STANDARD
MICROSYSTEMS
CORPORATION

APPLICATION NOTE 6.1

ETX.EXE - ETHERNET DEBUG SOFTWARE

By Nestor Bersamira & Tony Lagana

ETX FEATURES

- EVALUATION SOFTWARE FOR LAN91C92, LAN91C94 & LAN91C100 ETHERNET CONTROLLERS
- LOW LEVEL DEBUG SOFTWARE, WORKS WITHOUT DRIVER AND NETWORK OPERATING SYSTEM
- REGISTER MAP DISPLAY AND EDITOR
- TEST UTILITIES
  - REGISTER READ/WRITE TEST
  - BUFFER READ/WRITE TEST
  - INTERNAL & EXTERNAL LOOPBACK
  - SERIAL EEPROM READ/WRITE TEST
  - MMU TEST
  - PONG (SEND PACKETS BACK AND FORTH)
  - SINGLE KEY TEST OF ETHERNET CARD WITH TESTS SELECTABLE FROM TEST MENU
- TRANSMIT & RECEIVE UTILITIES
  - SINGLE KEY TRANSMIT (AUTOMATICALLY ALLOCATES, WRITE TO PACKET BUFFER, TRANSMIT AND DISPLAY STATUS)
  - FULL CONTROL OF PACKET PARAMETERS
  - TRANSMIT FROM ONE PACKET TO CONTINUOUS LOOP
  - PACKET DISPLAY UTILITY
  - SINGLE KEY RELEASE OF PACKETS
- CONFIGURATION (SERIAL EEPROM) UTILITIES
- PCMCIA SUPPORT
  - INTEL PCIC ENABLER (OTHER CONTROLLER IN FUTURE REV)
  - READ/WRITE TO THE ROCKWELL & ATT MULTIFUNCTION CONTROLLER
  - PROGRAMS CARD INFORMATION STRUCTURE (CIS) FLASH RAM
- MII INTERFACE SUPPORT
  - READ/WRITE TO THE NATIONAL PHY CHIP
- FORCE COLLISIONS
- GENERATE RUNT PACKETS
- STORE AND READ PACKETS FROM EXTERNAL FILES

80 Arkay Drive
Hauppauge, NY 11788
(516) 435-6000
FAX (516) 273-3123

INTRODUCTION

The ETX.EXE evaluation software was specifically developed for the LAN91CXX family of Ethernet controllers. ETX is a standalone utility that requires neither a driver nor a network operating system (e.g. NETWARE). ETX is used specifically to evaluate the physical layer. This is an advantage to any system designer or test engineer who is in any stage of evaluation. The complexity of evaluation is reduced to a more manageable level since ETX allows the user to communicate directly to the Ethernet controller, thereby eliminating software driver and network operating system from the debug equation. In addition, if there are hardware difficulties, ETX has the capability to isolate the faults to the offending block. ETX has test utilities for all major blocks of the Ethernet controller.

ETX has the utilities necessary to simulate countless network scenarios. It is possible to simulate networks with varying complexities, from a two node network to a heavily loaded network complete with collisions and runts.

As for example, two nodes sending a packet back and forth simulates the simplest network. This is simulated with the PONG utility of ETX. By adding a third node, various scenarios become feasible. The third node can be configured as a traffic generator, resulting in collisions and deferments. The third node can also be configured as a runt generator, or as a node continuously forcing a collision. All this is easily done with ETX. Of course, adding more nodes, each running different functions, will generate a corresponding increase in the difficulties for the network.

ETX is easy to use. Most functions are one keystroke away. To run, just enter ETX at the DOS prompt. The software will search for the network controller, and will display the type of controller on the main menu. Once in the main menu, and with **minimal keystrokes**,
ETX can immediately :

- Transmit a single packet, or transmit in continuous loop
- Force collisions (minimum of two nodes) and transmit runt packets
- Receive continuously
- Send a packet between two node continuously while keeping track of error statistics
- Internal or external loopback
- Test functional blocks (register, packet buffer, EEPROM, mmu, interrupt)
- Test the network card

In addition to working with the Ethernet controller, ETX has utilities that **communicate with the following peripheral chips:**

- **Serial EEPROM** -The serial EEPROM contains the Ethernet controller's configuration, including the ID. There are utilities to display, test and program the configuration EEPROM. The programming utility is capable of programming Ethernet boards in full production since ETX can program sequential ID's.
- **PCMCIA controller** - There is an enabler for PCMCIA card applications that interfaces with INTEL PCIC chip. The enabler programs the INTEL PCIC to activate the PCMCIA card.
- **PCMCIA configuration controller** - Also for PCMCIA applications, ETX can communicate with Rockwell or ATT PCMCIA configuration controllers. As a consequence, the Card Information Structure (CIS) stored in a flash ROM can be accessed and programmed. The CIS is essential for a PCMCIA card to be fully PCMCIA compliant.
- **National PHY** - In addition to the configuration chips, ETX when running on LAN91C100 can communicate to the National PHY chip through the media independent interface (MII). This allows the LAN91C100 to fully exploit the capability of the National PHY.

RUNNING ETX.EXE

From the DOS prompt,

A:> ETX <optional base IO address>

Without the optional base IO address, ETX will search for the Ethernet controller. For PCMCIA applications that use the INTEL PCIC chip, ETX will use the **built in enabler** to power up the Ethernet card during the search.

With the base address specified, ETX will not do the search. There are several reasons for specifying the base IO address. For PCMCIA applications that use their own enabler or one that uses a different PCIC chip, ETX will forego the enabling function, leaving the previous enabler's setting intact. The other reason for entering the base IO address is if there is difficulty in reading the Ethernet controller's registers (due to bus problems, defective chips, or for a myriad of other reasons) which will result in ETX not finding the card. Entering an IO address will allow ETX to work with the Ethernet card if the search fails.

ETX identifies what controller is present. The controller type is displayed on top of the register map. The controller type determines test flows and the parameters used in the utilities. If a controller is not detected, the top of the register map will display 'NO CHIP FOUND'.

Notes on **moving around in ETX** :

- Generally, the arrow keys move the highlighted cursor bar. The exceptions are the SETUP and TEST menus, where the left and right arrow keys are used to toggle the parameters.
- When the cursor is on, ETX expects a hex or alphanumeric character.
- On the MAIN and SETUP menus, the characters in front of the items are hot keys. Pressing the hot keys will toggle the parameter, or run the utility.
- Pressing any key while looping on a test will terminate the loop
- Pressing ESC will always exit out of a menu.
- Pressing END in main menu will exit ETX.

MAIN SCREEN

The main screen is comprised of three portions:

Register map - The register map of the LAN91CXX controller is displayed at the top of the screen. The register map is updated after running a function or by pressing the spacebar. On top of the register map is the controller type.

Packet status - The status of transmit and receive packets, identified by the transmit done fifo and the top of the receive fifo of the fifo port register, are displayed in the middle of the main screen. The display lists the status, packet length, source and destination id. This screen is left blank if there are no transmit or receive packets. Any error in posted by the status word will cause an ERROR display to blink.

Main menu - At the bottom portion of the screen is the main menu. The main menu lists all the available functions and utilities. To select an item, press the hot key or arrow to the function and press CR. See the summary of all the utilities at the end of this note.

SOFTWARE RESET ('F1'-EPH RESET ,'e'-MMURST)

Pressing 'F1' does a **software reset** of the LAN91CXX by setting bit 16 (SOFT RST) of the RCR register for 100ms.
Pressing 'e' does an **mmu reset** by writing a reset mmu command to the MMU COMMAND register.

WRITING TO LAN91CXX REGISTERS ('F9')

This function allows the user to modify any of the LAN91CXX registers. Press 'F9' and a cursor bar appears on the register map, with the rest of the screen replaced by the **bit assignments of the highlighted register**. Move the cursor to select the register (the bit assignment will change with the register) and enter a hex word to write to the selected register. The register map is updated after the word is entered.

To illustrate, we can do the software reset by writing to the registers :

- From the main screen, press F9
- A cursor bar will appear on the register map and the bit definitions are displayed at the bottom
- Using the arrow keys, move to the TCR register
- Enter 8000 (SOFT RESET bit high)
- Register map is updated
- Enter 0000 (SOFT RESET bit low)
- Register map is updated
- Press ESC, and we're back to the main screen

MODIFY PARAMETERS ('F10')

Pressing 'F10' opens a menu that lets ETX modify packet parameters and LAN91CXX bit settings. The settings are used by the transmit, receive and loopback utilities. The following packet parameters are specified in this menu (shown with the default values in parenthesis) :

Packet size	(minimum packet size of 42h)
Source id	(node id)
Destination id	(node id)
Data type	(incrementing data)

There are **six data types** available :

INCR - incrementing data
AAAA - data are all a's
5555 - data are all five's
ZERO - data are all zeroes
ONES - data are all ones

FILE - Reads the packet to transmit from an external file. The packet size and id's will be updated with the data read from the file A sample packet is included in section READ AND WRITE A PACKET TO AN EXTERNAL FILE.

If the data type selected is not FILE, the packet size is written after the source ID. This is used by the receive loop to keep track of incorrect packet lengths transmitted.

To modify the packet size and the id's, enter the complete parameter. There is no provision to back space. If you entered incorrectly, press ESC and enter again. To toggle the other parameters, either use the hot keys or use the left or right arrow keys.

TRANSMIT A PACKET ('F5')

Pressing 'F5' will transmit a single packet. The packet size, destination id, source id, and data type are specified in the packet parameter menu (F10).

The function allocates a packet buffer, writes the data into the buffer, turns on the transmitter, queues the packet, then polls for completion. The parameters and status are then posted in the main screen. An error that occurs will display a flashing ERROR sign. The register map display is then updated. The packet is not released. The following transmit flowchart illustrates the register read/write needed for one transmit operation.

TRANSMIT FLOWCHART

// **turn on the transmitter**
write 0 to bank select register
set the txen bit of the tcr register
// **allocate a packet**

write 2 to bank select register
for LAN91C100 - write 20h (allocate) to the mmucommand register
for LAN91C92/94 - write 20h + (packet size/256) (allocate) to the mmucommand register
read the failed bit of the arrpnr register
read the high byte arrpnr register (allocated packet number)
write the allocated packet number to the low byte of the arrpnr register
// **write to the pointer register**
write 4000h to the pointer register (transmit,autoincrement,write,point to first word)
// **write to the packet buffer**
write 0 to the data register (written by controller with transmit status after transmit is completed)
write packet length to data register
write first word of the destination id to data register
write second word of the destination id to data register
write third word of the destination id to data register
write first word of the source id to data register
write second word of the source id to data register
write third word of the source id to data register
write the packet data to the data register
write the control word (last word)
// **queue the packet**
write 60h (enqueue) to the mmu command register
// **poll for the completion**
loop until the txint bit of the interrupt register is set (for a maximum of 100ms)
// **read the status word of the packet**
write 6000h to the pointer register (transmit, autoincrement, read, pointing to first word)
read the data register

TRANSMIT LOOP ('f')

The transmit loop will transmit the same packets continuously at a rate that can approach the maximum data rate. The data rate will vary with packet size and the number of packets allocated for transmit. The burst count specifies the number of packets allocated for transmit. Enter the burst count at the prompt. The maximum burst count is the maximum number of packets that can be allocated. The packets are queued while the transmitter is disabled. Once the transmitter is enabled, the packets are transmitted separated by the minimum interpacket gap.

The packet size, destination and source id's, and data type are all specified in the packet parameter menu. The packets contain a counting pattern, used in conjunction with the receive loop ('g') to **monitor dropped packets**. If the receiver is enabled during the transmit loop, the receiver will count the incoming packets and post receive errors. All receive packets are released.

The loop will terminate if a transmit packet does not set the TXINT bit of the INTERRUPT register within 100ms after the packet is enqueued.

RECEIVE

If the receiver is enabled (RXEN of the RCR register is set), any packets received will update the register map. The MIR register will count down since the received packet allocated packet memory, and the status of the packet at the top of the receiver FIFO is displayed. The receive status screen displays the packet number, packet size, status word, destination and source id's This display is on whenever a packet is in the receiver FIFO and will only clear if all receiver packets are released. The following receive flowchart illustrates the register read/write to handle a receive packet.

RECEIVE FLOWCHART

// **turn on the receiver**
write 0 to bank select register
set the rxen bit of the rcr register

// **loop until the a packet is received by polling the rxint bit of the interrupt register**
write 2 to bank select register
loop until the rxint of the interrupt register is set
// **the receive packet will be the top of the rx fifo if no previous packet was received**
//**write to pointer register to point to the first data to be read**
write to e0h to the pointer register (rcv,autoincrement,read
//**read the receive packet**
read the data register - status word
read the data register - packet length
read the data register - first word of the destination id
read the data register - second word of the destination id
read the data register - third word of the destination id
read the data register - first word of the source id
read the data register - second word of the source id
read the data register - third word of the source id
read the data register for (packet size-10h) - packet data
read the data register - control word

RECEIVE LOOP ('g')

Pressing 'g' will continuously receive packets. The status of the receive packet is displayed and the packet is then released. Used in conjunction with the transmit loop which transmit sequential packets, the receive loop checks to see if packets are being dropped, reported as 'packet not sequential', and also if the packet received has the correct packet size, reported as 'wrong byte count'. In order to properly work with transmit loop, the receiving node should be a faster machine since the receive loop is slower due to the packet buffer reads. The receiver parameters are set in the packet parameter menu of F10.

PONG ('m' MAIN, 'n' PARTNER NODE)

Pong sends a single or multiple packets back and forth between two nodes (main and partner nodes, except in loopback modes where main and partner is the same node) collecting error, defer and collision statistics. The RCV_BAD bit of the rcr register is set to receive bad packets. The main node initiates each loop by transmitting first, which the partner immediately re-transmits back to the main node. A packet that did not return to the main node within 55ms will increment the noping count (see pong statistic table). The statistics are updated after each loop. If multiple packets are chosen (burst count is greater than one), the packets are transmitted by the main node with minimum spacing, traffic allowing.

The packet size and data type are specified in the setup page of the main node. The id's do not have to be specified; the main node will automatically search for a partner node. Just make sure that the media type is the same on both nodes. During pong, the nodes are set to receive bad packets and the receivers are set to strip CRC.

To start pong, press 'n' on the partner node and then press 'm' on the main node. The main node will prompt you to enter the burst count. Enter the burst count or press 'm' if the count is correct. In the case where a loopback mode is selected (either EPH LOOP or FDUPLX bits of the TCR register is set), pressing 'm' will loopback continuously on one node, hence no partner node is required. Pressing any key will stop pong.

There is a special type of pong that sends different size packets, from 16h to the maximum packet that the chip can send. Press 'k' and pong will send different size packets. Pong will terminate after the maximum packet or if a failure has occured.

Pressing 'r' during pong will clear the statistics. The following statistics are kept during pong:

PONG STATISTICS

PARAMETER	COMMENT
txcount	Number of successful transmissions
rxcount	Number of packets received
noping	Main node did not receive the return packet within 55ms
dmaactive**	The internal dma machine is in an abnormal state
allocfail	Cannot allocate a packet for transmit
algnerr	Alignment error on the receive packet
lostcar	The transmitter lost carrier during the transmission
txunrn	The transmitter had an underrun error
badcrc	The packet was received with a CRC error
compfail *	The data received by the initiator did not match the one it sent
latcol	A collision occurred 64 bytes into the transmit packet
exctxdef	The transmitter deferred by more than 16 times
txdef	The transmitter deferred
mulcol	The transmitter had multiple collisions sending a packet
sglcol	A single collision occurred sending a packet
sqet	Failed signal quality test after a transmission
16col	16 or more collisions occurred during the last transmit
tooshort	The packet received is shorter than 64 bytes
toolong	The packet received is longer than 1518 bytes
rxoverrun	Receiver overrun count
wrbytecount	The receive packet has the wrong byte count

* valid on main node only

** valid for burst count of one

** dmaactive is incremented if the test register is in an abnormal state

LOOPBACK ('F6' EXTERNAL LOOPBACK ,'e' INTERNAL LOOPBACK)

Pressing 'F6' will externally (out of LAN91CXX) loopback a packet. Pressing 'e' will internally (the transmitted packet never leaves the LAN91CXX) loopback a packet. A software reset is issued before the loopback test. If the received and transmitted data are the same and no error bits are set, the loopback test passed, otherwise the test failed. If the loopback test passed, a green PASS bar is flashed on the screen while a loopback with error will flash a red FAIL bar.

For external loopback, the FDUPLX and LOOP bits of the TCR register are set. For internal loopback, the EPHLOOP bit of the TCR register bit is set. In addition, the destination id of the packet is the id of the LAN91CXX and the receiver is enabled. All the other packet parameters are set by the setup menu. The packets are not released.

DISPLAY TRANSMIT ('x') AND RECEIVE ('y') PACKETS

Pressing 'x' will display the packet specified by the PNR register. Pressing 'y' will display the packet specified by the TOP OF RX FIFO register. The packet is displayed 256 bytes at a time, and pressing the spacebar will display the next page. Pressing ESC or spacebar after the last page will return to the main menu.

You can easily switch to a different packet by entering a packet number on the upper left hand corner of the screen, pointed to by the cursor.

RELEASING TRANSMIT ('o') AND RECEIVE ('u') PACKETS

Pressing 'o' will release a transmit packet. Pressing 'u' will release the receive packet that is at the top of the receive FIFO register. The following flowcharts illustrate how to release a transmit and receive packets.

RELEASE A TRANSMIT PACKET FLOWCHART

//**write the transmit packet number into the arrpnr register**
write 2 to bank select register
write transmit packet into low byte of arrpnr register
//**issue release specific command**
write (release specific packet) a0h to mmu command register
// **release is complete if busy bit is cleared**
poll busy bit of the mmu command register
// **acknowledge the transmit interrupt**
write 2 to interrupt register

RELEASE A RECEIVE PACKET FROM TOP OF THE RX FIFO FLOWCHART

//**issue remove and release top of the rx fifo command**
write 2 to bank select register
write 80h (remove and release top of the rx fifo) to mmu command register
// **release is complete if busy bit is cleared**
poll busy bit of the mmu command register

CONFIGURATION EEPROM UTILITIES ('F4')

Press 'F4' and the contents of the configuration eeprom is displayed. There are three columns associated with each eeprom address: SW is the switch setting , ADDR is the EEPROM address and DATA. Each switch uses up two words of EEPROM; the even address is loaded to the configuration register while the odd address is loaded to the base register. What gets loaded to the configuration and base registers is dictated by the value of the IOS (switch) pins.

There are **three ways ETX can program the EEPROM** :

Program a single address - Move the cursor to the address and enter a hex word. The eeprom display is refreshed after the data is written.

The next two ways read the data from a file. See the table below for the file format. Press 'p' while in EEPROM memory map and ETX will program the eeprom with data read from a file (should be named ETX.CFG). The file contains the id and is updated after a board is programmed. This is meant to write sequential id's when programming multiple boards. The utility can program a single switch or all the switch settings:

Programming a single switch with data from ETX.CFG - Move the cursor bar to SWITCH and change to desired switch by pressing the left or right arrow. Modify any other parameters, and press 'p' then ENTER to program the EEPROM location corresponding to the switch. Here one switch is programmed with the base, configuration, and board id.

Programming all switches with data from ETX.CFG - Move the cursor bar to IOS and change setting to RESTORE ALL by pressing the left or right arrow. Modify parameters as needed, and press 'p' then ENTER to program the EEPROM. Here all seven switches are programmed simultaneously with values ETX.CFG.

The **eeprom default values** are the following. For all the default settings, rom is disabled, no wait state, and 16 bit operations is set.

EEPROM DEFAULT VALUES (no ETX.CFG file)			
SWITCH	**BASE ADDRESS**	**IRQ**	**MEDIA**
0	300h	INTR2	10BASET
1	280	INTR2	10BASET
2	320	INTR2	10BASET
3	340	INTR2	10BASET
4	360	INTR2	10BASET
5	300	INTR0	10BASET
6	300	INTR3	10BASET
7 (unused)			

The following is the ETX.CFG file format (default values shown, do not include any comments). ETX.CFG is a text file and can be edited with any text editor.

Note that the id can be incremented (or decremented) using the arrow keys. This is useful for programming in a production type environment where many boards are to be programmed. Once the ID is updated, write the data into ETX.CFG by pressing 'w'.

ETX.CFG FILE FORMAT	
DATA	**COMMENT**
10a4	written to specified switch (configuration) unless writing the entire eeprom, then data is written to switch 0
1800	written to specified switch (base) unless writing the entire eeprom, then the data is written to switch 0
8000	first word of the node id, written to ia10 register
6a0f	second word of node id, written to ia32 register
0000	third word of node id, written to ia54 register
10a4	written to switch 1 (configuration)
1900	written to switch 1 (base)
10a4	written to switch 2 (configuration)
1a00	written to switch 2 (base)
10a4	written to switch 3 (configuration)
1b00	written to switch 3 (base)
10a4	written to switch 4 (configuration)
1c00	written to switch 4 (base)
10a0	written to switch 5 (configuration)
1800	written to switch 5 (base)
10a6	written to switch 6 (configuration)
1800	written to switch 6 (base)

Testing the EEPROM

While in the EEPROM memory map screen, pressing 't' will test the EEPROM together with the various load and store functions of LAN91CXX. If the operations are successful, a green bar indicates a pass. Otherwise a red bar indicates a failure. The EEPROM contents are restored after testing.

PCMCIA ENABLER

When ETX is invoked without the optional IO base address, the program will search for the card. A part of the search is to detect the presence of the Intel PCIC chip. If the Intel chip is found, ETX will write to the PCIC registers to enable the Ethernet card. The card is configured for IO base 300h and INT3. The enabled configuration is retained when we return to DOS. At this point, the LAN91CXX card is enabled and other software (e.g. driver without an enabler and no Cards and Socket Services) can now communicate with the card.

PCMCIA CONFIGURATION CONTROLLER('3')

In PCMCIA dual function applications (Ethernet/modem), the **ATT and Motorola XILINX controllers** are used to control the Ethernet and modem function of the PCMCIA card. ETX automatically detects the controller used. Pressing '3' will display the register map of the PCMCIA controller (where the main menu was). To write to a register, use the arrow keys to select the register and enter the byte. The LAN91CXX and PCMCIA controller register maps are updated after the write.

PCMCIA CARD INFORMATION STRUCTURE ('1' DISPLAY , '2' WRITE)

To display the Card Information Structure flash ram, press '1'. To write to the CIS, press '2'. To write to the CIS, a file (entered after prompted by ETX) containing the CIS information is read and is written to the flash. An example of the file format is in appendix B. These menus will appear in the main menu only in PCMCIA applications.

COMMUNICATING WITH NATIONAL PHY THROUGH MII ('5')

This menu allows the Ethernet controller to communicate with the National Phy chip through the Media Independent Interface (MII) serial interface. To modify individually, use the arrow keys to cursor to the desired address location and enter the hex data word. To modify from an external file ETX.PHY, press 'l'. To store the PHY configuration into ETX.CFG, press 'w'.

There are **hot keys that can toggle often used parameters** :

z - reset	i - isolate
s - 10Base-T/100Base-TX	x - full duplex
l - loopback	n - auto negotiate

TEST UTILITIES

REGISTER TEST ('F2') - read/write test of the register set.

BUFFER TEST ('F3') - read/write test to the buffer memory. The test detects the type of controller under test and automatically configures the test to look at the entire packet memory.

EEPROM TEST ('F4' then 't') - Read/write test of the serial EEPROM. Also, the STORE and RELOAD commands, together with the EEPROM SELECT bit of the CONTROL register are tested.

LOOPBACK - see loopback

PONG - see pong

MMU TEST('F7') - Tests all the mmu commands. The allocate, reset mmu, remove receive packet, remove and release receive packet, release specific packet, enqueue packet, reset tx fifo, and the noop commands are all tested.

INTERRUPT TEST('F8') - Tests all sources of interrupts, uses INTR0.

The following tests are for very specific events and are **not included in the test board utility**:

TRANSMIT ERROR TEST('T') - Test for sqet error to generate an ephint. Test requires a MAU that has an SQET switch. Turn off SQET and set media type of AUI before running the test.

LINK TEST('L') - Test for link transition to generate ephint. To run, press 'L' then disconnect the cable.

COUNTER ROLLOVER TEST('C') - Test for ephint generated by counter rollover. This test requires another node generating continuous traffic or collision to increment the counters.

EARLY TRANSMIT TEST('c') - Tests the early transmit feature. To run, just press 'c'.

RECEIVE DISCARD('d') - Tests the receive discard feature. To run, press 'd'. ETX will wait for a packet from a second node to discard. Transmit a packet from a second node using the maximum packet size.

TEST BOARD('t') - This is a board level test. By pressing a single key 't', ETX will test the boards functionality in a single pass. ETX automatically chooses the test flow appropriate for the device. The test flows were designed to be compatible to the controller's evaluation boards. Each of the test flow requires a specific test setup, dictated by the required partner node for PONG. The table below lists the default test flow for each controller and the test setup.

DEFAULT TEST BOARD FLOW AND TEST SETUP

	LAN91C92 OR LAN91C94 ISA/VLBUS	LAN91C92 OR LAN91C94 PCMCIA	LAN91C100/ NATIONAL PHY CHIP	LAN91C100/ LAN BACON PHY CHIP/ LAN694
BUFFER TEST	ON	ON	ON	ON
REGISTER TEST	ON	ON	ON	ON
10BT INTERNAL LOOPBACK	OFF	ON	ON	ON
AUI INTERNAL LOOPBACK	OFF	ON	OFF	ON
10BT FDUPLX LOOPBACK	ON	OFF	ON	OFF
AUI FDUPLX LOOPBACK	ON	OFF	OFF	OFF
MII PONG*	OFF	OFF	ON	ON
MII 10BASE-T PONG*	OFF	OFF	OFF	ON
10BASE-T PONG*	ON	ON	ON	ON
AUI PONG*	ON	OFF	OFF	OFF
EEPROM TEST	ON	ON	ON	ON
MMU TEST	ON	ON	ON	ON
INTR0 TEST	ON	ON	ON	ON
TEST SETUP	SETUP #1	SETUP #2	SETUP #3	SETUP #4

*Pong tests require a partner node running PING 'n'.

Setup #1
Eval Board's AUI and 10Base-T ports connected to a 10Base-T concentrator. The partner node is connected to the concentrator.

Setup #2
Eval Board's 10Base-T port connected to a 10Base-T concentrator. The partner node is connected to the concentrator.
-or-
Eval Board's 10Base-T port connected to partner node's 10Base-T port with a crossover cable.

Setup #3
Eval Board's 10Base-T/100Base-TX port connected to a 100Base-TX concentrator. The 100Base-TX partner is connected to the 100Base-TX concentrator with a CAT5 cable.
-or-
Eval Board's 10Base-T/100Base-TX port connected to a 100Base-TX partner with a CAT5 crossover cable.

Setup #4
Eval Board's 10Base-T port connected to a 10Base-T concentrator and the 100Base-TX connected to a 100Base-TX concentrator. Two partner nodes required, one as a 100Base-TX partner and the second as a 10Base-T partner each connected to the appropriate concentrator.
-or-
Eval Board's 10Base-T port connected 10Base-T partner with a crossover cable, and the 100Base-TX port connected to a partner with a CAT5 crossover cable.

The test flow is modified by selecting the **h-TEST MENU** utility. The test menu allows the user to toggle ON or OFF individual tests, or in one fell swoop by loading the test menu file ETX.MEN by pressing 'l'. Once the menu is modified, pressing 'w' will modify ETX.MEN to reflect the changes.

READ AND WRITING PACKET TO AN EXTERNAL FILE

To **read a packet from a file**, go to the SETUP page and move the cursor to TX DATA TYPE. Press the left or right arrow until TX PACKET FILE appears. When this happens, enter the packet file and the corresponding packet size and id's will be displayed if the packet is successfully read. Refer to the sample packet below for the packet format, or save a packet to use as a reference.

To **save a packet to a file, press 'i'** and you will be prompted for the packet number you wish to save and the file name. Enter the packet number and the file name.

PACKET FILE FORMAT
(LAN91CXX packet length of 42h)

	pkt length	destination id			source id		
0000	0042	8000	660f	deaf	8000	660f	face
FFFF	0001	0203	0405	0607	0809	0a0b	0c0d
0e0f	1011	1213	1415	1617	1819	1a1b	1c1d
1e1f	2021	2223	2425	2627	2829	2a2b	2c2d
4000							

First data 0000h (before packet length) is over written with the status word by the LAN91CXX.
Data is from FFFF to 2c2d.
Control word is 4000h at the end of the packet.
The shaded portion plus two CRC words are transmitted on the cable.

MULTIPLE BOARDS ('b')

ETX can communicate to multiple networking cards in the same PC, assuming of course that each network card has a different base address. To move from one card to another, press 'b' and enter the base address of the networking card.

FORCE A COLLISION ('j')

The force collision feature allows the controller to collide with incoming packets. The controller detects the CRS (CRS100 for FEAST MII) for incoming packets. Once CRS is valid, the controller sends out a colliding packet. After the collision, the controller rearms for the next CRS. Press any key to end collision generation.

GENERATE RUNT PACKETS ('6')

A runt is a short (below minimum packet size) packet that does not contain a valid CRC. The runt generator continuously transmit runts of various lengths, from six to 64 bytes. The delay between runts is specified by the user in units of ms.

WRITING TO EXTERNAL REGISTERS ('v')

The external registers are read and written to with this utility. The external registers are accessed by reading and writing the first registers of bank seven.

SUMMARY OF ETX FUNCTIONS	
KEY	**FUNCTION**
END	Exit ETX
F1	Software reset
F2	Register read/write test
F3	Packet buffer read/write test
F4	Configuration eeprom read/write test
F5	Single packet transmit
F6	External loopback test
F7	MMU test
F8	Interrupt test, uses INTR0
F9	Modify Ethernet controller's registers
F10	Setup page, modify packet and controller parameters
1	Display attribute ram (only for PCMCIA applications)
2	Write attribute ram (only in PCMCIA applications)
3	Write to configuration register (only in PCMCIA applications)
4	Read/write to external registers
5	Read/write to the national phy
6	Generate runt packets
a	Allocate transmit packet
b	Allows etx to communicate to multiple boards
c	Early transmit test
C	Cr enable test
d	Receive discard
e	Internal loopback test
f	Transmit loop
g	Receive loop
h	Test menu
i	Write buffer into a file
j	Force collision
k	Pong from packet length of 16 to maximum size0
l	Reload configuration from serial eeprom
L	Le enable test
m	Press on primary node to start pong test
n	Press on partner node for pong test
o	Release tx packet
p	Software powerdown
q	Enqueue tx packet
r	MMU reset
s	Store configuration to serial eeprom
t	Test network card
T	Te enable test
u	Release rx packet
v	Read/write to the external registers
w	Write packet to packet memory
x	Display tx packet
y	Display rx packet
z	Write zero to all of packet buffer