# 5 DELQA-PLUS Installation

This chapter contains installation instructions for the DELQA-PLUS board.

These instructions supplement the basic installation instructions in the *DELQA User's Guide*, which apply to all DELQA boards.

> **NOTE**
>
> There can be more than one DELQA-PLUS board on the same Q-bus. For convenience, this chapter always refers to the first DELQA-PLUS board on a two-DELQA-PLUS-board Q-bus system (unless otherwise noted). For more information about running more than one DELQA-PLUS board on a Q-bus, see Section 5.3.

## 5.1 Introduction

The DELQA-PLUS board can operate in two modes, DELQA-normal mode and DELQA-T mode. For more information on the differences between DELQA-normal and DELQA-T modes, see Chapter 6 in this *Addendum to DELQA User's Guide*.

This chapter describes how to install the DELQA-PLUS board when it will be used in DELQA-T mode. For information on using the DELQA-PLUS board as a DELQA-normal board, see to the *DELQA User's Guide*.

The basic steps to install a DELQA-PLUS board when it will be used as a DELQA-T board are:

- Setting switches—Make sure the on-board switches are set correctly for DELQA-T mode operation.

- Make sure each DELQA-PLUS board is properly installed in the host chassis and is powered up and running.
- Running more than one DELQA-PLUS board—If there is more than one DELQA-PLUS board on the same system, follow the procedures in Section 5.3 in order to distinguish the two boards.

After you have performed these steps, the DELQA-PLUS board will be ready to operate as a DELQA-T board. To verify that the DELQA-PLUS board is running the proper ROM version for DELQA-T mode operation, see Appendix D.

For instructions on how to program the DELQA-PLUS board, see Chapter 6 in this *Addendum to DELQA User's Guide.*

## 5.2 Setting Switches On The DELQA-T Board

These are the same switches that are on the DELQA-normal board; the only differences are that they mean different things in DELQA-T mode than in DELQA-normal mode. The meanings of the switches in DELQA-T mode are described in Table 5–1.

Figure 5–1 shows the location of the five switches on the DELQA-T board.

Make sure the five switches on the DELQA-T board are in the CLOSED position (which is the factory default).

**CAUTION**

Do not set the switches while the board is powered ON.

**Figure 5–1: Switches on the DELQA-T board**

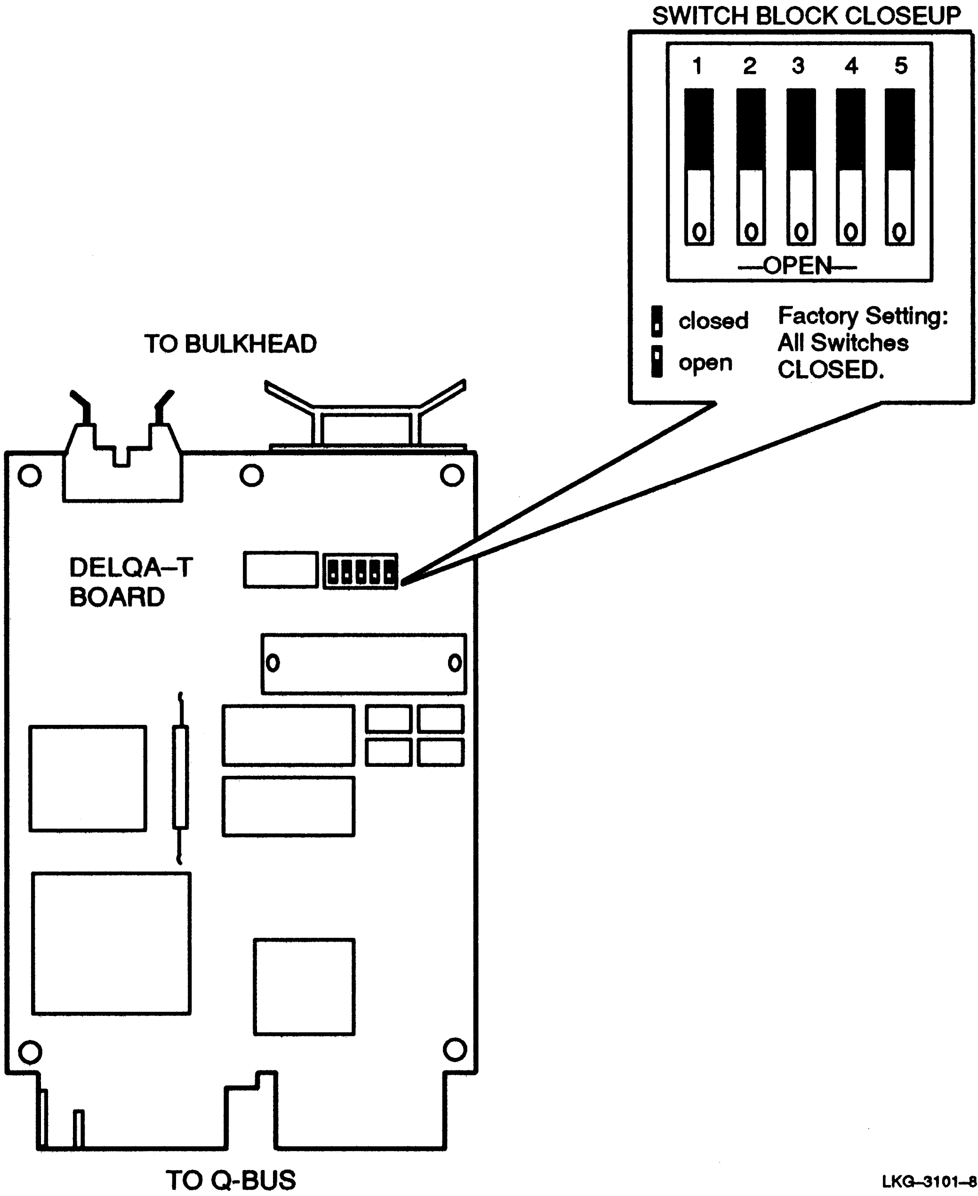


Table 5–1: Switches In DELQA-T Mode

| Switch | Setting | Meaning |
|---|---|---|
| S1 | CLOSED | This is the first DELQA-PLUS device; this device resides at Q-bus address 1777 4440 (octal). |
| | OPEN | This is the second DELQA-PLUS device; this device resides at Q-bus address 1777 4460 (octal). |
| S2 | | (Reserved.) |
| S3 | CLOSED | Selects DELQA-normal mode |
| | OPEN | Selects DEQNA-lock mode (Board can operate in DEQNA mode only, not in DELQA modes.) |
| S4 | CLOSED | HIT (host inactivity timer) initially disabled |
| | OPEN | HIT timer initially enabled |
| S5 | CLOSED | T-mode enabled (Board can operate as DELQA-normal board or DELQA-T board.) |
| | OPEN | T-mode disabled (Board can operate only as a DELQA-normal board or DEQNA board.) |

### 5.2.1 Switch S1 Identifies the Device

Switch S1 establishes the DELQA-PLUS board's Q-bus address and also allows you to distinguish two DELQA-PLUS boards that are on the same Q-bus:

- On the first DELQA-PLUS board, set switch S1 to CLOSED.
- On the second DELQA-PLUS board, set switch S1 to OPEN.

See also Table 5–1.

### 5.2.2 Switches S3 and S5 Select the DELQA-PLUS Board's Mode

Switches S3 and S5 let you restrict the DELQA-PLUS board's operating mode to DELQA-only (switch S3) or DEQNA-only (switch S5).

To make sure the DELQA-PLUS board can run in DELQA-T mode, make sure both switches S3 and S5 are CLOSED.

### 5.2.3 Switch S4 Sets Both HIT And Reboot Features

Switch S4 OPEN means "HIT initially enabled" in DELQA-T mode and "rebooting enabled" in DELQA-normal mode; in other words, switch S4 enables these two features together.

## 5.3 Running Two DELQA-PLUS Boards on the Same Q-bus System

There can be either one or two DELQA-PLUS boards on the same Q-bus system.

If there are two DELQA-PLUS boards on the same system, configure the two DELQA-PLUS boards as follows:

1. The first DELQA-PLUS must have switch S1 CLOSED.

   This sets the first DELQA-PLUS board's Q-bus address to 1777 4440 (octal).

2. The second DELQA-PLUS must have switch S1 OPEN.

   This sets the second DELQA-PLUS board's Q-bus address to 1777 4460 (octal).

Make sure the driver(s) uses these two different addresses to contact the two DELQA-PLUS boards correctly.

# 6
# DELQA-PLUS Programming

This chapter gives basic information about programming the DELQA-PLUS board.

## 6.1 Introduction

The DELQA-PLUS hardware consists of a DELQA board with ROM (read-only memory) firmware of revision at least 2.0.0. This firmware allows the DELQA-PLUS board to operate in both a DELQA-normal mode and a new mode, called Turbo mode or DELQA-T mode.

The firmware resides in ROMs that reside on the DELQA-PLUS board. To determine your DELQA-PLUS board's ROM firmware version, follow the instructions in Appendix D of this *Addendum to DELQA User's Guide.*

### 6.1.1 Terminology

**DELQA-T mode and DELQA-normal mode** — For convenience, we have described the two modes in which the DELQA-PLUS board can operate as two different boards. When the DELQA-PLUS board is operating in Turbo mode, we call it a DELQA-T board or say that it is operating in DELQA-T mode. When the DELQA-PLUS board is not operating in Turbo mode, we call it a DELQA-normal board or say that it is operating in DELQA-normal mode.

**Device driver software** — Although it is possible to operate individual features of the DELQA-PLUS board from any software that has access to the Q-bus on which the DELQA-PLUS board resides, this chapter describes programming the DELQA-PLUS board in terms of a how a single device driver would operate the DELQA-PLUS board in an orderly way when the DELQA-PLUS board is in DELQA-T mode.

**Multiple DELQA-PLUS boards on a system** — There can be more than one DELQA-PLUS board on the same Q-bus. For convenience, this chapter always refers to the first DELQA-PLUS board on a two-DELQA-PLUS-board Q-bus system (unless otherwise noted). For more information about running more than one DELQA-PLUS board on a Q-bus, see Section 5.3 in this *Addendum to DELQA User's Guide.*

### 6.1.2 Overview of DELQA-PLUS Functions

The DELQA-PLUS board when in DELQA-T mode performs the same data transfer functions as the original DELQA board (called the DELQA-normal board)—it transfers network message data between the host's memory and the network to which the board is connected. The DELQA-PLUS board in DELQA-T mode also provides added functionality to that of the DELQA-normal board. You can still use all the DELQA-normal board features; in fact, you must use them in order to cause the DELQA-PLUS board to run in DELQA-T mode.

The differences between the DELQA-normal and the DELQA-PLUS boards are:

- The DELQA-PLUS board offers a superset of the functionality of the original DELQA board.
- In DELQA-T mode, the DELQA-PLUS board has a higher throughput rate than in DELQA-normal mode.
- The programming interface to the DELQA-T board is simpler than that of the DELQA-normal board—ownership of the transmitted and received data is unambiguous.
- In DELQA-T mode, the DELQA-PLUS board does not run the DECnet Maintenance Operations Protocol (MOP) on-board; it does support MOP in DELQA-normal mode.

In addition, ROM version 2.0.0 adds the following functionality to the DELQA-PLUS board's DELQA-normal mode:

- In DELQA-normal mode, the DELQA-PLUS board will transmit on the network a DECnet system ID message that contains the correct device ID for DELQA-T boards. (This device ID is 75 (decimal).)
- There are two new on-board registers, XCR0 and XCR1, which allow host software to cause the DELQA-PLUS board to go into DELQA-T mode.

- Setting the boot password to the value zero allows only passwords of value zero (0) to cause the DELQA-PLUS board to reboot its host, not all passwords to reboot it.

The information in the *DELQA User's Guide* applies to the DELQA-PLUS board when it is in DELQA-normal mode; Chapter 3 of the *DELQA User's Guide* describes how to program a DELQA-normal board.

### 6.1.3 What Does the Device Driver Do?

The device driver for the DELQA-T board resides in host memory. It communicates with the DELQA-T board through a series of registers that reside on the DELQA-T board.

Briefly, to operate the DELQA-T board, the driver must provide a set of transmit and receive buffers, then order the board through one of the registers to read or write data to and from these buffers. (The buffers reside in host memory, and the registers reside on the board; the registers appear in Q-bus memory space.) The driver also performs other operations, such as starting and stopping the DELQA-T board, as needed.

In more detail, the normal sequence of events that a device driver follows in using the DELQA-T board is:

1. Orderly start-up— Verify ROM version, select DELQA-T mode, and start the board running.
2. Perform transmit and receive operations as required and modify operating parameters as needed.
3. Orderly shutdown—Stop board and return to DELQA-normal mode.

This sequence of events can be broken down into essential tasks; these tasks are listed in the next section.

### 6.1.4 Summary of Driver's Major Tasks

The major tasks that a device driver for the DELQA-T board can perform are:

- Select DELQA-T mode—Prepares the DELQA-PLUS board to function as a DELQA-T board rather than a DELQA-normal board.
- Start the DELQA-T board—Enables the DELQA-T board to perform data transfer operations on the network.

- Transmit—The transmit operation involves three main steps: 1) setting up the data to be transferred, 2) notifying the DELQA-T board that the data is ready (the DELQA-T board then performs the transfer), and 3) checking status information afterwards.
- Receive—Like the transmit operation, the receive operation involves three main steps: 1) providing buffers for the DELQA-T board to write received data, 2) notifying the the DELQA-T board that the buffers are ready (the DELQA-T board then performs the receive), and 3) checking status information afterwards.
- Stop the DELQA-T board—Disables the DELQA-T board from performing data transfer operations.
- Software reset of the DELQA-PLUS board—Moves the DELQA-T board to DELQA-normal mode.
- Change the DELQA-T board's operating parameters—Allows the driver to substitute a new init block to the DELQA-T board; requires the driver to stop and the restart the DELQA-T board.
- Interrupts—The DELQA-T board can notify the driver by means of an interrupt that an operation is complete.
- Block interrupts from the DELQA-T board—Causes the DELQA-T board to save interrupts but not send them to the driver until the driver issues an unblock request.
- Unblock interrupts from the DELQA-T board—Allows the DELQA-T board to resume sending interrupts to the driver.
- Return to DELQA-normal mode—Moves the DELQA-T board to DELQA-normal mode in an orderly fashion.

These tasks can be arranged into a state diagram, as shown in Figure 6–1.

The following sections explain in detail how the driver performs each of the above-mentioned tasks. Each section describes the task briefly, then lists the steps the driver must perform to accomplish the task, then shows the steps in the form of a time-sequence diagram.

### 6.1.5 The DELQA-T Board's Address

The DELQA-T board resides in Q-bus memory space. Q-bus addresses are 22 bits long. (The software that operates the DELQA-T board may be required to address the DELQA-T board differently depending on the mapping hardware of the host machine.)

In this addendum, we refer to the address of each DELQA-T board as BASE, whether it is the first or the second board on the system. To find out the BASE address of your DELQA-T board, see the setting of switch S1 on the board and read the address listed in Table 5–1.

### 6.1.6 Summary of Data Structures

To perform the tasks listed in Section 6.1.4, the device driver uses a number of important data structures. The data structures are the:

- Init block
- Transmit descriptor ring
- Transmit buffers
- Receive descriptor ring
- Receive buffers

These structures are shown in Figure 6–2 and are described in detail in Section 6.14.

### 6.1.7 Summary of DELQA-T Registers

The driver also uses the DELQA-T board's on-board registers to perform the tasks listed in Section 6.1.4. These registers are:

- The ARQR, SRQR, ICR, IBAH, and IBAL which the driver uses to talk to the board, and
- The SRR, which the board uses to talk to the driver.

These registers are shown in Figure 6–2 and are described in detail in Section 6.13.

## Figure 6–1: State Diagram of DELQA-PLUS Board

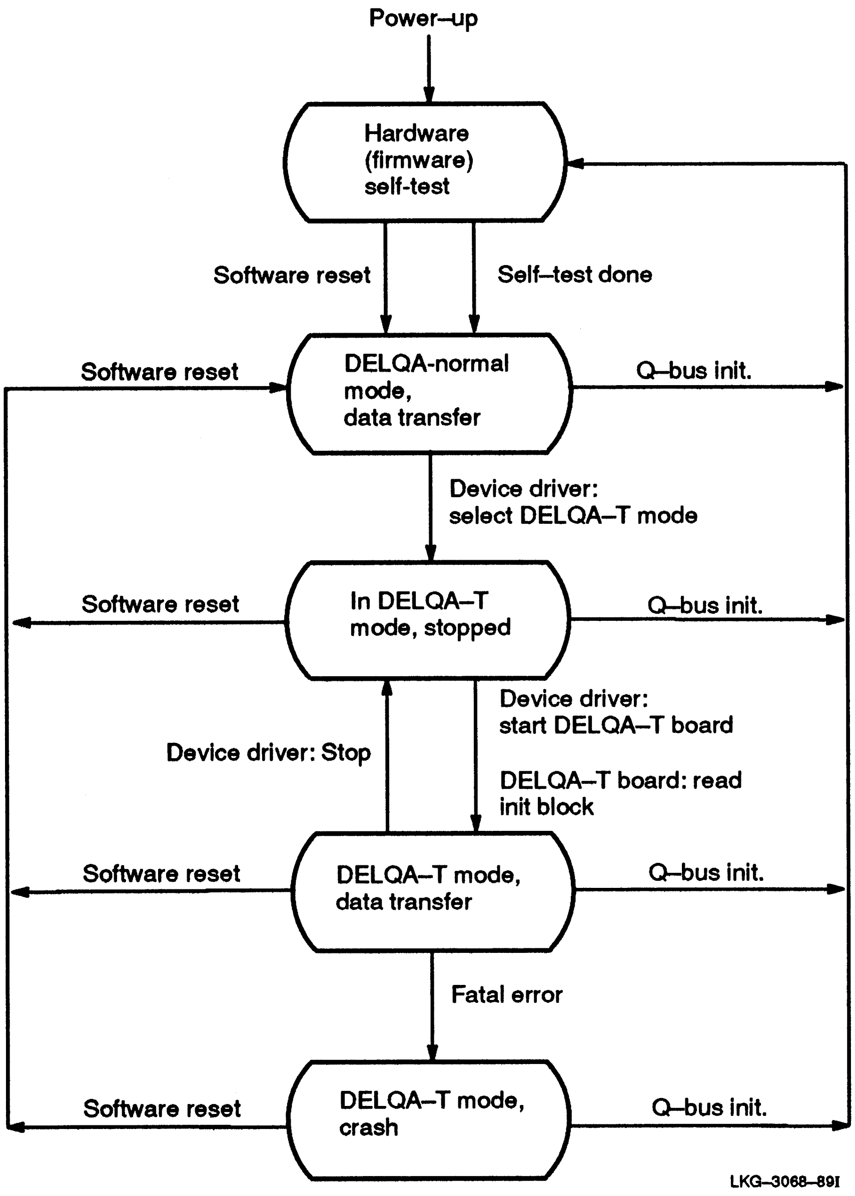
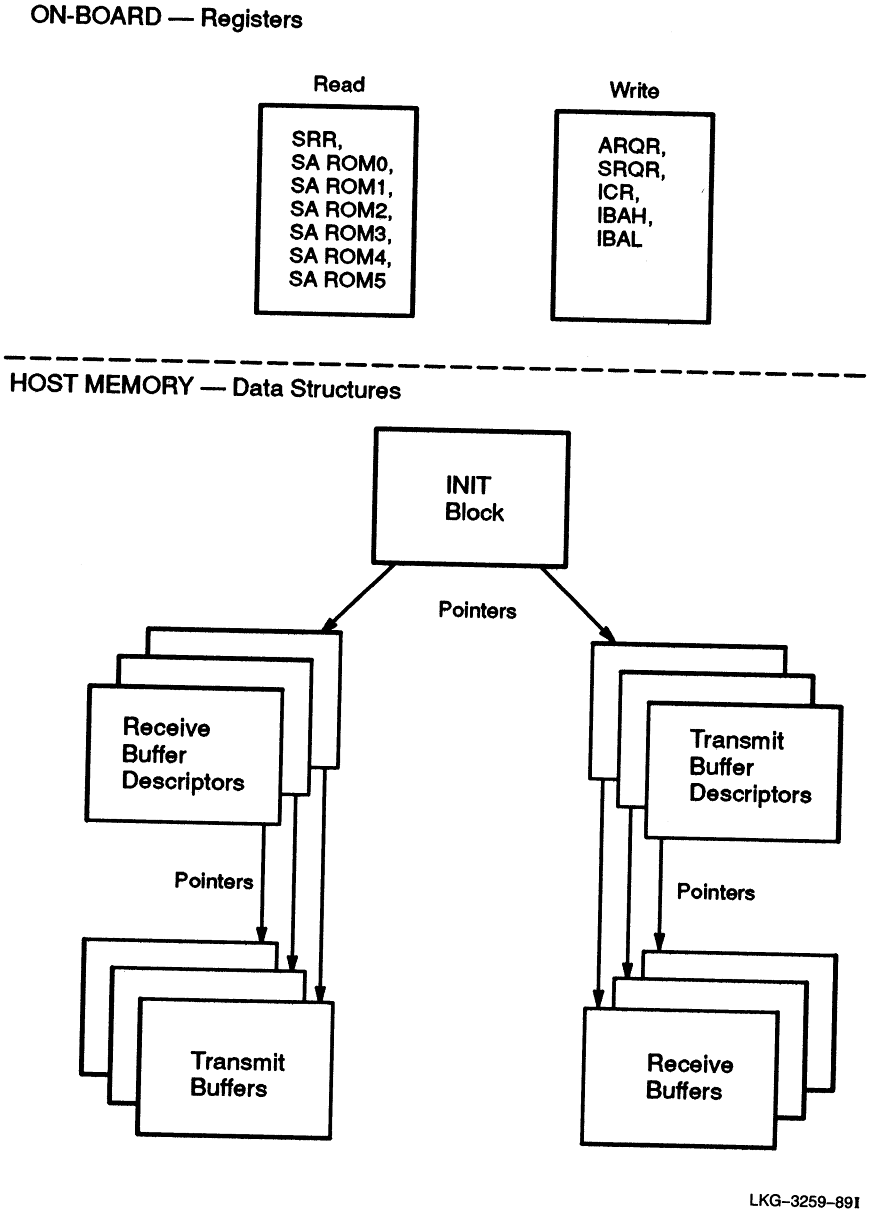


Figure 6–2: DELQA-T Registers and Host Memory Data Structures



## 6.2 Select DELQA-T Mode

This section describes the driver's task of selecting DELQA-T mode.

### 6.2.1 Select DELQA-T Mode—Description

This operation consists of moving the DELQA-PLUS board from DELQA-normal mode to DELQA-T mode.

Beforehand, you must perform the basic installation tasks that are described in Chapter 5 of this *Addendum to DELQA User's Guide.*

After you have installed the DELQA-PLUS board, follow the steps listed below. Figure 6–3 shows these steps in diagram form.

### 6.2.2 Select DELQA-T Mode—Steps

To move an installed DELQA-PLUS board from DELQA-normal mode to DELQA-T mode, the device driver follows these steps:

1. The driver constructs a block of initialization data (called the init block) in host memory.

   For complete information on the structure and contents of init blocks, see Section 6.14.3.

   Remember, the DELQA-T board will not actually read the init block until after the driver issues a start request.

2. The driver gives the DELQA-PLUS board the order to change from DELQA-normal mode to DELQA-T mode by writing the following:

   a. 0BAF (hexadecimal) to the XCR0 register.

   b. Then FF00 (hexadecimal) to the XCR1 register.

   The XCR0 and XCR1 registers reside on the DELQA-normal board at Q-bus addresses BASE + 0 and BASE + 2, respectively, where BASE is the Q-bus address of the DELQA-PLUS board.

3. The driver reads the response field (bits 01 and 00) in the SRR (status and response register) for the value 01 (binary). The value 01 (binary) means the DELQA-PLUS board has moved from DELQA-normal mode to DELQA-T mode. (The SRR register resides at the same address on the DELQA-T board as the VAR register does on the DELQA-normal board, which is Q-bus address BASE + 14 (octal). The driver may read the SRR as often as desired.)

If the RESP field (bits 01-00) in the SRR register does not contain the value 01 (binary) within 1 second, then the DELQA-PLUS board has failed to go into DELQA-T mode. Check the previous steps in this list, especially the settings of the on-board switches.

4. The driver informs the board of the location of the init block by writing the host memory address of the init block to the IBAL and IBAH registers. The IBAL and IBAH registers reside on the DELQA-T board; their Q-bus addresses are also BASE + 0 and BASE + 2, respectively.

   Remember, the DELQA-T board has a different Q-bus address depending on whether it is the first or second board on the Q-bus system. For information on the DELQA-T board's two Q-bus addresses, see Chapter 5 of this *Addendum to DELQA User's Guide*.

5. Since the DELQA-T board comes up in stopped mode, the driver must start the DELQA-T board running. To do so, the driver follows the steps in Section 6.3.

   Note that the DELQA-T board does not actually read the init block until this point.

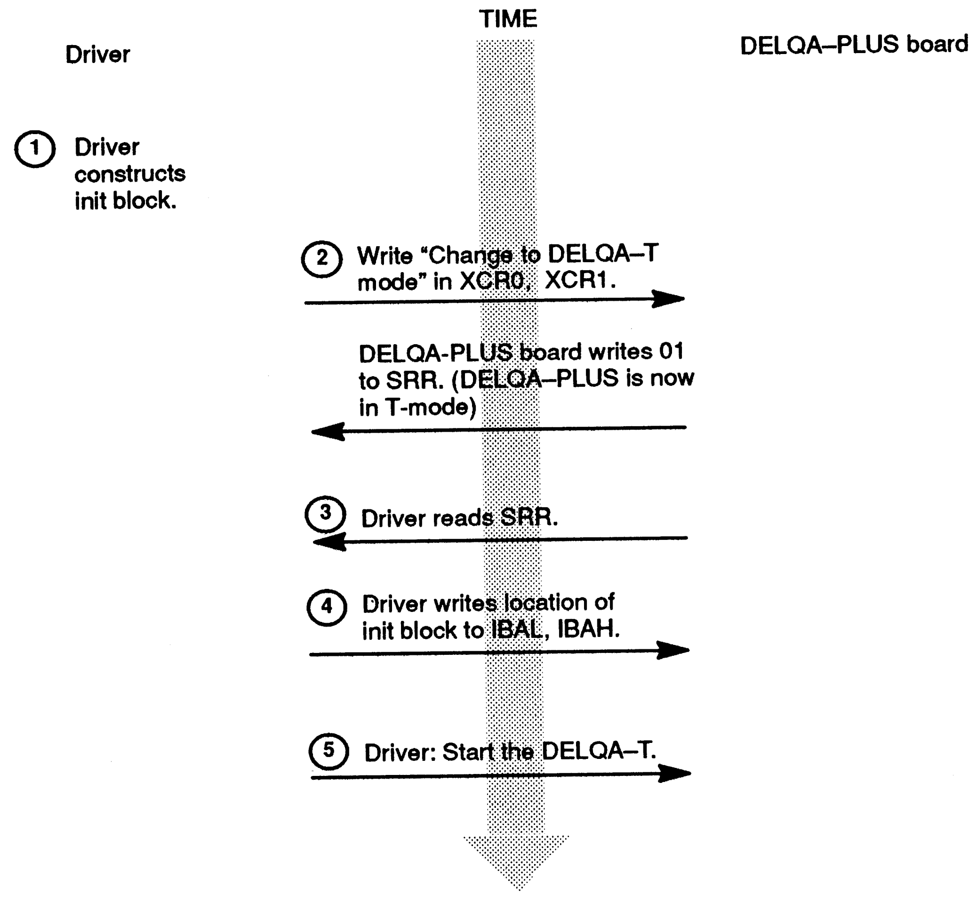
   After a successful start, the DELQA-T board is ready to transmit and receive network message data.

For information on how the driver conducts transmit and receive operations with the DELQA-T board, see Section 6.4 and Section 6.5.

**NOTE**

> After the DELQA-T board is up and running, you should enable the host inactivity timer (HIT). (The DELQA/DELQA-T board is shipped with the timer initially disabled.)
>
> To enable the HIT, follow the directions in Section 6.2.3.

Figure 6–3: Select DELQA-T Mode—Diagram


TIME
Driver
DELQA–PLUS board
1 Driver constructs init block.
2 Write "Change to DELQA–T mode" in XCR0, XCR1.
DELQA-PLUS board writes 01 to SRR. (DELQA–PLUS is now in T-mode)
3 Driver reads SRR.
4 Driver writes location of init block to IBAL, IBAH.
5 Driver: Start the DELQA–T.

LKG–3100–89I

### 6.2.3 The Host Inactivity Timer (HIT)

The purpose of the HIT is to put the DELQA-T board back into DELQA-normal mode when the driver has not communicated with the DELQA-T board for a certain amount of time. When the driver does not communicate with the DELQA-T board, the DELQA-T board assumes the host has crashed and cannot reboot itself. By going back to DELQA-normal mode, the DELQA-PLUS board can now respond to DECnet Maintenance Operations Protocol (MOP), which means the DELQA-PLUS board can respond to a request from a remote user to reboot the host machine.

For more information on rebooting, see the *DELQA User's Guide*.

#### 6.2.3.1 Setting the Host Inactivity Timer (HIT)

You enable and set the HIT differently, depending on when you do it, either before or after the DELQA-PLUS board has begun to operate in DELQA-T mode.

- Before the DELQA-PLUS board powers up, you must make sure that switch S4 (Enable/Disable HIT) on the DELQA-PLUS board is OPEN.
- After the DELQA-PLUS board powers up but before it has begun to operate in DELQA-T mode, the driver must:
  1. Set the HIT field (bit 01) in the OPTION field (BASE + 22 (octal)) in the init block.
  2. Set the time interval after which the HIT should go off by setting the HIT timeout value (BASE + 34 (octal)) in the init block to the desired time interval, in seconds.

Remember, after the DELQA-T board is running, set the HIT timer by having the driver enable it in the init block. Do not set it by throwing switch S4 on the board. To throw switches at that point, you must power down the board, which will erase the configuration of the DELQA-T board you've done so far.

#### 6.2.3.2 The HIT Timeout Value

Before the driver begins to operate the DELQA-PLUS board as a DELQA-T board (and assuming switch S4 is OPEN), the HIT timeout value will be 3 minutes by default.

After the driver gives the DELQA-PLUS board the command to start running in DELQA-T mode, the HIT timeout value will be the timeout value in the init block.

#### 6.2.3.3 How the HIT Timer Works

The HIT timer will expire if the driver does not write to either the DELQA-T-resident Synchronous Request Register (SRQR) or Asynchronous Request Register (ARQR) for the time limit specified in the HIT timeout value field in the init block. (The SRQR is for start and stop commands; the ARQR is for transmit and receive commands.)

When the HIT expires, the DELQA-PLUS board returns to DELQA-normal mode. The DELQA-normal board does not interrupt the host to announce it has completed this transition.

## 6.3 Start the DELQA-T Board

This section describes the driver's task of starting to operate the DELQA-PLUS board in DELQA-T mode.

### 6.3.1 Start the DELQA-T Board—Description

This operation involves the driver supplying a block of initialization data, then notifying the board of the block's location, then issuing the start request, then verifying that the board has read the block.

The DELQA-T board starts running with the following defaults:

- Starts processing at the beginning of the rings of transmit and receive buffer descriptors, regardless of where it may have stopped. (The beginnings of these rings are noted in the init block.)
- Interrupts are unblocked.

At this time (start-up) the driver can help the DELQA-T board operate more efficiently by giving the board as many receive buffers as possible (that is, by setting the ownership in each buffer's descriptor to "DELQA-T").

### 6.3.2 Start The DELQA-T Board—Steps

Figure 6–4 shows these steps in diagram form.

1. The driver constructs an init block for the DELQA-T board; the init block resides in host memory. For complete information on the structure and contents of init blocks, see Section 6.14.3.
2. The driver writes the most significant bits of the init block's host memory address to the IBAH (init block address, high-order) register.
3. The driver writes the least significant bits of the init block's host memory address to the IBAL (init block address, low-order) register.

   The IBAL and IBAH registers reside on the DELQA-T board. Their Q-bus addresses are BASE + 0 and BASE + 2, respectively.
4. The driver writes the start request (value = 10 (binary)) to the REQ field (bits 01 and 00) of the SRQR register. The SRQR register resides on the DELQA-T board at Q-bus address BASE + 12 (octal).

5. The driver reads the SRR. If there are errors, the driver handles the errors; if there are no errors, the driver responds to the status information; if there are no errors or status information, and none appear for one second, the driver times out.

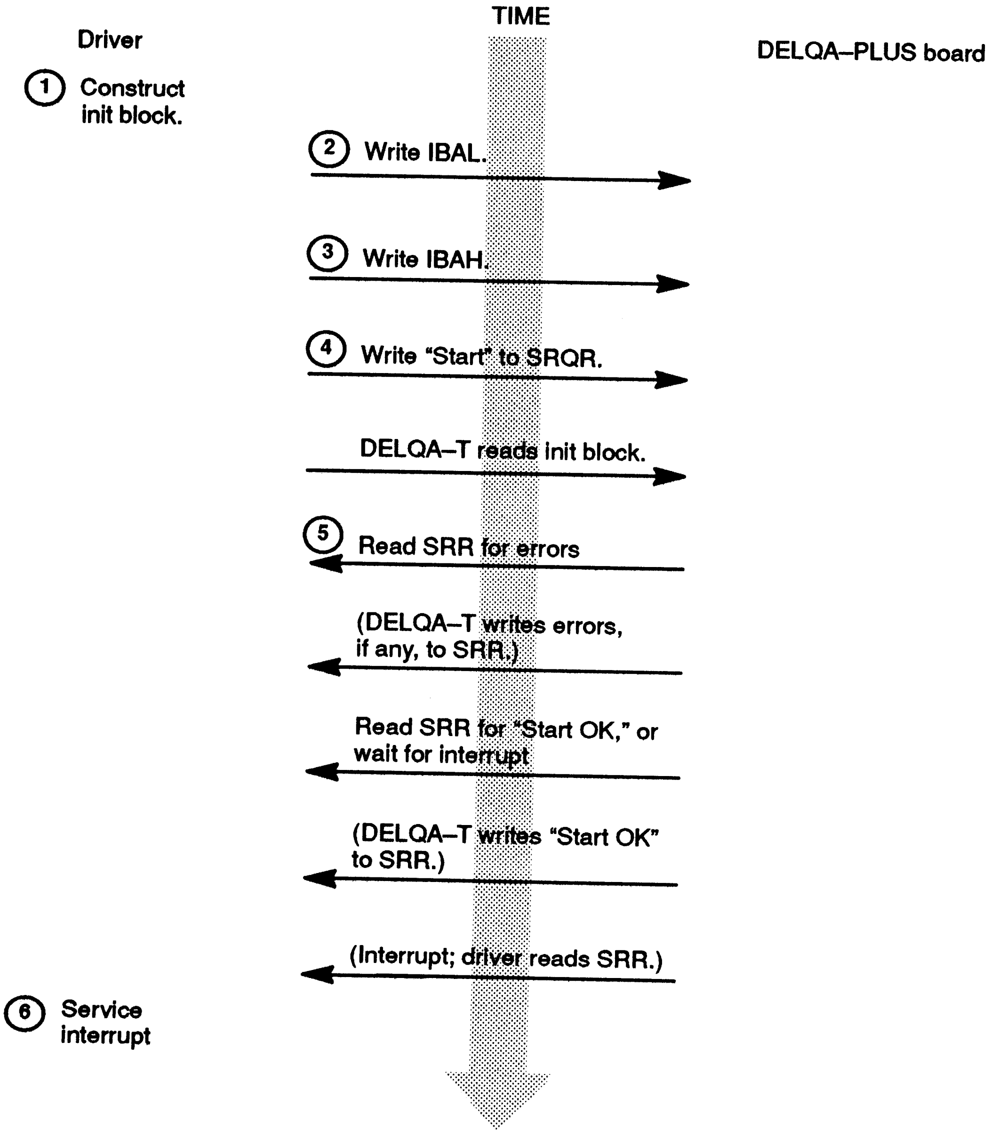
   Successful status information is: the RESP field (bits 01 and 00) in the SRR has value 10 (binary) and the FES field (bit 15) has value 0.

   If the driver times out before obtaining errors or status from the SRR, the driver may then receive an interrupt from the DELQA-T board (if interrupts are enabled; see bit 00 in the OPTIONS field in the init block).

6. After receiving an interrupt, the driver reads the SRR. If there are errors, the driver handles the errors; if there are no errors, the driver responds to the status information.

After these steps, the DELQA-T board is running and is ready to transmit and receive network message data.

## Figure 6–4: Start-Up—Diagram


TIME
Driver
DELQA–PLUS board
1 Construct init block.
2 Write IBAL.
3 Write IBAH.
4 Write "Start" to SRQR.
DELQA–T reads init block.
5 Read SRR for errors
(DELQA–T writes errors, if any, to SRR.)
Read SRR for "Start OK," or wait for interrupt
(DELQA–T writes "Start OK" to SRR.)
(Interrupt; driver reads SRR.)
6 Service interrupt

LKG–2961–89I

## 6.4 Transmit

This section describes how the DELQA-T device driver causes the DELQA-T board to transfer network message data from host memory to the DELQA-T board's internal buffer memory and from there onto the network.

### 6.4.1 Transmit—Description

Briefly, the driver gives the initial request to transmit some data by first setting the ownership of a transmit buffer descriptor to "DELQA-T" (setting bit 15 in word 3 to 0 (zero)), then writing the ARQR to notify the DELQA-T board to perform the transfer. The DELQA-T board then copies the data out of the transmit buffers and transmits it on the network.

On completion of the transmission, the DELQA-T provides a response to the driver. The driver gives this response by writing fields in the transmit buffer descriptors. If interrupts are not blocked, the DELQA-T board will also issue an interrupt after every transmitted buffer (or buffer pair—see Section 6.4.4).

### 6.4.2 How the DELQA-T Board Transmits

Whenever the driver writes the transmit request (value = 8000 (hex)) to the ARQR, the DELQA-T board will start reading the next transmit buffer descriptor in the transmit buffer descriptor ring. (The location of the transmit buffer ring is given by pointers in the init block).

Note that the next buffer may be the first buffer in the ring under the following conditions:

- This is the first transmit operation after a board startup, or
- The board has worked its way around the ring.

The DELQA-T board proceeds through the ring sequentially, reading descriptors and transmitting buffers, until it encounters a descriptor whose owner is not "DELQA-T." This could be because the DELQA-T board has come all the way around the ring and encountered the first buffer it transmitted again, or it could be that the ring had less than 12 entries to transmit in the first place.

**NOTE**

In this discussion, "buffer ring" means the buffers as pointed to by the ring of descriptors. The buffers themselves may or may not be in a ring.

The DELQA-T board then stops examining the ring, transmits the last buffer, if any, it has obtained because that buffer had its ownership set to "DELQA-T", posts status in that buffer's descriptor, and also attempts to give an interrupt.

The DELQA-T board transmits buffers that have their ownership set to "DELQA-T" (bit 15 word 3 of the buffer's descriptor), setting the ownership back to "Driver" after transmitting each buffer.

Meanwhile, the driver can be servicing the ring, reloading buffers, and setting the ownership in the buffer descriptors back to "DELQA-T" so that the DELQA-T board may never have to stop transmitting.

When the DELQA-T board encounters a pair of buffers that are chained together (see Section 6.4.4), it will transmit both buffers as one packet and give only one interrupt on completion of the transmit. (In addition to the interrupt, the DELQA-T board will also write status and ownership information for both buffers.)

### Size Restrictions—Transmitted Data

The driver must make sure the packets it transmits on the network comply with the Ethernet/IEEE 802.3 size restriction, which is that the packet must be between 64 and 1518 (decimal) bytes inclusive, including CRC.

The driver must also make sure to comply with following size restrictions for buffers:

- The driver must not transmit a buffer whose size field contains a value larger than the maximum or smaller than the minimum legal size (see Table 6–1). The DELQA-T board is not guaranteed to perform the transmit correctly if the size field (and/or the buffer) is too large or too small.

  Table 6–1 summarizes the size restrictions for all buffers that the driver can transmit via the DELQA-T board. Note that the size restrictions are complex and have effects on each other.

- When chaining buffers during transmit operations:

  1. The first buffer must be at least 100 (decimal) bytes long, and

  2. The second buffer must be at least 1 byte long but not more than 1418 (decimal) bytes long.

- The driver must never send buffers of 0 bytes in length or larger than 1518 (decimal) bytes in length.

The driver notes the size of each buffer to be transmitted in the BCT in that transmit buffer's descriptor.

**Table 6–1: Size Restrictions For Transmitted Buffers**

| If these fields are set as follows: | | | | The minimum and maximum sizes, in bytes, are: | |
|---|---|---|---|---|---|
| LOP | DTC | FOT | FOT in previous descriptor | Minimum (decimal) | Maximum (decimal) |
| 0 | 0 | 0 | 0 | 60 | 1514 |
| 0 | 0 | 1 | 0 | 100 | 1513 |
| 0 | 0 | 0 | 1 | 1 | 1414 |
| 0 | 1 | 0 | 0 | 64 | 1518 |
| 0 | 1 | 1 | 0 | 100 | 1517 |
| 0 | 1 | 0 | 1 | 1 | 1418 |
| 1 | X | 0 | 0 | 32 | 32 |

Key:

LOP—Loopback mode; resides in init block
DTC—Disable CRC on transmit; resides in init block
FOT—First-of-two buffers; resides in transmit descriptor

### 6.4.3 Transmit—Steps

Figure 6–5 shows these steps in diagram form.

1. The driver builds a descriptor; the descriptor points to a buffer that is ready to be transmitted.
2. The driver sets the ownership to "DELQA-T" (sets bit 15 of word 3 in the buffer descriptor is set to 0 (zero)).
3. The driver notifies the DELQA-T board to begin transmitting by writing a transmit request (value = 8000 (hex)) to the ARQR.

   The DELQA-T board will then begin:

   a. Sequentially reading transmit buffer descriptors,

   b. Reading each descriptor's associated buffer, and

c. Transmitting each buffer.

On completing the transmission of each buffer (or buffer pair if the buffers are chained), the DELQA-T board will:

a. Write status information to the SRR.

b. Write status information into the buffer descriptor.

c. Set the ownership in the buffer descriptor back to "Driver."

d. If the DELQA-T board's interrupts are currently unblocked, interrupt the driver.

4. The driver must now obtain the transmit-complete information from the board, which exists in the above-mentioned forms. The suggested procedure is:

   a. The driver reads the SRR.

      If there are errors, the driver handles the errors.

   b. If there are no errors, the driver checks the appropriate buffer descriptors for the following information:

      - Ownership is set back to "Driver."
      - Status information about the transfer.

5. The driver can now re-use the descriptor for subsequent transmits.

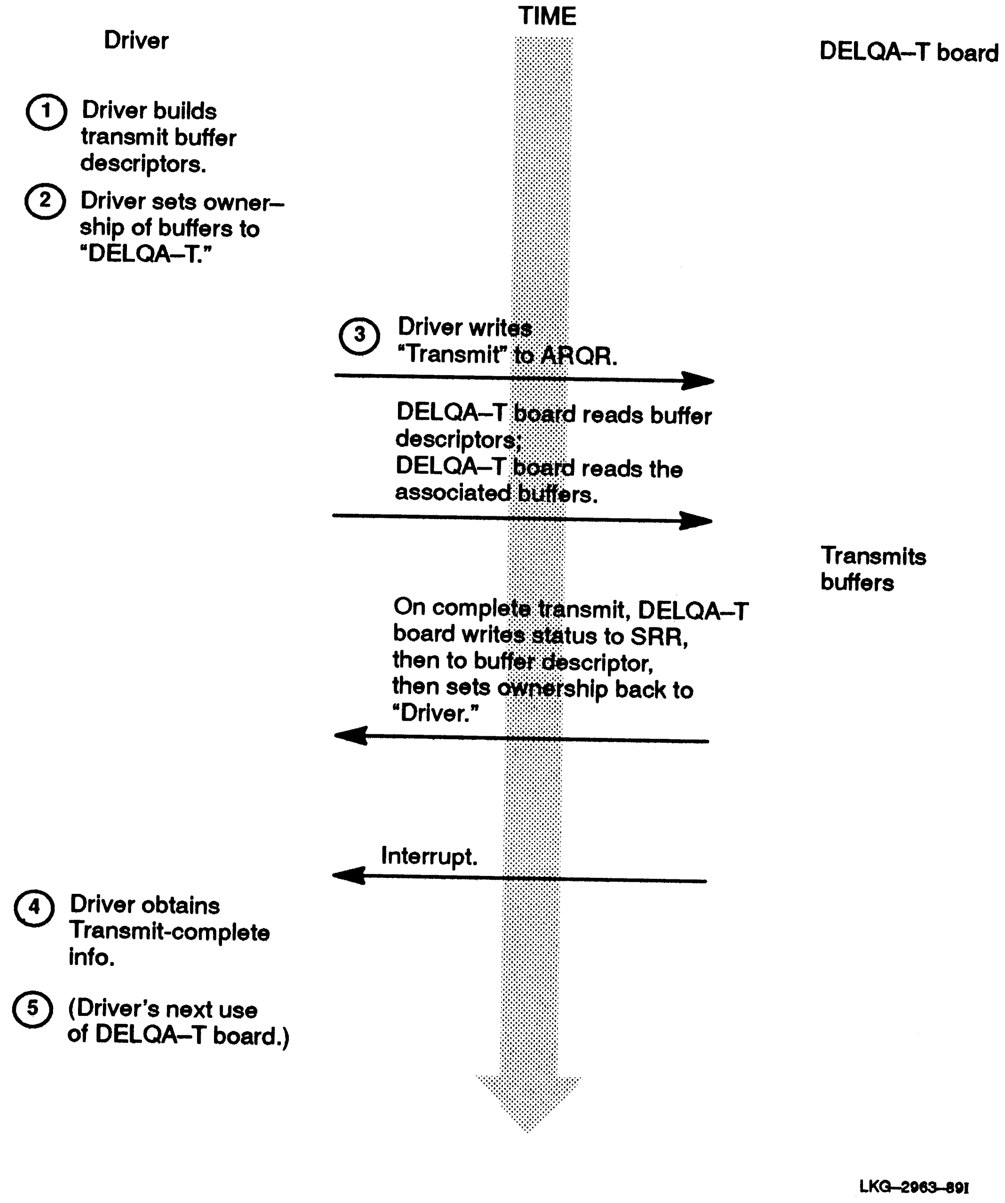
### 6.4.4 Chaining Buffers

Chaining is useful for allowing higher-level software to pass the driver a packet as two buffers; for example, the packet header travels in one buffer and the packet data travels in the other. Chaining allows this packet to be reunited and transmitted as a unit.

To chain two transmit buffers together, the driver must

1. Set the FOT (first-of-two) field (bit 14 in word 3) in the first transmit buffer's descriptor to 1.

Figure 6–5: Transmit—Diagram


TIME
Driver
DELQA–T board
1 Driver builds transmit buffer descriptors.
2 Driver sets owner– ship of buffers to "DELQA–T."
3 Driver writes "Transmit" to ARQR.
DELQA–T board reads buffer descriptors;
DELQA–T board reads the associated buffers.
Transmits buffers
On complete transmit, DELQA–T board writes status to SRR, then to buffer descriptor, then sets ownership back to "Driver."
Interrupt.
4 Driver obtains Transmit-complete info.
5 (Driver's next use of DELQA–T board.)
LKG–2963–89I

2. Make sure the FOT field in the second (that is, the next) transmit buffer's descriptor to 0.

3. Give the buffers to the board in reverse order, chronologically. ("Give" here means set the ownership.)

This guarantees that the DELQA-T board will own both buffers by the time it must transmit them rather than possibly having to wait for the second buffer. If the driver does not follow this procedure, the DELQA-T board is not guaranteed to transmit chained buffers correctly.

This will cause the DELQA-T board to transmit both buffers as one packet before reporting completion. The DELQA-T board will transmit the buffers in correct order, that is, the first buffer at the beginning of the packet and the second buffer at the end of the packet.

**NOTE**

When chaining, it is the driver's responsibility to make sure the DELQA-T board does not transmit oversize packets on the network.

## 6.5 Receive

This section describes the driver's task of receiving data using the DELQA-T board.

### 6.5.1 Receive—Description

Receiving a packet involves transferring the packet from the network to the DELQA-T board's internal buffer memory and from there to host memory.

Even if the driver doesn't ask the DELQA-T board to receive, the DELQA-T board will begin collecting packets from the network that are addressed to the local node as soon as the the DELQA-T board completes startup and starts running.

The DELQA-T board will buffer up to to 16 (decimal) maximum-size Ethernet/IEEE 802.3 packets before it begins discarding further incoming packets.

The DELQA-T board will operate more efficiently if the driver gives the DELQA-T board as many receive buffers as possible.

**NOTE**

> It is not necessary for the driver to get a receive response from DELQA-T board before the driver issues further receive requests. The driver can have multiple unacknowledged receive requests ready on the receive ring.

### 6.5.2 Receive—Steps

To receive a buffer of data from the DELQA-T board, the driver follows these steps. Figure 6–6 shows these steps in diagram form. (The steps assume the DELQA-T board has data that it has received from the network.)

1. The driver sets the ownership in the buffer descriptor to "DELQA-T."
2. The driver notifies the DELQA-T board to begin receiving, that is, begin writing received data into the receive buffers. To do this, the driver writes the receive request (value = 8000 (hex)) to the ARQR register.

   The DELQA-T board will then begin:

   a. Reading buffer descriptors, and

   b. Filling each descriptor's associated buffer with received data.

On filling each buffer, the DELQA-T board will

  - Write status information to the SRR.
  - Write status information into the buffer descriptor.
  - Set the ownership in the descriptor to "driver" (DELQA-T board sets bit 15 of word 3 of the receive buffer descriptor to 1).
  - If interrupts are unblocked, it will then interrupt the driver.

3. The driver must now obtain the receive-complete information from the board, which exists in several forms. The suggested procedure is:

   - The driver reads the SRR.
     - If there are errors, the driver handles the errors.
     - If there are no errors, the driver checks the appropriate buffer descriptors for the following information:
       - Ownership is set back to "Driver."
       - Status information about the transfer.

   After establishing that the receive proceeded correctly, the driver can give the buffer to the user software/higher-level software. The driver can then re-use the descriptor for subsequent receives.

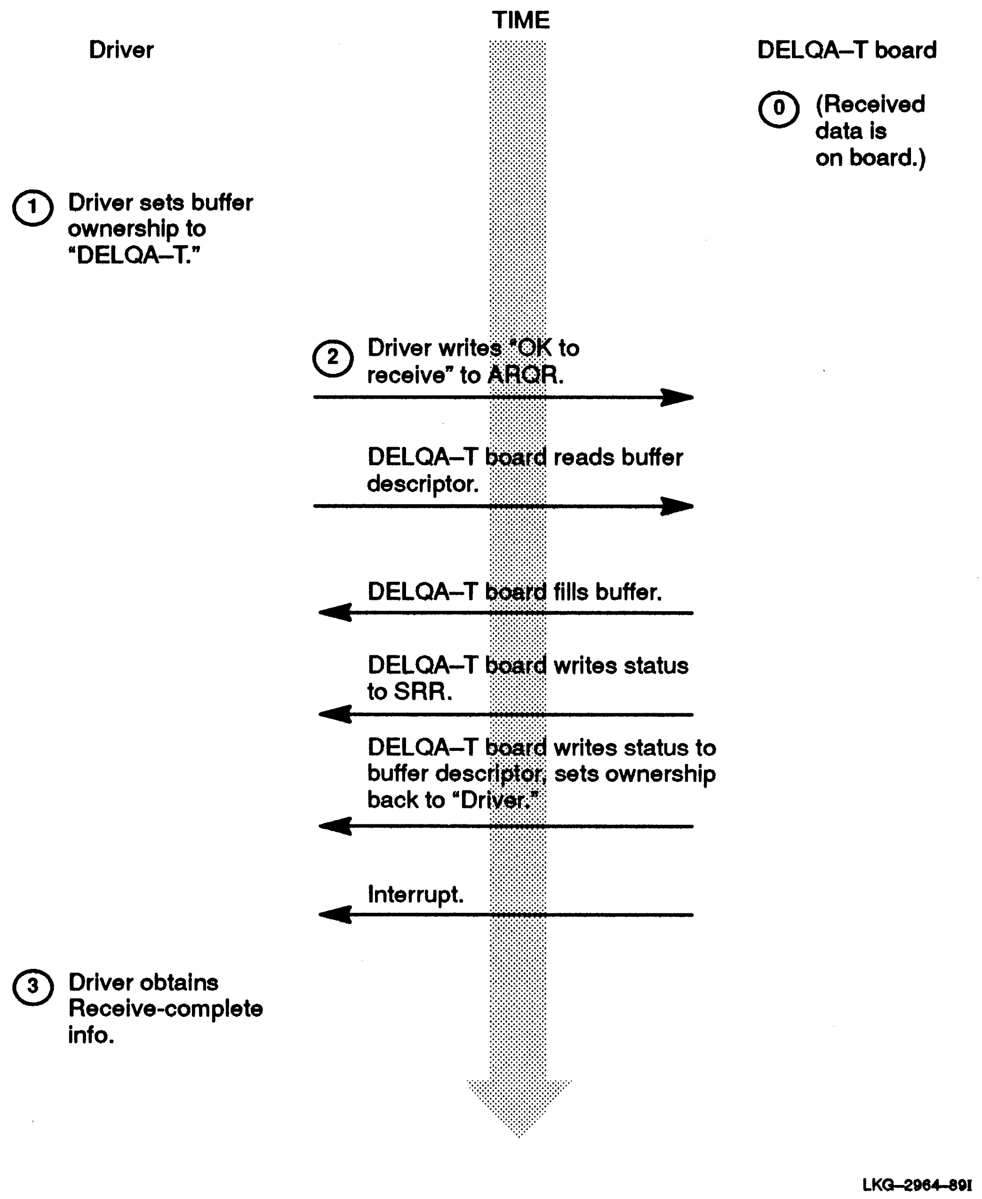
### 6.5.3 Size Restrictions—Received Data

If the DELQA-T board receives an illegal (oversize) packet from the network, the DELQA-T board will split the packet across two or more buffers, taking as many receive buffers as it needs to contain the packet. But the driver must discard these buffers since they contain an illegally large packet.

To determine whether a buffer contains part of an oversize packet or not, the driver should read the STP and ENP fields in the receive buffer descriptor:

- Normally there is one packet per buffer, so the DELQA-T board sets both these fields.

## Figure 6–6: Receive—Diagram


TIME
Driver
DELQA–T board
0 (Received data is on board.)
1 Driver sets buffer ownership to "DELQA–T."
2 Driver writes "OK to receive" to ARQR.
DELQA–T board reads buffer descriptor.
DELQA–T board fills buffer.
DELQA–T board writes status to SRR.
DELQA–T board writes status to buffer descriptor, sets ownership back to "Driver."
Interrupt.
3 Driver obtains Receive-complete info.
LKG–2964–89I

- But if the DELQA-T board receives an oversize packet, the DELQA-T board sets only one or neither of these fields, as follows:
  - Sets STP in the descriptor that points to the buffer that contains the beginning of the packet.

- Sets ENP in the descriptor that points to the buffer that contains the end of the packet.

- Sets neither STP or ENP in descriptors that point to buffers that contain intermediate pieces of the packet.

If the driver sees a receive buffer descriptor with only the STP set, only the ENP set, or neither set, it should discard the buffer since it contains a piece of an oversize packet.

See Section 6.14.6 for more information on the ENP and STP fields.

## 6.6 Stop the DELQA-T Board

This section describes the driver's task of stopping the DELQA-T board.

### 6.6.1 Stop The DELQA-T Board—Description

Stopping the DELQA-T board is useful in the following situations:

- No users at the host are using the network.
- The driver wants to perform an orderly shutdown of the DELQA-T board.
- The driver wants to restart the DELQA-T board with a new init block.
- The driver wants to change the parameters of a transmit or receive buffer descriptor.

After the DELQA-T board stops, it will perform no processing except for HIT (host inactivity timer) timeouts. For more on the HIT, see Section 6.2.3.

The driver may, however, change the ownership of buffer descriptors while the DELQA-T board is stopped. This is useful if the driver wishes to the recover buffers that the DELQA-T board owns.

**NOTE**

> The driver should not issue a stop request while it has outstanding transmits and receives since there may be some loss of outstanding (that is, to-be-transmitted) data.

### 6.6.2 Stop the DELQA-T Board—Steps

To stop the DELQA-T board, the driver follows these steps:

1. The driver writes the stop request (11 (binary)) to the REQ field (bits 01-00) in the SRQR register.
2. The driver must not request any more transmits or receives to and from the DELQA-T board after this step.

   The DELQA-T board stops all transmit and receive operations.

   The DELQA-T board will set the RESP field (bits 01-00) in the SRR to 11 (binary).

3. The driver must now obtain the stop-complete information from the DELQA-T board, which exists in several forms. The suggested procedure is:

    a. Check the SRR for errors.

    b. Check the RESP field in the SRR for the stop response.

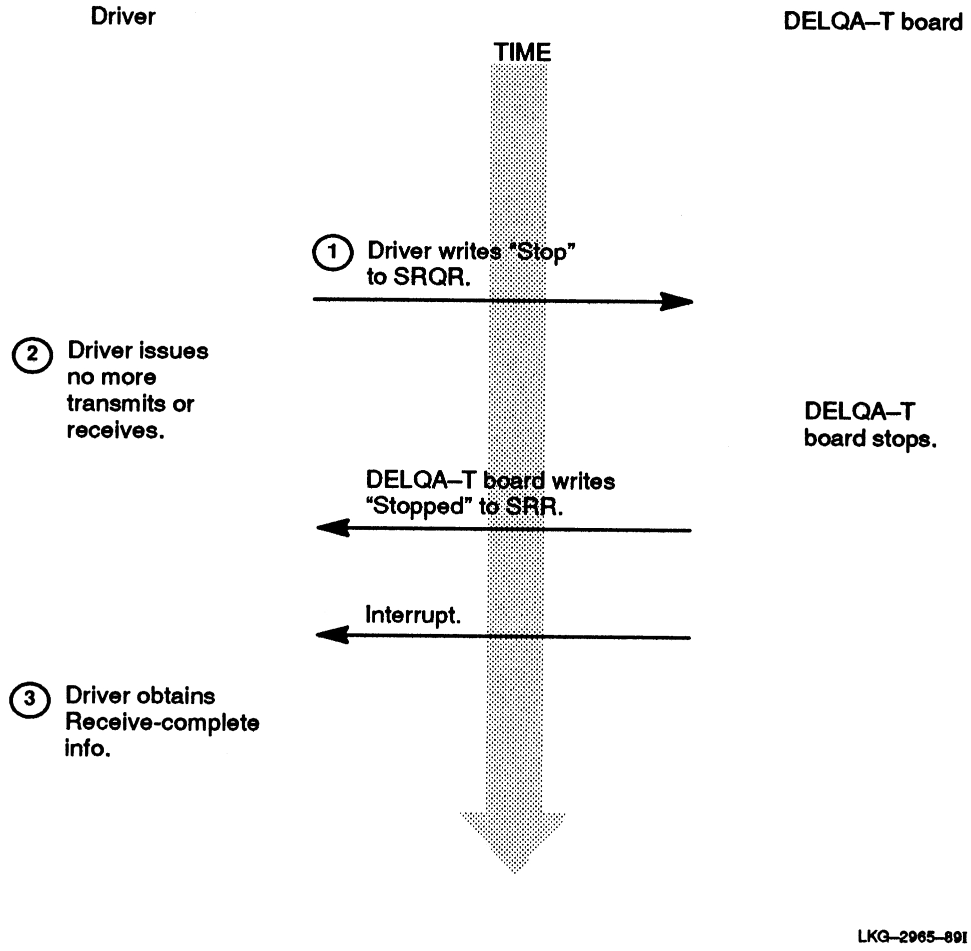
    c. Wait for an interrupt, timeout after 1 second; after the interrupt, check the SRR for errors, then check the SRR for the stop response.

    The driver may either process or discard status information about outstanding transmits and receives that the DELQA-T board returns during this period.

After the DELQA-T board is stopped, it will not perform any more data transfer operations nor do any processing, except if the HIT is enabled and expires, until the driver restarts the board.

For more information on the HIT, see Section 6.2.3. Figure 6–7 shows the above steps in diagram form.

## Figure 6–7: Stop—Diagram


Driver
DELQA–T board
TIME
1 Driver writes "Stop" to SRQR.
2 Driver issues no more transmits or receives.
DELQA–T board stops.
DELQA–T board writes "Stopped" to SRR.
Interrupt.
3 Driver obtains Receive-complete info.
LKG–2965–89I

## 6.7 Software Reset of the DELQA-PLUS Board

This section describes the driver's task of doing a software reset of the the DELQA-PLUS board.

### 6.7.1 Software Reset of the DELQA-PLUS Board—Description

This operation consists of moving the DELQA-PLUS board from whatever mode it is in to DELQA-normal mode. From there, the DELQA-PLUS board can be returned to DELQA-T mode.

Resetting the DELQA-PLUS board is useful if, when in DELQA-T mode, the board has not responded to the DELQA-T device driver for more than one second, or if the driver wants to asynchronously return the board to DELQA-normal mode.

### 6.7.2 Software Reset—Steps

To reset the DELQA-PLUS board, the device driver must do the following:

1. Give the reset notification:
   - For a DELQA-T board:
     a. Write the value 0002 to the DELQA-T board's ARQR.

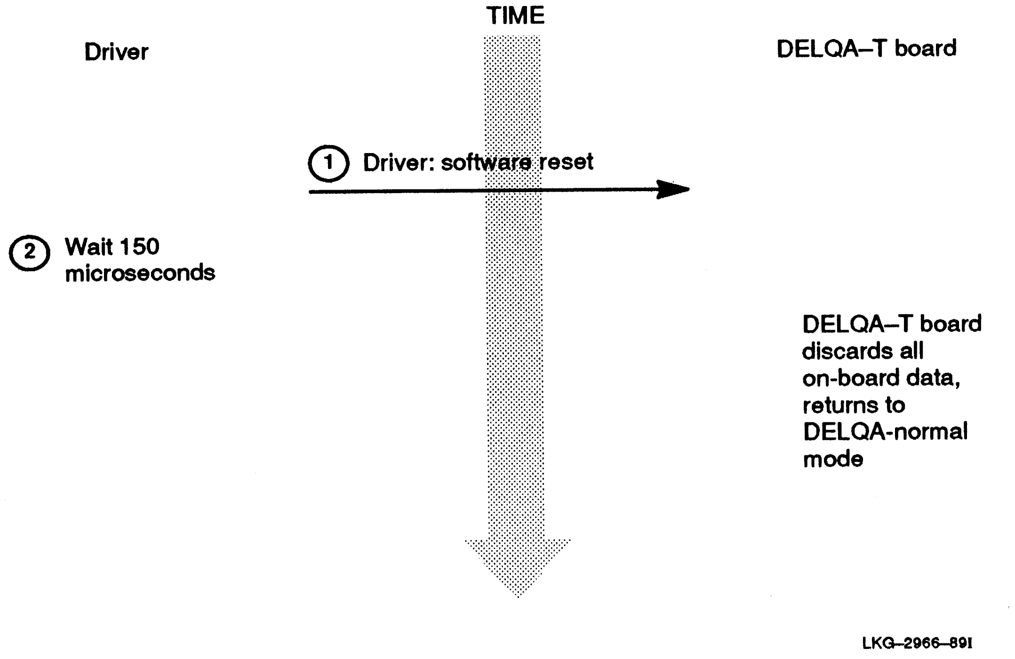
     b. Immediately write the value 0000 to the ARQR.
   - For a DELQA-normal board:
     a. Write the value 0002 to CSR (the DELQA-normal board's control and status register; this sets bit 01).

     b. Immediately write the value 0000 to CSR (clears bit 01).
2. Delay 150 microseconds.

This will put the DELQA-PLUS board into DELQA-normal mode.

Figure 6–8 shows these steps in diagram form.

## Figure 6–8: Software Reset—Diagram



## 6.8 Changing DELQA-T Operating Parameters

This section describes the driver's task of changing the DELQA-T board's operating parameters.

### 6.8.1 Changing DELQA-T Operating Parameters—Description

Changing the DELQA-T board's operating parameters is useful for various reasons, including:

- The driver needs to add or remove an address to or from the multicast filter.
- The driver needs to enable promiscuous mode.
- The driver needs to enable/disable loopback.

The only way to put these new parameters into effect is to provide a new init block in which the fields for these items are set appropriately and put the DELQA-T board through DELQA-T mode start-up again.

Section 6.8.2 lists the steps to do this. Figure 6–9 shows these steps in diagram form.

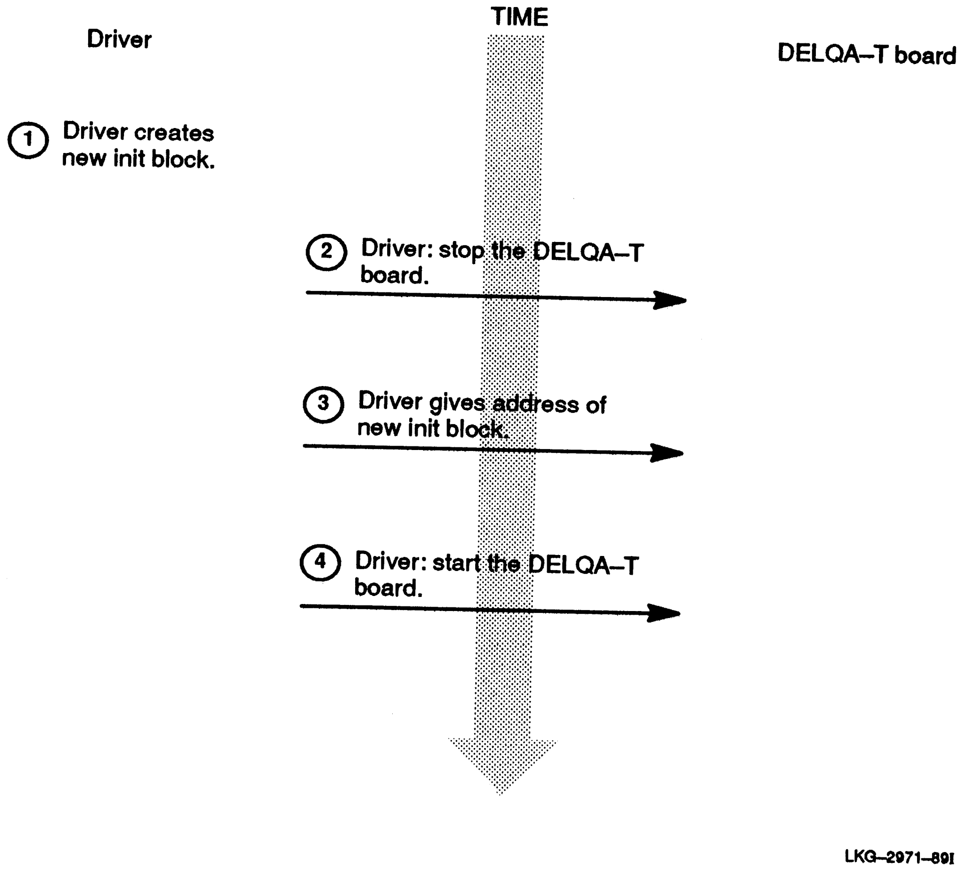
To change the parameters of a transmit or receive buffer, see Section 6.6.

### 6.8.2 Changing DELQA-T Operating Parameters—Steps

1. The driver creates a new init block in host memory.
2. The driver stops the DELQA-T board, as described in Section 6.6.
3. The driver gives the DELQA-T board the Q-bus address of the new init block by writing:
   - The most significant bits of the init block's host memory address to the IBAH register.
   - The least significant bits of the init block's host memory address to the IBAL register.

   For more information on the init block, see Section 6.14.3.
4. The driver starts the DELQA-T board, as described in Section 6.3.

## Figure 6–9: Changing DEQLA-T Operating Parameters—Diagram


TIME
Driver
DELQA–T board
1 Driver creates new init block.
2 Driver: stop the DELQA–T board.
3 Driver gives address of new init block.
4 Driver: start the DELQA–T board.
LKG–2971–89I

## 6.9 Interrupts

This section describes the driver's task of handling interrupts from the DELQA-T board.

### 6.9.1 Interrupts—Description

The DELQA-T board interrupts the host machine (and hence the device driver) using standard Q-bus interrupt procedure.

The device driver provides the interrupt vector in the INT VECTOR field in the DELQA-T init block.

The driver enables the DELQA-T board to issue interrupts on the Q-bus by setting the INT-ENABLE field (bit 00) in the OPTION field in the init block to the value 1. The DELQA-T board can then interrupt the host processor.

### 6.9.2 Blocking And Unblocking Interrupts

The action of blocking and unblocking interrupts from the DELQA-T board allows the driver to prevent itself from being interrupted while it is already servicing an interrupt.

The driver can issue blocks and unblocks even if it hasn't received an interrupt yet but is only expecting one.

- To block interrupts from the DELQA-T board, the driver writes the "block" request to the on-board ICR (interrupt control register); that is, it writes the value 0 to the ICR.

  Note that blocking causes the DELQA-T board to note interrupts but not actually to issue any interrupts until the driver unblocks interrupts. (The DELQA-T board notes all the interrupts that occur during this time by setting a single internal flag.)

- To unblock interrupts, the driver writes the "unblock" request to the ICR (writes the value 1 to the ICR).

  Immediately after unblocking interrupts, if the DELQA-T board had tried to give one or more interrupts while blocked, it will give a single interrupt now. The driver should then follow its normal interrupt service routine, which is described below.

The DELQA-T board will continue to give interrupts until the driver blocks interrupts again.

**NOTE**

> All discussions of blocking and unblocking interrupts assume the driver has initially enabled interrupts in the init block. If the driver has not enabled interrupts, blocking or unblocking them is irrelevant.

### 6.9.3 Value of Blocking and Unblocking Interrupts

The point of blocking interrupts is, overall, to have the lowest ratio of interrupts-to-packets as possible. Here's how this this can work: Normally, each transmit or receive operation can result in an interrupt. But while its interrupts are blocked, the DELQA-T board can continue to perform tasks such as transmits and receives, but it will accumulate only a single interrupt. When the driver finally gets this interrupt, the driver can then service in a single service pass all the buffers that have been transmitted or received. This single pass can be more efficient than a one-buffer-per-pass/one-buffer-per-interrupt procedure.

### 6.9.4 Interrupt Defaults At DELQA-T Startup

The DELQA-T board starts up with interrupts unblocked.

Remember, though, that if the driver has not enabled interrupts, blocking or unblocking them is irrelevant.

### 6.9.5 When Do Interrupts Occur?

The DELQA-T board will interrupt under the following conditions:

- The following interrupts will always take place, regardless of whether the driver has blocked interrupts or not:
  - The DELQA-T board completes board start-up.
  - The DELQA-T board completes board stop.
  - A fatal error occurs on the DELQA-T board.
- The following interrupts will take place unless the device driver has blocked interrupts:
  - The DELQA-T board completes transmission of data.

- The DELQA-T board completes reception of data.

### 6.9.6 Basic Interrupt Service Routine

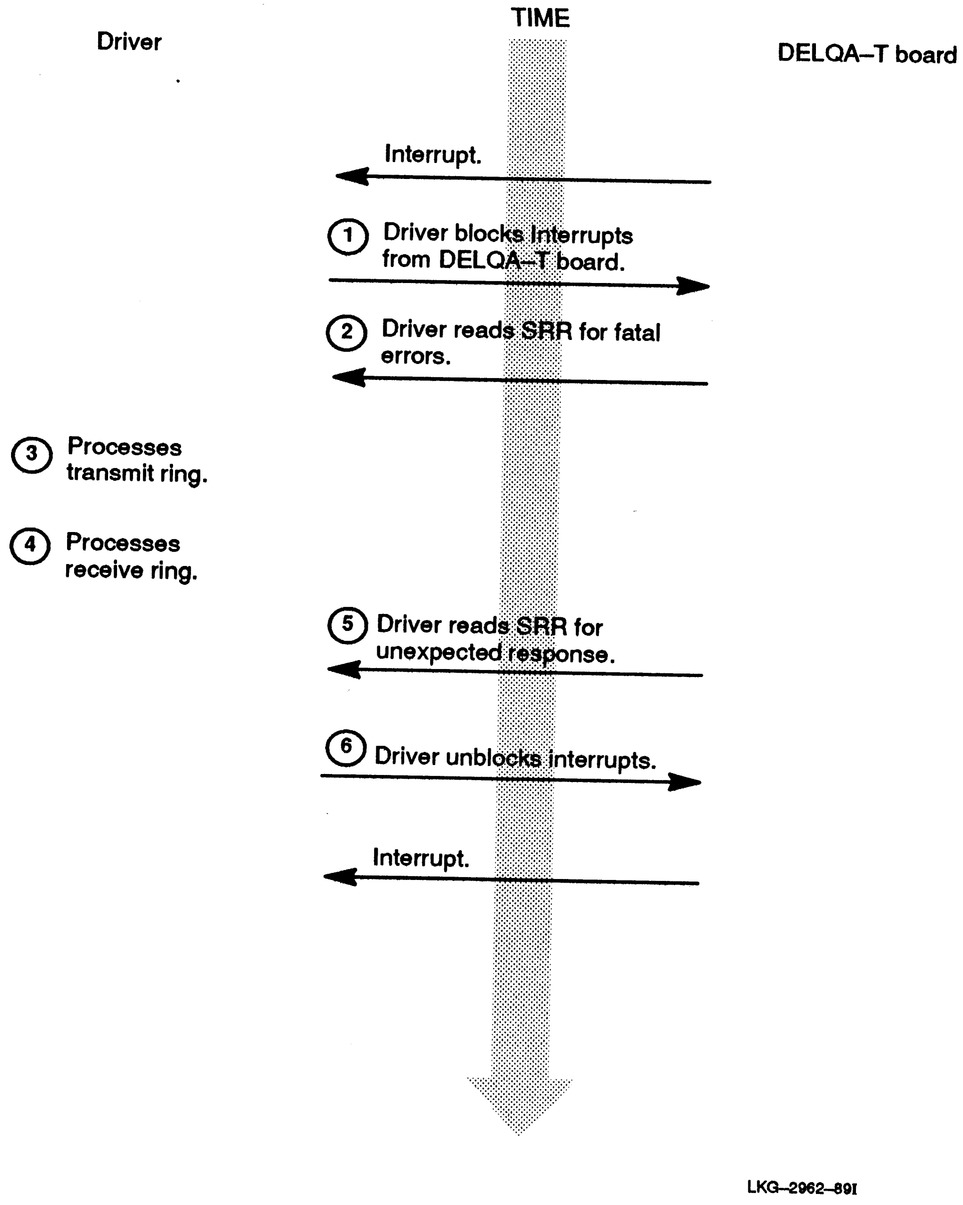
This section gives the basic steps for a standard interrupt service routine for the DELQA-T board. Figure 6–10 shows these steps in diagram form.

After it receives an interrupt from the DELQA-T board, the device driver should:

1. Block interrupts (recommended).
2. Read the SRR for fatal errors.
3. Process the transmit buffer ring for all returned buffers, that is, those that have their ownership set back to "driver." Remember, a buffer whose ownership is "driver" can also be a buffer that the driver has never given to the DELQA-T board in the first place.
4. Process the receive buffer ring for all returned entries.
5. Check the SRR for any expected response, for example, a response to a stop-device.
6. Unblock interrupts (necessary if the block was performed).

It is the driver's responsibility to keep track of the number of outstanding transmits and receives it has, and what responses to requests it is expecting (for example, if the driver has made a stop-board request which is still outstanding).

## Figure 6–10: Interrupt Service Routine—Diagram


TIME
Driver
DELQA–T board
Interrupt.
1 Driver blocks interrupts from DELQA–T board.
2 Driver reads SRR for fatal errors.
3 Processes transmit ring.
4 Processes receive ring.
5 Driver reads SRR for unexpected response.
6 Driver unblocks interrupts.
Interrupt.
LKG–2962–89I

## 6.10 Block Interrupts

This section describes the driver's task of blocking interrupts from the DELQA-T board.

### 6.10.1 Block Interrupts—Description

The action of blocking interrupts will prevent the DELQA-T board from generating further interrupts, which the board normally does after it has transmitted or received data on the network.

Notice that blocking affects some interrupts and not others. Blocking will prevent the DELQA-T board from generating the following interrupts:

- Response to a transmit request
- Response to a receive request

Blocking will not prevent the DELQA-T board from generating the following interrupts:

- Response to a start-up request
- Response to a stop request
- Indication that a fatal error has occurred on the board

During data transfer operations, when the driver can expect only interrupts from transmits and receives, which are blockable interrupts, the driver should always block interrupts immediately following an interrupt so it can process the current interrupt before being interrupted again.

**NOTE**

> It is possible to get one interrupt just after blocking if the DELQA-T board is already in the process of generating an interrupt.

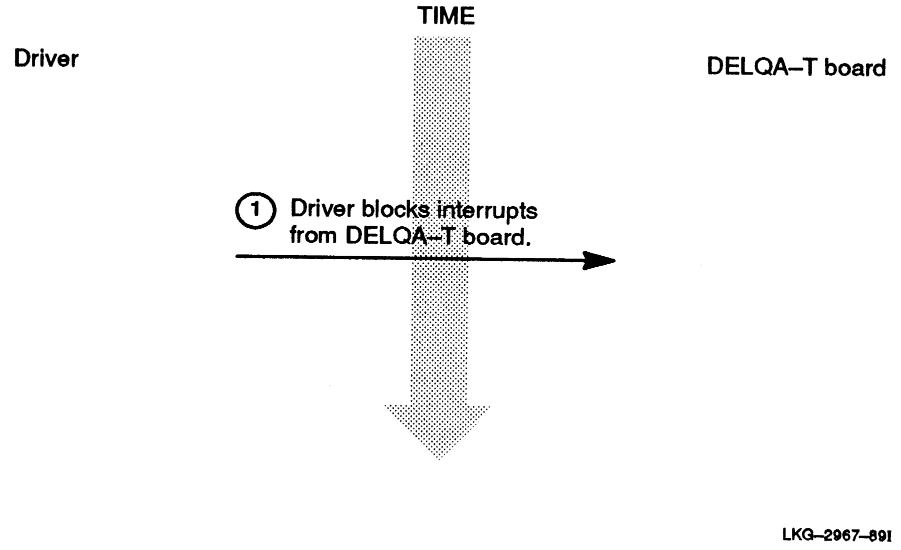
### 6.10.2 Block Interrupts—Steps

To block interrupts, the driver follows this step:

1. Write "block" to the ICR.

Figure 6–11 shows this step in diagram form.

Figure 6–11: Block Interrupts—Diagram



## 6.11 Unblock Interrupts

This section describes the driver's task of unblocking interrupts from the DELQA-T board.

### 6.11.1 Unblock Interrupts—Description

Unblocking interrupts removes the effects of a previous block of interrupts. The DELQA-T board will then generate interrupts following the transmit and receive operations that it completes, until the next block request.

Remember, on unblock, if any interrupts occurred while the DELQA-T board was blocked, the board will immediately give a single interrupt. The driver can then examine the SRR and the transmit and receive descriptor rings to determine the reasons for the interrupt.

The interrupt service routine should always unblock interrupts once it has finished processing returned status on the receive and transmit rings.

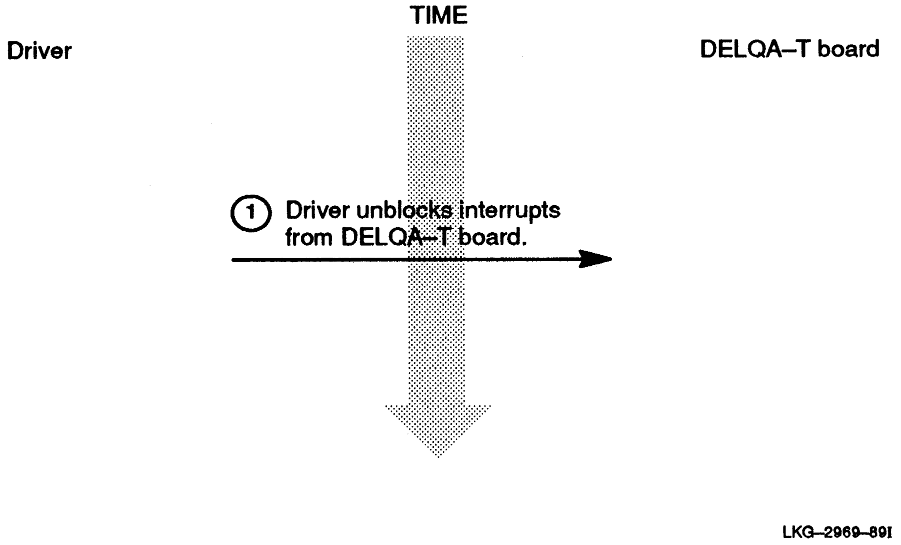
### 6.11.2 Unblock Interrupts—Steps

To unblock interrupts, the driver follows this step:

1. Write "unblock" to the ICR.

Figure 6–12 shows this step in diagram form.

**Figure 6–12: Unblock Interrupts—Diagram**



## 6.12 Return to DELQA-normal Mode

This section describes the steps to return the DELQA-T board to DELQA-normal mode in an orderly way.

### 6.12.1 Return to DELQA-normal Mode—Description

This task is different from simply resetting or stopping the DELQA-T board in that it incorporates both those tasks.

### 6.12.2 Return to DELQA-normal Mode—Steps

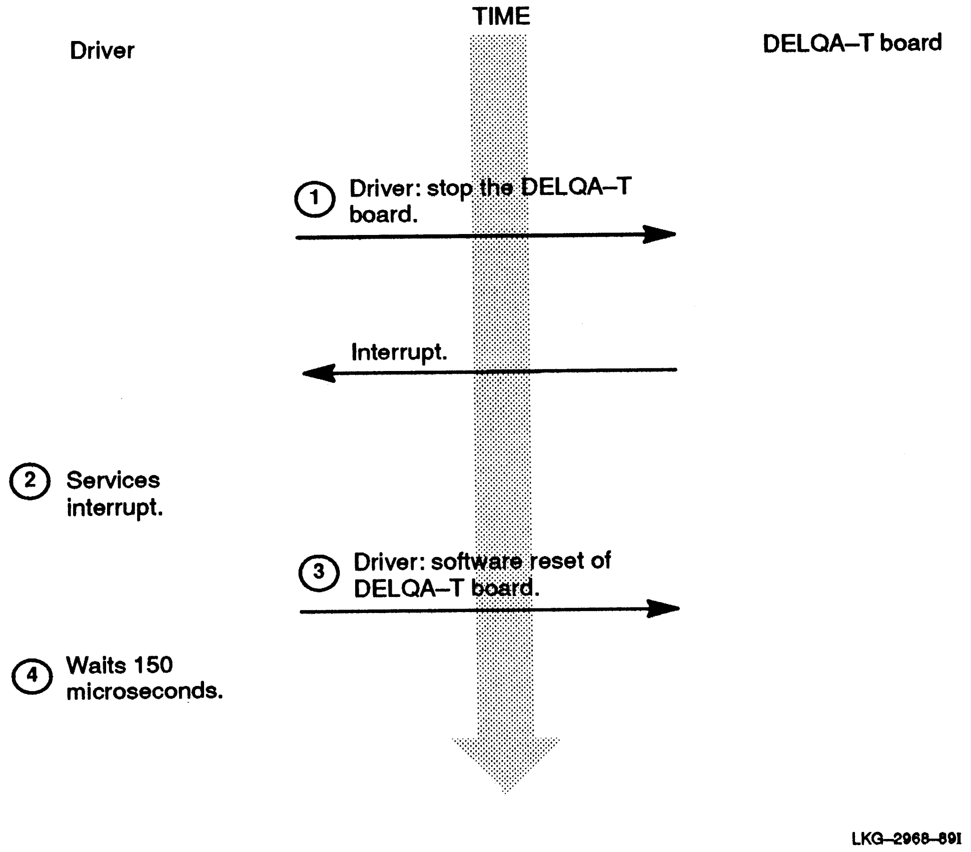
To return the DELQA-T board to DELQA-normal mode, the driver follows these steps:

1. The driver stops the DELQA-T board, as described in Section 6.6.
2. The driver waits for the interrupt from the DELQA-T board that results from the stop, and it services the interrupt, as described in Section 6.9.6.
3. The driver does a software reset of the DELQA-T board, as described in Section 6.7.
4. The driver waits 150 microseconds.

   After this time, the DELQA-PLUS board will be in DELQA-normal mode.

Figure 6–13 shows these steps in diagram form.

## Figure 6–13: Return to DELQA-normal Mode—Diagram


TIME
Driver
DELQA–T board
1 Driver: stop the DELQA–T board.
Interrupt.
2 Services interrupt.
3 Driver: software reset of DELQA–T board.
4 Waits 150 microseconds.
LKG–2968–89I

## 6.13 Registers on the DELQA-T Board

The DELQA-T board provides a block of registers that let the device driver control the board.

The registers are:

- The ARQR, SRQR, ICR, IBAH, and IBAL which the driver uses to talk to the board.
- The SRR, which the board uses to talk to the driver.
- The SA ROM registers, which contain the board's unique 48-bit physical address.

These registers reside at the same Q-bus address as the block of registers for DELQA-normal mode. But the registers for the DELQA-T board are different from the registers for the DELQA-normal board and their contents are different as well.

The following sections describe the contents of the DELQA-T registers in detail. Figure 6–14 shows the registers in diagram form.

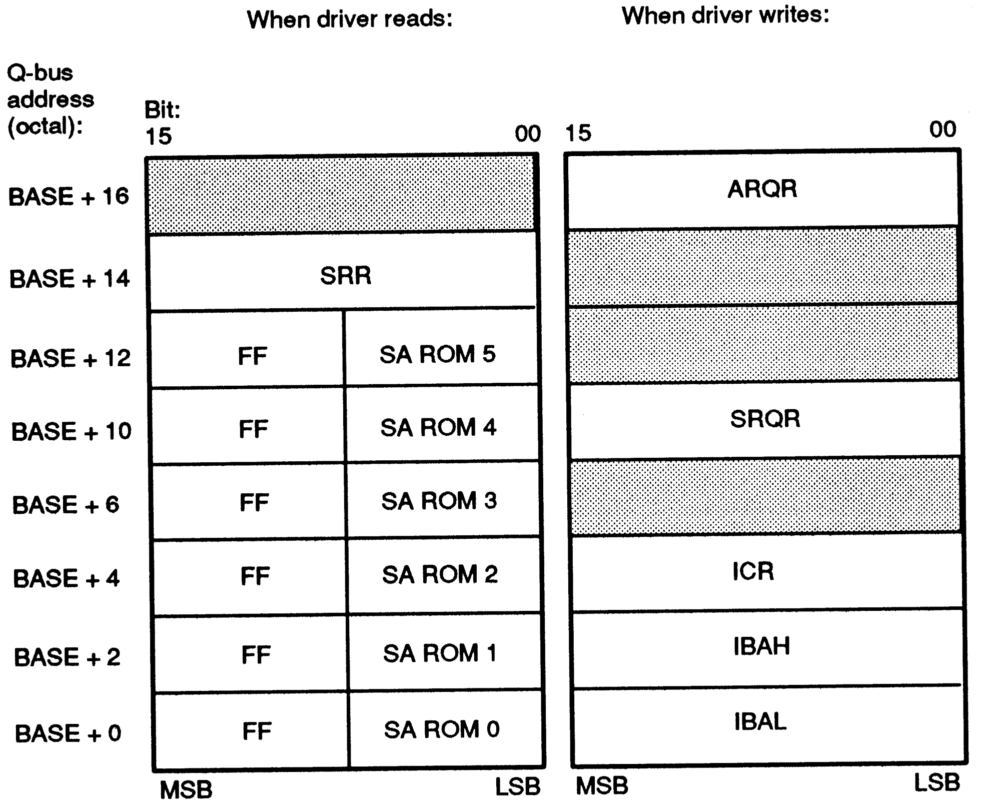
### 6.13.1 Reserved Fields

Reserved fields are denoted by shading in the illustrations and by the word "Reserved" in the descriptions of the fields.

With these fields, the driver must

- Always write reserved fields as 0 (zero).
- Always interpret reserved fields as 0 (zero), even if they read as something else.

## Figure 6–14: Registers on the DELQA-T board



| Q-bus address (octal): | When driver reads: Bit: 15–00 (MSB–LSB) | | When driver writes: 15–00 (MSB–LSB) |
|---|---|---|---|
| BASE + 16 | ▨ | | ARQR |
| BASE + 14 | SRR | | ▨ |
| BASE + 12 | FF | SA ROM 5 | ▨ |
| BASE + 10 | FF | SA ROM 4 | SRQR |
| BASE + 6 | FF | SA ROM 3 | ▨ |
| BASE + 4 | FF | SA ROM 2 | ICR |
| BASE + 2 | FF | SA ROM 1 | IBAH |
| BASE + 0 | FF | SA ROM 0 | IBAL |

LKG–2972–89I

### 6.13.2 The Status And Response Register (SRR)

The DELQA-T board uses the SRR register to report its status after data transfers. (The transfers are those that occur between the board and the driver.)

The driver reads the SRR either directly or indirectly:

- Indirectly, for all transmit and receive operations, which do not require an immediate response by the driver after each operation

  or

- Directly, for start-board and stop-board operations, which do require an immediate response by the board after each operation.

### CAUTION

> The driver **must not write** to the SRR register because unpredictable results can occur.

### Contents of SRR

Figure 6–15 shows the SRR in diagram form, and Table 6–2 describes the contents of each field.

**Figure 6–15: Contents of the Status and Response Register (SRR)**

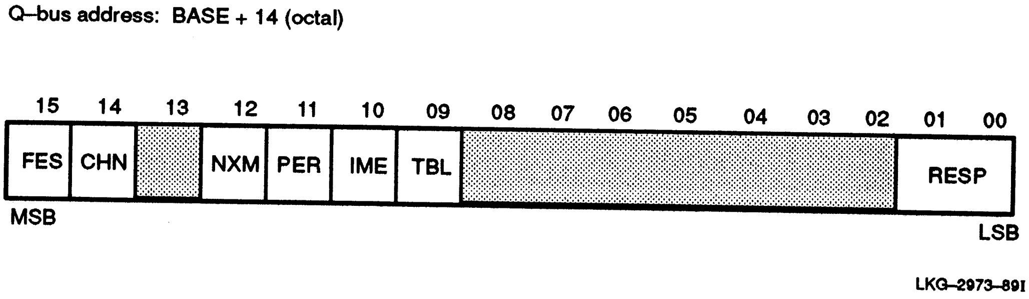


**Table 6–2: Fields in the SRR Register**

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| SRR[15] | FES | Fatal Error Summary.<br>DELQA-T board sets this to indicate that a fatal error has occurred. This field is the logical or of CHN, NXM, PER, IME and TBL.<br>Software-reset clears this.<br>Note that FES is NOT set on a BBL error (see T/RMD2). | Read |
| SRR[14] | CHN | Chaining Error. | Read |

Table 6–2 (Cont.): Fields in the SRR Register

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| | | DELQA-T board sets this to indicate that a chaining error has occurred, that is, two sequential transmit descriptors had the FOT field set.<br>Software-reset clears this. | |
| SRR[13] | | (Reserved.) | |
| SRR[12] | NXM | Non-existent Memory Error (on the Q-bus).<br>DELQA-T board sets this to 1 to indicate that a Q-bus/host memory access has timed-out.<br>Software-reset clears this.<br>NXM usually means a bad address in the init block. | Read |
| SRR[11] | PER | Parity Error (on the Q-bus).<br>DELQA-T sets this to value 1 to indicate that a parity error has occurred on an access to Q-bus/host memory.<br>Software-reset clears this. | Read |
| SRR[10] | IME | Internal Memory Error.<br>DELQA-T board sets this to indicate that an internal memory access error has occurred.<br>Software-reset clears this. | Read |
| SRR[09] | TBL | Transmit Buffer Too Long.<br>DELQA-T board sets this to indicate that that the driver gave a transmit request with a buffer size that was either zero or illegally large.<br>Software-reset clears this. | Read |
| SRR[08:02] | | (Reserved.) | |
| SRR[01:00] | RESP | DELQA-T mode Synchronous Response Field. | Read |

Table 6–2 (Cont.): Fields in the SRR Register

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| | | DELQA-T board sets this to return a synchronous response to the driver, following a synchronous request from the driver. Range:<br>Value Meaning<br>00 - No Response<br>01 - Response to a select T-mode request<br>10 - Response to a start-device request<br>11 - Response to a stop-device request<br>The DELQA-T board never clears this field; subsequent responses will overwrite the previous value. | |

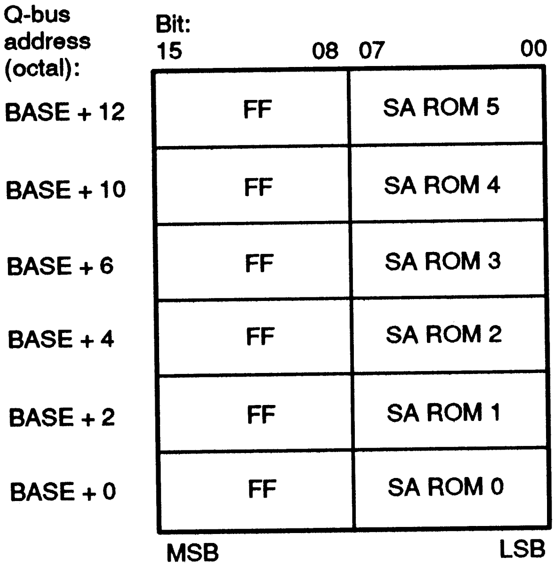
### 6.13.3 Station Address ROM (SA ROM) Locations

The station address ROM locations contain the unique 48-bit physical address of the DELQA-T board. This address comes from the station address (SA) ROM chip.

Notice that this address resides in only the least significant bytes of each word; the contents of the most significant bytes are all FF (hex).

Figure 6–16 shows the SA ROM registers.

Figure 6–16: The Station Address ROM Locations



| Q-bus address (octal): | Bit: 15 — 08 | 07 — 00 |
|---|---|---|
| BASE + 12 | FF | SA ROM 5 |
| BASE + 10 | FF | SA ROM 4 |
| BASE + 6 | FF | SA ROM 3 |
| BASE + 4 | FF | SA ROM 2 |
| BASE + 2 | FF | SA ROM 1 |
| BASE + 0 | FF | SA ROM 0 |
|  | MSB | LSB |

LKG–2974–89I

### 6.13.4 Synchronous Request Register (SRQR)

The SRQR lets the driver ask the DELQA-T board to start or stop. The SRQR offers only these two functions. (To have the DELQA-T board transfer data, the driver uses the ARQR (Asynchronous Request Register).)

The DELQA-T board always responds to requests to the SRQR by writing a value to the appropriate field in the SRR. If interrupts are enabled (that is, the INT field (bit 00) in the OPTION field of the init block is set to 1), the board will also respond with an interrupt.

The driver must not send another request to the SRQR until the previous request has completed (hence, the name synchronous request register).

### Contents of SRQR

Figure 6–17 shows the SRQR in diagram form, and Table 6–3 describes the contents of each field.

**Figure 6–17: Contents of the Synchronous Request Register (SRQR)**

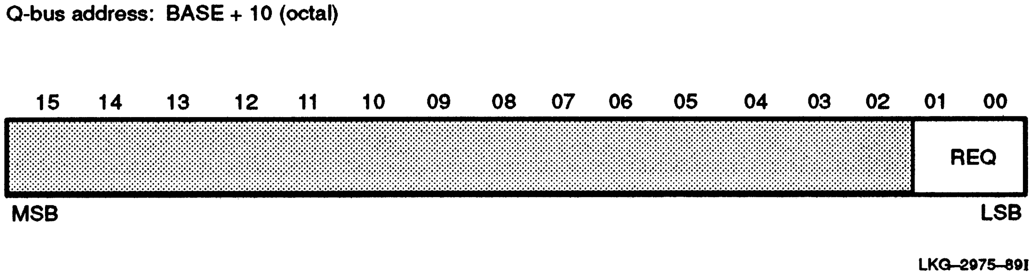


**Table 6–3: Fields in the SRQR Register**

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| SQR[15:02] | | (Reserved.) | |
| SQR[01:00] | REQ | Synchronous Request Field.<br>The driver sets this to issue a synchronous request to the DELQA-T board.<br>`Value   Meaning`<br>`00 - (Invalid Request)`<br>`01 - (Invalid Request)`<br>`10 - Request to start the DELQA-T`<br>`11 - Request to stop the DELQA-T` | Write |

## 6.13.5 Asynchronous Request Register (ARQR)

The ARQR lets the driver tell the DELQA-T board to transfer data between host memory and the network. Actually, the driver only notifies the DELQA-T board that there is a request to transfer data. The driver will have already written the request into the appropriate field in the transmit and/or receive buffer descriptors; and the DELQA-T board must then go and read this request or requests from those descriptors.

On completing the requested operation, the DELQA-T board will

1. Note errors, if any, by writing appropriate fields in the SRR.
2. Set the ownership in the appropriate transmit or receive buffer's descriptor.
3. Post an interrupt. If interrupts are enabled and the driver does not block interrupts (which it would normally do if it were checking the SRR and then reading a buffer descriptor after a transfer), the DELQA-T board will post an interrupt.

For specific information on how the driver should handle each operation, see the "Steps" section under that operation's section earlier in this chapter.

Keep in mind that the driver can issue further ARQR requests without waiting for responses (hence, the name asynchronous request register). On the other hand, the driver may have to wait after it has transmitted all its transmit buffers or made all its receive buffers available for incoming data.

The ARQR also allows the driver to do a software reset of the DELQA-T board; there is no response after a software reset since the DELQA-T board reverts to DELQA-normal mode.

### Contents of ARQR

Figure 6–18 shows the ARQR in diagram form, and Table 6–4 describes the contents of each field.

Figure 6–18: Contents of the Asynchronous Request Register (ARQR)

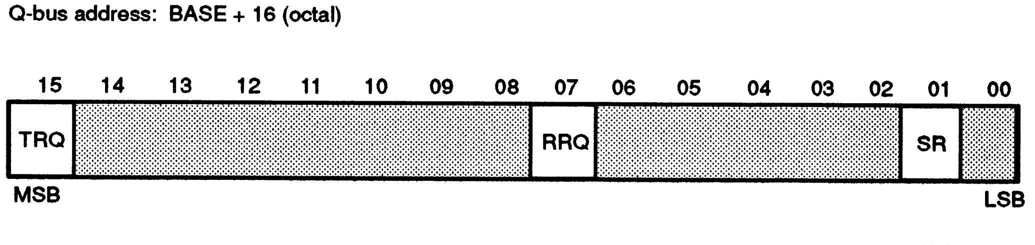


Table 6–4: Fields in the ARQR Register

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| CSR[15] | TRQ | Transmit Request.<br>Driver clears by writing a '1', which requests the DELQA-T board to perform a transmit; writing a '0' to this bit has no effect.<br>DELQA sets on entering T-mode, and when it has recognized a transmit request. | Write |
| CSR[14:08] | | (Reserved.) | |
| CSR[07] | RRQ | Receive Request<br>Driver clears by writing a '1', which requests the DELQA-T board to perform a receive; writing a '0' to this bit has no effect.<br>DELQA-T board sets on entering T-mode, and when it has recognized a receive request. | Write |
| CSR[06:02] | | (Reserved.) | |
| CSR[01] | SR | Software Reset.<br>Driver sets to '1' as step one of issuing a request to software-reset the board.<br>Driver sets to '0' immediately afterwards as step two of issuing a Software Reset.<br>Note that after issuing a software-reset the driver must delay for 150 microseconds for device DMA to terminate. | Write |
| CSR[00] | | (Reserved.) | |

### 6.13.6 Interrupt Control Register (ICR)

The ICR register lets the driver enable or disable interrupts from the DELQA-T board.

After the DELQA-T board has started running, it can also block and unblock interrupts; see Section 6.9.2.

### Contents of ICR

Figure 6–19 shows the ICR in diagram form, and Table 6–5 describes the contents of each field.

Figure 6–19: Contents of the Interrupt Control Register (ICR)

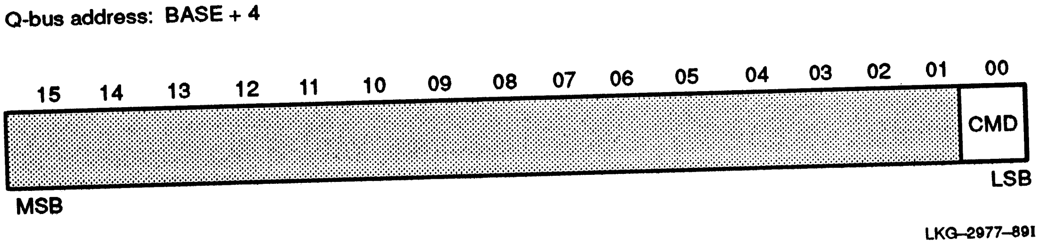


Table 6–5: Fields in the ICR Register

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| ICR[15:01] | | (Reserved.) | |
| ICR[00] | CMD | Written by the driver, read by the DELQA-T board.<br>The driver can issue the following commands in this field:<br>0—Block transmit/receive interrupts<br>1—Unblock transmit/receive interrupts<br>This field is initialized to '1' (Unblock Interrupts) when the driver issues a request to start the board. | Write |

### 6.13.7 Init Block Registers (IBAH and IBAL)

The driver uses the IBAH and IBAL registers to give the base address of the init block to the DELQA-T board. IBAL contains the least significant bits of the address, and IBAH contains the most significant bits of the address.

The DELQA-T board can then read the location of the init block from these registers. The DELQA-T board will read these registers in response to only a start-board request from the driver, and it won't read them again until the next start-board request.

The IBAL and IBAH registers reside at Q-bus address BASE + 0 and BASE + 2, respectively.

The init block resides in host memory. For more information on the init block, see Section 6.14.3.

## 6.14 Data Structures In Host Memory

The DELQA-T device driver must maintain some data structures in host memory:

The main data structures are:

- An initialization block
- Transmit and receive buffers (each of which has a descriptor block associated with it)

The driver will require the following data structures for each set of buffer descriptors (transmit and receive):

- Two pointers, one to the place to put the next buffer and the other to the place from which to get the next buffer
- A counter for the number of outstanding transmits

The following sections describe the main data structures in detail.

### 6.14.1 Reserved Fields

Reserved fields are denoted by shading in the illustrations and by the word "Reserved" in the descriptions of the fields.

With these fields, the driver must

- Always write reserved fields as 0 (zero).
- Always interpret reserved fields as 0 (zero), even if they read as something else.

### 6.14.2 Data Structures On the DELQA-T Board

It's not necessary (or possible) for the driver to maintain any data on the board; the driver reads and writes only registers, specifically, the ARQR, SRQR, IBAH, and IBAL.

### 6.14.3 The Init Block

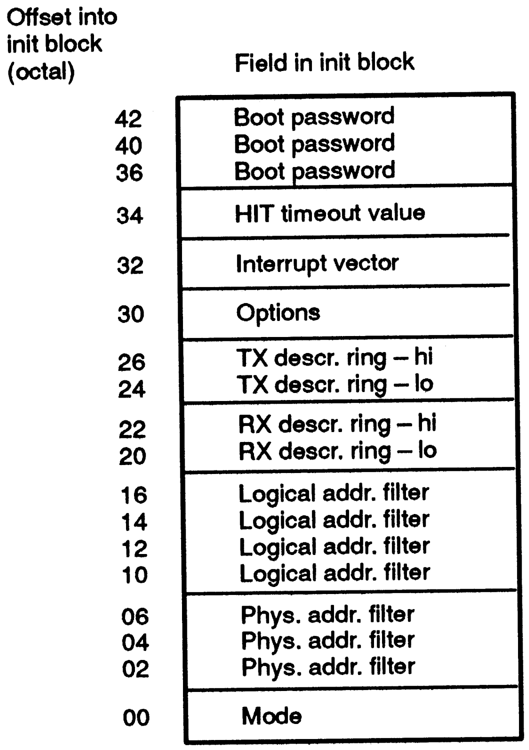
The device driver must supply the init block so that the DELQA-T board can understand how to communicate with the driver. The block includes the interrupt vector that the board should use, pointers to the transmit- and receive-buffer rings, and timeout periods, among other information.

The device driver supplies the init block to the board by writing two on-board registers (the IBAL and IBAH registers) so that they contain the least significant and most significant bits, respectively, of the init block's address.

#### Contents Of The Init Block

Figure 6–20 shows the init block in diagram form, and the sections that follow describe the contents of each field.

## Figure 6–20: Contents of the Init Block



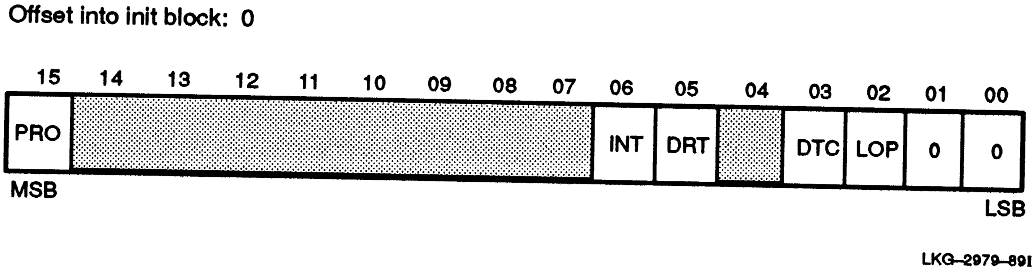
| Offset into init block (octal) | Field in init block |
|---|---|
| 42 | Boot password |
| 40 | Boot password |
| 36 | Boot password |
| 34 | HIT timeout value |
| 32 | Interrupt vector |
| 30 | Options |
| 26 | TX descr. ring – hi |
| 24 | TX descr. ring – lo |
| 22 | RX descr. ring – hi |
| 20 | RX descr. ring – lo |
| 16 | Logical addr. filter |
| 14 | Logical addr. filter |
| 12 | Logical addr. filter |
| 10 | Logical addr. filter |
| 06 | Phys. addr. filter |
| 04 | Phys. addr. filter |
| 02 | Phys. addr. filter |
| 00 | Mode |

LKG–2978–89I

### 6.14.3.0.1 Mode Field in Init Block

The MODE field lets the driver specify various standard operating modes. Figure 6–21 shows the MODE field in diagram form and Table 6–6 describes the subfields in the MODE field.

## Figure 6–21: Contents of the MODE Field



## Table 6–6: Bits in the MODE field

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| MR[15] | PRO | Promiscuous Mode.<br>1 = enable promiscuous mode; that is, enable reception from the LAN of all packets.<br>0 = disable promiscuous mode; that is, enable reception from the LAN of only those packets that match the physical address (PADR) or logical address filter (LADRF) fields in the init block, or that are broadcast packets. | Write |
| MR[14:07] |  | (Reserved.) |  |
| MR[06] | INT | Internal Loopback.<br>1 = run in internal loopback mode.<br>0 = run in external loopback mode.<br>NOTE: This field is ignored if the LOP field is set to '0'. | Write |
| MR[05] | DRT | Disable Retry.<br>1 = disable transmit retry.<br>0 = enable transmit retry. | Write |
| MR[04] |  | (Reserved.) |  |
| MR[03] | DTC | Disable Transmit CRC<br>1 = disable transmit CRC generation. | Write |

Table 6–6 (Cont.): Bits in the MODE field

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| | | 0 = enable transmit CRC generation. | |
| MR[02] | LOP | Loopback.<br>1 = enable loopback mode.<br>0 = disable loopback mode.<br>NOTE: In loopback mode, the transmit byte count (BCT field in the transmit buffer descriptor) is limited to 32. | Write |
| MR[01:00] | | Must be 0 (zero). | |

### 6.14.3.0.2 Physical Address Filter in Init Block

The physical address is the unique 48-bit address that the driver assigns to the DELQA-T board.

Remember, there can be more than one DELQA-T board on a single Q-bus system.

| Offset into init block (bytes): | | |
|---|---|---|
| 06 | P4 | P5 |
| 04 | P2 | P3 |
| 02 | P0 | P1 |
| | MSB | LSB |

Where: P0 through P5 are the most significant to least significant bytes, respectively, in the physical address filter.

### 6.14.3.0.3 Logical Address Filter in Init Block

This is a 64-bit hash filter that works similarly to that on the LANCE (Local-Area Network Controller, Ethernet). It gives imperfect filtering of multicast addresses.

| Offset into init block (bytes, octal): | | |
|---|---|---|
| 16 | L6 | L7 |
| 14 | L4 | L5 |
| 12 | L2 | L3 |
| 10 | L0 | L1 |
| | MSB | LSB |

Where: L0 through L7 are the most significant to least significant bytes, respectively, in the logical address filter.

### 6.14.3.0.4 Address of Receive Descriptor Ring

The address of the ring of receive descriptors resides in the init block. The address is 32 bits long and occupies two 16-bit words, as follows:

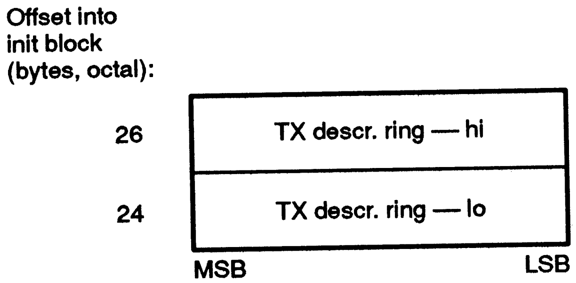
| Offset into init block (bytes, octal): | | |
|---|---|---|
| 22 | RX descr. ring — hi | |
| 20 | RX descr. ring — lo | |
| | MSB | LSB |

Where:

- RDRA-H contains the most significant bits of the Q-bus address of the first descriptor in the ring of receive descriptors.
- RDRA-L contains the least significant bits of the Q-bus address of the first descriptor in the ring of receive descriptors.

#### 6.14.3.0.5 Address of Transmit Descriptor Ring

The address of the ring of transmit descriptors resides in the init block. The address is 32 bits long and occupies two 16-bit words, as follows:



| Offset into init block (bytes, octal): | |
|---|---|
| 26 | TX descr. ring — hi |
| 24 | TX descr. ring — lo |
| | MSB LSB |

Where:

- TDRA-H is the most significant bits of the Q-bus address of the first descriptor in the ring of transmit descriptors.
- TDRA-L is the least significant bits of the Q-bus address of the first descriptor in the ring of transmit descriptors.

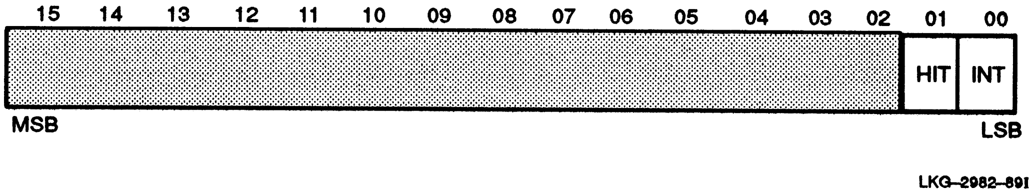
#### 6.14.3.0.6 Options Field in Init Block

The OPTIONS field lets the driver enable/disable interrupts from the DELQA-T board and enable/disable the HIT timer on the DELQA-T board.

Figure 6–22 shows the OPTIONS field in diagram form and Table 6–7 describes the subfields in the OPTIONS field.

Figure 6–22: Contents of the OPTIONS Field

Offset into init block: 30 (octal)



### Table 6–7: Bits in the OPTION field

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| OPT[15:02] | | (Reserved.) | |
| OPT[01] | HIT | Host Inactivity Timeout Flag.<br>Set by the driver to enable the inactivity timeout.<br>Cleared by the driver to disable the inactivity timeout.<br>After a select-T-mode request, the DELQA-T board enables HIT; it is up to the driver to explicitly enable/disable HIT in the init block. | Write |
| OPT[00] | INT | Interrupt Enable Flag.<br>1 = enable the DELQA-T board to interrupt the host.<br>0 = disable the DELQA-T board to interrupt the host. | Write |

### 6.14.3.0.7 Interrupt Vector in Init Block

This is the interrupt vector that the DELQA-T board will use when interrupting the host.

The interrupt vector's offset into the init block is 32 (octal).

### 6.14.3.0.8 Host Inactivity Timer (HIT) Timeout Value

This is the timeout value for the host inactivity timer (HIT). Units = 1 second.

After a select T-mode, the DELQA-T board uses a default HIT timeout value of 180 seconds (three minutes), until it reads in a new init block in response to a START request from the driver.

The HIT timeout value's offset into the init block is 34 (octal).

### 6.14.3.0.9 Boot Password in Init Block

This is the value of the password that the DELQA-normal board will use to verify the boot verification code that arrives in a MOP remote console boot message.

The DELQA-PLUS board maintains this password across:

- A HIT timeout (at which the DELQA-T board always moves back to DELQA-normal mode), and
- A software reset, which moves the DELQA-PLUS board from DELQA-T mode to DELQA-normal mode.

The DELQA-PLUS board does not maintain this password across a power-down.

| Offset into init block (bytes, octal): | Bit: 15 08 | 07 00 |
|---|---|---|
| 42 | BP4 | BP5 |
| 40 | BP2 | BP3 |
| 36 | BP0 | BP1 |
|  | MSB | LSB |

Where: BP0 through BP5 are the most significant to least significant bytes, respectively, in the boot password.

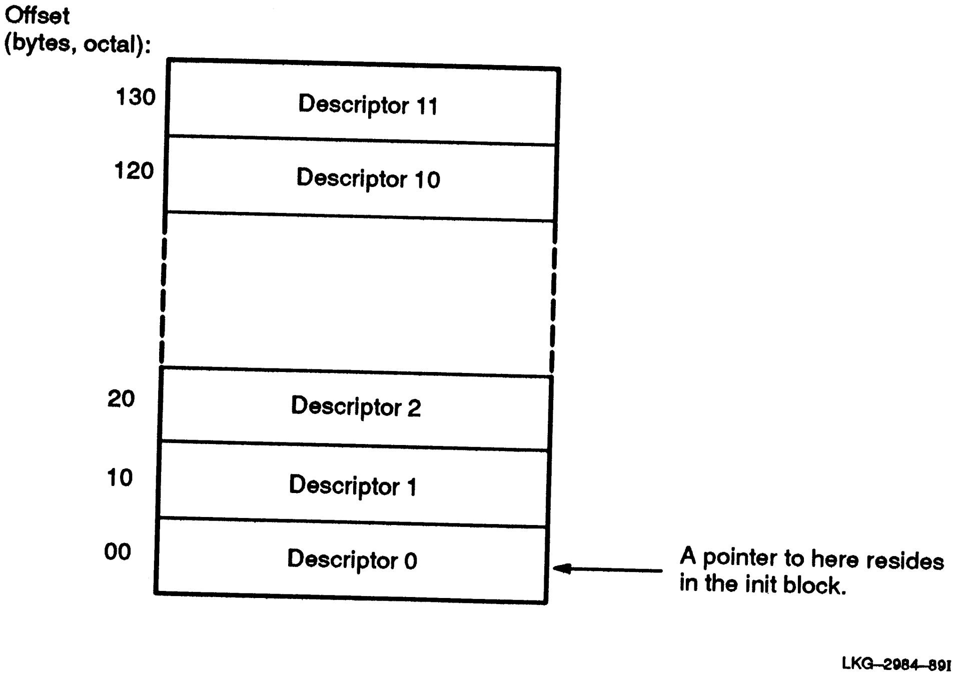
### 6.14.4 The Transmit And Receive Rings

The transmit and receive rings are rings of descriptors. Each descriptor contains information about a single buffer, including its size, location, and status.

Within each ring, the descriptors must reside contiguously in memory, and the first descriptor in the ring must be quadword-aligned; the DELQA-T board will always read the last two least-significant bits of the address of the ring (which resides in the init block) as 0 (zero).

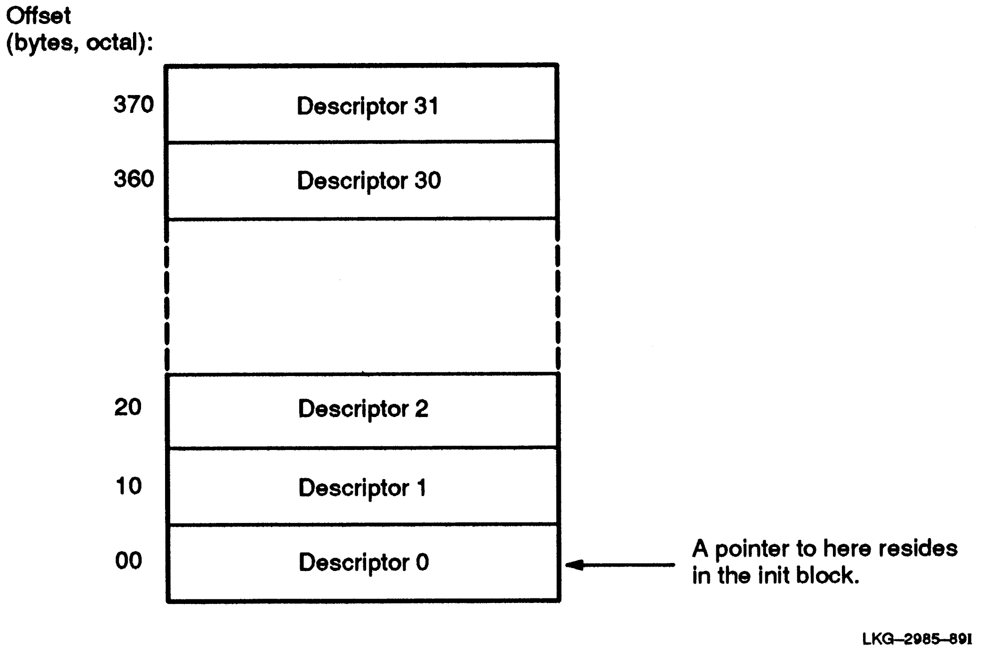
The transmit ring contains 12 (decimal) entries. Figure 6–23 shows the transmit ring's size and structure.

**Figure 6–23: Transmit Descriptor Ring**



The receive ring contains 32 (decimal) entries. Figure 6–24 shows the receive ring's size and structure.

**Figure 6–24: Receive Descriptor Ring**



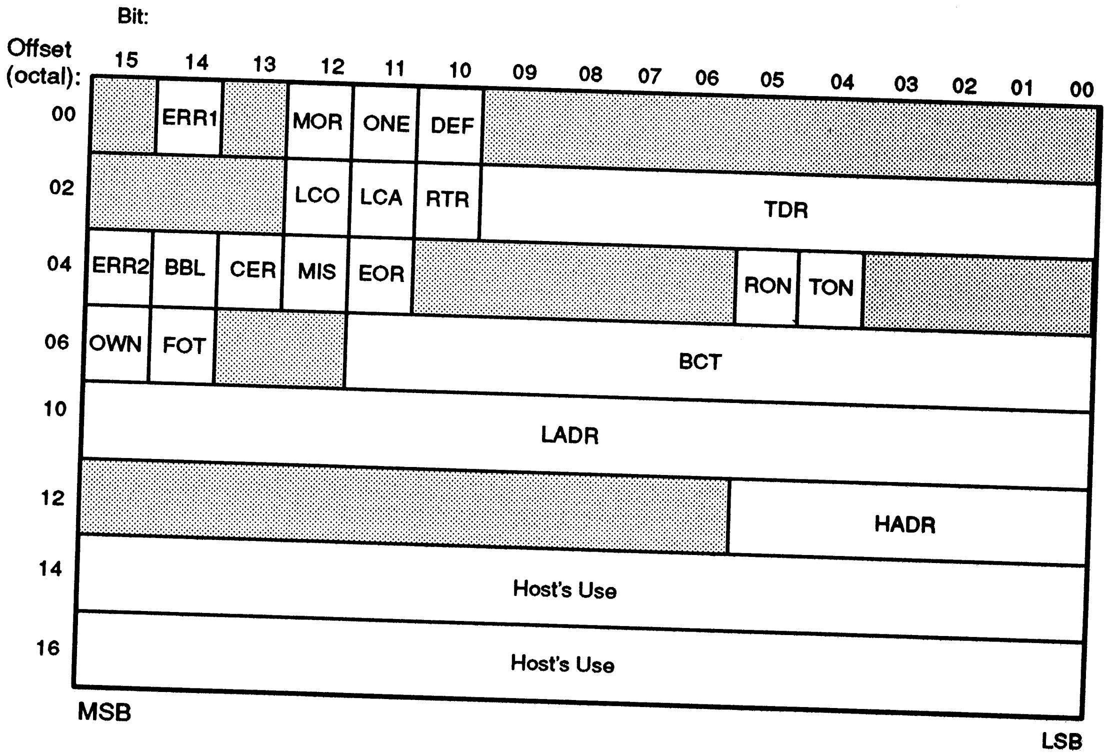
### 6.14.5 Transmit Buffer Descriptor

The purpose of a single transmit buffer descriptor is to provide information about a single buffer of data. The information is for the driver and the DELQA-T board to use when transmitting the buffer.

Keep in mind that some of the information is read-only or write-only for either the driver or the DELQA-T board, and some of the information is for sharing between the driver and the DELQA-T board.

Figure 6–25 shows the contents of a single transmit buffer descriptor in diagram form. Table 6–8 describes the contents in detail, and column four in this table indicates whether the driver may read and/or write the given field.

Figure 6–25: Contents of a Transmit Buffer Descriptor



LKG–2986–891

## 6.14.5.1 Fields in the Transmit Buffer Descriptor

Table 6–8 lists the fields in the transmit buffer descriptor.

**NOTE**

The DELQA-T board reports the information in TMD2 as soon as possible, and hence this information may be about operations other than the transmit operation.

Table 6–8: Fields in the Transmit Buffer Descriptor

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| TMD0[15] | | (Reserved.) | |

Table 6–8 (Cont.): Fields in the Transmit Buffer Descriptor

<table>
<tr><th>Bits</th><th>Mnemonic</th><th>Description</th><th>Can driver Read/Write?</th></tr>
<tr><td>TMD0[14]</td><td>ERR1</td><td>Error summary. This field is the "OR" of TMD1 fields LCO, LCA, and RTR.</td><td>Read</td></tr>
<tr><td>TMD0[13]</td><td></td><td>(Reserved.)</td><td></td></tr>
<tr><td>TMD0[12]</td><td>MOR</td><td>More than one retry on transmit.</td><td>Read</td></tr>
<tr><td>TMD0[11]</td><td>ONE</td><td>One retry on transmit.</td><td>Read</td></tr>
<tr><td>TMD0[10]</td><td>DEF</td><td>Deferral during transmit.</td><td>Read</td></tr>
<tr><td>TMD0[09:00]</td><td></td><td>(Reserved.)</td><td></td></tr>
<tr><td>TMD1[15:13]</td><td></td><td>(Reserved.)</td><td></td></tr>
<tr><td>TMD1[12]</td><td>LCO</td><td>Late collision on transmit— packet not transmitted.</td><td>Read</td></tr>
<tr><td>TMD1[11]</td><td>LCA</td><td>Loss of carrier on transmit— packet not transmitted.</td><td>Read</td></tr>
<tr><td>TMD1[10]</td><td>RTR</td><td>Retry error on transmit— packet not transmitted.</td><td>Read</td></tr>
<tr><td>TMD1[09:00]</td><td>TDR</td><td>Time Domain Reflectometry value (See LANCE spec).</td><td>Read</td></tr>
<tr><td>TMD2[15]</td><td>ERR2</td><td>Error summary. This field is the "OR" of BBL, CER, MIS in TMD2.</td><td>Read</td></tr>
<tr><td>TMD2[14]</td><td>BBL</td><td>Babble error on transmit.</td><td>Read</td></tr>
<tr><td>TMD2[13]</td><td>CER</td><td>Collision error on transmit.</td><td>Read</td></tr>
<tr><td>TMD2[12]</td><td>MIS</td><td>Packet lost on receive.</td><td>Read</td></tr>
<tr><td>TMD2[11]</td><td>EOR</td><td>End Of Receive Ring Reached<br>The driver should interpret the EOR and MIS fields together as follows:
<table>
<tr><th>EOR</th><th>MIS</th><th>Meaning</th></tr>
<tr><td>0</td><td>0</td><td>No data loss</td></tr>
<tr><td>1</td><td>0</td><td>No data loss; end of of receive ring</td></tr>
<tr><td>0</td><td>1</td><td>Data loss from lack of on-board buffers</td></tr>
<tr><td>1</td><td>1</td><td>Data loss from lack of host-memory buffers</td></tr>
</table></td><td>Read</td></tr>
<tr><td>TMD2[10:06]</td><td></td><td>(Reserved.)</td><td></td></tr>
</table>

Table 6–8 (Cont.): Fields in the Transmit Buffer Descriptor

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| TMD2[05] | RON | Receiver On. | Read |
| TMD2[04] | TON | Transmitter On. | Read |
| TMD2[03:00] | | (Reserved.) | |
| TMD3[15] | OWN | Ownership field.<br>DELQA-T board sets to value 1 to return ownership of this descriptor to the driver, after the DELQA-T board has written status in TMD0, TMD1 and TMD2.<br>Driver sets to value 0 to give ownership of this descriptor to the DELQA-T board, after the driver has written the transmit fields TMD3, TMD4 and TMD5.<br>Note that:<br>1 = Driver Ownership—the DELQA-T board may not write to this descriptor, and must not use any information read from this descriptor.<br>0 = DELQA-T Ownership—the driver may not .write to this descriptor, and must not use any information read from this descriptor. | Read-Write |
| TMD3[14] | FOT | First Of Two flag.<br>1 = This descriptor points to the first of two buffers in a chained transmit.<br>0 = This descriptor points either to a single buffer transmit, or points to the second of two entries in a chained transmit.<br>NOTE: It is a fatal device error if the driver sets FOT in two successive descriptors. | Read-Write |
| TMD3[13:12] | | (Reserved.) | |
| TMD3[11:00] | BCT | Byte Count. | Read/Write |

Table 6–8 (Cont.): Fields in the Transmit Buffer Descriptor

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| | | Driver writes this to indicate to the DELQA-T board the length in bytes of the transmit buffer that this descriptor points to. If CRC generation is disabled, this byte count must be inclusive of the CRC.<br>See Table 6-1 for the size restrictions that apply. | |
| TMD4[15:00] | LADR | Least significant bits of the data buffer's address.<br>Driver sets to give the associated buffer to the DELQA-T board. | Write |
| TMD5[05:00] | HADR | Most significant bits of the data buffer's address.<br>Driver sets to give the associated buffer to the DELQA-T board. | Write |
| TMD5[15:06] | | (Reserved.) | |

### 6.14.5.2 Transmit Data Buffers

The transmit buffer holds the data to be transmitted.

The transmit buffer's address resides in the transmit descriptor as follows:

TMD5[05:00] contains bits 21:16 of the buffer's address.
TMD4[15:00] contains bits 15:00 of the buffer's address.

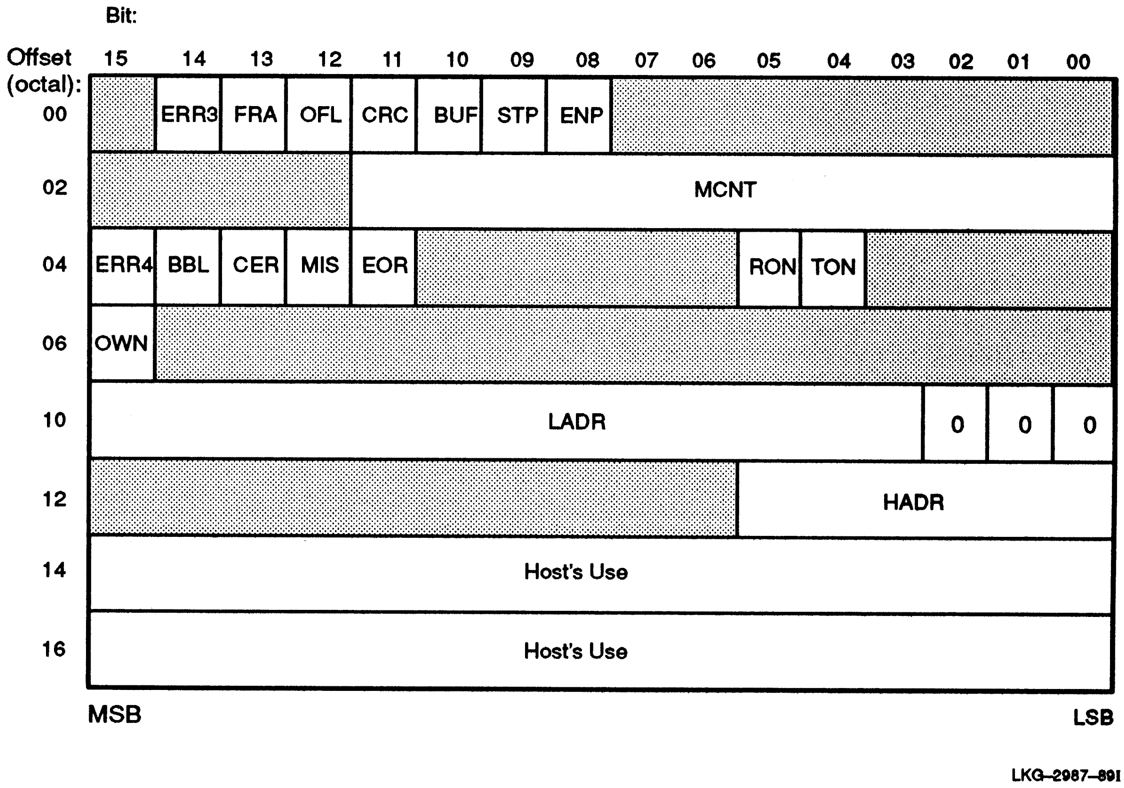
### 6.14.6 Receive Buffer Descriptor

The purpose of a single receive buffer descriptor is to provide information about a single buffer of data. The information is for the driver and the DELQA-T board to use when receiving the buffer.

Keep in mind that some of the information is read-only or write-only for either the driver or the DELQA-T board, and some of the information is for sharing between the driver and the DELQA-T.

Figure 6–26 shows the contents of a single receive buffer descriptor in diagram form. Table 6–9 describes the contents in detail, and column four in this table indicates whether the driver may read and/or write the given field.

**Figure 6–26: Contents of a Receive Buffer Descriptor**



### 6.14.6.1 Fields in the Receive Buffer Descriptor

Table 6–9 lists the fields in the transmit buffer descriptor.

**NOTE**

> The DELQA-T board reports the information in RMD2 as soon as possible, and hence this information may be about operations other than the receive operation.

**Table 6–9: Fields in the Receive Buffer Descriptor**

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| RMD0[15] | | (Reserved.) | |
| RMD0[14] | ERR3 | Error summary. This field is the "OR" of FRA, CRC, OFL and BUF. | Read |
| RMD0[13] | FRA | Framing error on receive. | Read |
| RMD0[12] | OFL | Overflow error on receive. When set this indicates that part of an oversized packet has been lost. | Read |
| RMD0[11] | CRC | CRC error on receive. | Read |
| RMD0[10] | BUF | Internal device buffer error. When set this indicates that part of an oversized packet has been lost. | Read |
| RMD0[09:08] | STP,ENP | Start of packet, end of packet. Normally, the DELQA-T board sets both fields.<br>When the DELQA-T board sets only one or the other or neither of these bits, this means the buffer contains the start, end, or middle of an oversize packet. For more information on oversize packets, see the section on oversize packets in this chapter. | Read, Read |
| RMD0[07:00] | | (Reserved.) | |
| RMD1[15:12] | | (Reserved.) | |
| RMD1[11:00] | MCNT | Message byte count. MCNT is the length of the received message, in bytes, that has been copied to this receive buffer.<br>This is only valid if the STP field is set to '1'; if the STP field is clear then the MCNT field must be interpreted as being 1518. Note that if an oversize packet is received, it will be copied into more than one receive buffer. Note that MCNT always includes the CRC. | Read |
| RMD2[15] | ERR4 | Error summary. This bit is the "OR" of BBL, CER, MIS in RMD2. | Read |
| RMD2[14] | BBL | Babble error on transmit. | Read |
| RMD2[13] | CER | Collision error on transmit. | Read |
| RMD2[12] | MIS | Packet lost on receive. | Read |

**Table 6–9 (Cont.): Fields in the Receive Buffer Descriptor**

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| RMD2[11] | EOR | End Of Receive Ring.<br>The driver should interpret the EOR and MIS fields together as follows:<br>EOR MIS Meaning<br>0 0 No data loss<br>1 0 No data loss; end of of receive ring<br>0 1 Data loss from lack of on-board buffers<br>1 1 Data loss from lack of host-memory buffers | Read |
| RMD2[10:06] | | (Reserved.) | |
| RMD2[05] | RON | Receiver On. | Read |
| RMD2[04] | TON | Transmitter On. | Read |
| RMD2[03:00] | | (Reserved.) | |
| RMD3[15] | OWN | Ownership field.<br>DELQA-T board sets this to value 1 to return ownership of this descriptor to the driver, after the DELQA-T has written status to RMD0, RMD1 and RMD2.<br>Driver sets this to value 0 to give ownership of this descriptor to the DELQA-T board, after the driver has written the receive fields RMD3, RMD4, RMD5.<br>Note that:<br>1 = Driver Ownership - the DELQA-T board may not write to this descriptor, and must not use any information read from this descriptor.<br>0 = DELQA-T Ownership - the driver may not write to this descriptor, and must not use any information read from this descriptor. | Read-Write |
| RMD3[14:00] | | (Reserved.) | |

Table 6–9 (Cont.): Fields in the Receive Buffer Descriptor

| Bits | Mnemonic | Description | Can driver Read/Write? |
|---|---|---|---|
| RMD4[15:03] | LADR | Least significant bits of the data buffer's address.<br>Driver sets this to give the associated buffer to the DELQA-T board.<br>Note that bits 2-0 must be '0'; they will be read by the DELQA-T board as '0's. | Write |
| RMD5[05:00] | HADR | Most significant bits of the data buffer's address. Driver sets this to give the associated buffer to the DELQA-T board. | Write |
| RMD5[15:06] | | (Reserved.) | |

### 6.14.6.2 Receive Data Buffers

The receive buffer holds the data that the DELQA-T board receives.

Each receive buffer must be at least 1518 (decimal) bytes in length.

The receive buffer's address resides in the receive buffer descriptor as follows:

> RMD5[05:00] contains bits 21:16 of the buffer's address.
> RMD4[15:02] contains bits 15:02 of the buffer's address.

All local receive buffers must be quad-word aligned (begin on byte-addresses that are divisible by eight).

# D

# Reading The DELQA-PLUS Board's ROM Version

This appendix contains instructions for reading the DELQA-PLUS board's ROM version. This lets you verify that the DELQA-PLUS board's ROM version is appropriate to run the DELQA-PLUS board as a DELQA-T board.

## D.1 Introduction

For a DELQA-PLUS board to operate as a DELQA-T board, the DELQA-PLUS board's ROM version must be at least 2.0.0.

To verify that this is true, host software starts the DELQA-PLUS board, then sends the DELQA-normal board a special request for the ROM revision level. The request takes the form of a MOP element block.

**NOTE**

> The host software is not running MOP at this point and need not engage in any other MOP-related activities.
>
> For more information on MOP, see the *DECnet Maintenance Operation Protocol (MOP) Functional Specification.*

The next two sections list the steps the host software, which we will refer to as the device driver from now on for convenience, should take to verify that the DELQA-PLUS board's ROM version is correct for DELQA-T mode operation. There are two main tasks to obtain the DELQA-PLUS board's ROM version:

- Test startup
- Request and read ROM version

## D.2 Test Startup—Steps

**NOTE**

> These steps assume the DELQA-PLUS board is properly installed in the host system's chassis and is powered up.

Figure D–1 shows these steps in diagram form.

1. The device driver causes the DELQA-PLUS board to perform a software reset. For information on software reset, see Chapter 6 in this *Addendum to DELQA User's Guide.*
2. The driver waits 150 microseconds.
3. The driver sets the Identity Test (IT) field (bit 00 in the DELQA-normal VAR register) to the value 1. The value 1 in the VAR means the board is a DELQA board, as opposed to a DEQNA board.

   The driver should then immediately check that the value of the IT field is 1, since a DEQNA board will not allow the IT field's value to become 1.

   Finally, the driver should clear the IT field to 0 (zero).
4. The driver checks the Mode Select (MS) field (bit 15) in the VAR.
   - If the value of MS = 1, the driver should proceed.
   - If the value of MS = 0, the board is a DELQA in DEQNA-only mode, which means on-board switch S3 is open.

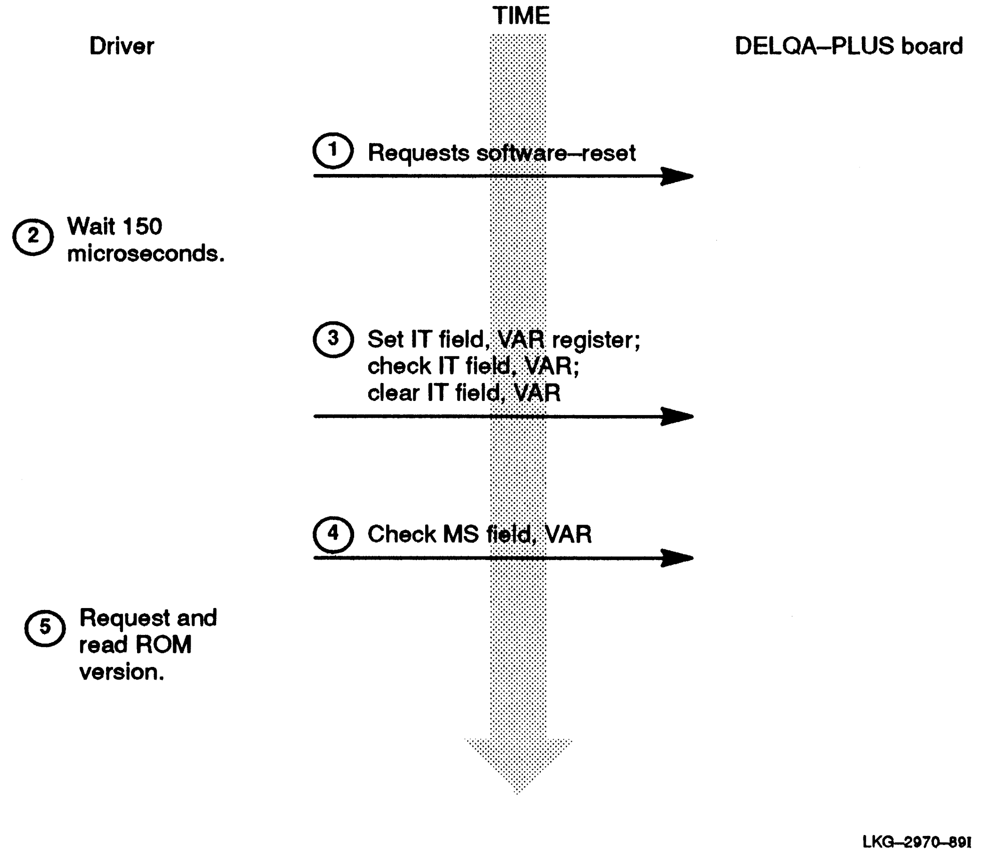
     Close switch S3 before allowing the driver to proceed.
5. The driver checks the DELQA-PLUS board's ROM version. See the steps in Section D.3.

The next section describes in more detail the steps to obtain the DELQA-PLUS board's ROM version level.

After the driver has verified that the ROM version is correct, the driver may proceed to move the DELQA-PLUS board to DELQA-T mode and start the

DELQA-T board running. For instructions in how to do this, see Chapter 6 in this *Addendum to DELQA User's Guide.*

**Figure D–1: Test Start-Up and Obtaining ROM Version**


TIME
Driver
DELQA–PLUS board
1 Requests software–reset
2 Wait 150 microseconds.
3 Set IT field, VAR register; check IT field, VAR; clear IT field, VAR
4 Check MS field, VAR
5 Request and read ROM version.
LKG–2970–89I

## D.3 Request/Read DELQA-PLUS Board's ROM Version— Steps

To obtain the DELQA-PLUS board's ROM version level, the driver must

1. Make sure the DELQA-PLUS board is in DELQA-normal mode; see Chapter 6 in this *Addendum to DELQA User's Guide* for instructions on how to place the DELQA-PLUS board in DELQA-normal mode.

2. Build an extended setup packet in host memory; see the *DELQA User's Guide* for instructions on how to build extended setup packets.

3. Insert a MOP element block (MEB), type 10, in that setup packet.

   - The MEB must be a type 10 and be the only MEB in the extended setup packet; otherwise, the DELQA-normal board may report the ROM version erroneously.

   - The MEB type 10's buffer must contain all zeros.

4. Send the extended setup packet to the DELQA-normal board; follow the instructions in the *DELQA User's Guide* for sending extended setup packets.

5. Read the appropriate buffer in the driver's receive ring in host memory. This buffer should be the same one the driver sent with the MEB type 10, and it should now contain the DELQA-PLUS board's ROM version.

   See Section D.3.3 for more information on this buffer.

The following sections describe the extended setup packet, the MOP element block (MEB) type 10 within that packet, and the MEB type 10's buffer.

### D.3.1 The Extended Setup Packet

The extended setup packet lets the driver give the DELQA-PLUS board a special command, which causes the DELQA-PLUS board to supply the driver with the DELQA-PLUS board's ROM revision level. This command is a MOP element block (MEB) type 10, which the driver places inside the extended setup packet.

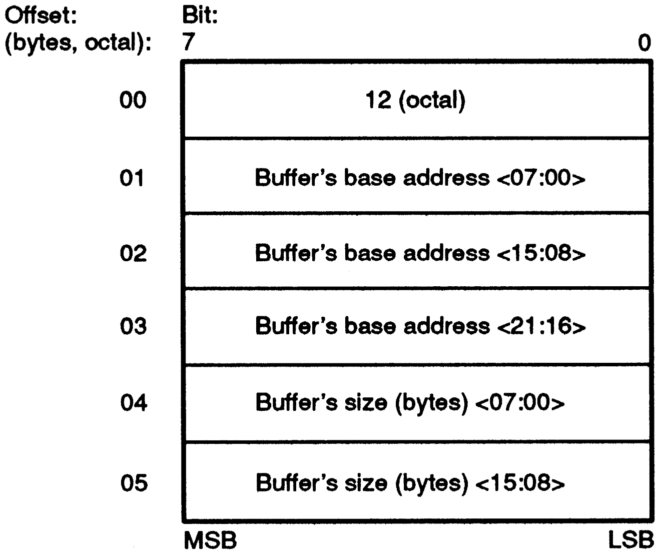
For complete information on how to construct extended setup packets for the DELQA board, see the *DELQA User's Guide.*

### D.3.2 MOP Element Block Type 10

MEB type 10 is called Read ROM Version, and its purpose is to cause the DELQA-PLUS board to report its ROM revision level back to the driver. MEB type 10 has the format shown in Figure D–2. This format consists of:

- The MEB itself, which is simply a pointer and a size, and
- A buffer (that the pointer points to), which contains the actual type-10 request(s).

## Figure D–2: MOP Element Block Type 10



| Offset: (bytes, octal): | Bit: 7 … 0 |
|---|---|
| 00 | 12 (octal) |
| 01 | Buffer's base address <07:00> |
| 02 | Buffer's base address <15:08> |
| 03 | Buffer's base address <21:16> |
| 04 | Buffer's size (bytes) <07:00> |
| 05 | Buffer's size (bytes) <15:08> |
|  | MSB … LSB |

LKG–3103–89I

The buffer that is associated with the MEB is described in Section D.3.3.

### D.3.3 The MOP Element Block (MEB) Type 10's Buffer

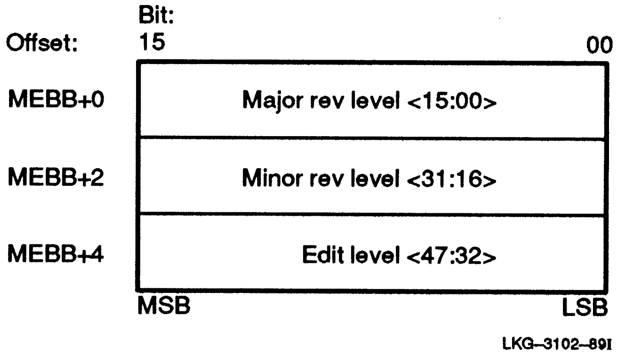
The purpose of the MEB Type 10's buffer is to hold both:

- The driver's request to the DELQA-PLUS board for the board's ROM revision level, and
- The ROM revision level that the DELQA-PLUS board writes back in response.

The MEB Type 10 buffer must reside in host memory and be at least three 16-bit words in length. The words must be contiguous.

Figure D–3 shows the MEB type 10 buffer in diagram form.

### Figure D–3: MEB Type 10's Buffer



| Offset: | Bit: 15 … 00 |
|---|---|
| MEBB+0 | Major rev level <15:00> |
| MEBB+2 | Minor rev level <31:16> |
| MEBB+4 | Edit level <47:32> |
| | MSB … LSB |

Example: ROM version 2.0.0 will appear as two in the first word (MEBB+0), zero in the second word, and zero in the third word.

The currently existing ROM versions are listed in Table D–1.

For a DELQA-PLUS board to operate as a DELQA-T board, the DELQA-PLUS board's ROM version must be at least 2.0.0.

### Table D–1: Current ROM Version Levels

| ROM Version | Summary |
|---|---|
| 0.10.37 | Firmware for the initial release of the DELQA product.<br>NOTE: See Section D.3.4 for special instructions on how to read this version number.<br>Does not support chaining of buffer descriptors on transmit. |
| 1.0.0 | Update of the initial firmware.<br>Supports chaining of buffer descriptors on transmit operations. |
| 1.9.0 | Field Test version of the DELQA-T firmware. |
| 2.0.0 | Firmware for the initial release of the DELQA-PLUS product. |

### D.3.4 ROM Version 0.10.37

The driver must verify this ROM version differently from the other ROM versions. This section explains why this is and how the driver must perform the verification.

ROM version 0.10.37 does not support MEB type 10. If the driver sends an MEB type 10 to a DELQA that is running firmware version 0.10.37, the DELQA board will return an error. (For information on handling errors, see the *DELQA User's Guide.*)

The error will not tell the driver whether the DELQA board made an error or the driver sent a bad setup packet.

Therefore, to verify ROM version 0.10.37, the driver must send the DELQA board an extended setup packet

- Of known validity and
- With the MEB type 10 as the only MEB in the packet.

The driver must then assume that if the DELQA board returns an error, the DELQA board is running ROM version 0.10.37.

# Glossary of Acronyms

This glossary lists acronyms associated with the DELQA-PLUS board.

| | |
|---|---|
| **ARQR** | Asynchronous request register (DELQA-T mode) |
| **CSR** | Control and status register (DELQA-normal mode) |
| **FOT** | First of two |
| **HIT** | Host inactivity timer |
| **IBAH** | Init block address register, high-order bits (DELQA-T mode) |
| **IBAL** | Init block address register, low-order bits (DELQA_T mode) |
| **ICR** | Interrupt control register (DELQA-T mode) |
| **LSB** | Least significant bit |
| **MOP** | Maintenance operations protocol |
| **MSB** | Most significant bit |
| **RMD** | Receive message descriptor |
| **ROM** | Read-only memory |
| **SA ROM** | Station address ROM |
| **SRQR** | Synchronous request register (DELQA-T mode) |
| **SRR** | Status/response register (DELQA-T mode) |
| **TMD** | Transmit message descriptor |
| **VAR** | Vector address register (DELQA-normal mode) |
| **XCR** | Extended control register (DELQA-normal mode) |

# Index