//

SOFTWARE COMPONENT SPECIFICATION

SYSTEM: LEVEL 6 MOD400 OPERATING SYSTEM

SUBSYSTEM: LAN FACILITY

COMPONENT: SYSTEM MANAGEMENT

PLANNED RELEASE: MOD400 4.1

SPECIFICATION REVISION NUMBER: B

DATE: JULY 25,1985

AUTHOR: D.E. O'SHAUGHNESSY

This specification describes the current definition of the subject software component, and may be revised in order to incorporate design improvements.

TABLE OF CONTENTS

1.0 INTRODUCTION
1.1 BACKGROUND
1.2 PURPOSE
1.3 BASIC STRUCTURE
1.4 BASIC OPERATION

2.0 L6 SYSTEM MANAGEMENT EXTERNAL DESCRIPTION
2.1 DATA STRUCTURES
2.2 EXTERNAL INTERFACES
2.3 INITIALIZATION REQUIREMENTS
2.4 TERMINATION REQUIREMENTS
2.5 ENVIRONMENT
2.6 TIMING AND SIZE REQUIREMENTS
2.7 ASSEMBLY AND LINKING
2.8 TESTING CONSIDERATIONS
2.9 DOCUMENTATION CONSIDERATIONS
2.10 OPERATING PROCEDURES
2.11 ERROR MESSAGES

3.0 L6 SYSTEM MANAGEMENT INTERNAL DESCRIPTION
3.1 OVERVIEW
3.2 SUBCOMPONENT DESCRIPTION
3.3 FUTURE DEVELOPMENT AND MAINTENANCE

4.0 L6 SYSTEM MANAGEMENT PROCEDURAL DESIGN

5.0 ISSUES

6.0 LACS SYSTEM MANAGEMENT EXTERNAL DESCRIPTION
6.1 DATA STRUCTURES
6.2 EXTERNAL INTERFACES
6.3 INITIALIZATION REQUIREMENTS
6.4 TERMINATION REQUIREMENTS
6.5 ENVIRONMENT
6.6 TIMING AND SIZE REQUIREMENTS
6.7 ASSEMBLY AND LINKING
6.8 TESTING CONSIDERATIONS
6.9 DOCUMENTATION CONSIDERATIONS
6.10 OPERATING PROCEDURES
6.11 ERROR MESSAGES

7.0 LACS SYSTEM MANAGEMENT INTERNAL DESCRIPTION
7.1 OVERVIEW
7.2 SUBCOMPONENT DESCRIPTION
7.3 FUTURE DEVELOPMENT AND MAINTENANCE

8.0 LACS SYSTEM MANAGEMENT PROCEDURAL DESIGN

9.0 ISSUES

A.1 APPENDIX 1 SMPDU EXAMPLE AND OVERVIEW

REFERENCES

[1] Engineering Product Specification (H/W), Local Area Controller Subsystem (LACS), Rev F, A. C. Hirtle, Oct 4, 1984.

[2] Engineering Product Specification, LAN Software, R. Dhondy, July 31, 1985.

[3] Engineering Component Specification, LAN Data Structures, July , 1985

[4] Procedural Design Language Memo, PDL Description, H.King, July , 1985

[5] Engineering Component Specification, LACS Driver Interface Services, P.Stopera July , 1985

[6] Engineering Component Specification, LACS Driver Megabus Services, P.Stopera July , 1985

[7] Engineering Component Specification, LAN Configuration Services, L.Vivaldi, July , 1985

[8] IEEE 802.1 Systems Management Part B, Revision H, June 14, 1985

[9] CCITT Recommendation X.409, Message Handling Systems: Presentation Transfer Syntax and Notation, September,1984

[10] ISO/DIS 7498 Information Processing Systems -- Open Systems Interconnection -- Basic Reference Model, October 15,1984

[11] DSA 70, Distributed Systems Administration and Control Architecture xxxxx xx,19xx

[12] DSA 71, Distributed Systems Administration and Control Administrative Exchange Protocol xxxxx xx,19xx

[13] DSA 72a, Distributed Systems Administration and Control Network Control Language xxxxx xx,19xx

[14] DSA 76, Distributed Systems Maintainability xxxxx xx,19xx

[15] 09-0016-00 ESPL Software Technical Reference Manaual, Vol.1, Kernel and Support software (Bridge Communications, Inc.) xxxxx xx,19xx

1.0 INTRODUCTION

1.1 BACKGROUND

Management facilities must be provided to support local area network (LAN) operation. It provides the capabilities to start up, carry traffic, reconfigure, restart and close down the LAN. The system management function is the mechanism which provides these capabilities for LANs interfaced to a L6 through a local area controller subsystem (LACS) board. Architecturely, system management sits beside each of the defined layers. This allows it to directly interface with Transport layer management, Network layer management, LLC layer management, MAC layer management, and Physical layer management. It also provides a direct interface to a management application which sits on top of system management. This is the local user of system management services. A local user of system management could be the DSA Node ADministrator task (NAD), a Test and Verification (T&V) task, or a standalone administrative application (ADAP). A NAD interface to system management could provide all the management tools currently available to manage DSA networks toward administering the LAN. A T&V user would provide the capabilities to test LAN hardware. An ADAP type of user could be any type of application ranging from a simple statistical gathering type of program to a sophisticated LAN network operator interface. In addition to the local user interface, a remote interface can be provided to system management through the concept of system management data service interface (SMDSI). A remote interface is not to be initially supported. At present only NAD and T&V are planned as users of system management. A block diagram of the architecture of system management is shown in figure 1.1.a.

The System Management function administers the LAN layers by operating on the data structures which control and describe the operation of each of the layers. These data structures are viewed slightly differently in a DSA environment than in a totally IEEE802 environment. In a DSA environment, control is exercised over data structures called objects. In an IEEE802 environment control is exercised over data structures called components. The LACS system management function provides services which span both of the different perspectives and are from herein referred to as objects. Objects defined for a Honeywell IEEE802 LAN are :

Controller(CT)
: Describes the LACS board; maintained by the L6 system management function.

```
                                   ¶-------------¶
                 4-SMDSI           ¶             ¶
               ¶-------------------¶             ¶
 (TSAP)        V                   ¶             ¶
¶-(*)...(*)---(*)--¶------¶        ¶   SYSTEM    ¶
¶       TRANSPORT  :XPORT ¶ 4-LMI  ¶             ¶
¶  Protocol Entity : LME  ¶<------>¶             ¶
¶------------------:------¶        ¶             ¶
                 3-SMDSI           ¶             ¶
                ¶------------------¶ MANAGEMENT  ¶
  (NSAP)        V                  ¶             ¶
¶---(*)........(*)-¶------¶        ¶             ¶
¶       NETWORK    :Netwrk¶ 3-LMI  ¶             ¶
¶ Protocol Entity  : LME  ¶<------>¶             ¶
¶------------------:------¶        ¶             ¶
                        2b-SMDSI   ¶             ¶
                ¶------------------¶ APPLICATION ¶
  (LSAP)        V                  ¶             ¶
¶---(*)........(*)-¶------¶        ¶             ¶
¶       LLC     -  : LLC  ¶ 2b-LMI ¶             ¶
¶ Protocol Entity  : LME  ¶<------>¶             ¶
¶------------------:------¶        ¶             ¶
                                   ¶  PROCESS    ¶
        (MACSAP)                   ¶             ¶
¶---------(*)------¶------¶        ¶  (SMAP)     ¶
¶       MAC        : MAC  ¶ 2a-LMI ¶             ¶
¶ Protocol Entity  : LME  ¶<------>¶             ¶
¶------------------:------¶        ¶             ¶
                                   ¶             ¶
        (PSAP)                     ¶             ¶
¶---------(*)------¶------¶        ¶             ¶
¶       PHY        : PHY  ¶ 1-LMI  ¶             ¶
¶ Protocol Entity  : LME  ¶<------>¶             ¶
¶------------------:------¶        ¶-------------¶
```

LME = Layer Management Entity
LMI = Layer Management Interface
SMDSI = System Management Data Services Interface

FIGURE 1.1.a MANAGEMENT MODEL

System Management Administrative Function(SMAF)
Describes the System management administrative function; maintained in L6.

Physical Line (PL)
Describes the characteristics of the adapter boards' hardware and firmware. Described by attributes and statistical information available from both the MAC and Physical layers. The Physical Line is maintained in the LACS.

Physical Connection (PC)
Describes the physical connection onto the LAN. The Physical Connection object is not supported for the LAN.

Logical Line (LL)
Describes the characteristics of a LSAP. Described by the set of attributes for an LSAP. Maintained in the LACS board.

Link Connection (LK)
Describes the characteristics of the link connection between two LSAPs. Described by a set of statistics maintained by the LLC layer on the link connection. Maintained in the LACS. The LK object is not initially supported by the LLC layer (type II services not supported).

The interfaces to the system management services, in both the L6 and the LACS, is IEEE802 oriented. It is based on the structure of system management PDUs described in the standard for IEEE802.1, Part B. System management supports three types of PDUs, a request SM PDU, a response SM PDU, and an event SM PDU. A user issues a request PDU to system management and system management responds to that request with a response PDU. The request and response PDU contain routing and operation information. The routing information identifies what layer and object the operation is on. The layer is identified by fields describing the layer , the sublayer and the layer instance. The object is identified by the layer internal selector. This field is defined to contain a set of DSA-like selection parameters; name, class, type, venue, and state of an object. The operation information specifies what operation to perform. The following operations are supported by the system management layer server:

GET - Read a specified attribute or group of attributes associated with this layer.

SET - Set an object attribute or group of attributes to the specified value(s).

ACTION - Perform the action on the layer entity or object. Action provides the ability to control an objects state or perform layer specific operations. The following common actions have been defined for LAN operations:

UPDATESTATE - Update the state of an object or component.

CREATE - Create an object or component. This is applicable only for LACS resident objects. In initial releases all objects and components are created during initialization and configuration of the LACS.

DELETE - Delete, remove or destroy an object or component. This is applicable only for LACS resident objects.

LIST - List the selection parameters (name, class, type, venue) for the specified class of objects or components.

TEST - Test an object or component.

DUMP - Dump memory of the LACS board.

LOAD - Load LACS memory from the given LACS bound unit. Start execution from the given address.

In addition the interface between the LACS system management function and the layer management functions will be implemented to insure an IEEE802 compatibility.

DSA objects and IEEE802 components have a set of states which describe their present operational capabilities. The current states identified for IEEE802 components still lack any detailed definition, so the DSA state representation has been used to describe each of the objects. DSA defines seven possible states which can describe any object. These states are used for administration and should not be confused with the substates used for operational control. The seven DSA states defined for the objects of a LAN are as follows:

IN-USE - The object is fully operational and is currently being used by a higher layer object.

ENABLED - The object is fully operational but is not currently being used by a higher layer object. The Enabled state is not currently supported.

DOWN - The object is not available for service due to a detected hardware failure. Applicable only to the controller and MAC (physical line).

DISABLED - The object is not available for use by a higher layer object due to a lower layer supporting object being unavailable. The Disabled state is not currently supported.

LOCKED The object is unavailable for service to other objects. Entering the LOCKED state is an abortive operation.

TEST - The object is under exclusive control of a test function.

SHUTDOWN - The object is undergoing a graceful transition to the LOCKED state. The Shutdown state is not currently supported.

NONEXISTENCE - This is not truly a representative state but is used to describe objects with which there is not an associated data structure.

A summary of the DSA states to be implemented is shown in figure 1.1.b.

System management will also report unsolicited events and statistical information to an administrative function when requested. The system management user must issue a request containing a buffer which can be used by the system management function to post a SM Event PDU back to the user. Only unsolicited events relating to error conditions are currently reported through the system management function. The reporting of events due to specified conditions (such as the opening or closing of a connection) is for future study.

	CONTROLLER	SYSTEM MGT. ADMIN.FCT.	PHYSICAL LINE	LOGICAL LINE	LINK CONNECTION
IN-USE	To Be Supported	To Be Supported	To Be Supported	To Be Supported	To Be Supported in Future
ENABLED	For Future Study	For Future Study	For Future Study	For Future Study	For Future Study
DISABLED	Not Applicable	For Future Study	For Future Study	For Future Study	For Future Study
LOCKED	To Be Supported	To Be Supported	To Be Supported	For Future Study	Not Applicable
DOWN	To Be Supported	Not Applicable	To Be Supported	Not Applicable	Not Applicable
TEST	To Be Supported	For Future Study	To Be Supported	For Future Study	Not Applicable
SHUTDOWN	For Future Study	Not Applicable	For Future Study	For Future Study	For Future Study

Figure 1.1.b DSA STATES SUPPORTED ON 802 LAN

1.2 PURPOSE

System management provides administrative and maintenance services through its interfaces to the Transport, Network, LLC, MAC, and Physical layer management functions. These services are provided to perform the following functions :

- LACS Loading/Dumping
- LACS Initialization
- LACS Object Control and Status
- LACS Statistics
- LACS Testing

In addition to the service provided to the LACS board, System management services are provided to initialize and control the data structures defined in L6 memory concerning the LAN. System management also controls the controller object and the system managment administrative function.

1.3 BASIC STRUCTURE

System Management services are provided to a Node Administration Facility, Administrative Application Program or T&V users. Figure 1.3.a shows the relationships of the various components involved in performing administration and maintenance functions. In a DSA environment the System Management functions are invoked by NAD (Node Administrator) which in turn is driven by an operator interfacing with an Network Operator Interface (NOI) module in a local or remote node. In a non-DSA environment an Administration Application Program (ADAP) can be provided to perform similar functions as NAD but more limited in its scope. All LAN configurations must support a T&V user which can interface to system management. In addition, initial releases require NAD to serve as the user of system managment for administrative and maintainance of the LAN.

A request made to system management is handled in a similar manner as any other layer server. The LACS Driver Interface service (LDIS) software is invoked due to an IORB request from a system management user. It processes the IORB and calls the system management layer server (SM server or SM LS). The SM server starts processing the request contained in the IORB buffer. The request is presented in the syntax and format of an IEEE defined protocol data unit (PDU). The SM server will call the proper layer management routine in the L6 to perform the operation specified in the users request. If the service can not be completed in the L6, the SM server issues the IEEE802 formatted request to the system management process (SM layer instance) in the LACS. This is accomplished by requesting the LACS Driver Megabus services (LDMS) software to issue a LAN Control Block (LCB) to the LACS. On the LACS, the IO Dispatcher process receives the request, determines it is for system management and issues a message to the SM layer instance. The LACS SM layer instance routes a request to the proper layer manager to complete the operation and returns the result along with the completed LCB to the L6. The SM server in the L6 posts the IORB back to the user.

The SM server also provides a layer management function to administer and maintain the LACS controller and the system management administrative function.

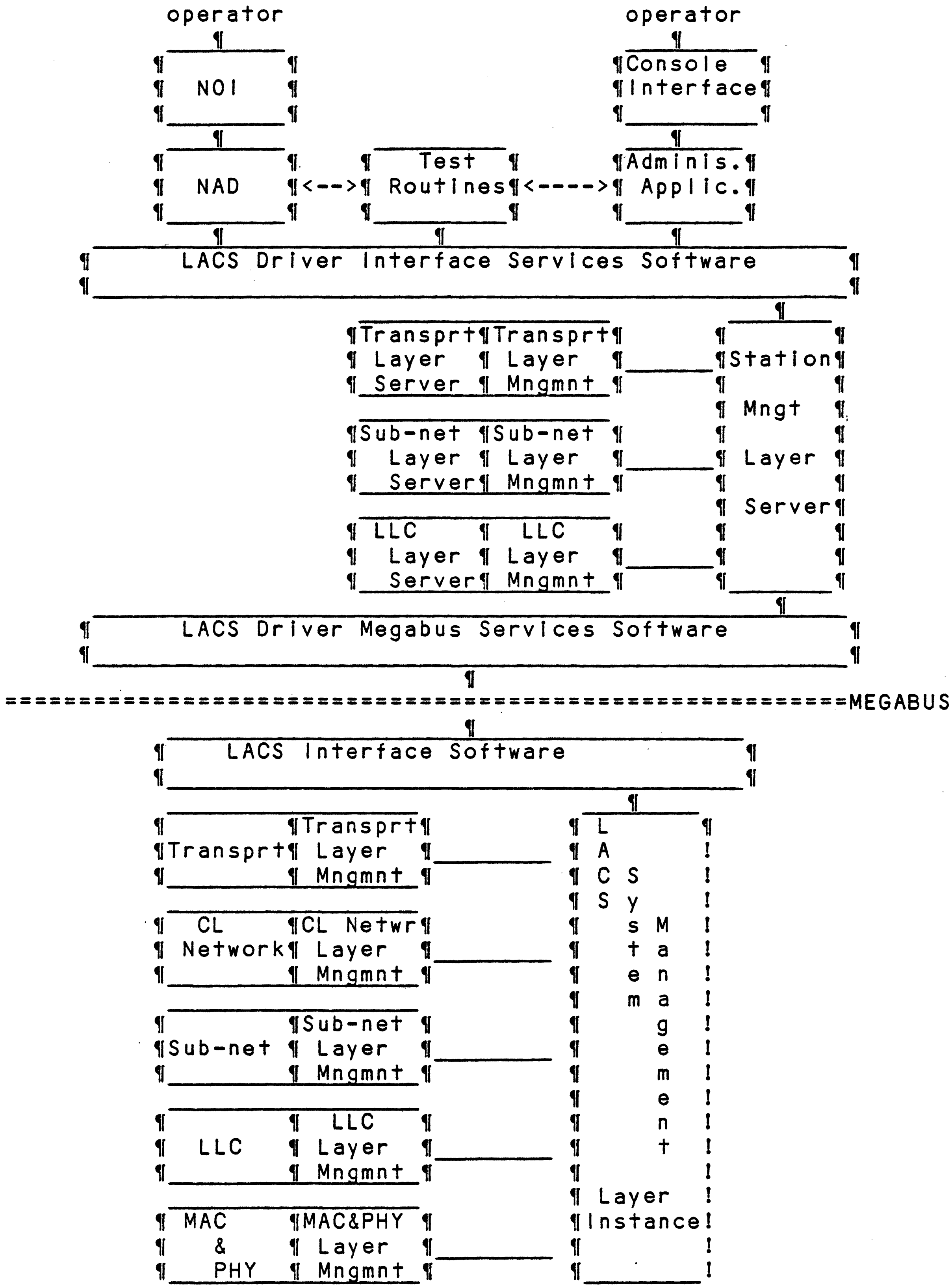

Figure 1.3.a System Management Components

1.4 BASIC OPERATION

System management is physically split across the Megabus into two separate functions. The system management function residing in the L6 provides the interface to the NAD, ADAP, or test routines requiring system management services. This is referred to as the System Manager Layer Server (SM server or SM LS). Requests are made to SM server through the use of IORBs passed as part of a RQIO call. The SM server function will make use of the same LDIS routines that other layer servers use. The LDIS processes the IORB, queues the IRB on the System management users Resource Control Table (RCT), then calls the SM server. All IORBs received by the SM server will point to a buffer containing a system management PDU. The system management PDU will be formatted as specified in X.409 and IEEE802 standards. The SM server will interpret control information in the IORB to determine whether it is a request PDU or a buffer to to be used to issue an event PDU back to the application. In the case of a request which can be performed entirely within the L6, the SM server will provide the requested service by requesting service from the proper L6 layer management routine. Results are returned as part of the IORB completion. Requests which require services of the system management function residing in the LACS board must be issued to the LACS system management function (SM layer instance) by issuing an LCB. The LCB must contain a pointer to the PDU received by the SM server in the IORB. It must also contain a pointer to a task request block (TRB) to be used by the LDMS interrupt routine. The LCB is then passed to the LDMS routine to be issued to the LACS board. The SM server then terminates itself until the LCB is completed.

When the LCB is received by the system management function on the LACS, the SM PDU pointed to by the LCB is interpreted. System management will decide whether the command can be handled directly or whether service will be required of one or more of layer management functions defined on the LACS. Any results due to the commanded operation are returned as part of the LCB completion and a response PDU pointed to by the LCB. The LDMS interrupt routine is invoked due to the LCB completion. It requests the task described by the LCBs TRB to be executed. The SM server has set the start address in the TRB to the address of the code it wishes to begin execution and a pointer to a block of memory containing the context of the SM server when the LCB was originally issued. The completed LCB will point to the same buffer as it was issued but the buffer will now contain a response PDU. In the case where results needed to be obtained from the L6 and the LACS, the SM server will return the results to the user combined in a single response PDU.

Unsolicited events are also reported to the user. The user must issue an IORB with a buffer that can be used to deliver unsolicited event PDUs from the system management to the user. The SM server will post that IORB to the user when an unsolicited event is reported to it by one of the layer management functions.

The SM server also provides the layer management function for LACS controller and the system management administrative function (SMAF). The layer manager function provides the ability to read the attributes and statistical counts of the controller and SMAF. It provides the ability to load and dump the controller. It also provides a controller TicToc function which periodically requests the controller to respond to a special LCB request to insure the controller is operational. The SMAF provides the capability of listing the selection parameters describing all of a specified type of object or component.

2.0 L6 SYSTEM MANAGEMENT EXTERNAL DESCRIPTION

2.1 L6 DATA STRUCTURES

System management directly and indirectly operates on and controls the data structures associated with every defined object. These data structures are the same as those accessed by the LDIS routines, the layer servers, and the LDMS routines. Theses data structures are described in more detail in the LAN System Data Structures Component Specification. A block diagram of these data structures is shown in figure 2.1.a. The system control block contains a pointer to the LAN controller directory. The LAN controller directory contains a pointer to a controller table for each LACS present in the system. It also contains a pointer to a set of SAP directories, a local SAP directory and a remote SAP directory, for each layer defined for LAN operation. At present, there are only local directories for management SAPs and physical SAPs. There will always be a remote and local SAP directory for any link, network, and transport defined SAPs within the system.

A controller table contains points to eight layer tables. Layer table zero being a management layer table, and layer table one through seven assigned according to the seven layer ISO model (physical being table one, application being table seven). Each layer table in turn contains pointers for up to eight layer instance table, one for each instance of layer per controller. The controller tables contain the attributes of each controller such as name and state information as well as parameters allowing the number of LCBs to be issued to the controller to be restricted. The layer instance table contains information about the type of protocol it represents, what L6 interrupt level it has been assigned and a queue of active LCBs which has been issued to the layer instance for this controller.

Each user interface to the LAN is assigned a RCT. The RCT maintains a queue of active IRBs issued on it's assigned LRN. A transfer directory is also maintained for each RCT. The first entry in the transfer directory is a pointer to the transfer table for connectionless operations while the remaining entries are assigned one per connection. Each transfer table contains parameters allowing the flow of read and write type LCBs to be controlled on a user basis and a pointer to the layer instance table for this transfer table.

2.1.1 MOD400 DATA STRUCTURES IMPLEMENTED

The following system owned data structures are referenced by the SM server:

Task Control Block (TRB)
System Control Block (SCB)
Logical Resource Table (LRT)
Group Control Block (GCB)
Resource Control Table (RCT)
Intermediate Request Block (IRB)

2.1.2 SYSTEM MANAGEMENT SAP TABLE

A pointer to the system management sap table is pointed to by each entry in the System management SAP directory for each SAP defined for system management. A system management SAP table follows the standard SAP description and contains a unique portion as well. The format of the attribute table is as follows:

NAME

CLASS

TYPE

VENUE

SM_STE	[xx]	SM Object State. 1 word with two 8-bit fields. First byte is state: State — Major state of System Manager. Default value set to indicate locked - 03. At CLM set to LOCKED. Second byte is substate: Substate — Null for initial implementation. Default value of zero. At CLM set to RESET t
MFG_ID	[xx]	Manufacturer name and country. The manufacturer name consists of the character string 'Honeywell Informations Systems,USA'.
STN_ID	[xx]	Manufacturer specific station type. TBS
OPT_SP	[xx]	Options supported. 1 word. Null for initial implementation. Default value of zero.
MX_SZE	[xx]	Maximum SM PDU size in bytes. 1 word. Default value of 1k bytes if not specified in the configuration file.
IDL_SZ	[xx]	Ideal SM PDU size in bytes. 1 word. Default value of 1k bytes if not specified in the configuration file.

MAX_CR [xx] 16 bit integer.Maximum number of LCBs allowed for SM server. Returned as SM PDU credit to user for an activate request. Default value of xx.

EV_LCB [xx] Implicit address
Address of outstanding event LCB. Initial value is null until the event LCB is received.

2.1.2 LACS CONTROLLER ATTRIBUTES

The controller attributes are stored in the controller table. A description of the controller table is contained in the LAN data structures component specification. The parameter ids of those attributes residing in the L6 controoler table are defined as follows:

2.2 L6 EXTERNAL INTERFACES

External interfaces to the System Management function within the L6 must be provided for the NAD, ADAP, and test routines. The interface is through the executive monitor call, RQIO and an associated IORB. This provides the interface to users of system management. Along with each IORB, a system management command buffer must be specified which contains the system management PDU. A user of system management will issue IORBs with a command buffer containing either a request PDU or a buffer to receive an event indicate PDU. The command buffer of a completed IORB will contain either a response PDU or an event indicate PDU. The user must insure that the buffer space allocated for request PDUs is large enough to contain the expected response PDU. In the event where the size of the response PDU exceeds the size of the IORB buffer, the IORB will be returned with a status indicating the buffer was too small and the size of the required buffer placed in the actual size field of the IORB.

A system management PDU is used for both internal L6 system management processing and can also be passed to the LACS system management function through a LCB.

The system management server also interfaces to the LDMS routine to communicate with the LACS board. The LDMS routine is called directly and passed an LCB to be issued to the LACS board.

The SM server provides a direct call interface to support the LDIS routines and the initialization of the LAN in the system. In particular this provides support for activating a local SAP, the SM server insures that LAN facilities are initialized.

2.2.1 IORB FORMAT

The IORB format is the standard IORB format with a LAN extension for the MOD400 operating system. Every IORB contains the standard portion of the IORB as well as a LAN extension. The LAN extension is unique for each LAN IORB function code defined for bits C-F in the CT2 word (rb_ct2) of the IORB. The SM server processes two types of IORB extensions, the SM request/response IORB and the event indicate IORB. The IORB format for an Activate local SAP IORB is described in the LDIS component specification. The format of standard IORB is as follows:

rb_lrx contains extended lrn, indicators
 bits 4-f - extended lrn if rb_ct2 bits 0-7 = fd
 bit 0 - rb_adr points to a buffer descriptor block when set

rb_rrb reserved for system use

rb_ct1 contains indicators, status
 bits 0-7 - return status
 60 -> success
 64 -> invalid IORB
 6B -> aborted IORB
 6C -> inconsistent request
 bit a - rb_adr points to a buffer descriptor when set
 bit f - must be equal to 1

rb_ct2 contains lrn, indicators, function code
 bits 0-7 - lrn, if lrn = x'fd' then the lrn is extended, to find the extended lrn go to rb_lrx field to obtain the lrn
 bit 9 - buffer starts in the right byte when set
 bit a - iorb is extended when set (must be set)
 bits c-f - function code
 6 - system management request.
 A - Activate Local SAP request
 - refer to LDIS for an Activate
 E - event request

rb_adr contains pointer to buffer address, or a pointer to a buffer descriptor block (bdb)

rb_rng contains buffer range in bytes, or total range of all the buffers in the buffer descriptor block in bytes

rb_dvs contains device specific information
 bit e - disconnect with queue abort when set and iorb function code = disconnect with queue abort (b). This is not applicable to a SM IORB.

rb_rsr contains residual range, or total residual range of all the buffers in bytes, total number of bytes not used.

rb_st1 contains status
- bit F - invalid function code when set
- bit E - ram memory exhausted when set
- bit D - ram location non-existent when set
- bit C - ram parity error when set
- bit B - level 6 memory yellow when set
- bit A - level 6 memory non-existent when set
- bit 9 - level 6 bus parity error when set
- bit 8 - level 6 memory red when set

rb_ext contains LAN iorb extension
- see following sections for format

2.2.1.1 SM REQUEST/RESPONSE IORB LAN EXTENSION

The following parameters are defined for an IORB intended for system management as a request/response IORB:

rb_fsf contains function specific function code
- 0 - local request PDU

rb_fss contains function specific status
- 0 - success
- 1 - invalid function specific function code
- 2 - invalid IORB
- 3 - buffer range too small

rb_asz contains the actual total size of the required buffer when status indicates the buffer was too small.

rb_lcb reserved for LCB, xxxxxx words are reserved for use of LAN software to create an LCB for this IORB request

2.2.1.2 SM EVENT INDICATE IORB LAN EXTENSION

The following parameters are defined for an IORB intended for system management as an event indicate IORB:

rb_fsf contains function specific function code
- 4 - SM event indicate

rb_fss contains function specific status
- 0 - success
- 1 - invalid function specific function code
- 2 - invalid IORB
- 3 - buffer range too small

rb_asz contains the actual total size of the required buffer when status indicates the buffer was too small.

rb_evt contains the event mask for reporting SM events.
1 - data indicates
not applicable to SM requests
2 - error
4 - additional write credit
8 - SAP deactive

rb_lcb reserved for LCB, xxxxxx words are reserved for use of LAN software to create an LCB for this IORB request

2.2.1.3 SM ACTIVATE IORB LAN EXTENSION

2.2.1.4 BUFFER DESCRIPTOR BLOCK (BD)

The bd is used by an application to pass data to the lan subsystem when more than one buffer must be passed. The bd only exists if the user bit is set in the rb_ct1 field of the iorb.

bd_sy1 reserved for system use

bd_sy2 reserved for system use

- the following 3 fields will be repeated the number of times in the bd_bct field above (note: buffers must start on a word boundry)

bd_adr contains buffer address

bd_rng contains buffer range in bytes

bd_rsr contains residual range in bytes

bd_id1 contains indicators
bit 0 - buffer starts on an odd byte when set
bit 1 - last buffer in list when set

2.2.2. System Management Command Buffer

The system management command buffer contains a System management Protocol Data Unit (SM PDU) in the form of an IEEE802 SM PDU. The system management PDU is formatted to conform to X.409 and IEEE802.1 standards. A user of system management must also conform to additional Honeywell private definitions allowed within the context of an IEEE802 SM PDU as specified in the following section.

2.2.2.1 System Management Protocol Data Unit Description

The L6 system management layer server can handle three different types of system management PDUs. It can receive a request PDU from a user to perform a specified service or issue a response PDU at the completion of performing a requested service. The system manager layer server also issues event PDUs to a user at the occurrence of an unsolicited event. A PDU is formatted according to the syntax described in the I.E.E.E. 802.1 and X.409 standards. An example of a SM PDU construction is located in appendix 1.

The syntax of a system management PDU is as follows:

```
SystemsManagementPDU ::= Choice  §

    [1] Implicit RequestPDU,
    [2] Implicit ResponsePDU,
    [3] Implicit EventPDU  †
```

The syntax of a request PDU is as follows:

```
RequestPDU ::= Sequence of RequestInfo

RequestInfo ::= Choice  §

    [1] Implicit GetRQ,
    [2] Implicit SetRQ, (Not initially supported)
    [4] Implicit ActionRQ  †
```

The RequestInfo record identifies the type of operation to be performed. Its layer info field identifies the layer, sublayer, and layer instance to system management. The system management layer server can service request PDUs which specify only single operations. Every IORB containing a request PDU issued to the system management server is returned to the user pointing to a buffer containing a response PDU.

The syntax of the response PDU is as follows:

```
ResponsePDU ::= Choice  §

    [0] Implicit Status,
    [1] Implicit Sequence of ResponseInfo  †
```

```
Status ::= Sequence  §

   [0] Implicit StatusCode,
   [1] StatusInfo  †


StatusCode ::= Integer  §

   success              (0)
   bad operation        (1)
   bad layer            (2)
   bad sublayer         (3)
   bad layer instance   (4)
   TBS  †


StatusInfo ::= Any  §

   PrivateStatus Info [0] ::= Implicit Octetstring

   The present implementation
   requires a two octet record which is
   equivalent to the reason code
   defined in DSA. †

ResponseInfo ::= Choice  §

   [1] Implicit GetRSP,
   [2] Implicit SetRSP, (not initially supported)
   [4] Implicit ActionRSP  †
```

The following operations are supported by the system management layer server:

GET - Get a specified attribute or group of attributes associated with this layer.

SET - Set an object attribute or group of attributes to the specified value(s). Not initially supported .

ACTION - Perform the action on the layer entity or object. Action provides the ability to control an object's state or perform layer specific operations. The following common actions have been defined for LAN operations:

LISTALL - List all the selection parameters (name, class, type, venue, mapping) for each object or component of the specified class.

UPDATESTATE - Update the state of an object or component.

LOAD - Load LACS memory from the given LACS bound unit. Start execution from the given address.

DUMP - Dump memory of the LACS board to the given pathname.

CREATE - Create an object or component. This is applicable only from a LACS board perspective. In initial releases all objects and components are created during initialization and configuration of the LACS. Not initially supported except during LAN facility initialization.

TEST - Test the specified object or component of the specified class.

For more information on the available commands to each layer, refer to the proper specification.

2.2.2.2. Get Request

A Get request PDU reads the set of attributes specified and returns a Get response PDU back to the user. The syntax of a Get request PDU is as follows:

```
GetRQ ::= Sequence  §

   [0] ResourceID,
   [1] Implicit Exchange ID,
   [2] Access control,
   [3] Implicit ParameterList  †


ResourceID ::= Choice  §

   PrivateResourceID [0] Any,
   [1] Implicit Layer Info  †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §Null†,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †

Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


LayerInternalSelector ::= Any  §

   PrivateInternalSelector [0] Implicit sequence of
                               SelectionParameters  †


   [0] SelectionParameters ::= Sequence  §
```

[0] Class ::= Implicit Integer §

16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

[1] Name ::= Implicit IA5 string DEFAULT §Null†

4 Octets. Identifies a unique system name of an object. Always 8 ASCII characters, left justified and null padded. Default is 8 null characters.

[2] ObjectState ::= Implicit Sequence §

[0] Implicit State,
[1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class.

AnyType (0000)
LACS Controller (LNCT)
IEEE 802.1 SM (8021)
IEEE 802.2 LLC (8022)
IEEE 802.3 MAC (8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue (0)
Local (1)
Image (2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length.
The first octet specifies class and the second octet
specifies length of the object name. The last 8
octets
indicate the object or component name to
which the object is mapped. The name is always 8
ASCII
characters, left justified and null padded.
Not initially supported. †

ExchangeId ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. The exchangeid will be returned in the response PDU.

AccessControl ::= Implicit Octetstring

The current implementation requires that the octetstring for access control consist of 2 octets and consist of all nulls.
Not initially supported.

ParameterList ::= Set of Parameter

```
Parameter ::= Choice  §

   [0] Implicit Status,
   DefinedParameter [1] Any  †

DefinedParameter ::= Private Parameter  §

   [0] Code ::= Integer  §

   AllAttributes (12)
   AllStatistics (4)  ††
```

2.2.2.3. Get Response

A get response PDU returns the values for the set of attributes or a status of the operation. The syntax of a get response PDU is as follows:

```
GetRSP ::= Sequence  §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   Choice  §

       [2] Implicit Status,
       [3] Implicit ParameterList  ††

ResourceID ::= Choice  §

   PrivateResourceID [0] Any,
   [1] Implicit Layer Info  †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §Nullt,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †


Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


Layer Instance ::= Implicit integer

       16 bit integer. It is equivalent to
       the controller number it resides on
       and the index to the layer instance
       table.
```

LayerInternalSelector ::= Any §

PrivateInternalSelector [0] Implicit sequence of
SelectionParameters †

[0] SelectionParameters ::= Sequence §

[0] Class ::= Implicit Integer §

16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

[1] Name ::= Implicit IA5 string DEFAULT §Null†

4 Octets. Identifies a unique system name of an object. Always 8 ASCII characters, left justified and null padded. Default is 8 null characters.

[2] ObjectState ::= Implicit Sequence §

[0] Implicit State,
[1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState	(0)
Reset	(1)
Halted	(2)
Loaded	(3)
Started	(4)
Operational	(5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

Anytype	(0000)
LACS Controller	(LNCT)
IEEE 802.1 SM	(8021)
IEEE 802.2 LLC	(8022)
IEEE 802.3 MAC	(8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue	(0)
Local	(1)
Image	(2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length. The first octet specifies class and the second octet specifies length of the object name. The last 8 octets indicate the object or component name to which the object is mapped. The name is always 8 ASCII characters, left justified and null padded. Not initially supported. †

ExchangeId ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. It is the same as the exchangeid in the request PDU.

Status ::= Sequence §

[0] Implicit StatusCode,
[1] StatusInfo †

StatusCode ::= Integer §

success	(0)
bad layer internal selector	(6)
bad parameter id	(7)
TBS †	

StatusInfo ::= Any §

PrivateStatus Info [0] ::= Implicit Octetstring

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification. †

ParameterList ::= Set of Parameter

Parameter ::= Choice §

[0] Implicit Status,
Defined Parameter [1] Any †

DefinedParameter ::= Private Parameter

The syntax and description of the the parameter ids can be found in section x.x.x.x containing the description of all attributes.

2.2.2.4. Set Request

A set request PDU sets the specified attributes and returns a set response PDU back to the user. This PDU is not initially supported. The syntax of a set request PDU is as follows:

```
SetRQ ::= Sequence  §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   [2] AccessControl DEFAULT §Nullt,
   [3] Implicit ParameterList  †


ResourceID ::= Choice  §

   Private ResourceID [0] Any,
   [1] Implicit LayerInfo †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §Nullt,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †


Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


Layer Instance ::= Implicit integer

      16 bit integer. It is equivalent to
      the controller number it resides on
      and the index to the layer instance
      table.

LayerInternalSelector ::= Any  §
```

PrivateInternalSelector [0] Implicit sequence of
SelectionParameters †

[0] SelectionParameters ::= Sequence §

[0] Class ::= Implicit Integer §

16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

[1] Name ::= Implicit IA5 string DEFAULT §Null†

4 Octets. Identifies a unique system name of an object. Always 8 ASCII characters, left justified and null padded. Default is 8 null characters.

[2] ObjectState ::= Implicit Sequence §

[0] Implicit State,
[1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

Anytype	(0000)
LACS Controller	(LNCT)
IEEE 802.1 SM	(8021)
IEEE 802.2 LLC	(8022)
IEEE 802.3 MAC	(8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue	(0)
Local	(1)
Image	(2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length.
The first octet specifies class and the second octet specifies length of the object name. The last 8
octets
indicate the object or component name to which the object is mapped. The name is always 8
ASCII
characters, left justified and null padded.
Not initially supported. †

Exchangeld ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. The exchangeid will be returned in the response PDU.

AccessControl ::= Implicit Octetstring

The current implementation requires that the octestring for access control consist of 2 octets and consist of all nulls.
Not initially supported.

```
ParameterList ::= Set of Parameter

Parameter ::= Choice §

    [0] Implicit Status,
    Defined Parameter [1] Any  †
```

2.2.2.5. Set Response

A set response PDU returns the status of each attribute set or error information about the set request. This PDU is not initially supported. The syntax of a set response PDU is as follows:

```
SetRSP ::= Sequence §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   Choice §

      [2] Implicit Status,
      [3] Implicit ParameterList  ††


ResourceID ::= Choice  §

   PrivateResourceID [0] Any,
   [1] Implicit Layer Info  †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §Nullt,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †


Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


Layer Instance ::= Implicit integer

         16 bit integer. It is equivalent to
         the controller number it resides on
         and the index to the layer instance
         table.
```

```
LayerInternalSelector ::= Any  §

   PrivateInternalSelector [0] Implicit sequence of
                               SelectionParameters  †


   [0] SelectionParameters ::= Sequence  §


   [0] Class ::= Implicit Integer  §

       16  bit integer. Class identifies the
       type of object within a layer.

   Controller   (13)
   SMAF         (15)
   PhysicalLine (4)
   LogicalLine  (5)  †

   [1] Name ::= Implicit IA5 string  DEFAULT §Null†

       4 Octets. Identifies a unique  system
       name  of  an  object.  Always 8 ASCII
       characters, left justified  and  null
       padded.    Default    is    8    null
       characters.

   [2] ObjectState ::= Implicit Sequence  §

          [0] Implicit State,
          [1] Implicit Substate  †

   State ::= Integer  §

       8   bit  integer.  Describes major state.
       Default to anystate.

   AnyState (0)
   Down     (2)
   Locked   (3)
   Disabled (4)
   Enabled  (6)
   In-Use   (7)
   Test     (8)
   Shutdown (9)  †
```

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

Anytype	(0000)
LACS Controller	(LNCT)
IEEE 802.1 SM	(8021)
IEEE 802.2 LLC	(8022)
IEEE 802.3 MAC	(8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue (0)
Local (1)
Image (2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length.
The first octet specifies class and the second octet specifies length of the object name. The last 8

octets

indicate the object or component name to which the object is mapped. The name is always 8

ASCII

characters, left justified and null padded.
Not initially supported.

Exchangeld ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. It is the same as the exchangeid in the request PDU.

Status ::= Sequence §

[0] Implicit StatusCode,
[1] StatusInfo †

StatusCode ::= Integer §

success (0)
bad layer internal selector (6)
bad parameter id (7)
TBS †

StatusInfo ::= Any §

PrivateStatus Info [0] ::= Implicit Octetstring

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification. †

ParameterList ::= Set of Parameter

Parameter ::= Choice §

[0] Implicit Status,
DefinedParameter [1] Any †

The syntax and description of the the parameter ids can be found in section x.x.x.x containing the description of all attributes.

2.2.2.6. Action Request

An action request PDU performs the specified action and returns a action response PDU back to the user. The syntax of a action request PDU is as follows:

```
ActionRQ ::= Sequence  §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   [2] Access Control DEFAULT §NulIt,
   [3] Implicit Action  †


ResourceID ::= Choice  §

   Private ResourceID [0] Any,
   [1] Implicit LayerInfo †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §NulIt,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †


Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


Layer Instance ::= Implicit integer

        16 bit integer. It is equivalent to
        the controller number it resides on
        and the index to the layer instance
        table.
```

LayerInternalSelector ::= Any §

PrivateInternalSelector [0] Implicit sequence of
SelectionParameters †

[0] SelectionParameters ::= Sequence §

[0] Class ::= Implicit Integer §

16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

[1] Name ::= Implicit IA5 string DEFAULT §Null†

4 Octets. Identifies a unique system name of an object. Always 8 ASCII characters, left justified and null padded. Default is 8 null characters.

[2] ObjectState ::= Implicit Sequence §

[0] Implicit State,
[1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

```
AnySubState (0)
Reset       (1)
Halted      (2)
Loaded      (3)
Started     (4)
Operational (5)  †
```

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

```
Anytype         (0000)
LACS Controller (LNCT)
IEEE 802.1 SM   (8021)
IEEE 802.2 LLC  (8022)
IEEE 802.3 MAC  (8023)
```

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

```
AnyVenue (0)
Local    (1)
Image    (2)
```

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length.
The first octet specifies class and the second octet specifies length of the object name. The last 8 octets indicate the object or component name to which the object is mapped. The name is always 8 ASCII characters, left justified and null padded.
Not initially supported. †

ExchangeId ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. The exchangeid will be returned in the response PDU. It is equivalent to the command number.

AccessControl ::= Implicit Octetstring

The current implementation requires that the octetstring for access control consist of 2 octets and consist of all nulls.
Not initially supported.

Action ::= Any §

PrivateHoneywellAction [0] ::= Sequence §

[0] Code,
[1] HoneywellActionInfo †

Code ::= Integer §

List All (9)
Update State (5)
Load (52)
Dump (53)
Create (xx)
Test (xx) †

HoneywellActionInfo ::= Choice §

[0] ListAllInfo
[1] UpdateStateInfo
[2] LoadInfo
[3] DumpInfo
[4] CreateInfo
[5] TestInfo †

ListAllInfo ::= Implicit Class §
16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

UpdateStateInfo ::= Implicit Sequence §

[0] Implicit RequestedState,
[1] Implicit RequestedSubstate,

RequestedState ::= Integer §

Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

RequestedSubstate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

LoadInfo ::= Implicit Sequence §

[0] BUPathname ::= Implicit IA5 string DEFAULT §Null†

Full pathname of a file containing a LACS bound unit. If the default is specified, the chosen pathname is >>>SID>LACSB.U. Variable length string.

[1] RestartInd ::= Implicit IA5 String

1 octet. A restart indication indicates that load should be executed, a no restart indication will indicate no action other than loading memory will occur.

Restart (RS),
No Restart (NR)

[2] StartAddress ::= Implicit integer DEFAULT §Nullt

32 bit integer. If specified, the start address will be used as the point to begin program execution rather than the address specified in the LACS bound unit. Not initially supported. †

Load action is only supported by the controller object.

DumpInfo ::= Implicit Sequence §

[0] DumpTS ::= Implicit IA5 string

Text string to append to file which is to contain memory dump. Variable length. Not supported initially.

[1] DumpPathname ::= Implicit IA5 string DEFAULT §Nullt

Full pathname of a file to dump LACS memory formatted as a LACS bound unit. If the default value is specified the pathname of the dump file chosen is TBS.

[2] LowAddress ::= Implicit integer

32 bit integer. Specifies the LACS address from which to start the dump of memory.

[3] HighAddress ::= Implicit integer

32 bit integer. Specifies the last LACS address from which to end the dump of memory. †

Dump action is only supported by the controller object.

CreateInfo ::= Implicit ParameterIdList

Each object or component is created with the attributes described in the parameteridlist specified for a create operation. The state of the object or component is defaulted to the locked state unless specified in the parameteridlist. Other default values are specific to an object or component. Refer to section xxxxxxxx for more details on the creation of an object or component.

```
TestInfo ::= Sequence of §

    TestParameter ::= Implicit Integer
                16 bit integer.
    TestData ::= Choice  §

   [0] ManagementTestData
   [1] CSMAPhysicalTestData
   [2] CSMAMACTestData
   [7] LLCTestData
   [xx] InternetTestData
   [xx] SDNACPTestData
   [xx] ISO TransportTestData
   [xx] ControllerTestData  †
                          †
```

The tests which can be performed for each object or component are described in more detail in the Test and Verification specification, xxxxxxx, section xxxxxxxxx.

2.2.2.7. Action Response

An action response PDU returns the results of each action or error information about the action request. The syntax of an action response PDU is as follows:

```
ActionRSP ::= Sequence  §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   Choice §

      [2] Implicit Status,
      [3] ActionReport  ††


ResourceID ::= Choice  §

   Private ResourceID [0] Any,
   [1] Implicit LayerInfo †


LayerInfo ::= Sequence  §

   [0] Implicit Layer,
   [1] Implicit Sublayer,
   [2] Implicit LayerInstance DEFAULT §Null†,
   [3] Implicit LayerInternalSelector  †

Layer ::= Integer  §

   management (0)
   physical   (1)
   link       (2)
   network    (3)
   transport  (4)  †


Sublayer ::= Integer §

   MAC (0),
   LLC (1)  †


Layer Instance ::= Implicit integer

        16 bit integer. It is equivalent to
        the controller number it resides on
        and the index to the layer instance
        table.
```

LayerInternalSelector ::= Any §

PrivateInternalSelector [0] Implicit sequence of
SelectionParameters †

[0] SelectionParameters ::= Sequence §

[0] Class ::= Implicit Integer §

16 bit integer. Class identifies the type of object within a layer.

Controller (13)
SMAF (15)
PhysicalLine (4)
LogicalLine (5) †

[1] Name ::= Implicit IA5 string DEFAULT §Null†

4 Octets. Identifies a unique system name of an object. Always 8 ASCII characters, left justified and null padded. Default is 8 null characters.

[2] ObjectState ::= Implicit Sequence §

[0] Implicit State,
[1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

Anytype (0000)
LACS Controller (LNCT)
IEEE 802.1 SM (8021)
IEEE 802.2 LLC (8022)
IEEE 802.3 MAC (8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue (0)
Local (1)
Image (2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length.
The first octet specifies class and the second octet specifies length of the object name. The last 8 octets indicate the object or component name to which the object is mapped. The name is always 8 ASCII characters, left justified and null padded.
Not initially supported.

```
Exchangeld ::= Implicit Octetstring

   The    present    implementation
   requires the exchangeid value  to  be
   no   greater  than  2  octets.  It is
   the same as the exchangeid in the
   request PDU.


Status ::= Sequence  §

   [0] Implicit StatusCode,
   [1] StatusInfo  †


StatusCode ::= Integer  §

   success                        (0)
   bad layer internal selector    (6)
   bad action                     (7)
   TBS  †


StatusInfo ::= Any  §

   PrivateStatus Info [0] ::= Implicit Octetstring

   The    present    implementation
   requires a two octet record which  is
   equivalent   to   the   reason   code
   defined in each layer component specification. †

ActionReport ::= Choice  §

   [0] Status,
   DefinedReport [1] Any  †


PrivateHactionReport [0] ::= Sequence  §

   [0] Code,
   [1] HoneywellActionReport  †

 Code ::= Integer §

   List All      (9)
   Update State  (5)
   Load          (52)
   Dump          (53)
   Create        (xx)
   Test          (xx)  †
```

```
  PrivateHoneyeilActionReport [0] ::= Choice  §

   [0] ListAllReport,
   [1] UpdateStateReport,
   [2] LoadReport,
   [3] DumpReport,
   [4] CreateReport,
   [5] TestReport  †

ListAllReport ::= Choice §

   [0] Status,
   [1] Set of ListAllInfo  †

   Status ::= Implicit OctetString

   The    present    implementation
   requires a two octet record which  is
   equivalent   to   the   reason   code
   defined in each layer component specification.


   Set of ListAllInfo ::= Implicit Sequence  §

      [0] Selection Parameters,
      [1] Layer Instance  †


   [0] SelectionParameters ::= Sequence  §

   [0] Class ::= Implicit Integer  §

       16  bit integer. Class identifies the
       type of object within a layer.

   Controller   (13)
   SMAF         (15)
   PhysicalLine (4)
   LogicalLine  (5)  †

   [1] Name ::= Implicit IA5 string  DEFAULT §Null†

       4 Octets. Identifies a unique  system
       name  of  an  object.  Always 8 ASCII
       characters, left justified  and  null
       padded.   Default   is   8   null
       characters.
```

[2] ObjectState ::= Implicit Sequence §

    [0] Implicit State,
    [1] Implicit Substate †

State ::= Integer §

8 bit integer. Describes major state. Default to anystate.

AnyState (0)
Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

Substate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

[3] Type ::= Implicit IA5 string

2 octets. Identifies a particular type of object within a class

Anytype (0000)
LACS Controller (LNCT)
IEEE 802.1 SM (8021)
IEEE 802.2 LLC (8022)
IEEE 802.3 MAC (8023)

[4] Venue ::= Implicit Integer

8 bit integer. Specifies whether object is a local or image of a remote object.

AnyVenue (0)
Local (1)
Image (2)

[5] Mappings ::= Implicit Octetstring

This octet string is 10 octets in length. The first octet specifies class and the second octet specifies length of the object name. The last 8 octets indicate the object or component name to which the object is mapped. The name is always 8 ASCII characters, left justified and null padded. Not initially supported. †

[1] Layer Instance ::= Implicit Integer

16 bit integer. It is equivalent to the controller number it resides on and the index to the layer instance able.

UpdateStateReport ::= Sequence §

[0] Status,
[1] Implicit RequestedState,
[2] Implicit RequestedSubstate †

Status ::= Implicit OctetString

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification.

RequestedState ::= Integer §

Down (2)
Locked (3)
Disabled (4)
Enabled (6)
In-Use (7)
Test (8)
Shutdown (9) †

RequestedSubstate ::= Integer §

8 bit integer. Describes minor LAN state. Default to anysubstate.

AnySubState (0)
Reset (1)
Halted (2)
Loaded (3)
Started (4)
Operational (5) †

The states for each object or component are described in more detail in section xxxxxxxxx.

LoadReport ::= Implicit Status

Status ::= Implicit OctetString

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification.

DumpReport ::= Implicit Status

Status ::= Implicit OctetString

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification.

CreateReport ::= Implicit Status
Status ::= Implicit OctetString

The present implementation requires a two octet record which is equivalent to the reason code defined in each layer component specification.

TestReport ::= Sequence of §

TestParameter ::= Implicit Integer
16 bit integer.
TestReportData ::= Choice §

```
        [0] ManagementTestReportData
        [1] CSMAPhysicalTestReportData
        [2] CSMAMACTestReportData
        [7] LLCTestReportData
        [xx] InternetTestReportData
        [xx] SNDCPTestReportData
        [xx] XPORTTestReportData
[xx] ISO TransportTestReportData
[xx] ControllerTestReportData   †
                     †
```

The tests which can be performed for each object or component are described in more detail in the Test and Verification specification, xxxxxxx, section xxxxxxxxx.

2.2.2.8 Event PDU format

Event PDUs are issued from system management to a user for any unsolicated event occuring during LAN operation. The syntax for an event PDU indication is as follows:

```
EventPDU ::=  Sequence of EventInfo  §

EventInfo ::= Sequence  §

   [0] ResourceID,
   [1] Implicit ExchangeID,
   [2] Implicit EventTime  DEFAULT §Nullt,
   [3] Implicit AcknowledgedRequired DEFAULT §Falset,
   [4] Implicit LayerEventInfo  tt
```

Exchangeld ::= Implicit Octetstring

The present implementation requires the exchangeid value to be no greater than 2 octets. The exchangeid will be two octets containing all zeroes in an event PDU.

```
EventTime ::= Set  §

   [0] = Implicit GeneratlizedTime,
   [1] = Implicit LocalTime  t
```

The present implementation requires the event time must always be set to null values.

AcknowledgedRequired ::= Boolean

1 octet. This boolean must always be set to indicate an event acknowledement is not required (False = FF (hex))

```
LayerEventInfo ::= Any  §

  Private LayerEventInfo [0] ::= Sequence  §

   [0] Status,
   [1] SystemSpecificInfo  t
```

Status ::= Implicit OctetString

The present implementation requires
a two octet record which is equivalent
to the reason code in each layer component specification.

SystemSpecific Info ::=

Information which supplements the
reason code given above. If not
applicable, a value of zero is used.

2.2.3 LAN Control Blocks (LCB)

The interface between the L6 system management and the LACS system management function is through the LDMS routine and will utilize LAN Control Blocks (LCB). These blocks are transferred across the megabus to the LACS board by signaling the board resident software with IOLD instructions. That software will nhen transfer an image of the LCB to its own memory via DMA transfers from the L6 memory. Not all of the LCB is transferred across the megabus to the LACS board. The control block is divided into two parts, that portion used by both L6 and LACS software and that portion used by L6 software only. The L6 unique portion of the block begins at a displacement of zero and expands in the positive direction. The common L6 and LACS required area begins at offset xx. The common L6 and LACS portion is further subdivided into a standard LAN part and a function specific part. The format of a SM LCB is as follows:

The LCB is used by the LAN subsystem to pass information from the L6 SM server across the megabus to the lacs SM layer instance software. The IORB has an extension that is used by the sap driver as the LCB.

Structure layout for LCBs are as follows:

the following fields are L6 specific

cb_pri priority queuing value
contains priority queuing value

cb_ncb next LCB pointer
initialized by sm is
contains pointer to next LCB in queue, initial value is null

cb_rct rct pointer
initialized by sm is
contains pointer to the rct

cb_lit lit pointer
initialized by sm is
contains pointer to lit

cb_fss function specific status
    initialized by lacs software
    contains function specific status
        bit 1 - LCB was aborted when set
    referenced by sm ls

cb_cbs completion word
    initialized by sm ls
    contains completion word, number of buffers
        bit 0 - LCB is complete when set
        bit 1 - buffer is too small when bit 0 is also set

cb_trg total number of bytes
    initialized by lacs software
    contains total number of bytes available for transfer

cb_asz actual size of buffer
    set by lacs software
    contains actual size of buffer when status indicates the specified buffer was too small

cb_nbf number of buffers
    initialized by sm ls
    contains number of buffers

The following fields are repeated the number of times represented by the right byte of the cd_nbf field (up to eight times). Those fields not used must be nulled.

cb_adr0 level 6 buffer byte address
    initialized by sm ls
    contains level 6 buffer address in bytes

cb_rng0 range word
    initialized by sm ls
    contains range in bytes of the cb_adr0 field

cb_adr1 level 6 buffer byte address
    initialized by sm ls
    contains level 6 buffer address in bytes

cb_rng1 range word
    initialized by sm ls
    contains range in bytes of the cb_adr1 field

cb_adr2 level 6 buffer byte address
    initialized by sm ls
    contains level 6 buffer address in bytes

cb_rng2 range word
    initialized by sm ls
    contains range in bytes of the cb_adr2 field

cb_adr3 level 6 buffer byte address
    initialized by sm ls

contains level 6 buffer address in bytes

cb_rng3 range word
initialized by sm ls
contains range in bytes of the cb_adr3 field

cb_adr4 level 6 buffer byte address
initialized by sm ls
contains level 6 buffer address in bytes

cb_rng4 range word
initialized by sm ls
contains range in bytes of the cb_adr4 field

cb_adr5 level 6 buffer byte address
initialized by sm ls
contains level 6 buffer address in bytes

cb_rng5 range word
initialized by sm ls
contains range in bytes of the cb_adr 5field

cb_adr6 level 6 buffer byte address
initialized by sm ls
contains level 6 buffer address in bytes

cb_rng6 range word
initialized by sm ls
contains range in bytes of the cb_adr6 field

cb_adr7 level 6 buffer byte address
initialized by sm ls
contains level 6 buffer address in bytes

cb_rng7 range word
initialized by sm ls
contains range in bytes of the cb_adr7 field

2.2.3.2 START I/O LCB FORMAT

The format of the Start I/O is defined in the H/W EPS and repeated here only to aid understanding of the system manager layer server. The Start I/O LCB must be issued with an IOLD instruction, function code xxxx, and a protocol id of xxxx. The following fields are LACS specific:

cb_icw interrupt control word
initialized by sm ls
contains interrupt control word
bits 0-5 - rsu and mbz
bits 6-9 - cpu number to interrupt

bits a-f - level to interrupt the cpu
referenced by: lacs megabus interface software

cb_fsf function code
initialized by sm ls
must be zero

cb_cts controller status
initialized by lacs software
contains controller status
bit 8 - invalid function code when set
bit 9 - ram memory exausted when set
bit a - ram location non-existent when set
bit b - ram parity error when set
bit c - level 6 memory yellow when set
bit d - level 6 memory non-existent when set
bit e - level 6 bus parity error when set
bit f - level 6 memory red when set
referenced by sm ls

cb_fss function specific status
must be zero

cb_cbs completion word
initialized by sm ls
contains completion word, number of buffers
bit 0 - LCB is complete when set

cb_str LACS ecection starting address
set by lacs software
contains physical address in LACS to begin execution.

cb_ext total number of bytes
initialized by SM server
contains total number of bytes involved in transfer

cb_dta Start I/O data address
set by lacs software
contains address of start I/O information concerning the creation of LACS processes.

2.2.3.3 LOAD/DUMP LCB FORMAT

The format of the Load/Dump is defined in the H/W EPS and repeated here only to aid understanding of the system manager layer server. The Load/Dump LCB must be issued with an IOLD instruction, function code xxxx, and a protocol id of xxxx. The following fields are LACS specific:

cb_icw interrupt control word
initialized by sm ls

```
cb_icw  interrupt control word
    initialized by sm ls
    contains interrupt control word
            bits 0-5 - rsu and mbz
            bits 6-9 - cpu number to interrupt
            bits a-f - level to interrupt the cpu
    referenced by:  lacs megabus interface software

cb_fsf  function code
    initialized by sm ls
    contains channel specific function code
                    D - Update current TicToc count

cb_cts  controller status
    initialized by lacs software
    contains controller status
            bit 8 - invalid function code when set
            bit 9 - ram memory exausted when set
            bit a - ram location non-existent when set
            bit b - ram parity error when set
            bit c - level 6 memory yellow when set
            bit d - level 6 memory non-existent when set
            bit e - level 6 bus parity error when set
            bit f - level 6 memory red when set
    referenced by sm ls

cb_fss  function specific status
    must be zero

cb_cbs  completion word
    initialized by sm ls
    contains completion word, number of buffers
            bit 0 - LCB is complete when set

cb_cnt  Current TicToc count maintained by the LACS.
    set by lacs software
    contains actual count.
```

```
        contains interrupt control word
                bits 0-5 - rsu and mbz
                bits 6-9 - cpu number to interrupt
                bits a-f - level to interrupt the cpu
        referenced by:  lacs megabus interface software

    cb_fsf  function code
        initialized by sm ls
        contains channel specific function code
                        0 - Dump LAC RAM to memory
                        1 - Load LAC RAM from memory
                        2 - Dump  Configuration  information  from
                            LAC RAM to memory

    cb_cts  controller status
        initialized by lacs software
        contains controller status
                bit 8 - invalid function code when set
                bit 9 - ram memory exausted when set
                bit a - ram location non-existent when set
                bit b - ram parity error when set
                bit c - level 6 memory yellow when set
                bit d - level 6 memory non-existent when set
                bit e - level 6 bus parity error when set
                bit f - level 6 memory red when set
        referenced by sm ls

    cb_fss  function specific status
        must be zero

    cb_cbs  completion word
        initialized by sm ls
        contains completion word, number of buffers
                bit 0 - LCB is complete when set

    cb_adr  L6 address
        set by lacs software
        contains physical buffer byte address in L6 memory


    cb_ext  total number of bytes
        initialized by SM server
        contains total number of bytes involved in transfer

    cb_ram  Ram address
        set by lacs software
        contains physical byte address in LACS RAM.
```

2.2.3.5 CONTROLLER TICTOC LCB FORMAT

The format of the TicToc is defined below. The Load/Dump LCB must be issued with an IOLD instruction, function code xxxx, and a protocol id of xxxx. The following fields are LACS specific:

2.2.4 L6 LAYER MANAGEMENT INTERFACE

Layer management functions are split between the L6 and the LACS board in a similar fashion as System management. It is the responsibility of system management to make requests to either or both of the functions. In general system management will make a request against the layer management function within the L6 and combine it with the results obtained from the LACS system management layer instance. Each layer instance table contains the address of the layer management routine to service a request on a layer instance. A request is made to layer manager by passing the address of a buffer containing a service primitive request. The SM server lunges to L6 layer mangagement routines, passing a pointer to a request block. The layer manager returns any results as a confirm in the same memory allocated for the request block. Events are issued to the SM server by a layer management routine lunging to the SM event indicate routine with a pointer to an event block. Initially L6 layer management functionality is required of system management for the controller object and the system management administrative function. Refer to the component specification for each layer server for more detailed information. The format of the request and confirm blocks is as follows:

Each request block contains information concerning the operation to be performed and identifies the object or component within the layer to perform the operation on. Each operation will describe operation information required to perform the operation. The confirm block to a request will be identical except for the status and confirm information pointer. The format of the block information fields for request and confirm operations is as follows:

0	15
layer internal selector	
status	
operation code	
size of operation information	
operation information pointer	

layer internal selector
status
operation code
size of operation information
operation information pointer

The layer internal selector describes the selection parameters used by the layer manager to determine which object or component within the layer this operation is to be performed on. It contains the following fields:

name 4 words
8 ASCII characters. Null when not to be used. Set by the SM server.

class 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero. Unique class numbers have been assigned for each type of object and component across all layers. Set by the SM server.

Controller	(13)
System Management Administrative Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)
Transport Connection	(xx)

type 2 words

4 ASCII characters. Unique type is assigned for each class of object. Set by the SM server.

LACS Controller	(LNCT)
SMAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	(SNCP)
TSAP	(CLS4)

venue 1 word

8 bit integer in bits 8-F. Bits 0-7 must be zero. Set by the SM server..

any	(0)
local	(1)
image	(2)

object state 1 word. Set by the SM server.

state Bits 0-7. 8 bit integer

any	(00)
locked	(03)
enabled	(04)
disabled	(05)
test	(06)
down	(07)
shutdown	(08)
in-use	(09)

substate Bits 8-F. 8 bit integer.

any	(00)
reset	(01)
halted	(02)
loaded	(03)
started	(04)
operational	(05)

The internal layer selector must be matched by the layer management function before the operation is carried out. If a match is not located than the layer management function must return status to indicate which selection parameter did not match.

Status contains two fields, the status code and a pointer to status info. The status code will identify the particular source of the status and a status id indicating success or the reason for failure. Additional status information may be specified by using the memory block allocated for operation information. In that case, status info is any operation specific status information. The first word of status info indicates the size of the buffer containing status info (including the status info size word) followed by a pointer to status info data (this must be a pointer to the same memory passed as part of the operation information). Where there is no additional status infromation, these fields will be null. The format of the status field is as follows:

Status code 1 word

Source: Bits 0-7. 8 bit integer. Set by layer management.

management	(0)
physical(MAC)	(1)
link(LLC)	(2)
network	(3)
transport	(4)

Status ID: Bits 0-7. 8 bit integer, unique values assigned for each layer. Set by layer management.

StatusInfoIng 1 word

16 bit integer, indicates the number of bytes contained in status info data. Set by layer management.

StatusInfoPtr 2 words

Pointer to additional layer specific information. Must point to memory allocated in operation information. Set by layer management.

The operation code specifies one of three possible operations, LM_GET_VALUE service primitive, LM_SET_VALUE service primitive, and LM_ ACTION service primitive. The operation information field for each operations request or confirm is unique to the operation. The operations for the LAN layer management functions are described as follows:

operation code 1 word

8 bit integer in bits 8-F. Bits 0-7 must be zero. Set by SM server. Values for layer management operations are:

LM_GET_VALUE	(01)
LM_SET_VALUE	(02)
LM_ACTION	(04)

operation info size 1 word

16 bit integer. Set by the SM server.

operation info pointer 2 words

pointer to operation information. The operation information is defined unique to each operation. The memory pointed to by this pointer is allocated by system management.

LM_GET_VALUE service primitive - Operation code 1

Read the specified attributes of an object or component within the layer. The format for a LM_GET_VALUE service primitive operation information field is as follows:

0 15
parameter id
parameter size
parameter list

The operation information field for a Get operation specifies the value of the parameter id corresponding to the parameter desired. Only one parameter id may be specified. The confirm to a get request must contain a status with respect to the completion of the operation and the value of the parameter. The parameter size specifies the number of bytes containing the parameter value. The parameter list contains the value of the parameter or a list of values containing all Honeywell attributes or all Honeywell statistics in an IEEE802 defined format for a private parameter id. Refer to the proper layer specification for more information on a particular layers confirm. The parameter id's for requesting Honeywell parameters are

parameter id — 1 word. 16 bit integer. set by SM server.

All Attributes	(xx)
All Statistics	(xx)

- see layer component specifications for return parameter id and parameter list formats.

parameter size — 1 word. 16 bit integer. Number of bytes in parameter value. Set by the SM server and layer management.

The values of the status id returned in the confirm block can be

Success	(00)
Not supported	(xx)
Bad parameter id	(xx)
Bad layer internal selector	(xx)
TBU	

LM_SET_VALUE service primitive - Operation code 2

Set the specified attributes of an object or component within the layer. Only a single attribute may be specified per LM_ SET_ VALUE service primitive request. The format of a LM_SET_VALUE service primitive operation is as follows:

parameter id
parameter size
parameter value

The operation information field for a LM_SET_VALUE service primitive operation specifies the value of the parameter id corresponding to the parameter desired. Only one parameter id may be specified. The parameter size specifies the number of bytes containing the parameter value. The parameter value will contain the new value for the parameter. The confirm block will contain status on completion of the operation. Refer to the proper layer specification for more information on a particular layers parameter id's.

parameter id	1 word. 16 bit integer. set by SM server.
parameter size	1 word. 16 bit integer. Number of bytes in parameter value. Set by the SM server and layer management.

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad parameter id	(xx)
Bad layer internal selector	(xx)
Bad parameter value	(xx)

TBU

LM_ACTION service primitive - Operation code 4

Action provides the ability to perform layer specific functions as well as operations common to all layers. There are four common Action requests which are supported by all layers, Update state, Create, List, and Test. While each layer performs these action operations in a unique fashion, they are a common set of requests to any layer management function. Update state instructs layer management functions to change the state of the specified object or component. Create provides the ability to create a new object or component within a layer. The list action operation causes the layer management to return a list of all objects or components of the specified type. The test action operation can only be performed on objects and components in the locked state and cause the execution of a unique layer test procedure. The operation information field for each action operation is unique except for the action operation field and described below.

action operation 1 word.

8 bit integer in bits 8-F. Bits 0-7 must be zero. Set by SM server. Values for action operations are:

UPDATESTATE	(01)
LIST	(00)
LOAD	(02)
DUMP	(03)

UPDATE STATE

Update state provides the ability to control the state of a component or object within a layer. The definition of the states for each layer can be found in appendix xxxxxx, of the SW EPS or the layer component spec. The format of the operation information field for an update state action request is as follows:

```
______________________________________________
¶                                            ¶
¶        Action operation                    ¶
¶____________________________________________¶
¶                                            ¶
¶        New object state                    ¶
¶        (State¶Substate)                    ¶
¶____________________________________________¶
```

The action operation field must specify the update state operation. The state and substate immediately follow.

update state action operation (xx)

```
new object state        1 word.  Set by the SM server.
    newstate    Bits 0-7. 8 bit integer
        locked      (03)
        enabled     (04)
        disabled    (05)
        test        (06)
        down        (07)
        shutdown    (08)
        in-use      (09)
    newsubstate Bits 8-F. 8 bit integer.
        reset       (01)
        halted      (02)
        loaded      (03)
        started     (04)
        operational (05)
```

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
Bad state	(xx)
Bad substate	(xx)
Illegal state change	(xx)

TBU

LIST

The LIST action operation provides the ability to obtain a list of all objects or components which meet the internal layer selection criteria. The confirm contains name, class, type, venue and state for every object meeting the criteria. The format of an action list operation is as follows:

Action operation
Number of objects
Object 1 selection parameters
. . .
Object n selection parameters

The action operation field must specify the LIST action operation. No additional information is required for a LIST request. The layer manager will return the selection parameters for all defined objects.

list action operation (xx)

number of objects 1 word.
16 bit integer. Set by the layer manager.

for each object listed

name 4 words
8 ASCII characters. Set by the layer server.

class 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero. Unique class numbers have been assigned for each type of object and component across all layers. Set by the layer server.

Controller	(13)
System Management Admin Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)

type 2 words
4 ASCII characters. Unique type is assigned for each class of object. Set by the layer server.

LACS Controller	(LNCT)
layerAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	(SNCP)
TSAP	(CLS4)

venue 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero. Set by the layer server.

any	(0)
local	(1)
image	(2)

object state 1 word. Set by the layer server.
state Bits 0-7. 8 bit integer

locked	(03)
enabled	(04)
disabled	(05)
test	(06)
down	(07)
shutdown	(08)
in-use	(09)

substate Bits 8-F. 8 bit integer.

reset	(01)
halted	(02)
loaded	(03)
started	(04)
operational	(05)

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
TBCU	

TEST

The test action operation provides the ability to perform layer specific tests. The definition of such tests can be found for each layer can be found in section xxxxxxx, of document, xxxxx. The format of the operation information field for an update state action request is as follows:

Action operation
Test parameter
Length of Test Information
Test Information

The action operation field must specify the test operation. The test parameter is provided for every test as well as test information. The nature and definition of this information is contained in reference xxxxxxxxxx.

test action operation	(xx)
test parameter	1 word 16 bit integer. Set by SM server.
length of test info	1 word 16 bit integer. Set by SM server and the layer manager. THe size returned in the confirm must not exceed that passed in the request.
test info	Unique for each test operation.Set by SM server and the layer manager.

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
Bad test parameter	(xx)
Bad test	(xx)
TBCU	

For more information on the available commands to each layer, refer to the proper entity specification.

EVENT BLOCKS

Event blocks received by system management are of the following format:

TBD

0	15
Layer Info (layer/sublayer)	
Layer Instance (CT# and LI#)	
layer internal selector	
Event status	

The layer info and layer insatance are used to identify the layer server which generated the event.

Layer Info — 1 word.

- layer — Bits 0-7. 8 bit integer. Value restricted between 0-7.
  - Management (0)
  - Physical (1)
  - Link (2)
  - Network (3)
  - Transport (4)
- sublayer — Bits 8-F. 8 bit integer. Value restricted between 0-1. For Link only, all other layers will have a zero value.
  - MAC (0)
  - LLC (1)

Layer Instance 1 word.

- Controller Number — Bits 0-7. 8 bit integer. Value restricted between 0-F.
- Layer Instance Number — Bits 8-F. 8 bit integer. Value restricted between 0-7.

The layer internal selector describes the selection parameters of the object or component within the layer this event was involved. It contains the following fields:

name 4 words
8 ASCII characters.

class 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero. Unique class numbers have been assigned for each type of object and component across all layers.

Controller	(13)
System Management Administrative Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)
Transport Connection	(xx)

type 2 words
4 ASCII characters. Unique type is assigned for each class of object.

LACS Controller	(LNCT)
SMAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	(SNCP)
TSAP	(CLS4)

venue 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero.
local (1)

object state 1 word.
state Bits 0-7. 8 bit integer

locked	(03)
enabled	(04)
disabled	(05)
test	(06)
down	(07)
shutdown	(08)
in-use	(09)

substate Bits 8-F. 8 bit integer.

reset	(01)
halted	(02)
loaded	(03)
started	(04)
operational	(05)

Event status contains two fields, the event status code and event status info. The event status code will identify the particular source of the event and a event status id indicating the reason for event. Additional status information may be specified by event status info. Event status info is any event specific status information. The first word of event status info indicates the size of the buffer containing event status info (including the status info size word) followed by event status info data. The format of the event status field is as follows:

EventStatus code 1 word

Eventsource: Bits 0-7. 8 bit integer. Set by layer management.

management	(0)
physical(MAC)	(1)
link(LLC)	(2)
network	(3)
transport	(4)

EventStatus ID: Bits 0-7. 8 bit integer, unique values assigned for each layer. Set by layer management.

EventStatusInfolng 1 word

16 bit integer, indicates the number of bytes contained in event status info data. Set by layer management.

EventStatusInfoPtr 2 words

Pointer to additional layer specific information. Must point to memory allocated in operation information. Set by layer management.

2.2.5 MOD400 EXECUTIVE SOFTWARE ROUTINES

2.2.5.1 ZHCOMM - Null address

function: Will load the null address when referenced i.e. ldb $b5,<zhcomm will load $b5 with the null address.

2.2.5.2 ZXD_TR - Internal terminate

entry: lnj $b5,zxd_tr

input: $r2 = completion status for request
$b4 = new default start address or null
$b5 = return

output: $r1 = 0 - no error on internal terminate
$b1 = address of IRB for next request or is null if hardware interrupt
$b4 = address of RB for next request

modifies: any register may be modified

function: Dequeue and post currently dispatched request. Get next request in queue of task requests. Delete task if queue empty and delete bit on. Suspend task until next request or hardware interrupt at issuing task's level if queue empty and delete bit off.

2.2.6 MOD400 EXTERNAL DATA STRUCTURES IMPLEMENTED

The following system owned data structures are referenced by the SM server:

Task Control Block (TRB)
System Control Block (SCB)
Logical Resource Table (LRT)
Group Control Block (GCB)
Resource Control Table (RCT)
Intermediate Request Block (IRB)

2.2.6 LACS DRIVER INTERFACE SERVICES ROUTINE USED

The SM server uses the following ldis routines, the definition of the input and output parameters can be found in the ldis component specification:

isblcb - build lcb
isvlob - validate iorb
istmtk - terminate task
isevnt - event routines
isasvb - assign segiment visiblity
isabsl - absoultize buffers

2.2.7 LACS DRIVER MEGABUS SERVICES ROUTINES USED

The 802 llc is uses the following ldms routines, the definition of the input and output parameters can be found in the ldms compnent specification.

msiior - issue iold or issue io routine

2.3 INITIALIZATION REQUIREMENTS

Two types of CLM directives must be defined for the system management layer server. A Layer Instance directive containing a symbolic name, layer type, layer instance number, interrupt level and hardware masks must be defined for the SM server. Additionally a Local User directive specifying a symbolic name and task level for a user interface to the system management server as well as a Local User directive specifying the sysmbolic name of the T&V interface. All local users of system management must run at the same task level. These three directives must always be present for operations of the LAN. It is recommended that only two Local User directives (T&V and 1 other user) be defined for system management in any LAN configuration.

A user of system management must follow the same procedures as a user of other layer servers to initialize its local interface. A user must first execute an associate local user monitor call (ALU) to obtain an LRN for the system management interface. The user must then issue an activate local sap request through an IORB request using the LRN just obtained. The Activate Local SAP request will insure that all LAN software and hardware is properly initialized.

The T&V software must always be provided with an interface to system management. The T&V routines must first execute an Associate Monitor call referencing the reserved symbolic name, T&V_USR (this name is the convention for a T&V interface), for T&V users. In the same manner as other LAN users, the T&V user must then issue an activate RQIO to insure that its interface is properly initialized.

During IST processing the SM server IST code must load a pointer to the SM server code in all RCTs belonging to SM server users. The pointer is used by the LDIS routines to call the SM server after an IORB request has been received for it.

2.3.1 SPECIAL FILES

There are two types of files essential to LAN initialization, the File containing the bound unit for the LACS board and the LAN configuration file required to initialize all LAN service access points.

2.3.1.1 LACS BOUND UNIT

Each controller table contains a pointer to a bound unit load table. The bound unit load table in turn contains a pointer or pointers to the pathname of the bound units to be loaded in the LACS. The initial implementation will allow only a single bound unit to be loaded in the LACS. The LACS bound unit is created in a UNIX environment using Bridge development tools. A companion file to each LACS bound unit is the list file describing the memory map of the LACS code in memory. This file must be located in the same directory as the bound unit, with the same pathname except for the suffix '.out '.

2.3.1.2 CONFIGURATION FILE

The configuration file provides detailed information about the network configuration. It contains configuration directives which describe the LACS controllers in the system and all service access points (SAPs) defined for each instance of a protocol layer. Presently 5 different protocols have been identified which could exist on a single contoller:

1. ISO Class IV Transport contains TSAPs.
2. SNAcP Network contains SNSAPs.
3. Null Network contains NSAPs.
4. IEEE 802.2 LLC contains LSAPs.
5. IEEE 802.3 MAC contains MACSAPs.

A controller is described by a symbolic name, megabus address, flow control parameters and other attributes. It's name is assigned at CLM based upon its megabus address slot. Other attributes can be assigned through a unique CT directive specifying attribute values or they may be defaulted as specified in the controllers attribute section, xxxxxx, in this documents CT directives are optional in the configuration file.

Every layer instance directive defined in CLM must also be defined within the configuration file. It need only contain the symbolic name, type and controller information. Following each layer instance directive are the SAP directives defining all the SAPs for this layer instance. Each type of SAP will have an unique group of attributes which describe. In general, each SAP directive will contain a symbolic name, an address, flow control parameters and mapping to the next lower layer. Both local and remote SAPs are defined.

Each SAP is defined differently for a particular type of protocol. A local SAP provides an interface to the next higher layer. If a local user directive has also been defined in the CLM, the interface will be to a user of the LAN. From SAP information, configuration processing will build the local and remote SAP directories and complete the initialization of L6 data structures. Configuration processing will then initialize the data structures in LACS memory through system management requests to the layer management functions on the LACS.

The format of the Configuration file, as well as more detailed information about configuration file processing, can be found in the LAN Configuration Services Component Specification.

2.3.2 LAN INITIALIZATION CONTROL FLOW

The initialization of LAN software, from bootup until an operational state, is brought about in a sequence of steps. CLM processing ensures that memory has been allocated for L6 MOD400 system data structures used in LAN processing. Two types of CLM directives are required for LAN operation, the Layer Instance directive and the Local User directive. CLM processing ensures that a controller directory, control table, layer table and layer instance table are constructed for each controller and layer instance defined in the system. An RCT is created for each local user defined and a user directory created assigning an LRN to each local user defined. At the completion of CLM processing, the LACS hardware is reset waiting to be loaded, LAN software is not operational.

All users of LAN software are required to issue Associate user and Activate SAP calls before attempting to request LAN services. First a user issues an associate monitor call specifying the symbolic name of a SAP. The LAN software searches the user directory for a match with the name and returns the LRN assigned to it during the CLM process. A user must then issue an Activate Local SAP IORB request to the LAN software using the LRN obtained from the previous Associate call.

The Activate Local SAP request provides a dual function. It requests a layer server to make a SAP operational and allows the system management function to initialize the LACS hardware and LAN software. After queueing the request off the RCT, the LDIS software calls the system management code responsible for initializing the LAN. This initialization code checks the controller directory to determine if it is to be initialized. If the controller state indicates it is ready for initialization (null state), initialization processing is started. Initialization is performed in three steps, configuration file processing, LACS loading and LACS layer initialization.

Configuration file processing involves processing the information contained in the configuration directives and updating and establishing the LAN data structures. The tables established during CLM processing are updated to reflect configuration directives or set to predefined default values for a particular structure. Local and remote SAP directories are created for each layer. Each directory entry will point to a set of attributes describing the SAP. These values are specified in the SAP directives in the configuration file. At the completion of configuration file processing, all data structures in the L6 required for LAN operation will have been created. The controllers and local user SAPs are in a reset state, not yet operational.

Immediately following configuration file processing, the loading of LACS memory with LAN software is performed. At the completion of loading each controller, the SM server issues a Start I/O request to start the execution of the kernel software. The kernel initialization table specifies that the I/O Dispatcher, DMA service, and system management layer insatance processes are created. The SM layer instance in the LACS begins its initialization process registers its master data structure pointer with the kernel registers its mailbox directory with the I/O Dispatcher process, and suspends operation until the Start I/O is received from the I/O dispatcher. When the Start I/O LCB is received, the SM layer instance creates and runs all layer management processes for each layer instance specified in the Start I/O request. Each layer management process registers its master data structure pointer with the kernel, creates its transmit and receive process at the priority received from system management as a process creation parameter, and registers its mailbox directory with the I/O Dispatcher process. At the completion of starting up all LACS processes, all controllers are operational (state is set to IN-USE).

Layer initialization of each controller requires that each layer instance on the controller create a SAP for each SAP defined in the local and remote SAP directories. System Management will issue a series of create SAP system management requests to each controller. It will start with the lowest layer SAPs first (MAC) and on up to the highest layer SAPs (Transport). The create SAP request is passed on to the proper layer management process on the LACS board. The create request specifies the symbolic name, physical address, path information (for remote SAPS), flow control parameters, type of SAP and unique attributes of this SAP. The layer management process creates the data and control structures for the SAP and makes an entry in its SAP directory). The state of the SAP is set to inactive (locked and operational). The SAP is not useful until an activate command is received from the user the SAP. At the completion of layer initialization, all layers are operational but there are no SAPS available for servicing requests and the layers have not registered with the next lower layers. The SM server will then return to the task which called it. This task was processing an activate local SAP request and must finish its processing. If other activate requests were received by system management the SM server will perform a system semaphore with wait until initialization process is completed.

At resumption of the activate processing by the layer server, an activate local SAP LCB is issued to the LACS layer instance. The state of the SAP is updated to IN-USE and the LACS layer server process determines the logical address for the SAP. The logical address is returned at the completion of the request. The layer server in the L6 appends the controller and layer instance number to the address (in order to make it unique) and stores it in the RCT for the SAP.

In instances where an activate local SAP request has been received on a RCT already, the LAN server will not issue an activate local SAP LCB but return status that it occurred.

Before any data transfer can occur, a user must issue an activate remote SAP to obtain a logical address for the SAP. When a layer server receives the activate remote SAP request it must check the next lower layer interface the remote SAP maps to. Each remote SAP maps to a remote SAP in the next lower layer. The layer server must issue an activate remote SAP to the next lower layer. At the completion of the activate requests, the layer server is returned logical addresses for the remote SAP. It now determines a logical address to return to the user originating the activate remote request. At the completion of the activate remote request, a layer instance will have registered with the next lower layer, and resolved mappings to the SAPs to the lower layer.

2.3.3 L6 INITIALIZATION CODE

The clm process will load the SM server into system memory, and insure only one task level is assigned to the SM server to run under. When the SM server is loaded into memory, it will perform an initialization subroutine. This routine will:

1. Reclaim patch space.
2. Initialize SM lunge interface (provide addresses for routines used by the LDIS, LDMS, and layer server software)

No initialization processing is done when the Sm server is activated initially by a request.

2.4 TERMINATION REQUIREMENTS

The SM server will be active as long as mod400 is active, therefore there are no termination requirements.

2.5 ENVIRONMENT

The following items are required by the SM server for it to perform it's task:

1. Mod400 operating system.

2. Any l6 computer model except 6/10 and 6/20.

3. A lacs attached to the l6 megabus.

4. Lan clm.

5. LAN Configuration Fileerver.

6. At least one user of system mamagement (T&V).

2.6 TIMING AND SIZE REQUIREMENTS

Currently memory usage and timing requirements are not an issue. However, the code should be be as efficient as possible.

2.7 ASSEMBLY AND LINKING

The software will be written in Series 6 Assembly Language using a subset of the instruction set that is present on all Series 6 systems. Assembly and linking is accomplished through the use of MOD400 MAP and LINK facilities. Assembly source files are located in the System Management LAN directory, xxxxxx. The following EC is used for all MAPs:

TBS

The following EC is used to link all routines:

System Management Component Specification

TBS

The list of all source and macro files used for linking is:

TBS

2.8 TESTING CONSIDERATIONS

Since the product is new, all functions will be tested by the developer, and the software test function.

2.9 DOCUMENTATION CONSIDERATIONS

Documentation of this product follows the recommendations of the Honeywell software documentations guidelines. In addition, all code descriptions will be accompanied by a procedural design language description described in reference [4].

2.10 OPERATING PROCEDURES

All system management users are administrative or maintainence applications. A user must make an Associate Monitor call to obtain an LRN followed by an Activate Local SAP RQIO before requesting system management services. In systems with more than one system management user, it is the users responsibility to avoid conflicting service requests on the same LAN resources (they must be in a cooperating environment).

2.11 ERROR MESSAGES

All errors are assigned a class as specified in the LAN Software EPS. The system management error messages have been assigned the following classes:

Unreportable Catastrophic Error Class Messages

TBS

Reportable Catastrophic Error Class Messages

TBS

Fatal Operation Error Class Messages

TBS

Non-Fatal Operation Error Class Messages

TBS

Recoverable Error Class Messages

TBS

SM Protocol Error Class Messages

TBS

L6 Error Class Messages

TBS

the routine for the layer manager function in the layer instance table. The layer management routine will return status indicating the success of the operation. If status indicates a successful completion or a failure, the SM layer server will start to build and then post a response PDU back to the user. However, if status indicates an incomplete or unsupported operation, the SM server must issue a request to the layer management function on the LACS board. The L6 layer management response is saved and an LCB is created to issue the SM PDU to the system management function on the LACS board. Memory for the LCB is obtained from the reserved area of the IORB. The absolute address of the IORB buffer(s) is obtained and placed in the LCB. All required fields within the LCB are filled in. The SM server calls the flow control routine and passes the address of the LCB. The flow control routine determines if the number of requests on system management by this user has exceeded any flow control requirements. The routine passes a credit value and a status on its' return back to the SM server. Status indicates whether or not the LCB has been queued on the SM layer instance table. The flowcontrol credit value is not presently used during this process. If the LCB has not been queued due to flowcontrol, the SM server issues the LCB by calling the LDMS service routine. At successful completion of issuing the LCB or if the LCB is queued by the flowcontrol routine, the SM server terminates its' task level.

At the completion of the LCB, the LDMS interrupt routine will invoke the SM server by requesting the execution of the TRB pointed to by the LCB. The successfully completed LCB buffer will contain a response PDU corresponding to the request. The SM server calls the flow control routine with a pointer to the completed LCB. It updates its flowcontrol parameters and issues any queued LCBs if possible. The flowcontrol credit value is passed back to the SM server to eventually be returned as part of the IORB completion. When the SM server is returned to, it must construct a single response PDU from the LACS response PDU and any response from a L6 layer management function. The response PDU is returned to the user in the buffer(s) specified in the IORB. If the the buffer space is not large enough to contain the response PDU, the IORB will be completed with a status indicating the buffer was too small and the size of the response PDU will be contained in the residual range word of the IORB. The SM server then releases visibility to the user buffer, dequeue the IRB from the SM RCT and posts back the IORB to the user. The SM server then terminates its' task.

Event IORBs are handled by the SM server in a different fashion. A user of system management is required to issue an event IORB to allow the SM server to report unsolicited events. The SM server is required to issue an event LCB to the LACS system management function on every LACS controller configured in the system. Whenever an event LCB completes, the SM server will issue another event LCB to the LACS for the next event. Similarly, the user of system management must issue another event IORB to the SM server to allow the reporting of the next

3.0 L6 INTERNAL DESCRIPTION

3.1 INTERNAL OVERVIEW

System management is a function which responds to requests from a higher level application such as NAD or ADAP. The processing required to execute a request is determined from the SM PDU passed as part of the station management command buffer. The station management task must interpret the SM PDU request contained in the command buffer and route it to the layer management function within the L6 and if necessary to the LACS system management layer instance to process it. The SM server also provides a system manager layer manager function. This is analogous to the layer management functions defined for other layers. The system management layer manager function processes requests to control the system manager administrative function and the LACS controller. Additionally, the SM server initializes the system for users of LAN services.

The SM server is partitioned into three distinct functions. These functions can be further partitioned into smaller modules or routines. Modules are unique to the particular function or a common module can be used by more than one. The highest level of partitioning breaks the system management task in the L6 along functional similarities. Three major functions are defined, the system management user request services, system management administrative functions, and LAN intialization processing. XA block diagram of the major system management partitions is shown in figure 3.1.a.

3.1.1 SYSTEM MANAGEMENT USER REQUEST SERVICES

System management user requests services are provided through a monitor call, RQIO, with an associated IORB and extension. The user requesting system management must specify the IORB as described in section 2.1, L6 External Interfaces. The LDIS routines are executed and perform the initial processing on the IORB. The system manager layer server is invoked when the IRB has been queued on a system management RCT and with a pointer to the IORB in a $B register. There are two types of requests which the system manager must process, a SM PDU request for service or a request to receive a buffer for issuing event indicate SM PDUs back to the user. In either case, the system manager layer server must validate the IORB parameters and request visibility of the users buffer(s) from the executive.

A SM PDU request is processed differently than an event indicate IORB request. The buffer(s) pointed to by the SM PDU request IORB contain an SM PDU request for service. The SM server must first validate the PDU and insure it is correct. A copy of the PDU is created to allow the response PDU to be placed in the IORB buffer(s) during later processing. The SM server then interprets the PDU and routes a request to the proper layer manager function. Routing information is obtained from the resource id record of the PDU. The SM server insures it contains a valid resource id and operation request then calls event. The SM server saves the address of the IORB in the transfer table of the user. Along with each event IORB, the user can specify what class of event it wishes reported. The SM server will queue all received events of a class specified by the event class mask in each system management users transfer table. When an event is received by the SM server, a search is made of the event destination table to determine which users are available to report it to. The event destination table contains the address of the users RCT for each local entry. If the event class mask indicates this event is to be reported to this user, the SM server moves the event PDU into the buffer specified in the event IORB. After successfully posting the event IORB or if the event class indicates it is not to be reported, the SM server will look for the next entry in the event destination table. In the case where the transfer table of an RCT does not contain the address of an IORB, the SM server queues the event LCB on the tail end of the event LCB queue in the transfer table. When an event IORB is received the SM server must check to make sure there are no pending events to be reported in the event queue.

During initialization of the system for LAN services, the SM server issues an event LCB to each controller that is configured. The event class mask for each system manager user indicates that no events are to be reported. Each system manager user must issue an associate user, activate local sap request, and an event IORB request before any events can be reported. The SM server enters an entry for a local user in the event destination table when an activate local sap is received. Currently, only local entries are allowed in the event destination table. A user can request a set of events to be reported by specifying a new event class mask in the event IORB request. If the number of event LCBs queued of a users transfer table exceeds the number of events allowed for this user, all user IORB requests will be rejected until the number of events queued is within the specified limits. Only one outstanding event IORB is allowed per user. If a second event IORB is received before the first IORB has been used by the SM server, the first IORB is posted back with status indicating a second event IORB has been received. All event IORBs are received with a pointer to a buffer to report the event in. The SM server fills the buffer with an event PDU before posting the completed event IORB back to the user.

The following modules are utilized to process system management user request services:

User Request Service routine
IORB Validation routine
PDU Validation routine
PDU Decode routine
PDU Formatter routine
PDU to Layer Manger Formatter
Layer Manager to PDU Formatter

Event Service routine
Event Handler routine
Activate System Management "SAP" routine

3.1.2 SYSTEM MANAGEMENT LAYER MANAGER FUNCTION

The system management layer manager function provides the same type of service as other layer management functions. It is responsible for the control of the station management administrative function and the LACS controller object. The system management request services calls the system management layer manager function in the same manner as any other layer management functions. The address of the administrative function routine is located in the layer instance table for system management. The administrative function routes a request to either a routine operating on the system management administrative function object or a routine operating on the controller object. The following functions can be performed on the system management administrative function and the controller:

Get All Attributes of System Management Administrative Function
Get All Statistics of System Management Administrative Function
Action List Object Selection Parameters
Get All Attributes of LACS Controller
Get All Statistics of LACS Controller
Action Update State of LACS Controller
Action Dump LACS Controller
Action Load LACS Controller

A Get operation on all attributes or statistics will return a list of values in a IEEE802 defined format for a parameter list (see section 2.2.2 on SM PDU construction). An all attributes request is completed entirely within the L6 for the SMAF and the Controller. Status is returned to indicate the operation is complete. An all Statistics request on the SMAF is also completed within the L6. An all statistics request on the controller can not be completed (some statistics are kept in LACS memory) and status must be returned to indicate only a partial operation was completed. An Action List request will return a list of the selection. Parameters for a specified type of object or component. The SMAF accesses the local and remote SAP directories to obtain the selection parameters for the type of object or component specified in the request. The result is returned as a object list described in section 2.2.2. An Action Update state command will change the state of the controller and issues the appropriate IO instructions to the LACS controller. The Dump and Load requests will dump and load the LACS memory onto or from a specified file. The format of a Dump or Load file is a modified MOD400 Rev3.1 bound unit format as specified in section 2.3.

The state of each object is checked before any operation is performed. The SMAF must be in the IN_USE state in order to successfully complete the request. The controller must be in the LOCKED/HALTED state or TEST state in order to complete a dump request. The controoler must be in a TEST state or a LOCKED and a substate of RESET or HALTED before a Load request is successfully completed. The SM layer management function informs each layer instance that a controller is unavailable for service as a result of the controller entering the LOCKED or DOWN state. Each layer instance then deactivates all SAPS within the instance.

The state of the controller and the states of the SMAF are descibed as follows:

SYSTEM MANAGEMENT ADMINISTRATIVE FUNCTION (SMAF) STATES

IN-USE STATE

DESCRIPTION

A system management administrative function is in the IN-USE state when a LAN adapter has been fully initialized and an activate SAP request has been received from a user of system management services.

CAUSE OF TRANSITION

The system management administrative function can transition into the IN-USE state from the LOCKED state. The transition from the LOCKED state will occur at the successful completion of an Activate Local SAP request on the system management SAP.

NEXT STATES

NONEXISTENCE — The system administrative function must remain in the IN-USE state as long as the system is to remain operational with the LAN.

AFFECTS ON STATES OF OTHER OBJECTS

The system management administrative function can transition into the IN-USE only from the LOCKED state. This does not affect the state of any other object.

ENABLED STATE

For future study.

DISABLED STATE

For future study.

LOCKED STATE

DESCRIPTION

A system management administrative function in the LOCKED state is unavailable for service to system management users.

CAUSE OF TRANSITION

The system management administrative function is in the LOCKED state only after LAN initialization has occurred and an Activate Local SAP request has not been received.

NEXT STATES

IN-USE   Due to operator command or during system configuration.

AFFECTS ON STATES OF OTHER OBJECTS

None. While the system management administrative function is locked all requests from users.

DOWN STATE

Not Applicable

TEST STATE

For future study.

SHUTDOWN

For future study.

NON-EXISIENCE

The system management administrative function will be created in LOCKED state at system initialization.

CONTROLLER STATES

IN-USE STATE

DESCRIPTION

The controller is fully operational and able to service user request for controller processes. This requires that the LACS board has been fully initialized (LACS software has been loaded, LACS kernel and processes are running, and all SAPs defined for this LACS have been created).

CAUSE OF TRANSITION

The controller object may transition into the IN-USE state from the LOCKED state. It will transition from the LOCKED state due to an operator command or during system configuration.

NEXT STATES

LOCKED	Result of a request by operator.
DOWN	Result of a hardware fault detected on the LACS board.

AFFECTS ON STATES OF OTHER OBJECTS

NONE

ENABLED STATE

For future study.

DISABLED STATE

Not applicable to the controller object.

LOCKED STATE

DESCRIPTION

A controller in the LOCKED state is not available for service by any non-adminstrative users. While residing in the LOCKED state , the controller can be in one of four substates which correspond to the operational capabilities of the LACS board.

CAUSE OF TRANSITION

The controller can be transitioned into the LOCKED state from the IN-USE or DOWN states as a result of an operator command. It can also enter the LOCKED state from TEST state under control of the test program.

NEXT STATES

IN-USE	Due to operator command or system configuration.
DOWN	Due to detected hardware failure.
TEST	Due to test program.

AFFECTS ON STATES OF OTHER OBJECTS

A controller object transitioning to a LOCKED state will return all user requests (except for administrative requests) for the controller's services (the LACS driver Megabus services is responsible for checking its state).

DOWN STATE

DESCRIPTION

A hardware fault has been detected in the controller.

CAUSE OF TRANSITION

LACS board firmware has detected a fault associated with the controller or a LACS controller timeout has occurred.

NEXT STATES

LOCKED	Due to operator command.
NON-EXISTENCE	

AFFECTS ON STATES OF OTHER OBJECTS

A controller object transitioning to a DOWN state will return all user requests (except for administrative requests) for the controller's services (the LACS driver Megabus services is responsible for checking its state).

TEST STATE

DESCRIPTION

A controller object in the TEST state is under control of a test program.

CAUSE OF TRANSITION

A test program transitions a controller to the TEST state from the LOCKED state due to a test program request.

SUBSTATES

RESET	LACS board hardware and firmware has been reset. LACS firmware functions are operational. May transition to the LOCKED and RESET state from any other state.
HALTED	LACS processes have been halted. LACS hardware and firmware functions have not been reset and LACS firmware functions are operational. May transition to the LOCKED and HALTED state from any other state.
LOADED	LACS board memory has been loaded. LACS firmware functions are operational. May transition to the LOADED substate only from the RESET or HALTED substate.
STARTED	The Kernel software and LACS software is executing. LACS firmware functions are not operational. May transition to the STARTED substate only from the LOADED substate.

NEXT STATES

LOCKED	Due to test program.
DOWN	Due to test program.

AFFECTS ON STATES OF OTHER OBJECTS

Specific to nature of test program.

SHUTDOWN STATE

For future study.

NON-EXISTENCE

The controller object is initially created in the locked state at system initialization.

The SM server is responsible for determining whether a controller is available for service. It is possible that an unreportable catastrophic error occur on a LACS. The SM server must insure that the users of the LAN are notified. However, if there are not any further requests outstanding, the SM server needs another mechanism to dsetect a DOWN controller. The TicToc LCB is intended to be used for this purpose. The SM server will issue a CRB to the Executive to occur at a time specified for this configuration. When the CRB invokes the SM server it will determine whether a LACS has reported back since the last CRB and issue a new one. If a DOWN LACS is detected, the SM server will deactivate all SAPs defined for the LACS. This insures that all outstanding user requests are properly returned.

The following routines are utilized by the SM server in order to process system management administrative functions:

SM Layer Management Interface routine
Get SMAF service routine
SMAF Action List service routine

Get Controller service routine
Controller Action Update State service routine
Controller Action Dump service routine
Controller Action Load service routine
Bad operation service routine
Controller Tictoc service routine

3.1.3 LAN INITIALIZATION PROCESSING

All users of LAN facilities are required to issue Associate user monitor calls and Activate Local SAP requests before attempting to request LAN services. The Activate Local SAP request provides a dual function. It requests a layer server to make a SAP operational and allows the system management function to initialize the LACS hardware and LAN software. On every activate local SAP request, the LDIS activate routine calls the system management code responsible for initializing the LAN. This initialization code checks the controller directory to determine if it is to be initialized. If the state of a LACS controller indicates it is locked and reset then initialization processing is started. Initialization is performed in three steps, configuration file processing, LACS loading and LACS layer initialization.

Configuration file processing involves processing the information contained in the configuration directives and updating and establishing the LAN data structures. The initialization function calls the configuration file processing routine to establish the values for all L6 LAN data structures from the Configuration file. At the completion of configuration file processing, all data structures in the L6 required for LAN operation will have been created. Immediately following configuration file processing, the loading of LACS memory with LAN software is performed. The system management administrative request to load each controller is made for each Controller. The controller is then issued a Start I/O LCB. The Start I/O LCB will point to information instructing the system management layer instance to create and start LACS layer instances.

Layer initialization of each LACS requires that each layer processor for LACS create a SAP for each SAP defined in the local and remote SAP directories. System Management will issue a series of create SAP system management requests to each controller. The create SAP request is passed on to the proper layer management process on the LACS board. The create request specifies the symbolic name, physical address, path information (for remote SAPS), flow control parameters, type of SAP and unique attributes of this SAP. The layer management process creates the data and control structures for the SAP. The SAP is not useful until an activate request is received from the user of the SAP. At the completion of layer initialization, all layer servers are able to accept user requests. The SM server will then return to the task which called it. If the SM server had previously initialized the LAN, it will immediately return to the calling task.

The following routines are utilized by the SM server in order to perform the LAN Initialization services:

LAN Initialization routine
Configuration File Processing routine
SAP Initializer routine

3.2 L6 MODULE DESCRIPTION

3.2.1 User Request Service routine

Module name: SMURQS

Purpose

This routine services a users request after the LDIS routines have queued the users IRB and posts the completed request back to the user.

Description

The user request service routine is branched to from the LDIS services. It calls the IORB Validate routine to validate the IORB (common) and then determines whether it is a SM request or an Event indicate IORB. If it is a SM request it branches to the SM Routing routine, otherwise it will branch to the Event Service routine. The Post IORB routine (common) is called to post the IORB back to the user in the case where an invalid IORB is detected and the task is terminated.

Inputs

$B4	Contains a pointer to the IORB
IRB	Queued on RCT

Outputs

$B4	Contains a pointer to the IORB

Required Modules

IORB Validate routine	Validate the IORB
SM Routing routine	Processes SM requests and routes the request to the proper layer management routine.
Event xxxxxxx routine	Processes event requests.
Post IORB routine	Posts the IORB back to user.

Restrictions

None.

3.2.2 System Management Routing routine

Module name: SMROUT

Purpose

This routine processes all request made to the system manager in the form of a SM PDU.

Description

The system management routing routine receives a pointer to an IORB for a system management request. This routine obtains visibility to the users buffer(s) specified in the IORB by calling the visibility routine (common). It calls the routine, Validate PDU, to validate that the PDU is a properly formed and a Honeywell supported IEEE802 system management PDU. An invalid PDU is returned to the user as a response PDU in the IORB. A copy of a valid PDU is created by the Copy PDU and saved until its specified operation is completed. The Decode PDU routine is called to obtain the routing and operation information. This information is passed to the xxxxxx to be issued to the layer management routine specified by the routing information. The status from layer manager routine will indicate whether a full or partial operation has been completed. All operation fully completed (successful or not) are to be returned to the user immediately. The routine, layer management to PDU, is called to format the response PDU. The response PDU is copied by the routine, Store PDU, into the buffer(s) specified in the IORB and memory containing the copy of the request PDU is released. The IORB is posted back by calling the routine, Post IORB. The task is then terminated. Status is returned to indicate a valid PDU, a badly formed PDU, or an unsupported operation. If layer management status indicates the operation is only partially completed, this routine must request service of the LACS system management function. The absolute address of the SM command buffer containing the PDU is obtained from the routine which obtains an absolute physical address from the virtual address, Absolutize address (common). The routine, create LCB, is called to initialize the LCB which will pass the absolutized address of the request SM PDU. After completing the LCB, the LCB Handling routine is called to issue the LCB to the megabus services. At the completion of the LCB, the results of the L6 layer management request and the response PDU from the LACS request are combined intro a single response PDU by the PDU Formatter routine. The response PDU is copied by the routine, Store PDU, into the buffer(s) specified in the IORB and memory containing the copy of the request PDU is released. The IORB is posted back by calling the routine, Post IORB. The task is then terminated.

Inputs

$B4 Contains a pointer to the IORB.

Outputs

None.

Required Modules

Post IORB routine (ZSMPRB)
Posts the IORB back to user.

PDU Validation routine (ZSMVDU)
Checks the format of a received PDU.

PDU Decode routine (ZSMDDU)
Obtains the routing and operation information from a PDU.

PDU Formatter routine (ZSMFDU)
Create a SM PDU.

PDU to Layer Manger Formatter (ZSMPLM)
Formats a request for a Layer management function.

Layer Manager to PDU Formatter (ZSMLMP)
Interprets the results from a layer management function.

SM Layer Management Interface routine (ZSMLMI)
Processes request to the system management "layer manager".

Restrictions

None.

3.2.3 IORB Validation routine

Module name: ZSMVRB

Purpose

Validate SM server specific fields in the IORB.

Description

This routine checks the channel specific function code and the device specific function code of an IORB received from the LDIS services. An invalid IORB will be returned with the status set in the IORB status field and R1 set to indicate an invalid IORB. A valid IORB will be returned unchanged and a valid IORB status returned in R1.

Inputs

$B4	Pointer to IORB

Outputs

$R1	Status

Required Modules

None.

Restrictions

None.

3.2.4 PDU Validation routine

Module name: ZSMVDU

Purpose
Checks the format of a PDU.

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.5 PDU Decode routine

Module name: ZSMDDU

Purpose
Obtains the routing and operation information from a SM PDU.

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.6 PDU Formatter routine

Module name: ZSMFDU

Purpose

Creates a single SM response PDU from a L6 layer management result and a LACS SM response PDU result.

Description

This routine checks the L6 layer management response status to determine whether L6 results must be combined with the LACS response PDU. The two records are compared until a difference is found between the PDUs. If a valid PDU can be created for that record, a single response PDU will be made. Otherwise a single response PDU with a status indicating a a response PDU could not be formatted will be created and returned to the calling routine.

Inputs

$B2	Pointer to LACS response PDU
$B3	Pointer to L6 response PDU
$B4	Pointer to store combined response PDU

Outputs

$B4	Pointer to response PDU to be returned to the user.
$R1	Status

Required Modules

None.

Restrictions

None.

3.2.7 PDU to Layer Manger Formatter

Module name: ZSMPLM

Purpose
Formats a request for a layer management function.

Description
This routine creates a layer management request block to be created from the information contained in the request PDU. It recognizes the basic operations and formats it into as described in the layer management interface description in section 2.2.4.

Inputs

$B4	Pointer to request info field of PDU
$B2	Pointer to buffer fort layer management interface block.

Outputs

$R1	Status.

Required Modules
None.

Restrictions
None.

3.2.8 Layer Manager to PDU Formatter

Module name: ZSMLMP

Purpose

Create a response SM PDU from results obtained in a layer management result block.

Description

This routine creates a SM response PDU from information contained in the request PDU and the results obtained from the layer management function. The routing information and the exchange id are copied from the request PDU. The result of the operation is assumed to be in a IEEE802 format and joined with the routing portion of the PDU.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.9 Event Service routine

Module name: ZSMEVS

Purpose
Processes requests from the user to receive events.

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.10 Event Handler routine

Module name: ZSMEVH

Purpose
Processes events received from a LACS.

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.11 Activate System Management "SAP" routine

Module name: ZSMACT

Purpose

Processes activate local SAP requests from a user of system management.

Description

This routine processes an IORB which specifies an activate local SM sap. The routine calls the system management LAN initialization routine to insure that LAN services are operational before processing the request. The routine then checks the SAPs RCT to determine if an activate request has been previously received. If an activate request has been received then status is returned to indicate so. Otherwise, the RCT is flagged to indicate an activate has been received and the event destination table is updated to contain an entry for this RCT. The IORB is then posted back to the user.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.12 SM Layer Management Interface routine

Module name: ZSMLMI

Purpose

Provides the layer management interface for the system management layer management function.

Description

This module receives all "layer management" request for the system management administrative function. It receives a layer management interface block and calls the proper service routine based on the operation code and class. There are two classes of objects it services, the system management administrative object and the LACS controller object. It inspects the class and fetches the address of the routine based on the objects operation. Upon completion of the operation, the layer management interface block is returned completed.

Inputs

$B4 pointer to layer management interface block

Outputs

$R1 Status

Required Modules

- Get SMAF service routine (ZSMGSM)
  - Reads the attributes and statistical values of the system management administrative function.
- SMAF Action List service routine (ZSMALS)
  - Reads the selection parameters of a specified type of object or component.
- Get Controller service routine (ZSMGCT)
  - Reads the attributes and the statistical values of the controller.
- Controller Action Update State service routine (ZSMAUP)
  - Updates the state of the controller.
- Controller Action Dump service routine (ZSMADP)
  - Reads LACS memory and copies to a specified file.
- Controller Action Load service routine (ZSMALD)
  - Loads LACS memory with a specified file.
- Bad operation service routine (ZSMBOS)
  - Handles requests for unsupported or illegal operations.

Restrictions

None.

3.2.13 Get SMAF service routine

Module name: ZSMGSM

Purpose

Reads the attributes and statistical values of the system management administrative function.

Description

This routine receives a parameter id and returns the value for that parameter. A table is kept which maps the IEEE802.1 parameter id to the position in the system management attribute list and the size of the value to be read. There are two special parameter ids, all attributes and all statistics. For either of these two parameter ids, a IEEE 802 formatted parameter list containing all system management attributes or statistics is created and returned as the result of the request. The format of these lists is defined in section 2.2.x.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.14 SMAF Action List service routine

Module name: ZSMALS

Purpose

Reads the selection parameters of all objects or components of a specified type.

Description

This routine searches the local and remote SAP and object directories of the specified object or component. The selection parameter for each object in the directory are placed in an IEEE802 list and returned as a result.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.15 Get Controller service routine

Module name: ZSMGCT

Purpose

Reads the attributes and statistical values of the controller object.

Description

This routine receives a parameter id and returns the value for that parameter. A table is kept which maps the IEEE802.1 parameter id to the position in the controller attribute list and the size of the value to be read. There are two special parameter ids, all attributes and all statistics. For either of these two parameter ids, a IEEE 802 formatted parameter list containing all controller attributes or statistics is created and returned as the result of the request. The format of these lists are defined in section 2.2.x.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.16 Controller Action Update State service routine

Module name: ZSMAUP

Purpose

The purpose of this routine is to update the state of the controller to a specified state.

Description

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.17 Controller Action Dump service routine

Module name: ZSMADP

Purpose

The purpose of this routine is to dump the contents of a LACS boards memory into a file in a LACS Bound Unit format.

Description

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.18 Controller Action Load service routine

Module name: ZSMALD

Purpose

The purpose of this routine is to load the contents of a LACS boards memory from a file in a LACS Bound Unit format.

Description

This routine first check the state of the controller to load. The load operation is only allowed if a controller is in the LOCKED state and a substate of RESET, HALTED, or LOADED. The LACS bound unit to load is found from the Bound unit pathname pointer in the LACS controller table. If the xxxxxx indicator has been set, the LACS will be issued a Start I/O request to begin the execution of the software.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.19 Bad operation service routine

Module name: ZSM...

Purpose

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.20 LAN Initialization routine

Module name: ZSMLIT

Purpose

The purpose of this routine is initialize the system for LAN services.

Description

All layer servers must call this routine when processing an activate local SAP request. The LAN Initialization routine checks the state of each controller. Any controller in the LOCKED and RESET state is loaded with LAN software by a call to the Controller Action Load routine. It then calls the SAP Initializer routine to create all SAP data structures on the LACS. At the completion of creating all the SAPs, the LACS controller is in the IN-USE state. The process is then repeated until all controllers that were RESET are in the IN-USE state.

Inputs

Outputs

Required Modules

Restrictions

None.

3.2.21 Configuration File Processing routine

Module name: ZSMCFP

Purpose
Initializes LAN data structures from configuration file.

Description

Inputs

Outputs

Required Modules

Restrictions
None.

3.2.22 SAP Initializer routine

Module name: ZSMSIP

Purpose
Create all SAPS defined for a controller.

Description
This routine accesses all SAP directories defined for the LAN. It must determine whether a SAP is assigned to the specified controller.For each entry found, a SM PDU is created containing an Action Create SAP request and issued to the LACS on which the SAP exists. The routine first searches the system management directory then the physical directory and down to last non-null layer SAP directory in the order defined in LAN information table.

Inputs
$R1 Controller Number

Outputs

Required Modules

Restrictions
None.

3.2.23 LCB REQUEST MODULE DESCRIPTION

Module name: ZSMLCB

Purpose

The purpose of this module is to create an LCB to be issued to the LACS board from an IORB and a SM server buffer and issue that LCB to the LDMS routines.

Description:

A request for services of the LACS Manager is initiated through any command module to the LCB_req_handler module. The LCB_request_handler is passed a pointer to the IORB and SM server buffer containing the request for LACS services.

Input:

B4	address of IORB
B5	Return address

Output:

R1	Status

Required Modules:

None.

Restrictions

None.

3.2.24 SM LPS IST MODULE DESCRIPTION

Module name (ZSMIST)

Purpose

Provide initialization of System management layer server task.

Description

This routine is invoked only at system initialization. It searches the system management directory and initializes all system management layer instance tables to contain the address of the system management Layer Management Interface routine. This routine returns unused patch space to the system and registers the SM server lunge interfaces for use by other bound units.

Inputs

Outputs

Required Modules

Restrictions

None.

3.3 L6 FUTURE DEVELOPMENT AND MAINTENANCE

To be supplied at a later date.

4.0 L6 SYSTEM MANAGEMENT PROCEDURAL DESIGN

To be supplied at a later date.

5.0 L6 SYSTEM MANAGEMENT ISSUES

The following represent the most current issues associated with the system management function in the L6 :

1. What are the objects and components of ISO Transport and Network?
   - what are there attributes and statistical values?
   - what states are represented?
   - what is the view of these objects for users of system management?

3. How are the system management software modules to be maintained and linked?
   - will all L6 IEEE802 software be maintained as a single bound unit; are there any advantages to maintaining LLC and system management as separate bound units?

4. What is required (if anything) of system management in order to allow a smooth termination of a user due to a task group abort?

5. Can the LACS HW lose an IOLD?
   - introduce sequence numbers again?

6. Can visibility be maintained after termination of the SM task and released at interrupt level or task level after an interrupt?

6.0 LACS SYSTEM MANAGEMENT EXTERNAL DESCRIPTION

6.1 LACS DATA STRUCTURES

6.1.1 Mailbox Directory

The mailbox directory will contain the mailbox ids for all required processes. There are 64 possible entries for layer management processes. Those that are not created are to contain nulls. System management will fill in the values of each entry during intialization.

IO dispatcher 16 bit integer

DMA services 16 bit integer

MAC service 16 bit integer

Physical layer management mailboxes instance 0-7
- this includes MAC
16 bit integer

Link layer management mailboxes instances 0-7
16 bit integer

Network layer management mailboxes instances 0-7
16 bit integer

Transport layer management mailboxes instances 0-7
16 bit integer

REMAINING ENTRIES ARE RESERVED FOR FUTURE USE

6.1.3 I/O DISPATCHER FUNCTION CODE MAILBOX DIRECTORY

The function code mailbox directory will contain the mailbox ids for the function specific function code (FSFC) in the second word of an IOLD instruction. There are 16 possible function codes (values 0-F), each can be assigned its' own mailbox id. The SM layer instance process supports the following FSFC values:

6 - SM Request/Response PDU
A - SM Activate
D - TicToc
E - SM Event

All other values are not supported and wtreated as an error. The function code mailbox directory is a table of 16 mailbox ids, the FSFC value corresponds to the the entry in the table. The SM layer instance creates the table and issues it to the I/O Dispatcher process during its' process initialization.

6.1.3 SYSTEM MANAGEMENT ATTRIBUTES

The attributes of system management administrative function are contained in a table with the following elements.

SM_MGR_NAME [xx] Implicit IA5 string
Name of the SM user. Must be 8 ASCII characters. Under ordinary conditions there should only be one system manager interface. During testing operations, the T&V routines will require a dedicated interface with a unique name, T&VMGR.

CLASS [xx]

TYPE [xx]

VENUE [xx]

LMGR_STATE [xx] Implicit Sequence {

[0] State Implicit Octet string
1 octet. State of System Manager. Default value set to indicate locked - 03. At CLM set to LOCKED.

[1] Substate Implicit Octet string
1 octet. Null for initial implementation. Default value of zero. At CLM set to RESET
}

ManufacturerId [xx] Implicit IA5 string
Manufacturer name and country. The manufacturer name consists of the character string 'Honeywell Informations Systems,USA'.

StationTypeId [xx] Implicit Octet string
Manufacturer specific station type. TBS

OPT_SUPP [xx] Impicit Integer
2 octets.Options supported. Null for initial implementation. Default value of zero.

MAX_LCB [xx] Implicit Integer
16 bit integer.Maximum number of LCBs allowed for SM server. Default value of xx.

EVENT_LCB_POINTER [xx] Implicit address
Address of outstanding event LCB. Initial valu is null until the event LCB is received.

6.1.4 LAYER INSTANCE TABLE

The system management layer instance creates a common data structure for every layer instance it creates during the LACS initailization. The layer instance's use this table to hold any common parameters and variables required between the transmit, receive, and layer management processes. Each type of layer instance (protocol type) will have a unique table format known through use of a naming convention all layer protocol entities use (TBD).

6.2 LACS EXTERNAL INTERFACES

The LACS system management layer instance has interfaces to the L6 system management layer server as well as to each of the three layer management functions residing in the LACS. The L6 system management layer server interface is provided by the transmission of LCBs across the Megabus through the use of the DMA firmware routine. The layer management interfaces are provided by mailbox messages to the specified layer management function. Events which have been enabled by system management are reported to system management through an event mailbox by a layer management function.

The L6 system management layer server transmits and receives LCBs to the system management layer instance in the LACS. The LCBs contain commands and responses to and from the LACS system management. The LACS board initially receives an IOLD order across the Megabus to LACS I/O Dispatcher software, it contains information about where the LCB is in L6 memory.

The LCB Receive routine (see LACS Software entity Specification, section xxx) performs the actual transfer of the LCB information into a LCB image in the LACS procedure RAM. A pointer to the LCB image is then issued to the LACS system management layer instance. In the case of a system management LCB pointing to a SM PDU, a copy of the system management command buffer will also be returned. At the completion of the requested service the LACS system management layer instance will request that the portion of the LCB in L6 memory which needs to be updated from its image, through the DMA firmware routine.

6.2.1 LCB FORMAT

The format for system management LCBs has been previously described in section 2. 2, L6 External Interfaces, it is repeated here only to aid in the reading of this document.

the following fields are lacs specific

```
cb_icw  interrupt control word
    initialized by sm ls
    contains interrupt control word
            bits 0-5 - rsu and mbz
            bits 6-9 - cpu number to interrupt
            bits a-f - level to interrupt the cpu
    referenced by:  lacs megabus interface software

cb_csf  channel specific function code
    initialized by sm ls
    contains channel specific function code
            0 - SM request/response
            1 - SM event indicate

cb_cts  controller status
    initialized by lacs software
    contains controller status
```

bit f - invalid function code when set
bit e - ram memory exausted when set
bit d - ram location non-existent when set
bit c - ram parity error when set
bit b - level 6 memory yellow when set
bit a - level 6 memory non-existent when set
bit 9 - level 6 bus parity error when set
bit 8 - level 6 memeory red when set
referenced by sm ls

cb_fss function specific status
initialized by lacs software
contains function specific status
bit 0 - LCB was aborted when set
referenced by sm ls

cb_cbs completion word
initialized by sm ls
contains completion word, number of buffers
bit 0 - LCB is complete when set
bits 8-f - number of buffers

cb_lrs logical remote sap address
Not initially supported
contains logical remote sap address

cb_lls logical local sap address
Not initially supported
contains logical local sap address

cb_trg total number of bytes
initialized by lacs software
contains total number of bytes read for reads
what about the total write ????

The following 2 fields are repeated the number of times represented by the right byte of the cd_cbs field.

cb_adr level 6 buffer byte address
initialized by sm ls
contains level 6 buffer address in bytes

cb_rng range word
initialized by sm ls
contains range in bytes of the cb_adr field

6.2.2 System Management Data Buffer

The format for system management command buffer has been previously described in section 2.2, L6 External Interfaces, it is repeated here only to aid in the reading of this document. The system management buffer contains all command, modification, test and response information required for a particular system management service.

**** INCLUDE SM PDU DESCRIPTION FROM SECTION 2.2.2.1 ****

6.2.3 Layer Management Interface Description

The LACS system management layer instance interfaces with a layer management function for each layer instance defined on the LACS board. This interface is provided through kernel messages between a layer manager process and the system manager process. The format of a layer management message is as follows:

0 15
message header
exchange id
layer internal selector
access control
status
operation code
operation information

There are three different message types, a request message, a confirm message and an event indicate message. A request message is always from system management to a layer manager. A confirm message is from a layer manager to the system manager due to completion of a request message. An event indicate message is an unsolicited message from a layer manager to report an event to the system manager.

The message header for request and confirm messages is the standard message header used for the LAN processes. Request messages contain the message type for request and the system management mailbox id for a return mailbox. The layer manager will return a confirm message type.

message	type value
request message type	(xx)
confirm message type	(xx)
event indicate message type	(xx)

Each request message contains information concerning the operation to be performed and identifies the object or component within the layer to perform the operation on. Each operation will describe operation information required to perform the operation. The confirm message to a request will be identical except for the status and operation information. The format of the request and confirm messages is as follows:

exchange id
layer internal selector
access control
status
operation code
operation information

6.2.3.1 EXCHANGE ID

The exchange id is specified in each request to a layer manager. System management obtains it from the original request and attaches it to each request. Layer management must return the exchange id with the confirm message.

exchange id	16 bit integer

6.2.3.2 LAYER INTERNAL SELECTOR

The layer internal selector describes the selection parameters used by the layer manager to determine which object or component within the layer this operation is to be performed on. It contains the following fields:

name
 8 ASCII characters. Null when not to be used

class
 8 bit integer. Unique class numbers have been assigned for each type of object and component across all layers.

Controller	(13)
System Management Administrative Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)
Transport Connection	(xx)

type
 4 ASCII characters. Unique type is assigned for each class of object.

LACS Controller	(LNCT)
SMAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	
TSAP	
Transport Connection	

```
venue
    8 bit integer
        any           (0)
        local         (1)
        image         (2)
state
    8 bit integer
        any           (00)
        locked        (03)
        enabled       (04)
        disabled      (05)
        test          (06)
        down          (07)
        shutdown      (08)
        in-use        (09)
substate
    8 bit integer
        any           (00)
        reset         (01)
        halted        (02)
        loaded        (03)
        started       (04)
        operational   (05)
```

The internal layer selector must be matched by the layer management function before the operation is carried out. If a match is not located than the layer management function must return status to indicate which selection parameter did not match.

6.2.3.3 ACCESS CONTROL

The access control is passed to each layer manager and can be used by the layer manager to determine if the operation is allowed. The initial implementation will not make use access control. Access control will always be a zero.

Access Control (00)
- Always zero

6.2.3.4 STATUS

Status contains two fields, the status code and a pointer to status info. The status code will identify the particular source of the status and a status id indicating success or the reason for failure. Status info is any operation specific status action. The first word of status info indicates the size of the buffer containing status info (including the status info size word) followed by status info data. The format of the status field is as follows:

Status code		
Source	8 bit integer	
	management	(0)
	physical(MAC)	(1)
	link(LLC)	(2)
	network	(3)
	transport	(4)
Status ID	8 bit integer, unique values assigned for each layer.	
Status Info		
Statuslength	16 bit integer, indicates the number of bytes contained in status info data.	
StatusInfoData	Variable length,unique to each layer.	

6.2.3.5 OPERATION CODE

The operation code specifies one of three possible operations, LM_GET_VALUE service primitive, LM_SET_VALUE service primitive, and LM_ACTION service primitive. The operation information field for each operations request or confirm is unique to the operation. The operations for the LAN layer management functions are described as follows:

LM_GET_VALUE	(01)
LM_SET_VALUE	(02)
LM_ACTION	(04)

6.2.3.6 LM_GET_VALUE SERVICE PRIMITIVE

Read the specified attributes of an object or component within the layer. The format for a LM_ GET_ VALUE service primitive operation information field is as follows:

0 — 15
parameter id
length of confirm information
confirm information

The operation information field for a Get operation specifies the value of the parameter id corresponding to the parameter desired. Only one parameter id may be specified. In addition to any layer unique parameter ids, every layer manager must be able to accept a request to read all

Honeywell attributes or all Honeywell statistical values. The confirm to a get request must contain a status with respect to the completion of the operation and a pointer to the value of the parameter. The confirm information contains the value of the parameter or a list of values containing all Honeywell attributes or all Honeywell statistics in an IEEE802 defined format for a private parameter id. Refer to the proper layer specification for more information on a particular layers confirm. The parameter id's for Honeywell parameters are

All Attributes	(xx)
All Statistics	(xx)

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad parameter id	(xx)
Bad layer internal selector	(xx)
TBU	

The status info which can be returned is as follows:

TBS

6.2.3.7 LM_SET_VALUE SERVICE PRIMITIVE

Set the specified attributes of an object or component within the layer. Only a single attribute may be specified per LM_SET_VALUE service primitive request. The format of a LM_SET VALUE service primitive operation is as follows:

parameter id
length of parameter value
parameter value

The operation information field for a LM_SET_VALUE service primitive operation specifies the value of the parameter id corresponding to the parameter desired. Only one parameter id may be specified. The parameter value will contain the new value for the parameter. The set confirm will contain status on completion of the operation. Refer to the proper layer specification for more information on a particular layers parameter id's. The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad parameter id	(xx)
Bad layer internal selector	(xx)
Bad parameter value	(xx)

TBU

6.2.3.8 LM_ACTION SERVICE PRIMITIVE

Action provides the ability to perform layer specific functions as well as operations common to all layers. There are four common Action requests which are supported by all layers, Update state, Create, List, and Test. While each layer performs these action operations in a unique fashion, they are a common set of requests to any layer management function. Update state instructs layer management functions to change the state of the specified object or component. Create provides the ability to create a new object or component within a layer. The list action operation causes the layer management to return a list of all objects or components of the specified type. The test action operation can only be performed on objects and components in the locked state and cause the execution of a unique layer test procedure. The operation information field for each action operation is unique and described below.

6.2.3.8.1 UPDATE STATE

Update state provides the ability to control the state of a component or object within a layer. The definition of the states for each layer can be found in section xxxxxxx, of this document. The format of the operation information field for an update state action request is as follows:

Action operation
State|Substate

The action operation field must specify the update state operation. The state and substate immediately follow.

update state action operation (xx)

state		8 bit integer
locked	(03)	
enabled	(04)	
disabled	(05)	
test	(06)	
down	(07)	
shutdown	(08)	
in-use	(09)	

substate		8 bit integer
reset	(01)	
halted	(02)	
loaded	(03)	
started	(04)	
operational	(05)	

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
Bad state	(xx)
Bad substate	(xx)
Illegal state change	(xx)

TBU

6.2.3.8.2 CREATE

The create action operation allows new objects or components to be created within a layer. The definition of the information required to create an object within a layer is specified in detail in that layer managements component description. The format of an action create request is formatted as follows:

Action operation
Length of Create Information
Create Information

The action operation field must specify the create action operation. The create information is unique to the object or layer it is defining.

create action operation (xx)

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
Bad create information	(xx)
TBU	

6.2.3.8.3 DELETE

The DELETE action operation allows objects or components to be destroyed within a layer. The definition of deleting an object and whether that operation is possible is specified in that layer managements component description which defines that object. The format of an action create request is as follows:

Action operation

The action operation field must specify the DELETE action operation. No additional information is required

delete action operation (xx)

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
TBU	

6.2.3.8.4 LIST

The LIST action operation provides the ability to obtain a list of all objects or components which meet the internal layer selection criteria. The confirm contains name, class, type, venue and state for every object meeting the criteria. The format of an action list operation is as follows:

Action operation
Number of objects
Object 1 selection parameters
//
Object n selection parameters

The action operation field must specify the LIST action operation. No additional information is required

list action operation	(xx)
number of objects	16 bit integer. Request value is a don't care.

for each object listed
name
8 ASCII characters. Null when not to be used

class
8 bit integer. Unique class numbers have been assigned for each type of object and component across all layers.

Controller	(13)
System Management Administrative Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)
Transport Connection	(xx)

type
4 ASCII characters. Unique type is assigned for each class of object.

LACS Controller	(LNCT)
SMAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	
TSAP	
Transport Connection	

venue
8 bit integer

any	(0)
local	(1)
image	(2)

state
8 bit integer

locked	(03)
enabled	(04)
disabled	(05)
test	(06)
down	(07)
shutdown	(08)
in-use	(09)

substate
8 bit integer

reset	(01)
halted	(02)
loaded	(03)
started	(04)
operational	(05)

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
TBU	

6.2.3.8.5 TEST

The test action operation provides the ability to perform layer specific tests. The definition of such tests can be found for each layer can be found in section xxxxxxx, of document, xxxxx. The format of the operation information field for an update state action request is as follows:

Action operation
Test parameter
Length of Test Data
Test Data

The action operation field must specify the test operation. The test parameter is provided for every test as well as test information. The nature and definition of this information is contained in the xxxxxxxxxx document.

test action operation	(xx)
test parameter	16 bit integer
length of test data	16 bit integer
test data	unique for each test operation.

The values of the status id returned in the confirm can be

Success	(00)
Not supported	(xx)
Bad action operation	(xx)
Bad layer internal selector	(xx)
Bad test parameter	(xx)
Bad test	(xx)
TBU	

For more information on the available commands to each layer, refer to the proper layer instance component specification.

6.2.4 Event Messages

The event message received by system management are of the following format:

0	15
message header	
Layer Info (layer/sublayer)	
Layer Instance (CT# and LI#)	
layer internal selector	
Event status	

The message header is the standard message header used for the LAN process messages. It contains the message type for event and a null return mailbox id. The SM layer instance is responsible for the memory containing the message once it is received.

The layer info and layer instance are used to identify the layer server which generated the event.

Layer Info — 1 word.

- layer — Bits 0-7. 8 bit integer. Value restricted between 0-7.
  - Management (0)
  - Physical (1)
  - Link (2)
  - Network (3)
  - Transport (4)
- sublayer — Bits 8-F. 8 bit integer. Value restricted between 0-1. For Link only, all other layers will have a zero value.
  - MAC (0)
  - LLC (1)

Layer Instance — 1 word.

- Controller Number — Bits 0-7. 8 bit integer. Value restricted between 0-F.
- Layer Instance Number — Bits 8-F. 8 bit integer. Value restricted between 0-7.

The layer internal selector describes the selection parameters of the object or component within the layer this event was involved. It contains the following fields:

name 4 words
8 ASCII characters.
class 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero. Unique class numbers have been assigned for each type of object and component across all layers.

Controller	(13)
System Management Administrative Function	(16)
MAC (physical line)	(04)
LSAP (logical line)	(05)
SNSAP	(xx)
TSAP	(xx)
Transport Connection	(xx)

type 2 words
4 ASCII characters. Unique type is assigned for each class of object.

LACS Controller	(LNCT)
SMAF	(8021)
MAC	(8023)
LSAP	(8022)
SNSAP	(SNCP)
TSAP	(CLS4)

venue 1 word
8 bit integer in bits 8-F. Bits 0-7 must be zero.
local (1)
object state 1 word.
state Bits 0-7. 8 bit integer

locked	(03)
enabled	(04)
disabled	(05)
test	(06)
down	(07)
shutdown	(08)
in-use	(09)

substate Bits 8-F. 8 bit integer.

reset	(01)
halted	(02)
loaded	(03)
started	(04)
operational	(05)

Event status contains two fields, the event status code and event status info. The event status code will identify the particular source of the event and a event status id indicating the reason for event. Additional status information may be specified by event status info. Event status info is any event specific status information. The first word of event status info indicates the size of the buffer containing event status info (including the status info size word) followed by event status info data. The format of the event status field is as follows:

EventStatus code 1 word

Eventsource Bits 0-7. 8 bit integer. Set by layer management.

management	(0)
physical(MAC)	(1)
link(LLC)	(2)
network	(3)
transport	(4)

EventStatus ID Bits 0-7. 8 bit integer, unique values assigned for each layer. Set by layer management.

EventStatusInfolng 1 word

16 bit integer, indicates the number of bytes contained in event status info data. Set by layer management.

EventStatusInfoPtr 2 words

Pointer to additional layer specific information. Must point to memory allocated in operation information. Set by layer management.

TBD

6.2.5 Procreate Layer Processes

The procreate request to create the layer instances on the LACS will specify the layer instance which it has been assigned. The format of the procreate call is as specified in reference [15].

layer instance data strucutre pointer
: the common data structre must specify that layer management has been invoked and what layer instance number has been assigned this process. The layer instance number must be between 0-7.

6.2.6 EXTERNALLY DEFINED MESSAGES

The SM layer instance must also issue messages to other LACS processes. These include the I/O Dispatcher, the DMA process, and Bridge kernel messages. The messages required by system management are as follows:

I/O Dispatcher message formats

- IOLD message format
- Function code mailbox directory registration message format

DMA message formats

- LCB to L6 message format
- LCB to LACS message format

Bridge Kernel message formats

- Breceive
- Resolve
- Procreate
- Prorun
- TBU

6.3 LACS INITIALIZATION

The LACS board must be properly initialized in order for the LACS system management layer instance to service requests. The LACS board initialization is performed in three steps. The first step is only performed during system start-up. In this step the megabus address of the controller and the adapters' system ids are verified against the parameters specified in the configuration directives. Step two involves the loading of LACS software into the LACS board memory. The third and final step is to initialize the Bridge kernal, LLC software and System Management software residing in the LACS. This is done by issuing a 'Start I/O' I/O command. These steps are performed due to the first activate local SAP request from a L6 user.

The system management layer instance is created by the kernel initialization process. When it is first created the system management layer instance must allocat memory and initialize its data structures. Then it registers a pointer to the begining of its data structures with the kernel. It also registers a function code mailbox directory with the IO Dispatcher and allocates memory for message space when no memory is available. The system management layer instance then suspends operation until a Start I/O message is received from the I/O Dispatcher. The Start I/O information is used to create and run other layer instances on the LACS.

6.4 TERMINATION REQUIREMENTS

Termination requirements are defined in Reference [1] Local Area Controller H/W EPS and in reference [15] Kernel and Support software ESPL Software Technical Reference Manaual, Vol.1. In addition to these requirements, the system management layer instance attempts to report any terminiation of the LACS software to the L6 system management layer server.

6.5 ENVIRONMENT

This software is intended to be executed on a LACS controller. It must be present with at least one LLC, MAC and Megabus interface software processes. The revisiopn and hardware requirements are to be added at a later date when the LACS hardware, firmware and the Bridge kernel receive an identification number.

6.6 TIMING AND SIZE REQUIREMENTS

To be supplied at a later date.

6.7 COMPILATION AND LINKING

All source code is compiled and linked through the use of the Bridge C language development utilities. These facilities are avalilable only in an Unix environment. They are currently avalaible on a ALTOS micro computer of the L6 running Unix. C source files are located in the System Management LAN directory, xxxxxx. Bridge development defoined makefile are used for all compilations and links. The following makefile is used for system management compilations:

TBS

The following makefile is used to link all routines:

TBS

The list of all source and include files used for linking is:

TBS

6.8 TESTING CONSIDERATIONS

Since the product is new, all functions will be tested by the developer, and the software test function.

6.9 DOCUMENTATION CONSIDERATIONS

Documentation of this product follows the recommendations of the Honeywell software documentations guidelines. In addition, all code descriptions will be accompanied by a procedural design language description described in reference [4].

6.10 OPERATING PROCEDURES

The LACS system management must receive a Start I/O request before any other requests can take place. There should always be an event LCB available to report events. The SM layer instance will use two event LCBs, one to report a catastrophic error and one to report all other events.

6.11 ERROR MESSAGES

All errors are assigned a class as specified in the LAN Software EPS. The system management error messages have been assigned the following classes:

Unreportable Catastrophic Error Class Messages

TBS

Reportable Catastrophic Error Class Messages

TBS

Fatal Operation Error Class Messages

TBS

Non-Fatal Operation Error Class Messages

TBS

Recoverable Error Class Messages

TBS

SM Protocol Error Class Messages

TBS

LACS Error Class Messages

TBS

7.0 LACS INTERNAL DESCRIPTION

7.1 LACS Internal Overview Description

The LACS system management layer instance receives requests from the L6 system management layer server in the form of messages describing LCBs which contain request and control information. The LCB is issued by the L6 system management layer server to the LDMS routine. The LDMS routine in turn issues an IOLD instruction across the megabus to the I/O Dispatcher firmware routine in the LACS. The I/O Dispatcher receives the issued IOLD and acknowledges the IOLD. The I/O Dispatcher then issues the IOLD information in a message to system management layer instance. The IOLD is queued up for the LCB handler which requests the DMA firmware to actually transfer the LCB to LACS memory. At the completion of the DMA transfer, the DMA interrupt firmware delivers the LCB to the system management request mailbox as part of the message. System management interprets the request in the LCB and requests the DMA service to copy the SM PDU into the LACS if there is one. System management services can be categorized into three types of services, initialization services, user request services, and system management "layer management". Initialization services provide the ability to initialize the LACS board, and layer object or components. This includes the ability to load and dump the LACS memory as well as initializing the hardware and data structures associated with each of the defined components. User request services provide the ability to service request PDUs and issue a response PDU. It also supports event notification services. Event notification provides the ability to notify system management of significant events detected by a layer management function. System management layer management provides control over the controller object and the system management administrative function.

In addition to servicing requests initiated from the L6 system management layer server, the LACS system management layer instance must also provide the ability to accomplish administrative and maintenance services with remote system management entities. This requires the ability to initiate requests to remote system management entities as well to respond to requests from a remote system management entities. All services available to a local system management entity are available to a remote system management entity. This ability will not be possible in the initial release.

7.1.1 Initialization Services

The initialization services provided by system management are concerned with preparing system management to accept requests for service. At the initialization of the controller object by the L6 system management layer server, the Bridge kernel, MAC firmware, LLC software, system management, and any other layer software must be downloaded and initialized. The system management initialization service is accomplished in two parts. The first is due to the execution of the kernel initialization process which acts as the parent process for all other LACS processes. It creates the system management process and megabus interface processes from information downloaded to the LACS in the kernels system initialization table (refer to reference xxxxx). The system management process must initialize any data structures required, register a pointer to the data structures with the kernel, and prepare to receive requests for its services. This is accomplished through the call to the kernels Breceive process. System management process will be awakened when there is a message in any of its mailboxes and there is no higher priority process requesting to be executed. System management messages can be addressed to the following mailboxes; L6 request mailbox, layer management interface mailbox, and the system management event mailbox. The implementation should allow the addition of new mailboxes for future growth. The L6 request mailbox is used the DMA firmware to deliver messages containing LCBs issued by the L6. The layer management interface mailbox and the system management event mailbox provide the communication path between system management and the different layer management entities.

The very first message system management must receive is from the event mailbox with a Start I/O message from the I/O Dispatcher process. The L6 system management layer server issues it to the LACS board during its' intialization sequence. The LACS megabus interface firmware recieves the request and uses it to initate execution of the software. The I/O Dispatcher process than delivers it to the system managment layer instance. The initialization services request the DMA services to fetch the LCB for the Start I/O. In the Start I/O LCB is information on which layer processes are to be created and identifies the layer instance number to be assigned to each layer. System management creates each process and passes the process the assigned layer instance number. A prorun request is then made to start the process running and the system managment layer instance stores the mailbox id for each process in the layer instance mailbox directory. The initialization services then wait for an event message from all layers to indicate they are ready. The LCB is then completed back to the L6.

7.1.2 USER REQUEST SERVICES

User request services support receiving and transmitting SM PDUs. User request supports the IEEE802 defined primitives, Get, Set, and Action. The Get and Set operations includes configuration parameters and statistical counters, variables and meters. The action operation supports requests to update the state of an object, create or delete an object, and run specialized layer tests. When a SM PDU request is received by system management, it determines in what entity and what layer the request is intended for, from the routing information in the SM PDU. System management in turn issues a request to the proper layer management service to perform the desired operation. The layer management will return the message to system management at the completion of performing the operation. The message contains any results as well as status indicating whether the operation was successful or not.

System management event notification services provide the means to notify the L6 system management layer server of significant events occurring within the LACS. The event can be detected by the LACS system management process or by any other layer management processes and reported to the LACS system management through its event mailbox. The L6 system management layer server is responsible for issuing an event LCB to the LACS so that the LACS system management layer instance may issue an event indicate PDU to the L6 user. When an event LCB is not available at some instant , the LACS system management insures that all events are queued on its mailbox to be eventually delivered.

7.1.3 System Management Layer Management Function

The system management administrative function provides the same type of service as other layers "layer management" function. It is responsible for the control of the system management administrative function and the LACS controller object. The system management request services calls the system management administrative function directly. The SM "layer manager" can access its tables directly and issues message to the process responsible for the controller object. Service request to the controller are routed to the "controller layer management" process. The following functions can be performed on the system management administrative function and the controller:

Get All Attributes of System Management Administrative Function
Get All Statistics of System Management Administrative Function
Get All Attributes of LACS Controller
Get All Statistics of LACS Controller

7.2 LACS SUBFUNCTION DESCRIPTION

7.2.1 LACS MANAGER MODULE DESCRIPTION

NAME lacs_manager

PURPOSE

The lacs_manager module provides the IEEE System Management functionality required for LACS operation. This includes the proper initialization of the lacs_manager process, handling of LCB requests from the L6, interfacing with other layer management processes, and servicing of events.

DESCRIPTION

The lacs_manager must first insure that the initialization of itself as a process is completed. This is accomplished through a call to the module, sm_init. At the return of sm init, the lacs_manager will begin the management of mailboxes which can invoke lacs manager services. The lacs manager processes each request for service sequentially, that is, a request is always processed to completion before another service request is attempted. The system manager is therefore required to issue a clock alarm message in the event that an expected message is not returned. The lacs_manager provides only two interfaces for which a request can be made, the IOLD message mailbox and the system management event mailbox. A third interface for a remote SMDSI interface to each layer management entity is under study for future implementation. The lacs manager will open both the IOLD message mailbox and the system management event mailbox and suspend its processing until a message is received by either one. When a message is received, both mailboxes are shutoff for notification of further messages. The lacs_manager will make a call to either the LCB receive module or to the event_handler module for processing of the message. At the completion of servicing the message request both modules return to the lacs_manager with an indication of success or failure and reason for failure. Upon successful completion of the request, both the IOLD message and event message mailbox are then turned and the lacs_manager again suspends operation until the next message arrives in one of the mailboxes.

INPUTS

IOLD message.
EVENT message.

OUTPUTS

None.

REQUIRED MODULES

sm_init Provides the initialization of the lacs_manager as a bridge kernel process performing IEEE 802 system management functions.

LCB_receive Handles IOLD request messages and fetches LCBs from L6 memory. It then requests the LCB be delivered to the proper module.

event_hndlr Receives and processes all events occurring on the LACS board.

RESTRICTIONS

All messages contained in mailboxes defined by modules in the bound unit for the lacs manager will be destroyed and the allocation of memory is unknown.

7.2.2 INITIALIZATION MODULE DESCRIPTION

NAME

sm_init

PURPOSE

The sm_init module is responsible for the initialization of the lacs_manager as a process operating with the bridge kernel. This includes the allocation of memory, the registering of mailboxes with the kernel and IO Dispatcher, and identifying mailbox ids of other processes.

DESCRIPTION

The sm_init module is called by the lacs_manager immediately after the kernel has created the lacs_manager as a process. The sm_init requests the kernel to allocate memory for emergency messages to be delivered to kernel services and to the DMA software. It also requests space for the system management data structures and registers a pointer to them with the kernel. After the allocation of memory has been completed, the resolution of mailbox ids and registration of ids is performed. The lacs_manager has only three mailboxes with which it can receive messages from other processes which are not ackowledgements or responses to a previous lacs_manager request. All mailboxes must first be registered as well known mailboxes with the kernel and the id of each must be obtained through a resolve kernel call. The first is the IOLD message mailbox, it must be registered with the IOLD Dispatch software for each of the eight layer instance channels interfaced with the Megabus. The event message mailbox is used to receive unsolicited event messages from all other processes. The default lacs manager mailbox is reserved for kernel and alarm type messages to the lacs_manager. In addition to these three mailboxes, the lacs_manager requires one additional mailbox for all other message transfers. This is the mailbox which is specified as the return mailbox in request messages to other processes. The sm_init module must also obtain the mailbox ids of the DMA software process and IOLD Dispatch processes which are present on the LACS. The sm_init must also initialize the event enable table maintained by the event_handler module. Initially all event reporting to a system management application (NAD, ADAP) is disabled. Catastrophic error events will always be reported to the system management in the L6. The system management state can only be updated through the receipt of a Start IO LCB or a update state action request to the LACS manager for the system management entity. The system management entity state is maintained in the system management attribute table. The routine then performs a Breceive on the event mailbox. A Start I/O message is the only valid message which will be serviced at this point. When the Start I/O is received, the DMA service is requested to copy the Start I/O LCB over to LACS memory. The LCB contains information on which processes must be created. As system management creates each layer process it passes the layer instance number to newly created layer process. The layer instance number is used by each layer to register with the IOLD dispatcher. The init routine it stores the mailbox ids of each created layer process for later use. The sm_init module must also obtain the mailbox ids of the DMA software process and all layer management processes which are present on the LACS. The Physical layer management mailbox, a MAC layer management mailbox are a special case. System management registers with MAC as described in the LACS Hardware EPS. A table is maintained for each adapter on the ids of its layer management processes. At the completion of initialization, the Start I/O is completed and a request made to the LCB request routine to post it to the L6.

INPUTS

OUTPUTS
None.

REQUIRED MODULES

RESTRICTIONS
None.

7.2.3 ROUTER MODULE DESCRIPTION

NAME

sm_router

PURPOSE

This routine processes all request made to the system manager in the form of a SM PDU.

Description

The system management routing routine receives a pointer to an LCB for a system management request. It calls the routine, Validate PDU, to validate that the PDU is a properly formed and a Honeywell supported IEEE802 system management PDU. An invalid PDU is returned to the user as a response PDU in the LCB. The Decode PDU routine is called to obtain the routing and operation information. This information is passed to the PDU to the layer manger routine specified by the routing information. The status from layer manager routine will indicate whether the operation completed successfully or not. The routine, layer management to PDU, is called to format the response PDU. The LCB is posted back by calling the routine, Post LCB. Status is returned to indicate a valid PDU, a badly formed PDU, or an unsupported operation.

Inputs

Pointer to the LCB.

Outputs

None.

Required Modules

Post LCB routine
  Posts the LCB back to user.
PDU Validation routine
  Checks the format of a received PDU.
PDU Decode routine
  Obtains the routing and operation information from a PDU.
PDU to Layer Manger Formatter
  Formats a request for a Layer management function.
Layer Manager to PDU Formatter
  Interprets the results from a layer management function.

7.2.4 LMI INTERFACE MODULE DESCRIPTION

NAME lmi_interface

PURPOSE

Issues layer management service primitive requests to layer management processes.

DESCRIPTION

This routine receives a pointer to buffer containing a layer management request and a pointer to routing information obtained from a SM PDU. A check is made to determine which layer manager it is intended for. A message is issued to the mailbox for the proper layer manager for all layers except system management. Processing is then suspended until the layer manager returns the message. If the request is for system management, then a call is made to the SM layer management services routine. At completion of the request the results are returned in buffer supplied by the calling routine.

INPUTS

OUTPUTS

None.

REQUIRED MODULES

RESTRICTIONS

None.

7.2.5 PDU Validation routine

NAME SM PDU Validate

PURPOSE
Checks the format of a PDU.

DESCRIPTION

INPUTs

OUTPUTs

REQUIRED MODULES

RESTRICTIONS
None.

7.2.6 PDU Decode routine

NAME SM PDU Decode

PURPOSE
Obtains the routing and operation information from a SM PDU.

DESCRIPTION
This routine examines a SM PDU to obtain the routing and operation information. The routing information is obtained first and must be of a fixed format as described for a Honeywell specific PDU. The operation information is then obtained. The format of the operation information will vary depending on the desired object.

INPUTs

OUTPUTs

REQUIRED MODULES

RESTRICTIONS
None.

7.2.7 PDU Formatter routine

NAME SM PDU Formatter

PURPOSE
Creates a SM PDU from a buffer in a specified format.

DESCRIPTION
This routine receives a buffer of a SM PDU. The request info or response info record of the PDU is created first. It requires that any records contained within the request info or response info record be already formatted. The ResourceId record is then formatted and finally combined to form the completed request or response PDU. In the case of an event indicate PDU, the event info filed is created first then the event PDU record.

INPUT

OUTPUT

REQUIRED MODULES
None.

RESTRICTIONS
None.

7.2.8 PDU to Layer Manger Formatter

NAME PDU to Layer management formatter

PURPOSE
Formats a request for a layer management function.

DESCRIPTION
This routine creates a layer management request block to be created from the information contained in the request PDU. It recognizes the basic operations and formats it into as described in the layer management interface description [in section 6.2.3.

INPUT

OUTPUT

REQUIRED MODULES
None.

RESTRICTIONS
None.

7.2.9 Layer Manager to PDU Formatter

NAME Layer Manager to PDU routine

PURPOSE

Create a response SM PDU from results obtained in a layer management result block.

DESCRIPTION

This routine creates a SM response PDU from information contained in the request PDU and the results obtained from the layer management function. The routing information and the exchange id are copied from the request PDU. The result of the operation is assumed to be in a IEEE802 format and joined with the routing portion of the PDU.

INPUT

OUTPUT

REQUIRED MODULES

RESTRICTIONS

None.

7.2.10 EVENT HANDLER MODULE DESCRIPTION

NAME

sm_event_hndlr

PURPOSE

DESCRIPTION

INPUTS

name. desc

OUTPUTS

None.

REQUIRED MODULES

RESTRICTIONS

None.

7.2.11 SM LAYER MANAGEMENT SERVICES MODULE DESCRIPTION

NAME sm_services

PURPOSE

This routine services all request for the system management layer manager.

DESCRIPTION

This routine is called by the layer management interface routine to process system management layer manager requests. A check is made to determine if the request is made for the controller object or the system management administrative object. If the request is for the controller object, then a message is issued to the controller management code and then a Breceive is done until a completed response is returned. If the request is to perform an operation on the system management administrative function then the proper routine is called. The results are returned in the same format as all layer management messages.

INPUTS

OUTPUTS

REQUIRED MODULES

RESTRICTIONS

None.

7.2.12 GET SMAF SERVICE ROUTINE

NAME Get SMAF Service

PURPOSE

Reads the attributes and statistical values of the system management administrative function.

DESCRIPTION

This routine receives a parameter id and returns the value for that parameter. A table is kept which maps the IEEE802.1 parameter id to the position in the system management attribute list and the size of the value to be read. There are two special parameter ids, all attributes and all statistics. For either of these two parameter ids, a IEEE 802 formatted parameter list containing all system management attributes or statistics is created and returned as the result of the request. The format of these lists is defined in section 2.2.3.

INPUTS

Outputs

Required Modules

Restrictions

None.

7.2.14 LCB Receive Routine

NAME LCB_receive

PURPOSE
This routine will copy an LCB and the buffer(s) it specifies into LACS procedure memory.

DESCRIPTION
This routine receives IOLD information and allocates memory in LACS procedure memory. It then requests the DMA services to transfer the LCB across the megabus and suspends operation until the message is returned. The routine then examines the LCB function code to determine if the buffer must also be moved across. The buffer is not moved for a LCB with a function code specifying an event LCB. Otherwise memory will be allocated and the LCB buffer will be moved across the megabus. While the DMA transfers the buffer(s), all system management operations are suspended until the operation is completed.

INPUTS

OUTPUTS

REQUIRED MODULES

RESTRICTIONS
None.

7.2.14 LCB POST MODULE DESCRIPTION

NAME LCB_post

PURPOSE

This routine posts a completed LCB and any required buffers to the L6.

DESCRIPTION

This routine receives the IOLD information on where in L6 memory the LCB must be completed. The routine first requests that the DMA services transfer the buffers across to the L6. All operations are suspended by doing a Breceive until the DMA services complete the operation. When the buffer transfer has been completed the LCB is then transferred, again operations are suspended waiting for the DMA to return the message.

INPUTS

OUTPUTS

REQUIRED MODULES

RESTRICTIONS

None.

7.2.15 TICTOC IOLD ROUTINE DESCRIPTION

NAME tictoc

PURPOSE

This routine processes a tictoc request. It increments the current tictoc count by one and then post the LCB back to the L6.

DESCRIPTION

INPUTS

OUTPUTS

REQUIRED MODULES

RESTRICTIONS

None.

7. 3 LACS FUTURE DEVELOPMENT AND MAINTENANCE

To be supplied at a later date.

8. 0 LACS SYSTEM MANAGEMENT PROCEDURAL DESIGN

To be supplied at a later date.

9.0 LACS SYSTEM MANAGEMENT ISSUES

1. How is a request to change the state of a physical line to be interpreted by LACS system management ?

   - should a distinction be made between setting the state of the MAC and Physical layers - this would be more 802 compatible or is it not worth the trouble (will end up with a MAC and Physical entity or follow 802. 2 with just a system entity?) - Right now MAC and physical are mapped into the physical layer.

2. What are the requirements for editing a dump of LACS memory?

3. What are the requirements are formatting of LACS bound units.

A.1 SM PDU OVERVIEW AND EXAMPLE

All IEEE802 system management PDUs are specified and constructed according to the syntax described in the X409 standard. The X409 standard is a description of a standard notation and a standard representation to be used to describe information passed between two applications. This tutorial attempts to explain the use of X409 in the description of SM PDUs. This is not meant to be a tutorial on the X409 standard itself.

All X409 records contain an identifier field and a length field (a record or field is not a X409 definition, X409 refers to data elements as a general description of information within a PDU, however it was felt a record and field are intuitivley correct for most peoples understanding). A record can also contain a content field. The identifier is further partitioned into a class, form, and id code. There are four classes, Universal, Application, Context Specific, and Private Use. The universal class is used to identify records defined within the X409 standard definitions (i.e integer or IA5 string), the Application class is used to define records defined in the application it is being used in (in this case it would identify IEEE802 specific definitions), Context specific identifies records which must be interpretted based upon there position or context in a record, and finally there may is a private type to allow implementation specific records (this allows Honeywell specific records to be defined). IEEE802 SM PDUs are defined using only context specific record types.

Form describes whether or not there are any further records imbedded in this record. A record can be of two forms, a contructor or a primitive. A primitive record contains no further records and when a content is specified, the contents will specify some value to be associated with the given id code. A constuctor identifies a record containing another record or a series of records, its' content is more records.

The last field within an identifier is the id code. This distinguishes one record from another. All SM PDU records are context specific. The id code is therefore not unique between different records but unique only within the context of the record it is defined for (i.e. the id code 0 is used in more than one record type to describe different records). In practice this allows id codes to repeated for the definition of records defined for a constuctor form of record.

The length field specifies the total length of a record. X409 specifies three types of length, short, long, and indefinite. The short form specifies length in one byte (a byte is called an octet in standarnese) for records not exceeding a length of 128 octets. The long form allows the number of bytes describing the length to be up to 128 octets long. Our implemetation restricts the length of all SM PDUs to within a more resonable limit of 2 octets. The indefinite form allows a special end of contents record to terminate a record, this is not supported in our implementation.

System Management Component Specification

The attached pages provide a detailed description of the system management PDU record construction and format. Indention is used to show the position of a field within another, as well as the characters { and } to mark the beginning and end of a construct type field. Each octet is bounded as [octet value]. All records are context specific as shown in each identifier as a CS. The following examples represent system management PDU exchanges between a system management application (NAD, ADAP, or T&V) and the LAN SM server.

These abbreviations were in the following examples:

CS - Context specific
Construct - Constructor
Id - Id code
Lngth - Length

Example 1 - Get request of MAC parameters by the system manager application. The PDU issued by the system manager application and received by the LAN SM server is as follows:

```
[CS,Construct,Id=01][Lngth=0xx]          RequestPDU
{
. [CS,Construct,Id=01][Lngth=0xx]         GetRQ
. {
. . [CS,Construct,Id=00][Lngth=0xx]        ResourceID
. . {
. . . [CS,Construct,Id=01][Lngth=0xx]       LayerInfo
. . . {
. . . . [CS,Primitive,Id=00][Lngth=001]     Layer (management)
. . . . [0]
. . . . [CS,Primitive,Id=01][Lngth=001]     Sublayer (MAC)
. . . . [0]
. . . . [CS,Primitive,Id=02][Lngth=001]     LayerInstance
. . . . [10]                                 (Ct 1, layer inst.numb. 0)
. . . . [CS,Construct,Id=03][Lngth=0xx]     LayerInternalSelector
. . . . {
. . . . . [PU,Construct,Id=00][Lngth=0xx]    Selection Parameters
. . . . . {
. . . . . . [CS,Primitive,Id=00][Lngth=001]     Class (CT=13)
. . . . . . [13]
. . . . . . [CS,Primitive,Id=01][Lngth=008]     Name (CTRL01)
. . . . . . [43][54][52][4C][30][31][00][00]
. . . . . . [CS,Construct,Id=02][Lngth=0xx]     Object State
. . . . . . {
. . . . . . . [CS,Primitive,Id=00][Lngth=001]     State (Anystate)
. . . . . . . [00]
. . . . . . . [CS,Primitive,Id=01][Lngth=001]     Substate (Anysubstate)
. . . . . . . [00]
. . . . . . }
. . . . . . [CS,Primitive,Id=02][Lngth=004]     Type (LNCT)
. . . . . . [4C][4E][43][54]
. . . . . . [CS,Primitive,Id=04][Lngth=001]     Venue (Local)
. . . . . . [00]
. . . . . . [CS,Primitive,Id=05][Lngth=010]     Mappings (Null)
. . . . . . [00][00][00][00][00][00][00][00][00][00]
. . . . . }
. . . . }
. . . }
. . [CS,Primitive,Id=01][Lngth=002]        ExchangeId (0123)
. . [01] [23]
. . [CS,Primitive,Id=02][Lngth=002]        Access Control (0000)
. . [00] [00]
. . [CS,Construct,Id=03][Lngth=0xx]        ParameterList
. . {
. . . [CS,Construct,Id=01][Lngth=0xx]        Defined Parameter
. . . {
. . . . [CS,Construct,Id=00][Lngth=0xx]        Private Parameter
. . . . {
. . . . . [CS,Primitive,Id=00][Lngth=001]      Code (All Statistics)
. . . . . [4]
. . . . }
. . . }
. . }
. }
}
```

The response PDU issued by the SM server back to the system manager application is as follows:

```
[CS,Construct,Id=02,Lngth=0xx]              ResponsePDU
{
. [CS,Construct,Id=01][Lngth=0xx]           Sequence of ResponseInfo
. {
. . [CS,Construct,Id=01][Lngth=0xx]         GetRSP
. . {
. . . [CS,Construct,Id=00][Lngth=0xx]       ResourceID
. . . {
. . . . [CS,Construct,Id=01][Lngth=0xx]     LayerInfo
. . . . {
. . . . . [CS,Primitive,Id=00][Lngth=001]   Layer (management)
. . . . . [0]
. . . . . [CS,Primitive,Id=01][Lngth=001]   Sublayer (MAC)
. . . . . [0]
. . . . . [CS,Primitive,Id=02][Lngth=001]   LayerInstance
. . . . . [10]                               (CT 1, layer inst.num. 0)
. . . . . [CS,Construct,Id=03][Lngth=0xx]   LayerInternalSelector
. . . . . {
. . . . . . [PU,Construct,Id=00][Lngth=0xx]  Selection Parameters
. . . . . . {
. . . . . . . [CS,Primitive,Id=00][Lngth=001]  Class (CT=13)
. . . . . . . [13]
. . . . . . . [CS,Primitive,Id=01][Lngth=008]  Name (CTRL01)
. . . . . . . [43][54][52][4C][30][31][00][00]
. . . . . . . [CS,Construct,Id=02][Lngth=0xx]  Object State
. . . . . . . {
. . . . . . . . [CS,Primitive,Id=00][Lngth=001]   State (Anystate)
. . . . . . . . [00]
. . . . . . . . [CS,Primitive,Id=01][Lngth=001]   Substate (Anysubstate)
. . . . . . . . [00]
. . . . . . . }
. . . . . . . [CS,Primitive,Id=02][Lngth=004]  Type (LNCT)
. . . . . . . [4C][4E][43][54]
. . . . . . . [CS,Primitive,Id=04][Lngth=001]  Venue (Local)
. . . . . . . [00]
. . . . . . . [CS,Primitive,Id=05][Lngth=010]  Mappings (Null)
. . . . . . . [00][00][00][00][00][00][00][00][00][00]
. . . . . . }
. . . . . }
. . . . }
. . . }
. . . [CS,Primitive,Id=02][Lngth=002]       ExchangeId (0123)
. . . [01] [23]
. . . [CS,Construct,Id=03][Lngth=0xx]       ParameterList
. . . {
. . . . [CS,Construct,Id=01][Lngth=0xx]     Defined Parameter
. . . . {
. . . . . [CS,Construct,Id=00][Lngth=0xx]   Private Parameter
. . . . . {
. . . . . . [CS,Primitive,Id=xx][Lngth=002]  Total number of LCBs
. . . . . . [02] [33]                        issued (233 Hex)
. . . . . . [CS,Primitive,Id=xx][Lngth=001]  Number of LCBs Nak'd
. . . . . . [03]                             (3)
. . . . . . [CS,Primitive,Id=xx][Lngth=001]  Number of queued LCBs
. . . . . . [01]                             (1)
. . . . . }
. . . . }
. . . }
. . }
```