omninet

PROGRAMMER'S GUIDE

PRELIMINARY

CORVUS

CHAPTER 1

Onminet Local Network

Table of Contents

Chapter 1 -- Omninet Local Network

Introduction

Programming Overview

Transporter Commands

Transporter Return Codes

Chapter 2 -- Network Servers

Disk Server Installation and Programming Guide

Chapter 3 -- Transporter Card Installation and Programming Guide

Apple II Transporter Installation and Programming Guide

LSI-11 Transporter Installation and Programming Guide

INTRODUCTION

OMNINET is a local network which operates with many popular microcomputers, ranging from the Apple II to the LSI-11. It provides the microcomputer users with cost effective installation and growth capability. Network benifits include the ability to access a common data base, share use of printers and other peripherals, inter-computer communications, and multifunctional capabilities at each station. The OMNINET local network transfers data at 1 million bits per second over an RS-422 twisted pair wire cable up to 4000 feet in length.

Each device attached to the OMNINET local network has an interface controller called a transporter. The transporter utilizes a Motorola 6801 microprocessor, a custom gate array device to control high speed direct memory access (DMA) data transfers, and associated support components.

The transporter interfaces directly to both the RS-422 serial line and the host microcomputer's memory. In order to reduce the software burden placed on the host computer, the transporter performs many of the high level network tasks which are usually the responsibility of the host computer. The transporter automatically handles message acknowledgement, error detection and retransmission, and detection of duplicate messages. Basically, the transfer of data to or from the host microcomputer is performed without intervention by host software.

The OMNINET local network is a distributed control network; hence, no master controller is required. Network control is assumed by any transporter which has a message to send as soon as the network is available.

Since OMNINET is a shared access local network that allows any device to transmit at any time, a collision avoidance scheme is implemented in the transporters. Collision avoidance is performed using a combination of two methods. First, the popular Carrier-Sense Multiple-Access (CSMA) mechanism is utilized to determine when the network is available and second, the transporter computes a randomized transmit start time to minimize the probability of two devices trying to access an available network at the same time.

Unlike many other local networks, OMNINET does not require a collision detection mechanism. This allows OMNINET to be implemented on an RS-422 twisted pair wire and eliminates the cost associated with collision detection hardware.

Error detection and retransmission activities are performed by the transporters without software intervention. Thus, the host software burden is substantially reduced while overall system performance is improved. OMNINET utilizes a positive acknowledgment method to ensure that a message has reached its destination without error. If a positive acknowledgment is not received for a message, retransmission is automatically performed by the transporter until it is successful, or a user-specified number of retries have been performed. If transmission is unsuccessful, the sending host is informed as to the nature of the error.

Device addressing within the OMNINET local network allows a message to be sent to any device attached to the network or a message to be broadcast to all devices on the network. In addition to device addressing, OMNINET supports receive sockets. A receive socket provides additional addressing capability by allowing a message to be sent to a particular buffer within the selected host microcomputer. Four sockets can be defined for each device to aid in the separation of different types of messages within the host microcomputer.

The transporter accepts two major categories of commands from the host microcomputer to control the overall flow of messages within the local network. The two major categories are:

- Send Message
- Setup Receive

The send message command transmits a message up to 2047 bytes in length to any designated host socket. The send message command includes a command code, a result address, a destination host socket number, the message address and length, and optional user control information up to 255 bytes in length. The transporter utilizes DMA to transfer the message and optional user control information without additional host software involvement. When the message is delivered, or failed to be delivered after retries, the result status code is set to reflect the status of the send message command. Some of the possible results are:

- Message delivered successfully
- Message failed after 'n' retries
- Receiving socket not set up
- Message too long for receiving socket
- Invalid socket or host device number

Setup receive commands prepare host sockets to receive incoming messages. The setup receive command includes a command code, a result address, a socket number, a data buffer address and maximum message length. The optional user control information length is also specified.

When optional user control information is sent, the user control information can be placed into a different memory address than the message data. This feature allows device drivers to place the message data directly into the user's buffer while placing the driver control information into the driver's buffer, thus eliminating the need for the driver to move the message data to the user's buffer.

OMNINET is capable of supporting a variety of shared peripherals. Shared peripherals are attached to the network using a device called a server. The Corvus disk server can be used with OMNINET to supply 5 to 80 million bytes of shared Winchester disk storage. The disk server appears as just another device on the network to OMNINET, but performs disk sharing for the other microcomputers on the network. The disk server contains both an OMNINET network interface and a Corvus disk system interface. Additional shared peripherals will be added to the OMNINET local network in the future.

Various types of network protocols can be implemented using the OMNINET local network. Corvus Systems has implemented its field-proven CONSTELLATION software protocol on the OMNINET local network. The CONSTELLATION software allows multiple microcomputers to share the Corvus disk system.

PROGRAMMING OVERVIEW

The host computer communicates with the transporter by first ① formatting a command control block and then ② giving the address of the command control block to the transporter. The method of giving the command control block to the transporter varies with the type of microcomputer being used. Chapter Three , "Transporter card installation and programming guide", describes the transporter card installation procedure and how to send a command control block address to the transporter for the specific computer type. The command control block always contains the address of a result record and may contain additional command dependent information. The result record is used to indicate the status of the command.

The transporter transmits one message at a time, but may be activated to receive up to four messages using four different socket addresses. The socket defines the memory location and length of the receive buffer. Two of the sockets, $A0 and $B0, may also have an optional user control data buffer up to 255 bytes in length. The other two sockets, $80 and $90, are not allowed to receive user control data and hence do not have a user control data buffer. The user control data buffer is always located 4 bytes after the start of the result record.

All transporter addresses and lengths must be stored with the most significant byte first and the least significant byte last. Commands are initiated from the host by setting up the command control block and initializing the first byte of the result record, the return code, to hex $FF. Next the command control block address is given to the transporter in the form of 3 bytes with the most significant byte given first. After the address of the command has been given to the transporter, the command is processed and the result record is updated. The first byte of the result record, the return code, is set to a value of other than hex $FF to indicate the the command is completed, and to indicate the status of the command. On host microcomputers that support interrupts, an interrupt is generated after the return code is updated.

TRANSPORTER COMMANDS

The transporter supports the following commands:

- SEND MESSAGE
- SETUP RECEIVE
- END RECEIVE
- INITIALIZE TRANSPORTER
- WHO AM I
- ECHO
- PEEK/POKE

The command control block and result record for each command is shown below with a description of the command.

SEND MESSAGE

initialize to $FF

COMMAND CONTROL BLOCK

COMMAND = $40	
RESULT	msb
RECORD	
ADDRESS	lsb
DESTINATION SOCKET	
DATA	msb
ADDRESS	
	lsb
DATA	msb
LENGTH	lsb
CONTROL LENGTH	
DESTINATION HOST	

RESULT RECORD

RETURN CODE
(unused)
(unused)
(unused)
USER CONTROL DATA 0 to 255 bytes of user control info. to be transmittted with message.

The SEND MESSAGE command directs the transporter to send a message to the indicated DESTINATION HOST and DESTINATION SOCKET. The message consist of DATA LENGTH bytes of data from DATA ADDRESS and CONTROL LENGTH bytes of data from the USER CONTROL DATA area. Valid values for DATA LENGTH are between 0 and 2047. If the DESTINATION SOCKET is $80 or $90 then CONTROL LENGTH must be zero. If the DESTINATION SOCKET is $A0 or $B0, then CONTROL LENGTH can be any value between 0 and 255 however, it must exactly match the value setup by the receive socket.

For a broadcast command, DESTINATION HOST number is set to hex FF.

The message will be retransmitted, if necessary, until sucessful or until the retry count has been exceeded. The retry count has a default value of 10, but may be altered by the PEEK/POKE command. The user must not modify the message buffers or attempt to send any command to the transporter until the command has finished as indicated by the RETURN CODE.

no queue

The RETURN CODE will be one of the following values upon completion:

$00	-- Message sent successfully without retries
$nn	-- Message sent successfully after nn retries
$80	-- Message was not acknowledged (retry count exceeded)
$81	-- Message DATA LENGTH too long for recerver's buffer
$82	-- Message sent to uninitialized socket
$83	-- Message CONTROL LENGTH did not match receiver's CONTROL LENGTH
$84	-- Invalid DESTINATION SOCKET number in command control block
$85	-- Invalid DESTINATION HOST # in command control block

SETUP RECEIVE

COMMAND CONTROL BLOCK

COMMAND = $F0	
RESULT	msb
RECORD	
ADDRESS	lsb
SOCKET NUMBER	
DATA	msb
ADDRESS	
	lsb
DATA	msb
LENGTH	lsb
CONTROL LENGTH	

RESULT RECORD

RETURN CODE	
SOURCE HOST	
DATA	msb
LENGTH	lsb
USER CONTROL DATA 0 to 255 bytes of user control info. receive with the message.	

inconsistent with Send

A very restrictive as it accepts msg only from one place

if prespecified presumably not

poor description on broadcast

why not just one setup for all messages - let pacing be controlled by Send receive?

The SETUP RECEIVE command prepares the socket SOCKET NUMBER for the receiving of a single message. The data will be received at DATA ADDRESS provided the transmitted message is not longer than DATA LENGTH bytes. The DATA LENGTH field of the RESULT RECORD will contain the actual length of the data received from the SOURCE HOST. If the SOCKET NUMBER is $A0 or $B0, USER CONTROL DATA can also be received as part of the message. The sender must send exactly CONTROL LENGTH bytes of USER CONTROL DATA. If the SOCKET NUMBER is $80 or $90, then CONTROL LENGTH must be zero for both the sender and receiver.

Seems too strict. Why not
1- always accept DL, length (with too long indication
2- let Receive end determine control length.

The SETUP RECEIVE command has two replies; The first reply indicates that the command has been processed and the receive socket is ready to receive a message or an error has occured. After the receive socket is setup, a second reply occurs when a message is actually received for the setup socket.

For the first reply the RETURN CODE will be one of the following:

$84 -- Invalid SOCKET NUMBER in command control block
$85 -- Receive SOCKET NUMBER in use.
$FE -- Socket SOCKET NUMBER successfully setup to receive

If the first reply has a REASON CODE of $FE, a second reply is received when a message arrives for the setup socket. The RETURN CODE is set to $00 to indicate a message was received successfully. Before the RETURN CODE is updated, the following fields of the RESULT RECORD are updated. The SOURCE HOST is set to the transporter device address of the message originator. The DATA LENGTH field is set to the number of bytes received at DATA ADDRESS. The USER CONTROL DATA field will contain CONTROL LENGTH bytes of user control data as sent by the sender.

If a message is received in which the data portion is longer than the DATA LENGTH of the setup socket, or the user control data is not exactly the same length as specified by the CONTROL LENGTH field in the receiver's COMMAND CONTROL BLOCK, the message is rejected. The sending host is informed of the error but the receiving host is not. The receive socket remains setup as if no message was received.

should be more forgiving

END RECEIVE

COMMAND CONTROL BLOCK

COMMAND = $10	
RESULT	msb
RECORD	
ADDRESS	lsb
SOCKET NUMBER	

RESULT RECORD

RETURN CODE

The END RECEIVE command tells the transporter to release the specified SOCKET NUMBER. This disables reception of any further messages for the socket until another SETUP RECEIVE command is issued for the socket.

// see previous comments

One of the following RETURN CODES is returned upon completion:

$00 -- Operation complete
$84 -- Invalid SOCKET NUMBER

INITIALIZE TRANSPORTER

COMMAND CONTROL BLOCK

COMMAND = $20	
RESULT	msb
RECORD	
ADDRESS	lsb

RESULT RECORD

RETURN CODE

The INITIALIZE TRANSPORTER command initializes the transporter as in a hardware reset or a power-up. All parameters are set to their default values and event counters are reset to zero.

The RETURN CODE is set to the transporter's device address.

WHO AM I

COMMAND CONTROL BLOCK

COMMAND = $01	
RESULT	msb
RECORD	
ADDRESS	lsb

RESULT RECORD

RETURN CODE

The WHO AM I command causes the RETURN CODE to be set to the transporter's device address.

ECHO

COMMAND CONTROL BLOCK

COMMAND = $02	
RESULT	msb
RECORD	
ADDRESS	lsb
DESTINATION HOST	

RESULT RECORD

RETURN CODE

The ECHO command request the transporter to send an echo packet to the DESTINATION HOST. The ECHO command is a means by which one network device can verify the presence of another network device without disturbing that device. The transporter with the DESTINATION HOST number receives the packet and acknowledges without informing the attached host computer. It is not necessary for the DESTINATION HOST to have any sockets setup for the command to operate.

One of the following RETURN CODES is returned upon completion:

$80 -- Packet was not acknowledged
$86 -- Invalid DESTINATION HOST number
$C0 -- Destination transporter acknowledged successfully

PEEK/POKE

COMMAND CONTROL BLOCK

COMMAND = $08	
RESULT	msb
RECORD	
ADDRESS	lsb
6801	msb
ADDRESS	lsb
$00=PEEK, $FF=POKE	
POKE DATA BYTE	

RESULT RECORD

RETURN CODE

The PEEK/POKE command allows the host computer to examine or alter data within the transporters internal memory.

Upon completion of the command, the RETURN CODE is $00 for a POKE request and contains the value of the byte at the 6801 ADDRESS for a PEEK command.

The hex addresses within the 6801 internal memory that may be of interest to the user are:

is this memory, EAROM or RAM? If latter, how initialized? Instead ROM?

$00E1 -- Maximun # of retries on transmit message failure. The default value is $0A (10 decimal).

$00E2 -- Number of closing flags to send after each packet. The default value is 7. User supplied values must be greater than one for correct operation.

funct?

$00E3 -- Initial value to be used to scale the random delay in the retry logic. The default value is $07. User supplied values must be $01, $03, $07, $0F, $1F, $3F, $7F or $FF.

ANSPORTER RETURN CODES

The RETURN CODES were described within the context of the various commands. Below is a list of all possible command RETURN CODES for ease of reference:

$00 -- Command successfully completed
$00-$3F -- Transporter device address
$01-$7F -- Transmit retry count
$80 -- Transmit failure (no acknowledgement after max retries)
$81 -- DATA LENGTH too long for receive buffer
$82 -- Message sent to uninitialized socket
$83 -- Transmit CONTROL LENGTH not equal to receive CONTROL LENGTH
$84 -- Invalid socket number (must be $80, $90, $A0, or $B0)
$85 -- Receive socket in use
$86 -- Invalid transporter device address (must be $00-7F, or $FF)
$C0 -- Received an acknowledgement for an ECHO command
$FE -- Receive socket setup sucessfully

CHAPTER 2

Network Servers

DISK SERVER INSTALLATION AND PROGRAMMING GUIDE

INSTALLATION

The Corvus disk server contains a Dip switch with eight microswitches. Microswitches 1-6 are used to set the unique OMNINET device address. Normally, the disk server OMNINET device address is set to zero. The switch setting for each of the possible 64 device addresses are described in the OMNINET Installation Guide.

Microswitch number 7 is used to set a bias offset on the OMNINET trunk cable. It is recommended that the disk server be used as the network bias device. Therefore, switch number 7 should be on for the disk server and off for all other network devices. A network bias is used to reduce the effect of noise on the line when it is idle.

Microswitch number 8 is reserved for network termination. It should be off for all network devices because terminators are physically installed at both ends of the network as described in the OMNINET Installation Guide.

PROGRAMMING GUIDE

Chapter One of this OMNINET Programmer's Guide describes the commands that can be used by a computer with an OMNINET network interface card, called a transporter. These commands are used to send messages to, and receive messages from, the disk server. The disk server, in turn, interfaces with the Corvus Disk system which performs the actual disk operations.

Any computer on the OMNINET local network can communicate with the disk server by formatting a command control block and sending the message to the disk server. The Computer communicating with the disk server must first setup a receive socket to receive the response, and then send the message to the disk server. This insures that the receive socket is setup for the disk server's response. The disk server supports every disk command that the Corvus disk system supports. The content of the message sent to the disk server is basically a standard disk system command block.

For a complete list of Corvus disk commands and additional techinical information on the Corvus disk systems see the Corvus Disk Systems Technical Reference Manual. For a detailed description of OMNINET commands, control block formats, and return codes refer to the Chapter One. Refer to Chapter Three for the Transporter card installation and programming guide for the specific computer's transporter card.

Need one of these

Since the disk server has limited buffer space, disk commands that are longer than 4 bytes in length are performed in two transfers. The first transfer is used to send the first 4 bytes of the disk command to the disk server. The computer then waits until a "GO" response is received from the disk server. After the "GO" is received, the rest of the data for the disk command is sent to the disk server. This allows the disk server to control the use of its buffer space. The disk server is capable of queuing one request for each OMNINET network device and processing any disk command that the Corvus disk system supports.

The disk server uses socket address $B0 to receive all disk commands with a length of 4 bytes or less. For disk commands greater than 4 bytes in length, the socket address $B0 is used to receive the first 4 bytes of the disk command and the socket address $A0 is used to receive the second part of the disk command. The second part of the disk command is not sent until after a "GO" is received from the disk server. The disk server sends all of its responses to socket address $B0.

The disk server uses socket address $80 to receive broadcast messages. A send message command with a broadcast address ($FF) can be used to determine the transporter device address of the disk server.

BROADCAST COMMANDS

To broadcast a message to the disk server, the computer must perform two transporter commands as follows:

STEP 1

Use the setup receive command to setup receive socket $B0.

STEP 2

Use the send message command to broadcast a message to socket $80.

The command control block details for STEP 1, the setup receive command, are as follows:

COMMAND CODE	=	$F0
RESULT RECORD ADDRESS	=	(see below)
SOCKET NUMBER	=	$B0
DATA ADDRESS	=	Address of buffer for disk command response
DATA LENGTH	=	Length of buffer at DATA ADDRESS
CONTROL LENGTH	=	3

will contain (?)

The RESULT RECORD contains the following:

RETURN CODE	=	Transporter status code
SOURCE HOST #	=	Disk server transporter device address
DATA LENGTH	=	Length of data received at DATA ADDRESS
USER CONTROL DATA	=	Information from disk server (3 bytes)

Length of disk command response including status byte (2 bytes)
Disk command status(1 byte)

The command control block details for STEP 2, the broadcast message command, are as follows:

```
COMMAND CODE            = $40
RESULT RECORD ADDRESS   = Address for transporter RETURN CODE
DESTINATION SOCKET      = $80
DATA ADDRESS            = (see below)
DATA LENGTH             = Length of data to send at DATA ADDRESS
CONTROL LENGTH          = 0
DESTINATION HOST #      = $FF (broadcast)
```

The DATA contains the following:

```
(3 bytes)       Special code of hex 01FE01
(2 bytes)       Send length of disk command (4 bytes max.)
(2 bytes)       Receive length of disk command excluding status byte
(4 bytes max.)  Disk command (i.e., Read boot block)
```

When data is received in the receive socket $B0, the most significant bit of the first byte of the USER CONTROL DATA must be checked. If this bit is on, the disk has been reset and the operation should be restarted from STEP 1 above. The RESULT RECORD contains a field called SOURCE HOST #. This field is used to determine the disk server's transporter number.

SHORT DISK COMMANDS

To send any disk command to the disk server that is 4 bytes or less in length, the computer must perform the following two transporter commands:

STEP 1

Use the setup receive command to setup receive socket $B0.

STEP 2

Use the send message command to send a message to socket $B0.

The command control block details for STEP 1, the setup receive command, are as follows:

```
COMMAND CODE            = $F0
RESULT RECORD ADDRESS   = (see below)
SOCKET NUMBER           = $B0
DATA ADDRESS            = Address of buffer for disk command response
DATA LENGTH             = Length of buffer at DATA ADDRESS
CONTROL LENGTH          = 3
```

The RESULT RECORD contains the following:

```
RETURN CODE             = Transporter status code
SOURCE HOST #           = Disk server transporter device address
DATA LENGTH             = Length of data received at DATA ADDRESS
USER CONTROL DATA       = Information from disk server (3 bytes)
   Length of disk command response including status byte (2 bytes)
   Disk command status (1 byte)
```

The command control block details for STEP 2, the send message command, are as follows:

COMMAND CODE	= $40
RESULT RECORD ADDRESS	= (see below)
DESTINATION SOCKET	= $B0
DATA ADDRESS	= Address of disk command
DATA LENGTH	= Length of disk command (4 bytes max.)
CONTROL LENGTH	= 4
DESTINATION HOST #	= Disk server transporter number

The RESULT RECORD contains the following:

RETURN CODE	= Transporter status code
RESERVED	= (3 bytes unused)
USER CONTROL DATA	= Information for disk server (4 bytes)

Send length of disk command (2 bytes)
Receive length of disk command excluding status byte (2 bytes)

When data is received in receive socket $B0, the RESULT RECORD contains a field called SOURCE HOST #. This field should be checked to insure the request is from the disk server. Additionally, the most significant bit of the first byte of the USER CONTROL DATA must be checked. If this bit is on, the disk has been reset and the operation should be restarted from STEP 1 above.

LONG DISK COMMANDS

To send any disk command to the disk server that is 5 bytes or more in length, the computer must perform the following four transporter commands:

STEP 1

Use the setup receive command to setup receive socket $B0 for the "GO" response.

STEP 2

Use the send message command to send the first 4 bytes of the disk command to the disk server's socket address $B0.

STEP 3

Use the setup receive command to setup receive socket $B0 to receive the response from the disk.

STEP 4

Use the send message command to send the remaining bytes of the disk command, starting with the fifth byte, to the disk server's socket address $A0.

The command control block details for STEP 1, the setup receive command, are as follows:

```
COMMAND CODE           = $F0
RESULT RECORD ADDRESS = (see below)
SOCKET NUMBER          = $B0
DATA ADDRESS           = Address of buffer to receive two byte "GO".
DATA LENGTH            = Length of buffer at DATA ADDRESS (2 byte min
CONTROL LENGTH         = 0
```

The RESULT RECORD contains the following:

```
RETURN CODE            = Transporter status code
SOURCE HOST #          = Disk server transporter device address
DATA LENGTH            = Length of data received at DATA ADDRESS (2 byte
USER CONTROL DATA      = None
```

The command control block details for STEP 2, the send message command for the first 4 bytes of the disk command, are as follows:

```
COMMAND CODE           = $40
RESULT RECORD ADDRESS = (see below)
DESTINATION SOCKET     = $B0
DATA ADDRESS           = Address of disk command (first 4 bytes)
DATA LENGTH            = 4
CONTROL LENGTH         = 4
DESTINATION HOST #     = Disk server transporter number
```

The RESULT RECORD contains the following:

```
RETURN CODE            = Transporter status code
RESERVED               = (3 bytes unused)
USER CONTROL DATA      = Information for disk server (4 bytes)
   Send length of disk command including current 4 bytes (2 bytes)
   Receive length of disk command excluding status byte (2 bytes)
```

When data is received in receive socket $B0, the RESULT RECORD contains a field called SOURCE HOST #. This field should be checked to insure the request is from the disk server. Additionally, the most significant bit of the first byte of the DATA must be checked. If this bit is on, the disk has been reset and the operation should be restarted from STEP 1 above. The first two bytes of the DATA should contain the upper case ASCII characters "GO". After a valid "GO" is received steps three and four should be performed.

The command control block details for STEP 3, the second setup receive command, are as follows:

```
COMMAND CODE           = $F0
RESULT RECORD ADDRESS = (see below)
SOCKET NUMBER          = $B0
DATA ADDRESS           = Address of buffer for disk command response
DATA LENGTH            = Length of buffer at DATA ADDRESS
CONTROL LENGTH         = 3
```

The RESULT RECORD contains the following:

```
RETURN CODE            = Transporter status code
SOURCE HOST #          = Disk server transporter device address
```

DATA LENGTH	= Length of data received at DATA ADDRESS
USER CONTROL DATA	= Information from disk server (3 bytes) Length of disk command response including status byte (2 bytes) Disk command status byte (1 byte)

The command control block details for STEP 4, the send message command for the remaining bytes of the disk command, are as follows:

COMMAND CODE	= $40
RESULT RECORD ADDRESS	= Address for transporter status code only
DESTINATION SOCKET	= $A0
DATA ADDRESS	= Address of disk command (bytes 5-N)
DATA LENGTH	= Length of remaining bytes of disk command
CONTROL LENGTH	= 0
DESTINATION HOST #	= Disk server transporter number

When data is received in receive socket $B0, the RESULT RECORD contains a field called SOURCE HOST #. This field should be checked to insure the request is from the disk server. Additionally, the most significant bit of the first byte of the USER CONTROL DATA must be checked. If this bit is on, the disk has been reset and the operation should be restarted from STEP 1 above.

CHAPTER 3

Transporter Carc Installation and Programming Guiae

APPLE II TRANSPORTER INSTALLATION AND PROGRAMMING GUIDE

INSTALLATION

The Apple II OMNINET interface card, called a transporter, contains a Dip switch with eight microswitches. Microswitches 1-6 are used to set the unique OMNINET device address. The switch setting for each of the possible 64 device addresses are described in the OMNINET Installation Guide.

Microswitch number 7 is reserved as a network bias offset switch and is disabled on Apple II transporter cards. It is recommended that the disk server be used as the network bias device and hence switch number 7 should be on for the disk server and off for all transporter cards. In Apple networks without a disk server, a network junction can be used to set the network bias for the segment. A network bias is used to reduce the effect of noise on the line when it is idle.

Microswitch number 8 is reserved for network termination and is disabled on the Apple II transporter cards. Switch 8 should be off for all transporter cards and terminators must be physically installed at both ends of the network as described in the OMNINET Installation Guide.

The transporter card may be installed in any Apple II slot except slot 0. When used with Corvus CONSTELLATION software, the transporter card is normally installed in slot 6 of the Apple II.

PROGRAMMING GUIDE

Chapter One of this OMNINET Programmer's Guide describes the commands that can be used with the transporter. The Apple II communicates with the transporter by first formatting a command control block and then sending the command control block address to the transporter through the use of one control register. This control register is referred to as the Status and Command Address Register (SCAR). When the command is completed, a return code is placed in the result record address as specified in the command control block. For a detailed description of commands, control block formats, and return codes see Chapter One.

The Status and Command Address Register (SCAR) is an 8-bit register. The SCAR address is determined by the slot the transporter is installed in as shown in the chart that follows:

SLOT #	ADDRESS Hex	Decimal	Decimal
1	$C090	49296	-16240
2	$C0A0	49312	-16224
3	$C0B0	49328	-16208
4	$C0C0	49344	-16192
5	$C0D0	49360	-16176
→ 6	$C0E0	49376	-16160
7	$C0F0	49392	-16144

The sign bit of the SCAR is the Transporter ready bit (RDY). When set, this bit indicates the transporter is ready to receive the next address byte of the three byte command control block address. To issue a command to the transporter, the three byte address of the command control block must be given to the transporter one byte at a time. Every time an address byte is placed into the SCAR, the RDY bit of the SCAR goes low and the next byte cannot be sent until the RDY bit returns high again. The three byte address is sent with the most significant byte first. For the Apple II the first byte is always zero since the Apple II address space only requires two bytes.

SOFTWARE NOTES

While the transporter is receiving a packet from the network, it cannot process a byte moved into the SCAR, so the RDY bit of the SCAR remains low until the transporter can process the next byte. This leads to a situation where a software I/O driver may have to wait up to several milliseconds before the RDY goes high again.

A command return code is generated after the completion of each command issued to the transporter. The return code is placed into the address specified by the result record address field of the command control block. The program should initialize the command return code to a hex value of $FF before the command control block address is sent to the transporter. After the command control block address is sent to the transporter, the program can periodically check the return code for a value of other than hex $FF to determine when the operation has completed. Two return codes are generated for a valid setup receive command. The first return code indicates the command was accepted and the socket is setup to receive a message. The second return code occurs when a message is received.

Since the transporter processes one command at a time, the computer should not place any additional data into the SCAR after it has issued a command, until the command has completed as indicated by the command return code.

Seems unnecessarily primitive

LSI-11 TRANSPORTER INSTALLATION AND PROGRAMMING GUIDE

JUMPERS AND SWITCHES

The LSI-11 OMNINET interface board, called a transporter, contains jumpers to select the LSI-11 control and status register (CSR) address, the interrupt vector address, and interrupt priority. There is also a jumper to enable/disable the bootstrap.

The transporter contains a Dip switch with eight microswitches. Microswitches 1-6 are used to set the unique OMNINET device address. The switch setting for each of the possible 64 device addresses are described in the OMNINET Installation Guide.

Microswitch number 7 is used to set a bias offset on the OMNINET cable to reduce the effect of noise on the line when it is idle. Exactly one device on the network should have this switch set on.

Microswitch number 8 is reserved for network termination. Normally, switch 8 is off for all transporters because terminators are physically installed at both ends of the network as described in the OMNINET Installation Guide.

BOOTSTRAP

The transporter board has a 256 word bootstrap area with a starting address of 773000. The bootstrap sockets accept two 256 x 8 proms compatible with AMI 6309-1J or TI 74S471. Location U23 contains the low order bytes and location U16 contains the high order bytes of the bootstrap code. When shipped, the bootstrap is enabled and contains the boot code for a DEC RL01 disk drive or the Corvus RL01 compatible disk system. The bootstrap can be disabled by removing the jumper between pins J8 and J13.

DEVICE ADDRESS

The transporter hardware has support for a 20-bit address; However, an 18-bit address is normally used with Q-bus devices. The transporter contains jumpers to select bit 3 to bit 12 of the device CSR address. Pins used to set the CSR address are J1-J6 and J9-J12. Pin J7 is used as a ground. A jumper installed from an address pin to the ground pin results in a zero for that bit of the device address. Since there is a single ground pin, the jumpers are installed in a daisy-chained fashion. The CSR device address is preset to 766000 as shown in the chart that follows:

Bit	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	0
Pin	1	1	1	1	1	J1	J2	J3	J4	J5	J6	J9	J10	J11	J12	0	0	
766000	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	

Bit 17-13 are implied ones and bits 2-0 are implied zeroes. To create the preset device address of 766000, pins J1 and J4-J12 must be jumped to the ground pin J7. This can be performed with the following jumpers: J1-J4, J4-J5, J5-J6, J6-J7, J9-J10, J10-J11, J11-J12, and J12-J7.

INTERRUPT ADDRESS

The transporter contains jumpers to select bit 2 to bit 8 of the device interrupt vector address. Pins J14-J20 are used to set the interrupt vector address. Pin J21 is used as a ground pin. A jumper installed from an address pin to the ground pin results in a zero for that bit of the device interrupt vector address. Since there is a single ground pin, the jumpers are installed in a daisy-chained fashion. The device interrupt vector address is preset to 460 as shown in the chart that follows:

Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01
Pin	0	0	0	0	0	0	0	J14	J20	J19	J18	J17	J16	J15	0
460	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0

Bit 15-9 and bits 2-0 are implied zeroes. To create the preset interrupt address of 460 pins J20, J19, J16, and J15 must be jumped to the ground pin J21. This can be performed with the following jumpers:
J15-J16, J16-J19, J19-J20, and J20-J21.

DMA PRIORITY

The unbuffered version of the transporter has a maximum of 5.8 microseconds DMA latency before the transporter has overrun or underrun. The transporter must be the device with the highest DMA priority in the system.

INTERRUPT PRIORITY

The transporter contains jumpers to select the device interrupt priority. Pins J22-J25 are used to select the interrupt priority. To select the interrupt priority, one jumper is installed between the interrupt selected pin and pin J26 as shown in the table below:

Priority level	Jumper
4	J25-J26
5	J24-J26
6	J23-J26
7	J22-J26

The device interrupt priority is preset to level 4 to be compatible with the 11/2 and 11/23 CPU. This is performed by placing a jumper between pin J25 and pin J26.

PROGRAMMING GUIDE

Chapter One of this OMNINET Programmer's Guide describes the commands that can be used with the transporter. The LSI-11 communicates with the transporter by first formatting a command control block and then sending command control block address to the transporter through the use of two control registers. When the command is completed, a return code is placed in the result record address as specified in the command control block. An interrupt is generated when the operation is completed. For a detailed description of commands, control block formats, and return codes see Chapter One.

CSR - CONTROL AND STATUS REGISTER

The Control and Status Register (CSR) is a 16-bit register with a standard address of 766000. All bits can be read or written.

Bit 0-6 Not used

Bit 7 Interrupt Enable (IE)

This bit is set to 1 upon power up and hardware reset. If this bit is cleared, the transporter cannot interrupt the processor.

Bit 8-14 Not used

Bit 15 Transporter Ready (RDY)

When set, this bit indicates the transporter is ready to receive the next address byte of the three byte command control block address. This bit is cleared when a byte is moved into the Command Address Register (CAR).

CAR - COMMAND ADDRESS REGISTER

The Command Address Register (CAR) is a 16-bit write-only register with a standard address of 766002.

Bit 0-7 Command Address Byte

To issue a command to the transporter, the three byte address of the command control block must be given to the transporter one byte at a time. Every time an address byte is placed into the CAR, the RDY bit of the CSR goes low and the next byte cannot be sent until the RDY bit returns high again. The three byte address is sent with the most significant byte first.

Bit 8-15 Not used

SOFTWARE NOTES

While the transporter is receiving a packet from the network, it will not process a byte moved into the CAR so the RDY bit of the CSR remains low until the transporter can process the next byte. This leads to a situation where a software I/O driver may have to wait up to several milliseconds before the RDY goes high again.

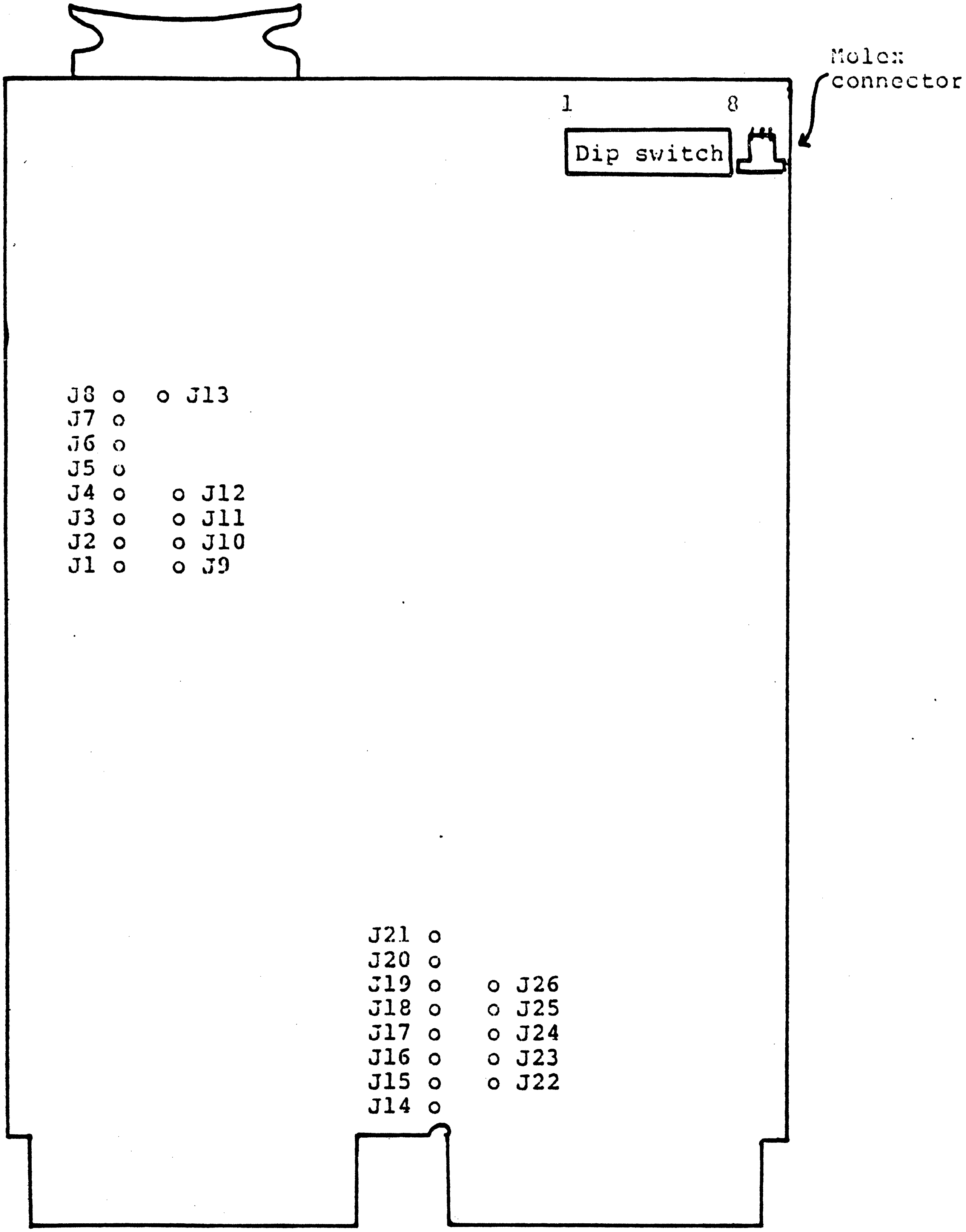
Since the transporter processes one command at a time, the computer should not place any additional data into the CAR after it has issued a command, until the command has completed as indicated by the command return code.

INTERRUPTS

An operation complete interrupt is generated after the completion of each command issued to the transporter. Before the interrupt is generated, a return code is placed in the address specified as the result record address in the command control block. Two interrupts are generated for a valid setup receive command. The first interrupt indicates the command was accepted and the socket is setup to receive a message. The second interrupt occurs when a message is received. The program should initialize the return code byte in the result record to hex $FF before the command code block is sent to the transporter. When a transporter interrupt occurs, the program must check the return code value of each active transporter command to determine which operation has just completed.

BYTE ORDER

All OMNINET addresses and lengths must be specified with the most significant byte first and the least significant byte last. Additionally, some addresses and lengths are not on word boundaries.



LSI-11 TRANSPORTER BOARD JUMPER LOCATIONS