Chapter 9

Synchronization

When do nodes synchronize in a CMMD program?

If all goes well, they synchronize whenever they need to, but not much more often than that.

This is not flippancy: rather, it is a reminder that

- In a message-passing program, the programmer is responsible for ensuring that synchronization does happen when needed.
- Failure to synchronize when needed can cause hard-to-track errors in your program.
- Adding extra synchronization to early versions of a program under development can greatly ease the debugging effort.

Synchronization can occur automatically, as a side effect of some operation that demands synchrony (such as a global operation); or, it can be caused by an explicit call to a synchronization function. This chapter briefly reviews those operators that cause synchronization and offers full descriptions of three functions used for explicit synchronization:

CMMD_sync_with_node
CMMD_set_global_or
CMMD_get_global_or

Note: If interrupts are enabled on a node that is waiting for a synchronization (such as a **sync_with_nodes** or a global operation), that node can be interrupted to receive a message. If interrupts are not enabled, all pending messages must wait until after the synchronization occurs.

9.1 Automatic Synchronization

Two or more nodes synchronize automatically when they participate in one of the blocking message-passing functions described in Chapter 3.

All nodes synchronize automatically when

- Your application begins or ends.
- Your application performs one of the global operations described in Chapter 10 (broadcasts, scans, reductions, concatenation).
- Your application performs global I/O (that is, it opens or closes a global file, or performs any I/O on files that are in synchronous-broadcast or synchronous-sequential mode), as described in Chapter 12.
- Your application performs certain system-level functions, such as **`CMMD_reset_partition_size`** (see Section 15.1) and **`CMMD_global_suspend_servers`** (see Section 16.8).

9.2 Explicit Synchronization

In addition to the automatic synchronization mentioned above, you can explicitly synchronize nodes in the following ways:

- At any reasonable point in your application, you can have the nodes call the global function **`CMMD_sync_with_nodes`**. (This function is described below, in Section 9.3. Host/node programs use a variant of this call to include the host in the synchronization; see Section 16.4 for details.)
- To synchronize the nodes at the completion of some sequence of computation or communications, you can use CMMD's global OR facility. Each node contributes a value asynchronously to this logical OR; any node can then read the value of the OR as a test for synchronization purposes. Section 9.3, below, describes the global OR facility and the two functions that access it.

  This type of synchronization is sometimes called a "soft" synchronization, since it allows the nodes to do useful things while "waiting" for the synchronization point.

- In order to force nodes to synchronize at the completion of an asynchronous communications pattern — that is, to synchronize on the emptying of the Data Network after some series of asynchronous send and receive calls — each node can use the functions **`CMMD_all_msgs_done`** (described in Section 6.5) or **`CMMD_all_msgs_wait`** (described in Section 6.6).

  To synchronize at the completion of a communications pattern that uses channels, the equivalent functions are **`CMMD_all_channels_free`** and **`CMMD_all_channels_free_wait`**. (See Section 7.3.7 for details.)

  Note that these routines do not, themselves, synchronize the nodes. Rather, they are prerequisites to synchronization, either via an explicit synchronization call or via some other action that forces synchronization. (For example, a node should never consider its share of a program finished until it knows it has received all messages intended for it.)

9.3 Global Synchronization Functions

Global synchronization functions, as their name implies, serve to synchronize all nodes at a given point in a program.

CMMD offers several ways for nodes to synchronize. The first is an explicit synchronization function, to be called by all nodes at any point in a program.

`CMMD_sync_with_nodes`	Allows all nodes to synchronize themselves.

This function must be called by all nodes in the partition. Once any node has begun this synchronization call, the CM hardware keeps track of responses, and allows none of the calls to return until all nodes have made their call.

In addition, two asysnchronous global logical OR routines allow nodes to signal to each other by contributing to a global logical OR and reading its results.

`CMMD_set_global_or`	Sent by any node, this call adds a value (0 or non-zero) towards the creation of a global OR.
`CMMD_get_global_or`	Invoked by any node, this call reads the current value of the global OR.

By using the **`CMMD_set_global_or`** function, each processor contributes to the logical OR at an appropriate time; the hardware checks and updates the global value at frequent intervals; and individual processors read the value when desired. Thus, the global OR mechanism can be used as a non-blocking method of determining when all processors have reached a given state. All processors would start a task, for instance, by sending a 1. As each finished its share of the task, it would send a 0. By checking the value of the global OR (which would change to 0 only when all processors had finished), a processor could determine whether the whole task was complete and thus select its own next action. Note that this does require synchronizing at the beginning of the task to ensure that all nodes contribute and see a 1 before any node finishes its task and sends a 0.

Note: These asynchronous global OR functions should not be confused with the synchronous global-OR reduction operation, which is described in Section 8.2.

9.3.1 Explicit Synchronization via Function Call

FUNCTION **CMMD_sync_with_nodes**

C SYNTAX

void **CMMD_sync_with_nodes(**void**)**

FORTRAN SYNTAX

SUBROUTINE **CMMD_sync_with_nodes()**

RETURNS

This function does not return until all nodes in the partition have called the function. It has no return value.

DESCRIPTION

CMMD_sync_with_nodes synchronizes the calling node with all other nodes. Once one node has made this call, all nodes must; the function does not return until they all do.

Handlers. These functions cannot call or be called from handler functions.

9.3.2 Asynchronous Global OR

FUNCTIONS **CMMD_set_global_or**
CMMD_get_global_or

C SYNTAX

int **CMMD_set_global_or(**int *value***)**
int **CMMD_get_global_or()**

FORTRAN SYNTAX

INTEGER FUNCTION **CMMD_set_global_or(***value***)**
integer *value*

INTEGER FUNCTION **CMMD_get_global_or()**

ARGUMENT

value A non-negative integer.

RETURN VALUES

Both these functions return the current value of the global OR.

NOTE

If **CMMD_set_global_or** has not already set a value for the global OR, calls to **CMMD_get_global_or** return unpredictable results.

DESCRIPTION

CMMD_set_global_or allows any node to contribute a value (either 0 or some non-zero integer) to a global logical OR. The new value replaces any previous value sent by the node. The function returns when the value has been sent; it does not wait for participation by any other processor. It returns the current value of the global OR (either 1 or 0), including the contribution just made by the calling node.

CMMD_get_global_or returns the current value of the global OR function. If **CMMD_set_global_or** has not already set a value for the global OR, calls to **CMMD_get_global_or** return unpredictable results.

These functions are asynchronous; they require participation by no other processors.

As contributions to this global OR may be asynchronous, the hardware checks the value at frequent intervals and updates it as needed. Note, however, that some network delay exists during reception and propagation of values; thus, there is a small but actual window between the time at which a processor sends a **set_global_or** message and the time by which that message can affect the result of another node's **get_global_or** request. A global synchronization propagates the global OR value.

Handlers. These functions can be called from handler functions.