J

v3.11 HSM Conversion

Introduction

This appendix is a checklist of the major points required to convert a NetWare v3.11 server driver HSM to a NetWare v4.0 server driver HSM. Only drivers that were developed with the v3.11 MSM tools can be converted using these instructions.

The instructions here are in an overview form. Refer to the main document for details on implementing the conversion steps.

Conversion Process

1. Create a *DriverParameterBlock* structure in OSDATA. (Chapter 3 describes the *DriverParameterBlock*.)

2. PUBLIC declaration changes.

Note: All current required PUBLIC and EXTRN declarations are provided in the DRIVER.INC file.

Delete the following PUBLIC declarations. The *DriverParameterBlock* now passes all required information to the MSM.

In the OSDATA segment:

public DriverAdapterDataSpace
public DriverAdapterDataSpaceTemplate
public DriverConfigTable
public DriverKeywordText
public DriverKeywordTextLen
public DriverProcessKeywordTab

In the OSCODE segment:

public DriverAES
public DriverCallBack
public DriverISR
public DriverPoll
public DriverPostInit
public DriverReset
public DriverShutdown
public DriverMulticastChange
public DriverSend

The only PUBLIC declarations required are:

public DriverInit
public DriverRemove (new)

3. Delete the following variables from the OSDATA segment. These are now part of the *DriverParameterBlock* structure.

   DriverAdapterDataSpaceSize
   DriverFirmwareBuffer
   DriverFirmwareSize
   DriverEndOfChainFlag
   DriverNumKeywords
   DriverStatisticsTable

4. EXTRN declaration changes.

   Delete these EXTRN declarations. *MSMMaxFrameHeaderSize* is now an equate (see steps 13 & 14). *MSMHoldRcvEvent* and *MSMFastRcvEvent* have been replaced (see step 10).

   extrn MSMMaxFrameHeaderSize
   extrn MSMHoldRcvEvent
   extrn MSMFastRcvEvent

   Add these EXTRN declarations. These procedures are new. Refer to Chapters 6 & 7 for information on these procedures.

   extrn MSMAlloc:near
   extrn MSMAllocPages:near
   extrn MSMInitAlloc:near
   extrn MSMFree:near
   extrn MSMFreePages:near
   extrn MSMInitFree:near
   extrn MSMAllocateRCB:near
   extrn MSMDriverRemove:near
   extrn MSMParseCustomKeywords:near
   extrn MSMReadEISAConfig:near
   extrn MSMRegisterMLID:near
   extrn MSMReturnNotificationECB:near
   extrn MSMReadPhysicalMemory:near
   extrn MSMWritePhysicalMemory:near
   extrn <TSM>RegisterHSM:near
   extrn <TSM>ProcessGetRCB:near
   extrn <TSM>FastProcessGetRCB:near

   The following macros are new for drivers supporting hub management.

   MSMReturnNotificationECB
   MSMFastReturnNotificationECB

5. Configuration Table changes.

   - The Configuration Table version is 1.12.

   - The MSM sets *MLIDNodeAddress* to all FFh during the *<TSM>RegisterHSM*. If *MLIDNodeAddress* is still all FFh after returning from *MSMRegisterHardwareOptions*, the node address was not overridden and the HSM must set the field to correct node address.

     If the driver is for hardware that passes non-canonical addresses and the driver supports bit-swapping, the HSM must **not** use *MLIDNodeAddress* after *MSMRegisterMLID* is called. In this case the field may contain a canonical format node address. Instead the HSM should use the node address in *MSMPhysNodeAddress*.

   - Place zeroes in the *MLIDCardName, MLIDMajorVersion,* and *MLIDMinorVersion* fields; the MSM fills these fields using information in the NLM header, derived from the linker definition file.

   - Note that the *MLIDCardType* field at 40h has been changed to *MLIDReserved0*.

   - Add the *MLIDLookAheadSize* field at 4Ch. *MLIDReserved1* field is now 8 bytes.

   - Note that the *MLIDChannelNumber* has been added at A2h taking up 2 bytes ofthe *MLIDIOReserved* field (which is now only 6 bytes). Only multichannel adapters will use this field, other drivers place a zero value in this field.

   - Note the new bit assignments in the *MLIDModeFlags* and *MLIDFlags* bytes.

6. Statistics Table changes.

   - The Statistics Table version is 3.00

   - Additional standard counters and media specific counters have been added to the statistics table. Ensure that the HSM adds these to the table and increments the appropriate mandatory counters. See Chapter 3.

   - If there are more then 32 standard and media specific counters, a 2nd *CounterMask* is placed after the 32nd counter to indicate the status of the remaining counters. See Chapter 3.

7. The following procedures were added to the HSM driver specification (complete descriptions are in Chapter 5):

   DriverRemove (required)

   DriverPromiscuousChange (recommended)
   DriverEnableInterrupt (recommended)
   DriverDisableInterrupt (recommended)

   DriverStatisticsChange (optional)
   DriverRxLookAheadChange (optional)
   DriverManagement (optional)

8. Make the following adjustments in *DriverInit*:

   - Preserve EBP, EBX, ESI, and EDI.

   - Copy the stack pointer to the *DriverStackPointer* field in the *DriverParameterBlock*, then call *<TSM>RegisterHSM*.

   - *MSMScheduleInterruptTimeCallBack* has been shortened to *MSMScheduleIntTimeCallBack* to accommodate NLMLINK and NLMLINKP.

   - A new routine, *MSMReadEISAConfig*, is available. Refer to the routine's description in Chapter 7 for details.

   - Before returning successfully to the operating system, the HSM must call *MSMRegisterMLID* (this registration was transparent to the HSM previously).

   - The routines listed below have the zero flag set on successful returns. If the zero flag is clear, EAX points to the error message string that the HSM must display prior to exiting.

     MSMParseDriverParameters
     MSMRegisterHardwareOptions
     MSMSetHardwareInterrupt
     MSMScheduleIntTimeCallBack
     MSMScheduleAESCallBack
     MSMEnablePolling

   - If *DriverInit* fails to initialize the adapter hardware, it must call *MSMReturnDriverResources*.

9. Delete *DriverPostInit*. If the driver contained any code in this routine, execute that code after calling *MSMRegisterMLID* during *DriverInit*.

10. Make all necessary changes to the routines below, which have been changed due to the separation of the TSM module code from the general MSM. Replace <TSM> with EtherTSM, TokenTSM, RXNetTSM, PCN2LTSM, or FDDITSM.

MSMGetNextSend	*is now*	*<TSM>GetNextSend*
MSMGetRCB	*is now*	*<TSM>GetRCB*
MSMRcvComplete	*is now*	*<TSM>RcvComplete*
MSMFastRcvComplete	*is now*	*<TSM>FastRcvComplete*
MSMSendComplete	*is now*	*<TSM>SendComplete*
MSMFastSendComplete	*is now*	*<TSM>FastSendComplete*
MSMUpdateMulticast	*is now*	*<TSM>UpdateMulticast*

Two routines have been replaced:

MSMHoldRcvEvent	*is now*	*<TSM>ProcessGetRCB*
MSMFastRcvEvent	*is now*	*<TSM>FastProcessGetRCB*

The new routines process RCBs with completely filled data buffers just like their predecessors. The new routines also return a new RCB, unless no RCBs are available.

<TSM>ProcessGetRCB does not enable interrupts.
<TSM>FastProcessGetRCB may enable interrupts.

11. If the HSM uses shared RAM, observe the following constraints:

    - The HSM must call *MSMRegisterHardwareOptions* before accessing its shared RAM. On return from this routine, the logical address for the shared RAM is in the *MLIDLinearMemory0* and *MLIDLinearMemory1* fields.

    - If the HSM must access shared RAM to obtain information prior to registering the hardware options with the operating system, it may use *MSMReadPhysicalMemory* and *MSMWritePhysicalMemory*.

    - Bus-master logical-to-physical and physical-to-logical address conversions may still be made using *LogicalToPhysical* and *PhysicalToLogical*.

12. Realize that *<TSM>GetRCB* returns a pointer to a fragment structure rather than a continuous buffer. The HSM must be able to copy the received packet into fragment buffers.

13. The HSM should read *MSMMaxFrameHeaderSize* before each call to *<TSM>GetRCB* because the value may change dynamically. *MSMMaxFrameHeaderSize* is now an equate rather than a public variable, and is equal to the value in *MLIDLookAheadSize* plus the maximum media header size.

14. All MSM equates are offsets from the adapter data space passed to driver routines in EBP. The equates should be used as this example for *MSMTxFreeCount* shows:

```
inc  [ebp].MSMTxFreeCount
```

15. Note that *MSMAllocateRCB* is now a routine, not a macro.

16. If you can control interrupts at the adapter, you should implement the *DriverEnableInterrupt* and *DriverDisableInterrupt* routines described in Chapter 5. The MSM and TSM will call these routines at the appropriate times. The *MSMEnableHardwareInterrupt, MSMDisableHardwareInterrupt,* and *MSMDoEndOfInterrupt* macros should only be used if interrupts can not be enabled and disabled at the adapter. Also, the specification recommends that you not use *cli* and *sti* instructions in your HSM code.

17. Be sure to look over the example Linker Definition File and Server Driver Template provided with this document before creating a v4.x HSM. Apppendix A illustrates a Linker definition file. Appendix I contains a sample server driver HSM template. Include files are also provided on the diskettes.