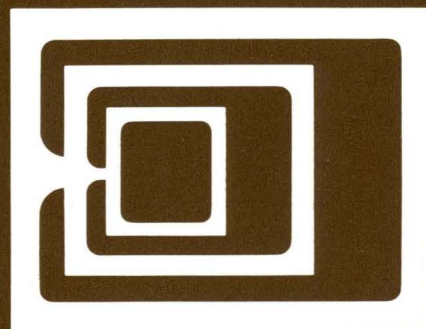


# COBOL

**PRO/Tool Kit COBOL-81  
Installation Guide and  
Documentation Supplement**

Order No. AA-Y285A-TH



**digital**  
software

# **PRO/Tool Kit COBOL-81 Installation Guide and Documentation Supplement**

Order No. AA-Y285A-TH

---

**September 1983**

This document contains installation instructions and release notes for COBOL-81 running under the PRO/Tool Kit. It describes how to compile and build applications and contains information specific to this release. This is a new manual.

**OPERATING SYSTEM AND VERSION:** PRO/Tool Kit V5  
P/OS V1

**SOFTWARE VERSION:** PRO/Tool Kit V1  
PRO/Tool Kit COBOL-81 V2.1

digital equipment corporation, maynard, massachusetts

First Printing, September 1983

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The specifications and drawings herein are the property of Digital Equipment Corporation and shall not be reproduced or copied or used in whole or in part as the basis for the manufacture or sale of items without written permission.

Copyright © 1983 by Digital Equipment Corporation.  
All Rights Reserved.

The following are trademarks of Digital Equipment Corporation:

CTI BUS	MASSBUS	Rainbow
DEC	PDP	RSTS
DECmate	P/OS	RSX
DECsystem-10	PRO/BASIC	Tool Kit
DECSYSTEM-20	PRO/Communications	UNIBUS
DECUS	Professional	VAX
DECwriter	PRO/FMS	VMS
DIBOL	PRO/RMS	VT
<b>digital</b>	PROSE	Work Processor
	PROSE PLUS	

---

#### HOW TO ORDER ADDITIONAL DOCUMENTATION

In Continental USA and Puerto Rico call 800-258-1710

In New Hampshire, Alaska, and Hawaii call 603-884-6660

In Canada call 613-234-7726 (Ottawa-Hull)  
800-267-6146 (all other Canadian)

##### DIRECT MAIL ORDERS (USA & PUERTO RICO)\*

Digital Equipment Corporation  
P.O. Box CS2008  
Nashua, New Hampshire 03061

\*Any prepaid order from Puerto Rico must be placed  
with the local Digital subsidiary (809-754-7575)

##### DIRECT MAIL ORDERS (CANADA)

Digital Equipment of Canada Ltd.  
940 Belfast Road  
Ottawa, Ontario K1G 4C2  
Attn: A&SG Business Manager

##### DIRECT MAIL ORDERS (INTERNATIONAL)

Digital Equipment Corporation  
A&SG Business Manager  
c/o Digital's local subsidiary or  
approved distributor

---

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Northboro, Massachusetts 01532

---

# CONTENTS

Page

PREFACE		v
CHAPTER 1	INTRODUCTION	
1.1	PRO/TOOL KIT COBOL-81 . . . . .	1-1
1.2	THE APPLICATION DEVELOPMENT PROCESS . . . . .	1-2
1.3	DIFFERENCES BETWEEN PRO/TOOL KIT COBOL-81 AND COBOL-81 ON RSX-11 SYSTEMS . . . . .	1-2
1.3.1	Language . . . . .	1-3
1.3.2	Run-Time Support . . . . .	1-3
1.3.3	Installation . . . . .	1-4
1.3.4	Use of the Compiler and Other Components . . . . .	1-4
CHAPTER 2	PRO/TOOL KIT COBOL-81 INSTALLATION AND VERIFICATION	
2.1	INSTALLATION REQUIREMENTS . . . . .	2-1
2.2	COBOL-81 KIT CONTENTS . . . . .	2-1
2.3	INSTALLING PRO/TOOL KIT COBOL-81 . . . . .	2-1
2.3.1	The Installation Process . . . . .	2-1
2.3.2	Message Summary . . . . .	2-2
CHAPTER 3	DEVELOPING THE APPLICATION	
3.1	WRITING THE SOURCE PROGRAM . . . . .	3-2
3.1.1	Language . . . . .	3-2
3.1.2	Conventions for Calling P/OS Facilities . . . . .	3-2
3.1.2.1	Parameter-Passing . . . . .	3-2
3.1.2.1.1	Integers . . . . .	3-3
3.1.2.1.2	Text . . . . .	3-3
3.1.2.2	Calling Graphics Routines . . . . .	3-3
3.1.2.3	Using ACCEPT/DISPLAY Extensions and P/OS Services . . . . .	3-4
3.1.2.3.1	DETACH . . . . .	3-4
3.1.2.3.2	ATTACH . . . . .	3-4
3.2	COMPILING THE SOURCE PROGRAM . . . . .	3-5
3.3	BUILDING APPLICATION TASK IMAGES . . . . .	3-5
3.3.1	Running the BLDODL Utility . . . . .	3-5
3.3.1.1	BLDODL Switches . . . . .	3-6
3.3.1.2	The Command (CMD) File . . . . .	3-9
3.3.1.3	The Overlay Descriptor Language (ODL) File . . . . .	3-10
3.3.1.4	The Installation (INS) File . . . . .	3-11
3.3.2	Building the Application Using LINK @ . . . . .	3-11
3.3.3	Building the Application Using LINK/C81 . . . . .	3-11
3.3.4	Writing the Installation (INS) File . . . . .	3-11
3.4	INSTALLING THE APPLICATION WITH FAST INSTALL . . . . .	3-12
CHAPTER 4	RELEASE NOTES	
APPENDIX A	SAMPLE APPLICATION	
APPENDIX B	APPLICATION DEVELOPMENT EXAMPLE	
B.1	USING THE REFORMAT UTILITY TO CONVERT THE SOURCE FILE TO ANSI FORMAT . . . . .	B-1
B.2	COMPILING THE SOURCE FILE . . . . .	B-2

## CONTENTS

B.3	BUILDING THE COMMAND AND OVERLAY DESCRIPTOR	
	LANGUAGE FILES . . . . .	B-2
B.4	BUILDING THE SAMPLE APPLICATION . . . . .	B-4
B.5	CREATING AN APPLICATION DIRECTORY ON THE	
	PROFESSIONAL . . . . .	B-4
B.6	INSTALLING THE SAMPLE APPLICATION . . . . .	B-4
B.7	EXECUTING THE SAMPLE APPLICATION . . . . .	B-5



## PREFACE

### MANUAL OBJECTIVES

This manual describes installation and verification procedures for PRO/Tool Kit COBOL-81 and gives instructions for its use, including information specific to this release of PRO/Tool Kit COBOL-81.

### INTENDED AUDIENCE

Anyone involved with installing or using PRO/Tool Kit COBOL-81 should read this manual.

### STRUCTURE OF THIS DOCUMENT

This manual is organized as follows:

- Chapter 1 introduces PRO/Tool Kit COBOL-81, and lists its software components. This chapter contains a list of differences between PRO/Tool Kit COBOL-81 and COBOL-81 on RSX-11 systems.
- Chapter 2 tells how to install and verify PRO/Tool Kit COBOL-81.
- Chapter 3 describes each of the steps in developing an application for the Professional. This chapter explains how to call P/OS facilities in the program, and how to compile, build, and test the application.
- Chapter 4 contains release notes (information about bugs and restrictions that are specific to this release of PRO/Tool Kit COBOL-81).
- Appendix A gives the source file of the sample application used in Appendix B.
- Appendix B contains an application development example that serves as a hands-on introduction to application development for the Professional.

### ASSOCIATED DOCUMENTS

In the course of developing an application for the Professional, you may need to refer to documentation in three broad areas:

- For more information about COBOL-81, refer to the other manuals that came with PRO/Tool Kit COBOL-81:
  - The COBOL-81 Language Reference Manual provides information about the language itself.

## PREFACE

- The COBOL-81 RSX-11M/M-PLUS User's Guide provides information about using COBOL-81 on the RSX operating system. Most of this information applies to PRO/Tool Kit COBOL-81 as well.

Note that if any information in this manual conflicts with information in any of the above manuals, the information in this manual takes precedence.

- For more information about the operating system (RSX-11M/M-PLUS) on which you are developing your applications, refer to the Information Directory that came with the operating system.
- For more information about the Professional Tool Kit and the Professional environment, refer to the Tool Kit Documentation Directory. The Tool Kit User's Guide is the focus for all application development activities; it is an excellent starting point.

## CONVENTIONS USED IN THIS DOCUMENT

The following conventions are observed in this manual:

- Uppercase words and letters used in examples indicate that you should type the word or letter exactly as shown.
- Lowercase words or letters used in examples indicate that you are to substitute a word or value of your choice.
- Brackets ([]) may indicate optional elements, or may be required syntax in a user identification code or directory specification. This is generally clear from context; if not, the accompanying text makes it clear.
- Vertical or horizontal ellipsis points ( : or ...) indicate omission of information, often repetitive information.
- In examples of terminal dialog, red ink indicates input that you supply, to distinguish it from prompts and other system output.
- Unless otherwise specified, all commands that you give at the terminal are terminated by a carriage return.
- <RET> represents the RETURN key.
- In text, <CTRL/Z> represents a CONTROL-Z combination; that is, you press the CTRL key and Z at the same time.

The term "Professional" refers to the Professional 350 computer. The term "PRO/Tool Kit" refers to the application development environment in which a programmer can write, compile, build, execute and debug a user-installable P/OS application on the Professional, without use of a host system.

## CHAPTER 1

### INTRODUCTION

This chapter introduces PRO/Tool Kit COBOL-81, and shows how PRO/Tool Kit COBOL-81 fits into the Professional family of software. It is recommended that you read this chapter first. Then, based on what you read here and on your current level of knowledge about the topics discussed, read elsewhere in the Professional documentation before you attempt to use PRO/Tool Kit COBOL-81.

The following subjects are covered in this chapter:

- Section 1.1 introduces PRO/Tool Kit COBOL-81 and lists its software components.
- Section 1.2 sketches the process that you must follow to develop applications for use on the Professional.
- Section 1.3 outlines the differences between PRO/Tool Kit COBOL-81 and COBOL-81 on RSX-11 systems. This section is very important, especially if you have had prior experience with COBOL-81.

#### 1.1 PRO/TOOL KIT COBOL-81

PRO/Tool Kit COBOL-81 is a modified version of the COBOL-81/RSX-11 compiler and Object Time System (OTS).

The PRO/Tool Kit COBOL-81 OTS, which provides run-time support for applications, is supplied as part of every Professional operating system (P/OS).

The following software components are included in the PRO/Tool Kit COBOL-81 package:

- C81 - the COBOL-81 compiler
- BLD - the BLDODL Utility, which builds command (CMD) and overlay descriptor language (ODL) files
- RFM - the REFORMAT Utility, which converts terminal format source programs to conventional ANSI format, and vice versa
- COBOL-81 Symbolic Debugger, which is included in all tasks using the /DEB switch
- COBOL-81 Run-Time Libraries



## INTRODUCTION

- Documentation
  - PRO/Tool Kit COBOL-81 Installation Guide and Documentation Supplement
  - COBOL-81 RSX-11M/M-PLUS User's Guide
  - COBOL-81 Language Reference Manual

### 1.2 THE APPLICATION DEVELOPMENT PROCESS

The development of an application requires a number of steps, some of which may have to be repeated in a cycle. The Tool Kit User's Guide describes all these steps, and is the reference document for many of them.

In general, you can use your Professional 350 as a minicomputer to create application source files, compile programs, task build and test an application. If necessary, you edit, recompile, and rebuild the application, all on the Professional.

Here, briefly, is a list of the steps you must follow; more detailed information is presented in Chapter 3. The application development example in Appendix B walks you through most of these steps with a sample application.

1. Design the application and related help, menu, and message files.
2. Code the application, using the editor provided with the PRO/Tool Kit.
3. Compile the source file(s) with the PRO/Tool Kit COBOL-81 compiler.
4. Build the application, using either the BLDODL utility and the LINK @ command, or the LINK/C81 command.
5. Install the application.
6. Execute the task image via the RUN command.

### 1.3 DIFFERENCES BETWEEN PRO/TOOL KIT COBOL-81 AND COBOL-81 ON RSX-11 SYSTEMS

This section describes the differences between PRO/Tool Kit COBOL-81 and COBOL-81 on RSX-11 systems.

For the most part, the function and use of PRO/Tool Kit COBOL-81 and COBOL-81 on RSX-11 systems are identical. For this reason, you can refer to the RSX-11 COBOL-81 documentation supplied with PRO/Tool Kit COBOL-81 for most of the information that you need. However, there are some important differences, which are summarized in this section. These differences are also noted at appropriate points in the other chapters.

## INTRODUCTION

In general, when a subject is not mentioned in this manual, you can assume that the treatment of that subject in the COBOL-81 RSX-11M/M-PLUS User's Guide applies to PRO/Tool Kit COBOL-81.

### 1.3.1 Language

The PRO/Tool Kit COBOL-81 language is as described in the COBOL-81 Language Reference Manual, with the following exceptions:

- The Professional computer supports the 8-bit DEC Multinational Character Set, which is an extension of the ASCII set (see the Terminal Subsystem Manual for full details). PRO/Tool Kit COBOL-81 handles the DEC Multinational Character Set in the same manner as COBOL-81 on RSX-11 systems. The following list describes differences, not in the way the two compilers function, but rather in the way COBOL-81 handles the 8-bit compatible DEC Supplemental Graphics characters in relation to the 7-bit compatible ASCII subset:
  - All characters of the DEC Multinational Character Set are valid in string literals and comments, when you compile the application, and as input or output when you run the application on the Professional.
  - The PRO/Tool Kit COBOL-81 compiler does not allow DEC Supplemental Graphics Characters in user-defined words. (See the COBOL-81 Language Reference Manual.)
  - You cannot do comparisons involving the 8-bit-compatible (DEC Supplemental Graphics) characters in conditional expressions.
- The SORT/MERGE verbs are not supported in PRO/Tool Kit COBOL-81. Do not use SORT/MERGE.
- Input/Output to tape devices is not supported on the Professional.

### 1.3.2 Run-Time Support

There are differences both in the run-time support itself and in the way you include it when you build the application. These differences are:

- The PRO/Tool Kit COBOL-81 OTS is contained in an object module library and a clusterable resident library.
- There is support of Record Management Services (RMS-11) indexed files only for applications running on P/OS Hard Disk, not for applications running on P/OS Diskette.
- Run-time error messages are generally the same, but may be formatted differently on the screen. Every run-time error message is followed by this statement:

Please write down the above message -- Press RESUME to continue.

## INTRODUCTION

### 1.3.3 Installation

Chapter 2 of this manual describes the installation and verification procedure for PRO/Tool Kit COBOL-81, and lists the contents of the installation media.

Unlike most other versions of COBOL-81, the PRO/Tool Kit COBOL-81 compiler is distributed in the form of a prebuilt task image. You need not (and cannot) edit a compiler build file to tailor PRO/Tool Kit COBOL-81 to your environment. Similarly, the Object Time System task is prebuilt and cannot be modified. Installation simply consists of copying files from the distribution volume to the appropriate directories.

### 1.3.4 Use of the Compiler and Other Components

An experienced user of COBOL-81 on an RSX-11 system will find several differences in the way the compiler and other system components are invoked and used. These differences are summarized here, and discussed in more detail in Chapter 3.

- You use the COBOL command to invoke the PRO/Tool Kit COBOL-81 compiler. Refer to the COBOL-81 RSX-11M/M-PLUS User's Guide for a complete list of command qualifiers that may be used during compilation of the source program.

#### NOTE

Note that the CODE:CIS qualifier must not be used at compile time because the Professional does not support the commercial instruction set.

- You choose the method to build your application depending on whether or not the program accesses any P/OS services, Form Management System (FMS), or CORE graphics library (CGL) routines. If it does access any of these, use the BLDODL utility with appropriate switches followed by the LINK @ command; otherwise, use the LINK/C81 command with appropriate qualifiers. (See Chapter 3 and the COBOL-81 RSX-11M/M-PLUS User's Guide.)

## CHAPTER 2

### PRO/TOOL KIT COBOL-81 INSTALLATION AND VERIFICATION

This chapter contains installation instructions for COBOL-81 running under the PRO/Tool Kit.

#### 2.1 INSTALLATION REQUIREMENTS

The following hardware and software are required for the installation of PRO/Tool Kit COBOL-81 on the Professional 350:

- HARDWARE
  - RD51-A 10 megabyte Winchester drive
  - MSC11-CK memory option upgrade
- SOFTWARE
  - PRO/TOOL KIT V1.0

#### 2.2 COBOL-81 KIT CONTENTS

- Two RX50s
  - LABEL: PROCOBOL1
  - LABEL: PROCOBOL2
- Installation Instructions

#### 2.3 INSTALLING PRO/TOOL KIT COBOL-81

This section describes the installation process for PRO/Tool Kit COBOL-81 and summarizes the screen messages that you see as you progress.

##### 2.3.1 The Installation Process

The installation process consists of these three steps:

- STEP 1: Execute the PRO/Tool Kit application from the Main Menu.
- STEP 2: Place the diskette labeled PROCOBOL1 into drive 1.  
Place the diskette labeled PROCOBOL2 into drive 2.
- STEP 3: When the \$ prompt is displayed, enter the following command line:  
  
\$ @DZ1:[INSTALL]INSTALL

## PRO/TOOL KIT COBOL-81 INSTALLATION AND VERIFICATION

### 2.3.2 Message Summary

After you enter the INSTALL command, the following message appears on the screen. You must respond with "y" or "Y" to proceed with the installation.

```
> *****
> *
> *          INSTALLATION OF PRO/TOOL KIT COBOL-81          *
> *                      VERSION 2.1                      *
> *
> *****
>
> Please verify that :
> 1) the diskette labeled  PROCOBOL1 is in drive 1.
> 2) the diskette labeled  PROCOBOL2 is in drive 2.
>
> NOTE :
> This procedure will delete all files in the [PROCOBOL] directory
> on BIGVOLUME!. Please move any files you wish to save to another
> directory before continuing this installation.
>
>
> * Are you ready to continue with the installation <Y/N> ?
```

If you choose to continue, the following message is displayed, indicating that the installation is in progress:

```
> *****
> *
> *          INSTALLATION OF PRO/TOOL KIT COBOL-81          *
> *                      VERSION 2.1                      *
> *
> *****
>
>
>          INSTALLATION IN PROGRESS .....
```

If you have not placed the correct diskette (PROCOBOL2) into drive 2, the following message prompts you to do so, then to press <RET> to continue with the installation:

```
>
> ATTENTION:
> Please place the diskette labeled  PROCOBOL2 into DRIVE 2.
>
> (If the PROCOBOL2 diskette is in drive 2 it may be upside down.)
> * *          Press <RETURN> to continue
```

## PRO/TOOL KIT COBOL-81 INSTALLATION AND VERIFICATION

After all files have been copied from the kit media to the Professional, the completion message is displayed and you are prompted to select or reject file placement verification:

```
>| *****
>| *
>| *      INSTALLATION OF PRO/TOOL KIT COBOL-81      *
>| *                      VERSION 2.1                  *
>| *
>| *                      >>>> COMPLETED <<<<          *
>| *
>| *****
>|
>| All files have been copied from the kit media to the
>| Winchester drive on your PROFESSIONAL 350. The next
>| portion of the installation is verification of file
>| placement on BIGVOLUME!. Please note that this verifi-
>| cation is optional.
>|
>|
>| * Verify file placement <Y/N> ?
```

If you respond to the "Verify file placement" prompt with "Y", and if an error was encountered during the verification phase, the following message is displayed. This message identifies the file containing the list of errors and prompts you to display the file on the screen:

```
>|
>| >>>>> ERROR <<<<<
>|
>| The verification procedure detected errors during the
>| verification process. The file containing a list of
>| the errors is in DW1:[USERFILES]RESULT.DAT.
>|
>| * Display the error file on the terminal <Y/N> ?
```

After the file has been displayed (or if you choose not to have the file displayed), press <RET> to continue:

```
>| * *      Press <RETURN> to continue
```

If you responded to the "Verify file placement" prompt with "Y", and no errors were encountered, the success message is displayed:

```
>|
>| >>>>> SUCCESS <<<<<
>|
>| No errors were detected by the verification procedure
>|
```

If no errors were encountered, you are prompted to delete the error message file.

```
>| * Delete file DW1:[USERFILES]RESULT.DAT      <Y/N/G/Q> ?
```

If you respond with "Y" or "G", the error message file is deleted; if you respond with "N" or "Q", the error message file is not deleted.

## PRO/TOOL KIT COBOL-81 INSTALLATION AND VERIFICATION

You will then be prompted to verify image execution:

>\* Verify image execution <Y/N> ?

After the elements are tested (or if you responded to the "Verify file placement" prompt with "N" following the "INSTALLATION ... COMPLETED" message), the following message appears on the screen:

```
-----  
I D I I G I I T I A I L I  
-----
```

COBOL-81

VERSION 2.1

Installation Verification Program

Testins:

Interaction with clustered COBOL-81 OTS	SUCCESSFUL
Interaction with clustered RMS	SUCCESSFUL
Interaction with clustered POSRES	SUCCESSFUL
Interaction with clustered CGLEIS	SUCCESSFUL
Image execution test	SUCCESSFUL

There are three indications of an unsuccessful test:

1. The "Interaction with ..." message remains flashing.
2. "\*\*\* FAILED \*\*\*" is displayed to the right of the "Interaction with ..." message.
3. The "Image execution test SUCCESSFUL" message is not displayed.

Any of these three indications signals that the installation was not successful, and that you should run the entire installation and verification procedure again.

If the installation is still not successful after the second attempt, contact your software support center.



## CHAPTER 3

### DEVELOPING THE APPLICATION

This chapter explains each of the steps you must perform to develop an application using PRO/Tool Kit COBOL-81. Note, however, that the descriptions in this chapter are restricted to aspects of the development process that are unique to PRO/Tool Kit COBOL-81. For complete information on developing applications, refer to the COBOL-81 RSX-11M/M-PLUS User's Guide, and to the appropriate manuals in the Professional Tool Kit documentation.

PRO/Tool Kit COBOL-81 requires that the syntax for compiler qualifiers be in DCL format. See the COBOL-81 RSX-11M/M-PLUS User's Guide for more information. The BLDODL utility uses switches in MCR format. Part I, Appendix D, of the COBOL-81 RSX-11M/M-PLUS User's Guide lists and describes these switches. The body of that manual uses the DCL format syntax.

The development cycle for PRO/Tool Kit COBOL-81 applications consists of the following steps:

1. Writing/updating the COBOL-81 source program(s).
2. Compiling the source program file(s) with C81, the PRO/Tool Kit COBOL-81 compiler:
  - Compiling with /DEB to use the COBOL-81 symbolic debugger at run time
  - Compiling without /DEB when the program is error-free.
3. Building the application task image using either:
  - The BLDODL utility with appropriate switches followed by the LINK @ command, or
  - The LINK/C81 commandand specifying the /DEB switch when you want to use the COBOL-81 debugger.
4. Installing the application with the Fast Install facility.
5. Running the application task on the Professional, and, if desired, debugging using a terminal attached to the Professional's printer port.
6. Continuing the cycle until the application is error-free.
7. When the application is error-free, running the Application Diskette Builder to create final application diskettes.

## DEVELOPING THE APPLICATION

The following sections present specific details of each phase of PRO/Tool Kit COBOL-81 application development for P/OS Hard Disk systems. For a complete description of each stage of PRO/Tool Kit application development, particularly the requirements for applications running on P/OS Diskette, you should refer to the Tool Kit User's Guide.

### 3.1 WRITING THE SOURCE PROGRAM

You can use any text editor available on the Professional to create a source program. Refer to the appropriate text editor manual for more information.

The following sections provide information specific to PRO/Tool Kit COBOL-81.

#### 3.1.1 Language

The PRO/Tool Kit COBOL-81 language is as described in the COBOL-81 Language Reference Manual, with the exceptions regarding DEC Supplemental Graphics characters and the SORT/MERGE verbs noted in Section 1.3.1 of this manual.

#### 3.1.2 Conventions for Calling P/OS Facilities

Your COBOL-81 source program can include external subroutine calls to access P/OS facilities. These external subroutine calls are described in detail in the Tool Kit User's Guide and the CORE Graphics Library Manual. The following paragraphs supplement that generic information with some COBOL-specific points and examples.

P/OS facilities exist in the following areas:

1. P/OS user interface support. Your application has access to P/OS menu, help, message, and other system services. The ability to display error messages, status, online HELP, and menus on the screen is provided through the P/OS Service Routines Library (POSRES).
2. Extended software functions. Your application can invoke callable system services and PRO/Communications.
3. Bitmap graphics routines. Your application has access to the graphics routines in the CORE Graphics Library.

Conventions that your program must observe when calling P/OS facilities from PRO/Tool Kit COBOL-81 are described below. The particular facilities your program calls may affect the CMD and INS files, as described later in this chapter.

##### 3.1.2.1 Parameter-Passing

P/OS facilities are accessed through the COBOL-81 CALL BY REFERENCE statement (parameters must be passed BY REFERENCE). This is the default parameter passing technique used by PRO/Tool Kit COBOL-81. Note that you cannot specify literals in the CALL statement. You must define a data item and ensure that that item contains the correct value before using it as a parameter.

## DEVELOPING THE APPLICATION

The next two sections describe how to pass integers or text to P/OS services from PRO/Tool Kit COBOL-81.

**3.1.2.1.1 Integers** - Integers should be described as PIC S9(4) COMP items, as follows:

```
01 A PIC S9(4) COMP.  
    *  
    *  
    *  
CALL "ssss" USING A.
```

**3.1.2.1.2 Text** - Text should be described as PIC X(n), where "n" is the maximum length of your text string. Normally, when you pass a text string to a P/OS service, you will be required to supply an additional integer parameter that specifies the length of the text string, as follows:

```
01 LEN      PIC S9(4) COMP VALUE 20.  
01 TEXT-STR PIC X(20).  
    *  
    *  
    *  
CALL "ssss" USING TEST-STR LEN.
```

### 3.1.2.2 Calling Graphics Routines

If you intend to call graphics routines from PRO/Tool Kit COBOL-81, you should be aware that most graphics routines require real numbers as parameters. PRO/Tool Kit COBOL-81 does not support real numbers explicitly in the syntax. However, you can use the routines described below to convert integers to real numbers:

#### 1. CONIFL

In the CONIFL routine, the integers must be described as PIC S9(4) COMP and the real numbers must be described as PIC X(4). Once you have stored a real number in a field described as PIC X(4), you can only use that item's contents as a parameter to a routine that expects and can handle a real number.

The syntax of the routine is:

```
CALL "CONIFL" USING  int real ...
```

Examples:

```
CALL "CONIFL" USING  INT1 REAL1.
```

```
CALL "CONIFL" USING  
    INT1 REAL1  
    INT2 REAL2  
    INT3 REAL3.
```

```
CALL "CONIFL" USING INT-ARRAY(I) REAL-ARRAY(I).
```

## DEVELOPING THE APPLICATION

Once you have converted the integers to real numbers, you should pass the real number data item to the CGL routine. See the program source file, C81IVP.CBL, Appendix A for an example using CONIFL to call graphics routines.

### 2. INTFLT

The INTFLT routine provides the same basic function as the CONIFL routine, except that it allows non-integer parameters. Input arguments to INTFLT must be declared as PIC S9(4)V9(4) COMP data items; output arguments must be PIC X(4) data items. As with CONIFL, INTFLT may be called with any number of paired arguments.

For example:

Data Division.

```
01 Pseudo-Float      PIC S9(4)V9(4) COMP.  
01 Real-Float        PIC X(4).
```

Procedure Division.

```
CALL "INTFLT" USING  
Pseudo-Float-0      Real-Float-0  
      .              .  
      .              .  
      .              .  
Pseudo-Float-(N-1)  Real-Float-(N-1)  
Pseudo-Float-N      Real-Float-N.
```

### 3.1.2.3 Using ACCEPT/DISPLAY Extensions and P/OS Services

When you use the screen extensions of the ACCEPT/DISPLAY verbs, the terminal will be attached when the application begins execution; this is required to properly handle type-ahead. Two routines have been supplied, DETACH and ATTACH, for using the screen extensions of ACCEPT/DISPLAY when you expect to call P/OS service routines that require use of the terminal. (See the COBOL-81 Language Reference Manual for more information on the ACCEPT/DISPLAY verbs.)

**3.1.2.3.1 DETACH** - Use this routine before calling a P/OS routine that requires use of the terminal, for example, WTRES (wait for RESUME key):

```
CALL "DETACH".  
CALL "WTRES".
```

**3.1.2.3.2 ATTACH** - Use this routine to re-attach the terminal. For example,

```
CALL "DETACH".  
CALL "WTRES".  
CALL "ATTACH".
```

## DEVELOPING THE APPLICATION

### 3.2 COMPILING THE SOURCE PROGRAM

In general, use of the PRO/Tool Kit compiler is very similar to the use of the COBOL-81 compiler for RSX-11 systems, as described in the COBOL-81 RSX-11M/M-PLUS User's Guide. Input and output files are specified in the same way. The compiler qualifiers for the RSX compiler are valid for the PRO/Tool Kit COBOL-81 compiler, with one exception: the CODE:CIS qualifier cannot be used at compile time. With this exception, the use and default settings of compiler qualifiers are the same for PRO/Tool Kit COBOL-81 and COBOL-81 on RSX-11 systems.

The command line to compile your source program is:

```
$ COBOL[qualifiers] filename[qualifiers]...
```

Refer to the COBOL-81 RSX-11M/M-PLUS User's Guide for more information on the command line format and a complete list of compiler qualifiers.

### 3.3 BUILDING APPLICATION TASK IMAGES

To build application task images, you may use either:

- The BLDODL utility, followed by the LINK @ command when your application accesses P/OS services, FMS, or CORE Graphics routines
- The LINK/C81 command whenever your application does not use P/OS services, FMS, or CORE Graphics.

#### 3.3.1 Running the BLDODL Utility

After compiling the program, use the BLDODL utility to generate a command (CMD) file and overlay descriptor (ODL) file, and optionally, an installation (INS) file for your application. The PRO/Tool Kit LINK facility uses these files to define how libraries are referenced, and to specify special-purpose buffers, logical unit numbers (LUNs), and event flags (EFNs). The BLDODL utility uses the SKL file created by the compiler as input.

First, invoke the BLDODL utility:

```
$ RUN BLD
```

When BLD prompts, provide the file specifications, switches, and switch values in the following format:

```
BLD>output[/switch]...=input1...[,inputn[/switch]...]...
```

Note that no switches, or any number of valid switches, may be specified, as long as no switch is repeated. The Professional-specific BLDODL switches are discussed in Section 3.3.1.1. See Part I, Appendix D, of the COBOL-81 RSX-11M/M-PLUS User's Guide for more information on the command line format for the BLDODL utility.

Press <CTRL/Z> to exit the BLDODL utility.

## DEVELOPING THE APPLICATION

### 3.3.1.1 BLDODL Switches

The Professional-specific BLDODL utility switches, which are explained in this section, are:

```
/INS  
/CGL  
/FMS  
/POS  
/DEB  
/DIA
```

These two sample command lines use combinations of the Professional BLDODL switches.

```
BLD> TEST = TEST/DIA  
BLD> TEST = TEST/INS:"TEST APPLICATION"/CGL/DEB:RED/POS:SSM:HTF
```

#### /INS

The /INS switch activates automatic building of the installation file required by the Fast Install Application and the Install Application of Disk Services. The INS file will have the same name as the CMD and ODL files created by the BLD utility. This name is the user-supplied name to the left of the equal sign on the BLD utility command line.

For example:

```
BLD> example=examp/ins
```

creates three files:

- 1) example.CMD
- 2) example.ODL
- 3) example.INS

To keep the application task image name the same as the application name, use /INS. If you want a different application name, use:

```
/INS:"application name"
```

where "application name" is an alphanumeric string containing from 1 to 40 characters. For example:

```
BLD> TEST = TEST/INS:"test program"
```

creates an INS file with "test program" as the application name. When the /CGL switch is specified on the same BLD command line as the /INS switch, the installation file will contain the required statements to support use of the CORE Graphics Library.

#### /CGL

The /CGL switch provides support of the CORE Graphics Library through modification of the CMD file (and the INS file if the /INS switch was specified).

#### /FMS

The /FMS switch provides support of clustered FMS on the Professional. If your COBOL-81 application calls P/OS services in conjunction with FMS, the /POS switch (with corresponding values) should be used.

## DEVELOPING THE APPLICATION

### /POS

The /POS switch provides support of P/OS services specific to the Professional environment. Whenever the /POS switch is specified on the BLDODL utility command line, at least one of the switch values must be specified. The syntax for the /POS switch is:

```
BLD> filename=filename/POS:switchvalue[:switchvalue...]
```

The following syntax is illegal because no switch value is given:

```
BLD> filename = filename/POS
```

/POS switch values correspond to the five P/OS services supported by COBOL-81:

<u>SWITCH VALUE</u>	<u>P/OS SERVICE</u>
SSM	Static Single Choice Menu
DSM	Dynamic Single Choice Menu
MCM	Multiple Choice Menu
HTF	Help Text File
FSS	File Selection Services

You may supply from one to five switch values on the command line, but do not repeat any switch value on the line.

### /DEB

The /DEB switch provides support for the COBOL-81 symbolic debugger. The /DEB does not redirect debugger I/O to the printer port; use /DEB:RED if redirection of debugger I/O to the printer port is desired.

### /DIA

The /DIA switch provides interactive dialog with the user, allowing the user to be prompted for each individual switch and switch value. Dialog for Professional-specific switches is listed below (the default answer is in brackets [ ]):

### /INS

Professional installation files contain a "Name" statement containing the name of the application. The name is defined by the user and may contain from 1 to 40 characters. The BLDODL utility prompts for this name as follows:

```
PLEASE ENTER APPLICATION NAME["task name"]:
```

where "task name" is the name of the task and is from 1 to 6 characters long.

If "task name" is desired as the name of the application, press <RET> and "task name" will be accepted as the application name. Otherwise, enter the application name desired and press <RET>. If more than 40 characters are entered, the cursor remains at the right margin and the terminal bell beeps. The delete key may be used to remove incorrect characters before <RET> is pressed; however, any other <control key> terminates input.



## DEVELOPING THE APPLICATION

### /CGL

The /CGL switch provides support of the CORE Graphics Library. The BLDODL utility asks:

DOES THIS APPLICATION USE CORE GRAPHICS <Y/N> [N]?

### /FMS

The /FMS switch provides support of clustered FMS on the Professional. The BLDODL utility asks:

DOES THIS APPLICATION USE FMS <Y/N> [N]?

### /POS

The /POS switch provides support of P/OS services specific to the Professional environment. The BLDODL utility asks:

DOES THIS APPLICATION USE P/OS SERVICES <Y/N> [N]?

When the user responds with "N" or <RET>, the following dialog will be skipped:

USE DEFAULTS FOR P/OS SERVICES <Y/N> [Y] ?

For a default CMD file that supports all P/OS functions, press <RET> and the "Y" response is taken by default. If a customized CMD file is desired, enter "N" and press <RET>. If "N" was entered, the following prompts appear :

DOES THIS APPLICATION USE :  
STATIC SINGLE CHOICE MENU(S) <Y/N> [N] ?  
DYNAMIC SINGLE CHOICE MENU(S) <Y/N> [N] ?  
MULTIPLE CHOICE MENU(S) <Y/N> [N] ?  
HELP TEXT OR HELP MENU(S) <Y/N> [N] ?  
FILE SPECIFICATION OR SELECTION <Y/N> [N] ?

Each of the above items appears singly and in order. To select an item, enter "Y" and press <RET>. Press <RET> to reject an option.

### /DEB

The /DEB switch provides support for the COBOL-81 symbolic debugger. The BLDODL utility asks:

DO YOU WANT TO INCLUDE THE INTERACTIVE DEBUGGER <Y/N> [N] ?

The BLDODL utility asks if redirection of the debugger to the printer port is desired:

REDIRECT DEBUGGER INPUT/OUTPUT TO PRINTER PORT <Y/N> [N] ?

## DEVELOPING THE APPLICATION

### 3.3.1.2 The Command (CMD) File

The CMD file created by the BLDODL utility, with no switches specified, for a program named "EXAMPL" looks like this:

```
#PAB COMMAND FILE CREATED ON 20-JUL-83 AT 14:09:28
EXAMPL/CP=EXAMPL/MP
TASK=EXAMPL
CLSTR=C81LIB,POSRES,RMSRES:RO
; ***** LUN assignments *****
GBLDEF=TT$EFN:001 ; terminal event flag
GBLDEF=MS$LUN:002 ; message frame file
GBLDEF=MN$LUN:000 ; menu frame file
GBLDEF=HL$LUN:000 ; help frame file
GBLDEF=WC$LUN:000 ; file selection / specification
GBLDEF=TT$LUN:003 ; terminal I/O
GBLDEF=G$LUN:000 ; CGL graphics
UNITS=003
ASG=TI:001:003
//
```

Each line in the file specifies a command or option to the LINK facility. In general, you can specify many options in a CMD file (see the RSX-11M/M-PLUS Task Builder Manual for complete details). Note that when you call certain P/OS services, you may need to edit the CMD file to alter information or include additional information depending on the service you are calling. Some P/OS services require buffer space, additional resident libraries, and additional LUN assignments. See the Tool Kit User's Guide for information specific to the service you plan to use.

Following are brief explanations of items in the CMD file created by BLD. By default the CMD file automatically sets minimum conditions for the P/OS environment.

1. Task Name - The CMD file assigns a task name to the task image using a command line of the form:

```
TASK=task-name
```

The output filename is the one to the left of the equals sign in the BLDODL command line. This filename is used as the task name. If you do not want the filename to be the task name, you must edit the CMD file. The value task-name is a 1- to 6-character name identifying the task; it is the installed name of the task.

2. Clusterable Libraries - Applications must reference the POSRES library as a shareable, clusterable library. The CMD file automatically clusters the PRO/Tool Kit COBOL-81 OTS with RMSRES and POSRES libraries, as shown on the line:

```
CLSTR = C81LIB,RMSRES,POSRES:RO
```

After POSRES, other libraries should be listed in the order of greatest use.

## DEVELOPING THE APPLICATION

3. Extend Section Commands - BLDODL does include some common extend section commands in the CMD file for allocating additional buffer space. However, you may be required to supply these commands if your application uses P/OS user interface services in the POSRES library which are not supported by the BLDODL utility. See the section on task building in Chapter 2 of the Tool Kit User's Guide for instructions on calculating buffer sizes for P/OS user interface services.
4. Logical Unit Numbers - P/OS user interface services require logical unit numbers (LUNs) to perform file I/O to menu, help, and message definition files. LUN assignments for P/OS I/O are made with global definition commands (GBLDEF) in the CMD file:

```
GBLDEF = MS$LUN:002      ; message frame file
```

If one type of P/OS service, such as help, is never used, specify 0 in the global definition line for message files, so that you will not get "undefined" errors at task build time. Do not omit any global definition lines. BLDODL assigns LUNs not always required by COBOL to 0.

5. Event Flags - P/OS user interface routines also require an event flag (EFN) to perform terminal I/O. The event flag assignment must not conflict with event flag assignments made for other purposes by the application. Language OTS routines use DIGITAL reserved event flags (flags 25-32). The event flag assigned with the TT\$EFN line should not be a DIGITAL reserved flag.
6. The two slash characters (//) signify the end of the CMD file.

### NOTE

The ASSIGN (ASG) and UNITS commands list decimal numbers. Extend section (EXTSCT) and global definition (GBLDEF) lines specify octal numbers.

#### 3.3.1.3 The Overlay Descriptor Language (ODL) File

The ODL file created by the BLDODL utility, with no switches specified, for a program named "EXAMPL" looks like this:

```
;MERGED ODL FILE CREATED ON 20-JUL-83 AT 14:09:28
@EXAMPL.SKL
SCOBJ$: .FCTR TESTER.OBJ
@LB:[1,1]RMSRLX.ODL
      .NAME RMS$TR
RMSTR$: .FCTR RMS$TR-RMSALL
RMS$: .FCTR RMSROT
SCLIB$: .FCTR LB:[1,1]C81LIB/LB
OBJRT$: .FCTR SCOBJ$-SCLIB$-RMS$
      .ROOT OBJRT$,RMSTR$
      .END
```

You may need to edit the ODL file (for example, if your application uses Communications Services). Refer to the Tool Kit User's Guide for details about editing the ODL file.

## DEVELOPING THE APPLICATION

### 3.3.1.4 The Installation (INS) File

The unedited installation (INS) file for a program named "EXAMPL", created by specifying the /INS switch (but not the /CGL switch) on the BLDODL utility command line looks like this:

```
! COBOL-81 INS FILE CREATED : 12-JUL-83 AT 09:35:32
NAME "EXAMPL"
FILE EXAMPL.TSK/DELETE
INSTALL [ZZSYS]C81LIB.TSK/LIBRARY
INSTALL [ZZSYS]C81LEIS.TSK/LIBRARY
INSTALL EXAMPL.TSK/TASK
RUN EXAMPL
```

Some P/OS services require additional lines in the INS file. See the Tool Kit User's Guide for information specific to the service you plan to use, and for a description of what should go into an INS file. In particular, note the differences between applications running under P/OS Diskette and those running under P/OS Hard Disk.

### 3.3.2 Building the Application Using LINK @

When you have the desired CMD and ODL files, use the PRO/Tool Kit LINK @ facility to create an application task image (file.TSK).

Invoke the PRO/Tool Kit LINK facility as follows:

```
$ LINK @filename
```

The @filename is the filename of the CMD file.

To build the application, LINK references the CMD file with the file name specified, and creates a TSK file.

### 3.3.3 Building the Application Using LINK/C81

For information on building the application using LINK/C81, refer to the COBOL-81 RSX-11M/M-PLUS User's Guide. In general, LINK/C81 may be used whenever your program does not use FMS, P/OS Services, or the CORE Graphics Library. The format of the LINK/C81 command is:

```
$ LINK/C81[/qualifier] file-spec[, ...] [/qualifier][, ...]
```

### 3.3.4 Writing the Installation (INS) File

The next step in the development cycle is to install the application on the Professional for final test runs. If you did not use the BLDODL utility and specify the /INS switch, you must create an installation (INS) file before you can install the application on the Professional. Refer to Section 3.3.1.4 for a description of the INS file.

## DEVELOPING THE APPLICATION

### 3.4 INSTALLING THE APPLICATION WITH FAST INSTALL

Use the Fast Install facility to install your new application with the P/OS menu structure. The Fast Install program source is located on the same diskette as the Application Diskette Builder. If Fast Install has not been installed, use the Install Application with Disk Services to install Fast Install before proceeding.

When you execute Fast Install, you are prompted to enter a directory name. Enter the name of the directory that contains your INS and TSK files (and any other required files). The name of the INS file must be identical to the directory name. Press DO when you have entered the directory name.

Fast Install now prompts you to select a menu group within which to install the new application. After you have chosen the menu group, you have an opportunity to change both the application name and the menu group name. Press DO when you are satisfied with both names.

Fast Install reports the success or failure of the installation process. Refer to the Tool Kit User's Guide for more information.

When your application is error free, run the Application Diskette Builder to create a final diskette. See the Tool Kit User's Guide for complete information on Professional Application Builder commands.

## CHAPTER 4

### RELEASE NOTES

This chapter contains information specific to this release of PRO/Tool Kit COBOL-81.

There are no known bugs in the software. There are no restrictions to the software as it is described in this document.





APPENDIX A  
SAMPLE APPLICATION

This chapter contains the source file (C81IVP.CBL) of the sample application used in Appendix B.

```
*
*      COBOL-81 Installation verification Procedure
*
*      NOTE: This program uses a routine CONIFL to convert
*             integer numbers to real numbers. CONIFL is in
*             the COBOL-81 runtime library and is documented
*             in the supplement. DETACH is also a COBOL-81
*             routine, and is also described in the supplement.
*
IDENTIFICATION DIVISION.
*
* LANGUAGE : COBOL-81, version 2.1
*
* DEVELOPMENT SYSTEM : PROFESSIONAL 350 with PRO/TOOL KIT COBOL-81
*
* TARGET MACHINE :
*                  PROFESSIONAL 350   {P/OS Hard Disk and P/OS Diskette}
*                  PROFESSIONAL 325   {P/OS Diskette}
*
* This program is targeted for the Professional -- input key acceptance
* conventions assume key codes not generated by a VT 52 or VT 100 series
* terminal -- DO, HELP, SELECT
*
*
* USAGE : This program is intended for demonstration or display purposes
*         only.
*
PROGRAM-ID. C81IVP.
ENVIRONMENT DIVISION.
DATA DIVISION.
```

# SAMPLE APPLICATION

## WORKING-STORAGE SECTION.

```

* ++++++
*
*      ACCEPT - DISPLAY DEMONSTRATION DATA
*
* -----
01 double-height-line.
    02 top-line.
        05 escape-code PIC 9(4) COMP VALUE 27.
        05 escape redefines escape-code.
            10 escape-ident PIC X.
            10 FILLER PIC X.
        05 top-line-code PIC XX VALUE "#3".
        05 message-text PIC X(15) VALUE "ldllsliltla11" .
        05 FILLER PIC X.
    02 bottom-line.
        05 escape-code PIC 9(4) COMP VALUE 27.
        05 escape redefines escape-code.
            10 escape-ident PIC X.
            10 FILLER PIC X.
        05 bottom-line-code PIC XX VALUE "#4".
        05 message-text PIC X(15) VALUE "ldllsliltla11" .

01 double-width-line.
    05 escape-code PIC 9(4) COMP VALUE 27.
    05 escape redefines escape-code.
        10 escape-ident PIC X.
        10 FILLER PIC X.
    05 double-width-line-code PIC XX VALUE "#6".
    05 message-text PIC X(8) VALUE "COBOL-81".

*      The above data structures demonstrate the use of DOUBLE WIDTH
*      and DOUBLE HEIGHT lines with the extended DISPLAY verb in COBOL 81.
*      Further explanation is available in the COBOL 81 USER'S GUIDE
*

77 ERROR-FLAG PIC 9 VALUE ZERO.
77 GRAND-TOTAL PIC 9(8)V99.
* Holds sum of all item totals

77 LINE-POINTER PIC 99.
* Index parameter used to indicate current line

77 ENTRY-POINTER PIC 99.
* Index parameter used to indicate current entry or item

77 LINE-NUMBER PIC 99.
* Line-number is used as an index to the line-data array

77 LAST-DEFINED-LINE PIC 99.

77 DUMMY PIC XXXX.
* Just an extra parameter used for ACCEPT statements {Entirely optional}

01 GL-LINE-NUM PIC 99.
    88 FIRST-LINE VALUE 4.
    88 LAST-LINE VALUE 22.

* Conditionals used to indicate out of field conditions

01 GL-COL-NUM PIC 99.
    88 PART-COL VALUE 5.
    88 COST-COL VALUE 23.
    88 QUAN-COL VALUE 41.

```

## SAMPLE APPLICATION

\* Conditionals used to indicate current field for accept statement  
 \* in conjunction with GL-LINE-NUM. For example, if GL-LINE-NUM = 3  
 \* and PART-COL = true then we would update the part number in the  
 \* first line data item -- See CHANGE-ITEM

```
01 TOTAL-FLAG          PIC XXXXX.
   88 COMPUTE-TOTAL    VALUE "TRUE".
```

```
01 START-FLAG          PIC XXXXX.
   88 START-RUN        VALUE "TRUE".
```

```
01 SELECTOR            PIC 99.
```

\* Case selector used with GO TO ..... DEPENDING ON ....

```
01 LINE-RECORD.
   05 LINE-LIMIT        PIC 99.
   05 LINE-DATA OCCURS 20.
       10 PART-NUMBER    PIC X(10).
       10 PART-COST      PIC 9(4)V99.
       10 PART-COST-SPECIFIED REDEFINES PART-COST
                               PIC X(6).
       10 PART-QUANTITY  PIC 9999.
       10 PART-QUANTITY-SPECIFIED REDEFINES PART-QUANTITY
                               PIC X(4).
       10 ITEM-TOTAL     PIC 9(6)V99.
```

\* The followings are conditional definitions enabling easy recognition  
 \* of the identity of the last key pressed. These definitions are not  
 \* necessary for operation of the program, but are provided for instructional  
 \* purposes

```
01 CONTROL-KEY.
   02 FIRST-CHAR-CONTROL-KEY          PIC X.
*   The first character is a recognition character required by P/OS;
*   the programmer may effectively ignore it

   02 REMAINING-CONTROL-KEY           PIC XXXX.
       88 UP-ARROW                     VALUE 'A'.
       88 DOWN-ARROW                   VALUE 'B'.
       88 RIGHT-ARROW                   VALUE 'C'.
       88 LEFT-ARROW                   VALUE 'D'.
       88 PF1                           VALUE 'P'.
       88 PF2                           VALUE 'Q'.
       88 PF3                           VALUE 'R'.
       88 PF4                           VALUE 'S'.
       88 ENTER-KEY                     VALUE 'M'.
       88 MAIN-SCREEN-KEY                VALUE '20~'.
       88 EXIT-KEY                      VALUE '21~'.
       88 HELP-KEY                      VALUE '28~'.
       88 FIND-KEY                      VALUE '1~'.
       88 SELECT-KEY                    VALUE '4~'.
       88 DO-KEY                        VALUE '29~'.
       88 INPUT-VALID-KEY                VALUE 'A','B','C','D','P','Q','R','S'.
       88 MENU-VALID-KEY                 VALUE 'A','B','28~','21~'.
       88 MOVEMENT-KEY                   VALUE 'A','B','C','D'.
       88 VALID-DEMO-KEY                  VALUE '29~','A','B','C','D','21~','4~',
                                           '28~','20~'.
```

\* The above 88 level entries identify key labels relating to the  
 \* respective key codes. INPUT-VALID-KEY and MENU-VALID-KEY are used  
 \* to determine if a key pressed is valid response fro a given situation--  
 \* whether the user responded to a menu item or is entering or manipulating  
 \* data with the demo application.

# SAMPLE APPLICATION

```

* ++++++
*
*      GRAPHICS DATA
*
* -----

01 X-ARRAY.
    02 CURVE-X          PIC      X(4) OCCURS 20 TIMES.
01 Y-ARRAY.
    02 CURVE-Y          PIC      X(4) OCCURS 20 TIMES.
01      TEXT-CODE      PIC      S9(4) COMP VALUE 16.
01      NEW-FRAME      PIC      S9(4) COMP VALUE 92.
01      SET-ORIGIN     PIC      S9(4) COMP VALUE 86.
01      SET-WINDOW     PIC      S9(4) COMP VALUE 80.
01      MOVE-ABS-2     PIC      S9(4) COMP VALUE 1.
01      MOVE-REL-2     PIC      S9(4) COMP VALUE 2.
01      LINE-ABS-2     PIC      S9(4) COMP VALUE 4.
01      RECTANGLE-ABS-2 PIC      S9(4) COMP VALUE 10.
01      SET-MARKER-SYMBOL PIC      S9(4) COMP VALUE 37.
01      MARKER-REL-2   PIC      S9(4) COMP VALUE 34.
01      SET-LINestyle  PIC      S9(4) COMP VALUE 12.
01      ARC-ABS-2      PIC      S9(4) COMP VALUE 39.
01      CURVE-REL-2    PIC      S9(4) COMP VALUE 41.
01      SET-CHARJUST   PIC      S9(4) COMP VALUE 26.
01      ANGLE          PIC      S9(4) COMP VALUE 360.
*
*      Prompt array
*
01      RESARR.
    02 RES      PIC 9(4) COMP OCCURS 6 TIMES.
01      RESCNT  PIC 9(4) COMP.
*
*      Numeric constant data.
*
01 INITIALIZE-CORE      PIC S9(4) COMP VALUE 90.
01 NUM-CONST-NEG-8      PIC S9(4) COMP VALUE -8.
01 NUM-CONST-NEG-7      PIC S9(4) COMP VALUE -7.
01 NUM-CONST-NEG-6      PIC S9(4) COMP VALUE -6.
01 NUM-CONST-NEG-5      PIC S9(4) COMP VALUE -5.
01 NUM-CONST-NEG-4      PIC S9(4) COMP VALUE -4.
01 NUM-CONST-NEG-1      PIC S9(4) COMP VALUE -1.
01 NUM-CONST-0          PIC S9(4) COMP VALUE 0.
01 NUM-CONST-1          PIC S9(4) COMP VALUE 1.
01 NUM-CONST-2          PIC S9(4) COMP VALUE 2.
01 NUM-CONST-4          PIC S9(4) COMP VALUE 4.
01 NUM-CONST-5          PIC S9(4) COMP VALUE 5.
01 NUM-CONST-6          PIC S9(4) COMP VALUE 6.
01 NUM-CONST-7          PIC S9(4) COMP VALUE 7.
01 NUM-CONST-8          PIC S9(4) COMP VALUE 8.
01 NUM-CONST-16         PIC S9(4) COMP VALUE 16.
01 NUM-CONST-17         PIC S9(4) COMP VALUE 17.
01 NUM-CONST-18         PIC S9(4) COMP VALUE 18.
01 NUM-CONST-19         PIC S9(4) COMP VALUE 19.
01 NUM-CONST-20         PIC S9(4) COMP VALUE 20.
01 NUM-CONST-50         PIC S9(4) COMP VALUE 50.
01 NUM-CONST-72         PIC S9(4) COMP VALUE 72.
*
*      String constant data.
*
01 STR-CONST-CORE      PIC X(4) VALUE 'CORE'.
01 STR-CONST-GRAPHICS  PIC X(8) VALUE 'Graphics'.
01 STR-CONST-LIB       PIC X(7) VALUE 'Library'.
01 STR-CONST-TITL      PIC X(4) VALUE 'TITL'.
01 STR-CONST-CLRB      PIC X(4) VALUE 'CLRB'.
01 STR-CONST-GHLP      PIC X(4) VALUE 'GHLP'.

```

# SAMPLE APPLICATION

```

01 STR-CONST-PRMT      PIC X(4) VALUE "PRMT".
01 STR-CONST-TEXT01    PIC X(6) VALUE "TEXT01".
01 STR-CONST-TEXT02    PIC X(6) VALUE "TEXT02".
01 STR-CONST-SPACES    PIC X(80) VALUE SPACE.
01 TEXT01              PIC X(72) VALUE
    "Use the arrow keys and Select key to select any combination".
01 TEXT02              PIC X(72) VALUE
    "or all of the following Graphics functions".
01 GHLP                PIC X(8)  VALUE "GRDEMO".
01 PRMT                PIC X(72) VALUE
    "Make your selection(s) and Press DO:(EXIT returns to main menu)".
01 TITL                PIC X(50) VALUE
    "COBOL-81 and CORE Graphics Library Demonstration".

*
*      Status
*
01      STATUS-ARRAY.
    02 STATUS-WORD-1      PIC 9(4) COMP.
        88 DO-KEY VALUE 1.
        88 ERROR-COND VALUE 0.
    02 STATUS-WORD-2      PIC 9(4) COMP.

*
*      Menu options.
*
01 OPTION-LIST.
    02 OPTION PIC X(20) OCCURS 6 TIMES.

*
*      Real number storage for calls to graphics routines.
*      (Each must be 2 words long to hold 32 bit floating number).
*
01 REAL1  PIC X(4).
01 REAL2  PIC X(4).
01 REAL3  PIC X(4).
01 REAL4  PIC X(4).
01 REAL5  PIC X(4).
*
*      PERFORM VARYING loop counter
*
01 I      PIC S9(4) COMP.
01 J      PIC S9(4) COMP.
01 K      PIC S9(4) COMP.
01 L      PIC S9(4) COMP.

* ++++++
*
*      END OF GRAPHICS DEMO DATA
*
* -----

```

PROCEDURE DIVISION.

STARTUP SECTION.

```

*      Display the company logo and prompt the user to select either
*          i) the PRO graphics demo
*          ii) ACCEPT / DISPLAY demo
*          iii) EXIT PROGRAM

```

PROCESS-LOGO.

```

    DISPLAY " "
    AT LINE 1
    AT COLUMN 1
    ERASE TO END OF SCREEN.

```

## SAMPLE APPLICATION

DISPLAY top-line OF double-height-line  
AT LINE 2  
AT COLUMN 13  
REVERSED  
BOLD.

DISPLAY bottom-line OF double-height-line  
AT LINE 3  
AT COLUMN 13  
REVERSED  
BOLD.

DISPLAY double-width-line  
AT LINE 5  
AT COLUMN 17  
BOLD.

DISPLAY-MENU-MASK.

DISPLAY "PRO Graphics DEMO"  
LINE 8  
COLUMN 34.

DISPLAY "Accept/Display DEMO"  
LINE 10  
COLUMN 34.

DISPLAY "Quit / Exit Program"  
LINE 12  
COLUMN 34.

MOVE 12 TO GL-LINE-NUM.

DISPLAY "==>"  
LINE GL-LINE-NUM  
COLUMN 28  
BOLD  
REVERSED.

PERFORM  
MENU-GENERATOR  
UNTIL

DO-KEY OF REMAINING-CONTROL-KEY  
OR EXIT-KEY  
OR PF1  
OR PF2.

IF EXIT-KEY OR PF2 THEN  
GO TO EXIT-PROGRAM.

MOVE SPACES TO CONTROL-KEY.

SUBTRACT 6 FROM GL-LINE-NUM GIVING GL-LINE-NUM.

DIVIDE GL-LINE-NUM BY 2 GIVING SELECTOR.

GO TO  
GRAPHICS-DEMO  
ACCEPT-DISPLAY-DEMO  
EXIT-PROGRAM

DEPENDING ON SELECTOR.

## SAMPLE APPLICATION

MENU-GENERATOR.

ACCEPT DUMMY CONTROL KEY IN CONTROL-KEY  
LINE GL-LINE-NUM  
COLUMN 27  
WITH NO ECHO.

DISPLAY " " "  
LINE GL-LINE-NUM  
COLUMN 28.

IF NOT MENU-VALID-KEY OR HELP-KEY

THEN

DISPLAY "<up arrow> move up, <down arrow> move down,  
<DO> accept, <EXIT> exit program"

LINE 23  
COLUMN 1  
BOLD

ELSE

IF UP-ARROW

THEN

SUBTRACT 2 FROM GL-LINE-NUM GIVING GL-LINE-NUM.

IF GL-LINE-NUM LESS THAN 8

THEN

MOVE 12 TO GL-LINE-NUM.

IF DOWN-ARROW

THEN

ADD 2, GL-LINE-NUM GIVING GL-LINE-NUM.

IF GL-LINE-NUM GREATER THAN 12

THEN

MOVE 8 TO GL-LINE-NUM.

DISPLAY "==" "  
LINE GL-LINE-NUM  
COLUMN 28  
BOLD  
REVERSED.

EXIT-PROGRAM.

DISPLAY " "  
AT LINE 1  
AT COLUMN 1  
ERASE TO END OF SCREEN.  
STOP RUN.

CLEAR-SCREEN.

DISPLAY " "  
LINE 1  
COLUMN 1  
ERASE TO END OF SCREEN.



## SAMPLE APPLICATION

```
*****  
*  
*  
*      >>>>>>>>>>>> BEGIN ACCEPT DISPLAY DEMO  
*  
*  
*****
```

ACCEPT-DISPLAY-DEMO SECTION.

ACCEPT-DISPLAY-DEMO.

MOVE ALL "?" TO LINE-RECORD.

MOVE 4 TO LAST-DEFINED-LINE.

```
MOVE 'FALSE' TO TOTAL-FLAG.
```

```
MOVE "TRUE" TO START-FLAG.
```

GO TO DEMO-DRIVER.

CHANGE-ITEM.

```
* Change-item accepts data from the user based on the column and line
* position of the cursor. Cursor position is determined by examination
* of the conditionals defined under GL-COL-NUM {global-column-number}
*
* GL-LINE-NUM is an indirect index into the LINE-DATA array, relating
* directly to the current line position of the cursor. The line
* position of the cursor may vary between 4 and 23, while the array
* is indexed from 1 to 20. Thus, GL-LINE-NUM - 3 provides an index to
* the LINE-DATA array.
```

```

DISPLAY * *
  LINE GL-LINE-NUM
  COLUMN 1
  ERASE LINE.

```

```
MOVE GL-LINE-NUM TO LINE-POINTER.
```

```
MOVE "FALSE" TO TOTAL-FLAG.
```

PERFORM DISPLAY-ENTRY.

SUBTRACT 3 FROM GL-LINE-NUM GIVING LINE-NUMBER.

```
* Compute the array index by subtracting 3 from the global-line-number
```

```
IF GL-LINE-NUM IS GREATER THAN LAST-DEFINED-LINE
  THEN
```

MOVE GL-LINE-NUM TO LAST-DEFINED-LINE.

```
* Dynamically expand our range of lines as the user defines succeeding
* entries.
```

# SAMPLE APPLICATION

```

IF PART-COL
THEN
    ACCEPT PART-NUMBER ( LINE-NUMBER )
    LINE GL-LINE-NUM
    COLUMN 5
    PROTECTED
    REVERSED
    WITH CONVERSION
    CONTROL KEY IN CONTROL-KEY
*      Accept a new part number from the user -- at the specified line
*      protected so the user can't overwrite other fields.

    DISPLAY PART-NUMBER ( LINE-NUMBER )
    LINE GL-LINE-NUM
    COLUMN 5
    BOLD
*      Display the new data {in reverse video} so the user can view
*      the changes. Notice that WITH CONVERSION is not used here
*      because the data type is ALPHA NUMERIC

ELSE

    IF COST-COL
    THEN
        ACCEPT PART-COST(LINE-NUMBER)
        LINE GL-LINE-NUM
        COLUMN 23
        PROTECTED
        REVERSED
        WITH CONVERSION
        CONTROL KEY IN CONTROL-KEY
        ON EXCEPTION
            GO TO BAD-DATA.
*      Accept a new part cost from the user -- at the specified line
*      protected so the user can't overwrite other fields.

        IF PART-COST-SPECIFIED ( LINE-NUMBER ) = ALL '?'
        THEN
            DISPLAY "???????"
            LINE GL-LINE-NUM
            COLUMN 23
        ELSE
            DISPLAY PART-COST(LINE-NUMBER)
            LINE GL-LINE-NUM
            COLUMN 23
            BOLD
            WITH CONVERSION.
*      Display the new data in reverse video -- WITH CONVERSION
*      since we are dealing with REAL {floating} data type

        IF QUAN-COL
        THEN
            ACCEPT PART-QUANTITY(LINE-NUMBER)
            LINE GL-LINE-NUM
            COLUMN 41
            PROTECTED
            REVERSED
            WITH CONVERSION
            CONTROL KEY IN CONTROL-KEY
            ON EXCEPTION
                GO TO BAD-DATA.
*      Accept a new part quantity from the user -- at the specified line
*      protected so the user can't overwrite other fields.

```

# SAMPLE APPLICATION

```

IF PART-QUANTITY-SPECIFIED ( LINE-NUMBER ) = ALL "?"
THEN
    DISPLAY "?????"
    LINE GL-LINE-NUM
    COLUMN 41
ELSE
    DISPLAY PART-QUANTITY ( LINE-NUMBER )
    LINE GL-LINE-NUM
    COLUMN 41
    BOLD
    WITH CONVERSION.
*      Display the new data in reverse video -- WITH CONVERSION
*      since we are dealing with REAL (floating) data type

GO TO GET-KEY-2.

*      >>>>>>>> END OF CHANGE-ITEM <<<<<<<<<

MOVEMENT.

* Movement interprets movement commands from the user, computing new cursor
* positions and resolving any OUT-OF-RANGE conditions by wrapping the cursor.

IF UP-ARROW
THEN
    IF NOT FIRST-LINE
    THEN
        SUBTRACT 1 FROM GL-LINE-NUM GIVING GL-LINE-NUM
    ELSE
        MOVE 22 TO GL-LINE-NUM.
*      If the user types an up-arrow and we're not at the top line then
*      subtract 1 from the line-number. If we're at the top line, then
*      wrap him up to the last line. Note that line movement is indep-
*      endant of column movement.

IF DOWN-ARROW
THEN
    IF NOT LAST-LINE
    THEN
        ADD GL-LINE-NUM , 1 GIVING GL-LINE-NUM
    ELSE
        MOVE 4 TO GL-LINE-NUM.
*      If the user types a down-arrow and we're not at the bottom line
*      then add 1 to the line-number. If we're at the bottom line, then
*      wrap him up to the first line

IF LEFT-ARROW
THEN
    IF NOT PART-COL
    THEN
        SUBTRACT 18 FROM GL-COL-NUM GIVING GL-COL-NUM
    ELSE
        MOVE 41 TO GL-COL-NUM.
*      If the user types a left-arrow and we're not at the part column
*      then move to the previous column. If we're at the part column, then
*      wrap him up to the quantity column

```

# SAMPLE APPLICATION

```

IF RIGHT-ARROW
  THEN
    IF NOT QUAN-COL
      THEN
        ADD GL-COL-NUM , 18 GIVING GL-COL-NUM
      ELSE
        MOVE 5 TO GL-COL-NUM.
*   If the user types a right-arrow and we're not at the quantity
*   column then move to the next column. If we're at the quantity
*   column then wrap the user to the part column

GO TO GET-KEY.
*   Display the cursor at the new position

*   >>>>>>> END OF MOVEMENT <<<<<<<<

DISPLAY-ENTRY.

MOVE 0 TO ERROR-FLAG.

SUBTRACT 3 FROM LINE-POINTER GIVING ENTRY-POINTER.

DISPLAY PART-NUMBER ( ENTRY-POINTER )
  LINE LINE-POINTER
  COLUMN 5.

IF (PART-COST-SPECIFIED (ENTRY-POINTER) = ALL "?")
  THEN
    DISPLAY PART-COST-SPECIFIED ( ENTRY-POINTER )
      LINE LINE-POINTER
      COLUMN 23
    MOVE 1 TO ERROR-FLAG
  ELSE
    DISPLAY PART-COST ( ENTRY-POINTER )
      LINE LINE-POINTER
      COLUMN 23
      WITH CONVERSION.

IF (PART-QUANTITY-SPECIFIED (ENTRY-POINTER) = ALL "?")
  THEN
    DISPLAY PART-QUANTITY-SPECIFIED ( ENTRY-POINTER )
      LINE LINE-POINTER
      COLUMN 41
    MOVE 1 TO ERROR-FLAG
  ELSE
    DISPLAY PART-QUANTITY ( ENTRY-POINTER )
      LINE LINE-POINTER
      COLUMN 41
      WITH CONVERSION.

IF ERROR-FLAG = 0
  THEN
    MULTIPLY PART-COST ( ENTRY-POINTER )
      BY PART-QUANTITY ( ENTRY-POINTER )
      GIVING ITEM-TOTAL ( ENTRY-POINTER )

    ADD GRAND-TOTAL , ITEM-TOTAL ( ENTRY-POINTER )
      GIVING GRAND-TOTAL

    DISPLAY ITEM-TOTAL ( ENTRY-POINTER )
      LINE LINE-POINTER
      COLUMN 53
      WITH CONVERSION.

```

## SAMPLE APPLICATION

```
IF COMPUTE-TOTAL AND ERROR-FLAG = 1
  THEN
    DISPLAY " "
    LINE LINE-POINTER
    COLUMN 1
    ERASE LINE
    DISPLAY "**** INVALID DATA IN THIS LINE ****"
    LINE LINE-POINTER
    COLUMN 10
    BOLD
    BLINKING.
```

HELP.

```
DISPLAY " "
  LINE 1
  COLUMN 1
  ERASE TO END OF SCREEN.

DISPLAY "COMMAND OPTIONS : "
  LINE 2
  COLUMN 8
  BOLD.

DISPLAY "Movement : "
  LINE 4
  COLUMN 12.

DISPLAY "Use arrow keys to move in specified direction "
  LINE 5
  COLUMN 16.

DISPLAY "Change entry : "
  LINE 7
  COLUMN 12.

DISPLAY "Use SELECT key to change current field -- editing and inserting"
  LINE 8
  COLUMN 16.

DISPLAY "Compute totals : "
  LINE 10
  COLUMN 12.

DISPLAY "Use DO key to compute line totals and grand total "
  LINE 11
  COLUMN 16.

DISPLAY "Exit / Return to Main Menu : "
  LINE 13
  COLUMN 12.

DISPLAY "Use EXIT key to exit or MAIN SCREEN to return to Main Menu"
  LINE 14
  COLUMN 16.

DISPLAY "Help : "
  LINE 16
  COLUMN 12.

DISPLAY "Use HELP key to display this screen"
  LINE 17
  COLUMN 16.
```

# SAMPLE APPLICATION

```

DISPLAY ">>>>>> Press <RETURN> to continue <<<<<<"
LINE 22
COLUMN 18
BOLD.

```

```

ACCEPT DUMMY
LINE 23
COLUMN 40
WITH NO ECHO.

```

DEMO-DRIVER.

```

MOVE 4 TO GL-LINE-NUM.
MOVE 5 TO GL-COL-NUM.

```

NEW-SCREEN.  
MASK-1.

\* Clear the screen and display the input screen

PERFORM CLEAR-SCREEN.

```

DISPLAY "  part number      cost each      quantity"
LINE 2
COLUMN 3.

```

```

DISPLAY "  -----      -----      -----"
LINE 3
COLUMN 3.

```

\* >>>>>>> END OF MASK-1 <<<<<<<<

IF NOT START-RUN  
THEN

MOVE 0 TO GRAND-TOTAL

\* Initialize the grand-total

PERFORM DISPLAY-ENTRY VARYING LINE-POINTER

FROM 4

BY 1

UNTIL

LINE-POINTER > LAST-DEFINED-LINE

IF COMPUTE-TOTAL

THEN

```

DISPLAY "=====
LINE LAST-DEFINED-LINE PLUS 1
COLUMN 51

```

```

DISPLAY "  total : "
LINE LAST-DEFINED-LINE PLUS 2
COLUMN 40

```

```

DISPLAY GRAND-TOTAL
LINE LAST-DEFINED-LINE PLUS 2
COLUMN 51
WITH CONVERSION

```

# SAMPLE APPLICATION

```
MOVE 4 TO GL-LINE-NUM
MOVE 5 TO GL-COL-NUM
MOVE 'FALSE' TO START-FLAG.
MOVE 'FALSE' TO TOTAL-FLAG.
```

GET-KEY.

```
SUBTRACT 3 FROM GL-COL-NUM GIVING GL-COL-NUM.
```

```
DISPLAY "-->"
LINE GL-LINE-NUM
COLUMN GL-COL-NUM .
```

```
ACCEPT DUMMY CONTROL KEY IN CONTROL-KEY
LINE GL-LINE-NUM
COLUMN GL-COL-NUM
WITH NO ECHO.
```

```
DISPLAY " "
LINE GL-LINE-NUM
COLUMN GL-COL-NUM .
```

```
DISPLAY " "
LINE LAST-DEFINED-LINE PLUS 1
COLUMN 1
ERASE TO END OF SCREEN.
```

```
ADD GL-COL-NUM , 3 GIVING GL-COL-NUM.
```

GET-KEY-2.

```
IF NOT VALID-DEMO-KEY
THEN
GO TO GET-KEY.
```

```
IF MOVEMENT-KEY
THEN
GO TO MOVEMENT
ELSE
IF HELP-KEY
THEN
GO TO HELP
ELSE
IF SELECT-KEY
THEN
MOVE 'FALSE' TO START-FLAG
GO TO CHANGE-ITEM
ELSE
IF DO-KEY OF REMAINING-CONTROL-KEY
THEN
MOVE 'TRUE' TO TOTAL-FLAG
GO TO NEW-SCREEN
ELSE
IF MAIN-SCREEN-KEY OR EXIT-KEY
THEN
GO TO PROCESS-LOGO.
```

```
GO TO GET-KEY.
```

## SAMPLE APPLICATION

BAD-DATA.

DISPLAY "Please enter only NUMERIC data in this field"

LINE 22

COLUMN 17

BOLD

BLINKING

ERASE TO END OF SCREEN.

DISPLAY ">>>>> Press <RETURN> to continue <<<<<"

LINE 23

COLUMN 18

BOLD.

ACCEPT DUMMY

LINE 24

COLUMN 40

WITH NO ECHO.

GO TO GET-KEY.

```
*****
*
*
*      <<<<<<<<< END OF ACCEPT DISPLAY DEMONSTRATION PROGRAM
*
*
*****
```



# SAMPLE APPLICATION

```

*****
*
*
*       >>>>>>>>> BEGIN PROFESSIONAL GRAPHICS DEMONSTRATION PROGRAM
*
*
*****
GRAPHICS SECTION.
GRAPHICS-DEMO.

*
*       Initialize the graphics system.
*

CALL "CGL" USING INITIALIZE-CORE.

*
*       Pack and display a multiple choice menu.
*

CALL "MPACK" USING
        STATUS-ARRAY
        STR-CONST-CLRB   NUM-CONST-4
        STR-CONST-TITL   NUM-CONST-4 TITL   NUM-CONST-50
        STR-CONST-TEXT01 NUM-CONST-6 TEXT01 NUM-CONST-72
        STR-CONST-TEXT02 NUM-CONST-6 TEXT02 NUM-CONST-72
        STR-CONST-GHLP   NUM-CONST-4 GHLP   NUM-CONST-8
        STR-CONST-PRMT   NUM-CONST-4 PRMT   NUM-CONST-72.

*
*       Pass all the options to MMENU.
*

MOVE "Draw a rectangle"      TO OPTION(1).
MOVE "Draw straight lines"   TO OPTION(2).
MOVE "Draw points"           TO OPTION(3).
MOVE "Draw curves"           TO OPTION(4).
MOVE "Draw a circle"         TO OPTION(5).
MOVE "Draw graphics text"    TO OPTION(6).

LOOP.
CALL "MMENU" USING STATUS-ARRAY OPTION-LIST
        NUM-CONST-20 NUM-CONST-6 NUM-CONST-6
        RESCNT RESARR
        NUM-CONST-0 STR-CONST-SPACES NUM-CONST-20.
IF NOT DO-KEY OF STATUS-WORD-1 OF STATUS-ARRAY
THEN
        GO TO PROCESS-LOGO.

IF RESCNT = 0
THEN
        GO TO LOOP.

*
*       Set up the screen
*

CALL "CGL" USING NEW-FRAME.
CALL "CGL" USING SET-ORIGIN NUM-CONST-0.

*
*       Use CONIFL to convert integers to floating point numbers.
*

CALL "CONIFL" USING
        NUM-CONST-NEG-1 REAL1
        NUM-CONST-19   REAL2
        NUM-CONST-NEG-8 REAL3
        NUM-CONST-8    REAL4.
CALL "CGL" USING SET-WINDOW REAL1 REAL2 REAL3 REAL4.
CALL "CGL" USING MOVE-ABS-2 REAL1 REAL3.

```

# SAMPLE APPLICATION

```

*
*      Now do everything requested
*
      PERFORM DO-IT VARYING I FROM 1 BY 1 UNTIL I > RESCNT.
      DISPLAY "Hit return to continue" LINE 23 COLUMN 2.
      ACCEPT REAL1.

*
*      Detach the terminal
*
      CALL "DETACH".
      GO TO LOOP.
DO-IT SECTION.
PO,
      GO TO
          DRAW-RECTANGLE
          DRAW-LINE
          DRAW-POINTS
          DRAW-CURVE
          DRAW-CIRCLE
          DRAW-TEXT
          DEPENDING ON RES(I).
DRAW-RECTANGLE.
      CALL "CONIFL" USING
          NUM-CONST-19 REAL1
          NUM-CONST-8 REAL2.
      CALL "CGL" USING RECTANGLE-ABS-2 REAL1 REAL2.
      GO TO DO-IT-EXIT.
DRAW-LINE.
      CALL "CONIFL" USING
          NUM-CONST-0 REAL1
          NUM-CONST-6 REAL2
          NUM-CONST-NEG-6 REAL3
          NUM-CONST-17 REAL4.

      CALL "CGL" USING MOVE-ABS-2 REAL1 REAL2.
      CALL "CGL" USING LINE-ABS-2 REAL1 REAL3.
      CALL "CGL" USING MOVE-ABS-2 REAL4 REAL1.
      CALL "CGL" USING LINE-ABS-2 REAL1 REAL1.
      GO TO DO-IT-EXIT.
DRAW-POINTS.
      CALL "CONIFL" USING
          NUM-CONST-0 REAL1
          NUM-CONST-1 REAL2
          NUM-CONST-NEG-7 REAL3.
      CALL "CGL" USING MOVE-ABS-2 REAL1 REAL1.
      CALL "CGL" USING SET-MARKER-SYMBOL NUM-CONST-2 NUM-CONST-0
      PERFORM LOOP2 VARYING J FROM 1 BY 1 UNTIL J > 17.
      CALL "CGL" USING MOVE-ABS-2 REAL1 REAL3.
      PERFORM LOOP3 VARYING J FROM 1 BY 1 UNTIL J > 13.
      GO TO DO-IT-EXIT.
LOOP2.
      CALL "CGL" USING MARKER-REL-2 REAL2 REAL1.
LOOP3.
      CALL "CGL" USING MARKER-REL-2 REAL1 REAL2.
DRAW-CURVE.
      MOVE 1 TO K.
      MOVE 4 TO L.
      PERFORM FILL-X VARYING J FROM 0 BY 1 UNTIL J > 19.
      CALL "CGL" USING
          MOVE-ABS-2 CURVE-X(1) CURVE-Y(1).
      CALL "CGL" USING
          CURVE-REL-2 X-ARRAY Y-ARRAY NUM-CONST-18 NUM-CONST-0.
      GO TO DO-IT-EXIT.

```

# SAMPLE APPLICATION

```

FILL-X,
  CALL "CONIFL" USING
      J CURVE-X(K)
      L CURVE-Y(K),
  COMPUTE L = - L,
  ADD 1 TO K,
DRAW-CIRCLE,
  CALL "CONIFL" USING
      NUM-CONST-NEG-5      REAL1
      NUM-CONST-5         REAL2
      NUM-CONST-18        REAL3
      NUM-CONST-16        REAL4
      NUM-CONST-NEG-6     REAL5,

  CALL "CGL" USING
      SET-LINESTYLE
      NUM-CONST-1 NUM-CONST-0 NUM-CONST-0,

  CALL "CGL" USING
      MOVE-ABS-2 REAL3 REAL1,

  CALL "CGL" USING
      ARC-ABS-2
      REAL4 REAL1 ANGLE,

  GO TO DO-IT-EXIT,
DRAW-TEXT,
  CALL "CGL" USING SET-CHARJUST NUM-CONST-2 NUM-CONST-2,
  CALL "CONIFL" USING
      NUM-CONST-16      REAL1
      NUM-CONST-NEG-4   REAL2
      NUM-CONST-0       REAL3
      NUM-CONST-NEG-1   REAL4,

  CALL "CGL" USING MOVE-ABS-2 REAL1 REAL2,
  CALL "CGL" USING
      TEXT-CODE STR-CONST-CORE NUM-CONST-4,
  CALL "CGL" USING MOVE-REL-2 REAL3 REAL4,
  CALL "CGL" USING
      TEXT-CODE STR-CONST-GRAPHICS NUM-CONST-8,
  CALL "CGL" USING MOVE-REL-2 REAL3 REAL4,
  CALL "CGL" USING
      TEXT-CODE STR-CONST-LIB NUM-CONST-7,

  GO TO DO-IT-EXIT,
DO-IT-EXIT,
  EXIT,

*****
*
*
*   <<<<<<<<<< END OF PROFESSIONAL GRAPHICS DEMONSTRATION PROGRAM
*
*
*****

```

## APPENDIX B

### APPLICATION DEVELOPMENT EXAMPLE

This appendix contains an application development example that serves as a hands-on introduction to COBOL-81 application development on the Professional, and shows how a COBOL-81 application uses Professional facilities.

This example uses the sample program listed in Appendix A. The sample program source file, C81IVP.CBL, has been placed in DW1:[COBOL] as part of the installation procedure. You can work through the sample application in DW1:[COBOL], or copy the file to another area and work there. The assumption is made here that the sample application will be carried out in DW1:[COBOL], and that you have made DW1:[COBOL] your default directory.

Follow these steps to work through the sample application:

- Convert the source file C81IVP.CBL from terminal format to ANSI format, using the REFORMAT utility, RFM. Note that this step is not actually required to build the application, but it ensures that the REFORMAT utility has been correctly copied from the kit.
- Compile the source file, using the PRO/Tool Kit COBOL-81 compiler.
- Build the command (CMD) and overlay descriptor language (ODL) files, using the BLDODL utility.
- Build the application task image file, using LINK, the PRO/Tool Kit LINK utility.
- Create a directory to receive the application files.
- Install the sample application.
- Execute the sample application.

Each of these steps is described in the sections that follow.

#### B.1 USING THE REFORMAT UTILITY TO CONVERT THE SOURCE FILE TO ANSI FORMAT

Note that this step is included here simply to test the REFORMAT utility. The compiler will accept source code in either terminal format or ANSI format. (C81IVP.CBL is initially in terminal format.)

To invoke the REFORMAT Utility, type:

```
$ RUN RFM
```

## APPLICATION DEVELOPMENT EXAMPLE

REFORMAT executes and prompts you with this message:

```
$ REFORMAT - ANSI-to-terminal conversion mode [Y/N] ?
```

Answer "N" and press the RETURN key.

REFORMAT then asks for input and output file specifications, and the contents of the identification area:

```
REFORMAT - Terminal-format input file spec :
REFORMAT -      ANSI-format output file spec :
REFORMAT -      Columns 73 to 80:
```

Use "C81IVP.CBL" for the input file, "C81IVP.ANS" for the output file, and any string of eight characters for the identification area.

Type <CTRL/Z> to exit when the conversion is completed.

### B.2 COMPILING THE SOURCE FILE

The next step is to compile the ANSI-formatted source file C81IVP.ANS. To invoke the PRO/Tool Kit COBOL-81 compiler, C81, enter the following command line:

```
$ COBOL/ANSI/DEBUG/LIST C81IVP.ANS
```

This command produces both object and listing files. The /ANSI qualifier is required because the source file has been converted to ANSI format in the first step. Generally you would create your source files in terminal format and, therefore, would not need the /ANSI qualifier. The /DEBUG qualifier is not normally needed; it has been included to test that the symbolic debugger files are correctly installed.

### B.3 BUILDING THE COMMAND AND OVERLAY DESCRIPTOR LANGUAGE FILES

After compiling the sample source file, use BLD, the BLDODL utility, to build the command (CMD) file, the overlay descriptor language (ODL) file, and the installation (INS) file. The CMD and ODL files are required by the PRO/Tool Kit LINK utility to define how libraries are referenced, and to specify special-purpose buffers, logical unit numbers (LUNs), and event flags (EFNs). The INS file is used by the Fast Install application to install the C81IVP task images.

The file C81IVP.SKL, which is created by the compiler, is the input for the BLDODL utility. The /DEB switch is used here only to ensure that the debugger has been correctly installed.

To invoke the BLDODL utility, enter the following command lines:

```
$ RUN BLD
```

If you have a terminal attached to the printer port, enter:

```
BLD>C81IVP = C81IVP/DEB:RED/CGL/POS:SSM:MCM/INS:"COBOL-81 SAMPLE"
```

/DEB:RED directs the symbolic debugger I/O to the terminal attached to the printer port.

If you do not have a terminal attached to the printer port, enter:

```
BLD>C81IVP = C81IVP/DEB/CGL/POS:SSM:MCM/INS:"COBOL-81 SAMPLE"
```

## APPLICATION DEVELOPMENT EXAMPLE

Press <CTRL/Z> to exit the BLDODL utility.

Using the OBJ and SKL files supplied by the compiler, the BLDODL utility creates the following three files:

1. The CMD (command) file:

```
;PAB COMMAND FILE CREATED ON 12-JUL-83 AT 09:35:32
C81IVP/FU/CP,,C81IVP=C81IVP/MP
TASK=C81IVP
CLSTR=C81LIB,CGLEIS,POSRES,RMSRES:RO
; ***** buffer allocation *****
EXTSCT=MN$BUF:4540 ; static single choice menu
EXTSCT=MM$BUF:1000 ; multi choice menu
; ***** LUN assignments *****
GBLDEF=TT$EFN:001 ; terminal event flag
; LUN 002 implicitly allocated to DEBUGGER I/O redirection *
GBLDEF=MS$LUN:004 ; message frame file
GBLDEF=MN$LUN:005 ; menu frame file
GBLDEF=HL$LUN:000 ; help frame file
GBLDEF=WC$LUN:000 ; file selection / specification
GBLDEF=TT$LUN:006 ; terminal I/O
GBLDEF=G$LUN:007 ; CGL graphics
UNITS=007
ASG=TI:001:006:007
ASG=SY:004
GBLPAT=MACSUB:$UTI:002 ; redirect DEBUGGER input *
GBLPAT=MACSUB:$DTI:52124 ; and output to printer port*
//
```

\* These three lines will be absent if you entered the second command line (with the switch /DEB).

2. The ODL (overlay descriptor) file:

```
;MERGED ODL FILE CREATED ON 12-JUL-83 AT 09:35:32
@C81IVP,SKL
SCOBJ$: .FCTR C81IVP.OBJ
@LB:[1,1]RMSRLX,ODL
; .NAME RMS$TR
RMSTR$: .FCTR RMS$TR-RMSALL
RMS$: .FCTR RMSROT
SCLIB$: .FCTR LB:[1,1]C81LIB/LB
@LB:[1,1]C81DBG,ODL
OBJRT$: .FCTR $DROOT-SCOBJ$-SCLIB$-RMS$
; .ROOT OBJRT$,RMSTR$, $DALL
.END
```

3. The INS (P/OS installation) file:

```
! COBOL-81 INS FILE CREATED : 12-JUL-83 AT 09:35:32
NAME "COBOL-81 SAMPLE"
FILE C81IVP.TSK/DELETE
INSTALL [ZZSYS]C81LIB.TSK/LIBRARY
INSTALL [ZZSYS]CGLEIS.TSK/LIBRARY
INSTALL C81IVP.TSK/TASK
RUN C81IVP
```

The above CMD and ODL files automatically specify certain minimum conditions for the P/OS environment.

## APPLICATION DEVELOPMENT EXAMPLE

### B.4 BUILDING THE SAMPLE APPLICATION

The PRO/Tool Kit LINK utility links the various object modules produced by the compiler, as well as object modules and symbolic definitions from other sources, into a task that can run on the Professional. For this sample application, use the command file C81IVP.CMD and the overlay description language file C81IVP.ODL, created in the preceding steps, as input to LINK. You need only invoke LINK and give it the name of the command file.

To invoke LINK, enter the following command line:

```
$ LINK @C81IVP
```

The result of this operation is the creation of the files C81IVP.TSK and C81IVP.STB. The STB file is created only if you have requested use of the symbolic debugger.

### B.5 CREATING AN APPLICATION DIRECTORY ON THE PROFESSIONAL

Before you can test the application, you must create a directory that will receive the files. This directory must have the same name as the application installation file. For the sample application, the installation file is named C81IVP.INS, so you must create the directory [C81IVP]. You should also make [C81IVP] your default directory as follows:

```
$ CREATE/DIR DW1:[C81IVP]
```

```
$ SET DEFAULT DW1:[C81IVP]
```

The application should now be moved to the newly created directory:

```
$ COPY DW1:[COBOL]C81IVP.TSK *.*
```

```
$ COPY DW1:[COBOL]C81IVP.STB *.*
```

```
$ COPY DW1:[COBOL]C81IVP.INS *.*
```

### B.6 INSTALLING THE SAMPLE APPLICATION

Install the sample application with the Fast Install application that was supplied with P/OS. Fast Install requires only that you supply the name of the directory in which the application resides ([C81IVP], in this case) and an application group in which to place the application.

Also, make sure that [C81IVP] is your default directory, so that the debugger can find C81IVP.STB.

## APPLICATION DEVELOPMENT EXAMPLE

### B.7 EXECUTING THE SAMPLE APPLICATION

You can now select the sample application, either from the Main Menu or from an Additional Applications menu, depending on where you installed it. The default name of the sample application is "COBOL-81 SAMPLE Application." If all has gone well, you should get the debugger prompt. At this point you can enter PROCEED to allow the application to continue. After running the application, you may want to step through it with the debugger. Refer to the COBOL-81 RSX-11M/M-PLUS User's Guide for information on the use of the debugger and debugger commands. You can also type "HELP" to the debugger prompt for online help information. You may want to print the listing created at the compilation phase and refer to it while you are stepping through with the debugger.





### READER'S COMMENTS

**NOTE:** This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

---

---

---

---

---

---

---

---

Did you find errors in this manual? If so, specify the error and the page number.

---

---

---

---

---

---

---

---

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) \_\_\_\_\_

Name \_\_\_\_\_ Date \_\_\_\_\_

Organization \_\_\_\_\_

Street \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_  
or Country

— — Do Not Tear - Fold Here and Tape — — — — —

**digital**



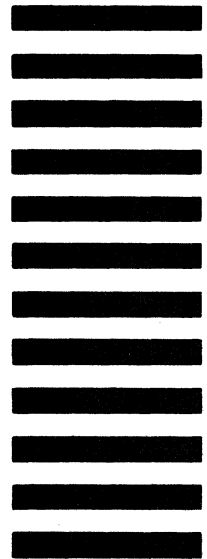
No Postage  
Necessary  
if Mailed in the  
United States

**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

BSSG PUBLICATIONS ZK1-3/J35  
DIGITAL EQUIPMENT CORPORATION  
110 SPIT BROOK ROAD  
NASHUA, NEW HAMPSHIRE 03061



— — Do Not Tear - Fold Here — — — — —