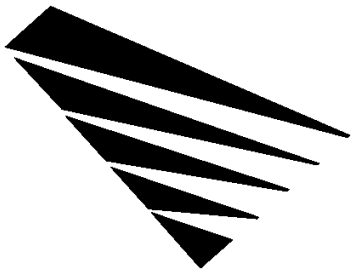

Chapter 8
Accessing an NLM from a Vxd

NLMs and Vxds

This chapter details the methods a Vxd uses to access exported NLM functions and describes the function calls and macros that are available.

Vxd Init, Deinit, and Dependencies

During Vxd initialization you should first determine if the NIOS interfaces are available before attempting to invoke an NLM service. This is accomplished by calling the Vxd service **NiosVxdGetVersion**.

Because NLM's are unloadable, Vxd's must take care to not attempt access to an NLM that is unloaded. This can be accomplished in two ways.

Method #1

The first method requires that the Vxd inform NIOS about the NLMs it is going to access. Until the Vxd informs NIOS that it is no longer accessing an NLM, NIOS will refuse to allow the NLM to unload from the system. This is accomplished by using the **NiosVxdBeginNlmUse** and **NiosVxdEndNlmUse** Vxd services. Typically a Vxd will call **NiosVxdBeginNlmUse** during *Device_Init* or *Init_Complete* and call **NiosVxdEndNlmUse** during *System_Exit* or *Sys_Crit_Exit*. It is important that the Vxd calls **NiosVxdEndNlmUse** during Windows exit since the NLM won't be allowed to unload until all external references to it have been removed.

Dynamically loadable Vxd's should call **NiosVxdEndNlmUse** when the module is unloaded.

A Vxd need not call **NiosVxdBeginNlmUse** or **NiosVxdEndNlmUse** on NIOS.NLM since this NLM cannot be unloaded inside of Windows.

Method #2

The second method does not make use of the **NiosVxdBeginNlmUse** or **NiosVxdEndNlmUse** services. Instead the Vxd registers with NIOS to receive notification of when NLM modules are unloaded. This is done using the *NIOS MODULE UNLOADED* NESL event and watching for the named module. If the event occurs, the Vxd must immediately stop accessing the NLM.

Many NIOS and NLM services require a module handle parameter. A Vxd needing to invoke such a service should first create a pseudo module handle using the **NiosCreateModuleHandle** service. The Vxd must destroy the module handle when it unloads or during system exit. This is accomplished using the **NiosDestroyModuleHandle** service.

Tools

A Vxd will typically include and use the NIOSVXD.INC file. This file contains definitions for *NlmCall*, *NlmJmp*, and so forth.

NIOSVXD.INC is compatible with both the MASM5.EXE and ML.EXE (Masm 6.x) assemblers.

A Vxd can include most NLM include files to gain access to NLM definitions

Debugging

NlmCall and *NlmJmp* expand to non-code values, specifically an Int 20h with two 32-bit values after it. After the Int 20h is executed, NIOS replaces the 10-bytes of information with valid code. Because of this, it is not possible to p (proceed) over an *NlmCall* that hasn't been executed yet, since this will cause an Int 3 to be inserted in the values that NIOS needs to fixup the call. To proceed over an *NlmCall* set an execution breakpoint 10 bytes after the Int 20h.

Examples

A sample Vxd called VXDTONLM.386 is available which gives examples of how to use the services outlined in this chapter.

NiosVxdBeginNlmUse

Description

Determines if the specified NOLM is present in the system, and builds a dependency between the calling Vxd and the specified NLM.

Syntax

```
#include <niosvxd.inc>

void
NiosVxdBeginNlmUse (
        UINT8       *nlmName );
```

Parameters

nlmName Offset of an ASCIIZ string of the NLM that the Vxd is no longer using. This is a case insensitive string.

Returns

0 = NLM is NOT present in the system. The Vxd may choose to call NIOS to load the NLM.

!0 = Function successful.

"C" registers are preserved.

Remarks

This function is typcially used during Vxd initailization.

As long as a dependency is present the NLM will not be allowed to unload from the system. It is important that the Vxd invoke the NiosVxdEndNlmUse macro as soon as it's done using the specified NLM. A static Vxd (not dynamically loadable/unloadable) must invoke the NiosVxdEndNlmUse macro when Windows is exiting, no later than the Sys_Critical_Init callout. A dynamic Vxd should invoke NiosVxdEndNlmUse when it is unloaded.

Example:

```
IpxNlmName      db     'IPX.NLM",0

VxdCall   NiosVxdBeginNlmUse,<OFFSET32 IpxNlmName>
test   eax, eax
jz     NlmIsntPrsent
```

See Also

NiosVxdEndNlmUse

NiosVxdEndNlmUse

Description

Destroys the dependency between the calling Vxd and the specified NLM allowing the NLM to be subsequently unloaded from the system.

Syntax

#include <niosvxd.inc>

void
NiosVxdEndNlmUse (
UINT8 *nlmName);

Parameters

nlmName Offset of an ASCIIZ string of the NLM that the Vxd is no longer using. This is a case insensitive string.

Returns

Nothing
"C" registers are preserved

Remarks

.Example:

```
IpxNlmName      db     'IPX.NLM",0

VxdCall   NiosVxdEndNlmUse,<OFFSET32 IpxNlmName>
```

See Also

NiosVxdBeginNlmUse

NiosVxdGetVersion

Description

Returns the NIOS version information and a value that signals whether or not NIOS has completed initialization..

On Entry

Nothing

On Exit

Carry flag set = NIOS is NOT loaded
AX, ECX are undefined

Carry flag clear = NIOS is loaded
AX = Major, Minor version of NIOS Vxd
ECX = 0 if NIOS has initialized, carry must be clear.
= !0 if NIOS has not completed it Device Init initialization.
All other NIOS Vxd services as well as *NlmCall* services are unavailable.

Remarks

Example:

```
VxdCall NiosVxdGetVersion
jc     NiosNotPresent

test   ecx, ecx
jnz    NiosInitNotFinished
```

See Also

Macros

NlmCall

#include<niosvxd.inc>

NlmCall FuncName [,<<Cparm0>, ... ,<CparmN>>]

Used to invoke an exported NLM API function, *NlmCall*
works similiarly to the VxdCall macro in that you can invoke both register and "C" based functions as well as pass "C" based stack parameters with the macro. If "C" parameters are passed with the macro, this macro will clean the stack before returning.

Note that "C" NLM functions preserve registers EBX,ESI,EDI,EBP with EAX used for function return information.

Example:

```
NlmCall NiosGetSystemDirectory,<<OFFSET32 retBuf>,retBufLen>
test    eax, eax
jnz     FuncFailed
```

NlmJmp

#include<niosvxd.inc>

NlmJmp FuncName

Used to jump to an exported NLM API function, *NlmJmp*
works similiarly to the VxdJmp macro.

Example:

```
NlmJmp NiosGetVersion
...Doesn't return
```