INTRODUCTION TO INTERPROCESS MESSAGE FORMATS

Section C of this manual describes the formats of all of the messages used in interprocess communication. A message consists of a 6 word header and up to 106 words of data in the body of the message. Messages may be sent from a kernel process or a supervisor process to any other kernel or supervisor process. The message header is defined by the following structure:

```
struct msghdr {
        int     *mslink;        /* link to next message on input queue */
        int     msfrom;         /* process from which message was  received */
        int     msto;           /* process to which message is to be sent */
        char    mssize;         /* bits 0-2size  in multiples of 16 words
                                   bit 3:   allocated bit
                                   bit 4:   acknowledgment bit
                                   bit 5:   iolock bit
                                   bit 6:   capability bit
        char    mstype;         /* message type */
        int     msident;        /* message identification word only used by sender */
        char    msstat;         /* status byte set either by sender of message or by the kernel
                                   */
        char    msseqnum;       /* message sequence number */
}
```

If the message is a capability message, the message header is followed by a capability structure:

```
struct cp_clist {
        int     cpm_num;        /* capability number */
        int     cpm_owner;      /* capability owner */
        int     cpm_cap;        /* capability */
};
```

The sender need only fill in the capability number *cpm_num;* the kernel *sendcpmsg* EMT then fills in the capability owner and the value of the capability in the other two words of the structure before sending the message onto the *msto* process. It is the receiver's responsibility to validate the capability.

The data in the body of the message must be filled in directly by the sender of the message and varies with the message type as well as the receiving process. Normally the sender need only fill in the *msto* and *mstype* fields of the message header.

In the case of a kernel process "sender", an *alocmsg(nwords)* EMT call is required to alocate space for the message in the kernel message buffer pool. This call fills in the appropriate "size" and "allocate" bits in the *mssize* field of the message. The *msfrom* field must also be filled in by the "sender". The *msident* field may be filled in by the "sender" only if he wishes to recognize the acknowledgement to this message. The kernel process may then fill in the body of the message and send it to the intended receiver by means of the *queuem* or *queuemn* (no acknowledgement expected from receiving process) EMT calls. The kernel queues the message on the input queue of the receiving process using the *mslink* word. The kernel also fills in the *msseqnum* byte, which is used strictly as a debugging aid. The *msstat* field of the message is filled in by the receiving process to pass back error status. The value of −1 is reserved by the kernel to indicate that the receiving process does not exist or received an abnormal termination.

In the case of a supervisor process "sender", the *mssize* field must be filled in. Here the *mssize* field is now the number of words in the body of the message excluding the header. The complete message is formed in a message buffer in the supervisor address space. This message is sent to the intended receiver using the "sendmsg" EMT call. The kernel allocates space for the message in the kernel message buffer pool and copies in the message from the supervisor address space, converting the *mssize* word to the appropriate bit field in so doing. The *mslink* and *msseqnum* words are filled in by the kernel as in the case of a kernel process. In addition the kernel fills in the *msfrom* word in the message header.

Upon sending a message, the receiver is sent a message event to inform it of the presence of a message on its input queue. In the case of a kernel process receiver, no copy of the message occurs; a "dequeuem" or "dqtype" EMT call returns a pointer to the message buffer, allowing the receiver to directly access the body of the message. In the case of a supervisor process, a call to *getmsg* or *gettype* will initiate the copy of the message from the kernel message buffer pool to the supervisor buffer provided by the receiver, freeing up the space used by this message in the kernel message buffer pool in the process. The receiver need only fill in the maximum size message which he expects to receive.

This section of the manual details all message types for the processes sending and receiving messages. In each case the 6 word header is identical. Only the contents of the body of the message varies. The processes discussed include:

File Manager
Process Manager
Memory Manager
Scheduler
I/O Processes

Message types are discussed according to what types of messages each process is willing to accept.