LatticeECP3 and Broadcom 10 Gbps Physical/MAC Layer Interoperability

February 2012 Technical Note TN1218

Introduction

This technical note describes a Physical/MAC layer 10-Gigabit Ethernet interoperability test between a LatticeECP3™ device and the Broadcom BCM56800 network switch. The test exercises the Physical/MAC layer (up to MAC client interface) of the 10-Gigabit Ethernet protocol stack on the LatticeECP3 device.

Specifically, the document discusses the following topics:

- Overview of LatticeECP3 devices and the Broadcom BCM56800 network switch
- Physical/MAC Layer interoperability setup, testing, and results.

A significant aspect of the interoperability test needs to be highlighted:

The BCM56800 uses a CX-4 HiGig™ port, whereas the LatticeECP3 Serial Protocol Board provides a SMA connector. So a CX-4 to SMA conversion board is used as a physical medium interface to create a physical link between both boards. The SMA side of the CX-4 to SMA conversion board has four differential TX/RX channels (10 Gbps bandwidth total).

XAUI Interoperability

XAUI is a high-speed interconnect that offers a reduced pin count and the ability to drive up to 20” of PCB trace on standard FR-4 material. In order to connect a 10-Gigabit Ethernet MAC to an off-chip PHY device, an XGMII interface is used. The XGMII is a low-speed parallel interface for short range (approximately 2”) interconnects.

XAUI interoperability is based on the 10-Gigabit Ethernet standard (IEEE Standard 802.3ae-2002). Two XAUI link partners can be directly plugged into a XAUI backplane. Both boards are capable of generating and checking packets.

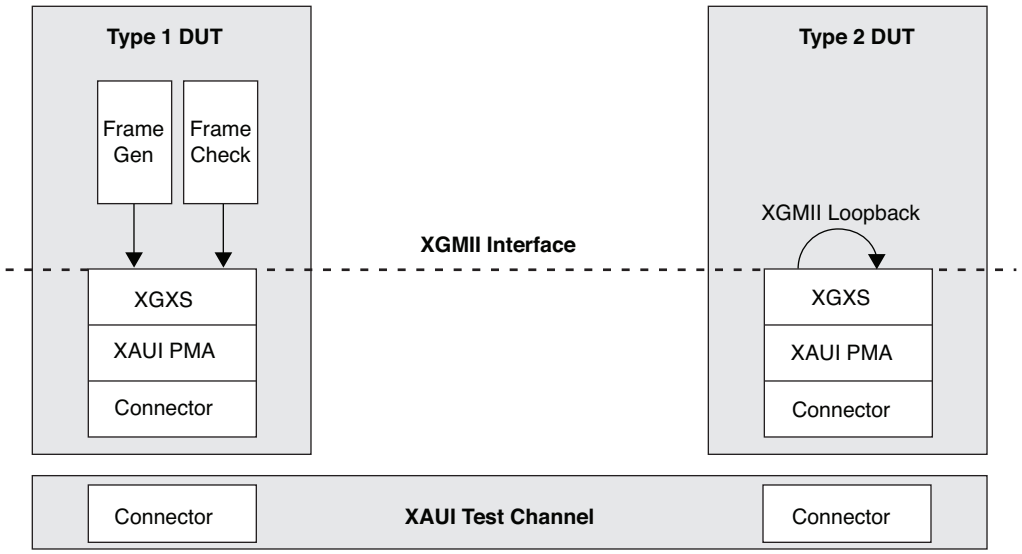
The board that sources packets is capable of keeping a detailed count of the number of packets transmitted while the sink board is capable of keeping detailed statistics on the number of packets received and errors associated with the packets. The XAUI backplane is also called the XAUI test channel. A typical test setup is shown in Figure 1.

Each reference station must be a line card that is directly plugged into the XAUI test channel. Both DUTs are required to have their own clock domain. Synchronous clocking (distributing a single clock to the two DUTs) is not allowed. Local management indicators on the DUT (reference stations) that provide information on link level errors such as CRC errors are also needed. A DUT is called a Type #1 device if it is capable of transmitting and checking packets.

A DUT is called a Type #2a device if it receives packets and does a RX to TX loopback through XGMII and sends the packets back to the transmitting station, which is a Type #1 device. The Type #1 device then checks the received packets for errors. Figure 1 shows a setup where one DUT is of Type #1 and the other is of Type #2a.

The LatticeECP3 and Broadcom BCM56800 interoperability exercises the LatticeECP3 Physical and MAC layers. Also, in this test, the XAUI backplane channel is replaced with a CX-4 to SMA conversion board as stated above.

Figure 1. Typical XAUI Interoperability Test Setup



LatticeECP3 Overview

LatticeECP3 Features

The LatticeECP3 FPGA family combines a high-performance FPGA fabric, high-performance I/Os and up to 16 channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic. The PCS logic can be configured to support numerous industry-standard, high-speed serial data transfer protocols.

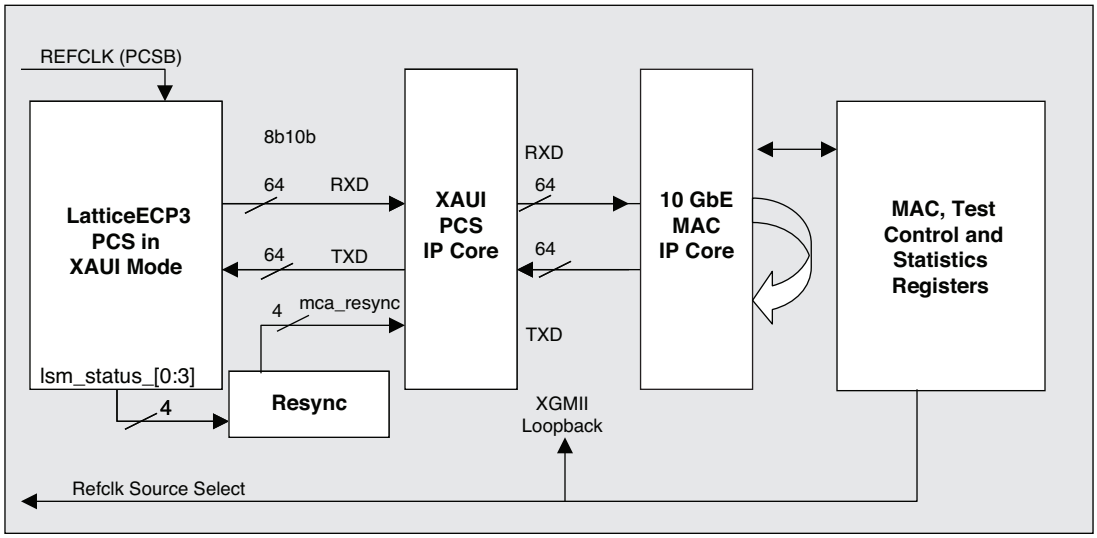
Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed, full-duplex serial data transfer at data rates up to 3.2 Gbps. The PCS logic in each channel can be configured to support an array of popular data protocols including GbE, XAUI, PCI Express, Serial RapidIO, CPRI, SD-SDI, HD-SDI and 3G-SDI.

LatticeECP3 in 10 GbE MAC Mode

The LatticeECP3 10 Gbps Ethernet MAC/PCS reference design shown in Figure 2 was used for the interoperability exercise. The design includes the LatticeECP3 SERDES/PCS, the LatticeECP3 XAUI IP core as well as the 10 Gb Ethernet MAC IP core. ORCAstra logic controls and monitors the MAC registers defined in IPUG39, 10 Gb+ Ethernet MAC IP Core User's Guide. An additional register (0x00A1A) was created to control the LatticeECP3 PCS REFCLK source (bit 0) optional XGMII far end loopback (bit 1).

For more information on the LatticeECP3 XAUI and 10 Gb Ethernet MAC IP cores, please refer to IPUG68, XAUI IP Core User's Guide and IPUG39, 10 Gb+ Ethernet MAC IP Core User's Guide. The LatticeECP3 SERDES/PCS in XAUI mode along with the XAUI and MAC soft IPs provide full compatibility from Serial I/O to the MAC client interface of the IEEE 802.3-2005 standard.

Figure 2. LatticeECP3 10 Gbps Ethernet MAC/PCS Reference Design



Transmit Path MAC Functionality (From LatticeECP3 MAC Client to XGMII):

- Data padding for short frames when FCS generation is enabled
- Generation of a pause frame to stop frame transmission when a pause frame is received by the receive MAC
- Implement link fault signaling logic and transmit appropriate sequences based on the remote link status

Transmit Path PCS Functionality (From LatticeECP3 XGMII to Line):

- LatticeECP3 XAUI IP core:
  - Transmit State Machine, which performs translation of XGMII, idles to proper ||A||, ||K||, ||R|| characters according to the IEEE 802.3ae-2002 specification
- LatticeECP3 SERDES/PCS:
  - 8b10b encoding and data serialization

Receive Path PCS Functionality (From Line to LatticeECP3 XGMII):

- LatticeECP3 SERDES/PCS:
  - Word alignment based on IEEE 802.3-2002 defined alignment characters
  - 8b10b decoding
  - Link State Machine functions incorporating operations defined in PCS Synchronization State Diagram of the IEEE 802.3ae-2002 specification
- LatticeECP3 XAUI IP core:
  - Multi-channel alignment
  - Clock Tolerance Compensation logic capable of accommodating clock domain differences
  - Receive State Machine compliant to the IEEE 802ae.3-2002 specification

Receive Path MAC Functionality (From XGMII to MAC Client):

- Checks the frame for a valid SOF and SFD symbols
- Determines whether the frame should be received by analyzing the Destination Address
- Determines the type of the frame by analyzing the Length/Type field
- Checks for any errors in the frame by recalculating the CRC and comparing it with the expected value

Other Control Functionality:

- resync block: This block resets the multi-channel alignment and CTC logic in the XAUI IP core.
- tx_powerup (not shown): This block implements the TX Reset State machine shown in TN1176, LatticeECP3 SERDES/PCS Usage Guide.
- rx_powerup (not shown): This block implements the RX Reset State machine shown in TN1176, LatticeECP3 SERDES/PCS Usage Guide.

Broadcom BCM56800 Overview

BCM56800 Features

The BCM56800 network switch is a high density, 10-Gigabit Ethernet switching chip solution with 20 ports. Each of these flexible ports supports 10-Gigabit Ethernet or 1-Gigabit Ethernet. Additionally, the BCM56800 integrates all the SERDES required to interface to applicable copper and fiber physical interfaces. The integrated SERDES functionality includes 10-Gbps XAUI interfaces and 1-Gbps SGMII PHY interfaces. The integrated SERDES complies with the CX-4 standard and PICMG3.1 standard, which ensures interoperability with Ethernet line cards in an Advanced TCA chassis.

BCM56800 10 GbE/HiGig Ports

The BCM56800 has twenty 10 GbE/1 GbE ports. The BCM56800 is based on StrataXGS™ field-proven, robust architecture. It has integrated high-performance SERDES: integrated XAUI SERDES for all twenty 10-GbE ports, and it uses single SERDES lane per port at GbE speeds. The device supports 200-Gbps switching capacity at line rate.

Test Equipment

The equipment used in the interoperability test is described below.

Broadcom BCM56800 Network Switch

Figure 3 shows the BCM56800 network switch.

Figure 3. Broadcom BCM56800 Network Switch

One can configure the Broadcom ports by connecting its serial port to a PC and starting a HyperTerminal session. Figure 3 shows a serial cable connected to the serial port at the back of the BCM56800.

Figure 3 also shows a CX-4 connector inserted into one 10 GbE/HiGig port available on the front right side of the BCM56800.

This port, referred to as xe0/hg0, was selected for the interoperation with the LatticeECP3 device. It was configured in XAUI mode.

LatticeECP3 Serial Protocol Board

The LatticeECP3 Serial Protocol Board provides a stable yet flexible platform designed to help the user quickly evaluate the performance of the LatticeECP3 SERDES and FPGA, or aid in the development of custom designs. The LatticeECP3 Serial Protocol Board features:

- PCI Express x4 edge connector interfaces
  - Allow demonstration of PCI Express (x4) interfaces
  - x4 is non-compliant but will demonstrate x4 functionality with an open-frame motherboard
- Allow control of SERDES PCS registers using the Serial Client Interface (ORCAstra)
- Serial ATA interfaces for host and target configurations
- RJ45 interface to 10/100/1000 Ethernet
- On-board Boot Flash
  - 64M Serial SPI Flash
  - Parallel Flash via MachXO™ PLD programming bridge
- DDR2 and DDR3 memory components
- Switches, LEDs, displays for demo purposes
- Several debug and analysis connections
- Input connection for lab-power supply
- Power connections and power sources
- ispVM programming support
- On-board and external reference clock sources

Figure 4 shows the LatticeECP3 10 Gbps Ethernet MAC/PCS reference design and other components on the LatticeECP3 Serial Protocol Board. All board components are described in detail in EB44, LatticeECP3 Serial Protocol Board - Revision D User's Guide. Table 1 provides a description of the reference design signals accessible on the LatticeECP3 Serial Protocol Board.

Figure 4. LatticeECP3 Serial Protocol Board

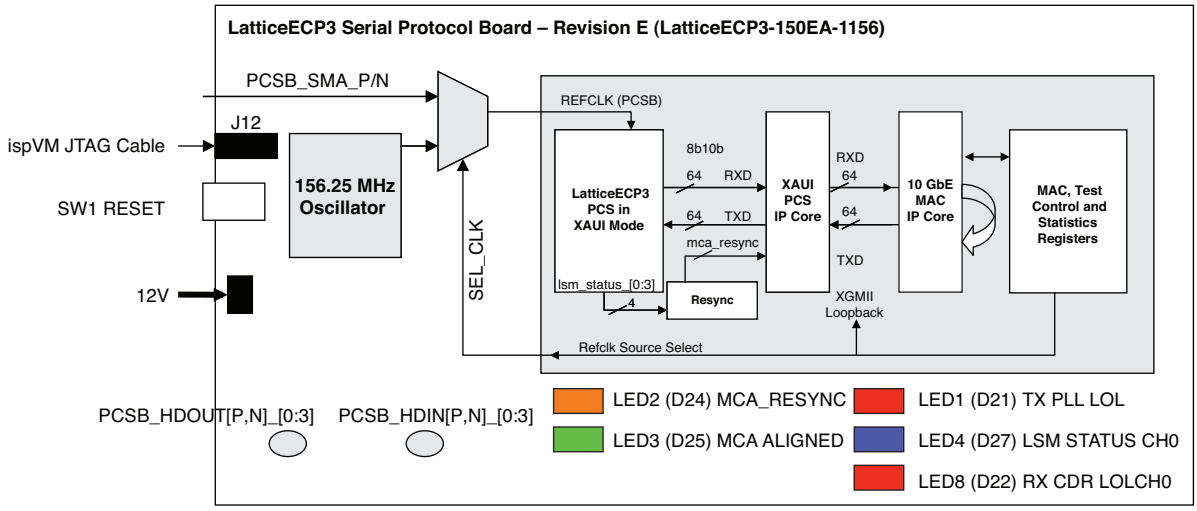


Table 1. Reference Design Signals

LatticeECP3 Signal Name	Signal Type	LatticeECP3 Serial Protocol Board Version C (or Newer) Connection	Description
General Signals			
reset_n	I	SW1 Push Button	FPGA Global active low reset
Reference Design Signals			
PCS TX PLL LOSS OF LOCK	O	LED1 (D21)	Red LED. LatticeECP3 TX PLL loss of lock indication. The LED does not glow when a valid 156.26 MHz reference clock is provided to the LatticeECP3 PCS.
PCS CH0 RX CDR LOSS OF LOCK	O	LED8 (D22)	Red LED. LatticeECP3 channel 0 RX CDR loss of lock indication. The LED does not glow when valid 3.125 Gbps data is provided to the LatticeECP3 Channel 0 SERDES inputs.
XAUI PCS IP MCA ALIGNED	O	LED3 (D25)	Green LED. This LED glows when the XAUI IP core Multi-channel Alignment Logic is aligned to valid //A// columns.
XAUI PCS IP MCA_RESYNC	O	LED2 (D24)	Yellow LED. This LED glows the XAUI PCS IP core Multi-channel Alignment Logic is being reset.
PCS CH0 RX LINK STATE MACHINE OK	O	LED4 (D27)	Blue LED. Indicates that the LatticeECP3 channel RX XAUI link state machine is successfully synchronized to incoming Ethernet traffic. The LED glows when valid 3.125 Gbps Ethernet data is provided to the LatticeECP3 Channel 0 SERDES inputs.
JTAG Signals			
tck	I	To J12 JTAG Header	Connect the ispVM USB download cable to this header
tdi	I		
tdo	O		
Tms	I		

Table 1. Reference Design Signals (Continued)

LatticeECP3 Signal Name	Signal Type	LatticeECP3 Serial Protocol Board Version C (or Newer) Connection	Description
PCS Quad			
PCSC_REFCLKP/N	I	Output of U18B on-board clock MUX	LatticeECP3 PCS Quad C reference clock. The source of this clock is controlled by register 0x0A1A, bit 0: Set bit 0 to 0 (default) to select the Y2 125 MHz on-board oscillator. Set bit 0 to 0 to select the J30/J34 SMA clock inputs.
PCSB_HDOUT[P,N]_[0:3]	0	SMAs	Four differential SMA pairs associated with PCS Quad B.
PCSB_HDIN[P,N]_[0:3]	I	SMAs	Four differential SMA pairs associated with PCS Quad B.

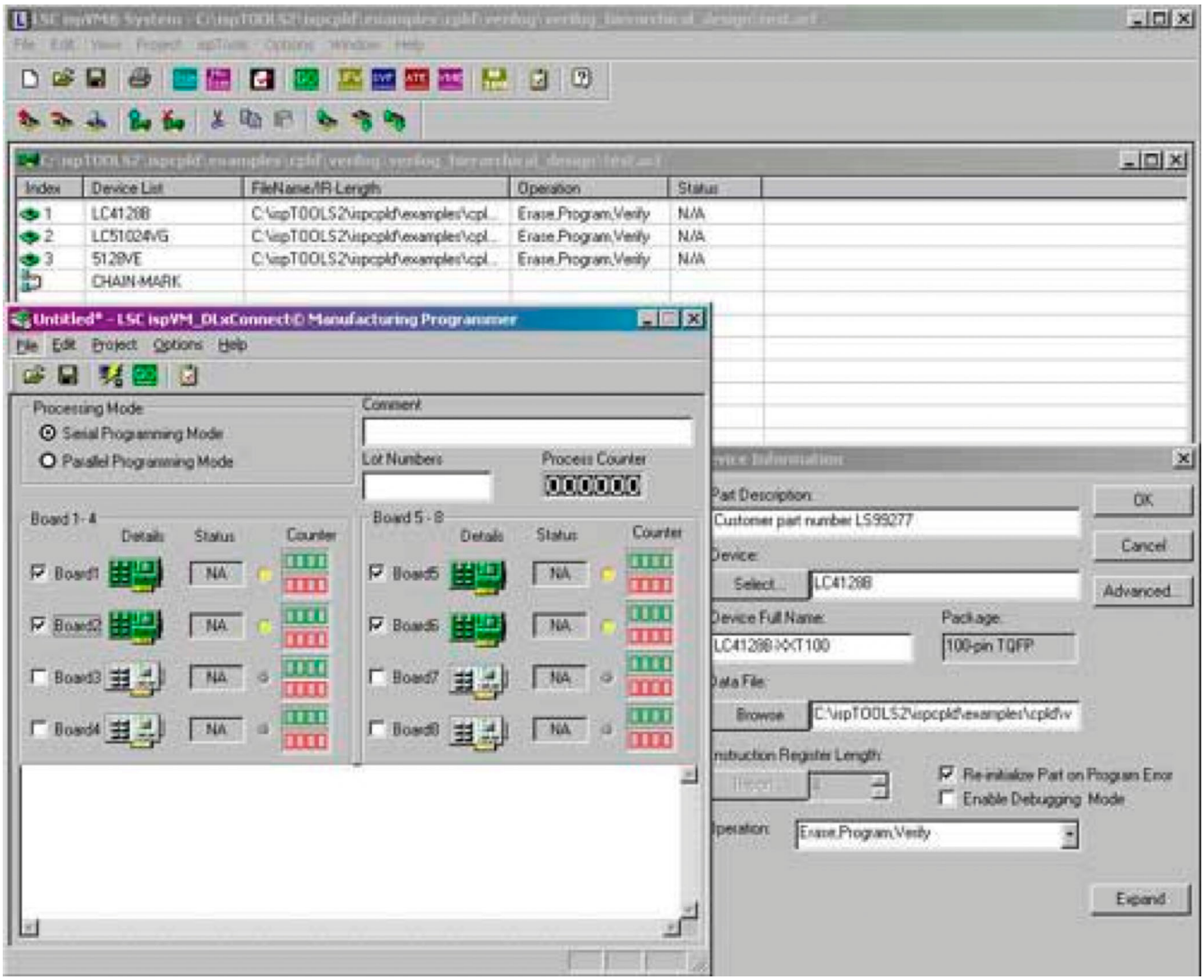
ispVM™ System

The ispVM System software is included with the Lattice ispLEVER software, and is also available as a stand-alone device programming manager. The ispVM System is a comprehensive design download package that provides an efficient method of programming in-system programmable devices using JEDEC and bitstream files generated by Lattice Semiconductor, and other, design tools. This complete device programming tool allows the user to quickly and easily download designs through an ispSTREAM to devices and includes features that facilitate ispATE™, ispTEST™, and ispSVF programming as well as gang-programming with DLxConnect™.

The ispVM System software is used in this interoperability test to download the LatticeECP3 bitstream, which configures the device in 10-Gigabit Ethernet mode (XAUI).

Figure 5 shows a screen shot of the ispVM System software.

Figure 5. ispVM System



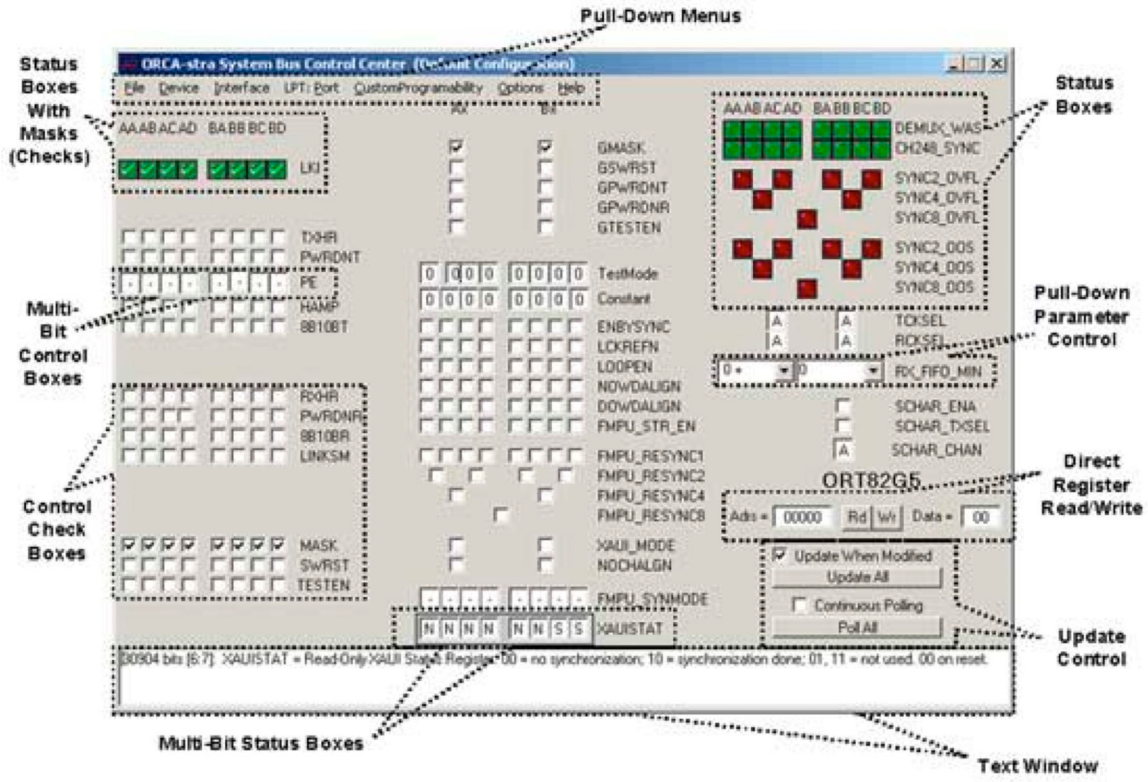
ORCAstra System

The Lattice ORCAstra software is a PC-based graphical user interface that allows the user to configure the operational mode of an FPGA by programming control bits in the on-chip registers. This helps users quickly explore configuration options without going through a lengthy re-compile process or making changes to their board.

Configurations created in the GUI can be saved to memory and re-loaded for later use. A macro capability is also available to support script-based configuration and testing. The GUI can also be used to display system status information in real time. Use of the ORCAstra software does not interfere with the programming of the FPGA portion of the LatticeECP3.

Figure 6 is a screen shot of the ORCAstra software.

Figure 6. ORCAstra System



Interoperability Testing

This section provides details on the 10 GbE MAC interoperability test between the LatticeECP3 device and the Broadcom BCM56800 network switch. The purpose of this test is to implement interoperability between one Type #2 DUT (LatticeECP3) and one Type #1 DUT (BCM56800).

The test has the following characteristics:

- Independent (asynchronous) +/- 100 ppm clock sources clock the LatticeECP3 and BCM56800 devices. For these particular devices, the data rate across four lanes is 4*8/10*20*F, where F is the source clock frequency. The data rate in XAUI mode is 10 Gbps. This means an independent clock source of 156.25 MHz (+/- 100 ppm) or larger clocks each device.
- The Broadcom switch transmits continuous Ethernet frames to the LatticeECP3 device
- The LatticeECP3 device loops the data at its MAC client interface back to the Broadcom switch.

By the end of the test:

- The BCM56800 device visual window RX ERR counter should remain at zero
- The LatticeECP3 device visual window RX ERR counter should remain at zero
- The BCM56800 device visual window counters should report as many TX packets generated as RX packets received
- The amount of test time should be longer than 30 minutes to ensure the error rate is less than 10-12 with 99.999999% accuracy

Interoperability Hardware Test Setup

The setup includes:

- The Broadcom BCM56800 network switch
- LatticeECP3 Serial Protocol Board with LFE3-150EA 7FN1156CES device on a socket
- A CX-4 to SMA conversion board was used as a physical medium interface to create a physical link between both boards. The SMA side of the CX-4 to SMA conversion board has four differential TX/RX channels, or 16 SMA connectors for a total bandwidth of 10 Gbps (12.5 Gbps aggregated rate). All four differential TX/RX channels were connected to the LatticeECP3 side (as shown in Figure 7).
- A PC for software control/monitoring
- On-board 156.25 MHz differential LVDS clock oscillator to provide the reference clock to the LatticeECP3 PCS/SERDES quad. The Broadcom BCM56800 board also contains a built-in oscillator.
- Cables
  - 16 SMA for LatticeECP3 SERDES channels 0 through 3
  - CX-4 for BCM56800 HiGig port xe0/hg0
  - ispVM JTAG cable for downloading the LatticeECP3 bitstream and ORCAstra GUI access
  - Serial cable for BCM56800 HyperTerminal access

Figure 7 shows the Broadcom BCM56800 board, the LatticeECP3 Serial Protocol Board, and the Tyco backplane connections.

Figure 8 is a block diagram of the test setup.

Figure 7. Board Connections

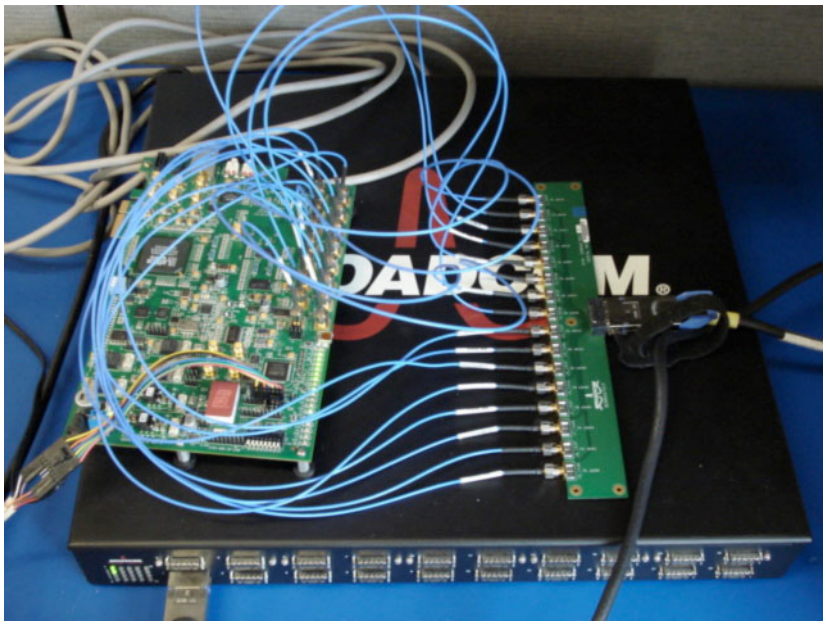
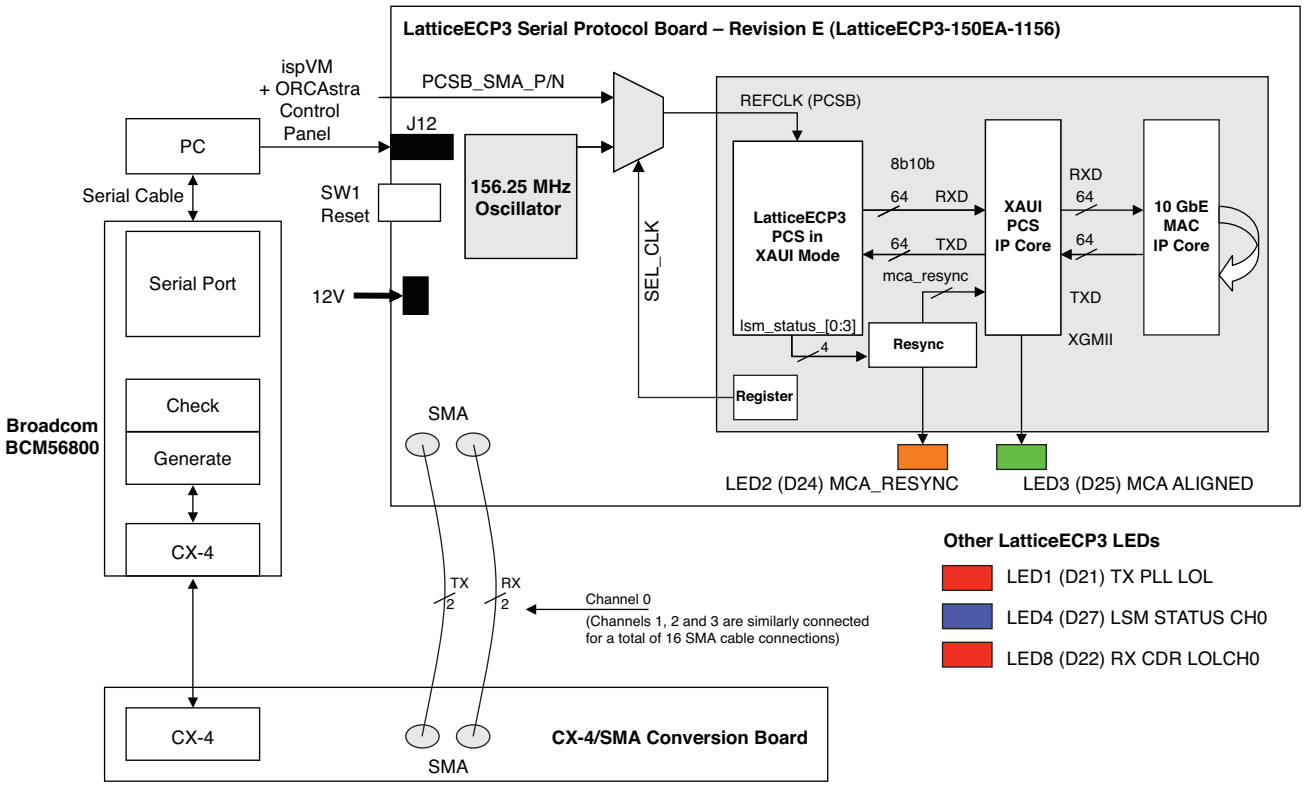


Figure 8. Test Setup Block Diagram



Test Description

This section describes how each interoperability partner is set up for 10-Gigabit Ethernet Physical/MAC Layer interoperability.

Broadcom BCM56800 Setup

The BCM56800 switch generates and checks full protocol compliant Ethernet packets. The BCM56800 is configured in 10-Gigabit Ethernet.

Figure 9 illustrates the sequence of commands performed in a HyperTerminal from startup to configure HiGig port 0 (xe0) of BCM56800 in 10-Gigabit Ethernet, while disabling auto-negotiation. To prevent port xe0 statistics counters from overflowing, a few lines are added to create Ethernet traffic on port xe1, and then redirect it to xe0.

As shown in Figure 9, xe1 generates continuous Ethernet packets defined by the content of the bc_DA_1022 file and redirects the content to port xe0. Port xe0 then continuously outputs this data on channel 0 of its HiGig port. bc_DA_1022 is a VLAN tagged packet with 1022 bytes of data. The destination address in bc_DA_1022 is set to 00.00.00.00.00.02.

Figure 9. BCM56800 10-Gigabit Ethernet Configuration Commands

```
'SETUP BC BOARD
'TRANSMIT PACKETS FOR 3600 seconds (1hr)
rc
'ENABLE VLAN PACKETS ON ALL PORTS
vlan remove 1 pbm=0x00000000001fffff
vlan add 1 pbm=0x00000000001fffff ubm=0x000000
vlan show
'SETUP XE0 AND XE1 IN 10 GBE MODE
'REDIRECT XE1 TRAFFIC TO XE0
fp qset clear
fp qset add InPorts
fp group create 1 1
fp entry create 1 1
fp qual 1 InPorts 0x1 0xffffff
fp action add 1 RedirectPbmp 0x2
fp entry install 1
fp entry create 1 2
fp qual 2 InPorts 0x2 0xffffff
fp action add 2 RedirectPbmp 0x1
fp entry install 2
port xe0 speed=10000 AN=OFF
port xe1 speed=10000 AN=OFF
port xe1 lb=phy
ps
clear counters
'RUN TRAFFIC FOR 1 HR, THEN SHOW COUNTERS
tx 4 pbm=xe1 file=bc_DA_1022
sleep 3600
port xe1 lb=none
sleep 5
show counters
```

LatticeECP3 Serial Protocol Board Setup

The internal 156.25 MHz clock oscillator sources the LatticeECP3 PCS reference clock to the PCS/SERDES quad. The reference clock is multiplied internally by 20 to achieve a 10 Gbps data rate (12.5 Gbps aggregated rate)

In the RX direction, the LatticeECP3 SERDES recovers the packets from the Broadcom BCM56800 network switch and the XAUI IP core converts them into XGMII format. The MAC interface presents these frames to the client interface along with statistical information that ORCAstra scripts can recover. Any unfiltered frame is looped back at the client interface and re-transmitted to the Broadcom BCM56800 network switch.

LatticeECP3 PCS Auto Configuration (.txt) file

Appendix A lists the auto configuration file settings for the LatticeECP3 PCS used in the LatticeECP3 10 Gbps Ethernet MAC/PCS reference design.

ORCAstra Setup

After the Serial Protocol board is powered-up, and the LatticeECP3 bitstream is downloaded, the following steps explain the procedure for configuring the MAC registers via ORCAstra:

1. Start ORCAstra from the ispLEVER installation directory.

2. Select **Interface -> 3. ispVM JTAG USB Interface**. If the **Select Target JTAG Device** window comes up, select the first device and click **OK**.
3. From the ORCAstra main window, click on **ORCAstra -> CustomProgrammability -> MACRO**.
4. From new window, **File –> Open -> < PROJECT_PATH>\ unicast_settings.fpm**. This file is shown in Appendix B. It configures the MAC in UNICAST mode and the local address is set to 00 00 00 00 00 02. This matches the destination address defined in the Broadcom script. Click **Run**. The script also performs a MAC client RX to TX loopback and prevents CRC removal in RX direction and insertion in TX direction.
5. Following the running of this script as well as the Broadcom script, the Broadcom box and LatticeECP3 LEDs should indicate a proper link.

Upon the termination of the test, the following steps are used to record the LatticeECP3 MAC statistics counters:

1. From the ORCAstra main window, click on **ORCAstra -> CustomProgrammability -> Scripts -> VBScripts**.
2. From the new window, select **File -> Open -> < PROJECT_PATH>\reg_stats_10ge.vbs**. This file is shown in Appendix C. Click **Run**. This results in an output window similar to the one shown in Figure 10.

Figure 10. Result of Running reg_stats_10ge.vbs Script

```
RX VLAN PKT = 2025327954
RX PAUSE PKT = 0
RX FILTERED(CONTROL,BC, MC) PKTS = 0
RX UNSUPPORTED CODE PKT = 0
RX BROADCAST PKT = 0
RX MULTICAST PKT = 0
RX LENGTH ERROR OR Short PKT = 0
RX LONG  PKT = 0
RX CRC ERROR = 0
RX PKT DISCARDED (UNIC,CONTROL,MC,BC,Short)= 0
RX PKT IGNORED = 0
RX FRAGMENT PKT = 0
RX JABBER  PKTS = 0
RX = 64 BYTE PKT = 0
RX  65-127 BYTE PKT = 0
RX  128-255 BYTE PKT = 0
RX  256-511 BYTE PKT = 0
RX  512-1023 BYTE PKT = 0
RX  1024-1518 BYTE PKT = 2025327954
RX  UNDERSIZE PKT = 0
RX  UNICAST PKT = 2025327954
RX  RECEIVED PKT = 2025327954
RX < 64-byte GOOD CRC PKT = 0
RX  > 1518-byte GOOD CRC PKT = 0
RX  1519-2047 BYTE PKT = 0
RX  2048-4095 BYTE PKT = 0
RX  4096-9216 BYTE PKT = 0
RX  9217-16383 BYTE PKT = 0
```

```
TX TERMINATE ERR PKT = 0
TX UNDERRUN PKT = 0
TX CRC ERR PKT = 0
TX LENGTH ERROR PKT = 0
TX LONG PKT = 0
TX MULTICAST PKT = 0
TX BROADCAST PKT = 0
TX Control PKT = 0
TX JUMBO PKT = 0
TX PAUSE PKT = 0
TX VLAN TAGGED PKT = 2025327954
TX OK PKT = 2025327954
TX = 64-byte PKT = 0
TX 65-127 byte PKT = 0
TX 128-255 byte PKT = 0
TX 256-511 byte PKT = 0
TX 512-1023 byte PKT = 0
TX 1024-1518 byte PKT = 2025327954
TX > 1518-byte PKT = 0
TX FRAME ERROR PKT = 0
TX 1519-2047 byte PKT = 0
TX 2048-4095 byte PKT = 0
TX 4096-9216 byte PKT = 0
TX 9217-16383 byte PKT = 0
```

Results

Figure 11 illustrates the section of the HyperTerminal output that resulted from running the last few commands in the BCM56800 10-Gigabit Ethernet Configuration sequence of Figure 9.

As shown in Figure 11, HiGig ports 0 (xe0) and 1 (xe1) of BCM56800 were configured for 10-Gigabit Ethernet.

The last “show counter” command reports the status of BCM56800 port xe0 TX and RX packet (GTPKT.xe0 and GRPKT.xe0) and byte (GTBYT.xe0 and GRBYT.xe0) counters. Figures 10 and 11 do not show any errors. This is an indication that the test between the two ports ran error-free. Additionally, the TX and RX packet counters in Figures 10 and 11 all show an identical number of frames (2025327954) recorded.

Figure 11. Output of BCM56800 Configuration Script

```
BCM.0> ps
      ena/  speed/  link  auto    STP                    lrn  inter   max  loop
port  link  duplex  scan  neg?   state   pause  discrd  ops   face frame  back
 xe0  up    10G FD    SW   No    Forward          None    FA  XGMII 16360
 xe1  up    10G FD    SW   No    Forward  TX RX   None    FA  XGMII 16360   PHY
 xe2  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe3  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe4  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe5  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe6  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe7  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe8  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
 xe9  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe10  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe11  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe12  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe13  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe14  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe15  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe16  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe17  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe18  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
xe19  down    -       SW   Yes   Forward  TX RX   None    FA  XGMII 16360
BCM.0> clear counters
BCM.0> tx 4 pbm=xe1 file=bc_DA_1022
BCM.0> sleep 3600
Sleeping for 3600 seconds
BCM.0> port xe1 lb=none
BCM.0> sleep 5
Sleeping for 5 seconds
BCM.0> show counters
RUC.xe0          :          2,025,327,954      +2,025,327,954
ITPKT.xe0        :          2,025,327,954      +2,025,327,954
IT1518.xe0       :          2,025,327,954      +2,025,327,954
ITBYT.xe0        :      2,114,442,383,976  +2,114,442,383,976
IR1518.xe0       :          2,025,327,954      +2,025,327,954
IRPKT.xe0        :          2,025,327,954      +2,025,327,954
IRBYT.xe0        :      2,114,442,383,976  +2,114,442,383,976
```

Summary

In conclusion, the LatticeECP3 family offers a 10-Gigabit Ethernet Physical/MAC layer solution that is fully interoperable with the Broadcom BCM56800 network switch.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
July 2010	01.0	Initial release.
February 2012	01.1	Updated document with new corporate logo.

Appendix A. LatticeECP3 PCS Auto-Configuration File

```
# This file is used by the simulation model as well as the ispLEVER bitstream
# generation process to automatically initialize the PCSD quad to the mode
# selected in the IPexpress. This file is expected to be modified by the
# end user to adjust the PCSD quad to the final design requirements.

DEVICE_NAME "LFE3-95E"
CH0_PROTOCOL              "XAUI"
CH1_PROTOCOL              "XAUI"
CH2_PROTOCOL              "XAUI"
CH3_PROTOCOL              "XAUI"
CH0_MODE                  "RXTX"
CH1_MODE                  "RXTX"
CH2_MODE                  "RXTX"
CH3_MODE                  "RXTX"
CH0_CDR_SRC         "REFCLK_EXT"
CH1_CDR_SRC         "REFCLK_EXT"
CH2_CDR_SRC         "REFCLK_EXT"
CH3_CDR_SRC         "REFCLK_EXT"
PLL_SRC                   "REFCLK_EXT"
TX_DATARATE_RANGE         "HIGH"
CH0_RX_DATARATE_RANGE     "HIGH"
CH1_RX_DATARATE_RANGE     "HIGH"
CH2_RX_DATARATE_RANGE     "HIGH"
CH3_RX_DATARATE_RANGE     "HIGH"
REFCK_MULT                "20X"
#REFCLK_RATE              156.25
CH0_RX_DATA_RATE          "FULL"
CH1_RX_DATA_RATE          "FULL"
CH2_RX_DATA_RATE          "FULL"
CH3_RX_DATA_RATE          "FULL"
CH0_TX_DATA_RATE          "FULL"
CH1_TX_DATA_RATE          "FULL"
CH2_TX_DATA_RATE          "FULL"
CH3_TX_DATA_RATE          "FULL"
CH0_TX_DATA_WIDTH         "16"
CH1_TX_DATA_WIDTH         "16"
CH2_TX_DATA_WIDTH         "16"
CH3_TX_DATA_WIDTH         "16"
CH0_RX_DATA_WIDTH          "16"
CH1_RX_DATA_WIDTH          "16"
CH2_RX_DATA_WIDTH          "16"
CH3_RX_DATA_WIDTH          "16"
CH0_TX_FIFO         "ENABLED"
CH1_TX_FIFO         "ENABLED"
CH2_TX_FIFO         "ENABLED"
CH3_TX_FIFO         "ENABLED"
CH0_RX_FIFO          "ENABLED"
CH1_RX_FIFO          "ENABLED"
CH2_RX_FIFO          "ENABLED"
CH3_RX_FIFO          "ENABLED"
CH0_TDRV      "4"
CH1_TDRV      "4"
```

```
CH2_TDRV        "4"
CH3_TDRV        "4"
#CH0_TX_FICLK_RATE      156.25
#CH1_TX_FICLK_RATE      156.25
#CH2_TX_FICLK_RATE      156.25
#CH3_TX_FICLK_RATE      156.25
#CH0_RXREFCLK_RATE        "156.25"
#CH1_RXREFCLK_RATE        "156.25"
#CH2_RXREFCLK_RATE        "156.25"
#CH3_RXREFCLK_RATE        "156.25"
#CH0_RX_FICLK_RATE      156.25
#CH1_RX_FICLK_RATE      156.25
#CH2_RX_FICLK_RATE      156.25
#CH3_RX_FICLK_RATE      156.25
CH0_TX_PRE              "DISABLED"
CH1_TX_PRE              "DISABLED"
CH2_TX_PRE              "DISABLED"
CH3_TX_PRE              "DISABLED"
CH0_RTERM_TX            "50"
CH1_RTERM_TX            "50"
CH2_RTERM_TX            "50"
CH3_RTERM_TX            "50"
CH0_RX_EQ               "DISABLED"
CH1_RX_EQ               "DISABLED"
CH2_RX_EQ               "DISABLED"
CH3_RX_EQ               "DISABLED"
CH0_RTERM_RX            "50"
CH1_RTERM_RX            "50"
CH2_RTERM_RX            "50"
CH3_RTERM_RX            "50"
CH0_RX_DCC              "AC"
CH1_RX_DCC              "AC"
CH2_RX_DCC              "AC"
CH3_RX_DCC              "AC"
CH0_LOS_THRESHOLD       "2"
CH1_LOS_THRESHOLD       "2"
CH2_LOS_THRESHOLD       "2"
CH3_LOS_THRESHOLD       "2"
PLL_TERM                "50"
PLL_DCC                 "AC"
PLL_LOL_SET             "0"
CH0_TX_SB               "DISABLED"
CH1_TX_SB               "DISABLED"
CH2_TX_SB               "DISABLED"
CH3_TX_SB               "DISABLED"
CH0_RX_SB               "DISABLED"
CH1_RX_SB               "DISABLED"
CH2_RX_SB               "DISABLED"
CH3_RX_SB               "DISABLED"
CH0_TX_8B10B            "ENABLED"
CH1_TX_8B10B            "ENABLED"
CH2_TX_8B10B            "ENABLED"
CH3_TX_8B10B            "ENABLED"
CH0_RX_8B10B            "ENABLED"
```

```
CH1_RX_8B10B            "ENABLED"
CH2_RX_8B10B            "ENABLED"
CH3_RX_8B10B            "ENABLED"
CH0_COMMA_A             "1100000101"
CH1_COMMA_A             "1100000101"
CH2_COMMA_A             "1100000101"
CH3_COMMA_A             "1100000101"
CH0_COMMA_B             "0011111010"
CH1_COMMA_B             "0011111010"
CH2_COMMA_B             "0011111010"
CH3_COMMA_B             "0011111010"
CH0_COMMA_M             "1111111100"
CH1_COMMA_M             "1111111100"
CH2_COMMA_M             "1111111100"
CH3_COMMA_M             "1111111100"
CH0_RXWA                "ENABLED"
CH1_RXWA                "ENABLED"
CH2_RXWA                "ENABLED"
CH3_RXWA                "ENABLED"
CH0_ILSM                "ENABLED"
CH1_ILSM                "ENABLED"
CH2_ILSM                "ENABLED"
CH3_ILSM                "ENABLED"
CH0_CTC                 "DISABLED"
CH1_CTC                 "DISABLED"
CH2_CTC                 "DISABLED"
CH3_CTC                 "DISABLED"
CH0_CC_MATCH4           "0100011100"
CH1_CC_MATCH4           "0100011100"
CH2_CC_MATCH4           "0100011100"
CH3_CC_MATCH4           "0100011100"
CH0_CC_MATCH_MODE       "1"
CH1_CC_MATCH_MODE       "1"
CH2_CC_MATCH_MODE       "1"
CH3_CC_MATCH_MODE       "1"
CH0_CC_MIN_IPG          "0"
CH1_CC_MIN_IPG          "0"
CH2_CC_MIN_IPG          "0"
CH3_CC_MIN_IPG          "0"
CCHMARK                 "9"
CCLMARK                 "7"
CH0_SSLB                "DISABLED"
CH1_SSLB                "DISABLED"
CH2_SSLB                "DISABLED"
CH3_SSLB                "DISABLED"
CH0_SPLBPORTS           "DISABLED"
CH1_SPLBPORTS           "DISABLED"
CH2_SPLBPORTS           "DISABLED"
CH3_SPLBPORTS           "DISABLED"
CH0_PCSLBPORTS          "DISABLED"
CH1_PCSLBPORTS          "DISABLED"
CH2_PCSLBPORTS          "DISABLED"
CH3_PCSLBPORTS          "DISABLED"
INT_ALL                 "ENABLED"
QD_REFCK2CORE           "ENABLED"
```

Appendix B. unicast_settings.fpm ORCAstra Macro

```
'load 03 '  set the TX Config reg,
load 03 ' disable CRC insertion
write 00a02
'load 1C '  set the RX Config reg
load 5E '  set the RX Config reg, drop control, do not remove CRC
write 00a03
'load 01 '  set 12byte  IPG value
'write 00a03
load 00' write the MAC address
write 00a0C
load 00
write 00a0B
load 00
write 00a0A
load 00
write 00a09
load 00
write 00a08
load 02
write 00a07
load 02 ' do not swap DA,SA in loopback
write 00b00
load 03 ' Write the MODE reg
write 00a01
```

Appendix C. reg_stats_10ge.vbs ORCAstra Script

```
Sub xPut(ByRef Container, Val, N)
    Dim Sz: Sz = 15
    If (Len(Container) < (N * Sz)) Then _
        Container = Container & String((N * Sz) - Len(Container), "0")
    Dim Cprev: Cprev = Mid(Container, 1, (N * Sz))
    Dim Cfoll: Cfoll = Mid(Container, (((N + 1) * Sz) + 1))
    Container = Cprev & String(Sz - Len(CStr(Val)), "0") & CStr(Val) & Cfoll
End Sub

Function xGet(ByRef Container, N)
    Dim Sz: Sz = 15
'    msgbox "Container1 = """ & Container & """" & vbCrLf & _
'           "N = " & N & vbCrLf & _
'           """" & Mid(Container, ((N * Sz) + 1), Sz) & """"
    If (Len(Container) < ((N +1) * Sz)) Then _
        Container = Container & String((((N + 1) * Sz) - Len(Container)), "0")
'    msgbox "Container2 = """ & Container & """" & vbCrLf & _
'           "N = " & N & vbCrLf & _
'           """" & Mid(Container, ((N * Sz) + 1), Sz) & """"
    xGet = CCur(Mid(Container, ((N * Sz) + 1), Sz))
End Function



Sub Main()

'  do
dim TagVar: TagVar = V.Tag
V.Show_Display()
V.Clear_Display()
V.Echo("Statistics Reading")
V.Echo("")

' RX Packet Length
'temp0 = V.SGet(&h900)
'temp1 = V.SGet(&h901)
'temp2 = V.SGet(&h902)
'temp3 = V.SGet(&h903)
'temp4 = V.SGet(&h904)
'temp5 = V.SGet(&h905)
'temp6 = V.SGet(&h906)
'temp7 = V.SGet(&h907)
'temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
'out = temp - xGet(TagVar,0)
'xPut TagVar, temp, 0
'V.Echo("RX Packet Good (except  runt/long ) Length = " & out  &"")

' RX VLAN
temp0 = V.SGet(&h908)
temp1 = V.SGet(&h909)
temp2 = V.SGet(&h90a)
temp3 = V.SGet(&h90b)
temp4 = V.SGet(&h90c)
temp5 = V.SGet(&h90d)
temp6 = V.SGet(&h90e)
temp7 = V.SGet(&h90f)
```

```
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,1)
xPut TagVar, temp, 1
V.Echo("RX VLAN PKT = " & out  &"")

' RX PAUSE
temp0 = V.SGet(&h910)
temp1 = V.SGet(&h911)
temp2 = V.SGet(&h912)
temp3 = V.SGet(&h913)
temp4 = V.SGet(&h914)
temp5 = V.SGet(&h915)
temp6 = V.SGet(&h916)
temp7 = V.SGet(&h917)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,2)
xPut TagVar, temp, 2
V.Echo("RX PAUSE PKT = " & out  &"")


' RX Filtered (CONTROL,BC,MC)
temp0 = V.SGet(&h918)
temp1 = V.SGet(&h919)
temp2 = V.SGet(&h91a)
temp3 = V.SGet(&h91b)
temp4 = V.SGet(&h91c)
temp5 = V.SGet(&h91d)
temp6 = V.SGet(&h91e)
temp7 = V.SGet(&h91f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,3)
xPut TagVar, temp, 3
V.Echo("RX FILTERED(CONTROL,BC, MC) PKTS = " & out  &"")

' RX UNSUPPORTED
temp0 = V.SGet(&h920)
temp1 = V.SGet(&h921)
temp2 = V.SGet(&h922)
temp3 = V.SGet(&h923)
temp4 = V.SGet(&h924)
temp5 = V.SGet(&h925)
temp6 = V.SGet(&h926)
temp7 = V.SGet(&h927)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,4)
xPut TagVar, temp, 4
V.Echo("RX UNSUPPORTED CODE PKT = " & out  &"")


' RX BROADCAST
temp0 = V.SGet(&h928)
temp1 = V.SGet(&h929)
temp2 = V.SGet(&h92a)
temp3 = V.SGet(&h92b)
temp4 = V.SGet(&h92c)
```

```
temp5 = V.SGet(&h92d)
temp6 = V.SGet(&h92e)
temp7 = V.SGet(&h92f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,5)
xPut TagVar, temp, 5
V.Echo("RX BROADCAST PKT = " & out  &"")


' RX MULTICAST
temp0 = V.SGet(&h930)
temp1 = V.SGet(&h931)
temp2 = V.SGet(&h932)
temp3 = V.SGet(&h933)
temp4 = V.SGet(&h934)
temp5 = V.SGet(&h935)
temp6 = V.SGet(&h936)
temp7 = V.SGet(&h937)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,6)
xPut TagVar, temp, 6
V.Echo("RX MULTICAST PKT = " & out  &"")


' RX LENGTH ERROR
temp0 = V.SGet(&h938)
temp1 = V.SGet(&h939)
temp2 = V.SGet(&h93a)
temp3 = V.SGet(&h93b)
temp4 = V.SGet(&h93c)
temp5 = V.SGet(&h93d)
temp6 = V.SGet(&h93e)
temp7 = V.SGet(&h93f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,7)
xPut TagVar, temp, 7
V.Echo("RX LENGTH ERROR OR Short PKT = " & out  &"")



' RX SFD NOT FOUND (OLD)
' RX LONG PACKETS

temp0 = V.SGet(&h940)
temp1 = V.SGet(&h941)
temp2 = V.SGet(&h942)
temp3 = V.SGet(&h943)
temp4 = V.SGet(&h944)
temp5 = V.SGet(&h945)
temp6 = V.SGet(&h946)
temp7 = V.SGet(&h947)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,8)
```

```
xPut TagVar, temp, 8
'V.Echo("RX SFD NOT FOUND PKT = " & out  &"")
V.Echo("RX LONG  PKT = " & out  &"")


' RX PREAMBLE SHRINK (OLD)
' RX CRC ERROR
temp0 = V.SGet(&h948)
temp1 = V.SGet(&h949)
temp2 = V.SGet(&h94a)
temp3 = V.SGet(&h94b)
temp4 = V.SGet(&h94c)
temp5 = V.SGet(&h94d)
temp6 = V.SGet(&h94e)
temp7 = V.SGet(&h94f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,9)
xPut TagVar, temp, 9
'V.Echo("RX PREAMBLE SHRINK PKT = " & out  &"")
V.Echo("RX CRC ERROR = " & out  &"")



' RX IPG VIOLATION
' RX PKT DISCARDED
temp0 = V.SGet(&h950)
temp1 = V.SGet(&h951)
temp2 = V.SGet(&h952)
temp3 = V.SGet(&h953)
temp4 = V.SGet(&h954)
temp5 = V.SGet(&h955)
temp6 = V.SGet(&h956)
temp7 = V.SGet(&h957)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,10)
xPut TagVar, temp, 10
'V.Echo("RX IPG VIOLATION PKT = " & out  &"")
V.Echo("RX PKT DISCARDED (UNIC,CONTROL,MC,BC,Short)= " & out  &"")


' RX LONG                                  94F94
' RX PKT IGNORED
temp0 = V.SGet(&h958)
temp1 = V.SGet(&h959)
temp2 = V.SGet(&h95a)
temp3 = V.SGet(&h95b)
temp4 = V.SGet(&h95c)
temp5 = V.SGet(&h95d)
temp6 = V.SGet(&h95e)
temp7 = V.SGet(&h95f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,11)
xPut TagVar, temp, 11
'V.Echo("RX LONG PACKET = " & out  &"")
V.Echo("RX PKT IGNORED = " & out  &"")
```

```
' RX CRC ERROR
' RX FRAGMENT PKT
temp0 = V.SGet(&h960)
temp1 = V.SGet(&h961)
temp2 = V.SGet(&h962)
temp3 = V.SGet(&h963)
temp4 = V.SGet(&h964)
temp5 = V.SGet(&h965)
temp6 = V.SGet(&h966)
temp7 = V.SGet(&h967)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
'MsgBox hexx(temp7, 2)  &  hexx(temp6, 2)  & hexx(temp5, 2)  & hexx(temp4, 2)  & hexx(temp3,
2)  & hexx(temp2, 2)  & hexx(temp1, 2)  & hexx(temp0, 2)
out = temp - xGet(TagVar,12)
xPut TagVar, temp, 12
'V.Echo("RX CRC ERROR PKT = " & out  &"")
V.Echo("RX FRAGMENT PKT = " & out  &"")


' RX DROPPED PKTS
' RX JABBER  PKTS
temp0 = V.SGet(&h968)
temp1 = V.SGet(&h969)
temp2 = V.SGet(&h96a)
temp3 = V.SGet(&h96b)
temp4 = V.SGet(&h96c)
temp5 = V.SGet(&h96d)
temp6 = V.SGet(&h96e)
temp7 = V.SGet(&h96f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,13)
xPut TagVar, temp, 13
'V.Echo("RX DROPPED (UNIC,CONTROL,MC,BC,Short) PKTS = " & out  &"")
V.Echo("RX JABBER  PKTS = " & out  &"")


' RX IGNORED
' RX RECEIVED 64 BYTE PKT
temp0 = V.SGet(&h970)
temp1 = V.SGet(&h971)
temp2 = V.SGet(&h972)
temp3 = V.SGet(&h973)
temp4 = V.SGet(&h974)
temp5 = V.SGet(&h975)
temp6 = V.SGet(&h976)
temp7 = V.SGet(&h977)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,14)
xPut TagVar, temp, 14
'V.Echo("RX IGNORED PKT = " & out  &"")
```

```
V.Echo("RX = 64 BYTE PKT = " & out  &"")

'NEW

' RX RECEIVED 65-127 BYTE PKT
temp0 = V.SGet(&h978)
temp1 = V.SGet(&h979)
temp2 = V.SGet(&h97A)
temp3 = V.SGet(&h97B)
temp4 = V.SGet(&h97C)
temp5 = V.SGet(&h97D)
temp6 = V.SGet(&h97E)
temp7 = V.SGet(&h97F)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,15)
xPut TagVar, temp, 15
V.Echo("RX  65-127 BYTE PKT = " & out  &"")


' RX RECEIVED 128-255 BYTE PKT
temp0 = V.SGet(&h980)
temp1 = V.SGet(&h981)
temp2 = V.SGet(&h982)
temp3 = V.SGet(&h983)
temp4 = V.SGet(&h984)
temp5 = V.SGet(&h985)
temp6 = V.SGet(&h986)
temp7 = V.SGet(&h987)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,16)
xPut TagVar, temp, 16
V.Echo("RX  128-255 BYTE PKT = " & out  &"")

' RX RECEIVED 256-511 BYTE PKT
temp0 = V.SGet(&h988)
temp1 = V.SGet(&h989)
temp2 = V.SGet(&h98A)
temp3 = V.SGet(&h98B)
temp4 = V.SGet(&h98C)
temp5 = V.SGet(&h98D)
temp6 = V.SGet(&h98E)
temp7 = V.SGet(&h98F)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,17)
xPut TagVar, temp, 17
V.Echo("RX  256-511 BYTE PKT = " & out  &"")


' RX RECEIVED 512-1023 BYTE PKT
temp0 = V.SGet(&h990)
temp1 = V.SGet(&h991)
temp2 = V.SGet(&h992)
temp3 = V.SGet(&h993)
temp4 = V.SGet(&h994)
temp5 = V.SGet(&h995)
temp6 = V.SGet(&h996)
```

```
temp7 = V.SGet(&h997)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,18)
xPut TagVar, temp, 18
V.Echo("RX  512-1023 BYTE PKT = " & out  &"")


' RX RECEIVED 1024-1518 BYTE PKT
temp0 = V.SGet(&h998)
temp1 = V.SGet(&h999)
temp2 = V.SGet(&h99A)
temp3 = V.SGet(&h99B)
temp4 = V.SGet(&h99C)
temp5 = V.SGet(&h99D)
temp6 = V.SGet(&h99E)
temp7 = V.SGet(&h99F)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,19)
xPut TagVar, temp, 19
V.Echo("RX  1024-1518 BYTE PKT = " & out  &"")



' RX UNDERSIZE PKT
temp0 = V.SGet(&h9A0)
temp1 = V.SGet(&h9A1)
temp2 = V.SGet(&h9A2)
temp3 = V.SGet(&h9A3)
temp4 = V.SGet(&h9A4)
temp5 = V.SGet(&h9A5)
temp6 = V.SGet(&h9A6)
temp7 = V.SGet(&h9A7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,20)
xPut TagVar, temp, 20
V.Echo("RX  UNDERSIZE PKT = " & out  &"")


' RX UNICAST PKT
temp0 = V.SGet(&h9A8)
temp1 = V.SGet(&h9A9)
temp2 = V.SGet(&h9AA)
temp3 = V.SGet(&h9AB)
temp4 = V.SGet(&h9AC)
temp5 = V.SGet(&h9AD)
temp6 = V.SGet(&h9AE)
temp7 = V.SGet(&h9AF)
'MsgBox hexx(temp7, 2)  &  hexx(temp6, 2)  & hexx(temp5, 2)  & hexx(temp4, 2)  & hexx(temp3,
2)  & hexx(temp2, 2)  & hexx(temp1, 2)  & hexx(temp0, 2)

temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,21)
xPut TagVar, temp, 21
V.Echo("RX  UNICAST PKT = " & out  &"")
```

```
' RX RECEIVED PKT
temp0 = V.SGet(&h9B0)
temp1 = V.SGet(&h9B1)
temp2 = V.SGet(&h9B2)
temp3 = V.SGet(&h9B3)
temp4 = V.SGet(&h9B4)
temp5 = V.SGet(&h9B5)
temp6 = V.SGet(&h9B6)
temp7 = V.SGet(&h9B7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,22)
xPut TagVar, temp, 22
V.Echo("RX  RECEIVED PKT = " & out  &"")


' RX 64-byte GOOD CRC PKT
temp0 = V.SGet(&h9B8)
temp1 = V.SGet(&h9B9)
temp2 = V.SGet(&h9BA)
temp3 = V.SGet(&h9BB)
temp4 = V.SGet(&h9BC)
temp5 = V.SGet(&h9BD)
temp6 = V.SGet(&h9BE)
temp7 = V.SGet(&h9BF)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,23)
xPut TagVar, temp, 23
V.Echo("RX < 64-byte GOOD CRC PKT = " & out  &"")



' RX 1518-byte GOOD CRC PKT
temp0 = V.SGet(&h9C0)
temp1 = V.SGet(&h9C1)
temp2 = V.SGet(&h9C2)
temp3 = V.SGet(&h9C3)
temp4 = V.SGet(&h9C4)
temp5 = V.SGet(&h9C5)
temp6 = V.SGet(&h9C6)
temp7 = V.SGet(&h9C7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,24)
xPut TagVar, temp, 24
V.Echo("RX  > 1518-byte GOOD CRC PKT = " & out  &"")


' RX 1519-2047 BYTE PKT
temp0 = V.SGet(&h9C8)
temp1 = V.SGet(&h9C9)
temp2 = V.SGet(&h9CA)
temp3 = V.SGet(&h9CB)
temp4 = V.SGet(&h9CC)
temp5 = V.SGet(&h9CD)
temp6 = V.SGet(&h9CE)
temp7 = V.SGet(&h9CF)
```

```
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,25)
xPut TagVar, temp, 25
V.Echo("RX  1519-2047 BYTE PKT = " & out  &"")



' RX 2048-4095 BYTE PKT
temp0 = V.SGet(&h9D0)
temp1 = V.SGet(&h9D1)
temp2 = V.SGet(&h9D2)
temp3 = V.SGet(&h9D3)
temp4 = V.SGet(&h9D4)
temp5 = V.SGet(&h9D5)
temp6 = V.SGet(&h9D6)
temp7 = V.SGet(&h9D7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,26)
xPut TagVar, temp, 26
V.Echo("RX  2048-4095 BYTE PKT = " & out  &"")


' RX 4096-9216 BYTE PKT
temp0 = V.SGet(&h9D8)
temp1 = V.SGet(&h9D9)
temp2 = V.SGet(&h9DA)
temp3 = V.SGet(&h9DB)
temp4 = V.SGet(&h9DC)
temp5 = V.SGet(&h9DD)
temp6 = V.SGet(&h9DE)
temp7 = V.SGet(&h9DF)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,27)
xPut TagVar, temp, 27
V.Echo("RX  4096-9216 BYTE PKT = " & out  &"")



' RX 9217-16383 BYTE PKT
temp0 = V.SGet(&h9E0)
temp1 = V.SGet(&h9E1)
temp2 = V.SGet(&h9E2)
temp3 = V.SGet(&h9E3)
temp4 = V.SGet(&h9E4)
temp5 = V.SGet(&h9E5)
temp6 = V.SGet(&h9E6)
temp7 = V.SGet(&h9E7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,28)
xPut TagVar, temp, 28
V.Echo("RX  9217-16383 BYTE PKT = " & out  &"")
```

```
V.Echo("")
V.Echo("")



' TX PKT LENGTH
'temp0 = V.SGet(&h800)
'temp1 = V.SGet(&h801)
'temp2 = V.SGet(&h802)
'temp3 = V.SGet(&h803)
'temp4 = V.SGet(&h804)
'temp5 = V.SGet(&h805)
'temp6 = V.SGet(&h806)
'temp7 = V.SGet(&h807)
'temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
'out = temp - xGet(TagVar,29)
'xPut TagVar, temp, 29
'V.Echo("TX PKT LENGTH  = " & out  &"")


' TX TERMINATE ERR
temp0 = V.SGet(&h808)
temp1 = V.SGet(&h809)
temp2 = V.SGet(&h80a)
temp3 = V.SGet(&h80b)
temp4 = V.SGet(&h80c)
temp5 = V.SGet(&h80d)
temp6 = V.SGet(&h80e)
temp7 = V.SGet(&h80f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,30)
xPut TagVar, temp, 30
V.Echo("TX TERMINATE ERR PKT = " & out  &"")




' TX UNDERRRUN
temp0 = V.SGet(&h810)
temp1 = V.SGet(&h811)
temp2 = V.SGet(&h812)
temp3 = V.SGet(&h813)
temp4 = V.SGet(&h814)
temp5 = V.SGet(&h815)
temp6 = V.SGet(&h816)
temp7 = V.SGet(&h817)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,31)
xPut TagVar, temp, 31
V.Echo("TX UNDERRUN PKT = " & out  &"")


' TX CRC ERR
temp0 = V.SGet(&h818)
```

```
temp1 = V.SGet(&h819)
temp2 = V.SGet(&h81a)
temp3 = V.SGet(&h81b)
temp4 = V.SGet(&h81c)
temp5 = V.SGet(&h81d)
temp6 = V.SGet(&h81e)
temp7 = V.SGet(&h81f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,32)
xPut TagVar, temp, 32
V.Echo("TX CRC ERR PKT = " & out  &"")


' TX LENGTH ERROR
temp0 = V.SGet(&h820)
temp1 = V.SGet(&h821)
temp2 = V.SGet(&h822)
temp3 = V.SGet(&h823)
temp4 = V.SGet(&h824)
temp5 = V.SGet(&h825)
temp6 = V.SGet(&h826)
temp7 = V.SGet(&h827)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,33)
xPut TagVar, temp, 33
V.Echo("TX LENGTH ERROR PKT = " & out  &"")


' TX LONG
temp0 = V.SGet(&h828)
temp1 = V.SGet(&h829)
temp2 = V.SGet(&h82a)
temp3 = V.SGet(&h82b)
temp4 = V.SGet(&h82c)
temp5 = V.SGet(&h82d)
temp6 = V.SGet(&h82e)
temp7 = V.SGet(&h82f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,34)
xPut TagVar, temp, 34
V.Echo("TX LONG PKT = " & out  &"")


' TX MULTICAST
temp0 = V.SGet(&h830)
temp1 = V.SGet(&h831)
temp2 = V.SGet(&h832)
temp3 = V.SGet(&h833)
temp4 = V.SGet(&h834)
temp5 = V.SGet(&h835)
temp6 = V.SGet(&h836)
temp7 = V.SGet(&h837)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,35)
```

```
xPut TagVar, temp, 35
V.Echo("TX MULTICAST  PKT = " & out  &"")


' TX BROADCAST
temp0 = V.SGet(&h838)
temp1 = V.SGet(&h839)
temp2 = V.SGet(&h83a)
temp3 = V.SGet(&h83b)
temp4 = V.SGet(&h83c)
temp5 = V.SGet(&h83d)
temp6 = V.SGet(&h83e)
temp7 = V.SGet(&h83f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,36)
xPut TagVar, temp, 36
V.Echo("TX BROADCAST PKT = " & out  &"")



' TX Control
temp0 = V.SGet(&h840)
temp1 = V.SGet(&h841)
temp2 = V.SGet(&h842)
temp3 = V.SGet(&h843)
temp4 = V.SGet(&h844)
temp5 = V.SGet(&h845)
temp6 = V.SGet(&h846)
temp7 = V.SGet(&h847)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,37)
xPut TagVar, temp, 37
V.Echo("TX Control  PKT = " & out  &"")


' TX JUMBO
temp0 = V.SGet(&h848)
temp1 = V.SGet(&h849)
temp2 = V.SGet(&h84a)
temp3 = V.SGet(&h84b)
temp4 = V.SGet(&h84c)
temp5 = V.SGet(&h84d)
temp6 = V.SGet(&h84e)
temp7 = V.SGet(&h84f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,38)
xPut TagVar, temp, 38
V.Echo("TX JUMBO PKT = " & out  &"")



' TX PAUSE
temp0 = V.SGet(&h850)
temp1 = V.SGet(&h851)
temp2 = V.SGet(&h852)
```

```
temp3 = V.SGet(&h853)
temp4 = V.SGet(&h854)
temp5 = V.SGet(&h855)
temp6 = V.SGet(&h856)
temp7 = V.SGet(&h857)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,39)
xPut TagVar, temp, 39
V.Echo("TX PAUSE  PKT = " & out  &"")


' TX VLAN
temp0 = V.SGet(&h858)
temp1 = V.SGet(&h859)
temp2 = V.SGet(&h85a)
temp3 = V.SGet(&h85b)
temp4 = V.SGet(&h85c)
temp5 = V.SGet(&h85d)
temp6 = V.SGet(&h85e)
temp7 = V.SGet(&h85f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,40)
xPut TagVar, temp, 40
V.Echo("TX VLAN TAGGED PKT = " & out  &"")



' TX OK
temp0 = V.SGet(&h860)
temp1 = V.SGet(&h861)
temp2 = V.SGet(&h862)
temp3 = V.SGet(&h863)
temp4 = V.SGet(&h864)
temp5 = V.SGet(&h865)
temp6 = V.SGet(&h866)
temp7 = V.SGet(&h867)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,41)
xPut TagVar, temp, 41
V.Echo("TX OK  PKT = " & out  &"")



' TX 64-byte PKT
temp0 = V.SGet(&h868)
temp1 = V.SGet(&h869)
temp2 = V.SGet(&h86a)
temp3 = V.SGet(&h86b)
temp4 = V.SGet(&h86c)
temp5 = V.SGet(&h86d)
temp6 = V.SGet(&h86e)
temp7 = V.SGet(&h86f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,42)
```

```
xPut TagVar, temp, 42
V.Echo("TX = 64-byte  PKT = " & out  &"")


' TX 65-127 byte
temp0 = V.SGet(&h870)
temp1 = V.SGet(&h871)
temp2 = V.SGet(&h872)
temp3 = V.SGet(&h873)
temp4 = V.SGet(&h874)
temp5 = V.SGet(&h875)
temp6 = V.SGet(&h876)
temp7 = V.SGet(&h877)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,43)
xPut TagVar, temp, 43
V.Echo("TX 65-127 byte  PKT = " & out  &"")



' TX 128-255 byte PKT
temp0 = V.SGet(&h878)
temp1 = V.SGet(&h879)
temp2 = V.SGet(&h87a)
temp3 = V.SGet(&h87b)
temp4 = V.SGet(&h87c)
temp5 = V.SGet(&h87d)
temp6 = V.SGet(&h87e)
temp7 = V.SGet(&h87f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,44)
xPut TagVar, temp, 44
V.Echo("TX 128-255 byte PKT = " & out  &"")


' TX 256-511 byte
temp0 = V.SGet(&h880)
temp1 = V.SGet(&h881)
temp2 = V.SGet(&h882)
temp3 = V.SGet(&h883)
temp4 = V.SGet(&h884)
temp5 = V.SGet(&h885)
temp6 = V.SGet(&h886)
temp7 = V.SGet(&h887)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,45)
xPut TagVar, temp, 45
V.Echo("TX 256-511 byte  PKT = " & out  &"")



' TX 512-1023 byte PKT
temp0 = V.SGet(&h888)
temp1 = V.SGet(&h889)
temp2 = V.SGet(&h88a)
```

```
temp3 = V.SGet(&h88b)
temp4 = V.SGet(&h88c)
temp5 = V.SGet(&h88d)
temp6 = V.SGet(&h88e)
temp7 = V.SGet(&h88f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,46)
xPut TagVar, temp, 46
V.Echo("TX 512-1023 byte PKT = " & out  &"")


' TX 1024-1518 byte
temp0 = V.SGet(&h890)
temp1 = V.SGet(&h891)
temp2 = V.SGet(&h892)
temp3 = V.SGet(&h893)
temp4 = V.SGet(&h894)
temp5 = V.SGet(&h895)
temp6 = V.SGet(&h896)
temp7 = V.SGet(&h897)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,47)
xPut TagVar, temp, 47
V.Echo("TX 1024-1518 byte  PKT = " & out  &"")



' TX 1518-byte PKT
temp0 = V.SGet(&h898)
temp1 = V.SGet(&h899)
temp2 = V.SGet(&h89a)
temp3 = V.SGet(&h89b)
temp4 = V.SGet(&h89c)
temp5 = V.SGet(&h89d)
temp6 = V.SGet(&h89e)
temp7 = V.SGet(&h89f)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,48)
xPut TagVar, temp, 48
V.Echo("TX > 1518-byte  PKT = " & out  &"")

' TX FRAME ERROR
temp0 = V.SGet(&h8A0)
temp1 = V.SGet(&h8A1)
temp2 = V.SGet(&h8A2)
temp3 = V.SGet(&h8A3)
temp4 = V.SGet(&h8A4)
temp5 = V.SGet(&h8A5)
temp6 = V.SGet(&h8A6)
temp7 = V.SGet(&h8A7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,49)
xPut TagVar, temp, 49
V.Echo("TX FRAME ERROR PKT = " & out  &"")
```

```
' TX 1519-2047 byte PKT
temp0 = V.SGet(&h8A8)
temp1 = V.SGet(&h8A9)
temp2 = V.SGet(&h8Aa)
temp3 = V.SGet(&h8Ab)
temp4 = V.SGet(&h8Ac)
temp5 = V.SGet(&h8Ad)
temp6 = V.SGet(&h8Ae)
temp7 = V.SGet(&h8Af)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,50)
xPut TagVar, temp, 50
V.Echo("TX 1519-2047 byte PKT = " & out  &"")


' TX 2048-4095 byte
temp0 = V.SGet(&h8B0)
temp1 = V.SGet(&h8B1)
temp2 = V.SGet(&h8B2)
temp3 = V.SGet(&h8B3)
temp4 = V.SGet(&h8B4)
temp5 = V.SGet(&h8B5)
temp6 = V.SGet(&h8B6)
temp7 = V.SGet(&h8B7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,51)
xPut TagVar, temp, 51
V.Echo("TX 2048-4095 byte  PKT = " & out  &"")



' TX 4096-9216 byte PKT
temp0 = V.SGet(&h8B8)
temp1 = V.SGet(&h8B9)
temp2 = V.SGet(&h8Ba)
temp3 = V.SGet(&h8Bb)
temp4 = V.SGet(&h8Bc)
temp5 = V.SGet(&h8Bd)
temp6 = V.SGet(&h8Be)
temp7 = V.SGet(&h8Bf)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,52)
xPut TagVar, temp, 52
V.Echo("TX 4096-9216 byte PKT = " & out  &"")


' TX 9217-16383 byte
temp0 = V.SGet(&h8C0)
temp1 = V.SGet(&h8C1)
temp2 = V.SGet(&h8C2)
temp3 = V.SGet(&h8C3)
temp4 = V.SGet(&h8C4)
temp5 = V.SGet(&h8C5)
temp6 = V.SGet(&h8C6)
```

```
temp7 = V.SGet(&h8C7)
temp = temp0 + temp1*2^8 + temp2*2^16 + temp3*2^24 + temp4*2^32 + temp5*2^40 + temp6*2^48 +
temp7*2^54
out = temp - xGet(TagVar,53)
xPut TagVar, temp, 53
V.Echo("TX  9217-16383 byte  PKT = " & out  &"")


V.Tag = TagVar


End Sub



Function HexX(Num, Lngth)
    TempStr = Hex(Num)
    HexX = Mid((String(8 - Len(TempStr), "0") & TempStr), (9 - Lngth))
End Function
```