LatticeECP3 Marvell SGMII Physical/MAC Layer Interoperability

February 2012 Technical Note TN1197

Introduction

This technical note describes an SGMII physical/MAC layer interoperability test between a LatticeECP3™ device and the Marvell 88E1111 PHY.

Specifically, the document discusses the following topics:

- Overview of LatticeECP3 devices and Marvell 88E1111 PHY
- SGMII physical/MAC layer interoperability setup and results

LatticeECP3 Overview

The LatticeECP3 (EConomy Plus Third generation) family of FPGA devices is optimized to deliver high-performance features such as an enhanced DSP architecture, high-speed SERDES and high-speed source synchronous interfaces in an economical FPGA fabric. This combination is achieved through advances in device architecture and the use of 65nm technology making the devices suitable for high-volume, high-speed, low-cost applications.

The LatticeECP3 device family also features high-speed SERDES with dedicated PCS functions. High jitter tolerance and low transmit jitter allow the SERDES plus PCS blocks to be configured to support an array of popular data protocols including PCI Express, SMPTE, Ethernet (XAUI, Gigabit Ethernet and SGMII) and CPRI. Transmit Pre-emphasis and Receive Equalization settings make the SERDES suitable for transmission and reception over various forms of media.

LatticeECP3 PCS

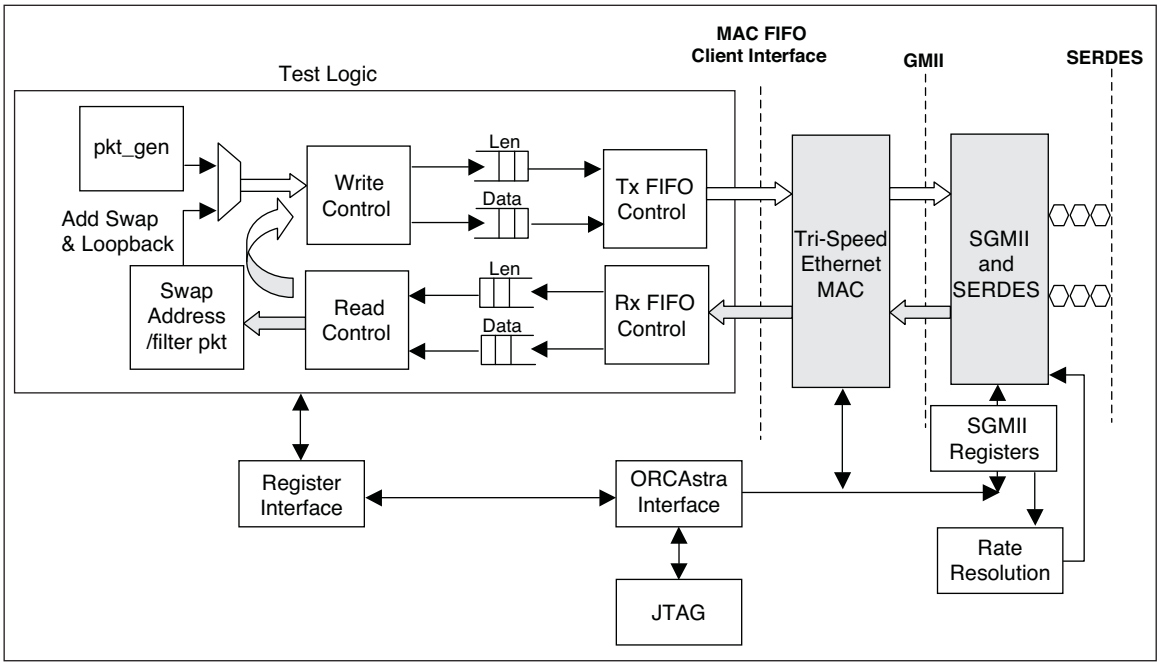
Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed, full-duplex serial data transfer at data rates up to 3.2 Gbps. The PCS logic in each channel can be configured to support an array of popular data protocols including Gigabit Ethernet, XAUI, PCI Express, Serial RapidIO, CPRI, OBSAI, SD-SDI, HD-SDI and 3G-SDI. In addition, the protocol-based logic can be fully or partially bypassed in a number of configurations to allow users flexibility in designing their own high-speed data interface.

The PCS also provides bypass modes that allow a direct 8-bit or 10-bit interface from the SERDES to the FPGA logic. Each SERDES pin can also be independently DC-coupled and can allow for both high-speed and low-speed operation on the same SERDES pin for applications such as Serial Digital Video.

LatticeECP3 SGMII PCS/Tri-Speed Ethernet MAC Reference Design

Figure 1 describes the Tri-Speed Ethernet MAC/SGMII reference design targeting the LatticeECP3 device. The reference design includes the SGMII and Gigabit Ethernet IP Core, the Tri-Speed Ethernet MAC IP Core, as well as test logic. ORCAstra logic controls and monitors the test logic, Tri-Speed Ethernet MAC and SGMII/Gb Ethernet PCS IP core registers. Register mapping information for all registers is described in Appendix A. Please refer to the Tri-Speed Ethernet MAC IP Core User's Guide and the SGMII and Gb Ethernet PCS IP Core User's Guide for more information on the specific IP registers.

Figure 1. LatticeECP3 Tri-Speed Ethernet MAC + SGMII and Gb Ethernet IP Reference Design



Lattice SGMII/Gb Ethernet PCS IP Core

The Lattice SGMII and Gb Ethernet PCS IP core implements the PCS functions of both the Cisco SGMII and the IEEE 802.3z (1000BASE-X) specifications. The PCS mode is pin-selectable. This IP core may be used in bridging applications and/or PHY implementations.

Features:

- Implements PCS functions of the Cisco SGMII Specification, Revision 1.7
- Implements PCS functions for IEEE 802.3z (1000BASE-X)
- Dynamically selects SGMII/1000BASE-X PCS operation
- Supports MAC or PHY mode for SGMII auto-negotiation
- Supports (G)MII data rates of 1Gbps, 100Mbps, 10Mbps
- Provides Management Interface Port for control and maintenance
- Includes Easy Connect option for seamless integration with the Lattice Tri-Speed Ethernet MAC IP core

Tri-Speed Ethernet MAC IP Core

The Tri-Speed Ethernet MAC transmits and receives data between a host processor and an Ethernet network. The main function of the Ethernet MAC is to ensure that the Media Access rules specified in the 802.3 IEEE standard are met while transmitting a frame of data over Ethernet.

Features:

- Compliant to IEEE 802.3z standard
- Generic 8-bit host interface
- 8-bit wide internal data path
- Generic transmit and receive FIFO interface
- Full-duplex operation in 1G mode

- Full- and half-duplex operation in 10/100 mode
- Transmit and receive statistics vector
- Programmable Inter-Packet Gap (IPG)
- Multicast address filtering
- Selectable MAC operating options
  - Classic Tri-Speed Ethernet MAC with G/MII
  - Gigabit MAC with GMII
  - SGMII Easy Connect MAC with GMII, configurable option available on LatticeECP3, LatticeECP2/M, and LatticeSC/M devices
- Supports:
  - Full-duplex control using PAUSE frames
  - VLAN tagged frames
  - Automatic re-transmission on collision
  - Automatic padding of short frames
  - Multicast and Broadcast frames
  - Optional FCS transmission and reception
  - Optional MII management interface module
  - Jumbo frames of any length

Marvell Alaska™ Ultra 88E1111 Overview

88E1111 Features

The Alaska Ultra 88E1111 Gigabit Ethernet Transceivers is a physical layer device for Ethernet 1000BASE-T, 100BASE-TX and 10BASE-T applications. It contain all the active circuitry required to implement the physical layer functions to transmit and receive data on standard CAT 3 and CAT 5 unshielded twisted pair.

The 88E1111 device supports the Gigabit Media Independent Interface (GMII), Reduced GMII (RGMII), Serial GMII (Gigabit Ethernet), the Ten-Bit Interface (TBI) and Reduced TBI (RTBI) for direct connection to a MAC/Switch port.

The 88E1111 device incorporates a 1.25 GHz SERDES, which may be directly connected to a fiber-optic transceiver for 1000BASE-T/SGMII media conversion applications. Additionally, the 88E1111 device may be used to implement 1000BASE-T Gigabit Interface Converter (GBIC) or Small Form Factor Pluggable (SFP) modules.

The Alaska Ultra 88E1111 features include:

- 10/100/100BASE-T IEEE 802.3 compliant
- Support for GMII, TBI, RGMII, RTBI and Gigabit Ethernet
- Integrated 1.25 GHz SERDES for SGMII fiber applications
- The 88E1112 device also supports 100BASE-FX
- Four RGMII timing modes
- Three energy detect modes and a low-power COMA mode
- Three loopback modes for diagnostics
- Fully integrated digital adaptive equalizers, echo cancellers and crosstalk cancellers
- Advanced digital baseline wander correction
- Automatic polarity correction
- IEEE 802.3u compliant auto-negotiation

- CRC checker, packet counter
- Automatic detection of fiber or copper media
- Virtual Cable Tester (VCT)
- Selectable MDC/MDIO interface or Two-Wire Serial Interface.

Test Equipment

Below is the equipment used in the interoperability test.

LatticeECP3 Serial Protocol Board (Version D)

Figure 2 shows the LatticeECP3 reference design and other components on the Serial Protocol Board. All board components are described in detail in EB44, LatticeECP3 Serial Protocol Board Revision D User's Guide.

An external SmartBits box auto-negotiates and transmits BASE-T frames to the Marvell 88E1111 PHY via the RJ-45 connector. The Marvell PHY converts this data to SGMII and sends it to the LatticeECP3 PCS (PCSC) channel 2 SERDES inputs. The data is transferred to the SGMII/Gb Ethernet PCS IP Core and the Tri-Speed Ethernet MAC IP Core. Test logic at the Tri-Speed Ethernet MAC client interface then loops data back in the other direction. The Tri-Speed Ethernet MAC keeps statistical information about Ethernet frames received and retransmitted. This information can be accessed via ORCAstra. ORCAstra can also control the 88E111 internal registers via an MDIO interface from the Tri-Speed Ethernet MAC to the 88E1111. 88E1111 Hardware Configuration options such as PHY address, HWCFG_MODE and auto-negotiation abilities are all accessed using this scheme. Refer to the Alaska Ultra 88E1111 Data Sheet for further information on programmable registers.

Table 1 provides a description of the LatticeECP3 Reference Design Signals accessible on the LatticeECP3 Serial Protocol Board.

Figure 2. LatticeECP3 Serial Protocol Board (Version D)

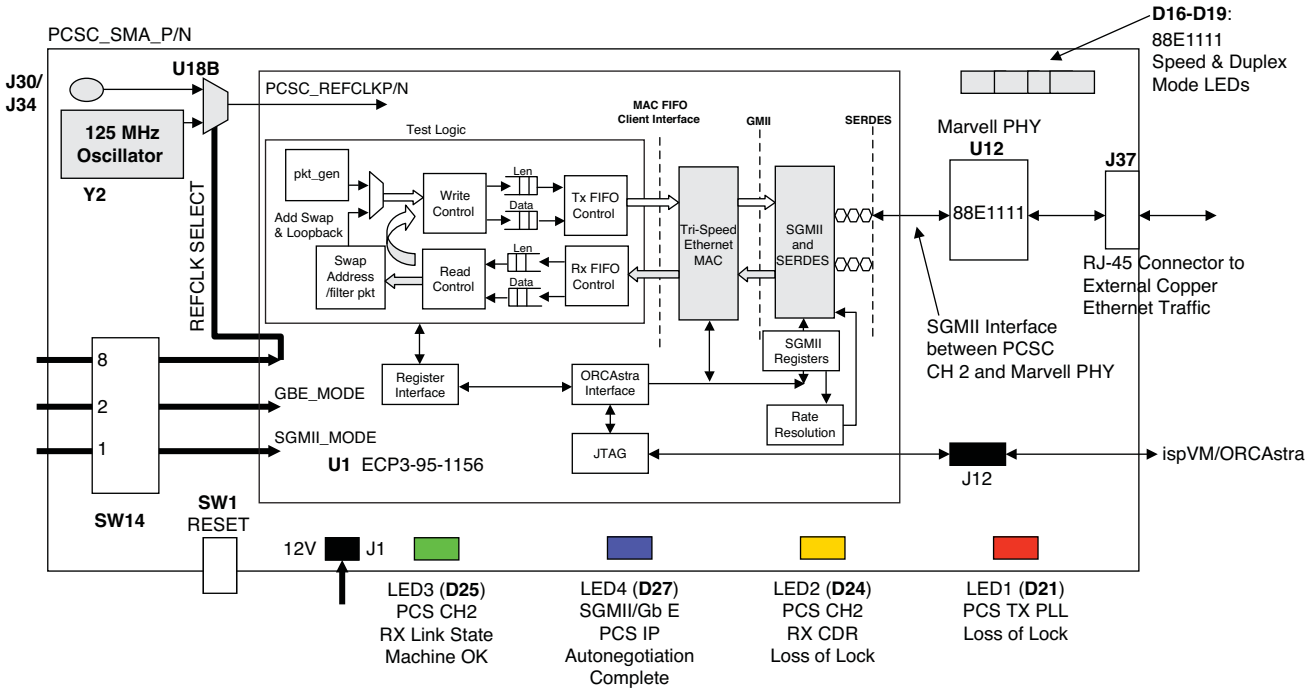


Table 1. LatticeECP3 Reference Design Signals

Signal Name	Signal Type	LatticeECP3 Serial Protocol Board (Version C or Newer) Connection	Description
General Signals			
reset_n	I	SW1 Pushbutton	FPGA Global active low reset
Reference Design Signals			
GBE_MODE	I	Output of SW14, switch 2	LatticeECP3 SGMII/Gb Ethernet IP GBE_MODE input. This signal is controlled by SW14, switch 2. Press switch 2 down (0V) to set GBE_MODE low. The SGMII/Gb Ethernet IP is running in SGMII mode. Pull switch 2 up (3.3V) to set GBE_MODE high. The SGMII/Gb Ethernet IP is running in SGMII mode.
SGMII_MODE	I	Output of SW14, switch 1	LatticeECP3 SGMII/Gb Ethernet IP SGMII_MODE input. This input is valid when GBE_MODE=0 (SGMII mode). This signal is controlled by SW14, switch 1. Press switch 1 down (0V) to set SGMII_MODE low. The SGMII/Gb Ethernet IP is in MAC mode. Pull switch 1 up (3.3V) to set SGMII_MODE high. The SGMII/Gb Ethernet IP is in PHY mode.
PCS TX PLL LOSS OF LOCK	O	LED1 (D21)	Red LED - LatticeECP3 Tx PLL loss of lock indication. The LED will not glow when a valid 125 MHz reference clock is provided to the LatticeECP3 PCS.
PCS CH2 RX CDR LOSS OF LOCK	O	LED2 (D24)	Yellow LED - LatticeECP3 channel 2 RX CDR loss of lock indication. The LED will not glow when valid 1.25 Gbps data is provided to the LatticeECP3 Channel 2 SERDES inputs.
SGMII/GBE PCS IP AUTONEG COMPLETE	O	LED4 (D27)	Blue LED - This LED will glow upon successful SGMII auto-negotiation with the Marvell PHY. Traffic will not flow through the system unless auto-negotiation completes.
PCS CH2 RX LINK STATE MACHINE OK	O	LED3 (D25)	Green LED - Indicates that the LatticeECP3 channel Rx Gb Ethernet link state machine is successfully synchronized to incoming Ethernet traffic. The LED will not glow when valid 1.25 Gbps Ethernet data is provided to the LatticeECP3 Channel 2 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		
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 SW14, switch 8. Press switch 8 down (0V) to select the Y2 125 MHz on-board oscillator. Pull switch 8 up (3.3V) to select the J30/J34 SMA clock inputs.

Spirent SmartBits 2000 Protocol Analyzer

Spirent Communications' SmartBits 2000 (SMB-2000) has become the industry standard for measuring the performance limits of everything from an emerging technology in a development lab to the largest enterprise network. The SMB- 2000 is widely used to test a variety of network devices and complex network configurations, including 10/100 Mbps Ethernet, Gigabit Ethernet, ATM, and Frame Relay.

Figure 3 illustrates the SMB-2000 analyzer with a 100/1000BASE-T copper module (GX-1420B, rightmost module with RJ45 cable attached).

Figure 4 illustrates the GUI command console for the SMB-2000. The GX-1420B module is also shown to the right of the screen.

Figure 5 illustrates the Transmit Setup Window for the GX-1420B module. This window controls the size and field format of transmitted Ethernet Frames.

Figure 3. SmartBits 2000 Protocol Analyzer with 100/1000BASE-T Module

Figure 4. SmartBits 2000 GUI Controls

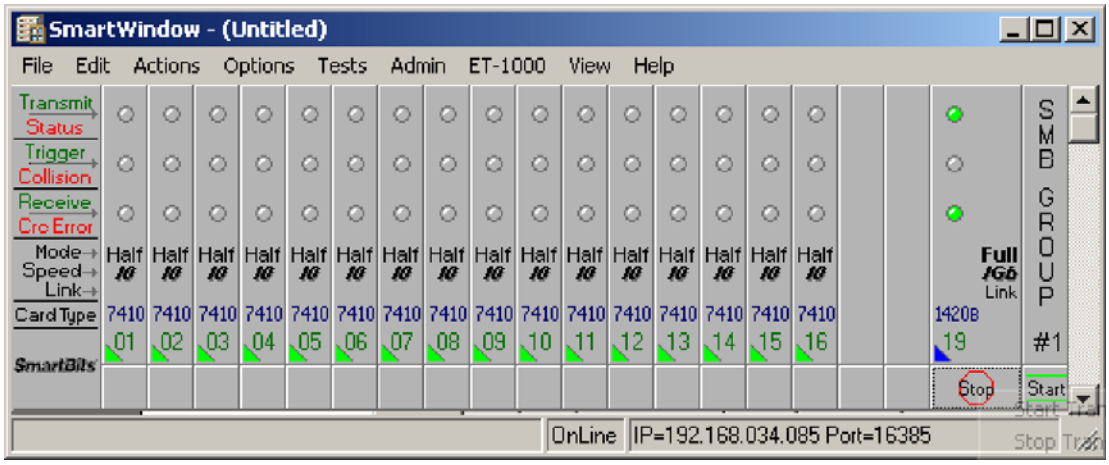
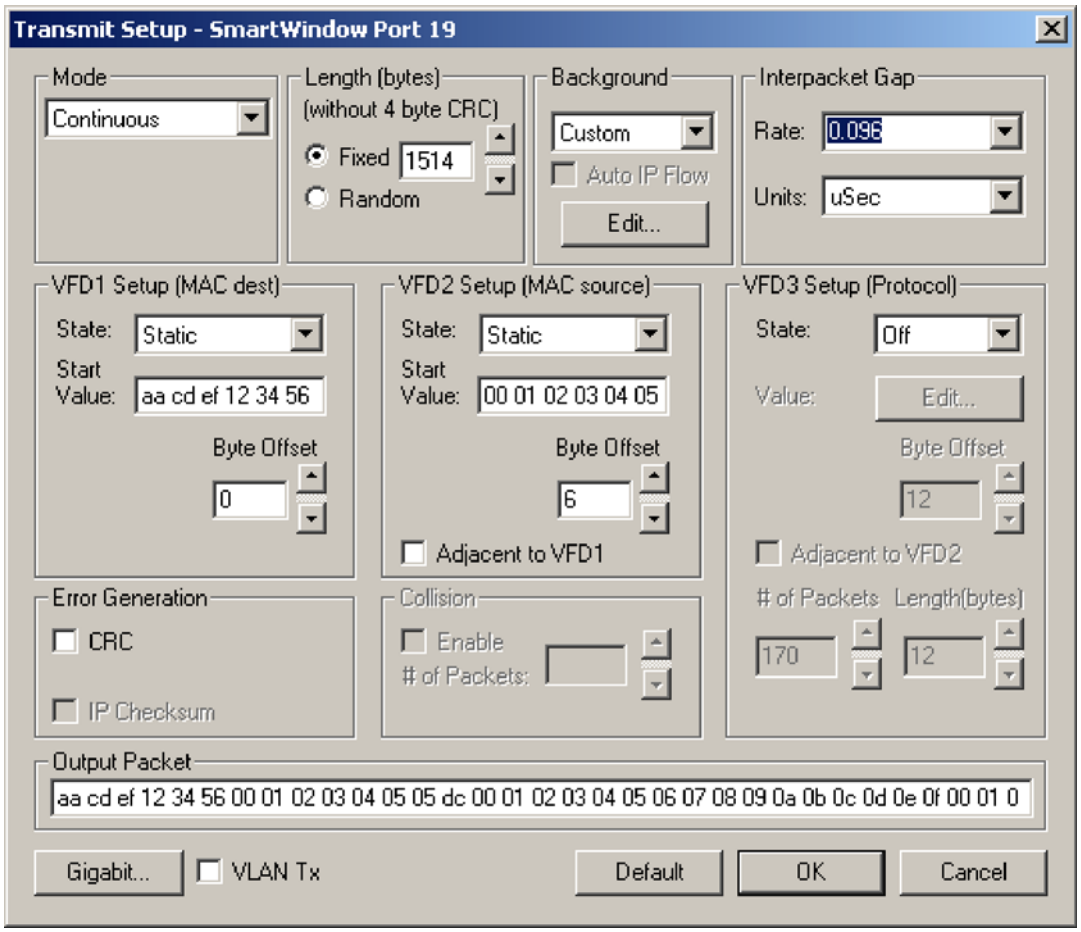


Figure 5. SmartBits Transmit Setup Window



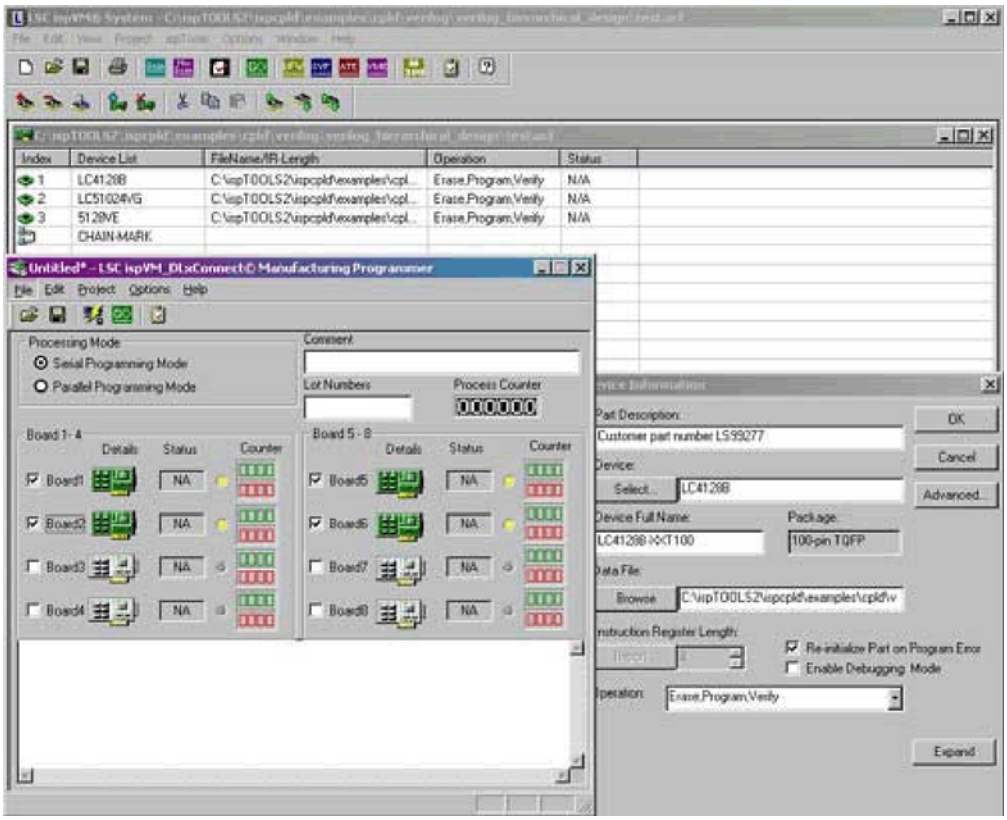
ispVM System

The ispVM System software is included with Lattice's ispLEVER® development tool suite, and is also available as a stand-alone device programming manager. ispVM 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 design tools and tools from other vendors. 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 is used in this interoperability test to download the LatticeECP3 bitstream, which configures the flexiPCS™ in Gigabit Ethernet mode.

Figure 6 shows a screen shot of the ispVM System.

Figure 6. ispVM System



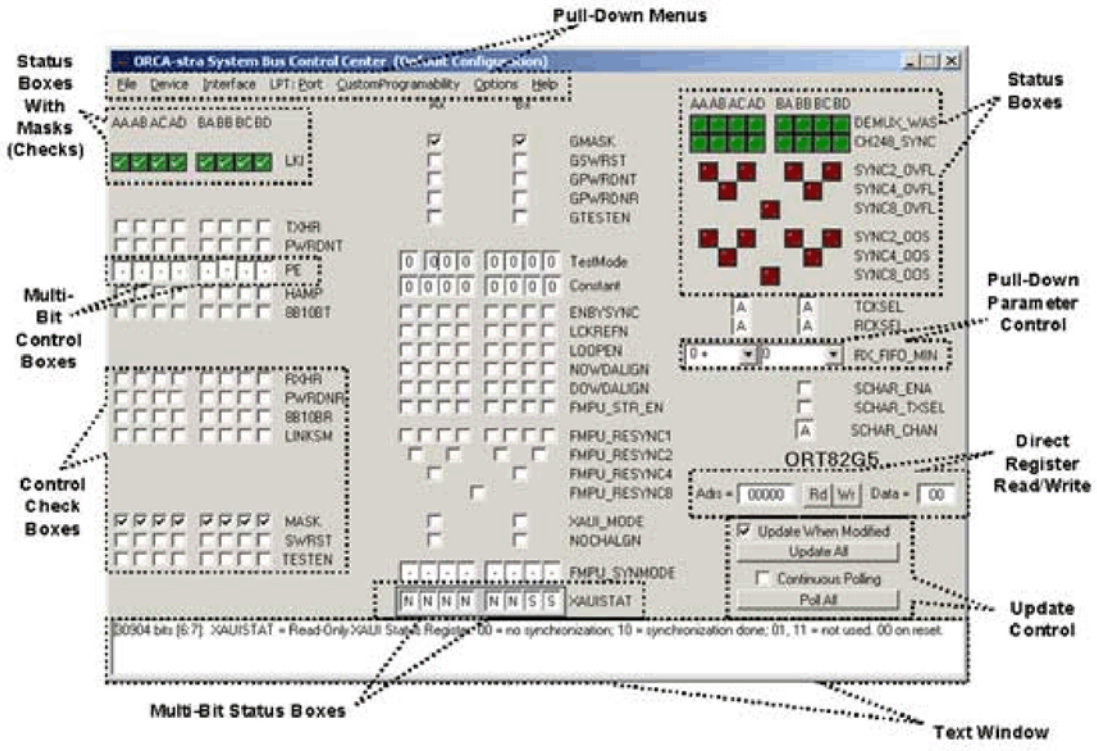
ORCAstra System

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 7 is a screen shot of the ORCAstra system.

Figure 7. ORCAstra System



Interoperability Testing

This section provides details on the SGMII Physical/MAC layer interoperability between a LatticeECP3 device and the Marvell 88E1111 PHY. This interoperability tests the correct processing of SGMII data from the 88E1111 PHY to the Tri-Speed Ethernet MAC IP Core, and then back in the reverse direction. The test verifies the ability to transfer packets across the system in an asynchronous way.

LatticeECP3 Serial Protocol Board Setup

- Ensure the ispVM JTAG cable is connected to the J12 header.
- Ensure an RJ-45 source of Ethernet traffic (SmartBits, PC) is connected to the 88E111 to the RJ-45 J37 connector.
- SW14, switch 2 should be in the upper position for Gigabit Ethernet SGMII mode and pressed down for SGMII mode. For this application, the switch is in the lower position.
- SW14, switch 1 should be in the upper position for SGMII PHY mode and pressed down for SGMII MAC mode (Note: this is only valid in SGMII mode). For this application, the switch position is in the lower (SGMII MAC mode) position.
- Make sure SW14, switch 8 is pressed down (selects internal Y2 125 MHz oscillator as reference clock for the LatticeECP3 reference design).
- Make sure the 88E1111 hardware configuration switches on the back of the LatticeECP3 Serial Protocol Board are set according to Table 2 (HWCFG_MODE= 0100 + other settings). Note that ORCAstra scripts are used after the LatticeECP3 loads to configure/monitor both the LattieECP3 reference design and 88E1111 registers.

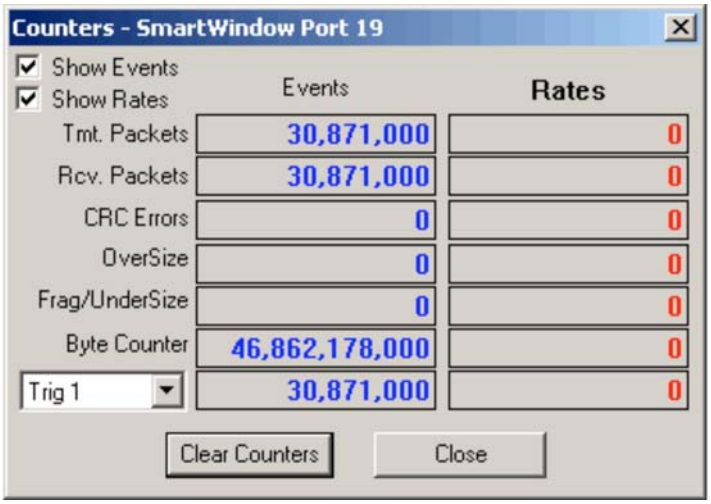
Table 2. LatticeECP3 Serial Protocol Board, Alaska Ultra 88E1111 HWCFG_MODE Settings

Switch#/CONFIG#	Switch # Pulled Down (Other Switches in Upper Position)	Selection/Value
SW7/CONFIG0	8	VSS/000
SW8/CONFIG1	4	LED_LINK1000/100
SW9/CONFIG2	1	VDD/111
SW10/CONFIG3	4	LED_LINK1000/100
SW11/CONFIG4	4	LED_LINK1000/100
SW12/CONFIG5	2	LED_LINK10
SW13/CONFIG6	8	VSS/000

SmartBits Setup

The SmartBits box is configured to advertise and run at 100BASE-T speed. The Transmit Traffic Setup is as shown in Figure 5. Once a link is established and auto-negotiation completes (100 Mbps) on both the copper and SGMII Serial SERDES sides, traffic is started and the SmartBits counters record statistical information on frames transmitted and received back error-free. This is shown in Figure 8.

Figure 8. SmartBits Counters



ORCAstra Setup

After the LatticeECP3 Serial Protocol Board is powered up, and the LatticeECP3 bitstream is downloaded, the following steps explain the procedure for configuring the Test Logic/Tri-Speed Ethernet MAC/SGMII IP/Marvell PHY registers via ORCAstra:

1. Ensure that a valid Ethernet link exists at the J37 RJ-45 connector and that the SmartBits Box is up and running.
2. Start ORCAstra from the ispLEVER installation directory
3. Select **Interface > 3. ispVM JTAG USB Interface**. If the **Select Target JTAG Device** window appears, select the first device, and click **OK**.
4. From the ORCAstra main window, select **ORCAstra > CustomProgrammability > Scripts > VBScripts**.
5. From the new window, choose **File > Open > < PROJECT_PATH>\init_mac_1000BASE-T_ALL.vbs**. This file is shown in Appendix B. This file configures both LatticeECP3 (MAC/SGMII AUTONEG) and Marvell PHY regis-

ters to run at 1000 Mbps full duplex and advertises full pause capabilities. It also configures the MAC in UNICAST mode and the local address is set to AA CD EF 12 34 56. This means that a link partner sending frames to the LatticeECP3 Serial Protocol Board will have to set the destination address to this value. Note that the SmartBits box traffic options do configure the Ethernet frames destination value to this address as shown in Figure 5. Click **Run**. The scripts will perform loops as it constantly checks whether auto-negotiation completes or restarts and for the link partner advertised PAUSE, ASYM pause capabilities to reconfigure its own MAC pause capabilities. init_mac_1000BASE-T_ALL.vbs is configured for 1000BASE-T but advertises 100/10 Mbps as well as full/half duplex for those speeds and re-configures itself based on LP abilities.

6. The 88E1111 D16-19 LED will indicate a valid copper side negotiated speed (D19 lit for dull duplex, and D17 lit for 100 Mbps). D25 should also be green indicating a good LatticeECP3 PCS link. D21 should be blue, indicating proper SGMII side auto-negotiation is complete. To exit the loop, use the **Shift** key while the mouse is inside the VBScript window.

Results

SmartBits traffic was run for just over an hour. Figure 8 shows that about 31 million Ethernet packets were successfully transmitted and recovered error-free.

Summary

In conclusion, the LatticeECP3 FPGA family offers users built-in SGMII Physical/MAC layer support and is fully inter-operable with the Marvell 88E1111 PHY.

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
December 2009	01.0	Initial release.
February 2012	01.1	Updated document with new corporate logo.

Appendix A. Lattice Tri-Speed Ethernet MAC/SGMII Reference Design Test Logic Map and Bit Descriptions

There are three address spaces in the test application design.

The first address space (Table 3) starts at offset 0x00 and ends at 0x35. This space contains the SGMII Tri-Speed Ethernet MAC core user registers as described in the Tri-Speed Ethernet MAC IP Core User's Guide.

The second address space (Table 4) starts at offset 0x40 and ends at 0x4D. This space contains SGMII/Gb Ethernet PCS IP registers as described in the SGMII and Gb Ethernet PCS IP Core User's Guide.

The third address space (Table 5) starts at offset 0x880 and ends at 0x8C1. This space contains ID, control, status and statistics registers used by the Test Logic Application.

Table 3. Tri-Speed Ethernet MAC Internal Registers

Register Description	Mnemonic	I/O Address	POR Value
Mode Register	MODE	00H - 01H	0000H
Transmit and Receive Control Register	TX_RX_CTL	02H - 03H	0000H
Maximum Packet Size Register	MAX_PKT_SIZE	04H - 05H	05EEH
Inter Packet Gap Register	IPG_VAL	08H - 09H	0048H
Tri-Speed MAC Address Register 0	MAC_ADDR_0	0AH - 0BH	0000H
Tri-Speed MAC Address Register 1	MAC_ADDR_1	0CH - 0DH	0000H
Tri-Speed MAC Address Register 2	MAC_ADDR_2	0EH - 0FH	0000H
Transmit and Receive Status	TX_RX_STS	12H - 13H	0000H
GMII Management Interface Control Register	GMII_MNG_CTL	14H - 15H	0000H
GMII Management Data Register	GMII_MNG_DAT	16H - 17H	0000H
VLAN Tag Length/type Register	VLAN_TAG	32H - 33H	0000H
Multicast_table_0	MLT_TAB_0	22H - 23H	0000H
Multicast_table_1	MLT_TAB_1	24H - 25H	0000H
Multicast_table_2	MLT_TAB_2	26H - 27H	0000H
Multicast_table_3	MLT_TAB_3	28H - 29H	0000H
Multicast_table_4	MLT_TAB_4	2AH - 2BH	0000H
Multicast_table_5	MLT_TAB_5	2CH - 2DH	0000H
Multicast_table_6	MLT_TAB_6	2EH - 2FH	0000H
Multicast_table_7	MLT_TAB_7	30H - 31H	0000H
Pause_opcode	PAUS_OP	34H - 35H	0080H

Table 4. SGMII/Gb Ethernet PCS IP Registers

Register Description	Mnemonic	I/O Address	POR Value
Control Register LO [7:0]	Control Register	40H	00H
Control Register HI [15:8]	Control Register	41H	10H
Status Register LO [7:0]	Status Register	42H	—
Status Register HI [15:8]	Status Register	43H	—
Advertised Ability LO[7:0]	mr_adv_ability	48H	01H ** read-only in
Advertised Ability HI[15:8]	mr_adv_ability	49H	40H MAC Mode
Link Partner Ability LO[7:0]	mr_lp_adv_ability	4AH	—
Link Partner Ability HI[15:8]	mr_lp_adv_ability	4BH	—
Auto Neg Expansion LO [7:0]	Auto Neg Expansion	4CH	—
Auto Neg Expansion HI [15:8]	Auto Neg Expansion	4DH	—

The REGINTF logic block in the test application provides the address decoding for the read-only and read/write registers used by the test logic and statistics counters. All registers are 8 bits wide and are byte addressable. Note that the statistics counter registers are composed of two 8-bit registers, a low and a high byte register. Therefore, in order to access these registers, two byte accesses must be made. For example, to access to all 16 bits of the RXOKCNT requires an access to both 0x899 (high byte) and 0x898 (low byte). Note that since the statistics counter registers are Clear On Read (COR), the high byte should be read first, before reading the low byte, since a read of the low byte clears all the combined 16 bits of the low and high registers. The address map for the test application related resisters is provided in Table 5.

Table 5. Test Logic Application Related Registers

Address	Register Description	Mnemonic	Type
0x0880	VERsion/IDentification Register	VERID	RO
0x0881	TeST CoNTroL Register	TSTCNTL	RW
0x0882	TeST CoNTroL Register 2	TSTCNTL_2	RW
0x0883	MAC CoNTroL Register	MACCNTL	RW
0x0884	PAUSe TiMeR Register - Low byte	PAUSTMRL	RW
0x0885	PAUSe TiMeR Register - High byte	PAUSTMRH	RW
0x0886	FIFO Almost Full Threshold Register - Low	FIFOAFTL	RW
0x0887	FIFO Almost Full Threshold Register - High	FIFOAFTH	RW
0x0888	FIFO Almost Empty Threshold Register - Low	FIFOAETL	RW
0x0889	FIFO Almost Empty Threshold Register - High	FIFOAETH	RW
0x088a	RX Status Register	RXSTATUS	RO/COR
0x088b	TX Status Register	TXSTATUS	RO/COR
0x088c, 0x088d	RX Packet Ignored Counter Register (L,H)	RXPICNT	RO/COR
0x088e, 0x088f	RX Length Check Error CouNTer (L,H)	RXLCECNT	RO/COR
0x0890, 0x0891	RX Long Frames CouNTer Register (L,H)	RXLFCNT	RO/COR
0x0892, 0x0893	RX Short Frames CouNTer Register (L,H)	RXSFCNT	RO/COR
0x0894, 0x0895	RX IPG violations CouNTer Register (L,H)	RXIPGCNT	RO/COR
0x0896, 0x0897	RX CRC errors CouNTer Register (L,H)	RXCRCCNT	RO/COR
0x0898, 0x0899	RX OK packets CouNTer Register (L,H)	RXOKCNT	RO/COR
0x089a, 0x089b	RX Control Frame CouNTer Register (L,H)	RXCFCNT	RO/COR
0x089c, 0x089d	RX Pause Frame CouNTer Register (L,H)	RXPFCNT	RO/COR
0x089e, 0x089f	RX Multicast Frame CouNTer Register (L,H)	RXMFCNT	RO/COR
0x08a0, 0x08a1	RX Broadcast Frame CouNTer Register (L,H)	RXBFCNT	RO/COR
0x08a2, 0x08a3	RX VLAN tagged Frame CouNTer Register (L,H)	RXVFCNT	RO/COR
0x08a4, 0x08a5	TX Unicast Frame CouNTer Register (L,H)	TXUFCNT	RO/COR
0x08a6, 0x08a7	TX Pause Frame CouNTer Register (L,H)	TXPFCNT	RO/COR
0x08a8, 0x08a9	TX Multicast Frame CouNTer Register (L,H)	TXMFCNT	RO/COR
0x08aa, 0x08ab	TX Broadcast Frame CouNTer Register (L,H)	TXBFCNT	RO/COR
0x08ac, 0x08ad	TX VLAN tagged Frame CouNTer Register (L,H)	TXVFCNT	RO/COR
0x08ae, 0x08af	TX BAD FCS Frame CouNTer Register (L,H)	TXBFCCNT	RO/COR
0x08b0, 0x08b1	TX Jumbo Frame CouNTer Register (L,H)	TXJFCNT	RO/COR
0x08b2, 0x08b3	UNUSED		
0x08b4, 0x08b5	TX Packet Generator DEST ADD (B1,B2)	PG_DA_W1	RW
0x08b6, 0x08b7	TX Packet Generator DEST ADD (B3,B4)	PG_DA_W2	RW
0x08b8, 0x08b9	TX Packet Generator DEST ADD (B5,B6)	PG_DA_W3	RW
0x08ba, 0x08bb	TX Packet Generator SRC ADD (B1,B2)	PG_SA_W1	RW
0x08bc, 0x08bd	TX Packet Generator SRC ADD (B3,B4)	PG_SA_W2	RW

Table 5. Test Logic Application Related Registers (Continued)

Address	Register Description	Mnemonic	Type
0x08be, 0x08bf	TX Packet Generator SRC ADD (B5,B6)	PG_SA_W3	RW
0x08c0, 0x08c1	TX Packet Generator PYLD LEN (L,H)	PG_PL	RW
0x08c2 - 0x0FFF	UNUSED		

Register Bit Descriptions

0x0880 **VERsion/IDentification Register** VERID Read Only
default value = 0xA3

0x0881 **TeST CoNTroL Register** TSTCNTL Read/Write
default value = 0x00

bit_0 = destination/source address swap - 1 = swap, 0 = no swap
bit_1 = loop back enable - 1 = loopback, 0 = no loopback
bit_2 = reset_phy_n - 1 = no reset, 0 = reset PHY device
bit_3 = pkt_loop_clksel 1 = sys_clk tied to rx_clk, 0 = sys_clk from external pin
bit_4 = flag_large_pkt_en – 1 = enables Rx pkts larger than those set in Max_PKT_SIZE reg to be flagged
bit_5 = flag_errored_pkt_en – 1 = enables Rx errored pkts (Rx_error set) to be flagged
bit_6 = flag_pause_pkt_en – 1 = enables pause pkts be flagged
bit_7 = drop_flagged_en – 1 = Enables the Discarding of flagged pkts.

0x0882 **TeST CoNTroL Register 2** TSTCNTL 2 Read/Write
default value = 0x00

bit_0 = pkt_gen_en – 1 = enable Tx packet generator
bit_1 = pkt_mode[0] – mode 00 = gen single pkt, 01 = continous pkts, 10 = a burst of pkts
bit_2 = pkt_mode[1] – mode 00 = gen single pkt, 01 = continous pkts, 10 = a burst of pkts
bit_3 = pkt_gen_burst_size [0] = 0000 – 1 pkt, 0001 – 2pkts , …, 1111 – 15 pkts
bit_4 = pkt_gen_burst_size [1] = 0000 – 1 pkt, 0001 – 2pkts , …, 1111 – 15 pkts
bit_5 = pkt_gen_burst_size [2] = 0000 – 1 pkt, 0001 – 2pkts , …, 1111 – 15 pkts
bit_6 = pkt_gen_burst_size [3] = 0000 – 1 pkt, 0001 – 2pkts , …, 1111 – 15 pkts
bit_7 = not used

0x0883 **MAC CoNTroL Register** MACCNTL Read/Write
default value = 0x00

bit_0 = send pause request - 1 = send request, 0 = don't send request
This reg bit gets ORed with the tx_fifo almost full signal. The ORed output sets the tx_sndpausreq pin on the tsmac core - see TSMAC user's guide for more information.

bit_1 = fifo control frame
This reg bit sets the tx_fifoctrl pin on the Tri-Speed Ethernet MAC IP Core. See the Tri-Speed Ethernet MAC IP Core User's Guide for more information.

bit_2 = Not used.

bit_3 = tx_fifo empty
This register bit is "OR'd" with the tx_fifo empty signal. The OR'd output sets the tx_fifo_empty pin on the Tri-Speed Ethernet MAC IP Core. See the Tri-Speed Ethernet MAC IP Core User's Guide for more information. Note that by setting this bit you can mimic the TX FIFO being empty.

bit_4 = ignore next packet
This register bit sets the ignore_next_pkt pin on the Tri-Speed Ethernet MAC IP Core. See the Tri-Speed Ethernet MAC IP Core User's Guide for more information.

bit_5 = Rx_fifo_flush – 1 = flush Rx FIFO

bit_6 = Tx_fifo_flush – 1 = flush Tx FIFO

bit_7 =Not used.

0x0884 **PAUSe TiMeR Register - Low byte** PAUSTMRL Read/Write
0x0885 **PAUSe TiMeR Register - High byte** PAUSTMRH Read/Write
default value = 0x0000

Low Register:

bit[7:0] = pause timer low bits
These register bits set the tx_sndpaustim[7:0] pins on the Tri-Speed Ethernet MAC IP Core. See the Tri-Speed Ethernet MAC IP Core User's Guide for more information.

High Register:

bit[7:0] = pause timer high bits
These register bits set the tx_sndpaustim[15:8] pins on the Tri-Speed Ethernet MAC IP Core. See the Tri-Speed Ethernet MAC IP Core User's Guide for more information.

0x0886 **FIFO Almost Full Threshold Register - Low** FIFOAFTL Read/Write
0x0887 **FIFO Almost Full Threshold Register - High** FIFOAFTH Read/Write
default value = 0x0000

Low Register:

bit[7:0] = FIFO Almost Full Threshold - Low
These reg bits set the loopback fifo threshold [7:0] bits

High Register:

bit[0] = FIFO Almost Full Threshold - High
This reg bit set the loopback fifo [8] bit

bit[7:1] = Not Used

0x0888 **FIFO Almost Empty Threshold Register - Low** FIFOAETL Read/Write
0x0889 **FIFO Almost Empty Threshold Register - High** FIFOAETH Read/Write
default value = 0x0000

Low Register:

bit[7:0] = FIFO Almost Empty Threshold - Low
These register bits set the loopback FIFO threshold [7:0] bit.

High Register:

bit[0] = FIFO Almost Empty Threshold - High
This register bit sets the loopback FIFO [8] bit.

bit[7:1] = Not used.

0x088a **RX Status Register** RXSTATUS Read Only/clear on read
default value = 0x00

0x088b **TX Status Register** TXSTATUS Read Only/clear on read
default value = 0x00

RX status:

bit_0 = latches rx_error value from tsmac core pin
bit_1 = latches rx_fifo_error value from tsmac core pin
bit_2 = latches rx_fifo_full value from test logic loopback fifo
bit_3 = Not used
bit_4 = Not used
bit_5 = Not used
bit_6 = Not used
bit_7 = Not used

TX status:

bit_0 = latches tx_discfrm value from tsmac core pin
bit_1 = latches tx_fifo_full value from test logic loopback fifo
bit_2 = Not used
bit_3 = Not used
bit_4 = Not used
bit_5 = Not used
bit_6 = Not used
bit_7 = Not used

The following counter registers are all 16 bits with 8-bit low and 8-bit high address locations. The counters count different Rx and Tx statistics as defined by the statistics vectors in the Tri-Speed Ethernet MAC IP Core User's Guide. All counters have a power-on default value of 0x0000.

0x088c	**RX Packet Ignored Counter Register**	RXPICNT	RO/COR
0x088e	**RX Length Check Error CouNTer**	RXLCECNT	RO/COR
0x0890	**RX Long Frames CouNTer Register**	RXLFCNT	RO/COR
0x0892	**RX Short Frames CouNTer Register**	RXSFCNT	RO/COR
0x0894	**RX IPG violations CouNTer Register**	RXIPGCNT	RO/COR
0x0896	**RX CRC errors CouNTer Register**	RXCRCCNT	RO/COR

0x0898	**RX OK packets CouNTer Register**	RXOKCNT	RO/COR
0x089a	**RX Control Frame CouNTer Register**	RXCFCNT	RO/COR
0x089c	**RX Pause Frame CouNTer Register**	RXPFCNT	RO/COR
0x089e	**RX Multicast Frame CouNTer Register**	RXMFCNT	RO/COR
0x08a0	**RX Broadcast Frame CouNTer Register**	RXBFCNT	RO/COR
0x08a2	**RX VLAN tagged Frame CouNTer Register**	RXVFCNT	RO/COR
0x08a4	**TX Unicast Frame CouNTer Register**	TXUFCNT	RO/COR
0x08a6	**TX Pause Frame CouNTer Register**	TXPFCNT	RO/COR
0x08a8	**TX Multicast Frame CouNTer Register**	TXMFCNT	RO/COR
0x08aa	**TX Broadcast Frame CouNTer Register**	TXBFCNT	RO/COR
0x08ac	**TX VLAN tagged Frame CouNTer Register**	TXVFCNT	RO/COR
0x08ae	**TX BAD FCS Frame CouNTer Register**	TXBFCCNT	RO/COR
0x08b0	**TX Jumbo Frame CouNTer Register**	TXJFCNT	RO/COR

Appendix B. init_mac_1000BASE-T_all.vbs ORCAstra Visual Basic Script

```
Sub Main()

'NOTE; 1000 Mbps Half Duplex is not supported

        Call V.SPut(&H00800,&h00) ' to host bus - mode reg 00 DISABLE TX/RX

        Call V.SPut(&H00884,&h0F) ' TX PAUTE TIME LOW BYTE-0F

    Call V.SPut(&H00886,&hC1) 'fifo AFL C1
    Call V.SPut(&H00887,&h01) 'fifo AFH 01
    Call V.SPut(&H00888,&h05) 'fifo AEL 05
     Call V.SPut(&H00881,&hFE) 'tstcntl (b0=swap,b1=loop_en,b2=reset_phy_n,b3(0=sys_clk from
external pin),b4=flaglargepackets,b5=flarerrorpkt,b6=flagpause,b7-dropflagged)-FE
          ' 1518 bytes = 0x5ee
    'Call V.SPut(&H008b2,&h77) 'max_pkt_sz L    EE
    'Call V.SPut(&H008b3,&hA0) 'max_pkt_sz H    05

    ' Pkt generator registers
        Call V.SPut(&H008b4,&h0A) '  pkt_gen_destadd_B1 0A
        Call V.SPut(&H008b5,&h0B) '  pkt_gen_destadd_B2 0B
        Call V.SPut(&H008b6,&h0C) '  pkt_gen_destadd_B3 0C
        Call V.SPut(&H008b7,&h0D) '  pkt_gen_destadd_B4 0D
        Call V.SPut(&H008b8,&h0E) '  pkt_gen_destadd_B5 0E
        Call V.SPut(&H008b9,&h0F) '  pkt_gen_destadd_B6 0F
        Call V.SPut(&H008ba,&h01) '  pkt_gen_srcadd_B1 01
        Call V.SPut(&H008bb,&h02) '  pkt_gen_srcadd_B2 02
        Call V.SPut(&H008bc,&h03) '  pkt_gen_srcadd_B3 03
        Call V.SPut(&H008bd,&h04) '  pkt_gen_srcadd_B4 04
        Call V.SPut(&H008be,&h05) '  pkt_gen_srcadd_B5 05
        Call V.SPut(&H008bf,&h06) '  pkt_gen_srcadd_B6    06

           ' 46 bytes = 0x2e
        Call V.SPut(&H008c0,&h2E) '  pkt_gen_pyld_len L 2e
        Call V.SPut(&H008c1,&h00) '  pkt_gen_pyld_len H

    ' MDIO registers
'       `ifdef MIIM_MODULE
'        Call V.SPut(&H00816,&h00) ' to  - MDIO DATA reg
'        Call V.SPut(&H00817,&h80) ' to  - MDIO DATA reg
'        Call V.SPut(&H00814,&h00) ' to  - MDIO ACCESS CTL reg
'        Call V.SPut(&H00815,&h21) ' to  - MDIO ACCESS CTL reg
'    #20000
'        jtag_drv.JTAG_read_byte(&H00814, read_data) ' to  - MDIO ACCESS CTL reg
'        jtag_drv.JTAG_read_byte(&H00815, read_data) ' to  - MDIO ACCESS CTL reg
'       `endif


    ' MAC registers  AA CD EF 12 34 56
        Call V.SPut(&H0080a,&hCD)           ' MAC Addr reg 0 CD
        Call V.SPut(&H0080b,&hAA)           ' MAC Addr reg 0 AA
        Call V.SPut(&H0080c,&h12)           ' MAC Addr reg 1 12
        Call V.SPut(&H0080d,&hEF)           ' MAC Addr reg 1 EF
        Call V.SPut(&H0080e,&h56)           ' MAC Addr reg 2 56
        Call V.SPut(&H0080f,&h34)           ' MAC Addr reg 2 34

        'Don't Drop Control (will be dropped by test logic), UNICAST :9A
        Call V.SPut(&H00802,&h9A) ' to host bus - TX_RX_CTL H reg 9A
```

```
    Call V.SPut(&H00803,&h00) ' to host bus - TX_RX_CTL L -NO short pkts

'12 Bytes IPG
    Call V.SPut(&H00808,&h0C) ' to host bus - IPG reg 0C
'Gbit enable
    Call V.SPut(&H00800,&h0F) ' to host bus - mode reg 0F

'WRITE TO 88E1111 CONFIG REGS


'write to register 22 to switch to page 0 (copper mode)
Temp0="&H00"  'Most significant byte
Temp1="&H00" 'Least significant , page 0
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 22  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)



'write to register 4 (COPPER AUTONEG register
'Most significant byte,bits 15=NP,14=Ack,13=remotefault,12=reserved
'Most significant byte,bits 11(asym pause=1),10(Pause=1),9=100BASE-T4,8=100BASE-TX-FD
Temp0="&H0D"
'Least significant byte,bits 7=100BASE-TX-HD,6=10BASE-TX-FD),5=10BASE-TX-HD,4=0
'Least significant byte,bits 3-0=0001
Temp1="&HE1" 'Least significant byte,don't advertise 10/100 speeds
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 4  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'write to register 9 (COPPER 1000BASE-T control  register
'bits 15-13=000(NORMAL),12=0=Automatic Master Slave
'bits 11=0=ManualconfigSlave),10=0(Prefer single port),9=1000BASE-T-FD,8=1000BASE-T-HD
Temp0="&H02"  'Most significant byte,bit9(Full Duplex=1), bit8(Half Duplex=0),
'Least significant byte,bits 7-0=RESERVED
Temp1="&H00" 'Least significant Reserved
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 9  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'write to register 16 (PHY SPECIFIC  control  register
Temp0="&H03"  'Most significant byte
Temp1="&H68" 'Least significant , bit4=Disable 125CLK=0
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 16  'address
```

```
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'write to register 27 (Extended PHY SPECIFIC  status  register
Temp0="&H84"  'Most significant byte, bit15=1 (disable  copper/fiber auto detection
Temp1="&H84" 'Least significant , bit3:0=HWCFG_MODE=0100=SGMII without clock
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 27  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'write to register 22 to switch to page 1 (fiber mode)
Temp0="&H00"  'Most significant byte
Temp1="&H01" 'Least significant , page 1
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 22  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'write to register page 1, reg 0 (Fiber Control Register to enable SGMII AN
'BIT15=Reset,BIT14=Loopback,BIT13=SPEED(0), BIT12=ANEGENABLE
'BIT11=PWRDWN,BIT10=Isolate,BIT9=ANRESTART, BIT8=ENDUPLEX
Temp0="&H11"  'Most significant byte
'BIT7=ENCOL,BIT6=SPEED(1),BIT5:4=RESERVED
'BIT3:0=RESERVED
Temp1="&H40" 'Least significant
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 0  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)



'PROGRAM Lattice SGMII  AN register!
' mr_adv_ability[7:0], SGMII MAC MODE: bits 7-1=0, bit0=1
    Call V.SPut(&H00848,&h01)           ' mr_adv_ability[7:0] 01
' mr_adv_ability[15:8], SGMII MAC MODE: bit14=1 Others =0
    Call V.SPut(&H00849,&h40)           ' mr_adv_ability[15:8] 40
' mr_autoneg expansion [7:0], bit1=mr_page_rx=0
    Call V.SPut(&H0084C,&h00)           ' mr_autoneg expansion [7:0] 00
' mr_control[15:8], bit 15=ANRESET=0, bit12=ANENABLE=1, bit9=ARRESTART=1
    Call V.SPut(&H00841,&h12)           ' mr_control[15:8] 12
V.Wait(100)


'write to register 22 to switch to page 0 (copper mode)
Temp0="&H00"  'Most significant byte
Temp1="&H00" 'Least significant , page 0
```

```
    'GMII_MNGMT_DAT [15:0]
    Call V.SPut(&H00816,Temp1)'[7:0]
    Call V.SPut(&H00817,Temp0)'[15:8]
    addr = 22  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)


    'write to register page 0, reg 0 (Copper Control Register to Soft Reset
    'BIT15=Reset,BIT14=Loopback,BIT13=SPEED(0), BIT12=ANEGENABLE
    'BIT11=PWRDWN,BIT10=Isolate,BIT9=ANRESTART, BIT8=ENDUPLEX
    Temp0="&H91"  'Most significant byte
    'BIT7=ENCOL,BIT6=SPEED(1),BIT5:4=RESERVED
    'BIT3:0=RESERVED
    Temp1="&H40" 'Least significant
    'GMII_MNGMT_DAT [15:0]
    Call V.SPut(&H00816,Temp1)'[7:0]
    Call V.SPut(&H00817,Temp0)'[15:8]
    addr = 0  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)

    'write to register 22 to switch to page 0 (copper mode)
    Temp0="&H00"  'Most significant byte
    Temp1="&H00" 'Least significant , page 0
    'GMII_MNGMT_DAT [15:0]
    Call V.SPut(&H00816,Temp1)'[7:0]
    Call V.SPut(&H00817,Temp0)'[15:8]
    addr = 22  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)



        'Send a single pkt from pkt generator
       'Call V.SPut(&H00882,&h01) '  TSTCNTL2 - pkt_gen control reg  01
    'V.Wait(100)
    'MsgBox "Write completed"

'V.Show_Display()
'V.Clear_Display()
'write to register 22 to switch to page 0 (copper mode)
Temp0="&H00"  'Most significant byte
Temp1="&H00" 'Least significant , page 0
'GMII_MNGMT_DAT [15:0]
Call V.SPut(&H00816,Temp1)'[7:0]
Call V.SPut(&H00817,Temp0)'[15:8]
addr = 22  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
V.Wait(100)

'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:")
addr = 1  'address
Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
Call V.SPut(&H00815,&H00)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
```

```
V.Wait(100)
'GMII_MNGMT_DAT [15:0]
Temp0 = V.SGet(&H00816)  '[7:0]
Temp1 = V.SGet(&H00817)  '[15:8]
Total = Temp0 + (Temp1*2^8)
myhex = hex(Total)
'V.Echo("BITS 15-9 are ABILITY registers")
'V.Echo("BIT15=100BASE-T4,BIT14=100BASE-X-FD,BIT13=100BASE-X-HD), BIT12=10Mbps-FD")
'V.Echo("BIT11=10Mbps-HD ,BIT10=100BASE-T2-FD,BIT9=100BASE-T2-HD, BIT8=ExtendedStatus")
'V.Echo("BIT7=RESERVED,BIT6=MFPreambSupp,BIT5=Copper-AN-Complete,BIT4=COPPER_REMOTE_FAULT")
'V.Echo("BIT3=AN_ABILITY(1),BIT2=Copper_LINK_STATUS,BIT1=JABBER_DETECTED,BIT0=EXTENDEDREGIS-
TERS")
'V.Echo("=0x"& HexX(Total,4) &"")
'V.Echo("")

ANStatusReg=HexX(Total,4) '2 bytes
'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:= 0x" & ANStatusReg &"")
ANcomplete= "&H" & ANStatusReg And &H0020 'bit 5
'V.Echo("ANcomplete= 0x" & HexX(ANcomplete,4) &"")

Do
    Do until ( ANcomplete <> &H0000)   'wait until ANcomplete bit is 1
        'V.Echo("WAITING FOR ANCOMPLETE=1")
        'write to register 22 to switch to page 0 (copper mode)
        Temp0="&H00"  'Most significant byte
        Temp1="&H00" 'Least significant , page 0
        'GMII_MNGMT_DAT [15:0]
        Call V.SPut(&H00816,Temp1)'[7:0]
        Call V.SPut(&H00817,Temp0)'[15:8]
        addr = 22  'address
        Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
        Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
        V.Wait(100)

        'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:")
        addr = 1  'address
        Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
        Call V.SPut(&H00815,&H00)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
        V.Wait(100)
        'GMII_MNGMT_DAT [15:0]
        Temp0 = V.SGet(&H00816)  '[7:0]
        Temp1 = V.SGet(&H00817)  '[15:8]
        Total = Temp0 + (Temp1*2^8)
        myhex = hex(Total)
        'V.Echo("BITS 15-9 are ABILITY registers")
        'V.Echo("BIT15=100BASE-T4,BIT14=100BASE-X-FD,BIT13=100BASE-X-HD), BIT12=10Mbps-FD")
          'V.Echo("BIT11=10Mbps-HD ,BIT10=100BASE-T2-FD,BIT9=100BASE-T2-HD, BIT8=ExtendedSta-
tus")
                                    'V.Echo("BIT7=RESERVED,BIT6=MFPreambSupp,BIT5=Copper-AN-Com-
plete,BIT4=COPPER_REMOTE_FAULT")
         'V.Echo("BIT3=AN_ABILITY(1),BIT2=Copper_LINK_STATUS,BIT1=JABBER_DETECTED,BIT0=EXTEND-
EDREGISTERS")
        'V.Echo("=0x"& HexX(Total,4) &"")
        'V.Echo("")
        ANStatusReg=HexX(Total,4) '2 bytes
        'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:= 0x" & ANStatusReg &"")
        ANcomplete= "&H" & ANStatusReg And &H0020 'bit 5
        'V.Echo("ANcomplete= 0x" & HexX(ANcomplete,4) &"")
        V.DO_EVENTS 'allow other events to happen while in loop
```

```
        If V.shiftkey_pressed Then Exit Sub 'If shift Key pressed, then exit

    Loop

    'write to register 22 to switch to page 0 (copper mode)
    Temp0="&H00"  'Most significant byte
    Temp1="&H00" 'Least significant , page 0
    'GMII_MNGMT_DAT [15:0]
    Call V.SPut(&H00816,Temp1)'[7:0]
    Call V.SPut(&H00817,Temp0)'[15:8]
    addr = 22  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)

'    V.Echo("PAGE 0, REG 5 :Link Partner Ability-Base Page-Copper[15:0]:")
    addr = 5  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H00)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)
    'GMII_MNGMT_DAT [15:0]
    Temp0 = V.SGet(&H00816)  '[7:0]
    Temp1 = V.SGet(&H00817)  '[15:8]
    Total = Temp0 + (Temp1*2^8)
    myhex = hex(Total)
'    V.Echo("BIT15=NP_ABLE,BIT14=ACK,BIT13=REMOTE_FAULT, BIT12=TECH_ABILITY")
'    V.Echo("BIT11=ASYM_PAUSE ,BIT10=PAUSED,BIT9=100BASE-T4, BIT8=100BASE-TX-FD")
'    V.Echo("BIT7=100BASE-TX-HD,BIT6=10BASE-TX-FD,BIT5=10BASE-TX-HD,BIT4=SelectField[[4]")
'    V.Echo("BIT3-0: SelectField[[3:0]")
'    V.Echo("=0x"& HexX(Total,4) &"")
'    V.Echo("")

    LPAdvReg=HexX(Total,4) '2 bytes
   'V.Echo("PAGE 0, REG 5 :Link Partner Ability-Base Page-Copper[15:0] = 0x" & LPAdvReg &"")
    LPPause=  "&H" &  LPAdvReg And &H0400 'bit 10
    'V.echo ("LPPause=" & HexX(LPPause,4) & "")'
    LPAsymPause=  "&H" &  LPAdvReg And &H0800 ' bit 11
    'V.echo ("LPAsymPause=" &  HexX(LPAsymPause,4) & "")
    LP100FDUP=  "&H" &  LPAdvReg And &H0100 ' bit 8
    LP100HDUP=  "&H" &  LPAdvReg And &H0080 ' bit 7
    LP10FDUP=  "&H" &  LPAdvReg And &H0040 ' bit 6
    LP10HDUP=  "&H" &  LPAdvReg And &H0020 ' bit 5


'    V.Echo("PAGE 0, REG 10 : 1000BASE-T Status Register-Copper [15:0]:")
    addr = 10  'address
    Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
    Call V.SPut(&H00815,&H00)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
    V.Wait(100)
    'GMII_MNGMT_DAT [15:0]
    Temp0 = V.SGet(&H00816)  '[7:0]
    Temp1 = V.SGet(&H00817)  '[15:8]
    Total = Temp0 + (Temp1*2^8)
    myhex = hex(Total)
'                                                                                V.Echo("BITS
15=MASTER_SLAVE_FAULT,14=MASTER_NOTSLAVE,13=LOCAL_RECEIVER_OK,12=REMOTE_RECEIVER_OK")
'    V.Echo("BIT11=LP_1000BASE-T-FD-ABLE ,BIT10=LP_1000BASE-T-HD-ABLE,BIT9-8=RESERVED")
'    V.Echo("BIT7-0=IDLE ERROR COUNTER-CLEARONREAD")
'    V.Echo("=0x"& HexX(Total,4) &"")
```

```
'    V.Echo("")
    LP1000BStatus=HexX(Total,4) '2 bytes
    LP1000FDUP=  "&H" &  LP1000BStatus And &H0800 ' bit 11
    LP1000HDUP=  "&H" &  LP1000BStatus And &H0800 ' bit 11

       If  (LP1000FDUP <> &H0000) Then ' 1000 Mbps FULL DUPLEX
           If  (LPPause = &H0000) Then
               If  (LPAsymPause = &H0000) Then
                   'PAUSE=0, ASYM_PAUSE=0
                   'DISABLE PAUSE TRANSMIT AND RECEIVE
                   Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                   NEWMACCONTROLREG=  &H92 ' DISABLE RX PAUSE, BIT3
                   Call V.SPut(&H00802,NEWMACCONTROLREG)
                   Call V.SPut(&H00800,&H0D) ' DISABLE TX  MAC PAUSE , bit 1

               Else
                   'PAUSE=0, ASYM_PAUSE=1
                   'DISABLE PAUSE TRANSMIT , Enable PAUSE RECEIVE
                   Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                   NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
                   Call V.SPut(&H00802,NEWMACCONTROLREG)
                   Call V.SPut(&H00800,&H0D) ' DISABLE TX  MAC PAUSE , bit 1

               End If

        Else
            'PAUSE=1, ASYM_PAUSE=DON'T CARE
            'Enable PAUSE TRANSMIT , Enable PAUSE RECEIVE
            Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
               NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
            'V.echo ("NEWMACCONTROLREG=" & Hex(NEWMACCONTROLREG) & "")'
            Call V.SPut(&H00802,NEWMACCONTROLREG)
            Call V.SPut(&H00800,&H0F) ' Enable TX  MAC PAUSE , bit 1
        End If

       ElseIf  (LP100FDUP <> &H0000) Then ' 100 Mbps FULL DUPLEX
           If  (LPPause = &H0000) Then
               If  (LPAsymPause = &H0000) Then
                   'PAUSE=0, ASYM_PAUSE=0
                   'DISABLE PAUSE TRANSMIT AND RECEIVE
                   Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                   NEWMACCONTROLREG=  &H92 ' DISABLE RX PAUSE, BIT3
                   Call V.SPut(&H00802,NEWMACCONTROLREG)
                   Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1

               Else
                   'PAUSE=0, ASYM_PAUSE=1
                   'DISABLE PAUSE TRANSMIT , Enable PAUSE RECEIVE
                   Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                   NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
                   Call V.SPut(&H00802,NEWMACCONTROLREG)
                   Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1

               End If

        Else
            'PAUSE=1, ASYM_PAUSE=DON'T CARE
            'Enable PAUSE TRANSMIT , Enable PAUSE RECEIVE
            Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
```

```
            NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
         'V.echo ("NEWMACCONTROLREG=" & Hex(NEWMACCONTROLREG) & "")'
         Call V.SPut(&H00802,NEWMACCONTROLREG)
         Call V.SPut(&H00800,&H0E) ' Enable TX  MAC PAUSE , bit 1
     End If

    ElseIf  (LP100HDUP <> &H0000) Then ' 100 Mbps Half DUPLEX
            'DISABLE PAUSE TRANSMIT AND RECEIVE
            Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
            NEWMACCONTROLREG=  &HB2 ' DISABLE RX PAUSE, BIT3, enable HDUP
            Call V.SPut(&H00802,NEWMACCONTROLREG)
            Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1


    ElseIf  (LP10FDUP <> &H0000) Then ' 10 Mbps FULL DUPLEX
        If  (LPPause = &H0000) Then
            If  (LPAsymPause = &H0000) Then
                'PAUSE=0, ASYM_PAUSE=0
                'DISABLE PAUSE TRANSMIT AND RECEIVE
                Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                NEWMACCONTROLREG=  &H92 ' DISABLE RX PAUSE, BIT3
                Call V.SPut(&H00802,NEWMACCONTROLREG)
                Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1

            Else
                'PAUSE=0, ASYM_PAUSE=1
                'DISABLE PAUSE TRANSMIT , Enable PAUSE RECEIVE
                Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
                NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
                Call V.SPut(&H00802,NEWMACCONTROLREG)
                Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1

            End If

     Else
         'PAUSE=1, ASYM_PAUSE=DON'T CARE
         'Enable PAUSE TRANSMIT , Enable PAUSE RECEIVE
         Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
            NEWMACCONTROLREG=  &H9A ' Enable RX PAUSE,BIT3
         'V.echo ("NEWMACCONTROLREG=" & Hex(NEWMACCONTROLREG) & "")'
         Call V.SPut(&H00802,NEWMACCONTROLREG)
         Call V.SPut(&H00800,&H0E) ' Enable TX  MAC PAUSE , bit 1
     End If

    ElseIf  (LP10HDUP <> &H0000) Then ' 10 Mbps Half DUPLEX
            'DISABLE PAUSE TRANSMIT AND RECEIVE
            Call V.SPut(&H00800,&H00)' DISABLE MAC FIRST
            NEWMACCONTROLREG=  &HB2 ' DISABLE RX PAUSE, BIT3, enable HDUP
            Call V.SPut(&H00802,NEWMACCONTROLREG)
            Call V.SPut(&H00800,&H0C) ' DISABLE TX  MAC PAUSE , bit 1

    Else  ' NO MATCH
     Call V.SPut(&H00800,&H00)' DISABLE TX/RX MAC


End If
```

```
    Do until ( ANcomplete = &H0000)   'wait until ANcomplete bit is 0
        'V.Echo("WAITING FOR ANCOMPLETE=0")
        'write to register 22 to switch to page 0 (copper mode)
        Temp0="&H00"  'Most significant byte
        Temp1="&H00" 'Least significant , page 0
        'GMII_MNGMT_DAT [15:0]
        Call V.SPut(&H00816,Temp1)'[7:0]
        Call V.SPut(&H00817,Temp0)'[15:8]
        addr = 22  'address
        Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
        Call V.SPut(&H00815,&H20)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
        V.Wait(100)

        'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:")
        addr = 1  'address
        Call V.SPut(&H00814, addr)'[7:0] bits 4-0=addr
        Call V.SPut(&H00815,&H00)'[15:8] bit15=CMD_FIN, bit13=Write bit12:8=PHYADDDR
        V.Wait(100)
        'GMII_MNGMT_DAT [15:0]
        Temp0 = V.SGet(&H00816)  '[7:0]
        Temp1 = V.SGet(&H00817)  '[15:8]
        Total = Temp0 + (Temp1*2^8)
        myhex = hex(Total)
        'V.Echo("BITS 15-9 are ABILITY registers")
        'V.Echo("BIT15=100BASE-T4,BIT14=100BASE-X-FD,BIT13=100BASE-X-HD), BIT12=10Mbps-FD")
          'V.Echo("BIT11=10Mbps-HD ,BIT10=100BASE-T2-FD,BIT9=100BASE-T2-HD,  BIT8=ExtendedSta-
tus")
                                  'V.Echo("BIT7=RESERVED,BIT6=MFPreambSupp,BIT5=Copper-AN-Com-
plete,BIT4=COPPER_REMOTE_FAULT")
         'V.Echo("BIT3=AN_ABILITY(1),BIT2=Copper_LINK_STATUS,BIT1=JABBER_DETECTED,BIT0=EXTEND-
EDREGISTERS")
        'V.Echo("=0x"& HexX(Total,4) &"")
        'V.Echo("")
        ANStatusReg=HexX(Total,4) '2 bytes
        'V.Echo("PAGE 0, REG 1 : Status Register-Copper [15:0]:= 0x" & ANStatusReg &"")
        ANcomplete= "&H" & ANStatusReg And &H0020 'bit 5
        'V.Echo("ANcomplete= 0x" & HexX(ANcomplete,4) &"")
        V.DO_EVENTS
        If V.shiftkey_pressed Then Exit Sub 'If shift Key pressed, then exit
    Loop

    V.DO_EVENTS

Loop

end sub

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