1 Introduction

This application note discusses non-transparent (NT) bridging applications for the ExpressLane™ PEX 8696, PEX 8680 and PEX 8664 family of PCI Express Gen 2 Multi-Root switches. These switches have added NT-specific features and improvements over previous generations of PLX switches, providing improved systems compatibility and greater ease of programming. Covered in this document are: NT operating modes, NT-specific configuration registers, and a step-by-step example for setting up and demonstrating a basic NT bridge between two PC platforms using the PEX 8696RDK board.

Related Documents:

- ***ExpressLane PEX 8696-AA 96-Lane, 24-Port PCI Express Gen 2 Multi-Root Switch Data Book, version 0.95***
- ***ExpressLane PEX 8696-16U8D BB RDK Hardware Reference Manual, version 0.5***

2 NT-Specific Strapping Signals

The simplest way to configure the switch as an NT bridge is through the use of configuration strapping signals. These strapping signals are listed below. See the data book for detailed functional descriptions of each. Note that it is also possible to configure the switch by way of serial EEPROM or I^2C. Those options are not covered in this document.

Table 2-1. NT-Specific Strapping Signals

Strapping Signals	Description
STRAP_VS_MODE[2:0]	Configures the switch as either a single switch (Base Mode), or as multiple virtual switches isolated from one another (Virtual Switch Mode). NT port functionality is available in Base Mode only (STRAP_VS_MODE[2:0] = LLL).
STRAP_NT_ENABLE#	Pulling this strap low enables NT bridging on the selected port.
STRAP_NT_UPSTRM_PORT SEL[4:0]	Selects the port number to which NT bridging will be applied. Only one port of the switch can function as an NT bridge. Note also that the selected port number must exist in the station. For example, if Port 7 is selected as the NT bridge, Station 1 must be configured as x4,x4,x4,x4 or Port 7 will not exist.
STRAP_NT_P2P_EN#	Selects whether the NT endpoint appears directly on the same internal virtual PCI bus segment as other ports in the switch (Legacy NT Mode), or whether the NT endpoint appears behind a PCI-to-PCI bridge on its own dedicated virtual PCI bus segment. This mode was added to provide compatibility with PCI Express Base 2.1, which requires all ports on the internal Virtual PCI bus of the switch to present Type 1 headers to configuration software. The NT-Virtual Endpoint presents a Type 0 header. See section 14.2 in the data book for details.

3 NT-Specific Configuration Registers

3.1 Virtual Switch Debug Register (A30h)

The Virtual Switch Debug register exists at offset A30h of the upstream port. There are two bits in this register that need to be programmed. Bit A30h[24] enables enumeration of the NT-Virtual port, and is set to 1b (enabled) by default. Bit A30h[25] enables the NT-Link interface, and is cleared to 0b (disabled) by default. Bit A30h[25] must be set to 1b by way of serial EEPROM, I2C or the upstream port before attempting to enumerate the NT-Link port from the System Host PC. ***(Note: The NT-Link port will retry all configuration requests until A30h[25]=1.)***

3.2 NT-Virtual Port Registers Used for Addressing and Routing

NT-Virtual port registers specific to addressing and routing are listed in Table 3-1 below. For those who are familiar with previous generations of PLX switches, note that it is no longer necessary to program station shadow registers explicitly from EEPROM. EEPROM writes to NT-Virtual port registers are automatically broadcast to station shadow registers. This new feature greatly simplifies EEPROM programming.

Table 3.1 NT-Virtual Port Registers for Addressing and Routing

Port	Offset	Register	Description
NT-Virtual	10h	BAR0	32-bit BAR0 or lower half of 64-bit BAR0/1 register indexing internal Memory-mapped configuration registers. Programmed by system software at enumeration time.
NT-Virtual	14h	BAR1	Upper half of 64-bit BAR 0/1
NT-Virtual	D0h	Configuration BAR Setup	Bits [1:0]: Configures NT-Virtual BAR 0/1 00 = Disable BAR0 and BAR1 01 = Reserved 10 (default) = Enable BAR0 and disable BAR1 (32-bit BAR) 11= Enable 64-bit BAR (BAR0/1)
NT-Virtual	18h	BAR2	32-bit BAR2 or lower half of 64-bit BAR2/3. BAR2 type is programmed in D4h[2:1]
NT-Virtual	D4h	BAR2 Setup	Enables BAR2, determines the address range and prefetchable/non-prefetchable. Programmed by EEPROM or I2C before enumeration. Minimum window size is 1 MB.
NT-Virtual	C3Ch	BAR2 Translation	Link-side address. Must be a multiple of the address space size.
NT-Virtual	1Ch	BAR3	32-bit BAR3 or upper half of 64-bit BAR2/3. BAR3 type is programmed in D4h[2:1]
NT-Virtual	D8h	BAR3 Setup	Enables BAR3 when BAR3 Setup[31]=1. Defines upper 32 bits of address range for 64-bit BAR2/3.
NT-Virtual	C40h	BAR3 Translation	Link-side address. Must be a multiple of the address space size.
NT-Virtual	20h	BAR4	32-bit BAR4 or lower half of 64-bit BAR4/5. BAR4 type is programmed in DCh[2:1]
NT-Virtual	DCh	BAR4 Setup	Enables BAR4, determines the address range and prefetchable/non-prefetchable. Programmed by EEPROM or I2C before enumeration. Minimum window size is 1 MB.
NT-Virtual	C44h	BAR4 Translation	Link-side address. Must be a multiple of the address space size.
NT-Virtual	24h	BAR5	32-bit BAR 5 or upper half of 64-bit BAR4/5. BAR5 type is programmed in DCh[2:1]

Port	Offset	Register	Description
NT-Virtual	E0h	BAR5 Setup	Enables BAR5 when BAR5Setup[31]=1. Defines upper 32 bits of address range for 64-bit BAR4/5.
NT-Virtual	C48h	BAR5 Translation	Link-side address. Must be a multiple of the address space size.
NT-Virtual	D94h	LUT Entry 0	Requester ID look-up tables for routing of Configuration, Message, and Completion transactions. Note that this table is specific for Legacy NT mode. For NT P2P mode, a 32-entry LUT is used. See section 14.3 of the data book for overview. Programming details see section 15.15.1 [2:0] Function # [7:3] Device # [15:8] Bus # [29:16] Reserved (0) [30] No Snoop [31] Enable LUT Entry
NT-Virtual	D98h	LUT Entry 1	
NT-Virtual	D9Ch	LUT Entry 2	
NT-Virtual	DA0h	LUT Entry 3	
NT-Virtual	DA4h	LUT Entry 4	
NT-Virtual	DA8h	LUT Entry 5	
NT-Virtual	DACh	LUT Entry 6	
NT-Virtual	DB0h	LUT Entry 7	

3.3 NT-Link Port Registers Used for Addressing and Routing

Table 3.2 NT-Virtual Port Registers for Addressing and Routing

Port	Offset	Register	Description
NT-Link	10h	BAR0	32-bit BAR0 or lower half of 64-bit BAR0/1. Programmed by system software at enumeration time.
NT-Link	14h	BAR1	Upper half of 64-bit BAR0/1, when enabled.
NT-Link	E4h	Configuration BAR Setup	Bits[1:0] configures NT-Virtual BAR 0/1 00 = Disable BAR0 and BAR1 01 = Reserved 10 (default) = Enable BAR0 and disable BAR1 (32-bit BAR) 11= Enable 64-bit BAR (BAR1:0)
NT-Link	18h	BAR2	32-bit BAR2 or lower half of 64-bit BAR2/3. BAR2 type is programmed in E8h[2:1]
NT-Link	E8h	BAR2 Setup	Enables BAR2, determines the address range and prefetchable/non-prefetchable. Programmed by EEPROM or I2C before enumeration. Minimum window size is 1 MB.
NT-Link	C3Ch	BAR2 Translation	Virtual-side address. Must be a multiple of the address space size.
NT-Link	1Ch	BAR3	32-bit BAR3 or upper half of 64-bit BAR2/3. BAR3 type is programmed in E8h[2:1].
NT-Link	ECh	BAR3 Setup	Enables BAR3 when BAR3Setup[31]=1. Defines upper 32 bits of address range for 64-bit BAR3/2.
NT-Link	C40h	BAR3 Translation	Virtual-side address. Must be a multiple of the address space size.
NT-Link	20h	BAR4	32-bit BAR4 or lower half of 64-bit BAR4/5. BAR4 type is programmed in F0h[2:1].

<table>
<tr><th>Port</th><th>Offset</th><th>Register</th><th>Description</th></tr>
<tr><td>NT-Link</td><td>F0h</td><td>BAR4 Setup</td><td>Enables BAR4, determines the address range and prefetchable/non-prefetchable. Programmed by EEPROM or I2C before enumeration. Minimum window size is 1 MB.</td></tr>
<tr><td>NT-Link</td><td>C44h</td><td>BAR4 Translation</td><td>Virtual-side address. Must be a multiple of the address space size.</td></tr>
<tr><td>NT-Link</td><td>24h</td><td>BAR5</td><td>32-bit BAR5 or upper half of 64-bit BAR4/5. BAR5 type is programmed in F0h[2:1].</td></tr>
<tr><td>NT-Link</td><td>F4h</td><td>BAR5 Setup</td><td>Enables BAR5 when BAR5 Setup[31]=1. Defines upper 32 bits of address range for 64-bit BAR4/5.</td></tr>
<tr><td>NT-Link</td><td>C48h</td><td>BAR5 Translation</td><td>Virtual-side address. Must be a multiple of the address space size.</td></tr>
<tr><td>NT-Link</td><td>DB4h</td><td>LUT Entry 0_1</td><td>Requester ID look-up tables for routing of Configuration, Message, and Completion transactions. See section 16.15.1 of the data book for details.
<table>
<tr><th>LUT Entry 0, 2, 4, …</th><th>LUT Entry 1, 3, 5, …</th></tr>
<tr><td>[0] – Enable LUT Entry</td><td>[16] – Enable LUT Entry</td></tr>
<tr><td>[1] – No Snoop</td><td>[17] – No Snoop</td></tr>
<tr><td>[7:3] – Device #</td><td>[23:19] – Device #</td></tr>
<tr><td>[15:8] – Bus #</td><td>[31:24] – Bus #</td></tr>
</table></td></tr>
<tr><td>NT-Link</td><td>DB8h</td><td>LUT Entry 2_3</td><td></td></tr>
<tr><td>NT-Link</td><td>DBCh</td><td>LUT Entry 4_5</td><td></td></tr>
<tr><td>NT-Link</td><td>DC0h</td><td>LUT Entry 6_7</td><td></td></tr>
<tr><td>NT-Link</td><td>DC4h</td><td>LUT Entry 8_9</td><td></td></tr>
<tr><td>NT-Link</td><td>DC8h</td><td>LUT Entry 10_11</td><td></td></tr>
<tr><td>NT-Link
NT-Link
NT-Link</td><td>DCCh
DD0h
DD4h</td><td>LUT Entry 12_13
LUT Entry 14_15
LUT Entry 16_17</td><td></td></tr>
<tr><td>NT-Link</td><td>DD8h</td><td>LUT Entry 18_19</td><td></td></tr>
<tr><td>NT-Link
NT-Link
NT-Link</td><td>DDCh
DE0h
DE4h</td><td>LUT Entry 20_21
LUT Entry 22_23
LUT Entry 24_25</td><td></td></tr>
<tr><td>NT-Link</td><td>DE8h</td><td>LUT Entry 26_27</td><td></td></tr>
<tr><td>NT-Link</td><td>DCCh</td><td>LUT Entry 28_29</td><td></td></tr>
<tr><td>NT-Link</td><td>DF0h</td><td>LUT Entry 30_31</td><td></td></tr>
</table>

4 PEX 8696RDK NT Example

This chapter describes how to set up and demonstrate a simple NT bridge between two PC platforms using the PEX 8696 RDK. There are several parts to this process, including:

- RDK Hardware Setup - Cables, configuration modules, and strapping switches
- EEPROM Programming - To enable the NT-Link port enumeration and configure NT BARs
- Run-Time Programming - To set up translation addresses across the NT bridge
- Passing Traffic - Passing traffic from Local Domain to NT Domain, and vice-versa

4.1 RDK Hardware Setup

In this example, the PEX 8696 RDK is configured with Port 0 as the upstream port, x16, connected by mini-SAS cables to the Local host PC. This is the default configuration for the RDK, so there are no switches or

configuration modules to change here. See the Figure 2 of the PEX 8696 RDK Hardware Reference Manual (HRM) for a diagram of the cable connections needed.

For the NT-link side, Port 4 will be configured as the NT bridge, with a x8 connection to the NT-side, or System Host, platform. For this, Station 1 is configured as x8, x8. Also, configuration modules CM-124 and CM-109 must be installed at sockets J13 and J14, respectively. In this configuration, the System Host PC connects by way of SAS cables with Lanes 0-3 on connector IP13, and Lanes 4-7 on connector IP14. See Figure 7 in the HRM for a connection example.

Relevant switch settings, configuration modules, and connector/port lane assignments are listed in Tables 4-1 and 4-2 below.

Table 4-1. RDK Strapping Settings

DIP Switch	Switch #	PEX 8696 Strapping Signals	Description	Switch Setting
SW8	1-3	STRAP_VS_MODE[2:0]	Virtual Switch Enable	LLL (default)
SW2	1-2	STRAP_STN0_PORTCFG[1:0]	Station 0 Port Configuration	LH (default)
SW12	1-5	STRAP_UPSTRM_PORTSEL[4:0]	Selects Upstream Port #	LLLLL (default)
SW3	1-2	STRAP_STN1_PORTCFG[1:0]	Station 1 Port Configuration	HL (default)
SW14	1-5	STRAP_NT_PORTSEL[4:0]	Selects NT Port #	LLHLL (default)
SW14	6	STRAP_NT_ENABLE#	Enables NT Port	L
SW13	1	STRAP_NT_P2P_ENABLE#	Enables NT behind P2P bridge	H (default)

Note: L = switched closed, strap low, or zero. H = switch open, strap high, or one.

Table 4-2. Configuration Modules and Cable Lane Assignments

Port #	Config. Module Socket(s)	Config. Module Installed	Port Config	Lanes 0-3	Lanes 4-7	Lanes 8-11	Lanes 12-15
0 (Up)	J12	CM-090	X16 (cable)	IP4	IP3	IP2	IP1
4 (NT)	J13, J14	CM-124, CM109	X8 (cable)	IP13	IP14	n/a	n/a

4.2 EEPROM Programming

Certain configuration registers must be programmed by way of EEPROM or I2C before the ports can be enumerated on their respective platforms. In this example, EEPROM is used. EEPROM contents are programmed by way of the upstream port using the PEX Device Editor running on the Local Host platform. Following EEPROM programming, the PC must be restarted to enumerate the ports with the new NT-Virtual and NT-Link configuration register programming.

4.2.1 Virtual Switch Debug Register (A30h)

First, EEPROM must program Port0:A30h[25:24] = 11b, as described in 3.1 above.

4.2.2 BAR Setup Registers

Next, from EEPROM, the NT-Virtual BAR setup registers will be configured with BAR2 providing a 1 MByte wide prefetchable window mapping into the NT-Link (System Host) domain. Similarly, the NT-Link

BAR0/1 setup register will be configured with BAR0 providing memory-mapped access to switch registers and BAR2 providing a 1 MByte window mapped into the NT-Virtual (Local Host) domain.

4.2.3 Requester ID Lookup Tables

Completion transactions are routed using their respective Requester ID values. To support ID-based routing in a system where there may be multiple devices requesting access to the NT bridge, the switch provides routing ID look-up tables for both NT-Virtual and NT-Link ports. This translation mechanism is described in section 14.3 of the data book.

At least one entry in the LUT must be programmed in each of the NT-Virtual and NT-Link ports before passing traffic across the link. In this simple example, there is only one requester on each side of the link, that being the CPU of each system. Typically, the system CPU resides at Bus/Device/Function = 0/0/0. That may not be the case for all systems, however.

Additionally, there may be other devices in the PCI device hierarchy, besides the CPU, that generate traffic targeting the NT bridge. Each requester in the NT-Virtual side of the bridge requires an entry in the NT-Virtual Requester ID Translation Lookup Table (NT-Virtual: D94h-DB0h). Similarly, each requester in the NT-Link domain requires an entry in the NT-Link Requester ID Translation Lookup Table (NT-Link: DB4h-DF0h)

Determining Requester IDs

The PEX Device Editor provides a handy way of determining requester ID's in the system. To find the requester ID of the CPU in each system, launch the PEX Device Editor utility. In the upper left corner of the application window is a Device Selector dialog box. From the pull-down, select "All PCI Devices", as shown in Figure ## below. This lists all devices in the PCI hierarchy.

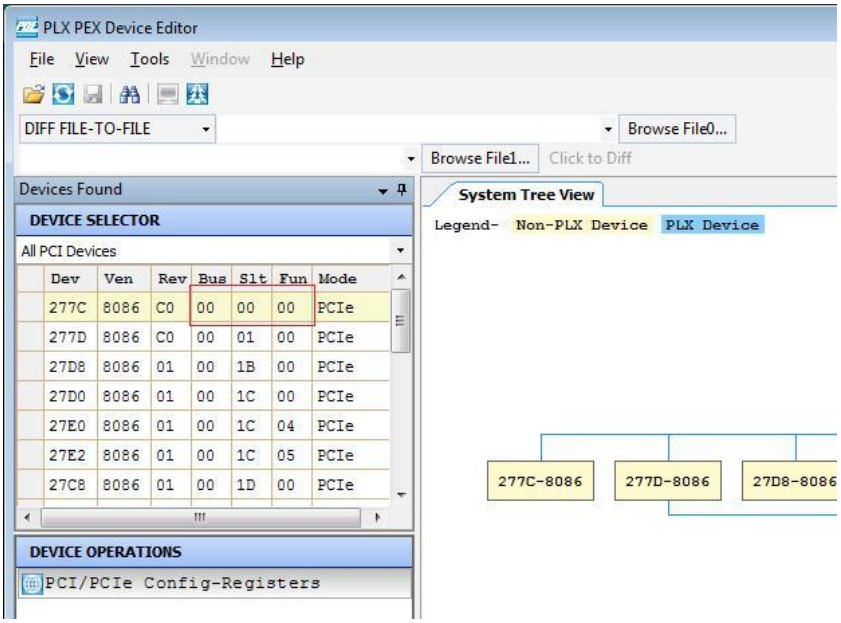

Figure 4-1. Displaying Bus/Slot/Function Numbers for all PCI devices

4.2.4 Serial EEPROM Content

For this example, the register content shown in Table 4-3 must be programmed into the serial EEPROM using PEX Device Editor. Once these values are written to the EEPROM, the Local Host PC can be restarted. Once the Local Host PC has restarted, the link-side, or System Host PC can be powered on as well. Verify that the links are trained by checking the Port0 and Port4 lane status LEDs on the RDK. Lane status can also be checked using PEX Device Editor on the Local Host platform. Figure 4-2 shows these EEPROM values as they appear in the PEX Device Editor.

Table 4-3. Example EEPROM Content

Port	Offset (hex)	Value (hex)	Comments
0	A30	03000000	Virtual Switch Debug – set A30[25:24] = 11b
NT-Link	E4	00000002	NT-Link BAR0/1 Setup – enables BAR0 access to internal registers (default setting)
NT-Link	E8	FFF00008	NT-Link BAR2 Setup - maps 1 MByte window size, prefetchable
NT-Link	C3C	00100000	NT- Link BAR2 Translation – placeholder value. Must be updated with actual address at run-time before passing traffic.
NT-Link	DB4	80000000	NT- Link Requester ID (B/D/F = 0/0/0)
NT-Virtual	D4	FFF00008	NT-Virtual BAR2 Setup - maps 1 MByte window size, prefetchable
NT-Virtual	C3C	00100000	NT-Virtual BAR2 Translation – placeholder value. Must be updated with actual address at run-time before passing traffic.
NT-Virtual	D94	80000000	NT-Virtual Requester ID (B/D/F = 0/0/0)

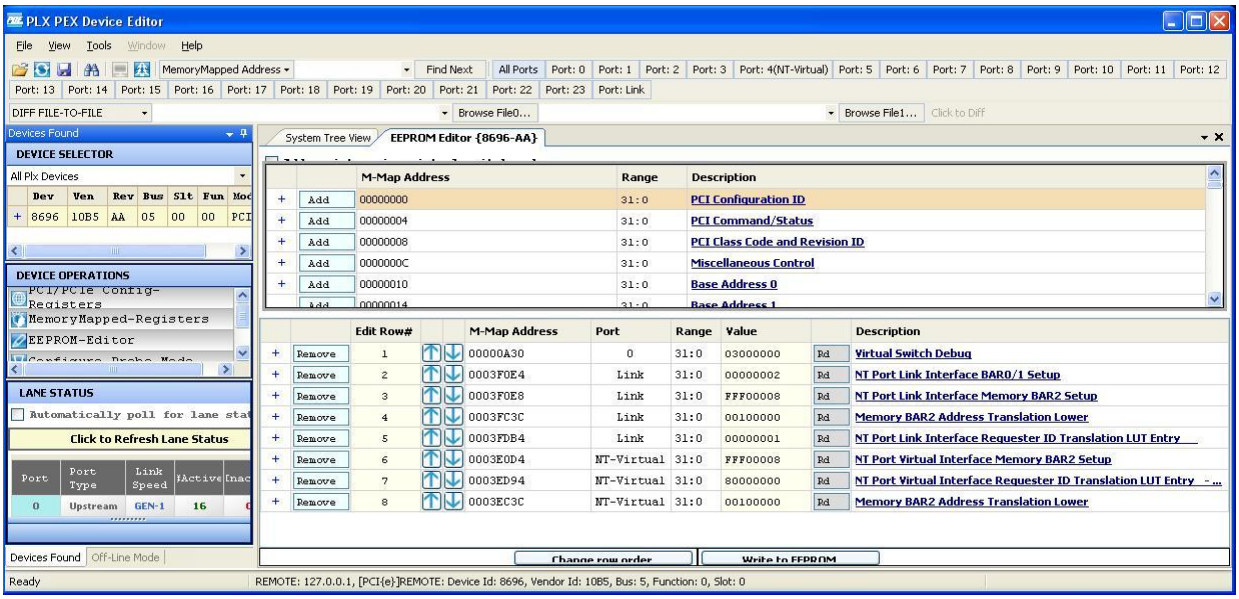

Figure 4-2. EEPROM contents as they appear in PEX Device Editor

4.2.5 Additional Registers for Back-to-Back NT Operation

For applications where two switches are joined back-to-back by way of their respective NT-Link ports, there will be no host processor to enumerate the NT Link registers. In such applications, the serial EEPROM or I2C can be used to enumerate the NT-Link ports. The table below lists sample EEPROM content required to enumerate BAR0 and BAR2 in a back-to-back NT link.

Table 4-4. Additional EEPROM content for back-to-back NT connection

Port	Offset (hex)	Value (hex)	Comments
NT-Link	04	00100006	NT-Link Command – Set Memory Space, Bus Master Enables
NT-Link	10	00200000	NT- Link BAR0
NT-Link	18	00100000	NT- Link BAR2

4.3 Run-Time Programming

Once the Local Host and System Host PCs are successfully linked to the RDK, the physical addresses of the memory buffers on each platform must be determined, and the BAR 2 Translation Address registers for NT-Virtual and NT-Link ports programmed accordingly.

4.3.1 Finding the Physical Address of the Local Host Buffer

For this step, the PlxMon utility is used. On the Local Host platform, launch the PlxMon utility from the PEX SDK (Start/All Programs/PEX SDK…./PlxMon). From the PlxMon application menu, select 'Command/Select A Device'. A dialog similar to Figure 4-3 should be displayed. From this dialog, select 'PLX PCIe NT Virtual-side port', as shown.

Next, in the PlxMon console, type 'vars <enter>'. The physical address of the Local Host memory is displayed as shown in Figure 4-4.

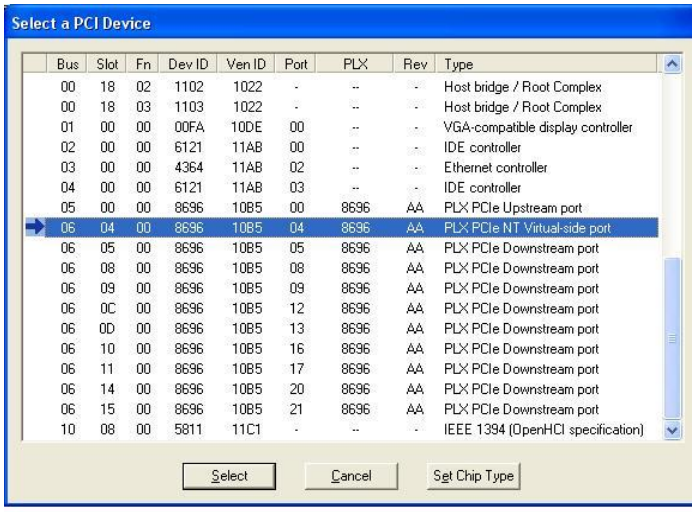

Figure 4-3. Selecting the NT-Virtual port device using PlxMon

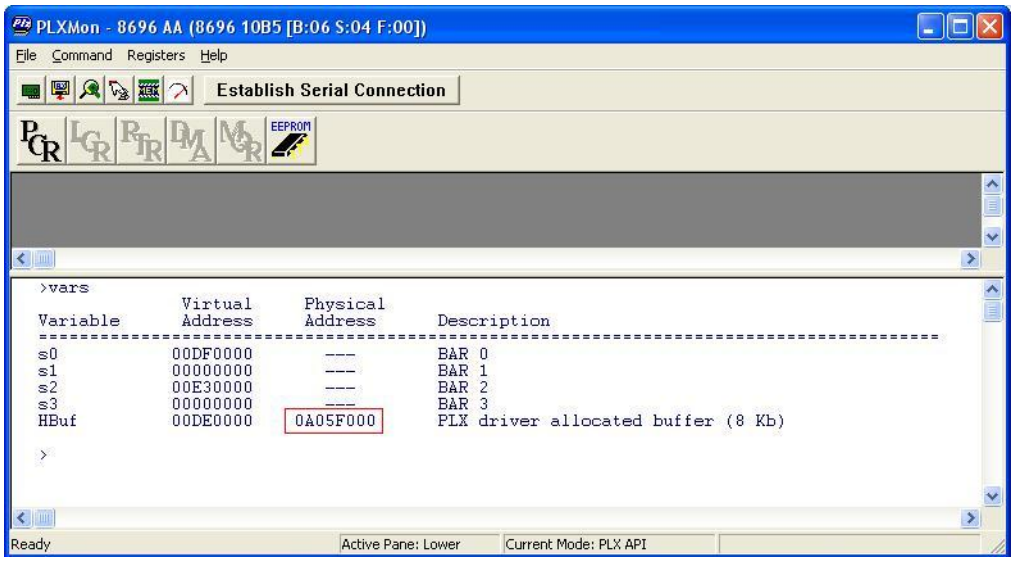

Figure 4-4. Displaying the physical address of the Local Host memory.

Note that the physical address of the Local Host memory may not lie on a multiple of the NT-Virtual BAR size. In this case, it does not. From the Local Host, use PEX Device Editor to program the NT-Link BAR2 Translation Lower register (NT-Link: C3Ch) using the logical AND of the Local Host memory address and the NT-Link BAR2 Setup register value (excluding bits 3:0). For this example, 0x0A05F000 AND 0xFFF00000 = 0x0A000000. Using PEX Device Editor, program this value into NT-Link:C3Ch.

The remaining bits of the Local Host physical address, in this case, 0x0005F000, are the offset added to the NT-Link BAR2 register when accessing Local Host memory from the System Host. We will come back to this later.

4.3.2 Finding the Physical Address of the System Host Buffer

On the System Host PC, launch PlxMon, same as was done for the Local Host PC in the previous step. From the device selector, select 'PLX PCIe NT Link-side port', as shown in Figure 4-5. In the PlxMon command console, enter 'vars' as before to display the physical address of the System Host memory, as shown in Figure 4-6. In this case, the physical address is 0x3ED26000.

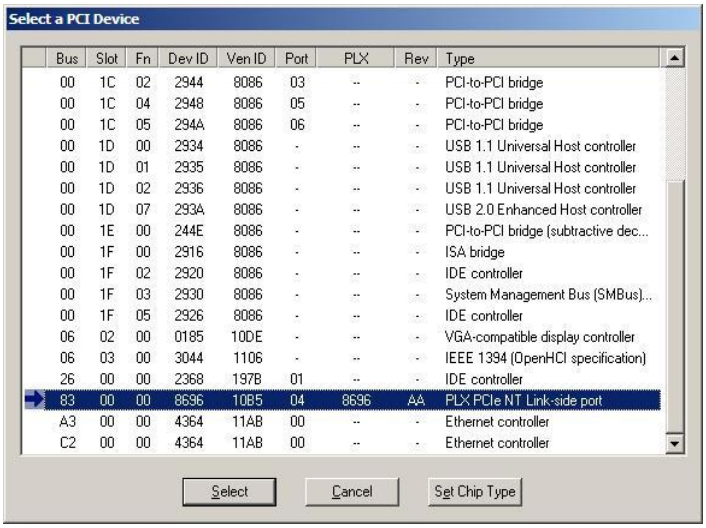

Figure 4-5. Selecting the NT-Link side port in the System Host platform.

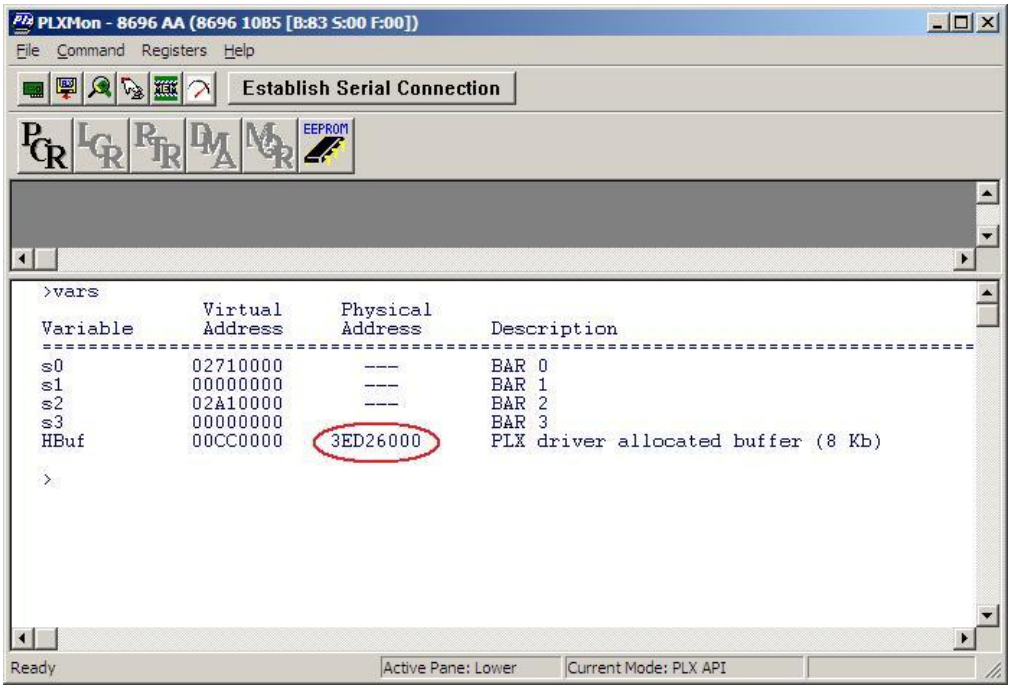

Figure 4-6. Finding the physical address of the System Host buffer.

Using this address, the NT-Virtual BAR2 Translation address (NT-Virt:C3Ch), is programmed with the logical AND of the NT-Virtual BAR2 Setup register, excluding bits 3:0, and the physical address of the memory in the System Host. So, 0xFFF00000 AND 0x3ED26000 = 0x3ED00000. Using PEX Device Editor, write 0x3ED00000 into the NT-Virtual BAR2 Translation register (NT-Virt:C3Ch).

The remaining physical address bits, in this case 0x00026000, are the offset added to the NT-Virtual BAR2 address when accessing the System Host memory from the Local Host.

4.4 Passing Traffic Across the NT Bridge

At this point, the switch is configured and ready for traffic in both directions. The System Host buffer is accessible from the Local Host, and the Local Host buffer is accessible from the System Host.

4.4.1 Local Host Accesses to System Host Domain

To demonstrate accesses to System Host memory from the Local Host PC, the PlxMon utility is again used. There are three basic steps. First, on the System Host, the buffer memory is initialized with a known pattern, other than zero. Next, we read and display the contents of the System Host buffer from the Local Host. Finally, from the Local Host, we write back to the System Host memory, and display the values written from the System Host side.

Initializing the System Host Memory

On the System Host platform, in the PlxMon application, select the memory dialog by clicking the 'Mem' icon. In the memory dialog, click the DMA Buffer radio button, then click 'Read Block' to display memory contents of the buffer (should be all zeroes). Using the Memory Fill button, initialize the memory to a known value, as shown in Figure 4-7 below. Leave this window open for the next steps.

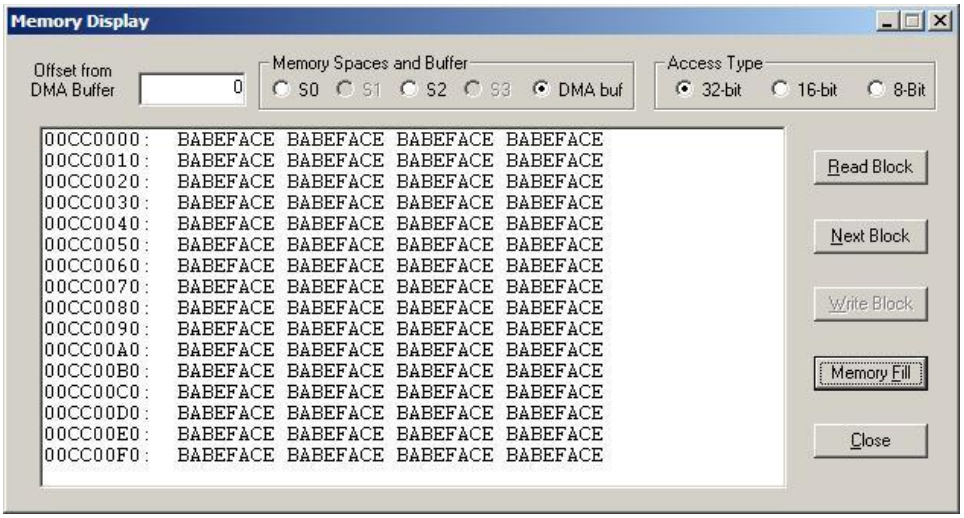

Figure 4-7. Initializing the System Host buffer

Reading the System Host Buffer from the Local Host

Next, from the Local Host PC, use PlxMon to display the contents of the System Host buffer. From the PlxMon console window, enter the command 'dl s2+26000'. You should now see the contents of the System Host buffer displayed as shown in Figure 4-8 below.

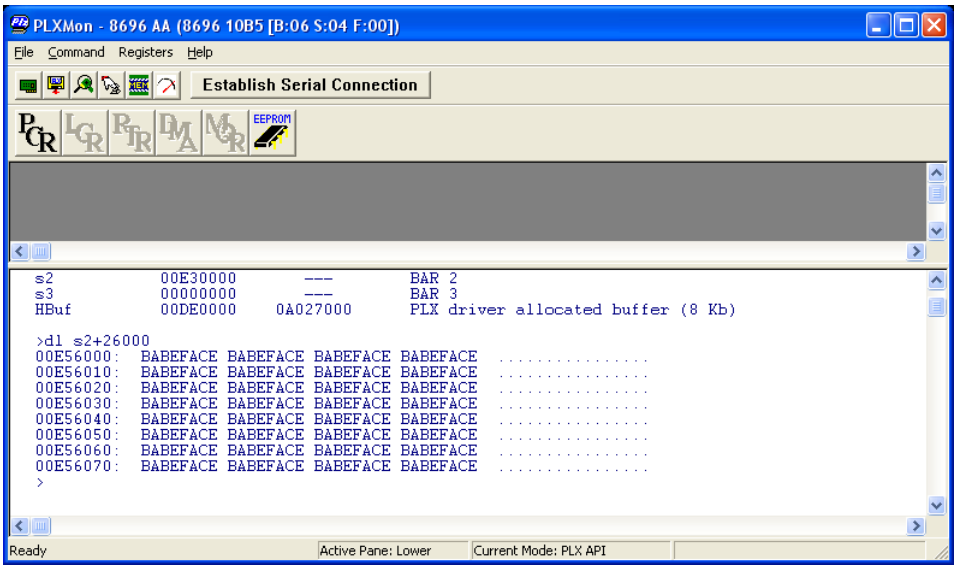

Figure 4-8. Displaying System Host buffer from the Local Host console.

Writing to System Host Memory from the Local Host

Finally, try writing a value back to the System Host memory using PLX mon. From the Local Host, in the PlxMon console, enter the command 'el s2+26000 00112233'. That will write the value 0x00112233 to the System Host buffer. On the System Host, using PlxMon, confirm the write by clicking 'Read Block' in the memory dialog. You should see the value written as shown in Figure 4-9. Alternatively, you can also display the system host buffer contents from the PlxMon console using the command 'dl hbuf'.

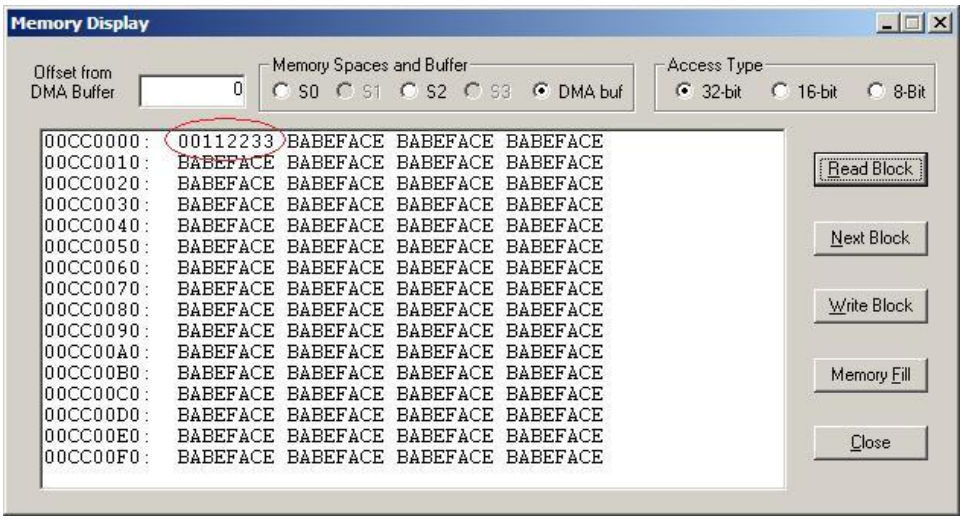

Figure 4-9. Displaying the System Host memory modified by the Local Host

4.4.2 System Host Accesses to Local Host Domain

To demonstrate System Host accesses to the Local Host memory, do the same steps as in 4.4.1 above, only reversed. First, on the Local Host, use PlxMon to initialize the local host memory, same as was done for the

System Host memory in 4.4.1.1 above. Next, from the System Host, using PlxMon, enter the command ‘dl s2+5F000’ to display the Local Host memory in the System Host console. Finally, from the System Host console, write a value back to the Local Host memory with the command ‘el s2+5F000 00112233’. Confirm the write on the Local Host by entering the command ‘dl hbuf’ in the PlxMon console.