Software Kinetics

VOLUME 3
SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.02.0
Copy #3 05 May 1988

VOLUME 3
SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY PROJECT

Submitted to: C.R.C.
Ottawa, Ontario

SKL Document #1500-15-031.02.0
Copy #3 05 May 1988

SOFTWARE DETAILED DESIGN DOCUMENT
FOR THE
INTERNETWORK GATEWAY PROJECT

VOLUME 3

Contract No. 36001-6-3535/02-ST

05 May 1988

Prepared for:

Communications Research Centre
Ottawa, Ontario

Prepared by:

Software Kinetics Ltd.
65 Iber Road, P.O. Box 680
Stittsville, Ontario Canada
K0A 3G0

SKL Document #1500-15-031.02.0

Document Approval Sheet

for the

Internetwork Gateway Project

Document No: 1500-15-031.02.0

Document Name: Software Detailed Design Document
for the Internetwork Gateway Project

Approvals	Signature	Date
Project Engineer:	R. D. Bradford	5 May 1988
Project Manager:	T. M. Symchych	May 5/88
Technical Authority:	P. Labbe - CRC	7 June 88

Document Revision History

Revision	Description of Changes	Origin Date
01	New Document Issued	23 September 1987
02	Coding and Integration Revisions	05 May 1988

TABLE OF VOLUMES

TABLE OF CONTENTS

3.3.4 X.25 Device Driver (XDD) TLC

The X.25 Device Driver TLC contains the software that is used to control one IXIB X.25 interface from the IGW. There is a separate copy of this device driver in IGW memory for each IXIB board that is installed on the IGW (Maximum 9). This software is used to process incoming and outgoing commands and IP datagrams.

The driver consists of three LLCs, each of which operates as a separate process. The XDD_Supervisor LLC presents a common interface to the other TLCs in the gateway. All datagrams and commands for the IXIB are sent through the XDD_Supervisor. Similarly, all datagrams and commands from the IXIB pass through the XDD_Supervisor before being sent to the appropriate TLC.

The XDD_Transmitter LLC is responsible for the the transmission of commands and datagrams to the IXIB. Transmit interrupts are processed by this LLC.

The XDD_Receiver LLC is responsible for the the reception of commands and datagrams from the IXIB. Receive interrupts are processed by this LLC.

There is a separate copy of this device driver in IGW memory for each

IXIB board that is installed on the IGW.

3.3.4.1 X.25 Device Driver TLC Architecture

The X.25 Device Driver TLC consists of the following LLCs and units as shown in Figure 3-4:

1) XDD_Supervisor LLC - XDD_Supervisor LLC presents a common interface to the other TLCs in the gateway. The transfer of all datagrams and commands between the IXIB and other IGW TLCs is controlled by this LLC. This LLC is composed of the following units.

    1) XDD_Supervisor_Main - This unit is the main unit for the XDD_Supervisor LLC and contains the code for reading request given to the X.25 Device Driver TLC by other TLCs.

    2) Cmd_In - This unit is used to process incoming commands from the IXIB that have been received by the XDD_Receiver LLC.

    3) Cmd_Out - This unit is used to process outgoing commands that are to be sent to the IXIB by the XDD_Transmitter LLC.

    4) IP_In - This unit is used to process incoming IP datagrams from the IXIB that have been received by the XDD_Receiver LLC.

    5) IP_Out - This unit is used to process outgoing IP datagrams that are to be sent to the IXIB by the XDD_Transmitter LLC.

2) XDD_Receiver LLC - This LLC handles receive interrupts and gathers received commands and data from the IXIB. The following units comprise this LLC:

    1) XDD_Rx_Main - This unit is used to process IXIB receiver interrupts and to receive commands and data from the IXIB board.

```
                                 +------+
                                 | EDD  |
                                 | TLC  |
                                 +--+---+
                                    |
                     +--------------+---------------+
                     |                              |
             +-------+-------+              +-------+--------+
             | EDD_Interrupt |              | EDD_Supervisor |
             |      LLC      |              |      LLC       |
             +-------+-------+              +-------+--------+
                     |                              |
        +------------++---------+            +------+------+-------+
        |             |         |            |             |       |
+-------+-------+ +---+----+ +---+----+  +----+-----+ +------+----+  |
| EDD_Intr_Main | | EDD_Rx | | EDD_Tx |  | ARP_Free | | ARP_Input |  |
+---------------+ +--------+ +--------+  +----------+ +-----------+  |
                                                                     |
      +----------------+---------------+---------------+-------------+
      |                |               |               |             |
+-----+------+ +-------+-------+ +------+------+ +------+------+     |
| ARP_Lookup | | ARP_New_Entry | | ARP_Request | | ARP_Resolve |     |
+------------+ +---------------+ +-------------+ +-------------+     |
                                                                     |
      +-------------+-----------+-------------+----------------------+
      |             |           |             |
+-----+-----+ +-----+-----+ +----+-----+ +-----+------+
| ARP_Timer | | EDD_Input | | EDD_Main | | EDD_Output |
+-----------+ +-----------+ +----------+ +------------+
```

Figure 3-5

3) XDD_Transmitter LLC - This LLC handles transmit interrupts and the transfer of commands and data to the IXIB. This LLC is composed of the following units:

   1) XDD_Tx_Main - This units is used to process IXIB transmitter interrupts and to transfer commands and data to the IXIB.

3.3.4.2 Global Data

The following global data is defined for this TLC:

1) IXIB_STAT - This constant defined as 0x0c hex defines the command identifier for sending stat commands to the IXIB and receiving stat packets from the IXIB

2) IXIB_LOG - This constant defined as 0x06 indicates that an IXIB packet contains logging information.

3) IXIB_REBOOT - This constant defined as 0x1c indicates that an IXIB packet contains a reboot request message.

4) IXIB_MORE - This constant defined as 0x80 indicates that an IXIB packet is incomplete and there is more to follow.

5) IXIB_PSIZE - This constant defined as 251 indicates the maximum size of a packet to be transferred across the IXIB FIFO.

6) IXIB_TX_TO - This constant defined as 10 * CLOCK_INT indicates that a transmit interrupt from the IXIB was not generated as expected within 10 seconds.

3.3.4.3 X.25 Device Driver LLCs

The following LLCs are defined for the X.25 Device Driver TLC:

1) The XDD_Supervisor LLC - This LLC is used for communication between the X.25 Device Driver TLC and the other IGW TLCs.

2) The XDD_Transmitter LLC - This LLC is used to transmit commands and IP datagrams to the IXIB.

3) The XDD_Receiver LLC - This LLC is used to receive commands and IP datagrams from the IXIB.

These LLCs are described in the following subsections.

3.3.4.3.1 XDD_Receiver LLC

The XDD_Receiver LLC is responsible for the reception of commands and IP datagrams from the IXIB interface. The commands and datagrams are placed in ERTE Messages and then sent to the XDD_Supervisor LLC. The units that the XDD_Receiver LLC is composed of are:

1) The XDD_Rx_Main Unit.

3.3.4.3.1.1 Inputs

The following inputs are received by the XDD_Receiver LLC:

1) IXIB_Mailbox - This input is the MailBox register of the IXIB interface. It specifies the type of command or data to be received from the board.

2) IXIB_Fifo - This item is the IXIB FIFO register used to transfer data from the IXIB board.

3.3.4.3.1.2 Outputs

The following outputs are produced by the XDD_Receiver LLC:

1) Rx_Mesg_Hdr - This output consists of an array of 2 ERTE Message headers used to send received IXIB commands and datagrams to the XDD_Supervisor LLC. The first entry in the array is used to send received datagrams to the XDD_Supervisor using the XDD_Supervisor's datagram message queue. The second entry is used to send commands to the XDD_Supervisor using the the XDD_Supervisor's command message queue.

3.3.4.3.1.3 Local Data

No local data is defined for this LLC.

3.3.4.3.1.4 Processing

The XDD_Receiver LLC performs the following functions:

1) Receives command and IP packets from the IXIB and handles the associated receive interrupts.

2) Sends the received commands and IP packets to the XDD_Supervisor LLC.

3.3.4.3.1.5 Limitations

There are no limitations defined for the XDD_Receiver LLC.

3.3.4.3.2 XDD_Supervisor LLC

The XDD_Supervisor LLC is responsible for the communication between the XDD TLC and other TLCs comprising the IGW. The units that the XDD_Supervisor LLC is composed of are:

1) The XDD_Supervisor_Main Unit.

2) The Cmd_In Unit.

3) The Cmd_Out Unit.

4) The IP_In Unit.

5) The IP_Out Unit.

3.3.4.3.2.1 Inputs

The following inputs are received by the XDD_Supervisor LLC:

1) Mesg_Hdr - This item is the header of an ERTE message received by the LLC. The messages arrive in 4 types:

   1) Receive IP Datagrams - IP datagrams received by the IXIB are received by this LLC from the message queue IXIB_RX_IP.

   2) Receive IXIB Commands - Commands from the IXIB are collected from the message queue IXIB_RX_CMD.

   3) Transmit IXIB Datagrams - Datagrams to be transmitted on the IXIB interface are collected from the message queue IXIB_TX_CMD.

   4) Transmit IXIB Commands - Commands to be sent to the IXIB are received on the message queue IXIB_TX_CMD.

   5) ACT_Table - This global input is obtained from the global data area and contains the X.25 Address Configuration Table.

3.3.4.3.2.2 Outputs

The following outputs are produced by the XDD_Supervisor LLC:

1) Mesg_Hdr - This item is the header of an ERTE message forwarded by this LLC. The messages are forwarded to five different places:

   1) If the message contains an IP datagram received from the XDD_Receiver LLC, then the message is sent to the IP TLC.

   2) If the message contains an IP datagram from the IP TLC, then the message is sent to the XDD_Transmitter LLC on via the XDD_Transmitter's datagram message queue.

   3) If the message contains an IXIB command from the OI or STAT TLCs, then the message is sent to the XDD_Transmitter LLC on via the XDD_Transmitter's command message queue.

   4) If the message contains an IXIB log message, then the message is sent to the Console Device Driver TLC.

   5) If the message contains IXIB statistics data, then the message is sent to the STAT TLC.

3.3.4.3.2.3 Local Data

No local data is defined for the XDD_Supervisor LLC.

3.3.4.3.2.4 Processing

The XDD_Supervisor LLC performs the following functions:

1) Transfers IP packets from the IP TLC to the XDD_Transmitter LLC.

2) Transfers IP packets from the XDD_Receiver LLC to the IP TLC.

3) Processes incoming commands from the XDD_Receiver LLC.

4) Processes commands from the STAT and OI LLCs.

3.3.4.3.2.5 Limitations

There are no limitations defined for the XDD_Supervisor LLC.

3.3.4.3.3 XDD_Transmitter LLC

The XDD_Transmitter LLC is responsible for the transmission of commands and IP datagrams to the IXIB interface, and for the processing of IXIB transmit interrupts. The units that the XDD_Transmitter LLC is composed of are:

1) The XDD_Tx_Main Unit

3.3.4.3.3.1 Inputs

The following inputs are received by the XDD_Transmitter LLC:

1) Mesg_Hdr - This item is the header of an ERTE Message used to receive IXIB commands and datagrams from the XDD_Supervisor LLC. Command messages are read from the command message queue, and datagram messages are read from the datagram message queue.

3.3.4.3.3.2 Outputs

The following outputs are produced by the XDD_Transmitter LLC:

1) IXIB_Mailbox - This output is the MailBox register of the IXIB interface. It specifies the type of command or data to be transmitted to the IXIB board.

2) IXIB_Fifo - This item is the IXIB FIFO register used to transfer data to the IXIB board.

3.3.4.3.3.3 Local Data

The following local data is defined for the XDD_Transmitter LLC:

1) Error - This local data item is an integer value used to hold the return values of function calls.

3.3.4.3.3.4 Processing

The XDD_Transmitter LLC performs the following functions:

1) Receives command and IP packets from the XDD_Supervisor LLC.

2) Sends the command and IP packets to the IXIB.

3) Handles IXIB transmit interrupts.

3.3.4.3.3.5 Limitations

There are no limitations defined for the XDD_Transmitter LLC.

3.3.4.4 X.25 Device Driver Units

The following subsections contain the unit descriptions for the units comprising the X.25 Device Driver TLC.

3.3.4.4.1 Cmd_In Unit

The Cmd_In Unit performs the function of processing incoming command requests that have been received by the XDD_Rx_Main Unit from the IXIB.

3.3.4.4.1.1 Inputs

The following inputs are received by the Cmd_In Unit:

1) Mesg_Hdr - This 32 bit input parameter is a pointer to the header for an ERTE Message containing the received command to be processed.

3.3.4.4.1.2 Outputs

The following outputs are produced by the Cmd_In Unit:

1) Mesg_Hdr - This output is a 32 bit pointer to the header for an ERTE Message containing the received command to be forwarded to the Console Device Driver (CDD) or the STAT TLC.

3.3.4.4.1.3 Local Data

The following local data is defined for the Cmd_In Unit:

1) Command - This local data item is used to hold the command portion of the incoming messages.

2) Error - This local data item contains an error code indicating if an error occurred during the processing of the incoming command packet.

3.3.4.4.1.4 Processing

```
Move the Command field from the message referenced by Mesg_Hdr to
 Command
Switch Command
        Case IXIB_STAT
                Set M_Offset pointer of Mesg_Hdr to point to statistics
                 portion of message
                Error = Message_Send(STAT_QUEUE, Mesg_Hdr)
        Case IXIB_LOG:
                Set M_Offset field of Mesg_Hdr to point to log message
                 portion of message
                Error = Message_Send(CDD_Transmit_QUEUE, Mesg_Hdr)
        Case IXIB_REBOOT:
                Call Panic(Error message)
        Otherwise:
                Call OI_Print(Error message)
                Error = ERROR
Endcase
```

```
If Error isn't NOERROR
        Call Message_Discard(Mesg_Hdr)
Return
```

3.3.4.4.1.5 Limitations

There are no limitations defined for the Cmd_In Unit.

3.3.4.4.2 Cmd_Out Unit

The Cmd_Out Unit is called by the XDD Supervisor to transmit commands to the XDD_Transmitter LLC. This unit sends the ERTE Message containing the command to the XDD_Transmitter LLC.

3.3.4.4.2.1 Inputs

The following inputs are received by the Cmd_Out Unit:

1) Mesg_Hdr - This input is a 32 bit pointer to the header of the ERTE Message containing the command to be processed.

2) ACT_Table - This global input is the Address Configuration Table used to translate X.25 addresses into IP addresses and vice versa.

3.3.4.4.2.2 Outputs

The following outputs are produced by the Cmd_Out Unit:

1) Mesg_Hdr - This item is a 32 bit pointer to the header of the ERTE Message containing the command to be processed and sent to the XDD_Transmitter LLC.

3.3.4.4.2.3 Local Data

The following local data is defined for the Cmd_Out Unit.

3.3.4.4.2.4 Processing

```
Move the command from message buffer referenced by Mesg_Hdr to Command
Case Command
        IXIB_ACT:

        or

        IXIB_STAT:
                Error = Message_Send(Mesg_Hdr, IXIB_TX_CMD)
                If Error is equal NOERROR
                        Return
                Endif
        Otherwise:
                Call OI_Print(Error message)
Endcase
Call Message_Discard(Mesg_Hdr)
Return
```

3.3.4.4.2.5 Limitations

There are no limitations defined for the Cmd_Out Unit.

3.3.4.4.3 IP_In Unit

The IP_In Unit performs the function of passing incoming IP packets that have been received by the XDD_Rx_Main Unit to the IP TLC.

3.3.4.4.3.1 Inputs

The following inputs are received by the IP_In Unit:

1) Mesg_Hdr - This input parameter contains the header for the message buffer containing the received IP packet to be processed.

3.3.4.4.3.2 Outputs

The following outputs are produced by the IP_In Unit:

1) Mesg_Hdr - This item contains the header for the message buffer containing the IP packet to be sent to the IP TLC.

3.3.4.4.3.3 Local Data

The following local data is defined for the IP_In Unit:

1) Error - This local data item is used to hold the error code returned by the Message_Send function.

3.3.4.4.3.4 Processing

```
Error = Message_Send(Mesg_Hdr, IP_QUEUE)
If Error isn't NOERROR
        Call Message_Discard(Mesg_Hdr)
Endif
Return
```

3.3.4.4.3.5 Limitations

There are no limitations defined for the IP_In Unit.

3.3.4.4.4 IP_Out Unit

The IP_Out Unit is used to transfer ERTE Messages containing IP datagrams from the XDD_Supervisor LLC to the XDD_Transmitter LLC.

3.3.4.4.4.1 Inputs

The following inputs are received by the IP_Out Unit:

1) Mesg_Hdr - This input contains the header of the ERTE Message containing the IP packet to be sent to the XDD_Transmitter LLC.

3.3.4.4.4.2 Outputs

The following outputs are produced by the IP_Out Unit:

1) Mesg_Hdr - This output contains the header of the ERTE Message containing the IP packet to be sent to the XDD_Transmitter LLC.

3.3.4.4.4.3 Local Data

The following local data is defined for the IP_Out Unit:

1) Error - This local data item is used to hold the return value of the Message_Send function call.

3.3.4.4.4.4 Processing

```
Error = Message_Send(Mesg_Hdr, IXIB_TX_IP)
If Error isn't NOERROR
        Call Message_Discard(Mesg_Hdr)
Endif
```

3.3.4.4.4.5 Limitations

There are no limitations defined for the IP_Out Unit.

3.3.4.4.5 XDD_Rx_Main Unit

The XDD_Rx_Main Unit is responsible for the reception of commands and IP datagrams from the IXIB interface. The commands and IP datagrams are sent to the XDD_Supervisor LLC. IXIB receive interrupts are processed by this unit.

3.3.4.4.5.1 Inputs

The following inputs are received by the XDD_Rx_Main unit:

1) IXIB_Mailbox - This input is the MailBox register of the IXIB interface. It specifies the type of command or data to be received from the board.

2) IXIB_Fifo - This item is the IXIB FIFO register used to transfer data from the IXIB board.

3.3.4.4.5.2 Outputs

The following outputs are produced by the unit:

1) Rx_Mesg_Hdr - This output consists of an array of 2 ERTE Message headers used to send received IXIB commands and datagrams to the XDD_Supervisor LLC. The first entry in the array is used to send received datagrams to the XDD_Supervisor using the XDD_Supervisor's datagram message queue. The second entry is used to send commands to the XDD_Supervisor using the the XDD_Supervisor's command message queue.

3.3.4.4.5.3 Local Data

The following local data is defined for the XDD_Receiver LLC.

1) Rx_Reas - This local data contains a pointer to a message buffer header that is used to receive incoming command and data packets from the IXIB. The message buffer header that this pointer points to are contained in the Rx_Mesg_Hdr data item. 2) Q_Id - This local data item is used to store the queue ID of the queue that incoming packets are to be placed on when sent to the XDD_Supervisor LLC.

4) Command - This 16 bit item is used to hold incoming commands from the IXIB.

5) Count - This 16 bit data item is used to hold the length of incoming packets from the IXIB.

3.3.4.4.5.4 Processing

```
Loop
        Call Wait_Event(IXIB_RX_INTR)
        Move IXIB_Mailbox data to Command
        Move Command to Count
        Set Command to the bitwise AND of Command and MBX_CMD
        Set Count to the bitwise AND of Count and MBX_CNT
        If the bitwise AND of Command and the complement of IXIB_MORE
         is equal XC_IP_DATA
                Q_ID = IXIB_RX_IP
                Rx_Reas = Address of Rx_Mesg_Hdr[0]
        Else
                Q_ID = IXIB_RX_CMD
                Rx_Reas = Address of Rx_Mesg_Hdr[1]
        Endif
        If the header referenced Rx_Reas is empty
                Error = Message_Get(Rx_Reas)
                If Error isn't NOERROR
                        Call OI_Print(Error Message)
                        While Count > 0
                                Read FIFO
                                Decrement Count
                        Endwhile
                        Continue next loop iteration
                Endif
                Move Command bitwise anded with the complement of
                 IXIB_MORE to the location given in the M_Addr
                 field of Rx_Reas
                 If IP buffer M_Length field of Rx_Reas=16
                Else
                 M_Length field of Rx_Reas=2
        Endif
        While Count > 0
                Move IXIB_Fifo data to address given by M_Addr field
                 of Rx_Reas + M_Length field of Rx_Reas
                Increment M_Length field of Rx_Reas
                Decrement Count
        Endwhile
        If bitwise AND of Command and IXIB_MORE equals 0
                Error = Message_Send(Q_ID, Rx_Reas)
                If Error isn't NOERROR
                        Call Message_Discard(Rx_Reas)
                Endif
                Rx_Reas = Null
        Endif
Endloop
```

3.3.4.4.5.5 Limitations

There are no limitations defined for the XDD_Rx_Main Unit.

3.3.4.4.6 XDD_Supervisor_Main Unit

The XDD_Supervisor_Main Unit is responsible for the communication between the XDD TLC and other TLCs comprising the IGW. Commands and IP datagrams are transferred between the XDD_Transmitter LLC, the XDD_Receiver LLC, and the other IGW TLCs that access the XDD.

3.3.4.4.6.1 Inputs

The following inputs are received by the XDD_Supervisor_Main Unit:

1) Mesg_Hdr - This input is the header to an ERTE Message as described in the XDD_Supervisor LLC.

3.3.4.4.6.2 Outputs

The following outputs are produced by the XDD_Supervisor_Main Unit:

1) Mesg_Hdr - This output is the header to an ERTE Message as described in the XDD_Supervisor LLC.

3.3.4.4.6.3 Local Data

The following local data is defined for the XDD_Supervisor_Main Unit:

1) Error - This local data item is used to hold the return error code from the Message_Receive and Message_Get function calls.

2) Event - This local data item is used to determine if the MSG_ARRIVE event occurred before a timeout did.

3) XDD_Stat_To - This local data item contains the time to expire before an IXIB_STAT command is sent to the IXIB.

3.3.4.4.6.4 Processing

```
Move 300 * CLOCK_INT to XDD_Stat_To
Save_Time = Get_Time()
Loop
    Clear No_Wait
    Error = Message_Receive(IXIB_RX_IP, Mesg_Hdr)
    If Error is equal NOERROR
        Call IP_In(Mesg_Hdr)
    Else
        If Error isn't M_QEMPTY
            Call OI_Print(Error message)
            No_Wait++
        Endif
        Error = Message_Receive(IXIB_RX_CMD, Mesg_Hdr)
        If Error is equal NOERROR
            Call Cmd_In(Mesg_Hdr)
        Else
            If Error isn't M_QEMPTY
                Call OI_Print(Error Message)
                No_Wait++
            Endif
            Error = Message_Receive(IXIB_CMD, Mesg_Hdr)
            If Error is equal NOERROR
                Call Cmd_Out(Mesg_Hdr)
            Else
                If Error isn't M_QEMPTY
                    Call OI_Print(Error Message)
                    No_Wait++
                Endif
                Error = Message_Receive(Mesg_Hdr, IXIB_IP)
                If Error is equal NOERROR
                    Call IP_Out(Mesg_Hdr)
                Else if Error is M_QEMPTY
                    Event = Wait_Timeout(MSG_ARRIVE, XDD_Stat_To)
                    If Event isn't 0
                        Cur_Time = Get_Time()
                        Add Save_Time to XDD_Stat_To
                        Subtract Cur_Time from XDD_Stat_To
                        Move Cur_Time to Save_Time
                        If EDD_Stat_To is greater than 0
                            Continue next loop iteration
                        Endif
                    Endif
                    Move 300 * CLOCK_INT to XDD_Stat_To
                    Error = Message_Get(Mesg_Hdr)
                    If Error equals NOERROR
                    Move IXIB_STAT to IXIB command portion of
                     message buffer referenced by Mesg_Hdr

                        Call Cmd_Out(Mesg_Hdr)
                    Endif
                Else
                    Call OI_Print(Error message)
                Endif
            Endif
        Endif
    Endif
Endloop
```

3.3.4.4.6.5 Limitations

There are no limitations defined for the XDD_Supervisor_Main Unit.

3.3.4.4.7 XDD_Tx_Main Unit

The XDD_Tx_Main Unit is responsible for the transmission of commands and IP datagrams to the IXIB interface. The commands and IP datagrams are obtained in ERTE messages from the XDD_Supervisor LLC. Two message queues are used: one for IXIB commands, and one for datagrams.

3.3.4.4.7.1 Inputs

The following inputs are received by the XDD_Tx_Main Unit:

1) Mesg_Hdr - This item is the header of an ERTE Message used to receive IXIB commands and datagrams from the XDD_Supervisor LLC. Command messages are read from the command message queue, and datagram messages are read from the datagram message queue.

3.3.4.4.7.2 Outputs

The following outputs are produced by the XDD_Tx_Main Unit:

1) IXIB_Mailbox - This output is the MailBox register of the IXIB interface. It specifies the type of command or data to be transmitted to the IXIB board.

2) IXIB_Fifo - This item is the IXIB FIFO register used to transfer data to the IXIB board.

3.3.4.4.7.3 Local Data

The following local data is defined for the XDD_Tx_Main Unit:

1) Error - See local data description in the XDD_Transmitter LLC section.

3.3.4.4.7.4 Processing

```
Loop
        Error = Message_Receive(Mesg_Hdr, IXIB_Tx_Cmd)
        If Error is equal NOERROR
                Move command byte at first location in message buffer
                 referenced by Mesg_Hdr to Cmd
        Else if Error is equal M_QEMPTY
                Error = Message_Receive(Mesg_Hdr, IXIB_Tx_IP)
                If Error is equal NOERROR
                        Move the logical OR of IXIB_IP_DATA and
                         IXIB_MORE to Cmd
                Else
                        If Error is equal M_QEMPTY
                                Call Wait_Event(MSG_ARRIVE)
                        Else
                                Call OI_Print(Error message)
                        Endif
                        Continue next loop iteration
                Endif
        Else
```

```
                Call OI_Print(Error message)
                Continue next loop iteration
        Endif
        While data remains in message buffer referenced by Mesg_Hdr
                For Cnt from 1 to IXIB_PSIZE
                        If M_Offset field of Mesg_Hdr >= M_Length field
                         of Mesg_Hdr
                                Exit Loop
                        Endif
                        Move current byte  at M_Addr + M_Offset in
                         Mesg_Hdr from message buffer to IXIB_Fifo
                        Increment M_Offset field of Mesg_Hdr
                Endfor
                If M_Offset field of Mesg_Hdr >= M_Length field
                 of Mesg_Hdr
                        Clear IXIB_MORE_BIT of Cmd
                Endif
                Load Mailbox with Cmd and length of current partial
                 packet
                Error = Wait_Timeout(IXIB_TX_INTR, IXIB_TX_TO)
                If Error not equal IXIB_TX_INTR
                        Call Panic(error message)
                Endif
        Endwhile
        Error = Message_Discard(Mesg_Hdr)
        If Error isn't equal NOERROR
                Call printf(Error message)
        Endif
Endloop
```

3.3.4.4.7.5 Limitations

There are no limitations defined for the XDD_Tx_Main Unit.

3.3.5 Ethernet Device Driver (EDD) TLC

The Ethernet Device Driver TLC contains the software that is used to control the DEQNA Ethernet interface from the IGW. This software is used to process incoming and outgoing IP datagrams as well as implement the ARP protocol.

There is a separate copy of this device driver in IGW memory for each DEQNA board that is installed on the IGW.

The driver consists of two software LLCs, each of which runs as a separate process under ERTE. The Supervisor LLC forms a common interface to the rest of the rest of the IGW processes, while the Interrupt LLC handles the response to DEQNA interrupts.

3.3.5.1 Ethernet Device Driver TLC Architecture

The Ethernet Device Driver consists of the following LLCs and units as shown in Figure 3-5.

1) EDD_Supervisor - This LLC receives datagrams from the IP TLC and forwards them to the EDD_Interrupt LLC once an Ethernet address has been determined using the ARP protocol. Incoming datagrams from the Ethernet are also processed by this LLC. The EDD_Interrupt LLC forwards incoming datagrams to this LLC which forwards IP datagrams to the IP TLC and processes ARP datagrams internally. The following units comprise this LLC:

```
                           +-----+
                           | XDD |
                           | TLC |
                           +--+--+
                              |
            +-----------------+------------------+
            |                 |                  |
    +-------+------+ +--------+-------+ +--------+--------+
    | XDD_Receiver | | XDD_Supervisor | | XDD_Transmitter |
    |      LLC     | |       LLC      | |       LLC       |
    +-------+------+ +--------+-------+ +--------+--------+
            |                 |                  |
     +------+------+          |           +------+-------+
     | XDD_Rx_Main |          |           | XDD_Tx_Main |
     +-------------+          |           +-------------+
                              |
    +-----------+----------+--+------+-----------------+
    |           |          |         |                 |
+---+----+ +----+----+ +---+---+ +---+----+ +----------+----------+
| Cmd_In | | Cmd_Out | | IP_In | | IP_Out | | XDD_Supervisor_Main |
+--------+ +---------+ +-------+ +--------+ +---------------------+
```

Figure 3-4

1) ARP_Free - This unit is used to free an entry in the ARP table.

2) ARP_Input - This unit is used to process an incoming ARP packet.

3) ARP_Lookup - This unit is used to locate an entry in the ARP table.

4) ARP_New_Entry - This unit is used to add a new entry to the ARP table.

5) ARP_Request - This unit is used to send out an ARP request.

6) ARP_Resolve - This unit is used to resolve an IP to Ethernet address translation.

7) ARP_Timer - This unit is used to age entries in the ARP table.

8) EDD_Input - This unit is used by the EDD_Supervisor to process an incoming packet received by the EDD_Rx unit.

9) EDD_Main - This unit is the main unit for the EDD_Supervisor LLC. It performs initialization and the loops waiting for data to become ready to process by calling other routines.

10) EDD_Output - This unit is called by the EDD_Supervisor to process an outgoing packet obtained from the EDD Queue written to by IP.

2) EDD_Interrupt LLC - This LLC processes receive and transmit interrupts for the DEQNA. Packets received from the Ethernet are forwarded to the EDD_Supervisor, while packets from the EDD_Supervisor are transmitted.

1) EDD_Intr_Main - This unit is used to determine the cause of a DEQNA interrupt and call the appropriate unit to process the interrupt.

2) EDD_Rx - This unit is used to process interrupts caused by received data being available.

3) EDD_Tx - This unit is used to process interrupts caused by the sending of data to the DEQNA.

3.3.5.2 Global data

This section describes the format and contents if the data which is defined to be globally used between the units contained within the Ethernet Device Driver TLC. The following are defined:

1) Ether_Pkt - This structure describes the format of packets received and transmitted by the DEQNA. This structure is composed of the following fields:

   Ether_Dst - This 6 byte field contains the hardware ethernet address of the destination ethernet station.

   Ether_Src - This 6 byte field contains the hardware ethernet address of the source ethernet station.

   Ether_Type - This 16 bit field contains the protocol type of the packet. Protocol types supported by the IGW are:

   ETHER_IP - Internet Protocol.
   ETHER_ARP - Address Resolution Protocol.

   See 3) and 4) below.

   Ether_Data - This field contains between 42 and 1500 bytes of data.

2) ARP_Pkt - This structure describes the format of ARP packets. This structure is composed of the following fields:

   ARP_Fmt - This 16 bit field contains the format of the hardware address. For the IGW this field is defined as:

   ARP_FMT_ETHER (0x01) - Ethernet address format.

   ARP_Proto - This 16 bit field contains the format of the protocol address. Address formats supported by the IGW are:

ETHER_IP - Internet Protocol.
ETHER_ARP - Address Resolution Protocol.

See 3) and 4) below.

ARP_Hln - This 8 bit field contains the length of the hardware address expressed in bytes. For the IGW this field is defined as 6.

ARP_Pln - This 8 bit field contains the length of the protocol address expressed in bytes. For the IGW this field is defined as 4.

ARP_Op - This 16 bit field contains the operation code of the ARP packet. Valid values for this field are:

ARP_OP_REQUEST (0x01) - ARP Request packet.
ARP_OP_REPLY (0x02) - ARP Reply packet.

ARP_Snd_Ether - This 6 byte field contains the senders ethernet address.

ARP_Snd_IP - This 4 byte field contains the senders IP address.

ARP_Tar_Ether - This 6 byte field contains the target ethernet address.

ARP_Tar_IP - This 4 byte field contains the target IP address.

3) ETHER_IP - This constant defined as 0x0800 hex is used to specify that the Internet Protocol is being used.

4) ETHER_ARP - This constant defined as 0x0806 hex is used to specify the that Address Resolution Protocol is being used.

5) AT_BUCKET_SIZE - This constant defined as 10 defines the number of ARP_Table entries in a bucket.

6) AT_NUM_BUCKET - This constant defined as 19 defines the number of bucket in ARP_Table.

7) ETHER_MIN_SIZE - This constant defined as 42 defines the minimum number of bytes in an Ethernet packet.

3.3.5.3 Ethernet Device Driver LLCs

The following LLCs are defined for the Ethernet Device Driver TLC:

1) The EDD_Supervisor LLC - This LLC is used for communication between the Ethernet Device Driver LLC and the IP TLC. In addition to this, ARP is implemented within this LLC.

2) The EDD_Interrupt LLC - This LLC is used to process interrupts generated by the DEQUNA interface.

These LLCs are described in the following subsections.

3.3.5.3.1 EDD_Interrupt LLC

The EDD_Interrupt LLC is responsible for the processing of interrupts generated by the DEQNA interface. The Units that the EDD_Interrupt LLC is composed of are:

1) The EDD_Intr_Main Unit

2) The EDD_Rx Unit

3) The EDD_Tx Unit

3.3.5.3.1.1 Inputs

The following inputs are used by the EDD_Interrupt LLC:

1) Mesg_Hdr - This input is the header of an ERTE Message for messages received from the EDD_Supervisor LLC for transmission to the DEQNA.

2) DEQNA_CSR - This input is the control and status register for the DEQNA device.

3.3.5.3.1.2 Outputs

The following outputs are produced by the EDD_Interrupt LLC:

1) Mesg_Hdr - This is the header of an ERTE Message for messages received from the EDD_Supervisor LLC for transmission to the DEQNA.

2) DEQNA_CSR - This output is the control and status register for the DEQNA device interface.

3) DEQNA_RX_BDL - This output is the device register where the start of the receive Buffer Descriptor List (BDL) is written to begin a receive operation.

4) DEQNA_TX_BDL - This output is the device register where the start of the transmit Buffer Descriptor List (BDL) is written to begin a transmit operation.

3.3.5.3.1.3 Local Data

The following local data is defined for the EDD_Interrupt LLC:

1) BDL_Tx - This local data item contains the Transmit Buffer Descriptor List. This list is made up of two entries. Each entry contains the following fields:

   1. BD_Flag - This field is 16 bits and is made up of the following flags:

      BDF_INIT (0x8000) - Initialized value
      BDF_USED (0xc000) - DEQNA is using buffer

   2. BD_Addr_Descriptor_Bits - This field is 10 bits and is made up of the following bits:

      BD_ADB_VALID (0x8000) - Valid address
      BD_ADB_EOM (0x2000) - End of message

   3. BD_Addr_Bits - This field is 22 bits and contains the address of the buffer associated with this descriptor.

   5. BD_Buffer_Length - This field is 16 bits and contains the length of the buffer.

   6. BD_Status_1 - This field is 16 bits and contains the first status word. This status word contains the following bits:

      BD_STAT_ERR_USED (0x4000) - An error has occurred.

   7. BD_Status_2 - This field is 16 bits and contains second status word. This status word is unused by the EDD_Interrupt LLC.

2) BDL_Rx - This local data item contains the Receive Buffer Descriptor List. This list is made up of two entries. Each entry contains the same fields as described for the BDL_Tx local data item.

3.3.5.3.1.4 Processing

The EDD_Interrupt LLC performs the following functions:

1) Receives, from the EDD_Supervisor, packets to be transmitted to the DEQNA.
2) Sends packets to the DEQNA.
3) Sends packets to the EDD_Supervisor LLC for processing.
4) Receives packets from the DEQNA.
5) Processes nonexistent memory interrupts.

3.3.5.3.1.5 Limitations

No limitations are defined for the EDD_Interrupt LLC.

3.3.5.3.2 EDD_Supervisor LLC

The EDD_Supervisor LLC is responsible for the communication between the EDD TLC and the IP TLC. In addition to this, ARP is implemented within this LLC. The units that the EDD_Supervisor LLC is composed of are:

1) The EDD_Main Unit.
2) The EDD_Input Unit.
3) The EDD_Output Unit.
4) The ARP_Free Unit.

5) The ARP_Input Unit.

6) The ARP_Lookup Unit.

7) The ARP_New_Entry Unit.

8) The ARP_Request Unit.

9) The ARP_Resolve Unit.

10) The ARP_Table_Hash.

11) The ARP_Timer Unit.

3.3.5.3.2.1 Inputs

The following inputs are used by the EDD_Supervisor LLC:

1) Mesg_Hdr - This input is the header of an ERTE Message used to contain packets arriving from the IP TLC or the EDD_Interrupt LLC.

3.3.5.3.2.2 Outputs

The following outputs are produced by the EDD_Supervisor LLC:

1) Mesg_Hdr - This input is the header of an ERTE Message used to contain packets being sent to the IP TLC or the EDD_Interrupt LLC.

3.3.5.3.2.3 Local Data

The following local data is defined for the EDD_Supervisor LLC:

1) ARP_Table - This local data item is a table of ARP address translation entries. Entries are grouped into AT_NUM_BUCKET buckets each containing AT_BUCKET_SIZE entries. Each entry in this table contains the following fields:

   1. AT_IP_Addr - This field consists of a 32 bit value containing the IP address for the ARP translation entry.

   2. AT_Ether_Addr - This field consists of a 6 byte value containing the Ethernet address for the ARP translation entry.

   3. AT_Flags - This field consists of an 8 bit value containing various flags describing the ARP translation entry. These flags are:

      AT_INUSE (0x01) - This entry is being used.

      AT_COM (0x02) - This entry is complete (Ethernet address is valid).

   4. AT_Timer - This field consists of an 8 bit value used to store the age of ARP translation entries in minutes.

   5. AT_Mess_Hdr - This field is a Message_Header structure used to hold a copy of the message header of the last message buffer that was attempted to be sent before a completed ARP translation entry was available.

3.3.5.3.2.4 Processing

The EDD_Supervisor LLC performs the following functions:

1) Receives IP packets from the IP TLC and forwards these packets with correct Ethernet addresses to the EDD_Interrupt LLC for transmission.

2) Receives IP packets from the EDD_Interrupt LLC and forwards them to the IP TLC.

3) Implements the ARP protocol.

3.3.5.3.2.5 Limitations

No limitations are defined for the EDD_Supervisor LLC.

3.3.5.4 Ethernet Device Driver Units

The following subsections contain the unit descriptions for the units comprising the Ethernet Device Driver TLC.

3.3.5.4.1 ARP_Free Unit

The ARP_Free Unit performs the function of freeing up an entry in the ARP table used for address translation by the EDD TLC.

3.3.5.4.1.1 Inputs

The following inputs are used by the ARP_Free Unit:

1) ARP_Entry - This 32 bit input parameter is the address of the entry in ARP_Table that is to be freed.

3.3.5.4.1.2 Outputs

The following outputs are produced by the ARP_Free Unit:

1) ARP_Entry - The global ARP_Table entry referenced by this 32 bit pointer is freed by clearing the following fields of the entry:

   - AT_Hold
   - AT_Timer
   - AT_Flags
   - AT_IP_Addr

3.3.5.4.1.3 Local Data

No local data is defined for the ARP_Free Unit.

3.3.5.4.1.4 Processing

```
If M_addr of the AT_Mess_Hdr field of ARP_Entry isn't null
        Call Message_Discard(address of AT_Mess_Hdr of the
          ARP_Entry)
        Clear M_addr of the AT_Mess_Hdr field of ARP_Entry
Endif
Clear AT_Timer field of ARP_Entry
Clear AT_Flags field of ARP_Entry
Clear AT_IP_Addr field of ARP_Entry
Return
```

3.3.5.4.1.5 Limitations

No limitations are defined for the ARP_Free Unit.

3.3.5.4.2 ARP_Input Unit

The ARP_Input Unit processes all incoming ARP packet that have been received from the DEQNA interface.

3.3.5.4.2.1 Inputs

The following inputs are used by the ARP_Input Unit:

1) Mesg_Hdr - This 32 bit input parameter is a pointer to ERTE Message header for the incoming ARP packet to be processed by this unit.

2) ARP_Table - This global input item is the ARP address translation table as defined in the EDD_Supervisor LLC local data section.

3) IP_Local - This global item (EDD_Supervisor LLC local data) is the IP address of the gateway on this Ethernet.

4) Ether_Local - This global item (EDD_Supervisor LLC) is the Ethernet address of the gateway on this Ethernet.

3.3.5.4.2.2 Outputs

The following outputs are produced by the ARP_Input Unit:

1) ARP_Table - This output consists of the ARP address translation table as defined in the EDD_Supervisor LLC local data section.

3.3.5.4.2.3 Local Data

The following local data is defined for the ARP_Input Unit:

1) IS_Addr - This 32 bit item contains the source IP address obtained from the incoming ARP packet.

2) IT_Addr - This 32 bit item contains the destination IP address obtained from the incoming ARP packet.

3) ARP_Entry - This 32 bit item is a pointer to an entry in ARP_Table.

3.3.5.4.2.4 Processing

```
If (M_Length field of Mesg_Hdr is less then ETHER_MIN_SIZE) or
 (Protocol_Type field in ARP packet isn't ETHER_IP) or
 (Source Ethernet address in ARP packet equals Ether_Local)
        Call Message_Discard(Mesg_Hdr)
        Return
Endif
If source IP address in ARP packet is equal IP_Local
        Call Message_Discard(Mesg_Hdr)
        Return
Endif
Move ARP_Snd_IP field of ARP packet to IS_Addr
Move ARP_Tar_IP field of ARP packet to IT_Addr
ARP_Entry = ARP_Lookup(IS_Addr)
If ARP_Entry isn't Null
        If AT_COM bit isn't set in AT_Flags field of ARP_Entry
                Move ARP_Snd_Ether field of ARP packet to AT_Ether_Addr
                 field of ARP_Entry
                Set AT_COM bit of AT_Flags field of ARP_Entry
                If (M_addr of the AT_Mess_Hdr field of ARP_Entry
                  isn't null)
                        Call EDD_Output(address of AT_Mess_Hdr
                          field, ETHER_IP)
                        Clear (M_addr of the AT_Mess_Hdr field)
                Endif
        Endif
Else if IT_Addr equals IP_Local
        ARP_Entry = ARP_New_Entry(IS_Addr)
        Move ARP_Snd_Ether field of ARP packet to AT_Ether_Addr field of
         ARP_Entry
        Set AT_COM bit of AT_Flags field of ARP_Entry
Endif
IF ARP_OP field of ARP packet isn't ARP_OP_REQUEST
        Call Message_Discard(Mesg_Hdr)
        Return
Endif
If IT_Addr isn't equal IP_Local
        Call Message_Discard(Mesg_Hdr)
        Return
Endif
Move ARP_Snd_Ether field of ARP packet to ARP_Tar_Ether field of
  ARP packet
Move ARP_Snd_IP field of ARP packet to ARP_Tar_IP field of
  ARP packet
Move Ether_Local to ARP_Snd_Ether field of ARP packet
Move IT_Addr to ARP_Snd_IP field of ARP packet
Move ARP_Tar_Ether field of ARP packet to Ether_Dst field of
  Ethernet packet
Move ARP_Snd_Ether field of ARP packet to Ether_Src field of
  Ethernet packet.
Move ETHER_ARP to Ether_Type field of Ethernet packet
Set M_Length field of Mesg_Hdr to ETHER_MIN_SIZE
Decrement M_Offset field Msg_Hdr by the size of the Ethernet header
Call Message_Send(Mesg_Hdr, DEQNA_TX)
Return
```

3.3.5.4.2.5 Limitations

No limitations are defined for this unit.

3.3.5.4.3 ARP_Lookup Unit

The ARP_Lookup Unit is used to locate an ARP entry in the ARP address translation table. This unit when given an IP address will return the address of the ARP_Table entry that contains the address translation information for that address.

3.3.5.4.3.1 Inputs

The following inputs are used by the ARP_Lookup Unit:

1) IP_Dest - This input is passed as a 32 bit parameter to this unit and contains the IP address of the ARP entry that is to be located in ARP_Table.

2) ARP_Table - This global item is the ARP address translation table defined in the EDD Supervisor LLC section.

3.3.5.4.3.2 Outputs

The following outputs are produced by the ARP_Lookup Unit:

1) ARP_Entry - This output is the return parameter of this unit and contains the ARP_Table address of the requested ARP entry or 0 if no matching entry was located.

3.3.5.4.3.3 Local Data

The following local data items are defined for the ARP_Lookup Unit:

1) AT_Index - This local data item contains the index of the bucket in ARP_Table corresponding to the requested IP address.

3.3.5.4.3.4 Processing

```
AT_Index = ARP_Table_Hash(IP_Dest) * AT_BUCKET_SIZE
ARP_Entry = Address of entry in ARP_Table indexed by AT_Index
For N = 1 to AT_BUCKET_SIZE
        If AT_IP_Addr field of ARP_Entry equals IP_Dest
                Return ARP_Entry
        Endif
        Increment ARP_Entry to point to next entry
Endfor
Return 0
```

3.3.5.4.3.5 Limitations

No limitations are defined for the ARP_Lookup Unit.

3.3.5.4.4 ARP_New_Entry Unit

The ARP_New_Entry unit is used to add a new partial entry to the ARP address translation table. If no empty slot is found in the appropriate bucket, the oldest entry in that bucket is replaced with the new entry.

3.3.5.4.4.1 Inputs

The following inputs are used by the ARP_New_Entry Unit:

1) IP_Dest - This input is passed as a 32 bit parameter to this unit and contains the IP address of the entry that is to added to the ARP table.

2) ARP_Table - This global item is the ARP address translation table (defined in the XDD_Supervisor LLC section).

3.3.5.4.4.2 Outputs

The following outputs are produced by the ARP_New_Entry Unit:

1) ARP_Table - This global item is the ARP address translation table (defined in the XDD_Supervisor LLC section).

3.3.5.4.4.3 Local Data

The following local data is defined for the ARP_New_Entry Unit:

1) ARP_Oldest_Entry - This local data item contains a pointer to the oldest ARP_Table entry.

2) Oldest - This local data item contains the age of the oldest ARP_Table entry.

3) AT_Index - This local data item contains the index of the bucket in ARP_Table corresponding to the IP address of the ARP entry to be added.

4) ARP_Entry - This local data item is used as a pointer to step through a bucket in ARP_Table.

3.3.5.4.4.4 Processing

```
Clear ARP_Oldest_Entry
Move -1 to Oldest
AT_Index = ARP_Table_Hash(IP_Dest) * AT_BUCKET_SIZE
ARP_Entry = Address of entry in ARP_Table indexed by AT_Index
For N = 1 to AT_BUCKET_SIZE
        If AT_Flags field of ARP_Entry is 0
                Move IP_Dest to AT_IP_ADDR field of ARP_Entry
                Move AT_INUSE flags to AT_Flags field of ARP_Entry
                Return ARP_Entry
        Endif
        If AT_Timer field of ARP_Entry is greater than Oldest
                Move ARP_Entry to ARP_Oldest_Entry
                Move AT_Timer field of ARP_Entry to Oldest
        Endif
        Increment ARP_Entry to point to next entry
Endfor
If ARP_Oldest_Entry is Null
        Return Null
Endif
Call ARP_Free(ARP_Oldest_Entry)
Move IP_Dest to AT_IP_ADDR field of ARP_Oldest_Entry
Move AT_INUSE field to AT_Flags field of ARP_Oldest_Entry
Return ARP_Oldest_Entry
```

3.3.5.4.4.5 Limitations

No limitations are defined for the ARP_New_Entry Unit.

3.3.5.4.5 ARP_Request Unit

The ARP_Request Unit sends out an ARP request packet requesting the Ethernet address corresponding to a specified IP address.

3.3.5.4.5.1 Inputs

The following inputs are used by the ARP_Request Unit:

1) IP_Dest - This input is passed as a 32 bit input parameter to this unit and contains the IP address of the host whose Ethernet address is desired.

2) IP_Local - This global item (EDD_Supervisor LLC local data) is the IP address of the gateway on this Ethernet.

3) Ether_Local - This global item (EDD_Supervisor LLC local data) is the Ethernet address of the gateway on this Ethernet.

3.3.5.4.5.2 Outputs

The following outputs are produced by the ARP_Request Unit:

1) Mesg_Hdr - This item contains the header of an ERTE Message containing the outgoing ARP request packet.

3.3.5.4.5.3 Local Data

The following local data is defined for the ARP_Request Unit:

1) Error - This local data item contains the error code obtained from the Message_Get function call.

3.3.5.4.5.4 Processing

```
Error = Message_Get(Mesg_Hdr)
If Error isn't NOERROR
        Call OI_Print(Error message)
        Return
Endif
Set M_Offset field of Mesg_Hdr to 0
Set M_Length field of Mesg_Hdr to the size of a completed ARP datagram
Move Ethernet broadcast address to Ethernet Destination address field
  in message buffer referenced by Mesg_Hdr
Move ETHER_ARP to Type in message buffer referenced by Mesg_Hdr
Move ARP_FMT_ETHER to ARP_Fmt field of ARP packet in message buffer
Move ETHER_IP to ARP_Proto field of ARP packet in message buffer
Move 6 to ARP_HLN field of ARP packet in message buffer
Move 4 to ARP_PLN field of ARP packet in message buffer
Move ARP_OP_REQUEST to ARP_Op field of ARP packet in message buffer
Move Ether_Local to ARP_Snd_Ether field of ARP packet in message buffer
Move IP_Local to ARP_Snd_IP field of ARP packet in message buffer
Move IP_Dest to ARP_Tar_IP field of ARP packet in message buffer
Call Message_Send(Mesg_Hdr, DEQNA_TX)
Return
```

3.3.5.4.5.5 Limitations

No limitations are defined for the ARP_Request Unit.

3.3.5.4.6 ARP_Resolve Unit

The ARP_Resolve is used to resolve IP to ethernet address translations. If an address resolution entry is not found in the address resolution table an ARP request sequence is initiated.

3.3.5.4.6.1 Inputs

The following inputs are used by the ARP_Resolve Unit:

1) Mesg_Hdr - This input is passed as a 32 bit parameter and contains the header of the ERTE Message containing the IP packet that caused the ARP_Resolve Unit to be called.

2) IP_Dest - This input is passed as a 32 bit parameter and contains the IP address that the Ethernet address is desired for.

3) Ether_Local - This global item (EDD_Supervisor LLC) it the local Ethernet address of the gateway on this Ethernet.

4) ARP_Table - This global item (EDD_Supervisor LLC) is the ARP address translation table.

3.3.5.4.6.2 Outputs

The following outputs are produced by the ARP_Resolve Unit:

1) Ether_Dest - This output contains the Ethernet address corresponding to the IP_Dest IP address input.

2) TRUE - This constant is returned as a return parameter to indicate that the Ethernet address was found for the input IP address.

3) FALSE - This constant is returned as a return parameter to indicate that the Ethernet address was not found for the input IP address.

3.3.5.4.6.3 Local Data

The following local data is defined for the ARP_Resolve Unit:

1) ARP_Entry - This local data item is a pointer to an entry in ARP_Table.

3.3.5.4.6.4 Processing

```
If host portion of IP_Dest is 0
        Move Ethernet broadcast address to Ether_Dest
        Return TRUE
Endif
If IP_Dest is local IP address
        Move Ether_Local to Ether_Dest
        Return TRUE
Endif
Arp_Entry = ARP_Lookup(IP_Dest)
If Arp_Entry is null
        Arp_Entry = ARP_New_Entry(IP_Dest)
        Move Mesg_Hdr to AT_Hold field of ARP_Entry
        Call ARP_Request(IP_Dest)
        Return FALSE
Endif
Clear AT_Timer field of ARP_Entry
```

```
If AT_COM bit is set in AT_Flags field of ARP_Entry
        Move AT_Ether_Addr field of ARP_Entry to Ether_Dest
        Return TRUE
Endif
If AT_Hold field of ARP_Entry isn't null
        Call Message_Discard(Mesg_Hdr)
Endif
Copy Mesg_Hdr to AT_Mess_Hdr field of ARP_Entry
Call ARP_Request(IP_Dest)
Return FALSE
```

3.3.5.4.6.5 Limitations

No limitations are defined for the ARP_Resolve Unit.

3.3.5.4.7 ARP_Table_Hash Unit

The ARP_Table_Hash unit applies a hash function to an IP address to select an entry in the ARP address translation table.

3.3.5.4.7.1 Inputs

The following inputs are used by the unit:

1. IP_Addr - This 32 bit input parameter contains the IP address to use to select the ARP table entry.

2. ARP_Table - This global item is the ARP address translation table (see EDD_Supervisor LLC).

3.3.5.4.7.2 Outputs

The following outputs are produced by the unit:

1. Bucket - The bucket index into ARP_Table generated by the hash function.

3.3.5.4.7.3 Local Data

No local data is defined for this unit.

3.3.5.4.7.4 Processing

```
Move IP_Addr to Bucket
Right shift Bucket 16 bit positions
Exclusive or Bucket with IP_Addr
Bit and Bucket with 0x7fff (hex)
Bucket = Bucket Mod AT_NUM_BUCKET
Return Bucket
```

3.3.5.4.7.5 Limitations

There are no limitations defined for this unit.

3.3.5.4.8 ARP_Timer Unit

The ARP_Timer Unit is called once a minute to age ARP table entries. Incomplete ARP table entries that are 3 minutes old or older are removed. Completed entries that have not been used in the last 20 minutes are also removed.

3.3.5.4.8.1 Inputs

The following inputs are used by the ARP_Timer Unit:

1) ARP_Table - This global item is the ARP address translation table (see EDD_Supervisor LLC).

3.3.5.4.8.2 Outputs

The following outputs are produced by the ARP_Timer Unit:

1) ARP_Table - This global item is the ARP address translation table (see EDD_Supervisor LLC).

3.3.5.4.8.3 Local Data

The following local data is defined for the ARP_Timer Unit:

1) ARP_Entry - This 32 bit item is a pointer to an entry in ARP_Table.

3.3.5.4.8.4 Processing

```
For each table entry ARP_Entry in ARP_Table
    If AT_Flags field of ARP_Entry is not 0
        Increment AT_Timer field of ARP_Entry
        If AT_COM bit is set in AT_Flags field of ARP_Entry
            If AT_Timer field of ARP_Entry >= 20
                Call ARP_Free(ARP_Entry)
            Endif
        Else if AT_Timer field of ARP_Entry >= 3
            Call ARP_Free(ARP_Entry)
        Endif
    Endif
Endfor
Return
```

3.3.5.4.8.5 Limitations

No limitations are defined for the ARP_Timer Unit.

3.3.5.4.9 EDD_Input Unit

The EDD_Input Unit processes all incoming packets from the EDD_Interrupt LLC. IP packets are sent to IP_QUEUE, and ARP_Input is called to process ARP packets.

3.3.5.4.9.1 Inputs

The following inputs are used by the EDD_Input Unit:

1) Mesg_Hdr - This 32 bit parameter input is the pointer to the header of an ERTE Message containing the packet that has been received.

3.3.5.4.9.2 Outputs

The following outputs are produced by the EDD_Input Unit:

1) Mesg_Hdr - This 32 bit item is the pointer to the header of an ERTE Message containing the packet that has been received and is to be sent on.

3.3.5.4.9.3 Local Data

No local data is defined for the EDD_Input Unit.

3.3.5.4.9.4 Processing

```
Increment M_Offset field of Mesg_Hdr to skip Ethernet header
If Type field of Ethernet header = ETHER_IP
        Call Message_Send(Mesg_Hdr, IP_QUEUE)
Else if Type field of Ethernet header = ETHER_ARP
        Call ARP_Input(Mesg_Hdr)
Endif
Return
```

3.3.5.4.9.5 Limitations

No limitations are defined for the EDD_Input Unit.

3.3.5.4.10 EDD_Intr_Main Unit

The EDD_Intr_Main unit is signaled to run whenever a device interrupt occurs. A device interrupt is caused by data being received, data being successfully transmitted, or by a nonexistent memory error.

3.3.5.4.10.1 Inputs

The following inputs are used by the EDD_Intr_Main Unit:

1) DEQNA_CSR - This input is read from the DEQNA Control Status Register. See the DEQNA ETHERNET User's Guide for the format of this register.

2) EDD_Start - This input indicates if the transmitter needs to be kick started to begin data transmission. EDD_Start is defined in the EDD_Interrupt LLC.

3.3.5.4.10.2 Outputs

The following outputs are produced by the EDD_Intr_Main Unit:

1) DEQNA_CSR - This output is written to the DEQNA Control Status Register. See the DEQNA ETHERNET User's Guide for the format of this register.

3.3.5.4.10.3 Local Data

The following local data is defined for the EDD_Intr_Main Unit:

1) Csr - This local data item is used to store the DEQNA Control Status Register.

2) Mesg_Hdr - This local data item holds the message header for the initial receive message buffer.

3) Error - This local data item is used to store the error code returned by the Message_Get function.

3.3.5.4.10.4 Processing

```
Clear BD_Addr_Descriptor_Bits in BDL_Tx entries
Clear BD_Addr_Descriptor_Bits in BDL_Rx entries
Error = Message_Get(Mesg_Hdr)
If Error equals NOERROR
        Move physical address of message buffer referenced
         by Mesg_Hdr to BD_Addr_Bits in First BDL_Rx entry
Else
        Call Panic(Error message)
Endif
Move the logical OR of EDD_RECV_ENABLE, EDD_INT_ENABLE,
 EDD_XMIT_INT, EDD_RCV_INT, and QE_ILOOP to DEQNA_CSR
Move TRUE to EDD_Start
Loop
        Events = Wait_Event(logical OR of EDD_INTR and MSG_ARRIVE)
        If EDD_INTR bit is set in Events
                Move DEQNA_CSR to Csr
                Move logical OR of EDD_RECV_ENABLE,
                 EDD_INT_ENABLE, EDD_XMIT_INT, EDD_RCV_INT, and
                 QE_ILOOP to DEQNA_CSR
                If Csr & EDD_RCV_INT
                        Call EDD_Rx_()(address of Mesg_Hdr)
                If Csr & EDD_XMIT_INT
                        Call EDD_Tx_()(address of EDD_Start_
                If Csr & EDD_NEX_MEM_INT
                        Call Panic(Error message)
        Else if EDD_Start equals TRUE
                Call EDD_Tx()(address of EDD_Start)
        Endif
Endloop
```

3.3.5.4.10.5 Limitations

No limitations are defined for the EDD_Intr_Main Unit.

3.3.5.4.11 EDD_Main Unit

The EDD_Main Unit performs EDD initialization and calls the appropriate units to transfer data between the IP TLC and the EDD_Interrupt LLC.

3.3.5.4.11.1 Inputs

The following inputs are used by the EDD_Main Unit:

1) Mesg_Hdr - This input is a header for an ERTE Message which contains Ethernet datagrams from the IP TLC or the EDD_Interrupt LLC.

2) Ether_Addr_Prom - This item is a six 16 bit word register used to discover the interface's Ethernet address. See DEQNA ETHERNET User's Guide for description.

3) Save_Time - This 32 bit integer is used to obtain the gateway time.

4) Cur_Time - This 32 bit integer is used to obtain the gateway time.

3.3.5.4.11.2 Outputs

The following outputs are produced by the EDD_Main Unit:

1) IP_Local - This global item is written with the local IP address of the gateway on this Ethernet (see EDD_Supervisor LLC).

2) Ether_Local - This global item is written with the local Ethernet address of the gateway on this interface (see EDD_Supervisor LLC).

3.3.5.4.11.3 Local Data

The following local data is defined for the EDD_Main Unit:

1) ARP_Timeout - This local data item contains the time before the ARP_Timer unit is to be called.

2) Error - This local data item is used to hold the error value returned by the Message_Receive unit.

3) Event - This local data item is used to determine if the MSG_ARRIVE event occurred before a timeout did.

3.3.5.4.11.4 Processing

```
Move IP address of IGW on interface to IP_Local
Move Ethernet address of interface from Ether_Addr_Prom
 to Ether_Local
Move 60 * CLOCK_INT to ARP_Timeout
Save_Time = Get_Time()
Loop
        Error = Message_Receive(EDD_RX, Mesg_Hdr)
        If Error equals NOERROR
                Call EDD_Input(Mesg_Hdr)
        Else if Error equals M_QEMPTY
                Error = Message_Receive(EDD_QUEUE, Mesg_Hdr)
                If Error equals NOERROR
                        Call EDD_Output(Mesg_Hdr, ETHER_IP)
                Else if Error equals M_QEMPTY
                        Event = Wait_Timeout(MSG_ARRIVE, ARP_Timeout)
                        If Event isn't equal 0
                                Cur_Time = Get_Time()
                                Add Save_Time to ARP_Timeout
                                Subtract Cur_Time from ARP_Timeout
                                Move Cur_Time to Save_Time
                                If ARP_timeout greater than 0
                                        Continue next loop iteration
                                Endif
                        Endif
                        Move 60 * CLOCK_INT to ARP_Timeout
                        Call ARP_Timer()
                Else
                        Call OI_Print(Error Message)
                        Continue next loop iteration
                Endif

        Else
                Call OI_Print(Error message)
                Continue next loop iteration
        Endif
Endloop
```

3.3.5.4.11.5 Limitations

No limitations are defined for the EDD_Main Unit.

3.3.5.4.12 EDD_Output Unit

This unit is called by to send a packet to the EDD_Tx Unit for transmission to the DEQNA.

3.3.5.4.12.1 Inputs

The following inputs are used by the EDD_Output Unit:

1) Mesg_Hdr - This 32 bit input parameter is a pointer to the header of the message buffer containing the packet that is to be transmitted.

2) Type - This input is passed to this unit as a parameter and contains the type of Ethernet packet that is to be sent.

3) Ether_Local - This global item contains the Ethernet address of the gateway on this Ethernet.

3.3.5.4.12.2 Outputs

The following outputs are produced by the EDD_Output Unit:

1) Mesg_Hdr - This 32 bit item is a pointer to the header of the message buffer containing the packet that is to be transmitted.

3.3.5.4.12.3 Local Data

The following local data items are defined for this unit:

1) IP_Dest - This 32 bit item contains the IP address for the destination of the packet.

2) EN_Dest - This 48 bit item contains the Ethernet address of the destination of the packet.

3.3.5.4.12.4 Processing

```
Move dest IP addr from buffer referenced by Mesg_Hdr to IP_Dest
Zero EN_Dest
Status = ARP_Resolve(Mesg_Hdr, IP_Dest EN_Dest)
If Status equals TRUE
    Move Ether_Local to message buffer referenced by Mesg_Hdr
    Move EN_Dest to message buffer referenced by Mesg_Hdr
    Move Type to type field in message buffer referenced by Mesg_Hdr
    Call Message_Send(DEQNA_TX, Mesg_Hdr)
Endif
Return
```

3.3.5.4.12.5 Limitations

No limitations are defined for the EDD_Output Unit.

3.3.5.4.13 EDD_Rx Unit

This unit is invoked whenever a packet is ready to be read from the DEQNA interface. This unit will transfer the incoming packet to the EDD_Supervisor LLC.

3.3.5.4.13.1 Inputs

The following inputs are used by the EDD_Rx Unit:

1) BDL_Rx - This global item is the Buffer Descriptor List for receiving Ethernet packets (see EDD_Interrupt LLC).

2) Mesg_Hdr - This item is used to hold the header of the ERTE message used to receive incoming Ethernet packets.

3.3.5.4.13.2 Outputs

The following outputs are produced by the EDD_Rx Unit:

1) Mesg_Hdr - This item is used to hold the header of the ERTE Message used to receive incoming Ethernet packets.

2) DEQNA_RX_BDL - This output is the Receive BDL Start Address Register for the Ethernet device, and contains the starting address of the receive buffer descriptor list (BDL_Rx)

3) BDL_Rx - This global item is the Buffer Descriptor List for receiving Ethernet packets (see EDD_Interrupt LLC).

3.3.5.4.13.3 Local Data

The following local data is defined for the EDD_Rx Unit:

1) Error - This local data item is used to hold the return error code from the Message_Send function.

3.3.5.4.13.4 Processing

```
If BD_STAT_ERR_USED bit of BD_Status_1 field of first entry in
 BDL_Rx is clear
        Call Message_Send(DEQNA_RX, Mesg_Hdr)
        Loop
                Error = Message_Get(Mesg_Hdr)
                If Error is NOERROR
                        Exit loop
                Else
                        Call Sleep(CLOCK_INT)
                Endif
        Endloop
Endif
Move physical address of message buffer (using m_addr) from
 Mesg_Hdr to BD_Addr_Bits field of first entry in BDL_Rx
Clear BD_Status_1 field of first entry in BDL_Rx
```

```
Move address of first entry in BDL_Rx to DEQNA_RX_BDL
Return
```

3.3.5.4.13.5 Limitations

No limitations are defined for the EDD_Rx Unit.

3.3.5.4.14 EDD_Tx Unit

The EDD_Tx Unit is invoked to send a packet to the DEQNA interface.

3.3.5.4.14.1 Inputs

The following inputs are used by the EDD_Tx Unit:

1) EDD_Start - This input is used to determine if there are any message buffers to be discarded.

3.3.5.4.14.2 Outputs

The following outputs are produced by the EDD_Tx Unit:

1) BDL_Tx - This is the Buffer Descriptor List for the transmissions of packets. See local data description in the EDD_Interrupt LLC section.

2) DEQNA_TX_BDL - This output is the Transmit BDL Start Address Register, which is written with the starting address of the transmit buffer descriptor list (BDL_Tx).

3) EDD_Start - This output is written with TRUE or FALSE depending if the EDD transmit interrupt handler should

be kick started when transmit buffers become available. EDD_Start is defined in the EDD_Interrupt LLC local data section.

3.3.5.4.14.3 Local Data

The following local data is defined for the EDD_Tx Unit:

1) Error - This local data item is used to hold the error code returned by the Message_Receive function.

2) Mesg_Hdr - This local data item is declared as static and contains the message header of the message buffer used to transmit packets.

3.3.5.4.14.4 Processing

```
If EDD_Start equals FALSE
        Call Message_Discard(Mesg_Hdr)
Endif
Error = Message_Receive(DEQNA_TX, Mesg_Hdr)
If Error isn't equal NOERROR
        Move TRUE to EDD_Start
        If Error isn't equal M_QEMPTY
                Call OI_Print(Error message)
        Endif
        Return
Endif
Move FALSE to EDD_Start
Move physical address of data referenced by Mesg_Hdr (m_addr +
 m_offset) to BD_Addr_Bits field of first entry in BDL_Tx
Move length of Ethernet packet (m_length field of Mesg_Hdr) to
 BD_Buffer_Length field of first entry in BDL_Tx
Move bitwise or of BD_ADB_VALID and BD_ADB_EOM to
 BD_Addr_Descriptor_Bits field of first entry in BDL_Tx
Move BDF_INIT to BD_Flag field of first entry in BDL_Tx
Move address of first entry in BDL_Tx to DEQNA_TX_BDL
Return
```

3.3.5.4.14.5 Limitations

No limitations are defined for the EDD_Tx Unit.

3.3.6 Console Device Driver (CDD) TLC

The console device driver TLC manages the access to the console device on the IGW. All messages sent to the driver from other IGW processes are presumed to be text to be displayed on the IGW operator's console. All input from the console is interpreted as operator commands which are forwarded to the operator interface. The driver assumes the console is running at 9600 bps, using 8 bit characters with no parity and one stop bit. The driver interprets and processes XON and XOFF, delete and backspace characters. The driver also interprets control-R as a retype current input line request.

3.3.6.1 Console Device Driver TLC Architecture

The driver consists of three IGW processes. The Supervisor process presents a common interface to the other IGW TLCs. The supervisor routes all console input and output to the appropriate IGW process for processing.

The Transmitter process accepts messages from the supervisor or the

Receiver process and transmits each character in the messages to the console device. The Transmitter process does not interpret the messages in any way except to insert an ASCII CR character ahead of every ASCII LF character.

The Receiver process accumulates input characters from the console device until an ASCII CR is read. The CR (which marks the end of a line) is converted into and ASCII LF, stored with the previously received characters of the input line, and the whole line is then sent in a message to the Supervisor. The receiver echoes each input character by sending it in a message to the transmitter. If a control-R character is read, then the Receiver will send a copy of the entire input line to the transmitter for display on the console. If XOFF is read, then the Receiver will set a CONSOLE_XOFF event. If XON is read the Receiver will clear the CONSOLE_XOFF event. The Receiver also interprets delete and backspace as character delete keys. The Receiver echoes a "backspace-space-backspace" sequence for character deletion.

The Console Device Driver consists of the following Units (Figure 3-6):

1) CDD_Supervisor - This unit performs the functions of the Supervisor process.

2) CDD_Transmitter - This unit performs the functions of the Transmitter process.

```
                               +------+
                               | CDD  |
                               | TLC  |
                               +--+---+
                                  |
      +-----------------+---------+---------+----------------+
      |                 |                   |                |
+------+-------+ +--------+--------+ +---------+---------+ +-----+-----+
| CDD_Receiver | | CDD_Supervisor  | | CDD_Transmitter  | | Error_Msg |
+--------------+ +-----------------+ +------------------+ +-----------+
```

Figure 3-6

3) CDD_Receiver - This unit performs the functions of the Receiver process.

4) Error_Msg - This unit is used by the CDD_Supervisor to prepare a console driver error message. The message is formatted and sent to the CDD_Transmitter unit.

3.3.6.2 Console Device Driver Global Data

No global data is defined for this TLC.

3.3.6.3 Console Device Driver LLCs

No LLCs are defined for this TLC.

3.3.6.4 Console Device Driver Units

The following four units are defined for the Console Device Driver TLC.

3.3.6.4.1 Error_Msg Unit

The Error_Msg unit prepares an error message in a message buffer and sends the message buffer to the CDD_Transmitter for display on the operator's console.

3.3.6.4.1.1 Inputs

The following input is used by the unit:

1) Message_String - This input parameter is a pointer to the string of characters to be displayed on the console.

3.3.6.4.1.2 Outputs

The following output is produced by the unit:

1) Error_Message - This output is the header of an ERTE Message which contains the error message. The message is sent to the CDD_Transmitter process.

3.3.6.4.1.3 Local Data

The following local data is defined for the unit:

1) Status - This item is a thirty-two bit integer used to hold a return value from a called unit.

3.3.6.4.1.4 Processing

```
Status = Message_Get(Error_Message)
If (Status != NOERROR)
    Return
Endif

Copy each character in Message_String to the message buffer
  referenced by the M_Addr field of Error_Message.
Set M_Length field to length of Error_Message.
Call Message_Send(Error_Message, CDD_TRANSMIT_QUEUE)
If result != NOERROR
     Message_Discard (ERROR_Message)
Endif
```

3.3.6.4.1.5 Limitations

No limitations are defined for the unit.

3.3.6.4.2 CDD_Receiver Unit

The CDD_Receiver unit receives characters from the console, stores them in a message buffer, and then sends the message buffer to CDD_Supervisor when an ASCII CR is typed. The unit also echoes each character typed by sending it to the CDD_Transmitter unit.

3.3.6.4.2.1 Inputs

The following inputs are used by the unit:

1) Receiver_Data - This input is the data register for the console receiver. Received characters are read from this register.

3.3.6.4.2.2 Outputs

The following outputs are produced by the unit:

1) Echo - This output is the header of an ERTE Message used to echo each typed character back to the operator's console.

2) Input_Line - This output is the header of an ERTE Message containing the line of input received from the console device. A line is terminated by an ASCII LF character.

3.3.6.4.2.3 Local Data

The following local data is defined for the unit:

1) Console_Character - This item is the eight bit character received from the console device via the console receive register.

2) Status - This thirty-two bit integer is used to hold returned status from called units.

3) Echo_Flag - This thirty-two bit integer is used to indicate whether or not to echo characters.

3.3.6.4.2.4 Processing

```
Echo_Flag = FALSE
NO_Ibuf = TRUE
Loop
    If NO_Ibuf
        While (Status != NOERROR)
            Status = Message_Get(Input_Line)
        Endwhile
    Else
        M_Length = 0
    Endif

    Loop

        If Echo_Flag = FALSE
            Status = Message_Get(Echo)
            If (Status != NOERROR)
                Echo_Flag = FALSE
            Else
                Echo_Flag = TRUE
            Endif

        Else
            M_Length = 0
        Endif

        Wait_Event(CDD_RCV)
        Set Console_Character to Receiver_Data

        Case Console_Character of


            XON    :     Set_Event(CONSOLE_XON)
        XOFF  :     Clear_Event(CONSOLE_XON)

        DEL or
        BS    :     IF M_Length of input line > 0
                        Subtract one from M_Length field of
                         Input_Line
                        If (Echo_Flag = TRUE)
                            Copy the string "BS SP BS" to Echo
                             message buffer
                        Set M_Length field of Echo to 3
                        Call Message_Send(CDD_TRANSMIT_QUEUE, Echo)
                          Echo_Flag = (Result of message send)
                    Endif

       CTRL_R :     If (Echo_Flag = TRUE)
                        Copy the string "LF" to message buffer
                         referenced by Echo message header
                        Add 1 to M_Length field of Echo
                        Copy all the characters in Input_Line
                         message buffer to end of Echo message
                         buffer
                        Add M_Length field of Input_Line to
                         M_Length field of Echo
                        Call Message_Send(CDD_TRANSMIT_QUEUE, Echo)
                         Echo_Flag = (Result of message send)
                    Endif

        CR:
                    Console_Character = LF
                    Perform Next Case

        otherwise : If (M_Length field of Input_Line < 255 or
                     Console_Character = LF)
                        Append Console_Character to Input_Line
                        Add 1 to M_Length field of Input_Line
                        If (Echo_Flag = TRUE)
                            Copy Console_Character to Echo message
                             buffer
                            Set M_Length field of Echo to 1
                            Call Message_Send(CDD_TRANSMIT_QUEUE,
                             Echo)
                            Echo_Flag = (Result of message send)
                        Endif
                    Endif

    Endcase

While(Console_Character != LF)
```

```
    Message_Send(CDD_INPUT_QUEUE, Input_Line)
    NO_Ibuf = (RESULT of message send)
While(TRUE)
```

3.3.6.4.2.5 Limitations

There are no limitations defined for this unit.

3.3.6.4.3 CDD_Supervisor Unit

The CDD_Supervisor unit functions as the interface unit for the console device of the IGW. All outputs to the console are sent by IGW processes to this unit, and all console inputs pass through this unit which then forwards them to the appropriate IGW process. This unit operates as an IGW process.

3.3.6.4.3.1 Inputs

The following inputs are used by the unit:

1) Console_Output - This input is the header of an ERTE Message containing a string of characters to be output to the console.

2) Console_Input - This is the header of an ERTE Message containing a line of input from the operators console. The line of input is at most 255 characters long and is terminated by an ASCII LF character. The message is received from the CDD_Receiver unit.

3.3.6.4.3.2 Outputs

The following outputs are produced by the unit:

1) Operator_Input - This output is the header of an ERTE Message containing a line of console input entered by the IGW operator. This message is sent to the Operator Interface (OI) process.

2) Operator_Output - This output is the header of an ERTE message containing a string of output characters to be sent to the operator's console. This message is sent to the CDD_Transmitter unit.

3) Receiver_CSR - This output is the device register for setting the control and status of the console receiver.

4) Transmitter_CSR - This output is the device register for setting the control and status of the console transmitter.

3.3.6.4.3.3 Local Data

The following local data is defined for the unit:

1) Status - This thirty-two bit integer is used to hold the status returned by units called by this unit.

3.3.6.4.3.4 Processing

```
Write Receiver_CSR to enable data reception and receive interrupts
Write Transmitter_CSR to enable data transmission and transmit
 interrupts

Status = Open_Message_Queue(CONSOLE_INPUT_QUEUE)
If (Status != NOERROR)
    Call Error_Msg("Can't open console input queue")
    Call Exit()
Endif

Status = Open_Message_Queue(CONSOLE_OUTPUT_QUEUE)
If (Status != NOERROR)
    Call Error_Msg("Can't open console output queue")
    Call Exit()
Endif
```

```
Loop

    Status = Message_Receive(CONSOLE_INPUT_QUEUE, Message)
    If (Status = M_EMPTY)
        Status = Message_Receive(CONSOLE_OUTPUT_QUEUE, Message)
        If (Status =M_QEMPTY)
            Wait_Event(MSG_ARRIVE)
        Else
            Message_Send(CDD_TRANSMIT_QUEUE, Message)
        Endif
    Else
        Message_Send(OI_QUEUE, Message)
    Endif

While (FOREVER)
```

3.3.6.4.3.5 Limitations

There are no limitations defined for this unit.

3.3.6.4.4 CDD_Transmitter Unit

The CDD_Transmitter Receiver unit receives message buffers from the CDD_Supervisor unit and writes the characters in the buffer to the console. If the character to be written is an ASCII LF, then the unit will send an ASCII CR to the console ahead of the LF. The unit also checks the event CONSOLE_XOFF if this event is set, then no output is forwarded until the event is cleared. The event is set and cleared by the CDD_Receiver unit.

3.3.6.4.4.1 Inputs

The following inputs are used by the unit:

1) Output_Message - This input is the header to an ERTE Message which contains a string of characters to be displayed on the console.

3.3.6.4.4.2 Outputs

The following outputs are produced by the unit:

1) Transmit_Data - This output is the device output data register for the console transmitter. Characters to be displayed on the console are written to this register.

3.3.6.4.4.3 Local Data

The following local data is defined for the unit:

1) Output_Character - This item is an 8 bit character to be written to the console.

2) Status - This thirty-two bit integer is used to hold returned status from called units.

3.3.6.4.4.4 Processing

```
Status = Open_Message_Queue(CDD_TRANSMIT_QUEUE)
If (Status != NOERROR)
    Call Exit()
Endif

Loop
    Loop
        Status = Message_receive(CDD_TRANSMIT_QUEUE,
         Output_Message)
        IF Status == M_QEMPTY
            Call Wait_Event(MSG_ARRIVE)
        Endif
    While Status != NOERROR

    For each character in message buffer of Output_Message
        Set Output_Character to the current character in buffer
        Call Wait_Event(CONSOLE_XON)
        If Output_Character = LF
            Set Transmit_Data to CR
            Wait_Event(CDD_XMIT)
        Endif
        Set Transmit_Data to Output_Character
        Wait_Event(CDD_XMIT)
    Endfor
    Message_Discard(Output_Message)

While(TRUE)
```

3.3.6.4.4.5 Limitations

The unit does not check or handle output errors.

SOFTWARE DETAILED DESIGN DOCUMENT FOR THE INTER-NETWORK GATEWAY PROJECT