Software Component Specification

SYSTEM:	Level 6
SUBSYSTEM:	LACS Controller
COMPONENT:	Interface Software Memory Access
PLANNED RELEASE:	Rel 2
SPECIFICATION REVISION NUMBER:	Preliminary 0
DATE:	June 7,1985
AUTHOR:	K.Yu

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

TABLE OF CONTENTS

REFERENCES

[1] 60149766 Engineering Product Specification, Part 1, version G

[2] 60149817 Lan Software EPS-1

[3] 09-0016-00 ESPL Software Technical Reference Manual, Vol. 1, Kernel and Support Software (Bridge Communications, Inc.)

DEFINITION

CPU	Central Processor Unit
DMA	Direct Memory Access
IOLD	Input/Output Load
LAC	Local Area Controller
LAN	Local Area Network
LCB	LAN Control Block

1. INTRODUCTION AND OVERVIEW

This is one of several Interface Software Modules developed by the Hardware Develpment Group to provide an interface between the Communication Software and the Lacs hardware. Basically, it is a firmware module controlling hardware functions and yet is written in "c" language and is running under the control of the Kernel Operating System environment. The interface supports the data transfer between L6 and Lacs memories. This module can be viewed as an extension of the Kernel Service function.

1.2 BASIC PURPOSE

The function of the module is to honor requests to carry out DMA operations to move information between a buffer and/or lcb in lacs ram and one or more buffers in Level 6 memory. It supports scatter/ gather operation. It performs Level Interrupt function and deferred interrupt handling.

1.3 BASIC STRUCTURE

The module is organized into three segments. The initialization segment initializes the mc68440 chip for the dma channel. Register DMA data_transfer_complete interrupt routine with the Kernel. Creates a semaphore to protect the internal data structure. The main routine issues a breceive call to receive any DMA Memory Request messages from user. The last segment is entered upon receiving a message. It programs the associated hardware to carryout the DMA operations. It interrupts the Level 6 if an interrupt is required. It returns message to the user upon complete of the DMA function and is then ready for another message.

1.4 BASIC OPERATION

The module is running under the Kernel control. The initialzation and main entries are registered with the CSl file. The basic flow is that on the completion of the initialization segment it enters the main routine. This main routine issues a breceive call to the Kernel to receive a message from its mailbox. If a message is received it calls on the message processing function to program the hardware for the DMA function. Upon the completion of the DMA function it returns to the main routine for another breceive call. If there is no message in the mailbox the Kernel will suspend the module until there is one.

2. EXTERNAL SPECIFICATION

2.1 OWNED DATA STRUCTURES

The following message data structures are used for all communications between this module and external users.

"c" declaration

```
#define LCBIO struct lcbio
        struct  lcbio{
                MSGX    mx;     /* extended message header */
                caddr_t l6_addr;/* Level 6 addresses */
                short   range;  /* Level 6 range     */
                caddr_t ram_addr;/* lac ram address  */
                };

#define BUFIO struct bufio
        struct bufio{
                MSGX   mx;      /* extended message header */
                short  l6_buf_cnt;/* number of 16 buffers */
                L6_DES l6[9];   /* Level 6 buffer descriptor */
                };

#define BUFIOX struct bufiox
        struct bufiox{
                MSGX    mx;     /* extended message header */
                L6_LIST *l6ptr; /* pointer to 16 buffer list */
                };

#define BFLCBIO struct bflcbio
        struct bflcbio{
                MSGX    mx;     /* extended message header */
                L6_LIST *l6ptr; /* pointer to 16 buffer list */
                caddr_t l6_addr;/* Level 6 address */
                short   range;  /* Level 6 range */
                caddr_t ram_addr;/* Lac ram address */
                };

#define MSGX struct msgx
        struct msgx{
                MSG   m;        /* kernel header */
                short chnlev;   /* reserved */
                short ch_lv;    /* channel,cpu and int level */
                                /* bit 0-5 =channel,6-9 = cpu */
                                /* 10-15 = interrupt level */
                short rqsid;    /* user id */
                MBID  mbid;     /* return message mailbox id */
                short status;   /* return messge l6 status */
                };
```

```
type struct
        {
                caddr_t  address;/* Level 6 address */
                short    range;  /* range */
                }L6_DES;

type struct
                {
                short   16_buf_cnt;/* number of 16 buffers */
                L6_DES[9];         /* 16 buffer descriptor */
                }L6_LIST;

#define MSG struct MSG
        struct MSG{
                MSG    *m_fwd;   /* kernel queue pointers */
                MSG    *m_bwd;
                PID    m_sender; /* process ID of sender */
                BD     *m_bufdes;/* ptr to buffer descriptor */
                short m_prio;    /* message priority */
                short m_type;    /* user message type */
                };
```

One of the following message type must be used to send message to this module.

a. LCB_TO_L6
   This message type is to return lcb to Level 6 using LCBIO message structure.

b. LCB_TO_LAC
   This message type is to get lcb from Level 6 to lac memory using LCBIO message data structure.

c. BUF_TO_L6
   This type of message is to transfer data to Level 6 using BUFIO message data structure.

d. BUF_TO_LAC
   This type of message is to move data from Level 6 memory to Lac buffer memory using BUFIO message data structure.

e. BUFX_TO_L6
   This type of message is to transfer data from buffer memory to Level 6 memory using BUFIOX message data structure.

f. BUFX_TO_LAC
   This type of message is to move data from Level 6 memory to lac buffer memory using BUFIOX message data structure.

g. BFLCB_TO_L6
   This type of message is to move data from buffer memory to Level 6 memory and return lcb using BUFLCBIO message data structure.

2.2 EXTERNAL INTERFACES

All communications between the external users and this module is via Kernel message call. Seven type of messages are available for usage. Any process may send a message to this module's mailbox whose mailbox id is "MEMDMA". After the message has been processed it will be returned to the sender with status. The user must check the status to determine any errors associated with the returned message.

The module runs under the control of the Kernel Operation System. Therefore it uses the O.S. system resources that available to process. These resources are available via procedure calls. The followings procedure calls are used by this module.

- semacreate()
- regintrpt()
- resolve()
- breceive()
- semawait()
- sendmsg()
- semarelease()

Refer to Bridge Kernel Reference Manual Section 5.3 for definition of these procedure calls.

2.3 INITIALIZATION REQUIREMENTS

The "main" and "init" entries pointers are in the CS1 file. The mailbox id "MEMDMA" and the process priority also registered with CS1 file. When the Kernel is up this process will be part of the system initialization, mailbox id will be in the well-known table and the process will be spawned.

2.4 TERMINATION REQUIREMENTS

This process is always on the ready list. It may not be active because of waiting for resources or messages.

2.5 ENVIRONMENT

The process runs under the control of the Operating System. It responds to message calls and hardware interrupts.

2.6 TIMING AND SIZE REQUIREMENTS

The size of this process is approximately about 8k words.

2.7 ASSEMBLY AND LINKING

The source file is lac_ dma.c under the directory of /usr/dvlp/megabus The binary file lac_ dma.b must be linked with kernel /usr/dvlp/kernel to create a bound unit for loading.

2.8 TESTING CONSIDERATIONS

none

2.9 DOCUMENTATION CONSIDERATIONS

Since the source of this module is written "c" it is a self-explanatory. Comments in the heading of each main functions make it a stand-alone document.

2.10 OPERATING PROCEDURES

none

2.11 ERROR MESSAGES

When a message is returned to the user a status word accompanies with it. Each bit is decribed as follow:

0 - 7 = zero
8 = invalid format in the message
9 = dma hardware error
10 = non existence lac ram address
11 = lac ram parity error
12 = L6 memory yellow (not fatal)
13 = L6 non existent memory address error
14 = L6 megabus parity error
15 = L6 memory red

3. INTERNAL SPECIFICATION

3.1 OVERVIEW

The initialization and process creation is part of the system startup.Once the system is up this process is ready to receive messages from its own default mailbox. If a message is received, the process sets up the hardware to perform the dma function. This may involve multi-loading the mc68440 chip, megabus control registers, handling hardware DMA termination interrupts;it may have to interrupt Level 6 to return IOLDs or indicate certain actions to be taken by the CPU.

3.2 SUBCOMPONENT DESCRIPTION

3.2.1 OWNED DATA STRUCTURE

An internal data structure is allocated and shared between the dma main() whose function is to receive message and the hardware interrupt handler. The message receiver puts the original message into the data structure and sets up the action to be taken by the interrupt routine. Only one process can access the data structure at any given time. It is locked by the receiver and released by the interrupt handler using the combinations of procedure calls such as semacreate(),semawait() and semarelease().

3.2.2 RECEIVE MESSAGE

This routine should issue a brecieve procedure call to recieve from its default mailbox. The process will be suspended if there is no message available in the mailbox. However,if a message is received it calls dma message decode to see which one of the seven messages to act on. Upon the return of he message decoder this routine must check the data structure that has been released by the interrupt handler. If the data structure is released, then it looks for a message from interrupt handler to return a message back to the user before attempting to get another message from its mailbox.

3.2.3 MESSAGE DECODER

The main function of this routine is to decode the message into one of the seven possible messages. Those that cannot be recognized will be sent back to sender immediately and the data structure released. The primary function after the message type has been decoded is to setup the hardware registers,mc68440 chip, etc to carryout the DMA function. It also determines the action codes for the DMA completion interrupt handler. This action code may be to reload megabus control registers for multi buffer data transfer,scatter and gather function, or interrupt Level 6 cpu functions. Certain message formats must be checked to insure against conditions that might cause hardware errors. The megabus registers and mc68440 are designed to transfer data on a word basis. Therefore those messages that are involved with buffer descriptors must further check the odd even combinations of the Level 6 and Lacs buffers descriptor's addresses and ranges.

3.2.4 DEDICATED BUFFER

512 words in the buffer ram should be dedicated for use as a transition buffer to move data between the L6 memory and the microprocessor ram. The reason is that the the hardware can access the buffer ram only. This buffer is big enough for transfer of lcbs and small enough not to take away buffer memory from O.S.

3.2.5 LEVEL 6 INTERRUPT HANDLER

When a DMA request calls for an interrupt the Level 6, this process will inspect the pending interrupt queue for that cpu. If there are any pending interrupts in that queue of equal or higher priority, the interrupt will not be attempted but added to the queue.If it is higher than any entry, it will be attempted and if accepted the message is returned to sender; if rejected it will added to the queue. Queue members having a different levels will be linked vertically according to priority,and queue members having an identical level will be linked horizontally. The forward and backward pointers in the message header are used for this purpose.

3.2.6 DMA TRANSFER COMPLETE INTERRUPT HANDLER

This routine looks for the interrupt action code to determine what it is supposed to do. It may continue to setup registers for further dma function; it may interrupt Level 6 and finally it may inform the message receiver to return messages back to their senders.

3.2.7 LEVEL 6 INTERRUPT PENDING RETRY HANDLER

When the Level 6 cpu switches level it sends a Resume Interrupt signal over the megabus. The hardware recognizes this and causes a level 1 68000 interrupt. The handler will attempt to send the interrupt at the top of queue for each cpu. This module can handle up to 16 cpu. Any interrupts which are accepted are removed from the queue and passed to the message receiver for return. Any interrupts that are rejected are left in the queue.

3.3 FUTURE DEVELOPMENT AND MAINTENANCE

as required

4. PROCEDURAL DESIGN

none