Software Component Specification

SYSTEM:	MOD400
SUBSYSTEM:	LACS SOFTWARE
COMPONENT:	Connectionless (Type 1) Logical Link Layer Entity
PLANNED RELEASE:	Release 4.0
SPECIFICATION REVISION NUMBER:	2.1
DATE:	Aug. 31, 1986
AUTHOR:	Michael Lu

This specification describes the current definition of the subject software component. LACS LLC Software has been released on Aug. 21 1986. Further modification will be done after Dec. 1986 release.

Table of Contents

REFERENCES

[1]	IEEE Standard 802.1	Local Area Netwark Overview
[2]	IEEE Standard 802.2	Logical Link Control, 1985.
[3]	IEEE Standard 802.3	CSMA/CD, MAC 1985
[4]	IEEE Standard 802.4	Token Bus, MAC 1985.
[5]	IEEE Standard 802.5	Token Ring, MAC 1985.
[6]	ISO 7498-1984	Information Processing Systems Open systems interconnection
[7]	ISO/DIS 8348/DAD 1	Information Processing Systems Data communications, Network.
[8]	ISO/DIS 8348/DAD 2	Addendum to the Network Service Definition convering Network Layer addressing.
[9]	ISO/DP 8473	Information Processing Systems Data Communications, Protocal.
[10]	09-0016-00	ESPL Software Technical Reference Manual. vol 1. Kernel and Support Software (Bridge Communications Inc.)
[11]	60149817	LAN Software EPS-1.
[12]	60149766	Local Area Controller Subsystem (LACS) EPS-1.
[13]	60149824	DPS6 Local Area Network Controller PFS.
[14]	DSA-41	Local Area Networks.
[15]	ISO 7498	Open System Interconnection Reference Model.
[16]	MC68000UM	MC68000 User's Manual. (Motorola Inc.)

DEFINITIONS/ABBREVIATIONS

ACK	ACKnoledge
CCITT	International Telegram and Telephone Consultative Committee
CSMA/CD	Carrier Sense Multiple Access/Collision Detection
DA	Destination Address.
DIS	Draft International Standard.
DISC	DISConnected Mode.
DMA	Direct Memory Acess.
DTE	Data Terminal Equipment
FCS	Frame Check Sequence.
HDLC	High Level Data Link Control.
IEEE	Institue of Electrical and Electronic Engineers
ISO	International Organization for Standardization
LAN	Local Area Network.
LACS	Local Area Controller Subsystem.
LCB	Logical Control Block (DPS 6 Level 6 User)
LLC	Logical Link Control Layer.
LSAP	Logical Link Layer Service Acess Point
LSB	least Significant Bit.
MSB	Most Significant Bit.
OSI	Open System Interconnection.
PDU	Protocol Data Unit.
PRIVATE	Catagory of memory known to the communication kernel. It consists of the bank of memory local to the 68000 processor on the LACS.
RCV	LLC layer recieve process.
REJ	REJect.

SA	Source Address.
SAP	Service Access Point.
SHARED	Catagory of memory known to the communication kernel. It consists of the bank of memory local to the Layers on the LACS.
SSAP	Source Service Access Point.
TEST	TEST function on LLC Layer.
UA	Unnumbered Acknowledgment.
UI	Unnumbered Information.
XID	eXchange IDentification.
XMIT	Transmit process.

1. INTRODUCTION AND OVERVIEW

1.1 BACKGROUND

The Logical Link Control Layer for LACS oftware and Hardware development is designed and implemented to meet IEEE 802.2 standard. The logical link layer is the conceptual layer of control or processing logic existing in the hierachical structure of a station that is responsible for maintaining control of the data link. The logical link layer functions provide an interface between the station higher layer logic and the data link. These functions include address/control field interpretation, channel access and command PDU/response PDU generation, transmission and interpretation.

The IEEE 802 standard identifies two distinct "classes" of LLC operation Currently, we support Class I which provides data-link-connectionless service only. (Class II provides data-link-connection-oriented service plus data-link-connectionless service, this will be support later).

In the IEEE 802 model, the LLC sublayer is responsible for supporting logical access points called LSAPs. The sublayer accepts write primitives from the Network layer above and formats them into write primitives for the MAC sublayer below. Incoming data in the form of indicate primitives from the MAC sublayer are prepared for the Network layer. Addresses associated with each layer are either added or removed as is appropriate for the direction of data flow and the layer involved.

1.2 BASIC PURPOSE

The service of the Logical Link Layer is the capabilities which it offers to a user in the next higher layer (or lower layer) . In order to provide its services, LLC layer builts its functions on the services which it requires from the next lower layer (MAC layer). The LLC layer's primitive services are:

REQUEST

The request primitive is passed from the LLC layer's user (it could be the higher layer: Level 6, or Network layer, or Transport layer, or could be the lower layer: MAC layer), to the LLC layer's sublayer to request that a service be initiated.

INDICATION

The indication primitive is passed from the LLC layer's sublayer to the LLC layer's user to indicate an event which is significant to the LLC layer's user. This event may be logically related to a remote servicerequest, or may be cause by an event internal to the LLC sublayer.

CONFIRM

The confirm primitive is passed from the LLC layer to the LLC layer's user to convey the results of one or more associated previous service request(s). This primitive may indicate either failure to comply or some level of compliance. It does not necessarily indicate any activity at the remote peer interface. IN THE CURRENT RELEASE, WE DO NOT USE THIS MECHANISM TO HANDLE THE LAYER INDICATION.

The Logical service access points called LSAPs are managed by the LLC sublayer and the LLC sublayer management services. Communications take place between local LSAPs and their peer enities, remote LSAPs, in the form of protocol data units (PDU). The Network layer or layers above LLC will select and use LSAPs to route messages to and from Network (or SubNetwork, or Transport) service access points called NSAPs (or TSAP's).

1.3 BASIC STRUCTURE

The LLC Sublayer and the LLC Sublayer Management functions are provided by a set of 3 processes that run under the Bridge Kernel operating system. The 3 processes are a layer management process (LM), a transmit process (XMT) and a receive process (RCV). Each process will have public access to tables that contain the state and status of the process itself as well as the various service access points (SAPs).

The mailboxes for a LLC Layer instance interface with the megabus software, the MAC layer and System Management. A mailbox will be created for each interface as follows.

	Mailbox Names (For Layer Instance 0-3)
Layer Management LM default mailbox	LNMLME LLC_LM0 LLC_LM1 LLC_LM2 LLC_LM3 Default mailbox for the LM process. All System Management messages arrive here and alarm messages returned from the kernel are routed here.
Default mailbox for Transmit process XMIT	llc_tx0 llc_tx1 llc_tx2 llc_tx3 Layer Management messages from LLC LM to transmit process LLC XMIT arrive here.
Default mailbox for Receive process RCV	llc_rx0 llc_rx1 llc_rx2 llc_rx3 Layer Management messages from LLC LM to receive process LLC RCV arrive here.
Megabus mbx	IODISP MEMDMA IOLD messages and returning DMA requests arrive here.
MAC_Data mbx	MAC_DATA.indication messages and returning MAC_DATA.request messages arrive here.

1.4 BASIC OPERATION

The Logical Link Control Layer in the IEEE 802 Local Area Network Protocol is common to the various medium access methods that are defined and supported by the IEEE 802 activity.

The Logical Link Control Layer interface service to the Network (Transport) Layer provides various services that the LLC layer, plus underlying layers and sublayer, offer to the Network (Transport) layer, as viewed from the Network (Transport) layer. These services are defined so as to be independent of the form of the medium access methodology, and of the nature of the medium itself.

In order to support the LACS, a software driver will be written for the DPS 6 to provide the interface to the different layers. In addition to this, System Management facilities will be provided in the DPS 6 and LACS to help control and administer the local LACS as well as the entire network.

The following diagrams illustrate the Logical Link Layer serves to different configuration of Local Area Network Protocals

LLC LAYER AT THE CURRENT RELEASE LAN CONFIGURATION

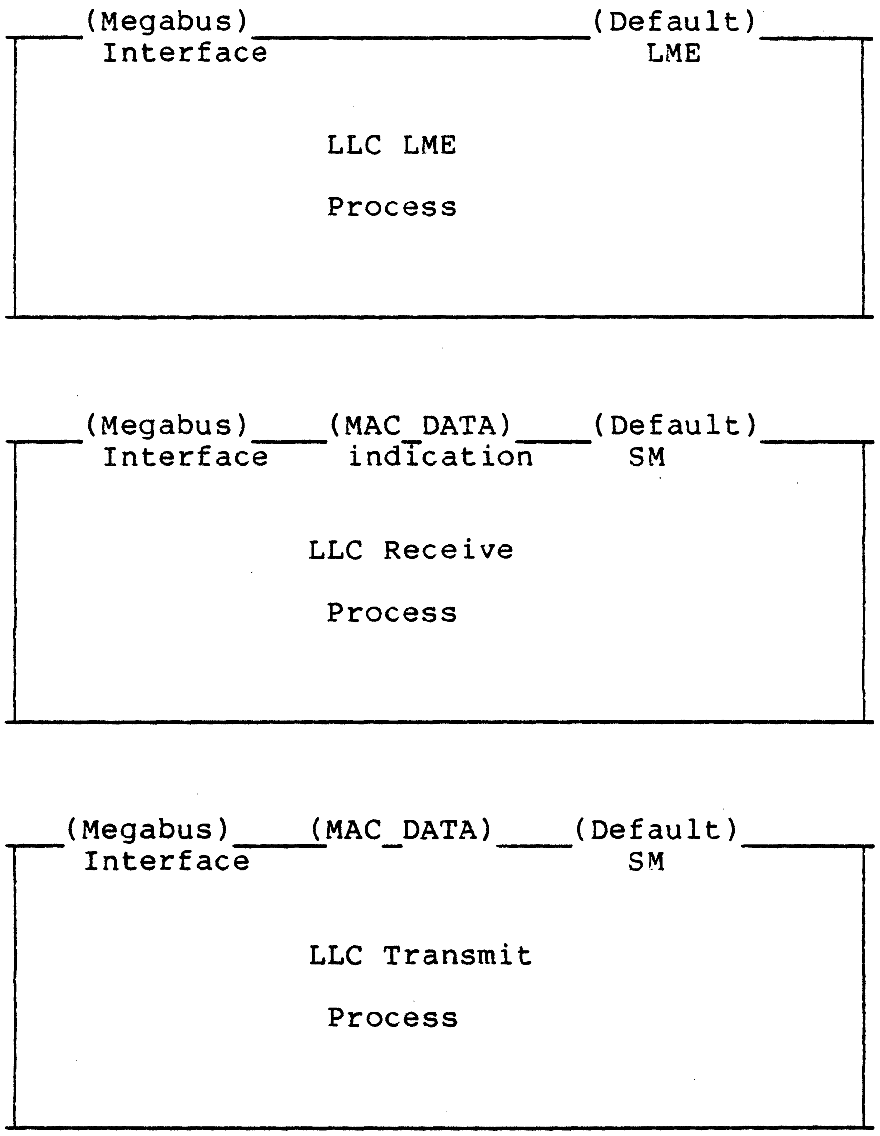
```
          +--------------------------------------------------------+
          |                                                        |
          |             DPS 6  USER                                |
          |             (MOD 400 4.1 Operating System)             |
          |                                                        |
          ++------------------------------------------------------++
           |                                                      |
           |            Distributed System Architecture           |
           |            (Honeywell DSA Network System)            |
           |                                                      |
           ++----------------------------------------------------++
            |           Interface Level 6 Software               |
            ++--------------------------------------------------++
             |          Megabus interface                       |
             ++------------------------------------------------++
+------<====>           LACS Megabus Interface Module (DMA)     |
|       |   |__________|IOLD message|___|Logical Control Blk|________
|       |                                                           |
|  SME  | |               LOGICAL LINK CONTROL LAYER                |
|       +=>                                                         |
|       | |  MAC DATA INDICATE MESSAGE    MAC DATA REQUEST MESSAGE  |
|       | |                                                         |
|System<=+__Instance-0__Instance-1__Instance-2__Instance-3__________|
|Manage|    | V  ^  |   | V  ^  |   | V  ^  |   | V  ^  |
|Module|    +-------+   +-------+   +-------+   +-------+
|     ===>  |Medium |   |Medium |   |Medium |   |Medium |
|     <===  |Access |   |Access |   |Access |   |Access |
|       |   |Control|   |Control|   |Control|   |Control|
+-------+   |   0   |   |   1   |   |   2   |   |   3   |
            +--| |--|   |--| |--|   |--| |--|   |--| |--|
           |Physical|  |Physical|  |Physical|  |Physical|
           |Medium 0|  |Medium 1|  |Medium 2|  |Medium 3|
```

LLC LAYER AT THE NEXT RELEASE LAN CONFIGURATION
  (With Network, Transport layers)

```
        +---------------------------------------------------------+
        |                                                         |
        |             DPS 6  USER                                 |
        |             (MOD 400 4.1 Operating System)              |
        |                                                         |
        ++-------------------------------------------------------++
         |                (Option)                              |
         |            Distributed System Architecture           |
         |            (Honeywell DSA Network System)            |
         |                                                      |
         |            (Can be other User application)           |
         |                                                      |
         ++----------------------------------------------------++
          |           Interface Level 6 Software               |
          ++--------------------------------------------------++
           |          Megabus interface                       |
           ++------------------------------------------------++
+------<====>         LACS Megabus Interface Module (DMA)     |
|      |    |--------|IOLD message|---|Logical Control Blk|--|
|      |    +--------+            +---+                   +--+
|      <====>         Transport Layer                         |
|      |    +----------------| TRN Message |----------------+
|      <====>         Network Layer                           |
|      |    +----------------| Net Message |----------------+
|      | ________________________________________________________
|  SME | |                LOGICAL LINK CONTROL LAYER             |
|      +=>                                                      |
|      | |  MAC DATA INDICATE MESSAGE   MAC DATA REQUEST MESSAGE |
|      | |                                                      |
|System<=+__Instance-0__Instance-1__Instance-2__Instance-3______|
|Manage|  ___| V  ^ |___| V  ^ |___| V  ^ |___| V  ^ |___
|Module|      +--------+  +--------+  +--------+  +--------+
|      ===>   | Medium |  | Medium |  | Medium |  | Medium |
|      <===   | Access |  | Access |  | Access |  | Access |
|      |      | Control|  | Control|  | Control|  | Control|
+------+      |   0    |  |   1    |  |   2    |  |   3    |
              +--| |---|  |--| |---|  |--| |---|  |--| |---|
                 | |        | |         | |         | |
            ___________  ___________  ___________  ___________
            | Physical|  | Physical|  | Physical|  | Physical|
            | Medium 0|  | Medium 1|  | Medium 2|  | Medium 3|
            -----------  -----------  -----------  -----------
```

Code for the LLC and LLC layer mamagement functions will be loaded with the rest of the LACS board code and begin operating when the process has been activated by System Management. Data exchanged with other processes such as MAC or megabus interface software, will take the form of messages delivered to process mailboxes via services provided by the Kernel.

LLC Layer Instance Block Diagram



The Process for transmit is the following:

1. recieve "xmit" LCB from level 6.

2. Validate LCB.

3. detect if data in LCB directly. If it is, go to step 5.

4. If data is not in LCB, call DMA bring data buffer cross the MEGABUS interface from level 6.

5. allocate llc_trans block.

6. convert LCB data to transaction block.

7. prepend the LLC header.

8. send message LLC_PDU (llc_trans) th MAC layer for transmit.

9. Set up all statiatic counters.

The Process for recieve is of the following:

There are three cases, recieve LCB arrival, PDU arrival, and XID, TEST response.

I. recieve LCB

1. recieve "READ" LCB from level 6.

2. validate LCB.

3. If PDU queue has waiting PDU block, then convert each of the PDU block (in the queue structure order) to LCB data format, andsend each one (in order) back to level 6. Untill PDU in queue are sent out completely. Complete the READ LCB.

4. If PDU queue is empty, allocate READ LCB transaction block to transfer the LCB data to READ transaction block. Hang up the transaction block at the READ LCB queue. (If the queue is already full, drop this transaction block return the memory allocated).

5. set statistic counter.

II. recieve PDU

1. recieve PDU data block from MAC layer.

2. validate PDU. If illigal PDU, drop it.

3. If PDU is valid, strip the LLC header.

4. convert PDU block to llc_trans block. If there are outstanding read LCB in the LCB queue, send PDU llc_trans block to level 6. If read LCB queue is empty, hang the PDU transaction block in the PDU queue. If the queue is full, drop the PDU, and clean the PDU memory.

5. update statistic counters.

III. recieve XID/TEST:

1. upon receipt of an XID command PDU from MAC prepare and send response XID PDU to requesting station.

2. upon receipt of a TEST command PDU from MAC prepare and send response TEST PDU to requesting station.

2. LOGICAL LINK LAYER EXTERNAL SPECIFICATION

2.1 OWNED DATA STRUCTURES

Each process in the set that make up a LLC layer instance will have access to layer data structures via an Instance Common Area. Pointers to local and remote LSAP tables, the IOLD function directory, and mailbox IDs for the various interfaces will all be contained in the Instance Common Area.

Layer Instance Common Area
Data structure type LICA:

Data Structure discription	LLC Layer entry name
Function directory mailbox ID array	fc_mbx_dir[16]
Recieve default mailbox	rx_default
Receive mailbox	rx_mbx
Receive MAC data indication	rx_mac
Megabus DMA mailbox ID "MEMDMA"	dma_mbx
Megabus IO Dipatcher ID "IODISP"	iodisp_mbx
Controller layer MGR ID "LNMLME"	ct_lm_mbx
Transmit default mailbox	tx_default
Transmit mailbox	xmit_mbx
LLC layer LME mailbox	lme_mbx
System MGR event mailbox	sm_event
MAC data request mailbox	mac_data
MAC activate mailbox	mac_act_mbx
Number of local sap dir entries	l_lsap_dir_sz
LSAP table pointer array	l_lsap_dir
Number of remote sap dir entries	r_lsap_dir_sz
Index to next free entry in remote	nxt_r_dir_index
Remote LSAP table pointer array	r_lsap_dir
Emergency event message pointer	emer_event
PDUs to undefined remote LSAPs	ctr_und_pdu
Layer instance major state	li_majorstate
Layer instance substate	li_substate
Max PDU size for the layer instance	max_pdu_sz
Layer instance number	inst_numbr
Layer number	layer
Layer instance init step	init_step

Unknown error counter	ctr_unk_err
IOLDs counter for layer manager	ctr_IOLDs_l
LCB to level 6 counter for LM	ctr_LCB_l6_l
LCB to lacs counter for receive	ctr_LCB_lac_l
IOLDs counter for receive	ctr_IOLDs_r
LCB to level 6 counter for receive	ctr_LCB_l6_r
LCB to lacs counter for receive	ctr_LCB_lac_r
IOLDs counter for xmit	ctr_IOLDs_x
LCB to level 6 counter for xmit	ctr_LCB_l6_x
LCB to lacs counter for xmit	ctr_LCB_lac_x
Buffer to level 6 counter	ctr_BUF_l6
Buffer to lacs counter	ctr_BUF_lac
Number of xmit PDU dropped	ctr_PDUd_tx
Number of receive PDU dropped	ctr_PDUd_rx

2.1.1 LLC Local SAP Components

LSAPs are represented in the LLC layer by SAP componemt tables. The table is created when directed by system management and exsist until deleted by system management messages. SAP Component tables will contain the following entries.

Data Structure discription	LLC Layer entry name
LSAP symbolic name	lsap_name[16]
class of LSAP	class
LSAP type	type[4]
LSAP venue	venue
Major administrative state	majorstate
LSAP sub administrative state	substate
MAC instance number	mac_inst
Length of MAC address (bytes)	mac_adr_lngth
LSAP address (8 bits)	lsap_adr
MAC address for this lsap (48 bits)	mac_adr[6]
IEEE 802.2 states	I802_st
IEEE 802.2 type	type_vct
Logical local LSAP address	log_adr
SAP indicator and flag word	sap_ind_flag
Pointer to event LCB	event_trans
Event indicator and flag word	event_ind_flag
Network layer mailbox for this sap	net_mbx
Current number of active remote sap	cur_num_rmts
pointer to activate remote sap dir	r_act_dir
Max transmit bytes	max_xmit_bytes
Max receive bytes	max_rcv_bytes
Max rtransmit credit	max_tx_credit

Receive LCB queue counter	rcvq_lcb_count
Receive PDU queue counter	rcvq_pdu_count
Transmit LCB queue counter	xmitq_lcb_count
Transmit PDU queue counter	xmitq_pdu_count
Actual transmit credit	act_xmit_credit
Actual receive credit	act_rcv_credit
Max PDU size	max_pdsz
First LCBI transaction in queue	lcb_fwd
LCBI root back pointer	lcb_bwd
Dummy LCB last forward in queue	lcb_end_f
Dummy LCB last backward in queue	lcb_end_b
First PDU message in queue	pdu_fwd
PDU root back pointer	pdu_bwd
Dummy PDU last forward in queue	pdu_end_f
Dummy PDU last backward in queue	pdu_end_b
First EVENT transaction in queue	evn_f
EVENT root back pointer	evn_b
Dummy EVENT last forward in queue	evn_end_f
Dummy EVENT last backward in queue	evn_end_b
Array if IEEE 802 group address	grp_addr[64]
Number of data octets transmitted	ctr_do_tx
Number of data octets received	ctr_do_rx
Number of frames with unknown command	ctr_uk_fr
Number of LCB transmitted	ctr_LCB_tx
Number of LCB received	ctr_LCB_rx
Number of receive connection IOLDs	ctr_IOLDs_rc
Number of event IOLDs	ctr_IOLDs_ev
Number of active IOLDs	ctr_IOLDs_ac
Number of UI frames transmitted	ctr_UI_tx
Number of UI frames received	ctr_UI_rx
Number of XID commands transmitted	ctr_XIDc_tx
Number of XID commands received	ctr_XIDc_rx
Number of XID responses transmitted	ctr_XIDr_tx
Number of XID responses received	ctr_XIDr_rx
Number of TEST commands transmitted	ctr_TESTc_tx
Number of TEST commands received	ctr_TESTc_rx
Number of TEST responses transmitted	ctr_TESTr_tx
Number of TEST responses received	ctr_TESTr_rx
Number of transmitt PDU dropped	ctr_PDUd_tx
Number of receive PDU dropped	ctr_PDUd_rx

IEEE802 Private states(DSA)
- IN_USE
- ENABLED
- DISABLED
- LOCKED
- TEST
- SHUTDOWN

2.1.2 IEEE 802 Connection Components

Connection oriented service is not provided in this implmentation.

2.1.3 LLC Layer LCBI Formats

The portion of the LCB actually transferred across the megabus is all that is used by LACS board code. Equivalent 'C' code structures are used to locate and modify portions of the LCB while the image is resident on the LACS board. The format for LACS resident LCB blocks, known as LCBIs, is given below in 'C' notation. Be aware that this structure represents only the portion of the L6 defined LCB that is actually transferred across the megabus.

Data Structure discription	LLC Layer entry name
Interrupt control word	cbicw
Function specific function code	cbfsf
Buffer indicators	cbind
Total Byte range	cbtrg
Number of buffer	cbbct
BUFFER DESCRIPTOR	cbdes[3]
Buffer address	buf_adr
Buffer indicator	buf_ind
Buffer range	buf_rng
Buffer residule range	buf_rsr
LCBI TYPES	lcbi_type
Generic LCBI block	generic_lcbi
Reserved spaces	cb_s[32]
Activate local LCBI	act_l_lcbi
Symbolic name	cbsym[16]
Proposed read max credit	cbprc
Proposed read SDU size	cbpms
Reserved spaces	cb_s[12]
Maximum SDU size	cbmss
Ideal SDU size	cbiss
Max pending read count	cbmpr
Write credit count	cbwcc
Maximum number connections	cbmcc
Logical address - local sap selector	cblsa
Deactivate local LCBI	da_l_lcbi
Logical address - local sap selector	cblsa
Reserved spaces	cb_s[30]
Activate remote LCBI	act_r_lcbi
Logical address - local sap selector	cblsa
Symbolic name	cbsym
Reserved spaces	cbrsv
Reserved spaces	cb_s[18]
Remote logical address	cblra

Read LCBI information	read_lcbi
Local lsap logical address	cblsa
Reserved spaces	cb_s[21]
Local actual buffer address	cblbs
SAP additional read credit	
Actual buffer size	cbabs
Remote sap indicator	cblri
Remote logical address	cblra
Write LCB information	trans_lcbi
Logical locap sap address	cblsa
Remote logical address	cblra
Class of service	cbcls
Reserved spaces	cb_s[26]
Write credit	cbawc
Event LCBI information	event_lcbi
Connect indicate event information	cn_ind
Function specification wd	cb_s23
Quality of service	cbcls
Expidited data option	cbexd
Return sap indicator	cbrsi
Logical remote address	cblra
Connection id	cbcid
Data arrival event information	data_arr
Reserved spaces	cbs[6]
Actual buffer size	cbabs
Additional write credit event	add_wc
Reserved spaces	cb_s[7].
Additional write credits	cbawc
SAP error event	sap_err
Reserved spaces	cb_s[7]
SAP error reason code	cbrcd
Controller status	cbcts
Indicator word	cbfss
Completion word	cbcbs

2.1.4 Layer Function Directory

Entries in the Function Directory used by the LLC Layer Instance are limited to receive, transmit, activate, and event.

2.1.5 Mesasge Formats

According to the message MSG data structure definition of the Bridge Communication Kernel, the message common data structure is as follow:

Data Structure Description	Kernel entry names
Next message on circular list	m_fwd
Last message on circular list	m_bwd
Processid of sending process	m_sender
Buffer description	m_bufdes
Urgent or normal	m_prio
User message type	m_type

2.1.6 MAC_DATA request message format

The following is a generic Medium Acess Control Layer data request mwssage format for different MAC layers.

Data Structure Description	MAC-LLC entry names
Normal message header	mh
Frame control channel	frame_ctl
Pointer to desc build area	pkt_desc
Return mailbox id	return_id
Return status field	status
Class of service parameters	cl_of_srvc
ether type field or 802.3 length	type_fld
LAN destination address	mac_DA

2.2 EXTERNAL INTERFACES

As mentioned above, all interfaces with other processes on the LACS board will be by mailbox messages. All of the messages seen by the LLC process have their formats defined by the respective process. Consult the MAC layer, the megabus interface software, an the system management services for the message formats.

In the execution of a request by one of these layers, several steps involving exchanges of messages with other processes will take place. Once a message has been "mailed" to another process the context and state of the "transaction" is lost. Therefore some mechanism must be in place to retrieve the context and state when the next step is to be executed. The method used in this code consist of a "transaction block" message.

The transaction block message is composed of a header area that is large enough to hold any of the messages that must be exchanged with other processes. After the header area, pointers to tables and values that are part of the transaction context become part of the transaction and are passed along with the message.

Transaction blocks are created at two times, once when an IOLD is received and once when the MAC layer sends a data indicate message.

Transaction blocks are used to send the following messages.

LCBIO	DMA request to move LCBs across the megabus
BUFIO	DMA requests to move buffers across the megabus
CONFMSG	MAC_DATA.confirm messages
L_DATA_msg	L_DATA.request and .indication messages

In addition, the transaction block carries pointers to the LCBI related to this transaction, the L6 memory address of the LCB, the length of the LCB, and the channel control word to be used when the LCB is returned to the L6.

The Logical Link layer transaction block data structure is listed at the following. At the top of the structure is a union of several message types used in the processing of a transaction to reduce the decoding and table lookup that would otherwise required.

Data Structure discription	LLC Layer entry name
Transaction server	serv
Transaction blk queue pointers	trn_blk
DMA requests for LCBIs	lcbio
DMA requests for buffers	bufio
MAC_DATA request messages	mdr
Pointer to a MAC data indicate msg	mac_ind
Data buffer descriptor pointer	data_bd
The length of LCBI	lcbi_leng
The channel involved	lcb_chan
Pointer to the buffer descriptor	l6_mem_ptr
Level 6 memory pointer to LCBI	l6_mem_ptr
Pointer to the buffer descriptor	bdi_blk
Level 6 memory pointer to buffer des.	l6_bdi_ptr
Pointer ro the local LSAP table	l_lsap_table
Pointer to the remote LSAP table	r_lsap_table
Local logical address	l_log_addr
Remote logical address	r_log_addr

2.3 INITIALIZATION REQUIREMENTS

Tables that describe the adapter that is being supported by this LLC layer process and a SAP component for LLC Layer Management will have been created by the initialization code of the process when it is created by the layer management.

2.4 TERMINATION REQUIREMENTS

There are no termination requirements for the LLC layer since it will be active in one form or another as long as the LACS board is active.

2.5 ENVIRONMENT

All LLC layer functions will be coded in the 'C' language, compile to 68000 assembly language by the cc68 compiler, and run under the Bridge real time kernel. Macros and header files from the Bridge communication development environment will be used where they can without alteration. The LLC layer will also use the header files from the megabus interface software, the MAC layer, and system management to define message formats and content. Refer to the respective header files for the details.

2.6 TIMING AND SIZE REQUIREMENTS

The code should be as efficient as prosible.

2.7 ASSEMBLY AND LINKING

All source code for the LLC layer will be under the version and release control of the SCCS (Source Code Control System) facilities provide by the UNIX development environment and the project source code administrator Assembler ad68 which is provided by the Bridge Communication development is used. Linking will be under the control of the master product Makefile used to combine all modules that constitute the product. Linker ld68 which is provided by the Bridge Communication development is used. Refer to UCOS UNIX MENU and the Bridge Communication Software Specification for the detailed description of these facilities.

2.8 TESTING CONSIDERATIONS

All product's sofeware functions are tested by the developer.

2.9 DOCUMENTATION CONSIDERATIONS

All LACS Logical Link Control Layer documentations will be written by follow the recommendations of The Honeywell software documentation guidelines. Also, all code design descriptions will be accompanied by a Procedural Design Language description.

2.10 OPERATING PROCEDURES

The LLC layer must recieve an IOLDs request before any other request take place. The emergency event transaction block should always availible for catastrophic error report event message in the case of kernel running out of memory.

2.11 ERROR MESSAGES

Logical Link Control Layer errors can be divided into the following classes:

* Catastrophic Errors
    * Unreportable Catastrophic Errors.
    * Refortable Catastrophic Errors.

* Fatal Operartion Errors
    * Fatal Operation Error that must be reported to the initiator or the operation.
    * Fatal Operation errors that must be reported to the initiator of the operation and as an event to the System Management.

* Non-Fatal Operation errors.

* Recoverable Errors.

* Protocol Errors.

* DPS6 System Error.

2.11.1 ERROR CONDITION DEFINITIONS

Unreportable catastrophic errors are errors where the integrity of the LACS controller is corrupted sufficiently to warrant halting the LACS controller. Futhermore the controller cannot be trusted to event report the error. These are typically hardware errors such as parity error while executing instructions, LACS internal bus errors.

Reportable catastrophic errors are errors where the integrity of the LACS system is corrupted sucfficiently to halt the LACS but it is possible to make an effort to report the error. These are typically caused by software errors where the LACS system is not set up correctly.

Fatal operational errors are errors which cause the process to abort the particular operation that it is performing and require that the layer server deactivate a SAP or disconnect a connection.

Non-fatal operation errors are errors which cause the process to abort the particular operation that it is performing.

Recoverable errors are errors which are temporary in nature typically caused by a temporary lack of controller resoiurce.

The protocol errors and DPS-6 system errors do not fall in the classes described above.

2.11.2 ERROR CONDITION HANDLING

For the unreportable catastrophic errors, the approach taken in handling such errors is to immediately halt all LACS processing in order to preserve the state of the LACS at the time of the errors occurence. The DPS6 detects a LACS fatal unreportable catastrophic error through the absence of any LACS response or as a result of exceeding the allowed limits for a IO instruction. There is a possibility that the LACS will not support a dump of it's memory due to the severity of a nonreportable catastrophic errors.

If the errors are reportable catastrophic errors, any process detecting these errors must record the location and type of the error in a common area associated with the process and report it to the layer management process it associated with. An error event indication is sent to LACS system management. The LACS system management process will attempt to report to the SM layer in the DPS 6. In the event that the system management process failed to report it to the DPS6, it will request the controller to halt the LACS. The controller process will also save an indication of the type of error in a known system area and set up the controller to all I/O orders.

2.11.3 ERROR MESSAGE CODES

All errors that detected by the LLC layer are handled at the llc err routine. The Codes listed below included all detected error conditions.

ERROR CODES(decimal)	ERROR CONDITIONS	LLC LAYER ERROR FLAGS
-1	No match on a search address or name	NO_MATCH
-2	For error but not exit, to calling	ERR_RETURN
20	Bad function spec. function code	BAD_FSFC
22	Invalid function code	INVALID_FC
24	Bad layer instance state	BAD_LAYER_STATE
26	Bad or unknown remote SAP	BAD_REMOTE_SAP
28	Bad or unknown local SAP	BAD_LOCAL_SAP
30	SAP already active	SAP_ACTIVE
32	SAP deactivated	SAP_DACTIVE
34	Multiple activate requests on SAP	MULTIPLE_ACT
36	Bad remote SAP state	BAD_R_LSAP_STATE
38	Bad local SAP state	BAD_L_LSAP_STATE
40	Create local LSAP error	CR_L_LSAP_ERR
42	Create remote LSAP error	CR_R_LSAP_ERR
44	SAP not available for services	SAP_NOT_AVAILABLE
46	Bad DSAP in receive PDU	BAD_RCV_DSAP
48	Bad SSAP in receive PDU	BAD_RCV_SSAP
50	Bad DMA transfer of data buffer	BAD_BUFF_TRAN
52	Bad DMA transfer of buffer descriptor	BAD_BDI_TRAN
54	Bad transmit request desc.	BAD_XMIT_REQ
56	Bad PDU size	BAD_PDU_SIZE
58	LCB buffer size too small for PDU	BUFF_TOO_SMALL
60	Bad DSAP state	BAD_DSAP_STATE
62	PDU exceeds max PDU size	PDU_TOO_BIG
64	Receive LCB, PDU queue full	QUEUE_FULL
66	Invalid transfer	INV_TRANSF
68	Return LCB error	RT_LCBI_ERR

70	Bad LCBI response - big err	BAD_LCBI_RP
72	Incorrect venue value	BAD_VENUE
74	Data too long for data in LCB	TOO_LONG_DLCB
75	LLC layer event code	LLC_EVENT
76	DMA of a buffer failed	DMA_BUF_FAIL
77	Send message fail	SENDMSG_ERR
78	Bad register mail box directory	BAD_REG_MBXDIR
79	Mail box delete fail	MBOX_DL_ERR
80	Bad receive PDU	BAD_RCV_REQ
81	Mail box turn off fail	MBOX_OFF_ERR
82	Invalid System Management request	BAD_SM_REQ
83	Mail box turn on fail	MBOX_ON_ERR
84	XID transaction error	XID_XMIT_ERR
85	register mail box fail	BAD_REG_MBOX
86	XID respound error	XID_RSPD_ERR
87	Resolve mailbox fail	BAD_RESOLVE_MBOX
88	Unknown message type	UNKN_MSG
89	Allocate no memory availible	NO_MEM_ALO
90	Bad io message	BAD_IO_MSG
91	Getbuf could not allocate	NO_MEM_GET
92	Invalid SM return mailbox	BAD_SM_MBOX
93	Genaric default error	DEFAULT_ERR
94	SM return mailbox full	SM_MBOX_FU
95	MAC return mailbox full	MAC_MBOX_FU
96	Invalid DMA mailbox	BAD_DMA_MBOX
97	Invalid MAC mail box	BAD_MAC_MBOX
98	DMA return mailbox full	DMA_MBOX_FU

3. LOGICAL LINK LAYER INTERNAL SPECIFICATION

3.1 OVERVIEW

Code that implements the LLC functionality will have been linked with the Kernel, the System Management, and the Interface Software before being loaded on the LACS board. All required LLC processes will be activated via the actions of the system management process. When an adapter is to be supported, system management will create a LLC layer instance process for that adapter. The initialization code in the LLC layer process will in turn create the receive and transmit process passing a pointer to table structures for the instance.

3.2 SUBCOMPONENT DESCRIPTION

Each process supporting LLC layer activity for an instance, is a collection of procedures that will perform a series of steps required to execute the various commands supported. Requests for LLC actions are in the form of Kernel mailbox messages. Each message is a request to perform the next step in the sequence required by the type of request.

Type 1 operations will conform to two operating states, active and inactive. In the inactive state, the LSAP has been defined but will not process any messages addressed to that address. To transition to the active state, the LSAP must have been defined via a create LSAP command from system management and then receive an activate LSAP command from the user via a LCB. The circumstances under which the states change are outlined in the following table.

Type 1 Operations States

Current State	Event	Action	Next State
INACTIVE STATE	SAP ACTIVATION REQUEST	REPORT STATUS ACTIVE	ACTIVE STATE
ACTIVE STATE	RECEIVE_UI	DATA_INDICATE	ACTIVE_STATE
	DATA_REQUEST	SEND_UI	ACTIVE_STATE
	XID_REQUEST	SEND_XID_C	ACTIVE_STATE
	RECEIVE_XID_C	SEND_XID_R	ACTIVE_STATE

RECEIVE_XID_R	XID_INDICATE	ACTIVE_STATE
TEST_REQUEST	SEND_TEST_C	ACTIVE_STATE
RECEIVE_TEST_C	SEND_TEST_R	ACTIVE_STATE
RECEIVE_TEST_R	TEST_INDICATE	ACTIVE_STATE
SAP_DEACTIVATE REQUEST	REPORT_STATUS SAP INACTIVE	INACTIVE STATE

The terms used in the above table are defined below.

INACTIVE_STATE	The LSAP is defined but will not respond to incoming messages. User requests will be returned with an inactive status indicated. with an inactive status indicated.
ACTIVE_STATE	The LSAP is ready to process incoming messages, or user requests.
SAP_ACTIVATION REQUEST	The activate request LCB from a user.
RECEIVE_UI	An unnumbered information frame was received by the MAC layer and resulted in a message in the LLC layer mailbox. The message type is MAC_DATA.indication.
DATA_REQUEST	An L_DATA.request message from a network layer or the megabus has been received.
XID_REQUEST	A send XID command was received from system management.
RECEIVE_XID_C	An XID command was received for this LSAP.
RECEIVE_XID_R	The XID command has returned as this message type.
TEST_REQUEST	A request to send a TEST message was received from system management.
RECEIVE_TEST_C	A TEST command was received for this LSAP.
RECEIVE_TEST_R	A TEST command response was received from the LSAP addressed in the TEST request.
SAP_DEACTIVATION REQUEST	A request has been received from SM or the user to deactivate the LSAP.

DATA_INDICATE	Generate a L_ DATA.indication message from the MAC DATA.indicate received.
SEND_UI	Generate a MAC_ DATA.request message from the L DATA.request adding the local and remote SAP addresses.
SEND_XID_C	Create a XID command from the system management request and send it.
SEND_XID_R	When a XID command arrives from a remote LSAP, return the message as a XID response.
XID_INDICATE	When the XID response to a XID command returns, the system management message is returned as a XID INDICATE.
SEND_TEST_C	Create a TEST command from this system management request and send it to the LSAP addressed.
SEND_TEST_R	When a TEST command is received at this LSAP, the TEST response message is returned.
TEST_INDICATE	The LSAP component has received the TEST response from a remote SAP. An indication of this event is returned to System Mgmnt.
REPORT_STATUS SAP_INACTIVE	This is the response returned to System Management when requested to deactivate an LSAP.

3.3 LOGICAL LINK CONTROL Layer Initialization

Each instance of the code executes as three independent processes under the control and scheduling of the kernel. System Management will use the kernel procreate service call to initiate the LLC layer management process and pass the pointer to a layer instance table. There are two parts to each process, an initialization phase and a normal execution phase. During the initialization phase, each process will test the step number in the instance table to determine which elements are to be initialized. Each step will save the pointer to the instance table passed by system management for future use. Step 0 will cause layer management related initialization steps to take place including the allocation of a function table for mailbox IDs. After the LME mailbox ID is entered in the proper place, the table is registered with IOLD software. The step count is incremented to 1, and the instance table pointer is passed in a receive process via a create call to the kernel services. This time receive initialization steps are performed. The instance table pointer is saved, receive mailboxes created and registered in the function table, and the step number incremented to 3.

The transmit process is created passing the pointer to the instance table as a parameter. Again the instance table is saved for the transmit process to use in future events and transmit related initialization takes place. Mailboxes ar created and their IDs entered into the function table in th transmit positions.

Each process suspends execution at the end of initialization and waits for messages. The kernel will give the process control when the criteria for priority and messages waiting for the process have been met.

3.4 Normal operations

Two parameters are present when the process is restarted, a pointer to the message and the mailbox ID used by the kernel to determine process state. This process will accept either of two messages from system management at the mailbox assigned when the process was created. The first to arrive will be a request to create a local LSAP. Table space will be allocated from PRIVATE memory large enough for the LSAP status. The pointer is then entered into the LSAP directory for this layer instance. The table will have entries for LSAP name, LSAP address, type, and the local MAC address to be used filled in. After setting the state of the LSAP to inactive, the message is returned to system Management with an indication of success or failure status.

The next type of messsage is the create remote LSAP. This message will cause the process to allocate a remote LSAP table from PRIVATE memory and add it to the layer's remote LSAP directory. The name, type, address, and remote MAC address will be filled in and the LSAP left in an inactive state. This message is also returned to system Management.

3.4.1 Activate LSAPs

When the activate LSAP message is received via LCBs, the named local LSAP is located in the layers LSAP directory. The MAC associated with this LSAP is determined and a registration message exchanged with the MAC layer. After saving the MAC layer mailbox ID, placing the LSAP in the active state, the number of IO credits and logical address (for local and remote lsap, must be in the LSAP directories range) is returned in the LCB. The LSAP is now ready to process messages.

A similar set of steps are performed for activate remote LSAP except for the registration with any MAC layer. The remote LSAP is now active and messages from this address will now be recognized.

Until a local and a remote LSAP are both defined and placed in the active state, messages with these addresses will be ignored.

3.4.2 IOLD Messages

This is the first step in any request. The message indicates that a LCB block of a given length at a L6 address is to be transfered across the megabus. A message with the length and L6 address is sent to DMA services. When it returns, the next step will be performed.

3.4.3 Connectionless Event Indication

A test is made for the existence of the LSAP specified in the request. If the LSAP is present, a test is made for the active state. If both tests indicate that a LSAP is present and active, the addressed LSAP tables are scanned for L_DATA.indicate messages. If one or more messages are waiting, the LCBI is completed and a request to transfer it back to the L6 is made to DMA services. When there are no messages waiting, the LCBI will be queued on the LSAP table to await the arrival of a message.

3.4.4 Read Connectionless Data

The same tests that are made for Connectionless Event Indicatic commands in 3.4.3, are made for Read Connectionless Data commands. If the LSAP is both present and active, the message waiting queue is checked. If there are messages waiting, the buffer is transferred across the megabus to the memory address found in the LCB for the length specified. The LCBI is marked completed and returned across the megabus also. When the message queue is empty, the LCBI is queued to await the arrival of a message.

3.4.5 Write Connectionless Data

When a LSAP is both defined and active, as in 3.7, Write will cause the buffer to be transferred across the megabus. The Write request is changed into a L_DATA.request transaction and sent to the LLC layer for transmission. When the confirm is returned, the LCBI is completed and returned across the megabus.

3.5 Local Area Control Subsystem Code -- Layer Management

Code that will create the layer SAP lists, and make or delete entries in those lists will reside in the LACS Code Layer Management. Inquiries returning the physical address or the logical route number as is appropriate for the layer will also be supported by this code. Layer management consist of procedures that process the messages listed below.

3.5.1 L_TEST.request

This request will cause the SAP component to transmit a TEST frame with the buffer passed as the data field. The command-response bit will be set to command.

THIS FUNCTION IS NOT IMPLEMENTED AT THIS VERSION YET> IT SHOULD BE IMPLEMENTED AT THE NEXT VERSION, AS PLANNED.

3.5.2 L_TEST.indication

TEST commands are processed at the SAP component since they do not require a logical link to be active. If the command-response bit indicates that this message is a response, a search will be made for the TEST.request command that will be waiting for completion. When found, the message will be returned with the buffer.

When the command-response bit is set to command, and the optional information field is present, it will be returned without modification as a TEST response. The SAP address fields are exchanged, the command-response bit set to response, and the message queued to the transmit process. No states or modes are affected.

3.5.3 XID.indication

This request will cause LLC to issue a XID command and wait for the response to return from the addressed LSAP. When it arrives, the XID information field will be returned in the indicate.

3.5.4 SM_DATA.request

THIS FUNCTION IS NOT IMPLEMENTED AT THIS VERSION YET> IT SHOULD BE IMPLEMENTED AT THE NEXT VERSION, AS PLANNED.

3.5.5 SM_DATA.indicate

THIS FUNCTION IS NOT IMPLEMENTED AT THIS VERSION YET> IT SHOULD BE IMPLEMENTED AT THE NEXT VERSION, AS PLANNED.

3.5.6 LM_SET_VALUE.request

This message is a request to set one of the values in the parameter set being maintained for a layer. The message will contain a return mailbox, a parameter identifier, a parameter value access control information, and an empty field for return status.

When the message is returned as a LM_SET_VALUE.confirm, the status will be set to the results of the action. Resulting status may be success or failure for one of the following reasons: success, fail_operation, fail_parameter, fail_value fail_access control. Access control will be allocated but nc used in this first release.

THIS FUNCTION IS NOT IMPLEMENTED AT THIS VERSION YET> IT SHOULD BE IMPLEMENTED AT THE NEXT VERSION, AS PLANNED.

3.5.7 LM_GET_VALUE.request

This message is the request to read out the value of a parameter. The message will contain a return mailbox, a parameter identifier, access_ control_ information, a field for the returning parameter_value, and a field for return status. The return message, a LM_READ_VALUE.confirm, will contain status and the value requested.

THIS FUNCTION IS NOT IMPLEMENTED AT THIS VERSION YET> IT SHOULD BE IMPLEMENTED AT THE NEXT VERSION, AS PLANNED.

3.5.8 LM_ACTION.request

This message will contain a return mailbox, an action_identifier, an action_value, access_control_information and a field for returning status. When the message returns as LM_ACTION.confirm, the status and action_value will have been set. The actions defined thus far include the following.

3.5.9 LM_EVENT.indicate

3.7 FUTURE DEVELOPMENT AND MAINTENANCE

Future development is expected to include the Transport and Network modules. Both will conform to OSI standards to insure compatability with other vendors in the domestic and international markets. The future of Honeywell product lines depends heavily on the adherence to these standards. These two layers are described in this text only to delineate the compatability issues.

NEXT RELEASE IMPLEMENTATION ISSUES

In the next release (Dec. 1986), some of the issues to implement in the Logical Link Control layer for the LACS communication software are listed as following:

Implement generic LCBI data structure in order to coordinate with Network and Transport layers.

Improve error checking algorithm, and error condition handling.

Improve source code readibility

Set up mail box message limitation to 16 or so. Detect the mailbox full situation, and status counter set up for message dropped.

Transport, Network layer interfaces.

Aging counter set up for PDU, LCB. Or using LIFO or FIFO mechanism to handle queue full condition.
(currently, FIFO mechanism for PDU, and LCB queue is implemented)

System management GET function support.

System management UPDATE function support.

Efficiency improvement at IOLDs, Transmit, Receive processes.

4. PROCEDURAL DESIGN

Design procedul is as the following:

1. IEEE 802 standard review and implementation algorithms discussion.
2. Data structure design.
3. PDL presudo code implementation.
4. C code implementation.
5. Compiling, assembling, and linking debugging.
6. LAN software testing debugging.
7. Release.
8. Support and efficience improving for the future version.

APPENDIX B

The following standard BNF definitions describe the various system mamagement messages that can be received by the LLC Layer and the parameters involved.

```
LAYER MANAGEMENT MESSAGE ::=         MESSAGE HEADER, MESSAGE TYPE,
                                     MESSAGE INFORMATION
MESSAGE HEADER ::= Kernel message header
MESSAGE TYPE ::= REQUEST | CONFIRM | EVENT
REQUEST ::= xx
CONFIRM ::= xx
EVENT ::= xx
MESSAGE INFORMATION ::=    EXCHANGE ID, LAYER INTERNAL SELECTOR,
                           ACCESS CONTROL, STATUS, OPERATION CODE,
                           OPERATION INFORMATION POINTER
EXCHANGE ID ::= A 16 bit integer
LAYER INTERNAL SELECTOR ::=          NAME, CLASS, TYPE, VENUE, STATE,
                                     SUBSTATE
NAME ::= 8 ASCII characters
CLASS ::= 05
TYPE ::= 8022
VENUE ::= LOCAL | IMAGE
LOCAL ::= 1
IMAGE ::= 2
STATE ::=        ANY | LOCKED | ENABLED | DISABLED | TEST |DOWN |
                 SHUTDOWN | INUSE
SUBSTATE ::= ANY | RESET | HALTED | LOADED | STARTED |   OPERATIONAL
ANY ::= 00
LOCKED ::= 03
ENABLED ::= 04
DISABLED ::= 05
TEST ::= 06
DOWN ::= 07
SHUTDOWN ::= 08
INUSE ::= 09
RESET ::= 01
HALTED ::= 02
LOADED ::= 03
STARTED ::= 04
OPERATIONAL ::= 05
ACCESS CONTROL ::= 00
STATUS ::= STATUS CODE, STATUS INFO
STATUS CODE ::= SOURCE, STATUS ID
SOURCE ::= 02
STATUS ID ::= 8 bit integer
STATUS INFO ::=  STATUS LENGTH, STATUS INFO DATA
STATUS LENGTH ::= 16 bit integer
STATUS INFO DATA ::= xx
```

```
OPERATION CODE ::= LM_GET_VALUE | LM_SET_VALUE | LM_ACTION
LM_GET_VALUE ::= 1
LM_SET_VALUE ::= 2
LM_ACTION ::= 4
LM_GET_VALUE STATUS ID ::= STD STATUS
LM_SET_VALUE STATUS ID ::= STD STATUS | BAD PARAMETER VALUE
LM_ACTION STATUS ID ::=    STD STATUS | BAD STATE |  BAD SUBSTATE |
                           ILLEGAL STATE CHANGE
STD STATUS ::=   SUCCESS | NOT SUPPORTED | BAD ACTION OPERATOR | BAD
                 LAYER INTERNAL SELECTOR
SUCCESS ::= 00
NOT SUPPORTED ::= xx
BAD ACTION OPERATOR ::= xx
BAD LAYER INTERNAL SELECTOR ::= xx
BAD PARAMETER VALUE ::= xx
BAD SUBSTATE ::= xx
ILLEGAL STATE CHANGE ::= xx
BAD LAYER INTERNAL SELECTOR ::= xx
OPERATION INFORMATION ::=  LM_GET_VALUE OP INFO |LM_SET_VALUE OP INFO|
                           LM_ACTION OP INFO
LM_GET_VALUE OP INFO ::=   PARAMETER ID, CONFIRM INFORMATION POINTER
LM_SET_VALUE OP INFO ::=   PARAMETER ID, PARAMETER VALUE
LM_ACTION OP INFO ::=      ACTION OPERATION, STATE | SUBSTATE
PARAMETER ID ::= xx
CONFIRM INFORMATION POINTER ::= values | list of values
PARAMETER VALUE POINTER ::= pointer to new value
ACTION OPERATION ::= UPDATE STATE | CREATE | DELETE | LIST | TEST ACT
UPDATE STATE ::= xx
CREATE ::= xx
DELETE ::= xx
LIST ::= xx
TEST ACT ::= xx
```

APPENDIX B

THE PROCEDURAL DESIGN LANGUAGE LISTING OF LOGICAL LINK CONTROL LAYER

Table of Contents

```
/*********************************************************************
**********************************************************************
***
***
***
***                              HONEYWELL
***
***                          Local Area Network
***
***              L O G I C A L   L I N K   C O N T R O L
***
***                             L A Y E R
***
***                       Layer Management module
***
***
***     Copyright: Honeywell Information System
***                All rights reserved.
***
***     Cereate Date:      1/25/86
***     By:                D. Oshaughn
***     Discription:
***
***     Revision #:    1
***     Date:          3/25/86
***     By:            M. Lu
***     Discription:
***
***     Revision #:
***     Date:
***     By:
***     Discription:
***
***
***
***     Functions & Parameters:
*
*   llc_lme (mbxptr,mbx_name)
*      MSG *mbxptr;
*      MBID mbx_name;
*   lme_lcb(mbxptr,lica_p)
*      MSG *mbxptr;             * Pointer to lcb message           *
*      LICA *lica_p;            * Pointer to llc layer inst. table *
*   sm_req(mbxptr,lica_p)
*      MSG *mbxptr;              * Pointer to llc layer inst. table *
*      LICA *lica_p;            * Pointer to llc layer inst. table *
*   mu_reg_ak(mbxptr,lica_p)
*      MSG *mbxptr;
*      LICA *lica_p;  * Pointer to the process variables *
*   reg_normal(mbxptr,lica_p)
*      MSG *mbxptr;
*      LICA *lica_p;  * Pointer to the process variables *
*   lm_set(mbxptr,lica_p)
*       MSG *mbxptr;
*       LICA *lica_p;  * Pointer to the process variables *
*    lm_cmpr_n_set(mbxptr,lica_p)
*       MSG *mbxptr;
*       LICA *lica_p;  * Pointer to the process variables *
*    lm_get(mbxptr,lica_p)
*       MSG *mbxptr;
*       LICA *lica_p;  * Pointer to the process variables *
*    lm_action(mbxptr,lica_p)
*       MSG *mbxptr;
*       LICA *lica_p;  * Pointer to the process variables *
*    ageing_alarm(mbxptr,lica_p)
*       MSG *mbxptr;
*       LICA *lica_p;  * Pointer to the process variables *
*    search_r_sap(remote_addr,r_dir_table,dir_index,lica_p)
*       char *remote_addr; * address of the remote LSAP to search       *
*       R_LSAP_DIR *r_dir_table; * Pointer to a remote LSAP directory table *
*       ushort *dir_index;         * Pointer to index into remote SAP directory *
*       LICA *lica_p;             * Pointer to the layer instance table       *
*    srch_name(lsap_name,lsap_dir,dir_sz,dir_index,lica_p)
*       char *lsap_name;      * Pointer to the LSAP name for search  *
*       R_LSAP_DIR *lsap_dir;     * Pointer to a LSAP directory table *
*       ushort dir_sz;              * Size of SAP directory *
*       ushort *dir_index;          * Pointer to index into SAP directory *
*       LICA *lica_p;             * Pointer to the layer instance table       *
*    next_sap(r_dir_table,dir_size,dir_index)
*       R_LSAP_DIR *r_dir_table; * Pointer to a remote LSAP directory table *
*       ushort dir_size;      * Size of directory being searched *
*       ushort *dir_index;  * Pointer to index into remote SAP directory *
*
***
**********************************************************************
**********************************************************************
*/

/* ********************************************************************
*
* ....................LLC Layer Management....................
*  This is the LLC Layer management that exist once for each
* LLC Layer instance created. The combination of this process,
* the recieve process and the transmit process constitutes one
* LLC Layer instance.
*
*         TITLE:            LLC LAYER MANAGEMENT INITIALIZATION FUNCTION
*         FUNCTION:         TBS
*         INPUT:
*         OUTPUT:
*
* The following conventional 'C' code header file is used
* by this process. See the respective files for definitions,
* structures and macros.
*
***********************************************************************
*/
```

```
/**********************************************************************
/*
*        TITLE:              LLC LAYER MANAGEMENT FUNCTION
*
*        FUNCTION:           Receives all message directed to LLC LM mbox.  It
*                            then determines the correct procedure to call
*                            to service the message.
*
*        INPUT:              Pointer to message
*                            Mailbox name
*
*        OUTPUT:
*
*
* .......................LLC Layer Management.........................
* This code is the operational layer management for each LLC layer
* instance. Mailbox messages received will be decoded by the switch
* statement into a procedure call for each case listed below. The
* pointer to the message and the pointer to the LICA table will be
* passed on to each procedure as parameters.
*********************************************************************
*/

   /* pointer to the LICA table */

    /* Retrieve the Layer Instance Common Area pointer */

    /* DO CASE on message type */

       /* CASE  An IOLD request message */

          /* increase counter */
          /* Call common procedure to handle the IOLD request */

      /* CASE The confirm to DMA a LCBI to the LACS has returned  */

          /* increase counter */
          /* Call the procedure which processes the LCBI */

      /* CASE The confirm to DMA a LCBI to the L6 has returned */

          /* increase counter */
          /* Call the procedure which will clean up after completion of LCBI */

      /* CASE  SM request message has been received */

          /* Call the procedure which processes system management requests */

      /* CASE  IO Disp. and DMA mailbox id response message returned */

          /* Call procedure to store mbx ids and
               initiate FC mbx dir. registr. */
```

```
        /* case serve to transport layer activate request */
          /* call transport interface activate procedure */

        /* CASE  FC mbx dir. registration response message has returned */

          /* Call procedure to cleanup after registration response returned */

       /* CASE  MAC Activate message returned form MAC layer mgr */

       /* case (MU_REG_RQ + CNFRM_MSG): */

          /* Call procedure to cleanup after activation response returned */

        /* OTHERWISE */
          /*******************************************************************/
          /* The message can not be decoded into any that are expected by this
          * process.
          * At this stage,
          * the space used by the message is returned to the kernel
          * until a method is designed to handle the problem.
          *
          * Also alarm messages should be added
          ****************************************************************
          */

          /*     mfree(moxptr); */
                                /* return to the calling */
                         /* if ERR_RETURN */

   /* ENDCASE End of the message type decoding */
        /* switch */

        /* llc_lme */

           /* Pointer to lcb message           */
        /* Pointer to llc layer inst. table */
             /* Error status */
         /* Pointer to the transaction block */

          /* A casting step */

          /* DO CASE on function specific function code in the LCBI */
            /* CASE An event order */
            /* CASE The activate LSAP order */
            /* deactivate case */
            /* OTHERWISE (invalid order for this channel) */
              /* Set completion word in the LCBI                  */

              /***************************************************************/
              /* Request DMA services to transfer LCB                        */
              /*            This is one of the difficult errors to handle    */
              /***************************************************************/

                /* if ERR_RETURN */
```

```
                                             /* return to the calling */
                                  /* if ERR_RETURN */

                      /* Report the presence of an invalid function code */
                 /* ENDCASE */
                       /* switch */
             /* llc_lme_lcb */

/****************************************************************************/
/*
*           TITLE:              SM REQUEST FOR LLC LM FUNCTION
*
*           FUNCTION:           Receives a SM request message and determines
*                               whether it is a valid request. The meesage is then
*                               forwarded to the function which can execute the
*                               request.
*
*           INPUT:              Pointer to mailbox message
*                               Pointer to layer instance table
*
*           OUTPUT:
*************************************************************************
*/

                  /* Pointer to llc layer inst. table */
             /* Pointer to llc layer inst. table */

 /* Pointer to the SM request */
     /* Error code                                */

                /* A casting step */

                /* DO CASE on operation code in the request */

                    /* CASE A GET request */

                    /* CASE A SET request */
                  case SET:

                    /* CASE A Compare and Set request */

                    /* CASE An ACTION request */

                    /* OTHERWISE (invalid request) */
                                  /* if ERR_RETURN */

               /* ENDCASE */
                       /* switch */

               /* Set message type to indicate a SM Confirm message       */

               /* Send the response message to SM     */
```

```
                                        /* return to the calling */
                            /* if ERR_RETURN */

                                        /* return to the calling */
                            /* if ERR_RETURN */
                                        /* return to the calling */
                            /* if ERR_RETURN */

        /* llc_sme_req */

/*****************************************************************/
/* The procedures in this section support the layer management functions
 * for the LLC layer.
 * ------------------------ LLC Layer Management ---------------------
 * The MAC sign-in message has returned
 ****************************************************************
 */

  /* Pointer to the process variables */

    /* This message is returning from the MAC sign-in process. */

 /* Pointer to the returning query message */

    /* Save the mailbox ID to send MAC_DATA.request messages */

    /* Release the message block to free memory again */
        /* mu_req_ak */

/*********************************************************************/
/* ------------------------ LLC Layer Management -------------------- */
/* The IOLD sign-in message has returned */
/*********************************************************************/

/* Pointer to the process variables */

   /*****************************************************************/
   /* This message has been returned from IOLD sign-in and is counted
   * as part of the transition to the 'ready state' for the instance.
   * Each message received increments the count by one.
   *****************************************************************
   */


    /* IF the registration was a success */

        /* Record the return of one of the REG_NORMAL messages */
    /* ENDIF */

    /* LLC LME Iniitialization complete                       */

    /* NEED TO FIND OUT IF EVENT SHOULD BE ISSUED HERE OR IS IT OPTIONAL */

    /* Release the message block to free memory */
```

```
        /* reg_normal */

/***********************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/* LM_SET.request */
/***********************************************************************/
/* Pointer to the process variables */

    /*************************************************************/
    /* DO CASE on parameter_identifier  */
    /* CASE each valid parameter  */
    /* IF the access_control_information is correct then */
    /*************************************************************/

       /* IF this parameter may be altered then  */
          /* IF the new parameter is in range for this item */

              /* set the parameter to the new value  */
              /* select success status  */
          /* ELSE  */
              /* select fail_value status */
          /* ENDIF */
       /* ELSE */
           /* select fail_operation status */
       /* ENDIF  */
     /* ELSE  */
       /* select fail_access_control status */
    /* ENDIF  */
    /* OTHERWISE  */
    /* select fail_parameter status */
    /* ENDCASE */
    /* return a LM_SET_VALUE.confirmation message with selected status  */
        /* lm_set */

/*****************************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/* LM_COMPARE_AND_SET.request  */
/*****************************************************************************/

/* Pointer to the process variables */

    /* DO CASE on parameter_identifier  */
    /* CASE each valid parameter  */
    /* IF the access_control_information is correct then  */
        /* IF this parameter may be altered then  */
           /* IF the new parameter is in range for this item  */
               /* IF the expected_parameter_value is correct then  */
                   /* set the new parameter value  */
                   /* select success status  */
               /* ELSE  */
                   /* select fail_expexted_value status  */
               /* ENDIF  */
             /* ELSE  */
                /* select fail_value status  */
```

```
            /* ENDIF  */
          /* ELSE  */
             /* select fail_operation status  */
             /* ENDIF  */
       /* ELSE  */
          /* select fail_access_control status  */
        /* ENDIF  */
        /* OTHERWISE  */
        /* select fail_parameter status  */
        /* ENDCASE  */
        /* lm_cmpr_n_set */

/*****************************************************************************/
/* ------------------------ LLC Layer Management --------------------- */
/* LM_GET.request message  */
/*****************************************************************************/

/* Pointer to the process variables */

     /* DO CASE on parameter_identifier  */
     /* CASE any valid parameter  */
     /* IF access_control_information is correct  */
        /* IF parameter is readable  */
            /* read the parameter value  */
            /* select success status  */
         /* ELSE  */
             /* select fail_operation status  */
          /* ENDIF  */
       /* ELSE  */
         /* select fail_access_control status */
       /* ENDIF  */
       /* OTHERWISE  */
       /* select fail_parameter status  */
       /* ENDCASE  */
       /* return LM_GET.confirmation with selected status  */
        /* lm_get */

/*****************************************************************************/
/* ------------------------ LLC Layer Management --------------------- */
/* LM_GET.request message  */
/*****************************************************************************/

  /* Pointer to the process variables */

     /* DO CASE on parameter_identifier  */
     /* CASE any valid parameter  */
     /* IF access_control_information is correct  */
        /* IF parameter is readable  */
            /* read the parameter value  */
            /* select success status  */
         /* ELSE  */
             /* select fail_operation status  */
          /* ENDIF  */
       /* ELSE  */
```

```
         /* select fail_access_control status */
       /* ENDIF  */
       /* OTHERWISE  */
       /* select fail_parameter status  */
       /* ENDCASE  */
       /* return LM_DEL.confirmation with selected status  */
        /* lm_del */

/***************************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/*
*        TITLE:          ACTION REQUEST PROCEDURE
*
*        FUNCTION:       Determines whether it is a valid action request
*                        and routes the request to the procedure required
*                        to perform the operation.
*
*        INPUT:          Pointer to message
*                        Pointer to LLC common data table
*
*        OUTPUT:
*/
/*.......................LLC Layer Management...........................*/
/* LM_ACTION.request message  */
/**************************************************************************/

/* Pointer to the process variables */

    /* SHOULD LLCERR code BE INITIALIZED ????          */

    /* Cast action request onto the message      */
    /* DO CASE on action_identifier  */

        /* CASE of a LIST ALL request             */
          /* IF request is to list all remote LSAP        */
            /* THEN                                        */
              /* Call list all remote LSAP procedure            */

           /* ELSE if request is to list all local LSAP       */
              /* Call list all local LSAP procedure            */
           /* OTHERWISE it must be a error                  */
         /* ENDIF                                         */

         /* CASE of a UPDATE STATE request              */

           /* Call update LSAP state routine               */
       break;

         /* CASE of a CREATE LSAP request              */
       case CREATE:
          /* IF request is for a remote LSAP creation       */
            /* THEN                                        */
              /* Call create remote LSAP procedure            */
            /* ELSEIF request is to create a local LSAP      */
```

```
               /* Call create local LSAP procedure                     */
             /* OTHERWISE it must be a error                          */
            /* ENDIF                                                 */

         /* CASE of a DELETE LSAP   request                 */

         /* DEFAULT an unsupported action identifier          */
           /* Cast action response on message                       */
          /* Set status to indicate an unsupported action identifier */

        /* ENDCASE  */
    }    /* switch */
}        /* lme_action */

/**************************************************************************/
/* ------------------------ LLC Layer Management --------------------- */
/* Message aging alarm message */
/**************************************************************************/

    /**********************************************************************/
    /* This message is the returning alarm request from setalarm. All l_pdu
    *messages attached to all LSAPs will be aged one time increment. Any message
    * found to be over the age limit will be released back to the buffer pool.
    * The assumption is that overaged messages are unexpected and a LCB will
    * never be issued.
    **********************************************************************
    */

    /* IF this instance is in the ready state then */
       /* FOR each IEEE802 defined LSAP that can exist */
          /* Point to the LSAP table */

          /* IF the LSAP has been defined */
             /* DO for each message on the waiting-for-LCB queue */
                /* Increment the aging counter */

                /* IF the aging counter is GTE the limit */
                   /* Remove the message from the queue */

                   /* Release the buffer space to the pool */
                   /* update the queue counter */

                   /* Release the memory to the memory pool */

                   /* Notify SM of the event */
                /* ENDIF */
                /* Roll thru the queue to the next entry */
             /* ENDDO */
          /* ENDIF */
        /* ENDFOR */
    /* ENDIF */

    /* Request the next aging alarm period */
```

```
/****************************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/*
*       TITLE:              SEARCH REMOTE LSAP ADDRESS PROCEDURE
*
*       FUNCTION:           This procedure searches the remote LSAP directory
*                           from a supplied address for a match to a given
*                           address.  It returns pointer to a value containing
*                           an index into the remote LSAP directory table if it
*                           is already defined, otherwise it will return
*                           a negative error value .
*
*       INPUT:              Remote LSAP Address
*                           Pointer to remote LSAP table directory
*                           Pointer to directory index value
*
*       OUTPUT:
*                           Error status
*
**************************************************************************
*/

    /*
     *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
     *******  a GENERIC LAYER FUNCTION                 *******
    */

    /* Initialize flags, indexes, and counts                          */

    /* DO WHILE the remote directory index value is less than or equal to
       max number of entries in the directory  */

        /* IF the directory index points to a valid entry (non null entry)   */
          /* THEN                                    */

            /* Fetch the pointer to the address in the remote LSAP table */
            /* Set the status flag to indicate
                       a match initially (null value)       */

            /* Initialize count to zero                       */

            /* Compare the two addresses an octet(byte) at a time to determine
                if there is a match    */
            /* DO WHILE the number of address octets checked
                               is less or equal to 7 xxx Lu.
                               and the address octets match */
                /* If the given address octet is
                               equal to the address octet in the remote
                                               LSAP table             */
                  /* THEN                                */

                    /* Increment the pointer to the address
                               octet in the remote LSAP teable */
                    /* Increment the number of octets checked */
                  /* ELSE there is no address match     */
                    /* Set status flag to indicate a no match */

                    /* Increment the directory index value      */
                   /* ENDIF */
              /* ENDDO */

        /* ELSE directory entry is not valid (null) */

            /* Set status flag to indicate a no match */

            /* Increment the directory index value      */

          /* ENDIF */

        /* Set status flag to reflect match flag status          */

        /* ENDDO */
    /* RETURN */

/*********************************************************************/
/*
*       TITLE:          SEARCH LSAP NAME PROCEDURE
*
*       FUNCTION:       This procedure searches a local or remote LSAP directory
*                       for a match to a given name. If a match is found ,
*                       it returns pointer to a value containing
*                       an index into the LSAP directory table .
*                       Otherwise it will return a negative error value .
*
*       INPUT:          Pointer to LSAP Name
*                       Pointer to LSAP table directory
*                       Size of LSAP directory
*                       Pointer to directory index value
*                       Pointer to layer instance table
*
*       OUTPUT:
*                       Error status
*
**********************************************************************
*/

    /*
     *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
     *******  a GENERIC LAYER FUNCTION                 *******
     *******  ALSO make presentable                    *******
    */

    /* Initialize flags,indexes, and counts                              */

    /* DO WHILE the remote directory index value is less than or equal to
                                max number of entries in the directory  */
       /* IF the directory index points to a valid entry (non null entry)   */
```

```
          /* THEN */
            /* Fetch the pointer to the name in the  LSAP table             */

            /* Compare the two addresses an octet(byte) at a time to determine
                if there is a match                                          */

            /* Set the match flag to indicate a
                        match initially (null value)        */

            /* Initialize count to zero                    */

            /* DO WHILE the number of address octets checked is
                              less or equal to 16
                              and the address octets match             */
                /* If the given name octet is equal to
                        the name octet in the remote
                                                LSAP table             */
                  /* THEN */
                     /* Increment the pointer to the
                              name in the remote LSAP table */
                     /* Increment the number of octets checked             */
                 /* ELSE there is no address match */
                    /* Set status flag to indicate a no match */
                    /* Increment the directory index value */
                    /* ENDIF */
                /* ENDDO */
         /* ELSE directory entry i  not valid (null)     */
            /* Set match flag to indicate a no match */

            /* Increment the directory index value */
            /* ENDIF                                 */
        /* Set the status flag to reflect the compare operation */
       /* ENDDO                 */
    /* RETURN */
}       /* srch_name */

/***********************************************************************/
/*
*       TITLE:            SEARCH FOR NEXT AVAILABLE DIRECTORY ENTRY PROCEDURE
*
*       FUNCTION:         This procedure searches the remote LSAP directory
*                         from a supplied address for a null entry
*                         address.  It returns pointer to a value containing
*                         an index into the remote LSAP directory table if it
*                         is already defined, otherwise it will return
*                         a negative error value .
*
*       INPUT:            Pointer to LSAP table directory
*                         LSAP table directory size
*                         Pointer to directory index value
*
*       OUTPUT:
*                         Error status
*
```

```
**************************************************************************
*/

    /*
     *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
     *******  a GENERIC LYAER FUNCTION                 *******
    */

    /* Initialize flags, indexes, and counts                                */

    /* DO FOR the remote directory index value is less than or equal to
        max number of entries in the directory
                        AND the directory entry is null                     */

        /* Increment the directory index */
        /* ENDDO */
    /* IF the directory index points to an available entry (null entry) */
     /* THEN */
        /* Set the status flag to indicate a success (null value)          */
      /* ELSE there is no address match */
        /* Set status flag to indicate a no available entry */
        /* ENDIF */
    /* RETURN */
```

```
/*****************************************************************
*****************************************************************
***
***
***
***                               HONEYWELL
***
***                          Local Area Network
***
***              L O G I C A L   L I N K   C O N T R O L
***
***                              L A Y E R
***
***                     layer initialization module
***
***
***      Copyright: Honeywell Information System
***                 All rights reserved.
***
***      Cereate Date:      3/25/86
***      By:                M. Lu
***      Discription:
***
***      Functions & Parameters:
***
*
*   llc_lme_init (startup_param)
*      STARTUP_PARAM *startup_param;  * Pointer to startup parameter block  *
*   llc_req_io_dma(return_mbox,lica_p)
*      MBID return_mbox;
*      LICA *lica_p;
*   rcv_mbx_ids(mbxptr,lica_p)
*      MSG *mbxptr;
*      LICA *lica_p;  * Pointer to the process variables *
*   reg_fcmbx_dir(lica_p)
*      LICA *lica_p;              * Pointer to Layer Instance table  *
*   create_l_lsap(mbxptr,lica_p)
*      MSG *mbxptr;          * Pointer to create local LSAP message *
*      LICA *lica_p;  * Pointer to the intance table *
*   create_r_lsap(mbxptr,lica_p)
*      MSG *mbxptr;          * Pointer to create local LSAP message *
*      LICA *lica_p;  * Pointer to the instance taole *
*
***
***      Revision #:
***      Date:
***      By:
***      Discriotion:
***
***      Revision #:
***      Date:
***      By:
***      Discription:
```

```
***
***
**********************************************************************
**********************************************************************
*/

/* *******************************************************************
*
* ......................LLC Layer Management......................
*  This is the LLC Layer management that exist once for each
* LLC Layer instance created. The combination of this process,
* the recieve process and the transmit process constitutes one
* LLC Layer instance.
*
*        TITLE:           LLC LAYER MANAGEMENT INITIALIZATION FUNCTION
*        FUNCTION:        TBS
*        INPUT:
*        OUTPUT:
*
* The following conventional 'C' code header file is used
* by this process. See the respective files for definitions,
* structures and macros.
*
**********************************************************************
*/

/* *******************************************************************
*  This is the initial entry point for The process. The
* code is executed once as an initialization step. All other
* entries are at llc_lme.
**********************************************************************
*/

 /* Pointer to startup parameter block */

                       /* index variable for array initialization */
        /* clear block size */
             /* Priority of receive process        */
             /* Priority of transmit process       */
   /* Pointer to the layer instance common area */
  /* Pointer to this layer instance receive process name */
  /* Pointer to this layer instance xmit process name */
   /* Return code from kernel calls */
  /* Pointer to IOLD signin message */
    /* Pointer to the receive process PCB */
    /* Pointer to the transmit process PCB */

  /* How do you make this a variable    */

    /* Start of LLC layer management initialization     */

    /* Allocate space for the LLC Layer Instance Common Area table */
   /* memory alocation error handl */
                                   /* return to the calling */
```

```
                                /* if ERR_RETURN */

    /* record the lica at reserved table */

     /* Initialize all of layer instance table  to NULL                 */
    /* set layer instance table pointer   */
                 /* Clear it all    */

     /* Save the pointer to the LICA table for this process */

     /* Allocate space for the LLC local LSAP directory table */

         /* memory alocation error handl */
                                   /* return to the calling */
                           /* if ERR_RETURN */

                 /* record the l_lsap_dir pointer at reserved table */

     /* Initialize all local lsap entries to NULL              */
     /* Allocate space for the LLC remote LSAP directory table */

         /* memory alocation error handl */
                                   /* return to the calling */
                           /* if ERR_RETURN */

                 /* record the r_lsap_dir pointer at reserved table */

     /* Initialize all remote lsap entries to NULL             . */

     /* Allocate space for the LLC emergency message to SM */
         /* memory alocation error handl */
                                   /* return to the calling */
                           /* if ERR_RETURN */
     /* Initialize all LLC emergency message entries to NULL              */
                 /* Clear it all    */

     /* Create a second mailbox for LLC LME */

     /* Add LLC Layer Management mailbox ID to the function code directory */
     /* THIS IS A TEMPORARY PIECE FOR DEBUG                   */

     /* Resolve mailbox for Controller LME and Store in layer instance table */

                                   /* return to the calling */
                           /* if ERR_RETURN */

     /* Set layer to equal LLC (layer =2)                         */

     /* Save the instance number passed by SM */

     /* Fetch the priority of the receive process                   */

     /* Fetch the priority of the transmit process                   */
```

```
/* DO CASE on the layer instance number to determine process name    */

   /* CASE This is layer instance zero                         */
     /* Register the well known mailbox name           */

                                 /* return to the calling */
                         /* if ERR_RETURN */

     /* Set layer instance process name for instance 0      */
    /* set pointer to receive process name */
    /* set pointer to transmit process name */

   /* CASE This is layer instance one                         */
     /* Register the well known mailbox name           */
                                 /* return to the calling */
                         /* if ERR_RETURN */

     /* Set layer instance process name for instance 1      */
       /* set pointer to receive process name */
       /* set pointer to transmit process name */

   /* CASE This is layer instance two                         */
     /* Register the well known mailbox name           */

                                 /* return to the calling */
                         /* if ERR_RETURN */

     /* Set layer instance process name for instance 2      */
         /* set pointer to receive process name */
         /* set pointer to transmit process name */

   /* CASE This is layer instance three                         */
     /* Register the well known mailbox name           */

                                 /* return to the calling */
                         /* if ERR_RETURN */

     /* Set layer instance process name for instance 3      */
    /* set pointer to receive process name */
    /* set pointer to transmit process name */

   /* DEFAULT - error                                               */
                 /* if ERR_RETURN */

/* END CASE                                         */
     /* switch */

/* SHOULD THE RCV AND XMIT PROCESS MBOXXES BE STORED AWAY ???    */

/* Create a receive process for this LLC instance */

/* Create a transmit process for this LLC instance */

/* Request the DMA and IO mbox ids                                  */
```

```
                                  /* return to the calling */
                          /* if ERR_RETURN */

    /* Now it a matter of waiting for request mbx ids message to return  */

/************************************************************************/
/* Request Mailbox Ids for IO Dispatcher and DMA procedure    */
/*
*         TITLE:    REQUEST MAILBOX IDs for IO DISPATCHER and DMA PROCEDURE
*
*        FUNCTION: Issues a message to the Controller layer manager requesting
*                  the IO Dispatcher and DMA  mailbox ids.
*
*        Input:    Access to common system management data structures
*                  Return mailbox id
*
*        Output:   Error Status
*
***********************************************************************
*/

                  /* IO and DMA mbox id req. msg. */
                          /* clear block pointer */
                  /* block size */
          /* clear block counter */

   /* Allocate some space for an IOLD sign in message */
                                  /* return to the calling */
                          /* if ERR_RETURN */

                  /* clean up all mem mbid_ptr */

    /* Set the priority and clear the buffer descriptors */

    /* Set the message type to register */

    /* Set return mailbox ID */

    /* Clear the return code */

    /* Turn response mailbox on */
        /* return to the calling */
        /* if ERR_RETURN */

     /* Send message to CT LM to fetch IO Dis. and DMA      */

                                  /* return to the calling */
                          /* if ERR_RETURN */

    /* Return                                               */

        /* reg_io_dma */

/*************************************************************************/
```

```
/* ------------------------ LLC Layer Management --------------------- */
/* The IO Dispatcher and DMA mailbox id request has returned */
/*
*          TITLE:    RECEIVE MAILBOX IDs for IO DISPATCHER and DMA PROCEDURE
*
*         FUNCTION: Receives a message from the Controller layer manager
*                   returning the IO Dispatcher and DMA  mailbox ids. These
*                   mailbox ids are stored away in the layer instance table
*                   and the registration of the the function code mailbox
*                   directory with the Controller LM is initiated.
*
*         Input:    Access to common system management data structures
*
*         Output:   Error Status
*
*************************************************************************
*/

  /* Pointer to the process variables */
     /* Pointer to the returning message */
      /* Error status                           */

    /* The sign-in message has returned fron IOLD interface software */

    /* This message has been returned from IOLD sign-in */
                            /* return to the calling */
                    /* if ERR_RETURN */

                    /* return to the calling */
                    /* if ERR_RETURN */

    /* STORE away the iodisp and dma mailbox   */

    /* Release the message block to free memory */

    /* Register with IO Dispatcher the function code mailbox directory    */
                            /* return to the calling */
                    /* if ERR_RETURN */

    /* Now it a matter of waiting for IOLD registration to return  */

    /* Return                                        */

        /* rcv_mox_ids */

/***********************************************************************/
/*
*         TITLE:          REGISTER FUNCTION CODE MAILBOX DIRECTORY
*
*         FUNCTION:       Issues a registration message to the IO Dispatcher
*                         which contains a pointer to the mailbox directory
*                         of a LLC layer instance.  May be used by other layer
*                         protocols by modifying layer value (i.e XPORT - set
*                         layer = 4) and redirecting error processing.
```

```
*
*        INPUT:            Pointer to function code mailbox directory
*                          Return mailbox id
*                          Value of layer instance
*
*        OUTPUT:           Error code
*                                  0 - success
*
***********************************************************************
*/

 /* Register Function Code Mailbox Directory routine    */
           /* Pointer to Layer Instance table  */

                           /* Pointer to fc mbox directory registration message */
      /* Error code                                 */
                  /* clear block pointer */
        /* block size */
/* clear block counter */

  /* Allocate some space for an IO dispatcher sign in message */
                                  /* return to the calling */
                           /* if ERR_RETURN */

                  /* clean up all mem mbid_ptr */

   /* Set the priority and clear the buffer descriptors */

   /* Set the message type to register */

   /* Set return mailbox ID to this layer manager instance mailbox */

   /* Set pointer to function code mailbox directory */

   /* set channel for layer (shifted to left 3 bits)  and layer instance   */

   /* Clear the return code */

   /* Send message to IO Dispatcher */
                                  /* return to the calling */
                           /* if ERR_RETURN */

   /* Release the message block to free memory */

   /* Return with error code                             */
        /* reg_fcmbox_dir */

/*****************************************************************/
/*
* .........................LLC LAYER....................................
*  This is a collection of procedures that implement and support the three
* processes that are a LLC layer instance. They are constructed to take
* advantage of the 'C' code parameter passing by value and automatic
* variables to implement the reentrant characteristics that will be
* required to support use by several processes under the kernel. Some
* procedures, such as the megabus support routines, will be used by all three
* processes in each of the four posible instances. The value parameters
* and automatic variables will allow simultaneous use by the three processes
* within any one instance and the layer instance common area tables will
* provide the context for each individual instance.
*******************************************************************
*/

/*********************************************************************/
/*
*       TITLE:          CREATE LOCAL LSAP PROCEDURE
*
*       FUNCTION:       Receives a creat local LSAP request message ,
*                       creates a LSAP table and intitializes it the values
*                       contained in the message.  All statistical counters
*                       are reset to zero.  No values are defaulted.
*
*       INPUT:          Pointer to mailbox message
*                       Pointer to layer instance table
*
*       OUTPUT:
*
*************************************************************************
*/

/**************************************************************************/
/* ------------------------ LLC Layer Management --------------------- */
/*  This procedure creates LSAP tables and sets initial values. The procedure
* must be called with Create message pointer
* and a pointer to the instance table that
* the LSAP is associated with.
**************************************************************************
*/

        /* Pointer to create local LSAP message */
  /* Pointer to the intance table */
                           /* pointer to create local lsap message */
    /* Pointer to a LSAP table memory block */
     /* Number of the LSAP  */
     /* Index integer for arrays */
     /* LLC error status   */
   /* block size */
        /* clear blocks pointer */

    /* Cast on create lsap message              */

    /* Fetch lsap number from LLC address            */
   /* fix for all dir start at 1 */

    /* IF lsap already existed                  */

    /* THEN there is an error                   */
      /* return notice to the calling module, bad_create request */
    /* ENDIF                                                   */

    /* Allocate some space for the LSAP table */

    /* IF space was available then */

       /* Set all initial values in the LSAP table */
       /* Clear the name                               */

       /* Allocate space for the activated remote LSAP directory and clear it */

       /* Initialize all remote lsap entries to NULL */

       /*   !!! WHAT TO DO ABOUT GROUP ADDRESSES           */
            /*   !!! WHAT TO DO ABOUT GROUP ADDRESSES           */
       /* Clear the group addresses                             */
        /* Clear group addresses */

       /* copy LSAP name from message */

       /* copy type 8022 from message */

       /* copy LSAP address from message NEED TO REVISIT */

       /* copy LSAP address from message */
       /* NEED TO REVISIT - WHAT to do about
                                   address length        */
       /*lsap_table->lsap_adr++;        * fix for start from 1 */

       /* set class to LL */
       /* set venue to  local  */
       /* set state */
       /* set substate */

       /* set MAC mapping  */
       /* set to maximum number of transmit bytes  */
       /* set to maximum number of receive bytes    */
       /* set to type 1   */
       /* set to maximum PDU size in bytes    */
       /* set maximum receive credit                */
       /* set maximum transmit credit                */

       /* Initialize all working queues/buffers and lists        */
                                   /* NEED TO REVISIT       */

         /* clear indicator flag   */

       /* End of the list */
        /*Top of the list */
       /* End of the list */

       /* Reset all statistical counters                        */
                                   /* NEED TO REVISIT       */
```

```
          /* recieve lcb queue counter */
          /* recieve pdu queue counter */
          /* recieve lcb queue counter */
          /* recieve pdu queue counter */
        /* Data octets transmitted */
        /* Data octets received  */
        /* Unknown commands */
          /* number of LCB  transmitted */
          /* number of LCB received */
                  /* receive connection IOLDs */
                  /* number of event IOLDs */
                  /* number of active IOLDs */
        /* UI frames transmitted */
        /* UI frames received */
                  /* XID commands transmitted */
                  /* XID commands received */
                  /* XID responses transmitted */
                  /* XID responses received */
          /* TEST commands transmitted */
          /* TEST commands received */
          /* TEST responses transmitted */
          /* TEST responses received */
                  /* number of xmit pdu droped */
                  /* number of rcv pdu droped */

         /* Save the address of the LSAP table in the adapter table */

         /* Set status                        */
         /* set source to indicate its an LLC status  */
         /* zero out statusid to indicate success */
         /* set to 2 bytes */
         /* set data to zero */

      /* ELSE   a bad request      */

        /* Set status                        */
        /* set source to indicate its an LLC status  */
        /* zero out statusid to indicate failure */
        /* set to 2 bytes */
        /* set data to zero */

        /* ENDIF */

      /* Set message type to indicate a SM Confirm message      */
          /* create_l_lsap */

/***********************************************************************/
/*
 *       TITLE:            CREATE REMOTE LSAP PROCEDURE
 *
 *       FUNCTION:         Receives a create remote LSAP request message ,
 *                         creates a remote LSAP table and intitializes it the
 *                         values contained in the message.  All statistical
 *                         counters are reset to zero.  No values are defaulted.
```

```
*
*        INPUT:            Pointer to mailbox message
*                          Pointer to layer instance table
*
*        OUTPUT:
***********************************************************************
*/

/**************************************************************************
/* ------------------------ LLC Layer Management --------------------- */
/*  This procedure creates LSAP tables and sets initial values. The procedure
* must be called with Create message pointer
* and a pointer to the instance table that
* the LSAP is associated with.
*************************************************************************
*/

          /* Pointer to create local LSAP message */
/* Pointer to the instance table */
                                  /* pointer to create local lsap message */
      /* Pointer to a LSAP table memory block */
        /* Number of the LSAP  */
       /* Index integer for arrays */
      /* Index into remote SAP directory */
          /* clear block pointer */
     /* block size */
       /* LLC error status   */

      /* Cast on create remote lsap message                     */

      /* Call routine to search remote SAP directory for remote address   */
          /*return(CR_R_LSAP_ERR);*/
          /* return bad status */

      /* Call routine to search remote SAP directory for next available entry   */
                                  /* return to the calling */
                  /* if ERR_RETURN */

      /* Allocate some space for the remote LSAP table */

      /* IF no memory available for the remote LSAP table */
      /* THEN there is an error                   */
                                  /* return to the calling */
                          /* if ERR_RETURN */

        /* ENDIF                                     */

        /* Set pointer to remote LSAP into next directory entry */
                    /*increment index */

      /* Initialize the remote LSAP table                 */

      /*   !!! WHAT TO DO ABOUT REMOTE GROUP ADDRESSES ???        */
                                /* Clear group addresses */
     /* copy LSAP name from message */

     /* copy type 8022 from message */

     /* Copy Lsap address length into remote lsap table    */

     /* copy LSAP address from message  NEED TO REVISIT address length   */

     /* set class to LL */
     /* set venue to  remote  */
     /* set state */
     /* set substate */

     /* set to type 1  */

  /* initialize log. addr. */
   /* clear indicator flag   */
    /* clear current number of act.   */

     /* Reset all statistical counters                            */
                                      /* NEED TO REVISIT      */
    /* Data octets transmitted */
    /* Data octets received  */
    /* Unknown commands */
    /* UI frames transmitted */
    /* UI frames received */
      /* XID commands transmitted */
      /* XID commands received */
      /* XID responses transmitted */
      /* XID responses received */
    /* TEST commands transmitted */
    /* TEST commands received */
    /* TEST responses transmitted */
    /* TEST responses received */

     /* Set status                  */
     /* set source to indicate its an LLC status  */
     /* zero out statusid to indicate success */
     /* set to 2 bytes */
     /* set data to zero */

         /* create remote lsap */
```

```
/*****************************************************************
******************************************************************
***
***
***                         HONEYWELL
***
***
***                     Local Area Network
***
***          L O G I C A L   L I N K   C O N T R O L
***
***                        L A Y E R
***
***                     receive module
***
***
***     Copyright: Honeywell Information System
***                All rights reserved.
***
***     Cereate Date:     1/25/85
***     By:               D. Oshaughn
***     Discription:
***
***     Revision #:    1
***     Date:          3/25/86
***     By:            M. Lu
***     Discription:
***
***     Revision #:
***     Date:
***     By:
***     Discription:
***
***
***     Functions & Parameters:
*
*               llc_rx_inir (lica_p)
*                 LICA *lica_p;  * Pointer to the layer instance common area *
*               llc_rx (mbxptr,mbx_name)
*                 MSG *mbxptr;
*                 MEID mbx_name;
*               rcv_lcbi_to_lac(mbxptr,lica_p)
*                 MSG *mbxptr;
*                 LICA *lica_p;  * Pointer to the process variables *
*               rcv_buf_to_l6(mbxptr,lica_p)
*                 MSG *mbxptr;
*                 LICA *lica_p;
*               mac_ind(mbxptr,lica_p)
*                 MSG *mbxptr;
*                 LICA *lica_p;  * Pointer to the process variables *
*               validate_pdu(indmsg,llc_frames,r_log_addr,lica_p)
*                 DATA_IND_MSG *indmsg; * Pointer to the data indicate message *
*                 LLC_FRMS *llc_frames;  * Pointer to incoming frames *
*                 u_long *r_log_addr;  * Pointer to remote LSAP logical address *
*                   LICA *lica_p;  * Pointer to the process variables *
*                 send_buf_to_l6(llc_trans,lica_p)
*                   LLC_TRANS *llc_trans; * Pointer to the LLC transaction block *
*                   LICA *lica_p;  * Pointer to the process variables *
*                 aging_alarm(mbxptr,lica_p)
*                   MSG *mbxptr;
*                   LICA *lica_p;  * Pointer to the process variables *
***
***
********************************************************************
********************************************************************
*/

/*****************************************************************/
/*
* ....................LLC Receive Initialization............................
*  This code initializes a receive process for a LLC Layer instance.
*
*       TITLE:              LLC RECEIVE INITIALIZATION PROCEDURE
*
*       FUNCTION:           Initialize the Receive process by storing
*                           the pointer to the layer instance common table
*                           in the receive process PCB.  This procedure
                            also create the mailboxes required for transmit
                            operation and stores their id in the function
*                           code mailbox directory
*
*       INPUT:              Pointer to Layer Instance common table
*
*       OUTPUT:
*
*
*****************************************************************
*/

/*************************************************************/
/* The following conventional 'C' code header file is used
* by this process. See the respective files for definitions,
* structures and macros.
*************************************************************
*/

   /* START  */
    /* Save the pointer to the LICA table for this instance */

    /* NEED TO REVISIT            */

    /* Create a mailbox for receive functions, save default for alarms  */
    /* need limit to 16 or so, and error condition detecting handling */

    /* Add LLC Receive mailbox ID to the function directory */

    /* Create the mailbox for MAC_DATA.indication messages */
    /* NEED TO ALLOCATE AN EMERGENCY MESSAGE                    */

    /* WHY DO WE NEED THIS                                      */

    /* Declare step 2 initialization completed */

    /* END    */

}       /* llc_rx_init */

/*****************************************************************/
/*
* .....................LLC Receive Process ..................
*   This is the operational receive process code for each LLC
* layer instance. Mailbox messages received will be decoded by
* the switch statement into procedure calls for the cases listed
* below. Each call will pass a pointer to the message and the
* pointer to the LICA table as parameters.
*
*       TITLE:          LLC RECEIVE PROCEDURE
*
*       FUNCTION:       Receives all messages directed to the LLC receive
*                       process for a layer instance.  It then determines
*                       the correct procedure to call to process the request.
*
*       INPUT:          Pointer to message
*                       Mailbox name     ------- what for ?
*
*       OUTPUT:
*
* .....................LLC Receive Process ..................
*   This is the operational receive process code for each LLC
* layer instance. Mailbox messages receiveed will be decoded by
* the switch statement into procedure calls for the cases listed
* below. Each call will pass a pointer to the message and the
* pointer to the LICA table as parameters.
****************************************************************
*/
    /* Retrieve the Layer Instance Common Area pointer */

   /* DO CASE on message type */
      /* CASE  DATA.indicate message from MAC */
         /* Call procedure to receive a PDU from the MAC layer  */
         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */

      /* CASE  An IOLD request message */
         /* increase IOLDs counter for statistics */
         /* Call common routine to transfer LCBI to LACS from IOLD msg */
         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */
      /* CASE The request to DMA a LCBI to the LACS has returned */
         /* Call procedure to interpret new LCBI              */
         /* increase LCB to lacs counter for statistics */
         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */
      /* CASE The request to DMA a buffer descriptor to the LACS has returned */
         /* increase buffer to lacs counter for statistics */
         /* Call routine to fetch data buffer from L6    */
         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */

      /* CASE The request to DMA a buffer to the L6 has returned */
         /* increase buffer to L6 counter for statistics */
         /* Call routine to request lcbi be moved to L6  */

         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */

      /* CASE The request to DMA a LCBI to the L6 has returned */
         /* Call procedure to terminate LCBI and free all memory          */
         /* increase LCB to L6 counter for statistics */
         /* SHOULD BE be making sure all memory returned        */

         /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE   */

      /* OTHERWISE */
         /* ******************************************************************
          *  The message can not be decoded into any that are expected by this
          * process.
          * At this stage, the space used
          *  by the message is returned to the kernel
          * until a method is designed to handle the problem.
          ******************************************************************
         */

         /* set up unknoen error counter */
         /* send event message to the sme */

    /* ENDCASE End of the message type decoding */
}       /* llc_rx */

/***********************************************************/
/* -------------------------- Megabus Interface ----------------------- */
/* The request to DMA a LCBI from the L6 to the LACS has returned */
/*
*       TITLE:              INTERPRET RECEIVE LCB REQUESTS PROCEDURE
*
*       FUNCTION:           Receives all DMA confirm messages due to the completion
*                           of a transfer of a LCBI to the LACS memory.  This
*                           routine determines if an additional transfer of a
*                           buffer descriptor is required and sends a message to
*                           the DMA software if it is neccesary. If a buffer
*                           descriptor is not associated with the LCB then this
*                           procedure will determine whether this is an event LCB
*                           or connectionless read request and call the appropriate
*                           procedure.
*
*       INPUT:              Pointer to message
*                           pointer to the layer instance table
*
*       OUTPUT:
```

```
*
***************************************************************
*/

   /* ***********************************************************
    *  This message has been returned from DMA services. The following
    * code will check the results of the
    * transfer request and determine if it was
    * successful.
    *************************************************************
   */
                          /* Local lsap index (equivalent to logocal address) */

    /* Cast on a transacation block structure      */

    /* IF the request was successful */
      /* THEN                                                   */
         /* DO CASE on the function code in the LCBI               */

            /* CASE of an event LCB                                    */
              /* EVENTS ARE NOT SERVICED UNTIL LATER                      */

            /* CASE of a Connectionless read request                   */
              /* fetch local sap index */

              /* IF the LSAP number is valid and the
                        LSAP is ACTIVE (INUSE  state) */

                  /* Set the local lsap table pointer into transaction block */

                  /* Set the local lsap logical address into
                        transaction block   */
                  /* ************************************************************
                  *  When the buffer pointer is in the LCBI, a search of the
                  *  'waiting pdu ' queue is made.  If one is found, the LCBI
                  *  will be completed and returned, otherwise the LCBI will
                  *  be added to the 'waiting LCBI' queue.
                  *************************************************************
                  */

                  /* If the pdu waiting queue is empty */
                        /* check bound limit */
                            /* over the bound limit, drop the request */

                            /* Set LCBI function specific
                                        status to  queue full */
                            /* set the read credit count for
                                        lcb user flow control */
                            /* Indicate LCBI completion in the status */
                            /* Call common procedure to return LCBI     */
                         }      /* if */
                        /* else go ahead to queue up the request */
                            /* update the queue counter */
                            /* Add the transaction blk to the end of the
```

```
                    waiting list */
                }               /* else */
            /* ELSE  (There are pdu's waiting) */
               /* Fetch first transaction blk containing the
                       data indicate  */
               /* Set the LCBI actual size from the waiting
                       PDU trans.blk. */
                       /* subtract three for LLC header   */

               /* IF LCB buffer is large enough to contain the PDU */
                       /* Need to subtract the llc address header */

                 /* THEN send it back to the L6         */

                    /* Remove the first message from the
                               waiting queue */
                               /* delete it from queue */
                    /* update the queue counter */

                    /* Strip off LLC header from
                       PDU and set buffer desc into trans.blk */
                    /* EXTRA STRIP OFF HEADER GET OUT BY M. LU 2/19/86

                     /* Copy all pertinent info from
                     /* L6 memory pointer */
                     /* Buffer Descriptor pointer */
                     /* L6 Buffer Descriptor pointer */
                     /* LCBI size  */
                     /* channel stuff  */

                     /* Return the buffer to the L6 */

                     /* Free memory of  transaction block
                               containing the LCBI    */
                 }      /* if */
                 /* ELSE Return LCB with status         */
                    /* Set LCBI function specific
                       status to buffer too small */

                    /* set the read credit count for
                       lcb user flow control */

                    /* Indicate LCBI completion in the status */

                    /* Call common procedure to return LCBI    */
               /* ENDIF */
             /* ENDIF                                          */

         /* ELSE bad LSAP     */

              /* Update LCBI to reflect bad local LSAP     */

              /* set the read credit count for lcb user flow control */
```

```
                        /* Indicate LCBI completion in the status */
                        /* Return LCBI to L6                          */
                      /* ENDIF   */

         /* OTHERWISE    bad function code                         */
           /* Update LCBI to reflect bad function code     */
           /* set the read credit count for lcb user flow control */
           /* Indicate LCBI completion in the status */
           /* Return LCBI to L6                          */
         /* ENDCASE                             */
     /* ELSE */
         /* Set invalid status in the LCBI */
         /* Update LCBI to reflect bad function code     */
         /* Set status in HW error word ??????          */
         /* set the read credit count for lcb user flow control */
         /* Indicate LCBI completion in the status */
         /* Try to send it back anyway                         */
      /* ENDIF */
}               /* rcv_lcbi_to_lac */

/*****************************************************************/
/* ------------------------ Megabus Interface ---------------------- */
/* The request to DMA a buffer to the L6 has returned */
/*****************************************************************/

    /* ****************************************************************
     * This message is returned by DMA services when it completes the
     * transfer of a buffer from the LACS board to the L6 as part of a read
     * transaction. This step will set the status in the LCBI and request DMA
     * services to return the LCB to the L6.
     *****************************************************************
    */

    /* IF Buffer transfered without any problems              */
         /* THEN                                               */
         /* set the read credit count for lcb user flow control */
         /* set remote logical address */
         /* Indicate LCBI completion in the status */
         /* Return the LCBI to L6 memory */
      }          /* if NULL */
    /* ELSE                                          */
         /* Set status in LCBI accordingly                          */
         /* Set status in HW error word ??????          */
         /* set the read credit count for lcb user flow control */
         /* Indicate LCBI completion in the status */
         /* ALL MEMORY WILL BE FREED AT COMOPLETION OF LCBI TRANSFER  */
         /* Return the LCBI to L6 memory */
      }          /* else */
    /* ENDIF                                               */
}               /* rcv_buf_to_l6 */

/*******************************************************************
*
*       LLC LAYER XID RESPONSE PROCEDURE
```

```
*
*        call:    llc_xid(indmsg,lica_p)
*
*        FUNCTION:
*                 When recieve a XID request from the LAN
*                 llc layer will set up response message
*                 and send back to the XID request source.
*
*        DESCRIPTION:
*
*******************************************************************
*/

  /* Point to the buffer */
  /* Call procedure to validate the PDU request for tnis SAP     */
    /* IF it is a valid PDU                                      */
         /* Fetch the local LSAP table
                       ignore group addresses for now */
         /* swap the llc address */
         /* cast to data request message */
         /* updating xid xid counter */
         /* count out address for xid message */
         /* put in xid response information */
         /* stander info */
         /* put lsap name */
         /* get lsap name */
         /* set message type */
         /* set the priority */
         /* swap the mac address */
          }      /* for */
         /* putin own mac address */
         /*strcpy(rindmsg->layer.mac.source_addr.octet,l_lsap_tbl->mac_adr);*/
         /*for (index = 0; index < l_lsap_tbl->mac_adr_lngth; index++)
          }      * for */
         /* send to mac for tranfmit out */
             /* free the message */
             /* update message dropped counter */
             /* free the message */
          }      /* else if M3Xfull */
     }  /* if llcerr */
         /* Drop the incoming pdu, the LSAP is not defined */
         /* update pud drop counter */
         /* Free the MAC message */
     }  /*else */
}       /* end llc_xid */

/******************************************************************
*
*        LLC LAYER TEST REQUEST PROCEDURE
*
*        call:    llc_test(indmsg,lica_p)
*
*        FUNCTION:
*                 When recieve a TEST request from the LAN
*                 llc layer will set up response message
*                 and send back to the TEST request source.
*
*        DESCRIPTION:
*
*******************************************************************
*/

  /* Point to the buffer */
  /* Call procedure to validate the PDU request for this SAP     */
    /* IF it is a valid PDU                                      */
         /* Fetch the local LSAP table
                     ignore group addresses for now */
         /* swap the llc address */
         /* updating test counter */
         /* cast to data request message */
         /* set message type */
         /* set the priority */
         /* swap the mac address */
         /* putin own mac address */
         /*strcpy(rindmsg->layer.mac.source_addr.octet,l_lsap_tbl->mac_adr);*/
         /*for (index = 0; index < l_lsap_tbl->mac_adr_lngth; index++)
         /* send to mac for tranfmit out */
             /* free the message */
             /* update message dropped counter */
             /* free the message */
          }     /* else if MBXfull */
     }  /* if llcerr */
    else        /* bad test pdu */
         /* Drop the incoming pdu, the LSAP is not defined */
         /* update pud drop counter */
         /* Free the MAC message */
}       /* end llc_test */

/******************************************************************/
/* ----------------------- LLC SAP Component -------------------- */
/*
*        TITLE:          VALIDATE RECEIVED PDU PROCEDURE
*
*        FUNCTION:       Validates a pdu received from the MAC data
*                        indicate mailbox.  It determines if local and
*                        remote LSAPs have been defined for the layer
*                        instance and whether they are in the proper state.
*                        In addition the PDU size is checked against the
*                        maximum PDU size for this SAP.  The remote lsap
*                        logical address is returned as well as the error
*                        status.
*
*        INPUT:          Pointer to MAC data indicate message
*                        Pointer to LLC PDU
*                        Pointer to Layer Instance common table
*
*        OUTPUT:         Pointer to Remote logical address
*                        Error status
```

```
*
*************************************************************************
*/

 /* A MAC_DATA.indicate message */
    /* Clear an error flag                              */
    /* Validate the logical address of the remote and local SAPs */
    /* IF the local log. SAP addr. (same as the DSAP)
      is greater than the local LSAP dir. size */
       /* Report a failure */
       /* Set error code */
    /* ENDIF */
    /* Fetch the local LSAP table
                      ignore group addresses for now */
    /* Validate that the local SAP exists                    */
    /* IF the local LSAP does not exist  */
       /* Set the state of the LSAP and Layer Instance accordingly  */
       /* Set LCBI status words                         */
       /* Report a LSAP failure */
       /* Set error code */
    /* ENDIF */
    /* Validate the state of the layer instance              */
    /* IF the layer instance is not in the INUSE state */
       /* Set the state of the LSAP and Layer Instance accordingly  */
       /* Set LCBI status words                         */
       /* Report a failure */
       /* Set error code */
    /* Validate the state of the local LSAP             */
    /* IS THERE A PROBLEM HERE IF THIS CODE IS EXECUTED EVEN THOUGH
                SAP IS NON-EXISTENT
                                 access undefined memory
                                        - bus error      */
    /* IF the major state of the LSAP is not INUSE   */
       /* Set the state of the LSAP and Layer Instance accordingly  */
       /* Set LCBI status words                         */
       /* Report a LSAP failure */
       /* Set error code */
    /* ENDIF */
    /* Validate that the LSAP limits are not exceeded            */
    /* IF the PDU size exceeds the maximum PDU size        */
       /* Update statistical counters for LSAP and Layer Instance accordingly */
       /* Set LCBI status words                         */
       /* Report a PDU failure */
       /* Set error code */
    /* ENDIF */
    /* Validate the Remote LSAP                              */
    /* Fetch the full remote LSAP address (llc + mac)                  */
       /* Fetch the remote LSAP table , ignore group addresses for now */
       /* If the SSAP (remote SAP) has been not activated
        and local LSAP permits dynamic  addition to Activate directory */
           /* THEN              */
           /* Add the remote to activated list  */
           /* Allocate memory for a remote LSAP table   */
           /* Call procedure to search for next available LSAP entry */
```

```
         /* ELSE IF remote SAp has not been activated    */
            /* Update statistical counters for
                       LSAP and Layer Instance accordingly  */
            /* Set LCBI status words                        */
            /* Report a Bad remote SAP failure */
            /* Set error code */
      /* END IF */
      /* Set remote logical address to activate directory index            */
    }   /* end llcerr == NULL */
}              /* validate pdu */
/**************************************************************
*
*       LLC LAYER MANEGER  DEACTIVATE FUNCTION
*
*       This function will clean up all lcb queue llc_trans
*       blocks, clean up all pdu queue llc_trans blocks.
*       set up locked state, and post back all lcbs
*
**************************************************************/

          /* get the l_sap_table ptr */
          /* validate lsap table */
                  /* set up layer NULL ***LOCK state */
           }      /* end if */
                  /* return not valid lcb error */
                  /* completed with error */
                  /* return the bad lcb */
                  /* get out here */
           }      /* else */
         /* pull the local l_lsap_table address */
          /* get the l_sap_table ptr */
          /* delete from the lcb queue */
                  /* kernel macro function foir queue */
                  /* get the on just take of the queue */
           }      /* if */
          /* loop to clean up lcb queue trans blk */
                  /* for level 5 information, lcb return */
                  /* level 5 completion bit set up, and err status set */
                  /* post back lcbi */
                  /* delete from the lcb queue */
                  /* kernel macro function foir queue */
                  /* get the on just take of the queue */
           }      /* while */
          /* get the original trans */
                  /* keep track the pdu droped */
                  /* get the next pdu trans at the queue */
                  /* dump the trans blk */
                  /* get the next one to check out */
           }      /* if */
          /* clean the pdu queue */
                  /* keep track the pdu droped */
                  /* get the next pdu trans at the queue */
                  /* dump the trans blk */
                  /* get the next one to check out */
          }     /* while */
         /* get the original trans */
         /* for clean the r_lsap table */
         /* loop to clean when r_lsap table not empty */
                /* NULL the r_lsap table pointers */
                /* counter */
          }     /* while */
     /* send event message to system management
        /*
         send event message to system management module *
        */
           /* return event lcbi for completed deactive */
         /* the last lcbi need to set up and send back */
         /* set up for level 6 completion bit and err status */
         /* post back lcbi */
}       /* llc_lme_da */

/****************************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/* Message aging alarm message */
/****************************************************************************/

   /* ***********************************************************************
    * This message is the returning alarm request from setalarm. All l_pdu
    * messages attached to all
    * LSAPs will be aged one time increment. Any message
    * found to be over the age limit will be released back to the buffer pool.
    * The assumption is that overaged messages are unexpected and a LCB will
    * never be issued.
    ************************************************************************
   */
   /* IF this instance is in the ready state then */
       /* FOR each IEEE802 defined LSAP that can exist */
          /* Point to the LSAP table */
          /* IF the LSAP has been defined */
             /* DO for each message on the waiting-pdu queue */
                /* Increment the aging counter */
                /* IF the aging counter is GTE the limit */
                   /* Remove the message from the queue */
                   /* update the queue counter */
                   /* Release the memory to the memory pool */
                   /* Notify SM of the event */
                /* ENDIF */
                /* Roll thru the queue to the next entry */
             /* ENDDO */
          /* ENDIF */
       /* ENDFOR */
    /* ENDIF */
    /* Request the next aging alarm period */
```

```
/*****************************************************************
*****************************************************************
***
***
***                              HONEYWELL
***
***
***                          Local Area Network
***
***            L O G I C A L   L I N K   C O N T R O L
***
***                              L A Y E R
***
***                          transmit module
***
***
***     Copyright: Honeywell Information System
***                All rights reserved.
***
***     Cereate Date:       1/25/85
***     By:                 D. Oshaughn
***     Discription:
***
***     Functions & Parameters:
***
*
*       llc_tx_init (lica_p)
*         LICA *lica_p;  * Pointer to the layer instance common area *
*       llc_tx (mbxptr,mbx_name)
*         MSG *mbxptr;
*         MBID mbx_name;
*       xmit_lcpi_to_lac(mbxptr,lica_p)
*         MSG *mbxptr;
*         LICA *lica_p;  * Pointer to the process variables *
*       mac_conf_err(mbxptr,lica_p)
*         MSG *mbxptr;
*         LICA *lica_p;
*       xmit_buf_to_lac(mbxptr,lica_p)
*         MSG *mbxptr;
*         LICA *lica_p;
*       validate_data_req(llc_trans,lica_p)
*         LLC_TRANS *llc_trans;
*         LICA *lica_p;
*       iss_mac_dreq(llc_trans,lica_p)
*         LLC_TRANS *llc_trans; * Pointer to transaction block for LLC PDU *
*         LICA *lica_p;              * Pointer to the instance table *
*
***
***     Revision #:     1
***     Date:           3/25/85
***     By:             M. Lu
***     Discription:
***
```

```
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***
*********************************************************************
*********************************************************************
*/

/*********************************************************************/
/*
* ....................LLC Layer Management....................
*   This is the LLC Transmit Process that exists once for each
* LLC Layer instance created. The combination of this process,
* the recieve process and the layer management process constitutes
* one LLC Layer instance.
*/
/*
*        TITLE:           LLC TRANSMIT INITIALIZATION PROCEDURE
*
*        FUNCTION:        Initialize the Transmit process by storing
*                         the pointer to the layer instance common table
*                         in the transmit process PC3.  This procedure
                          also create the mailboxes required for transmit
                          operation and stores their id in the function
*                         code mailbox directory>
*
*        INPUT:           Pointer to Layer Instance common table
*
*        OUTPUT:
*
*....................LLC Transmit Initialization............................
*  This code initializes a transmit process for a LLC Layer instance.
*
**************************************************************************
*/

/* ***********************************************************************
*  The following conventional 'C' code header file is used
* by this process. See the respective files for definitions,
* structures and macros.
**************************************************************************
*/
    /* START                                */
    /* Save the pointer to the LICA table for this instance */
    /* NEED TO REVISIT                                        */
    /* Create mailbox for transmit operations                           */
    /*                                           save default for alarms */
    /* Add LLC Transmit mailbox ID to the function directory */
    /* NEED TO ALLOCATE MEMORY FOR AN EMERGENCY MESSAGE */
    /* WHY DO WE NEED THIS                        */
    /* Declare step 3 initialization completed */
    /* END                                  */
```

```
}       /* llc_tx_init */

/*
*****************************************************************************
*
*       TITLE:          LLC TRANSMIT PROCEDURE
*
*       FUNCTION:       Receives all messages directed to the LLC transmit
*                       process for a layer instance.  It then determines
*                       the correct procedure to call to process the request.
*
*       INPUT:          Pointer to message
*                       Mailbox name      ------- what for ?
*
*       OUTPUT:
*
* ....................LLC Transmit Process ..................
*   This is the operational transmit process code for each LLC
* layer instance. Mailbox messages transmited will be decoded by
* the switch statement into procedure calls for the cases listed
* below. Each call will pass a pointer to the message and the
* pointer to the LICA table as parameters.
****************************************************************************
*/
    /* Retrieve the Layer Instance Common Area pointer */
    /* DO CASE on message type */
      /* CASE  An IOLD request message */
          /* Call common routine to transfer LCBI to LACS from IOLD msg */
          /* update IOLDs counter for tx for statistics */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
      /* CASE The request to DMA a LCBI to the LACS has returned */
          /* update LCB to lac tx counter for statistics */
          /* Call procedure to interpret new LCBI              */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
      /* CASE The request to DMA a buffer descriptor to the LACS has returned */
          /* Call routine to fetch data buffer from L6     */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
      /* CASE The request to DMA a buffer to the LACS has returned */
          /* update BUF to lac tx counter for statistics */
          /* Call routine to create LLC PDU and deliver to the MAC layer    */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
      /* CASE The LACS Network Layer has a Data Request for LLC  */
          /* Call routine to create LLC PDU and deliver to the MAC layer    */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
      /* CASE The request to DMA a LCBI to the L6 has returned */
          /* Call procedure to terminate LCBI and free all memory           */
          /* update LCB to l6 tx counter for statistics */
          /* SHOULD THIS JUST BE "return_lcbi" procedure call               */
          /* SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    */
       /* CASE The MAC data request has returned as
                an error or a confirm error *
        * Call procedure to handle error and free all memory         *
        * SHOULD BE RECEIVING AND CHECKING AN ERROR CODE HERE    *
       */
```

```
        /* OTHERWISE */
           /*************************************************************
            *  The message can not be decoded into any that are expected by this
            * process.
            * At this stage, the space used by the
            * message is returned to the kernel
            * until a method is designed to handle the problem.
            ************************************************************
           */
         /* ENDCASE End of the message type decoding */
    }      /* switch */
}          /* llc_tx */
/*******************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/* The request to DMA a LCBI from the L6 to the LACS has returned */
/*
*        TITLE:              INTERPRET TRANSMIT LCB REQUESTS PROCEDURE
*
*        FUNCTION:           Receives all DMA confirm messages due to the completion
*                            of a transfer of a LCBI to the LACS memory.  This
*                            routine determines if any other transfers are required
*                            to fetch the data associated with the request.  When
*                            the data is contained in the LCB, this procedure
*                            fetches a buffer to treansfer the data to MAC, attaches
                             a LLC header and issues a data request to MAC.
                             Otherwise a request is made to transfer the data buffer
*                            or the buffer descriptor to the LACS.
*
*        INPUT:              Pointer to message
*                            pointer to the layer instance table
*
*        OUTPUT:
*
*********************************************************************
*/

     /*************************************************************
      *  This message has been returned from DMA services. The following
      * code will check the results of the transfer request and if it was
      * successful.
      *************************************************************
     */
      /* IF the request was successful */
        /* THEN                                                   */
        /* DO CASE on the function specific function code         */
           /* CASE of Connectionless Transmit request               */
            /* DO CASE on the type of data buffer in LCBI */
               /* CASE of data buffer contained in the LCB           */
                 /* Fetch a buffer to move the data to MAC             */
                 /* Set the total range */
                 /* Get a buffer that is the requested
                          length + 4 bytes for headers */
                 /* allocate extra head 32 bytes and tail 128 bytes */
                 /* space for MAC handle small buffer  M. Lu */
    /* Reduce the buffer size by 4 bytes for DMA */
    /* and extra header space for mac continguous buffer */
    /* here because the old kernel not garantee return
     new buffer discripter */
    /* so we keep using the old one.
           not legal for the new kernel M. Lu */
    /* now load in the data to buffer */
     /* reserved space for data in lcb */
    /* Clear all remaining fields */
    /* Format it as a LLC unnumbered information frame  */
    /* Issue it as a MAC data request                   */
    /* Complete the LCBI status and completion word     */
    /* Issue message to the DMA                         */
    /* set the read credit count for lcb user flow control */
    /* Set the LCBI completion bit                         */
    /* Return the LCBI to L6 memory */
  /* CASE of data buffer pointer(s) contained in the LCBI */
    /************************************************************
     * When the buffer pointer is in the LCBI, the buffer
     * containing the message must be transfered across from the L6
     * to LACS buffer memory. Space is allocated for the message and
     * a request submitted to DMA services to make the move.
     ************************************************************
     */
    /* Request the buffer be transferred from the L6 */
    /* Have the message return to the DMA done mailbox */
    /* Set priority to normal */
    /* Set the number of L6 buffers       */
    /* Initiailize total range to zero                  */
    /* DO FOR all buffers in the LCBI                       */
       /* Set the L6 buffer descriptors */
       /* Set the L6 buffer range */
       /************************************************************
        *  There is a problem if the total range exceeds 32K words
        *                  fortunately LLC will always be less BUT
        *                  DO WE NEED TO VALIDATE RANGE and
        *                  WHAT ABOUT LCB FORMAT ? (32 or 16 bits)
        *                  and WHAT ABOUT LCB FORMAT - short now
        ************************************************************
       */
       /* Set the total range */
       /* ENDDO                        */
    }       /* switch */
  /* Get a buffer that is the requested
           length + 4 bytes for headers */
  /* allocate extra head 32 bytes and tail 128 bytes */
  /* space for MAC handle small buffer  M. Lu */
  /* Reduce the buffer size by 4 bytes for DMA */
  /* and extra header space for mac continguous buffer */
  /* here because the old kernel
     not garantee return new buffer discripter */
  /* so we keep using the old one.
           not legal for the new kernel M. Lu */
  /* Clear all remaining fields */
```

```
              /* Send the request message to DMA services */
                  /* special routine for llc_trans release */
               }        /* else if full */
           /* CASE of a buffer descriptor pointer contained in the LCBI */
            /************************************************************
              *  When a buffer descriptor pointer is in the LCBI, the buffer
              * descriptor must be transfered across from the L6
              * to LACS buffer memory. Space is allocated for the message and
              * a request submitted to DMA services to make the move.
              *
              *  This is identical to the LCB transfer performed by the IOLDs
              * procedure.
              ************************************************************
            */
              /* Request the buffer be transferred from the L6 */
              /* Set invalid status in the LCBI */
              /* set the write credit count for lcb user flow control */
              /* Set the LCBI completion bit                        */
              /* Return the LCBI to L6 memory */
           /* OTHERWISE */
              /***************************************************************
               * Can not decode the buffer type into anything
               * that can be processed by this function. Update the LCBI
               * status to reflect the problem and return the LCBI to
               * the L6.
               **************************************************************
              */
              /* Set invalid status in the LCBI */
              /* set the write credit count for lcb user flow control */
              /* Set the LCBI completion bit                        */
              /* Return the LCBI to L6 memory */
           /* ENDCASE */
        }       /* switch */
      /* OTHERWISE */
      /* Only a connectionless transmit request is currently supported */
      /**************************************************************
       *
       * Can not decode the requested function into anything
       * that can be processed by this function. Update the LCBI
       * status to reflect the problem and return the LCBI to
       * the L6.
       **************************************************************
      */
      /* Set invalid status in the LCBI */
       /* set the write credit count for lcb user flow control */
      /* Set the LCBI completion bit                           */
      /* Return the LCBI to L6 memory */
     /* ENDCASE */
   }    /* switch */
   /* ELSE */
      /* Set invalid status in the LCBI */
      /* Try to update status in LCB and free memory or something */
   /* ENDIF */
}       /* xmit_lcbi_to_lac */
```

```
/************************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/*  The L_DATA.request transaction has returned fromm the LLC layer as a
* L_DATA.confirm. The buffer and message memory must be released.
*
**************************************************************************
*/

   * This message is the returning MAC_conf or MAC error.
   * Release the buffer space *
   * Return the MAC message to LACS memory *
*/

/************************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/* The request to DMA a data buffer from the L6 to the LACS has returned */
/*
*       TITLE:           ISSUE MAC DATA REQUESTS PROCEDURE
*
*       FUNCTION:        This routine is invoked when a buffer is available
*                        to be issued to MAC as an LLC PDU.  Checks are made
*                        to make sure that the layer instance and the LSAP are
*                        in the INUSE state.  All validation of the request
*                        is performed, such as PDU parameters, local PDU limits
*                        remote LSAP limits and param,eters, etc.  A LLC
*                        header for an unnumbered I frame is appended to the
*                        buffer and passed to the MAC layer mapped to this layer
*                        layer instance.  The LCBI is then marked completed and
*                        the DMA firmware is requested to transfer it back to
*                        the L6.
*
*       INPUT:           Pointer to message
*                        pointer to the layer instance table
*
*       OUTPUT:
*                        The request to DMA a buffer from the L6 has returned *
**************************************************************************
*/
   /*************************************************************
    *
    *  This message is returned by DMA services when it completes the
    * transfer of a buffer from the L6 to the LACS board as part of a
    * transmit transaction. This step will add the LLC layer addressing
    * to the message and send the lpdu to MAC for transmission.
    *************************************************************
   */
    /* Clear an error flag                          */
    /* Validate that the transfer was succesful             */
    /* IF the transfer was not a success */
       /* Set the state of the LSAP and Layer Instance accordingly  */
       /* Set LCBI status words                        */
       /* set the write credit count for lcb user flow control */
       /* Set the LCB status to completed */
```

```
      /* Return the LCBI to L6 memory */
      /* Report a DMA failure */
      /* Set error code */
   /* ELSE every thing is honky-dory                      */
     /* Set a pointer to the local and remote
                       lsap tables into transaction blk. */
     /* Masking MS 16 bits of log
                       addr.is restrictive but might save some grief */
     /* Call procedure to validate the data request            */
          /* set the write credit count for lcb user flow control */
          /* set error code */
          /* Set status in the LCBI                                          */
         /* Call procedure to transfer completed LCBI                       */
     /* Fetch the address of the local table
     from the layer instance local LSAP directory */
     /* Fetch the address of the remote table
     from the activate directory of the LSAP */
     /* update UI_tx counter for statistics */
     /* Call procedure to create PDU and issue it to MAC               */
     /* IF no errors have been detected yet                      */
       /***************************************************************
       /* THEN
        *
        * Update Local SAP statistics
        ***************************************************************
       */
       /* Update Layer Instance statistics                        */
       /* *************************************************************
        * IF Remote SAP Statistic flags indicate to always collect
        *                        statistics or only if local SAP indicates to
        *                        collect it on the remote
        **************************************************************
       */
       /* THEN */
       /* Update remote SAP statistics                                 */
       /* ENDIF                                                          */
       /* set the write credit count for lcb user flow control */
       /* Set status in the LCBI                                         */
       /* Call procedure to transfer completed LCBI                      */
   /* ELSE Error        */
       /* set the write credit count for lcb user flow control */
       /* and error */
      /* Return the LCBI to L6 memory */
   /* ENDIF                                               */
  /* ENDIF */

/**************************************************************************/
/* ---------------------------- Megabus Interface ---------------------- */
/* Validate a LLC transmit data request procedure
*
       TITLE:          VALIDATE LLC DATA REQUESTS PROCEDURE
*
*      FUNCTION:       This routine is invoked when a buffer is available
*                              to be issued to MAC as an LLC PDU.  Checks are made
*                              to make sure that the layer instance and the LSAP are
*                              in the INUSE state.  All validation of the request
*                              is performed, such as PDU parameters, local PDU limits
*                              remote LSAP limits and param,eters, etc.
*
*       INPUT:                 Pointer to transaction block
*                              pointer to the layer instance table
*
*       OUTPUT:
*
************************************************************************
*/

     /* Clear an error flag                                     */
     /* Validate the logical address of the remote and local SAPs */
     /* IF the local log. SAP addr. is greater than the local LSAP dir. size */
        /* Report a failure */
        /* Set error code */
     /* ENDIF */
     /* IF the remote logical SAP addr.
                                is greater than the remote LSAP dir.sz. */
        /* Report a failure */
        /* Set error code */
     /* ENDIF */
     /* Validate the state of the layer instance              */
     /* IF the layer instance is not in the INUSE state */
        /* Set the state of the LSAP and Layer Instance accordingly  */
        /* Set LCBI status words                              */
        /* Report a failure */
        /* Set error code */
     /* ENDIF */
     /* Validate that the local SAP exists                    */
     /* IF the local LSAP does not exist  */
        /* Set the state of the LSAP and Layer Instance accordingly  */
        /* Set LCBI status words                              */
        /* Report a LSAP failure */
        /* Set error code */
     /* ENDIF */
     /* Validate the state of the local LSAP              */
     /*********************************************************************/
     /* IS THERE A PROBLEM HERE IF THIS CODE IS EXECUTED EVEN THOUGH
     *           SAP IS NON-EXISTENT
     *                                         access undefined memory
     *                                                  - bus error
     ********************************************************************
     */
     /* IF the major state of the LSAP is not INUSE   */
        /* Set the state of the LSAP and Layer Instance accordingly  */
        /* Set LCBI status words                              */
        /* Report a LSAP failure */
        /* Set error code */
     /* ENDIF */
     /* Validate that the LSAP limits are not exceeded           */
```

```
    /* IF the PDU size exceeds the maximum PDU size        */
      /* Set the state of the LSAP and Layer Instance accordingly  */
      /* Set LCBI status words                             */
      /* Report a PDU failure */
      /* Set error code */
    /* ENDIF */
    /* Validate the Remote LSAP                             */
    /* IF the Remote SAP is not active  */
      /* Set the state of the LSAP and Layer Instance accordingly  */
      /* Set LCBI status words                             */
      /* Report a LSAP failure */
      /* Set error code */
    /* ENDIF */
    /* Just watch how */
    /* Return with error code                                  */
}       /* validate_data_req */

/*************************************************************************/
/* ------------------------ LLC SAP Component -------------------- */
/*
*        TITLE:           FORMAT UI LLC PDU PROCEDURE
*
*        FUNCTION:        Receives a transaction block containing a pointer to
*                         a buffer of data.  The source and destination are
*                         are added to the LLC header and the unnumbered
*                         control header is added.  A MAC data request message
*                         is prepared and issued to the MAC transmit process
*                         mailbox.
*
*        INPUT:           Pointer to LLC transaction block
*                         Pointer to layer instance table
*
*        OUTPUT:          Error Code
*
*************************************************************************
*/
    /*********************************************************************
    *  The message is a LLC transaction block complete with pointers to the
    * the local LSAP table, the remote LSAP table, the LCBI and the buffer.
    ********************************************************************
    */
    /* Allocate memory for a MAC data request message */
    /* Set the message type into a MAC DATA.request */
    /* Set the priority                          */
    /* Set the return mailbox ID */
    /* SHOULD THIS BE NULLED ???*/
    /* / ???  mac_dreq_msg->return_id = lica_p->xmit_mbx; */

    /* Add 3 bytes of LLC address to the begining of the buffer */
        /*                            for LLC header - what about MAC */
    /* Add the LLC addressing (remote and local SAPs) to the PDU buffer */
    /* Use a type I unnumbered I-frame command */
    /* Set the class of service into the message        */
    /* Move the destination MAC address into the message */
```

```
     /* NEED TO TALK ABOUT THIS _ WOULD COPY BUFFER WORK BETTER     */
     /* Set the message buffer descriptor address to point to the LLC PDU */
     /* Send the transaction to MAC for transmission */
        /* free the message */
        /* update message dropped counter */
        /* free the message */
     }   /* else if MBXfull */
     /* Return    */
}         /* iss_mac_dreq */
```

```
/*
*   This file (llc_const.h) defines the constants used in the LLC
* Layer processes for the LACS board.
*/

/* Define commands, next read, next received, poll-final bit masks */
/* for LLC frames. (See struct llc_frames)  */

/* Supervisory frames */
  /* Supervisory bit */
  /* Receiver Ready command */
  /* Reject command */
  /* Receiver Not Ready */
  /* Poll-Final */
  /* Next Received count */

/* Information frames */
 /* Information bit */
 /* Next Send count */
 /* Poll-Final */
/*          N_r_         0XFE00     Next Received count */

/* Unnumbered Frames */
    /* Unnumbered bits */
    /* Set Async Bal Mode Ext */
    /* Disconnect */
    /* Unnumbered Acknowledge */
    /* Disconnect Mode */
    /* Frame Reject */#
  /* Exchange ID P bit clear */
  /* Exchange ID P bit set */
  /* Test command P bit clear */
  /* Test command P bit set */
 /* Poll-Final */

/* Length of UI PDU Header  */

/* add special for mac layer send small package, preallocate header 32 bytes */
/* M. Lu 2/86 */
/* 128 bytes more tail for mac */

/* Bit masks for words in the LCBI memory block (See struct lcbi_mblk) */

/* The interrupt control word (cbicw) */
   /* Bits 0-7 Channel */
   /* Bits 8,9 CPU     */
   /* Bits 10-15 Intr level */

/* The function code word (cbfsf) */
   /* Bit mask for function codes */
   /* Values that may appear in the function code field */
   /* Connectionless Receive function code */
   /* Connectionless Transmit function code */
   /* Event function code */
```

```
    /* Activate local function code */
    /* Activate remote function code */
    /* Deactivate local function code */
    /* Deactivate remote function code */

    /* Activate for local or remote function code */

/* The controller status word (cocts) */
    /* Invalid function code */
    /* RAM memory exhaust */
    /* RAM location non-existent */
    /* RAM parity error */
    /* L6 memory yellow */
    /* Non-existent L6 memory */
    /* L6 bus parity error */
    /* L6 memory red */

/* The function specific status word (cbfss) */
    /* SAP not active */
    /* Lack of resources */
    /* Controller unavailable */
    /* SM error */
    /* SAP already active */
    /* Undefined */
    /* SAP already deactivated */
    /* Receive buffer too small */
    /* Illegal logical address */
    /* Invalid LCB */
    /* Write credit violation */
    /* Read credit violation */

/* The completion word (cdcbs) */
    /* Completion bit */
    /* LCB not processed */

/* Bit masks for the IOLD command word */

/* The "range word"  */
    /* Function code */
    /* Length in bytes */

/* The "Channel Number or Layer Instance Number"   */
    /* Controller address */
    /* Layer number */
    /* Layer instance */

/* Index values for the Function Directory Mailbox array */
/*  See struct lica, element fc_mbx[16]  */

    /* Receive */
    /* Transmit */
    /* Event */
    /* Activate */
    /* Deactivate */
/* Data buffer type definitions --- NEED TO COORDINATE WITH Lo     */

  /* Data Buffer pointer(s) in the LCB  */
   /* All data contained in the LCB  */
  /* Buffer Descriptor pointer in the LCB  */

/* LSAP Table indicator flag definitions --- */
  /* Activate remote saps dynamically */

        /* no more read credits can accepted */
/* bad lcb set up bit */
  /* LCBI completion status */


/* LLC Layer Number              */
 /* LLC Layer number is equal to 2  */

/* The definition of 'instance numbers' */

/* The conditions under which messages age out are defined here */
  /* Alarm period */
 /* About 3 minutes */

   Maximum directory sizes   */
              /* 64 for time being   */
              /* until a better suggestion    */
              /* who knows when               */

/* States for local LSAPs */
 /* Active state */
 /* Inactiiive status */
 /* MAC error code */

/* ERRORS and success                                                  */

        /* success                        */

/* BAd function spec. function code */
/* Invalid function code  */
        /* Bad layer instance state */
/* Bad or unknown remote SAP  */
/* Bad or unknown local SAP  */
/* SAP already active      */
/* SAP deactivated       */
        /* Multiple activate requests on SAP  */
        /* Bad remote SAP state  */
        /* Bad local SAP state  */
        /* create local lsap error */
        /* create remote lsap error */
        /* SAP not available for services */
        /* Bad DSAP in received PDU  */
        /* Bad SSAP in received PDU  */
        /* Bad DMA transfer of data buffer  */
        /* Bad DMA transfer of buffer desc.  */
        /* Bad transmit request desc.  */
        /* Bad PDU  size  */
/* LCB buffer size too small for PDU  */
/* Bad DSAP state  */
/* PDU exceeds max PDU size  */
/* recieve transmit queue full */
/* invalid transfer */
/* return lcbi error */
/* bad lcbi response - big err */
        /* incrrect venue value */
/* data too long for data in lcb */
        /* DMA of a buffer failed */
/* bad register mail box directory */
/* Bad received PDU  */
/* invalid SM request */
        /* xid transaction error */
        /* xid respound error */
/* unknown message type */
/* bad io message */
/* invalid SM return mail box */
/* SM return mailbox full */
/* invalid DMA mail box */
/* DMA return mailbox full */
        /* invalid MAC mail box */
/* MAC return mailbox full */
/* genaric default error */
        /* getbuf could not allocate */
        /* allocate no mem avail */
        /* resolve mail box failed */
        /* registe mail box failed */
/* mail box turn on failed */
        /* mail box turn off failed */
/* mail box delete failed */
/* send massege  failed */
        /* llc layer event code */

        /* No match on a search address or name */
        /* for error but not exit return, to calling */
```

```
 /****************************************************************
*
*   This file (llc_struct.h) define the structures used in the
* LLC Layer processes for the LACS board.
*/
/* Define a structure for LLC frames */
/****************************************************************/

        /* Structure for supervisory commands */
        /* Structure for unnumbered commands */
        /* Structure for determining PDU type */
/* Define a structure used to enqueue llc_trans on a LSAP table */
/* Define a structure for LCBI memory blocks */
          /* Read LCB Information        */
          /* Write LCB Information                */
               /* Data Arrival event information  */
               /* Additional write credits event information */
               /* SAP Error event  */
/* Define a structure for a buffer descriptor image      */
/* Define a structure for remote LSAP tables */
                  /* NEED TO REORDER FOR STANDARDIZATION     */
                                  /*
                                          Bit 0 - Maintain Stats on all requests
                                          Bit 8 - Maintain Stats if Local match
                                          Bit15 - Dynamic RSAP
                                                                            */
   /* THE FOLLOWING STATS FOR REMOTES MUST BE REVISITED              */
/* Define a structure for a remote LSAP directory */
                  /* NEED TO REEXAMINE FOR CORRECTNESS    */
/* Define a structure for local LSAP tables */
                  /* NEED TO REORDER FOR STANDARDIZATION     */
                                  /*
                                          Bit 0 - Maintain Stats on this LSAP
                                          Bit 8 - Maintain Stats on Remotes
                                          Bit15 - Megabus Exposed LSAP
                                  /*
                                          Bit 0 - Report Data Arrival
                                          Bit 1 - Report Write Credits - N/A
                                          Bit 2 - LSAP Errors
                                  */
/* Define a structure for a local LSAP directory */
                  /* NEED TO REEXAMINE FOR CORRECTNESS    */
/* A structure for the LLC Instance Common Area */
   /* Mailbox IDs used by the LLC layer */
   /*  Local LSAP directory pointers */
   /*  Remote LSAP directory pointers */
   /* layer emergency event message ptr */
   /* Remote LSAP event counters   */
   /* Layer Instance Parameters */
/* *****************************************************************
 LLC layer transaction structure. The top of this structure is
 a union of several message types used in the processing of a transaction.
* Pointers to other memory blocks are carried along with the transaction
* to reduce the decoding and table lookup that would otherwise be required.
```

```
***********************************************************************
*/
/*--------------------------------------------------------------------*/
/* extern (lacs_cs1) LACS software crash error log reserved location */
/*--------------------------------------------------------------------*/
        /* llc layer 0 error message stack */
/* extern llc LAYER 1 reserved block */
        /* llc layer 1 error message stack */
/* extern llc LAYER 2 reserved block */
        /* llc layer 2 error message stack */
/* extern llc LAYER 3 reserved block */
        /* llc layer 3 error message stack */
```

```
/*
*  This file (llcincludes.h) is the master include file for LLC Layer
* processes. The header files used are listed below.
*/

/* Include all kernel header files */

/* Include all LACS message definitions */

/* Include all Station Management header files */

/* Include all megabus header files */

/* Include all mac header files */

/* Include all LLC Layer macros */

/* Include all LLC Layer constants */

/* Include all LLC Layer structures */
```

```
/*****************************************************************
*****************************************************************
***
***
***
***                          HONEYWELL
***
***                      Local Area Network
***
***          L O G I C A L   L I N K   C O N T R O L
***
***                        L A Y E R
***
***                   mac layer interface module
***
***
***     Copyright: Honeywell Information System
***                All rights reserved.
***
***     Cereate Date:      3/25/86
***     By:                M. Lu
***     Discription:
***
***     Functions & Parameters:
***
***
***     Revision #:
***     Date:
***     By:
***     Discription:
***
***     Revision #:
***     Date:
***     By:
***     Discription:
***
***
*******************************************************************
*******************************************************************
*/

/*******************************************************************/
/*
*       TITLE:              SEND ACTIVATE MESSAGE TO MAC LAYER PROCEDURE
*
*       FUNCTION:           All mailboxes are turned off until the MAC
*                           activate message is returned.  This is to prevent
*                           a number of MAC activate requests being sent due
*                           to multiple activates received.    A special MAC
*                           Activate mailbox is created to return the message
*                           frrom MAC.  The activate message is sent to the
*                           MAC layer manager and the LLC Activate message is
*                           sent back to the LLC mailbox.
*                            local LSAP.
*
*        INPUT:              Pointer to layer instance table
*
*        OUTPUT:
*
**********************************************************************
*/

   /* Create and Store
                 MAC registration mailbox to be used for a one time message*/
                 /* set limit to one message before full */

   /* Allocate some space for a MAC query message */

   /* DO CASE on the layer instance number of
       /* CASE Instance 0 */
         /* Get MAC mailbox ID for layer instance 0 */

       /* CASE Instance 1 */
         /* Get MAC mailbox ID for layer instance 1 */

       /* CASE Instance 2 */
         /* Get MAC mailbox ID for layer instance 2 */

       /* CASE Instance 3 */
         /* Get MAC mailbox ID for layer instance 3 */

       /* OTHERWISE */
         /* Unknown instance number, fatal error */
       /* ENDCASE */
    }   /* switch */

   /* Send message to MAC layer manager                                  */

   /* Declare the message type to be 'mac activate' */

   /* Set the priority and clear the buffer descriptors */

   /* Fill in mailbox for DATA.indicate messages */

   /* Clear the returning mailbox IDs */

   /* Declare the return mailbox */

   /* Send the MAC query message */

  /* Return             */
}       /* mac_activate */

/*************************************************************************/
/* ----------------------- LLC Layer Management ---------------------- */
/*
*        TITLE:              MAC REGISTRATION COMPLETE PROCEDURE
```

```
*
*        FUNCTION:          The MAC activate request message has returned
*                           completed to LLC. A successfully completed request
*                           contains the mailbox id for MAC data request mailbox
*                           which is saved in the layer instance table. All
*                           LLC mailboxes are turned and the mac registration
*                           mailbox is deleted.  The state of the layer instance
*                           is set to In-Use state. An unsuccesful completion
*                           of the MAC registration will cause the state of the
*                           layer instance to be set to ????(DISABLED??)
*
*        INPUT:             Pointer to message
*                           Pointer to LLC common layer instance table
*
*        OUTPUT:
*
*.......................LLC Layer Management............................*
* MAC_ACTIVATE_COMPLETED.request message
**********************************************************************
*/

     /* Turn on LLC LME mailboxes used during normal operations */

     /* Delete MAC activate message mailbox                    */

     /* IF the registration was a success */
     /* ELSE                                              */
     /* ENDIF */

     /* Release the message block to free memory */

   /* RETURN */
}        /* mac_act_comp */

/***********************************************************************/
/* ----------------------- LLC SAP Component -------------------- */
/* A MAC_DATA.indicate message */
/*
*        TITLE:             PROCESS A MAC DATA INDICATE PROCEDURE
*
*        FUNCTION:          A mac data indicate message is recieved with
*                           a buffer of data intended for an LSAP.  The
*                            type of PDU is determined and processed accordinly.
*                           For unumber information frames, the LLC header
*                           information is stripped off and delivered to the
*                           Network layer or across the MEgabus if a read LCBI
*                           is available.  If there is no LCBI available the message
*                           is queud as part of a transaction block until one is
*                           available.
*
*
*        INPUT:             Pointer to MAC data indicate message
*                           Pointer to Layer Instance common table
*
```

```
*       OUTPUT:         Error status
*
*************************************************************************
*/

    /* Point to the buffer */

    /* DO CASE on the frame type received                              */

      /* CASE of Unnumbered information frame          */
         /* Call procedure to validate the PDU request for this SAP     */
         /* If it is a valid PDU                                       */

             /* Then                                                        */

             /* Fetch the local LSAP table
                  ignore group addresses for now */

             /* increase UI_rx counter */

             /* Fetch the remote lsap table      */

             /* IF local LSAP is mapped to a LACS network layer    */

                 /* THEN                                                    */

                 /* Call procedure to strip off LLC header                  */

                 /* Create a llc_data_indicate message to be
                      delivered to network */

                 /* Send message to mailbox in LSAP table for Network   */

              }         /* if l_lsap_tbl->net_mbx NULL */

             /* ELSE IF Megabus interface and a read LCB is available          */

                  /* Dequeue the first  LCBI */
                 /* update the queue counter */

                 /* DO WHILE the LCBI does not have a buffer large enough
                     to hold the data or LCBI read queue is not empty  */

                     /* Set status in LCBI to indicate buff too small  */

                     /* Set actual size of buffer                  */

                     /* lcbi ! set the read credit count for
                        lcb user flow control */

                     /* Set LCBi completion bit                             */
                     /* Return LCBI to the L6                               */

                      /* Dequeue the next LCBI from read LCBI queue          */
```

```
                /* update the queue counter */

            }               /* while */

        /* IF LCB buffer(s) can contain all of PDU */

            /* THEN everything is okey dokey                       */

            /* Complet the LCBI status                             */

            /* Set the actual size */
            /* Set the remote logical address */
            /* lcbi_trans->lcbi_blk->lcbi_type.read_lcbi.cblra =
                  r_lsap_log_addr;*/
            /* Strip off LLC header from PDU

            /* Call procedure to complete remainder
            /* Fix    initial mac_inc FFFFF err. M. Lu. 3/86 */

            }       /* if cbtrg */

        /* ELSE last LCBI buffer too small and no
                available LCBS left on queue */

            /* Create LLC transaction block for PDU      */

            /* Allocate some space for a LLC transaction block */

            /* Save a pointer to the MAC indicate
                        message in the trans. blk. */

            /* Save the LSAP table pointers  and what else */
            /* save remote sap logical addr */
                        /* save local lsap table addr */
                        /* save remote lsap table addr */

           /* Clear the LCBI pointer since there is not one known yet */

            /* Strip off LLC header from PDU and set
                        buffer desc into trans.blk */

            /* Send LCBI tranaction block back to the L6          */

            /* Clear the data buffer in the transaction .blk */

            /* Set actual size of buffer                           */

            /* Set status in LCBI to indicate buff too small          */

            /* lcbi set the read credit count for lcb flow control */

            /* Set LCBi completion bit                             */
            /* Return LCBI to the L6    */
```

```
                    /* If data arrival event is to be
                         reported and an event LCBI is handy */

                    /* Then      */

                    /* Queue MAC indicate transaction block on
                         pdu waiting queue         */

                    /* Fetch and clear event LCBI pointer from
                                 LSAP table         */

                    /* Set buffer size and event status into LCBI          */

                    /* Return LCBI to th LS      */

                    /* Start ageing of message  */

                    /* ELSE IF data arrival event is to be
                        reported but there is not an event LCBI available */

                    /* Queue MAC indicate transaction
                         block onto event data arrival queue */

                    /* Start ageing of message  */

                    /* ELSE                         */
                    /* check bound limit */
                        /* over the bound limit, drop the request */

                        /* update the pdu packet dropped */
                        /* dump the trans blk */
                      }  /* if bound li */
                        /* update the queue counter */
                        /* Queue MAC indicate trans block on to
                                pdu waiting queue    */
                      }  /* else */

                        /* Clear the aging counter   ?????????    */

                        /* Start ageing of message      */

                        /* ENDIF         */

                }             /* if_else cotrg */
               /* ENDIF         */

            }           /* if_else_if */

             /* ELSE must be Megabus interface but no outstanding read LCB */

               /* Create LLC transaction block for PDU                  */

               /* Allocate some space for a LLC transaction block */
```

```
                /* Save a pointer to the MAC indicate
                        message in the trans. blk. */

                /* Save the LSAP table pointers  and what else */
                /* save remote sap logical addr */
                        /* save local lsap table addr */
                        /* save remote lsap table addr */

                /* Clear the LCBI pointer since there is not one known yet */

                /* Strip off LLC header from PDU
                        and set buffer desc into trans.blk */

                /* IF data arrival event is to be
                        reported and an event LCBI is handy */

                /* Then                    */

                /* Queue MAC indicate
                        transaction block on pdu waiting queue        */

                /* Fetch and clear event
                        LCBI pointer from LSAP table        */

                /* Set buffer size and event status into LCBI */

                /* Return LCBI to th Lo                */

                /* Start ageing of message     */

                /* ELSE IF data arrival event is to be
                     reported but there is not an event LCBI available */

                /* Queue MAC indicate transaction
                         block onto event data arrival queue */

                /* Start ageing of message     */

                /* ELSE        */
                /* check bound limit */
                    /* over the bound limit, drop the request */

                    /* update the pdu packet dropped */
                    /* dump the trans blk */
                 }      /* if */
                    /* update the queue counter */
                    /* Queue MAC indicate trans block
                       on to pdu waiting queue   */
                 }      /* else */
               /* Clear the aging counter  ?????????    */

               /* Start ageing of message      */

               /* ENDIF                        */
```

```
                /* ENDIF */

           /* ELSE an error must have occurred                                 */

              /* WHAT ABOUT STATS                                              */

              /* Drop the incoming pdu, the LSAP is not defined */

              /* Free the MAC message */
          /* ENDIIF */

          /* CASE of an XID PDU P bit clear case  */
          /* CASE of an XID PDU P bit set case    */

      /* CASE of a TEST pdu P bit clear case      */
      /* CASE of a TEST pdu P bit set case        */

      /* OTHERWISE itmust be a very bad PDU                                    */

  /* ENDCASE                                                           */
  }               /* switch */

 }                /* mac_ind */
```

```
/*****************************************************************
*****************************************************************
***
***
***
***                              HONEYWELL
***
***                         Local Area Network
***
***             L O G I C A L   L I N K   C O N T R O L
***
***                            L A Y E R
***
***                      transport - network layer
***                          interface module
***
***
***      Copyright: Honeywell Information System
***                 All rights reserved.
***
***      Cereate Date:      3/25/86
***      By:                M. Lu
***      Discription:
***
***      Functions & Parameters:
***
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***
*******************************************************************
*******************************************************************
*/

/**********************************************************************/
/*
*       TITLE:            TRANSPORT LAYER INTERFACE ACTIVATE LSAP PROCEDURE
*
*       FUNCTION:         Receives an activate LSAP request message , from tran
*                         local lsap request cause a search of the local LSAP
*                         directory to insure the LSAP exists and the LSAP
*                         has not been previously activated or in a state
*                         prohibiting service to the user.  Remote lsap requests
*                         are also checked to insure the LSAP does exist.  In
*                         addition, a pointer to the remote LSAP table is placed
*                             in the activated LSAP directory of the local LSAP.
*
*       INPUT:                Pointer to mailbox message
*                             Pointer to layer instance table
*
*       OUTPUT:
**********************************************************************
*/

    /* LLC_TRANS *activate_lsap; * pointer to create local lsap message */
    /*
      *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
      *******  AN ACTIVATE REQUEST FROM A NETWORK LAYER *******
    */
    /* Cast on activate transaction block message              */
    /* activate_lsap = (LLC_TRANS *)mbxptr; */
    /* IF layer instance state indicates llc not registered with
             MAC (state=NULL) */
       /* THEN                                                         */
          /* Fetch pointer to lsap name in ACTV_MSG                 */
          /* Call search procedure to determine if LSAP exists            */
          /* IF LSAP exists in LSAP directory and LSAP state is Nulled     */
             /* THEN                     */
                /* Call mac_activate procedure to activate MAC layer         */
                /* Re-issue activate message to the llc_mailbox     */
         /* ELSE LSAP doesn't exist                   */
                /* Set status of LCBI to indicate bad LSAP */
                /* Set completion word in the LCBI     */
                /* Request DMA services to transfer LCB   */
                /*   This is one of the difficult errors to handle       */
            /* ENDIF                         */
            /* ELSEIF layer instance is in the INUSE state                  */
            /* DO CASE on venue to determine
                whether a remote or local LSAP activate */
               /* CASE of Local venue */
                /* case ACTV_LOC: */
                 /* Call procedure to Activate a Local LSAP for this request  */
                 /* Set status of LCBI to indicate operation status */
                /* CASE of Remote LSAP */
                /* case ACTV_RMT: */
                 /* Call procedure to Activate a
                       Remote LSAP for this request  */
                 /* Set status of LCBI to indicate operation status */
                /* OTHERWISE unknown venue */
                 /* Can this really happen and not
                       be previously detected ?       */
                 /* Set status of LCBI to
                       indicate bad function specific function code */
            /* ENDCASE           */
          /* Set completion word in the LCBI                              */
          /* activate_lsap->lcbi_blk->cbcbs = LCBI_COMPLETED; */
          /* Request DMA services to transfer LCB                       */
          /*             This is one of the difficult errors to handle      */
         /* llcerr_code = return_lcbi(activate_lsap,lica_p->lme_mbx,lica_p); */
```

```
        /* ELSE  layer instance not able to support request              */
          /* Set status of LCBI to indicate layer unable to support request   */
          /* Set completion word in the LCBI                                  */
          /* activate_lsap->lcbi_blk->cbcbs = LCBI_COMPLETED | LCB_ERR; */
          /* Request DMA services to transfer LCB                          */
          /*              This is one of the difficult errors to handle      */
          /* llcerr_code = return_lcbi(activate_lsap,lica_p->lme_mbx,lica_p); */
        /* ENDIF                                                           */
        /* Return                                                          */
}       /* llc_trn_ac */

/**********************************************************************/
/*
*
*       TITLE:          ACTIVATE LOCAL LSAP PROCEDURE
*
*       FUNCTION:       Receives an activate LSAP request message for a
*                       local LSAP.
*                       The request causes a search of the local LSAP
*                       directory to insure the LSAP exists and the LSAP
*                       has not been previously activated or in a state
*                       prohibiting service to the user.
*
*       INPUT:          Pointer to mailbox message
*                       Pointer to layer instance table
*
*       OUTPUT:
*********************************************************************
*/

      /* Fetch pointer to lsap name in LCBI                              */
      /* Call search procedure to determine if LSAP exists              */
      /* IF LSAP exists in LSAP directory and LSAP state is Nulled      */
      /* THEN                                                            */
         /* Fetch a pointer to the LSAP table                             */
        /* DO CASE on the major state of the
            /* CASE OF THE LSAP state is uininitialized (
              /* SET LSAP TO A MEGABUS INTERFACE ***
                NEED TO REVISIT FOR NETWORK INTERFACE */
              /* Set LSAP state to IN-USE and Operational */
              /* Set logical address into return message or LCB */
              /* Set max number of bytes in PDU */
              /* Set read and write credit values into LCB */
              /* Set status to indicate successful operation */
            /* CASE of LSAP state is already IN-USE */
              /* Set logical address into return message or LCB */
              /* Set max number of bytes in PDU            */
              /* Set read and write credit values into LCB      */
              /* Set status to indicate SAP is already activated */
            /* CASE of LSAP is not available for service (LOCKED,DOWN,etc.) */
              /* Set status to indicate SAP is unavailable */
          /* ENDCASE                                                       */
    /* ELSE LSAP is non existent                                     */
      /* Set status to indicate SAP is bad                               */
    /* ENDIF
   /* RETURN                                    */

/*************************************************************************/
/*
*       TITLE:          ACTIVATE REMOTE LSAP PROCEDURE
*
*       FUNCTION:       Receives an activate LSAP request message ,
*                       Activate remote lsap requests are made with respect
*                       to a local LSAP.  The layer instances local LSAP
*                       directory is checked to insure the local LSAP
*                       exists. If it exists then a check is made on the
*                       local LSAP activated remote directory to insure that
*                       the remote has not been already activated.  The
*                       state of the remote LSAP is set to Activated if not
*                       already and a count of the number of activates
*                       received is incremented.  Finally a pointer
*                       to the remote LSAP table is placed
*                       in the activated LSAP directory of the local LSAP.
*
*       INPUT:          Pointer to mailbox message
*                       Pointer to layer instance table
*
*       OUTPUT:
************************************************************************
*/
   /* if local lsap address is within dir rang */
     /* then there is a valid ligical address for the local lsap */
       /* Fetch a pointer to the local LSAP table                          */
       /* IF LSAP exists in LSAP directory and LSAP state is in INUSE state */
         /* THEN                                                   */
           /* Fetch pointer to lsap name in LCBI                           */
           /* Call search procedure to determine if
                    LSAP already activated for SAP */
           /* IF Remote LSAP is not activated already for this SAP */
             /* THEN */
               /* Search remote SAP directory for remote LSAP name    */
               /* IF lsap not defined in directory                    */
                 /* THEN there is an error                      */
                   /* Set status to indicate remote
                      LSAP are ready activated for this local */
                 /* ELSE Activate remote SAP for this local */
                   /* Set status to indicate successful LCB             */
                   /* Fetch a pointer to the remote LSAP table          */
                   /* Call procedure to fetch the next available
                      remote LSAP entry        */
                   /* Set pointer to remote SAP table into activate
                      of local LSAP table */
                   /* Increment count of number of activates
                      existing for this remote LSAP */
                   /* Set logical address
                      into LCBI for the Remote LSAP           */
                   /* over write the logical addr long word */
```

```
                /* ENDIF */
            /* ELSE  Remote LSAP already activated for this SAP */
                /* Set logical address into LCSI for the Remote LSAP */
                /* Set status to indicate remote LSAP are
                         ready activated for this local */
             /* ENDIF */
         /* ELSE  Local LSAP not
                         available for service or nonexistent */
            /* Set status to indicate
                         local LSAP is unavailable or non-existent */
           /* ENDIF                 */
           /* end first if */
        /* ELSE  Local LSAP not
                         available for service or nonexistent          */
         /* Set status to indicate local
                         LSAP is unavailable or non-existent */
     /* Return             */
```

```
/******************************************************************
*******************************************************************
***
***
***
***                              HONEYWELL
***
***                          Local Area Network
***
***            L O G I C A L   L I N K   C O N T R O L
***
***                             L A Y E R
***
***                        megabus interface module
***
***
***      Copyright: Honeywell Information System
***                 All rights reserved.
***
***      Cereate Date:        3/25/86
***      By:                  M. Lu
***      Discription:
***
         Functions & Parameters:

***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***
**********************************************************************
**********************************************************************
*/

/**********************************************************************/
/*
*        TITLE:              ACTIVATE LSAP PROCEDURE
*
*        FUNCTION:           Receives an activate LSAP request message ,
*                            local lsap request cause a search of the local LSAP
*                            directory to insure the LSAP exists and the LSAP
*                            has not been previously activated or in a state
                             prohibiting service to the user.  Remote lsap requests
                             are also checked to insure the LSAP does exist.  In
*                            addition, a pointer to the remote LSAP table is placed
*                            in the activated LSAP directory of the local LSAP.
```

```
*
*       INPUT:              Pointer to mailbox message
*                           Pointer to layer instance table
*
*       OUTPUT:
*********************************************************************
*/

    /*
     *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
     *******  AN ACTIVATE REQUEST FROM A NETWORK LAYER *******
    */

   /* Cast on activate transaction block message                */

   /* IF layer instance state indicates llc not
                registered with MAC (state=NULL) */
      /* THEN                                                            */

       /* Fetch pointer to lsap name in LCBI                              */

       /* Call search procedure to determine if LSAP exists              */

       /* IF LSAP exists in LSAP directory and LSAP state is Nulled      */
         /* THEN                             */
           /* Call mac_activate procedure to activate MAC layer             */

           /* Re-issue activate message to the llc_mailbox                  */
       /* ELSE LSAP doesn't exist                         */
           /* Set status of LCBI to indicate bad LSAP */

           /* Set completion word in the LCBI         */

           /* Request DMA services to transfer LCB                          */
           /*              This is one of the difficult errors to handle      */
       /* ENDIF                              */
    }   /* if */
   /* ELSEIF layer instance is in the INUSE state                  */
       /* DO CASE on venue to determine whether a
                        remote or local LSAP activate */
           /* CASE of Local venue                                           */
             /* Call procedure to Activate a Local LSAP for this request  */
                 /* Set completion word in the LCBI       */

                 /* Set status of LCBI to indicate
                         layer unable to support request   */
                 /* Request DMA services to transfer LCB        */
                 /* This is one of the difficult errors to handle */
             /* Set status of LCBI to indicate operation status     */

           /* CASE of Remote LSAP                                           */
             /* Call procedure to Activate a Remote LSAP for this request  */

             /* Set status of LCBI to indicate operation status */
```

```
            /* OTHERWISE unknown venue                                               */
              /* Can this really happen and not be
                                   previously detected ?       */
              /* Set status of LCBI to indicate bad
                                   function specific function code */
            /* ENDCASE                                                               */

       /* Set completion word in the LCBI                                          */

       /* Request DMA services to transfer LCB                                    */
       /*            This is one of the difficult errors to handle                  */
     /* ELSE  layer instance not able to support request                        */
        /* Set status of LCBI to indicate layer unable to support request          */

        /* Set completion word in the LCBI                                          */

        /* Request DMA services to transfer LCB                                    */
        /*            This is one of the difficult errors to handle                 */

     /* ENDIF                                                                     */

     /* Return                                                                   */
}      /* llc_lme_ac */

/****************************************************************************************/
/*
*
*       TITLE:          ACTIVATE LOCAL LSAP PROCEDURE
*
*       FUNCTION:       Receives an activate LSAP request message for a
*                       local LSAP.
*                       The request causes a search of the local LSAP
*                       directory to insure the LSAP exists and the LSAP
*                       has not been previously activated or in a state
*                       prohibiting service to the user.
*
*       INPUT:          Pointer to mailbox message
*                       Pointer to layer instance table
*
*       OUTPUT:
******************************************************************************
*/

    /* Fetch pointer to lsap name in LCBI                                        */

    /* Call search procedure to determine if LSAP exists                         */

    /* IF LSAP exists in LSAP directory and LSAP state is Nulled                 */
      /* THEN                                                                      */
        /* Fetch a pointer to the LSAP table                                        */

        /* DO CASE on the major state of the LSAP to determine
                          actions required  */
```

```
          /* CASE OF THE LSAP state is uninitialized
                      (LSAP major state is null)       */
            /* SET LSAP TO A MEGABUS INTERFACE ***
               NEED TO REVISIT FOR NETWORK INTERFACE */

            /* Set LSAP state to IN-USE and OPerational                        */

            /* Set logical address into return message or LCB    */

            /* Set max number of bytes in PDU                  */

            /* Set read and write credit values into LCB       */

            /* Set status to indicate successful operation                     */

          /* CASE of LSAP state is already IN-USE */
            /* Set logical address into return message or LCB     */

            /* Set max number of bytes in PDU                     */

            /* Set read and write credit values into LCB          */
            /* Set status to indicate SAP is already activated                 */

          /* CASE of LSAP is not available for service (LOCKED,DOWN,etc.) */
            /* Set status to indicate SAP is unavailable                       */
         /* ENDCASE                                                        */
     /* ELSE LSAP is non existent                                */
       /* Set status to indicate SAP is bad                            */
     /* ENDIF

    /* RETURN                            */
}        /* act_l_lsap */

/***********************************************************************/
/*
*       TITLE:             ACTIVATE REMOTE LSAP PROCEDURE
*
*       FUNCTION:          Receives an activate LSAP request message ,
*                          Activate remote lsap requests are made with respect
*                          to a local LSAP.  The layer instances local LSAP
*                          directory is checked to insure the local LSAP
*                          exists. If it exists then a check is made on the
*                          local LSAP activated remote directory to insure that
*                          the remote has not been already activated.  The
*                          state of the remote LSAP is set to Activated if not
*                          already and a count of the number of activates
*                          received is incremented.  Finally a pointer
*                          to the remote LSAP table is placed
*                          in the activated LSAP directory of the local LSAP.
*
*       INPUT:             Pointer to mailbox message
*                          Pointer to layer instance table
*
*       OUTPUT:
```

```
*
*************************************************************************
*/

   /* if local lsap address is within cir rang */
     /* then there is a valid ligical address for the local lsap */
       /* Fetch a pointer to the local LSAP table                         */

       /* IF LSAP exists in LSAP directory and LSAP state is in INUSE state */
         /* THEN                                                  */
           /* Fetch pointer to lsap name in LCBI                            */
           /* Call search procedure to determine if
                LSAP already activated for SAP */
                /* Search remote SAP directory for remote LSAP name   */
                /* If lsap not defined in directory                   */
                  /* THEN there is an error                  */
                      /* Set status to indicate remote
                /* ELSE Activate remote SAP for this local */
                     /* Set status to indicate successful LCB          */

                     /* Fetch a pointer to the remote LSAP table */
                     /* Call procedure to fetch the next
                      /* Set pointer to remote SAP
                      /* Increment count of number of

                      /* Set logical address
                        into LCBI for the Remote LSAP          */
               /* ELSE  Remote LSAP already activated for this SAP */

                 /* Set logical address into LCBI for the Remote LSAP  */

                 /* Set status to indicate remote
                        LSAP are ready activated for this local */
                 /* ENDIF */
           }     /* if */
          /* ELSE  Local LSAP not available for service or nonexistent */
             /* Set status to indicate local
                LSAP is unavailable or non-existent */

          /* ENDIF   */
          /* end first if */
          /* ELSE  Local LSAP not available for service or nonexistent */

          /* Set status to indicate local LSAP is unavailable or non-existent */
        /* Return                                                         */
}       /* act_r_lsap */

/************************************************************************/
/* --------------------------- Megabus Interface ---------------------- */
/* The request to DMA a LCBI from the L6 to the LACS has returned */
/*
*       TITLE:          SEND DATA BUFFER TO L6 PROCEDURE
*
*       FUNCTION:       This routine requests the DMA services to transfer
*                           a data buffer to L6 memory. Data can be transfered in
*                           3 ways: in the LCB itself, to buffer described by
*                           the LCB, or into buffers specifed by a buffer descriptor
*                           pointed to by the LCB.
*
*       INPUT:              Pointer to MAC data indicate message
*                           Pointer to the LCBI transaction block
*                           pointer to the layer instance table
*
*       OUTPUT:
*
**********************************************************************
*/

    /* ***************************************************************
     *   This message has been returned from DMA services. The following
     * code will check the results of the transfer request and if it was
     * successful.
     ****************************************************************
    */

        /* DO CASE on the type of data buffer in LCBI */

           /* CASE of data buffer contained in the LCB          */
                   /*data in lcb, data buffer transfer */

                   /* for call rcv_buf_to_l6 no poolem set */

                   /* set the read credit count for lcb user flow control */

                   /* set remote logical address */

                   /* Set the LCBI completion bit                          */
                   /* get correct buffer and return */

            /* CASE of data buffer pointer(s) contained in the LCBI */
                   /* Request the buffer be transferred to the L6 */

                   /* Have the message return to the DMA done mailbox */

                   /* Set priority to normal */

                   /* Set the number of L6 buffers        */

                   /* Set a residual range pointer to point to
                           buffer 0 residual range */

                   /* Initiailize total range to pdu minus llc header         */

                   /* DO FOR all buffers in the LCBI                          */
                           /* Set the L6 buffer descriptors */

                           /* Set the L6 buffer range */
```

```
                /* ******************************************************
                *  There is a problem if the total range exceeds 32K words
                *                     fortunately LLC will always be less BUT
                *                     DO WE NEED TO VALIDATE RANGE and
                *                     WHAT ABOUT LCB FORMAT ? (32 or 16 bits)
                *                     and WHAT ABOUT LCB FORMAT - short now
                ********************************************************
                */
                        /* Set the residual range and update total range */
                        /* IF the total range is equal to zero          */

                              /* THEN set residual range to
                                      equal the whole buffer  */
                              /* *rsr_ptr */

                        /* ELSE IF total range is less than the buffer    */
                               /* THEN set residual range equal
                                      to buffer minus the total range */
                               /* Set the total range to zero*/

                       /* ELSE residual range is zero  */
                               /* *rsr_ptr = 0;* set residual range to zero*/

                               /* Set the total range minus the buffer size */
                       /* ENDIF                 */

                       /* Decrement the residual range
                             pointer to point to the next */

                       /* ENDDO        */

     /* Set buffer from llc_ttrans block  into dma message  */
     /* Clear channel and interrupt level  */

     /* Clear all remaining fields */
     /* Send the request message to DMA services */

       /* special routine for llc_trans release */

     }      /* else if full */

  /* CASE of a buffer descriptor pointer contained in the LCBI */
     /* Set invalid status in the LCBI */
     /* set the read credit count for lcb user flow control */
     /* Set the LCBI completion bit                          */
     /* Return the LCBI to L6 memory */

    /* OTHERWISE */

    /*************************************************************
     * Can not decode the buffer type into anything
     * that can be processed by this function. Update the LCBI
     * status to reflect the problem and return the LCBI to
     * the L6.
```

```
          ***************************************************************
         */

          /* Set invalid status in the LCBI */

          /* set the read credit count for lcb user flow control */

          /* Set the LCBI completion bit                              */
          /* Return the LCBI to L5 memory */

      /* ENDCASE */
  }      /* switch */

}        /* send_buf_to_l5 */
```

```
/*******************************************************************
********************************************************************
***
***
***
***                                   HONEYWELL
***
***                              Local Area Network
***
***             L O G I C A L   L I N K   C O N T R O L
***
***                                 L A Y E R
***
***                               debug module
***
***
***      Copyright: Honeywell Information System
***                 All rights reserved.
***
***      Cereate Date:      1/25/86
***      By:                M. Lu
***      Discription:
***
***      Functions & Parameters:
***
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***
***********************************************************************
***********************************************************************
*/

/* *****************************************************************
**
**
**      This module is using for debugging software break point
**      set up and othe tool for debugging. the entire module
**      and the related debugging statements in the llc layer
**      codes should all take out.
**
**      I believe this is the more efficient way to develop
**      software. I t will be proved.
**
**      Autho: M. Lu
```

```
**
**
**
*****************************************************************************
*/

/*******************************************************
**
**      Software (C) break points rounte
**
**
*******************************************************
*/

/*******************************************************
**
**      print LBC rounte
**
**
*******************************************************
*/

/*******************************************************
**
**
**      print PCB rounte
**
**
*******************************************************
*/

/*******************************************************
**
**
**      print ECB rounte
**
**
*******************************************************
*/

/*******************************************************
**
**
**      print message rounte
**
**
*******************************************************
*/

/*******************************************************
**
**
**      print status rounte
**
```

```
**
*********************************************************
*/

/********************************************************
**
**
**      print startup parameters rounte
**
**
*********************************************************
*/
```

```
/*****************************************************************
*****************************************************************
***
***
***                              HONEYWELL
***
***
***                         Local Area Network
***
***            L O G I C A L   L I N K   C O N T R O L
***
***                            L A Y E R
***
***                          common module
***
***
***     Copyright: Honeywell Information System
***                All rights reserved.
***
***     Cereate Date:      1/25/86
***     By:                D. Oshaughn
***     Discription:
***
***     Functions & Parameters:
***
***
***     Revision #:    1
***     Date:          3/25/86
***     By:            M. Lu
***     Discription:
***
***     Revision #:
***     Date:
***     By:
***     Discription:
***
***
*******************************************************************
*******************************************************************
*/

/* ------------------------ LLC SAP Component -------------------- */
/* *******************************************************************
* A procedure to return a 'not active' status in the LCBI that has
* just been transfered and the LSAP was not active.
*
* The parameter is a pointer to the llc transaction.
*
* FUNCTIONS list:
*
*         lsap_not_active(llc_trans)
*            LLC_TRANS *llc_trans;  * A pointer to the llc transaction *
*         IOLDs(mbxptr,ret_mbx,lica_p)
*              MSG *mbxptr;
*              MBID ret_mbx;  * The return mailbox ID *
*              LICA *lica_p;  * Pointer to the process variables *
*         return_lcbi(llc_trans,ret_mbx,lica_p)
*              LLC_TRANS *llc_trans;
*              MBID ret_mbx;  * The return mailbox ID *
*              LICA *lica_p;
*         return_buff(llc_trans,lica_p)
*              LLC_TRANS *llc_trans;
*              LICA *lica_p;
*         lcbi_to_l6(mbxptr,lica_p)
*              MSG *mbxptr;
*              LICA *lica_p;  * Pointer to the process variables *
*         buf_to_lac(mbxptr,lica_p)
*              MSG *mbxptr;
*              LICA *lica_p;
*         buf_to_l6(mbxptr,lica_p)
*              MSG *mbxptr;
*              LICA *lica_p;
*         short llc_alarm(mem_blk,msg_type,alrm_per)
*              AMSG *mem_blk;  * Pointer to the memory block *
*              ushort msg_type; * LLC message type  *
*              u_long alrm_per;  * Alarm period *
*
*************************************************************************


  /* A pointer to the llc transaction */
     /* Kernel return codes */
    /* Set the status for LSAP not active */
        /* lsap_not_active */


/* -------------------------- Megabus Interface ------------------------ */
/* ******************************************************************** */
/* An IOLD request message */
/* This message is from the megabus interface software and states */
/*
*  TITLE:     RECEIVE IOLD MESSAGE and REQUEST LCBI FORM L6 PROCEDURE
*
*  FUNCTION:
*        An IOLD command was issued by the L6 for this board and instance.
*   All function specific function codes for the LLC layer
*   and instance are all routed here at sign-in time. The object of this
*   code is to find space for the LCBI and request DMA services to make
*   the transfer across the megabus. The message will be returned when
*   DMA services completes the transfer or finds a reason why the step
*   cannot be performed.
*     A LLC transation block is used from this point on to process requests.
*   The top end of the block is a union of message types used to DMA the
*   LCBI, to DMA any buffers involved, and to request L_DATA.request. The
*   instance, the LSAP table, and the LCBI all have pointers to them carried
*   along with the transaction block.
*
```

```
*        Input:     Pointer to IOLD message
*                   Return mailbox id
*                   Access to common system management data structures
*
*        Output:    Error Status
*
*************************************************************************
*/

  /* The return mailbox ID */
  /* Pointer to the process variables */
      /* Pointer to the LLC transaction block */
      /* Kernel error codes */

         /* clear block pointer */
         /* clear block counter */

    /* Get some space for a LLC transaction */

                          /* return to the calling */
                          /* if ERR_RETURN */
    /* initial all null */

    /* Set the size of the transfer per the IOLD request */

    /* Save the length for the return trip later */

    /* Save the channel bits for the return trip */

    /* Get and save the pointer to a memory block
     the size requested plus queue pointers */

    /* IF no space is available */
                                  /* return to the calling */
                          /* if ERR_RETURN */

    /* ENDIF */
          /* clean up all mem llc_trans->lcbi_blk */

    /* Point to the LCBI block */

    /* Have the message return to the supplied mailbox */

    /* Set request type to 'move LCB to LACS' */

    /* Set the L6 address from the IOLD request */

    /* Save the L6 address for the return trip later */

    /* Copy channel and level information */

    /* Set priority to normal */

    /* Clear all remaining fields */
    /* Send the request message to DMA */

                           /* return to the calling */
                           /* if ERR_RETURN */

    /* Free the IOLD request message block */
    /* An ERROR CODE SHOULD BE PART OF THE RETURN      */

    /* RETurn                                           */
        /* IOLDs */

/*****************************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/* A common function to return LCBI blocks across the megabus */
/*
*        TITLE:    RETURN LCBI TO L6 PROCEDURE
*
*       FUNCTION: This function receives a transaction block containing a
*       a LCBI.  It creates amessage to for the DMA software to return it
*       to L6 memory .  The LCBI status must have been set by the caller
*
*       Input:    Pointer to transaction block
*                 Return mailbox id
                  Access to common system management data structures

*       Output:   Error Status
*
*************************************************************************
*/

  /* The return mailbox ID */
     /* Kernel error return code */
    /* Convert the message into a DMA LCBI request */

    /* Have the message return to the DMA done mailbox */

    /* Set priority to normal */

    /* Point to the LCBI to be transfered */

    /* Declare the length to be transferred */

    /* Move the L6 memory address to the DMA request */

    /* Copy CPU, channel, and level from LCBI */

    /* Clear all remaining fields */

    /* set error detection bit for level 6 software */

    /* Send the request message to DMA services */
```

```
                         /* return to the calling */
                         /* if ERR_RETURN */

    /* Return                                       */

        /* return_lcbi */

/****************************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/*
* A common procedure to return message buffers to the L6 memory.
* The parameters passed are: the LLC transaction, the indication message,
* and the variables for LLC.
***************************************************************************
*/

     /* Kernel return codes */

    /* Convert the llc transaction into a buffer DMA request */

    /* Have the message return to the DMA done mailbox */

    /* Set priority to normal */

    /* Set the buffer count to 1.. Note: There may be more in the future. */

    /* Set the L6 buffer descriptors */

    /* Set the L6 buffer range */

    /* Copy CPU, channel, and level from LCBI */
    /* Clear all remaining fields */

    /* Send the request message to DMA services */

                         /* return to the calling */
                         /* if ERR_RETURN */

        /* return_buf */

/****************************************************************************/
/* --------------------------- Megabus Interface ------------------------ */
/* The request to DMA a LBCI to the L6 Has returned */
/*
*       TITLE:           LCBI TRANSFER TO L6 COMPLETED PROCEDURE
*
*       FUNCTION:        A receive , xmit or lme
*                        LCB data request has been completed and
*                        the L6 memory LCB has been updated by the LACS DMA.
*                        This routine releases the LACS memory for the LCBI,
*                        for the transaction block, and the buffer descriptor
*                        block (if there is one), for the data buffer(if there
*                         is one), and a mac data indicate message(if there is
*                         one).
```

```
*
*       INPUT:          Pointer to LLC transaction block
*                       Pointer to layer instance table
*
*       OUTPUT:         Error Code
*
**************************************************************************
*/

   /* Pointer to the process variables */
     /* *****************************************************************
      *  This message is being returned by DMA services after LLC has made
      * a request to transfer a LCB back to the L6. The following code will
      * check the results of the request and free the message if all went well.
      *****************************************************************
     */

      /* Pointer to the LLC transaction block */

     /* IF the transfer was successful */
     /* ELSE */
         /* Decide what to do with DMA failures at this point..*/
    /* ENDIF */
         /* lcbi_to_l6 */

  ****************************************************************************/
/*
*       TITLE:          CLEAN LLC_TRANS BLOCK FUNCTION
*
*       FUNCTION:       A receive , xmit or lme
*                       This routine releases the LACS memory for the LCBI,
*                       for the transaction block, and the buffer descriptor
*                       block (if there is one), for the data buffer(if there
*                        is one), and a mac data indicate message(if there is
*                        one).
*
*       INPUT:          Pointer to LLC transaction block
*
*       OUTPUT:         Error Code
*
**************************************************************************
*/


   /* Pointer to the trans block */

/*
      * IF the transaction block points to service block not NULL *
           * THEN release the serv  *
          * ENDIF                                                     *

      /* IF the transaction block points to mac data indicate message      */
           /* THEN release the mac indicate message                     */
         /* ENDIF                                                     */
     /* IF the transaction block points to data buffer           */
          /* THEN release the data buffer                    */
     /* ENDIF                                                   */

     /* If the LCBI points to lcbi */
          /* THEN release the lcbi */
     /* ENDIF                                                   */

     /* IF the LCBI points to buffer descriptor              */
          /* THEN release the buffer descriptor                    */
       /* ENDIF                                                   */
     /* Release the LLC transaction block */
       /* trans_clr */

/***************************************************************************/
/* -------------------------- Megabus Interface ----------------------- */
/* The request to DMA a buffer from the L6 has returned */
/***************************************************************************/

      /* Pointer to the l_pdu */
      /* Array index */
    /* ****************************************************************
     * This message is returned by DMA services when it completes the
     * transfer of a buffer from the L6 to the LACS board as part of a
     * transmit transaction. This step will add the LLC layer addressing
     * to the message and send the lpdu to MAC for transmission.
     ****************************************************************
    */

      /* Pointer to the LLC transaction block */
      /* Kernel return codes */

    /* IF the transfer was a success */
        /* Change the transaction into a L_DATA.request */

        /* Send to the LLC layer */

    /* ELSE */
       /* Report a DMA failure */

       /* Set the LCB status to completed */

       /* Return the LCBI to L6 memory */
    /* ENDIF */
        /* buf_to_lac */

/***************************************************************************/
/* -------------------------- Megabus Interface ----------------------- */
/* The request to DMA a buffer to the L6 has returned */
/***************************************************************************/

    /* ********************************************************************
     * This message is returned by DMA services when it completes the
```

```
      * transfer of a buffer from the LACS board to the L6 as part of a read
      * transaction. This step will set the status in the LCBI and request DMA
      * services to return the LCB to the L6.
      ***********************************************************************
     */

      /* Pointer to the LLC transaction block */
      /* Kernel return code */

     /* Indicate LCBI completion in the status */

     /* Return the LCBI to L6 memory */
}        /* buf_to_l6 */

/* End of LLC procedures */

/*************************************************************************/
/* ------------------------- LLC Layer Management --------------------- */
/*
*  This function will request an alarm from the kernel via a
* message. The caller must supply a block of memory the size of
* the alarm message (AMSG), the message type for the returning
* message, and the period to wait before the alarm message is
* returned via the process default mailbox.
 **********************************************************************


 /* Pointer to the memory block */
 /* LLC message type  */ -
/* Alarm period */

     /* Set the priority and clear the buffer descriptors */

     /* Set the message type to aging alarm */

     /* Set length of timeout interval in seconds */

     /* Request the next aging alarm period */

         /* llc_alarm */
```

```
/*****************************************************************
*****************************************************************
***
***
***
***                             HONEYWELL
***
***                         Local Area Network
***
***               L O G I C A L   L I N K   C O N T R O L
***
***                             L A Y E R
***
***                        event message module
***
***
***       Copyright: Honeywell Information System
***                  All rights reserved.
***
***       Cereate Date:       3/5/86
***       By:                 M. Lu
***       Discriotion:
***
***       Functions & Parameters:
***
***
***       Revision #:
***       Date:
***       By:
***       Discriotion:
***
***       Revision #:
***       Date:
***       By:
***       Discription:
***
***
********************************************************************
********************************************************************
*/

/***********************************************************************/
/* ------------------------ LLC Layer Management -------------------- */
/* A LLC Layer management event message */
/*
*       FUNCTION
*               Handling all error and crash situations.
*               It will eventually grow up to a large complicate
*               module.
*               The event message recieved by the llc layer management
*               module is of the format*
*                   __ layer info
*                   __ layer inst
*                          __ layer inter. selector
*                          __ event status
*
*
*      INPUT:   pointer to event message mbox.
*               pointer to layer instance table.
*
*      OUTPUT:
*
*/
/***************************************************************************/

        /* all function moved to common event_msg routine */
                /* ptr to event msg */
        /* iteration index */
        /* return code */
                        /* clear block pointer */
        /* block size */
        /* clear block counter */

          /* allocate space for event msg*/
                                /* return to the calling */
                         /* if ERR_RETURN */

                         /* if */
                                /* clean up all mem msgptr */

                   /* set up event msg */

                   /* load layer internal selector params from llc_trans
                         into event msg */

                         /* for */
                         /* for */

                   /* set up event status */

                   /* OR error class with specificevent code to get event code */

                   /* send event msg to SM -- use event mailbox ID in LLC_TRANS
                    if error, call mexit */

                                /* return to the calling */
                         /* if ERR_RETURN */

                         /* if */
                         /* else */
                         /* llc_ev_msg */


        /* Error status          */
   /* lsap index */

      /* An event request has been received via LLC layer management.  */
```

```
    /* get the l_sap_table ptr */
    /* validate lsap table */
        /* pull the local l_lsap_table address */
        /* put the event LCB block to the lsap table */
       /* if */
         /* return not valid lcb error */
        /* completed with error */
        /* return the bad lcb */
        /* get out here */
       /* else */

    /* the common event_msg routine will handle the event LCB return */
       /* llc_lme_ev */

       /* Pointer to activate  LSAP message */
 /* Pointer to the instance table */
               /* pointer to event transaction block message */
     /* Index of remote SAP directory */
    /* LLC error status   */
       /* pointer to lsap name */

   /*
    *******  NEED TO REVISIT THIS IN ORDER TO SUPPORT *******
    *******  AN EVENT MESSAGE FROM A NETWORK LAYER *******
   */

   /* An event request has been received via LLC layer management.  */

   /* Cast on event transaction block message                   */

   /* Set completion word in the LCBI                                   */

   /* Request DMA services to transfer LCB                             */
   /*              This is one of the difficult errors to handle       */
                              /* return to the calling */
                        /* if ERR_RETURN */

   /* IF layer instance state indicates
               llc not registered with MAC (state=NULL) */

   /* THEN                                                              */

       /* Fetch pointer to lsap name in LCBI                              */

       /* Call search procedure to determine if LSAP exists              */

       /* IF LSAP exists in LSAP directory and LSAP state is Nulled      */

       /* THEN                   */

           /* Call mac_activate procedure to activate MAC Layer             */

           /* Re-issue event message to the llc_mailbox                     */
```

```
                                        /* return to the calling */
                                /* if ERR_RETURN */

                                        /* return to the calling */
                                /* if ERR_RETURN */

                                        /* return to the calling */
                                /* if ERR_RETURN */

                      /* ELSE LSAP doesn't exist                         */

                      /* Set status of LCBI to indicate bad LSAP */

                 /* Set completion word in the LCBI   */

                 /* Request DMA services to transfer LCB                         */
                 /*              This is one of the difficult errors to handle      */
                                        /* return to the calling */
                                /* if ERR_RETURN */

                      /* ENDIF                                   */

            /* ELSEIF layer instance is in the INUSE state                  */
             /* DO CASE on venue to determine whether a */

             /* Set completion word in the LCBI                                 */

             /* Request DMA services to transfer LCB                            */
             /*              This is one of the difficult errors to handle      */
                                        /* return to the calling */
                                /* if ERR_RETURN */
          /* ELSE  layer instance not able to support request             */

            /* Set status of LCBI to indicate layer unable to support request   */

            /* Set completion word in the LCBI                                  */

            /* Request DMA services to transfer LCB                            */
            /*              This is one of the difficult errors to handle      */
                                        /* return to the calling */
                                /* if ERR_RETURN */

     /* ENDIF                                                                */
     /* Return                                                              */
}          /* llc_lme_ev */
```

```
/**********************************************************************
***********************************************************************
***
***
***
***                              HONEYWELL
***
***                          Local Area Network
***
***              L O G I C A L   L I N K    C O N T R O L
***
***                             L A Y E R
***
***                        error handling module
***
***
***      Copyright: Honeywell Information System
***                 All rights reserved.
***
***      Cereate Date:       3/25/85
***      By:                 M. Lu
***      Discription:
***
***      Functions & Parameters:
***         llc_err(errcode,errmsg,err_trans)
***         short errcode;
***         char *errmsg;
***         LLC_TRANS *err_trans;
***      error cases:
***      case SAP_NOT_AVAILABLE:
***      case BAD_FSFC:
***      case BAD_LAYER_STATE:
***      case BAD_REMOTE_SAP:
***      case BAD_LOCAL_SAP:
***      case INVALID_FC:
***      case SAP_ACTIVE:
***      case MULTIPLE_ACT:
***      case BAD_R_LSAP_STATE:
***      case BAD_L_LSAP_STATE:
***      case BAD_BUFF_TRAN:
***      case BAD_BDI_TRAN:
***      case BAD_XMIT_REQ:
***      case BAD_PDU_SIZE:
***      case BUFF_TOO_SMALL:
***      case BAD_DSAP_STATE:
***      case BAD_RCV_DSAP:
***      case BAD_RCV_SSAP:
***      case PDU_TOO_BIG:
***      case BAD_RCV_REQ:
***      case XID_XMIT_ERR:
***      case XID_RESP_ERR:
***      case UNKN_MSG             0x0024  * unknown message type *
***      case BAD_SM_RMBOX        0x0025  * invalid SM return mail box *
***      case SM_RMBOX_FU         0x0026  * SM return mailbox full *
***      case BAD_SM_REQ          0x0027  * invalid SM request *
***      case NO_MEM_GET          0x0028  * getbuf could not allocate *
***      case NO_MEM_ALO          0x0029  * allocate no mem avail *
***      case RT_LCBI_ERR         0x0030  * return lcbi error *
***      case BAD_DMA_MBOX        0x0031  * invalid DMA mail box *
***      case DMA_MBOX_FU         0x0032  * DMA return mailbox full *
***      case BAD_LCBI_RP         0x0033  * bad lcbi response - big err *
***      case DMA_BUF_FAIL        0x0035  * DMA of a buffer failed *
***      default :
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***      Revision #:
***      Date:
***      By:
***      Discription:
***
***
**********************************************************************
**********************************************************************


                    /* counters */

          /* event code for event msg */
  /* get layer instance table */
  /* pointer at the ring for last message */
                    /* ERR_MSG_COUNT in the error message ring */
                    /* ring counter */
          /* save error message */
                    /* pointer at the ring for last message */
                    /* ERR_MSG_COUNT in the error message ring */
                    /* ring counter */
  /* save the err data block pointer address */
  /* save the lica_p */

           /* LLC layer error condition case      SAP_NOT_AVAILABLE*/

           /* LLC layer error condition case      BAD_FSCF */

           /* LLC layer error condition case      BAD_LAYER_STATE */

           /* LLC layer error condition case      BAD_REMOTE_SAP */

           /* LLC layer error condition case      BAD_LOCAL_SAP */

           /* LLC layer error condition case      INVALID_FC */

           /* LLC layer error condition case      SAP_ACTIVE */
```

```
        /* LLC layer error condition case      MULTIPLE_ACT */

        /* LLC layer error condition case      BAD_R_LSAP_STATE */

        /* LLC layer error condition case      BAD_L_LSAP_STATE */

        /* LLC layer error condition case      BAD_BUFF_TRAN */

        /* LLC layer error condition case      BAD_BDI_TRAN */

        /* LLC layer error condition case     BAD_XMIT_REQ */

        /* LLC layer error condition case      BAD_PDU_SIZE */

        /* LLC layer error condition case      BUF_TOO_SMALL */

        /* LLC layer error condition case      BAD_DSAP_STATE */

        /* LLC layer error condition case      BAD_RCV_DSAP */

        /* LLC layer error condition case      BAD_RCV_SSAP */

        /* LLC layer error condition case      PDU_TOO_BIG */

        /* LLC layer error condition case      BAD_RCV_REQ */

        /* LLC layer error condition case      XID_XMIT_ERR */

        /* LLC layer error condition case      XID_RESP_ERR */

        /* LLC layer error condition case       */

        /* LLC layer error condition case       BAD_SM_RMBOX */
         /* invalid SM return mail box */

        /* LLC layer error condition case       SM_RMBOX_FU */
       /* SM return mailbox full */

        /* LLC layer error condition case       BAD_SM_REQ */
              /* invalid SM request */

        /* LLC layer error condition case       NO_MEM_GET */
              /* getbuf could not allocate */

        /* LLC layer error condition case       NO_MEM_ALO */
                      /* allocate no mem availible */

        /* LLC layer error condition case       RT_LCBI_ERR */
              /* return lcbi error */

        /* LLC layer error condition case       BAD_DMA_MBOX */
              /* invalid DMA mail box */
        /* LLC layer error condition case       DMA_MBOX_FU */
               /* DMA return mailbox full */

        /* LLC layer error condition case       BAD_MAC_MBOX */
               /* invalid MAC mail box */

        /* LLC layer error condition case       MAC_MBOX_FU */
               /* MAC return mailbox full */

        /* LLC layer error condition case       BAD_LCBI_RP */
               /* bad lcbi response - big err */

        /* LLC layer error condition case       TOO_LONG_DLCB */
       /* data too long to put in lcb */

        /* LLC layer error condition case       DAM_BUF_FAIL */
/* DMA of a buffer failed */

        /* LLC layer error condition case BAD_RESOLVE_MBOX      */
        /* LLC layer error condition case BAD_REG_MBOX     */

        /* LLC layer error condition case MBOX_ON_ERR     */

        /* LLC layer error condition case MBOX_OFF_ERR     */

        /* LLC layer error condition case MBOX_DL_ERR     */

        /* LLC layer error condition case SENDMSG_ERR     */

        /* LLC layer error condition case BAD_IO_MSG     */

        /* LLC layer error condition case BAD_REG_MBXDIR      */

        /* LLC layer error condition case INV_TRANSF     */

        /* LLC layer error condition case QUEUE_FULL     */

        /* LLC layer error condition case DEFAULT_ERR     */

        /* LLC layer error condition case -1     */

        /* LLC layer error condition case 1     */

        /* LLC layer error condition case 0     */

      /* switch */

      /*llc_event_cd = EVENT_INDIC;     * event indicate mssage */
     /* event indicate mssage */
      /* send event message to system management */

    /* end llc_err */

/* central handling for exit condition */
```

```
        /* kernel call, shut down the lacs */
        /* end llc_exit */

         /* only return to the calling. not exit */
        /* -2 for all error case */
        /* end llc_err_return */
```

Document Summary

```
Document Name:LLCSPN       Edition:  .042          Protect:No     (Yes or No)
Author:       MICHAEL LU
Title:         IEEE 802.2 LLC LAYER SPECIFICATION
Comments:      Michael Lu  After Release
Lines/Page:   64
Total Pages:  42
Widow Length: 05     (Pagination)
Zone Length:  05     (Hyphenation)

Pitch:        1      (1=10 chars per inch, 2=12 cpi, 3=15 cpi)
Justify Right:Yes    (Yes or No)
Font:         001    (1=Non-Proportional)

Function:     Date:       Time:      Operator:     Volume:
Typed        85/08/19    13:56      OSHAUGHNES    F1UR31
Changed      86/09/25    15:09      LU            F1USR2
Printed      86/09/25    15:11      LU
Filed        86/09/02    13:59      LU            F1USR2
```