PM7375

LASAR-155 SOFTWARE USER MANUAL

Preliminary Information

Issue 1: March, 1996

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Scope

This document describes the evaluation software for the LASAR-155 Evaluation Platform. The software is designed to exercise features of a LASAR-155 adapter card on a PCI bus host running the Windows 3.1X operating system. It provides interactive controls for examining features of a LASAR-155 device and a script command window for automated test.

1.2 Audience

This document is provided for users of the LASAR-155 Evaluation Platform. The reader should have basic familiarity on use of Windows 3.1X software and typically would be an applications engineer, design engineer or product test engineer.

1.3 Objective

The goal of the user manual is to:

- describe the components of the graphical user interface (GUI)
- provide a reference for users wishing to write a script command file
- describe the various file formats
- describe how to install the software
- describe the syntax of the log file

1.4 References

[1] PMC-931127 LOCAL ATM Segmentation and Reassembly & Physical Layer Interface, Issue 3, March 1996.

[2] PCI Special Interest Group, PCI BIOS Specification, August 26, 1994, version 2.1.

[3] PCI Special Interest Group, PCI Local Bus Specification, June 1, 1995, version 2.1.

2 SOFTWARE INSTALLATION

2.1 Platform Requirements

The LASAR-155 adapter card must be inserted into a PCI slot of a PC compatible computer. The minimum requirements of the PC computer are:

- 8 Mbytes of RAM
- Windows 3.1 (3.11) operating system
- 486 CPU
- Spare PCI slot in motherboard to accept the LASAR-155 adapter card
- 2 MBytes hard disk space

The recommended requirements of a PC computer, such that the sample script files may be run is:

- 16 Mbytes of RAM
- Windows 3.1 (3.11) operating system
- Pentium CPU
- Spare PCI slot in motherboard to accept the LASAR-155 adapter card
- 2 MBytes hard disk space

2.2 Installation Procedure

The following steps are required before running the software.

1) Create a directory, `C:\PMC\LASAR155` to store the software files.

1) Copy the zip file `LASAR155.ZIP` to the directory created in the previous step and run pkunzip. All the components in table 1 should exist. The `README.TXT` file should be read, as it contains the latest information.

2) Copy all files to the directory created in the previous step.

3) Move the file BWCC.DLL to the `C:\WINDOWS\SYSTEM` directory. Ensure the file is no longer in the `C:\PMC\LASAR155` directory by deleting it if necessary.

4) Check the `C:\WINDOWS\SYSTEM` directory for the file, `CTL3DV2.DLL`. If this file does not already exist then move this file from C:\PMC\LASAR155 directory into

the C:\WINDOWS\SYSTEM directory. If the file exists then ensure the file is removed from the C:\PMC\LASAR155 directory by deleting it if necessary.

5) Edit the file C:\WINDOWS\SYSTEM.INI file and add the following lines below the [386Enh] section (keywords are case sensitive):

```
device=c:\pmc\lasar155\ule.386
LasarLockedPages=500h
LasarMinimumPage=800h
LasarPollTimeout=400h
```

6) Exit to DOS and then run Windows, thereby installing and initializing the virtual device driver, ULE.386. If the initialization failed an error message file is automatically created in the file C:\LASAR.TXT. The "LasarLockedPages" or "LasarMinimumPage" keyword may need to be modified depending on the RAM configuration of the PC. The assigned values are in multiples of 4K Byte pages (ie 100h is equivalent to 1 Mbyte).

7) Erase the LASAR.TXT file, exit to DOS and re-run Windows after modifying the keywords if the initialization failed.

8) Create a program item within Windows for the executable, C:\PMC\LASAR155\LASAR.EXE.

9) Run the executable file by double clicking on the ICON just created.

Table 1. List of Software Components

Name	Description
ULE.386	The driver executable file.
LASAR.EXE	The application executable file.
BWCC.DLL	Dynamic link library for the application.
BC450RTL.DLL	Dynamic link library for the application.
BIDS45.DLL	Dynamic link library for the application.
OWL250.DLL	Dynamic link library for the application.
CTL3DV2.DLL	Optional dynamic link library. It must be installed if not already present within Windows system directory.
LR.TXT	Text file listing LASAR-155 registers which are used in the list box control of the LASAR register display.
README.TXT	Latest information on installation and support of the evaluation software.
*.RUN	Sample script command files.

2.3 Optional Installation Steps

The following optional keywords may be added to the Windows "system.ini" file:

- LasarTestmodeIrqLine
- LasarPipeSize
- LasarResetTimeout
- LasarOpenTimeout
- LasarVcTimeout

Each of these is described in table 2. If a keyword is not specified the default value is assumed by the software driver.

Table 2. List of Driver Keywords

Keyword	Default Value	Description
LasarOpenTimeout	0	This keyword specifies the software timeout for opening the adapter. The timeout is specified in units of milliseconds. The open timeout is the time period in which the driver waits to complete any asynchronous tasks outlined in the **open_adapter** script command.
LasarResetTimeout	1	This keyword specifies the software timeout for reset of the adapter. The timeout is specified in units of milliseconds. The reset timeout is the time period in which the driver waits to complete any asynchronous tasks outlined in the **reset_adapter** script command.
LasarCloseTimeout	0	This keyword specifies the timeout for closing the adapter. The timeout is specified in units of milliseconds. The close timeout is the time period in which the driver waits to complete any asynchronous tasks outlined in the **close_adapter** script command.
LasarPollTimeout	0	This keyword specifies the timeout for polling the adapter. The timeout is specified in units of milliseconds, and a value of zero indicates polling is disabled. The poll timeout is the time period in which the driver waits between polling the performance and error counters.

LasarVcTimeout	1	This keyword specifies the timeout for completing a COPS table access. The timeout is specified in units of milliseconds. The VC timeout is the time period in which the driver waits to complete any asynchronous tasks outlined in the **cops_access** script command.
LasarTestmodeIrqLine	0	This keyword specifies the Interrupt line on the host motherboard which is assigned to the PCI bus. This value may be different among manufacturer's motherboards. The reader should run "msd" at the DOS command line and interogate the list of interrupt lines to find the IRQ which is cascaded to IRQ2 of the master programmable interrupt controller (PIC). This keyword must be assigned in order to run the software in test mode.
LasarPipeSize	0x10000	This keyword specifies the size of the FIFO, in bytes, which is used to store indication and asynchronous confirm messages. These messages are written by the driver and read by the application software.
LasarLockedPages	No Default Value	This keyword specifies the number of pages within host RAM which is locked by the driver. This memory is used to hold the LASAR-155 data structures, transmit data and receive data. This value must be specified. One page is 4KBytes.

LasarMinimumPage	No Default Value	This keyword is used to assign the physical start address of the locked RAM specified by the LasarLockedPages keyword. The operating system will attempt to lock RAM for the LASAR-155 at the page number specified. If the page number is already assigned to another device, or the operating system, the next highest "free" page will be assigned. If this keyword is unassigned the lowest available block of RAM will be locked. One page is 4KBytes.

3 SYSTEM OVERVIEW

3.1 Typical Evaluation System

A typical system in which the software will run is shown in figure 1. The executable files are located on the hard disk and loaded into RAM by the host operating system. The host motherboard must be running the Windows 3.1 (or 3.11) operating system and must be equipped with a PCI bus slot for insertion of the LASAR-155 adapter card. In a typical system the interrupt pins of all PCI bus slots are routed individually to the PCI interface/bridge where they are multiplexed onto a single interrupt line of the host's Programmable Interrupt Controller. For this reason the interrupt service routines of multiple LASAR-155 adapter cards must share the same interrupt line.

Figure 1. Simplified Block Drawing of a Typical PC System

3.2 Software Driver Features

The features of the software driver are as follows:

- The driver manages the adapter via the LASAR-155 register space and the interrupt line assigned to the PCI bus. The driver is capable of simultaneously managing a number of LASAR-155 devices on the PCI bus.
- The driver maintains LASAR-155 queues, buffers and descriptors in host RAM.

- The driver provides services to the application software. The services include transmit of RAM data via transmit queues, provisioning/unprovisioning of VC's, access to receive data in RAM, access to RAM data structures, access to the LASAR-155 registers, indication of LASAR-155 interrupts and periodic indication of counter values and alarm states.

- The driver runs in a number of modes, some of which may be combined. The RX_LOOPBACK mode cannot be combined with the TX_BUFFERED, TX_BUFFERED_REPEAT_TABLE or TX_BUFFERED_REPEAT_QUEUE modes.

  - RX_LOOPBACK mode places receive data into the low priority transmit queue, effectively causing a loopback of receive data across the PCI bus.

  - RX_CAPTURE mode captures receive data references into the driver's capture queue. Management references and packet references are distinguished by a value written to an unused field of the descriptor.

  - RX_FILTER mode only acts on data of one receive channel, ignoring all other data. Descriptors of ignored data are immediately returned to the free queue and there is no receive indication associated with the ignored data.

  - TX_BUFFERED mode causes a number of transmit packets to be queued within a stimulus table of the driver. A timeout between successive transmit packets can be specified. The buffered transmit requests are queued within the LASAR-155 transmit queues when the **startd** script command is issued.

  - TX_BUFFERED_REPEAT_TABLE mode acts as the TX_BUFFERED mode except that the buffered transmit requests continue after the end of the stimulus table is reached. This is accomplished by repeating the transmit requests from the beginning of the stimulus table. In this mode, the current packet in the stimulus table, and all packets following the current packet, will not be queued on a transmit queue until the current packet is available for re-transmission. Therefore, packets destined for lower bandwidth VC's may hold up transmission of packets destined for higher bandwidth VC's.

  - TX_BUFFERED_REPEAT_QUEUE mode acts as the TX_BUFFERED mode except that once the LASAR-155 completes transmission of a packet, and places the packet on the free queue, the driver re-queues the packet on the low priority ready queue. In this mode, packets destined for lower bandwidth VC's will not hold up transmission of packets destined for higher bandwidth VC's. Each provisioned VC can be continuously transmitting data.

4 THE GRAPHICAL USER INTERFACE

The graphical user interface (GUI) is a Windows application that lets the user run script files, launch displays, and view event logs. The Windows application communicates with the software driver via the Windows API. Processing of hardware events is done by the driver at a higher priority than any Windows applications. Thus notification of hardware events is presented to the GUI some time after the driver has processed events and when the driver no longer requires CPU time.

All the displays of the GUI accept the usual Windows editor commands via the keyboard; The keyboard combinations of CTRL-Z, CTRL-X, CTRL-C and CTRL-V, can be used to undo, cut, copy and paste portions of text highlighted by the mouse. This text is held in the clipboard and can be placed in any other Windows edit control.

4.1 Main Window

The main window of the interface is illustrated in Figure 2. It encloses a script file window and a log file window, neither of which are shown at startup. The user selects **Session | Load Script** from the main menu before running a script command file. Successful loading of the script file automatically creates the script file window and the log file window.

The **Session | Test Mode** menu item allows the software to run with or without a LASAR-155 adapter card present. The test mode of the LASAR-155 Device Driver replaces the current PCI configuration and register spaces with RAM addresses allocated by the software. This allows the user to run the driver when there is no PCI device present. In this mode the user can still use the interactive displays to modify the "emulated" LASAR-155 registers.

The **Session | Invoke Isr** menu item can be used to execute the driver's interrupt service routine, whether or not there is an adapter card present. This command invokes the interrupt service routine without the LASAR-155 interrupt pin being active.

Toggling the **Session | Test Mode** menu item may reset and remove registry of a LASAR-155 adapter that has previously been added via the **add_adapter** script command, even though the adapter card was added when test mode was inactive.

Once a script file has been loaded the various menu items are enabled for the user. The following sections describe the functionality and menu items associated with the various displays.

The **Window | Tile** menu item aligns the log and script windows within the main window if they had been moved.

The **Window | Exit** menu item allows the user to exit from the application. If the log file window and script file windows are open, and have not been saved, the user is

prompted to save these files before the application exits. Exiting from the application does not unload the driver software. The driver is only unloaded when exiting from the Windows operating system. In most circumstances, exiting from the application may reset the software driver and any adapter cards which have been registered with the driver via the **add_adapter** command.

Figure 2. Evaluation Software Main Window

4.2 Script File Window

The script file is a textual file that can be created using a text editor such as notepad. The script file can also be edited within the script file window after it is loaded (or created) using the **Session | Load Script** menu item. Select the **Script Control | Save** menu item to save changes to the script file.

The script file commands are described in section 5. A typical script file may have comments, as identified by the asterisk character ('*') in the first column. The last command of the script file must be **end** and each line must have no more than one command.

The **Script Control | Reset Script** menu item positions the window cursor to run from the beginning of the script. The **Script Control | Run** menu item will execute script

commands following the currently highlighted line. The currently highlighted line may be seen by choosing the menu item **Script Control | Hilight Last Line**.

The script file window can be used to run commands as an interpreter. The user may move the mouse to click on the script file window and then type commands directly in the window. The main menu is used to run these commands.

4.3 Log File Window

The log file window is created when a script file is loaded via the **Session | Load Script** menu command. The log file window displays the script file commands and the driver events as they occur.

On loading a script file the log file window shows the physical base address and number of bytes of host RAM which have been reserved for use by the driver and the LASAR-155 hardware. These two values specify the block of contiguous physical memory which is available to assign the LASAR-155 queues, descriptor tables and data buffers.

The **Log Control | Clear** menu item is available to clear all text from the log file window. Alternatively the **Log Control | Save** menu item is available to copy the text within the window to the log file. If the log file exists the text is appended to the end of the log file. When the log file does not exist the file is created and the text is written to the empty file. The log file window is cleared after saving the text to the log file. The log file is named with the same prefix as the script file and the suffix is "log".

It is important that text in the log window be cleared before it's length exceeds 32K characters (the number of characters is shown at the bottom of the main window). When the log window is full, the speaker will "beep" repeatedly (This is a limitation of an edit control in Windows 3.1X). Performance of the GUI is indeterminate after this occurs, but manually clearing the log file window by selecting **Log Control | Clear** may restore the performance of the GUI.

The log file window only accepts the CTRL-X or CTRL-C keystrokes to cut, or copy portions of text highlighted by the mouse.

4.4 Alarm Status Display

When **Displays | Alarm Status** is selected from the main menu a dialog box appears in which the adapter card index must be entered. After responding to this dialog box the alarm status display is created for the specified LASAR-155 adapter card.

Figure 3. Alarm Status Display

The alarm status display shows the alarms as either red, green or grey. The grey color indicates that the alarm has not been updated. The red color indicates that the alarm is active. The green color indicates that the alarm is inactive.

The row of alarms, labeled as current, indicates the state of the LASAR-155 hardware as of the last LASAR-155 interrupt. The row of alarms, labeled history, indicates whether any alarms were active. The history can be deleted by choosing the delete history button.

The alarm status display is interrupt driven and will not be updated if the relevent LASAR-155 interrupts are disabled or the motherboard interrupt line is masked (refer to the **open_adapter** script command in section 5.2).

4.5 Performance Counter Display

The performance counter display shows the LASAR-155 counters. The counters are polled at the default rate specified by the LasarPollTimeout keyword of the "system.ini" file. A value of zero disables polling. Alternatively, the poll timeout can be specified via the **setup_driver** script command. The default value of poll timeout is zero, unless overriden via a keyword in the "system.ini" file.

Figure 4. Performance Counter Display

The display accumulates the LASAR-155 count registers and may be reset by choosing the clear button. Each count is stored in a 32-bit variable and when a counter reaches 0xFFFFFFFF the text, OVERFLOW, is shown instead of the count. The bottom of the display shows the number of updates to the display as a result of poll timeouts within the driver.

4.6 Host RAM Display

Selecting **Displays | Host Ram** from the main menu launches the host RAM display. This display can be used to inspect or modify the data which resides in locked pages of RAM. The block of RAM that may be accessed is specified by the appropriate keywords within the "system.ini" file when Windows is started (see the installation procedure in section 2). The top row of this display specifies the block of RAM that was locked by the driver.

Figure 5. Host RAM Display

The display shows the data in DWORD alignment (ie. BYTE3,BYTE2, BYTE1, BYTE0 BYTE7, BYTE6, BYTE5, BYTE4 etc.). The address of the data is specified in the left hand column as an offset from the physical base address of the RAM block locked by the driver. The **DISPLAY** button may be selected to view the data at another address. For pre-defined locations of RAM, the address can be added to a list by selecting "Add address to listbox" or "Add address and display it" in the RAM address display, which is created when the **DISPLAY** button is pressed. The address will then be displayed in the listbox and added to the text file "ram.txt". The same address may be inspected at another time by simply using the mouse to select the textual description from the list box within the host RAM display.

The data shown within this display can be modified by clicking the mouse over the data and entering a new value. The **Commit** button can then be used to write the data from the display to RAM. The **UPDATE** button can be used to read the data from RAM and refresh the display. Alternatively, the **Data In** and the **Data Out** buttons can be used to transfer data between RAM and a file (see section 6 for the file formats supported).

The **RATE** button may be selected to read from RAM and refresh the display periodically. The rate is entered as a digit, in multiples of 500 msec.

The scroll bar is used to scroll through the RAM. Using the mouse to click on the arrow scrolls up or down by the display size. Clicking on the scroll bar scrolls up or down by one page (4Kbytes).

The **CANCEL** button closes the RAM display.

4.7 LASAR-155 Register Display

This display is used to access the register space of a LASAR-155 adapter card. When **Displays | LASAR Registers** is selected from the main menu a dialog box prompts the user for the index of the n'th LASAR-155 adapter card on the PCI bus. Valid indexes are 0, 1 or 2. Specifying an invalid index or cancel will default to an index of 0. The Lasar register display is created after the index is specified and a button on the dialog box is pressed. The display will be created only if the **add_adapter** script command has been successfully issued via the script file window.

Figure 6. LASAR-155 Register Display

The behaviour of the display is the same as for the host RAM display, except that only one 32-bit register is displayed at a time. Also, there is a pre-defined list of registers that may be selected. The list is read from the "lr.txt" file that was installed during the installation. The "lr.txt" file is read from the current working directory, so the user should ensure that the file is installed in the working directory.

The **CANCEL** button closes the LASAR register display.

4.8 Driver LCB Display

This display is used to inspect the LASAR control block of a LASAR-155 adapter card. When **Displays | Driver LCB** is selected from the main menu a dialog box prompts the user for the index of the n'th LASAR-155 adapter card on the PCI bus. Valid indexes are 0, 1 or 2. Specifying an invalid index or cancel will default to an index of 0. The driver LCB display is created after the index is specified and a button on the dialog box is pressed. This display is used just as the host RAM display except that the list of addresses for the list box is stored in the file "lcb.txt".

Figure 7. Driver LCB Display

The **CANCEL** button closes the driver LCB display.

4.9 Driver Stimulus Table Display

This display is used to inspect the stimulus table associated with a LASAR-155 adapter card. When **Displays | Driver Stimulus Table** is selected from the main menu a dialog box prompts the user for the index of the n'th LASAR-155 adapter card on the PCI bus. Valid indexes are 0, 1 or 2. Specifying an invalid index or cancel will default to an index of 0. The driver stimulus table display is created after the index is specified and a button on the dialog box is pressed. The use of this display is identical to the host RAM display, except that the list of addresses for the list box is specified in the file "st.txt".

Figure 8. Driver Stimulus Table Display

The stimulus table display can be edited just as for the host RAM display. Each row within the display represents one entry of the stimulus table. The format of a row is:

- DWORD0 is the transmit packet reference
- DWORD1 is 1 if the packet is destined for the high priority queue, 0 for the low priority queue.
- DWORD2 is the number of milliseconds to wait before queueing the next transmit packet reference.
- DWORD3 is reserved.

The **CANCEL** button closes the driver stimulus table display.

4.10 Driver Adapter Table

This display is used to inspect the adapter table of a LASAR-155 adapter card. When **Displays | Driver Adapter Table** is selected from the main menu a dialog box prompts the user for the index of the n'th LASAR-155 adapter card on the PCI bus. Valid indexes are 0, 1 or 2. Specifying an invalid index or cancel will default to an index of 0. The adapter table display is created after the index is specified and a

button on the dialog box is pressed. The use of this display is identical to the host RAM display, except that the list of addresses for the list box is stored in the file "at.txt".

Figure 9. Driver Adapter Table Display

Useful fields within the adapter table are the *poll count* and the *ISR count* as described below:

- DWORD1 is the *ISR count*. Each time the interrupt service routine is run this value is incremented.
- DWORD2 is the *poll count*. Each time a poll timeout occurs the performance counters are read and this value is incremented.

The **CANCEL** button closes the driver adapter table display.

4.11 Automated Jitter Test Display

When **Displays | Automated Jitter Test** is selected from the main menu the user is prompted to enter the index of the LASAR-155 adapter card. The automated jitter test display is created on entering an index and pressing a button on the dialog box.

Figure 10. Automated Jitter Test Display

The automated jitter test is useful for performing jitter tolerance and jitter transfer tests from an external controlling PC. The external PC would run a program that communicates with HPIB instrumentation such as an SJ300 jitter analyzer. The controlling PC communicates with the PC under test via the serial port and a null modem cable attaching the two PC's. The controlling PC asserts the following commands to the PC holding the LASAR-155 adapter, to periodically query the LASAR-155 registers which contain the section BIP-8 count.

- "C000 00 WR" causes the software to write 0 to LASAR-155 register 0 and send the command string back to the controlling PC.
- "C000 80 WR" causes the software to write 0x80 to LASAR-155 register 0 and send the command string back to the controlling PC.
- "C012 00 WR" causes the software to write 00 to LASAR-155 register 0x12 and send the command string back to the controlling PC.
- "C012 RD" causes the software to read from LASAR-155 register 0x12, convert the register value into text, concatenate the text value to the command string, and then send the string to the controlling PC.
- "C013 RD" causes the software to read from LASAR-155 register 0x13, convert the register value into text, concatenate the text value to the command string, and then send the string to the controlling PC.

The user must press the **Start** button of the display in order to specify which serial port is activated. A digit, representing COM1, COM2, or COM3 should be entered. The display is updated with the section BIP-8 count when the "C013 RD" command is received and processed . The accumulated count is cleared after the **Start** button is pressed and the serial port is activated or re-activated.

The **CANCEL** button closes the automated jitter test display and disables processing of serial port commands.

4.12 RAM Address Display

This display prompts the user for a memory address whenever a user presses the **DISPLAY** button within one of the following displays:

- host RAM display
- LASAR register display
- driver LCB display
- driver adapter table display
- driver stimulus table display

The **RAM Address** field is a combination of as many as eight of the characters, 0 thru 9, and A thru F (without case sensitivity). The address is a 32-bit value.

Figure 11. RAM Address Display

The user must click the appropriate address type. The address type is a physical address, a virtual address or an offset from a base address. A physical address specifies the hardware address as seen by the LASAR-155 on the PCI bus. A virtual address is the address used internally to the motherboards CPU and by the driver software to access RAM. An offset address is a byte offset into the contiguous memory accessible by one of the displays listed above.

This display also allows the option of labeling the address with a textual description and permanently storing it in the list box. This allows for easy retrieval of data without pressing the **DISPLAY** button to navigate through memory addresses.

4.13 Transfer Data To/From File System Display

This display is created whenever the **Data In** or the **Data Out** button is pressed within one of the following displays:

- host RAM display
- LASAR register display

- driver LCB display
- driver adapter table display
- driver stimulus table display

The DOS file name and file path must be specified. The file name must be either a binary data file, a "*.b" data file, or a "*.d" data file. Each file type is explained in section 6.

Figure 12. Transfer Data Display

Another user input to this display is a hexadecimal RAM address and an address type. The hexadecimal address is a combination of upto eight of the characters, 0 thru 9, and A thru F (with case insensitivity). The address is a 32-bit value representing a byte location in RAM. The *Size* field specifies the number of bytes following the address which can be written into RAM (or read from RAM). The data files will not overwrite RAM beyond the number of bytes specified in this field, regardless of the contents of the data file.

The address type is a physical address, a virtual address or an offset from a base address. A physical address specifies the hardware address on the PCI bus. A virtual address is the address used internally to the motherboard CPU and by the driver software to access RAM. An offset address is an offset into the contiguous memory accessible by one of the displays listed above.

The *Start Index* field is optional. It specifies the byte offset within a binary data file from which data is read or written. This field is only required when specifying a binary data file.

5 COMMAND DESCRIPTION

5.1 Understanding the Command Syntax

The commands are text characters which can be entered via a DOS or Windows editor such as notepad, or directly in the script file window. Each command must be delimited by the carriage return and line feed characters. These characters are inserted by typing the return key and are not visible within an editor.

Blank lines are valid and are ignored by the command parser.

Comments may be inserted into a script file by placing the asterisk character in the first column of a line which has comments.

Fields of a command are delimited by an arbitrary number of blank characters (spaces) or tab characters. The fields of a command must be entered in the order specified and as described below:

- A field with no brackets implies that the text is to be typed exactly as shown.
- <> brackets around a field implies that the textual representation of a number must be entered.
- {} brackets around a field, or fields, implies that the field(s) are optional.
- | between two fields implies that there is a choice between two or more options. The set of options are enclosed with () brackets.
- [] brackets around a field implies a sequence of fields represented by the descriptive text between the brackets.

5.2 Command List

5.2.1 Script and Log Control

The following commands act on the script file, or the log file. They do not make calls to the driver or access the LASAR-155 adapter card.

- compare
- end
- pause
- register_value
- savelog
- to_physical
- to_relative
- wait
- td_template

5.2.2 Software Driver Control

The following commands control the driver. These may also be listed as accessing the LASAR-155 registers, where the command involves control of the driver and access to LASAR-155 registers.

- add_adapter
- capture
- close_adapter
- filter
- mask
- open_adapter
- reset_adapter
- remove_adapter

- setup_driver
- startd
- stopd
- startq
- stopq
- timeout
- transmit

5.2.3 Access to LASAR-155 Normal Mode Registers

The following commands involve a read and/or write access to a LASAR-155 register.

- close_adapter
- cops_access
- init_ram
- open_adapter
- read
- reset_adapter
- startd
- stopd
- transmit
- write

5.2.4 Access to the LASAR-155 Configuration Registers

The following commands involve a read and/or write access to a LASAR-155 configuration register.

- add_adapter
- locate
- read

- write

5.2.5 Access to the Locked RAM

The following commands involve read and/or write access to the block of RAM locked by the driver. This RAM is accessed by the LASAR-155 and holds the descriptors, references, queues and buffers associated with transmit and receive packets.

- create_packet
- dump
- init_ram
- load
- load_data
- rm_free
- rp_free_large
- rp_free_small

5.2.6 PCI BIOS Function Calls

The following commands involve a PCI BIOS function call. The function call may access a LASAR-155 configuration register, and for this reason the commands may also be listed as accessing the LASAR-155 configuration registers.

- locate
- pciinfo
- read
- write

5.3 Script File Commands

The script file commands are described in the following pages. The commands are listed alphabetically.

add_adapter

This command causes the driver to locate a PCI adapter which has the LASAR-155 vendor and device identifiers within its configuration space. It then initializes structures within the driver that enable further commands to be handled.

The driver issues a find_pci_device function call to the motherboard's PCI BIOS as specified in reference [2]. The driver then accesses the LASAR-155 configuration registers via the PCI BIOS. The following LASAR-155 registers are read:

- LASAR-155 Memory Base Address Register (DWORD at 0x10)
- External Device Memory Base Address Register (DWORD at 0x14)
- Expansion ROM Base Address Register (DWORD at 0x30)
- INT_LINE (BYTE at 0x3C)

Syntax:

add_adapter <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

The LASAR-155 adapter should be inserted in the PCI slot, or the session should be run in test mode, for this command to be successful.

Logged Return Values:

None

Example:

add_adapter 0

capture

This command configures the buffering of receive descriptor references within the driver's capture queue. This command specifies whether a reference should be written over with the latest receive reference when the queue is full.

Syntax:

capture <index> stop_on_full | continue_on_full

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command should be issued after an **add_adapter** and **setup_driver** command have successfully completed.

Logged Return Values:

None

Example:

capture 0 continue_on_full

close_adapter

This command places the driver and LASAR-155 hardware in the closed state. It masks the interrupt line attached to the PCI bus, thereby no longer handling interrupts. It also stops polling LASAR-155 counters if polling was enabled.

Syntax:

close_adapter <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command should follow an **open_adapter** command. Processing of this command may result in an asynchronous return depending on the value specified for the LasarCloseTimeout (see the installation procedure). A **wait** command may be required following this command, such that processing associated with this command completes before another command is issued.

Logged Return Values:

None

Example:

close_adapter 0

compare

This command compares the value held in REGISTER_VALUE with the <value> parameter.

Syntax:

compare **<value>**

Parameter Description:

<value> is a 32-bit value, or DWORD, that is compared with the value held in REGISTER_VALUE.

Precautions:

None

Logged Return Values:

None

Example:

compare 0

cops_access

This command is used to access the COPS block of the LASAR-155 to provision/unprovision a receive/transmit VC. The following sequence of events occur on issue of this command:

1) Read the LASAR-155 COPS Control register (0x280) if it has not yet been read (This register is read during the first **cops_access** command or during the **open_adapter** command, whichever comes first in the script file. It is re-read during the first **cops_access** command, or **open_adapter** command, that follows a **reset_adapter** or **close_adapter** command.).

2) Determine and write the VCNUM field of the COPS VC Number register (0x288) based on the <vpi_reg> and <vci_reg> fields of the command.

3) Write the <vc_control_status>, <vpi_reg>, <vci_reg>, <vc_parms> fields to the appropriate COPS registers, as required.

4) Write the <access_control> field to the COPS Parameter Access Control register (0x284). Query the BUSY bit of this register to determine if the access completed. If it did not complete then start a timer of duration VcTimeout (Default is 1 millisecond if it was not redefined via a keyword in the "system.ini" file).

5) On expiry of the timer query the BUSY bit. Issue an asynchronous confirm message stating the success or failure of the COPS access along with the return values. A failure only indicates that the COPS access failed to complete within the user configured "VcTimeout". The COPS access may complete after the timeout expires.

6) The next **cops_access** command that is received before expiration of the timer, and before the BUSY bit has reset, will fail.

Syntax:

cops_access <index> <access_control> <vpi_reg> <vci_reg> <vc_control_status> {<vc_parameters>}

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<access_control> is 16-bit value which specifies the COPS access and is written to the COPS Parameter Access Control register (0x284). Refer to the hardware specification [1] for further information.

<vpi_reg> is a 32-bit value read from or written into the COPS VPI register (0x28C).

<vci_reg> is a 32-bit value read from or written into the COPS VCI register (0x290).

<vc_control_status> is a 32-bit value read from or written into the COPS Control and Status register (0x294).

<vc_parms> is a 32-bit value read from or written into the COPS Control and Status register (0x294).

Precautions:

This command should follow an **add_adapter** command and should precede a **close_adapter** or **reset_adapter** command. This command is associated with an asynchronous confirm message and may need to be followed with **wait** commands.

Logged Return Values:

The adapter **index** and the following LASAR-155 register values are returned:

- **COPS VPI**
- **COPS VCI**
- **COPS Control and Status**
- **COPS VC Parameters**

Example:

cops_access 0 0x0 0x10 0x81 0x8000 0xFFF

dump

This command places an image of RAM into a file.

Syntax:

dump <filename> <start offset address> <size> {file index}

Parameter Description:

<filename> is a valid DOS path and or file name. The format of data placed in the file is either binary, "*.b" or "*.d", depending on the file name chosen, as specified in section 5.

<start offset address> is a 32-bit, DWORD, value which represents the first byte within locked RAM which is written into the file. The value is an offset from the physical base of RAM locked by the driver.

<size> is the number of physically contiguous bytes which are copied to the file.

<file index> is an optional field that must be specified for a filename that implies a binary data file format. This field specifies the byte offset within the file where the first byte is written to, instead of the beginning of the file.

Precautions:

None

Logged Return Values:

None

Example:

dump c:\tmp\second.d 0 100
dump test.bin 0x100000 0x100000
dump third.b 0x400 1

end

This command terminates execution of a script file. It should be inserted at the last line of a script command file. It specifies the last line of execution in a script command file.

Syntax:

end

Parameter Description:

None

Precautions:

If this command is not inserted in the script then the execution will terminate after 1000 empty lines are encountered in the script window.

Logged Return Values:

None

Example:

end

filter

This command causes filtering of all references received on a single VC. All references received with the specified VCNUM are processed according to the **setup_driver** command, while all other references are immediately returned to a receive free queue.

Syntax:

filter <index> <VCNUM>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<VCNUM> is an integer in the range 0 thru 127 which represents a COPS table index associated with the VPI and VCI of a provisioned RVC. Refer to [1] for a description on translation of VPI and VCI into a VCNUM.

Precautions:

This command is valid after the **add_adapter** command has successfully completed.

Logged Return Values:

None

Example:

filter 0 0
filter 0 127

init_ram

This command writes data to the host RAM block locked by the driver, as specified by the commands in a RAM initialization file.

Syntax:

init_ram <filename>

Parameter Description:

<filename> is a valid DOS path and file name. This file must be a RAM initialization file as specified in section 5.3.

Precautions:

None

Logged Return Values:

None

Example:

init_ram c:\pmc\lasar155\ram.txt

load

This command writes RAM with data from a file.

Syntax:

load <filename> {<start offset address> <size><file index>}

Parameter Description:

<filename> is a valid DOS path and file name. The format of data read from the file is either binary, "*.b" or "*.d", depending on the file name chosen, and as specified in section 5. Data written to RAM is sourced from this file.

<start offset address> is a 32-bit, DWORD, value which represents the first address within locked RAM which is written. The value is an offset from the physical base of RAM locked by the driver. This field must be specified for a filename that implies a binary data file format.

<size> is the number of physically contiguous bytes which are written to RAM. This field must be specified for a filename that implies a binary data file format.

<file index> is an optional field that must be specified for a filename that implies a binary data file format. This field specifies the byte offset within the file where the first byte is written to, instead of the beginning of the file.

Precautions:

None

Logged Return Values:

None

Example:

```
load c:\tmp\second.d
load test.bin 0x100000 0x100000 0
load third.b
```

locate

This command invokes the motherboard's PCI BIOS function call to locate a device on the PCI bus, as specified in reference [2].

Syntax:

locate <index> (find_pci_device | find_pci_class_code) <search key>

Parameter Description:

<index> is an integer which specifies the n'th pci device that matches the search key.

<search key> depends on whether **find_pci_device** or **find_pci_class_code** is specified:

- for **find_pci_device** its a DWORD value.The device id is the most significant WORD and the vendor id is the least significant WORD.
- for **find_pci_class_code** its a three byte value specifying the class code.

Precautions:

None

Logged Return Values:

- Bus Number, Device Number, and Function Number: This WORD value is arranged as follows: Bus Number is returned in the high byte, Device Number is returned in the top five bits of the low byte and Function Number is returned in the bottom three bits of the low byte.

Example:

locate 0 find_pci_device 0x737511F8

mask

The command enables or disables the driver from placing a message into the pipe. This prevents the graphical user interface from displaying the specified indication message or asynchronous confirm message. Masking a message may prevent the pipe from overflowing in an unusual test scenario. The performance of the device driver may also improve when messages are blocked.

Syntax:

mask <index> enable | disable <message identifier>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<message identifier> is one of the following driver event messages:

CNF_TRANSMIT_REQUEST
IND_RX_PACKET
IND_RX_MANAGEMENT
IND_SW_TRACE
IND_INT_MASTER
IND_INT_PCID
IND_INT_RALP
IND_INT_TALP
IND_INT_TATS
IND_INT_SAR_PMON
IND_INT_TACP
IND_INT_RACP
IND_INT_RPOP
IND_INT_RLOP
IND_INT_RSOP

Precautions:

All messages are enabled following an **open_adapter** command.

Logged Return Values:

None

Example:

mask 0 disable IND_INT_PCID

open_adapter

This command causes the following actions:

- install an interrupt service routine for the adapter card and unmasks the motherboard's programmable interrupt controller.
- begin polling of the LASAR-155 count registers
- report the status of LASAR-155 alarms to the alarms display
- initialize the driver data structures with an image of the LASAR-155 registers.
- verify that the LASAR-155 registers have meaningfull values.
- activate the transmit request machine by setting the TRMEN bit within the PCID CONTROL register of the LASAR-155 device.

Syntax:

open_adapter <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

The **add_adapter** command should return successfully before this command is run. The **reset_adapter** command is recommended before issuing this command.

Logged Return Values:

None

Example:

open_adapter 0

pause

This command pauses execution of the script file until the user chooses **Script Control | Run** from the main menu.

Syntax:

pause

Example:

pause

pciinfo

This command requests information about the motherboard BIOS, the driver and the location of fixed physical RAM locked by the driver. The RAM address and size are fixed, and data within this RAM space is not moved or swapped to the file system.

Precautions:

None

Syntax:

pciinfo

Logged Return Values:

- **Hardware Mechanism**: This 8-bit value is the hardware mechismism, as specified by the PCI SIG [2].
- **Last PCI Bus**: This 8-bit value is the last PCI bus, as specified by the PCI SIG [2]. A PCI bus on the motherboard is numbered starting at 0 and running up to this value.
- **Entry Point**: This 32-bit value specifies the address of the PCI BIOS services. A zero value indicates that the PCI BIOS services do not exist, or could not be found.
- **PCI BIOS Revision**: This is the revision number specified by the motherboard BIOS manufacturer. This 16-bit value is is formatted as specified by the PCI SIG [2].
- **RAM Physical Base Address**: This 32-bit value is the physical base address of the block of RAM that is locked by the driver when Windows is launched. The user may request the size and location of the RAM block by specifying keywords in the "system.ini" file (see the installation procedures in section 2).
- **RAM Linear Base Address**: This 32-bit value is the linear base address of the block of RAM that is locked by the driver when Windows is launched. The user may request the size and location of the RAM block by specifying keywords in the "system.ini" file (see the installation procedures in section 2).
- **RAM Number of Bytes**: This 32-bit value is the number of bytes within the block of RAM that is locked by the driver when Windows is launched. The user may request the size and location of the RAM block by specifying keywords in the "system.ini" file (see the installation procedures in section 2).
- **Driver Build Number**: This 8-bit value is used to track the version of the driver that is installed, and which manages the LASAR-155 PCI device.

Example:

pciinfo

read

This command causes the driver to read a register that is mapped into memory space. The register resides within the LASAR-155 adapter and may belong to the LASAR-155 configuration space, the LASAR-155 normal mode register space, the external device or the ROM. The register that is read is stored in the variable REGISTER_VALUE.

Syntax:

read <index> <register space> <offset> {<compare value> {<mask value>}}

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<register space> is either:

- **config byte**: An 8-bit value within the LASAR-155 configuration register space.
- **config word**: A 16-bit value within the LASAR-155 configuration register space.
- **config dword**: A 32-bit value within the LASAR-155 configuration register space.
- **rom**: An 8 bit value within the ROM space attached to the LASAR-155.
- **exd**: A 16-bit value within the external device register space which is attached to the LASAR-155.
- **lasar**: A 32-bit value within the LASAR-155 normal mode register space.

<offset> is a valid register offset from the base of the register space.

<compare value> is a value which is compared with the value read. This command fails if the value read is different from the compare value.

<mask value> a binary AND operation is performed on the value read before the value is compared with the <compare value>. This field is especially useful when reading from a 32-bit LASAR-155 register that has unused (or undefined) bit fields.

Precautions:

The LASAR-155 adapter should be inserted in the PCI slot, or the session should be run in test mode. The **locate** command must be completed successfully in order to read from LASAR-155 configuration space. The **add_adapter** command must be completed successfully in order to read from the LASAR-155 normal mode register space, the external device register space, or the ROM space.

Logged Return Values:

- **REGISTER_VALUE**: This is the value read from the register. It may be an 8-bit value, a 16-bit value or a 32-bit value, depending on the register space accessed.

Example:

```
read 0 config dword 0x10
read 0 lasar 0x358 0x010 0xFFFF
```

register_value

The register_value is a 32-bit variable within the LASAR-155 application software. A register value is automatically loaded into this variable when the **read** command is completed. The variable may also be written to a hardware register via the **write** command. This command can be used to modify the variable after it is read, and before it is written to a register. The **to_physical** and **to_offset** commands are also available to modify the variable.

Syntax:

register_value <assignment operation> <value>

Parameter Description:

<assignment operation> is used to directly assign <value> to register_value, or to logically OR, AND or XOR register_value with <value>. Each assignment operator is shown below:

- =
- |=
- &=
- ^=

<value> is a 32-bit value used in the assignment.

Precautions:

None

Logged Return Values:

None

Example:

register_value &= 0xFFFF

remove_adapter

This command reverses the effect of the **add_adapter** command.

Syntax:

remove_adapter <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command must be issued any time after a **reset_adapter**, **add_adapter** or **close_adapter** command is successfully completed, but not after an **open_adapter** command.

Logged Return Values:

None

Example:

remove_adapter 0

reset_adapter

This command causes the driver software to reset its internal data structures and to reset the LASAR-155 adapter as follows:

1) write 0x0 to LASAR Master Test register (0x400)

2) write 0x8000 to LASAR Master Reset register (0x000)

3) reset TRMEN bit within the PCID Control register (0x014)

4) write 0x0000 to LASAR Master Reset register (0x000)

5) write 0x7 to COPS Control register (0x280)

6) set INIT bit within the LASAR Master Control register (0x004)

7) run a timer with duration equal to the user specified ResetTimeout (see the installation procedure in section 2).

8) on timeout of the timer verify that the INI_STAT bit of the LASAR Master Control register () is reset. Issue an asynchronous confirm message showing whether the reset procedure was successful.

Syntax:

reset_adapter <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

The **add_adapter** command must be completed successfully before this command is run. This command should not be run after the **remove_adapter** command. This command involves an asynchronous confirmation and may require **wait** commands following the **reset_adapter** command.

Logged Return Values:

None

Example:

reset_adapter 0

savelog

This command causes the contents of the log file window to be written to the file system. The log file window is cleared.

Syntax:

savelog

Parameter Description:

None

Precautions:

The log file window is a Windows 3.1 edit control and has a limitation in that it cannot hold more than 32K text characters. Characters that are logged in excess of 32K are lost and the user is warned by the PC speaker. (The indicator at the bottom of the main window shows how many characters are in the log file window). Using the **savelog** command throughout a script command file should ensure that this error condition is avoided.

Logged Return Values:

None

Example:

savelog

setup_driver

This command specifies the mode of the device driver in handling a LASAR-155 adapter card. This command is optional, and when it is not used the defaults are assumed (ie no buffered transmit requests, no loopback of receive data, no capture or filter of receive data). The default timeouts may be specified by keywords in the "system.ini" file (see the installation procedures of section 2).

The rx_loopback field instructs the driver to route incoming receive data onto the low priority transmit queue. Each receive descriptor's buffer is linked to a transmit descriptor. The receive data is translated into a transmit packet by linking the transmit descriptors. Transmit descriptors are taken from the base of the transmit descriptor table and marked "in use". The setup_driver command must specify the number of transmit descriptors that are reserved for loopback. On completion of transmit the driver marks the descriptors as "free".

The tx_buffered field instructs the driver to hold transmit requests in the stimulus table until the **startd** command is issued. On issue of **startd**, the driver queues all transmit requests into the high or low priority transmit queue, with timeouts between each queue access specified via the **timeout** command.

The tx_buffered_table mode is like the tx_buffered mode except that after the last transmit request of the stimulus table is queued, the driver repeats the transmit requests from the first entry of the stimulus table.

The tx_buffered_queue mode is like the tx_buffered mode except that after each transmit request is completed and the LASAR-155 places the reference on the free queue, the driver repeats transmit of the packet by placing the reference from the free queue onto the low priority transmit queue.

The rx_filter_enabled field instructs the driver to process data associated with only one RVC (see the **filter** command). Processing of this data is loopback and/or capture, as specified with the other fields of the **setup_driver** command.

The rx_capture_enabled field instructs the driver to capture receive data.

Syntax:

setup_driver <index> <poll timeout> <close timeout> {rx_filter_enabled} {rx_capture_enabled} {(tx_buffered | tx_buffered_table | tx_buffered_queue)} {rx_loopback <number of transmit descriptors>}

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<poll timeout> is 32-bit value, representing the time in milliseconds between running the poll timeout routine. This routine polls the LASAR-155 counters. The polling commences on issue of the **open_adapter** command.

<close timeout> is a 32-bit value, representing the time in milliseconds in which the driver waits to complete transmit and/or receive of data before the motherboard interrupt line is masked, the provisioned VC's are unprovisioned, and the driver is placed in the LCB_REGISTERED state. The timeout commences on issue of the **close_adapter** command. If VC's are active following this timeout, the VC's are unprovisioned and the asynchronous confirm for the close_adapter command has a failure status.

<number of transmit descriptors> is a 16-bit value which specifies the number of transmit descriptors, beginning with reference 0, which are reserved for creating transmit packets. This field is relevent only in the rx_loopback mode.

Precautions:

This command should be issued after the **add_adapter** command and before the **open_adapter** command.

Logged Return Values:

None

Examples:

setup_driver 0 1000 0 tx_buffered_table
setup_driver 0 500 0 rx_loopback 0xFF
setup_driver 0 0 0 rx_filter_enabled rx_capture_enabled

startd

This command causes the driver to start queuing transmit references on the high or low priority transmit ready queue, as specified in the stimulus table. This command is only valid when the driver is run in the tx_buffered, tx_buffered_table or tx_buffered_queue mode, as specified via the **setup_driver** command.

Syntax:

startd <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command should follow a **stopq** command (ie. the transmit references, queue flag and timeouts should be assigned to the stimulus table).

Logged Return Values:

None

Example:

statd 0

stopd

This command causes the driver to stop queuing transmit references from the stimulus table onto the high or low priority transmit ready queue. This command is only valid when the driver is run in the tx_buffered, tx_buffered_table or tx_buffered_queue mode, as specified via the **setup_driver** command.

Syntax:

stopd <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command should follow a **startd** command. This command will fail if the driver is in the tx_buffered mode and the last reference in the stimulus table is queued before this command is issued.

Logged Return Values:

None

Example:

stopd 0

startq

This command instructs the driver that all **transmit** commands and all **timeout** commands following this command are to be assigned to the stimulus table, instead of queueing a transmit reference directly to a LASAR-155 transmit ready queue.

Syntax:

startq <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

The driver should be in the tx_buffered mode as specified via a **setup_driver** command.

Logged Return Values:

None

Example:

startq 0

stopq

This command instructs the driver that all **transmit** commands and all **timeout** commands following this command should not be assigned to the stimulus table. Transmit references are directly assigned to a LASAR-155 transmit ready queue.

Syntax:

stopq <index>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

Precautions:

This command should follow a **startq** command.

Logged Return Values:

None

Example:

stopq 0

timeout

This command assigns a timeout, in milliseconds, between successive transmit references that are queued onto a LASAR-155 transmit ready queue when the driver is running in tx_buffered, tx_buffered_table, or tx_buffered_queue mode. The **timeout** command should be placed between **transmit** commands of the script file window to assign the timeout between the two **transmit** commands.

Syntax:

timeout <index> <duration>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<duration> is a 32-bit value which specifies the timeout in milliseconds.

Precautions:

This command should be placed between **transmit** commands of a script file. It should follow a **startq** command and precede a **stopq** command.

Logged Return Values:

None

Example:

timeout 0 0x100

to_physical

This command converts an offset address into a physical address. The offset address is held in REGISTER_VALUE following a **read** command or a **register_value** command.

Syntax:

to_physical

Parameter Description:

None

Precautions:

This command should follow a **read** command or a **register_value** command.

Logged Return Values:

None

Example:

to_physical

to_relative

This command coverts a phsyical address into an offset address. The phsyical address is held in the software holding register following a **read** command or a **register_value** command.

Syntax:

to_relative

Parameter Description:

None

Precautions:

This command should follow a **read** command or a **register_value** command.

Logged Return Values:

None

Example:

to_relative

transmit

This command causes the driver to write a packet reference to the high priority transmit ready queue, or the low priority transmit ready queue. Or if the driver is in the tx_buffered, tx_buffered_table or tx_buffered_queue mode, and the command lies between a **startq** and **stopq** command of the script file, the driver places the reference and queue flag into the stimulus table.

Syntax:

transmit <index> (high | low) <reference>

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<reference> is a 16-bit value identifying the first transmit descriptor of the transmit packet. References are defined in the LASAR-155 hardware specification [1].

Precautions:

This transmit packet should be defined within RAM (see **init_ram** command). This command must be issued following an **open_adapter** command but before a **reset_adapter** or a **close_adapter** command.

Logged Return Values:

None

Example:

transmit 0 low 0x1

wait

This command causes a timeout before the next script command is issued. The timeout is approximately 500 milliseconds in duration.

Syntax:

wait

Parameter Description:

None

Precautions:

None

Logged Return Values:

None

Example:

wait

write

This command causes the driver to write to a register that is mapped into memory space. The register resides within the LASAR-155 adapter and may belong to the LASAR-155 configuration space, the LASAR-155 normal mode register space, the external device or the ROM. The value or the address of the register can be specified in the command fields or taken from the variable, register_value.

Syntax:

write <index> <register space> <offset> | register_value <value> | register_value

Parameter Description:

<index> is an integer which specifies the n'th LASAR-155 adapter located by the motherboard's BIOS.

<register space> is either:

- **config byte**: An 8-bit value within the LASAR-155 configuration register space.
- **config word**: A 16-bit value within the LASAR-155 configuration register space.
- **config dword**: A 32-bit value within the LASAR-155 configuration register space.
- **rom**: An 8 bit value within the ROM space attached to the LASAR-155.
- **exd**: A 16-bit value within the external device register space which is attached to the LASAR-155.
- **lasar**: A 32-bit value within the LASAR-155 normal mode register space.

<offset> is a valid register offset from the base of the register space.

Precautions:

The LASAR-155 adapter should be inserted in the PCI slot, or the session should be run in test mode. The **locate** command must be successful in order to read from LASAR-155 configuration space. The **add_adapter** command must be successful in order to read from the LASAR-155 normal mode register space, the external device register space, or the ROM space.

Logged Return Values:

None

Example:

write 0 config dword 0x10 0x10000000
write 0 lasar 0x31C register_value

5.4 Ram Initialization File Commands

The RAM initialization file commands are described in the following pages. The commands are listed alphabetically.

The RAM initialization commands are used to place data into the block of RAM allocated by the driver. These commands must be place in a seperate file from the script command file. The filename for the RAM initialization commands is specified as a parameter in the **init_ram** script command of section 5.2.

These commands assign data to physical RAM addresses based on the values specified within the LASAR-155 registers, therefore the registers should be initialized before the **init_ram** command is run.

create_packet

This command initializes all descriptors associated with a packet. It assigns values to all the fields of each descriptor and links descriptors as required to form the packet. The descriptors are also linked to the data buffers. This command does not load the data buffers, the **load_data** command should be used.

Syntax:

create_packet <TVC> <packet length> <buffer start offset> {buffer_end <buffer end offset>} [descriptor linkage]

where [descriptor linkage] is defined as the following fields:

<reference> <buffer size> <td_template identifier> {for <sequence count>} {[descriptor linkage]}

Parameter Description:

<TVC> is an integer between 0 and 127 which specified the VCNUM of the provisioned VC that shall transmit the packet.

<packet length> is a 16-bit value, which specifies the total number of bytes within the buffers of a packet. Each buffer that is attached to a descriptor must be full of data, and the last buffer may, or may not, be full.

<buffer start offset> specifies the RAM offset address of the buffer which is attached to each descriptor. The first descriptor in <descriptor linkage> will have a physical buffer address which matches the <buffer start offset>. Descriptors within <descriptor linkage> will be assigned buffer addresses, in sequence, depending on the buffer size specified for each descriptor. This value is a 32-bit address, or DWORD.

<buffer end offset> is an optional field. It is a 32-bit address, or DWORD, specified as an offset from the base of RAM that was locked by the driver. When this field is specified the descriptors are linked to buffers, in sequence, except that when the next buffer start address of a descriptor needs to be assigned, and the address is equal to <buffer end offset>, the <buffer start offset> is assigned to that descriptor. Following descriptors will be assigned buffers following the <buffer start offset>.

<reference> is a 16-bit, WORD, value which represents an index into the LASAR-155 descriptor table.

<buffer size> is a 16-bit, WORD, value which represents the size of buffer attached to each descriptor in [descriptor linkage].

<td_template identifier> is a 16-bit, WORD, value which defines fields of a descriptor according the the td_template command.

<sequence count> is a signed integer which specifies the sequence and number of descriptors linked in the chain beginning with <reference>. Each descriptor derived from <sequence count> is assigned the same <td_template identifier> and <buffer_size> as the descriptor associated with <reference>.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers
- All start, write, read and end index registers

Example:

td_template 0 0x4201 0x0000
td_template 1 0x8201 0x0000

create_packet 127 0x40 0x00310000 1 0x4000 0
create_packet 0 0x100 0x00310000 20 0x20 0 for -7 2 0x20 1

load_data

This command initializes a contiguous block of physical memory. The prbs23 field can be specified to insert a prbs23 data pattern into physical memory, otherwise data is taken from a file or as specified by an inline data pattern.

Syntax:

load_data <start offset> <size> <inline data pattern> | <filename> | prbs23

Parameter Description:

<start offset> is a 32-bit value which specifies an offset address within RAM locked by the driver. This address is the first byte to be loaded with data.

<size> is a 32-bit value which specifies the number of bytes begining at the <start offset> which is written with data.

<inline data pattern> is the inline data pattern written to RAM. The data pattern must be specified in hexadecimal format. The pattern is written into RAM repeatatively until the block of RAM specified by the <start offset> and the <size> is filled. A data pattern which is a BYTE, WORD or DWORD may be inserted into RAM as an incrementing or decrementing pattern. For example:

- **0x30313233343536** is written into RAM as a repeating pattern of bytes. (ie. 0x3031323334353630313233343536 ...). This pattern is not a BYTE, WORD or DWORD size and cannot have a signed increment.
- **0x00** initializes the RAM block with zeros.
- **0x00 +5** initializes RAM with a BYTE sequence that increments by 5. (ie. 0x00050a0f04 ...)
- **0xFFFFFFFF -1** initializes RAM with a decrementing sequence of DWORDs. (ie. 0xFFFFFFFFFFFFFFFEFFFFFFFDFFFFFFFC ....)

<filename> is either a binary data file, a "*.b" data file, or a "*.d" data file.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers

- All start, write, read and end index registers

Example:

```
load_data 0x00310000 0x30000 prbs23
load_data 0x00000000 0x10000 0x123456789abcdef
load_data 0x00100000 0x00100000 c:\pmc\lasar155\ramdata.bin
```

rm_free

This command initializes descriptors within the receive management table and places them in the receive management free queue. It does not change the LASAR-155 queue start, write, read or end index registers.

Syntax:

rm_free <start index> <buffer start offset> <buffer size> [reference sequence]

where [reference sequence] is:

<reference> {for <sequence count>} {[reference sequence]}

Parameter Description:

<start index> is an integer which specifies the first queue element, following the start index which is written.

<buffer start offset> is a 32-bit, or DWORD, RAM offset address which locates the block of memory which is linked to a descriptor.

<buffer size> is a 16-bit, WORD, value that specifies the size of the buffer associated with each descriptor.

<reference> is a 16-bit, WORD, value which represents an index into the LASAR-155 descriptor table.

<sequence count> is a signed integer which represents the number of descriptors that are initialized, and the sequence of references written to the queue.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers
- All start, write, read and end index registers

Example:

rm_free 0 0x00308000 0x40 0 for 0x1FF

rp_free_large

This command initializes descriptors within the receive packet table and places them in the receive packet large free queue. It does not change the LASAR-155 queue start, write, read or end index registers.

Syntax:

rp_free_large <start index> <buffer start offset> <buffer size> [reference sequence]

where [reference sequence] is:

<reference> {for <sequence count>} {[reference sequence]}

Parameter Description:

<start index> is an integer which specifies the first queue element, following the start index which is written.

<buffer start offset> is a 32-bit, or DWORD, RAM offset address which locates the block of memory which is linked to a descriptor.

<buffer size> is a 16-bit, WORD, value that specifies the size of the buffer associated with each descriptor.

<reference> is a 16-bit, WORD, value which represents an index into the LASAR-155 descriptor table.

<sequence count> is a signed integer which represents the number of descriptors that are initialized, and the sequence of references written to the queue.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers
- All start, write, read and end index registers

Example:

rp_free_large 0 0x00308000 0x40 0x100 for 0x1FF

rp_free_small

This command initializes descriptors within the receive packet table and places them in the receive packet small free queue. It does not change the LASAR-155 queue start, write, read or end index registers.

Syntax:

rp_free_small <start index> <buffer start offset> <buffer size> [reference sequence]

where [reference sequence] is:

<reference> {for <sequence count>} {[reference sequence]}

Parameter Description:

<start index> is an integer which specifies the first queue element, following the start index which is written.

<buffer start offset> is a 32-bit, or DWORD, RAM offset address which locates the block of memory which is linked to a descriptor.

<buffer size> is a 16-bit, WORD, value that specifies the size of the buffer associated with each descriptor.

<reference> is a 16-bit, WORD, value which represents an index into the LASAR-155 descriptor table.

<sequence count> is a signed integer which represents the number of descriptors that are initialized, and the sequence of references written to the queue.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers
- All start, write, read and end index registers

Example:

rp_free_small 0 0x00308000 0x40 0 for 0xFF

td_template

This command defines a common template for a transmit descriptor.

Syntax:

td_template <td_template identifier> <flags> <cpaal5>

Parameter Description:

<td_template identifier> is an integer between 0 and 49, which identifies the template.

<flags> is a 16-bit, or WORD value which initializes the flags of a descriptor. The flags are contained in the second WORD of a transmit descriptor.

<cpaal5> is a 16-bit, WORD, value which initializes the cpaal5 fields of a descriptor. The cpaal5 fields are contained within the tenth WORD of a transmit descriptor.

Precautions:

The LASAR-155 register space must be accessible, and the following LASAR-155 registers must be correctly assigned before this command is run. The block of RAM locked by the driver must encompass the addresses for which the LASAR-155 registers are programmed.

- All transmit and receive descriptor tables
- All queue base address registers
- All start, write, read and end index registers

Example:

td_template 0 0x4201 0x0000

6 FILE FORMAT DESCRIPTION

6.1 Script File

The script file is a textually formatted file containing valid commands, as specified in section 5.1. Any DOS file name is valid, but the recommended filename suffix is ".run". The contents of the file should be ANSI characters with the carriage return and line feed characters delimiting each line.

6.2 Ram Initialization File

The script file is a textually formatted file containing valid commands, as specified in section 5.1. Any DOS file name is valid, but the recommended filename suffix is ".ram". The contents of the file should be ANSI characters with the carriage return and line feed characters delimiting each line.

6.2 Log File

The log file is a textually formatted file that is created by the software and has a name identical to the script file name except that the suffix is replace with the ".log" suffix. The contents of a log file is ANSI characters with the carriage return and line feed characters delimiting each line.

6.3 List Box File

List box files are named according to the display which makes use of a list box file. A drop down list box within the following displays specifies a textual description for a memory address:

- host RAM display (file is named "ram.txt")
- LASAR register display (file is named "lr.txt")
- driver LCB display (file is named "lcb.txt")
- driver adapter table display (file is named "at.txt")
- driver stimulus table display (file is named "st.txt")

The format for such a file is:

<hexadecimal byte offset from base address><blank character(s)><text><carriage return and line feed>

The following two lines are valid formats when edited from a text editor such as notepad:

00000000 LASAR-155 MASTER RESET
00000004 LASAR-155 Master Configuration

6.4 Ram Initialization File

The ram initialization file is a textually formatted file containing valid commands, as specified in section 4.3. Any DOS filename is valid. The contents of the file are ANSI characters with the carriage return and line feed characters delimiting each line.

6.5 Binary Data File

The purpose of a binary data file is to hold an image of RAM. The data of a binary data file may be loaded into RAM or dumped from RAM to the file system.

A binary data file is named such that it does not have the ".b" or ".d" suffix (ie "ram.bin" is valid whereas "ram.d" or "ram.b" are invalid file names for the binary data format).

The data in a binary file is the same format as memory (ie. byte 0 followed by byte 1 followed by byte 2 etc.). The file can only be read via a binary or hex editor and the size of a binary file is equal to the number of bytes contained in the file. The advantage of using a binary file is that the amount of hard disk space to store the file is much less than any of the textual formats. For example, six MBytes of memory can be stored in a six MByte binary file, whereas the the file size would be more than 13 times as large for a filename with the ".b" suffix.

The binary data file may be created by pressing the **Data Out** button on a display in the following list, and choosing a filename that does not have the ".b" or ".d" suffix when prompted for a file name:

- host RAM display
- LASAR register display
- driver LCB display
- driver adapter table display
- driver stimulus table display

Similarly, a binary data file may be loaded into memory by pressing the **Data In** button on any display in the list above, and choosing a filename that does not have the ".b" or ".d" suffix.

The **load**, **dump** and **load_data** script commands described in section 4 may be used to transfer data between the file system and RAM. Binary data can be transfered only if a valid binary data file name is specified.

6.6 ".B" Data File

This file type is processed by the same displays and script commands as mentioned in the binary data file of section 5.5. The difference with the ".b" formated file is that the file name must have the ".b" (or ".B") suffix, and the file format is textual. This file type may be edited via a text editor such as notepad.

It has the following format:

<byte offset from base address in hex characters> <blank character(s)> <byte value in hex characters> <carriage return and line feed>

The following three lines are valid formats when edited from a text editor such as notepad:

00000000 0
00000003 1a
00000001 FF

The offset address of each line may not be contiguous or aligned to a DWORD. The software will only write one byte of data at the each specified offset into RAM.

6.7 ".D" Data File

This file type is processed by the same displays and script commands as mentioned in the binary data file of section 5.5. The difference with the ".b" formated file is that the file name must have the ".d" (or ".D") suffix, and the file format is textual. This file type may be edited via a text editor such as notepad.

It has the following format:

<byte offset from base address in hex characters> <blank character(s)> <dword value in hex characters> <carriage return and line feed>

The following three lines are valid formats when edited from a text editor such as notepad:

00000000 00
00000003 12345678
00000001 FFFFF

The offset address of each line may not be contiguous or aligned to a DWORD. The software will only write 4 bytes of data at each specified offset into RAM.

7 INTERPRETING THE LOG FILE

7.1 Contents of the Log File

The log file holds the results of script commands that are run via the script file window. Text for the following events is written to the log file:

- echo of each script commands that was issued
- success or failure of each script command issued
- return values for script commands
- driver event messages

Each event is logged in the order of occurance and with a timestamp.

7.2 Driver Event Messages

The driver responds to the PCI bus interrupt line, and to a system timer timeout, by the issue of a driver event message. Each driver event message is either an asynchronous confirm message or an indication message.

A confirm message indicates completion of a script command. It may occur immediately after the script command, or asynchronously, some time later, after the driver has waited for a hardware event to complete. The text "PENDING" is written to the log file for a script command which involves completion of a hardware event, and where the result of the command is not known until the hardware event occurs. When the hardware event occurs the asynchronous confirm message is placed into the log file. Confirm messages are written beginning with the text "CNF_".

The indication message is initiated by a hardware interrupt or system timer timeout. Indication messages are written beginning with the text "IND_".

The following messages may be issued by the driver (a message that is issued by the driver is logged only if the **mask** script command has not been used to disable the message):

- **CNF_CLOSE_ADAPTER** is issued if the LasarCloseTimeout keyword, or the **setup_driver** script command, assigns a non-zero value for the close adapter timeout. The system timer is started when the **close_adapter** script command is processed. The driver logs the result of processing the script command after the timer expires by issuing the CNF_CLOSE_ADAPTER message.
- **CNF_COPS_ACCESS** is issued if the LasarVcTimeout keyword assigns a non-zero value for the VC timeout. The system timer is started when the **cops_access** script command is processed. The driver logs the result of processing the script

command after the timer expires, or before the timer expires, if the next **cops_access** script command is issued and the COPS table access completed before expiry of the timer.

- **CNF_RESET** is issued if the LasarResetTimeout keyword assigns a non-zero value for the reset timeout. The system timer is started when the **reset_adapter** script command is processed. The driver logs the result of processing the script command after the timer expires by issuing the CNF_RESET message.

- **CNF_TRANSMIT_REQUEST** is issued if the IOC bit of a transmit descriptor had been set and the LASAR-155 had completed processing of the transmit packet by placing the descriptor reference in the transmit reference free queue and activating the LASAR-155 interrupt pin. The driver issues this message only if the interrupt line of the PCI bus was unmasked by issuing the **open_adapter** script command.

- **IND_INT_MASTER** is issued whenever the PCI bus interrupt line is active (and unmasked), and any LASAR-155 master interrupt enable bits are active and the corresponding LASAR-155 master interrupt status register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_PCID** is issued whenever the PCI bus interrupt line is active (and unmasked), and any LASAR-155 PCID interrupt enable bits are active and the corresponding LASAR-155 PCID status register bits are active.

- **IND_INT_RALP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any RALP interrupt status register bits and the corresponding RALP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_TALP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any TALP interrupt status register bits and the corresponding TALP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_TATS** is issued whenever the PCI bus interrupt line is active (and unmasked), and any TATS interrupt status register bits and the corresponding TATS interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_SAR_PMON** is issued whenever the PCI bus interrupt line is active (and unmasked), and both the INTE and INTR bits of the SAR PMON count change register are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_RACP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any RACP interrupt status register bits and the corresponding RACP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_RPOP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any RPOP interrupt status register bits and the corresponding RPOP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_RLOP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any RLOP interrupt status register bits and the corresponding RLOP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_INT_RSOP** is issued whenever the PCI bus interrupt line is active (and unmasked), and any RSOP interrupt status register bits and the corresponding RSOP interrupt enable register bits are active. This message is issued only if the LASAR-155 PCID status register had the MPEN bit active.

- **IND_RX_MANAGEMENT** is issued if a management cell has been received by the LASAR-155 and its reference has been placed in the receive management ready queue.

- **IND_RX_PACKET** is issued if a packet has been received by the LASAR-155 and its reference has been placed in the receive reference ready queue.

- **IND_SW_TRACE** is issued to trace execution of a failure condition or an unexpected event.

7.3 Success or Failure in Execution of a Script Command

Failed execution of a script command is noted in the log file. The command which failed is shown in the log file with the text "FAILURE" following it, or within the asynchronous confirm message.

Successful execution of a script command is not logged in the log file, except in the case of an asynchronous confirm message, where the text "SUCCESS" is shown within the message.

7.4 Trace of Software Execution

The software trace message is logged to help in debugging script file errors or adapter hardware problems. A software trace message is usually generated when a script command failed. A software trace message is written beginning with the text "IND_SW_TRACE".

7.5 Return Values

The return values for a script command are logged immediately after the script command, or in the case of a pending confirm, they are logged with the asynchronous confirm message. The description of a script command in this document specifies the return values.

NOTES

PMC-951014p1 ref 931127a3 Issue date: March, 1995.

Printed in Canada