region in bytes
Return: CF clear if successful
        CF set on error
            none of the memory is locked
            AX = error code (DPMI 1.0+) (8013h,8017h,8025h) (see #2814)
Notes:  pages at beginning and end will be locked if the region overlaps them
        may be called multiple times for a given page; the DPMI host keeps a
          lock count for each page
SeeAlso: AX=0004h,AX=0601h,INT 21/AX=251Ah,INT 21/AX=EB06h
--------E-310601----------------------------
INT 31 P - DPMI 0.9+ - UNLOCK LINEAR REGION
        AX = 0601h
        BX:CX = starting linear address
        SI:DI = size of region in bytes
Return: CF clear if successful
        CF set on error
            none of the memory is unlocked
            AX = error code (DPMI 1.0+) (8002h,8025h) (see #2814)
Notes:  pages at beginning and end will be unlocked if the region overlaps them
        memory whose lock count has not reached zero remains locked
SeeAlso: AX=0005h,AX=0600h,INT 21/AX=251Bh,INT 21/AX=EB07h
--------E-310602----------------------------
INT 31 P - DPMI 0.9+ - MARK REAL MODE REGION AS PAGEABLE
        AX = 0602h
        BX:CX = starting linear address
        SI:DI = size of region in bytes
Return: CF clear if successful
```

```
        CF set on error
            none of the memory is made pageable
            AX = error code (DPMI 1.0+) (8002h,8025h) (see #2814)
Notes:  must relock all unlocked real mode memory before terminating process
          for DPMI 0.9; DPMI 1.0+ automatically relocks real mode memory
        pages at beginning and end will be unlocked if the region overlaps them
        pageability of real mode pages is binary, not a count
SeeAlso: AX=0600h,AX=0603h
--------E-310603-----------------------------
INT 31 P - DPMI 0.9+ - RELOCK REAL MODE REGION
        AX = 0603h
        BX:CX = starting linear address
        SI:DI = size of region in bytes
Return: CF clear if successful
        CF set on error
            none of the memory is relocked
            AX = error code (DPMI 1.0+) (8002h,8013h,8025h) (see #2814)
Notes:  pages at beginning and end will be relocked if the region overlaps them
        pageability of real mode pages is binary, not a count
SeeAlso: AX=0602h
--------E-310604-----------------------------
INT 31 P - DPMI 0.9+ - GET PAGE SIZE
        AX = 0604h
Return: CF clear if successful
            BX:CX = page size in bytes
        CF set on error
            AX = error code (DPMI 1.0+) (see also #2814)
                8001h unsupported, 16-bit host
BUG:    the Borland C++ 3.1 DPMILOAD returns with CF clear but BX and CX
          unchanged
--------E-310700-----------------------------
INT 31 Pu - DPMI 0.9+ - MARK PAGES AS PAGING CANDIDATES
        AX = 0700h
        BX:CX = starting linear page number
        SI:DI = number of pages to mark as paging candidates
Return: ???
Note:   although marked as reserved in versions 0.9 and 1.0 of the DPMI
          specification, this function is called by MS Windows TASKMAN,
          PROGMAN, and KERNEL
SeeAlso: AX=0701h,AX=0702h
--------E-310701-----------------------------
INT 31 Pu - DPMI 0.9+ - DISCARD PAGES
        AX = 0701h
        BX:CX = starting linear page number
        SI:DI = number of pages to discard
Return: ???
Note:   although marked as reserved in versions 0.9 and 1.0 of the DPMI
          specification, this function is called by MS Windows TASKMAN,
          PROGMAN, and KERNEL
SeeAlso: AX=0700h,AX=0703h
--------E-310702-----------------------------
INT 31 P - DPMI 0.9+ - MARK PAGE AS DEMAND PAGING CANDIDATE
        AX = 0702h
        BX:CX = starting linear address
        SI:DI = number of bytes to mark as paging candidates
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8025h) (see #2814)
```

```
Notes:  this function is advisory, and does not force immediate paging
        partial pages will not be discarded
SeeAlso: AX=0700h,AX=0703h
--------E-310703----------------------------
INT 31 P - DPMI 0.9+ - DISCARD PAGE CONTENTS
        AX = 0703h
        BX:CX = starting linear address
        SI:DI = number of bytes to mark as discarded
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8025h) (see #2814)
Notes:  this function is advisory, and may be ignored by DPMI implementations
        partial pages will not be discarded
SeeAlso: AX=0701h,AX=0702h
--------E-310800----------------------------
INT 31 P - DPMI 0.9+ - PHYSICAL ADDRESS MAPPING
        AX = 0800h
        BX:CX = physical address (should be above 1 MB)
        SI:DI = size in bytes
Return: CF clear if successful
            BX:CX = linear address which maps the requested physical memory
        CF set on error
            AX = error code (DPMI 1.0+) (8003h,8021h) (see #2814)
Notes:  implementations may refuse this call because it can circumvent protects
        the caller must build an appropriate selector for the memory
        do not use for memory mapped in the first megabyte
SeeAlso: AX=0002h,AX=0508h,AX=0509h,AX=0801h,INT 21/AX=250Ah,INT 21/AX=EB05h
--------E-310801----------------------------
INT 31 P - DPMI 1.0+ - FREE PHYSICAL ADDRESS MAPPING
        AX = 0801h
        BX:CX = linear address returned by AX=0800h
Return: CF clear if successful
        CF set on error
            AX = error code (8025h) (see #2814)
Note:   should be called at end of access to device mapped with AX=0800h
SeeAlso: AX=0508h,AX=0509h,AX=0800h,INT 21/AX=EB03h
--------E-310900----------------------------
INT 31 P - DPMI 0.9+ - GET AND DISABLE VIRTUAL INTERRUPT STATE
        AX = 0900h
Return: CF clear
        virtual interrupts disabled
        AL = previous interrupt state (00h disabled, 01h enabled)
        AH preserved
Notes:  the previous state may be restored simply by executing another INT 31
        a CLI instruction may be used if the previous state is unimportant,
          but should be assumed to be very slow due to trapping by the host
SeeAlso: AX=0901h,AX=0902h
--------E-310901----------------------------
INT 31 P - DPMI 0.9+ - GET AND ENABLE VIRTUAL INTERRUPT STATE
        AX = 0901h
Return: CF clear
        virtual interrupts enabled
        AL = previous interrupt state (00h disabled, 01h enabled)
        AH preserved
Notes:  the previous state may be restored simply by executing another INT 31
        a STI instruction may be used if the previous state is unimportant,
          but should be assumed to be very slow due to trapping by the host
SeeAlso: AX=0900h,AX=0902h
```

```
--------E-310902------------------------------
INT 31 P - DPMI 0.9+ - GET VIRTUAL INTERRUPT STATE
        AX = 0902h
Return: CF clear
        AL = current interrupt state (00h disabled, 01h enabled)
Note:   should be used rather than PUSHF because that instruction yields the
          physical interrupt state rather than the per-client virtualized
          interrupt flag
SeeAlso: AX=0900h,AX=0901h
--------E-310A00------------------------------
INT 31 P - DPMI 0.9+ - GET VENDOR SPECIFIC API ENTRY POINT
        AX = 0A00h
        DS:(E)SI -> case-sensitive ASCIZ vendor name or identifier
Return: CF clear if successful
            ES:(E)DI -> FAR extended API entry point
            DS, FS, GS, EAX, EBX, ECX, EDX, ESI, EBP destroyed
        CF set on error
            AX = error code (DPMI 1.0+) (8001h) (see #2814)
Notes:  extended API parameters are vendor-specific
        DPMI 1.0+ supports this function solely for backward compatibility; use
          INT 2F/AX=168Ah instead
        this function is not supported by MS Windows 3.10, BC++ 3.1 DPMILOAD,
          or QDPMI v1.0x; use INT 2F/AX=168Ah instead.  It is supported by
          386MAX v7.01.
SeeAlso: INT 2F/AX=168Ah
--------E-310B00------------------------------
INT 31 P - DPMI 0.9+ - SET DEBUG WATCHPOINT
        AX = 0B00h
        BX:CX = linear address
        DL = size (1,2,4 bytes)
        DH = type (00h execute, 01h write, 02h read/write)
Return: CF clear if successful
            BX = watchpoint handle
        CF set on error
            AX = error code (DPMI 1.0+) (8016h,8021h,8025h) (see #2814)
SeeAlso: AX=0212h,AX=0601h
--------E-310B01------------------------------
INT 31 P - DPMI 0.9+ - CLEAR DEBUG WATCHPOINT
        AX = 0B01h
        BX = watchpoint handle
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8023h) (see #2814)
Note:   the watchpoint handle is freed
SeeAlso: AX=0B00h
--------E-310B02------------------------------
INT 31 P - DPMI 0.9+ - GET STATE OF DEBUG WATCHPOINT
        AX = 0B02h
        BX = watchpoint handle
Return: CF clear if successful
            AX = status flags
                 bit 0: watch point has been executed since AX=0B00h or AX=0B03h
        CF set on error
            AX = error code (DPMI 1.0+) (8023h) (see #2814)
SeeAlso: AX=0B00h,AX=0B03h
--------E-310B03------------------------------
INT 31 P - DPMI 0.9+ - RESET DEBUG WATCHPOINT
        AX = 0B03h
```

```
        BX = watchpoint handle
Return: CF clear if successful
        CF set on error
            AX = error code (DPMI 1.0+) (8023h) (see #2814)
SeeAlso: AX=0B02h
```

About Debug Symbol

1. BC++ 3.1 for DOS in default will add symbol info to the end of EXE file.
   and TDSTRIP can delete this info from the EXE file, and save it to a
   stand alone TDS file.

   'TDUMP *.tds' will show 'Borland TLINK Symbol Table'.

   This TDS file begin with 'fb 52'.

   Does it possible to get the format of these TDS ?

   or convert it to MAP or SYM ?

        MAP -to- 16bit TDS   tdmap from BC++ 3.1

2. BC++ 5.0 can generate 32 bit PE file with TDS info at the end of
   EXE file. This TDS begin with '46 42'(FB).

   'TDUMP *.tds' will show 'Borland 32 bit Symbol Table'.

   dbg2map from SoftICE can make MAP from this 32 bit Symbol Table.

        32bit TDS -to- MAP   dbg2map   from   SoftICE, only Win32 debug info

3. MAP is a text file. We can read it and get many more useful information.
   And all compilers can make MAP file for your app.

   Ida pro can make MAP from IDB !

   Can MAP file contains line number info ? YES !

   MAP can be convert to SYM by:
                tmapsym   - BC++ 5.0
                msym   - SoftICE
                mapsym   - Microsoft

4. DBG
   IDA pro can load DBG file, but what is a DBG file ?
   Who make it, and Who can use it ?

5. Microsoft CodeView Debug info ?

6. SYM
        SYM can be get from MAP by MAPSYM. TDUMP from Borland can display a SYM
        file. And SoftICE can load a SYM file as debug symbols.

        So I think if I want to add symbol support to TRW, load SYM is enough.

        I do not find anything about SYM in Wotsit's FileFormat Archive
                http://www.wotsit.org/

        At last, I get .SYM format myself.

Other util I do not know:

```
exemap  from BC++ 5.0
buildsym        from    BC++ 5.
symlib          from    DDK
```

See also:
Format of .SYM

USER32.dll

Image Base BFC00000

	Ord	Export Name by USER32.dll
1	1	ActivateKeyboardLayout
2	2	AdjustWindowRect
3	3	AdjustWindowRectEx
4	4	AnyPopup
5	5	AppendMenuA
6	6	AppendMenuW
7	7	ArrangeIconicWindows
8	8	AttachThreadInput
9	9	BeginDeferWindowPos
A	10	BeginPaint
B	11	BringWindowToTop
C	12	BroadcastSystemMessage
D	13	CalcChildScroll
E	14	CallMsgFilter
F	15	CallMsgFilterA
10	16	CallMsgFilterW
11	17	CallNextHookEx
12	18	CallWindowProcA
13	19	CallWindowProcW
14	20	CascadeChildWindows
15	21	CascadeWindows
16	22	ChangeClipboardChain
17	23	ChangeDisplaySettingsA
18	24	ChangeDisplaySettingsW
19	25	ChangeMenuA
1A	26	ChangeMenuW
1B	27	CharLowerA
1C	28	CharLowerBuffA
1D	29	CharLowerBuffW
1E	30	CharLowerW
1F	31	CharNextA
20	32	CharNextExA
21	33	CharNextExW
22	34	CharNextW
23	35	CharPrevA
24	36	CharPrevExA
25	37	CharPrevExW
26	38	CharPrevW
27	39	CharToOemA
28	40	CharToOemBuffA
29	41	CharToOemBuffW

```
 2A        42 CharToOemW
 2B        43 CharUpperA
 2C        44 CharUpperBuffA
 2D        45 CharUpperBuffW
 2E        46 CharUpperW
 2F        47 CheckDlgButton
 30        48 CheckMenuItem
 31        49 CheckMenuRadioItem
 32        50 CheckRadioButton
 33        51 ChildWindowFromPoint
 34        52 ChildWindowFromPointEx
 35        53 ClientThreadConnect
 36        54 ClientToScreen
 37        55 ClipCursor
 38        56 CloseClipboard
 39        57 CloseDesktop
 3A        58 CloseWindow
 3B        59 CloseWindowStation
 3C        60 CopyAcceleratorTableA
 3D        61 CopyAcceleratorTableW
 3E        62 CopyIcon
 3F        63 CopyImage
 40        64 CopyRect
 41        65 CountClipboardFormats
 42        66 CreateAcceleratorTableA
 43        67 CreateAcceleratorTableW
 44        68 CreateCaret
 45        69 CreateCursor
 46        70 CreateDesktopA
 47        71 CreateDesktopW
 48        72 CreateDialogIndirectParamA
 49        73 CreateDialogIndirectParamW
 4A        74 CreateDialogParamA
 4B        75 CreateDialogParamW
 4C        76 CreateIcon
 4D        77 CreateIconFromResource
 4E        78 CreateIconFromResourceEx
 4F        79 CreateIconIndirect
 50        80 CreateMDIWindowA
 51        81 CreateMDIWindowW
 52        82 CreateMenu
 53        83 CreatePopupMenu
 54        84 CreateWindowExA
 55        85 CreateWindowExW
 56        86 CreateWindowStationA
 57        87 CreateWindowStationW
 58        88 DdeAbandonTransaction
 59        89 DdeAccessData
```

```
 5A         90 DdeAddData
 5B         91 DdeClientTransaction
 5C         92 DdeCmpStringHandles
 5D         93 DdeConnect
 5E         94 DdeConnectList
 5F         95 DdeCreateDataHandle
 60         96 DdeCreateStringHandleA
 61         97 DdeCreateStringHandleW
 62         98 DdeDisconnect
 63         99 DdeDisconnectList
 64        100 DdeEnableCallback
 65        101 DdeFreeDataHandle
 66        102 DdeFreeStringHandle
 67        103 DdeGetData
 68        104 DdeGetLastError
 69        105 DdeImpersonateClient
 6A        106 DdeInitializeA
 6B        107 DdeInitializeW
 6C        108 DdeKeepStringHandle
 6D        109 DdeNameService
 6E        110 DdePostAdvise
 6F        111 DdeQueryConvInfo
 70        112 DdeQueryNextServer
 71        113 DdeQueryStringA
 72        114 DdeQueryStringW
 73        115 DdeReconnect
 74        116 DdeSetQualityOfService
 75        117 DdeSetUserHandle
 76        118 DdeUnaccessData
 77        119 DdeUninitialize
 78        120 DefDlgProcA
 79        121 DefDlgProcW
 7A        122 DefFrameProcA
 7B        123 DefFrameProcW
 7C        124 DefMDIChildProcA
 7D        125 DefMDIChildProcW
 7E        126 DefWindowProcA
 7F        127 DefWindowProcW
 80        128 DeferWindowPos
 81        129 DeleteMenu
 82        130 DestroyAcceleratorTable
 83        131 DestroyCaret
 84        132 DestroyCursor
 85        133 DestroyIcon
 86        134 DestroyMenu
 87        135 DestroyWindow
 88        136 DialogBoxIndirectParamA
 89        137 DialogBoxIndirectParamW
```

```
 8A       138 DialogBoxParamA
 8B       139 DialogBoxParamW
 8C       140 DispatchMessageA
 8D       141 DispatchMessageW
 8E       142 DlgDirListA
 8F       143 DlgDirListComboBoxA
 90       144 DlgDirListComboBoxW
 91       145 DlgDirListW
 92       146 DlgDirSelectComboBoxExA
 93       147 DlgDirSelectComboBoxExW
 94       148 DlgDirSelectExA
 95       149 DlgDirSelectExW
 96       150 DragDetect
 97       151 DragObject
 98       152 DrawAnimatedRects
 99       153 DrawCaption
 9A       154 DrawCaptionTempA
 9B       155 DrawCaptionTempW
 9C       156 DrawEdge
 9D       157 DrawFocusRect
 9E       158 DrawFrame
 9F       159 DrawFrameControl
 A0       160 DrawIcon
 A1       161 DrawIconEx
 A2       162 DrawMenuBar
 A3       163 DrawStateA
 A4       164 DrawStateW
 A5       165 DrawTextA
 A6       166 DrawTextExA
 A7       167 DrawTextExW
 A8       168 DrawTextW
 A9       169 EditWndProc
 AA       170 EmptyClipboard
 AB       171 EnableMenuItem
 AC       172 EnableScrollBar
 AD       173 EnableWindow
 AE       174 EndDeferWindowPos
 AF       175 EndDialog
 B0       176 EndPaint
 B1       177 EndTask
 B2       178 EnumChildWindows
 B3       179 EnumClipboardFormats
 B4       180 EnumDesktopWindows
 B5       181 EnumDesktopsA
 B6       182 EnumDesktopsW
 B7       183 EnumDisplaySettingsA
 B8       184 EnumDisplaySettingsW
 B9       185 EnumPropsA
```

```
BA       186 EnumPropsExA
BB       187 EnumPropsExW
BC       188 EnumPropsW
BD       189 EnumThreadWindows
BE       190 EnumWindowStationsA
BF       191 EnumWindowStationsW
C0       192 EnumWindows
C1       193 EqualRect
C2       194 ExcludeUpdateRgn
C3       195 ExitWindowsEx
C4       196 FillRect
C5       197 FindWindowA
C6       198 FindWindowExA
C7       199 FindWindowExW
C8       200 FindWindowW
C9       201 FlashWindow
CA       202 FrameRect
CB       203 FreeDDElParam
CC       204 GetActiveWindow
CD       205 GetAsyncKeyState
CE       206 GetCapture
CF       207 GetCaretBlinkTime
D0       208 GetCaretPos
D1       209 GetClassInfoA
D2       210 GetClassInfoExA
D3       211 GetClassInfoExW
D4       212 GetClassInfoW
D5       213 GetClassLongA
D6       214 GetClassLongW
D7       215 GetClassNameA
D8       216 GetClassNameW
D9       217 GetClassWord
DA       218 GetClientRect
DB       219 GetClipCursor
DC       220 GetClipboardData
DD       221 GetClipboardFormatNameA
DE       222 GetClipboardFormatNameW
DF       223 GetClipboardOwner
E0       224 GetClipboardViewer
E1       225 GetCursor
E2       226 GetCursorPos
E3       227 GetDC
E4       228 GetDCEx
E5       229 GetDesktopWindow
E6       230 GetDialogBaseUnits
E7       231 GetDlgCtrlID
E8       232 GetDlgItem
E9       233 GetDlgItemInt
```

```
 EA       234 GetDlgItemTextA
 EB       235 GetDlgItemTextW
 EC       236 GetDoubleClickTime
 ED       237 GetFocus
 EE       238 GetForegroundWindow
 EF       239 GetIconInfo
 F0       240 GetInputDesktop
 F1       241 GetInputState
 F2       242 GetInternalWindowPos
 F3       243 GetKBCodePage
 F4       244 GetKeyNameTextA
 F5       245 GetKeyNameTextW
 F6       246 GetKeyState
 F7       247 GetKeyboardLayout
 F8       248 GetKeyboardLayoutList
 F9       249 GetKeyboardLayoutNameA
 FA       250 GetKeyboardLayoutNameW
 FB       251 GetKeyboardState
 FC       252 GetKeyboardType
 FD       253 GetLastActivePopup
 FE       254 GetMenu
 FF       255 GetMenuCheckMarkDimensions
100       256 GetMenuContextHelpId
101       257 GetMenuDefaultItem
102       258 GetMenuItemCount
103       259 GetMenuItemID
104       260 GetMenuItemInfoA
105       261 GetMenuItemInfoW
106       262 GetMenuItemRect
107       263 GetMenuState
108       264 GetMenuStringA
109       265 GetMenuStringW
10A       266 GetMessageA
10B       267 GetMessageExtraInfo
10C       268 GetMessagePos
10D       269 GetMessageTime
10E       270 GetMessageW
10F       271 GetNextDlgGroupItem
110       272 GetNextDlgTabItem
111       273 GetNextQueueWindow
112       274 GetOpenClipboardWindow
113       275 GetParent
114       276 GetPriorityClipboardFormat
115       277 GetProcessWindowStation
116       278 GetPropA
117       279 GetPropW
118       280 GetQueueStatus
119       281 GetScrollInfo
```

```
11A       282 GetScrollPos
11B       283 GetScrollRange
11C       284 GetShellWindow
11D       285 GetSubMenu
11E       286 GetSysColor
11F       287 GetSysColorBrush
120       288 GetSystemMenu
121       289 GetSystemMetrics
122       290 GetTabbedTextExtentA
123       291 GetTabbedTextExtentW
124       292 GetThreadDesktop
125       293 GetTopWindow
126       294 GetUpdateRect
127       295 GetUpdateRgn
128       296 GetUserObjectInformationA
129       297 GetUserObjectInformationW
12A       298 GetUserObjectSecurity
12B       299 GetWindow
12C       300 GetWindowContextHelpId
12D       301 GetWindowDC
12E       302 GetWindowLongA
12F       303 GetWindowLongW
130       304 GetWindowPlacement
131       305 GetWindowRect
132       306 GetWindowRgn
133       307 GetWindowTextA
134       308 GetWindowTextLengthA
135       309 GetWindowTextLengthW
136       310 GetWindowTextW
137       311 GetWindowThreadProcessId
138       312 GetWindowWord
139       313 GrayStringA
13A       314 GrayStringW
13B       315 HideCaret
13C       316 HiliteMenuItem
13D       317 ImpersonateDdeClientWindow
13E       318 InSendMessage
13F       319 InflateRect
140       320 InitSharedTable
141       321 InitTask
142       322 InsertMenuA
143       323 InsertMenuItemA
144       324 InsertMenuItemW
145       325 InsertMenuW
146       326 InternalGetWindowText
147       327 IntersectRect
148       328 InvalidateRect
149       329 InvalidateRgn
```

```
14A       330 InvertRect
14B       331 IsCharAlphaA
14C       332 IsCharAlphaNumericA
14D       333 IsCharAlphaNumericW
14E       334 IsCharAlphaW
14F       335 IsCharLowerA
150       336 IsCharLowerW
151       337 IsCharUpperA
152       338 IsCharUpperW
153       339 IsChild
154       340 IsClipboardFormatAvailable
155       341 IsDialogMessage
156       342 IsDialogMessageA
157       343 IsDialogMessageW
158       344 IsDlgButtonChecked
159       345 IsHungThread
15A       346 IsIconic
15B       347 IsMenu
15C       348 IsRectEmpty
15D       349 IsWindow
15E       350 IsWindowEnabled
15F       351 IsWindowUnicode
160       352 IsWindowVisible
161       353 IsZoomed
162       354 KillTimer
163       355 LoadAcceleratorsA
164       356 LoadAcceleratorsW
165       357 LoadBitmapA
166       358 LoadBitmapW
167       359 LoadCursorA
168       360 LoadCursorFromFileA
169       361 LoadCursorFromFileW
16A       362 LoadCursorW
16B       363 LoadIconA
16C       364 LoadIconW
16D       365 LoadImageA
16E       366 LoadImageW
16F       367 LoadKeyboardLayoutA
170       368 LoadKeyboardLayoutW
171       369 LoadMenuA
172       370 LoadMenuIndirectA
173       371 LoadMenuIndirectW
174       372 LoadMenuW
175       373 LoadStringA
176       374 LoadStringW
177       375 LockWindowStation
178       376 LockWindowUpdate
179       377 LookupIconIdFromDirectory
```

```
17A       378 LookupIconIdFromDirectoryEx
17B       379 MapDialogRect
17C       380 MapVirtualKeyA
17D       381 MapVirtualKeyExA
17E       382 MapVirtualKeyExW
17F       383 MapVirtualKeyW
180       384 MapWindowPoints
181       385 MenuItemFromPoint
182       386 MessageBeep
183       387 MessageBoxA
184       388 MessageBoxExA
185       389 MessageBoxExW
186       390 MessageBoxIndirectA
187       391 MessageBoxIndirectW
188       392 MessageBoxW
189       393 ModifyAccess
18A       394 ModifyMenuA
18B       395 ModifyMenuW
18C       396 MoveWindow
18D       397 MsgWaitForMultipleObjects
18E       398 OemKeyScan
18F       399 OemToCharA
190       400 OemToCharBuffA
191       401 OemToCharBuffW
192       402 OemToCharW
193       403 OffsetRect
194       404 OpenClipboard
195       405 OpenDesktopA
196       406 OpenDesktopW
197       407 OpenIcon
198       408 OpenInputDesktop
199       409 OpenWindowStationA
19A       410 OpenWindowStationW
19B       411 PackDDElParam
19C       412 PaintDesktop
19D       413 PeekMessageA
19E       414 PeekMessageW
19F       415 PlaySoundEvent
1A0       416 PostMessageA
1A1       417 PostMessageW
1A2       418 PostQuitMessage
1A3       419 PostThreadMessageA
1A4       420 PostThreadMessageW
1A5       421 PtInRect
1A6       422 RedrawWindow
1A7       423 RegisterClassA
1A8       424 RegisterClassExA
1A9       425 RegisterClassExW
```

```
1AA       426 RegisterClassW
1AB       427 RegisterClipboardFormatA
1AC       428 RegisterClipboardFormatW
1AD       429 RegisterHotKey
1AE       430 RegisterLogonProcess
1AF       431 RegisterNetworkCapabilities
1B0       432 RegisterSystemThread
1B1       433 RegisterTasklist
1B2       434 RegisterWindowMessageA
1B3       435 RegisterWindowMessageW
1B4       436 ReleaseCapture
1B5       437 ReleaseDC
1B6       438 RemoveMenu
1B7       439 RemovePropA
1B8       440 RemovePropW
1B9       441 ReplyMessage
1BA       442 ReuseDDElParam
1BB       443 ScreenToClient
1BC       444 ScrollDC
1BD       445 ScrollWindow
1BE       446 ScrollWindowEx
1BF       447 SendDlgItemMessageA
1C0       448 SendDlgItemMessageW
1C1       449 SendMessageA
1C2       450 SendMessageCallbackA
1C3       451 SendMessageCallbackW
1C4       452 SendMessageTimeoutA
1C5       453 SendMessageTimeoutW
1C6       454 SendMessageW
1C7       455 SendNotifyMessageA
1C8       456 SendNotifyMessageW
1C9       457 SetActiveWindow
1CA       458 SetCapture
1CB       459 SetCaretBlinkTime
1CC       460 SetCaretPos
1CD       461 SetClassLongA
1CE       462 SetClassLongW
1CF       463 SetClassWord
1D0       464 SetClipboardData
1D1       465 SetClipboardViewer
1D2       466 SetCursor
1D3       467 SetCursorPos
1D4       468 SetDebugErrorLevel
1D5       469 SetDeskWallpaper
1D6       470 SetDesktopBitmap
1D7       471 SetDlgItemInt
1D8       472 SetDlgItemTextA
1D9       473 SetDlgItemTextW
```

```
1DA       474 SetDoubleClickTime
1DB       475 SetFocus
1DC       476 SetForegroundWindow
1DD       477 SetInternalWindowPos
1DE       478 SetKeyboardState
1DF       479 SetLastErrorEx
1E0       480 SetLogonNotifyWindow
1E1       481 SetMenu
1E2       482 SetMenuContextHelpId
1E3       483 SetMenuDefaultItem
1E4       484 SetMenuItemBitmaps
1E5       485 SetMenuItemInfoA
1E6       486 SetMenuItemInfoW
1E7       487 SetMessageExtraInfo
1E8       488 SetMessageQueue
1E9       489 SetParent
1EA       490 SetProcessWindowStation
1EB       491 SetPropA
1EC       492 SetPropW
1ED       493 SetRect
1EE       494 SetRectEmpty
1EF       495 SetScrollInfo
1F0       496 SetScrollPos
1F1       497 SetScrollRange
1F2       498 SetShellWindow
1F3       499 SetSysColors
1F4       500 SetSysColorsTemp
1F5       501 SetSystemCursor
1F6       502 SetThreadDesktop
1F7       503 SetTimer
1F8       504 SetUserObjectInformationA
1F9       505 SetUserObjectInformationW
1FA       506 SetUserObjectSecurity
1FB       507 SetWindowContextHelpId
1FC       508 SetWindowFullScreenState
1FD       509 SetWindowLongA
1FE       510 SetWindowLongW
1FF       511 SetWindowPlacement
200       512 SetWindowPos
201       513 SetWindowRgn
202       514 SetWindowTextA
203       515 SetWindowTextW
204       516 SetWindowWord
205       517 SetWindowsHookA
206       518 SetWindowsHookExA
207       519 SetWindowsHookExW
208       520 SetWindowsHookW
209       521 ShowCaret
```

```
20A       522 ShowCursor
20B       523 ShowOwnedPopups
20C       524 ShowScrollBar
20D       525 ShowWindow
20E       526 ShowWindowAsync
20F       527 SubtractRect
210       528 SwapMouseButton
211       529 SwitchDesktop
212       530 SwitchToThisWindow
213       531 SysErrorBox
214       532 SystemParametersInfoA
215       533 SystemParametersInfoW
216       534 TabbedTextOutA
217       535 TabbedTextOutW
218       536 TileChildWindows
219       537 TileWindows
21A       538 ToAscii
21B       539 ToAsciiEx
21C       540 ToUnicode
21D       541 TrackPopupMenu
21E       542 TrackPopupMenuEx
21F       543 TranslateAccelerator
220       544 TranslateAcceleratorA
221       545 TranslateAcceleratorW
222       546 TranslateMDISysAccel
223       547 TranslateMessage
224       548 UnhookWindowsHook
225       549 UnhookWindowsHookEx
226       550 UnionRect
227       551 UnloadKeyboardLayout
228       552 UnlockWindowStation
229       553 UnpackDDElParam
22A       554 UnregisterClassA
22B       555 UnregisterClassW
22C       556 UnregisterHotKey
22D       557 UpdateWindow
22E       558 UserClientDllInitialize
22F       559 UserSignalProc
230       560 ValidateRect
231       561 ValidateRgn
232       562 VkKeyScanA
233       563 VkKeyScanExA
234       564 VkKeyScanExW
235       565 VkKeyScanW
236       566 WNDPROC_CALLBACK
237       567 WaitForInputIdle
238       568 WaitMessage
239       569 WinHelpA
```

```
23A       570 WinHelpW
23B       571 WinOldAppHackoMatic
23C       572 WindowFromDC
23D       573 WindowFromPoint
23E       574 YieldTask
23F       575 keybd_event
240       576 mouse_event
241       577 wsprintfA
242       578 wsprintfW
243       579 wvsprintfA
244       580 wvsprintfW
```

```
ord seg offset name

  1   1  86bc  SETBKCOLOR
  2   1  86f0  SETBKMODE
  3   1  8126  SETMAPMODE
  4   1  8775  SETROP2
  5   1  9934  SETRELABS
  6   1  87ff  SETPOLYFILLMODE
  7   1  87ba  SETSTRETCHBLTMODE
  8   1  7c64  SETTEXTCHARACTEREXTRA
  9   1  868a  SETTEXTCOLOR
 10   1  7c36  SETTEXTJUSTIFICATION
 11   1  8844  SETWINDOWORG
 12   6  0655  SETWINDOWEXT
 13   1  8876  SETVIEWPORTORG
 14   6  0686  SETVIEWPORTEXT
 15   1  80f5  OFFSETWINDOWORG
 16   6  06e8  SCALEWINDOWEXT
 17   1  80c4  OFFSETVIEWPORTORG
 18   6  06b7  SCALEVIEWPORTEXT
 19   4  07a5  LINETO
 20   4  07d7  MOVETO
 21   1  82b3  EXCLUDECLIPRECT
 22   1  8352  INTERSECTCLIPRECT
 23  15  166a  ARC
 24  15  1639  ELLIPSE
 25  19  08ff  FLOODFILL
 26  15  169b  PIE
 27  10  0000  RECTANGLE
 28  15  16fd  ROUNDRECT
 29   1  b216  PATBLT
 30   1  7a4a  SAVEDC
 31  24  002e  SETPIXEL
 32   1  8506  OFFSETCLIPRGN
 33   1  8924  TEXTOUT
 34   1  821d  BITBLT
 35   1  81e5  STRETCHBLT
 36  25  0000  POLYGON
 37   4  0847  POLYLINE
 38   8  0d94  ESCAPE
 39   1  7a19  RESTOREDC
 40  26  0000  FILLRGN
 41  26  0076  FRAMERGN
 42   1  8613  INVERTRGN
 43  26  003e  PAINTRGN
 44   1  83b6  SELECTCLIPRGN
 45   1  864b  SELECTOBJECT
```

```
46   1  0000  __GP
47   1  8538  COMBINERGN
48   1  7b2d  CREATEBITMAP
49   1  7b76  CREATEBITMAPINDIRECT
50   6  048a  CREATEBRUSHINDIRECT
51   6  04c7  CREATECOMPATIBLEBITMAP
52   1  787a  CREATECOMPATIBLEDC
53   1  7911  CREATEDC
54  15  0340  CREATEELLIPTICRGN
55  15  1537  CREATEELLIPTICRGNINDIRECT
56   5  2b75  CREATEFONT
57   5  2be5  CREATEFONTINDIRECT
58   6  040e  CREATEHATCHBRUSH
59   1  64d8  WEP
60   6  0452  CREATEPATTERNBRUSH
61   1  7aa9  CREATEPEN
62   1  7af0  CREATEPENINDIRECT
63  15  1574  CREATEPOLYGONRGN
64   1  c071  CREATERECTRGN
65   1  83ee  CREATERECTRGNINDIRECT
66   6  0b59  CREATESOLIDBRUSH
67   1  7c92  DPTOLP
68   1  79df  DELETEDC
69   1  7a7b  DELETEOBJECT
70  12  0828  ENUMFONTS
71   6  05d1  ENUMOBJECTS
72   1  842c  EQUALRGN
73   1  0538  EXCLUDEVISRECT
74   1  7ff4  GETBITMAPBITS
75   1  7d96  GETBKCOLOR
76   1  7dc4  GETBKMODE
77   1  8317  GETCLIPBOX
78   1  7d0c  GETCURRENTPOSITION
79   1  7d3a  GETDCORG
80   1  802e  GETDEVICECAPS
81   1  7ed8  GETMAPMODE
82   6  061b  GETOBJECT
83  24  0000  GETPIXEL
84   1  7eaa  GETPOLYFILLMODE
85   1  7e4e  GETROP2
86   1  5fbd  GETRELABS
87   1  7fbe  GETSTOCKOBJECT
88   1  7e7c  GETSTRETCHBLTMODE
89   7  0000  GETTEXTCHARACTEREXTRA
90   1  7df2  GETTEXTCOLOR
91   1  81a4  GETTEXTEXTENT
92   5  2f41  GETTEXTFACE
93   5  2f8c  GETTEXTMETRICS
```

94	1	7f90	GETVIEWPORTEXT
95	1	7f62	GETVIEWPORTORG
96	1	7f34	GETWINDOWEXT
97	1	7f06	GETWINDOWORG
98	1	0534	INTERSECTVISRECT
99	1	7ccf	LPTODP
100	24	0063	LINEDDA
101	1	84d7	OFFSETRGN
102	1	ca6d	OFFSETVISRGN
103	1	85e4	PTVISIBLE
104	1	849c	RECTVISIBLE
105	1	04a0	SELECTVISRGN
106	1	805c	SETBITMAPBITS
117	1	e530	SETDCORG
119	11	1c31	ADDFONTRESOURCE
121	1	6863	DEATH
122	1	68aa	RESURRECTION
123	22	002e	PLAYMETAFILE
124	22	02e5	GETMETAFILE
125	23	003a	CREATEMETAFILE
126	23	0077	CLOSEMETAFILE
127	22	0000	DELETEMETAFILE
128	1	9810	MULDIV
129	1	2c95	SAVEVISRGN
130	1	2d22	RESTOREVISRGN
131	1	6809	INQUIREVISRGN
132	13	04f3	SETENVIRONMENT
133	13	04a2	GETENVIRONMENT
134	1	816a	GETRGNBOX
135	19	09d8	SCANLR
136	14	06bd	REMOVEFONTRESOURCE
148	1	8255	SETBRUSHORG
149	1	7d68	GETBRUSHORG
150	1	8284	UNREALIZEOBJECT
151	23	0000	COPYMETAFILE
153	1	78b2	CREATEIC
154	1	8096	GETNEARESTCOLOR
155	8	0111	QUERYABORT
156	6	0505	CREATEDISCARDABLEBITMAP
159	23	00a5	GETMETAFILEBITS
160	23	00d3	SETMETAFILEBITS
161	1	85b5	PTINREGION
162	1	7c08	GETBITMAPDIMENSION
163	1	7bda	SETBITMAPDIMENSION
169	1	910e	ISDCDIRTY
170	1	9175	SETDCSTATUS
172	1	8586	SETRECTRGN
173	1	5f86	GETCLIPRGN

```
175  22  0129  ENUMMETAFILE
176  22  009d  PLAYMETAFILERECORD
179   1  269d  GETDCSTATE                nodoc
180   1  26c2  SETDCSTATE
181   1  8461  RECTINREGION
188   5  2d7f  GETTEXTEXTENTEX
190   8  2310  SETDCHOOK
191   8  22c2  GETDCHOOK
192   1  c5ac  SETHOOKFLAGS
193   1  c758  SETBOUNDSRECT
194   1  c60a  GETBOUNDSRECT
195   1  a9f6  SELECTBITMAP
196  23  0101  SETMETAFILEBITSBETTER
201  18  0137  DMBITBLT
202  18  013c  DMCOLORINFO
206  18  0150  DMENUMDFONTS
207  18  0156  DMENUMOBJ
208  18  0146  DMOUTPUT
209  18  0141  DMPIXEL
210  18  014b  DMREALIZEOBJECT
211  18  00db  DMSTRBLT
212  18  015b  DMSCANLR
214  18  0000  DMEXTTEXTOUT
215  18  011e  DMGETCHARWIDTH
216  18  0123  DMSTRETCHBLT
217  18  0128  DMDIBBITS
218  18  012d  DMSTRETCHDIBITS
219  18  0132  DMSETDIBTODEV
220  18  029d  DMTRANSPOSE
230  27  0000  CREATEPQ
231  27  00db  MINPQ
232  27  00df  EXTRACTPQ
233  27  0039  INSERTPQ
234  27  01a3  SIZEPQ
235  27  0192  DELETEPQ
240   8  26c4  OPENJOB
241   8  2b9e  WRITESPOOL
242   8  2d0a  WRITEDIALOG
243   8  2e6a  CLOSEJOB
244   8  2f6e  DELETEJOB
246   8  2b0c  STARTSPOOLPAGE
247   8  2de2  ENDSPOOLPAGE
248   8  321e  QUERYJOB
250   8  0d51  COPY
253   8  2dd6  DELETESPOOLPAGE
254   8  3132  SPOOLFILE
266  42  0883  OPENPRINTERA
267  42  0879  STARTDOCPRINTERA
```

```
268  42  0874  STARTPAGEPRINTER
269  42  086f  WRITEPRINTER
270  42  086a  ENDPAGEPRINTER
271  42  085a  ABORTPRINTER
272  42  0865  ENDDOCPRINTER
274  42  0860  CLOSEPRINTER
280   8  374c  GETREALDRIVERINFO
281   8  3d80  DRVSETPRINTERDATA
282   8  3ea2  DRVGETPRINTERDATA
299   5  4148  ENGINEGETCHARWIDTHEX
300  12  09eb  ENGINEENUMERATEFONT
301  14  079a  ENGINEDELETEFONT
302   5  3f7d  ENGINEREALIZEFONT
303   5  4148  ENGINEGETCHARWIDTH
304  12  0a56  ENGINESETFONTCONTEXT
305  13  1709  ENGINEGETGLYPHBMP
306  13  0b32  ENGINEMAKEFONTDIR
307  13  065f  GETCHARABCWIDTHS
308  13  06b2  GETOUTLINETEXTMETRICS
309  13  0581  GETGLYPHOUTLINE
310  13  0534  CREATESCALABLEFONTRESOURCE
311  13  0603  GETFONTDATA
312  37  0edd  CONVERTOUTLINEFONTFILE
313  13  0865  GETRASTERIZERCAPS
314  37  0d73  ENGINEEXTTEXTOUT
315   5  3dec  ENGINEREALIZEFONTEXT
316   5  41f9  ENGINEGETCHARWIDTHSTR
317  13  1734  ENGINEGETGLYPHBMPEXT
330  12  0872  ENUMFONTFAMILIES
332  13  06ed  GETKERNINGPAIRS
345   1  7e20  GETTEXTALIGN
346   1  8735  SETTEXTALIGN
348  15  16cc  CHORD
349   5  2cfb  SETMAPPERFLAGS
350   5  2c22  GETCHARWIDTH
351   1  88a8  EXTTEXTOUT
352   1  33e0  GETPHYSICALFONTHANDLE
353   5  2ccd  GETASPECTRATIOFILTER
354   1  6808  SHRINKGDIHEAP
355  46  01dd  FTRAPPING0
360   2  01f1  CREATEPALETTE
361   2  0629  GDISELECTPALETTE
362   2  1f08  GDIREALIZEPALETTE
363   2  027b  GETPALETTEENTRIES
364   2  037f  SETPALETTEENTRIES
365   2  1ea1  REALIZEDEFAULTPALETTE
366   2  03ea  UPDATECOLORS
367   2  01b4  ANIMATEPALETTE
```

```
368   2  0351  RESIZEPALETTE
370   2  02b8  GETNEARESTPALETTEINDEX
372  19  092d  EXTFLOODFILL
373   2  03bc  SETSYSTEMPALETTEUSE
374   2  0323  GETSYSTEMPALETTEUSE
375   2  02e6  GETSYSTEMPALETTEENTRIES
376   8  0f1c  RESETDC
377   8  0e77  STARTDOC
378   8  0e46  ENDDOC
379   8  0eb4  STARTPAGE
380   8  0e15  ENDPAGE
381   8  0ee2  SETABORTPROC
382   8  0de4  ABORTDOC
400   1  3888  FASTWINDOWFRAME
401  11  1c9d  GDIMOVEBITMAP
403  11  1cab  GDIINIT2
404  37  1eb3  GETTTGLYPHINDEXMAP
405  11  0d68  FINALGDIINIT
407   1  a600  CREATEUSERBITMAP
409   6  0cd0  CREATEUSERDISCARDABLEBITMAP
410  22  1a9e  ISVALIDMETAFILE
411   1  5f60  GETCURLOGFONT
412   2  156d  ISDCCURRENTPALETTE
439   3  2cdc  STRETCHDIBITS
440   3  2a00  SETDIBITS
441   3  2c4a  GETDIBITS
442   3  2a92  CREATEDIBITMAP
443   3  2b8f  SETDIBITSTODEVICE
444  15  02ee  CREATEROUNDRECTRGN
445   6  0543  CREATEDIBPATTERNBRUSH
449   3  220d  DEVICECOLORMATCH
450  25  004f  POLYPOLYGON
451  15  15d7  CREATEPOLYPOLYGONRGN
452   4  2318  GDISEEGDIDO
461   1  8a63  SETOBJECTOWNER
462   1  64e3  ISGDIOBJECT
463   1  a576  MAKEOBJECTPRIVATE
464  22  1cc7  FIXUPBOGUSPUBLISHERMETAFILE
465   1  8f45  RECTVISIBLE_EHH
466   1  8f48  RECTINREGION_EHH
467   1  ebbd  UNICODETOANSI
468  40  003a  GETBITMAPDIMENSIONEX
469  40  00ae  GETBRUSHORGEX
470  40  0074  GETCURRENTPOSITIONEX
471  40  0378  GETTEXTEXTENTPOINT
472  40  0196  GETVIEWPORTEXTEX
473  40  015c  GETVIEWPORTORGEX
474  40  0122  GETWINDOWEXTEX
```

475	40	00e8	GETWINDOWORGEX
476	40	020a	OFFSETVIEWPORTORGEX
477	40	0247	OFFSETWINDOWORGEX
478	40	0000	SETBITMAPDIMENSIONEX
479	40	02c1	SETVIEWPORTEXTEX
480	40	0446	SETVIEWPORTORGEX
481	40	0284	SETWINDOWEXTEX
482	40	0408	SETWINDOWORGEX
483	4	0809	MOVETOEX
484	40	02fe	SCALEVIEWPORTEXTEX
485	40	033b	SCALEWINDOWEXTEX
486	40	01d0	GETASPECTRATIOFILTEREX
489	3	0000	CREATEDIBSECTION
490	34	efd4	CLOSEENHMETAFILE
491	34	f00d	COPYENHMETAFILE
492	34	f04e	CREATEENHMETAFILE
493	34	f0b1	DELETEENHMETAFILE
495	34	f138	GDICOMMENT
496	34	f175	GETENHMETAFILE
497	34	f1b3	GETENHMETAFILEBITS
498	34	f1e7	GETENHMETAFILEDESCRIPTION
499	34	f21f	GETENHMETAFILEHEADER
501	34	f253	GETENHMETAFILEPALETTEENTRIES
502	31	0000	POLYBEZIER
503	31	0061	POLYBEZIERTO
504	34	f28f	PLAYENHMETAFILERECORD
505	34	f2f1	SETENHMETAFILEBITS
506	1	89aa	SETMETARGN
508	1	895f	EXTSELECTCLIPRGN
511	33	0000	ABORTPATH
512	33	0032	BEGINPATH
513	33	0064	CLOSEFIGURE
514	33	0096	ENDPATH
515	33	00c8	FILLPATH
516	33	00fa	FLATTENPATH
517	33	012c	GETPATH
518	33	01a5	PATHTOREGION
519	33	01e1	SELECTCLIPPATH
520	33	0226	STROKEANDFILLPATH
521	33	0258	STROKEPATH
522	33	028a	WIDENPATH
523	33	03a9	EXTCREATEPEN
524	33	02bc	GETARCDIRECTION
525	33	02eb	SETARCDIRECTION
526	33	0330	GETMITERLIMIT
527	33	036b	SETMITERLIMIT
528	11	20a8	GDIPARAMETERSINFO
529	2	0243	CREATEHALFTONEPALETTE

```
602   3  2dfb  SETDIBCOLORTABLE
603   3  2e39  GETDIBCOLORTABLE
604  41  0000  SETSOLIDBRUSH
605   1  8a92  SYSDELETEOBJECT
606   1  8ac1  SETMAGICCOLORS
607   1  89dc  GETREGIONDATA
608   1  8a16  EXTCREATEREGION
609   1  758b  GDIFREERESOURCES
610   1  e42c  GDISIGNALPROC32
611   1  8af0  GETRANDOMRGN
612   5  01e8  GETTEXTCHARSET
613  12  08b5  ENUMFONTFAMILIESEX
614  11  108d  ADDLPKTOGDI
615   5  2de1  GETCHARACTERPLACEMENT
616   5  2ed5  GETFONTLANGUAGEINFO
650   3  11d9  BUILDINVERSETABLEDIB
701  42  09d3  GDITHKCONNECTIONDATALS
702  43  030c  FT_GDIFTHKTHKCONNECTIONDATA
703  42  0822  FDTHKCONNECTIONDATASL
704  42  2552  ICMTHKCONNECTIONDATASL
820  42  2578  ICMCREATETRANSFORM
821  42  2573  ICMDELETETRANSFORM
822  42  256e  ICMTRANSLATERGB
823  42  2569  ICMTRANSLATERGBS
824  42  2564  ICMCHECKCOLORSINGAMUT
```

USER.EXE export functions

1	MESSAGEBOX
2	OLDEXITWINDOWS
3	ENABLEOEMLAYER
4	DISABLEOEMLAYER
5	INITAPP
6	POSTQUITMESSAGE
7	EXITWINDOWS
8	BEAR8
10	SETTIMER
11	BEAR11
12	KILLTIMER
13	GETTICKCOUNT
14	GETTIMERRESOLUTION
15	GETCURRENTTIME
16	CLIPCURSOR
17	GETCURSORPOS
18	SETCAPTURE
19	RELEASECAPTURE
20	SETDOUBLECLICKTIME
21	GETDOUBLECLICKTIME
22	SETFOCUS
23	GETFOCUS
24	REMOVEPROP
25	GETPROP
26	SETPROP
27	ENUMPROPS
28	CLIENTTOSCREEN
29	SCREENTOCLIENT
30	WINDOWFROMPOINT
31	ISICONIC
32	GETWINDOWRECT
33	GETCLIENTRECT
34	ENABLEWINDOW
35	ISWINDOWENABLED
36	GETWINDOWTEXT
37	SETWINDOWTEXT
38	GETWINDOWTEXTLENGTH
39	BEGINPAINT
40	ENDPAINT
41	CREATEWINDOW
42	SHOWWINDOW
43	CLOSEWINDOW
44	OPENICON
45	BRINGWINDOWTOTOP
46	GETPARENT

47 ISWINDOW
48 ISCHILD
49 ISWINDOWVISIBLE
50 FINDWINDOW
51 BEAR51
52 ANYPOPUP
53 DESTROYWINDOW
54 ENUMWINDOWS
55 ENUMCHILDWINDOWS
56 MOVEWINDOW
57 REGISTERCLASS
58 GETCLASSNAME
59 SETACTIVEWINDOW
60 GETACTIVEWINDOW
61 SCROLLWINDOW
62 SETSCROLLPOS
63 GETSCROLLPOS
64 SETSCROLLRANGE
65 GETSCROLLRANGE
66 GETDC
67 GETWINDOWDC
68 RELEASEDC
69 SETCURSOR
70 SETCURSORPOS
71 SHOWCURSOR
72 SETRECT
73 SETRECTEMPTY
74 COPYRECT
75 ISRECTEMPTY
76 PTINRECT
77 OFFSETRECT
78 INFLATERECT
79 INTERSECTRECT
80 UNIONRECT
81 FILLRECT
82 INVERTRECT
83 FRAMERECT
84 DRAWICON
85 DRAWTEXT
86 BEAR86
87 DIALOGBOX
88 ENDDIALOG
89 CREATEDIALOG
90 ISDIALOGMESSAGE
91 GETDLGITEM
92 SETDLGITEMTEXT
93 GETDLGITEMTEXT
94 SETDLGITEMINT

95 GETDLGITEMINT
96 CHECKRADIOBUTTON
97 CHECKDLGBUTTON
98 ISDLGBUTTONCHECKED
99 DLGDIRSELECT
100 DLGDIRLIST
101 SENDDLGITEMMESSAGE
102 ADJUSTWINDOWRECT
103 MAPDIALOGRECT
104 MESSAGEBEEP
105 FLASHWINDOW
106 GETKEYSTATE
107 DEFWINDOWPROC
108 GETMESSAGE
109 PEEKMESSAGE
110 POSTMESSAGE
111 SENDMESSAGE
112 WAITMESSAGE
113 TRANSLATEMESSAGE
114 DISPATCHMESSAGE
115 REPLYMESSAGE
116 POSTAPPMESSAGE
117 WINDOWFROMDC
118 REGISTERWINDOWMESSAGE
119 GETMESSAGEPOS
120 GETMESSAGETIME
121 SETWINDOWSHOOK
122 CALLWINDOWPROC
123 CALLMSGFILTER
124 UPDATEWINDOW
125 INVALIDATERECT
126 INVALIDATERGN
127 VALIDATERECT
128 VALIDATERGN
129 GETCLASSWORD
130 SETCLASSWORD
131 GETCLASSLONG
132 SETCLASSLONG
133 GETWINDOWWORD
134 SETWINDOWWORD
135 GETWINDOWLONG
136 SETWINDOWLONG
137 OPENCLIPBOARD
138 CLOSECLIPBOARD
139 EMPTYCLIPBOARD
140 GETCLIPBOARDOWNER
141 SETCLIPBOARDDATA
142 GETCLIPBOARDDATA

143	COUNTCLIPBOARDFORMATS
144	ENUMCLIPBOARDFORMATS
145	REGISTERCLIPBOARDFORMAT
146	GETCLIPBOARDFORMATNAME
147	SETCLIPBOARDVIEWER
148	GETCLIPBOARDVIEWER
149	CHANGECLIPBOARDCHAIN
150	LOADMENU
151	CREATEMENU
152	DESTROYMENU
153	CHANGEMENU
154	CHECKMENUITEM
155	ENABLEMENUITEM
156	GETSYSTEMMENU
157	GETMENU
158	SETMENU
159	GETSUBMENU
160	DRAWMENUBAR
161	GETMENUSTRING
162	HILITEMENUITEM
163	CREATECARET
164	DESTROYCARET
165	SETCARETPOS
166	HIDECARET
167	SHOWCARET
168	SETCARETBLINKTIME
169	GETCARETBLINKTIME
170	ARRANGEICONICWINDOWS
171	WINHELP
172	SWITCHTOTHISWINDOW
173	LOADCURSOR
174	LOADICON
175	LOADBITMAP
176	LOADSTRING
177	LOADACCELERATORS
178	TRANSLATEACCELERATOR
179	GETSYSTEMMETRICS
180	GETSYSCOLOR
181	SETSYSCOLORS
182	BEAR182
183	GETCARETPOS
184	QUERYSENDMESSAGE
185	GRAYSTRING
186	SWAPMOUSEBUTTON
187	ENDMENU
188	SETSYSMODALWINDOW
189	GETSYSMODALWINDOW
190	GETUPDATERECT

191	CHILDWINDOWFROMPOINT
192	INSENDMESSAGE
193	ISCLIPBOARDFORMATAVAILABLE
194	DLGDIRSELECTCOMBOBOX
195	DLGDIRLISTCOMBOBOX
196	TABBEDTEXTOUT
197	GETTABBEDTEXTEXTENT
198	CASCADECHILDWINDOWS
199	TILECHILDWINDOWS
200	OPENCOMM
201	SETCOMMSTATE
202	GETCOMMSTATE
203	GETCOMMERROR
204	READCOMM
205	WRITECOMM
206	TRANSMITCOMMCHAR
207	CLOSECOMM
208	SETCOMMEVENTMASK
209	GETCOMMEVENTMASK
210	SETCOMMBREAK
211	CLEARCOMMBREAK
212	UNGETCOMMCHAR
213	BUILDCOMMDCB
214	ESCAPECOMMFUNCTION
215	FLUSHCOMM
216	USERSEEUSERDO
217	LOOKUPMENUHANDLE
218	DIALOGBOXINDIRECT
219	CREATEDIALOGINDIRECT
220	LOADMENUINDIRECT
221	SCROLLDC
222	GETKEYBOARDSTATE
223	SETKEYBOARDSTATE
224	GETWINDOWTASK
225	ENUMTASKWINDOWS
226	LOCKINPUT
227	GETNEXTDLGGROUPITEM
228	GETNEXTDLGTABITEM
229	GETTOPWINDOW
230	GETNEXTWINDOW
231	GETSYSTEMDEBUGSTATE
232	SETWINDOWPOS
233	SETPARENT
234	UNHOOKWINDOWSHOOK
235	DEFHOOKPROC
236	GETCAPTURE
237	GETUPDATERGN
238	EXCLUDEUPDATERGN

239 DIALOGBOXPARAM
240 DIALOGBOXINDIRECTPARAM
241 CREATEDIALOGPARAM
242 CREATEDIALOGINDIRECTPARAM
243 GETDIALOGBASEUNITS
244 EQUALRECT
245 ENABLECOMMNOTIFICATION
246 EXITWINDOWSEXEC
247 GETCURSOR
248 GETOPENCLIPBOARDWINDOW
249 GETASYNCKEYSTATE
250 GETMENUSTATE
251 SENDDRIVERMESSAGE
252 OPENDRIVER
253 CLOSEDRIVER
254 GETDRIVERMODULEHANDLE
255 DEFDRIVERPROC
256 GETDRIVERINFO
257 GETNEXTDRIVER
258 MAPWINDOWPOINTS
259 BEGINDEFERWINDOWPOS
260 DEFERWINDOWPOS
261 ENDDEFERWINDOWPOS
262 GETWINDOW
263 GETMENUITEMCOUNT
264 GETMENUITEMID
265 SHOWOWNEDPOPUPS
266 SETMESSAGEQUEUE
267 SHOWSCROLLBAR
268 GLOBALADDATOM
269 GLOBALDELETEATOM
270 GLOBALFINDATOM
271 GLOBALGETATOMNAME
272 ISZOOMED
273 CONTROLPANELINFO
274 GETNEXTQUEUEWINDOW
275 REPAINTSCREEN
276 LOCKMYTASK
277 GETDLGCTRLID
278 GETDESKTOPHWND
279 OLDSETDESKPATTERN
280 SETSYSTEMMENU
281 GETSYSCOLORBRUSH
282 SELECTPALETTE
283 REALIZEPALETTE
284 GETFREESYSTEMRESOURCES
285 BEAR285
286 GETDESKTOPWINDOW

287	GETLASTACTIVEPOPUP
288	GETMESSAGEEXTRAINFO
289	KEYBD_EVENT
290	REDRAWWINDOW
291	SETWINDOWSHOOKEX
292	UNHOOKWINDOWSHOOKEX
293	CALLNEXTHOOKEX
294	LOCKWINDOWUPDATE
299	MOUSE_EVENT
300	UNLOADINSTALLABLEDRIVERS
301	BOZOSLIVEHERE
306	BEAR306
308	DEFDLGPROC
309	GETCLIPCURSOR
319	SCROLLWINDOWEX
320	SYSERRORBOX
321	SETEVENTHOOK
322	WINOLDAPPHACKOMATIC
323	GETMESSAGE2
324	FILLWINDOW
325	PAINTRECT
326	GETCONTROLBRUSH
331	ENABLEHARDWAREINPUT
332	USERYIELD
333	ISUSERIDLE
334	GETQUEUESTATUS
335	GETINPUTSTATE
337	GETMOUSEEVENTPROC
358	ISMENU
359	GETDCEX
362	DCHOOK
364	LOOKUPICONIDFROMDIRECTORYEX
368	COPYICON
369	COPYCURSOR
370	GETWINDOWPLACEMENT
371	SETWINDOWPLACEMENT
372	GETINTERNALICONHEADER
373	SUBTRACTRECT
375	DRAWTEXTEX
376	SETMESSAGEEXTRAINFO
378	SETPROPEX
379	GETPROPEX
380	REMOVEPROPEX
381	USRMPR_THUNKDATA16
382	SETWINDOWCONTEXTHELPID
383	GETWINDOWCONTEXTHELPID
384	SETMENUCONTEXTHELPID
385	GETMENUCONTEXTHELPID

389 LOADIMAGE
390 COPYIMAGE
391 SIGNALPROC32
394 DRAWICONEX
395 GETICONINFO
397 REGISTERCLASSEX
398 GETCLASSINFOEX
399 CHILDWINDOWFROMPOINTEX
400 FINALUSERINIT
402 GETPRIORITYCLIPBOARDFORMAT
403 UNREGISTERCLASS
404 GETCLASSINFO
406 CREATECURSOR
407 CREATEICON
408 CREATECURSORICONINDIRECT
409 INITTHREADINPUT
410 INSERTMENU
411 APPENDMENU
412 REMOVEMENU
413 DELETEMENU
414 MODIFYMENU
415 CREATEPOPUPMENU
416 TRACKPOPUPMENU
417 GETMENUCHECKMARKDIMENSIONS
418 SETMENUITEMBITMAPS
420 _WSPRINTF
421 WVSPRINTF
422 DLGDIRSELECTEX
423 DLGDIRSELECTCOMBOBOXEX
427 FINDWINDOWEX
428 TILEWINDOWS
429 CASCADEWINDOWS
430 LSTRCMP
431 ANSIUPPER
432 ANSILOWER
433 ISCHARALPHA
434 ISCHARALPHANUMERIC
435 ISCHARUPPER
436 ISCHARLOWER
437 ANSIUPPERBUFF
438 ANSILOWERBUFF
441 INSERTMENUITEM
443 GETMENUITEMINFO
445 DEFFRAMEPROC
446 SETMENUITEMINFO
447 DEFMDICHILDPROC
448 DRAWANIMATEDRECTS
449 DRAWSTATE

450 CREATEICONFROMRESOURCEEX
451 TRANSLATEMDISYSACCEL
452 CREATEWINDOWEX
454 ADJUSTWINDOWRECTEX
455 GETICONID
456 LOADICONHANDLER
457 DESTROYICON
458 DESTROYCURSOR
459 DUMPICON
460 GETINTERNALWINDOWPOS
461 SETINTERNALWINDOWPOS
462 CALCCHILDSCROLL
463 SCROLLCHILDREN
464 DRAGOBJECT
465 DRAGDETECT
466 DRAWFOCUSRECT
470 STRINGFUNC
471 LSTRCMPI
472 ANSINEXT
473 ANSIPREV
475 SETSCROLLINFO
476 GETSCROLLINFO
477 GETKEYBOARDLAYOUTNAME
478 LOADKEYBOARDLAYOUT
479 MENUITEMFROMPOINT
480 GETUSERLOCALOBJTYPE
481 HARDWARE_EVENT
482 ENABLESCROLLBAR
483 SYSTEMPARAMETERSINFO
498 BEAR498
499 WNETERRORTEXT
501 WNETOPENJOB
502 WNETCLOSEJOB
503 WNETABORTJOB
504 WNETHOLDJOB
505 WNETRELEASEJOB
506 WNETCANCELJOB
507 WNETSETJOBCOPIES
508 WNETWATCHQUEUE
509 WNETUNWATCHQUEUE
510 WNETLOCKQUEUEDATA
511 WNETUNLOCKQUEUEDATA
512 WNETGETCONNECTION
513 WNETGETCAPS
514 WNETDEVICEMODE
515 WNETBROWSEDIALOG
516 WNETGETUSER
517 WNETADDCONNECTION

518 WNETCANCELCONNECTION
519 WNETGETERROR
520 WNETGETERRORTEXT
521 WNETENABLE
522 WNETDISABLE
523 WNETRESTORECONNECTION
524 WNETWRITEJOB
525 WNETCONNECTDIALOG
526 WNETDISCONNECTDIALOG
527 WNETCONNECTIONDIALOG
528 WNETVIEWQUEUEDIALOG
529 WNETPROPERTYDIALOG
530 WNETGETDIRECTORYTYPE
531 WNETDIRECTORYNOTIFY
532 WNETGETPROPERTYTEXT
533 WNETINITIALIZE
534 WNETLOGON
600 GETSHELLWINDOW
601 DOHOTKEYSTUFF
602 SETCHECKCURSORTIMER
604 BROADCASTSYSTEMMESSAGE
605 HACKTASKMONITOR
606 FORMATMESSAGE
608 GETFOREGROUNDWINDOW
609 SETFOREGROUNDWINDOW
610 DESTROYICON32
620 CHANGEDISPLAYSETTINGS
621 ENUMDISPLAYSETTINGS
640 MSGWAITFORMULTIPLEOBJECTS
650 ACTIVATEKEYBOARDLAYOUT
651 GETKEYBOARDLAYOUT
652 GETKEYBOARDLAYOUTLIST
654 UNLOADKEYBOARDLAYOUT
655 POSTPOSTEDMESSAGES
656 DRAWFRAMECONTROL
657 DRAWCAPTIONTEMP
658 DISPATCHINPUT
659 DRAWEDGE
660 DRAWCAPTION
661 SETSYSCOLORSTEMP
662 DRAWMENUBARTEMP
663 GETMENUDEFAULTITEM
664 SETMENUDEFAULTITEM
665 GETMENUITEMRECT
666 CHECKMENURADIOITEM
667 TRACKPOPUPMENUEX
668 SETWINDOWRGN
669 GETWINDOWRGN

800	CHOOSEFONT_CALLBACK16
801	FINDREPLACE_CALLBACK16
802	OPENFILENAME_CALLBACK16
803	PRINTDLG_CALLBACK16
804	CHOOSECOLOR_CALLBACK16
819	PEEKMESSAGE32
820	GETMESSAGE32
821	TRANSLATEMESSAGE32
822	DISPATCHMESSAGE32
823	CALLMSGFILTER32
824	ISDIALOGMESSAGE32
825	POSTMESSAGE32
826	POSTTHREADMESSAGE32
827	MESSAGEBOXINDIRECT
850	USRTHKCONNECTIONDATALS
851	MSGTHKCONNECTIONDATALS
853	FT_USRFTHKTHKCONNECTIONDATA
854	FT_USRF2THKTHKCONNECTIONDATA
855	USR32THKCONNECTIONDATASL
890	INSTALLIMT
891	UNINSTALLIMT

Int 22h

Int 22h service is a VMM fault hooked in VMM.vxd.

```
INT 22h - Win32 Protected mode interface requests API
        AX=02h -- Converts a physical address to a linear address in curent context
                  entry:
                      ECX=phisycal address
                  returns:
                      ESI=linear address
                      AX= 1 if success , otherwise 0
        AX=07h -- Check to see if an address is within a VxD object
                   entry:
                      DS:ESI = buffer to receive object name
                      BX  =  thread number
                      EDX  = linear address to query
                   returns:
                      If EAX == 0, EDX = base address of object
                      If EAX != 0, error
        AX=08h --  Get PDE for a specific context
                    entry:
                     BX = thread number
                     EDX = linear address
                    returns:
                     if EAX == 0, ECX = PDE
                     if EAX != 0, error
        AX=0Ah --    Get LDT base
                    entry:
                     BX = Thread ID
                    returns:
                      if EAX == 0
                      EDI = pointer to LDT
                      ECX = LDT limit
                      if EAX != 0, error
I find these:
        AX=00h -- Return AX=0f386h means debug installed.

        AX=01h -- Debug Query.
                entry:
                        esi: points to Dot cmd lines(after the dot).

        AX=02h -- stc and return

        AX=03h -- verify memory, esi:addr,cx:len

        AX=04h -- nothing
        AX=05h -- nothing

        AX=06h -- Get VxD service address
                entry:
                        ebx= VxD service num,(00010001 for Get_Cur_VM_Handle)
                returns:
                        eax= VxD service address, 0 for error


        AX=09h -- ?
                entry:


        AX=0bh -- Return AX= current Thread ID

        AX=0ch -- ?
                entry:
                        ebx=
```

in DDK95's help, describes

Programmer's guide/Chapter 6: Windows Interrupt 2Fh interface

Service Functions Get Windows Installed State (Interrupt 2Fh Function 1600h)
Get Windows Entry-Point Address (Interrupt 2Fh Function 1602h)
Release Current VM Time-Slice (Interrupt 2Fh Function 1680h)
Begin Critical Section (Interrupt 2Fh Function 1681h)
End Critical Section (Interrupt 2Fh Function 1682h)
Get Current Virtual Machine ID (Interrupt 2Fh Function 1683h)
Get Device Entry Point Address (Interrupt 2Fh Function 1684h)
Switch VMs and CallBack (Interrupt 2Fh Function 1685h)
Detect Interrupt 31h Services (Interrupt 2Fh Function 1686h)

Notification Functions
Windows Initialization Notification (Interrupt 2Fh Function 1605)
Windows Termination Notification (Interrupt 2Fh Function 1606h)
Device Call Out (Interrupt 2Fh Function 1607h)
Windows Initialization Complete Notification (Interrupt 2Fh Function 1608h)
Windows Begin Exit (Interrupt 2Fh Function 1609h)

Virtual-Display Device Services and Notifications
Enable VM-Assisted Save/Restore (Interrupt 2Fh Function 4000h)
Notify Background Switch (Interrupt 2Fh Function 4001h)
Notify Foreground Switch (Interrupt 2Fh Function 4002h)
Enter Critical Section (Interrupt 2Fh Function 4003h)
Exit Critical Section (Interrupt 2Fh Function 4004h)
Save Video Register State (Interrupt 2Fh Function 4005h)
Restore Video Register State (Interrupt 2Fh Function 4006h)
Disable VM-Assisted Save/Restore (Interrupt 2Fh Function 4007h)

Int2f,ax=1608h Enhanced Mode Win Broadcast

In vmm.vxd:

```
Int2f_1608_BroadCast proc

var_6C          = byte ptr -6Ch

        pusha
        sub   esp, 6Ch
        push        edi
        lea    edi, [esp+70h+var_6C]
        VMMcall        Save_Client_State
        pop  edi
        VMMcall        Begin_Nest_Exec
        VMMcall        Get_Cur_VM_Handle
        mov ebp, [ebx+VMcb.CB_Client_Pointer]
        mov word ptr [ebp+ClientReg.Client_EAX], 1608h
        mov eax, 2Fh ; '/'  ; int 2f,1608, Enhanced mode Win Broadcast
        VMMcall        Exec_Int
        VMMcall        End_Nest_Exec
        push        esi
        lea    esi, [esp+70h+var_6C]
        VMMcall        Restore_Client_State
        pop  esi
        add  esp, 6Ch
        popa
        clc
        retn
Int2f_1608_BroadCast endp
```

Int 68h

System.Ini

[enh386]

LoadDebugOnlyObjs=1 ; 4 weeks to get it !

Who know this ?

Int 68h

ah=43h,int 68h, SoftICEW hooks this to make it ret F386.
Real part of VMM.vxd check this.
ret F386 means 'Debug Device find'.

AX=
43xx
4401
45xx
48xx
4fxx
5080
5090
5091
5180
52xx

1.AH=43H Get_Version
ret AX=F386H

2.AH=44H Make WINICE Control_Proc Selector
CX=offset in GDT of WINICE Control_Proc selector (48H and 50H)
DS:SI is pointer to GDT
ret ES:EDI=WINICE's Control_Proc Entry (Selector:Offset)
This Control_Proc is in real part of WinICE, not in VxD. Win95 will pass message to this Control_Proc during initialization.

3.report WINDOWS95 is loading the Obj (VXD)
AX=5080H （Data Obj)
AX=5081H (Code Obj)
ES:DI point to Obj Information
Obj Information: (Len 1AH)
+00 DW Obj serial number
+02 DW CS Selector(Code) or DS Selector(Data) after load
+04 DD start address of Obj after load
+08 DD Obj Len
+0C DD VxD Name offset
+10 DW VxD Name Segment
+12 DD +16 DW far pointer to string(for example "VXD32")
+18 DW DS Selector(Code) or CS Selector(Data) after load
WINICE will save Entry of WINICE VXD here.

4.AX=9000H do not know this. WINICE save its EDX.

Control_Proc:(run in protect-mode)
1.AL = 0
DS:EDI = point to IDT
Tell DEBUG DEVICE to change IDT(int 1,2,3,6,B,C,D,E,41).

```
IDT command in WINICE is the IDT before this change.
     2.AL = 1  unknown
     3.AL = 3   unknown
        EBX points to the PDE of MMGR, of second page DEBUG DEVICE reserved.
        EDX points to the virtual address of second page DEBUG DEVICE reserved.
        all these will call WINICE VXD's Entry
     4.AL = 5   unknonn
       ret ESI = 0,AX =0 ;ECX= 0.

IMT 68H:
    1.AH=43H   Get_Version
      ret AX=F386H
    2.AH=44H   Make WINICE Control_Proc Selector
      CX=为WINICE的消息处理过程安排的段描述符在GDT中的偏移。（48H and 50H)
      DS:SI is pointer  to GDT
      ret ES:EDI=WINICE's Control_Proc Entry (Selector:Offset)
      这里的消息处理（Control_Proc)是WINICE的16位消息处理，不是VXD中的。
Win95在启动时向该Control_Proc传递Message.
    3.报告WINDOWS95正装载Obj(VXD的段）
      AX=5080H （Data Obj)
      AX=5081H  (Code Obj)
      ES:DI point to Obj Information
      Obj Information: (Len 1AH)
      +00 DW  Obj 序号
      +02 DW  装载后的CS Selector(Code) or DS Selector(Data)
      +04 DD  Obj在装载后的起始地址
      +08 DD  Obj Len
      +0C DD  VxD Name offset
      +10 DW  VxD Name Segment
      +12 DD +16 DW far pointer to string(for example "VXD32")
      +18 DW  装载后的DS Selector(Code) or CS Selector(Data)
     在这个处理中，WINICE记下WINICE VXD的Entry.
     4.AX=9000H  我不知道这是干什么，WINICE中保存EDX。

Control_Proc:(run in protect-mode)
     1.AL = 0
       DS:EDI = point to IDT
       通知DEBUG DEVICE修改IDT，如1,2,3,6,B,C,D,E,41等中断，在WINICE下，
IDT命令看到的是修改前的。
     2.AL = 1
       不清楚。
     3.AL = 3
       目的不清楚。"EBX指向为DEBUG DEVICE保留的第二个页面的MMGR二次页表项。
EDX指向为DEBUG DEVICE保留的第二个页面的虚拟地址。"(什么意思呢？）
     以上均调用WINICE VXD的Entry
     4.AL = 5
       目的不清楚。ret ESI = 0,AX =0 ;ECX= 0.
     (资料：《Windows 3.1深入剖析--初试化篇》武大 魏晋鹏）
     以上有些是我剖析的，你可以在WIN95的DOS框下用DEBUG跟踪INT 68H来
进一步剖析。另外，DEBUG DEVICE的VXD DDB如果Refence Data不为零，则表一
Entry.Win95 call this as AX=0;ESI pointer to a struct(在VMM的Init期间）
ESI+02 是DEBUG DEVICE的Control_Proc Entry,有可能这是让DEBUG DEVICE验证
Entry 的有效性。
     总的来说，我也不是很清楚。我想只要有Control_Proc就行了，具体怎么
处理并不重要。(对我们来说）
```

Page Memory Manage in Win95

Source: Get aliase address of another context

```
00010053        _PageAllocate
00010054        _PageReAllocate
00010055        _PageFree
00010056        _PageLock
00010057        _PageUnLock
00010058        _PageGetSizeAddr
00010059        _PageGetAllocInfo
0001005A        _GetFreePageCount
0001005B        _GetSysPageCount
0001005C        _GetVMPgCount
0001005D        _MapIntoV86
0001005E        _PhysIntoV86
0001005F        _TestGlobalV86Mem
00010060        _ModifyPageBits
00010061        _CopyPageTable
00010062        _LinMapIntoV86
00010063        _LinPageLock
00010064        _LinPageUnLock
00010065        _SetResetV86Pageable
00010066        _GetV86PageableArray
00010067        _PageCheckLinRange
00010068        _PageOutDirtyPages
00010069        _PageDiscardPages
0001006A        _GetNulPageHandle
0001006B        _GetFirstV86Page
0001006C        _MapPhysToLinear
0001006D        _GetAppFlatDSAlias
0001006E        _SelectorMapFlat
0001006F        _GetDemandPageInfo
00010070        _GetSetPageOutCount
00010071        Hook_V86_Page
00010072        _Assign_Device_V86_Pages
00010073        _DeAssign_Device_V86_Pages
00010074        _Get_Device_V86_Pages_Array
00010075        MMGR_SetNULPageAddr
0001011D        _PageReserve
0001011E        _PageCommit
0001011F        _PageDecommit
00010120        _PagerRegister          ULONG EXTERNAL _PagerRegister(PPD ppd);
00010121        _PagerQuery
00010122        _PagerDeregister

        PD      Page Directory
        PT      Page Table

typedef struct tagOFFS {
    unsigned ofs    : 12; // offset portion of linear addr
    unsigned        : 20;
    } OFFS;

typedef struct tagPAGENUM {
    unsigned        : 12;
```

```
    unsigned number : 20; // page number
    } PAGENUM;

typedef struct tagDIRNUM {
    unsigned        : 22;
    unsigned number : 10; // directory number
    } DIRNUM;

typedef union tagADDR {
    OFFS offset;
    PAGENUM page;
    DIRNUM  dir;
    DWORD   addr;
    } ADDR;

#define BASE_PTE    0xff800000
#define BASE_PDE    0xffbfe000

typedef struct tagPTE {
    unsigned present   : 1;     // bit 0
    unsigned writeable : 1;
    unsigned user      : 1;
    unsigned           : 2;     // - reserved bits
    unsigned accessed  : 1;
    unsigned dirty     : 1;
    unsigned           : 2;     // - reserved bits
    unsigned committed : 1;     // -
    unsigned unknown   : 1;     // - These three bits are in the AVL field
    unsigned physical  : 1;     // -
    unsigned pageframe : 20;
    } PTE, *PPTE;

typedef PTE PDE;
typedef PDE* PPDE;

typedef struct tagPF {
    struct tagVP * pVP;
    DWORD  pagerdata;
    WORD   cLock;
    WORD   cRef;
    BYTE   state;
    } PF, *PPF;

// Bit assignments for the VP 'flags' byte
#define VP_TAINTED  0x80
#define VP_DIRTY    0x40
#define VP_SWAPPED  0x20
#define VP_PHYSICAL 0x10
#define VP_IDLE     0x08
#define VP_W        0x04
#define VP_BUSY     0x02
#define VP_X        0x01


typedef struct tagVP {
    WORD cRef;
    BYTE hPD;
    BYTE flags;
```

```
    WORD iAR;
    union {
        DWORD pagerdata;
        PPF   pPF;
        DWORD sf;
        };
    } VP, *PVP;
```

```
#include <vmm.h>

typedef ULONG _cdecl FUNPAGE(PULONG ppagerdata, PVOID ppage,
    ULONG faultpage);
typedef FUNPAGE *PFUNPAGE;

typedef struct pd_s {
    PFUNPAGE pd_virginin;
    PFUNPAGE pd_taintedin;
    PFUNPAGE pd_cleanout;
    PFUNPAGE pd_dirtyout;
    PFUNPAGE pd_virginfree;
    PFUNPAGE pd_taintedfree;
    PFUNPAGE pd_dirty;
    ULONG pd_type;
} PD, *PPD;
```

```
        each page is 1000h (4K) len
        each Page Table manage 400h (1024d) pages. 4K len
        Page Directory manage  400h (1024d) Page Tables, 4K len, always present
        0-ffff ffff (4G) = 1000h * 400h * 400h = 4k * 1024 * 1024
        32 bit = 10 + 10 + 12

        CR3 & 0xfffff000 = Phys addr of Page Directory
        Linear addr of Page Directory 0xffbfe000
        MapPhysToLinear(CR3 & 0xfffff000) =functional equal to= 0xffbfe000


        Base linear addr of Page Table 0xffbf e000
        Total Page Table need 4K*1024=1000h*400h=40 0000h=4*4G/4K=4M

        LinAddr = ( (IPDE<<10 +IPTE) <<12 )+ AddrInPage
        LinAddr >> 12 = IPDE << 10 +IPTE
        PDE = Phys2Lin(CR3 & 0xfffff000) + IPDE*4
            = 0xffbfe000 + IPDE*4
            = 0xffbfe000 - 0x3fe000 + 0xff800*4 + (LinAddr >>12 >>10)
            = 0xff800000 + ( 0xff800000 + (LinAddr >> 12)*4 ) >> 12 *4
        PTE = Phys2Lin(*PDE & 0xfffff000) + IPTE*4
            = 0xff800000 + (LinAddr >> 12)*4
            = 0xff800000 + IPTE*4 + (IPDE << 10)*4
            = ( 0xff800000 + (IPDE << 10)*4 ) + IPTE*4

        so

        Phys2Lin(*PDE & 0xfffff000) == 0xff800000 + (IPDE << 10)*4
```

```
pIfVMcb         proc stdcall LineAddress:DWORD

                mov     edx, [LineAddress]
                mov     eax, edx
                shr     eax, 0Ch
                lea     ecx, [eax*4+0FF800000h]
                shr     ecx, 0Ch
                test    byte ptr [ecx*4+0FF800000h], 1          ;PDE
                jz      short @@1

                test    byte ptr [eax*4+0FF800000h], 1          ;PTE
                jz      short @@1

                cmp     dword ptr [edx+10h], 'bcMV'
                mov     eax, 1
                jz      short @@2
@@1:
                sub     eax, eax
@@2:
                ret
pIfVMcb         endp
pIfVMcb         proc near               ; CODE XREF: _ModifyPageBits+92p
                                        ; mmMapPhys+73p ...

a_LineAddress   = dword ptr  8

                push    ebp
                mov     ebp, esp
                mov     edx, [ebp+a_LineAddress]
                mov     eax, edx
                shr     eax, 0Ch
                shl     eax, 2
                lea     ecx, [eax+0FF800000h]
                and     ecx, 0FFFFF3FFh

                shr     ecx, 0Ah
                test    byte ptr [ecx+0FF800000h], 1
                jz      short loc_8CCD
                test    byte ptr [eax+0FF800000h], 1
                jz      short loc_8CCD
                cmp     dword ptr [edx+10h], 'bcMV'
                mov     eax, 1
                jz      short loc_8CCF

loc_8CCD:                               ; CODE XREF: pIfVMcb+24j pIfVMcb+2Dj
                sub     eax, eax

loc_8CCF:                               ; CODE XREF: pIfVMcb+3Bj
                pop     ebp
                retn    4
pIfVMcb         endp
```

```
in eax=linear address
 and     eax, 0FFFFF3FFh
 shr     eax, 0Ah
 sub     eax, -0FF800000h       ;add eax,0ff800000h
```

```
or
        shr     eax,12
        shl     eax,2
        add     eax,0ff800000h
```

_PageLock

```
include vmm.inc

VMMcall _PageLock, < hMem, nPages, PageOff, flags >
or      eax, eax        ; nonzero if locked, zero if error
jz      not_locked

Locks one or more pages in the specified memory block.
Uses EAX, ECX, EDX, and Flags.
```

```
<a name=10063><b>_LinPageLock</b>
include vmm.inc
VMMcall _LinPageLock, < page, npages, flags >

#include < vmm.h >
ULONG EXTERNAL _LinPageLock(ULONG page, ULONG npages, ULONG flags);


Locks one or more pages starting at the specified linear page number.
Locking a pages forces it to become physically present and to remain
so until it is unlocked. This service is similar to the _PageLock
service. Uses EAX, ECX, EDX, and Flags.

page :
        Page number of the first page to check. A page number is a ring-0
        linear address shifted right by 12 bits.


        ;eax = linear address
        shr     eax, 12
        VMMcall _LinPageLock, < eax, 1, 0 >
        or      eax, eax        ; nonzero if locked, zero if error
```

```
_MapPhysToLinear
include vmm.inc

VMMcall _MapPhysToLinear, < PhysAddr, nBytes, 0 >

cmp     eax, 0FFFFFFFFh     ; 0FFFFFFFFh if not addressable
je      not_addressable
mov     [Address], eax      ; address of first byte


Returns the ring-0 linear address of the first byte in the specified range of
physical addresses. Uses EAX, ECX, EDX and Flags.

Because physical addresses do not move, the linear address returned by this
```

service remains valid until the system is shut down. Virtual devices should be careful not to use this service in a manner that wastes linear address space.

Device ID		
0001	VMM	VMM
0002	DEBUG	
0003	VPICD	
0004	VDMAD	
0005	VTD	
0006	V86MMGR	
0007	PAGESWAP	
000a	VDD	
000b	VSD	
000c	VMD	
000d	VKD	
000e	VCD	
0010	IOS	
0011	VPCPD	
0012	EBIOS	
0014	VNETBIOS	
0015	DOSMGR	
0017	SHELL	
0018	VMPoll	
001A	DOSNET	
001B	VFD	
001C	VDD2	
0020	Int13	
0021	PageFile	
0026	VPOWERD	
0027	VXDLDR	
002A	VWIN32	
002B	VCOMM	
0033	CONFIGMG	
0036	VFBACKUP	
0037	VMINI	
0038	VCOND	
0040	IFSMgr	
0048	PERF	
0481	VRedir	
048B	VCache	
097C	PCCARD	
011f	??	
0207	??	
2972	??	WINCAM
304c	??	VCAD

See also:

- Win32 VxD Services

FFFFFFFF	VMM
00010000	Get_VMM_Version
00010001	Get_Cur_VM_Handle
00010002	Test_Cur_VM_Handle
00010003	Get_Sys_VM_Handle
00010004	Test_Sys_VM_Handle
00010005	Validate_VM_Handle

```
00010006        Get_VMM_Reenter_Count
00010007        Begin_Reentrant_Execution
00010008        End_Reentrant_Execution
00010009        Install_V86_Break_Point
0001000A        Remove_V86_Break_Point
0001000B        Allocate_V86_Call_Back
0001000C        Allocate_PM_Call_Back
0001000D        Call_When_VM_Returns
0001000E        Schedule_Global_Event
0001000F        Schedule_VM_Event
00010010        Call_Global_Event
00010011        Call_VM_Event
00010012        Cancel_Global_Event
00010013        Cancel_VM_Event
00010014        Call_Priority_VM_Event
00010015        Cancel_Priority_VM_Event
00010016        Get_NMI_Handler_Addr
00010017        Set_NMI_Handler_Addr
00010018        Hook_NMI_Event
00010019        Call_When_VM_Ints_Enabled
0001001A        Enable_VM_Ints
0001001B        Disable_VM_Ints
0001001C        Map_Flat
0001001D        Map_Lin_To_VM_Addr
0001001E        Adjust_Exec_Priority
0001001F        Begin_Critical_Section
00010020        End_Critical_Section
00010021        End_Crit_And_Suspend
00010022        Claim_Critical_Section
00010023        Release_Critical_Section
00010024        Call_When_Not_Critical
00010025        Create_Semaphore
00010026        Destroy_Semaphore
00010027        Wait_Semaphore
00010028        Signal_Semaphore
00010029        Get_Crit_Section_Status
0001002A        Call_When_Task_Switched
0001002B        Suspend_VM
0001002C        Resume_VM
0001002D        No_Fail_Resume_VM
0001002E        Nuke_VM
0001002F        Crash_Cur_VM
00010030        Get_Execution_Focus
00010031        Set_Execution_Focus
00010032        Get_Time_Slice_Priority
00010033        Set_Time_Slice_Priority
00010034        Get_Time_Slice_Granularity
00010035        Set_Time_Slice_Granularity
00010036        Get_Time_Slice_Info
00010037        Adjust_Execution_Time
00010038        Release_Time_Slice
00010039        Wake_Up_VM
0001003A        Call_When_Idle
0001003B        Get_Next_VM_Handle
0001003C        Set_Global_Time_Out
0001003D        Set_VM_Time_Out
0001003E        Cancel_Time_Out
```

```
0001003F        Get_System_Time
00010040        Get_VM_Exec_Time
00010041        Hook_V86_Int_Chain
00010042        Get_V86_Int_Vector
00010043        Set_V86_Int_Vector
00010044        Get_PM_Int_Vector
00010045        Set_PM_Int_Vector
00010046        Simulate_Int
00010047        Simulate_Iret
00010048        Simulate_Far_Call
00010049        Simulate_Far_Jmp
0001004A        Simulate_Far_Ret
0001004B        Simulate_Far_Ret_N
0001004C        Build_Int_Stack_Frame
0001004D        Simulate_Push
0001004E        Simulate_Pop
0001004F        _HeapAllocate
00010050        _HeapReAllocate
00010051        _HeapFree
00010052        _HeapGetSize
00010053        _PageAllocate
00010054        _PageReAllocate
00010055        _PageFree
00010056        _PageLock
00010057        _PageUnLock
00010058        _PageGetSizeAddr
00010059        _PageGetAllocInfo
0001005A        _GetFreePageCount
0001005B        _GetSysPageCount
0001005C        _GetVMPgCount
0001005D        _MapIntoV86
0001005E        _PhysIntoV86
0001005F        _TestGlobalV86Mem
00010060        _ModifyPageBits
00010061        _CopyPageTable
00010062        _LinMapIntoV86
00010063        _LinPageLock
00010064        _LinPageUnLock
00010065        _SetResetV86Pageable
00010066        _GetV86PageableArray
00010067        _PageCheckLinRange
00010068        _PageOutDirtyPages
00010069        _PageDiscardPages
0001006A        _GetNulPageHandle
0001006B        _GetFirstV86Page
0001006C        _MapPhysToLinear
0001006D        _GetAppFlatDSAlias
0001006E        _SelectorMapFlat
0001006F        _GetDemandPageInfo
00010070        _GetSetPageOutCount
00010071        Hook_V86_Page
00010072        _Assign_Device_V86_Pages
00010073        _DeAssign_Device_V86_Pages
00010074        _Get_Device_V86_Pages_Array
00010075        MMGR_SetNULPageAddr
00010076        _Allocate_GDT_Selector
00010077        _Free_GDT_Selector
```

```
00010078        _Allocate_LDT_Selector
00010079        _Free_LDT_Selector
0001007A        _BuildDescriptorDWORDs
0001007B        _GetDescriptor
0001007C        _SetDescriptor
0001007D        _MMGR_Toggle_HMA
0001007E        Get_Fault_Hook_Addrs
0001007F        Hook_V86_Fault
00010080        Hook_PM_Fault
00010081        Hook_VMM_Fault
00010082        Begin_Nest_V86_Exec
00010083        Begin_Nest_Exec
00010084        Exec_Int
00010085        Resume_Exec
00010086        End_Nest_Exec
00010087        Allocate_PM_App_CB_Area
00010088        Get_Cur_PM_App_CB
00010089        Set_V86_Exec_Mode
0001008A        Set_PM_Exec_Mode
0001008B        Begin_Use_Locked_PM_Stack
0001008C        End_Use_Locked_PM_Stack
0001008D        Save_Client_State
0001008E        Restore_Client_State
0001008F        Exec_VxD_Int
00010090        Hook_Device_Service
00010091        Hook_Device_V86_API
00010092        Hook_Device_PM_API
00010093        System_Control
00010094        Simulate_IO
00010095        Install_Mult_IO_Handlers
00010096        Install_IO_Handler
00010097        Enable_Global_Trapping
00010098        Enable_Local_Trapping
00010099        Disable_Global_Trapping
0001009A        Disable_Local_Trapping
0001009B        List_Create
0001009C        List_Destroy
0001009D        List_Allocate
0001009E        List_Attach
0001009F        List_Attach_Tail
000100A0        List_Insert
000100A1        List_Remove
000100A2        List_Deallocate
000100A3        List_Get_First
000100A4        List_Get_Next
000100A5        List_Remove_First
000100A6        _AddInstanceItem
000100A7        _Allocate_Device_CB_Area
000100A8        _Allocate_Global_V86_Data_Area
000100A9        _Allocate_Temp_V86_Data_Area
000100AA        _Free_Temp_V86_Data_Area
000100AB        Get_Profile_Decimal_Int
000100AC        Convert_Decimal_String
000100AD        Get_Profile_Fixed_Point
000100AE        Convert_Fixed_Point_String
000100AF        Get_Profile_Hex_Int
000100B0        Convert_Hex_String
```

```
000100B1        Get_Profile_Boolean
000100B2        Convert_Boolean_String
000100B3        Get_Profile_String
000100B4        Get_Next_Profile_String
000100B5        Get_Environment_String
000100B6        Get_Exec_Path
000100B7        Get_Config_Directory
000100B8        OpenFile
000100B9        Get_PSP_Segment
000100BA        GetDOSVectors
000100BB        Get_Machine_Info
000100BC        GetSet_HMA_Info
000100BD        Set_System_Exit_Code
000100BE        Fatal_Error_Handler
000100BF        Fatal_Memory_Error
000100C0        Update_System_Clock
000100C1        Test_Debug_Installed
000100C2        Out_Debug_String
000100C3        Out_Debug_Chr
000100C4        In_Debug_Chr
000100C5        Debug_Convert_Hex_Binary
000100C6        Debug_Convert_Hex_Decimal
000100C7        Debug_Test_Valid_Handle
000100C8        Validate_Client_Ptr
000100C9        Test_Reenter
000100CA        Queue_Debug_String
000100CB        Log_Proc_Call
000100CC        Debug_Test_Cur_VM
000100CD        Get_PM_Int_Type
000100CE        Set_PM_Int_Type
000100CF        Get_Last_Updated_System_Time
000100D0        Get_Last_Updated_VM_Exec_Time
000100D1        Test_DBCS_Lead_Byte
000100D2        _AddFreePhysPage
000100D3        _PageResetHandlePAddr
000100D4        _SetLastV86Page
000100D5        _GetLastV86Page
000100D6        _MapFreePhysReg
000100D7        _UnmapFreePhysReg
000100D8        _XchgFreePhysReg
000100D9        _SetFreePhysRegCalBk
000100DA        Get_Next_Arena
000100DB        Get_Name_Of_Ugly_TSR
000100DC        Get_Debug_Options
000100DD        Set_Physical_HMA_Alias
000100DE        _GetGlblRng0V86IntBase
000100DF        _Add_Global_V86_Data_Area
000100E0        GetSetDetailedVMError
000100E1        Is_Debug_Chr
000100E2        Clear_Mono_Screen
000100E3        Out_Mono_Chr
000100E4        Out_Mono_String
000100E5        Set_Mono_Cur_Pos
000100E6        Get_Mono_Cur_Pos
000100E7        Get_Mono_Chr
000100E8        Locate_Byte_In_ROM
000100E9        Hook_Invalid_Page_Fault
```

```
000100EA        Unhook_Invalid_Page_Fault
000100EB        Set_Delete_On_Exit_File
000100EC        Close_VM
000100ED        Enable_Touch_1st_Meg
000100EE        Disable_Touch_1st_Meg
000100EF        Install_Exception_Handler
000100F0        Remove_Exception_Handler
000100F1        Get_Crit_Status_No_Block
000100F2        _GetLastUpdatedThreadExecTime
000100F3        _Trace_Out_Service
000100F4        _Debug_Out_Service
000100F5        _Debug_Flags_Service
000100F6        VMMAddImportModuleName
000100F7        VMM_Add_DDB
000100F8        VMM_Remove_DDB
000100F9        Test_VM_Ints_Enabled
000100FA        _BlockOnID
000100FB        Schedule_Thread_Event
000100FC        Cancel_Thread_Event
000100FD        Set_Thread_Time_Out
000100FE        Set_Async_Time_Out
000100FF        _AllocateThreadDataSlot
00010100        _FreeThreadDataSlot
00010101        _CreateMutex
00010102        _DestroyMutex
00010103        _GetMutexOwner
00010104        Call_When_Thread_Switched
00010105        VMMCreateThread
00010106        _GetThreadExecTime
00010107        VMMTerminateThread
00010108        Get_Cur_Thread_Handle
00010109        Test_Cur_Thread_Handle
0001010A        Get_Sys_Thread_Handle
0001010B        Test_Sys_Thread_Handle
0001010C        Validate_Thread_Handle
0001010D        Get_Initial_Thread_Handle
0001010E        Test_Initial_Thread_Handle
0001010F        Debug_Test_Valid_Thread_Handle
00010110        Debug_Test_Cur_Thread
00010111        VMM_GetSystemInitState
00010112        Cancel_Call_When_Thread_Switched
00010113        Get_Next_Thread_Handle
00010114        Adjust_Thread_Exec_Priority
00010115        _Deallocate_Device_CB_Area
00010116        Remove_IO_Handler
00010117        Remove_Mult_IO_Handlers
00010118        Unhook_V86_Int_Chain
00010119        Unhook_V86_Fault
0001011A        Unhook_PM_Fault
0001011B        Unhook_VMM_Fault
0001011C        Unhook_Device_Service
0001011D        _PageReserve
0001011E        _PageCommit
0001011F        _PageDecommit
00010120        _PagerRegister
00010121        _PagerQuery
00010122        _PagerDeregister
```

```
00010123        _ContextCreate
00010124        _ContextDestroy
00010125        _PageAttach
00010126        _PageFlush
00010127        _SignalID
00010128        _PageCommitPhys
00010129        _Register_Win32_Services
0001012A        Cancel_Call_When_Not_Critical
0001012B        Cancel_Call_When_Idle
0001012C        Cancel_Call_When_Task_Switched
0001012D        _Debug_Printf_Service
0001012E        _EnterMutex
0001012F        _LeaveMutex
00010130        Simulate_VM_IO
00010131        Signal_Semaphore_No_Switch
00010132        _ContextSwitch
00010133        _PageModifyPermissions
00010134        _PageQuery
00010135        _EnterMustComplete
00010136        _LeaveMustComplete
00010137        _ResumeExecMustComplete
00010138        _GetThreadTerminationStatus
00010139        _GetInstanceInfo
0001013A        _ExecIntMustComplete
0001013B        _ExecVxDIntMustComplete
0001013C        Begin_V86_Serialization
0001013D        Unhook_V86_Page
0001013E        VMM_GetVxDLocationList
0001013F        VMM_GetDDBList
00010140        Unhook_NMI_Event
00010141        Get_Instanced_V86_Int_Vector
00010142        Get_Set_Real_DOS_PSP
00010143        Call_Priority_Thread_Event
00010144        Get_System_Time_Address
00010145        Get_Crit_Status_Thread
00010146        Get_DDB
00010147        Directed_Sys_Control
00010148        _RegOpenKey
00010149        _RegCloseKey
0001014A        _RegCreateKey
0001014B        _RegDeleteKey
0001014C        _RegEnumKey
0001014D        _RegQueryValue
0001014E        _RegSetValue
0001014F        _RegDeleteValue
00010150        _RegEnumValue
00010151        _RegQueryValueEx
00010152        _RegSetValueEx
00010153        _CallRing3
00010154        Exec_PM_Int
00010155        _RegFlushKey
00010156        _PageCommitContig
00010157        _GetCurrentContext
00010158        _LocalizeSprintf
00010159        _LocalizeStackSprintf
0001015A        Call_Restricted_Event
0001015B        Cancel_Restricted_Event
```

```
0001015C        Register_PEF_Provider
0001015D        _GetPhysPageInfo
0001015E        _RegQueryInfoKey
0001015F        MemArb_Reserve_Pages
00010160        Time_Slice_Sys_VM_Idle
00010161        Time_Slice_Sleep
00010162        Boost_With_Decay
00010163        Set_Inversion_Pri
00010164        Reset_Inversion_Pri
00010165        Release_Inversion_Pri
00010166        Get_Thread_Win32_Pri
00010167        Set_Thread_Win32_Pri
00010168        Set_Thread_Static_Boost
00010169        Set_VM_Static_Boost
0001016A        Release_Inversion_Pri_ID
0001016B        Attach_Thread_To_Group
0001016C        Detach_Thread_From_Group
0001016D        Set_Group_Static_Boost
0001016E        _GetRegistryPath
0001016F        _GetRegistryKey
00010170        Cleanup_Thread_State
00010171        _RegRemapPreDefKey
00010172        End_V86_Serialization
00010173        _Assert_Range
00010174        _Sprintf
00010175        _PageChangePager
00010176        _RegCreateDynKey
00010177        _RegQueryMultipleValues
00010178        Boost_Thread_With_VM
00010179        Get_Boot_Flags
0001017A        Set_Boot_Flags
0001017B        _lstrcpyn
0001017C        _lstrlen
0001017D        _lmemcpy
0001017E        _GetVxDName
0001017F        Force_Mutexes_Free
00010180        Restore_Forced_Mutexes
00010181        _AddReclaimableItem
00010182        _SetReclaimableItem
00010183        _EnumReclaimableItem
00010184        Time_Slice_Wake_Sys_VM
00010185        VMM_Replace_Global_Environment
00010186        Begin_Non_Serial_Nest_V86_Exec
00010187        Get_Nest_Exec_Status
00010188        Open_Boot_Log
00010189        Write_Boot_Log
0001018A        Close_Boot_Log
0001018B        EnableDisable_Boot_Log
0001018C        _Call_On_My_Stack
0001018D        Get_Inst_V86_Int_Vec_Base
0001018E        _lstrcmpi
0001018F        _strupr
00010190        Log_Fault_Call_Out
00010191        _AtEventTime


FFFFFFFF        DEBUG
```

```
00020000        DEBUG_Get_Version
00020001        DEBUG_Fault
00020002        DEBUG_CheckFault
00020003        _DEBUG_LoadSyms


FFFFFFFF        VPICD
00030000        VPICD_Get_Version
00030001        VPICD_Virtualize_IRQ
00030002        VPICD_Set_Int_Request
00030003        VPICD_Clear_Int_Request
00030004        VPICD_Phys_EOI
00030005        VPICD_Get_Complete_Status
00030006        VPICD_Get_Status
00030007        VPICD_Test_Phys_Request
00030008        VPICD_Physically_Mask
00030009        VPICD_Physically_Unmask
0003000A        VPICD_Set_Auto_Masking
0003000B        VPICD_Get_IRQ_Complete_Status
0003000C        VPICD_Convert_Handle_To_IRQ
0003000D        VPICD_Convert_IRQ_To_Int
0003000E        VPICD_Convert_Int_To_IRQ
0003000F        VPICD_Call_When_Hw_Int
00030010        VPICD_Force_Default_Owner
00030011        VPICD_Force_Default_Behavior
00030012        VPICD_Auto_Mask_At_Inst_Swap
00030013        VPICD_Begin_Inst_Page_Swap
00030014        VPICD_End_Inst_Page_Swap
00030015        VPICD_Virtual_EOI
00030016        VPICD_Get_Virtualization_Count
00030017        VPICD_Post_Sys_Critical_Init
00030018        VPICD_VM_SlavePIC_Mask_Change


FFFFFFFF        VDMAD
00040000        VDMAD_Get_Version
00040001        VDMAD_Virtualize_Channel
00040002        VDMAD_Get_Region_Info
00040003        VDMAD_Set_Region_Info
00040004        VDMAD_Get_Virt_State
00040005        VDMAD_Set_Virt_State
00040006        VDMAD_Set_Phys_State
00040007        VDMAD_Mask_Channel
00040008        VDMAD_UnMask_Channel
00040009        VDMAD_Lock_DMA_Region
0004000A        VDMAD_Unlock_DMA_Region
0004000B        VDMAD_Scatter_Lock
0004000C        VDMAD_Scatter_Unlock
0004000D        VDMAD_Reserve_Buffer_Space
0004000E        VDMAD_Request_Buffer
0004000F        VDMAD_Release_Buffer
00040010        VDMAD_Copy_To_Buffer
00040011        VDMAD_Copy_From_Buffer
00040012        VDMAD_Default_Handler
00040013        VDMAD_Disable_Translation
00040014        VDMAD_Enable_Translation
00040015        VDMAD_Get_EISA_Adr_Mode
```

```
00040016        VDMAD_Set_EISA_Adr_Mode
00040017        VDMAD_Unlock_DMA_Region_No_Dirty
00040018        VDMAD_Phys_Mask_Channel
00040019        VDMAD_Phys_Unmask_Channel
0004001A        VDMAD_Unvirtualize_Channel
0004001B        VDMAD_Set_IO_Address
0004001C        VDMAD_Get_Phys_Count
0004001D        VDMAD_Get_Phys_Status
0004001E        VDMAD_Get_Max_Phys_Page
0004001F        VDMAD_Set_Channel_Callbacks
00040020        VDMAD_Get_Virt_Count
00040021        VDMAD_Set_Virt_Count


FFFFFFFF        VTD
00050000        VTD_Get_Version
00050001        VTD_Update_System_Clock
00050002        VTD_Get_Interrupt_Period
00050003        VTD_Begin_Min_Int_Period
00050004        VTD_End_Min_Int_Period
00050005        VTD_Disable_Trapping
00050006        VTD_Enable_Trapping
00050007        VTD_Get_Real_Time
00050008        VTD_Get_Date_And_Time
00050009        VTD_Adjust_VM_Count
0005000A        VTD_Delay


FFFFFFFF        V86MMGR
00060000        V86MMGR_Get_Version
00060001        V86MMGR_Allocate_V86_Pages
00060002        V86MMGR_Set_EMS_XMS_Limits
00060003        V86MMGR_Get_EMS_XMS_Limits
00060004        V86MMGR_Set_Mapping_Info
00060005        V86MMGR_Get_Mapping_Info
00060006        V86MMGR_Xlat_API
00060007        V86MMGR_Load_Client_Ptr
00060008        V86MMGR_Allocate_Buffer
00060009        V86MMGR_Free_Buffer
0006000A        V86MMGR_Get_Xlat_Buff_State
0006000B        V86MMGR_Set_Xlat_Buff_State
0006000C        V86MMGR_Get_VM_Flat_Sel
0006000D        V86MMGR_Map_Pages
0006000E        V86MMGR_Free_Page_Map_Region
0006000F        V86MMGR_LocalGlobalReg
00060010        V86MMGR_GetPgStatus
00060011        V86MMGR_SetLocalA20
00060012        V86MMGR_ResetBasePages
00060013        V86MMGR_SetAvailMapPgs
00060014        V86MMGR_NoUMBInitCalls
00060015        V86MMGR_Get_EMS_XMS_Avail
00060016        V86MMGR_Toggle_HMA
00060017        V86MMGR_Dev_Init
00060018        V86MMGR_Alloc_UM_Page


FFFFFFFF        PageSwap
```

```
00070000        PageSwap_Get_Version
00070001        PageSwap_Invalid_Service1
00070002        PageSwap_Invalid_Service2
00070003        PageSwap_Invalid_Service3
00070004        PageSwap_Invalid_Service4
00070005        PageSwap_Invalid_Service5
00070006        PageSwap_Test_IO_Valid
00070007        PageSwap_Read_Or_Write
00070008        PageSwap_Grow_File
00070009        PageSwap_Init_File


FFFFFFFF        VDD
000A0000        VDD_Get_Version
000A0001        VDD_PIF_State
000A0002        VDD_Get_GrabRtn
000A0003        VDD_Hide_Cursor
000A0004        VDD_Set_VMType
000A0005        VDD_Get_ModTime
000A0006        VDD_Set_HCurTrk
000A0007        VDD_Msg_ClrScrn
000A0008        VDD_Msg_ForColor
000A0009        VDD_Msg_BakColor
000A000A        VDD_Msg_TextOut
000A000B        VDD_Msg_SetCursPos
000A000C        VDD_Query_Access
000A000D        VDD_Check_Update_Soon
000A000E        VDD_Get_Mini_Dispatch_Table
000A000F        VDD_Register_Virtual_Port
000A0010        VDD_Get_VM_Info
000A0011        VDD_Get_Special_VM_IDs
000A0012        VDD_Register_Extra_Screen_Selector
000A0013        VDD_Takeover_VGA_Port
000A0014        VDD_Get_DISPLAYINFO
000A0015        VDD_Do_Physical_IO
000A0016        VDD_Set_Sleep_Flag_Addr


FFFFFFFF        VSD
000B0000        VSD_Get_Version
000B0001        VSD_Bell
000B0002        VSD_SoundOn
000B0003        VSD_TakeSoundPort


FFFFFFFF        VMD
000C0000        VMD_Get_Version
000C0001        VMD_Set_Mouse_Type
000C0002        VMD_Get_Mouse_Owner
000C0003        VMD_Post_Pointer_Message
000C0004        VMD_Set_Cursor_Proc
000C0005        VMD_Call_Cursor_Proc
000C0006        VMD_Set_Mouse_Data
000C0007        VMD_Get_Mouse_Data
000C0008        VMD_Manipulate_Pointer_Message
000C0009        VMD_Set_Middle_Button
```

```
000C000A        VMD_Enable_Disable_Mouse_Events
000C000B        VMD_Post_Absolute_Pointer_Message


FFFFFFFF        VKD
000D0000        VKD_Get_Version
000D0001        VKD_Define_Hot_Key
000D0002        VKD_Remove_Hot_Key
000D0003        VKD_Local_Enable_Hot_Key
000D0004        VKD_Local_Disable_Hot_Key
000D0005        VKD_Reflect_Hot_Key
000D0006        VKD_Cancel_Hot_Key_State
000D0007        VKD_Force_Keys
000D0008        VKD_Get_Kbd_Owner
000D0009        VKD_Define_Paste_Mode
000D000A        VKD_Start_Paste
000D000B        VKD_Cancel_Paste
000D000C        VKD_Get_Msg_Key
000D000D        VKD_Peek_Msg_Key
000D000E        VKD_Flush_Msg_Key_Queue
000D000F        VKD_Enable_Keyboard
000D0010        VKD_Disable_Keyboard
000D0011        VKD_Get_Shift_State
000D0012        VKD_Filter_Keyboard_Input
000D0013        VKD_Put_Byte
000D0014        VKD_Set_Shift_State


FFFFFFFF        VCD
000E0000        VCD_Get_Version
000E0001        VCD_Set_Port_Global
000E0002        VCD_Get_Focus
000E0003        VCD_Virtualize_Port
000E0004        VCD_Acquire_Port
000E0005        VCD_Free_Port
000E0006        VCD_Acquire_Port_Windows_Style
000E0007        VCD_Free_Port_Windows_Style
000E0008        VCD_Steal_Port_Windows_Style
000E0009        VCD_Find_COM_Index
000E000A        VCD_Set_Port_Global_Special
000E000B        VCD_Virtualize_Port_Dynamic
000E000C        VCD_Unvirtualize_Port_Dynamic


FFFFFFFF        IOS
00100000        IOS_Get_Version
00100001        IOS_BD_Register_Device
00100002        IOS_Find_Int13_Drive
00100003        IOS_Get_Device_List
00100004        IOS_SendCommand
00100005        IOS_BD_Command_Complete
00100006        IOS_Synchronous_Command
00100007        IOS_Register
00100008        IOS_Requestor_Service
00100009        IOS_Exclusive_Access
0010000A        IOS_Send_Next_Command
0010000B        IOS_Set_Async_Time_Out
```

```
0010000C        IOS_Signal_Semaphore_No_Switch
0010000D        IOSIdleStatus
0010000E        IOSMapIORSToI24
0010000F        IOSMapIORSToI21
00100010        PrintLog


FFFFFFFF        VMCPD
00110000        VMCPD_Get_Version
00110001        VMCPD_Get_Virt_State
00110002        VMCPD_Set_Virt_State
00110003        VMCPD_Get_CR0_State
00110004        VMCPD_Set_CR0_State
00110005        VMCPD_Get_Thread_State
00110006        VMCPD_Set_Thread_State
00110007        _VMCPD_Get_FP_Instruction_Size
00110008        VMCPD_Set_Thread_Precision


FFFFFFFF        EBIOS
00120000        EBIOS_Get_Version
00120001        EBIOS_Get_Unused_Mem


FFFFFFFF        VNETBIOS
00140000        VNETBIOS_Get_Version
00140001        VNETBIOS_Register
00140002        VNETBIOS_Submit
00140003        VNETBIOS_Enum
00140004        VNETBIOS_Deregister
00140005        VNETBIOS_Register2
00140006        VNETBIOS_Map
00140007        VNETBIOS_Enum2


FFFFFFFF        DOSMGR
00150000        DOSMGR_Get_Version
00150001        _DOSMGR_Set_Exec_VM_Data
00150002        DOSMGR_Copy_VM_Drive_State
00150003        _DOSMGR_Exec_VM
00150004        DOSMGR_Get_IndosPtr
00150005        DOSMGR_Add_Device
00150006        DOSMGR_Remove_Device
00150007        DOSMGR_Instance_Device
00150008        DOSMGR_Get_DOS_Crit_Status
00150009        DOSMGR_Enable_Indos_Polling
0015000A        DOSMGR_BackFill_Allowed
0015000B        DOSMGR_LocalGlobalReg
0015000C        DOSMGR_Init_UMB_Area
0015000D        DOSMGR_Begin_V86_App
0015000E        DOSMGR_End_V86_App
0015000F        DOSMGR_Alloc_Local_Sys_VM_Mem
00150010        DOSMGR_Grow_CDSs
00150011        DOSMGR_Translate_Server_DOS_Call
00150012        DOSMGR_MMGR_PSP_Change_Notifier
```

```
FFFFFFFF        SHELL
00170000        SHELL_Get_Version
00170001        SHELL_Resolve_Contention
00170002        SHELL_Event
00170003        SHELL_SYSMODAL_Message
00170004        SHELL_Message
00170005        SHELL_GetVMInfo
00170006        _SHELL_PostMessage
00170007        _SHELL_ShellExecute
00170008        _SHELL_PostShellMessage
00170009        SHELL_DispatchRing0AppyEvents
0017000A        SHELL_Hook_Properties
0017000B        SHELL_Unhook_Properties
0017000C        SHELL_Update_User_Activity
0017000D        _SHELL_QueryAppyTimeAvailable
0017000E        _SHELL_CallAtAppyTime
0017000F        _SHELL_CancelAppyTimeEvent
00170010        _SHELL_BroadcastSystemMessage
00170011        _SHELL_HookSystemBroadcast
00170012        _SHELL_UnhookSystemBroadcast
00170013        _SHELL_LocalAllocEx
00170014        _SHELL_LocalFree
00170015        _SHELL_LoadLibrary
00170016        _SHELL_FreeLibrary
00170017        _SHELL_GetProcAddress
00170018        _SHELL_CallDll
00170019        _SHELL_SuggestSingleMSDOSMode
0017001A        SHELL_CheckHotkeyAllowed
0017001B        _SHELL_GetDOSAppInfo


FFFFFFFF        VMPoll
00180000        VMPoll_Get_Version
00180001        VMPoll_Enable_Disable
00180002        VMPoll_Reset_Detection
00180003        VMPoll_Check_Idle


FFFFFFFF        DOSNET
001A0000        DOSNET_Get_Version
001A0001        DOSNET_Send_FILESYSCHANGE
001A0002        DOSNET_Do_PSP_Adjust


FFFFFFFF        VFD
001B0000        VFD_Get_Version


FFFFFFFF        VDD2
001C0000        VDD2_Get_Version


FFFFFFFF        Int13
00200000        Int13_Get_Version
00200001        Int13_Device_Registered
00200002        Int13_Translate_VM_Int
00200003        Int13_Hooking_BIOS_Int
```

```
00200004        Int13_Unhooking_BIOS_Int


FFFFFFFF        PageFile
00210000        PageFile_Get_Version
00210001        PageFile_Init_File
00210002        PageFile_Clean_Up
00210003        PageFile_Grow_File
00210004        PageFile_Read_Or_Write
00210005        PageFile_Cancel
00210006        PageFile_Test_IO_Valid
00210007        PageFile_Get_Size_Info
00210008        PageFile_Set_Async_Manager
00210009        PageFile_Call_Async_Manager


FFFFFFFF        VPOWERD
00260000        _VPOWERD_Get_Version
00260001        _VPOWERD_Get_APM_BIOS_Version
00260002        _VPOWERD_Get_Power_Management_Level
00260003        _VPOWERD_Set_Power_Management_Level
00260004        _VPOWERD_Set_Device_Power_State
00260005        _VPOWERD_Set_System_Power_State
00260006        _VPOWERD_Restore_Power_On_Defaults
00260007        _VPOWERD_Get_Power_Status
00260008        _VPOWERD_Get_Power_State
00260009        _VPOWERD_OEM_APM_Function
0026000A        _VPOWERD_Register_Power_Handler
0026000B        _VPOWERD_Deregister_Power_Handler
0026000C        _VPOWERD_W32_Get_System_Power_Status
0026000D        _VPOWERD_W32_Set_System_Power_State


FFFFFFFF        VXDLDR
00270000        VXDLDR_GetVersion
00270001        VXDLDR_LoadDevice
00270002        VXDLDR_UnloadDevice
00270003        VXDLDR_DevInitSucceeded
00270004        VXDLDR_DevInitFailed
00270005        VXDLDR_GetDeviceList
00270006        VXDLDR_UnloadMe
00270007        _PELDR_LoadModule
00270008        _PELDR_GetModuleHandle
00270009        _PELDR_GetModuleUsage
0027000A        _PELDR_GetEntryPoint
0027000B        _PELDR_GetProcAddress
0027000C        _PELDR_AddExportTable
0027000D        _PELDR_RemoveExportTable
0027000E        _PELDR_FreeModule
0027000F        VXDLDR_Notify
00270010        _PELDR_InitCompleted
00270011        _PELDR_LoadModuleEx


FFFFFFFF        VWIN32
002A0000        VWIN32_Get_Version
002A0001        VWIN32_DIOCCompletionRoutine
```

```
002A0002        _VWIN32_QueueUserApc
002A0003        _VWIN32_Get_Thread_Context
002A0004        _VWIN32_Set_Thread_Context
002A0005        _VWIN32_CopyMem
002A0006        _VWIN32_Npx_Exception
002A0007        _VWIN32_Emulate_Npx
002A0008        _VWIN32_CheckDelayedNpxTrap
002A0009        VWIN32_EnterCrstR0
002A000A        VWIN32_LeaveCrstR0
002A000B        _VWIN32_FaultPopup
002A000C        VWIN32_GetContextHandle
002A000D        VWIN32_GetCurrentProcessHandle go
002A000E        _VWIN32_SetWin32Event
002A000F        _VWIN32_PulseWin32Event
002A0010        _VWIN32_ResetWin32Event                 ????
002A0011        _VWIN32_WaitSingleObject
002A0012        _VWIN32_WaitMultipleObjects
002A0013        _VWIN32_CreateRing0Thread
002A0014        _VWIN32_CloseVxDHandle
002A0015        VWIN32_ActiveTimeBiasSet
002A0016        VWIN32_GetCurrentDirectory      go
002A0017        VWIN32_BlueScreenPopup
002A0018        VWIN32_TerminateApp
002A0019        _VWIN32_QueueKernelAPC
002A001A        VWIN32_SysErrorBox
002A001B        _VWIN32_IsClientWin32
002A001C        VWIN32_IFSRIPWhenLev2Taken


FFFFFFFF        VCOMM
002B0000        VCOMM_Get_Version
002B0001        _VCOMM_Register_Port_Driver
002B0002        _VCOMM_Acquire_Port
002B0003        _VCOMM_Release_Port
002B0004        _VCOMM_OpenComm
002B0005        _VCOMM_SetCommState
002B0006        _VCOMM_GetCommState
002B0007        _VCOMM_SetupComm
002B0008        _VCOMM_TransmitCommChar
002B0009        _VCOMM_CloseComm
002B000A        _VCOMM_GetCommQueueStatus
002B000B        _VCOMM_ClearCommError
002B000C        _VCOMM_GetModemStatus
002B000D        _VCOMM_GetCommProperties
002B000E        _VCOMM_EscapeCommFunction
002B000F        _VCOMM_PurgeComm
002B0010        _VCOMM_SetCommEventMask
002B0011        _VCOMM_GetCommEventMask
002B0012        _VCOMM_WriteComm
002B0013        _VCOMM_ReadComm
002B0014        _VCOMM_EnableCommNotification
002B0015        _VCOMM_GetLastError
002B0016        _VCOMM_Steal_Port
002B0017        _VCOMM_SetReadCallBack
002B0018        _VCOMM_SetWriteCallBack
002B0019        _VCOMM_Add_Port
```

```
002B001A        _VCOMM_GetSetCommTimeouts
002B001B        _VCOMM_SetWriteRequest
002B001C        _VCOMM_SetReadRequest
002B001D        _VCOMM_Dequeue_Request
002B001E        _VCOMM_Enumerate_DevNodes
002B001F        VCOMM_Map_Win32DCB_To_Ring0
002B0020        VCOMM_Map_Ring0DCB_To_Win32
002B0021        _VCOMM_Get_Contention_Handler
002B0022        _VCOMM_Map_Name_To_Resource


FFFFFFFF        CONFIGMG
00330000        _CONFIGMG_Get_Version
00330001        _CONFIGMG_Initialize
00330002        _CONFIGMG_Locate_DevNode
00330003        _CONFIGMG_Get_Parent
00330004        _CONFIGMG_Get_Child
00330005        _CONFIGMG_Get_Sibling
00330006        _CONFIGMG_Get_Device_ID_Size
00330007        _CONFIGMG_Get_Device_ID
00330008        _CONFIGMG_Get_Depth
00330009        _CONFIGMG_Get_Private_DWord
0033000A        _CONFIGMG_Set_Private_DWord
0033000B        _CONFIGMG_Create_DevNode
0033000C        _CONFIGMG_Query_Remove_SubTree
0033000D        _CONFIGMG_Remove_SubTree
0033000E        _CONFIGMG_Register_Device_Driver
0033000F        _CONFIGMG_Register_Enumerator
00330010        _CONFIGMG_Register_Arbitrator
00330011        _CONFIGMG_Deregister_Arbitrator
00330012        _CONFIGMG_Query_Arbitrator_Free_Size
00330013        _CONFIGMG_Query_Arbitrator_Free_Data
00330014        _CONFIGMG_Sort_NodeList
00330015        _CONFIGMG_Yield
00330016        _CONFIGMG_Lock
00330017        _CONFIGMG_Unlock
00330018        _CONFIGMG_Add_Empty_Log_Conf
00330019        _CONFIGMG_Free_Log_Conf
0033001A        _CONFIGMG_Get_First_Log_Conf
0033001B        _CONFIGMG_Get_Next_Log_Conf
0033001C        _CONFIGMG_Add_Res_Des
0033001D        _CONFIGMG_Modify_Res_Des
0033001E        _CONFIGMG_Free_Res_Des
0033001F        _CONFIGMG_Get_Next_Res_Des
00330020        _CONFIGMG_Get_Performance_Info
00330021        _CONFIGMG_Get_Res_Des_Data_Size
00330022        _CONFIGMG_Get_Res_Des_Data
00330023        _CONFIGMG_Process_Events_Now
00330024        _CONFIGMG_Create_Range_List
00330025        _CONFIGMG_Add_Range
00330026        _CONFIGMG_Delete_Range
00330027        _CONFIGMG_Test_Range_Available
00330028        _CONFIGMG_Dup_Range_List
00330029        _CONFIGMG_Free_Range_List
0033002A        _CONFIGMG_Invert_Range_List
0033002B        _CONFIGMG_Intersect_Range_List
0033002C        _CONFIGMG_First_Range
```

```
0033002D        _CONFIGMG_Next_Range
0033002E        _CONFIGMG_Dump_Range_List
0033002F        _CONFIGMG_Load_DLVxDs
00330030        _CONFIGMG_Get_DDBs
00330031        _CONFIGMG_Get_CRC_CheckSum
00330032        _CONFIGMG_Register_DevLoader
00330033        _CONFIGMG_Reenumerate_DevNode
00330034        _CONFIGMG_Setup_DevNode
00330035        _CONFIGMG_Reset_Children_Marks
00330036        _CONFIGMG_Get_DevNode_Status
00330037        _CONFIGMG_Remove_Unmarked_Children
00330038        _CONFIGMG_ISAPNP_To_CM
00330039        _CONFIGMG_CallBack_Device_Driver
0033003A        _CONFIGMG_CallBack_Enumerator
0033003B        _CONFIGMG_Get_Alloc_Log_Conf
0033003C        _CONFIGMG_Get_DevNode_Key_Size
0033003D        _CONFIGMG_Get_DevNode_Key
0033003E        _CONFIGMG_Read_Registry_Value
0033003F        _CONFIGMG_Write_Registry_Value
00330040        _CONFIGMG_Disable_DevNode
00330041        _CONFIGMG_Enable_DevNode
00330042        _CONFIGMG_Move_DevNode
00330043        _CONFIGMG_Set_Bus_Info
00330044        _CONFIGMG_Get_Bus_Info
00330045        _CONFIGMG_Set_HW_Prof
00330046        _CONFIGMG_Recompute_HW_Prof
00330047        _CONFIGMG_Query_Change_HW_Prof
00330048        _CONFIGMG_Get_Device_Driver_Private_DWord
00330049        _CONFIGMG_Set_Device_Driver_Private_DWord
0033004A        _CONFIGMG_Get_HW_Prof_Flags
0033004B        _CONFIGMG_Set_HW_Prof_Flags
0033004C        _CONFIGMG_Read_Registry_Log_Confs
0033004D        _CONFIGMG_Run_Detection
0033004E        _CONFIGMG_Call_At_Appy_Time
0033004F        _CONFIGMG_Fail_Change_HW_Prof
00330050        _CONFIGMG_Set_Private_Problem
00330051        _CONFIGMG_Debug_DevNode
00330052        _CONFIGMG_Get_Hardware_Profile_Info
00330053        _CONFIGMG_Register_Enumerator_Function
00330054        _CONFIGMG_Call_Enumerator_Function
00330055        _CONFIGMG_Add_ID
00330056        _CONFIGMG_Find_Range
00330057        _CONFIGMG_Get_Global_State
00330058        _CONFIGMG_Broadcast_Device_Change_Message
00330059        _CONFIGMG_Call_DevNode_Handler
0033005A        _CONFIGMG_Remove_Reinsert_All


FFFFFFFF        VFBACKUP
00360000        VFBACKUP_Get_Version
00360001        VFBACKUP_Lock_NEC
00360002        VFBACKUP_UnLock_NEC
00360003        VFBACKUP_Register_NEC
00360004        VFBACKUP_Register_VFD
00360005        VFBACKUP_Lock_All_Ports
```

```
FFFFFFFF        VMINI
00370000        VMINI_GetVersion
00370001        VMINI_Update
00370002        VMINI_Status
00370003        VMINI_DisplayError
00370004        VMINI_SetTimeStamp
00370005        VMINI_Siren
00370006        VMINI_RegisterAccess
00370007        VMINI_GetData
00370008        VMINI_ShutDownItem
00370009        VMINI_RegisterSK


FFFFFFFF        VCOND
00380000        VCOND_Get_Version
00380001        VCOND_Launch_ConApp_Inherited


FFFFFFFF        IFSMgr
00400000        IFSMgr_Get_Version
00400001        IFSMgr_RegisterMount
00400002        IFSMgr_RegisterNet
00400003        IFSMgr_RegisterMailSlot
00400004        IFSMgr_Attach
00400005        IFSMgr_Detach
00400006        IFSMgr_Get_NetTime
00400007        IFSMgr_Get_DOSTime
00400008        IFSMgr_SetupConnection
00400009        IFSMgr_DerefConnection
0040000A        IFSMgr_ServerDOSCall
0040000B        IFSMgr_CompleteAsync
0040000C        IFSMgr_RegisterHeap
0040000D        IFSMgr_GetHeap
0040000E        IFSMgr_RetHeap
0040000F        IFSMgr_CheckHeap
00400010        IFSMgr_CheckHeapItem
00400011        IFSMgr_FillHeapSpare
00400012        IFSMgr_Block
00400013        IFSMgr_Wakeup
00400014        IFSMgr_Yield
00400015        IFSMgr_SchedEvent
00400016        IFSMgr_QueueEvent
00400017        IFSMgr_KillEvent
00400018        IFSMgr_FreeIOReq
00400019        IFSMgr_MakeMailSlot
0040001A        IFSMgr_DeleteMailSlot
0040001B        IFSMgr_WriteMailSlot
0040001C        IFSMgr_PopUp
0040001D        IFSMgr_printf
0040001E        IFSMgr_AssertFailed
0040001F        IFSMgr_LogEntry
00400020        IFSMgr_DebugMenu
00400021        IFSMgr_DebugVars
00400022        IFSMgr_GetDebugString
00400023        IFSMgr_GetDebugHexNum
00400024        IFSMgr_NetFunction
00400025        IFSMgr_DoDelAllUses
```

```
00400026        IFSMgr_SetErrString
00400027        IFSMgr_GetErrString
00400028        IFSMgr_SetReqHook
00400029        IFSMgr_SetPathHook
0040002A        IFSMgr_UseAdd
0040002B        IFSMgr_UseDel
0040002C        IFSMgr_InitUseAdd
0040002D        IFSMgr_ChangeDir
0040002E        IFSMgr_DelAllUses
0040002F        IFSMgr_CDROM_Attach
00400030        IFSMgr_CDROM_Detach
00400031        IFSMgr_Win32DupHandle
00400032        IFSMgr_Ring0_FileIO
00400033        IFSMgr_Win32_Get_Ring0_Handle
00400034        IFSMgr_Get_Drive_Info
00400035        IFSMgr_Ring0GetDriveInfo
00400036        IFSMgr_BlockNoEvents
00400037        IFSMgr_NetToDosTime
00400038        IFSMgr_DosToNetTime
00400039        IFSMgr_DosToWin32Time
0040003A        IFSMgr_Win32ToDosTime
0040003B        IFSMgr_NetToWin32Time
0040003C        IFSMgr_Win32ToNetTime
0040003D        IFSMgr_MetaMatch
0040003E        IFSMgr_TransMatch
0040003F        IFSMgr_CallProvider
00400040        UniToBCS
00400041        UniToBCSPath
00400042        BCSToUni
00400043        UniToUpper
00400044        UniCharToOEM
00400045        CreateBasis
00400046        MatchBasisName
00400047        AppendBasisTail
00400048        FcbToShort
00400049        ShortToFcb
0040004A        IFSMgr_ParsePath
0040004B        Query_PhysLock
0040004C        _VolFlush
0040004D        NotifyVolumeArrival
0040004E        NotifyVolumeRemoval
0040004F        QueryVolumeRemoval
00400050        IFSMgr_FSDUnmountCFSD
00400051        IFSMgr_GetConversionTablePtrs
00400052        IFSMgr_CheckAccessConflict
00400053        IFSMgr_LockFile
00400054        IFSMgr_UnlockFile
00400055        IFSMgr_RemoveLocks
00400056        IFSMgr_CheckLocks
00400057        IFSMgr_CountLocks
00400058        IFSMgr_ReassignLockFileInst
00400059        IFSMgr_UnassignLockList
0040005A        IFSMgr_MountChildVolume
0040005B        IFSMgr_UnmountChildVolume
0040005C        IFSMgr_SwapDrives
0040005D        IFSMgr_FSDMapFHtoIOREQ
0040005E        IFSMgr_FSDParsePath
```

```
0040005F        IFSMgr_FSDAttachSFT
00400060        IFSMgr_GetTimeZoneBias
00400061        IFSMgr_PNPEvent
00400062        IFSMgr_RegisterCFSD
00400063        IFSMgr_Win32MapExtendedHandleToSFT
00400064        IFSMgr_DbgSetFileHandleLimit
00400065        IFSMgr_Win32MapSFTToExtendedHandle
00400066        IFSMgr_FSDGetCurrentDrive
00400067        IFSMgr_InstallFileSystemApiHook
00400068        IFSMgr_RemoveFileSystemApiHook
00400069        IFSMgr_RunScheduledEvents
0040006A        IFSMgr_CheckDelResource
0040006B        IFSMgr_Win32GetVMCurdir
0040006C        IFSMgr_SetupFailedConnection
0040006D        _GetMappedErr
0040006E        ShortToLossyFcb
0040006F        IFSMgr_GetLockState
00400070        BcsToBcs
00400071        IFSMgr_SetLoopback
00400072        IFSMgr_ClearLoopback
00400073        IFSMgr_ParseOneElement
00400074        BcsToBcsUpper


FFFFFFFF        PERF
00480000        PERF_Get_Version
00480001        PERF_Server_Register
00480002        PERF_Server_Deregister
00480003        PERF_Server_Add_Stat
00480004        PERF_Server_Remove_Stat


FFFFFFFF        VRedir
04810000        VRedir_Get_Version
04810001        VRedir_Register
04810002        VRedir_MakeMailSlot
04810003        VRedir_DeleteMailSlot
04810004        VRedir_ServerEnum


FFFFFFFF        VCache
048B0000        VCache_Get_Version
048B0001        VCache_Register
048B0002        VCache_GetSize
048B0003        VCache_CheckAvail
048B0004        VCache_FindBlock
048B0005        VCache_FreeBlock
048B0006        VCache_MakeMRU
048B0007        VCache_Hold
048B0008        VCache_Unhold
048B0009        VCache_Enum
048B000A        VCache_TestHandle
048B000B        VCache_VerifySums
048B000C        VCache_RecalcSums
048B000D        VCache_TestHold
048B000E        VCache_GetStats
048B000F        VCache_Deregister
```

```
048B0010        VCache_AdjustMinimum
048B0011        VCache_SwapBuffers
048B0012        VCache_RelinquishPage
048B0013        VCache_UseThisPage
048B0014        _VCache_CreateLookupCache
048B0015        _VCache_CloseLookupCache
048B0016        _VCache_DeleteLookupCache
048B0017        _VCache_Lookup
048B0018        _VCache_UpdateLookup


FFFFFFFF        PCCARD
097C0000        PCCARD_Get_Version
097C0001        PCCARD_Card_Services



        UNDEFINED       0x0000
        VMM             0x0001 /* Used for dynalink table */
        DEBUG           0x0002
        VPICD           0x0003
        VDMAD           0x0004
        VTD             0x0005
        V86MMGR         0x0006
        PAGESWAP        0x0007
        PARITY          0x0008
        REBOOT          0x0009
        VDD             0x000A
        VSD             0x000B
        VMD             0x000C
        VKD             0x000D
        VCD             0x000E
        VPD             0x000F
        BLOCKDEV        0x0010
        VMCPD           0x0011
        EBIOS           0x0012
        BIOSXLAT        0x0013
        VNETBIOS        0x0014
        DOSMGR          0x0015
        WINLOAD         0x0016
        SHELL           0x0017
        VMPOLL          0x0018
        VPROD           0x0019
        DOSNET          0x001A
        VFD             0x001B
        VDD2            0x001C /* Secondary display adapter */
        WINDEBUG        0x001D
        TSRLOAD         0x001E /* TSR instance utility ID */
        BIOSHOOK        0x001F /* Bios interrupt hooker VxD */
        INT13           0x0020
        PAGEFILE        0x0021 /* Paging File device */
        SCSI            0x0022 /* SCSI device */
        MCA_POS         0x0023 /* MCA_POS device */
        SCSIFD          0x0024 /* SCSI FastDisk device */
        VPEND           0x0025 /* Pen device */
        APM             0x0026 /* Power Management device */
        VPOWERD         0x0026 /* We overload APM since we replace it */
```

```
VXDLDR          0x0027 /* VxD Loader device */
NDIS            0x0028 /* NDIS wrapper */
BIOS_EXT        0x0029 /* Fix Broken BIOS device */
VWIN32          0x002A /* for new WIN32-VxD */
VCOMM           0x002B /* New COMM device driver */
SPOOLER         0x002C /* Local Spooler */
WIN32S          0x002D /* Win32S on Win 3.1 driver */
DEBUGCMD        0x002E /* Debug command extensions */
RESERVED        0x002F /* Not currently in use */
ATI_HELPER      0x0030 /* grabbed by ATI */
VNB             0x0031 /* Netbeui of snowball */
SERVER          0x0032 /* Server of snowball */
CONFIGMG        0x0033 /* Configuration manager (Plug&Play) */
DWCFGMG         0x0034 /* Configuration manager for win31 and DOS */
SCSIPORT        0x0035 /* Dragon miniport loader/driver */
VFBACKUP        0x0036 /* allows backup apps to work with NEC */
ENABLE          0x0037 /* for access VxD */
VCOND           0x0038 /* Virtual Console Device - check vcond.inc */
EFAX            0x003A /* EFAX VxD ID       */
DSVXD           0x003B /* Dbl Space VxD ID */
ISAPNP          0x003C /* ISA P&P Enumerator */
BIOS            0x003D /* BIOS P&P Enumerator */
WINSOCK         0x003E /* WinSockets */
WSIPX           0x003F /* WinSockets for IPX */
IFSMgr          0x0040 /* Installable File System Manager */
VCDFSD          0x0041 /* Static CDFS ID */
MRCI2           0x0042 /* DrvSpace compression engine */
PCI             0x0043 /* PCI P&P Enumerator */
PELOADER        0x0044 /* PE Image Loader */
EISA            0x0045 /* EISA P&P Enumerator */
DRAGCLI         0x0046 /* Dragon network client */
DRAGSRV         0x0047 /* Dragon network server */
PERF            0x0048 /* Config/stat info */
AWREDIR         0x0049 /* AtWork Network FSD */
ETEN            0x0060 /* ETEN DOS (Taiwan) driver */
CHBIOS          0x0061 /* CHBIOS DOS (Korean) driver */
VMSGD           0x0062 /* DBCS Message Mode driver */
VPPID           0x0063 /* PC-98 System Control PPI */
VIME            0x0064 /* Virtual DOS IME */
VHBIOSD         0x0065 /* HBIOS (Korean) for HWin31 driver */
```

System Control Messages

```
include vmm.inc



SYS_CRITICAL_INIT               0000H
DEVICE_INIT                     0001H
INIT_COMPLETE                   0002H
SYS_VM_INIT                     0003H
SYS_VM_TERMINATE                0004H
SYSTEM_EXIT                     0005H
SYS_CRITICAL_EXIT               0006H
CREATE_VM                       0007H
VM_CRITICAL_INIT                0008H
VM_INIT EQU                     0009H
VM_TERMINATE                    000AH
VM_NOT_EXECUTEABLE              000BH
DESTROY_VM                      000CH
VM_SUSPEND                      000DH
VM_RESUME                       000EH
SET_DEVICE_FOCUS                000FH
BEGIN_MESSAGE_MODE              0010H
END_MESSAGE_MODE                0011H
REBOOT_PROCESSOR                0012H
QUERY_DESTROY                   0013H
DEBUG_QUERY                     0014H   see: dot command of debug
BEGIN_PM_APP                    0015H
END_PM_APP                      0016H
DEVICE_REBOOT_NOTIFY            0017H
CRIT_REBOOT_NOTIFY              0018H
CLOSE_VM_NOTIFY                 0019H
POWER_EVENT                     001AH
SYS_DYNAMIC_DEVICE_INIT         001BH
SYS_DYNAMIC_DEVICE_EXIT         001CH
CREATE_THREAD                   001DH
THREAD_INIT                     001EH
TERMINATE_THREAD                001FH
THREAD_Not_Executeable          0020H
DESTROY_THREAD                  0021H
PNP_NEW_DEVNODE                 0022H
W32_DEVICEIOCONTROL             0023H
Sys_VM_Terminate2               0024H
System_Exit2                    0025H
Sys_Critical_Exit2              0026H
```

```
VM_Terminate2                   0027H
VM_Not_Executable2              0028H
Destroy_VM2                     0029H
VM_Suspend2                     002AH
End_Message_Mode2               002BH
End_PM_App2                     002CH
Device_Reboot_Notify2           002DH
Crit_Reboot_Notify2             002EH
Close_VM_Notify2                002FH

KERNEL32_INITIALIZED            0031H
KERNEL32_SHUTDOWN               0032H


SYS_CRITICAL_INIT       EQU     0000H
DEVICE_INIT     EQU     0001H
INIT_COMPLETE   EQU     0002H
SYS_VM_INIT     EQU     0003H
SYS_VM_TERMINATE        EQU     0004H
SYSTEM_EXIT     EQU     0005H
SYS_CRITICAL_EXIT       EQU     0006H
CREATE_VM       EQU     0007H
VM_CRITICAL_INIT        EQU     0008H
VM_INIT EQU     0009H
VM_TERMINATE    EQU     000AH
VM_NOT_EXECUTEABLE      EQU     000BH
DESTROY_VM      EQU     000CH
VNE_CRASHED_BIT EQU     00H
VNE_CRASHED     EQU     (1 SHL VNE_CRASHED_BIT)
VNE_NUKED_BIT   EQU     01H
VNE_NUKED       EQU     (1 SHL VNE_NUKED_BIT)
VNE_CREATEFAIL_BIT      EQU     02H
VNE_CREATEFAIL  EQU     (1 SHL VNE_CREATEFAIL_BIT)
VNE_CRINITFAIL_BIT      EQU     03H
VNE_CRINITFAIL  EQU     (1 SHL VNE_CRINITFAIL_BIT)
VNE_INITFAIL_BIT        EQU     04H
VNE_INITFAIL    EQU     (1 SHL VNE_INITFAIL_BIT)
VNE_CLOSED_BIT  EQU     05H
VNE_CLOSED      EQU     (1 SHL VNE_CLOSED_BIT)
VM_SUSPEND      EQU     000DH
VM_RESUME       EQU     000EH
SET_DEVICE_FOCUS        EQU     000FH
BEGIN_MESSAGE_MODE      EQU     0010H
END_MESSAGE_MODE        EQU     0011H
REBOOT_PROCESSOR        EQU     0012H
QUERY_DESTROY   EQU     0013H
DEBUG_QUERY     EQU     0014H
BEGIN_PM_APP    EQU     0015H
```

```
BPA_32_BIT      EQU     01H
BPA_32_BIT_FLAG EQU     1
END_PM_APP      EQU     0016H
DEVICE_REBOOT_NOTIFY    EQU     0017H
CRIT_REBOOT_NOTIFY      EQU     0018H
CLOSE_VM_NOTIFY EQU     0019H
CVNF_CRIT_CLOSE_BIT     EQU     0
CVNF_CRIT_CLOSE EQU     (1 SHL CVNF_CRIT_CLOSE_BIT)
POWER_EVENT     EQU     001AH
SYS_DYNAMIC_DEVICE_INIT EQU     001BH
SYS_DYNAMIC_DEVICE_EXIT EQU     001CH
CREATE_THREAD   EQU     001DH
THREAD_INIT     EQU     001EH
TERMINATE_THREAD        EQU     001FH
THREAD_Not_Executeable  EQU     0020H
DESTROY_THREAD  EQU     0021H
PNP_NEW_DEVNODE EQU     0022H
W32_DEVICEIOCONTROL     EQU     0023H
DIOC_GETVERSION EQU     0H
DIOC_OPEN       EQU     DIOC_GETVERSION
DIOC_CLOSEHANDLE        EQU     -1
SYS_VM_TERMINATE2       EQU     0024H
SYSTEM_EXIT2    EQU     0025H
SYS_CRITICAL_EXIT2      EQU     0026H
VM_TERMINATE2   EQU     0027H
VM_NOT_EXECUTEABLE2     EQU     0028H
DESTROY_VM2     EQU     0029H
VM_SUSPEND2     EQU     002AH
END_MESSAGE_MODE2       EQU     002BH
END_PM_APP2     EQU     002CH
DEVICE_REBOOT_NOTIFY2   EQU     002DH
CRIT_REBOOT_NOTIFY2     EQU     002EH
CLOSE_VM_NOTIFY2        EQU     002FH
GET_CONTENTION_HANDLER  EQU     0030H
KERNEL32_INITIALIZED    EQU     0031H
KERNEL32_SHUTDOWN       EQU     0032H
MAX_SYSTEM_CONTROL      EQU     0032H


See also:
        VMMcall System_Control  go
```

VMM_GetVxDLocationList

```
VMM_GetVxDLocationList

include vmm.inc

VMMCall VMM_GetVxDLocationList
jz      ErrorHandler
mov     [TableAddress], eax
mov     [VxDCount], edx
mov     [TableSize], ecx


Returns the address of the VxD location list in EAX,
returns the count of VxDs in EDX, and returns the
list size (in bytes) in ECX. Uses EAX, ECX, and EDX.

        Returns the address of the VxD if the function
        succeeds; otherwise, sets the zero flag.

The VxD location list is a packed array of variable-length
Device_Location_List structures.
```

```
/******************************************************************************
*
*   Object types supported by the vxd loader
*
*  Notes : Low bit of all CODE type objects should be set (VXDLDR uses this)
*           Also Init type objects should be added to the second part of the
*           list (which starts with ICODE_OBJ).
*
******************************************************************************/

#define RCODE_OBJ       -1

#define LCODE_OBJ       0x01
#define LDATA_OBJ       0x02
#define PCODE_OBJ       0x03
#define PDATA_OBJ       0x04
#define SCODE_OBJ       0x05
#define SDATA_OBJ       0x06
#define CODE16_OBJ      0x07
#define LMSG_OBJ        0x08
#define PMSG_OBJ        0x09

#define DBOC_OBJ    0x0B
#define DBOD_OBJ    0x0C

#define PLCODE_OBJ      0x0D
#define PPCODE_OBJ      0x0F

#define ICODE_OBJ       0x11
#define IDATA_OBJ       0x12
#define ICODE16_OBJ     0x13
```

```
#define IMSG_OBJ        0x14


struct ObjectLocation {
    ULONG OL_LinearAddr ;
    ULONG OL_Size ;
    UCHAR  OL_ObjType ;
} ;

#define MAXOBJECTS  25

/*****************************************************************************
 *
 *      Device_Location structure
 *
 *****************************************************************************/

struct Device_Location_List {
    ULONG DLL_DDB ;
    UCHAR DLL_NumObjects ;
    struct ObjectLocation DLL_ObjLocation[1];
};
```

```
PVOID VMM_GetVxDLocationList(PDWORD pVxDCount, PDWORD pTableSize) ;

The VMM_GetVxDLocationList service returns the address of the VxD
location list, returns the count of VxDs, and returns the list size
(in bytes).

Parameter       Description

pVxDCount       Address of location to receive the count of VxDs in the list.
pTableSize      Address of location to receive size of table.

Returns

If the service succeeds, it returns the address of an array of
_DeviceData structures.  Otherwise returns zero.

Comments

The VxD location list is a packed array of variable-length
_DeviceData structures.  Each entry corresponds to one statically
loaded VxD. To get the list of dynamic VxDs, see
VXDLDR_GetDeviceList.

typedef struct  _SegmentData
{
        PVOID           sd_Base;        // base address
        DWORD           sd_Size;        // size of segment
        BYTE            sd_Type;        // segment type code
} SEGMENTDATA, *PSEGMENTDATA;

typedef struct _DeviceData
{
```

```
        PDDB            dd_DDB;         // pointer to ddb
        BYTE            dd_nSegments;   // number of segments
        SEGMENTDATA dd_SegData[0];      // array of SEGMENTDATAs for driver
} DEVDATA, *PDEVDATA;
```

```
see:
        debugsys.inc    from Win95 DDK
```

```
        AX=10h -- ?
        AX=7eh -- DS_GetDebuggerInfo
```

```
        AX=00h --  Display character on debug terminal
                  entry :
                        AL = character to display


        AX=01h -- Read character from debug terminal
                  returns:
                         AL = readed char

        AX=02h -- Displays a string on debug terminal
                   entry:
                         DS:ESI pointer to null terminated string to display

        AX=12h -- Displays a string on debug terminal (called by 16 bit code )
                  entry:
                         DS:SI pointer to null terminated string to display


        AX=40h -- Run debugee until specified  CS:IP  is reached
                  entry :
                         CX = desired CS
                         BX = desires IP

        AX=70h -- Register dot command (32 bit code )
                  entry:
                          BL = dot command to register
                          ESI = linear address of the handler routine
                          EDI = linear address of the help text
                  returns:
                          AX == 0 if successful
                          AX != 0 if registration failed

        AX=71h -- Register dot command (called by 16 bit code )
                   entry:
                          BL = dot command to register
                          CX:SI = linear address of the handler routine
                          DX:DI = linear address of the help text
                  returns:
                          AX == 0 if successful
                          AX != 0 if registration failed


        AX=72h -- Unregister dot command (unregister dot commands registered by both 70h & 71h)
                 entry:
                          BL = dot command to de-register


        AX=73h -- Debug prinf ( C like printf function  >  >  output on debugger terminal ) 32 bi
                  entry:
                         DS:ESI = address of format string
                         DS:EDI = address of first parameter passed ( all parameter are DWORD's )
                  returns:
                         EAX = nr. of characters printed on debug terminal

        AX=74h -- Debug printf (C like printf function  >  >  out on debugger terminal) 16 bit
                  entry:
                        DS:SI = address of format string
                        ES:DI = address of the start of the word or dword arguments
                  returns:
```

```
                        AX = nr of chars outputed

        AX=75h -- Get Register Set
                   entry :
                        DS:ESI = address of a SaveRegs_Struc type  structure

        AX=76h -- Set Alternate Register Set
                    entry:
                         CX = thread ID (0 for current thread)
                         DS:ESI =  address of a SaveRegs_Struc type structure


        AX=77h -- Get Command Line Chararacter
                    entry:
                        BL = 0 - >  get char ,  text pointer not incremented , leading space not
                           = 1 - >  get char , increment text pointer , leading blank is skipped
                           = 2 -? get char , text pointer not incremented ,leading blank is skipp
                    exit:
                        AL = command line character retrieved
                        AH = 0 if EOL encountered , !0 if more characters await parsing

        AX=78h -- Evaluate Expression
                  entry:
                        ds:esi expression to evaluate

                  returns:
                        AX: - >  0, returns a data value
                            - >  !0 returns a linear address
                 CX = TID
                 EBX = evaluated value

        AX=79h -- Verify Memory
                   entry:
                        ECX = length of memory region
                        DS:ESI = starting address of memory to verify

                   returns:
                         AX: - >  0 OK
                             - >  !0 memory range is invalid

        AX=7A -- Directs debugger to dump current registers

        AX=7b -- Directs debugger to perform a stack dump
                  entry:
                       BX:  - >  01h - verbose stack dump
                            - >  02h - 16 bit stack dump
                            - >  04h - 32 bit stack dump


        AX=7dh -- Execute Debugger Command
                   entry:
                        DS:ESI = pointer to the command script
                        CX =     size in bytes of script

;******************************************************************************
;
;  Protected mode Debugger services:
;
;


Debug_Serv_Int       equ 41h    ; Interrupt that calls Deb386 to perform
                                ; debugging I/O, AX selects the function as
                                ; described by the following equates

DS_Out_Char     equ     0       ; function to display the char in DL

DS_In_Char      equ     1       ; function to read a char into AL

DS_Out_Str      equ     2       ; function to display a NUL terminated string
                                ; pointed to by DS:ESI
```

```
DS_Is_Char      equ     3       ; Non blocking In_Chr

DS_Out_Str16    equ    12h      ; function to display a NUL terminated string
                                ; pointed to by DS:SI
                                ; (same as function 2, but for 16 bit callers)

DS_ForcedGO16   equ    40h      ; enter the debugger and perform the equivalent
                                ; of a GO command to force a stop at the
                                ; specified CS:IP
                                ; CX is the desired CS
                                ; BX is the desired IP

DS_LinkMap      equ    45h      ; DX:(E)DI = ptr to paragraph in front of map

DS_UnlinkMap    equ    46h      ; DX:(E)DI = ptr to paragraph in front of map

DS_CheckMap     equ    47h      ; DX:(E)DI = pointer to module name
                                ; returns AX != 0, map found
                                ;         AX == 0, map not found

DS_IsAutoLoadSym equ   48h      ; returns AX != 0, auto load symbols
                                ;         AX == 0, don't auto load symbols

DS_DebLoaded    equ    4Fh      ; check to see if the debugger is installed and
                                ; knows how to deal with protected mode programs
                                ; return AX = F386h, if true
DS_DebPresent   equ   0F386h

DS_LoadSeg      equ    50h      ; define a segment value for the
                                ; debugger's symbol handling
                                ; SI type   0  - code selector
                                ;           1  - data selector
                                ;          80h - code segment
                                ;          81h - data segment
                                ; BX segment #
                                ; CX actual segment/selector
                                ; DX data instance
                                ; ES:(E)DI pointer to module name

DS_LoadSeg_32   equ  0150h      ; Define a 32-bit segment for Windows 32
                                ; SI type   0  - code selector
                                ;           1  - data selector
                                ; DX:EBX points to a D386_Device_Params STRUC
                                ; with all the necessaries in it

DS_MoveSeg      equ    51h      ; notify the debugger that a segment has moved
                                ; BX old segment value
                                ; CX new segment value

DS_FreeSeg      equ    52h      ; notify the debugger that a segment has been
                                ; freed
                                ; BX segment value

DS_FreeSeg_32   equ  0152h      ; notify the debugger that a segment has been
                                ; freed
                                ; BX segment number
                                ; DX:EDI pointer to module name

DS_DGH          equ    56h      ; register "dump global heap" handler
                                ; BX is code offset
                                ; CX is code segment

DS_DFL          equ    57h      ; register "dump free list" handler
                                ; BX is code offset
                                ; CX is code segment

DS_DLL          equ    58h      ; register "dump LRU list" handler
                                ; BX is code offset
                                ; CX is code segment

DS_StartTask    equ    59h      ; notify debugger that a new task is starting
                                ; BX is task handle
```

```
                                ; task's initial registers are stored on the
                                ; stack:
                                ;       push    cs
                                ;       push    ip
                                ;       pusha
                                ;       push    ds
                                ;       push    es
                                ;       push    ss
                                ;       push    sp

DS_Kernel_Vars  equ     5ah     ; Used by the Windows kernel to tell the
                                ; debugger the location of kernel variables
                                ; used in the heap dump commands.
                                ; BX = version number of this data (03a0h)
                                ; DX:CX points to:
                                ;       WORD     hGlobalHeap    ****
                                ;       WORD     pGlobalHeap    ****
                                ;       WORD     hExeHead       ****
                                ;       WORD     hExeSweep
                                ;       WORD     topPDB
                                ;       WORD     headPDB
                                ;       WORD     topsizePDB
                                ;       WORD     headTDB        ****
                                ;       WORD     curTDB         ****
                                ;       WORD     loadTDB
                                ;       WORD     LockTDB
                                ;       WORD     SelTableLen    ****
                                ;       DWORD    SelTableStart  ****
                                ;
                                ; The starred fields are used by the
                                ; heap dump commands which are internal
                                ; to WDEB386.


DS_VCPI_Notify  equ     5bh     ; notify debugger that DOS extender is
                                ; running under a VCPI implementation,
                                ; and register VCPI protect mode interface
                                ; ES:DI points to a data structure used to
                                ; get from V86 mode to Pmode under VCPI.
                                ; This is defined in the VCPI version
                                ; 1.0 spec.

DS_ReleaseSeg   equ     5ch     ; This does the same as a DS_FreeSeg, but
                                ; it restores any breakpoints first.

DS_User_Vars    equ     5dh     ; DS:SI = pointer to an array of offsets:
                                ; BX = windows version
                                ; CX = number of words in array
                                ;       WORD = fDebugUser (1 = DEBUG, 0 = RETAIL)
                                ;       WORD = 16 bit offset to hHmenuSel
                                ;       WORD = 16 bit offset to hHwndSel
                                ;       WORD = 16 bit offset to pclsList
                                ;       WORD = 16 bit offset to pdceFirst
                                ;       WORD = 16 bit offset to hwndDesktop
                                ; This array MUST BE COPIED it goes away
                                ; when we return from this service.

DS_POSTLOAD     =       60h     ; Used by the RegisterPTrace interface
DS_EXITCALL     =       62h     ; Somebody will fill these in if we ever
DS_INT2         =       63h     ; figure out what they are supposed to do.
DS_LOADDLL      =       64h
DS_DELMODULE    =       65h

DS_NEWTASK      =       0BH
DS_FLUSHTASK    =       0CH
DS_SWITCHOUT    =       0DH
DS_SWITCHIN     =       0EH

DS_IntRings     equ     20h     ; function to tell debugger which INT 1's & 3's
                                ; to grab
                                ; BX = 0, grab only ring 0 ints
                                ; BX != 0, grab all ints
```

```
DS_IncludeSegs  equ    21h      ; function to tell debugger to go ahead and
                                ; process INT 1's & 3's which occur in this
                                ; DX:DI points to list of selectors
                                ;   (1 word per entry)
                                ; CX = # of selectors (maximum of 20)
                                ;   CX = 0, to remove the list of segs
MaxDebugSegs = 20

DS_CondBP       equ 0F001h      ; conditional break pt, if the command line
                                ; switch /B is given when the debugger is run
                                ; or the conditional flag is later set, then
                                ; this int should cause the program to break
                                ; into the debugger, else this int should be
                                ; ignored!
                                ; ESI points to a nul terminated string to
                                ; display if break is to happen.

DS_ForcedBP     equ 0F002h      ; break pt, which accomplishes the same thing
                                ; as an INT 1 or an INT 3, but is a break point
                                ; that should be permanently left in the code,
                                ; so that a random search of source code would
                                ; not result in the accidental removal of this
                                ; necessary break_pt

DS_ForcedGO     equ 0F003h      ; enter the debugger and perform the equivalent
                                ; of a GO command to force a stop at the
                                ; specified CS:EIP
                                ; CX is the desired CS
                                ; EBX is the desired EIP

DS_HardINT1     equ 0F004h      ; check to see if INT 1 hooked for all rings
                                ; ENTER: nothing
                                ; EXIT: AX = 0, if no, 1, if yes

DS_Out_Symbol   equ    0Fh      ; find the symbol nearest to the address in
                                ; CX:EBX and display the result in the format
                                ; symbol name  < +offset >
                                ; the offset is only included if needed, and
                                ; no CR&LF is displayed

DS_Disasm_Ins   equ    10h      ; function to disassemble the instruction
                                ; pointed to by DS:ESI

DS_JumpTableStart      equ 70h


;**     DS_RegisterDotCommand
;
;       This interface is used to register wdeb386 dot commands by FLAT 32
;       bit code.  The following conditions apply:
;
;       * The code will be run at ring 0
;       * Interrupts may be enabled
;       * Must not access any invalid pages or load invalid selectors
;       * Must stay on the stack called with when calling INT 41 services
;       * Must not change DS or ES from the FLAT selector
;
;       The help text is printed when .? is executed in the order of
;       registration.  The text must include CR/LF at the end; nothing
;       is added to the help text.
;
;       ENTRY:  (AX) = 0070h
;               (BL) = dot command to register
;               (ESI) = linear address of dot command routine
;                   Dot command routine:
;                       ENTRY:  (AL) = command character
;                               (DS, ES) = flat data selector
;
;                       EXIT:   (AX) == 0, no errors
;                               (AX) != 0, command line or option error
;
```

```
;                       NOTE:   MUST return with a 32 bit FAR return (retfd)
;               (EDI) = linear address of help text
;
;       EXIT:   (AX) == 0, no errors
;               (AX) != 0, dot command already used or out of dot commands

DS_RegisterDotCommand  equ 70h


;**     DS_RegisterDotCommand16
;
;       This interface is used to register wdeb386 dot commands by 16 bit
;       code.  The following conditions apply:
;
;       * The code will be run at ring 0 or in real mode
;       * Interrupts may not be enabled
;       * Must not access any not present pages or load invalid selectors
;       * Must stay on the stack called with when calling INT 41 services
;
;       The help text is printed when .? is executed in the order of
;       registration.  The text must include CR/LF at the end; nothing
;       is added to the help text.
;
;       ENTRY:  (AX) = 0071h
;               (BL) = dot command to register
;               (CX:SI) = address of dot command routine
;                   Dot command routine:
;                       ENTRY:  (AL) = command character
;                               (DS, ES) = debugger's data selector
;
;                       EXIT:   (AX) == 0, no errors
;                               (AX) != 0, command line or option error
;
;                       NOTE:   MUST return with a 16 bit FAR return (retf)
;               (DX:DI) = address of help text
;
;       EXIT:   (AX) == 0, no errors
;               (AX) != 0, dot command already used or out of dot commands

DS_RegisterDotCommand16 equ 71h


;**     DS_DeRegisterDotCommand
;
;       This interface is used to de-register wdeb386 dot commands registered
;       by the above 16 or 32 bit services.  Care should be used not to
;       de-register dot commands that weren't registered by your code.
;
;       ENTRY:  (AX) = 0072h
;               (BL) = dot command to de-register
;
;       EXIT:   NONE

DS_DeRegisterDotCommand equ 72h

;**     DS_Printf
;
;       This function allows formatted output with the standard "C"
;       printf syntax.
;
;       ENTRY:  (AX) = 0073h
;               (DS:ESI) = address of format string
;               (ES:EDI) = address of the start of the dword arguments
;
;       EXIT:   (EAX) = number of characters printed
;
;       Supported types are:
;
;       %%                                                      %
;       %[l][h]c                                                character
;       %[-][+][ ][0][width][.precision][l|h|p|n]d              decimal
;       %[-][0][width][.precision][l|h|p|n]u                    unsigned decimal
```

```
;       %[-][#][0][width][.precision][l|h|p|n]x                 hex
;       %[-][#][0][width][.precision][l|h|p|n]X                 hex
;       %[-][0][width][.precision][l|h|p|n]o                    octal
;       %[-][0][width][.precision][l|h|p|n]b                    binary
;       %[-][width][.precision][l|h|a|F|R|P]s                   string
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]A       address
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]S       symbol
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]G       group:symbol
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]M       map:group:symbol
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]g       group
;       %[-][width][.precision][l|h|a|p|n|F|L|R|L|H|N|Z]m       map
;
;       Where "width" or "precision" is a decimal number or the '*'
;       character; '*' causes the field width or precision to be picked
;       up from the next parameter. []'ed parameters are optional.
;
;
;       "\r", "\t", "\n", "\a", "\b", are supported directly.
;
;       Prefixes
;       --------
;
;       Used with c,d,u,x,X,o,b:
;
;       Parameter Argument Size
;       -----------------------
;       word                                    h
;       dword                                   l
;
;       Used with s,A,S,G,M,g,m:
;
;       Address Argument Size
;       ---------------------
;       16 bit DS relative                      h
;       16:16 segment:offset                    hF or Fh
;       32 bit flat relative                    l
;       16:32 segment:offset (2 dwords)         lF or Fl
;       pointer to AddrS structure              a
;       segment is a real mode segment          R
;       segment is a protected mode selector    P
;
;       Default segment type is the current code type.
;
;       Used with A,S,G,M,g,m:
;
;       Address Display Size or Format
;       ------------------------------
;       16 bit offset                           H
;       32 bit offset                           L
;       offset only                             N
;       no address                              Z
;
;       Default display size depends on the "386env" flag setting.
;
;       Used with S,G,M,g,m:
;
;       gets the previous symbol                p
;       gets the next symbol                    n
;
;       Used with A:
;
;       gets the previous symbol address        p
;       gets the next symbol address            n
;
;       Used with d,u,x,X,o,b:
;
;       gets the previous symbol offset         p
;       gets the next symbol offset             n
;

DS_Printf       equ     73h
```

```
;**     DS_Printf16
;
;       This function allows formatted output with the standard "C"
;       printf syntax.
;
;       ENTRY:  (AX) = 0074h
;               (DS:SI) = address of format string
;               (ES:DI) = address of the start of the word or dword arguments
;
;       EXIT:   (AX) = number of characters printed
;
;       The format options and parameters are the same as DS_Printf except
;       the default parameter size is a word (the h option is implicit).
;

DS_Printf16 equ 74h

;**     DS_GetRegisterSet
;
;       This function copies the current register set.
;
;       ENTRY:  (AX) = 0075h
;               (DS:ESI) = address of SaveRegs_Struc structure
;
;       EXIT:   NONE
;

DS_GetRegisterSet equ 75h

;**     DS_SetAlternateRegisterSet
;
;       This function temporary sets the debugger's registers to values
;       passed in the structure.  If an "r" command is executed or the
;       debugged code is returned to (via the "g", "t" or "p" commands),
;       the register set reverts to the debugged code's registers.
;
;       ENTRY:  (AX) = 0076h
;               (CX) = thread ID, 0 use current thread ID
;               (DS:ESI) = address of SaveRegs_Struc structure
;
;       EXIT:   NONE
;

DS_SetAlternateRegisterSet equ 76h

;**     DS_GetCommandLineChar
;
;       This services gets the next character off the command line.
;
;       ENTRY:  (AX) = 0077h
;               (BL) = 0 just peek the character, don't increment text pointer
;                        leading white space isn't ignored
;               (BL) = 1 get the character, increment text pointer
;                        leading white space is skipped
;               (BL) = 2 peek the character, don't increment text pointer
;                        leading white space is skipped
;
;       EXIT:   (AL) = command line character
;               (AH) == 0 if no more characters (EOL)
;               (AH) != 0 if more characters
;

DS_GetCommandLineChar equ 77h

;**     DS_EvaluateExpression
;
;       Expressions can be numbers of various radices, symbols, addresses
;       or an combination of the above hooked together with various
;       operators.  Expressions are separated by blanks or commas.  This
;       function is passed a pointer to the beginning of the text of the
;       expression (i.e. "%80003444+4232").  The expression is either
;       evaluated down into a dword value if there are no addresses or
```

```
;       into a linear address.
;
;       ENTRY:  (AX) = 0078h
;
;       EXIT:   (AX) == 0, returning a data value
;               (AX) != 0, returning a linear address
;               (CX) = thread id
;               (EBX) = return value
;
;       NOTE:   If the expression is invalid, this service will not
;               return.  A message is printed and control returns to
;               the command loop.
;

DS_EvaluateExpression equ 78h

;**     DS_VerifyMemory
;
;       ENTRY:  (AX) = 0079h
;               (ECX) = length of memory region
;               (DS:ESI) = address of memory to verify
;
;       EXIT:   (AX) == 0, no errors
;               (AX) != 0, invalid memory

DS_VerifyMemory equ 79h

;**     DS_PrintRegisters
;
;       This function prints (just like the "r" command) the either the
;       debugged code's registers or the alternate register set, set with
;       DS_SetAlternateRegisterSet function.
;
;       ENTRY:  (AX) = 007ah
;
;       EXIT:   NONE
;
;       NOTE:   If the CS:EIP is invalid, this service will not return
;               because of an error when the code is disassembled.  A
;               message is printed and control returns to the command loop.
;

DS_PrintRegisters equ 7ah

;**     DS_PrintStackDump
;
;       This function prints (just like the "k" command) the stack dump
;       based on the current register set that may have been set with
;       DS_SetAlternateRegisterSet function.
;
;       ENTRY:  (AX) = 007bh
;               (BX) = flags
;                       01h - verbose stack dump
;                       02h - 16 bit stack dump
;                       04h - 32 bit stack dump
;
;       EXIT:   NONE
;
;       NOTE:   If the CS:EIP or SS:EBP are invalid, this service will not
;               return because of an error when accessing the stack.  A
;               message is printed and control returns to the command loop.
;

DS_PrintStackDump equ 7bh

;**     DS_SetThreadID
;
;       This function sets what the debugger thinks the thread ID is
;       for memory address in other address contexts.  It stays set
;       until the debugged code is returned to (via "g", "t" or "p")
;       or set back to 0.
;
```

```
;       ENTRY:  (AX) = 007ch
;               (CX) = thread ID or 0 for currently executed thread
;
;       EXIT:   NONE

DS_SetThreadID  equ  7ch

;**     DS_ExecDebugCommand
;
;       This service allows any debugger command to be executed.  In can
;       be a multi-lined script with the lines separated by CR, LF.  MUST
;       have a "g" command at the end of script so the debugger doesn't
;       stop while in the INT 41.
;
;       ENTRY:  (AX) = 007dh
;               (DS:ESI) = pointer to debugger command script string
;               (CX) = size of script
;
;       EXIT:   NONE
;
;       NOTE:   If the any kind of error happens, this service will not
;               return.  A message is printed and control returns to the
;               command loop.
;

DS_ExecDebugCommand equ  7dh

;**     DS_GetDebuggerInfo
;
;       This service returns various debugger info and routines.
;
;       ENTRY:  (AX) = 007eh
;               (DS:ESI) = pointer to DebInfoBuf structure
;               (ECX) = size of the above buffer.  Only this many bytes are
;                       copied to the buffer; this allows more info to be
;                       passed in future versions without breaking anything.
;
;       EXIT:   (AX) == 0, no errors
;               (AX) != 0, error:   (AX) == 007eh, function not implemented
;                                   (AX) == anything else, invalid buffer

DS_GetDebuggerInfo equ 7eh

;**     DS_CheckFault
;
;       This service checks if the debugger wants control on the fault.
;
;       ENTRY:  (AX) = 007fh
;               (BX) = fault number
;               (CX) = fault type mask
;                       DEBUG_FAULT_TYPE_V86
;                       DEBUG_FAULT_TYPE_PM
;                       DEBUG_FAULT_TYPE_RING0
;                       DEBUG_FAULT_TYPE_FIRST
;                       DEBUG_FAULT_TYPE_LAST
;
;       EXIT:   (AX) == 0, handle fault normally
;               (AX) != 0, handled by debugger

DS_CheckFault equ 7fh

;**     DS_SetBreak
;
;       This service allows an error break or ctrl-c handler to be
;       set.  The old value that is returned must be save and set
;       back to remove the break handler.
;
;       ENTRY:  (AX) = 0080h
;               (DS:ESI) = pointer to BreakStruc with the CS:EIP and
;               SS:ESP values to be used when a error break or ctrl-c
;               happens.  The old value is copied into this buffer.
;
```

```
;       EXIT:   (AX) == 0, no error
;               (AX) != 0, error on BreakStruc address

DS_SetBreak equ 80h

;**     DS_RedirectExec
;
;       This service redirects the input and output streams to the
;       specified buffers for the debugger command line passed.
;
;       ENTRY:  (AX) = 0081h
;               (DS:ESI) = pointer to RedirectExecStruc
;
;       EXIT:   (ECX) = number of bytes written
;               (AX) == 0, no error
;               (AX) != 0, error
;                       1 to 10 = memory access error
;                       -1      = buffer overflow
;                       -2      = invalid parameter, not on a 386, or reentered

DS_RedirectExec equ 81h

;**     DS_PassOnDebugCommand
;
;       Used to tell the debugger to pass this dot command on to the
;       next handler.
;
;       ENTRY:  (AX) = 0082h
;
;       EXIT:   NONE
;

DS_PassOnDebugCommand equ 82h

;**     DS_TrapFault
;
;       Allows ring 3 code to send a fault to the debugger
;
;       ENTRY:  (AX) = 0083h
;               (BX) = fault number
;               (CX) = faulting CS
;               (EDX) = faulting EIP
;               (ESI) = fault error code
;               (EDI) = faulting flags
;
;       EXIT:   (CX) = replacement CS
;               (EDX) = replacement EIP

DS_TrapFault equ 83h

;**     DS_SetStackTraceCallBack
;
;       Sets the "k" command callback filter used to back trace
;       thru thunks.
;
;       ENTRY:  (AX) = 0084h
;               (EBX) = linear address of call back routine, zero to uninstall
;               (ECX) = linear address of the end of the call back routine
;               (EDX) = EIP to use for for faults in call back routine
;
;       EXIT:   NONE
;
;       CALLBACK:
;               ENTRY:  (EAX) = linear base of SS
;                       (EBX) = linear address of SS:EBP
;                       (DS, ES) = flat ds
;                       (SS) = NOT flat ds !!!!!!!!!
;
;               EXIT:   (EAX) = FALSE, no thunk
;                               TRUE, is a thunk
;                                       (CX:ESI) = new SS:EBP
;                                       (DX:EDI) = new CS:EIP
```

```
;

DS_SetStackTraceCallBack equ 84h

;**     DS_RemoveSegs
;
;       Removes all the undefined groups from a map file.
;
;       ENTRY:  (AX) = 0085h
;               (ES:EDI) pointer to module name
;
;       EXIT:   NONE
;

DS_RemoveSegs equ 85h

;**     DS_DefineDebugSegs
;
;       Defines the debugger's code and data symbols.
;
;       ENTRY:  (AX) = 0086h
;
;       EXIT:   NONE
;

DS_DefineDebugSegs equ 86h

;**     DS_SetBaudRate
;
;       Sets the com port's baud rate
;
;       ENTRY:  (AX) = 0087h
;               (BX) = baud rate
;
;       EXIT:   NONE
;

DS_SetBaudRate equ 87h

;**     DS_SetComPort
;
;       Sets the com port's baud rate
;
;       ENTRY:  (AX) = 0088h
;               (BX) = com port number
;
;       EXIT:   (AX) == 0, ok
;               (AX) != 0, error bad com port
;

DS_SetComPort equ 88h

;**     DS_ChangeTaskNum
;
;       Changes a task number to another task number or
;       indicates that the task has gone away.
;
;       ENTRY:  (AX) = 0089h
;               (CX) = old task number
;               (DX) = new task number or -1 if process terminated.
;
;       EXIT:   NONE
;

DS_ChangeTaskNum equ 89h

;**     DS_ExitCleanup
;
;       Called when Windows exits.
;
;       ENTRY:  (AX) = 008ah
;
```

```
;       EXIT:   NONE
;

DS_ExitCleanup equ 8ah

;**     DS_InstallVGAHandler
;
;       Called when the Debug VxD is initializing (during Device Init),
;       to specify an alternate I/O handler for VGA.  The handler accepts the
;       following inputs:
;
;           BX == subfunction #:
;                   0 == save screen state (switch to debugger context)
;                        No inputs/outputs
;                   1 == restore screen state (switch to windows context)
;                        No inputs/outputs
;                   2 == display output character (ie, OEMOutputCharCOM)
;                        on input, AL == character
;                   3 == check for input character (ie, OEMScanCharCOM)
;                        on output, ZR if no chars, else NZ and AL == char
;
;       ENTRY:  (AX) = 008bh
;               (DX:EDI) == 16:32 address to call, with BX == subfunction above
;
;       EXIT:   NONE
;

DS_InstallVGAHandler equ 8bh

;**     DS_GetComBase
;
;       Called when Debug VxD is initializing (during Device Init),
;       to get the base of the com port being used by wdeb386.
;
;       Entry:
;               (AX) == 008ch
;
;       Exit:
;               (AX) = base of COM port.
;

DS_GetComBase   equ     8ch

;**     DS_GetSymbol
;
;       Looks up a symbol and returns the linear address and segment/offset.
;
;       ENTRY:  (AX) == 008dh
;               (DS:ESI) = ptr to null-terminated symbol
;
;       EXIT:   (AX) == 0, no error
;               (AX) == 1, symbol not found
;               (AX) == 2, memory not loaded yet
;               (ECX) = linear address of variable  (if AX == 0)
;               (EDX) = seg:offset of variable      (if AX == 0)
;

DS_GetSymbol equ 8dh

;**     DS_CopyMem
;
;       Copys memory from one AddrS to another AddrS
;
;       ENTRY:  (AX) == 008eh
;               (DS:ESI) = pointer to source AddrS block
;               (ES:EDI) = pointer to destination AddrS block
;               (CX) = number of bytes
;
;       EXIT:   (AX) == 0, no error
;               (AX) != 0, invalid address
;
```

```
DS_CopyMem equ 8eh

DS_JumpTableEnd equ 8eh

SaveRegs_Struc  struc
        Debug_EAX               dd      ?
        Debug_EBX               dd      ?
        Debug_ECX               dd      ?
        Debug_EDX               dd      ?
        Debug_ESP               dd      ?
        Debug_EBP               dd      ?
        Debug_ESI               dd      ?
        Debug_EDI               dd      ?
        Debug_ES                dw      ?
        Debug_SS                dw      ?
        Debug_DS                dw      ?
        Debug_FS                dw      ?
        Debug_GS                dw      ?
        Debug_EIP               dd      ?
        Debug_CS                dw      ?
                                dd      ?
        Debug_EFlags            dd      ?
        Debug_CR0               dd      ?
        Debug_GDT               dq      ?
        Debug_IDT               dq      ?
        Debug_LDT               dw      ?
        Debug_TR                dw      ?
        Debug_CR2               dd      ?
        Debug_CR3               dd      ?
        Debug_DR0               dd      ?
        Debug_DR1               dd      ?
        Debug_DR2               dd      ?
        Debug_DR3               dd      ?
        Debug_DR6               dd      ?
        Debug_DR7               dd      ?
        Debug_DR7_2             dd      ?
        Debug_TR6               dd      ?
        Debug_TR7               dd      ?
        Debug_TrapNumber        dw      -1      ; -1 means no trap number
        Debug_ErrorCode         dw      0       ; 0 means no error code
SaveRegs_Struc ends

DebInfoBuf      struc
        DIB_MajorVersion        db      0
        DIB_MinorVersion        db      0
        DIB_Revision            db      0
                                db      0       ; reserved
        DIB_DebugTrap16         dd      0       ; send 16 bit trap to debugger
        DIB_DebugTrap32         df      0       ; send 32 bit trap to debugger
        DIB_DebugBreak16        dd      0       ; 16 bit break in debugger
        DIB_DebugBreak32        df      0       ; 32 bit break in debugger
        DIB_DebugCtrlC16        dd      0       ; 16 bit check for ctrl C
        DIB_DebugCtrlC32        df      0       ; 32 bit check for ctrl C
DebInfoBuf      ends

BreakStruc      struc
        BS_BreakEIP             dd      0       ; CS:EIP, SS:ESP to go to
        BS_BreakCS              dw      0       ;  on a error or ctrlc break
        BS_BreakESP             dd      0
        BS_BreakSS              dw      0
BreakStruc      ends

RedirectExecStruc       struc
        RDE_fpbufDebugCommand   df      0       ; debugger command script
        RDE_cbDebugCommand      dw      0       ; debugger command script len
        RDE_fpszInput           df      0       ; input stream pointer
        RDE_usFlags             dw      0       ; reserved (must be 0)
        RDE_cbOutput            dd      0       ; size of output buffer
        RDE_fpbufOutput         df      0       ; output buffer pointer
RedirectExecStruc       ends

REPEAT_FOREVER_CHAR     equ     0feh            ; send next character until
```

```
                                                ;  end of debugger command

AddrS   struc                                   ; for printf service
        AddrOff         dd      0
        AddrSeg         dw      0
        AddrType        db      0
        AddrSize        db      0
        AddrTask        dw      0
AddrS   ends

AddrTypeSize    equ     word ptr AddrType

EXPR_TYPE_SEG           equ     00000001b       ; address type segment:offset
EXPR_TYPE_SEL           equ     00001001b       ; address type selector:offset
EXPR_TYPE_LIN           equ     00000010b       ; address type linear
EXPR_TYPE_PHY           equ     00001010b       ; address type physical
EXPR_TYPE_LOG           equ     00001000b       ; logical address (no sel yet)

DEBUG_FAULT_TYPE_V86            equ     00000001b
DEBUG_FAULT_TYPE_PM             equ     00000010b
DEBUG_FAULT_TYPE_RING0          equ     00000100b
DEBUG_FAULT_TYPE_FIRST          equ     00001000b
DEBUG_FAULT_TYPE_LAST           equ     00010000b



INT 41h .............
       1.装载检查
         Entry:    AL=4FH
         Return :  AX=F386H
       2.装入段
         Entry:    AL=50H , BX=逻辑段号 , CX=段址或选择子
                   SI=标志（1-数据段，0-代码段）
                   ES:（E）DI=段所属的模块名
         Return:   None
       3.释放段
         Entry:    AL=52H , BX=释放段的选择子
         Return:   None
       4.Entry:    AL=5AH , BX=Windows Version ,
                   DX:CX指向THHOOK
         Return:   None
       5.释放段（在KERNEL启动时调用，CS，DS各一次）
         Entry:    AL=5CH , BX=段选择子
         Return:   None
       6.程序终止
         Entry:    AL=62H , BL=退出码
         Return:   None
       7.DLL启动
         Entry:    AX=64H , CX:BX指向DLL入口点出 , SI=模块句柄
         Return:   None
       8.移去模块
         Entry:    AX=65H , ES=模块句柄
         Return:   None
       以上是Windows 3.1的INT41H接口，我发现WIN95中还有很多新的调用，
 我研究出两个：AX=150H和AX=152H，这是载入/释放32位段，EBX指向一数据结构，
 包括模块名，地址等，具体结构我还不是很清楚。
```

Make Files

- VC++ 2.0 make Win32 console
- BC++ 5.0 TO MAKE A WIN32 CONSOLE PROGRAM

VC++ 2.0 make Win32 console

```
# set path to your VC++ 2.0 directory

PATHVC = E:\VXD\C32
PATH = $(PATHVC)\BIN;$(PATH)

OBJS= trwload.obj

LIBS=kernel32.lib user32.lib

win32app.exe: $(OBJS)
   SET LIB=$(PATHVC)\LIB
   SET LINK= $(LIBS) /NOLOGO /SUBSYSTEM:console /INCREMENTAL:no \
        /PDB:none /MACHINE:I386
   link /OUT:TRWLOAD.exe $(OBJS)

.c.obj:
   SET INCLUDE=$(PATHVC)\INCLUDE
   SET CL=/c /ML /GX /YX /Od /D "WIN32" /D "NDEBUG" /D "_CONSOLE"
   cl  $<


# THIS IS MAKFILE FOR BC++5.0 TO MAKE A WIN32 CONSOLE PROGRAM

FTLINK32=-v -LF:\BC5\LIB -Tpe -ap -V4.0 -c -x
LIBS    =import32.lib cw32mt.lib

FBCC32  =-c -w -R -v -vi -H -H=F.csm -WM -W


ALL     = f.exe
OBJS    = f.obj

f.exe : $(OBJS)
        SET LIB=F:\BC5\LIB
        TLINK32 $(FTLINK32) c0x32.obj $(OBJS), $<,$*,$(LIBS)

f.obj : f.cpp
SET INCLUDE=F:\BC5\INCLUDE
```

```
BCC32 $(FBCC32) -o$@ f.cpp
```

IDC overall

See also:
IDA pro
Structures defined in IDC
Enums defined in IDC

```
IDC files:

        Sym_in.idc by myself            go
        Fultree.idc                     go
```

```
        IDC define of ENUMs
VxDcall
Ring0Win32Services
VMM_MESSAGES
error_code

//--------------------------------------------------------------------------
// Information about enum types

static Enums(void) {
        auto id;                // enum id
        auto cid;               // const id


        id = AddEnum(0,"VxDcall",0x1100000);
        cid = AddConst(id,"K0_3_PageReserve",           0x10000);
        cid = AddConst(id,"K0_5_PageCommit" ,           0x10001);
        cid = AddConst(id,"K0_3_PageDecommit",          0x10002);
        cid = AddConst(id,"K0_1_PageRegister",          0x10003);
        cid = AddConst(id,"K0_2_PageQuery",             0x10004);
        cid = AddConst(id,"K0_2_HeapAllocate",          0x10005);
        cid = AddConst(id,"K0_0_ContextCreate",         0x10006);
        cid = AddConst(id,"K0_1_ContextDestroy",        0x10007);
        cid = AddConst(id,"K0_4_PageAttach",            0x10008);
        cid = AddConst(id,"K0_2_PageFlush",             0x10009);
        cid = AddConst(id,"K0_2_PageFree",              0x1000a);
        cid = AddConst(id,"K0_1_ContextSwitch",         0x1000b);
        cid = AddConst(id,"K0_3_HeapReAllocate" ,       0x1000c);
        cid = AddConst(id,"K0_4_PageModifyPermisions",  0x1000d);
        cid = AddConst(id,"K0_3_PageQuery" ,            0x1000e);
        cid = AddConst(id,"K0_0_GetCurrentContext" ,    0x1000f);
        cid = AddConst(id,"K0_2_HeapFree",              0x10010);
        cid = AddConst(id,"K0_3_RegOpenKey",            0x10011);
        cid = AddConst(id,"K0_3_RegCreateKey",          0x10012);
        cid = AddConst(id,"K0_1_RegCloseKey",           0x10013);
        cid = AddConst(id,"K0_2_RegDeleteKey" ,         0x10014);
        cid = AddConst(id,"K0_5_RegSetValue" ,          0x10015);
        cid = AddConst(id,"K0_2_RegDeleteValue" ,       0x10016);
        cid = AddConst(id,"K0_4_RegQueryValue" ,        0x10017);
        cid = AddConst(id,"K0_4_RegEnumKey" ,           0x10018);
        cid = AddConst(id,"K0_8_RegEnumValue" ,         0x10019);
        cid = AddConst(id,"K0_6_RegQueryValueEx",       0x1001a);
        cid = AddConst(id,"K0_6_RegSetValueEx",         0x1001b);
        cid = AddConst(id,"K0_1_RegFlushKey" ,          0x1001c);
        cid = AddConst(id,"K0_6_VmmUnkSub1" ,           0x1001d);
        cid = AddConst(id,"K0_2_GetDemandPageInfo" ,    0x1001e);
        cid = AddConst(id,"K0_2_BlockOnID",             0x1001f);
        cid = AddConst(id,"K0_1_SignalID" ,             0x10020);
        cid = AddConst(id,"K0_3_RegLoadKey" ,           0x10021);
        cid = AddConst(id,"K0_2_RegUnLoadKey" ,         0x10022);
        cid = AddConst(id,"K0_3_RegSaveKey" ,           0x10023);
        cid = AddConst(id,"K0_2_RegRemapPreDefKey" ,    0x10024);
        cid = AddConst(id,"K0_5_PageChangePager" ,      0x10025);
        cid = AddConst(id,"K0_5_RegQueryMultipleValues",0x10026);
        cid = AddConst(id,"K0_4_RegReplaceKey" ,        0x10027);
        cid = AddConst(id,"K0_0_10028" ,                0x10028);
```

```
        cid = AddConst(id,"K0_20000",0x20000);

        cid = AddConst(id,"VxDcall_90000",0x90000);
        cid = AddConst(id,"VxDcall_90001",0x90001);

        cid = AddConst(id,"K0_0_GetVersion" ,           0x2A0000);
        cid = AddConst(id,"K0_a_LoadVWin32CodePointers",0x2A0001);
        cid = AddConst(id,"K0_0_GetSystemTime" ,        0x2A0002);
        cid = AddConst(id,"K0_8_GetVWin32PointersFromCaller" , 0x2A0003);
        cid = AddConst(id,"K0_1_BlockOnSemaphore" ,     0x2A0004);
        cid = AddConst(id,"K0_1_Signal_Sema_NoSwtch",   0x2a0005);
        cid = AddConst(id,"K0_3_Create_Semaphore",      0x2a0006);
        cid = AddConst(id,"K0_1_Destroy_Semaphore" ,    0x2A0007);
        cid = AddConst(id,"K0_9_VWIN32_CreateThread" ,  0x2A0008);
        cid = AddConst(id,"K0_1_Sleep",                 0x2a0009);
        cid = AddConst(id,"K0_2_WakeThread" ,           0x2A000a);
        cid = AddConst(id,"K0_2_TerminateThread" ,      0x2A000b);
        cid = AddConst(id,"K0_1_2a000c",                0x2a000c);
        cid = AddConst(id,"K0_3_VWIN32_QueueUserApc" ,  0x2A000d);
        cid = AddConst(id,"K0_0_VWIN32_Initialize" ,    0x2A000e);
        cid = AddConst(id,"K0_4_VWIN32_QueueKernelApc" ,0x2A000f);
        cid = AddConst(id,"K0_2_Int21_Call",            0x2a0010);
        cid = AddConst(id,"K0_6_IFSMgr_Win32DupHandle", 0x2a0011);
        cid = AddConst(id,"K0_1_BlockThreadSetBit" ,    0x2A0012);
        cid = AddConst(id,"K0_2_AdjustThreadExecPrio",  0x2a0013);
        cid = AddConst(id,"K0_2_GetThreadContext",      0x2a0014);
        cid = AddConst(id,"K0_2_SetThreadContext",      0x2a0015);
        cid = AddConst(id,"K0_5_ReadProcessMemory" ,    0x2A0016);
        cid = AddConst(id,"K0_5_WriteProcessMemory" ,   0x2A0017);
        cid = AddConst(id,"K0_1_VMCPD_Get_CR0_State" ,  0x2A0018);
        cid = AddConst(id,"K0_2_VMCPD_Set_CR0_State" ,  0x2A0019);
        cid = AddConst(id,"K0_1_SuspendThread",         0x2a001a);
        cid = AddConst(id,"K0_1_ResumeThread",          0x2a001b);
        cid = AddConst(id,"K0_0_2A001c",                0x2a001c);
        cid = AddConst(id,"K0_1_WaitCrst",              0x2a001d);
        cid = AddConst(id,"K0_1_WakeCrst",              0x2a001e);
        cid = AddConst(id,"K0_b_2a001f",                0x2a001f);
        cid = AddConst(id,"K0_1_VMCPD_Get_Version" ,    0x2A0020);
        cid = AddConst(id,"K0_2_Set_Thread_Win32_Pri",  0x2a0021);
        cid = AddConst(id,"K0_3_Boost_With_Decay" ,     0x2A0022);
        cid = AddConst(id,"K0_4_Set_Inversion_Pri" ,    0x2A0023);
        cid = AddConst(id,"K0_2_Release_Inversion_Pri_ID",0x2A0024);
        cid = AddConst(id,"K0_1_Release_Inversion_Pri" ,0x2A0025);
        cid = AddConst(id,"K0_2_Attach_Thread_to_Group",0x2a0026);
        cid = AddConst(id,"K0_2_Set_Thread_Static_Boost" ,0x2A0027);
        cid = AddConst(id,"K0_2_Set_Group_Static_Boost" , 0x2A0028);
        cid = AddConst(id,"K0_2_Int31_Call",            0x2a0029);
        cid = AddConst(id,"K0_1_Int41_Call" ,           0x2A002a);
        cid = AddConst(id,"K0_0_BlockForTermination" ,  0x2A002b);
        cid = AddConst(id,"K0_1_TerminationHandler_1" , 0x2A002c);
        cid = AddConst(id,"K0_2_2a002d_2",              0x2a002d);
        cid = AddConst(id,"K0_3_BlockSingleWnod" ,      0x2A002e);
        cid = AddConst(id,"K0_5_BlockMultipleWnod" ,    0x2A002f);
        cid = AddConst(id,"K0_2_VWIN32_SetEvent" ,      0x2A0030);
        cid = AddConst(id,"K0_0_2A0031",                0x2a0031);
        cid = AddConst(id,"K0_1_2a0032",                0x2a0032);
        cid = AddConst(id,"K0_1_InitUserAPCList" ,      0x2A0033);
        cid = AddConst(id,"K0_1_2a0034",                0x2a0034);
```

```
        cid = AddConst(id,"K0_1_Signal_Sema_NoSwth" ,   0x2A0035);
        cid = AddConst(id,"K0_1_SysCtl_Krnl32_initialized",0x2a0036);
        cid = AddConst(id,"K0_3_CommonFaultPopup" ,     0x2A0037);
        cid = AddConst(id,"K0_0_VWIN32_ForceCrsts" ,    0x2A0038);
        cid = AddConst(id,"K0_2_2a0039",                0x2a0039);
        cid = AddConst(id,"K0_0_FreezeAllThreads" ,     0x2A003a);
        cid = AddConst(id,"K0_0_UnFreezeAllThreads" ,   0x2A003b);
        cid = AddConst(id,"K0_1_IFSMgr_Ring0_FileIO",   0x2a003c);
        cid = AddConst(id,"K0_2_Attach_Thread_To_Group",0x2A003d);
        cid = AddConst(id,"K0_0_ActiveTimeBiasSet" ,    0x2A003e);
        cid = AddConst(id,"K0_5_ModifyPagePermission" , 0x2A003f);
        cid = AddConst(id,"K0_4_2a0040",                0x2a0040);
        cid = AddConst(id,"K0_2_ForceLeaveCrst" ,       0x2A0041);
        cid = AddConst(id,"K0_3_ForceEnterCrst" ,       0x2A0042);
        cid = AddConst(id,"K0_2_VMCPD_SetThreadExcptType",0x2A0043);
        cid = AddConst(id,"K0_1_VTD_Get_Real_Time" ,    0x2A0044);
        cid = AddConst(id,"K0_0_Sysctl_SetDeviceFocus", 0x2A0045);
        cid = AddConst(id,"K0_1_UnFreezeThread" ,       0x2A0046);
        cid = AddConst(id,"K0_1_VMM_Replace_Global_Env",0x2A0047);
        cid = AddConst(id,"K0_0_Sysctl_KERNEL32_SHUTDOWN",0x2A0048);
        cid = AddConst(id,"K0_3_SubUnk4" ,              0x2A0049);
        cid = AddConst(id,"K0_2_VWIN32_AddSysCrst" ,    0x2A004a);
        cid = AddConst(id,"K0_3_VWIN32_AddSysCrst" ,    0x2A004b);
        cid = AddConst(id,"K0_1_Cancel_Time_Out_1" ,    0x2A004c);
        cid = AddConst(id,"K0_1_2a004d",                0x2a004d);
        cid = AddConst(id,"K0_1_Set_Reflect_Kkey" ,     0x2A004e);

//      cid = AddConst(id,"K0_VWIN32_Int41Dispatch" ,   0x2A002a);

        cid = AddConst(id,"K0_2b0015_?",0x2b0015);

        cid = AddConst(id,"K0_380002",0x380002);
        cid = AddConst(id,"K0_38000a",0x38000a);
        cid = AddConst(id,"K0_38000c",0x38000c);
        cid = AddConst(id,"K0_38000d",0x38000d);
        cid = AddConst(id,"K0_38000e",0x38000e);
        cid = AddConst(id,"K0_380013",0x380013);
        cid = AddConst(id,"K0_380014",0x380014);
        cid = AddConst(id,"K0_380015",0x380015);
        cid = AddConst(id,"K0_380017",0x380017);
        cid = AddConst(id,"K0_380018",0x380018);
        cid = AddConst(id,"K0_38001b",0x38001b);
        cid = AddConst(id,"K0_38001d",0x38001d);
        cid = AddConst(id,"K0_38001e",0x38001e);
        cid = AddConst(id,"K0_380020",0x380020);
        cid = AddConst(id,"K0_380021",0x380021);
        cid = AddConst(id,"K0_380022",0x380022);
        cid = AddConst(id,"K0_380023",0x380023);
        cid = AddConst(id,"K0_38002e",0x38002e);
        cid = AddConst(id,"K0_38002f",0x38002f);
        cid = AddConst(id,"K0_380030",0x380030);
        cid = AddConst(id,"K0_380031",0x380031);
        cid = AddConst(id,"K0_380032",0x380032);
        cid = AddConst(id,"K0_380033",0x380033);
```

```
id = AddEnum(1,"VMM_MESSAGES",0x1100000);
cid = AddConst(id,"SYS_CRITICAL_INIT",          0);
cid = AddConst(id,"DEVICE_INIT",                0x1);
cid = AddConst(id,"INIT_COMPLETE",              0x2);
cid = AddConst(id,"SYS_VM_INIT",                0x3);
cid = AddConst(id,"SYS_VM_TERMINATE",           0x4);
cid = AddConst(id,"SYSTEM_EXIT",                0x5);
cid = AddConst(id,"SYS_CRTICAL_EXIT",           0x6);
cid = AddConst(id,"CREATE_VM",                  0x7);
cid = AddConst(id,"VM_CRITICAL_INIT",           0x8);
cid = AddConst(id,"VM_INIT",                    0x9);
cid = AddConst(id,"VM_TERMINATE",               0xa);
cid = AddConst(id,"VM_NOT_EXECUTABLE",          0xb);
cid = AddConst(id,"DESTROY_VM",                 0xc);
cid = AddConst(id,"VM_SUSPEND",                 0xd);
cid = AddConst(id,"VM_RESUME",                  0xe);
cid = AddConst(id,"SET_DEVICE_FOCUS",           0xf);
cid = AddConst(id,"BEGIN_MESSAGE_MODE",         0x10);
cid = AddConst(id,"END_MESSAGE_MODE",           0x11);
cid = AddConst(id,"REBOOT_PROCESOR",            0x12);
cid = AddConst(id,"QUERY_DESTROY",              0x13);
cid = AddConst(id,"DEBUG_QUERY",                0x14);
cid = AddConst(id,"BEGIN_PM_APP",               0x15);
cid = AddConst(id,"END_PM_APP",                 0x16);
cid = AddConst(id,"DEVICE_REBOOT_NOTIFY",       0x17);
cid = AddConst(id,"CRIT_REBOOT_NOTIFY",         0x18);
cid = AddConst(id,"CLOSE_VM_NOTIFY",            0x19);
cid = AddConst(id,"POWER_EVENT",                0x1a);
cid = AddConst(id,"SYS_DYNAMIC_DEVICE_INIT",    0x1b);
cid = AddConst(id,"SYS_DYNAMIC_DEVICE_EXIT",    0x1c);
cid = AddConst(id,"CREATE_THREAD",              0x1d);
cid = AddConst(id,"THREAD_INIT",                0x1e);
cid = AddConst(id,"TERMINATE_THREAD",           0x1f);
cid = AddConst(id,"THREAD_NOT_EXECUTABLE",      0x20);
cid = AddConst(id,"DESTROY_THREAD",             0x21);
cid = AddConst(id,"PNP_NEW_DEVNODE",            0x22);
cid = AddConst(id,"V32_DEVICEIOCONTROL",        0x23);

cid = AddConst(id,"Kernel32_Initialized",       0x31);
cid = AddConst(id,"Kernel32_Shoutdown",         0x32);


id = AddEnum(1,"error_code",0x1100000);
cid = AddConst(id,"ERR_INVALID_FUNCTION",       0x1);
cid = AddConst(id,"ERR_FILE_NOT_FOUND",         0x2);
cid = AddConst(id,"ERR_PATH_NOT_FOUND",         0x3);
cid = AddConst(id,"ERR_TOO_MANY_OPEN_FILES",    0x4);
cid = AddConst(id,"ERR_ACCESS_DENIED",          0x5);
cid = AddConst(id,"ERR_INVALID_HANDLE",         0x6);
cid = AddConst(id,"ERR_ARENA_TRASHED",          0x7);
cid = AddConst(id,"ERR_NOT_ENOUGH_MEMORY",      0x8);
cid = AddConst(id,"ERR_INVALID_BLOCK",          0x9);
cid = AddConst(id,"ERR_BAD_ENVIRONMENT",        0xa);
cid = AddConst(id,"ERR_BAD_FORMAT",             0xb);
cid = AddConst(id,"ERR_INVALID_ACCESS",         0xc);
cid = AddConst(id,"ERR_INVALID_DATA",           0xd);
cid = AddConst(id,"ERR_OUTOFMEMORY",            0xe);
cid = AddConst(id,"ERR_INVALID_DRIVE",          0xf);
```

```
        cid = AddConst(id,"ERR_CURRENT_DIRECTORY",      0x10);
        cid = AddConst(id,"ERR_NOT_SAME_DEVICE",        0x11);
        cid = AddConst(id,"ERR_NO_MORE_FILES",          0x12);
        cid = AddConst(id,"ERR_WRITE_PROTECT",          0x13);
        cid = AddConst(id,"ERR_BAD_UNIT",               0x14);
        cid = AddConst(id,"ERR_NOT_READY",              0x15);
        cid = AddConst(id,"ERR_BAD_COMMAND",            0x16);
        cid = AddConst(id,"ERR_CRC",                    0x17);
        cid = AddConst(id,"ERR_BAD_LENGTH",             0x18);
        cid = AddConst(id,"ERR_SEEK",                   0x19);
        cid = AddConst(id,"ERR_NOT_DOS_DISK",           0x1a);
        cid = AddConst(id,"ERR_SECTOR_NOT_FOUND",       0x1b);
        cid = AddConst(id,"ERR_OUT_OF_PAPER",           0x1c);
        cid = AddConst(id,"ERR_WRITE_FAULT",            0x1d);
        cid = AddConst(id,"ERR_READ_FAULT",             0x1e);
        cid = AddConst(id,"ERR_GEN_FAILURE",            0x1f);
        cid = AddConst(id,"ERR_SHARING_VIOLATION",      0x20);
        cid = AddConst(id,"ERR_LOCK_VIOLATION",         0x21);
        cid = AddConst(id,"ERR_WRONG_DISK",             0x22);
        cid = AddConst(id,"ERR_SHARING_BUFFER_EXCEEDED",0x24);
        cid = AddConst(id,"ERR_HANDLE_EOF",             0x26);
        cid = AddConst(id,"ERR_HANDLE_DISK_FULL",       0x27);
        cid = AddConst(id,"ERR_NOT_SUPPORTED",          0x32);
        cid = AddConst(id,"ERR_REM_NOT_LIST",           0x33);
        cid = AddConst(id,"ERR_DUP_NAME",               0x34);
        cid = AddConst(id,"ERR_BAD_NETPATH",            0x35);
        cid = AddConst(id,"ERR_NETWORK_BUSY",           0x36);
        cid = AddConst(id,"ERR_DEV_NOT_EXIST",          0x37);
        cid = AddConst(id,"ERR_TOO_MANY_CMDS",          0x38);
        cid = AddConst(id,"ERR_ADAP_HDW_ERR",           0x39);
        cid = AddConst(id,"ERR_BAD_NET_RESP",           0x3a);
        cid = AddConst(id,"ERR_UNEXP_NET_ERR",          0x3b);
        cid = AddConst(id,"ERR_BAD_REM_ADAP",           0x3c);
        cid = AddConst(id,"ERR_PRINTQ_FULL",            0x3d);
        cid = AddConst(id,"ERR_NO_SPOOL_SPACE",         0x3e);
        cid = AddConst(id,"ERR_PRINT_CANCELLED",        0x3f);
        cid = AddConst(id,"ERR_NETNAME_DELETED",        0x40);
        cid = AddConst(id,"ERR_NETWORK_ACCESS_DENIED",  0x41);
        cid = AddConst(id,"ERR_BAD_DEV_TYPE",           0x42);
        cid = AddConst(id,"ERR_BAD_NET_NAME",           0x43);
        cid = AddConst(id,"ERR_TOO_MANY_NAMES",         0x44);
        cid = AddConst(id,"ERR_TOO_MANY_SESS",          0x45);
        cid = AddConst(id,"ERR_SHARING_PAUSED",         0x46);
        cid = AddConst(id,"ERR_REQ_NOT_ACCEP",          0x47);
        cid = AddConst(id,"ERR_REDIR_PAUSED",           0x48);
        cid = AddConst(id,"ERR_FILE_EXISTS",            0x50);
        cid = AddConst(id,"ERR_CANNOT_MAKE",            0x52);
        cid = AddConst(id,"ERR_FAIL_I24",               0x53);
        cid = AddConst(id,"ERR_OUT_OF_STRUCTURES",      0x54);
        cid = AddConst(id,"ERR_ALREADY_ASSIGNED",       0x55);
        cid = AddConst(id,"ERR_INVALID_PASSWORD",       0x56);
        cid = AddConst(id,"ERR_INVALID_PARAMETER",      0x57);
        cid = AddConst(id,"ERR_NET_WRITE_FAULT",        0x58);
        cid = AddConst(id,"ERR_NO_PROC_SLOTS",          0x59);
        cid = AddConst(id,"ERR_TOO_MANY_SEMAPHORES",    0x64);
        cid = AddConst(id,"ERR_EXCL_SEM_ALREADY_OWNED", 0x65);
        cid = AddConst(id,"ERR_SEM_IS_SET",             0x66);
        cid = AddConst(id,"ERR_TOO_MANY_SEM_REQUESTS",  0x67);
```

```
        cid = AddConst(id,"ERR_INVALID_AT_INTERRUPT_TIME",0x68);
        cid = AddConst(id,"ERR_SEM_OWNER_DIED",         0x69);
        cid = AddConst(id,"ERR_SEM_USER_LIMIT",         0x6a);
        cid = AddConst(id,"ERR_DISK_CHANGE",            0x6b);
        cid = AddConst(id,"ERR_DRIVE_LOCKED",           0x6c);
        cid = AddConst(id,"ERR_BROKEN_PIPE",            0x6d);
        cid = AddConst(id,"ERR_OPEN_FAILED",            0x6e);
        cid = AddConst(id,"ERR_BUFFER_OVERFLOW",        0x6f);
        cid = AddConst(id,"ERR_DISK_FULL",              0x70);
        cid = AddConst(id,"ERR_NO_MORE_SEARCH_HANDLES", 0x71);
        cid = AddConst(id,"ERR_INVALID_TARGET_HANDLE",  0x72);
        cid = AddConst(id,"ERR_INVALID_CATEGORY",       0x75);
        cid = AddConst(id,"ERR_INVALID_VERIFY_SWITCH",  0x76);
        cid = AddConst(id,"ERR_BAD_DRIVER_LEVEL",       0x77);
        cid = AddConst(id,"ERR_CALL_NOT_IMPLEMENTED",   0x78);
        cid = AddConst(id,"ERR_SEM_TIMEOUT",            0x79);
        cid = AddConst(id,"ERR_INSUFFICIENT_BUFFER",    0x7a);
        cid = AddConst(id,"ERR_INVALID_NAME",           0x7b);
        cid = AddConst(id,"ERR_INVALID_LEVEL",          0x7c);
        cid = AddConst(id,"ERR_NO_VOLUME_LABEL",        0x7d);
        cid = AddConst(id,"ERR_MOD_NOT_FOUND",          0x7e);
        cid = AddConst(id,"ERR_PROC_NOT_FOUND",         0x7f);
        cid = AddConst(id,"ERR_WAIT_NO_CHILDREN",       0x80);
        cid = AddConst(id,"ERR_CHILD_NOT_COMPLETE",     0x81);
        cid = AddConst(id,"ERR_DIRECT_ACCESS_HANDLE",   0x82);
        cid = AddConst(id,"ERR_NEGATIVE_SEEK",          0x83);
        cid = AddConst(id,"ERR_SEEK_ON_DEVICE",         0x84);
        cid = AddConst(id,"ERR_IS_JOIN_TARGET",         0x85);
        cid = AddConst(id,"ERR_IS_JOINED",              0x86);
        cid = AddConst(id,"ERR_IS_SUBSTED",             0x87);
        cid = AddConst(id,"ERR_NOT_JOINED",             0x88);
        cid = AddConst(id,"ERR_NOT_SUBSTED",            0x89);
        cid = AddConst(id,"ERR_JOIN_TO_JOIN",           0x8a);
        cid = AddConst(id,"ERR_SUBST_TO_SUBST",         0x8b);
        cid = AddConst(id,"ERR_JOIN_TO_SUBST",          0x8c);
        cid = AddConst(id,"ERR_SUBST_TO_JOIN",          0x8d);
        cid = AddConst(id,"ERR_BUSY_DRIVE",             0x8e);
        cid = AddConst(id,"ERR_SAME_DRIVE",             0x8f);
        cid = AddConst(id,"ERR_DIR_NOT_ROOT",           0x90);
        cid = AddConst(id,"ERR_DIR_NOT_EMPTY",          0x91);
        cid = AddConst(id,"ERR_IS_SUBST_PATH",          0x92);
        cid = AddConst(id,"ERR_IS_JOIN_PATH",           0x93);
        cid = AddConst(id,"ERR_PATH_BUSY",              0x94);
        cid = AddConst(id,"ERR_IS_SUBST_TARGET",        0x95);
        cid = AddConst(id,"ERR_SYSTEM_TRACE",           0x96);
        cid = AddConst(id,"ERR_INVALID_EVENT_COUNT",    0x97);
        cid = AddConst(id,"ERR_TOO_MANY_MUXWAITERS",    0x98);
        cid = AddConst(id,"ERR_INVALID_LIST_FORMAT",    0x99);
        cid = AddConst(id,"ERR_LABEL_TOO_LONG",         0x9a);
        cid = AddConst(id,"ERR_TOO_MANY_TCBS",          0x9b);
        cid = AddConst(id,"ERR_SIGNAL_REFUSED",         0x9c);
        cid = AddConst(id,"ERR_DISCARDED",              0x9d);
        cid = AddConst(id,"ERR_NOT_LOCKED",             0x9e);
        cid = AddConst(id,"ERR_BAD_THREADID_ADDR",      0x9f);
        cid = AddConst(id,"ERR_BAD_ARGUMENTS",          0xa0);
        cid = AddConst(id,"ERR_BAD_PATHNAME",           0xa1);
        cid = AddConst(id,"ERR_SIGNAL_PENDING",         0xa2);
        cid = AddConst(id,"ERR_MAX_THRDS_REACHED",      0xa4);
```

```
        cid = AddConst(id,"ERR_LOCK_FAILED",            0xa7);
        cid = AddConst(id,"ERR_BUSY",                   0xaa);
        cid = AddConst(id,"ERR_CANCEL_VIOLATION",       0xad);
        cid = AddConst(id,"ERR_ATOMIC_LOCKS_NOT_SUPPORTED",0xae);
        cid = AddConst(id,"ERR_INVALID_SEGMENT_NUMBER", 0xb4);
        cid = AddConst(id,"ERR_INVALID_ORDINAL",        0xb6);
        cid = AddConst(id,"ERR_ALREADY_EXISTS",         0xb7);
        cid = AddConst(id,"ERR_INVALID_FLAG_NUMBER",    0xba);
        cid = AddConst(id,"ERR_SEM_NOT_FOUND",          0xbb);
        cid = AddConst(id,"ERR_INVALID_STARTING_CODESEG",0xbc);
        cid = AddConst(id,"ERR_INVALID_STACKSEG",       0xbd);
        cid = AddConst(id,"ERR_INVALID_MODULETYPE",     0xbe);
        cid = AddConst(id,"ERR_INVALID_EXE_SIGNATURE",  0xbf);
        cid = AddConst(id,"ERR_EXE_MARKED_INVALID",     0xc0);
        cid = AddConst(id,"ERR_BAD_EXE_FORMAT",         0xc1);
        cid = AddConst(id,"ERR_ITERATED_DATA_EXCEEDS_64k",0xc2);
        cid = AddConst(id,"ERR_INVALID_MINALLOCSIZE",   0xc3);
        cid = AddConst(id,"ERR_DYNLINK_FROM_INVALID_RING",0xc4);
        cid = AddConst(id,"ERR_IOPL_NOT_ENABLED",       0xc5);
        cid = AddConst(id,"ERR_INVALID_SEGDPL",         0xc6);
        cid = AddConst(id,"ERR_AUTODATASEG_EXCEEDS_64k",0xc7);
        cid = AddConst(id,"ERR_RING2SEG_MUST_BE_MOVABLE",0xc8);
        cid = AddConst(id,"ERR_RELOC_CHAIN_XEEDS_SEGLIM",0xc9);
        cid = AddConst(id,"ERR_INFLOOP_IN_RELOC_CHAIN", 0xca);
        cid = AddConst(id,"ERR_ENVVAR_NOT_FOUND",       0xcb);
        cid = AddConst(id,"ERR_NO_SIGNAL_SENT",         0xcd);
        cid = AddConst(id,"ERR_FILENAME_EXCED_RANGE",   0xce);
        cid = AddConst(id,"ERR_RING2_STACK_IN_USE",     0xcf);
        cid = AddConst(id,"ERR_META_EXPANSION_TOO_LONG",0xd0);
        cid = AddConst(id,"ERR_INVALID_SIGNAL_NUMBER",  0xd1);
        cid = AddConst(id,"ERR_THREAD_1_INACTIVE",      0xd2);
        cid = AddConst(id,"ERR_LOCKED",                 0xd4);
        cid = AddConst(id,"ERR_TOO_MANY_MODULES",       0xd6);
        cid = AddConst(id,"ERR_NESTING_NOT_ALLOWED",    0xd7);
        cid = AddConst(id,"ERR_BAD_PIPE",               0xe6);
        cid = AddConst(id,"ERR_PIPE_BUSY",              0xe7);
        cid = AddConst(id,"ERR_NO_DATA",                0xe8);
        cid = AddConst(id,"ERR_PIPE_NOT_CONNECTED",     0xe9);
        cid = AddConst(id,"ERR_MORE_DATA",              0xea);
        cid = AddConst(id,"ERR_VC_DISCONNECTED",        0xf0);
        cid = AddConst(id,"ERR_INVALID_EA_NAME",        0xfe);
        cid = AddConst(id,"ERR_EA_LIST_INCONSISTENT",   0xff);
        cid = AddConst(id,"ERR_NO_MORE_ITEMS",          0x103);
        cid = AddConst(id,"ERR_CANNOT_COPY",            0x10a);
        cid = AddConst(id,"ERR_DIRECTORY",              0x10b);
        cid = AddConst(id,"ERR_EAS_DIDNT_FIT",          0x113);
        cid = AddConst(id,"ERR_EA_FILE_CORRUPT",        0x114);
        cid = AddConst(id,"ERR_EA_TABLE_FULL",  0x115);
        cid = AddConst(id,"ERR_INVALID_EA_HANDLE",      0x116);
        cid = AddConst(id,"ERR_EAS_NOT_SUPPORTED",      0x11a);
        cid = AddConst(id,"ERR_NOT_OWNER",              0x120);
        cid = AddConst(id,"ERR_TOO_MANY_POSTS",         0x12a);
        cid = AddConst(id,"ERR_MR_MID_NOT_FOUND",       0x13d);
        cid = AddConst(id,"ERR_INVALID_ADDRESS",        0x1e7);
        cid = AddConst(id,"ERR_ARITHMETIC_OVERFLOW",    0x216);
        cid = AddConst(id,"ERR_PIPE_CONNECTED",         0x217);
        cid = AddConst(id,"ERR_PIPE_LISTENING",         0x218);
        cid = AddConst(id,"ERR_EA_ACCESS_DENIED",       0x3e2);
```

```
        cid = AddConst(id,"ERR_OPERATION_ABORTED",      0x3e3);
        cid = AddConst(id,"ERR_IO_INCOMPLETE",          0x3e4);
        cid = AddConst(id,"ERR_IO_PENDING",             0x3e5);
        cid = AddConst(id,"ERR_NOACCESS",               0x3e6);
        cid = AddConst(id,"ERR_SWAPERROR",              0x3e7);
        cid = AddConst(id,"ERR_STACK_OVERFLOW",         0x3e9);
        cid = AddConst(id,"ERR_INVALID_MESSAGE",        0x3ea);
        cid = AddConst(id,"ERR_CAN_NOT_COMPLETE",       0x3eb);
        cid = AddConst(id,"ERR_INVALID_FLAGS",          0x3ec);
        cid = AddConst(id,"ERR_UNRECOGNIZED_VOLUME",    0x3ed);
        cid = AddConst(id,"ERR_FILE_INVALID",           0x3ee);
        cid = AddConst(id,"ERR_FULLSCREEN_MODE",        0x3ef);
        cid = AddConst(id,"ERR_NO_TOKEN",               0x3f0);
        cid = AddConst(id,"ERR_BADDB",                  0x3f1);
        cid = AddConst(id,"ERR_BADKEY",                 0x3f2);
        cid = AddConst(id,"ERR_CANTOPEN",               0x3f3);
        cid = AddConst(id,"ERR_CANTREAD",               0x3f4);
        cid = AddConst(id,"ERR_CANTWRITE",              0x3f5);
        cid = AddConst(id,"ERR_REGISTRY_RECOVERED",     0x3f6);
        cid = AddConst(id,"ERR_REGISTRY_CORRUPT",       0x3f7);
        cid = AddConst(id,"ERR_REGISTRY_IO_FAILED",     0x3f8);
        cid = AddConst(id,"ERR_NOT_REGISTRY_FILE",      0x3f9);
        cid = AddConst(id,"ERR_KEY_DELETED",            0x3fa);
        cid = AddConst(id,"ERR_NO_LOG_SPACE",           0x3fb);
        cid = AddConst(id,"ERR_KEY_HAS_CHILDREN",       0x3fc);
        cid = AddConst(id,"ERR_CHILD_MUST_BE_VOLATILE", 0x3fd);
        cid = AddConst(id,"ERR_NOTIFY_ENUM_DIR",        0x3fe);
        cid = AddConst(id,"ERR_DEPENDENT_SERVICES_RUNNING",0x41b);
        cid = AddConst(id,"ERR_INVALID_SERVICE_CONTROL",0x41c);
        cid = AddConst(id,"ERR_SERVICE_REQUEST_TIMEOUT",0x41d);
        cid = AddConst(id,"ERR_SERVICE_NO_THREAD",      0x41e);
        cid = AddConst(id,"ERR_SERVICE_DATABASE_LOCKED",0x41f);
        cid = AddConst(id,"ERR_SERVICE_ALREADY_RUNNING",0x420);
        cid = AddConst(id,"ERR_INVALID_SERVICE_ACCOUNT",0x421);
        cid = AddConst(id,"ERR_SERVICE_DISABLED",       0x422);
        cid = AddConst(id,"ERR_CIRCULAR_DEPENDENCY",    0x423);
        cid = AddConst(id,"ERR_SERVICE_DOES_NOT_EXIST", 0x424);
        cid = AddConst(id,"ERR_SERVICE_CANNOT_ACCEPT_CTRL",0x425);
        cid = AddConst(id,"ERR_SERVICE_NOT_ACTIVE",     0x426);
        cid = AddConst(id,"ERR_FAILED_SERVICE_CONTROLLER_CONNECT",0x427);
        cid = AddConst(id,"ERR_EXCEPTION_IN_SERVICE",   0x428);
        cid = AddConst(id,"ERR_DATABASE_DOES_NOT_EXIST",0x429);
        cid = AddConst(id,"ERR_SERVICE_SPECIFIC_ERROR", 0x42a);
        cid = AddConst(id,"ERR_PROCESS_ABORTED",        0x42b);
        cid = AddConst(id,"ERR_SERVICE_DEPENDENCY_FAIL",0x42c);
        cid = AddConst(id,"ERR_SERVICE_LOGON_FAILED",   0x42d);
        cid = AddConst(id,"ERR_SERVICE_START_HANG",     0x42e);
        cid = AddConst(id,"ERR_INVALID_SERVICE_LOCK",   0x42f);
        cid = AddConst(id,"ERR_SERVICE_MARKED_FOR_DELETE",0x430);
        cid = AddConst(id,"ERR_SERVICE_EXISTS",         0x431);
        cid = AddConst(id,"ERR_ALREADY_RUNNING_LKG",    0x432);
        cid = AddConst(id,"ERR_SERVICE_DEPENDENCY_DELETED",0x433);
        cid = AddConst(id,"ERR_BOOT_ALREADY_ACCEPTED",  0x434);
        cid = AddConst(id,"ERR_SERVICE_NEVER_STARTED",  0x435);
        cid = AddConst(id,"ERR_DUPLICATE_SERVICE_NAME", 0x436);
        cid = AddConst(id,"ERR_END_OF_MEDIA",           0x44c);
        cid = AddConst(id,"ERR_FILEMARK_DETECTED",      0x44d);
        cid = AddConst(id,"ERR_BEGINNING_OF_MEDIA",     0x44e);
```

```
        cid = AddConst(id,"ERR_SETMARK_DETECTED",       0x44f);
        cid = AddConst(id,"ERR_NO_DATA_DETECTED",       0x450);
        cid = AddConst(id,"ERR_PARTITION_FAILURE",      0x451);
        cid = AddConst(id,"ERR_INVALID_BLOCK_LENGTH",   0x452);
        cid = AddConst(id,"ERR_DEVICE_NOT_PARTITIONED", 0x453);
        cid = AddConst(id,"ERR_UNABLE_TO_LOCK_MEDIA",   0x454);
        cid = AddConst(id,"ERR_UNABLE_TO_UNLOAD_MEDIA", 0x455);
        cid = AddConst(id,"ERR_MEDIA_CHANGED",          0x456);
        cid = AddConst(id,"ERR_BUS_RESET",              0x457);
        cid = AddConst(id,"ERR_NO_MEDIA_IN_DRIVE",      0x458);
        cid = AddConst(id,"ERR_NO_UNICODE_TRANSLATION", 0x459);
        cid = AddConst(id,"ERR_DLL_INIT_FAILED",        0x45a);
        cid = AddConst(id,"ERR_SHUTDOWN_IN_PROGRESS",   0x45b);
        cid = AddConst(id,"ERR_NO_SHUTDOWN_IN_PROGRESS",0x45c);
        cid = AddConst(id,"ERR_IO_DEVICE",              0x45d);
        cid = AddConst(id,"ERR_SERIAL_NO_DEVICE",       0x45e);
        cid = AddConst(id,"ERR_IRQ_BUSY",               0x45f);
        cid = AddConst(id,"ERR_MORE_WRITES",            0x460);
        cid = AddConst(id,"ERR_COUNTER_TIMEOUT",        0x461);
        cid = AddConst(id,"ERR_FLOPPY_ID_MARK_NOT_FOUND",0x462);
        cid = AddConst(id,"ERR_FLOPPY_WRONG_CYLINDER",  0x463);
        cid = AddConst(id,"ERR_FLOPPY_UNKNOWN_ERROR",   0x464);
        cid = AddConst(id,"ERR_FLOPPY_BAD_REGISTERS",   0x465);
        cid = AddConst(id,"ERR_DISK_RECALIBRATE_FAILED",0x466);
        cid = AddConst(id,"ERR_DISK_OPERATION_FAILED",  0x467);
        cid = AddConst(id,"ERR_DISK_RESET_FAILED",      0x468);
        cid = AddConst(id,"ERR_EOM_OVERFLOW",           0x469);
        cid = AddConst(id,"ERR_NOT_ENOUGH_SERVER_MEMORY",0x46a);
        cid = AddConst(id,"ERR_POSSIBLE_DEADLOCK",      0x46b);
        cid = AddConst(id,"ERR_BAD_DEVICE",             0x4b0);
        cid = AddConst(id,"ERR_CONNECTION_UNAVAIL",     0x4b1);
        cid = AddConst(id,"ERR_DEVICE_ALREADY_REMEMBERED",0x4b2);
        cid = AddConst(id,"ERR_NO_NET_OR_BAD_PATH",     0x4b3);
        cid = AddConst(id,"ERR_BAD_PROVIDER",           0x4b4);
        cid = AddConst(id,"ERR_CANNOT_OPEN_PROFILE",    0x4b5);
        cid = AddConst(id,"ERR_BAD_PROFILE",            0x4b6);
        cid = AddConst(id,"ERR_NOT_CONTAINER",          0x4b7);
        cid = AddConst(id,"ERR_EXTENDED_ERROR",         0x4b8);
        cid = AddConst(id,"ERR_INVALID_GROUPNAME",      0x4b9);
        cid = AddConst(id,"ERR_INVALID_COMPUTERNAME",   0x4ba);
        cid = AddConst(id,"ERR_INVALID_EVENTNAME",      0x4bb);
        cid = AddConst(id,"ERR_INVALID_DOMAINNAME",     0x4bc);
        cid = AddConst(id,"ERR_INVALID_SERVICENAME",    0x4bd);
        cid = AddConst(id,"ERR_INVALID_NETNAME",        0x4be);
        cid = AddConst(id,"ERR_INVALID_SHARENAME",      0x4bf);
        cid = AddConst(id,"ERR_INVALID_PASSWORDNAME",   0x4c0);
        cid = AddConst(id,"ERR_INVALID_MESSAGENAME",    0x4c1);
        cid = AddConst(id,"ERR_INVALID_MESSAGEDEST",    0x4c2);
        cid = AddConst(id,"ERR_SESSION_CREDENTIAL_CONFLICT",0x4c3);
        cid = AddConst(id,"ERR_REMOTE_SESSION_LIMIT_EXCEEDED",0x4c4);
        cid = AddConst(id,"ERR_DUP_DOMAINNAME",         0x4c5);
        cid = AddConst(id,"ERR_NOT_ALL_ASSIGNED",       0x514);
        cid = AddConst(id,"ERR_SOME_NOT_MAPPED",        0x515);
        cid = AddConst(id,"ERR_NO_QUOTAS_FOR_ACCOUNT",  0x516);
        cid = AddConst(id,"ERR_LOCAL_USER_SESSION_KEY", 0x517);
        cid = AddConst(id,"ERR_NULL_LM_PASSWORD",       0x518);
        cid = AddConst(id,"ERR_UNKNOWN_REVISION",       0x519);
        cid = AddConst(id,"ERR_REVISION_MISMATCH",      0x51a);
```

```
        cid = AddConst(id,"ERR_INVALID_OWNER",          0x51b);
        cid = AddConst(id,"ERR_INVALID_PRIMARY_GROUP",  0x51c);
        cid = AddConst(id,"ERR_NO_IMPERSONATION_TOKEN", 0x51d);
        cid = AddConst(id,"ERR_CANT_DISABLE_MANDATORY", 0x51e);
        cid = AddConst(id,"ERR_NO_LOGON_SERVERS",       0x51f);
        cid = AddConst(id,"ERR_NO_SUCH_LOGON_SESSION",  0x520);
        cid = AddConst(id,"ERR_NO_SUCH_PRIVILEGE",      0x521);
        cid = AddConst(id,"ERR_PRIVILEGE_NOT_HELD",     0x522);
        cid = AddConst(id,"ERR_INVALID_ACCOUNT_NAME",   0x523);
        cid = AddConst(id,"ERR_USER_EXISTS",            0x524);
        cid = AddConst(id,"ERR_NO_SUCH_USER",           0x525);
        cid = AddConst(id,"ERR_GROUP_EXISTS",           0x526);
        cid = AddConst(id,"ERR_NO_SUCH_GROUP",          0x527);
        cid = AddConst(id,"ERR_MEMBER_IN_GROUP",        0x528);
        cid = AddConst(id,"ERR_MEMBER_NOT_IN_GROUP",    0x529);
        cid = AddConst(id,"ERR_LAST_ADMIN",             0x52a);
        cid = AddConst(id,"ERR_WRONG_PASSWORD",         0x52b);
        cid = AddConst(id,"ERR_ILL_FORMED_PASSWORD",    0x52c);
        cid = AddConst(id,"ERR_PASSWORD_RESTRICTION",   0x52d);
        cid = AddConst(id,"ERR_LOGON_FAILURE",          0x52e);
        cid = AddConst(id,"ERR_ACCOUNT_RESTRICTION",    0x52f);
        cid = AddConst(id,"ERR_INVALID_LOGON_HOURS",    0x530);
        cid = AddConst(id,"ERR_INVALID_WORKSTATION",    0x531);
        cid = AddConst(id,"ERR_PASSWORD_EXPIRED",       0x532);
        cid = AddConst(id,"ERR_ACCOUNT_DISABLED",       0x533);
        cid = AddConst(id,"ERR_NONE_MAPPED",            0x534);
        cid = AddConst(id,"ERR_TOO_MANY_LUIDS_REQUESTED",0x535);
        cid = AddConst(id,"ERR_LUIDS_EXHAUSTED",        0x536);
        cid = AddConst(id,"ERR_INVALID_SUB_AUTHORITY",  0x537);
        cid = AddConst(id,"ERR_INVALID_ACL",            0x538);
        cid = AddConst(id,"ERR_INVALID_SID",            0x539);
        cid = AddConst(id,"ERR_INVALID_SECURITY_DESCR", 0x53a);
        cid = AddConst(id,"ERR_BAD_INHERITANCE_ACL",    0x53c);
        cid = AddConst(id,"ERR_SERVER_DISABLED",        0x53d);
        cid = AddConst(id,"ERR_SERVER_NOT_DISABLED",    0x53e);
        cid = AddConst(id,"ERR_INVALID_ID_AUTHORITY",   0x53f);
        cid = AddConst(id,"ERR_ALLOTTED_SPACE_EXCEEDED",0x540);
        cid = AddConst(id,"ERR_INVALID_GROUP_ATTRIBUTES",0x541);
        cid = AddConst(id,"ERR_BAD_IMPERSONATION_LEVEL",0x542);
        cid = AddConst(id,"ERR_CANT_OPEN_ANONYMOUS",    0x543);
        cid = AddConst(id,"ERR_BAD_VALIDATION_CLASS",   0x544);
        cid = AddConst(id,"ERR_BAD_TOKEN_TYPE",         0x545);
        cid = AddConst(id,"ERR_NO_SECURITY_ON_OBJECT",  0x546);
        cid = AddConst(id,"ERR_CANT_ACCESS_DOMAIN_INFO",0x547);
        cid = AddConst(id,"ERR_INVALID_SERVER_STATE",   0x548);
        cid = AddConst(id,"ERR_INVALID_DOMAIN_STATE",   0x549);
        cid = AddConst(id,"ERR_INVALID_DOMAIN_ROLE",    0x54a);
        cid = AddConst(id,"ERR_NO_SUCH_DOMAIN",         0x54b);
        cid = AddConst(id,"ERR_DOMAIN_EXISTS",          0x54c);
        cid = AddConst(id,"ERR_DOMAIN_LIMIT_EXCEEDED",  0x54d);
        cid = AddConst(id,"ERR_INTERNAL_DB_CORRUPTION", 0x54e);
        cid = AddConst(id,"ERR_INTERNAL_ERROR",         0x54f);
        cid = AddConst(id,"ERR_GENERIC_NOT_MAPPED",     0x550);
        cid = AddConst(id,"ERR_BAD_DESCRIPTOR_FORMAT",  0x551);
        cid = AddConst(id,"ERR_NOT_LOGON_PROCESS",      0x552);
        cid = AddConst(id,"ERR_LOGON_SESSION_EXISTS",   0x553);
        cid = AddConst(id,"ERR_NO_SUCH_PACKAGE",        0x554);
        cid = AddConst(id,"ERR_BAD_LOGON_SESSION_STATE",0x555);
```

```
        cid = AddConst(id,"ERR_LOGON_SESSION_COLLISION",0x556);
        cid = AddConst(id,"ERR_INVALID_LOGON_TYPE",     0x557);
        cid = AddConst(id,"ERR_CANNOT_IMPERSONATE",     0x558);
        cid = AddConst(id,"ERR_RXACT_INVALID_STATE",    0x559);
        cid = AddConst(id,"ERR_RXACT_COMMIT_FAILURE",   0x55a);
        cid = AddConst(id,"ERR_SPECIAL_ACCOUNT",        0x55b);
        cid = AddConst(id,"ERR_SPECIAL_GROUP",          0x55c);
        cid = AddConst(id,"ERR_SPECIAL_USER",           0x55d);
        cid = AddConst(id,"ERR_MEMBERS_PRIMARY_GROUP",  0x55e);
        cid = AddConst(id,"ERR_TOKEN_ALREADY_IN_USE",   0x55f);
        cid = AddConst(id,"ERR_NO_SUCH_ALIAS",          0x560);
        cid = AddConst(id,"ERR_MEMBER_NOT_IN_ALIAS",    0x561);
        cid = AddConst(id,"ERR_MEMBER_IN_ALIAS",        0x562);
        cid = AddConst(id,"ERR_ALIAS_EXISTS",           0x563);
        cid = AddConst(id,"ERR_LOGON_NOT_GRANTED",      0x564);
        cid = AddConst(id,"ERR_TOO_MANY_SECRETS",       0x565);
        cid = AddConst(id,"ERR_SECRET_TOO_LONG",        0x566);
        cid = AddConst(id,"ERR_INTERNAL_DB_ERROR",      0x567);
        cid = AddConst(id,"ERR_TOO_MANY_CONTEXT_IDS",   0x568);
        cid = AddConst(id,"ERR_LOGON_TYPE_NOT_GRANTED", 0x569);
        cid = AddConst(id,"ERR_NT_CROSS_ENCRYPTION_REQUIRED",0x56a);
        cid = AddConst(id,"ERR_NO_SUCH_MEMBER",         0x56b);
        cid = AddConst(id,"ERR_INVALID_MEMBER",         0x56c);
        cid = AddConst(id,"ERR_TOO_MANY_SIDS",          0x56d);
        cid = AddConst(id,"ERR_LM_CROSS_ENCRYPTION_REQUIRED",0x56e);
        cid = AddConst(id,"ERR_NO_INHERITANCE",         0x56f);
        cid = AddConst(id,"ERR_FILE_CORRUPT",           0x570);
        cid = AddConst(id,"ERR_DISK_CORRUPT",           0x571);
        cid = AddConst(id,"ERR_NO_USER_SESSION_KEY",    0x572);
        cid = AddConst(id,"ERR_INVALID_WINDOW_HANDLE",  0x578);
        cid = AddConst(id,"ERR_INVALID_MENU_HANDLE",    0x579);
        cid = AddConst(id,"ERR_INVALID_CURSOR_HANDLE",  0x57a);
        cid = AddConst(id,"ERR_INVALID_ACCEL_HANDLE",   0x57b);
        cid = AddConst(id,"ERR_INVALID_HOOK_HANDLE",    0x57c);
        cid = AddConst(id,"ERR_INVALID_DWP_HANDLE",     0x57d);
        cid = AddConst(id,"ERR_TLW_WITH_WSCHILD",       0x57e);
        cid = AddConst(id,"ERR_CANNOT_FIND_WND_CLASS",  0x57f);
        cid = AddConst(id,"ERR_WINDOW_OF_OTHER_THREAD", 0x580);
        cid = AddConst(id,"ERR_HOTKEY_ALREADY_REGISTERED",      0x581);
        cid = AddConst(id,"ERR_CLASS_ALREADY_EXISTS",   0x582);
        cid = AddConst(id,"ERR_CLASS_DOES_NOT_EXIST",   0x583);
        cid = AddConst(id,"ERR_CLASS_HAS_WINDOWS",      0x584);
        cid = AddConst(id,"ERR_INVALID_INDEX",          0x585);
        cid = AddConst(id,"ERR_INVALID_ICON_HANDLE",    0x586);
        cid = AddConst(id,"ERR_PRIVATE_DIALOG_INDEX",   0x587);
        cid = AddConst(id,"ERR_LISTBOX_ID_NOT_FOUND",   0x588);
        cid = AddConst(id,"ERR_NO_WILDCARD_CHARACTERS", 0x589);
        cid = AddConst(id,"ERR_CLIPBOARD_NOT_OPEN",     0x58a);
        cid = AddConst(id,"ERR_HOTKEY_NOT_REGISTERED",  0x58b);
        cid = AddConst(id,"ERR_WINDOW_NOT_DIALOG",      0x58c);
        cid = AddConst(id,"ERR_CONTROL_ID_NOT_FOUND",   0x58d);
        cid = AddConst(id,"ERR_INVALID_COMBOBOX_MESSAGE",       0x58e);
        cid = AddConst(id,"ERR_WINDOW_NOT_COMBOBOX",    0x58f);
        cid = AddConst(id,"ERR_INVALID_EDIT_HEIGHT",    0x590);
        cid = AddConst(id,"ERR_DC_NOT_FOUND",           0x591);
        cid = AddConst(id,"ERR_INVALID_HOOK_FILTER",    0x592);
        cid = AddConst(id,"ERR_INVALID_FILTER_PROC",    0x593);
        cid = AddConst(id,"ERR_HOOK_NEEDS_HMOD",        0x594);
```

```
        cid = AddConst(id,"ERR_GLOBAL_ONLY_HOOK",       0x595);
        cid = AddConst(id,"ERR_JOURNAL_HOOK_SET",       0x596);
        cid = AddConst(id,"ERR_HOOK_NOT_INSTALLED",     0x597);
        cid = AddConst(id,"ERR_INVALID_LB_MESSAGE",     0x598);
        cid = AddConst(id,"ERR_SETCOUNT_ON_BAD_LB",     0x599);
        cid = AddConst(id,"ERR_LB_WITHOUT_TABSTOPS",    0x59a);
        cid = AddConst(id,"ERR_DESTROY_OBJECT_OF_OTHER_THREAD",0x59b);
        cid = AddConst(id,"ERR_CHILD_WINDOW_MENU",      0x59c);
        cid = AddConst(id,"ERR_NO_SYSTEM_MENU",         0x59d);
        cid = AddConst(id,"ERR_INVALID_MSGBOX_STYLE",   0x59e);
        cid = AddConst(id,"ERR_INVALID_SPI_VALUE",      0x59f);
        cid = AddConst(id,"ERR_SCREEN_ALREADY_LOCKED",  0x5a0);
        cid = AddConst(id,"ERR_HWNDS_HAVE_DIFFERENT_PARENT",    0x5a1);
        cid = AddConst(id,"ERR_NOT_CHILD_WINDOW",       0x5a2);
        cid = AddConst(id,"ERR_INVALID_GW_COMMAND",     0x5a3);
        cid = AddConst(id,"ERR_INVALID_THREAD_ID",      0x5a4);
        cid = AddConst(id,"ERR_NON_MDICHILD_WINDOW",    0x5a5);
        cid = AddConst(id,"ERR_POPUP_ALREADY_ACTIVE",   0x5a6);
        cid = AddConst(id,"ERR_NO_SCROLLBARS",          0x5a7);
        cid = AddConst(id,"ERR_INVALID_SCROLLBAR_RANGE",        0x5a8);
        cid = AddConst(id,"ERR_INVALID_SHOWWIN_COMMAND",        0x5a9);
        cid = AddConst(id,"ERR_EVENTLOG_FILE_CORRUPT",  0x5dc);
        cid = AddConst(id,"ERR_EVENTLOG_CANT_START",    0x5dd);
        cid = AddConst(id,"ERR_LOG_FILE_FULL",          0x5de);
        cid = AddConst(id,"ERR_EVENTLOG_FILE_CHANGED",  0x5df);
        cid = AddConst(id,"RPC_S_INVALID_STRING_BINDING",       0x6a4);
        cid = AddConst(id,"RPC_S_WRONG_KIND_OF_BINDING",        0x6a5);
        cid = AddConst(id,"RPC_S_INVALID_BINDING",      0x6a6);
        cid = AddConst(id,"RPC_S_PROTSEQ_NOT_SUPPORTED",        0x6a7);
        cid = AddConst(id,"RPC_S_INVALID_RPC_PROTSEQ",  0x6a8);
        cid = AddConst(id,"RPC_S_INVALID_STRING_UUID",  0x6a9);
        cid = AddConst(id,"RPC_S_INVALID_ENDPOINT_FORMAT",      0x6aa);
        cid = AddConst(id,"RPC_S_INVALID_NET_ADDR",     0x6ab);
        cid = AddConst(id,"RPC_S_NO_ENDPOINT_FOUND",    0x6ac);
        cid = AddConst(id,"RPC_S_INVALID_TIMEOUT",      0x6ad);
        cid = AddConst(id,"RPC_S_OBJECT_NOT_FOUND",     0x6ae);
        cid = AddConst(id,"RPC_S_ALREADY_REGISTERED",   0x6af);
        cid = AddConst(id,"RPC_S_TYPE_ALREADY_REGISTERED",      0x6b0);
        cid = AddConst(id,"RPC_S_ALREADY_LISTENING",    0x6b1);
        cid = AddConst(id,"RPC_S_NO_PROTSEQS_REGISTERED",       0x6b2);
        cid = AddConst(id,"RPC_S_NOT_LISTENING",        0x6b3);
        cid = AddConst(id,"RPC_S_UNKNOWN_MGR_TYPE",     0x6b4);
        cid = AddConst(id,"RPC_S_UNKNOWN_IF",           0x6b5);
        cid = AddConst(id,"RPC_S_NO_BINDINGS",          0x6b6);
        cid = AddConst(id,"RPC_S_NO_PROTSEQS",          0x6b7);
        cid = AddConst(id,"RPC_S_CANT_CREATE_ENDPOINT", 0x6b8);
        cid = AddConst(id,"RPC_S_OUT_OF_RESOURCES",     0x6b9);
        cid = AddConst(id,"RPC_S_SERVER_UNAVAILABLE",   0x6ba);
        cid = AddConst(id,"RPC_S_SERVER_TOO_BUSY",      0x6bb);
        cid = AddConst(id,"RPC_S_INVALID_NETWORK_OPTIONS",      0x6bc);
        cid = AddConst(id,"RPC_S_NO_CALL_ACTIVE",       0x6bd);
        cid = AddConst(id,"RPC_S_CALL_FAILED",          0x6be);
        cid = AddConst(id,"RPC_S_CALL_FAILED_DNE",      0x6bf);
        cid = AddConst(id,"RPC_S_PROTOCOL_ERROR",       0x6c0);
        cid = AddConst(id,"RPC_S_UNSUPPORTED_TRANS_SYN",        0x6c2);
        cid = AddConst(id,"RPC_S_SERVER_OUT_OF_MEMORY", 0x6c3);
        cid = AddConst(id,"RPC_S_UNSUPPORTED_TYPE",     0x6c4);
        cid = AddConst(id,"RPC_S_INVALID_TAG",          0x6c5);
```

```
        cid = AddConst(id,"RPC_S_INVALID_BOUND",        0x6c6);
        cid = AddConst(id,"RPC_S_NO_ENTRY_NAME",        0x6c7);
        cid = AddConst(id,"RPC_S_INVALID_NAME_SYNTAX",  0x6c8);
        cid = AddConst(id,"RPC_S_UNSUPPORTED_NAME_SYNTAX",      0x6c9);
        cid = AddConst(id,"RPC_S_UUID_NO_ADDRESS",      0x6cb);
        cid = AddConst(id,"RPC_S_DUPLICATE_ENDPOINT",   0x6cc);
        cid = AddConst(id,"RPC_S_UNKNOWN_AUTHN_TYPE",   0x6cd);
        cid = AddConst(id,"RPC_S_MAX_CALLS_TOO_SMALL",  0x6ce);
        cid = AddConst(id,"RPC_S_STRING_TOO_LONG",      0x6cf);
        cid = AddConst(id,"RPC_S_PROTSEQ_NOT_FOUND",    0x6d0);
        cid = AddConst(id,"RPC_S_PROCNUM_OUT_OF_RANGE", 0x6d1);
        cid = AddConst(id,"RPC_S_BINDING_HAS_NO_AUTH",  0x6d2);
        cid = AddConst(id,"RPC_S_UNKNOWN_AUTHN_SERVICE",        0x6d3);
        cid = AddConst(id,"RPC_S_UNKNOWN_AUTHN_LEVEL",  0x6d4);
        cid = AddConst(id,"RPC_S_INVALID_AUTH_IDENTITY",        0x6d5);
        cid = AddConst(id,"RPC_S_UNKNOWN_AUTHZ_SERVICE",        0x6d6);
        cid = AddConst(id,"EPT_S_INVALID_ENTRY",        0x6d7);
        cid = AddConst(id,"EPT_S_CANT_PERFORM_OP",      0x6d8);
        cid = AddConst(id,"EPT_S_NOT_REGISTERED",       0x6d9);
        cid = AddConst(id,"RPC_S_INCOMPLETE_NAME",      0x6db);
        cid = AddConst(id,"RPC_S_INVALID_VERS_OPTION",  0x6dc);
        cid = AddConst(id,"RPC_S_NO_MORE_MEMBERS",      0x6dd);
        cid = AddConst(id,"RPC_S_INTERFACE_NOT_FOUND",  0x6df);
        cid = AddConst(id,"RPC_S_ENTRY_ALREADY_EXISTS", 0x6e0);
        cid = AddConst(id,"RPC_S_ENTRY_NOT_FOUND",      0x6e1);
        cid = AddConst(id,"RPC_S_NAME_SERVICE_UNAVAILABLE",     0x6e2);
        cid = AddConst(id,"RPC_S_CANNOT_SUPPORT",       0x6e4);
        cid = AddConst(id,"RPC_S_NO_CONTEXT_AVAILABLE", 0x6e5);
        cid = AddConst(id,"RPC_S_INTERNAL_ERROR",       0x6e6);
        cid = AddConst(id,"RPC_S_ZERO_DIVIDE",          0x6e7);
        cid = AddConst(id,"RPC_S_ADDRESS_ERROR",        0x6e8);
        cid = AddConst(id,"RPC_S_FP_DIV_ZERO",          0x6e9);
        cid = AddConst(id,"RPC_S_FP_UNDERFLOW",         0x6ea);
        cid = AddConst(id,"RPC_S_FP_OVERFLOW",          0x6eb);
        cid = AddConst(id,"RPC_X_NO_MORE_ENTRIES",      0x6ec);
        cid = AddConst(id,"RPC_X_SS_CHAR_TRANS_OPEN_FAIL",      0x6ed);
        cid = AddConst(id,"RPC_X_SS_CHAR_TRANS_SHORT_FILE",     0x6ee);
        cid = AddConst(id,"RPC_X_SS_IN_NULL_CONTEXT",   0x6ef);
        cid = AddConst(id,"RPC_X_SS_CONTEXT_MISMATCH",  0x6f0);
        cid = AddConst(id,"RPC_X_SS_CONTEXT_DAMAGED",   0x6f1);
        cid = AddConst(id,"RPC_X_SS_HANDLES_MISMATCH",  0x6f2);
        cid = AddConst(id,"RPC_X_SS_CANNOT_GET_CALL_HANDLE",    0x6f3);
        cid = AddConst(id,"RPC_X_NULL_REF_POINTER",     0x6f4);
        cid = AddConst(id,"RPC_X_ENUM_VALUE_OUT_OF_RANGE",      0x6f5);
        cid = AddConst(id,"RPC_X_BYTE_COUNT_TOO_SMALL", 0x6f6);
        cid = AddConst(id,"RPC_X_BAD_STUB_DATA",        0x6f7);
        cid = AddConst(id,"ERR_INVALID_USER_BUFFER",    0x6f8);
        cid = AddConst(id,"ERR_UNRECOGNIZED_MEDIA",     0x6f9);
        cid = AddConst(id,"ERR_NO_TRUST_LSA_SECRET",    0x6fa);
        cid = AddConst(id,"ERR_NO_TRUST_SAM_ACCOUNT",   0x6fb);
        cid = AddConst(id,"ERR_TRUSTED_DOMAIN_FAILURE", 0x6fc);
        cid = AddConst(id,"ERR_TRUSTED_RELATIONSHIP_FAILURE",0x6fd);
        cid = AddConst(id,"ERR_TRUST_FAILURE",          0x6fe);
        cid = AddConst(id,"RPC_S_CALL_IN_PROGRESS",     0x6ff);
        cid = AddConst(id,"ERR_NETLOGON_NOT_STARTED",   0x700);
        cid = AddConst(id,"ERR_ACCOUNT_EXPIRED",        0x701);
        cid = AddConst(id,"ERR_REDIRECTOR_HAS_OPEN_HANDLES",0x702);
        cid = AddConst(id,"ERR_PRINTER_DRIVER_ALREADY_INSTALLED",0x703);
```

```
        cid = AddConst(id,"ERR_UNKNOWN_PORT",           0x704);
        cid = AddConst(id,"ERR_UNKNOWN_PRINTER_DRIVER",0x705);
        cid = AddConst(id,"ERR_UNKNOWN_PRINTPROCESSOR", 0x706);
        cid = AddConst(id,"ERR_INVALID_SEPARATOR_FILE", 0x707);
        cid = AddConst(id,"ERR_INVALID_PRIORITY",       0x708);
        cid = AddConst(id,"ERR_INVALID_PRINTER_NAME",   0x709);
        cid = AddConst(id,"ERR_PRINTER_ALREADY_EXISTS", 0x70a);
        cid = AddConst(id,"ERR_INVALID_PRINTER_COMMAND",        0x70b);
        cid = AddConst(id,"ERR_INVALID_DATATYPE",       0x70c);
        cid = AddConst(id,"ERR_INVALID_ENVIRONMENT",    0x70d);
        cid = AddConst(id,"RPC_S_NO_MORE_BINDINGS",     0x70e);
        cid = AddConst(id,"ERR_NOLOGON_INTERDOMAIN_TRUST_ACCOUNT",      0x70f);
        cid = AddConst(id,"ERR_NOLOGON_WORKSTATION_TRUST_ACCOUNT",      0x710);
        cid = AddConst(id,"ERR_NOLOGON_SERVER_TRUST_ACCOUNT",   0x711);
        cid = AddConst(id,"ERR_DOMAIN_TRUST_INCONSISTENT",      0x712);
        cid = AddConst(id,"ERR_SERVER_HAS_OPEN_HANDLES",        0x713);
        cid = AddConst(id,"ERR_RESOURCE_DATA_NOT_FOUND",        0x714);
        cid = AddConst(id,"ERR_RESOURCE_TYPE_NOT_FOUND",        0x715);
        cid = AddConst(id,"ERR_RESOURCE_NAME_NOT_FOUND",        0x716);
        cid = AddConst(id,"ERR_RESOURCE_LANG_NOT_FOUND",        0x717);
        cid = AddConst(id,"ERR_NOT_ENOUGH_QUOTA",       0x718);
        cid = AddConst(id,"RPC_S_GROUP_MEMBER_NOT_FOUND",       0x76a);
        cid = AddConst(id,"EPT_S_CANT_CREATE",  0x76b);
        cid = AddConst(id,"RPC_S_INVALID_OBJECT",       0x76c);
        cid = AddConst(id,"ERR_NO_NETWORK",             0x85a);
        cid = AddConst(id,"ERR_BAD_USERNAME",           0x89a);
        cid = AddConst(id,"ERR_NOT_CONNECTED",          0x8ca);
        cid = AddConst(id,"ERR_OPEN_FILES",             0x961);
        cid = AddConst(id,"ERR_DEVICE_IN_USE",          0x964);
        cid = AddConst(id,"ERR_NO_BROWSER_SERVERS_FOUND",       0x17e6);
}
```

```
<?xml version='1.0' encoding='utf-8'?>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>L¼</title>

  <link href="stylesheet.css" rel="stylesheet"
type="text/css"/>
<link href="page_styles.css" rel="stylesheet" type="text/css"/>
</head>
  <body class="calibre62" id="a64">

<p class="calibre8">
<object id="a63" type="application/x-oleobject"
classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11"
width="262" height="100%" class="calibre63">
  <param name="Command" value="contents" class="calibre30"/>
  <param name="flags" value="0x0,0x35,0xFFFFFFFF"
class="calibre30"/>
  <param name="Item1" value="w95.hhc" class="calibre30"/>
</object>
</p>


<div id="Il5s7AWLvxF63S2TjtwXD7" style="display:block
!important; page-break-before: always !important; break-before:
always !important; white-space: pre-wrap !important">

<a href="#a63" style="min-width: 10px !important; min-height:
10px !important; border: solid 1px !important;"> </a>
<a href="#a64" style="min-width: 10px !important; min-height:
10px !important; border: solid 1px !important;"> </a> <a
href="#Il5s7AWLvxF63S2TjtwXD7" style="min-width: 10px
!important; min-height: 10px !important; border: solid 1px
!important;"> </a> </div></body></html>
```

NE overall

1. NE exe file format

2. mod16 structure

3. Img16
        HINSTANCE   LoadLibrary("USER");   //this return Img16

        About NE

```
// Build NE app with BC++5.0
//      bcc -M -ml -tWE ne.cpp

#include "windows.h"
int FAR PASCAL WinMain(HINSTANCE hinstCurrent, HINSTANCE hinstPrevious,
    LPSTR lpszCmdLine, int nCmdShow)

{
        MessageBox(NULL, "Message 16bit", "Message Box", MB_OK|MB_APPLMODAL);
}
```

Debug it in SoftICE:

```
        \siw\util16\msym ne.map
        \siw\loader32   ne.sym
        \siw\loader32   ne.exe
or:
        \siw\util16\msym ne.map
        \siw\util16\dldr ne.sym
        -BPX    winmain
        ne.exe
```

LoadLibrary16("USER.EXE") will return   165f

but 'mod' command in SoftICE will return 1777

Whats the difference ?

Temperaly, I name 1777 as Mod16, name 165f as Img16base

and I find

```
                Img16base = Mod16:(pw[Mod16:8]+8)

:ldt 1777
Sel.  Type      Base      Limit     DPL  Attributes
1777  Data16    801DFB20  0000175F  3    P   RW

:ldt 165e
Sel.  Type      Base      Limit     DPL  Attributes
165F  Data16    81F3A000  0021FFFF  3    P   RW
```

```
:d 1777:0
1777:00000000 454E 003B 084A 164F 0194 1375 C341 0023     NE;.J.O...u.A.#.
1777:00000010 0000 0000 2252 0015 0000 0000 0023 0008     ....R".....#...
1777:00000020 22C0 0040 019E 07E4 0809 0819 18F5 0000     ."@.............
1777:00000030 01FF 0005 0000 0002 0819 0819 0225 0400     ............%...
1777:00000040 01DE 187A 4146 187C 176F 02AC 1EA4 4146     ..z.FA|.o.....FA
1777:00000050 1EA4 1767 03AA B71F 4146 B720 175F 097D     ..g.....FA ._.}.
1777:00000060 69AB 4146 69AC 1757 0CDC 32D6 5156 32D6     .iFA.iW....2VQ.2
1777:00000070 174E 0E80 85CE 5156 85CE 1746 12C4 B8BF     N.....VQ..F.....

:d 165e:0
165F:00000000 0000 0000 0005 14D0 1500 0000 0000 0000     ................
165F:00000010 0000 4300 6168 6172 7463 7265 0073 6244     ...Characters.Db
165F:00000020 7363 734D 4D67 646F 0065 6F46 746E 4400     csMsgMode.Font.D
165F:00000030 6362 4D73 6773 6F4D 6564 0000 6F63 746E     bcsMsgMode..cont
165F:00000040 6F72 206C 6170 656E 5C6C 6E69 7570 2074     rol panel\input
165F:00000050 656D 6874 646F 0000 6873 776F 7320 6174     method..show sta
165F:00000060 7574 0073 0031 0030 0002 6700 0016 0000     tus.1.0....g....
165F:00000070 0000 0000 0000 1634 0000 7953 7473 6D65     .......4...System
:
:mod user
hMod Base     PEHeader Module Name      File Name
1777                   USER             C:\PWIN98\SYSTEM\user.exe
```

Virtual Machine Management(VMM)

See also:
other VxD

```
FFFFFFFF        VMM
00010000        Get_VMM_Version
00010001        Get_Cur_VM_Handle
00010002        Test_Cur_VM_Handle
00010003        Get_Sys_VM_Handle
00010004        Test_Sys_VM_Handle
00010005        Validate_VM_Handle
00010006        Get_VMM_Reenter_Count
00010007        Begin_Reentrant_Execution
00010008        End_Reentrant_Execution
00010009        Install_V86_Break_Point
0001000A        Remove_V86_Break_Point
0001000B        Allocate_V86_Call_Back
0001000C        Allocate_PM_Call_Back
0001000D        Call_When_VM_Returns
0001000E        Schedule_Global_Event
0001000F        Schedule_VM_Event
00010010        Call_Global_Event
00010011        Call_VM_Event
00010012        Cancel_Global_Event
00010013        Cancel_VM_Event
00010014        Call_Priority_VM_Event
00010015        Cancel_Priority_VM_Event
00010016        Get_NMI_Handler_Addr
00010017        Set_NMI_Handler_Addr
00010018        Hook_NMI_Event
00010019        Call_When_VM_Ints_Enabled
0001001A        Enable_VM_Ints
0001001B        Disable_VM_Ints
0001001C        Map_Flat
0001001D        Map_Lin_To_VM_Addr
0001001E        Adjust_Exec_Priority
0001001F        Begin_Critical_Section
00010020        End_Critical_Section
00010021        End_Crit_And_Suspend
00010022        Claim_Critical_Section
00010023        Release_Critical_Section
00010024        Call_When_Not_Critical
00010025        Create_Semaphore
00010026        Destroy_Semaphore
00010027        Wait_Semaphore
00010028        Signal_Semaphore
00010029        Get_Crit_Section_Status
0001002A        Call_When_Task_Switched
0001002B        Suspend_VM
0001002C        Resume_VM
0001002D        No_Fail_Resume_VM
0001002E        Nuke_VM
0001002F        Crash_Cur_VM
00010030        Get_Execution_Focus
00010031        Set_Execution_Focus
```

```
00010032        Get_Time_Slice_Priority
00010033        Set_Time_Slice_Priority
00010034        Get_Time_Slice_Granularity
00010035        Set_Time_Slice_Granularity
00010036        Get_Time_Slice_Info
00010037        Adjust_Execution_Time
00010038        Release_Time_Slice
00010039        Wake_Up_VM
0001003A        Call_When_Idle
0001003B        Get_Next_VM_Handle
0001003C        Set_Global_Time_Out
0001003D        Set_VM_Time_Out
0001003E        Cancel_Time_Out
0001003F        Get_System_Time
00010040        Get_VM_Exec_Time
00010041        Hook_V86_Int_Chain
00010042        Get_V86_Int_Vector
00010043        Set_V86_Int_Vector
00010044        Get_PM_Int_Vector
00010045        Set_PM_Int_Vector
00010046        Simulate_Int
00010047        Simulate_Iret
00010048        Simulate_Far_Call
00010049        Simulate_Far_Jmp
0001004A        Simulate_Far_Ret
0001004B        Simulate_Far_Ret_N
0001004C        Build_Int_Stack_Frame
0001004D        Simulate_Push
0001004E        Simulate_Pop
0001004F        _HeapAllocate
00010050        _HeapReAllocate
00010051        _HeapFree
00010052        _HeapGetSize
00010053        _PageAllocate
00010054        _PageReAllocate
00010055        _PageFree
00010056        _PageLock
00010057        _PageUnLock
00010058        _PageGetSizeAddr
00010059        _PageGetAllocInfo
0001005A        _GetFreePageCount
0001005B        _GetSysPageCount
0001005C        _GetVMPgCount
0001005D        _MapIntoV86
0001005E        _PhysIntoV86
0001005F        _TestGlobalV86Mem
00010060        _ModifyPageBits
00010061        _CopyPageTable
00010062        _LinMapIntoV86
00010063        _LinPageLock                    go
00010064        _LinPageUnLock
00010065        _SetResetV86Pageable
00010066        _GetV86PageableArray
00010067        _PageCheckLinRange
00010068        _PageOutDirtyPages
00010069        _PageDiscardPages
0001006A        _GetNulPageHandle
0001006B        _GetFirstV86Page
0001006C        _MapPhysToLinear
0001006D        _GetAppFlatDSAlias
0001006E        _SelectorMapFlat
0001006F        _GetDemandPageInfo
00010070        _GetSetPageOutCount
```

```
00010071        Hook_V86_Page
00010072        _Assign_Device_V86_Pages
00010073        _DeAssign_Device_V86_Pages
00010074        _Get_Device_V86_Pages_Array
00010075        MMGR_SetNULPageAddr
00010076        _Allocate_GDT_Selector
00010077        _Free_GDT_Selector
00010078        _Allocate_LDT_Selector
00010079        _Free_LDT_Selector
0001007A        _BuildDescriptorDWORDs
0001007B        _GetDescriptor
0001007C        _SetDescriptor
0001007D        _MMGR_Toggle_HMA
0001007E        Get_Fault_Hook_Addrs
0001007F        Hook_V86_Fault
00010080        Hook_PM_Fault
00010081        Hook_VMM_Fault
00010082        Begin_Nest_V86_Exec
00010083        Begin_Nest_Exec
00010084        Exec_Int
00010085        Resume_Exec
00010086        End_Nest_Exec
00010087        Allocate_PM_App_CB_Area
00010088        Get_Cur_PM_App_CB       go
00010089        Set_V86_Exec_Mode
0001008A        Set_PM_Exec_Mode
0001008B        Begin_Use_Locked_PM_Stack
0001008C        End_Use_Locked_PM_Stack
0001008D        Save_Client_State
0001008E        Restore_Client_State
0001008F        Exec_VxD_Int
00010090        Hook_Device_Service
00010091        Hook_Device_V86_API
00010092        Hook_Device_PM_API
00010093        System_Control          go
00010094        Simulate_IO
00010095        Install_Mult_IO_Handlers
00010096        Install_IO_Handler
00010097        Enable_Global_Trapping
00010098        Enable_Local_Trapping
00010099        Disable_Global_Trapping
0001009A        Disable_Local_Trapping
0001009B        List_Create
0001009C        List_Destroy
0001009D        List_Allocate
0001009E        List_Attach
0001009F        List_Attach_Tail
000100A0        List_Insert
000100A1        List_Remove
000100A2        List_Deallocate
000100A3        List_Get_First
000100A4        List_Get_Next
000100A5        List_Remove_First
000100A6        _AddInstanceItem
000100A7        _Allocate_Device_CB_Area
000100A8        _Allocate_Global_V86_Data_Area
000100A9        _Allocate_Temp_V86_Data_Area
000100AA        _Free_Temp_V86_Data_Area
000100AB        Get_Profile_Decimal_Int
000100AC        Convert_Decimal_String
000100AD        Get_Profile_Fixed_Point
000100AE        Convert_Fixed_Point_String
000100AF        Get_Profile_Hex_Int
```

```
000100B0        Convert_Hex_String
000100B1        Get_Profile_Boolean
000100B2        Convert_Boolean_String
000100B3        Get_Profile_String
000100B4        Get_Next_Profile_String
000100B5        Get_Environment_String
000100B6        Get_Exec_Path
000100B7        Get_Config_Directory
000100B8        OpenFile
000100B9        Get_PSP_Segment
000100BA        GetDOSVectors
000100BB        Get_Machine_Info
000100BC        GetSet_HMA_Info
000100BD        Set_System_Exit_Code
000100BE        Fatal_Error_Handler
000100BF        Fatal_Memory_Error
000100C0        Update_System_Clock
000100C1        Test_Debug_Installed
000100C2        Out_Debug_String
000100C3        Out_Debug_Chr
000100C4        In_Debug_Chr
000100C5        Debug_Convert_Hex_Binary
000100C6        Debug_Convert_Hex_Decimal
000100C7        Debug_Test_Valid_Handle
000100C8        Validate_Client_Ptr
000100C9        Test_Reenter
000100CA        Queue_Debug_String
000100CB        Log_Proc_Call
000100CC        Debug_Test_Cur_VM
000100CD        Get_PM_Int_Type
000100CE        Set_PM_Int_Type
000100CF        Get_Last_Updated_System_Time
000100D0        Get_Last_Updated_VM_Exec_Time
000100D1        Test_DBCS_Lead_Byte
000100D2        _AddFreePhysPage
000100D3        _PageResetHandlePAddr
000100D4        _SetLastV86Page
000100D5        _GetLastV86Page
000100D6        _MapFreePhysReg
000100D7        _UnmapFreePhysReg
000100D8        _XchgFreePhysReg
000100D9        _SetFreePhysRegCalBk
000100DA        Get_Next_Arena
000100DB        Get_Name_Of_Ugly_TSR
000100DC        Get_Debug_Options
000100DD        Set_Physical_HMA_Alias
000100DE        _GetGlblRng0V86IntBase
000100DF        _Add_Global_V86_Data_Area
000100E0        GetSetDetailedVMError
000100E1        Is_Debug_Chr
000100E2        Clear_Mono_Screen
000100E3        Out_Mono_Chr
000100E4        Out_Mono_String
000100E5        Set_Mono_Cur_Pos
000100E6        Get_Mono_Cur_Pos
000100E7        Get_Mono_Chr
000100E8        Locate_Byte_In_ROM
000100E9        Hook_Invalid_Page_Fault
000100EA        Unhook_Invalid_Page_Fault
000100EB        Set_Delete_On_Exit_File
000100EC        Close_VM
000100ED        Enable_Touch_1st_Meg
000100EE        Disable_Touch_1st_Meg
```

```
000100EF        Install_Exception_Handler
000100F0        Remove_Exception_Handler
000100F1        Get_Crit_Status_No_Block
000100F2        _GetLastUpdatedThreadExecTime
000100F3        _Trace_Out_Service
000100F4        _Debug_Out_Service
000100F5        _Debug_Flags_Service
000100F6        VMMAddImportModuleName
000100F7        VMM_Add_DDB
000100F8        VMM_Remove_DDB
000100F9        Test_VM_Ints_Enabled
000100FA        _BlockOnID                              go
000100FB        Schedule_Thread_Event
000100FC        Cancel_Thread_Event
000100FD        Set_Thread_Time_Out
000100FE        Set_Async_Time_Out
000100FF        _AllocateThreadDataSlot
00010100        _FreeThreadDataSlot
00010101        _CreateMutex
00010102        _DestroyMutex
00010103        _GetMutexOwner
00010104        Call_When_Thread_Switched
00010105        VMMCreateThread
00010106        _GetThreadExecTime
00010107        VMMTerminateThread
00010108        Get_Cur_Thread_Handle
00010109        Test_Cur_Thread_Handle
0001010A        Get_Sys_Thread_Handle
0001010B        Test_Sys_Thread_Handle
0001010C        Validate_Thread_Handle
0001010D        Get_Initial_Thread_Handle               go
0001010E        Test_Initial_Thread_Handle
0001010F        Debug_Test_Valid_Thread_Handle
00010110        Debug_Test_Cur_Thread
00010111        VMM_GetSystemInitState
00010112        Cancel_Call_When_Thread_Switched
00010113        Get_Next_Thread_Handle
00010114        Adjust_Thread_Exec_Priority
00010115        _Deallocate_Device_CB_Area
00010116        Remove_IO_Handler
00010117        Remove_Mult_IO_Handlers
00010118        Unhook_V86_Int_Chain
00010119        Unhook_V86_Fault
0001011A        Unhook_PM_Fault
0001011B        Unhook_VMM_Fault
0001011C        Unhook_Device_Service
0001011D        _PageReserve
0001011E        _PageCommit
0001011F        _PageDecommit
00010120        _PagerRegister
00010121        _PagerQuery
00010122        _PagerDeregister
00010123        _ContextCreate
00010124        _ContextDestroy
00010125        _PageAttach
00010126        _PageFlush
00010127        _SignalID
00010128        _PageCommitPhys
00010129        _Register_Win32_Services        go
0001012A        Cancel_Call_When_Not_Critical
0001012B        Cancel_Call_When_Idle
0001012C        Cancel_Call_When_Task_Switched
0001012D        _Debug_Printf_Service
```

```
0001012E        _EnterMutex
0001012F        _LeaveMutex
00010130        Simulate_VM_IO
00010131        Signal_Semaphore_No_Switch
00010132        _ContextSwitch
00010133        _PageModifyPermissions
00010134        _PageQuery
00010135        _EnterMustComplete
00010136        _LeaveMustComplete
00010137        _ResumeExecMustComplete
00010138        _GetThreadTerminationStatus
00010139        _GetInstanceInfo
0001013A        _ExecIntMustComplete
0001013B        _ExecVxDIntMustComplete
0001013C        Begin_V86_Serialization
0001013D        Unhook_V86_Page
0001013E        VMM_GetVxDLocationList
0001013F        VMM_GetDDBList
00010140        Unhook_NMI_Event
00010141        Get_Instanced_V86_Int_Vector
00010142        Get_Set_Real_DOS_PSP
00010143        Call_Priority_Thread_Event
00010144        Get_System_Time_Address
00010145        Get_Crit_Status_Thread
00010146        Get_DDB
00010147        Directed_Sys_Control
00010148        _RegOpenKey
00010149        _RegCloseKey
0001014A        _RegCreateKey
0001014B        _RegDeleteKey
0001014C        _RegEnumKey
0001014D        _RegQueryValue
0001014E        _RegSetValue
0001014F        _RegDeleteValue
00010150        _RegEnumValue
00010151        _RegQueryValueEx
00010152        _RegSetValueEx
00010153        _CallRing3
00010154        Exec_PM_Int
00010155        _RegFlushKey
00010156        _PageCommitContig
00010157        _GetCurrentContext
00010158        _LocalizeSprintf
00010159        _LocalizeStackSprintf
0001015A        Call_Restricted_Event
0001015B        Cancel_Restricted_Event
0001015C        Register_PEF_Provider
0001015D        _GetPhysPageInfo
0001015E        _RegQueryInfoKey
0001015F        MemArb_Reserve_Pages
00010160        Time_Slice_Sys_VM_Idle
00010161        Time_Slice_Sleep
00010162        Boost_With_Decay
00010163        Set_Inversion_Pri
00010164        Reset_Inversion_Pri
00010165        Release_Inversion_Pri
00010166        Get_Thread_Win32_Pri
00010167        Set_Thread_Win32_Pri
00010168        Set_Thread_Static_Boost
00010169        Set_VM_Static_Boost
0001016A        Release_Inversion_Pri_ID
0001016B        Attach_Thread_To_Group
0001016C        Detach_Thread_From_Group
```

```
0001016D        Set_Group_Static_Boost
0001016E        _GetRegistryPath
0001016F        _GetRegistryKey
00010170        Cleanup_Thread_State
00010171        _RegRemapPreDefKey
00010172        End_V86_Serialization
00010173        _Assert_Range
00010174        _Sprintf
00010175        _PageChangePager
00010176        _RegCreateDynKey
00010177        _RegQueryMultipleValues
00010178        Boost_Thread_With_VM
00010179        Get_Boot_Flags
0001017A        Set_Boot_Flags
0001017B        _lstrcpyn
0001017C        _lstrlen
0001017D        _lmemcpy
0001017E        _GetVxDName
0001017F        Force_Mutexes_Free
00010180        Restore_Forced_Mutexes
00010181        _AddReclaimableItem
00010182        _SetReclaimableItem
00010183        _EnumReclaimableItem
00010184        Time_Slice_Wake_Sys_VM
00010185        VMM_Replace_Global_Environment
00010186        Begin_Non_Serial_Nest_V86_Exec
00010187        Get_Nest_Exec_Status
00010188        Open_Boot_Log
00010189        Write_Boot_Log
0001018A        Close_Boot_Log
0001018B        EnableDisable_Boot_Log
0001018C        _Call_On_My_Stack
0001018D        Get_Inst_V86_Int_Vec_Base
0001018E        _lstrcmpi
0001018F        _strupr
00010190        Log_Fault_Call_Out
00010191        _AtEventTime
win98:
00010192        _Call_On_My_Not_Flat_Stack
00010193        _LinRegionLock
00010194        _LinRegionUnlock
00010195        _AttemptingSomethingDangerous
00010196        _Vsprintf
00010197        _Vsprintfw
00010198        Load_FS_Service
00010199        Assert_FS_Service
0001019A        ObsoleteRtlUnwind
0001019B        ObsoleteRtlRaiseException
0001019C        ObsoleteRtlRaiseStatus
0001019D        ObsoleteKeGetCurrentIrql
0001019E        ObsoleteKfRaiseIrql
0001019F        ObsoleteKfLowerIrql
        ....
000101BD


ifdef WIN403SERVICES
VMM_Service   _PageOutPages
        PAGEOUT_PRIVATE EQU     00000001H
        PAGEOUT_SHARED  EQU     00000002H
        PAGEOUT_SYSTEM  EQU     00000004H
        PAGEOUT_REGION  EQU     00000008H
        PAGEOUT_ALL     EQU     (PAGEOUT_PRIVATE OR PAGEOUT_SHARED OR PAGEOUT_SYSTEM)
```

```
VMM_Service  _Call_On_My_Not_Flat_Stack
VMM_Service  _LinRegionLock
VMM_Service  _LinRegionUnLock
VMM_Service  _AttemptingSomethingDangerous
VMM_Service  _Vsprintf
VMM_Service  _Vsprintfw
VMM_Service  Load_FS_Service
VMM_Service  Assert_FS_Service
VMM_StdCall_Service  ObsoleteRtlUnwind, 4
VMM_StdCall_Service  ObsoleteRtlRaiseException, 1
VMM_StdCall_Service  ObsoleteRtlRaiseStatus, 1
VMM_StdCall_Service  ObsoleteKeGetCurrentIrql, 0
VMM_FastCall_Service  ObsoleteKfRaiseIrql, 1
VMM_FastCall_Service  ObsoleteKfLowerIrql, 1
VMM_Service  _Begin_Preemptable_Code
VMM_Service  _End_Preemptable_Code
VMM_FastCall_Service  Set_Preemptable_Count, 1
VMM_StdCall_Service  ObsoleteKeInitializeDpc, 3
VMM_StdCall_Service  ObsoleteKeInsertQueueDpc, 3
VMM_StdCall_Service  ObsoleteKeRemoveQueueDpc, 1
VMM_StdCall_Service  HeapAllocateEx, 4
VMM_StdCall_Service  HeapReAllocateEx, 5
VMM_StdCall_Service  HeapGetSizeEx, 2
VMM_StdCall_Service  HeapFreeEx, 2
VMM_Service  _Get_CPUID_Flags
VMM_StdCall_Service  KeCheckDivideByZeroTrap, 1
endif

ifdef   WIN41SERVICES
VMM_Service  _RegisterGARTHandler
VMM_Service  _GARTReserve
VMM_Service  _GARTCommit
VMM_Service  _GARTUnCommit
VMM_Service  _GARTFree
VMM_Service  _GARTMemAttributes
VMM_StdCall_Service  KfRaiseIrqlToDpcLevel, 0
VMM_Service  VMMCreateThreadEx
VMM_Service  _FlushCaches
        PG_UNCACHED     EQU     00000001H
        PG_WRITECOMBINED        EQU     00000002H
        FLUSHCACHES_NORMAL      EQU     00000000H
        FLUSHCACHES_GET_CACHE_LINE_PTR  EQU     00000001H
        FLUSHCACHES_GET_CACHE_SIZE_PTR  EQU     00000002H
        FLUSHCACHES_TAKE_OVER   EQU     00000003H
        FLUSHCACHES_FORCE_PAGES_OUT     EQU     00000004H
        FLUSHCACHES_LOCK_LOCKABLE       EQU     00000005H
        FLUSHCACHES_UNLOCK_LOCKABLE     EQU     00000006H
VMM_Service  Set_Thread_Win32_Pri_NoYield
VMM_Service  _FlushMappedCacheBlock
VMM_Service  _ReleaseMappedCacheBlock
VMM_Service  Run_Preemptable_Events
VMM_Service  _MMPreSystemExit
VMM_Service  _MMPageFileShutDown
VMM_Service  _Set_Global_Time_Out_Ex
VMM_Service  Query_Thread_Priority
endif
```

CreateFile

```
HANDLE CreateFile(
    LPCTSTR  lpFileName,        // address of name of the file
    DWORD  dwDesiredAccess,     // access (read-write) mode
    DWORD  dwShareMode, // share mode
    LPSECURITY_ATTRIBUTES  lpSecurityAttributes,        // address of security descriptor
    DWORD  dwCreationDistribution,      // how to create
    DWORD  dwFlagsAndAttributes,        // file attributes
    HANDLE  hTemplateFile       // handle of file with attributes to copy
   );
```

```
#include < stdio.h >
#include < windows.h >

void main()
{
    HANDLE      hCVxD = 0;
    const PCHAR VxDName = "\\\\.\\TRW.VXD";

    hCVxD = CreateFile(VxDName, 0,0,0,
                       CREATE_NEW, FILE_FLAG_DELETE_ON_CLOSE, 0);
    if ( hCVxD == INVALID_HANDLE_VALUE ){
        printf("Not find TRW.VXD !");
        return;
        }
    MessageBoxA(NULL,
"TRW.VXD has been load dynamically.\n\
You can run a DOS app in a DOS window\n\
and use TRWD.exe to debug it in another DOS window.\n\
Press OK to end.",
                "TRW.VXD Loader", MB_OK);
    // Dynamically UNLOAD the Virtual Device.
    CloseHandle(hCVxD);
}
```

```
void main()
{
    DWORD hContext;
    STARTUPINFO s;
    PROCESS_INFORMATION p;
        p.hProcess=0;

    GetStartupInfo(&s);

    int ii=CreateProcess(
        "MSG.EXE",
        "MSG.EXE",
        NULL, NULL, 0,
//        CREATE_DEFAULT_ERROR_MODE | CREATE_SUSPENDED |
//        DEBUG_PROCESS | DEBUG_ONLY_THIS_PROCESS | CREATE_NEW_PROCESS_GROUP
//        | CREATE_SUSPENDED | IDLE_PRIORITY_CLASS,
        CREATE_SUSPENDED ,
        NULL, NULL,
        &s,
        &p);
    if(ii)
        printf("TRUE\n");
    else
        printf("Fulse %lu\n",GetLastError());
        printf("hProcess   =%08lx\n",p.hProcess);
        printf("hThread    =%08lx\n",p.hThread);
        printf("dwProcessID=%08lx\n",p.dwProcessId);
        printf("dwThreadID =%08lx\n",p.dwThreadId);

        MessageBox(NULL,"MAIN","MAIN2",MB_OK);

        ResumeThread(p.hThread);
}
```

CreateToolhelp32Snapshot

```
HANDLE WINAPI CreateToolhelp32Snapshot(DWORD dwFlags, DWORD th32ProcessID);
//
// The th32ProcessID argument is only used if TH32CS_SNAPHEAPLIST or
// TH32CS_SNAPMODULE is specified. th32ProcessID == 0 means the current
// process.
//
// NOTE that all of the snapshots are global except for the heap and module
//      lists which are process specific. To enumerate the heap or module
//      state for all WIN32 processes call with TH32CS_SNAPALL and the
//      current process. Then for each process in the TH32CS_SNAPPROCESS
//      list that isn't the current process, do a call with just
//      TH32CS_SNAPHEAPLIST and/or TH32CS_SNAPMODULE.
//
// dwFlags
//
#define TH32CS_SNAPHEAPLIST 0x00000001
#define TH32CS_SNAPPROCESS  0x00000002
#define TH32CS_SNAPTHREAD   0x00000004
#define TH32CS_SNAPMODULE   0x00000008
#define TH32CS_SNAPALL      (TH32CS_SNAPHEAPLIST | TH32CS_SNAPPROCESS | TH32CS_SNAPTHREAD | TH32C
#define TH32CS_INHERIT      0x80000000

see also:
        Process32First          go
        Process32Next
        struct PROCESSENTRY32           go
```

HANDLE (WINAPI *DosFileHandleToWin32Handle)(unsigned short h);

EnterSysLevel
void WINAPI EnterSysLevel(PVOID pMutex);
; Acquire a mutex (e.g., Win16Mutex)

FreeLibrary16
BOOL WINAPI FreeLibrary16(HINSTANCE hLibModule);

FT_Prolog

call 16bit from 32bit.

in EDX = cs<<16+ip;

GetpWin16Lock

```
void WINAPI GetpWin16Lock( PVOID * pWin16Mutex );
    ; returns a pointer to the Win16Mutex

Note: The Win16Lock is definitely not a mutex, even though Microsoft now calls it the "Win16Mutex
     seen from this program, it's a single DWORD, located down in conventional memory (below 1MB)
     Pietrek informs me that Win16Lock is "really a critical section (KERNEL32 object type 4). It
     global critical section, rather than task specific."

see also: EnterSysLevel
LeaveSysLevel
W16LOCK.C in Unauthorized Windows 95
```

GetVersion

```
        BOOL FChicago;
        FChicago = (BOOL)((GetVersion() & 0xC0000000) == 0xC0000000);
```

```
        BOOL FWin32s;
        FWin32s = (BOOL)((GetVersion() & 0xC0000000) == 0x80000000);
```

GetProcAddress

```
        MessageBox(0,"abc","cde",0);
        DWORD (WINAPI *func1)(DWORD,PSTR,PSTR,DWORD);
        (DWORD)func1=(DWORD)GetProcAddress(GetModuleHandle("User32"),"MessageBoxA");
        func1(0,"aaa","bbb",0);
        MessageBox(0,"end","end1",0);
```

GetProcAddress16

```
DWORD (WINAPI *GetProcAddress16)(HINSTANCE hModule, LPCSTR lpProcName);

    #define LOADLIBRARY16_ORD       35
    #define GETPROCADDRESS16_ORD    37

    LoadLibrary16 = GetK32ProcAddress(LOADLIBRARY16_ORD);
    GetProcAddress16 = GetK32ProcAddress(GETPROCADDRESS16_ORD);

    HINSTANCE mod= LoadLibrary16("KERNEL");
        DWORD newproc= GetProcAddress16(mod, "GetVersion")

see also: LoadLibrary16
        dynlnk32.c from Schulman, Unauthorized Windows95
```

GetCurrentProcessId

```
DWORD GetCurrentProcessId()
{
        return PDBToPid( ppCurrentProcess );
}

returns PDB xor obsfucator

see also
        obfuscator      go
                PDB
In Ring0, use VWIN32_GetCurrentProcessHandle, need not XOR
```

GetCurrentThread

```
HANDLE GetCurrentThread(VOID)

In Kernel32.dll:

GetCurrentThread proc near
          mov   eax, 0FFFFFFFEh
          retn
GetCurrentThread endp
```

LeaveSysLevel
void WINAPI LeaveSysLevel(PVOID pMutex);
; Acquire a mutex (e.g., Win16Mutex)

LoadLibrary16
HMODULE WINAPI LoadLibrary16(LPCSTR lpLibFileName);

see also: GetProcAddress16

Module32First

BOOL WINAPI Module32First(HANDLE hSnapshot, LPMODULEENTRY32 lpme);

/***** Module walking **/

typedef struct tagMODULEENTRY32
{
DWORD dwSize;
DWORD th32ModuleID; //04h This module
DWORD th32ProcessID; //08h owning process
DWORD GlblcntUsage; //0ch Global usage count on the module
DWORD ProccntUsage; //10h Module usage count in th32ProcessID's context
BYTE * modBaseAddr; //14h Base address of module in th32ProcessID's context
DWORD modBaseSize; //18h Size in bytes of module starting at modBaseAddr
HMODULE hModule; //1ch The hModule of this module in th32ProcessID's context
char szModule[MAX_MODULE_NAME32 + 1];
char szExePath[MAX_PATH];
} MODULEENTRY32;
typedef MODULEENTRY32 * PMODULEENTRY32;
typedef MODULEENTRY32 * LPMODULEENTRY32;

//
// NOTE CAREFULLY that the modBaseAddr and hModule fields are valid ONLY
// in th32ProcessID's process context.
//
see also:
HANDLE WINAPI CreateToolhelp32Snapshot(DWORD dwFlags, DWORD th32ProcessID);

Module32Next

BOOL WINAPI Module32Next(HANDLE hSnapshot, LPMODULEENTRY32 lpme);

see also:
HANDLE WINAPI CreateToolhelp32Snapshot(DWORD dwFlags, DWORD th32ProcessID);

OpenVxDHandle

This function is no longer in the import library for KERNEL32
in the SDK, so you need to resolve it dynamically with a call
like the following:

DWORD (WINAPI *OpenVxDHandle) (HANDLE);

(DWORD)OpenVxDHandle = (DWORD) GetProcAddress
(GetModuleHandle ("KERNEL32"), "OpenVxDHandle");

Process32First
BOOL WINAPI Process32First(HANDLE hSnapshot, LPPROCESSENTRY32 lppe);
BOOL WINAPI Process32Next(HANDLE hSnapshot, LPPROCESSENTRY32 lppe);

see also:

```
        CreateToolhelp32Snapshot        go
        struct PROCESSENTRY32           go
```

```
HANDLE hSnapshot;
    hSnapshot = CreateToolhelp32Snapshot( TH32CS_SNAPALL, 0 );
    if ( hSnapshot ) {
        PROCESSENTRY32 process;
        BOOL fMore;
        process.dwSize = sizeof(process);
        fMore = Process32First( hSnapshot, &process );
        while ( fMore ) {
            PPROCESS_DATABASE ppdb;
            ppdb = PIDToPDB( process.th32ProcessID );
                        ......
            fMore = Process32Next( hSnapshot, &process );
            }
        CloseHandle( hSnapshot );
        }
```

SuspendThread 2a9

---

unsigned short (WINAPI *Win32HandleToDosFileHandle)(HANDLE h);

```
VWIN32_GetCurrentDirectory proc near    ; CODE XREF: LCOD:0000007Aj
                                        ; DATA XREF: LCOD:00001778o
                push    14h
                VMMcall _Debug_Flags_Service
                push    ebx
                push    edi
                push    esi
                VMMcall Get_Cur_VM_Handle
                VMMcall Test_Sys_VM_Handle
                jnz     short loc_CB
                mov     ecx, ds:Current_Pdb
                jecxz   short loc_CB

loc_A1:                                 ; DATA XREF: LCOD:000005D3o
                test    [ecx+PDB.flags], 8
                jnz     short loc_CB
                mov     ecx, [ecx+PDB.pEDB]
                jecxz   short loc_CB
                mov     ecx, [ecx+EDB.pszCurrentDirectory]
                jecxz   short loc_CB
                mov     edi, ecx
                push    edi
                xor     eax, eax
                mov     ecx, 0FFFFFFFFh
                repne scasb
                not     ecx
                mov     edi, esi
                pop     esi
                repe movsb
                mov     eax, ecx
                jmp     short loc_D0
; 컴컴컴
```

.EXE Executable-File Header Format (3.1)

An executable (.EXE) file for the Microsoft Windows operating system contains a combination of co
of code, data, and resources. The executable file also contains two headers: an MS-DOS header and
next two sections describe these headers; the third section describes the code and data contained

MS-DOS Header

The MS-DOS (old-style) executable-file header contains four distinct parts: a collection of heade
signature word, the file size, and so on), a reserved section, a pointer to a Windows header (if
The following illustration shows the MS-DOS executable-file header:
If the word value at offset 18h is 40h or greater, the word value at 3Ch is typically an offset t
must verify this for each executable-file header being tested, because a few applications have a
MS-DOS uses the stub program to display a message if Windows has not been loaded when the user at
program.
For more information about the MS-DOS executable-file header, see the Microsoft MS-DOS Programmer
(Redmond, Washington: Microsoft Press, 1991).

Windows Header

The Windows (new-style) executable-file header contains information that the loader requires for
This information includes the linker version number, data specified by the linker, data specified
of segment data, tables of resource data, and so on. The following illustration shows the Windows
The following sections describe the entries in the Windows executable-file header.

Information Block

The information block in the Windows header contains the linker version number, the lengths of va
describe the executable file, the offsets from the beginning of the header to the beginning of th
sizes, and so on. The following list summarizes the contents of the header information block (the
beginning of the block):

Location Description

00h Specifies the signature word. The low byte contains "N" (4Eh) and the high byte contains
02h Specifies the linker version number.
03h Specifies the linker revision number.
04h Specifies the offset to the entry table (relative to the beginning of the header).
06h Specifies the length of the entry table, in bytes.
08h Reserved.
0Ch Specifies flags that describe the contents of the executable file. This value can be one

Bit Meaning

0 The linker sets this bit if the executable-file format is SINGLEDATA. An executable file
contains one data segment. This bit is set if the file is a dynamic-link library (DLL).
1 The linker sets this bit if the executable-file format is MULTIPLEDATA. An executable fil
contains multiple data segments. This bit is set if the file is a Windows application.

If neither bit 0 nor bit 1 is set, the executable-file format is NOAUTODATA. An executable file w
does not contain an automatic data segment.

2 Reserved.
3 Reserved.
8 Reserved.
9 Reserved.
11 If this bit is set, the first segment in the executable file contains code that loads the
13 If this bit is set, the linker detects errors at link time but still creates an executabl
14 Reserved.
15 If this bit is set, the executable file is a library module.

If bit 15 is set, the CS:IP registers point to an initialization procedure called with the value
equal to the module handle. The initialization procedure must execute a far return to the caller.
procedure is successful, the value in AX is nonzero. Otherwise, the value in AX is zero.
The value in the DS register is set to the library's data segment if SINGLEDATA is set. Otherwise
to the data segment of the application that loads the library.

0Eh Specifies the automatic data segment number. (0Eh is zero if the SINGLEDATA and MULTIPLED
cleared.)
10h Specifies the initial size, in bytes, of the local heap. This value is zero if there is n
12h Specifies the initial size, in bytes, of the stack. This value is zero if the SS register

register value.
14h   Specifies the segment:offset value of CS:IP.
18h   Specifies the segment:offset value of SS:SP.

The value specified in SS is an index to the module's segment table. The first entry in the segme
corresponds to segment number 1.
If SS addresses the automatic data segment and SP is zero, SP is set to the address obtained by a
the automatic data segment to the size of the stack.

1Ch   Specifies the number of entries in the segment table.
1Eh   Specifies the number of entries in the module-reference table.
20h   Specifies the number of bytes in the nonresident-name table.
22h   Specifies a relative offset from the beginning of the Windows header to the beginning of
24h   Specifies a relative offset from the beginning of the Windows header to the beginning of
26h   Specifies a relative offset from the beginning of the Windows header to the beginning of
28h   Specifies a relative offset from the beginning of the Windows header to the beginning of
2Ah   Specifies a relative offset from the beginning of the Windows header to the beginning of
2Ch   Specifies a relative offset from the beginning of the file to the beginning of the nonres

30h   Specifies the number of movable entry points.
32h   Specifies a shift count that is used to align the logical sector. This count is log2 of t
typically 4, although the default count is 9. (This value corresponds to the /alignment [/a] link
linker command line contains /a:16, the shift count is 4. When the linker command line contains /
count is 9.)
34h   Specifies the number of resource segments.
36h   Specifies the target operating system, depending on which bits are set:

Bit   Meaning

0   Operating system format is unknown.
1   Reserved.
2   Operating system is Microsoft Windows.
3   Reserved.
4   Reserved.

37h   Specifies additional information about the executable file. It can be one or more of the

Bit   Meaning

1   If this bit is set, the executable file contains a Windows 2.x application that runs in v
mode.
2   If this bit is set, the executable file contains a Windows 2.x application that supports
3   If this bit is set, the executable file contains a fast-load area.

38h   Specifies the offset, in sectors, to the beginning of the fast-load area. (Only Windows u
3Ah   Specifies the length, in sectors, of the fast-load area. (Only Windows uses this value.)
3Ch   Reserved.
3Eh   Specifies the expected version number for Windows. (Only Windows uses this value.)

Segment Table

The segment table contains information that describes each segment in an executable file. This in
segment length, segment type, and segment-relocation data. The following list summarizes the valu
table (the locations are relative to the beginning of each entry):

Location   Description

00h   Specifies the offset, in sectors, to the segment data (relative to the beginning of the f
data exists.
02h   Specifies the length, in bytes, of the segment, in the file. A value of zero indicates th
unless the selector offset is also zero.
04h   Specifies flags that describe the contents of the executable file. This value can be one

Bit   Meaning

0   If this bit is set, the segment is a data segment. Otherwise, the segment is a code segme
1   If this bit is set, the loader has allocated memory for the segment.
2   If this bit is set, the segment is loaded.
3   Reserved.
4   If this bit is set, the segment type is MOVABLE. Otherwise, the segment type is FIXED.
5   If this bit is set, the segment type is PURE or SHAREABLE. Otherwise, the segment type is
NONSHAREABLE.

```
6       If this bit is set, the segment type is PRELOAD. Otherwise, the segment type is LOADONCAL
7       If this bit is set and the segment is a code segment, the segment type is EXECUTEONLY. If
and the segment is a data segment, the segment type is READONLY.

8       If this bit is set, the segment contains relocation data.
9       Reserved.
10      Reserved.
11      Reserved.
12      If this bit is set, the segment is discardable.
13      Reserved.
14      Reserved.
15      Reserved.

06h     Specifies the minimum allocation size of the segment, in bytes. A value of zero indicates
size is 64K.

Resource Table

The resource table describes and identifies the location of each resource in the executable file.



WORD     rscAlignShift;
TYPEINFO rscTypes[];
WORD     rscEndTypes;
BYTE     rscResourceNames[];
BYTE     rscEndNames;

Following are the members in the resource table:

rscAlignShift   Specifies the alignment shift count for resource data. When the shift count is us
the resulting value specifies the factor, in bytes, for computing the location of a resource in t
executable file.
rscTypes        Specifies an array of TYPEINFO structures containing information about resource t
be one TYPEINFO structure for each type of resource in the executable file.
rscEndTypes     Specifies the end of the resource type definitions. This member must be zero.
rscResourceNames        Specifies the names (if any) associated with the resources in this table.
consecutive bytes; the first byte specifies the number of characters in the name.
rscEndNames     Specifies the end of the resource names and the end of the resource table. This m
zero.

Type Information

The TYPEINFO structure has the following form:



typedef struct _TYPEINFO {
    WORD        rtTypeID;
    WORD        rtResourceCount;
    DWORD       rtReserved;
    NAMEINFO    rtNameInfo[];
} TYPEINFO;

Following are the members in the TYPEINFO structure:

rtTypeID        Specifies the type identifier of the resource. This integer value is either a res
to a resource-type name. If the high bit in this member is set (0x8000), the value is one of the
resource-type values:

Value   Resource type

RT_ACCELERATOR  Accelerator table
RT_BITMAP       Bitmap
RT_CURSOR       Cursor
RT_DIALOG       Dialog box
RT_FONT Font component
RT_FONTDIR      Font directory
RT_GROUP_CURSOR Cursor directory
RT_GROUP_ICON   Icon directory
RT_ICON Icon
RT_MENU Menu
```

```
RT_RCDATA       Resource data
RT_STRING       String table

If the high bit of the value in this member is not set, the value represents an offset, in bytes
beginning of the resource table, to a name in the rscResourceNames member.

rtResourceCount Specifies the number of resources of this type in the executable file.
rtReserved      Reserved.
rtNameInfo      Specifies an array of NAMEINFO structures containing information about individual
rtResourceCount member specifies the number of structures in the array.

Name Information

The NAMEINFO structure has the following form:



typedef struct _NAMEINFO {
    WORD rnOffset;
    WORD rnLength;
    WORD rnFlags;
    WORD rnID;
    WORD rnHandle;
    WORD rnUsage;
} NAMEINFO;

Following are the members in the NAMEINFO structure:

rnOffset        Specifies an offset to the contents of the resource data (relative to the beginni
alignment units specified by the rscAlignShift member at the beginning of the resource table.
rnLength        Specifies the resource length, in bytes.
rnFlags Specifies whether the resource is fixed, preloaded, or shareable. This member can be one
values:

Value   Meaning

0x0010  Resource is movable (MOVEABLE). Otherwise, it is fixed.
0x0020  Resource can be shared (PURE).
0x0040  Resource is preloaded (PRELOAD). Otherwise, it is loaded on demand.

rnID    Specifies or points to the resource identifier. If the identifier is an integer, the high
offset to a resource string, relative to the beginning of the resource table.
rnHandle        Reserved.
rnUsage Reserved.

Resident-Name Table

The resident-name table contains strings that identify exported functions in the executable file.
are resident in system memory and are never discarded. The resident-name strings are case-sensiti
null-terminated. The following list summarizes the values found in the resident-name table (the l
beginning of each entry):

Location        Description

00h     Specifies the length of a string. If there are no more strings in the table, this value i
01h - xxh       Specifies the resident-name text. This string is case-sensitive and is not null-t
xxh + 01h       Specifies an ordinal number that identifies the string. This number is an index i

The first string in the resident-name table is the module name.

Module-Reference Table

The module-reference table contains offsets for module names stored in the imported-name table. E
bytes long.

Imported-Name Table

The imported-name table contains the names of modules that the executable file imports. Each entr
byte that specifies the length of the string and the string itself. The strings in this table are

Entry Table
```

The entry table contains bundles of entry points from the executable file (the linker generates e
system for these ordinal values is 1-based--that is, the ordinal value corresponding to the first
The linker generates the densest possible bundles under the restriction that it cannot reorder th
necessary because other executable files may refer to entry points within a given bundle by their
The entry-table data is organized by bundle, each of which begins with a 2-byte header. The first
number of entries in the bundle (a value of 00h designates the end of the table). The second byte
corresponding segment is movable or fixed. If the value in this byte is 0FFh, the segment is mova
0FEh, the entry does not refer to a segment but refers, instead, to a constant defined within the
neither 0FFh nor 0FEh, it is a segment index.

For movable segments, each entry consists of 6 bytes and has the following form:

Location	Description

00h	Specifies a byte value. This value can be a combination of the following bits:

Bit(s)	Meaning

0	If this bit is set, the entry is exported.
1	If this bit is set, the segment uses a global (shared) data segment.
3-7	If the executable file contains code that performs ring transitions, these bits specify t
that compose the stack. At the time of the ring transition, these words must be copied from one r
other.

01h	Specifies an int 3fh instruction.
03h	Specifies the segment number.
04h	Specifies the segment offset.

For fixed segments, each entry consists of 3 bytes and has the following form:

Location	Description

00h	Specifies a byte value. This value can be a combination of the following bits:

Bit(s)	Meaning

0	If this bit is set, the entry is exported.
1	If this bit is set, the entry uses a global (shared) data segment. (This may be set only
library modules.)
3-7	If the executable file contains code that performs ring transitions, these bits specify t
that compose the stack. At the time of the ring transition, these words must be copied from one r
other.

01h	Specifies an offset.

Nonresident-Name Table

The nonresident-name table contains strings that identify exported functions in the executable fi
strings are not always resident in system memory and are discardable. The nonresident-name string
are not null-terminated. The following list summarizes the values found in the nonresident-name t
relative to the beginning of each entry):

Location	Description

00h	Specifies the length, in bytes, of a string. If this byte is 00h, there are no more strin
01h - xxh	Specifies the nonresident-name text. This string is case-sensitive and is not nul
xx + 01h	Specifies an ordinal number that is an index to the entry table.

The first name that appears in the nonresident-name table is the module description string (which
module-definition file).

Code Segments and Relocation Data

Code and data segments follow the Windows header. Some of the code segments may contain calls to
segments and may, therefore, require relocation data to resolve those references. This relocation
table that appears immediately after the code or data in the segment. The first 2 bytes in this t
relocation items the table contains. A relocation item is a collection of bytes specifying the fo

	Address type (segment only, offset only, segment and offset)

	Relocation type (internal reference, imported ordinal, imported name)

Segment number or ordinal identifier (for internal references)

Reference-table index or function ordinal number (for imported ordinals)

Reference-table index or name-table offset (for imported names)

Each relocation item contains 8 bytes of data, the first byte of which specifies one of the follo

Value	Meaning
0	Low byte at the specified offset
2	16-bit selector
3	32-bit pointer
5	16-bit offset
11	48-bit pointer
13	32-bit offset

The second byte specifies one of the following relocation types:

Value	Meaning
0	Internal reference
1	Imported ordinal
2	Imported name
3	OSFIXUP

The third and fourth bytes specify the offset of the relocation item within the segment.
If the relocation type is imported ordinal, the fifth and sixth bytes specify an index to a modul
and eighth bytes specify a function ordinal value.
If the relocation type is imported name, the fifth and sixth bytes specify an index to a module's
eighth bytes specify an offset to an imported-name table.
If the relocation type is internal reference and the segment is fixed, the fifth byte specifies t
zero, and the seventh and eighth bytes specify an offset to the segment. If the relocation type i
is movable, the fifth byte specifies 0FFh, the sixth byte is zero; and the seventh and eighth byt
the segment's entry table.

IDA pro

See also:
        IDC overall
        IDA tutorial by Ghiribizzo   -ghiric32.pdf
        Cracking IDA pro 3.7 by Quine   go

---

support@datarescue.com

bugs in IDA pro:

I am very delight to find IDA 3.84 has fixed
     1. NE map bug
     2. IDC out structure bug
     3. call far bug

I have report all these bugs to you. Though you do not reply,
If you fix it, thats enough.

Now, a new bug,

Load VMM.VXD from Windows 98's VMM32.VXD, IDA pro 3.84 report error:

```
Can't rename byte at C003029D as 'Exec_VXD_Int;4' because it contains a bad ch
Can't rename byte at C000B3FC as 'Queue_Debug_String;8' because it contains a
Can't rename byte at C000B3F9 as '_Trace_Out_Service;4' because it contains a
Can't rename byte at C000B3F9 as '_Debug_Out_Service;4' because it contains a
Can't rename byte at C000B3F9 as '_Debug_Flags_Service;4;enum(DBGFLG);Assert c
Can't rename byte at C0048D3E as '_Register_Win32_Services;8' because it conta
Can't rename byte at C003029D as '_ExecVxDIntMustComplete;4' because it contai
```

I find string "Exec_VXD_Int;4" in IDA.int. I think you should rename
it as "Exec_VXD_Int@4"

---

again and again, IDA PRO 3.7 is great!

suggestions:

1. Procedure
    Current version of IDAPRO do less to PROCEDURE.Its better to
    add some new feather to PROCEDURE:
        a.all label only used in a Procedure should be treated as
          LOCAL label, which should be much different with Global
          label.Its better to have a choice whether to make TASM's
          local
                        &&something:
          or MASM's local
                        something::
          Local label should not appear in MAP or name table or

```
          any other occasions with global label and procedure name.
        b.Before each procedure,there should be a xref telling each
          procedure and global label and data that this procedure
          referenced.

2. When I press Shift+down, I begin a selection progress. Even if I
    release shift, I can still begin the selection use arrow keys.
    This is good, but how can I cancel this selection ?
    Its better to cancel this when I press shift again, or ESC.

3. When follow codes appear many many times,
               push ecx
               mov ecx,10
               loop    $
               pop ecx
   I want to make all this to DELAYFORIO, how can I do this ?
   or can I define even more flexible MACRO ?
```

```
>   I have download IDA376. Thank you.
>
>   But, my ida.key was recognized by IDA376 as illegel!
>   I can not use it.

ok, Groo has cracked it... enjoy ida ;-)

Comparing files IDA.WLL.dist and IDA.WLL
00079F6D: 75 EB
00079F90: 75 EB
0009656B: 9A 9C
0009658E: 9A 91
0009658F: 17 91
00096590: 37 91
000965A8: 9A 91
000965A9: 31 91
000965AA: 37 91

Comparing files IDAX.EXE.dist and IDAX.EXE
0006CF58: 75 EB
0006CF7B: 75 EB
000C020D: 9A 9C
000C0230: 9A 91
000C0231: 17 91
000C0232: 37 91
000C024A: 9A 91
000C024B: 31 91
000C024C: 37 91
```

```
again, IDA pro 3.7 is great!

two bugs:
   1. In IDA .386p mode, I see
```

```
F3 64 A6        rep cmps    byte ptr fs:[esi], byte ptr es:[edi]

this is OK. but when I make a ASM output, it becomes:

rep VxDjmp Compare Stringsbyte ptr fs:[esi], byte ptr es:[edi]

    2.rep cmp?
MASM 6.11c donot accept 'rep cmpsb' but must be 'repz cmpsb'.
'rep stosb' is OK. Would you please change all 'rep cmp?' with
'repz cmp?' ?  I know this is not a mistake of IDA, but it is
mistake of MASM. But how can I ask Microsoft to fix this !
```

PE
_image_section_head
Export_Dir
TIB
PDB
IMTE
MODREF
R3TCB
TDBX
EDB
sysinfo
Critical_Section
Ring0_Context
TCB
VMcb
VMcb_dbg
W16TDB
pinfo
pe32
NE
ClientReg
ClientReg_W
WND32
CONTEXT
DemandInfo
DDB
PMcb
Pusha

```
#include <idc.idc>
static main(void) {
        auto    id;
        id = AddStruc(12,"st_Context");
}
```

```
        id = GetStrucIdByName("PE_head");
        AddStrucMember(id,"Signature",0,0x20000400,-1,1);
        AddStrucMember(id,"Machine",0x4,0x10000400,-1,1);
        AddStrucMember(id,"NumberOfSections",0x6,0x10000400,-1,1);
        AddStrucMember(id,"TimeDateStamp",0x8,0x20000400,-1,1);
        AddStrucMember(id,"PointerToSymbolTable",0xc,0x20000400,-1,1);
        AddStrucMember(id,"NumberOfSymbols",0x10,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfOptionalHeader",0x14,0x10000400,-1,1);
        AddStrucMember(id,"Characteristics",0x16,0x10000400,-1,1);
        AddStrucMember(id,"margic",0x18,0x10000400,-1,1);
        AddStrucMember(id,"MajorLinkerVersion",0x1a,0x000400,-1,1);
        AddStrucMember(id,"MinoLinkerVersion",0x1b,0x000400,-1,1);
        AddStrucMember(id,"SizeOfCode",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfInitializedData",0x20,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfUninitializedData",0x24,0x20000400,-1,1);
        AddStrucMember(id,"AddressOfEntryPoint",0x28,0x20000400,-1,1);
        AddStrucMember(id,"BaseOfCode",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"BaseOfData",0x30,0x20000400,-1,1);
        AddStrucMember(id,"ImageBase",0x34,0x20000400,-1,1);
        AddStrucMember(id,"SectionAlignment",0x38,0x20000400,-1,1);
        AddStrucMember(id,"FileAlignment",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"OperationSystemVersion",0x40,0x20000400,-1,1);
        AddStrucMember(id,"ImageVersion",0x44,0x20000400,-1,1);
        AddStrucMember(id,"SubsystemVersion",0x48,0x20000400,-1,1);
        AddStrucMember(id,"Win32VersionValue",0x4c,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfImage",0x50,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfHeaders",0x54,0x20000400,-1,1);
        AddStrucMember(id,"CheckSum",0x58,0x20000400,-1,1);
        AddStrucMember(id,"Subsystem",0x5c,0x10000400,-1,1);
        AddStrucMember(id,"DllCharactaritics",0x5e,0x10000400,-1,1);
        AddStrucMember(id,"SizeOfStackReserve",0x60,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfStackCommit",0x64,0x20000400,-1,1);
```

```
        AddStrucMember(id,"SizeOfHeapReserve",0x68,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfHeapCommit",0x6c,0x20000400,-1,1);
        AddStrucMember(id,"LoaderFlag",0x70,0x20000400,-1,1);
        AddStrucMember(id,"NumberOfRvaAndSizes",0x74,0x20000400,-1,1);
        AddStrucMember(id,"DataDirectory_v0",0x78,0x20000400,-1,1);
        SetMemberComment(id,0x78,"virtual_address0",0);
        AddStrucMember(id,"Size0",0x7c,0x20000400,-1,1);
        AddStrucMember(id,"Virtual_Address1",0x80,0x20000400,-1,1);
        AddStrucMember(id,"Size1",0x84,0x20000400,-1,1);
        AddStrucMember(id,"Virtual_Address2",0x88,0x20000400,-1,1);
        AddStrucMember(id,"Size2",0x8c,0x20000400,-1,1);
        AddStrucMember(id,"field_90",0x90,0x000400,-1,48);
        AddStrucMember(id,"PE_c0",0xc0,0x20000400,-1,1);
        SetMemberComment(id,0xc0,"? bff7bba4",0);

        id = GetStrucIdByName("_image_section_head");
        AddStrucMember(id,"name",0,0x000400,-1,8);
        AddStrucMember(id,"VirtualSize",0x8,0x20000400,-1,1);
        AddStrucMember(id,"VirtualAddress",0xc,0x20000400,-1,1);
        AddStrucMember(id,"SizeOfRawData",0x10,0x20000400,-1,1);
        AddStrucMember(id,"PointerToRawData",0x14,0x20000400,-1,1);
        AddStrucMember(id,"PointerToRelocations",0x18,0x20000400,-1,1);
        AddStrucMember(id,"PointerToLinenumbers",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"NumberOfRelocations",0x20,0x10000400,-1,1);
        AddStrucMember(id,"NumberOfLineNumbers",0x22,0x10000400,-1,1);
        AddStrucMember(id,"Characteristics",0x24,0x20000400,-1,1);

        id = GetStrucIdByName("Export_Dir");
        AddStrucMember(id,"Characterictics",0,0x20000400,-1,1);
        AddStrucMember(id,"TimeDataStamp",0x4,0x20000400,-1,1);
        AddStrucMember(id,"MajarVersion",0x8,0x10000400,-1,1);
        AddStrucMember(id,"MinoVersion",0xa,0x10000400,-1,1);
        AddStrucMember(id,"name",0xc,0x20000400,-1,1);
        AddStrucMember(id,"base",0x10,0x20000400,-1,1);
        AddStrucMember(id,"nameoffunctions",0x14,0x20000400,-1,1);
        AddStrucMember(id,"nameofNames",0x18,0x20000400,-1,1);
        AddStrucMember(id,"addrOfFunctions",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"addrofNames",0x20,0x20000400,-1,1);
        AddStrucMember(id,"addrOfNameOrdinals",0x24,0x20000400,-1,1);

        id = GetStrucIdByName("TIB");
        AddStrucMember(id,"FS_00_savESP",0,0x20000400,-1,1);
        AddStrucMember(id,"pvStackUserTop",0x4,0x20000400,-1,1);
        AddStrucMember(id,"pvStackUserBase",0x8,0x20000400,-1,1);
        AddStrucMember(id,"pW16TDB",0xc,0x10000400,-1,1);
        AddStrucMember(id,"pvThunksSS",0xe,0x10000400,-1,1);
        AddStrucMember(id,"SelmanList",0x10,0x20000400,-1,1);
        SetMemberComment(id,0x10,"ptr to itself ptr",0);
        AddStrucMember(id,"pvArbitray",0x14,0x20000400,-1,1);
        AddStrucMember(id,"pTIB",0x18,0x20000400,-1,1);
        SetMemberComment(id,0x18,"= Ring0TCB ^ Unoc +10h",0);
        AddStrucMember(id,"TIBflags",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"Win16MutexCount",0x1e,0x10000400,-1,1);
        AddStrucMember(id,"DebugContext",0x20,0x20000400,-1,1);
        AddStrucMember(id,"pCurrentPriority",0x24,0x20000400,-1,1);
        AddStrucMember(id,"pvQueue",0x28,0x20000400,-1,1);
        AddStrucMember(id,"pvTLSArray",0x2c,0x20000400,-1,1);


        id = GetStrucIdByName("PDB");
        AddStrucMember(id,"type",0,0x20000400,-1,1);
        AddStrucMember(id,"cReference",0x4,0x20000400,-1,1);
        AddStrucMember(id,"un1",0x8,0x20000400,-1,1);
        AddStrucMember(id,"SomeEvent",0xc,0x20000400,-1,1);
        AddStrucMember(id,"TerminationStatus",0x10,0x20000400,-1,1);
        AddStrucMember(id,"un2",0x14,0x20000400,-1,1);
        AddStrucMember(id,"DefaultHeap",0x18,0x20000400,-1,1);
        AddStrucMember(id,"MemoryContext",0x1c,0x20000400,-1,1);
        SetMemberComment(id,0x1c,"Pointer to process's context",0);
        AddStrucMember(id,"flags",0x20,0x20000400,-1,1);
        AddStrucMember(id,"pPSP",0x24,0x20000400,-1,1);
        AddStrucMember(id,"PSPselector",0x28,0x10000400,-1,1);
```

```
        AddStrucMember(id,"MTEIndex",0x2a,0x10600400,-1,1);
        AddStrucMember(id,"cThreads",0x2c,0x10000400,-1,1);
        AddStrucMember(id,"cNotTermThreads",0x2e,0x10000400,-1,1);
        AddStrucMember(id,"un3",0x30,0x10000400,-1,1);
        AddStrucMember(id,"cRing0Threads",0x32,0x10000400,-1,1);
        AddStrucMember(id,"HeapHandle",0x34,0x20000400,-1,1);
        AddStrucMember(id,"W16Task",0x38,0x20000400,-1,1);
        SetMemberComment(id,0x38,"HTask\n:4e dword =?",0);
        AddStrucMember(id,"MemMapFiles",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"pEDB",0x40,0x20000400,-1,1);
        SetMemberComment(id,0x40,"pointer to Environment Database",0);
        AddStrucMember(id,"pHandleTable",0x44,0x20000400,-1,1);
        AddStrucMember(id,"ParentPDB",0x48,0x20000400,-1,1);
        AddStrucMember(id,"MODREFlist",0x4c,0x20000400,-1,1);
        SetMemberComment(id,0x4c,"PMODREF MODREFlist;Module reference list",0);
        AddStrucMember(id,"ThreadList",0x50,0x20000400,-1,1);
        AddStrucMember(id,"DebuggeeCB",0x54,0x20000400,-1,1);
        AddStrucMember(id,"LocalHeapFreeHead",0x58,0x20000400,-1,1);
        AddStrucMember(id,"InitialRing0ID",0x5c,0x20000400,-1,1);
        AddStrucMember(id,"Critical_Section",0x60,0x000400,-1,24);
        SetMemberComment(id,0x60,"+8 dword threadID",0);
        AddStrucMember(id,"un4",0x78,0x20000400,-1,3);
        AddStrucMember(id,"pConsole",0x84,0x20000400,-1,1);
        AddStrucMember(id,"tlsInUseBits1",0x88,0x20000400,-1,1);
        AddStrucMember(id,"tlsInUseBits2",0x8c,0x20000400,-1,1);
        AddStrucMember(id,"ProcessDWORD",0x90,0x20000400,-1,1);
        AddStrucMember(id,"ProcessGroup",0x94,0x20000400,-1,1);
        SetMemberComment(id,0x94,"struct _PROCESS_DATABASE *PrecessGroup\n= MODREF ?",0);
        AddStrucMember(id,"pExeMODREF",0x98,0x20000400,-1,1);
        AddStrucMember(id,"TopExcFilter",0x9c,0x20000400,-1,1);
        AddStrucMember(id,"BasePriority",0xa0,0x20000400,-1,1);
        AddStrucMember(id,"HeapOwnList",0xa4,0x20000400,-1,1);
        AddStrucMember(id,"HeapHandleBlockList",0xa8,0x20000400,-1,1);
        AddStrucMember(id,"pSomeHeapPtr",0xac,0x20000400,-1,1);
        AddStrucMember(id,"pConsoleProvider",0xb0,0x20000400,-1,1);
        AddStrucMember(id,"EnvironSelector",0xb4,0x10000400,-1,1);
        AddStrucMember(id,"ErrorMode",0xb6,0x10000400,-1,1);
        AddStrucMember(id,"peventLoadFinished",0xb8,0x20600400,-1,1);
        AddStrucMember(id,"UTState",0xbc,0x10000400,-1,1);

        id = GetStrucIdByName("_IMTE");
        AddStrucMember(id,"un1",0,0x20000400,-1,1);
        AddStrucMember(id,"pNTHdr",0x4,0x20000400,-1,1);
        SetMemberComment(id,0x4,"PIMAGE_NT_HEADERS",0);
        AddStrucMember(id,"field_8",0x8,0x20000400,-1,1);
        SetMemberComment(id,0x8,"un2",0);
        AddStrucMember(id,"pszFileName",0xc,0x20000400,-1,1);
        AddStrucMember(id,"pszModName",0x10,0x20000400,-1,1);
        AddStrucMember(id,"cbFileName",0x14,0x10000400,-1,1);
        AddStrucMember(id,"cbModName",0x16,0x10000400,-1,1);
        AddStrucMember(id,"field_18",0x18,0x20000400,-1,1);
        SetMemberComment(id,0x18,"un3",0);
        AddStrucMember(id,"cSections",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"field_20",0x20,0x20000400,-1,1);
        SetMemberComment(id,0x20,"un5",0);
        AddStrucMember(id,"baseAddress",0x24,0x20000400,-1,1);
        AddStrucMember(id,"hModule16",0x28,0x10000400,-1,1);
        AddStrucMember(id,"cUsage",0x2a,0x10000400,-1,1);
        AddStrucMember(id,"un7",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"pszFileName2",0x30,0x20000400,-1,1);
        AddStrucMember(id,"cbFileName2",0x34,0x10000400,-1,1);
        AddStrucMember(id,"pszModName2",0x36,0x20000400,-1,1);
        AddStrucMember(id,"cbModName2",0x3a,0x10000400,-1,1);

        id = GetStrucIdByName("_MODREF");
        AddStrucMember(id,"next",0,0x20000400,-1,1);
        AddStrucMember(id,"field_4",0x4,0x20000400,-1,1);
        AddStrucMember(id,"field_8",0x8,0x20000400,-1,1);
        AddStrucMember(id,"field_C",0xc,0x20000400,-1,1);
        AddStrucMember(id,"mteIndex",0x10,0x10000400,-1,1);
        AddStrucMember(id,"field_12",0x12,0x10000400,-1,1);
        AddStrucMember(id,"field_14",0x14,0x10000400,-1,1);
```

```
        SetMemberComment(id,0x14,"should be word,flag of something",0);
        AddStrucMember(id,"field_16",0x16,0x10000400,-1,1);
        AddStrucMember(id,"ppdb",0x18,0x20000400,-1,1);
        AddStrucMember(id,"field_1C",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"field_20",0x20,0x20000400,-1,1);
        AddStrucMember(id,"field_24",0x24,0x20000400,-1,1);

        id = GetStrucIdByName("R3TCB");
        AddStrucMember(id,"type",0,0x20000400,-1,1);
        AddStrucMember(id,"cReference",0x4,0x20000400,-1,1);
        AddStrucMember(id,"pProcess",0x8,0x20000400,-1,1);
        AddStrucMember(id,"someEvent",0xc,0x20000400,-1,1);
        AddStrucMember(id,"pvExcept",0x10,0x20000400,-1,1);
        AddStrucMember(id,"TopOfStack",0x14,0x20000400,-1,1);
        AddStrucMember(id,"StackLow",0x18,0x20000400,-1,1);
        AddStrucMember(id,"W16TDB",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"StackSelector16",0x1e,0x10000400,-1,1);
        AddStrucMember(id,"SelmanList",0x20,0x20000400,-1,1);
        SetMemberComment(id,0x20,"this is the PROCESSENTRY32 used by \nProcess32First\nProcess32N
        AddStrucMember(id,"UserPointer",0x24,0x20000400,-1,1);
        AddStrucMember(id,"pTIB",0x28,0x20000400,-1,1);
        AddStrucMember(id,"TIBflag",0x2c,0x10000400,-1,1);
        AddStrucMember(id,"Win16MutexCount",0x2e,0x10000400,-1,1);
        AddStrucMember(id,"DebugContext",0x30,0x20000400,-1,1);
        AddStrucMember(id,"pCurrentPriority",0x34,0x20000400,-1,1);
        AddStrucMember(id,"MessageQueue",0x38,0x20000400,-1,1);
        AddStrucMember(id,"pTLSArray",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"pProcess2",0x40,0x20000400,-1,1);
        AddStrucMember(id,"Flags",0x44,0x20000400,-1,1);
        AddStrucMember(id,"TerminationStatus",0x48,0x20000400,-1,1);
        AddStrucMember(id,"TIBselector",0x4c,0x10000400,-1,1);
        AddStrucMember(id,"EmulatorSelector",0x4e,0x10000400,-1,1);
        AddStrucMember(id,"cHandles",0x50,0x20000400,-1,1);
        AddStrucMember(id,"WaitNodeList",0x54,0x20000400,-1,1);
        AddStrucMember(id,"field_58",0x58,0x20000400,-1,1);
        AddStrucMember(id,"Ring0TCB",0x5c,0x20000400,-1,1);
        AddStrucMember(id,"pTDBX",0x60,0x20000400,-1,1);
        AddStrucMember(id,"StackBase",0x64,0x20000400,-1,1);
        AddStrucMember(id,"TerminationStack",0x68,0x20000400,-1,1);
        AddStrucMember(id,"EmulatorBase",0x6c,0x20000400,-1,1);
        AddStrucMember(id,"LastErrorCode",0x70,0x20000400,-1,1);
        AddStrucMember(id,"DebuggerCB",0x74,0x20000400,-1,1);
        AddStrucMember(id,"DebuggerThread",0x78,0x20000400,-1,1);
        AddStrucMember(id,"ThreadContext",0x7c,0x20000400,-1,1);
        AddStrucMember(id,"Except16List",0x80,0x20000400,-1,1);
        AddStrucMember(id,"ThunkConnect",0x84,0x20000400,-1,1);
        AddStrucMember(id,"NegStackBase",0x88,0x20000400,-1,1);
        AddStrucMember(id,"CurrentSS",0x8c,0x20000400,-1,1);
        AddStrucMember(id,"SStable",0x90,0x20000400,-1,1);
        AddStrucMember(id,"ThunkSS16",0x94,0x20000400,-1,1);
        AddStrucMember(id,"TLSarray",0x98,0x20000400,-1,64);
        AddStrucMember(id,"DeltaPriority",0x198,0x20000400,-1,1);
        AddStrucMember(id,"un5",0x19c,0x20000400,-1,7);
        AddStrucMember(id,"dSuspendCount",0x1b8,0x20000400,-1,1);
        SetMemberComment(id,0x1b8,"by ld. pCreateData16 ?",0);
        AddStrucMember(id,"APIsuspendCount",0x1bc,0x20000400,-1,1);
        AddStrucMember(id,"un6",0x1c0,0x20000400,-1,1);
        AddStrucMember(id,"WOWChain",0x1c4,0x20000400,-1,1);
        AddStrucMember(id,"wSSbig",0x1c8,0x10000400,-1,1);
        AddStrucMember(id,"un7",0x1ca,0x10000400,-1,1);
        AddStrucMember(id,"lp16SwitchRec",0x1cc,0x20000400,-1,1);
        AddStrucMember(id,"un8",0x1d0,0x20000400,-1,6);
        AddStrucMember(id,"pSomeCritSect1",0x1e8,0x20000400,-1,1);
        AddStrucMember(id,"pWin16Mutex",0x1ec,0x20000400,-1,1);
        AddStrucMember(id,"pWin32Mutex",0x1f0,0x20000400,-1,1);
        AddStrucMember(id,"pSomeCritSect2",0x1f4,0x20000400,-1,1);
        AddStrucMember(id,"un9",0x1f8,0x20000400,-1,1);
        AddStrucMember(id,"ripString",0x1fc,0x20000400,-1,1);
        AddStrucMember(id,"LastTlsSetValueEIP",0x200,0x20000400,-1,64);

        id = GetStrucIdByName("TDBX");
        AddStrucMember(id,"pTDB",0,0x20000400,-1,1);
```

```
        AddStrucMember(id,"pPDB",0x4,0x20000400,-1,1);
        AddStrucMember(id,"ContextHandle",0x8,0x20000400,-1,1);
        AddStrucMember(id,"un1",0xc,0x20000400,-1,1);
        AddStrucMember(id,"TimeOutHandle",0x10,0x20000400,-1,1);
        AddStrucMember(id,"WakeParam",0x14,0x20000400,-1,1);
        AddStrucMember(id,"BlockHandle",0x18,0x20000400,-1,1);
        AddStrucMember(id,"BlockState",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"SuspendCount",0x20,0x20000400,-1,1);
        AddStrucMember(id,"SuspendHandle",0x24,0x20000400,-1,1);
        AddStrucMember(id,"MustCompleteCount",0x28,0x20000400,-1,1);
        AddStrucMember(id,"WaitExFlags",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"SyncWaitCount",0x30,0x20000400,-1,1);
        AddStrucMember(id,"QueuedSyncFuncs",0x34,0x20000400,-1,1);
        AddStrucMember(id,"UserAPCList",0x38,0x20000400,-1,1);
        AddStrucMember(id,"KernAPCList",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"pPMPSPselector",0x40,0x20000400,-1,1);
        AddStrucMember(id,"BlockedOnID",0x44,0x20000400,-1,1);
        AddStrucMember(id,"un2",0x48,0x20000400,-1,7);
        AddStrucMember(id,"TraceRefData",0x64,0x20000400,-1,1);
        AddStrucMember(id,"TraceCallBack",0x68,0x20000400,-1,1);
        AddStrucMember(id,"TraceEventHandle",0x6c,0x20000400,-1,1);
        AddStrucMember(id,"TraceOutLastCS",0x70,0x10000400,-1,1);
        AddStrucMember(id,"K16TDB",0x72,0x10000400,-1,1);
        AddStrucMember(id,"K16PDB",0x74,0x10000400,-1,1);
        AddStrucMember(id,"DosPDBSeg",0x76,0x10000400,-1,1);
        AddStrucMember(id,"ExceptionCount",0x78,0x10000400,-1,1);

        id = GetStrucIdByName("EDB");
        AddStrucMember(id,"pszEnvironment",0,0x20000400,-1,1);
        AddStrucMember(id,"un1",0x4,0x20000400,-1,1);
        AddStrucMember(id,"pszCmdLine",0x8,0x20000400,-1,1);
        AddStrucMember(id,"pszCurrentDirectory",0xc,0x20000400,-1,1);
        AddStrucMember(id,"pStartupInfo",0x10,0x20000400,-1,1);
        AddStrucMember(id,"hStdIn",0x14,0x20000400,-1,1);
        AddStrucMember(id,"hStdOut",0x18,0x20000400,-1,1);
        AddStrucMember(id,"hStdError",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"Un2",0x20,0x20000400,-1,1);
        AddStrucMember(id,"InheritConsole",0x24,0x20000400,-1,1);
        AddStrucMember(id,"BreakType",0x28,0x20000400,-1,1);
        AddStrucMember(id,"BreakSem",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"BreakEvent",0x30,0x20000400,-1,1);
        AddStrucMember(id,"BreakThreadID",0x34,0x20000400,-1,1);
        AddStrucMember(id,"BreakHandles",0x38,0x20000400,-1,1);

        id = GetStrucIdByName("sysinfo");
        AddStrucMember(id,"dwOemID",0,0x20000400,-1,1);
        AddStrucMember(id,"dwPageSize",0x4,0x20000400,-1,1);
        AddStrucMember(id,"minAppAddress",0x8,0x20000400,-1,1);
        AddStrucMember(id,"maxAppAddress",0xc,0x20000400,-1,1);
        AddStrucMember(id,"dwActiveProcessMask",0x10,0x20000400,-1,1);
        AddStrucMember(id,"dwNumberOfProcessor",0x14,0x20000400,-1,1);
        AddStrucMember(id,"dwProcessorType",0x18,0x20000400,-1,1);
        AddStrucMember(id,"dwAllocationGran",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"wProcessorLevel",0x20,0x10000400,-1,1);
        AddStrucMember(id,"wProcessorRevision",0x22,0x10000400,-1,1);

        id = GetStrucIdByName("Critical_Section");
        AddStrucMember(id,"DebugInfo",0,0x20000400,-1,1);
        SetMemberComment(id,0,"PCritical_Section",0);
        AddStrucMember(id,"LockCount",0x4,0x20000400,-1,1);
        AddStrucMember(id,"RecursionCount",0x8,0x20000400,-1,1);
        AddStrucMember(id,"OwningThread",0xc,0x20000400,-1,1);
        AddStrucMember(id,"LockSemaphore",0x10,0x20000400,-1,1);
        AddStrucMember(id,"field_14",0x14,0x20000400,-1,1);

        id = GetStrucIdByName("pe32");
        AddStrucMember(id,"field_0",0,0x20000400,-1,1);
        AddStrucMember(id,"field_4",0x4,0x20000400,-1,1);
        AddStrucMember(id,"th32ProcessId",0x8,0x20000400,-1,1);
        AddStrucMember(id,"th32DefaultHeapId",0xc,0x20000400,-1,1);

        id = GetStrucIdByName("pinfo");
```

```
        AddStrucMember(id,"hProcess",0,0x20000400,-1,1);
        AddStrucMember(id,"hThread",0x4,0x20000400,-1,1);
        AddStrucMember(id,"dwProcessId",0x8,0x20000400,-1,1);
        AddStrucMember(id,"dwThreadId",0xc,0x20000400,-1,1);

        id = GetStrucIdByName("W16TDB");
        AddStrucMember(id,"tdb_next",0,0x10000400,-1,1);
        AddStrucMember(id,"tdb_taskSP",0x2,0x10000400,-1,1);
        AddStrucMember(id,"tdb_taskSS",0x4,0x10000400,-1,1);
        AddStrucMember(id,"tdb_nEvents",0x6,0x10000400,-1,1);
        AddStrucMember(id,"tdb_priority",0x8,0x000400,-1,1);
        AddStrucMember(id,"tdb_thread_ordinal",0x9,0x000400,-1,1);
        AddStrucMember(id,"tdb_thread_next",0xa,0x10000400,-1,1);
        AddStrucMember(id,"tdb_thread_tdb",0xc,0x10000400,-1,1);
        AddStrucMember(id,"tdb_thread_list",0xe,0x10000400,-1,1);
        AddStrucMember(id,"tdb_thread_free",0x10,0x10000400,-1,1);
        AddStrucMember(id,"tdb_thread_count",0x12,0x10000400,-1,1);
        AddStrucMember(id,"tdb_FCW",0x14,0x10000400,-1,1);
        AddStrucMember(id,"tdb_flags",0x16,0x10000400,-1,1);
        AddStrucMember(id,"tdb_errMode",0x18,0x10000400,-1,1);
        AddStrucMember(id,"tdb_ExpWinVer",0x1a,0x10000400,-1,1);
        AddStrucMember(id,"tdb_hInstance",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"tdb_hNE",0x1e,0x10000400,-1,1);
        AddStrucMember(id,"tdb_Queue",0x20,0x10000400,-1,1);
        AddStrucMember(id,"tdb_parent",0x22,0x10000400,-1,1);
        AddStrucMember(id,"field_24",0x24,0x000400,-1,48);
        AddStrucMember(id,"tdb_Rin3TCB",0x54,0x20000400,-1,1);
        AddStrucMember(id,"field_58",0x58,0x10000400,-1,1);
        AddStrucMember(id,"field_5A",0x5a,0x000400,-1,152);
        AddStrucMember(id,"tdb_modName",0xf2,0x000400,-1,8);
        AddStrucMember(id,"tdb_sig",0xfa,0x10000400,-1,1);
        SetMemberComment(id,0xfa,"'TD'",0);

        id = GetStrucIdByName("VMcb");
        AddStrucMember(id,"CB_VM_status",0,0x20000400,-1,1);
        AddStrucMember(id,"CB_High_Linear",0x4,0x20000400,-1,1);
        AddStrucMember(id,"CB_Client_Pointer",0x8,0x20000400,-1,1);
        AddStrucMember(id,"CB_VMID",0xc,0x20000400,-1,1);
        AddStrucMember(id,"CB_Signature",0x10,0x20000400,-1,1);
        AddStrucMember(id,"VM_Exec_Time",0x14,0x20000400,-1,1);
        AddStrucMember(id,"field_18",0x18,0x10000400,-1,1);
        AddStrucMember(id,"VM_ldt",0x1a,0x10000400,-1,1);
        AddStrucMember(id,"VM_MemBlock",0x28,0x20000400,-1,1);
        AddStrucMember(id,"VM_Next_VMHandle",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"field_30",0x30,0x10000400,-1,1);
        AddStrucMember(id,"VM_?Flags",0x48,0x20000400,-1,1);
        AddStrucMember(id,"field_4C",0x4c,0x000400,-1,-108);
        AddStrucMember(id,"CB_TCB?",0xfffffffe0,0x20000400,-1,1);
        SetMemberComment(id,0xfffffffe0,"-20",0);
        AddStrucMember(id,"CB_thread_of_VM",0xfffffffe4,0x20000400,-1,1);
        SetMemberComment(id,0xfffffffe4,"-1c",0);
        Debug version of VMcb
        id = GetStrucIdByName("VMcb");
        AddStrucMember(id,"CB_VM_status",0,0x20000400,-1,1);
        AddStrucMember(id,"CB_High_Linear",0x4,0x20000400,-1,1);
        AddStrucMember(id,"CB_Client_Pointer",0x8,0x20000400,-1,1);
        AddStrucMember(id,"CB_VMID",0xc,0x20000400,-1,1);
        AddStrucMember(id,"CB_Signature",0x10,0x20000400,-1,1);
        AddStrucMember(id,"VM_Exec_Time",0x18,0x10000400,-1,1);
        AddStrucMember(id,"VM_ldt",0x1a,0x10000400,-1,1);
        AddStrucMember(id,"VM_Event_List_Count",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"VM_LDTsel",0x1e,0x10000400,-1,1);
        AddStrucMember(id,"VM_VMM_Flags",0x20,0x20000400,-1,1);
        AddStrucMember(id,"VM_Event_List",0x24,0x20000400,-1,1);
        AddStrucMember(id,"VM_MMGR_flags",0x28,0x20000400,-1,1);
        AddStrucMember(id,"VM_LDTAddr",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"VM_List_Link",0x30,0x20000400,-1,1);
        AddStrucMember(id,"VM_Inst_Buf_ptr",0x38,0x20000400,-1,1);
        AddStrucMember(id,"VM_hVM",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"VM_quota_left",0x40,0x20000400,-1,1);
        AddStrucMember(id,"VM_?Flags",0x48,0x20000400,-1,1);
        AddStrucMember(id,"field_4C",0x4c,0x000400,-1,180);
```

```
        AddStrucMember(id,"field_100",0x100,0x000400,-1,-372);
        AddStrucMember(id,"VM_hheapVMCB",0xffffff8c,0x20000400,-1,1);
        SetMemberComment(id,0xffffff8c,"-74",0);
        AddStrucMember(id,"VM_App_Num",0xffffff96,0x10000400,-1,1);
        SetMemberComment(id,0xffffff96,"-6a",0);
        AddStrucMember(id,"VM_List_of_?",0xffffffbc,0x20000400,-1,1);
        SetMemberComment(id,0xffffffbc,"-44",0);
        AddStrucMember(id,"VM_PM_APP_CB",0xffffffc0,0x20000400,-1,1);
        SetMemberComment(id,0xffffffc0,"-40",0);
        AddStrucMember(id,"VM_LDTcpg",0xffffffc8,0x20000400,-1,1);
        AddStrucMember(id,"VM_hTimeOut",0xffffffd4,0x20000400,-1,1);
        SetMemberComment(id,0xffffffd4,"-2c",0);
        AddStrucMember(id,"VM_Ring0TCB",0xffffffdc,0x20000400,-1,1);
        SetMemberComment(id,0xffffffdc,"-24",0);
        AddStrucMember(id,"VM_ThreadListHead",0xffffffe0,0x20000400,-1,1);
        SetMemberComment(id,0xffffffe0,"-20",0);
        AddStrucMember(id,"VM_TS_Sched_Count",0xffffffe4,0x20000400,-1,1);
        SetMemberComment(id,0xffffffe4,"-1c",0);
        AddStrucMember(id,"VM_SuspendCount",0xffffffe8,0x20000400,-1,1);
        SetMemberComment(id,0xffffffe8,"-18",0);
        AddStrucMember(id,"VM_ppteVM",0xffffffec,0x20000400,-1,1);
        AddStrucMember(id,"VM_pdeVM",0xfffffff0,0x20000400,-1,1);
        SetMemberComment(id,0xfffffff0,"-10",0);
        AddStrucMember(id,"VM_Int_Enable_Count",0xfffffff8,0x20000400,-1,1);

        id = GetStrucIdByName("ClientReg");
        AddStrucMember(id,"Client_EDI",0,0x20000400,-1,1);
        AddStrucMember(id,"Client_ESI",0x4,0x20000400,-1,1);
        AddStrucMember(id,"Client_EBP",0x8,0x20000400,-1,1);
        AddStrucMember(id,"Client_EBX",0x10,0x20000400,-1,1);
        AddStrucMember(id,"Client_EDX",0x14,0x20000400,-1,1);
        AddStrucMember(id,"Client_ECX",0x18,0x20000400,-1,1);
        AddStrucMember(id,"Client_EAX",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"Client_ECODE",0x20,0x20000400,-1,1);
        AddStrucMember(id,"Client_EIP",0x24,0x20000400,-1,1);
        AddStrucMember(id,"Client_CS",0x28,0x20000400,-1,1);
        AddStrucMember(id,"Client_EFlags",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"Client_ESP",0x30,0x20000400,-1,1);
        AddStrucMember(id,"Client_SS",0x34,0x20000400,-1,1);
        AddStrucMember(id,"Client_ES",0x38,0x20000400,-1,1);
        AddStrucMember(id,"Client_DS",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"Client_FS",0x40,0x20000400,-1,1);
        AddStrucMember(id,"Client_GS",0x44,0x20000400,-1,1);

        id = GetStrucIdByName("CltRegW");
        AddStrucMember(id,"__DI",0,0x10000400,-1,1);
        AddStrucMember(id,"__SI",0x4,0x10000400,-1,1);
        AddStrucMember(id,"__BP",0x8,0x10000400,-1,1);
        AddStrucMember(id,"__BX",0x10,0x10000400,-1,1);
        AddStrucMember(id,"__DX",0x14,0x10000400,-1,1);
        AddStrucMember(id,"__CX",0x18,0x10000400,-1,1);
        AddStrucMember(id,"__AX",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"__CS",0x24,0x10000400,-1,1);
        AddStrucMember(id,"__IP",0x28,0x10000400,-1,1);
        AddStrucMember(id,"__Flag",0x2c,0x10000400,-1,1);
        AddStrucMember(id,"__SP",0x30,0x10000400,-1,1);
        AddStrucMember(id,"__SS",0x34,0x10000400,-1,1);
        AddStrucMember(id,"__ES",0x38,0x10000400,-1,1);
        AddStrucMember(id,"__DS",0x3c,0x10000400,-1,1);
        AddStrucMember(id,"__FS",0x40,0x10000400,-1,1);
        AddStrucMember(id,"__GS",0x44,0x10000400,-1,1);

        id = GetStrucIdByName("R0TCB");
        AddStrucMember(id,"TCB_Flags",0,0x20000400,-1,1);
        AddStrucMember(id,"TCB_signature",0xc,0x20000400,-1,1);
        AddStrucMember(id,"TCB_ClientPtr",0x10,0x20000400,-1,1);
        AddStrucMember(id,"TCB_VMHandle",0x14,0x20000400,-1,1);
        AddStrucMember(id,"TCB_ThreadID",0x18,0x10000400,-1,1);
        AddStrucMember(id,"TCB_PMLockOrigSS",0x1a,0x10000400,-1,1);
        AddStrucMember(id,"TCB_PMLockOrigESP",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"TCB_PMLockOrigEIP",0x20,0x20000400,-1,1);
        AddStrucMember(id,"TCB_PMLockStackCount",0x24,0x20000400,-1,1);
```

```
        AddStrucMember(id,"TCB_PMLockOrigCS",0x28,0x10000400,-1,1);
        AddStrucMember(id,"TCB_PMPSPSelector",0x2a,0x10000400,-1,1);
        AddStrucMember(id,"TCB_ThreadType",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"TCB_pad1",0x30,0x10000400,-1,1);
        AddStrucMember(id,"TCB_pad2",0x32,0x000400,-1,1);
        AddStrucMember(id,"TCB_extErrLocus",0x33,0x000400,-1,1);
        AddStrucMember(id,"TCB_extErr",0x34,0x10000400,-1,1);
        AddStrucMember(id,"TCB_extErrAction",0x36,0x000400,-1,1);
        AddStrucMember(id,"TCB_extErrClass",0x37,0x000400,-1,1);
        AddStrucMember(id,"TCB_extErrPtr",0x38,0x20000400,-1,1);
        AddStrucMember(id,"TCB_ExecPriority",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"TCB_Next",0x40,0x20000400,-1,1);
        AddStrucMember(id,"TCB_fEvent",0x44,0x20000400,-1,1);
        AddStrucMember(id,"TCB_?48",0x48,0x20000400,-1,1);
        SetMemberComment(id,0x48,"24cb",0);
        AddStrucMember(id,"field_4C",0x4c,0x000400,-1,28);
        AddStrucMember(id,"TCB_?68",0x68,0x10000400,-1,1);
        SetMemberComment(id,0x68,"1d6a",0);
        AddStrucMember(id,"field_6A",0x6a,0x000400,-1,-182);
        AddStrucMember(id,"TCB_ExecTime",0xffffffb4,0x20000400,-1,1);
        SetMemberComment(id,0xffffffb4,"-4c",0);
        AddStrucMember(id,"field_FFFFFFB8",0xffffffb8,0x000400,-1,21);
        AddStrucMember(id,"TCB_mustComplete",0xffffffcd,0x000400,-1,1);
        SetMemberComment(id,0xffffffcd,"-33",0);

        id = GetStrucIdByName("R0Context");
        AddStrucMember(id,"PGTPTR",0,0x20000400,-1,1);
        AddStrucMember(id,"Tables",0x4,0x20000400,-1,1);
        AddStrucMember(id,"pStruc1",0x8,0x20000400,-1,1);
//      AddStrucMember(id,"Context_oMinMax",0x8,0x20000400,-1,1);
        AddStrucMember(id,"Next",0xc,0x20000400,-1,1);
        AddStrucMember(id,"Mutex",0x10,0x20000400,-1,1);

        id = GetStrucIdByName("NE");
        AddStrucMember(id,"ne_signature",0,0x10000400,-1,1);
        AddStrucMember(id,"ne_cref",0x2,0x10000400,-1,1);
        AddStrucMember(id,"ne_npEntryTable",0x4,0x10000400,-1,1);
        AddStrucMember(id,"ne_Next",0x6,0x10000400,-1,1);
        AddStrucMember(id,"ne_FileInfo",0xa,0x10000400,-1,1);
        AddStrucMember(id,"ne_segIndex",0xe,0x10000400,-1,1);
        AddStrucMember(id,"ne_cseg",0x1c,0x10000400,-1,1);
        AddStrucMember(id,"ne_cModules",0x1e,0x10000400,-1,1);
        AddStrucMember(id,"ne_cbNonResNamesTab",0x20,0x10000400,-1,1);
        AddStrucMember(id,"ne_SegTab",0x22,0x10000400,-1,1);
        AddStrucMember(id,"ne_ModuleName",0x26,0x10000400,-1,1);

        id = GetStrucIdByName("WND32");
        AddStrucMember(id,"hWndNext",0,0x20000400,-1,1);
        SetMemberComment(id,0,"W95SPS p172",0);
        AddStrucMember(id,"hWndChild",0x4,0x20000400,-1,1);
        AddStrucMember(id,"hWndParent",0x8,0x20000400,-1,1);
        AddStrucMember(id,"hWndOwner",0xc,0x20000400,-1,1);
        AddStrucMember(id,"rectWindow",0x10,0x10000400,-1,4);
        AddStrucMember(id,"rectClient",0x18,0x10000400,-1,4);
        AddStrucMember(id,"hQueue",0x20,0x10000400,-1,1);
        AddStrucMember(id,"hrgnUpdate",0x22,0x10000400,-1,1);
        AddStrucMember(id,"wndClass",0x24,0x10000400,-1,1);
        AddStrucMember(id,"hInstance",0x26,0x10000400,-1,1);
        AddStrucMember(id,"lpfnWndProc",0x28,0x20000400,-1,1);
        AddStrucMember(id,"dwFlags",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"dwStyleFlags",0x30,0x20000400,-1,1);
        AddStrucMember(id,"dwExStyleFlags",0x34,0x20000400,-1,1);
        AddStrucMember(id,"moreFlags",0x38,0x20000400,-1,1);
        AddStrucMember(id,"ctrlID",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"windowTextOffset",0x40,0x10000400,-1,1);
        AddStrucMember(id,"scrollBar",0x42,0x10000400,-1,1);
        AddStrucMember(id,"properties",0x44,0x10000400,-1,1);
        AddStrucMember(id,"hWnd16",0x46,0x10000400,-1,1);
        AddStrucMember(id,"lastActive",0x48,0x20000400,-1,1);
        AddStrucMember(id,"hMenuSystem",0x4c,0x20000400,-1,1);
        AddStrucMember(id,"classAtom",0x56,0x10000400,-1,1);
        AddStrucMember(id,"alternatePID",0x58,0x20000400,-1,1);
```

```
        AddStrucMember(id,"alternateTID",0x5c,0x20000400,-1,1);

        id = GetStrucIdByName("CONTEXT");
        AddStrucMember(id,"ContextFlags",0,0x20000400,-1,1);
        AddStrucMember(id,"Dr0",0x4,0x20000400,-1,1);
        AddStrucMember(id,"Dr1",0x8,0x20000400,-1,1);
        AddStrucMember(id,"Dr2",0xc,0x20000400,-1,1);
        AddStrucMember(id,"Dr3",0x10,0x20000400,-1,1);
        AddStrucMember(id,"Dr6",0x14,0x20000400,-1,1);
        AddStrucMember(id,"Dr7",0x18,0x20000400,-1,1);
        AddStrucMember(id,"Floating_save_area",0x1c,0x000400,-1,112);
        AddStrucMember(id,"SegGs",0x8c,0x20000400,-1,1);
        AddStrucMember(id,"SegFs",0x90,0x20000400,-1,1);
        AddStrucMember(id,"SegEs",0x94,0x20000400,-1,1);
        AddStrucMember(id,"SegDs",0x98,0x20000400,-1,1);
        AddStrucMember(id,"Edi",0x9c,0x20000400,-1,1);
        AddStrucMember(id,"Esi",0xa0,0x20000400,-1,1);
        AddStrucMember(id,"Ebx",0xa4,0x20000400,-1,1);
        AddStrucMember(id,"Edx",0xa8,0x20000400,-1,1);
        AddStrucMember(id,"Ecx",0xac,0x20000400,-1,1);
        AddStrucMember(id,"Eax",0xb0,0x20000400,-1,1);
        AddStrucMember(id,"Ebp",0xb4,0x20000400,-1,1);
        AddStrucMember(id,"Eip",0xb8,0x20000400,-1,1);
        AddStrucMember(id,"SegCs",0xbc,0x20000400,-1,1);
        AddStrucMember(id,"EFlags",0xc0,0x20000400,-1,1);
        AddStrucMember(id,"Esp",0xc4,0x20000400,-1,1);
        AddStrucMember(id,"SegSs",0xc8,0x20000400,-1,1);

        id = GetStrucIdByName("DemandInfo");
        AddStrucMember(id,"DILin_Total_Count",0,0x20000400,-1,1);
        AddStrucMember(id,"DIPhys_Count",0x4,0x20000400,-1,1);
        AddStrucMember(id,"DIFree_Count",0x8,0x20000400,-1,1);
        AddStrucMember(id,"DIUnlock_Count",0xc,0x20000400,-1,1);
        AddStrucMember(id,"DILinear_BaseAddr",0x10,0x20000400,-1,1);
        AddStrucMember(id,"DILin_Total_Free",0x14,0x20000400,-1,1);
        AddStrucMember(id,"DIPage_Faults",0x18,0x20000400,-1,1);
        AddStrucMember(id,"DIPage_Ins",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"DIPage_Outs",0x20,0x20000400,-1,1);
        AddStrucMember(id,"DI_PageDiscards",0x24,0x000400,-1,1);
        AddStrucMember(id,"DIInstance_Faults",0x25,0x20000400,-1,1);
        AddStrucMember(id,"DIPagingFileMax",0x29,0x20000400,-1,1);
        AddStrucMember(id,"DIPagingFileInUse",0x2d,0x20000400,-1,1);
        AddStrucMember(id,"DICommit_Count",0x31,0x20000400,-1,1);
        AddStrucMember(id,"DIReserved",0x35,0x20000400,-1,2);

        id = GetStrucIdByName("DDB");
        AddStrucMember(id,"DDB_Next",0,0x20000400,-1,1);
        AddStrucMember(id,"DDB_SDK_version",0x4,0x10000400,-1,1);
        AddStrucMember(id,"DDB_Device_ID",0x6,0x10000400,-1,1);
        AddStrucMember(id,"DDB_Version",0x8,0x10000400,-1,1);
        AddStrucMember(id,"DDB_Flags",0xa,0x10000400,-1,1);
        AddStrucMember(id,"DDB_Device_Name",0xc,0x000400,-1,8);
        AddStrucMember(id,"DDB_Init_Order",0x14,0x20000400,-1,1);
        AddStrucMember(id,"DDB_Control_Proc",0x18,0x20000400,-1,1);
        AddStrucMember(id,"DDB_V86_API_Proc",0x1c,0x20000400,-1,1);
        AddStrucMember(id,"DDB_PM_API_Proc",0x20,0x20000400,-1,1);
        AddStrucMember(id,"DDB_V86_API_csip",0x24,0x20000400,-1,1);
        AddStrucMember(id,"DDB_PM_API_csip",0x28,0x20000400,-1,1);
        AddStrucMember(id,"DDB_Reference_Data",0x2c,0x20000400,-1,1);
        AddStrucMember(id,"DDB_VxDServiceTablePtr",0x30,0x20000400,-1,1);
        AddStrucMember(id,"DDB_VxDServiceTableSize",0x34,0x20000400,-1,1);
        AddStrucMember(id,"DDB_Win32_VxD_Service_Table",0x38,0x20000400,-1,1);
        AddStrucMember(id,"DDB_Prev",0x3c,0x20000400,-1,1);
        AddStrucMember(id,"DDB_Size",0x40,0x20000400,-1,1);

        id = GetStrucIdByName("PMcb");
        AddStrucMember(id,"PMcb_Flags",0,0x20000400,-1,1);
        AddStrucMember(id,"PMcb_parent",0x4,0x20000400,-1,1);
        AddStrucMember(id,"PMcb_PSP",0x8,0x10000400,-1,1);
        AddStrucMember(id,"PMcb_signature",0xa,0x10000400,-1,1);
        SetMemberComment(id,0xa,"0cbb0",0);
```

```
        id = GetStrucIdByName("PUSHA");
        AddStrucMember(id,"_EDI",0,0x20000400,-1,1);
        AddStrucMember(id,"_ESI",0x4,0x20000400,-1,1);
        AddStrucMember(id,"_EBP",0x8,0x20000400,-1,1);
        AddStrucMember(id,"_EBX",0x10,0x20000400,-1,1);
        AddStrucMember(id,"_EDX",0x14,0x20000400,-1,1);
        AddStrucMember(id,"_ECX",0x18,0x20000400,-1,1);
        AddStrucMember(id,"_EAX",0x1c,0x20000400,-1,1);
```

obfuscator

```
In Windows 95, the GetCurrentProcessID() and GetCurrentThreadID()
functions return "obfuscated" IDs.  KERNEL32 takes the actual address
of the data structure and XORs it with a magic number to generate the
value returned from GetCurrentXXXID():

KERNEL32!GetCurrentProcessId:
    mov     eax, [PTR_CURRENT_PROCESS_ID]
    push    dword ptr [eax]
    call    OBFUSCATOR
    ret
OBFUSCATOR:
    mov     eax, [MAGIC_XOR_PATTERN]    ; e.g., 7EAE7049h
    test    eax, eax
    jnz     do_xor
    xor     eax, eax
    jmp     do_ret
do_xor:
    xor     eax, [esp+4]
do_ret:
    ret     4
```

```
static DWORD GetXORPattern(char *funcname)
{
    #define GET_PROC(mod, func) \
        GetProcAddress(GetModuleHandle(mod), (func))
    DWORD (WINAPI *func)(VOID) = GET_PROC("KERNEL32", funcname);
    BYTE *p = (BYTE *) func;
    DWORD id = (*func)();
    DWORD cb, ***pppcb;
    #define PUSH_CODE   0x6A
    #define MOV_CODE    0xA1
    if (*p == PUSH_CODE)
        p += 7;         // adjust for debug kernel: PUSH 1Ah, CALL XXX
    if (*p != MOV_CODE)
        return 0;       // something wrong!
    p ++;               // skip past MOV to [XXXXXXXX]
    pppcb = (DWORD ***) p;
    cb = ***pppcb;
    return (id ^ cb);
}
-------------------
    DWORD               Unobsfucator = 0;
```

```
void InitUnobsfucator()
{
    DWORD   tid = GetCurrentThreadId();
    __asm {
          mov     eax, 18h
          mov     eax, fs:[eax]
          sub     eax, 10h
          xor     eax,[tid]
          mov     [Unobsfucator], eax
          }
}
```

Cracking Interactive Disassembler Pro v3.7

Interactive Disassembler Pro v3.7 Demo

This is a brand new version which is much superior to version 3.6.

Source:

http://www.datarescue.com/ida.htm (homepage)

http://www.datarescue.com/ida/demo37.zip (9,884,100 bytes)

Tools used:

W32DASM 8.9 (soon to be a thing of the past :-)

BoundsChecker Pro 5.0 (look for the poorly protected demo on NuMega's site)

SoftICE for NT v3.2 (that new video driver is amazing)

(if you can't get Pinnacle's version to install, change the byte at 2B6C in setup.ins from 2B to B8)

Ultraedit32 (just for multifile searching)

HexWorkshop32 (any hex editor will do the trick-including Ultraedit32)

Sections in this article:

APPENDIX Stack frames and function calls

I. Why IDA Pro 3.7 is so great

IDA Pro is by far the best disassembler available for PCs (and probably for any platform). It is ultimately, I think, much, much better than W32dasm. Why? To begin with, IDA Pro disassembles properly. (1) It starts disassembling at the program entry point and then follows every possible execution route from there. Having done that, it then looks for functions which are not directly accessible from the main program flow (e.g., window procedures, thread procedures, and other callback functions of various sorts). This method of disassembly enables to perform much greater levels of analysis of the target program. For example, the beginning and ending of functions are identified and properly marked. Passed arguments and local variables (both referenced off the stack) are identified and marked. Switch statements are identified and the case values are determined (W32dasm does this to a very limited extent). Furthermore, you have complete decision over what it marks as code and what it marks as data. This allows it to disassemble code "hidden" or located in the data section, which happens more often than you might think, because W32dasm can't disassemble it.
Also, a trick I saw in a dongle driver (ssidppd.drv
from the program WiT) complete flusters disassemblers like w32dasm, which disassemble blindly straight through the code segment. Here's the trick:
ANTI-WDASM trick

```
        mov eax, edx
```

```
        jmp loc_1
        db 0F
loc_1:  inc eax
        jmp loc_2
        db 85
loc_2:  call sub_1
... and so on
```

w32dasm produces garbage for this code, but IDA Pro does it right because it's following the path of execution. (2) IDA can recognize an amazing range of library functions within a target's body. This greatly reduces the amount of code to plow through when trying to get an understanding of the target. It also, of course, provides a wealth of clues about what a program is doing at any particular place. (3) IDA also has a fairly robust macro language which enables high levels of customization. (4) It can disassemble damn near anything: all pc-based binary formats (all exe formats, lib files, obj files, etc.), code for almost any microprocessor (several of which I've never heard of), Java classes --- that's right full blown interactive Java disassembly, which, as fravia+ says, is the future of cracking! (5) Names, labels, etc. can be changed on the fly so that you can gradually accumulate and save more and more information about the target (working the fields, as I like to think of it). (6) Comments can be added. There are undoubtedly more features that I am leaving out, but you get the idea. In other words, this is __the__ disassembling tool for our trade. Combined with SoftICE and BoundsChecker (more on the power of BoundsChecker below), no target is safe any more (not that they had been safe howhever :-)

II. What's disabled in the demo version

(1) It expires on Jan 1, 1998

(2) It cannot load files larger than 64k

(3) It cannot load saved databases (project files)

(4) It cannot produce list (.lst) or asm source files

Limitation

(1) will prove, as you might have expected, to be trivial to crack.

(2) is the biggest issue. The size limitation makes the demo almost useless as it is. Cracking this limitation is the main objective for this discussion.

(3) would be very nice to crack, but perhaps long and hard if the loading code is simply not present in the demo (I think, however, that it is).

There are two big problems this limitation (3) creates:

(a) IDA's power comes at a price---it's slow, so it would be nice not to have to have it redisassemble with every session.

(b) Any comments or name changes, etc. will not be accessible after you quit a session.

Problem (b), fortunately, is avoidable, because the demo will let you create a macro file (a file with an idc extension) that records all of your changes which can be reloaded and run for future sessions.

(4) I consider to be the least important because looking at a dumb listing in a text editor is a pale comparison to viewing it in the

IDA environment. It might be fun, though, to try to produce compilable asm files from targets. Now, on to the good stuff.

III. Cracking the 64k file size limitation

I will be concerned with the win32 version of IDA. This version is run by executing the idaw.exe file. This file is quite small and does nothing more than load ida.wll [sic], which is a large dll that does most of the work. This is the file that needs cracking. The other files with w32 extensions are the various disassembler modules. pc.w32 is the modules to handle x86 code. When you start idaw, it loads ida.wll (hereinafter simply referred to as ida), which asks you which file you want to disassemble. ida determines which module to load based on the file type and then hands control over to that module. The disassembly module then drives execution for the rest of the session, calling functions in ida where necessary. This is a brilliant design, because it allows the programmer to quite easily add modules for completely different file types without having to rewrite the whole program (witness the Java module. One would think Java disassembly is so different from x86 disassembly that the two could not happily co-exist in one environment). ida is responsible for loading the file and for all subsequent file i/o along with a lot of other things, so let's start by loading ida.wll into w32dasm (I spoke harshly of w32dasm above and that was unfair. I do have a fondness for it---however, now that I have a useable IDA Pro, I will never go back to it :-).

The first thing to do is find the place where IDA decides that a file

bigger 64k is too big. Our first bet might be to look for the text of the message that pops up when you try to open a file that is too big: "The demo version..." Unfortunately it is not in the list of strings w32dasm provides for ida.wll. So, I used UltraEdit32 to do a file search through the entire idademo directory for the string in question. It turns out to be in a file called ida.hlp (which is not a Windows help file, but is in a proprietary format).

Looking at ida.hlp with a hex editor (it's not strictly an ASCII file) we see that the strings are zero-terminated and appear to be prefixed with a word which gives their length. Also, at the beginning of the file is a long series of dwords that appear to be offsets into the file. You guessed it, the dwords point to the length/string pairs. This is undoubtedly how the program gets at the strings. There are D (i.e., 13 (decimal)) bytes at the beginning of the file before the dword list begins. So, the index to a particular string can be calculated in the following way:

1. find the string in ida.hlp and record the offset where the length word starts
2. look up the offset in the list of dwords at the beginning of the file and record the offset of the dword.
3. Subtract D from the offset of the dword and divide the result by 4
4. What you end up with is the index ida uses to reference a string in the help file.

The index for the file-too-big message is 556. Using w32dasm to search for this value in ida.wll, we find the following code:

```
:00403CE7 81FD00000100          cmp ebp, 00010000 ; cmp file size,64k
:00403CED 760B                  jbe 00403CFA ; go ahead and load file
```

```
:00403CEF 6856050000             push 00000556 ; ida.hlp index of demo msg
:00403CF4 E85FD10500             call 00460E58 ; message box routine
```

This looks too good to be true (don't worry - it is). It compares ebp (which must contain the size of the file) with 10000 (i.e., 64k) and jumps past the bad message if ebp is less than 64k. No problem. Let's patch the program to force the jump, replacing 76 at 403CED with EB (jmp) and see what happens.

Running the program, we find that it now lets us open files of any size and it goes about disassembling them. The problem is that it appears to be disassembling only a small part of the file (I'm using ida.wll, by the way, as the large test file to disassemble). Fairly quickly into the disassembly the message "Execution flows beyond limits" repeatedly appears at the bottom of the screen and nothing past (what appears to be) the 64k boundary is disassembled. In fact, nothing past the 64k boundary is even represented by raw bytes in the disassembly listing. There's another check somewhere.

I was stuck at this point for some time. I tried using the help file method above to search out references to "Execution flows beyond limits", but, while I found one, no hacking around in that area of the program seemed to help. It then occurred to me that maybe ida never even loaded more than 64k of the file. However, that couldn't be right because it would load the entire DATA segment for ida.wll, which is well past the 64k mark. Maybe it only loaded 64k of each segment. To investigate this, I ran IDA with BoundsChecker in order to look at how much of the file was actually being read in.

So, fire up BoundsChecker (other API spies will probably work, but they won't give you the wealth of information BC does), and load idaw.exe. In the program settings, be sure to set it to collect all event data and to load the module ida.wll. Run the program from BC and open ida.wll from ida. Let it run for a while (at least until you start getting the "Execution flows beyond limits" messages), and then quit ida. You've got one hell of a lot of API calls recorded in BC. In BC, search for calls to CreateFile (which, remember also opens files in Win32) until you find one that passes "ida.wll" as the file name. The return value from CreateFile is the handle to ida.wll, so write that down and start searching for the handle (this will catch all API calls having to do with ida.wll). You'll come across a whole bunch of calls to SetFilePointer and to ReadFile. A lot of these calls set the file pointer to places in the PE Header and read 200 bytes. Forget about these-it's just reading in relevant info about the file. Eventually, though, you'll hit a call the sets the pointer to the beginning of the code segment and reads 7A00 bytes, and then another that sets the pointer to the beginning of the data segment and reads DE00 bytes. 600 is the offset to the beginning of the code segment and 88000 is the offset to the DATA segment. Why is it reading 7A00 bytes instead of 10000 bytes? We'll answer that question in a moment. Write down the location in ida.wll (4316DC) that called ReadFile (this can be found in the right hand pane in BC --- isn't BC great? NuMega wins again) and switch over to w32dasm to see what's going on there.

It was in switching over to w32dasm at this point that I had a bit of dumb luck (dumb because I should have figured this out rather than stumbling across it). W32dasm happened to be positioned at the top of

the file where it tells you the segment information. Guess what the size of the code segment is? 87A00. The size of the data segment? DE00. So, it's only taking the low 2 bytes of the segment sizes to get the number of bytes to read in from the file. That's why all of the DATA segment but only part of the CODE segment is read. This can be verified if you load ida.wll into ida and jump to address 408A00. At exactly that address, the code cuts off and you get question marks instead of data/code.

Now, there are two ways to proceed. The quick and easy way versus the methodical way. I naturally first opted for the quick and easy way which is to search through the w32dasm listing for 0000FFFF and'ed with something else (searching for ",{space}0000FFFF" is a sufficiently narrow choice). This, I assumed, was what the program did to prevent longer files from being read. While I found some places where there were such and instructions, nothing panned out. I won't bore you with the mess I created fiddling around with code in these sections. IDA is designed to disassemble 16 bit programs as well as others, so 64k is relevant to it for many other reasons than the protection scheme (the length of a segment in 16 bit mode is 64k), so there are a lot of 64k red herrings in the program. Well, the quick and easy way wasn't quick or easy and didn't work. I'm impatient (a bad quality in a cracker), but it's always better to try to understand what the code is doing, rather than trying to get lucky. If you're methodical, the luck will find you. Time for SoftICE.

The strategy now is to set a breakpoint in SI at the ReadFile call, with the condition that the 3rd parameter be equal to 7A00 (otherwise we'd have to wade through the hundreds of other 200 byte calls). This

can be done with the following command:

```
bpx 4316DC if *(esp+8)==7a00
```

Load idaw.exe into SoftICE, set the breakpoint, and load ida.wll into IDA. When the bp hits, we need to trace the 7A00 back until we find the point at which it was changed from 87A00. This will be done by unwinding the stack within softice, which I will explain as we go along.

So, the breakpoint has hit and we're sitting in a function which begins at 4316AC. Looking at the code, we know that 7A00 came into this function as the third argument passed on the stack --- [ebp+10] (see the Appendix to this article for a discussion of parameter passing in C and C++ programs). The strategy is to find the function that called the one we're in and figure out where it got the value 7A00 that it passed to the function we're in. We continue to apply this method, walking back through the call stack until we find out how 87A00 got to be 7A00. Here's how it works.

Use the command dd esp to display the memory at the top of the stack. The first address within the program's code starting from the stack top and going forward (i.e., higher) in memory is the return address for the call to the function we're in. Disassemble from that address and scroll up a little. Directly above the address from which you disassembled you'll see the call instruction. So, 4316AC was called by the function at 43088C:

```
:004308CD 8B4D10        mov ecx, dword ptr [ebp+10] ; 3rd arg passed to this
```

```
                                                    ; func == 7A00
:004308D0 51            push ecx    ; arg3 to 004316AC
:004308D1 50            push eax    ; arg2 to 004316AC
:004308D2 8B4508        mov eax, dword ptr [ebp+08]
:004308D5 50            push eax    ; arg1 to 004316AC
:004308D6 E8D10D0000    call 004316AC
```

>From this code we see that 7A00 came in as the 3 rd argument passed to 43088C. So, who called 43088C. Look a little further up the stack (which you should keep in the SoftICE data window) to find a call at 42EB4C, which is in the function that starts at 42EAD4. Keep following the same stack tracing method until you hit a call at 43A395 to a function at 42EC04 (it should only be 2 more steps). 43A395 is in the function that starts at 43A314 and this where we hit pay-dirt. I have included almost the entire function below because what happens here is very interesting. I've commented many of the lines and will also refer to the code as I continue. At 43A389, we notice that the local variable [ebp-18] is passed as the relevant argument to 43088C.
Therefore, it contains 7A00. Let's trace backwards from 43A389 and see how [ebp-18] gets its value. It comes in through the cx register, which of course can only hold a 16 bit (i.e., <=64k) value. There's the key to our problem. One more trace back up the stack will take us to the conversion from 87A00 to 7A00. We find the call to 43A314 at 48A836 and looking at the code fragment immediately below we can see where the conversion occurs:

```
:0043A822 8BD7          mov edx, edi
:0043A824 8BC3          mov eax, ebx
```

```
:0043A826 E885FFFFFF      call 0043A7B0
:0043A82B 53              push ebx  ; == start addr of segment == 401000
:0043A82C 8BCF            mov ecx, edi  ; == end addr of segment == 488A00
:0043A82E 662BCB          sub cx, bx ; ==seg length== 7A00 this is our bad guy
:0043A831 8BD6            mov edx, esi
:0043A833 8B45FC          mov eax, dword ptr [ebp-04]
:0043A836 E8D9FAFFFF      call 0043A314 ; cx and edx are passed to this func
```

Now we have to figure out how to crack it. That means we've got to change a 16 bit data type into a 32 bit data type in the functions at 48A804 and 48A314. In 48A804 it's not so hard. sub cx, bx needs to be sub ecx, ebx. This can be done by changing 66 to 90 (nop) at 43A82E. 66 is the opcode prefix representing an operand size override.

A brief digression

Intel chips can operate in 32 bit or 16 bit mode.
Win32 programs run in 32 bit mode and therefore default to accessing the 32 bit registers and 32 bit memory operands.
However, an instruction in 32 bit mode with the operand size override prefix accesses 16 bit registers and 16 bit memory operands.
End digression.

So, changing 66 to a nop (90) switches the instruction back to accessing 32 bit registering. This strategy is very useful for 48A314 as well. Turning our attention to that function, however, we see that things are much more complicated. The instruction at 43A31D has an opsize prefix, but it accessing a memory location ([ebp-06]) as well as a register. We're going to need 2 more bytes of memory from sonewhere. [ebp-06] is a local variable and we can't have it writing over other local variables. So, let's lay out the local

variables on the stack for this function:

```
ebp          : prev ebp
ebp-04       : local_1 (4 bytes)
ebp-06       : our little friend, local_2 (2 bytes)
ebp-0C       : 10000 (see function code below) (4 bytes)
```

and so on.

ebp-0C is a dword pointer and therefore only takes up 4 bytes. It stops at ebp-08. What is there at ebp-08? [ebp-08] is never referenced in the function, so it looks like there's nothing there. Guess what we're going to put there, though? You got it, the rest of the segment length. This can be done by changing all the references to ebp-06 to ebp-08. So, our instruction at 43A31D needs to change from:

```
:0043A31D 66894DFA        mov word ptr [ebp-06], cx
```

to:

```
:0043A31D 90              nop
:0043A31E 894DF8          mov dword ptr [ebp-08], ecx
```

Removing the 66 takes care of changing cx to ecx and word ptr to dword ptr, while changing FA to F8 changes the ebp offset. The same strategy can be applied at 43A35E, 43A3ED, and 43A414. The instructions at 43A369 and 43A377 also need to be changed. Movzx means move with sign extended and is used for moving smalling operands into larger operands while preserving the sign. We want to change

these to simple move instructions, moving our new 32 bit local variable into a 32 bit register. This can be done fairly easily:

```
:0043A369 0FB745FA        movzx eax, word ptr [ebp-06]
:0043A377 0FB755FA        movzx edx, word ptr [ebp-06]
```

becomes

```
:0043A369 908B45F8        (nop) mov eax, dword ptr [ebp-08]
:0043A377 908B55F8        (nop) mov edx, dword ptr [ebp-08]
```

That takes care of all the instances of our local variable, but we're not done. At first, it looks like [ebp-0C] is doing some dirty work here. It gets assigned 10000, which is then compared with the # of bytes to read. If the number of bytes is bigger, then only 10000 bytes are read. It looks for all the world like part of the protection scheme, but it isn't. If, after having applied the patches I've mentioned so far, you force the jump at 43A370, the program crashes. What ebp-0C does is simply make sure that the prog is reading only 64k at a time. Notice that most of the function is a loop that reads 64k chunks until it's read everything it needs to. Another 64k red herring. However, the instruction at 43A324 needs to be changed. Remember that edx contains the start address of the segment. For the code segment, this is less than 64k (i.e., 600), but for the data segment it's > 64k. I missed this instruction at first and ended up with a very frustrating problem. The patches I've described so far work, but IDA was lining up the data segment in entirely the wrong place. I spent hours looking for some alternate protection scheme, before I came back and realized what I had missed.

43A324 was the culprit, so that needs to be patched in the same manner as 43A369 and 43A377.

That's it. No more 64k boundary and all files are disassembled properly.

IV. Reflections on the protection scheme

At first, I thought that this must have been done in assembly. Operand override prefixes seemed pretty arcane. However, it was odd to leave that 2 byte whole in the stack. If you're programming in assembly, why do that? Of course, even if that whole hadn't been there, we could have simply added space for a local variable (see the Appendix). Upon further reflection, I realized what the programmer did and he did it in C/C++. The function at 43A314 was changed from taking a long int argument to taking a short int argument and the argument passed to it at 43A804 was cast from a long int to a short int. Two tiny changes in the source code produced exactly this effect (changes, of course, from the real, unlimited version of the program). The 2 bytes we needed on the stack were there because all compilers align 32bit values on dword boundaries and the other surrounding local variable were 32 bit. Furthermore, when you run ida.wll through the cracked IDA you see that the call at 0043A395 to 43EC04 is actually a call to the Borland C library routine _fread (read from file). Why are some of the arguments to 43A314 passed through registers instead of on the stack? See the description of the _fastcall calling convention in the Appendix. All in all, however, this is an interesting protection scheme, and certainly not of a kind that I have ever heard of before. Now all we need to do is get those idb files

loaded and produce asm and lst files (I think the code is in there,
it's just a matter of getting to it).

V. The Expiration Date

This crack is utterly trivial. The code that checks the date is
immediately after the test for 64k files at the beginning of the
program. I'll leave this crack as a mindless exercise.

VI. Summary of the patch

```
:00403CED 760B           jbe 00403CFA  (change 76 to EB)
:0043A82E 662BCB         sub cx, bx  (change 66 to 90)
:0043A31D 66894DFA       mov word ptr [ebp-06], cx (change to 90894DF8)
:0043A324 0FB7C2         movzx eax, dx  (change to 908BC2)
:0043A35E 66837DFA00     cmp word ptr [ebp-06], 0000 (change to 90837DF800)
:0043A369 0FB745FA       movzx eax, word ptr [ebp-06] (change to 908B45F8)
:0043A377 0FB755FA       movzx edx, word ptr [ebp-06] (change to 908B55F8)
:0043A3ED 66295DFA       sub word ptr [ebp-06], bx (change to 90295DF8)
:0043A414 66837DFA00     cmp word ptr [ebp-06], 0000 (change to 90837DF800)
```

VII. The function at 43A314

```
:0043A314 55             push ebp
:0043A315 8BEC           mov ebp, esp
:0043A317 83C4E0         add esp, FFFFFFE0
:0043A31A 53             push ebx
:0043A31B 56             push esi
```

```
:0043A31C 57              push edi
:0043A31D 66894DFA        mov word ptr [ebp-06], cx      ; here we go!
       ; 7A00 is passed through cx (a 16bit register-that's no good!)
:0043A321 8945FC          mov dword ptr [ebp-04], eax
:0043A324 0FB7C2          movzx eax, dx           ; what's this about?
         ; it's about hours of headache for me (see above)
:0043A327 8BD0            mov edx, eax
:0043A329 8B45FC          mov eax, dword ptr [ebp-04]
:0043A32C 8B7D08          mov edi, dword ptr [ebp+08]
:0043A32F E83065FEFF      call 00420864
:0043A334 F60532EF490010  test byte ptr [0049EF32], 10
:0043A33B C745F400000100  mov [ebp-0C], 00010000 ; 64k?! is this relevant? NO
:0043A342 BA02000000      mov edx, 00000002
:0043A347 7501            jne 0043A34A
:0043A349 4A              dec edx

|:0043A347(C)
|
:0043A34A 8955F0          mov dword ptr [ebp-10], edx
:0043A34D 8B4DF0          mov ecx, dword ptr [ebp-10]
:0043A350 0FAF4DF4        imul ecx, dword ptr [ebp-0C]
:0043A354 51              push ecx
:0043A355 E8028BFFFF      call 00432E5C
:0043A35A 59              pop ecx
:0043A35B 8945EC          mov dword ptr [ebp-14], eax
:0043A35E 66837DFA00      cmp word ptr [ebp-06], 0000 ; cmp 7A00, 0
:0043A363 0F86B6000000    jbe 0043A41F

|:0043A419(C)
```

```
|
:0043A369 0FB745FA        movzx eax, word ptr [ebp-06] ; needs patching
:0043A36D 3B45F4          cmp eax, dword ptr [ebp-0C] ; [ebp-c]==10000
:0043A370 7605            jbe 0043A377
:0043A372 8B55F4          mov edx, dword ptr [ebp-0C]
:0043A375 EB04            jmp 0043A37B

|:0043A370(C)
|
:0043A377 0FB755FA        movzx edx, word ptr [ebp-06] ; [ebp-06]==7A00

|:0043A375(U)
|
:0043A37B 8955E8          mov dword ptr [ebp-18], edx ;edx==7A00
:0043A37E 8BC7            mov eax, edi
:0043A380 E84B51FEFF      call 0041F4D0
:0043A385 8B4DFC          mov ecx, dword ptr [ebp-04]
:0043A388 51              push ecx
:0043A389 8B45E8          mov eax, dword ptr [ebp-18] ;==7A00
:0043A38C 50              push eax  ; # of bytes to read
:0043A38D 8B55F0          mov edx, dword ptr [ebp-10]
:0043A390 52              push edx
:0043A391 8B4DEC          mov ecx, dword ptr [ebp-14]
:0043A394 51              push ecx
:0043A395 E86A48FFFF      call 0042EC04  ; this is the call that ends up
                                         ; at ReadFile and takes #of bytes
                                         ; to read as third arg

. . . . . some irrelevant code removed . . . . .
```

```
|:0043A3AF(C), :0043A3C8(U), :0043A3D4(C)
|
:0043A3ED 66295DFA        sub word ptr [ebp-06], bx ; this needs patching
:0043A3F1 3B5DE8          cmp ebx, dword ptr [ebp-18]
:0043A3F4 7419            je 0043A40F
:0043A3F6 8B55EC          mov edx, dword ptr [ebp-14]
:0043A3F9 52              push edx
:0043A3FA E8F589FFFF      call 00432DF4
:0043A3FF 59              pop ecx
:0043A400 68BE000000      push 000000BE ; message from ida.hlp:
                         ; "Error during read, not all of file read"
:0043A405 E872640200      call 0046087C
:0043A40A 59              pop ecx
:0043A40B 33C0            xor eax, eax
:0043A40D EB1F            jmp 0043A42E

|:0043A3F4(C)
|
:0043A40F E8D8FEFFFF      call 0043A2EC
:0043A414 66837DFA00      cmp word ptr [ebp-06], 0000 ; needs patching
:0043A419 0F874AFFFFFF    ja 0043A369

|:0043A363(C)
|
:0043A41F 8B55EC          mov edx, dword ptr [ebp-14]
:0043A422 52              push edx
:0043A423 E8CC89FFFF      call 00432DF4
:0043A428 59              pop ecx
```

```
:0043A429 B801000000      mov eax, 00000001


|:0043A40D(U)
|
:0043A42E 5F              pop edi
:0043A42F 5E              pop esi
:0043A430 5B              pop ebx
:0043A431 8BE5            mov esp, ebp
:0043A433 5D              pop ebp
:0043A434 C20400          ret 0004
```

APPENDIX: Stack frames and calling conventions

The most common way for a program to set up the stack during a function call is to use the ebp register (the base register) to hold the base of the stack relative to that function. An amount is then subtracted from esp which represents the amount of space reserved for local variables. This is done with the following code which is at the beginning of most functions:

```
push ebp
mov ebp,esp
add esp, -10 (FFFFFFF0) ; 10 bytes for local variables
```

The funcion can then use positive offsets to ebp to reference arguments and negative offsets to reference local variables.

For example:

```
loc1:   push arg3
loc2:   push arg2
loc3:   push arg1
loc4:   call sub
loc5:   mov [ebp-8], eax

proc sub
sub1:   push ebp       ; save ebp from calling function
sub2:   mov ebp, esp   ; set ebp to point at stack base for this
function
sub3:   sub esp, 10    ; reserve 10 bytes for four local variables
sub4:   mov eax, [ebp+8]   ; move arg1 into eax
sub5:   mov ecx, [ebp+C]   ; move arg2 into ecx
sub6:   mov edx, [ebp+10]  ; move arg3 into edx
sub7:   mov [ebp-4], eax   ; move arg1 into local_1
sub8:   mov [ebp-8], edx   ; move arg3 into local_2
sub9:   push [ebp+C]       ; push arg2 onto stack

At sub1, the stack looks like this:

arg3    <-- esp-C
arg2    <-- esp-8
arg1    <-- esp-4
loc5    <-- esp

at sub4 it looks like this:

arg3    <-- ebp+10
```

```
arg2    <-- ebp+C
arg1    <-- ebp+8
loc5    <-- the return address
ebp from calling function  <-- current ebp
local_1 <-- ebp-4
local_2 <-- ebp-8
local_3 <-- ebp-C
local_4 <-- ebp-10   <-- esp
```

Things are not always this pretty, unfortunately. There are three factors that can disrupt this happy picture.

1. Stack frame optimization
2. Alternate calling conventions
3. Enregistered local variables

1. Most compilers these days (certainly M$ VC++ and Borland C++) have an optimization setting that allows you to turn off ebp based stack frames. This makes the function overhead smaller and frees up the ebp register for other uses. The function still references arguments and local variables off the stack, but the ebp register doesn't point to the function's stack base. Instead, esp is used to reference the srguments and locals. The problem is that every time something is pushed onto or popped off of the stack, the value of esp changes. The compiler is able to compensate for this by adjusting the amount esp is offset when referencing the arguments and locals. Here's an example:

```
sub1:   mov eax, [esp+4]   ; mov arg1 into eax
```

```
sub2:   push ebx        ; push ebx onto the stack for whatever reason
sub3:   mov edx, [esp+8] ; move arg1 into edx---the offset has changed

                        ; because we pushed ebx onto the stack at sub2
```

This is annoying because it makes much harder for things that aren't compilers (like us) to keep track of what's getting referenced where. The nice thing about IDA Pro is that marks all these references for you! That's why it *must* be cracked.

2. A calling convention determines the order in which arguments are passed and how they are passed. The two most common calling convention are the Pascal and C calling conventions, which fravia+ has explained quite well elsewhere. However, there are two other conventions, which pass arguments through registers, that are worth being aware of. The first is the _thiscall convention, which follows the C calling convention, but also passes a pointer to the object this (in C++, this refers to the current object) through register ecx. The second is the _fastcall convention, which passes the first two arguments that are 32 bits or less through registers ecx and edx, and then passes the rest of the arguments on the stack with the C calling order.

3. Local variables are not always stored on the stack after the return address. For optimization purposes (it's much faster to get a value from a register than from memory), registers are sometimes used to hold local variables. Of course, there being a limited number of general registers in Intel chips (RISC chips can have 5-10 times as many registers) this can only be true to a limited extent.

Furthermore, the compiler has to use some registers as what are called scratch registers (i.e., places to temporarily hold values) while it moves values in and out of memory. Detecting enregistered local variables can be difficult.