



EUROTHERM

THE RESOURCE MANAGER

A Guide to Tuning CMS

User Guide

© COPYRIGHT MCMXCIV EUROTHERM LIMITED

All rights strictly reserved. No part of this document may be stored in a retrieval system, or transmitted, in any form or by any means without prior written permission from Eurotherm Ltd

HA024105C004 2

M Fox



Contents

1	Scope	5
2	Related Documents	5
3	The Resources	5
3.1	Global	5
3.1.1	Shared Memory	5
3.1.2	Buffers	5
3.2	Process	6
3.2.1	Buffers	6
3.2.2	Process Queue Entries	6
4	What Happens	6
5	How to Tune	7
5.1	Global	7
5.1.1	Shared Memory	7
5.1.2	Buffers	7
5.2	Process	7
5.2.1	Buffers	7
5.2.2	Queue Entries	7
6	How to Determine What to Tune	8
6.1	Buffers	8
6.1.1	Reducing the Number of Buffers of a Size	8
6.1.2	Selection of Different Buffer Sizes	8
6.1.3	Reducing Buffer Sizes	9
6.2	Process Queue Entries	9
6.3	The Router	9
7	The Loader	9
8	The Router	10

9	Tools	10
9.1	cmsears	10
9.2	cmsspy	11
9.3	cmsedit	13
9.4	cmsunld	14
9.5	msuspy	14

VERSION HISTORY

Version	Date	Changes
1	March 10, 1994	Initial issue
2	December 20, 1994	Update for version 1.2

1 Scope

This document describes how CMS (Version 1.2) may be tuned, and what criteria should be used to determine what should be tuned.

Tuning is not a sequential, but an iterative process. It will be necessary to read this document in a similar fashion, as a single pass will not be sufficient to grasp the inter-relationships.

Coarse tuning is fairly easy to do and will probably yield results quickly. On the other hand fine tuning is a complex and time consuming process and should only be undertaken if there is seen to be a real need.

2 Related Documents

- [1] HA024105C005 A Guide to Setting Up CMS Networks

3 The Resources

There are resources requiring tuning at two levels

- The global resources, ie those that affect all processes.
- Private process resources.

3.1 Global

All global resources are located at `cmsload` (§7) time.

3.1.1 Shared Memory

When the CMS is loaded some shared memory is allocated for the database plus the process resources.

The amount of shared memory includes the router (§8) resources which may also be tuned, but this has relatively little impact on memory.

3.1.2 Buffers

There are a number of buffers allocated for global use, these buffers are used by processes when they have exhausted their own buffers.

3.2 Process

3.2.1 Buffers

Each process has allocated to it a set of buffers of various sizes. Each buffer is always in one of 3 states :

FREE Not being used

IN USE Whilst in this state one of several things may be happening :

- The buffer is being composed by the source process.
- The CMS is encoding/decoding another buffer into this buffer.
- The buffer is on the input queue of the destination process
- The buffer is being read, but not yet freed, by the destination process.

WAITING Waiting for a free input queue entry on the destination process.

The contents of each buffer is held in one 2 formats :

Native In a format directly readable on the host processor.

Universal In a format used for interchange between routers, the format is not readable on any processor.

3.2.2 Process Queue Entries

Each process has a number of queue entries distributed between 2 queues :

The Free Queue All entries in this queue are free for use

The Input Queue All entries in this queue are buffers from processes waiting to be read.

There may also be a single buffer which is **held**. This entry is a buffer from a process which has been read but not yet processed and freed.

4 What Happens

This section summarises the processes involved from the time of a process message request to the time the reply is received or the request is aborted.

- A buffer from the CMS is requested which is large enough to hold the message, if no buffer is available then the operation has failed.
- The message is composed and then issued to the CMS to send
- The CMS determines if the message can be delivered to a local process, if it cannot then the CMS requests another of its own buffers into which the message is encoded in Universal format and the initial buffer is freed.
- The CMS then attempts to deliver the message to the target CMS (which will be the local router if the message is in Universal format).

- If there is a free input queue entry on the destination process then this is allocated to the message, and thus the message is delivered. If no queue entry is available then the message is held locally on a waiting queue until an input queue entry becomes available.
- If the message is being delivered by routers then eventually the message is either delivered to the destination process input queue by the router local to the destination process or the message is lost if it is not possible to deliver it. This last router will deliver the message to the destination process in universal format and the process will then decode it before delivering it.

5 How to Tune

This section describes all the tunable resources and how to tune them (§6 describes how to determine what needs tuning).

5.1 Global

5.1.1 Shared Memory

The default shared memory is only 20000 bytes and is insufficient for many applications. During tuning it is best to allocate as much as is possible, say 500000, and make the shared memory size the last thing tuned, unless it is causing a problem using so much excessive memory.

The **msuspy** tool (§9.5) determines how much memory has been used, and may be used as the argument to the CMS loader (§7).

5.1.2 Buffers

The set of global buffers is selected with the **-i** option of **cmsload** (§7).

5.2 Process

5.2.1 Buffers

The buffer distribution of a process is generally altered by use of a **-i** option (Consult the relevant guide for the CMS process).

To any buffer distribution CMS adds one buffer of maximum buffer size (currently 4096). This buffer is added to ensure that at least one instance of any message can be generated by each process.

5.2.2 Queue Entries

CMS allocates one queue entry for every CMS buffer. This works on the principle that for every request (requiring a buffer) there will be a corresponding reply. However the messaging is not synchronous and therefore this balance may not be correct. It is possible to increase the number of queue entries with the **cmsedit** tool (§9.3).

6 How to Determine What to Tune

A CMS nodes and its processes may be tuned to according to several criteria :

- To reduce memory usage.
- To reduce delays.

In general a balance will have to be struck.


6.1 Buffers

The distribution of CMS buffers is the most complex area of tuning and several iterations may be required to obtain a very fine tune. This applies equally to private process buffers and to global buffers.

6.1.1 Reducing the Number of Buffers of a Size

Having selected a buffer distribution it may be observed using the tools (`cmsspy` §9.2) that certain buffers are never used, if this is the case then these buffers may be eliminated.

6.1.2 Selection of Different Buffer Sizes

By selecting a small set of buffer sizes, say maximum 6, this will help reduce the number of buffers and so the time required to allocate a buffer. These selected buffer sizes ought not to be close together unless after significant examination this is found to be ideal for this process, in general it will not. It is not in general a good idea to allocate a buffer for every envisaged operation as all operations will not be concurrent. It is also not a good idea to allocate small amounts of a buffer size, in general it would be better to allow such operations to use a larger buffer. 

Examples of generally poor distributions :

- “2:500 2:520 2:560 3:584 4:640 6:4000”, sizes too close
- “8:256 1:1024 6:4096”, single buffer of 1024

Examples of generally good distributions :

- “8:640 6:4000”
- “4:64 4:256 4:1024 4:4096”

The frequency of use of a particular size ought to be observed, and if it is noted that a particular size is very rarely used it might prove beneficial to merge with the next larger size, and possibly reduce the overall number of buffers.

It is advisable to reduce the numbers of private process buffers that are allocated for very occasional use and to instead rely upon global buffers unless it is known that these buffers are “essential” are therefore they must be available when required.

6.1.3 Reducing Buffer Sizes

All buffers are initialised to contains all zeros, and thus if the tail end of buffer is seen to contain only zeros then it may be possible to deduce that not all of the buffer has been used. When a buffer is obtained the byte at the size requested is written to with a non-zero value to assist in detection of usage (see buffer usage column in figure 2). By inspecting a group of buffers (using **cmsspy** §9.2) of the same size, and inspecting the whole of each buffer it may be possible to deduce that the full size of the buffers is not used, and therefore the size of this group of buffers may be reduced. It is not possible to determine that the space has never been used, only to determine that a minimum has been used.

This type of tuning is very difficult, and ought only to employed to obtain a very fine tune.

6.2 Process Queue Entries

If a process is being tuned for speed, then the tools (**cmsears** §9.1 and **cmsspy** §9.2), should be used while the process is running to determine if any process has had to hold the delivery of a buffer due to the lack of queue entries at destination process(es). The inspection should attempt to determine if any process in particular is holding up the delivery of messages, the number of queue entries for this process should be increased until the number of WAITs reduces to an acceptable level, and then the increment added using the tools (**cmsedit** §9.3) whenever the process is loaded.

6.3 The Router

The router process will be less easy to determine, and will require more experimentation. The tuning of the router is very important as all inter node messages go through the routers.

As a general rule the router requires large amounts of buffers of maximum size to operate at speed, however this can become quite costly in terms of memory very quickly. The reason is that every incoming inter node message is obtained in Universal format in a buffer of the largest size.

The number of queue entries allocated to the process by default ought to be sufficient, but at worst it will need as many queue entries as there are buffers on the whole node.

7 The Loader

cmsload accepts the following tuning options:

- a <MaxAccessPoints> The maximum number of access points in the node, default 16.
- e <MaxAEs> The maximum number of application entities in the node, default 8. This must be <= <MaxAccessPoints>.
- i <Buffers> The global buffer distribution, default is "".
- p <MaxProtocols> The maximum number of protocols supported by this node, default 4. This must be <= <MaxAEs>.
- quiet Suppress informational messages.
- size <Size> The amount of shared memory to be allocated.
- u <MaxProcesses> The maximum number of processes in the node, default 10.
- y <Key> Shared memory key. Only required if more than one CMS loaded on a node.

The numbers of addressable quantities can be obtained from **cmsspy**, see figure 3.

8 The Router

The following tuning options apply to the router.

-i <Buffers> The CMS buffer distribution

For a full description of the router options see [1].

9 Tools

The following tools are supplied with CMS to assist in tuning and debugging. Each of the tools also accepts the **-y <Key>** option if more than one CMS is loaded on the node.

9.1 cmsears

The **cmsears** tool is a dynamically updating tool that displays the current values of certain CMS parameters. All the information is displayed in tabular form :

Identities Informational, not related to tuning.

Proc The process name (this may be truncated if long), and the process number.

O/S The operating system identifier.

Lock By The process number of the process currently locking the input queue. (0 implies no-one is locking).

Queues The process queues.

Free The number of currently used queue entries.

In The number of buffers on the input queue.

Buffers The buffers owned by this process.

Free The number of unused buffers.

Out The number of in use buffers.

Wait The number of buffers held waiting on a destination process.

Statistics The CMS statistics, the most useful for tuning.

Load The number of times this process has been loaded.

Get Fail The number of times the process has failed to get a buffer, because either there was not free buffer large enough.

Wait Out The number of times a buffer has been held waiting for delivery to another process.

Wait In The number of times this process has caused other buffers to held waiting because of insufficient queue entries.

The following options may be applied to **cmsears**

-z <Seconds> Refresh rate, -1 implies single-shot.

-notANSI Do not use ANSI escape sequences. This means update is a scroll.

Leaving **cmsears** running may degrade performance.

Figure 1 is a sample output from **cmsears**. This shows 8 processes where 2 processes “tst3:second” and “tst3:first” have buffers on their input queues from processes “tst3:second” and “tst3:third”. The router is process “Router” marked by *.

Figure 1: Example output from cmsears

CMS 149.121.128.29												
<--Identities-->			<Queues>			<--Buffers-->			<----Statistics---->			
Proc	O/S	Lock	Free	In	Free	Out	Wait	Load	Get	Wait	Wait	
		By							Fail	Out	In	
* Router	1	7413	0	17	0	17	0	0	1	0	0	0
Networker	2	7416	0	9	0	9	0	0	1	0	0	0
RouterLogger	3	7415	0	5	0	5	0	0	1	0	0	0
tst2:unix1	4	8069	0	22	0	22	0	0	1	0	0	0
tst2:unix2	5	8070	0	25	0	25	0	0	1	3	0	0
tst3:second	6	8142	0	9	1	9	1	0	1	0	0	0
tst3:third	7	8143	0	10	0	8	2	0	1	0	0	0
tst3:first	8	8141	0	8	2	10	0	0	1	0	0	0

9.2 cmsspy

The cmsspy tool may be used to peruse the CMS information of a process. This tool is especially useful to perform post-mortem analysis. The tool is interactive and allows the user to inspect such things as

Statistics A comprehensive statement of CMS statistics, including a breakdown of which processes held this process waiting and which processes it held waiting.

Buffer Uses The number of times each buffer has been used.

Buffer contents The contents of a buffer. If used with the -b -z options then the whole buffer can be displayed including how much of the tail of the buffer is still all zeros.

The following options may be applied to cmsspy to alter the defaults

-buffer Display all of the buffer (past BuffSize), default is to terminate display at BuffSize.

-format <Format> The display format of each byte in a buffer, default is [%02X].

-quiet Do not print entry text and prompts (for use in scripts).

-zero Count up trailing zeros and display only the count.

These switches may be applied to any interactive command line to alter these parameters from the default for that line only.

cmsspy prompts for the input it requires which is one of the items in the list that it has just printed. At any point one of the following commands may be entered.

help Give help on the current context.

help <Item> Give help on an item listed in the above help.

quit Quit the tool.

Figure 2 is a sample cmsspy session

This example shows process “tst2:unix2” being inspected. The following observations can be made

- This process has not been held waiting nor has it held any other process waiting.

Figure 2: Example output from cmsspy

```

Global [g], Process [p], Router[r]> p
  ID Name
  1 Router
  2 Networker
  3 RouterLogger
  4 tst2:unix1
  5 tst2:unix2
Enter process> tst2:unix2
Addresses [a], Buffers [b], Identifiers [i], Queues [q], Statistics [s]> i
Process Name:      tst2:unix2
Process Number:    5
Operating system Id: 8070
Addresses [a], Buffers [b], Identifiers [i], Queues [q], Statistics [s]> s
Init():            1
GetGlobalBuffer(): 1
GetBuffer() failures: 3
Deliver() failures: 0
Encode() failures:  0
Decode() failures:  0
Total waits:       0
Access points [a], Application Entities [e], Protocols[o]>
Addresses [a], Buffers [b], Identifiers [i], Queues [q], Statistics [s]> b
  Address  Max  State  Uses  Usage  Size  MsgSize  AccessPt  Format
11033498   64  Free    25    51    47     47     71  Native
1103341C   64  Free    23    51    47     47     71  Native
110333A0   64  Free    21    51    47     47     71  Native
11033324   64  Free    21    51    47     47     71  Native
110332A8   64  Free    21    51    47     47     71  Native
1103322C   64  Free    21    51    47     47     71  Native
110331B0   64  Free    21    51    47     47     71  Native
11033134   64  Free    21    51    47     47     71  Native
11032DD0   64  Free
110337CC  288  Free    25   118   118    118     71  Native
11033670  288  Free     3    81    84     84     71  Native
11033514  288  Free     3    81    84     84     71  Native
110199C8 1024  Free     2   316   316    316     71  Native
11019E04 1024  Free     1    81    84     84     71  Native
1101A240 1024  Free     1    81    84     84     71  Native
1101A67C 1024  Free     1    81    84     84     71  Native
110182B4 4096  Free     1    81    84     84     71  Native
1101AAB8 4096  Free     1    81    84     84     71  Native
Enter buffer address> 11033670 -b -z
  Address  Max  State  Uses  Usage  Size  MsgSize  AccessPt  Format
11033670  288  Free     3    81    84     84     71  Native
[02] [00] [00] [00] [0F] [02] [00] [00] [27] [8E] [95] [2B] [00] [00] [02] [00]
[03] [00] [00] [00] [95] [79] [80] [1D] [00] [00] [00] [00] [00] [00] [00] [00]
[00] [00] [00] [00] [00] [00] [00] [00] [52] [4D] [50] [00] [45] [00] [00] [00]
[00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00] [00]
[00] [00] [00] [00] [00] [00] [00] [00] [01] [00] [10] [04] [01] [00] [00] [10]
[21] <*3 ZEROS*><*END SIZE*><*204 ZEROS*><*END MAX*>
Enter buffer address>
Addresses [a], Buffers [b], Identifiers [i], Queues [q], Statistics [s]> q
Free [f], Input [i], Hold [h]> i
  Address  Max  State  Uses  Usage  Size  MsgSize  AccessPt  Format
11032C14   384  Out    27   118    47     47     69  Native
  Length = 1

```

Figure 3: Example output from cmsspy

```

The CMS Spy Tool, Version 1.2
(c) Copyright 1992, 1993, 1994 Eurotherm Controls Limited

Global [g], Process [p], Router[r]> g
Addressing[a], Buffers[b], Deleted Resources[d], Sizing[s]> s
  5/10 Processes
  7/16 AccessPoints
  4/8 ApplicationEntities
Addressing[a], Buffers[b], Deleted Resources[d], Sizing[s]> a
AccessPoints[a], ApplicationEntities[e], Node[n]> a
AccessPt Protocol Process
    33 ROUT    Router
    34 ****    Router
    35 ROUT    Networker
    36 ROUT    RouterLogger
    37 RMP      tst2:unix2
    38 RMP      tst2:unix2
    39 RMP      tst2:unix1
AccessPoints[a], ApplicationEntities[e], Node[n]> e
Name                      AccessPt Protocol Process
Router                     33 ROUT    Router
tst2:unix2                  37 RMP      tst2:unix2
tst2:                       38 RMP      tst2:unix2
tst2:unix1                  39 RMP      tst2:unix1
AccessPoints[a], ApplicationEntities[e], Node[n]> n
149.121.128.29
AccessPoints[a], ApplicationEntities[e], Node[n]> quit

```

- The process has failed to get a buffer 3 times. The buffer that it was requesting must have been greater than 64 bytes as there is at least one buffer of that size that has never been used.
- The process requires many buffers of size 81 bytes, or less, yet very few greater than that. Therefore some of the buffers of sizes 288, 1024 and 4096 could be reduced in size, to say 320 bytes sufficient to hold the other messages of 316 and 118 bytes.
- The process has one buffer on its input queue.

9.3 cmsedit

The `cmsedit` tool may be used to modify the CMS for a process.

`cmsedit <Name> -e <Num Entries>` To add queue entries (but not buffers) to an unloaded or uninitialised process. Can be used in a start up script before the process is loaded.

`cmsedit <Name> -z` To reset the CMS statistics on an process.

`cmsedit` also accepts the following options

`-loaded` Do the operation on the process even if it is still loaded.

`-quiet` Suppress informational messages.

Figure 4: Example output from msuspy

```
The Shared Memory Spy Tool, Version 1.2
(c) Copyright 1992, 1993, 1994 Eurotherm Controls Limited
CMS shared memory
1048576 bytes starts at 0x11000000
833464 bytes free from 0x11034848
215112 bytes used
```

9.4 cmsunld

The `cmsunld` is used to unload and/or delete CMS processes. To unload a CMS process is to terminate that process and make all its addresses invalid, ie it can no longer be addressed. To delete a process is to unload and to then free up its buffers and process queue entries so that they can be re-allocated to another process on loading.

The tool operates in 3 modes.

Whole node `cmsunld`. This is always a delete operation.

Single process `cmsunld <Process Name>`. This is an unload unless `-delete` supplied in which case the process is also deleted. it can be used to delete an already unloaded process.

All processes `cmsunld *`. All the processes are unloaded but the CMS shared memory database is still in existence.

`cmsunld` accepts the following options

-delete Delete as well as unload the process.

-notermminate Do not attempt to terminate the CMS processes. This is used in environments where to attempt to terminate a non-existent process may cause problems.

9.5 msuspy

The `msuspy` tool can be invoked to determine how much shared memory has been used. This may then be used as the `-size` parameter to the `cmsload` (§7).

Figure 4 shows sample output from `msuspy -g CMS`.

