Chapter 7

Asynchronous Message-Passing Auxiliary Functions

This chapter describes the auxiliary CMMD functions that can be used with the asynchronous message-passing functions. The key function here is `CMMD_free_mcb`, which is needed to free a message control block after a node has sent or received a message. In addition, there are functions for

- getting the self-address of a node and the size of its partition
- determining when there is a message waiting to be received
- getting information about a message from its MCB
- determining when an asynchronous message has been completed
- determining the availability of send and receive ports
- setting the internal message-matching function

7.1 Releasing Message Control Blocks

The `CMMD_free_mcb` function allows the specified message control block to be freed as soon as the message in question has been sent and received. There is no return value.

Note that this function actually marks the MCB as suitable for freeing; it does not actually free it. If the function is invoked before the message is completed, the MCB is not freed until the message has completed. If the function is invoked after the message has completed, the MCB is freed immediately.

Copyright © 1995 Thinking Machines Corporation

The reference page for this function is supplied in Section 7.9.

7.2 Node Information Functions

Processors must address each other explicitly during message passing. They do so through node identifiers, the logical addresses of the nodes. Asynchronous operators can use the same node information functions described in Section 5.1 to get the address of a node or the size of a partition. To summarize:

- `CMMD_self_address` returns the address of the node that calls it.
- `CMMD_partition_size` returns the number of nodes in the partition.
- `CMMD_host_node` returns the address of the host node in host/node programs (see Section 19.2).

7.3 Pending Message Functions

The function `CMMD_mcb_pending` tests whether there is a message waiting to be received. If it detects an available message, it creates and returns an MCB describing the message. Any node can call `CMMD_mcb_pending` to test for the availability of a message, and then issue a `receive_block` or `receive_async` call when it detects that another node is waiting to send a message. Thus, the node can do some useful task, check for pending messages, do some other task, check again, and so on.

The reference page for this function is supplied in Section 7.9.

7.4 MCB Accessor Functions

These functions are used to extract information from the message control blocks returned by the asynchronous functions `CMMD_send_async`, `CMMD_receive_async`, and `CMMD_mcb_pending`.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

These routines return the specified information about the message associated with the given message control block:

`CMMD_mcb_node`	Returns the identifier for the partner node; it can be used by either the sending or receiving node.
`CMMD_mcb_source`	Returns the identifier for the source (sending) node; this routine should be called only by receiving nodes.
`CMMD_mcb_dest`	Returns the identifier for the destination (receiving) node; this routine should be called only by sending nodes.
`CMMD_mcb_tag`	Returns the user tag of the specified message.
`CMMD_mcb_bytes`	Returns the number of bytes in the specified message.
`CMMD_mcb_buffer`	Returns the memory location of the beginning of the array to be sent or received.

Reference pages for these functions are supplied in Section 7.9.

7.5 Message Completion Functions

CMMD supplies four functions for detecting the completion of an asynchronous message. These tests can be used to sequence other operations (for example, when a node detects that a given message out of a series has been received, it can send the next one).

The functions are

```
CMMD_msg_done
CMMD_all_msgs_done
CMMD_all_sends_done
CMMD_all_receives_done
```

Their reference pages are supplied in Section 7.9.

Copyright © 1995 Thinking Machines Corporation

7.6 Waiting for Asynchronous Messages

CMMD supplies a function that can be used to make a node wait until a given asynchronous message has been transmitted, and a function that can make a node wait until all messages have been transmitted:

```
CMMD_msg_wait
CMMD_all_msgs_wait
```

Their reference pages are supplied in Section 7.9.

7.7 Determining Resource Availability

CMMD uses internal resources called send ports and receive ports in transmitting messages (see Chapter 6). The resources are limited. (At Version 3.3 of CMMD, each node has approximately 127 of each.) A node with many sends or receives pending can run out of ports.

Two functions, `CMMD_available_sends` and `CMMD_available_receives`, return the number of send ports and receive ports available. They are discussed in Section 5.4.

7.8 Message-Matching Function

CMMD uses an internal *matching function* to match up sending and receiving message-passing calls. You have the option of installing your own custom matching function; for more information, see Chapter 18.

7.9 Reference Pages

This section provides reference pages for asynchronous auxiliary functions.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMMD_free_mcb

Frees the specified MCB for re-use.

SYNOPSIS

C Syntax

```
void CMMD_free_mcb(CMMD_mcb mcb)
```

Fortran Syntax

```
SUBROUTINE CMMD_free_mcb(mcb)
    INTEGER (mcb)
```

ARGUMENT

mcb	The message control block ID returned by a call to `CMMD_send_async` or `CMMD_receive_async`.

DESCRIPTION

The `CMMD_free_mcb` function allows the specified message control block to be freed as soon as the message in question has been sent and received. There is no return value.

Please note: All messages inolve two MCBs: one on the sending node and one on the receiving node.

With most types of message-passing calls, the MCBs are never seen by the user. Only the two asynchronous functions, `CMMD_send_async` and `CMMD_receive_async`, allow the user program on each node to access and query the MCB associated with that call.

The MCB, and thus the corresponding send port or receive port, must be freed explicitly by the user. Until the user program calls `CMMD_free_mcb`, the MCB will not be freed. After the user program calls `CMMD_free_mcb`, it can no longer query the MCB.

Copyright © 1995 Thinking Machines Corporation

Two events, therefore, must occur before an MCB will be freed:

- The relevant message must be sent and received. (Yes, we do mean that both actions must happen before either MCB can be freed.)
- The node must mark the MCB for freeing. This cannot happen until the user program calls `CMMD_free_mcb`.

MCBs are a limited resource. If they are not freed after use, the program may run out of them, which produces an error condition.

Note that the call to `CMMD_free_mcb` merely provides permission for the node to free the MCB when the node itself has no further need of it. CMMD will not actually free the MCB until the message it tracks has completed. Your program, therefore, can issue the `free_mcb` call as soon as it likes. If you do not intend to query the MCB, the `free_mcb` call can even be made part of the send or receive call, as in the following example:

```
CMMD_free_mcb(CMMD_send_async(3, CMMD_DEFAULT_TAG, &foo, 512));
```

Callback and Handler Functions. This function can be called from callback functions and handler functions.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMMD_mcb_pending

Returns the MCB of the pending message, or NULL if no message is pending.

SYNOPSIS

C Syntax

```
CMMD_mcb CMMD_mcb_pending(int node, int tag)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_mcb_pending(node, tag)
    INTEGER node, tag
```

ARGUMENTS

node	Integer identifying a node (must be between 0 and partition-size minus 1, may be CMMD_ANY_NODE).
tag	Integer identifying a tag (must be nonnegative, may be CMMD_ANY_TAG).

RETURNS

CMMD_mcb_pending returns the MCB associated with a pending message. It returns NULL if no message is pending.

DESCRIPTION

This function is used to determine whether any messages are available for receipt by the calling node.

CMMD_mcb_pending returns the MCB associated with any pending message. If there are multiple messages pending, it returns the MCB for the first message whose node and tag fields match those in the call. It returns NULL if no message is pending.

Copyright © 1995 Thinking Machines Corporation

If `CMMD_ANY_NODE` was specified, the function `CMMD_mcb_node` can then be called to get the node identifier of the pending sender; if `CMMD_ANY_TAG` was specified, `CMMD_mcb_tag` returns the tag of the pending message.

Callback and Handler Functions. This function may be called from a callback function or handler function.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMMD_mcb_node
CMMD_mcb_source
CMMD_mcb_dest

These three functions identify the partner node in an asynchronous message transmission.

SYNOPSIS

C Syntax

```
int CMMD_mcb_node(CMMD_mcb mcb)
int CMMD_mcb_source(CMMD_mcb mcb)
int CMMD_mcb_dest(CMMD_mcb mcb)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_mcb_node(mcb)
INTEGER FUNCTION CMMD_mcb_source(mcb)
INTEGER FUNCTION CMMD_mcb_dest(mcb)
    INTEGER   mcb
```

ARGUMENT

mcb	A message control block identifier.

RETURNS

These functions return the identifier for the partner node in an asynchronous message transmission.

DESCRIPTION

These three functions return the identifier for the partner node in an asynchronous message transmission. `CMMD_mcb_node` returns the identifier for the sending node when called by the receiving node; it returns the identifier for the receiving node when called by the sending node.

Copyright © 1995 Thinking Machines Corporation

`CMMD_mcb_source` can only be called by the receiving node; it returns the identifier of the sending node. `CMMD_mcb_dest` can only be called by the source node; it returns the identifier for the receiving node.

Callback and Handler Functions. This function may be called from callback functions or handler functions.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMMD_mcb_tag
CMMD_mcb_bytes
CMMD_mcb_buffer

These functions return information about the specified message.

SYNOPSIS

C Syntax

```
int CMMD_mcb_tag(CMMD_mcb mcb)
int CMMD_mcb_bytes(CMMD_mcb mcb)
void *CMMD_mcb_buffer(CMMD_mcb mcb)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_mcb_tag(mcb)
INTEGER FUNCTION CMMD_mcb_bytes(mcb)
<ARRAY> FUNCTION CMMD_mcb_buffer(mcb)
    INTEGER mcb
```

ARGUMENT

mcb	A message control block identifier.

RETURNS

`CMMD_mcb_tag` returns the user tag of the specified message. `CMMD_mcb_bytes` returns the number of bytes in the specified message. `CMMD_mcb_buffer` returns the memory location of the beginning of the array to be sent or received. (The address for a parallel array is in ALL-DP instruction space.)

Callback and Handler Functions. These functions may be called from callback functions or handler functions.

Copyright © 1995 Thinking Machines Corporation

CMMD_msg_done

Returns 1 if specified message is completed, 0 otherwise.

SYNOPSIS

C Syntax

```
int CMMD_msg_done(CMMD_mcb mcb)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_msg_done(mcb)
    INTEGER mcb
```

ARGUMENT

mcb	A message control block identifier.

RETURNS

`CMMD_msg_done` returns 1 if the calling node has finished its work regarding the message specified by the MCB. For a sending node, this means that the message has been sent; for a receiving node, it means that the message has been received. The function returns 0 otherwise.

DESCRIPTION

This routine (and the three related routines `CMMD_all_msg_done`, `CMMD_all_sends_done`, and `CMMD_all_receives_done`, whose reference pages follow) are used to determine whether any messages are trying to be sent or received by the calling node. They may be used to check the status of the node's own outgoing messages, or to synchronize some action with the completion of message-passing traffic.

A message is considered "done" when the calling node has finished its involvement with the transmission for that message. A sending node can safely re-use the source buffer of a "done" message; a receiving node can safely read the destination buffer of a "done" message.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

The information supplied by these functions can be extremely important for the proper functioning of a program. For example, every program that uses asynchronous message-passing functions should call either `CMMD_all_msgs_done` or `CMMD_all_msgs_wait` on all nodes before it returns or exits, to make sure that no unfinished messages are still pending.

Callback and Handler Functions. These functions may be called from callback functions and from handler functions.

Copyright © 1995 Thinking Machines Corporation

CMMD_all_msgs_done

Returns 1 if this node's messages are all complete, 0 otherwise.

SYNOPSIS

C Syntax

```
int CMMD_all_msgs_done(void)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_all_msgs_done()
```

RETURNS

Returns 1 if all messages (sends and/or receives) by the calling node have been completed; 0 otherwise.

SEE ALSO

CMMD_msg_done

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMMD_all_sends_done
CMMD_all_receives_done

Returns 1 if this node's sends (or receives) are all complete, 0 otherwise.

SYNOPSIS

C Syntax

```
int CMMD_all_sends_done(void)
int CMMD_all_receives_done(void)
```

Fortran Syntax

```
INTEGER FUNCTION CMMD_all_sends_done()
INTEGER FUNCTION CMMD_all_receives_done()
```

RETURNS

Returns 1 if all sends (or receives) by the calling node have been completed; returns 0 otherwise.

SEE ALSO

CMMD_msg_done

Copyright © 1995 Thinking Machines Corporation

CMMD_msg_wait
CMMD_all_msgs_wait

These two functions cause a node to wait for the completion of one or more asynchronous messages.

SYNOPSIS

C Syntax

void **CMMD_msg_wait**(CMMD_mcb *mcb*)
void **CMMD_all_msgs_wait**(void)

Fortran Syntax

```
SUBROUTINE CMMD_msg_wait(mcb)
    INTEGER mcb

SUBROUTINE CMMD_all_msgs_wait()
```

ARGUMENT

mcb	A message control block identifier.

DESCRIPTION

CMMD_msg_wait and CMMD_all_msgs_wait are used for synchronizing a node with the completion of one or more asynchronous messages. CMMD_msg_wait forces the calling node to block, or wait, until a particular message has been transmitted. CMMD_all_msgs_wait causes the node to block until all its pending messages have completed.

Callback and Handler Functions. These functions may be called from callback functions or handler functions.

Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation