

*Context  
Manager/VM™*



# *Context Manager/VM Manual*

Copyright © 1987 by Convergent Technologies, Inc.,  
San Jose, CA. Printed in USA.

**Second Edition (May 1987) 09-01050-01**

All rights reserved. No part of this document may be reproduced, transmitted, stored in a retrieval system, or translated into any language without the prior written consent of Convergent Technologies, Inc.

Convergent Technologies makes no representations or warranties with respect to the contents hereof and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Convergent Technologies reserves the right to revise this publication and to make changes from time to time in its content without being obligated to notify any person of such revision or changes.

Convergent Technologies and NGEN are registered trademarks of  
Convergent Technologies, Inc.

Art Designer, Chart Designer, ClusterCard, ClusterNet, ClusterShare,  
Context Manager/VM, Convergent, CT-DBMS, CT-MAIL, CT-Net, CTIX,  
CTOS, CTOS/VM, DISTRIX, Document Designer, The Operator, AWS,  
CWS, IWS, S/50, S/120, S/160, S/220, S/320, S/640, S/1280, Multibus,  
TeleCluster, Voice/Data Services, Voice Processor, WGS/Calendar,  
WGS/Desktop Manager, WGS/Mail, and X-Bus are trademarks of  
Convergent Technologies, Inc.

1 *Before You Begin*

|                                      |     |
|--------------------------------------|-----|
| What Is Context Manager/VM?.....     | 1-1 |
| Ways to Use Context Manager/VM ..... | 1-2 |
| Business Tasks .....                 | 1-2 |
| Programming .....                    | 1-2 |
| Communications.....                  | 1-2 |
| CM Features .....                    | 1-3 |
| CM and Window Services .....         | 1-3 |
| How CM Works.....                    | 1-4 |
| Terms Used in this Manual .....      | 1-4 |
| Manual Organization .....            | 1-5 |
| Related Documentation .....          | 1-6 |

2 *Experimenting with Context Manager/VM*

|  |      |
|--|------|
| Using the Mouse.....   | 2-1  |
| The Context Manager Screen .....   | 2-2  |
| The Action Key .....   | 2-5  |
| Summary of Status Terms Displayed .....  | 2-6  |
| An Exercise with CM .....  | 2-6  |
| Starting CM .....  | 2-7  |
| Moving the Highlight Bar .....   | 2-7  |
| Starting an Application .....  | 2-8  |
| Overlaying the CM Screen on an Application<br>Screen; Returning to the Current Context ..... | 2-8  |
| Starting a Second Application .....  | 2-9  |
| Switching Contexts Without Using the CM<br>Screen .....                                      | 2-10 |

|  |      |
|--|------|
| Starting More Contexts .....                   | 2-11 |
| Finishing from Within a Context .....          | 2-11 |
| Eliminating a Context Without Saving .....     | 2-12 |
| Ending Your Session with CM (Logging out) .... | 2-12 |

### 3 *Basic Concepts*

|   |      |
|---|------|
| Starting an Application .....                       | 3-1  |
| Assigning Function Keys .....                       | 3-2  |
| Function Key Menu .....                             | 3-2  |
| Function Keys .....                                 | 3-2  |
| Preassigned Function Keys .....                     | 3-3  |
| Assigned Function Keys .....                        | 3-3  |
| Starting Applications that Require Parameters ..... | 3-4  |
| Contexts You Can Return To .....                    | 3-5  |
| Cycling Through Contexts .....                      | 3-5  |
| Halting a Context .....                             | 3-6  |
| The Bullet .....                                    | 3-6  |
| Status Terms .....                                  | 3-6  |
| Swapping a Context to Disk .....                    | 3-7  |
| How Swapping Works .....                            | 3-7  |
| Contexts that Are Not Swapped .....                 | 3-8  |
| Time Savings: Swapping Versus Starting .....        | 3-9  |
| Using the Context Manager to Cut and Paste .....    | 3-9  |
| The Cut and Paste procedure .....                   | 3-11 |
| Getting Out .....                                   | 3-13 |
| Finishing a Context .....                           | 3-13 |
| Eliminating a Context Without Saving .....          | 3-13 |
| Ending a Session (Logging out) .....                | 3-14 |
| CM Version .....                                    | 3-14 |
| CM Configuration File Name .....                    | 3-15 |
| Mail Reporting .....                                | 3-15 |

### 4 *Using Windows and the Mouse*

|                         |     |
|-------------------------|-----|
| What Is a Window? ..... | 4-2 |
| Using Windows? .....    | 4-2 |
| Example .....           | 4-3 |
| Starting Windows .....  | 4-3 |
| The Windows Menu .....  | 4-3 |

|   |      |
|---|------|
| Looking at Windows .....                                | 4-7  |
| Making a Window Current .....                           | 4-7  |
| Sizing and Moving Windows .....                         | 4-8  |
| Using the Mouse with Windows .....                      | 4-9  |
| What Is a Mouse? .....                                  | 4-9  |
| Using the Mouse .....                                   | 4-10 |
| Moving the Mouse .....                                  | 4-11 |
| Using the Mouse Buttons .....                           | 4-13 |
| Exercises .....   | 4-17 |
| Managing Windows Using the Mouse .....                  | 4-17 |
| Opening Windows .....                                   | 4-17 |
| Sizing Windows .....                                    | 4-18 |
| Moving Windows .....                                    | 4-20 |
| Activating Commands on the Windows Menu .....           | 4-22 |
| Using the Tile Command .....                            | 4-22 |
| Using the Icon Command .....                            | 4-25 |
| Using the Full Command and the Restore<br>Command ..... | 4-28 |
| Using Windows with the Keyboard .....                   | 4-30 |
| Exercises .....   | 4-31 |
| Managing Windows Using the Keyboard .....               | 4-31 |
| Opening Windows—Activating the Windows<br>Menu .....    | 4-32 |
| Sizing a Window .....                                   | 4-32 |
| Moving Windows .....                                    | 4-35 |
| Using the Tile Command .....                            | 4-35 |
| Using the Icon Command .....                            | 4-38 |
| Using the Full Command and the Restore<br>Command ..... | 4-41 |

## 5 *Setting Up Context Manager/VM*

|   |     |
|---|-----|
| Overview .....                            | 5-1 |
| The Swap File .....                       | 5-2 |
| Creating the Swap File .....              | 5-2 |
| CM and Installed System Services .....    | 5-3 |
| Installation and Deinstallation .....     | 5-3 |
| CM Version Number .....                   | 5-4 |
| Editing the User Configuration File ..... | 5-4 |
| Configuring the Mouse .....               | 5-5 |
| A Note About the Number of Contexts ..... | 5-6 |

|   |      |
|---|------|
| Using the CM Configuration File Editor .....    | 5-6  |
| Entering from the Executive .....               | 5-7  |
| Experimenting with the CM Editor .....          | 5-8  |
| Screen Areas and Functions .....                | 5-8  |
| Message Line .....                              | 5-8  |
| Input/Error Line .....                          | 5-11 |
| Command Editing Area .....                      | 5-11 |
| Function Key Menu .....                         | 5-19 |
| Command List Area .....                         | 5-21 |
| Exiting the CM Editor .....                     | 5-22 |
| Examples .....                                  | 5-22 |
| A Complete New Entry .....                      | 5-22 |
| Editing a Command .....                         | 5-25 |
| Removing a Command .....                        | 5-25 |
| Renaming a Command .....                        | 5-26 |
| A Common Error: Inconsistent Entries .....      | 5-26 |
| Changing a Configuration with Executive         |      |
| Commands .....                                  | 5-26 |
| CM Add Application .....                        | 5-27 |
| CM Remove Application .....                     | 5-28 |
| Applications You Can Start from CM .....        | 5-28 |
| Defining an Executive Submit Command in CM .... | 5-29 |
| Create a New Command .....                      | 5-30 |
| Create a Submit File .....                      | 5-30 |
| Test Your Submit Command .....                  | 5-31 |
| Define the Command in the CM Editor .....       | 5-31 |
| Finish up .....                                 | 5-31 |
| Invoking the Submit File From CM .....          | 5-31 |
| Text of the Configuration File .....            | 5-32 |
| Allowing for More Contexts .....                | 5-33 |
| Example .....                                   | 5-34 |

## 6 *Notes for the Programmer*

|  |     |
|--|-----|
| CM and Its Relationship to the Operating System .. | 6-1 |
| What Runs under CM? .....                          | 6-1 |
| Programs that Write Directly to the Screen Map .   | 6-2 |
| "Busy Wait" Loops .....                            | 6-4 |
| Low-Memory Interrupt Vector Table .....            | 6-4 |
| Positioning the Cursor .....                       | 6-5 |

|   |      |
|---|------|
| Applications Suspended in Background .....          | 6-5  |
| Programs That Run Under Windows .....               | 6-6  |
| Exit Run File .....                                 | 6-8  |
| Applications that Cannot Be Swapped .....           | 6-8  |
| Communication with CM .....                         | 6-9  |
| Communication Between Applications .....            | 6-9  |
| Intercontext Message Server .....                   | 6-9  |
| Procedural Interfaces .....                         | 6-10 |
| CM as a Server .....                                | 6-11 |
| Parent/Child Relationship of Contexts .....         | 6-11 |
| Context Handle .....                                | 6-12 |
| Starting and Switching Contexts .....               | 6-13 |
| Check for Installation .....                        | 6-14 |
| CM Screen .....                                     | 6-14 |
| Procedural Interfaces .....                         | 6-15 |
| Placing Information in the CM Configuration File .. | 6-16 |
| Invoking Programs that Use Run-Time Libraries ...   | 6-16 |
| Using CrashDump.sys as Your Swap File .....         | 6-17 |
| Estimating Memory Requirements .....                | 6-17 |
| Determining the Size of a Partition .....           | 6-19 |
| A Variable Sized Partition .....                    | 6-19 |
| A Fixed Sized Partition .....                       | 6-20 |
| Note for the System Service Writer .....            | 6-21 |

## 7 *Operations*

|                                 |      |
|---------------------------------|------|
| CMCurrentVersion .....          | 7-2  |
| CMQueryConfigFile .....         | 7-4  |
| CMQueryContextHandle .....      | 7-6  |
| CMQueryErc .....                | 7-7  |
| CMQueryParent .....             | 7-8  |
| CMSetParent .....               | 7-10 |
| CMStartAppl .....               | 7-12 |
| CMStartApplByBlock .....        | 7-17 |
| CMStartApplByName .....         | 7-20 |
| CMSwitchContext .....           | 7-23 |
| CMSwitchToExistingContext ..... | 7-24 |
| CMTerminateContext .....        | 7-26 |
| CMTranslateChToPh .....         | 7-28 |
| CMTranslatePhToCh .....         | 7-30 |
| ICMSCheck .....                 | 7-32 |

|                          |      |
|--------------------------|------|
| ICMSCurrentVersion ..... | 7-34 |
| ICMSFlush .....          | 7-35 |
| ICMSSend .....           | 7-36 |
| ICMSWait .....           | 7-38 |
| NotifyCM .....           | 7-40 |

## 8 Troubleshooting

|                                   |     |
|-----------------------------------|-----|
| Status Codes .....                | 8-1 |
| Context Manager/VM .....          | 8-1 |
| Intercontext Message Server ..... | 8-4 |
| Status Messages .....             | 8-5 |

|                       |            |
|-----------------------|------------|
| <i>Glossary</i> ..... | <i>G-1</i> |
|-----------------------|------------|

|                    |            |
|--------------------|------------|
| <i>Index</i> ..... | <i>I-1</i> |
|--------------------|------------|

## List of Figures

| Figure |   | Page |
|--------|---|------|
| 2-1.   | The Mouse and the Mark Button .....               | 2-2  |
| 2-2.   | Context Manager Screen .....                      | 2-3  |
| 3-1.   | The Cut Menu .....                                | 3-11 |
| 3-2.   | Cutting Text .....                                | 3-12 |
| 4-1.   | Three Overlapping Windows .....                   | 4-2  |
| 4-2.   | The Windows Menu .....                            | 4-4  |
| 4-3.   | The Mouse Buttons: Mark, Menu, and<br>Bound ..... | 4-10 |
| 4-4.   | A Resized Window .....                            | 4-19 |
| 4-5.   | Two Tiled Windows .....                           | 4-23 |
| 4-6.   | Four Tiled Windows .....                          | 4-24 |

| <b>Figure</b> |                                      | <b>Page</b> |
|---------------|--------------------------------------|-------------|
| 4-7.          | A Current Window and an Icon .....   | 4-27        |
| 4-8.          | A Resized Window .....               | 4-34        |
| 4-9.          | Two Tiled Windows .....              | 4-36        |
| 4-10.         | Four Tiled Windows .....             | 4-37        |
| 4-11.         | A Current Window and an Icon .....   | 4-40        |
| 5-1.          | CM Editor Screen.....                | 5-9         |
| 5-2.          | Example of a Configuration File..... | 5-32        |

### *List of Tables*

| <b>Table</b> |  | <b>Page</b> |
|--------------|--|-------------|
| 4-1.         | Menu Commands .....                        | 4-5         |
| 4-2.         | Mouse Buttons .....                        | 4-14        |
| 4-3.         | Summary: Windows and the Mouse .....       | 4-15        |
| 4-4.         | Making Hidden Window Contents Visible..... | 4-21        |
| 4-5.         | Summary: Windows and the Keyboard.....     | 4-30        |
| 5-1.         | CM Editor Function Key Menu .....          | 5-10        |



## *What Is Context Manager/VM?*

Context Manager/VM is a software product that works with the operating system to allow you to run several applications on your workstation at once.

Without Context Manager/VM (from now on called CM), you worked with one application at a time. Whenever you wanted to start a new application, you had to first finish the application in which you were working.

With CM, when one application is on the screen, you can instantaneously switch to another with a single keystroke. Then you can return to the first one and continue working at exactly the point where you were before.

# *Ways to Use Context Manager/VM*

## **BUSINESS TASKS**

You may be in a situation where you need to read an electronic mail message from your company's finance department, finish up the spreadsheets you are working on in Extended Multiplan, copy them to a floppy disk using the Executive, write a cover memo in the Document Designer, and send the spreadsheets out to the address supplied in the finance department memo.

With CM, you can do more than one of these tasks at once: Having finished the spreadsheets, you can start the **Copy** command in the Executive, and then switch to the Document Designer to write your memo while Copy is still running.

## **PROGRAMMING**

You can write code in the current context while another program compiles in a background context. You can check the status of the background context at any time by pressing **Action-Go** to look at the CM screen.

If you need to see display output from a context that is running in background, many applications allow you to pause the output until you choose to examine it in foreground.

## **COMMUNICATIONS**

You may be logged onto an information data base using Asynchronous Terminal Emulator (ATE), or to a mainframe using SNA 3270. You bring up the CM screen and notice that electronic mail has arrived.

You can switch contexts, read your mail, and then return to ATE or SNA without ever having to terminate your communications session.

## *CM Features*

CM has a single screen of its own, called the CM screen, to which you can return at any time. On the right, this screen lists the applications that you can start. On the left, it shows applications you have started and the status of each one. (See Figure 2-2.)

You can select an application from the CM screen by using either your keyboard or a hand-held pointing device called a Mouse. (The Mouse is an optional feature.) If you have a Mouse, you select an application by clicking one of the Mouse buttons. (See "Using the Mouse" in Chapter 4.)

CM is not limited to a fixed set of applications. You can configure your system to run whatever applications are important to you.

Applications may be CTOS-based, or they may run under operating systems hosted by CTOS (for example, MS-DOS).

Because the CM environment is flexible, you can add your own application to it with little or no special programming.

### **CM AND WINDOW SERVICES**

Window Services (from now on called Windows) is a system service that interacts with CM to allow you to manipulate your contexts with windows. (A *window* is the rectangular portion of your screen within which an application is displayed.)

If you have Windows, you can arrange your applications on the screen so that you can see multiple applications at once. You work with one application while the others remain in view on the screen; you can switch to another by using either the Mouse or keyboard. You can use the Mouse or keyboard to easily change the size of these applications' windows and/or move them to different locations on the screen.

For more information on Windows, see Chapter 4, "Using Windows and the Mouse."

CM, with or without Windows, allows you to run multiple copies of the same application. For example, when you are programming, you can run several copies of the Executive. With Windows, you can view all these copies at the same time.

## *How CM Works*

Once an application has been started, it is called a *context*. You interact with one context at a time. This context is called the *current context*. The current context is the only context that responds to the use of the Mouse and keyboard. When you switch contexts, you make the next context the current context.

You may need to start one more application than will fit in workstation memory. When this occurs, one or more applications can be swapped to a hard disk (stored temporarily in a suspended state in a Disk Swap file). If you have an 80286 Processor, applications can be swapped into extended memory. (See the *CTOS/VM Concepts Manual* and the CTOS/VM Release Notice for more information on swapping to extended memory.)

CM can handle as many as 10 contexts at once; the operating system switches them back and forth from workstation memory to disk, as necessary.

## *Terms Used in this Manual*

Some common words and phrases have specific meanings as they are used in this manual. They are italicized in the text when they are defined. These terms (except where noted) are also used with Windows.

To *start an application*, choose it from the list on the CM screen and press **Go**. Once an application is started, it is called a *context* and is said to be *active*. You see and interact with one context on the screen. It is called the *current context* and is said to *own the screen*. With Windows, although you may see more than one context, you still interact with only one of them. This context is called the *current window* and is also said to own the screen.

The context that owns the screen (the current context) is *running in foreground*. Any other active context is *running in background* unless it is swapped. An active context is *swapped* if it has been stored temporarily on a hard disk. While a context is swapped, it is not running, but is *suspended*.

When you hold down one key and press another, such as **Action** and **Go** or **Action** and a function key (fn), the combination is symbolized as **Action-Go** or **Action-fn**. When you press certain keys to cause one context to be replaced by another on the screen, you are *switching contexts*. In this way, you can choose to work in any one of the active contexts by making it the current context.

## *Manual Organization*

Before you read this manual, you should be familiar with material in the *Operator's Guide* (including keyboard layout) and in the manuals for the various applications you plan to run under CM. This manual assumes you have this knowledge.

For memory and operating system requirements and for instructions for placing CM software on your system, see the Context Manager/VM Release Notice.

The rest of this manual consists of the following chapters; each has a different purpose.

Chapter 2, "Experimenting with Context Manager/VM," takes you through a series of exercises designed to show you how CM works.

Chapter 3, "Basic Concepts," gives more specific information about CM.

Chapter 4, "Using Windows and the Mouse," describes the operation of Windows with the Mouse and keyboard. Step-by-step exercises are provided. If you are using Windows and a Mouse for the first time, you need to read this chapter, along with Chapter 2.

Chapter 5, "Setting up Context Manager/VM," tells how to prepare for and configure CM. If you are setting up your own system, read Chapters 3 and 5.

Chapter 6, "Notes for the Programmer," contains information for the programmer writing applications to run under CM. If you are writing such programs, you must read this. You may also find it helpful to read Chapters 3 and 5.

Chapter 7, "Operations," describes procedural interfaces for the CM and ICMS operations.

Chapter 8, "Troubleshooting," describes common errors, along with suggestions for correcting them. Status codes for CM and ICMS are also reported.

## *Related Documentation*

The documents described below provide additional information related to the contents of this manual.

For a complete list of Convergent Technologies publications, see the "Guide to Technical Documentation" in the *Executive Manual* or a similar command-line interpreter manual for your operating system.

### **Introductory**

- Executive Manual
- Operator's Guide
- Status Codes Manual

### **Operating System**

- CTOS/VM Concepts Manual
- CTOS/VM Reference Manual

### **Program Development Tools**

- Editor
- Linker/Librarian Manual

## **Office Automation**

Document Designer Reference Manual  
Word Processing Reference Manual

## **Related System Services**

Mouse Services Manual  
Window Services Manual

## **Release Notices**

Release Notice for Context Manager/VM  
Release Notice for CTOS/VM

The following paragraphs outline the contents of these manuals.

The *Executive Manual* describes the interactive command interpreter, the program that first interacts with the user when the system is turned on. It describes available commands and discusses command execution, file management, program invocation, and system management. It also addresses status inquiry and volume management.

The *Operator's Guide* addresses the needs of the average user for operating instructions. It describes the workstation switches and controls, keyboard function, and floppy disk handling.

The *Status Codes Manual* contains a complete listing of all status codes that can be generated by a CTOS workstation or a Shared Resource Processor (SRP). It includes bootstrap ROM error codes and CTOS initialization codes. The codes are listed in numerical order with messages and explanations. The manual also describes and interprets crash status codes.

The *CTOS/VM Concepts Manual* and the *CTOS/VM Reference Manual* describe the operating system. They specify services for managing processes, messages, memory, exchanges, tasks, video, disk, keyboard, printer, timer, communications, and files. In particular, they specify the standard file access methods: SAM, the sequential access method; RSAM, the record sequential access method; and DAM, the direct access method.

The *Editor Manual* describes the text editor.

The *Linker/Librarian Manual* describes the Linker, which links separately compiled object files, and the Librarian, which builds and manages libraries of object modules.

The *Document Designer Reference Manual* is a reference tool for users of the Document Designer. The manual discusses the capabilities of the Document Designer and describes in detail each of its commands. Information is included on integrating application systems with the Document Designer, generating automatic tables of contents, including voice annotations in documents, list processing, text manipulation, and system configuration.

The *Word Processing Reference Manual* is a reference tool for users already familiar with the Word Processor. The manual discusses efficient use of the various facilities of the Word Processor and describes in detail each Word Processing command. Information is included on list processing, programmer-specific operations, and printer and print wheel configurations.

The *Mouse Services Manual* describes the Mouse Server and object module library for applications programmers. It also includes a short description of end-user commands.

The *Window Services Manual* describes the procedures that are necessary to manipulate windows.

This chapter provides a general introduction to CM by taking you through a series of steps that show you how CM works. To learn more about CM, see Chapter 3, "Basic Concepts."

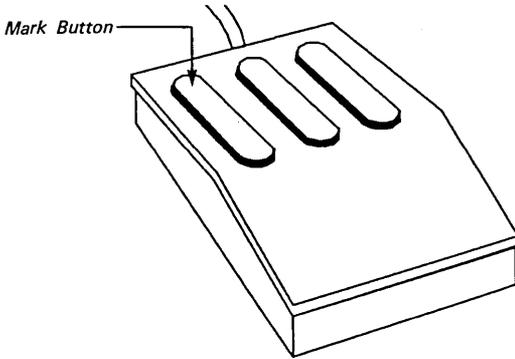
You can select applications from the CM screen by using either your Mouse or keyboard.

### *Using the Mouse*

If you have a Mouse, you can choose an application by highlighting the selected area with the Mouse cursor.

The Mouse has three buttons. To start an application from the CM screen, click (press and release) the **Mark** Mouse button. (See Figure 2-1.) The other two buttons, along with the **Mark** button, are used to manipulate windows, as explained in Chapter 4. (For detailed information on handling the Mouse, see "Moving the Mouse" in Chapter 4.)

(The Mouse is set up for right-handed people, but it can be changed for left-handed use. See "Configuring the Mouse" in Chapter 5, or contact your system administrator.)



1185-001

**Figure 2-1. The Mouse and the Mark Button**

## *The Context Manager Screen*

Figure 2-2 shows an example of the CM screen. The boxed summary on the next page tells you how to use the screen.

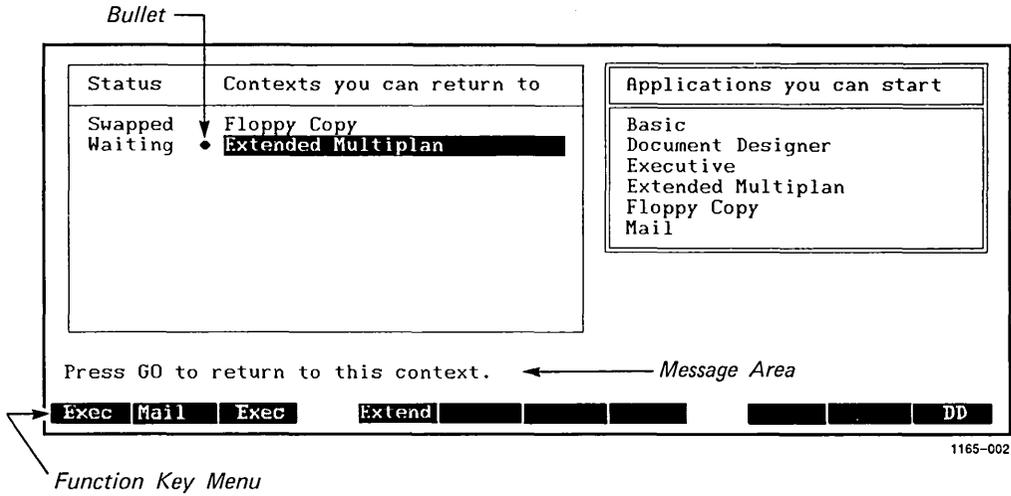


Figure 2-2. Context Manager Screen

---

## SUMMARY: USING CM

### When the CM screen is up

The **Left** and **Right Arrow** keys move the highlight bar between "Applications you can start" and "Contexts you can return to."

The **Up** and **Down Arrow** keys move the highlight bar within the lists under "Applications you can start" and "Contexts you can return to."

If the highlight bar is on the right, under "Applications you can start," pressing **Go** starts a new application. If the highlight bar is on the left, under "Contexts you can return to," pressing **Go** returns you to the highlighted context.

If you have a Mouse, moving the Mouse cursor highlights a selected area on the CM screen. If the highlight bar is on the right, clicking the **Mark Mouse** button starts a new application. If the highlight bar is on the left, clicking the **Mark Mouse** button returns you to the highlighted context.

Pressing **Cancel** removes the CM screen and returns you to the current context.

The message area, above the function key menu, tells you what you can do or explains what has gone wrong.

Pressing **Code-Help** displays the CM Configuration file name.

---

## *The Action Key*

The **Action** key is used in combination with other keys to manipulate applications. (The same key combinations are also used if you have Windows. See "Using Windows with the Keyboard" in Chapter 4.)

- |                         |  |
|-------------------------|--|
| <b>Action-Go</b>        | Pressing <b>Action-Go</b> overlays the CM screen on the current context screen.  |
| <b>Action-fn</b>        | Pressing <b>Action</b> and an assigned function key (denoted as fn) switches from one context to another, or from the CM screen to a context.      |
| <b>Action-Next</b>      | Pressing <b>Action-Next</b> repeatedly cycles through the active contexts without displaying the CM screen.  |
| <b>Action-Minus (-)</b> | Pressing <b>Action-Minus (-)</b> (the minus sign on your numeric keypad, not the hyphen) cycles through the active contexts in the opposite order. |
| <b>Action-S</b>         | Pressing <b>Action-S</b> stops all processes of the current context. Pressing <b>Action-S</b> again restarts them.                                 |
| <b>Action-Finish</b>    | Pressing <b>Action-Finish</b> eliminates a context without saving the files created.   |

## SUMMARY OF STATUS TERMS DISPLAYED

Status terms are found to the left of "Contexts you can return to" on the CM screen. The meanings of these terms are discussed further under "Status Terms," in Chapter 3. (These terms are also used with Windows.)

|         |  |
|---------|--|
| Running | This context is running.   |
| Waiting | This context (other than the Executive) is waiting for keyboard input.                                 |
| Done    | In the Executive, the last command you invoked has been completed; the Executive is waiting for input. |
| Swapped | This context has been swapped to disk. It is suspended.  |
| Stopped | This context is in background, but is not running because it is allowed to run only in foreground.     |
| Halted  | This context has been halted by the use of <b>Action-S</b> .   |

### *An Exercise with CM*

CM is designed to be self-explanatory, so you can learn your way around CM quickly by experimenting.

Do not worry about making mistakes. If you try to do something that is not allowed, a screen message tells you so, but no damage results.

*NOTE: If you have Windows installed, first complete the CM exercises below, and read Chapter 3. Then refer to Chapter 4, "Using Windows and the Mouse."*

Refer to the CM screen in Figure 2-2 and to the boxed summary in the previous section as you do the exercises below.

## STARTING CM

1. Sign on to your workstation (by completing the SignOn form), and press **Go**. Your system administrator can help you if you do not know your user name or password. (The SignOn form is explained in detail in the *Executive Manual*.)

In most cases, the first display after you sign on is the CM screen.

If you have a Mouse, notice the Mouse cursor that is now visible on your screen. (Depending on the capabilities of your system, you may notice a small block or arrow.)

2. If the first display after you sign on is the command line of the Executive, type **Install Context Manager** (or an abbreviation such as **I C M**), and press **Go** to display the CM screen. (See the *Executive Manual* for allowed abbreviations.)
3. If the first display is that of an application, press **Action-Go** to display the CM screen.

(See Chapter 3, "Basic Concepts," for more information on starting CM.)

## MOVING THE HIGHLIGHT BAR

1. If you have a Mouse, move your Mouse around so that the Mouse cursor highlights contexts listed under "Applications you can start." (If any contexts are listed under "Contexts you can return to," use the Mouse to highlight these contexts, as well.)
2. If you do not have a Mouse, experiment with the **Up** and **Down Arrow** keys to see what they do. (The **Arrow** keys are located on the upper right-hand corner of your keyboard.) If any contexts are listed under "Contexts you can return to," try the **Left** and **Right Arrow** keys, as well.

## STARTING AN APPLICATION

1. Move the Mouse cursor to highlight an application to start, and click the **Mark Mouse** button.
2. If you do not have a Mouse, use the **Up** and **Down Arrow** keys. Move the highlight bar to select an application to start, and press **Go**.

Any application that has been started is called a *context*.

If you have Windows, notice that the context may be surrounded by a highlighted border. For more information on Windows, see Chapter 4, "Using Windows and the Mouse."

## OVERLAYING THE CM SCREEN ON AN APPLICATION SCREEN; RETURNING TO THE CURRENT CONTEXT

1. Press **Action-Go**.

The CM screen is overlaid on the screen of the application you just started (the current context).

Notice the function key menu at the bottom of your CM screen (see Figure 2-2). The function key menu represents the 10 function keys, **F1** through **F10**, across the top of your keyboard. An area on the function key menu is now highlighted to reflect the application you selected in the previous step, and an abbreviation for that application appears in the menu. (For detailed information on function keys, see Chapter 3, "Basic Concepts.")

Notice the positions of the highlight bar and the bullet under "Contexts you can return to." (The bullet shows which context is current.)

2. If you have a Mouse, move the Mouse cursor between "Contexts you can return to" and "Applications you can start."

If you do not have a Mouse, use the **Left** and **Right Arrow** keys.

Notice the screen messages as you perform these actions.

3. If you have a Mouse, move the Mouse cursor to the area behind the CM screen where a portion of the current context is displayed. Click the **Mark** Mouse button to return to the current context.
4. If you do not have a Mouse, when in the CM screen, press **Cancel** to return to the current context.

## STARTING A SECOND APPLICATION

1. Press **Action-Go** again to see the CM screen.
2. Move the highlight bar with the Mouse or **Arrow** keys to the "Applications you can start" area.
3. Choose another application, and either click the Mouse **Mark** button or press **Go**.

Depending on the kind of system you have, there may be a small delay, and the message area may tell you at this point that a context is being swapped. This message is normal. (The concept of swapping is explained under "Swapping a Context to Disk" in Chapter 3.)

*NOTE: If you have Windows, when starting several applications, you may notice at some point that your video is replaced with dots. Do not be alarmed. This is normal and indicates that a context has been swapped.*

## SWITCHING CONTEXTS WITHOUT USING THE CM SCREEN

1. While holding down the **Action** key, press the function key assigned to the first application you started (the first context).

If you cannot recall which key is the assigned function key, perform the following steps:

- a. Press **Action-Go** to return to the CM screen.
- b. Refer to the function key menu to see which spaces are highlighted.
- c. Press **Go** to return to the context you just left.
- d. Hold down the **Action** key, and press the appropriate function key.

From now on, this key combination is denoted as **Action-fn**.

2. Press **Action-fn** for the second context.
3. Press **Action-Next**. (The **Next** key is on the right side of your keyboard, in the numeric pad.)

Notice that the next context appears on the screen. (When you have several contexts running, if you continue to use the **Action-Next** key, you can cycle through all your context

4. Now try **Action-Minus (-)**. (**Minus** is the key above **Next** in the numeric pad. The hyphen key does not have the same effect.)

When you have several contexts running, if you continue to use the **Action-Minus (-)** key, you can cycle through all your contexts in the reverse order.

With Windows, you can also switch contexts without returning to the CM screen by using the same action keys described above. When you press these keys, the window you switch to becomes the current window. (See "Making a Window Current," in Chapter 4.)

## STARTING MORE CONTEXTS

1. Press **Action-Go**, and start several more (perhaps three or four) applications.

With each one, note the messages displayed in the message area and the term indicated under Status. At some point, you may see a message concerning swapping, if you did not see one under "Starting a Second Application," above.

2. Now try **Action-Next** and **Action-Minus (-)** again.

*NOTE: Some applications cannot be started more than once. This will be indicated by the message*

*There is already a user named xxxx*

## FINISHING FROM WITHIN A CONTEXT

1. Press **Action-Go**.
2. Move the highlight bar with the Mouse or **Arrow** keys to choose an item under "Contexts you can return to."
3. Click the **Mark Mouse** button, or press **Go** to return to the context you selected in step 2.

With this context on the screen, use whatever method is specified to finish the application involved. (When you finish an application in this way, it is automatically saved.) In most cases, you need to press the **Finish** key. If you want to finish from the **Executive**, type **Finish Executive** (or a unique abbreviation, such as **f e**) in the command field, and press **Go**.

## ELIMINATING A CONTEXT WITHOUT SAVING

If you choose to eliminate a context without saving, follow the steps below.

1. Press **Action-Go** to return to the CM screen.
2. Move the highlight bar with the Mouse or **Arrow** keys to the first context that you started (listed under "Contexts you can return to").
3. Press **Action-Finish**.

**Caution:** *Action-Finish eliminates a context (that is selected from the CM screen) without saving your work. You should only use it for those contexts where there is nothing to save (for example, an Executive with a Done status).*

## ENDING YOUR SESSION WITH CM (LOGGING OUT)

When you have either finished your contexts from within or eliminated them without saving, you should no longer have any contexts listed under "Contexts you can return to."

**Caution:** *Ending a session without finishing all contexts first is the same as eliminating each context with Action-Finish.*

From the CM screen, perform the following steps:

1. Press **Finish**
2. Press **Go**.

Your session with CM has ended, and the SignOn form is now on your screen.

This exercise has shown you most of the ways in which you can manipulate applications using CM. There is more than one way to do some of the things you have done, as you will see in later chapters.

## *Starting an Application*

In most cases, CM is installed automatically when you sign on to your system, and the first display is the CM screen. (The term "install," as it is used here, means "to place in memory, ready to use." For details about installation, see Chapter 5, "Setting up Context Manager/VM.")

If the first display after you sign on is the command line of the Executive, type **Install Context Manager**, or an abbreviation such as **I C M**, and press **Go**. (The Executive needs only as many characters as are necessary to uniquely identify the command. If more than one word appears in the command name, each word can be abbreviated. For details on using these abbreviations in the Executive, see the *Executive Manual*.)

If the first display after you sign on is that of an application, and if you saw the messages

Installing Context Manager . . . done.

Autostarting . . .

after you signed on, your CM has been configured to autostart (start automatically) one or more applications as soon as it is loaded. You can display the CM screen by pressing Action-Go.

## *Assigning Function Keys*

Under CM, each context is assigned to a function key. You can use the function keys to switch contexts without having to see the CM screen.

### **FUNCTION KEY MENU**

Across the top of your keyboard are 10 *function keys*, **F1** through **F10**. These keys are redefined to do different things at different times and within different applications. The highlighted strip at the bottom of the CM screen is the *function key menu*, a set of temporary labels for these keys.

On your system, these labels may all be blank, or some may show abbreviations of certain application names. These labels can be configured (set up) differently for each user. (If you are setting up your own system, see "Function Key (1...10)" in Chapter 5 for details on configuration.)

If labels already appear on your function key menu, they have been preassigned for frequently used applications. This pre-assignment, which is done during configuration, saves time. Every time you use CM, these labels will appear on the same function keys.

### **FUNCTION KEYS**

When the highlight bar in the CM screen is within the list of "Applications you can start," the message area says

Select application, optionally choose function key,  
then press GO.

Select an application by using the Mouse cursor or **Arrow** keys to highlight the item you want under "Applications you can start." When you start the application, an area on the function key menu that represents the application becomes highlighted.

A function key may be assigned to an application in any of three ways:

- The key may have been preassigned by a previous user or system administrator.
- You can choose a key and assign it to a context.
- You can let the system assign a key by default.

### **Preassigned Function Keys**

If your function key menu shows an abbreviation for an application, this indicates that a function key already has been preassigned to the application you have chosen. After this application has been started, you can use this preassigned function key (in combination with **Action**) to switch to this context from any other context or from the CM screen.

### **Assigned Function Keys**

If no function key has been preassigned for the application you are starting, you can choose your own blank function key from the function key menu and assign it to this context:

1. While in the CM screen, use your Mouse cursor or **Arrow** keys to highlight the application you are going to start.
2. Decide on which blank function key you want to assign to this application.
3. Press the function key you have chosen, and then press **Go**, or highlight the area you have chosen with your Mouse cursor, and click the **Mark Mouse** button.

An abbreviation for the application now appears on that function key in the function key menu.

You do not have to choose a function key: One can be assigned to your application by default. It will be the first available blank area on the function key menu, starting from the left side of the menu.

1. Select an application from the CM screen.
2. Press **Go** or click the **Mark Mouse** button.

The screen of your chosen application has appeared and replaced the CM screen.

You can see which function key has been assigned to you by returning to your CM screen and looking at the function key menu.

If there is a preassigned function key for the application you are starting, you do not have to use it. You can assign another function key to this particular context. This feature is useful when you want to start multiple copies of the same application: two Executives, for example.

*NOTE: A function key label created within a session exists only for that session. When you finish a session (log out) and sign on again, only the preassigned labels are present.*

## *Starting Applications that Require Parameters*

You may choose to start an application, such as Floppy Copy, to which you want to supply the value of a parameter or parameters. Once you have chosen the application name from the list under "Applications you can start" and pressed **Go**, the first screen displayed is the appropriate form in the Executive:

Floppy Copy

[Number of Copies] \_\_\_\_\_

[Overwrite ok?] \_\_\_\_\_

[Dual floppy?] \_\_\_\_\_

[Suppress verification?] \_\_\_\_\_

[Device name(s)] \_\_\_\_\_

[Device password(s)] \_\_\_\_\_

After you fill in this form and press **Go**, the Floppy Copy command is executed.

## *Contexts You Can Return To*

To return to a context from the CM screen, you can do any one of four things:

1. Press **Action-fn**.
2. Use the **Mouse** or **Arrow** keys to move the highlight bar to the context you want in the list of "Contexts you can return to," and click the **Mark Mouse** button or press **Go**.
3. Select an assigned function key, and then press **Go**.

When you switch from one context to another, you do not finish the first context. It remains active, but moves from foreground to background.

4. Select an assigned function key with the **Mouse** cursor, and click the **Mark Mouse** button.

### **CYCLING THROUGH CONTEXTS**

To quickly move from one active context to another without looking at the CM screen and without having to remember which function key is assigned to which context, press **Action-Next**. Pressing this key combination has the same effect as displaying the CM screen and then pressing **Left Arrow**, **Down Arrow**, and **Go**.

You can press **Action-Next** repeatedly to cycle through all the active contexts in this way. You can move through the contexts in the opposite order by pressing **Action-Minus (-)**. (Remember, **Minus** on the numeric pad is not equivalent to the hyphen key on the top row of your typewriter pad.)

If you have **Windows**, you can get from one active window to another without looking at the CM screen in one of two ways: You can either click the **Mark Mouse** button in the selected window, or use the keyboard (as described above). For more information, see "Making a Window Current" in Chapter 4.

## HALTING A CONTEXT

If you want to stop the current context at any time (for example, if it is displaying a list or drawing a picture and you want it to stop so you can examine the partially complete display), press **Action-S**. When you do so, all processes of the current context are stopped. A stopped context is described as "Halted" under "Contexts you can return to" in the CM screen.

To restart the context, press **Action-S** again.

## THE BULLET

Notice the small bullet indicator to the left of one item in the Context list. This bullet shows which context is current. You can return to it simply by pressing **Cancel**, even if you have moved the highlight bar to another place on the screen. (This is useful if you only want to look at the CM screen to check on the status of another context and then go right back to what you were doing.)

## STATUS TERMS

The Status column under "Contexts you can return to" may show any one of six words: *Waiting*, *Running*, *Done*, *Swapped*, *Halted*, or *Stopped*.

A context that is *Waiting* is doing literally that: waiting for you to type something at the keyboard.

The status *Running* appears next to a context in which processing is continuing and where no input from you is needed at the moment.

The Executive, unlike other applications, tells CM what command has been invoked and what the status of that command is. If, for example, you give a Files command to the Executive and then press **Action-Go** to return to the CM screen, you see "Files," and not "Executive," on the list of "Contexts you can return to." When the Files command is complete, its status changes from Running to Done. (Notice that Done does not mean that the Executive itself is finished, only that it has completed the command you asked for.)

The status *Done* refers only to completed tasks in the Executive. When a long task in any other application is completed, its status is reported as Waiting.

The status *Swapped* means that the operating system has swapped that context to disk (stored it there temporarily). If you are using an 80286 Processor, a context may also be swapped to extended memory. (See the *CTOS/VM Concepts Manual* or the CTOS/VM Release Notice for more information.)

The status *Halted* indicates that you have stopped all processes of this context by pressing **Action-S**.

The status *Stopped* means that this application is in background in a partition, but CM does not allow it to continue processing there. (For example, this arrangement can prevent a program that writes directly to the screen from interfering with CM's normal operation. Such a program can run only in foreground.)

## *Swapping a Context to Disk*

### HOW SWAPPING WORKS

At some point, you may start one more application than will fit into memory. When this occurs, the operating system may have to swap out one or more contexts; that is, it may be necessary for the operating system to swap one or more contexts to disk. The message area may say

Swapping contexts . . .

There is a short pause, during which the operating system clears a space in memory for your new application. The operating system does this by storing one or more contexts temporarily in a *Swap file* on a hard disk. The screen of the new application then replaces the CM screen. (For more information on the Swap file, see "The Swap File," in Chapter 5.)

If you need to have these two contexts switch places later, there may not be room in the Swap file to swap out the new one before bringing the old one back in. If this situation in fact occurs when you try to switch these two contexts, the following message appears:

There is not enough room in the swap file to swap the context.

You must then finish one or more contexts to make room in memory for the context you want.

A context that has been swapped to disk is not actually running: It temporarily suspends its activity until it is called back by CM. But it is still a started application. You can use **Action-fn** to return immediately to the same place within it where you were working.

## CONTEXTS THAT ARE NOT SWAPPED

Two kinds of contexts are not swapped: communications applications, such as Asynchronous Terminal Emulator, or SNA RJE and real-time applications, such as direct printing in the Word Processor. (Real-time applications are applications that must respond to events within strict time constraints. If, for example, the Word Processor were swapped during direct printing, it could lose communication with the printer.) If you ask CM to do something that would require CM to swap out such a context, the following message appears:

A context in memory cannot be swapped out.

This situation can arise if, for example, you start Extended Multiplan, then start a communications application that requires Extended Multiplan to be swapped out, and then try to switch back to Extended Multiplan. Extended Multiplan must come back into the same physical place in memory where it was started, which would require swapping out the communications application. CM cannot swap Extended Multiplan in because the communications application cannot be swapped out, and the screen message appears. To get Extended Multiplan back into memory, you must finish the communications application.

In addition, you cannot run two programs that share the same communications port. For example, you cannot run simultaneously two communications programs that both need to use channel A.

CM can keep track of 10 contexts at a time. On most systems, if 10 contexts exist, some of them will be swapped to disk and suspended. Those in memory partitions will be able to run.

### **TIME SAVINGS: SWAPPING VERSUS STARTING**

Part of the benefit of CM to the user lies in the fact that swapping a context from disk to memory is much faster than starting an application. Starting an application often requires opening several files. These files are kept open when a context is swapped to disk.

## *Using the Context Manager to Cut and Paste*

You can copy text from one context to another using the Context Manager's Cut and Paste feature. Characters you select (cut) are stored in a buffer and can be copied (pasted) anywhere you like, in a different context or even within the same context.

For example, you can copy text from a mail message into a text editor document (for example, a Word Processor or Document Designer document). You can copy a file name from an Executive Files display into a parameter field of an Executive command form. You can also copy from an MS-DOS application running under the Context Manager into a CTOS application.

When characters from a cut are pasted into the new context, that context accepts the characters as keyboard input.

Once you make a cut, it is held in memory until you logout. You can paste it several times if you wish.

Sometimes CM cannot copy everything you see on the screen into a context. For example, some lines shown on the screen, the highlight bars used in forms, and the rectangles used to show function key menus cannot be copied into a text editor because they are screen attributes, not characters. The text editors cannot display screen attributes as part of a document. In this case, as much information as possible is transferred.

You also cannot cut and paste to or from the Art Designer because it is graphics-based, not character-based.

**Caution:** *You must account for the capabilities of the application into which you paste before you make the paste. Because the application will accept the pasted characters as keyboard input, you should be sure that the application is in a state where the characters can be accepted as if you typed them.*

*For example, if you paste into a Multiplan or Extended Multiplan spreadsheet in normal mode, any string of alphabetic characters is interpreted as a string of commands. Your spreadsheet could be destroyed. Use Cut and Paste with Extended Multiplan only when the spreadsheet is in Alpha mode.*

*If you paste into a text editor (for example, the Word Processor or Document Designer) when a menu is on the screen, the application interprets the pasted characters as menu choices and could change characteristics of your document that you do not wish to change.*

*If you paste a string of characters into an application that can only accept one line of characters at a time, the application may beep at you or do something else that is unpredictable.*

## THE CUT AND PASTE PROCEDURE

To use Cut and Paste:

1. Start or switch to the context from which you want to copy text. For example, start the Executive.
2. Press **Action-C**. The Cut menu appears.

---

Begin Cut-Paste operation

To dismiss form and begin selection, press **GO** or select **GO** with Mouse.

Use Mouse or **MARK, BOUND** and cursor keys to specify characters to cut.

Then switch to destination context and press **ACTION-p** to Paste.

|                     |           |        |                |
|---------------------|-----------|--------|----------------|
| Mark Mode:          | Rectangle | Stream | (Press R or S) |
| Add New Line Chars? | Yes       | No     | (Press Y or N) |

---

**Figure 3-1. The Cut Menu**

Notice that any areas on the screen that were highlighted changed their appearance. This happens so that the characters that are highlighted can still be visible after you make your selection.

3. Press **R** to choose a rectangle cut or **S** to choose a stream cut.

A rectangle cut selects only the characters and spaces that appear within the rectangle defined when you press **Mark**, then **Bound**.

A stream cut, by contrast, selects characters and blank spaces as they flow across the screen from the character indicated when you press **Mark**, to the last character in the selection, indicated when you press **Bound**. See Figure 3-2 below.

---

A stream cut would work best to copy text that runs all the way across the screen like this does. It is especially useful when the text lines up this way.

A Rectangle cut is more useful when text lines up this way or when you want to copy a specific area on the screen.

---

### Figure 3-2. Cutting Text

4. If you want line feeds added when you paste the cut, press **Y**. This will ensure that each line of the cut still starts on a new line when pasted. Otherwise press **N**.
5. Press **Go** or select **Go** on the Cut menu with the Mouse to indicate that you are ready to select the screen area to be cut.
6. Move the cursor to the first character you want included in the cut.
7. Press **Mark** on the keyboard or Mouse (left button).
8. Move the cursor to the last character you want included in the cut.
9. Press **Bound** on the keyboard or Mouse (right button).

The cut is highlighted. If you do not want the area selected, just reselect as in steps 5 through 8. Press **Cancel** if you decide not to make a cut.

10. To finalize the cut, press **Go**.

Moving to another context in any way, for example by pressing **Action-Next** or **Finish**, also finalizes the cut.

11. Switch to the context where you want to make the Paste.

12. Move the cursor to the point where you want the copied text to begin.
13. Press **Action-P**. The text from the cut is copied into the new context.

Notice that spaces have been added as place holders for any empty screen space that was included in the original cut.

## *Getting Out*

### **FINISHING A CONTEXT**

If you are working within a context, you can save your work and finish it in the same way as you normally would. When you have given the necessary commands to finish, the CM screen returns, and the message area says

Finishing . . .

Then the context is removed from the list of "Contexts you can return to."

If you are working in the Executive and want to finish it and return to CM, type **Finish Executive** (or a unique abbreviation, such as **f e**), and press **Go**.

At any time during a session, of course, you can finish a context that you no longer need.

### **ELIMINATING A CONTEXT WITHOUT SAVING**

If the CM screen is displayed, you can eliminate any context, including the Executive: Move the highlight bar to the selected context's name on the "Contexts you can return to" list, and press **Action-Finish**.

*NOTE: Action-Finish eliminates the context without saving your work from this session. You should use it only for those contexts where there is nothing to save (for example, an Executive with a Done status).*

## ENDING A SESSION (LOGGING OUT)

After you have finished all listed contexts from within the applications and the CM screen has returned, press **Finish** and then **Go**. Your session with CM is over, and the normal SignOn form returns to the screen.

Ending a session without finishing all contexts first is the same as eliminating each context with **Action-Finish**. If you press **Finish** when there are active contexts, the following screen message appears:

### WARNING

*There are active contexts, press Go to logout or Cancel to deny.*

**Caution:** *If you press Go, your work in any active contexts is not saved. Your session is ended, and you are returned to the SignOn form.*

If you have the command line of the Executive on the screen when you decide to end a session, and all other contexts are finished, you can end a session directly from the Executive by typing **Logout** in the command line and pressing **Go**. If you do this when there are active contexts, the Executive is finished, the CM screen returns, and the above warning message is displayed.

## CM Version

If you need to see what version of CM is running on your system, press **Action-Go**, if necessary, to display the CM screen, and then press the **Help** key. The following message appears:

This is Context Manager version n.n

where n.n is the version number of your CM.

## *CM Configuration File Name*

If you need to see the CM Configuration file name, press **Action-Go**, if necessary, to display the CM screen, and then press **Code-Help**.

## *Mail Reporting*

The CM screen reports that electronic mail has arrived for you in the same way that the Executive does, by displaying the messages

You have mail

Urgent mail

in the center of the top border of the CM screen.



This chapter introduces you to the interaction between CM and Windows.

CM and Windows also interact with the Mouse Services. The Mouse Services is a system service that allows you to use a Mouse to select items from the CM screen and to manipulate windows.

This chapter also gives you practice in selecting, sizing, and moving windows with either a Mouse or the keyboard. Step-by-step exercises are provided in the following sections, later in this chapter:

- Managing Windows Using the Mouse
- Managing Windows Using the Keyboard

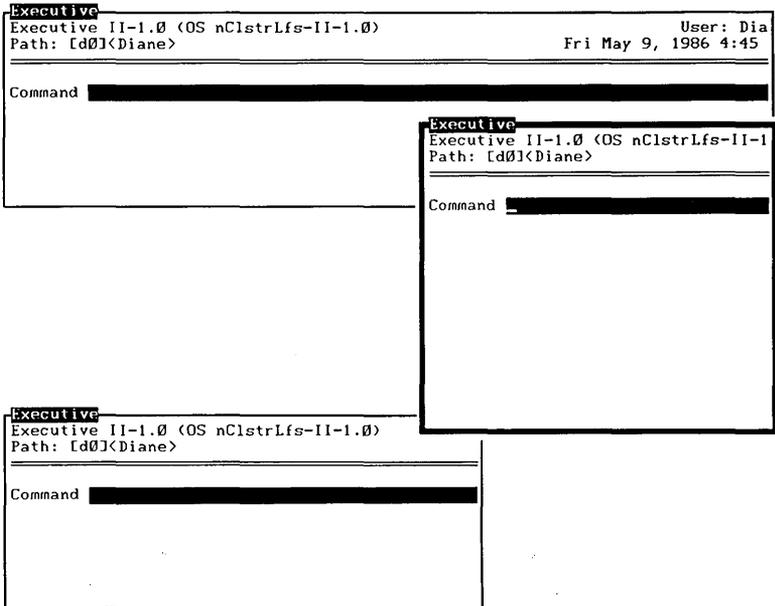
*NOTE: To use Windows with CM, you must have the Window Services installed on your workstation. To use a Mouse, you must have the Mouse Services installed. You must also be familiar with the material in Chapter 2, "Experimenting with Context Manager/VM," before you read this chapter.*

# What Is a Window?

A *window* is the rectangular portion of your screen within which an application is displayed. Your CM screen is a window. When you start an application from CM, the window of that application appears on top, and the CM screen disappears.

## Using Windows

If you have Windows installed on your workstation, your application does not have to occupy the full screen: You can now vary the size of its window. You can also view more than one application at the same time, move them to different locations on the screen, and make them overlap, if necessary. See Figure 4-1, below.



1165-003

Figure 4-1. Three Overlapping Windows

## EXAMPLE

You have just started the Copy command in the Executive and now need to telephone someone about one of your files. With Windows, the Executive can remain visible on your screen (while the Copy continues to run), and at the same time you can scan your files list in another Executive to obtain the information you need. Both applications can remain open, side by side. If you need a larger work area, you can make one Executive smaller and position it in another area on your screen. (You may want to keep it open if you plan to return to it again.)

You can also start a third application and still have the two Executives remain visibly open.

## *Starting Windows*

To start an application using Windows, use your CM screen areas as outlined in Chapter 2. Your CM screen and the way you use it have not changed.

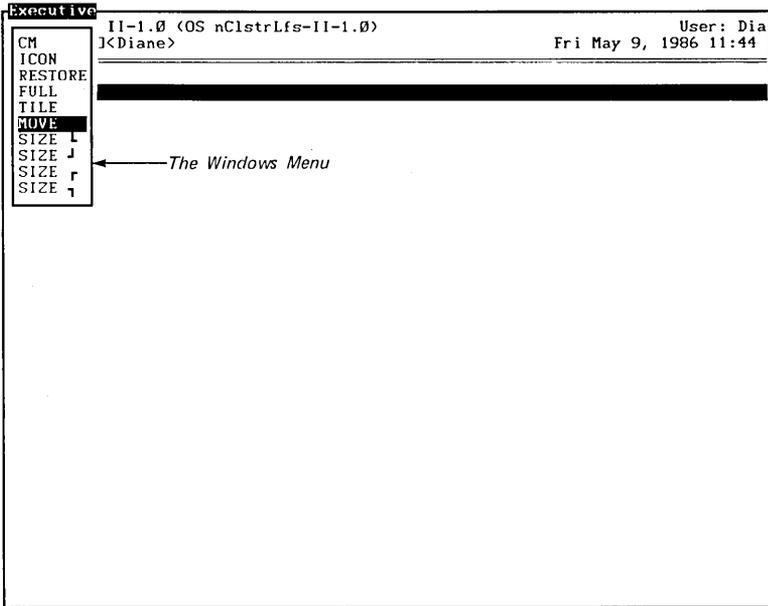
The next few pages provide detailed information and summaries about Windows and the Mouse. A tutorial that gives hands-on instructions follows.

## THE WINDOWS MENU

The *Windows menu* is a small box that "pops-up" on the screen and lists various command options. After you start an application, you can size and move its window by selecting a command from the Windows menu.

With the keyboard, you display the menu by pressing **Action-W**. (See Figure 4-2.)

Note that you cannot activate the Windows menu from the CM screen.



1165-004

**Figure 4-2. The Windows Menu**

If you have a Mouse, display the menu by placing the Mouse cursor on a window border and pressing the middle Mouse button, known as the **Menu** button. (See "Using the Mouse Buttons," below.) When you use the Mouse to display the menu, five commands are shown: **CM**, **Icon**, **Restore**, **Full**, and **Tile\***. (The **Size** and **Move** commands, shown on the menu in Figure 4-2, are not listed when you activate the Windows menu with your Mouse: You can size and move windows by using your Mouse buttons.)

Table 4-1, below, summarizes the commands available to you on the Windows menu.

---

\*On a window that must always occupy the full screen, the Windows menu displays only the **CM** and **Icon** commands. (For more information on full screen windows, see "Looking at Windows," below.)

On a window that occupies the full screen after the **Full** command is activated, the Windows menu displays only the **CM**, **Icon**, and **Restore** commands.

**Table 4-1**  
**MENU COMMANDS**  
(Page 1 of 2)

---

| Command Name    | Function   |
|-----------------|--|
| <b>CM*</b>      | Returns you to the CM screen.  |
| <b>Icon*</b>    | <p>Creates a tiny window that moves to the bottom of the screen and appears over its assigned function key.</p> <p>An icon is never the current window, so it never owns the keyboard or Mouse. It cannot be sized or moved around the screen.</p> <p>The application program represented by an icon may still be running in the background or may be swapped out.</p> |
| <b>Restore*</b> | <p>Returns the window to the size and position it was in before the <b>Full</b> command was executed.</p> <p>The <b>Restore</b> command is not displayed on the menu if the current window must always occupy a full screen. It is displayed if the window is expanded to a full screen size when the <b>Full</b> command is selected.</p>                             |
| <b>Full*</b>    | <p>Expands the window to the size of the whole screen.</p> <p>The full screen does not have window borders.</p> <p>The <b>Full</b> command is not displayed on the menu if the current window must occupy a full screen.</p>   |

---

\*If you are using a Mouse, only these commands appear on the menu.

\*\*For the **Size** commands, note that a window that occupies the full screen cannot be sized.

**Table 4-1**  
**MENU COMMANDS**  
(Page 2 of 2)

---

| <b>Command Name</b> | <b>Function</b>  |
|---------------------|--|
| <b>Tile*</b>        | <p>Divides the screen among the number of contexts that are currently active. You can view each context that is running in one "Tile" of the screen.</p> <p>The Tile command is not displayed on the menu if the current window must occupy a full screen.</p> |
| <b>Move</b>         | <p>Moves the window to a new location without changing its size. A window that occupies the full screen cannot be moved.</p>   |
| <b>Size**</b>       | <p>Selects the bottom left corner of a window. When you move this corner, the size of the window changes.</p>  |
| <b>Size**</b>       | <p>Selects the bottom right corner of a window. When you move this corner, the size of the window changes.</p>   |
| <b>Size**</b>       | <p>Selects the top left corner. When you move this corner, the size of the window changes.</p>   |
| <b>Size**</b>       | <p>Selects the top right corner. When you move this corner, the size of the window changes.</p>  |

---

\*If you are using a Mouse, only these commands appear on the menu.

\*\*For the Size commands, note that a window that occupies the full screen cannot be sized.

## *Looking at Windows*

After you select an application and press **Go**, a context appears with a highlighted window border around it. (A *window border* is a box that surrounds a window.) The border has a title bar (in reverse video) on top, displaying the name of the context.

*NOTE: A program may appear on your screen without a window border and title bar. Do not be alarmed if this occurs. Some programs require a full-screen window. You cannot move these windows or change their size. (You can only make them into tiny windows called icons.) You can place other windows on top of them that can be moved and sized.*

*When you activate the Windows menu on a window that must occupy the entire screen, only the CM and Icon commands are displayed.*

*When a program with a full-screen window is on top, it covers all other windows and takes up the whole screen. (See "Programs That Run Under Windows," in Chapter 6.)*

### **MAKING A WINDOW CURRENT**

With Windows, you can view several contexts at once; however, you can only interact with one of them at a time. The one with which you are interacting is called the *current window*.

The current window is the only window that responds to the use of the keyboard (and Mouse, if you have one) and updates its screen based on your actions. It is the only window that can be moved and sized. You can tell which window is a current window, because its border appears brighter than the other window borders on the screen.

Make a window current by clicking the **Mark Mouse** button in the selected window. (See "Using the Mouse Buttons," below.) On the keyboard, you can press **Action-Next**, **Action-Minus (-)**, or **Action-fn** (where fn represents the function key assigned to the application).

You can also make a window current by using your CM screen:

1. Display your CM screen (by selecting CM from the Windows menu or by pressing **Action-Go**).
2. Select a context from the list of "Contexts you can return to" (with your Mouse cursor or arrow keys).
3. Press **Go**.

You can return to a context without selecting an item from the CM screen: If the CM screen does not completely cover a window (that is, a part of the window is visible from behind the CM screen), you can return to this window by moving the Mouse cursor into it and clicking the **Mark** button.

When you have many contexts open, you can perform the following functions:

- If you move the Mouse cursor into a window and click the **Mark** button, that window becomes current.
- If you press **Action-Next**, the next context in the list of "Contexts you can return to" becomes the current window.
- If you press **Action-Minus (-)**, the previous context in the list of "Contexts you can return to" becomes the current window. (**Minus** is the key above **Next** in the numeric pad on the right side of the keyboard.)
- If you press **Action-fn**, the application associated with the function key becomes the current window.

(See "Cycling Through Contexts," in Chapter 3, for more information.)

## *Sizing and Moving Windows*

You can change the size of a window by stretching or shrinking its borders. The contents of the window do not change until the new size has been defined.

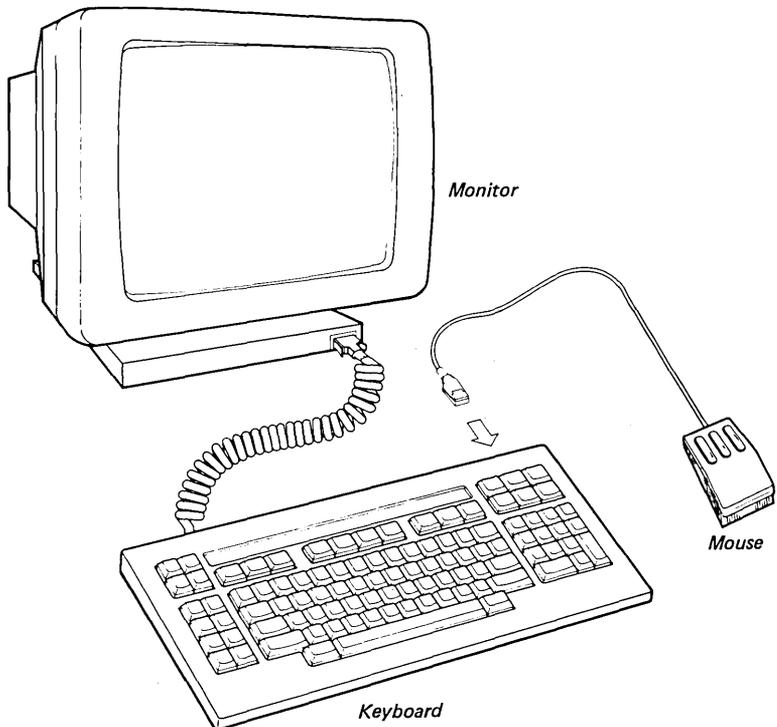
You can also change the location of a window (without changing its size) by moving its borders. The contents of the window do not move until the new position has been defined.

(See "Managing Windows Using the Mouse" and "Managing Windows Using the Keyboard," below.)

## *Using the Mouse with Windows*

### **WHAT IS A MOUSE?**

A *Mouse* is an electronic, hand-held pointing device that plugs into your keyboard and sits on your desk.



1165-005

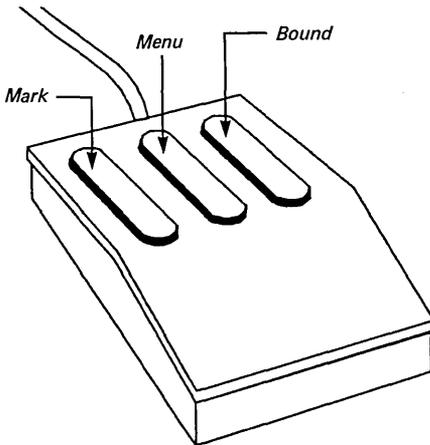
By moving the Mouse on your desktop, you move a cursor on your screen.

### USING THE MOUSE

The Mouse has three buttons: **Mark**, **Menu** (the middle button), and **Bound**. (See Figure 4-3.)

You can use the Mouse buttons, in place of your keyboard, to

- select an application from the CM screen
- display the Windows menu
- select a command from the Windows menu
- change the size of a window
- select the current window
- move a window to a new location



1165-006

**Figure 4-3. The Mouse Buttons: Mark, Menu, and Bound**

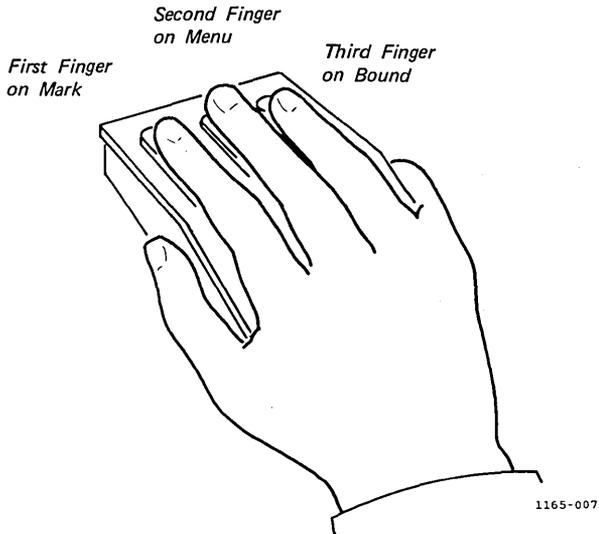
(For more information on the Mouse buttons, see Table 4-2.)

## *Moving the Mouse*

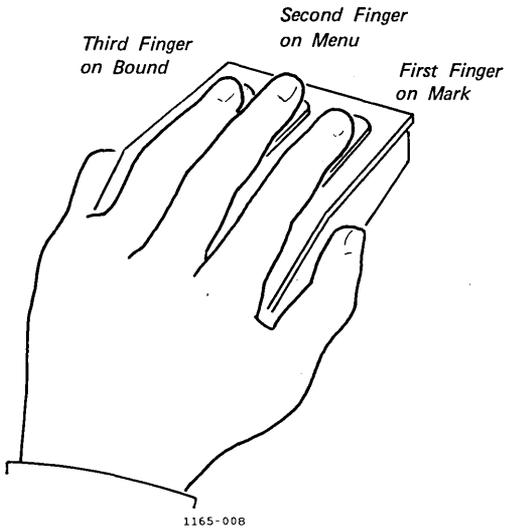
If your Mouse is not attached to your system, plug it into one of the ports on the back of your keyboard. (If you are right-handed, plug it into the right side of your keyboard. If you are left-handed, plug it into the left side.)

The Mouse is set up for right-handed people. It can also be configured for left-handed use. (See "Configuring the Mouse" in Chapter 5.)

If you are right-handed, place your first finger on the **Mark** button, your second finger on the **Menu** button (the middle button), and your third finger on the **Bound** button.



If you are left-handed, the **Mark** and **Bound** buttons are reversed.



Move the Mouse around on your desk, and watch the cursor move on the screen.

*NOTE: Note that the Mouse can be configured so that the cursor moves slowly or quickly. (See "Configuring the Mouse" in Chapter 5.)*

## USING THE MOUSE BUTTONS

You use the Mouse buttons in four ways:

### Press

When you press a button, you hold it down.

### Click

When you click a button, you press it down and release it immediately.

### Point

When you point, you move the Mouse until the cursor rests on the place you have selected.

### Drag

When you drag, you hold down a button and move the Mouse at the same time.

When you work with windows, you use all three Mouse buttons. Table 4-2, below, describes how the Mouse buttons interact with Windows.

If you want to experiment with the Mouse buttons, you can do so without damaging anything.

*NOTE: To display the Windows menu or to move or size a window, be sure that the Mouse cursor is resting on a window boundary before you press the appropriate Mouse button.*

**Table 4-2**  
**MOUSE BUTTONS**

---

| <b>Button</b> | <b>Description</b>   |
|---------------|--|
| <b>Mark</b>   | selects an application from the CM screen<br><br>selects a command from the Windows menu<br><br>makes a window current<br><br>moves a window by dragging the window border to a new position |
| <b>Bound</b>  | changes the size of a window by dragging the window border to a new position   |
| <b>Menu</b>   | displays the Windows menu  |

---

The summary in Table 4-3 is a quick reference guide to manipulating your windows using the Mouse.

**Table 4-3**  
**SUMMARY: WINDOWS AND THE MOUSE**  
(Page 1 of 2)

---

| <b>If you want to...</b>           | <b>Then...</b>   |
|------------------------------------|--|
| Activate Windows menu              | Place the Mouse cursor on a window border, and click the <b>Menu</b> (middle) button.  |
| Select command                     | Move the Mouse into the menu, point to the selection, and click the <b>Mark</b> button.  |
| Quickly change window size         | Select <b>Icon</b> , <b>Full</b> , or <b>Tile</b> from the Windows menu, and click the <b>Mark</b> button.                               |
| Return to CM screen                | Select <b>CM</b> from the Windows menu, and click the <b>Mark</b> button, or press <b>Action-Go</b> on the keyboard.                     |
| Cancel Windows menu                | Place the cursor outside the menu, and click the <b>Mark</b> button.   |
| Size windows                       | Place the Mouse cursor on a window border, press the <b>Bound</b> button, drag the border up or down, and release the button.            |
| Move windows                       | Place the Mouse cursor on a window border, press the <b>Mark</b> button, drag the border to a new position, and then release the button. |
| View "hidden" contents of a window | Press <b>Action</b> in combination with the <b>Arrow</b> keys or <b>Scroll</b> keys.   |

---

**Table 4-3**  
**SUMMARY: WINDOWS AND THE MOUSE**  
 (Page 2 of 2)

| If you want to...                          | Then...   |
|--|---|
| Make a different window the current window | Move the cursor into another window, and click the <b>Mark</b> button.  |
| Activate Windows menu on a full screen     | Place the cursor in the upper left-hand corner of the screen, and click the <b>Menu</b> (middle) button.  |
| Bring back a context from an icon          | Place the cursor in the Icon window, and click the <b>Mark</b> button,<br><br>or<br><br>Select <b>CM</b> from the Windows menu to return to the CM screen, look under "Contexts you can return to," and highlight the context that corresponds to the icon. |
| Quit from within an application            | Press <b>Finish</b> (or type <b>f e</b> , if in the Executive).   |
| Discard work without saving                | Press <b>Action-Finish</b> (from the CM screen, after selecting the context's name under "Contexts you can return to").   |
| Quit CM                                    | Press <b>Finish</b> , and then <b>Go</b> from the CM screen.  |

## *Exercises*

### MANAGING WINDOWS USING THE MOUSE

The exercises below give you practice in sizing and moving windows using the Mouse. If you do not have a Mouse, see "Managing Windows Using the Keyboard," below.

Start by opening a few Executive windows from the CM screen.

#### Opening Windows

1. Point your Mouse cursor to the **Executive** under "Applications you can start," and click the **Mark** button.

The Executive window appears, with a highlighted window border and a title bar displaying the name of the context.

2. Point your Mouse cursor to one of the window borders, and click the **Menu** (middle) button.

The Windows menu appears.

(When you want to cancel the menu, position the cursor outside the menu, and click the **Mark** button.)

3. Point to the **CM** command with your Mouse cursor, and click the **Mark** button.

The CM screen has placed its window over the application you have just started.

4. Start a second Executive from the list, "Applications you can start." (See step 1, above.)

This second Executive, which is now the current window, has overlaid its window over the first Executive. (If you start a third Executive, it would overlay its windows over the first two Executives, and so on.)

## Sizing Windows

You can change a window size by dragging the border of the current window to a new position. You can also drag a corner of the border. When the corner is dragged, the window changes size in two dimensions at the same time.

1. Move the Mouse so the cursor points to the top window border.
2. Press the **Bound** button (the border dims), and at the same time move the Mouse down (the border becomes highlighted again). This is called "dragging" the Mouse.

As the Mouse moves down, the top border of the window moves down, and the window gets smaller.

Notice that the border moves in one direction.

3. Release the button when the window has reached the size you want.

The contents of the window have been redrawn.

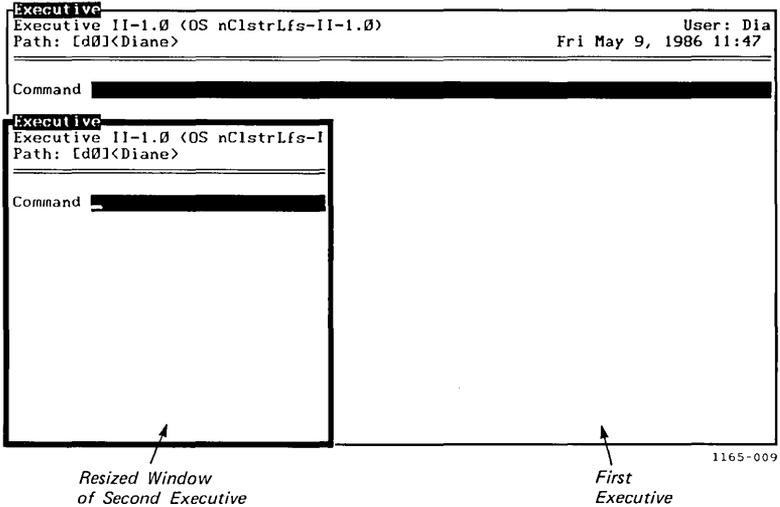
Notice that the first Executive, which was underneath, appears. Since the first Executive is not the current window, its borders are not highlighted.

4. Place the Mouse cursor on other highlighted window borders, and practice dragging these borders (using the **Bound** button) to make the window bigger or smaller. (For an example of a resized window, see Figure 4-4.)
5. Point to any corner on the window border of the current window.
6. Press the **Bound** button, and, at the same time, move the Mouse.

Notice how the border moves in two directions.

7. Release the button when the window has reached the size you want.

The contents of the window have been redrawn.



**Figure 4-4. A Resized Window**

8. Now make the other window current by moving the Mouse and placing the cursor in this other window.
9. Click the **Mark** button.

The other window has now become the current one. The current window has bright borders.

---

**How can I see a window that is behind the current window?**

Use the **Bound** button to make the current window smaller. (See steps 5 to 7.) When you release the button, the hidden window is revealed.

---

Continue sizing the two windows by moving the borders and corners.

*NOTE: When you make a window smaller, the contents of the entire screen are not always visible in the smaller window (for example, the command line in the Executive may not show up in the smaller window). This can occur with certain programs that are not aware that their sizes have changed. (See "Programs That Run Under Windows" in Chapter 6.)*

*You can view the "hidden" contents of the resized window by using your Action key in combination with your ARROW keys and Scroll keys. When you use these key combinations, the contents of the window that were not visible are moved into view. See Table 4-4, below.*

## Moving Windows

During the **Move** operation, only the borders of the current window move; the contents of the window do not move until the new position has been defined. To move a window,

1. Move the Mouse so that it points to a border on the current window.
2. Press the **Mark** button, and drag the border to a new position.
3. Release the button when the window is in the location you want.

The contents of the window have been redrawn.

**Table 4-4**  
**MAKING HIDDEN WINDOW CONTENTS VISIBLE**

---

| <b>If you want to...</b>                           | <b>Then...</b>  |
|--|---|
| See the previous line within the window            | Press <b>Action-Up Arrow</b> to move the window contents up one line at a time.                 |
| See the next line within the window                | Press <b>Action-Down Arrow</b> to move the window contents down one line at a time.             |
| See the characters on the right side of the window | Press <b>Action-Left Arrow</b> to move the window contents to the left one character at a time. |
| See the characters on the left side of the window  | Press <b>Action-Right Arrow</b> to move the contents to the right one character at a time.      |
| Move quickly to the window contents at the top     | Press <b>Action-Scroll Down</b> .   |
| Move quickly to the window contents at the bottom  | Press <b>Action-Scroll Up</b> .   |

---

*NOTE: To quit working at any time, follow the procedures you normally would when in CM (refer to the summary in Table 4-1, "Menu Commands," above, or see "Getting out," in Chapter 3). To continue, keep both Executives open on your screen, and follow the next exercise.*

## ACTIVATING COMMANDS ON THE WINDOWS MENU

When you use a Mouse to activate the Windows menu, the following five commands are displayed: **CM**, **Icon**, **Restore**, **Full**, and **Tile**.

When you activate the Windows menu on a context that must occupy the entire screen and cannot be sized or moved, only the **CM** and **Icon** commands are displayed. On a window that occupies the entire screen after the **Full** command is activated, the Windows menu displays the **CM**, **Icon**, and **Restore** commands.

The **CM** command returns you to the CM screen. (**Action-Go**, on the keyboard, performs this same function.)

### Using the Tile Command

The **Tile** command allows you to view all the windows you have opened. When you execute this command, each window is positioned in one "tile" of the screen.

1. Move your Mouse to make the cursor point to any border area of the current window.
2. Click the **Menu** (middle) button to make the Windows menu appear.

(Remember, when you want to cancel a menu, position the cursor outside of the menu, and click the **Mark** button.)

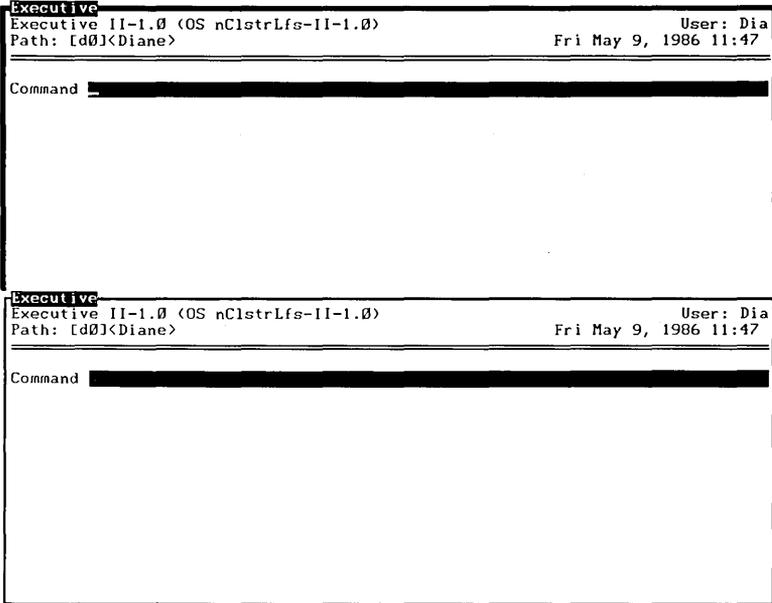
3. Select the **Tile** command by moving your Mouse into the menu and pointing to the command.

The selected command appears in reverse video.

4. Click the **Mark** button to execute the command.

The menu has disappeared, and each **Executive** appears in one tile of the screen. (See Figure 4-5.)

The current window has the brighter border.



1165-010

**Figure 4-5. Two Tiled Windows**

5. Move the Mouse, and place the cursor in the other window (the one that does not have bright borders).

6. Click the **Mark** button.

The other window has now become the current one.

7. Activate the **Windows** menu by clicking the **Menu** (middle) button on a window border.

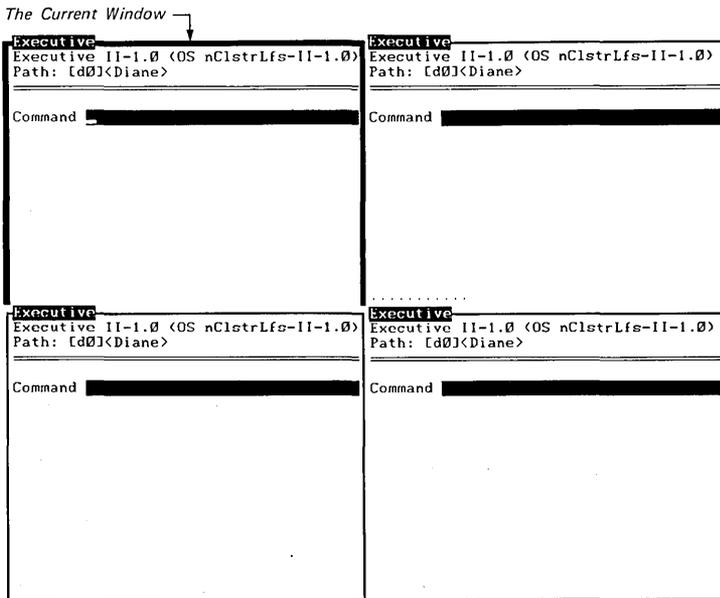
8. Select **CM**, and click **Mark** to return to the **CM** screen.
9. Start a third **Executive**. Then repeat steps 7 and 8 to start a fourth **Executive**.
10. Activate the **Windows** menu, and select the **Tile** command again.

You now see four **Executives** represented on the screen. The last one that you started is the current window. (See Figure 4-6.)

11. Experiment with the **Tile** command to open additional **Executive** windows.

Notice how the tile format changes when more windows are displayed on the screen.

Remember to use **Action** in combination with the **Arrow** keys and **Scroll** keys to make the hidden contents visible.



1165-011

**Figure 4-6. Four Tiled Windows**

---

### **What if a window appears with dotted lines?**

This indicates that the context has been swapped. You can recall it by placing the Mouse cursor in its window and clicking the **Mark** button.

---

Practice making the other windows current. (Place the Mouse cursor in a different window, and click the **Mark** button.)

Experiment also with **Action-Next**, **Action-Minus (-)**, and **Action-fn** to make different windows current.

### **Using the Icon Command**

To make a window iconic, you select the **Icon** command from the Windows menu.

The **Icon** command shrinks the windows that are not presently required: You may want only one Executive open. Perhaps you do not need the other three at this moment and would like a larger work space on your screen. (You may be planning to return to these other windows later.)

When you execute the **Icon** command, the selected window shrinks and moves to the bottom of the screen. Its title bar shortens. The contents of this window have now been replaced with the window's associated function key. (See Figure 4-7.) Notice that this function key is represented on the screen in relation to its position on the keyboard. Since it is still represented on the screen, it can remind you that you still have a context that is running.

An icon is never the current window. It cannot be sized or moved around the screen.

You can bring an icon back to its previous size by placing the Mouse cursor in the icon window and clicking the **Mark** button.

You can also bring back the icon by returning to the CM screen, looking under "Contexts you can return to," and selecting the context that corresponds to the icon.

To create some iconic windows, follow the steps below:

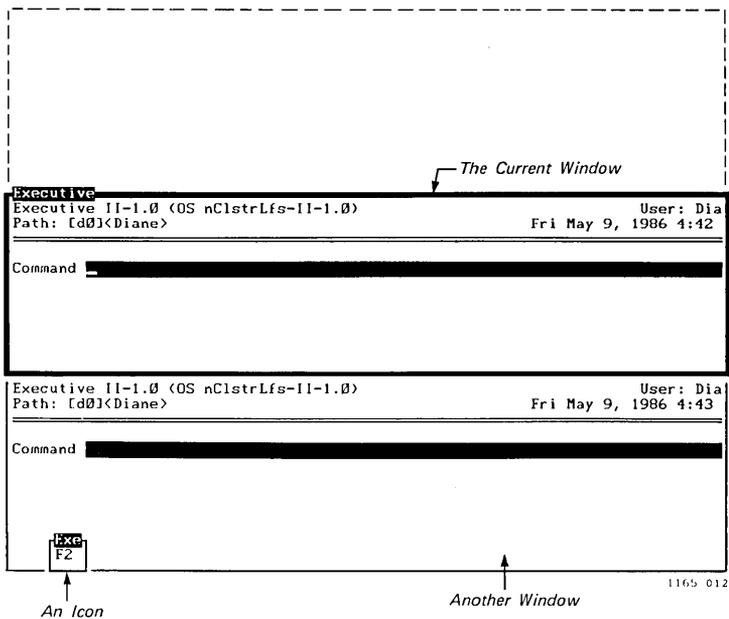
1. Make three Executives active. You may have to finish any additional Executives (by entering **f e** in the command line of the current window). Remember, you can also discard your work by highlighting the selected context from the CM screen and pressing **Action-Finish**.
2. Activate the Windows menu (by pointing anywhere on the bright window border and clicking the **Menu** button), and select the **Tile** command.

Your screen should display three "tiled" Executives.

3. Make the window on top the current one (if it does not already appear that way).
4. Activate the Windows menu again, point to the **Icon** command, and click the **Mark** button.

The current window has moved to the bottom of the screen and has become iconic. Another window is now the current window. See Figure 4-7.

5. Follow step 4 again to create another icon.



**Figure 4-7. A Current Window and an Icon**

---

### What if my icons are not visible?

If the current window has borders that are at the bottom of the screen, you cannot see the icons. They are hidden behind the current window. Follow the steps below to make the icons visible.

To uncover hidden icons,

1. Point to the bottom border of the current window border.
2. Press the **Bound** button, and drag up the border.

When you release the button, you can see the two icons that were previously covered.

---

To bring your icons back to their previous window size, you can do one of the following:

- Place the Mouse cursor in the icon window, and press the **Mark** button.
- Return to the CM screen (by selecting **CM** from your Windows menu), look under "Contexts you can return to," and highlight and select the context that corresponds to the icon.

*NOTE: The icon that is brought back last becomes the current window. It places itself over the other windows.*

### Using the Full Command and the Restore Command

The **Full** command creates a window that takes up the full screen. It does not have a border or a title bar.

To activate the Windows menu on a full screen, move the Mouse cursor to the upper left-hand corner, and click the **Mark** button. The commands displayed on the menu are **CM**, **Icon**, and **Restore**.

*NOTE: Once a context becomes full-screen, you cannot move it or change its size. However, you can make it iconic by selecting the Icon command from the Windows menu.*

The **Restore** command returns the window to the size and position it was in before the **Full** command was executed.

Continue to keep three Executives open. (If you have left the exercises and are returning, start three Executives from your CM screen.)

1. When in an Executive window, activate the Windows menu and select the **Full** command.
2. Click the **Mark** button to execute the command.

Notice that the context has taken up the whole screen. It does not have a border or a title bar. (The other Executive screens are in the background.)

3. Move the Mouse so that the cursor is in the upper left-hand corner of your screen. (Make sure it is exactly in the corner and cannot move up anymore.)
4. Click the **Menu** (middle) Mouse button to make the Windows menu appear.

There are three commands on the menu: **CM**, **Icon**, and **Restore**.

5. Select the **Restore** command by clicking the **Mark** button.

The window appears with its border and title bar.

It should be the size it was before the **Full** command was executed. (For example, if your screen displayed three tiled windows before you executed the **Full** command, these three tiles would reappear if you selected **Restore**.)

You now have a flexible way of arranging your screen. Later, when you apply these techniques to your own applications, you will be able to work quickly and efficiently.

## Using Windows with the Keyboard

If you do not have a Mouse, you can use the keyboard to select, size, and move windows.

The summary in Table 4-5 is a quick reference guide to manipulating your windows using the keyboard.

**Table 4-5**  
**SUMMARY: WINDOWS AND THE KEYBOARD**  
(Page 1 of 2)

| If you want to...          | Then...   |
|----------------------------|---|
| Activate Windows menu      | Press <b>Action-W</b> .   |
| Select command             | Press the <b>Arrow</b> keys, or use the <b>Return</b> or <b>Next</b> key.   |
| Size windows               | Press <b>Action-W</b> , and then press the <b>Arrow</b> keys or the <b>Return</b> or <b>Next</b> key to select a <b>Size</b> command. Press <b>Go</b> , press the <b>Up</b> or <b>Down Arrow</b> keys, and press <b>Go</b> again. |
| Quickly change window size | Select <b>Tile</b> , <b>Icon</b> , or <b>Full</b> from the Windows menu, and press <b>Go</b> .  |
| Return to CM screen        | Select <b>CM</b> from the Windows menu, or press <b>Action-Go</b> .   |
| Move windows               | Press <b>Action-W</b> , and press the <b>Arrow</b> keys or the <b>Return</b> or <b>Next</b> key to select the <b>Move</b> command. Press <b>Go</b> , press the <b>Up</b> or <b>Down Arrow</b> keys, and press <b>Go</b> again.    |
| Cancel Windows menu        | Press <b>Cancel</b> .   |

**Table 4-5**  
**SUMMARY: WINDOWS AND THE KEYBOARD**  
(Page 2 of 2)

---

| <b>If you want to...</b>                   | <b>Then...</b>  |
|--|---|
| View "hidden" window contents              | Press <b>Action</b> in combination with the <b>Arrow</b> keys or <b>Scroll</b> keys.  |
| Make a different window the current window | Press <b>Action-Next</b> or <b>Action-fn</b> .  |
| Bring back a context from an icon          | Press <b>Action-Next</b> , <b>Action-fn</b> , or look at "Contexts you can return to," and select the context that corresponds to the icon. |
| Quit from within an application            | Press <b>Finish</b> (or type <b>f e</b> if in the Executive).   |
| Discard work without saving                | Press <b>Action-Finish</b> from the CM screen (after selecting the context's name under "Contexts you can return to").                      |
| Quit CM                                    | Press <b>Finish</b> and then <b>Go</b> from the CM screen.  |

---

## *Exercises*

### **MANAGING WINDOWS USING THE KEYBOARD**

The exercises below give you practice in sizing and moving windows with your keyboard.

Start by opening a few Executives from the CM screen and activating the Windows menu.

## Opening Windows—Activating the Windows Menu

To open two Executives from the CM screen, perform the following steps:

1. Use your **Arrow** keys to highlight **Executive** under "Applications you can start," and press **Go**. (If **Executive** is already highlighted, press **Go**.)

The Executive window appears with a highlighted window border and a title bar displaying the name of the context.

2. Press **Action-Go** (to return to the CM screen), and start a second Executive from the list, "Applications you can start." (You can also return to the CM screen by selecting CM from the Windows menu.)

This second Executive, which is now the current window, has overlaid its window over the first Executive. (If you start a third Executive, it would overlay its windows over the first two Executives, and so on.)

3. Press **Action-W** to bring up the Windows menu.

The rectangular menu appears in the upper left-hand corner of the current window. The **Move** command has automatically been selected. It appears in reverse video.

The Windows menu displays the commands described in Table 4-1, above. You select a command by either pressing the **Return** or **Next** key or by pressing the **Arrow** keys. (The **Arrow** keys are located in the upper right-hand corner of your keyboard.)

To cancel the menu, press **Cancel**. (Note that you must cancel the menu first if you use **Action-Go** to return to the CM screen.)

### Sizing a Window

You can size a window by using your **Arrow** keys. The **Arrow** keys move the corner of a window up, down, left, or right. As the **Arrow** keys are moved, the borders next to the selected corner move, changing the size of the window.

1. Press the **Down Arrow** key on the Windows menu to select **Size** (bottom left corner).

(Remember, you can also use your **Return** or **Next** key to select the command.)

The selected command appears in reverse video.

2. Press **Go** to execute the command.

Notice that the window border has lost its brightness, and the Windows menu has disappeared.

3. Press the **Up Arrow** key to move the window border.

The window border is highlighted again, and the selected corner and its adjacent borders move. The other two sides remain stationary.

4. Experiment by moving the other **Arrow** keys right, left, and down.

Notice how the selected corner and its borders move.

5. When the window is the size you want, press **Go**.

The window is redrawn in the new size.

Notice that the first Executive, which was underneath, appears. Since the first Executive is not the current window, its borders are not highlighted.

*NOTE: When you make a window smaller, the contents of the entire screen are not always visible in the smaller window. This can occur with certain programs that are not aware that their sizes have changed. (See "Programs That Run Under Windows" in Chapter 6.)*

*You can view the "hidden" contents of the resized window by using your Action key in combination with the Arrow keys and Scroll keys. When you use these key combinations, the contents of the window that were not visible are moved into view. See Table 4-4, above.*

6. Now press **Action-Next**.

CM selects the next context (in this case, the other Executive) as the current window. This window is brought to the top and has the brighter border.

When you have several contexts running, you can cycle through all your contexts by pressing **Action-Next**. The "next" context always becomes the current window.

Pressing **Action** and the function key assigned to the other application also selects the next context as the current window. This key combination is denoted as **Action-fn**. (If you cannot recall which key is the assigned function key, press **Action-Go**, or select CM from the Windows menu to return to the CM screen. Refer to the function key menu at the bottom of the screen, and press **Go** to return to your context.)

7. Press **Action-W** to bring back the menu, and experiment on your own by selecting **Size** **↓**, **Size** **↖**, or **Size** **↗**. (For an example of a resized window, see Figure 4-8.)

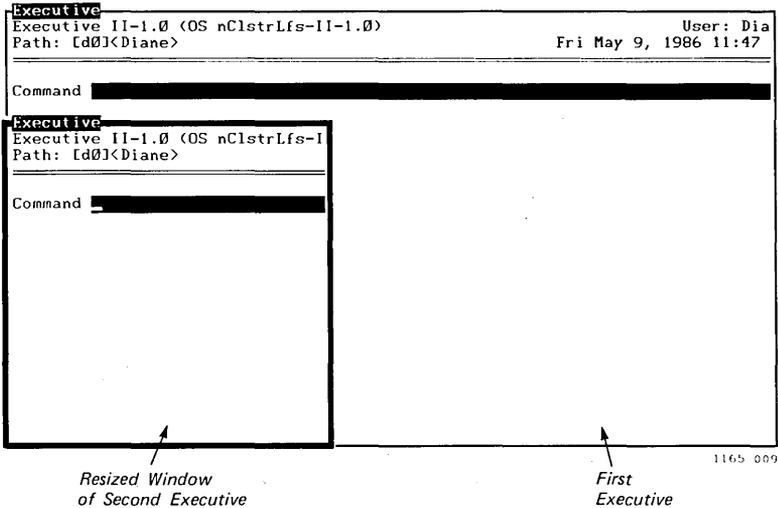


Figure 4-8. A Resized Window

*NOTE: To quit working at any time, follow the procedures you normally would when in CM. (Refer to Table 4-5 above, or see "Getting out" in Chapter 3.) To continue, keep both Executives open on your screen, and follow the next exercise.*

## **Moving Windows**

You can practice moving a window after you have changed its size. (The window must be smaller than the screen in order to be moved around.)

When you execute the **Move** command, only the borders of the window move. The contents of the window move after you press **Go** to select the position you want. The window is then redrawn in this new location. To move a window,

1. Start two Executives, if you have not already done so.
2. Press **Action-W** to display the Windows menu, and notice that the **Move** command is automatically selected. (Remember, you may have to use the **Size** commands to resize the Executive before you can move it.)
3. Press **Go**.
4. Use the **Arrow** keys to move the window up, down, left, or right.
5. Press **Go** again to redraw the window in the location you have selected.

## **Using the Tile Command**

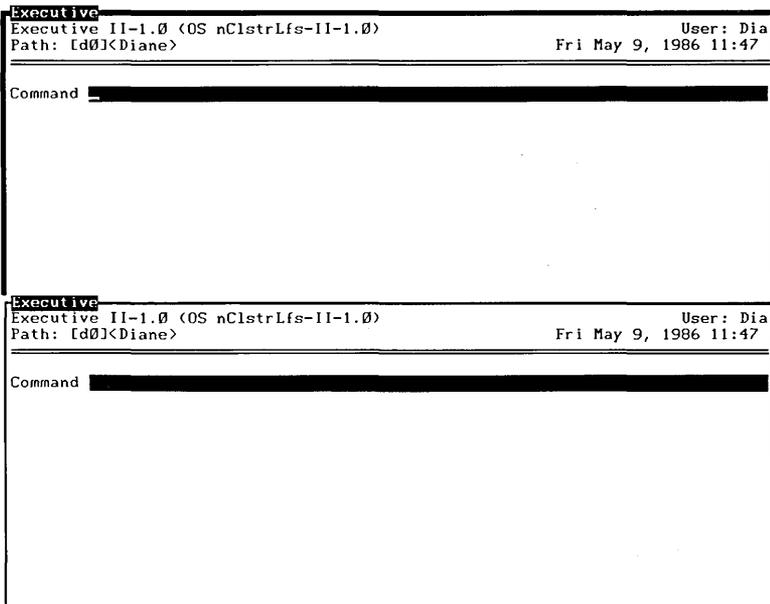
The **Tile** command allows you to view all the windows you have opened. When you execute this command, each window is positioned in one tile of the screen.

While you still have two Executives open, perform the following steps:

1. Activate the Windows menu by pressing **Action-W**.
2. Select the **Tile** command by either pressing **Return** or **Next**, or by pressing **Up Arrow**.
3. Press **Go**.

You see two Executives evenly represented on the screen. (See Figure 4-9.)

(The current window has highlighted borders. You can make the next window current by pressing **Action-Next** or **Action-fn**.)

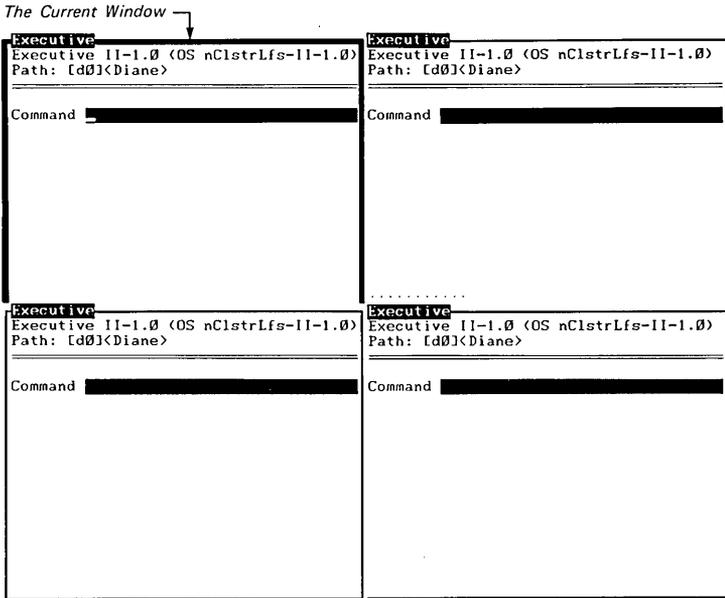


1165-010

**Figure 4-9. Two Tiled Windows**

4. Press **Action-Go**, or select **CM** from the **Windows** menu, and start a third Executive and then a fourth Executive.
5. Activate the **Windows** menu, and select the **Tile** command again.
6. Press **Go**.

You now see four Executives represented on the screen. The last one that you started is the current window. (See Figure 4-10.)



1165-011

**Figure 4-10. Four Tiled Windows**

7. Experiment with the **Tile** command to open additional Executive windows.

Notice how the tile format changes when more windows are displayed on the screen.

---

#### **What if a window appears with dotted lines?**

This indicates that the context has been swapped. It can be recalled by pressing **Action-Next**, **Action-Minus (-)**, or **Action-fn**.

---

Experiment with **Action-Next**, **Action-Minus (-)**, and **Action-fn** to make different windows current.

#### **Using the Icon Command**

To make a window iconic, select the **Icon** command from the Windows menu.

The **Icon** command shrinks the windows that are not presently required. You may want only one Executive open. Perhaps you do not need the other three at this moment and would like a larger work space on your screen. (You may be planning to return to these other windows later.)

When you execute the **Icon** command, the selected window shrinks and moves to the bottom of the screen. Its title bar shortens. The contents of this window have now been replaced with the window's associated function key. (See Figure 4-11.) Notice that this function key is represented on the screen in relation to its position on the keyboard. Since it is still represented on the screen, it can remind you that you still have a context that is running.

An icon is never the current window. It cannot be sized or moved around the screen.

You can bring an icon back to its previous size by pressing **Action-Next** or **Action-fn.** (Fn, the assigned function key, is visible on the icon.)

You can also bring back the icon by returning to your CM screen, pressing **Action-Go**, looking under "Contexts you can return to," and selecting the context that corresponds to that icon.

To create some iconic windows, follow the steps below.

1. Make three Executives active. You may have to finish any additional Executives (by entering **f e** in the command line). Remember, you can also discard your work by highlighting the selected context from the CM screen and pressing **Action-Finish**.

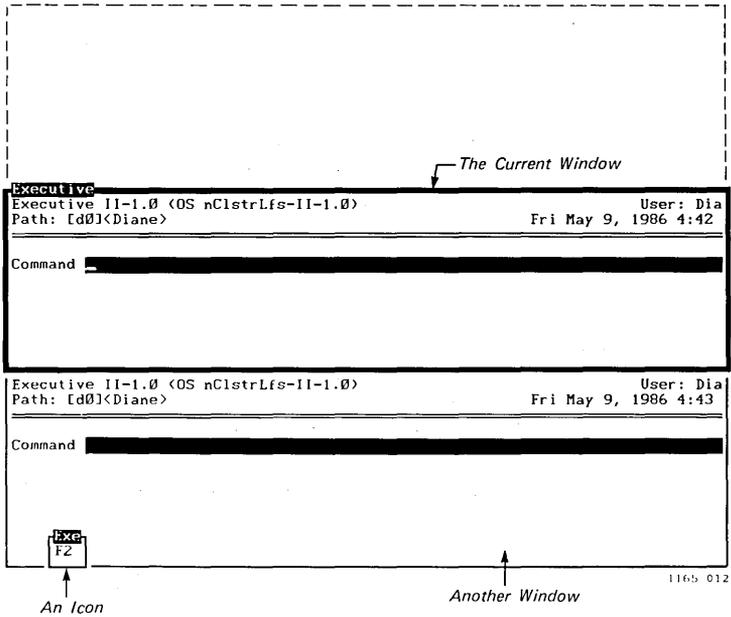
2. Activate your Windows menu, and select the **Tile** command.

Your screen should display three "tiled" Executives.

3. Make the window on top the current one (if it does not already appear that way).
4. Activate the Windows menu again, and select the **Icon** command.
5. Press **Go** to activate the command.

The current window has moved to the bottom of the screen and has become iconic. Another window is now the current window. See Figure 4-11.

6. Repeat steps 4 and 5 to create another icon.



**Figure 4-11. A Current Window and an Icon**

### What if my icons are not visible?

If the current window has borders that are at the bottom of the screen, you cannot see the icons. They are hidden behind the current window. Follow the steps below to make the icons visible.

To uncover hidden icons,

1. Activate the Windows menu.
2. Select **Move** and press **Go**.
3. Use the **Up Arrow** key to move the window.
4. Press **Go** to redraw the current window in a new location.

Now you can see the two icons that were previously covered.

To bring your icons back to their original window size, press **Action-Next** or **Action-fn**. (You can also return to your CM screen. Press **Action-Go**, look under "Contexts you can return to," and select the context that corresponds to that icon.)

*NOTE: The icon that is brought back last becomes the current window. It places itself over the other windows. Press Action-Next to make another window current.*

### **Using the Full Command and the Restore Command**

The **Full** command creates a window that takes up the full screen. It does not have a border or a title bar.

*NOTE: Once a context becomes full-screen, you cannot move it or change its size. However, you can make it into an icon by selecting Icon from the Windows menu.*

When you press **Action-W** on a full screen, three commands appear on the windows menu: **CM**, **Icon**, and **Restore**. (On a context that must always run in a full screen, only the **CM** and **Icon** commands are displayed.)

The **Restore** command returns the window to the size and position it was in before the **Full** command was executed.

Continue to keep three Executives open. (If you have left the exercises and are returning, start three Executives from your CM screen.)

1. When in an Executive window, activate the Windows menu, and execute the **Full** command.

Notice that the context has taken up the whole screen. It does not have a border or a title bar. (The other Executive screens are in the background.)

2. Press **Action-W**.

There are three commands on the menu: **CM**, **Icon**, and **Restore**.

3. Select the **Restore** command.

The window appears with its border and title bar.

It should be the size it was before the **Full** command was executed. (For example, if your screen displayed three tiled windows before you executed the **Full** command, these three tiles would reappear if you selected **Restore**.)

You now have a flexible way of arranging your screen. Later, when you apply these techniques to your own applications, you will be able to work quickly and efficiently.

## *Overview*

This chapter describes the steps you should follow to prepare for and configure CM.

Configuring CM involves

- editing a User Configuration file to start CM automatically at signon
- using the CM Configuration File Editor to create the CM Configuration file or files

The CM Configuration File Editor is a related program that greatly simplifies system setup.

You can configure CM differently for each user, if you want to. Note that you do not need the various application Run files in order to do this configuration.

For memory and operating system requirements and instructions for placing the software on the system, see the Context Manager/VM Release Notice.

## THE SWAP FILE

All swapping is handled by the operating system. The operating system can swap to a file that it has designated on a local hard disk, or it can swap over the cluster to the hard disk attached to the master workstation. On an 80286 or 80386 Processor, applications can be swapped into extended memory above one megabyte.

The Swap file should be contiguous (not broken up among different sectors on the disk) so that swapping can be done as rapidly as possible.

For more information on the Swap file, and swapping to extended memory, see the *CTOS/VM Concepts Manual* and the CTOS/VM Release Notice.

### Creating the Swap File

The operating system creates a Swap file by default when it is needed. However, if you want, you can name and/or size such files yourself.

To create the Swap file, you must edit the file [Sys]<Sys>OSconfig.sys. In this file, the following two entries concern you:

| Entry Identifier | Default                 |
|------------------|-------------------------|
| :SwapFileSize:   | 2048                    |
| :SwapFileName:   | [sys]<sys>CrashDump.sys |

The operating system reads [Sys]<Sys>OSconfig.sys to determine the Swap file size. If you have not specified a size, the size by default is 2048 sectors.

The operating system then looks at what is specified as the Swap file name. If you have not entered a name, the name by default is [Sys]<Sys>CrashDump.sys. (See "Using CrashDump.sys as Your Swap File" in Chapter 6.)

If the file indicated in the SwapFileName field cannot be accessed or created, the operating system searches for and creates, if necessary, [Sys]<Sys>swaparea.nn (nn represents a value of 00, which can be incremented to 01 or 02, and so on).

(For general information on swapping, see "Swapping a Context to Disk" in Chapter 3.)

## CM AND INSTALLED SYSTEM SERVICES

CM is not an installed system service. The user can log out of CM and continue to run the system without rebooting. System services are described in the *CTOS/VM Concepts Manual*.

*NOTE: All system services must be installed before loading CM. Commands such as Install Spooler are not permitted after CM has been installed.*

If you try to install a system service after loading CM, status code 206 (invalid user number) is returned.

In addition to the regular user name or names, you should define an administrative user name, such as **Exec.user** or **NoCM.user** on every system. This user should not be configured to install CM at signon, but should load the Executive. Such a user name allows you access to the system for system service installation or troubleshooting.

## INSTALLATION AND DEINSTALLATION

CM can be installed (loaded from disk to memory) automatically after signon or manually through the Executive command, **Install Context Manager** (or a unique abbreviation, such as **ICM**). Installation at signon is specified in the User Configuration file.

While running, CM manipulates applications in response to **Action** keystrokes in combination with other keystrokes. (See Chapter 3, "Basic Concepts," for details about commands. For a general description of how CM works, see Chapter 1, "Before You Begin.")

CM is deinstalled when the user logs out.

For information on the files required by CM, refer to your Release Notice.

## *CM Version Number*

When the Context Manager screen is displayed, you can press the **Help** key to display the following message:

```
This is Context Manager version n.n
```

where n.n is the version number of CM you are running.

## *Editing the User Configuration File*

Most users prefer to have CM installed when they sign on. In such a case, each User Configuration file (other than that of the administrative user) should contain the following lines:

```
:SignOnExitFile:[Sys]<Sys>SignOn.run  
:SignOnChainFile:[Sys]<Sys>CmInstall.run  
'Install Context Manager'  
[Sys]<Sys>ConfigFileName
```

where ConfigFileName is the default (CmConfig.sys) or the file name you define for this user when you use the CM Configuration File Editor. (See "Using the CM Configuration File Editor," below.)

You can edit the User Configuration file directly, or you can use the User Configuration File Editor to do so. (See the *Executive Manual*.)

Note that this type of installation allows CM to run in an environment that may not have the Executive.

If you prefer not to have CM installed when the user signs on, the user must install it from the command line of the Executive by giving the command **Install Context Manager** (or a unique abbreviation, such as **I C M**) and pressing **Go**.

It is a good practice to create on each system one administrative user name that does not automatically install CM when you sign on. You can use this name to gain access to the Executive for system service installation or troubleshooting.

## CONFIGURING THE MOUSE

You must edit your User Configuration file to configure the Mouse for left-handed use.

To do this, modify (or insert) the following line:

```
:LeftHanded:Y
```

If you do not insert **Y**, the default is **N** (meaning right-handed).

You can also edit this file to specify how quickly you want the Mouse cursor to move.

To do this, modify (or insert) the following line:

```
:MouseSpeed:1..10
```

The default value is zero. (That is, the system automatically assigns zero if you do not specify a number.) The cursor moves faster as you specify a higher number, up through 10. Note that raising the Mouse speed makes the Mouse more responsive to your hand movements and, consequently, somewhat harder to control.

You can insert the Mouse configuration information at any point in your User Configuration file. Capitalization does not matter, but you should spell out phrases completely.

## *A Note about the Number of Contexts*

Should you create a great many contexts, you may at some point receive this screen message:

This version of the OS cannot support any  
more contexts.

For more information, see "Allowing for More Contexts," below.

## *Using the CM Configuration File Editor*

The CM Configuration File Editor is not part of CM. It is a separate program that allows you to supply information about how each system is to be set up. This information includes details about the applications that you want a particular CM to be able to run.

All this information is placed in a CM Configuration file. When CM is installed, it refers to this file for specific details on how it is expected to function.

The CM Configuration file is not the same as the User Configuration file. The User Configuration file is more general in scope. (See "User Configuration," under "Advanced Concepts," in the *Executive Manual*.)

## ENTERING FROM THE EXECUTIVE

To start the CM Configuration File Editor from the Executive, use the command **CM Config File Editor**, or a unique abbreviation, such as **c c f e**.

```
CM Config File Editor  
[Config file name]
```

---

It is preferable to enter the name of the Configuration file at this point (if you do not want the default), although you can provide it within the CM Configuration File Editor (hereafter called the CM Editor).

The default file name is **[Sys]<Sys>CmConfig.sys**. If you press **Go** without entering a file name in the above command, the next display shows you the default file name and gives you the opportunity to accept or change it. The default file name appears in a highlighted line. To accept this default name, press **Go**. To define a new file name, edit within this line as you want, and then press **Go**.

If you enter a CM Configuration file name that does not exist and press **Go**, the message line informs you that the file name does not exist and asks for confirmation that you want to create this file. Press **Go** to confirm that you do, or **Cancel** to discard it.

If not all users are to have the same configuration, use the CM Editor several times to define the various configurations you want, and give distinct names to these Configuration files.

Each user's own User Configuration file must contain the name of that user's CM Configuration file. Enter the CM Configuration file name when editing the User Configuration file. (See "Editing the User Configuration File," above.)

## EXPERIMENTING WITH THE CM EDITOR

As with CM itself, it is not difficult to learn your way around the CM Editor by experimentation. The software is well-protected, and a message tells you if you try something that is not allowed.

After you use the **CM Config File Editor** command, fill in the Configuration file name, and press **Go**, the CM Editor screen appears. All the necessary editing is done in this single screen and its pop-up menus. Its main areas are labeled in Figure 5-1.

In general, **Return**, **Next**, or the **Up** and **Down Arrow** keys move the highlight from field to field within a given area. The **Left** and **Right Arrow** keys move the edit cursor within a field. The usual editing commands (**Delete**, **Code-Delete**, **Overtyp**e, and so on) work within a field. (See the *Forms Manual* for details.)

The CM Editor has a main function key menu. Its commands are listed in Table 5-1, below.

A step-by-step example of an editing session is given under "Examples," below, and details about each area and field of the CM Editor screen appear under "Screen Areas and Functions."

## SCREEN AREAS AND FUNCTIONS

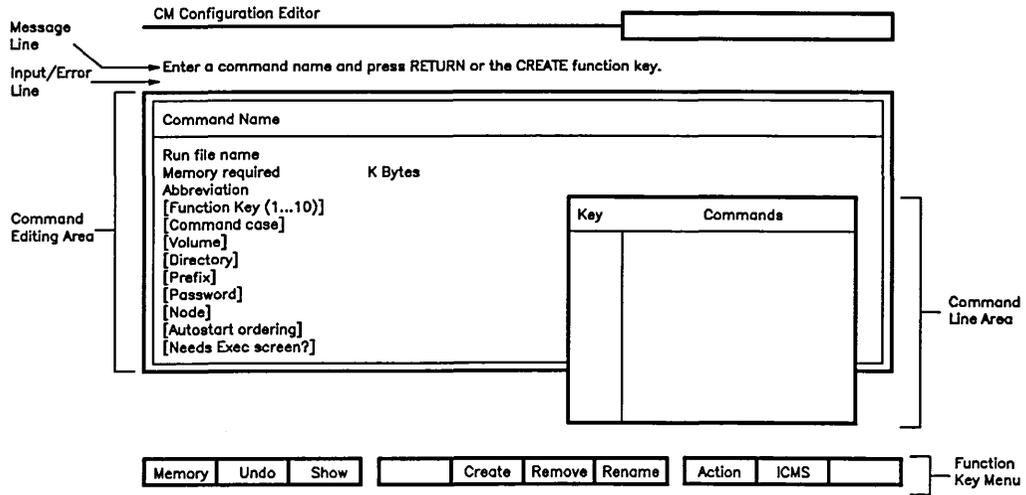
This section describes in detail the functions of each screen area. It is not a sequential tutorial. To see what a step-by-step session with the CM Editor is like, refer to "Examples," at the end of this section.

### Message Line

The first line below the status line of the CM Editor screen is the message line. During most of your work, it tells you what you can or should do next.

When the CM Editor screen first appears, the message line says

Enter command name and press Return or the  
Create function key.



1050-001

Figure 5-1. CM Editor Screen

**Table 5-1**  
**CM EDITOR FUNCTION KEY MENU**

---

| <b>Function Key Name</b> | <b>Purpose</b>  |
|--------------------------|---|
| <b>Memory (F1)</b>       | Allows you to specify that fonts and the application bit map be swapped to upper memory for performance improvement.  |
| <b>Undo (F2)</b>         | Replaces the value in the current field with the immediately previous value.  |
| <b>Show (F3)</b>         | Replaces the CM Editor function key menu display with a display of the current values on the CM screen function key menu.   |
| <b>Create (F5)</b>       | Records the command name entered in the Command Name field as a new command.  |
| <b>Remove (F6)</b>       | Removes the name in the Command Name field from the command list.   |
| <b>Rename (F7)</b>       | Allows renaming of the command entered in the Command Name field.   |
| <b>Action (F8)</b>       | Changes alphabetic characters used with the <b>Action</b> key from the default to a character of your choice. The default combinations you can change are Action-C (cut), Action-P (paste), and Action-W (windows). |
| <b>ICMS (F9)</b>         | Overlays part of the main editing area with an editing area for specifying the name of the ICMS Run file and the number and size of messages.   |
| <b>More (F10)</b>        | Overlays the command editing area with an editing area for entry of program-specific information to be read from the CM Configuration file.   |

---

## Input/Error Line

The next line of the CM Editor screen is the input/error line. It has the main function of telling you when and why you cannot do something that you have tried. Meanwhile, the message line tells you what to do next.

The messages that appear in these two lines are self-explanatory.

## Command Editing Area

You can add an application (such as Document Designer or Extended Multiplan) or an Executive command (such as **Floppy Copy** or **Files**) to the list that CM recognizes. You can also create an Executive Submit command (**SubCmd**), give it a command name, and identify it to CM as a regular command. (Instructions are given below under "Defining an Executive Submit Command in CM.")

No matter which category your choice belongs in, it is called a *command* in the CM Editor screen, because that is its function. Once it appears in the list on the CM screen, however, it is called an *application*.

The large rectangular box surrounded by a double rule is the command editing area. (See Figure 5-1 above.) You can use it to name and describe a command or application that you want to add to the list of applications that the user can start. You also use this area to edit or remove commands that already are listed.

In the CM Editor, you do not have to edit fields in a certain order, and you do not have to complete all fields in one session.

If you press **Cancel** when the highlight bar is on any of the command editing area fields, the current entry in the Command Name field and all the edits in the command area fields are discarded, and the highlight bar returns to the Command Name field.

**Command Name Field.** In this field, enter the name of the command or application you want to add, edit, or remove. (Note that if you are editing a command rather than defining a new one, you can enter only as much of the command name as is needed to make it unique.) Type the name of a new command exactly as you want it to appear in the list of "Applications you can start" on the CM screen (for example, Word Processor).

A name 32 characters long can be displayed on the command list in this screen. A 28-character name can be displayed on the list of "Applications you can start" and "Contexts you can return to." (Although the lists show only the first 32 and 28 characters, respectively, you can define a name of up to 50 characters on both screens.)

If you wish to prevent an application from ever running in the background, you can mark it explicitly as "dirty" by adding an asterisk at the end of the name of the command:

Command Name ScreenWriter\*

A dirty application is an application that writes directly to the screen. (See Chapter 6, "Notes for the Programmer," for a discussion of clean and dirty programs.)

You can press either **Return** or **Create (F5)** at this point. If you are adding a new command, press **Create (F5)**.

If you are editing information for an existing command, press **Return**. If the name you have entered is not identical to one on the list, the command is created.

If you are renaming a command without changing any other information about it, or if you are removing a command from the list, you can press **Rename (F7)** or **Remove (F6)** without pressing **Return** first. You can use **Rename (F7)** to add an asterisk to an existing command name, thus telling CM that this is a dirty program.

If you press **Return** or **Create (F5)** after filling in the Command Name field, the highlight bar that was on this field moves to the next field, the Run File Name field.

If you enter a new command name in the Command Name field and press **Go**, the new name is listed immediately in the command list area.

**Run File Name.** In this field, you can enter or edit the name of the Run file for the application or command you entered in the Command Name field. You do not have to have this Run file present on your system to do this configuration.

For a discussion of Run file names for applications that require parameters or assume an Executive screen, see "Applications You Can Start from CM," below.

In the command editing area, **Return**, **Next**, or the **Up** or **Down Arrow** keys cycle the highlight bar within this area only.

**Memory Required.** In this field, you enter the amount of memory required for this application. The amount you enter controls the size of the partition within which this application will run. The partition can be a fixed size or a variable size. A variable sized partition can shrink and grow (up to the maximum specified) based on information about its memory needs.

(The Context Manager/VM Release Notice lists memory requirements for standard applications. See Chapter 6, "Notes for the Programmer," for an estimation of memory requirements for user-written applications.)

If you want a variable sized partition to be created, type **<nnn** (where nnn represents the number of bytes, multiplied by 1024) or **<0**; if you want a fixed sized partition, type **0** (or leave the space blank), or **nnn**.

Create a variable sized partition as follows:

- When you specify a less-than sign before the size (<nnn), this indicates that the partition is to have a maximum size. For example, specifying <150 means that the maximum size of the partition is not to exceed 150K bytes. When you specify an amount in this way, the operating system creates a partition of a more precise size because the sizing information is taken from the Run file. (A Run file with sizing information specifies the minimum and maximum amount of memory required by your program.) Thus, if you chain from one program to another, and the program you chain to requires less than the maximum amount of memory that you entered, the partition size changes (to the size indicated in the Run file) to accommodate this other program. When that program chains, the partition is resized again, and so on.
- When you specify a less-than sign before a zero (<0), a partition of a more precise size is created in the same way that entering <nnn does. When you enter <0, though, the size of the partition is not limited by the amount that you have specified. A partition is created that does not exceed the maximum amount in the Run file or in available memory. (Available memory is the amount of memory left after the operating system, CM, and all system services have been loaded.)

You create a fixed sized partition as follows:

- When you specify 0 or leave a blank space, a partition is created that is equal to the size of available memory when the Run file is loaded. For example, if available memory is 800K bytes, and you load an Executive that has a maximum size of 180K bytes indicated in the Run file, a partition is created of 800K.
- When you specify an amount without a less-than sign (nnn), a partition is created that is equal to the size you entered. Thus, if you enter 150, a fixed sized partition is created of 150K bytes. This size does not change: If you chain to another program, the size of the partition still remains at 150K bytes.

**Caution:** *If you do not want your programs to use all available memory, use the less-than sign (<0 or <nnn) to specify that a partition be created of a maximum size.*

For more information on partition sizes, see "Determining the Size of a Partition" in Chapter 6.

*NOTE: It is to your advantage to create variable sized partitions by specifying <nnn or <0. They allow you to run more programs in memory at the same time.*

To examine the sizes of Run files, you can refer to the Release Notice for each product or the Context Manager/VM Release Notice, or you can use the Partition Status utility.

To use Partition Status for this purpose, you must have the appropriate applications running on the system: Run Partition Status in 30K bytes, and then your test application in a very large amount of memory.

Create the command **Partition Status** in the command editing area of the CM Editor. The command **Executive** should already be present in [Sys]<Sys>**CmConfig.sys**.

Configure the Executive to run in a very large partition. Start the Executive, and then give a command to run the application you want from that Executive. Then switch contexts to Partition Status to examine the size of the Run file in the large partition.

The information you get from Partition Status can also help you decide on more accurate application sizes. In most cases, an application Run file requires an amount of memory slightly larger than itself because of partition overhead. Some applications, however, "tune" themselves to fit into available memory space and do not require this allowance. Among these are Document Designer, Editor II, Bootstrap, and Linker.

For details about Partition Status, see the *Executive Manual*. Use the **Update** function key in Partition Status to see the latest information it can provide.

**Abbreviation.** This abbreviated name for the command appears on the function key menu in the CM screen when you start this application.

The default value is the first six characters of the Command Name field entry. Edit this field if you want to.

**[Function Key (1...10)].** You can choose to preassign a function key to this application so that the user does not have to do so at each session. (For a description of how these labeled function keys are used, see Chapter 3, "Basic Concepts.")

You can see which function keys are already preassigned by looking at the Key column at the left of the command list area. Enter the number of any available function key.

You can press **Show (F3)** to display the preassigned function keys on your own function key menu. In this way, you can check the arrangement of function keys to see if it is convenient for the user. Press **Cancel** to dismiss the function key menu displayed by **Show (F3)**.

You do not have to make any entry in this field.

*NOTE: Do not assign all 10 function keys to applications. At least one should be left blank so that the user can start applications that do not have preassigned function keys.*

**[Command Case].** Some applications (notably the Document Designer) carry out different processes depending on what command was used to invoke them. (For the Document Designer, these commands are **Document Designer** and **Resume Document Designer**.) The command case is a two-character parameter that is passed to the application to tell it which command has been given, and thus what process should be performed. (See the section on parameter management in the *CTOS/VM Concepts Manual* for a further discussion of command case. See also "Adding a New Command," in the *Executive Manual*.)

If the application you are describing can be invoked by more than one command, you must enter a value for the command case parameter in this field. For the correct parameter value, see the release notice for that application.

Some applications, when started, require parameters. These applications must be started from the Executive. For such an application, the Run file should be given as [Sys]<Sys>Exec.run and should require the entry CM in the Command Case field. (For details, see "Applications You Can Start from CM," below.)

The default value of Command Case is 00.

**[Volume].** You can specify the path to be used when an application is started. This field allows you to specify the volume for that path.

If no entry is made in this and the next four fields, the new application will be written to the same volume in which CM is running.

**[Directory].** Specify in this field the directory to be used in the default path for the application.

**[Prefix].** Specify the prefix, if any, to be used in the default path for the application.

**[Password].** Give the password, if any is needed, to be used as a default by the application.

**[Node].** Specify the node, if any, to be used in the default path for the application.

**[Autostart ordering].** The Autostart feature allows you to specify applications that should be started as soon as CM is loaded. You assign the number 1 in this field for the first application to be started by CM, 2 for the second, and so on, up to 9. Leaving this field blank or specifying 0 indicates that this application should not be autostarted.

The last application started becomes the current context and gains control of the screen and keyboard. Thus, if you gave the Executive an Autostart ordering value of 1, the Document Designer a value of 2, and Electronic Mail a value of 3, the user would sign on and see the messages

Installing Context Manager . . . done

Autostarting . . .

and would then see the first screen of Electronic Mail. Under Autostart, the CM screen is not displayed after the user signs on.

If you assign duplicate numbers in this field, CM starts those applications in alphabetical order.

**[Needs Exec screen?].** Some applications assume that the Executive's screen is present and begin their video output without clearing the screen and setting it up themselves. For such applications, respond **yes** in this field to cause such a screen to be set up for the application before it starts.

**[Color].** If you have a color monitor, you can specify that the default color for the application be amber, blue, green, yellow, or white.

## Function Key Menu

For a general discussion of the use of the function keys in CM, see Chapter 3, "Basic Concepts."

Table 5-1, above, briefly describes the effect of each function key in the CM Editor.

At any given time while you are working in the CM Editor, only those labels that are appropriate at that time are shown on the menu.

Most of the commands in this menu have been discussed in earlier subsections. Some require further information.

**Memory (F1).** This command allows bitmaps and fonts to be swapped to upper memory rather than to disk during a context switch. This improves performance. When you press **F1**, the Resident Bitmap and Font Memory menu is displayed.

For the Font Reserve, you can specify up to 64k of memory. The average font requires 8k.

For the Bitmap, you can also specify up to 64k. Since the memory required by the bitmap varies depending on the complexity of the current screen display, 64k is recommended for this entry. Smaller amounts result in smaller improvements to performance.

**Undo (F2).** This command replaces the value in the current field (the field indicated by the highlight bar) with the immediately previous value in that field. In numeric fields, if the immediately previous value was blank, **Undo (F2)** replaces the current value with 0.

**Show (F3).** This command displays the current values on the CM screen function key menu so that, while making new assignments, you can see what keys are already assigned.

**Remove (F6).** This command allows deletion of a command name from the Command Name list. The highlight bar can be on any field of the command editing area (including the Command Name field) when you press this key.

A screen message asks you to confirm deletion of the command. Press **Go** to confirm deletion, or **Cancel** to retain the command on the list.

**Rename (F7).** This command allows you to change the name of a command. Enter the old name of the command in the Command Name field. You can press **Rename (F7)** immediately, without pressing **Return**.

Edit the field to show the new name of this command. When you press **Go**, the highlight bar moves to the Run File Name field. You are not, however, required to edit the Run file name.

Press **Go** to record this change of name, or **Cancel** to discard it.

**Action (F8).** To change the default characters used with the **Action** key to invoke Cut, Paste, or Windows, press the **Action (F8)** function key. Then use the form displayed to change the default alphabetic key to the one you want to use. You can disable Cut or Paste by leaving the characters for those functions blank.

**ICMS (F9).** If you plan to run applications that pass messages to each other, these applications require the presence of the Intercontext Message Server (ICMS). Some integrated applications, such as Document Designer, make use of CM in this way.

If you need ICMS on this system, press the **ICMS (F9)** function key. The ICMS editing area will pop up on the upper part of the command editing area. In this area, fill in the name of the ICMS Run file, the number of messages, and the size of each message.

Typically, the Run file is [Sys]<Sys>ICMS.run. To save space and to ensure identical versions throughout a cluster, a Run file may be placed on the master. In this case, the Run file name is [Sys]<Sys>ICMS.run.

The number and message size fields default to 1 and 200 bytes, respectively.

If you specify a Run file name for ICMS here, ICMS is installed when the user installs CM. Alternatively, if ICMS is to be used all the time, you may want to have ICMS installed during system initialization by including the following line in the JCL file:

```
$Run [Sys]<Sys>ICMS.run
```

For details, see the Engineering Update for the CTOS/VM System Administrator's Guide.

**More (F10).** This command is available after you have entered a command name in the Command name field and pressed **Return** or **Create (F5)**. It causes an editing area entitled "More Information" to pop up over the command editing area. This area allows you to enter any information that your program needs to read when it is running. (For details, see "Placing Information in the CM Configuration File," in Chapter 6.)

### **Command List Area**

The boxed list of commands with their assigned function key numbers, shown in Table 5-1, is the command list area. This list shows the currently allowed commands, that is, those recognized by CM under this Configuration file. (On the CM screen, these commands are called "Applications you can start.")

This screen area is not an editing area, but reflects what has been recorded from other editing areas in this session and in previous ones.

CM can recognize 17 commands that you define through the CM Editor. CM can manage a maximum of 10 contexts, or started applications, at once. You are not restricted to defining 10 commands.

Note that if you have entered the CM Editor while running under CM in order to edit your own Configuration file, the changes you make do not appear on the "Applications you can start" list in the CM screen until you log out and then reinstall CM, at which time CM reads the CM Configuration file again.

### **Exiting the CM Editor**

When you have completed your editing session, press **Finish** to exit the CM Editor. A screen message asks you to confirm that you want to finish. The input/error line tells you if any inconsistencies have been found. You have the opportunity to remain in the CM Editor and correct them.

If you want to overwrite the Configuration file with your edits, press **Go**. Note that it is only at this point that any of your edits become part of the Configuration file, even though some of your changes have already appeared in the command list area in the screen.

If you decide not to save your changes, press **Cancel**.

### **EXAMPLES**

#### **A Complete New Entry**

1. In the Executive, type in the command **CM Config File Editor** (or a unique abbreviation, such as **c c f e**) and press **Go**.

The default file name [Sys]<Sys>CMconfig.sys is displayed.

2. Press **Go** to access the default file.

If you want, you can assign a different file name: Type in the new file name and press **Go**. For example, a file name for a specific user might be [Sys]<Sys>UserCMConfig.sys, where **User** identifies a unique name associated with the individual user.)

The CM Editor screen appears with the highlight bar on the Command Name field. You want this user to be able to start the Document Designer.

3. Type in the name **Document Designer** and press **Create (F5)**.
4. Type in the name of the Document Designer Run file (for example, [Sys]<Sys>DocumentDesigner.run) and press **Return**.
5. Type **375** (the amount of memory required for the Document Designer) and press **Return**.

The highlight bar moves to the Abbreviation field, and the default entry **Docume** has been clipped from your entry in the Command Name field.

6. Suppose you feel that **D D** would be a better abbreviation. Hold down **Code** and press **Delete** to remove the current entry, type **D D**, and press **Return**.

The highlight bar moves to the Function Key field. You decide to assign function key **F1** to Document Designer. (If **F1** has already been assigned, choose a function key number that is available.)

7. Type **1** in the Function Key field, and press **Return**.
8. Do not make an entry in the Command Case field if **00** is already specified.

The Document Designer is an application that does require a Command Case field entry. The command case for the Document Designer function is **00** (and **01** is for resume Document Designer).

9. Press **Return**.

10. Leave the Volume field and the next four fields (Directory, Prefix, Password, and Node) blank.

You want Document Designer for this user to be pathed into the directory where documents are kept. You would specify the volume, directory, prefix, password, and node for this path if they differ from CM's path. (If, for example, your documents are kept in the directory named **docs** on volume D1, you would have to indicate that in the Volume and Directory fields.)

11. Assign **1** as the value in the Autostart ordering field.

You want Document Designer to be started automatically when this user signs on. If you do not assign larger numbers to other applications in this field, Document Designer will be the first application displayed after the user signs on.

12. Do not make an entry in the Needs Exec screen field.

The default **n** indicates that Document Designer does not require an Executive screen.

13. Press **Go** to record your entries for this command. (If you press **Return** instead, the highlight bar goes back to the Run File Name field.)

The name **Document Designer** and the function key name **1** appear in the command list area, and your entries for this particular command are complete.

14. Press **Finish** when you have completed all your work on this Configuration file.

15. Press **Go** when prompted to confirm recording the entries you have made in this file.

To see the effect of your new entry, finish the Executive (and other active contexts you want to save), logout (by pressing **Finish** and then **Go**), and sign on to reinstall CM.

Note that you can also add a command using the **CM Add Application** command in the Executive, as described below in "Changing a Configuration with Executive Commands."

## Editing a Command

When you were entering new commands in this Configuration file, perhaps you added Extended Multiplan and assigned it to function key **F6**. The user now asks you to change the function key abbreviation from **Extend** to **X MP**.

Enter the CM Editor, and press **Go** to accept the default.

In the Command Name field, type the command name **Extended Multiplan** and press **Go**. When the highlight bar enters the command editing area, use **Return**, **Next**, or the **Down Arrow** key to move it to the Abbreviation field. Press **Code-Delete** to remove **Extend**, and type in **X MP**. Press **Go**.

If you have no other changes to make, press **Finish**, and confirm overwriting this user's file by pressing **Go**.

## Removing a Command

Perhaps when you first defined this Configuration file, you added the **Copy** command to the list of available commands because you thought **Copy** would be used often. This turns out not to be true, and the user wants it removed.

As before, enter the CM Editor, and press **Go**.

In the Command Name field, type the name of the command you want to remove, **Copy**. You can now press **Remove (F6)**, and then press **Go** in response to a message that asks you to confirm the deletion. The name is removed from the list. If you have no other changes, press **Finish** and **Go** to overwrite the file.

You can also use the **CM Remove Application** command from the Executive to remove a command, as described below in "Changing a Configuration with Executive Commands."

## Renaming a Command

On this command list, you originally typed **Asynchronous Terminal Emulator** as a command name. You would like the name to read **ATE**.

After entering the CM Editor, type in the Command Name field the old name, **Asynchronous Terminal Emulator** (or only as much of that name as you need to distinguish it from other commands).

You can now press **Rename (F7)**, delete the old name, and type in the new name: **ATE**. When you now press **Go**, the highlight bar moves to the Run File Name field, but you do not have to edit this field, even though you are given the opportunity to do so. (In this case, the Run file does not change.) Press **Go** again to record your change. The new name replaces the old one on the list.

## A Common Error: Inconsistent Entries

Where an error cannot be compensated for by an entry in another field, the current field does not allow you to exit. For example, if you specify function key **20** in the Function Key field, you are not allowed to exit that field, because no other entry can make **20** correct.

## *Changing a Configuration with Executive Commands*

Two Executive commands allow you to change the contents of the CM Configuration file without using the CM Editor. They are **CM Add Application** and **CM Remove Application**.

## CM ADD APPLICATION

This command allows you to add an application to a specified Configuration file. It is analogous to the **New Command** command for creation of Executive commands and is useful when you are installing new software. The command form is as follows:

|                      |       |
|----------------------|-------|
| CM Add Application   |       |
| [CM Config File]     | _____ |
| Application          | _____ |
| Run file             | _____ |
| Memory required      | _____ |
| [Function key]       | _____ |
| [Command case]       | _____ |
| [Volume]             | _____ |
| [Directory]          | _____ |
| [Prefix]             | _____ |
| [Password]           | _____ |
| [Node]               | _____ |
| [AutoStart ordering] | _____ |
| [Needs Exec screen?] | _____ |
| [Overwrite OK?]      | _____ |
| [More information]   | _____ |
| [Default Color]      | _____ |

The fields in this form are analogous to the similarly named fields in the CM Editor. The [Overwrite OK?] field allows you to overwrite old information about a command with new information about the same command. But if you overwrite an existing application without explicitly supplying path and/or Autostart information, the old path and/or Autostart information is kept.

## CM REMOVE APPLICATION

This command allows you to remove an application from a specified Configuration file. It is analogous to the **Remove Command** command in the Executive. The command form is as follows:

```
CM Remove Application  
Application _____  
[CM Config File] _____
```

### *Applications You Can Start from CM*

Applications that can start directly from CM must be able to be invoked without parameters.

Examples of such applications include BASIC, the Executive, Mail, and Extended Multiplan. When you are using the CM Editor to define these commands, you can specify their associated Run files in the Run File Name field.

Applications that do not follow this rule still can be invoked indirectly from CM. They are invoked through the Executive.

As an example, you want to invoke the **Selective Backup** command from CM. **Selective Backup** has parameters that must be specified (the file list to be backed up) and optional parameters that are passed to it through the Executive.

To add **Selective Backup** to the list of applications you can start, use the CM Editor as follows:

1. In the Command Name field, type **Selective Backup**.
2. In the Run File Name field, type `[Sys]<Sys>Exec.run` (instead of `[Sys]<Sys>SelectiveBackup.run`).
3. Enter the appropriate memory size, abbreviation, and optional preassigned function key.
4. In the Command Case field, enter **CM**.

Note that, if you wanted to define the Executive itself as a command rather than **Selective Backup**, you would have left the Command Case field blank.

Under this arrangement, when you choose **Selective Backup** from the list of applications you can start in CM, the Executive is actually invoked.

Part of the Executive's purpose is to supply a convenient way to pass parameters to an application. When the Executive begins execution, it checks to see what case value was used when it was invoked. The case value CM indicates that it was invoked from CM. The name of the command you specified (**Selective Backup**) is passed to the Executive, which then brings up the appropriate form. You fill out the form and press **Go** to start **Selective Backup**. When it is complete, it returns to CM.

Some applications do not recognize command case values, but assume that they have been called from the Executive and derive the equivalent information from the number of lines in the command form. These applications also should be invoked through the Executive and defined with CM as the command case value. If you define one of these applications directly, the following message appears when you try to start it in CM:

This application must be invoked through the Exec;  
edit the Config File.

Use the CM Editor to specify [sys]<sys>**Exec.run** as the Run file and CM as the command case for such an application.

## *Defining an Executive Submit Command in CM*

An Executive **Submit** command is a streamlined form of the Submit facility that allows you to define a series of Executive commands to be carried out in sequence when you give one command.

You can create an Executive **Submit** command and give it a command name for use under CM. Suppose you want to create a **Submit** command to initialize floppy disks using **IVolume**, because you do this operation often. Perform the following steps:

1. Create a new command to be the **Submit** command.
2. Create a **Submit** file containing the Executive command **IVolume** and its details.
3. Test your **Submit** command.
4. Enter the **Submit** command as a command in the CM Editor.
5. Deinstall and reinstall CM to cause the new CM Configuration file to take effect.
6. Test your **Submit** command through CM.

## CREATE A NEW COMMAND

Use the **New Command** command in the Executive. Choose a command name that you want to use to invoke this sequence of actions in CM. For the **IVolume** example, you might use **IVFloppy**. In the Run File field of **New Command**, you must specify **[Sys]<Sys>Subcmd.run**. This name is the only acceptable one.

## CREATE A SUBMIT FILE

For details on creating a Submit file, see the *Executive Manual*. The name of the Submit file must be of the form

```
[Sys]<Sys>CommandName.sub
```

In this example, it must be

```
[Sys]<Sys>IVFloppy.sub
```

If there are spaces in the command name you created, delete those spaces in forming the Submit file name.

If you create your Submit file using the **Executive Record** and **Stop Record** commands, you must then invoke the Editor and delete from this file the words **stop record** and the trailing bullet that represents the **Go** keystroke at the end of the file.

## TEST YOUR SUBMIT COMMAND

In the Executive, give the new command (**IVFloppy**) to verify that it works.

## DEFINE THE COMMAND IN THE CM EDITOR

Enter the CM Editor. In the Command Name field, enter your new **Submit** command name (**IVFloppy**). In the Run File field, you must enter

```
[Sys]<Sys>Exec.run
```

In the [Command case] field, you must enter **CM**.

## FINISH UP

Press **Finish** and then **Go** to deinstall CM, and sign on, re-installing CM. Doing so makes the needed changes effective in the CM Configuration file.

Your SubCmd should appear on the list of "Applications you can start." Test it in the same way that you would give any other command under CM.

## *Invoking the Submit File from CM*

You can run any Submit file directly from CM. For example, you may have a Submit file called **Cleanup.sub**. (Be sure it is in [Sys]<Sys>.) To invoke this file from CM, use the **CM Config File Editor** to add the application, Cleanup, with the Run file set to [Sys]<Sys>SubCmd.run.

## *Text of the Configuration File*

You can use the **Type** command in the Executive to look at the CM Configuration file itself.

The example of a file in Figure 5-2 contains one command (the **Executive**). A series of entries like that shown here for the Executive should exist for each command you have described using the CM Editor.

This file has been created by the Configuration File Editor for use by CM. The file consists of fields that are recognized by **FieldName**, followed by a parameter. Modification of this file should be done carefully.

It is also important to note the following:

1. FieldNames must begin in column 1; blanks are significant after the colon.
2. MemorySize is specified in K bytes.
3. Command descriptions are assumed to be in sorted order.

---

```
:ICMSFile:  
:NumberOfMessages:  
:SizeOfMessage:200  
  
:CommandName:Executive  
:RunFileName:[sys]<sys>Exec.run  
:CommandAbbreviation: Exec  
:MemorySize:200  
:CommandCase:00  
:FunctionKey:1  
:Autostart:0  
:Invoker:n  
:Volume:  
:Directory:  
:Prefix:  
:Password:  
:Node:
```

For  
each  
command

---

**Figure 5-2. Example of a Configuration File**

## *Allowing for More Contexts*

Each version of the operating system is built to support some maximum number of programs running at one time. Each program runs in a separate partition of memory. Thus, the total number of partitions (nPartitions) needed for any system is the sum of the following:

- 1 for the operating system partition
- + 1 for the Context Manager partition
- + 1 for the Debugger (if used)
- + x number of contexts desired
- + y number of system services

---

= nPartitions

As each application is started, CM creates a partition and then loads the application into that partition. If you try to start an application that results in the creation of one more partition than the system was built to handle, the following message is displayed:

This version of the OS cannot support any more contexts.

The number of partitions is specified in the Prefix file for your particular operating system. The naming of the Prefix file is based on the operating system version. See the CTOS/VM Release Notice, "Building an Operating System," to determine the prefix file for your workstation.

The Prefix file should contain a line similar to the following example:

```
%Set(nPartitions, 12)
```

## EXAMPLE

You want seven contexts and have three system services installed. You want to use the Debugger. The number of partitions to specify is

|       |                   |
|-------|-------------------|
| 1     | (OS partition)    |
| + 1   | (Context Manager) |
| + 1   | (Debugger)        |
| + 7   | (contexts)        |
| + 3   | (system services) |
| <hr/> |                   |

- 13

Once the appropriate Prefix file has been modified, follow the directions in the CTOS/VM Release Notice for assembling the SysGen files and linking the new version of the operating system.

From the application programmer's viewpoint, CM makes itself part of the operating system, to provide an environment that allows you to run several applications at once.

### *CM and Its Relationship to the Operating System*

The operating system resides in one partition of memory, and CM is in another partition. Initially, the rest of memory is unallocated. When you start an application, the operating system creates a partition in memory for it and then moves the application into this partition.

### *What Runs under CM?*

With CM installed, most existing applications can run without change, and in fact without recompiling or relinking. But to create more precise partitions, it is recommended that you relink with the Bind command. (See "Activating the Linker," in the *Linker/Librarian Manual*.)

Programs that require change to run with CM installed are the following:

- a program that writes directly to the screen map in memory
- a program that contains a "busy wait" loop that does not allow the processor to give up the keyboard when the program is waiting to read keystrokes
- a program that changes a value in the low-memory interrupt vector table directly in memory
- a program that positions the cursor using the video controller port
- a program that changes the Exit Run file

## **PROGRAMS THAT WRITE DIRECTLY TO THE SCREEN MAP**

Each memory partition has a system structure called the *video pointer map* and an associated *application character map*. The video pointer map is an array of pointers, one for each line of the screen, that always point to the location of the associated line of the application's screen. The application character map is a memory area where lines of the screen (the video lines) are stored when an application is running in background.

CM controls the mapping of the video lines to ensure that the output from all write requests goes to the correct place: either to the real screen or to the application character map. The Video Access Method (VAM) and Video Display Management (VDM) do this mapping invisibly to the application. Therefore, programs that use these calls work automatically. Programs that write directly to the screen, however, cannot be detected; the output from their write requests goes directly to the screen whether they own the lines of the screen or not.

*NOTE: Rather than write directly to the screen, it is strongly recommended that you use VAM to write to the display, especially if you intend your application to run in a windowed environment.*

To allow your programs that write directly to the screen to work, you can use three system common operations:

LockVideo

UnLockVideo

GetpStructure

For details on the use of these operations, see the *CTOS/VM Reference Manual*.

During initialization, you must first get a pointer to the video pointer map. (The name of the video pointer map is **rgpVidMemLine**.) You do this by calling **GetpStructure** and passing a value of **IGetpRgpVidMemLine**. The returned pointer points to the video pointer map.

Later in the program, when you want to move characters to the screen, you must lock all the video structures so that they will not be changed by other programs while you are using them. This is done by a call to **LockVideo**. To do the actual write, you need to have a structure that looks like a single line of the video and is based on the appropriate pointer entry in the video pointer map. When your write is finished, you must unlock the structures by calling **UnLockVideo**. The series of locking, writing, and unlocking should be done on a line-by-line basis.

Because these operations lock the video during a write, they ensure that one program writing to the screen does not interfere with another one that is writing at the same time.

*NOTE: If these calls are not used, CM does not permit the program to run in background: It suspends the program and displays the status Stopped.*

## "BUSY WAIT" LOOPS

In dealing with the keyboard, a program should not use a "busy wait" loop.

The normal way to read keystrokes from the keyboard is to issue a CTOS **ReadKbd** with mode 0. A busy wait may be generated by a program that uses calls to **ReadKbdDirect** with mode 1. Since mode 1 means that the operating system returns immediately whether a character is available or not, the program loops, waiting for a character to arrive.

When a program is running in foreground, CM raises its priority (that is, gives it a lower numeric value); when a program runs in background, its priority goes back to normal. If a program contains a busy loop and is running in foreground, its priority is higher than those of other programs in background. Because this foreground program never waits, background programs can never get any processor time, and therefore they stop running.

## LOW-MEMORY INTERRUPT VECTOR TABLE

If a single application is running, it may change any value in the low-memory interrupt vector table without problems. Because more than one application can run under CM, each application must inform the operating system that it wants to change one of the values in this table. To do this, use either of the following CTOS calls:

**SetIntHandler**

**SetTrapHandler**

Use **SetIntHandler** if your application should not be swapped, for example, if the handler that you are providing is a hardware interrupt service routine.

Use **SetTrapHandler** to allow your application to be swapped out. For example, if the handler that you are providing is a software interrupt service routine, your program can be swapped, because no interrupts of this type can occur while the program is swapped out. (See the *CTOS/VM Reference Manual* for more information.)

## POSITIONING THE CURSOR

Since a program cannot know whether it is running in foreground or background, it must not pass values to the video controller port to move the cursor. Instead, use the CTOS procedure `PosFrameCursor` to position the cursor.

## APPLICATIONS SUSPENDED IN BACKGROUND

An application that writes to the screen directly, uses the video controller port, or in any other way interferes with normal operation under CM is called a "dirty" program. To avoid unpredictable results, contexts are referred to as clean or dirty according to the following rules:

|                   |   |
|-------------------|---|
| Tentatively clean | Contexts start in this state.   |
| Tentatively dirty | Contexts in the tentatively clean state go to this state if they do a <code>ResetVideo</code> .   |
| Absolutely clean  | Contexts go into this state if they do any of the following:<br><br><code>GetPStructure (IprgpVidMemLine)</code><br><code>InitVidFrame</code><br><code>NotifyCM (not IMarkDirty)</code> |
| Absolutely dirty  | Contexts go into this state if they do <code>NotifyCM (IMarkDirty)</code> .   |

When a dirty context is not on the screen (that is, when it is in the background), it is suspended. This is true whether a context is tentatively or absolutely dirty.

*NOTE: If you have Windows installed on your system, the rules described above do not apply to your program. Programs that run under CM with Windows must conform to a different set of rules. See "Programs That Run Under Windows," below, for more information.*

Note that the procedure calls mentioned above do not contain complete parameter lists. See "Communication with CM," below, for a description of the NotifyCM procedure call. The other calls are described in the *CTOS/VM Reference Manual*.

Detection of dirty programs is not flawless. If you have such a program, you should change it: Unpredictable and unfavorable results, such as multiple cursors, overlapping text from several applications, and so on, may occur.

## PROGRAMS THAT RUN UNDER WINDOWS

Although any program that runs under CM can also run under Windows, you must follow certain rules with those programs that run under both:

- Do not get a pointer to the video map. (That is, do not use GetpStructure to access `rgpVidMemLine`.)
- Use only VAM to write to the display.

When programs abide by the rules by using VAM and/or Windows, they are called "conforming" programs. When they "break" the rules by using pointers to the video map, they are called "nonconforming."

The programs that run under Windows are the following.

1. Conforming programs that are window-aware
  - UConforming programs that are window-aware use VAM and Windows.
  - The windows of these programs have borders and can be sized.
  - These applications may be informed of changes to the window size and decide to reformat the data displayed in the window. Thus, the program becomes aware that its size has changed.

The Windows program notifies the application program about the change in window size by responding to an outstanding Window Event request. (See the *Window Services Manual* for a complete description of window requests.)

## 2. Conforming programs that are not window-aware

- Conforming programs that are not window-aware use only VAM.
- The windows of these programs have borders and can be sized.

When the window changes its size, the Windows program does not notify the application, because there is no Window Event request outstanding. Thus the program is not aware that its window size has changed. The program continues to operate as if it had a full screen. If you make the window smaller, the contents of the entire screen will not be visible in the smaller window. (For example, the function key menu or messages may not show up on the smaller screen.) The program has not been informed that its size has changed, so it does not reformat its window accordingly. It still operates as if everything were visible.

A user of nonwindow-aware programs can make the "hidden" contents of a window visible by using the **Action** key in combination with the **Arrow** keys and **Scroll** keys. (See Table 4-4, "Making Hidden Window Contents Visible.")

## 3. Nonconforming programs that are not window-aware

- Nonconforming programs that are not window-aware use the video pointer map to write directly to the character map. That is, they use `rgpVidMemLine` by calling `GetpStructure`.

These programs might use VAM.

- The windows of these programs do not have borders and cannot be sized. The screen must always be full size.

When the window of one of these programs is on top, it covers all other windows.

- When these programs go into background, they continue to run, although their video is suspended.

All video I/O goes to the application character map.

## EXIT RUN FILE

When the user selects an application to start from CM, CM loads the Run file and sets the Exit Run file to **CmNull.run**. Therefore, when the selected application finishes, **CmNull.run** is executed. **CmNull.run** sends a message to CM asking CM to terminate this context. Upon receiving this message, CM terminates the context that sent it.

If you write an application that changes the Exit Run file, CM never gets the message that the program has ended. Therefore, in such a case, you must provide a way to run **CmNull.run**.

For example, the Executive sets itself as the Exit Run file. Thus, when an application that was started from the Executive finishes, the Executive is reloaded. The Executive command **Finish Executive** invokes **CmNull.run**, thus allowing the user to return to CM.

## *Applications that Cannot Be Swapped*

Certain calls to CTOS, such as **SetCommIsr** and **SetIntHandler**, identify an application as one that cannot be swapped. Real-time and communications applications that include such calls are recognized as not swappable, and CM never tries to swap them to the disk.

If you wish to ensure that your application is not swapped out, add a call to the CTOS routine **SetSwapDisable**. From there on, the application cannot be swapped out. (See the *CTOS/VM Reference Manual* for more information.)

## *Communication with CM*

Applications can communicate with CM using the Interprocess Communication facility (IPC) in CTOS. (See the *CTOS/VM Concepts Manual* for more information on the use of IPC.) CM waits for the user to type some **Action**-keystroke or for a message from an application. An example of the latter is when the Executive sends a message to CM telling it the last command that the user entered.

The communication is achieved by means of an operating system call, `NotifyCM`, which is defined in Chapter 7, "Operations."

## *Communication Between Applications*

### **INTERCONTEXT MESSAGE SERVER**

Applications that are running in separate partitions under CM can communicate with each other by passing messages using the Intercontext Message Server (ICMS). Messages can be of arbitrary length and format.

A context sends a message to another context by sending it to ICMS, specifying the context handle of the receiver. (Each context is identified by a context handle that CM assigns when it is first loaded. See "CM as a Server," below, for a discussion of the context handle.) The receiver can either wait for a message to arrive or periodically check for the arrival of a message.

In this regard, ICMS is similar to IPC. It differs from IPC in one important way. Under IPC, only the address of the message is transferred. If the context sending the message is swapped to disk before the message is received, this method may not work. Even though IPC has the address of the message, the memory space occupied by the message may be overwritten by another context that was swapped into the same physical partition.

Under ICMS, the message itself is copied into ICMS' memory space, where it waits for the receiver to claim it. Because the actual message is copied, you can write programs to use ICMS to pass messages without worrying about swapping problems.

When ICMS' memory space for messages is exhausted, ICMS writes the messages to a disk file. The user configures the amount of memory allocated for ICMS message saving using the CM Editor. Function key **F9** in the CM Editor is assigned to ICMS. Pressing this key causes an editing area to "pop-up" for the ICMS Run file and the number and size of messages. The default is one message of 200 bytes. (These values are correct for the Document Designer.)

If ICMS is to be used constantly, you may wish to have it installed at system initialization through an entry in the JCL file. See "Function Key Menu" in Chapter 5 for details.

ICMS queues messages and dispenses them on a first-in-first-out basis.

For more information, see the Engineering Update for the *CTOS/VM System Administrator's Guide*.

## PROCEDURAL INTERFACES

The following procedural interfaces are described in detail in Chapter 7, "Operations."

ICMSCurrentVersion

ICMSCheck

ICMSFlush

ICMSSend

ICMSWait

## *CM as a Server*

CM provides programmatic interfaces to its internal functions of starting, switching, and terminating contexts. In short, anything the user can do from the CM screen is available as a procedure call.

Together with ICMS, which allows programs in different partitions to communicate, these calls form tools to build a system of integrated programs.

The Document Designer makes extensive use of the functionality described here. The Document Designer allows a user to prepare a document that may contain text, graphics, spreadsheets, and/or voice. The user is unaware that separate programs are actually running to handle the different pieces of the document. CM and ICMS provide services that allow the complete integration of full applications such as Word Processor, Multiplan, and the Business Graphics Package.

### **PARENT/CHILD RELATIONSHIP OF CONTEXTS**

Every context has a parent. When the user starts a new application from the CM screen, that context's parent is CM. When a program calls CM to start a new application, the caller is considered to be the parent context, and the new application is referred to as the child.

When a child context is terminated, CM assigns the keyboard, screen, and Mouse (if there is one) to the parent context.

A context may be dependent on its parent for survival, so that when its parent is terminated, it also is terminated.

If a parent context is terminated for any reason, all its dependent child contexts are terminated, and CM becomes the parent of contexts that were not dependent.

## CONTEXT HANDLE

When an application is first loaded, CM assigns to it a unique identifier known as its context handle (ch). The context handle is needed for all subsequent calls to CM regarding that context.

A context handle differs from a user number in that it allows for differentiation of contexts over time. For example: A user installs CM and chooses to start application A from the CM screen. The user finishes from application A and the CM screen reappears. The user again chooses application A. The first and second invocations of application A have the same user number. However, they have different context handles so that it is possible to tell them apart.

If a program is to switch to another context, the program must supply the context handle of the target context. Also, to send a message by means of ICMS, a program must determine the context handle of the receiver. The following are three examples of how to do so.

1. The sender is a parent context that wishes to send a message to one of its child contexts. The context handle of the child is returned by CM when the parent starts the child.
2. The sender is a child context that wishes to send a message to its parent. The child can call `CMQueryParent` to determine the context handle of its parent.
3. The sender and receiver are running concurrently. Both contexts should use the `SetPartitionName` routine to set the name of their partitions. If context A wishes to switch to context B, context A can issue a `GetPartitionHandle` using the name of B. Then, using the returned user number (`userNum`), it can call `TranslatePhToCh` to find the context handle of B.

*NOTE: If a program is loaded by means of a call to Chain or Exit, the context handle stays the same. For example, the context handle for all programs started from the Executive in a partition is the same. Only after a Finish Executive command is given and a new Executive is reloaded does the context handle change.*

There are two reserved context handles. The context handle of CM is 0, and 0FFFFh represents the nil context handle.

## STARTING AND SWITCHING CONTEXTS

A program running in one context can call CM to start a new application in another context. Similarly, a program in one context can call CM to switch ownership of the keyboard, screen, and Mouse (if there is one) to another context.

The following is a discussion of a problem that may be encountered when starting a new context or switching to an existing one.

In some cases, the caller may need to swap out before the next context can be loaded. However, the outstanding request to start the new context causes the caller to be marked unswappable, thereby causing a deadlock. To solve this problem, the caller context must use two requests.

The first request supplies the information about the new context to start. CM responds to the first request, and then attempts to start the new context. The second request is sent so that CM can return any error that was generated during the process of starting the new context.

To make this sequence easier for the programmer, the routines CMStartAppl and CMSwitchContext are provided. They make calls to start or switch, and then call to check the status code. Using these routines, you can perform a start or switch by coding a single call. The use of these routines is highly recommended.

Whenever a new application is started, any path information not explicitly entered in the Configuration file for that application is inherited from the parent context.

## CHECK FOR INSTALLATION

If you are writing code that depends on CM and, optionally, ICMS being installed, your code should include calls to `CMCurrentVersion` and, if appropriate, `ICMSCurrentVersion`. These routines check to ensure that the proper CM (and ICMS) loadable requests that come from the file, **Request.7.sys**, were loaded when the workstation was booted. If CM (and ICMS) are not installed, calls to CM or ICMS return status code 33 (no such service). If the loadable request file is not present, such calls return status code 31 (service not available).

## CM SCREEN

CM is designed to allow you to create a system of cooperating programs that take advantage of CM functionality without using the CM screen. In fact, you can write a system of programs that use CM configured in such a way that, if desired, the user never sees the CM screen.

If you use the Autostart facility, you can configure CM at installation to automatically start one or more applications. (For more information, see "Autostart Ordering" under "Screen Areas and Functions," in Chapter 5.) The screen appears as follows:

```
Installing Context Manager . . . done.
```

```
AutoStarting . . .
```

The next thing the user sees is the last application autostarted. The CM screen does not appear. When a context calls the routines to start a new application or switch to an existing context, the CM screen does not appear. Finally, when **SignOn.run** is run in a context and there are no other active contexts, or all other context(s) are Executives at the command line, the user is logged out without seeing the CM screen.

Thus you can design your own interface to CM. A small "main screen" driver can be written to run in a small partition. This program can be autostarted when CM is installed. The driver can then present the user with a list of things to start, in whatever form you design. When the choice is made, the driver calls CM to start the new application. Since the applications are started from the driver, the driver is the parent of all contexts. Whenever a child context is finished, the driver screen reappears. Finally, when the user wishes to log out, the driver program chains to **SignOn.run**, CM is deinstalled, and the user is presented with the SignOn screen. If the partition for the driver is too small for SignOn to run, you can simulate SignOn by making the following call:

```
erc = NotifyCM(3, 0, 0);
```

CM will think that your program is SignOn and log you out.

While CM is installed, the CM screen is always present and updated properly, but the user can see it only by pressing **Action-Go**. This mode of operation is analogous to that of the Debugger screen.

## PROCEDURAL INTERFACES

The following procedural interfaces are described in detail under "Operations," in Chapter 7.

- CMCurrentVersion
- CMQueryConfigFile
- CMQueryContextHandle
- CMQueryErc
- CMQueryParent
- CMSetParent
- CMStartAppl
- CMStartApplByBlock
- CMStartApplByName
- CMSwitchContext
- CMSwitchToExistingContext
- CMTerminateContext
- CMTranslateChToPh
- CMTranslatePhToCh

## *Placing Information in the CM Configuration File*

You can place in the CM Configuration file any information that your program needs to read when it is running. To do so, use the **More (F10)** key in the CM Editor to overlay an editing area for this information on the command editing area. Information should be provided in the syntax

:Field:Value

CM does not check this field for information.

As an example, the Document Designer identifies cooperating programs by use of the following field:

:DDObjectEdited:nnn

where nnn specifies the object type that the current entry edits.

See Chapter 5, "Setting up Context Manager/VM," for a description of the CM Editor.

## *Invoking Programs that Use Run-Time Libraries*

In some languages, for example, versions of Pascal previous to 9.1, a program that is linked with the standard run-time library cannot be invoked directly from CM. You can invoke programs such as these by responding **yes** to the [Needs Exec screen?] field in the CM Editor. Alternatively, you can write a small program without the run-time library that clears the screen and then chains to your program.

## *Using CrashDump.sys as Your Swap File*

It is convenient to use the **CrashDump.sys** file, created when the disk is initialized, as a Swap file because this file is contiguous, and this use of disk space is efficient. (When you initialize your disk using the IVolume utility, you must specify the size of this file; otherwise, its length will be 0.) If you use this file for swapping, be sure that you have an alternate administrative user name on your system that does not load CM at signon. (See "Creating the Swap File," in Chapter 5 and the *CTOS/VM Concepts Manual*.)

After a system failure, the boot ROM writes the contents of memory to **CrashDump.sys** as part of the boot sequence. Suppose you sign on afterward as a user configured to load CM and autostart the Document Designer. You want to start the Executive in order to examine the contents of **CrashDump.sys**. If the system is configured in such a way that the Document Designer must be swapped out at this point, it overwrites the contents of **CrashDump.sys**.

The solution is to sign on after the system failure using the administrative user name, which loads the Executive directly. (Such a user name also enables you to avoid booting the system from a floppy disk in the case of a reproducible system failure while CM or an autostarted application is being loaded.)

## *Estimating Memory Requirements*

The memory requirement of each defined application must be supplied to the CM Configuration File Editor when the system is configured.

For an application that you write, typically you can obtain the Run file size from the Map file generated by the Linker. Find the last entry in the second column of the Map file: This hexadecimal entry is the size in bytes. In the example below, the correct entry, 07D10H, is shown in boldface.

| Start  | Stop          | Length | Name         | Class  |
|--------|---------------|--------|--------------|--------|
| 00000H | 0016CH        | 016DH  | EXBUF_CODE   | CODE   |
| 0016EH | 00516H        | 03A9H  | EXCLOCK_CODE | CODE   |
| .      | .             | .      | .            | .      |
| .      | .             | .      | .            | .      |
| 075C0H | 07D0FH        | 0750H  | STACK        | STACK  |
| 07D10H | 07D10H        | 0000H  | MEMORY       | MEMORY |
| 07D10H | <b>07D10H</b> | 0000H  | ??SEG        |        |

Convert the boldfaced value, above, to decimal.

Examine all calls to AllocMemoryLL and AllocMemorySL, find the total of all memory that the program could allocate while running, and add this total to the above value.

If the Run file is made with the Bind command, take the minimum and maximum data size from the binding. Add the minimum size to the above boldfaced value to get a total minimum, and add the maximum size to get a total maximum. This gives the range of partition sizes in which the program will run.

Divide the result by 1024, rounding up if there is a fractional remainder.

If your program has overlays, the procedure is the same, except that you should take from the Map file the last entry in the second column before the Overlay 0 subheading.

In the case of a program that tunes itself based on the amount of memory available (that is, calls AllocAllMemorySL), the situation is less well-defined. The best approach is to instruct the user to create as large a partition as is practical to accommodate this Run file, and then to identify a Run file size to fit this partition.

## *Determining the Size of a Partition*

When you enter the amount of memory required by an application (in the Memory Required field of the CM Configuration File Editor), you can create one of the following types of partitions:

- a variable sized partition, specified by entering <nnn (where nnn is the size in K bytes) or by entering <0
- a fixed sized partition, specified by entering nnn, or 0 (or leaving a blank space)

### **A VARIABLE SIZED PARTITION**

When you specify you want a variable sized partition by using a less-than sign (<nnn), you are indicating that the size of the partition should not exceed a given maximum amount. CM and the operating system then create a partition based on the sizing information in your Run file. (A sized Version 6 Run file has this information. It specifies the minimum and maximum amount of memory required by your program. To create a Version 6 Run file, you must use the **Bind** command. See "Activating the Linker," in the *Linker/Librarian Manual*.)

*NOTE: Unsized Version 6 Run files and Version 4 Run files do not contain sizing information and will therefore use the maximum amount of memory specified.*

You are also creating a variable sized partition when you specify <0. A partition is created that does not exceed the maximum amount in the Run file or in available memory.

The partition that is created is determined by the following:

- If the maximum size in the Run file is less than the amount you entered (the amount you entered is hereafter called the CM size), a partition is created equal to the maximum size in the Run file.
- If the CM size is less than the maximum size in the Run file, a partition is created equal to the CM size.

For example, suppose you start an Executive that is a sized Version 6 Run file that has a minimum of 120K bytes of memory and a maximum of 200K bytes. You enter <300 in the Memory Required field. In this case, the amount in the Run file overrides the amount you entered, and a partition of 200K bytes is created. If, in this example, the maximum had been 400K bytes instead of 200K, a partition would have been created equal to the CM size of 300K.

## A FIXED SIZED PARTITION

When you specify you want a fixed size partition (nnn), CM and the operating system create a partition based on the amount you entered.

If you enter 0 (or leave the space blank), a partition is created that is equal to the size of available memory when the Run file is loaded.

**Caution:** *If you do not want your programs to use all available memory, use the less-than sign (<0 or <nnn) to specify that a partition be created of a maximum size.*

If you do not have a sized Version 6 Run file, the partition that is created is equal to the CM size.

If you do not have a sized Version 6 Run file and enter <0 or 0, the partition that is created is equal to the size of available memory.

*NOTE: It is recommended that you use properly sized Version 6 Run files: You can create partitions that are more precise and thus run more programs in memory at the same time.*

For more information on variable and fixed partitions, see "Memory Required" under "Screen Areas and Functions," in Chapter 5.

### *Note for the System Service Writer*

On systems that run CM, system services should handle swapping requests as described in the *CTOS/VM Concepts Manual*.

Only those applications that do not have outstanding requests are allowed to swap. In order for applications that have outstanding requests to be able to swap, system services need to be changed. To respond to CTOS swap requests, system services must either finish servicing the application request or return the request to the operating system for queuing when the application swaps back in. Depending on the timing and the ability of the system service to respond to a swap request for a client, an application may at one time be swappable and at another time not swappable. See the *CTOS/VM Concepts Manual* for information on handling swap requests.



This chapter describes procedural interfaces for the CM and ICMS operations. For descriptions of other operations mentioned in this note, see "Operations" in the *CTOS/VM Reference Manual*.

If you want to write code that uses any of the following calls, you must link your Run file with the following two files:

CmCalls.obj  
CmRqLabl.obj

These files resolve references to CM requests and object module operations within your program. They are placed on your hard disk when CM is first installed. (See the Context Manager/VM Release Notice for details about installation.)

---

## CMCurrentVersion

---

CMCurrentVersion (pbVersion, pbRevision): ErcType

### DESCRIPTION

CMCurrentVersion allows a program to determine the current version and revision levels of CM. It checks to ensure that

1. CTOS is Version 9 or greater
2. CM is installed
3. **Request.7.sys** loadable requests are installed

### PARAMETERS

pbVersion is the memory address of a byte into which CM will return the current version.

pbRevision is the memory address of a byte into which CM will return the current revision.

### ERRORS

| Decimal Value | Meaning |
|---------------|---------|
| 12099         | ercNoCM |

*NOTE: If the CM version is earlier than 2.0, CmCurrentVersion returns 0 and 0 as the version and revision.*

---

CMCurrentVersion

(continued)

---

**REQUEST BLOCK**

CMCurrent Version is an object module procedure.

---

## CMQueryConfigFile

---

CMQueryConfigFile (pbBuf, cbBuf, pcbBufRet): ErcType

### DESCRIPTION

The CMQueryConfigFile service allows a program to determine the name of the current CM Configuration file.

The CMQueryConfigFile returns the fully qualified Configuration file name.

### PARAMETERS

pbBuf            describe the buffer where the name of the  
bBuf            Configuration file is to be returned.

pcbBufRet       is the memory address of a word value where the  
                 actual size of the name of the Configuration file  
                 is placed.

### ERRORS

| Decimal Value | Meaning              |
|---------------|----------------------|
| 12007         | ConfigBufferTooSmall |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 2               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 8077h           |
| 12            | reserved     | 6                       |                 |
| 16            | pbBuf        | 4                       |                 |
| 20            | cbBuf        | 2                       |                 |
| 22            | pbBufRet     | 4                       |                 |
| 26            | cbBufRet     | 2                       | 2               |

---

---

## CMQueryContextHandle

---

CMQueryContextHandle (pchRet): ErcType

### DESCRIPTION

The CMQueryContextHandle service allows a program to determine its own context handle. This program can then identify itself to another program by including its context handle in a message.

### PARAMETERS

**pchRet** is the memory address of a word value where the context handle is returned.

### REQUEST BLOCK

---

| Offset | Field     | Size (bytes) | Contents |
|--------|-----------|--------------|----------|
| 0      | sCntInfo  | 1            | 6        |
| 1      | RtCode    | 1            | 0        |
| 2      | nReqPbCb  | 1            | 0        |
| 3      | nRespPbCb | 1            | 1        |
| 4      | userNum   | 2            |          |
| 6      | exchResp  | 2            |          |
| 8      | ercRet    | 2            |          |
| 10     | rqCode    | 2            | 8076h    |
| 12     | reserved  | 6            |          |
| 18     | pchRet    | 4            |          |
| 22     | schRet    | 2            | 2        |

---

---

## CMQueryErc

---

CMQueryErc (pErcRet): ErcType

### DESCRIPTION

The CMQueryErc service is used by a program to retrieve the error code from the preceding call to CMStartApplByName, CMStartApplByBlock, or CMSwitchToExistingContext.

### PARAMETERS

pErcRet is the memory address of a word value where the previous erc is returned.

### REQUEST BLOCK

---

| Offset | Field     | Size (bytes) | Contents |
|--------|-----------|--------------|----------|
| 0      | sCntInfo  | 1            | 6        |
| 1      | RtCode    | 1            | 0        |
| 2      | nReqPbCb  | 1            | 0        |
| 3      | nRespPbCb | 1            | 1        |
| 4      | userNum   | 2            |          |
| 6      | exchResp  | 2            |          |
| 8      | ercRet    | 2            |          |
| 10     | rqCode    | 2            | 807Ch    |
| 12     | reserved  | 6            |          |
| 18     | pErcRet   | 4            |          |
| 2      | sErcRet   | 2            | 2        |

---

---

## CMQueryParent

---

CMQueryParent (ch, pchRet): ErcType

### DESCRIPTION

The CMQueryParent service is used by a program to find out the context handle of the parent of any context.

### PARAMETERS

|        |   |
|--------|---|
| ch     | is the context handle of the context whose parent is desired. If ch is 0, the context handle of the calling context is used.  |
| pchRet | is the memory address of a word value where the context handle is returned. Note that a context handle of zero means that the parent is CM. If a parent context is terminated for any reason, the parent of the child context becomes CM. |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 1               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 8078h           |
| 12            | reserved     | 6                       |                 |
| 18            | pchRet       | 4                       |                 |
| 22            | schRet       | 2                       | 2               |

---

---

## CMSetParent

---

CMSetParent (chToSet, chNewParent): ErcType

### DESCRIPTION

The CMSetParent service is used by a program to change the parent of any context.

### PARAMETERS

**chToSet** is the context handle of the context whose parent is to be changed. If chToSet is 0, the context handle of the calling context is used.

**chNewParent** is the context handle of the new parent context. Note that a context handle of 0 means that the new parent is to be CM.

### ERRORS

| <b>Decimal Value</b> | <b>Meaning</b>         |
|----------------------|------------------------|
| 12003                | ercNoSuchContextHandle |
| 12012                | ercNoSuchParentCh      |
| 12013                | ercNoSelfParent        |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 1               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 8082h           |
| 12            | chToSet      | 2                       |                 |
| 14            | chNewParent  | 2                       |                 |
| 16            | reserved     | 2                       |                 |

---

---

## CMStartAppl

---

CMStartAppl (fName, pbStartData, cbStartData, pVLPB,  
fDependentChild, pchRet): ErcType

### DESCRIPTION

CMStartAppl is called to start a new application. An application can be started in two ways: by name or by block. If an application is started by name, the name supplied must exactly match a name in CM's current Configuration file. If an application is started by block, a block of information must be supplied to CM. This block provides the same information normally supplied by the Configuration file.

### PARAMETERS

|                            |  |
|----------------------------|--|
| fName                      | is TRUE to start an application by name, or FALSE to start an application by a block of information.   |
| pbStartData<br>cbStartData | If fName is TRUE, describe the name of an application that matches one specified in CM's Configuration file.<br><br>If fName is FALSE, describe a block whose definition is given below. |

- pVLPB** is either NIL or is a pointer to a variable length parameter block (VLPB) that CM is to use as the new context's VLPB. Therefore, if pVLPB is non-NIL, then the command case specified in the StartData block is ignored.
- fDependentChild** If true, the child context is marked as dependent on its parent context for survival. That is, if the parent is terminated, the child is also terminated.
- pchRet** is the memory address of a word to which the context handle of the new context is returned.

Definition of the StartData block:

| Offset | Field            | Size<br>(bytes) |
|--------|------------------|-----------------|
| 0      | sbAppName        | 80              |
| 80     | sbRunFile        | 80              |
| 160    | sbAbbrev         | 7               |
| 167    | wMemorySize      | 2               |
| 169    | sbCase           | 3               |
| 172    | bPresetFKey      | 1               |
| 173    | fPresetFKey      | 1               |
| 174    | bAutoStart       | 1               |
| 175    | sbVolume         | 13              |
| 188    | sbDirectory      | 13              |
| 201    | sbPrefix         | 13              |
| 214    | sbPassword       | 13              |
| 227    | sbNode           | 13              |
| 240    | fDirty           | 1               |
| 241    | fNeedsExecScreen | 1               |

where

|             |  |
|-------------|--|
| sbApplName  | is the name of the application to start. The first byte is the size.                 |
| sbRunFile   | is the fully qualified file specification for the Run file to be loaded.             |
| sbAbbrev    | is the abbreviation to be displayed in the CM screen in the associated function key. |
| wMemorySize | is the amount of memory needed to run this Run file.                                 |
| sbCase      | is the case value to be used when invoking the Run file.                             |
| bPresetFKey | is ignored.  |
| fPresetFKey | is ignored.  |
| bAutoStart  | is ignored.  |
| sbVolume    | is the default volume to be used for the new application.                            |
| sbDirectory | is the default directory to be used for the new application.                         |
| sbPrefix    | is the default prefix to be used for the new application.                            |

---

|                  |   |
|------------------|---|
| sbPassword       | is the default password to be used for the new application.   |
| sbNode           | is the default node to be used for the new application.   |
| fDirty           | is TRUE if the application should only run when it is in foreground.  |
| fNeedsExecScreen | is TRUE if the application requires the video to be initialized with frames as though it had been invoked from the Executive. |

*NOTE: Any path information that is not specified is inherited from the parent context.*

**ERRORS**

| <b>Decimal Value</b> | <b>Meaning</b>        |
|----------------------|-----------------------|
| 12004                | ercNoSuchCommand      |
| 12005                | ercSizeTooLarge       |
| 12007                | ercMissingCommandName |
| 12008                | ercCommandNameTooLong |
| 12009                | ercMissingRunFile     |
| 12010                | ercRunFileTooLong     |
| 12011                | ercMissingMemorySize  |

**REQUEST BLOCK**

CMStartAppl is an object module procedure. Depending on the value of fName, CMStartAppl calls either CMStartApplByName or CMStartApplByBlock, and then calls CMQueryErc to return the proper status code.

---

## CMStartApplByBlock

---

CMStartApplByBlock (pbBlock, cbBlock, pVLPB, sVLPB, fDependentChild, pchRet): ErcType

### DESCRIPTION

The CMStartApplByBlock service is used by a program to request that CM start a new application in another partition. This call should be immediately followed by a call to CMQueryErc.

### PARAMETERS

|         |  |
|---------|--|
| pbBlock | describe a block of information about the new application to be started. For the definition of the start block, refer to the documentation for the related routine, CMStartAppl.   |
| cbBlock |  |
| pVLPB   | pVLPB is either NIL or points to a variable length parameter block. NIL implies that CM should create a VLPB as though the user had selected this application from the CM screen. Otherwise, CM copies the VLPB described by pVLPB, sVLPB and uses it as the VLPB for the new application. |
| sVLPB   |  |

**fDependentChild** is true if the child context is marked as dependent on its parent context for survival. If the parent is terminated, the child is also terminated.

**pchRet** is the memory address of a word value to which the context handle of the new application is to be returned.

## **ERRORS**

| <b>Decimal Value</b> | <b>Meaning</b>        |
|----------------------|-----------------------|
| 12005                | ercSizeTooLarge       |
| 12007                | ercMissingCommandName |
| 12008                | ercCommandNameTooLong |
| 12009                | ercMissingRunFile     |
| 12010                | ercRunFileTooLong     |
| 12011                | ercMissingMemorySize  |

## **SEE ALSO**

**CMStartAppl**

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b>    | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|-----------------|-------------------------|-----------------|
| 0             | sCntInfo        | 1                       | 6               |
| 1             | RtCode          | 1                       | 0               |
| 2             | nReqPbCb        | 1                       | 2               |
| 3             | nRespPbCb       | 1                       | 1               |
| 4             | userNum         | 2                       |                 |
| 6             | exchResp        | 2                       |                 |
| 8             | ercRet          | 2                       |                 |
| 10            | rqCode          | 2                       | 8079h           |
| 12            | fDependentChild | 1                       |                 |
| 13            | reserved        | 5                       |                 |
| 18            | pCMInfo         | 4                       |                 |
| 22            | sCMInfo         | 2                       |                 |
| 24            | pVLPB           | 4                       |                 |
| 28            | sVLPB           | 2                       |                 |
| 30            | pchRet          | 4                       |                 |
| 34            | schRet          | 2                       | 2               |

---

---

## CMStartApplByName

---

CMStartApplByName(pbName, cbName, pVLPB, sVLPB,  
fDependentChild, pchRet): ErcType

### DESCRIPTION

The CMStartApplByName service is used by a program to request that CM start a new application in another partition. This call should be immediately followed by a call to CMQueryErc.

### PARAMETERS

|        |   |
|--------|---|
| pbName | describe the name of the application to be started. The name must match the name of an application in CM's known list of applications supplied in its Configuration file.   |
| cbName |   |
| pVLPB  | pVLPB is either NIL or points to a variable length parameter block. NIL implies that CM should create a VLPB as though the user had selected this application from the Context Manager's screen. Otherwise, CM copies the VLPB described by pVLPB, sVLPB and uses it as the VLPB for the new application. |
| sVLPB  |   |

**fDependentChild** is true if the child context is marked as dependent on its parent context for survival. If the parent is terminated, the child also is terminated.

**pchRet** is the memory address of a word value to which the context handle of the new application is to be returned.

## **ERRORS**

| <b>Decimal Value</b> | <b>Meaning</b> |
|----------------------|----------------|
| 12004                | NoSuchAppl     |

## **SEE ALSO**

**CMStartAppl**

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b>    | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|-----------------|-------------------------|-----------------|
| 0             | sCntInfo        | 1                       | 6               |
| 1             | RtCode          | 1                       | 0               |
| 2             | nReqPbCb        | 1                       | 2               |
| 3             | nRespPbCb       | 1                       | 1               |
| 4             | userNum         | 2                       |                 |
| 6             | exchResp        | 2                       |                 |
| 8             | ercRet          | 2                       |                 |
| 10            | rqCode          | 2                       | 807Ah           |
| 12            | fDependentChild | 1                       |                 |
| 13            | reserved        | 5                       |                 |
| 18            | pCMInfo         | 4                       |                 |
| 22            | sCMInfo         | 2                       |                 |
| 24            | pVLPB           | 4                       |                 |
| 28            | sVLPB           | 2                       |                 |
| 30            | pchRet          | 4                       |                 |
| 32            | schRet          | 2                       | 2               |

---

---

## CMSwitchContext

---

CMSwitchContext (ch): ErcType

### DESCRIPTION

CMSwitchContext switches the screen and keyboard to the context specified by the given context handle.

### PARAMETERS

ch is the context handle of the context to switch to.

### ERRORS

| Decimal Value | Meaning  |
|---------------|----------|
| 12003         | NoSuchCh |

### REQUEST BLOCK

CMSwitchContext is an object module procedure. CMSwitchContext calls CMSwitchToExistingContext, and then calls CMQueryErc to return the proper status code.

---

## CMSwitchToExistingContext

---

CMSwitchToExistingContext (ch): ErcType

### DESCRIPTION

The CMSwitchToExistingContext service is used by a program to ask CM to switch the screen and keyboard to an existing context. This call should be immediately followed by a call to CMQueryErc.

### PARAMETERS

ch is the context handle of the context to switch to.

### ERRORS

| Decimal Value | Meaning  |
|---------------|----------|
| 12003         | NoSuchCh |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 0               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 807Bh           |
| 12            | ch           | 2                       |                 |
| 14            | reserved     | 4                       |                 |

---

---

## CMTerminateContext

---

CMTerminateContext (ch): ErcType

### DESCRIPTION

The CMTerminateContext service allows a program to terminate a context in another partition. A program cannot terminate itself or CM in this way.

### PARAMETERS

ch                    is the context handle of the context to be terminated.

### ERRORS

| <b>Decimal Value</b> | <b>Meaning</b> |
|----------------------|----------------|
| 12003                | NoSuchCh       |
| 12006                | CannotKillSelf |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 0               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 807Dh           |
| 12            | ch           | 2                       |                 |
| 14            | reserved     | 4                       |                 |

---

---

## CMTranslateChToPh

---

CMTranslateChToPh (ch, pphRet): ErcType

### DESCRIPTION

The CMTranslateChToPh service allows a program to translate a context handle into a user number.

### PARAMETERS

ch is the context handle to be translated.

pphRet is the memory address of a word value where the translated user number is to be returned.

### ERRORS

| <b>Decimal Value</b> | <b>Meaning</b> |
|----------------------|----------------|
| 12003                | NoSuchCh       |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 1               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 807Eh           |
| 12            | ch           | 2                       |                 |
| 14            | reserved     | 4                       |                 |
| 18            | pphRet       | 4                       |                 |
| 22            | sphRet       | 2                       | 2               |

---

---

## CMTranslatePhToCh

---

CMTranslateuserNumToCh (userNum, pchRet): ErcType

### DESCRIPTION

The CMTranslateuserNumToCh service allows a program to translate a user number into a context handle.

### PARAMETERS

userNum            is the user number to be translated.

pchRet            is the memory address of a word value where the translated context handle is to be returned.

### ERRORS

| <b>Decimal Value</b> | <b>Meaning</b> |
|----------------------|----------------|
| 12001                | NoSuchuserNum  |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 1               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 807Fh           |
| 12            | userNum      | 2                       |                 |
| 14            | reserved     | 4                       |                 |
| 18            | pchRet       | 4                       |                 |
| 22            | schRet       | 2                       |                 |

---

---

## ICMSCheck

---

ICMSCheck (pbMsg, cbMsg, pcbMsgRet): ErcType

### DESCRIPTION

The ICMSCheck service allows a program to check for and possibly receive an arbitrary message sent by another context by means of the Intercontext Message Server (ICMS). If a message is queued waiting for this context, the message is returned.

### PARAMETERS

|           |   |
|-----------|---|
| pbMsg     | describe a buffer for the message supplied by the                                     |
| cbMsg     | calling program.  |
| pcbMsgRet | is the memory address of a word value where the actual size of the message is placed. |

### ERRORS

| Decimal Value | Meaning |
|---------------|---------|
|---------------|---------|

|       |  |
|-------|--|
| 12103 | ercNoMsgAvailable<br>There is no message available for this context. |
| 12106 | ercMsgTooLong<br>The message is bigger than the area supplied.       |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 2               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 8011h           |
| 12            | reserved     | 6                       |                 |
| 18            | pbMsg        | 4                       |                 |
| 22            | cbMsg        | 2                       |                 |
| 24            | pcbMsgRet    | 4                       |                 |
| 28            | cbMsgRet     | 2                       | 2               |

---

---

## ICMSCurrentVersion

---

ICMSCurrentVersion (pbVersion, pbRevision): ErcType

### DESCRIPTION

The ICMSCurrentVersion routine allows a program to determine the current version and revision levels of ICMS. It checks to ensure that

1. CTOS is Version 9 or greater
2. Request.7.sys is installed

### PARAMETERS

pbVersion is the memory address of a byte into which ICMS will return the current version.

pbRevision is the memory address of a byte into which ICMS will return the current revision.

### ERRORS

| Decimal Value | Meaning   |
|---------------|-----------|
| 12108         | ercNoICMS |

### REQUEST BLOCK

ICMSCurrentVersion is an object module procedure.

---

## ICMSFlush

---

ICMSFlush (ch): ErcType

### DESCRIPTION

The ICMSFlush service allows a program to flush any messages that may be waiting in the Intercontext Message Server (ICMS) for a given context. This call might be used, for example, when program A in a given partition chains to program B. Program B might initially flush waiting messages intended for A.

### PARAMETERS

ch is the context handle of the context whose messages are to be flushed.

### REQUEST BLOCK

---

| Offset | Field     | Size (bytes) | Contents |
|--------|-----------|--------------|----------|
| 0      | sCntInfo  | 1            | 6        |
| 1      | RtCode    | 1            | 0        |
| 2      | nReqPbCb  | 1            | 0        |
| 3      | nRespPbCb | 1            | 0        |
| 4      | userNum   | 2            |          |
| 6      | exchResp  | 2            |          |
| 8      | ercRet    | 2            |          |
| 10     | rqCode    | 2            | 8012h    |
| 12     | ch        | 2            |          |
| 14     | reserved  | 4            |          |

---

---

## ICMSSend

---

ICMSSend (ch, pbMsg, cbMsg): ErcType

### DESCRIPTION

The ICMSSend service allows a program to send an arbitrary message to another context by means of the Intercontext Message Server (ICMS).

### PARAMETERS

ch is the context handle of the context where the message is to be sent.

pbMsg describes the message to be sent.  
cbMsg

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 1               |
| 3             | nRespPbCb    | 1                       | 0               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 800Fh           |
| 12            | ch           | 2                       |                 |
| 14            | reserved     | 4                       |                 |
| 18            | pbMsg        | 4                       |                 |
| 22            | cbMsg        | 2                       |                 |

---

---

## ICMSWait

---

ICMSWait (pbMsg, cbMsg, pcbMsgRet): ErcType

### DESCRIPTION

The ICMSWait service allows a program to wait for an arbitrary message sent by another context by means of the Intercontext Message Server (ICMS).

### PARAMETERS

|                |   |
|----------------|---|
| pbMsg<br>cbMsg | describe a buffer for the message supplied by the calling program.                    |
| pcbMsgRet      | is the memory address of a word value where the actual size of the message is placed. |

### ERRORS

| Decimal Value | Meaning   |
|---------------|---|
| 12106         | MsgTooLong<br>The message is bigger than the area supplied. |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 0               |
| 3             | nRespPbCb    | 1                       | 2               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 8010h           |
| 12            | reserved     | 6                       |                 |
| 18            | pbMsg        | 4                       |                 |
| 22            | cbMsg        | 2                       |                 |
| 24            | pcbMsgRet    | 4                       |                 |
| 28            | cbMsgRet     | 2                       | 2               |

---

---

## NotifyCM

---

NotifyCM (msgType, pbMsg, cbMsg): ErcType

### DESCRIPTION

NotifyCM passes information from a context to CM, stating a condition of that context or asking for action by CM.

### PARAMETERS

msgType is a word value code for one of the following messages:

- 0 Test to see if CM is installed. Returns status code 0 (ercOK) if CM is installed and status code 33 if CM is not installed. (This code also works in CM Versions 1.0 and 1.1.)
- 1 Terminate this context and pass an erc (sent by CmNull).
- 2 New command (sent by the Executive).
- 3 Logout (sent by SignOn).
- 4 Terminate this context if you want (sent by the Executive).

- 5 Change the parent of the calling context to be CM (see "Communication Between Applications," in Chapter 6).
- 6 Graphics application (sent automatically by all graphics applications).
- 7 Terminate this context and pass an error message (sent by CmNull).
- 8 This context can run in background (clean) (sent by any application).
- 9 This context cannot run in background (dirty) (sent by any application).
- 10 This context is no longer using graphics.
- 11 Not currently used.
- 12 Not currently used.
- 13 Lock this context from user access.
- 14 Unlock this context from user access.
- 15 Terminate this context when its parent terminates.

- 16 Do not terminate this context when its parent terminates.
- 17 Log out unconditionally.
- 18 Change the label of the function key appearing on the CM screen.

pbMsg/cbMsg describes the appropriate message. These are only meaningful to CM for messages of types 1, 2, 4, 7, and 18. Values 8 and 9 deserve further explanation: Depending on the sequence of calls that a program makes to video routines, CM marks that context as clean or dirty (able or unable to run in background). Any program can override this marking by including an explicit call to NotifyCM with the values 8 or 9.

## ERRORS

| Decimal Value | Meaning       |
|---------------|---------------|
| 12000         | ercUnknownMsg |

**REQUEST BLOCK**

---

| <b>Offset</b> | <b>Field</b> | <b>Size<br/>(bytes)</b> | <b>Contents</b> |
|---------------|--------------|-------------------------|-----------------|
| 0             | sCntInfo     | 1                       | 6               |
| 1             | RtCode       | 1                       | 0               |
| 2             | nReqPbCb     | 1                       | 1               |
| 3             | nRespPbCb    | 1                       | 0               |
| 4             | userNum      | 2                       |                 |
| 6             | exchResp     | 2                       |                 |
| 8             | ercRet       | 2                       |                 |
| 10            | rqCode       | 2                       | 102h            |
| 12            | msgType      | 2                       |                 |
| 14            | reserved     | 4                       |                 |
| 18            | pbMsg        | 4                       |                 |
| 22            | cbMsg        | 2                       |                 |

---



In this chapter, common errors that occur when you are using CM are described, along with suggestions as to what to do about them.

In most cases, a screen message reports the error. Each such error is described under its message on the following pages.

## *Status Codes*

### **CONTEXT MANAGER/VM**

The following errors are most commonly reported during software development.

| Decimal Value | Meaning  |
|---------------|--|
| 12000         | CM has received an unknown message.  |
| 12001         | No such user number.   |
| 12002         | The Configuration file name is bigger than the buffer that was allocated.      |
| 12003         | No such context handle.  |
| 12004         | The application specified does not match one listed in the Configuration file. |
| 12005         | The size of the application information block is too large.                    |
| 12006         | A context cannot kill itself.  |
| 12007         | Using start by block, missing command name.                                    |
| 12008         | Using start by block, command name too long.                                   |
| 12009         | Using start by block, missing Run file.  |
| 12010         | Using start by block, Run file too long.                                       |
| 12011         | Using start by block, missing memory size.                                     |

---

| <b>Decimal Value</b> | <b>Meaning</b> |
|----------------------|----------------|
|----------------------|----------------|

---

The following errors are reported during normal execution of a program that uses CM as a server.

|       |   |
|-------|---|
| 12084 | The Run file needed for this application does not exist.              |
| 12085 | There are no more function keys available.                            |
| 12086 | There are no active contexts.   |
| 12087 | The Swap file is full—cannot swap any more contexts.                  |
| 12088 | The Run file is too large to run in any partition.                    |
| 12089 | You cannot logout; there are context(s) that must be finished.        |
| 12090 | Warning: There are active contexts.                                   |
| 12091 | This context cannot be finished from CM.                              |
| 12092 | The file specified as the Swap file does not exist.                   |
| 12093 | You can run only one graphics application at a time.                  |
| 12094 | Not currently used.   |
| 12095 | There is no current context.  |
| 12096 | An existing context cannot be swapped out to start a new application. |
| 12097 | Not enough room in the Swap file to switch contexts.                  |
| 12098 | You cannot switch to this context—it is currently locked.             |
| 12099 | CM is not installed.  |

## INTERCONTEXT MESSAGE SERVER

---

| Decimal Value | Meaning   |
|---------------|---|
| 12100         | ICMS is already installed.                          |
| 12101         | ICMS internal error.                                |
| 12102         | Cannot open the ICMS Disk Message file.             |
| 12103         | No message available.                               |
| 12104         | No free messages.                                   |
| 12105         | Not implemented.                                    |
| 12106         | The message sent was too long.                      |
| 12107         | This context is already waiting for messages.       |
| 12108         | ICMS is not installed.                              |
| 12109         | Incorrect version or missing <b>Request.7.sys</b> . |

## *Status Messages*

Status messages are described on the following pages.

---

**Not enough memory available.**  
(Error 400)

---

## **EXPLANATION**

When you select an application to start and press **Go**, the CM message area says, "Loading . . .," then, "Finishing . . .," and then gives the above error.

This series of messages occurs when CM tries to start an application in a partition that is too small for the application. The error is in the CM Configuration file: The number specified in the Memory Required field of the command editing area is too small.

Suppose you edit the Memory Required field to state that the Executive requires 50K bytes to run, when actually it requires at least 110K bytes. When CM tries to start the Executive in that 50K-byte partition, these messages are displayed.

## **OPERATOR RESPONSE**

Use the **CM Config File Editor** command to edit your CM Configuration file. Bring up the offending application, and change the entry in the Memory Required field to match the value given in the Context Manager/VM Release Notice. If you are using a sized Version 6 Run file, you may want to specify a less than amount (<nnn or <0). Save the Configuration file changes, log out, and reinstall CM.

---

**This version of the OS cannot support any more contexts.**  
(Error 801)

---

#### **EXPLANATION**

When you try to start a new application from the CM screen, the above message is displayed. The operating system has run out of user numbers.

#### **OPERATOR RESPONSE**

Refer to the Context Manager/VM Release Notice for an explanation of how to generate a new version of the operating system that supports more contexts.

---

**A context in memory cannot be swapped out.**  
(Error 813)

---

## **EXPLANATION**

A context in memory cannot be swapped out because it cannot be quieted; that is, the program has requests outstanding after swapping requests have been issued by the operating system. This error is usually caused by servers that do not handle swapping correctly.

## **OPERATOR RESPONSE**

Either wait until the context in memory is finished, or finish the context that is in memory, and then try to switch your context again.

*NOTE: Status codes 813 and 815 have the same status message.*

---

**A context in memory cannot be swapped out.**  
(Error 815)

---

#### **EXPLANATION**

A context in memory cannot be swapped out because it has served a request, served interrupts, or has a communication or parallel printer resource in use.

#### **OPERATOR RESPONSE**

Either wait until the context in memory is finished, or finish the context that is in memory, and then try to switch your context again.

*NOTE: Status codes 813 and 815 have the same status message.*

---

**This application must be invoked through the Exec;  
edit the Config File.**

---

## **EXPLANATION**

The application you have chosen requires that you enter a parameter or parameters that should be supplied from the Executive, by means of an Executive command form.

## **OPERATOR RESPONSE**

Use the CM Editor to edit your CM Configuration file. Bring up the offending application. Change the entry in the Run file field to be [Sys]<Sys>Exec.run. Change the entry in the [Command case] field to be CM. Save the Configuration file changes, log out, and reinstall CM.

---

**An existing context cannot be swapped out to start a new application.**

---

### **EXPLANATION**

If you select a new application and press **Go**, and the above message is displayed, all memory partitions into which the new application would fit are already occupied by contexts that are not allowed to swap out.

### **OPERATOR RESPONSE**

Either wait until the context(s) in memory are finished, or finish a context that is in memory, and then try to start your application again.

---

**You cannot switch to this context – it is currently locked.**

---

## **EXPLANATION**

You select a context by pressing **Action-fn** or from the CM screen, and the above error message is displayed.

The context that you chose marked itself as "locked from user interaction." You cannot switch to it directly from CM. This context may have been started by another (parent) context and be accessible only through its parent.

As an example, suppose you start the Business Graphics Package from the Document Designer. When you finish your work with the Business Graphics Package, it marks itself as locked to show that it is there only so it will be available if you need to go to it from the Document Designer again.

**Active.** An application that has been started under CM (a context) is said to be active.

**Application.** An application is a program with which a user can carry out tasks such as word processing or financial spreadsheet operations. An application is not a context until it has been started by CM.

**Application character map.** The application character map is an array where lines of the screen are stored when an application is running in background.

**Autostart.** A CM configuration in which the first display after you sign on is that of an application.

**Available memory.** Available memory is the amount of memory that is available to the application program after the operating system, CM, and all system services have been loaded.

**Background.** This indicates a context that is still running, but is not the active context.

**Border.** See **Window border**.

**Bound button.** The Bound button is one of the Mouse buttons. It is used to change the size of a window.

**Bullet.** On the CM screen, the bullet is the small dot indicator to the left of the name of the current context under "Contexts you can return to."

**Click.** Click is a Mouse button term. To click a Mouse button, press it down and release it immediately.

**CM Configuration file.** The CM Configuration file contains information required by the system to run CM for a given user.

**Command.** A command is an instruction that performs a specific action. The Windows menu contains a list of commands that allow you to move and size a window.

**Command case.** The command case is a two-character parameter that is passed to an application to tell it which command has been given, and thus what process should be done.

**Conforming program.** Under Window Services, a conforming program uses only the Video Access Method (VAM) to write to the display.

**Configure.** To configure a system is to set it up, specifying certain parameter values or characteristics to complete the information that is needed so that it can run.

**Context.** A context is an application that has been started under CM.

**Context handle.** A context handle is a unique identifier that is assigned to an application when the application is first loaded into a vacant partition.

**Current context.** The current context is the context you can see on the screen and interact with.

**Current window.** The current window is the window with which you can interact. It is the only window that can be sized and moved. If the current window has a window border, it will be brighter than the other window borders on the screen.

**Default value.** A default value is a value assigned automatically if you do not specify one.

**Done.** This status term indicates that the last command you invoked in the Executive has been completed.

**Drag.** Drag is a Mouse button term. To drag a Mouse, hold down a button and move the Mouse at the same time.

**Eliminating a context.** To eliminate a context, choose it in the list of "Contexts you can return to" and press **Action-Finish**. This method discards work done in this context in this session.

**Exit Run file.** An Exit Run file is a user-specified file that is loaded and activated when an application system exits. Each application partition has its own Exit Run file.

**Finishing a context.** To finish a context is to carry out the usual steps to finish and save within that application.

**Fixed partition.** A fixed partition is a space of a fixed size that is created in memory for a specific application. The size of this space is determined by the amount of memory specified in the Memory Required field of the CM Configuration File Editor. When you specify 0 or leave the space blank, a partition is created that is equal to the size of available memory.

**Foreground.** The current context, which owns the screen, keyboard, and Mouse (if you have one) runs in foreground.

**Full.** A command that appears on the Windows menu. When the Full command is selected, a window is created that is the size of the full screen. This window does not have a border or title bar.

**Function keys.** The function keys are keys **F1** through **F10** across the top of the keyboard. In CM, they can either be assigned during a session or preassigned to applications.

**Halted.** This status term indicates that all processes of the current context have been stopped.

**Highlighted.** A screen area, word, or character that is highlighted appears brighter than other objects on the screen.

**Icon.** An icon represents an application whose window has been greatly reduced. It is also the name of a command on the Windows menu. When the **Icon** command is selected, an icon is created that appears at the bottom of the screen over its assigned function key. The application represented by the icon may still be running in background, or it may be swapped out.

**Install.** The term "install" has two meanings: to place software on a system, and to load a program from disk storage to workstation memory. In this manual, the second meaning has been used throughout.

**Intercontext Message Server (ICMS).** Applications that are running in separate partitions can communicate with each other by passing messages using ICMS. Under ICMS, a message is copied into ICMS' memory space, where it waits for the receiver to claim it.

**Interprocess Communication (IPC).** IPC is a facility of the operating system that allows communication between processes in a way that is similar to the Intercontext Message Server. But under IPC, only the address of a message is transferred.

**Mark button.** The **Mark** button is one of the Mouse buttons. It is used to select a command on the Windows menu and move a window to a new position.

**Menu.** A menu lists various command options, from which you can make one choice. When an option is chosen from the Windows menu, a window can be sized and moved.

**Menu button.** The **Menu** button is the middle button on the Mouse. It is used to bring up the Windows menu.

**Mouse.** A Mouse is an electronic pointing device that plugs into the keyboard. Moving the Mouse moves a cursor on the screen. The Mouse also has three buttons that manipulate windows in various ways.

**Mouse Services.** The Mouse Services is a system service that allows you to use a Mouse to select items from the CM screen and manipulate windows.

**Move.** A command that appears on the Windows menu. When the **Move** command is selected, the window is moved to a new location. The size of the window does not change.

**Nonconforming program.** Under the Windows Services, a non-conforming program "breaks" the rules by using a pointer to the video map. (That is, it uses a GetpStructure to get access to rgpVidMemLine.)

**Partition.** A partition is a defined portion of workstation memory.

**Point.** Point is a Mouse term. To point a Mouse is to move it until the cursor rests on the selected area.

**Preassigned function key.** A preassigned function key is one assigned to an application through the CM Editor. Preassigned keys appear on the function key menu whenever CM is installed.

**Press.** Press is a Mouse button and keyboard term. When used in reference to the keyboard, it means to press and release a key. When used in reference to the Mouse, it means to hold down a Mouse button.

**Real-time applications.** Real-time applications are applications in which response to input is fast enough to affect subsequent input, such as a process control system or a computer-assisted instruction system.

**Restore.** A command that appears on the Windows menu. When the **Restore** command is selected, the window is restored to the size and position it was in before the **Full** or **Tile** command was executed.

**Reverse video.** This term indicates a reversal of brightness and darkness in order to highlight a screen area, word, or character.

**Run file.** A Run file is a disk file that contains a program in a form that can be read into memory by the operating system loader when the program is to be run.

**Running.** This status term indicates that the context shown to its right is running.

**Sized Version 6 Run file.** A sized Version 6 Run file is a type of Run file that has been created with the **Bind** command and contains sizing information from the minimum and maximum data fields.

**Status terms.** Status terms, which appear to the left of the items in the list of "Contexts you can return to," tell whether each context is Running, Waiting, Done, Swapped, or Stopped.

**Stopped.** This status term indicates that the context shown to its right is in background, but is not running because it is allowed to run only in foreground.

**Suspended.** A context that has been swapped to disk (stored temporarily in a hard disk file) is said to be suspended because it is not actually running.

**Swapped.** This status term indicates that the context shown to its right has been stored temporarily on a hard disk. This context is active although suspended.

**Swap to disk.** To swap a context to disk is to place it temporarily in a designated Swap file on a hard disk. A swapped context is still active (can be recalled with a keystroke) but is suspended (not actually running).

**Switching contexts.** To switch contexts is to press certain keys to cause one context to be replaced on the screen by another.

**Tile.** A command that appears on the Windows menu. When the **Tile** command is selected, the screen is divided among the number of contexts that are currently active.

**Title bar.** A title bar is a bar that appears in the upper left-hand corner of the window border. It displays (in reverse video) the name of the application that is in the window.

**Type-ahead buffer.** Characters that cannot yet be shown on the screen are stored in the type-ahead buffer until it becomes possible to display them.

**User Configuration file.** The User Configuration file contains general information needed to run the system for a given user.

**Variable sized partition.** A variable sized partition is created when you enter a less-than sign before a numeral or a zero (<nnn or <0) in the Memory Required field of the CM Configuration File Editor. The less-than sign indicates that the partition can shrink and grow (up to the maximum specified) based on information about its memory needs. This information is stored in the Run file.

**Video Access Method (VAM).** The Video Access Method is a set of system-common procedures that is part of the installable video or Windows system service. These procedures provide direct access to the characters and attributes of each frame in the application character map. VAM procedures can put characters anywhere in a frame, scroll a frame up or down a specified number of lines, position a cursor in a frame, and reset a frame.

**Video Display Management (VDM).** Video Display Management is that part of the installable video or Windows system service that handles requests directed to the video exchange. VDM provides direct control over the video hardware. With it, an application system can determine the level of video capability, load a new character font into the font RAM, change screen attributes, stop video refresh, calculate the amount of memory needed for the character map based on the desired number of columns and lines and the presence or absence of character attributes, initialize each of the frames, and initialize the character map.

**Video pointer map.** The video pointer map is an array of pointers, one for each line of the screen, that always point to the location of the associated line of the application's screen.

**Waiting.** This status term indicates that the context shown to its right is waiting for input.

**Window.** A window is the rectangular portion of the screen within which an application's character map is displayed. If the Window Services program has been installed, some windows are surrounded by borders and have title bars. These windows can be sized and moved.

**Window-aware program.** A window-aware program uses only the Video Access Method to access the display. The program is aware that the size of its window has changed and reformats itself to fit into the new size.

**Window border.** A window border is a box that surrounds the window. The border has a title bar on top that displays the name of the context that is in the window.

**Windows menu.** The Windows menu is a small box that "pops up" on the screen and displays Window commands. These commands allow a window to be moved and sized.

**Window Services.** The Window Services system service allows multiple contexts to be visible on the screen at once. Each context "owns" a portion of the screen called a window. Under Window Services, the windows can be moved and sized.

- Abbreviation field in CM Editor, 5-16, 5-23
- Action keys, 1-5, 2-5
- Action-Finish, 2-12, 2-5
- Action-fn, 2-10, 2-5, 4-8
- Action-Go, 2-5
- Action-Minus, 2-10, 2-5, 4-8, 3-5
- Action-Next, 2-10, 2-5, 3-5, 4-8
- Action-S, 2-5, 3-6
- Action-W, 4-30
- Active context, 1-4
- Adding an application. See **CM Add Application**.
- Application character map, 6-2 to 6-3
- Applications
  - adding to a configuration file, 5-26 to 5-28
  - dirty. See **Dirty programs**.
  - estimating memory requirements of, 6-17 to 6-21
  - invoked indirectly from CM through the Executive, 5-28 to 5-29
  - not requiring parameters, 5-28
  - not swapped, 3-8 to 3-9, 6-8
  - removing from a configuration file, 5-28
  - requiring parameters, 3-4, 5-17, 5-28 to 5-29
  - starting from CM, 5-21 to 5-22
    - example, 5-28 to 5-29
    - second one, 2-9
  - suspended in background, 6-5 to 6-6
  - writing directly to the screen, 6-2 to 6-3, 6-5 to 6-6
- Arrow keys, 2-4
- Assigning function keys, 3-2 to 3-4
- Autostart, 3-1, 5-18, 5-24, 6-14 to 6-15
- Autostart Ordering field in CM Editor, 5-18, 5-24
- Background, 1-2, 1-5, 5-12, 6-4, 6-6
- Bibliography, 1-5 to 1-8
- Bind command, 6-18
- Border in Windows, 4-7
- Bound Mouse button, 4-14
- Bullet, 2-8, 3-6
- Busy wait loops, 6-4
  
- Cancel
  - in CM, 2-9, 3-6
  - in CM Editor, 5-11, 5-22
- Cancelling Windows menu using the keyboard, 4-30
- using the Mouse, 4-15, 4-17
- Changes
  - deinstalling CM to record changes, 5-22
- Check for installation, 6-14
- Clean programs, 6-5
- Clicking the Mouse, 2-1, 4-13
- CM
  - definition, 1-1
  - features, 1-3
  - operating system, 6-1
  - server, 6-11
  - uses, 1-1 to 1-2
- CM command on Windows menu, 4-5
- CM Add Application, 5-26 to 5-28
- CM Configuration File Editor. See **CM Editor**.
- CM Configuration file, 6-16
- CM Editor
  - adding a Submit command, 5-29 to 5-31
  - exiting, 5-22
  - making an entry in, 5-22 to 5-24, 5-12
  - starting from the Executive, 5-6 to 5-7
- CM Remove application, 5-28
- CMCurrentVersion, 6-14, 7-2 to 7-3

- CMNull.run, 6-8
- CMQueryConfigFile, 7-4 to 7-5
- CMQueryContextHandle, 7-6
- CMQueryErc, 7-7
- CMQueryParent, 6-12, 7-8 to 7-9
- CMSetParent, 7-10 to 7-11
- CMStartAppl, 6-13, 7-12 to 7-16
- CMStartApplByBlock, 7-17 to 7-19
- CMStartApplByName, 7-20 to 7-22
- CMSwitchContext, 6-13, 7-23
- CMSwitchToExistingContext, 7-24 to 7-25
- CMTerminateContext, 7-26 to 7-27
- CMTranslateChToPh, 7-28 to 7-31
- Color field in CM Editor, 5-18
- Command Case, 5-17 to 5-18, 5-23
  - CM defined as, 5-29
- Command editing area in CM Editor, 5-11
- Command list area in CM Editor, 5-21 to 5-22
- Command Name field in CM Editor, 5-12
- Commands. See also **Executive commands**.
  - creating, 5-11 to 5-22
  - editing, 5-11 to 5-22, 5-25
  - name length, 5-12
  - removing, 5-12, 5-25 to 5-26
  - renaming, 5-12, 5-26
  - vs. applications, 5-12
  - on Windows menu, 4-4 to 4-6. See also **Windows menu commands**.
- Communication
  - between applications, 6-9 to 6-10
  - with CM, 6-9
- Communications applications and swapping, 3-9, 6-8
- Configuration
  - CM, 5-1
  - example, 5-32
  - Mouse
    - for cursor speed, 5-5 to 5-6
    - for left-hand use, 5-5 to 5-6
- Conforming programs, 6-6 to 6-8
- Context handle, 6-9, 6-12 to 6-13
- Contexts
  - copying from one to another, 3-9 to 3-13
  - current, 1-4. See also **Current window**.
  - definition, 1-4
    - halting, 3-6
    - number of, 1-4, 5-6, 5-33 to 5-34
    - parent/child relationships, 6-11, 6-15
    - returning to, 2-8 to 2-9, 3-5
    - starting and switching, 1-5, 6-13
    - swapped, 3-8 to 3-9, 6-8
- Copying
  - one context to another, 3-9 to 3-13
- CrashDump.sys as a swap file, 5-2, 6-17
- Create (f5), 5-10, 5-12, 5-23
- Creating a command, 5-11 to 5-22
- Current window, 1-4, 4-7 to 4-8
  - returning to, 4-8, 4-31
  - seeing what is behind, 4-19
  - with the keyboard, 4-31 to 4-35
  - with the Mouse, 4-16
- Cut and Paste, 3-9 to 3-13
- Deinstallation of CM, 5-4, 5-22
- Directory field in CM Editor, 5-17, 5-24
- Dirty programs, 5-12, 6-5 to 6-6
- Done, 2-6, 3-6 to 3-7
- Dragging the Mouse, 4-13, 4-18
- Editing a command, 5-11 to 5-22
  - example, 5-25
- Editing the User Configuration file, 5-4 to 5-5
- Eliminating a context, 2-12, 3-13
- Errors. See **Troubleshooting**.
- Examples. See **Exercises**.
- Executive, 3-4, 3-14
  - commands
    - Bind, 6-18
    - New Command, 5-29
    - Partition Status, 5-15
    - Submit, 5-29 to 5-31
  - Exit Run file, 6-8
  - finishing, 2-11, 3-13
  - invoking applications from, 5-28 to 5-29
  - starting the CM Editor, 5-6 to 5-7

- Exercises, using
  - CM Editor, 5-22 to 5-26
  - CM, 1-2, 2-6 to 2-12
  - Windows and the Mouse, 4-17 to 4-29
- Exit Run file, 6-8
- Exiting CM Editor, 5-22
- Finishing
  - in CM Editor, 5-31
  - in Windows, 4-16
- Finishing a context, 2-11, 3-13
- Fixed sized partitions, 5-14, 6-20 to 6-21
- Foreground, 1-2, 1-5, 6-4
- Full command, 4-5, 4-28, 4-41 to 4-42
- Full window, 4-7
  - activating Windows menu with the Mouse, 4-16, 4-22
  - creating using the keyboard, 4-41
  - creating with the Mouse, 4-28 to 4-29
- Function Key field in CM Editor, 5-16, 5-23
- Function key menu, 2-8
  - in CM, 3-2
  - in CM Editor, 5-19
  - definition, 3-2
- Function keys, 3-2 to 3-4
  - More (f10), 6-16
  - preassigned, 3-2 to 3-4, 5-16
- GetPartitionHandle, 6-12
- GetpStructure, 6-2 to 6-3, 6-6, 6-7
- Halted, 2-6, 3-6 to 3-7
- Help key, 5-4
- Highlight bar
  - in CM, 2-7
  - in CM Editor, 5-11
- ICMS. See **InterContext Message Server**.
- ICMS function key, 5-10, 5-20 to 5-21
- ICMSCheck, 7-32 to 7-33
- ICMSCurrentVersion, 6-14, 7-34
- ICMSFlush, 7-35
- ICMSSend, 7-36 to 7-37
- ICMSWait, 7-38 to 7-39
- Icon command, 4-5, 4-25 to 4-28, 4-38 to 4-41
- Iconic windows
  - creating using the keyboard, 4-38 to 4-41
  - creating with the Mouse, 4-25 to 4-28
  - deselecting, 4-16, 4-31
  - that are not visible, 4-27, 4-40
- Input/error line in CM Editor, 5-11
- Installation, 3-1, 5-3 to 5-4
  - check for, 6-14
- Intercontext Message Server (ICMS), 5-20 to 5-21, 6-9 to 6-10
- Interprocess Communication facility (IPC), 6-9
- Key combinations, 1-5, 2-5
- Keyboard
  - cancelling Windows menu with, 4-30
  - moving a window, 4-35, 4-30
  - opening a window, 4-31 to 4-32
  - selecting Windows menu commands, 4-32
  - sizing a window, 4-30, 4-32 to 4-35
- LockVideo, 6-2 to 6-3
- Logging out, 2-12, 3-14, 4-16, 4-31, 5-4, 5-22
- Low-memory interrupt vector table, 6-4

- Mapping of video lines, 6-2 to 6-3
- Mark Mouse button, 2-1, 2-4, 4-8
  - description, 4-14
  - return to CM screen with, 4-15
- Memory requirements, 6-17 to 6-21
- Memory required field in CM Editor, 5-13 to 5-15, 6-19 to 6-21
- Menu. See **Windows menu**, **Function key menu**.
- Message area in CM, 2-3, 2-9
- Message line in CM Editor, 5-8
- More (f10), 5-10, 5-21, 6-16
- Mouse
  - buttons, 4-10
    - bound, 4-14
    - mark, 2-1, 2-4, 4-14, 4-8
    - menu, 4-4, 4-14
    - using, 4-13 to 4-15, 4-18
  - cancelling Windows menu with, 4-15, 4-17
  - clicking, 2-1, 4-13
  - configuring
    - cursor speed, 5-5 to 5-6
    - left-hand use, 4-11 to 4-12, 5-5 to 5-6
  - cursor, 2-1, 4-10, 4-13
  - definition, 4-9
  - installing, 4-11
  - left-handed use, 4-11 to 4-12, 5-5 to 5-6
  - moving, 4-11 to 4-12
  - moving a window with, 4-15, 4-21 to 4-22
  - opening a window with, 4-17
  - regulating cursor speed, 4-12
  - right-handed use, 4-11 to 4-12
  - selecting Windows menu commands with, 4-15
  - sizing a window with, 4-18 to 4-20
  - starting an application with, 2-1 uses of, 4-10
- Mouse Services, 4-1
- Move command, 4-6
- Moving
  - mouse, 4-11 to 4-12
  - window, 4-8
    - using the keyboard, 4-35
    - with the Mouse, 4-15, 4-21 to 4-22
- Needs Exec screen field in CM Editor, 5-18, 5-24
- New Command command
  - creating, 5-29
- Node field in CM Editor, 5-17, 5-24
- Nonconforming programs, 6-6 to 6-8
- Nonwindow-aware programs, 4-7, 4-20, 6-7 to 6-8
- NotifyCM, 6-9, 7-40 to 7-43
- Number of contexts, 1-4, 5-6, 5-33 to 5-34
- Opening a window
  - with the Mouse, 4-17
  - with the keyboard, 4-31 to 4-32
- Operating system and CM, 6-1
- Operations, 6-2 to 6-3, 7-1 to 7-43. See also individual listings for each operation.
- Parameters
  - applications that require, 3-4, 5-17
- Parent/child relationships of contexts, 6-11, 6-15
- Partitions
  - creating, 5-13 to 5-15, 6-19 to 6-22
  - sizes, 5-13 to 5-15, 6-19 to 6-22
    - fixed, 5-14, 6-20 to 6-21
    - variable, 5-13 to 5-15, 6-19 to 6-21
- Partition Status command, 5-15
- Password field in CM Editor, 5-17
- Pasting, 3-9 to 3-13
- Pausing a context, 3-6
- Pointing the Mouse, 4-13
- Positioning cursor, 6-5
- Preassigned function keys, 3-3 to 3-4
- Prefix field in CM Editor, 5-17, 5-24
- Pressing the Mouse, 4-13

- Programs
  - clean, 6-5
  - compatible with CM, 6-1 to 6-2
  - dirty, 5-12, 6-5 to 6-6
  - nonwindow-aware, 4-7, 4-20, 6-7 to 6-8
  - requiring alteration, 6-1 to 6-2
  - that run under Windows, 6-6 to 6-8
    - conforming and nonwindow-aware, 6-7 to 6-8
    - conforming and window-aware, 6-7
    - nonconforming and nonwindow-aware, 4-7, 4-20, 6-7 to 6-8
    - window-aware, 6-7 to 6-8
- Real-time applications and swapping, 3-9, 6-9
- Remove (f6), 5-20, 5-10, 5-12
- Removing a command, 5-25
- Removing an application. See **CM Remove Application.**
- Rename (f7), 5-10, 5-12, 5-20, 5-26
- Renaming a command, 5-26
- Request.7.sys, 6-14, 7-2
- Restore command, 4-1 to 4-2, 4-5, 4-28 to 4-29
- Restored window
  - creating using the keyboard, 4-41 to 4-42
  - creating with the Mouse, 4-28 to 4-29
- Returning to current context, 2-8 to 2-9, 3-5, 4-8
- Returning to current window, 4-31
- Run File field in CM Editor, 5-23
- Run File Name field in CM Editor, 5-13
- Run files
  - sizes of, 5-15
- Run-time libraries
  - programs that use, 6-16
- Running, 2-6, 3-6 to 3-7
  - messages, 2-9
  - full-screen window, 4-7
  - ownership of, 1-5
  - using CM without CM screen, 6-14 to 6-15
- Screen map
  - programs that write to, 6-2 to 6-3
- Server, 6-11
- SetCommIsr, 6-8
- SetIntHandler, 6-4 to 6-5, 6-8
- SetPartitionName, 6-12
- SetSwapDisable, 6-8
- SetTrapHandler, 6-4 to 6-5
- Show (f3), 5-10, 5-16, 5-19
- SignOn.run, 6-14
- Size commands, 4-6
- Sizes of partitions. See **Partition sizes.**
- Sizes of Run files, 5-15
- Sizing a window, 4-8
  - using the keyboard, 4-32 to 4-35
  - with the Mouse, 4-15, 4-18 to 4-20
- Starting an application, 1-4, 2-4, 2-8
  - starting a second application, 2-9
  - with Arrow keys, 2-4, 2-8
  - with the Mouse, 2-1, 2-4, 2-8
- Starting and switching contexts, 6-13
- Starting CM, 2-7, 3-1, 5-3 to 5-4
- Starting more contexts, 2-11
- Starting the CM Editor, 5-6 to 5-7
- Starting Windows, 4-3
- Status codes, 8-2 to 8-9
  - CM, 8-2 to 8-4
  - InterContext Message Server, 8-5
- Status Messages, 8-6 to 8-12
- Status terms, 2-6, 3-6 to 3-7
- Stopped, 2-6, 3-6 to 3-7
- Submit file
  - invoking from CM 5-29 to 5-31
- Suspended context, 1-4, 6-6
- Swap file, 3-8, 5-2 to 5-3
  - creating, 5-2 to 5-3
  - using CrashDump.sys, 5-2, 6-17
- Swap requests, 6-21
- Swapping, 2-6, 3-6 to 3-7
  - and the operating system, 5-2 to 5-3
  - contexts that are not swapped, 3-8 to 3-9, 6-8
  - to disk, 1-5, 3-7 to 3-9
  - communications applications, 3-9, 6-8

- Swapping (**con't.**)
  - real-time applications, 3-9, 6-8
- Switching contexts, 1-5
  - without using CM screen, 2-10, 3-5
- System services
  - writing, 6-21
  - and CM, 5-3
  
- Tile command, 4-6, 4-22 to 4-25, 4-35 to 4-38
- Tiled windows
  - creating using the keyboard, 4-35 to 4-38
  - creating with the Mouse, 4-22 to 4-25
- TranslatePhToCh, 6-12
- Troubleshooting, 8-1 to 8-12
- Tutorial. See **Exercises.**
  
- Undo (f2), 5-10, 5-19
- UnLockVideo, 6-2 to 6-3,
- User Configuration file
  - configuring the Mouse, 5-5 to 5-6
  - editing, 5-4 to 5-5
- Using CM. See **Exercises.**
- Using CM Editor. See **Exercises.**
- Using the Mouse buttons, 4-13 to 4-15
- Using Windows and the Mouse. See **Exercises.**
  
- VAM. See **Video Access Method.**
- Variable sized partitions, 5-13 to 5-15, 6-19 to 6-20
- VDM. See **Video Display Management.**
- Version 4 Run files, 6-19
- Version 6 Run files, 6-19 to 6-21
  
- Version number of CM, 5-4
- Video Access Method (VAM), 6-2, 6-6 to 6-8
- Video Display Management (VDM), 6-2
- Video pointer map, 6-2 to 6-3
- Volume field in CM Editor, 5-17, 5-24
  
- Waiting, 2-6, 3-6 to 3-7
- Window Services. See also **Windows.**
  - definition, 1-3
- Window-aware programs, 6-7
- Windows
  - adjusting contents, 4-20
  - border, 4-7
  - copying from one to another, 3-9 to 3-13
  - current, 1-4, 4-7 to 4-8, 4-31
    - using the keyboard to make current, 4-31 to 4-35
    - using the Mouse to make current, 4-16
  - definition, 4-2
  - dotted lines, 4-25 to 4-28, 4-38
  - examples, 4-3
  - features, 1-3 to 1-4, 4-2 to 4-3
  - finishing, 4-16, 4-31
  - full
    - using the keyboard, 4-41
    - using the Mouse, 4-28
  - hidden contents
    - making visible, 6-8
  - iconic
    - using the keyboard, 4-38 to 4-41
    - using the Mouse, 4-25 to 4-28
  - menu, 4-3 to 4-6
    - activating, 4-3
      - using the keyboard, 4-31 to 4-32
      - with the Mouse, 4-15 to 4-16, 4-22
  - cancelling
    - using the keyboard, 4-30
    - with the Mouse, 4-15, 4-17

**Windows (con't.)**

- commands
  - CM, 4-5
  - Full, 4-4, 4-5, 4-7, 4-28, 4-41
  - Icon, 4-25 to 4-28, 4-38 to 4-41
  - Move, 4-6
  - Restore, 4-5, 4-28 to 4-29, 4-41 to 4-42
  - selecting, 4-32, 4-15
  - Size, 4-6
  - Tile, 4-6, 4-22 to 4-25, 4-35 to 4-38
  - when window is always full screen, 4-4
- moving, 4-8
  - using the keyboard, 4-30, 4-35
  - with the Mouse, 4-15, 4-21 to 4-22
- opening
  - using the keyboard, 4-31 to 4-32

- with the Mouse, 4-17
- programs that run under, 6-6 to 6-8
- requiring full screen, 4-7, 6-7 to 6-8
- restored
  - using the keyboard, 4-41 to 4-42
  - using the Mouse, 4-28 to 4-29
- seeing what is behind, 4-19
- sizing, 4-8
  - using the keyboard, 4-30, 4-32 to 4-35
  - with the Mouse, 4-15, 4-18 to 4-20
- starting, 4-3
- tiled
  - using the keyboard, 4-35 to 4-38
  - using the Mouse, 4-22 to 4-25
- viewing hidden contents of, 4-15, 4-20, 4-30

# USER'S COMMENT SHEET

---

Context Manager/VM, Second Edition  
09-01050-01 DTA-610

---

*We welcome your comments and suggestions. They help us improve our manuals. Please give specific page and paragraph references whenever possible.*

*Does this manual provide the information you need? Is it at the right level? What other types of manuals are needed?*

*Is this manual written clearly? What is unclear?*

*Is the format of this manual convenient in arrangement, in size?*

*Is this manual accurate? What is inaccurate?*

Name \_\_\_\_\_ Date \_\_\_\_\_

Title \_\_\_\_\_ Phone \_\_\_\_\_

Company Name/Department \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_

*Thank you. All comments become the property of Convergent Technologies, Inc.*



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES

**BUSINESS REPLY MAIL**  
FIRST CLASS MAIL PERMIT NO. 1807 SAN JOSE, CA

POSTAGE WILL BE PAID BY ADDRESSEE

**Convergent Technologies**  
**Attn: Technical Publications**  
2700 North First Street  
PO Box 6685  
San Jose, CA 95150-6685



Fold Here

# Convergent

2700 North First Street  
San Jose, CA 95150-6685

*Printed in USA*