Chapter 6

Asynchronous Message-Passing Auxiliary Functions

This chapter describes the auxiliary CMMD functions that can be used with the asynchronous message-passing functions. These include 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
- freeing the MCB when the message has been transmitted
- setting the internal message-matching function

6.1 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 4.1 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 size of the partition.
- **`CMMD_host_node()`** returns the address of the host node in host/node programs (see Section 16.2).

6.2 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 a message is waiting to be sent. Thus, the node can do some useful task, check for pending messages, do some other task, check again, and so on.

FUNCTION **CMMD_mcb_pending**

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. (May be **CMMD_ANY_NODE**.)
tag	Integer identifying a tag. (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_msg_pending** returns the MCB associated with any pending message. It returns **NULL** if no message is pending.

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.

Handlers. This function may be called from a handler function.

6.3 MCB Accessor Functions

These functions are used to extract information from the message control blocks (MCBs) returned by the asynchronous functions **CMMD_send_async** and **CMMD_receive_async**.

FUNCTIONS

CMMD_mcb_node
CMMD_mcb_source
CMMD_mcb_dest
CMMD_mcb_tag
CMMD_mcb_bytes
CMMD_mcb_buffer

C SYNTAX

int **CMMD_mcb_node(**CMMD_mcb *mcb***)**
int **CMMD_mcb_source(**CMMD_mcb *mcb***)**
int **CMMD_mcb_dest(**CMMD_mcb *mcb***)**
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_node(***mcb***)**
INTEGER FUNCTION **CMMD_mcb_source(***mcb***)**
INTEGER FUNCTION **CMMD_mcb_dest(***mcb***)**
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

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 buffer used for sending (or receiving) the message.

HANDLERS

These functions may be called from handler functions.

6.4 Releasing Message Control Blocks

FUNCTION **CMMD_free_mcb**

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.

Each node is responsible for releasing its own MCBs after the messages they control have been delivered and the node has no further need to query the MCB for status information. If MCBs are not freed after use, the program may run out of them; this produces an error condition. Note that CMMD will not prematurely free an MCB. Your program can, therefore, issue the **free_mcb** call as soon as it likes; the 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));
```

Handlers. **CMMD_free_mcb** can be called from handler functions.

6.5 Message Completion Functions

The CMMD functions described here are used to detect 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).

FUNCTIONS

CMMD_msg_done
CMMD_all_msgs_done
CMMD_all_sends_done
CMMD_all_receives_done

C SYNTAX

int **CMMD_msg_done(**CMMD_mcb *mcb***)**
int **CMMD_all_msgs_done(**void**)**
int **CMMD_all_sends_done(**void)
int **CMMD_all_receives_done(**void**)**

FORTRAN SYNTAX

INTEGER FUNCTION **CMMD_msg_done(***mcb***)**
integer *mcb*
INTEGER FUNCTION **CMMD_all_msgs_done()**
INTEGER FUNCTION **CMMD_all_sends_done()**
INTEGER FUNCTION **CMMD_all_receives_done()**

ARGUMENT

mcb A message control block ID.

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.

CMMD_all_msgs_done, **CMMD_all_sends_done**, and **CMMD_all_receives_done** return 1 if all sends and/or receives by the calling node have been completed; they return 0 otherwise.

DESCRIPTION

These routines 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.

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** (explained later in this chapter) on all nodes before it returns or exits, to make sure that no unfinished messages are still pending.

Handlers. These functions may be called from handler functions.

6.6 Waiting for Asynchronous Messages

The functions described here can be used to make a node wait until a given asynchronous message has been transmitted, or until all messages have been transmitted.

FUNCTIONS **CMMD_msg_wait**
CMMD_all_msgs_wait

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.

Handlers. These functions may be called from handler functions.

6.7 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, to allow your program to perform useful kinds of messaging operations. For more information, see Section 15.1.