# Software Component Specification

| | |
|---|---|
| SYSTEM: | MOD400 |
| SUBSYSTEM: | LAN Facility |
| COMPONENT: | Interface S/W - I/O Dispatcher |
| PLANNED RELEASE: | Release 4.0 |
| SPECIFICATION REVISION NUMBER: | 1 |
| DATE: | Aug. 16,1985 |
| AUTHOR: | K.Yu |

This specification describes the current definition of the subject software component, 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

| | |
|---|---|
| IO | Input/Output |
| IOLD | Input/Output Load |
| CPU | Central Processor Unit |
| LCB | LAN Control Block |

# 1. INTRODUCTION AND OVERVIEW

## 1.1 BACKGROUND

This is one of several Interface Software modules developed by the Hardware Development Group to provide an interface between the Communication Software and 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.

This interface supports a physical I/O interface between the Level 6 and Lan controller. All IOLDs that are destined for this controller are assembled and dispatched to various layers which require communication with Level 6. This module can be viewed as an extension of the Kernel Service function.

## 1.2 BASIC PURPOSE

The basic function of the module is to recognize the ioIds that are issued by a CPU and dispatch them to its destination according to an entry in the channel table which is previously setup by the user.

## 1.3 BASIC STRUCTURE

This module essentially consists of three sections.The first section initializes the IOLD queues and hardware enviroment for accepting IOLDs from Level 6. The second section contains the main routine looking for messages from its own mailbox to perform mailbox registration and presentation of channel mailbox information that specifies where an IOLD may be dispatched. The third section is the IOLD interrupt handler. It assembles the incoming IOLDs from the IO queues into messages and dispatches them to their destination according to their channel mailbox table entry. It also manages the queues for the hardware to continue to accept IOLDs from the megabus.

## 1.4 BASIC OPERATION

The process starts with the initialization code. It programs the dma chip with two queue pointers and sets up the control registers ready for IOLDs. The queues are cleared to zero. The interrupt routine to handle the next IOLD queue is registered with the Kernel. Now it is ready to accept messages from the user to register their mailbox entries. When an IOLD arrives the channel associated with it will be used to pickup the entry. It is this entry that the IOLD will be dispatched to. During the registration no IOLD will be allowed until an event registration is completed.Then the start_io ioId information will be sent to the event mailbox. Only this time, a signal is given to hardware to enable megabus IO reception. The operation is under the Kernel O.S. environment.

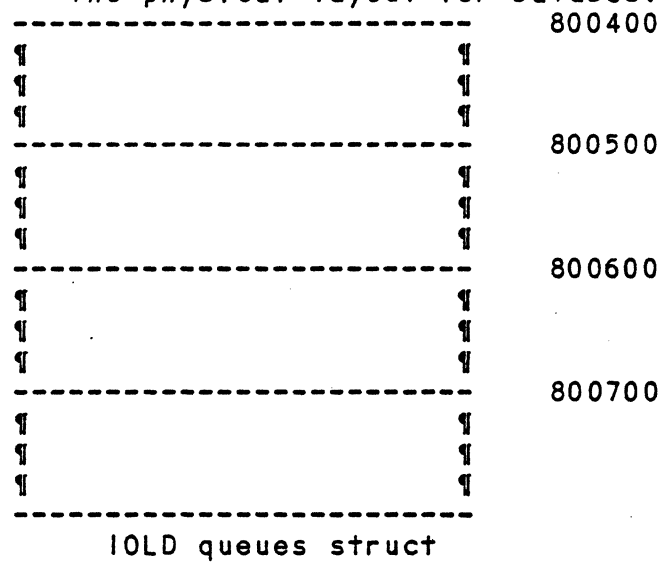
## 2. EXTERNAL SPECIFICATION

### 2.1 OWNED DATA STRUCTURES

A data structure is created to handle I/O orders. The "c" Declaration:

```
struct §   short ch_fc;       /* channel number and function code */
           long  *l6_address;/* Level 6 memory address */
           short range;       /* range */
           †IOITEM
struct          §
          IOITEM io_list0[32];      /* first io queue list */
          IOITEM io_list1[32];      /* second io queue list */
          IOITEM io_list2[32];      /* third io queue list */
          IOITEM io_list3[32];      /* fourth io queue list */
              †DataDes;
```

The physical layout for DataDes:




IOLD queues struct

A mailbox table for 64 channels and an Event is created during startup. This table contains all mailbox ids. When an Iold comes the channel associated with this Iold will be used to pickup the mailbox id from the table and dispatch to that mailbox. An zero entry will dispatch to the event mailbox. A read or write command to the table is permitted any time.

2.2 EXTERNAL INTERFACES

The IO Dispatcher is driven by two external events. A channel mailbox registration(write channel mailbox id) or read channel mailbox id is accomplished with the O.S. sendmsg call. The message type is register normal,register event, read normal,or read event. The "c" Declaration for the message:

```
struct chan_mbid $
        MSG   m          /* kernel message header        */
        short chan_nmb   /* channel number bit 10 - 15   */
        MBID  mbid       /* mailbox id for this channel  */
        MBID  ret_mbid   /* return mailbox id            */
        short ret_code   /* normal = 0,undefined type=-1*/
        t;
```

When an event mailbox is registered for the first time the start I/O iold being queued up by the prom firmware is sent to the event mailbox and hardware is enabled to accept any IO oders. When the cpu issues an IO order to iac controller the hardware put it into the queue and cause a level 4 68000 interrupt. The IO Dispatcher is activated and uses the following "c" declaration to dispatch the iold:

```
struct ioldmsg$
 MSG   m          /* kernel message header   */
 short chanfc     /* channel number and function code*/
                  /* bit 0 -9 channel# 10 - 15 = IOLD*/
 long  *16_addr   /* icb address in Level 6      */
 short icb_inf    /* protocal id and icb size    */
                   t;
```

2.3 INITIALIZATION REQUIREMENTS

The initialization and main entries are part of the system table,CS1 file. When the system is up this process is also ready. The mailbox id for this process is "IODISP". The mailbox table is created and is cleared to zero. IO queues are setup and cleared to zero. The dma chip is setup and is ready. However, Registration of the interrupt routine to handle IO with the O.S. must done when the system is powered up. No IO orders would be accepted until event registration is completed first.

2.4 TERMINATION REQURIEMENTS

There is no termination requirement for this IO Dispatcher as long as the Lacs board is up.

## 2.5 ENVIRONMENT

The code is in "c" control megabus interface and operate under the environment of the Bridge Kernel.

## 2.6 TIMING AND SIZE REQUIRMENTS

The size of this module is approxmately 2k words.

## 2.7 ASSEMBLY AND LINKING

The source code name for this is lac_io.c and is in the directory /usr/dvip/megabus. The binary file lac_io.b must linked with the usr/dvip/kernel to create a loadable bound unit.

## 2.8 TESTING CONSIDERATIONS

None

## 2.9 DOCUMENATION CONSIDERATIONS

The source code is written in "c" language and is self-explanatory.

## 2.10 OPERATING PROCEDURES

None

## 2.11 ERROR MESSAGES

None

# 3 INTERNAL SPECIFICATION

## 3.1 OVERVIEW

The basic function of the IO Dispatch is to honor the requests from users to register or obtain channel mailbox id for a given channel. An IOLD interrupt handler is implemented to assemble and dispatch IOLDS that are placed by the hardware in the pre-assigned queues. The module is linked and run under the control of the Bridge Kernel.

## 3.2 SUBCOMPONENT DESCRIPTION

### 3.2.1 IO HANDLER

This handler is hardware interrupt driven. Upon entry into the routine it scans the hardware queue for iold. The channel number associated with the iold will be used as an index to the channel mailbox table to obtain the mailbox id. Then the iold will be dispatched to that mailbox id. The routine resumes scan until all iolds are dispatched.

### 3.2.2 CHANNEL MAILBOX REGISTRATION

A channel mailbox table is setup to accommodate all 64 channels plus and event mailbox id. A user may send a registration message to request to put an entry for or read a particular channel or event id. When a user registers the event mailbox id with this component for the first time it will dispatch the start IO IOLD which is placed by the prom firmware in specific temporary buffer to the event mailbox id. The component is always ready to receive from its own mailbox for messages. A breceive is called.

### 3.2.3 IO QUEUE REQUEST INTERRUPT HANDLER

When the DMA chip runs out a queue it interrupts for another one. The routine responds the request by consulting the list which will specify which one of the four queues should be programmed into the chip . Before returning to the Kernel it must reset the block transfer complete bit in the channel status register.

## 3.3 FUTURE DEVELOPMENT AND MAINTENANCE

as required

## 4. PROCEDURAL DESIGN

none

# Software Component Specification

| | |
|---|---|
| SYSTEM: | MOD400 |
| SUBSYSTEM: | LAN Facility |
| COMPONENT: | Interface S/W - I/O Dispatcher |
| PLANNED RELEASE: | Release 4.0 |
| SPECIFICATION REVISION NUMBER: | 1 |
| DATE: | Aug. 16,1985 |
| AUTHOR: | K.Yu |

This specification describes the current definition of the subject software component, 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

| | |
|---|---|
| IO | Input/Output |
| IOLD | Input/Output Load |
| CPU | Central Processor Unit |
| LCB | LAN Control Block |

# 1. INTRODUCTION AND OVERVIEW

## 1.1 BACKGROUND

This is one of several Interface Software modules developed by the Hardware Development Group to provide an interface between the Communication Software and 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.

This interface supports a physical I/O interface between the Level 6 and Lan controller. All IOLDs that are destined for this controller are assembled and dispatched to various layers which require communication with Level 6. This module can be viewed as an extension of the Kernel Service function.

## 1.2 BASIC PURPOSE

The basic function of the module is to recognize the iolds that are issued by a CPU and dispatch them to its destination according to an entry in the channel table which is previously setup by the user.

## 1.3 BASIC STRUCTURE

This module essentially consists of three sections.The first section initializes the IOLD queues and hardware enviroment for accepting IOLDs from Level 6. The second section contains the main routine looking for messages from its own mailbox to perform mailbox registration and presentation of channel mailbox information that specifies where an IOLD may be dispatched. The third section is the IOLD interrupt handler. It assembles the incoming IOLDs from the IO queues into messages and dispatches them to their destination according to their channel mailbox table entry. It also manages the queues for the hardware to continue to accept IOLDs from the megabus.

## 1.4 BASIC OPERATION

The process starts with the initialization code. It programs the dma chip with two queue pointers and sets up the control registers ready for IOLDs. The queues are cleared to zero. The interrupt routine to handle the next IOLD queue is registered with the Kernel. Now it is ready to accept messages from the user to register their mailbox entries. When an IOLD arrives the channel associated with it will be used to pickup the entry. It is this entry that the IOLD will be dispatched to. During the registration no IOLD will be allowed until an event registration is completed.Then the start_io iold information will be sent to the event mailbox. Only this time, a signal is given to hardware to enable megabus IO reception. The operation is under the Kernel O.S. environment.

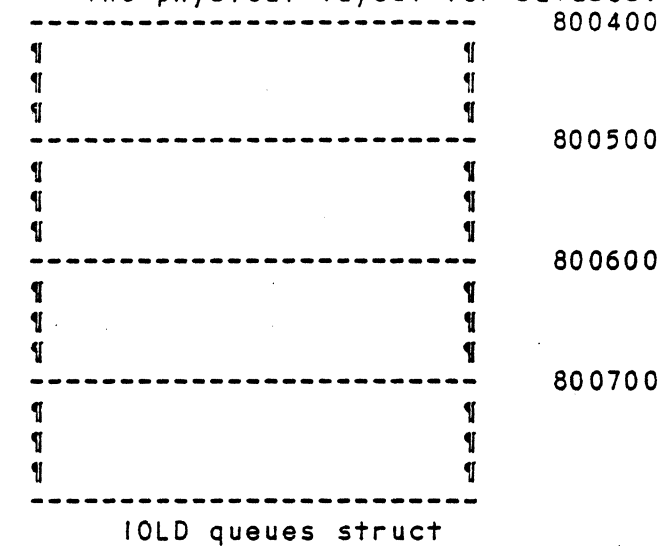
## 2. EXTERNAL SPECIFICATION

### 2.1 OWNED DATA STRUCTURES

A data structure is created to handle I/O orders. The "c" Declaration:

```
struct §    short ch_fc;        /* channel number and function code */
            long  *l6_address;/* Level 6 memory address */
            short range;        /* range */
            †IOITEM
struct          §
           IOITEM io_list0[32];      /* first io queue list */
           IOITEM io_list1[32];      /* second io queue list */
           IOITEM io_list2[32];      /* third io queue list */
           IOITEM io_list3[32];      /* fourth io queue list */
                †DataDes;
```

The physical layout for DataDes:




IOLD queues struct

A mailbox table for 64 channels and an Event is created during startup. This table contains all mailbox ids. When an iold comes the channel associated with this iold will be used to pickup the mailbox id from the table and dispatch to that mailbox. An zero entry will dispatch to the event mailbox. A read or write command to the table is permitted any time.

## 2.2 EXTERNAL INTERFACES

The IO Dispatcher is driven by two external events. A channel mailbox registration(write channel mailbox id) or read channel mailbox id is accomplished with the O.S. sendmsg call. The message type is register normal,register event, read normal,or read event. The "c" Declaration for the message:

```
struct chan_mbid §
        MSG   m          /* kernel message header          */
        short chan_nmb   /* channel number bit 10 - 15     */
        MBID  mbid       /* mailbox id for this channel    */
        MBID  ret_mbid   /* return mailbox id              */
        short ret_code   /* normal = 0,undefined type=-1*/
        t;
```

When an event mailbox is registered for the first time the start I/O iold being queued up by the prom firmware is sent to the event mailbox and hardware is enabled to accept any IO oders. When the cpu issues an IO order to lac controller the hardware put it into the queue and cause a level 4 68000 interrupt. The IO Dispatcher is activated and uses the following "c" declaration to dispatch the iold:

```
struct ioldmsg§
 MSG   m          /* kernel message header   */
 short chanfc     /* channel number and function code*/
                  /* bit 0 -9 channel# 10 - 15 = IOLD*/
 long  *16_addr   /* icb address in Level 6      */
 short icb_inf    /* protocal id and icb size    */
                   t;
```

## 2.3 INITIALIZATION REQUIREMENTS

The initialization and main entries are part of the system table,CS1 file. When the system is up this process is also ready. The mailbox id for this process is "IODISP". The mailbox table is created and is cleared to zero. IO queues are setup and cleared to zero. The dma chip is setup and is ready. However, Registration of the interrupt routine to handle IO with the O.S. must done when the system is powered up. No IO orders would be accepted until event registration is completed first.

## 2.4 TERMINATION REQURIEMENTS

There is no termination requirement for this IO Dispatcher as long as the Lacs board is up.

## 2.5 ENVIRONMENT

The code is in "c" control megabus interface and operate under the environment of the Bridge Kernel.

## 2.6 TIMING AND SIZE REQUIRMENTS

The size of this module is approxmately 2k words.

## 2.7 ASSEMBLY AND LINKING

The source code name for this is lac_io.c and is in the directory /usr/dvip/megabus. The binary file lac_io.b must linked with the usr/dvip/kernel to create a loadable bound unit.

## 2.8 TESTING CONSIDERATIONS

None

## 2.9 DOCUMENATION CONSIDERATIONS

The source code is written in "c" language and is self-explanatory.

## 2.10 OPERATING PROCEDURES

None

## 2.11 ERROR MESSAGES

None

# 3. INTERNAL SPECIFICATION

## 3.1 OVERVIEW

The basic function of the IO Dispatch is to honor the requests from users to register or obtain channel mailbox id for a given channel. An IOLD interrupt handler is implemented to assemble and dispatch IOLDS that are placed by the hardware in the pre-assigned queues. The module is linked and run under the control of the Bridge Kernel.

## 3.2 SUBCOMPONENT DESCRIPTION

### 3.2.1 IO HANDLER

This handler is hardware interrupt driven. Upon entry into the routine it scans the hardware queue for iold. The channel number associated with the iold will be used as an index to the channel mailbox table to obtain the mailbox id. Then the iold will be dispatched to that mailbox id. The routine resumes scan until all iolds are dispatched.

### 3.2.2 CHANNEL MAILBOX REGISTRATION

A channel mailbox table is setup to accommodate all 64 channels plus and event mailbox id. A user may send a registration message to request to put an entry for or read a particular channel or event id. When a user registers the event mailbox id with this component for the first time it will dispatch the start IO IOLD which is placed by the prom firmware in specific temporary buffer to the event mailbox id. The component is always ready to receive from its own mailbox for messages. A brecelve is called.

### 3.2.3 IO QUEUE REQUEST INTERRUPT HANDLER

When the DMA chip runs out a queue it interrupts for another one. The routine responds the request by consulting the list which will specify which one of the four queues should be programmed into the chip . Before returning to the Kernel it must reset the block transfer complete bit in the channel status register.

## 3.3 FUTURE DEVELOPMENT AND MAINTENANCE

as required

4. PROCEDURAL DESIGN
none