Chapter 8
DOS/Windows Structures, Definitions, and Events

This chapter contains miscellaneous structures, typedefs, and so on for the NIOS DOS environment.

Note: For all C calls, the registers *ebp, esp, ebx, esi,* and *edi* are preserved. The state of the direction flag is undefined.

DOS/MS Windows Structures

ClientRegStruc

DOS NIOS client register structure.

```
#include <dosvmm.h>

ClientRegStruc struc
  CrsEDI       dd ?
  CrsESI       dd ?
  CrsEBP       dd ?
               dd ? ; ESP before pushad
  CrsEBX       dd ?
  CrsEDX       dd ?
  CrsECX       dd ?
  CrsEAX       dd ?
               dd ? ; For compatibility w/ Windows crs
  CrsEIP       dd ?
  CrsCS        dw ?,0
  CrsEFlags    dd ?
  CrsESP       dd ?
  CrsSS        dw ?,0
  CrsES        dw ?,0
  CrsDS        dw ?,0
  CrsFS        dw ?,0
  CrsGS        dw ?,0
  CrsReserved dd  9 dup (?)
}ClientRegStruc ends;
```

Note: Refer to *Register Aliases* and *Processor Flag Definitions* in the *Bit Definitions and Return Codes* section of this chapter.

DosInfoBlock

Exported data structure containing useful information about DOS.

```
#include <dosvmm.h>

DosInfoStruc struc
   DOSMinorVersion         db ?
   DOSMajorVersion         db ?
   DOSNiosPSP              dw ?
   DOSCountryInfo          dd ?   ;As returned by DOS
                                  ;func 6501h
   DOSUppercaseTable       dd ?   ;As returned by DOS
                                  ;func 6502h
   DOSFileNameUpperTable   dd ?   ;As returned by DOS
                                  ;func 6504h
   DOSInvalidCharTable     dd ?   ;As returned by DOS
                                  ;func 6505h
   DOSCollatingTable       dd ?   ;As returned by DOS
                                  ;func 6506h
   DOSListOfListLinAddr    dd ?
   DOSInDosFlagLinAddr     dd ?
   DOSCritErrFlagLinAddr   dd ?
   DOSSmallSDALinAddr      dd ?
   DOSSmallSDALength       dd ?   ;Multiple of 4
   DOSInt10ActiveFlag      dd ?   ;Will be NULL if not
                                  ;supported
   DOSInt13ActiveFlag      dd ?   ;Will be NULL if not
                                  ;supported
   DOSInt15ActiveFlag      dd ?   ;Will be NULL if not
                                  ;supported
   DOSInt16ActiveFlag      dd ?   ;Will be NULL if not
                                  ;supported
   DOSInfoFlags            dd ?   ;Various info flags
}DosInfoStruc ends;
```

The DOSInt??ActiveFlag's each point to a UINT8 size flag that is non-zero when the specified V86 interrupt is active.

Possible DOSInfoFlags values

```
DIF_NOVDOS_BIT  equ  00000001h   ;Set if NOVDOS
DIF_MSDOS_BIT   equ  00000002h   ;Set if DR/Novell DOS
```

Int2FInfoStruc

Structure used by **DosRegisterV86Int2F** function.

```
#include <dosvmm.h>
Int2FInfoStruc struc
   I2FLink           dd ?
   I2FHandler        dd ?
   I2FAhValue        db ?
   I2FReserved       db 3 dup (?)
}Int2FInfoStruc ends;
```

NiosWin32EntryPoints

Structure passed to **DeviceIoControl** function when locating the NIOS entry points. This structure is filled out by NIOS on return.

```
typedef struct NiosWin32EntryPointsStruc
   UINT32 Win32NiosMajorVersion;
   UINT32 Win32NiosMinorVersion;
   UINT32 (*Win32NiosFarCall)(
                         UINT32 function,
                         ...);
   UINT32 (*Win32InvokeCNlmApi)(
                         UINT32 nlmApiAddress,
                         UINT32 apiParmCount,
                         ...);
   void   *Win32Reserved0;         // For future use
   void   *Win32Reserved1;         // For future use
}NiosWin32EntryPoints;
```

Functions available through the Win32NiosFarCall entry point.

```
#define  WIN32_NIOS_BEGIN_USE_API   0x00000000
#define  WIN32_NIOS_END_USE_API     0x00000001
#define  WIN32_NIOS_COPY_MEM        0x00000002
#define  WIN32_NIOS_COPY_STRING     0x00000003
#define  WIN32_NIOS_MAP             0x00000004
#define  WIN32_NIOS_UNMAP           0x00000005
```

PopupInfoStruc

Structure returned by **DosVidGetPipupInfo** function.

```
#include <dosvmm.h>
typedef struct PopupInfoStruc
   UINT8 *PILineDrawChars;
   UINT8 PIStartCol;
   UINT8 PIStartRow;
   UINT8 PINumCols;
   UINT8 PINumRows;
   UINT8 PIUserSpaceAttr;
   UINT8 PITitleAttr;
   UINT8 PISubtitleAttr;
   UINT8 PIPromptAttr;
   UINT8 PIDisplayType;
}PopupInfo;
```

PIStartCol Defines the starting column number of the popup's user area.

PIStartRow Defines the starting row number of the popup's user area.

PINumCols Defines the number of columns in the popup's user area.

PINumRows Defines the number of rows in the popup's user area.

PIUserSpaceAttr Defines the background/foreground attribute used for normal text inside of the popup's user area.

PITitleAttr Defines the background/foreground attribute used for text in the popup's title.

PISubtitleAttr Defines the background/foreground attribute used for text in the popup's subtitle.

PIPromptAttr Defines the background/foreground attribute used for text in the popup's prompt.

PIDisplayType Specifies the mechanism used to display the popup. See DT_??? definitions below for possible values.

PILineDrawChars Pointer to array of UINT8 characters which contain the characters used for drawing a popup's borders. See the LD_???? definitions below for indices into this array.

DT_??? definitions for PIDisplayType field

```
#define  DT_POPUP_USED_BIOS         0x00
            // Used BIOS functions for output
#define  DT_POPUP_USED_DIRECT       0x01
            // Direct video memory access
#define  DT_POPUP_USED_WIN_MSG_MODE 0x02
            // Used Windows MessageMode services
```

LD_??? indices into PILineDrawChars array

```
#define  LD_TOP_LEFT_CORNER      0x00
#define  LD_TOP_RIGHT_CORNER     0x01
#define  LD_BOT_LEFT_CORNER      0x02
#define  LD_BOT_RIGHT_CORNER     0x03
#define  LD_VERT_LINE            0x04
#define  LD_HORZ_LINE            0x05
```

SDBInfo

Structure used with DosSharedBufGetInfo function.

```
#include <dosvmm.h>

SDBInfo struct
   SDBSize           dd ? ;Size of buffer in bytes
   SDBAddress        dd ? ;Linear address of buffer
   SDBSegment        dd ? ;Paragraph address of buffer
SDBInfo ends
```

VmCbStruc

VM control block structure.

```
#include <dosvmm.h>

VmCbStruc struc
   VMCBStatus             dd ?  ;See VXD.INC for def's
   VMCBHighLinAddress     dd ?
   VMCBClientRegStruc     dd ?
   VMCBVmId               dd ?
VmCbStruc ends
```

WinEventStruc

Structure passed to MS Windows VMM event consumer handlers as the first custom parameter. This defines the register information at the time the event was produced by the Windows VMM.

Consumers can modify this structure to set up return information as needed by the event.

```
WinEventStruc   struc
      WesEFlags       dd      ?
      WesEDI          dd      ?
      WesESI          dd      ?
      WesEBP          dd      ?
                      dd      ?    ; ESP before pushad
      WesEBX          dd      ?
      WesEDX          dd      ?
      WesECX          dd      ?
      WesEAX          dd      ?
WinEventStruc   ends
```

UserCmdStruc

```
#include <cmdcom.h>
typedef struct UserCmdStruc

   struct UserCmdStruc  *UCLink;
   modHandle            UCOwner;
   void                 (*UCHandler)(
                        struct UserCmdStruc
                                  *cmdBlock,
                        UINT8     *cmdLine,
                        UINT32    argCount,
                        UINT8     *argVector[]);
   UINT8                *UCText;
   UINT32               UCReserved;
}UserCmd;
```

UCOwner Contains a pointer to the module handle of the module which registered the command.

UCHandler

Function called when the registered command is invoked.

UCText Pointer to length-preceded ASCIIZ string defining the name of the command. It must be uppercase and cannot exceed 10 bytes (including the len and NULL bytes).

UCReserved

Reserved for future use.

Bit Definitions, Return Codes, and Parameters

Nios Win32 DeviceIoControl Parameters

Valid *dwIoControlCode* values when calling the "\\,\NIOS" device through the **Win32 DeviceIoControl** API function.

```
#include <nlmapi.h>
#define  WIN32_GET_NIOS_INTERFACE   0xDDDD0000
```

DOS NIOS Int 2Fh Query ID

```
#include <nlmapi.h>

DOS_NIOS_INT2F_ID    equ  0D8C1h
WIN_NIOS_INT2F_ID    equ  0D8C3h
```

Register Aliases

```
CrsAL       equ      byte ptr CrsEAX
CrsAH       equ      byte ptr CrsEAX+1
CrsAX       equ      word ptr CrsEAX
CrsBL       equ      byte ptr CrsEBX
CrsBH       equ      byte ptr CrsEBX+1
CrsBX       equ      word ptr CrsEBX
CrsCL       equ      byte ptr CrsECX
CrsCH       equ      byte ptr CrsECX+1
CrsCX       equ      word ptr CrsECX
CrsDL       equ      byte ptr CrsEDX
CrsDH       equ      byte ptr CrsEDX+1
CrsDX       equ      word ptr CrsEDX
CrsSI       equ      word ptr CrsESI
CrsDI       equ      word ptr CrsEDI
CrsBP       equ      word ptr CrsEBP
CrsSP       equ      word ptr CrsESP
CrsIP       equ      word ptr CrsEIP
CrsFlags    equ      word ptr CrsEFlags
```

Processor Flag Definitions

```
EF_CARRY_BIT         equ  00000001h
EF_PARITY_BIT        equ  00000004h
```

```
EF_AUXC_BIT          equ  00000010h
EF_ZERO_BIT          equ  00000040h
EF_SIGN_BIT          equ  00000080h
EF_TRACE_BIT         equ  00000100h
EF_INTERRUPT_BIT     equ  00000200h
EF_DIRECTION_BIT     equ  00000400h
EF_OVERFLOW_BIT      equ  00000800h
EF_IOPL_BITS         equ  00003000h
EF_NESTED_TASK_BIT   equ  00004000h
EF_RESUME_BIT        equ  00010000h
EF_VM_BIT            equ  00020000h
EF_CPUID_BIT         equ  00200000h
```

Max Number of VMs Supported

```
#include <dosvmm.h>
MAX_NUM_VM  equ   32
```

DosCreate - File createAttributes

```
CREATE_NORMAL           equ   0h
CREATE_HIDDEN           equ   2h
CREATE_SYSTEM           equ   4h
CREATE_HIDDEN_SYSTEM    equ   6h
```

DosOpen - File openAttributes

OR one of the open modes with one of the sharing modes.

#include <dosvmm.h>

Open Modes

```
OPEN_READ_ONLY       equ  0
OPEN_WRITE_ONLY      equ  1
OPEN_READ_WRITE      equ  2
```

Sharing Modes

```
OPEN_DENY_ALL           equ     10h
OPEN_DENY_WRITE         equ     20h
OPEN_DENY_READ          equ     30h
OPEN_DENY_NONE          equ     40h
```

DosSeek Type Values

```
SEEK_SET              equ  0
SEEK_CURRENT          equ  1
SEEK_END              equ  2
```

CharOut Macro

Assembly macro used to display a character on the debug terminal if DEBUG is defined. No code is generated if DEBUG is *not* defined. This function preserves all registers.

An NLM that uses this macro must include **NiosDebugCharOut** in the module's linker function import list.

```
#include <nios.inc>
Usage:  CharOut 'c'
```

NwEnableLogging Parameter

SYSTEM.INI parameter for Windows v4.x NIOS that tells NIOS to initially enable or disable logging. Logging is performed with the MT_LOG_STATUS message type for printf along with some instances of MT_DEBUG_OUT, MT_DEBUG_TRACE and NiosDprintf, depending on your system setup. It can be controlled at runtime with the NiosEnableLogging api or the ENABLE/DISABLE LOGGING command.

Example:

```
[386enh]
nwenablelogging=TRUE
```

NwHomeDir Parameter

SYSTEM.INI parameter for Windows v4.x NIOS that tells NIOS where NetWare related files are located. This is typically where the NetWare client modules will be found as well as other associated files. The default path is "C:\", but should be changed as needed. The NIOS NiosGetSystemDirectory API function returns this path.

Example:

```
[386enh]
nwhomedir=C:\NWCLIENT
```

NwNumV86pages Parameter

SYSTEM.INI parameter for Windows NIOS that configures the number of conventional memory pages reserved by NIOS for NLM usage. By default NIOS allocates 2 pages (8K) for this purpose. The parameter may need to be increased if an NLM fails to load or some function fails due to an inability to allocate conventional (below 1meg) memory.

Example SYSTEM.INI:

```
[386enh]
nwnumV86pages=3
```

DOS/MS Windows Events

Well-Known DOS Task Switcher Event Types

The events described in this section are produced when DOS generates a task switcher callout.

Standard Mode Windows does not issue the "TASK SWITCHER INIT" event, therefore an NLM should hook the "TASK CREATE SESSION" event to be notified when a task switcher loads, otherwise the NLM can check the DosSwitcherActive global variable to determine this.

NIOS will refuse a task switch query if the task switcher tries to switch while an NLM is running. Therefore an NLM need not worry about a task switch occurring while it is executing in protected mode or calling Ring 3 code in the context of a nested execution block.

For events that define a success/fail value, which is usually signaled with a zero/non-zero return code, consumer should return NESL_EVENT_CONSUMED in case of a fail. If the return value would be zero, the consumer should specify NESL_EVENT_NOT_CONSUMED to indicate they are in a state which allows the task switch event to occur.

The custom parameter passed to the consumer event handler is the task switcher session ID for the following events:

- Query Suspend
- Activate Session
- Session Active
- Create Session
- Destroy Session

For all other events this field is not set and should be ignored:

- Switcher Init
- Switcher Terminate

Task switcher events are not normally registered in the system. They will be registered when the first task switcher is loaded and de-registered when the last task switcher is unloaded. Therefore, consumer will NOT receive NESL_OK upon register for the event if no task switcher is loaded at the time. Instead the consumer will receive NESL_CONSUMER_NOT_FOUND.

Define the M-DOS task switcher event service strings NIOS clients may register for event notification as follows:

```
#include <tasksw.h>

NE_TASK_INIT                equ     "TASK SWITCHER INIT",0
NE_TASK_QUERY_SESSION       equ     "TASK QUERY SUSPEND",0
NE_TASK_SUSPEND_SESSION     equ     "TASK SUSPEND SESSION",0
NE_TASK_ACTIVATE_SESSION    equ     "TASK ACTIVATE SESSION",0
NE_TASK_SESSION_ACTIVE      equ     "TASK SESSION ACTIVE",0
NE_TASK_CREATE_SESSION      equ     "TASK CREATE SESSION",0
NE_TASK_DESTROY_SESSION     equ     "TASK DESTORY SESSION",0
NE_TASK_TERMINATE           equ     "TASK SWITCHER TERM",0
```

Define the callback information structure as follows:

```
TaskCallbackInfoStruc    struc
    TASKFlink         dd  ?  ;Pointer to next callback info
                             ;  struct
    TASKFunction      dd  ?  ;Pointer to notification
                             ;  function
    TASKReserved      dd  ?  ;reserved
    TASKAPIInfoStrucs dd  ?  ;Address of zero-terminated
                             ;  list of API
TaskCallbackInfoStruc ends   ;info structures
```

WIN16 GETPROCADDR AVAIL

This event is generated during Windows initialization to signal that the **Win16GetProcAddress** service is available. This service is unavailable during a small window of time during Windows initialization. This event has no event data parameter. This event is NOT consumable.

```
NE_WIN16_GETPROCADDR_AVAIL  equ  "WIN16 GETPROCADDR AVAIL",0
```

WIN INT2F 1605 INIT

This event is produced on the back end of the Int 2F AX=1605h callout issued by MS Windows during its startup. The first custom parameter passed to a consumer event handler is a pointer to the active **ClientRegStruc**.

If an event consumer wishes to abort MS Windows, it must set CrsCX to a non-zero value and display an error message describing the reason for the MS Windows abort. Also, the handler must consume the event.

Consumers can assume that the CrsCX register coming into the consumer handler will be zero (that is, MS Windows *not* aborted). If MS Windows is aborted after a consumer has processed the event, then the "WIN INT2F 1606 TERM" event will be produced to allow consumers a chance to clean up any MS Windows-specific initialization.

See DOSVMM.INC for more information.

```
NE_WIN_2F_INIT       equu      "WIN INT2F 1605 INIT", 0
NE_WIN_2F_TERM       equ       "WIN INT2F 1606 TERM", 0
```

WIN INT2F 1606 TERM

This event is produced when enhanced-mode Windows issues its Int 2F AX=1606h callout. The first custom parameter passed to a consumer event handler is a pointer to the active **ClientRegStruc**. This event is *not* consumable.

```
NE_WIN_2F_TERM       equ        "WIN INT2F 1606 TERM",0
```

WIN VXD REAL MODE INIT

This event is produced immediately before VNIOS returns from its real-mode init function. All return values defined for a VxD's return from real-mode initialization are set at this point. A consumer can modify these values if needed.

The first custom parameter passed to the consumer event handler is a pointer to the active **ClientRegStruc.**

```
NE_WIN_REAL_MODE_INIT   equ "WIN VXD REAL MODE INIT",0
```

WIN VMM EVENTS

The following events are produced when the MS Windows VMM generates an event callout.

For events that define a success/fail return value, which is usually signaled with the carry flag, consumers should generally consume the event if they return with the carry flag set.

The first custom parameter passed to the consumer event handler is a pointer to a **WinEventStruc**, which defines the registers and flags at the time the MS Windows VMM generated the event. The event handler can modify this structure as needed (for example setting the carry bit in the *WesEFlags* field.)

Note that the Carry flag in the WesEFlags will always be clear before the event is produced; therefore consumers do not need to explicitly clear the carry flag.

```
NE_WIN_BEGIN_MSG_MODE   equ   "WIN BEGIN MSG MODE", 0
NE_WIN_BEGIN_PM_APP     equ   "WIN BEGIN PM APP", 0
NE_WIN_CLOSE_VM_NOTIFY  equ   "WIN CLOSE VM NOTIFY", 0
NE_WIN_CREATE_VM        equ   "WIN CREATE VM", 0
NE_WIN_CRIT_REBT_NOTIFY equ   "WIN CRIT REBOOT NOTIFY", 0
NE_WIN_DEBUG_QUERY      equ   "WIN DEBUG QUERY", 0
NE_WIN_DESTROY_VM       equ   "WIN DESTROY VM", 0
NE_WIN_DEVICE_INIT      equ   "WIN DEVICE INIT", 0
NE_WIN_DEV_REBT_NOTIFY  equ   "WIN DEVICE REBOOT NOTIFY",0
NE_WIN_END_MSG_MODE     equ   "WIN END MSG MODE", 0
NE_WIN_END_PM_APP       equ   "WIN END PM APP", 0
NE_WIN_INIT_COMPLETE    equ   "WIN INIT COMPLETE", 0
NE_WIN_POWER_EVENT      equ   "WIN POWER EVENT", 0
NE_WIN_QUERY_DESTROY    equ   "WIN QUERY VM DESTROY", 0
NE_WIN_REBT_PROCESSOR   equ   "WIN REBOOT PROCESSOR", 0
NE_WIN_SET_DEVICE_FOCUS equ   "WIN SET DEVICE FOCUS", 0
NE_WIN_SYS_CRIT_EXIT    equ   "WIN SYS CRIT EXIT", 0
NE_WIN_SYS_CRIT_INIT    equ   "WIN SYS CRIT INIT", 0
NE_WIN_SYS_VM_INIT      equ   "WIN SYS VM INIT", 0
NE_WIN_SYS_VM_TERM      equ   "WIN SYS VM TERM", 0
NE_WIN_SYS_EXIT         equ   "WIN SYS EXIT", 0
NE_WIN_VM_CRIT_INIT     equ   "WIN VM CRIT INIT", 0
NE_WIN_VM_INIT          equ   "WIN VM INIT", 0
NE_WIN_VM_NOT_EXEC      equ   "WIN VM NOT EXEC", 0
NE_WIN_VM_RESUME        equ   "WIN VM RESUME", 0
NE_WIN_VM_SUSPEND       equ   "WIN VM SUSPEND", 0
NE_WIN_VM_TERM          equ   "WIN VM TERM", 0
```

New events for Windows v4.0

```
NE_WIN_DYNA_DEV_INIT    equ   "WIN DYNA DEV INIT", 0
NE_WIN_DYNA_DEV_EXIT    equ   "WIN DYNA DEV EXIT", 0
NE_WIN_CREATE_THREAD    equ   "WIN CREATE THREAD", 0
NE_WIN_THREAD_INIT      equ   "WIN THREAD INIT", 0
NE_WIN_THREAD_TERM      equ   "WIN THREAD TERM", 0
NE_WIN_THREAD_NOT_EXEC  equ   "WIN THREAD NOT EXEC", 0
NE_WIN_DESTROY_THREAD   equ   "WIN DESTROY THREAD", 0
NE_WIN_PNP_NEW_DEVNODE  equ   "WIN PNP NEW DEVNODE", 0
NE_WIN_W32_DEV_IOCTL    equ   "WIN W32 DEV IOCTL", 0
```