## Chapter 17

# CMMD Timers

The CMMD library provides a set of timing routines for use by the nodes in message-passing programs. (Timers for the host are not provided.)*

Each node manages its own timers. The timers are called by a single node and record the time spent by that node on whatever block of code the programmer wishes timed. They have no connection with timers used on any other node.

Like other CM timers, these timers measure three values:

- Busy time is the time during which the user program is executing user code.
- Idle time is the time during which the user program is looping in the operating system's dispatch loop. A CMMD program typically has very little (if any) idle time.
- Elapsed time is the sum of busy time and idle time; that is, it is the total time during which the program was scheduled for execution on the CM. (It does not include time during which the program was swapped out.)

Each node may have up to 64 timers running simultaneously. The timers are identified by the integers 0 through 63.

Timers can be nested. Thus, one timer can provide timings for an entire program, while a second timer provides timings for a particular routine within the program, and a third timer provides timings for a particular code block

* CM Fortran and C* programs may use either CMMD timers or CMOST timers, but not both in the same program. For nodal programs, the two sets of timers behave alike.


*Copyright © 1995 Thinking Machines Corporation*

— such as an if-then-else block or a case statement — within the routine. Each timer can record timings of up to 43 hours.

## 17.1 Using the CMMD Timers

All timers share the same general pattern of use:

- First, you must call `CMMD_node_timer_clear` with an integer *timer ID*. This creates a timer and initializes it to zero.
- Then you call `CMMD_node_timer_start`, to start the timer.
- The next timer call should be `CMMD_node_timer_stop`. This records or "freezes" the current values within the timer. The values remain unchanged, available for reading, until the next time `CMMD_node_timer_start` is called.
- The call to `CMMD_node_timer_stop` can thus be followed by calls to information-reading functions and by a call to `CMMD_node_timer_start`; these calls can be issued whenever you like. Timings for this particular timer will accumulate until you call `CMMD_node_timer_clear` again.

## 17.2 Node Timing Functions

The node timing functions may be called by any node to manipulate timers for that node. No node can directly manipulate another node's timers.

Nodes can, however, use timers cooperatively; for example, by performing global operations on the results read from local timers. Cooperative use of local timers can thus achieve global timing across the nodes.

The remainder of this section provides reference pages for the node timing functions.


Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

# CMMD_node_timer_clear

Clears (zeros out) the values for the specified timer.

---

### SYNOPSIS

#### C Syntax

```
int CMMD_node_timer_clear(int timer)
```

#### Fortran Syntax

```
INTEGER FUNCTION  CMMD_node_timer_clear(timer)
    INTEGER timer
```

---

### ARGUMENT

| | |
|---|---|
| *timer* | A *timer ID*, an integer from 0 to 63, identifying the timer to be manipulated. |

### RETURNS

The function returns 0 for success, −1 for an out-of-range timer ID.

### DESCRIPTION

This function sets the total elapsed time, total CM busy time, and number of starts for the specified timer to zero. It must be the first function called for any timer.

The function returns −1 if the timer ID argument is out of the range 0−63; otherwise, it returns 0.

---


*Copyright © 1995 Thinking Machines Corporation*

# CMMD_node_timer_start

Starts the clock running for the specified node timer.

---

### SYNOPSIS

#### C Syntax

```
int CMMD_node_timer_start(int timer)
```

#### Fortran Syntax

```
INTEGER FUNCTION CMMD_node_timer_start(timer)
    INTEGER timer
```

---

### ARGUMENT

| | |
|---|---|
| *timer* | A *timer ID*, an integer from 0 to 63, identifying the timer to be manipulated. |

### RETURNS

The function returns 0 for success, −1 for an out-of-range timer ID.

### DESCRIPTION

This function starts the clock running for the specified timer. Elapsed time and CM busy time are accumulated. Number of starts is increased by one. (If you wish to zero out these values before restarting a timer, you must first call CMMD_node_timer_clear.)

---


Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

# CMMD_node_timer_stop

Stops the clock for the specified timer, allowing values to be updated and read.

---

### SYNOPSIS

#### C Syntax

```
int CMMD_node_timer_stop(int timer)
```

#### Fortran Syntax

```
INTEGER FUNCTION CMMD_node_timer_stop(timer)
    INTEGER timer
```

---

### ARGUMENT

| | |
|---|---|
| *timer* | A *timer ID,* an integer from 0 to 63, identifying the timer to be manipulated. |

### RETURNS

The function returns 0 for success, −1 for an out-of-range timer ID.

### DESCRIPTION

This function stops the clock running for the specified timer and updates the timer's state variables for CM elapsed time, CM busy time, and idle time. Once the timer is stopped, information can be read from it. The timer can then be cleared and/or restarted. (If it is restarted without being cleared, times will be accumulated. Thus, if you start a particular timer each time a routine begins and stop it each time the routine finishes, you can read, at the end of the program, the total amount of time spent inside the routine over the course of the program.)

---


*Copyright © 1995 Thinking Machines Corporation*

# CMMD_node_timer_elapsed
# CMMD_node_timer_busy
# CMMD_node_timer_idle

Retrieve a value from the specified timer.

---

## SYNOPSIS

### C Syntax

```
double CMMD_node_timer_elapsed(int timer)
double CMMD_node_timer_busy(int timer)
double CMMD_node_timer_idle(int timer)
```

### Fortran Syntax

```
REAL*8 FUNCTION  CMMD_node_timer_elapsed(timer)
    INTEGER timer

REAL*8 FUNCTION  CMMD_node_timer_busy(timer)
    INTEGER timer

REAL*8 FUNCTION  CMMD_node_timer_idle(timer)
    INTEGER timer
```

---

## ARGUMENT

| | |
|---|---|
| *timer* | A *timer ID*, an integer from 0 to 63, identifying the timer to be read. |

## RETURNS

These functions return the elapsed, busy, or idle time, respectively, (in seconds) recorded by the specified timer between the last time the timer was cleared and the last time it was stopped.

Busy time is the time during which the timer was active and the program was actually executing, within its own OS timesharing slice, on the calling node. Idle time is the time during which the timer was active but the program was waiting in the


Version 3.3, April 1995
Copyright © 1995 Thinking Machines Corporation

CMOST dispatch loop. (There is typically very little, if any, of this last type of time in many CMMD programs.)

**NOTE:** `CMMD_node_timer_idle` and `CMMD_node_timer_busy` are relics related to early CMOST and CMMD features that no longer exist. They are irrelevant in any *pure* CMMD program (i.e., any CMMD program not containing global/local sections; see the section on global/local in the *CMMD User's Guide)*. The function `CMMD_node_timer_idle` will always return zero, and `CMMD_node_timer_busy` will always return the same value as `CMMD_node_timer_elapsed`. Therefore, `CMMD_node_timer_idle` and `CMMD_node_timer_busy` should be used only when programming in global/local mode, and not in pure CMMD programs.


*Copyright © 1995 Thinking Machines Corporation*